

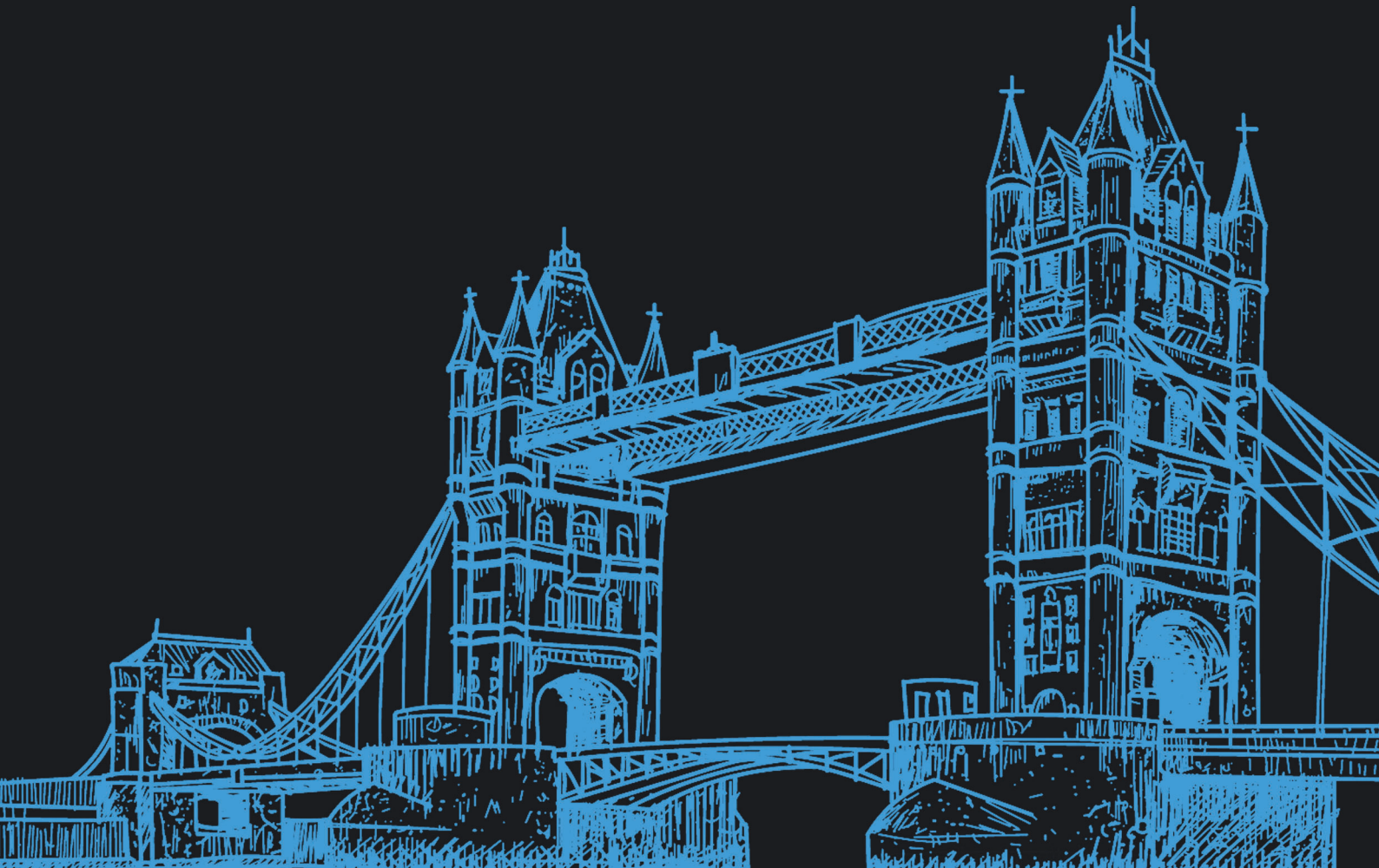
1 “Hello World”

2 English Language Skills

3 .forEach(programmer)

4

5



THE MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

KYIV NATIONAL UNIVERSITY
OF TECHNOLOGIES AND DESIGN

Inna Makhovych

Hello, World!
English Language Skills for Programmers

The educational textbook

Recommended by the Academic Council
of Kyiv National University of Technologies and Design
for the Bachelor's degree students

Kyiv
2025

UDC 811.111:37.091.315.7(075.8)
M36

Reviewers:

N. O. Aristova

Doctor of Science (Pedagogy), Professor, Head of the Department of International
Relations and Scientific Cooperation,
Institute of Pedagogy of the National Academy of Educational Sciences of Ukraine

Ye. P. Isakova

PhD in Philology, Associate Professor, Head of the Department of Philology and
Translation,

Kyiv National University of Technologies and Design

O. I. Makhovych

PhD in Technical Sciences, Assistant Professor, Department of Network and
Internet Technologies,

Taras Shevchenko National University of Kyiv

Recommended by the Academic Council
of Kyiv National University of Technologies and Design
as an educational textbook for the Master's degree students
(protocol No 3 from 22nd October, 2025)

Makhovych, Inna

M36 Hello, world! English language skills for programmers : an academic
textbook / I. Makhovych. Kyiv : KNUTD, 2025. 256 p. Text English
ISBN 978-617-7763-59-7

The textbook “*Hello, World! English Language Skills for Programmers*”
is intended for undergraduate students majoring in Software Engineering and
related IT fields. It develops learners’ professional communicative competence
in English through authentic materials, interactive activities, and real-world IT
contexts. The book consists of eight thematic units combining linguistic and
professional content, and features gamification elements and digital tools to
enhance motivation and engagement.

UDC 811.111:37.091.315.7(075.8)

ISBN 978-617-7763-59-7

© I. Makhovych, 2025
© KNUTD, 2025

CONTENTS

CONTENTS.....	3
INTRODUCTION	5
Unit 1 Product Development Overview	7
Introduction to software engineering practices	9
Different approaches to product development: Agile, Waterfall	16
Pros and cons of various development methods	25
Unit 1 Vocabulary Revision.....	35
Unit 2 Jobs and Responsibilities	39
Overview of software engineering roles: frontend, backend, full-stack, and QA	41
Detailed responsibilities of different software engineering positions	51
Best practices and advice from experienced software engineers.....	58
Unit 2 Vocabulary Revision.....	66
Unit 3 Programming Languages	69
Introduction to Programming Languages: Syntax, Semantics, and Paradigms...	71
Comparing and contrasting popular programming languages: Python, Java, JavaScript, C++, C#	82
Data base	94
Unit 3 Vocabulary Revision.....	101
Unit 4 Software Development Process Overview.....	104
Introduction to various software development methodologies: Test-Driven Development (TDD), Behaviour-Driven Development (BDD)	105
Explanation of version control systems: Git (local repositories), GitHub (cloud-based), and GitLab	113
Overview of DevOps practices, including Continuous Integration and Continuous Deployment (CI/CD)	119
Unit 4 Vocabulary Revision.....	126
Unit 5 Product Management Tools	128

Detailed look at the software development lifecycle	129
Common project management challenges and solutions, key steps in drawing up a Gantt Chart.....	138
Overview of project management software: Jira, Trello, Asana.....	147
Unit 5 Vocabulary Revision	153
Unit 6 Communication Tools	155
Overview of team chat tools: Slack, Microsoft Teams	156
Effective of use professional chat apps	162
Best Practices for Team Conversations via Email and Chat.....	172
Unit 6 Vocabulary Revision	179
Unit 7 Presentations	182
Presentation techniques for software engineers	183
How to describe and interpret visuals like graphs, charts, and diagrams	189
Using presentation software tools efficiently (e.g., PowerPoint, Keynote, Google Slides, Prezi, Canva)	198
Unit 7 Vocabulary Revision	205
Unit 8 Software Quality Assurance	207
Software quality attributes.....	208
Software Testing.....	214
Web Application Testing.....	221
Unit 8 Vocabulary Revision	228
List of Audio Recording Links for the Texts (Units 1–8).....	230
Alphabetical Glossary of Key Terms (Units 1–8)	239
REFERENCES.....	252

INTRODUCTION

The rapid development of information technologies has made English the dominant language of global communication, scientific research, and professional collaboration in the IT industry. For programmers, software engineers, and other IT professionals, English proficiency is not only a means of communication but also an essential professional skill that directly affects career opportunities, teamwork efficiency, and access to up-to-date knowledge resources. Therefore, mastering English for professional purposes is an integral component of future IT specialists' training at higher education institutions.

The textbook “Hello, World! English Language Skills for Programmers” has been designed to meet the educational needs of undergraduate students majoring in *Software Engineering* and related technical fields who study the course “Foreign Language for Professional Purposes.” It aims to develop learners' communicative competence in the professional context of information technologies and to foster confidence in using English for academic, social, and workplace communication.

The structure of the textbook reflects a balance between linguistic, professional, and communicative components. It comprises eight units, each focusing on relevant topics from the IT field, such as teamwork and communication in software development, project management tools, Agile and DevOps practices, cybersecurity, and emerging technologies. Every unit integrates tasks that develop reading, listening, speaking, and writing skills through authentic materials and real-life scenarios. Students are encouraged to analyse, discuss, and apply professional terminology in meaningful contexts, simulating typical workplace situations.

In accordance with the communicative and competence-based approaches to language teaching, the materials emphasize interaction, problem-solving, and task performance rather than rote memorization. The textbook combines individual and collaborative activities to develop both language proficiency and soft skills that are crucial for IT professionals: teamwork, critical thinking, adaptability, creativity, and leadership.

A distinctive feature of this textbook is the systematic implementation of gamification and digital tools that enhance motivation and engagement. Each unit contains interactive online exercises on platforms such as Quizlet, Kahoot!, and Mentimeter, which can be used both in the classroom and for independent study. Students can practice new vocabulary through *Flashcards*, *Learn*, and *Match* modes, compete in *Quizlet Live* games, and reinforce the material through team-based activities such as *Alias* and *Describer & Guesser*. These activities promote active

learning and create a dynamic environment where students learn through collaboration, competition, and creativity.

The integration of QR codes and hyperlinks provides seamless access to multimedia content, including video materials, listening tasks, online quizzes, and supplementary reading. This design supports the principles of blended and digital learning, enabling flexibility and personalization of the educational process. Each unit contains clear learning objectives, pre-reading and pre-listening tasks, post-activity reflection, and assignments for self-assessment, encouraging students to monitor their progress and take responsibility for their learning.

The textbook also places particular emphasis on authenticity and relevance. The texts and tasks are based on up-to-date materials from real-world IT contexts – documentation, technical blogs, software development platforms, and professional communication channels. This ensures that students are exposed to genuine language used by professionals and can gradually adapt to the discourse conventions of their future workplace.

Furthermore, the material encourages interdisciplinary connections between language learning and students' major subjects. Through content-based learning, students simultaneously acquire linguistic knowledge and deepen their understanding of professional concepts and tools. This integration reflects the modern educational paradigm that views language not as an isolated subject, but as a medium for developing professional identity and intercultural competence.

The authors' approach also considers the importance of learner autonomy and differentiation. The variety of tasks allows students to choose their pace and learning strategies, fostering independence and personalized learning trajectories. The inclusion of creative assignments, such as designing IT-related presentations, discussing case studies, and simulating professional dialogues, helps learners apply language in practical and meaningful ways.

Overall, “Hello, World! English Language Skills for Programmers” is designed to support the professional and linguistic development of students in technical specialties. It reflects modern pedagogical principles – communicative orientation, professional relevance, gamified learning, and digital integration – providing a comprehensive and engaging resource for both classroom and self-study use.

The textbook can be effectively implemented in courses of English for Specific Purposes (ESP), project-based learning environments, and interdisciplinary educational programs. It is expected to serve as a practical guide for students and educators who aim to bridge the gap between language learning and professional communication in the rapidly evolving world of information technology.

1

UNIT

Product Development Overview



This unit introduces students to fundamental concepts and vocabulary related to modern software development processes. Learners will explore different approaches such as Agile and Waterfall, and analyze their strengths and weaknesses. Through reading, listening, and grammar-focused activities, students will develop key language skills to describe and compare software development methodologies.

Unit overview

1.1 Introduction to software engineering practices.

Lesson outcome: Learners can use key terms related to software engineering and correctly apply dependent prepositions with verbs in context.

Skills practised:

Reading comprehension

Vocabulary: key concepts in software engineering

Grammar: Verb + preposition (dependent prepositions)

1.2 Different approaches to product development: Agile, Waterfall

Lesson outcome: Learners can describe Agile and Waterfall development models and apply dependent prepositions after adjectives when expressing opinions.

Skills practised:

Reading comprehension

Vocabulary: software lifecycle, workflows

Grammar: Adjective + preposition (dependent prepositions)

1.3 Pros and cons of various development methods: Agile, Waterfall

Lesson outcome: Learners can discuss advantages and disadvantages of Agile and Waterfall, and distinguish time expressions such as during, for, while in speech.

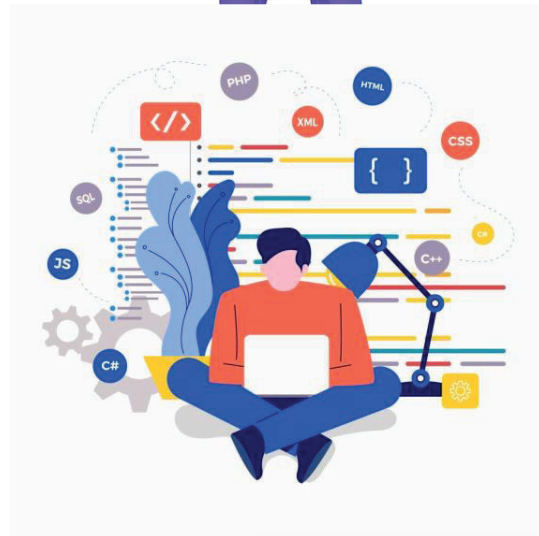
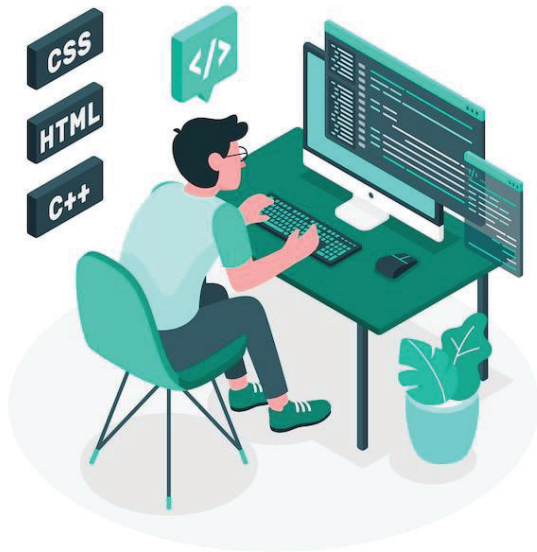
Skills practised:

Listening for gist and detail

Vocabulary: advantages/disadvantages

Grammar: During, for, while

What Is Software?




- *Can you name any types of software?*
- *Who creates software?*
- *What kind of people work on software development?*

 **Watch the video [from [0:00–2:07](#)].**

What is the **main message** of the video? Choose the correct answer:

- a) Computers are smarter than humans
- b) Programming is a creative and logical activity
- c) Programming is boring but useful



 **Listen to the video [\[0:00–2:07\]](#) and fill in the missing words in each quote.**

1. (0:16–0:20) “They can do only precisely what they’re _____.”
2. (0:26–0:29) “The code of computer programmers is _____.”
3. (0:46–0:51) “Give me a real high. It must be the way _____ feel.”
4. (1:19–1:24) “It was love at first _____. I could sit all night with that machine.”
5. (2:00–2:05) “Speak to the machine in a way that will make it do the little ____ you want it to do.”

Lesson 1

Introduction to software engineering practices

Lead-in 1 Discuss with your partner:

Why do you think it's important to follow a clear plan when creating software?

Reading 2 Read the text

1.1.

WHAT IS SOFTWARE ENGINEERING?

Software systems are highly sophisticated intellectual creations. To ensure that such systems fulfil user needs, stay within budget, and are delivered on schedule, a structured and reliable approach is essential. This is precisely what gave rise to the concept of **software engineering**, a term first introduced at a NATO conference in 1968. The aim was to **advocate** for applying engineering principles to the development of software. Since that time, the discipline has evolved rapidly, making **remarkable** strides in both theory and application.

At its core, **software engineering** focuses on research, education, and hands-on application of processes and methods designed to improve productivity (P) and quality (Q) while minimizing cost (C) and delivery time (T) – collectively known as PQCT. To **accomplish** these outcomes, practitioners rely on consistent and measurable techniques that optimize performance.

Software development involves a sequence of activities that convert an initial concept into a functioning product. These include **software specification**, software design, implementation, testing, deployment, and **maintenance**. The specification stage identifies what users and stakeholders expect from the software, documenting functional and non-functional requirements. **Software design** then outlines the solution, describing how the system will meet those needs. This phase includes planning the **software architecture**, which defines the core components and the way they interact. It also includes interface design and the high-level logic used by the system.

Throughout development, **software quality assurance (QA)** plays a critical role. QA ensures that each phase of the process meets predefined standards through activities such as software requirements document (SRD) and software design document (SDD), are

produced and conform to quality standards; and the software system will fulfill the requirements. These steps help guarantee that the end product not only works but does so **consistently**, and according to user expectations.

Software now plays a pivotal role in nearly every aspect of daily life, which introduces significant ethical and social considerations. **In this regard**, software engineers must be aware of the broader consequences of their work. The “Software Engineering Code of Ethics and Professional Practice” was established to guide behaviour in areas such as client confidentiality, intellectual property, and public safety. Engineers are expected to **commit** to high ethical standards and demonstrate this **commitment** both professionally and personally.

Above all, integrity is central. Without **integrity**, even the most technically perfect systems can fail to earn trust or serve their intended purpose. Software engineers must not only be skilled coders but also responsible professionals, fully aware of the impact their decisions can have. ↻

Software Engineering Code of Ethics and Professional Practice (Short Version)

PREAMBLE

Software engineers shall **commit** themselves to making the **analysis**, specification, design, development, testing and **maintenance** of software a beneficial and respected profession. **In accordance with** their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

1. PUBLIC – Software engineers shall act **consistently** with the public interest.
2. CLIENT AND EMPLOYER – Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
3. PRODUCT – Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT – Software engineers shall maintain **integrity** and independence in their professional judgment.
5. MANAGEMENT – Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. PROFESSION – Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. COLLEAGUES – Software engineers shall be fair to and supportive of their colleagues.
8. SELF – Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Figure 1 ACM Code of Ethics and Professional Conduct <https://www.acm.org/code-of-ethics/software-engineering-code#:~:text=The%20Software%20Engineering%20Code%20of%20Ethics%20and%20Professional%20Practice>

Vocabulary 3 Match the highlighted words in the text with their definitions

1. software design	a. to support; to be in favour of
2. accomplish	b. to do, make happen, succeed in, carry through
3. consistently	c. unusual, extraordinary, worthy of attention
4. remarkable	d. where customers and engineers define the software that is to be produced and the constraints on its operation
5. software specification	e. produces a software solution to realize the software requirements
6. advocate	f. the overall software structure that depicts the major system components and how they relate, interface, and interact with each other
7. analysis	g. a discipline focused on research, education, and practice of engineering processes, methods, and techniques to significantly increase software productivity and quality while reducing costs and time to market
8. software engineering	h. honesty, high moral standards; an unimpaired condition, completeness, soundness
9. software quality assurance	i. in connection with the point previously mentioned
10. in this regard	j. to dedicate oneself to a cause or activity with strong intention and responsibility
11. commit	k. a promise or pledge to do something
12. commitment	l. a detailed examination of the elements or structure of something
13. software architecture	m. the work that is done to keep something in good condition
14. maintenance	n. continually; regularly
15. integrity	o. ensure that the development activities are carried out correctly

Vocabulary

4 🖱️ Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3>



Close Reading

5. Read the text again carefully. Then read the following statements and determine whether they are True or False. For each False statement, explain why it is incorrect based on the information in the text.

1. Software engineering focuses only on developing new methods for faster software production. (T / F)
2. Software quality assurance (QA) activities are conducted separately from the development process. (T / F)
3. The main goal of software engineering is to improve software productivity, quality, cost, and time to market (PQCT). (T / F)
4. Software engineers are expected to consider ethical responsibilities in their work, including confidentiality and intellectual property protection. (T / F)
5. The architecture of a software system refers to the major system components and how they interact. (T / F)

6. Answer the following questions:

1. What is the main objective of software engineering, as defined in the text?
2. What does PQCT stand for in the context of software engineering, and why is it important?
3. What are the key activities involved in the software development process?
4. How does software specification differ from software design?
5. What role does software quality assurance (QA) play in the development process?
6. What is the purpose of software testing, and when does it occur during the development process?
7. According to the text, why is it essential for software engineers to consider social and ethical responsibilities?
8. What is the function of the ACM/IEEE-CS Software Engineering Code of Ethics?
9. List the Eight Principles outlined in the Software Engineering Code of Ethics.
10. How does the text suggest that software engineers maintain professional integrity and independence?
11. What is the significance of lifelong learning in the context of software engineering ethics?
12. Why is the balance between aspirations and details important in the Software Engineering Code of Ethics

Ethical Dilemma Discussion

7. Applying the Software Engineering Code of Ethics in Real-Life Situations

Consider the following ethical scenario. Then discuss the questions below using principles from the Software Engineering Code of Ethics.








Scenario:

A college student has an important job interview coming up, but his laptop has stopped working. He asks to borrow his girlfriend's laptop for the weekend to prepare. However, the laptop belongs to her employer and is used for work purposes. She is unsure whether it's appropriate to let him use it.

Discussion Questions:

1. What would you advise the girlfriend to do in this situation? Why?
2. Which ethical principles from the Software Engineering Code of Ethics are relevant here (e.g., public interest, client confidentiality, professional responsibility)?
3. Have you ever faced — or can you imagine — a similar ethical dilemma in the workplace or in your studies? How would you handle it?

Grammar for Revision Verb + prepositions: Dependent prepositions

<p> to</p> <p>belong to somebody refer to sth listen to sb look forward to sth say sth to sb talk to sb write to sb</p>	<p> for</p> <p>ask for sth apply for sth look for sth pay for sth wait for sth or sb</p>	<p> at</p> <p>arrive at a place or event look at sth or sb laugh at sth or sb shout at sth or sb smile at sth or sb</p>
<p> in</p> <p>arrive in a town, city, country, etc. believe in sth invest in sth succeed in sth or doing sth result in sth</p>	<p> on</p> <p>insist on sth depend on sth spend money on sth work on a task, a project rely on sb</p>	<p> of</p> <p>remind of sb or sth think of sth or doing sth approve of sb/sth consist of sth</p>
<p> with</p> <p>argue with sb agree with sb compare sth with sth else spend time with sb stay with sb</p>	<p>verbs & prepositions</p>	<p> about</p> <p>argue about sth complain about sth dream about sth talk about sth think about sb or sth</p>

8. Match the verb with the correct preposition

Example: apply → for

Verb	Preposition options		Verb	Preposition options	
apply	at / for / to	a job	rely	on / in / at	your teammates
depend	with / about / on	the result	approve	of / for / at	the plan
believe	in / on / about	yourself	belong	with / to / from	a team
focus	on / in / of	the task	talk	at / with / to	the manager
deal	for / with / by	a problem	listen	to / with / at	the podcast
work	on / over / at	a project	write	at / for / to	your professor
complain	at / about / of	the noise	consist	in / of / from	three modules
wait	to / at / for	the bus	result	from / in / of	an error
invest	with / to / in	new technology	agree	with / on / in	your colleague
refer	on / to / about	a document	laugh	at / with / on	the joke

9. Complete the sentences with the correct preposition

(Prepositions: **on, for, with, in, to, about, at, from**)

1. I'm currently working _____ a project related to mobile app development.
2. Please apply _____ the internship before Friday.
3. This task depends _____ accurate time estimation.
4. Our QA engineer complained _____ the inconsistent documentation.
5. We believe _____ open-source collaboration.
6. Let's focus _____ user-friendly interface design.

10. Error correction

Each sentence contains one incorrect verb-preposition combination. Correct them.

1. We applied to the software engineering internship at Google.
2. The manager dealt about the client issue personally.
3. They focused in UI instead of UX.
4. She complained with her slow laptop again.
5. He contributed in the final product release.
6. I'm waiting at your reply.
7. The report consists from several sections.

11. Complete the mini-dialogues using the correct preposition

(Prepositions: **on, for, to, with, of, about, at, in, from**)

1. **A:** Did you apply ____ the UX designer role?
B: Yes, I sent my CV yesterday.
2. **A:** What are you currently working ____?
B: A cross-platform mobile app.
3. **A:** She complained _____ the outdated documentation.
B: Yeah, we need to update the onboarding files.
4. **A:** I always rely ____ him for testing feedback.
B: He's very thorough, that's true.
5. **A:** We succeeded ____ reducing the app's load time by 30%.
B: That's impressive!
6. **A:** Did you agree ____ the team's decision?
B: Not entirely — I had some doubts.
7. **A:** He referred me ____ the documentation for the API.
B: Great, I'll check it out!

Extra Online Practice

Vocabulary:

Quizlet – Unit 1.1 Flashcards & Learn

Grammar:

Verbs + Prepositions (Dependent prepositions)

Exercises 1–3 on Test-English.com



- ☒ Recommended activity before starting Lesson 2:

Play Quizlet Live (INDIVIDUAL MODE, 3 TIMES) using vocabulary from Unit 1.1:

 <https://quizlet.com/ua/943274861/unit-11-flash-cards/?i=j03j6&x=1jqt>

Lesson 2

Different approaches to product development: Agile, Waterfall

Lead-in 1 Take a class poll:

Do you prefer planning everything in advance or adapting as you go?

Would you rather work on small parts of a project step-by-step, or build it all at once?

Today we'll compare two major approaches to software development – Agile and Waterfall – and see how they reflect these styles.

Pre-reading 2

Match the highlighted words in the text below with their definitions.

- | | |
|--------------------|--|
| 1. exhibit | a. continuing for a long period of time |
| 2. rapid expansion | b. the process of improving the quality, amount, or strength of something |
| 3. sequence | c. to think, believe; to consider, have an opinion |
| 4. requirement | d. the order in which things happen or should happen |
| 5. resemble | e. being planned or in progress |
| 6. deem | f. to be like or similar to |
| 7. in the pipeline | g. something that is necessary, that must be done |
| 8. embedded system | h. a computer system that is part of a larger machine and controls its operation |
| 9. capabilities | i. project review points used to assess progress and performance |
| 10. long-lasting | j. to produce or provide something official |
| 11. enhancement | k. to make a product available for the public to buy, often with a celebration |
| 12. issue (v) | l. product acceptance is growing and investors become very interested |
| 13. rigorously | m. to show, display, present, or demonstrate |
| 14. milestones | n. in a careful way, ensuring every part is correct or safe |
| 15. release (v) | o. the ability to do something |

3 Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3><https://quizlet.com/942958429/match?funnelUUID=6dbbf04f-81e1-4138-ad68-c3d9d93c23b8>



Reading 4 Read the text quickly and choose the correct answer (a, b, or c) for each question below.

 **1.2.**

SOFTWARE PROCESS

1. What is the main purpose of introducing a software process?
 - a) To reduce software size
 - b) To structure the development process in phases
 - c) To avoid writing code manually
2. What is a major benefit of the waterfall process?
 - a) High flexibility
 - b) Easy planning and scheduling
 - c) Early user involvement
3. Why might the waterfall process not work well today?
 - a) It is too expensive
 - b) Users change passwords too often
 - c) It does not easily support changes during development
4. What do agile processes promote?
 - a) Fixed project plans
 - b) Collaboration and adaptability
 - c) Equipment control systems
5. How is building a theme park similar to software development?
 - a) Both require expensive machinery
 - b) Both involve phased development and planning
 - c) Both rely on the same design tools

Advances in computer science, especially non-numerical computation and relational database systems in the 1960s, led to a **rapid expansion** of computer applications in the business sector. The ad hoc development approaches that existed then could not satisfy the needs of business organizations. The notion of a software development process or software process was introduced. It is defined as follows:

*A **software process** is a series of phases of activities performed to construct a software system. Each phase produces some artifacts that are the input to other phases. Each phase has a set of entrance criteria and a set of exit criteria.*

For example, the *waterfall process* **exhibits** a straight **sequence** of development activities that begin with **requirements** analysis, followed by software design, implementation, testing, deployment, and maintenance. It is called a waterfall process because its straight sequence of

activities **resembles** a waterfall when the activities are shown vertically one after another. The requirements phase produces the requirements specification, the design

phase produces the design, and so on. Frequently, the entrance criteria specify the software artifacts that must be available. For example, the entrance criteria for the software design phase are that the requirements specification must have been produced and reviewed. The exit criteria specify the software artifacts that must be produced. For example, the exit criteria for software design are that the architectural design and high-level design must have been produced and reviewed.

The waterfall process has been used since the 1970s, and some organizations still use it. The straight sequence of phases of the waterfall process simplifies project planning, project scheduling, and project status reporting. Because of these, it is deemed a predictable process. Moreover, the straight sequence of functional activities allows function-oriented project organization. In such an organization, the functional teams are specialized in different functional areas such as requirements analysis, design, implementation, and testing. Projects are carried out by the functional teams in the pipeline manner. Finally, the waterfall process is appropriate for developing large, complex, long-lasting, embedded systems. Examples include mail-sorting and routing systems, process control systems, and many other types of computerized equipment. Such systems need to respond to numerous hardware-generated events, process huge amounts of incoming data, and control the behaviors of hardware devices. Usually, the capabilities or requirements of such systems are jointly defined by the equipment manufacturer and the customer. In many cases, the system to be developed is merely a major enhancement of the functionality, performance, and security of an existing system. The vendor has experience and good knowledge of the application and what the customer wants; hence, major changes to the requirements are rare. Once the purchase order is issued, the customer does not have access to the equipment until the acceptance testing stage, although the customer participates in reviews and prototype demonstrations. The phased approach allows each phase to be performed rigorously to ensure that the system runs reliably and satisfies performance and timing constraints.


The waterfall process has a number of drawbacks. First, the strict sequence of phases and related **milestones** makes it difficult to respond to requirements change. This is because requirements change requires considerable rework. Unfortunately, requirements change is the way of life, due to business competition, new regulations or standards, or advances in technology. For many applications, the long development duration is unacceptable because the requirements were identified long ago and business needs have changed dramatically. In addition, the users cannot experiment with the system to provide feedback until it is **released**. Experiences show that early user feedback helps in detecting misconception of business needs and problems in user interface design. Finally, the customer cannot reap the benefits of the new system during the long development period.

Recent trends in software engineering focus on agile processes, design patterns, and **test-driven development (TDD)**. Agile processes stress collaboration, adaptability, rapid deployment of small software increments, and close cooperation with customers and users. **Design patterns** provide proven solutions to common design challenges, fostering software reuse and improving team communication. By encoding design principles, patterns also help less experienced engineers produce high-quality software. TDD promotes creating test scripts before coding, ensuring that software is testable and allowing for frequent, immediate testing.

To understand the importance of processes and methodologies, consider the analogy of building an amusement park. The **development process** for a theme park includes various phases such as planning, analysis, design, financing, construction, and equipment installation. Each phase has specific activities—such as feasibility studies during planning or site investigation and park layout design during analysis and design—that must be performed to complete the project. However, simply knowing the overall process is not enough. The team must also understand how to execute each activity, much like a software team would need a methodology to guide them through coding, testing, and deployment.

In the case of an amusement park, the conventional approach is to finalize the master design early on. Once completed, it becomes difficult to modify without incurring significant costs. Similarly, in traditional software development, altering a finalized design late in the process can be expensive.

Agile processes aim to solve this problem by allowing the project requirements to evolve throughout the development cycle. For example, a theme park can be developed incrementally, with different clusters of attractions built and deployed one at a time. This approach enables early feedback, allowing changes to be made to the requirements, budget, and schedule before the entire park is completed. The same applies to software development: **agile methodologies** facilitate rapid deployment of small increments, allowing stakeholders to provide input and enabling the system to evolve as needed.

This flexibility makes agile processes ideal for adapting to changing conditions. By focusing on teamwork, iterative development, and continuous feedback, agile methodologies enable projects to deliver results earlier, make adjustments more easily, and maintain high quality throughout the development process. 

Close Reading

5. Read the text carefully again. Then answer the following questions in your own words. Use full sentences:

- 1) What historical or technical developments made ad hoc approaches to software development insufficient?
- 2) How is a **software process** structured, and what do “artefacts” refer to in this context?
- 3) What do the **entrance and exit criteria** ensure during the development phases?
- 4) What kind of organizational structure does the waterfall process support, and how does this impact the project workflow?

- 5) Give an example of a real-world system that would be suitable for development using the **waterfall model**, and explain why.
- 6) According to the text, why can long development durations be problematic in modern business contexts?
- 7) How do **design patterns** improve communication within software teams?
- 8) What key practice defines **test-driven development (TDD)**, and how does it benefit software testing?
- 9) How does the **amusement park analogy** help explain the difference between traditional and agile development?
- 10) In what specific ways does agile methodology help manage unexpected changes during development?

Compare and Decide: Which Approach Would You Choose?

6. Task: Imagine you are part of a software development team that is about to start a new project. *Discuss or write a short paragraph (80–100 words) answering the following:*

1. Would you choose Agile or Waterfall for your project?
2. What kind of project are you working on (e.g., mobile app, control system, e-commerce website)?
3. Give two reasons for your choice using information from the text.
4. Mention at least one potential challenge you might face with that approach.

SPEAKING (FOR PAIR/GROUP WORK):

7. Work with a partner. One of you is in favour of Agile, the other prefers Waterfall. Try to convince each other why your approach is better for a software development team."

Grammar for Revision Adjective & prepositions: Dependent prepositions

<p>to</p> <p>Addicted to sb/sth Close to somebody Different to sb/sth Kind to sb Married to sb Rude to sb Similar to sth/sb</p>	<p>of</p> <p>Afraid of sth or sb Capable of sth Fond of sth or sb Proud of sb or sth Tired of sb or sth</p>	<p>with</p> <p>Angry with sb !!! Bored with sth Fed up with sth/sb Obsessed with sb/sth Pleased with sth/sb</p>	<p>in</p> <p>Interested in sth or sb, or in doing sth</p>
<p>about</p> <p>Angry about sth !!! Excited about sth Sorry about sth Worried about sth or sb</p>	<p>for</p> <p>Bad for sth or sb Famous for sth Good for sth or sb Sorry for sth/sb, or for doing sth</p>	<p>at</p> <p>Good at sth or doing sth Bad at sth or doing sth Angry at sb</p>	<p>on</p> <p>Keen on sth or doing sth Hooked on sth</p>

8. Match the adjective with the correct preposition

Adjective	Preposition options	Adjective	Preposition options
afraid	of/at/on She's afraid ___ spiders.	keen	on/at/in She's keen ___ UX design.
angry	in/about/for He's angry ___ the delay.	kind	for/to/about They were kind ___ the intern.
angry	with/of/on I'm angry ___ him.	married	on/to/at She's married ___ a software engineer.
bad	on/at/in I'm bad ___ maths.	nice	to/for He was nice ___ the new colleague.
bored	of/with/at She's bored ___ her job.	proud	of/at/in I'm proud ___ our project.
good	in/on/at He's good ___ programming.	excited	for/to/about We're excited ___ our upcoming vacation
interested	at/of/in They're interested ___ AI.	responsible	for/of/at She's responsible ___ documentation.
famous	about/for This city is famous ___ its architecture.	rude	to/for/on He was rude ___ the client.
fed up	of/from/with I'm fed up ___ all these bugs.	tired	for/of/in I'm tired ___ debugging this code.
frightened	with/of/about He's frightened ___ failure.	worried	of/in/about They're worried ___ the deadline.

9. Complete the sentences with the correct preposition

(Prepositions: about, at, for, in, of, on, to, with)

1. She's really interested ____ cybersecurity and data privacy.
2. I'm tired ____ debugging the same issue again and again.
3. Are you married ____ someone in the tech industry?
4. He's afraid ____ presenting in front of large groups.
5. This tool is famous ____ its speed and flexibility.
6. They were angry ____ the delayed update.
7. I'm not very good ____ managing time under pressure.
8. He's obsessed ____ writing clean, readable code.
9. We were very pleased ____ the results of the test.
10. The manager was rude ____ the new intern.

10. Error correction

*Ten of the sentences below contain one incorrect preposition. **TWO ARE CORRECT.** Correct mistakes.*

1. She's afraid from failing the exam.
2. I'm bored of this task already.
3. He's proud for his team's work.
4. They were angry to the manager.
5. This program is famous about its simplicity.
6. Are you interested for machine learning?
7. He's kind to animals.
8. I'm not very good in solving algorithms.
9. She's excited on the new release.
10. He was rude with the technician.
11. She's tired of waiting for feedback.
12. We were pleased with the final results.

Speaking

11. Work in pairs. Take turns asking and answering the questions below. Try to use adjective + preposition combinations from the lesson.

Interview Your Partner

Use follow-up questions to keep the conversation going.

1. What are you currently **interested in** learning or doing?
2. Is there anything you're **afraid of** in your studies or career?
3. What kind of tasks are you **good at**? What about the ones you're bad at?
4. Have you ever been really **angry with** someone during a group project?
5. Are you **proud of** something you've recently achieved?
6. What are you **tired of** hearing or doing in everyday life?

Extra Online Practice

Vocabulary:

[Quizlet – Unit 1.2 Flashcards & Learn](#) 

Grammar:

Adjective + Prepositions (Dependent prepositions)

[Exercises 1–3 on Test-English.com](#) 

☒ Recommended activity before starting Lesson 3:

Play Quizlet Live (INDIVIDUAL MODE, 3

TIMES) using vocabulary from Unit 1.2:

 <https://quizlet.com/ua/942958429/unit-1-2-flash-cards/?i=j03j6&x=1jqt>



Lesson 3

Pros and cons of various development methods

Lead-in 1 Read the statements. Do you agree, disagree, or not sure? Discuss with a partner:

1. *“The best way to develop software is to follow a strict, step-by-step plan.”*
2. *“Clients should not be involved in software development after signing the contract.”*
3. *“Fast results are more important than perfect documentation.”*
4. *“Changing requirements during development is normal and expected.”*

Listening 2 Watch the video from 8:23 to 12:10. Choose the correct option.



Figure 2 Software Development Life Cycle: Explained [Source: https://youtu.be/SaCYkPD4_K0]

1. What does DevOps stand for?
 - a) Device Optimization Service
 - b) Development and Operations
 - c) Deployment Over Process Solutions
2. What does the **infinity loop** in DevOps represent?
 - a) Endless testing
 - b) Continuous team rotation
 - c) Development and operations working as one cycle
3. What's the key idea behind **CI/CD**?
 - a) Rare updates and big releases
 - b) Continuous testing, integration, and delivery
 - c) Manual deployment of new features
4. How many deployments does Netflix reportedly perform daily?
 - a) About 20
 - b) Around 300
 - c) Over 20,000
5. What helps DevOps teams deliver code quickly and often?
 - a) More meetings
 - b) High automation and collaboration
 - c) Extra testing phases

2. Complete the summary (use words from the video)

DevOps is a development approach that combines _____ and _____ into a single, collaborative team. It promotes a cultural shift and relies on practices like _____ and _____, which allow developers to integrate and release code more frequently. This leads to faster feedback, fewer errors, and the ability to deliver value to users in _____.

💡 4. Reflect and Compare

Work in pairs or small groups. Use the prompts below to discuss what you learned from the video.

1. What are the main **advantages** of DevOps compared to traditional development methods like Waterfall?
2. Do you think the **high number of deployments per day** (e.g. at Netflix) is a good idea? Why or why not?
3. Would DevOps work well for **student group projects**? Why or why not?
4. What skills or mindsets are needed to work in a **DevOps team**?

Pre-reading 5

Match the highlighted words in the text below with their definitions.

1. increment	a. A slight change made to something to make it fit, work better, or be more suitable.
2. incremental software	b. Extremely important; vital in resolving something.
3. crucial	c. A method of building software products in which a system is built piece-by-piece.
4. adjustment	d. A process that repeats a series of steps over and over until the desired outcome is obtained.
5. timeline	e. One of a series of increases, an enlargement, increase, or addition.
6. deliverable	f. Following a particular order, arranged serially.
7. iterative	g. A product, service, or result created by a project.
8. sequential	h. A graphic representation of the passage of time as a line.

Vocabulary

6 🖱️ Play the Quizlet Match game by following this link:

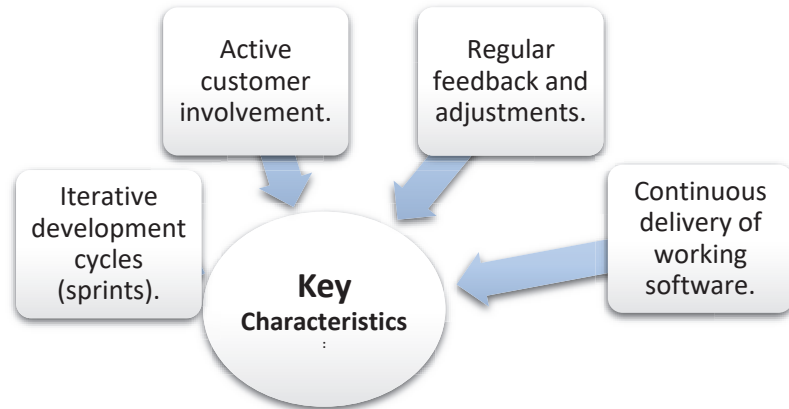
<https://quizlet.com/945632173/match?funnelUUID=a799eda1-e709-48de-aa07-33c0934dc4d5>



Pros and Cons of Agile and Waterfall Development Methods

In software development, choosing the right methodology is **crucial** to delivering a successful project. While there are numerous methodologies like Scrum, Kanban, and DevOps, we will focus on Pros and Cons of two widely used methods: **Agile** and **Waterfall**.

Agile is a flexible and iterative approach that allows changes and improvements throughout the software development process. It emphasizes collaboration, adaptability, and frequent delivery of smaller software **increments**.



Pros of Agile:

Improved Quality: Frequent testing and feedback cycles help catch errors early.

Faster Time to Market: Continuous delivery allows early versions of the product to be released and tested in real environments.

Customer Feedback: Regular interactions with the customer ensure that the product evolves according to their needs.

Flexibility: Changes can be made even in late stages of development.

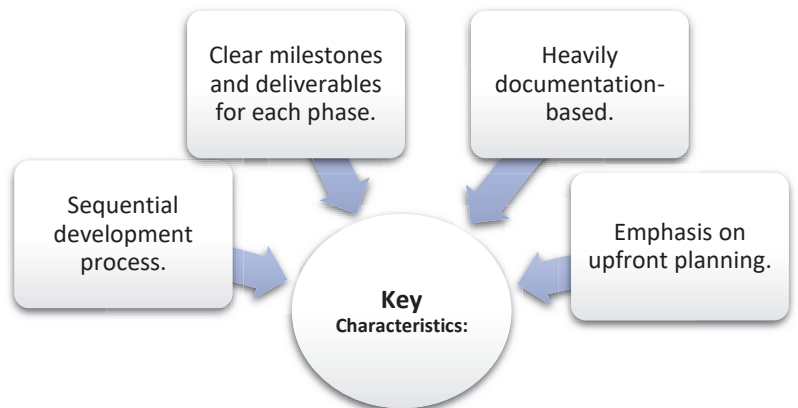
Cons of Agile:

Scope Creep: The evolving nature of Agile can lead to uncontrolled changes in project scope.

High Commitment from Stakeholders: Constant involvement from the customer and team members is required.

Less Predictability: Since changes can happen at any time, it may be difficult to predict timelines and costs.

Waterfall is a traditional, linear approach to software development, where each phase must be completed before moving on to the next. This method follows a structured sequence of development activities.



Pros of Waterfall:

Simplicity for Large Projects: Works well for projects where requirements are unlikely to change.

Predictability: Budget, timeline, and scope are usually well-defined at the start of the project.

Clear Structure: Each phase has defined objectives and deliverables, making project management easier.

Cons of Waterfall:

Long Development Time: It may take a long time to complete the project before delivering any working software.

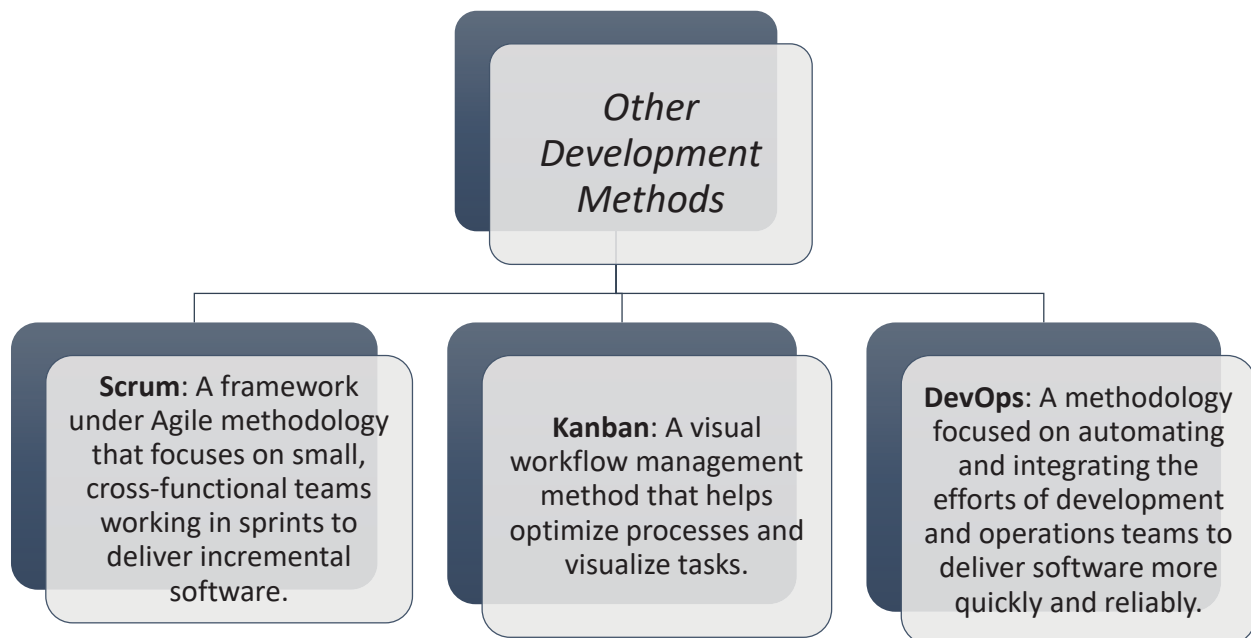
Customer Involvement: Limited interaction with the customer after the requirements phase.

Late Testing: Testing happens near the end of the development cycle, which means issues might not be identified until late.

Inflexibility: Once a phase is completed, it's difficult to go back and make changes.

Comparison Table

<i>Aspect</i>	<i>Agile</i>	<i>Waterfall</i>
Approach	Iterative, flexible	Sequential, rigid
Customer Involvement	Continuous, throughout development	Mostly during initial requirements phase
Change Management	Adaptable, changes are welcome	Difficult to manage changes
Timeline	Difficult to predict, evolves with project	Predetermined, fixed
Documentation	Less focus, evolves with development	Heavy upfront documentation
Risk	Lower, issues can be addressed early	Higher, issues identified late



Reading 7 Read the text quickly to understand the main ideas. Then decide whether the following statements are **True (T)** or **False (F)**. If a statement is false, **correct it** using information from the text.

