Full Stack QA - Software Testing Introduction: 13/12/2023

Full Stack QA → Manual Testing
- Java
- Selenium
- Maven, Jenkins.
- Git, Github, Docker etc.

☐ Manual Testing:
- Module 1: Testing Concepts (What)
- Module 2: Testing Project (How).
- Module 3: Agile Process - Jira
  - Agile test management tool used to track all the steps of SDLC (Software Development Life Cycle).

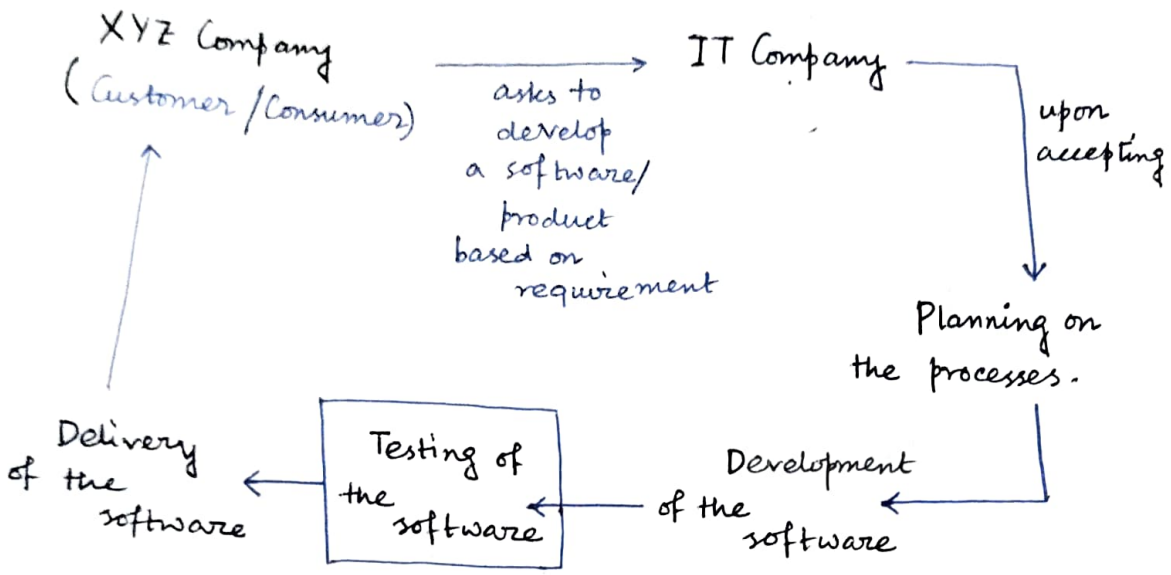
☐ Module 1: Testing Concepts: [Section 1: Software Testing Introduction - Part I]

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

Software
- System Software → Softwares required to run the system, Ex: Device drivers, Operating system, servers, ultilities etc.
- Application Software → Softwares that are used by the users including various applications.
  Eg: Web applications, Mobile Apps, Desktop applications etc.
- Programming Software: Softwares that are used to execute or debug the programs written for specific tasks are known as programming softwares
  Eg: Compilers, debugger, interpreters etc.

☐ What is software testing?

General flow of software development:



☐ Software testing:

① Part of software development process

② It is the activity to detect and identify the defects in a software.

Objective: release quality product to the client

based on customer justification (of all the requirements)

☐ Parameters of Software Quality:

① Bug-free

② On-time delivery

③ Within budget

④ Meets client requirements/expectations

⑤ Maintainable → once installed in the consumer environment the customer shall be able to maintain or handle minor issues

⊙ Project: Software application developed for a specific customer based on their requirements is known as project (mainly developed by service based companies) → TCS, Cognizant, Accenture etc.

⊙ Product: Software application developed for multiple customer based on the market requirements is known as product (mainly developed by product based companies). → Google, Microsoft, Oracle etc.

☐ Why do we need testing? (Objectives of software testing).

1. to ensure bug-free software
2. to ensure that system meets customer requirements and software specifications.
3. to ensure that system meets end user expectations.
4. fixing the bugs after release is more expensive.

☐ Error, bug/defect, Failure:

A. Error: Any incorrect human action that produces any problem in the system is called an error.

B. Bug/defect: Any deviation from the expected behaviour to the actual behaviour of the system is called a bug/defect.

C. Failure: Any deviation identified by the end-user while using the system is called a failure.

☐ Why the softwares have bugs?

1. Miscommunication or no communication between developer, tester etc.
2. Software complexity
3. Programming errors
4. Changing requirements
5. Lack of skilled testers.

[Section 2: Software Testing Concepts][Lecture 2]

□ Software Development Life Cycle (SDLC):

↳ Process used by software industry to design, develop and test softwares.

⊙ 3 P's of any company:

1) People
2) Process
3) Product

□ Phases of Software Development Life Cycle:

① Requirement Analysis: Collect and understand the requirement of the customers.

② Design: Designer designs the software based on customer requirement.

③ Development: Developer writes various codes / programs.

④ Testing: Testing of various features of software before releasing.

⑥ Maintenance: End user starts using the software, maintenance starts.

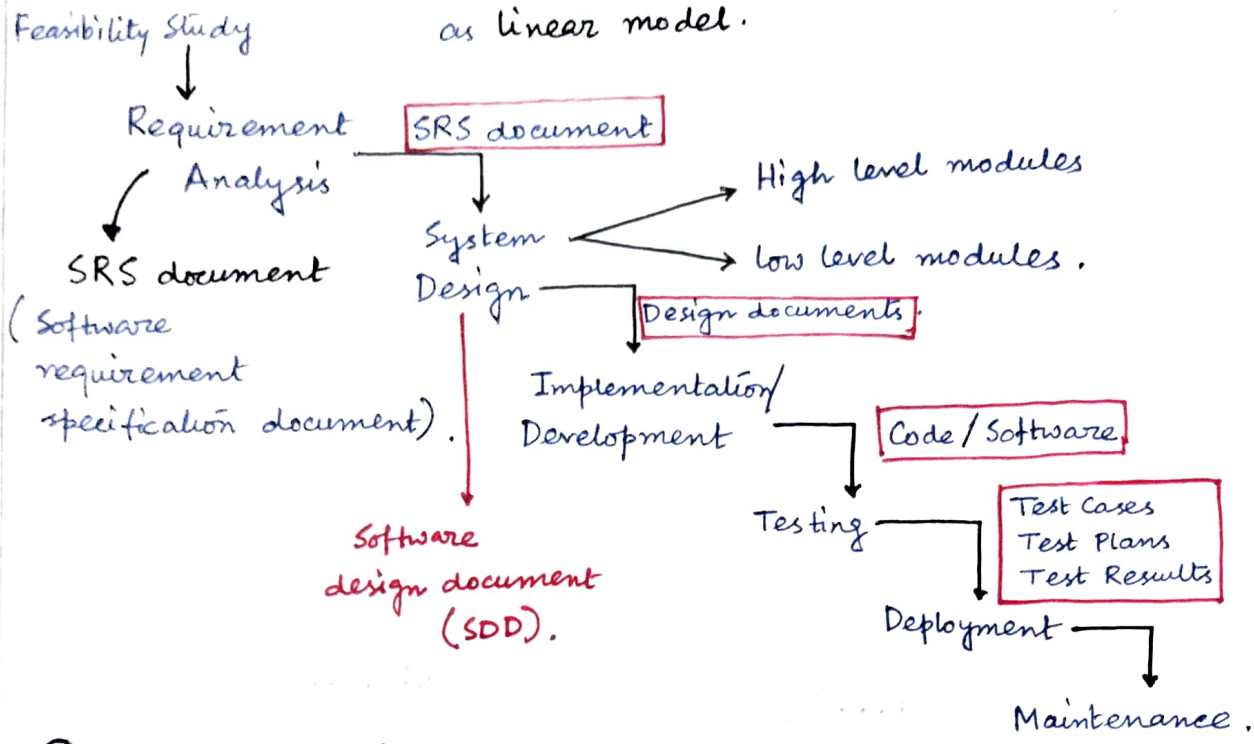
⑤ Deployment: Delivering the software to the customer.

□ Requirement of SDLC:

① Ensure quality software
② Manage risks and cost
③ Improve efficiency and productivity

☐ Various models followed in SDLC:

1. Waterfall Model: Very old and traditional model, also known as linear model.



⊗ Documentation is very important in each stage of this model.

Features:

① Every phase requires an input from the previous phase and provides an outcome.

② Every phase dependant on the previous phase.

☐ Advantages of Waterfall Model:

(1) Quality of the project/product will be good.

(2) As requirement change is not allowed, chances of finding bugs are less.

(3) Initial investment is low, as testers are hired at later stages.

(4) Generally followed for small projects where requirements are freezed.

