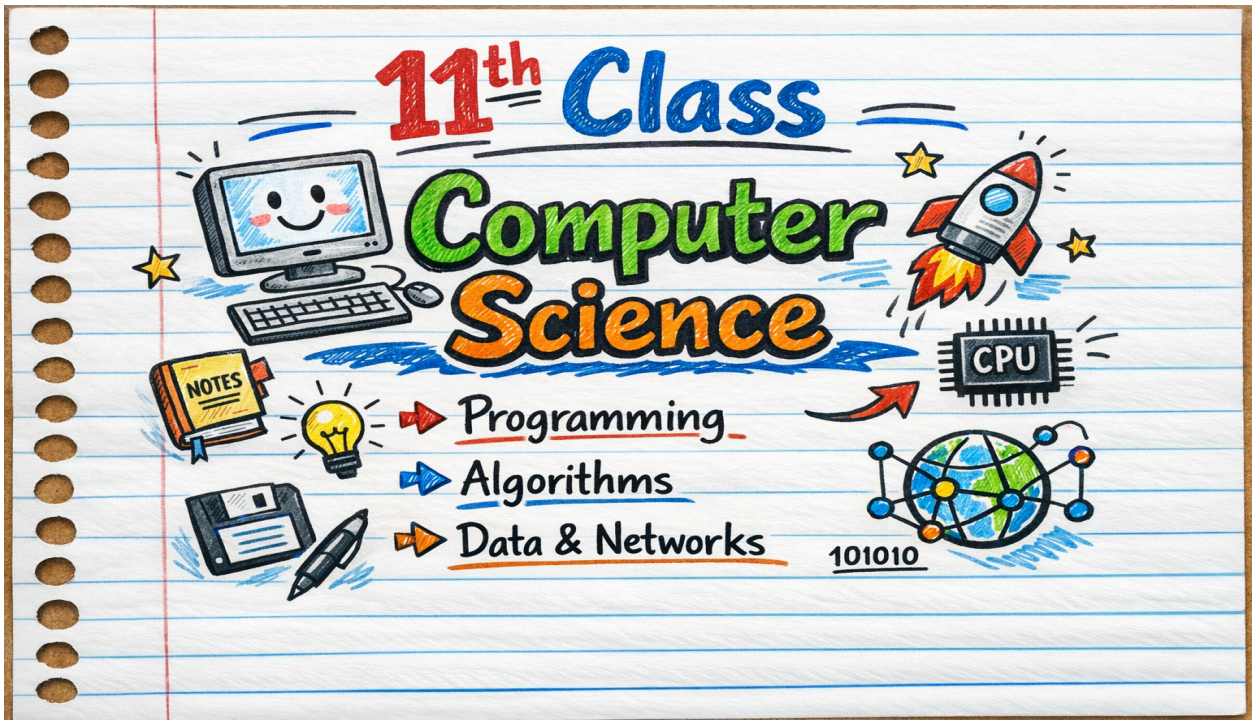


## Class: 11th

## Subject: Computer Science

## Unit 1: Introduction to Software Development

---

## ❖ Important MCQs:

**1. The primary purpose of SDLC is to:**

(a) increase coding speed

(b) deliver high-quality software within time and cost constraints ✅

(c) eliminate testing

(d) design hardware

**2. SDLC is defined as:**

(a) a programming language

(b) a framework that defines stages of software development ✅

(c) a database system

(d) an operating system

**3. The first phase of SDLC is:**

(a) Design

(b) Coding

(c) Requirement Gathering ✅

(d) Testing

**4. The main goal of requirement gathering is to:**

(a) write code

(b) understand user needs and expectations ✅

(c) test the system

(d) deploy software

**5. Interviews and surveys are used to:**

(a) test software

(b) collect user requirements ✅

(c) design interface

(d) deploy applications

**6. Observations help to:**

(a) improve coding

(b) identify real-world user problems ✅

(c) reduce system cost

(d) design database

**7. Functional requirements describe:**

(a) how the system performs

(b) what the system should do ✅

(c) system performance only

(d) system security only

**8. Non-functional requirements describe:**

(a) system tasks

(b) how the system performs ✅

(c) user interaction

(d) system features only

**9. “User registration” is an example of:**

(a) Non-functional requirement

(b) Functional requirement ✅

(c) Technical requirement

(d) Operational requirement

**10. “System handles 1000 users” is an example of:**

(a) Functional requirement

(b) Non-functional requirement ✅

(c) Design requirement

(d) Coding requirement

**11. “System available 99.9% of the time” refers to:**

(a) Performance

(b) Reliability ✅

(c) Security

(d) Functionality

**12. The main benefit of using a framework is:**

(a) no need for coding

(b) availability of predefined components ✅

(c) elimination of testing

(d) automatic deployment

**13. Frameworks help developers to:**

(a) manage hardware

(b) avoid rewriting common functionalities ✅

(c) only design UI

(d) perform testing

**14. Django is an example of a:**

(a) Programming language

(b) Framework ✅

(c) Database

(d) Operating system

**15. Software development is the process of:**

(a) designing hardware

(b) creating programs for specific tasks ✅

(c) managing networks

(d) repairing systems

**16. SDLC ensures that software is:**

(a) randomly developed

(b) systematically developed and maintained ✅

(c) only coded

(d) only tested

**17. A non-functional requirement example is:**

(a) User login

(b) Book borrowing

(c) System security ✅

(d) Account creation

**18. A functional requirement example is:**

(a) System performance

(b) System reliability

(c) Book search feature ✅

(d) System availability

**19. Document review in requirement gathering is used to:**

(a) improve coding

(b) study existing system information ✅

(c) design UI

(d) test system

**20. SDLC helps to:**

(a) avoid planning

(b) develop software in structured stages ✅

(c) reduce user involvement

(d) eliminate testing

