
Part 1: Introduction to Software Development

1. Q: What is the fundamental definition of software development?
A: Software development is a systematic process that transforms user needs into software products. It involves a series of stages, from initial analysis through design, coding, testing, and deployment.

2. Q: According to the Student Learning Outcomes, what key terminology should a student be able to describe?
A: Students should be able to describe key terminology including Software Development Life Cycle (SDLC), debugging, testing, and design patterns.

3. Q: Why is understanding the software development process considered crucial?
A: It is crucial for creating reliable, maintainable, and scalable software solutions.

4. Q: What are some real-world examples of how software makes our lives easier, as mentioned in the text?
A: Software allows us to communicate with friends through social media, manage money with banking apps, and makes learning fun with educational games.

5. Q: What is the Software Development Life Cycle (SDLC) and what is its primary purpose?
A: The SDLC is a framework that defines the processes used by organizations to build an application from its initial conception to its deployment and maintenance. Its primary purpose is to deliver high-quality software that meets or exceeds customer expectations, is completed within time and cost estimates, and works efficiently.

6. Q: What is a "framework" in software engineering, and what are its benefits?
A: A framework is a standardized and reusable set of concepts, practices, and tools that provides a structured foundation for developing software. Its benefits include promoting efficiency, consistency, and code reusability, which improves the overall quality and maintainability of software.

7. Q: Explain the analogy used for a framework like Django.
A: Using a framework like Django is compared to using a pre-designed blueprint to build a house instead of designing everything from scratch. Django comes with ready-made features like user login, database management, and page templates.

Part 2: The Stages of the SDLC

8. Q: What is the primary goal of the "Requirement Gathering" phase?
A: The goal is to understand and collect what the software needs to achieve. This involves talking to stakeholders to find out their needs and expectations.

9. Q: What are the three key activities listed for the Requirement Gathering phase?
A:

1. **Interviews and Surveys:** Asking questions and collecting feedback from potential users.
2. **Observations:** Watching how users interact with current systems to identify problems and opportunities.
3. **Document Review:** Looking at existing documents like reports and user manuals to gather information.

10. Q: What analogy is used in the "DID YOU KNOW?" box for requirement gathering?
A: It is compared to planning a big event, like a wedding. Just as you need to know the preferences of the bride and groom, developers need to know what users want from the software.

11. Q: What is the core difference between Functional and Non-Functional Requirements?
A:

- **Functional Requirements** describe *what* the system should do (its specific behaviors, tasks, and functions).
- **Non-Functional Requirements** define *how* the system should perform (its quality attributes like performance, usability, and security).

12. Q: List the five examples of Functional Requirements for a Library Management System.
A:

1. **User Registration:** The system should allow users to register and create an account.
2. **Book Borrowing:** The system should enable users to search for and borrow books.
3. **Book Return:** The system should allow users to return borrowed books.
4. **Inventory Management:** Librarians should be able to add, update, and remove books.
5. **Notification:** The system should send notifications about due dates and overdue books.

13. Q: List the five examples of Non-Functional Requirements for a Library Management System.
A:

1. **Performance:** The system should handle up to 1000 simultaneous users without performance degradation.
2. **Usability:** The user interface should be intuitive and easy to navigate.
3. **Reliability:** The system should be available 99.9% of the time.

4. **Security:** User data should be encrypted, and access controlled through secure authentication.
5. **Scalability:** The system should be able to scale to accommodate an increasing number of users and books.

14. Q: What happens during the Design phase, and what are its key activities?
A: In the design phase, the team plans how the software will look and work, like creating a blueprint. Key activities include:

- **Create Diagrams:** Show how different parts will connect and work together.
- **Develop Models:** Create mockups of the user interface.
- **Plan the Architecture:** Decide the overall structure of the software.
- **Specify Requirements:** Clearly define what each part of the software needs to do.

15. Q: What is the Development (Implementation) phase, and what analogy is used to describe it?
A: This is the phase where programmers write the actual code. It's compared to following a recipe, where the design specifications are the recipe and coding is the process of mixing and baking the ingredients.

16. Q: What is the Testing phase, and what three types of testing does it include?
A: Testing is the process of checking the software to identify bugs, errors, or issues. It includes:

1. **Functionality Testing:** Ensuring all features work according to specifications.
2. **Performance Testing:** Checking if the software performs well under different conditions (e.g., high traffic).
3. **Compatibility Testing:** Making sure the software works well on various devices and operating systems.

17. Q: What is the Deployment phase, and what three steps does it often involve?
A: Deployment is the process of making the software available for users. It often involves:

1. **Installation:** Installing the software on the user's system or server.
2. **Configuration:** Adjusting the software to fit specific user needs (e.g., setting preferences).
3. **Testing in the Real World:** Ensuring it works correctly in its intended environment.

