Lecture 6

Testing Plan

What is a Test Plan?

A test plan is a document that consists of all future testing-related activities.

It is prepared at the project level and in general, it defines work products to be tested, how they will be tested, and test type distribution among the testers.
Before starting testing there will be a test manager who will be preparing a test plan. In any company whenever a new project is taken up before the tester is involved in the testing the test manager of the team would prepare a test Plan.

- The test plan serves as the blueprint that changes according to the progressions in the project and stays current at all times.
- It serves as a base for conducting testing activities and coordinating activities among a QA team.
- It is shared with Business Analysts, Project Managers, and anyone associated with the project.

A test plan is a comprehensive document outlining all testing-related activities for a project. It details what will be tested, how, and by whom, serving as a blueprint for testing and coordinating among the QA team.

What is a Test Plan?

Factors	Roles
Who writes Test Plans?	Test lead, Test Manager, Test Engineer
Who reviews the Test Plan?	Test Lead, Test Manager, Test Engineer, Customer, Development Team
Who approves the Test Plan?	Customer, Test Manager
Who writes Test Cases?	Test Lead, Test Engineer
Who reviews Test Cases?	Test Engineer, Test Lead, Customer, Development Team
Who approves Test Cases?	Test Manager, Test Lead, Customer

Why is Test Plan creation important?

- **Defines Objectives:** A test plan clearly outlines the testing objectives and the scope of testing activities, ensuring that all team members understand what needs to be achieved.
- **Structured Approach** : It provides a systematic approach to testing, detailing the steps and processes involved, which helps in organizing the testing effort.
- **Avoids Scope Creep** : By defining what will and will not be tested, the test plan helps manage the scope of testing activities, preventing unnecessary work and focusing on irrelevant areas.
- **Resource Allocation** : Helps in identifying the necessary resources, including personnel, tools, and environments, ensuring they are available when needed.
- **Identifies Risks** : A test plan identifies potential risks and outlines mitigation strategies, helping to address issues proactively rather than reactively.
- **Contingency Plans** : These include contingency plans for dealing with unexpected events or issues that may arise during testing.

Why is Test Plan creation important?

- **Stakeholder Alignment** : Facilitates communication among stakeholders, including developers, testers, project managers, and clients, ensuring everyone is aligned on the testing objectives, approach, and schedule.
- **Documentation** : Serves as a comprehensive document that can be referred to by all team members, aiding in knowledge sharing and transparency.
- **Resource Optimization** : Helps in efficiently utilizing available resources, including time and personnel, by providing a clear plan of action.
- **Focus on Priorities** : Ensures that testing efforts are focused on high-priority areas that are critical to the success of the project.

Objectives of the Test Plan

Overview of testing activities: provides an overview of the testing activities and where to start and stop the work.

Provides timeline: helps to create the timeline for the testing activities based on the number of hours and the workers needed.

Helps to estimate resources: helps to create an estimate of the number of resources needed to finish the work.

Serves as a blueprint: serves as a blueprint for all the testing activities, it has every detail from beginning to end.

Helps to identify solutions: helps the team members They consider the project's challenges and identify the solutions.

Serves as a rulebook: serves as a rulebook for following rules when the project is completed phase by phase.

Components and Attributes of Test Plan

Components and Attributes of Test Plan

1. Objective: It describes the aim of the test plan, whatever the good process and procedure they are going to follow to give quality software to customers. The overall objective of the test is to find as many defects as possible and to make software bug-free. The test objective must be broken into components and sub-components. In every component following activities should be performed.

- List all the functionality and performance to be tested.
- Make goals and targets based on the application feature.

2. Scope: It consists of information that needs to be tested concerning an application. The scope can be divided into two parts:`

In-Scope: The modules that are to be tested precisely.

Out Scope: The modules that are not to be tested precisely.

Example: In an application A, B, C, and D features have to be developed, but the B feature has already been designed by other companies. So the development team will purchase B from that company and perform only integrated testing with A, B, and C.

Components and Attributes of Test Plan

3. Testing Methodology: The methods that are going to be used for testing the application.
The testing methodology is decided based on the feature and application requirements.
Since the testing terms are not standard, one should define what kind of testing will be used in the testing methodology. So that everyone can understand it.

4. Approach: The approach of testing different software is different. It deals with the flow of applications for future reference. It has two aspects:

High-Level Scenarios: For testing critical features high-level scenarios are written. For Example, login to a website, and book from a website.

The Flow Graph: It is used when one wants to make benefits such as converging and merging easy.

Components and Attributes of Test Plan

5. Assumption: In this phase, certain assumptions will be made.

Example:

The testing team will get proper support from the development team.

