



Manual Testing Notes By
THIRU

Manual Testing Course Content

Introduction to Software Testing

- What is Software Testing

- Why Software Testing

- Benefits of Software Testing

- What is Quality

Overview of Software Testing

Why Testing Required

Why Testing Job

Testing Roles & Responsibilities

- Software Test Engineer Responsibilities

- Sr. Software Test Engineer Responsibilities

- Test Lead Responsibilities

- Test Manager Responsibilities SDLC–

Software Development Life Cycle

- Phases in SDLC :

SDLC Real Time Process Implementation

Types of SDLC Models

- Waterfall model

- V - model

- Agile - Process

Software Testing Methodologies

- Block Box Testing

- White box testing

Real Time Folder Structure To implement STLC

STLC – Software Testing Life Cycle

- STLC Phases

Test Plan

- Entry/Exit Criteria to prepare Test Plan :

- Contents in Test Plan

Test Scenarios

- How to create a Test Scenario

- Entry/Exit Criteria to Identify Test Scenarios :

- Test Scenarios Template

- Sample Example of TestScenarios

Test Case

- What is Test case?

- The benefits of an effective test case include:

- Entry/Exit Criteria to Identify Test Cases :

- Test Case Template

- Sample Example of TestCases

Test Case Design Techniques

-Black box techniques

-White box techniques

Test Execution - Bug or Defect Management

-Bug / Defect Template

-Sample Example of BugReporting

Bug or Defect Life Cycle

-What is a defect/bug lifecycle in software testing?

-Difference between Defect /Bug/Error/Failure

JIRA Tool:

-Jira Login Process:

-Steps to Create Test Cases in JIRA :

-How to Pass a test case:

-Steps to Report Bugs in JIRA

Types of Testing

-SmokeTesting

-Sanity Testing

-Re Testing

-Regression Testing

-Static Testing

-Dynamic Testing

-Adhoc Testing

-Alpha Testing

-Beta Testing

-Functionality Testing

-Usability Testing

-Compatibility Testing

-Database Testing

-interfaceTesting

-Performance Testing

-Security Testing

Test Closure

When to stop Testing?

What is Software Testing

“Software testing is a process of executing the application with the intent of finding the defects by comparing the output behavior of the application with expected behavior (requirement).”

In other words it's comparing the actual behavior of an application with expected behavior.

Why Software Testing

Humans make mistakes all the time!!

“Software testing is really required to point out the defects and errors that were made during the development phases”.

We humans can't identify our mistakes in a work done by us. We should get someone else to check our work because another person may identify the mistakes done by us. In the same way software

developers may not identify the mismatches in a program or application implemented by them which can be identified by the another department called Software Test Engineer.

Benefits of Software Testing

“Software testing helps in finalizing the software application against business requirements.”

Software testing makes sure that the testing is being done properly and hence the system is ready for the customers to use.

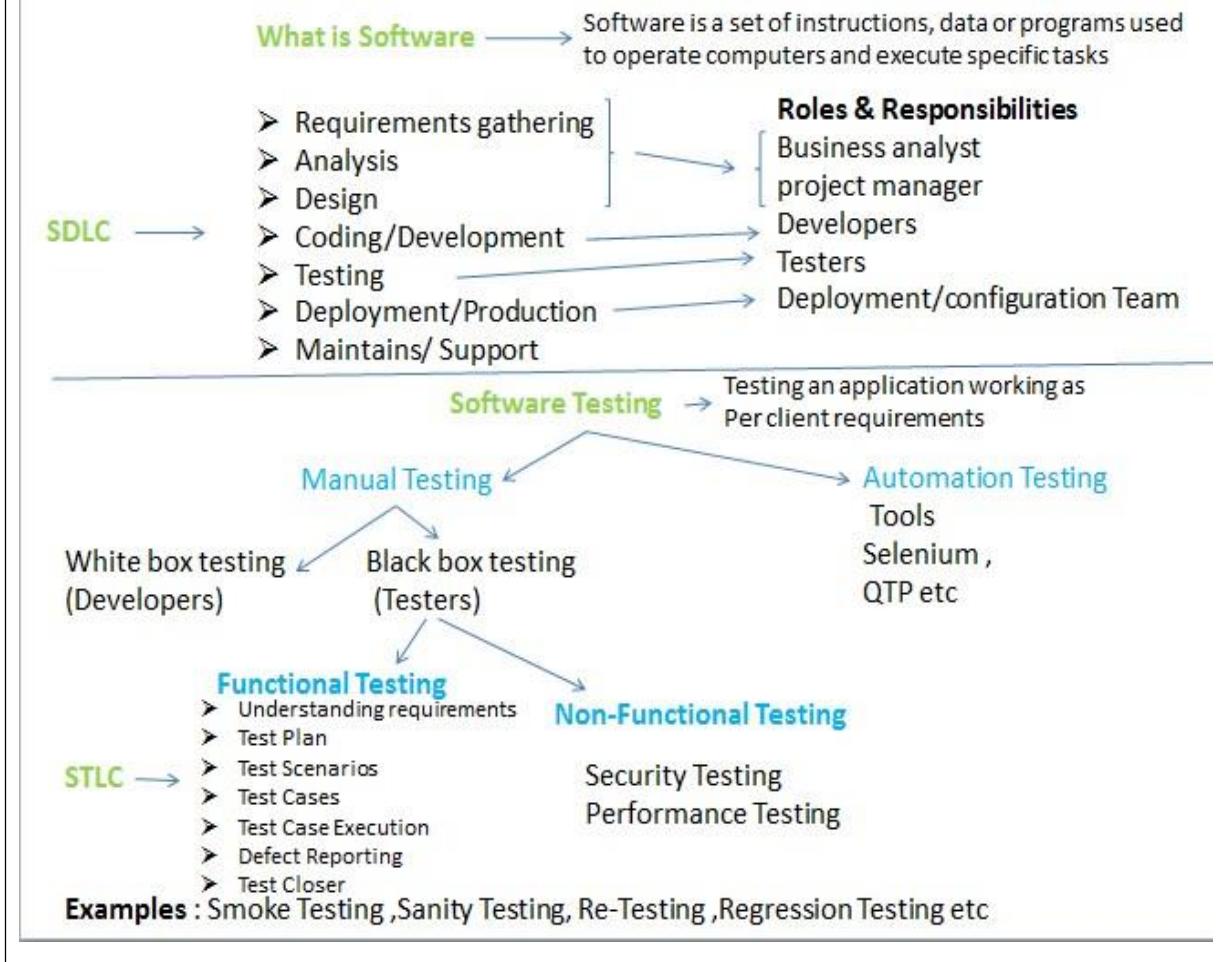
Below are few benefits of software testing.

- Finding the defects before delivery
- Gains the confidence about quality
- To Prevent defects
- Ensure the requirements are delivered to client

What is Quality

“Software quality is nothing but delivering a bug free application and delivered on time with all requirements.”

Overview of Software Testing



Why Testing Required

- To identify the defects in development phases
- To ensure Quality of the product
- Saves Money as defect identified in earlier stages
- To build customer confidence and business

Why Testing Job

- Software Testers Are Made for Challenging Work Environments
- You Can Enjoy Every Day of Work
- Flexible and Fun Work Environment
- It's Creative
- It Is a Secure Career Path
- There Is Attractive Remuneration and Room for Growth
- An Academic Background Isn't a Necessity

Testing Roles & Responsibilities

- Software Test Engineer(STE) – (0-4yrs)
- Sr. Software Test Engineer(SSTE) – (4-6yrs)
- Test Lead (TL) – (6-8yrs)
- Test Manager(TM) – (8-10yrs)
- Sr.Test Manager(STM) – (10+yrs)
- Assoc. Dir. – Software Testing
- VP – Software Testing

Software Test Engineer Responsibilities

- Understating the requirements of the application
- Identifying required Test Scenarios of the project
- Designing and preparing Test cases to validate application
- Execute test cases to validate application
- Logs Test Results(How many Test cases passes/failed)
- Defect Report and Tracking
- Retest fixed defects of previous builds
- Performed various Types of testing assigned by Test Lead(Sanity ,Functionality , Usability, Compatibility , etc)
- Preparing and Sending of status Reports to Lead on assigned tasks
- Participated in regular meetings , team meetings by lead & Manager
- Creating automation scripts for Regression testing

Sr. Software Test Engineer Responsibilities

- Same as test engineer responsibilities
 - +
- Participates in Review of Test Scenarios ,Test cases and defects
- Some Times involved in Test Plan preparation also.
- If required Leading the team when Team Lead is on Vacation

Test Lead Responsibilities

- Task Preparation and allocation to Team members
- Training Team members
- Team Management
- Test Scenarios & Test Cases Reviews
- Bug Reviews
- Preparation of Build summary report
- Conducting meetings with Team members

- MOM Preparation
- Test Plan preparation

Test Manager Responsibilities

- Project plan and Review of Test Plan
- Effort estimates
- Project Management
- Training Plan – Identify training need based on Resource skills
- Preparing monthly reports
- Client Communications
- MOM
- Provide regular status updates to core team
- Scheduling meetings with Development and Testing Team

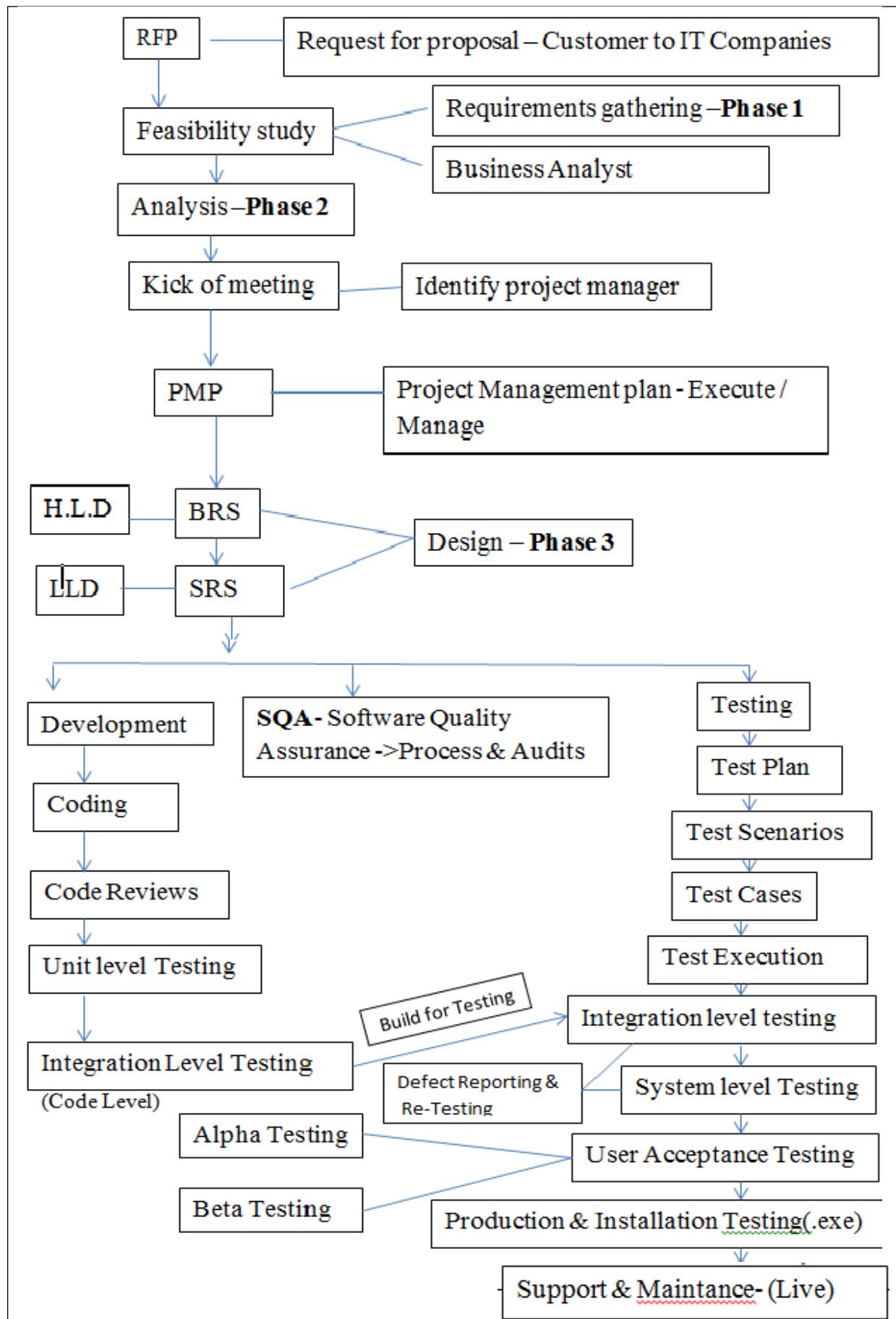
SDLC–Software Development Life Cycle

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software.

