**Software Testing Assignment**

**Module–1(Fundamental)**

Submitted By: Kunal Joshi

1. What is SDLC

* SDLC Stands for software Development Life Cycle. It is a process followed by software development teams to design, develop, test and deploy high-quality software.

1. What is software testing?

* Software testing is the process of evaluating a software application system to ensure that it meets the specified requirements and functionsas expected. The primary goal of software testing is to detect defects or errors in the software and ensure that it delivers the intended functionality and meets user expectations

1. What is agile methodology?

* Agile methodology is an iterative approach to software development that emphasizes collaboration, flexibility, and customer satisfaction. It is a set of principles and values that guide the development process to be more adaptive to changes and customer feedback

1. What is SRS

* SRS stands for Software Requirements Specification. It is a document that outlines the functional and non-functional requirements of a software system or application. The SRS document serves as a communication tool between the software development team and stakeholders, including clients, users, and project managers.

1. What is oops

* OOPs stands for Object-Oriented Programming. It is a programming paradigm that emphasizes the use of objects and classes to organize and structure code. In OOPs, objects are instances of classes that encapsulate data and behavior. A class defines the attributes and methods of the objects it creates.

1. **Write Basic Concepts of oops**

* Encapsulation: Encapsulation is the concept of bundling data and functions (methods) that operate on that data into a single entity, known as a class. The class acts as a blueprint for creating objects, and the data and methods are encapsulated within the class, meaning they cannot be accessed or modified from outside the class without the class's permission. This ensures data integrity and reduces the likelihood of errors and unintended consequences.
* Inheritance: Inheritance is the concept of creating new classes based on existing classes.The new class (subclass or derived class) inherits the attributes and behaviors of the existing class (superclass or base class) and can add new attributes and behaviors or override the existing ones.
* Polymorphism: Polymorphism is the concept of using a single interface to represent multiple types of objects.
* Abstraction: Abstraction is the concept of hiding the implementation details of a class from its users and exposing only the essential features and behaviors through a simplified interface.

1. **What is object**

In object-oriented programming, an object is an instance of a class that contains both data (attributes or properties) and functions (methods) that operate on that data. An object is a self-contained entity that can be manipulated and passed around within a program.

1. What is class?

* In object-oriented programming, a class is a blueprint or a template that defines the attributes and behaviors of a specific type of object. It is an abstract data type that encapsulates data and functions that operate on that data.

1. What is encapsulation

* Encapsulation is a fundamental concept of object-oriented programming that refers to the practice of bundling data and methods that operate on that data into a single unit, known as a class. Encapsulation enables us to hide the implementation details of a class from its users and restrict access to the data and methods, so they can be used only in ways that are intended and safe.

1. What is inheritance

* Inheritance is a fundamental concept of object-oriented programming that allows us to create new classes by deriving them from existing classes. Inheritance enables us to reuse code from an existing class and extend or modify its behavior without having to rewrite the entire class from scratch.

1. What is polymorphism

* Polymorphism is a fundamental concept in object-oriented programming (OOP) that allows objects of different classes to be treated as objects of a common superclass. It refers to the ability of an object to take on many forms or have multiple types. In OOP, classes are used to define objects that encapsulate data and behavior. Polymorphism allows you to write code that can work with objects of different classes, as long as they are related through inheritance or implement the same interface. This enables you to write more flexible and reusable code.

1. Draw Usecase on Online book shopping



1. Write SDLC phases with basic introduction

Requirements Gathering and Analysis:

In this phase, the project team gathers and analyzes the requirements of the software system. It involves understanding the needs of stakeholders, documenting user requirements, and defining the scope of the project. The goal is to identify what the software should accomplish and how it should function.

System Design:

In the system design phase, the software architecture and overall system structure are planned. This involves defining the system's components, modules, and their interactions. The design phase focuses on creating a blueprint for the software solution that meets the identified requirements.

Implementation or Coding:

The implementation phase involves translating the design specifications into actual code. Software developers write the program code, following coding standards and best practices. This phase also includes tasks such as unit testing, debugging, and ensuring code quality.

