**Test Cases for Django Views**

**User Registration API View**

Purpose:

To provide an endpoint for clients to create a new user.

Test Cases:

Valid User Registration:

Input: Valid user registration data (e.g., email, password).

Expected Output: HTTP 201 Created response with user details and authentication tokens.

Invalid User Registration:

Input: Invalid user registration data (e.g., missing required fields, invalid email format).

Expected Output: HTTP 400 Bad Request response with error details.

User Login API View

Purpose:

To authenticate existing users using their email and password.

Test Cases:

Valid User Login:

Input: Valid user login credentials.

Expected Output: HTTP 200 OK response with user details and authentication tokens.

Invalid User Login:

Input: Invalid user login credentials (e.g., incorrect password, non-existing user).

Expected Output: HTTP 400 Bad Request response with error details.

User Logout API View

Purpose:

To logout users by blacklisting their refresh tokens.

Test Cases:

Valid User Logout:

Input: Valid refresh token.

Expected Output: HTTP 205 Reset Content response indicating successful logout.

Invalid User Logout:

Input: Invalid or missing refresh token.

Expected Output: HTTP 400 Bad Request response with error details.

User Profile API View

Purpose:

To get and update user information.

Test Cases:

Get User Information:

Input: None.

Expected Output: HTTP 200 OK response with the user's information.

Update User Information:

Input: Valid user data for update.

Expected Output: HTTP 200 OK response with updated user information.

User Avatar API View

Purpose:

To get and update user avatar.

Test Cases:

Get User Avatar:

Input: None.

Expected Output: HTTP 200 OK response with the user's avatar.

Update User Avatar:

Input: Valid avatar image data.

Expected Output: HTTP 200 OK response with updated user avatar.

User Bookmark API View

Purpose:

To get, create, and delete favorite recipes for a user.

Test Cases:

Get User Bookmarks:

Input: None.

Expected Output: HTTP 200 OK response with the user's bookmarked recipes.

Create User Bookmark:

Input: Valid recipe ID.

Expected Output: HTTP 200 OK response indicating successful bookmark creation.

Delete User Bookmark:

Input: Valid recipe ID.

Expected Output: HTTP 200 OK response indicating successful bookmark deletion.

Password Change API View

Purpose:

To change the password for an authenticated user.

Test Cases:

Valid Password Change:

Input: Valid old and new passwords.

Expected Output: HTTP 200 OK response indicating successful password change.

Invalid Password Change:

Input: Invalid old or new password (e.g., incorrect old password, weak new password).

Expected Output: HTTP 400 Bad Request response with error details.

Recipe Application Test Report

Introduction

This test report provides an overview of the testing conducted on the upgraded recipe application. The testing includes functional testing of all APIs, ensuring their correctness, robustness, and adherence to requirements.

Test Environment

Application Version: [Version number or commit hash]

Testing Framework: [e.g., pytest, Django Test Framework]

Environment: [Local, Development server, Testing server]

Test Cases

1. Endpoint: /api/recipes/

Test Case 1: Retrieve All Recipes

Request Type: GET

Expected Status Code: 200 OK

Expected Response:

**[ { "id": 1, "title": "Spaghetti Carbonara", "author": "John Doe", "description": "Classic Italian pasta dish with eggs, cheese, bacon, and pepper.", ... }, ...]**

Test Steps:

Send a GET request to the endpoint /api/recipes/.

Verify that the response status code is 200.

Verify that the response body contains a list of recipes.

Test Case 2: Create a New Recipe (Unauthorized)

Request Type: POST

Expected Status Code: 401 Unauthorized

Test Steps:

Send a POST request to the endpoint /api/recipes/ without authentication.

Verify that the response status code is 401.

2. Endpoint: /api/recipes/{recipe\_id}/

Test Case 3: Retrieve Recipe by ID (Valid ID)

Request Type: GET

Expected Status Code: 200 OK

Expected Response:

**{**

**"id": 1,**

**"title": "Spaghetti Carbonara",**

**"author": "John Doe",**

**"description": "Classic Italian pasta dish with eggs, cheese, bacon, and pepper.",**

**...}**

Test Steps:

Send a GET request to the endpoint /api/recipes/1/.

Verify that the response status code is 200.

Verify that the response body contains the details of the recipe with ID 1.

Test Case 4: Retrieve Recipe by ID (Invalid ID)

Request Type: GET

Expected Status Code: 404 Not Found

Test Steps:

Send a GET request to the endpoint /api/recipes/9999/ with an invalid recipe ID.

Verify that the response status code is 404.

Conclusion

The testing conducted on the recipe application ensures that all APIs function correctly and meet the specified requirements. Further testing may be required for edge cases and error handling scenarios.