1. Agile projects are usually delivered all at once after the final phase.
2. One of the benefits of Agile is the ability to adapt to changes during development.
3. The Waterfall method requires customer feedback at every stage.
4. Agile is based on iterative cycles and frequent delivery of working software.
5. Waterfall is more flexible than Agile when dealing with changes in project requirements.
6. Agile development often requires strong involvement from customers.
7. In Waterfall, testing is done continuously throughout the project.
8. Scope creep can be a risk in Agile due to evolving requirements.
9. Waterfall is suitable for projects with changing requirements.
10. Both Agile and Waterfall aim to deliver working software incrementally.

8. Discussion Questions:

Question 1: Which method would work better for a university team project? Why?

Useful phrases:

- *I think _____ would be better because...*
- *In our team, we usually...*
- *Agile is more suitable for student work since...*
- *Waterfall might be too rigid because...*
- *It's easier to manage deadlines / tasks / roles using...*

Question 2: Which disadvantages seem most serious to you?

Useful phrases:

- *The biggest issue with Agile / Waterfall is...*
- *I'm worried about...*
- *One serious drawback is...*
- *In my opinion, the hardest part of using this method would be...*
- *I find it difficult to... when using...*

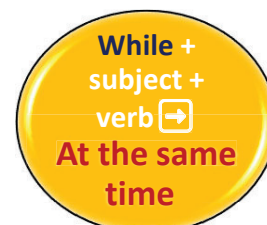
Grammar for Revision During, for, while



We discussed the bug during the morning meeting.
The system crashed during the update.
She took notes during the lecture.



He stayed in the office for the whole weekend.
They tested the software for two weeks.
We've been using Jira for several months.



While he was presenting, I took screenshots of the slides.
The team worked on UI improvements while QA was running tests.
While they were debugging, the client called again.

● We use **during** + **noun** to talk about **when** something happens. With during, we answer the question when.

● We use **for** + length of time to say how long something happens. With for, we answer the question how long.

● We use **while** + subject + verb to talk about two things that are happening at the same time.

The main difference between **during** and **while** is that we use **while** + **clause** (**subject** + **verb**), and we use **during** + noun.

The difference between **during** and **for** is that **during** refers to 'when' something happens and **for** refers to 'how long' something happens.

9. Choose the Correct Option (*during, for, or while*)

1. *We worked on the prototype _____ three days.*
2. *I stayed focused _____ the entire workshop.*
3. *He was fixing the login issue _____ she updated the database.*
4. *They tested the app _____ the demo session.*
5. *The tool crashed _____ I was exporting the report.*
6. *We didn't take breaks _____ the sprint.*
7. *She worked remotely _____ the summer.*

10. Error correction

*Ten of the sentences below contain one incorrect preposition. **TWO ARE CORRECT.** Correct mistakes.*

1. We worked on the design **during** three hours.
2. She stayed quiet **for** the entire interview.
3. The app crashed **for** the presentation.
4. I was checking emails **during** I was waiting.
5. They were coding **while** the meeting.
6. He took notes **during** the lecture.
7. We updated the system **while** the team was testing.
8. I joined the call **for** the discussion.
9. **During** we were preparing, the client sent a message.
10. I used GitHub **while** two months.
11. We had issues **for** the deployment.
12. The update ran **during** I was installing plugins.

Speaking

11. Work in pairs. Take turns asking and answering the questions below. Try to use **during, for, and while** in your answers. Use the suggested phrases to help you.

Questions to Ask Your Partner:

1. What do you usually do during a long Zoom meeting?
2. Have you ever worked on a project for more than a month? What was it about?
3. What do you like to do while you're traveling or commuting?
4. Tell me about a time when something went wrong during a group task.
5. How do you stay focused for long periods of time?
6. What do you usually listen to while working or studying?
7. What happened during your last team presentation?
8. Have you ever studied for an exam all night?

Extra Online Practice

Vocabulary:

[Quizlet – Unit 1.3 Flashcards & Learn](#)

Grammar:

During, for, while

[Exercises 1–3 on Test-English.com](#)

Useful Phrases:

I usually... during...

I worked on that for about...

While I was working, I...

It happened during our final sprint.


I often get distracted while...

We were discussing the task while...



Vocabulary Revision

Unit 1

 1. WORK IN PAIRS. TAKE TURNS PLAYING THE ROLES OF DESCRIBER AND GUESSER.

DESCRIBER:

- Choose a word or phrase from the vocabulary table.
- Describe it **without saying the word**.
- Use **synonyms, definitions, functions, examples, or context clues**.
- If your partner struggles, you may:
 - Give **another example**
 - Use **the word in a sentence (but skip the word)**
 - Tell **what it is not** ("It's not hardware, it's...")

GUESSER:

- Listen carefully and try to **guess the word**.
- You may ask for clarification or examples:
"IS IT A NOUN?" "IS IT SOMETHING WE USE IN CODING?"

 SWITCH ROLES AFTER EACH ROUND.

 EXAMPLE:

Describer: It's something we use to plan development tasks. It's visual and shows what stage each task is in.

Guesser: A Kanban board?

Describer: Correct!

 OPTIONAL EXTENSION:

- Keep score: 1 point for each correct guess
- Time challenge: Set a timer (e.g., 30 seconds per word)

Term	Definition
advocate	to support; to be in favour of
accomplish	to do, make happen, succeed in, carry through
remarkable	unusual, extraordinary, worthy of attention
software specification	where customers and engineers define the software that is to be produced and the constraints on its operation
software design	produces a software solution to realize the software requirements
software architecture	the overall software structure that depicts the major system components and how they relate, interface, and interact with each other
software engineering	a discipline focused on the research, education, and practice of engineering processes, methods, and techniques to increase software productivity and quality
software quality assurance	ensures that the development activities are carried out correctly
in this regard	in connection with the point previously mentioned
commit	to carry out or do
commitment	a promise or pledge to do something
analysis	a detailed examination of the elements or structure of something
specification	a detailed description of the design and materials used to make something
maintenance	the work that is done to keep something in good condition
consistently	continually; regularly
integrity	honesty, high moral standards; an unimpaired condition, completeness, soundness
exhibit	show, display, present, demonstrate
rapid expansion	product acceptance is growing and investors become very interested
sequence	the order in which things happen or should happen
requirement	something that is necessary, that must be done
resemble	to be like or similar to
deem	to think, believe; to consider, have an opinion
in the pipeline	being planned or in progress (refers to a sequence of steps or stages that data, tasks, or products go through to achieve a desired outcome)
embedded system	a computer system that is part of a larger machine and controls how that machine operates

capabilities	the qualities or abilities to perform a function
long-lasting	continuing for a long period of time
enhancement	the process of improving the quality, amount, or strength of sth.
issue (v)	to produce or provide something official
rigorously	in a careful way so that every part of something is looked at or considered to ensure it is correct or safe
milestones	formal project review points used to assess progress and performance
release (v)	to make a product available for the public to buy, often with a celebration; launch
increment	one of a series of increases; an enlargement, increase, addition
incremental software	a method of building software products in which a system is built piece-by-piece
crucial	extremely important; vital in resolving something
adjustment	a slight change made to something to make it fit, work better, or be more suitable, or the act of making such a change
timeline	a graphic representation of the passage of time as a line
deliverable	something that can be provided or achieved as a result of a process: A product, service, or result created by a project
iterative	a process that repeats a series of steps over and over until the desired outcome is obtained
sequential	following a particular order, arranged serially

2. PLAY THE QUIZLET LIVE GAME (TEAM MODE) to review the vocabulary learned in UNIT 1

<https://quizlet.com/ua/946430392/unit-1-11-12-13-flash-cards/?i=j03j6&x=1jqt>

3. Take the Unit 2 Vocabulary Quizlet Test to check your understanding of key terms from this unit.

<https://quizlet.com/946430392/test?answerTermSides=2&promptTermSides=4&questionCount=30&questionTypes=15&showImages=true>



4. THREE-SENTENCE CHALLENGE: GUESS THE WORD!

HOW TO PLAY (IN PAIRS):

1. **Choose three words** from the shared vocabulary list.
2. **Write three original sentences** – one for each word. Make sure each sentence clearly shows the meaning through context.
3. When reading aloud to your partner, **replace the chosen word with the word “gap.”**

Tips:

- Use real examples from IT, software development, or your own experience.
- Try not to make the sentences too obvious – make your partner think!

5. VOCABULARY GAME: ALIAS

HOW TO PLAY (TEAM VERSION):

1. **Divide the class into 2–4 teams** (depending on size).
2. Each team takes turns. On each turn, one player from the team becomes the **explainer**.
3. The explainer picks a card (or word from a shared list) and must describe the word to their team – **without saying the word itself or using parts of it**.
4. The team has **1 minute** to guess as many words as possible.
5. For every correct guess, the team gets 1 point.
If the explainer breaks the rules (e.g. says the word), they **lose a point** for that round.

WHAT THE EXPLAINER CAN DO:

- Use definitions
- Use synonyms or opposites
- Give examples or situations
- Use IT/software-related context (if relevant)

WHAT THE EXPLAINER CANNOT DO:

- Say the word (even partially!)
- Spell the word
- Translate the word
- Use gestures (if you want to keep it strictly verbal)

WINNING THE GAME:

- After each team has had a turn (or several rounds), the team with the most points wins.

For Printable Vocabulary Materials for the Game “Alias,” go to page 230.

2

UNIT

Jobs and Responsibilities



This unit explores common roles in a software engineering team and the responsibilities associated with each. Students will learn to describe frontend, backend, full-stack, and QA positions, explain task distributions, and reflect on best practices. Through grammar-focused activities, learners will master key verb patterns using gerunds and infinitives.

Unit overview

2.1 Overview of Software Engineering Roles

Lesson outcome: Learners can describe common positions in a development team and talk about what different specialists do.

Skills practised:

Reading comprehension

Vocabulary: frontend, backend, QA, full-stack

Grammar: Gerund or infinitive – do, to do, doing

2.2 Detailed Responsibilities of Different Software Engineering Positions

Lesson outcome: Learners can discuss daily tasks and compare professional roles using appropriate verb patterns.

Skills practised:

Speaking (description, comparison)

Vocabulary: technical responsibilities, debugging, deployment

Grammar: Gerund or infinitive – verb patterns (part II)

2.3 Best Practices and Advice from Experienced Engineers

Lesson outcome: Learners can interpret and respond to advice using correct grammar and express what professionals recommend doing.

Skills practised:

Listening and summarising

Vocabulary: career advice, team communication, productivity tips

Grammar: Verb + object + infinitive/gerund – verb patterns

Software Engineering Roles:

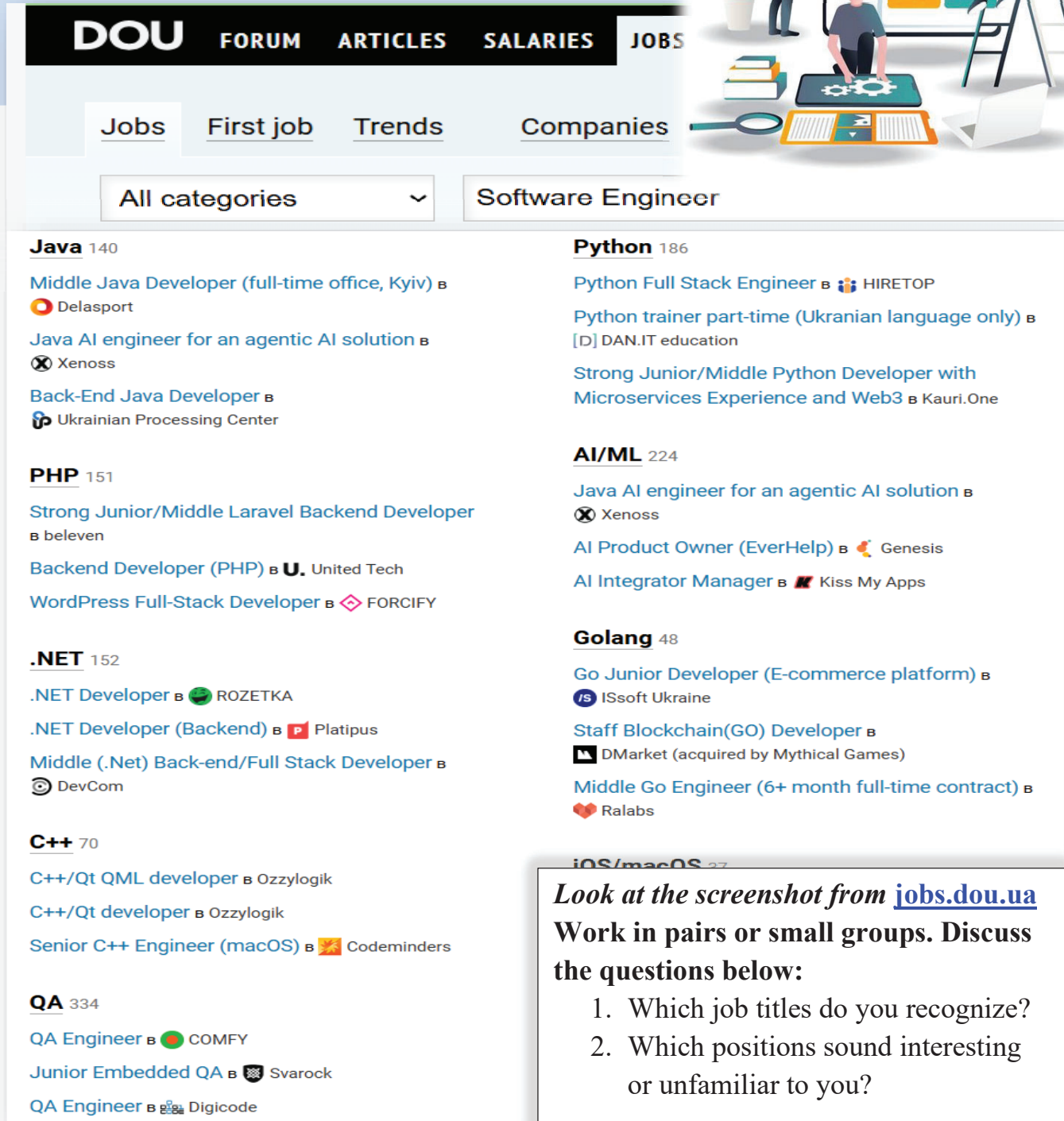


Figure 3 IT vacancies in Ukraine by role and tech stack (as of 09.07.2025) [Source: jobs.dou.ua]

Lesson 1

Overview of software engineering roles: frontend, backend, full-stack, and QA

Lead-in 1 What Does a Developer Do?

Discuss the questions below:

1. *Have you ever worked on or seen a software development project (e.g., website, app, system)?*
2. *What types of tasks do you think frontend and backend developers do?*
3. *What does a QA engineer do in a team?*
4. *What does it mean to be a full-stack developer?*

2. Role Match: Guess the Job

- | | |
|---|---------------|
| 1. Writes code that controls what the user sees on the screen | a. Frontend |
| 2. Tests the product to find bugs before release | b. Backend |
| 3. Builds and connects the database and server logic | c. Full-stack |
| 4. Handles both UI design and server-side functionality | d. QA |

3. Which of these jobs do you find most interesting or challenging? Why?

Use:

- *I'd like to try... because...*
- *That role sounds interesting / stressful / creative*

Vocabulary

Pre-reading 4 Match the highlighted words in the text below with their definitions.

word/phrase	definition
1. sought-after	a. To guess or calculate the cost, size, value, etc. of something
2. estimate	b. Wanted by many people and usually of high quality or rare
3. akin	c. Having some of the same qualities; similar
4. command	d. The mix of salary, benefits, and other incentives that employees receive from the organization
5. compensation package	e. To deserve and get something good, such as attention, respect, or a lot of money
6. varied	f. Incorporating a number of different types or elements; showing variation or variety
7. expand	g. To increase in size or amount
8. enterprise	h. A business organization in an area such as shipping, mining, railroads, or factories
9. encompass	i. Describes hardware, software, and network infrastructure to be used
10. architecture design	j. Individuals with versatile skills who can work across different technologies, tasks, or stages of a project, rather than focusing on a single area of expertise
11. UX	k. User Experience. This includes how a visitor/user navigates your site, how long they stay, whether or not they bounce
12. deployment	l. To develop; improve to reach an acceptable standard
13. data system	m. The fact of being moved to a particular place in order to be used there or to work there
14. shape up	n. The degree to which a system is easy to learn and efficient and satisfying to use
15. usability	o. The processes used to capture and the methods used to store data from various sources; the creation of a database
16. generalists	p. To encircle, go or reach around; to enclose; to include with a certain group or class

5  Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3>

<https://quizlet.com/946446620/match?funnelUUID=9c157903-71db-43d8-bc48-5b40d6371574>



Reading 6. Read the text quickly and choose the correct answer (a, b, or c) for each question below

 **2.1.**

Types of Software Engineering Roles

1. What has changed most about software engineering since the 1960s?

- a) Fewer people work in the field now.
- b) It is considered a low-skill job.
- c) It has become a highly respected and in-demand profession.

2. What does the text suggest about the roles in software engineering today?

- a) Most engineers only write code.
- b) Roles are diverse and cover various tasks and technologies.
- c) All engineers specialize in a single programming language.

3. What is the main difference between generalists and specialists?

- a) Generalists earn higher compensation packages.
- b) Specialists usually work alone.
- c) Generalists work across many areas, while specialists focus deeply on one.

4. Which of the following is NOT mentioned as a possible area of work for a software engineer?

- a) Managing infrastructure and deployment
- b) Designing hardware components
- c) Developing enterprise-level applications


According to a 2022 **estimate**, there are approximately 26.3 million professional software engineers worldwide. Since the 1960s, when software was considered a secondary field within computer science, the industry has **expanded** significantly. Initially, software tasks were seen as routine and were often assigned to women, while men handled the more prestigious hardware roles. Some historical accounts describe early software roles as being **akin** to clerical work like filing, typing, or switchboard operations.

Today, the situation has drastically changed. Careers in software engineering have become some of the most **sought-after** globally. These positions often **command** highly competitive **compensation packages**,

reflecting their value in the tech industry. The range of work has also become incredibly **varied**, offering opportunities across a broad spectrum of responsibilities.

Software engineering roles now **encompass** the entire lifecycle of digital product creation: planning, **architecture design**, development, testing, **deployment**, and maintenance. These tasks can be applied in many domains. For instance, you might work on an **enterprise**-level HR system, a mobile delivery app, or an online multiplayer game.

Engineers can specialize based on the device or operating system (e.g., iOS or Android), the type of application, or the programming language required. Their work might also support broader business needs like workflows, data management, or cross-team collaboration. Some engineers focus on infrastructure or **data systems**, while others may work on **UX** to ensure strong **usability**—that is, how easily and effectively users interact with the software.

What role you take on will influence how your career **shapes up**. Some professionals are **generalists**, familiar with a wide variety of technologies and capable of managing end-to-end solutions, including client communication and team coordination. Others are specialists, deeply focused on a specific field—such as AI development for industries like healthcare or finance—with the expertise to solve complex, niche problems. 

Close Reading

7. Read the text again carefully. Then read the following statements and **determine** whether they are **True** or **False**.

1. Software engineering was once considered a low-skill clerical function similar to filing or typing.
2. Today, software engineering roles are some of the least sought-after professions.
3. The field of software engineering includes everything related to planning, design, and development, but not maintenance.
4. Programming languages used by software engineers may vary depending on the type of device and application.
5. Generalist software engineers tend to focus deeply on one specific area of work.
6. Specialist engineers may focus on specific industries, such as AI app engineering for healthcare.

Speaking

8. Specialist vs. Generalist – What’s Best for You?

Work in pairs. Discuss the question below and take notes.

Do you prefer to be a specialist or a generalist? Why?

Use the prompts to help guide your conversation:

1. *What are the main tasks of a specialist?*
2. *What are the responsibilities of a generalist?*
3. *Which path would fit your skills or personality better?*
4. *What are the advantages of each role?*
5. *What are the challenges of each role?*

Useful phrases:

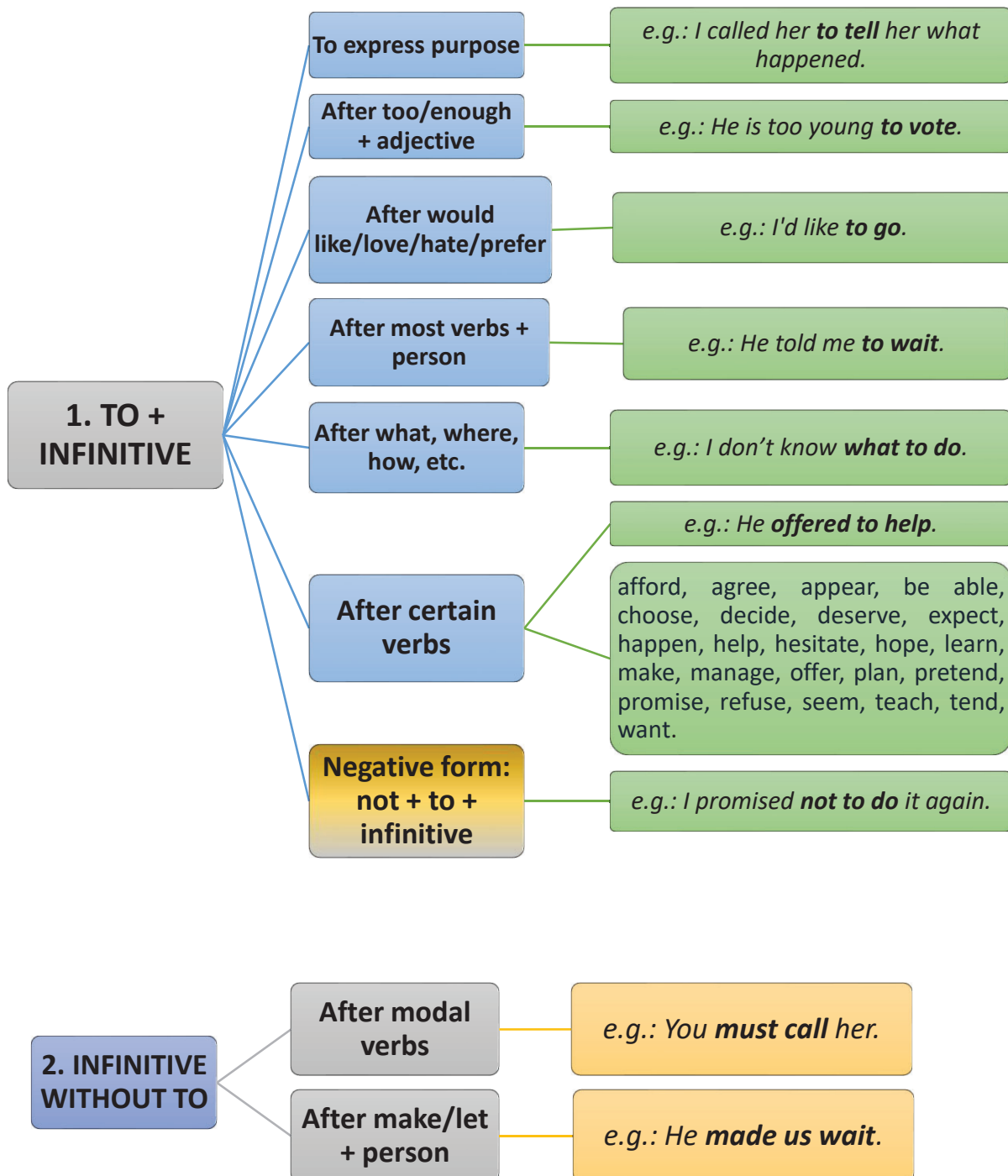
- *I’d prefer to be a specialist/generalist because...*
- *One advantage of being a specialist is...*
- *On the other hand...*
- *In the tech industry, it might be better to...*

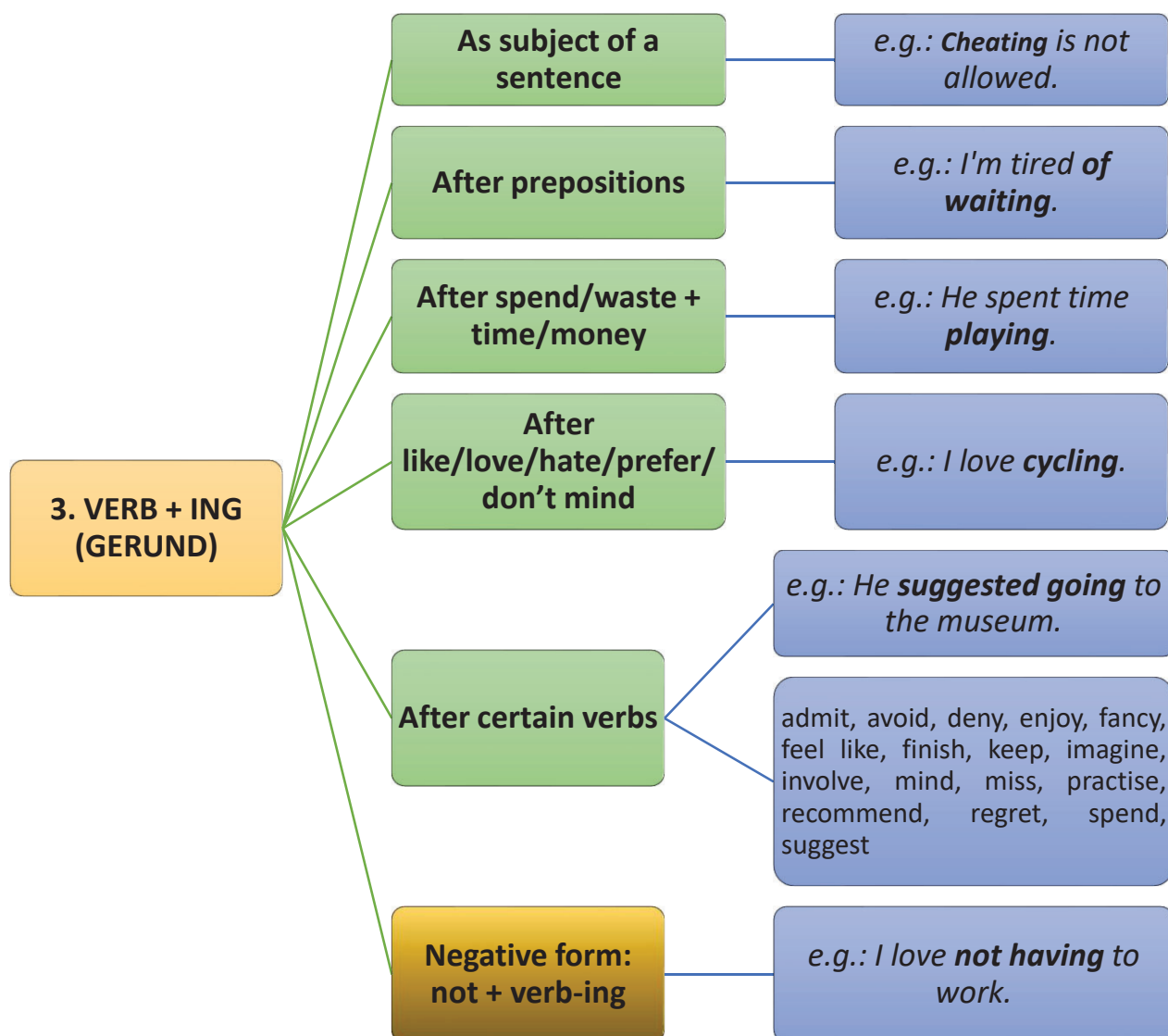
9. Writing Task (Optional)

Write a short opinion paragraph (80–100 words) answering the question:

Would you rather become a specialist or a generalist in your career? Explain your choice, and give one or two examples to support your answer.

Grammar for Revision Gerund or infinitive – do, to do, doing





Online Practice: Verb Patterns

10. Match each verb with its correct pattern as quickly as you can!

Play the Quizlet Match game by following this link:

<https://quizlet.com/1059509606/match?funnelUUID=5bfc4fe1-4877-4016-8786-e3a770a50d30>



Verbs that take gerund or infinitive with a change of meaning



11. Choose the correct gerund or infinitive form to complete the sentences

1. I decided **(start)** the new project next week.
2. She enjoys **(work)** on backend development.
3. We need **(test)** the feature before release.
4. He promised **(fix)** the bug by tomorrow.
5. They avoided **(use)** outdated libraries.
6. My manager wants me **(complete)** the task today.
7. I'm thinking about **(apply)** for a QA position.
8. They agreed **(deploy)** the new version on Friday.
9. Sometimes **(do)** nothing is more productive than doing something.

12. Match the sentence halves

Start of sentence

Ending

- | | |
|------------------|---|
| 1. He forgot | a. to update the GitHub repo. |
| 2. I'd like | b. working remotely every Friday. |
| 3. We suggested | c. to test the login function again. |
| 4. She admitted | d. writing the user guide herself. |
| 5. They plan | e. to submit the assignment. |
| 6. He promised | f. to restart the application. |
| 7. They avoided | g. to finish the report by Monday. |
| 8. We decided | h. asking too many technical questions. |
| 9. She suggested | i. updating the documentation before release. |
| 11. He keeps | j. to solve the issue without any help. |
| 10. managed | k. sending emails late at night. |

13. Error correction

Eight of the sentences below contain mistake in the use of gerund or infinitive.

TWO ARE CORRECT. Correct mistakes.

1. We avoid to use third-party tools in production.
2. She promised helping us with the release.
3. He decided going full-stack.
4. I enjoy to work on UI design.
5. They managed finishing the task on time.
6. I suggested to use automated testing.
7. We agreed to deploy the new version tonight.
8. She avoided to answer the security question.
9. He admitted breaking the code before the meeting.
10. Don't forget locking your screen when you leave.

13. Answer the questions using gerund or infinitive.

1. What do you enjoy doing most in a team project?
2. What tasks do you find difficult to start?
3. Have you ever promised to do something and then forgot?
4. Do you prefer to write or to debug code?

Extra Online Practice

Vocabulary:

[Quizlet – Unit 2.1 Flashcards & Learn](#)

Grammar:

Gerund or infinitive

[Exercises 1–3 on Test-English.com](#)



☒ Recommended activity before starting Lesson 2:

Play Quizlet Live (INDIVIDUAL MODE, 3 TIMES)

using vocabulary from Unit 2.1:

 <https://quizlet.com/ua/946446620/unit-21-flash-cards/?i=j03j6&x=1jqt>

Lesson 2

Detailed responsibilities of different software engineering positions

Lead-in 1 Role Guessing Game — "Who Am I?"

Description

Your Guess

1. *I turn visual designs into working websites and apps.*
2. *I build and manage the backend logic, APIs, and databases.*
3. *I write tests to find bugs before the software is released.*
4. *I handle both frontend and backend tasks.*
5. *I deploy and monitor code, automate testing, and manage infrastructure.*
6. *I build machine learning models and train them on large datasets.*
7. *I make sure software is protected from hackers.*
8. *I code mobile apps for Android and iOS.*
9. *I collect and organize massive amounts of data.*
10. *I create games and integrate sound, graphics, and controls.*

Options to Match (Frontend Developer; Cybersecurity Engineer; QA Engineer; Backend Developer; Full-Stack Developer; Game Developer; DevOps Engineer; Mobile App Developer; AI / Machine Learning Engineer; Data Engineer).

Pre-reading: 2. Read the following statements and decide if you think they are **true** or **false**. Then, after reading the text, **check your answers**.

1. DevOps engineers help with testing, deployment, and automation.
2. QA engineers are responsible for building software features.
3. AI engineers work with machine learning and large datasets.

Reading 3 Scan the text for job titles and numbers (e.g. salaries), then complete the table with each role's main responsibility and salary range.

Write down the key responsibility and typical salary range based on the reading.

Role	Main Responsibility	Salary Range
1. Frontend Developer		
2. Backend Developer		
3. Full-Stack Developer		
4. DevOps Engineer		
5. AI / ML Engineer		
6. QA Engineer		
7. Data Engineer		
8. Mobile App Developer		
9. Cybersecurity Engineer		
10. Game Developer		

Follow-up questions:

Which job pays the most?

Which job would you personally choose — and why?

2.2.

Different Types of Software Engineers

Not all software engineer roles are created equal. The skills, responsibilities, and compensation for each vary widely.

A **frontend engineer** is responsible for building the user interface we interact with, turning designs from the UX team into functional software. They work with technologies like HTML, CSS, and JavaScript to convert design visions into working interfaces, write reusable UI components, and

ensure smooth backend integrations. Frontend engineers typically earn between \$113,000 and \$183,000, with **lead engineers** reaching up to \$260,000.

On the backend, engineers manage the server side of applications, dealing with architecture, business logic, databases, and APIs. Their work includes building **scalable** infrastructures, ensuring high performance and low **latency**, and preparing applications for deployment. Backend engineers earn between \$111,000 and \$217,980, with an average salary of \$155,800.

A **full-stack engineer** **handles** both frontend and backend tasks, taking **end-to-end** responsibility for the entire application development. They design architecture, build data structures, and write code for both frontend and backend, among other tasks. Full-stack engineers earn an average of \$125,600, with top earners reaching \$192,325, often with additional performance bonuses or **stock options** in startups.

DevOps engineers ensure smooth operations throughout the software development lifecycle, setting up the tools and processes needed to move code into functioning applications. They manage automation, **CI/CD pipelines**, and server infrastructure, with salaries ranging from \$133,750 to \$171,000 for experienced professionals.

AI engineers develop artificial intelligence and machine learning applications, designing scalable AI pipelines and deploying models on cloud platforms like AWS or Azure. Some AI engineers also create statistical models for **data mining**. This role is highly in demand, with salaries ranging from \$155,900 to \$338,000 for experienced engineers.

Game developers design and develop computer or console-based games, translating visual ideas into code using languages like C++ and Java. They test the user experience, integrate elements like graphics and audio,


and ensure stability across platforms. Game developers earn between \$116,189 and \$214,000, depending on experience.

A quality assurance engineer, also known as a tester, ensures software meets quality standards before **release**. They run manual and automated tests, analyse results, track quality issues, and help develop strategies for continuous improvement. QA engineers earn an average of \$107,235, with salaries ranging from \$89,000 to \$140,000.

Data engineers build systems that allow organizations to collect and analyse data. They develop data pipelines, automate **data cleaning** processes, and ensure **compliance** with security protocols. Data engineers typically earn between \$82,278 and \$196,879, with an average salary of \$127,275.

Mobile application developers create software for mobile devices, using Java or Kotlin for Android and Swift for iOS. Their work involves integrating **third-party libraries**, managing server-side components, and ensuring app security. Mobile developers earn between \$80,643 and \$203,794.

Lastly, **cybersecurity engineers** are tasked with securing applications, networks, and data. They manage **identity** and **access control**, develop incident response strategies, and conduct regular risk assessments. Cybersecurity engineers earn an average of \$154,000, with top earners reaching up to \$333,000.

These roles represent just a small portion of the many career paths available in software engineering. As you gain experience, you can explore numerous other roles within the field. 

Vocabulary 4 Match the highlighted words in the text with their definitions

word/phrase	definition
1. vary	a. to guess or calculate the cost, size, value, etc. of something
2. UI	b. wanted by many people and usually of high quality or rare
3. scalable	c. having some of the same qualities; similar
4. latency	d. the mix of salary, benefits, and other incentives that employees receive from the organization
5. handle	e. to deserve and get something good, such as attention, respect, or a lot of money
6. end-to-end	f. incorporating a number of different types or elements; showing variation or variety
7. stock option	g. to increase in size or amount
8. CI/CD pipeline	h. a business organization in an area such as shipping, mining, railroads, or factories
9. data mining	i. describes hardware, software, and network infrastructure to be used
10.compliance	j. user experience. this includes how a visitor/user navigates your site, how long they stay, whether or not they bounce
11.release	k. to develop; improve to reach an acceptable standard
12.data cleaning	l. the fact of being moved to a particular place in order to be used there or to work there
13.access control	m. the degree to which a system is easy to learn and efficient and satisfying to use
14.identity	n. the processes used to capture and the methods used to store data from various sources; the creation of a database
15.third-party library	o. to encircle, go or reach around; to enclose; to include with a certain group or class

5  Play the Quizlet Match game by following this link:

<https://quizlet.com/946654164/match?funnelUUID=cd0182f0-1537-4feb-a71d-4cdf50eb370b>



6. Fill in the blanks using words or phrases from Ex.4. Use the correct form if needed.

1. Frontend developers focus on building a clean and user-friendly _____.
2. Companies aim to reduce _____ so that apps respond faster.
3. The system must be _____ so it can handle millions of users.
4. Many startups offer a _____ to employees as part of their compensation.
5. Engineers use _____ pipelines to automate testing and deployment.
6. Before launching the new platform, developers had to do extensive _____ to remove errors in the dataset.
7. A good software engineer must be able to _____ problems under pressure.
8. Cybersecurity experts manage _____ to ensure only the right people can access systems.
9. Backend developers often integrate _____ to save time on coding common features.
10. Data engineers need to follow strict _____ standards, especially in healthcare and finance.
11. AI engineers use _____ techniques to discover patterns in large datasets.
12. A full-stack developer often works on _____ solutions, from design to deployment.
13. After months of testing, the team was ready for the final product _____.
14. A user's _____ must be verified before granting access to internal systems.
15. Tasks and responsibilities can greatly _____ depending on the company and role.

Speaking 7. Work in Pairs: Interview a Software Engineer

Instructions:

Choose a software engineering role from this lesson (e.g. QA Engineer, Backend Developer, AI Engineer).

Take turns role-playing: one of you is the **interviewer**, the other is a **software engineer**.

Interview Prompts (ask and answer in character):

- › Can you tell me about your current job?
- › What are your main responsibilities?
- › What tools or technologies do you use most?
- › What do you enjoy about your role?
- › What's the most challenging part of your work?
- › Do you work more independently or in a team?

 **Swap roles and repeat the interview with a different engineering position.**

☒ *Try to use vocabulary from the lesson (e.g. deployment, automation, API, user interface, testing, cloud, infrastructure, etc.).*

Extra Online Practice

Vocabulary:

[Quizlet – Unit 2.2 Flashcards & Learn](#)

Grammar:

Gerund or infinitive: Verb patterns

[Exercises 1–4 on Test-English.com](#)



☒ Recommended activity before starting Lesson 3:

Play Quizlet Live (INDIVIDUAL MODE, 3 TIMES) using vocabulary from Unit 2.2:

 <https://quizlet.com/ua/946654164/unit-22-flash-cards/?i=j03j6&x=1jqt>

Lesson 3

Best practices and advice from experienced software engineers

Lead-in 1 Read the following pieces of advice from real software engineers. Discuss whether you agree or disagree with each one, and why.

1. *“Attention to detail is what separates great developers from the rest.”*
2. *“Don’t tie your identity to a language. Be a developer, not just a ‘Python developer’.”*
3. *“Understanding how things work under the hood will make you a better engineer.”*

 **Discussion prompts:**

- › Which of these do you already follow?
- › Which advice do you think is the most important for junior developers?
- › Have you ever ignored advice that later turned out to be useful?

4. *“You don’t need many years of experience to be a good developer — curiosity matters more.”*
5. *“Don’t be afraid to learn something ‘unnecessary’ — it will pay off.”*
6. *“You’ll spend more time communicating with people than writing code — be good at both.”*
7. *“It’s okay to switch roles or paths. Your first job doesn’t define your whole career.”*

2. Watch the video: How to Become a Great Software Developer — Best Advice from Top-Notch Engineers

As you watch, **match each piece of advice** from Exercise 1 to the person who gave it in the video.

		Advice #
Speaker 1	Egor Tolstoy - Kotlin Project Lead / JetBrains	—
Speaker 2	Pavel Veller - Chief Technologist / EPAM Systems	4; —
Speaker 3	Roman Elizarov - ex-Kotlin Project Lead	—
Speaker 4	Andrey Breslav - ex-Kotlin Lead Language Designer	—
Speaker 5	Dmitry Jemerov - co-author of "Kotlin in	—; 5

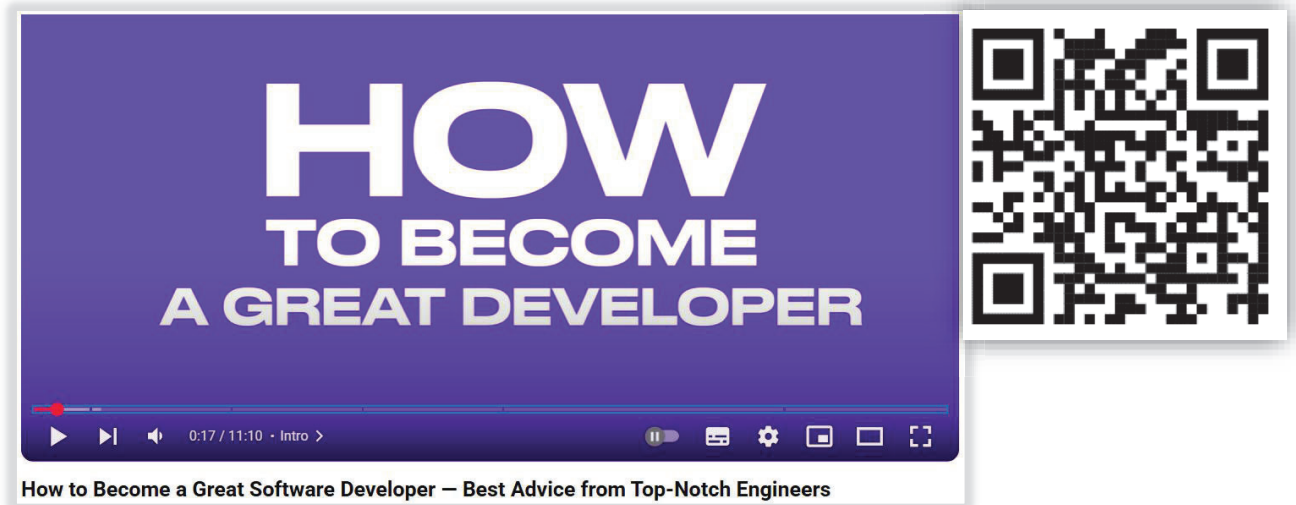


Figure 4 How to Become a Great Software Developer — Best Advice from Top-Notch Engineers <https://youtu.be/suATPK45sjk>

Vocabulary

3. Match the words or phrases from the video with their definitions.

- | | |
|--------------------|--|
| 1. algorithm | a. eager to learn something or know more |
| 2. curious | b. the state of being receptive to other ways of looking at things |
| 3. open source | c. the way you think and approach learning or problems |
| 4. open-mindedness | d. the ability of a program to execute multiple tasks simultaneously |
| 5. fundamentals | e. a set of routines, protocols, and tools for building software applications |
| 6. under the hood | f. what happens in a system beyond the user interface |
| 7. concurrency | g. a step-by-step procedure for solving a problem |
| 8. mindset | h. software with source code that anyone can inspect, modify, and enhance. |
| 9. API | i. basic, essential knowledge or principles |
| 10. data structure | j. a particular way of organizing and storing data such as an array, table, etc. |

4 Play the Quizlet Match game by following this link:

<https://quizlet.com/946723072/match?funnelUUID=66cfe53a-57fa-4e48-a1bb-e49d9ca03501>



Listening

5. Watch the video again. Rearrange the following phrases from the video into the correct order.

- a. "Don't tie your identity to a specific technology."
- b. "Learning different programming languages is beneficial."
- c. "Attention to detail is crucial for professional developers."
- d. "Understanding how things work under the hood makes you a better engineer."
- e. "Embrace curiosity and a willingness to learn."
- f. "You should aim higher and not underestimate your abilities."
- g. "Collaboration and communication are essential skills."
- h. "It's important to find a healthy balance in your life."
- i. "Keep learning new things and stay updated with trends."
- j. "Great junior developers can contribute significantly."

 **Write numbers 1–10 next to each phrase.**

Comprehension Questions.

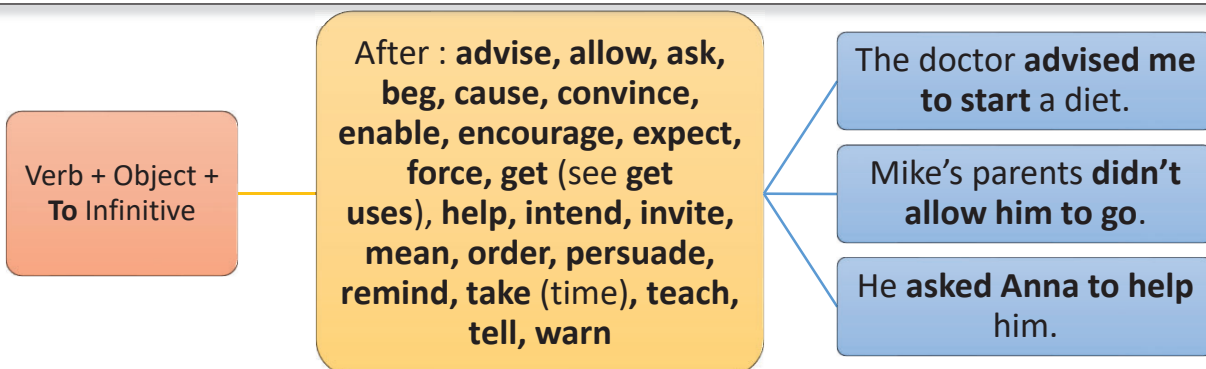
6. Answer the following questions after watching the video.

- 1. What qualities differentiate professional developers from junior developers?
- 2. Why is it beneficial for developers to understand how systems work under the hood?
- 3. How does the speaker suggest that developers should approach learning new technologies?
- 4. What advice does the speaker give regarding career paths in software development?

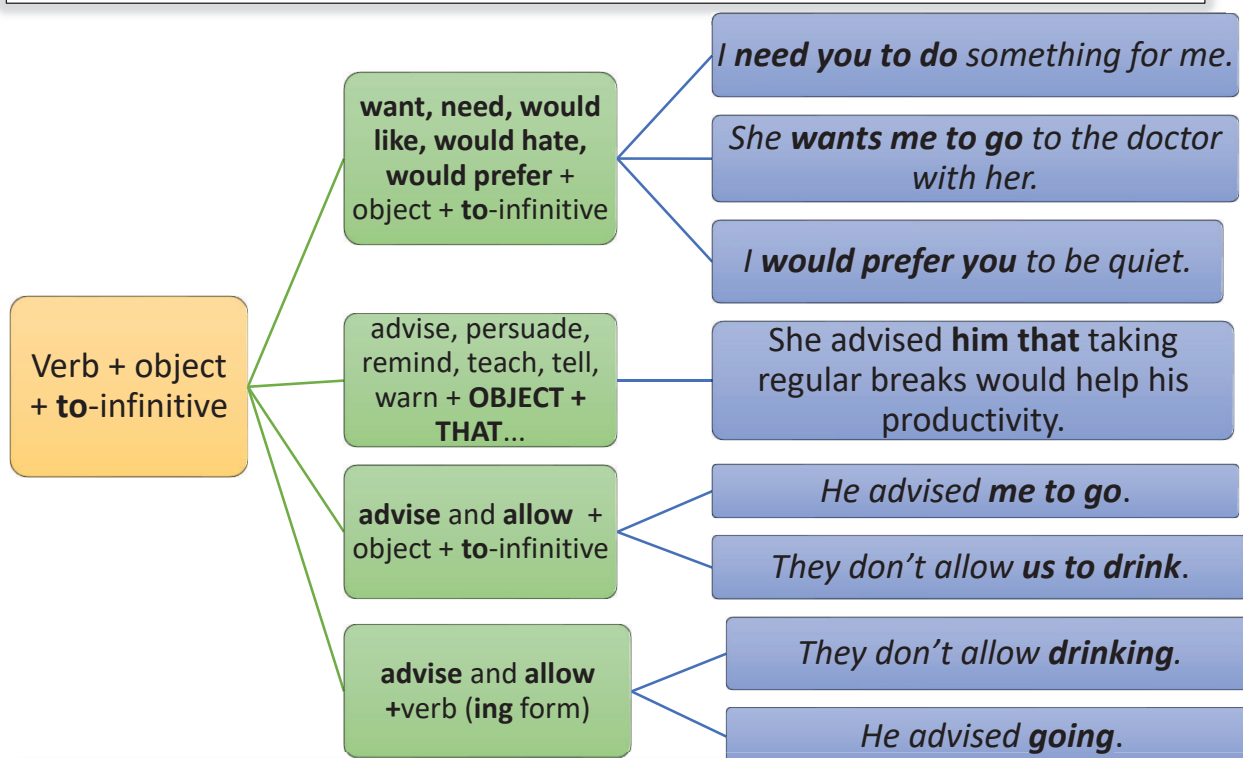
Grammar for Revision Verb + object + infinitive/gerund – verb patterns

Verb + object + to-infinitive

Used with verbs such as *advise, allow, ask, beg, cause, convince, enable, encourage, expect, force, get, help, intend, invite, mean, order, persuade, remind, teach, tell, warn*.



