**Manual Software Testing**

**Module-1: Software Testing Concepts**

* What is Software?
* Types of Software's?
* What is Software Testing?
* What is Software Quality?
* Project Vs Product
* Why do we need Testing?
* Error, Bug & Failure
* Why the software has bugs?
* SDLC & STLC
* Waterfall Model
* Spiral Model
* V-Model
* QA & QC & QE
* Different levels of Software Testing
* White Box & Black Box Testing
* Static Testing & Dynamic Testing > Verification & Validation
* System Testing Types
* GUI Testing
* Functional & Non-Functional Testing
* Test Design Techniques
* Re-Testing & Regression testing
* Exploratory Testing
* Adhoc Testing
* Sanity & Smoke Testing
* End-To-End Testing
* STLC (Software Testing Life Cycle)
* Use case, Test scenario & Test case
* Test Environment and Execution
* Defect Reporting
* Test Closure
* Test Metrics

**Module-2: Software Testing Project**

* Project introduction
* Understanding Functional Requirements from FRS
* Creating Test Scenarios
* Creating Test Cases
* Test Execution
* Bug Reporting & Tracking
* Test Sign off

**Module-3: Agile Testing + Jira Tool**

* Agile/Scrum Process:
* What is Agile
* What is Scrum / Scrum Team
* What is Sprint
* What is User Story
* How to give story points / How to estimate user story > What is Definition of Done and Definition of Ready
* Different Sprint Activities:
* Sprint Planning / Backlog Refinement / Sprint Review / Sprint Retrospective

**Jira Tool**

* How to install and configure JIRA tool > How to create an EPIC/User Stories in JIRA
* Creating sprints in Jira
* Sprint life cycle in JIRA
* Backlogs in JIRA
* Creating bugs in Jira
* How to write test cases in JIRA with Zephyr plugin
* Creating Test Cycles and Execute Test cases in Jira

**Manual testing**

Module 1: Testing concepts (Theory) What?

Module 2: Testing Project(Practical) How?

Module 3: Agile process - Jira

**What is Software?**

A Software is a collection of computer programs that helps us to perform a task.

**Types of Software's?**

1) System software

Ex: Device drivers, Operating Systems, Servers, Utilities, etc.

2) Programming software

Ex: compilers, debuggers, interpreters, etc.

3) Application software

Ex: Web Applications, Mobile Apps, Desktop Applications etc.

**What is software Testing?**

Software Testing is a part of software development process.

Software Testing is an activity to detect and identify the defects in the software.

The objective of testing is to release quality product to the client.

**Software Quality**

* Bug-free
* Delivered on time
* Within budget
* Meets requirements and/or expectations
* Maintainable

**Project & Product ?**

**Project:**

* Project means developed based on customer requirement i.e., specific to customer.
* Example all service-based companies (Value Labs, Zen Q, etc.)

**Product:**

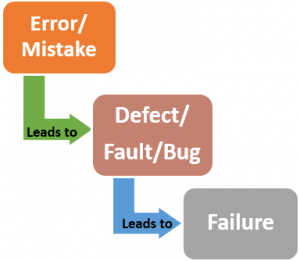
* Product means developed based on market requirement i.e., first we need to capture all market requirements or user requirements and then we will develop product.
* Example all service-based companies (Microsoft , google, Oracle, etc.)

**Why do we need Testing?**

To delivery quality product to customer

**Error, Bug/defect & Failure?**

A mistake in coding is called Error, error found by tester is called Defect, defect accepted by development team then it is called Bug, build does not meet the requirements then it Is Failure.



**Why the software has bugs?**

Software complexity

Miscommunication or no communication

Programming errors

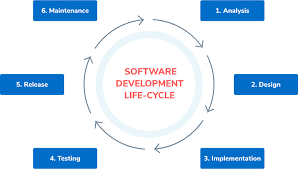
Changing requirements

Lack of skilled testers

**3P's (People, Process, Product)**

**What is SDLC ?**

* Stands for Software development life cycle
* SDLC is complete software development process for beginning to end



**Popular SDLC Models :**

1. Waterfall Model/ Linear Model
2. Iterative Model/ Spiral Model / Version Control Model
3. V Model

**Waterfall Model/ Linear Model**



Phase1 : Requirement Analysis : Product owner or Product manager will collect all requirements from customer and prepare BRS/SRS document

**Input** : Customer Requirements

**Output**: BRS/SRS Documents

Phase2 : Design : Solution Architect/Business Architect will prepare HLD & LLD document based on BRS/SRS Documents