☐ Disadvantages of Waterfall Model:

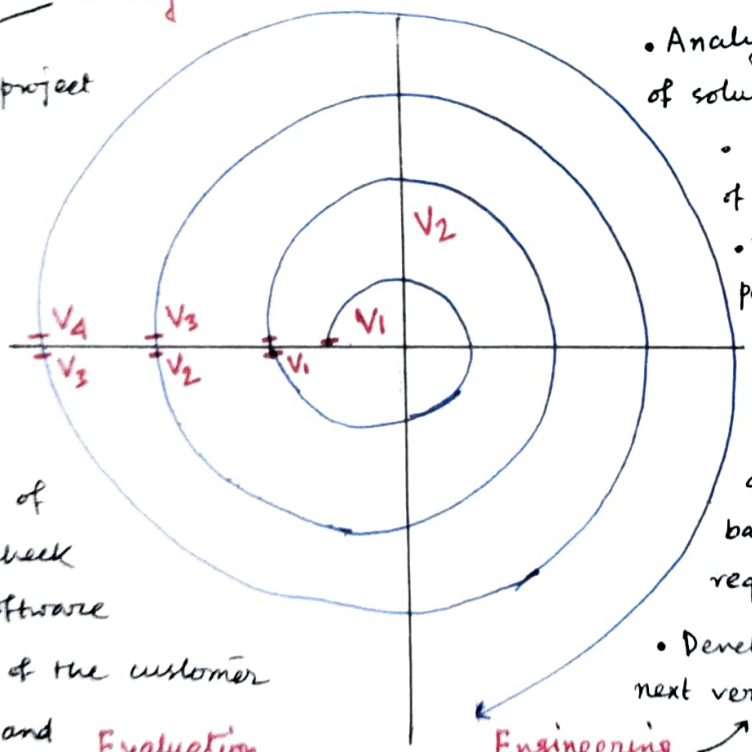
(1) Requirement changes are not allowed.

(2) If there is any defect in requirement, it will accumulate and continue to later phases.

(3) Total investment is very high, as if any defects are found, the cost of rework is high and also time consuming.

(4) Testing can be started only after coding is completed.

☐ Spiral Model:



- Scope of the project
- Requirement analysis
- Planning of the next iteration

- Analysis of/Evaluation of solutions.
- Risk analysis of the solutions
- Prototype of best possible solution

- Evaluation of software to check if it meets software requirements of the customer
- Quality check and assurance
- Customer evaluate the current version.

- Software development based on the requirement
- Development of the next version
- Testing of the software.

Features of Spiral Model:

① It is an iterative model. Also known as version-control model.

② It overcomes the drawbacks or limitations of the waterfall model.

③ New version of software is released after each circle.

④ The software is released in multiple versions.

⑤ This model is implemented when there is a dependency on the modules.

↳ the software released after each version can be called as modules.

Advantages:

① Testing is done ~~after~~ during each cycle, then we continue with the new cycle.

② Customer gets to use the software in every cycle (every version of the software)

③ Requirement changes are allowed after completing one cycle and before begining of the next cycle.

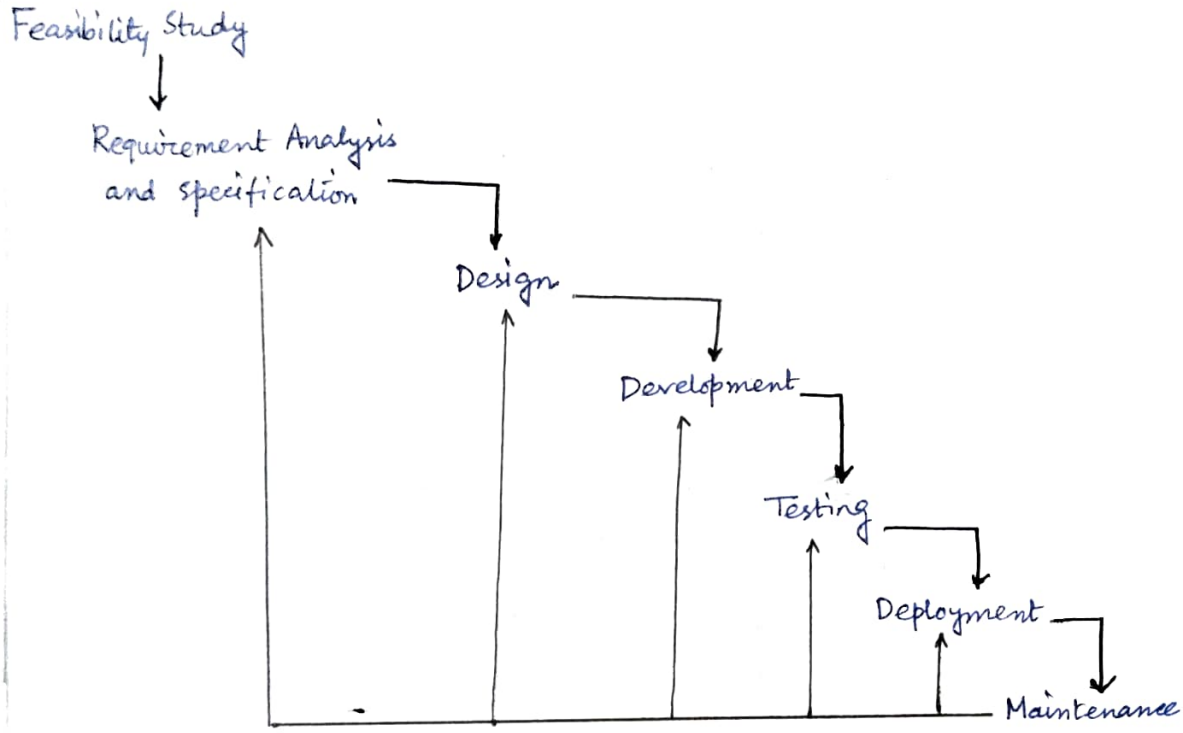
Disadvantages:

① Requirement changes can be done after completion of a cycle only, not in between

② Each cycle of the spiral model follows the waterfall model only.

③ There is no testing during the requirement and design phase.

☐ Iterative Waterfall Model: In a practical environment of software development, the use of classical waterfall model is not very useful. To overcome this problem, feedback paths was introduced along with waterfall model, from every phase to its preceding phase. This is the main difference.



When errors are detected at a later stage, these feedback paths allow for correction of the errors committed by the programmers.

The feedback path allows to rework the errors in the phase of occurance and it is reflected on the later stages.

□ Advantages:

① Feedback path

② Cost effective → to change plan/requirements

③ Well organized

④ Reduced risks

⑤ Improved customer satisfaction

⑥ Greater control over development process.

□ Disadvantages:

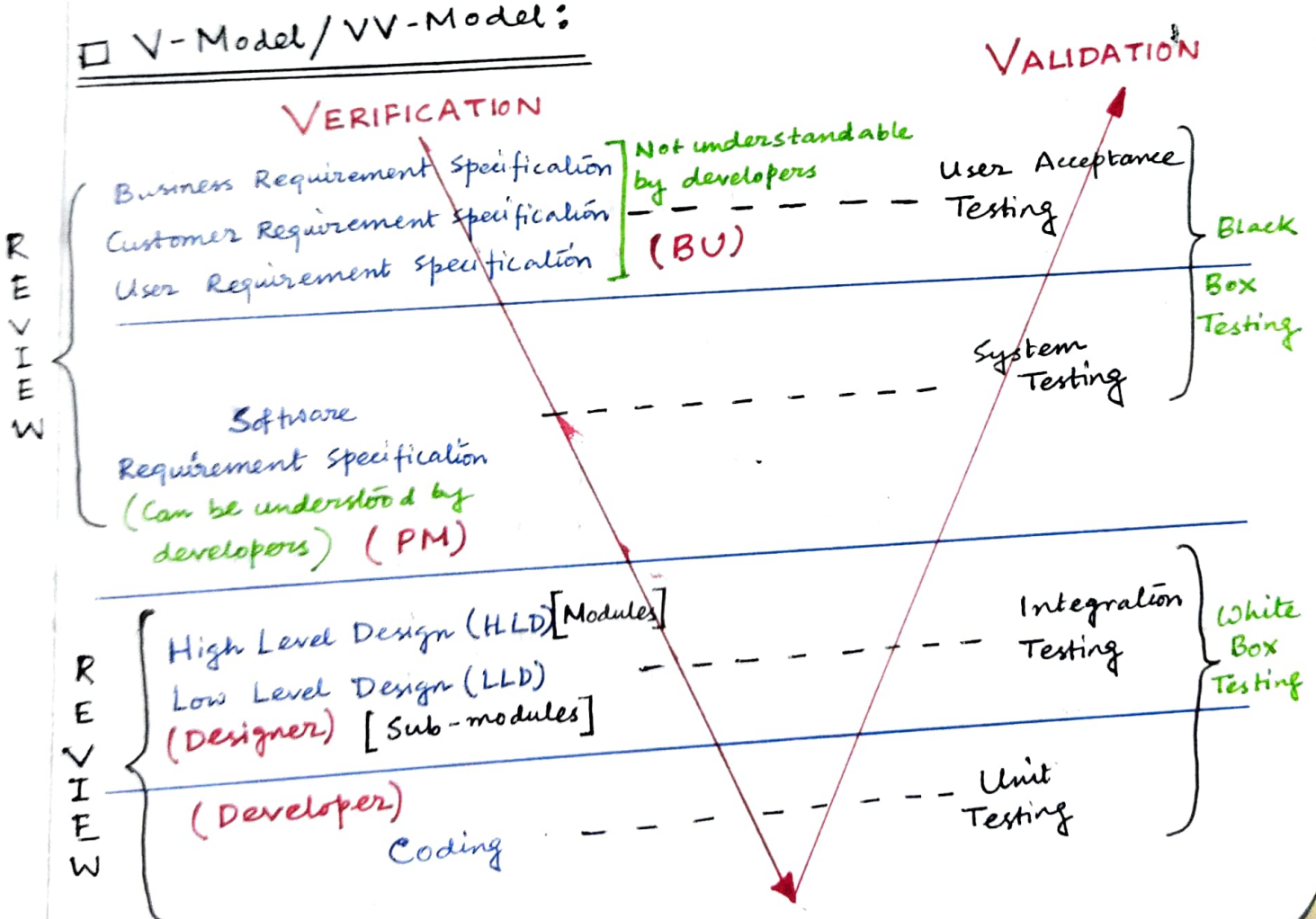
① Difficult to incorporate change requests.