Phases in SDLC :

- Requirements gathering
- Analysis
- Design
- Coding/Development
- Testing
- Deployment/Production
- Maintains/Support

SDLC Real Time Process Implementation :

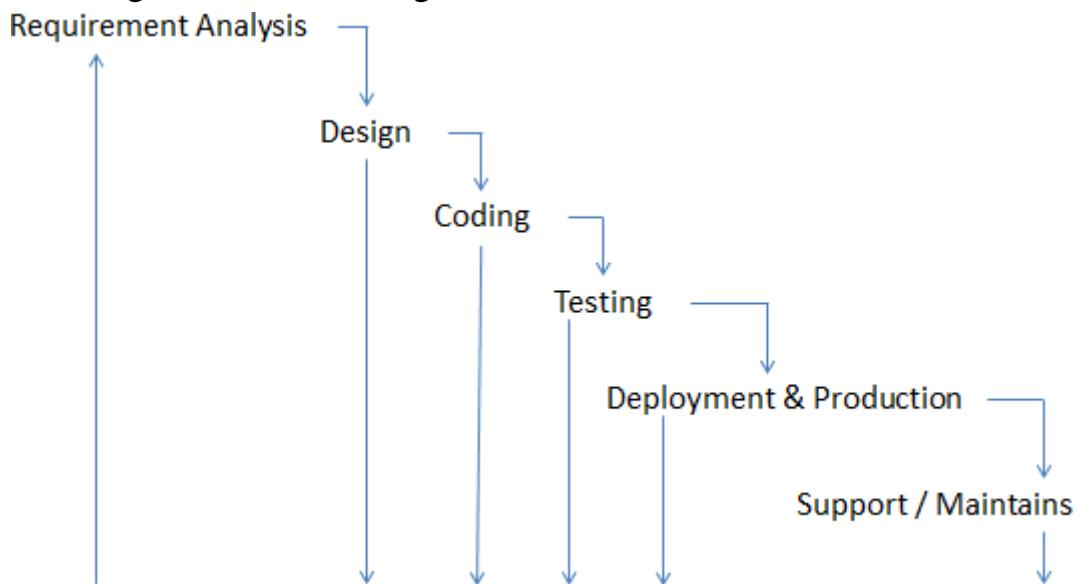


Types of SDLC Models

- Waterfall model or Life Cycle Model or Linear Sequential Model
- V- Model or Verification & Validation Model
- Agile ...etc

Waterfall model

- This model suggests a systematic and sequential approach to software development that begins at requirements analysis and progress through all life cycle phases sequentially
- Suitable for projects where requirements are clearly defined
- Small and medium term duration
- Having Domain knowledge



Advantages :

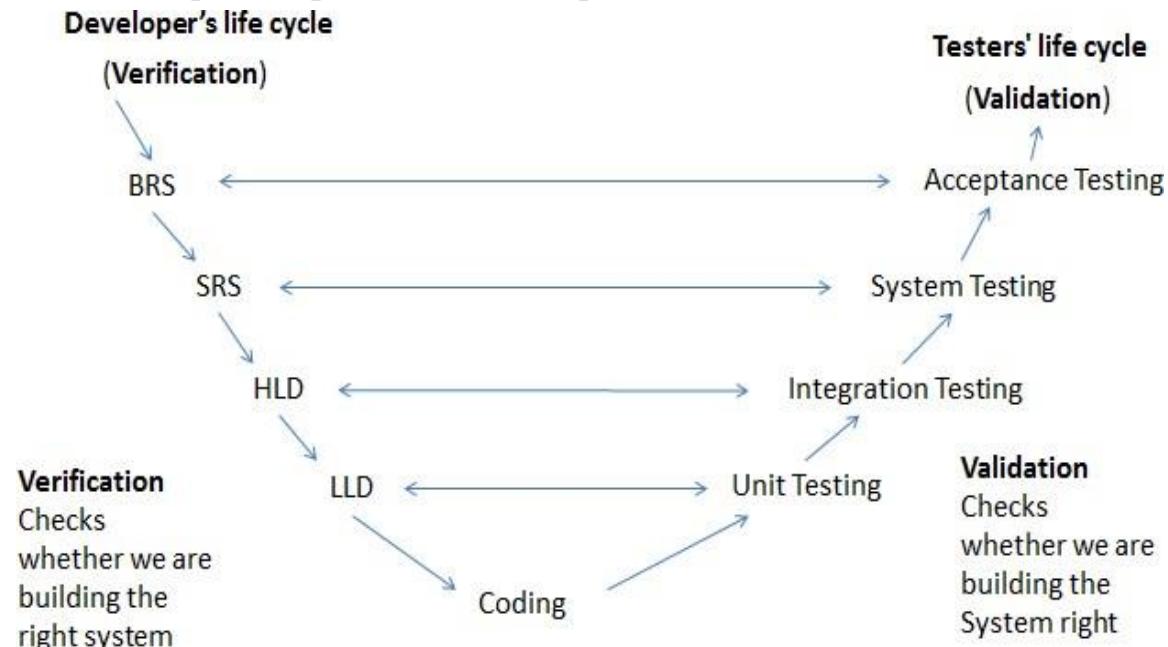
- Project under Control
- Pre-defined outputs at every phase
- Tracking changes is easy

Disadvantages:

- Not suitable for requirements changes
- Does not support going back to previous phase
- If any defect found need to go back to the originating phase

V - model

- V model means verification and validation model, the V shaped life cycle is a sequential path of execution process.



Advantages :

- Simple and easy to use
- Testing activities like planning and test design will be done before coding
- Testing planned parallel to development
- Bug detection in early phases

Disadvantages:

- If any changes happen in midway, then the test document along with requirements documents has to be updated.
- Not Suitable for large and complex projects
- The client sees the only final project ,not intermediate modules

Difference between Verification & Validation :

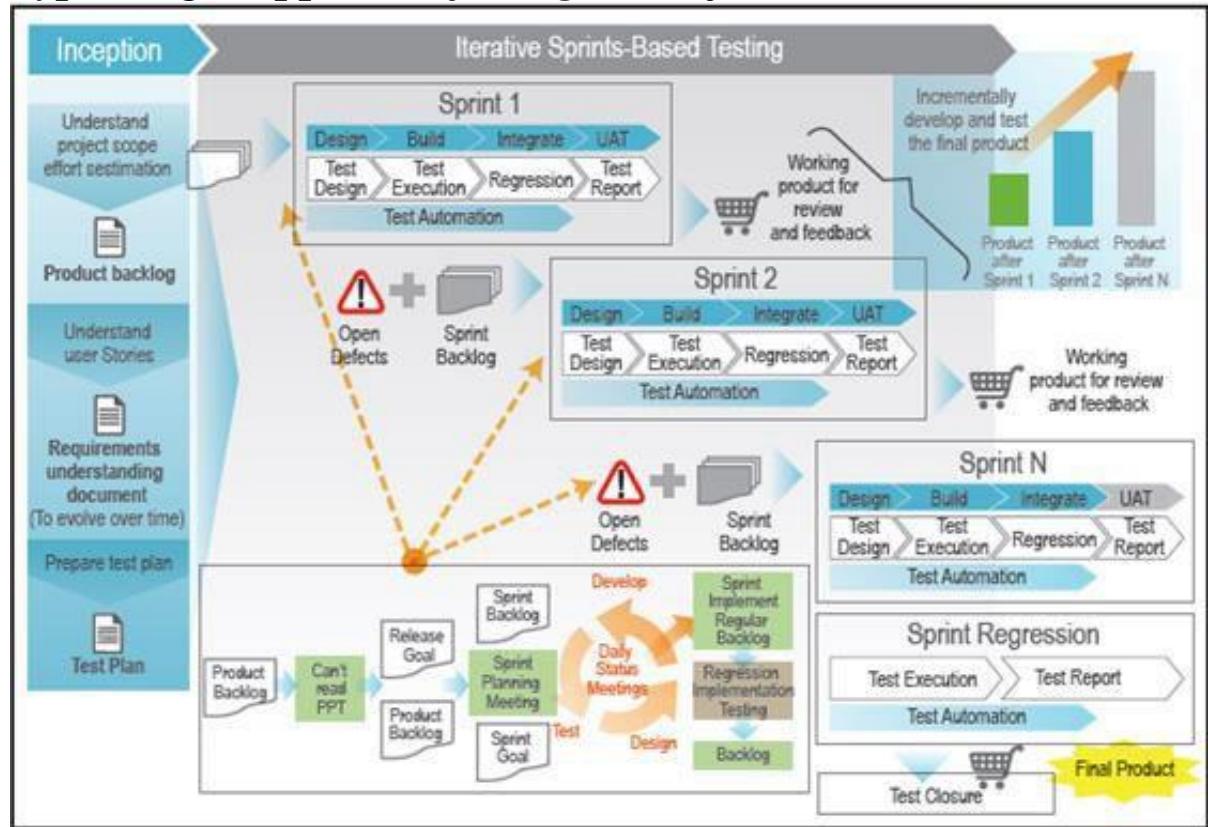
Verification	Validation
check whether we are developing the right product or not.	check whether the developed product is right.
Verification includes different methods like Inspections, Reviews, and Walkthroughs	In the validation testing, we can find those bugs, which are not caught in the verification process.
In verification testing, we can find the bugs early in the development phase of the product.	Validation includes testing like functional testing, system testing, integration, and User acceptance testing.
The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.

Agile - Process

Overview of Agile Process

- Customer satisfaction by rapid delivery of useful software.
- Welcome changes in requirements ,even late in development
- Working software is delivered frequently (weeks rather than months)
- Close ,daily cooperation between business professionals and developers
- Continuous attention to technical excellence and good design

Typical QA Approach for Agile Projects



Agile - Terms

- **User Story** – A shorthand requirements document.
- **Product Backlog** -- A prioritized list of stories that are waiting to be worked on.
- **Product Owner** -- person whom holds the vision for the product.
- **Scrum Master Role** The Scrum Master is a facilitator for the team and product owner.
- **Sprint** -- A development process
- **Stand-up Meeting**– a short (15 minutes or less) daily meeting during which team members report on what they have accomplished since the last meeting, what they plan to accomplish today and report any impediments or blockers to making progress.
- **Scrum Meeting** – To discuss on Sprint planning, development and Review.

