**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.

SDLC Phase

* Requirements Collection/Gathering
* Analysis
* Design
* Implementation
* Testing
* Maintenance

Requirement Gathering

Features

* Usage scenarios
* Although requirements may be documented in written form, they may be incomplete, unambiguous, or even incorrect.
* Requirements will Change! Inadequately captured or expressed in the first place User and business needs change during the project
* Functional and Non-Functional

Analysis Phase

* This phase starts with the requirement document delivered by the requirement phase and maps the requirements into architecture.
* The architecture defines the components, their interfaces and behaviors.
* The deliverable design document is the architecture.
* This phase represents the “how” phase.

Design Phase

* + Design Architecture Document
  + Implementation Plan
  + Critical Priority Analysis
  + Performance Analysis
  + Test Plan
  + The Design team can now expand upon the information established in the requirement document.

Implementation Phase

* + 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.
  + For example, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline.
  + Implementation – Code
  + Critical Error Removal
* The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging.

Testing Phase

* 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.
* A customer satisfied with the quality of a product will remain loyal and wait for new functionality in the next version.

Maintenance Phase

* 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.
* The developing organization or team will have some mechanism to document and track defects and deficiencies.
* configuration and version management

**What is software testing?**

Software Testing is a process used to identify the correctness, completeness, and quality of developed computer software.

Testing is a process that’s take place throughout the Software Development Life Cycle (SDLC).

* Static Testing: It can test and find defects without executing code. Static Testing is done during verification process. This testing includes reviewing of the documents (including source code) and static analysis. This is useful and cost effective way of testing. For example: reviewing, walkthrough, inspection, etc.
* Dynamic Testing: In dynamic testing the software code is executed to demonstrate the result of running tests. It’s done during validation process. For example: unit testing, integration testing, system testing, etc.
* Planning: We need to plan as what we want to do. We control the test activities, we report on testing progress and the status of the software under test.
* Preparation: We need to choose what testing we will do, by selecting test conditions and designing test cases.

Testing Activities

* Planning and control
* Choosing test conditions
* Designing test cases
* Checking results
* Evaluating completion criteria
* Reporting on the testing process and system under test
* Finalizing or closure (e.g. after a test phase has been completed)
* Testing also includes reviewing of documents (including source code) and static analysis

Test Objectives

* Finding defects
* Gaining confidence in and providing information about the level of quality.
* Preventing defects
* Both dynamic testing and static testing can be used as a means for achieving these objectives
* By designing tests early in the project life cycle it can help to prevent defects from being introduced into code

**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.

**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).

Types of Requirements

* Requirements are categorized in several ways. The following are common categorizations of requirements that relate to technical management:
* Customer Requirements
* Functional Requirements
* Non-Functional Requirements

Customer Requirements

The customers are those that perform the eight primary functions of systems engineering, with special emphasis on the operator as the key customer. Operational requirements will define the basic need.

Functional Requirements

Functional Requirements are very important system requirements in the system design process. These requirements are the technical specifications, system design parameters and guidelines, data manipulation, data processing, and calculation modules etc, of the proposed system.

Non-Functional Requirements

Non-functional requirements are requirements that specify criteria that can be used to judge the operation of a system, rather than specific behaviors. Non-functional requirements are qualities or standards that the system under development must have or comply with, but which are not tasks that will be automated by the system.

**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.

The two steps of Object Oriented Programming

* + Making Classes: Creating, extending or reusing abstract data types.
  + Making Objects interact: Creating objects from abstract data types and defining their relationships.

**Write Basic Concepts of oops**

* Object
* Class
* Encapsulation
* Inheritance
* Polymorphism
  + Overriding
  + Overloading
* Abstraction

**What is object**

* An object represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain.
* An "object" is anything to which a concept applies.
* This is the basic unit of object oriented programming(OOP).
* That is both data and function that operate on data are bundled as a unit called as object.

**What is class**

* When you define a class, you define a blueprint for an object.
* This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.
* A class represents an abstraction of the object and abstracts the properties and behavior of that object.
* Class can be considered as the blueprint or definition or a template for an object and describes the properties and behavior of that object, but without any actual existence.
* An object is a particular instance of a class which has actual existence and there can be many objects (or instances) for a class.
* In the case of a car or laptop, there will be a blueprint or design created first and then the actual car or laptop will be built based on that.

**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 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 objectoriented programming provides you framework to place the data and the relevant functions together in the same object.
* 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).
* Encapsulation enables data hiding, hiding irrelevant information from the users of a class and exposing only the relevant details required by the user.
* We can protect the internal state of an object by hiding its attributes from the outside world (by making it private), and then exposing them through setter and getter methods. Now modifications to the object internals are only controlled through these methods.

**What is inheritance**

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.

This is a very important concept of object-oriented programming since this feature helps to reduce the code size.

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**

* Polymorphism means “having many forms”.
* It allows different objects to respond to the same message in different ways, the response specific to the type of the object.
* The most important aspect of an object is its behaviour (the things it can do). A behaviour is initiated by sending a message to the object (usually by calling a method).
* The ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism.
* The ability to change form is known as polymorphism.
* There is two types of polymorphism in Java
* Compile time polymorphism(Overloading)
* Runtime polymorphism(Overriding)

**Write SDLC phases with basic introduction**

SDLC Phase

* Requirements Collection/Gathering
* Analysis
* Design
* Implementation
* Testing
* Maintenance

Requirement Gathering

Features

* Usage scenarios
* Although requirements may be documented in written form, they may be incomplete, unambiguous, or even incorrect.
* Requirements will Change! Inadequately captured or expressed in the first place User and business needs change during the project
* Functional and Non-Functional

Analysis Phase

* This phase starts with the requirement document delivered by the requirement phase and maps the requirements into architecture.
* The architecture defines the components, their interfaces and behaviors.
* The deliverable design document is the architecture.
* This phase represents the “how” phase.