**Input** : SRS Documents

**Output**: HLD & LLD Documents

|  |  |  |
| --- | --- | --- |
| **Difference between High Level Design and Low-Level Design :** | | |
| S.No | HIGH LEVEL DESIGN | LOW LEVEL DESIGN |
| 1 | High Level Design is the general system design means it refers to the overall system design. | Low Level Design is like detailing HLD means it refers to component-level design process. |
| 2 | High Level Design in short called as HLD. | Low Level Design in short called as LLD. |
| 3 | It is also known as macro level/system design. | It is also known as micro level/detailed design. |
| 4 | It describes the overall description/architecture of the application. | It describes detailed description of each and every module. |
| 5 | High Level Design expresses the brief functionality of each module. | Low Level Design expresses details functional logic of the module. |
| 6 | It is created by solution architect. | It is created by designers and developers. |
| 7 | Here in High Level Design the participants are design team, review team, and client team. | Here in Low Level Design participants are design team, Operation Teams, and Implementers. |
| 8 | It is created first means before Low Level Design. | It is created second means after High Level Design. |
| 9 | In HLD the input criteria are Software Requirement Specification (SRS). | In LLD the input criteria are reviewed High Level Design (HLD). |
| 10 | High Level Solution converts the Business/client requirement into High Level Solution. | Low Level Design converts the High-Level Solution into Detailed solution. |
| 11 | In HLD the output criteria are data base design, functional design, and review record. | In LLD the output criteria are program specification and **unit test plan.** |

Phase3 : Implementation : Developers develop the application based on HLD & LLD Documents.

**Input** : HLD & LLD Documents

**Output**: Software

Phase4 : Testing : Tester start testing on software

**Input** : Software

**Output**: Test Plan ,Test Cases ,Test Executions & Test Results.

Phase5 : Deployment /Release : Deployment team will prepare release document based on Test Results.

**Input** : Test Plan ,Test Cases ,Test Executions & Test Results.

**Output**: Release Document

Phase6 : Maintenance : Deployment team will deploy/install the application on customer environment

**Input** : Software & Release Document

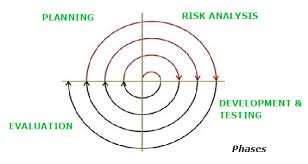
**Advantages of Waterfall Model:**

1. Quality of the product will be good.
2. Since Requirement changes are not allowed, chances of finding bugs will be less.
3. Initial investment is less since the testers are hired at the later stages.
4. Preferred for small projects where requirements are feezed

**Disadvantages of water model:**

1. Requirement changes are not allowed.
2. If there is defect in Requirement that will be continued in later phases.
3. Total investment is more because time taking for rework on defect is time consuming which leads to high investment.
4. Testing will start only after coding.

**Iterative Model/ Spiral Model / Version Control Model:**



**Advantages of Spiral Model:**

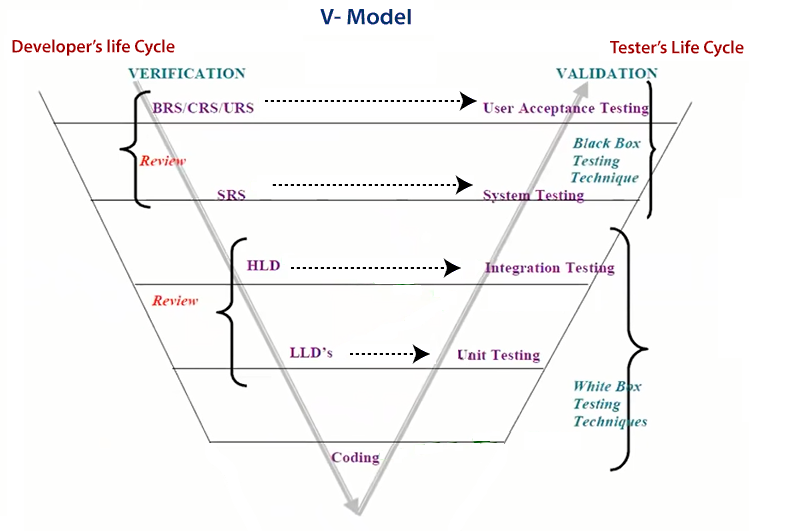
1. Testing is done in every cycle, before going to the next cycle.
2. Customer will get to use the software for every module.
3. Requirement changes are allowed after every cycle before going to the next cycle.
4. Preferred for Product based companies.

**Disadvantages of Spiral Model:**

1. Requirement changes are NOT allowed in between the cycle.
2. Every cycle of spiral model looks like waterfall model.
3. There is no testing in requirement & design phase.

**V Model:**

* Before agile we have V -model
* In v-model both development and testing will go parallelly



**Static testing:**

Testing the project related documents is called as static testing.

**Static testing techniques:**

1. **Review**

A review is a systematic examination of a document by one or more people with the main aim of finding and removing errors early in the software development life cycle. Reviews are used to verify documents such as,

Requirement Reviews

Design Reviews

Code Reviews

Test plan reviews

Test cases reviews etc.

