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

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 identify by the another department called Software Test Engineer.

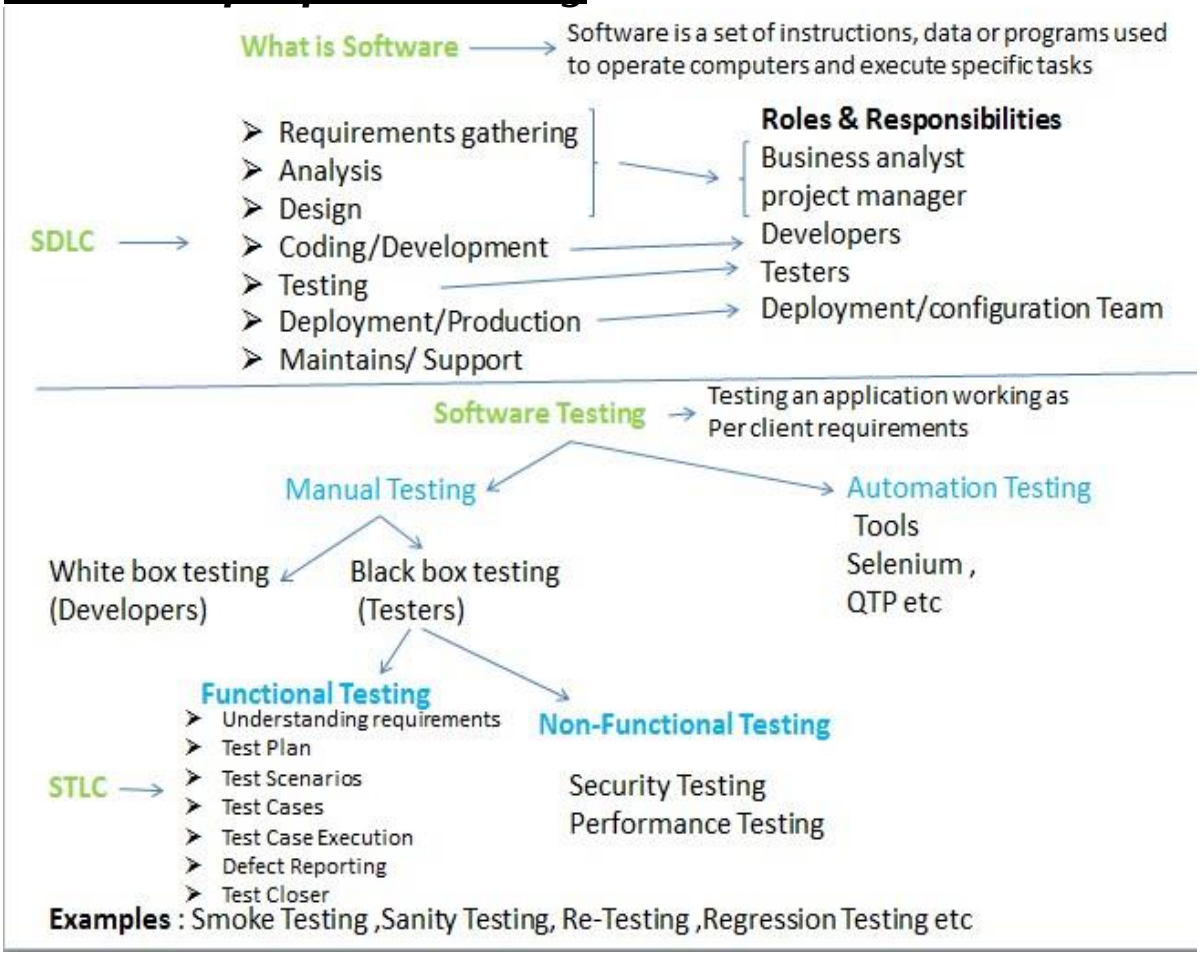
Benefits of Software Testing

"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
- Gaines 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

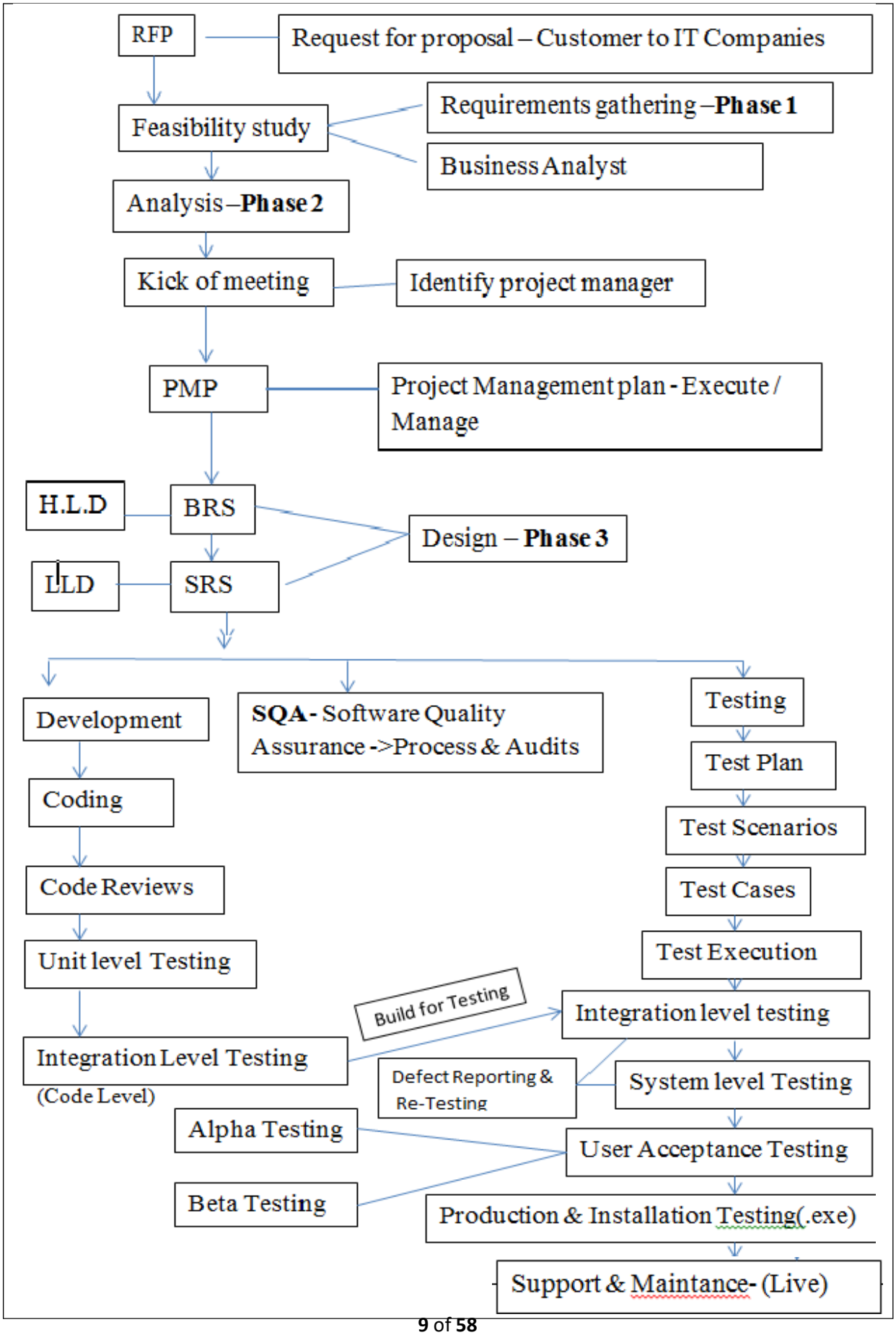
SDLC–Software Development Life Cycle

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 :

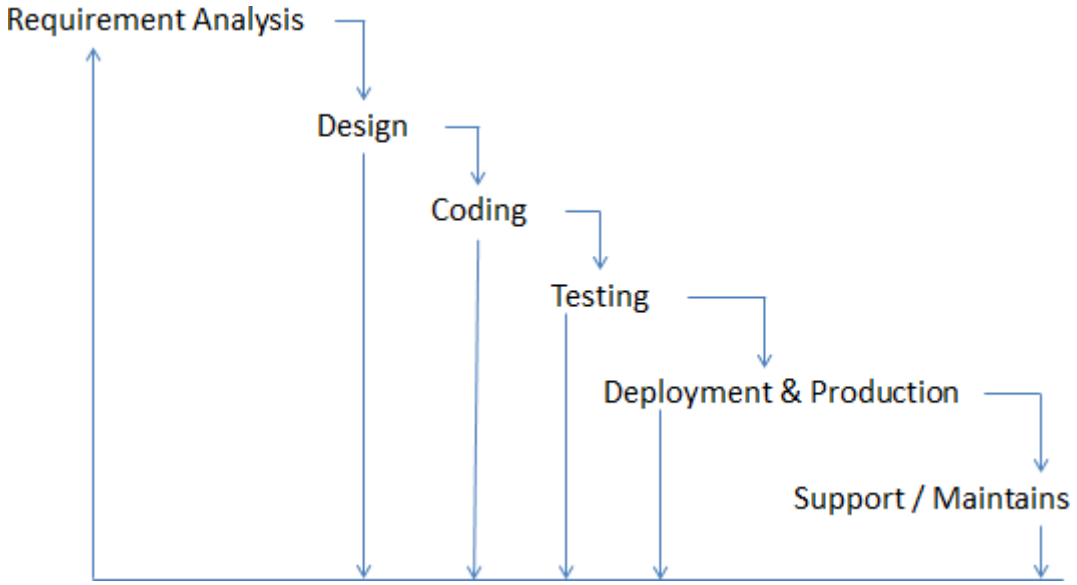

RFP
Request for proposal – Customer to IT Companies
Feasibility study
Requirements gathering –Phase 1
Business Analyst
Analysis –Phase 2
Kick of meeting
Identify project manager
PMP
Project Management plan - Execute / Manage
H.L.D
BRS
LLD
SRS
Design – Phase 3
Development
SQA- Software Quality Assurance ->Process & Audits
Testing
Coding
Test Plan
Code Reviews
Test Scenarios
Unit level Testing
Test Cases
Integration Level Testing
(Code Level)
Test Execution
Build for Testing
Integration level testing
Defect Reporting & Re-Testing
System level Testing
Alpha Testing
User Acceptance Testing
Beta Testing
Production & Installation Testing(.exe)
Support & Maintance- (Live)

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 :

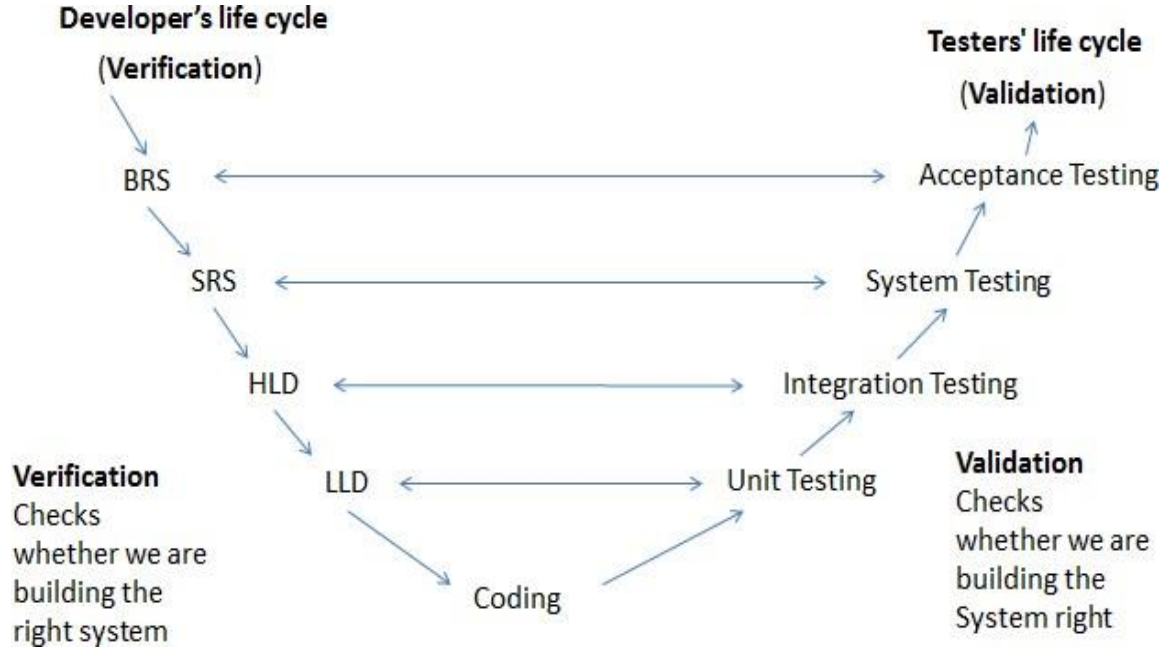
- Project under Control
- Pre-defined outputs at every phase
- Tracking changes is easy

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 :

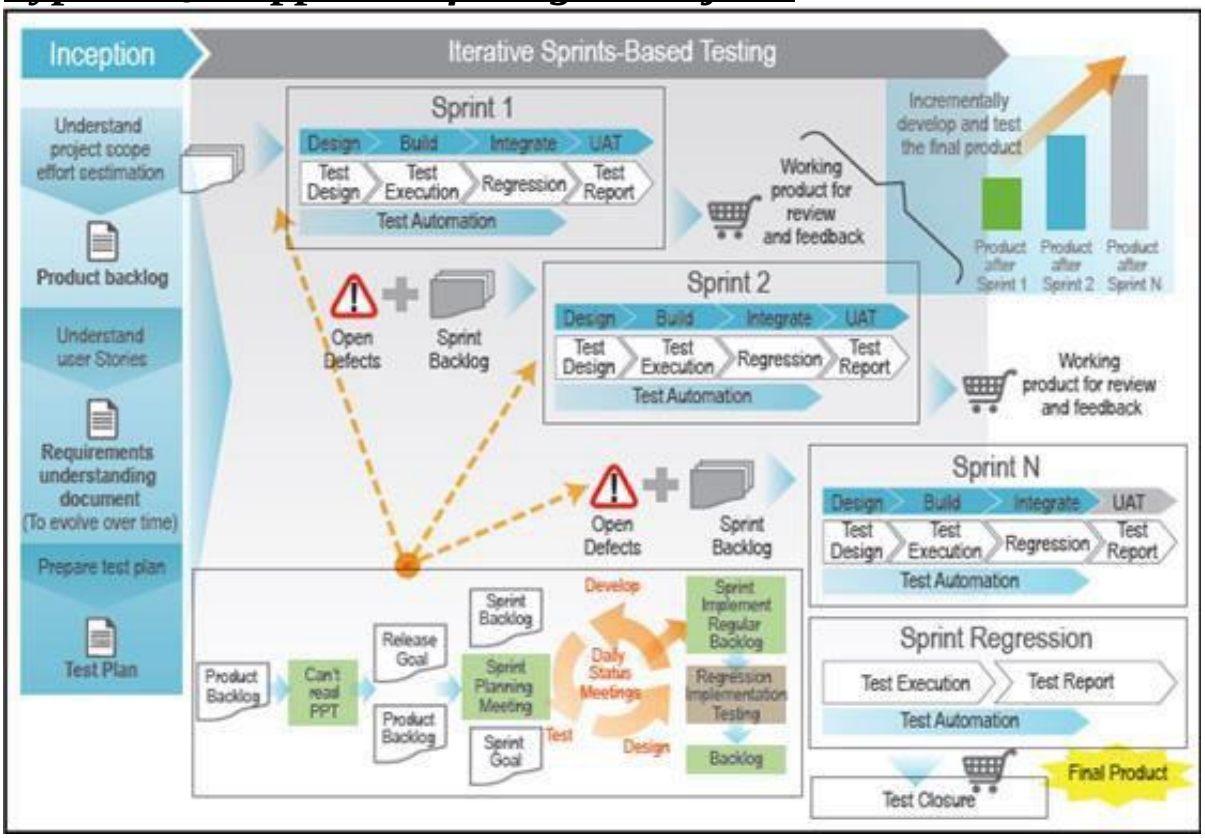
Verification	Validation
check whether we are developing the right product or not.	check whether the developed product is right.
Verification includes different methods like Inspections, Reviews, and Walkthroughs	In the validation testing, we can find those bugs, which are not caught in the verification 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:

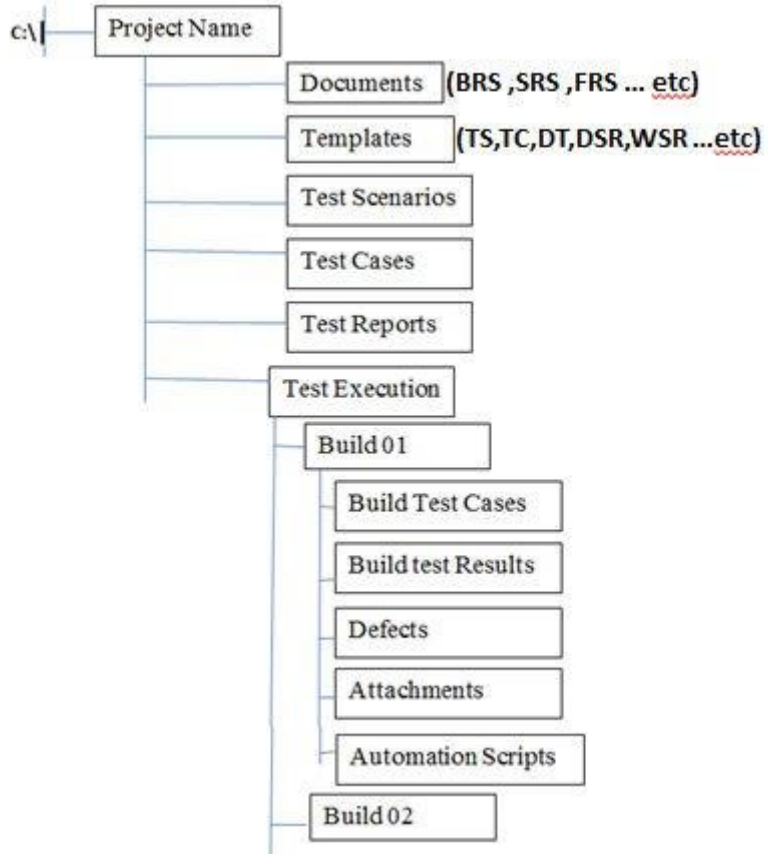
Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time,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

Company Logo			Proiect ID		Reviewed By	
			Project Name		Date Reviewed	
			Identified by		Approved By	
			Date Identified		Date Approved	
TS#	Req#	Main Functionality	TestScenario Description	Test Case Name/TC ID	Document ID/Reference	Comments

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 refering to design the Scenarios
Comments : In case of any spefic 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

Company Logo		Project History				Test Cases				
		Project ID				Total No. of TCs				
		Project Name				Passed				
						Failed				
		Test Case History				Not Executable				
		Created By		Date Created		Defects Reported				
		Reviewed By		Date Reviewed						
		Approved By		Date Approved						
		Test Information								
		Test Executed By		Date Executed						
		Version		Build						
TC #	TS#	Test Design/ Steps	Input Data	Expected Result	Actual Result	Pass	Fail	Not Executable	Comments	Defect Id

--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 : Tester name
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 matching consider as PASS
FAIL : Expected Result& Actual Result both are not matching consider as FAIL**Not Executable :** Test Case Which are not executed for that current Release **Comments :** In case of any spefic description need to be mention.
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	URL : Username : Password :	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	URL : Username : Password :	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 [ ] (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 [ ] **(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 Template

