Unit 1: Introduction to Software Development

Short Question Answers

Software Development

Q.1 What is software development?

Answer: Software development is the process of making computer programs that carry out particular tasks. It includes writing code, testing it, and correcting problems that appear. This process helps solve real-life problems and makes life easier with technology.

Q.2 Why is the Software Development Life Cycle (SDLC) important?

Answer: The SDLC gives software development an organised and step-by-step method. It helps make sure that the software meets user needs, stays within the budget, is delivered on time, and has good quality. By following the SDLC, teams can lower risks, plan better, and use resources more effectively.

Q.3 What is a framework in software development?

Answer: A framework provides ready-made tools and structures that help developers build software more quickly and efficiently. It contains pre-built parts, so developers do not have to start from the beginning. This saves time, keeps work consistent, and makes future maintenance easier.

Q.4 Give an example of a web development framework.

Answer: Django is a popular framework for building websites quickly. It has built-in features such as login systems, database management, and page templates. Django helps developers avoid writing everything from the beginning and speeds up the development process.

Q.5 What is the purpose of Requirement Gathering in SDLC?

Answer: Requirement gathering is the first phase of the SDLC. In this phase, developers find out what users need from the software. They use interviews, surveys, and observations to collect clear information. Correct requirements help the final product meet user expectations and prevent expensive rework.

Q.6 How many phases are there in the SDLC?

Answer: The Software Development Life Cycle (SDLC) has six phases:

1. Requirement Gathering
2. Design
3. Coding or Implementation
4. Testing
5. Deployment
6. Maintenance

Q.7 What are functional requirements?

Answer: Functional requirements explain what a system should do, such as registering a user or processing an order. They describe the exact actions and interactions between the system and its users. These requirements help define the main functions during development.

Q.8 What are non-functional requirements?

Answer: Non-functional requirements explain how well a system should work. They include speed, dependability, and security, as well as limits such as response time, availability, and data protection. These requirements help the system work properly in real-life conditions.

Q.9 How do functional and non-functional requirements differ?

Answer: Functional requirements focus on what the system does, while non-functional requirements focus on how well it does it. Functional requirements are related to user actions and system tasks. Non-functional requirements are related to performance, ease of use, and dependability.

Q.10 What happens during the Design phase of the SDLC?

Answer: During the design phase, developers prepare plans for the software by using diagrams and models. They plan the system structure, layout, and the way its parts will interact. This phase gives the team a clear plan before actual coding begins.

Q.11 Why is the design phase compared to creating house blueprints?

Answer: House blueprints show the structure and layout of a house before it is built. In the same way, the design phase shows the structure and layout of software. It helps all team members understand the system flow and functions, reducing confusion and coding errors.

Q.12 What occurs during the Coding phase?

Answer: During the coding phase, programmers write the actual code according to the design plan. They use programming languages such as Python, Java, or C# to create the features. This phase changes the design into a working software solution.

Q.13 What is the Testing phase in the SDLC?

Answer: The Testing phase checks the software for bugs, errors, and other problems. It includes tests for functions, performance, and compatibility. Testing confirms that the software meets user requirements and works smoothly.

Q.14 Why is testing important in software development?

Answer: Testing finds bugs early, when they are less expensive and easier to fix. It also makes sure that the software behaves as expected and meets user needs. Without proper testing, software may fail after release, causing a poor user experience and damage to the organisation's reputation.

Q.15 What is the Deployment phase?

Answer: Deployment is the phase in which software is installed and made available for users. It includes setting up, installing, and testing the software in a real environment. The goal is to make sure that it works correctly under live conditions.

Q.16 What is the Maintenance phase of the SDLC?

Answer: Maintenance means updating, improving, and fixing problems in the software after it has been released. It may include adding new features, repairing security weaknesses, and improving performance. This phase keeps the software useful over time.

Software Development Methodologies

Q.17 What are software development methodologies?

Answer: Methodologies are organised approaches that guide the way software is developed. Examples include Waterfall, Agile, and DevOps. They provide a plan for managing tasks, schedules, and teamwork.

Q.18 What role do software process models play in development?

Answer: Process models bring predictability, efficiency, and quality to software projects. They help teams follow a clear path, manage risks, and maintain standards during development. Models such as the SDLC help teams achieve consistent results in different projects.

Q.19 What is the Waterfall model?

Answer: The Waterfall model is a linear, step-by-step approach to software development. Each phase must be completed before the next phase begins. It works best for small projects with fixed requirements.

Q.20 What are the benefits of the Waterfall model?

Answer: The Waterfall model is simple to understand and easy to manage because of its step-by-step structure. Progress is easy to track, and documentation is detailed. It suits projects with clearly defined goals and few expected changes.

