==========

Tutor: **Pavan Kumar**

Reference: **Youtube**

Course: **Manual Testing (Full Course)**

Content: **Manual Testing - Theory**

(Manual Software Testing Training Part-1 to Part-9)

==========

1. **Document prepared by:**
    a. Rajat Verma
        i. https://github.com/rajatt95
        ii. https://rajatt95.github.io/
        iii. https://www.linkedin.com/in/rajat-v-3b0685128/

2. **More documents:**
    a. https://github.com/rajatt95/Documents

3. **Last worked on this Document:**
    a. Dec 26, 2022

4. **Learnings from Tutor (Code Repository):**
    a. **This course**
        i. https://github.com/stars/rajatt95/lists/youtube-pavan-manual-testing
    b. **Other course(s)**
        i. https://github.com/stars/rajatt95/lists/youtube-pavan-kumar

—----------

5. **Course content:**

    a. **Manual Software Testing Training Part-1**
        i. What is Software?
        ii. Types of Software?
        iii. What is Software Testing?
        iv. What is Software Quality?
        v. Project vs Product
        vi. Why do we need Testing?
        vii. Error, Bug & Failure
        viii. Why the software has bugs?

—----------

—----------

b. Manual Software Testing Training Part-2

i. SDLC
ii. Waterfall Model (Linear Model)
iii. Spiral Model (Iterative model)
iv. V-Model or VV-Model
v. White Box & Black Box Testing
vi. Static Testing & Dynamic Testing
vii. Verification & Validation

c. Manual Software Testing Training Part-3

i. Static Testing Techniques
ii. QA & QC & QE (Quality Assurance, Control, Engineering)
iii. Different Levels of Sofware Testing
    1. Unit Testing
    2. Integration Testing
    3. System Testing
    4. User Acceptance Testing

d. Manual Software Testing Training Part-4

i. System Testing Types
ii. Functional Testing
    1. GUI Testing
    2. Functional Testing
        a. Object Properties Testing
        b. Database Testing (Back end Testing)
        c. Error Handling
        d. Calculations/Manipulations Testing
        e. Links existence and execution
        f. Cookies and Session
iii. Non-Functional Testing
    1. Performance Testing
        a. Load Testing
        b. Stress Testing
        c. Volume Testing
    2. Security Testing
    3. Recovery Testing
    4. Compatibility Testing
    5. Configuration Testing
    6. Installation Testing
    7. Sanitation/Garbage Testing

—----------

—----------

e. **Manual Software Testing Training Part-5**
   i. Regression Testing
      1. Unit Regression Testing
      2. Regional Regression Testing
      3. Full Regression Testing
   ii. Re-Testing
   iii. Re-Testing vs Regression
   iv. Sanity & Smoke Testing
   v. Exploratory Testing
   vi. Adhoc Testing
   vii. Monkey/Gorilla Testing
   viii. **Adhoc** Testing vs **Monkey/Gorilla** Testing vs **Exploratory** Testing
   ix. Positive Testing
   x. Negative Testing
   xi. Positive vs Negative Test cases
   xii. End-To-End Testing
   xiii. Globalization/Internationalization(I18N) and Localization Testing

f. **Manual Software Testing Training Part-6**
   i. **Test** Design Techniques/ **Test Data** Design Techniques/ **Test Case** Design Techniques
      1. Equivalence Class Partitioning (ECP)
      2. Boundary Value Analysis (BVA)
      3. Decision Table-based testing
      4. State Transition
      5. Error Guessing

g. **Manual Software Testing Training Part-7**
   i. STLC (Software Testing Life Cycle)
      1. Requirements Analysis
      2. Test Planning
      3. Test Design
      4. Test case Development
      5. Environment Setup
      6. Test Execution
         a. Bug/Defect Reporting and Tracking
      7. Test Closure

—----------

—----------

h. Manual Software Testing Training Part-8

i. Test Plan
ii. Use case, Test scenario & Test case
iii. Use case vs Test case
iv. Test case vs Test scenario
v. Test Suite
vi. Test case Template
vii. Requirements Traceability Matrix (RTM)
viii. Test Environment
ix. Test Execution
x. Defects/Bugs
    1. Report Contents
    2. Classification
        a. Severity
        b. Priority
    3. Defect Resolution

i. Manual Software Testing Training Part-9

i. Defect/Bug life cycle
ii. Test Cycle Closure
iii. Test Metrics
iv. QA/Testing Activities
v. Principles of Software Testing

—----------

—----------

1. **Learnings from Course (Youtube - Pavan Kumar - Manual Testing (Full course))**

    a. Learning (Module/Phase wise)
        i. **Software Testing concepts (Theory - What to Test?)**
            1. What is Software?
                a. It is a **collection of computer programs** that help to perform a task.
                    i. Programs - Instructions are provided for the software
            2. Types of Software?
                a. **System** (OS, Device Drivers, Utilities)
                b. **Programming** (Compiler, Debugger, Interpreter)
                c. **Application** (Desktop, Web, Mobile)
            3. What is Software Testing?
                a. It is a part of the Software Development process.
                b. It is an activity to **detect and identify defects** in the software.
                c. The objective of Testing is to release **quality products** to the client.
            4. What is Software Quality?
                a. Quality is Customer justification
                    i. How well the software or product is working
            5. Project vs Product
                a. **Specific customer requirements** - **Project**.
                b. **Multiple customers based on the market requirements** - **Product**.
            6. Why do we need Testing?
                a. Bug-free
                b. Delivered on-time.
                c. Within Budget
                d. Meets Requirements/Expectations
                e. Maintainable
            7. Error, Bug & Failure
                a. Error - Incorrect human action
                    i. Environment - DEV (Identified by Developer)
                b. Bug - Deviation of Actual and Expected
                    i. Environment - TEST (Identified by Tester)
                c. Failure - Deviation identified by End User
                    i. Environment - PRODUCTION

—----------

—----------

8. Why the software has bugs?
    a. Miscommunication or No communication
    b. Software Complexity
    c. Programming Errors
    d. Changing Requirements
    e. Lack of skilled Testers
9. SDLC
    a. It is a step-by-step process used by the Software industry to design, develop and test the software and finally, to deliver quality software to the customer.
    b. Phases/Modules
        i. Requirements Analysis
        ii. Design
        iii. Development/Coding
        iv. Testing
        v. Maintenance
10. Waterfall Model (Linear Model)
    a. Every phase is dependent on the previous phase
    b. Requirement changes are not allowed
    c. Testing will start only after Coding/Implementation phase
11. Prototype Model
12. Spiral Model (Iterative model/ Version Control Model)
    a. Testing is done in every cycle before going to the next cycle
    b. Requirement changes are allowed after every cycle
13. V-Model or VV-Model
    a. Testing is involved in each and every phase

Verification	**Validation**
Checks whether we are building the **right** **product**.	Checks whether we are building the **product** **right**.
Focus is on **Documents**	Focus is on **Software**
It involves **Static Testing Techniques** 1. Review 2. Walkthrough 3. Inspections	It involves **Dynamic Testing Techniques** (Testing the actual software) 1. Unit 2. Integration 3. System 4. UAT
	Takes place after Verifications are done/completed.

14. White Box & Black Box Testing
15. Static Testing & Dynamic Testing
16. Verification & Validation
17. QA & QC & QE (Quality Assurance, Control, Engineering)
    i.

QA	**QC**
Process related	**People** related (Tested) Actual Testing of the software
Involved in **all phases/stages** of SDLC QA is for entire SDLC	Involved **only in Testing** phase/stage of SDLC QC is only for Testing phase
Focus is on **building in quality**	Focus is on **testing for quality**
Prevention - Preventing Defects If we follow the process correctly, we can prevent the Defects in future	**Detection - Detecting** Defects
Process oriented	**Product** oriented

    b. **QE** - It is a team that contains **Automation Testers** who are involved in writing code
18. Different Levels of Sofware Testing
    a. **Unit Testing**
        i. A unit is a single component/module of software
        ii. **Unit Testing Techniques**
            1. Basis Path Testing
            2. Control Structure Testing
            3. Conditional Coverage
            4. Loops Coverage
            5. Mutation Testing

    b. **Integration Testing**
        i. It is performed **between 2 or more modules**
        ii. It focuses on checking **Data communication** between multiple modules
        iii. It is a White box Testing technique
        iv. Integration Testing
            1. Developers do this at coding level
            2. Testers do at the Application level

—----------

v. Types
    1. **Incremental Integration Testing**
        a. Approaches
            i. Top Down
            ii. Bottom Up
            iii. Sandwich/Hybrid
    2. Non-Incremental Integration Testing
        a. This is not recommended

c. **System Testing**
i. **Testing overall functionality/features of the application** w.r.t. customers'/clients' requirements
ii. It focuses on aspects
    1. User Interface Testing (GUI)
    2. Functional Testing
    3. Non-Functional Testing
    4. Usability Testing

d. **User Acceptance Testing**
i. Conducted by customers before accepting the software
ii. Types
    1. Alpha Testing
    2. Beta Testing

19. System Testing Types
20. GUI Testing
    a. Graphical User-Interface Testing
    b. It is a process of testing the UI of the application
    c. GUI includes elements
        i. Textboxes, Links, Checkboxes, Buttons, Colors, Images
    d. The End User will interact with the application using the **Front end** and all the operations will happen in the **Back end**

—----------

—----------

21. **Functional Testing**
    a. Functionality
        i. The behavior of the application
    b. Types
        i. **Object Properties Testing**
            1. Checks the property of the objects
                a. Enable, Disable, Visible, Focus
        ii. **Database Testing (Back end Testing)**
            1. This is to ensure that the UI operations are making the required changes in the Database or not
                a. This is **Grey Box Testing** (a combination of Black box and white box)
        iii. **Error Handling**
            1. Verifying the Error messages while performing incorrect actions on the application
        iv. **Calculations/Manipulations Testing**
        v. **Links existence and execution**
        vi. **Cookies and Session**
            1. **Cookies** are created on the **Client side**
            2. **Sessions** are created on the **Server side**

22. Non-Functional Testing
    a. Focused on **Customer expectations** instead of Customer requirements
    b. Performance Testing - Speed
        i. Load Testing (**Gradually** increasing/decreasing the load)
        ii. Stress Testing (**Suddenly** increasing/decreasing the load)
        iii. Volume Testing (Check how much data an application is able to handle)
    c. Security Testing
        i. Authentication
        ii. Authorization/Access Control
    d. Recovery Testing
    e. Compatibility Testing
    f. Configuration Testing
    g. Installation Testing
    h. Sanitation/Garbage Testing

23. Regression Testing
    a. Unit Regression Testing
    b. Regional Regression Testing
    c. Full Regression Testing
24. Re-Testing
    a. The tester verifies the Bug fix only
25. Re-Testing vs Regression
    a. Regression →
        i. Validate the whole application (The changed part should not affect the unchanged part)
    b. Re-Testing →
        i. The tester verifies the Bug fix only
26. Sanity & Smoke Testing
27. Exploratory Testing
28. Adhoc Testing
29. Monkey/Gorilla Testing
30. **Adhoc** Testing vs **Monkey/Gorilla** Testing vs **Exploratory** Testing
31. Positive Testing
    a. Testing the application with **valid inputs**
32. Negative Testing
    a. Testing the application with **invalid inputs**
33. Positive vs Negative Test cases
34. End-To-End Testing
    a. Testing the **overall functionalities** of the system that includes data integration among all the modules.
35. Globalization/Internationalization(I18N) and Localization Testing
    a. Globalization
        i. Application is supported globally
            1. Has the support of multiple languages
36. **Test** Design Techniques/ **Test Data** Design Techniques/ **Test Case** Design Techniques
    a. Equivalence Class Partitioning (ECP)
        i. Partition/Divide/Classify/Categorize Data into various classes
            1. Select the Data according to class
    b. Boundary Value Analysis (BVA)
        i. Focus is on the boundaries of the input
    c. Decision Table-based testing
        i. The decision Table is also known as **Cause-Effect** table
        ii. Conditions and Actions are considered

—----------

—----------

d. State Transition
    i. Used when the application has multiple states
    ii. Input conditions are changed on the basis of application state
        1. Positive and Negative input values → To evaluate the application behavior
e. Error Guessing
    i. Based on Testers' prior experience and Analytical skills
        1. No specific rules
    ii. Example
        1. Submit the form (Without filling in mandatory details)

