Manual testing notes

===

Why we go for Manual Testing?

----

1. Human eye captures more defects.

2. Everything cannot be automated like Captcha, Image, Animation, Video, etc.

3. Report prepared by the manual testers will be always detailed because manual testers are strong in Domain knowledge.

   while automation testers are strong in technical knowlege.

Software Testing:

----

1. Functional Testing

2. Non Functional Testing

1.Functional Testing:

Checking the functionalities of the application.

---> Manual Testing (Testing the functionalities by manual)

---> Automation Testing (Testing the functionalities by using tools/scripts)

---> Webservices Testing (Validating request and response)

Automation Tools:

1.Selenium - Web Based Applications

2. Appium - Mobile Applications

3. QTP(Quick Test Professional)/UFT(Unified Functional Testing) - Desktop Related Applications [Ex: MS Office Applications,

  Web Browsers,etc]

4. Test Complete - Both Web Based and Desktop Based Applications

2.Non-Functional Testing:

Checking the non-functional aspects of the application.

Types of Non-Functional Testing:

1.Performance Testing:

----

*They add virtual number of users accessing the application simultaneously and at a same time to check the

performance of the application.

*Tools Used: i.)J-Meter ---> Open Source Tool

ii.)Load Runner ---> Paid Version

2.Load/Stress Testing:

Process of adding load and making the application stress is called Load/Stress Testing.

3.Usability Testing:

To check how good the application is user friendly.

4.Accessibility Testing:

----

*To check how good the application is accessible to different users.

[Note: This testing mainly focus on physically challenged peoples]

*Tools Used: JAWS and NVDA

Principles of Software Testing:

----

1. Testing shows the presence of defect not their absence.

2. Exhastive testing is not possible (Testing all the combinations of valid or invalid data is impossible).

3. Early testing saves time and money.

4. Defects cluster together (Concentrating the module even it is small which has more defects and testing it thoroughly

will raise of more defect).

It was found that 80% of defects comes from 20% of the modules.

5. Beware of pesticide paradox (Aware of which module the testing is required. Don't Spend time in testing the module

which doesn't have any defects).

6. Testing is context dependent (Whatever the application we are going to test, we have to act based on the context as

what the application related to).

7. Absence of error is a fallacy (Instead of telling bug free, make sure the environments are matching with the client reqirement).

==
==

Levels of Testing:

--> The testing done at various levels and by whom it is done is called Levels of Testing.

--> There are 4 levels of testing. If theses 4 levels of testing is completed only, we can say the application is ready for release.

1.Unit Testing

2.Integration Testing

3.System Testing

4.Acceptance Testing

1.Unit Testing:

--> Testing an individual module in an application

--> Done by Developers

--> Test based on White Box Testing Technique.

2.Integration Testing:

--> Combining or merging two modules

--> Done by Testers

--> Test based on Black Box Testing Technique

--> Bottom-up - Higher level combinations of units

--> Top-down - Lower level combinations of units

3.System Testing:

--> Complete Application Testing

--> Done by Testers

--> Test based on Black Box Testing Technique

--> Types of System Testing:

    --> Functional Testing - To test the functionalities of the application

    --> Usability Testing - To test whether the application is user friendly and it is easily understandable.

    --> Performance Testing - To test whether the application is not getting crashed or down when there is load/stress

    --> Security Testing - To test whether the application is secure or not

--> If theses 4 levels of testing is completed only, we can say system testing is completed

4.Acceptance Testing:

--> Getting approval from client

--> Done by client side

--> Testing will be done at Development/Testing Environment

--> Satisfying client requirement.

===
===

SDLC Overview:

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality software.

Phases of SDLC:

1.Requirement gathering and analysis

2.Design

3.Development

4.Testing---->>STLC

5.Deployment

6.Operation and Maintanenece

Software Development Life Cycle:

1.Requirement gathering and analysis:

--> A Contract will be signed between client and marketing team.

--> Business Analyst(BA) will try to get the requirement from the client and prepare a document called

   Business Requirement Document(BRD) or Software Requirements Specification (SRS) Document

2.Design:

--> Then a document will be prepared called Design Specification Document, which will give details about how the front and back end

   could interact.

3.Development: (Dev Environment)

--> Then Developers will develop the software by the use of any coding language and they will do a test called Unit Testing.

4.Testing:(QA or Testing Environment)

--> Once developer coded for the functionality of the software then they will move their code or build into testing enviornment

   called Testing and once developer moved the code into testing stage ,tester will test the applications as per the flow STLC.

5.Deployment:(Live Environment)

--> Once a program has passed the testing phase, it is ready for deployment.

--> Deploy the application in the live environment.

--> Typically it happens at Non Peak Hours.

6.Opreation and Maintenance:

--> Maintenance of software can include software upgrades, repairs, and fixes of the software if it breaks.

STLC Overview:

--> Software Testing Life Cycle (STLC) is defined as a sequence of activities conducted to perform Software Testing.

--> Each of these stages have a definite Entry and Exit criteria;

Entry Criteria:

Entry Criteria gives the prerequisite items that must be completed before testing can begin.