The tester will get proper knowledge transfer from the development team.

Proper resource allocation will be given by the company to the testing department.

6. Risk: All the risks that can happen if the assumption is broken. For Example, in the case of wrong budget estimation, the cost may overrun. Some reason that may lead to risk is:

Test Manager has poor management skills.

Hard to complete the project on time.

Lack of cooperation.

Components and Attributes of Test Plan

7. Mitigation Plan: If any risk is involved then the company must have a backup plan, the purpose is to avoid errors. Some points to resolve/avoid risk:

Test priority is to be set for each test activity.

Managers should have leadership skills.

Training course for the testers.

8. Roles and Responsibilities: All the responsibilities and role of every member of a particular testing team has to be recorded.

Example:

Test Manager: Manages the project, takes appropriate resources, and gives project direction.

Tester: Identify the testing technique, verify the test approach, and save project costs.

.

Components and Attributes of Test Plan

9. Schedule: Under this, it will record the start and end date of every testing-related activity. For Example, writing the test case date and ending the test case date.

10. Defect Tracking: It is an important process in software engineering as lots of issue arises when you develop a critical system for business. If there is any defect found while testing that defect must be given to the developer team. There are the following methods for the process of defect tracking:

Information Capture: In this, we take basic information to begin the process.

Prioritize: The task is prioritized based on severity and importance.

Communication: Communication between the identifier of the bug and the fixer of the bug.

Environment: Test the application based on hardware and software.

Example: The bug can be identified using bug-tracking tools such as Jira, Mantis, and Trac.

Components and Attributes of Test Plan

11. Test Environments: It is the environment that the testing team will use i.e. the list of hardware and software, while testing the application, the things that are said to be tested will be written under this section. The installation of software is also checked under this.

Example:

Software configuration on different operating systems, such as Windows, Linux, Mac, etc.

Hardware Configuration depends on RAM, ROM, etc.

Components and Attributes of Test Plan

12. Entry and Exit Criteria: The set of conditions that should be met to start any new type of testing or to end any kind of testing.

Entry Condition:

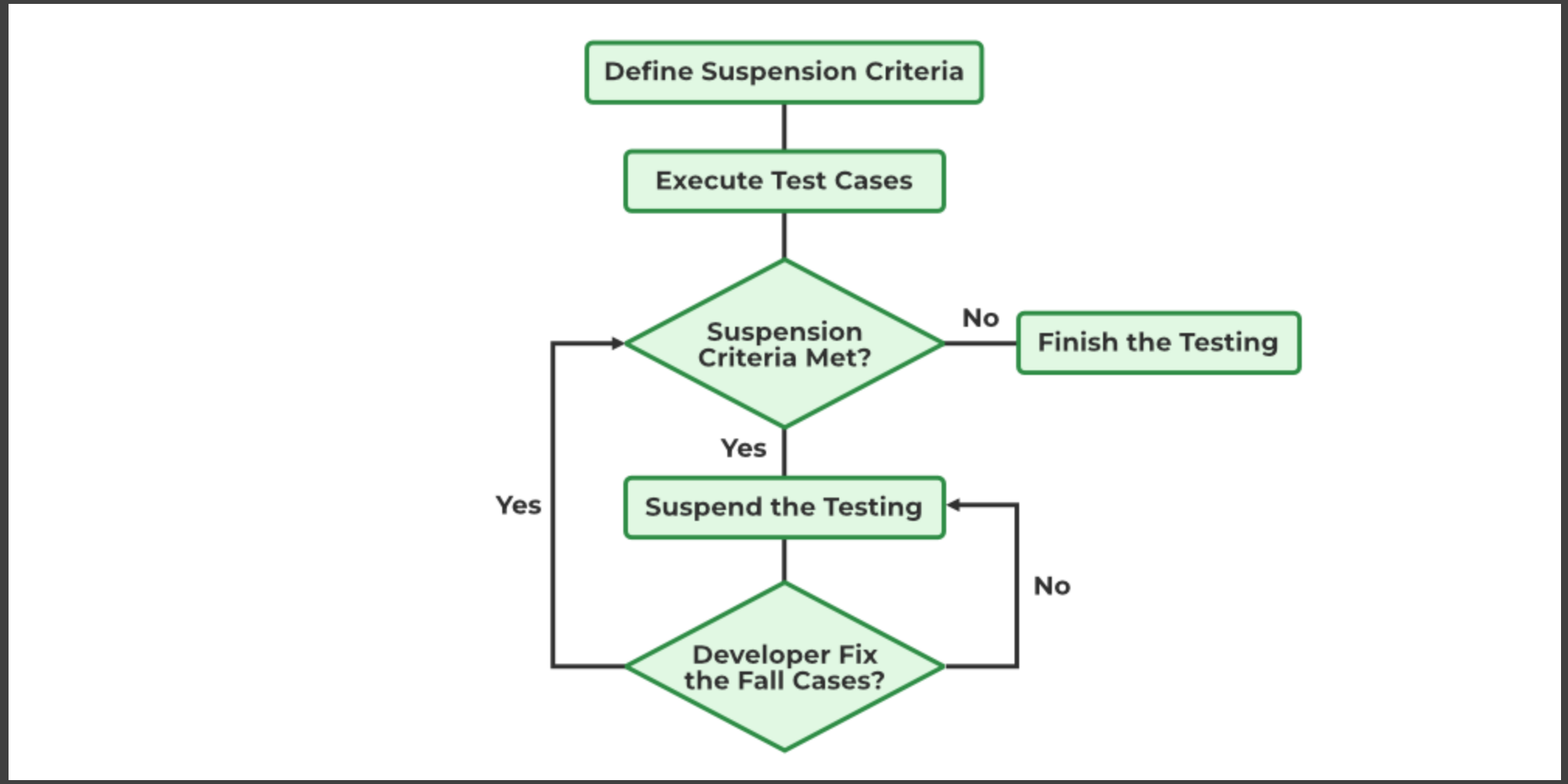
- Necessary resources must be ready.
- The application must be prepared.
- Test data should be ready.