Testing:

Testing is a crucial phase where the software is systematically checked for defects and validated against the requirements. Different types of testing, such as unit testing, integration testing, system testing, and user acceptance testing, are performed to identify and fix any issues or discrepancies.

Deployment:

Once the software passes the testing phase, it is deployed or released for use. This involves preparing the software for production environments, configuring servers, installing the software, and conducting final checks to ensure a smooth transition from development to live operation.

Maintenance:

The maintenance phase involves post-deployment activities, such as addressing user feedback, fixing bugs, adding new features, and making enhancements to the software. It aims to ensure the continued functionality, performance, and security of the software over its lifecycle.

1. Explain Phases of the waterfall model

Requirements Gathering and Analysis:

In this phase, the requirements for the software system are gathered from stakeholders. The project team analyzes and documents the requirements, defining what the software should do and how it should function. The goal is to have a clear understanding of the project scope and deliverables.

System Design:

In the system design phase, the overall system architecture and design are planned based on the gathered requirements. The design includes defining the software components, their interactions, and the data flow. This phase focuses on creating a blueprint for the software solution.

Implementation:

The implementation phase involves the actual coding and development of the software. Programmers write the code based on the design specifications. This phase follows the "code once" principle, where the entire software is developed in one iteration, without incremental updates.

Testing:

Once the implementation is complete, the software undergoes testing. This phase includes various types of testing, such as unit testing, integration testing, system testing, and user acceptance testing. The objective is to identify and fix any defects or issues in the software.

Deployment:

After the software passes the testing phase, it is deployed or released for use. The deployment phase involves preparing the software for production environments, configuring servers, installing the software, and ensuring a smooth transition from development to live operation.

Maintenance:

The maintenance phase involves post-deployment activities, such as addressing user feedback, fixing bugs, making enhancements, and providing ongoing support. This phase ensures that the software remains functional, secure, and up-to-date throughout its lifecycle.

1. Write phases of spiral model

Identification of Objectives:

In this phase, the project objectives, requirements, and constraints are identified. This involves gathering information about the project scope, goals, and any specific requirements or risks that need to be addressed.

Risk Analysis:

The risk analysis phase involves a detailed assessment of potential risks and uncertainties associated with the project. Risks can include technical, operational, schedule, or budget-related challenges. Risk analysis helps in understanding and mitigating these risks by developing strategies and contingency plans.

Development and Planning:

In this phase, a development plan is formulated based on the requirements and risks identified. The plan includes defining the project's deliverables, development milestones, resource allocation, and scheduling. It also includes selecting the appropriate development methodology and tools.

Engineering:

The engineering phase involves the actual development and implementation of the software. It includes activities such as coding, testing, integration, and documentation. This phase follows the chosen development methodology, whether it's waterfall, iterative, or agile.

Evaluation and Review:

During this phase, the developed software is evaluated, reviewed, and validated against the project objectives and requirements. This includes conducting inspections, walkthroughs, and assessments to ensure the software meets the specified quality standards.

Planning the Next Iteration:

Based on the evaluation and review results, the next iteration is planned. This phase involves refining and updating the development plan, addressing any identified issues, and determining the scope and objectives for the next iteration.

1. Write agile manifesto principles

**Individuals and Interactions over Processes and Tools:**

This principle emphasizes the importance of valuing individuals and their interactions within a development team. It highlights the need for effective communication, collaboration, and teamwork over relying solely on processes and tools. Agile methodologies encourage face-to-face conversations and close collaboration between team members to foster a productive and efficient work environment.

**Working Software over Comprehensive Documentation:**

This principle emphasizes the importance of delivering a working software product as the primary measure of progress. While documentation is necessary, Agile values working software as the ultimate goal. It promotes frequent iterations and incremental development, allowing for early and continuous delivery of value to customers.

**Customer Collaboration over Contract Negotiation:**

This principle emphasizes the significance of involving customers and stakeholders throughout the development process. Agile methodologies prioritize active customer collaboration, seeking feedback and involvement to ensure the software meets their needs. It values flexibility and adapting to changing requirements based on customer feedback, rather than relying solely on predefined contracts or specifications.