**21. In the SDLC design phase, the primary focus is to:**

(a) implement code directly

(b) translate requirements into a structured solution plan ✅

(c) perform system testing

(d) deploy software to users

**22. A flowchart in software design is mainly used to:**

(a) execute programs

(b) represent logical flow of processes visually ✅

(c) store data

(d) test performance

**23. A software model (mockup) is used to:**

(a) debug programs

(b) represent user interface and system interaction before coding ✅

(c) replace testing phase

(d) manage deployment

**24. Software architecture refers to:**

(a) programming syntax

(b) overall system structure and interaction of components ✅

(c) database only

(d) user interface design only

**25. Which phase converts design into executable code?**

(a) Testing

(b) Coding/Development ✅

(c) Deployment

(d) Maintenance

**26. The main purpose of coding is to:**

(a) design system flow

(b) implement system logic using programming language ✅

(c) test software performance

(d) install software

**27. Testing is best described as:**

(a) designing system structure

(b) verifying software quality by detecting errors and bugs ✅

(c) writing program code

(d) deploying software

**28. Which type of testing checks system behavior under heavy load?**

(a) Functionality Testing

(b) Performance Testing ✅

(c) Compatibility Testing

(d) Unit Testing

**29. Compatibility testing ensures that software:**

(a) runs only on one device

(b) works across different devices and operating systems ✅

(c) is error-free in code

(d) is fast in execution

**30. The maintenance phase in SDLC is mainly responsible for:**

(a) initial coding

(b) continuous updates, bug fixes, and system improvement ✅

(c) requirement collection

(d) software design

**31. Software development methodologies are mainly used to:**

(a) increase coding speed only

(b) guide structured planning and execution of software projects ✅

(c) remove testing phase

(d) design hardware systems

**32. Software process models are best described as:**

(a) programming languages

(b) abstract representations of SDLC processes ✅

(c) debugging tools

(d) database systems

**33. A key benefit of process models is:**

(a) reducing team size

(b) improving predictability and risk management ✅

(c) eliminating coding

(d) avoiding planning

**34. Efficiency in software process models means:**

(a) more coding errors

(b) reduced wasted effort during development ✅

(c) slower development

(d) no testing required

**35. Quality in process models is ensured by:**

(a) skipping testing

(b) integrating QA throughout SDLC stages ✅

(c) reducing documentation

(d) avoiding design phase

**36. The Waterfall Model is best described as:**

(a) iterative model

(b) linear and sequential model ✅

(c) incremental model

(d) spiral model

**37. In Waterfall Model, the next phase starts when:**

(a) previous phase is partially completed

(b) previous phase is fully completed ✅

(c) testing starts

(d) deployment is done

**38. The correct order of Waterfall phases is:**

(a) Design ⟶ Requirements ⟶ Coding

(b) Requirements ⟶ Design ⟶ Implementation ⟶ Testing ⟶ Deployment ⟶ Maintenance ✅