37. STLC (Software Testing Life Cycle)
    a. Requirements Analysis
    b. Test Planning
    c. Test Design
    d. Test case Development
    e. Environment Setup
    f. Test Execution
        i. Bug/Defect Reporting and Tracking
    g. Test Closure
38. Test Plan
39. Use case, Test scenario & Test case
    a. Use case
        i. Actor -> User
        ii. Action -> Operation
        iii. Goal -> Outcome
    b. Test scenario
        i. What to Test?
        ii. 1 test scenario may have n no. of test cases
    c. Test case
        i. How to Test?
        ii. Contains
            1. Steps
            2. Actual Result
            3. Expected Result
40. Use case vs Test case
41. Test case vs Test scenario
42. Test Suite
43. Test case Template
44. Requirements Traceability Matrix (RTM)

45. Test Environment
46. Test Execution
47. Defects/Bugs
    a. Report Contents
    b. Classification
        i. Severity
        ii. Priority
    c. Defect Resolution
48. Defect/Bug life cycle
49. Test Cycle Closure
50. Test Metrics
51. QA/Testing Activities
52. Principles of Software Testing

—----------

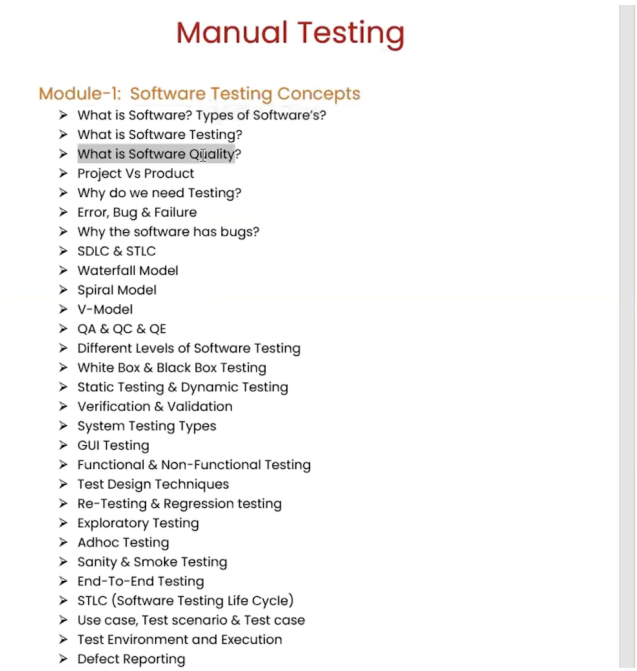
==========

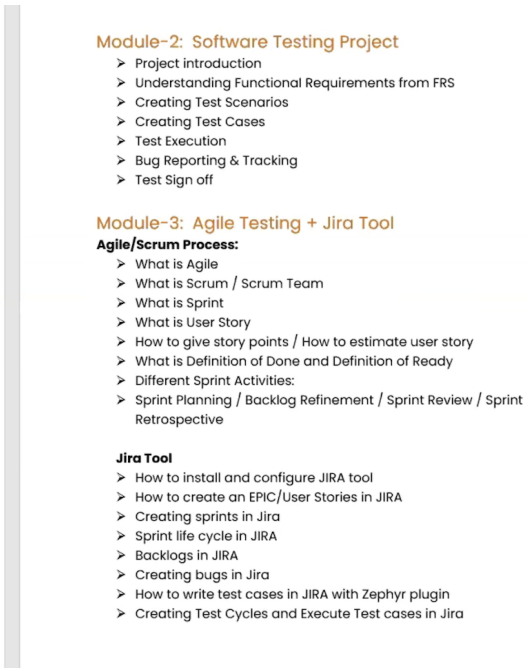
========1_Manual Software Testing Training Part-1==========

---

1. Learning (Module/Phase wise)
    - **a. Software Testing concepts (Theory)**
        - i. Very basics
        - ii. What to Test?
    - **b. Testing Project (Practical)**
        - i. How to Test?
    - c. Agile process
    - d. JIRA
        - i. To Track all activities of project
            1. From Requirement Gatherings to Sowftare Delivery to the Customer

—----------

1. 



    - a. Module-1
        - i. Test Closure
        - ii. Test Metrics

—----------

1. The tutor will share SQL videos
    - a. After completing the Manual Testing basics

—----------

—----------

1. **What is Software?**
    a. It is a collection of computer programs that help to perform a task.
        i. Programs
            1. Instructions are provided to the software
    b. A Mobile is a machine that has Software
        i. WhatsApp
        ii. Maps

—----------

2. **Types of Software?**

    a. System software
        i. Used to run the system
        ii. Example
            1. Device Drivers
                a. Printer
                b. Keyboard
                    i. If you have connected your keyboard to the computer, How will the computer be able to identify the instructions coming from Keyboard?
                c. Bluetooth devices
            2. Operating Systems
                a. On top of the OS, we install other software like
                    i. MS Office
                    ii. Outlook
                b. OS works as a base
                c. OS types
                    i. Windows
                    ii. MAC
                    iii. Linux
            3. Servers
            4. Utilities

    b. Programming software
        i. Compilers
            1. When we write the code, how can your machine understand that code?
            There is something working internally to understand the inputs and give the output.
        ii. Debuggers
        iii. Interpreters

—----------

—----------

c. **Application software**

i. **Desktop applications**

1. We can install these applications on our laptop/Desktop
    a. Outlook
    b. MS Office
    c. Browsers
        i. Chrome
        ii. Firefox
        iii. Edge
    d. Calculator
    e. Paint

ii. **Web applications**

1. We need the Internet and a Browser to access these applications
2. Actual applications are running on the remote servers
    a. https://www.google.co.in/
    b. https://www.flipkart.com/
    c. https://www.youtube.com/

iii. **Mobile applications**

1. 2 OS
    a. Android
        i. Playstore
    b. iOS
        i. Appstore
2. We can download the applications from these stores
    a. Applications
        i. LinkedIn
        ii. WhatsApp

—----------

1. What is Software Testing?

a. Example -1

i. You buy a mobile
ii. You install some applications
iii. You start using those applications
iv. But, these applications are not working properly
    1. Crashing sometimes
    2. Taking a very long time to start
v. How will you feel about the Buggy applications?
    1. Worst
vi. Why does this happen?
    1. Lack of Testing
    2. Testing is not properly conducted

—----------

b. Example -2

i. X-Bank reaches an IT company with a specific set of requirements with Budget and Time
    1. Requirements may be
        a. To automate their processes/services like
            i. Fixed/Recurring Deposit
                1. Create
                2. Break
            ii. Customer support (24*7)
    2. These requirements will be reviewed by the IT team
        a. Factors like
            i. Budget
            ii. Time
                1. Will be considered for any outcome promise

ii. After all formalities,
    1. IT company will
        a. Develop the Sofware
        b. **Test** the Software
            i. Why?
                1. To release/deliver a quality product to the customer.
        c. Deliver the Software to Bank
    2. IT Team members
        a. Managers
        b. Designers
        c. Developers
        d. Testers

—----------

c. Software Testing

i. It is a part of the Software Development process.
    1. Software Development is not only about writing the code by Developers
    2. We have to understand the customer's requirements and have to verify that things are moving in the right direction

ii. It is an activity to detect and identify defects in the software.

iii. The objective of Testing is to release quality products to the client.

—----------

—----------

1. **What is Software Quality?**
    a. Quality is Customer justification
        i. How well the software or product is working
    b. Parameters
        i. Bug-free
        ii. Delivered on-time.
        iii. Within Budget
        iv. Meets Requirements/Expectations
        v. Maintainable

—----------

1. **Project vs Product**
    a. If the software application is developed for
        i. **Specific customer requirements**, then, it is called **Project**.
            1. Working for JetBlue Airlines
        ii. **Multiple customers based on the market requirements**, then, it is called **Product**.
            1. Flipkart
            2. WhatsApp
            3. LinkedIn
            4. MS Office
    b. Companies
        i. Service-based
            1. TCS
            2. Accenture
            3. Nagarro
            4. IBM
        ii. Product-based
            1. Google (Products - Gmail, Youtube, Maps)
            2. Microsoft (Products - MS Office, WIN)
            3. Oracle

—----------

1. **Why do we need Testing?**
    a. The objective of Testing is to release quality products to the client.
    b. Quality is Customer justification
        i. How well the software or product is working
    c. Parameters
        i. Bug-free
        ii. Delivered on-time.
        iii. Within Budget
        iv. Meets Requirements/Expectations
        v. Maintainable

—----------

1. **Error, Bug & Failure**

    a. **Error**
        i. Human mistake
        ii. Incorrect human action
        iii. Environment - Development (Identified by Developer)

    b. Bug
        i. Deviation of Actual and Expected
        ii. Environment - Testing (Deviation Identified by Tester)
        iii. Example
            1. Login
                a.

Username	**Password**	**Result**	**Result**
Valid	Valid	Able to login	**PASS**
Valid	Invalid	Able to login	**FAIL**
Invalid	Valid	Able to login	**FAIL**
Invalid	Invalid	Able to login	**FAIL**

    c. Failure
        i. End User action
        ii. Environment - Production (Deviation Identified by End User)
            1. We delivered the product to the customer (We developed and tested it)
            2. And, the product is being used by customers in their environment

—----------

1. **Why the software has bugs?**
    a. Miscommunication or No communication
        i. Business Analysts - Developers - Testers
        ii. Everyone should be in sync
        iii. Requirements should be very very clear
    b. Software Complexity
    c. Programming Errors
        i. Developers' responsibility
    d. Changing Requirements
    e. Lack of skilled Testers

—----------

========2_Manual Software Testing Training Part-2==========

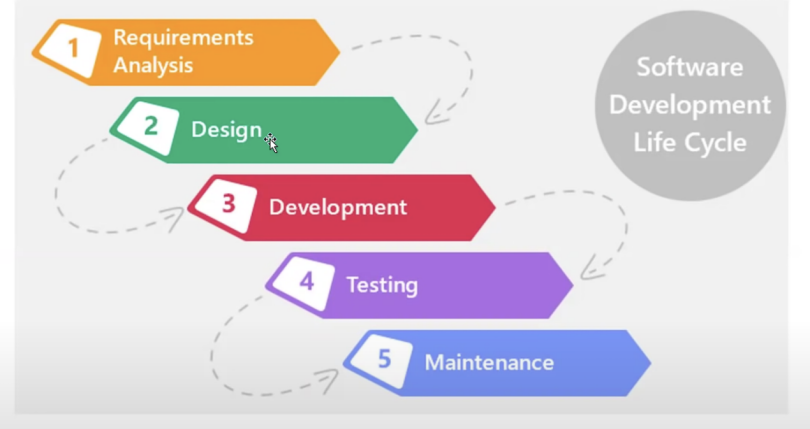
---

1. **SDLC**
    a. Life Cycle for any Product (Not specific to IT only)
        i. Example - Pharmaceutical company
            1. Raw materials (taken as input) -> Processing -> Release the product
            2.
    b. The full form of SDLC is **Software Development Life Cycle**
        i. IT companies
            1. They take customer requirements
            2. Based on the requirements gathered, they design the software
            3. They implement/develop the code
            4. Conduct Testing
            5. Release the software and deliver it to the Client

    c. SDLC
        i. It is a step-by-step process used by the Software industry to design, develop and test the software and finally, to deliver quality software to the customer.

    d. **3 pillars** for any company
        i. P - People
        ii. P - Process
        iii. P - Product (we deliver the software/product to the customer)

    e. Phases/Stages in SDLC

        i. 