② Incremental delivery not supported → software delivered upon fully developed.

③ No overlapping of phases allowed

④ No mechanism for risk handling.

□ V-Model / VV-Model:



☐ Static Testing: Testing the project related documents are called static testing.

→ ① Review
② Walk through
③ Inspection

☐ Dynamic Testing: Testing of the actual software.

→ Unit testing ①
② Integration Testing
③ System Testing
④ User acceptance testing (UAT).

Verification	Validation
① Checks whether we are building the right product	① Checks whether we are building the product right (in the right way)
② Happens before the software is built.	② Happens after the software has been built.
③ Focuses on the document-ation part	③ Focuses on the software
④ Testing process follows review, walkthrough, inspection	④ Testing process follows Unit testing, integration testing, System testing, user acceptance testing

☐ Advantages of V-Model:

① Testing is done in every stage.

☐ Disadvantages of V-Model:

① Documentation is more

② Initial investment is more.

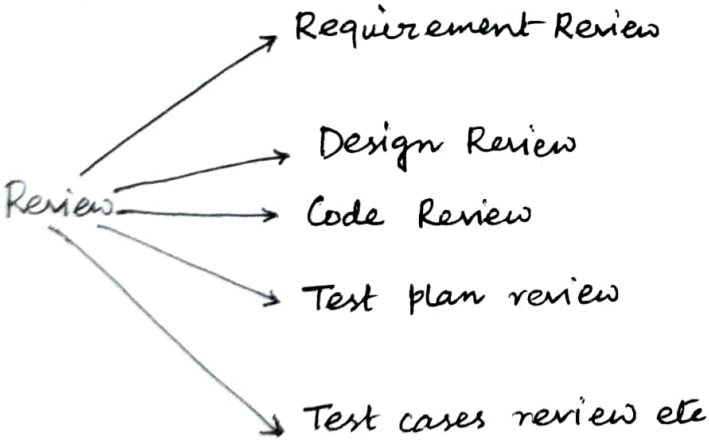
Section 2: Software Testing Concepts [Lecture 3]

☐ Static Testing Methods:

- Review
- Walk through
- Inspection

☐ Review: Focused mainly on documentation.

↳ Conducted on documents to check its correctness and completeness.



□ Walkthrough: also focused on documentation

↳ it is more like an informal review, therefore no meeting is needed to be arranged to hold a walk-through.

⊙ During walk-through, the author reads the documents and discusses with peers

⊙ Not pre-planned and can be done on requirement basis

⊙ Does not require the minutes of the meet

□ Inspection: also focused on documentation

⊙ Its the most formal type of review.

⊙ Here 3-8 members, preferably all the team members will attend the meeting.

3 Roles → (1) Reader: generally author of the documents, will read it out to the team.

(2) Writer: generally will write down the outcomes and minutes of the meeting

(3) Moderator: generally the organizer of the meeting.

⊙ Pre scheduled meeting, intimated through email previously

⊙ Minutes of the meet is noted

- Dynamic Testing: focused on the software aspects.

① Unit testing: developer tests the code units/modules.

② Integration Testing: Different module of the software is integrated and checked if they work properly and meet customer requirements. This is also generally done by developers.

③ System testing: Generally testers perform these tests. Before deployment of the software, the whole software is tested to check if it meets the customer requirements. Bug check is also done in this step.

④ User acceptance testing: (UAT): After deploying the software, users use the software and test it. It is generally performed by the testers, who setup the software environment for the customers and monitor the software.

QA, QC, QE:

① QA (Quality Assurance): Quality assurance is process-related. Generally the top level management of a company is the people maintaining QA.

They: ① define the processes (from requirement to deployment).
② ensure that others are following/adhering to the set standard of processes.

QA is involved in every stage of SDLC.

③ QA focuses on building in quality → to maintain the quality in each stage of SDLC.

④ QA is for preventing the defects from occuring.

⑤ QA is process oriented.

⑥ QA shall be done through the entire life cycle.

Quality Control (QC): Quality control is the actual testing of the software.

QC is related to people (mostly testers), and are involved in the testing part.

They: ① verify and validate the software.

② QC focuses on testing for quality.
Checks if the software developed meets the quality standards and requirements. [find bugs etc.]

③ QC is for detecting the bugs/defects. and the report/feed back to the developers,

④ QC is product oriented.

⑤ QC is for the testing phase of SDLC.

⑥ QC is just one part of QA.

Quality Engineering (QE): Writing different codes to test the software.

People involved in writing the code for testing purpose of the software are known as quality engineer.

Generally, QE contains the automation testers.

□ Levels of Software Testing:

Unit testing → Integration testing → System testing → User acceptance testing (UAT).

① Unit testing: small module/component of the software.

↳ during unit testing, developers test the small units of the code of the software.

② Integration testing: Different units are pieced together and checked if they align and work properly together. This is known as integration testing.

③ System testing: testing the overall functionality of the application.

to check if the software meets all the customer requirements.

④ UAT (User acceptance testing): after completion of system testing, the testers along with customers perform UAT.

□ **Unit Testing:** As soon as the coding for application is started, the developers themselves also start unit testing simultaneously.

Unit: Single component or a single module of the software.

Features: ① Unit testing is conducted on a single program or single module.

② White box testing technique.

③ testing conducted on the program itself

④ conducted by developers.

□ **Unit Test Techniques:**

① **Basis path testing:** testing method where it is checked that every line of code is executed at least once.

② **Control Structure testing:** Different control structures are tested through this test.

A. Conditional coverage: Developers check if all the conditional statements used in the code is executing properly. [if, if-else, if-else-if etc.]

They check with all possible ~~unit~~ inputs for the conditions.

B. Loops coverage: It is checked if all the loops used in the codes are working properly [while, for, do while etc.]

③ Mutation testing: testing the code with multiple sets of data - repititive testing.
↳ different types of data. using different combinations.

□ Integration Testing:

① Performed between 2 or more modules

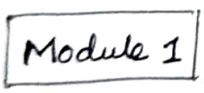
② focuses on checking the data communication between 2 or more modules.
↳ data flow between the modules.
(one module will give some output which will act as the input for the next module).

③ White box testing.

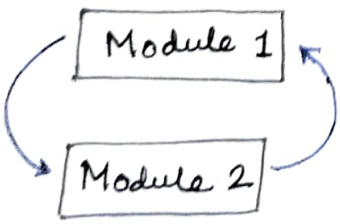
Integration Testing → Incremental Integration Testing
→ Non-incremental Integration Testing

① Incremental Integration Testing: Incrementally adding the modules of the software and testing if they are integrating properly.

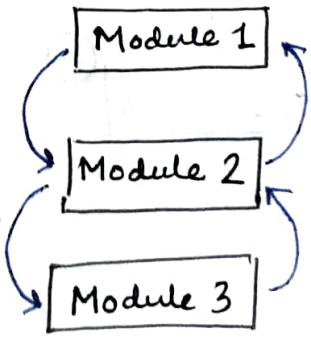
Stage 1:



Stage 2:



Stage 3:



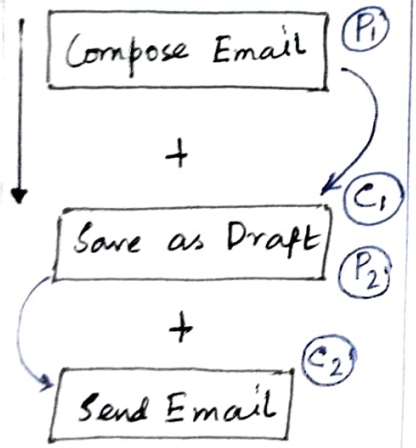
Incremental Integration Testing

Top-down approach

① Incrementally add the modules and check the data flow between the modules

② Ensure the module added, is the child of the previous module

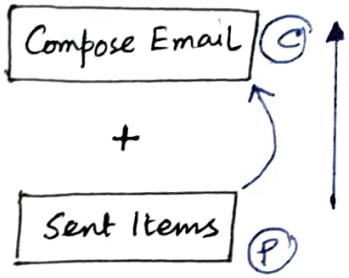
Example can be understood by the example of Gmail application:



Bottom-up approach

① Incrementally adding the modules and check the dat flow b/w the modules

② Ensure the module added, is the parent of the previous model



Hybrid Approach / Sandwich approach

Combination of Top-down and bottom-up approach.

