Lami Kaan Kösesoy - 2021719093

SWE 573 Fall 2022

03.01.2023

Project: Building Website with Django Framework

Git repository: https://github.com/LamiKaan/SWE-573

Git tag version: https://github.com/LamiKaan/SWE-573/releases/tag/v0.9

Deployment URI: http://52.44.201.119/

HONOR CODE

Related to the submission of all the project deliverables for the Swe573 2022 Fall semester project reported in this report, I (Lami Kaan Kösesoy) 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.

Lami Kaan Kösesoy

TABLE OF CONTENTS

TABLE OF FIGURES

DECLARATION

I declare that all work in the project I am submitting is performed by myself, and my usage of 3rd party software during the project abides by the licenses.

USER IDS AND OTHER INFORMATION TO TEST THE DEPLOYED SYSTEM

Usernames of registered users:

- lkk (admin)
- ahmet
- sahin
- mehmet
- gencay
- cansin
- eralp
- ali
- veli

User passwords:

- For the user "lkk" → "password"
- For all other users → same with the username (e.g. password is "ahmet" for the user with username "ahmet")

REFERENCES

1. A course on Django Framework on Youtube: Python Django 7 Hour Course - YouTube
2. Official Django documentation: Django documentation | Django documentation | Django (djangoproject.com)
3. Official Docker tutorial: Overview | Docker Documentation
4. 100s of StackOverflow questions, other websites and videos all over the internet that I didn't keep track of

PROJECT DETAILS

Overview

The project is a website whose main purpose Is to provide its users to save any kind of information (primarily via usage of links) that they may encounter on the internet, so that they can access it at a later time as they wish. The website is designed in a way that the information is kept in a structured unit called “content”. Each content object has a set of attributes (such as header, tags, link etc.) which enable users to categorize the information they wish to save, which in turn makes accessing the desired saved content easier and quicker by providing users with search and filter functionalities on these attributes. Apart from search and filter, the website also offers other functionalities including some basic ones such as registering, logging in and logging out which are required to use the website, and some other more advanced functionalities such as following other users, liking and commenting on contents.

Software Requirements Specification

Creating a User Account

1. Creating a user account is required to use the application.
2. Providing real names is not mandatory.
3. User can choose any suitable (not already taken) username.
4. The system shall approve user's password as long as it conforms to length and character requirements.
5. User can reset and recover passwords.
6. There should be an admin role that is able to interfere in case of reports from other users.

Content Creation

7. User shall be able to capture a publicly accessible info and save it on their profile (in the form of a content).
8. Contents shall be public (visible by other users) by default and shall be made private if desired by the user.
9. Each user shall be given a limited amount of memory for storing info.
10. The system shall enable the modification/deletion/duplication of info after creation.
11. Once a content has been posted, it shall be modified within the next 5 minutes.
12. A user shall comment, tag, their content.
13. The user shall create their own tags to label the content.
14. The system shall enable user to attach files from their device while creating content.
15. The user shall store content as private, shared or public.

User Interactions

16. Users shall see other people's profiles as long as the profiles are not set as private.
17. Users shall follow other users.
18. Users shall comment, tag on other user's posts.
19. Application shall not provide any direct interaction (messaging etc.) between users. Interaction shall only be made through profiles and posts.

20. The system shall allow sharing of contents among users.
21. When info is public, every user shall access the info.
22. When info is private, only the owner shall access it.
23. When info is shared, the sharing criteria can be selected by the owner. (friends only, specific user(s) only)

Other

24. The system shall run on a cloud server.
25. The system shall run on a website environment.
26. The application shall have integration with selected social media platforms.
27. The application shall have a recommendation page that suggests user other info.
28. The user shall receive notifications if someone has commented, reacted on their content.
29. The system shall provide an advanced search function that enables user to search information based on various content attributes.