18. Q: What is the Maintenance phase, and what analogy is used in the "Tidbits" box?
A: The final phase involves ongoing maintenance and updates to ensure the software continues to function correctly and adapt to changes. It is compared to "taking care of a plant" that needs regular watering to stay healthy.

Part 3: Software Development Methodologies

19. Q: Why are software process models like Waterfall and Agile important?
A: They are important because they provide:

- **Predictability:** Teams can predict outcomes and manage risks more effectively.
- **Efficiency:** They streamline the development process, reducing wasted effort.
- **Quality:** They ensure quality assurance practices are integrated throughout the lifecycle.

20. Q: Describe the Waterfall Model and its main phases.
A: The Waterfall Model is a linear, sequential approach where each phase must be completed before the next begins. The main phases are: Requirements, Design, Implementation, Testing, and Deployment.

21. Q: What are the main benefits and limitations of the Waterfall Model?
A:

- **Benefits:** Simple and easy to understand, sequential process makes it easy to track, and suitable for small projects with fixed requirements.
- **Limitations:** Inflexible (costly to make changes), not ideal for complex projects, and assumes all requirements are known upfront, which is risky.

22. Q: Describe the Agile Methodology and how it differs from Waterfall.
A: Agile is a flexible and adaptive approach that focuses on delivering small, functional parts of the software quickly in short cycles (iterations or sprints). Unlike the rigid, linear Waterfall model, Agile is designed to adapt to changes and gather feedback throughout the process.

23. Q: What are three specific practices included in Agile methods?
A:

1. **Continuous Integration:** Regularly merging code changes into a central repository to detect and fix issues early.
2. **Test-Driven Development:** Writing tests before writing the code to ensure the software works as expected.
3. **Pair Programming:** Two developers working together at one workstation, with one writing code and the other reviewing it.

24. Q: What are the key benefits and limitations of the Agile Methodology?
A:

- **Benefits:** High flexibility to adapt to new needs, and improved customer satisfaction due to frequent updates.
- **Limitations:** Can be difficult to manage on large projects (scaling challenges), requires active stakeholder involvement, and can be less predictable in its final timeline and scope.

25. Q: What is Scrum, and what are its key components (Roles, Events, Artifacts)?
A: Scrum is a popular Agile framework for managing complex projects. Its key components are:

- **Roles:** Product Owner (defines the work), Scrum Master (facilitates the process), and Development Team (delivers the product).

- **Events:** Sprints (time-boxed iterations), Sprint Planning, Daily Standups, Sprint Reviews, and Sprint Retrospectives.
- **Artifacts:** Product Backlog (prioritized list of features), Sprint Backlog (tasks for a sprint), and Increment (the working product).

26. Q: For what types of projects is Scrum suitable and not suitable?
A:

- **Suitable:** Projects with evolving requirements and frequent updates, like mobile apps and web development.
- **Not Suitable:** Projects with well-defined requirements and little expected change, such as embedded systems or safety-critical software (e.g., medical or aerospace).

27. Q: What are the 7 Key Principles of Lean Software Development?
A:

1. Eliminate Waste
2. Amplify Learning
3. Decide as Late as Possible
4. Deliver as Fast as Possible
5. Empower the Team
6. Build Integrity In
7. See the Whole

28. Q: What is DevOps, and what is its primary goal?
A: DevOps is a set of practices that combines Software Development (Dev) and IT Operations (Ops). Its primary goal is to shorten the development lifecycle, improve software quality, and provide continuous delivery with high reliability.

29. Q: What interesting fact is mentioned about the origin of the term "Waterfall"?
A: The term was first introduced by Dr. Winston W. Royce in 1970, but he did not advocate for its use without iteration.

Part 4: Project Planning, Management, and Quality

30. Q: What are the five phases of a Project Management Plan shown in the diagram?
A: 1. Initiation, 2. Planning, 3. Execution, 4. Performance Monitoring, 5. Project Closure.

31. Q: According to the "DID YOU KNOW?" box, what is the significance of Microsoft's market capitalization?
A: As of 2023, Microsoft's market capitalization exceeded $2 trillion, highlighting the immense economic importance of software development.

32. Q: What are the five key factors involved in estimating a software project's cost?
A:

1. **Development Team:** Number, expertise, and rates of developers.
2. **Technology Stack:** Choice of programming languages and tools.
3. **Project Duration:** Longer projects cost more.
4. **Risk Management:** Including contingency funds for unforeseen issues.
5. **Quality Assurance:** Costs associated with testing and bug fixing.