(c) Coding ⟶ Testing ⟶ Requirements

(d) Deployment ⟶ Design ⟶ Coding

**39. The main limitation of Waterfall Model is:**

(a) too flexible

(b) difficulty in handling changes after phase completion ✅

(c) requires no documentation

(d) no testing phase

**40. Waterfall Model is most suitable for:**

(a) large dynamic projects

(b) projects with fixed and clear requirements ✅

(c) AI-based systems

(d) continuously changing systems

**41. Agile Methodology is mainly based on:**

(a) fixed planning only

(b) flexibility and iterative development ✅

(c) hardware design

(d) sequential coding only

**42. Agile development works in:**

(a) single phase

(b) long cycles

(c) short iterations (sprints) ✅

(d) random steps

**43. Continuous Integration in Agile means:**

(a) writing code once

(b) frequently merging code to detect errors early ✅

(c) skipping testing

(d) delaying deployment

**44. Test-Driven Development (TDD) means:**

(a) testing after deployment

(b) writing tests before writing code ✅

(c) no testing required

(d) testing only final product

**45. Pair Programming involves:**

(a) one developer coding alone

(b) two developers working together at one workstation ✅

(c) team testing only

(d) automated coding

**46. One major advantage of Agile is:**

(a) fixed requirements

(b) flexibility to handle changing requirements ✅

(c) no customer involvement

(d) no testing required

**47. Agile improves customer satisfaction because:**

(a) software is never delivered

(b) frequent updates and feedback are provided ✅

(c) no communication is needed

(d) testing is skipped

**48. A limitation of Agile is:**

(a) too structured

(b) difficulty in managing large teams and projects ✅

(c) no flexibility

(d) no coding required

**49. Agile projects are less predictable because:**

(a) no planning is done

(b) requirements evolve during development ✅

(c) no testing is done

(d) coding is not used

**50. Stakeholder involvement in Agile is:**

(a) optional

(b) continuous and essential throughout the project ✅

(c) only at the end

(d) not required

**51. Project planning in software development is similar to:**

(a) cooking food

(b) planning a trip with time, cost, and destination considerations ✅

(c) coding only

(d) testing only

**52. Comprehensive project planning means:**

(a) starting coding immediately

(b) considering all project details before development starts ✅

(c) skipping design phase

(d) reducing team size

**53. Project timelines are used to:**

(a) increase cost

(b) decide duration of each project phase ✅

(c) eliminate testing

(d) reduce documentation

**54. Setting timelines helps to:**

(a) delay project delivery

(b) keep project on schedule and organized ✅

(c) avoid planning

(d) increase risks

**55. Cost estimation in software projects refers to:**

(a) predicting project bugs

(b) estimating total expenses required for completion ✅

(c) designing UI

(d) testing software only

**56. One key factor affecting cost estimation is:**

(a) color scheme

(b) development team size and expertise ✅

(c) number of meetings

(d) user feedback only

**57. Technology stack affects project cost because:**

(a) it has no impact

(b) some technologies require more resources and expertise ✅

(c) it reduces testing

(d) it removes coding

**58. Longer project duration usually results in:**

(a) lower cost

(b) higher cost due to resource usage over time ✅

(c) no cost change

(d) no development needed

**59. Risk management in software projects involves:**

(a) ignoring problems

(b) identifying and controlling potential risks ✅

(c) removing testing phase

(d) increasing errors

**60. The first step in risk management is:**

(a) testing

(b) identifying risks ✅

(c) coding

(d) deployment

**61. Risk analysis involves:**

(a) writing code

(b) evaluating probability and impact of risks ✅

(c) designing UI

(d) fixing bugs only

**62. Mitigation strategies are used to:**

(a) increase risks

(b) reduce risk impact or likelihood ✅

(c) skip planning

(d) eliminate development

**63. Execution phase mainly involves:**

(a) requirement gathering

(b) actual coding and development work ✅

(c) cost estimation

(d) risk analysis

**64. Execution phase requires:**

(a) no teamwork

(b) coordination and continuous updates among team members ✅

(c) only planning

(d) no communication

**65. Quality Assurance (QA) ensures that:**

(a) software is expensive

(b) software meets quality standards and works correctly ✅

(c) coding is skipped

(d) requirements are ignored

**66. Graphical representation of software systems is used to:**

(a) increase coding speed

(b) simplify system understanding using visual diagrams ✅

(c) remove testing phase

(d) replace programming

**67. UML stands for:**

(a) Unified Modeling Language ✅

(b) Universal Machine Language

(c) Unified Management Logic

(d) User Modeling Language

**68. UML is mainly used to:**

(a) write program code

(b) visualize software system design ✅

(c) execute software

(d) store data

**69. Use case diagrams represent the system from the perspective of:**

(a) developers

(b) users (actors) ✅

(c) database

(d) hardware

**70. A use case describes:**

(a) system architecture

(b) interaction between user and system to achieve a goal ✅

(c) coding structure

(d) database tables

**71. Use case diagrams are mainly used for:**

(a) hardware design

(b) capturing functional requirements ✅

(c) writing code

(d) performance testing

**72. The first step in identifying use cases is:**

(a) writing code

(b) identifying actors ✅

(c) testing system