Exit Condition:

- There should not be any major bugs.
- Most test cases should be passed.
- When all test cases are executed.

Example: If the team member reports that 45% of the test cases failed, then testing will be suspended until the developer team fixes all defects.

Components and Attributes of Test Plan



Components and Attributes of Test Plan

13. Test Automation: It consists of the features that are to be automated and which features are not to be automated.

If the feature has lots of bugs then it is categorized as Manual Testing.

If the feature is frequently tested then it can be automated.

14. Effort Estimation: This involves planning the effort that needs to be applied by every team member.

Components and Attributes of Test Plan

15. Test Deliverables: It is the outcome from the testing team that is to be given to the customers at the end of the project.

Before the testing phase :

- Test plan document.
- Test case document.

During the testing phase :

- Test scripts.
- Test data.
- Error logs.

After the testing phase :

- Test Reports.
- Defect Report.
- Installation Report.

It contains a test plan, defect report, automation report, assumption report, tools, and other components that have been used for developing and maintaining the testing effort.

Components and Attributes of Test Plan

16. Template: This is followed by every kind of report that is going to be prepared by the testing team. All the test engineers will only use these templates in the project to maintain the consistency of the product.

Types of Test Plans:

- **Master Test Plan:** In this type of test plan, includes multiple test strategies and has multiple levels of testing. It goes into great depth on the planning and management of testing at the various test levels and thus provides a bird's eye view of the important decisions made, tactics used, etc. It includes a list of tests that must be executed, test coverage, the connection between various test levels, etc.
- **Phase Test Plan:** In this type of test plan, emphasis is on any one phase of testing. It includes further information on the levels listed in the master testing plan. Information like testing schedules, benchmarks, activities, templates, and other information that is not included in the master test plan is included in the phase test plan.
- **Specific Test Plan:** This type of test plan, is designed for specific types of testing especially non-functional testing for example plans for conducting performance tests or security tests.

How to create a Test Plan

1. Analyze the product: This phase focuses on analyzing the product, Interviewing clients, designers, and developers, and performing a product walkthrough. This stage focuses on answering the following questions:

- What is the primary objective of the product?
- Who will use the product?
- What are the hardware and software specifications of the product?
- How does the product work?

2. Design the test strategy: The test strategy document is prepared by the manager and details the following information:

- Scope of testing which means the components that will be tested and the ones that will be skipped.
- Type of testing which means different types of tests that will be used in the project.
- Risks and issues that will list all the possible risks that may occur during testing.
- Test logistics mentions the names of the testers and the tests that will be run by them.

How to create a Test Plan

3. Define test objectives: This phase defines the objectives and expected results of the test execution.

Objectives include:

A list of software features like functionality, GUI, performance standards, etc.

The ideal expected outcome for every aspect of the software that needs testing.

4. Define test criteria: Two main testing criteria determine all the activities in the testing project:

- **Suspension criteria:** Suspension criteria define the benchmarks for suspending all the tests.
- **Exit criteria:** Exit criteria define the benchmarks that signify the successful completion of the test phase or project.
  These are expected results and must match before moving to the next stage of development.