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

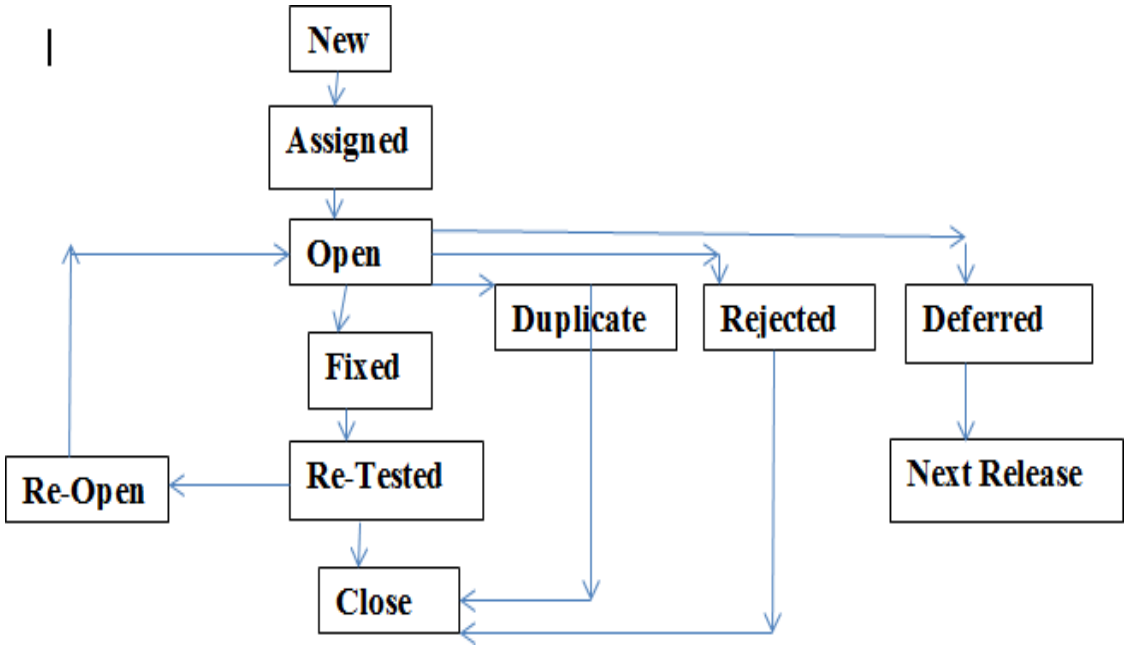
Sample Examples :
High Priority & High Severity: An error which occurs on the basic functionality of the application and will not allow the user to use the system. (Eg. A site maintaining the student details, on saving record if it, doesn't allow to save the record then this is high priority and high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

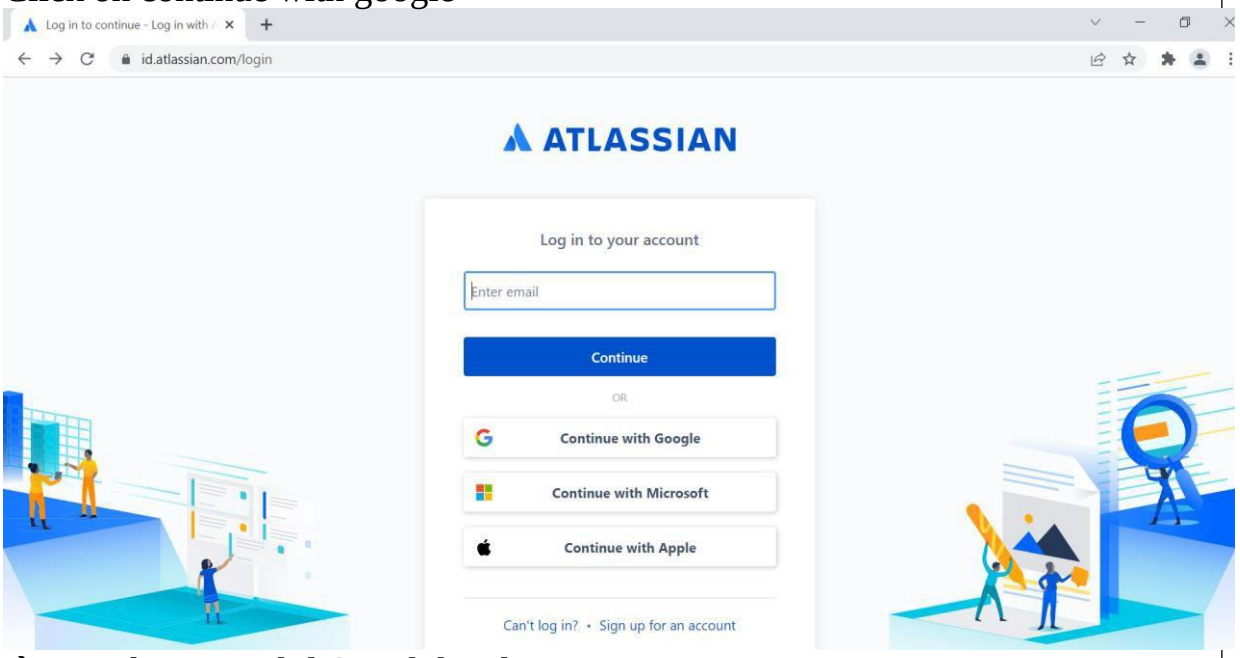
JIRA Tool

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

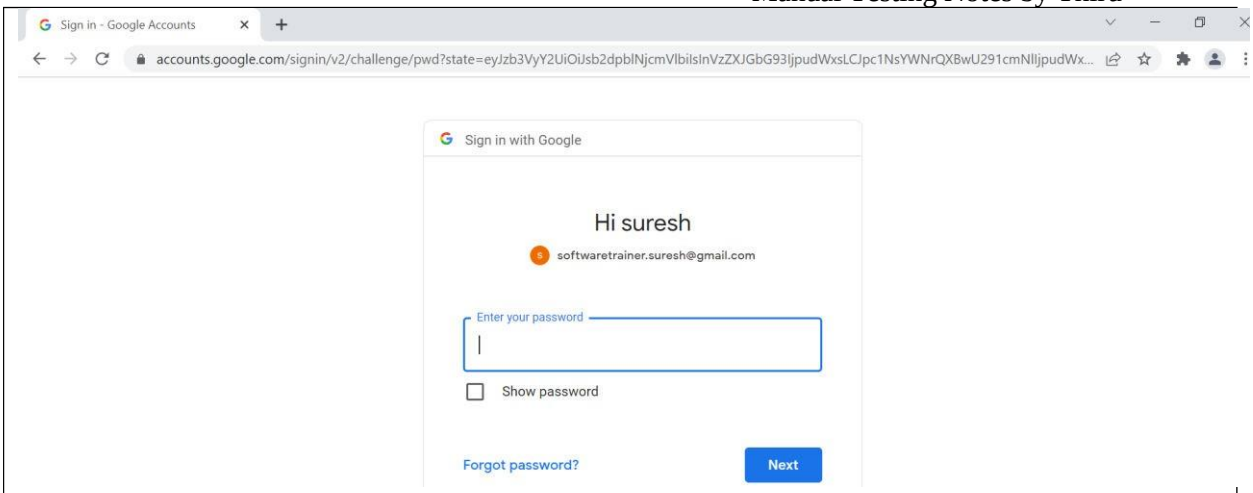
To create a JIRA issue, you need the Create Issue project permission for the issue's relevant project.

Steps to set up Jira

→Launch browser and enter url as - https://id.atlassian.com/login
Click on continue with google

