SWE-573 Software Development Practice

2022 Fall Project

Final Project Report

Project name: MyInfoSpace Web Application

Submitted on: 03.01.2023

Submitted to: Prof. Suzan Uskudarli

Report by: Maria Asif

Git repository: https://github.com/MariaAsif135/SWE573-Project/tree/main/ProjectCode

Git tag version: https://github.com/MariaAsif135/SWE573-Project/releases/tag/v0.9

Deployment URL: ec2-44-207-30-21.compute-1.amazonaws.com/welcome

Honor Code:

I Maria Asif declare that: - I am a student in the Software Engineering MS program at Bogazici University and am registered for Swe573 course during the 2022 Fall 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 **Signature**

-User ids and other information needed when testing my deployed system:

Username: mar Password:123

Admin Username: 26078273 Password: Mummy123daddy

New account can be created too, with desired username and email.

- I declare that all work in the project that I am submitting is performed by yourself, and the declaration of any 3rd party software is abided by the licenses.

Table of contents

Overview

In this project, a web application was designed that allows the users to save and share links for future purposes. It is an information sharing platform where users can post and like posts. They can also follow users whose posts they want to keep track of. It is designed to save time so that the users can gather all their important links in one place and save it for later. Users can also search for their favorite users and visit their profile to see more of their content. Moreover, they can choose a category and stay uptodate with the latest news.

Software Requirements Specification

The requirements specification of the application can be divided into two parts: User requirements and system requirements

System requirements:

- The system must allow the user to save the links of the resources
- The system must allow the user to access the saved links
- The system must allow the user to accumulate information in user account
- The system must allow the user to group similar subject
- The system must allow the user to add labels to the saved links
- The system must allow the user to choose their privacy settings
- The system must allow the user to access other public accounts
- The system must allow the user to mark the read content as READ
- The system must allow the user to discuss the topic with followers
- The system must allow the user to request to follow other people
- The system must allow the user to see the followers shared links
- The system must suggest people to follow with similar labels
- The system must suggest people to follow with similar saved resources links
- The system must remind the user of the saved content
- The system must show the number of followers that user has
- The system must show the number of people the user is following
- The system must show the number of posts that the user has posted
- The system must allow the user to sign in
- The system must allow the user to sign up
- The system must allow the user to logout.
- The system must filter the post according to the people that user follows.

User requirements:

- The user must be able to sign up into the system.
- The user must be able to log in to the system

- The user must be able to post a link of his/her choice.
- The user must be able to add description to the link added.
- The user must be able to like other posts
- The user must be able to see other people's post.
- The user must be able to search for other users
- The user must be able to visit other people's profiles
- The user must be able to see other users' posts on their profile
- The user must be able to follow other users
- The user must be able to see the number of posts that the users have posted
- The user must be able to logout of the system

Design (Software)

In order to design this application, Django 3.8 was used as a backend development platform. Python was chosen as the programming language. HTML, CSS and bootstrap templates were used for frontend development. MySQL database was used for registration, sign in and user authentication of the application. the Application was dockerized using Docker desktop [] and was deployed on AWS []. The application was also dockerized on cloud.

Design (Mockups and UML diagrams)

Figure 1: Welcome page:

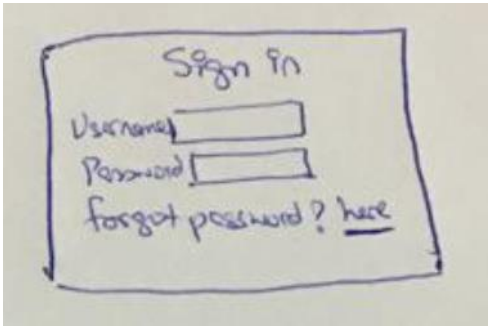

Figure 2: Sign In page

Figure 3: Sign Up page

Figure 4: Home page after logging in

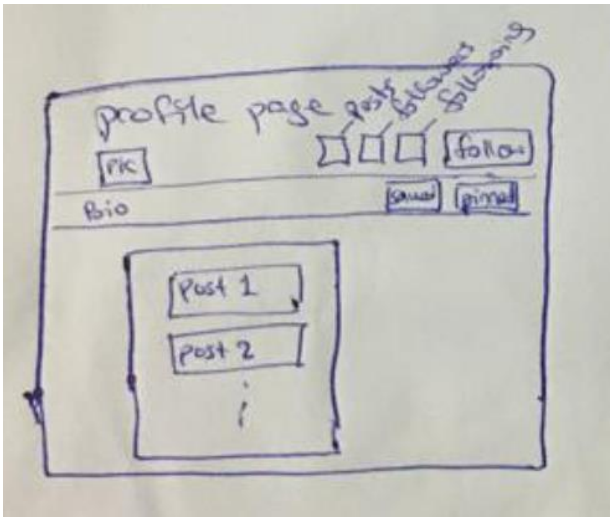

Figure 5: User Profile page

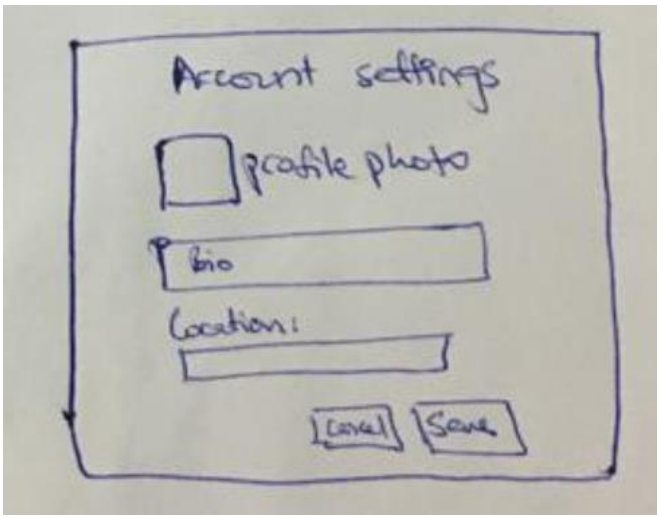

Figure 6: User Account settings page

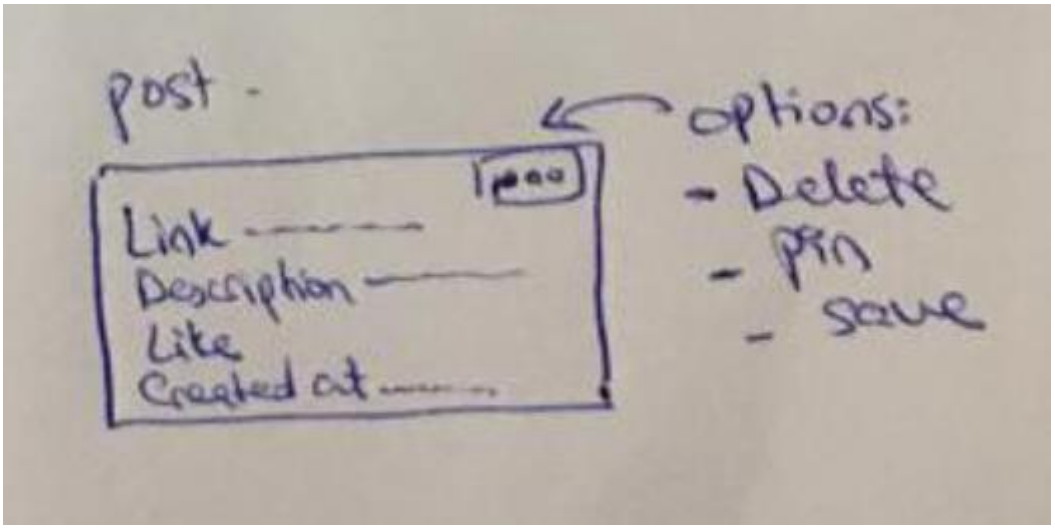

Figure 7: Posts area

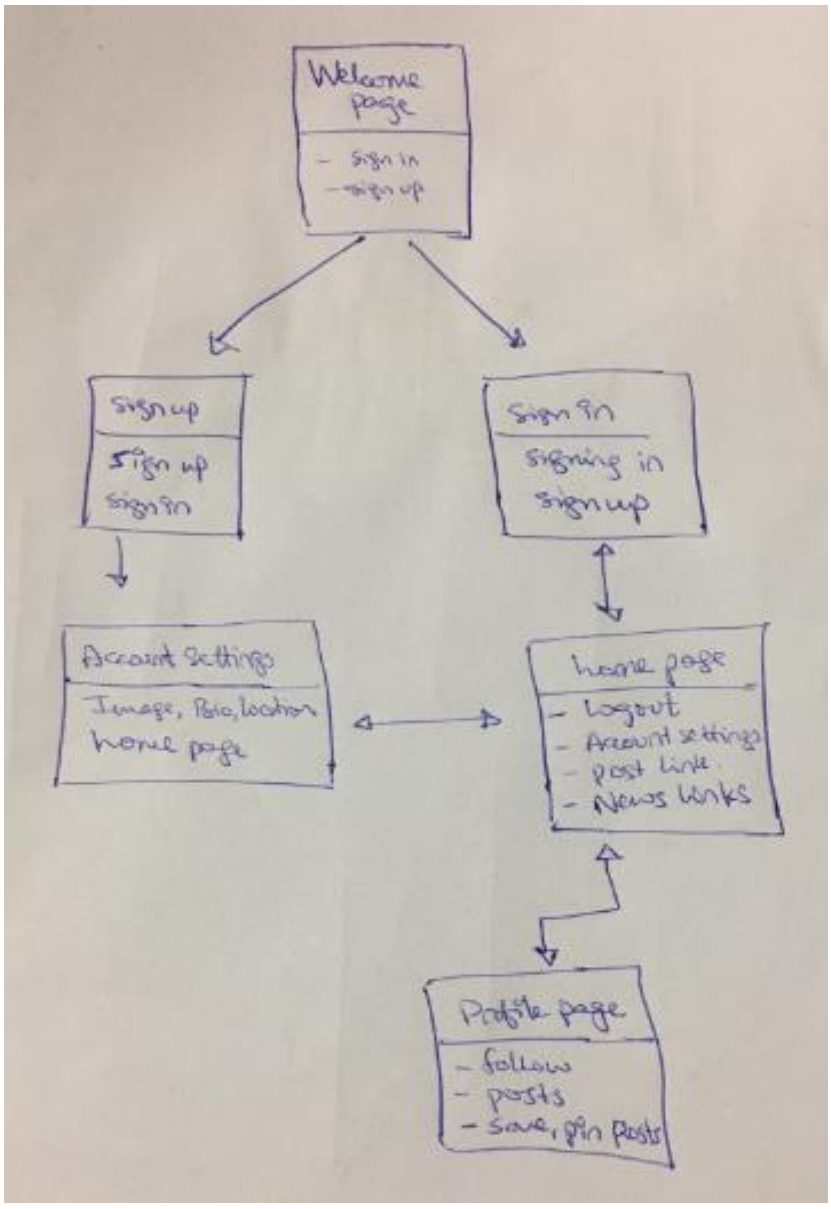

Figure 8: UML diagram of the application

Status of the project

Tasks	Status
Login	Completed
Sign Up	Completed
Adding MySQL database	Completed
Account Settings	Completed
Profile settings	Completed
Posting links	Completed
Liking posts	Completed
Following users	Completed
Keeping count of followers and posts	Completed

Deleting posts	In Progress
Welcome Page	Completed
Filtering feed according to followed users	In Progress
Exploring latest news	Completed
Saving posts	In Progress
Pinning posts	In Progress
Showing list of followers	In Progress
logout	Completed
Searching users	Completed

Status of Deployment

The project is successfully dockerized and deployed on AWS app services.

System manual

The project contains many python files.

Settings.py: The database related settings and all the other code settings are made here

url.py: the URL of each page and assigned function to the URL is defined

views.py: the function that will be executed when the URL is implemented is written in this file.

Models.py: Models are created here. These models are used to save data and use them for later purposes.

Templates folder: all html files are contained here

Static folder: all images and files that enhance the front end of application are included in this folder.

Requirements.txt file: this file contains all the packages that need to be installed to run this project in another environment.

To run this application on local machine, follow the steps below:

Install and start the docker desktop

Prepare docker-compose.yml file with container definitions for the application and the database. The container for MySQL must have a data folder in the docker volume and port 3306 must be mapped to an unused port on the host

for debugging. We setup the app container and provide it with the application files. Port mapping between the database and application must be correctly setup their respective containers (credentials in settings.py and views.py must match with the docker credentials for the database container). The name of the docker volume specified in docker compose file must not clash with the existing volumes' names on the host.

The host ports chosen for the app and the database must be available on the host machine.

