Module-1

1. What is SDLC

Ans: SDLC stands for Software Development Life Cycle. It is a structured approach to software development that defines the processes and stages involved in building software from conception to deployment and maintenance.

1. What is agile methodology?

Ans: Agile methodology is an approach to software development that emphasizes flexibility, collaboration, and incremental delivery of software in short iterations. Unlike traditional waterfall methods, where development is linear and each phase is completed before the next begins, agile methodologies promote adaptive planning, evolutionary development, early delivery, and continuous improvement.

Key principles and characteristics of agile methodology include:

1. **Iterative and Incremental Development**: Agile projects are divided into small iterations or sprints (usually 1-4 weeks long), where cross-functional teams work on small features or parts of the software. Each iteration results in a potentially shippable product increment.
2. **Collaboration and Communication**: Agile teams emphasize daily communication and collaboration among all team members, including developers, testers, designers, and product owners. This ensures that everyone is aligned on goals, progress, and challenges.
3. **Adaptive Planning**: Agile projects embrace changing requirements and priorities throughout the development process. Instead of a fixed plan, agile teams use adaptive planning techniques to respond to feedback and adjust their priorities and tasks accordingly.
4. **Continuous Improvement**: Agile methodologies promote a culture of continuous improvement through regular retrospectives. Team members reflect on their processes, identify areas for improvement, and implement changes to enhance productivity and quality.
5. **Customer Involvement**: Agile methodologies emphasize frequent and early delivery of working software to customers. This allows for rapid feedback, enabling teams to validate assumptions, gather requirements, and make adjustments based on real-world usage.
6. **Flexible and Responds to Change**: Agile methodologies are designed to be flexible and responsive to changes in requirements, market conditions, or technology advancements. This agility helps teams deliver value quickly and effectively.
7. What is SRS

Ans: SRS stands for Software Requirements Specification. It is a detailed document that describes the functional and non-functional requirements of a software system. The SRS serves as a contract between the development team and the customer or stakeholders, outlining what the software should do, how it should behave, and the constraints under which it must operate.

1. What is oops

Ans: OOPS stands for Object-Oriented Programming System or Object-Oriented Programming (OOP). It is a programming paradigm that organizes software design around objects and data, rather than actions and logic.

1. Write Basic Concepts of oops

Ans: Object-Oriented Programming (OOP) is built upon several fundamental concepts that help in organizing and designing software systems. Here are the basic concepts of OOP:

1. **Class**: A class is a blueprint or template that defines the attributes (data members) and behaviours (methods) that objects of the class should have. It serves as a blueprint from which objects are created.
2. **Object**: An object is an instance of a class. It represents a specific entity or instance of the class, possessing its own state (attributes) and behaviour (methods).
3. **Encapsulation**: Encapsulation is the bundling of data (attributes) and methods (functions) that operate on the data into a single unit (class). It hides the internal state of an object from the outside world and only exposes a controlled interface for interacting with the object.
4. **Inheritance**: Inheritance is a mechanism where a class (subclass or derived class) inherits attributes and behaviours from another class (superclass or base class). It promotes code reuse and allows the creation of a hierarchy of classes.
5. **Polymorphism**: Polymorphism means the ability of different objects to respond to the same message or method call in different ways. It allows objects of different classes to be treated as objects of a common superclass through method overriding and method overloading.
6. **Abstraction**: Abstraction refers to the process of simplifying complex systems by modelling classes appropriate to the problem, and working at the most relevant level of inheritance to create new classes

6. What is object

Ans: An object is an instance of a class, which serves as a blueprint or template defining its structure and behaviour.

7. What is class

Ans: A class is a blueprint or template for creating objects. It defines a data structure that encapsulates data (attributes) and behaviours (methods or functions) that operate on the data. Essentially, a class serves as a blueprint from which objects are instantiated.

8. What is encapsulation

Ans: Encapsulation is a fundamental principle in object-oriented programming (OOP) that describes the bundling of data (attributes) and methods (functions that operate on the data) into a single unit called a class. The class serves as a blueprint for creating objects (instances), which are instances of the class with their own unique data.

9. What is inheritance

Ans: Inheritance is another fundamental concept in object-oriented programming (OOP) where a new class (derived class or subclass) is created based on an existing class (base class or superclass). The derived class inherits the attributes and methods of the base class, allowing it to reuse the code of the base class and extend its functionality.

10. What is polymorphism

Ans: Polymorphism, in the context of object-oriented programming (OOP), refers to the ability of different objects to be treated as instances of a common superclass. It allows objects of different classes to be processed uniformly if they exhibit a certain behavior or share a common interface.

11. What is RDBMS

Ans: RDBMS stands for **Relational Database Management System**. It is a type of database management system (DBMS) that organizes data into tables, which are composed of rows and columns.

