Software Testing Assignment

Module–1(Fundamental)

1. What is SDLC ?

→ SDLC is structure imposed on the development of a software product that defines the process for planning, implementation, testing , documentation, deployment and ongoing maintenance and support.

→ Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time possible. SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use.

→ 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.

1. What is software testing ?

→ Software Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not.

→ Software testing is a process of executing a program or application with the intent of finding the software bugs. It is a process used to identify the correctness, completeness, and quality of developed computer software.

→ Software testing is the act of examining the artifacts and the behaviour of the software under test by validation and verification.

Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation.

→ ‘The process consisting of all life cycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.’

→ It can also be stated as the process of validating and verifying that a software program or application or product:

• Meets the business and technical requirements that guided it’s design and development

• Works as expected

1. 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.

→ At the end of the iteration a working product is displayed to the customer and important stakeholders.

→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.

1. 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

* Costumer 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 and, at a minimum, answer the questions posed in the following listing:

• Operational distribution or deployment: Where will the system be used?

• Mission profile or scenario: How will the system accomplish its mission objective?

• Performance and related parameters: What are the critical system parameters to accomplish the mission?

• Utilization environments: How are the various system components to be used?

• Effectiveness requirements: How effective or efficient must the system be in performing its mission?

• Operational life cycle: How long will the system be in use by the user ?

**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.

• For Example: The following are the requirements of Google Email Service

• The system shall support the ability to receive emails

• The system shall support the ability to send emails

• The system shall support the ability to create new folders

• The system shall support the ability to filter emails in different folders

• The system shall support the ability to attach different kind of attachments

• The system shall support the ability to create and maintain address book

• The system shall support the ability to create unlimited user accounts with different email addresses

**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 behaviour . 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.

Non-functional requirements can be divided into following categories:

• Usability • Reliability • Performance • Security

1. 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.

1. Write Basic concept of oops .

• Object

• Class

• Encapsulation

• Inheritance

• Polymorphism : 1. Overriding 2. Overloading

• Abstraction

1. What is object ?

→ Object is instance if class. It is real world entity. object will occupies memory. Objects are structures that contain both data and procedures.

→ For example, a student is an object which has name and age.

1. What is class?

→ Class is Collection of data members and member functions.

→ Class doesn't occupies any memory.

→ Class is a blue print of object.

→ A class is a template that explains the details of an object

1. What is encapsulation?

→ Encapsulation means Wrapping up of data into single unit.

process of making class is encapsulation.

10. What is inheritance ?

→ Inheriting one class properties into another class is called Inheritance. Inheritance used to re the code duplication. Inheritance is a technique to re-use existing code again and again. Class that is inherited is called base class and a class which it inherits is called derived class.

11 . What is polymorphism ?