Want/Need/Would like/Hate/Prefer + Object + To-Infinitive
Expressing desires or intentions

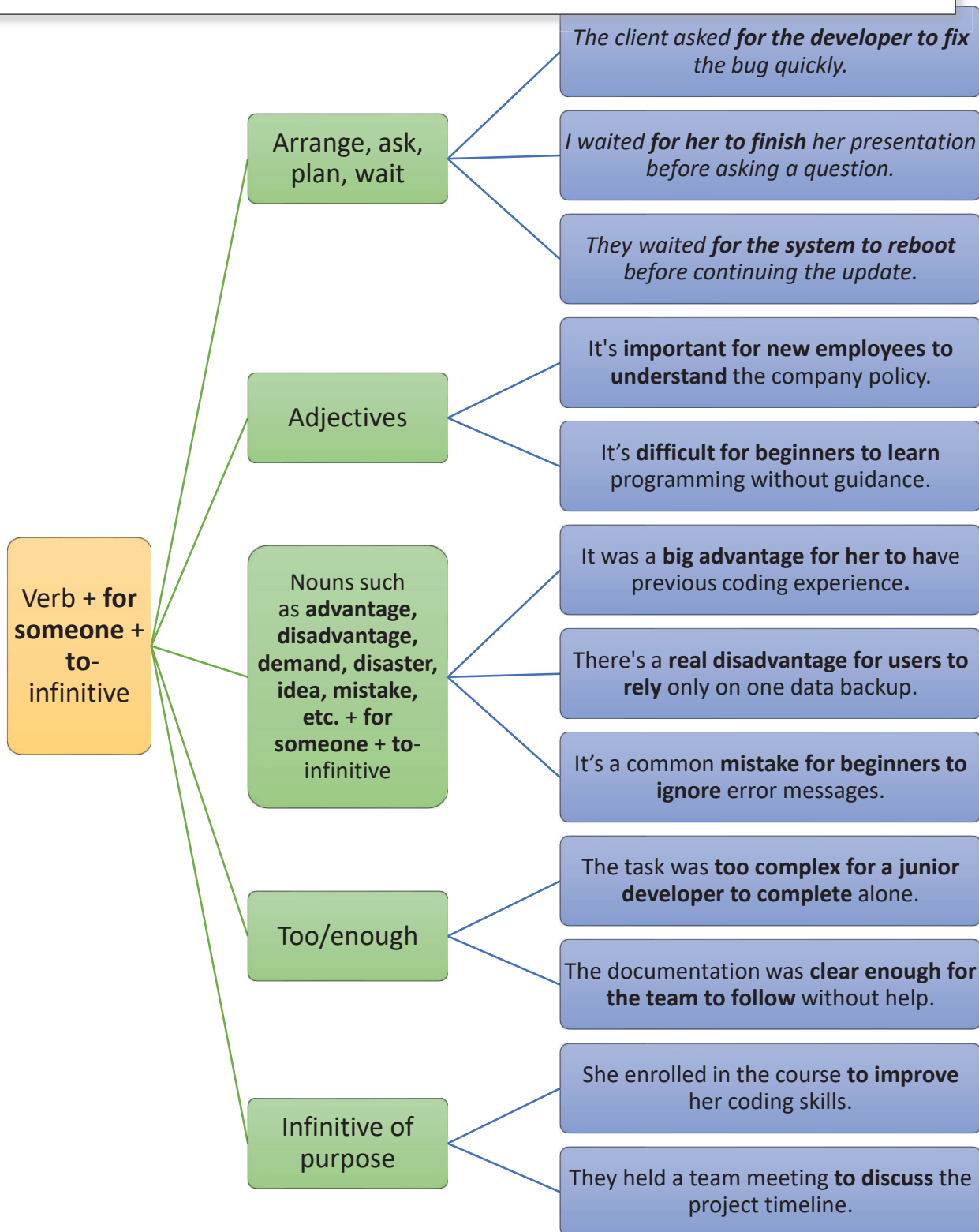


With verbs like *advise, persuade, remind, teach, tell, warn* — you can also use a *that* clause instead of *to-infinitive*:

Verb + for someone + to-infinitive

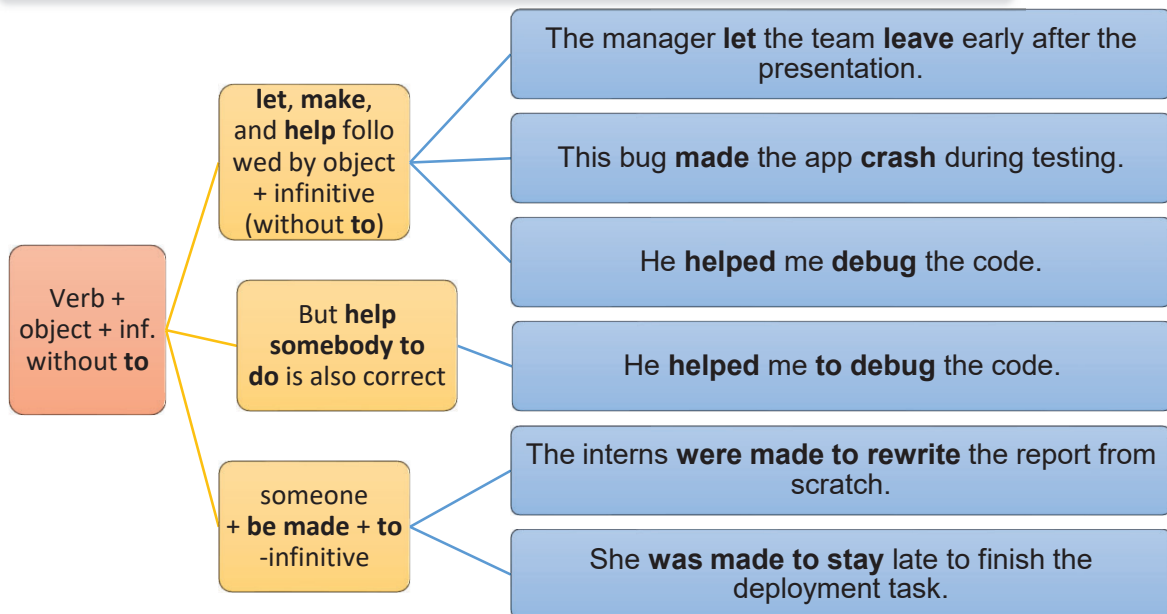
Verb + For Someone + To-Infinitive

Used with verbs like arrange, ask, plan, wait, certain adjectives, and nouns



Verb + object + infinitive without to

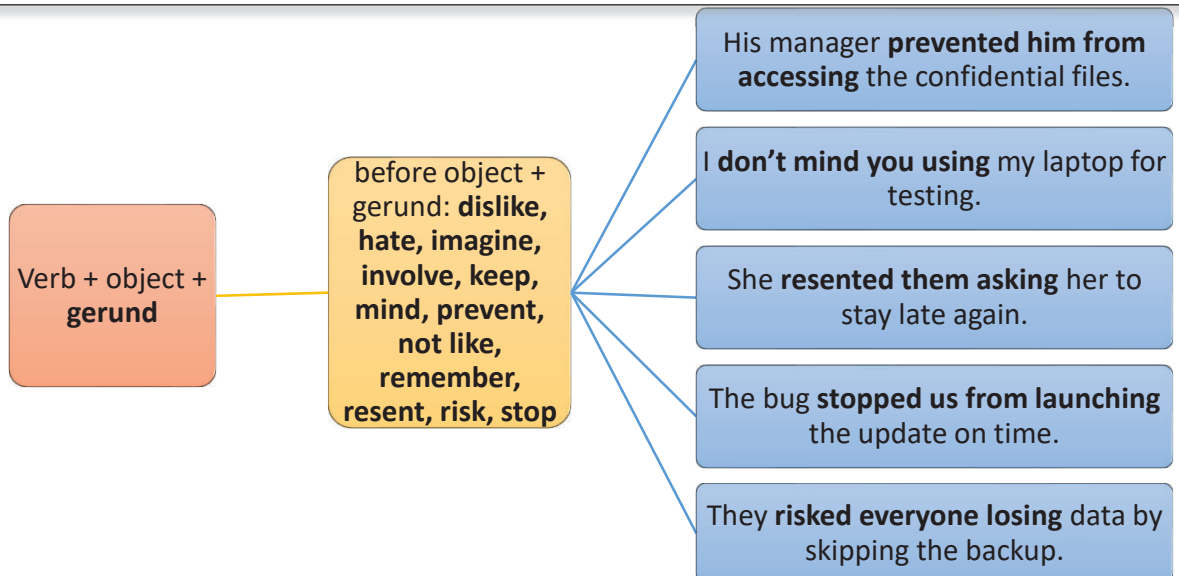
With **let**, **make**, **help**, and silent bare infinitive constructions after hear, see, notice for short actions



Verb + object + gerund

Some **verbs** take an **object followed by a gerund**:

dislike, hate, imagine, involve, keep, mind, prevent, not like, remember, resent, risk, stop



7. Complete each sentence with the correct form of the verb in brackets.

1. The lead developer **advised** the junior team member _____ more comments in the code. (*use*)
2. They **made** me _____ the whole test suite again after the update. (*run*)
3. I would hate you _____ that I didn't appreciate your help. (*think*)
4. We arranged _____ the consultant _____ the API integration. (*for / handle*)
5. The project manager reminded us _____ the staging server before deployment. (*check*)
6. I **don't mind** you _____ me during the team meeting. (*correct*)
7. Our mentor **taught us** _____ clean, readable code. (*write*)
8. He persuaded the team _____ a different framework for this project. (*try*)
9. The recruiter told me _____ a portfolio with real projects. (*prepare*)
10. The team **kept** me _____ even after the deadline had passed. (*wait*)
11. She warned the interns _____ the logs carefully before deleting anything. (*read*)
12. The instructor explained **that** small changes in code could have big consequences. (*use a that-clause*)
13. It's important _____ junior developers _____ feedback well. (*for / receive*)
14. Our manager let us _____ the stand-up meeting remotely. (*join*)
15. It was a mistake _____ the update before final testing. (*release*)

8. Choose three of your completed sentences and turn them into spoken advice you might give a real junior developer.

Example: "I'd advise you to comment your code properly."

9. Match the sentence beginnings (1–10) with the correct endings (A–J)

- | | |
|---|--|
| 1. The mentor advised the junior developer... | A. ...to submit the report before the end of the day. |
| 2. They arranged for the tech lead... | B. ...fix the bugs again after the code review. |
| 3. She was made... | C. ...asking for help during the sprint. |
| 4. The team made me... | D. ...to complete the code review before the meeting. |
| 5. She reminded us... | E. ... to submit the patch without a peer review. |
| 6. It's important for every developer... | F. ...make changes without version control. |
| 7. I'd rather not... | G. ...against using the wrong dataset for testing. |
| 8. He didn't mind me... | H. ...to build the test environment ahead of the workshop. |
| 9. Our manager warned us... | I. ...to document their work clearly. |
| 10. They kept... | J. ...to refactor the login form. |

Extra Online Practice

Vocabulary:

[Quizlet – Unit 2.3 Flashcards & Learn](#)

Grammar:

Gerund or infinitive: Verb patterns

[Exercises 1–3 on Test-English.com](#)



- ☒ Recommended activity before starting **Revision Unit 2:**

Play Quizlet Live (INDIVIDUAL MODE, 3 TIMES) using vocabulary from Unit 2.3:

 <https://quizlet.com/ua/946723072/unit-23-flash-cards/?i=j03j6&x=1jqt>

Vocabulary Revision

Unit 2

 1. WORK IN PAIRS. TAKE TURNS PLAYING THE ROLES OF **DESCRIBER** AND **GUESSER** USING VOCABULARY FROM UNIT 2.

◇ Detailed instructions can be found on page 37

Term	Definition
advocate	To support; to be in favour of
accomplish	To do, make happen, succeed in, carry through
remarkable	Unusual, extraordinary, worthy of attention
software specification	Where customers and engineers define the software that is to be produced and the constraints on its operation
software design	Produces a software solution to realize the software requirements
software architecture	The overall software structure that depicts the major system components and how they relate, interface, and interact with each other
software engineering	A discipline focused on the research, education, and practice of engineering processes, methods, and techniques to increase software productivity and quality
software quality assurance	Ensures that the development activities are carried out correctly
in this regard	In connection with the point previously mentioned
commit	To carry out or do
commitment	A promise or pledge to do something
analysis	A detailed examination of the elements or structure of something
specification	A detailed description of the design and materials used to make something
maintenance	The work that is done to keep something in good condition
consistently	Continually; regularly
integrity	Honesty, high moral standards; an unimpaired condition, completeness, soundness
exhibit	Show, display, present, demonstrate
rapid expansion	Product acceptance is growing and investors become very interested

sequence	The order in which things happen or should happen
requirement	Something that is necessary, that must be done
resemble	To be like or similar to
deem	To think, believe; to consider, have an opinion
in the pipeline	Being planned or in progress (refers to a sequence of steps or stages that data, tasks, or products go through to achieve a desired outcome)
embedded system	A computer system that is part of a larger machine and controls how that machine operates
capabilities	The qualities or abilities to perform a function
long-lasting	Continuing for a long period of time
enhancement	The process of improving the quality, amount, or strength of something
issue (v)	To produce or provide something official
rigorously	In a careful way so that every part of something is looked at or considered to ensure it is correct or safe
milestones	Formal project review points used to assess progress and performance
release (v)	To make a product available for the public to buy, often with a celebration; launch
increment	One of a series of increases; an enlargement, increase, addition
incremental software	A method of building software products in which a system is built piece-by-piece
crucial	Extremely important; vital in resolving something
adjustment	A slight change made to something to make it fit, work better, or be more suitable, or the act of making such a change
timeline	A graphic representation of the passage of time as a line
deliverable	Something that can be provided or achieved as a result of a process: A product, service, or result created by a project
iterative	A process that repeats a series of steps over and over until the desired outcome is obtained
sequential	Following a particular order, arranged serially

2. THREE-SENTENCE CHALLENGE. Each participant **selects three words** from the shared word list and forms three sentences using these words. Work in pairs. Read your sentences to your partner, replacing the chosen word with "gap" while reading. Your partner needs to guess the missing word.

3. **PLAY THE QUIZLET LIVE GAME**

(TEAM MODE) to review the vocabulary learned in **UNIT 2**

<https://quizlet.com/ua/946742712/units-2-23-22-21-flash-cards/?i=j03j6&x=1jqt>



4. **TAKE the UNIT 2 VOCABULARY QUIZLET TEST** to check your understanding of key terms from this UNIT.

<https://quizlet.com/946742712/test?answerTermSides=2&promptTermSides=4&questionCount=30&questionTypes=15&showImages=true>



5. 🎲 **VOCABULARY GAME: ALIAS**

Play a fun team-based vocabulary challenge!

◇ **See full game instructions on page 37.**

Get ready to explain, guess, and compete using key terms from Unit 2!

For Printable Vocabulary Materials for the Game “Alias,” go to page 231.

3

UNIT

Programming Languages



This unit introduces learners to essential concepts in programming languages, including syntax, semantics, and language paradigms. Students will examine the similarities and differences among popular programming languages such as Python, Java, JavaScript, C++, and C#. The final lesson lays the groundwork for understanding databases within the context of software development. Additionally, grammar practice in this unit focuses on phrasal verbs and quantifiers commonly used in technical descriptions and comparisons.

Unit overview

3.1 Introduction to Programming Languages: Syntax, Semantics, and Paradigms

Lesson outcome: Learners can explain the core elements of programming languages and describe different programming paradigms.

Skills practised:

Reading for key concepts

Vocabulary: syntax, semantics, paradigm, interpreted, compiled

Grammar: B1 phrasal verbs (Part 1)

3.2 Comparing and Contrasting Popular Programming Languages

Lesson outcome: Learners can compare programming languages and express technical preferences using appropriate vocabulary and grammar.

Skills practised:

Speaking (comparison, expressing opinion)

Vocabulary: Python, Java, JavaScript, C++, C#, object-oriented

Grammar: B1 phrasal verbs (Part 2)

3.3 Introduction to Databases

Lesson outcome: Learners can describe basic database concepts and discuss use cases in software development.

Skills practised:

Reading comprehension



Vocabulary: database, query, SQL, table, record, relationship

Grammar: B1 phrasal verbs (Part 3)

Is One Programming Language Better Than Another?

Matching Game: Guess the Programming Language

Match each code snippet (A–E) with the correct programming language (1–5).

1.  Java _____
2. JavaScript _____
3. C++ _____
4.  Python _____
5. C# _____

```
public class Main {  
    int x = 5;  
  
    public static void main(String[] args) {  
        Main myObj = new Main();  
        System.out.println(myObj.x);  
    }  
}
```

A

```
#include <iostream>  
using namespace std;  
  
class MyClass {           // The class  
public:                   // Access specifier  
    void myMethod() {     // Method/function  
        cout << "Hello World!";  
    }  
};
```

C

```
int main() {  
    MyClass myObj;        // Create an object of MyClass  
    myObj.myMethod();     // Call the method  
    return 0;  
}
```

```
using System;  
  
namespace MyApplication  
{  
    class Car  
    {  
        string color = "red";  
  
        static void Main(string[] args)  
        {  
            Car myObj = new Car();  
            Console.WriteLine(myObj.color);  
        }  
    }  
}
```

E

```
class Person:  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
  
p1 = Person("John", 36)  
  
print(p1.name)  
print(p1.age)
```

B

```
<script>  
class Car {  
    constructor(name, year) {  
        this.name = name;  
        this.year = year;  
    }  
    age() {  
        const date = new Date();  
        return date.getFullYear() - this.year;  
    }  
}  
  
const myCar = new Car("Ford", 2014);  
document.getElementById("demo").innerHTML =  
    "My car is " + myCar.age() + " years old."  
</script>
```

D

Lesson 1

Introduction to Programming Languages: Syntax, Semantics, and Paradigms

Lead-in 1 “What Makes a Language?”

 Work in pairs or small groups. Discuss the following questions:

1. How many programming languages can you name?
(Make a list together — then share with the class.)
2. Which programming languages have you used or heard of? What were they used for?
(e.g. games, websites, mobile apps, data analysis)
3. Do you think all programming languages are similar? Why or why not?
(Think about how they look, what they're used for, how easy they are to learn.)
4. What do you think the words *syntax*, *semantics*, and *paradigm* might mean when we talk about programming?

Reading: Scan for Key Concepts.

2. Skim the text and find answers to the following questions:

1. What are the three main programming paradigms mentioned in the text?
2. Which paradigm uses "pure functions" and avoids shared state?
3. What does the word "method" refer to in Object-Oriented Programming?
4. What is the difference between syntax and semantics?
5. How is Chomsky's "Colourless green ideas sleep furiously" related to programming?
6. Why is it important to do type checking or semantic analysis in programming?

3.1.

Introduction to Programming Languages

What is a programming paradigm? It is a style of programming, a way of thinking about software construction. A programming paradigm does not refer to a specific language but rather to a methodology, a particular way to approach programming.

Let's dive into the three programming paradigms that interest us today:

Procedural Programming (PP), also known as **inline** programming, takes a top-down approach. It involves writing a **sequence** of instructions to tell the computer what to do, step by step. This paradigm relies on procedures or routines, which are blocks of code designed to perform specific tasks.

Object-Oriented Programming (OOP) focuses on **encapsulating** data and behaviour into objects. An OOP application is built using a collection of objects, each of which knows how to perform certain actions and how to interact with other elements of the application. For example an object could be a person. That person would have a name (that would be a **property** of the object), and would know how to walk (that would be a method). In this paradigm, a method in OOP can be considered similar to a procedure in PP, but it belongs to a specific object. Another important concept in OOP is the class, which serves as a **blueprint** for creating objects.

Functional Programming (FP) emphasizes passing data through a series of functions to achieve a result. In FP, functions are treated as data, meaning they can be passed as parameters, returned from other functions, and combined to create new functions. Functions in FP are required to be **pure**, meaning they avoid **shared state**, side effects, and rely on **immutable** data. A pure function is one that, given the same input, will always return the same output, independent of any external state. Shared state, on the other hand, is a state that is accessible by multiple functions or data structures. Managing shared state adds complexity and reduces **modularity**, making it harder to understand and maintain code.

Programming languages, the tools we use to communicate with computers, are **intricate** systems with their own rules and structures. To effectively **harness** their power, we must understand the fundamental concepts of syntax and semantics. Syntax governs the structure of a program, **akin** to grammar in human language, while

semantics defines the meaning of these structures. By grasping these concepts, we can write programs that are not only syntactically correct but also logically sound.


In the field of linguistics, Noam Chomsky's famous counter-example illustrates the complexity of syntax and semantics. The sentence "*Colourless green ideas sleep furiously*" is syntactically well-formed, with "ideas" as the noun, "sleep" as the verb, and "furiously" as an adverb. These are all modifiers that form a noun phrase. However, the sentence's meaning is unclear. For instance, what does it mean for an idea to sleep? This difficulty in interpretation is evident even when trying to visualize the sentence - how do you represent colourless, green ideas that sleep furiously?

This challenge is not limited to natural language. In computer programming languages like Python, we encounter similar notions. For example, adding two numbers with `"1 + 1"` or concatenating two strings with `"hello + world"` yields expected results. However, attempting to add an integer to a string, like `"1 + hello,"` doesn't make sense and will cause a runtime error in Python. The interpreter, unable to process the command, essentially throws up its hands and says, "I don't know."

When programming, it's crucial to avoid these kinds of semantic errors. Consider a few program fragments: `"2 + 2"` evaluates correctly to 4, but `"Mars = Earth + 1"` causes an error because "Earth" is undefined. Similarly, adding an integer to a string like `"Mars + 2"` results in another error because addition isn't defined between these types.

To ensure that our programs run correctly, especially when writing an interpreter for JavaScript in HTML, we need to distinguish between well-formed and erroneous programs. This process, known as *type checking* or semantic analysis, involves examining a program's source code to determine if it's well-behaved.

This concept extends to natural language as well, where words like "execute" have different meanings depending on the context. In English and Greek, syllepsis—a form of wordplay—exploits this semantic incongruity. For example, the phrase "*she made no reply, up her mind, and a dash for the door*" uses the word "made" in different senses, creating a humorous effect. Another example is the line, "*She lowered her standards by raising her glass, her courage, her eyes, and his hopes,*" where "lowered" and "raised" are applied to different objects, each with its own meaning.

In essence, programming languages, like natural languages, require a nuanced understanding of their structure and meaning. By differentiating between syntax and semantics, we lay the groundwork for constructing reliable and efficient programs. While syntax provides the blueprint, semantics breathes life into the code, ensuring it performs as intended. Just as a grammatically correct sentence can be semantically nonsensical, a syntactically correct program might still produce unexpected results if its semantics **are flawed**. A deep comprehension of both syntax and semantics is therefore essential for any aspiring programmer. 

Vocabulary 3.

Match the highlighted words in the text with their definitions.

1. inline	a. a detailed plan or design that serves as a guide for constructing something.
2. sequence	b. the quality of being complex and detailed, often requiring careful attention.
3. encapsulating	c. a characteristic or attribute that belongs to something, especially in programming, it refers to data associated with an object.
4. property	d. the act of containing or covering something within another, often used to describe the grouping of data and methods within objects in OOP.
5. blueprint	e. a series of related events, actions, or items that follow each other in a particular order.
6. immutable	f. the act of using or controlling something effectively, such as power or resources.
7. intricate	g. something that is unchangeable or cannot be altered once it has been created.
8. harness	h. a deviation from what is expected, leading to a sense of inconsistency or contradiction.
9. akin	i. similar in nature or function; having a resemblance to something else.
10. incongruity	j. to contain defects or imperfections; not being perfect.
11. be flawed	k. incorrect, containing mistakes
12. concatenate	l. to put things together as a connected series
13. erroneous	m. within a program or procedure, often referring to code that is placed directly in the sequence of execution.

4. 🎮 Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3> <https://quizlet.com/932616895/match?funnelUUID=71f9da18-285d-4a48-9031-f1db28363c88>



5. Complete the sentences below using the following words:

modularity, integer, string, runtime error, interpreter, erroneous

1. An _____ is a whole number, often used in programming for counting or indexing.
2. In Python, a _____ is a sequence of characters enclosed in quotes, such as "hello".
3. If you try to add an integer to a string, the program will likely produce a _____ because the operation doesn't make sense.
4. A program's _____ checks the source code for errors and executes it line by line.
5. Good programming practices emphasize _____, which involves breaking down a program into smaller, reusable components.
6. An _____ program might appear correct at first glance but will fail when executed due to logical flaws.

6. Translate the following terms into your native language:

1. Integer _____
2. String _____
3. Runtime error _____
4. Interpreter _____
5. Modularity _____
6. Erroneous _____

7. Translate these sentences into your native language:

“For example, the phrase *"she made no reply, up her mind, and a dash for the door"* uses the word "made" in different senses, creating a humorous effect. Another example is the line, *"She lowered her standards by raising her glass, her courage, her eyes, and his hopes,"* where "lowered" and "raised" are applied to different objects, each with its own meaning.”

Close Reading

8. Read the text again carefully. Match each programming paradigm with its key characteristics.

<i>Paradigm</i>	<i>Definition</i>
1. <i>Procedural Programming</i>	a) Uses objects that encapsulate data and behaviour
2. <i>Object-Oriented Prog.</i>	b) Uses pure functions and avoids shared state
3. <i>Functional Programming</i>	c) Uses a top-down structure and instructions step by step

9. Read the following statements and determine whether they are True or False.

If the statement is **false**, explain **why** based on the information in the text.

1. Procedural programming (PP) uses a top-down approach and relies on procedures or routines.
2. In Object-Oriented Programming (OOP), a class can be considered as a method within an object.
3. Syntax in programming is akin to grammar in human language, while semantics defines the meaning of these structures.
4. A syntactically correct program will always produce the expected results, regardless of its semantics.
5. A programming paradigm is the same as a specific programming language.
6. In procedural programming, the program is built from objects and their interactions.
7. Functional programming avoids shared state and side effects.
8. A method in OOP is similar to a procedure in procedural programming.

9. Syntax in programming defines how code should look and be structured.
10. "Colourless green ideas sleep furiously" is an example of correct syntax but unclear semantics.
11. Programming languages and natural languages face similar challenges in structure and meaning.
12. Functional programming relies on changing values and using global variables.

10. Answer the following questions:

1. What is a pure function?
2. Why is shared state problematic?
3. What is the main focus of object-oriented programming (OOP)?
4. What is the main focus of functional programming (FP)?
5. Why is understanding semantics important in programming?

Grammar for Revision The most common intermediate phrasal verbs 1-30 (Part 1)

Phrasal verbs 1-10

BE OVER	-If something is over, it has finished.	Example: <i>The meeting is over, you can go now.</i>
BREAK DOWN	-If a car or machine breaks down, it stops working.	Example: <i>Our car broke down on the way to the airport.</i>
BREAK IN / BREAK INTO	-To enter a building by force (usually to steal).	Example: <i>Someone broke into the office last night.</i>
BREAK UP	-To end a romantic relationship or marriage.	Example: <i>They broke up after five years together.</i>
BRING UP	-To raise and influence a child until they are grown.	Example: <i>She was brought up by her grandparents.</i>
CALL FOR	-To go and collect someone from where they are.	Example: <i>I'll call for you at 7 p.m. before the movie.</i>
CARRY ON	-To continue doing something.	Example: <i>Please carry on working while I'm away.</i>
CARRY OUT	-To perform or complete a task or instruction.	Example: <i>The engineers carried out a safety inspection.</i>
CHECK OUT	-To leave a hotel or guest house after paying.	Example: <i>We checked out at 11 a.m. and headed to the airport.</i>
COME ON	-Used to hurry someone or encourage action.	Example: <i>Come on, we're going to be late!</i>

11. Complete the sentences with the correct phrasal verb from the list above.
Use the correct form (e.g. past simple).

1. We had to stop in the middle of the highway because our car suddenly _____.
2. The final test will _____ all the procedures we've learned so far.
3. They were sad to _____ after three years of dating.
4. Can you _____ me at 6 p.m.? I'll be ready by then.
5. When will the concert finally _____? It's getting late.
6. Please don't stop — just _____ while I go grab a coffee.
7. She was _____ by her aunt after her parents moved abroad.
8. The thieves tried to _____ through the back window.
9. We need to _____ of the hotel before noon.
10. _____, we don't have all day to finish this task!

Phrasal verbs 11-20

◇ CROSS OUT	If you cross out words on a page, you draw a line through them, usually because they are wrong.	▪ Example: Please cross out the incorrect answer.
◇ CUT UP	If you cut something up, you cut it into several pieces.	▪ Example: He cut up the paper into tiny pieces.
◇ DEAL WITH	When you deal with something or someone that needs attention, you give your attention to them and solve a problem or make a decision.	▪ Example: We need to deal with this issue before it grows.
◇ DEPEND ON (situation)	If something depends on something else, it is affected or determined by that thing.	▪ Example: Our launch date will depend on the test results.
◇ DEPEND ON (person)	If a person depends on another person, they need their support or help to exist or be okay.	▪ Example: Many people depend on public transport to get to work.
◇ END UP	If you end up in a place or situation, it means that you unexpectedly arrive there after a series of events.	▪ Example: We ended up taking a completely different route.
◇ FILL IN	If you fill in a document or form, you write the necessary information in it.	▪ Example: Please fill in your details on the registration form.
◇ FILL UP	If you fill up a container or tank, it becomes full.	▪ Example: I need to fill up the car before our trip.
◇ FIND OUT	If you find something out, you learn or discover something unknown.	▪ Example: I just found out that we have a new manager.
◇ GET ALONG	If you get along with someone, you have a friendly relationship.	▪ Example: Luckily, I get along well with my colleagues.
◇ GET BACK	If you get back (to a place, situation, or activity), you return to it.	▪ Example: What time did you get back last night?

12. Complete the sentences with the correct phrasal verb from the list above.

Use the correct tense.

1. You need to _____ this form before the interview.
2. I accidentally _____ my old photos while cleaning the drawer.
3. Let's _____ the tasks before the meeting starts.
4. Don't worry if you make a mistake — just _____ the wrong word and write again.
5. We _____ at a small café instead of the restaurant we planned.
6. I need to _____ the gas tank — we're almost empty.
7. I'll _____ what time the event starts and text you.
8. She doesn't _____ with her boss — they argue all the time.
9. We'll _____ to the office around 6 p.m., traffic depending.
10. Everything will _____ how quickly we finish the testing phase.
11. Many elderly people _____ their neighbours for support.

Phrasal verbs 21-30

◇ GET DOWN	- If you get down, you lower your body until you are sitting, kneeling, or lying on the ground. Also, if you get something down, you write something.	▪ Example: He got down on one knee and proposed.
◇ GET IN	- If you get in, you enter a place, especially when it's difficult.	▪ Example: I couldn't get in because the door was locked.
◇ GET OFF	- When you get off (a bus, a train, etc.), you descend from that vehicle.	▪ Example: She got off the bus near the station.
◇ GET ON (continue/start doing)	- If you get on with something, you start doing it or continue doing it.	▪ Example: Let's get on with the presentation.
◇ GET ON (progress)	- If you say how someone is getting on, you're talking about how well they are doing.	▪ Example: How are you getting on with your new job?
◇ GET RID OF	- If you get rid of something or someone, you remove it/them so you don't have to deal with them anymore.	▪ Example: I need to get rid of this old laptop.
◇ GIVE BACK	- If you give something back, you return it to the person who gave it to you.	▪ Example: She gave back the book after reading it.
◇ GIVE IN (submit work)	- If you give in a piece of work, you hand it to a person of authority.	▪ Example: I gave in my essay just before the deadline.
◇ GIVE IN (surrender)	- If you give in, you surrender or abandon a fight or argument.	▪ Example: After a long debate, he finally gave in .
◇ GIVE OUT	- If you give out something, you distribute it to each person in a group.	▪ Example: The teacher gave out the exam papers

12. Complete the sentences with the correct phrasal verb from the list above.

Use the correct tense.

1. I knocked, but I couldn't _____ — the door was jammed.
2. The printer is not working, can someone _____ the handouts manually?
3. The teacher asked us to _____ our assignments before Friday.
4. We need to _____ all the old files on the server — they're taking up space.
5. She _____ the wrong bus and ended up on the other side of town.
6. I don't know how you _____ with all that pressure — well done!
7. After hours of arguing, he finally _____ and agreed to our proposal.
8. Please _____ from the table and sit on the floor for the activity.
9. I borrowed this pen — I'll _____ it after class.
10. I have a lot to do, so I should _____ with writing this report.

Speaking

 **13. Work in pairs. Take turns interviewing each other.**

1. Which paradigm (procedural, OOP, or functional) do you find easier to **deal with**? Why?
2. Did you ever have to **carry out** a project using both frontend and backend?
3. What helps you **keep on** learning new things in programming?
4. Do you know someone who was **brought up** around technology? How did that influence them?
5. Which language do you prefer — Python, Java, etc.? How did you **find out** which one suited you?
6. What motivates you to **carry on** even when a bug takes hours to fix?

14. Choose three phrasal verbs from the list above. Create your own **questions** using those verbs.

 Then, **ask your partner** and discuss their answers.

Example:

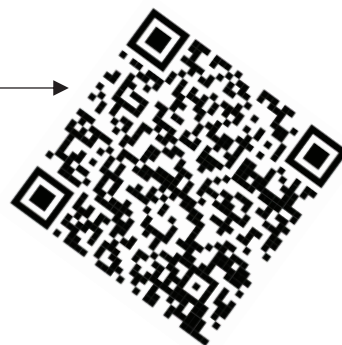
Phrasal verb: **break down**

Question: *Have you ever had a project that completely **broke down**? What happened?*

 **Extra Online Practice**

Vocabulary:

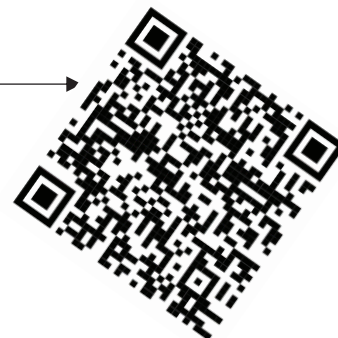
Quizlet – Unit 3.1 Flashcards & Learn



Grammar:

B1 Phrasal verbs 1-30

Exercises 1–3 on Test-English.com



☒ Recommended activity before starting Lesson 2:

Play Quizlet Live (INDIVIDUAL MODE, 3 TIMES)


using vocabulary from Unit 3.1:




 <https://quizlet.com/ua/932616895/unit-31-flash-cards/?i=j03j6&x=1jqt>

Lesson 2

Comparing and contrasting popular programming languages: Python, Java, JavaScript, C++, C#

Lead-in 1 “Which Language Would You Choose?”


 Look at the following real-world scenarios and choose which programming language (Python, Java, JavaScript, C++, C#, etc.) you think would be the best fit.

 **Use Mentimeter for Online Voting**
 **Organize an interactive class vote on programming languages using Mentimeter.**
 **Link for Teacher:**
<https://www.mentimeter.com/app/presentation/alkn11z9azh1bzvnqwdup7mxuh2qfz42/edit?source=share-invite-modal>




Instructions for the teacher:

1. Click “**Sign up**” or **log in** to Mentimeter.
2. Press “**Copy to My Presentations**”.
3. Go back to “**My Presentations**” in your Mentimeter dashboard.
4. Open “**Which Language Would You Choose?**”.
5. Click “**Share**” → “**Participation Link**” and send that link to your students.
6. Press “**Present**” to start the interactive voting session.

Scenarios:

1. You want to build a fast, high-performance video game. 
2. You’re designing a dynamic, interactive web interface.
3. You’re analysing huge datasets in a research lab.
4. You’re developing a mobile app for Android phones.
5. You’re writing code for a bank’s secure transaction system.

Pair Discussion

-  “Why did you choose that language?”
-  “Do you already know any of them?”
-  “Which language do you think is the hardest or most powerful?”

Skimming:

2. Choose the Best Summary. Read the text quickly. Which option below best summarizes the main idea of the text?

- A. It gives a detailed guide on how to code using JavaScript and Python.
- B. It compares five popular programming languages, describing their features, typical use cases, and strengths.
- C. It explains why JavaScript is the only language suitable for both frontend and backend development.
- D. It gives a historical overview of the development of programming languages from the 1980s to today

3.2.

Today's Programming Toolbox

In this section, we will explore five of the most widely adopted programming languages: JavaScript, Python, Java, C#, and C++. Each of these languages has played a critical role in shaping modern software development and is particularly well-suited for specific tasks and domains.

JavaScript is a dynamic, high-level programming language most commonly associated with client-side web development. Its primary role is in creating interactive, real-time user interfaces. Structurally, JavaScript employs elements such as **variables**, **functions**, and **control structures**, and shares syntactical similarities with C and Java.

Being an object-oriented language, JavaScript organizes data and behaviour into objects, enabling modular and reusable code. One of its key advantages is that it runs directly in the browser, eliminating the need for server-side execution in many cases.

Beyond browser use, JavaScript is also popular on the server side thanks to environments like Node.js, allowing developers to use the same language throughout

the full technology stack. It's also central to building single-page and progressive web applications (PWAs), as well as browser games.

Frameworks like Angular, React, and Vue.js, along with libraries such as jQuery, provide additional tools that simplify development by offering reusable components and efficient functionality.

Python is a versatile, general-purpose language known for its readable and minimalist syntax. It's especially popular in fields like data science, machine learning, and scientific computing, thanks to its **vast** ecosystem of specialized libraries such as NumPy, pandas, scikit-learn, TensorFlow, and PyTorch.

These tools support large-scale **computation**, model training, and data visualization, making Python a top choice for developers and data scientists alike. Its consistent syntax, extensive community support, and breadth of applications continue to make Python a **cornerstone** in both academia and industry.

Java. Originally released in 1995, Java is a general-purpose, object-oriented programming language developed by Sun Microsystems (now Oracle). Its guiding principle, "write once, run anywhere," reflects its platform independence via the Java Virtual Machine (JVM).

Java automates memory management through built-in **garbage collection**, allowing developers to focus on higher-level logic rather than manual memory handling. Like C++ and C#, Java uses syntax based on classes, methods, and variables, aiming for clarity and maintainability.

Commonly used in enterprise environments, Java supports robust backend services, mobile applications (especially Android), and even scientific and desktop applications.


C#. Developed by Microsoft, C# is part of the .NET ecosystem and is widely used for building Windows desktop applications and video games. Like Java, it is object-oriented, but it also incorporates features such as **generics**, **delegates**, and built-in garbage collection, making it both modern and efficient.

C# provides strong type-checking and is integrated with the extensive .NET framework, which includes powerful tools and libraries for rapid development. Its syntax is intuitive for those familiar with Java or C++.

This language is especially favoured in the game development industry, notably for scripting in the Unity engine. It also supports server-side and cloud-based application development.

C++ is a high-performance, low-level programming language derived from C, designed for **fine-grained control** over system resources. It allows direct memory access and is commonly used in areas where speed and efficiency are critical, such as operating systems, game engines, and simulations.

Though its syntax is more complex than higher-level languages, it includes powerful object-oriented features like classes and inheritance. Developers must manually manage memory, which gives them deeper control but requires precision and discipline.

Its ability to optimize system-level performance continues to make C++ a go-to language for performance-critical software. 

Close Reading

- 3. Read the text carefully. Then decide whether each statement below is true or false according to the information in the text.**
1. JavaScript is primarily used for server-side development.
 2. Python is often used in data science due to its extensive libraries and frameworks.
 3. Java's automatic memory management is handled by the developer, not the Java Virtual Machine (JVM).
 4. C# is commonly used in the development of mobile games.
 5. C++ is well-suited for high-performance applications like video games and operating systems.
 6. JavaScript code can run directly in web browsers.
 7. C++ is easier to learn than Python due to its simple syntax.

4. Answer the Questions

1. **What is a low-level language?**

- a) A programming language that is closer to machine code and provides fine control over system hardware.
- b) A language used mainly for scripting web pages.
- c) A language that uses natural language for programming.

2. **What are the typical uses of low-level languages?**

- a) Building websites and mobile apps.
- b) System programming, device drivers, and operating system development.
- c) Writing documents and editing images.

3. **What is a high-level language?**

- a) A language that provides more abstraction from the machine code, making it easier to write and understand.
- b) A language only used for writing complex mathematical equations.
- c) A language used exclusively for graphic design.

4. **Why are high-level languages considered easier to learn?**

- a) They have syntax and commands that are closer to human language, reducing complexity.
- b) They require knowledge of the machine's architecture.
- c) They use only numbers for coding.

5. **Which language from the lesson is best suited for data science and machine learning?**

- a) JavaScript
- b) Python
- c) C++

6. **What is garbage collection in programming?**

- a) A feature that manually removes unused code written by the programmer
- b) An automatic process that reclaims memory occupied by objects no longer in use
- c) A way to collect data from multiple sources for analysis

Vocabulary

5. Match the highlighted words in the text with their definitions.

- | | |
|-------------------------|--|
| 1. variables | a. a feature that allows types to be defined more flexibly, providing type safety without committing to specific data types. |
| 2. functions | b. blocks of code designed to perform specific tasks. |
| 3. control structures | c. Constructs used to control the flow of a program (e.g., loops, conditionals). |
| 4. vast | d. elements in a program that store data values. |
| 5. computation | e. extensive in size, amount, or scope. |
| 6. garbage collection | f. the ability to manage and manipulate system resources or program details with high precision. |
| 7. generics | g. the automatic process of reclaiming memory by removing objects that are no longer in use. |
| 8. delegates | h. the process of performing mathematical or logical operations. |
| 9. fine-grained control | i. something of great importance on which everything else depends |
| 10. cornerstone | j. types that represent references to methods with a specific parameter list and return type |

6. 🖱️ Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3https://quizlet.com/946812749/match?funnelUUID=f78b951e-f0b3-4e34-9581-e7cf5682cbf1>



7. Complete the sentences with the words from ex. 5

variables, functions, control structures, vast, computation,

garbage collection, generics, delegates, fine-grained control, cornerstone

1. In most programming languages, data is stored in _____ that can be updated or accessed as needed.
2. _____ like `if`, `else`, and `while` help determine the logical flow of a program.
3. Mathematical _____ such as addition and multiplication are processed very quickly in modern CPUs.
4. One of the key benefits of C++ is the level of _____ it provides over memory and system resources.
5. Python offers a _____ range of libraries for data science, from visualization to machine learning.
6. _____ are reusable blocks of code that take inputs and return outputs.
7. In managed languages like Java, _____ frees up memory that is no longer being used.
8. The concept of _____ allows developers to create flexible, reusable code without tying it to specific types.
9. In C#, _____ are used to pass methods as parameters to other methods, enhancing flexibility.
10. Understanding how to structure code is the _____ of successful software development.

8. Choose the correct term from ex. 5 to complete each sentence.

1. Loops and conditional statements are examples of _____ used to direct the flow of a program.
 - a) variables
 - b) control structures
 - c) delegates
2. In Python, memory management is handled by _____, which helps avoid memory leaks.

- a) computation
 - b) garbage collection
 - c) functions
3. One of the _____ of modern software is security—everything depends on getting it right.
- a) generics
 - b) cornerstone
 - c) variables
4. When you want to reuse a block of code to perform a specific task, you usually write a _____.
- a) function
 - b) variable
 - c) delegate
5. C++ is a language that gives developers _____ over system memory and performance.
- a) fine-grained control
 - b) garbage collection
 - c) computation
6. Java supports _____, which allow classes and methods to work with any data type.
- a) delegates
 - b) generics
 - c) control structures
7. Developers often work with a _____ amount of information when dealing with big data.
- a) fine-grained
 - b) vast
 - c) variables
8. In many programming languages, _____ are symbolic names assigned to data values.
- a) functions
 - b) variables
 - c) generics
9. Tasks like sorting or searching through data rely on efficient _____.
- a) garbage collection
 - b) computation
 - c) delegates
10. In C#, _____ allow methods to be passed as parameters and called dynamically.
- a) delegates
 - b) functions
 - c) control structures

Grammar for Revision The most common intermediate phrasal verbs 1-30 (Part 2)

Phrasal verbs 31-40

GIVE UP	To stop doing or using something.	■ Example: I decided to give up coffee for a while.
GIVE WAY	To let another vehicle go before you.	■ Example: You must give way at this intersection.
GO FOR	To try to achieve something.	■ Example: She's going to go for the promotion.
GO OFF	To stop working, make a loud noise, or explode.	■ Example: The alarm went off at 6 a.m.
GO ON	To continue or to be happening.	■ Example: What's going on in the next room?
GO OUT	To leave home for fun or be in a relationship.	■ Example: They love to go out on Fridays.
GO WITH	To match well in style or taste.	■ Example: That jacket goes with your shoes.
GO TOGETHER	To look or work well together.	■ Example: Blue and gold go together beautifully.
GROW UP	To become an adult.	■ Example: He grew up in a small village.
HAND IN	To submit work or return something found.	■ Example: Don't forget to hand in your homework.

9. Complete the sentences with the correct phrasal verb from the list. Use the correct form where necessary.

- I'm trying to _____ sugar because it's affecting my energy.
- They're planning to _____ a walk after dinner.
- I forgot to _____ my assignment — the deadline was today!
- When did you two start _____? You look so happy together!
- The music suddenly _____ in the middle of the presentation.
- That bag really _____ your new shoes — great style!
- I wonder what's _____ in the meeting room — they've been in there for hours.
- She decided to _____ the job even though it was a big risk.
- You need to _____ to traffic on the main road before turning.
- I want to be a doctor when I _____.