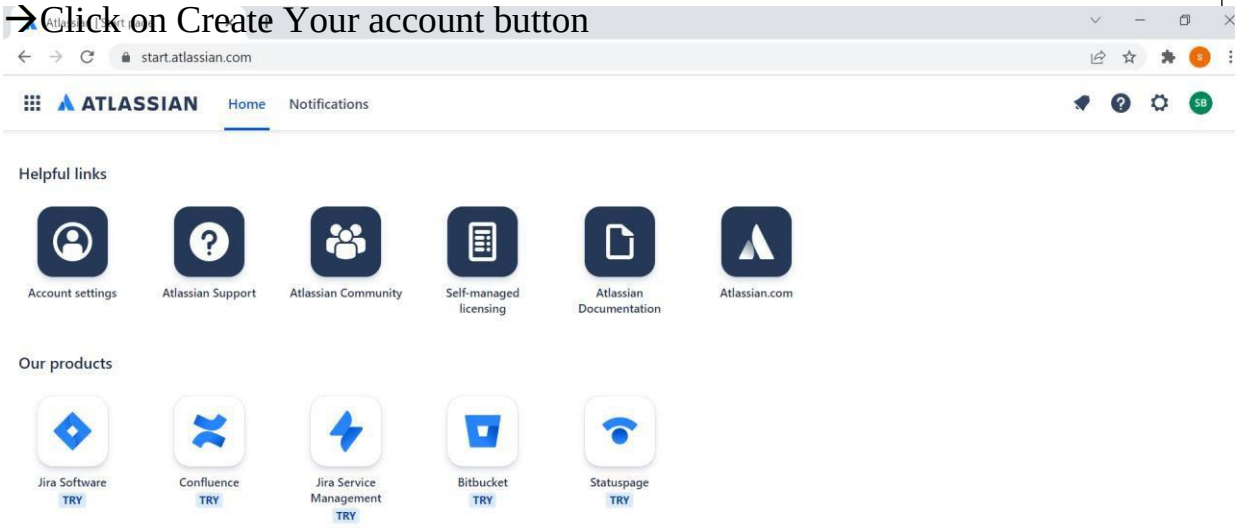


→Provide your valid Gmail details



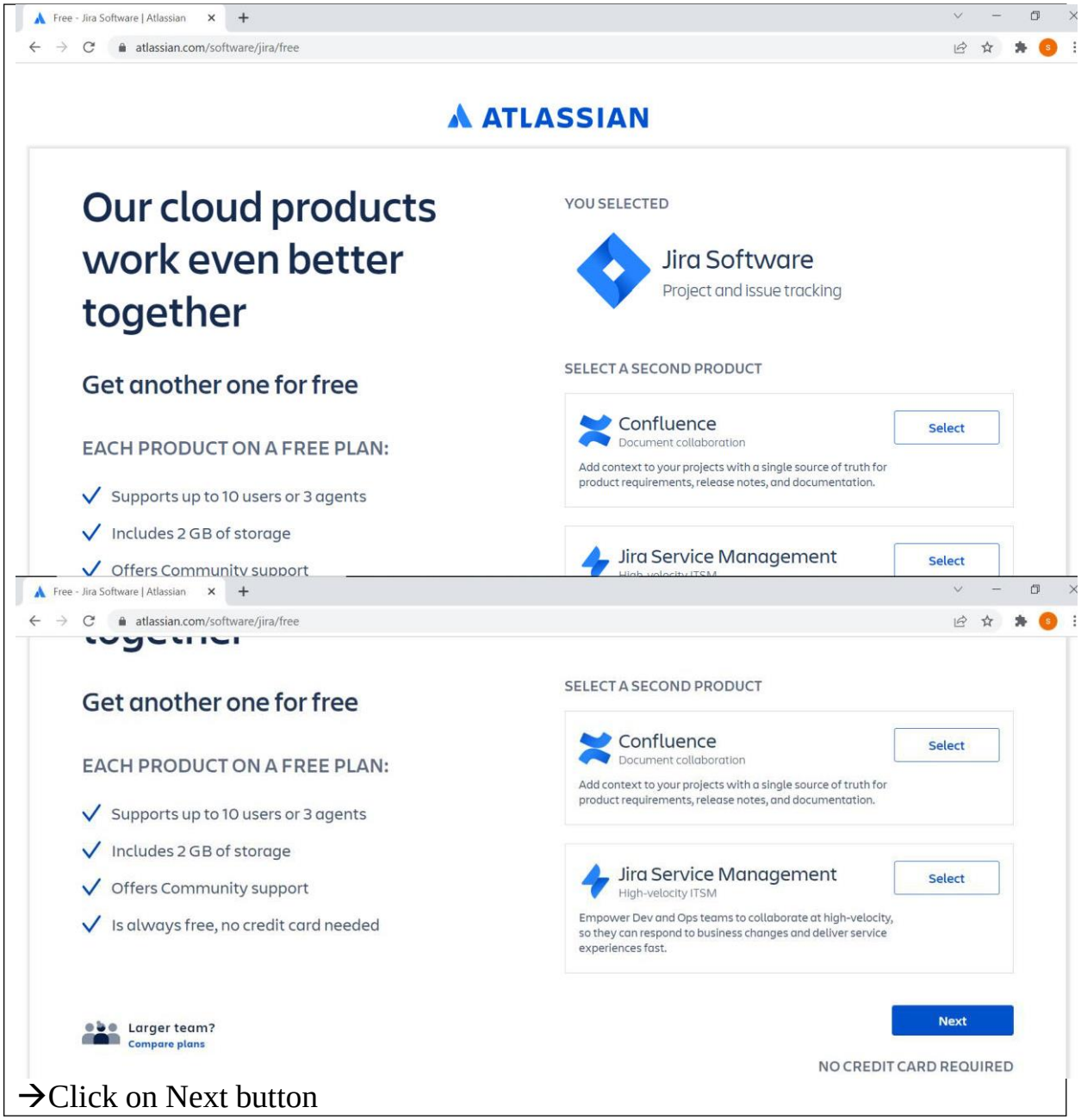
→Enter password

→Click on Create Your account button

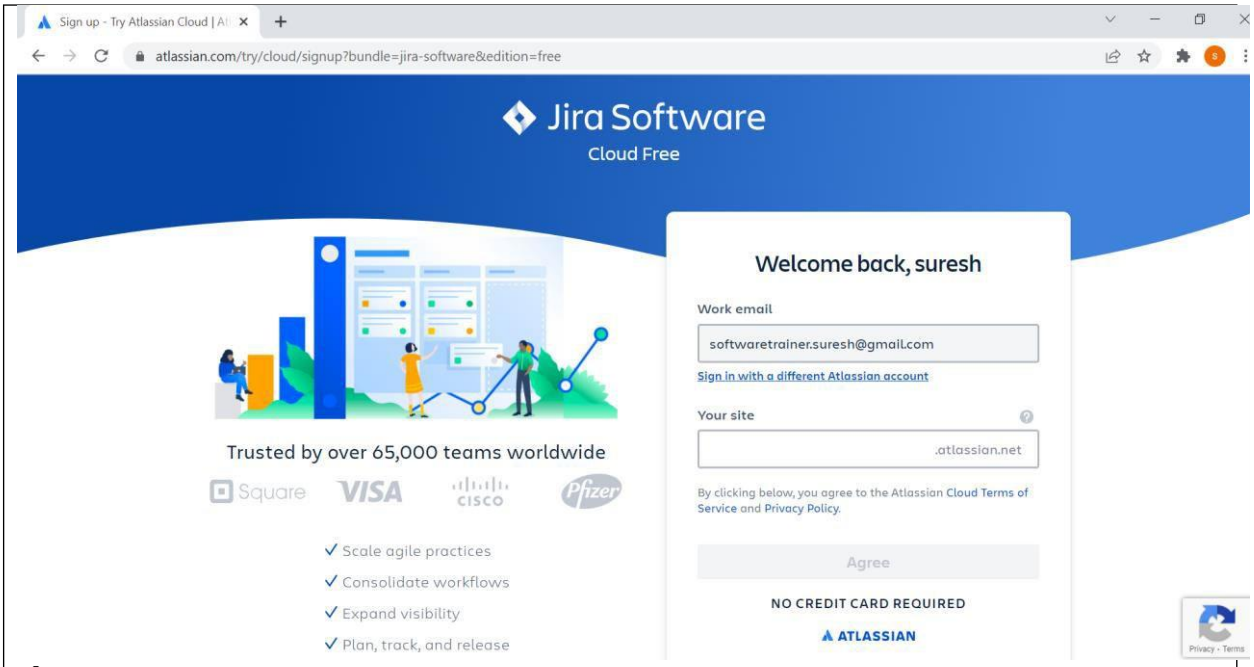


→Click on Jira Software Product

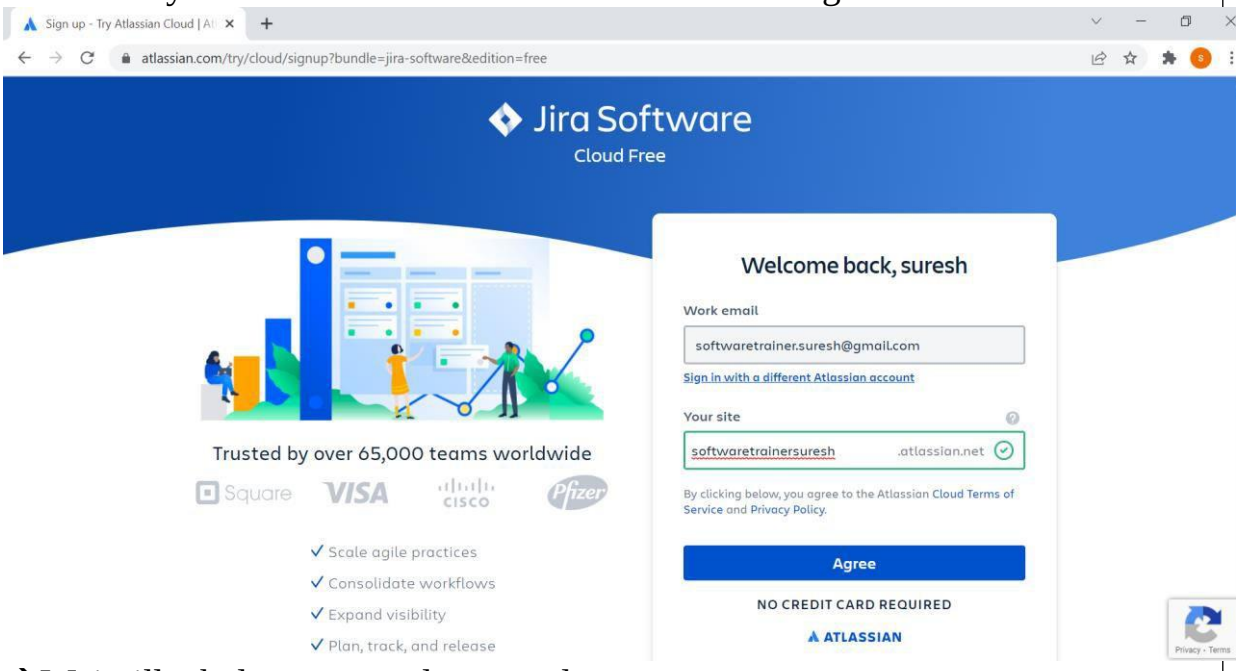
→Click on Get it Free button



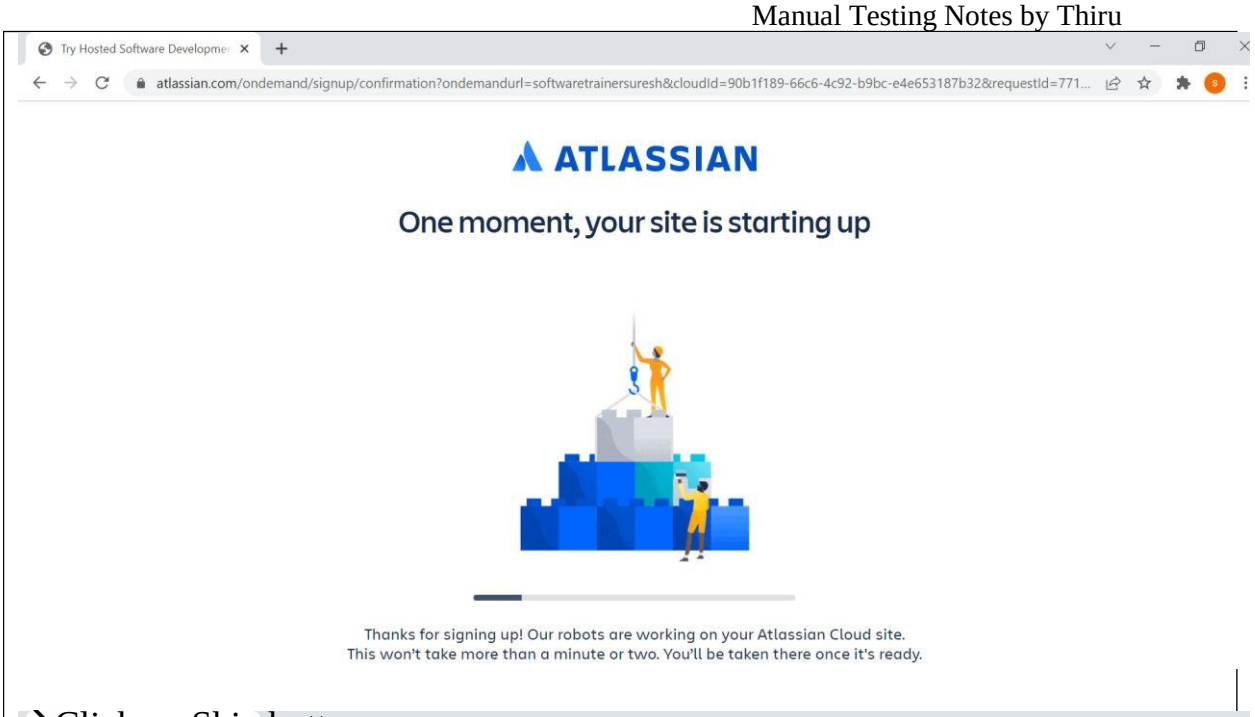
→Click on Next button



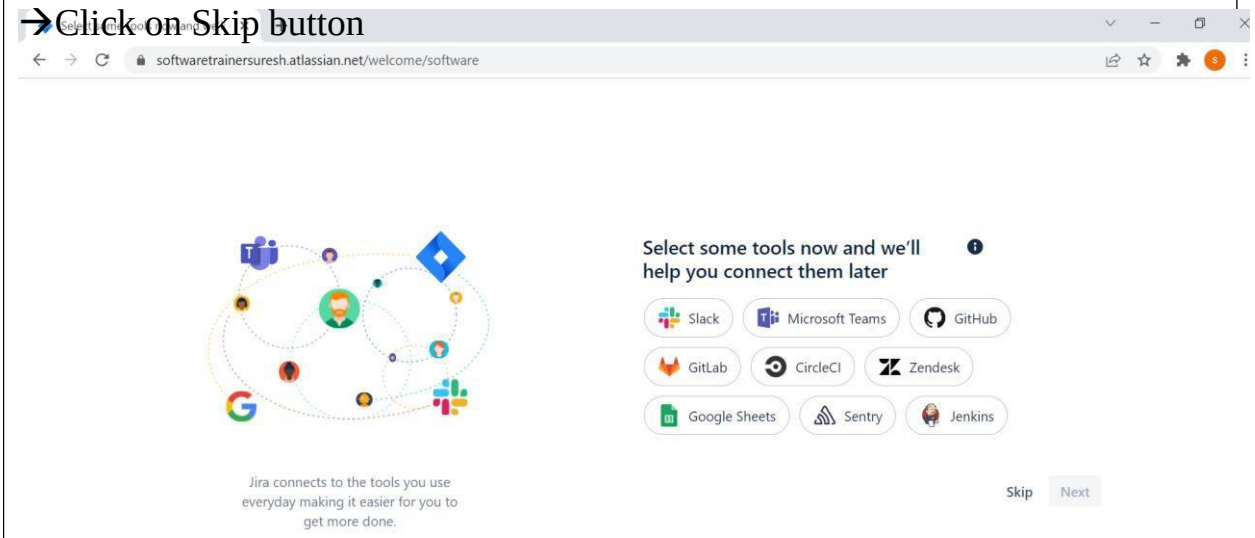
→Enter any name in **Your Site** textbox and click on Agree button



