# **Bug Reporting Task**

**Bug ID: BUG-001**

**Severity:** Critical  
**Priority:** High

**Title:** Login Page refreshes without logging-in the user after entering valid credentials

**Description:** After entering valid credentials and clicking the Login button, the page refreshes, but the user is not logged in.

**Steps to Reproduce:**

1. Open the website
2. Navigate to the Login Page
3. Enter a ‘valid username’ in the username field
4. Enter a ‘valid password’ in the password field
5. Click the ‘Login’ button

**Expected Result:**

The user should be redirected to the Home page after successfully being logged in.

**Actual Result:**

Clicking the login button refreshes the page, but the user remains on the Login Page without being logged in.

**Environment:**

**Device & OS:** MacBook Pro 2019’ 15 / macOS 13.7

**Browser:** Safar 18.0.1

**Account & Password Used: testuser / password123**

**Reproducibility:** 3/3

# **API Testing Task**

***Test Case ID: TC\_API\_001***

**API Endpoint**: GET /api/users

**Description**: Verify that the API returns a list of users.

**Preconditions**: The user must be authenticated.

**Test Steps**: Send a GET request to /api/users.

**Expected Result**:

• Status code: 200

• Response body contains an array of user objects.

• Each user object includes the following fields: id, name, email.

**Test Case ID: TC\_API\_002**

**API Endpoint**: POST /api/login

**Description**: Verify that the login API successfully authenticates a user with valid credentials.

**Preconditions**: None.

**Test Steps**: Send a POST request to /api/login with valid credentials (username and password).

**Expected Result**:

• Status code: 200

• Response body includes a token.

**Test Case ID: TC\_API\_003**

**API Endpoint**: GET /api/photos

**Description**: Verify that the API returns a list of photos.

**Preconditions**: User must be authenticated.

**Test Steps**: Send a GET request to /api/photos.

**Expected Result**:

• Status code: 200

• Response body contains an array of photo objects.

**Test Case ID: TC\_API\_004**

**API Endpoint**: DELETE /api/photos/{id}

**Description**: Verify that the API allows deletion of a photo by ID.

**Preconditions**: The photo must exist, and the user must be authenticated.

**Test Steps**: Send a DELETE request to /api/photos/{photoId}.

**Expected Result**: Status code: 204 (No Content).

# **Performance Testing Task**

**Objective**: To assess the performance, scalability, and stability of the web application under various load conditions.

**Scope**:

Test the following aspects of the web application:

**Load Testing**: Assess how the application behaves under expected load conditions.

**Stress Testing**: Determine the application’s breaking point by pushing it beyond normal load.

**Endurance Testing**: Verify the application's behavior under sustained load over an extended period.

**Spike Testing**: Evaluate how the application handles sudden, large spikes in traffic.

**Test Environment**:

Identify the hardware, software, and network configuration used during testing.

Ensure the environment mirrors the production environment as closely as possible.

**Test Scenarios**:

1. **Load Testing**:

• Simulate X users accessing the application concurrently.

• Monitor response times for key operations (e.g., login, API calls).

1. **Stress Testing** :

• Gradually increase the number of users until the application fails.

• Observe how the system responds to resource exhaustion.

1. **Endurance Testing**:

• Maintain a constant load of Y users for a period of 24 hours.

• Monitor memory and resource usage over time.

1. **Spike Testing**: Introduce sudden traffic increases (e.g., 200% of normal traffic) and observe system behavior.
2. **Performance Metrics**:

• Response Time: Time taken for the application to respond to requests.

• Throughput: Number of requests processed per unit time.

• Error Rate: Percentage of failed requests.

• Resource Utilization: CPU, memory, and bandwidth usage.

**Tools**:

List the performance testing tools you will use (e.g., JMeter, LoadRunner, or any other relevant tool).

**Reporting**:

Define how results will be reported, including key metrics, graphs, and recommendations for improvements.

**Schedule**:

Provide a timeline for the performance testing activities, including preparation, execution, and reporting phases.