1. **Walkthrough**

Walkthrough is a method of conducting informal group/individual review. In a walkthrough, author describes and explain work product in a informal meeting to his peers or supervisor to get feedback. Here, validity of the proposed solution for work product is checked.

It is cheaper to make changes when design is on the paper rather than at time of conversion. Walkthrough is a static method of quality assurance. Walkthrough is informal meetings but with purpose.

It's not pre-planned and can be done whenever required.

Also walkthrough does not have minutes of the meet.

1. **Inspection**

An Inspection is most formal review type.

In which at least 3-8 people will sit in the meeting 1- reader 2-writer 3- moderator plus concerned group review designed to identify problems as close to their point of origin as possible. Inspections improve reliability, availability, and maintainability of software product.

Inspection will have a proper schedule which will be intimated via email to the concerned developer/tester.

**Dynamic testing:**

Testing the actual software.

**Dynamic testing techniques:**

1. Unit testing
2. Integration
3. System testing
4. UAT

**Verification** is static testing technique.

Here we need to verify every document created for software development, before software development phase is started i.e., whether the document is properly prepared or not we need to ensure correctness and completeness of the document.

Verification done before software development started.

it mainly focusses on Documentation.

So as part of validation we perform

1. Reviews
2. Walkthroughs
3. Inspections

**Validation** is coming under testing part; it is dynamic testing technique.

once software is ready/developed then we perform the validation testing i.e., we check whether the software is working according to the customer or not.

* Takes place after verifications are completed.
* Focus on Software
* So as part of validation we perform

1. Unit testing
2. Integration
3. system testing
4. UAT testing

**Advantages**

Testing is involved in each and every phase.

**Disadvantages**

Documentation is more

Initial investment is more.

**Levels of testing**

* Unit Testing
* Integration Testing
* System Testing
* User Acceptance Testing(UAT)

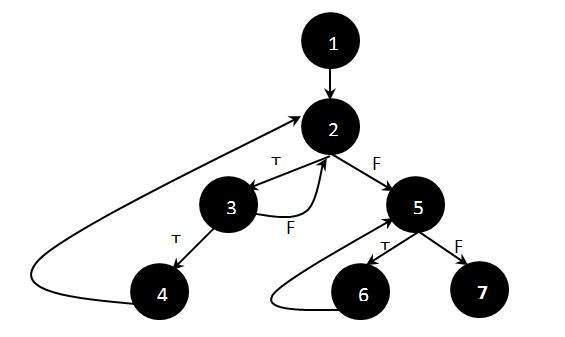
**Unit Testing:**

* A unit is a single component or module of a software.
* Unit testing is process of testing a single program /single component / single module or piece of software.
* This is also called white box testing.
* It will be performed by a programmer or developer.
* Unit testing techniques:
  1. Basis path testing

This technique used for designing test cases intended to examine all possible paths of execution at least once. Creating and executing tests for all possible paths results in 100% statement coverage and 100% branch coverage.

Example:

**Step 1 :**Draw the Flow Graph of the Function/Program under consideration as shown below



**Step 2** : Determine the independent paths.

Path 1: 1 - 2 - 5 - 7

Path 2: 1 - 2 - 5 - 6 - 7

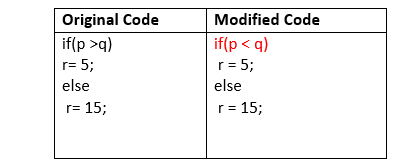
Path 3: 1 - 2 - 3 - 2 - 5 - 6 - 7

Path 4: 1 - 2 - 3 - 4 - 2 - 5 - 6 - 7

* 1. Control structure testing
     1. Conditional coverage
     2. Loops Coverage
  2. Mutation Testing

In mutation testing we insert errors purposely into a program (under test) to verify whether the existing test case can detect the error or not.

Example :



**Integration Testing:**

* it is process of testing the communication between 2 or more modules is called integration testing.

Diagram

Description automatically generated

This is also called internal testing or white box testing.

Developer will perform this testing

**Types of integration testing:**

1. Incremental Integration Testing

Incrementally adding the modules and testing the data flow between the modules.

It has three approaches

* 1. Top Down
     + Incrementally adding the modules and testing the data flow between the modules. And ensure the module added is the child of previous module.
  2. Bottom Up
     + Incrementally adding the modules and testing the data flow between the modules. And ensure the module added is the parent of the previous module.
  3. Sandwich/Hybrid Approach
     + Combination of Top-Down & Bottom-Up approach is called Sandwich Approach.

1. Non- Incremental Integration Testing

Adding all the modules in a single shot and test the data flow between modules.

Drawbacks:

* 1. We might miss data flow between some of the modules.
  2. If you find any defect we can’t understand the root cause of defect.