Software Testing Methodologies

- Block Box Testing
- White box testing

Black Box Testing:

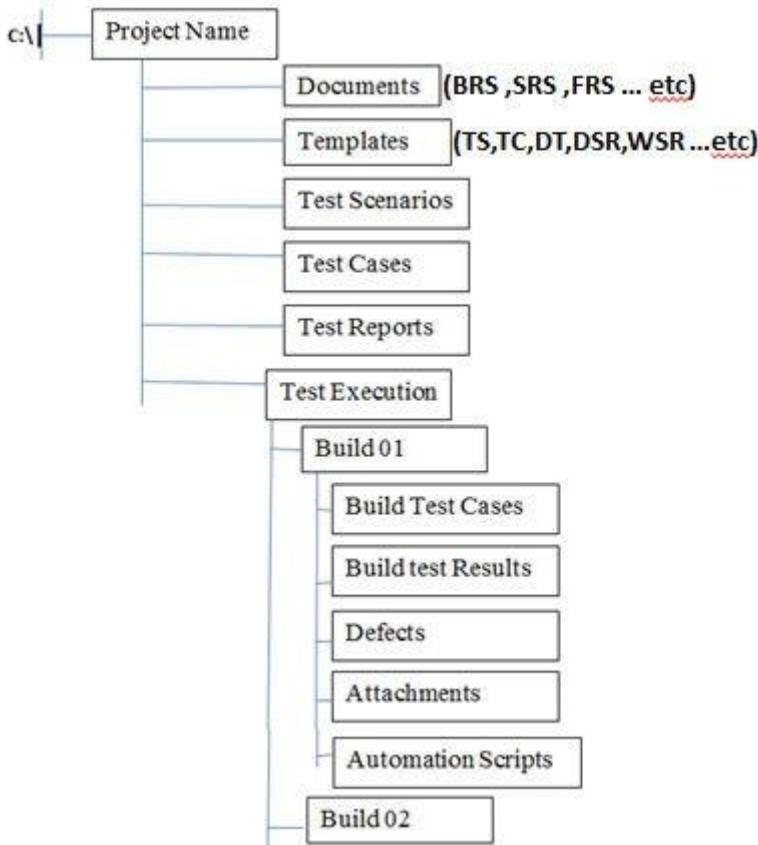
Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues and reliability issues.

Black box testing is a powerful testing technique because it exercises a system end-to-end. Just like end-users “don’t care” how a system is coded or architected, and expect to receive an appropriate response to their requests, a tester can simulate user activity and see if the system delivers on its promises. Along the way, a black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.

White Box Testing

White box testing involves testing an application with detailed inside information of its source code, architecture and configuration. It can expose issues like security vulnerabilities, broken paths or data flow issues, which black box testing cannot test comprehensively or at all.

Real Time Folder Structure To implement STLC



STLC – Software Testing Life Cycle

Software Testing Life Cycle (STLC) is a process used to test software and ensure that quality standards are met. Tests are carried out systematically over several phases. During product development, phases of the STLC may be performed multiple times until a product is deemed suitable for release.

- STLC is an part of Software Development Life Cycle (SDLC).STLC deals only with the testing phases.
- STLC starts as soon as requirements are defined.
- STLC provides a step-by-step process to ensure quality software.

STLC Phases

STLC has the following different phases but it is not mandatory to follow all phases. Phases are dependent on the software or the product, time and resources allocated for the testing and the model of SDLC that is to be followed.

- Requirements understanding (BRS, SRS, FRS, Mock-ups)
- Test Plan
- Test Scenarios
- Test Cases

- Test Execution
- Bug Reporting & Re-Testing
- Test Closer

Note:

BRS – Business Requirement Specification

SRS – Software Requirement Specification

FRS – Functional Requirement Specification

Mock-Ups : Screenshots

Requirements understanding:

When the requirement docs are ready and shared with testing team, then testers starts understanding the requirements to design test scenarios & testcases.

Test Plan:

Test Lead plans the test strategy & approach which is outlined in a test plan document. This strategy includes tools needed, testing steps, and roles and responsibilities. Part of determining this strategy is a risk and cost analysis and an estimated timeline for testing.

Test Scenarios:

(What need to tested)-Design the test scenarios based on scope and criteria's.

Test Cases:

(How to be tested) test cases are created. Each case defines test inputs, procedures, execution conditions, and anticipated results. Test cases should be transparent, efficient, and adaptable. Once all test cases are created, test coverage should be 100%.

Test Execution:

features are tested in the deployed environment, using the established test cases. Expected test results are compared to actual and results are gathered to report back to development teams.

Bug Reporting & Re-Testing : Reporting the bugs using bug template

Test Closer:

This is the last phase of the STLC, during which a test result report is prepared. This report should summarize the entire testing process and provide comparisons between expected results and actual. These comparisons include objectives met, time taken, total costs, test coverage, and any defects found.

Test Plan

Test Plan – A document describing the scope, approach, resources & schedule of testing activities. It identifies test items ,the features to be tested, the testing tasks who will be doing and any risks ...etc

Entry Criteria to prepare Test Plan :

- Approved PMP
- Approved SRS
- Test Plan guidelines
- Test Plan template

Exit Criteria of preparing Test Plan :

- Test Plan should be reviewed and approved.

Contents in Test Plan

1. Introduction

1.1 Test plan objectives

2. Scope of this document

3. Test Strategy

- 3.1 Smoke Testing
- 3.2 Sanity Testing
- 3.3 Functional Testing
- 3.4 Database Testing
- 3.5 Cross browser Testing
- 3.6 Automation Testing

4. Environment Requirements

5. Test Schedule

6. Control Procedures

- 6.1 Reviews
- 6.2 Bug Reviews

7. Functions To be tested

8. Functions not to be tested

9. Resources & Responsibilities

10. Deliverables and mile stones

11. Defect Management

12. Dependencies

- 12.1 personal dependencies
- 12.2 Software dependencies
- 12.3 Hardware dependencies
- 12.4 Test Data & Database

13. Risks

14. Tools

- 15. Documentation**
- 16. Approvals**
- 17. Entry / Exit for each Testing activity**
- 18. Test Suspension**
- 19. Test Resumption**
- 20. Test completion**

Test Scenarios

Identify all possible areas to be tested **or** What is to be tested.

A Test Scenario is a statement describing the functionality of the application to be tested. It is used for end-to-end testing of a feature and is generally derived from the requirement document.

Test scenarios can serve as the basis for lower-level test case creation. A single test scenario can cover one or more test cases. Therefore a test scenario has a one-to-many relationship with the test cases.

How to create a Test Scenario

- Carefully study the Requirement Document – Business Requirement Specification (BRS), Software Requirement Specification (SRS), Functional Requirement Specification (FRS) pertaining to the System Under Test (SUT).
- Isolate every requirement, and identify what possible user actions need to be tested for it. Figure out the technical issues associated with the requirement. Also, remember to analyze and frame possible system abuse scenarios by evaluating the software with a hacker's eyes.
- Enumerate test scenarios that cover every possible feature of the software. Ensure that these scenarios cover every user flow and business flow involved in the operation of the website or app.
- After listing the test scenarios, Get the scenarios reviewed by a team.

Entry Criteria to Identify Test Scenarios :

- Approved Test plan
- Approved SRS
- Test Scenarios Guidelines
- Test Scenario template

Exit Criteria for Test Scenarios :

- Test Scenarios should be reviewed & approved(mapping test scenarios with requirements)

Test Scenarios Template

Sample Example :

Company Logo : Logo of the company

Project ID : ID of the Project

Project Name : Name of the Project

Identified by : Name of the tester

Date Identified : Day of the started writing the Test Scenarios

Reviewed By : Test Lead Name

Date Reviewed : Future Date

Approved By : Test Lead Name

Date Approved : Future Date

TS# : Test Scenario ID

Req:# : Requirement ID

Main Functionality : Page Name

Test Scenario Description : What need to be Tested

Test case name /ID : Test Case ID for corresponding Test Scenario

Document ID / Reference : Which document we are referring to design the Scenarios

Comments : In case of any specific description need to be mention.

TS#	Req#	Main Functionality	TestScenario Description	Test Case Name/TC ID	Document ID/Reference	Comments
TS_001	3.2	Login Page	Verify Login functionality		SRS	
TS_002	3.2	Login Page	Verify forget password functionality		SRS	
TS_003	3.2	Login Page	Verify new user register functionality		SRS	

Test Case

What is Test case?

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

It is an in-details document that contains all possible inputs (positive as well as negative) and the navigation steps, which are used for the test execution process. Writing of test cases is a one-time attempt that can be used in the future at the time of regression testing.

Test case gives detailed information about testing strategy, testing process, preconditions, and expected output. These are executed during the testing process to check whether the software application is performing the task for that it was developed or not.

Test case helps the tester in defect reporting by linking defect with test case ID. Detailed test case documentation works as a full proof guard for the testing team because if developer missed something, then it can be caught during execution of these full-proof test cases.

The benefits of an effective test case include:

- Guaranteed good test coverage.
- Reduced maintenance and software support costs.
- Reusable test cases.
- Confirmation that the software satisfies end-user requirements.
- Improved quality of software and user experience.
- Higher quality products lead to more satisfied customers.
- More satisfied customers will increase company profits.

Entry Criteria to Identify Test Cases :

- Approved Test plan
- Approved SRS
- Approved FRS
- Approved Test Scenarios
- Test Case Guidelines
- Test Case Template

Exit Criteria for Test Cases :

- Test Cases should be reviewed & approved(mapping test scenarios with requirements)

Test Case Template

Manual Testing Notes by Thiru

--Sample Example :

Company Logo : Logo of the company

Project ID : ID of the project

Project Name : Name of the Project

Created By : Tester name

Review By : Test Lead name

Approved By : Test Lead name

Date Created : Day of the Started writing Test Cases

Date Reviewed : Future Date

Date Approved : Future Date

Test Executed By :

Version : Version of the Test Case Document

Date Executed : Form Which Date Test Case Execution need to start

Build : build number

Total No of Test Cases : The Total number of Test Cases Created for whole application.

Passed: Total number of Test Cases passed

Failed: Total number of Test Case Failed

Not Executable: not executed test case count

Defects Reported: Bugs reports for the Whole application.

TC# : Test Case ID

TS# : Test Scenario ID mentions in Test Scenario Document

Test Design Steps : Steps To be Followed for Testing particular Functionality

Input Date : Test Data need to be used to test particular Functionality

Expected Result : Result Excepting Based on Requirement Document

Actual Result : Result displayed in Application

PASS : Expected Result & Actual Result both are same.

FAIL : Expected Result & Actual Result both are not matching consider as FAIL

FAIL : Expected Result & Actual Result both are not matching consider as FAIL **Not Executable :** Test Case Which are not executed for that current

TALENT Executables : Test Case which are not executed for that current Release **Comments :** In case of any specific description need to be mention

Defect ID: Bug ID for failed test Cases

Defect ID : Bug ID for failed test Cases

TC #	TS#	Test Design/ Steps	Input Data	Expected Result	Actual Result	Pass	Fail	Not Executable	Comments	Defect Id
TC_001	TS_001	1.Launch browser 2.Enter URL 3.Enter Valid UserName 4.Enter Valid Password 5.Click on Login button	<u>URL :</u> <u>Username :</u> <u>Password :</u>	1.Login should be successful 2.Search hotel page need to display						
TC_002	TS_001	1.Launch browser 2.Enter URL 3.Enter Invalid UserName 4.Enter Valid Password 5.Click on Login button	<u>URL :</u> <u>Username :</u> <u>Password :</u>	Error message need to displayed as "Invalid UserName"						

Test Case Design Techniques

Test Case: Design techniques can broadly split in to 2 categories.

Black box techniques

White box techniques

Black box techniques

- Equivalence Class Partitioning(ECP)
- Boundary Value Analysis(BVA)
- State Transition
- Decision Table / Cause Effect Table

White box techniques

- Statement Testing
- Branch/Decision testing
- Data flow Testing
- Branch condition testing

Equivalence Class Partitioning (ECP) :

Divide a set of test conditions into groups which is having similar behavior to reduce number of test cases.

Age	<input type="text"/>	(accepts 1 to 60)
Invalid	Valid	Invalid
0	1 to 60	>=61

Boundary Value Analysis(BVA):

BVA is based on testing the boundaries of condition

Formula : Minimum ,maximum ,Min-1 ,Max+1

Valid Boundaries : Minimum , Maximum

Invalid Boundaries : Min-1 ,Max-1

Name	<input type="text"/>	(accepts 5 to 10 characters)
------	----------------------	------------------------------

Invalid (Min-1)	Valid (Min,Max)	Invalid (Min+1)
4	5 , 10	11

State Transition Table:

Application provides different output for same input based on previous stage

Username	Password	Result
Correct	Wrong	Invalid Password
Correct	Wrong	Invalid Password
Correct	Wrong	Invalid Password
Correct	Wrong	Account Locked

Decision Table :

Testing with different combination of inputs which produce different results.

