



**1st YEAR COMPUTER SCIENCE
(SOLVED ADDITIONAL/CR QUESTIONS)**

UNIT #01 — Introduction to software development

Prepared by: Mahmood Khan | MS Cyber Security
[Codymentor Software House Sahiwal](#) | 03186879182

Additional/Constructed Response Short questions

1. What is software development?

Answer: Software development is the process of creating computer programs that perform specific tasks. It includes writing code, testing it, and fixing any problems that come up.

Example: It allows us to chat with friends on social media, and learn through fun educational games.

2. What is software development life cycle (SDLC)?

Answer: The Software Development Life Cycle (SDLC) is a step-by-step process used to develop software from its initial idea to its final deployment and maintenance. Its main goal is to create high-quality software that meets customer needs, stays within budget, finishes on time, and works efficiently.

3. What is framework in software development?

Answer: A framework in software development is like a toolbox that provides ready-to-use tools, rules, and structures to help developers build software faster and more efficiently. It includes pre-made components so developers don't have to start from scratch. This saves time, ensures consistency, and makes the software easier to maintain.

Example: Imagine you want to create a website. Instead of writing all the code from scratch, you can use a framework like Django (for websites).

Django comes with ready-made features like user login, database management, and page templates.

4. What are software development methodologies?

Answer: Software development methodologies are structured approaches to software development that guide the planning, creation, and management of software projects. They help ensure that the development process is systematic, efficient, and produces high-quality software.

5. Explain the importance of software development models in software development.

Answer: Software process models are abstract representations of the processes involved in the SDLC. They provide a framework for planning, structuring, and controlling the development of software systems. The

importance of software process models lies in their ability to provide:

1. **Predictability:** By following a defined process, teams can predict outcomes and manage risks more effectively.
2. **Efficiency:** Structured methodologies streamline the development process, reducing wasted effort.
3. **Quality:** Adhering to a process model ensures that quality assurance practices are integrated throughout the SDLC

6. What happens during the execution phase of software development?

Answer: This is the phase where actual development work happens. The team writes codes, creates designs, and builds the software based on the project plan, it requires team work, coordination and regular updates to stay on track

7. What is quality assurance, and how does it help ensure a software project work properly?

Answer: Quality assurance ensures that a project meets set standards and works correctly. It involves methods such as testing, reviewing code, getting feedback from stakeholders, and regularly checking the project's progress.

8. What is the purpose of graphical representation of software system?

Answer: Graphical representation of software systems involves using visual diagrams to depict various aspects of a software system's structure and behavior. This approach helps in simplifying complex systems, making it easier for developers and stakeholders to understand, communicate, and manage the system.

9. How is UML diagram used in different stages in software development to enhance understanding and communication.

Answer: UML (unified modeling language) play a vital role in various stages software development by improving clarity, collaboration and planning. There are some key application:

- 1.Planning
- 2.Development
- 3.Communication

10.What are the application of design pattern in software design?

Answer: Design patterns are widely used in software development to solve common problems and create robust and maintainable code.

They help in:

- 1.Reducing code complexity by providing a clear structure.
 - 2.Enhancing code reusability by using proven solutions.
 - 3.Improving communication among developers by providing a common vocabulary.
- Improving communication among developers by providing a common understand.

11.Why are software debugging and testing critical stages in the development process?

Answer: Software debugging and testing are critical stages in the software development process that ensure the quality and reliability of a software product.

Effective debugging and testing help developers to confirm that the software meets the required specifications, functions as intended, and is free of critical errors.

12.What is debugging? And why is important in software development?

Answer: Debugging is the process of finding and fixing bugs or errors in a software.

Software debugging and testing are essential parts of software development. They ensure that the software works correctly and meets the user's requirements.

Effective debugging and testing help developers to confirm that the software meets the required specifications, functions as intended, and is free of critical errors.

13.What are software development tools?

Answer: Software development tools are essential for creating, testing, and maintaining software applications.

These tools help developers write code, find and fix errors, and manage software projects effectively.

14.What are online and offline computing platform?

Answer: These platforms provide environments where developers can write, run, and test their code.

Online Platforms: Cloud-based platforms accessible via the internet (e.g., Repl.it, Gitpod).

15. Define source code repositories.

Answer: Source code repositories are platforms where developers can store, manage, and track changes to their code. Repositories help in version control, allowing multiple developers to work on the same project without conflicts. Repositories keep track of code changes and maintain a history of all modifications.

