**Software:** a collection of computer program that helps us to perform a task

**types of software:**

**1.system software:** Operating system, servers, Utilities, device driver etc.

**2. programming software:** compilers, debuggers, interpreters

**3. application software:** web application, mobile apps, desktop apps

**software testing:** part of software developing process where it is an activity to detect and identify the defects in the software. The object of testing to release quality product(software) to the client

**software quality:** follow these parameter

* + Bug-free
  + Delivered on time
  + According to budget
  + Meets requirements/expectations
  + Maintainable

**project vs product**

If software is developed for specific customer based on requirements then it is called project.

If software is developed for multiple customers based on market requirements then it is called product.

**Error:** Human mistake // Identify by the developers

**Bug/Defect:** something is not working in application according to requirement/ expectation // Identify by the Tester

**Failure:** It is identifying by the end user while using software is called failure. // It happens production environments

**Why software has bugs:**

* + Miscommunication/no communication between developers, documentation of software and tester
  + software complexity
  + programming errors
  + changing requirements
  + lack of skilled tester

**SDLC:** software development life cycle is a process used by software industry to design, develop and test software.

**SDLC phase:**

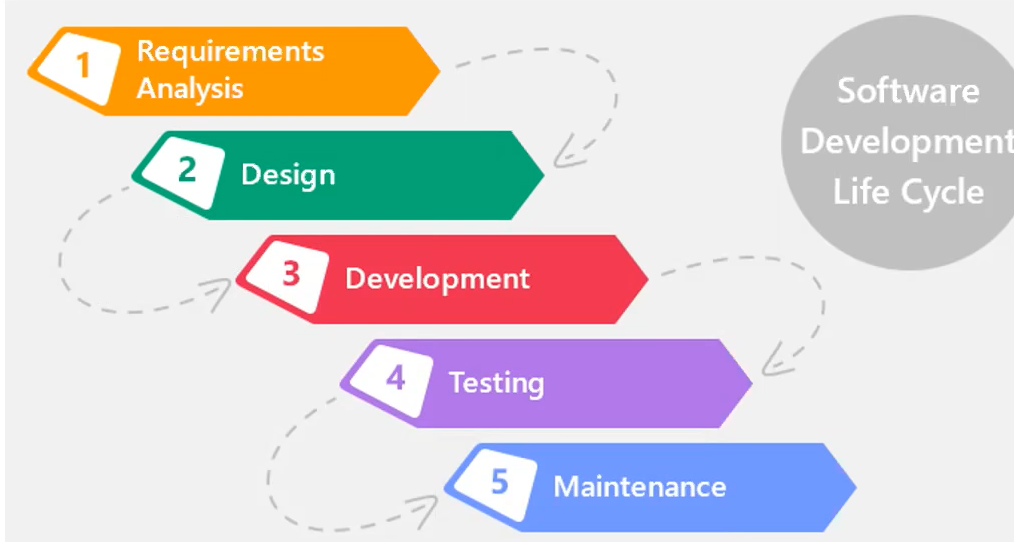
1. Requirement Analysis: collect, understand the requirements.

2. Design: Designer design the software (UI, navigation part)

3. Development: Coding part

4. Testing: Test the products

5. Maintenance: maintain the production



**Model of SDLC:**

**Waterfall Model:**

**1. Requirement Analysis:** Documentation is very important (create SRS (Software Requirement Specification)), involve project manager, client

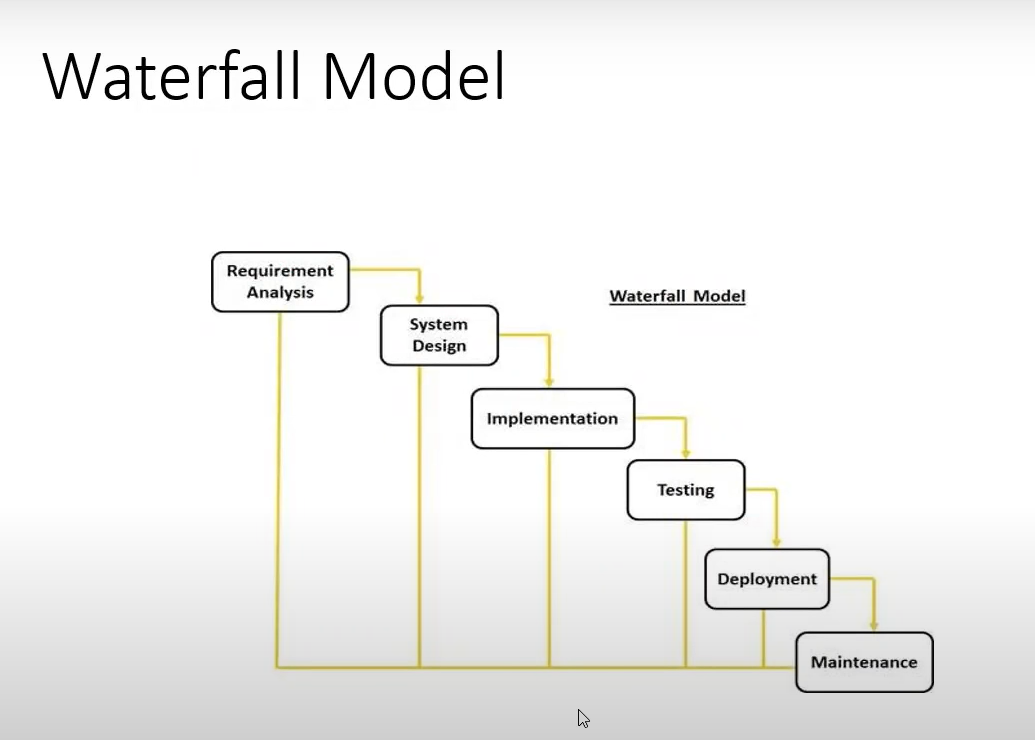
**2. System Design:** Design the software (Design small piece of software and finally add all parts)

**3. Implementation:** Coding part

**4. Testing:** Test the products

**5. Deployment:** deploy(install) the software in customer environment

**6. Maintenance:** maintain the production



**Advantage:**

1. Quality of product will be good

2. Since Requirement change are not allowed, chances of finding bugs will be less.

3. Initial investment is less, testers are hired in the last stage

4. Preferred for the small projects, requirements are freeze

**Disadvantage:**

1. Requirement change are not allowed

2. If there is defect is requirement that will be continued in every phase

3. Total investment is more, time taking for rework on defect where need to every step back for solving defect

4. Tester will start only after coding

**input: SRS -> Design doc -> program -> software testcase -> installation doc to customer -> maintain**

**Spiral Model:**

**Important points:**

-> Follow products base company

-> New requirements(model) depend on previous model(requirements)