☐ Non-Incremental Integration Testing: Adding all the modules in one go and test the data flow between the modules.

But generally this approach is not preferred due to its drawbacks:

① we might miss the data flow between some of the modules

② If any defect is found, it is hard to detect which module has the defect or the root cause of the defect.

***

☐ System Testing: Testing the overall functionality of the software with respect to client/customer requirements.

Features: ① It is a black box technique

② This testing is conducted by the testing team.

③ 3rd level of testing. Can be done after Unit testing and integration testing only.

④ It is important to be aware of the customer requirements, before performing system testing.

☐ Aspects of System Testing:

① Graphical User Interface (GUI) testing

② Functional Testing

③ Non-functional Testing

④ Usability Testing.

☐ User Acceptability Test: (UAT): generally done by users/customers. Testers may assist them.

☐ Alpha Testing: user does the testing in development environment

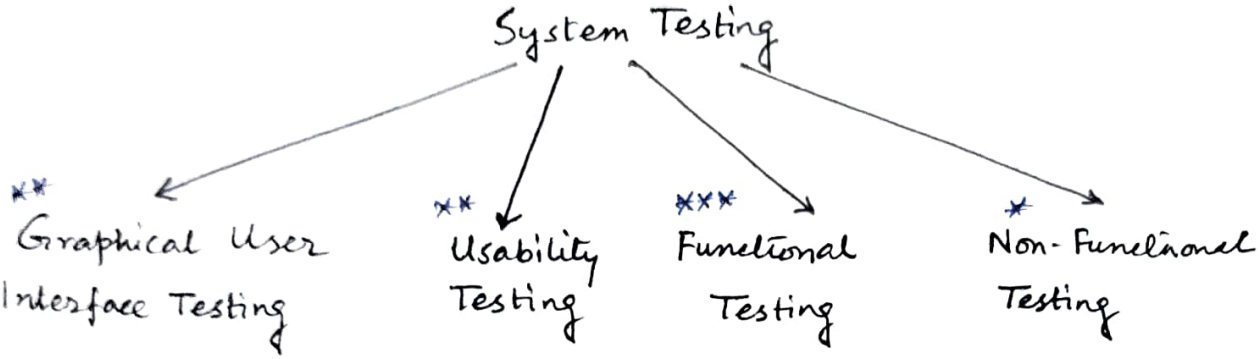
☐ Beta Testing: user does the testing in their own environment

Dt: 18/12/2023

☐ Section 2: Basics of Software Testing – Part 1:

☐ **System Testing:** testing overall functionality of the software/ application based on customer requirements.

Performed only after completing unit testing and integration testing.



① **Graphical User Interface (GUI) Testing:** process of testing the user interface of an application.

GUI → menus, checkboxes, buttons, color, fonts, sizes, icons, content, images etc.

☐ GUI Testing Checklist:

1) testing the size, position, width, height of elements.

2) testing of the error messages that are getting displayed.

3) testing the different sections in the screen (logo, header, footer etc.)

4) testing the readability of the font.

5) testing of the screen in different resolutions, by zooming in or out.

6) testing the alignment of the texts and other elements like icons, buttons etc for proper placement.

7) testing the colors of the fonts (avoid very bright colors).

8) testing whether the image has good clarity or not.

9) testing the alignment of images

10) testing of the spelling

11) user shall not get frustrated using the UI.

12) attractiveness of the UI

13) testing of the scrollbars according to the size of the page if any.

14) testing of the disabled fields if any.

15) testing of the size of the images.

16) testing of the heading, whether is properly aligned or not.

17) testing of the color of the hyperlink.

18) testing UI elements like button, textbox, textarea, check box, radio buttons, drop downs, links etc.

□ Usability Testing:

① check how easily the end users are able to understand and operate the application, is called usability testing.

② this testing validates application provided context sensitive help or not to the user.

□ Functional Testing:

◇ Functionality is nothing but the behaviours of the application.

◇ Functional testing talks about how your feature should work (according to customer requirements or not).

① Object Properties Testing:

Object → any element on the UI can be called an object.

Properties → can have various properties as per requirements:

for example: ① A textbox can have properties as follows:

textbox: [ ]
- enabled → user can enter text
- disabled → user can't enter text.

② Radio buttons:

Gender: ○ M ○ F

→ while one option is selected, the other shall get disabled automatically.

③ Dropdown:

Country [▽]

- India
- Indonesia

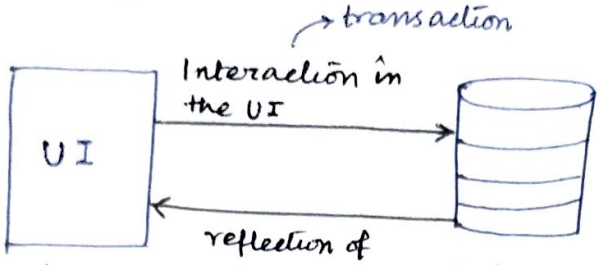
→ only one option shall be selectable from the drop down.

These are various properties of different objects and shall be checked as part of object properties testing.

② Database Testing: → Main focus on DML operations.

(Backend testing). → to check if user interaction of selecting, inserting, updating or deleting data, properly reflects on the database or not.

→ grey-box testing technique.



(Part of black box testing, as only user interface is being tested.)

Database (white box testing, as we write SQL queries to perform the checks).

\+

hence can be termed as grey-box testing

also the following tests are done in database testing:

- Table level validations (column type, column length, no. of columns etc.)
- Relation between the tables (normalization)
- Functions
- Procedures
- triggers
- Indexes
- views etc.

③ Error handling testing: if the error messages are showing properly when any unauthorized action is performed by the user. (verying error messages in the UI).

- ⊙ Error messages should be readable
- ⊙ " " shall be in user understandable language.

④ Calculation/Manipulations Testing: checking if the numerical calculations are working properly or not.

- ⊙ the checks shall be done with both valid and invalid data.

⑤ Links existance and Links execution:

↳ where exactly the links are placed

↳ on clicking the link, if it is properly working/redirecting or not.

Links → Internal links → another section in the same page
Links → External links → linked to a different page
Links → Broken links → won't navigate to anywhere, placed for future developments (have no target page).

⑥ Cookies and Sessions: → time slots created by the server.

↳ temporary files created by browser while browsering the pages through internet. in cookies checking we check if the browser is creating the cookies or not.

Sessions: when we open an application, sometimes it creates an instance of the application and a countdown starts on the server side. Once the countdown is crossed due to inactivity, the application closes from the server side. This are called sessions.

□ Non-Functional Testing: focused on customer expectations.

⊙ performed after functional testing is completed.

1) Performance testing:

↳ speed of the application or how quickly the application responses to the user requests.

Performance testing (generally done by creating virtual users in the required environment).

- Load testing: increasing the load on the application slowly and check the speed of the application. (checking the response time).
- Stress testing: increasing or decreasing the load on the application suddenly and check the response time (speed) of the application. (also stability).
- Volume testing: how much data our application is able to handle.

(2) Security Testing: how secure our application is from internal and external threats.

- authentication → checking if the users are valid or not
- authorization / access control: → how much access the user has access to in the application or ~~of~~ authorized to. (permissions of valid user).

eg: Admin has full control, whereas end-user has specific access.

(3) Recovery testing: If the application has recovery option or not. (mechanism)

↳ check the system change from abnormal to normal.

(4) Compatibility Testing: To check if our application is compatiable to different scenerios or not.

→ Forward compatibility: if our application is compatiable when upgrading to a newer version (the newer version works properly or not).

→ Backward compatibility: if our application's older version is compatiable or not is checked.

→ Hardware compatibility: If the application can be installed on different hardware environments or not. Also known as configuration testing.

(5) Installation Testing: If the software installs properly or not. Also, if the installation screens are working as expected or not. Checking the ease of installation instructions in the installation wizard is also checked. Also check for the uninstallation process.

(6) Sanitation / Garbage Testing: If the application offers any extra functionality than the customer requirement, then that is also considered as a bug.