Exit Criteria:

Exit Criteria defines the items that must be completed before testing can be concluded.

Phases of STLC:

1.Requirement Analysis

2.Test Planning

3.Test Design

4.Test execution

5.Sign Off

1.Requirement Analysis:

Entry Criteria: Contracts, BR Document

--> Understanding the requirements

--> Clarification of doubts in query logs through walkthrough session

--> Identify types of tests to be performed

--> Get application access

Exit Criteria: Finalized scope.

2.Test Planning:

Entry Criteria: Finalized scope

--> Preparation of test plan document for various types of testing

--> Test tool selection

--> Resource planning

Exit Criteria: Test Plan.

3.Test Design:

Entry Criteria: Test Plan

--> Create test scenarios, test cases, automation script, RTM(Requiremnet Tracability Matrix)

--> Create test data

--> Review test cases with peer or lead

--> Get Approval

Exit Criteria: Test Scenarios, Test Case, Test Data, RTM.

4.Test Execution:

Entry Criteria: Test Scenarios, Test Case, Test Data, RTM

--> Execute tests as per plan

--> Document test results, and log defects for failed cases

--> Track the defects to closure

--> Retest the Defect fixes

Exit Criteria: Execution Result, Defect Log.

5.Sign Off:

Entry Criteria: Execution Result, Defect Log

--> Evaluate cycle completion

--> Prepare Test closure report

--> Test result analysis

Exit Criteria: Closure Document.

Test Scenario:

It is high level Testcase. It is an idea of what we are going to test.

Test Case:

It is an idea of how we are going to test, it includes tc name, test data, expected & actual results,status,etc.

Test Data:

The actual input which we are going to use in the application for testing.

Defect:

When the expected and actual is not matching, then it is a defect. In other words, when the application does not conform to the

requirement specification.

Bug:

When the tester raise a defect to developer then it is called as bug

Different Types of Testing:

White Box Testing:

White Box Testing is also called as Glass Box, Clear Box, and Structural Testing. It is based on applications internal coding

knowledge. It is done by developers.

Black Box Testing:

Black Box Testing is a software testing method in which testers evaluate the functionality of the software under test without

looking at the internal coding knowledge. It is done by testers.

Positive Testing:

It is to determine what system supposed to do. It helps to check whether the application is justifying the requirements or not.

Negative Testing:

It is to determine what system not supposed to do. It helps to find the defects from the software.

Beta Testing:

Beta testing is done by a limited number of end users before delivery. Usually, it is done in the client place.

Live Environment Testing:

It it nothing but testing the application in live environment (i.e) After Release.

Smoke Testing:

Smoke Testing is done to make sure if the build we received from the development team is testable or not.

It is also called as “Day 0” check. It is done at the “build level”. It helps not to waste the testing time to simply testing the

whole application when the key features don’t work.

Sanity Testing:

Sanity Testing is done during the release phase to check for the main functionalities of the application without going deeper.

It is also called as a subset of Regression testing. It is done at the “release level”.

At times due to release time constraints rigorous regression testing can’t be done to the build, sanity testing does that part

by checking main functionalities.

Regression Testing:

It is done to make sure the existing functionalities of the application is not impacted whenever any new functionalities is added

to the application.

Formal Testing:

It is a process where the testers test the application by having pre-planned procedures and proper documentation.

Informal Testing:

It is a process where the testers test the application without having any pre-planned procedures and proper documentation.

Monkey Testing:

Perform abnormal action on the application deliberately in order to verify the stability of the application.

ReTesting:

Once the developer fix the bug and we need to test again whether the bug is fixed or not.

Testing Techniques:

Testers randomly test the application without any test cases or any business requirement document.

1.Equivalence case partitioning Technique

2.Decision Table Technique

3.State Transition Technique

4.Boundary Value Analysis

5.Error Guessing Technique

6.Adhoc Testing

Equivalence partitioning Technique:

In equivalence partitioning, the test cases are eqaully divided based upon positive and negative inputs.

Example: Consider that we have to select an age between 18-56,

Here the Valid inputs are 18-56 and the invalid inputs are <=17 and =>57.

Here we have one valid and two invalid inputs.

Decision Table Technique:

In Decision table technique, we deal with combinations of inputs.

To identify the test cases with decision table, we consider conditions and actions.

We take conditions as inputs and actions as outputs.

Example: Login validation - Allow user to login only when both the username and password is correct.

Conditions 1: Enter valid username and valid password and Click Login.

Actions 1: Display home page and Execute.

Conditions 2: Enter invalid username and valid password and Click Login.

Actions 2: Display Error message as invalid username.

State Transition Technique:

Using state transition testing, we pick test cases from an application where we need to test different system transitions.

We can apply this when an application gives a different output for the same input,depending on what has happened in the earlier state.

Example: Login with invalid username and password three times keeps the account page blocked until change password.

Boundary Value Analysis:

Boundary value analysis is based on testing the boundary values of valid and invalid partitions.

Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition.

