1). What is SDLC?

Software development lifecycle is a methodology or step-by-step approach to developing software with high quality at the lowest cost and within the minimum possible time by defining the phases: -

* Planning/ Requirement gathering.
* Analysing the requirements and preparing written documentation.
* Designing the architectural planning model.
* Implementing the planning model into the actual product along with source code.
* Testing the actual product along with features with every aspect from functional to non-functional.
* Maintenance is a part of SDLC where after deployment maintainability phase takes place to optimize and enhance the software with new updating.

**In a nutshell**

The software development lifecycle is a structure imposed on the development of a software product by defining various development phases.

2). What is software testing?

Software testing is a process of evaluating the system software or its components with the intent to find out whether it satisfies the specific requirement or not.

Software testing is a process to detect whether actually developed software satisfies the desired requirements or not.

Software testing is a process to detect any defects, bugs, errors, or any missing information in the software system developed as per specific requirements or not.

Software testing is done by a testing team, in some organizations there is an individual department for testing. Testing is required in every field where software is connected. For example: -

* Transport systems, Solar systems, telephone systems, Dewey decimal systems, Weapons systems, ecological systems, Space systems, and many other fields where software systems are connected.

**In a nutshell**

Software testing is a process of analyzing a software item to detect the differences between existing and required conditions and to evaluate its features in the software.

3). What is agile methodology?

It is a combination of iterative and incremental process models to develop system software that focuses on process adaptability, and customer satisfaction by giving rapid delivery of a working software product.

* Iterative: - Repeated Cycles.
* Incremental model-: Building in small proportions at a time.

**Pros**

* Very realistic approach.
* Rapid Delivery of working software model.
* Little or no planning is required.
* Promotes teamwork and cross-training.
* Functionality can be developed rapidly.
* Suitable for fixed requirements and changing requirements.
* Resources requirements are minimal.
* Gives flexibility to developers.

**Cons**

* Very highly dependent on Customer interactions.
* Highly individual dependency.
* Less documentation will not be carrying detailed information for future changes if required.
* More risk of sustainability without proper documents, maintainability with fewer documents, and lack of extensibility of coming up with new features while being in existing features.
* Not suitable for smaller projects.
* Not suitable for handling complex dependencies.

4). What is SRS?

A software requirement specification is a complete description of the behaviour of the system to be developed.

* Includes a set of use cases describing the user interactions with the software.
* Use cases also known as functional requirement in addition to supplementary requirements included in SRS that is non-functional requirements.
* Functional means direct interaction of users with software (for example login page, home page, cart, log out), and non-functional means indirect interaction of users with software (for example performance, load, stress, quality, design constraints, and much more).

**In a nutshell**

Software requirement specification is a written form of documentation that is noted right from the initial stage of requirement till it gets deployed. This mechanism of documentation will be very useful while maintaining the phase for optimization and enhancement of deployed software.

5). What is oops?

Object-oriented programming is used to identify objects and assign responsibilities to these objects.

Object-oriented programming is used to structure the software program into simple reusable code. OOP = Object + Class where object is the basic unit of OOP which is accessed by its properties called data member & member function(methods). It creates a memory for the class.

OOP is viewed as a collection of objects. It is used to structure the software program into simple re-usable code. Here it is referred as functional testing or Black Box testing.

**In a nutshell**

Object-oriented programming language is used to communicate with computers by assigning objects with responsibility for the execution of specific tasks.

6). Write Basic Concepts of oops?

Object-oriented programming is divided into concepts: -

1. Object: - Basic unit used that is data + function/methods.
2. Class: - Is a blueprint or a template to describe the properties and behaviour of the objects.
3. Encapsulation: -Hiding data + Binding data in a capsule form.
4. Inheritance: - One class inheritance the characteristics of another class.
5. Polymorphism known for having many forms. Polymorphism includes
6. Overriding: - Runtime.
7. Overloading. Compile time.
8. Abstraction: - Representation of the essential features of an object. “Encapsulated into an abstract data type”.

**In a nutshell**

OOP has two steps first one is making classes: that consist of data and its methods for creating, extending, and reusing abstract data types and the Second one is making objects interact: Creating objects from abstract data types and defining their relationships.

7). What is an object?