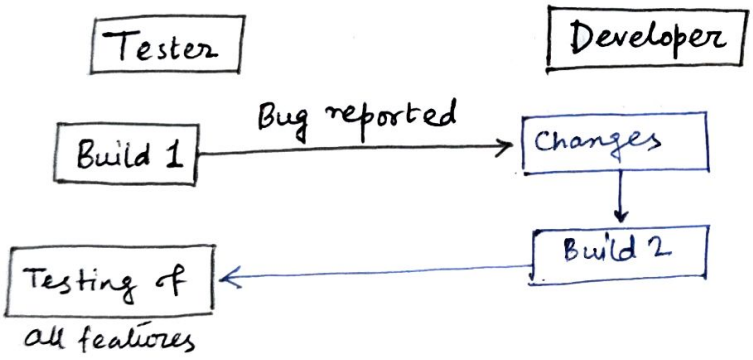
Removing these unwanted features is known as sanitation and garbage testing and is checked in this stage.

Functional Testing	Non-Functional Testing
(1) Validates functionality	(1) Verify the performance, security, reliability etc.
(2) describes what the software does	(2) describes how the software works
(3) Concentrates on user requirements	(3) Concentrates on user expectations.
(4) happens before non-functional testing	(4) happens after functional testing.

Section 2: Basics of Software Testing: (Part 5):

Software Testing Terminology:

Regression Testing: During these testing, tester will check for bug fixes, other modifications in the application, check newly added feature or deleted ones.



Once one bug is fixed, it shall not affect the functionality of the other dependant modules. This is the main focus of regression testing.

Testing conducted on modified build to make sure there will be no impact on existing functionality because of changes like adding / deleting / modifying features.

Regression Testing

Unit regression testing
- ⊙ testing only the changes done by the developers.

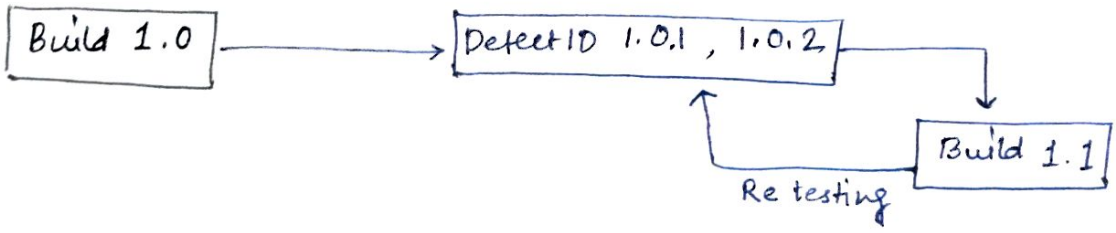
Regional regression testing
- ⊙ testing the modified module, along with impacted module
- ⊙ impact analysis meeting conducted to identify the impacted modules with QA and Dev.

Full regression testing
- ⊙ testing the main feature and remaining part of the application
- ⊙ generally performed when most of the modules undergo modifications

□ **Re-testing:** ~~very~~ verifying the previously reported bugs have been resolved or not. The testing is only done to check bug-fixes.

Tester will close the bug if it has been fixed, or reopen and send to dev.

This ensures that defects posted earlier is fixed or not in the current build.



☐ Re-testing V/s Regression Testing:

Let us assume that the application being tested, has three modules. Admin, Purchase, Finance

So here the Finance module is dependant on the purchase module.

Now, suppose a bug has been posted by the tester on the Purchase module. So any modifications on purchase module will also reflect on the finance module. After the dev fixes the posted bug:

→ tester tests the Purchase module → this is known as re-testing.

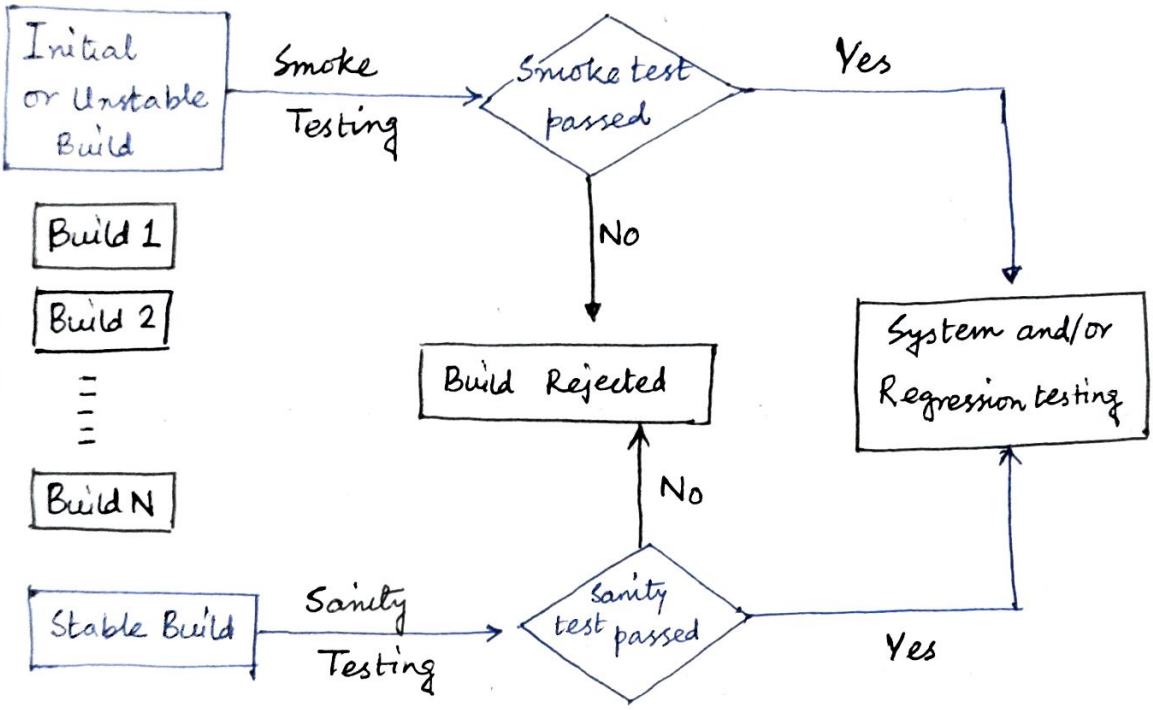
→ after testing the bug fix, the finance module shall also be tested
↳ this is known as regression testing.

☐ Smoke Testing: (to check the stability of the build).

- To check if the build recieved from the development team is testable/stable or not.
- Performed by both developers and testers. ← cross-check the stability ↳ properly test the stability
- Build maybe stable/unstable.
- done on initial builds
- Part of basic testing
- Done every time new build is released.
- This test checks the basic functionality of the application (login, etc.)
  also if the application is running properly

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

- done by testers alone
- build is relatively stable
- part of regression testing
- planned when there is not enough time to do in-depth testing.



☐ Exploratory Testing:

Testing engineer will do exploratory testing, when there is no requirements or any documentation has been prepared, but the software application is already been built.

during this testing, we explore the different aspects of the application, understand the application completely and test it. after understanding the document, we document all possible scenarios and use it for testing

Drawbacks:

① As no requirement or design documents are not present, we might misunderstand bugs as features or features as bugs. This way if any bugs are actually present, we won't be able to identify.

② Time consuming process of testing.

☐ Adhoc Testing: It is also an informal testing technique the aim of which is to break the application. There is also no test cases or any business requirement documents.

The tester shall have the knowledge of the application even though there is no test cases

This testing is usually an unplanned activity.

☐ Monkey/Gorilla Testing: It is also an informal testing technique with the aim to break the application. Here also, no test case or business requirement document is present.

But in addition to that, the tester here does not have the knowledge of the application also.

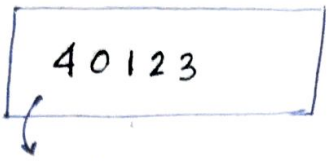
This testing is suitable for gaming applications.

Exploratory Testing	Adhoc Testing	Monkey Testing
(1) No documentation is available	(1) No documentation is available	(1) No documentation is available.
(2) No plan is prepared beforehand	(2) No plan is prepared beforehand	(2) No plan is prepared beforehand.
(3) It is an informal testing	(3) It is an informal testing	(3) It is an informal testing
(4) Testers doesn't know the application functionality	(4) Testers ~~doesn't~~ should know the application functionality	(4) Testers doesn't know the application functionality
(5) Random testing	(5) Random testing	(5) Random testing
(6) The intension is to learn/ explore the functionality of the application	(6) The intension is to break the application / find out corner defects	(6) The intension is to break the application / find out the corner defects.
(7) Any application which is new to the tester	(7) Any applications	(7) Gaming applications

☐ Positive Testing: testing with valid inputs.

→ checks whether the application behaves as expected with positive inputs.

Ex:



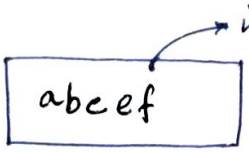
- textbox only accepts numbers
- range of numbers: 0 - 99999

testing with valid input [0 - 99999 in this case]

☐ Negative Testing: testing with invalid inputs.

→ checks whether the application behaves as expected with negative inputs.

Ex:



invalid input for the previous example.