Username	Password	Result
Correct	Correct	Login Successful
Correct	Wrong	Invalid Password
Wrong	Correct	Invalid Username
Wrong	Wrong	Invalid Username

Test Execution - Bug or Defect Management

Bug/Defect ID	Assigned To				
Status	Browser				
Severity	Found in Version				
Priority	Found in Build				
Module	Fixed in version				
Reported By	Fixed in build				
Title					

Description

Steps To Re-Produce :

- 1.
- 2.
- 3.

Expected Result :

Actual Result :

Sample Example :

Defect ID : ID of the new Defect

Status : Status of the Bug

Severity : Need to provide based on Functionality

Priority : Need to provide based on Client Expectation

Module : Page Name

Reported By : Tester Name

Assigned to : Developer or Test Lead Name

Browser Name : Name of the Browser

Found in Build: in which build bug had found

Found in Version: number of the version

Fixed in Version : Developer will provide this

Fixed in Build :Developer will provide this

Bug / Defect Template			
Defect ID	DEF_001	Assigned To	DEV
Status	NEW	Browser	Chrome
Severity		Found in Version	1
Priority		Found in Build	1
Module	Search Hotel Page	Fixed in version	
Reported By		Fixed in build	
Title	Validation is not working for check in date (accepting previous date)		

Description : Steps To Re-Produce :

- 1.Launch Browser .2.Enter URL
- 3.Enter valid Username
- 4.Enter valid password
- 5.Click on login
- 6.Select all required feilds
- 7.Enter check in date as previous date
- 8.Click on search

Expected Result :

Error message should be displayed as "check in date should be current date or feature date"

Actual Result :

Search hotel page is displayed

Defect Status : New ,Open ,Fixed ,closed ,Re-Open ,Not a bug, Duplicate ,Need More information, Can't reproduce .

Severity : Importance of defect with respective to functional point of view...means criticalness of defect with respective to application.

Severity classification could be : S1-Urgent ,S2-High ,S3-Medium ,S4-Low , CRITICAL , HIGH ,MEDIUM ,LOW

Priority : Importance of defect with respective to client point of view ... means how soon it should be fix.

Priority Classification could be : P1-Urgent ,P2-High ,P3-Medium ,P4-Low, CRITICAL , HIGH ,MEDIUM ,LOW

Sample Examples :

High Priority & High Severity: An error which occurs on the basic functionality of the application and will not allow the user to use the system. (Eg. A site maintaining the student details, on saving record if it, doesn't allow to save the record then this is high priority and high severity bug.)

High Priority & Low Severity: The spelling mistakes that happens on the cover page or heading or title of an application.

High Severity & Low Priority: An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the

user to use the system but on click of link which is rarely used by the end user.

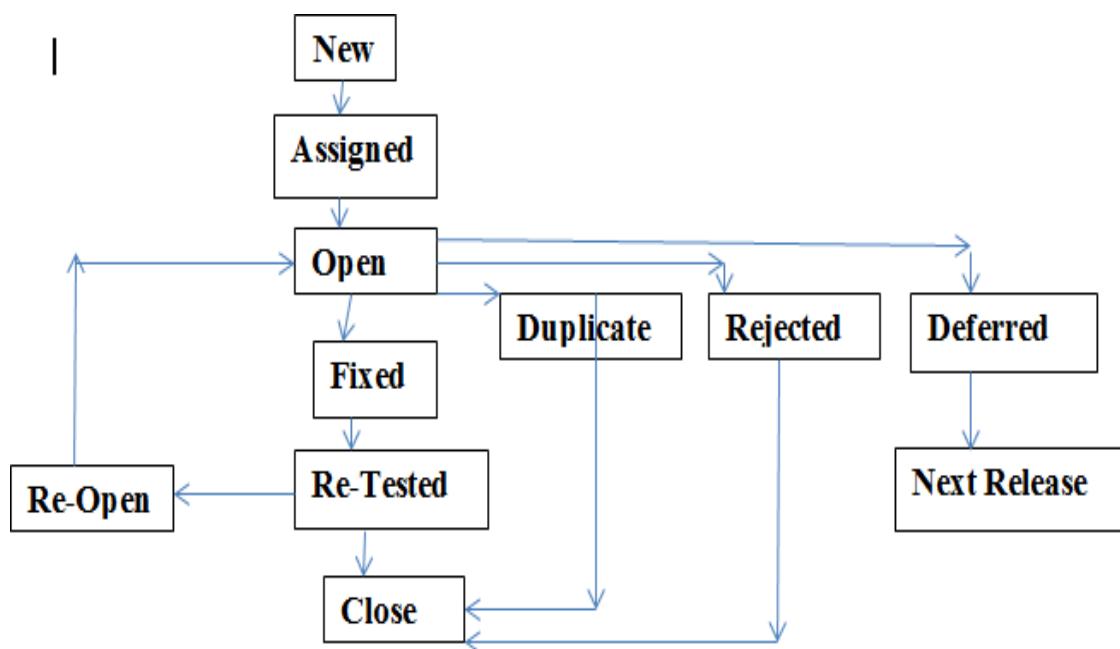
Low Priority and Low Severity: Any cosmetic or spelling issues which is within a paragraph or in the report (Not on cover page, heading, title).

Bug or Defect Life Cycle

What is a defect/bug lifecycle in software testing?

A defect lifecycle, or bug lifecycle, is a specific set of states that a software bug goes through from discovery to fixation.

The lifecycle may vary from organization to organization depending on factors like company policy, software developmental model (e.g., Agile, Waterfall, etc.), and project timeline. However, the actual extensive defect lifecycle is as below:



New: When a defect is logged and posted for the first time, its state is set as “new”.

Assigned: After the defect has been posted and verified as a bug by the testing team, it is assigned to the corresponding developer team.

Open: At this stage, the developer team has begun work on fixing the defect.

Fixed: After the necessary code changes are completed by the developers, the defect’s state is set to “fixed”.

Retest: At this stage, the testing team retests the code given to them by the developers.

Closed: Once the bug has been verified as fixed, the testing team closes the

issue.

Reopened: If the bug still exists, its state is set back to Open and the lifecycle restarts.

Duplicate: If the defect is repeated twice or if the defect corresponds to the same concept as the bug, the status is changed to “duplicate.”

Rejected: If the developer team feels that the defect is not a genuine defect, they will change the defect’s state to “rejected.”

Deferred: If the defect is not of a high priority and is expected to get fixed in the next release, then the defect is deferred.

Difference between Defect /Bug/Error/Failure

Defect : Problem which is identified on Developer machine at development phase

Bug : Problem which is identified on Testers machine at testing phase

Error: Problem which is related to coding.

Failure : Problem which is identified By end users at production phase

JIRA Tool

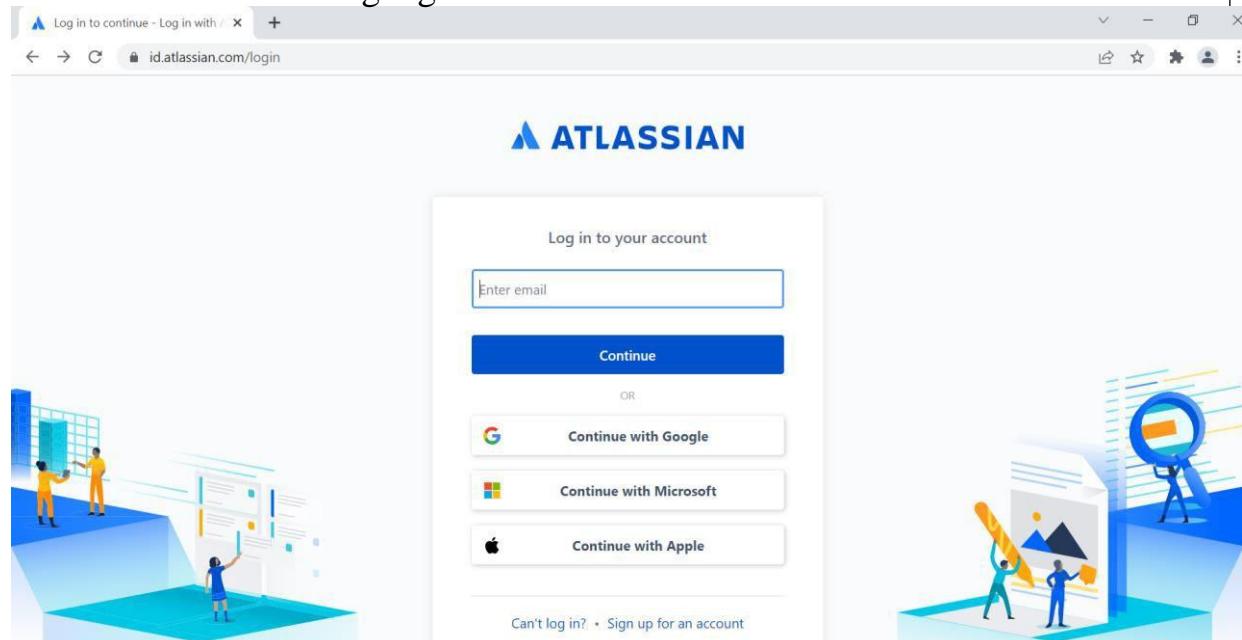
In JIRA, Test case, Bug, Epic and Story will be called as Issues.

To create a JIRA issue, you need the Create Issue project permission for the issue's relevant project.

Steps to set up Jira

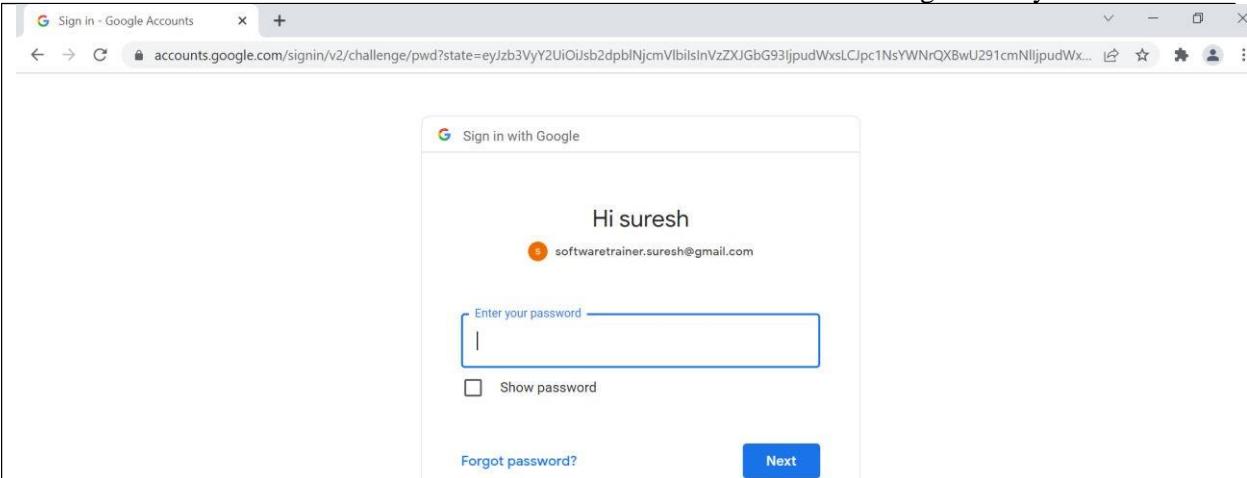
→Launch browser and enter url as - <https://id.atlassian.com/login>

Click on continue with google



→Provide your valid Gmail details

Manual Testing Notes by Thiru



→ Enter password

→ Click on Create Your account button

A screenshot of the Atlassian start page. The header includes the Atlassian logo and links for "Home" and "Notifications". Below the header are sections for "Helpful links" and "Our products". The "Helpful links" section contains icons for Account settings, Atlassian Support, Atlassian Community, Self-managed licensing, Atlassian Documentation, and Atlassian.com. The "Our products" section contains icons for Jira Software, Confluence, Jira Service Management, Bitbucket, and Statuspage, each with a "TRY" button below it.