Phrasal verbs 41-50

HAND OUT	To give something to each person in a group.	■ Example: The teacher handed out worksheets to the class.
HANG OUT	To spend time relaxing somewhere or with someone.	■ Example: We usually hang out at the cafe after school.
HANG UP	To end a phone call.	■ Example: He got angry and hung up on me.
HOLD UP	To cause delay.	■ Example: Sorry I'm late — traffic held me up.
KEEP IN	To make someone stay indoors.	■ Example: The teacher kept us in after class for talking.
KEEP ON	To continue doing something.	■ Example: She kept on asking the same question.
KEEP UP	To maintain progress or performance.	■ Example: You're doing great — keep it up!
KNOCK DOWN	To hit and cause someone to fall.	■ Example: A cyclist was knocked down by a car.
LIE DOWN	To move into a horizontal position.	■ Example: I need to lie down — I feel dizzy.
LOOK AFTER	To take care of someone or something.	■ Example: She looks after her little brother every day.

10. Complete the sentences with the correct phrasal verb from the list. Use the correct form where necessary.

1. A delivery truck almost _____ a pedestrian this morning.
2. Can you _____ my plants while I'm away?
3. Don't _____ — I still need to ask you something!
4. Even after several warnings, he _____ talking loudly.
5. Great job on the project! If you _____ this level of effort, you'll pass with distinction.
6. Heavy traffic might _____ the delivery by another hour.
7. I feel really tired — I think I'll just _____ for a while.
8. I usually _____ with my friends near the river on weekends.
9. Please _____ the flyers to everyone before the session starts.
10. The teacher had to _____ the whole class after school.

Phrasal verbs 51-60

LOOK FOR	- To try to find or get something.	■ Example: I'm looking for a new apartment.
LOOK FORWARD TO	- To feel happy about something that will happen.	■ Example: I look forward to meeting you soon.
LOOK OUT	- To warn someone of danger.	■ Example: Look out! There's a car coming!
LOOK UP	- To search for information.	■ Example: I looked up the word in the dictionary.
PASS ON	- To give something to someone else.	■ Example: Please pass this message on to Tom.
PICK UP	- To collect someone or something.	■ Example: I'll pick you up at 7 p.m.
PUT AWAY	- To return something to its proper place.	■ Example: She put away the groceries after shopping.
PUT DOWN	- To write something on paper.	■ Example: He put down his name on the list.
PUT OFF	- To delay or postpone something.	■ Example: They put off the meeting until next week.
PUT ON	- To wear clothes, glasses, or gain weight.	■ Example: She put on her coat and left.

11. Complete the sentences with the correct phrasal verb from the list. Use the correct form where necessary.

- I always _____ seeing my friends during the holidays.
- Can you _____ your name and email on this form?
- We had to _____ the appointment because of the storm.
- He didn't know the meaning, so he _____ the word in the dictionary.
- After the party, we helped _____ all the dishes.
- Don't forget to _____ your jacket — it's cold outside.
- She asked me to _____ the message to her supervisor.
- I've been _____ a part-time job for weeks now.
- I'll be late. Can you _____ the kids from school?
- _____! That ladder looks unstable!

Speaking

🗣️ 12. Work in pairs. Take turns interviewing each other.

1. **Have you ever tried to *give up*** a habit but found it really hard? What made it difficult?
2. If you could ***go for*** any opportunity right now—job, trip, or challenge—what would it be and why?
3. Who do you usually ***look after*** when someone in your family needs help?
4. Where do you and your friends usually ***hang out*** on the weekends?
5. Is there something you've been ***putting off*** lately? What's stopping you from doing it?
6. What's something you're really ***looking forward to*** this month?
7. Is there a skill or activity you're trying to ***keep up*** with regularly?
8. Do you remember the last time you had to ***pick someone up*** from somewhere? What happened?
9. Have you ever ***put on*** something unusual or funny for a costume party or event?
10. Can you think of a moment when someone had to shout "***Look out!***"? What happened?

13. Choose three phrasal verbs from the list above.

Create your own questions using those verbs. 🗣️ Then, ask your partner and discuss their answers.

Extra Online Practice

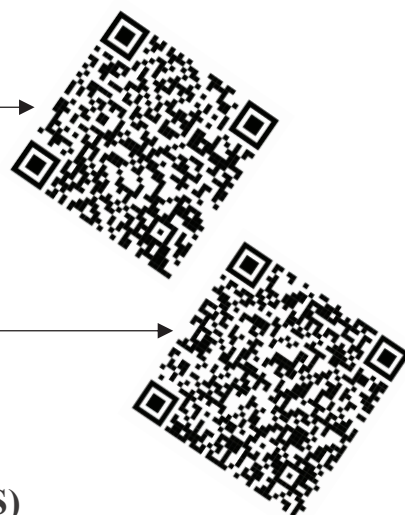
Vocabulary:

[Quizlet – Unit 3.2 Flashcards & Learn](#)

Grammar:

[B1 Phrasal verbs 31-60](#)

[Exercises 1–3 on Test-English.com](#)



☑ Recommended activity before starting Lesson 2:

Play Quizlet Live (**INDIVIDUAL MODE, 3 TIMES**)

using vocabulary from Unit 3.2:

🔗 <https://quizlet.com/ua/946812749/unit-32-flash-cards/?i=j03j6&x=1jqt>

Lesson 3

Data base

Lead-in 1. What's in a Database?

 **Work in pairs or small groups. Discuss the questions below:**

1. Where have you seen or used databases in real life (e.g. school, apps, websites, stores)?
2. Why do you think databases are important in software development?
3. Imagine you're designing an app—what kind of data would you need to store?
4. Which would you trust more: a spreadsheet or a proper database? Why?
5. Can a spreadsheet be called a database? ► Yes / No / Not sure

2. Work in pairs. Match sentences 1-4 to results a-d. Then rewrite the sentences using the words in brackets.



How to Read Decimal Numbers in English

1. Say **the whole number** before the dot.
2. Say **"point"** instead of the dot.
3. Say **each digit** after the point separately.



Examples:

2.5 → TWO POINT FIVE

7.03 → SEVEN POINT

ZERO THREE

0.9 → ZERO POINT NINE

15.47 → FIFTEEN POINT

FOUR SEVEN

! Do not read the digits after the point as a full number.

✗ FIFTEEN POINT FORTY-SEVEN → this is incorrect!

- | | |
|---|-------------------|
| 1 If we divide 8 by 2, we get 4. (divided by) | a) $8 + 2 =$ |
| 2 If we subtract 2 from 8, we get 6. (minus) | b) $8 - 2 =$ |
| 3 If we multiply 8 by 2, we get 16. (times) | c) $8/2 =$ |
| 4 The sum of 8 and 2 is 10. (plus) | d) $8 \times 2 =$ |

e.g. *8 divided by 2 is 4.*

3. Work in pairs. Each of you should write eight math problems, but don't show them to your partner. Then take turns reading your problems aloud for your partner to solve.

e.g. A: $[9/2=?]$ *What is 9 divided by 2?*

B: 4.5.

B: $[10 \times 4.6=?]$ *If you multiply 10 by 4.6, what do you get?*

A: 46.

A: ...

4. Read the sentences with words from the video. Guess the meaning of each **bold** word. Then write each word next to its definition.

A. The team **consists of** four Europeans and two Americans.

B. We are soon to count in **decimals**, measure our temperature in centigrade and, perhaps, drive on the right-hand side of the road.

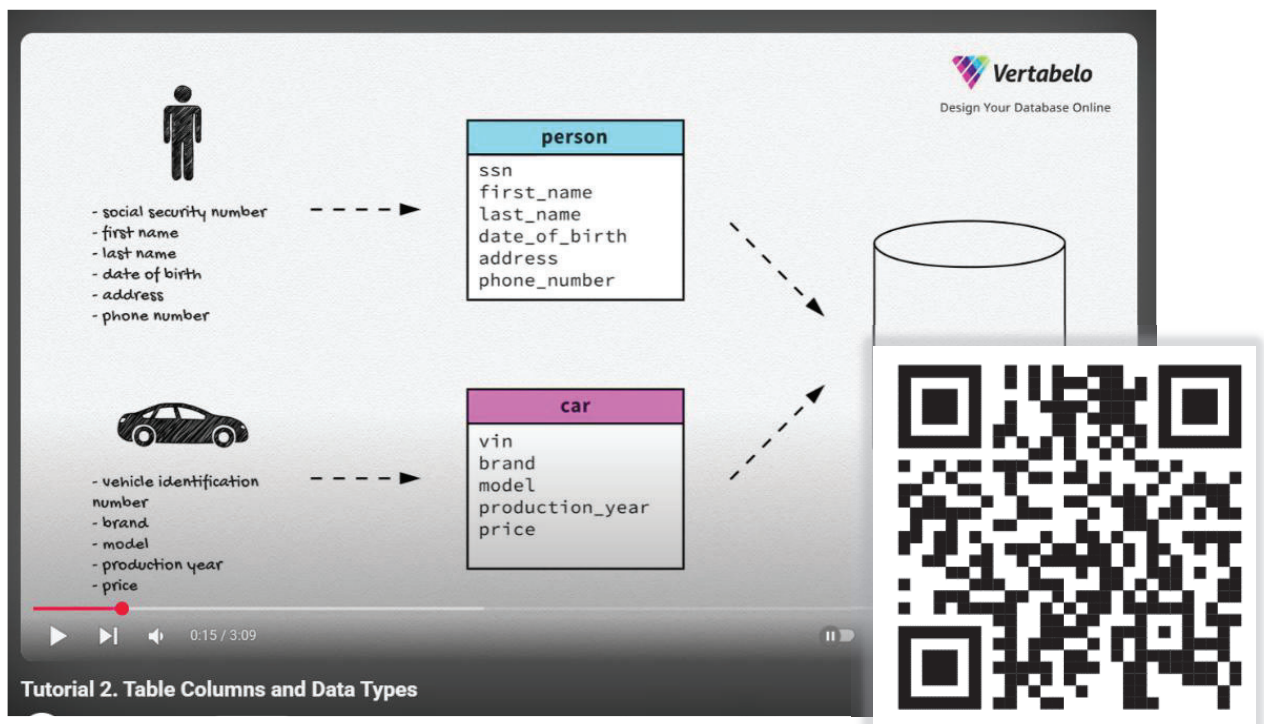
C. The numbers 5, 3, and 0 are **integers**.

D. If you type in the search **string** "ing", the computer will find all the words containing "ing".

1. _____ (n) a whole number and not a fraction:
2. _____ (phr v) means to be made up of, or composed of something.
3. _____ (n) a usually short piece of text consisting of letters, numbers, or symbols
4. _____ (adj) a number expressed using a system of counting based on the number ten.

5.  Watch the video Tutorial 2. Table Columns and Data Types

<https://youtu.be/Zpi2GLJgJzI> and complete the sentences below using the words from the box.



person

- ssn
- first_name
- last_name
- date_of_birth
- address
- phone_number

car

- vin
- brand
- model
- production_year
- price

Tutorial 2. Table Columns and Data Types

Figure 5 Tutorial 2. Table Columns and Data Types[<https://youtu.be/Zpi2GLJgJzI>]

Word Box:

columns data types numeric products money day
timestamp true false long text inexact

1. The structure of a table consists of _____. ⌚ [0:28]
2. Columns can have different _____, such as strings, _____, and date and time. ⌚ [0:34]
3. With the integer data type, you can store a number of _____, people, or numerical identifiers. ⌚ [1:01–1:09]
4. A float type holds _____ point numbers and is useful when exact precision is not required. ⌚ [1:12]
5. The decimal type provides high precision and is ideal for storing _____. ⌚ [1:21–1:23]
6. The date type stores a calendar date without time of _____, useful when exact time is not needed. ⌚ [1:42–1:45]
7. The _____ data type includes both date and time—use it when tracking the exact moment something happened. ⌚ [1:50–1:53]
8. A boolean type stores logical values such as _____ or _____. ⌚ [1:58]
9. A binary large object (BLOB) stores files such as images or _____. ⌚ [2:08]

6. 🎮 Kahoot Quiz: “Database Table Columns and Data Types”

🔗 **Link for the teacher** to host the game:


<https://create.kahoot.it/share/database-table-columns-and-data-types/0f1a2dbc-1b9f-469b-803a-e0dffeaddcc9>



7. Reading Task: Database Design Basics

 Go to the official Microsoft Support page and read the article:

 **Database Design Basics**

 **Task:** After reading, complete the exercise below to check your understanding.

8. True or False: Justify Your Answer

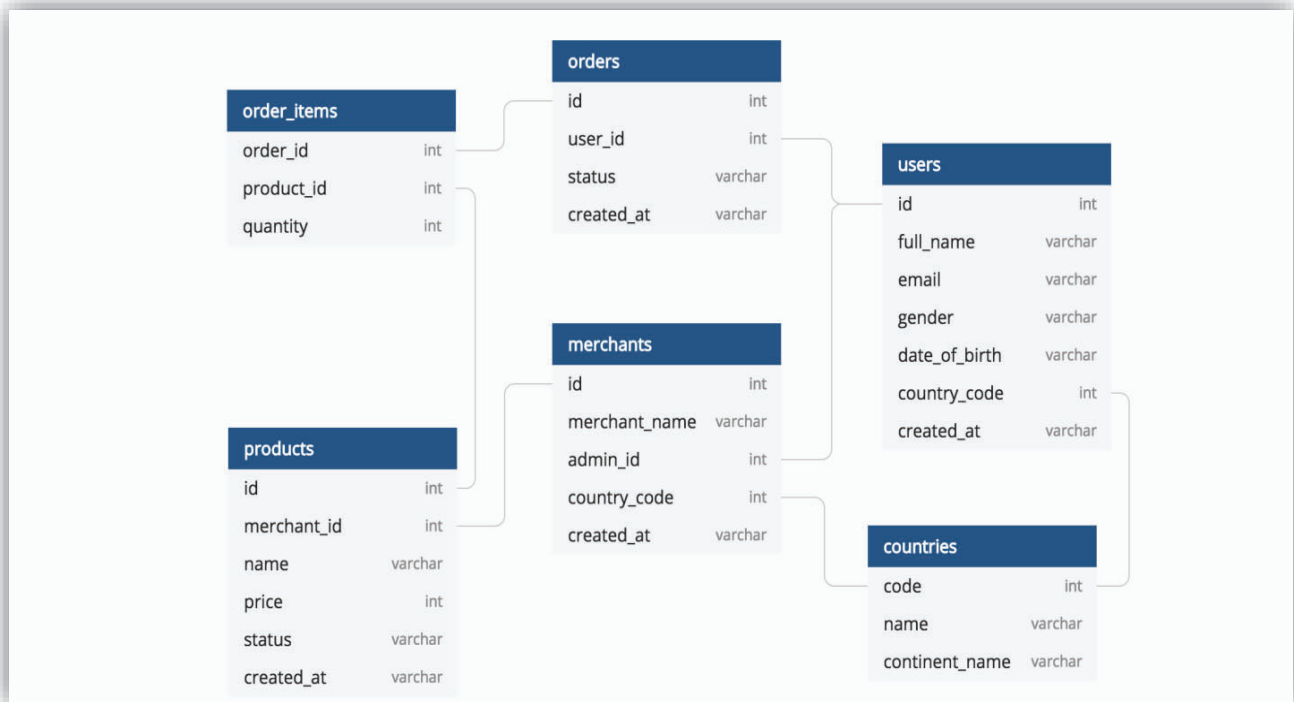
Read each statement carefully. Decide if it is **True** or **False** based on the article, and **briefly explain your reasoning** using information from the text.

1. A good database design accommodates both data processing and reporting needs.
2. Duplicate information in a database is considered beneficial because it ensures data integrity.
3. Redundant data helps save disk space in a database.
4. When designing a database, it is best to place all information in a single table for simplicity.
5. The primary key is a column used to uniquely identify each table.
6. To ensure the accuracy of information, a database should be divided into subject-based tables.
7. A well-developed mission statement for a database project is unnecessary.
8. Breaking down each piece of information into its smallest useful parts is not important in database design.
9. Database design does not need to consider international data if the database is used only domestically.
10. Deleting a product record in a table that contains both product and supplier information will only delete facts about the product and not the supplier.



9. Look at the Database diagram and answer the questions.

1. How many tables are in the database?
2. What type of relationships are between the tables?
3. What types of keys are in the table?



10. Discussion

Scenario: Imagine you are tasked with designing a database for a small online store.

Task:

- What major **entities** or **subjects** (e.g., customers, orders, products) would you include in your database?
- How would you **organize the information into tables**?
- What **columns** (fields) would each table contain?

◆ **Discuss** your ideas in pairs or small groups. Be prepared to share your table structure and explain your design choices.

11. Fill in the gaps using the appropriate words from the list below. Each word is used once.

Word Bank:

decimals, column, numeric, record, integers, row, field, divided by, string, redundant, primary key, subtract, multiply, consists of

A database is an organized collection of data. Each table in a database is made up of rows and columns. A (1)_____ refers to a horizontal set of data, while a (2)_____ is a vertical set. Each (3)_____ in a table stores a specific piece of information.

In databases, different data types are used to store information. (4)_____ data types, such as (5)_____, are used to store numbers without fractional parts, while (6)_____ can store numbers with fractional parts. Text data is stored in a (7)_____ format.

Each (8)_____ in a table is uniquely identified by a (9)_____, which helps prevent (10)_____ data.

In mathematics, operations like (11)_____, (12)_____, and (13)_____ are often performed on (14)_____ or (15)_____ values. For example, the result of 15 (16)_____ 3 is 5.

12. 🖱️ Play the Quizlet Match game by following this link:

<https://quizlet.com/948094076/match?funnelUUID=7b230eb9-5c12-4a89-ac8f-5c206e9e61cc>



🗨 13. SPEAKING TASK: Design & Discuss a Database

Work in pairs (or small groups of 3).

1. Imagine your team is hired to **design a database for a specific organization** (choose from the list below or create your own).
2. Use the prompts to guide your discussion and take turns speaking.
3. Present your ideas to the class afterward.

🧩 Choose a scenario:

- A small online bookstore
- A gym membership system
- A school student database
- A movie streaming service
- A travel agency

💡 Discuss the following:

1. What **tables** will you create (e.g., Users, Orders, Products)?
2. What **fields/columns** will each table include?
3. What will be the **primary key** in each table?
4. How will the tables be **linked** (relationships)?
5. How will you avoid **redundant** data?
6. Will you include **numeric**, **string**, or **date/time** types? Why?

🔍 Extra Online Practice

Vocabulary:

Quizlet – Unit 3.3 Flashcards & Learn

Grammar:

B1 Phrasal verbs (Part 3)

Exercises 1–3 on Test-English.com



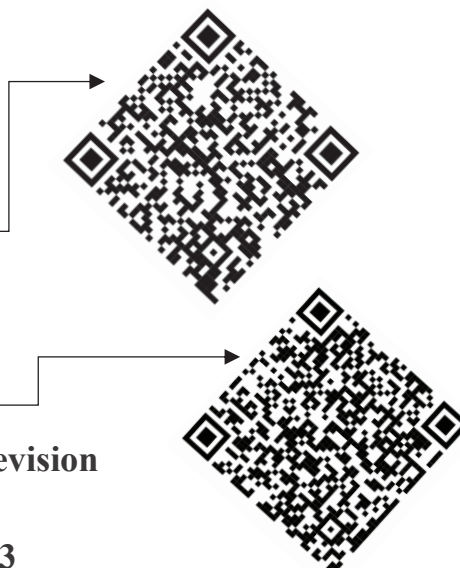
Recommended activity before starting **Revision**

Unit 3:

Play Quizlet Live (**INDIVIDUAL MODE, 3 TIMES**) using vocabulary from Unit 3.3:



<https://quizlet.com/ua/948094076/unit-33-flash-cards/?i=j03j6&x=1jqt>



Vocabulary Revision

Unit 3

 1. WORK IN PAIRS. TAKE TURNS PLAYING THE ROLES OF **DESCRIBER** AND **GUESSER** USING VOCABULARY FROM UNIT 3.

◇ Detailed instructions can be found on page 37

Term	Definition
decimals	a number expressed using a system of counting based on the number ten
integers	(n) a whole number and not a fraction
string	(n) a usually short piece of text consisting of letters, numbers, or symbols
column	A vertical series of cells in a table.
numeric	describes data that consists of numbers.
row	a horizontal group of cells in a worksheet identified by numbers
field	a single characteristic of data that appears in a table as a column
record	information provided to a journalist that can be released and attributed by name to the source
redundant	(adj.) extra, excess, more than is needed; repetitive;
primary key	a field that uniquely identifies a record in a table
divided by	÷ or /
subtract	to take away
multiply	The operation of repeated addition of the same number.
consists of	is made up of
variables	Elements in a program that store data values.
functions	Blocks of code designed to perform specific tasks.
control structures	Constructs used to control the flow of a program (e.g., loops, conditionals).
vast	Extensive in size, amount, or scope.
computation	The process of performing mathematical or logical operations.

garbage collection	The automatic process of reclaiming memory by removing objects that are no longer in use.
generics	A feature that allows types to be defined more flexibly, providing type safety without committing to specific data types.
delegates	Types that represent references to methods with a specific parameter list and return type.
fine-grained control	The ability to manage and manipulate system resources or program details with high precision.
inline	Within a program or procedure, often referring to code that is placed directly in the sequence of execution.
sequence	A series of related events, actions, or items that follow each other in a particular order.
encapsulating	The act of containing or covering something within another, often used to describe the grouping of data and methods within objects in OOP.
property	A characteristic or attribute that belongs to something, especially in programming, it refers to data associated with an object.
blueprint	A detailed plan or design that serves as a guide for constructing something.
immutable	Something that is unchangeable or cannot be altered once it has been created.
intricate	The quality of being complex and detailed, often requiring careful attention.
harness	The act of using or controlling something effectively, such as power or resources.
akin	Similar in nature or function, having a resemblance to something else.
incongruity	A deviation from what is expected, leading to a sense of inconsistency or contradiction.
be flawed	To contain defects or imperfections, not being perfect.
incongruity	A deviation from what is expected, leading to a sense of inconsistency or contradiction.
be flawed	To contain defects or imperfections, not being perfect.

2. *THREE-SENTENCE CHALLENGE.* Each participant **selects three words** from the shared word list and forms three sentences using these words. Work in pairs. Read your sentences to your partner, replacing the chosen word with "gap" while reading. Your partner needs to guess the missing word.

3. *PLAY THE QUIZLET LIVE GAME (TEAM MODE)*

to review the vocabulary learned in *UNIT 3*

<https://quizlet.com/ua/948371787/unit-3-33-32-31-flash-cards/?i=j03j6&x=1jqt>



4. TAKE the UNIT 3 VOCABULARY QUIZLET TEST to check your understanding of key terms from this UNIT.

<https://quizlet.com/948371787/test?answerTermSides=6&promptTermSides=6&questionCount=30&questionTypes=15&showImages=true>



5. 🎲 VOCABULARY GAME: ALIAS

Play a fun team-based vocabulary challenge!

♦ See full game instructions on page 37.

Get ready to explain, guess, and compete using key terms from Unit 3!

For Printable Vocabulary Materials for the Game “Alias,” go to page 232.

4

UNIT

SOFTWARE DEVELOPMENT PROCESS OVERVIEW



This unit provides an introduction to modern software development practices and tools. Students will explore key methodologies such as Test-Driven Development (TDD) and Behaviour-Driven Development (BDD), and understand the importance of version control systems like Git, GitHub, and GitLab. The final lesson introduces DevOps practices, including CI/CD, which enable faster, more reliable software delivery.

Unit overview

4.1 Introduction to Software Development Methodologies: TDD and BDD

Lesson outcome: Learners can describe the principles and benefits of Test-Driven and Behaviour-Driven Development and discuss when to use each approach.

Skills practised

Skim reading

Vocabulary: key terms related to development approaches

Grammar: quantifiers: much, many, a lot, little, few, some, any, no

4.2 Understanding Version Control: Git, GitHub, GitLab

Lesson outcome: Learners can explain the role of version control systems in collaborative development and describe how Git and cloud repositories function.

Skills practised:

Reading (for gist and detail)

Speaking: pair discussions, collaborative tasks

Vocabulary: technical terms related to Git, GitHub, GitLab

Grammar: quantifiers: all, both, both of, all of

4.3 DevOps and CI/CD Fundamentals

Lesson outcome: Learners can define DevOps practices and explain the benefits of Continuous Integration and Continuous Deployment in software development.

Skills practised:

Listening: identifying key information from a video (gist/detail)

Reading: understanding and interpreting a technical text on DevOps

Speaking: discussing DevOps practices using quantifiers (both/either/neither)

Vocabulary: key DevOps and CI/CD terminology

Grammar: quantifiers – both, either, neither

Lesson 1

Introduction to various software development methodologies: Test-Driven Development (TDD), Behaviour-Driven Development (BDD)

Lead-in 1 Then vs. Now – How Has Software Development Changed?

Discuss in pairs or small groups:

1. What do you think makes software successful — detailed planning or constant testing?
2. Why might writing long documentation BEFORE coding not always lead to the best results?
3. Imagine you're on a software team with a tight deadline — would you prefer to:
 1. Write full documentation before starting?
 2. Build and test features as you go?
 3. Something else?

2. Skim the text. Sort the statements below into the correct column:

☒ True for Agile / TDD / BDD or ☒ Not true (NT) / not mentioned (NM).

Statement	Agile	TDD	BDD	NT/ NM
1. Focuses on small, frequent releases				
2. Features are tested only after the code is written				
3. Requirements are discovered through user feedback				
4. All features must be fully tested before moving on				
5. Encourages writing tests before writing code				
6. Developers write stories in natural language to describe behaviour				
7. Testing is optional in TDD				
8. Agile encourages long-term upfront documentation				

4.1.


Introduction to Software Development

Methodologies: Agile and Beyond

Before agile development, companies spend tremendous efforts in preparing analysis and design documents. This is partly due to standards audits and partly due to the beliefs that “good software comes from good design documentation, and good design documentation comes from good analysis models.” These beliefs are true, but only partly. Many software engineers have experienced that in some cases it is impossible to determine the real requirements, or whether the design works, until the code is written and tested. In these cases, comprehensive documentation won't help and might be harmful because it gives the illusion that a working solution has been found. Comprehensive documentation also means less time is available to coding and testing, which are the only means in these cases to identify the real requirements and the needed design.

Agile software development emphasizes small, incremental releases and frequent iterations. Agile projects are developed and deployed in manageable increments, delivering use cases, user stories, or features iteratively. This approach provides several advantages: it makes project progress visible, allows users to learn a few new features at a time, encourages user feedback, and reduces the risk of project failure by focusing on smaller, manageable releases.


A key aspect of agile development is the frequent delivery of software products. Before agile, iterative processes like the Spiral and Unified Process also existed, but agile methods differ in their emphasis on frequent, small-scale releases. Various agile methods recommend different iteration lengths: for example, Dynamic Systems Development Method (DSDM) suggests two to six weeks, Extreme Programming (XP) uses one to four weeks, and Scrum defines iterations, known as sprints, lasting 30 days. Many software development teams favor two-week iterations as they provide a realistic timeframe for developers to complete specific tasks. The methodology presented in this context recommends iterations of two to four weeks.

Another principle of agile is to complete each **feature** fully before moving on to the next. This means that a feature must be 100% implemented and **thoroughly** tested before starting a new one. Test-driven development (TDD) and test coverage criteria play a crucial role in ensuring thorough testing. TDD requires that tests for each feature are written before implementation begins, encouraging programmers to fully understand the feature **upfront**. This allows for immediate and frequent testing. Test coverage criteria, such as 100% **branch** coverage, ensure that every branch of each conditional statement in the code is tested at least once, which many companies **adopt** as a standard for thorough testing. 

Vocabulary 3.

Match the highlighted words in the text with their definitions.

- | | |
|------------------|---|
| 1. tremendous | a) to accept or start to use something new |
| 2. due to | b) a part of something larger |
| 3. comprehensive | c) covering or including everything |
| 4. harmful | d) to be different from something in some way, vary |
| 5. emphasize | e) because of |
| 6. incremental | f) give special importance or prominence to (something) in speaking or writing. |
| 7. manageable | g) To be real. To be found; to occur. To stay alive. |
| 8. iterative | h) an important part of something |
| 9. exist | i) causing or likely to cause harm; damaging |
| 10. differ | j) increasing gradually by regular degrees or additions |
| 11. feature | k) a process that repeats a series of steps over and over until the desired outcome is obtained |
| 12. thoroughly | l) capable of being controlled, directed, or accomplished |
| 13. upfront | m) in a detailed and careful way |
| 14. branch | n) extraordinarily large in size or extent or amount or power or degree |
| 15. adopt | o) before goods or services are received |

4.  **Play the Quizlet Match game by following this link:**
<https://quizlet.com/948384872/match?funnelUUID=50ea8fc4-5a23-46df-8ae9-7bbebd91a000>



5. Complete the text below by filling in the blanks with the correct words from the box.

Word Bank:

iterative adopt manageable features harmful incremental due to thoroughly

1. Agile development emphasizes _____ delivery of small parts of the software, allowing frequent feedback and quick adjustments.
2. The new system offers several enhanced _____, including stronger security and a more intuitive user interface.
3. Agile methodologies follow an _____ process, where development steps are repeated and refined until the desired result is achieved.
4. Each feature must be tested _____ before developers move on to the next one.
5. The team decided to _____ an agile methodology to improve productivity and time management.
6. The project was delayed _____ unexpected technical issues that required urgent attention.
7. The original project plan was simplified to make it more _____ for the available team.
8. Even a minor bug can become _____ if not identified and addressed early in the development cycle.

Close Reading

6. Read the text again carefully. Match each methodology or concept with its description:

- | | |
|---|--|
| 1. Dynamic Systems Development Method (DSDM) | a. Uses one to four-week iterations. |
| 2. Extreme Programming (XP) | b. Requires that tests for each feature are written before implementation begins. |
| 3. Scrum | c. Suggests iterations lasting two to six weeks. |
| 4. Test-driven development (TDD) | d. Defines iterations, known as sprints, lasting 30 days. |

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3>

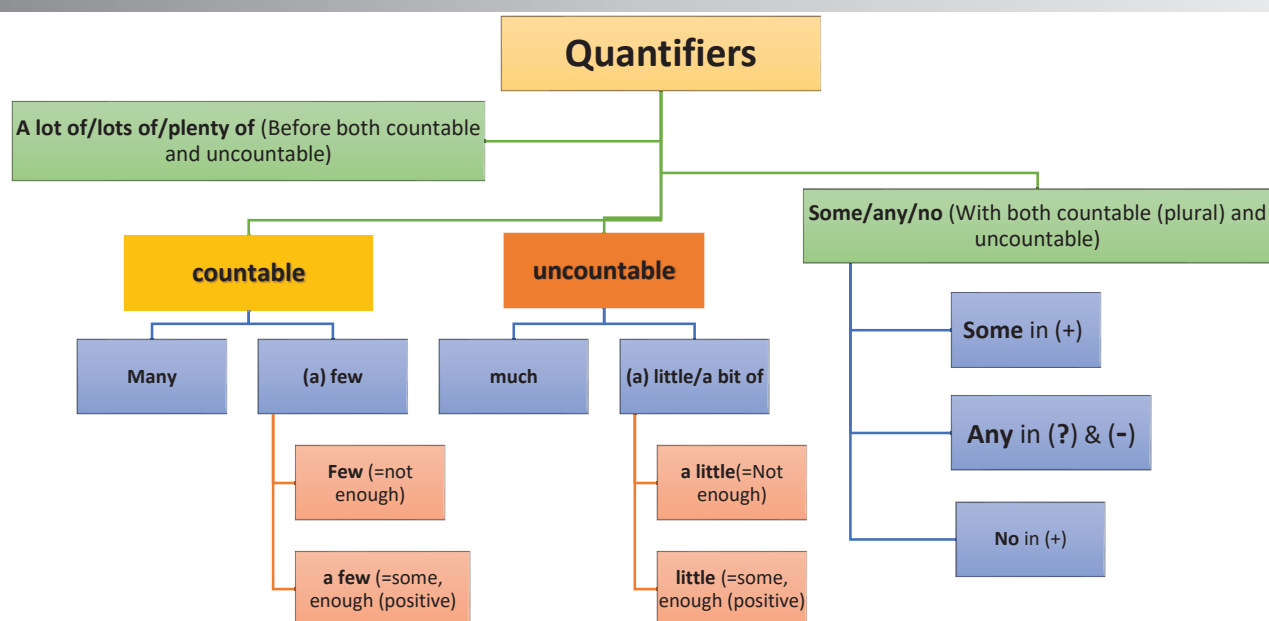
7. Answer the questions

1. What were two primary reasons companies focused on preparing extensive

analysis and design documents before the agile development approach?

2. Why might comprehensive documentation be considered harmful in certain software development cases?
3. What are the main benefits of adopting an agile software development approach?
4. How do agile methods differ from earlier iterative processes such as the Spiral and Unified Process?
5. What are some examples of iteration lengths recommended by various agile methodologies like DSDM, XP, and Scrum?
6. Why is it important in agile development to fully complete and test a feature before moving on to the next one?
7. How do Test-Driven Development (TDD) and test coverage criteria contribute to ensuring the thorough testing of software in agile methodologies?

Grammar: Much, many, a lot, little, few, some, any, no: Quantifiers



Much / Many

Many = plural countable nouns (in questions or negatives)

Much = uncountable nouns (in questions or negatives)

Examples:

- *I don't have many books left.*
- *There isn't much time to relax.*

A lot of / Lots of / Plenty of

Use before both countable and uncountable nouns (mostly in positive sentences)

Example:

- *She has a lot of friends and plenty of free time.*

A few / Few

A few (countable) = some, enough (positive meaning)

Few (countable) = almost none, not enough (negative)

Examples:

- *We have a few tasks to complete. (enough)*
- *Few experts attended the meeting. (almost none)*

A little / Little

A little (uncountable) = some, enough (positive)

Little (uncountable) = almost none, not enough (negative)

Examples:

- *She has a little patience left.*
- *There's little milk in the fridge.*

Some / Any / No

Some used in positive statements (and questions for offers/requests)

Any used in negatives and questions

No used in positive statements to mean zero

Examples:

- *I have some questions.*
- *Do you have any issues?*
- *There are no mistakes in the report.*

8. Choose the correct option from the two given in brackets.

1. There are only _____ people at the meeting today. (**few / a few**)
2. I have _____ time before my next call — let's grab a coffee. (**little / a little**)
3. We didn't find _____ useful information in that article. (**much / many**)
4. He made _____ mistakes in the report, but nothing serious. (**few / a few**)
5. She has _____ friends in London, so she visits often. (**a few / few**)
6. Do you have _____ milk left for tea? (**any / some**)
7. I don't have _____ money with me right now. (**much / many**)
8. They offered me _____ help with the project. (**some / any**)
9. There is _____ traffic this morning — we'll be on time! (**little / a little**)
10. I have _____ interest in that topic. I prefer history. (**no / any**)

9. Fill in the Blanks. Complete the sentences below using the correct quantifier:
(**much, many, few, little, some, any, no**)

1. There were only a _____ developers available to fix the bug over the weekend.
2. We didn't have _____ time left before the product launch.
3. Are there _____ open issues in the project backlog?
4. The team made _____ progress after switching to Agile.
5. _____ developers understand the full benefits of test-driven development.
6. I don't see _____ reason to rewrite the code from scratch.
7. We didn't encounter _____ serious bugs during testing.
8. The intern had very _____ experience with version control systems.
9. Did the client provide _____ feedback after the last sprint?
10. There is _____ documentation for this old module — we'll need to figure it out ourselves.

10. Choose the Correct Option to Make a Question

Complete the following questions by choosing the correct word: **much** or **many**, **some**, **any**, **no**, **few**, **little**.

1. How **many** / **much** programming languages have you tried so far? Which one do you use most often?
2. Do you think developers read **many** / **much** documentation before starting a new project?
3. How **little** / **few** planning do you usually do before writing code?
4. Are there **any** / **no** tools you can't imagine working without?
5. How **much** / **many** time do you usually spend debugging?
6. Have you ever had **too little** / **too few** time to test a feature properly?
7. Do you usually write **any** / **some** tests before coding?
8. Have you ever worked on a project with **no** / **any** clear requirements?
9. Are there **some** / **few** practices you always follow when developing software?
10. How **many** / **much** unit tests do you usually write for a small feature?

🗨️ 11. Work in pairs. Interview each other. Take turns asking and answering the questions from Ex. 9.

🔑 Extra Online Practice

Vocabulary:

Quizlet – Unit 4.1 Flashcards & Learn

Grammar:

Much, many, a lot, little, few, some, any, no: Quantifiers

Exercises 1–3 on Test-English.com



☑ Recommended activity before starting Lesson 2:

Play Quizlet Live (INDIVIDUAL MODE, 3 TIMES) using vocabulary from Unit 4.1:

🔗 <https://quizlet.com/ua/948384872/unit-41-flash-cards/?i=j03j6&x=1jqt>

Lesson 2

Explanation of version control systems: Git (local repositories), GitHub (cloud-based), and GitLab

Lead-in 1. Version Control in Real Life. Work in pairs or small groups:

Discuss the following scenario and questions.

Imagine you and your classmates are working on a group project. Each person is responsible for writing different parts of a shared document or program.

After a few days, everyone emails their parts to one person to compile. Some files have the same name, others have overlapping changes, and it's unclear who wrote what version last.

Questions for discussion:

1. What kinds of problems could happen in this situation?
2. How would you solve or prevent these problems without a version control system?
3. Have you ever experienced version mix-ups in group work (e.g., Google Docs, Word, or coding projects)? What happened?
4. What do you think a *version control system* does, and how might it help in this scenario?

Pre-Reading

2. Read the short descriptions below. Try to guess whether each one describes **Git**, **GitHub**, or **GitLab**. Work in pairs and discuss your reasoning.

1. A platform that lets developers create pull requests and manage issues.
2. A system used locally on a developer's computer to track code changes.
3. A cloud-based tool known for its large open-source community.
4. A version control tool that includes built-in CI/CD pipelines.
5. A tool that works offline and allows for branching and merging.

4.2.

Version Control in Modern Software Development: Git, GitHub, and GitLab

Version control systems play a crucial role in today's software development processes. They allow programmers to monitor code changes, handle different versions of a project, and work efficiently as a team. Git, a leading VCS, is especially popular. It uses a **distributed** architecture, which means every contributor keeps a complete local version of the repository. Developers typically start by **cloning** a remote repository, which creates a full copy on their local machine, enabling offline work and greater flexibility. Git tracks changes by creating **branches**, which enable developers to work on different features or fixes without affecting the main codebase.

In Git, once you've made changes, you add them to the **staging area** before you **commit** the changes to your local **repository**. A commit acts as a snapshot of the project at a particular point in time. Later, developers can **merge** their changes back into the main branch or other branches, resolving any **conflicts** that arise.

GitHub and **GitLab** are cloud-based platforms built around Git, providing additional features like project management, team collaboration, and **pull requests**, where developers submit changes for review before merging them into the main codebase. While both platforms offer similar features, GitHub is known for its large open-source community, and GitLab is appreciated for its built-in **Continuous Integration/Continuous Deployment (CI/CD)** features. 

Close Reading 3. Read the text above. Match each term or tool with its correct description:

- | | |
|-----------|--|
| 1. Git | a. A cloud-based platform for hosting Git repositories with collaboration tools. |
| 2. GitHub | b. A cloud-based Git platform with built-in CI/CD capabilities. |
| 3. GitLab | c. A distributed version control system that operates locally. |
| 4. Commit | d. A separate line of development in a Git repository. |
| 5. Branch | e. Saving changes in a version control system as a snapshot of the project. |

4. Read the text again carefully. Choose the correct answer (a, b, or c) for each question.

1. What makes Git a *distributed* version control system?
 - a) It stores data only in the cloud
 - b) Each developer has a full local copy of the repository
 - c) It allows only one person to work at a time
2. What is the purpose of the **staging area** in Git?
 - a) To publish code online
 - b) To review other people's code
 - c) To prepare changes before committing
3. What is a **commit** in Git?
 - a) A file that tracks deleted lines
 - b) A snapshot of the project at a specific moment
 - c) A tool for tracking merge conflicts
4. How do **GitHub** and **GitLab** extend Git's functionality?
 - a) By allowing offline development
 - b) By offering visual editors for programming
 - c) By providing cloud storage, collaboration tools, and pull requests
5. What is a **pull request**?
 - a) A way to automatically fix bugs
 - b) A suggestion to delete a branch
 - c) A request to review and merge code into the main branch
6. Which platform is especially known for supporting open-source communities?
 - a) Git
 - b) GitHub
 - c) GitLab
7. Which feature does GitLab offer that is highlighted in the reading?
 - a) Built-in CI/CD tools
 - b) Better branch visualization
 - c) Offline access
8. What potential issue can occur when merging branches?
 - a) Loss of data
 - b) Duplicate commits
 - c) Code conflicts

Vocabulary

5. Match the highlighted words in the text with their definitions.

Word/Phrase	Definition
1. repository	a. shared or spread out across multiple locations
2. distributed	b. a place to store changes before they are committed
3. merge	c. a request to merge changes from one branch into another
4. branch	d. to save changes to a local repository
5. clone	e. a situation where changes in two versions of a file contradict each other
6. commit	f. a system for managing changes to code or documents over time
7. pull request	g. a central location where data is stored and managed
8. version control	h. to combine multiple versions or changes into one unified version
9. conflict	i. to create a copy of an existing repository
10. staging area	j. a separate line of development in version control systems

6. Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3><https://quizlet.com/948411349/match?funnelUUID=538feaaf-3de8-439a-94d0-b7cce627ab64>

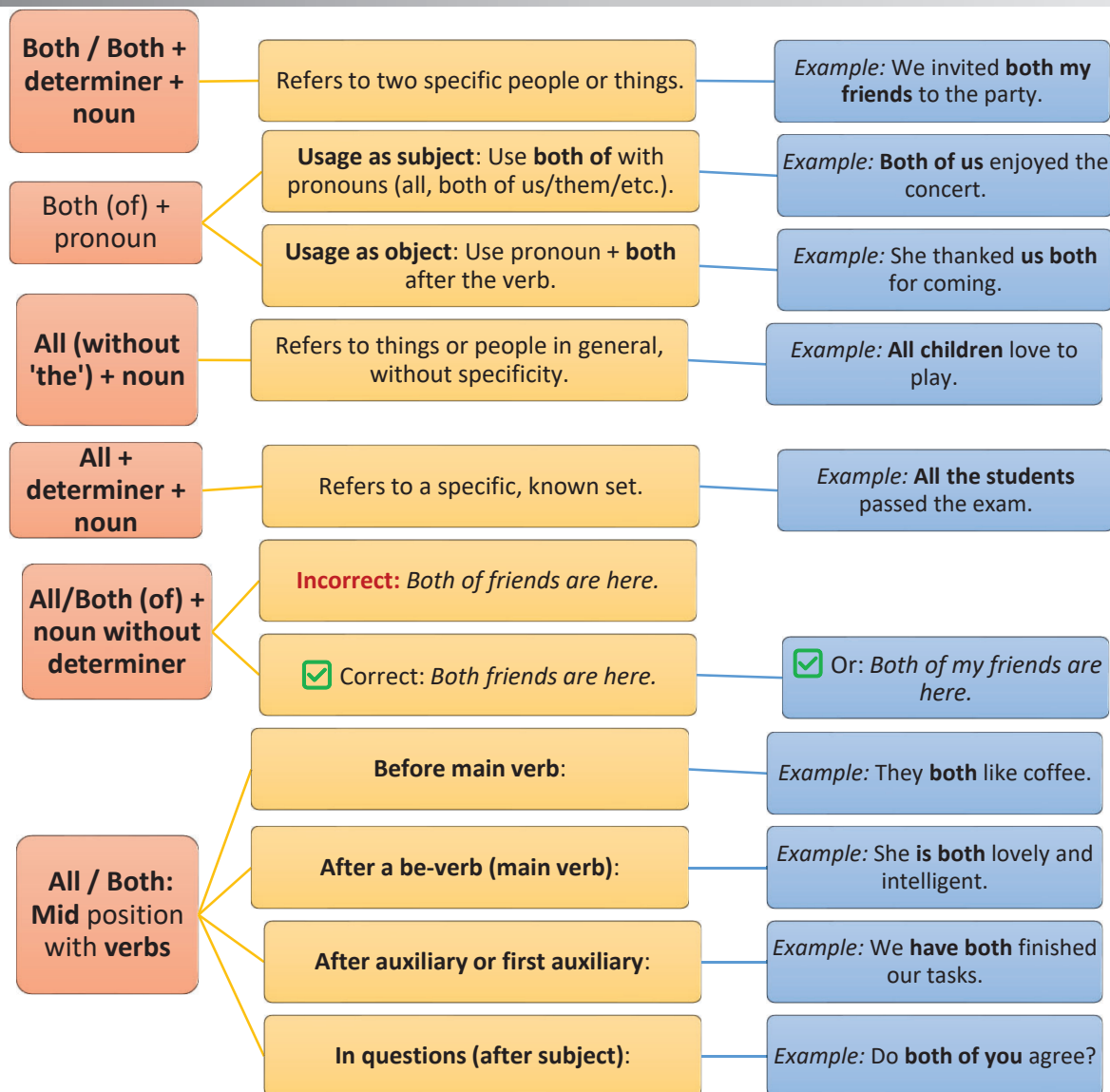


7. Fill in the blanks with the correct words from the list below.

Word Box: distributed, repository, branch, clone, merge, commit, conflict, staging area

Version control systems like Git allow developers to track changes and work collaboratively. Each developer has a copy of the entire _____ on their local machine, which is known as a _____ system. When developers work on new features, they create a _____ to keep their changes separate from the main codebase. Once they are ready to save their work, they add the changes to the _____ and then _____ them to their local copy of the repository. After completing their work, developers _____ their branch with the main branch, resolving any _____ that may arise. Finally, they can push the changes to the cloud, making them available to the entire team.

Grammar for Revision All/both



8. Choose the correct option from the two given in brackets.

- _____ my sisters live in different countries. (**both** / **all**)
- We met _____ of our teachers at the conference. (**both** / **all**)
- Did you invite _____ of them to your birthday party? (**both** / **all**)
- I've read _____ the articles you sent me. (**all** / **both**)
- He was so generous — he paid for _____ of us. (**all** / **both**)
- The twins are talented — _____ of them play instruments. (**both** / **all**)

7. They _____ arrived late because of the storm. (**they both / both they**)
8. I saw _____ your parents at the meeting. (**both of / all of**)
9. She gave _____ her books away to charity. (**all / all of**)
10. _____ students must bring their ID cards to the exam. (**All / All the**)

Speaking

9. Choose the correct option (only one is correct).

Select the **correct quantifier** for each question. Then discuss your answers in pairs.

1. Do you use **both / both of / all** GitHub and GitLab in your projects?
2. Do **all of / both / all the of** your projects use a version control system?
3. Which tools do you prefer — **both / all of / both the** GitHub and GitLab?
4. Have you worked with **both / all / both of** a local and a remote repository?
5. Do you usually commit **all / both of / all of the** your changes at once?

10. Now take turns asking and answering the questions above. Try to use **all / both / all of / both of the** in your answers naturally.

Example:

Q: Do you use **both** GitHub and GitLab in your projects?

A: Yes, I use **both** depending on the team. But **most of our code** is hosted on GitHub.