Examples:

GitHub: Popular platform for open-source projects

Bitbucket: Used for both private and public repositories

16. Why is the design phase often compared to creating blueprints for a house in software development?

Ans: The design phase serves as the foundation for software development, much like blueprints guide the construction of a house. It involves visualizing and planning the structure, layout, and functionality of the software before actual development begins. This ensures clarity, organization, and alignment with user requirements, just as blueprints ensure a house is built according to plan and meets the needs of its residents.

17. What makes software companies like Microsoft so valuable?

Answer: Software companies are valued highly due to scalable products, recurring revenues, innovation, and strong eco-systems. Their success underscores software's essential role in driving global digital transformation and economic growth.

18. Why is acceptance testing also called User Acceptance Testing (UAT)?

Answer: Because it is typically performed by the end-users to ensure the software meets their requirements.

Additional/Constructed Response Long questions

1. Discuss the various stages involved in software development life cycle (SDLC).

Ans: Stages (Phases or Steps) involved in SDLC

The SDLC consists of several stages (phases or steps), each with specific goals and activities. These stages are interconnected and ensure that the software is developed in an organized and efficient manner.

1) Requirement Gathering

This is the first and most critical phase of the SDLC. Its primary objective is to understand what the software needs to achieve and gather detailed information about user expectations. This involves interacting with stakeholders, such as end-users, clients, and business analysts, to identify the software's purpose and functionality,

Key Activities in this Phase:

Interviews and Surveys: Conducting interviews and surveys to collect feedback from potential users.

Observations: Performing observations to analyze how users interact with existing systems.

Document Review: Reviewing existing documents, such as reports, user manuals, and technical specifications, to gather additional insights.

Functional and Non-Functional Requirements

Requirements are generally categorized into two types, functional and non-functional requirements.

Functional Requirements

Functional requirements describe the specific behaviors or functions of a system. These requirements outline what the system should do and include tasks, services, and functionalities that the system must perform.

They define the interactions between the system and its users or other systems.

Example: Some functional requirements for a Library Management System are:

User Registration: The system should allow users (students and faculty) to register and create an account.

Book Borrowing: The system should enable users to search for books and borrow them.

Inventory Management: Librarians should be able to add, update, and remove books

Non-Functional Requirements

Non-functional requirements define the quality attributes, performance criteria, and constraints of the system. These requirements specify how the system performs a function rather than what the system should do.

Example: Some non-functional requirements for a Library Management System are:

Performance: The system should handle up to 1000 simultaneous users without performance decline.

Reliability: The system should be available 99.9% of the time, ensuring high availability and minimal downtime.

Security: User data should be encrypted, and access should be controlled through secure authentication mechanisms.

2) Design

Once the requirements are gathered, the next step is the Design phase, where developers create a blueprint for the software. This phase focuses on planning how the software will look, function, and interact with users and other systems.

Key Activities in this Phase

Create Diagrams: Creating diagrams (e.g., flowcharts, UML diagrams) to visualize the software's structure and workflows.

Develop Models: Developing models and mockups of the user interface to show how the software will appear to users and how it will look like.

Plan the Architecture: Planning the architecture of the software, including how different components will interact and communicate.

Specify Requirements: Specifying detailed design requirements to ensure all features are planned out accordingly.

TIDBITS

3) Coding/Development

In the Development phase, programmers write the actual code for the software based on the design specifications. This is where the software begins to take shape. Programmer will write code in a programming language (e.g., Python, Java, C#)

4) Testing

Testing phase is a critical step where the software is strictly checked for bugs, errors, and issues. The goal is to ensure that the software works as expected and meets all requirements.

Key Activities in this Phase

Functionality Testing: Ensuring all functions of software work according to expectations.

Performance Testing: Ensuring software performance how software behaves under various conditions such as high (heavy) data traffic load.

Compatibility Testing: Running compatibility testing to ensure the software works across different devices, operating systems, and browsers.

5) Deployment

Once the software has been thoroughly tested and all issues have been resolved, it moves to the Deployment phase. In this phase, the software is installed and made available for users to access and use.

Key Activities in this Phase

Installation: The software is installed on the user's system or server. This may involve running an installation program that copies files and sets up necessary configurations.

Configuration: The software is adjusted to fit the specific needs of the user or organization. It can include setting up user preferences, network settings, and database connections.

