Software Testing Assignment

Module–1(Fundamental)

1) What is SDLC

**SDLC** stands for **Software Development Life Cycle**. It's a structured process used by software developers and project managers to design, develop, test, and deploy software efficiently and with high quality.

 What is software testing?

**Software testing** is the process of **evaluating and verifying that a software application works as expected**. The goal is to find bugs or issues and ensure the software meets the specified requirements before it goes live.

 What is agile methodology?

Agile methodology is a project management approach that emphasizes flexibility, collaboration, and continuous improvement in delivering software or products. It breaks projects into small, iterative cycles (sprints or iterations), allowing teams to adapt quickly to changes and deliver value incrementally.

 What is SRS

In software engineering, SRS stands for Software Requirements Specification. It's a document that details what a software system should do and how it should behave. Essentially, it's a contract between the developers, the users, and other stakeholders, outlining the functionality and performance expectations of the software

 What is oops

In computer programming, "OOPS" refers to Object-Oriented Programming System, a programming paradigm that organizes code around objects and classes. It's a way of designing programs using objects, which are like real-world entities with their own properties and behaviors. OOPs is widely used in modern programming languages like Java and Python

 Write Basic Concepts of oops

Object-Oriented Programming (OOP) revolves around the concept of objects, which are instances of classes, which act as blueprints for creating objects. The main principles of OOP include encapsulation, inheritance, polymorphism, and abstraction.

1. Classes and Objects:

* A class is a blueprint or template that defines the characteristics (attributes) and behaviors (methods) of objects.
* An object is a specific instance of a class, created using the class definition.
* Think of a class like a car model (e.g., a sedan), and an object like a specific car (e.g., a blue Toyota Camry) based on that model.

2. Encapsulation:

* **Encapsulation**: is the principle of bundling data (attributes) and the methods that operate on that data into a single unit, creating an object.
* It protects the internal state of an object from external interference and ensures data integrity by limiting direct access to the attributes.
* Encapsulation can be achieved using access modifiers (e.g., private, public) that control how data is accessed.

3. Inheritance:

* **Inheritance**: allows a new class (the "child" or "derived" class) to inherit the attributes and methods of an existing class (the "parent" or "base" class).
* This promotes code reusability and reduces redundancy by allowing the child class to build upon the parent class's functionality.
* Inheritance establishes a relationship between classes, often visualized as a hierarchy.

4. Polymorphism:

* **Polymorphism**: means "many forms" and refers to the ability of objects of different classes to respond differently to the same message or method call.
* It enables flexible and dynamic code by allowing you to treat objects of different classes as if they were of the same class.
* Polymorphism can be achieved through method overriding (where a child class provides its own implementation of a method inherited from the parent class).

5. Abstraction:

* **Abstraction**: is the process of hiding complex implementation details from the user and providing a simplified interface.
* It focuses on the essential features of an object and its behavior, rather than the underlying mechanisms.
* Abstraction helps simplify complex systems by allowing users to interact with objects at a higher level of abstraction.

 What is object

Object is instance of class

 What is class

* A class is a blueprint or template that defines the characteristics (attributes) and behaviors (methods) of objects.

 What is encapsulation

Encapsulation in Object-Oriented Programming (OOP) is the bundling of data (variables) and methods (functions) that operate on that data into a single unit, typically a class. It restricts direct access to some of these data components, promoting data hiding and security.

 What is inheritance

In Object-Oriented Programming (OOP), inheritance is a mechanism where a class (child or derived class) inherits properties and methods from another class (parent or base class). This allows for code reuse and creates a hierarchical structure where classes are organized based on their relationships.

 What is polymorphism

In Object-Oriented Programming (OOP), polymorphism means "many forms." It's the ability of an object to take on many forms, allowing a single interface to represent different types of objects and behaviors. This is achieved through mechanisms like function overloading and overriding, enabling a single function name to be used for different operations or to perform the same operation differently on different objects.

 Draw Usecase on online bill payment system (paytm)

File attache with Document

 Draw Usecase on banking system for customers.

 Draw Usecase on Broadcasting System.

 Write SDLC phases with basic introduction

**Key Phases of the SDLC:**

1. **Planning**  
   Define the project goals, scope, resources, schedule, and risks.
2. **Requirements Gathering and Analysis**  
   Understand what the users need and document the functional and non-functional requirements.
3. **Design**  
   Create architecture and design for the system—like UI design, database schemas, and system interfaces.
4. **Implementation / Coding**  
   Developers write the code based on the design documents.
5. **Testing**  
   Test the software to find and fix bugs. This includes unit testing, integration testing, system testing, and acceptance testing.
6. **Deployment**  
   Release the product to users or move it to a production environment.
7. **Maintenance**  
   Fix bugs, make improvements, and update the system after deployment.

 Explain Phases of the waterfall model

The Waterfall model in software development follows a sequential, linear approach with distinct phases, each completed before the next begins. These phases are: Requirements, Analysis, Design, Implementation, Testing, Deployment, and Maintenance

1. **1. Requirements:**

This initial phase focuses on gathering and documenting all project requirements, including functional and non-functional aspects, and ensuring that the project goals are clearly defined.