**System Testing :**

* Here System means complete software or application.
* System testing is process of testing end to end functionality of an application w.r.t customer requirement( i.e., testing entire/complete application functionality)
* This is also called as black box testing
* This testing is conducted by testing team.
* After completion of component and integration level testing's we start System testing. Before conducting system testing we should understand the customer requirements.
* Preparing the test cases , executing the testcase and reporting the bugs all these comes under system testing.
* System testing focusses on below aspects.

**What are the types of system testing?**

There are Four types of system testing

1. Functional Testing
2. Non-Functional Testing

**Functional Testing:**

* Functionality is nothing but behavior of application.
* Functional testing talks about how your feature should work.
* Under FT we will test application UI(GUI), Object Properties Testing (flows, navigations), Database testing, error handling testing, cookies & Session’s testing , links testing & Usability Testing.

**User Interface Testing (GUI) :**

* Graphical User-interface Testing or GUI testing is a process of testing the user interface of an application.
* A graphical user interface includes all the elements such as menus, checkbox, buttons, colors, fonts, sizes, icons, content, and images are correctly placed, aligned, visible or not.

**GUI Testing Checklist:**

* Testing the size, position, width, height of the elements.
* Testing of the error messages that are getting displayed.
* Testing the different sections of the screen.
* Testing of the font whether it is readable or not.
* Testing of the screen in different resolutions with the help of zooming in and zooming out.
* Testing the alignment of the texts and other elements like icons, buttons, etc. are in proper place or not.
* Testing the colors of the fonts.
* Testing whether the image has good clarity or not.
* Testing the alignment of the images.
* Testing of the spelling.
* The user must not get frustrated while using the system interface. Testing whether the interface is attractive or not.
* Testing of the scrollbars according to the size of the page if any.
* Testing of the disabled fields if any.
* Testing of the size of the images.
* Testing of the headings whether it is properly aligned or not.
* Testing of the color of the hyperlink.
* Testing UI Elements like button, textbox, text area, check box, radio buttons, drop downs, links etc.

**Object Properties Testing:**

Object properties testing: Check the properties of objects present on the Application.

Ex: Enable, disable, visible, Focus.

**Database Testing/Backend testing:**

* DML Operations (Data Manipulation Language)

**Insert, delete, update, select**

* DDL Operations (Data Definition Language)

**create, drop, rename, alter etc.**

* Table & Column level validations ( Column type, column length, number of columns...)
* Relation between the tables (Normalization)
* Functions
* Procedures

**Error Handling Testing:**

* Verifying the error messages while performing incorrect actions on the application.

Error messages should be readable.

User understandable/Simple language.

**Cookies & Sessions Testing:**

* Cookies are temporary files created by Browser while browsing the pages through internet.
* Sessions are time slots created by the server. Session will be expired after some time (If you are idle for some time)

**Links Testing:**

* Verifying links are navigating to proper page or not.

Internal links.

External links

Brokens links

**Usability Testing:**

* During this testing validates application provided context sensitive help or not to the user.
* Checks how easily the end users are able to understand and operate the application is called usability testing.

**Non-Functional Testing:**

Once FT is completed then we will go non-FT, here we test

* + - Performance Testing : Speed of the application.
      * **Load**: Increasing the load gradually on system and test the speed of the application is called load testing.
      * **Stress** suddenly increase/decrease the load on the application and check speed of the application.
      * **Volume**: Check how much data is able to handle by the application.
    - Security testing: Verifying an application is secured or not.
      * Authentication : Users are valid or not
      * Authorization/Access Control : permissions of the valid user.
    - Recovery testing : Verifying the application changing properly from abnormal to normal.
    - Compatibility Testing : Verifying an application is properly working across different browsers, databases, operating systems (OS). They should be performed whenever a build becomes stable enough to undergo testing.

**Forward Compatibility** : Testing current build(software) is working properly in next update version of OS or Browsers.

**Backward Compatibility** : Testing current build(software) is working properly in previous version of OS or Browsers.

* + - Configuration/Hardware Testing: Testing an against various combinations of software and hardware to find out the best configuration under which the system can work without any flaws or issues
    - Installation testing: Verifying the installation process of an application.
      * Check screens are clear to understand.
      * Screens navigation
      * Simple or not
      * Un-installation
    - Sanitation/Garbage Testing : If any application provides extra features/functionality then we consider them as bug.

**Regression Testing:**

Testing conducts on modified build to make sure there will not be impact on existing functionality because of changes like adding/deleting/modifying features.

* Unit regression testing:
  + Testing only the changes/modifications done by the developer.
* Regional Regression Testing:
  + Testing the modified module along with the impacted modules
  + Impact Analysis meeting conducts to identify impacted modules with QA & Dev.
* Full Regression:
  + Testing the main feature & remaining part of the application.
  + Ex: Dev has done changes in many modules, instead of identifying impacted modules, we perform one round of full regression.