Testing in the Real World: After installation, the software is tested in its real-world environment to ensure it works correctly with other systems and meets user needs.

6) Maintenance

The final phase of the SDLC is Maintenance, where the software is continuously monitored, updated, and improved. This phase ensures that the software remains functional and adapts to changes in user needs or technology.

Key Activities in this Phase

Fixing bugs or errors that arise after deployment.

Adding new features or enhancements based on user feedback.

Updating the software to address security vulnerabilities or compatibility issues with new technologies.

2. What is the water fall model and what are its benefits and limitations (drawback)?

Answer: Waterfall Model

The Waterfall model is one of the earliest and most straightforward software development methodologies. It follows a linear and sequential approach, where each phase of the project must be completed before moving on to the next.

Main Phases of Waterfall Model

Think of it as a waterfall flowing from one stage to the next without going back.

The main phases of the Waterfall model are:

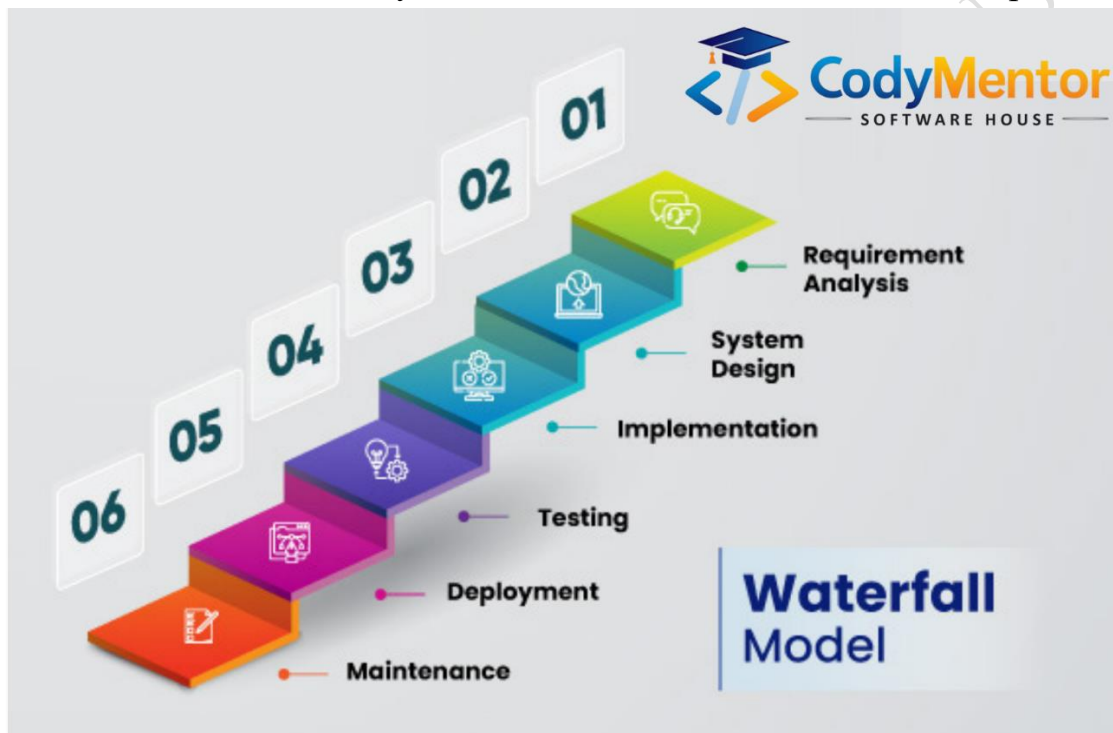
- 1) **Requirements:** Gather and document what the software needs to achieve.
- 2) **Design:** Plan how the software will be built, including its architecture and user interface. design specifications.
- 3) **Coding:** Write the actual code to create the software based on the design.
- 4) **Testing:** Check and fix any problems or bugs (errors) in the software.
- 5) **Deployment:** Release the software for users to access and use.
- 6) **Maintenance:** Make updates and fix issues that arise after the software is in use.

Benefits

- 1) **Simple and Easy to Understand:** The Waterfall Model is straightforward, with clear, distinct phases that are easy to follow.
- 2) **Sequential Process:** Each phase is completed one at a time, making it easier to manage and track progress.
- 3) **Suitable for Small Projects:** Works well for projects with clear, fixed requirements where changes are unlikely.