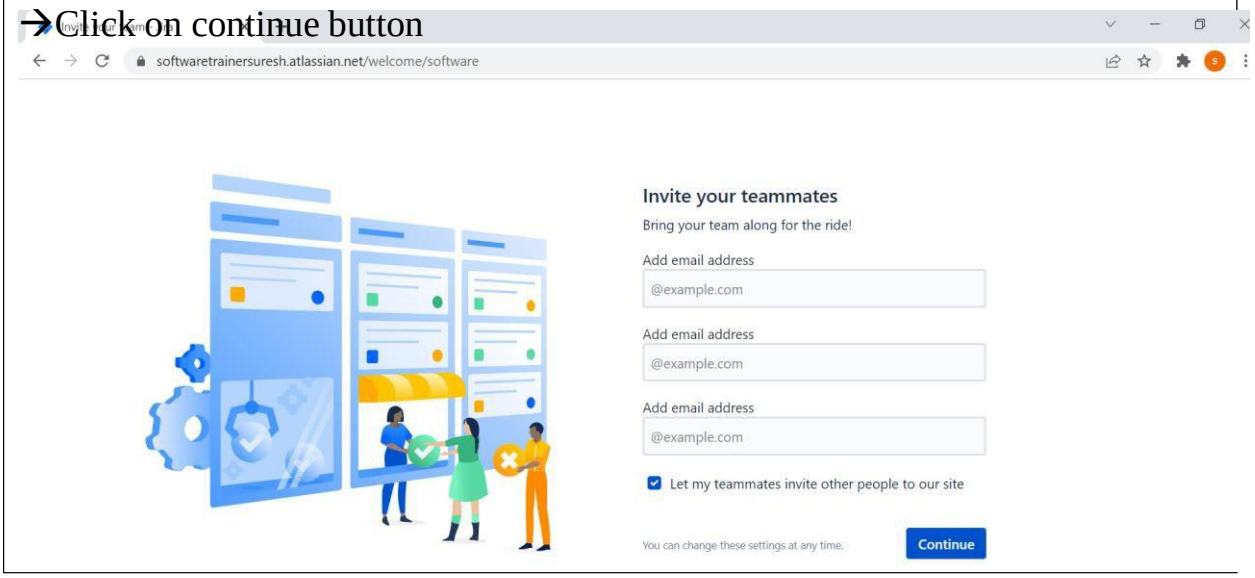
→Wait till whole setup need to complete



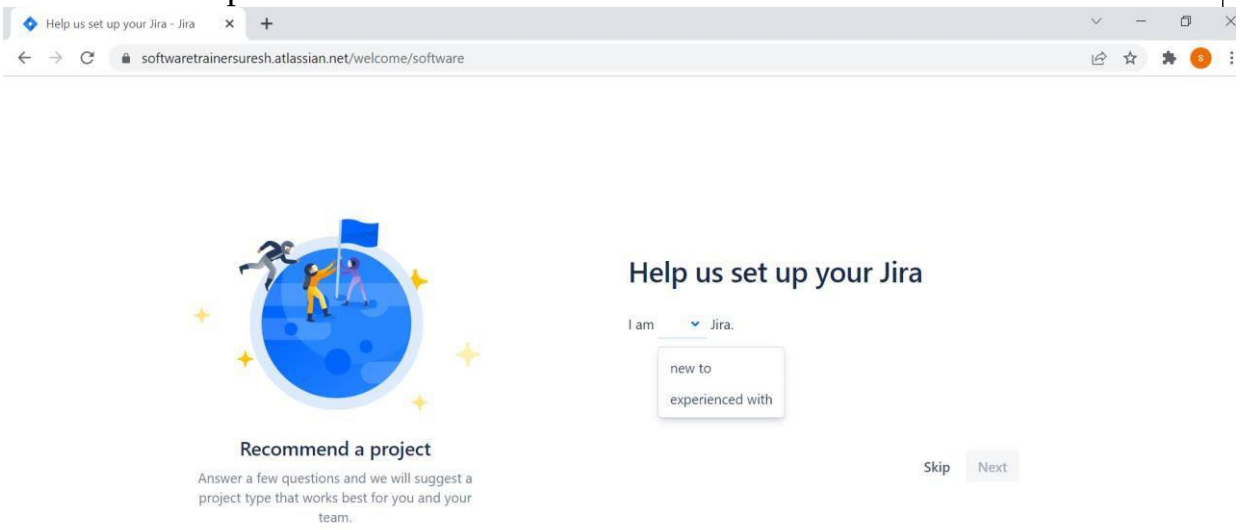
→Click on Skip button



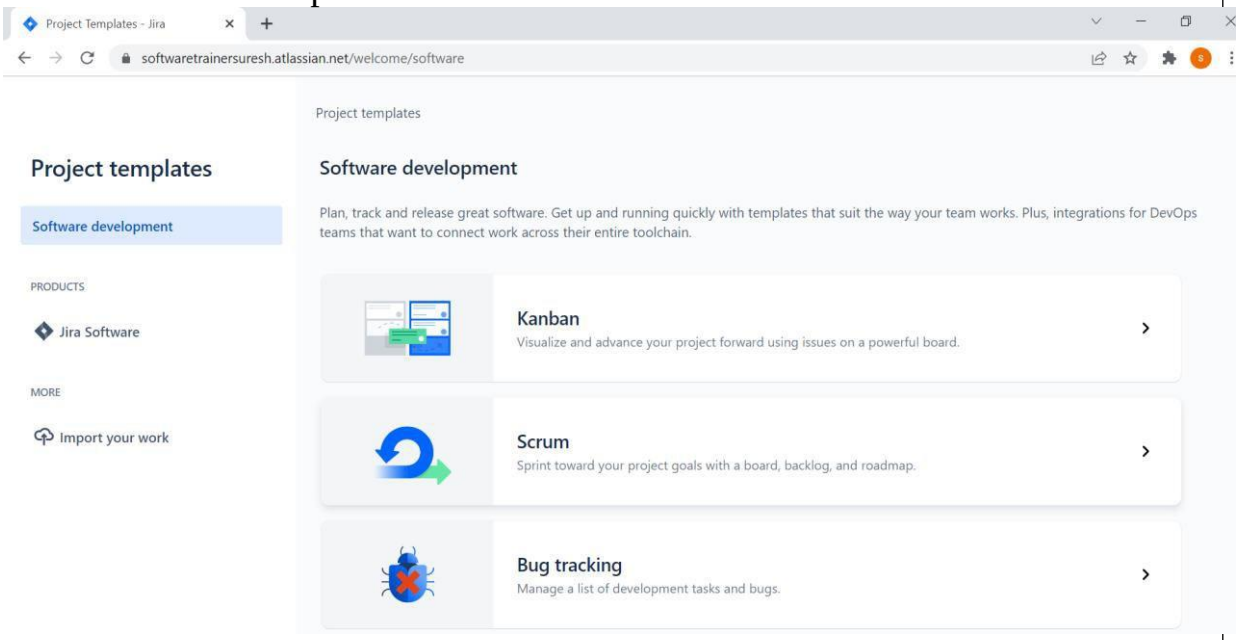
→Click on continue button



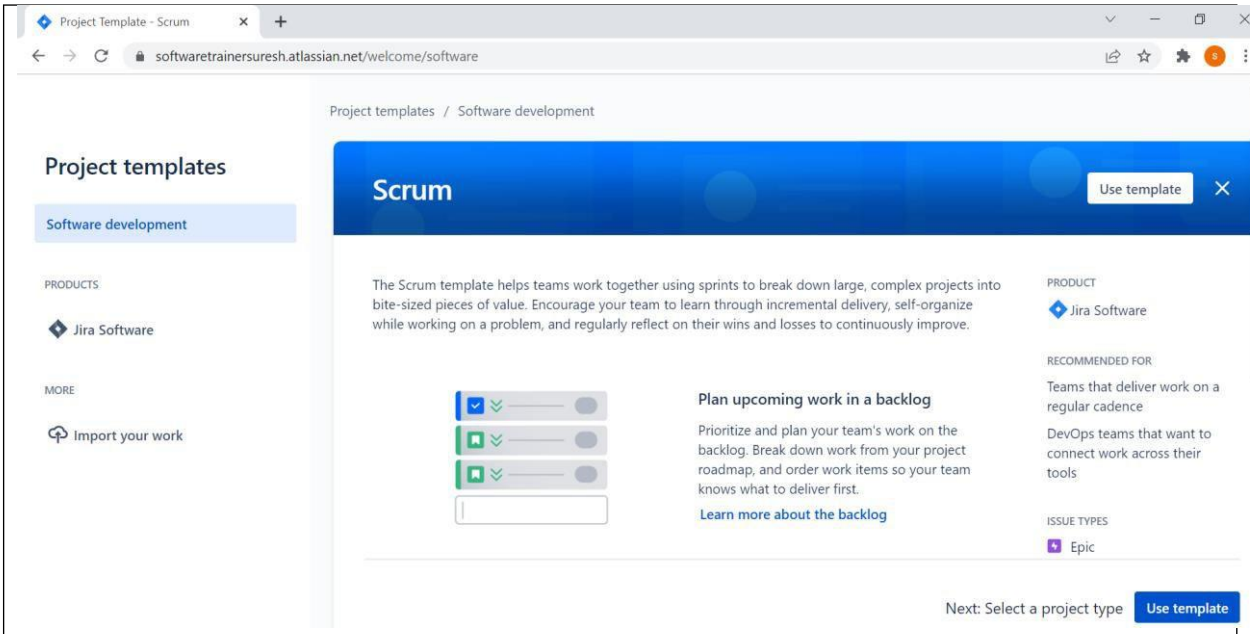
→Click on Skip button



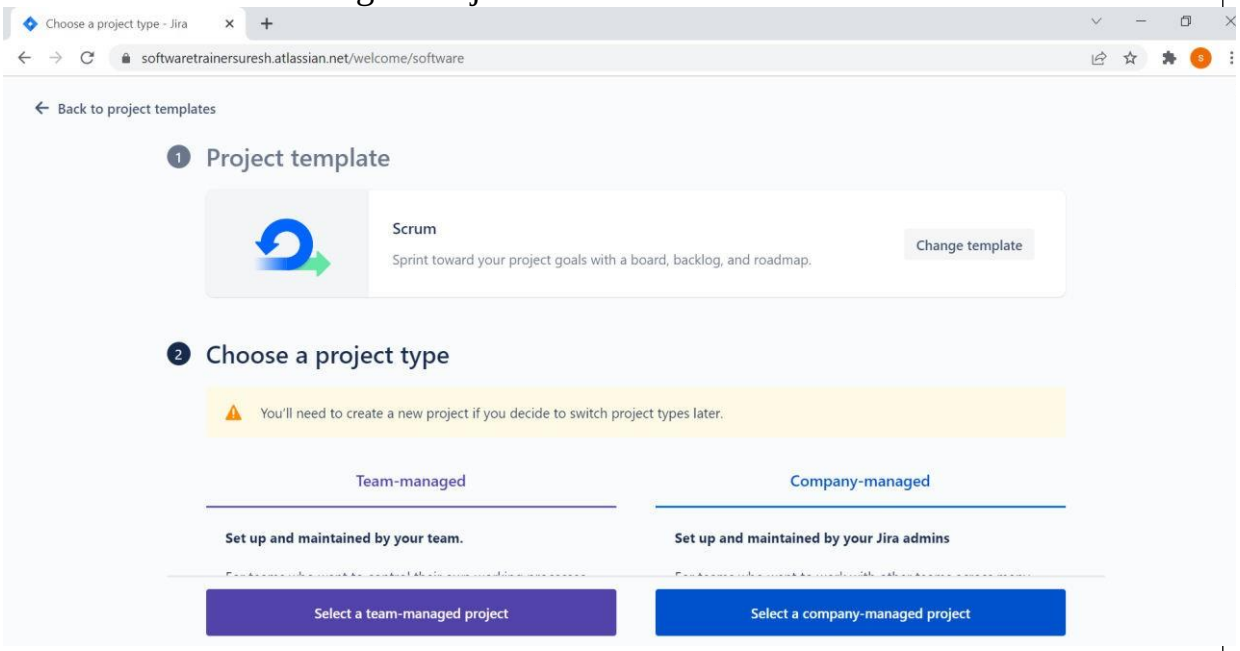
→Click on Scrum Option



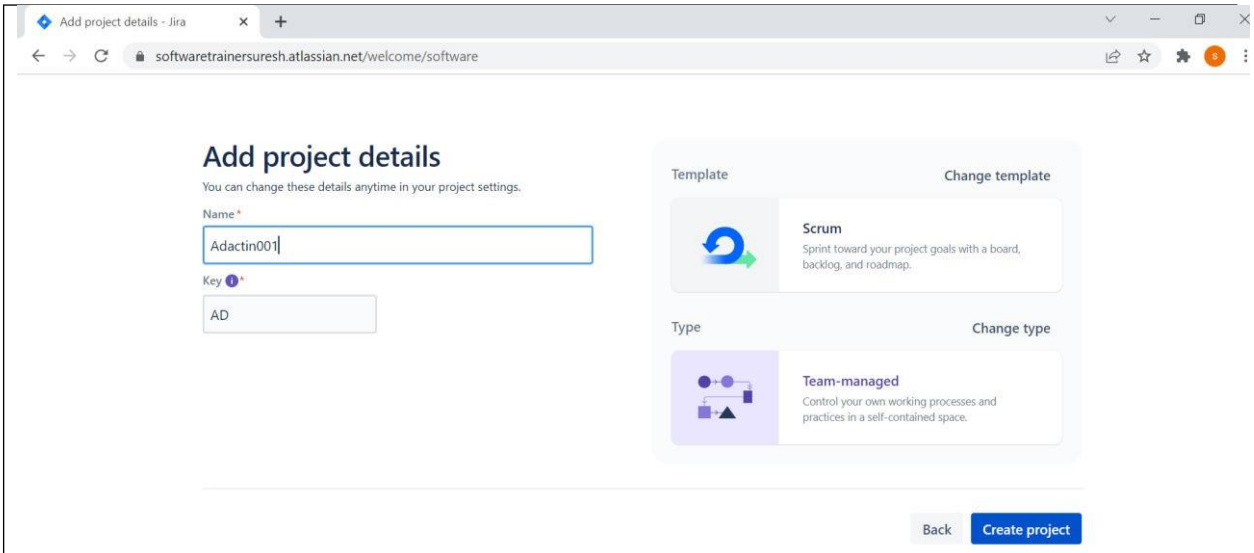
→Click on Use Template



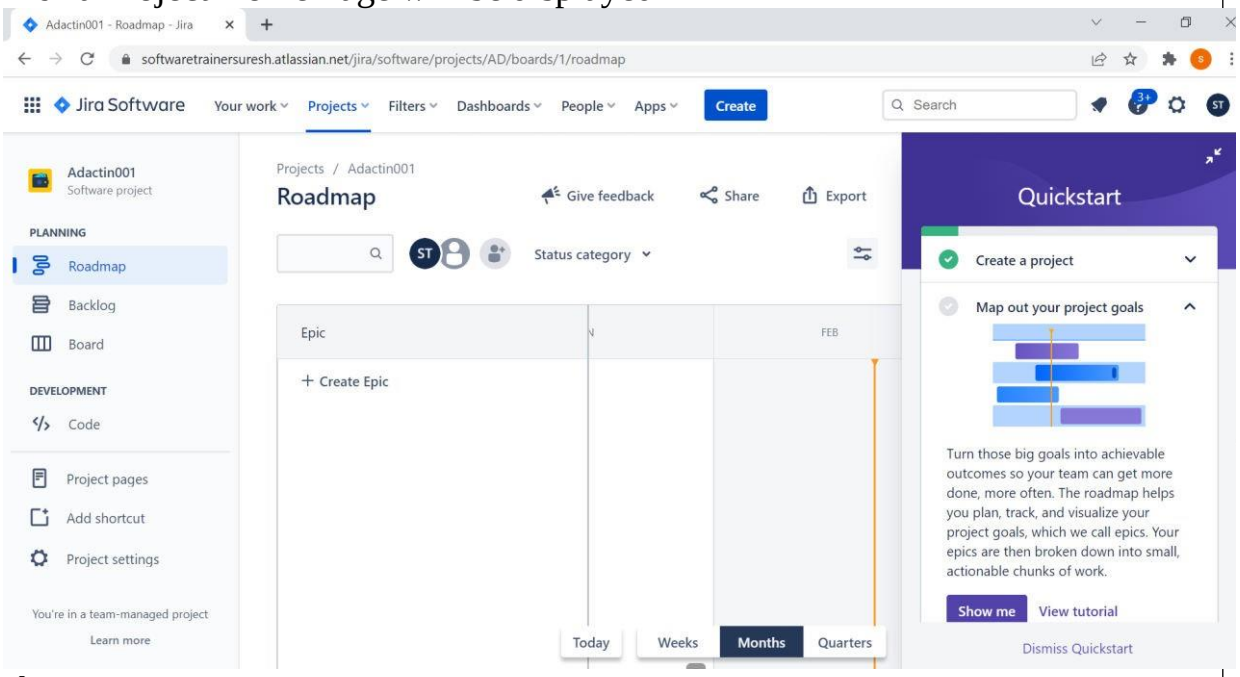
→Click on Team Managed Project button



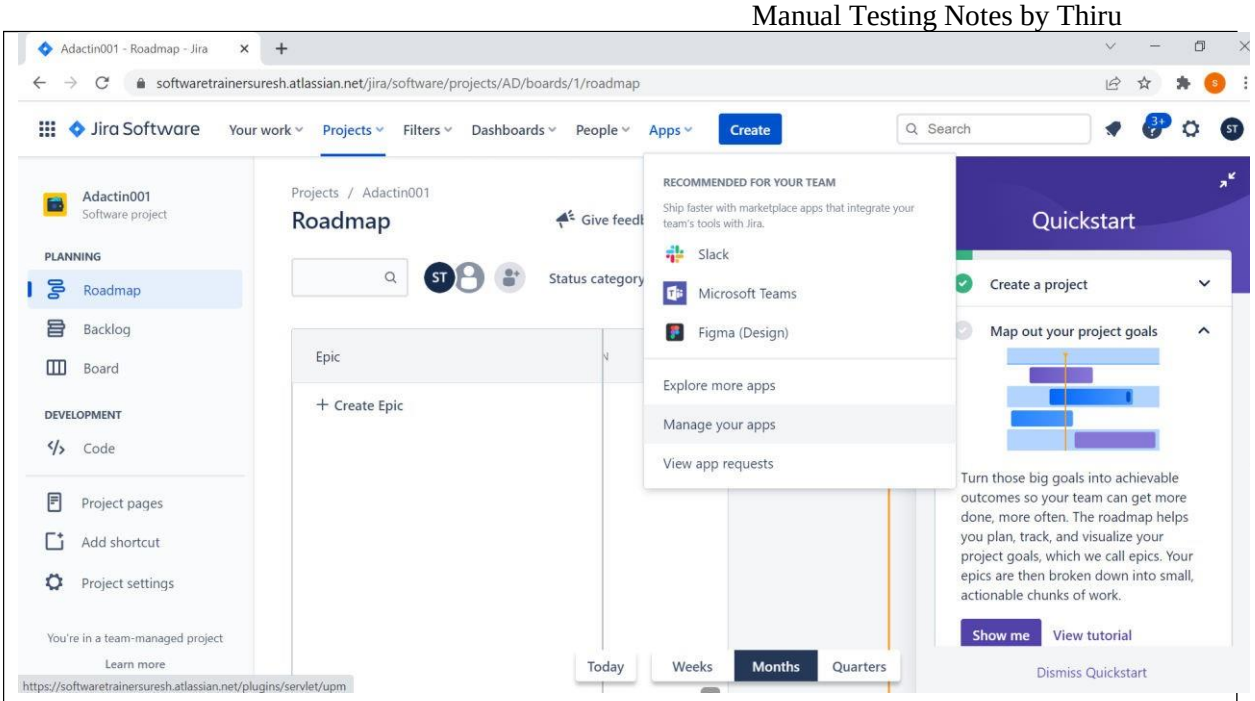
→Enter project name & Click on Create Project



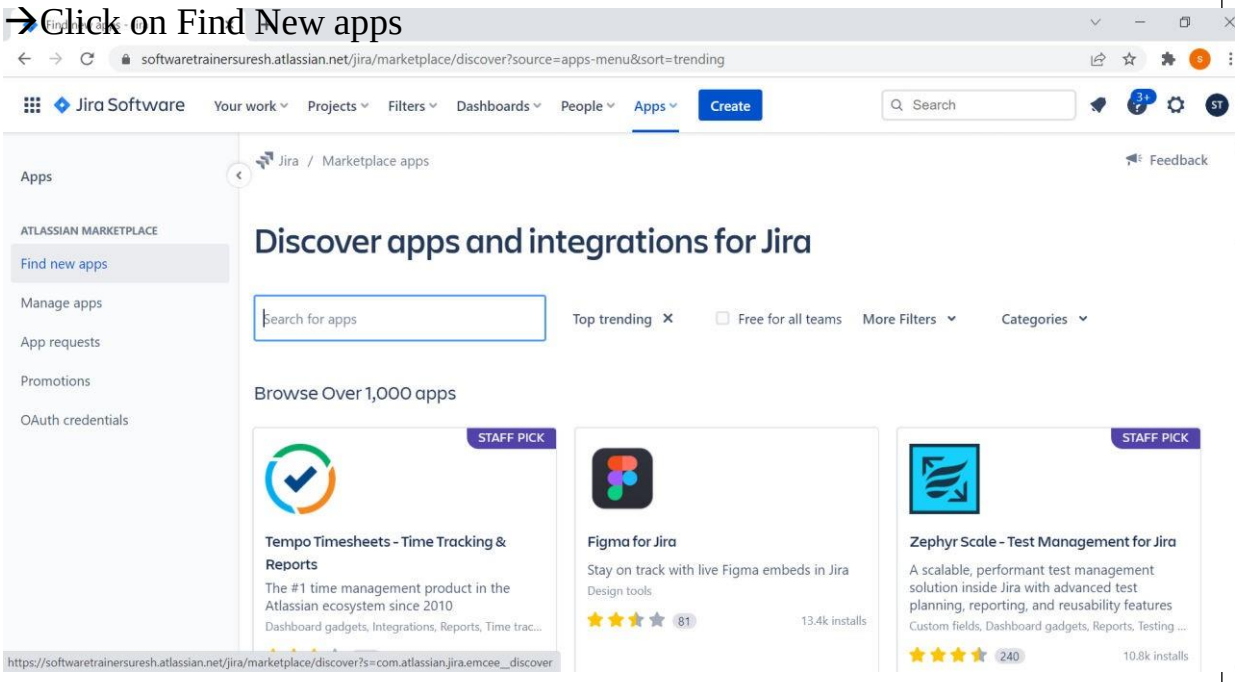
→Jira Project Home Page will be displayed



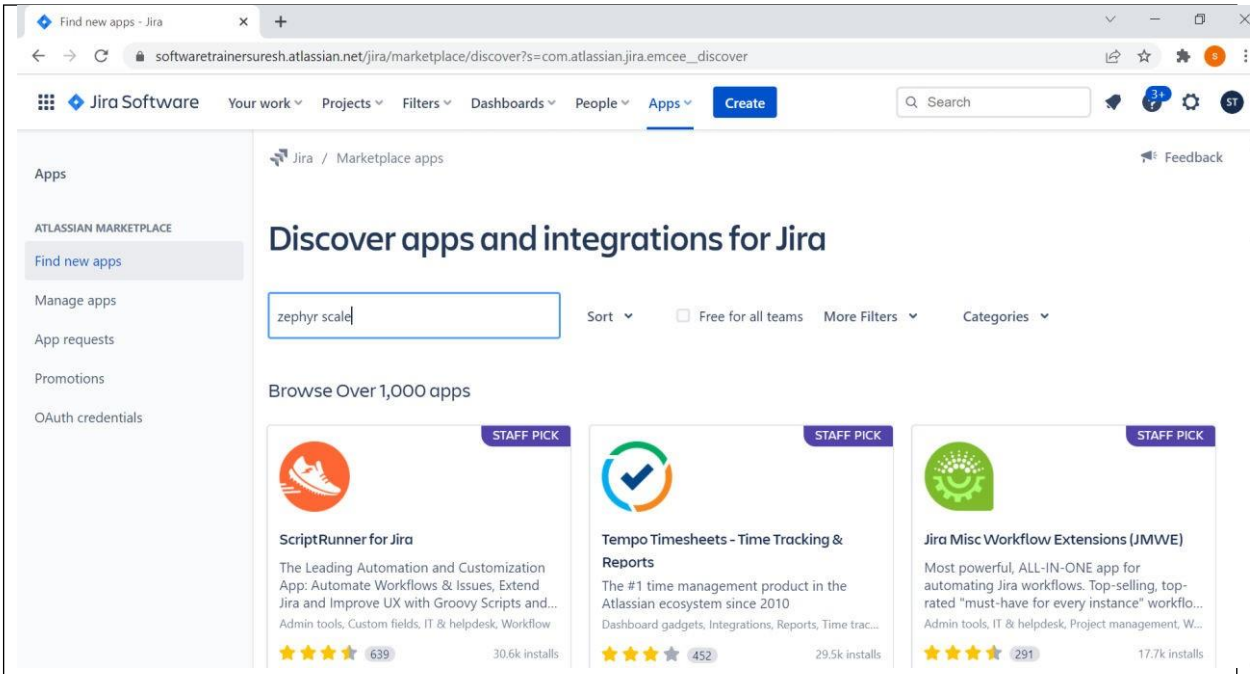
→Navigate to Apps Main menu & Click on Manage Apps Submenu



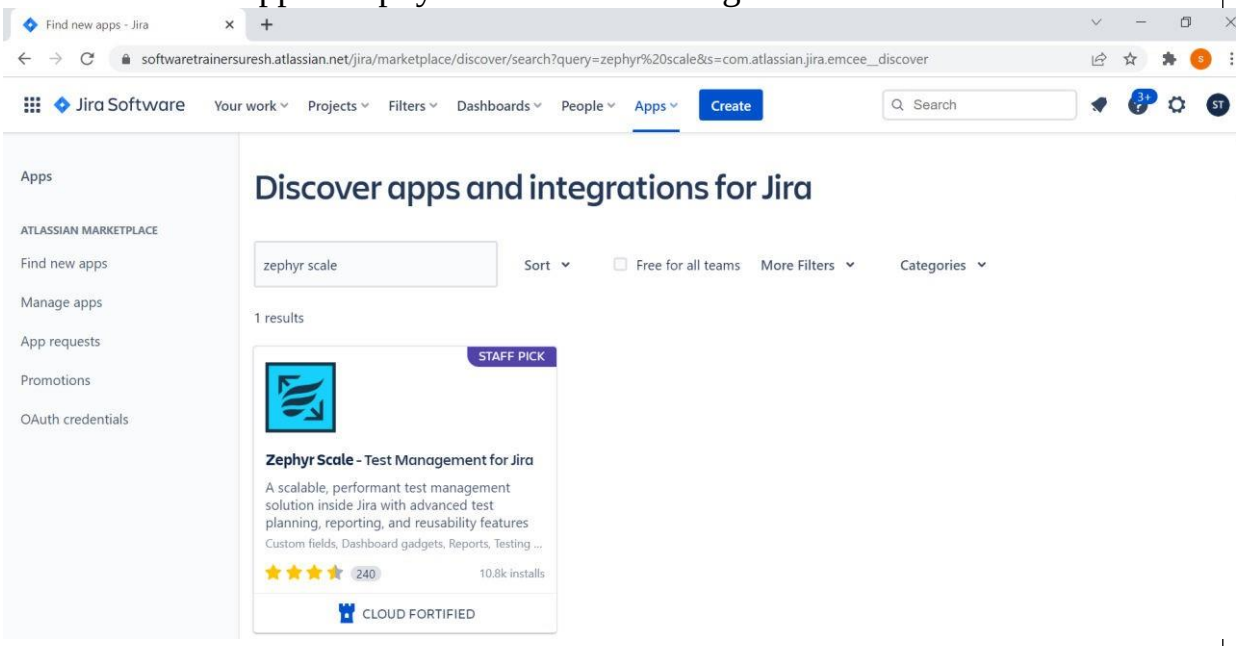
→Click on Find New apps



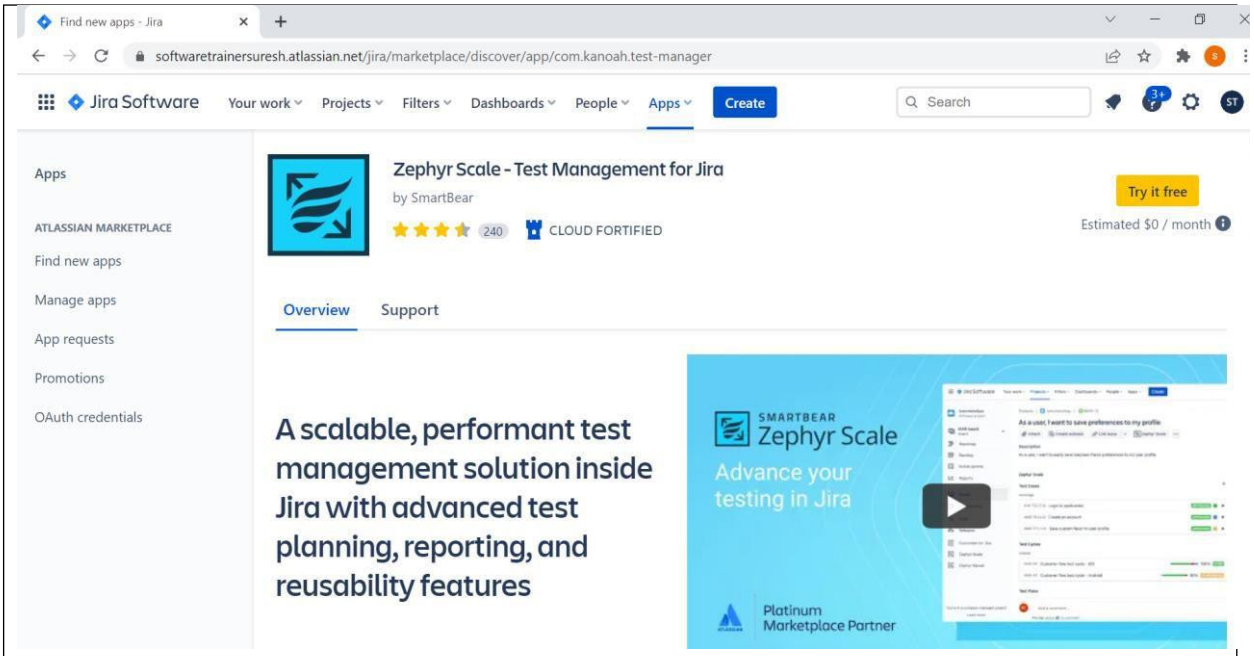
→Search for zephyr scale App – Enter zephyr scale in textbox & Click on ENTER button from keyboard



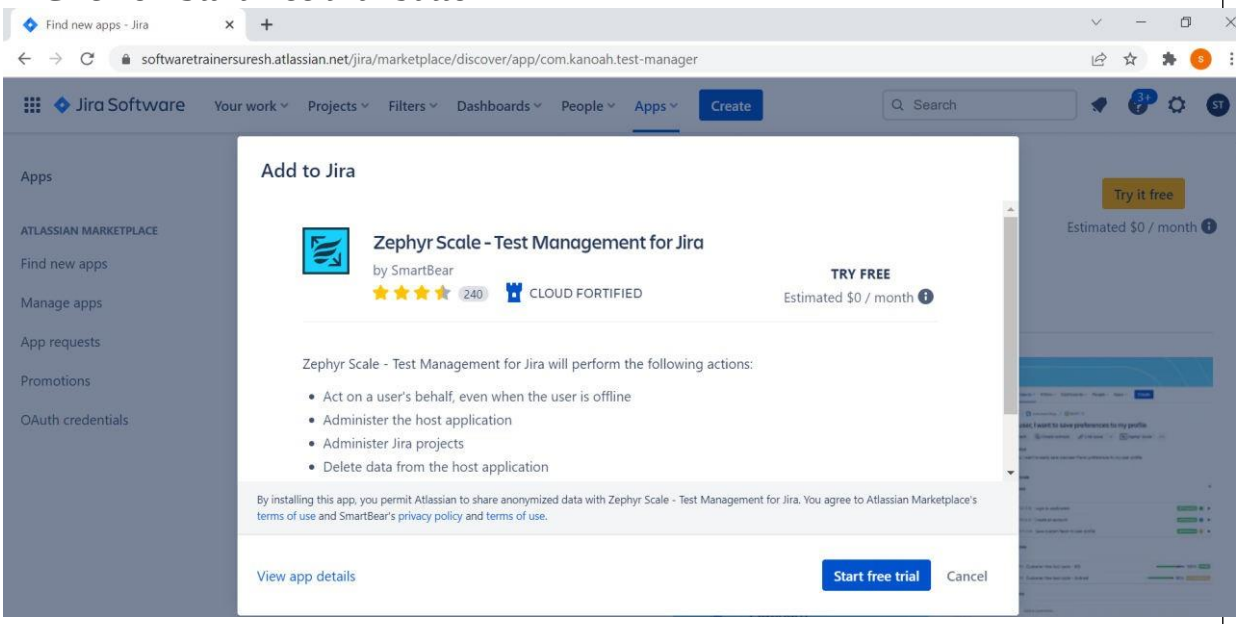
→Click on the App of zephyr scale – Test management for JIRA