(d) deployment

**73. In UML, an actor represents:**

(a) database

(b) user or external system interacting with system ✅

(c) program code

(d) hardware device only

**74. Class diagram is used to represent:**

(a) system flow

(b) structure and relationships of system components ✅

(c) user behavior

(d) testing process

**75. In class diagram analogy, a "box" represents:**

(a) method

(b) class/container structure ✅

(c) actor

(d) sequence flow

**76. Attributes in a class diagram represent:**

(a) actions

(b) properties or data of a class ✅

(c) system flow

(d) user roles

**77. Methods in a class diagram represent:**

(a) system design

(b) actions or behaviors of a class ✅

(c) database structure

(d) testing steps

**78. Sequence diagrams show:**

(a) system architecture

(b) interaction between objects in time order ✅

(c) database design

(d) class structure

**79. Sequence diagrams focus on:**

(a) static structure

(b) message flow between objects over time ✅

(c) hardware design

(d) UI design only

**80. Activity diagrams represent:**

(a) system classes

(b) flow of activities in a process ✅

(c) database tables

(d) user roles

**81. In a restaurant system example, activity diagram shows:**

(a) coding process

(b) order → preparation → delivery flow ✅

(c) database design

(d) system architecture

**82. UML is useful in software planning because it:**

(a) replaces coding

(b) helps design system before implementation ✅

(c) removes testing

(d) avoids documentation

**83. UML diagrams improve communication between:**

(a) hardware components

(b) technical and non-technical stakeholders ✅

(c) only programmers

(d) only testers

**84. During development, UML diagrams are used to:**

(a) write final code directly

(b) understand system structure and relationships ✅

(c) skip design phase

(d) deploy software

**85. Use case diagrams mainly help in:**

(a) database optimization

(b) understanding user-system interactions and requirements ✅

(c) hardware configuration

(d) coding syntax

**86. Design patterns in software development are:**

(a) programming languages

(b) common reusable solutions to software design problems ✅

(c) hardware components

(d) testing tools

**87. Design patterns mainly help developers to:**

(a) increase hardware speed

(b) make coding easier, faster, and more consistent ✅

(c) remove requirement gathering

(d) avoid deployment

**88. Singleton pattern ensures that:**

(a) multiple objects are created

(b) only one instance of a class exists in the system ✅

(c) all classes are deleted

(d) code is not reused

**89. The main purpose of Singleton pattern is to:**

(a) create multiple objects

(b) control single object creation and reuse it ✅

(c) improve UI design

(d) handle testing

**90. Factory pattern is used to:**

(a) delete objects

(b) create objects without exposing creation logic to the user ✅

(c) test software

(d) manage database

**91. Factory pattern is best described as:**

(a) manual object creation

(b) a system that creates objects based on input requests ✅

(c) testing framework

(d) design tool for UI only

**92. Observer pattern works on the principle of:**

(a) single user system

(b) one-to-many notification system for updates ✅

(c) database storage

(d) object deletion

**93. In Observer pattern, when a change occurs:**

(a) nothing happens

(b) all registered observers are automatically notified ✅

(c) system stops working

(d) only one user is updated manually

**94. Strategy pattern is best described as:**

(a) fixed method for all tasks

(b) selecting the best algorithm or approach based on the situation ✅

(c) database structure

(d) hardware configuration

**95. One major benefit of design patterns is:**

(a) increasing code complexity

(b) reducing code complexity and improving reusability ✅

(c) removing programming need

(d) eliminating testing phase

**96. Debugging in software development refers to:**

(a) designing software

(b) finding and fixing errors in code ✅

(c) deploying software

(d) writing documentation

**97. Bugs in software are:**

(a) features

(b) errors or mistakes in the program behavior ✅

(c) design patterns

(d) testing tools

**98. The main goal of debugging is to:**

(a) increase complexity

(b) correct errors in code and improve functionality ✅

(c) design UI

(d) deploy software

**99. A debugger is used to:**

(a) write code automatically

(b) step through code and inspect variables to find errors ✅

(c) design databases

(d) create UI

**100. Print statements in debugging are used to:**

(a) delete errors

(b) display variable values for tracking program flow ✅

(c) compile code

(d) test hardware

**101. Code reviews are helpful because they:**

(a) slow development

(b) help identify errors through peer evaluation ✅

(c) remove testing phase

(d) automate coding

**102. Testing in software development is used to:**

(a) design system

(b) evaluate software to ensure it meets requirements ✅

(c) write code faster

(d) deploy software only

**103. The first level of testing is:**

(a) System Testing

(b) Integration Testing

(c) Unit Testing ✅

(d) Acceptance Testing

**104. Unit testing focuses on:**

(a) entire system

(b) individual components or modules of software ✅

(c) user interface only

(d) deployment process

**105. The main goal of unit testing is to:**

(a) test hardware

(b) verify each component works correctly individually ✅

(c) test full system

(d) deploy software

**106. Integration testing checks:**

(a) individual functions only

(b) interaction between different modules or components ✅