            1. **Requirements Analysis**
                a. Collect and understand the requirements of the customer
                b. Project/Products Managers are included in this phase
                c. They prepare the documents such as

        i. SRS (Software Requirement Specification)
            1. Every requirement from the customer must be listed here

2. **Design**
    a. Designers will design the software based on the documents prepared in the Requirements Analysis phase
        i. High-Level Design
        ii. Low-Level Design
    b. Example
        i. Build a house
            1. We prepare a blueprint
3. **Development/Coding**
    a. Software Developers will write the different types of programs based on the design documents
    b. The design phase will work as an input for Coding/Implement phase
4. **Testing**
    a. Types
        i. Functional
        ii. Non-Functional
    b. After completion of Development, Testing teams start doing Testing for product
5. **Maintenance**
    a. We deploy the software in the customer's environment
    b. And, the customer starts using the software

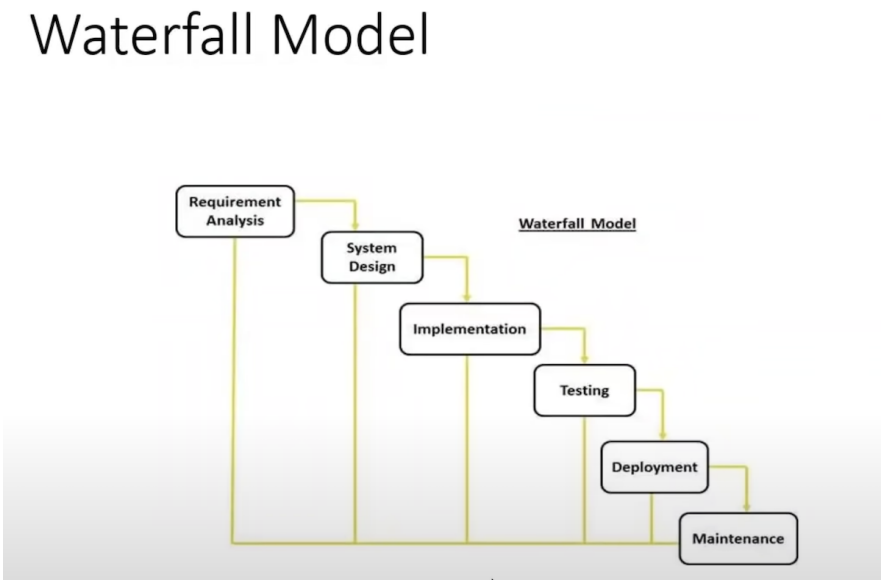
—----------

1. **All models come under SDLC**

—----------

—----------

1. **Waterfall Model (Linear Model)**



a.

   i. The waterfall model is the
      1. First and initial model (Software industry started)
   ii. Every phase is dependent on the previous phase
      1. The requirement Specification document will be the input for Design phase
      2. Design documents will be the input for the Implementation phase

b. Advantages

   i. Product quality will be good
      1. We have detailed documentation (Activities are documented)
   ii. Since Requirement changes are not allowed, chances of finding will be less
   iii. The initial investment is less since the Testers are hired at a later stage
   iv. Preferred for small projects where requirements are frozen

c. Disadvantages

   i. Requirement changes are not allowed
   ii. If there is a defect in the requirement phase, it will be continued in later phases/stages
   iii. Total investment is more because the time taken for Re-Work on defects is time-consuming which leads to high investment
   iv. Testing will start only after the Coding/Implementation phase

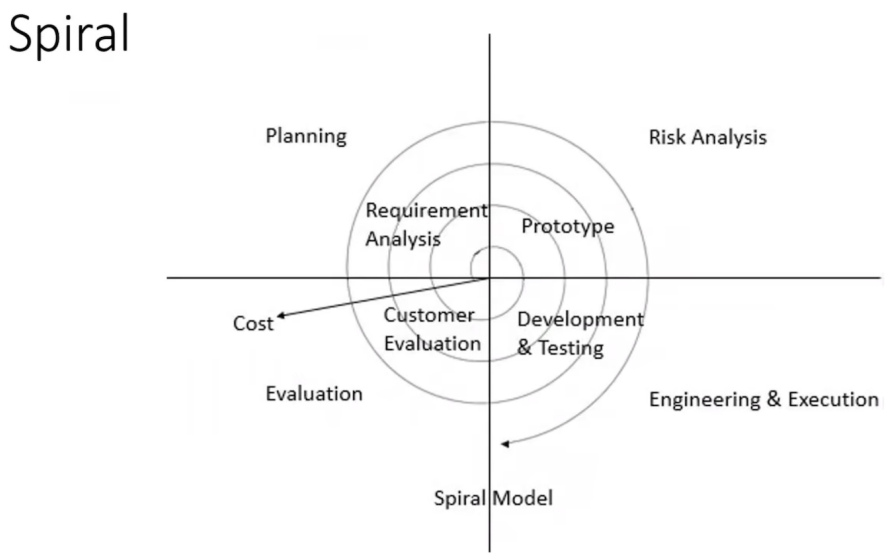
—----------

—----------

1. **Prototype Model**
    a. Prototype - Blueprint of the software
    b. We get the initial requirements from customer
    c. We only develop the **Prototype** instead of the whole product
        i. If the customer is satisfied, then, will for Design → Develop → Test phases
        ii. Example 1 - Gmail has modules such as
            1. Log in to the application
            2. Compose an Email
            3. Send an Email
            4. Receive an Email
        iii. Example 2 - Banking
            1. Login
            2. Check balance
            3. Fund Transfer
            4. Request statement
            5. Add Beneficiary
    d. This model comes in between Waterfall and Spiral model
        i. Waterfall → Prototype → Spiral

—----------

1. **Spiral Model (Iterative model/ Version Control Model)**



a.
    i. Terminologies
        1. Requirements Analysis
            a. Planning
            b. Risk Analysis
        2. Design and Development
            a. Engineering & Execution

3. Testing
    a. Evaluation

ii. Software is delivered in different versions
    1. All phases/stages will be followed for different versions
    2. V1 -> V2 -> V3
    3. In every cycle, all the stages/phases are repeated

iii. This model is recommended for Product based companies
    1. We get the updates (v1 -> v2 -> v3)
    2. Functionalities are added/updated

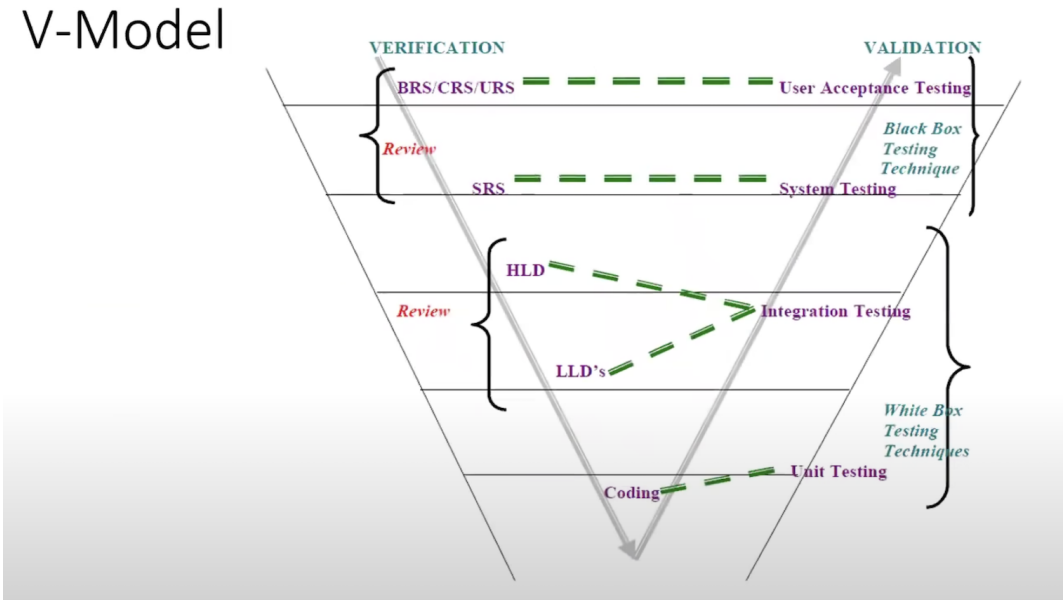
iv. Spiral Model
    1. It is an **Iterative model**
    2. It overcomes the drawbacks/limitations of the Waterfall model
        a. Requirement changes are allowed in Spiral model
    3. We follow the Spiral model whenever there is a dependency on module
    4. In every cycle, the new software will be released to the customer
        a. Software will be released in multiple versions
            i. So, it is also called **Version Control model**

b. Advantages
i. Testing is done in every cycle before going to the next cycle
ii. Customers will get to use the software for every module
iii. Requirement changes are allowed after every cycle

c. Disadvantages
i. Requirement changes are not allowed in between cycles
ii. Every cycle of the Spiral model looks like the Waterfall model
iii. There is no Testing in the Requirement and Design phase

—----------

—----------

1. **V-Model or VV-Model**
    a. Verification and Validation model
    b. 

    i. In every phase of Software Development, we conduct Testing
        1. SRS document will work as an input at the time of performing System Testing
    ii. Terminologies
        1. **BRS** - **Business** Requirements Specification
        2. **CRS** - **Customer** Requirements Specification
        3. **URS** - **User** Requirements Specification
    iii. BRS/CRS/URS is a document that contains the Business/Customer/User requirements
        1. **The business Unit team** will prepare the document
            a. Those people interact with the customer → Collect the requirements → Prepare the document
        2. This document will work as an input for performing **UAT** (User Acceptance Testing)
            a. This document is not understandable by Technical Team
                i. Developers
                ii. Testers
    iv. So, there is another document (**SRS** - Software Requirement Specification) prepared
        1. Certain diagrams are used in SRS
        2. SRS is prepared by Product/Project Managers
    v. Now, based on the SRS document,
        1. Designers will prepare the Designs
            a. HLD (High-Level Design)

b. LLD (Low-Level Design)

vi. To perform the Testing on these documents

1. There are **Software Techniques**

a. Review

b. Walkthrough

c. Inspection

i. These are

1. used to verify the documents.

2. Static Testing

a. Testing the Project related documents

b. We are not testing the software, we are testing the documents in the form of Review, Walkthrough, and Inspection.

e. Validation has 4 levels of Testing

i. Unit Testing

1. Module developed by Developer → Tests the single module

ii. Integration Testing

iii. System Testing

iv. User Acceptance Testing

f. White Box Testing Techniques

i. We have to know the internal logic of the programs

ii. Has to be performed by Developers

1. Unit Testing
2. Integration Testing

g. Black Box Testing Techniques

i. Don't need to be skilled in programming

ii. Testers have to check the

1. Functionalities
2. Workflows
   a. Working as per the customers' requirements or not

iii. This has to be performed by Tester

1. System Testing
2. User Acceptance Testing

h. Dynamic Testing

i. Testing the actual software using techniques

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

i. **Verification vs Validation**

i.

Verification	**Validation**
Checks whether we are building the **right** **product**.	Checks whether we are building the **product** **right**.
Focus is on **Documents**	Focus is on **Software**
It involves **Static Testing Techniques** 4. Review 5. Walkthrough 6. Inspections	It involves **Dynamic Testing Techniques** (Testing the actual software) 5. Unit 6. Integration 7. System 8. UAT
	Takes place after Verifications are done/completed.

j. **Advantages**

i. Testing is involved in each and every phase

k. **Disadvantage**

i. Documentation is more

ii. Initial investment is more

—----------

========3_Manual Software Testing Training Part-3==========

---

1. **Static Testing Techniques**

    a. **Review**
        i. Conducted on Documents to ensure
            1. Correctness
            2. Completeness
        ii. Types of Review
            1. Requirements Review
                a. Every requirement is listed/mentioned or not
                b. To understand the requirements better
            2. Design Review
                a. Will be conducted on the Design documents
            3. Code Review
                a. Coding standards
                b. Logic
            4. Test Plan Review
            5. Test cases Review
        iii. Review can be done by anybody anytime

    b. **Walkthrough**
        i. It is an **Informal Review**
        ii. **Author (person who created the document)** reads the documents/code → Discuss with peers
            1. Peers -> Team members
        iii. It is **not pre-planned**
        iv. Can be done whenever required

    c. **Inspection**
        i. It is the **most formal review type**
        ii. At least 3-8 people will sit in meeting
            1. **Reader (Author of the document)**
            2. **Writer**
                a. **Will write the Questions/Issues/Clarifications raised by the team members**
            3. **Moderator (Meeting Organiser)**
        iii. Inspection will have a proper schedule
            1. Intimated via Email to the concerned DEV/Tester

—----------

—----------

1. **Dynamic Testing Techniques**
    a. **Unit Testing**
        i. Separate modules will be tested with the help of unit test cases written by Developers
    b. **Integration Testing**
        i. Modules will be integrated and then, tested by Developers
        ii. Will check the Data flow between the components
    c. **System Testing**
        i. Testers will do the testing
            1. Deviation from Actual and Expected
            2. Software should work as per the customers' requirements
            3. Functionality is working fine or not
    d. **User Acceptance Testing**
        i. By Testers and Customers

—----------

1. **QA & QC & QE (Quality Assurance, Control, Engineering)**
    a. **QA vs QC**
        i.

