# **Module–1 (Fundamental)**

## **1. What is SDLC.**

* SDLC is a structure imposed on the development of a software product that defines the process for
* planning, implementation, testing, documentation, deployment, and ongoing maintenance and
* support.
* There are a number of different development models.
* A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the developmentand lifecycle management of an application or piece of software.

## **2. Write SDLC phases with basic introduction.**

* Requirement Gathering and Analysis
  + This phase involves collecting requirements from stakeholders (clients, users, business analysts).
  + The goal is to understand what the system should do.
  + A feasibility study is also conducted.
* System Design
  + In this phase, the software architecture is designed based on the gathered requirements.
  + It includes high-level design (HLD) and low-level design (LLD).
  + Outputs include data flow diagrams, system architecture, and database design.
* Implementation
  + Developers write code based on the system design.
  + Programming languages and tools are chosen in this phase.
  + It is the actual construction of the system.
* Testing
  + The developed software is tested to ensure it meets the required standards.
  + Various types of testing are done, including unit testing, integration testing, system testing, and user acceptance testing (UAT).
  + Bugs are identified and resolved.
* Maintenance
  + Ongoing support and bug fixing are handled in this phase.
  + Enhancements or updates may be released based on user feedback.
  + This phase continues as long as the software is in use.

## **3. What is software testing?**

* SoftwareTesting is a process used to identify the correctness, completeness, and quality of developed computer software.
* It can also be stated as the process of validating and verifying that a software program application or product:
  + Meets the business and technical requirements that guided it’s design and development
  + Works as expected
  + Can be implemented with the same characteristic

## **4. What is agile methodology?**

* Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.
* Agile Methods break the product into small incremental builds. These builds are provided in iterations.
* Each iteration typically lasts from about one to three weeks.
* Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.
* At the end of the iteration a working product is displayed to the customer and important stakeholders.

## **5. What is SRS?**

* **SRS** stands for **Software Requirements Specification**. It is a detailed document that describes the software system to be developed, its functional and non-functional requirements, and serves as a reference point for both the development team and stakeholders.

## **6. What is oops?**

* Identifying objects and assigning responsibilities to these objects.
* Objects communicate to other objects by sending messages.
* Messages are received by the methods of an object
* An object is like a black box. The internal details are hidden.
* Object is derived from abstract data type
* Object-oriented programming has a web of interacting objects, each house-keeping its own state.
* Objects of a program interact by sending messages to each other.

## **7. Write Basic Concepts of oops?**

* Class
* Object
* Encapsulation
* Inheritance
* Polymorphism
* Abstraction

## **8. What is object**

* Object is an instance of a class. It has state and behavior.
* This is the basic unit of object oriented programming(OOP).
* That is both data and function that operate data are bundled as unit called as object.

## **9. What is class?**

* Class is structured in which we have member functions and variables are there.
* A class represents an abstraction of the object and abstracts the properties and behavior of that object.
* An Object is a particular instance of a class which has actual existence and there can be many objects (or instances) for a class.

## **10. What is encapsulation?**

* To wrap data into a single unit is called encapsulation.
* Encapsulation is placing the data and the functions that work on that data in the same place. While working with procedural languages, it is not always clear which functions work on which variables but object- oriented programming provides you a framework to place the data and the relevant functions together in the same object.
* Encapsulation In Java is the process of wrapping up data (properties) and behavior (methods) of an object into a single unit; and the unit here is a Class (or interface).
* Encapsulate in plain English means to enclose or be enclosed in or as if in a capsule. In Java, a class is the capsule (or unit).

## **11. What is inheritance?**

* To access property of one class to another class.
* In general, Java supports single-parent, multiple-children inheritance and multilevel inheritance (Grandparent-> Parent -> Child) for classes and interfaces. Java supports multiple inheritances (multiple parents, single child) only through interfaces.