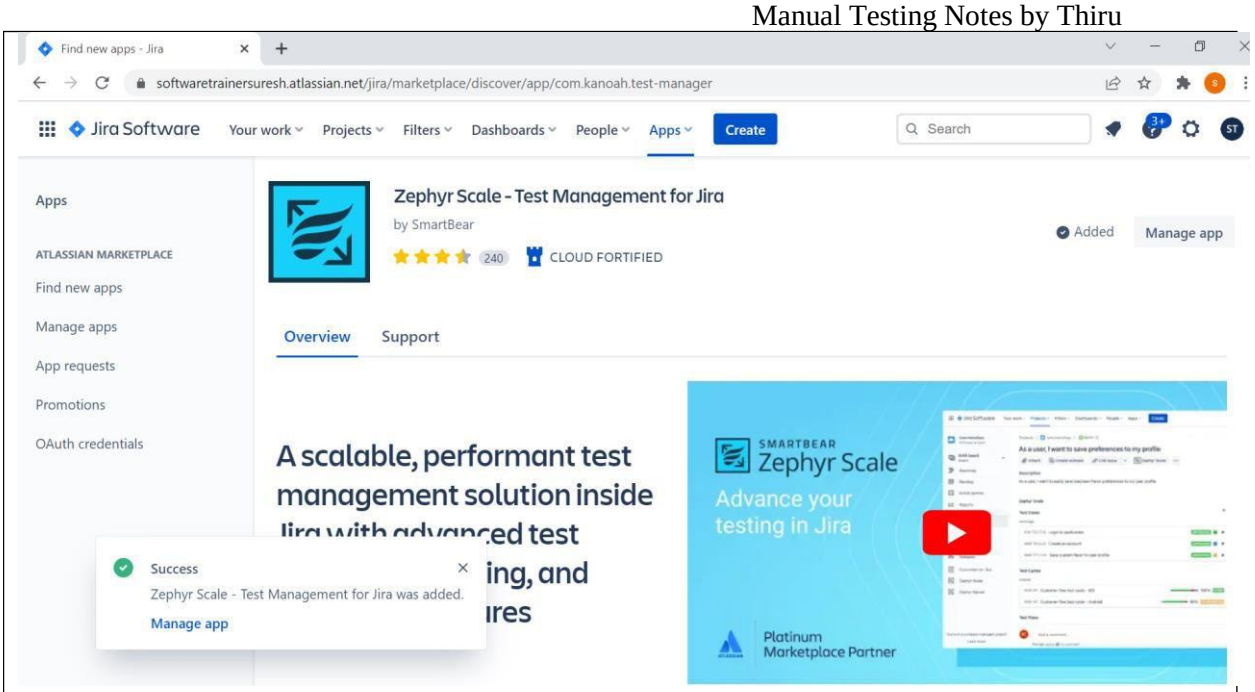
→Click on Try it Free button



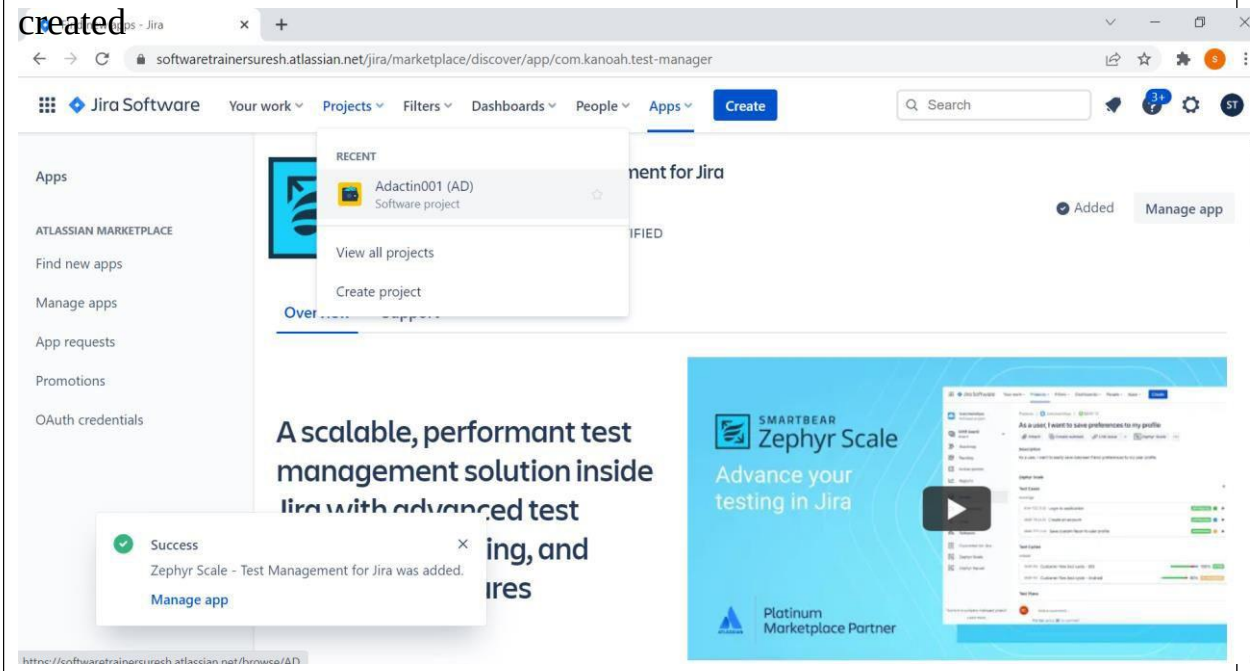
→Click on Start free trial button



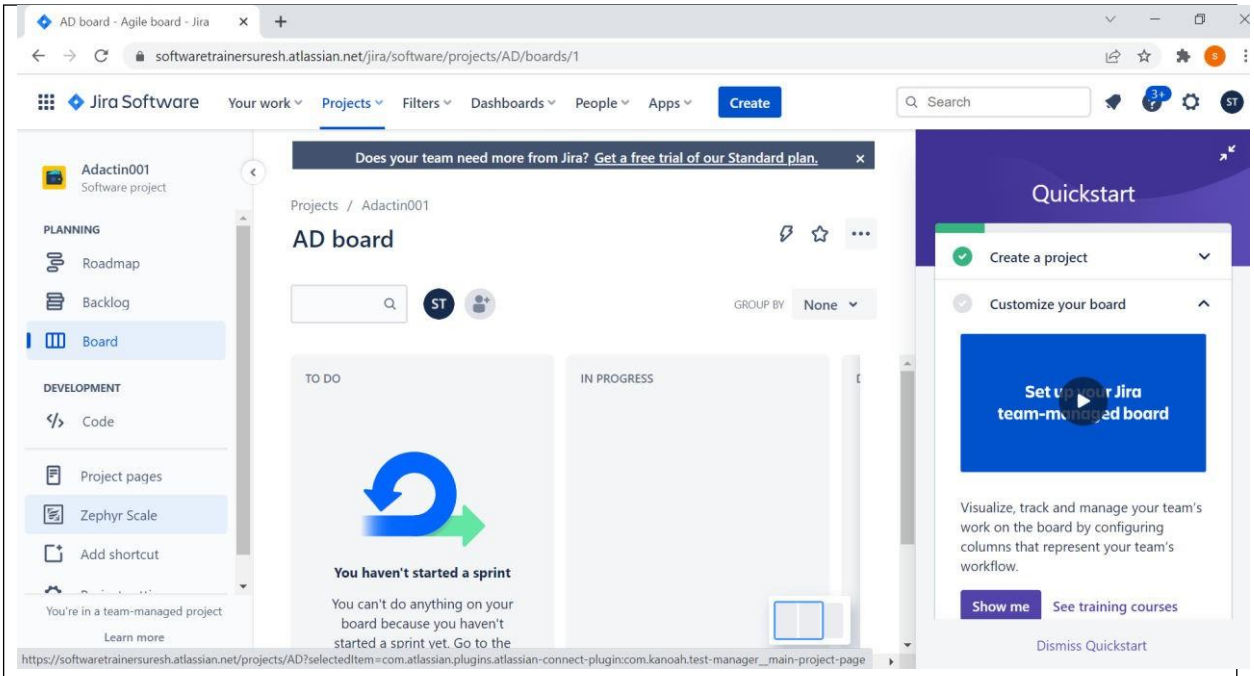
→Your will get success message



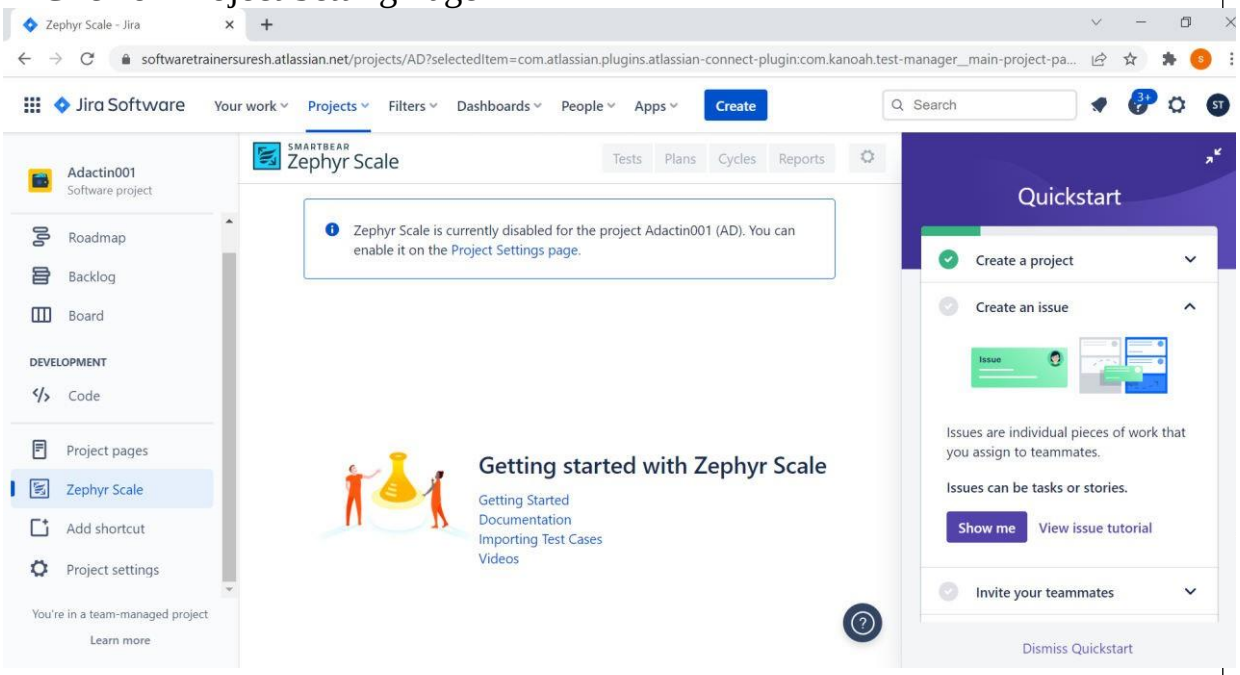
→Navigate to Projects Main menu and select the Project which is previously created



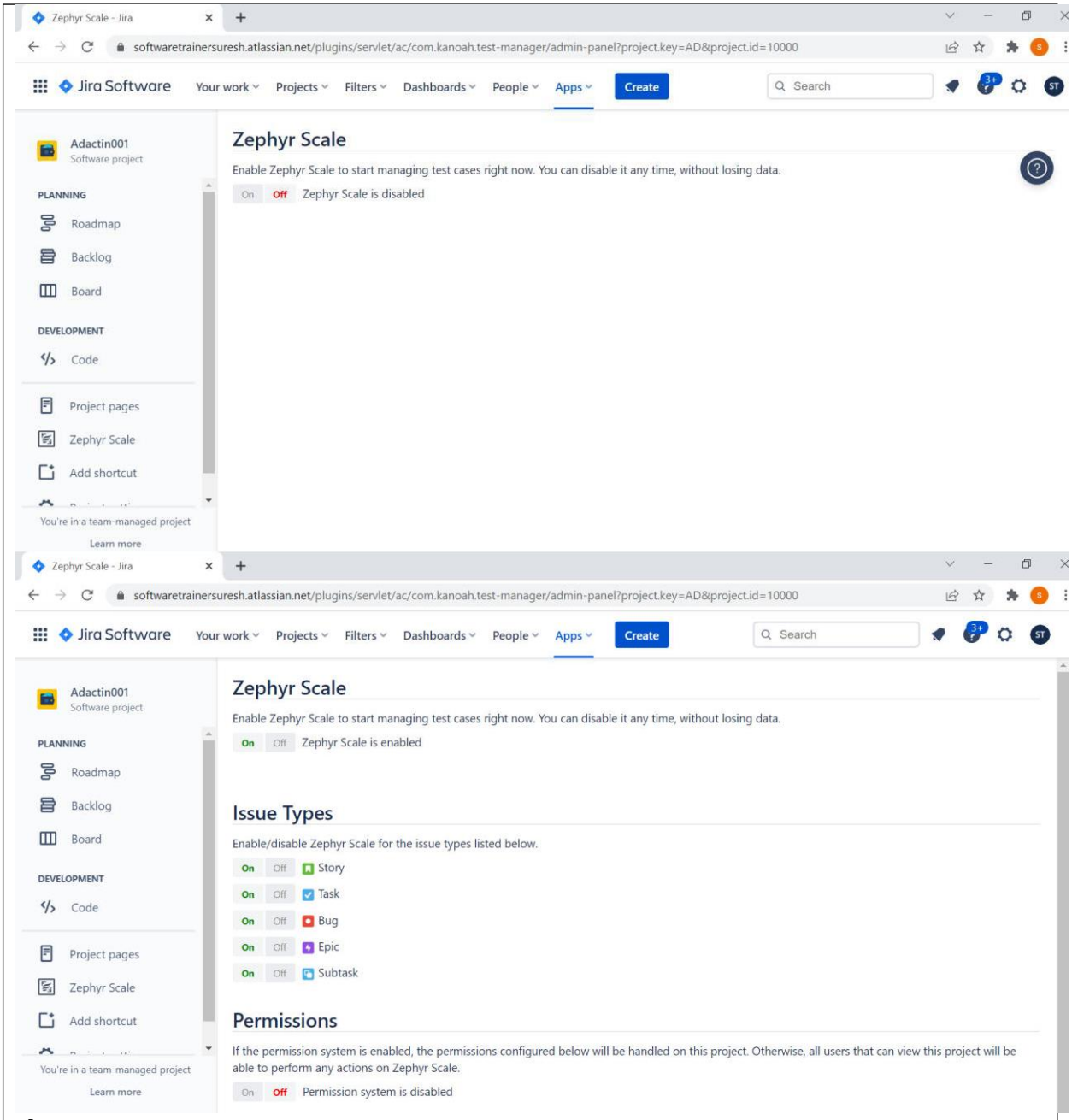
→Click on Zephyr Scale Option



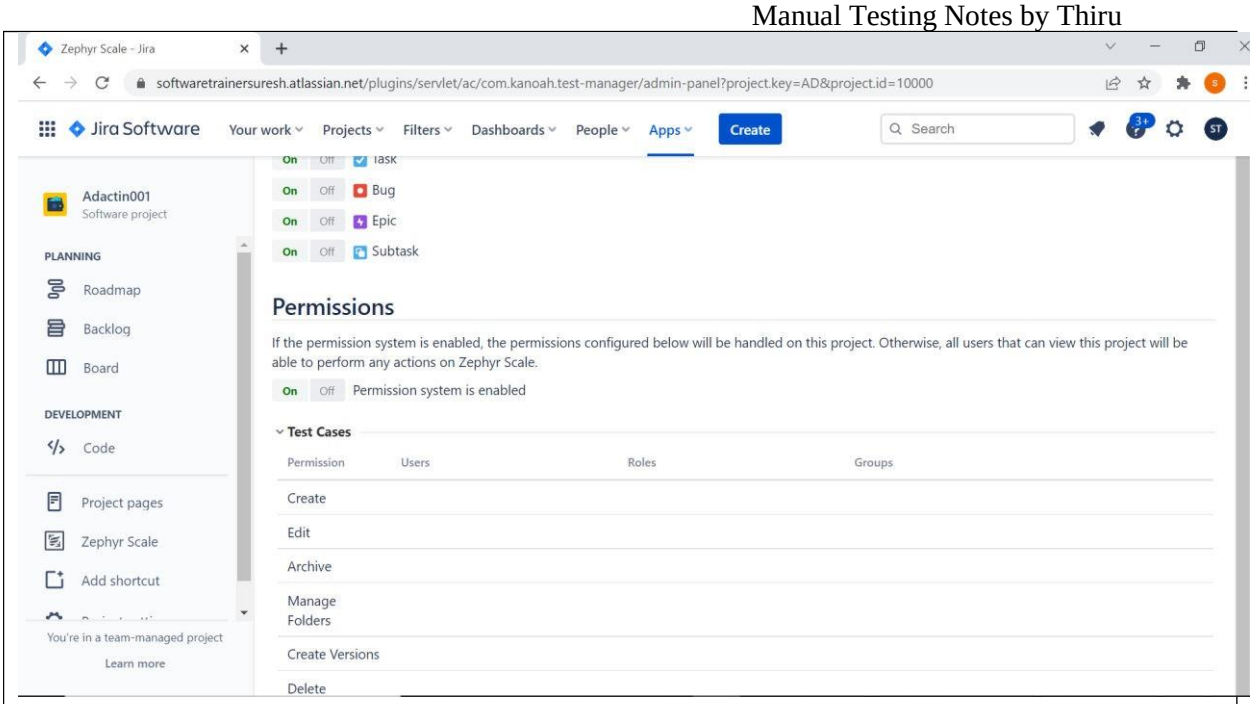
→Click on Project Setting Page



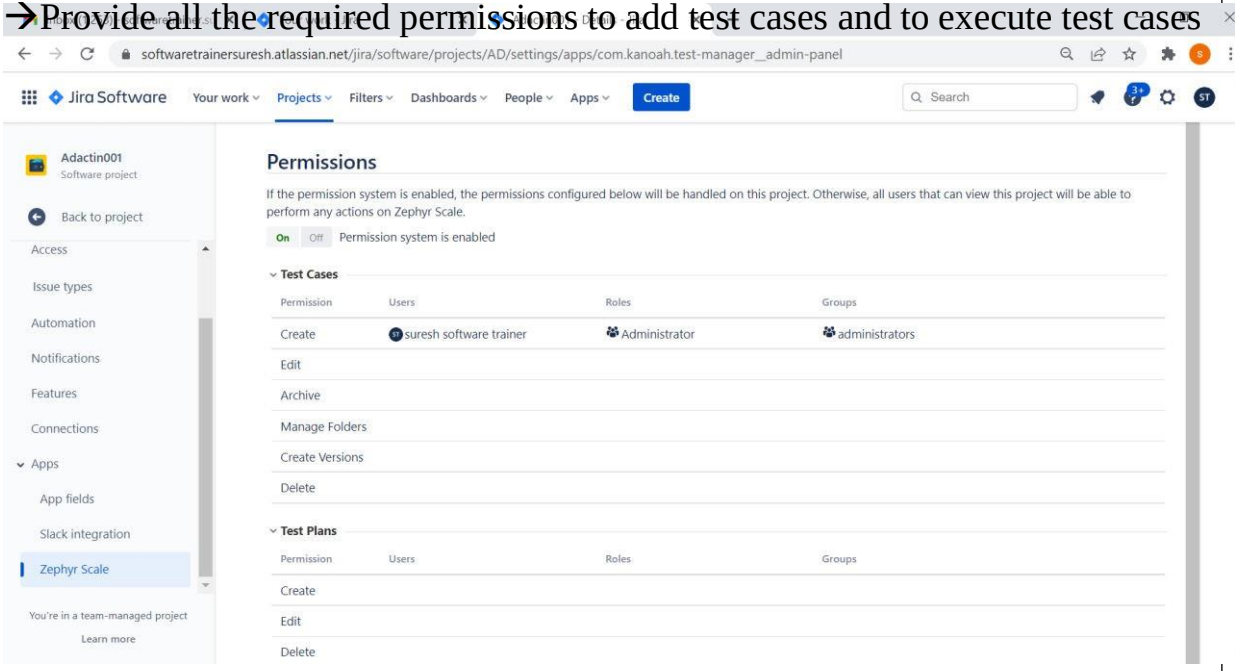
→Click - on button



→Click on button under permissions



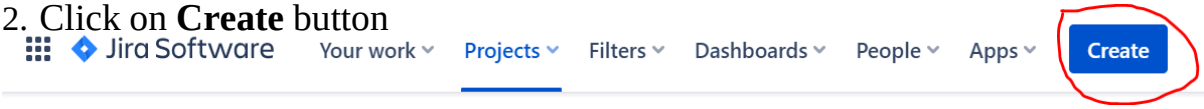
→Provide all the required permissions to add test cases and to execute test cases