QA	**QC**
Process related	**People** related (Tested) Actual Testing of the software
Involved in **all phases/stages** of SDLC QA is for entire SDLC	Involved **only in Testing** phase/stage of SDLC QC is only for Testing phase
Focus is on **building in quality**	Focus is on **testing for quality**
Prevention - Preventing Defects If we follow the process correctly, we can prevent the Defects in future	**Detection - Detecting** Defects
Process oriented	**Product** oriented

    b. **QE**
        i. Quality Engineering
            1. Developers → **SDE** → Write the code → To **develop** the software
            2. Testers → **SDET** → Write the code → To **test** the software
                a. Quality Engineer
                b. Automation Tester
            3. It is a team that contains **Automation Testers** who are involved in writing the code

—----------

1. **Levels of Software Testing**

    a. **Unit Testing**
        i. A unit is a single component/module of software
        ii. Unit testing conducts on a single program/module
        iii. It is conducted by Developers in their environment
        iv. It comes under White Box Techniques
        v. **Unit Testing Techniques**
            1. Basis Path Testing
            2. Control Structure Testing
            3. Conditional Coverage
            4. Loops Coverage
                a. For, for each, while
            5. Mutation Testing
                a. With different inputs

—----------

    b. **Integration Testing**
        i. Combining multiple units/modules as a single unit
        ii. Will check the Data flow and communication flow between those modules
        iii. It is performed **between 2 or more modules**
        iv. It is a White box Testing technique
        v. Integration Testing
            1. Developers do this at coding level
            2. Testers do at Application level
        vi. Types
            1. **Incremental Integration Testing**
                a. Incrementally adding the modules and testing the data flow between modules - Modules are added one after another
                b. Approaches
                    i. Top Down
                        1. Ensure that the module added is the **child** of the previous module.
                        2. Example - Gmail
                            a. Compose mail –> Sent mail → Deleted Items
                    ii. Bottom Up
                        1. Ensure that the module added is the **parent** of the previous module.
                    iii. Sandwich/Hybrid
                        1. Combination of Top-Down and Bottom-Up approach

2. **Non-Incremental Integration Testing**
    a. Adding/Integrating all the modules in a single shot and testing the data flow between modules
    b. This is not recommended
        i. **Drawbacks**
            1. We might have Data flow between some of the modules
            2. If you find any defect, we can't understand the root cause of the defect

—----------

c. System Testing

i. Testing the overall functionality of the application
    1. Whether it is working as per customers' requirements or not

ii. Will cover this in detail in later sessions

iii. Overview
    1. **Testing overall functionality/features of the application** w.r.t. customers'/clients' requirements
        a. Example - WhatsApp is an application that has features
            i. Send Image(s)
            ii. Send Location (Current, Live)
            iii. Send Contact
            iv. Update profile pic
    2. It is a Black box Testing technique
    3. Conducted by Testing team
    4. After completion of the component and integration level, -> System Testing has to be done
    5. It focuses on aspects
        a. User Interface Testing (GUI)
            i. Look and Feel of the application
        b. Functional Testing
            i. Functionalities - Banking application
                1. Login
                2. Money Transfer
                3. Account Statement
                4. Create/Break Deposits (Fixed, Recurring)
        c. Non-Functional Testing
            i. Security
            ii. Performance
            iii. Compatibility

—----------

—----------

d. Usability Testing

i. User manuals

1. Have all required content in the proper format or not

ii. Checks how easily the end user is able to understand and operate the application

—----------

d. User Acceptance Testing

i. Testers + Customers (End Users)

ii. Conducted by customers before accepting the software

iii. Types

1. **Alpha Testing -** Customers test the application in a development/testing environment
2. **Beta Testing -** Customers test the application in customers' environment

iv. After completion of Alpha and Beta Testing, we push the application in the production environment

—----------

========4_Manual Software Testing Training Part-4==========

---

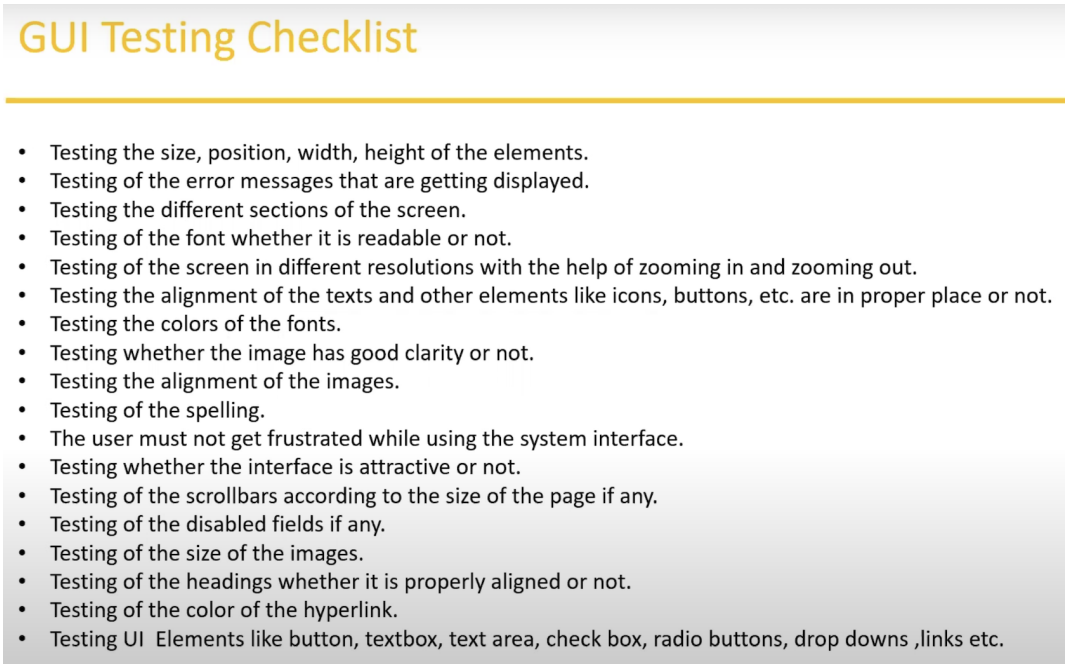
1. **System Testing Types**

    a. **Testing overall functionality/features of the application** w.r.t. customers'/clients' requirements
    b. After completion of Integration Testing, System Testing starts
        i. GUI Testing
        ii. Usability Testing
        iii. Functional Testing
        iv. Non-Functional Testing
    c. Testers will have a separate environment to perform the System Testing
        i. QA/Testing

—----------

1. **GUI Testing**
    a. Graphical User-Interface Testing
    b. It is a process of testing the UI of the application
    c. GUI includes elements
        i. Menus
        ii. Textboxes
        iii. Links
        iv. Checkboxes
        v. Buttons
        vi. Colors
        vii. Fonts
        viii. Sizes
        ix. Icons
        x. Content
        xi. Images
    d. The End User will interact with the application using the **Front end** and all the operations will happen in the **Back end**

e.



i. For testing the elements, Testers refer to Design documents
    1. Design docs - **Wireframes**
        a. CSS Properties like position, height, width, font color, font size → Listed/Mentioned in the Wireframe
    2. Before starting writing the actual code for the application, the design of the application is prepared

—----------

1. **Usability Testing**
    a. Checks how easily the end user is able to understand and operate the application

—----------

1. **Functional Testing**
    a. Functionality
        i. The behavior of the application
        ii. It talks about how the features of an app should work
    b. Types
        i. Object Properties Testing
            1. Checks the property of the objects
                a. Example
                    i. Enable
                    ii. Disable
                    iii. Visible
                    iv. Focus
                b. Textbox/Checkbox -> Enable/Disable
                c. Radio buttons -> Can only select one at a time
                d. Dropdown -> Can select only one option

—----------

—----------

ii. Database Testing (Back end Testing)

1. DML operations (Data Manipulation Language)
    a. SELECT
    b. INSERT
    c. UPDATE
    d. DELETE
2. Perform the operation via UI and validate at the Database level
    a. SQL -> Table
    b. No SQL -> Documents
3. This is to ensure that the UI operations are making the required changes in the Database or not
    a. This is **Grey Box Testing** (a combination of Black box and white box)
4. **Database Testing Checklist**
    a. Table Level Validations
        i. What kind of data can be added in Table
        ii. Column Type
        iii. Column Length
        iv. No. of columns
        v. No. of rows
    b. Relation between the Tables
        i. Normalization
    c. Functions
    d. Procedures
    e. Triggers
    f. Indexes
    g. Views

—----------

iii. Error Handling

1. Incorrect action on the UI should give a proper message to User
2. Verifying the Error messages while performing incorrect actions on the application
    a. Error messages should be
        i. Readable
        ii. User understandable/Simple language

—----------

—----------

iv. Calculations/Manipulations Testing

1. Example - Banking application
    a. Total Balance - 5000
    b. Money transferred - 2000
    c. The remaining balance should be 3000 (Total Balance - Money transferred)
2. More Data-centric
3. Verifies the calculations

—----------

v. Links existence and execution

1. Where exactly links are placed
2. Links are navigating to the proper page or not
3. Links
    a. Internal -> Click -> Different section of same page
    b. External -> Click -> Different Page
    c. Broken -> Click -> No action (No target)

—----------

vi. Cookies and Session

1. Example
    a. Login - Provided credentials
    b. The user is logged in
    c. Browser stores some Data so that when the User access the application again, the user does not need to provide the credentials, he/she is already logged in
2. **Cookies** are created on the **Client side**
    a. Cookies are the temporary files
    b. Created by the browser while browsing the pages through Internet
3. **Sessions** are created on the **Server side**
    a. Example - Login to HDFC net banking
        i. Do not do anything for 2 minutes
            1. This means the User is inactive
        ii. Then, try to navigate to some link
            1. Your session will be expired
            2. The user will be logged out and the application will ask to Re-Login
4. Sessions are time slots created by the Server.
    a. Session will be expired after some time.

—----------

—----------

1. **Non-Functional Testing**
    a. Focused on **Customer expectations** instead of Customer requirements
    b. Types
        i. **Performance Testing**
            1. Check the speed of the application
            2. Tools
                a. LoadRunner
                b. JMeter
            3. Parameters
                a. Response Time
                b. Throughput
                c. Process time
                d. Performance Metrics
        ii. **Types** of Performance Testing
            1. Load Testing
                a. **Gradually** increasing/decreasing the load on the application slowly, then, checking the speed of the application
                    i. 1 user
                    ii. 10 users
                    iii. 100 users
                    iv. 1000 users
                    v. 10000 users
            2. Stress Testing
                a. **Suddenly** increasing/decreasing the load on the application slowly, then, checking the speed of the application
                    i. 1 user
                    ii. 1000 users
                    iii. 10 users
            3. **Volume Testing**
                a. Size
                b. Check how much data an application is able to handle

—----------

        iii. **Security Testing**
            1. How secure an application is
            2. Authentication
                a. Only valid Users should be able to access the application
            3. Authorization/Access Control
                a. Permissions to the valid user
                b. Users
                    i. Normal
                    ii. Admin

—----------

—----------

iv. Recovery Testing

1. Example
    a. Delete some files -> Goes to Recycle Bin
        i. We can still recover these deleted files by using Restore option
    b. Gmail -> Composing an Email
        i. The system got shut down
        ii. We can still find our changes in Drafts section
2. Checks the system change from abnormal to normal.