Example: Positive v/s Negative Test cases:

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

☐ Positive Test Cases:

(1) accepts 6 characters

(2) accepts upto 20 characters length

(3) accepts any value between 6-20 characters length

(4) accepts all alphabets

☐ Negative Test Cases:

(1) should not accept less than 6 characters

(2) should not accept more than 20 characters

(3) should not accept special characters

(4) should not accept numericals.

☐ 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.

Globalization Testing	Localization Testing.
(1) ensure the application can run in any cultural or local environment	(1) testing the application for a specific geographical/cultural environment.
(2) different aspects are tested to ensure it supports every language and different attributes.	(2) check if supports the specific kind of language and usable only in specific region.
(3) tests different currency formats, phone number formats, address formats	(3) tests specific currency format, number format and address format.
(4) ex: facebook.com – global product	(4) Baidu.com – Chinese app. support only chinese standards – local product.

Dt: 20/12/2023

☐ Basics of Software Testing - Part 6:

☐ Test design Techniques: used to prepare data for testing.

While preparing test design techniques we have to keep two things in mind:

(1) Data - reducing the test data: to minimize the ~~th~~ amount of data to be tested. Testing huge datasets are lengthy and time consuming. We have to plan properly to reduce the data to be tested.

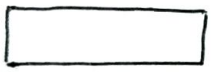
(2) Coverage: We have to take care that while using the least number of test cases, we cover all the scenarios that needs to be checked.

Types of Test design / Test data design / Test cases design techniques:

① Equivalence Class Partitioning: (ECP)

Partition or divide the data to be tested, into various classes and then test the data ~~co~~ according to the classes.

This reduces the no. of test cases and saves testing time.

Ex:  Textbox accepts values from 1-500.

Normal test data	Divide values into equivalance classes		Test data during ECP
1	-100 to 0 [-50 (invalid)]		-50
2			
3	1 - 100 [40]	[valid]	40
4	101 - 200 [150]		150
5	201 - 300 [240]		240
!	301 - 400 [370]		370
:	401 - 500 [420]		420
:	501 - 600 [550 (invalid)]		550.
.			
500			

☐ **Boundary Value Analysis (BVA):** To check the boundaries of the inputs.

Ex: Enter a age: [ ] [Range 18-35]

Boundary values to check:

(Min -1)	(Min)	(Min +1)	(Max -1)	Max	(Max +1)
17	18	19	34	35	36

☐ **Decision Table: (cause- effect table):**

→ Used when we have more number of conditions and corresponding actions

→ here we deal with combination of inputs

→ to identify the test cases, we consider the conditions and actions.

Ex: Suppose we have to transfer money to an account which is already added and approved.

Required conditions: ① account already approved (reciever's).
② OTP matches
③ Sufficient money in sender's account

Corresponding actions:

① ~~transfer~~ Show a message if sender has insufficient amount
② Transfer money if reciever's account is approved and the OTP matches.
③ Block the transaction for any suspicious activity.

Preparing the decision table:

		Test Cases				
		TC 1	TC 2	TC 3	TC 4	TC 5
Condition 1	Approved account	T	T	T	T	F
Condition 2	Matching OTP	T	T	F	F	F
Condition 3	Sufficient Amount	T	F	T	F	F
Action 1	Block suspicious transaction			Execute		
Action 2	Insufficient Amount		Execute			
Action 3	Transfer Money	Execute			Execute	Execute

☐ State Transition:

This is a technique where changes in the input conditions change the state of the application.

The tester can enter various input conditions in a sequence. For each input, the change of the state of the application shall be visible. Testers input both positive and negative values to evaluate the system behaviour.

This technique allows the tester to test the behaviour of an AUT.

Let's take the example of a login window which allows a limited no of attempts for logging in:

State	Login	Correct Password	Incorrect Password.
S1	First Attempt	S4	S2
S2	Second Attempt	S4	S3
S3	Third Attempt	S4	S5
S4	Home Page		
S5	"Account Locked. Please consult Admin"		

☐ **Error Guessing:** testing technique to find bugs in an application based on tester's prior experience.

- ⊙ No specific rules need to be followed.
- ⊙ Depends on the experience and analytical skills of the tester.

Ex: ① Submitting form without entering values
② Entering invalid values etc.

☐ **Section 3: Software Testing Life Cycle (Part 1):**

STLC is a part of SDLC only. This involves various stages of software testing. The stages of STLC are as follows:

Requirement Analysis
↓
Test Planning
↓
Test Design / Test Case Development
↓
Environment Setup
↓
Test Execution
↓
Defect / Bug reporting
↓
Test Cycle Closure

STLC:

□ **Test Planning:** Main focus on: ① what to test
② How to test
③ When to test

↳ Prepare a document clearly documenting the detailed test plan

⊙ Input of Test Planning: ① Project Plan → includes
→ How to test
→ when to test

Prepared by project/product manager

very high level document

Detailed project plan helps to setup the testing timeline

② Functional requirement Documents: ↝ includes what to test (Lead level people).

⊙ **Activities:** ① Identifying the resources
② Team formation
③ Test estimation
④ Preparation of test plan
⑤ Review of test plan
⑥ Test plan sign-off.

⊙ Responsibility: → Team Lead / Test lead (70%)
→ Test Manager (30%).

⊙ Outcome: Test Plan Document

☐ **Test Designing:** main focus is writing the test cases.

Input: Documents required:

① Project plan → How to test
→ When to test

② Functional Requirements → what to test

③ Test Plan documents

④ Design documents → to understand the UI and functionality in a better way.

⑤ Use cases → describes the requirements → generally in a diagramatic format.

Activities: ① Preparation of test scenarios → what to test

② Preparation of test cases → different areas of test scenarios that needs to be tested

③ Review of testcases → to ensure that all the possible test cases are included in the test plan

④ Traceability Matrix → Mapping document to identify which test case is for which requirement (RTM → requirement traceability matrix),

⑤ Test cases sign-off: → sign-off is important to proceed to the next phase.

Responsibility: Test lead / Team lead → 30%
Test Engineers → 70%

Outcome: ① Test cases document
② Traceability Matrix.

□ Test Execution : testo tests are run during this phase.

□ Input : ① Functional Requirement specifications
② Test plan
③ Test cases
④ Traceability Matrix
⑤ Builds from the development team.

□ Activities : ① Setup testing environment
② Executing test cases → based on priority
③ Preparation of Test report / Test log
④ Identifying the bugs / defects.

□ Responsibility : Test Lead / Team Lead (10%.)
Test Engineers (90%.)

□ Outcomes : Status Report
Test Reports / Test Logs

□ Defect Reporting and Tracking :

⊙ Inputs : ① Test cases ② Test reports / test logs

⊙ Activities : ① Preparation of defect report
② Reporting defect to developers.

⊙ Responsibility : Test lead / Test team lead → 10%.
Test engineers → 90%.

⊙ Outcome : Defect Report

□ Test closure / sign-off :

⊙ **Input :** ① Test reports ② Defect reports

⊙ **Activity :** ① Analyze / review test reports
② Analyze / review bug reports
③ Evaluating Exit criteria

⊙ **Responsibility :** Test lead / Test manager → 70%
Test engineers → 30%.

⊙ **Outcome :** Test Summary Reports.

□ Contents of Test Plan :

A test plan is a document which describes the test scope, test strategy, objectives, schedule, deliverables and resources required to perform the testing.

□ Contents :

- Overview
- Test scope
  - → Inclusions
  - → testing environment
  - → exclusions
- Test strategy
- Defect reporting procedure
- Roles / responsibilities
- Test schedule
- Test deliverables
- Pricing
- Entry / exit criteria
- Suspension / resumption criteria
- Tools
- Risk and mitigations
- Approvals

☐ Use case, test scenario and test case:

- pictorial view or data flow diagram (generally).
- describes the requirement
- contains three items: → Actor → user, who will interact with the process (single/group of people).
  - action → what shall be done to reach the final outcome
  - goal/outcome → what the successful user outcome.

Prepared by high level management

Test scenario: → what to test, a possible area to be tested.

- derived based upon the ~~xxxxxxx~~ use cases (mentioned in FRS)
- Prepared by testers.

Test case → generally describes how to test.

↳ step by step actions performed to validate the functionality of AUT.

↳ test cases contains test steps, expected result and actual results

↳ prepared by testers.

Use cases V/s Test Case:

Use cases	Test Case
• describes functional requirements	• describes test steps/procedures
• prepared by Business analysts	• Prepared by test engineers.

Test Scenario V/s Test Case:

Test Scenario	Test Case
• describes "what to test"	describes "how to test"

Example:

Test Scenario: checking the functionality of login button.