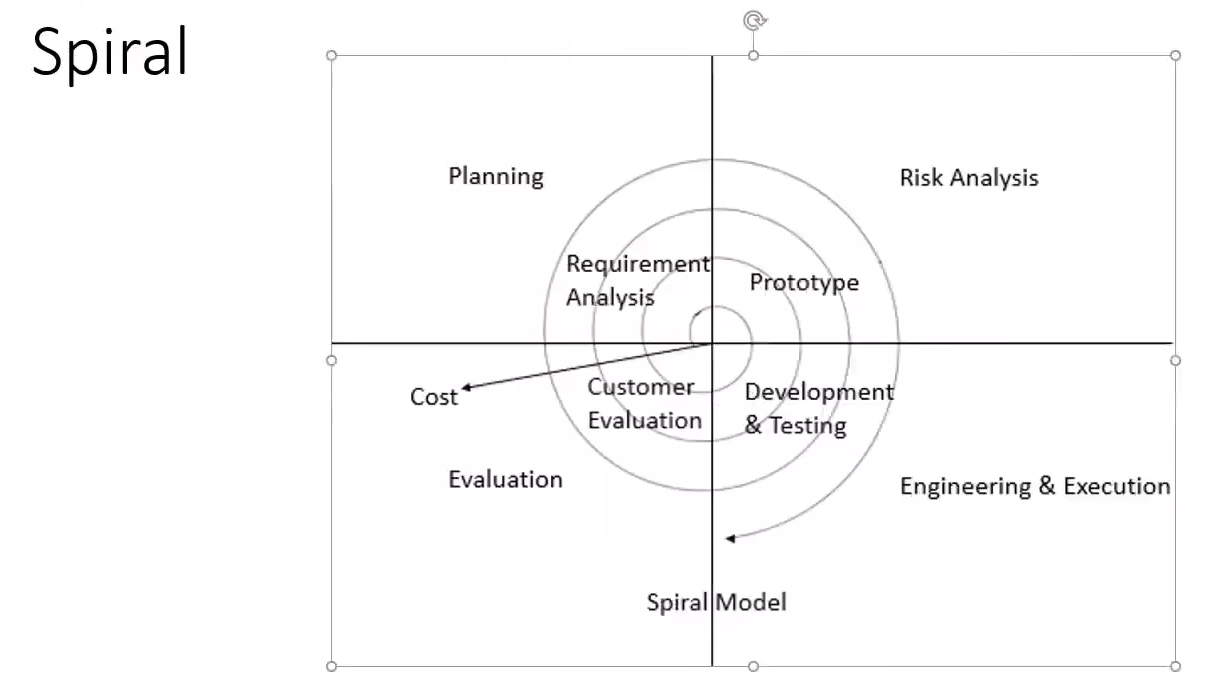
-> Every cycle follows the waterfall model

-> Software will be released multiple versions (version control model)

-> Follow Iterative model

-> Overcome drawbacks of waterfall model

-> Every cycle new software will be released to customer



**Advantage:**

1. Testing is done every cycle

2. Customer get every module of software to use

3. Requirements changes are allowed after every cycle

**Disadvantage:**

1. Requirements changes are not allowed between the cycle

2. No testing in requirement and design phase

**Prototype Model:**

-> It comes between waterfall & spiral models

-> Blue print of the software

-> Get initial requirement from customer

->Develop full prototype of software

->Showing the customer

-> If prototype is accepted by customer, then Design, coding, testing, deploying, maintaining

**V / VV (Verification, Validation) Model:**

-> every phase we will do testing

**1st phase**: BRS (Business Requirement Specification) / (Client Requirement Specification)/URS (User Requirement Specification) documents cannot understand for developer and used for test (User acceptance testing) and created by Business people

**2nd phase:** SRS is understandable for developer and tester (Technical format) & created by PM (project manager) & used for test (System testing

**3rd phase:** Base on SRS designer create HLD (high level design) & LLD (low level design) and created by designer & used for test (Integration testing)