Limitations

- 1) **Inflexibility:** Once a phase is completed, going back to make changes is difficult and costly.
- 2) **Not Ideal for Complex Projects:** For large and complex designs, this model can be challenging to use effectively.
- 3) **Risk and Uncertainty:** The model assumes that all requirements are known from the start, which can be risky if new needs or issues arise later in the process.



3. What is agile methodology? explain its key practices, benefits and limitations.

Agile Methodology

Agile Methodology is a flexible and adaptive approach to software development. It focuses on delivering small, functional parts of software quickly and adapting to changes as the project progresses.

Unlike the Waterfall Model, which follows a strict sequence of steps, Agile uses iterative development through short cycles called iterations or sprints. These cycles allow teams to deliver working software quickly, receive feedback early, and continuously improve the product.

Key Practices of Agile

Continuous Integration: Code changes are added regularly to one place. It helps identify and fix problems early.

Test-Driven Development (TDD): Tests are written before the actual code. It ensures that the software meets requirements and works correctly.

Pair Programming: Two developers work together at one workstation. One developer writes the code while the other reviews it in real time. It improves code quality and knowledge sharing.

Benefits of Agile

High Flexibility: Agile allows changes in requirements even after development has started. It makes it easier to adapt to new requirements and feedback.

Improved Customer Satisfaction: Working software is delivered frequently.

Customers can review progress and provide regular feedback. This helps create a product that better meets customer expectations.

Limitations of Agile

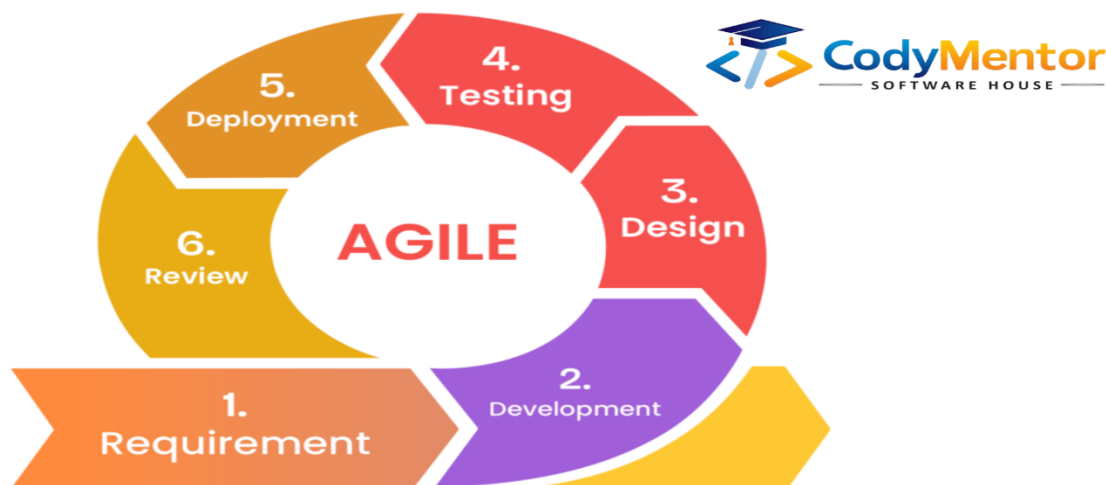
Scaling Challenges: Managing large projects with multiple teams can be difficult. It requires effective coordination and communication.

Stakeholder Involvement. Agile requires active participation from stakeholders.

Problems may occur if stakeholders are unavailable or not fully engaged.

Less Predictable: Agile projects evolve through continuous feedback and changes. Therefore, the exact timeline and scope of the final product can be harder to predict.

than traditional models such as Waterfall.