12. What is SQL

Ans: SQL (Structured Query Language) is a standardized programming language used for managing and manipulating relational databases. It provides a set of commands or statements for performing tasks such as querying data, inserting, updating, and deleting records, creating and modifying database schema (tables and indexes), controlling access permissions, and ensuring data integrity.

13. Write SQL Commands

Ans:

* **SELECT**: Retrieves data from a database.
* **INSERT**: Inserts new records into a table.
* **UPDATE**: Modifies existing records in a table.
* **DELETE**: Deletes records from a table.
* **CREATE**: Creates new database objects such as tables, views, indexes, etc.
* **ALTER TABLE**: Modifies an existing table (e.g., adding a new column).
* **DROP TABLE**: Deletes an entire table and its data.
* **REVOKE**: Revokes previously granted privileges from a user or role.

14. Draw Usecase on Online book shopping

15. Draw Usecase on online bill payment system (Paytm)

16. Write SDLC phases with basic introduction

Ans: Here are the phases of the SDLC:

1) **Requirements Gathering and Analysis**:

* **Introduction**: This phase involves gathering and documenting requirements from stakeholders, users, and other relevant parties. It aims to define the functionality, performance, and constraints of the software.
* **Activities**: Conduct interviews, workshops, and surveys to gather requirements. Analyse and prioritize requirements to create a detailed requirements specification document.

2) **System Design**:

* **Introduction**: In this phase, the system architecture and design are planned based on the requirements gathered. It defines how the software will be structured and how components will interact with each other.
* **Activities**: Design high-level and low-level system architecture. Specify database schema, user interface design, and system interfaces. Create design documents that serve as blueprints for developers.

3) **Implementation (Coding)**:

* **Introduction**: This phase involves the actual coding of the software based on the design specifications. Developers write code using programming languages and integrate different components to build the software.
* **Activities**: Write code according to design specifications. Perform unit testing to identify and fix defects at the module level. Collaborate with testers to ensure code meets functional and performance requirements.

4) **Testing**:

* **Introduction**: Testing verifies that the software meets the specified requirements and functions correctly. It ensures quality, reliability, and usability of the software before deployment.
* **Activities**: Plan and execute different types of testing such as unit testing, integration testing, system testing, and acceptance testing. Identify and report defects for correction. Ensure software meets quality standards.

5) **Deployment (Implementation)**:

* **Introduction**: Deployment involves releasing the software to end-users or customers for use. It includes installation, configuration, and deployment activities to make the software operational.
* **Activities**: Prepare deployment plan and release documentation. Install software on target environments and configure settings. Conduct user training and provide support during initial use.

6) **Maintenance**:

* **Introduction**: After deployment, the software enters the maintenance phase where it is monitored, updated, and enhanced to meet changing user needs and fix issues discovered post-deployment.
* **Activities**: Monitor software performance and user feedback. Address defects and issues through patches and updates. Implement changes and enhancements based on user requirements.

17. Explain Phases of the waterfall model

Ans: Here are the phases of the waterfall model:

1. **Requirements Gathering and Analysis:**
   * In this phase, the project team gathers and documents all requirements for the software. This includes functional requirements (what the software should do) and non-functional requirements (performance, security, etc.).
   * Requirements are typically documented in a Software Requirements Specification (SRS) document, which serves as a contract between the development team and the stakeholders.
2. **System Design:**
   * Once the requirements are gathered, the system design phase begins. In this phase, the overall architecture of the software system is designed. This includes defining the structure, components, interfaces, and data for the system.
   * The design phase produces a Design Specification document that outlines how the system will be implemented.
3. **Implementation (Coding):**
   * In the implementation phase, the actual code for the software is written based on the design specifications. Developers follow coding standards, guidelines, and best practices.
   * This phase focuses on converting the design into executable code.
4. **Testing:**
   * After the code is developed, it undergoes testing. The testing phase involves verification and validation activities to identify defects and ensure the software works as expected.
   * Testing includes unit testing (testing individual units or components of the software), integration testing (testing how units work together), system testing (testing the entire system as a whole), and acceptance testing (ensuring the software meets user requirements).
5. **Deployment (Installation):**
   * Once testing is complete and the software is approved, it is deployed to the production environment. This may involve installation, configuration, and setting up the software for end-users.
   * Deployment includes activities to make the software operational and accessible to its intended users.
6. **Maintenance:**
   * After deployment, the software enters the maintenance phase. This phase involves making modifications, enhancements, and updates to the software to address issues that were not discovered during testing or to adapt to changes in the operating environment.
   * Maintenance ensures the software remains useful and effective over time.