Extra Online Practice

Vocabulary:

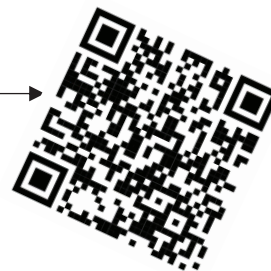
Quizlet – Unit 4.2 Flashcards & Learn



Grammar:

All, both: Quantifiers

Exercises 1–3 on Test-English.com



Recommended activity before starting Lesson 3:

Play Quizlet Live (INDIVIDUAL MODE, 3 TIMES)

using vocabulary from Unit 4.2:







<https://quizlet.com/ua/948411349/unit-42-flash-cards/?i=j03j6&x=1jqt>

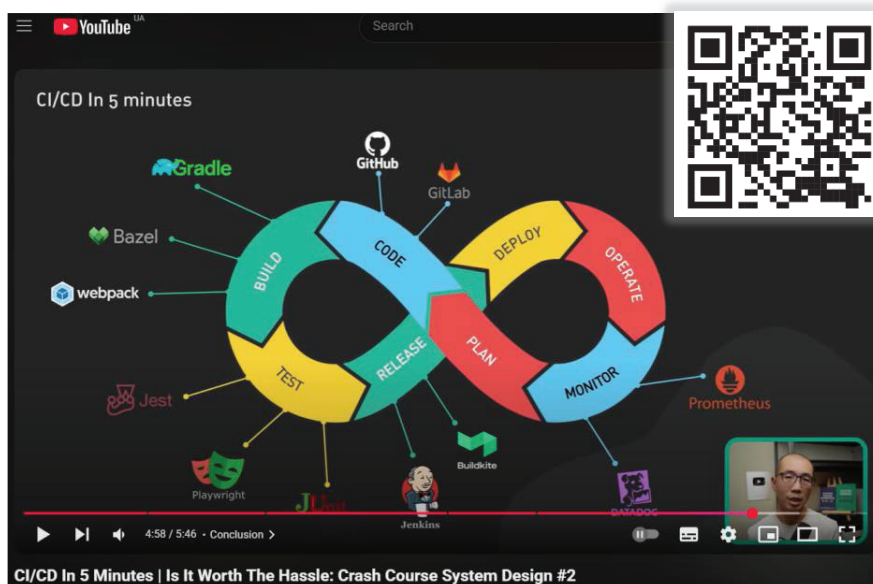
Lesson 3

Overview of DevOps practices, including Continuous Integration and Continuous Deployment (CI/CD)

Lead-in 1 Class Discussion

1. Think about the last time you received a software update (on your phone or PC).
 How often do updates appear? Do they always work smoothly?
2. Imagine you're working in a team of developers.
 What problems might happen if everyone writes and uploads code at the same time?
3. In your opinion:
 Why is it important to **test** software before it goes live?
 What could go wrong if new features are pushed without testing?

Watch the video.



Before Watching the Video

answer the following questions:

- › Have you ever heard of **CI/CD** in software development?
- › What do you think "**Continuous Integration**" and "**Continuous Deployment**" might mean?

Figure 6 CI/CD In 5 Minutes | Is It Worth The Hassle: Crash Course System Design #2 [<https://youtu.be/42UP1fxi2SY>]

2. Read each question and choose the correct answer (a, b, or c) based on the video.

1. What is the main purpose of CI/CD?

- a) To write code faster
- b) To automate and improve the software development and delivery process
- c) To replace programmers with AI

2. What does CI stand for?

- a) Continuous Installation
- b) Code Integration
- c) Continuous Integration

3. What is a common feature of Continuous Integration (CI)?

- a) Manual testing by developers
- b) Automated testing after each commit
- c) Waiting until the project is finished to test

4. What does CD stand for in the video?

- a) Continuous Deployment
- b) Code Debugging
- c) Component Design

5. What is a “feature flag” used for?

- a) To highlight errors in the code
- b) To separate code deployment from feature activation
- c) To block users from accessing the system

6. Why is full continuous deployment less common for complex systems?

- a) It's too expensive
- b) It requires large teams and is risky
- c) It doesn't work with cloud-based systems

3. Watch the video again and **fill in the blanks** with the missing words. *Try to listen for the exact vocabulary used in the video.*

1. CI/CD (Continuous Integration / Continuous Deployment) automates the _____ development process from the first code commit to final deployment. 🕒 [0:14–0:19]
2. It reduces the need for manual _____ and speeds up the release of better-quality _____. 🕒 [0:25–0:44]
3. CI helps teams merge code changes into a shared _____ early and often. 🕒 [0:58–1:08]
4. Each code _____ triggers an automated workflow that builds and tests the software. 🕒 [1:08–1:14]
5. A good CI process needs high _____ coverage — but this can take more time to run. 🕒 [1:19–1:34]
6. CD (Continuous Deployment) is more challenging and less common, especially for _____ systems like databases. 🕒 [3:01–4:31]
7. Teams use techniques like feature flags and canary deployment to reduce _____ when rolling out new features. 🕒 [3:30–4:13]
8. Some popular **CI/CD tools** mentioned in the video include:
 - GitHub Actions
 - _____
 - _____
 - ArgoCD (used with Kubernetes) 🕒 [1:42–5:11]

4. Match each term or phrase from the video with its correct definition

Term	Definition
A. Continuous Integration (CI)	1. A deployment strategy that releases code to a small group of users first.
B. Continuous Deployment (CD)	2. A method to automatically test and merge code changes into the main branch.
C. Feature Flag	3. A tool used to switch new features on or off without deploying code again.
D. Commit	4. The act of saving a snapshot of code changes in the version control system.
E. Canary Deployment	5. The automated release of code directly into production systems.

Reading 4. Read the text

4.3.


Introduction to DevOps Practices and CI/CD

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops) to shorten the development lifecycle and provide continuous delivery with high software quality. A key goal of DevOps is to foster collaboration between teams that historically worked in silos—developers, quality assurance, and operations teams.

One of the main principles of DevOps is **automation**. Automating repetitive tasks, such as testing and deployment, allows teams to focus on more complex and strategic issues. **Continuous Integration (CI)** is a practice where developers frequently integrate their code into a shared repository. Each **integration** is automatically tested to detect any issues early in the development process. Automated testing is a critical component of CI, as it ensures that changes are validated before they affect the broader codebase.

Continuous Deployment (CD) takes this one step further. It **automates** the **release** of code into production, meaning every change that passes the CI **pipeline** is automatically **deployed**. This speeds up the release cycle and reduces manual intervention, helping companies react faster to market needs. Both CI and CD are implemented through a **deployment pipeline**, a series of stages such as building, testing, and deploying, that ensures the software is always in a deployable state.

In addition to automation, DevOps emphasizes **monitoring** and a strong **feedback loop** to detect and respond to issues in real time. This helps teams identify bottlenecks, monitor system performance, and quickly resolve issues. In case of errors during deployment, a **rollback** mechanism can be used to revert the application to a previous stable state.

DevOps practices like CI/CD not only improve efficiency but also promote better collaboration, reduce errors, and speed up the delivery of high-quality software to users. 

Vocabulary

5. Match the highlighted words in the text with their definitions.

Word/Phrase	Definition
1. automation	a. a series of automated stages that ensure code quality and enable smooth software release
2. pipeline	b. a series of automated steps that move code from development to production
3. deploy	c. a system that provides feedback on the performance of a process or product
4. rollback	d. a version of the software made available for users
5. integration	e. the ongoing process of tracking and analysing the performance of software systems
6. monitoring	f. the process of combining code changes from multiple developers into a shared codebase
7. feedback loop	g. the process of reverting to a previous version of software after deployment issues occur
8. release	h. the use of technology to perform tasks without human intervention
9. automate	i. to release or distribute software to a production environment
10. deployment pipeline	j. to use technology to perform tasks without human intervention

6 🖱️ Play the Quizlet Match game by following this link:

<https://quizlet.com/948413530/match?funnelUUID=80be72f3-853c-462c-8efe-a796ccb70390>



7. Complete the following text by filling in the gaps with the appropriate words from the list:

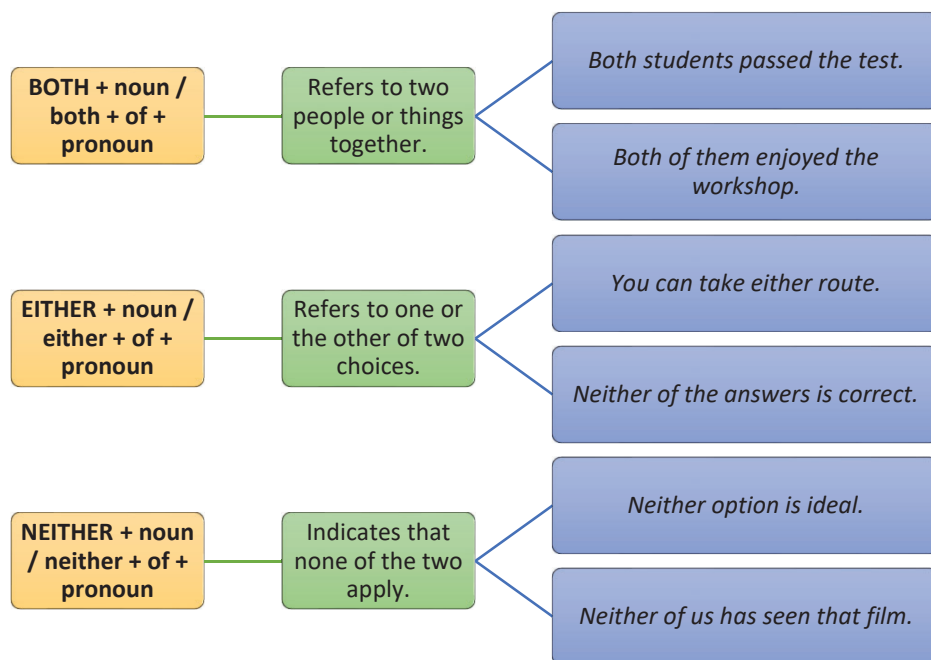
List of Words:

| automation, deploy, integration, rollback, pipeline, monitoring, feedback, automate |

1. DevOps encourages the use of _____ to handle repetitive tasks such as testing and deployment.
2. Continuous _____ (CI) ensures that code changes from multiple developers are integrated into the shared codebase frequently.
3. After passing through the _____, the software is ready to be released into production.
4. Teams rely on real-time _____ to track the health of their software systems.
5. In case an issue arises, a _____ process allows the team to revert to the previous stable version of the software.

6. A strong _____ loop ensures that teams can quickly respond to issues and improve the product.
7. The ability to _____ changes quickly reduces the time between development and user feedback.
8. With Continuous Deployment, companies can _____ updates to users automatically, without manual intervention.

Grammar for Revision **Both, either, neither: Quantifiers**



8. Choose the correct option from the two given in brackets.

1. _____ of the two laptops is suitable for gaming. (**Both / Either**)
2. I invited _____ of them, but neither could come. (**both / either**)
3. We can meet on Monday or Tuesday — _____ day works for me. (**Either / Neither**)
4. _____ of my parents are teachers. (**Both / Either**)
5. I tried two different solutions, but _____ worked. (**Neither / Either**)
6. He looked at _____ girls but didn't say anything. (**both / neither**)
7. Would you like tea or coffee? You can have _____. (**either / neither**)
8. _____ answer is correct — they both are wrong. (**Neither / Both**)
9. They _____ have experience in project management. (**both / either**)
10. I couldn't decide — I liked _____ of the options. (**both / neither**)

Speaking

🗣️ **9. Choose the correct quantifier (both / either / neither) in each sentence.**
Then, discuss the question with your partner, giving reasons and real-life examples if possible.

1. Do you think **both / either / neither** Continuous Integration and Continuous Deployment are necessary in modern development teams? Why or why not?
2. Have you ever used **both / either / neither** GitHub Actions or Jenkins for automating workflows? Which one do you prefer?
3. In your opinion, should DevOps teams focus on **both / either / neither** speed or code quality when deploying updates?
4. Would you feel comfortable setting up **both / either / neither** CI or CD pipelines yourself?
5. Have you worked on a project where **both / either / neither** the CI and CD processes were automated? How did it affect team productivity?

🔑 Extra Online Practice

Vocabulary:

[Quizlet – Unit 4.3 Flashcards & Learn](#)

Grammar:

Both, either, neither: Quantifiers

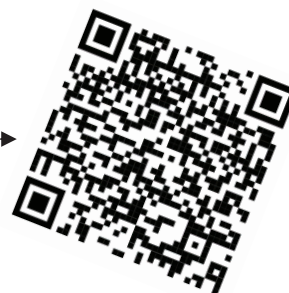
[Exercises 1–3 on Test-English.com](#)

☒ Recommended activity before starting **Revision**

Unit 4:

Play Quizlet Live (INDIVIDUAL MODE, 3 TIMES) using vocabulary from Unit 4.3:

🔗 <https://quizlet.com/ua/948413530/unit-43-flash-cards/?i=j03j6&x=1jqt>



Vocabulary Revision

Unit 4

1. WORK IN PAIRS. TAKE TURNS PLAYING THE ROLES OF DESCRIBER AND GUESSER USING VOCABULARY FROM UNIT 4.

◇ Detailed instructions can be found on page 37

Term	Definition
release	a version of the software made available for users
automate	to use technology to perform tasks without human intervention
deployment pipeline	a series of automated stages that ensure code quality and enable smooth software release
repository	a central location where data is stored and managed
distributed	shared or spread out across multiple locations
merge	to combine multiple versions or changes into one unified version
branch	a separate line of development in version control systems
clone	to create a copy of an existing repository
commit	to save changes to a local repository
pull request	a request to merge changes from one branch into another
version control	a system for managing changes to code or documents over time
conflict	a situation where changes in two versions of a file contradict each other
staging area	a place to store changes before they are committed
tremendous	extraordinarily large in size or extent or amount or power or degree
due to	because of
comprehensive	covering or including everything
harmful	causing or likely to cause harm; damaging
emphasize	give special importance or prominence to (something) in speaking or writing.
incremental	increasing gradually by regular degrees or additions
manageable	capable of being controlled, directed, or accomplished
iterative	a process that repeats a series of steps over and over until the desired outcome is obtained
exist	to be real; to be found; to occur; to stay alive.
differ	to be different from something in some way, vary

feature	an important part of something
thoroughly	in a detailed and careful way
upfront	before goods or services are received
branch	a part of something larger
adopt	to accept or start to use something new
deployment pipeline	a series of automated stages that ensure code quality and enable smooth software release
Continuous Integration (CI)	a method to automatically test and merge code changes into the main branch
Continuous Deployment (CD)	the automated release of code directly into production systems.
Feature Flag	a tool used to switch new features on or off without deploying code again
Canary Deployment	a deployment strategy that releases code to a small group of users first.

2. THREE-SENTENCE CHALLENGE. Each participant **selects three words** from the shared word list and forms three sentences using these words. Work in pairs. Read your sentences to your partner, replacing the chosen word with "gap" while reading. Your partner needs to guess the missing word.

3. PLAY THE QUIZLET LIVE GAME (TEAM MODE) to review the vocabulary learned in **UNIT 4**

<https://quizlet.com/ua/948414070/unit-4-42-43-41-flash-cards/?i=j03j6&x=1jqt>



4. TAKE the UNIT 4 VOCABULARY QUIZLET TEST to check your understanding of key terms from this UNIT.

<https://quizlet.com/948371787/test?answerTermSides=6&promptTermSides=6&questionCount=30&questionTypes=15&showImages=truehttps://quizlet.com/948414070/test?answertermsides=6&prompttermsides=6&questioncount=30&questiontypes=15&showimages=true>



5. VOCABULARY GAME: ALIAS

Play a fun team-based vocabulary challenge!

♦ See full game instructions on page 37.

Get ready to explain, guess, and compete using key terms from Unit 4!

For Printable Vocabulary Materials for the Game “Alias,” go to page 233.

5

UNIT

PRODUCT MANAGEMENT TOOLS



This unit introduces students to essential tools and practices used in modern software development and project management. Learners will explore common project management challenges, and gain practical knowledge of **project management software** like Jira, Trello, and Asana. Special emphasis is placed on collaborative development, Gantt chart planning, and selecting appropriate digital tools for various team and project needs.

Unit overview

5.1 Detailed look at the software development lifecycle

Lesson outcome: Learners can describe the main phases of SDLC and explain the importance of planning, testing, and maintenance.

Skills practised

Listening: identifying key information in an explanatory video

Skim reading

Vocabulary: SDLC phases and project terminology

Grammar: question forms – yes/no, wh-, tag, subject, choice

5.2 Common project management challenges and solutions, key steps in drawing up a Gantt Chart

Lesson outcome: Learners can identify key project challenges and apply Gantt charts to manage tasks and dependencies effectively.

Skills practised:

Reading (for gist and detail)

Speaking: problem-solving discussions

Vocabulary: project management terms (e.g., constraints, milestones)

Grammar: prepositions in questions – formal/informal usage

5.3 Overview of project management software: Jira, Trello, Asana, etc.

Lesson outcome: Learners can compare features of project management tools and describe how to choose the right one for a project.

Skills practised:

Reading: comparing features and tools

Speaking: discussing tool preferences

Vocabulary: tool-specific terminology (e.g., sprint, backlog, Kanban board)

Grammar: question tags (e.g., isn't it?, don't you?)

Lesson 1

Detailed look at the software development lifecycle

Lead-in 1.  Discuss in pairs or small groups:

1. What steps do you think are involved in developing a software product from scratch?
2. Which of these phases do you think is the most difficult or time-consuming? Why?
3. Have you ever faced a situation where a software didn't work as expected? What do you think might have gone wrong during the development process?

2. Put the stages in order

a. Planning – defining the scope, timeline, and resources.	1
b. Maintenance – fixing issues and improving the product over time.	6
c. Design – creating the software architecture and user interface.	
d. Testing – checking for bugs and ensuring the product meets requirements.	
e. Deployment – releasing the product to users or clients.	
f. Development – writing and integrating the actual code.	

Watch the video [\[0:10 -0:40\]](#) and check your answers.

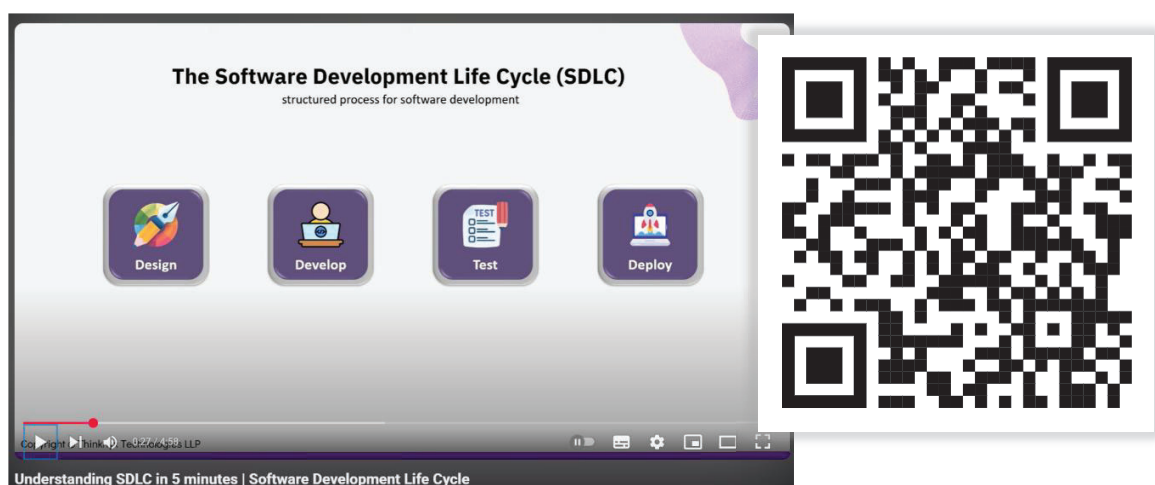


Figure 7 Understanding SDLC in 5 minutes | Software Development Life Cycle [Source: <https://youtu.be/jwzePkKhJc>]

3. Watch the video again until the end and complete the text below using the correct words from the box. You may use each word only once.

Word Bank:

feasibility requirements deployment agile development testing design
user interface spiral planning maintenance model project integration

The Software Development Life Cycle (SDLC) is a structured process that helps software teams design, develop, and deploy high-quality software. It begins with the (1)_____ [0:40–0:46] phase, where developers define the goals, timeline, and conduct a (2)_____ [0:56–1:01] study. During (3)_____ [0:49–0:53] gathering, the team gathers detailed information about functionality and performance expectations.

Next comes the (4)_____ [1:12–1:32] phase, where the system architecture and (5)_____ [1:27–1:32] are created.

In the (6)_____ [1:40–1:42] phase, programmers write the code. Then, different levels of (7)_____ [1:45–2:15] are performed, including unit, (8)_____ [2:00–2:07], system, and user acceptance testing.

After that, the (9)_____ [2:19–2:25] phase delivers the final product to the users. Ongoing (10)_____ [2:39–2:56] is important to fix bugs, adapt to changes, and improve performance.

Several SDLC models exist. The (11)_____ [3:12–3:19] model is a linear, step-by-step approach. The (12)_____ [3:23–3:27] model focuses on flexibility and collaboration. The (13)_____ [3:47–3:52] model combines iterative development with a focus on risk. Choosing the right (14)_____ [3:55–4:01] depends on the size, complexity, and nature of the (15)_____ [4:03–4:12].

Reading 3. Read the text carefully, then complete the following *true/false* based on the content.

5.1.

Introduction to the Software Development Lifecycle

Write **T** if the sentence is true, and **F** if it is false. *Correct the false statements.*

1. The main purpose of SDLC is to create simple software as quickly as possible.
2. The planning phase includes defining scope, timelines, and conducting feasibility studies.
3. The design phase does not involve planning the user interface.
4. Testing is only performed once the product is already deployed.
5. Maintenance ensures the software remains usable and up to date after deployment.

The software development **lifecycle** (SDLC) is a structured process used by software developers to design, develop, and test high-quality software. SDLC aims to produce software that meets or exceeds customer expectations and is completed within time and cost estimates. This lifecycle typically includes six key phases: Planning, Analysis, Design, Implementation, Testing, and Maintenance.

Planning: This phase involves defining the project scope, objectives, and **feasibility**. During the planning stage, project managers work closely with **stakeholders** to gather **requirements**, allocate resources, and set timelines. Feasibility studies are often conducted to assess whether the

project is **viable** in terms of time, cost, and technology.


Analysis: Once the plan is approved, the next phase is analysis. In this phase, software engineers collect detailed requirements from stakeholders and thoroughly analyse them to ensure the project aligns with business goals. This step helps in identifying possible risks and preparing strategies to mitigate them.

Design: The **design** phase involves creating a blueprint of the software solution. Here, both the system architecture and user interface are outlined, ensuring that the design meets all functional and technical requirements. Detailed designs allow developers to visualize how the system will function once completed.

Implementation (Coding): The design is then translated into code in the **implementation** phase. Programmers begin writing code according to the specifications provided in the design document. This phase is critical as it turns theoretical designs into a working product.

Testing: After the code is written, it must be thoroughly tested to ensure that it meets all requirements and functions correctly. During testing, software developers use various techniques such as unit testing, integration testing, and user acceptance testing (UAT) to validate the software. The goal is to identify and fix any bugs before deployment.

Deployment and maintenance: In the final phase, the product is deployed to users. After deployment, software requires ongoing maintenance to fix any issues, update features, and ensure the product remains functional over time. Maintenance can also involve performance tuning and updating the software to meet new security requirements.

Each phase of the SDLC is vital for ensuring a smooth, well-organized development process. By following this structured lifecycle, teams can deliver high-quality products that meet business goals and customer needs. 

4. Answer the questions

1. What are the six main phases of the Software Development Lifecycle (SDLC)?
2. How does the planning phase set the foundation for the success of the entire project?
3. Why is the analysis phase crucial for ensuring that the software aligns with business goals?
4. What is the purpose of the design phase, and how does it contribute to the development process?
5. How does testing ensure the quality and functionality of the software product?
6. Why is maintenance important after the software has been deployed?
7. How can a feasibility study help determine whether a project should move forward?
8. What are some common techniques used during the testing phase to ensure the software works correctly?

Vocabulary

5. Match the highlighted words in the text with their definitions.

Word	Definition
1. lifecycle	a. a person or group with an interest in the success of a project
2. feasibility	b. something necessary or compulsory in a project or system
3. deployment	c. practicable, capable of developing
4. viable	d. the practicality of a proposed plan or method
5. requirement	e. the process of creating a plan or drawing to show the function or workings of a system
6. maintenance	f. the process of evaluating the performance and quality of a system or product
7. design	g. the process of preserving or keeping something in good condition
8. implementation	h. the process of putting a decision or plan into effect; execution
9. testing	i. the release or distribution of software for use
10. stakeholder	j. the series of changes that a product or system goes through during its existence

6. 🖱️ Play the Quizlet Match game by following this link:

<https://quizlet.com/948672903/match?funnelUUID=a2a95440-4f8a-4d81-94b1-b2f690fcbfa3>



7. Complete the following text by filling in the gaps with the appropriate words from the list provided:

List of Words:

lifecycle, testing, design, feasibility, requirement, maintenance, implementation |

1. Evaluating the _____ of the project is crucial to avoid unrealistic timelines and budget overruns.
2. After coding is completed, the software enters the _____ phase to identify and fix any bugs.
3. The _____ phase involves creating detailed plans that outline how the system will function.
4. The process of translating the design into code happens during the _____ phase.
5. During the _____ phase, the team makes necessary adjustments and fixes to ensure the software remains functional.
6. The software development _____ consists of several distinct stages that help guide a project from conception to completion.
7. Each software feature is considered a _____ that must be met to ensure project success.

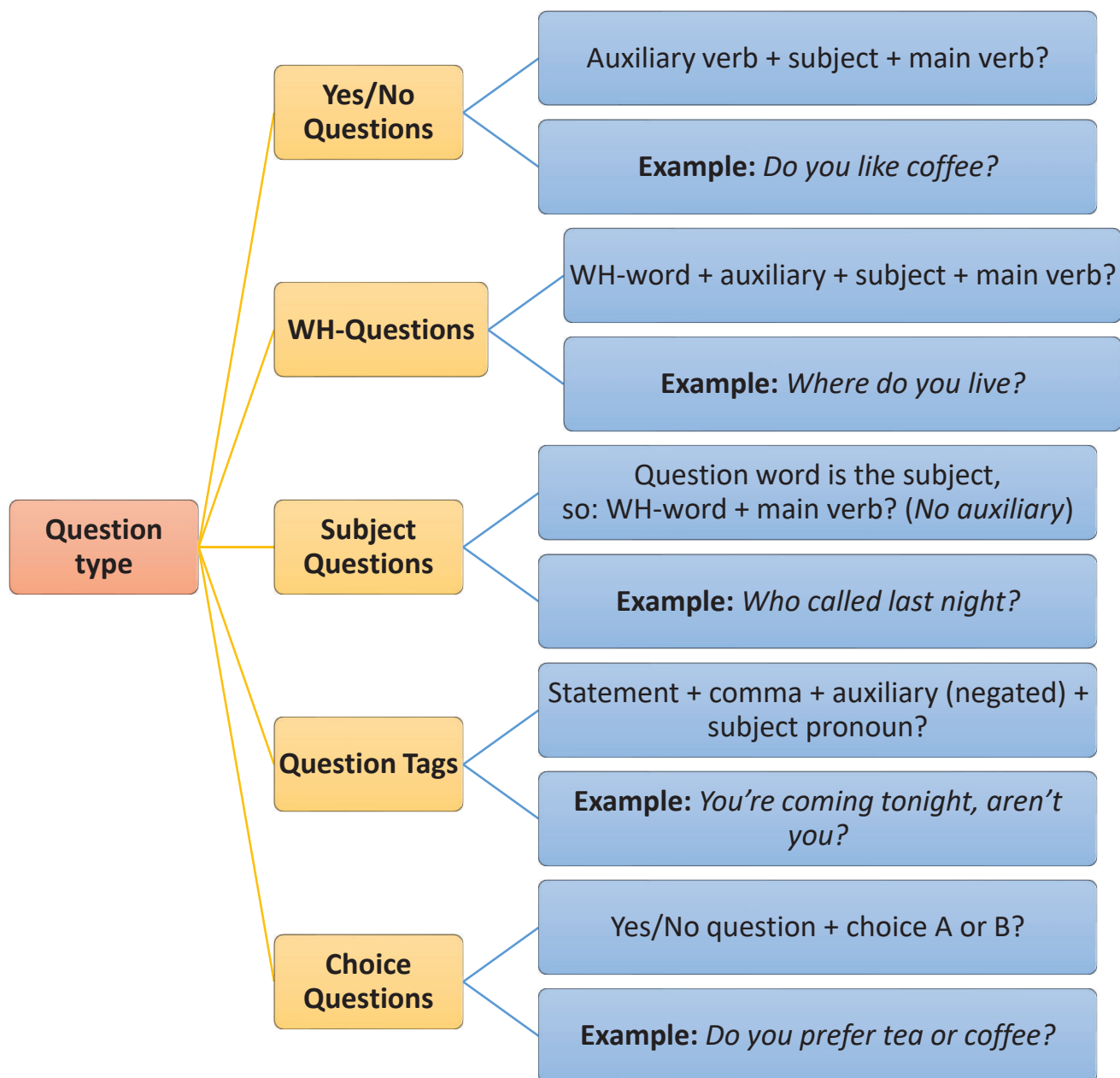
Grammar for Revision Question Forms in English

*In English, **questions are formed in various ways**, depending on their **type**:*

Yes/no questions are used when a simple confirmation is needed. These questions begin with an **auxiliary verb**, followed by the **subject** and the **main verb**.

Wh- questions start with words like "what," "where," "when," "why," "who," or "how." They follow a structure where the **Wh-word** comes **first**, followed by an **auxiliary verb**, the **subject**, and then the **main verb**. However, when the **Wh-word serves as the subject** of the question, the **auxiliary verb is omitted**, and the question is formed directly using the **subject** and the **main verb**.

Another type of question is the **tag question**, which consists of a **statement** followed by a **brief question at the end**. This is used to confirm or clarify the information presented in the statement.



8. Complete the questions using the correct structure

1. _____ you ever worked on a software project before? (*yes/no*)
→ (*Have / Do / Did*)
2. _____ is responsible for writing the initial project plan? (*wh-question*)
→ (*Who / What / When*)

3. This design looks solid, _____? (*tag question*)

→ (*isn't it / doesn't it / wasn't it*)

4. _____ team handles the testing phase in your company? (*wh-question – subject*)

→ (*What / Which / Who*)

5. You completed the requirements analysis last week, _____? (*tag question*)

→ (*did you / didn't you / do you*)

6. _____ do software engineers usually do during the implementation phase? (*wh-question*)

→ (*What / Who / How*)

7. _____ you need to gather stakeholder input before planning? (*yes/no*)

→ (*Are / Do / Have*)

8. We shouldn't skip the feasibility study, _____? (*tag question*)

→ (*should we / should we not / shouldn't we*)

9. Rewrite the following statements as questions. Use the correct question form (yes/no, wh-, or tag).

1. The team completed the testing phase yesterday.



2. She designs user interfaces for the mobile app.



3. They are working on the deployment plan now.



4. The software met all functional requirements.



5. This lifecycle model is suitable for complex projects.



10. **Work in pairs.** Take turns choosing a statement from **Exercise 7**.

Create a question about the statement using the correct **question form** (Yes/No, Wh-, Tag, Subject, or Choice question).

11. 🎲 Question Practice Game

🎲 **Game Instructions:** Work in pairs or small groups. Roll a dice and check the **dice key** on the right 🎲.

Choose a statement from the list below. **Make a question** based on your dice result and the chosen statement.

💬 Statements

1. We finished the testing phase last week.
2. Our team uses Agile methodology.
3. I usually write unit tests before committing my code.
4. GitHub is better for open-source collaboration.
5. Continuous Deployment is hard to implement.
6. We use feature flags to manage new releases.
7. The team meets every Monday to plan the sprint.
8. I've never worked with GitLab before.
9. Our last deployment caused several bugs.
10. The client changed the requirements halfway through the project.

🎲 Dice Roll for Question Type

Use a **6-sided dice** with this **key**:

- 1=Yes/No Question
- 2=Wh-Question
- 3=Tag Question
- 4=Subject Question
- 5=Choice Question
- 6=Any type (Student's choice)

12. Choose three words from Exercise 5. Create your own **questions** using those words. 🗣️ Then, **ask your partner** and discuss their answers.

🔍 Extra Online Practice

Vocabulary:

Quizlet – Unit 5.1 Flashcards & Learn →



Grammar:

Asking questions in English

Exercises 1–4 on Test-English.com →



☒ Recommended activity before starting Lesson 2:

Play Quizlet Live (INDIVIDUAL MODE, 3 TIMES) using vocabulary from Unit

5.1: 🔗 <https://quizlet.com/ua/932616895/unit-31-flash-cards/?i=j03j6&x=1jqt>

<https://quizlet.com/ua/948672903/unit-51-flash-cards/?i=j03j6&x=1jqt>

Lesson 2

Common project management challenges and solutions, key steps in drawing up a Gantt Chart

Lead-in 1.  **Discuss in pairs or small groups:**

1. What makes a group project successful?
2. What problems usually appear when working in teams?
3. How do you think professionals keep track of multiple tasks and deadlines?

2. Match the project management challenge (1–6) with the situation (A–F). Then, predict what kind of solution might help in each case.

 *Discuss in pairs before reading the full text.*

Challenge	Situation
1. Time Constraints	A. The project suddenly includes more features than originally planned.
2. Scope Creep	B. You can't start your task because another teammate hasn't finished theirs.
3. Resource Allocation Issues	C. The team lacks enough developers to meet the deadline.
4. Poor Communication	D. Important information wasn't shared, so tasks were repeated unnecessarily.
5. Risk Management Failures	E. A key supplier goes out of business mid-project.
6. Task Dependencies	F. The product launch is delayed because a prior task wasn't completed on time.

Reading 3. Read the text. Decide if the statements below are TRUE or FALSE.

 **5.2.**

Common Project Management Challenges and Solutions

In project management, challenges often arise that can **hinder** the completion of projects on time and within budget. Below are some of the most common issues and potential solutions:

Read the statements below. Decide if they are **TRUE** or **FALSE**. *Correct the false ones.*

1. Gantt charts help teams manage time and visualize project progress.
2. Scope creep is helpful because it allows teams to add new features without changing the schedule.
3. One solution to scope creep is to define clear objectives at the start.
4. Resource allocation is not a problem if there are enough people on the team.
5. Poor communication can lead to duplicated work and missed deadlines.
6. Risks cannot be predicted, so there is no point in trying to plan for them.
7. Gantt charts can help manage task dependencies.
8. Delays in one task can cause a

Time Constraints: one of the most prevalent challenges in project management is completing the project within the set timeframe. Delays can occur due to unforeseen circumstances, poor planning, or overestimating the team's capacity. *Solution:* using **milestones** and time-tracking tools such as Gantt charts can help teams visualize the project timeline and ensure that deadlines are met.

Scope Creep: scope creep occurs when the project's original objectives expand over time without adjusting deadlines or budgets, making it difficult to meet the original goals. *Solution:* clearly defining project objectives and using **baseline** documentation at the start of a project helps prevent scope **creep**. Any new requests should go through a formal change management process.


Resource Allocation Issues: misallocating resources can delay projects and increase costs. Teams may not have enough staff, materials, or budget to complete tasks as planned.

Solution: effective **resource allocation** requires regular monitoring and adjusting resources as needed, based on priority tasks.

Poor Communication: without proper communication, tasks may overlap, or important details may be missed, leading to inefficiencies and errors. *Solution:* establishing a clear communication plan that includes regular check-ins and updates with stakeholders can reduce misunderstandings and delays.

Risk Management Failures: unexpected risks, such as market changes or technical difficulties, can severely impact the project. *Solution:* Building a strong risk management plan and identifying potential risks early can help mitigate or eliminate negative impacts. Including contingency plans in the timeline and budget ensures flexibility.

Task Dependencies: often, one task must be completed before another can begin. When there is a delay in the first task, it creates a domino effect that can delay the entire project. *Solution:* managing dependencies using a tool like a Gantt chart helps identify critical tasks and adjust schedules to avoid unnecessary delays.

 Learn more about
Gantt charts: [gantt.com](https://www.gantt.com)

By addressing these challenges proactively, project managers can lead their teams to success. 

4. Rank the six project management challenges **from 1 (most serious) to 6 (least serious) based on how much impact you think they have on a project's success.**

After ranking, **explain your top two choices** in small groups or pairs:

- *Why do you think these challenges are the most serious?*
 - *Have you (or someone you know) experienced these in a real project?*
- A. Time constraints
 - B. Scope creep
 - C. Poor communication
 - D. Risk management failures
 - E. Task dependencies
 - F. Resource allocation issues


Vocabulary 5. Match the highlighted words in the text with their definitions.

Word/Phrase	Definition
1. constraint	a) an initial plan or budget used for comparison as the project progresses
2. milestone	b) identifying, assessing, and controlling threats to a project's objectives
3. resource allocation	c) the gradual growth or increase of something in a way that was not expected or wanted
4. deliverable	d) a relationship between tasks, where one task relies on another to begin or complete
5. contingency	e) a plan designed to take account of possible future events or circumstances
6. dependency	f) a tangible or intangible product/result produced as part of a project
7. hinder	g) the process of assigning and managing assets in a way that supports project goals
8. risk management	h) a significant point or event in a project
9. baseline	i) very seriously, in a very bad or serious way
10. creep	j) a limitation or restriction on time, resources, or scope
11. severely	k) to limit the ability of someone to do something, or to limit the development of something:

6. 🖱️ Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3><https://quizlet.com/948674161/match?funnelUUID=4c9843c3-0c6a-42a8-acd5-0160f8de27f5>



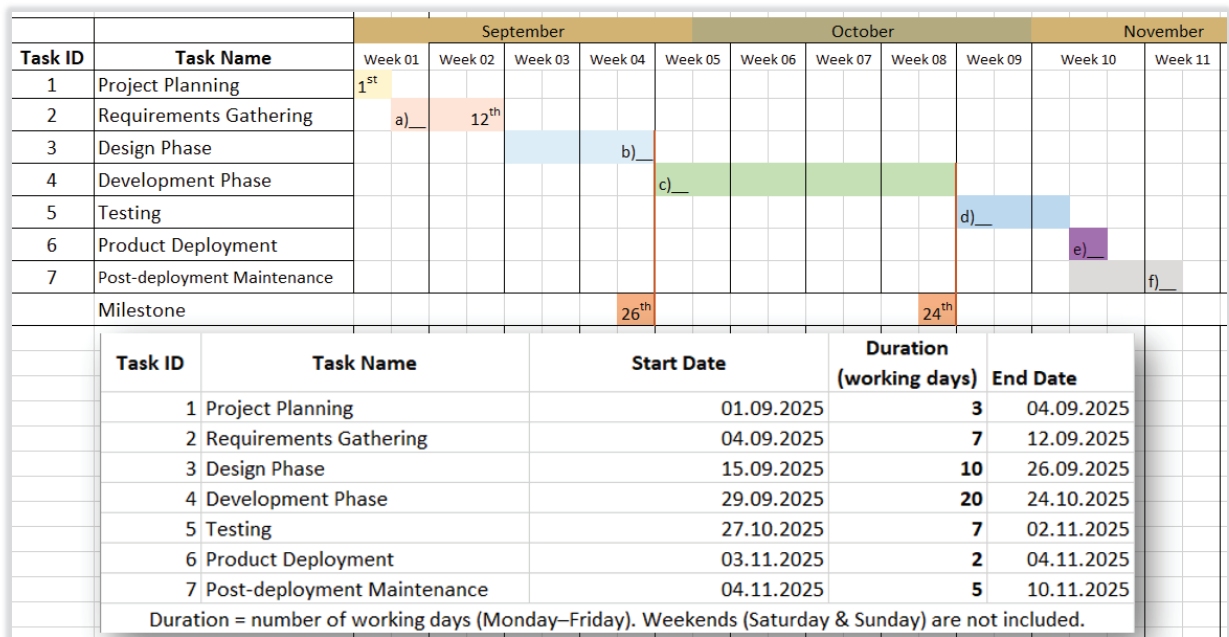
7.  Complete the sentences with words from the box. Use the words in the correct form.

Word Box:

constraint, milestone, resource allocation, deliverable, contingency,
dependency, hinder, risk management, baseline, creep, severely

1. One of the first steps in project planning is to establish a clear _____ for scope, time, and cost.
2. The final project _____ was a fully functional mobile application.
3. Poor _____ can lead to delays and missed deadlines, especially if key staff are overbooked.
4. A lack of communication can _____ team performance and lead to costly errors.
5. Scope _____ often happens when additional tasks are added without adjusting the budget or timeline.
6. The team celebrated completing their first major _____—launching the beta version.
7. When tasks are linked together, a delay in one can create a _____ for others.
8. Time _____, like a short delivery window, can pressure teams and reduce quality.
9. We included a _____ plan in case the supplier couldn't meet the delivery deadline.
10. Without proper _____, unexpected issues can disrupt even the best-planned project.
11. A bug in the payment system _____ impacted the product's launch.

8. Complete the summary of the project timeline based on the Gantt chart.



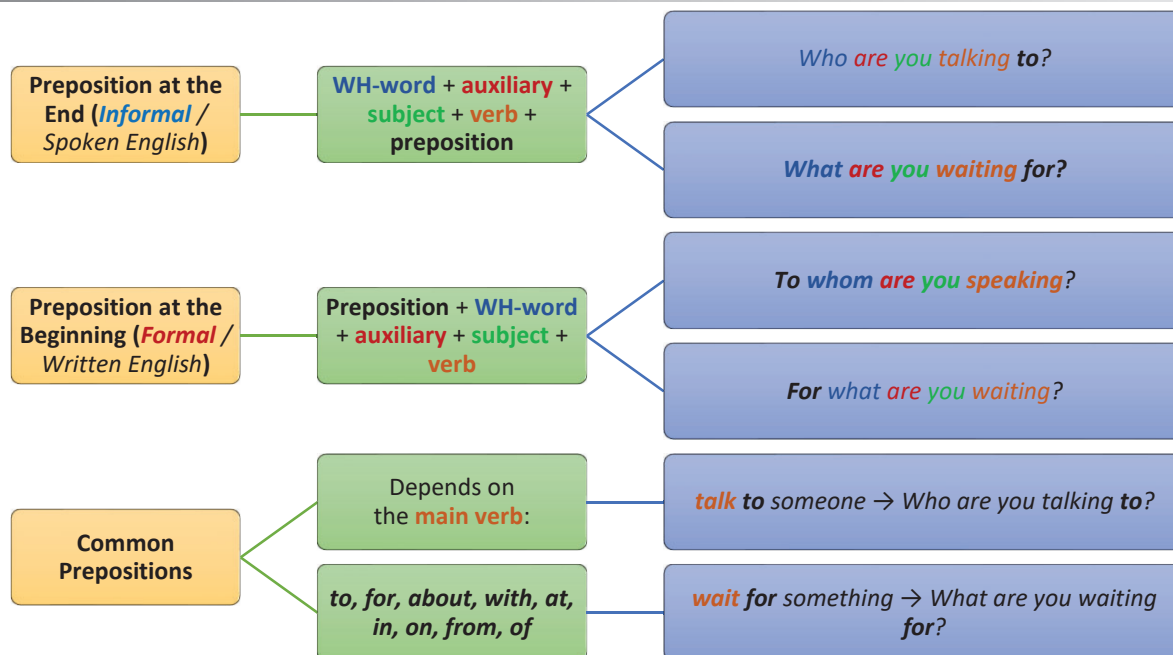
The project started with (1) _____, which lasted for three days. Then, the team moved on to (2) _____ gathering to understand the client's needs. In mid-September, the (3) _____ phase began, where system architecture and UI were planned. The (4) _____ phase was the most time-consuming, taking almost three weeks. Once coding was complete, the product went through (5) _____ to find and fix bugs. After this, the software was ready for (6) _____, when it was released to users. The final step was (7) _____, where the product was updated and supported.

A key (8) _____ was reached on September 26, marking the completion of the design phase. October 24 was another milestone, marking the end of the development phase.

9. Complete the date gaps a)–f) in the Gantt chart using the project schedule.

a) _____, b) _____, c) _____, d) _____, e) _____, f) _____

Grammar for Revision Questions with Prepositions



10. Rewrite each sentence as a question in two ways:

1. Using the preposition at the end (*informal / spoken English*).
2. Using the preposition at the beginning (*formal / written English*).

Example:

*The software was ready **for** deployment.*

➡ **Informal:** What was the software ready **for**?

➡ **Formal:** **For** what was the software ready?

1. The project started **with** planning.

Informal: ➡ What did the project start _____?

Formal: ➡ _____

2. The team moved **on to** requirements gathering.

Informal: ➡ What did the team move _____?

Formal: ➡ _____

3. The system architecture was created **during** the design phase.

Informal: ➞ What phase was the system architecture created _____?

Formal: ➞ During what phase _____

4. The product went **through** testing to find and fix bugs.

Informal: ➞ What did _____ to find and fix bugs?

Formal: ➞ **Through** what _____?

5. The software was ready **for** deployment.

Informal: ➞ What was the software _____?

Formal: ➞ _____

6. Poor resource allocation can lead **to** delays and missed deadlines.

Informal: ➞ What _____?

Formal: ➞ _____ can poor resource allocation lead?

7. A key milestone was reached **on** September 26.

Informal: ➞ Which date was the first key milestone reached _____?

Formal: ➞ _____ date was the milestone reached?

8. The milestone marked the completion **of** the design phase.

Informal: ➞ What did _____?

Formal: ➞ _____ did the milestone mark the completion?

9. October 24 marked the end of development.

Informal: ➞ What did October 24 mark the end _____?

Formal: ➞ _____ mark the end?

Speaking

🗨️ **Work in pairs. Take turns being the interviewer and the developer.**
Ask and answer questions about a **fictional software project**.

🎭 **Role-play this interview twice:**

Student 1: in an **informal style** (put the preposition at the **end** of the question).

Student 2: in a **formal style** (put the preposition at the **beginning** of the question).

Prompt (Statement)

1. The project started with planning.
2. The team moved on to testing.
3. Bugs were found during testing.
4. The software was ready for deployment.
5. Deployment occurred on October 10.
6. The design phase included creating the UI.
7. A milestone was reached after development.
8. Maintenance continued for two months.
9. Requirements were gathered from stakeholders.
10. Testing revealed compatibility issues.

🔑 Instructions:

- Choose a **prompt** and create a question based on it.
- Ask your partner.
- Your partner gives a **realistic answer**, imagining they are part of a software team.
- Then **switch roles**.

Example statement:

The software was ready for deployment.

🗨️ *Interviewer:* After testing, what was the software ready for?

👨‍💻 *Developer:* It was ready for deployment.