**software module 2 types**

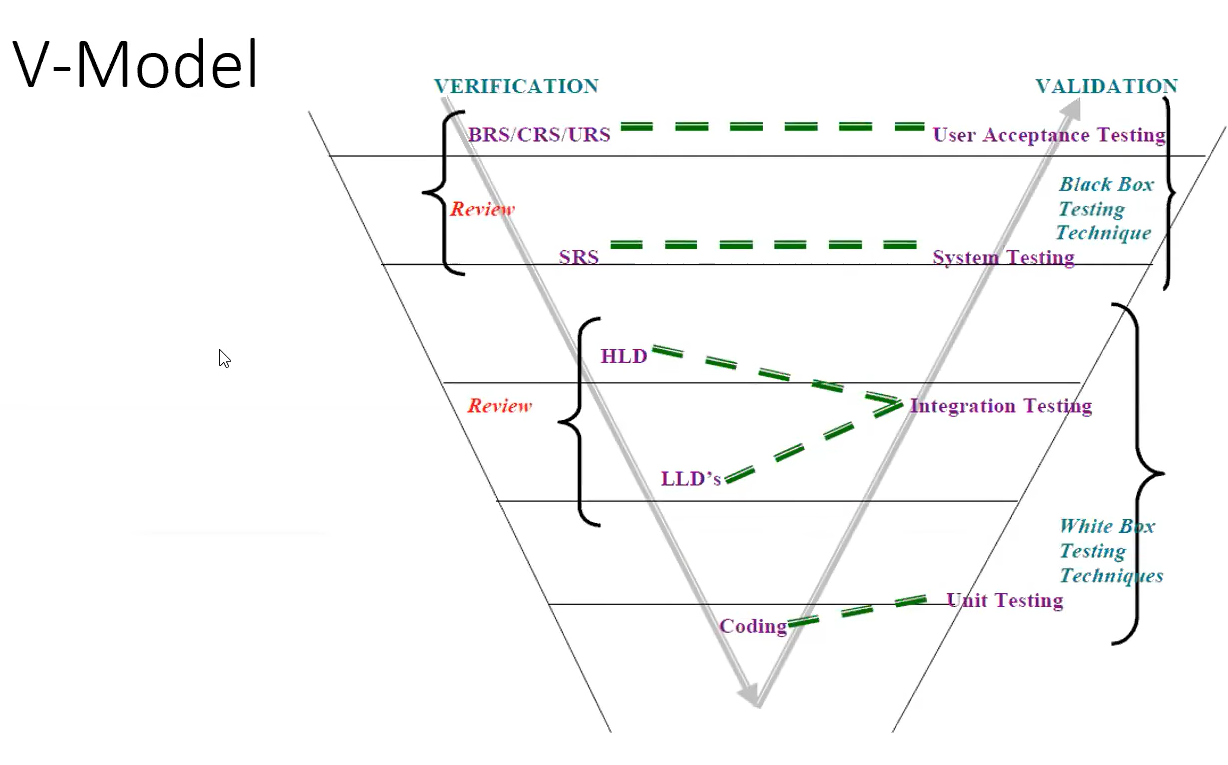
1. Main module

2. Sub module

**HLD doc->** have main module

**LLD doc->** have sub module

**4th phase:** Coding will start & used for test (Unit testing)

****

**Verification:**

1. check to start building the right products

2. Focus on documents

3. Use static testing technique

4. Done before software ready

**Validation:**

1. Check the building products are right or wrong

2. Focus on software

3. Use dynamic testing technique

4. It is actual testing

5. Done after software ready

**Advantage:**

1. Testing is involved in each and every phase

**Disadvantage:**

1. Documentation is more

2. Initial investment is more

**-> Every part of software is called module**

**Ex-> Gmail: login, inbox, sent, compose etc.**

**Static testing:** Testing the project related documents. by doing (Review, walkthrough, inception)

**Dynamic testing:** Testing the actual software by doing (unit, integration, system, UAT testing)

->developers do unit & integration testing (which call white box testing technique)

-> system testing is done by the tester (black box testing technique)

**->** user acceptance testing is done by tester and customer (black box testing technique)

**software module 2 types**

1. Main module

2. Sub module

**White box testing:** Test the internallogic of program, test the code directly, developer can do that

**Black box testing: Verify the functionality of the application whether** it is working according to customer requirement or not, do not need to know programming logic, know the functionality of application, how to flow, UI functionality, tester will do this.

**Static testing technique:**

**Review:** Conducts on documents to ensure correctness and completeness. Review can done by anyone anytime. Multiple reviews are given below:

1. Requirement review: got the requirements from the customer and review them
2. Design review: Document form of diagram and pictures
3. Code review: Code correctly returns output, logically correct, clean code
4. Test plan review: defect review, test case review, test plan review

**Walkthrough**:

1. It is informal review. Without any plan, any meeting arranging by software people
2. Author read the documents or code and discuss with peers (team member)
3. It’s not pre planned and can be done whenever required
4. Does not need time to meeting

**Inspection:**

1. Most formal review type (sending mail to team member, arrange meeting)
2. Have a proper schedule which will be intimated via email to the connect developer/tester
3. At least 3-8 people will sit in meeting 1. Reader (read the whole documents) 2. Writer (write the issue, clarification) 3. Moderator (organizer of meeting)

**Dynamic Testing Technique:** Coding part is done

1. Unit: testing the code by developers
2. Integration: integrate multiple modules, components and check the data flow between the components
3. System: test the software is working according to customer requirements or not
4. UAT (User acceptance testing): Customer test the software with tester and setup environment to the customer access

**QA (Quality Assurance):** It is process related, define the process, following the process, it involves whole SDLC phases, focus on building in quality, preventing defects, process oriented, example-project manager

**QC (Quality Control):** It is the actual testing of the software, it involves just testing phase of SDLC, focus on testing for quality, related to people(tester), detecting defects, product oriented, example- tester (Manual Tester)

**QE (Quality Engineering):** Write the code for testing the software (Automation Engineer)]

**Level of software testing:**

1. **Unit Testing:**

* A unit testing is a single components or module of a software
* Conducts on a single program or single module (Only testing through coding)
* It is white box testing technique
* Conducted by the developers
* A single module UI part testing is component testing not unit testing
* **Unit testing technique**:
  + Control structure testing
    - Conditional coverage: Verify the condition test
    - Loops coverage: repeat multiple times test
  + Mutation testing: Repeat the test different type of inputs. Example

If user = ‘A’ && pass= ‘123’

Allow login

else

Not allow login

* + Basic path testing: Every line of program should be executed for testing

1. **Integration Testing:**

* Performed between 2 or more modules
* Focus on checking data communication between 2 or more modules
* It is white box testing
* This test done by tester from UI level and done by developer from coding part
* **Types of integration testing:**
  + **Incremental integration testing:** Incrementally adding the module and testing the data flow between the modules. **3 approaches:**
    - **Top Down:** Incrementally adding the module and testing the data flow between the modules and ensure the module added is the child of previous module. **Example:** [Email module functionality] Compose message (parent module)->sent message (child module)->delete message (child).
    - **Bottom Up:** Incrementally adding the module and testing the data flow between the modules and ensure the module added is the parent of previous module. **Example:** sent message (child module)-> Compose message (parent module)
    - **Hybrid/Sandwich:** combination of top-down and bottom-up approach
  + **Non- Incremental integration testing:** Integrate all modules in single short and test data flow between modules. **Drawback:**
    - Miss data flow between some of modules
    - If find ant defect can not understand the root cause of defects

1. **System Testing**:

* Testing over all functionality of the application with respective client requirements
* It is black box testing technique
* Conducted by testing team
* We should know the customer requirements.
* **4 types of system testing:**

1. **Graphical User interface testing (GUI):** It is a process of testing the user interface of an application. It includes all the elements such as menus, checkbox, buttons, colors, fonts, size, content and image

**GUI testing checklist:**

1. Testing the size, positions, width, height of the elements
2. Testing the error message that are getting display
3. Testing the different sections of the screen
4. Testing the fonts whether it is readable or not
5. Testing of the screen in different resolutions with the help of zooming in and zooming out
6. Testing the alignments of the texts and other elements like buttons, icons, etc. are in proper place or not
7. Testing the colors of the fonts
8. Testing whether the image has good clarity or not
9. Testing the alignments of the images
10. Testing of the spelling
11. Testing whether the interface is attractive or note
12. Testing of the scrollbars according to the size of page if any
13. Testing of the disable fields if any
14. Testing of the size of the image (image have less size and good quality)
15. Testing of the heading whether it is properly aligned or not
16. Testing of the color of the hyperlink
17. Testing UI elements such as menus, checkbox, dropdown, radio, text area, textbox, link, buttons etc.
18. **Functional testing:**

* Functionality is nothing but behavior of the application
* It talks about how the feature should work
* **Types of functional testing:**
  + **Object properties testing:** Every element of web page is called object and every object have properties /attributes. Properties means for example radio button have 2 properties. Enable & disable and we perform one property at a time. Ex- enable, visible, disable, focus etc.
  + **Database testing/Backend testing:**

Focus on DML (Data Manipulation Language) testing

From UI side CRUD operation is working fine or not through database table

When we see some data from table in database, we will do some query in database which show white box testing

Database testing include black box and white box testing and for reason it is called gray box testing

Need to know SQL to perform database testing

Verify some important features are functions, procedures, triggers, index, view, insert select, delete, relation (normalization), table and column level validations

* **Error handling testing:**
* Tester verify the error messages while performing incorrect action on the application
* Error message should be readable and user understandable language
* Ex- ‘Incorrect data’ is too hard to understand than ‘invalid user’, when try to login incorrect data
* **Calculation testing:**
* Focus on calculation or not
* Check valid and invalid data for calculation

* **Link existence and Link execution:**
  + - Where exactly the link is placed = Link existence
    - Link are navigating to proper page or not = Link execution
    - Internal link = stay in same page but jump into another section
    - External link = navigate to another page
    - Brokens link = can not navigate to other pages
* **Cookies and Sessions:**
* Cookies are temporary file created by browser while browsing the pages through internet
* Browser save the cookies or not for new application checking in cookies testing
* Session will be created in the server side. In banking application time slot is created in server side after login and if we do nothing after login, after few minutes it need to login again because of session is expired.