(c) user requirements only

(d) system deployment

**107. System testing evaluates:**

(a) single functions

(b) entire software system as a whole ✅

(c) only database

(d) only UI design

**108. System testing includes checking:**

(a) only coding style

(b) functionality, performance, and security of the whole system ✅

(c) only user feedback

(d) only debugging tools

**109. Acceptance testing is performed to:**

(a) write code

(b) check if software meets user/client requirements before release ✅

(c) design system

(d) fix bugs in code

**110. Acceptance testing is usually done by:**

(a) developers only

(b) end users or clients ✅

(c) hardware engineers

(d) database administrators

**111. Software development tools are used to:**

(a) design hardware systems

(b) assist in writing, testing, debugging, and managing code ✅

(c) replace programming languages

(d) remove requirement phase

**112. A language editor is also called:**

(a) compiler

(b) code editor ✅

(c) debugger

(d) interpreter

**113. Which of the following is a code editor?**

(a) GCC

(b) Python

(c) VS Code ✅

(d) GitHub

**114. Translators in programming are used to:**

(a) design UI

(b) convert high-level code into machine code ✅

(c) store data

(d) debug hardware

**115. An interpreter translates code:**

(a) all at once

(b) line by line during execution ✅

(c) only after compilation

(d) randomly

**116. A compiler translates code:**

(a) line by line

(b) all at once before execution ✅

(c) only errors

(d) UI design

**117. GDB is an example of a:**

(a) code editor

(b) debugger tool for C/C++ ✅

(c) compiler

(d) IDE

**118. Debuggers are mainly used to:**

(a) write programs

(b) find and fix errors in code ✅

(c) compile code

(d) design software

**119. An IDE is a tool that:**

(a) only compiles code

(b) combines editor, compiler, and debugger in one platform ✅

(c) only stores code

(d) only tests software

**120. A source code repository is used to:**

(a) design software

(b) store, manage, and track code changes using version control ✅

(c) compile programs

(d) debug hardware

## ❖ Important Short Questions:

**Q1: What is software development?**

**Answer:**

Software development is the systematic process of creating computer programs to solve specific problems or perform specific tasks. It includes designing, coding, testing, and maintaining software.

**Example:**

Developing a mobile banking app that allows users to send money and check balance.

**Q2: Why is software development important?**

**Answer:**

Software development is important because it helps create applications that solve real-world problems, improve efficiency, and automate tasks in different fields like education, business, and healthcare.

**Example:**

Online learning platforms like LMS systems help students study from home.

**Q3: Define SDLC.**

**Answer:**

SDLC (Software Development Life Cycle) is a structured framework used to develop software step by step, from planning to maintenance.

**Example:**

Developing a school management system using SDLC phases like planning, design, coding, and testing.

**Q4: What is the main purpose of SDLC?**

**Answer:**

The main purpose of SDLC is to develop high-quality software that meets user requirements within time and budget.

**Example:**

Creating a hospital system that works efficiently without errors.

**Q5: Name the stages of SDLC.**

**Answer:**

The stages of SDLC are:

- Requirement Gathering
- Design
- Coding/Development
- Testing
- Deployment
- Maintenance

**Q6: What is requirement gathering?**

**Answer:**

Requirement gathering is the process of collecting information about what users need from the software.

**Example:**

Asking students what features they want in an online learning app.

**Q7: Name any two techniques used in requirement gathering.**

**Answer:**

- Interviews
- Surveys

**Example:**

Taking feedback from users through questionnaires or face-to-face interviews.

**Q8: What is the purpose of interviews in requirement gathering?**

**Answer:**

Interviews are used to directly communicate with users to understand their needs, expectations, and problems in detail.

**Example:**

A developer asks teachers what features they need in a school software system.

**Q9: Define functional requirements.**

**Answer:**

Functional requirements describe what the system should do, including its functions and features.

**Example:**

A system should allow users to log in and register accounts.

**Q10: Define non-functional requirements.**

**Answer:**

Non-functional requirements describe how the system performs, including performance, security, and reliability.

**Example:**

The system should load within 2 seconds and be 99.9% available.

**Q11: Give one example of a functional requirement.**

**Answer:**

The system should allow users to search for books.

**Q12: Give one example of a non-functional requirement.**

**Answer:**

The system should handle 1000 users at the same time without slowing down.

Here are clear exam-ready answers with simple examples

**Q13: What is the purpose of the design phase in SDLC?**

**Answer:**

The purpose of the design phase is to plan how the software will work and look before coding starts. It defines system structure, components, and interactions.

**Example:**

Designing a login system showing how users enter username and password and how the system verifies them.

**Q14: What is a flowchart used for?**

**Answer:**

A flowchart is used to visually represent the steps and flow of a program or process in a diagram form.