Q.21 What are the limitations of the Waterfall model?

Answer: After a phase is completed, going back to change it can be difficult and expensive. The model is not suitable for large or complex projects with changing requirements. It assumes that all requirements are known at the start, which is not often true in real-life projects.

Q.22 What is Agile Methodology?

Answer: Agile is a flexible and repeated approach to software development. It focuses on delivering small parts of the software quickly and adjusting to changes as they arise. Teams work in short periods called sprints and regularly involve users for feedback.

Q.23 What are key practices in Agile methodology?

Answer: Agile uses continuous integration, test-driven development, and pair programming. These practices improve code quality, encourage teamwork, and allow frequent updates based on user feedback.

Q.24 What are the benefits of Agile?

Answer: Agile allows quick adjustment to changing requirements. It increases customer satisfaction by involving customers throughout the development process. Delivering working software in short cycles supports continuous improvement.

Q.25 What are the limitations of Agile?

Answer: Managing large projects with many teams can be difficult. Agile also needs active involvement from stakeholders, which may not always be possible. Timelines and budgets can be harder to predict because the project scope may change.

Project Planning and Management

Q.26 What is comprehensive project planning?

Answer: Project planning includes defining goals, setting schedules, assigning responsibilities, and estimating costs. It makes sure that all parts of the project are considered before work begins. Good planning lowers risks and increases the chances of success.

Q.27 How do timelines contribute to successful project execution?

Answer: Timelines give the project structure and help the team track progress. They show when each task should be completed. Clear milestones make it easier to finish the project on time.

Q.28 Why is cost estimation important in project planning?

Answer: Cost estimation helps determine the money needed for a project. It considers factors such as team size, technology, and risk management. Accurate estimates prevent unexpected financial problems and help gain stakeholder approval.

Q.29 What is risk assessment in software projects?

Answer: Risk assessment means identifying possible threats to a project's success. It considers how likely each risk is and how serious its effect could be. Finding risks early helps teams prepare ways to reduce them.

Q.30 How does risk management contribute to project success?

Answer: By regularly checking and dealing with risks, teams can avoid major problems. Risk management supports smoother work, better use of resources, and delivery on time. It also builds confidence among stakeholders.

Q.31 What happens during the Execution phase?

Answer: During execution, the team writes code, designs interfaces, and builds the software. This work needs coordination, communication, and regular updates to keep everything on schedule. This is the stage where the software begins to take shape.

Q.32 What is Quality Assurance (QA)?

Answer: Quality Assurance makes sure that software meets the required standards and works correctly. It includes testing, code reviews, and collecting feedback. QA helps find and fix problems before the software reaches users.

Q.33 How does QA help ensure software works properly?

Answer: QA finds bugs early, improves code quality, and checks the user experience. It makes sure that the software works reliably in different conditions. Without QA, defects may remain hidden until after the software is released.

Graphical Representation

Q.34 What is graphical representation of software systems?

Answer: Graphical representation uses diagrams to show how a system is organised and how its parts interact. It makes complex ideas easier to understand and improves communication between developers and stakeholders. UML is a common tool used for this purpose.

Q.35 What is UML and why is it important?

Answer: UML, or Unified Modeling Language, is a standard way to show software design visually. It helps developers understand the behaviour and structure of a system. UML improves clarity, especially when people work in teams.

Q.36 How many types of UML diagrams are there?

Answer: There are four main types of UML diagrams: Use Case, Class, Sequence, and Activity diagrams. Each type has a different purpose and shows a different part of a system. These diagrams support both planning and documentation.

Q.37 What is a Use Case Diagram?

Answer: A Use Case Diagram shows how users interact with a system to achieve their goals. It identifies the actors and their interactions with the system. This diagram helps clarify functional requirements and system boundaries.

Q.38 What is a Class Diagram?

Answer: A Class Diagram shows the structure of a system by displaying its classes, attributes, and methods. It acts as a plan for object-oriented design. It helps developers organise code and understand the relationships between objects.

Q.39 What are Sequence Diagrams?

Answer: Sequence Diagrams show the order of interactions between objects over time. They help illustrate how messages move between system parts. These diagrams are useful for understanding system behaviour step by step.

Q.40 What are Activity Diagrams?

Answer: Activity Diagrams show the flow of activities or steps in a process. They are useful for showing business workflows or system logic. They can display decision points, parallel activities, and the overall flow of control.

Q.41 How is UML used in different stages of software development?

Answer: UML helps with planning by modelling requirements, designing system structure, and recording code details. During development, it helps developers understand system components. It also improves communication between technical and non-technical stakeholders.