The object is everything or anything in the world such as A flower, tree, chair, bench, student, professor, city, country, or university can become an object. Objects can be tangible things, interactions, incidents, roles, and also specifications.

Object = Data + methods/functions.

For example, there are students in class 10. When school starts, everyone is asked to fill out the basic detail form with names, addresses, and contact numbers. So whole collection of information received from class in the form of forms is called data. While adding data in schools by giving them roll numbers, Standard 10, division C, and interacting with that class students by their roll numbers assigned this interaction is called methods.

So here Object is Students = Basic detail form is data + Roll numbers/ Division with the standard is methods.

* Basic unit of OOP = Object.
* Object = Data + Function.
* Object = Code + Methods. For example: A car is an object. Car has attributes such as weight, color and methods drive, brake, accelerate.

**In a nutshell**

* An object is the basic unit of OOP which is accessed by its properties called data member and member function. It creates the memory for the class.

8). What is class?

Class defines as a blueprint of an object. This doesn’t define any data, but it does define what class means, that is what an object of class consists of, and operations need to perform on such object.

* A class is like an object, constructor or blueprint for creating objects.
* A class represents an abstraction of the object and properties and behavior of that object.
* Class = Blueprint of object.
* Class = Defining object.
* Class = Template for an object.
* Objects can be many for a class.
* Objects have their existence, but class has no existence.
* Example = car or laptop is an object. When class = blueprint is created then designing formed on initial stage then the actual car or laptop built based on that class. Here we do not buy blueprints but actual objects like cars or laptops.

**In a nutshell**

Class describing properties and behavior but without actual existence.

9). What is encapsulation?

Encapsulation is the fundamental of OOP that is bundling the data with the methods and operate on that data.

Encapsulation is used to hide the values or state of a structure data object inside a class, preventing unauthorized parties' direct access to them.

Main Advantage of Encapsulation

* Security of Data.
* Protects an object from unwanted access by clients.
* Allows access to a level without revealing the complex details about below that level.

Encapsulation binding the data with its related functionalities. Functionalist = methods + data. Functionality = Code + variable. Base is called class = variables and method in one place.

Java Encapsulation = hide (restrict access), to critical data members in your code which improves security.

Encapsulation

If a data member is declared “private” then it can only access to same class. No outside class can access data member (variable) of that class.

If you need to access these variables you have to use “public”, “getter” and “setter” methods.

**In a nutshell**

In Java encapsulation is a process wrapping up data and behavior of an object in a single unit, this single unit is a class for property data and method behavioral.

10). What is polymorphism

Most important concept in OOP is Polymorphism. In simple terms Poly means having many forms. The ability to change form is polymorphism. It allows different objects to respond to the same message in their own behavior.

* Polymorphism = Single function operating in many ways.
* Polymorphism = operator performing as per the ability of changing form.
* For example: Goal is communication and if you have smart phone then the mode you select is up to you on how to communicate. This mode can be a call, text, video call, audio call. Here Goal of communication can be solved in various ways as approach is different is called polymorphism.
* Two types of Polymorphisms are there: -

1. Compile time (overloading).
2. Running time (over riding).

Defining overloading when existing function or operator is made to operate on new data type, is said to be overloaded. In this method the class appears to have the same name but has a different signature.

For example: If we take the example of bank account. Then Account can be Parent class and its function that is deposit and withdraw can be child class. Here parent class having same function so there is no need to define it separately.

* Withdraw method for saving account is also called privilege class, when extending into check balance and then withdraw class.
* But when withdraw class uses check bank balance and then withdraw class is said to be privilege method or privilege class that is over riding.
* As custom withdraw action takes place is called privilege class and overdraft facility.

Defining overriding we will be able to override the methods of the base class so that meaningful implementation of the base class method can be derived class.

In java signature of method changes that include parameters type, order of parameters.

Overriding = can widely extend accessibility. But not narrow: if it is private in base class then child class can make it public but not vice versa. Example; Doctor -> works at hospital -> Treat Patient () -> then Treat Patient () -> Incision () Surgeon. Then in this first is Treat Patient () then overriding into Incision () surgeon.

**In a nutshell**

Polymorphism method recall name should be same in single class but its behaviour (Argument and data type) is different.

11). Write SDLC phases with a basic introduction.