—----------

v. Compatibility Testing

1. **Forward** compatibility
    a. V 2.0 of the software should be compatible with the devices on which Software V 1.0 was working fine
2. **Backward** compatibility
    a. Machine is compatible with both newer and older versions of software
3. **Hardware** compatibility/Configuration Testing
    a. Supports multiple OS or not
    b. 4 GB RAM is sufficient or not

—----------

vi. Configuration Testing

1. Supports multiple OS or not
2. 4 GB RAM is sufficient or not

—----------

vii. Installation Testing

1. Check screens are clear to understand
2. How screens are navigating
3. Simple or not
4. Un-installation should also be clear

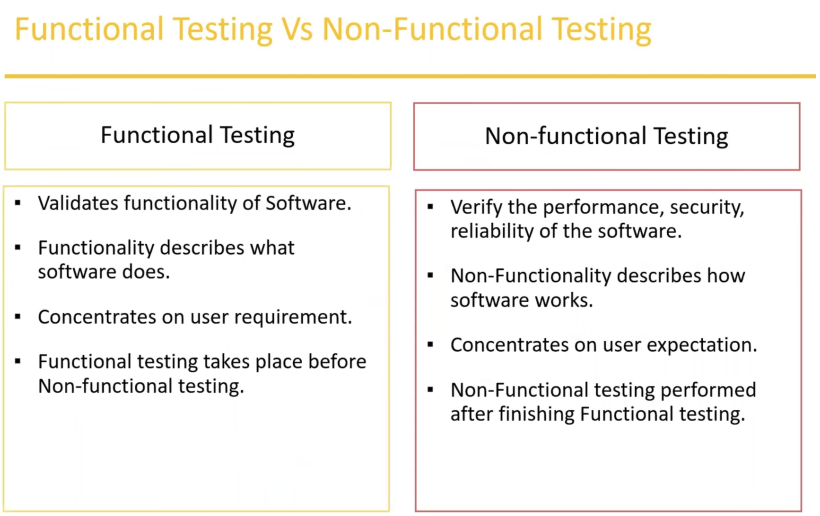
—----------

viii. Sanitation/Garbage Testing

1. If any application provides extra features/functionality, then, we consider them as **BUG**

—----------

—----------



—----------

========5_Manual Software Testing Training Part-5==========

---

1. **Regression Testing**
    a. To ensure that the changed part has not affected the unchanged part
        i. Existing functionalities should not be broken because of new changes or bug fixes
        ii. Example
            1. There are 4 modules in an application
            2. The tester found an issue in Module 2 and reported
            3. The developer resolves the Bug and comes up with a new build for application
            4. For Regression Testing, the Tester needs to verify the functionality for all modules to make sure that the fix has not affected the other modules.
                a. This is **Full Regression Testing**
    b. Regression Testing is conducted on Modified builds to make sure that there is no impact on the existing functionality because of changes like
        i. Adding
        ii. Deleting
        iii. Modifying features

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.

c.

d. Types of Regression Testing
   i. **Unit Regression Testing**
      1. Done by DEVs
      2. Tests only changes/modifications
   ii. **Regional Regression Testing**
      1. Tests the modified module along with impacted module
   iii. **Full Regression Testing**
      1. Tests the main feature and remaining part of the application

—----------

1. **Re-Testing**

Re-Testing

- Whenever the developer fixed a bug, tester will test the bug fix is called Re-testing.
- 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.

a.

b. Re-Testing →
   i. The tester verifies the Bug fix only

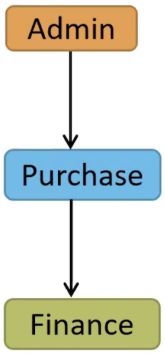
—----------

—----------

1. **Re-Testing vs Regression**

Example: Re-Testing Vs Regression Testing

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



a.

b. Regression →
   i. Validate the whole application (The changed part should not affect the unchanged part)

c. Re-Testing →
   i. The tester verifies the Bug fix only

—----------

1. **Sanity & Smoke Testing**

Smoke Vs Sanity Testing

- Smoke and Sanity Testing come into the picture after build release.

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 no enough time to do in-depth testing.

a.

—----------

—----------

b. Smoke Testing

i. It is **BVT** (Build Verification Test)
ii. It is performed at early/initial stages
iii. Development started → Build v1.0 –> Tester starts Testing with Smoke Testing
iv. Can be done by both DEV and Tester

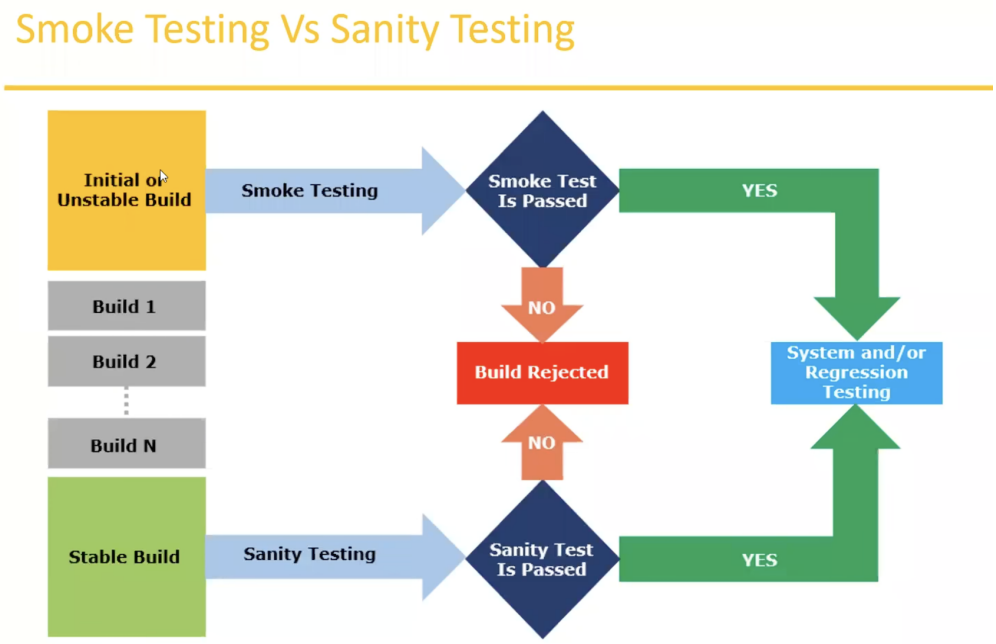
c. Sanity Testing

i. Done by Tester
ii. Focus is on the main functionalities of the application without going deeper

d. The tester can reject the build as well

i. If Smoke Testing and Sanity Testing are not up to the mark

e.



i. Build (1) -> Smoke Testing (Initial Build)
ii. Build (2) -> Smoke Testing
iii. Build (3) -> Smoke Testing
iv. Build (..) -> Smoke Testing
v. Build (..) -> Smoke Testing
vi. Build (..) -> Smoke Testing
vii. Build (n) -> Sanity Testing(Stable Build)

—----------

—----------

1. **Exploratory Testing**
    a. **Tester has to test the application by exploring the functionalities**
        i. Understand the application
        ii. Identify all possible scenarios
        iii. Document it
            1. And, use it for Testing
    b. It is conducted when the application is ready, but with no requirements
    c. **Drawbacks/Limitations**
        i. **The tester might misunderstand**
            1. Any feature as a Bug
            2. Any Bug as a feature
        ii. Time-consuming

—----------

1. **Adhoc Testing**
    a. **Testing the application randomly**
        i. **Without any test case(s)**
        ii. Any Business requirements
    b. It is an informal testing type
        i. Unplanned activity
    c. **The tester should have knowledge of the application**
        i. Previous experience

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.

    d.

—----------

—----------

1. **Monkey/Gorilla Testing**
    a. **Testing the application randomly**
        i. **Without any test case(s)**
        ii. Any Business requirements
    b. It is an informal testing type
        i. Unplanned activity
    c. **The tester does not have knowledge of the application**
    d. Suitable for gaming applications

—----------

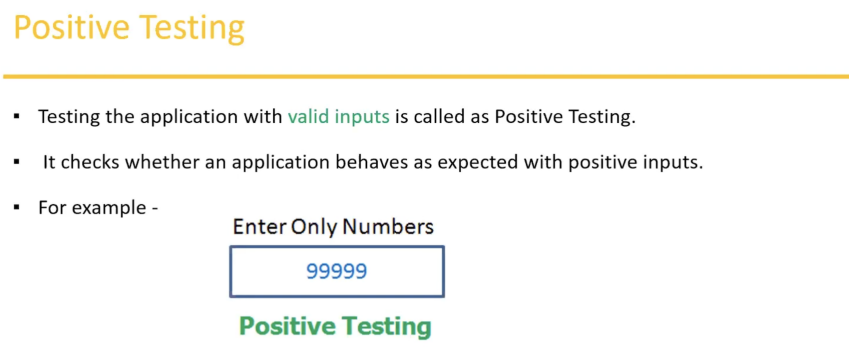
1. **Adhoc Testing vs Monkey/Gorilla Testing vs Exploratory Testing**

a.

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

—----------

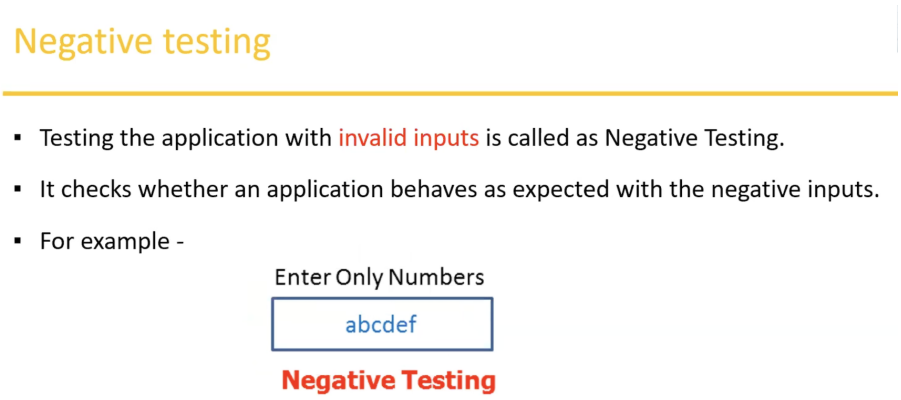
1. **Positive Testing**
    a. Testing the application with **valid inputs**
        i. Checks the behavior (Is application working as expected or not - **Positive** inputs)
    b. Example:
        i. Textbox -> Allows only numbers
            1. Testing it by **giving input as a number only**



c. 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.

—----------

1. **Negative Testing**
    a. Testing the application with **invalid inputs**
        i. Checks the behavior (Is application working as expected or not - **Negative** inputs)
    b. Example:
        i. Textbox -> Allows only numbers
            1. Testing it **by giving input as a Aphabets/Spaces/Special characters only**



c. Negative testing can be performed by entering characters A to Z or from a to z. Either software system should not accept the values or else it should throw an error message for these invalid data inputs.

—----------

—----------

1. **Positive vs Negative Test cases**

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 upto 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.

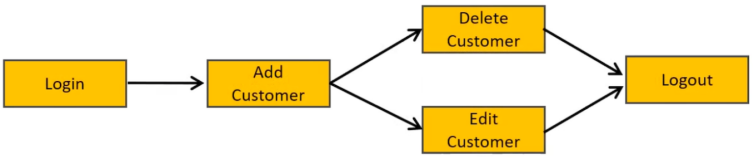
a.

—----------

1. **End-To-End Testing**
   a. Testing the **overall functionalities** of the system that includes data integration among all the modules.

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.



End-To-End Test

1) Login
2) ADD New Customer
3) Edit customer
4) Delete Customer
5) Logout

b.

—----------

—----------

1. **Globalization/Internationalization(I18N) and Localization Testing**

    a. **Globalization/Internationalization(I18N)**
        i. Application is supported globally
            1. Has the support of multiple languages
        ii. Example:
            1. Web application - Google, Facebook
            2. Mobile application - Paypal, WhatsApp

    b. **Localization**
        i. Application is supported locally (Country specific)
            1. Country - China, Japan
                a. Although, these applications are not even accessible from outside the country

    c.

Globalization and Localization Testing

- **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.
- It 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.