18. Write phases of spiral model

Ans: Here are the phases of the Spiral Model:

1. **Planning:**
   * The initial phase involves establishing the project's objectives, identifying constraints, defining deliverables, and planning out resources, schedules, and risks.
   * This phase aims to establish a clear understanding of what the software product is expected to accomplish and how it will be developed.
2. **Risk Analysis:**
   * In this phase, potential risks are identified and analyzed. Risks could include technical risks, such as integration issues or performance bottlenecks, as well as business risks, such as changes in market conditions or regulatory requirements.
   * Risk analysis helps prioritize risks based on their likelihood and potential impact on the project's success.
3. **Engineering (Development and Prototyping):**
   * This phase involves the actual development of the software. It may begin with prototyping and iterative refinement of the software design.
   * Prototypes are used to explore and validate critical aspects of the software, gather feedback from stakeholders, and refine requirements.
4. **Evaluation (Testing and Review):**
   * After each iteration or prototype, the software is evaluated. This evaluation includes testing the software against the requirements, conducting reviews, and gathering feedback from stakeholders.
   * The evaluation phase helps identify and resolve defects early in the development process, ensuring that the software meets quality standards.
5. **Iteration (Cycle Repeat):**
   * Depending on the evaluation results, the project may enter into another iteration of planning, risk analysis, engineering, and evaluation.
   * Each iteration around the spiral represents a cycle of refining the software product based on feedback and addressing identified risks and issues.

19. Write agile manifesto principles

Ans: Here are the Agile Manifesto principles:

1. Customer satisfaction through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development.
3. Deliver working software frequently, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity—the art of maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

20. What is join?

Ans: In the context of databases, a "join" is an operation that combines rows from two or more tables based on a related column between them. The purpose of a join is to retrieve data that spans across multiple tables in a relational database, allowing for more complex queries and data analysis.

21. Write type of joins.

Ans: **Types of Joins:**

* + **Inner Join:** Returns only the rows that have matching values in both tables based on the join condition.
  + **Left (Outer) Join:** Returns all rows from the left table (the first table mentioned in the query) and the matched rows from the right table.
  + **Right (Outer) Join:** Returns all rows from the right table and the matched rows from the left table.
  + **Full (Outer) Join:** Returns all rows when there is a match in either the left or right table. This join type includes rows from both tables where there is no match as well.

22. Explain working methodology of agile model and also write pros and cons.

### Ans: Working Methodology of Agile Model:

1. **Iterative Development:**
   * Agile projects are divided into small iterations or sprints, typically lasting from one to four weeks. Each iteration involves cross-functional teams working on a subset of features or user stories.
   * At the beginning of each iteration, there is a planning meeting where the team decides which features to prioritize based on business value and complexity.
2. **Collaboration:**
   * Agile emphasizes frequent communication and collaboration between developers, product owners, customers, and other stakeholders.
   * Daily stand-up meetings (or scrum meetings) are held to discuss progress, challenges, and plans for the day, ensuring everyone is aligned and aware of the project’s status.
3. **Incremental Delivery:**
   * Working software is delivered in small, incremental releases after each iteration. This allows stakeholders to see tangible progress and provide feedback early in the development process.
   * Feedback from customers and stakeholders is incorporated into subsequent iterations, ensuring the product meets their evolving needs.
4. **Adaptability:**
   * Agile methodologies are designed to be adaptive and responsive to change. Changes in requirements, priorities, or market conditions can be accommodated at the beginning of each iteration.
   * This flexibility helps teams deliver a product that better meets customer expectations and market demands.
5. **Continuous Improvement:**
   * Agile encourages continuous improvement through regular retrospectives at the end of each iteration. Team members reflect on what went well, what could be improved, and make adjustments for future iterations.
   * This iterative feedback loop promotes learning and enhances team efficiency and effectiveness over time.

**Pros of Agile Model:**

1. **Customer Satisfaction:** Agile prioritizes delivering working software early and frequently, ensuring that customers see value quickly and can provide feedback.
2. **Flexibility and Adaptability:** Agile allows for changes in requirements and priorities, which is crucial in dynamic business environments.
3. **Faster Time-to-Market:** Incremental releases enable faster delivery of useful features and improvements, potentially reducing time-to-market compared to traditional models.
4. **Improved Quality:** Regular testing and continuous integration in Agile help identify and address issues early, resulting in higher quality software.
5. **Enhanced Collaboration:** Agile promotes collaboration between developers, stakeholders, and customers, leading to better alignment and shared understanding.

**Cons of Agile Model:**

1. **Requires Experienced Team:** Agile relies heavily on self-organizing teams and constant communication, which may be challenging for inexperienced teams.
2. **Emphasis on Documentation:** Agile values working software over comprehensive documentation, which can lead to challenges in maintaining documentation consistency.
3. **Scope Creep:** Without strict control, frequent changes in requirements can lead to scope creep, impacting project timelines and budgets.
4. **Dependency on Customer Availability:** Agile requires active involvement and availability of customers and stakeholders for feedback, which may not always be feasible.
5. **Not Suitable for Large Projects:** Agile may face scalability issues when applied to large, complex projects that require extensive upfront planning and coordination.

23. Draw usecase on Online shopping product using COD.

24. Draw usecase on Online shopping product using payment gateway.