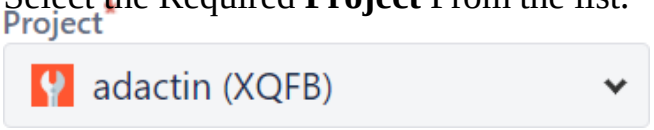
**By this Setup will be completed

→Steps to Report Bugs in JIRA Tool

1. Login to JIRA tool
2. Click on **Create** button

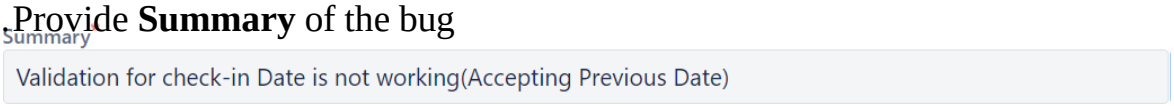


3. Select the Required **Project** From the list.

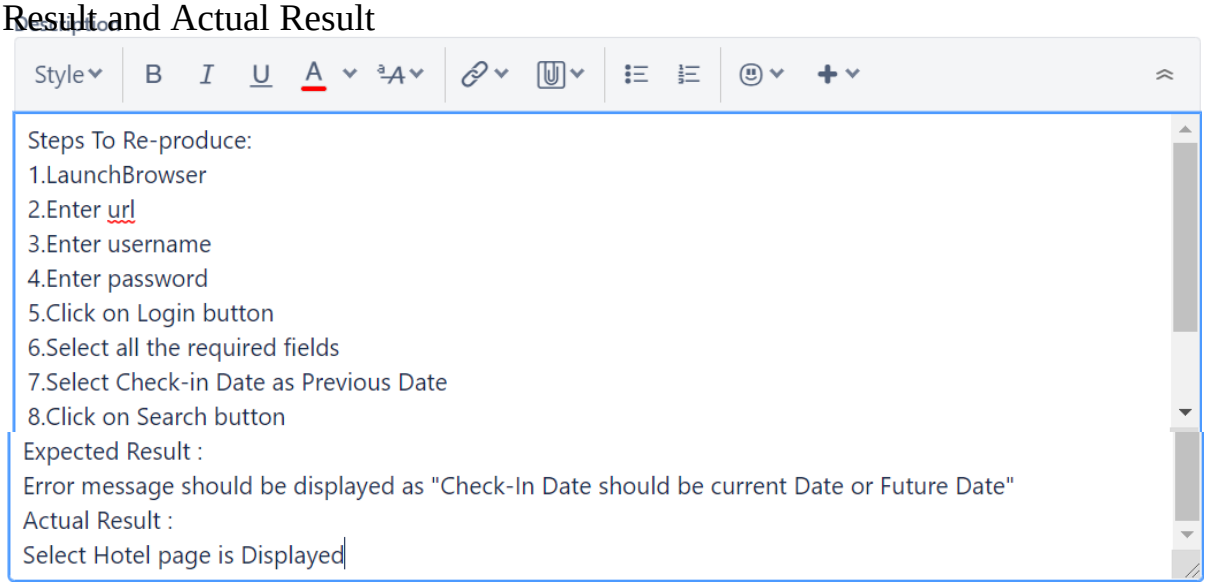


4. Select the **Bug** option from Issue Type dropdown

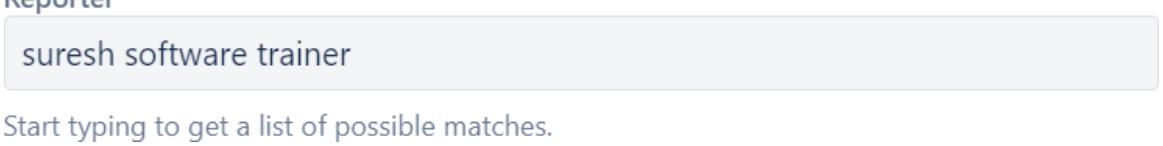
5. Provide **Summary** of the bug



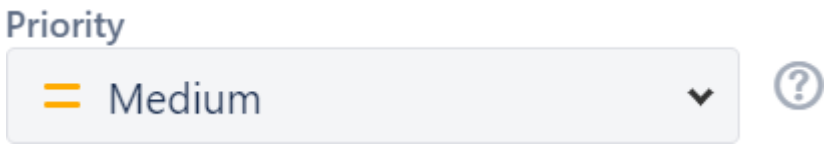
6. Provide the **Bug Description** which includes – Steps to re-produce , Expected Result and Actual Result



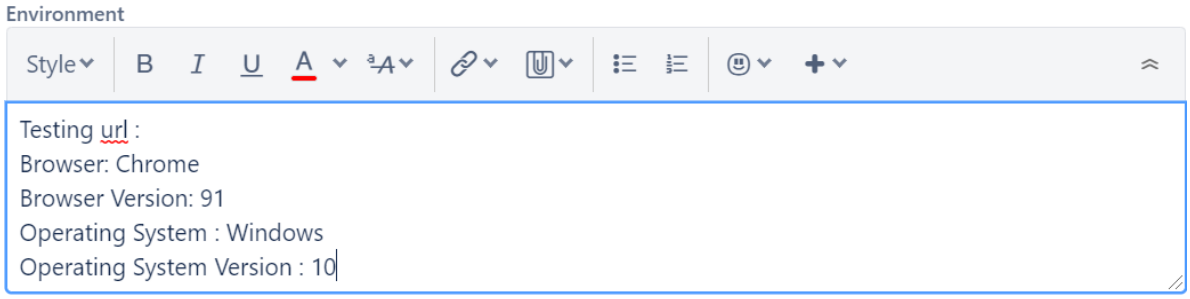
7. Provide Tester name



8. Provide Priority for the bug

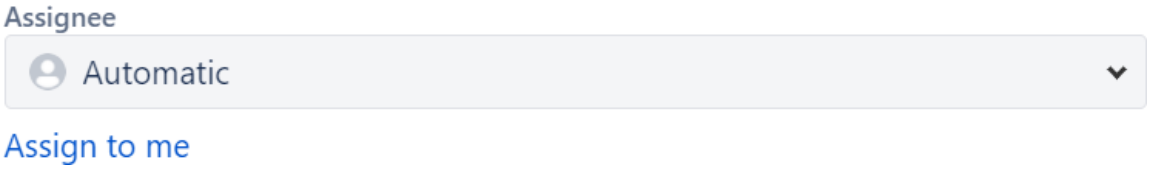


9. Mention Environment details



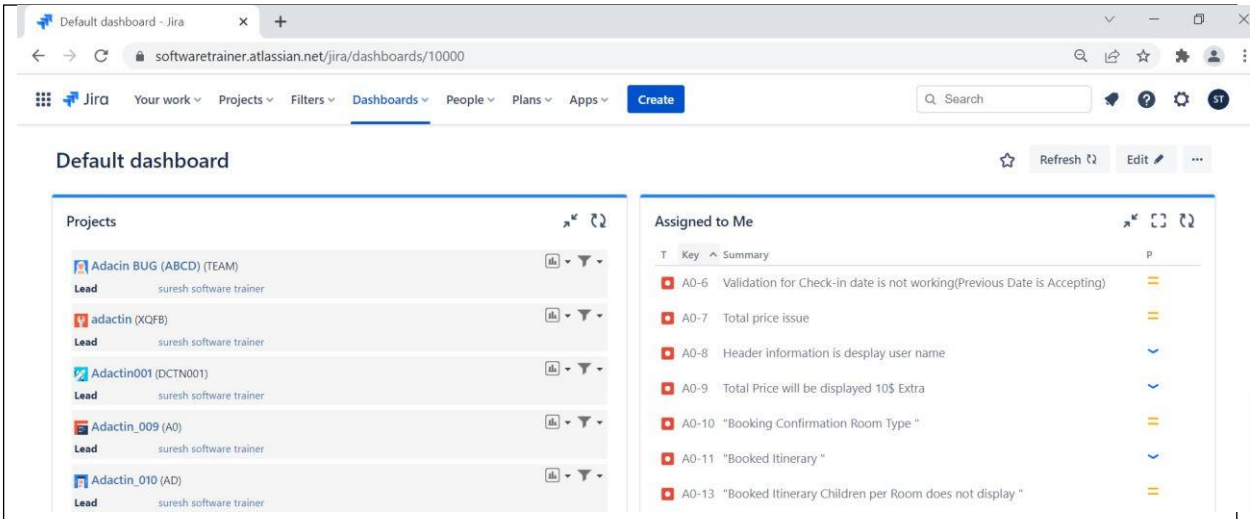
10. in case any screenshot attach the required files

11. Select the developer name from Assignee dropdown



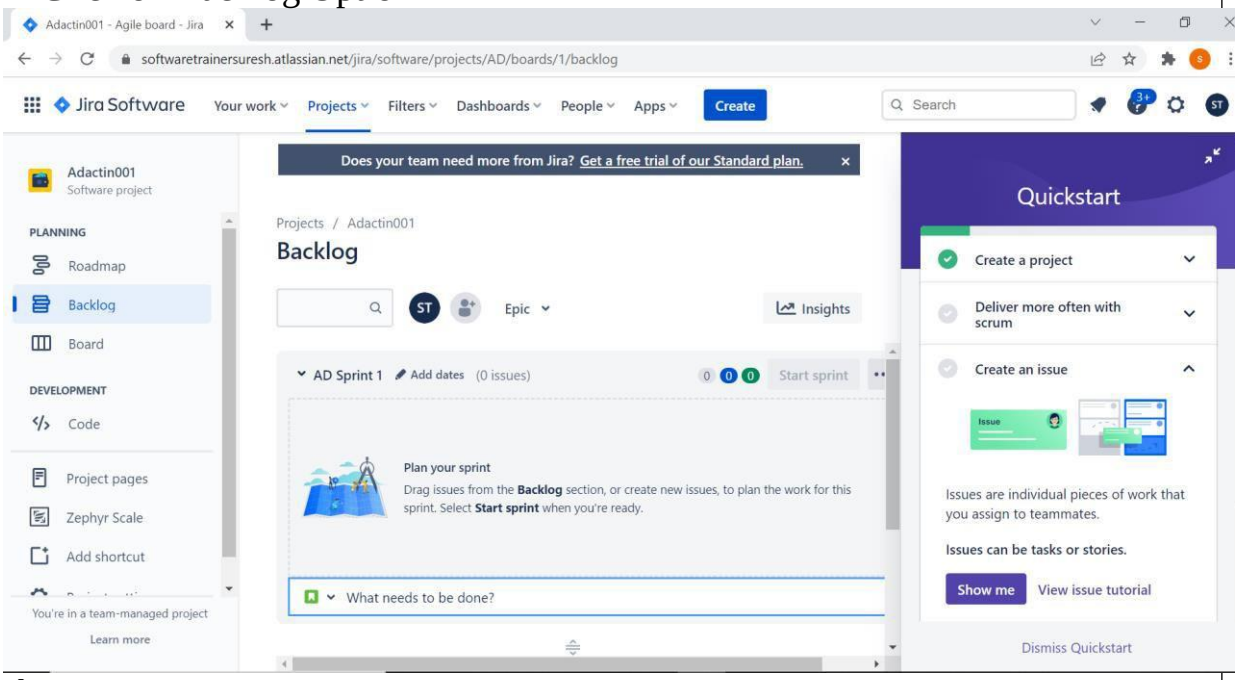
12. Click on Create button

13. Bug will be created.
14. Refer Default dashboard to check created bugs

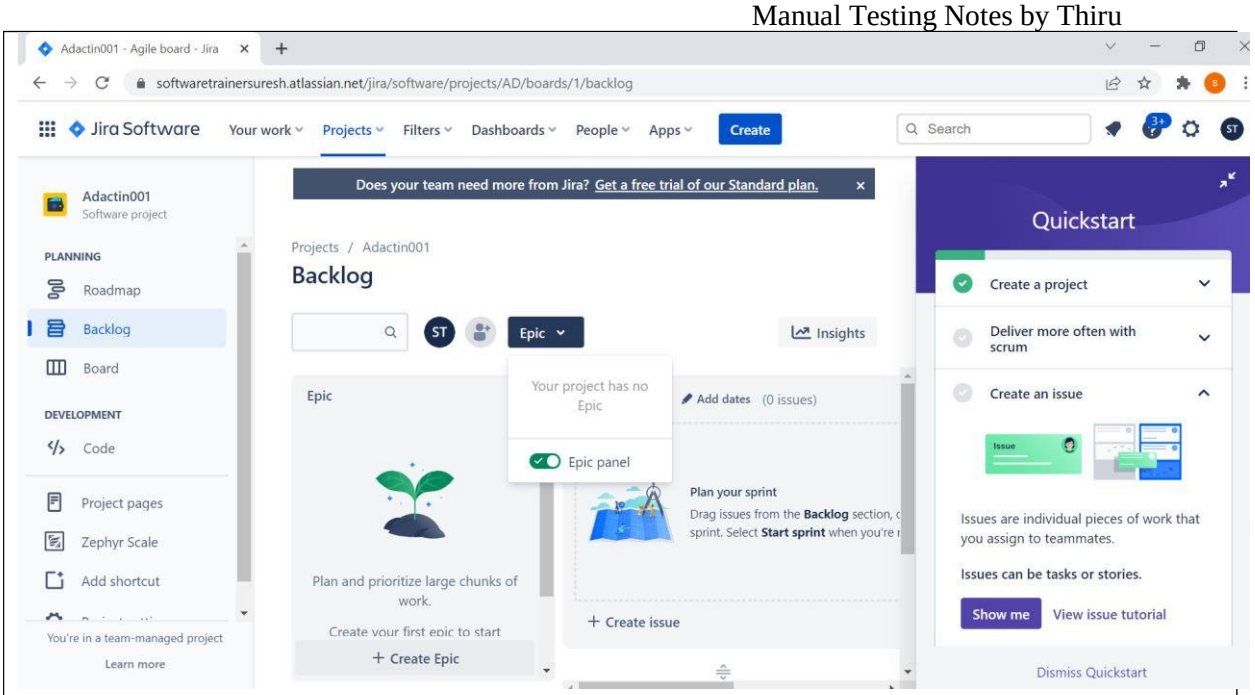


***Sprint Process

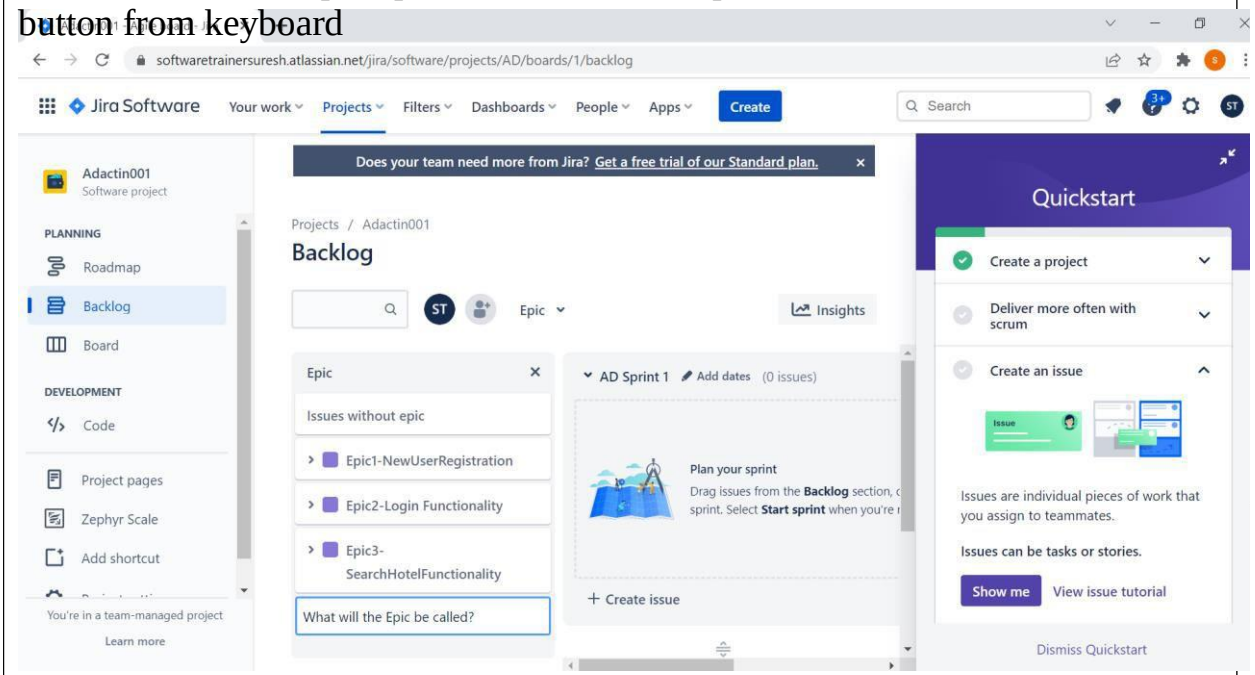
→Click on Backlog Option



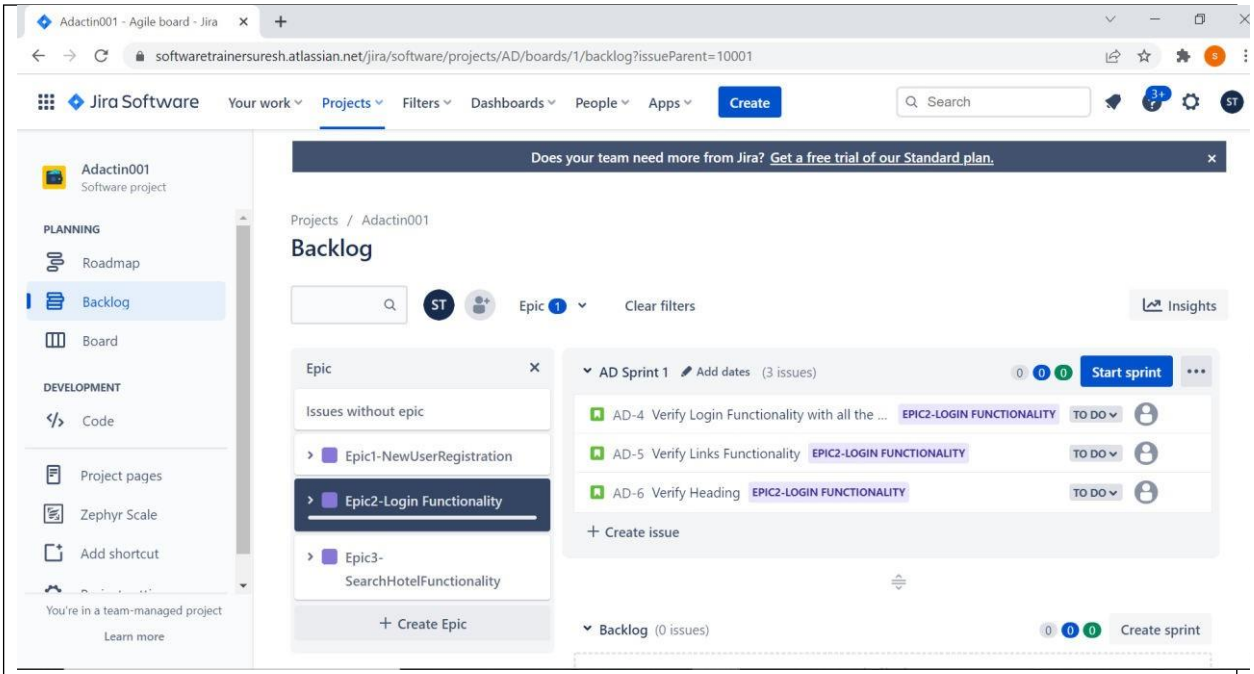
→Click on Epic option & Enable Epic Panel



