Q-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 development and lifecycle management of an application or piece of software.

SDLC Phases

|  |  |
| --- | --- |
| Requirements Collection/Gathering | Establish Customer Needs |
| Analysis | Model And Specify the requirements- “What” |
| Design | Model And Specify a Solution – “Why” |
| Implementation | Construct a Solution In Software |
| Testing | Validate the solution against the requirements |
| Maintenance | Repair defects and adapt the solution to the new requirements |

Requirement Gathering

• Features

• Usage scenarios

• Requirements will Change!

• Inadequately captured or expressed in the first place

• User and business needs change during the project

• Functional and Non-Functional

• Three types of problems can arise:

**Lack of clarity**: It is hard to write documents that are both precise and easy-toread.

**Requirements confusion:** Functional and Non-functional requirements tend to be intertwined. Requirements Amalgamation: Several different requirements may be expressed together.

**Requirements Amalgamation**: Several different requirements may be expressed together.

Requirement Gathering

**Functional Requirements**: describe system services or functions.

* Compute sales tax on a purchase
* Update the database on the server

**Non-Functional Requirements:** are constraints on the system or the development process.

* Non-functional requirements may be more critical than functional requirements.

Q-2 What is software 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.
* A customer satisfied with the quality of a product will remain loyal and wait for new functionality in the next version.
* Regression Testing
* Unit Testing Application T
* Stress Testing

Q-3 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

Advantages of agile method :

1. Frequent delivery
2. Face to face communication with the customer
3. Last time
4. Adaptability

Disadvantages of agile method:

1. less documentation
2. maintenance problem

Use case:

Extend relashionship = optional

Inclide relashionship = maintary

Q-4 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 IEEE 830-1998.
* This standard describes possible structures, desirable contents, and qualities of a software requirements specification.

Q-5 What is oops

* Object is derived from abstract data type
* Object-oriented programming has a web of interacting objects, each house-keeping its own state.
* An object is like a black box.
* Identifying objects and assigning responsibilities to these objects.
* The internal details are hidden.

Q-6 Write Basic Concepts of oops

* Object
* Class
* Encapsulation
* Inheritance
* Polymorphism

Overriding

Overloading

* Abstraction

Q-7 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.

Q-8 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.
* We do not actually buy these blueprints but the actual objects.

Q-9 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).

Q-10 What is inheritance

* Abstraction is the representation of the essential features of an object.
* These are ‘encapsulated’ into an abstract data type. Data abstraction refers to, providing only essential information to the outside word and hiding their background details, i.e., to represent the needed information in program without presenting the details.
* For example, a database system hides certain details of how data is stored and created and maintained. Similar way, C++ classes provides different methods to the outside world without giving internal detail about those methods and data.
* In plain English, abstract means a concept or idea not associated with any specific instance and does not have a concrete existence.

Q-11 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.
* Poly refers to many.
* That is a single function or an operator functioning in many ways different upon the usage is called 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).
* The ability to change form is known as polymorphism.
* There is two types of polymorphism in
* Java Compile time polymorphism(Overloading)
* Runtime polymorphism(Overriding

Q-12 Write SDLC phases with basic introduction

SDLC Phases

|  |  |
| --- | --- |
| Requirements Collection/Gathering | Establish Customer Needs |
| Analysis | Model And Specify the requirements- “What” |
| Design | Model And Specify a Solution – “Why” |
| Implementation | Construct a Solution In Software |
| Testing | Validate the solution against the requirements |
| Maintenance | Repair defects and adapt the solution to the new requirements |

Requirement Gathering

• Features

• Usage scenarios

• Requirements will Change!

• Inadequately captured or expressed in the first place

• User and business needs change during the project

• Functional and Non-Functional

• Three types of problems can arise:

**Lack of clarity**: It is hard to write documents that are both precise and easy-toread.

**Requirements confusion:** Functional and Non-functional requirements tend to be intertwined. Requirements Amalgamation: Several different requirements may be expressed together.

**Requirements Amalgamation**: Several different requirements may be expressed together.

Requirement Gathering

**Functional Requirements**: describe system services or functions.

* Compute sales tax on a purchase
* Update the database on the server

**Non-Functional Requirements:** are constraints on the system or the development process.

* Non-functional requirements may be more critical than functional requirements.