1. **Non-functional testing:**

* Once functional testing is stable, we perform non- functional
* Focus on performance, load it can take and security
* **Types of non- functional testing**:
  + **Performance testing:** Speed of an application, ex- web application. **3 Types:**
* **Load testing:** Increasing the load (people) slowly on the application and then check the speed of the application.
* **Stress testing:** Suddenly increase/decrease the load and then check the speed of the application
* **Volume testing:** Check how much data is able to handle by the application.
  + **Security testing:** How secure the application and focus on
    - **Authentication:** verify the user is valid or not
    - **Authorization/Access control:** Verify the permission of the valid user
  + **Recovery testing:** Check the system change to abnormal state to normal state. Ex- compose a mail but internet connection is cut off, then the mails save to draft section of Gmail functionality and that is recovery system.
  + **Compatibility testing:**
    - **Forward** **compatibility**: If software able to update from old version to new version then it called forward compatibility
    - **Backward** **compatibility**: A new version software able to install previous version environments is called backward compatibility
    - **Hardware compatibility:** software should able to install multiple hardware environments (many types of CPU like 4gb ram base. 16gb ram base). This is also called Configuration testing.
  + **Configuration testing:** software should able to install multiple hardware environments (many types of CPU like 4gb ram base. 16gb ram base).
  + **Installation testing:** 
    - Check screen are clear to understand or not
    - After uninstall software check all data are clear from environment
  + **Garbage testing:** Check extra functionality is available in the application that is not customer requirements then remove it because of consider as a bug

1. **Usability testing:**
   * During this testing validates application provided context sensitive help and from the help button user can easily able to understand and operate the application. If help button does not exist to the software, then it must be given the manual documents to the end user to give some instructions about application
2. **UAT (User acceptance testing):** Customer test the software with tester and setup environment to the customer access

**Some testing:**

**Regression Testing:**

* Testing conducts on modified build to make sure there will not be impact on existing functionality because of changing like adding/ deleting/ modifying features
* **Regression testing types:**
  + **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 & Developer.
  + **Full regression**:
    - Testing the main features and remaining part of application
    - Dev has done change in many modules, instead of identifying impacted modules, we perform all round of full regression
* **Re-testing:**
  + Whenever the developer fixed the bug, tester will test the bug fix (developer fixed bug) 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 is released, test team test the bug fix (defect Id 1.0.1), if they will find the bug, posted it
    - Developers are fix that bug and then next build 1.1 is released, now tester is testing the defects (defect ID 1.0.1) in this build is retesting
* **Re-testing Vs Regression testing**
  + If the tester found the bug on purchase module and posted. Once the bug is fixed. Tester need to do retesting to verify whether the bug related to purchase is fixed or not and also tester need to do regression testing to test the finance module which depends on the purchase module

