**1. 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

### **2. API Testing Task**

For each API endpoint, consider the following structure for your test cases:

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

1. **API Endpoint**: GET /api/users
2. **Description**: Verify that the API returns a list of users.
3. **Preconditions**: The user must be authenticated.
4. **Test Steps**:

* Send a GET request to /api/users.

1. **Expected Result**:
   1. Status code: 200
   2. Response body contains an array of user objects.
   3. Each user object includes the following fields: id, name, email.

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

1. **API Endpoint**: POST /api/login
2. **Description**: Verify that the login API successfully authenticates a user with valid credentials.
3. **Preconditions**: None.
4. **Test Steps**:

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

1. **Expected Result**:
   * 1. Status code: 200
     2. Response body includes a token.

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

1. **API Endpoint**: GET /api/photos
   1. **Description**: Verify that the API returns a list of photos.
   2. **Preconditions**: User must be authenticated.
   3. **Test Steps**:

* Send a GET request to /api/photos.
  1. **Expected Result**:
     1. Status code: 200
     2. Response body contains an array of photo objects.

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

1. **API Endpoint**: DELETE /api/photos/{id}
2. **Description**: Verify that the API allows deletion of a photo by ID.
3. **Preconditions**: The photo must exist, and the user must be authenticated.
4. **Test Steps**:

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

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

### **3. Performance Testing Task**

1. **Objective**:
   1. To assess the performance, scalability, and stability of the web application under various load conditions.
2. **Scope**:
   1. Test the following aspects of the web application:
      1. Load Testing: Assess how the application behaves under expected load conditions.
      2. Stress Testing: Determine the application’s breaking point by pushing it beyond normal load.
      3. Endurance Testing: Verify the application's behavior under sustained load over an extended period.
      4. Spike Testing: Evaluate how the application handles sudden, large spikes in traffic.
3. **Test Environment**:
   1. Identify the hardware, software, and network configuration used during testing.
   2. Ensure the environment mirrors the production environment as closely as possible.
4. **Test Scenarios**:
   1. **Load Testing**:
      1. Simulate X users accessing the application concurrently.
      2. Monitor response times for key operations (e.g., login, API calls).
   2. **Stress Testing**:
      1. Gradually increase the number of users until the application fails.
      2. Observe how the system responds to resource exhaustion.
   3. **Endurance Testing**:
      1. Maintain a constant load of Y users for a period of 24 hours.
      2. Monitor memory and resource usage over time.
   4. **Spike Testing**:
      1. Introduce sudden traffic increases (e.g., 200% of normal traffic) and observe system behavior.
5. **Performance Metrics**:
   1. Response Time: Time taken for the application to respond to requests.
   2. Throughput: Number of requests processed per unit time.
   3. Error Rate: Percentage of failed requests.
   4. Resource Utilization: CPU, memory, and bandwidth usage.
6. **Tools**:
   1. List the performance testing tools you will use (e.g., JMeter, LoadRunner, or any other relevant tool).
7. **Reporting**:
   1. Define how results will be reported, including key metrics, graphs, and recommendations for improvements.
8. **Schedule**:
   1. Provide a timeline for the performance testing activities, including preparation, execution, and reporting phases.