Design Phase

* + Design Architecture Document
  + Implementation Plan
  + Critical Priority Analysis
  + Performance Analysis
  + Test Plan
  + The Design team can now expand upon the information established in the requirement document.

Implementation Phase

* + 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.
  + For example, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline.
  + Implementation – Code
  + Critical Error Removal
* The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging.

Testing Phase

* 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.
* A customer satisfied with the quality of a product will remain loyal and wait for new functionality in the next version.

Maintenance Phase

* 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.
* The developing organization or team will have some mechanism to document and track defects and deficiencies.
* configuration and version management

**Explain Phases of the waterfall model**

* 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.
* Easy to arrange tasks.
* Process and results are well documented

Applications(When to use?)

* Requirements are very well documented, clear and fixed.
* Product definition is stable.
* Technology is understood and is not dynamic.
* There are no ambiguous requirements.
* Ample resources with required expertise are available to support the product.
* The project is short.

**Write phases of spiral model**

* Planning
* Risk Analysis
* Customer Evaluation
* Engineering

Planning

* This phase begins by gathering business requirements into a baseline spiral. In the subsequent spiral, all system, subsystem and unit requirements are identified at this stage as the product matures.
* This phase also includes understanding system requirements through ongoing customer and system analyst communication. The product will be deployed in the identified market at the end of the spiral. This includes iteration cost, schedule, and resource estimates. This includes understanding system requirements for ongoing communication between system analysts and customers.

Risk Analysis

* After the “plan” phase, the team prepares for the “risk” phase. The “risk” phase is designed to consider the variability in the rate at which a given product might fail. It is designed to account for the uncertainty in the rate at which a given product might fail. During the “risk” phase, the team evaluates various aspects of the current state of the product, such as the state of its code, the state of its design, and the state of its prototype. The team then makes adjustments to the current state of the product based on the changes made in the “plan” phase and then follows up with a “sales” phase to collect customer feedback.
* Once  risks are identified, risk mitigation strategies are planned and completed.
* Briefly, risk analysis involves identifying, estimating and monitoring technical feasibility and management risks such as schedule slippage and cost overrun. After testing the build, customers rate the software at the end of the first iteration and provide feedback.

Engineering

* In the next quadrant, prototypes are built and tested. This step includes architectural design, module design, physical product design and final design. Convert the proposals made in the first two quadrants  into usable software.
* This phase also includes the actual implementation of features in a project, which are verified by performing testing.

Customer Evaluation

* In this phase, the customer evaluates the software and gives feedback. The team prepares for the next phase of the planning process. The next phase of the planning process is known as the “spiral” phase.
* During the “spiral” phase, the team determines the order of events in the current state of the product and then follows these events up with a “revision” phase to “Revise” the current state of the product so that it is ready for production. The “revision” phase is also called the “reproduction” phase, one of the most important aspects of the planning process.

**Write agile manifesto principles**

* Individuals & Interactions
* Working Software
* Customer Collaboration
* Responding to change

Individuals & Interactions

* The key aspect of this value is communication to enhance interactions among the project management team instead of relying on processes and tools.
* When the team can collaborate effectively, it simplifies the whole process, making the project’s objectives achievable with ease. On the other hand, focusing on techniques and tools can make the team fail to meet the client’s project requirements, and this will move to users, and this can make the final product have a negative reception in the market.

Working Software

The agile manifesto focuses on the time matter instead of documentation processes. This enables the project management team to plan and work on the project effectively by providing software to get them started. This ensures the timely delivery of a high-quality product.

Customer Collaboration

In the agile manifesto, customers are encouraged to participate in the project development processes to ensure the final product meets the user’s needs. This is contrary to contract negotiations where the project manager and the customer only discuss the scope of the project when starting, thus ruling out the idea of customer collaboration during the different phases of product management.

Responding to change

Ideally, the basic software development process life cycle suggests following a specified plan instead of working on the required changes to minimize costs. In the agile manifesto, however, the life cycle of the product development process keeps changing based on a number of elements, especially the customer’s views. This value ensures the final product meets all the needs of end users, ensuring it meets the business objectives.

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

* 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. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.
* Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.
* Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

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 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.
* Transfer of technology to new team members may be quite challenging due to lack of documentation.

**Draw Usecase on Online book shopping**

Select Book

Add to Cart

Proceed to Checkout

Enter Payment Details

Confirm Purchase

Receive Confirmation

Open Website

Browse Books Catalog

**Draw Usecase on online bill payment system (paytm)**

Confirmation Message

UPI Pin

Paytm App

Electricity Bill Payment

Selects Electricity Board

Payment Method

Receive Bill Information

Payment Retrieves

Enter Consumer Number

**Draw usecase on Online shopping product using COD.**

Customer Receives

Customer’s Address

Online shopping Platform

Search the product

View Product

Conformation Notification

Customer Reviews

Cash On Delivery

Selects Product

**Draw usecase on Online shopping product using payment gateway**

Payment Confirmation

Enter UPI

Customer Reviews

Conformation Notification

Customer’s Address

Customer Receives

Online shopping Platform

Search the product

View Product

Select Payment Method

Selects Product

**Draw usecase on Online Money Transfer from One account to another**

Open Website

Login

Select Transfer Option

Receive Transfer Confirmation

Process Transfer

Confirm Details

Enter Transfer details