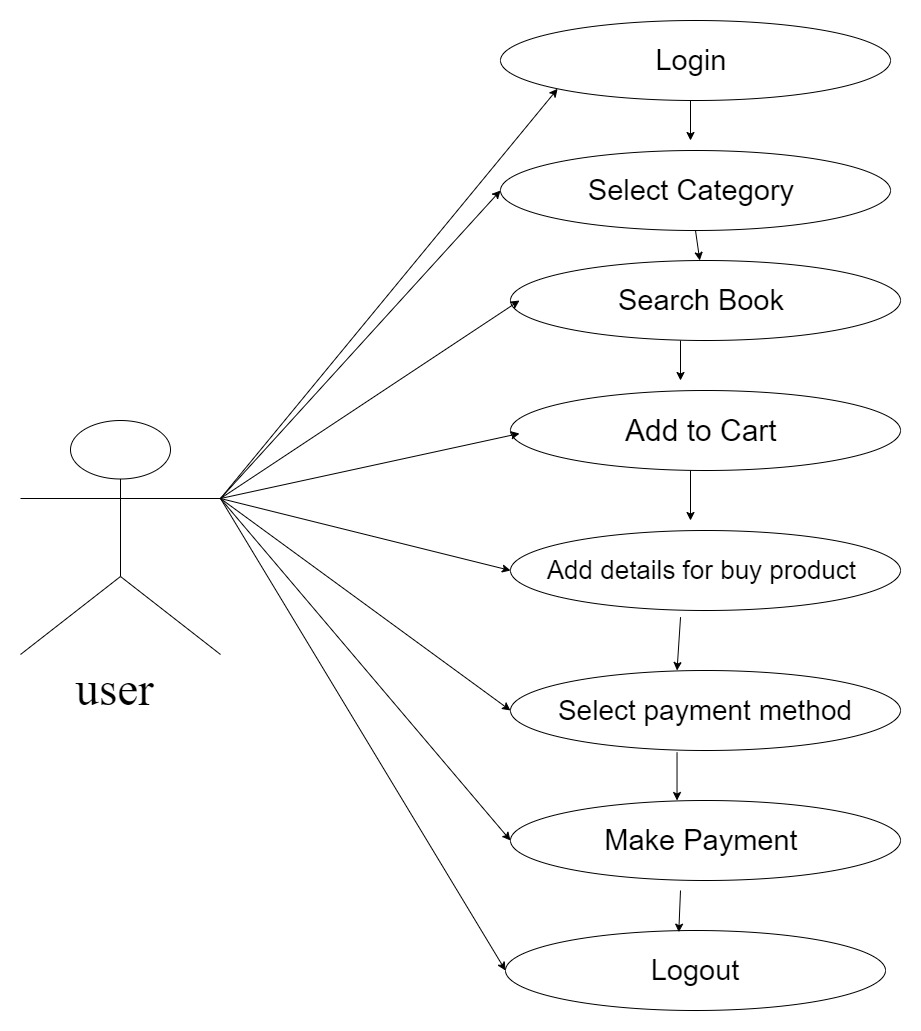
→ Polymorphism means many, which is requesting the same operation to perform differently.

→ poly means many

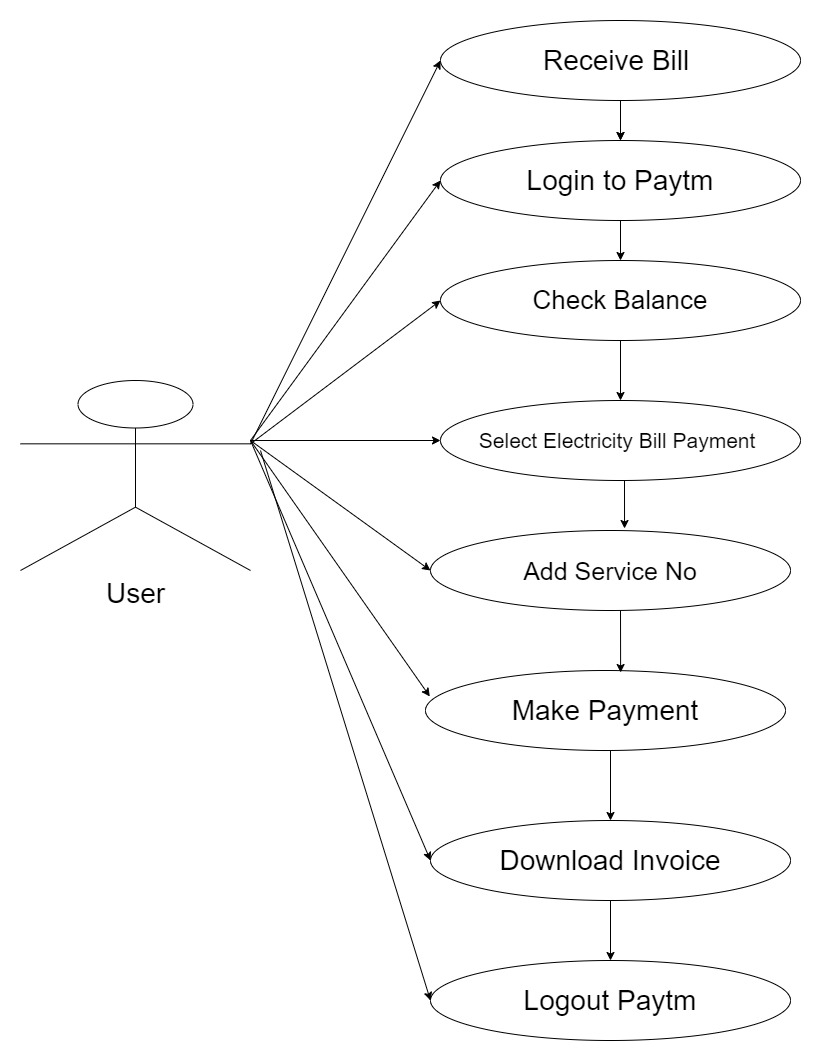
→ morphism means forms

→ so Polymorphism means one name having many forms.