Database ports must match the specified container side ports for the database container in the docker compose file the following changes to the code must be made:

In Settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mariadb',
        'USER': 'root',
        'PASSWORD': 'Mummy123daddy',
        'HOST': 'localhost',
        'PORT': '3306'
    }}
```

Figure 9: settings to run database on local machine

```
DATABASES = {

    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'django',
        'USER': 'root',
        'PASSWORD': '',
        'HOST': 'mysql_db',
        'PORT': '3306'
    }
}
```

Figure 10: settings for docker and AWS deployment

In views.py

```
    m=sql.connect(host="localhost", user = "root", password="Mummy123daddy", database ="mariadb")
    #m=sql.connect(host="mysql_db", user = "root", password="", database ="django")
```

Figure 11: using this line for login and signup functions for local host and using the commented line for docker and AWS. Comment the former line when using commented line for docker

In order to run the application on docker local, use the command docker-compose up –build and Docker file.

Setting up the project on local host:

1. Download and extract the zip file
2. open the project in VisuaSstudio code or any other platform.
3. open the project's directory in the platform's terminal. (Make sure that ProjectCode is the final folder. Use "cd ProjectCode" command if needed)
4. use python manage.py runserver command to run the application server
5. open the link and add "/welcome" next to the URL (Ex: http://127.0.0.1:8000/welcome)

To Setup the project on docker:

1. Download and extract the zip file
2. open the project in Visual Studio code or any other platform.
3. open the project's directory in the platform's terminal. (Make sure that ProjectCode is the final folder. Use "cd ProjectCode" command if needed)
4. Open Docker desktop
5. use 'docker-compose up --build' command to run the application server
6. open the link and add "/welcome" next to the URL (Ex: localhost:8000/welcome)
7. that's all

To run the project on cloud, use this link: ec2-44-207-30-21.compute-1.amazonaws.com/welcome

For dockerizing and deploying the application on cloud service provider (AWS in our case), following steps must be carried out:

- Creating Ec2 instance.
- Networking stuff:
  1. Attach an internet gateway through the VPC where ec2 instance is connected.
  2. Configure the route table to use the internet gateway for the incoming traffic and route this traffic to our ec2 instance.
  3. Configure the network ACL to allow traffic on the port on which the app is running.

  When we launch the instance, Virtual Private Clouds, Subnets and Network Access Control Lists are automatically created and must be configured to allow access to our application outside the AWS environment.

- Start the instance and connect.

- In the terminal:
  1. connect to project folder
  2. pull git from main branch
  3. ensure that the connection details of the project in docker compose file match the configuration (routing table, subnets, network ACL) on AWS for the ec2 instance.
  4. use “docker-compose up – build”
  5. use the public IP/DNS of the ec2 instance to access the web app outside AWS environment.

User manual

User is required to create an account using a unique username and email

After signing up, the user will be directed to account settings page where the user can set up a profile picture, write bio and mention location if user wants.

The user is then redirected to home page when he can share posts, read and like other people’s posts. User can also visit other people’s profile and follow them.

Link to project video

https://youtu.be/o_2TGxVGZ5o

Project repository

https://github.com/MariaAsif135/SWE573-Project

References

Resources used to learn Django user authentication:

https://www.youtube.com/watch?v=1UvTNMH7zDo&list=PLOeM1L4TLOx1bhDQvVrEo6HpQBMh1jEsZ&index=1

Resources used to learn Django coding:

https://www.youtube.com/watch?v=7bfk_Nj_nng&list=PLOeM1L4TLOx1bhDQvVrEo6HpQBMh1jEsZ&index=6

https://www.youtube.com/watch?v=xSUm6iMtREA&t=14165s

https://djangogirls.org/en/

Resources used to learn how to add MYSQL database:

https://www.youtube.com/watch?v=eq-e_n7lm2M&list=PLOeM1L4TLOx1bhDQvVrEo6HpQBMh1jEsZ&index=5&t=469s

Resources used to learn user testing:

https://www.youtube.com/watch?v=qwypH3YvMKc&list=PLOeM1L4TLOx1bhDQvVrEo6HpQBMh1jEsZ&index=7

Resources used to learn Docker:

https://www.docker.com/products/docker-desktop/

https://www.youtube.com/watch?v=BoM-7VMdo7s&list=PLOeM1L4TLOx1bhDQvVrEo6HpQBMh1jEsZ&index=9

https://www.youtube.com/watch?v=W5Ov0H7E_o4&list=PLOeM1L4TLOx1bhDQvVrEo6HpQBMh1jEsZ&index=8

Resources used to learn how to deploy on AWS:

https://www.youtube.com/watch?v=zs3tyVgiBQQ&list=PLOeM1L4TLOx1bhDQvVrEo6HpQBMh1jEsZ&index=12

https://www.youtube.com/watch?v=_OREV6KSEs8&list=PLOeM1L4TLOx1bhDQvVrEo6HpQBMh1jEsZ&index=13