**Example:**

A flowchart showing steps: Start ⟶ Enter data ⟶ Process ⟶ Display result.

**Q15: What is software architecture?**

**Answer:**

Software architecture is the overall structure of a software system, showing how different components interact with each other.

**Example:**

In a banking system, separate modules for login, transactions, and account management.

**Q16: What happens in the coding phase?**

**Answer:**

In the coding phase, programmers write the actual source code based on the design specifications.

**Example:**

Writing Python code to create a student registration form.

**Q17: What is the role of programmers in SDLC?**

**Answer:**

Programmers convert system design into working software by writing, testing, and fixing code.

**Example:**

A programmer develops a mobile app using Java or Python.

**Q18: What is software testing?**

**Answer:**

Software testing is the process of checking a software system to find errors and ensure it works correctly according to requirements.

**Example:**

Testing a login system to ensure correct and incorrect passwords are handled properly.

**Q19: What is unit testing?**

**Answer:**

Unit testing is testing individual components or modules of software separately.

**Example:**

Testing a function that calculates total marks in a student system.

**Q20: What is system testing?**

**Answer:**

System testing is testing the complete software system as a whole to ensure all parts work together properly.

**Example:**

Testing an entire school management system including login, attendance, and results.

**Q21: What is acceptance testing?**

**Answer:**

Acceptance testing is performed to check whether the software meets user requirements before final release.

**Example:**

A client tests a banking app before approving it for public use.

**Q22: Why is testing important?**

**Answer:**

Testing is important because it helps identify and fix bugs, ensures quality, and makes sure the software works correctly.

**Example:**

Testing prevents errors in an online shopping app so users can place orders without issues.

**Q23: What is debugging?**

**Answer:**

Debugging is the process of finding and fixing errors (bugs) in a software program.

**Example:**

Fixing a login error where the system does not accept correct passwords.

**Q24: Name two debugging tools.**

**Answer:**

GDB (GNU Debugger)

Visual Studio Debugger

**Example:**

Using GDB to step through C++ code and find errors line by line.

**Q25: What is the purpose of code review?**

**Answer:**

Code review is done to check code for errors, improve quality, and ensure best coding practices by having other developers review it.

**Example:**

A teammate checks your program and finds missing conditions in an if-statement.

**Q26: What is software deployment?**

**Answer:**

Software deployment is the process of installing and making software available for users to use.

**Example:**

Releasing a mobile app on Google Play Store.

**Q27: What is software maintenance?**

**Answer:**

Software maintenance is the process of updating, fixing, and improving software after it has been deployed.

**Example:**

Updating an app to fix bugs or add new features.

**Q28: Why is maintenance important?**

**Answer:**

Maintenance is important to fix errors, improve performance, and adapt software to new user needs or technology changes.

**Example:**

Updating a banking app to improve security and fix login issues.

**Q29: What is UML?**

**Answer:**

UML (Unified Modeling Language) is a standardized visual language used to design and represent software systems.

**Example:**

Using UML diagrams to show how users interact with an online shopping system.

**Q30: What is the purpose of a use case diagram?**

**Answer:**

A use case diagram is used to show how users (actors) interact with a system and what functions the system provides.

**Example:**

A student interacts with a system to register, view results, and submit assignments.

## ❖ Important Long Questions

**🌟 Q1: Explain Software Development Life Cycle (SDLC) and its stages in detail.**

**Answer:**

**Software Development Life Cycle (SDLC):**

SDLC (Software Development Life Cycle) is a structured and systematic process used to develop high-quality software. It defines a series of stages that guide software development from initial planning to final maintenance. The main purpose of SDLC is to produce reliable, efficient, and user-friendly software within time and budget constraints.

## ✦ Stages of SDLC:

### 1. Requirement Gathering

This is the first and most important stage. In this phase, developers collect and analyze user needs and expectations.

**Activities include:**

- Interviews with users
- Surveys and questionnaires
- Observation of current systems
- Document review

**Example:**

Understanding what features students need in a school management system.

### 2. Design Phase

In this phase, the system structure and working model are planned before coding starts.

**Activities include:**

- Creating flowcharts
- Designing system architecture
- Developing UI mockups
- Defining system components

**Example:**

Designing how login, dashboard, and result pages will work in an app.

### 3. Coding / Development Phase

In this phase, developers convert the design into actual program code using programming languages.

**Activities include:**

- Writing source code
- Implementing features
- Following design specifications

**Example:**

Writing Python or Java code for a student registration system.

### 4. Testing Phase

This phase is used to find and fix errors (bugs) in the software.

**Types of testing:**

- Unit Testing
- Integration Testing

- System Testing
- Acceptance Testing

**Example:**

Checking if the login system accepts correct passwords and rejects wrong ones.

### 5. Deployment Phase

In this phase, the software is installed and made available for users.

