Test Plan Contents

- A Test Plan is a document that describes the test scope, test strategy, objectives, schedule, deliverables and resources required to perform testing for a software product.
- Test plan template contents:
  - Overview
  - Scope
    - Inclusions
    - Test Environments
    - Exclusions
  - Test Strategy
  - Defect Reporting Procedure
  - Roles/Responsibilities
  - Test Schedule
  - Test Deliverables
  - Pricing
  - Entry and Exit Criteria
  - Suspension and Resumption Criteria
  - Tools
  - Risks and Mitigations
  - Approvals

Use case, Test Scenario & Test Case

- Use Case:
  - Use case describes the requirement.
  - Use case contains THREE Items.
    - **Actor**, which is the user, which can be a single person or a group of people, interacting with a process.
    - **Action**, which is to reach the final outcome
    - **Goal/Outcome**, which is the successful user outcome.
- Test Scenario:
  - A possible area to be tested (What to test)
- Test Case:
  - Step by step actions to be performed to validate functionality of AUT (How to test).
  - Test case contains test steps, expected result & actual result.

Use Case V/s Test Case

- **Use Case** – Describes functional requirement, prepared by Business Analyst(BA).
- **Test Case** – Describes Test Steps/ Procedure, prepared by Test Engineer.

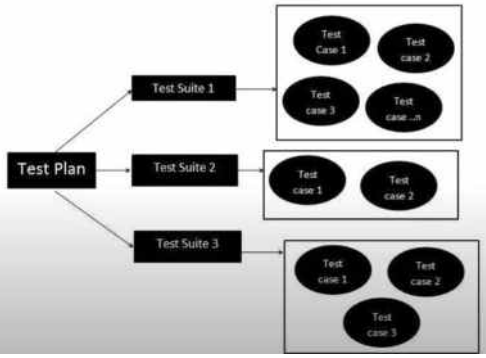
Test Scenario V/s Test Case

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

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

Test Suite

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



What is Test case?

- A Test Case is a set of actions executed to validate particular feature or functionality of your software application.

Test Case Contents

- Test Case ID
- Test Case Title
- Description
- Pre-condition
- Priority (P0, P1,P2,P3) – order
- Requirement ID
- Steps/Actions
- Expected Result
- Actual Result
- Test data

Test Case Template

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

Requirement Traceability Matrix(RTM)