—----------

========6_Manual Software Testing Training Part-6==========

---

1. **Test Design Techniques/ Test Data Design Techniques/ Test Case Design Techniques**
    a. These techniques are used to prepare the Test Data.
        i. Test Data - Data for Testing
        ii. This data must cover the different scenarios
        iii. Example
            1. Login functionality test
            2. Data
                a. Valid Username, Valid Password
                b. Valid Username, Invalid Password
                c. Invalid Username, Valid Password
                d. Invalid Username, Invalid Password
    b. Important factors
        i. Reduce Data
        ii. More Coverage
    c. Techniques
        i. Equivalence Class Partitioning (ECP)
        ii. Boundary Value Analysis (BVA)
        iii. Decision Table-based testing
        iv. State Transition
        v. Error Guessing

—----------

—----------

1. **Equivalence Class Partitioning (ECP)**
    a. Partition/Divide/Classify/Categorize Data into various classes
        i. Select the Data according to class
    b. Benefits
        i. It reduces the number of test cases
        ii. Saves time for Testing

c.

Equivalence Class Partition (ECP)

- Partition data into various classes and we can select data according to class then test. It reduce the number of test-cases and saves time for testing.

Enter a Number: [ ] * Allow Digits from 1--500

Normal Test Data	Divide values into Equivalence Classes	Test Data using ECP
1		
2		
3	-100 to 0 → **-50** (Invalid)	-50
4	1 – 100 → **30** (Valid)	30
.	101 – 200 → **160** (Valid)	160
.	201 – 300 → **250** (Valid)	250
.	301 – 400 → **320** (Valid)	320
.	401 – 500 → **450** (Valid)	450
500	501 – 600 → **550** (Invalid)	550

Equivalence Class Partition (ECP)

Name: [ ] * Allow only alphabets

Divide values into Equivalence Classes

	Test Data using ECP
A..Z → (Valid)	XYZ
a..z → (Valid)	zyz
Special Characters → (Invalid)	@#$%
Spaces → **250** (Invalid)	Xy z
Numbers → **320** (Invalid)	1234

d.

—----------

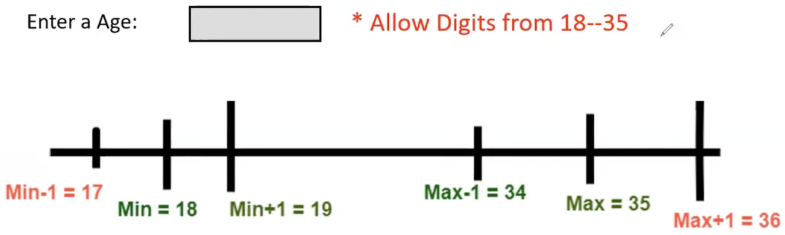
—----------

1. **Boundary Value Analysis (BVA)**
    a. Focus is on the boundaries of the input
        i. Minimum value
            1. Min - 1
            2. Min + 1
        ii. Maximum value
            1. Max - 1
            2. Max + 1

Boundary Value Analysis (BVA)

- BVA technique used to check Boundaries of the input.

b.



Min = 18 (Pass)
Min-1 =17 (Fail)
Min+1 =19 (Pass)

Max = 35 (Pass)
Max-1 =34 (Pass)
Max+1 =36 (Fail)

—----------

NOTE:

1. Equivalence Class Partitioning (**ECP**) and Boundary Value Analysis (**BVA**) are **Input Domain Testing Techniques.**
    a. The provided value will be verified in the input fields.

—----------

—----------

1. **Decision Table-based testing**
   a. Decision Table is also known as **Cause-Effect** table
   b. Conditions and Actions are considered
      i. **Action - Reaction combination**

Decision Table

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

c.

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
  - Account already approved
  - OTP (one time password) matched
  - Sufficient money in the account
- And the actions performed are
  - Transfer money
  - Show a message as insufficient amount
  - Block the transaction incase of suspicious transaction

d.

Decision Table Example...

		TC1	TC2	TC3	TC4	TC5
Condition1	Account already approved	TRUE	TRUE	TRUE	TRUE	FALSE
Condition2	OTP Matched	TRUE	TRUE	FALSE	FALSE	X
Condition3	Sufficient Money in the Account	TRUE	FALSE	TRUE	FALSE	X
Action1	Transfer Money	Execute				
Action2	Show message 'Insufficient Amount'		Execute			
Action3	Block the transaction Incase of Suspicios Transaction			Execute	Execute	X

e.

—----------

—----------

1. **State Transition**
    a. Used when the application has multiple states
    b. Input conditions are changed on the basis of application state
        i. Positive and Negative input values → To evaluate the application behavior

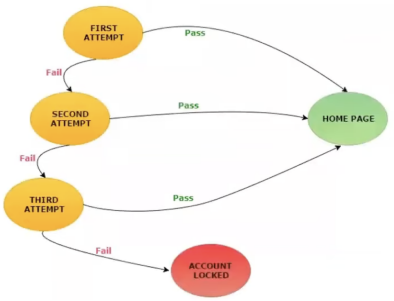
State Transition

- 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.

c.

State Transition Example

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



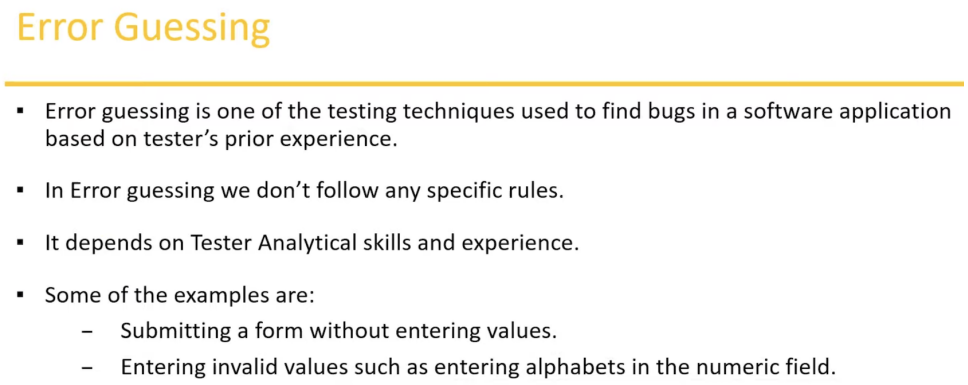
STATE	LOGIN	CORRENT PASSWORD	INCORRECT PASSWORD
S1	First Attempt	S4	S2
S2	Second Attempt	S4	S3
S3	Third Attempt	S4	S5
S4	Home Page		
S5	Display a message as "Account Locked, please consult Administrator"		

d.

—----------

—----------

1. **Error Guessing**
    a. Based on Testers' prior experience and Analytical skills
        i. No specific rules
    b. Example
        i. Submit the form (Without filling in mandatory details)



    c.

—----------

========7_Manual Software Testing Training Part-7==========

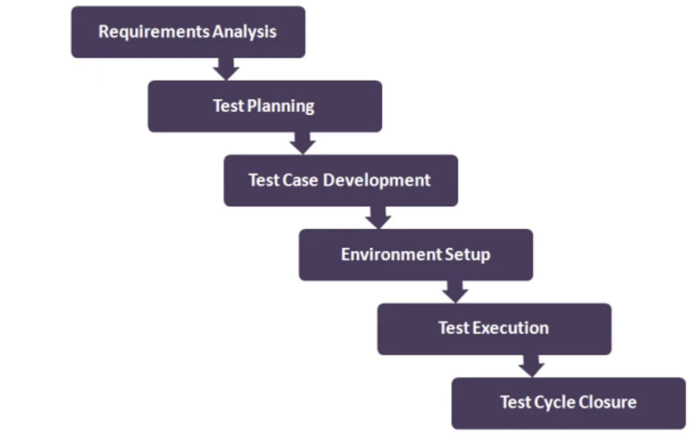
---

1. **SDLC (Software Development Life Cycle)**
    a. Requirements Gathering and Analysis
    b. Design
    c. Coding
    d. Testing
    e. Deployment
    f. Maintenance

—----------

2. **STLC (Software Testing Life Cycle)**
    a. Requirements Analysis
    b. Test Planning
    c. Test Design
    d. Test case Development
    e. Environment Setup
    f. Test Execution
        i. Bug/Defect Reporting and Tracking
    g. Test Closure

h.



i.

STLC

SNO	PHASE	Input	Activities	Responsibility	Out Come
1	**Test Planning**	Project Plan	➢Identify the Resources	Test Lead/Team Lead (70%)	Test Plan Document
	What to test	Functional Requirements	➢Team Formation	Test Manager (30%)	
	How to test		➢Test Estimation		
	when to test		➢Preparation of Test Plan		
			➢Reviews on Test Plan		
			➢Test Plan Sign-off		
2	**Test Designing**	Project Plan	➢Preparation of Test Scenarios	Test Lead/Team Lead(30%)	Test Cases Document
		Functional Requirements	➢Preparation of Test Cases	Test Engineers(70%)	Traceability Matrix
		Test Plan	➢Reviews on Test Cases		
		Design Docs	➢Traceability Matrix		
		Use cases	➢Test Cases Sign-off		
3	**Test Execution**	Functional Requirements	➢Executing Test cases	Test Lead/Team Lead(10%)	Status/Test Reports
		Test Plan	➢Preparation of Test Report/Test Log	Test Engineers (90%)	
		Test Cases	➢Identifying Defects		
		Build from Development Team			
4	**Defect Reporting & Tracking**	Test Cases	➢Preparation of Defect Report	Test Lead/Team Lead(10%)	Defect Report
		Test Reports/Test Log	➢Reporting Defects to Developers	Test Engineers (90%)	
5	**Test Closure/Sign-Off**	Test Reports	➢Analyzing Test Reports	Test Lead/Test Manger(70%)	**Test Summary Reports**
		Defect Reports	➢Analyzing Bug Reporting	Test Enginners(30%)	
			➢Evaluating Exit Criteria		

i. Test Planning

1. Inputs
    a. Project Plan - **How** and **When** to Test?
    b. Functional Requirements - **What** to Test?
2. Activities
    a. Identify the resources
        i. Who will be responsible for
            1. Manual Testing
            2. Automation Testing
    b. Team formation
        i. Recruit people having enough experience
    c. Test Estimation

i. Time is taken for application Testing

d. Preparation of Test Plan
   i. The **test plan is the document that contains all the planning activities**
      1. What to test
         a. Different types of Testing
         b. When will it be conducted
         c. Team members allocation
      2. What not to test
e. Reviews on Test Plan
f. Test Plan Sign-off

ii. Test Designing

1. Input
   a. Functional Requirements
      i. What all features/functionalities have to be tested?
2. Activities
   a. Preparation of Test scenarios and Test cases
      i. We can create n no. of test cases for 1 test scenario
         1. Example - Login is one scenario
            a. Test case - 1 (Valid Username, Valid Password - Should be able to log in)
            b. Test case - 2 (Invalid Username, Valid Password - Should not be able to log in)
            c. Test case - 3 (Valid Username, Invalid Password - Should not be able to log in)
            d. Test case - 4 (Invalid Username, Invalid Password - Should not be able to log in)
   b. Requirements Traceability Matrix (RTM)
      i. Mapping of test cases with Requirements

—----------

1. **Priority**
   a. **P0 ->** Critical
   b. **P1 ->** Major
   c. **P2 ->** Medium (Normal Priority)
   d. **P3 ->** Minor

