Scenario-based test case creation

Why?

Improves Test Coverage: Ensures testing covers realistic user journeys and critical paths.

Identifies Edge Cases: Helps discover scenarios that might be missed in regular test case writing.

Enhances Usability: Tests the application as users would, ensuring a better user experience.

What?

Scenario-Based Test Case: A test case derived from real-life scenarios and end-user actions.

Example: "User logs into the system, navigates to the profile page, and updates their contact information."

Where?

E-commerce Websites:

Testing the complete checkout process.

Example: Adding items to the cart, applying coupons, and making payments.

Banking Applications:

Testing fund transfers between accounts.

Example: Logging in, entering recipient details, and confirming the transaction.

Healthcare Systems:

Testing patient data entry workflows.

Example: Adding patient details, scheduling appointments, and generating bills.

Introduction

Scenario-based test case creation involves designing test cases based on realistic user journeys or workflows within a system. Unlike traditional test cases that focus on individual functionalities, scenario-based test cases simulate actual use cases, ensuring the software meets the needs of its users in real-world situations.

This approach is essential in web development and software testing because it provides comprehensive coverage, helps uncover hidden defects, and improves the user experience.

Key Concepts and Definitions

1. Scenario

A scenario represents a sequence of actions or events that a user might perform when interacting with a system.

Example: A user logs in, adds items to a shopping cart, and proceeds to checkout.

2. Test Case

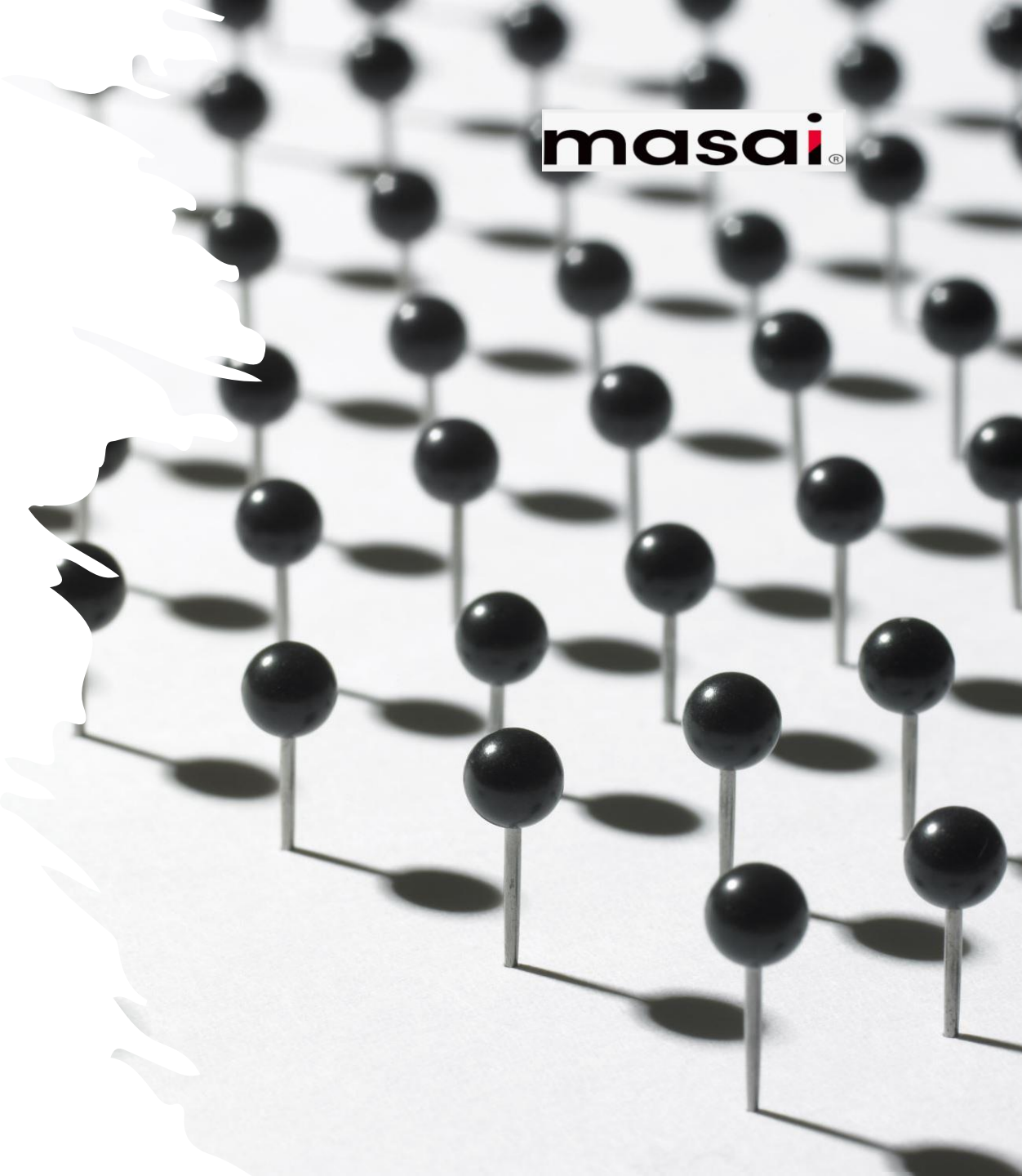
A test case is a set of conditions, inputs, and expected results used to verify the functionality of a specific feature.