→ Click on Jira Software Product

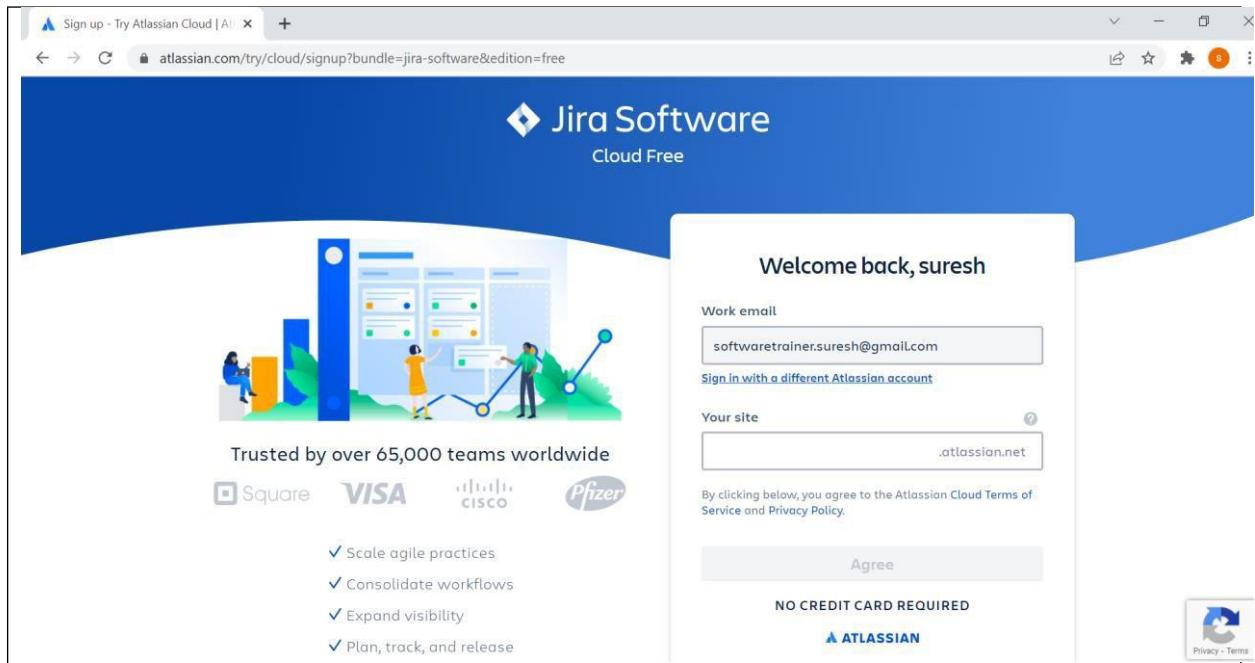
A screenshot of the Jira Software product page. The header features the Atlassian logo and navigation links for "Products", "For teams", "Support", "Buy now", and a search bar. Below the header is a promotional banner for "Team '22" with a "Sign up now" button. The main content area features the text "The #1 software development tool used by agile teams" and a "Get it free" button. To the right is a colorful illustration of people working with Jira boards and charts.

→ Click on Get it Free button

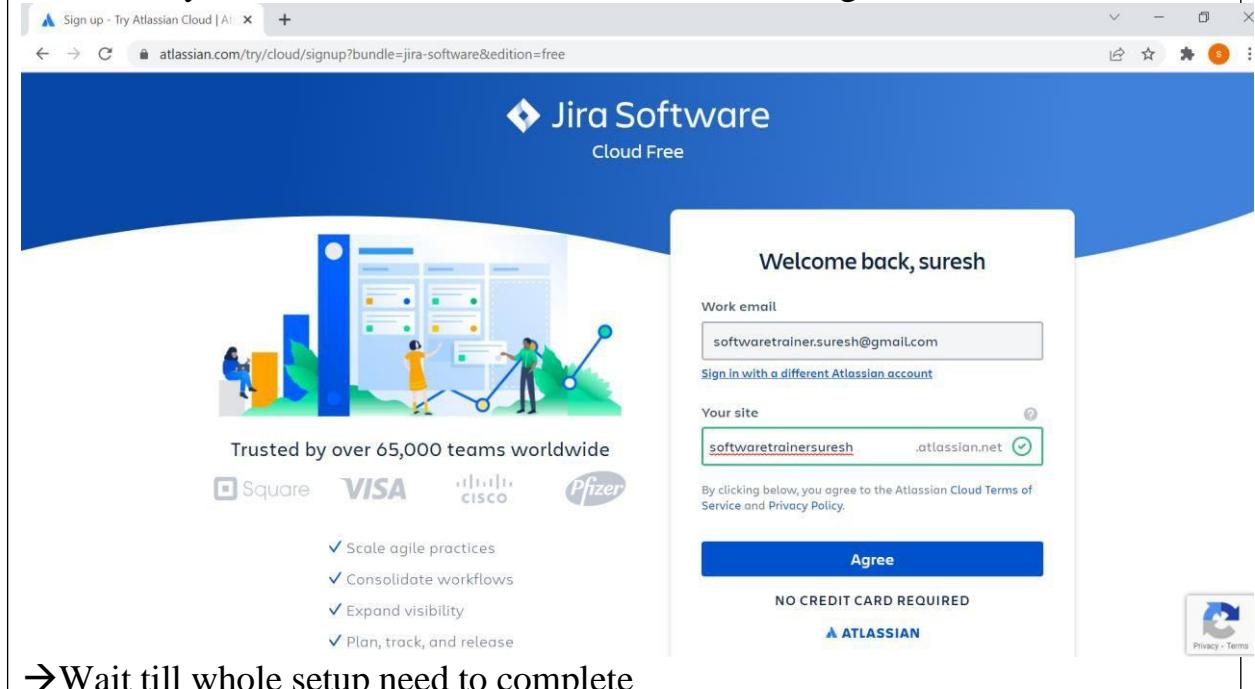
The screenshot shows a web browser window for 'Free - Jira Software | Atlassian' at atlassian.com/software/jira/free. The page features the Atlassian logo and a large heading: 'Our cloud products work even better together'. A section titled 'Get another one for free' lists three benefits: 'Supports up to 10 users or 3 agents', 'Includes 2 GB of storage', and 'Offers Community support'. Below this, a list of 'EACH PRODUCT ON A FREE PLAN:' includes: 'Supports up to 10 users or 3 agents', 'Includes 2 GB of storage', 'Offers Community support', and 'Is always free, no credit card needed'. On the right, under 'YOU SELECTED', is 'Jira Software Project and Issue tracking' with a blue diamond icon. Under 'SELECT A SECOND PRODUCT', there are two options: 'Confluence Document collaboration' and 'Jira Service Management High-velocity ITSM'. Each option has a 'Select' button. At the bottom left is a 'Larger team?' link with a user icon, and at the bottom right is a 'Next' button.

