*Software Testing Assignment*

Module – 1 (Fundamental)

1) What is SDLC?

Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software. SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step.

The goal of the SDLC life cycle model is to deliver high-quality, maintainable software that meets the user’s requirements. SDLC in software engineering models outlines the plan for each stage so that each stage of the software development model can perform its task efficiently to deliver the software at a low cost within a given time frame that meets users’ requirements.

2) What is Software Testing?

Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases. The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. The article focuses on discussing Software Testing in detail.

Software Testing is a method to assess the functionality of the software program. The process checks whether the actual software matches the expected requirements and ensures the software is bug-free. The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirements. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

Software testing can be divided into two steps:

1. Verification: It refers to the set of tasks that ensure that the software correctly implements a specific function. It means “Are we building the product, right?”.
2. Validation: It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. It means “Are we building the right product?”.

3) What is Agile Methodology?

Agile Software Development is a software development methodology that values flexibility, collaboration, and customer satisfaction. It is based on the Agile Manifesto, a set of principles for software development that prioritize individuals and interactions, working software, customer collaboration, and responding to change.

Agile Software Development is an iterative and incremental approach to software development that emphasizes the importance of delivering a working product quickly and frequently. It involves close collaboration between the development team and the customer to ensure that the product meets their needs and expectations.

The Agile Software Development Process:

1. Requirements Gathering: The customer’s requirements for the software are gathered and prioritized.
2. Planning: The development team creates a plan for delivering the software, including the features that will be delivered in each iteration.
3. Development: The development team works to build the software, using frequent and rapid iterations.
4. Testing: The software is thoroughly tested to ensure that it meets the customer’s requirements and is of high quality.
5. Deployment: The software is deployed and put into use.
6. Maintenance: The software is maintained to ensure that it continues to meet the customer’s needs and expectations.

Agile Software Development is widely used by software development teams and is considered to be a flexible and adaptable approach to software development that is well-suited to changing requirements and the fast pace of software development.

Agile is a time-bound, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver all at once.

Principles of Agile:

1. The highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. It welcomes changing requirements, even late in development.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shortest timescale.
4. Build projects around motivated individuals. Give them the environment and the support they need and trust them to get the job done.
5. Working software is the primary measure of progress.
6. Simplicity the art of maximizing the amount of work not done is essential.
7. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
8. By the amount of work that has been finished, gauge your progress.
9. Never give up on excellence.
10. Take advantage of change to gain a competitive edge.

4) What is SRS?

Software Requirement Specification (SRS) Format as the name suggests, is a complete specification and description of requirements of the software that need to be fulfilled for the successful development of the software system. These requirements can be functional as well as non-functional depending upon the type of requirement. The interaction between different customers and contractors is done because it is necessary to fully understand the needs of customers. Depending upon information gathered after interaction, SRS is developed which describes requirements of software that may include changes and modifications that is needed to be done to increase quality of product and to satisfy customer’s demand.

Uses of SRS document

• Development team require it for developing product according to the need.

• Test plans are generated by testing group based on the describe external behaviour.

• Maintenance and support staff need it to understand what the software product is supposed to do.

• Project manager base their plans and estimates of schedule, effort and resources on it.

• customer rely on it to know that product they can expect.

• As a contract between developer and customer.

• in documentation purpose.

5) What are OOPs?

Object-Oriented Programming or OOPs refers to languages that use objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

6) Write Basic Concepts of oops

1. Class
2. Objects
3. Data Abstraction
4. Encapsulation
5. Inheritance
6. Polymorphism
7. Dynamic Binding
8. Message Passing

1. Class:

A class is a user-defined data type. It consists of data members and member functions, which can be accessed and used by creating an instance of that class. It represents the set of properties or methods that are common to all objects of one type. A class is like a blueprint for an object.

2. Object:

It is a basic unit of Object-Oriented Programming and represents the real-life entities. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. An object has an identity, state, and behaviour. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other’s data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

3. Data Abstraction:

Data abstraction is one of the most essential and important features of object-oriented programming. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car, but he does not know about how on pressing the accelerator the speed is increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc in the car. This is what abstraction is.

4. Encapsulation:

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. In Encapsulation, the variables or data of a class are hidden from any other class and can be accessed only through any member function of their class in which they are declared. As in encapsulation, the data in a class is hidden from other classes, so it is also known as data-hiding.

5. Inheritance:

Inheritance is an important pillar of OOP (Object-Oriented Programming). The capability of a class to derive properties and characteristics from another class is called Inheritance. When we write a class, we inherit properties from other classes. So, when we create a class, we do not need to write all the properties and functions again and again, as these can be inherited from another class that possesses it. Inheritance allows the user to reuse the code whenever possible and reduce its redundancy.

6. Polymorphism:

The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. For example, A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So, the same person possess different behaviour in different situations. This is called polymorphism.