4. Compare and contrast waterfall and agile methodology.

Ans:

Feature	Waterfall Model	Agile Methodology
1.Model Type	Linear and sequential	Iterative and incremental
2.Phases	Requirements → Design → Implementation → Testing → Maintenance	Planning → Development → Testing → Review through iterative sprints
3.Flexibility to Change	Not flexible; changes are difficult once development has started	Highly flexible; changes can be made at any stage
4.Customer Involvement	Limited; mainly involved at the beginning and end	Continuous involvement throughout the project
5.Testing Phase	Testing occurs after development is complete	Continuous testing throughout development
6.Delivery Timeframe	Single final delivery	Frequent deliveries in small, functional increments

Feature	Waterfall Model	Agile Methodology
7.Focus	Fulfilling the predefined plan	Delivering working software and adapting to feedback
8.Risk Handling	Higher risk due to the rigid process	Lower risk due to continuous feedback and adaptability
9.Documentation	Extensive documentation and detailed planning	Prioritizes working software over extensive documentation
10.Best Suited For	Fixed-scope projects, such as government and regulatory projects	Dynamic-scope projects, such as startups and evolving applications

5. What are the important step involved in project planning and management for a software project?

Software Project Planning

Planning a software project is like planning a trip—you need to know your destination (project goals), the time it will take (timelines), and the budget required (cost estimation). Comprehensive project planning ensures that all details are considered before starting, reducing risks and ensuring smooth execution.

The 5 Phases of Project Management

- Initiation
- Planning
- Execution
- Performance Monitoring
- Project Closure
- Comprehensive Project Planning

Comprehensive project planning involves thinking through all aspects of the project before implementation. This includes:

1. Understanding Requirements: Clearly defining what needs to be done to meet user needs.

2. Assigning Roles: Identifying who will perform each task, such as:

- Developers
- Designers
- Testers

3. Defining Processes: Outlining how tasks will be completed, including the tools and methodologies to be used.

Importance:

This step is critical because it sets the foundation for the entire project, ensuring clarity and alignment among team members.

Setting Project Timelines

Setting project timelines involves deciding how long each phase or task will take. Timelines help keep the project on track and ensure timely delivery.

Example: Website Development Timeline

- Design Phase: 2 weeks
- Content Writing: 1 week
- Testing Phase: 1 week

Importance of Timelines:

Timelines provide structure and accountability, allowing teams to monitor progress and address delays promptly.

Estimating Project Costs

Estimating the cost of a software project is a critical step in project planning and management. It involves predicting the total expenses required to complete the project successfully.

Accurate cost estimation helps with:

- Budgeting
- Resource allocation
- Setting realistic expectations

Key Factors in Cost Estimation

1. Development Team

The cost depends on:

- Number of developers
- Developers' expertise
- Hourly rates

2. Technology Stack

The choice of technology, programming languages, and tools can affect the cost. Some technologies require more resources or specialized knowledge.

3. Project Duration

Longer projects generally require higher costs due to prolonged resource engagement and potential changes in scope.

4. Risk Management

Identifying potential risks and their mitigation strategies can add to the overall cost. Contingency funds are often included to address unforeseen issues.

5. Quality Assurance

Costs associated with:

- Testing
- Bug fixing
- Ensuring software meets quality standards

Risk Assessment and Management

Risk assessment and management are crucial aspects of any software project. They involve identifying potential risks that could impact the project's success, analyzing the likelihood and impact of these risks, and developing strategies to manage them.

Steps in Risk Assessment and Management

1. Identify Risks

List all potential risks that could affect the project. These may include:

- Technical risks
- Operational risks
- External risks

2. Analyze Risks

Evaluate:

- The likelihood of each risk occurring. The potential impact of each risk on the project.

3. Develop Mitigation Strategies

For each significant risk, develop a plan to:

- Reduce the likelihood of the risk. Minimize its impact.

4. Monitor and Review

Continuously monitor the project for new risks and review existing risks.

Adjust risk management strategies whenever necessary.

Execution

Execution is the phase where the actual development work takes place.

During this phase, the team:

- Writes code.
- Creates designs.
- Builds the software according to the project plan.

Successful execution requires teamwork, coordination, and regular updates to keep the project on track.

Quality Assurance (QA)

Quality Assurance ensures that a project meets the defined quality standards and works correctly.

Quality Assurance Activities

- Software Testing
- Stakeholder Feedback
- Code Review
- Progress Monitoring

Main Goal:

The main goal of Quality Assurance is to ensure that the software is reliable, functional, and meets the required quality standards.

6. What are the steps involved in Risk assessment and management in a software project?

Risk Assessment and Management

Risk assessment and management are crucial aspects of any software project. They involve identifying potential risks that could impact the project's success, analyzing the likelihood and impact of these risks, and developing strategies to manage them.

Steps in Risk Assessment and Management

1. Identify Risks

List all potential risks that could affect the project. These may include:

- Technical risks
- Operational risks
- **2. Analyze Risks**
- External risks

Evaluate:

- The likelihood of each risk occurring. The potential impact of each risk on the project.