**Re-Testing:**

* Whenever the developer fixed a bug, tester will test the bug fix is called Retesting.
* Tester close the bug if it worked otherwise re-open and send to developer.
* To ensure that the defects which were found and posted in the earlier build were fixed or not in the current build.
* Example

Build 1.0 was released. Test team found some defects (Defect Id 1.0.1, 1.0.2) and posted.

Build 1.1 was released, now testing the defects 1.0.1 and 1.0.2 in this build is retesting.

**Re-Testing Vs Regression Testing:**

* An Application Under Test has three modules namely login, Add User and Delete User.
* Delete User module depends on Add User module.
* If a tester found a bug on Add User module and posted. Once the bug is fixed, the tester needs to do Retesting to verify whether the bug related to the Add User is fixed or not and also tester needs to do Regression Testing to test the Delete User module which depends on the Add User module.

**Login 🡪 Add User🡪 Delete User**

**Smoke Vs Sanity Testing:**

* Smokeand Sanity Testing come into the picture after build release.
* Smoke is an initial level of testing, smoke is mainly focusing on installation part i.e., build Is properly installed or not or somewhere it is broken or not.
* Once the build is successfully installed, then we perform sanity testing to validate all basic scenarios (like login ,navigations, UI, paginations etc.)
* If sanity testing is successfully completed, then only we perform further testing process.

|  |  |
| --- | --- |
| Smoke Testing | Sanity Testing |
| Smoke Test is done to make sure the build we received from the development team is testable/stable or not | Sanity Test is done during the release phase to check for the main functionalities of the application without going deeper. |
| Smoke Testing is performed by both Developers and Testers | Sanity Testing is performed by Testers alone |
| Smoke Testing, build may be either stable or unstable | Sanity Testing, build is relatively stable |
| It is done on initial builds. | It is done on stable builds. |
| It is a part of basic testing. | It is a part of regression testing. |
| Usually it is done every time there is a new build release. | It is planned when there is not enough time to do in depth testing. |

Diagram

Description automatically generated

**What is difference between White box, Black box testing & gray box testing?**

* White box testing is process of testing internal login of a program by developer or programmer. Here we will not perform any system testing.
* Black box testing is process of testing end to end functionality of an application w.r.t customer requirement. Here we will not perform any internal testing.
* Gray box testing is combination of white & black box testing

Ex : Database testing

**What is difference between QA & QC ?**

|  |  |
| --- | --- |
| QA | QC |
| Stand for Quality Assurance | Stand for Quality Control |
| It defines the complete process of software development activities | QC is the actual testing of the software. |
| All management people will come under this QA (So these people are assurance for the quality of the product at the end) | All testing members are come under this QC ( So these people are responsible for quality of the product) |
| It takes care of entire SDLC process | It takes care of entire STLC process |
| QA focuses on building in quality. | QC focuses on testing for quality. |
| QA is preventing defects. | QC is detecting defects. |
| QA is process oriented. | QC is Product oriented. |

**What is QE ?**

* QE stands for Quality Engineer
* Testing members who write code to test application are come under QE

**Exploratory Testing:**

* We have to explore the application, understand completely, and test it.
* Understand the application, identify all possible scenarios, document it then uses it for

testing.

* We do exploratory testing when the Application ready but there is no requirement document.

i.e. (randomly going to screens, links, and functionalities).

* If we don’t have functionality knowledge and documents to test the application and the build is ready to test, then we go with exploratory testing (ex : new jonnies)
* Test Engineer will do exploratory testing when there is no requirement documents.

**Drawbacks:**

* You might misunderstand any feature as a bug (or) any bug as a feature since you do not have requirement.
* Time consuming
* If there is any bug in application, you will never know about it.

**Adhoc Testing:**

* Testing application randomly without any test cases or any business requirement document.
* Adhoc testing is an informal testing type with an aim to break the system.
* Tester should have knowledge of application even thou he doesn't have requirements/test cases.
* This testing is usually an unplanned activity.
* if testing the application is completed still you have some time then we can perform adhoc testing to find out the corner scenarios or corner defects.
* If we are unable to cover all test cases in a given deadline, then we can perform adhoc testing instead for those remaining testcases.