33. Q: Based on the detailed example, what is the total estimated cost for the online shopping app, and what does it include?
A: The total estimated cost is **PKR 19,644,000**. This includes costs for the Project Manager, Frontend Developers, Backend Developers, UI/UX Designer, QA Testers, Operational Costs (cloud services, tools), and a 10% Contingency Fund.

34. Q: What advice does the "Tidbits" box give to non-programmers who want to start a software project?
A: They can post their idea on freelancing platforms where developers can bid on it, or they can seek out investors to fund the idea.

35. Q: What are the four steps in Risk Assessment and Management?
A:

1. **Identify Risks:** List all potential risks (technical, operational, external).
2. **Analyze Risks:** Evaluate the likelihood and impact of each risk.
3. **Develop Mitigation Strategies:** Create a plan to reduce the impact of significant risks.
4. **Monitor and Review:** Continuously monitor for new risks and review existing ones.

36. Q: What is Quality Assurance (QA)?
A: Quality Assurance ensures that a project meets set standards and works correctly. It involves methods like testing, code reviews, and getting feedback from stakeholders.

Part 5: Graphical Representation and UML

37. Q: What is Unified Modeling Language (UML), and who created it?
A: UML is a standardized way to visualize the design of a software system. It was created by Grady Booch, James Rumbaugh, and Ivar Jacobson, who are often called the "Three Amigos" of UML.

38. Q: What is a Use Case Diagram and its purpose?
A: A Use Case Diagram visually shows the different ways users (actors) interact with a system. Its purpose is to capture functional requirements, understand user interactions, and aid in planning and testing.

39. Q: According to the Class Activity, who are the actors and what are the use cases for an online shopping platform?
A:

- **Actors:** Customer, Administrator, Delivery Personnel.

- **Use Cases:** Browse Products, Add Items to Cart, Make Purchase, Manage Product Listings, Process Orders, Handle Customer Inquiries, Update Delivery Status.

40. Q: Explain the "organizing a room" analogy for a Class Diagram.
A: A **Room** is the main structure. A **Box** inside is like a class. The **items** in the box (toys, books) are its attributes. The **actions** it can perform (open, close) are its methods. Specialized boxes like 'ToyBox' or 'BookBox' are like distinct instances (objects) of the general 'Box' class.

41. Q: What do Sequence and Activity diagrams illustrate?
A:

- **Sequence Diagrams** show how objects interact with each other in a particular sequence over time.
- **Activity Diagrams** illustrate the flow of activities or steps in a process, modeling the logic of complex operations.

Part 6: Design Patterns, Debugging, and Testing

42. Q: What are Design Patterns, and who popularized them?
A: Design patterns are proven, reusable solutions to common problems in software development. They were popularized by the book 'Design Patterns: Elements of Reusable Object-Oriented Software' by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, also known as the "Gang of Four".

43. Q: Explain the analogies for the Singleton, Factory, Observer, and Strategy patterns.
A:

- **Singleton:** Like having just one key to a special room that everyone must share.
- **Factory:** Like a special workshop that creates different products for you when you tell it what you need, without you needing to know how it's made.
- **Observer:** Like a group of people interested in updates from one source, who are automatically notified when something important happens.
- **Strategy:** Like having a toolbox full of different tools for different jobs, allowing you to pick the right one for the problem at hand.

44. Q: What is debugging, and what is the famous origin of the term?
A: Debugging is the process of finding and fixing bugs (errors) in software. The term was named after an incident in 1947 when a real bug (a moth) was found in a computer, causing it to malfunction.

45. Q: Describe the four main levels of testing in their typical hierarchy.
A:

1. **Unit Testing:** Testing individual components (e.g., a single function) in isolation.
2. **Integration Testing:** Testing how different components work together when combined.

3. **System Testing:** Testing the entire software system as a whole to evaluate its overall functionality, performance, and compliance.
4. **Acceptance Testing:** The final level of testing, often performed by end-users or clients, to ensure the software meets their expectations. It is sometimes called User Acceptance Testing (UAT).

46. Q: What is the difference between an Interpreter and a Compiler?
A:

- **Interpreter:** Translates and executes code one line at a time (e.g., Python interpreter).
- **Compiler:** Translates the entire code into machine code at once before execution (e.g., GCC for C/C++).

47. Q: What is an Integrated Development Environment (IDE)? Provide three examples.
A: An IDE is a comprehensive software suite that provides all the tools a developer needs in one place (editor, compiler, debugger, etc.). Examples include **Eclipse**, **Visual Studio**, and **PyCharm**.

48. Q: What are source code repositories, and what are two popular examples?
A: They are platforms where developers can store, manage, and track changes to their code, helping with version control and collaboration. Popular examples are **GitHub** and **Bitbucket**.