🔍 Extra Online Practice

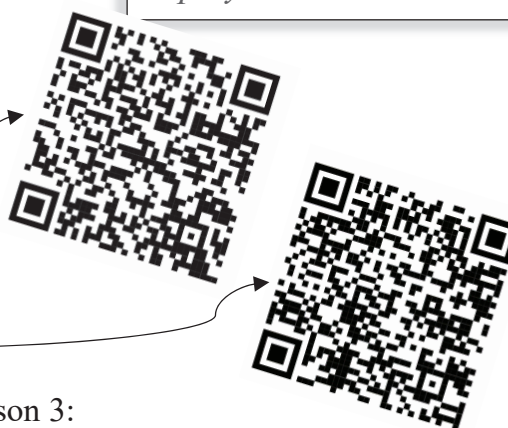
Vocabulary:

[Quizlet – Unit 5.2 Flashcards & Learn](#)

Grammar:

Questions with preposition

[Exercises 1–3 on Test-English.com](#)



☑ Recommended activity before starting Lesson 3:

Play Quizlet Live (**INDIVIDUAL MODE, 3 TIMES**) using vocabulary from Unit 5.2:

🔗 <https://quizlet.com/ua/932616895/unit-31-flash-cards/?i=j03j6&x=1jqt>

<https://quizlet.com/ua/948674161/unit-52-flash-cards/?i=j03j6&x=1jqt>

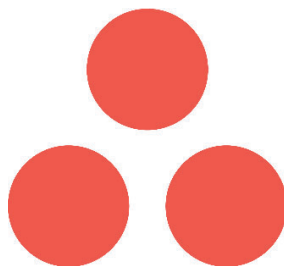
Lesson 3

Overview of project management software: Jira, Trello, Asana

Lead-in 1. Work in pairs or small groups. Discuss the questions below:

- *Have you ever used any digital tool to plan or manage tasks (e.g., to-do list apps, calendars, Google Docs)?*
- *What features do you think are useful in a tool designed for managing group projects?*

2. Look at the logos of Jira, Trello, and Asana. Have you seen or used any of these tools before? What do you think their purpose is?



Reading:

3. Skim the text and choose the best title that summarizes the main idea:

- a) How to Write Code with Jira and Asana
- b) Overview of Three Popular Project Management Tools
- c) Detailed History of Software Development
- d) Benefits of Kanban Boards in Agile Projects

💬 *Discuss your answer with a partner.*

5.3.

In today's fast-paced work environment, project management tools play a crucial role in helping teams organize tasks, track progress, and collaborate more effectively. Jira, Trello, and Asana are among the most widely used tools. Each tool offers unique features tailored to different types of projects and teams.

Jira

Jira is a highly customizable tool used primarily in Agile project management, particularly for software development. It allows teams to manage software development from planning to release. Its key features include:

- Kanban boards and scrum boards to visualize tasks.
- Sprints for setting work periods for task completion.
- Backlog management to prioritize tasks for future development.
- Integrations with tools like GitHub and Slack.
- Reporting features that include dashboards and detailed performance metrics.

Jira is especially useful for teams that use Agile methodologies, as it allows them to easily plan sprints, assign tasks, and monitor project progress in real time.

Trello

Trello is a flexible, easy-to-use project management tool that is great for organizing any kind of task, from business projects to personal to-do lists. Its main feature is its Kanban board layout, where tasks are organized in cards that can be moved through columns representing different stages of completion. Trello's features include:


- Task assignment using cards.
- Collaboration features, where team members can add comments, attach files, and mention each other on tasks.
- Checklists for breaking down tasks into smaller steps.
- It supports integration with platforms like Google Drive, Slack, and Dropbox.
- Automated task workflows through automation (e.g., moving cards to specific columns based on certain triggers).

Trello is often favored by teams looking for a simple, visually-oriented tool to manage tasks across different projects.

Asana

Another useful tool for managing projects is Asana, which helps teams keep track of their work. Asana provides more detailed task management and allows users to view tasks in several ways (list view, calendar view, or Kanban boards). Its features include:

- Task assignment with deadlines, priority levels, and project stages.
- Project timelines similar to Gantt charts, providing a visual overview of a project's progress.
- Notifications to alert team members of upcoming deadlines or task changes.
- Workflow management tools for automating repetitive tasks.
- Integrations with popular platforms like Microsoft Teams, Zoom, and Salesforce.

Asana is widely used by teams looking for a comprehensive project management solution that allows them to organize complex tasks, track deadlines, and ensure all project members stay informed. 

Close Reading

4. Read the text again and match each feature below to the correct tool or tools: Jira, Trello, or Asana. **Some features may apply to more than one tool.**

Feature	Tool(s)
1. Offers Kanban boards to visualize task progress.	_____
2. Commonly used in Agile software development with sprints and backlogs.	_____
3. Allows team members to comment on tasks and attach files directly.	_____
4. Provides visual timelines similar to Gantt charts for tracking progress.	_____
5. Useful for personal to-do lists and business projects with a simple visual layout.	_____
6. Offers performance reports, dashboards, and integration with GitHub.	_____
7. Automates workflows and sends deadline reminders.	_____
8. Enables multiple task views: list, calendar, and board.	_____
9. Integrates with platforms like Google Drive, Slack, or Dropbox.	_____
10. Helps organize tasks using cards that can be moved between columns.	_____

Vocabulary

5. Match the highlighted words in the text below with their definitions.

Word/Phrase	Definition
1. kanban board	a) automated alerts that inform team members about task updates or deadlines
2. sprint	b) a real-time display of important project metrics and data
3. backlog	c) working together as a team to achieve a common goal
4. workflow	d) the ability of software to connect and work with other tools or platforms
5. task assignment	e) allocating specific tasks to team members in a project
6. integration	f) a series of steps or processes needed to complete a task
7. collaboration	g) a list of tasks or work items that need to be addressed
8. dashboard	h) a set period during which specific tasks must be completed in agile project management
9. notifications	i) the use of software to perform repetitive tasks without manual intervention
10. automation	j) a visual tool to organize tasks and workflows using cards and columns

6.  Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3https://quizlet.com/948675490/match?funnelUUID=38e7a450-6168-45c6-b7c3-53ad5af8e52b>



7. Complete the following text by filling in the gaps with the appropriate words from the list:

List of Words: kanban board, collaboration, backlog, sprints, workflow, automation, notifications, dashboard

1. Trello uses a _____ to visually track tasks as they move from start to completion.
2. In Jira, teams can organize and prioritize upcoming tasks using the _____.
3. Asana sends _____ to team members to keep them informed about project

changes and deadlines.

4. Teams using Agile methodologies often organize their work into _____, which are set periods of time to complete specific tasks.
5. Automating repetitive tasks in Trello is possible using _____, helping to save time and reduce errors.
6. A real-time _____ in Jira allows project managers to monitor the overall health and progress of the project.
7. Clear _____ is key to ensuring that all team members are on the same page and working toward shared project goals.
8. Asana's _____ management tools allow teams to automate common tasks and streamline the completion of work.

8. Read the text and match each project management tool to its description:

Key Steps for Selecting the Right Project Management Tool

1. Jira 2. Trello 3. Asana

- a. A project management platform that offers different views, including lists, calendars, and timelines.
- b. A highly customizable tool for Agile project management, especially in software development.
- c. A flexible, visual tool organized around boards and cards for task management.

Identify Your Project Requirements: Before selecting a tool, it is important to understand the specific needs of your project. Are you managing a software development project or a marketing campaign? Do you need a tool that supports Agile methodologies, or are you looking for something simpler?

Consider Team Size and Workflow: The size and structure of your team will impact the tool that works best for you. Jira is ideal for larger, more technical teams, while Trello is better suited for smaller teams or simple workflows.

Look for Customization and Integrations: Make sure the tool can be customized to fit your needs and integrates with other software that your team is using (e.g., Slack, Google Drive, or GitHub).

Evaluate User Experience: The tool should be user-friendly and intuitive. It's important that team members can quickly understand how to use it without extensive training.

Test Automation Features: Automation can reduce the amount of manual work required for routine tasks. Look for tools like Trello and Asana, which offer automation for repetitive tasks.

Monitor and Adjust: Once you've selected a tool, monitor its usage and effectiveness. Make adjustments as needed to ensure that the tool supports your team's workflow and helps to improve productivity.

9. 🎲 Question Practice Game

🎲 Game Instructions:

Work in pairs or small groups. **Roll a dice** and check the **dice key** on the right 🎲.

Choose a statement from the text above.

Make a question based on your dice result and the chosen statement.

🎲 Dice Roll for Question Type

Use a **6-sided dice** with this **key**:

1= Yes/No Question

2= Wh-Question

3= Tag Question

4= Subject Question

5= Choice Question

6= Any type (Student's choice)

🔍 Extra Online Practice

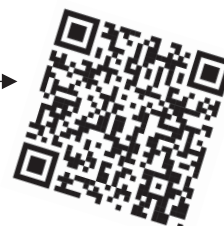
Vocabulary:

[Quizlet – Unit 5.3 Flashcards & Learn](#)

Grammar:

Question tags: Aren't you? don't you?

[Exercises 1–3 on Test-English.com](#)



☒ Recommended activity before starting Revision Unit 5:

Play Quizlet Live (**INDIVIDUAL MODE, 3 TIMES**) using vocabulary from Unit

5.3: 🔗 <https://quizlet.com/ua/932616895/unit-31-flash-cards/?i=j03j6&x=1jqt>

Vocabulary Revision

Unit 5

 1. WORK IN PAIRS. TAKE TURNS PLAYING THE ROLES OF **DESCRIBER** AND **GUESSER** USING VOCABULARY FROM UNIT 5.

◇ Detailed instructions can be found on page 37

Term	Definition
lifecycle	the series of changes that a product or system goes through during its existence
feasibility	the practicality of a proposed plan or method
deployment	the release or distribution of software for use
analysis	the detailed examination of the elements or structure of something
requirement	something necessary or compulsory in a project or system
maintenance	the process of preserving or keeping something in good condition
design	the process of creating a plan or drawing to show the function or workings of a system
implementation	the process of putting a decision or plan into effect; execution
testing	the process of evaluating the performance and quality of a system or product
stakeholder	a person or group with an interest in the success of a project
constraint	a limitation or restriction on time, resources, or scope
milestone	a significant point or event in a project
resource allocation	the process of assigning and managing assets in a way that supports project goals
deliverable	a tangible or intangible product/result produced as part of a project
contingency	a plan designed to take account of possible future events or circumstances
dependency	a relationship between tasks, where one task relies on another to begin or complete
stakeholder	a person or group with an interest in the outcome of the project
risk management	identifying, assessing, and controlling threats to a project's objectives
baseline	an initial plan or budget used for comparison as the project progresses
iteration	the repetition of a process to achieve a desired outcome
kanban board	a visual tool to organize tasks and workflows using cards and columns
sprint	a set period during which specific tasks must be completed in agile project management
backlog	a list of tasks or work items that need to be addressed
workflow	a series of steps or processes needed to complete a task
release	A version of the software made available for users

2. THREE-SENTENCE CHALLENGE. Each participant **selects three words** from the shared word list (*pick the ones you find most challenging*) and forms three sentences using these words. Work in pairs. Read your sentences to your partner, replacing the chosen word with "gap" while reading. Your partner needs to guess the missing word.

3. PLAY THE QUIZLET LIVE GAME (TEAM MODE)
to review the vocabulary learned in **UNIT 5**

<https://quizlet.com/ua/949499007/unit-5-51-52-53-flash-cards/?i=j03j6&x=1jqt>



4. TAKE the UNIT 5 VOCABULARY QUIZLET TEST
to check your understanding of key terms from this UNIT. <https://quizlet.com/948371787/test?answerTermSides=6&promptTermSides=6&questionCount=30&questionTypes=15&showImages=true>
<https://quizlet.com/949499007/test?answerTermSides=2&promptTermSides=2&questionCount=30&questionTypes=15&showImages=true>



5. 🎲 VOCABULARY GAME: ALIAS

Play a fun team-based vocabulary challenge!

♦ See full game instructions on page 37.

Get ready to explain, guess, and compete using key terms from Unit 5!

For Printable Vocabulary Materials for the Game “Alias,” go to page 235.

6

UNIT

COMMUNICATION TOOLS



This unit introduces learners to modern communication platforms and their role in software development and project management. Students will explore professional chat tools such as Slack and Microsoft Teams, practice effective communication strategies in chat apps, and learn best practices for using email and shorthand in workplace contexts.

Unit overview

6.1 Overview of team chat tools: Slack, Microsoft Teams

Lesson outcome: Learners can identify the features and functions of professional chat platforms and explain how they support collaboration in software development.

Skills practised

Reading (for gist and detail) Overview of Team Chat Tools

Speaking: discussion in pairs, role-play

Vocabulary: chat tool functions and collaboration terms

Grammar: questions: different types

6.2 Effective use of professional chat apps

Lesson outcome: Learners can discuss key practices for effective chat communication

Skills practised:

Reading: identifying good practices in slack communication

Speaking: Role-play: Chat Simulation

Vocabulary: Chat communication

Grammar: indirect questions

6.3 Best practices for team conversations via email and chat including common acronyms and shorthand (e.g., ASAP, BTW, FYI, EOD, NRN)

Lesson outcome: Learners can identify different types of team conversations and select the most appropriate one for a given project.

Skills practised:

Skim reading: identifying and matching workplace communication best practices

Speaking: pair discussions and role-plays comparing email vs. chat and explaining acronyms in context

Vocabulary: professional email/chat acronyms, shorthand

Grammar: Present Simple vs Present Continuous

Lesson 1

Overview of team chat tools: Slack, Microsoft Teams

Lead-in 1. 🗣️ Discuss in pairs or small groups:

1. **Which communication apps do you use most often — Discord, Instagram, Telegram, or something else?**
– *What do you usually use them for (chatting, file sharing, video calls, etc.)?*
2. **Have you ever used professional team chat tools like Slack or Microsoft Teams?**
– *If yes, how are they different from personal apps? If no, what do you think they might be used for?*
3. **What features would a good team chat tool need to support collaboration in a workplace or study setting?**
– *Think about notifications, channels/groups, task management, file sharing, etc.*

2. 📺 Watch the video: Best Team Communication Tools

While watching, tick (✓) the tools mentioned in the video.

- | | |
|---------------------------------------|--|
| <input type="checkbox"/> 1. Pumble | <input type="checkbox"/> 6. MS Teams |
| <input type="checkbox"/> 2. TeamSpeak | <input type="checkbox"/> 7. Mattermost |
| <input type="checkbox"/> 3. Slack | <input type="checkbox"/> 8. Basecamp |
| <input type="checkbox"/> 4. WhatsApp | <input type="checkbox"/> 9. Discord |
| <input type="checkbox"/> 5. Revolt | |

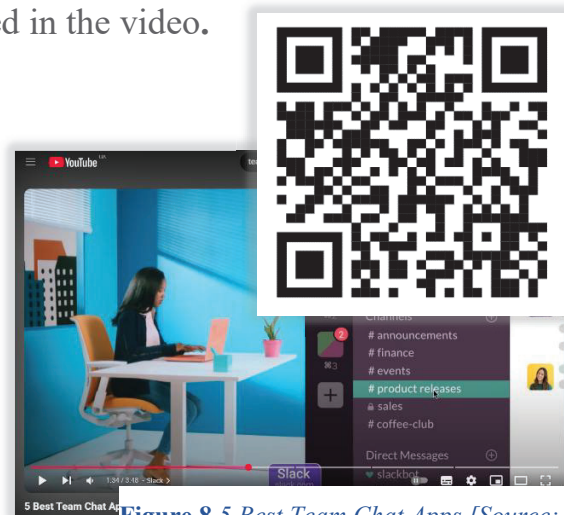


Figure 8 5 Best Team Chat Apps [Source: <https://youtu.be/hxyoVmMXmB8?t=5>]

3. Discuss in pairs:

1. Which of these tools have you heard of or used before?
2. Which tool do you think is best for student group projects? Why?

4. Watch the video again. Complete the text using the words in the box.

channels, mentions, open source, gamers, servers, departments, workplace, governmental direct messages |

A. ⌚ ___ Slack helps teams collaborate in organized _____ based on projects, _____ or office location.

B. ⌚ 1 Pumble offers collaboration through public and private channels, as well as individual and group _____.

C. ⌚ ___ Microsoft Teams is a collaboration app with 44 million users. It is primarily designed for larger teams, including companies with over 1,000 employees and _____ organizations.

D. ⌚ ___ Mattermost is an _____ platform that provides secure team communication for DevOps, IT security teams and governmental organizations.

E. ⌚ ___ Discord is popular among _____ but can also be used by business teams for communication via voice, video, or text.

F. ⌚ ___ Pumble also offers features such as web and mobile notifications, _____, file sharing, user groups and inviting external partners as guests.

G. ⌚ ___ The self-hosted version of Pumble allows organizations to **host** their workspaces on their own _____.

H. ⌚ ___ Slack is a _____ communication tool that connects users with their teams and essential tools, enabling seamless collaboration regardless of location or role.

 **Extra Challenge:**

Match the Sentences to the Timestamps

⌚ **Timestamps:**

1 (0:36-0:44); **2** (0:45-0:53);

3 (1:05-1:09); **4** (1:19-1:26);

5 (1:28-1:33); **6** (1:53-2:27);

7 (2:27- 2:58); **8** (3:03-3:14)

Reading 5. Read the text. Which feature is mentioned in both Slack and Teams?

6.1.

Overview of Team Chat Tools: Slack and Microsoft Teams

Effective communication is crucial for teams working on projects, especially in distributed environments. Two of the most widely used team chat tools are **Slack** and **Microsoft Teams**, which allow for **seamless** collaboration, messaging, and file sharing. While both tools are designed to improve team communication, they each offer unique features suited to different types of teams and workflows.

Slack is a popular team communication platform that organizes conversations into **channels** and **threads**. It's designed to reduce the need for long email chains by keeping all relevant communication in one place. Key features of Slack include:

Channels:

- Public or private channels dedicated to specific projects, teams, or topics.

Direct messages (DMs):

- Private conversations between individuals or small groups.

File sharing:

- Allows users to upload and share files, documents, and images within chats.

Integration with third-party tools

- like Google Drive, Trello, and GitHub, making collaboration more seamless.

Search functionality:

- Helps users quickly find specific messages, conversations, or files.

Notifications:

- Customizable alerts to notify team members of new messages or updates.

Video conferencing:

- Slack offers built-in video call functionality for meetings and team check-ins.

Slack is particularly useful for teams that prioritize real-time collaboration and want to keep their communication organized by topic or project.

Microsoft Teams is a comprehensive collaboration tool that integrates directly with the Microsoft 365 suite. Teams is more than just a chat app; it includes document collaboration, meetings, and video conferencing. Its main features include:

Channels:

- Similar to Slack, Teams allows for channel-based conversations, both public and private.

Direct messages (DMs):

- Enables private messaging between team members or small groups.

Integration with

- the entire Microsoft 365 suite, including Word, Excel, and OneDrive.

File sharing:

- Users can upload and collaborate on documents directly within chats.

Search functionality:


- Allows users to quickly locate previous conversations and files.

Video conferencing:

- Teams supports high-quality video calls, group meetings, and webinars.

Collaboration on documents:

- Microsoft Teams integrates directly with Office tools, allowing real-time collaboration on documents, spreadsheets, and presentations.

Microsoft Teams is an ideal tool for organizations that already use Microsoft 365 and want a communication platform that integrates with their existing workflows. 

6. Read the text again. Decide if the statements below are TRUE or FALSE.

1. Slack allows users to share files, documents, and images directly within chats.
2. Microsoft Teams does not support video conferencing.
3. Slack integrates with Microsoft Word and Excel for real-time document collaboration.
4. Both Slack and Microsoft Teams organize communication using public and private channels.

Vocabulary

7. Match the highlighted words in the text with their definitions.

Word/Phrase	Definition
1. channels	a) happening without any sudden changes, interruption, or difficulty
2. direct message (DM)	b) the tool that allows users to find specific conversations or files within the chat history
3. integration	c) alerts that inform users of new messages or activities within the app
4. threads	d) the feature that allows users to upload and share documents, images, or other files
5. file sharing	e) a series of related messages that allow team members to follow a specific conversation topic
6. notifications	f) the ability of chat tools to connect with other platforms, such as Google Drive or Trello
7. search functionality	g) a private message sent between two individuals or a small group in a chat app
8. seamless	h) dedicated spaces within a chat app where team members can discuss specific topics or projects

8. 🎮 Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3https://quizlet.com/948676692/match?funnelUUID=2c53ac92-170c-4cb7-bea3-1d617266f538>



9. Complete the following text by filling in the gaps with the words from the *List of Words*:

| channels, direct messages, integration, file sharing, notifications, threads, search functionality, video conferencing |

1. In both Slack and Teams, conversations are organized into _____, where team members can discuss specific topics or projects.
2. Team members can send private _____ to one another for direct communication.
3. Slack and Teams allow users to conduct _____, which is useful for remote meetings and

- check-ins.
4. Real-time collaboration is enhanced by _____ with external tools like Google Drive and Microsoft 365.
 5. Users can upload and share documents through the _____ feature, making collaboration easy.
 6. Team members can receive _____ to stay informed about updates or new messages.
 7. The _____ feature helps team members keep discussions organized by focusing on a specific topic.
 8. Both tools have _____ that makes it easy to locate past conversations and files.

10. 🎲 Question Practice Game

🎲 Game Instructions:

Work in pairs or small groups. **Roll a dice** and check the **dice key** on the right 🎲.

Choose a statement from the exercise 9. **Make a question** based on your dice result and the chosen statement.

🎲 Dice Roll for Question Type

Use a **6-sided dice** with this **key**:

1= Yes/No Question

2= Wh-Question

3= Tag Question

4= Subject Question

5= Choice Question

6= Any type (Student's choice)

11. Choose three words from Exercise 7. Create your own **questions** using those words. 🗣️ Then, **ask your partner** and discuss their answers.

🔍 Extra Online Practice

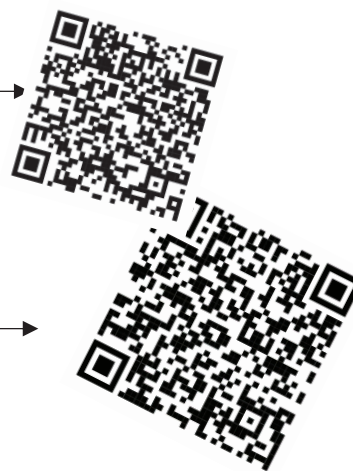
Vocabulary:

[Quizlet – Unit 6.1 Flashcards & Learn](#)

Grammar:

Questions: Different types

[Exercises 1–3 on Test-English.com](#)



☒ Recommended activity before starting Lesson 2:

Play Quizlet Live (INDIVIDUAL MODE, 3 TIMES) using vocabulary from Unit 6.1:

🔗 <https://quizlet.com/ua/932616895/unit-31-flash-cards/?i=j03j6&x=1jqt>

<https://quizlet.com/ua/948676692/unit-61-flash-cards/?i=j03j6&x=1jqt>

Lesson 2

Effective of use professional chat apps

Lead-in 1.  Discuss in pairs or small groups:

1. What are the **main differences** between using chat apps like Discord or Telegram **and** professional tools like Slack or Microsoft Teams?
2. What features do you think are **essential** in a chat app for work or study? (e.g., file sharing, video calls, message search, channels)

Reading 2. Read the text. Decide if the statements below are TRUE or FALSE.

6.2.

Key Practices for Effective Chat Communication:

Keep messages **concise**: In a fast-paced environment, **brevity** is essential. Keep your messages clear and to the point, focusing on the main information or **matter at hand**. Avoid adding too much detail unless necessary.

Use threads to maintain context: Many chat apps like Slack and Teams allow for **message threading**, which helps to organize discussions. When responding to specific questions or comments, use threads to keep the conversation focused and avoid clutter in the main chat.

@Mentions to grab attention: Use @mentions to notify team members of important information that requires their attention. For example, "@Tom, can you review this document?" or "@All, please note the updated meeting time."

Emojis and tone: While emoji etiquette is becoming more acceptable in professional environments, be mindful of how and when you use emojis. They can help convey tone and emotion in text-based conversations but should be used appropriately depending on the context. For example, a thumbs-up emoji is a quick way to confirm you've received a message, but overuse of informal emojis can come off as unprofessional.

Read the statements below and decide if they are **True (T)** or **False (F)** based on the text.
Correct the false statements.


1. Using threads in chat apps helps keep conversations organized and easy to follow.
2. It is always appropriate to use multiple emojis in a professional message to appear friendly.
3. Read receipts should be used carefully and not assumed to mean that someone will respond immediately.
4. You should avoid muting notifications because it's important to respond to every message right away.
5. In professional chats, it's acceptable to write messages in all to show importance.

Set expectations for response time: In professional chat environments where communication is often asynchronous, it's essential to communicate your expected response time. For urgent matters, be specific about when a reply is required. If a task is urgent, clearly state when you need a response. For example, "Please send the report by 3 PM today" or "I'll need your feedback by tomorrow EOD."

Mute notifications to avoid distractions: During meetings or focused work time, you may want to use the "mute" feature to pause notifications and avoid distractions. Let your team know when you're available for communication to maintain a balance between staying informed and minimizing interruptions.

Use read receipts strategically: Some chat apps include read receipts that let you know when someone has seen your message. This can help you gauge whether

your message has been received, but it's important not to over-rely on this feature to assume immediate action.

Be mindful of the tone: Text-based communication can often lead to misunderstandings if the tone of the message isn't clear. Avoid using all caps (capital letters), which can be interpreted as shouting, and be considerate of how your message might be perceived by others. 

3. Identify Good Practices in Slack Communication.

Read the example conversation on the right and **answer the questions**.

1. Which message in the conversation includes a clear deadline?
2. Why did Anna use @Tom and @Lisa in her message?
3. What is the purpose of Anna's status message and muting notifications?
4. Why did Anna use a DM for her question to Tom? Was that a good choice? Why or why not?
5. How did Anna ask team members to confirm attendance at the meeting? Was this an effective way to get quick responses?

4. Discussion:

- What emoji/tone would be acceptable in your work or study context? Why?
- When (if ever) is it appropriate to use **ALL CAPITAL LETTERS** in a chat message? What impression can it create?

Slack Conversation Example

#project-website-redesign (Public Channel)

Anna:

Hi team! 😊 The updated wireframes for the homepage are ready. Please review them by **TOMORROW EOD**.

🔗 [Link to wireframes]

@Tom, can you check the navigation section specifically?

@Lisa, I'd appreciate your input on the colour scheme.

Tom (in thread):

Thanks, Anna! I'll review the navigation today and get back to you by 4 PM. 👍

Lisa (in thread):

Got it! I'll check the colours and give feedback by noon tomorrow. 🎨

Anna (later):

@All Reminder: Our **client review meeting** is scheduled for **FRIDAY AT 10 AM**. Please confirm attendance by reacting with ☒

Tom reacted with ☒

Lisa reacted with ☒

Anna (DM to Tom):

Hey, just checking - did you get a chance to test the mobile version?

Tom:

Not yet - I'll do that after the design review. I'll send you an update by 5 PM.

Anna:


👍 No rush, just wanted to align timelines. Thanks!

Anna sets status: 🙋 **In focus mode - back at 3 PM**
(Notifications muted)

5. In pairs, simulate the following scenarios

(If possible, use professional communication platforms such as Slack, Microsoft Teams, Discord, or similar tools to complete this task)


Scenario 1:

- Your colleague missed an important update.
 Write a **concise message** using an **@mention** to notify them and ask for confirmation that they've seen it.


Example:

@Lena, just checking if you saw the update from this morning. Let me know, thanks!

Scenario 2:

- You're discussing a project in a busy chat.
 Start a new thread to ask a specific question related to the discussion without cluttering the main conversation.

Scenario 3:

- You're going offline for a few hours.
 Write a **professional message** to let your team know. Mention when you'll be available again and **mute notifications**.

Example:

Hi team, I'll be offline from 2–5 PM for focused work. Feel free to leave messages — I'll check them as soon as I'm back. 🚫🔔

6. Role-play: Chat Simulation. Work with your partner to **create a short chat conversation** about a fictional project (e.g., *planning an event, developing an app, launching a product*).

- Each of you should write **at least 5 messages**.
- Include features like: **@mentions**, **emojis**, or **file sharing** (e.g., "Here's the doc 📎").
- Try to apply some of the **chat communication tips** from the previous tasks.
(if possible, use professional communication apps).
- Present your chat by reading it aloud in pairs.

Vocabulary

7. Match the highlighted words in the text with their definitions.

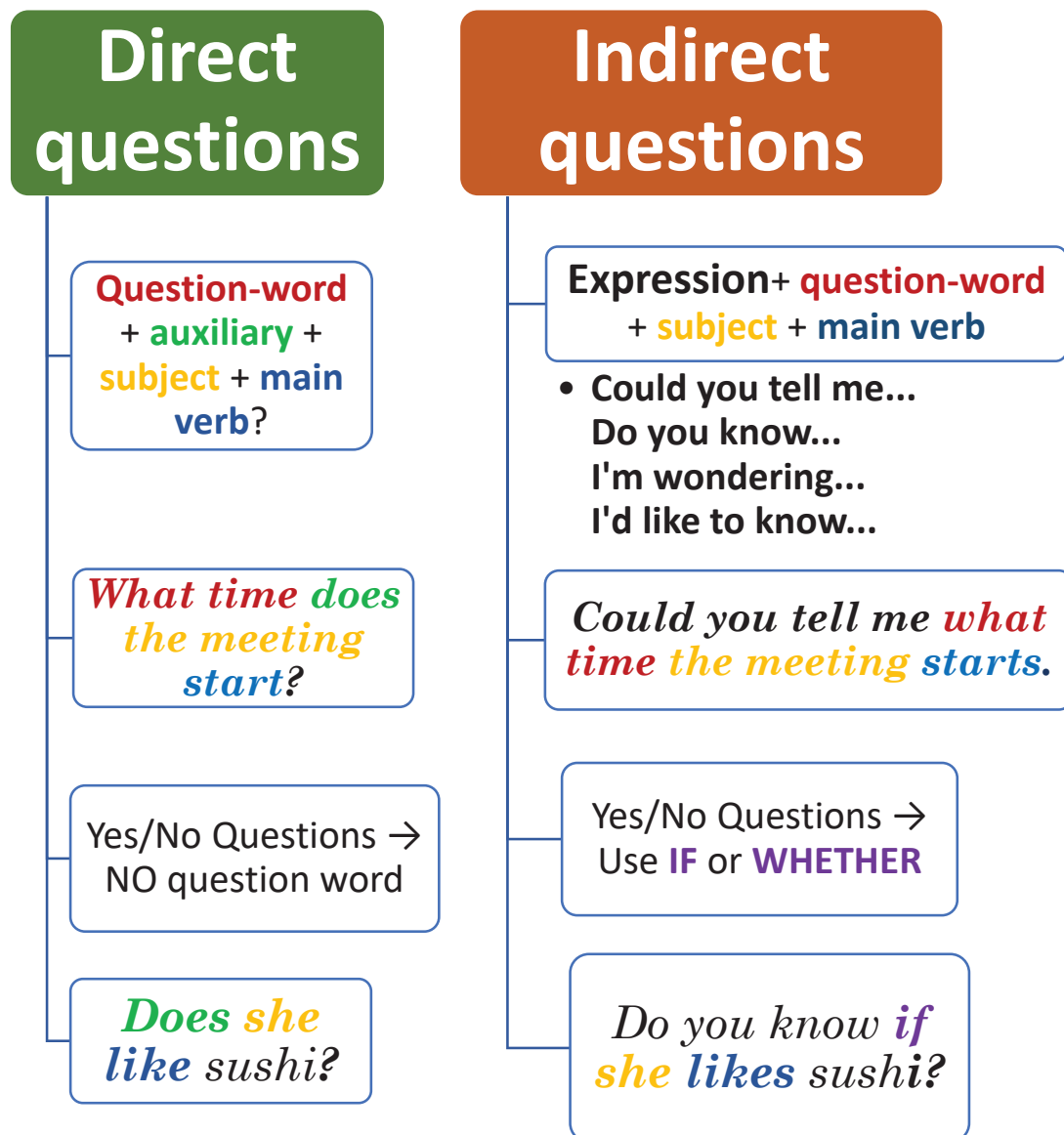
Word/Phrase	Definition
1. emoji etiquette	a. the job or matter that is important at the present moment
2. asynchronous communication	b. short and clear, expressing what needs to be said without unnecessary words
3. message threading	c. a feature that indicates whether the recipient has seen the message
4. gauge (v)	d. caring about and respectful of others:
5. distractions	e. a feature that allows users to directly notify or reference a specific person within a conversation
6. brevity	f. keeping conversations organized by replying to specific messages within a chat
7. concise	g. a form of communication where participants do not need to be present at the same time
8. the matter at hand	h. guidelines for appropriate and professional use of emoticons in work-related conversations
9. @mentions	i. to calculate an amount, especially by using a measuring device
10. read receipts	j. using only a few words or lasting only a short time
11. considerate	k. things that make it difficult to think or pay attention

8. 🖱️ Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3><https://quizlet.com/948678035/match?funnelUUID=59a1f739-6a57-474c-9719-ee012159d459>



Grammar for Revision Indirect questions



9. Rewrite the **direct questions** below as **indirect questions**

1. **Direct:** Where is the latest version of the report?

Indirect: Could you tell me _____?

2. **Direct:** When will the client presentation be ready?

Indirect: Do you know _____?

3. **Direct:** Is Tom available for a quick meeting now?

Indirect: Do you happen to know _____?

4. **Direct:** What time does the daily stand-up start?

Indirect: Can you remind me _____?

5. **Direct:** Who is responsible for updating the project timeline?

Indirect: Could you let me know _____?

6. **Direct:** Have you shared the wireframes with the design team?

Indirect: I was wondering _____.

7. **Direct:** Why was the task moved to the backlog?

Indirect: Do you have any idea _____?

8. **Direct:** Will we use Zoom or Teams for the meeting?

Indirect: Do you know _____?

9. **Direct:** Where can I find the project files?

Indirect: I'm not sure if you know, but could you tell me _____?

10. 🎲 Indirect Questions Challenge

? Choose the correct indirect question for each direct question below.

You can also play the interactive version on **Kahoot!**

🎮 **Link for Teachers** (use this link to launch the quiz): Indirect Questions Kahoot Game

[<https://create.kahoot.it/share/indirect-questions-challenge/deff670b-95e7-4878-86d4-32ed3ec505df>]



1. *What time is the team meeting?*

- A) Do you tell me what time the team meeting is?
- B) Could you tell me what time is the team meeting?
- C) Could you tell me what time the team meeting is?
- D) What is the time the team meeting is?

2. *Where can I find the project files?*

- A) Do you know where I can find the project files?
- B) Do you know where can I find the project files?
- C) Where I can find the project files do you know?
- D) Can I know where I find the project files?

3. *Who is managing the Jira board?*

- A) Can you tells who is managing the Jira board?
- B) Do you know who is managing the Jira board?
- C) Can I know who managing the Jira board?
- D) Do you know who managing the Jira board is?

4. *When does the sprint end?*

- A) Can you tell me when does the sprint end?
- B) Do you know when the sprint ends?
- C) Do you know when does end the sprint?
- D) I wonder when ends the sprint?

! Pay attention to
the correct **word**
order and the
question mark
("?")

5. *Is Slack integrated with Trello?*

- A) Do you know is Slack integrated with Trello?
- B) Do you know whether Slack is integrated with Trello?
- C) Do you know whether is Slack integrated with Trello?
- D) I want to know is Slack integrated with Trello?

6. *How do I mute notifications in Teams?*

- A) Do you know how do I mute notifications in Teams?
- B) Could you tell me how to mute notifications in Teams?
- C) Do you know how I do mute notifications in Teams?
- D) Could you tell me how can I mute notifications in Teams?

7. *Can I join the video call later?*

- A) I wonder if can I join the video call later.
- B) I want to know if I can join the video call later.

- C) I want to know can I join the video call.
D) I'm asking if can I join the video call.
8. *Did you send the report to the client?*
A) Do you know did you send the report to the client?
B) I wonder did you send the report to the client?
C) I wonder if you sent the report to the client.
D) Tell me did you send the report to client?
9. *Why is the deployment delayed?*
A) Do you know why is the deployment delayed?
B) Do you know why the deployment is delayed?
C) Do you know the deployment is delayed why?
D) I'd like to know why is delayed the deployment.
10. *What tool do we use for time tracking?*
A) Can you tell me what tool do we use for time tracking?
B) Can you remind me what tool we use for time tracking?
C) Do you know what tool are we using for time tracking?
D) Do you happen to know what do we use for time tracking?
11. *What time is the team meeting?*
A) Do you know what time is the team meeting?
B) Do you know what time the team meeting is?
C) Can you tell me what time is the meeting team?
D) I wonder what time is it the meeting?
12. *Where can I find the project files?*
A) Do you know where can I find the project files?
B) Would you mind telling me where I can find the project files.
C) Can you tell where I find the files?
D) Would you mind telling me where I can find the project files?

Speaking

11. Complete the prompts below with your own ideas

1. Could you tell me...
2. Do you know...
3. Can you remind me...
4. I was wondering...
5. Would you mind telling me...
6. Do you happen to know...
7. I'd like to know...
8. Could I ask you...
9. I'm not sure if you know, but...

 Choose at least three questions and ask your partner.

Examples:

*Would you mind telling me **how to set up a channel in Slack?***

Extra Online Practice

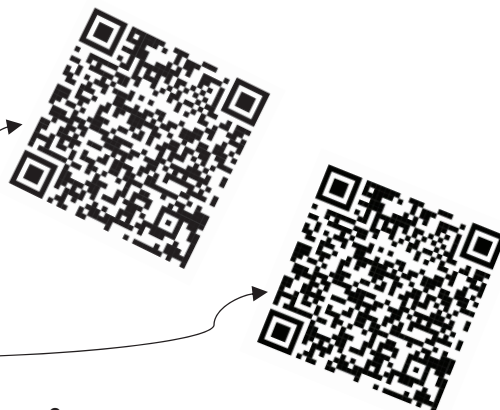
Vocabulary:

[Quizlet – Unit 6.2 Flashcards & Learn](#)

Grammar:

Indirect questions

[Exercises 1–3 on Test-English.com](#)



☒ Recommended activity before starting Lesson 3:

Play Quizlet Live (**INDIVIDUAL MODE, 3 TIMES**) using vocabulary from Unit 6.2:

 <https://quizlet.com/ua/932616895/unit-31-flash-cards/?i=j03j6&x=1jqt>

<https://quizlet.com/ua/948678035/unit-62-flash-cards/?i=j03j6&x=1jqt>

Lesson 3

Best Practices for Team Conversations via Email and Chat

Lead-in 1. Match these communication methods to their best use cases:

(Email, Chat, Video Call, Phone Call, Project Management Tool)

Example use cases:

- a. Giving detailed feedback
- b. Asking a quick question
- c. Scheduling a meeting
- d. Reporting a problem
- e. Sharing files with a large group

Reading:

2. Skim the text and choose the best title that summarizes the main idea of the text.

- a) Email vs. Chat: Choosing the Right Tool for Professional Communication
- b) Common Mistakes in Workplace Messaging
- c) Organizing Project Files via Chat Tools
- d) Formal Writing in Emails: A Beginner's Guide

6.3.

Why did you choose this title? What keywords in the text helped you decide?

In a professional setting, effective communication through email and chat is critical to ensure that tasks are completed, decisions are made efficiently, and misunderstandings are minimized. While chat tools like Slack or Microsoft Teams are typically used for quick, informal communication, email is often reserved for more formal conversations. Both mediums have unique guidelines and best practices.

Key Practices for Email and Chat Communication:

Choose the Right Medium: Use email for formal or detailed conversations, especially when communicating with external stakeholders or when you need to provide a record of the conversation.

Use chat for quick updates, questions, or informal team interactions.

Keep Subject Lines and Chat Topics Clear: In emails, use clear and specific subject lines to help the recipient immediately understand the topic. Example: “Project Update: Q3 Deadline Adjustment.”

In chat, label conversations to avoid confusion. Example: “[Design Feedback] Are these images finalized?”

Use Professional Language: Whether in email or chat, it's important to maintain a formal tone. Avoid slang or overly casual phrases, especially with clients or upper management.

Limit Acronyms and Shorthand: While acronyms like FYI, ASAP, or EOD are commonly understood, avoid using them **excessively** or in formal emails where clarity is **paramount**. Some acronyms can be misunderstood by colleagues, particularly those in different cultures or locations.

When using acronyms, ensure they are commonly recognized by the team. If not, provide a brief explanation the first time you use them.


Replying vs. Replying All: In emails, Reply All should only be used when everyone in the thread needs to see your response. For most replies, only the original sender needs a response. In chat, respond to individuals directly or use threads to keep conversations organized.

Be Concise: Both in email and chat, get to the point quickly. Avoid sending long paragraphs, and in chat, stick to one or two sentences per message. For example, instead of sending, “Hi, I just wanted to ask you if you have finished working on that report you were working on last week and if so, could you send it to me before lunch?”, try, “Hey, have you finished the report from last week? Please send it over by noon.”

Use Appropriate Sign-Offs and Greetings: In email, include a greeting and a closing (e.g., “Hello, I hope you’re doing well” or “Thank you for your help”). In chat, you can be more informal, but always remain respectful (e.g., “Thanks!” or “Talk soon!”).

Acknowledge Receipt: Whether in email or chat, acknowledging the receipt of important information helps the sender know that their message was received. A simple “Got it” or “Noted” can be enough.

Watch Your Tone: Without the benefit of facial expressions and tone of voice, written communication can often be misinterpreted. Avoid all caps, which can seem aggressive. Use emojis sparingly to convey tone in chat, but avoid them in professional emails unless your company culture is more casual.

Use BCC/CC Appropriately: CC (Carbon Copy) keeps other people informed but doesn’t require them to respond. BCC (Blind Carbon Copy) is useful for emailing large groups without exposing everyone’s email address. 

Reading for Detail

3. Read the text again carefully. Match each best practice (1–6) with its purpose or benefit (A–F). One extra is given as a challenge.

Practice	Purpose/Benefit
1. Use clear subject lines or labels	a. Keeps messages short and easy to understand
2. Acknowledge receipt	b. Prevents misunderstanding due to unclear tone
3. Choose the right communication tool	c. Helps sender know their message was received
4. Be concise	d. Ensures message is sent to appropriate people
5. Watch your tone	e. Helps recipient immediately grasp the topic
6. Use CC/BCC appropriately	f. Avoids delays in reading long emails

Vocabulary

4. Match the highlighted words in the text with their definitions.

Word/Phrase	Definition
1. EOD	a) adding someone to an email to keep them informed without needing a direct response
2. FYI	b) a style of communication used in business or professional settings, marked by politeness
3. NRN	c) Too Long; Didn't Read (used to summarize long content in brief)
4. ASAP	d) By the Way (used to introduce an additional piece of information)
5. BTW	e) As Soon As Possible (used to express urgency in completing a task or action)
6. TL;DR	f) No Response Needed (used to indicate that the message is informational, and no reply is required)
7. formal tone	g) For Your Information (used to share important information without expecting a response)
8. cc (carbon copy)	h) End of Day (used to signify the deadline for completing a task by the end of the workday)
9. excessively	i) more important than anything else:
10. paramount	j) in a way that is too much

5. 🖱️ Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3><https://quizlet.com/949497221/match?funnelUUID=140ae2b3-28d7-40e7-8862-351f11d6f25c>



6. Complete the sentences with the correct words from the list.

Words:

| EOD | FYI | NRN | CC | BCC | formal tone | ASAP | btw | excessively | paramount |

1. You can use _____ to copy your supervisor in the email without requiring them to take action.
2. If no response is needed, you can end your message with _____ to let the recipient know.
3. Maintaining a professional image is _____ when communicating with external stakeholders or upper management.
4. Please send the updated report _____, as we need it for the upcoming client meeting.
5. The task must be completed by _____ today, so please prioritize it accordingly.
6. Always use a _____ when writing to clients or external partners to maintain professionalism.
7. I've added some additional notes in the message — _____, you might want to check them before the meeting.
8. When emailing a large group, use _____ to keep recipients' email addresses private.
9. Avoid using acronyms _____, especially in formal emails where clarity is essential.

7. Read the mini-dialogues and fill in the blanks with the correct words from the box. Then role-play with a partner.

Word Bank: | EOD | NRN | excessively | paramount | btw |

Dialogue 1

A: Please finalize the presentation by _____. We need it for tomorrow's review.

B: Okay. Oh, _____, did you see the new branding guidelines? We should apply those.

Dialogue 2

A: I noticed the report had a lot of acronyms. It might confuse the client.

B: You're right. I'll avoid using them _____. Clarity is _____ for this audience.

8. 📝 Optional Writing Task

Write a short, professional email to remind a classmate to complete a task urgently (or create your own realistic scenario).

Send the email to your classmate and **cc** the **teacher at their email address**.

Use at least **three words from** the vocabulary list in **Exercise 4**.

☒ Keep your message concise and polite!

Grammar for Revision Present Simple vs Present Continuous

Present Simple

Form:
Subject + **base verb**
(add **-s** for he/she/it)
*I **work**, she **works**,
they **play***

Use:
Regular actions or habits
Facts and general truths
Timetables/schedules

Examples:
*He studies every day.
The train leaves at 6:30.*

Present Continuous

Form:
Subject + **am/is/are** + **verb** +
-ing
*I **am working**,
she **is playing**,
they **are studying***

Use:
Actions happening **right now**
Temporary situations
Future plans (with time reference)

Examples:
*She is studying at the moment.
We are meeting them tomorrow.*

9. Spot the Error & Justify the Correction

Each sentence below contains a **tense mistake** related to the **present simple** or **present continuous**.

1. **Find the mistake.**
2. **Rewrite the sentence correctly.**
3. **Explain your choice:** Why is the corrected tense correct in that context?

Sentences:

1. He is going to the gym every Saturday.
2. I'm thinking that this idea is better.
3. They build a new house on our street right now.
4. My friend usually is drinking coffee in the morning.
5. Why do you look at me like that?
6. I am not understand what you mean.
7. Sarah cooks dinner at the moment.
8. I take the bus to school this week because my bike is broken.
9. John is always forgetting his keys — it drives me crazy!
10. Does she watching the movie now?

Extension

In pairs, choose 2–3 corrected sentences and **discuss what difference** it would make if the other tense were used (How would the meaning change? Would it still be correct?).

🔍 Extra Online Practice

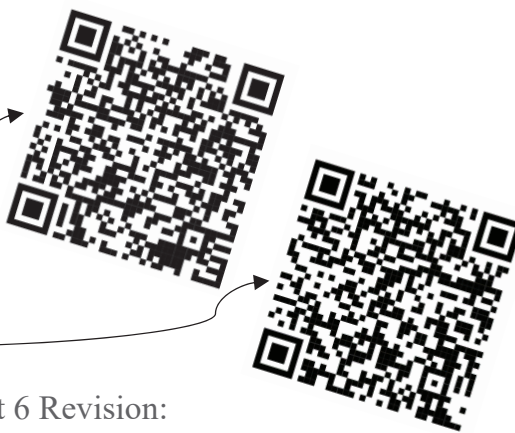
Vocabulary:

[Quizlet – Unit 6.3 Flashcards & Learn](#)

Grammar:

Indirect questions

[Exercises 1–3 on Test-English.com](#)



☒ Recommended activity before starting Unit 6 Revision:

Play Quizlet Live (**INDIVIDUAL MODE, 3 TIMES**) using vocabulary from Unit 6.3:

🔗 <https://quizlet.com/ua/932616895/unit-31-flash-cards/?i=j03j6&x=1jqt>

<https://quizlet.com/ua/949497221/unit-63-flash-cards/?i=j03j6&x=1jqt>

Vocabulary Revision

Unit 6

 1. WORK IN PAIRS. TAKE TURNS PLAYING THE ROLES OF **DESCRIBER** AND **GUESSER** USING VOCABULARY FROM UNIT 6.

◇ *Detailed instructions can be found on page 37*

Term	Definition
channels	dedicated spaces within a chat app where team members can discuss specific topics or projects
direct message (DM)	a private message sent between two individuals or a small group in a chat app
integration	the ability of chat tools to connect with other platforms, such as Google Drive or Trello
threads	a series of related messages that allow team members to follow a specific conversation topic
file sharing	the feature that allows users to upload and share documents, images, or other files
notifications	alerts that inform users of new messages or activities within the app
search functionality	the tool that allows users to find specific conversations or files within the chat history
video conferencing	a feature that allows teams to conduct video meetings directly within the chat platform
emoji etiquette	guidelines for appropriate and professional use of emoticons in work-related conversations
asynchronous communication	a form of communication where participants do not need to be present at the same time
message threading	keeping conversations organized by replying to specific messages within a chat

@mentions	a feature that allows users to directly notify or reference a specific person within a conversation
tone	the overall feel or style of your message, which can affect how it is interpreted by the recipient
read receipts	a feature that indicates whether the recipient has seen the message
concise communication	delivering clear and straightforward messages without unnecessary information or elaboration
mute notifications	temporarily disabling alerts to avoid distraction from non-essential messages
EOD	end of day (used to signify the deadline for completing a task by the end of the workday)
FYI	for your information (used to share important information without expecting a response)
NRN	no response needed (used to indicate that the message is informational, and no reply is required)
ASAP	as soon as possible (used to express urgency in completing a task or action)
BTW	by the way (used to introduce an additional piece of information)
TL; DR	too long; didn't read (used to summarize long content in brief)
formal tone	a style of communication used in business or professional settings, marked by politeness
cc (carbon copy)	adding someone to an email to keep them informed without needing a direct

2. THREE-SENTENCE CHALLENGE. Each participant **selects three words** from the shared word list (*pick the ones you find most challenging*) and forms **three sentences** using these words. Work in pairs. Read your sentences to your partner, replacing the chosen word with "gap" while reading. Your partner needs to guess the missing word.