**Graphical user interface

Description automatically generated**

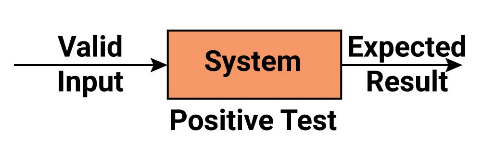
**Monkey/Gorilla Testing:**

* Testing application randomly without any test cases or any business requirement document.
* Adhoc testing is an informal testing type with an aim to break the system.
* Tester do not have knowledge of application.
* Suitable for gaming applications.

|  |  |  |
| --- | --- | --- |
| Adhoc Testing | Monkey Testing | Exploratory Testing |
| No Documentation | No Documentation | No Documentation |
| No Plan | No Plan | No Plan |
| Informal testing | Informal testing | Informal testing |
| Tester should know Application functionality | Testers doesn't know Application functionality | Testers doesn't know Application functionality |
| Random Testing | Random Testing | Random Testing |
| Intension is to break the application/find out corner defects | Intension is to break the application/find out corner defects | Intension is to learn or explore functionality of application |
| Any Applications | Gaming Applications | Any Applications which is new to tester |

**Positive Testing:**

* Testing the application with valid inputs is called as Positive Testing.
* It checks whether an application behaves as expected with positive inputs.



**Example:**

A picture containing graphical user interface

Description automatically generated

There is a text box in an application which can accept only numbers. Entering values up to 99999 will be acceptable by the system and any other values apart from this should not be acceptable.

To do positive testing, set the valid input values from 0 to 99999 and check whether the system is accepting the values.

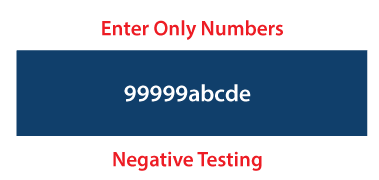
**Negative testing:**

* Testing the application with invalid inputs is called as Negative Testing.
* It checks whether an application behaves as expected with the negative inputs.

A picture containing shape

Description automatically generated

**Example:**



* Negative testing can be performed by entering characters 0 to 9. Either software system should not accept the values or else it should throw an error message for these invalid data inputs.

**Positive V/s Negative Test Cases :**

**Requirement:**

For Example if a text box is listed as a feature and in FRS it is mentioned as Text box accepts 6-20 characters and only alphabets.

**Positive Test Cases:**

* Textbox accepts 6 characters.
* Textbox accepts up to 20 chars’ length.
* Textbox accepts any value in between 6-20 chars’ length.
* Textbox accepts all alphabets.

**Negative Test Cases:**

* Textbox should not accept less than 6 chars.
* Textbox should not accept chars more than 20 chars.
* Textbox should not accept special characters.
* Textbox should not accept numerical.

**END-TO-END Testing:**

Testing the overall functionalities of the system including the data integration among all the modules is called end-to-end testing.

**Login 🡪 Add User🡪 Delete User🡪 logout**

**🡪 Modify User**

**Globalization Testing:**

* Performed to ensure the system or software application can run in any cultural or local environment.
* Different aspects of the software application are tested to ensure that it supports every language and different attributes.
* It tests the different currency formats; mobile number formats and address formats are supported by the application.
* For example, Facebook.com supports many of the languages and it can be accessed by people of different countries. Hence it is a globalized product.

**Localization Testing:**

* Performed to check system or software application for a specific geographical and cultural environment.
* Localized product only supports the specific kind of language and is usable only in specific region.
* Its testes the specific currency format, mobile number format and address format is working properly or not.
* For example, Baidu.com supports only the Chinese language and can be accessed only by people of few countries. Hence it is a localized product.

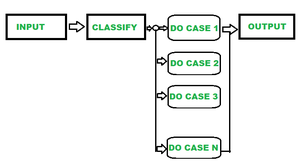
**Test design techniques.**

We have below techniques

1. Input domain testing
   1. Equivalence class partition ( validating valid or invalid input text)
   2. Boundary value analysis (length / range of input text)
2. Decision table technique
3. State Transition
4. Error guessing technique

**Input domain testing:**

IDT is verifying the input fields with minimum number of inputs to check the output of a system.



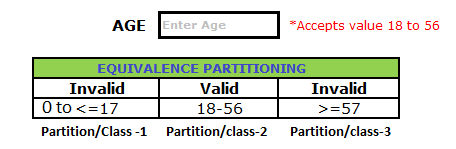
Here we use 2 techniques

1. Equivalence class partition ( validating valid or invalid input text)
2. Boundary value analysis (length / range of input text)

**Equivalence class partition:**

* (ECP) is a Black-Box testing technique
* It is a test design technique that divides the input test data of an application under test into each partition at least once of equivalent data from which test cases can be derived.
* An advantage of this approach is it reduces the time required for performing testing of a software due to less number of test cases.