7. Dynamic Binding:

In dynamic binding, the code to be executed in response to the function call is decided at runtime. Dynamic binding means that the code associated with a given procedure call is not known until the time of the call at run time. Dynamic Method Binding One of the main advantages of inheritance is that some derived class D has all the members of its base class B. Once D is not hiding any of the public members of B, then an object of D can represent B in any context where a B could be used. This feature is known as subtype polymorphism.

8. Message Passing:

It is a form of communication used in object-oriented programming as well as parallel programming. Objects communicate with one another by sending and receiving information to each other. A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results. Message passing involves specifying the name of the object, the name of the function, and the information to be sent.

7) What is Object

An object is the basic building block of OOPs. An object can be considered a real-life entity having a state and behaviour. An object is an instance of a class, and you can create many objects of the same class.

Objects in OOPs are conceptually identical to real-world objects. From a Technical point of view, an entity with a clearly defined structure and behaviour is called an object. An object in OOPs can include:

o A Variable.

o A Data Structure.

o A Function or

o A method.

8) What is Class

A class is a blueprint for creating objects (a particular data structure), providing initial values for state (member variables or attributes), and implementations of behaviour (member functions or methods).

The user-defined objects are created using the class keyword. The class is a blueprint that defines a nature of a future object. An instance is a specific object created from a particular class. Classes are used to create and manage new objects and support inheritance—a key ingredient in object-oriented programming and a mechanism of reusing code.

9) What is Encapsulation

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). Encapsulation describes bundling data and methods that work on that data within one unit, like a class in Java. We often often use this concept to hide an object’s internal representation or state from the outside. This is called information hiding.

The general idea of this mechanism is simple. For example, you have an attribute that is not visible from the outside of an object. You bundle it with methods that provide read or write access. Encapsulation allows you to hide specific information and control access to the object’s internal state.

10) What is Inheritance

Inheritance is one of the core features of object-oriented programming. It’s a programming procedure that allows you to reuse code by referencing the behaviours and data of an object. In other words, a class that inherits from another class shares all the attributes and methods of the referenced class.

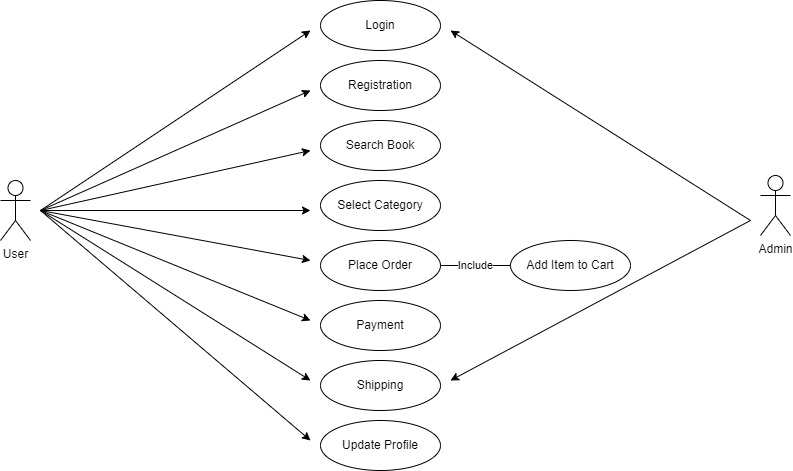
An inherited class is called a subclass or child class of the class it inherits from. And the class being inherited is called either a parent class, superclass, or base class. Inheritance is a core element of object-oriented programming that serves as a powerful instrument for reusing code.

11) What is Polymorphism

Polymorphism is one of the core concepts of object-oriented programming (OOP) that describes situations in which something occurs in several different forms. In computer science, polymorphism describes the concept that you can access objects of different types through the same interface. Each type can provide its own independent implementation of this interface.

You can perform a simple test to know whether an object is polymorphic. If the object successfully passes multiple is-a or instance of tests, it’s polymorphic. As described in our post about inheritance, all Java classes extend the class Object. Due to this, all objects in Java are polymorphic because they pass at least two instances of checks.

12) Draw Usecase on online book shopping



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



14) Write SDLC phases with basic introduction

The SDLC typically includes the following phases:

1. Requirements gathering and analysis: This phase involves gathering information about the software requirements from stakeholders, such as customers, end-users, and business analysts.

2. Design: In this phase, the software design is created, which includes the overall architecture of the software, data structures, and interfaces. It has two steps:

• High-level design (HLD): It gives the architecture of software products.

• Low-level design (LLD): It describes how each and every feature in the product should work and every component.

3. Implementation or coding: The design is then implemented in code, usually in several iterations, and this phase is also called as Development.

things you need to know about this phase:

• This is the longest phase in SDLC model.

• This phase consists of Front end + Middleware + Back-end.

• In front-end: Development of coding is done even SEO settings are done.

• In Middleware: They connect both the front end and back end.

• In the back-end: A database is created.

4. Testing: The software is thoroughly tested to ensure that it meets the requirements and works correctly.

5. Deployment: After successful testing, the software is deployed to a production environment and made available to end-users.

