1. What is SDLC?

Def: 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.

1. What is software testing?

Def: The process consisting of all lifecycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfied requirements, to demonstrate that they are fit for purpose and to detect defects.

1. What is agile methodology?

Def: Agile model is a combination of iterative and incremental process model with focus on process adaptability and customer satisfaction by rapid delivery of working software product.

1. What is SRS?

Def: It is software Requirement Specification .it is a complete description of the behaviour of the system to be developed.

1. What is OOPS?

Def: an object oriented programming language is one which easily supports object orientation.

1. Write a basic concept of OOPS?

Def: concepts: 1. Object

2. Class

3. Encapsulation

4. Inheritance

5. Polymorphism

-overloading

-overriding

6. Abstraction

1. What is Object?

Def: The object represents an individual, identifiable item, unit, or entity, either real or abstract, with well -defined role in the problem domain.

1. What is Class?

Def: it is define as a blueprint for an object.

1. What is Encapsulation?

Def: encapsulation is the process of including in an object everything it needs hidden from other objects.

1. What is Inheritance?

Def: inheritance means that one class inherits the characteristic of another class.

1. What is Polymorphism?

Def: 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.

1. Write SDLC phases with basic introduction.

Ans: 1. Requirement Gathering: Establish customer needs.

2. Analysis: Model and specify the requirements-”what”

3. Design: model and specify a solution-”why”

4. Implementation: construct a solution in software.

5. Testing: validate the solution against the requirement.

6. Maintenance: Repair defects and adapt the solution to the new requirements.

1. Explain phases of the waterfall model.

Ans: The classical software lifecycle models the software development as a step-by-step “waterfall” between the various development phases.

1. Requirement collection
2. Analysis
3. Design
4. Implementation
5. Testing
6. Maintenance
7. Write phases of spiral model.

Ans: 1. Planning

2. Risk analysis

3. Engineering

4. Customer evaluation

1. Write agile manifesto principles.

Ans: 1. Individual & Interaction

2. Working software

3. Customer collaboration

4. Responding to change

1. Explain working methodologies of agile model and also write pros and cons.

**Ans:**

* 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.

Pros

* Is a very realistic approach to software development
* Functionality can be developed rapidly and demonstrated.
* Promotes teamwork and cross training.
* 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.