DAY-6

WHAT is TEST PLAN:

A test plan is a document detailing the objectives, resources, and processes for a specific test session for a software or hardware product.

it defines work products to be tested, how they will be tested, and test type distribution among the testers.

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.

WHY test plans are important?

Quick guide for the testing process: The test plan serves as a quick guide for the testing process as it offers a clear guide for QA engineers to conduct testing activities.

Helps to avoid out-of-scope functionalities: The test plan offers detailed aspects such as test scope, test estimation, strategy, etc.

Helps to determine the time, cost, and effort: The Test serves as the blueprint to conduct testing activities thus it helps to deduce an estimate of time, cost, and effort for the testing activities.

Provide a schedule for testing activities: A test plan is like a rule book that needs to be followed, it thus helps to schedule activities that can be followed by all the team members.

Test plan can be reused: The test plan documents important aspects like test estimation, test scope, and test strategy which are reviewed by the Management Team and thus can be reused for other projects.

Steps to create test plan:

1. Define the release scope: it’s important to define the scope of testing for your release. This means defining the features or functions that need to

be included in the release.

2. Schedule timelines: Specify release deadlines to help you decide your testing time and routine.

3. Define test objectives: A test objective is a reason or purpose for designing and executing a test. These objectives ultimately help guide and define.

the scope of testing activities.

4. Determine test deliverables: Test deliverables are the products of testing that help track testing progress. Deliverables should meet your project’s

and client’s needs, be identified early enough to be included in the test plan, and be scheduled accordingly.

5. Design the test strategy: Test strategy helps determine test cost, test effort, and which features will be in-scope (planned to be tested) versus

out-of-scope (not planned to be tested).

6. Plan test environment and test data: Planning a test environment guarantees precise and robust testing. The test environment includes hardware, software, and network configurations for software testing.

Components and Attributes of a test plan:



=>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. Find as many defects for bug free software.

List all the functionality and performance to be tested.

Make goals and targets based on the application feature.

=> Scope: It consists of information that needs to be tested concerning an application.

IN-SCOPE: describes what is being tested, like all functionalities/features of a specific project/product/solution.

OUT-OF-SCOPE: identify all the features and combinations of features which will not be tested and the reasons.

=>Test Methodology: The methods that are going to be used for testing depend on application to application.

=>Approach:

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.

=>Assumption:

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.

=>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.

=>Mitigation Plan: If any risk is involved then the company must have a backup plan, the purpose is to avoid errors.

=>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.

=> 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.

=> 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.

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

Hardware Configuration depends on RAM, ROM, etc.

=>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.

=>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.

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

=> Test Deliverables: Itis the outcome from the testing team that is to be given to the customers at the end of the project.

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 i.e., 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.

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 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.

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.

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.

Meeting requirements**:**While writing the test case the client/customer/end-user requirements must be met.

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.

HOW TO WRITE TEST CASES:

Step#1: Test Case ID:

Each test case should be represented by a unique ID. It’s good practice to follow some naming convention for better understanding and discrimination purposes.

Step#2: Test Case Description:

Pick test cases properly from the test scenarios.

Example: Test scenario: Verify the login of Gmail  
Test case: Enter a valid username and valid password

Step#3-Pre-Conditions: Conditions that need to meet before executing the test case. Mention if any preconditions are available.

Ex: Need a valid Gmail account to do login

Step#4-Test Steps: execute test cases, you need to perform some actions. So write proper test steps. Mention all the test steps in detail and in the order how it could be executed from the end-user’s perspective.

Ex: Enter Username

Enter Password

Click login button

Step#5-Test Data: You need proper test data to execute the test steps. So gather appropriate test data. The data which could be used an input for the test cases.

Ex: username:abc@gmail.com

Password:123456

Step#6-Expected Result: The result which we expect once the test cases were executed. It might be anything such as Home Page, Relevant screen, Error message, etc.,

Ex: Successful Login

Step#7-Post Condition: Conditions that need to achieve when the test case was successfully executed.

Ex:  Gmail inbox is shown

Step#8-Actual Result: The result which system shows once the test case was executed. Capture the result after the execution. Based on this result and the expected result, we set the status of the test case.

Ex: Redirected to Gmail inbox

Step#9-Status:

Finally set the status as Pass or Fail based on the expected result against the actual result. If the actual and expected results are the same, mention it as Passed. Else make it as Failed.

Verification testing and Validation testing:

Verification Testing: A test of a system to prove that it meets all its specified requirements at a particular stage of its development.

Advantages of Verification Testing:

1. Early and frequent verification reduces the number of bugs and defects that may show up in later stages.

2. By verifying at each stage, dev’s, product managers can get more insight into what the product may need to be developed better in the coming stages.

3. Even if all the bugs can't be solved immediately, verifying helps estimate the issues and help them better prepare to handle those when they appear.

4. Verification helps keep software closely aligned with customers and business requirements at every stage.

Validation Testing: An activity that ensures that an end product stakeholder’s true needs and expectations are met.

validation testing occurs at the end of a specific module or even after the software has been entirely built. Its primary intent is to ensure the

final product matches the stakeholder and customer requirements.

Advantages of Validation Testing:

1. Any bugs missed during verification will be detected while running validation tests.

2. If specifications were incorrect and inadequate, validation tests would reveal their inefficacy.

3. Validation tests ensure that the product matches and adheres to customer demands, preferences, and expectations under different conditions.

4. These tests are also required to ensure the software functions flawlessly across different browser-device-OS combinations.

What is release cycle?

Release cycle  is a sequence of phases that a software product undergoes from its initial development to its public distribution.

what is sprint cycle?

A time-boxed period (usually 2-4 weeks) during which a potentially shippable product increment is created.

Difference b/w sprint cycle and release cycle

Time frame:

Sprint cycle is a short-term, iterative cycle with a fixed duration (e.g., two weeks).

Release cycle is a longer-term plan covering the timeline for delivering major versions of the software.

Scope:

Sprint cycle focuses on delivering a small, incremental piece of functionality within a short timeframe.

Release cycle focuses on delivering a more comprehensive and stable version of the software that may include multiple features developed over several sprints.

Frequency:

Sprint cycle focuses on delivering a small, incremental piece of functionality within a short timeframe.

Release cycle focuses on delivering a more comprehensive and stable version of the software that may include multiple features developed over several sprints.

Positive Testing:

 Positive testing is a type of software testing that is performed by assuming everything will be as expected. It is performed with the assumption that only valid and relevant things will occur.

Negative Testing:

Negative testing is a type of software testing that is performed to check the system for unexpected conditions.

plays a much important role in high-performance software development.

BURNDOWN CHART:

A burndown chart shows the amount of work that has been completed in an epic or sprint,

and the total work remaining.

The chart is updated to reflect progress and the project’s current status, and you’ll be able to

estimate when the project will be complete. This helps teams plan for deadlines and

determine whether they will meet them.