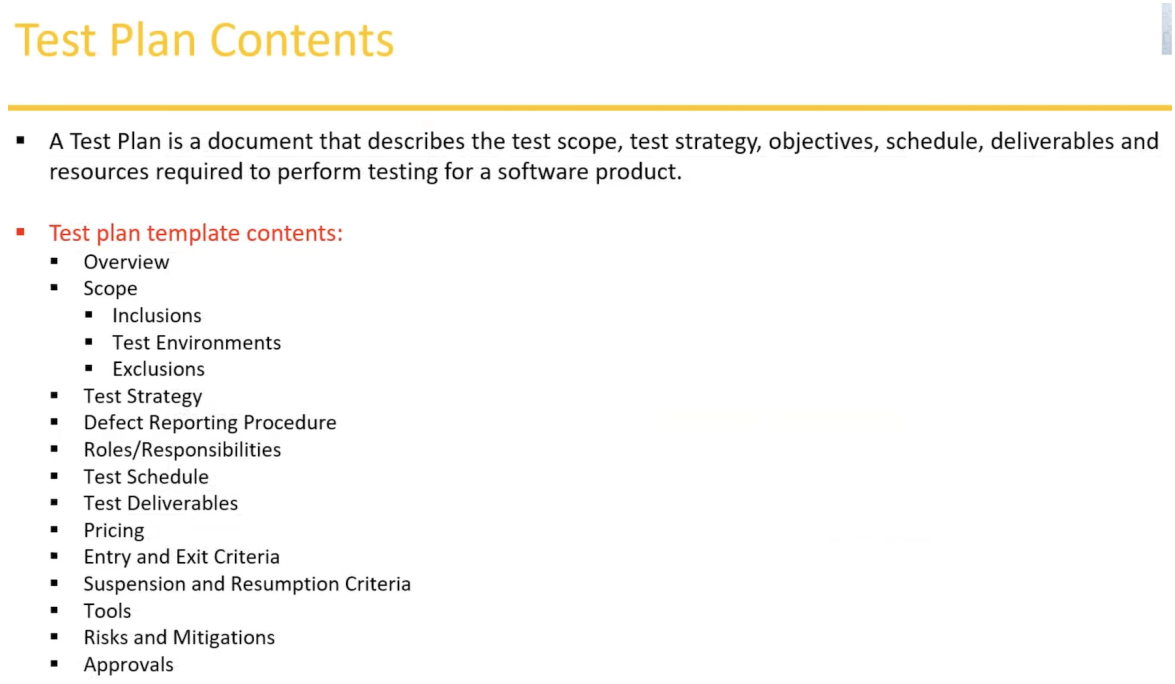
—----------

========8_Manual Software Testing Training Part-8==========

---

1. **Test Plan**

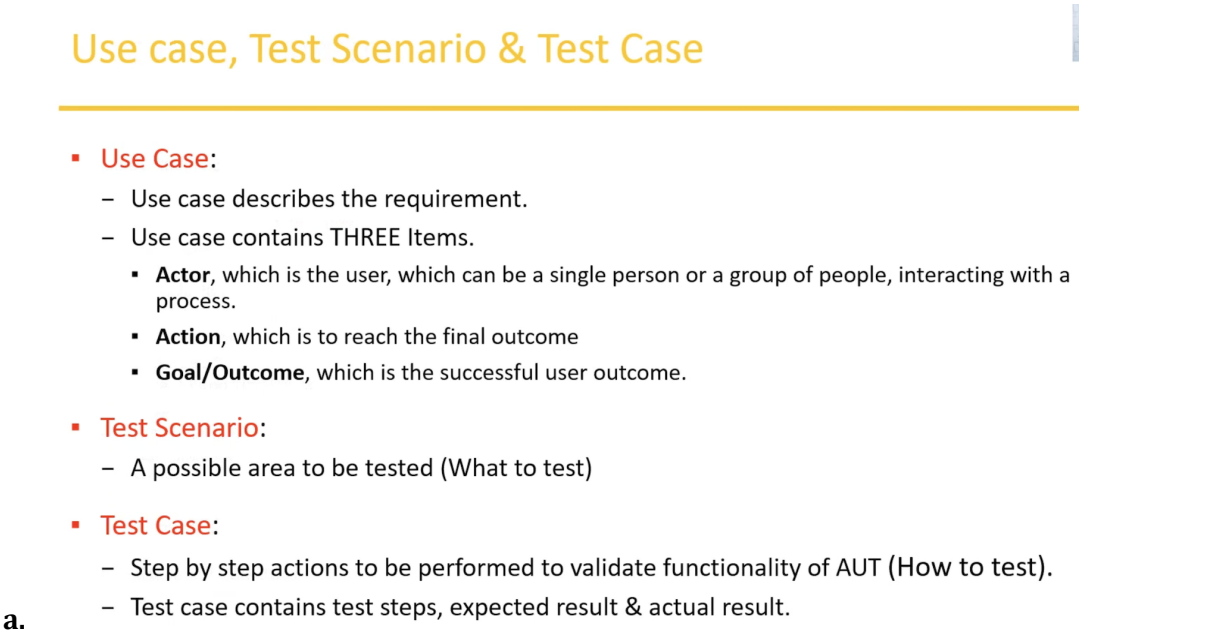
a.



—----------

1. **Use case, Test scenario & Test case**

a.



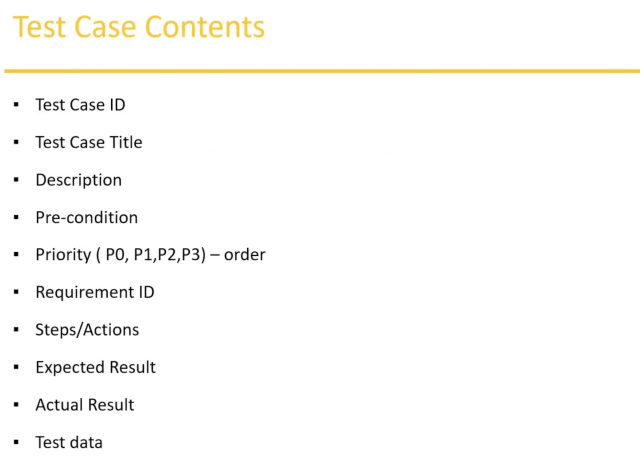
—----------

—----------

i. **Use case**
   1. Describes the requirement clearly using diagrams and data flows
   2. Major
      a. Actor -> User
      b. Action -> Operation
      c. Goal -> Outcome

ii. **Test scenario**
   1. **What to Test?**
   2. Scenario to be tested
   3. 1 test scenario may have n no. of test cases

iii. **Test case**
   1. **How to Test?**
   2. Step-by-Step operation performed to validate the functionality
   3. Contains
      a. Steps
      b. Actual Result
      c. Expected Result
   4. A test case is a set of actions executed to validate the feature/functionality of software/application.



5.

—----------

1. **Use case vs Test case**
   a. Use case
      i. Describes the functional requirement
      ii. Prepared by Business Analyst (BA)
   b. Test case
      i. Describes Test steps
      ii. Prepared by Tester

—----------

—----------

1. **Test case vs Test scenario**

Test Scenario V/s Test Case

- Test Scenario is 'What to be tested' and Test Case is 'How to be tested'.

- **Example:-**
- Test Scenario: Checking the functionality of Login button
  - TC1: Click the button without entering user name and password.
  - TC2: Click the button only entering User name.
  - TC3: Click the button while entering wrong user name and wrong password.

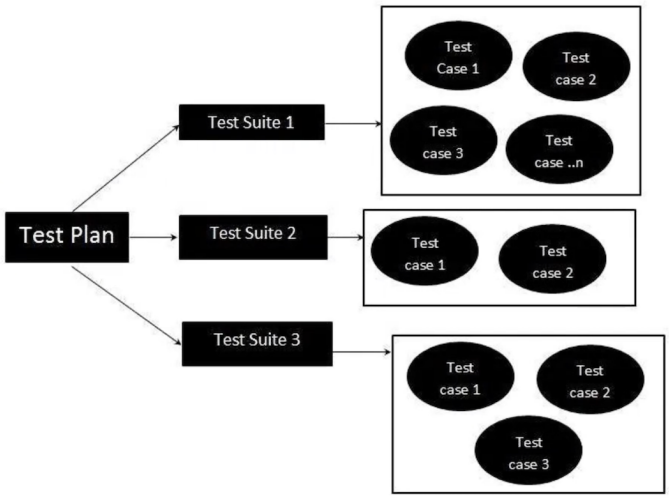
a.

—----------

1. **Test Suite**

Test Suite

- **Test** Suite is group of test cases which belongs to same category.



a.

b. Example

i. Smoke Test cases

ii. Sanity Test cases

iii. Regression Test cases

—----------

—----------

1. **Test case Template**

Test Case Template

b.

Microsoft Excel - Spicejet_BookAFlight _Testcase_2.0

	A	B	C	D	E	F	G	H	I	J	K
1	Module Name	Req ID	TestType	Priority	Test case ID	Test Scenario	Precondition	Testcases / Teststeps	Actual Result	Expected Result	Result
2	Homepage	1	+ve	p1	1	Verify the URL of the home page	1. Open the browser 2. Enter the URL:http://spicejet.com/ Click on Go or Press Enter	1. Open the browser 2. Enter the URL http://spicejet.com/ 3. Click on Go or Press Enter		Browser should navigate to the home page	
3	BookaFlight	2	GUI	P3	1	Verify the GUI of home page	Open the URL http://spicejet.com	1. Check the spell of all the fields 2. Check allignment of all the fields 3. Check the font of all the fields 4. Check the color of all the fields 5. Check the look and feel of the page	Application is maintaining the consistancy	Application should maintain the consistancy	Pass
	BookaFlight	2	+ve	P1	2	Verify the fields in bookaflight page	Open the URL http://spicejet.com	Check the below fields in home page Field Name — Field Type Round trip — Radio Oneway — Radio Leaving From — Dropdown Going To — Dropdown Date Picker1 — Date Picker Date Picker2 — Date Picker Adult — Dropdown Children — Dropdown Infants — Dropdown Indian armed forces personnel Checkbox Student — Check box Find flights — Button	Student discount is displaying instead of sudent	Application should display all the fields in bookaflight page	Fail

Book a flight | Home page | Homepage test metrics | Book a flight test metrics | DSR | Review Report

—----------

1. **Requirements Traceability Matrix (RTM)**

Requirement Traceability Matrix(RTM)

- What is RTM (Requirement Traceability Matrix)?
- RTM describes the mapping of Requirement's with the Test cases.
- The main purpose of RTM is to see that all test cases are covered so that no functionality should miss while doing Software testing.
- Requirement Traceability Matrix – Parameters include
  - Requirement ID
  - Req Description
  - Test case ID's

a.

Sample RTM

Req No	Req Desc	Testcase ID	Status
123	Login to the application	TC01,TC02,TC03	TC01-Pass TC02-Pass
345	Ticket Creation	TC04,TC05,TC06, TC07,TC08,TC09 TC010	TC04-Pass TC05-Pass TC06-Pass TC06-Fail TC07-No Run
456	Search Ticket	TC011,TC012, TC013,TC014	TC011-Pass TC012-Fail TC013-Pass TC014-No Run

b.

—----------

1. **Test Environment**

Test Environment

- Test Environment is a platform specially build for test case execution on the software product.
- It is created by integrating the required software and hardware along with proper network configurations.
- Test environment simulates production/real time environment.
- Another name of test environment is Test Bed.

a.

—----------

—----------

1. **Test Execution**

Test Execution

- During this phase test team will carry out the testing based on the test plans and the test cases prepared.
- **Entry Criteria:** Test cases , Test Data & Test Plan
- **Activities:**
  - Test cases are executed based on the test planning.
  - Status of test cases are marked, like Passed, Failed, Blocked, Run, and others.
  - Documentation of test results and log defects for failed cases is done.
  - All the blocked and failed test cases are assigned bug ids.
  - Retesting once the defects are fixed.
  - Defects are tracked till closure.
- **Deliverables:** Provides defect and test case execution report with completed results.

a.

Guidelines for Test Execution

- The Build being deployed to the QA environment is the most important part of the test execution cycle.
- Test execution is done in Quality Assurance (QA) environment.
- Test execution happens in multiple cycles.
- Test execution phase consists Executing the test cases + test scripts(if automation).

b.

—----------

—----------

1. Defects/Bugs

Defects/Bugs

- Any mismatched functionality found in a application is called as Defect/Bug/Issue.
- During Test Execution Test engineers are reporting mismatches as defects to developers through templates or using tools.
- Defect Reporting Tools:
  - Clear Quest
  - DevTrack
  - Jira
  - Quality Center
  - Bug Jilla etc.

a.

Defect Report Contents

- **Defect_ID** - Unique identification number for the defect.
- **Defect Description** - Detailed description of the defect including information about the module in which defect was found.
- **Version** - Version of the application in which defect was found.
- **Steps** - Detailed steps along with screenshots with which the developer can reproduce the defects.
- **Date Raised** - Date when the defect is raised
- **Reference-** where you Provide reference to the documents like . requirements, design, architecture or may be even screenshots of the error to help understand the defect
- **Detected By** - Name/ID of the tester who raised the defect
- **Status** - Status of the defect , more on this later
- **Fixed by** - Name/ID of the developer who fixed it
- **Date Closed** - Date when the defect is closed
- **Severity** which describes the impact of the defect on the application
- **Priority** which is related to defect fixing urgency. Severity Priority could be High/Medium/Low based on the impact urgency at which the defect should be fixed respectively

b.