5. Resource planning: This phase aims to create a detailed list of all the resources required for project completion.
For example, human effort, hardware and software requirements, all infrastructure needed, etc.

How to create a Test Plan

6. Plan test environment: This phase is very important as the test environment is where the QAs run their tests. The test environments must be real devices, installed with real browsers and operating systems so that testers can monitor software behaviour in real user conditions.

7. Schedule and Estimation: Break down the project into smaller tasks and allocate time and effort for each task. This helps in efficient time estimation. Create a schedule to complete these tasks in the designated time with a specific amount of effort.

8. Determine test deliverables: Test deliverables refer to the list of documents, tools, and other equipment that must be created, provided, and maintained to support testing activities in the project.

Best Practices for Creating an effective Test Plan:

Creating an effective test plan is essential for ensuring a comprehensive and systematic approach to software testing.
Here are some best practices to consider when developing a test plan:

1. Understand the Project Requirements:

Gather Information: Ensure a thorough understanding of both functional and non-functional requirements.

Stakeholder Input: Involve stakeholders to gather expectations and address specific concerns.

2. Define Clear Objectives and Scope:

Purpose of Testing : Clearly state the objectives and what you aim to achieve.

In-Scope and Out-of-Scope: Define what will be tested and what will not, to manage expectations and focus resources.

Best Practices for Creating an effective Test Plan:

3. Develop a Comprehensive Test Strategy:

Approach : Outline the types of testing to be performed (e.g., functional, regression, performance).

Techniques and Tools: Specify testing techniques (e.g., black-box, white-box) and tools (e.g., Selenium, JIRA) to be used.

4. Create Detailed Test Cases:

Test Case Design : Develop detailed test cases covering all scenarios, including positive, negative, edge, and boundary cases.

Traceability : Ensure each test case is traceable to specific requirements to verify comprehensive coverage.

Best Practices for Creating an effective Test Plan:

5. Establish a Test Environment:

Setup Requirements : Define hardware, software, network configurations, and tools required for testing.

Environment Management: Ensure the test environment closely mirrors the production environment to identify environment-specific issues.

6. Plan for Test Data and Reporting Mechanisms:

Data Requirements : Identify and manage realistic, consistent test data securely, especially if it includes sensitive information.

Status Reporting: Establish processes for regular status updates on testing progress, issues, and results, and use defect tracking systems effectively.

Best Practices for Creating an effective Test Plan:

Conclusion:

A test plan is a crucial document in the software testing lifecycle that provides a structured approach to validating and verifying the quality of a software product. It outlines the objectives, scope, resources, and methodologies for testing, ensuring that all aspects of the application are thoroughly assessed. By following best practices in test plan creation, such as understanding project requirements, defining clear objectives, and establishing a robust test environment, teams can effectively manage testing efforts and enhance the overall quality of the software. A well-crafted test plan not only aligns the team on testing goals but also helps in optimizing resources, mitigating risks, and ensuring stakeholder satisfaction.

Lecture 6

Test Case

What is a Test Case?

A test case is a defined format for software testing required to check if a particular application/software is working or not.

A test case consists of a certain set of conditions that need to be checked to test an application or software.

in more simple terms when conditions are checked it checks if the resultant output meets with the expected output or not.

A test case consists of various parameters such as ID, condition, steps, input, expected result, result, status, and remarks.

Parameters of a Test Case:

Module Name: Subject or title that defines the functionality of the test.

Test Case Id: A unique identifier assigned to every single condition in a test case.

Tester Name: The name of the person who would be carrying out the test.

Test scenario: **The test** scenario provides a brief description to the tester, as in providing a small overview to know about what needs to be performed and the small features, and components of the test.

Test Case Description: The condition required to be checked for a given software. for eg. Check if only numbers validation is working or not for an age input box.

Test Steps: Steps to be performed for the checking of the condition.

Prerequisite: The conditions required to be fulfilled before the start of the test process.

Test Priority: As the name suggests gives priority to the test cases that had to be performed first, or are more important and that could be performed later.

Test Data: The inputs to be taken while checking for the conditions.

Parameters of a Test Case:

Test Expected Result: The output which should be expected at the end of the test.

Actual Result: The output that is displayed at the end.

Environment Information: The environment in which the test is being performed, such as the operating system, security information, the software name, software version, etc.

Status: The status of tests such as pass, fail, NA, etc.

Comments: Remarks on the test regarding the test for the betterment of the software.

Just as a well-structured test case is essential for validating and verifying the functionality of software

When do we Write Test Cases?

Before development: Test cases could be written before the actual coding as that would help to identify the requirement of the product/software and carry out the test later when the product/software gets developed.