Example: Verifying that the login page accepts valid credentials.

3. Scenario-Based Testing

This testing method focuses on workflows or sequences of actions instead of isolated functions.

Example: Testing a complete purchase process on an e-commerce site.



What is a Test Case?

A test case is a detailed document designed for testing teams focusing on a very specific feature of the software.

The test case contains several aspects covering the whole testing process of a particular feature. For the most part, this document will include sections like test steps, expected results, real results, status, and so on.

Since it is a comprehensive document, test cases concentrate both on what to test and mainly on how to test, giving the testers a step-by-step guide to the testing process.

Advantages of Test Case

Detailed documentation contains all the important data about a specific feature

Focuses on how to test and clarifies the steps included in the testing efforts

Helps to understand if the software is functioning as it should

Reduces the possibility of missing essential testing steps

Allows testing team managers to effectively assign the workforce to the testing assignments

Example

Consider an e-commerce website where a user needs to purchase an item:

Scenario: "User buys a product on an e-commerce platform."

Precondition: The user has an active account and is logged in.

Steps:

Navigate to the homepage.

Search for the desired product.

Add the product to the cart.

Go to the checkout page.

Enter shipping information and payment details.

Confirm the order.

Expected Outcome: The order is placed successfully, and the user receives a confirmation message.

By using scenario-based test cases, you ensure your testing aligns with actual user behavior, improving the software's reliability and user experience.

Objective or Why Do We Write the Test Cases?

Test cases play a crucial role in the software testing lifecycle, as they discuss how testing must be executed. The main objective of writing test cases is to check whether the software works. Let’s take a look at some of the benefits ,

It helps check whether a particular module or software meets the specified requirement.

It provide a clear and structured description of the test scenarios, inputs, and expected results. It is valuable for future reference, maintenance, and regression testing, ensuring the software functions correctly even after updates.

It helps determine whether the software works under a given set of conditions.

When you create test cases, you will think about all the aspects of your software application, which will help you identify any software gaps easily.

Common Features of Test Cases

The following are some of the common features ,

Revised and updated regularly

Software requirements keep varying depending on customer preferences or business priorities. So, whenever there is a change in the requirements, testers will have to modify the test cases accordingly.

Involves clustering
In a single test scenario, test cases often need to be executed in a certain order or a group. In such cases, certain prerequisites of one test case will apply to other test cases within the same sequence.

They are interdependent
It often depend on each other, particularly in layered applications that have multi-tier business logic.

Used by both testers and developers
It's helpful for both developers and testers. When developers fix a bug, they can easily replicate it using the test cases.

Types of Test Cases
Several types can help test different aspects of the software. As a software tester, understanding the difference between them will help you focus on the efforts and pick the right test format. Here are some common types of test cases,

Functionality – These test cases focus on verifying the functional requirements of the software by testing individual features or functionality to ensure they work as intended.
Performance -Performance test cases evaluate the performance of the software. It helps ensure that the software works as intended in terms of speed, stability, and scalability.