→ Click on Next button

Manual Testing Notes by Thiru

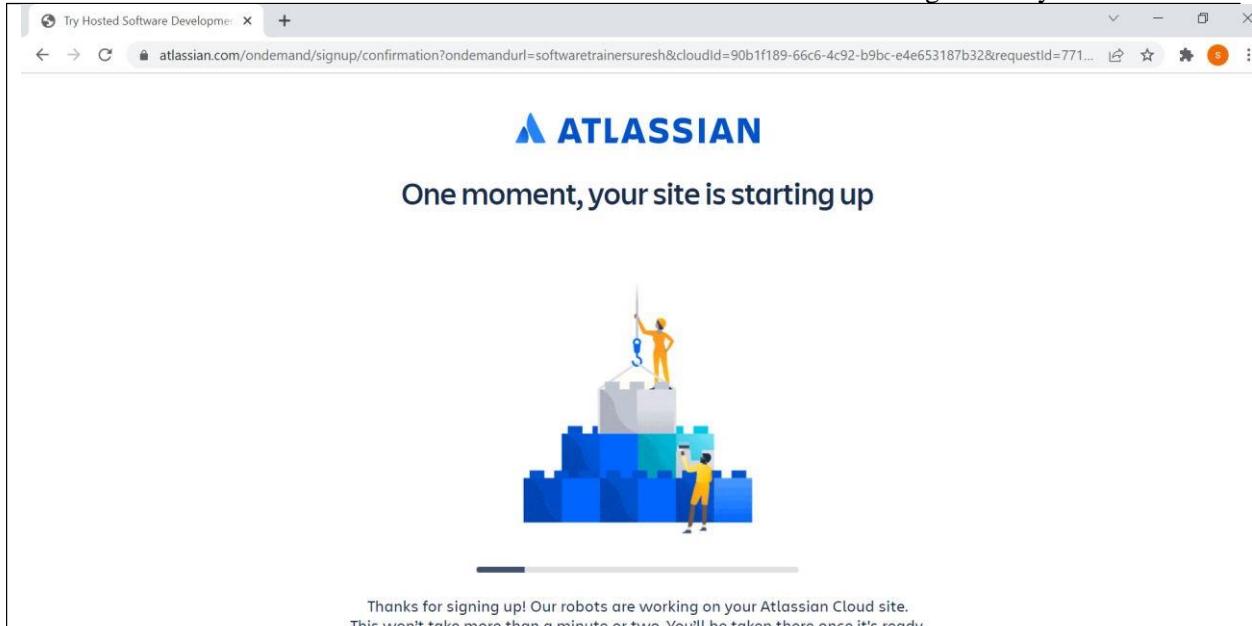


→ Enter any name in Your Site textbox and click on Agree button

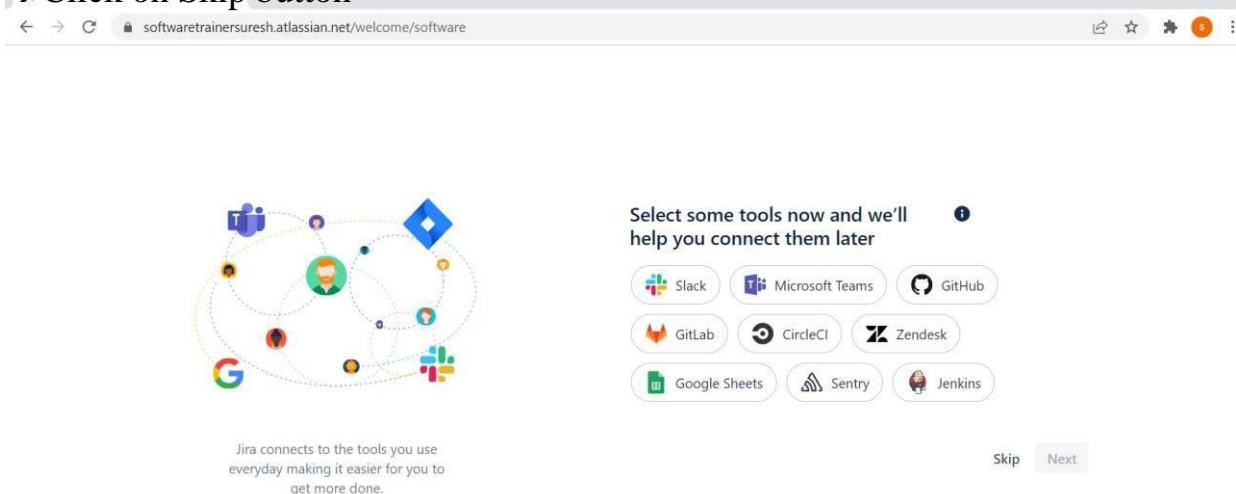


→ Wait till whole setup need to complete

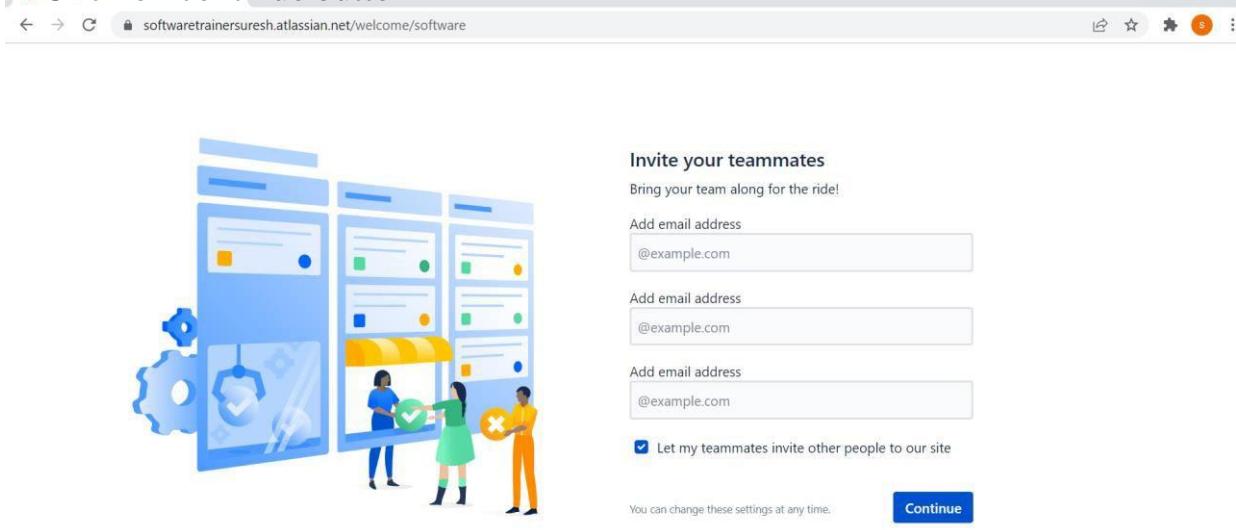
Manual Testing Notes by Thiru



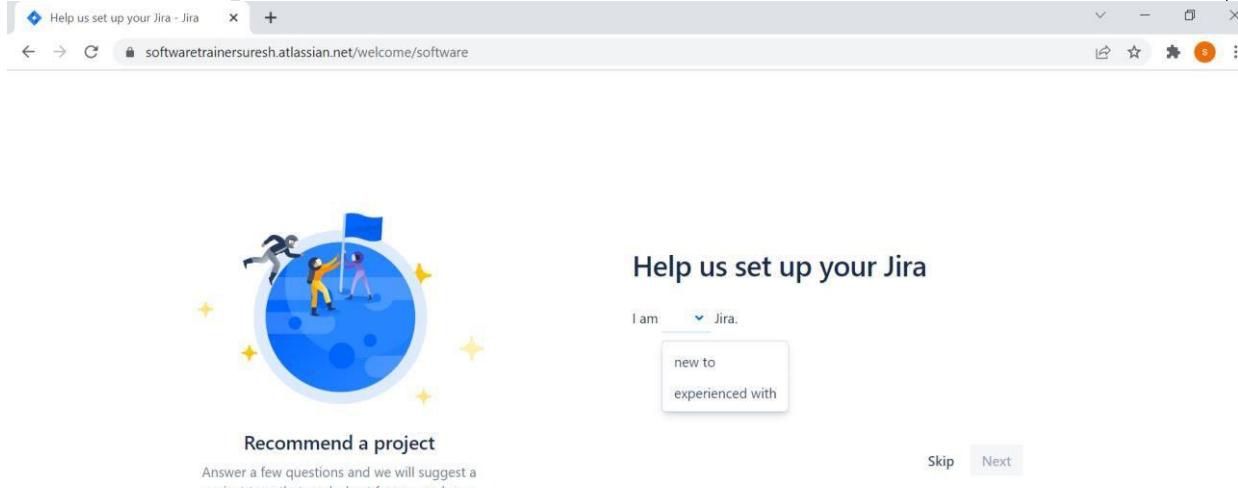
→ Click on Skip button



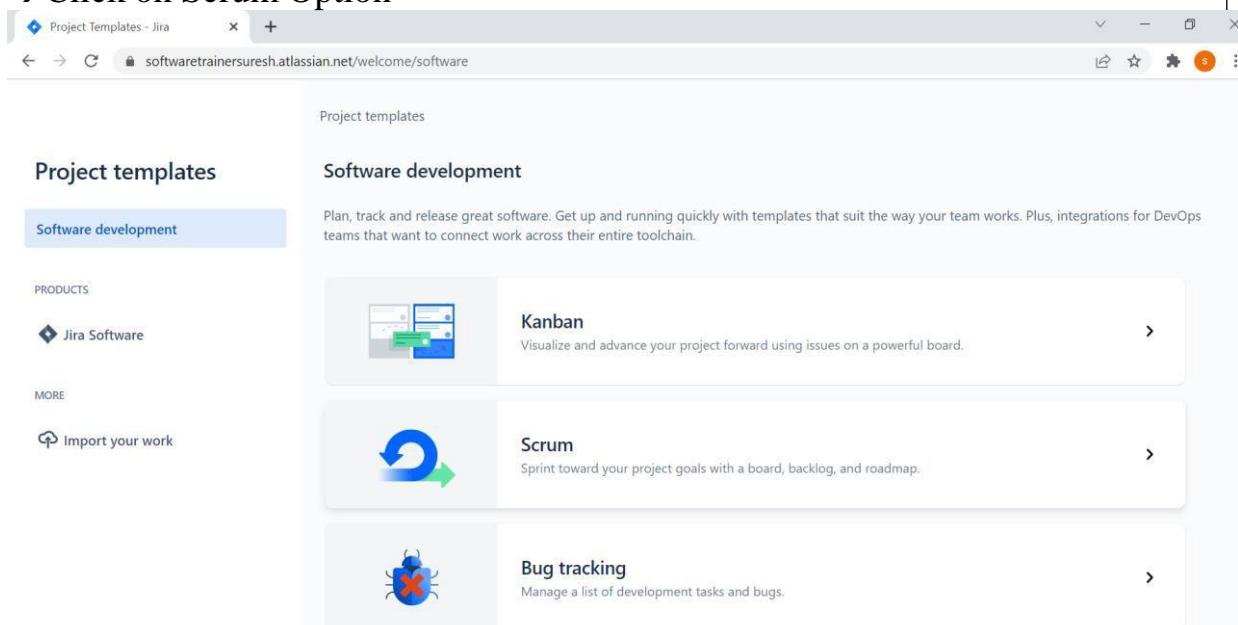
→ Click on continue button



→Click on Skip button



→Click on Scrum Option



→Click on Use Template

Manual Testing Notes by Thiru

The screenshot shows the Jira Project Templates interface for Software development. On the left, there's a sidebar with 'Project templates' and a 'Software development' tab selected. The main area displays the 'Scrum' template. It includes a brief description: 'The Scrum template helps teams work together using sprints to break down large, complex projects into bite-sized pieces of value. Encourage your team to learn through incremental delivery, self-organize while working on a problem, and regularly reflect on their wins and losses to continuously improve.' Below this is a visual representation of a backlog board with three items. To the right, there's a 'Plan upcoming work in a backlog' section with a link to 'Learn more about the backlog'. Further right are sections for 'PRODUCT' (Jira Software), 'RECOMMENDED FOR' (Teams that deliver work on a regular cadence, DevOps teams), and 'ISSUE TYPES' (Epic). A 'Use template' button is at the top right.

→ Click on Team Managed Project button

The screenshot shows the 'Choose a project type - Jira' page. It has two main sections: '1 Project template' and '2 Choose a project type'. The '1 Project template' section shows the Scrum template with its description and a 'Change template' button. The '2 Choose a project type' section has a warning message: '⚠ You'll need to create a new project if you decide to switch project types later.' Below this, there are two options: 'Team-managed' and 'Company-managed'. Each option has a sub-section: 'Set up and maintained by your team.' for Team-managed and 'Set up and maintained by your Jira admins' for Company-managed. At the bottom are two large buttons: 'Select a team-managed project' (purple) and 'Select a company-managed project' (blue).

→ Enter project name & Click on Create Project

Manual Testing Notes by Thiru

Add project details

You can change these details anytime in your project settings.

Name * Adactin001

Key * AD

Template Scrum

Type Team-managed

Back Create project

→ Jira Project Home Page will be displayed

Adactin001 - Roadmap - Jira

Projects / Adactin001

Roadmap

Give feedback Share Export

Epic

+ Create Epic

Today Weeks Months Quarters

Quickstart

Create a project

Map out your project goals

Turn those big goals into achievable outcomes so your team can get more done, more often. The roadmap helps you plan, track, and visualize your project goals, which we call epics. Your epics are then broken down into small, actionable chunks of work.

Show me View tutorial

Dismiss Quickstart

→ Navigate to Apps Main menu & Click on Manage Apps Submenu

Manual Testing Notes by Thiru

Jira Software - Adactin001 - Roadmap - Jira

Projects / Adactin001

Roadmap

Give feed!

RECOMMENDED FOR YOUR TEAM

Ship faster with marketplace apps that integrate your team's tools with Jira.

- Slack
- Microsoft Teams
- Figma (Design)

Explore more apps

Manage your apps

View app requests

Quickstart

- Create a project
- Map out your project goals

Turn those big goals into achievable outcomes so your team can get more done, more often. The roadmap helps you plan, track, and visualize your project goals, which we call epics. Your epics are then broken down into small, actionable chunks of work.

Show me View tutorial

Dismiss Quickstart

Planning

- Roadmap
- Backlog
- Board

Development

- Code
- Project pages
- Add shortcut
- Project settings

You're in a team-managed project

Learn more

<https://softwaretrainersuresh.atlassian.net/plugins/servlet/upm>

→ Click on Find New apps

Jira Software - Adactin001 - Marketplace discover - Jira

Discover apps and integrations for Jira

Search for apps

Top trending

Free for all teams

More Filters

Categories

Tempo Timesheets - Time Tracking & Reports

The #1 time management product in the Atlassian ecosystem since 2010

Dashboard gadgets, Integrations, Reports, Time trac...

Figma for Jira

Stay on track with live Figma embeds in Jira Design tools

13.4k installs

Zephyr Scale - Test Management for Jira

A scalable, performant test management solution inside Jira with advanced test planning, reporting, and reusability features

Custom fields, Dashboard gadgets, Reports, Testing ...

10.8k installs

STAFF PICK

STAFF PICK

https://softwaretrainersuresh.atlassian.net/jira/marketplace/discover?s=com.atlassian.jira.emcee__discover

→ Search for zephyr scale App – Enter zephyr scale in textbox & Click on ENTER button from keyboard

Manual Testing Notes by Thiru

The screenshot shows the Jira Marketplace search results for the term "zephyr scale". The search bar at the top contains "zephyr scale". Below it, there are three app cards displayed as "STAFF PICK".

- ScriptRunner for Jira**: The leading automation and customization app for Jira.
- Tempo Timesheets - Time Tracking & Reports**: The #1 time management product in the Atlassian ecosystem since 2010.
- Jira Misc Workflow Extensions (JMWE)**: Most powerful, ALL-IN-ONE app for automating Jira workflows.

Each card includes a star rating and the number of installs.

→Click on the App of zephyr scale – Test management for JIRA

The screenshot shows the Jira Marketplace search results for the term "zephyr scale". The search bar at the top contains "zephyr scale". Below it, there is one result displayed as a "STAFF PICK".

- Zephyr Scale - Test Management for Jira**: A scalable, performant test management solution inside Jira with advanced test planning, reporting, and reusability features.

Each card includes a star rating and the number of installs.

→Click on Try it Free button

Manual Testing Notes by Thiru

The screenshot shows the Jira Software interface with the 'Apps' tab selected. On the left sidebar, there are links for 'Find new apps', 'Manage apps', 'App requests', 'Promotions', and 'OAuth credentials'. The main content area displays the 'Zephyr Scale - Test Management for Jira' app by SmartBear. It has a 4-star rating from 240 reviews and is labeled 'CLOUD FORTIFIED'. A 'TRY IT FREE' button is visible, along with an estimated cost of '\$0 / month'. Below the app details, there are tabs for 'Overview' (which is selected) and 'Support'. To the right, there is a promotional image for the app.

→ Click on Start free trial button

The screenshot shows a modal dialog titled 'Add to Jira' for the Zephyr Scale app. The dialog contains the same information as the previous screenshot: the app's name, developer, rating, and a 'TRY FREE' button. It also includes a list of actions the app will perform, such as acting on behalf of users and administering Jira projects. At the bottom of the dialog, there are 'View app details' and 'Start free trial' buttons, with 'Cancel' being the default close button.