3. PLAY THE QUIZLET LIVE GAME (TEAM MODE) to review the vocabulary learned in **UNIT 6**

<https://quizlet.com/ua/949497514/unit-6-61-62-63-flash-cards/?i=j03j6&x=1jqt>



4. TAKE the UNIT 6 VOCABULARY QUIZLET TEST to check your understanding of key terms from this UNIT.

<https://quizlet.com/949497514/test?answerTermSides=6&promptestionCount=30&questionTypes=15&showImages=true>



5. 🎲 VOCABULARY GAME: ALIAS

Play a fun team-based vocabulary challenge!

♦ See full game instructions on page 37.

Get ready to explain, guess, and compete using key terms from Unit 6!

For Printable Vocabulary Materials for the Game “Alias,” go to page 235.



This unit introduces learners to presentation skills for IT professionals. Students will explore effective techniques for delivering presentations, learn how to describe and interpret visuals such as charts and diagrams, and practice using digital tools to create engaging and professional slides.

Unit overview

7.1 *Presentation techniques for software engineers*

Lesson outcome: Learners can deliver effective technical presentations tailored to different audiences

Skills practised

Reading: identifying key presentation techniques

Speaking: giving short technical presentations

Vocabulary: presentation terms

Grammar: past simple vs. present perfect

7.2 **How to describe and interpret visuals like graphs, charts, and diagrams**

Lesson outcome: Learners can describe and interpret visuals clearly in presentations

Skills practised:

Reading: recognizing types and purposes of visuals

Speaking: describing and interpreting data from visuals

Vocabulary: chart and graph terminology (e.g., axis, legend, data point)

Grammar: present perfect simple vs. present perfect continuous

7.3 **Using presentation software tools efficiently (e.g., PowerPoint, Keynote, Google Slides, Prezi, Canva)**

Lesson outcome: Learners can create and deliver professional presentations using digital tools effectively.

Skills practised:

Reading: comparing presentation tools and their features

Speaking: presenting with slides and explaining tool use

Vocabulary: software tool terms (e.g., template, transition, animation, collaborate)

Grammar: past tenses (past simple, past continuous, past perfect)

Lesson 1

Presentation techniques for software engineers

Lead-in 1.  Discuss in pairs or small groups:

1. Which presentations have you seen recently (at university, conferences, online)?
2. What makes a presentation engaging? What makes it boring?
3. Do you feel more nervous presenting in English or in your native language? Why?

2. Decide if you agree (☑) or disagree (✗). Be ready to explain.

1. Technical presentations should always include a lot of code.
2. The more text on the slide, the better the audience understands.
3. Storytelling is useful even in technical presentations.
4. Good presenters don't need to prepare extra slides for Q&A (Questions and Answers).

3. Read the text. While reading, check which of the statements from the previous exercise the author would agree or disagree with. Underline the parts of the text that support your answer.

7.1.

Presentation Techniques for Software Engineers

Delivering effective presentations is an important skill for software engineers, as they often need to present complex technical information to a range of audiences, including project managers, stakeholders, clients, and fellow engineers. Understanding how to simplify technical jargon, engage the audience, and present data clearly is key to communicating effectively.

Key Techniques for Effective Presentations:

Know Your Audience: Before you begin preparing your presentation, consider the audience's level of technical understanding. Avoid overloading non-technical audiences with jargon. Simplify complex concepts and use analogies or examples when needed.

Structure Your Presentation: Start with an introduction that outlines the key points you will cover. Use a logical flow throughout the presentation, moving from problem to solution, or from concept to implementation.

Organize your slides into clear sections, such as Introduction, Challenges, Solution/Approach, and Next Steps.

Engage Your Audience with Storytelling: Use real-world examples or stories to make your presentation more relatable. For example, when presenting a new feature of a software product, explain how it solves a common problem.

Start with a scenario or a problem statement, and build your narrative around how your project or solution addresses the issue.

Keep Slides Simple and Visual: Avoid cluttering slides with too much text. Use bullet points, short phrases, and relevant visuals like diagrams, graphs, or screenshots.

Focus on one key point per slide and ensure that your visuals complement your message. Software engineers can use code snippets sparingly, only when necessary to explain functionality.

Practice Delivery Techniques: Maintain eye contact with your audience to engage them. Practice speaking clearly and at a moderate pace, avoiding the temptation to read directly from the slides.

Use hand gestures to emphasize key points, but avoid overusing them, which can distract the audience.

Explain Technical Concepts with Clarity: Break down complex ideas into simpler concepts. When discussing algorithms or code, for example, explain how each step contributes to the final outcome.

Use analogies to bridge the gap between technical and non-technical audience members. For example, compare a coding process to assembling a machine, explaining how each part (code) contributes to the whole (working software).

Anticipate Questions and Practice Q&A: Be prepared for technical questions from engineers and high-level, conceptual questions from project managers or clients.

Keep a few extra slides ready to address possible questions or clarifications, such as “Next Steps” or a slide that further explains an algorithm you discussed.

Use Presentation Software Tools Effectively: Master the tools you’re using, whether it's PowerPoint, Google Slides, or Keynote. Know how to navigate between slides seamlessly and use features like transitions, animations, or slide notes when appropriate.

Learn to annotate slides during the presentation to highlight key points or write out brief clarifications in real-time. 

Vocabulary

4. Match the highlighted words in the text with their definitions.

Word/Phrase	Definition
1. technical jargon	a. actively involving your audience in the presentation through interaction or attention-grabbing techniques
2. slide deck	b. tools like charts, graphs, or images that help illustrate your points during a presentation
3. eye contact	c. the act of delivering content in a narrative format to make the presentation more engaging
4. storytelling	d. maintaining visual connection with your audience to engage them while presenting
5. visual aids	e. a collection of presentation slides used to deliver information during a talk
6. engagement	f. specialized language used in specific professions that may be difficult for outsiders to understand

5. Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3https://quizlet.com/949499946/match?funnelUUID=e43774dd-50f4-4f2c-9db8-dbca044678ca>



6. Complete the sentences using the appropriate words from the list:

| storytelling, technical jargon, eye contact, slide deck, engagement, visuals, Q&A, analogies |

1. When explaining a complex coding process, use _____ to help your audience understand by comparing it to something more familiar.
2. To keep your audience's attention, maintain _____ while speaking.
3. Use _____ like graphs or diagrams to support your main points on each slide.
4. After presenting, be prepared for _____ by answering audience questions.
5. A well-organized _____ will help you present information clearly and logically.
6. Avoid overwhelming your audience with too much _____, especially if they are not technical experts.

7. Presentation Planning Activity:

*Choose one of the following topics and plan a **3-minute** presentation using the techniques discussed:*

Topic 1: Presenting a new feature in a mobile app you developed

Topic 2: Explaining how your team used Agile methodology in a recent project

Topic 3: Introducing a new API to your software engineering team

Steps:

- Identify your audience (Is it technical or non-technical?)
- Plan the structure: Introduction, Key Points, Conclusion.

- Choose the visuals: What graphs, charts, or diagrams will you use to support your points?
- Practice delivery: Focus on body language, eye contact, and clear explanations.

8. Mini-Presentation Roleplay:

- Each student presents a **3-minute** mini-presentation based on their chosen topic from Ex.7.
- Classmates provide feedback on:
 - Clarity of message
 - Use of visuals
 - Engagement with the audience
 - Simplicity and explanation of technical concepts

9. Answer the questions:

1. Why is it important for software engineers to tailor presentations for different audiences?
2. How can **visual aids** enhance a technical presentation?
3. What role does **storytelling** play in making a presentation more engaging?
4. Why should you practice anticipating questions from the audience?

Best Practices for Presentations Recap:

- **Tailor the Presentation to Your Audience:** Simplify technical information for non-technical audiences. Avoid jargon and use **analogies** where possible.
- **Tell a Story:** Use **storytelling** to create a narrative arc in your presentation. This keeps the audience engaged and provides a clear context for technical details.
- **Use Visuals Effectively:** Include **visual aids** such as diagrams or charts, but don't clutter slides with too much text. Let your visuals speak for themselves and support your key points.
- **Engage Your Audience:** Keep the audience's attention through eye contact, clear body language, and interactive elements like questions or discussion points.

Be Clear and Concise: Practice delivering your key points in a way that is easy to understand. Break down complex concepts into simpler terms without oversimplifying

Grammar for Revision Past simple or present perfect?

We use **Past Simple** to talk about **finished actions at a specific time in the past**.

e.g. *I **presented** my project **yesterday**.*

We use **Present Perfect** to talk about **experiences or achievements without saying when or recent actions connected to now**.

e.g. *I've **given** several tech presentations **this semester**.*

e.g. *I've **just finished** preparing my slides.*

- ◆ **Past Simple** = When? (specific time)
- ◆ **Present Perfect** = Experience / Result (no time mentioned)

10. Choose the correct form (Past Simple or Present Perfect)

1. Our team _____ (finish) the slides for the software demo last night.
2. I _____ (give) three presentations about user interface design so far.
3. The project manager _____ (ask) me to prepare extra slides for the Q&A yesterday.
4. We _____ (already / test) the new presentation tool in class.
5. I _____ (use) too much text on my slides last semester, but now I focus on visuals.
6. The presenter _____ (just / explain) how to simplify technical terms.
7. We _____ (start) our talk with a real-life example — it worked really well!
8. I _____ (never / present) to a non-technical audience before.

🔍 Extra Online Practice

Vocabulary:

[Quizlet – Unit 7.1 Flashcards & Learn](#)

Grammar:

Past simple or present perfect?

[Exercises 1–3 on Test-English.com](#)

☒ Recommended activity before starting Lesson 2:

Play Quizlet Live (**INDIVIDUAL MODE, 3 TIMES**) using vocabulary from Unit 7.1:

🔗 <https://quizlet.com/ua/932616895/unit-31-flash-cards/?i=j03j6&x=ljqt>

<https://quizlet.com/ua/948678035/unit-62-flash-cards/?i=j03j6&x=ljqt>

<https://quizlet.com/ua/949499946/unit-71-flash-cards/?i=j03j6&x=ljqt>

Lesson 2

How to describe and interpret visuals like graphs, charts, and diagrams

Lead-in 1.  Discuss in pairs or small groups:

1. What types of visuals (graphs, charts, diagrams) have you used or seen in presentations before?
2. Why are visuals important in explaining complex information?
3. What are some common challenges in explaining graphs or charts to an audience?

Reading: Read the text. Decide if the statements below are TRUE or FALSE.

7.2.

Describing and Interpreting Visuals: Graphs, Charts, and Diagrams

Presenting data through visuals is a common practice in software engineering. Visual aids like graphs, charts, and diagrams make it easier to explain complex information clearly and quickly. However, effectively interpreting and describing these visuals during presentations is a key skill that software engineers need to master.

Types of Visuals:

Bar Charts: Bar charts use rectangular bars to represent data. Each bar's length or height is proportional to the value it represents, making it easy to compare different data sets.

Example: "In this bar chart, you can see that the sales in Q2 were significantly higher than in Q1, with a difference of 20%. The bars represent the quarterly sales figures for the last two years."


Line Graphs: Line graphs are used to show trends over time. Each data point is plotted on the graph and connected by lines to show the progression.

Example: “The line graph shows a steady increase in user sign-ups over the past six months, with a notable spike in March, which correlates with the release of our new feature.”

Pie Charts: Pie charts display data as proportional slices of a circle, showing how different parts make up the whole.

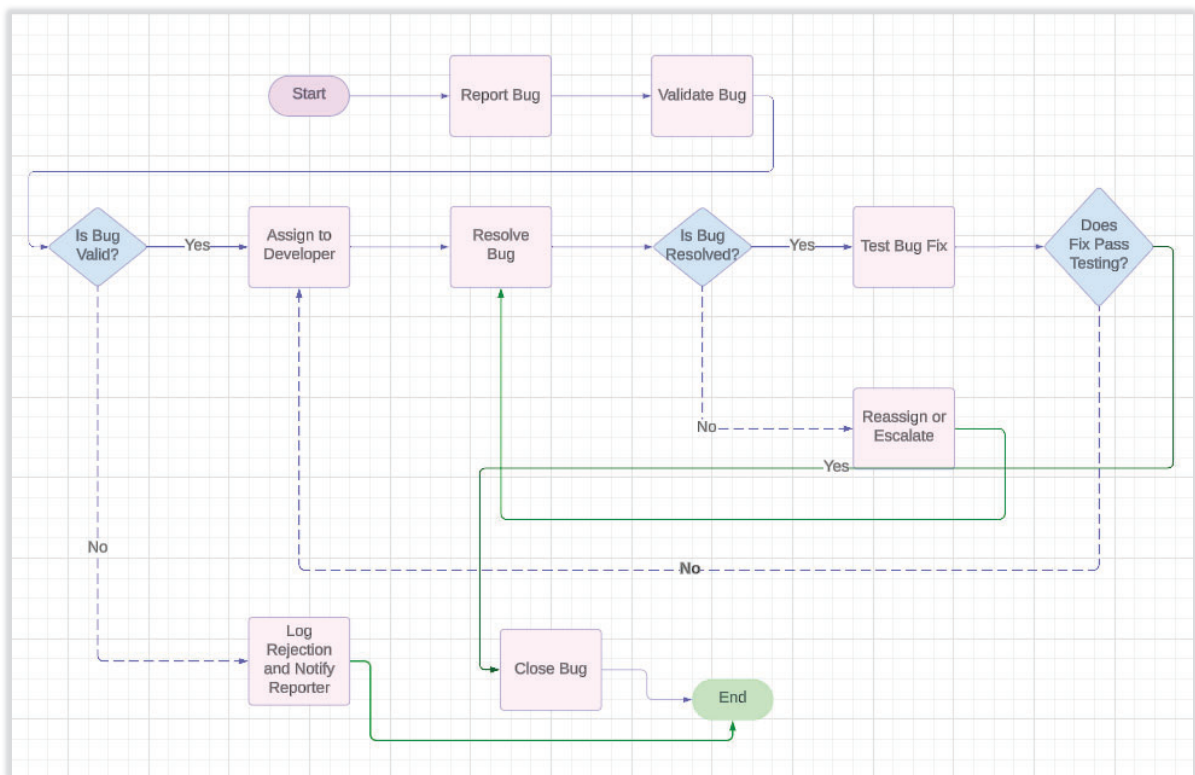
Example: “This pie chart represents the distribution of resources across departments. As you can see, 50% of the budget is allocated to R&D, while only 10% goes to marketing.”

Diagrams (Flowcharts):

Flowcharts show the steps in a process or system. They use shapes like rectangles (for actions) and diamonds (for decisions) to depict workflows or algorithms. 

TRUE or FALSE Statements

1. Bar charts are mainly used to show changes over time.
2. Line graphs connect data points to display trends.
3. Pie charts show how different parts make up a whole.
4. Flowcharts are used to compare numerical data between categories.
5. Each bar's length or height in a bar chart represents a specific value.
6. Visuals help explain complex information in a clear and quick way.
7. The example of the line graph shows a decrease in user sign-ups over time. →
8. Flowcharts can include shapes like rectangles and diamonds.
9. Pie charts are used to show trends over several months.
10. Interpreting and describing visuals is an important presentation skill for software engineers.



Example: “This flowchart illustrates the bug reporting process. As you can see, once the bug is reported, it is first validated, then assigned to a developer, and if resolved, it goes through testing before being closed.”

Vocabulary

4. Match the highlighted words in the text with their definitions.

Word/Phrase	Definition
1. Axis	a. The horizontal (x-axis) or vertical (y-axis) line on a graph showing data points
2. Legend	b. A key that explains what the symbols, colours, or patterns represent in a chart or graph
3. Bar Chart	c. A type of graph that uses rectangular bars to represent data comparisons
4. Line Graph	d. A graph that uses lines to show changes over time
5. Pie Chart	e. A circular chart divided into slices representing proportions
6. Flowchart	f. A diagram that shows a process or workflow
7. Data Point	g. A specific value or measurement plotted on a graph

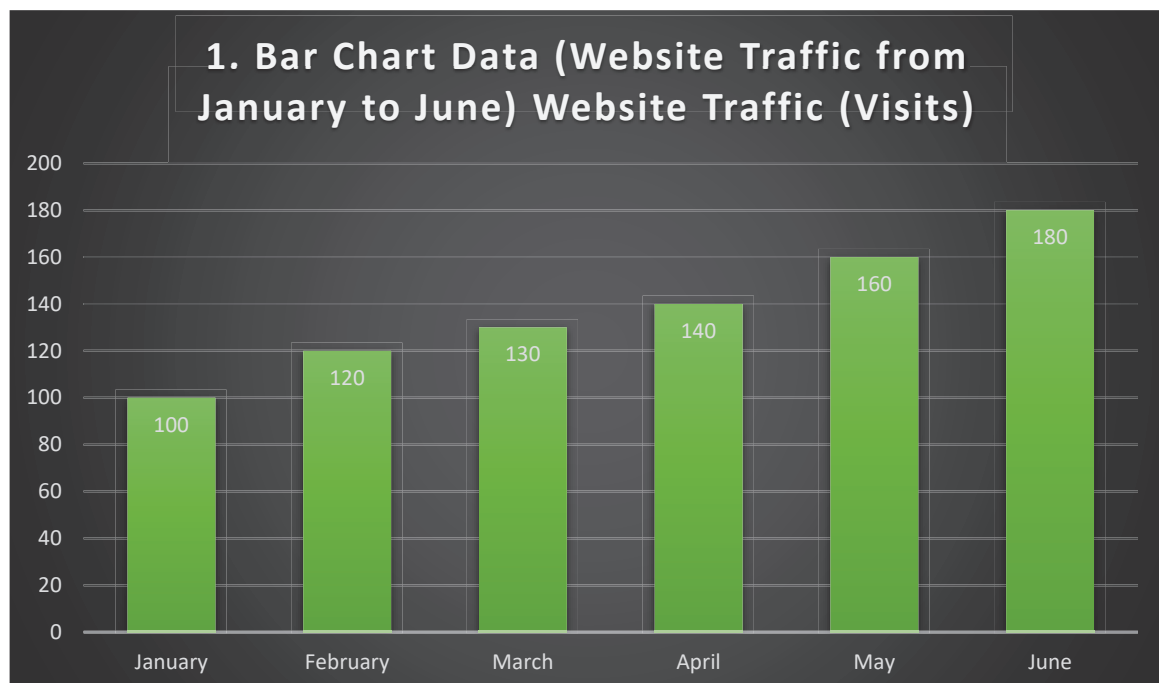
5. 🗑️ Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3><https://quizlet.com/949500322/match?funnelUUID=0ba-d9b94-9928-49d3-9dc7-4f1abc4499de>



6. Practice Describing Visuals: Look at the following visual representations and describe them using the key vocabulary:

Bar Chart:



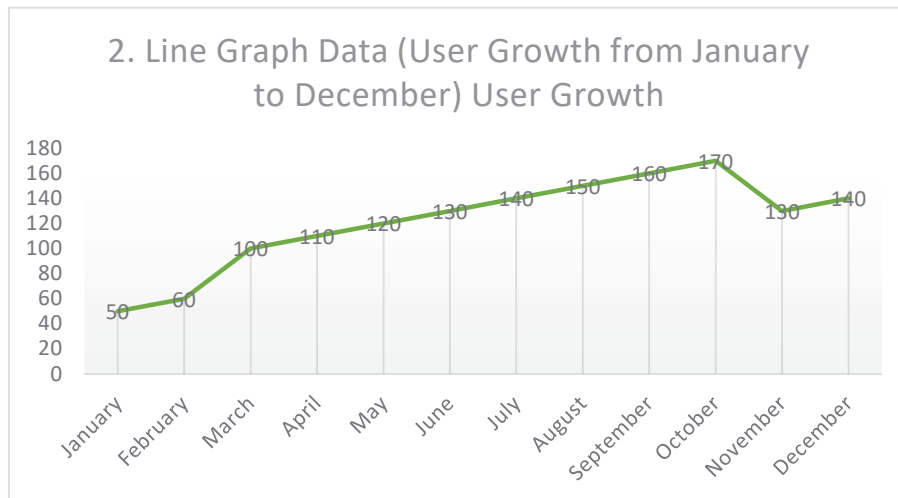
Bar chart showing website traffic from January to June, with January at 100k visits and June at 180k.

Complete the sentences using the appropriate words from the list:

| growth | traffic | steadily | reaching |

“This bar chart shows the _____ in website traffic from January to June. As you can see, _____ increased _____ each month, starting at 100,000 visits in January and _____ 180,000 visits by June.”

Line Graph:



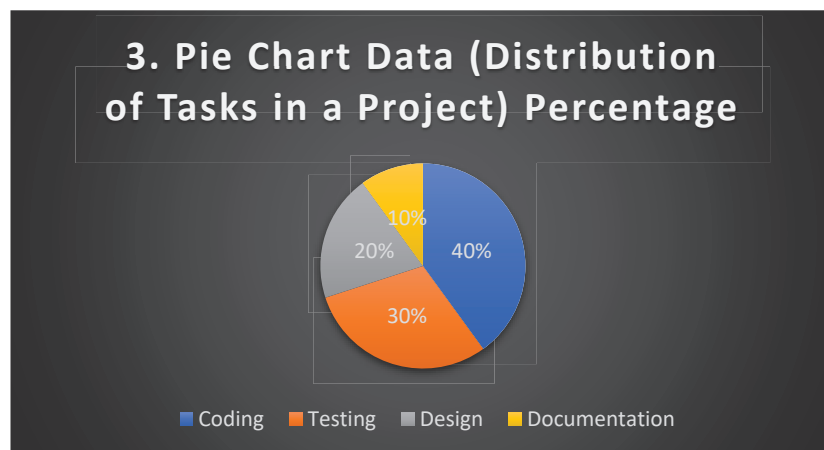
A line graph depicting user growth from January to December, with a steep rise in March and a dip in November.

Complete the sentences using the appropriate words from the list:

| rise | November | illustrates | slight | year |

“The line graph _____ our user growth over the _____. In March, there’s a significant _____, likely due to our marketing campaign. However, we see a _____ dip in _____, which may require further investigation.”

Pie Chart:



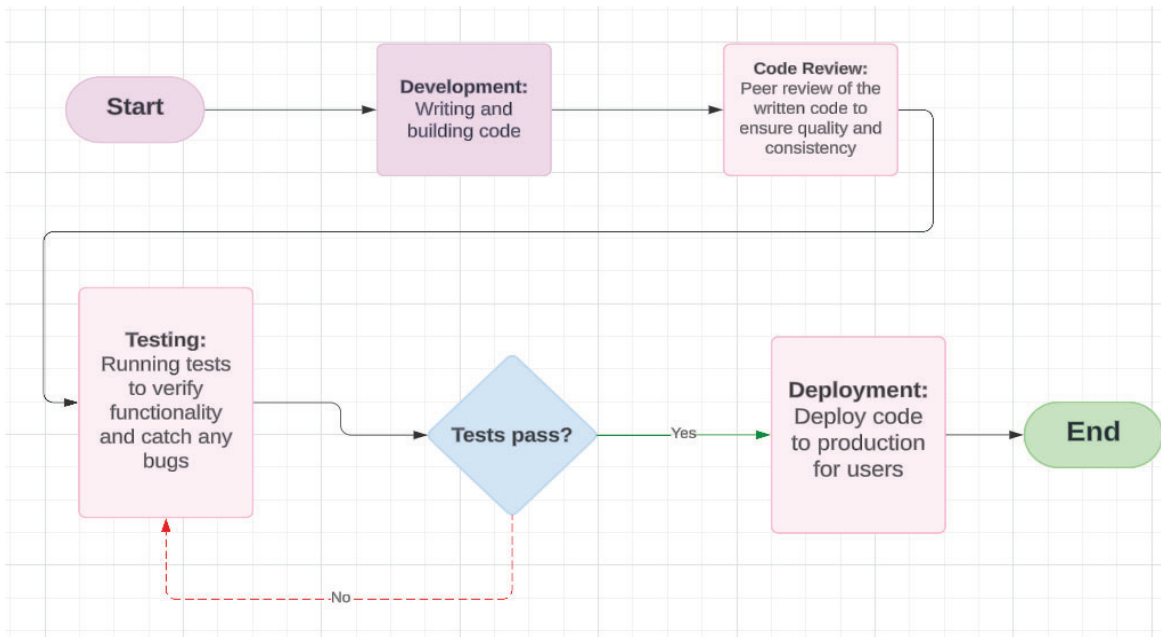
A pie chart showing the distribution of tasks in a project: 40% coding, 30% testing, 20% design, and 10% documentation.

Complete the sentences using the appropriate words from the list:

| coding | pie | total | documentation |

“This _____ chart shows the distribution of tasks in our project. As you can see, _____ takes up the largest portion, accounting for 40% of the _____ work, while _____ is the smallest at 10%.”

Flowchart:



A flowchart depicting the process of pushing code from development to production with stages: development, code review, testing, and deployment.

Complete the sentences using the appropriate words from the list:

| deployed | review stage | tests | code deployment |

“This flowchart outlines our _____ process. After development, the code goes through a _____, then it is tested, and once all _____ pass, it’s finally _____ to production.”

7. Activity: Pair Work

Work in pairs and take turns describing different visuals provided by your partner. Choose from bar charts, line graphs, pie charts, or flowcharts. Use the key vocabulary and tips provided in the lesson.

Example:

Partner A: Shows a line graph and describes it:

“The graph shows how product sales increased from January to July, with a significant peak in May due to the launch of a new version.”

Partner B: Adds details or corrections if needed.

8. Presentation Activity:

Each student selects a chart, graph, or diagram and gives a short (1-2 minute) presentation, explaining the data to the class. Focus on:

- Clearly describing the visual
- Interpreting what the data means
- Engaging the audience with questions like “What do you think caused the spike in March?”

9. Describing Visuals in Context:

1. Clarity and Precision:

When describing visuals, be clear and concise. Highlight the most important data points, trends, or comparisons. Avoid overwhelming your audience with too much detail.

2. Relating Data to Audience:

Make the data relatable to your audience by explaining why it matters. For instance, if you’re presenting to a marketing team, focus on how user growth impacts their strategy.

3. Using Visual Aids to Support Arguments:

Visuals should complement your message, not distract from it. Ensure that each chart or graph you present is easy to understand and directly supports the key points you’re making.

10. Complete the sentences using the correct word:

| words: | flowchart, axis, legend, data points, pie chart, bar chart |

1. In this _____, you can see that coding takes up the largest portion of our time at 40%.
2. Each bar in this _____ represents a different department’s budget for the quarter.
3. The x-_____ shows the months of the year, while the y-_____ represents the number of customers.
4. Each circle in this _____ represents a step in the decision-making process.
5. The _____ at the top of the chart explains what each color stands for.
6. In this graph, the _____ are plotted to show changes in sales over the year.

11. Best Practices for Describing Visuals Recap:

1. Be Clear and Concise:

- Focus on the key points in the visual, and avoid going into unnecessary detail.

2. Relate the Data to the Audience:

- Explain why the data is important to your audience and what they can learn from it.

3. Interpret the Visuals:

- Don't just describe what's on the chart or graph—interpret it. Highlight trends, changes, or anomalies that are significant to your presentation.

4. Use Appropriate Vocabulary:

- Familiarize yourself with the terms used to describe charts, graphs, and diagrams, such as “axis,” “legend,” and “data points.”

12. Answer the questions:

1. Why is it important to describe visuals clearly in a presentation?
2. How can visuals help make complex data easier to understand?
3. What are some common mistakes people make when explaining charts or graphs?
4. Why should you tailor your explanation of visuals to your audience?

Grammar for Revision Present perfect simple and present perfect continuous

We use Present Perfect Simple to talk about completed actions or results that have relevance now.

e.g. We've analyzed the data and created the final chart.
e.g. The number of users has increased by 20% since January.

We use Present Perfect Continuous to focus on the activity itself, its duration, or unfinished process.

e.g. We've been analyzing the data all morning.
e.g. Our team has been working on this infographic for two days.

- ◆ **Present Perfect Simple** → result / completed action
- ◆ **Present Perfect Continuous** → process / duration / still in progress

10. Choose the correct form (Past Simple or Present Perfect)

1. The development team _____ (collect) data for the performance chart since Monday.
2. Our designer _____ (create) a series of infographics to show project milestones.
3. The data visualization tool _____ (crash) twice this week.
4. We _____ (work) on improving the clarity of the bar chart.
5. I _____ (finish) the final version of the dashboard today.
6. The analytics team _____ (analyze) customer feedback for several days.
7. The number of users _____ (grow) steadily over the past few months.
8. We _____ (test) different color schemes to make the chart more readable.



Extra Online Practice

Vocabulary:

Quizlet – Unit 7.2 Flashcards & Learn


Grammar:

Present perfect simple and present perfect continuous
Exercises 1–3 on Test-English.com



- ☒ Recommended activity before starting Lesson 3:

Play Quizlet Live (**INDIVIDUAL MODE, 3 TIMES**) using vocabulary from Unit 7.2:

 <https://quizlet.com/ua/932616895/unit-31-flash-cards/?i=j03j6&x=1jqt>

<https://quizlet.com/ua/948678035/unit-62-flash-cards/?i=j03j6&x=1jqt>

<https://quizlet.com/ua/949500322/unit-72-flash-cards/?i=j03j6&x=1jqt>

Lesson 3

Using presentation software tools efficiently (e.g., PowerPoint, Keynote, Google Slides, Prezi, Canva)

Lead-in 1.  Discuss in pairs or small groups:

1. What presentation tools have you used before?
2. What do you like or dislike about these tools?
3. How do presentation tools help improve the quality of a presentation?

2. Decide if you agree (☑) or disagree (✗). Be ready to explain.

- a) Visuals are more important than the speaker's delivery.
- b) Simplicity is the key to a good presentation.
- c) Animation always makes slides more interesting.
- d) Using too much text can make slides boring.
- e) Presentation tools can replace real communication skills.

3. Read the text. While reading, check which of the statements from the previous exercise the author would agree or disagree with.

7.3.

Using Presentation Software Tools Efficiently

When creating presentations, it's important to choose the right tool based on your needs, audience, and the content you want to present. Common tools like **PowerPoint**, **Keynote**, **Google Slides**, **Prezi**, and **Canva** offer a range of features that can help make your presentations more engaging and professional.

Overview of Common Presentation Software Tools:
PowerPoint (Microsoft Office):

Key Features: Templates, transitions, animations, collaboration, presenter view, real-time co-authoring, exporting to video and PDF.

Best For: Traditional business presentations, especially when offline use is required.

TIP: Use the “Slide Master” to create consistent formatting across all slides.

Keynote (Apple):

Key Features: Beautiful templates, animations, seamless integration with Apple devices, collaboration, presenter view, export to video and PDF.

Best For: Users in the Apple ecosystem looking for polished, creative presentations.

TIP: Use Keynote’s advanced animations to create engaging, dynamic visuals.

Google Slides:

Key Features: Cloud-based, real-time collaboration, easy sharing, integration with other Google Workspace tools, templates, export options.

Best For: Collaborative presentations where team members can work simultaneously, especially for remote teams.

TIP: Use the “Explore” tool to quickly add images and formatting suggestions from the web.

Prezi:

Key Features: Zooming user interface, dynamic transitions, nonlinear navigation, cloud-based, templates.


Best For: Creative presentations with dynamic, non-linear storytelling.

TIP: Prezi is ideal for telling a story with zooming transitions rather than following a traditional slide-by-slide format.

Canva:

Key Features: Graphic design features, drag-and-drop interface, extensive library of images, icons, and templates, collaboration.

Best For: Visually appealing, graphic-heavy presentations.

TIP: Use Canva's pre-designed layouts to create visually stunning presentations quickly. 

Vocabulary

4. Match the highlighted words in the text with their definitions.

Word/Phrase	Definition
1. template	a) work together with others on the same document or presentation in real time
2. transition	b) save a file in a different format, such as PDF or video
3. slide deck	c) an animation that occurs when moving from one slide to the next
4. animation	d) a visual effect applied to text or objects to make them move, appear, or disappear
5. collaborate	e) a pre-designed slide layout that you can use to create presentations quickly and consistently
6. presenter view	f) a feature that allows the presenter to see notes and upcoming slides while the audience sees only the slides
7. export	g) a collection of slides used in a presentation

5. Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3https://quizlet.com/949531916/match?funnelUUID=41330843-22d7-4772-a2f0-3391b1b2a5c9>



6. Practice Using Presentation Tools:

Task: Choose a simple topic related to software engineering (e.g., “The Basics of Agile Methodology”) and create a short slide deck (5 slides) using one of the following tools: PowerPoint, Google Slides, Prezi, or Canva.

Steps:

1. Select a template to maintain consistent design across all slides.
2. Add bullet points, images, and charts to communicate your message clearly.

3. Use transitions and animations sparingly to avoid overwhelming your audience.
4. Ensure that each slide has a clear focus and isn't overloaded with information.
5. Use "presenter notes" to add cues or talking points for your presentation.

5. Presentation Software Best Practices:

- › **Keep it Simple:** Use clean designs and avoid cluttering slides with too much text or unnecessary visuals. Each slide should focus on a single idea.
- › **Use Visuals:** Visual elements like images, icons, and graphs make your presentation more engaging and easier to understand. Make sure visuals are relevant and add value.
- › **Engage with Animations Sparingly:** While animations and transitions can add interest, overusing them can be distracting. Use them only when they serve a purpose, such as highlighting key points.
- › **Know Your Audience:** Tailor the content and design of your presentation to your audience's needs. For example, business professionals may prefer a more formal, minimalist design, while a more creative audience may appreciate visually dynamic content.
- › **Be Consistent:** Use consistent fonts, colors, and layout throughout your presentation to maintain a professional look. This can be achieved by using pre-designed templates or creating your own style guide.
- › **Use Presenter View:** Tools like PowerPoint, Google Slides, and Keynote offer a "presenter view" feature, allowing you to see your notes and upcoming slides while the audience only sees the current slide. This helps keep you on track during your presentation.

6. Hands-On Activity: Collaboration in Google Slides

In this activity, you will work with a partner to create a collaborative presentation using Google Slides. Choose a topic from your field (e.g., "The Benefits of CI/CD

in DevOps”). Divide the work and simultaneously create a slide deck with each person working on different slides.

- **Step 1:** One person creates the presentation and shares the link with editing access.
- **Step 2:** Both participants work together in real-time, adding content, visuals, and notes.
- **Step 3:** Use the comments feature to provide feedback or ask questions about the content your partner has added.
- **Step 4:** Present your final slide deck to the class.

7. Activity: *Analysing a Slide Deck*

Choose a pre-designed slide deck template from Canva or Google Slides. Analyse the following:

- Is the template easy to read and visually appealing?
- How are animations and transitions used?
- Are the slides consistent in design and layout?
- What could be improved to make the presentation more effective?

8. Best Practices for Presentation Tools Recap:

Choose the Right Tool:

Consider the content and audience when selecting a presentation tool. For collaborative projects, Google Slides might be best. For dynamic storytelling, Prezi can be more engaging.

Keep It Simple and Professional:

Avoid overloading your slides with text. Use visuals and bullet points to keep your message clear and concise.

Engage Your Audience:

Use animations and transitions thoughtfully to emphasize key points, but don't overdo it.

Collaborate Effectively:

For group projects, use cloud-based tools like Google Slides or Canva to allow for real-time collaboration. Use commenting features to provide feedback.

9. Complete the sentences using the correct word:

| animation, slide deck, export, collaborate, template, presenter view |

1. I used the _____ feature in PowerPoint to see my notes while presenting to the audience.
2. This _____ is too long; try to reduce the number of slides by combining some of them.
3. Canva has a great _____ for creating modern presentations with graphic elements.
4. Prezi's zooming _____ made the presentation feel dynamic and engaging.
5. We will _____ on this presentation by sharing a Google Slides document.
6. You can _____ the final presentation as a PDF to make it easier to share with the team.

10. Answer the questions:

1. What are the benefits of using cloud-based presentation tools like Google Slides or Canva for collaboration?
2. Why is it important to keep presentation designs simple and consistent?
3. How can presenter view help you during a presentation?
4. Why should you tailor the style of your presentation to your audience?

Grammar for Revision Past simple, past continuous, past perfect

We use **Past Simple** to talk about **finished actions at a specific time in the past**.

e.g. *I **presented** my project **yesterday**.*

We use **Present Perfect** to talk about **experiences or achievements without saying when or recent actions connected to now**.

e.g. *I've **given** several tech presentations **this semester**.*

e.g. *I've **just finished** preparing my slides.*

- ◆ **Past Simple** = When? (specific time)
- ◆ **Present Perfect** = Experience / Result (no time mentioned)

10. Choose the correct form (Past Simple or Present Perfect)

1. Our team _____ (finish) the slides for the software demo last night.
2. I _____ (give) three presentations about user interface design so far.
3. The project manager _____ (ask) me to prepare extra slides for the Q&A yesterday.
4. We _____ (already / test) the new presentation tool in class.
5. I _____ (use) too much text on my slides last semester, but now I focus on visuals.
6. The presenter _____ (just / explain) how to simplify technical terms.
7. We _____ (start) our talk with a real-life example — it worked really well!
8. I _____ (never / present) to a non-technical audience before.

Extra Online Practice

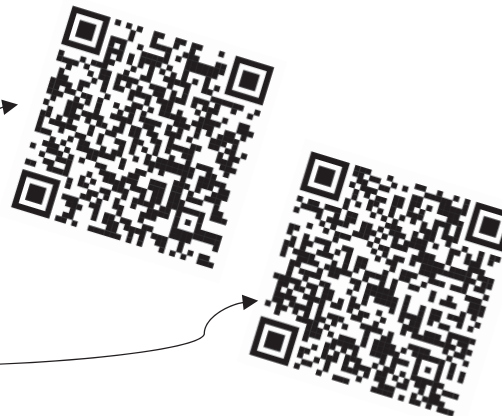
Vocabulary:

Quizlet – Unit 7.3 Flashcards & Learn

Grammar:

Past simple, past continuous, past perfect

Exercises 1–3 on Test-English.com



☒ **Recommended activity before starting Unit 7 Revision:**

Play **Quizlet Live** (INDIVIDUAL MODE, 3 TIMES) using vocabulary from Unit 7.3:



<https://quizlet.com/ua/932616895/unit-31-flash-cards/?i=j03j6&x=1jqt>

[https://quizlet.com/ua/948678035/unit-62-flash-](https://quizlet.com/ua/948678035/unit-62-flash-cards/?i=j03j6&x=1jqt)

[cards/?i=j03j6&x=1jqt](https://quizlet.com/ua/949531916/unit-73-flash-cards/?i=j03j6&x=1jqt)[https://quizlet.com/ua/949531916/unit-73-flash-](https://quizlet.com/ua/949531916/unit-73-flash-cards/?i=j03j6&x=1jqt)

[cards/?i=j03j6&x=1jqt](https://quizlet.com/ua/949531916/unit-73-flash-cards/?i=j03j6&x=1jqt)

Vocabulary Revision

Unit 7

 1. WORK IN PAIRS. TAKE TURNS PLAYING THE ROLES OF **DESCRIBER** AND **GUESSER** USING VOCABULARY FROM UNIT 7.

◇ Detailed instructions can be found on page 37

Term	Definition
template	a pre-designed slide layout that you can use to create presentations quickly and consistently
transition	an animation that occurs when moving from one slide to the next
slide Deck	a collection of slides used in a presentation
animation	a visual effect applied to text or objects to make them move, appear, or disappear
collaborate	work together with others on the same document or presentation in real time
presenter View	a feature that allows the presenter to see notes and upcoming slides while the audience sees only the slides
export	save a file in a different format, such as PDF or video
Axis	the horizontal (x-axis) or vertical (y-axis) line on a graph showing data points
Legend	a key that explains what the symbols, colors, or patterns represent in a chart or graph
Bar Chart	a type of graph that uses rectangular bars to represent data comparisons
Line Graph	a graph that uses lines to show changes over time
Pie Chart	a circular chart divided into slices representing proportions
Flowchart	a diagram that shows a process or workflow
Data Point	a specific value or measurement plotted on a graph
Technical Jargon	specialized language used in specific professions that may be difficult for outsiders to understand
Slide Deck	a collection of presentation slides used to deliver information during a talk
Eye Contact	maintaining visual connection with your audience to engage them while presenting
Storytelling	the act of delivering content in a narrative format to make the presentation more engaging

Visual Aids	tools like charts, graphs, or images that help illustrate your points during a presentation
Engagement	actively involving your audience in the presentation through interaction or attention-grabbing techniques
channels	dedicated spaces within a chat app where team members can discuss specific topics or projects

3. THREE-SENTENCE CHALLENGE. Each participant **selects three words** from the shared word list (*pick the ones you find most challenging*) and forms **three sentences** using these words. Work in pairs. Read your sentences to your partner, replacing the chosen word with "gap" while reading. Your partner needs to guess the missing word.

3. PLAY THE QUIZLET LIVE GAME (TEAM MODE) to review the vocabulary learned in **UNIT 6**

<https://quizlet.com/ua/949497514/unit-6-61-62-63-flash-cards/?i=j03j6&x=1jqt> <https://quizlet.com/ua/949533095/unit-7-73-72-71-flash-cards/?i=j03j6&x=1jqt>



4. TAKE the UNIT 7 VOCABULARY QUIZLET TEST to check your understanding of key terms from this UNIT. <https://quizlet.com/948371787/test?answerTermSides=6&promptTermSides=6&questionCount=30&questionTypes=15&showImages=true>
<https://quizlet.com/949533095/test?answerTermSides=2&promptTermSides=4&questionCount=20&questionTypes=15&showImages=true>



5. 🎲 VOCABULARY GAME: ALIAS

Play a fun team-based vocabulary challenge!

♦ See full game instructions on page 37.

Get ready to explain, guess, and compete using key terms from Unit 7!

For Printable Vocabulary Materials for the Game “Alias,” go to page 236.



This unit introduces learners to the principles of software quality assurance. Students explore key quality attributes, the stages and types of software testing, and methods for testing web applications to ensure performance, usability, and security.

Unit overview

8.1 *Software quality attributes*

Lesson outcome: Learners can identify and describe key software quality attributes and explain their importance in software development.

Skills practised

Reading: understanding descriptions of quality attributes

Speaking: discussing and prioritizing software quality factors

Vocabulary: terms for software

Grammar: present perfect simple and continuous

8.2 **Software Testing**

Lesson outcome: Learners can describe types and phases of software testing and explain their role in ensuring software reliability.

Skills practised:

Reading: identifying stages and types of software testing

Speaking: explaining testing processes and reporting bugs

Vocabulary: testing terms

Grammar: narrative tenses

8.3 *Web Application Testing*

Lesson outcome: Learners can explain and plan web application testing focusing on performance, security, and cross-browser compatibility.

Skills practised:

Reading: understanding types of web application testing

Speaking: presenting and discussing web testing plans and issues

Vocabulary: web testing terminology

Grammar: usually / used to / be used to / get used to

Lesson 1

Software quality attributes

Lead-in 1.  Discuss in pairs or small groups:

1. What does “software quality” mean to you?
2. Why do you think software quality is important in software development?
3. Can you name any attributes that define software quality?

2. Decide if you agree (☒) or disagree (✗). Be ready to explain.

- a) High-quality software is always expensive to develop.
- b) The main goal of software testing is to make software perfect.
- c) User satisfaction is the best measure of software quality.
- d) Performance is more important than security.
- e) Software quality depends only on developers, not users.

Reading:

3. Skim the text and find answers to the following questions:

1. What are software quality attributes?
2. Why are quality attributes important in software engineering?
3. What does the functionality attribute measure?
4. How is software reliability described in the text?
5. Why is usability important for end users?
6. Which attribute focuses on protecting user data and preventing unauthorized access?

8.1.

Software Quality Attributes

In software engineering, ensuring high-quality software is critical to success. Quality attributes are characteristics that determine the performance, usability, security, and overall effectiveness of a system. These attributes help software meet the needs of users and operate efficiently in various environments.

Key Software Quality Attributes:

Functionality: The core requirement of any software is to function as expected. It should meet the users' needs by offering the correct features, services, and behaviors. Functionality is measured by the ability of the software to perform specific tasks reliably, accurately, and consistently.

Reliability: Software reliability refers to the system's ability to function under specified conditions for a defined period of time. Reliable software doesn't fail during normal usage, making it essential for mission-critical applications, such as healthcare or banking systems.


Usability: Usability determines how easily end users can learn to use the software, its interface, and the overall user experience. Well-designed, user-friendly software encourages efficient work and reduces the likelihood of user error.

Efficiency: Efficiency involves how well software uses system resources, including memory, CPU, and network bandwidth. Highly efficient software minimizes resource consumption, allowing it to perform well even under heavy workloads.

Maintainability: Over time, software needs updates, fixes, and improvements. Maintainability is the ease with which developers can modify or extend the software. Code should be well-organized, documented, and structured to allow for quick changes.

Portability: With the growing diversity of devices and operating systems, portability is crucial. Portable software can operate on different hardware and software environments with minimal reconfiguration.

Security: Security ensures that the software can resist attacks, protect user data, and maintain integrity. With cyber threats on the rise, secure software is vital for preventing data breaches and unauthorized access.

Performance: Performance refers to the system's speed and responsiveness. Software should handle tasks quickly, even as the number of users or amount of data grows. Good performance is critical for real-time applications and large-scale systems. 

Vocabulary

4. Match the highlighted words in the text with their definitions.

Word/Phrase	Definition
1. functionality	a) the software's ability to work across different environments and platforms
2. reliability	b) the software's ability to use resources optimally
3. usability	c) the range of operations that can be performed by a system or software
4. efficiency	d) the ability to protect the software from unauthorized access or harm
5. maintainability	e) the ability of software to perform its functions under stated conditions
6. portability	f) how quickly and effectively the software responds and performs tasks
7. security	g) how easy and intuitive the software is for users
8. performance	h) how easily software can be modified or updated

5. Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3https://quizlet.com/949534167/match?funnelUUID=d3882b44-32d4-4dc3-8c0f-65b2e9e7da14>



4. Group Discussion Activity:

Scenario:

Imagine you are working on developing a banking application. Which of the following quality attributes would be most important in this context, and why? Discuss the relevance of each attribute:

- Functionality
- Reliability
- Usability
- Security
- Performance

Task:

In groups, discuss how these attributes affect a banking app and prioritize them in order of importance. Be prepared to present your reasoning to the class.