3. Develop Mitigation Strategies

For each significant risk, develop a plan to:

- Reduce the likelihood of the risk. Minimize its impact.

4. Monitor and Review

Continuously monitor the project for new risks and review existing risks.

Adjust risk management strategies whenever necessary.

7. What is UML? Describe use case diagram in detail. Also explain how to identify use cases.

Answer: Introduction to UML

Unified Modeling Language (UML) is a standardized way to visualize the design of a software system. It helps developers understand how a system works and communicates.

Use Case Diagrams: Use case diagrams provide a visual representation of the system's functionality from the user's perspective, helping to identify the requirements and the interactions between the users and the system.

Definition and Purpose: A use case is a description of a set of interactions between a user (actor) and a system to achieve a specific goal. Use cases are identified based on the functionalities that the system must support to meet the user's needs. Each use case represents a complete workflow from the user's perspective, detailing the steps involved in a particular task.

Use Case Diagrams are used for several purposes:

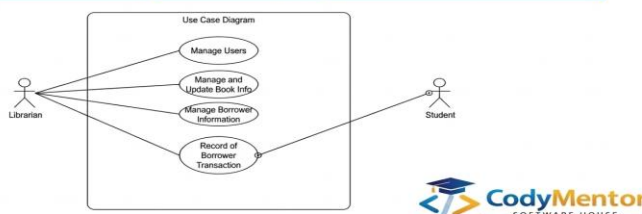
- 1. Capturing Functional Requirements:** They help in identifying and documenting the functional requirements of the system.
- 2. Understanding User Interactions:** They illustrate how different users will interact with the system.
- 3. Planning and Testing:** They aid in planning the development process and in designing test cases for validating system functionalities.

Identifying Use Cases:

The process of identifying use cases involves several steps:

- 1. Identify Actors:** Determine the different types of users who will interact with the system. Actors can be human users or other systems.
- 2. Define Goals:** For each actor, identify their goals or what they need to accomplish using the system.
- 3. Outline Interactions:** Describe the interactions between the actors and the system to achieve these goals. Each interaction that results in a significant outcome is a potential use case.
- 4. Validate Use Cases:** Review the identified use cases with stakeholders to ensure they accurately capture the required functionalities and interactions.

Use Case Diagram for Library Management System



8. What is class diagram? Explain with the help of an example.

Answer:

A class diagram is like a map that shows how things are organized in a system.

Example:

In the example of organizing your room as shown in Figure below:

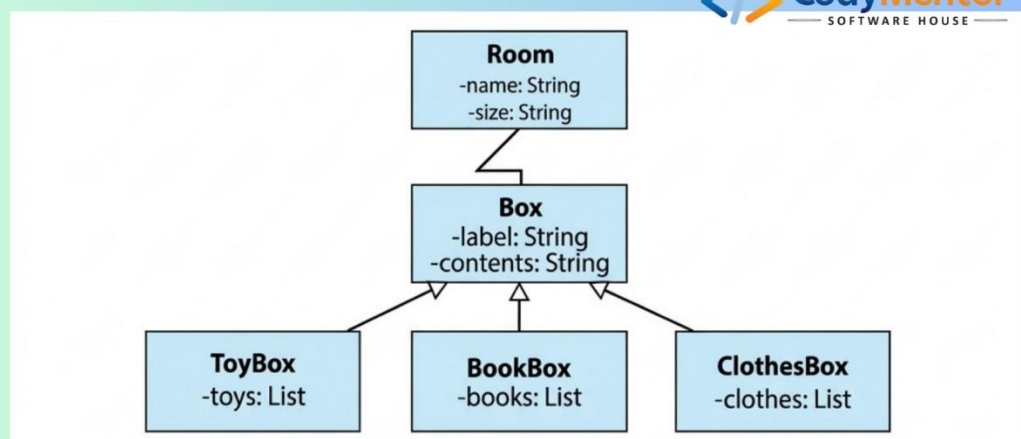
Room: Represents the overall space encompassing all other elements, analogous to the main structure in a class diagram.

Box: Serves as a container within the room, akin to a class in a diagram.

Attributes: Each box contains specific items, such as a 'Toy Box' holding toys or a 'Book Box' containing books.

Methods: Boxes can perform actions like 'open' or 'close,' similar to methods in a class diagram that define what the box can do.