→ Your will get success message

Manual Testing Notes by Thiru

The screenshot shows the Jira Marketplace page for the 'Zephyr Scale - Test Management for Jira' app. The app icon is a blue square with a white stylized 'Z'. The title is 'Zephyr Scale - Test Management for Jira' by SmartBear. It has a 4-star rating from 240 reviews and is labeled 'CLOUD FORTIFIED'. Below the title, there are two tabs: 'Overview' (selected) and 'Support'. A success message box is visible on the left: 'Success: Zephyr Scale - Test Management for Jira was added.' A 'Manage app' button is also present. To the right, there is a promotional banner for Zephyr Scale with the text 'A scalable, performant test management solution inside Jira with advanced testing, and more' and a video thumbnail.

→ Navigate to Projects Main menu and select the Project which is previously created

This screenshot is similar to the previous one but shows the 'Projects' menu item in the top navigation bar being selected. A dropdown menu appears, listing 'RECENT' projects: 'Adactin001 (AD Software project)'. Other options include 'View all projects' and 'Create project'. The rest of the page content is identical to the first screenshot, showing the Zephyr Scale app details and a success message.

→ Click on Zephyr Scale Option

Manual Testing Notes by Thiru

The screenshot shows the Jira Software interface for the project 'Adactin001'. The left sidebar has 'Zephyr Scale' selected. The main area displays the 'AD board' with columns 'TO DO' and 'IN PROGRESS'. A large blue circular arrow icon indicates no work has been started. A message says 'You haven't started a sprint yet.' A 'Quickstart' sidebar on the right provides options like 'Create a project', 'Customize your board', and 'Set up your Jira team-managed board'.

→Click on Project Setting Page

The screenshot shows the Jira Software interface for the project 'Adactin001'. The sidebar has 'Project settings' selected. The main area shows the 'Zephyr Scale' settings page with a message: 'Zephyr Scale is currently disabled for the project Adactin001 (AD). You can enable it on the Project Settings page.' A 'Quickstart' sidebar on the right provides options like 'Create a project', 'Create an issue', and 'Invite your teammates'.

→Click - on button

Manual Testing Notes by Thiru

The screenshot displays two stacked Jira Software windows. Both windows have the URL softwaretrainersuresh.atlassian.net/plugins/servlet/ac/com.kanoah.test-manager/admin-panel?project.key=AD&project.id=10000.

Top Window (Zephyr Scale Configuration):

- Left sidebar: Adactin001 Software project, PLANNING (Roadmap, Backlog, Board), DEVELOPMENT (Code), Project pages, Zephyr Scale, Add shortcut.
- Main content: **Zephyr Scale**. A message says "Enable Zephyr Scale to start managing test cases right now. You can disable it any time, without losing data." Below it is a switch button labeled "On" (highlighted) and "Off" (disabled). The status is "Zephyr Scale is disabled".

Bottom Window (Issue Types and Permissions):

- Left sidebar: Adactin001 Software project, PLANNING (Roadmap, Backlog, Board), DEVELOPMENT (Code), Project pages, Zephyr Scale, Add shortcut.
- Main content:
 - Issue Types**: A list of issue types with On/Off switches:
 - Story (On)
 - Task (On, checked)
 - Bug (On)
 - Epic (On)
 - Subtask (On)
 - Permissions**: A note: "If the permission system is enabled, the permissions configured below will be handled on this project. Otherwise, all users that can view this project will be able to perform any actions on Zephyr Scale." Below it is a switch button labeled "On" (highlighted) and "Off" (disabled). The status is "Permission system is disabled".

Annotation: A large arrow points to the "On" button under the "Permissions" section of the bottom window, with the text "→Click on button under permissions".

Manual Testing Notes by Thiru

The screenshot shows the Jira Software interface for the Adactin001 project. The left sidebar lists various project management sections like Roadmap, Backlog, and Board. The main content area is titled 'Permissions' and contains a note: 'If the permission system is enabled, the permissions configured below will be handled on this project. Otherwise, all users that can view this project will be able to perform any actions on Zephyr Scale.' A toggle switch is set to 'On'. Below this, under 'Test Cases', there is a table with columns: Permission, Users, Roles, and Groups. The 'Create' row shows 'suresh software trainer' in the 'Users' column and 'Administrator' in the 'Roles' column.

→Provide all the required permissions to add test cases and to execute test cases

This screenshot shows the same Jira Software interface as above, but with specific permissions assigned. In the 'Test Cases' section of the 'Permissions' table, the 'Create' row now includes the 'administrators' group in the 'Groups' column. The rest of the table rows (Edit, Archive, Manage Folders, Create Versions, Delete) remain empty.

**By this Setup will be completed

→Steps to Report Bugs in JIRA Tool

1. Login to JIRA tool

2. Click on **Create** button



3. Select the Required **Project** From the list.



4. Select the **Bug** option from Issue Type dropdown



Some issue types are unavailable due to incompatible field configuration and/or workflow associations.

5. Provide **Summary** of the bug

Validation for check-in Date is not working(Accepting Previous Date)

6. Provide the **Bug Description** which includes – Steps to re-produce , Expected Result and Actual Result

Style	B	I	U	A	³ A	C	U	E	E	⊕	⋮
<p>Steps To Re-produce:</p> <p>1.LaunchBrowser 2.Enter url 3.Enter username 4.Enter password 5.Click on Login button 6.Select all the required fields 7.Select Check-in Date as Previous Date 8.Click on Search button</p> <p>Expected Result :</p> <p>Error message should be displayed as "Check-In Date should be current Date or Future Date"</p> <p>Actual Result :</p> <p>Select Hotel page is Displayed</p>											

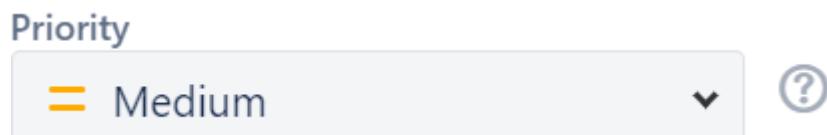
7. Provide Tester name

Reporter

suresh software trainer

Start typing to get a list of possible matches.

8. Provide Priority for the bug



9. Mention Environment details

Environment

Style	B	I	U	A	^A	C	U	≡	≡	⊕	⊗
Testing url: Browser: Chrome Browser Version: 91 Operating System : Windows Operating System Version : 10											

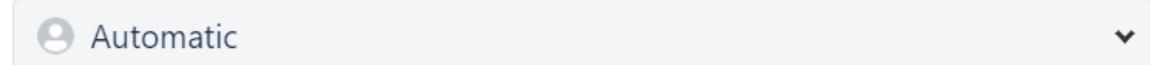
10. in case any screenshot attach the required files

Attachment



11. Select the developer name from Assignee dropdown

Assignee



[Assign to me](#)

12. Click on Create button



13. Bug will be created.

14. Refer Default dashboard to check created bugs

Manual Testing Notes by Thiru

The screenshot shows the Jira Default dashboard. On the left, under 'Projects', there are six items listed: 'Adacin BUG (ABCD) (TEAM)' (Lead: suresh software trainer), 'adactin (XQFB)' (Lead: suresh software trainer), 'Adactin001 (DCTN001)' (Lead: suresh software trainer), 'Adactin_009 (A0)' (Lead: suresh software trainer), 'Adactin_010 (AD)' (Lead: suresh software trainer). On the right, under 'Assigned to Me', there is a table with 13 rows of issues assigned to the user.

T	Key	Summary	P
■	A0-6	Validation for Check-in date is not working(Previous Date is Accepting)	=
■	A0-7	Total price issue	=
■	A0-8	Header information is display user name	<
■	A0-9	Total Price will be displayed 10\$ Extra	<
■	A0-10	"Booking Confirmation Room Type "	=
■	A0-11	"Booked Itinerary "	<
■	A0-13	"Booked Itinerary Children per Room does not display "	=

***Sprint Process

→ Click on Backlog Option

The screenshot shows the Jira Agile board for project 'Adactin001'. The left sidebar has 'Backlog' selected. The main area shows the 'Backlog' section for 'AD Sprint 1'. A 'Quickstart' panel is open on the right, providing instructions on how to use the backlog and creating issues.

Backlog

AD Sprint 1 (0 issues)

Plan your sprint
Drag issues from the Backlog section, or create new issues, to plan the work for this sprint. Select Start sprint when you're ready.

What needs to be done?

Quickstart

- Create a project
- Deliver more often with scrum
- Create an issue
 - Issue
 - Issue

Issues are individual pieces of work that you assign to teammates.
Issues can be tasks or stories.

Show me View issue tutorial

→ Click on Epic option & Enable Epic Panel

Manual Testing Notes by Thiru

The screenshot shows the Jira Software interface for the 'Adactin001' project. The left sidebar has 'Backlog' selected. The main area is titled 'Backlog' and shows a placeholder message: 'Your project has no Epic'. A tooltip for 'Epic panel' is open. On the right, there's a 'Quickstart' sidebar with options like 'Create a project', 'Deliver more often with scrum', and 'Create an issue'.

→ Click on Create Epic option & Provide all Epic details and Click on ENTER button from keyboard

The screenshot shows the Jira Software interface for the 'Adactin001' project. The left sidebar has 'Backlog' selected. The main area is titled 'Backlog' and shows a list of existing epics: 'Issues without epic', 'Epic1-NewUserRegistration', 'Epic2-Login Functionality', and 'Epic3-SearchHotelFunctionality'. A modal dialog is open, asking 'What will the Epic be called?'. On the right, there's a 'Quickstart' sidebar with options like 'Create a project', 'Deliver more often with scrum', and 'Create an issue'.

→ Select Required Epic and Create issue

Manual Testing Notes by Thiru

The screenshot shows the Jira Software interface for the 'Adactin001 - Agile board' project. The left sidebar has 'Backlog' selected. The main area is titled 'Backlog' and shows an 'Epic' section with 'Epic 1' expanded, revealing three issues: AD-4, AD-5, and AD-6, all under the 'EPIC2-LOGIN FUNCTIONALITY' label. A 'Start sprint' button is visible at the top right of the epic card.

→ Click on Start sprint & Provide all the sprint details and click on Start button

The screenshot shows the 'Start Sprint' dialog box overlaid on the Jira interface. The dialog box contains fields for 'Sprint name*' (Batch001_Sprint1), 'Duration*' (custom), 'Start date*' (20/02/2022 09:40), 'End date*' (06/03/2022 09:40), and 'Sprint goal' (Testing of Login Module). At the bottom are 'Start' and 'Cancel' buttons.

→ Sprint Home Page will be displayed as below

Manual Testing Notes by Thiru

The screenshot shows a Jira Software interface for a project named 'Batch001_Sprint1'. The 'Backlog' option is highlighted in the sidebar. A specific backlog item, 'Verify Login Functionality with all the combination', is selected, revealing its sub-tasks: 'AD-4', 'AD-5', and 'AD-6'. The interface includes a 'TO DO' section with 3 issues, an 'IN PROGRESS' section, and a 'DONE' section.

→ Click on BackLog main option and select required issue to create child issue(User Story)+

The screenshot shows the 'Backlog' screen for 'Batch001_Sprint1'. An epic named 'Epic2-Login Functionality' is selected. A child issue, 'AD-4 Verify Login Functionality with all...', is currently being edited. The edit screen displays the issue's title, description, and sub-tasks: 'AD-4', 'AD-5', and 'AD-6'. There is also a note indicating 'Your backlog is empty.'

→ Provide user stories and click on Create button

Manual Testing Notes by Thiru

The screenshot shows the Jira Software interface for the project 'Adactin001'. The left sidebar has 'Backlog' selected. The main area shows the 'Backlog' view with an 'Epic' section for 'Batch001_Sprint1'. This epic contains three issues: AD-4, AD-5, and AD-6, all under the 'EPIC2-LOGIN FUNCTIONALITY' epic. A modal window is open for issue AD-4, showing its details and a 'Zephyr Scale' section with a dropdown menu.