5. Exercise: Identify Quality Attributes in Real Software

Task:

Choose a piece of software you are familiar with (e.g., a web browser, mobile app, or desktop software). For each of the quality attributes below, explain how the software meets or fails to meet the attribute:

1. Functionality
2. Usability
3. Reliability
4. Security
5. Performance

Example:

- **Software:** Google Chrome
- **Functionality:** Offers a wide range of features, including tab management and bookmark syncing.
- **Usability:** Simple interface, easy to navigate.
- **Reliability:** Consistently works well with few crashes.
- **Security:** Regular security updates, incognito mode for privacy.
- **Performance:** Generally fast but can slow down when too many tabs are open.

6. Key Software Quality Attribute: Usability

Mini Case Study:

The mobile banking app "BankPro" received complaints from users about its complex navigation and slow performance. As a result, the company experienced a high number of uninstalls.

Discussion Questions:

1. How does poor usability affect the overall quality of the app?
2. What improvements could be made to enhance the usability of the app?

7. Best Practices for Ensuring Software Quality:

1. Conduct Regular Testing:

Continuous testing throughout the software development lifecycle ensures early identification of issues and improves quality.

2. Use Clear Documentation:

Well-organized and detailed documentation aids in maintaining and improving the software efficiently.

3. Ensure Good User Experience (UX):

Focus on intuitive design and easy navigation to enhance usability.

4. Monitor Performance Metrics:

Regularly assess the software's performance under different conditions, and optimize it for speed and efficiency.

5. Implement Strong Security Measures:

Protect user data and maintain software integrity by employing the latest security protocols.

8. Practice Exercise: Improving Software Quality

Task:

Imagine you are part of a team working on a new video streaming platform. After releasing the beta version, you receive feedback about performance issues (long buffering times), security concerns (data leaks), and low usability (confusing interface). Based on what you've learned about quality attributes, outline an action plan to address these issues.

Include the following:

- How to improve **performance**
- Steps to enhance **security**
- Changes to boost **usability**

9. Complete the sentences using the correct word:

| Words: | reliability, maintainability, security, portability, usability |

1. The system's _____ is essential, as it will be used across various operating systems.
2. We need to focus on _____ to ensure that users can navigate the application easily.
3. _____ is crucial for a healthcare application, as data privacy must be protected at all times.
4. One advantage of the software is its high _____; it can run without crashing for long periods.
5. The code's _____ is excellent, making it easy to implement updates or fixes.

10. Answer the questions:

1. Why is it important for software to be portable across different platforms?
2. What is the difference between usability and functionality?
3. How can improving the maintainability of software save time and resources?
4. Why is security considered one of the most critical software quality attributes?

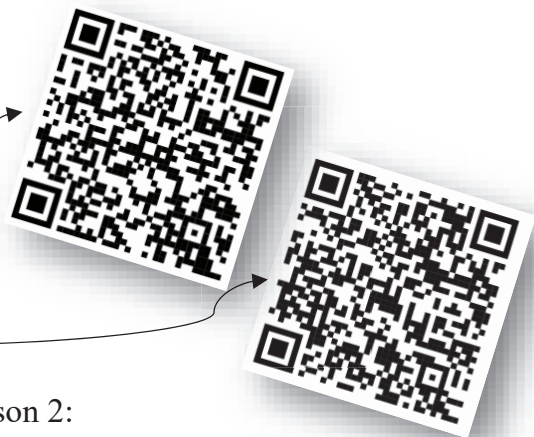
Extra Online Practice

Vocabulary:

Quizlet – Unit 8.1 Flashcards & Learn

Grammar:

Present perfect simple or continuous
Exercises 1–3 on Test-English.com



☒ Recommended activity before starting Lesson 2:

Play Quizlet Live (INDIVIDUAL MODE, 3 TIMES) using vocabulary from Unit 8.3:

 <https://quizlet.com/ua/949534167/unit-81-flash-cards/?i=j03j6&x=1jqt>

Lesson 2

Software Testing

Lead-in 1.  Discuss in pairs or small groups:

1. What do you think is the main purpose of software testing?
2. Have you ever encountered bugs or issues with software? What were they, and how did they affect your experience?
3. Why do you think testing is important before releasing software?

Reading:

2. Read the text. Decide if the statements below are **TRUE** or **FALSE**. (True/False)

8.2.

Software Testing

Once the subsystems developed by the various engineering teams are completed, they are integrated, and integration and system testing is carried out. System integration testing follows the system integration test plan, which was initially produced during the system design and allocation phase. However, changes that occurred during the subsystems' development phase may necessitate modifications to

1. System integration testing checks if all subsystems communicate correctly through their interfaces. (**True/ False**)
2. The system integration test plan never changes once it is created. (**True/ False**)
3. Specialized equipment may be used to test system performance in some cases. (**True/ False**)
4. Beta testing is done before the system is installed in its target environment. (**True/ False**)
5. Regression testing checks that new updates do not break existing features. (**True/ False**)
6. Manual testing is faster and more suitable for repetitive tasks than automated testing. (**True/ False**)
7. System testing ensures the whole system meets its requirements. (**True/ False**)
8. Acceptance testing is performed after the software is released to users. (**True/ False**)

the test plan. This phase of testing not only refines the test procedures but also ensures that the subsystems communicate properly through their interfaces.

Following system integration testing, system testing is performed based on the acceptance test plan. This step ensures that the system is capable of delivering all the capabilities outlined in the system requirements. Depending on the nature of the system, specialized equipment may be required for integration and system testing. For instance, telecommunication systems might use test equipment that can simulate millions of calls to assess the system's throughput, performance, and response times accurately.


Once system testing is completed, the system is shipped and installed in its target environment, where it undergoes user testing, also known as beta testing. During this phase, any defects identified by users are reported to the development team. These defects are addressed, and the system is retested. After the beta testing phase, the system transitions into the maintenance phase, where enhancements are made, and any remaining defects are resolved.

Software testing is a critical process that ensures the software meets its requirements and functions as intended. It helps identify and fix bugs, ensuring a smoother user experience and preventing costly issues after deployment. Testing also ensures that the software behaves well under different conditions and integrates smoothly with other systems.

Types of Software Testing

- **Unit Testing:** This type of testing focuses on individual components (units) of the software. Each unit is tested in isolation to ensure it functions correctly. Unit testing is often automated and is one of the first types of testing conducted during development.
- **Integration Testing:** After individual components are tested, integration testing checks how these components work together. It ensures that different parts of the system communicate and function properly as a whole.

- **Regression Testing:** When software is updated or modified, regression testing ensures that the existing features and functionalities still work correctly. It is vital to prevent new bugs from being introduced during updates.
- **System Testing:** This type of testing evaluates the complete, integrated system to ensure it meets the specified requirements. It covers all aspects of the software, including functionality, usability, security, and performance.
- **Acceptance Testing:** Conducted before release, acceptance testing verifies that the software meets the client's needs and is ready for production. This testing confirms that the system behaves as expected under real-world conditions.

Manual vs. Automated Testing: Manual testing involves human testers executing tests, while automated testing uses tools to run predefined test cases. Both approaches have their advantages: manual testing is flexible and can adapt to complex scenarios, while automated testing is faster and can handle repetitive tasks efficiently. 

Vocabulary

3. Match the highlighted words in the text with their definitions.

Word/Phrase	Description
1. unit testing	a) testing to confirm that the system is ready for production and meets user requirements.
2. integration testing	b) testing to check if different software components work together.
3. regression testing	c) testing individual components of the software.
4. system testing	d) verifying the entire system functions as expected.
5. word/phrase	e) definition
6. bug	f) the percentage of code tested by the test cases
7. test case	g) tests that are executed by human testers without the use of automated tools
8. unit testing	h) using tools to run tests automatically, without human intervention
9. integration testing	i) testing to confirm the system meets the requirements and is ready for release

10. regression testing	j) testing after changes have been made to ensure that existing functionality works
11. acceptance testing	k) testing the interaction between different software components
12. automated testing	l) testing individual components of software to ensure they function correctly
13. manual testing	m) a set of conditions under which a tester will determine whether an application works
14. test coverage	n) ensuring that the software development process follows established standards
15. quality assurance (QA)	o) a flaw or fault in a software program that causes it to operate incorrectly
16. acceptance testing	p) ensuring that changes or updates do not break existing features.

4.  Play the Quizlet Match game by following this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3><https://quizlet.com/949534167/match?funnelUUID=d3882b44-32d4-4dc3-8c0f-65b2e9e7da14>



5. Exercise: Identifying Bugs

Scenario: You are testing an e-commerce website, and you find the following issues:

1. The “Add to Cart” button doesn’t work on the product page.
2. The login page takes too long to load.
3. Users are logged out randomly while browsing.

Task: In groups, identify which types of testing would catch these bugs and explain why. Present your findings to the class.

6. Software Testing Phases

Mini Case Study: Imagine you are part of a team developing a new social media platform. The software is almost ready, but the company wants to ensure it's thoroughly tested before launch. Your team needs to plan the testing phases.

Phases of Testing:

1. **Unit Testing:** Ensures that individual features, such as posting, liking, and commenting, work correctly.
2. **Integration Testing:** Verifies that user interactions like sending messages or uploading photos work across different components.
3. **System Testing:** Assesses the entire platform's performance, including speed, usability, and security.
4. **Acceptance Testing:** Conducted with users to confirm that the platform behaves as expected.

Task: Outline the testing phases for this social media platform and describe how you would implement each phase. Which phase would you prioritize and why?

7. Software Testing Best Practices

1. **Start Testing Early:** Begin testing during the development process rather than waiting until the software is complete. Early testing helps catch issues sooner and prevents delays.
2. **Automate Where Possible:** Automating repetitive tests, such as unit tests and regression tests, saves time and reduces human error. Automation tools like Selenium, JUnit, and TestNG can be used.
3. **Perform Comprehensive Regression Testing:** Every time code is modified, regression testing ensures that new changes haven't affected existing functionality.
4. **Test Under Realistic Conditions:** Simulate real-world scenarios when testing, such as multiple users accessing the system at the same time, to

ensure the software performs well under actual conditions.

5. **Document Bugs Thoroughly:** Clear and detailed bug reports make it easier for developers to understand and fix the issue. Include steps to reproduce the bug, expected results, and screenshots if necessary.

8. Practice Exercise: Bug Reporting

Task:

You've found a bug while testing an app where the search function doesn't return results when searching for a specific category. Write a detailed bug report using the following template:

Bug Report Template:

- **Summary:** Briefly describe the issue.
- **Steps to Reproduce:** List the steps that lead to the bug.
- **Expected Results:** Describe what should happen when the user searches.
- **Actual Results:** Describe what happens instead.
- **Severity:** Rate the severity of the bug (e.g., critical, major, minor).
- **Attachments:** Include any screenshots or logs that help illustrate the issue.

9. Complete the sentences using the correct word:

| Words:

| automated, regression, acceptance, integration, manual |

1. After making changes to the system, we need to run _____ testing to ensure nothing else was broken.
2. The company used _____ testing to ensure that the system meets the client's requirements before release.
3. _____ testing involves combining different software components to see how they interact with one another.
4. It's important to use _____ testing to handle repetitive tasks and improve

efficiency.

5. The team chose _____ testing to check how the new feature works in various complex scenarios.

10. Answer the questions:

1. Why is **regression testing** important after making changes to the software?
2. What is the difference between **unit testing** and **integration testing**?
3. How can **automated testing** improve the software testing process?
4. Why should you start testing early in the software development lifecycle?
5. What types of issues can **system testing** help identify?

Extra Online Practice

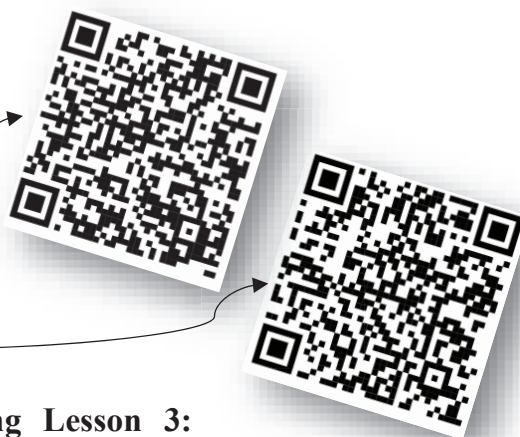
Vocabulary:

[Quizlet – Unit 8.2 Flashcards & Learn](#)

Grammar:


[Narrative tenses – all past tenses](#)

[Exercises 1–3 on Test-English.com](#)



☒ Recommended activity before starting Lesson 3:

Play **Quizlet Live** (INDIVIDUAL MODE, 3 TIMES) using

vocabulary from Unit 8.2:  <https://quizlet.com/ua/949537157/unit-82-flash-cards/?i=j03j6&x=1jqt>

Lesson 3

Web Application Testing

Lead-in 1.  Discuss in pairs or small groups:

1. What web applications do you use daily? What issues or bugs have you experienced while using them?
2. Why do you think web applications need specific testing methods compared to desktop software or mobile apps?
3. What factors make web application testing complex?

2. Decide if you agree (✓) or disagree (✗). Be ready to explain.

- a) Web applications are easier to test than mobile apps.
- b) Performance testing is not important for small web apps.
- c) Web browsers can affect how an application works.
- d) Security testing is one of the most important parts of web testing.

3. Read the text:

 8.3.

Web Application Testing

Web application testing is a critical process that ensures that a web app performs efficiently, securely, and meets users' expectations. Unlike traditional software, web applications need to function well on multiple platforms, devices, and browsers, making testing more complex. The testing process involves checking for security vulnerabilities, performance issues, browser compatibility, and overall user experience.

Types of Web Application Testing

Functional Testing: Verifies that the web application's features and functions work according to the requirements. This involves testing forms, buttons, links, and user interactions to ensure they function properly.

Cross-browser Testing: Ensures the web app works consistently across different browsers (e.g., Chrome, Firefox, Safari, Edge) and their versions. Cross-browser compatibility is crucial for providing a consistent user experience.


Responsiveness Testing: Ensures that the web app adapts to different screen sizes and devices, from desktops to tablets and mobile phones. With the increasing use of mobile devices, ensuring responsiveness is key.

Security Testing: Protects the web application from security threats such as SQL injection, cross-site scripting (XSS), and unauthorized access. Security vulnerabilities can result in data breaches or system exploitation.

Performance Testing: Evaluates the speed and responsiveness of the web application. This testing ensures that the web app loads quickly, especially under heavy traffic conditions, and performs well in real-world scenarios.

Load Testing: Simulates multiple users accessing the web application simultaneously to see how it handles high traffic volumes. It helps determine the breaking point of the app and its ability to scale.

Usability Testing: Checks how easy and intuitive the web app is to use. This ensures that users can navigate the app easily and complete tasks without confusion or difficulty.

API Testing: Tests the communication between the web application and other systems or services via APIs. This is important to ensure smooth data exchange and functionality between interconnected systems. 

Vocabulary

3. Match the highlighted words in the text with their definitions.

Word/Phrase	Definition
1. cross-browser testing	a) the ability of a web application to function well on various screen sizes
2. responsiveness	b) testing the web app to ensure it is easy to use and navigate
3. security testing	c) testing the communication between different systems or software via APIs
4. usability testing	d) testing how quickly the web app responds to user interactions
5. load testing	e) testing a web application to ensure it works correctly on different browsers
6. performance testing	f) evaluating the web app's performance under heavy user traffic
7. SSL/TLS	g) ensuring the web app works on different devices, browsers, and OS versions
8. SQL injection	h) ensuring the web app is protected against unauthorized access or attacks
9. compatibility testing	i) encryption protocols for secure communication over the internet
10. API testing	j) a security vulnerability where an attacker can execute arbitrary SQL queries

4. 🖱️ Play the Quizlet Match game by following

this link:

<https://quizlet.com/943274861/match?funnelUUID=5d03b6b2-d819-4b6f-818a-668c250d6fb3>

<https://quizlet.com/ua/949539068/uint-83-flash-cards/?i=j03j6&x=1jqt>



5. Group Activity: Web App Testing Plan

Task:

Work in groups to create a basic testing plan for a new e-commerce web application. Consider the following testing types:

- Functional Testing
- Cross-browser Testing
- Performance Testing
- Security Testing

For each type, write a few sentences explaining what you will test and why it is essential for the web app's success.

5. Exercise: Spot the Bugs

Scenario:

You are testing a web-based news platform. During testing, you encounter the following issues:

1. The “Subscribe” button doesn’t work in Firefox.
2. The website takes more than 5 seconds to load on mobile devices.
3. The user login form accepts passwords that are too short (fewer than 4 characters).

Task:

In pairs, identify the type of testing that would uncover each of these issues. Explain why each test is necessary for a good user experience.

6. Web Application Testing Phases

Mini Case Study:

You're part of a team developing a social networking web application. Before launch, the app needs to undergo extensive testing. Outline the different phases of testing, including the following:

- Functional Testing
- Usability Testing
- Security Testing
- Performance Testing

For each phase, explain the types of tools or methods you would use (e.g., Selenium for automated testing, JMeter for load testing, OWASP ZAP for security testing).

7. Web Application Testing Best Practices

1. Test Early and Often:

Begin testing during the development process. Early bug detection can save time and prevent major issues later in the development lifecycle.

2. Automate Repetitive Tests:

Use automated tools like Selenium, Cypress, or Puppeteer to automate repetitive tasks like functional and regression tests, freeing up testers for more complex issues.

3. Focus on Security:

Use tools like OWASP ZAP or Burp Suite to identify security vulnerabilities such as SQL injection or cross-site scripting (XSS). Security testing should be a priority to protect users' data.

4. Test on Real Devices:

Simulate real-world conditions by testing on actual devices and browsers. Emulators may not always reflect real performance or compatibility issues, especially on mobile devices.

5. Document Bugs Thoroughly:

Write clear and detailed bug reports with steps to reproduce, expected vs. actual behavior, and any relevant screenshots or logs. Clear documentation helps developers resolve issues faster.

8. Practice Exercise: Security Testing Example

Scenario:

You are testing an online banking application, and you suspect that it may be vulnerable to SQL injection attacks.

Task:

Write down the steps you would take to test for SQL injection vulnerabilities. Which tools might you use, and what should you look for in the web application's behaviour?

9. Complete the sentences using the correct word:

| Words: | **responsiveness**, **performance**, **API**, **cross-browser**, **security** |

1. The team needs to test the app's _____ to ensure that it works well across different browsers and their versions.
2. A key part of web app testing is _____ testing, where we verify that the app adapts to various screen sizes.
3. To protect user data, we need to perform thorough _____ testing to identify and resolve vulnerabilities.
4. _____ testing ensures that the web app can handle high traffic volumes without slowing down.
5. When different systems communicate via web services, we need to conduct _____ testing to ensure data exchange works properly.

10. Answer the questions:

1. Why is **cross-browser testing** crucial for web applications?
2. What is the importance of **security testing** in web app development?
3. How can **load testing** help improve the performance of a web application?
4. What is the role of **API testing** in web applications that rely on third-party services?
5. How does **usability testing** contribute to a better user experience?

Extra Online Practice

Vocabulary:

Quizlet – Unit 8.3 Flashcards & Learn

Grammar:

Usually, used to, be used to, get used to
Exercises 1–3 on Test-English.com



☒ Recommended activity before starting Lesson 2:

Play **Quizlet Live** (INDIVIDUAL MODE, 3 TIMES) using vocabulary from Unit 8.3:

 <https://quizlet.com/ua/949539068/unit-83-flash-cards/?i=j03j6&x=1jqt>

Vocabulary Revision

Unit 8

1. WORK IN PAIRS. TAKE TURNS PLAYING THE ROLES OF DESCRIBER AND GUESSER USING VOCABULARY FROM UNIT 8.

◇ Detailed instructions can be found on page 37

Term	Definition
functionality	the range of operations that can be performed by a system or software
reliability	the ability of software to perform its functions under stated conditions
usability	how easy and intuitive the software is for users
efficiency	the software's ability to use resources optimally
maintainability	how easily software can be modified or updated
portability	the software's ability to work across different environments and platforms
security	the ability to protect the software from unauthorized access or harm
performance	how quickly and effectively the software responds and performs tasks
bug	a flaw or fault in a software program that causes it to operate incorrectly
test case	a set of conditions under which a tester will determine whether an application works
unit testing	testing individual components of software to ensure they function correctly
integration testing	testing the interaction between different software components
regression testing	testing after changes have been made to ensure that existing functionality works
acceptance testing	testing to confirm the system meets the requirements and is ready for release
automated testing	using tools to run tests automatically, without human intervention
manual testing	tests that are executed by human testers without the use of automated tools
test coverage	the percentage of code tested by the test cases
quality assurance (QA)	ensuring that the software development process follows established standards
Cross-browser testing	Testing a web application to ensure it works correctly on different browsers.
Responsiveness	The ability of a web application to function well on various screen sizes.

2. THREE-SENTENCE CHALLENGE. Each participant **selects three words** from the shared word list (*pick the ones you find most challenging*) and forms **three sentences** using these words. Work in pairs. Read your sentences to your partner, replacing the chosen word with "gap" while reading. Your partner needs to guess the missing word.

3. PLAY THE QUIZLET LIVE GAME (TEAM MODE) to review the vocabulary learned in **UNIT 8**

<https://quizlet.com/ua/949539952/unit-8-81-82-83-flash-cards/?i=j03j6&x=1jqt>



4. TAKE the UNIT 8 VOCABULARY QUIZLET TEST to check your understanding of key terms from this UNIT.

<https://quizlet.com/949539952/test?answerTermSides=2&promptTermSides=4&questionCount=20&questionTypes=15&showImages=true>



5. 🎲 VOCABULARY GAME: ALIAS

Play a fun team-based vocabulary challenge!

♦ See full game instructions on page 37.

Get ready to explain, guess, and compete using key terms from Unit 8!

For Printable Vocabulary Materials for the Game “Alias,” go to page 237.

LIST OF AUDIO RECORDING LINKS FOR THE TEXTS (UNITS 1–8)

Audio 1.1 — Unit 1, Lesson 1 [<https://youtu.be/vIoDmO0vvR8>]

Audio 1.2 — Unit 1, Lesson 2 [https://youtu.be/5-C_fNYcsd4]

Audio 2.1 — Unit 2, Lesson 1 [https://youtube.com/shorts/jcs7E-_CRIs?feature=share]

Audio 2.2 — Unit 2, Lesson 2 [<https://youtu.be/CZ5jluqZfUQ>]

Audio 3.1 — Unit 3, Lesson 1 [<https://youtu.be/bbVt8XMTtjs>]

Audio 3.2 — Unit 3, Lesson 2 [<https://youtu.be/JM8gyljDWyU>]

Audio 4.1 — Unit 4, Lesson 1 [https://youtu.be/_plWIP1ACLI]

Audio 4.2 — Unit 4, Lesson 2 [<https://youtube.com/shorts/PznWgMOB3Es?feature=share>]

Audio 4.3 — Unit 4, Lesson 3 [https://youtube.com/shorts/_V_8gLheaMs?feature=share]

Audio 5.1 — Unit 5, Lesson 1 [<https://youtu.be/gJK7uPoKSNg>]

Audio 5.2 — Unit 5, Lesson 2 [<https://youtube.com/shorts/G-PM55dGG2w?feature=share>]

Audio 5.3 — Unit 5, Lesson 3 [<https://youtu.be/mL4kD4PCPwk>]

Audio 6.1 — Unit 6, Lesson 1 [<https://youtube.com/shorts/QPN0xbialrc?feature=share>]

Audio 6.2 — Unit 6, Lesson 2 [<https://youtube.com/shorts/2IkI79FKk7g?feature=share>]

Audio 6.3 — Unit 6, Lesson 3 [<https://youtu.be/0zZYv-x5BHg>]

Audio 7.1 — Unit 7, Lesson 1 [<https://youtu.be/NE5eKLINALM>]

Audio 7.2 — Unit 7, Lesson 2 [<https://youtu.be/swk9aq6YaqI>]

Audio 7.3 — Unit 7, Lesson 3 [<https://youtu.be/-S22fTfo3ro>]

Audio 8.1 — Unit 8, Lesson 1 [<https://youtube.com/shorts/hIeqjPGm-x0?feature=share>]

Audio 8.2 — Unit 8, Lesson 2 [<https://youtu.be/b3N6Fx8afVU>]

Audio 8.3 — Unit 8, Lesson 3 [https://youtube.com/shorts/DinezE3_Ivo?feature=share]

Printable Vocabulary Materials for Game Alias

Alias Game Cards — Unit 1 (Printable Material, p. 38)

advocate	integrity	rigorously
accomplish	exhibit	milestones
remarkable	rapid expansion	release (v)
software specification	sequence	increment
software design	requirement	incremental software
software architecture	resemble	crucial
software engineering	deem	adjustment
in this regard	in the pipeline	timeline
commit	embedded system	deliverable
commitment	capabilities	iterative
analysis	long-lasting	sequential
specification	enhancement	software QA
maintenance	issue (v)	

Alias Game Cards — Unit 2 (Printable Material, p. 65)

advocate	consistently	rigorously
accomplish	integrity	milestones
remarkable	exhibit	release (v)
software specification	rapid expansion	increment
software design	sequence	incremental software
software architecture	requirement	crucial
software engineering	resemble	adjustment
software quality assurance	deem	timeline
in this regard	in the pipeline	deliverable
commit	embedded system	iterative
commitment	capabilities	sequential
analysis	long-lasting	maintenance
specification	enhancement	issue (v)

Alias Game Cards — Unit 3 (Printable Material, p. 102)

decimals	consists of	encapsulating
integers	variables	property
string	functions	blueprint
column	control structures	immutable
numeric	vast	intricate
row	computation	harness
field	garbage collection	akin
record	generics	incongruity
redundant	delegates	be flawed
primary key	fine-grained control	incongruity
divided by	inline	be flawed
subtract	sequence	multiply

Alias Game Cards — Unit 4 (Printable Material, p. 126)

release	staging area	thoroughly
automate	tremendous	upfront
deployment pipeline	due to	branch
repository	comprehensive	adopt
distributed	harmful	deployment pipeline
merge	emphasize	Continuous Integration (CI)
branch	incremental	Continuous Deployment (CD)
clone	manageable	Feature Flag
commit	iterative	Canary Deployment
pull request	exist	feature
version control	differ	conflict

Alias Game Cards — Unit 5 (Printable Material, p. 153)

lifecycle	constraint	kanban board
feasibility	milestone	sprint
deployment	resource allocation	backlog
analysis	deliverable	workflow
requirement	contingency	release
maintenance	dependency	kanban board
design	stakeholder	iteration
implementation	risk management	stakeholder
testing	baseline	

Alias Game Cards — Unit 6 (Printable Material, p. 180)

channels	asynchronous communication	NRN
direct message (DM)	message threading	ASAP
integration	@mentions	BTW
threads	tone	TL; DR
file sharing	read receipts	formal tone
notifications	concise communication	cc (carbon copy)
search functionality	mute notifications	emoji etiquette
video conferencing	EOD	FYI

Alias Game Cards — Unit 7 (Printable Material, p. 205)

template	Legend	Eye Contact
transition	Bar Chart	Storytelling
slide Deck	Line Graph	Visual Aids
animation	Pie Chart	Engagement
collaborate	Flowchart	channels
presenter View	Data Point	Slide Deck
export	Technical Jargon	Axis

Alias Game Cards — Unit 8 (Printable Material, p. 228)

functionality	bug	test coverage
reliability	test case	quality assurance (QA)
usability	unit testing	Cross-browser testing
efficiency	integration testing	Responsiveness
maintainability	regression testing	manual testing
portability	acceptance testing	performance
security	automated testing	

ALPHABETICAL GLOSSARY OF KEY TERMS (UNITS 1–8)

A

acceptance testing	testing to confirm the system meets the requirements and is ready for release
access control	a security measure that defines who can access a computer, device, or network, when they can access it, and what actions they can take while accessing it.
accomplish	to do, make happen, succeed in, carry through
adjustment	a slight change made to something to make it fit, work better, or be more suitable, or the act of making such a change
adopt	to accept or start to use something new
advocate	to support; to be in favour of
akin	having some of the same qualities; similar
algorithm	a step-by-step procedure for solving a problem
analysis	a detailed examination of the elements or structure of something.
animation	a visual effect applied to text or objects to make them move, appear, or disappear
API	a set of routines, protocols, and tools for building software applications
API testing	testing the communication between different systems or software via APIs.
architecture design	describes hardware, software and network infrastructure to be used
ASAP	as soon as possible (used to express urgency in completing a task or action)
asynchronous communication	a form of communication where participants do not need to be present at the same time
automate	to use technology to perform tasks without human intervention
automated testing	using tools to run tests automatically, without human intervention
automation	the use of software to perform repetitive tasks without manual intervention

axis the horizontal (x-axis) or vertical (y-axis) line on a graph showing data points

B

backlog a list of tasks or work items that need to be addressed

bar chart a type of graph that uses rectangular bars to represent data comparisons

baseline an initial plan or budget used for comparison as the project progresses

be flawed to contain defects or imperfections, not being perfect.

blueprint a detailed plan or design that serves as a guide for constructing something.

branch a separate line of development in version control systems

brevity using only a few words or lasting only a short time

BTW by the way (used to introduce an additional piece of information)

bug a flaw or fault in a software program that causes it to operate incorrectly

C

Canary Deployment a deployment strategy that releases code to a small group of users first.

capabilities the ability to do something:

cc (carbon copy) adding someone to an email to keep them informed without needing a direct

channels dedicated spaces within a chat app where team members can discuss specific topics or projects

CI/CD pipeline (continuous integration and continuous deployment) a series of steps that must be performed in order to deliver a new version of software.

clerical (of a job or person) concerned with or relating to work in an office, especially routine documentation and administrative tasks

clone to create a copy of an existing repository

collaborate work together with others on the same document or presentation in real time

collaboration working together as a team to achieve a common goal

column	a vertical series of cells in a table.
command	to deserve and get something good, such as attention, respect, or a lot of money
commit	to save changes to a local repository
commitment	a promise or pledge to do something
compatibility testing	ensuring the web app works on different devices, browsers, and OS versions.
compensation package	the mix of salary, benefits, and other incentives that employees receive from the organization
compliance	the fact of obeying a particular law or rule, or of acting according to an agreement
comprehensive	covering or including everything
computation	the process of performing mathematical or logical operations.
concise	short and clear, expressing what needs to be said without unnecessary words
concurrency	the ability of a program to execute multiple tasks simultaneously
conflict	a situation where changes in two versions of a file contradict each other
considerate	caring about and respectful of others
consistently	continually; regularly;
consists of	is made up of
constraint	a limitation or restriction on time, resources, or scope
contingency	a plan designed to take account of possible future events or circumstances
Continuous Deployment (CD)	the automated release of code directly into production systems.
Continuous Integration (CI)	a method to automatically test and merge code changes into the main branch
control structures	constructs used to control the flow of a program (e.g., loops, conditionals).
cross-browser testing	testing a web application to ensure it works correctly on different browsers.

crucial extremely important; vital in resolving something

curious eager to know or learn something

D

dashboard a real-time display of important project metrics and data

data cleaning is the process of identifying and correcting errors and inconsistencies in data sets so that they can be used for analysis

data mining the use of a variety of statistical analysis tools to uncover previously unknown patterns in the data stored in databases or relationships among variables

Data Point a specific value or measurement plotted on a graph

Data Structure a particular way of organizing and storing data such as an array, table, etc.

data system includes the processes used to capture and the methods used to store data coming from a number of external and internal sources; the creation of a database

decimals a number expressed using a system of counting based on the number ten

deem (v.) to think, believe; to consider, have an opinion

delegates types that represent references to methods with a specific parameter list and return type.

deliverable something that can be provided or achieved as a result of a process: a tangible or intangible product/result produced as part of a project

dependency a relationship between tasks, where one task relies on another to begin or complete

Deploy to release or distribute software to a production environment

deployment the release or distribution of software for use

deployment pipeline a series of automated stages that ensure code quality and enable smooth software release

design the process of creating a plan or drawing to show the function or workings of a system

differ to be different from something in some way, vary

direct message (DM) a private message sent between two individuals or a small group in a chat app

distractions	things that make it difficult to think or pay attention
distributed	shared or spread out across multiple locations
divided by	÷ or / ; to calculate the number of times that one number fits (exactly) into another: 10 divided by 5 is/equals 2
due to	because of

E

efficiency	the software's ability to use resources optimally
embedded system	a computer system that is part of a larger machine and which controls how that machine operates
emoji etiquette	guidelines for appropriate and professional use of emoticons in work-related conversations
emphasize	give special importance or prominence to (something) in speaking or writing.
encapsulating	the act of containing or covering something within another, often used to describe the grouping of data and methods within objects in OOP.
encompass	(v) to encircle, go or reach around; to enclose; to include with a certain group or class
end-to-end	from the very beginning of a process to the very end
engagement	actively involving your audience in the presentation through interaction or attention-grabbing techniques
enhancement	the process of improving the quality, amount, or strength of something
enterprise	a business organization in an area such as shipping, mining, railroads, or factories
EOD	end of day (used to signify the deadline for completing a task by the end of the workday)
estimate	to guess or calculate the cost, size, value, etc. of something
excessively	more important than anything else
exhibit	show, display, present, demonstrate
exist	to be real. To be found; to occur. To stay alive.
expand	to increase in size or amount

export	save a file in a different format, such as PDF or video
eye contact	maintaining visual connection with your audience to engage them while presenting

F

feasibility	the practicality of a proposed plan or method
feature	an important part of something
feature flag	a tool used to switch new features on or off without deploying code again
feedback loop	a system that provides feedback on the performance of a process or product
field	a single characteristic of data that appears in a table as a column
file sharing	the feature that allows users to upload and share documents, images, or other files
fine-grained control	the ability to manage and manipulate system resources or program details with high precision.
flowchart	a diagram that shows a process or workflow
formal tone	a style of communication used in business or professional settings, marked by politeness
functionality	the range of operations that can be performed by a system or software
functions	blocks of code designed to perform specific tasks.
FYI	for your information (used to share important information without expecting a response)

G

garbage collection	the automatic process of reclaiming memory by removing objects that are no longer in use.
gauge	to calculate an amount, especially by using a measuring device
generics	a feature that allows types to be defined more flexibly, providing type safety without committing to specific data types.

H

handle	to deal with;
harmful	causing or likely to cause harm; damaging

harness	the act of using or controlling something effectively, such as power or resources.
host	to set up and manage a meeting or call in which people can speak to each other over the internet using their computers
I	
identity	a person's name and other facts about who they are:
immutable	something that is unchangeable or cannot be altered once it has been created.
implementation	the process of putting a decision or plan into effect; execution
in the pipeline	being planned or in progress (refers to a sequence of steps or stages that data, tasks, or products go through in order to achieve a desired outcome)
in this regard	in connection with the point previously mentioned
incongruity	a deviation from what is expected, leading to a sense of inconsistency or contradiction.
increment	one of a series of increases, an enlargement, increase, addition
incremental	increasing gradually by regular degrees or additions
incremental software	a method of building software products in which a system is built piece-by-piece
inline	Within a program or procedure, often referring to code that is placed directly in the sequence of execution.
integers	(<i>n</i>) a whole number and not a fraction
integration (<i>Unit 6</i>)	the ability of chat tools to connect with other platforms, such as Google Drive or Trello
integration (<i>Unit 5</i>)	the ability of software to connect and work with other tools or platforms
integration (<i>Unit 4</i>)	the process of combining code changes from multiple developers into a shared codebase
integration testing	testing the interaction between different software components
integrity	(<i>n.</i>) honesty, high moral standards; an unimpaired condition, completeness, soundness
intricate	the quality of being complex and detailed, often requiring careful attention.

issue (v)	to produce or provide something official
iteration	the repetition of a process to achieve a desired outcome
iterative	a process that repeats a series of steps over and over until the desired outcome is obtained

K

kanban board	a visual tool to organize tasks and workflows using cards and columns
---------------------	---

L

latency	the delay between an instruction to transfer (= move) computer information and the information being transferred, for example over the internet
legend	a key that explains what the symbols, colors, or patterns represent in a chart or graph
lifecycle	the series of changes that a product or system goes through during its existence
line graph	a graph that uses lines to show changes over time
load testing	evaluating the web app's performance under heavy user traffic.
long-lasting	continuing for a long period of time

M

@mentions	a feature that allows users to directly notify or reference a specific person within a conversation
maintainability	how easily software can be modified or updated
maintenance	the process of preserving or keeping something in good condition
manageable	capable of being controlled, directed, or accomplished
manual testing	tests that are executed by human testers without the use of automated tools
merge	to combine multiple versions or changes into one unified version
message threading	keeping conversations organized by replying to specific messages within a chat
milestone	a significant point or event in a project
milestones	formal project review points used to assess progress and performance

monitoring the ongoing process of tracking and analyzing the performance of software systems

multiply the operation of repeated addition of the same number.

N

notifications automated alerts that inform team members about task updates or deadlines

NRN no response needed (used to indicate that the message is informational, and no reply is required)

numeric describes data that consists of numbers.

O

open source software with source code that anyone can inspect, modify, and enhance.

P

paramount more important than anything else

performance how quickly and effectively the software responds and performs tasks

Performance testing testing how quickly the web app responds to user interactions.

pie chart a circular chart divided into slices representing proportions

pipeline a series of automated steps that move code from development to production

portability the software's ability to work across different environments and platforms

presenter View a feature that allows the presenter to see notes and upcoming slides while the audience sees only the slides

primary key a field that uniquely identifies a record in a table

property a characteristic or attribute that belongs to something, especially in programming, it refers to data associated with an object.

pull request a request to merge changes from one branch into another

Q

quality assurance (QA) ensuring that the software development process follows established standards

R

rapid expansion	product acceptance is growing and investors become very interested
read receipts	a feature that indicates whether the recipient has seen the message
record	information provided to a journalist that can be released and attributed by name to the source
redundant	(adj.) extra, excess, more than is needed; repetitive;
regression testing	testing after changes have been made to ensure that existing functionality works
release (n)	a version of the software made available for users
release (v)	to make a product available for the public to buy, often with a celebration; launch
reliability	the ability of software to perform its functions under stated conditions
remarkable	unusual, extraordinary, worthy of attention
repository	a central location where data is stored and managed
requirement	something necessary or compulsory in a project or system
resemble	to be like or similar to
resource allocation	the process of assigning and managing assets in a way that supports project goals
responsiveness	the ability of a web application to function well on various screen sizes.
rigorously	in a careful way so that every part of something is looked at or considered to make certain it is correct or safe
risk management	identifying, assessing, and controlling threats to a project's objectives
rollback	the process of reverting to a previous version of software after deployment issues occur
row	a horizontal group of cells in a worksheet identified by numbers

S

scalable	used to describe a business or system that is able to grow or to be made larger
seamless	happening without any sudden changes, interruption, or difficulty

search functionality	the tool that allows users to find specific conversations or files within the chat history
security	the ability to protect the software from unauthorized access or harm
security testing	ensuring the web app is protected against unauthorized access or attacks.
sequence	a series of related events, actions, or items that follow each other in a particular order.
sequential	following a particular order, arranged serially
shape up	to develop; improve to reach an acceptable standard
Slide Deck	a collection of presentation slides used to deliver information during a talk
software architecture	the overall software structure that depicts the major system components and how they relate, interface, and interact with each other.
software design	produces a software solution to realize the software requirements.
software engineering	a discipline is focused on the research, education, and practice of engineering processes, methods, and techniques to significantly increase software productivity and software quality while reducing software costs and time to market.
software quality assurance	ensure that the development activities are carried out correctly
software specification	where customers and engineers define the software that is to be produced and the constraints on its operation.
sought-after	wanted by many people and usually of high quality or rare
specification	a detailed description of the design and materials used to make something.
sprint	a set period during which specific tasks must be completed in agile project management
SQL injection	a security vulnerability where an attacker can execute arbitrary SQL queries
SSL/TLS	Encryption protocols for secure communication over the internet.
staging area	a place to store changes before they are committed
stakeholder	a person or group with an interest in the outcome of the project
starting point	a place or position where something begins

stock option	a contract for the right to buy and sell shares at a later date or within a certain period at a particular price
storytelling	the act of delivering content in a narrative format to make the presentation more engaging
string	(n) a usually short piece of text consisting of letters, numbers, or symbols
subtract	to take away

T

task assignment	allocating specific tasks to team members in a project
technical jargon	specialized language used in specific professions that may be difficult for outsiders to understand
template	a pre-designed slide layout that you can use to create presentations quickly and consistently
test case	a set of conditions under which a tester will determine whether an application works
test coverage	the percentage of code tested by the test cases
testing	the process of evaluating the performance and quality of a system or product
the matter at hand	the job or matter that is important at the present moment
third-party library	reusable software components created by external entities, not by an application's primary developers
thoroughly	in a detailed and careful way
threads	a series of related messages that allow team members to follow a specific conversation topic
timeline	a graphic representation of the passage of time as a line.
TL;DR	too long; didn't read (used to summarize long content in brief)
tone	the overall feel or style of your message, which can affect how it is interpreted by the recipient
transition	an animation that occurs when moving from one slide to the next
tremendous	extraordinarily large in size or extent or amount or power or degree

U

UI	user interface: the way in which the information on a computer, phone, etc. and instructions on how to use it are arranged on the screen and shown to the user
unit testing	testing individual components of software to ensure they function correctly
upfront	before goods or services are received
usability	the degree to which a system is easy to learn and efficient and satisfying to use; how easy and intuitive the software is for users
usability testing	testing the web app to ensure it is easy to use and navigate.
UX	User Experience. This includes how a visitor / user navigates your site, how long they stay, whether or not they bounce, how they access your content (from page views to downloads), how they move around etc.

V

variables	elements in a program that store data values.
varied	incorporating a number of different types or elements; showing variation or variety.
vary	to be different or diverse;
vast	extensive in size, amount, or scope.
version control	a system for managing changes to code or documents over time
visual aids	tools like charts, graphs, or images that help illustrate your points during a presentation

W

workflow	a series of steps or processes needed to complete a task
version control	a system for managing changes to code or documents over time
workflow	a series of steps or processes needed to complete a task

REFERENCES

1. Brown G., Sargent B. Cambridge International AS Level Information Technology Student's Book. Hodder Education Group, 2020. 500 p.
2. ByteByteGo. CI/CD In 5 Minutes | Is It Worth The Hassle: Crash Course System Design #2, 2023. *YouTube*. URL: <https://www.youtube.com/watch?v=42UP1fxi2SY> (date of access: 30.10.2025).
3. Cambridge Dictionary | English Dictionary, Translations & Thesaurus. *Cambridge Dictionary | English Dictionary, Translations & Thesaurus*. URL: <https://dictionary.cambridge.org/> (date of access: 11.08.2023).
4. Computer History Museum. The art of writing software, 2014. *YouTube*. URL: <https://www.youtube.com/watch?v=QdVFvsCWxrA> (date of access: 30.07.2025).
5. Consumer Desire. 6 BEST free project management software | (2024), 2023. *YouTube*. URL: <https://www.youtube.com/watch?v=gJCS1gqk8ME> (date of access: 30.07.2025).
6. Create, explore, and host kahoots | Kahoot!. *Kahoot!*. URL: <https://create.kahoot.it/share/indirect-questions-challenge/deff670b-95e7-4878-86d4-32ed3ec505df> (date of access: 30.07.2025).
7. Database design basics - Microsoft Support. *Microsoft Support*. URL: <https://support.microsoft.com/en-gb/office/database-design-basics-eb2159cf-1e30-401a-8084-bd4f9c9ca1f5> (date of access: 30.07.2025).
8. EngX Space. How to become a great software developer – best advice from top-notch engineers, 2024. *YouTube*. URL: <https://www.youtube.com/watch?v=suATPK45sjk> (date of access: 30.07.2025).
9. Grammar - Test-English. *Test-English*. URL: <https://test-english.com/grammar-points/> (date of access: 05.08.2023).

10. Kung D. C. Software Engineering. 2nd ed. New York : McGraw Hill LLC, 2024. 666 p.
11. Learn English. B1-B2 grammar. URL: <https://learnenglish.britishcouncil.org/grammar/b1-b2-grammar> (date of access: 11.08.2023).
12. Pumble. 5 Best Team Chat Apps, 2021. *YouTube*. URL: <https://www.youtube.com/watch?v=hxyoVmMXmB8> (date of access: 30.07.2025).
13. Quizlet for teachers. *Quizlet*. URL: <https://quizlet.com/teachers> (date of access: 07.02.2024).
14. The Software Engineering Code of Ethics and Professional Practice. *Association for Computing Machinery*. URL: <https://www.acm.org/code-of-ethics/software-engineering-code#:~:text=The%20Software%20Engineering%20Code%20of%20Ethics%20and%20Professional%20Practice> (date of access: 01.08.2025).
15. Types of Software Engineering Roles | ClickUp. *ClickUp*. URL: <https://clickup.com/blog/types-of-software-engineering/> (date of access: 30.07.2025).
16. Vertabelo. Tutorial 3. references, 2014. *YouTube*. URL: <https://www.youtube.com/watch?v=skHBFfw05Oo> (date of access: 30.07.2025).
17. W3Schools.com. *W3Schools Online Web Tutorials*. URL: <https://www.w3schools.com> (date of access: 30.07.2025).
18. Wordsmith T. K. Decoding the Secrets of Top Programming Languages: JavaScript, Python, Java, C#, and C++. *Medium*. URL: <https://medium.com/@keyboardwordsmith/decoding-the-secrets-of-top-programming-languages-javascript-python-java-c-and-c-8b184f966bb7> (date of access: 30.07.2025).

Для нотаток

Для нотаток

Навчальний посібник «Hello, World! English Language Skills for Programmers» призначений для здобувачів бакалаврського рівня вищої освіти спеціальності «Інженерія програмного забезпечення» та споріднених спеціальностей ІТ-галузі. Він спрямований на формування професійної іншомовної комунікативної компетентності студентів засобами автентичних матеріалів, інтерактивних завдань і реальних комунікативних ситуацій ІТ-сфери. Посібник складається з восьми тематичних модулів, у яких поєднано мовний і фаховий зміст, а також передбачено елементи гейміфікації та цифрові інструменти, що підвищують мотивацію й залученість здобувачів освіти.

Навчальне видання

Махович Інна Анатоліївна

HELLO, WORLD! АНГЛІЙСЬКА МОВА ДЛЯ ПРОГРАМІСТІВ

Навчальний посібник
(Англійською мовою)

Рекомендовано Вченою радою Київського національного університету технологій та дизайну як навчальний посібник для студентів бакалаврського рівня вищої освіти з дисципліни «Іноземна мова фахового спрямування» спеціальності F2 «Інженерія програмного забезпечення»

Автор І. Махович
Відповідальний за поліграфічне виконання Л. Л. Овечкіна

Підп. до друку 22.09.2025 р. Формат 60x84 1/16.
Ум. друк. арк. 14,87. Облік. вид. арк. 11,65. Наклад 15 пр. Зам. 2286.

Видавець і виготовлювач Київський національний університет технологій та дизайну.
вул. Мала Шияновська, 2, м. Київ-11, 01011.

Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготівників і розповсюджувачів видавничої продукції ДК № 993 від 24.07.2002.