**Example:**



**Normal Test Data :**

0,1,2,3 ……….56

**Divided input values into equivalence partition or classes**

From partition-1 (0 to 17) let **7** **invalid** input test data

From partition-1 (0 to 17) let **15** **invalid** input test data

From partition-2 (18 to 56) let **21** **valid** input test data

From partition-2 (18 to 56) let **35** **valid** input test data

From partition-2 (18 to 56) let **45** **valid** input test data

From partition-2 (18 to 56) let **55** **valid** input test data

From partition-3 (>=57) let **58** **invalid** input test data

From partition-3 (>=57) let **88** **invalid** input test data

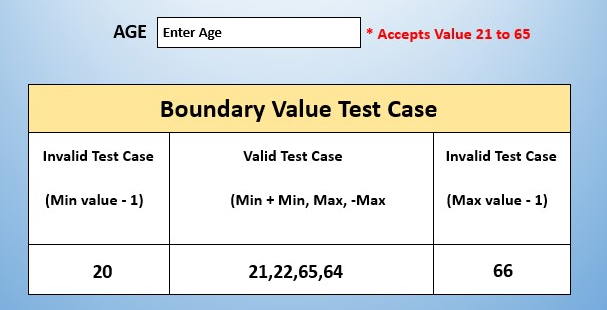
**Test Data using ECP:**

7, 15, 21, 35, 45, 55, 58, 88

**Boundary value analysis:**

Boundary Value Analysis (BVA) is **a Black-Box testing technique used to check the errors at the boundaries of an input domain**. The name comes from the Boundary, which means the limits of an area. So, BVA mainly focuses on testing both valid and invalid input parameters for a given range of a software component.

**Example:**



**Normal Test Data :**

21,22,23,..33.. ……….65

**Test Data using BVA:**

20,21,22,65,64,66

**Decision table technique:**

* Decision Table is also called as Cause-Effect Table.
* This technique will be used if we have more conditions and corresponding actions.
* Decision table technique, we deal with combinations of inputs.
* To identify the test cases with decision table, we consider conditions and actions.

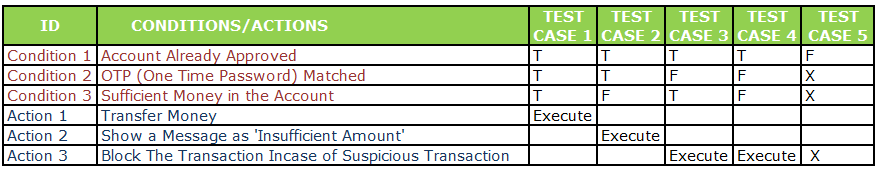
**Decision Table Example:**

* Take an example of transferring money online to an account which is already added and approved.
* Here the conditions to transfer money are

1. Account already approved
2. OTP (one time password) matched
3. Sufficient money in the account

And the actions performed are

1. Transfer money
2. Show a message as insufficient amount
3. Block the transaction in case of suspicious transaction

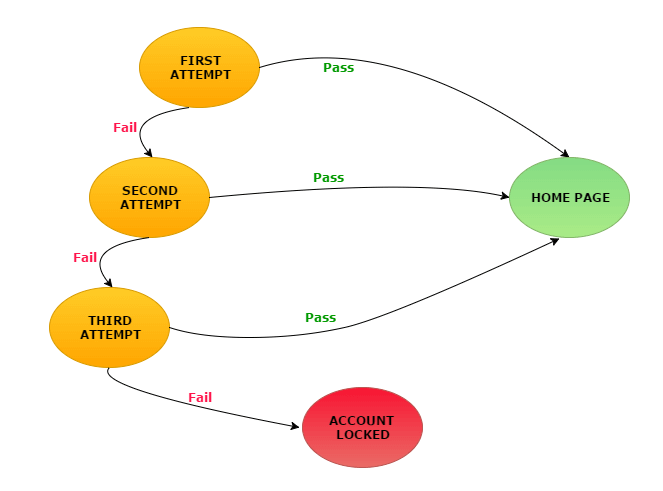


**State Transition:**

* In State Transition technique changes in input conditions change the state of the Application.
* This testing technique allows the tester to test the behavior of an AUT.
* The tester can perform this action by entering various input conditions in a sequence.
* In State transition technique, the testing team provides positive as well as negative input test values for evaluating the system behavior.

**State Transition Example:**

Take an example of login page of an application which locks the username after three wrong attempts of password.



|  |  |  |  |
| --- | --- | --- | --- |
| State | Login | Correct Password | Incorrect Password |
| S1 | First Attempt(S1) | S4 | S2 |
| S2 | Second Attempt(S2) | S4 | S3 |
| S3 | Third Attempt(S3) | S4 | S5 |
| S4 | Home Page(S4) |  |  |
| S5 | Account Locked(S5)  Display a message as  "Account locked, please contact Admin" |  |  |

**Manual Testing FAQ’s:**

1. **Software testing concepts**
2. **Software testing process(STLC)**
3. **Project**
4. **Software testing concepts:**

**What is difference between SDLC & STLC?**

**SDLC:**

* Stands for Software development life cycle
* SDLC is complete software development process for beginning to end

**STLC:**

* stand for software testing life cycle
* it is part of SDLC

**What is Input domain testing and what are the techniques will be used in IDT ?**

IDT is verifying the input fields w.r.t requirements.

