K.T.S.P.Madal's

Hutatma Rajguru Mahavidyalaya, Rajgurunagar

Tal-Khed, Dist.-Pune 410505.

TY.BSc (Computer Science)

Semester-VI

Subject- Software Testing Tools

According to new CBCS syllabus w.e.f.2019-2020

Prof.S.V.Patole

Department of Computer Science

Hutatma Rajguru Mahavidyalaya,

Rajgurunagar.

Unit 3. Test Cases and Test Plan

Definition

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.

Need Of Test Plan Document

A **Test Plan** is a detailed document that describes the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test manager.

Preparing a Test Plan

Step #1. Product Analysis

Requirements Analysis:

-
    - Initiate the test plan creation process by conducting a comprehensive analysis of software requirements. This forms the foundation for subsequent testing phases.

System Analysis:

-
    - Prioritize thorough system analysis to gain a holistic understanding of the software's architecture, functionalities, and interactions.

Website and Documentation Review:

-
    - Scrutinize the website and product documentation to extract detailed insights into software features, configurations, and operational procedures.

Stakeholder Interviews:

-

    - Engage in interviews with key stakeholders, including owners, end-users, and developers, to garner diverse perspectives and nuanced insights into the software.

Client Research:

-
    - Conduct in-depth research on the client, end-users, and their specific needs and expectations. Understand the client's business objectives and how the software aligns with those goals.

Key Questions for Analysis:

-
    - Pose critical questions to guide the analysis process:
        - What is the intended purpose of the system?
        - How will the system be utilized?
        - Who are the end-users, and how will they interact with the system?
        - What are the development requirements for implementing the system effectively?

Clarification Interviews:

- If any aspect of the system's requirements remains unclear, conduct interviews with clients and relevant team members for detailed clarification.

Step #2. Designing test strategy

Definitely, the scope of the testing is very important. To put it in simple words, know what you need to test and what you don't need to test. All the components that need to be tested can be put under "in scope," and the rest can be defined as "out of scope.".
It helps

- To give precise information on the testing being done
- Also, it helps testers to know exactly what they need to test.

But the main question that arises here is how you would know what needs to be "in scope" and what needs to be "out of scope."
There are a few points that you need to keep in mind while defining the scope of the test

- Look into what exactly customer requirements are
- What is the budget of your project?
- Focus nicely on Product Specification
- You should also take into account your team members' skills and talents.

Step #3. Identifying the Testing Type: which testing should happen

Now that we've established a thorough understanding of what needs to be tested and what doesn't, the next crucial step is defining the types of testing required. Given the diverse array of testing methodologies available for any software product, it's essential to precisely identify the testing types relevant to our software under test.

Prioritization becomes key, allowing us to focus on the most pertinent testing methodologies. This ensures that our testing efforts align with the specific needs and intricacies of the software, optimizing the overall quality assurance process.

You can consider the budget of the project, the time limitations, and your expertise to prioritize the testing type.

Step #4. Interpret test objectives

Defining precise test objectives is paramount for effective test execution, ensuring a systematic approach to identifying and resolving software bugs. The ultimate goal is to ascertain that the software is devoid of defects. The process of interpreting and documenting test objectives involves two critical steps:

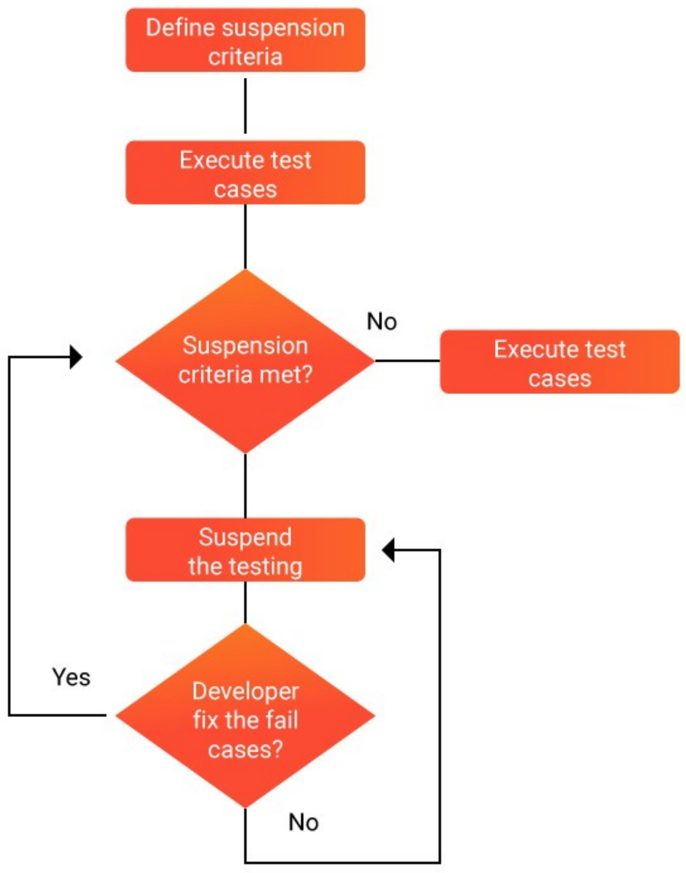
1. **Feature and Functionality Enumeration:**
    - Compile an exhaustive list of all system features, functionalities, performance criteria, and user interface elements. This comprehensive catalog serves as the foundation for targeted test scenarios.
2. **Target Identification:**
    - Based on the listed features, establish the desired end result or target. This involves defining the expected outcomes, performance benchmarks, and user interface standards that signify successful software operation.

