Test Plan and Report

Based on the following template/requirements:
https://drive.google.com/file/d/1hEGwTfw2rWCCfKVIPeDrpCSGH-5I3wnH/view

Heading

Test Plan and Report
Product name: WebWatch

Testing Methodology

We use a combination of automated testing and manual testing throughout development. Unit tests test our backend API routes, and manual testing is used for our frontend interface. This document outlines our most common and important tests from each of these categories.

Automated testing consists of formatting checks for both frontend and backend source code, frontend build success, and backend API unit tests using Pytest. Each of these tests is run against open pull requests in our GitHub repository to ensure a certain level of quality before merging into our main branch, which automatically deploys to our production server. Backend tests can be run locally during development by running `pytest` in the backend directory.

Frontend testing is typically done by starting the frontend and backend development servers and opening the web browser to https://localhost:5173 to view the web interface locally. Building the frontend can be tested locally by running `npm run build` in the frontend directory. Manual testing typically consists of creating an account logging in, and creating, modifying, and deleting tasks while observing their notification behavior. We also test our deployed instance manually after changes to our main branch, by visiting https://webwatch.live in a web browser.

System Test Scenarios by User Story

As a user, I would like to receive an email notification when a webpage's content has changed.

1. Create an account/login to https://webwatch.live
2. On /tasks, use the top text entry field to enter a URL, name, and interval. Click the bell icon to create the task.
3. Once the task is created, monitor email for a notification when the web page's content changes.

As a user, I would like a web interface that allows me to create and log in to my own account.

```
def register_user():
   response = client.post(
       "/users/register",
       json={"email": "test@webwatch.live", "password":
"Password1233!"},
   )
   assert response.status_code == 201
   return response.json()["access_token"]



def test_user_register_fail_duplicate():
   # Test user registration with duplicate username
   response = client.post(
       "/users/register",
       json={"email": "test@webwatch.live", "password":
"Password1233!!!"},
   )
   assert response.status_code == 409


def test_user_login_fail():
   # Test user login with wrong password
   response = client.post(
       "/users/login",
       data={
           "grant_type": "password",
           "username": "test@webwatch.live",
           "password": "WrongPassword1233!!",
       },
       headers={"Content-Type": "application/x-www-form-urlencoded"},
   )
   assert response.status_code == 400
   assert response.json() == {"detail": "Incorrect email or
password"}


def test_user_login_success():
   # Test user login with correct password
```

```
    response = client.post(
        "/users/login",
        data={
            "grant_type": "password",
            "username": "test@webwatch.live",
            "password": "Password1233!",
        },
        headers={"Content-Type": "application/x-www-form-urlencoded"},
    )
    assert response.status_code == 200
```

As a user, I would like a dashboard to create and modify monitoring tasks in my account. / As a user, I would like a graphical interface to view monitoring jobs in my account.

1. Create an account/login to https://webwatch.live
2. On /tasks, use the top text entry field to enter a URL, name, and interval. Click the bell icon to create the task.
3. Verify the task has been created and added to the "Running" section of the task list.
4. Click the 3 vertical dots to the right of a task for options to change the task's running status and details, or to delete it.

As a user, I want better account privacy and security, such that others aren't able to access or modify my tasks.

```
def test_task_retrieve_list():
    # Test task list retrieval
    response = client.get("/tasks", headers={"Authorization": "Bearer
" + access_token})
    assert response.status_code == 200


def test_task_retrieve_no_auth():
    # Test task list retrieval without authorization
    response = client.get("/tasks")
    assert response.status_code == 401
    assert response.json() == {"detail": "Not authenticated"}
```

As a user, I’d like to be able to reset my email and password.

```python
def test_user_reset_password_success():
    # Test password reset
    response = client.post(
        "/users/login",
        data={
            "username": "test@webwatch.live",
            "password": "Password1233!",
        },
        headers={"Content-Type": "application/x-www-form-urlencoded"},
    )
    assert response.status_code == 200
    access_token = response.json()["access_token"]
    response = client.post(
        "/users/reset_password",
        json={
            "new_password": "NewPassword1233!!",
            "confirm_password": "NewPassword1233!!",
        },
        headers={"Authorization": "Bearer " + access_token},
    )

    assert response.status_code == 200


def test_user_reset_password_mismatch():
    # Test password reset with mismatched passwords
    response = client.post(
        "/users/login",
        data={
            "username": "test@webwatch.live",
            "password": "NewPassword1233!!",
        },
    )
    headers = ({"Content-Type": "application/x-www-form-urlencoded"},)
    assert response.status_code == 200
    access_token = response.json()["access_token"]

    response = client.post(
        "/users/reset_password",
        json={
            "new_password": "NewPassword1233!!",
            "confirm_password": "MismatchedPassword1233!!",
        },
        headers={"Authorization": "Bearer " + access_token},
    )

    assert response.status_code == 422
```