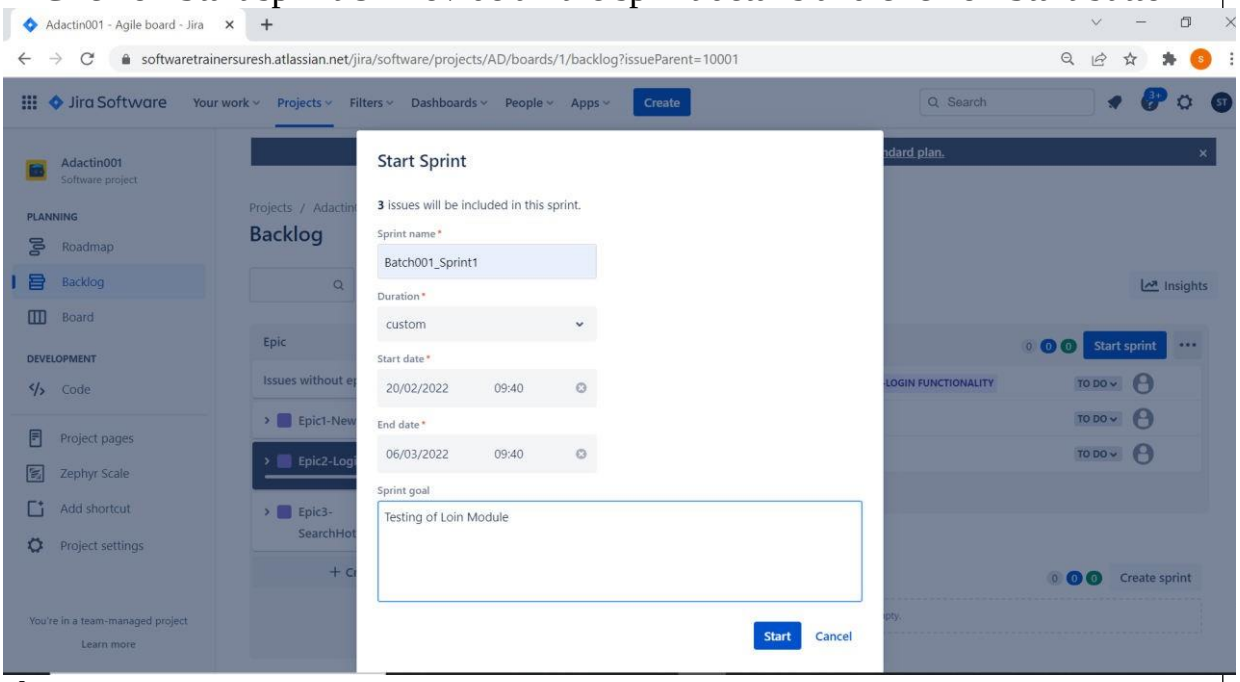
→Click on Create Epic option & Provide all Epic details and Click on ENTER button from keyboard



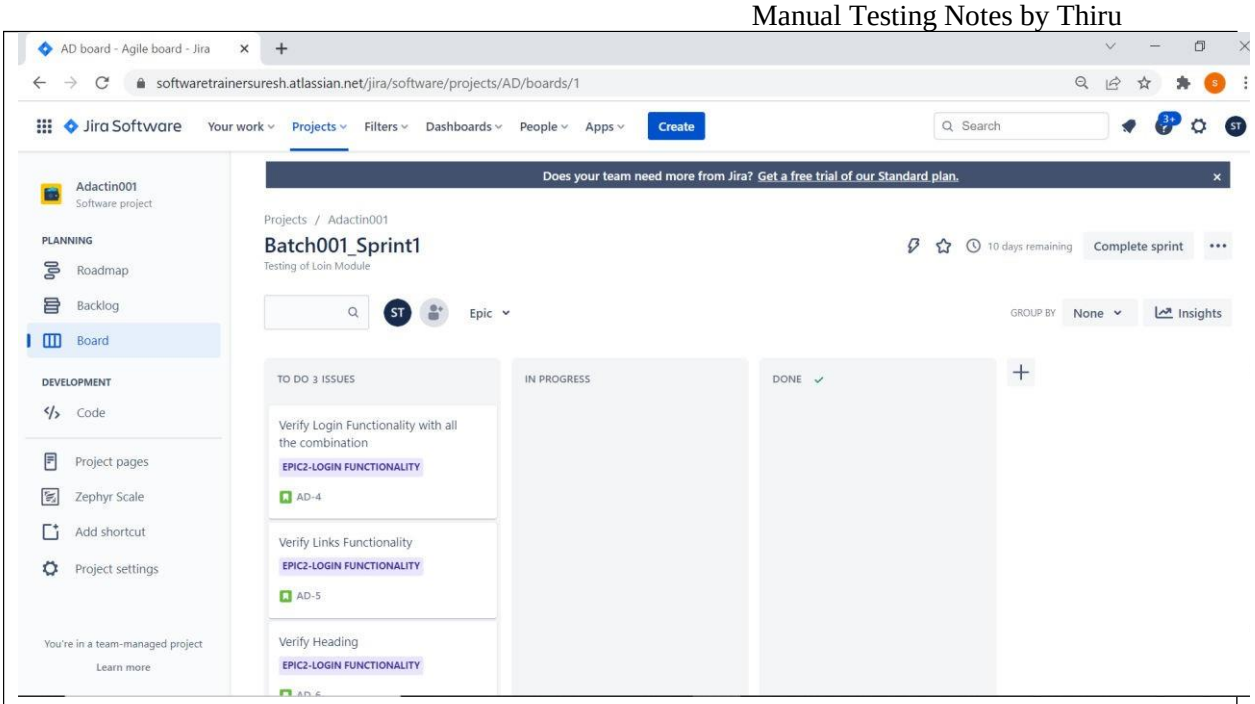
→Select Required Epic and Create issue



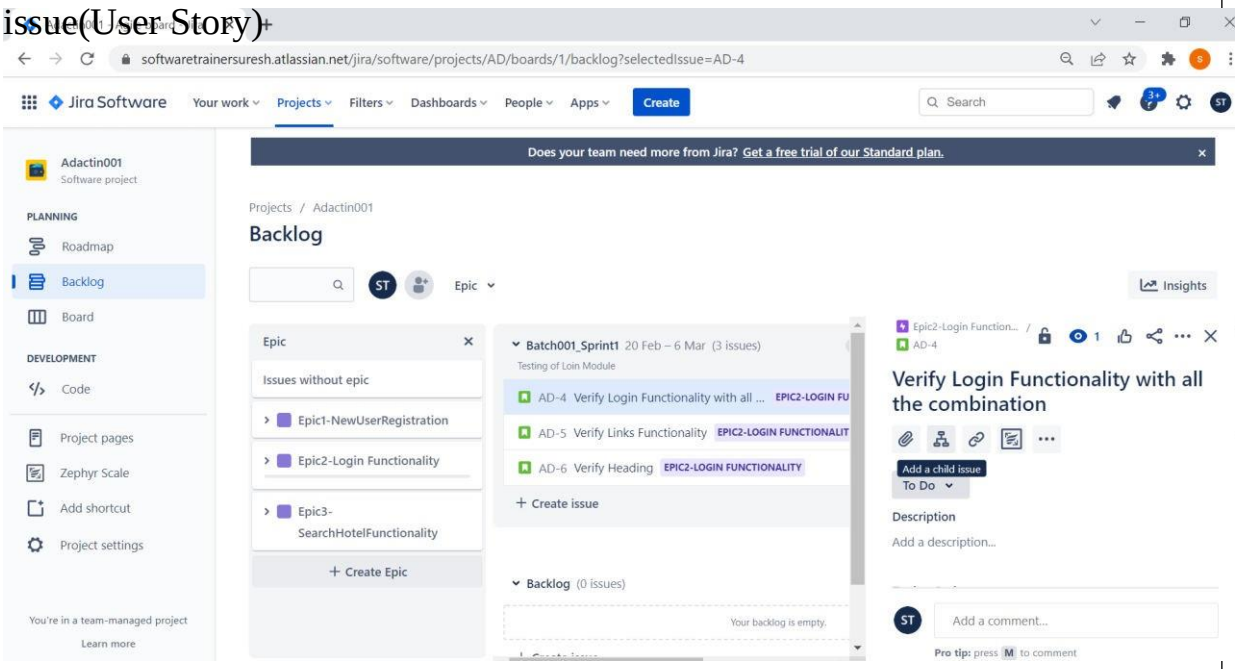
→Click on Start sprint & Provide all the sprint details and click on Start button



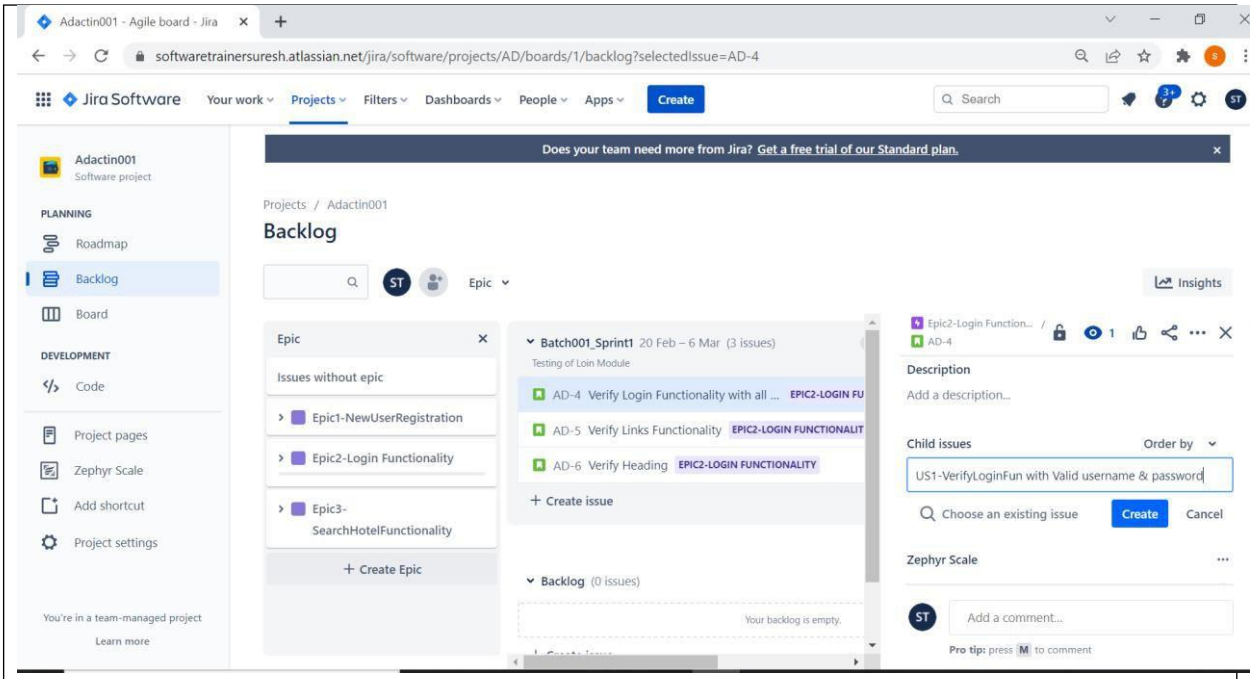
→Sprint Home Page will be displayed as below



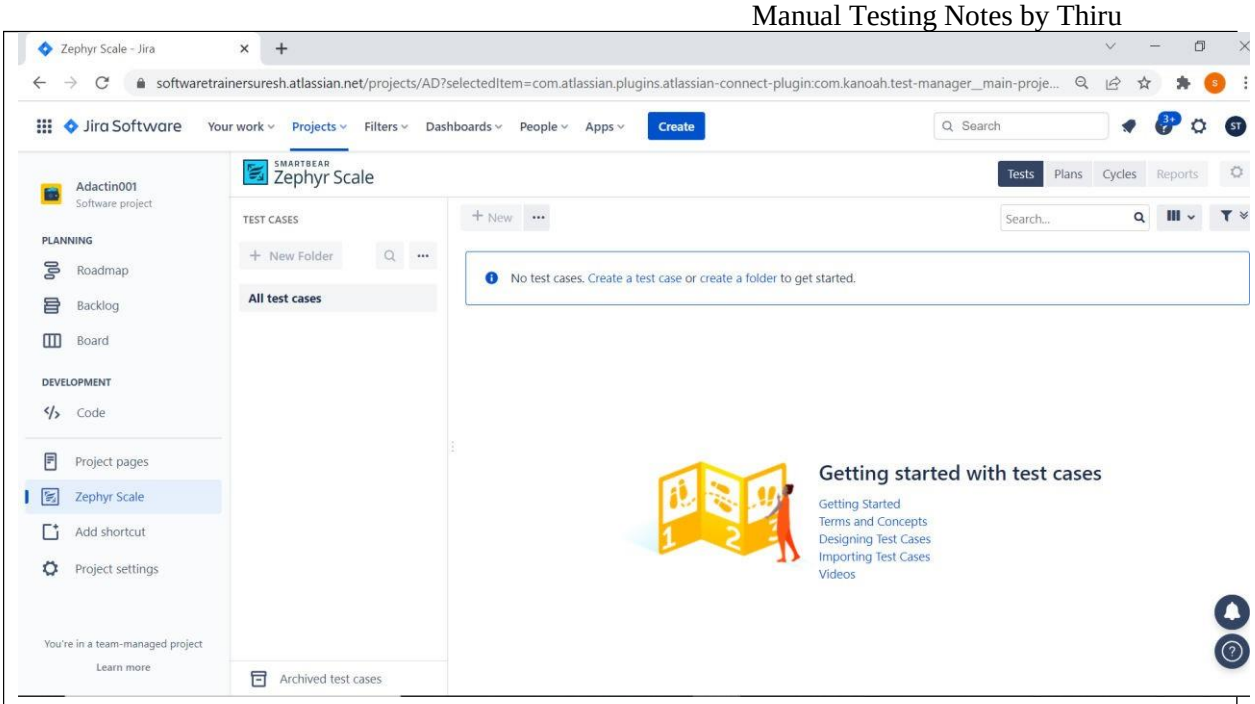
→Click on BackLog main option and select required issue to create child issue(User Story)



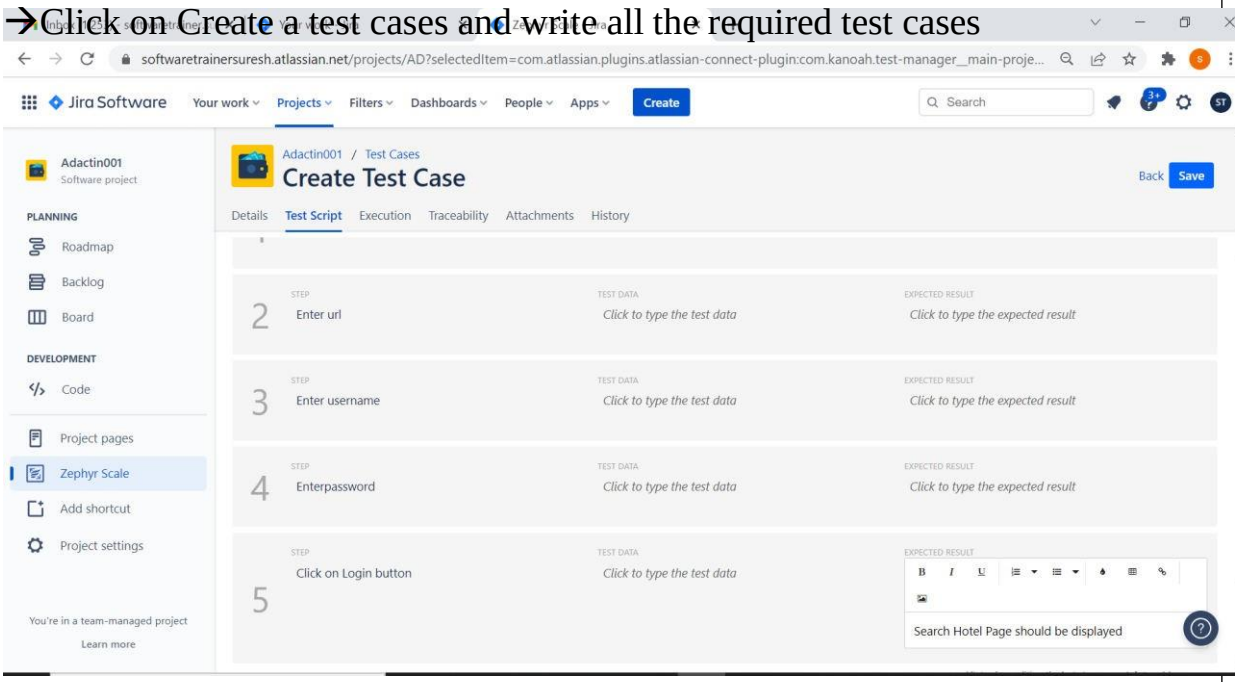
→Provide user stories and click on Create button



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



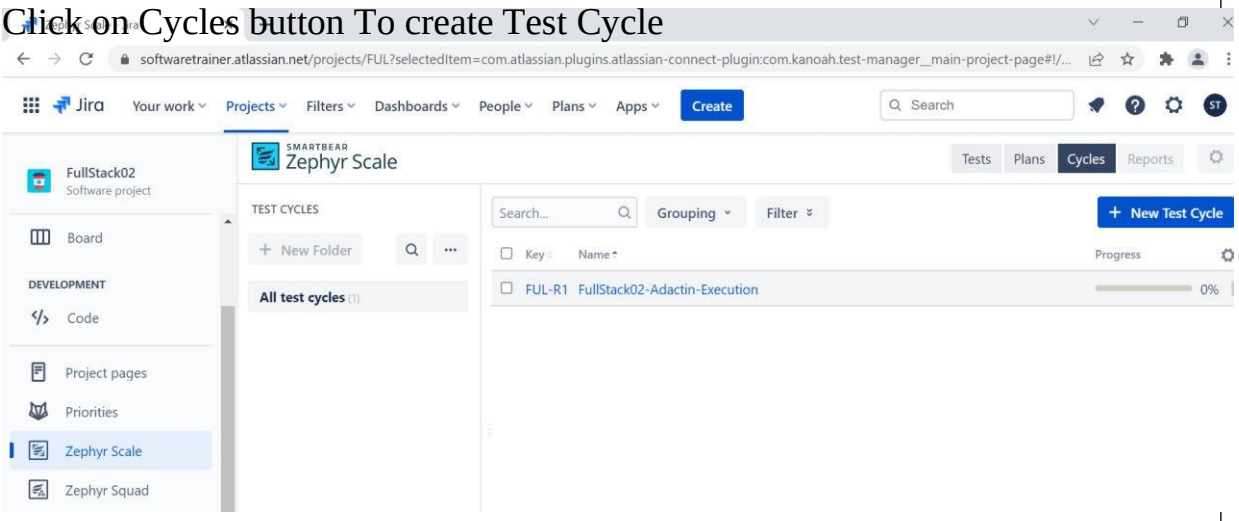
→Click on Create a test cases and write all the required test cases



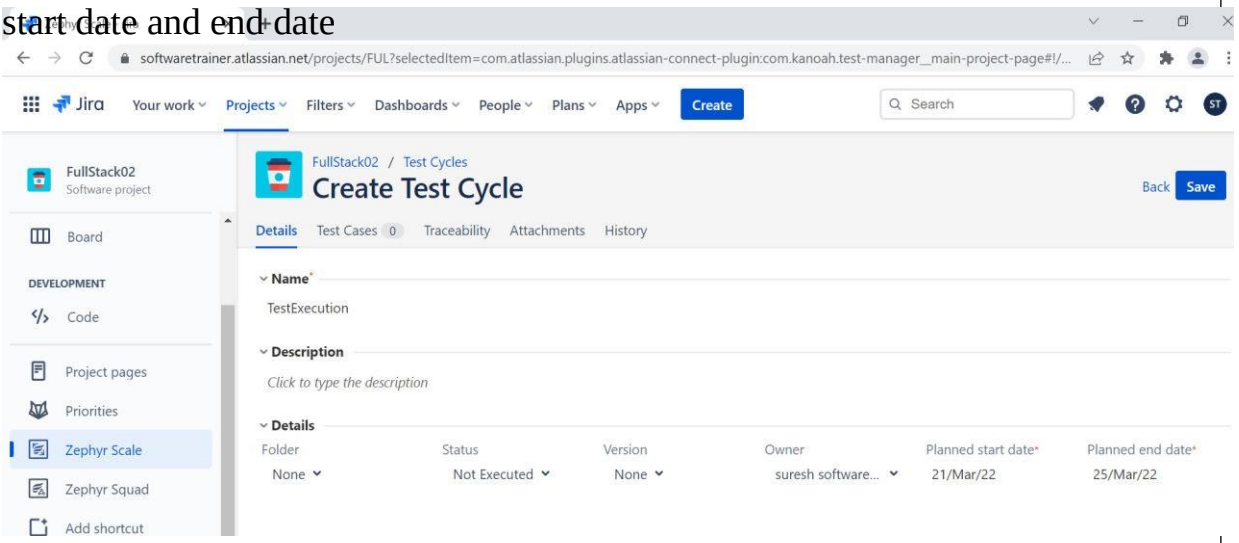
→Click on Save button & Continue in creating all the required test cases.

