**Software Testing Assignment**

**Module – 1 ( Fundamental )**

* **What is SDLC ?**

**Ans.: SDLC** stands for **Software Development Life Cycle**. It is a **process used by software developers and project teams to design, develop, test, and deploy software systems** efficiently and with quality.

* **What is Software testing ?**

**Ans.:**  Software testing is like checking a product before giving it to the customer to make sure it's working fine.

* **What Is Agile Methodology ?**

**Ans.:** Agile methodology is a **flexible and team-based** way of making software. It focuses on **creating and delivering small parts of the software quickly**, instead of waiting to finish the whole thing at once.

* **What Is SRS ?**

**Ans.:** A **Software Requirement Specification (SRS)** is a formal document that clearly describes the **functions**, **features**, and **behavior** of a software application. It serves as a **communication bridge** between the client (or user) and the development team. The main goal of an SRS is to **clearly define what the software must do** so that everyone involved in the project is on the same page.

* **What Is OOPS ?**

**Ans.:** **OOPS** stands for **Object-Oriented Programming System**.  
It is a way of writing and organizing code in programming that is **based on real-world objects** like a car, person, or bank account.

* **Write Basic Concepts of OOPS…**

**Ans.:** 1) Class 4) Polymorphism

2) Object 5) Abstraction

3) Inheritance 6) Encapsulation

* **What Is Object ?**

**Ans.:** An **object** is something you create **based on a class**.  
It is like a **real thing** that has **data (properties)** and can **do actions (methods)**.

* **What Is Class ?**

**Ans.:** A **class** is a **blueprint** or **template** used to create **objects**.

It defines **how an object should look** (its data) and **what it should do** (its functions).

* **What Is Encapsulation ?**

**Ans.:** **Encapsulation** is one of the main concepts of Object-Oriented Programming (OOPS).  
It means **hiding the internal details** of an object and **only showing what is necessary**.

* **What Is Inheritance ?**

**Ans.:** **Inheritance** allows a class (child) to use the properties and methods of another class (parent).

It promotes **code reusability**.

* **What Is Polymorphism ?**

**Ans.:** Polymorphism means that **same function name but having different functionality .**

* Draw Use-case on online bill payment system ( paytm ) .

**Ans.:** <https://app.diagrams.net/index.html#G1OSUXnjM-6pqlTqzjr-JxAL_PKwRQSNK9#%7B%22pageId%22%3A%22bMsqoK-PSgCk8Bz9sl4c%22%7D>

Draw Use-case on banking system for customers.

**Ans.:** <https://app.diagrams.net/index.html#G182b3kcbIlJe8Y1pLm9r5vQgGDYGKhfkY#%7B%22pageId%22%3A%22TJ1Gyt1tqYp3FCeERuQ1%22%7D>

Draw Use-case on Broadcasting System.

**Ans.:** [**https://app.diagrams.net/index.html#G1p-AqEV50OJOkI3DQjB9e-C168jlk9c0B#%7B%22pageId%22%3A%22PwFBTBvHyyJnFjZzAsF9%22%7D**](https://app.diagrams.net/index.html#G1p-AqEV50OJOkI3DQjB9e-C168jlk9c0B#%7B%22pageId%22%3A%22PwFBTBvHyyJnFjZzAsF9%22%7D)

* **Write SDLC Phases With Basic Introduction…**

**Ans.:** SDLC ( Software Development Life Cycle ) has 6 important phases :

1. Requirement Collection/Gathering
2. Analysis
3. Design
4. Implementation ( Coding )
5. Testing
6. Maintenance

**1) Requirement Collection/Gathering**

In this phase, we talk to the customer and understand what they want in the software.

**2) Analysis**

Here, we study the requirements and decide what is possible and how we can build it.

**3) Design**

We make a plan or layout of how the software will look and work.

**4) Implementation (Coding)**

In this step, developers write the actual code to create the software.

**5) Testing**

We check the software to make sure it works correctly and has no errors.

**6) Maintenance**

After the software is released, we fix any problems and make updates if needed.

* **Explain Phases of waterfall model…**

**Ans.:**

**1) Requirement Collection / Gathering :**This is the **first step** where the team talks to the client to understand **what they want** the software to do. All the needs and expectations are collected.

**Example**: A client says, “I want an app to order food online.”

**2) Analysis :**

After gathering the requirements, the team carefully **studies and understands** them. They also decide **what is possible**, what tools to use, and how to solve problems.

**Example**: The team checks if payment integration and live tracking are technically possible.

**3) Design :**

Now, the team **creates a plan or layout** for how the software will look and work. It includes screen designs, database structures, and system flow.

