**SAVANNAH INFORMATICS TEST**

**Manual Testing**

**Test Case Design:**

Write test cases covering positive and negative functionality scenarios while covering

the critical path of the web application.

**1. Landing Page**

**Positive Test Cases:**

Verify that the landing page loads successfully with all elements visible.

Check that the page has a 'Login' link and is clickable.

**Negative Test Cases:**

Remove CSS/JS files and confirm the page gracefully handles the missing assets.

**2. Login Page**

**Login Page (Google Authentication)**

**Positive Scenarios:**

Verify the Google login button is visible and functional.

Verify a user is redirected to the Google authentication page upon clicking the login button.

Verify successful login redirects the user to the Home page.

**Negative Scenarios:**

Verify the app handles Google login errors (e.g., incorrect credentials, denied permissions).

Verify the app displays an appropriate message if the Google authentication process fails.

Verify users cannot bypass the Google login page by manipulating the URL.

**3. Home Page (Authenticated)**

**Positive Test Cases:**

Verify access to the home page after successful login.

Ensure the home page displays previously restricted content like users, albums and photos.

Check that the logout button is present and working.

**Negative Test Cases:**

Attempt to access the home page without logging in and verify it redirects to the login page.

Modify cookies/session tokens manually and confirm the page denies access.

**4. User Page (Authenticated)**

**Positive Test Cases:**

Verify user information is displayed correctly (name, email, address etc).

**Negative Test Cases:**

Access the user page without logging in — should redirect to login.

**5. Album Page (Authenticated)**

**Positive Test Cases:**

Verify albums are listed correctly with appropriate titles alongside their owners.

**Negative Test Cases:**

Attempt to access albums without authentication — should redirect to login.

**6. Photo Page (Authenticated)**

**Positive Test Cases:**

Ensure photos load properly with titles and descriptions and the link to where it is stored.

**Negative Test Cases:**

Access the photo page without logging in — should redirect to login.

**7. Logout**

**Positive Test Cases:**

Verify clicking the 'Logout' button logs the user out and redirects to the landing page.

Ensure session cookies are cleared after logging out.

**Negative Test Cases:**

Try accessing any authenticated page after logout — should redirect to login.

**Bug Report**

**Title: Login with Google refreshes the page without authenticating the user**

**Severity:** Critical

**Priority:** High

**Environment:**

**Browser:** Chrome 120.0

**OS:** Windows 11(2021)

**Django Version:** 4.0

**allauth Version:** 0.54.0

**Preconditions:**

The application is running locally at http://127.0.0.1:8000

Google authentication has been integrated using Django allauth.

**Steps to Reproduce:**

Navigate to the landing page.

Click the Log in link

Click the "Sign In with Google" button.

Select a valid Google account and allow permissions.

Observe the page behavior after redirection.

**Expected Result:**

The user should be redirected to the Home page after a successful Google login.

**Actual Result:**

After selecting the Google account, the page refreshes, and the user remains on the landing page without being authenticated.

**Possible Cause:**

Incorrect redirect\_uri configured in Google OAuth settings.

Missing or misconfigured LOGIN\_REDIRECT\_URL in Django settings.

Incomplete Google OAuth setup or incorrect client ID/secret.

**Attachments:**

Screenshot of the network tab showing the failed redirect.

Log file capturing the response from Google.

**Additional Notes:**

Double-check the redirect\_uri in Google Developer Console.

Ensure LOGIN\_REDIRECT\_URL points to a valid route (e.g., /home/).

Test with a new Google OAuth client ID/secret to rule out misconfigurations.

**Endpoints Testcases**

**/users**  
Verify status code 200 - Ensure the API returns a 200 status code when the endpoint is accessed successfully.

Verify response structure - Validate that the response contains all expected fields. The response should have fields: id, name, username, email, address, phone, website, company.

Verify nested address fields - Ensure the address field contains all the expected nested fields. Address should have street, suite, city, zipcode, and geo (with lat and lng).

Verify email format - Check that the email field follows a valid email format.

Verify user ID is an integer - Ensure the id field is an integer. id should be an integer.

**/albums**

Verify successful retrieval of albums - Status code 200 OK. Response body contains album objects with fields: userId, id, and title

Verify response structure - Each album object in the response contains: userId (integer), id (integer), title (string)

**/photos**

Verify that the API returns a list of photos. The HTTP response should be 200 OK, and response contains a list of photos with correct fields: albumId, id, title, url, and thumbnailUrl.

Validate Photo Schema - Ensure that the photo response matches the expected schema. Ensure each photo object contains albumId, id, title, url, and thumbnailUrl.

**Performance Test Plan**

**1. Introduction**

This Performance Test Plan outlines the strategy and approach for testing the web app’s performance, ensuring it can handle user load, stress conditions, and provide quick responses. The plan focuses on Load Testing, Stress Testing, and Response Time Testing, measuring critical performance metrics.

**2. Objectives**

**Load Testing:** Identify the maximum number of users the website can handle under normal conditions.

**Stress Testing:** Evaluate how the website behaves when pushed beyond its limits.

**Response Time Testing:** Measure how quickly the website responds to user requests.

Ensure performance metrics — Response Time, Error Rate, and Latency — remain within acceptable thresholds.

**3. Test Environment**

**Hardware:** Specify server configuration (CPU, RAM, disk space)

**Software:** OS, database, web server versions

**Network:** Bandwidth, latency, and packet loss considerations

**Test Tools:** Use k6 for load, stress, and response time testing

**4. Test Scenarios**

**4.1 Load Testing Scenarios**

**Normal load:** Simulate typical daily traffic (e.g., 100 concurrent users)

**Peak load:** Test with maximum expected load (e.g., 500 concurrent users)

**Gradual ramp-up:** Increase users incrementally to monitor system performance trends

**4.2 Stress Testing Scenarios**

**Spike test:** Sudden increase in users (e.g., from 100 to 1000 users instantly)

**Sustained overload:** Continuous high load to check for memory leaks or crashes

**Failover testing:** Simulate server failure to check system recovery time

**4.3 Response Time Testing Scenarios**

Measure time taken to load key pages: Home, Users, Albums, and Photos

Test API response times for the following endpoints:

GET /users

GET /albums

GET /photos

Validate search functionality response time on the Photos page

**5. Performance Metrics**

**Response Time:** Time taken for the website to respond to requests (target: under 500ms)

**Error Rate:** Percentage of failed requests during testing (acceptable threshold: < 1%)

**Latency:** Delay before data transfer begins (target: under 200ms)

**6. Test Execution Plan**

**Load Testing:**

Ramp-up from 10 to 500 users over 10 minutes

Steady state for 15 minutes

**Stress Testing:**

Sudden spike from 100 to 1000 users

Sustained load at 1200 users for 20 minutes

**Response Time Testing:**

Measure response times for 100, 300, and 500 concurrent users

**7. Reporting and Analysis**

Metrics to capture: Response times, error rates, latency, throughput, and CPU/memory usage

Reporting tools: k6 output in JSON format, visualized with Grafana

Analysis: Identify bottlenecks, slow endpoints, and resource constraints

**8. Risk and Mitigation**

Network instability: Ensure backup network resources

Test environment limitations: Use cloud-based load generators if needed

Unexpected server crashes: Pre-define rollback strategies

**9. Sign-off**

All stakeholders (QA, DevOps, Project Managers) will review and approve the test results.