SWE573 Project Report

Atakan Özguner

December 21, 2024

Course

SWE573 Software Development Practice

Deployment URL

`http://34.234.74.196:3000`

Git Repository URL

`https://github.com/atakanozguner/swe573-ozguner`

Git Tag URL

`https://github.com/atakanozguner/swe573-ozguner/releases/tag/v0.9`

Honor Code

Related to the submission of all the project deliverables for the SWE573 Fall 2024 semester project reported in this report, I declare that:

- I am a student in the Software Engineering MS program at Boğaziçi University and am registered for SWE573 course during the Fall 2024 semester.
- All the material that I am submitting related to my project (including but not limited to the project repository, the final project report, and supplementary documents) have been exclusively prepared by myself.
- I have prepared this material individually without the assistance of anyone else with the exception of permitted peer assistance which I have explicitly disclosed in this report.

Name: Atakan Özgüner
Signature: ________________________

Contents

1 Project Details

1.1 Overview

The project, **SWE573 App**, is a web-based platform designed and implemented using modern software development practices, emphasizing end-to-end project delivery. It allows users to create, share, and explore any imaginable or obtainable item, while managing posts and interactions through a user-friendly interface. The development process included issue tracking, version control, and testing to ensure functionality and reliability, along with containerized deployment for ease of setup and scalability. This project not only delivers a functional application but also showcases effective software development management practices, including testing and deployment strategies.

1.2 Software Requirements Specification

1.2.1 Introduction

1.1 Purpose The purpose of this document is to define the Software Requirements Specification (SRS) for the **SWE573 App**. This document provides an outline of the functional, non-functional, and design requirements to ensure the proper development and deployment of the system.

1.2 Scope The **SWE573 App** allows users to create, view, interact with, and search for posts. It provides features such as:

- User registration and login.
- Post creation, editing, and deletion.
- Interaction features like commenting, voting, and showing interest in posts.
- A "Hot" page that displays posts sorted by interest count.
- Persistent storage and AWS-based deployment.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification.
- **CORS**: Cross-Origin Resource Sharing.
- **AWS**: Amazon Web Services.
- **EC2**: Elastic Compute Cloud.
- **Post**: A user-created content item that includes metadata, an optional image, and descriptive information.

1.4 References

- Docker Documentation: `https://docs.docker.com`
- AWS Documentation: `https://aws.amazon.com/documentation`

1.5 Overview This document covers the system's objectives, functionalities, and constraints. It ensures all stakeholders have a clear understanding of the project scope.

1.2.2 Overall Description

2.1 Product Perspective The system is a standalone web application comprising:

- A frontend built with React.
- A backend API developed using FastAPI.
- A PostgreSQL database for data persistence.

2.2 Product Functions The system provides the following functionalities:

- User authentication and session management.
- Post creation with optional metadata and images.
- Interaction features (comments, votes, interest toggling).
- Search functionality for posts.
- A "Hot" page displaying the most interacted-with posts.

2.3 User Characteristics

- **Users**: Users can register, log in, create posts, comment, vote, and search for content.

2.4 Constraints

- The system must be deployable on AWS EC2.
- Must handle at least 50 concurrent users without significant performance degradation.
- Backend must comply with CORS requirements for frontend communication.

2.5 Assumptions and Dependencies

- Users have access to a modern web browser.
- The system uses Docker for containerization.
- The server has persistent storage for images and database files.

1.2.3 Specific Requirements

3.1 Functional Requirements User Management

- Users must register with a unique username and password.
- Users can log in and maintain sessions.
- Users can log out and clear session data.

Post Management

- Users can create posts with:
  - Title (required).
  - Description (required).
  - Metadata (optional): material, dimensions, weight, etc.
  - Tags with labels, descriptions, and links.
  - Uploadable image.
- Users can edit or delete their posts.
- Post creators can mark posts as "resolved."

Interaction

- Users can comment on posts.
- Users can upvote or downvote comments.
- Users can toggle interest in posts, updating the interest count.