## **12. What is polymorphism?**

* Polymorphism Is known as the same function name but having different functionality.
* There is two types of polymorphism in Java Compile time polymorphism(Overloading) Runtime polymorphism(Overriding)
* **Overloading:** The concept of overloading is also a branch of polymorphism. When the existing operator or function is made to operate on a new data type, it is said to be overloaded.
  + The same method name (method overloading) or operator symbol (operator overloading) can be used in different contents.
  + In method overloading, multiple method having same name can appear in class, but with different signature.
* **Overriding:** Overriding is a concept in object-oriented programming (OOP), where a subclass provides its own specific implementation of a method that is already defined in its superclass (parent class).
  + In simpler terms, when a subclass has a method with the same name and signature as a method in its superclass, the subclass method *overrides* the superclass method. This allows the subclass to change or extend the behavior of that method without affecting the parent class.

## **13. Draw Usecase on online bill payment system (paytm)**

**Link:** <https://drive.google.com/file/d/1-R4-nydBBeaHbAmJN-cqbJwGRi-w0Pxi/view?usp=sharing>

## **14. Draw Usecase on banking system for customers.**

**Link:** <https://drive.google.com/file/d/1OrtZtKFE7R-8glzAo-rwhg9fZ9Kgibtb/view?usp=sharing>

## **15, Draw Usecase on Broadcasting System.**

**Link:** <https://drive.google.com/file/d/1-nT1ipQU7R9kYdr35HET4wHScyoSAGTZ/view?usp=sharing>

## **16. Explain Phases of the waterfall model.**

### **1. Requirements Gathering and Analysis**

* **Objective**: Understand and document what the stakeholders need.
* **Activities**: Collect all the requirements for the system to be built, typically involving meetings with stakeholders, clients, and users. This phase results in a detailed requirement specification document.
* **Output**: **Requirement Specification Document** (SRS - Software Requirement Specification).

### **2. System Design**

* **Objective**: Plan how the system will be built.
* **Activities**: The system's architecture is designed, detailing both high-level and low-level design aspects. This phase typically results in system design documents that specify the hardware and software requirements.  
  **Output**: **System Design Documents**, which describe the overall system architecture and individual components.

### **3. Implementation (Coding)**

* **Objective**: Translate design into actual working software.  
  **Activities**: Developers write the code based on the design specifications. This is typically the longest phase where the actual construction of the software happens.
* **Output**: **Source Code** for the system.

### **4. Integration and Testing (Verification)**

* **Objective**: Ensure the system works as expected and is free from defects.
* **Activities**: Once the software is built, it's integrated and tested to find defects. Testing may include unit testing, integration testing, system testing, and acceptance testing. The goal is to verify that the system meets the specified requirements.
* **Output**: **Test Reports** and **Bug Reports**. If defects are found, they are fixed and retested.

### **5. Deployment (Installation)**

* **Objective**: Make the system available for use.
* **Activities**: After successful testing, the system is deployed to the user's environment. This could involve system installation, data migration, and user training.
* **Output**: **Deployed System** ready for user operations.

### **6. Maintenance**

* **Objective**: Handle issues and make updates after deployment.
* **Activities**: Once the software is live, the team addresses any issues that arise (bug fixes, security patches, minor improvements). This phase continues as long as the system is in use.
* **Output**: **Patches** and **Updates** to the system as needed.

## **17. Write phases of spiral model.**

### **1. Planning (Objective Setting)**

* **Objective**: Define objectives, constraints, and system requirements.
* **Activities**: In this phase, the project team defines the goals, requirements, and constraints of the system. The project's scope, high-level risks, deliverables, and constraints are outlined.
* **Output**: **Initial Risk Analysis**, **Project Plan**, and **Objectives** for the next iteration.

### **2. Risk Analysis and Evaluation**

* **Objective**: Identify, analyze, and mitigate risks.
* **Activities**: The team identifies potential risks, evaluates them, and develops strategies to mitigate those risks. This can include technical risks, resource risks, and market risks. Prototypes or simulations may be created to better understand potential issues before proceeding further.  
  **Output**: **Risk Management Plan**, **Risk Mitigation Strategies**, and **Risk Analysis Report**.

