# Test Plan

## Test Plan for User Authentication

### 1. Introduction

This test plan outlines the testing strategy for the user authentication system of the EcoGo web application. The goal is to verify that users can successfully log in, log out, and that authentication-based access control functions correctly.

### 2. Scope

<u>In-Scope Features:</u>

- User Login with Valid Credentials
- User Login with Invalid Credentials
- User Logout
- Homepage Unauthorised Access Control

<u>Out-of-Scope Features:</u>

- Password reset and recovery mechanisms.
- Multi-factor authentication (MFA).

### 3. Test Objectives

- Verify that a registered user can log in successfully.
- Ensure that login fails when incorrect credentials are used.
- Confirm that an authenticated user can log out properly.
- Ensure that unauthenticated users are restricted from accessing the homepage.

### 4. Test Approach

The authentication system will be tested using automated unit tests with Django's built-in TestCase framework.

- Testing Methodology: Automated backend testing using Django's Test Client.
- Test Execution: The test cases will be run using Django's unittest framework.

### 5. Test Environment

Framework: Django

Testing Tool: Django TestCase

Database: Django default SQLite (test database)

Test Data: A predefined test user created in setUp().

## 6. Test Cases

**Test Case 1:** User Login with Valid Credentials

**Objective:** Ensure that a registered user can log in successfully.

**Test Steps:**

1. Send a POST request to self.login_url with valid credentials (testuser, testpass123).
2. Verify that the response redirects to the homepage (self.homepage_url).
3. Confirm that the session key is set, indicating successful authentication.

**Expected Outcome:**

- The response should redirect to the homepage after login.
- The session key should exist, confirming the user is authenticated.

**Test Case 2:** User Login with Invalid Credentials

**Objective:** Ensure that login fails when incorrect credentials are used.

**Test Steps:**

1. Send a POST request to self.login_url with an invalid password.
2. Verify that the response re-renders the login page (user/login.html) with errors.
3. Check that the response status code is 200, indicating login failure.
4. Confirm that the user is not authenticated (session key should not exist).

**Expected Outcome:**

- The login page is re-rendered with an error message.
- The response returns a 200 status code.
- The session key should not exist, confirming authentication failure.

**Test Case 3:** User Logout

**Objective**: Verify that an authenticated user can log out successfully.

**Test Steps:**

1. Log in the test user using a POST request with valid credentials.
2. Verify that the session contains the authenticated user.
3. Send a GET request to self.logout_url to log out.
4. Verify that the response redirects to the landing page (self.landing_url).

5. Confirm that the session is cleared, ensuring the user is logged out.

**Expected Outcome:**

- The user is logged out successfully.
- The response redirects to the landing page.
- The session key should be removed, confirming that the user is no longer authenticated.

**Test Case 4:** Homepage Access Requires Authentication (Login)

**Objective:** Verify that an unauthenticated user trying to access the homepage is redirected to the login page.

**Test Steps:**

1. Send a GET request to self.homepage_url without logging in.

2. Verify that the response returns a 302 status code (redirect).

3. Ensure that the redirect URL points to the login page (self.login_url).

**Expected Outcome:**

- The response should return status code 302 (indicating a redirect).
- The user is redirected to the login page, confirming that authentication is required.

## 7. Test Execution

- The tests will be executed using Django's TestCase framework.
- Results will be recorded, indicating pass/fail status.
- Any failed tests will be analyzed for errors, and necessary fixes will be implemented.

**<u>CardsTest</u>**

<u>Test Case: test_create_card</u>

- Ensures that cards can be created with and without an image.
    - Create a card named "uno" without an image.
    - Create a card named "Grunty" with an image "Test.jpg".
    - Attempt to create a card named "dos" with a non-existing image "Missing".

- Expected Results:
    - The card "uno" should be created with a default image path.
    - The card "Grunty" should be created with the specified image path
    - A *FileNotFoundError* should be raised for the card "dos"

Test Case: test_change_image

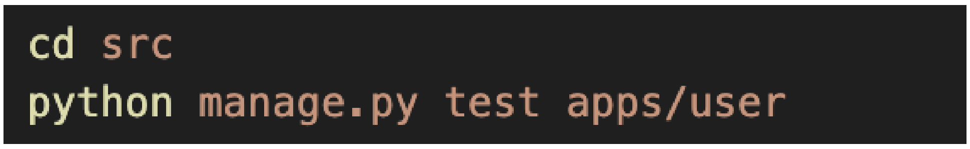
- Ensure that the card's image can be changed to an existing file.

- Expected results:
  - The image of "card0" should be updated to the new image path.
  - A *FileNotFoundError* should be raised for a non-existent file.

**To Run The Tests:**

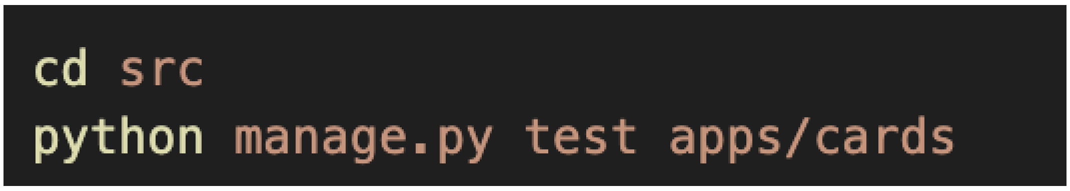
Before running the tests, ensure you take a look at the README file.

To run the tests, use the following command:
To conduct the user tests:



```
cd src
python manage.py test apps/user
```

To conduct the cards tests:



```
cd src
python manage.py test apps/cards
```