Unit – In general, software developers perform unity testing to verify their code or individual units. Unit test cases help check if each unit is working as intended and identify bugs in the early stages of development.

User Interface – UI test cases validate the software's graphical user interface (GUI), ensuring that elements such as buttons, menus, forms, and layouts are working correctly and are easy to use.

Security – These test cases focus on identifying vulnerabilities and weaknesses in the software's security measures. They aim to protect your application data against unauthorized access, malicious attacks and resist potential security threats.

Integration – Integration test cases verify that different software components are working as intended after integration. It helps check whether software components or modules integrate seamlessly to work as a complete product.

Database – The database is where all the application information will be stored. Database testing helps check what is happening in the background of an application. It is also called backed testing.

Database test cases make sure that the database is functioning correctly or not. Testers typically use SQL queries to create database test cases.

Usability – These test cases help check if users are able to use the application without any difficulty or confusion. It also checks factors such as ease of navigation, clarity of instructions, and overall user experience.
User Acceptance – User acceptance test cases help check the application from an end-user perspective. The test cases are broad and cover the entire application. UAT testing is typically the final step before the application goes to production.
Regression – Regression test cases help ensure that recent code changes do not affect the existing software functionality. They help maintain the stability and integrity of the software over time.

Typical Test Case Parameters

Let's talk about the parameters in detail,

Test Case ID – It is a unique identifier for the test case. It is usually represented using alphanumeric or numeric characters.

Test Scenario – It is a brief description of what needs to be performed by the testers.

Test Case Description – It is a detailed explanation of what function has to be tested.

Prerequisite – Prerequisites are conditions that are required to perform the test. Typically, testers must check or fulfill the prerequisites before starting the test process.

Test Steps -Test steps are a sequence of steps or actions to be executed to check a particular functionality or condition.

Test Data – Test data refers to the input data or values required to execute the test case. For example, username and password are the test data to test the email login.

Expected Result – It refers to the expected behavior or output that should be observed when executing the test case.
Actual Result – It is the actual output or behavior observed during the test case execution.
Test Environment – Test environment refers to the environment in which the test has to be performed, like operating system, version, browser, etc. It includes all hardware, software, and network configurations.
Test Priority -Test priority is all about what test cases should be prioritized and what can be performed later.
Status – It shows the status of tests, like pass, fail, or NA.
Comments – It includes remarks on how developers can improve the software quality.

Example of Test Case Format

The following is an example of a test case to check the login functionality,

https://docs.google.com/spreadsheets/d/1Wmf6mIWFbGQ9oUQb0K9UACjyBCLgzMuR1dy1qh0U5C4/edit?gid=0#gid=0

Popular test case management tools

- Test case management tools help you manage software and hardware development. These tools track your test cases, bugs, and other important information related to testing.

  Zephyr Scale-**Description**: A scalable test management solution designed for teams using JIRA.
- **Key Features**:
- Supports test planning, execution, and reporting.
- Advanced traceability between tests and requirements.
- Customizable test organization for large-scale projects.
- **Best For**: Teams needing a robust test management tool integrated into JIRA

- **Zephyr Squad-Description**: A simpler version of Zephyr designed for Agile teams.
- **Key Features**:
- Lightweight and user-friendly.
- Seamless JIRA integration for managing test cases within sprint workflows.
- Real-time reporting and dashboards.
- **Best For**: Small to medium-sized Agile teams.

Juno One

Description: A modern test management tool for tracking test cases, projects, and team performance.

Key Features:

Intuitive UI with test case tracking.

Integration with JIRA and CI/CD pipelines.

Advanced reporting and metrics.

Best For: Teams wanting simplicity and integration with development tools.

Klaros-Test management -**Description**: A comprehensive test management tool with a focus on automation and integration.

Key Features:

Supports manual and automated test cases.

Integration with popular CI/CD tools and test automation frameworks.

Customizable dashboards and reports.

Best For: Enterprises requiring both manual and automated test management.

QA Coverage -**Description**: A test management and quality assurance platform.

Key Features:

Comprehensive test case design and execution.

Customizable workflows for QA processes.

Integration with other tools like JIRA and Jenkins.

Best For: Teams needing end-to-end test management with flexibility.

Activity

Doubt clarification


THANK
YOU!