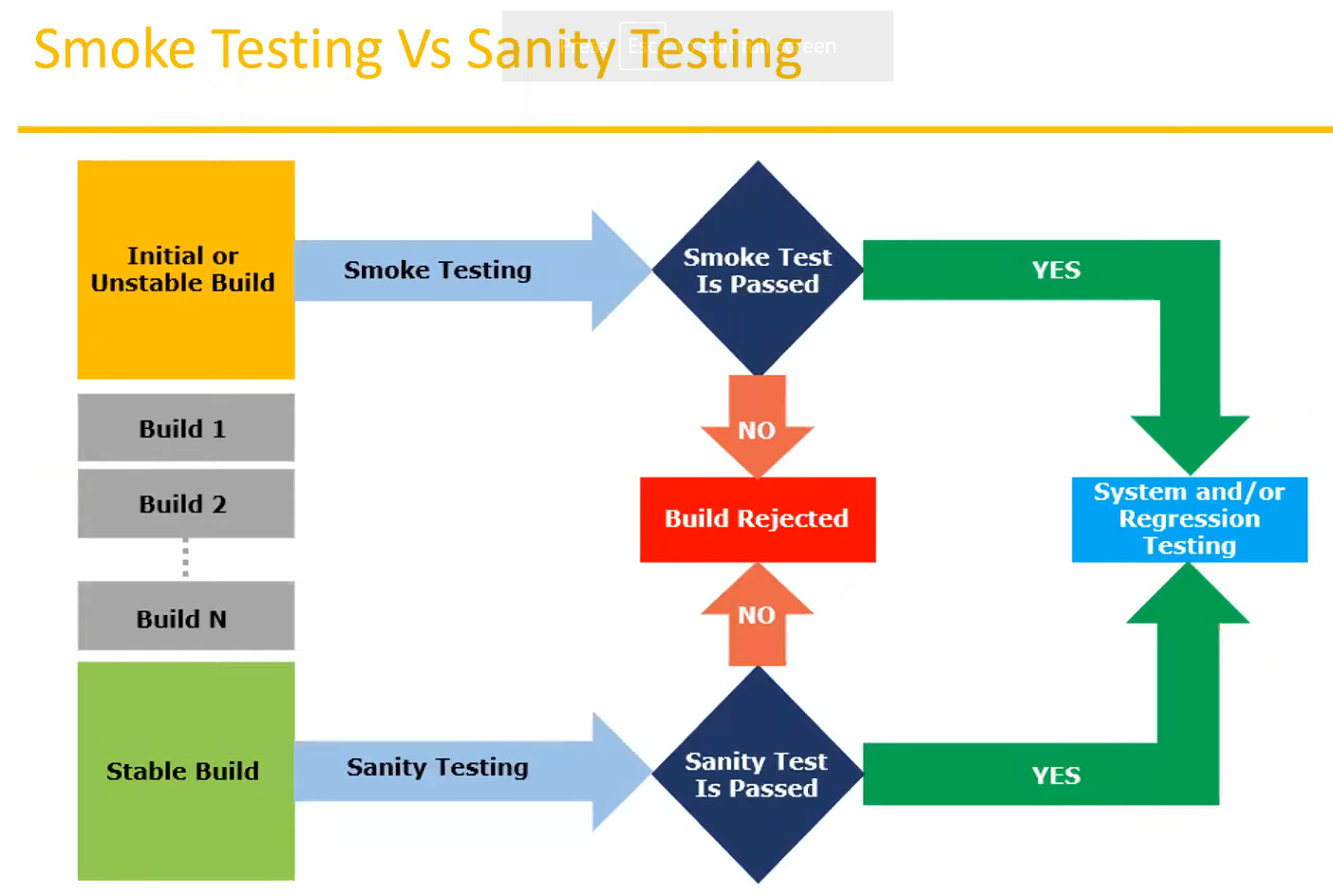
Admin

Purchase

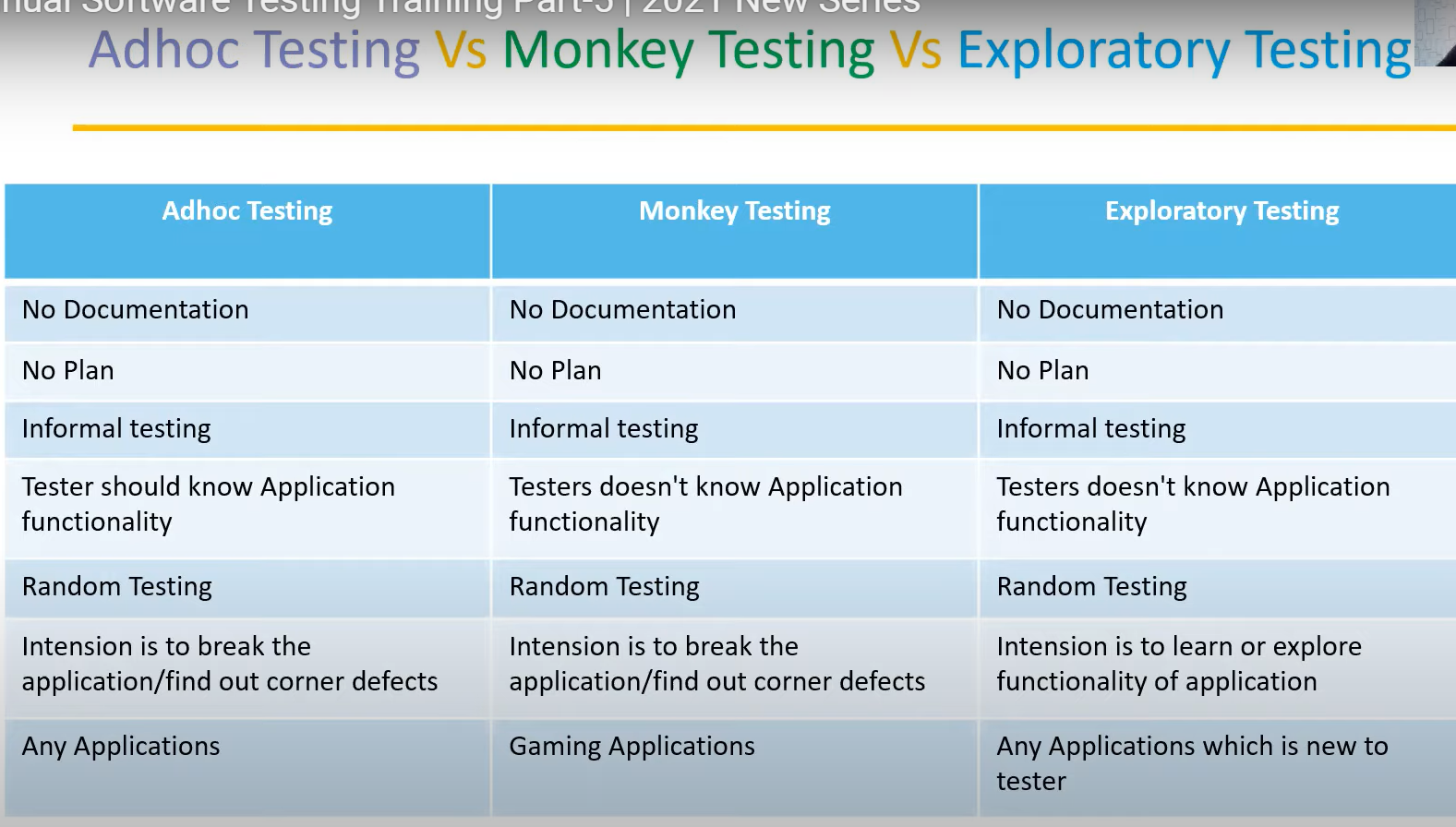
Finance

* **Smoke testing:** 
  + It is done to make sure the build we received from the developer is testable or not
  + Smoke testing is performed both dev and tester
  + In smoke testing, build may be sable or unstable
  + It is done on initial build (maybe unstable)
  + It is part of basic testing
  + It is done every time there is a new build release
  + Ex: full installation setup to the environment

* **Sanity testing:**
  + After smoke testing done and conformation for stable build, then check the main functionality of the application without going deeper
  + It is performed by tester along
  + It is done on stable build
  + It is part of regression testing
  + Build is relatively stable
  + It is plan when there is no enough time to do in depth testing



* **Exploratory testing:**
  + Explore the application without having any documents of application, understanding completely and then test it is called Exploratory testing
  + Understanding the application, identifying all the possible scenario, document it then using it for testing
  + Test engineer(experienced) will do this testing when there is no requirements
  + **Drawback:**
    - Time consuming
    - Existing bug never finding
    - Misunderstanding any features as bug because of no requirements
* **Ad-hoc testing:** 
  + Testing the application randomly without any test cases and requirements
  + It is Informal testing and tester always break the application
  + This is usually an unplanned activity
  + Tester should knowledge about application and previous experience
* **Monkey testing:**
  + Testing the application randomly without any test cases and requirements
  + It is Informal testing and tester always break the application
  + Suitable for gaming applications
  + Tester do not have knowledge about application and previous experience.

****

* **Positive testing:**
* Testing the application with valid inputs is called as positive testing
* It checks whether an application behaves as expected with positive inputs
* For example:

99999

Enter only number

* There is a text box in an application accepts only number. Enter input value from 0 to 9999 and check the system is accepting the value and others values should not acceptable
* **Negative testing:**
* Testing the application with invalid inputs is called as positive testing
* It checks whether an application behaves as expected with negative inputs
* For example:

abcdef

Enter only number

There is a text box in an application accepts only lowercase character. Enter input value from a to z and A to Z & check the system is accepting the value or not. Here an error message throw for invalid value

* **End-to-End testing:**
* Testing the overall functionality of the system including the data integration among all the modules is called End-to-End testing
* **For example:**

Delete ‘A’

Logout

Add ‘A’

Login

Edit ‘A’

* **Localization and Globalization testing:**
  + **Globalization testing:**
    - It tests the different currency formats, mobile number formats, address format are supported by the application
    - Ex: facebook.com support many languages and access by many countries people
    - Perform to ensure the system or application can run any local and cultural environment
  + **Localization testing:**
    - Perform to ensure the system or application can run specific local and cultural environment
    - Ex: Baidu.com support only Chinese languages and access by Chinese people
    - It tests the specific currency formats, mobile number formats, address format are supported by the application

**Test design technique:**

->To prepare data for testing and also reduce the data that all coverage the test scenario while testing.