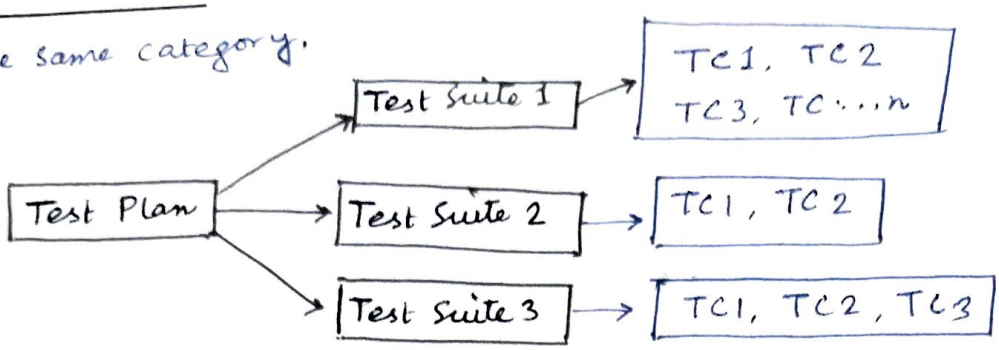
TC1: click without ID and pass.

TC2: Click with entering only ID

TC3: " " " only pass

TC4: " " " wrong ID and Password.

☐ **Test Suite:** Test suite is a group of test cases that fall under the same category.



☐ **Test Case:** Test case is a set of instructions that is executed to validate a particular feature or functionality of a software application.

It is also a document (primarily a document, after review of the test cases is done, it is turned into test cases using testing tools).

Contents of Test Case:

1) Test case ID
2) Test Case Title → short title
3) Description → detailed description of the test case
4) Pre-condition → conditions to be satisfied before the test case is satisfied.
5) Priority → (P0, P1, P2, ..., PN) order of execution
6) Requirement ID → for which requirement test is being done
7) Steps/Actions
8) Expected result
9) Actual result
10) Test data

☐ Requirement Traceability Matrix:

Document describing the mapping of test cases with their corresponding requirement IDs.

One requirement ID can have multiple test cases, so, RTM is important, so that no test cases are left behind and no functionality is missed during ~~the~~ testing of software.

Contents:

1) Requirement ID
2) Requirement description
3) Test case ID
4) Test case status.

☐ Test Environment: Test environment is a platform specially build for test case execution on the software product

It is created by integrating the required hardware and software along with proper network configurations and simulates production/real-time environment. Also known as test bed.

☐ Test Execution:

During the execution phase, the test team will carry out the testing based on the test plan and test cases.

Entry Criteria: Test plan, Test cases, test data.

Activities:

(1) Execution of test cases based on test plan
(2) Noting the status of tests completed
(3) Documentation of test results and defect logs is done.
(4) All the failed and blocked test cases are assigned bug IDs.
(5) Retesting once the defects are fixed
(6) Defects are tracked till closure.

Deliverables: Provides defect and test case execution report with complete results.

Guidelines for execution:

① Build being deployed to the QA environment is the most important part of test execution cycle

② Execution is done in QA environment.

③ " happens in multiple cycles.

④ Test execution phase consists of → ① executing the test cases
② test scripts (if automation)

***

☐ **Defect / Bugs:** Any mismatched functionality found between the expected outcome and the actual outcome is known as defects / bugs / issues

During execution, test engineers are reporting mismatches as defects to developer thorough templates or tools

Defect reporting tools:

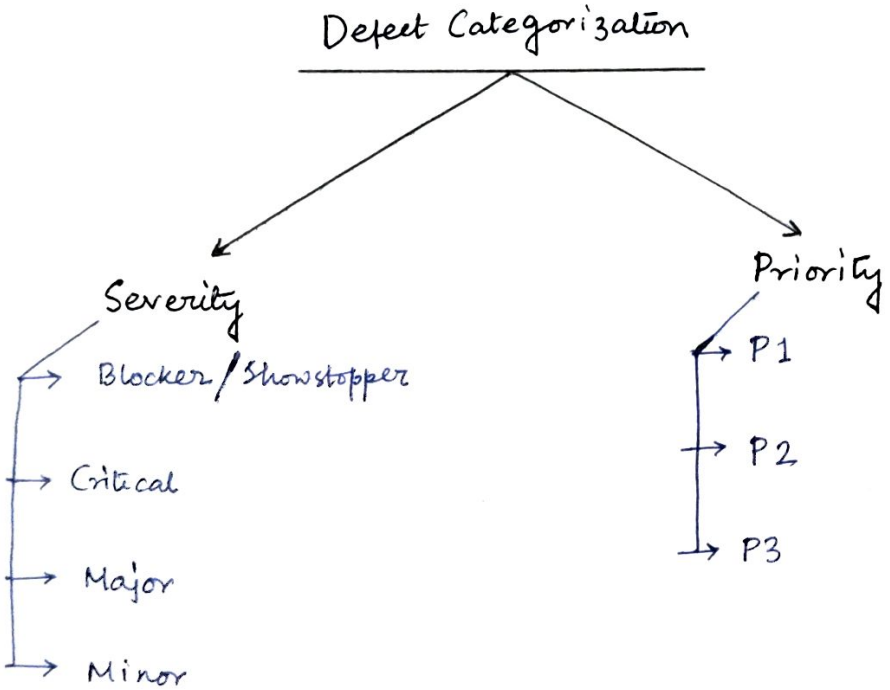
① Clear quest } Bug tracking only
② DevTrack }
③ Jira } Test management tools.
④ Quality Center }
⑤ Bug Jilla etc. } bug tracking only

Defect Report Contents:

- **Defect_ID:** unique identification number for defects
- **Description:** Detailed description of the defects along with the module it was found in.
- **Version:** version of the application it was found in.
- **Steps:** Detailed steps along with screenshots with which developers can reproduce the defects
- **Date Raised:** Date when the defect was raised

- Reference: reference to the requirement, design, architecture documents or screenshots of the error.
- Detected by: Name/ID of the tester who raised the defect
- Status: Status of the defect
- Fixed by: Name/ID of the developed who fixed the defect
- Date closed: Date when the defect is closed.
- Severity: the impact of the defect on the application
- Priority: Urgency of fixing the defect.

Severity Priority can be High/Medium/Low based on the impact urgency at which the defect shall be fixed respectively.



☐ Defect Severity: Severity describes the seriousness of the defect and how much impact it will have on the business workflow.

① Blocker / show stopper: Indicates that testing cannot be proceed further while the defect is present.

ex: application crashed, login not working etc.

② Critical: We can install the application and start the testing but basic/main functionalities are not working. Thus cannot proceed further.

ex: ① Fund transfer not working in internet banking
② Ordering product in e-commerce site not working

③ Major: The feature / application is functional, but is causing some undesirable behaviours

ex: ① After sending email, it does not move to 'Sent Items'.
② After booking a cab, there is no confirmation

④ Minor: It won't cause any major break-down of the system

ex: Look and feel issues, spelling, alignments etc.

☐ **Defect Priority:** It describes the importance of the defect and the order in which the defect shall be fixed.

P0 (High): must be resolved immediately as it affects the system severely and can't be used until fixed.

P1 (Medium): It can wait until a new version / build is created

P2 (Low): Developer can fix it in later releases.
↳ can have more than one versions in one release.

☐ **Combination of Severity - Priority of defects:**

Severity	Priority	Example
High	High	⊙ Login option is taking to blank page.
High	Low	⊙ About Us link is going to a blank page.
Low	High	⊙ After logging in, Home Page opens but there is spelling mistake in Home Page.
Low	Low	⊙ User opened contact page. There is a spelling mistake in the email ID. ⊙ Spelling mistake in a page not frequently navigated by the users.
High	Low	⊙ Application crashing in some corner case ⊙ Web page not found in link (page which is not frequently visited).
High	High	⊙ Issue with login functionality
Low	High	⊙ Slight changes in logo color/ spell mistake in company name.
Low	Low	⊙ Cosmetic/spelling issues which is within a paragraph or in the page.