→ Click on Zephyr Scale option To Create Test Cases for required User Story

Manual Testing Notes by Thiru

Zephyr Scale - Jira

Jira Software Your work Projects Filters Dashboards People Apps Create

Adactin001 Software project

PLANNING Roadmap Backlog Board

DEVELOPMENT Code

Project pages Zephyr Scale Add shortcut Project settings

You're in a team-managed project Learn more Archived test cases

SMARTBEAR Zephyr Scale

TEST CASES + New ...

All test cases

No test cases. Create a test case or create a folder to get started.

Getting started with test cases

Getting Started Terms and Concepts Designing Test Cases Importing Test Cases Videos

→ Click on Create a test cases and write all the required test cases

softwaretrainersuresh.atlassian.net/projects/AD?selectedItem=com.atlassian.plugins.atlassian-connect-plugin:com.kanoah.test-manager_main-project

Jira Software Your work Projects Filters Dashboards People Apps Create

Adactin001 / Test Cases

Create Test Case

Back Save

Details Test Script Execution Traceability Attachments History

STEP TEST DATA EXPECTED RESULT

2 Enter url Click to type the test data Click to type the expected result

3 Enter username Click to type the test data Click to type the expected result

4 Enterpassword Click to type the test data Click to type the expected result

5 STEP TEST DATA EXPECTED RESULT

Click on Login button Click to type the test data

Search Hotel Page should be displayed

→ Click on Save button & Continue in creating all the required test cases.

***Test Execution Steps :

Click on Cycles button To create Test Cycle

→ Click on New Test Cycle option & Under Details tab provide Name, Owner, start date and end date

→ Click on Test Cases Tab

Manual Testing Notes by Thiru

→ Click on Add Test Cases option

Add Existing Test Cases

FullStack02

TEST CASES

All test cases [2]

FUL-T1 (1.0) FullStack02-Adactin (DRAFT)

FUL-T2 (1.0) FullStack02-Adactin (DRAFT)

Add others Add Close

→ Select Required Test Cases and click on Add button

Create Test Cycle

Details Test Cases 2 Traceability Attachments History

+ Add test cases Delete Assign environment Assign tester

Search... No estimated time

P	Key	V	Name	Assigned To	Environment
<input checked="" type="checkbox"/>	FUL-T1	1.0	FullStack02-Adactin	suresh software trainer	
<input checked="" type="checkbox"/>	FUL-T2	1.0	FullStack02-Adactin	suresh software trainer	

→ Click on Save button

Create Test Cycle

Details Test Cases 2 Traceability Attachments History

+ Add test cases Delete Assign environment Assign tester

Search... No estimated time

P	Key	V	Name	Assigned To	Environment
<input checked="" type="checkbox"/>	FUL-T1	1.0	FullStack02-Adactin	suresh software trainer	
<input checked="" type="checkbox"/>	FUL-T2	1.0	FullStack02-Adactin	suresh software trainer	

→ click on Back Link

Manual Testing Notes by Thiru

The screenshot shows the Zephyr Scale application integrated into a Jira project. The 'Cycles' tab is selected. There are two entries in the list:

- FUL-R1 FullStack02-Adactin-Execution: Progress 0%, Status NOT EXECUTED
- FUL-R2 TestExecution: Progress 0%, Status NOT EXECUTED

→ Click on Required check box and then click on Run button.

The screenshot shows the same Zephyr Scale interface after selecting the checkbox for FUL-R2. The 'Run' button is now visible and highlighted in blue.

1 test cycle is selected. [Clear selection](#)

Test Cycle	Progress	Status
FUL-R1 FullStack02-Adactin-Execution	0%	NOT EXECUTED
<input checked="" type="checkbox"/> FUL-R2 TestExecution	0%	NOT EXECUTED

Note: Based on Admin rights it will enable the run button.

→ In case of having admin rights, click on Run button and then will provide Actual Result and Test Cases status(PASS/FAIL).

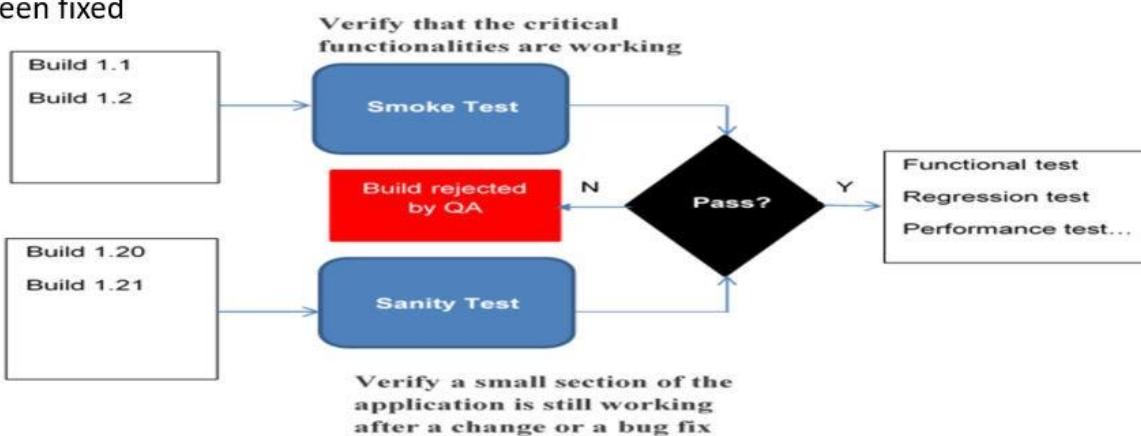
Types of Testing

- **Smoke Testing or BVT-Build Verification Testing or TAT – Testers Acceptance Testing**

It is first level of testing on any newly released build to check main functionality of the application.

- **Sanity Testing :**

Sanity Testing is done during the release phase to check for the main functionalities of the application. Sanity Testing is done to check the new functionality/bugs have been fixed



- **Re-Testing : Testing defects were fixed or not in the current build**

- **Regression Testing:** To check existing functionality is unaffected whenever the new change is added

Retesting

- To make sure the test cases which failed in last execution are working fine and the bugs are fixed
- Automation is not applicable in Retesting
- You can include the test cases which failed earlier/ functionality which failed in earlier built

Regression Testing

- Ensuring the bug fixes or enhancements a the module has not affected the other parts
- Automation plays a vital role in Regression
- You can include the test cases which passed earlier/ functionality which were working earlier

- **Static Testing:** Testing an application without performing any action.
Eg: GUI Testing , colors ,spelling ,alignment ... etc.

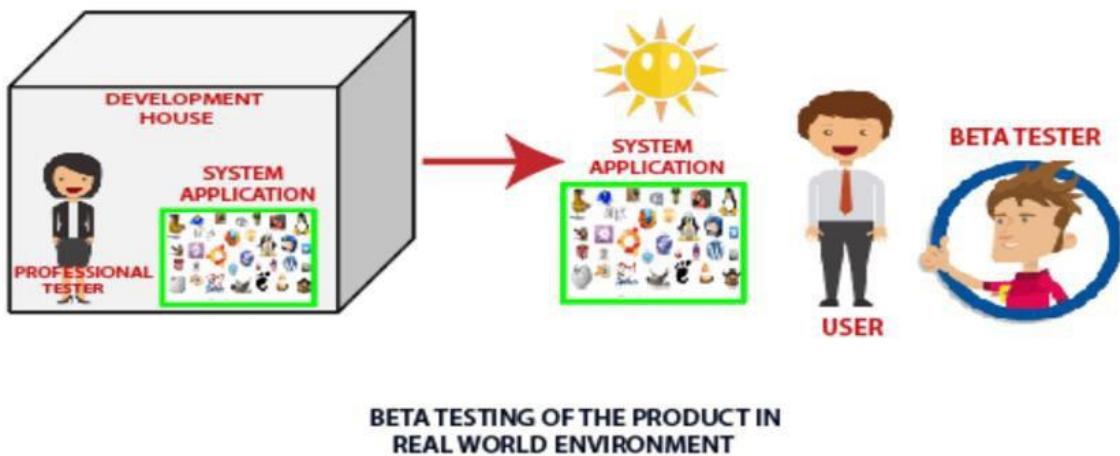
- **Dynamic Testing:** Testing an application by performing required actions.
Eg: Functionality Testing ,Textbox, button ... etc

- **Ad-hoc Testing:** Testing the application without any proper planning

Ad-hoc
Testing is
not
structured



- **Alpha Testing :**Final Testing on the application doing with in the development company
- **Beta Testing:** Testing doing in customer environment ,This testing will be done by the customer or third party test engineers.



Functionality Testing - Test for – all the links in web pages, database connection, forms used for submitting or getting information from the user in the web pages, Cookie testing etc.

Usability testing - Usability testing is nothing but the User-friendliness check. In Usability testing, the application flow is tested so that a new user can understand the application easily. Basically, system navigation is checked in Usability testing.



Compatibility testing - Compatibility testing is used to determine if your software is compatible with other elements of a system with which it should operate, e.g. Browsers, Operating Systems, or hardware.



Database Testing – Database connection and user entered Data in application is saving into respective database tables

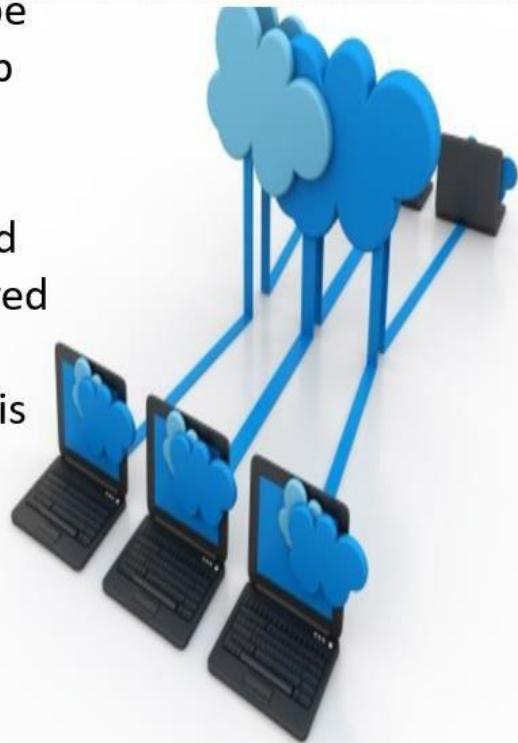


Interface testing-Three areas to be tested here are - Application, Web and Database Server

01. Application: Test requests are sent correctly to the Database and output at the client side is displayed correctly.

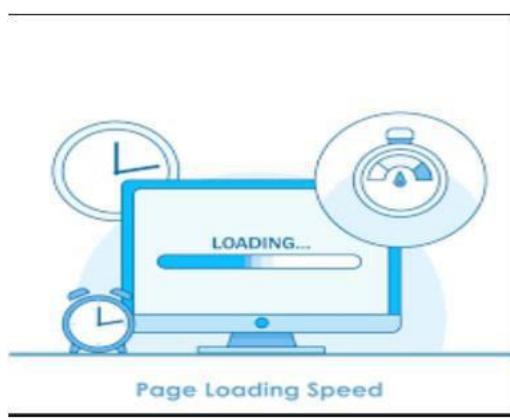
02. Web Server: Test Web server is handling all application requests without any service denial.

03. Database Server: Make sure queries sent to the database give expected results.



Performance testing – Testing page ,data, images load time

Security testing -Security Testing involves the test to identify any flaws and gaps from a security point of view.



Test Closure :

Test Closure is a document that is prepared prior to formally completing the testing process. This memo contains a report of test cases executed, passed, failed and number of defects found, fixed, re-tested, closed.

We will provide the path of all the required documents to client like test scenarios, test cases, test execution and bug reporting, based on this report it will decide to stop or continue the testing.

When to stop Testing?

This can be difficult to determine. Most modern software applications are so complex, and run in such an interdependent environment, that complete testing can never be done. Common factors in deciding when to stop are:

- Deadlines (release deadlines, testing deadlines, etc.)
- Test cases completed with certain percentage passed
- Test budget depleted
- Coverage of code/functionalities/requirements reaches a specified point
- Bug rate falls below a certain level
- Alpha & Beta testing period ends