Search

- Users can search posts by keywords in the title or description.

Hot Page

- Displays posts sorted by interest count in descending order.

3.2 Non-Functional Requirements Performance

- System should respond to API requests within 500ms under normal load.

Security

- User passwords must be hashed and securely stored.
- The system should comply with CORS for secure frontend-backend communication.

Availability

- The system must have 99% uptime when deployed on AWS.

Scalability

- Should handle at least 50 concurrent users.

Persistence

- Use Docker volumes for persistent storage of database and images.

1.3 Appendix

- **Backend Framework**: FastAPI.
- **Frontend Framework**: React.
- **Database**: PostgreSQL.
- **Containerization**: Docker.
- **Cloud Deployment**: AWS EC2.
- **Environment Variables**: Used for backend URL configuration and sensitive data.

2 Initial Design Images

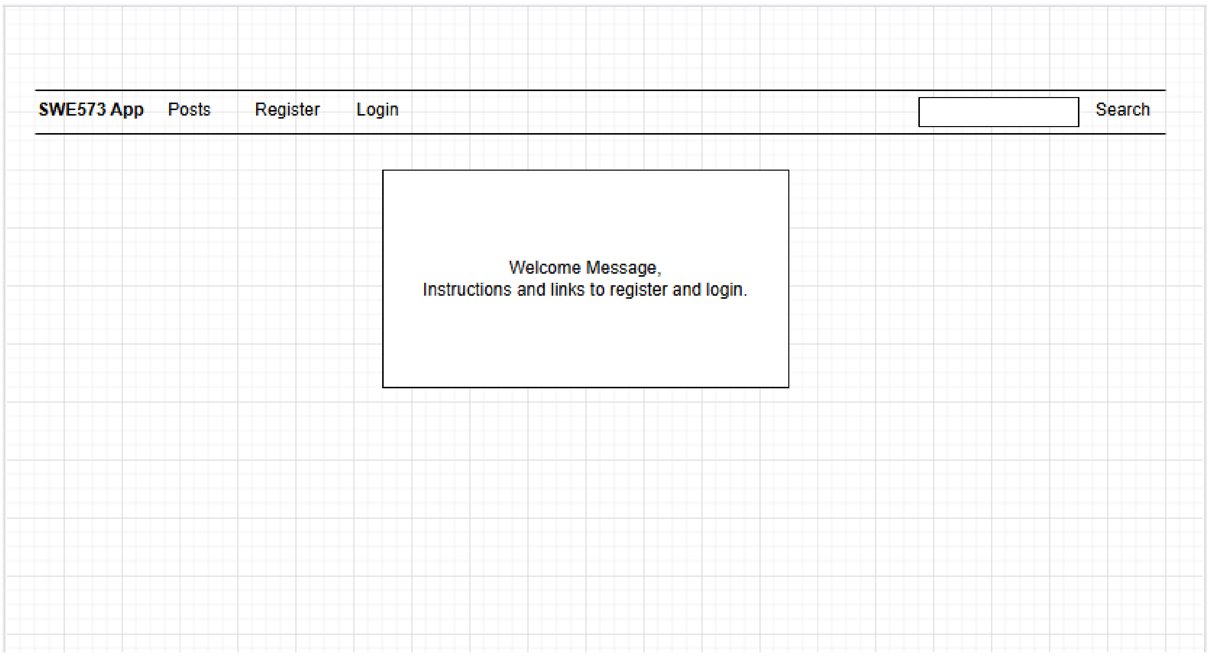

Figure 1: Homepage

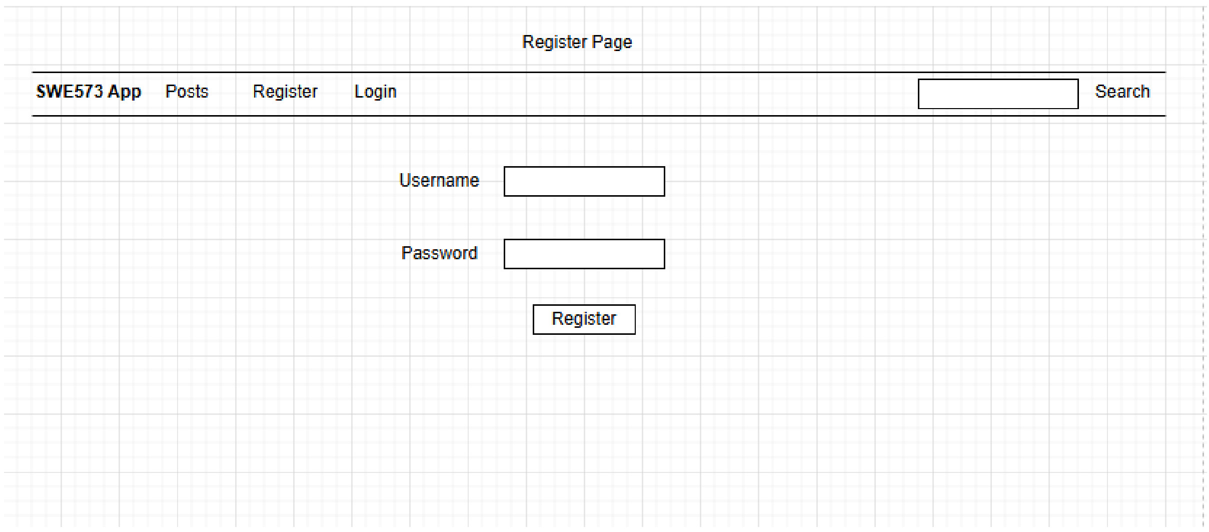

Figure 2: Register

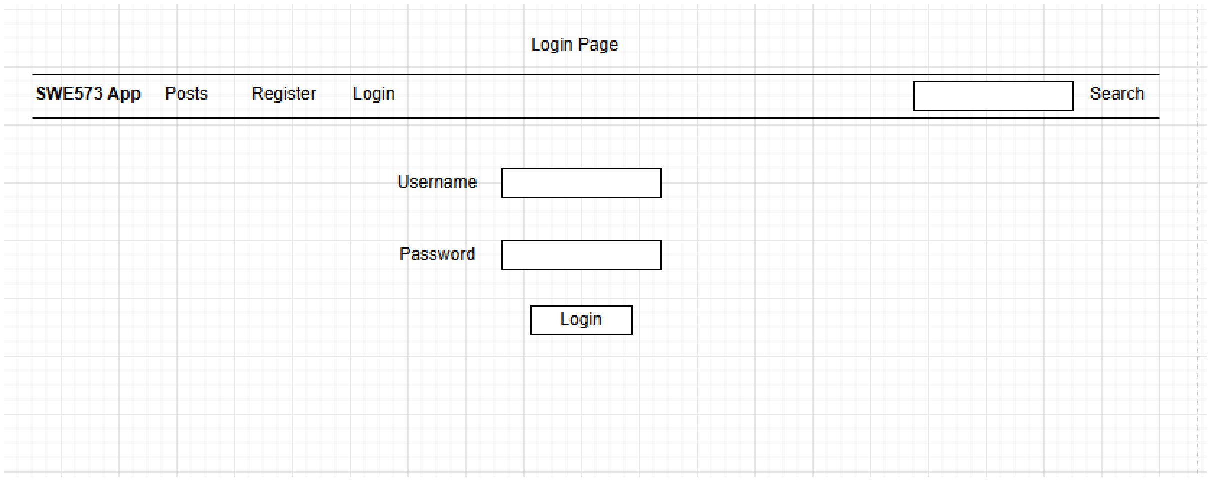

Figure 3: Login

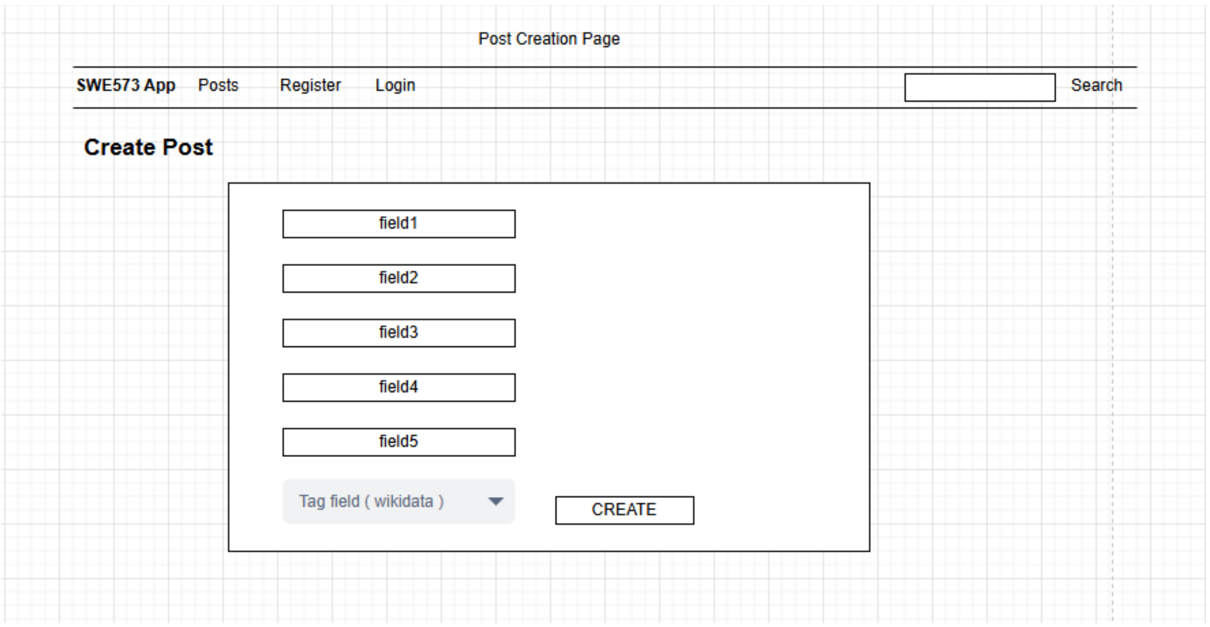

Figure 4: Post Creation

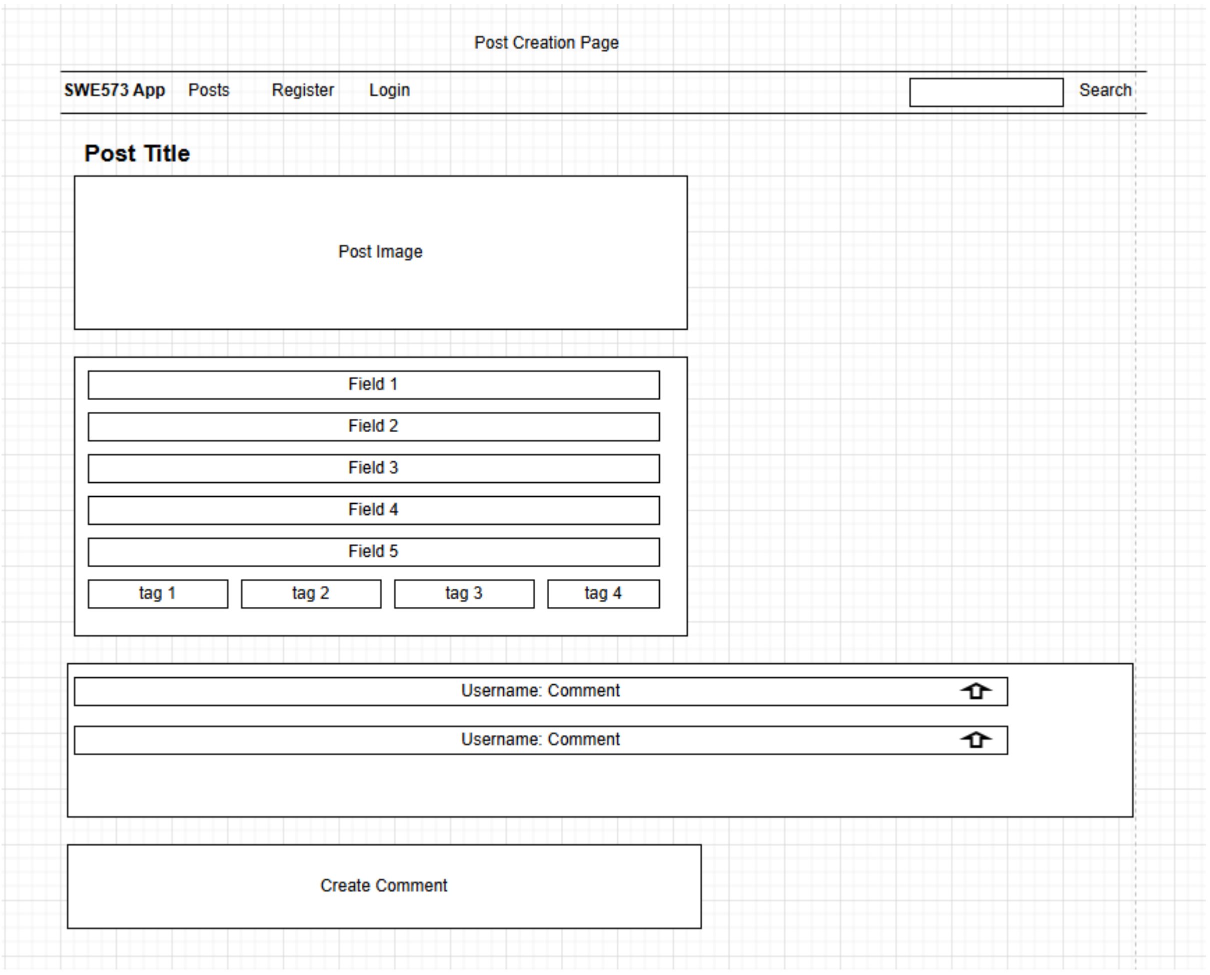

Figure 5: Post Details

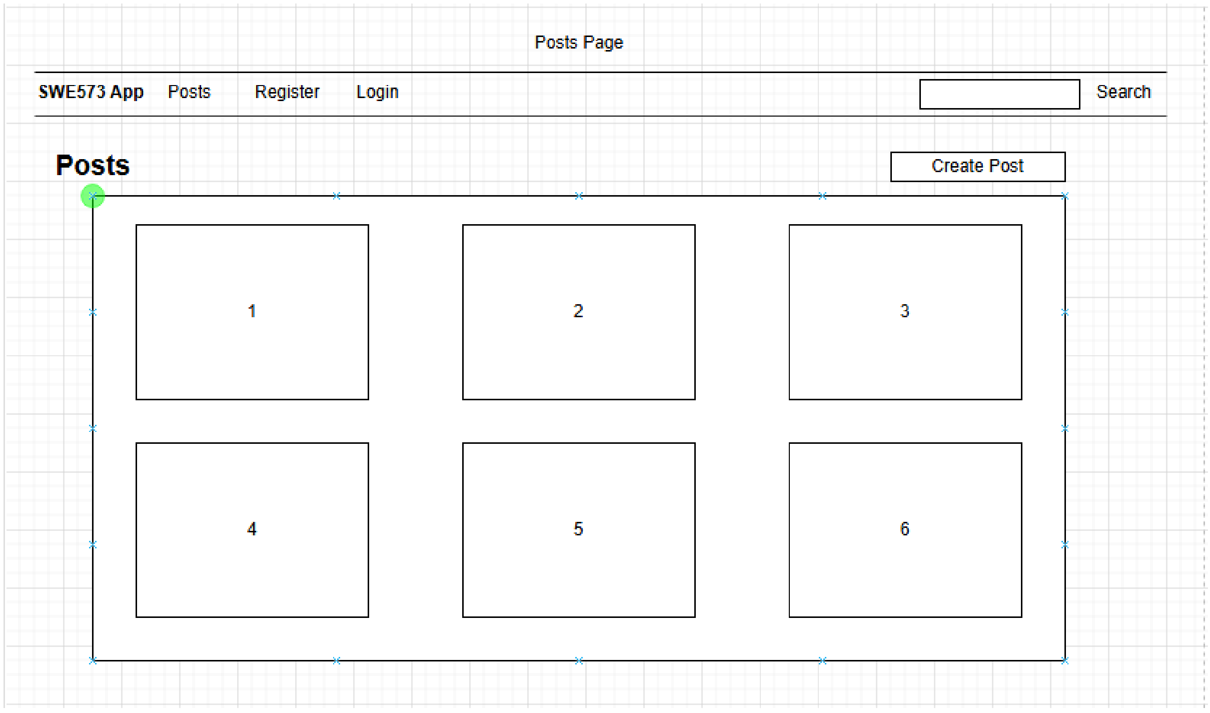

Figure 6: Posts Display

3 Status

The SRS document and Requirements list can also be accessed from the following URL's:

https://github.com/atakanozguner/swe573-ozguner/wiki/SRS-List-%E2%80%90-Final
https://github.com/atakanozguner/swe573-ozguner/wiki/Requirements-List-%E2%80%90-final