**Example**: Designing the home page of the food app, the order page, and how data will be stored.

**4) Implementation (Coding) :**

In this step, **developers start coding** based on the design. They create the actual working software.

**Example**: Developers build the login system, food menu, and payment gateway.

**5) Testing :**

After the software is built, it’s tested to **find and fix any mistakes or bugs**. The goal is to make sure it works correctly.

**Example**: Testers check if food orders are placed properly and if the app crashes anywhere.

**6) Maintenance :**

Once the software is released, the team handles any **problems, updates, or improvements** needed over time.

**Example**: If users request a “dark mode” or report a bug, the team updates the app.

* **Write Phases Of Spiral Model …**

**Ans.:** The **Spiral Model** is a **risk-driven** software development process. It combines the idea of **repeating steps (iteration)** with **careful planning and risk checking**. The process moves in a spiral shape — repeating phases in cycles until the final product is ready.

Each cycle (or spiral) has 4 main phases:

**1) Planning Phase :**

* In this phase, the team **collects requirements** and decides what to do in the next cycle.
* The goal is to understand what features or parts of the software should be developed.

**Example**: Planning to build the login and signup feature in the first cycle.

**2) Risk Analysis Phase :**

* The team checks for **possible problems or risks**, like technical issues or budget problems.
* They also plan how to **solve or avoid** these risks.

**Example**: If there’s a chance that payment might fail, the team finds a backup plan.

**3) Engineering Phase (Development and Testing) :**

* In this phase, the team **designs, codes, and tests** the selected part of the software.
* The product is developed **bit by bit in each cycle**.

**Example**: Building and testing only the login/signup part in this cycle.

**4) Evaluation Phase:**

* The team **shows the current version** to the customer, gets feedback, and reviews what was done.
* This helps in planning for the **next cycle**.

**Example**: Client reviews the login feature and suggests adding OTP verification.

* **Write Agile Manifesto Principles…**

**Ans.:** There is 5 manifesto principles as under :

**1. Customer Satisfaction Through Early and Continuous Delivery**

This is the heart of Agile. Deliver software in small parts and keep the customer happy throughout the project.

**2. Welcome Changing Requirements, Even Late**

Agile is flexible — changes are not a problem, but a **part of the process**.

**3. Deliver Working Software Frequently**

Short, regular releases help you get **quick feedback** and improve faster.

**4. Working Software is the Primary Measure of Progress**

Documents and plans don’t matter as much as software that **actually works**.

**5. Regular Reflection and Improvement**

Agile teams should keep learning and improving after each sprint or iteration.

* **Explain Working Methodology Of Agile Model And Also Write Pros And Cons …**

**Ans.:**

The **Agile Model** is a flexible and iterative approach to software development. Instead of building the whole software at once, it is built in **small parts called iterations or sprints** (usually 1 to 4 weeks).

Each iteration includes:

1. **Planning**
2. **Designing**
3. **Coding**
4. **Testing**
5. **Review and Feedback**

**How Agile Works:**

1. **Customer tells the team** what they want.
2. The team plans a **small part** of the project.
3. They **design, develop, and test** that part.
4. They **deliver the working software** to the customer.
5. The customer gives **feedback**, and changes are made.
6. The next part of the software is taken up in the next **iteration**.

This process **repeats** until the full software is complete.

**Pros of Agile Model :**

* **Quick delivery** of working software
* **Easy to make changes** anytime
* **Customer is always involved**
* **Better quality** through regular testing
* **Improves team communication and teamwork**
* **More focus on what users really need**

**Cons of Agile Model :**

* **Less documentation**, which can be a problem later
* Needs a **skilled and disciplined team**
* **Hard to predict time and budget**
* **Frequent changes** can cause confusion
* Not suitable for **very small or short-term projects**
* **Draw Use-Case On OTT Platform…**

**Ans.:** <https://app.diagrams.net/index.html#G1hPHOMRSKwTrZ36N4heU8_0zBQd0h74Yd#%7B%22pageId%22%3A%22BQ7esZMDXKvQ5ubbfvjg%22%7D>

* **Draw Use -Case On E – Commerce Application …**

**Ans.:** <https://app.diagrams.net/index.html#G1AJOyw8MBZhsp0gBYjD0jnXCko1QSH_06#%7B%22pageId%22%3A%22D1TcASS1_OcEyGjUBE6h%22%7D>

* **Draw use-case on online shopping product using payment gateway…**

**Ans.:** <https://app.diagrams.net/index.html#G1YQU4UraRaqP74_hpSAwxnvLc2SSrda9V#%7B%22pageId%22%3A%22Te1p6j-TxLLQCnJeUwl1%22%7D>