Introduction to Design Patterns

Q.42 What are design patterns in software development?

Answer: Design patterns are ready-made solutions to common problems in software design. They work like templates that help developers write better-organised code that is easier to maintain.

Q.43 Why do developers use design patterns?

Answer: Developers use design patterns to solve common problems quickly without starting over. These patterns make code more flexible, reusable, and easier for everyone on the team to understand.

Q.44 Name some commonly used design patterns.

Answer: Common design patterns include Singleton, Factory, Observer, and Strategy. Singleton makes sure that only one instance of a class exists. Factory manages the creation of objects. Observer informs dependent objects about changes. Strategy allows different algorithms to be exchanged.

Q.45 What is the Singleton Pattern?

Answer: The Singleton pattern makes sure that only one object of a class is created in the entire program. For example, it can create one database connection that can be reused whenever necessary.

Q.46 What is the Factory Pattern?

Answer: The Factory pattern creates objects without showing the details of how they are made. Like a real factory that produces different products, this pattern provides the correct object based on user input while hiding complex details behind a simple interface.

Q.47 Explain the Observer Pattern with an example.

Answer: In the Observer pattern, one object informs other objects when something changes. For example, subscribers may receive automatic updates from a news publisher whenever new content is posted.

Q.48 What is the Strategy Pattern used for?

Answer: The Strategy pattern allows different methods to be used for solving similar tasks. For example, a shopping cart can use different payment methods, such as a credit card or PayPal, depending on the user's choice.

Software Debugging and Testing

Q.49 What is debugging?

Answer: Debugging is the process of finding and correcting errors in code. It involves studying how a program behaves and finding the main cause of a problem. Effective debugging helps the software run as planned.

Q.50 Why is debugging important in software development?

Answer: Bugs can cause crashes, incorrect results, or security weaknesses. Debugging solves these problems early, which saves time and money. It helps improve software dependability and user satisfaction.

Q.51 What is testing in software development?

Answer: Testing means checking whether software works correctly and performs the tasks it is meant to perform. It helps find mistakes before users begin using the software.

Q.52 Why is testing important?

Answer: Testing makes sure that the software does not have serious errors and works properly. It helps prevent problems such as crashes or incorrect results after the software is released.

Q.53 Define types of testing.

Answer: The main types of testing are Unit, Integration, System, and Acceptance Testing. Unit testing checks individual components. Integration testing checks how modules work together. System testing checks the complete application. Acceptance testing checks whether the software meets user needs.

Q.54 What is unit testing?

Answer: Unit testing checks small parts of software, such as one function at a time. It makes sure that each part works correctly on its own.

Q.55 What is integration testing?

Answer: Integration testing checks how different parts of software work together. It finds problems that may occur when separate parts are connected.

Q.56 What is system testing?

Answer: System testing checks the entire software as one complete system. It makes sure that all parts work together properly and that the software meets user needs.

Q.57 What is acceptance testing?

Answer: Acceptance testing is done by real users or clients to decide whether the software is ready to use. It checks whether the software meets their expectations.

Q.58 Which type of testing comes first?

Answer: Unit testing is done first. It is followed by integration testing, system testing, and finally acceptance testing.

Q.59 Who usually does acceptance testing?

Answer: Acceptance testing is usually done by end users or clients, not by developers. They check whether the software works well for them before they accept it.

Software Development Tools

Q.60 What are software development tools?

Answer: Software development tools are programs that help developers write, test, and manage code. They make software creation easier by providing features such as error checking and code organisation.

Q.61 What is a code editor? Give two examples.

Answer: A code editor is a tool used to write and edit programming code. It makes coding easier with features such as colour coding and auto-completion. Two examples are Notepad++ and Visual Studio Code (VS Code).

Q.62 What do translators do in software development?

Answer: Translators change code written by humans into code that computers can understand. There are two main types: compilers, which translate the complete code at once, and interpreters, which translate the code one line at a time.

Q.63 What is a debugger? Why is it useful?

Answer: A debugger is a tool that helps find and correct errors in code. It lets developers see what is happening inside a program while it is running. This makes problem-solving faster and easier.

Q.64 What is an IDE? Name two popular IDEs.

Answer: An IDE, or Integrated Development Environment, is a complete package that includes tools such as a code editor, compiler, and debugger in one place. It makes coding more efficient. Two examples are Visual Studio and PyCharm.

Q.65 Define Source Code Repositories.

Answer: Source code repositories store and manage different versions of code. Platforms such as GitHub and Bitbucket allow developers to work together, track changes, and keep a history of the code. They are important for version control and teamwork.