The requirements that are present in the requirements list are fulfilled. Software is deployed on AWS EC2 instance, with some little faults, that are mentioned on the User Tests. Some features (register and login) seem to be laggy and may require several user inputs to fulfill, and this bug is created as issue, to be resolved, mocking the case where development continues after this course.
Some basic unit tests are introduced under a new branch for testing, test reports, coverage reports, and user tests are going to be included in this report.

3.1 Deployment URL

Deployment URL is: http://34.234.74.196:3000/
The app is dockerized. On AWS EC2 instance application is running (after including necessary .env content), only via docker-compose up.

3.2 Installation Instructions

Installation Instructions can also be found here:
https://github.com/atakanozguner/swe573-ozguner/wiki/Installation-Instructions

4 Installation Instructions

```
git clone https://github.com/atakanozguner/swe573-ozguner.git

cd swe573-ozguner/
```

Listing 1: Installation Steps

create necessary .env files.
/frontend/swe573-app/.env:

```
REACT_APP_BACKEND_URL=http://localhost:8000
```

/backend/.env:

```
SECRET_KEY="3
   da63acb346a5eb4c95378ee83b90dcbe5229673c7eb9fa19b40c4745ac7d786"
ALGORITHM="HS256"
ACCESS_TOKEN_EXPIRE_MINUTES=30
DATABASE_URL="postgresql://postgres:password@db/swe573_database"
```

5 User Manual

Step 1: Access the Application

1. Open your web browser.
2. Navigate to the following link: `http://34.234.74.196:3000/`.

Step 2: Register an Account

1. Click on the **"Register"** button in the navigation bar.
2. Fill in the required fields:
   - **Username**: Enter your desired username.
   - **Password**: Enter a password.
3. Click the **"Register"** button.

Note: On AWS Deployment, registration and login may be laggy and might require several consecutive clicks for the action to take effect on both the **Register** and **Login** pages.

Step 3: Login to Your Account

1. Click on the **"Login"** button in the navigation bar.
2. Enter your credentials:
   - **Username**: The username you registered with.
   - **Password**: Your password.
3. Click the **"Login"** button.
4. If successful, you will be redirected.

Note: On AWS Deployment, registration and login may be laggy and might require several consecutive clicks for the action to take effect on both the **Register** and **Login** pages.