Example: If we want to enter an amount between 100 to 1000.

Here we check based on boundaries for 100, we take 98,99,101,102 and for 1000, we take 998,999,1001,1002.

Error Guessing Technique:

Error Guessing is used to find bugs in software application based on testers prior experience.

Here we won't follow any specific rules.

Adhoc Testing:

Ad-hoc testing is quite opposite to the formal testing.

It is an informal testing type.

In Adhoc testing, testers randomly test the application without following any documents and test design techniques.

This testing is primarily performed if the knowledge of testers in the application under test is very high.

SDLC Methodologies:

1.WaterFall Model

2.V-Model Of Testing

3.Iterative Model

4.Agile Model

==

1.Waterfall Model:

----

* Simple and easy to understand and use.

* Each phase has specific deliverables and review processes.

* Documentation and artifacts maintained properly.

* Suitable for projects where requirements are well understood.

Disadvatages:

----

* Not suitable for projects where requirements are at a risk of changing.

* Cost of fixing defects is very high when detected at a later stage.

* Not a good model for complex and long projects.

2.V-Model:

----

* The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape.

* It is also known as Verification and Validation model.

3.Iterative Model:

We are going to iterate each functionality in to each part and design,develop and test and deploy.

4.Agile Model:

AGILE methodology is a practice that promotes continuous iteration of development and testing throughout the software development

lifecycle of the project. Here both development and testing activities are concurrent unlike the Waterfall model.

Why we go for Agile?

----

* More Control

* Better Productivity

* Better Quality

* Higher Customer Satisfaction

* Higher return on investment

Agile Manifesto(Principles):

--> Individual and team interactions over processes and tools

--> Working software over comprehensive documentation

--> Customer collaboration over contract negotiation

--> Responding to change over following a plan

[Note: Framework we are following in Agile is Scrum.]

Agile:

1.Roles

2.Artifacts/Documents

3.Cermonies/Meeting

Members/Role Players in Agile:

--> Product Owner:

Product Owner defines features of the product and decides release date.

Product Owner prioritize the features according to the market value and profitability of the product.

Product Owner can accept or reject work item result.

--> Scrum Master:

Scrum Master manages the team and look after the team's productivity.

Scrum Master is responsible for setting up the team, scrum meeting invite and removes obstacles to progress.

--> Scrum Team/Dev Team:

The team is usually about 5-9 members.

The whole team involved in development and testing.

Team manages its own work and organizes the work to complete the sprint or cycle.

And attend the scrum meetings

Different Artifacts used in Agile:

--> Product Backlog:

It is a collection of user stories captured for a scrum product.

The product owner prepares and maintains the product backlog. It is prioritized by product owner.

--> Userstory:

They are short explanation of functionalities of the system under test.

--> Acceptance Crtieria:

They are detailed explanation of functionalities of the system under test which is going to shows what we have to do.

--> Sprint:

It is a set period of time to complete the user stories, decided by product owner,usually 2-3 weeks of time.

--> Sprint Backlog:

It's a set of user stories to be completed in a sprint from the product backlog.

--> BurnUp and Burndown Chart:

The burn-up chart illustrates the amount of completed work in a project whereas the burn-down chart depicts the amount of work

remained to complete a project. Thus, the burn-up and burn-down charts are used to trace the progress of a project.

Qtest - Test Management Tool

JIRA - Defect Management Tool and Test Management Tool

Agile Ceremonies or Meetings:

1.Sprint Groooming:

----

* To understand the user stories.

* Product Owner will be explaining the user stories.

* Meeting Duration - 1 hour.

* Attended by - Product Owner, Scrum Master and Scrum Team.

2.Sprint Planning:

----

* Here the product owner will have already prioritized product backlog.
* Work is selected from the Product Backlog and pulled into the Sprint Backlog.
* Complexity of the user stories will be estimated based on Poker Card Technique.
* Meeting Duration - 1 to 3 hours.
* Attended by - Scrum Master and Scrum Team.

----------------------|Before Sprint---

3.Daily Scrum Meeting (or) Daily Stand up calls:

----

* Stand-up is designed to quickly inform everyone of what's going on across the team.
* It's not a detailed status meeting. Following will be discussed in that meeting

  --> What did I complete yesterday?

  --> What will I work on today?

  --> Am I blocked by anything?
* Meeting Duration - 15 minutes.
* Attended by - Scrum Master and Scrum Team.

----------------------|After Sprint--

4.Sprint Review Meeting:

----

* After completion of every sprint, Team will give the showcase to the product owner.
* Scrum team will show the demo of app with current sprint's completed stories.
* Meeting Duration - 1 hour.
* Attended by - Product Owner, Scrum Master and Scrum Team.

5.Sprint Retrospective Meeting:

----

* Retrospectives help the team to understand what worked well–and what didn't.

* Team will rate the sprint out of 10 and discuss the below

  --> What went right?

  --> What went wrong?

  --> What can be done better for next sprint?

* Meeting Duration - 1 hour.

* Attended by - Scrum Master and Scrum Team.