—----------

—----------

c.

Defect Classification

d.

Defect Severity

- Severity describes the seriousness of defect and how much impact on Business workflow.
- **Defect severity can be categorized into four class**
  - Blocker(Show stopper): This defect indicates nothing can proceed further.
    - Ex: Application crashed, Login Not worked
  - Critical : The main/basic functionality is not working. Customer business workflow is broken. They cannot proceed further.
    - Ex1: Fund transfer is not working in net banking
    - Ex2: Ordering product in ecommerce application is not working.
  - Major: It cause some undesirable behavior, but the feature/application is still functional.
    - Ex1: After sending email there is no confirm message
    - Ex2: After booking cab there is no confirmation.
  - Minor: It won't cause any major break-down of the system
    - Ex: Look and feel issues, spellings, alignments.

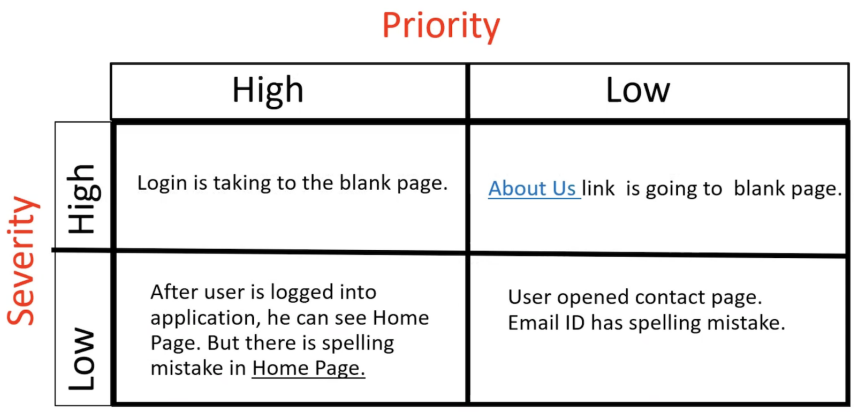
—----------

Defect Priority

- Priority describes the importance of defect.
- Defect Priority states the order in which a defect should be fixed.
- **Defect priority can be categorized into three class**
  - P0 (High) **:** The defect must be resolved immediately as it affects the system severely and cannot be used until it is fixed.
  - P1 (Medium): It can wait until a new versions/builds is created
  - P2 (Low): Developer can fix it in later releases.

e.

High severity, priority and low severity, priority defects



f.

More examples...

- Low priority-Low severity - A spelling mistake in a page not frequently navigated by users.
- Low priority-High severity - Application crashing in some very corner case.
- High priority-Low severity - Slight change in logo color or spelling mistake in company name.
- High priority-High severity - Issue with login functionality.(user is not able to login to the application)
- High Severity- Low Priority - Web page not found when user clicks on a link (user does not visit that page generally)
- Low Priority- Low Severity - Any cosmetic or spelling issues which is within a paragraph or in the page

g.

—----------

h. Defect Resolution

- After receiving the defect report from the testing team, development team conduct a review meeting to fix defects. Then they send a Resolution Type to the testing team for further communication.
- **Resolution Types:-**
  - Accept
  - Reject
  - Duplicate
  - Enhancement
  - Need more information
  - Not Reproducible
  - Fixed
  - As Designed

i.

1. This is the **Developers' opinion**
2. **JIRA - Resolution Type**

—----------

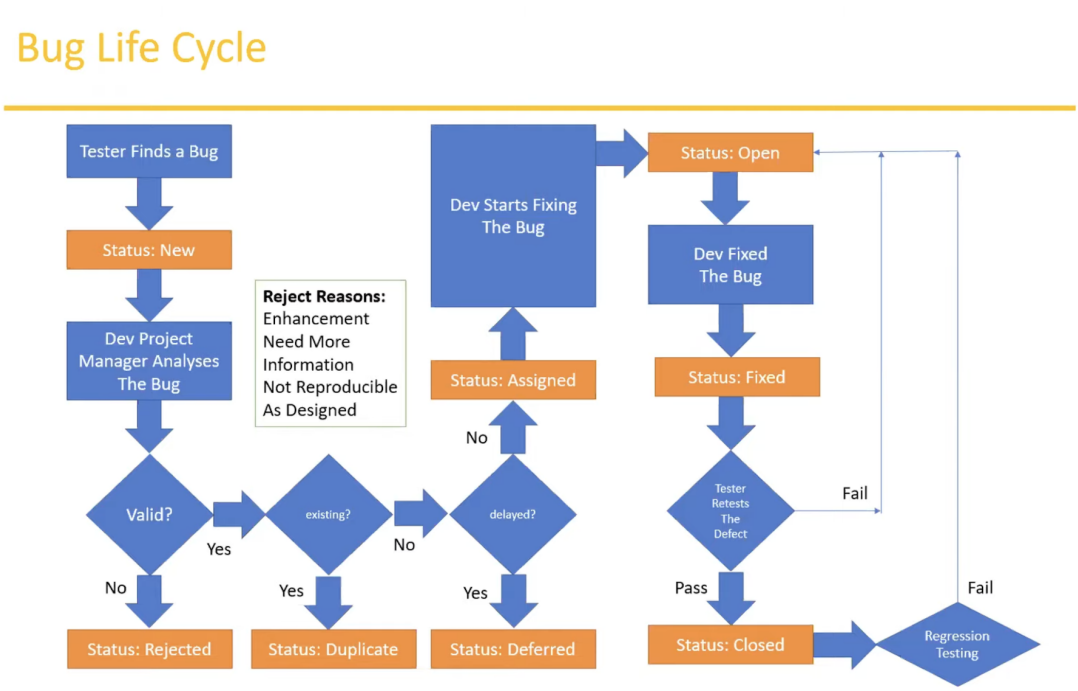
========9_Manual Software Testing Training Part-9==========

---

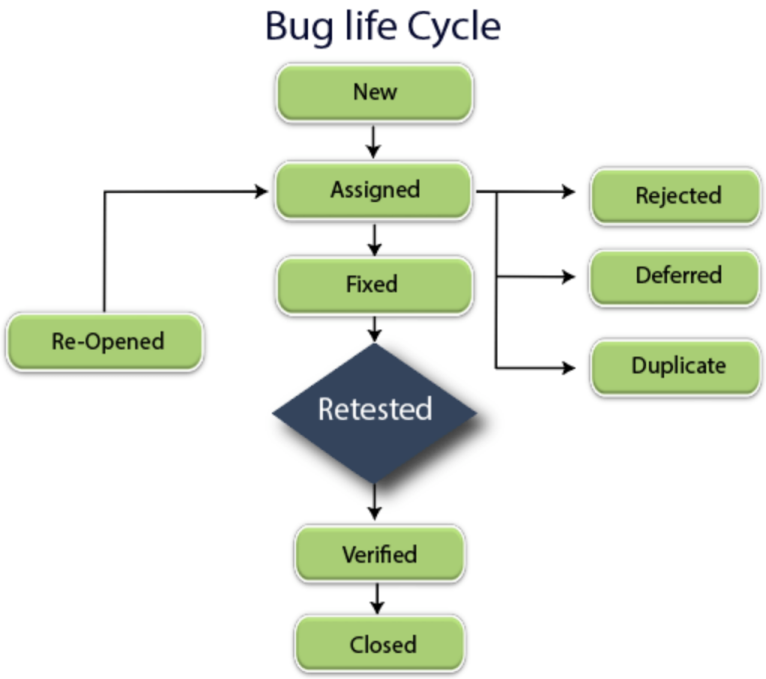
1. **Defect/Bug life cycle**
   - **a.** Status of the Defect/Bug
     - i. New
     - ii. Assigned
     - iii. Open
     - iv. Closed
     - v. Fixed
     - vi. Duplicate
     - **vii. Rejected**
       1. DEV -> Defect is not a genuine defect.
       2. Reasons
          a. Enhancement
          b. Need more information
          c. Not reproducible
          d. As Designed
     - **viii. Deferred**
       1. DEV -> Not of higher priority and can be solved in the next release, then the status changes to Deferred.

—----------

b.



c.



i. https://www.javatpoint.com/jira-bug-life-cycle

—----------

—----------

1. **Test Cycle Closure**

a.

Test Cycle Closure

- **Activities**
  - Evaluate cycle completion criteria based on Time, Test coverage, Cost, Software, Critical Business Objectives , Quality
  - Prepare test metrics based on the above parameters.
  - Document the learning out of the project
  - Prepare Test summary report
  - Qualitative and quantitative reporting of quality of the work product to the customer.
  - Test result analysis to find out the defect distribution by type and severity.
- **Deliverables**
  - Test Closure report
  - Test metrics

—----------

1. **Test Metrics**

a.

Test Metrics

SNO	Required Data
1	No. Of Requirements
2	Avg. No. of Test Cases written Per Requirement
3	Total No.of Test Cases written for all Requirement
4	Total No. Of test cases Executed
5	No.of Test Cases Passed
6	No.of Test Cases Failed
7	No.of Test cases Blocked
8	No. Of Test Cases Un Executed
9	Total No. Of Defects Identified
10	Critical Defects Count
11	Higher Defects Count
12	Medium Defects Count
13	Low Defects Count
14	Customer Defects
15	No.of defects found in UAT

Test Metrics

- **% of Test cases Executed:**
  No.of Test cases executed / Total No. of Test cases written) * 100
- **% of test cases NOT executed:**
  (No.of Test cases NOT executed/Total No. of Test cases written) * 100
- **% Test cases passed**
  (No.of Test cases Passed /Total Test cases executed) * 100
- **% Test cases failed**
  (No.of Test cases failed / Total Test cases executed) * 100
- **%Test cases blocked**
  (No.of test cases blocked / Total Test cases executed) * 100

b.

Test Metrics

- **Defect Density: Number of defects identified per requirement/s**
  No.of defects found / Size(No. of requirements)
- **Defect Removal Efficiency (DRE):**
  (A / A+B) * 100
  (Fixed Defects / (Fixed Defects + Missed defects)) * 100
  - A- Defects identified during testing/ Fixed Defects
  - B- Defects identified by the customer/Missed defects
- **Defect Leakage:**
  (No.of defects found in UAT / No. of defects found in Testing) * 100
- **Defect Rejection Ratio:**
  (No. of defect rejected /Total No. of defects raised) * 100
- **Defect Age:** Fixed date-Reported date
- **Customer satisfaction** = No.of complaints per Period of time

c.

—----------

—----------

1. **QA/Testing Activities**

a.

QA/Testing Activities

- Understanding the requirements and functional specifications of the application.
- Identifying required Test Scenario's.
- Designing Test Cases to validate application.
- Setting up Test Environment(Test Bed)
- Execute Test Cases to valid application
- Log Test results (How many test cases pass/fail).
- Defect reporting and tracking.
- Retest fixed defects of previous build
- Perform various types of testing's in application.
- Reports to Test Lead about the status of assigned tasks
- Participated in regular team meetings.
- Creating automation scripts.
- Provides recommendation on whether or not the application / system is ready for production.

—----------

1. **Principles of Software Testing**

a.

7 Principles of Software Testing

1. Start software testing at early stages. Means from the beginning when you get the requirements.
2. Test the software in order to find the defects.
3. Highly impossible to give the bug free software to the customer.
4. Should not do Exhaustive testing. Means we should not use same type of data for testing every time.
5. Testing is context based. Means decide what types of testing should be conducted based on type of application.
6. We should follow the concept of Pesticide Paradox. Means, if you are executing same cases for longer run, they wont be find any defects. We have to keep update test cases in every cycle/release in order to find more defects.
7. We should follow defect clustering. Means some of the modules contains most of the defects. By experience, we can identify such risky modules. 80% of the problems are found in 20% of the modules.

—----------

==========

==========

1. **More documents:**
    a. https://github.com/rajatt95/Documents

2. **Learnings from Tutor (Code Repository):**
    a. **This course**
        i. https://github.com/stars/rajatt95/lists/youtube-pavan-manual-testing
    b. **Other course(s)**
        i. https://github.com/stars/rajatt95/lists/youtube-pavan-kumar

3. **To connect**
    a. https://github.com/rajatt95
    b. https://rajatt95.github.io/
    c. https://www.linkedin.com/in/rajat-v-3b0685128/

==========