SWE573 - Software Development Practice

29.05.2023

Final Project:
MemoShare

github.com/dbaslan/SWE573-DBA

Deniz Baran ASLAN

2021719183

HONOR CODE

Related to the submission of all the project deliverables for the SWE573 2023 Spring semester project reported in this report, I Deniz Baran Aslan declare that:

- I am a student in the Software Engineering MS program at Bogazici University and am registered for Swe573 course during the Spring 2023 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.

Deniz Baran Aslan

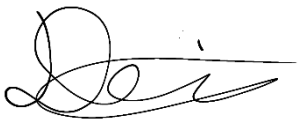


Table of Contents

Overview

MemoShare is a website built with Django, made for the purpose of sharing memories online with other users. It uses a PostgreSQL database to store information.

Software Requirements Specification

Below is a list of the project requirements, taken directly from the repository wiki.

Functional Requirements (prefix F)

1. The system shall be available through web browsers.
2. The system shall allow users to create accounts with a username and a password.
3. The system shall allow users to update their account information.
4. The system shall allow users to upload a profile picture.
5. The system shall allow users to delete their accounts.
6. The system shall allow users to make text-based posts.
7. The system shall allow users to add tags to their posts.
8. The system shall allow users to like posts.
9. The system shall allow users to comment on each other's posts.
10. The system shall allow users to follow each other.
11. The system shall allow users to delete their posts.
12. The system shall allow users to edit their posts.
13. The system shall allow users to upload images.
14. The system shall tag posts with date and time.
15. The system shall sort posts by user, tags, date, and location.
16. The system shall have a search functionality.
17. The system shall filter searches by tags, date and location.

Non-Functional Requirements (prefix NF)

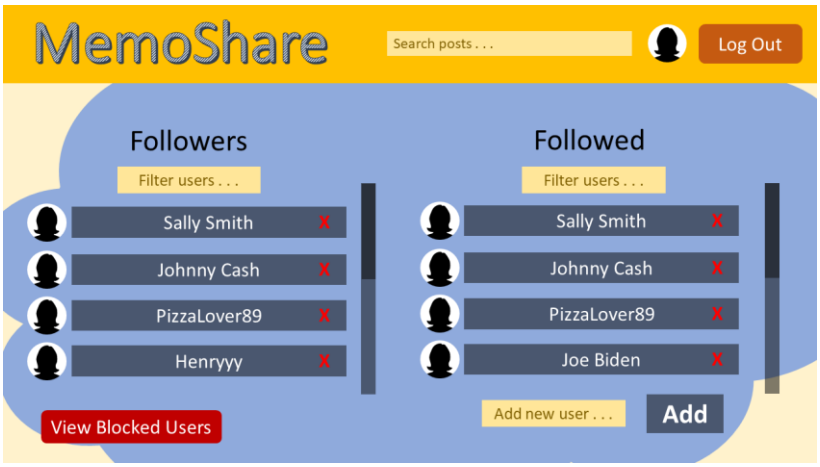
1. The system shall have at least 99% uptime.
2. The system shall be available in English.
3. The system shall have a dark and a light theme.
4. The system shall encrypt user data.

Design (Software & Mockups)

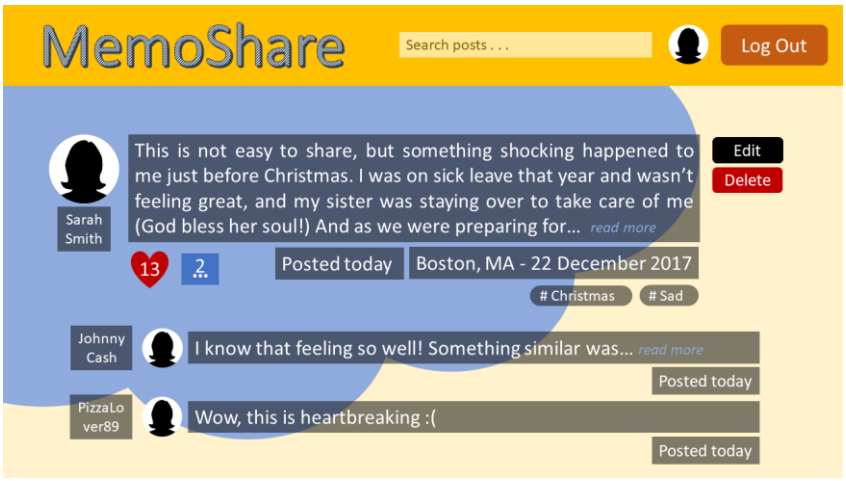
Below are the mockups initially created before the project was implemented:

User Settings Screen

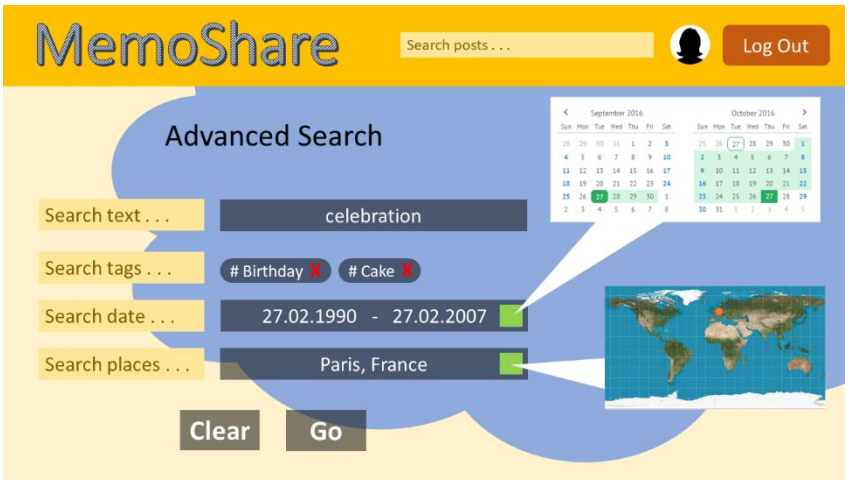
Feed Screen



Follower Screen



Post Scree

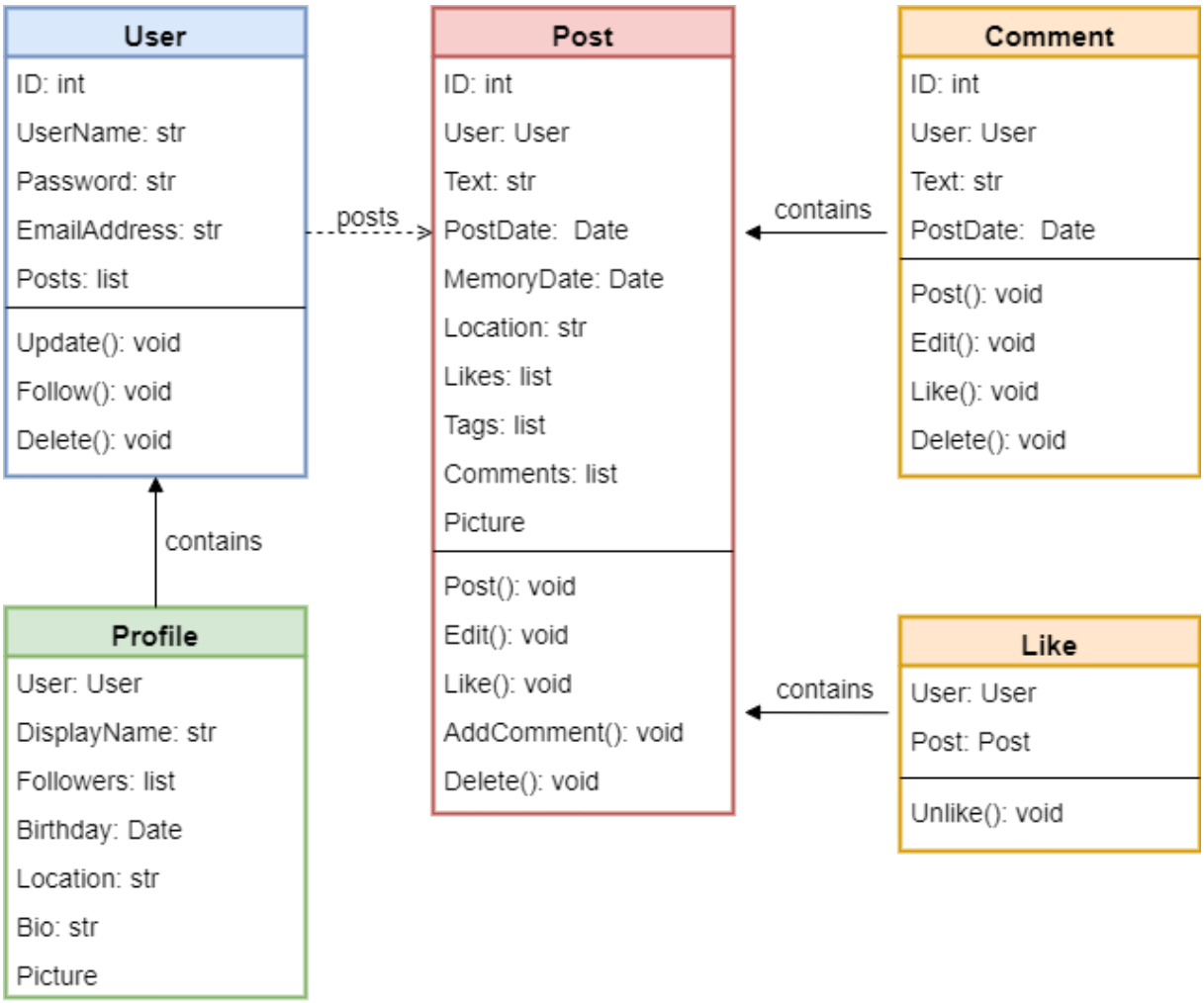


Search Screen

Profile Screen

Welcome Screen

Below is the original diagram that illustrates the class-based structure of the project contents.



Project Status

The project meets the requirements for most of the core functionalities:

- Account creation, update and deletion.
- Post creation, update and deletion.
- Searching posts by name, content, tag, time, and location.
- Adding tags to post (through taggit).
- Posting images (through Pillow).
- Visualization of memory location on map (through Django-easy-maps).
- User following and profile editing.

The project lacks two important features: geolocation and map search. These features were unable to be implemented due to a lack of technical experience. Locations are stored as strings. Another issue is that the time-based filter and search utility, while functional, is lacking in available options. As of release, only direct search (i.e. exact day) and interval search (i.e. between 19.09.2002 – 22.04.2020) are available. But further options, such as search by decade, year, season or month have not been implemented. The most critical issue is the deployment, discussed in the next section.

Deployment Status

Despite all efforts, the project was unable to be deployed online. While the application runs locally without issue, all attempts to host it online have failed, including on PythonAnywhere, Google Cloud App Engine, Amazon Elastic Beanstalk, and Amazon EC2 instances. The Dockerized container runs without issue, which is visible from the logs. However, the container simply does not allow access through browsers despite many attempts, which completely prevented containerized deployment. The same is true for direct deployment without Dockerization, where each Cloud provider failed to deploy, facing various errors.

User Manual

The system can be deployed locally through the repository on any system that has Python and pip installed. Note that the local deployment will use the Sqlite3 backend instead of PostgreSQL. The instructions below illustrate how to run the server:

- Open the project directory (cd SWE573-DBA/MemoShare)
- Copy the .env file provided in zip file into project directory for credentials
- Install project dependency modules (pip install -r requirements)
- Run server (python manage.py runserver)
- Access website (through url http://127.0.0.1:8000/)

Once the website is accessed, users can create their own account to test the features, or log into an existing account with the credentials provided below:

Username: CrazyFrog2
Password: asdgsfSGQR132

Users can also log into the admin account with the following credentials:

Username: admin
Password: admin

This account has admin privileges, and the user can access the administration panel through http://127.0.0.1:8000/admin for further information.

Tests

Unit tests are available in the repository in the Python script named tests.py. The script can be run to repeat the available tests locally.

Demonstration

A video demonstration is available on Youtube, accessible through the repository.