**Two main parts:**

* **Data**: it is user for testing
* **Coverage**: the amount of data for testing that coverage all the scenario of testing. Ex: valid username and valid password, valid username and invalid password etc.

**5 types of test design technique/ test case design technique:**

* **Equivalence class partitioning:**
  + Partition the data into various classes and we select the data according to the class then test. It reduces number of test case and saves the money
  + **Example**:

Enter the number

**\***Allow digits from 1-500

|  |  |  |
| --- | --- | --- |
| **Normal test case** | **Divide values into classes (-> take any value)** | **Test data Using ECP** |
| 1 | -100 to 0 -> -50 (invalid) | -50 |
| 2 | 1-100 (valid)-> 30 | 30 |
| 3 | 101-200 (valid) ->160 | 160 |
| . | 201-300 (valid)->250 | 250 |
| . | 301-400 (valid)->320 | 320 |
| . | 401-500 (valid)->450 | 450 |
| 500 | 501-600 (invalid)->550 | 550 |

**160** is remain in (201-300) so for all the value between (201-300) is valid test data and (201-300) is considered as a class

* **Boundary value analysis:** 
  + Focus on boundary on the value
  + It is used to check boundary of the input
  + **Example:**

Enter the age

**\* Allow digits from 18-35**

**min-1 =17 min=18 min+1=19 max-1=34 max=35 max+1=36**

min =18 (pass) max=35 (pass)

min-1 = 17 (fail) max-1 = 34(pass)

min+1 = 19 (pass) max +1 = 36 (fail)

* **The value will be verified in the text box/inputs fields is called input domain testing. ECP and BVA are input domain testing**
* **Decision table-based testing:**
  + It is cause effect technique
  + If more conditions are available then it is used corresponding actions
  + Deal with combinations of inputs
  + Identify the test case with decision table, we consider conditions and actions

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **TC1** | **TC2** | **TC3** | **TC4** | **TC5** |
| **Condition1** | Account already approved | **True** | **True** | **True** | **True** | **False** |
| **Condition2** | OTP Matched | **True** | **True** | **False** | **False** |  |
| **Condition3** | Sufficient money in the AC | **True** | **False** | **True** | **False** |  |
| **Action1** | Transfer money | **Execute** |  |  |  |  |
| **Action2** | Show msg insufficient AC |  | **Execute** |  |  |  |
| **Action3** | Block the transaction |  |  | **Execute** | **Execute** |  |

* **State transition:** 
  + If changes in input conditions , it change the sate of the application
  + Tester can perform the action by entering the various input conditions in a sequence
  + Here provides positive and negative inpu test values for evaluating the system behavior
  + **Example:** login page of an application which lock the user name after three wrong attemps of password

s1

pass s4

s2

pass

pass

s3

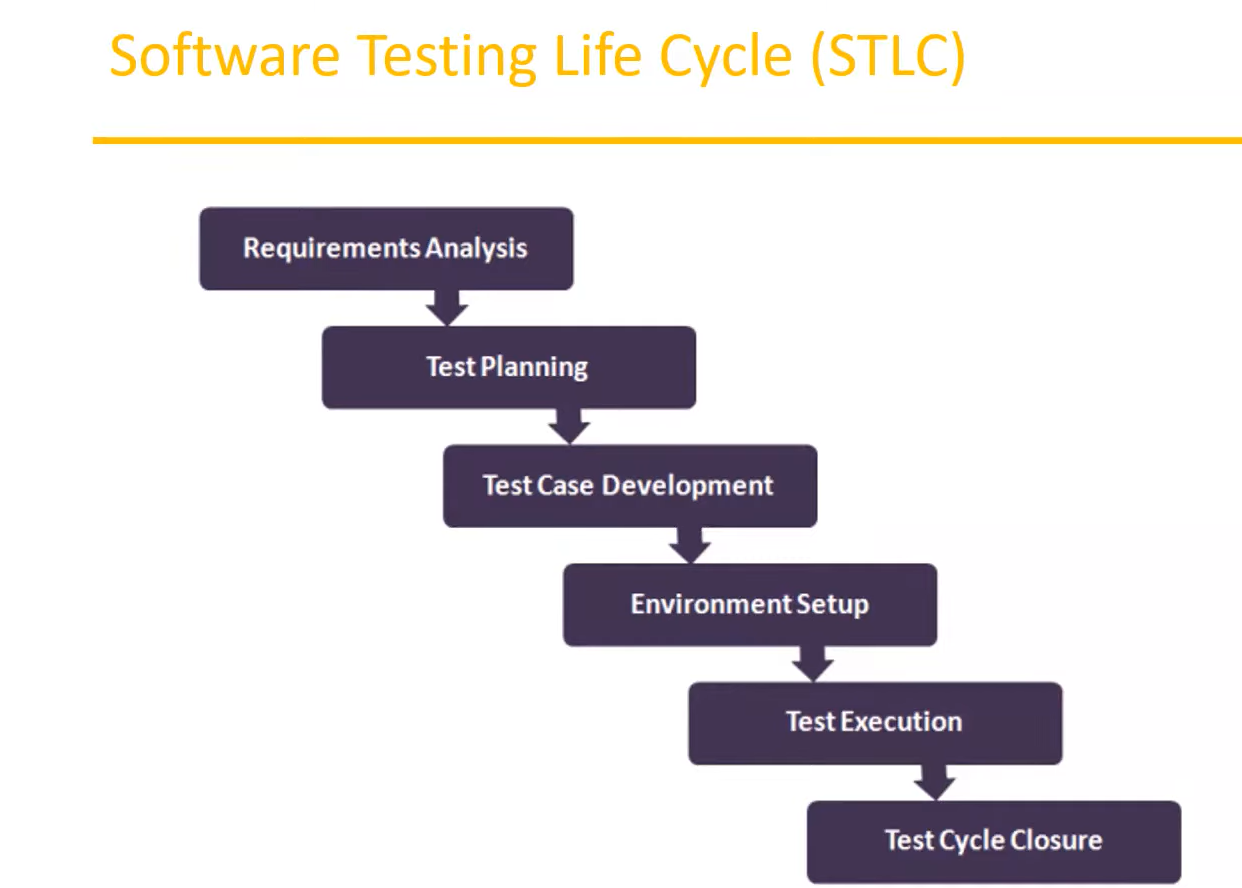
s5

fail

|  |  |  |  |
| --- | --- | --- | --- |
| state | login | Current pass | Wrong pass |
| S1 | 1st attempt | S4 | S2 |
| S2 | 2nd attempt | S4 | S3 |
| S3 | 3rd attempt | S4 | S5 |
| S4 | Home page |  |  |
| S5 | AC locked |  |  |