6. Maintenance: This phase includes ongoing support, bug fixes, and updates to the software.

15) Explain phases of the waterfall model

The Waterfall Model has six phases:

1. Requirements Gathering and Analysis: The first phase involves gathering requirements from stakeholders and analysing them to understand the scope and objectives of the project.

2. Design Phase: Once the requirements are understood, the design phase begins. This involves creating a detailed design document that outlines the software architecture, user interface, and system components.

3. Implementation and Unit Testing: The implementation phase involves coding the software based on the design specifications. This phase also includes unit testing to ensure that each component of the software is working as expected.

4. Integration and System Testing: In the testing phase, the software is tested as a whole to ensure that it meets the requirements and is free from defects.

5. Deployment: Once the software has been tested and approved, it is deployed to the production environment.

7. Maintenance: The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.

16) Write phases of spiral model

The Spiral Model is a risk-driven model, meaning that the focus is on managing risk through multiple iterations of the software development process. It consists of the following phases:

1. Planning

The first phase of the Spiral Model is the planning phase, where the scope of the project is determined and a plan is created for the next iteration of the spiral.

2. Risk Analysis

In the risk analysis phase, the risks associated with the project are identified and evaluated.

3. Engineering

In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.

4. Evaluation

In the evaluation phase, the software is evaluated to determine if it meets the customer’s requirements and if it is of high quality.

5. Planning

The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation.

The Spiral Model is often used for complex and large software development projects, as it allows for a more flexible and adaptable approach to software development. It is also well-suited to projects with significant uncertainty or high levels of risk.

The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below:

1. Objectives determination and identify alternative solutions: Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.

2. Identify and resolve Risks: During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.

3. Develop the next version of the Product: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

4. Review and plan for the next Phase: In the fourth quadrant, the Customers evaluate the so-far developed version of the software. In the end, planning for the next phase is started.

17) Write Agile manifesto principles

• Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

• Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

• Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

• Business people and developers must work together daily throughout the project.

• Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.

• The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

• Working software is the primary measure of progress.

• Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.

• Continuous attention to technical excellence and good design enhances agility.

• Simplicity—the art of maximizing the amount of work not done—is essential.

• The best architectures, requirements and designs emerge from self-organizing teams.

• At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

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

Agile project management is a collaborative, iterative project management approach that incorporates continuous testing and responsiveness to change.

It is essentially an incremental framework that allows teams to compile tasks into shorter sprints. Think of Scrum as the configuration and Sprint intervals as periods of work with set goals. The overall approach is known as “agile” – keeping everyone focused on speed without compromising quality.

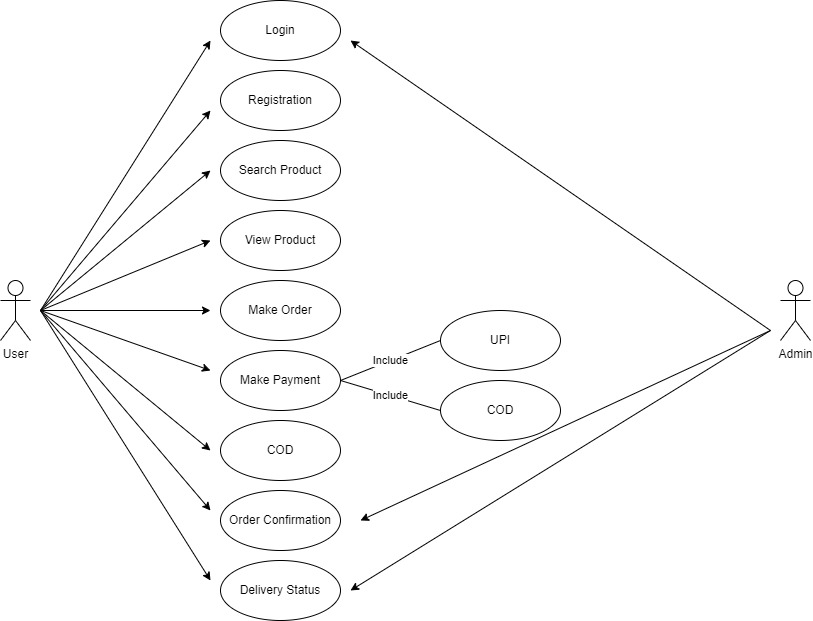
Companies have found success with the Agile approach, which allows them to quickly adapt and create products that are in demand on the market. It’s no surprise then it has become so popular in the software tech industry/development.

Ultimately, agile teams enjoy increased productivity and flexible working along with goodwill rewards, giving your organization exceptional results on their bottom line.

Agile working empowers teams with maximum flexibility. Not only does it give employees more control, but can also provide distinct benefits such as increased productivity and a happier workforce.

There’s no universal solution when introducing agile techniques into your organization, but setting clear boundaries and defining job roles ensures that everyone is clear on their responsibilities.

19) Draw Usecase on Online shopping product using COD.



20) Draw Usecase on Online shopping product using payment gateway.