**Activities include:**

- Installing software on user systems or servers
- Configuring settings
- Real-world testing

**Example:**

Publishing a mobile app on the Google Play Store.

### 6. Maintenance Phase

This is the final and continuous phase after deployment.

**Activities include:**

- Fixing bugs
- Updating software
- Improving performance
- Adding new features

**Example:**

Updating an app to fix crashes and improve security.

**Summary:**

SDLC ensures that software is developed in a structured way. It improves quality, reduces risks, and helps in delivering efficient and reliable software systems.

### 🌟 Q2: Explain Functional and Non-Functional Requirements with examples.

**Answer:**

In software engineering, requirements define what a system should do and how it should perform. They are mainly divided into two types: Functional Requirements and Non-Functional Requirements.

### Functional Requirements

**Definition:**

Functional requirements describe the specific functions, features, and services that a system must perform. They define what the system does.

**Explanation:**

These requirements focus on system behavior and user interactions with the system. They explain the tasks the system should complete.

**Examples:**

- A system should allow users to create an account and log in.
- A library system should allow users to search books and borrow books.
- An online system should allow users to place orders and make payments.

## Non-Functional Requirements

**Definition:**

Non-functional requirements describe the quality, performance, and constraints of a system. They define how the system works.

**Explanation:**

These requirements focus on system performance, security, reliability, and efficiency rather than specific functions.

**Examples:**

- The system should support 1000 users at the same time.
- The system should be 99.9% available without failure.
- The system should load pages within 2 seconds.
- User data should be secure and protected using encryption.

## Difference (Conceptual Explanation):

Functional requirements focus on what the system does, while non-functional requirements focus on how well the system performs.

**Summary:**

Both functional and non-functional requirements are essential. Functional requirements define system features, while non-functional requirements ensure the system is fast, secure, and reliable.

### 🌟 Q3: Explain Waterfall Model with its phases, advantages, and limitations.

**Answer:**

## Waterfall Model:

The Waterfall Model is a linear and sequential software development model in which each phase must be completed before the next phase begins. It is one of the simplest and oldest software development methodologies. Once a phase is completed, the process does not go back to the previous phase easily, which makes it rigid in nature.

## Phases of Waterfall Model:

### 1. Requirement Phase

In this phase, all user requirements are collected and documented in detail. Developers communicate with stakeholders to understand what the system should do.

**Example:**

Collecting requirements for a school management system such as attendance, result management, and fee tracking.

## 2. Design Phase

In this phase, the system design is prepared based on requirements. It includes system architecture, database design, and user interface design.

**Example:**

Designing login pages, dashboards, and database structure for storing student records.

## 3. Implementation (Coding) Phase

In this phase, developers convert the design into actual code using programming languages.

**Example:**

Writing code in Python or Java for student registration and login system.

## 4. Testing Phase

In this phase, the software is tested to find and fix errors (bugs). It ensures that the system works correctly according to requirements.

**Example:**

Checking whether login works correctly with valid and invalid passwords.

## 5. Deployment Phase

In this phase, the software is installed and delivered to users for real-world use.

**Example:**

Installing a school management system in schools or hosting a web application online.

### 6. Maintenance Phase

In this phase, the software is updated and improved after deployment. Bugs are fixed and new features may be added.

**Example:**

Updating the system to fix errors or improve performance.

Advantages of Waterfall Model:

- It is simple and easy to understand.
- Each phase has clear structure and goals.
- Easy to manage due to its sequential nature.
- Suitable for small projects with fixed requirements.

### Limitations of Waterfall Model:

- It is not flexible; changes are difficult after a phase is completed.
- Not suitable for large or complex projects.
- Testing is done late, so errors may be found late.
- It assumes all requirements are known in advance, which may not always be true.

**Summary:**

The Waterfall Model is a traditional and structured approach to software development. It is best suited for small and simple projects where requirements are clear and fixed. However, it is less effective for complex or changing projects due to its rigid nature.

## 🌟 Q4: Explain Agile Methodology with its features, advantages, and limitations.

**Answer:**

**Agile Methodology:**

Agile Methodology is a flexible and iterative software development approach in which software is developed in small parts called iterations or sprints. Instead of completing the whole project at once, Agile focuses on delivering working software in small, continuous updates. It allows changes in requirements even during development.

### Key Features of Agile Methodology:

**1. Iterative Development**

The software is developed in small cycles called sprints. Each sprint produces a working part of the software.

**Example:**

In a mobile app, login feature is developed first, then chat feature in the next sprint.

### 2. Customer Feedback

Customers are involved throughout the development process and provide feedback regularly.

**Example:**

Users test an app version and suggest improvements.

### 3. Continuous Improvement

Software is continuously improved based on feedback and testing results.