**Responding to Change over Following a Plan:**

This principle emphasizes the need for adaptability and responsiveness in software development. Agile methodologies recognize that requirements and circumstances can change over time. Instead of rigidly following a predetermined plan, Agile promotes embracing change and adjusting plans accordingly. It encourages iterative development, continuous improvement, and the ability to quickly respond to new information or changing priorities.

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

**Product Backlog Creation:**

The development team, along with the product owner, creates a prioritized list of features and requirements called the product backlog. The backlog is dynamic and can be updated as needed based on changing priorities and customer feedback.

**Sprint Planning:**

The team selects a subset of items from the product backlog to work on during a fixed time period called a sprint. In the sprint planning meeting, the team discusses and estimates the effort required for each item and determines how many items can be completed within the sprint.

**Sprint Execution:**

During the sprint, the team works collaboratively to develop the selected features and functionalities. Daily stand-up meetings are conducted to discuss progress, challenges, and any necessary adjustments to the plan. The work is broken down into smaller tasks, and team members self-organize to complete them.

**Incremental Delivery and Review:**

At the end of each sprint, a potentially shippable increment of the software is delivered. The team showcases the completed work to stakeholders, seeking their feedback and validation. This allows for early and continuous feedback, enabling adjustments and improvements to be made throughout the development process.

**Sprint Retrospective:**

After each sprint, the team holds a retrospective meeting to reflect on the sprint's successes and areas for improvement. The team identifies lessons learned and implements changes to enhance their productivity and effectiveness in subsequent sprints.

**Pros of the Agile Model:**

**Flexibility**: Agile allows for changes and adjustments to be made throughout the development process, accommodating evolving requirements and customer feedback.

**Customer Satisfaction:** The active involvement of customers and stakeholders ensures the software meets their needs and expectations, leading to higher customer satisfaction.

**Early and Continuous Delivery**: Agile promotes delivering working software in short iterations, allowing for early feedback and value realization.

**Collaboration and Communication**: Agile emphasizes collaboration and communication within the development team and with stakeholders, fostering a shared understanding and a more efficient work environment.

**Continuous Improvement:** Agile encourages reflection and adaptation through regular retrospectives, enabling continuous improvement in processes and practices.

**Cons of the Agile Model:**

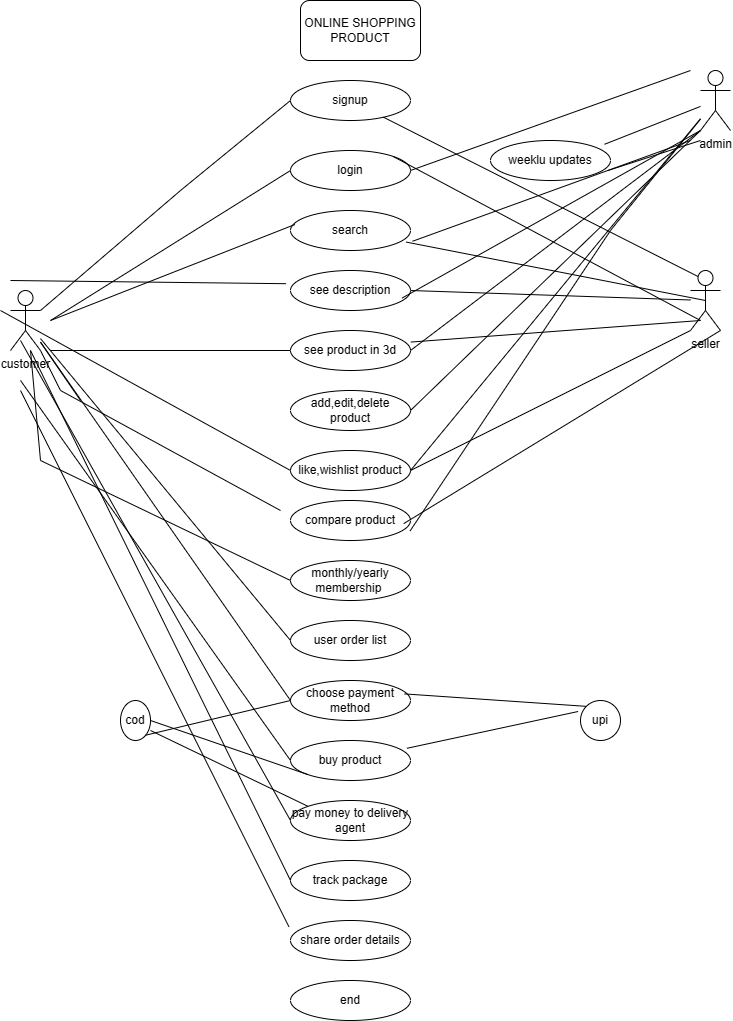
**Uncertainty with Fixed Timelines:** Agile development can be challenging to estimate in terms of time and effort, especially when requirements are unclear or changing.