* **Error guessing:**
  + Used to find bugs in a software based on tester’s experience
  + Do not follow specific rules
  + It depends on tester analytical skill and experience
  + **Example:**
    - Submitting form without entering value

**STLC (Software testing life cycle) phase:** it is part of SDLC



* **Requirement Analysis**
* **Test Planning:** 
  + A test plan is a document that describes the test scope, test strategy, objectives, schedule, deliverables and resources required to perform testing for a software
  + **Test plan template content:**
    - **Overview**
    - **Scope**
      * **Inclusions**
      * **Test environments**
      * **Exclusions**
    - **Test strategy**
    - **Defect reporting procedure**
    - **Roles**
    - **Test schedule**
    - **Test deliverables**
    - **Pricing**
    - **Entry and exit criteria**
    - **Suspension and resumption criteria**
    - **Tools**
    - **Risks and mitigations**
    - **Approvals**
* **Test Designing**

**Use case:**

* Use case describe the requirements like diagram to understand clear and it is prepared by Business analyst.
* **Contain three items**
* **Actor**: which is user, can be single or group of peoples, interacting with process
* **Action**: which is to reach the final outcome
* **Outcome**: which is the successful user outcome/goal

**Test scenario:**

* A possible area to be tested (**what is tested**)

**Test case:**

* Step by step action to be performed to validate functionality of software application (**how to test**)
* Test case contain tests steps, expected results and actual results and it is prepared by Business tester.

**Test suite:**

* It is group of test case which belong to same category
* GUI test suite, functional test suite etc.

**Test case content:**

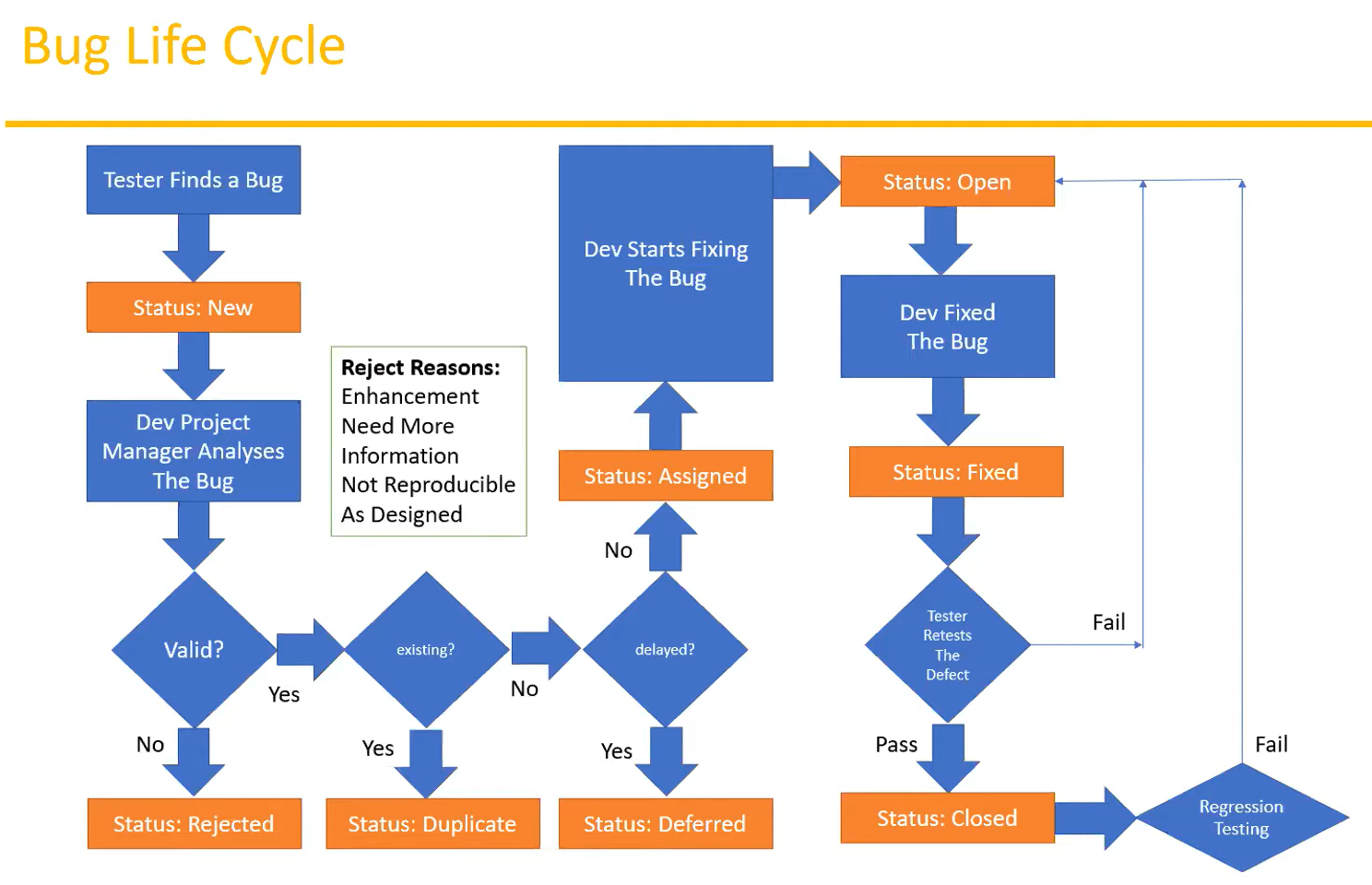
* Test case ID
* Test case title
* Description
* Pre-conditions
* Priority
* Requirement ID
* Steps
* Post condition
* Expected result
* Actual result
* Test data

**Requirement traceability matrix (RTM):**

* It is mapping of requirements with the test cases.
* The main purpose of RTM is to see all the test cases are covered so that no functionality should miss while doing software testing.it contain requirement ID corresponding to test cases ID.
* **RTM parameters:**
  + Requirement ID
  + Req description
  + Test case ID