After development: Test cases are also written directly after coming up with a product/software or after developing the feature but before the launching of a product/software as needed to test the working of that particular feature.

During development: Test cases are sometimes written during the development time, parallelly. so whenever a part of the module/software gets developed it gets tested as well.

So, test cases are written in such cases, as test cases help in further development and make sure that we are meeting all the needed requirements.

Why Write Test Cases?

Test cases are very important aspects of software engineering, Test cases are carried out for a very simple reason, to check if the software works or not. There are many advantages of writing test cases:

To check whether the software meets customer expectations: Test cases help to check if a particular module/software is meeting the specified requirement or not.

To check software consistency with conditions: determine if a particular module/software works with a given set of conditions.

Narrow down software updates: help to narrow down the software needs and required updates.

Better test coverage: help to make sure that all possible scenarios are covered and documented.

For consistency in test execution: help to maintain consistency in test execution. A well-documented test case helps the tester to just have a look at the test case and start testing the application.

Helpful during maintenance: Test cases are detailed which makes them helpful during the maintenance phase.

Test Case Template

Fields	Description
Test Case ID	Each test case should have a unique ID.
Test Case Description	Each test case should have a proper description to let testers know what the test case is about.
Pre-Conditions	Conditions that are required to be satisfied before executing the test case.
Test Steps	Mention all test steps in detail and to be executed from the end-user's perspective.
Test Data	Test data could be used as input for the test cases.
Expected Result	The result is expected after executing the test cases.
Post Condition	Conditions need to be fulfilled when the test cases are successfully executed.
Actual Result	The result that which system shows once the test case is executed.
Status	Set the status as Pass or Fail on the expected result against the actual result.

Test Case Template

Project Name	Name of the project to which the test case belongs.
Module Name	Name of the module to which the test case belongs.
Reference Document	Mention the path of the reference document.
Created By	Name of the tester who created the test cases.
Date of Creation	Date of creation of test cases.
Reviewed By	Name of the tester who reviews the test case.
Date of Review	When the test cases were reviewed.
Executed By	Name of the tester who executed the test case.
Date of Execution	Date when the test cases were executed.
Comments	Include comments which help the team to understand the test cases.

Test Case Template

Module Name:-	Login
Test Case ID	gfg_01
Tester Name	Geek
Test Case Description	To check login functionality of geeksforgeeks website.
Prerequisites:	1. Stable internet connection. 2. Browser(Chrome, Firefox, internet explorer, etc.)
Tester's Name	Geek
Enviromental Information:-	1. OS: Windows/Linux/Mac 2. System: Laptop/Desktop

Test Scenario	Checking that after entering the correct username and password, the user can login.

Test Case ID	Test Steps	Test Input	Expected Results	Actual Results	Status	Comments
1.	1. Enter Username	Username: geeksforgeeks	Welcome to geeksforgeeks!	Welcome to geeksforgeeks!	Pass	No issues found.
	2. Enter Password	Password: geek123				
	3. Click on login					

Best Practice for Writing Test Case

Simple and clear: Test cases need to be very concise, clear, and transparent. They should be easy and simple to understand not only for oneself but for others as well.

Maintaining the client/customer/end-user requirements must be unique **:** While writing the test cases, it's necessary to make sure that they aren't being written over and over again and that each case is different from the others.

Zero Assumptions: Test cases should not contain assumed data, and don't come up with features/modules that don't exist.

Traceability: Test cases should be traceable for future reference, so while writing it's important to keep that in mind,

Different input data: While writing test cases, all types of data must be taken into consideration.

Strong module name: The module name should be self-explanatory while writing the test case.

Minimal Description: The description of a test case should be small, one or two lines are normally considered good practice, but it should give the basic overview properly.

Maximum conditions: All kinds of conditions should be taken into consideration while writing a test, increasing the effectiveness.

Best Practice for Writing Test Case

Meeting requirements: While writing the test case the stakeholder requirements must be met.

Repetitive Results: The test case must be written in such a way that it should provide the same result.

Different Techniques: Sometimes testing all conditions might not be possible but using different testing with different test cases could help to check every aspect of a software.

Create test cases with the end user's perspective: Create test cases by keeping end-user in mind and the test cases must meet customer requirements.

Use unique Test Case ID: It is considered a good practice to use a unique Test Case ID for the test cases following a naming convention for better understanding.

Add proper preconditions and postconditions: Preconditions and postconditions for the test cases must be mentioned properly and clearly.

Test cases should be reusable: There are times when the developer updates the code, then the testers need to update the test cases to meet the changing requirements.

Specify the exact expected outcome: Include the exact expected result, which tells us what will be result of a particular test step.