# ASSIGNMENT : 1

# MODULE : 1

* What is SDLC.
* **SDLC** stands for **Software Development Life Cycle**.
* It is a structured process used by software developers and organizations to design, develop, test, and deploy software systems efficiently and with high quality.
  1. **Phases of SDLC**

1. Requirement Gathering
2. Analysis
3. Designing
4. Implementation
5. Testing
6. Maintenance

* What is software testing?
* **Testing** in software development is the process of evaluating and verifying that a software application or system meets the specified requirements and works correctly.
* It helps to identify bugs (errors), missing requirements, or gaps in the software.
* **Software Testing** is the process of executing a program or system with the intent of finding errors and ensuring that the software system meets the business and technical requirements.
* The process consisting of all life cycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.
* What is agile methodology?
* Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements.
* 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.
* 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.
* What is SRS.
* A software requirements specification (SRS) is a complete description of the behavior of the system to be developed.
* It includes a set of use cases that describe all of the interactions that the users will have with the software.
* Use cases are also known as functional requirements. In addition to use cases, the SRS also contains nonfunctional (or supplementary) requirements.
* Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance requirements, quality standards, or design constraints)
* Recommended approaches for the specification of software requirements are described by
* This standard describes possible structures, desirable contents, and qualities of a software requirements specification.
* 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-oriented programming has a web of interacting objects, each house-keeping its own state.
* It is like writing, to be a good writer needs lots of experience and lots of knowledge about the world.
* Objects of a program interact by sending messages to each other.

* Write Basic Concepts of oops.

1. Object

2. Class

3. Encapsulation

4. Inheritance

5. Polymorphism

6. Abstraction

* What is Object.
* **object** is a **self-contained unit** that combines **data** and **methods** (functions) that operate on that data. It is a **concrete instance of a class**
* Object is the item or component you are testing validating.
* GUI element (button,textbox,or drop-down).
* **object** is a fundamental building block in object-oriented programming. It is an **instance of a class** that contains both **data** and the **methods** that act on that data.
* What is class.
* **class** in programming, especially in **object-oriented programming (OOP)**, is essentially a **blueprint** or **template** for creating objects.
* It defines a **structure** for an object and specifies what **attributes** (data) and **methods** (functions) the objects created from the class will have.
* What is encapsulation.
* Encapsulation is the practice of including in an object everything it needs hidden from other objects.
* The internal state is usually not accessible by other objects.
* Encapsulation in Java is the process of wrapping up of 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).
* What is enheritance.
* Inheritance means that one class inherits the characteristics of another class. This is also called a “is a” relationship.
* One of the most useful aspects of object-oriented programming is code reusability.
* As the name suggests Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.
* Inheritance describes the relationship between two classes.
* A class can get some of its characteristics from a parent class and then add unique features of its own.
* What is polymorphism.
* Poly refers to many. That is a single function or an operator functioning in many ways different upon the usage is called polymorphism.
* The ability to change form is known as polymorphism.
* E.g. the message displayDetails() of the Person class should give different results when send to a Student object (e.g. the enrolment number).
* Draw usecase on online book shopping.
* DIAGRAME

|  |
| --- |
| User |

* Draw usecase on online bill payment system  
  (paytm).
* DIAGRAME

|  |
| --- |
| User |

|  |
| --- |
| Bank |

|  |
| --- |
| Payment |

* Write SDLC phase with basic introduction.
* Requirement collection/Gathering
* Although requirements may be documented in written form, they may be incomplete, unambiguous, or even incorrect.
* Inadequately captured or expressed in the first place
* Validation is needed throughout the software lifecycle, not only when the “final system” is delivered.
* Analysis
* The analysis phase defines the requirements of the system, independent of how these requirements will be accomplished.
* This analysis represents the “what” phase.
  1. Design
* Design Architecture Document
* The Design team can now expand upon the information established in the requirement document.
* The architecture team also converts the typical scenarios into a test plan.
  1. Implementation
* In the implementation phase, the team builds the components either from scratch or by composition.
* Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility.
* The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging.
  1. Testing
* Simply stated, quality is very important. Many companies have not learned that quality is important and deliver more claimed functionality but at a lower quality level.
* It is much easier to explain to a customer why there is a missing feature than to explain to a customer why the product lacks quality.
  1. Maintenance
* Software maintenance is one of the activities in software engineering, and is the process of enhancing and optimizing deployed software (software release), as well as fixing defects.
* Software maintenance is also one of the phases in the System Development Life Cycle (SDLC), as it applies to software development. The maintenance phase is the phase which comes after deployment of the software into the field.
* Explain phase of the waterfall model.
* The waterfall is unrealistic for many reasons, especially
* Requirements must be “frozen” to early in the life cycle.
* Requirements are validated too late.
* Requirements are very well documented, clear and fixed
* Ample resources with required expertise are available to support the product.
  1. Pros
* Simple and easy to understand and use.
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
* Phases are processed and completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Clearly defined stages.
* Well understood milestones.
  1. Cons
* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So risk and uncertainty is high with this process model.
* Adjusting scope during the life cycle can end a project.
* Write phases of spiral model.
* Spiral Model is very widely used in the software industry as it is in synch with the natural development process of any product i.e. learning with maturity and also involves minimum risk for the customer as well as the development firms.
* When costs there are a budget constraint and risk evaluation is important.
* For medium to high-risk projects.
* Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
* Customer is not sure of their requirements which are usually the case.
* New product line which should be released in phases to get enough customer feedback.
* Significant changes are expected in the product during the development cycle.
  1. Pros
* Changing requirements can be accommodated.
* Allows for extensive use of prototypes Requirements can be captured more accurate.
* Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.
  1. Cons
* Management is more complex.
* End of project may not be known early.
* Not suitable for small or low risk projects and could be expensive for small projects.
* Large number of intermediate stages requires excessive documentation.
* Write agile manifesto principles.

1. **Our highest priority is to satisfy the customer**  
   through early and continuous delivery of valuable software.
2. **Welcome changing requirements,**  
   even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**,  
   from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. **Business people and developers must work together daily**Close, daily cooperation between business stakeholders and developers is essential.
5. **Build projects around motivated individuals**  
   Give them the environment and support they need, and trust them to get the job done.
6. **Face-to-face conversation is the best form of communication**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**  
   Delivering valuable, functioning software is more important than comprehensive documentation.
8. **Sustainable development**  
   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** Focus on what truly adds value.
11. **Self-organizing teams produce the best architectures, requirements, and designs** The best outcomes emerge from empowered and autonomous teams.
12. **Regular reflection and adjustment** At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

* Explain working methodology of agile model and also write pros and cons.
* 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.
* 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.
* Pros
* Is a very realistic approach to software development
* Promotes teamwork and 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
* 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.
* Draw usecase on online shopping product using COD
* Draw usecase on online shopping product using payment gateway.
* DIAGRAME

User