12. Draw Usecase on Online book shopping.

→ 

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

→ 

14. Write SDLC phases with basic introduction

→ SDLC Phases Are

|  |  |
| --- | --- |
| 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 |

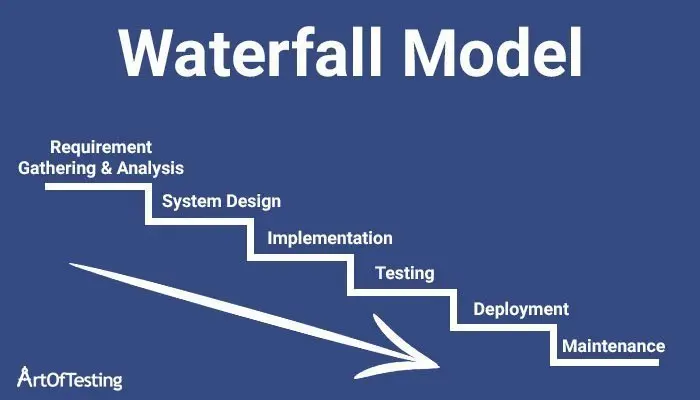
* Requirements Collection/Gathering
  + - Requirements gathering is one of the most essential parts of any project and adds value to a project on multiple levels. When it comes to smaller budgets, tighter timelines and limited scopes, exact documentation of all the project requirements become crucial.
    - Requirements gathering is easier said than done, it is generally an area that is given far less attention than it needs. Many projects start with basic lists of requirements only to find out down the line that many of the customers’ needs may not have been fully understood and implemented.
* Analysis
  + - In software testing, test analysis is the process of inspecting and analyzing the test artefacts in order to create test conditions or test cases. The goal of test analysis is to collect requirements and create test objectives so that test conditions can be established. As a result, it's also known as Test Basis

* Design
  + - The design phase of software development deals with transforming the customer requirements as described in the SRS documents into a form implementable using a programming language.
* Implementation
  + - 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.
* Testing
  + - Type of Testing
    - Regression Testing
    - Internal Testing
    - Unit Testing
    - Application Testing
    - Stress Testing
* Maintenance
  + - 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
    - 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
    - reengineering (redesigning and refactoring) , updating all analysis, design and user documentation , Repeatable, automated tests enable evolution and refactoring

15. Explain Phases of the waterfall model.

→ Phases of waterfall project management differ from one project to another. But generally, you can group the activities of the waterfall approach into five stages: **planning, design, implementation, verification, and maintenance**.

→ The waterfall model is one of the earliest models of the [Software Development Life Cycle](https://artoftesting.com/software-development-life-cycle-sdlc). The different phases in the waterfall model progress sequentially downwards, resembling a waterfall, hence the name–“Waterfallmodel”.  
  
Once a phase of the development cycle gets completed, there is no way to go back to that phase again in order to correct it or make any desired change to it. In this model, each phase must be completed before the next phase can begin. This is because the outcome of the previous phase will act as the input for the current phase.



**Phases of the Waterfall Model**

**Requirement Gathering & Analysis**

All the possible requirements of the system to be developed are captured in this phase. Here, the requirement feasibility analysis is done to ensure whether the requirements are feasible or not. In this phase, a Software Requirement Specification (SRS) document is created, containing both functional and non-functional requirements of the software to be developed.

**System Design**

In this phase, we gradually move forward to answer the ‘How’ of the system after answering the ‘What’ of the system in the previous phase. Here, we create design documents specifying the different modules/components of the system, their interfacing, data flow, etc.  
  
All of the data collected is stored in a document named Software Design Document (SDD). This document helps in establishing the architecture of software development projects.

**Implementation**

The implementation phase is also known as the coding phase. In this phase, based on the design documents created in the previous phase, the software product is developed. This phase makes use of a development environment, programming language, database, etc to create the software product.

**Testing**

In this phase, the software product developed in the previous phase is validated as per the functional and  [non-functional requirements](https://artoftesting.com/non-functional-requirements) specified during the requirement gathering and analysis phase.

