**1. What are the needed test scenarios?**

**Functional Test Scenarios**

1. **Basic Search Functionality**
   * Search for an organization by name.
   * Verify search results match the input keyword.
2. **Search with Partial Name**
   * Enter a partial name of an organization.
   * Verify search results include all possible matches.
3. **Search with Special Characters**
   * Enter search terms with special characters (e.g., &, @).
   * Verify search results handle special characters correctly.
4. **Empty Search Query**
   * Perform a search with an empty input.
   * Verify system handles it gracefully (e.g., displays a message).
5. **Case Insensitivity**
   * Perform searches with varying cases (e.g., ALFRED, alfred).
   * Verify results are case-insensitive.
6. **Invalid Search Query**
   * Search for non-existent organizations.
   * Verify appropriate message or no results are displayed.
7. **Search Result Details**
   * Verify details of search results (name, location, description).
8. **Pagination of Search Results**
   * Verify that results are paginated if too many to display on one page.

**Performance Test Scenarios**

1. **Load Testing**
   * Simulate concurrent users performing searches.
   * Measure response time and system performance.
2. **Stress Testing**
   * Gradually increase load beyond normal operational capacity.
   * Observe system behavior and identify breaking points.
3. **Scalability Testing**
   * Test system's ability to scale up with increasing load.

**API Test Scenarios**

1. **Valid Search Request**
   * Validate API response for a valid search query.
2. **Invalid Search Request**
   * Validate API response for invalid or malformed queries.
3. **Boundary Testing**
   * Test API with minimum and maximum input values.
4. **Error Handling**
   * Test API response for scenarios such as missing parameters or incorrect API keys.
5. **Performance of API**
   * Measure response time under different loads.

**2. What is the documentation process for these test scenarios?**

1. **Define Test Scenarios**
   * List all functional, performance, and API test scenarios.
2. **Create Test Cases**
   * Document detailed test cases for each scenario, including steps, expected results, and test data.
3. **Test Case Review**
   * Peer review test cases for completeness and accuracy.
4. **Test Plan Document**
   * Include test scenarios and cases in the test plan.
5. **Test Strategy Document**
   * Outline the approach, tools, and methods for executing the tests.
6. **Test Execution**
   * Execute test cases and document results.
7. **Test Report**
   * Prepare a test report summarizing test execution, results, and any defects found.
8. **Defect Management**
   * Log defects, track their resolution, and update documentation accordingly.

**3. Which tests should be automated?**

1. **UI Tests (Cypress)**
   * Basic search functionality.
   * Search with partial names and special characters.
   * Case insensitivity and empty search queries.
   * Validating search result details.
2. **API Tests (Postman/Cypress)**
   * Valid and invalid search requests.
   * Boundary testing and error handling.
   * Performance and response time measurements.
3. **Performance Tests (JMeter)**
   * Load and stress testing.
   * Scalability testing.

**4. How do we ensure that the search is performing well? Can it handle load? Are the search results being returned in a timely manner?**

**Approach to Ensure Search Performance**

1. **Performance Testing with JMeter**
   * Create a JMeter test plan to simulate concurrent users performing search operations.
   * Configure test to simulate up to 10,000 virtual users.
   * Measure response times, throughput, and error rates.
2. **Automated Performance Monitoring**
   * Set up monitoring tools to continuously track performance metrics.
   * Monitor server resources (CPU, memory, disk I/O) during peak loads.
   * Track API response times and page load times.
3. **Scalability Testing**
   * Gradually increase load and observe system behavior.
   * Ensure system can scale by adding more resources or instances.
4. **Load Testing Execution Plan**
   * Execute load tests during non-peak hours or in a staging environment.
   * Analyze results to identify performance bottlenecks.
5. **Optimizing System Performance**
   * Optimize database queries and indexing.
   * Implement caching mechanisms.
   * Optimize front-end code and assets (e.g., JavaScript, CSS).
6. **Regression Testing**
   * Ensure no performance degradation with new changes.
   * Regularly execute performance tests as part of the CI/CD pipeline.

**Example Test Case for Performance Testing with JMeter:**

jmeter

Copy code

Test Plan

Thread Group

- Number of Threads (Users): 10000

- Ramp-Up Period: 100 seconds

- Loop Count: 1

HTTP Request

- Server Name or IP: www.givelify.com

- Path: /search

- Parameters: {query=Alfred Street Baptist Church}

View Results in Table

Summary Report

Response Time Graph