Q-13 Explain Phases of the waterfall mode

* Business Requirement Analysis: This is the first phase in the development cycle where the product requirements are understood from the customer perspective. This phase involves detailed communication with the customer to understand his expectations and exact requirement. This is a very important activity and need to be managed well, as most of the customers are not sure about what exactly they need. The acceptance test design planning is done at this stage as business requirements can be used as an input for acceptance testing.
* System Design (System Requirement): Once you have the clear and detailed product requirements, it’s time to design the complete system. System design would comprise of understanding and detailing the complete hardware and communication setup for the product under development. System test plan is developed based on the system design. Doing this at an earlier stage leaves more time for actual test execution later.
* Architectural Design (Technical Specification): Architectural specifications are understood and designed in this phase. Usually more than one technical approach is proposed and based on the technical and financial feasibility the final decision is taken. System design is broken down further into modules taking up different functionality. This is also referred to as High Level Design (HLD). The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood and defined in this stage. With this information, integration tests can be designed and documented during this stage.
* Module Design (Program Specification): In this phase the detailed internal design for all the system modules is specified, referred to as Low Level Design (LLD). It is important that the design is compatible with the other modules in the system architecture and the other external systems. Unit tests are an essential part of any development process and helps eliminate the maximum faults and errors at a very early stage. Unit tests can be designed at this stage based on the internal module designs

Q-14 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. Following are the typical uses of Spiral model: 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. Requirements are complex and need evaluation to get clarity.
* 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

Q-15 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.

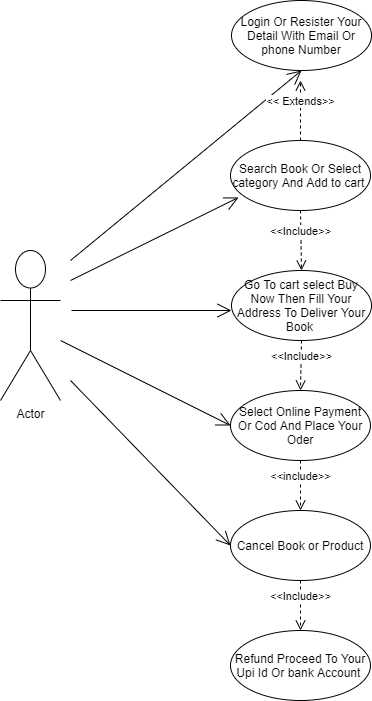
**Pros**

* 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

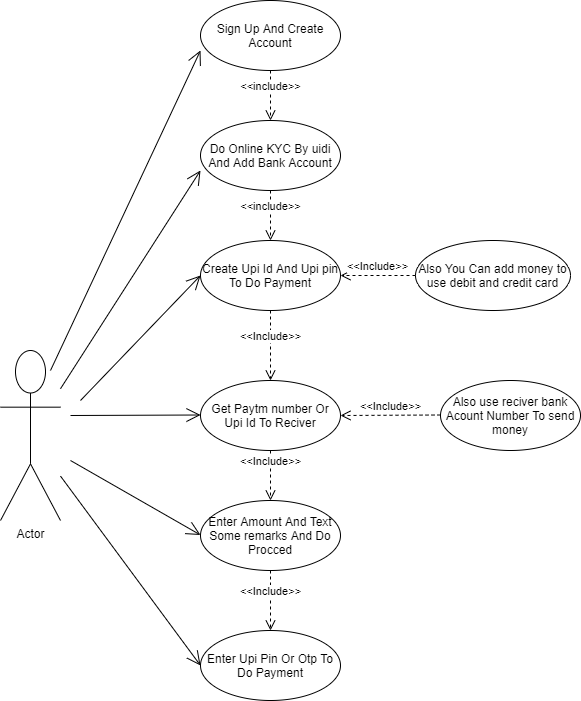
**Cons**

* Not suitable for handling complex dependencies. More risk of sustainability, maintainability and extensibility. A
* n 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.

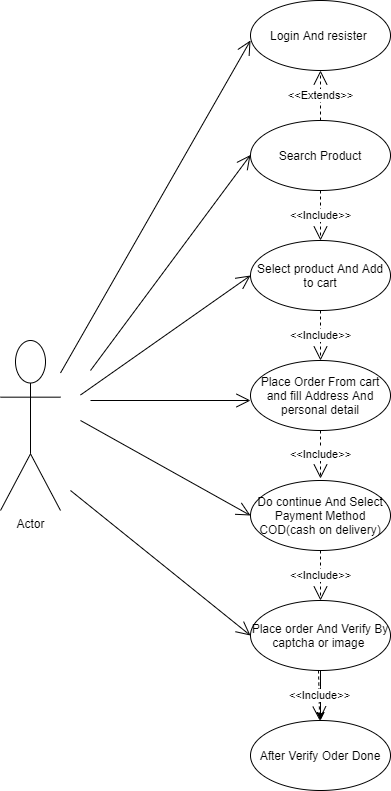
Q-16 Draw Use case on online book shopping



Q-17 Draw Use case on online bill payment system (paytm)



Q-18 • Draw use case on Online shopping product using COD



Q-19 Draw use case on Online shopping product using payment gateway.