**Deployment**

The deployment phase involves making the software live in the production/real environment after it tested for its tested thoroughly in the previous phase.

**Maintenance**

Over a period of time, a software product may require some updates in order to remain functional in the real-world environment. The maintenance phase takes care of this activity by timely tuning the software as per the requirement.

16. Write phases of spiral model.

→ **Spiral model** is one of the most important Software Development Life Cycle models, which provides support for **Risk Handling**. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a **Phase of the software development process.** The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.

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-

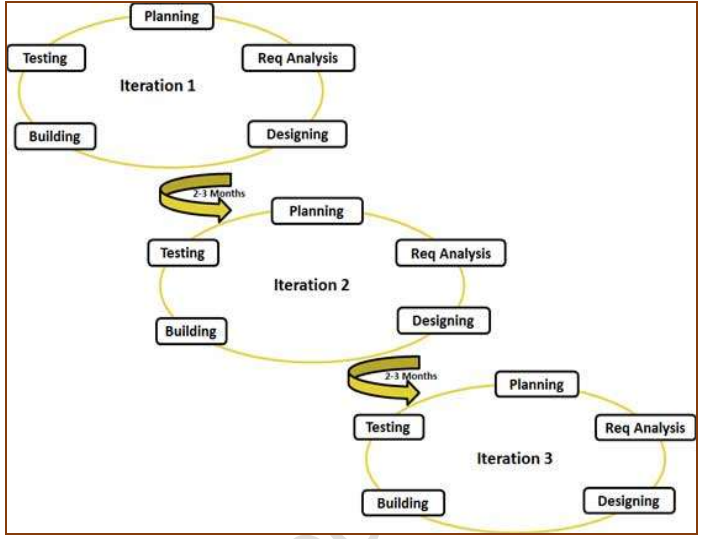
1. **Objectives determination and identify alternative solutions:** Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
2. **Identify and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

→ Agile Manifesto Principles • Individuals and interactions - in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming. • Working software - Demo working software is considered the best means of communication with the customer to understand their requirement, instead of just depending on documentation. • Customer collaboration - As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.. • Responding to change - agile development is focused on quick responses to change and continuous development.

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

→ Agile Model Work Flow



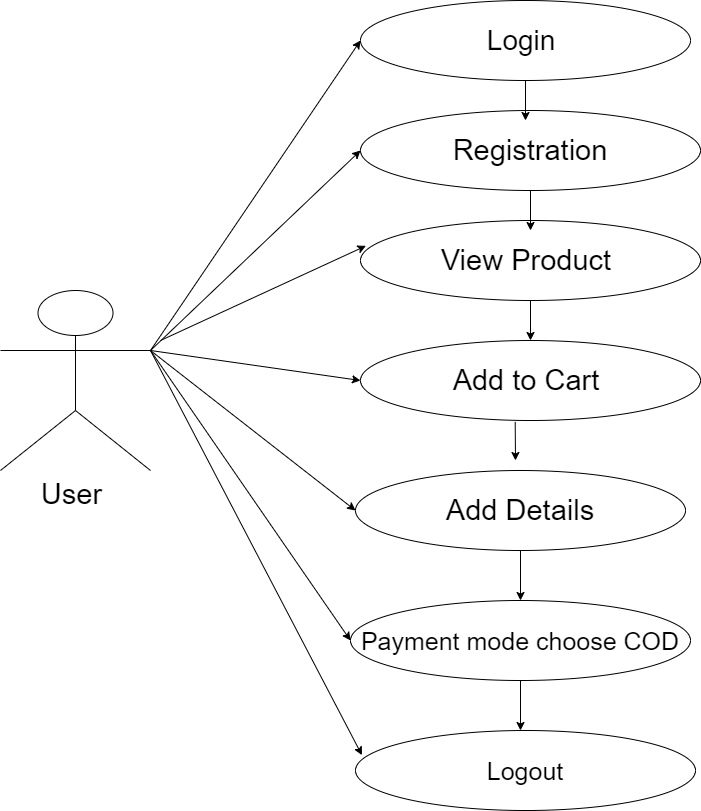
Pros

• 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

19. Draw usecase on Online shopping product using COD.

→ 

20. Draw usecase on Online shopping product using payment gateway.

→ 