Here we use 2 techniques

1. Equivalence class partition ( validating valid or invalid input text)
2. Boundary value analysis (length / range of input text)

**What is DB testing?**

Testing the DML(Data manipulation language) operations w.r.t databases.

**What is use case?**

Use case is requirement, which is described by the product owner or BA.

**What is test case?**

Test case describes the how to test a functionality.

**What is difference between test scenario & test case?**

Test scenario describes what to test i.e., what are the different areas need to cover / test.

Test case describes how to test a functionality.

**2.Software testing process(STLC):**

**What is the testing process followed in your company / in your project?**

Agile process

**Explain about process in your company?**

Explain agile end to end process i.e.,

Product owner will prepare product backlog >> sprint planning meeting>> selecting stories for the sprint >> then sprint cycle will be created >> estimation will be done >> add stories point >> pick the stories for the sprint >> develop and tester will crate the task >> task will be completed between sprint duration >>after completion of this will have sprint review meeting , daily scrum meeting .

**Explain about bug reporting process in your project?**

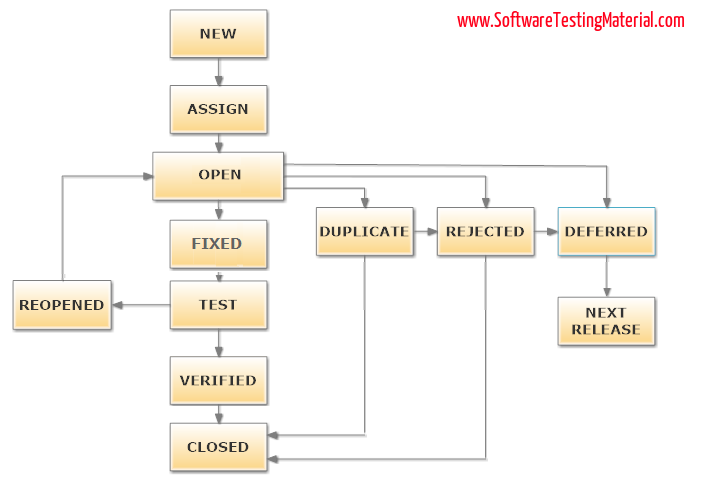
**Step** 1: Once I found a bug then immediately, I will try to reproduce it at least 2 times in 2 environments because there may a change to fail due to invalid input or environment issue or real issue in the application.

**Step** 2: once we conform it as a application issue then we give the severity and priority of the issue

**Step** 3: report bug using Jira tool

**What is defect life cycle?**





**What is priority and severity ?**

Priority describes the importance of the defect

Severity describes impact or serious or crashing the application in the defect

**Give an example for high severity & low priority, High priority & low severity?**

**How to start writing your test cases?**

Once we got user stories >> review those user stories >> then start writing testcases for those stories.

Here stories are the input for writing the testcase.

Once we done with writing test cases then it will be reviewed by the team and finally approved product owner.

**What are your responsibilities / day to day activities ?**

* Review and analyze user stories.
* Write the test cases for the user stories
* Convert test case into automation scripts
* Execute test cases (manual or automated) and analyze results
* Create automation log files
* Report bugs and errors to development teams
* Troubleshoot environment, automation code & CI issues.

**3.Project:**

**Explain your project? Domain, Client?**

**Explain technologies used in your project?**

**What is your team size?**

**How many test cases have you written for your project?**

It depends on the project. 5 test cases per day

**How many defects you reported in your project?**

It depends on the project. 40 bug in a small projects

**You found a defect in QA, but dev is not able to reproduce it. What will you do?**

We need to do in-detailed debug it or find exact root cause of the issue

* check in another environment
* check in data setup for the issue
* check in different browsers
* check memory issue
* check OS issue

**You reported a defect. Dev say its not a defect. It is as per requirement.**

Here we need to approach product owner for exact requirement.

Frequently asked programs.

1. swap 2 numbers
2. Check prime no or not
3. Factorial of a number
4. Fibonacci series
5. Sum of elements in array
6. Max and min element in array
7. Length of the list
8. Remove nth element of the list
9. Search an element in a list
10. Clear the list
11. Clone or copy a list
12. Count occurrences of an element
13. Sum of elements in list
14. multiply of elements in list
15. Max and min number in list
16. Find second largest no in list
17. Check string palindrome
18. Reverse words in a string
19. Sub presence in string
20. Length of the string
21. String contains any string
22. Check URL in a string