# Project: Automated Testing Documentation

## 1. Acceptance Criteria

The acceptance criteria for this project define the expected functionality and requirements that must be fulfilled for the application to be considered complete. These include:

- The user should be able to sign up with a username, email, and password, and the system should validate all inputs.

- Users should be able to log in and log out of their accounts.

- Users should be redirected to their profile page after logging in, where their BMI is displayed.

- Users should be able to download workout plans based on their BMI.

- Users should be able to update their profile information such as age, weight, and height.

- The system should render static pages like Home, About, and Sign-Up without errors.

## 2. Automated Tests

The following automated tests were implemented to verify the functionality of the application:

1. `test\_render\_signup\_view\_get`: Tests that the Sign-Up page renders correctly.

2. `test\_render\_signup\_view\_post`: Tests that users can successfully sign up with valid data.

3. `test\_render\_login\_view\_get`: Tests that the Log-In page renders correctly.

4. `test\_render\_login\_view\_post`: Tests that users can log in with correct credentials.

5. `test\_render\_home\_page`: Tests that the Home page renders correctly.

6. `test\_render\_about\_page`: Tests that the About page renders correctly.

7. `test\_load\_profile\_user\_data`: Tests that user profile data is loaded and BMI is calculated correctly.

8. `test\_download\_workout\_plan`: Tests that the workout plan download feature works as expected.

9. `test\_update\_profile`: Tests that users can update their profile data.

## 3. Test Code

Below is the automated test code implemented for the project:

import pytest  
from django.urls import reverse  
from django.contrib.auth import get\_user\_model  
import json  
  
User = get\_user\_model()  
  
@pytest.mark.django\_db  
def test\_render\_signup\_view\_get(client):  
 response = client.get(reverse('SignUp'))  
 assert response.status\_code == 200  
 assert 'SignUp.html' in [t.name for t in response.templates]  
  
@pytest.mark.django\_db  
def test\_render\_signup\_view\_post(client):  
 response = client.post(reverse('SignUp'), data={  
 'username': 'testuser',  
 'email': 'testuser@example.com',  
 'password1': 'testpassword123',  
 'password2': 'testpassword123',  
 'age': 25,  
 'height': 170,  
 'weight': 70,  
 })  
 assert response.status\_code == 302  
 assert response.url == reverse('Home')  
 user = User.objects.get(username='testuser')  
 assert user.email == 'testuser@example.com'  
  
# Additional tests follow similar structure...