**Test environment:**

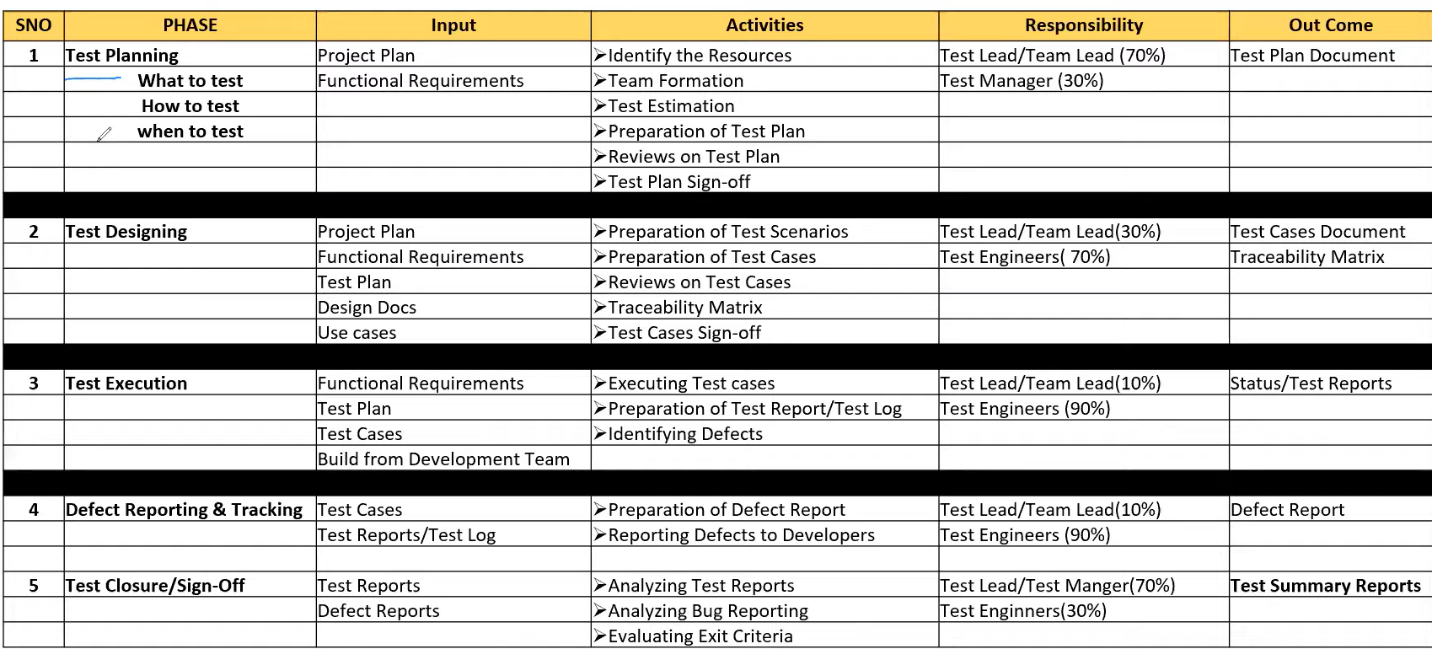
* It is a platform specially build for test case execution on the software products
* It is created by integrating the required software and hardware along with proper network configurations. That’s mean setup the customer environments to execute the software properly
* Another name is **test bed**
  + - **Test Execution:**
      * During this phase test team will carry out the testing based on the test plan and the test case prepared
      * **Entry criteria:** test case, test data, test plan
      * **Activities:**
        + Executed based on test planning
        + Status of test case marked like pass or fail
        + Documentation of test results and log defects for failed case is done
        + All the blocked test cases are assigned bug ids
        + Retesting once the bugs are fixed
        + Defects are tracked till closure
      * **Deliverables:** provides defects and test cases execution report with completed result**.**
* **Defect/Bug Reporting and Tracking:**
  + Any mismatched functionality found in a application is called a defect/bug/issue
  + **Defect reporting tools**
    - Clear Quest
    - DevTrack
    - Jira
    - Bug Jilla etc.
  + **Defect report content**
    - **Defect ID**
    - **Defect description**
    - **Version**
    - **Steps**
    - **Date raised**
    - **Reference**
    - **Detected by**
    - **Status**
    - **Fixed by**
    - **Date closed**
    - **Severity:** which describe the impact of the defect on theapplication
    - **Priority**
  + **Defect category:**
    - **Severity:** describe seriousness of defect and how much impact on business workflow. **Types of severity:**
      * **Blocker**
      * **Critical**
      * **Major**
      * **Minor**
    - **Priority:** describes the importance of defects
      * **P1 (High):** must be resolved immediately as it affects the system and cannot be sed until it is fixed
      * **P2 (Medium):** It can wait until a new version is created
      * **P3 (Low):** Developer can fix it in later releases
  + **Defect Resolution:**
    - After receiving the defect report from the testing team, development team conducts a review meeting to fix defects, the they send a resolution type to the testing team for further communication
    - **Resolution type:**
      * Accept
      * Reject
      * Duplicate
      * Enhancement
      * Need more information
      * Fixed
      * As designed
  + **Defect life cycle:**

****

* **Test Closure:**
  + **Activities:**
    - Evaluate cycle completion based on time, test coverage, cost, software, quality etc.
    - Prepare test metrics based on the above parameters
    - Prepare test summary reports
    - Test result analysis to find out the defect distribution by the type and severity
    - Document the learning out of the project
  + **Deliverables:**
    - Test closure report
    - Test metrics

**Table of STLC process:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SNO** | **Phase** | **Input** | **Activities** | **Responsibility** | **Out come** |
| 1 | Requirement analysis |  | Understand the customer requirements | Business people |  |
| 2 | Test planning  =>what to test  =>how to test  =>when to test | =>Project plan  =>Functional requirements | =>Identify the resources  =>Team formation  =>Test estimation  =>Preparation of test panning  =>Review on test plan  =>Test plan sign-off | Team lead  (70%)  Team manager (30%) | Test plan documents |
| 3 | Test designing | =>project plan  =>Functional requirements  =>test plan  =>design doc  =>use cases | =>preparation of test scenario  =>preparation of test case  =>review on test case  =>traceability matrix  =>test case sign-off (tester, developer, all need) | Team lead (30%)  Test engineer (70%) | =>Test case documents  =>traceability  matrix |
| 4 | Test execution | =>Functional requirements  => test plan  =>test case  =>Build from dev team | =>executing test case  =>preparation of test reports  =>identify the defects | Team lead (10%)  Test engineer (90%) | Test reports |
| 5 | Defect/Bug Reporting and Tracking | =>test case  => Test reports | => preparation of defect reports  =>Reporting defects to developer | Team lead/test lead (10%)  Test engineer (90%) | Defect reports |
| 6 | Test Closure/sign -off | =>test reports  =>Defects reports | =>analyzing test reports  =>analyzing bug reports  => Evaluating exit criteria | Team lead (70%)  Test engineer (30%) | Test summary reports |



* For one test scenario, have multiple test case
* Mapping between test case and requirements id is called traceability matrix. For one requirement how many test cases are created and that’s way many requirements id have lots of test cases and they are mapped to each other is called traceability matrix