Software development lifecycle is a methodology or step-by-step approach to producing software with high quality, and the lowest cost within a limited possible time by defining phases as follows: -

1. Requirement Gathering/ Planning: - It is a process of gathering information in written form supplemented by use cases, diagrams, and tables expressed in UML i.e., Unified modeling language. It's important to collect requirements with clarity without ambiguity.

Three problems occur while gathering information: -

1. Lack of clarity: The requirement has to be clear, and precise.
2. Lack of confusion: Clarity has to be there for functional and non-functionality.
3. Lack of amalgamation: Several different requirements may be expressed in one.
4. Analysis: - Analyzation of requirements is important where customer perspectives on their problems, deliverables, and end product are analyzed in the document.
5. Document clearly and precisely what is to be built.
6. Architectural document clarifying the behavior, interfaces, and components.
7. Engineering detailed document clarifying memory size, platforms, distributed architectural layering, interfaces, data structures, computer languages, machines to be used, algorithms and many other detailing comes under this phase.
8. Designing: - As the name says designing or making a blueprint on paper comes under this phase by referring to the analyzed document, designing architecture takes place.
9. Architectural team converts typical scenarios into a test plan.
10. Implementation plan.
11. Critical priority analysis.
12. Performance Analysis.
13. The Test plan all comes under this phase.
14. Implementation: - Here are references taken from the analysis phase and designing architectural plan the team builds the software either from scratch or from its composition.
15. There is still room for innovation, change, and flexibility while implementing involving source code.
16. This phase deals with issues like performance, baselines i.e., deliverable results, libraries i.e., pre-written code, class, data structure, procedures, scripts, configuration data, and much more.
17. The end result is the product itself i.e., deliverable working product.
18. Testing phase: - This phase requires identifying whether deliverable working software is performing as per requirement or not, whether the quality of the software is good or not, Functionalities which are required by the customer are working or not.
19. Quality is an important attribute to distinguish in a software developed indicating the level of excellence.
20. Testing is an approach that says a “Fresh eye or testing team will discover obvious errors, bugs much faster than the person who develops a software or developer who is reading or re-reading the same material again and again.
21. Testing is usually based on regression technique split into several major focuses namely internal, unit, application, and stress.
22. Maintenance phase: - In software engineering maintenance is one of the activities and process of enhancing and optimizing deployed software or delivered software as well as fixing defects.
23. The maintenance phase comes after software deployment.
24. Developing teams have certain mechanisms to document and track defects and deficiencies.
25. Also called configuration and version management.
26. Also says as Redesigning and refactoring.
27. Updating all analysis, designing, and user documentation.
28. Repeatable, automated tests enable evolution and refactoring.

Maintenance is a process of changing software after it has been deployed. There are three covering in this phase

1. Corrective maintenance: - Identifying and repairing defects.
2. Adaptive maintenance: - adapting the Existing solutions to the new platform.
3. Perfective maintenance: - implementing the new requirements.

**In a nutshell**

Software development lifecycle is a structure imposed on the development of a software product that defines the process for various phases into step by step implementation towards software building.

12). Explain the Phases of the waterfall model

The waterfall model is a classical software lifecycle model developed into a step-by-step waterfall between the various development phases.

The phases of the waterfall model approach are set by set into waterfall, where one phase is completed then only, we approach for next phase. Due to this end goal, one phase will be clear at a time.

1. Suitable for smaller projects where requirements are frozen and stable.
2. Product definition is stable.
3. Due to its rigidness in phases is it known as classical software lifecycle model.

Phases of the waterfall model start with

**In a nutshell**

The waterfall model works for small, simple, precise projects. Where every phase completion is clearly understood and completed then the next phase works on the same project there is no going upwards option in this model that’s why the name says waterfall model.

13). Write phases of the spiral model.

It's Very widely used in the software industry as it synchronized with the natural development process of any product where learning with maturity also involves because of prototype due to which minimum risk involved for customers and for business firms.