Step 4: View Posts

1. After logging in, navigate to the **"Posts"** page using the navigation bar.
2. Browse through the posts displayed on the page.

Step 5: Create a New Post

1. Click on the **"Create Post"** button in the navigation bar.
2. Fill in the form:
   - **Title**: Enter a title for your post.
   - **Description**: Write a description for your post.
   - **Tags**: Add relevant tags (optional).
     - **Wikidata Search**: As you enter your prompt, Wikidata search results will appear for you to select from, with descriptions. Select the correct tag to add for your post.
   - **Image**: Upload an image (optional).
   - Additional details like dimensions, weight, and color (optional).
3. Click **"Create Post"** to publish your post.
4. Your post will appear on the **Posts** page.

Step 6: View Hot Posts

1. Navigate to the **"Hot"** page using the navigation bar.
2. Browse the most popular posts sorted by user interest.

Step 7: Search for Posts

1. Use the search bar located in the navigation bar.
2. Enter a keyword related to the posts you want to find.
3. Click **"Search"**.
4. Relevant posts will be displayed.

Step 8: View Post Details

1. Click on a post's title or thumbnail on the **Posts** or **Hot** pages.
2. View the detailed information about the post.
3. Add comments or interact with the post.

Step 9: Add Comments

1. On a post's detail page, scroll to the comments section.
2. Enter your comment in the input box.
3. Click **"Submit Comment"** to post your comment.

Step 10: Vote on Comments

1. Use the **"Upvote"** or **"Downvote"** buttons on a comment to vote.
2. The comment's score will update accordingly.

Step 11: Logout

1. Click on the **"Logout"** button in the navigation bar.
2. You will be logged out and redirected to the home page.

6 Test Results

6.1 Unit Test Results

`https://github.com/atakanozguner/swe573-ozguner/wiki/Basic-Unit-Tests-%E2%80%90-Framework-&-Report-Generation` *Report generated on 21-Dec-2024 at 21:59:08 by pytest-md.*

6.1.1

Coverage

File	**Stmts**	**Miss**	**Cover**
`app/__init__.py`	0	0	100%
`app/database.py`	15	5	67%
`app/main.py`	88	30	66%
`app/models.py`	82	5	94%
`app/routers/post.py`	132	78	41%
`app/schemas.py`	102	3	97%
`app/tests/conftest.py`	0	0	100%
`app/tests/tests.py`	46	0	100%
`app/utils.py`	70	42	40%
TOTAL	**535**	**163**	**70%**

Table 1: Coverage Report

Coverage HTML written to `dir htmlcov`.

6.1.2 Summary

- **7 tests ran** in 1.85 seconds.
- **7 passed.**

6.2 User Test Results

https://github.com/atakanozguner/swe573-ozguner/wiki/User-Tests

Test Case	**Objective**	**Steps**	**Outcome**	**Result**	**Notes**
1	Verify user registration	Enter details & click *"Register."*	Redirected to login page.	Partial	"Success" message displayed on local tests, doesn't show up on AWS Deployment.
2	Test login functionality	Enter credentials & click *"Login."*	Redirected to *"Posts"* page.	Partial	Worked flawlessly on local, had to click several times on AWS Deployment.
3	View posts	Navigate to *"Posts"* page.	Posts displayed.	Success	No issues.
4	Search posts	Enter keyword & click *"Search."*	Relevant posts displayed.	Success	No issues.
5	Verify *"Hot Posts"* page	Navigate to *"Hot Posts"* page.	Sorted by popularity.	Success	Sorting worked as expected.
6	Create a post	Fill form & click *"Create Post."*	Post added to *"Posts"* page.	Success	No issues.
7	Comment on posts	Enter & submit a comment.	Comment appears below post.	Success	Works seamlessly.
8	Test comment voting	Click *"Upvote"* or *"Downvote."*	Score updates.	Success	No issues.
9	Verify logout	Click *"Logout."*	Redirected to home page.	Success	Works as expected.

Table 2: User Test Cases