**Dependency on Customer Availability**: Active customer involvement can sometimes be difficult to achieve if customers are not readily available or lack domain knowledge.

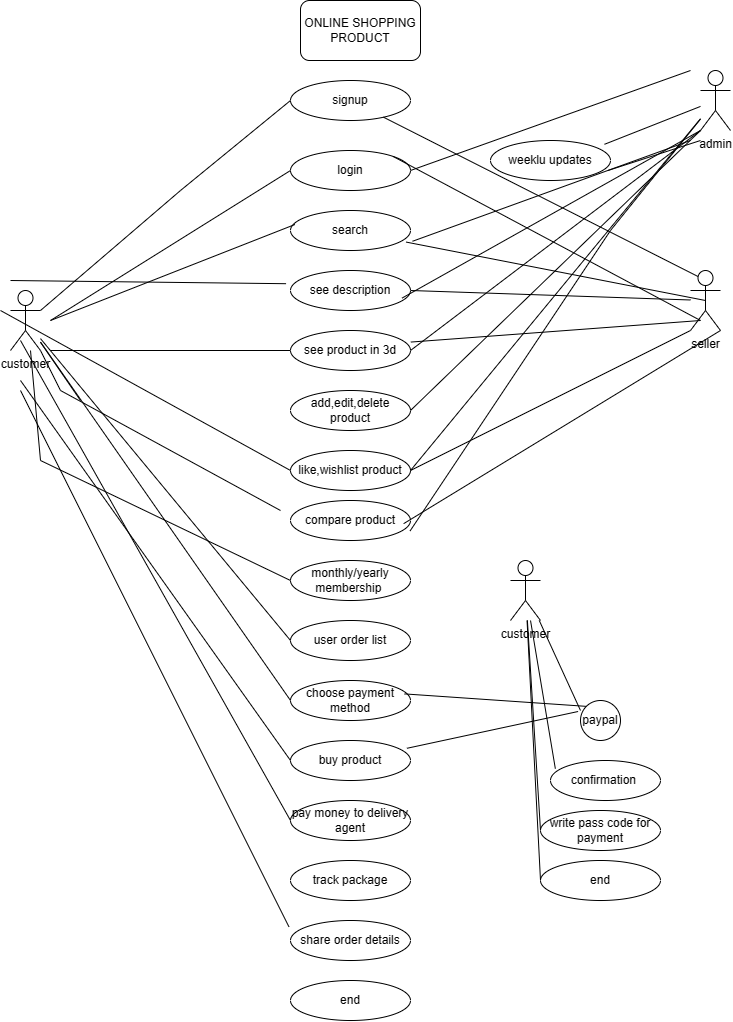
**Lack of Comprehensive Documentation:** Agile values working software over extensive documentation, which can be a challenge for organizations requiring comprehensive documentation for compliance or regulatory purposes.

**Potential for Scope Creep**: The iterative nature of Agile can lead to scope creep if changes and additions are not managed effectively, potentially impacting timelines and resources.

1. Draw usecase on Online shopping product using COD.



1. Draw usecase on Online shopping product using payment gateway



1. Draw Usecase on online bill payment system (paytm)