### 4. Flexibility

Agile allows changes in requirements at any stage of development.

### 5. Collaboration

Developers, testers, and stakeholders work together closely.

## Practices in Agile:

- Continuous Integration: Code is merged and tested frequently.
- Test-Driven Development (TDD): Testing is done before or during coding.
- Pair Programming: Two developers work together on one system.

## Advantages of Agile Methodology:

- It is highly flexible and supports changes easily.
- Customers get early versions of software quickly.

- Improves software quality through continuous testing.
- Reduces risk of project failure.
- Better communication between team and customers.

### Limitations of Agile Methodology:

- Difficult to manage large projects with many teams.
- Requires constant customer involvement, which may not always be possible.
- Less predictable in terms of time and cost.
- Needs highly skilled team members for effective results.

**Summary:**

Agile Methodology is a modern and flexible approach to software development. It is best suited for complex and changing projects where requirements are not fixed. It improves customer satisfaction and software quality through continuous development and feedback.

**🌟 Q5: Explain UML Diagrams and their types with examples.**

**Answer:**

### UML (Unified Modeling Language):

UML (Unified Modeling Language) is a standard visual language used in software engineering to design, describe, and document software systems. It helps developers and stakeholders understand how a system works through diagrams instead of only code.

UML is very useful in planning, designing, and communication during software development.

## Purpose of UML:

- To visualize system design before coding
- To understand system structure and behavior
- To improve communication between developers and users
- To make software design easier and clearer

## Types of UML Diagrams:

### 1. Use Case Diagram

A use case diagram shows the interaction between users (actors) and the system. It represents system functionality from the user's point of view.

**Example:**

In a library system, a student can search books, borrow books, and return books.

### 2. Class Diagram

A class diagram shows the structure of a system by representing classes, attributes, and relationships between them.

**Example:**

A “Student” class may have attributes like name, roll number, and methods like register() and login().

### 3. Sequence Diagram

A sequence diagram shows how objects interact with each other in a specific order over time.

**Example:**

A user enters login details ⟶ system verifies data ⟶ system allows access.

### 4. Activity Diagram

An activity diagram shows the flow of activities or steps in a process.

**Example:**

Order process: Select product ⟶ Add to cart ⟶ Payment ⟶ Delivery.

## Advantages of UML:

- Makes system design easy to understand
- Helps in better planning before coding
- Improves communication among team members
- Reduces development errors
- Provides clear documentation of system

## Limitations of UML:

- Can be complex for large systems

- Requires learning and practice
- Time-consuming to create detailed diagrams

**Summary:**

UML diagrams are an important part of software development as they help in visualizing and designing systems before implementation. They improve understanding, reduce errors, and make software development more structured and efficient.

**🌟 Q6: Explain Testing types and their importance in software development.**

**Answer:**

**Software Testing:**

Software testing is the process of evaluating and checking a software system to ensure that it works correctly according to requirements and is free from errors (bugs). It helps improve the quality, reliability, and performance of software before it is delivered to users.

## Types of Software Testing:

### 1. Unit Testing

Unit testing is the first level of testing in which individual components or modules of software are tested separately.

**Example:**

Testing a login function to check whether it correctly accepts valid usernames and passwords.

**2. Integration Testing**

Integration testing checks whether different modules of software work together correctly after being combined.

**Example:**

Checking if the login module correctly connects with the user dashboard module.

**3. System Testing**

System testing evaluates the complete software system as a whole to ensure it meets all requirements.

**Example:**

Testing an entire school management system including attendance, results, and fee modules.

**4. Acceptance Testing**

Acceptance testing is performed by end users or clients to check whether the software meets their expectations before final release.

**Example:**

A client tests a banking application to ensure all features work as required.

## Importance of Software Testing:

### 1. Detects Errors Early

- Testing helps identify bugs and errors before software is delivered to users.

### 2. Improves Software Quality

- It ensures that the software is reliable, efficient, and works as expected.

### 3. Ensures User Satisfaction

- Proper testing ensures the software meets user requirements and expectations.

### 4. Reduces Maintenance Cost

- Finding and fixing bugs early reduces future maintenance costs.

### 5. Ensures Performance and Security

- Testing ensures the system performs well and protects user data.

### Summary:

Software testing is a critical part of software development. It ensures that the final product is error-free, reliable, and meets user requirements. Different levels of testing help improve quality at every stage of development.

**Note:**

This chapter is designed to provide a solid foundation of knowledge, with the goal of deepening understanding and encouraging further exploration of the subject. The content has been carefully selected to support effective learning and inspire students to engage with the topic more deeply.

**Author: Muhammad Asghar**

Purpose: To contribute to education by offering insightful, valuable content that enhances learning and understanding.

## Copyright & Usage Policy