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

Sample RTM

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

Test Environment

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

Test Execution

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

Guidelines for Test Execution

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

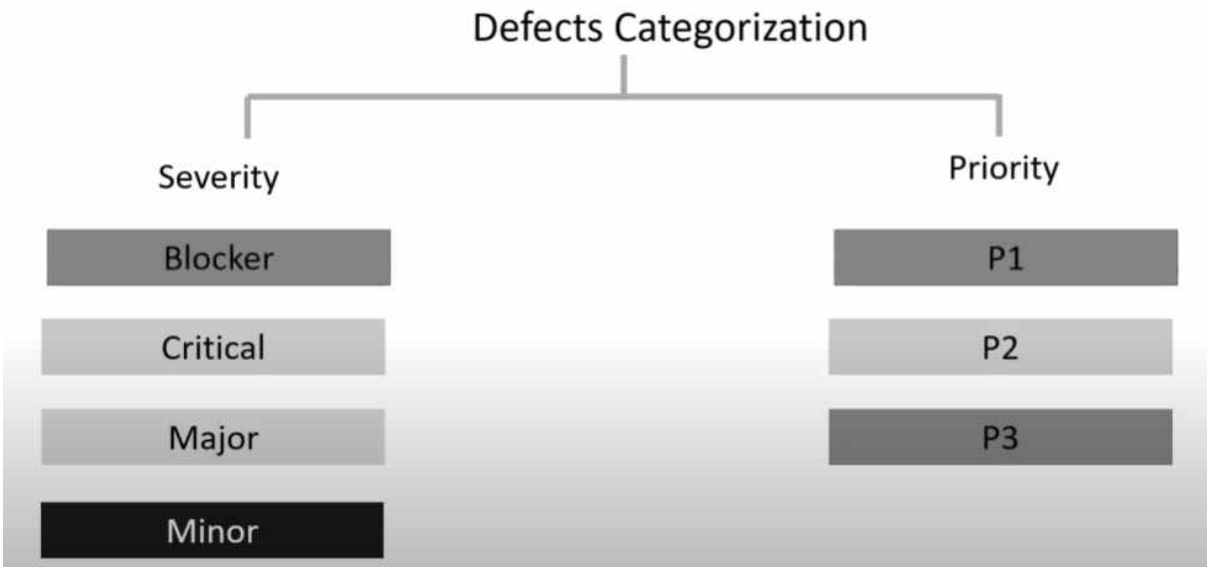
Defects/Bugs

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

Defect Report Contents

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

Defect Classification



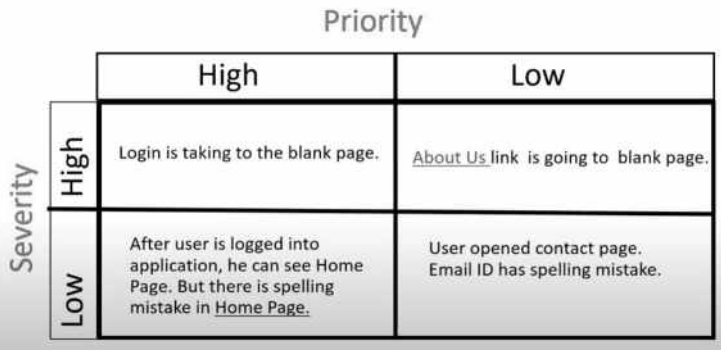
Defect Severity

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

Defect Priority

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

High severity, priority and low severity, priority defects



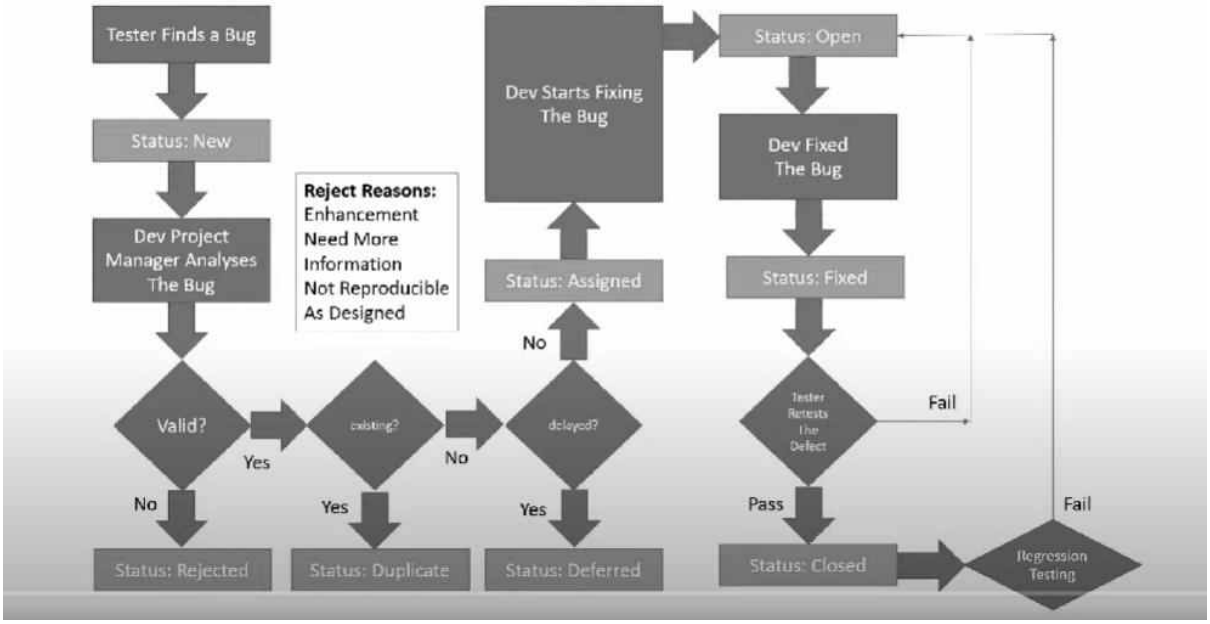
More examples...

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

Defect Resolution

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

Bug Life Cycle



Test Cycle Closure

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

Test Metrics

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

Test Metrics

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

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

QA/Testing Activities

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

7 Principles of Software Testing

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