*****Test Execution Steps :**

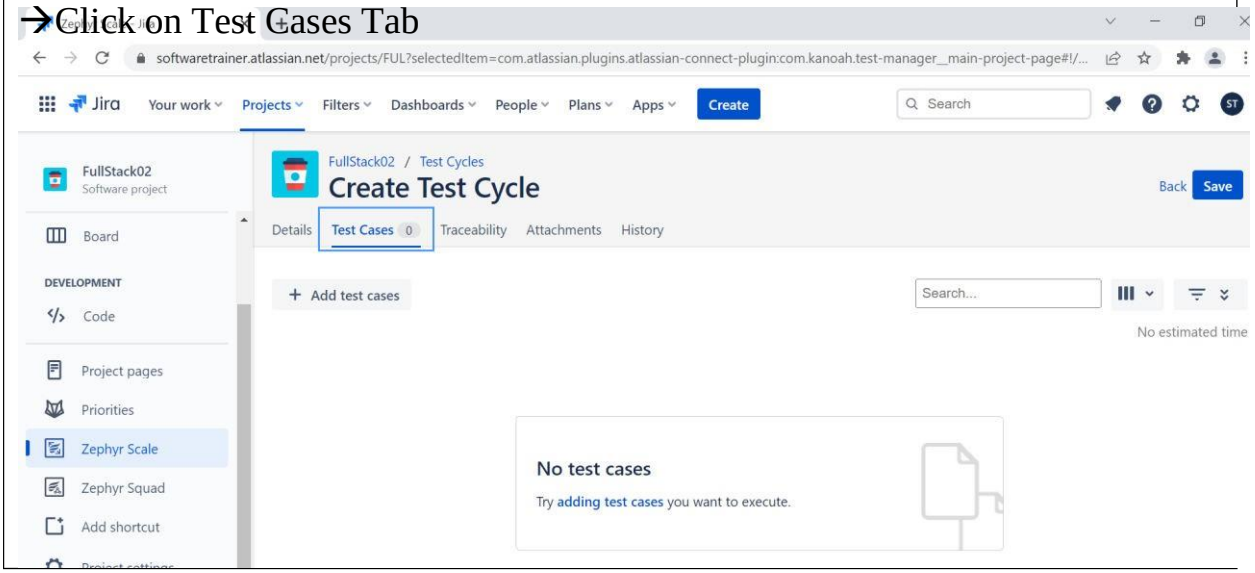
Click on Cycles button To create Test Cycle



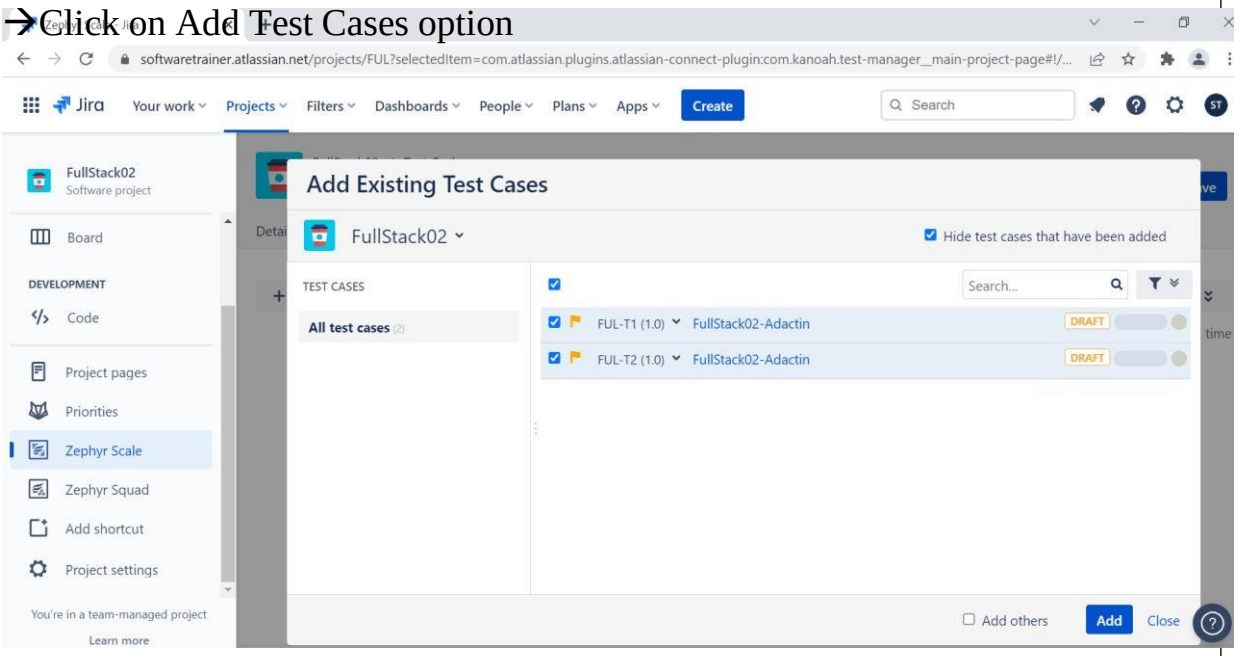
→Click on New Test Cycle option & Under Details tab provide Name,Owner , start date and end date



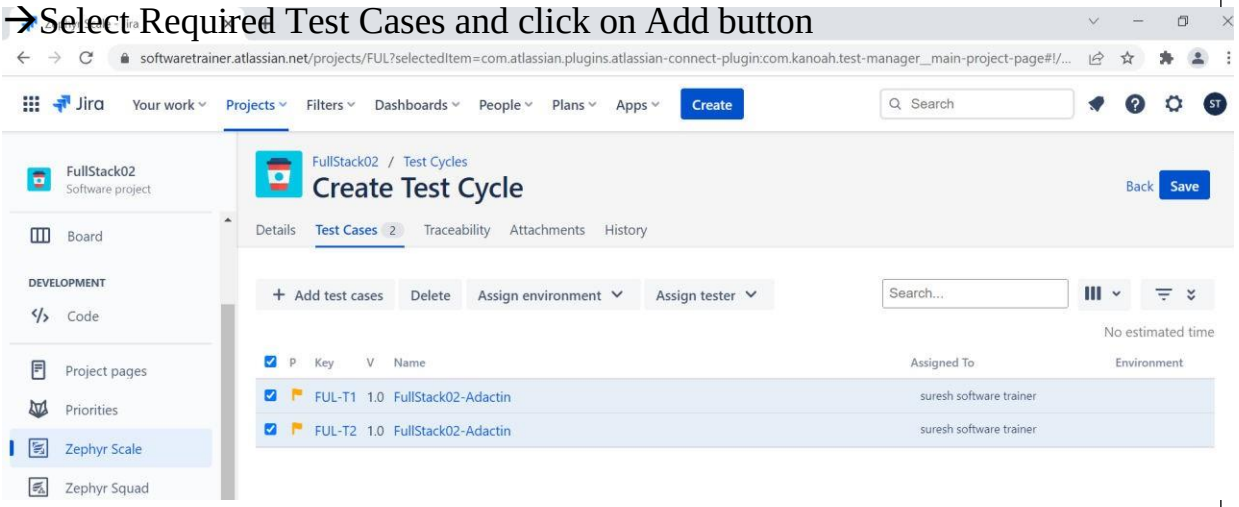
→Click on Test Cases Tab



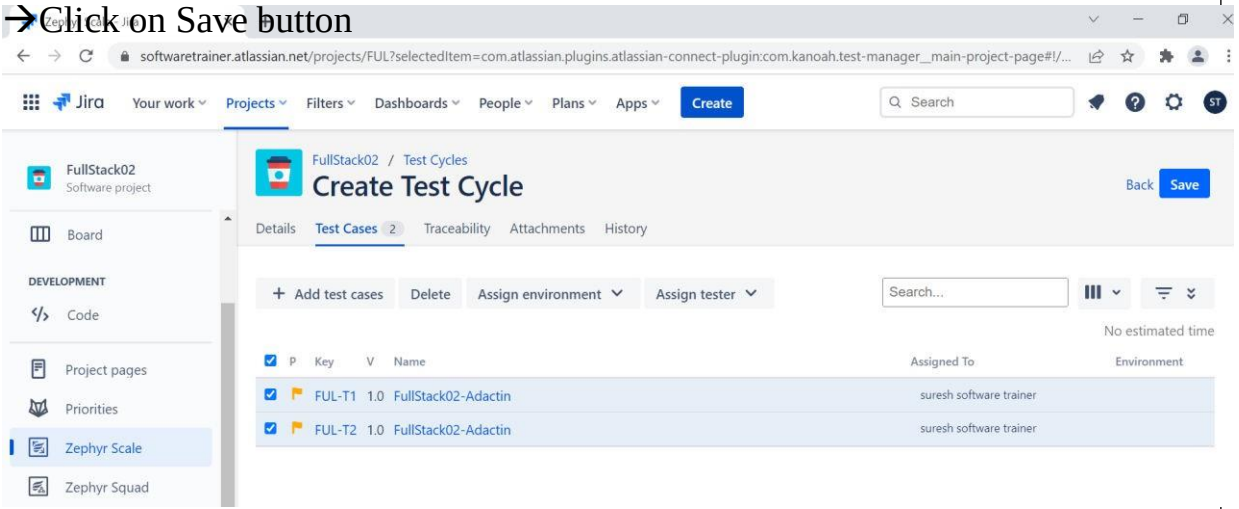
→Click on Add Test Cases option



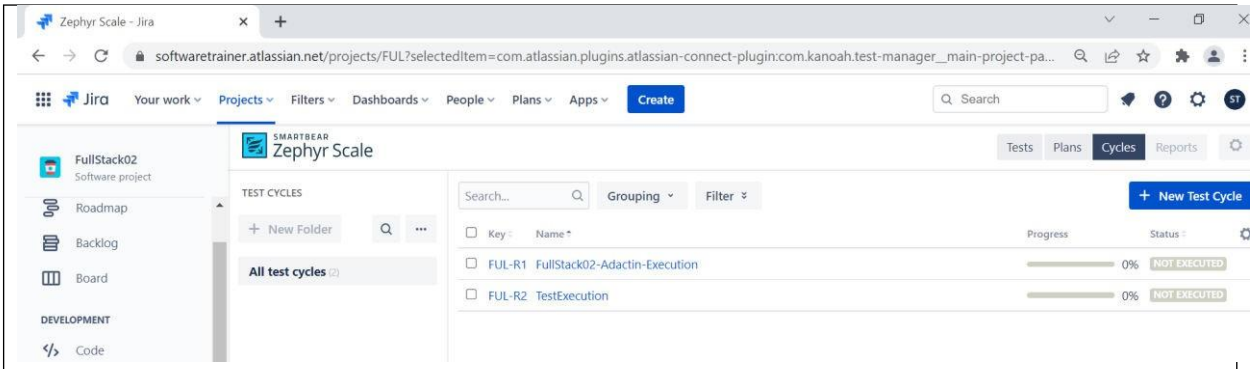
→Select Required Test Cases and click on Add button



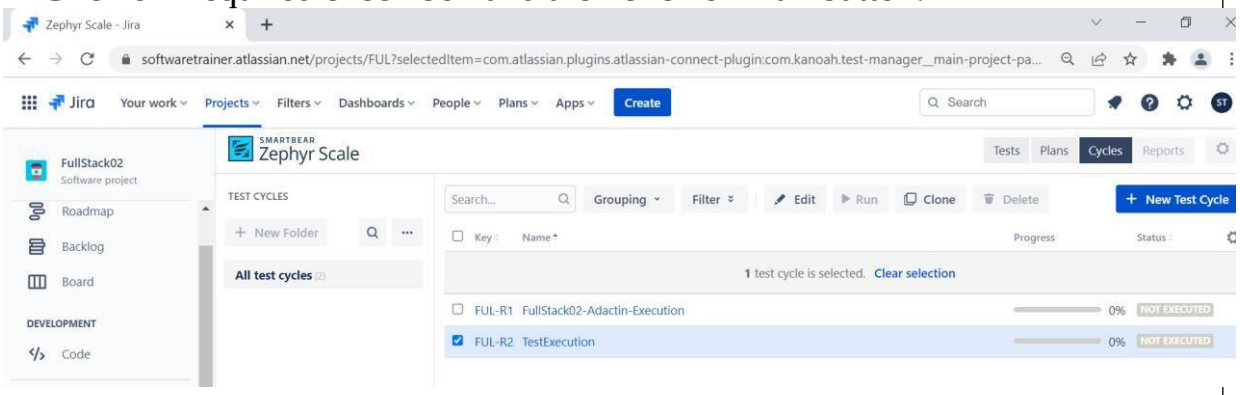
→Click on Save button



→click on Back Link



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



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).

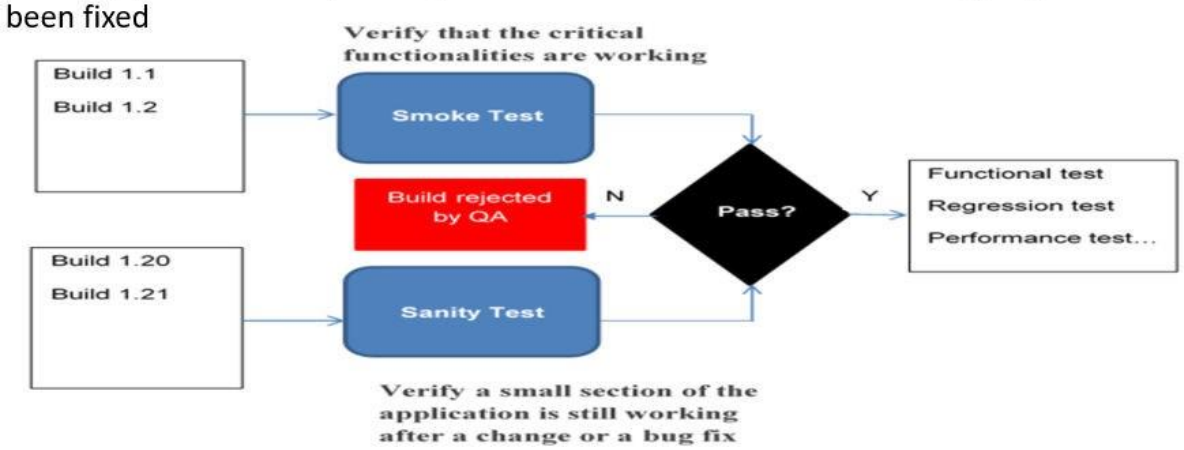
Types of Testing

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

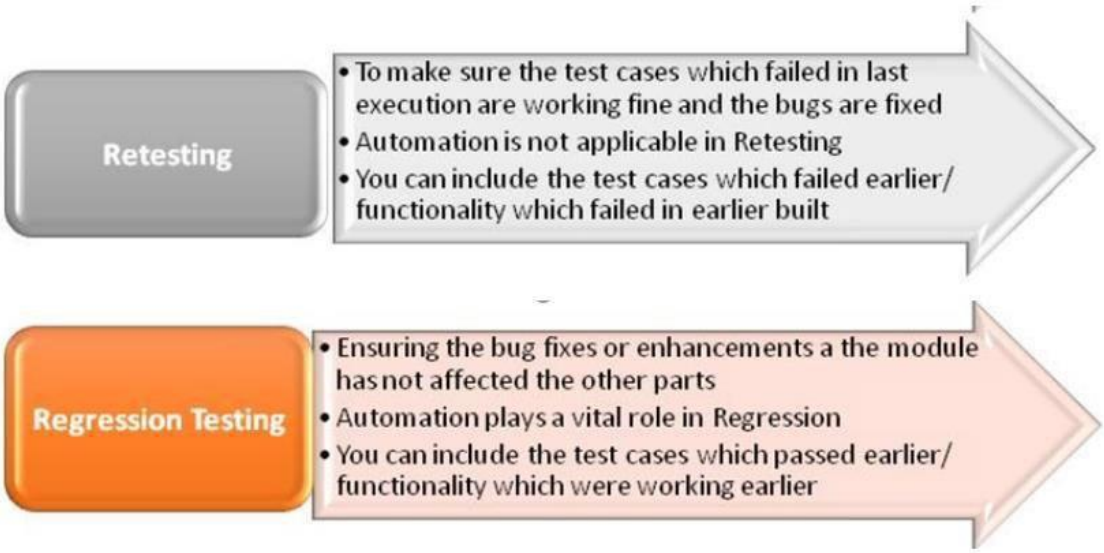
It is first level of testing on any newly released build to check main functionality of the application.

- **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 un affected whenever the new change is added



- **Static Testing:** Testing an application with out performing any action.

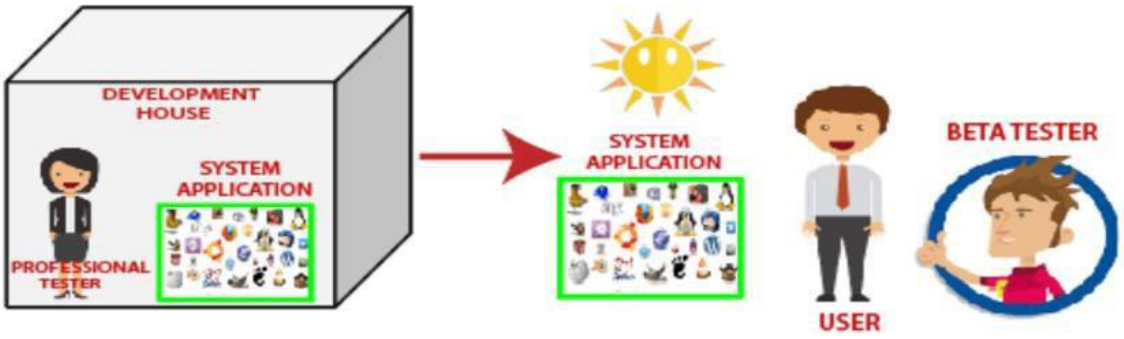
Eg: GUI Testing , colors ,spelling ,alignment … etc.

- **Dynamic Testing:** Testing an application by performing required actions.

Eg: Functionality Testing ,Textbox, button … etc

- **Ad-hoc Testing:** Testing the application with out any proper planning

- **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.



BETA TESTING OF THE PRODUCT IN REAL WORLD ENVIRONMENT

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

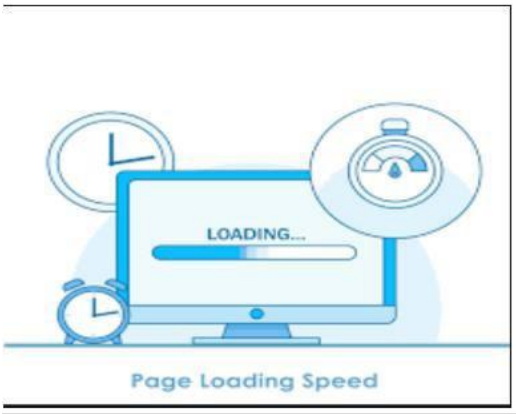
01. Application: Test requests are sent correctly to the Database and output at the client side is displayed correctly.

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/functionality/requirements reaches a specified point
- Bug rate falls below a certain level
- Alpha & Beta testing period ends