1. **2. Analysis:**

This phase involves analyzing the gathered requirements to identify potential problems and ensure that the project plan is feasible.

1. **3. Design:**

The design phase focuses on creating the architecture and detailed specifications for the software, including the system design, database design, and user interface design.

1. **4. Implementation:**

This phase involves coding and developing the software based on the design specifications. It includes activities like writing code, integrating software components, and unit testing.

1. **5. Testing:**

During this phase, the software is rigorously tested to identify and fix any defects, ensuring that it meets the defined requirements.

1. **6. Deployment:**

The software is then deployed to the intended environment, making it available for users to use.

1. **7. Maintenance:**

After deployment, the software is monitored, updated, and maintained to address any issues, improve performance, or add new features as needed.

 Write phases of spiral model

The Spiral Model in software development progresses through four key phases: Planning, Risk Analysis, Engineering (or Development), and Evaluation. Each iteration of this model, known as a spiral, repeats these phases as the project evolves.

Here's a more detailed look at each phase:

* **Planning:**

This initial phase involves defining the project's goals, outlining requirements, and creating a plan for the next iteration of the spiral.

* **Risk Analysis:**

A crucial aspect of the spiral model, this phase focuses on identifying and assessing potential risks, and developing strategies to mitigate them.

* **Engineering (Development):**

This phase involves the actual development of the software, incorporating the identified risks and the refined requirements from previous phases.

* **Evaluation:**

This phase involves reviewing the progress made, testing the developed software, and gathering feedback from users or stakeholders. This feedback is used to inform the next spiral and further refine the product.

In essence, the spiral model follows these steps:

1. **Start with a small set of requirements and a preliminary plan.**
2. **Repeat the four phases (Planning, Risk Analysis, Engineering, Evaluation) in each spiral.**
3. **With each spiral, add functionality, refine the design, and incorporate feedback.**
4. **Continue through multiple spirals until the final product is ready for deployment.**

 Write agile manifesto principles

1. **Individuals and Interactions over Processes and Tools**: Focuses on the importance of effective communication and collaboration among team members.
2. **Working Software over Comprehensive Documentation**: Prioritizes the delivery of functional software as the primary measure of progress.
3. **Customer Collaboration over Contract Negotiation**: Encourages customers and stakeholders to have active involvement throughout the development process.
4. **Responding to Change over Following a Plan**: On changing requirements, embracing flexibility and ability to adapt even late in the development process.

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

Agile Software Development Methodology is a process of software development similar to other software development methodologies like waterfall models, V-models, iterative models, etc. Agile methodology follows the iterative as well as incremental approach that emphasizes the importance of delivering of working product very quickly.

**Advantages of Agile Methodology**

Following are the advantages of agile methodology:

1. **Focus on Customer Value:** Agile places a high priority on providing customers with value by attending to their requirements and preferences. Agile guarantees that the most important features are produced first and that iterative changes are driven by customer feedback by dividing work down into small, manageable tasks.
2. **Enhanced Team Morale and Motivation:** Agile gives teams the freedom to own their work and decide together. Team members feel motivated, proud, and owned when they have this autonomy together with a focus on providing value and ongoing growth.
3. **Stakeholder Collaboration:**Throughout the development process, agile promotes strong coordination between product owners, developers, and other stakeholders. Better communication, a common understanding of the objectives, and ongoing feedback are all fostered by this partnership, which produces results that are higher quality and boost stakeholder satisfaction.
4. **Early and Continuous Delivery:**Agile encourages the tiny, incremental releases of functional software. This gives early access to observable progress and facilitates early input and validation for stakeholders. Continuous delivery reduces risks by spotting problems early on and taking action to fix them.
5. **Delivering high-quality software:** It is a key component of agile development, and this is emphasized by techniques like continuous integration, automated testing, and frequent inspection and modification. Agile guarantees that the software satisfies the required standards and lowers the likelihood of faults by integrating quality assurance throughout the development process.

**Disadvantages of the Agile Methodology**

Following are the disadvantages of the agile methodology:

1. **Lack of Predictability:** Project timeframes and outcomes might be difficult to predict with accuracy due to Agile iterative and incremental methodology. Stakeholders who need set budgets or timeframes may find this unpredictability troublesome.
2. **Dependency on Customer Availability:** Agile highly depends on ongoing customer and stakeholder feedback and participation. Customers who are unavailable or who don’t know enough about the domain can impede development and slow it down.
3. **Scaling Agile**: While Agile works effectively for small to medium-sized teams working on relatively basic projects, scaling Agile methods to bigger teams or more complicated projects can be more difficult. As the project grows, it gets harder to maintain coordination, alignment, and communication.
4. **Dependency on Team Dynamics:** Agile’s focus on self-organizing, cross-functional teams with the authority to reach decisions together is paramount. Inadequate communication within the team or a lack of experience or expertise among team members can negatively affect output quality and productivity.
5. **Increased Overhead:** Planning, coordinating, and communicating take more time and effort when using agile frameworks like Scrum. This overhead can take a lot of time, especially for projects with short deadlines or small teams.

 Draw usecase on OTT Platform.

 Draw usecase on E-commerce application

 Draw usecase on Online shopping product using payment gatewa