There are Four important phases in this model

1. Planning/ requirement gathering/ Feasibility study i.e., detailed analysis of critical aspects in the project and analysis will it be successful or not.
2. Determination of objectives, alternatives, all specifications come under the planning phase.
3. Risk analysis.
4. Alternatives analysis and identifying risk factors and resolution, as some risks will increase cost and some will increase delay for the software.
5. Engineering / Coding/developing the software.
6. Here evolving process starts where coding is done by the DEV team and the end result is the first prototype, a working model as per specific requirements.
7. Customer evaluation/ feedback/ Testing.
8. Here testing team will find bugs, errors, missing requirements, and even customer feedback is also included in this due to which it will be friendly for the customer to analyse the required software. Alpha demo testing takes place by the team or we can say User acceptance testing.

**In a nutshell**

The spiral model is widely used due to the first prototype, and the alpha demo was done by testing the software as per need, risk analysis will be helpful for projects that are budget constraints and highly covered under high risk.

14). Write agile manifesto principles

The agile model is a combination of both iterative and incremental process models. The development of the product takes place in repeated cycles i.e., iterative and incremental build where small proportions develop in limited time.

Agile works on four core values: -

1. Individuals and interactions, over processes and tools.

Suppose the team finds any issue in software then they will not find new processes or tools for resolution as in agile it is highly dependent on client interaction of what they want, the team discuss with client requirement then discusses with the team manager, and leader and try to resolve the issue.

1. Working software, over comprehensive documentation.

Client satisfaction is with working software, earliest possible time the working model will be beneficial for a client over documentation, documentation will be helpful while maintenance part arises but agile believes in working deliverable software and do the updating as per client requirement in incremental build. Documentation will be less in agile working software will be the priority.

1. Customer collaboration, over contract negotiation.

Customer collaboration is important, the interaction of what they want, and their satisfaction is the utmost priority, and requirement fulfilment is more important than negotiating about the budget. If the client was satisfied then negotiation got lined up as per our requirement. Client's requirement is satisfied then only negotiation will be helpful.

1. Responding to change, over following a plan.

Responding to change is important, you are following a specific build to complete the task you made a plan for but if the requirement changes suddenly then you have to respect the client's new requirement, at that time following the same plan will not work for the client, changes adaptability is important as per new requirement.

**In a nutshell**

The agile model is the best model at the current time because it focuses on adaptability, flexibility, and rapid delivery of a working software model, that satisfies the client and overall works very well if we compare it to other models. East or West Agile is the best.

15). Explain the working methodology of the agile model and also write the pros and cons.

The agile methodology works in a way by breaking it into several phases. Agile is based on iterative and incremental models, where each increment or build is getting tested and rapidly delivered software is getting ready.

Pros of agile model.

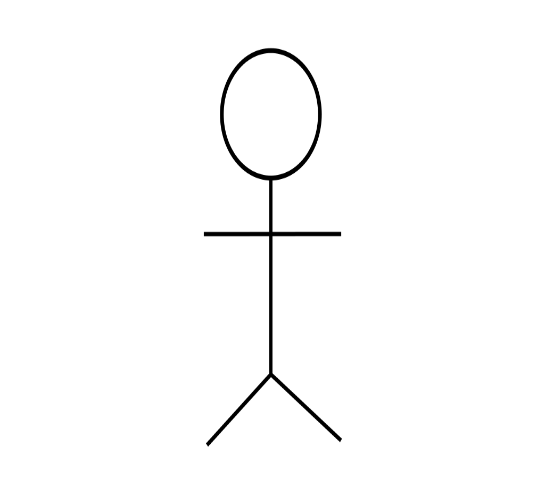
1. Very realistic approach due to adaptability, flexibility and
2. Little or no planning is required.
3. Less documentation required.
4. Functionalities developed rapidly.
5. Each increment is getting tested which is time-saving.
6. Rapid delivery of working software module.
7. Resources require minimum.
8. Works for Larger projects.
9. Promotes teamwork and cross-training.
10. Gives flexibility to developers.
11. Suitable for fixed and changing requirements.

Cons of an agile model.

1. Highly dependent on client interactions.
2. Minimum documentation generated.
3. More risk of sustainability maintainability and extensibility.
4. Very high individual dependency.
5. Not suitable for handling complex dependency.
6. Not useful for smaller projects.
7. Transfer of technology to new team members will be quite challenging due to lack of documentation.

16). Draw use case on Online shopping products using COD.

Amazon Shopping App.



Actor

17). Draw use case on Online shopping product using payment gateway.

Flipkart Shopping card payment.