Step #5. Outline test criteria

The test criteria are a rule or a standard on which the test procedure is based. 2 types of test criteria need to be resolved:

1. **Suspension Criteria:** Here, you specify the critical suspension criteria for a test. When the suspension criteria are met, the active test cycle is suspended.
2. **Exit Criteria:** Exit criteria specify the successful completion of a test phase.

Exit Criteria



For example, if 95% of all the tests pass, you can consider the test phase to be complete.
The run rate and pass rate are two prominent ways to define exit criteria
Run rate = the number of test cases executed/total test cases of the test specification.
Pass rate = numbers of test cases passed / test cases executed.
These are retrieved from test metrics documents.
The major Run rate has to be 100%. The exception can be considered if a clear and eligible reason is mentioned for a lower run rate.
The pass rate can be variable depending on the project scope. But certainly, a higher pass rate is always a desirable goal.

Step #6. Planning Resources

Resource planning, as the name implies, involves crafting a comprehensive overview of all essential resources essential for project execution. This encompasses a spectrum of elements, including human resources, hardware, software, and other necessary materials.

The significance of resource planning lies in its ability to detail the requirements crucial for the project's success.

By explicitly specifying the required resources, the test manager can formulate precise schedules and accurate estimations, facilitating the seamless and effective execution of the project. This process ensures optimal utilization of resources, contributing to the overall success of the testing project.

No.	**Member**	**Tasks**
1	Test Manager	Manages the entire project. Directs the team. Hires require efficient resources.
2	Tester	Identifies test techniques, tools, and automation architecture. Creates comprehensive test plans. Executes tests, logs results, and reports defects.
3	Developer in Test	Executes test cases, test suites, and related activities.
4	Test Administrator	creates and manages the test environment and its assets. Assists testers in effectively utilizing the test environment.

Some of the system resources you should look for are

1. Server
2. Test tool
3. Network
4. Computer

Step #7. Define test Environment

The test environment is a critical element, encompassing both hardware and software, where the test team executes test cases. It constitutes a real-time instance that mirrors the actual user experience, incorporating the physical environment, including servers and front-end interfaces. To comprehensively define the test environment:

1. **Hardware Configuration:**
    - Specify the hardware components required for testing, detailing server specifications, network configurations, and end-user devices.
2. **Software Configuration:**
    - Outline the software components crucial for testing, including operating systems, databases, browsers, and any specialized testing tools.
3. **User Environment:**
    - Consider the end-user experience by replicating the conditions they will encounter during actual system usage.
4. **Server Setup:**
    - Detail the server architecture, configurations, and any specific settings essential for testing server-side functionalities.
5. **Front-End Interface:**
    - Define the front-end interfaces, detailing the user interfaces, GUI elements, and any specific design considerations.
6. **Data Configuration:**
    - Specify the test data required for executing test cases, ensuring it accurately represents real-world scenarios.
7. **Dependencies:**

    - Identify any external dependencies, such as APIs, third-party integrations, or external services, is crucial for testing.
8. **Test Environment Documentation:**
    - Create comprehensive documentation detailing the entire test environment setup, configurations, and any unique considerations.

Step #8. Create Test Logistics

Creating effective test logistics involves addressing two crucial aspects:

1. Who Will Test?

- **Skill Analysis:** Conduct a thorough analysis of team members' capabilities and skills. Understand their strengths, expertise, and proficiency in specific testing types or tools.
- **Task Assignment:** Based on the skill set, assign appropriate testing tasks to team members. Ensure that each tester is aligned with testing activities that match their expertise.
- **Responsibility Allocation:** Clearly define roles and responsibilities within the testing team. Specify who is responsible for test case creation, execution, result analysis, and defect reporting.
- **Cross-Training:** Consider cross-training team members to enhance flexibility. This ensures that multiple team members can handle critical testing tasks, reducing dependencies.

2. When Will the Test Occur?

- **Timelines:** Establish strict timelines for testing activities to prevent delays. Define specific start and end dates for each testing phase, considering dependencies and overall project timelines.
- **Test Scheduling:** Develop a comprehensive test schedule that outlines when each testing activity will take place. This includes unit testing, integration testing, system testing, and any other relevant testing phases.
- **Parallel Testing:** If applicable, plan for parallel testing to expedite the overall testing process. This involves conducting multiple testing activities simultaneously.
- **Continuous Monitoring:** Implement a continuous monitoring mechanism to track progress against timelines. This helps identify potential delays early on and allows for timely corrective actions.
- **Coordination:** Foster clear communication and coordination among team members to ensure everyone is aware of the testing schedule and any adjustments made.