☐ Defect Resolution: After receiving the defect report from the testing team, development team conducts a meeting to review the defects. They send the resolution to the testing team for further communication

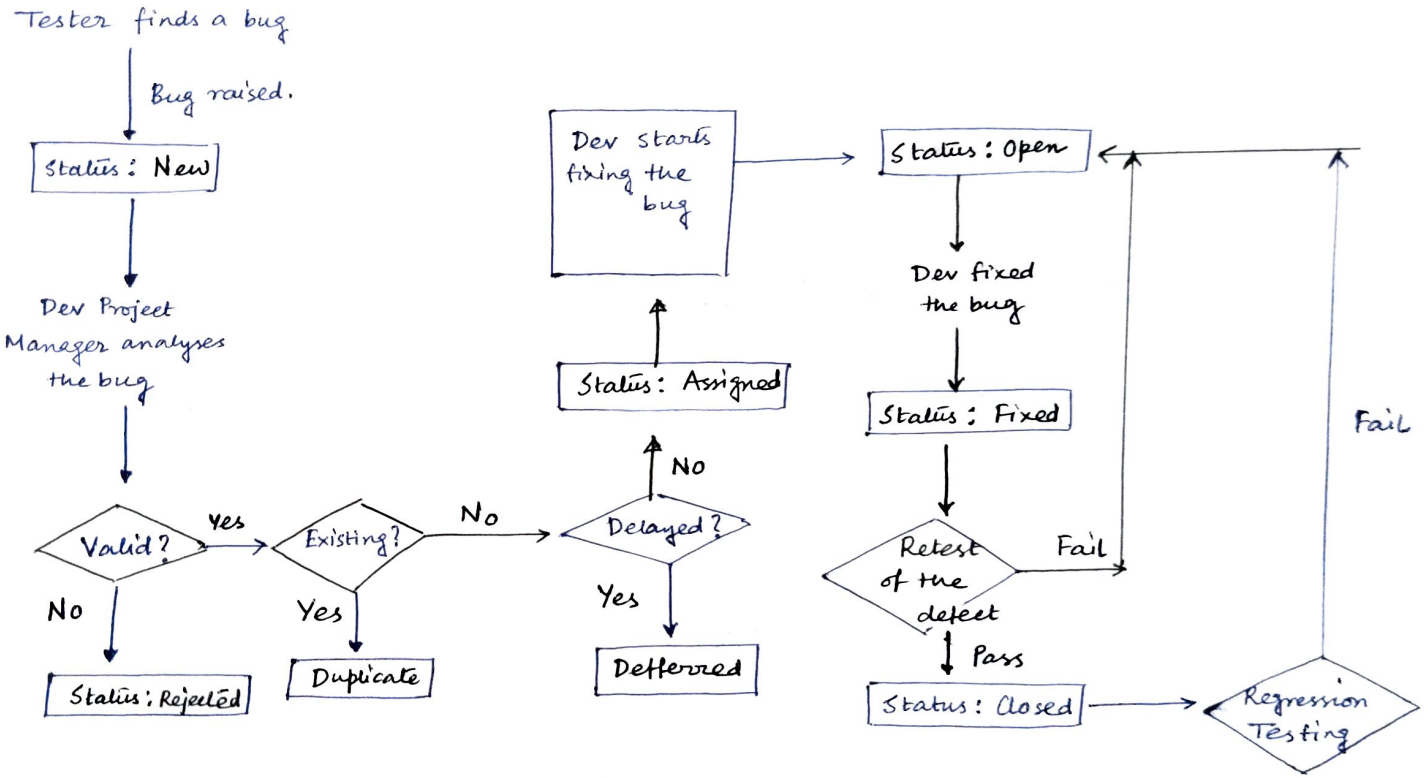
Types: → accept
→ reject
→ duplicate
→ Enhancement (new feature which is in development phase)
→ Need more information
→ Not reproducible (same defect not coming in the developer environment)
→ Fixed
→ As designed.

☐ Test Cycle Closure:

→ Evaluate cycle completion criteria based on Time, test coverage, cost, software, critical business objectives, Quality.
→ Prepare test metrics based on the above parameters
→ ~~Declare~~ Document the learning outcome of the project
→ Prepare test summary reports.
→ 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.

Bug Life Cycle :



☐ 7 Principles of Software Testing:

(1) Start software testing at early stages. (From requirement)

(2) Test the software in order to find the defects

(3) Highly impossible to provide a bug free software to the customer.

(4) Should not do Exhaustive testing. (Shall not use the same type of data for testing every time).

(5) Testing is context based. (Choose the tests to be performed depending on the application).

(6) Follow Pesticide Paradox. (If we conduct or execute the same test cases for each version/build, no defects will be found out. Keep updating the test cases in every cycle/release.

(7) Follow defect clustering. (Some modules contains most of the defects. We should prioritize the risky modules based on experience. 80% of the defects are found in 20% of the modules).

☐ Test Metrics:

1. No. of requirements
2. Avg. no of test cases written per requirement
3. Total no of " " " " "
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 not executed
9. Total no of defects identified.
10. Critical Defects count
11. High defects count
12. Medium defects count
13. Low defects count.
14. Customer defects
15. No. of defects found in UAT.

→ **% of test cases executed:**

(No. of TC executed / total no of TCs written) × 100

→ **% of test cases not executed:**

(No. of TC not executed / total no of TCs written) × 100.

→ **% of test cases passed:**

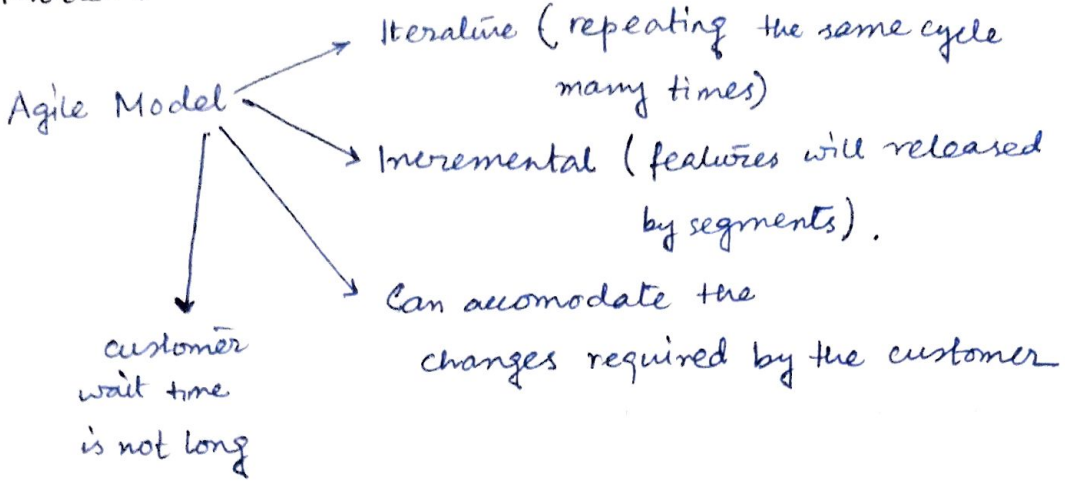
(No. of TC passed / No of TC executed) × 100

→ **% of test cases failed:**

(No. of TC failed / No of TC executed) × 100

→ **% of test cases blocked:** (No. of TC blocked / No of TC executed) × 100

Agile Model:



Benefits: (1) Communication b/w the customer, BA, Developer, testers are good.

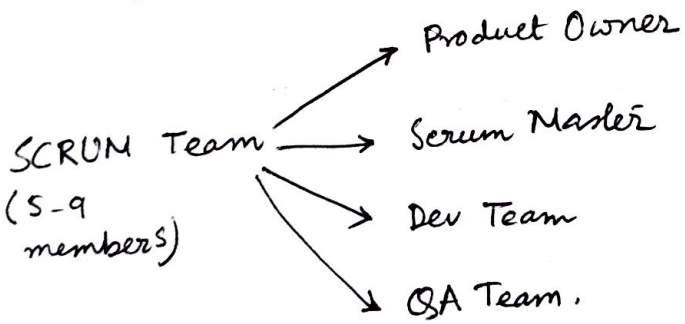
(2) Delivery is faster as small segments are deployed

(3) requirement changes allowed

~~(4) require~~

Disadvantages: (1) less focus on design and documentation as software delivery is faster

☐ SCRUM: defines how to follow the agile principles
↳ it is the framework
↳ Framework thorough which we build software product by following agile principles.



□ Product Owner: ① Define feature of the product

② Prioritize features based on their market value and adjust them as required

③ Accept or reject work result.

④ POC of the customer

□ Scrum Master: ① Main role: facilitating and driving the agile process.

② Others follow the agile process or not is checked by scrum master

□ Developer & QA: Develop and test the app.

□ Scrum Technology:

① User story: a feature/module in software } Product Owner

② Epic: collection of user stories

③ Product backlog: list of user stories prepared by PO (document)

④ Sprint: period of time to complete user stories. (Product owner + team)
or Iteration: generally 2-4 weeks.

⑤ Sprint Planning meeting: define what can be delivered in the sprint and duration.

⑥ Sprint backlog: user stories belonging to the specific sprint.

⑦ Scrum/Standup meeting: meeting → everyday 15 mins
→ Status tracking

⑧ Sprint retrospective meeting: meeting after one sprint is completed.

⑨ Story point: rough estimation of user stories, given by Dev & QA in the form of Fibonacci series.

□ Burndown chart: tracking how much work is remaining in the sprint. Prepared after daily scrum meeting by Scrum master.

□ JIRA Tool: Agile Management Tool.

↳ + JS → Project management.
plugin

Tools
- Project Management Tools
- Bug tracking tools and reporting
- Agile Tools (automation of agile process).

Jira, VersionOne, Teamcity etc.

□ JIRA Work flow:

① Agile Scrum Activities:

1. Creating the Project
2. Adding users / people into JIRA
3. Create Backlog (Product).
   - → Create version
   - → Create Epic.
   - → Create Stories
   - → Add story points.
4. Create sprint
5. Add stories to sprint
6. Starting Sprint.
7. Add sub task to story
8. Sprint life cycle.

② Test Management Activities: (Zepyre):

1. Test Cases
2. Test Cycle
3. Update test status
4. Report bugs
5. Reports