### **3. Engineering (Development and Validation)**

* **Objective**: Develop the system and validate its components.  
  **Activities**: In this phase, the system is designed, developed, and tested. This can involve building prototypes, coding, testing, and validating each part of the system. Depending on the project's scope, the engineering phase may focus on one iteration of the system, building and testing smaller portions of the project.
* **Output**: **System Design Documents**, **Prototypes**, **Test Results**, and **Code** for the system.

### **4. Evaluation (Customer Feedback)**

* **Objective**: Obtain feedback from stakeholders and adjust the project plan accordingly.  
  **Activities**: At the end of each cycle, feedback is gathered from stakeholders, including clients, end-users, and team members. This feedback helps to evaluate whether the system is progressing as expected and whether any adjustments are needed. The project team assesses the project's progress, revisits its goals, and adapts the next iteration.
* **Output**: **Customer Feedback**, **Revised Project Plans**, and **Decision on Proceeding** to the next spiral.

## **18. Write agile manifesto principles.**

1. **Customer Satisfaction**: Deliver valuable software early and continuously.
2. **Welcome Change**: Adapt to changing requirements, even late in development.
3. **Frequent Delivery**: Deliver working software regularly (every few weeks).
4. **Collaboration**: Business stakeholders and developers work closely.
5. **Motivated Individuals**: Build projects around motivated teams.
6. **Face-to-Face Communication**: Direct communication is most effective.
7. **Working Software**: The primary measure of progress.
8. **Sustainable Development**: Maintain a consistent pace.
9. **Technical Excellence**: Focus on good design and technical quality.
10. **Simplicity**: Maximize work that is not done.
11. **Self-Organizing Teams**: Best results come from empowered teams.
12. **Continuous Improvement**: Regular reflection on how to improve.

## **19. Explain the working methodology of agile model and also write pros and cons.**

1. **Iterative & Incremental**: Development is done in small cycles (sprints), with each cycle resulting in working software.
2. **Sprint Planning**: Select tasks (user stories) to complete in a sprint (usually 1-4 weeks).
3. **Daily Stand-ups**: Brief daily meetings to track progress and resolve blockers.
4. **Sprint Review**: Show the working product to stakeholders and gather feedback.
5. **Sprint Retrospective**: Reflect on the sprint and find ways to improve.
6. **Repeat**: Continue with the next sprint based on feedback and priorities.

**Pros**:

* Is a very realistic approach to software development Promotes teamworkand cross training.
* Functionality can be developed rapidly and demonstrated. Resource requirements are minimum.
* Suitable for fixed or changing requirements Delivers early partial working solutions.
* Good model for environments that change steadily. Minimal rules, documentation easily employed.
* Enables concurrent development and delivery within an overall
* planned context.
* Little or no planning required Easy to manage
* Gives flexibility to developers

**Cons:**

* Not suitable for handling complex dependencies.
* More risk of sustainability, maintainability and extensibility.
* An overall plan, an agile leader and agile PM practice is a must without which it will not work.
* Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
* Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
* There is very high individual dependency, since there is minimum documentation generated.
* Transferof technology to new team members may be quite challenging due to lack of documentation.

## **20. Draw use case on OTT Platform.**

**Link:** <https://drive.google.com/file/d/1HP7udIRQj217LwPire2ouHQHEGu0jvRo/view?usp=sharing>

## **21. Draw use case on E-commerce application.**

**Link:** <https://drive.google.com/file/d/1RqyfptwH-QPx3g5V1O8ZNRjra_2gB8-B/view?usp=sharing>

## **22. Draw use case on Online shopping product using payment gateway.**

**Link:** <https://drive.google.com/file/d/1blicFLDbCmoQ2k-kdLY6gEap5IGk36s5/view?usp=sharing>