

Functional performance testing.

Key Aspects of Functional Performance Testing:

1. Functional Testing:

- **Objective:** Verifies that the software or system performs its intended functions correctly. It focuses on checking whether the system operates as per the functional specifications.
- **Test Scope:** Includes features, workflows, and business logic of the application.
- **Examples:**
 - Verifying if a user can log in with valid credentials.
 - Checking if a system calculates discounts correctly.
 - Ensuring that an e-commerce site adds items to the shopping cart.

2. Performance Testing:

- **Objective:** Assesses how well the system performs under different loads, focusing on its responsiveness, stability, and scalability.
- **Key Performance Metrics:**
 - **Response Time:** How fast the application responds to user actions.
 - **Throughput:** The number of transactions processed within a specific time period.
 - **Scalability:** How the application handles increasing loads, whether horizontal or vertical scaling works effectively.
 - **Resource Utilization:** CPU, memory, and network bandwidth usage during load.

Approach to Functional Performance Testing:

1. Requirement Gathering:

- Understand the business functions the system is expected to support.
- Identify the system's key workflows and their expected outcomes.

2. Test Case Creation:

Team Member: Sayali Sutar

- Based on functional requirements, design test cases to verify individual functionalities (e.g., form submission, login/logout behavior, search queries).
- Define performance criteria for each function, like acceptable response time for a login action, or acceptable transaction rate for order processing.

3. Load and Stress Testing:

- **Load Testing:** Test the system's ability to handle expected loads (e.g., number of simultaneous users).
- **Stress Testing:** Push the system beyond normal operational limits to evaluate how it behaves under extreme conditions (e.g., how the system reacts when it exceeds its peak load).

4. End-to-End Workflow Validation:

- **Test End-to-End User Scenarios:** Verify the full workflow of user actions (e.g., a user adding items to the cart, checking out, and receiving an order confirmation) under typical and peak load conditions.
- **Check Integration Points:** Validate performance when the system interacts with third-party services or databases.

5. Performance Metrics Collection:

- Measure the performance of the system while functional tests are being executed under different conditions (e.g., normal load, peak load, etc.).
- Collect and analyze metrics such as response time, system throughput, CPU/memory usage, database queries per second, etc.

6. Analysis and Reporting:

- **Identify Bottlenecks:** Detect areas where the system may degrade under load (e.g., slow database queries, excessive memory consumption).
- **Performance Tuning:** Based on the results, recommend improvements, such as code optimizations, resource scaling, or hardware upgrades.
- **Compliance Check:** Ensure that the system meets the required performance benchmarks and functional requirements.

Functional Performance Testing Scenarios Examples:

1. Login Page Performance:

- **Functional Test:** Verify that a user can log in with valid credentials.

Team Member: Sayali Sutar

- **Performance Test:** Measure the response time for logging in with increasing numbers of concurrent users (e.g., 1000, 5000 users).

2. Search Functionality:

- **Functional Test:** Verify that a search query returns accurate results.
- **Performance Test:** Measure how long it takes to return results under different load conditions (e.g., when there are thousands or millions of records in the database).

3. Order Placement (E-commerce):

- **Functional Test:** Verify that a user can successfully place an order, including adding items to the cart, selecting shipping, and completing the checkout process.
- **Performance Test:** Measure the system's performance when multiple users are placing orders simultaneously, focusing on transaction throughput and response times.

4. File Upload Feature:

- **Functional Test:** Verify that a user can upload a file successfully and that the file is processed correctly.
- **Performance Test:** Test how the system handles uploading large files, multiple files simultaneously, and a large number of users uploading files concurrently.

Tools for Functional Performance Testing:

1. Load Testing Tools:

- **Apache JMeter:** A popular tool for performance and load testing, capable of simulating multiple users interacting with the system.
- **LoadRunner:** A comprehensive performance testing tool used to evaluate the behavior of applications under various load conditions.
- **Gatling:** An open-source tool for load testing that focuses on high performance and scalability.

2. Monitoring Tools:

- **New Relic:** Provides real-time performance monitoring and analysis, helping to identify bottlenecks and resource usage.
- **Dynatrace:** Offers automatic performance monitoring, including user behavior, application performance, and infrastructure metrics.

Team Member: Sayali Sutar

3. Automated Testing Frameworks:

- **Selenium:** A popular tool for automating functional UI tests, which can be integrated into performance testing pipelines.
- **Cypress:** A modern testing framework for functional testing with real-time reloading and fast execution, suitable for integrating with performance tests.

4. Cloud-Based Testing Services:

- **BlazeMeter:** A cloud-based performance testing service that supports JMeter scripts and load testing for web applications.
- **Flood.io:** A load testing service that allows you to simulate traffic from different locations and monitor performance.