Specific Boxes: Examples of specialized boxes include a 'Toy Box' for toys, a 'Book Box' for books, and a 'Clothes Box' for clothes, representing distinct instances of the general 'Box' class.



9. What are sequence diagram? Explain with an example?

Answer:

Sequence Diagram

A Sequence Diagram shows how objects in a system interact with each other in a particular sequence. It helps in understanding the flow of messages between objects over time.

Example: User Interacting with Boxes

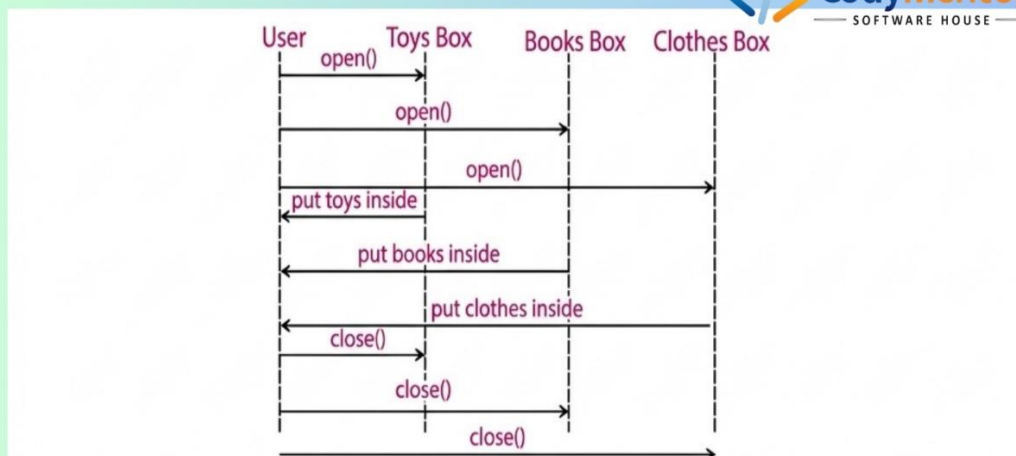
Interactions

Open Box: The user opens each box.

Put Items in Box: The user puts the respective items into the box, such as:

- Toys
- Books
- Clothes
- Close Box

The user closes each box.



10.Explain the concept of design patterns in software development. Discuss some commonly used design patterns.

Introduction to Design Patterns

Design pattern are common solutions to problem in software development, they act like templates to help make coding easier, faster and more consistent.

Commonly Used Design Patterns

Below are some of the most widely recognized design patterns:

1. Singleton Pattern

The Singleton Design Pattern is a way to make sure that a specific object or resource is created only once in a program and reused whenever needed.

2. Factory Pattern

The Factory Design Pattern is like having a special workshop that knows how to create different products, but you don't need to worry about the details of how those products are made. Instead, you just tell the factory what you need, and it gives you the finished product.

3. Observer Pattern

The Observer Design Pattern is like having a group of people who are interested in getting updates from one particular source. Whenever something important happens, the source automatically notifies all the interested people. It's a way to keep things in sync without everyone constantly checking for updates. I people. It's

4. Strategy Pattern

The Strategy Design Pattern is like having a toolbox full of different tools, each designed for a specific job. When you face a problem, you can pick the right tool from the box based on the task at hand.

11.What is testing? Explain its type.

Answer: Testing is the process of checking if the software works as it should and meets the requirements. It starts with testing small parts and moves step-by-step to testing the whole system, including getting feedback from users.

There are four types of testing:

1) Unit Testing: Unit testing is the first level of testing, where individual components or modules of the software are tested in isolation. Each unit is a small, testable part of the software, such as a function or method. The primary goal of unit testing is to verify that each component works correctly according to its design and performs as expected.

2) Integration Testing: After unit testing Integration Testing is performed to evaluate the

interaction between different components or modules. While unit testing focuses on isolated units, integration testing ensures that these units work together correctly when combined. This type of testing checks for interface errors, data flow between modules and other integration-related issues.

3) System Testing: System Testing is a higher level of testing where the entire software system is tested as a whole. At this stage, the software is treated as a complete entity, and testers evaluate its overall functionality, performance, security, and compliance with specified requirements.

4) Acceptance Testing: Acceptance Testing is conducted to determine whether the software is ready for release. It is often performed by the end-users or clients to ensure that the software meets their expectations and requirements.