Design

UML Class Diagrams

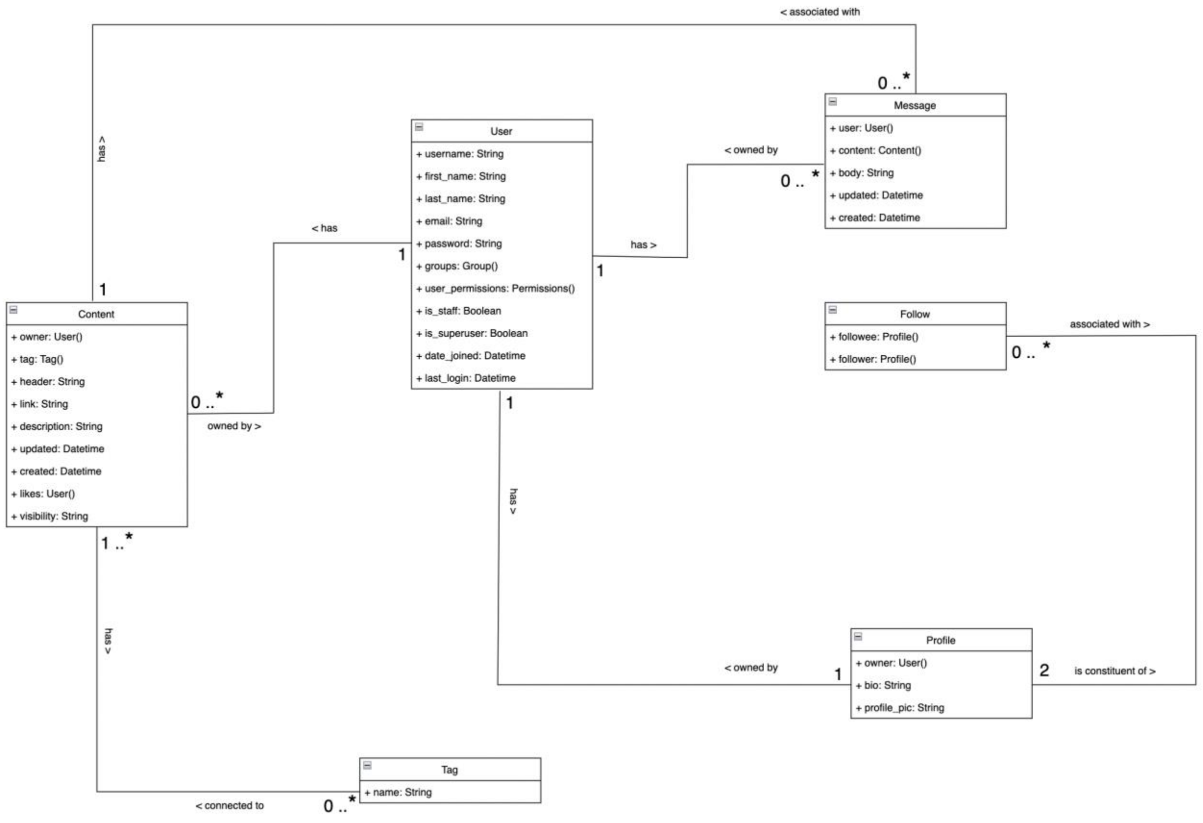


Figure 1. UML class diagrams for project models

Mock-up Screens

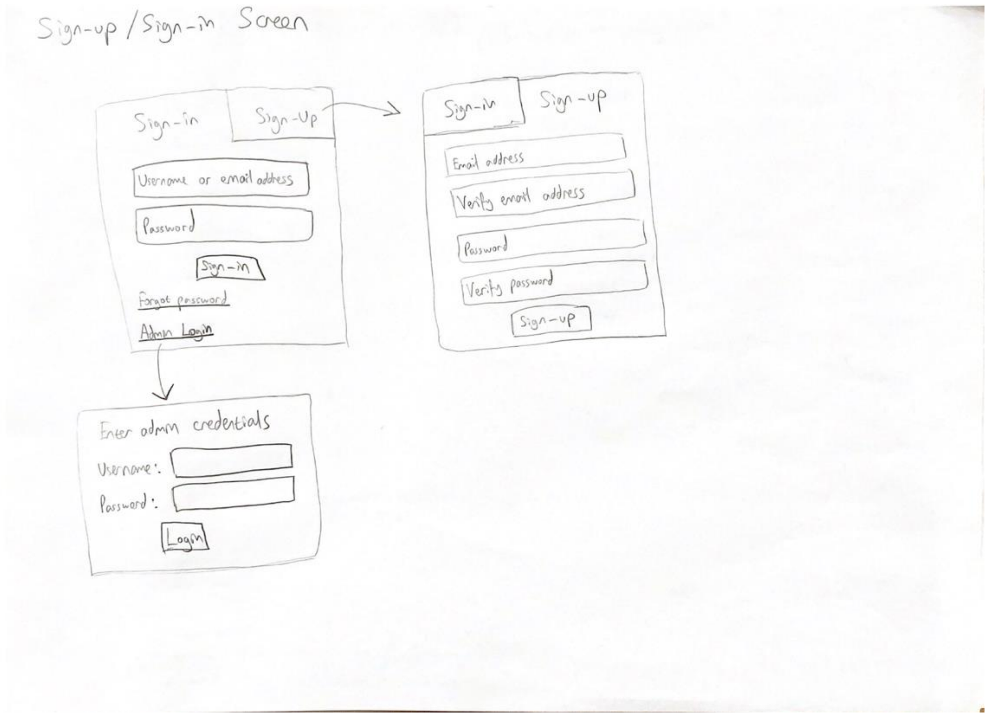


Figure 2. Mock up screen 1 - Sign up/sign in screen

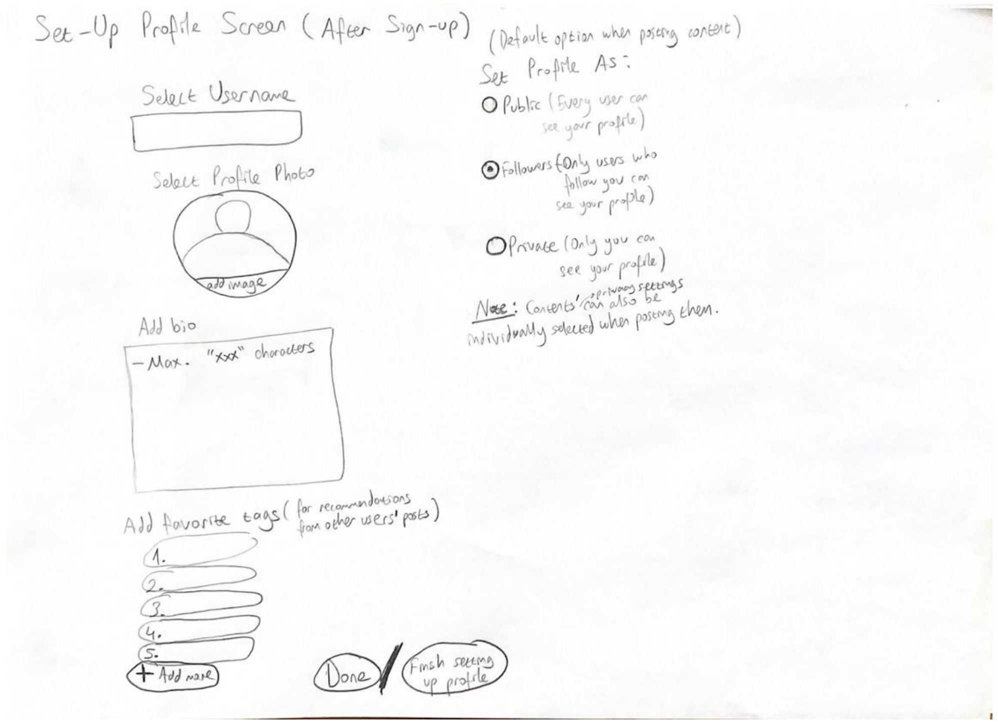

Set-Up Profile Screen (After Sign-up)
Select Username
Select Profile Photo
add image
Add bio
- Max. "xxx" characters
Add favorite tags (for recommendations from other users' posts)
1.
2.
3.
4.
5.
+ Add more
Done
Finish setting up profile
(Default option when posting content)
Set Profile As:
Public (Every user can see your profile)
Followers (Only users who follow you can see your profile)
Private (Only you can see your profile)
Note: Contents' privacy settings can also be individually selected when posting them.

Figure 3. Mock up screen 2 - Set up profile screen

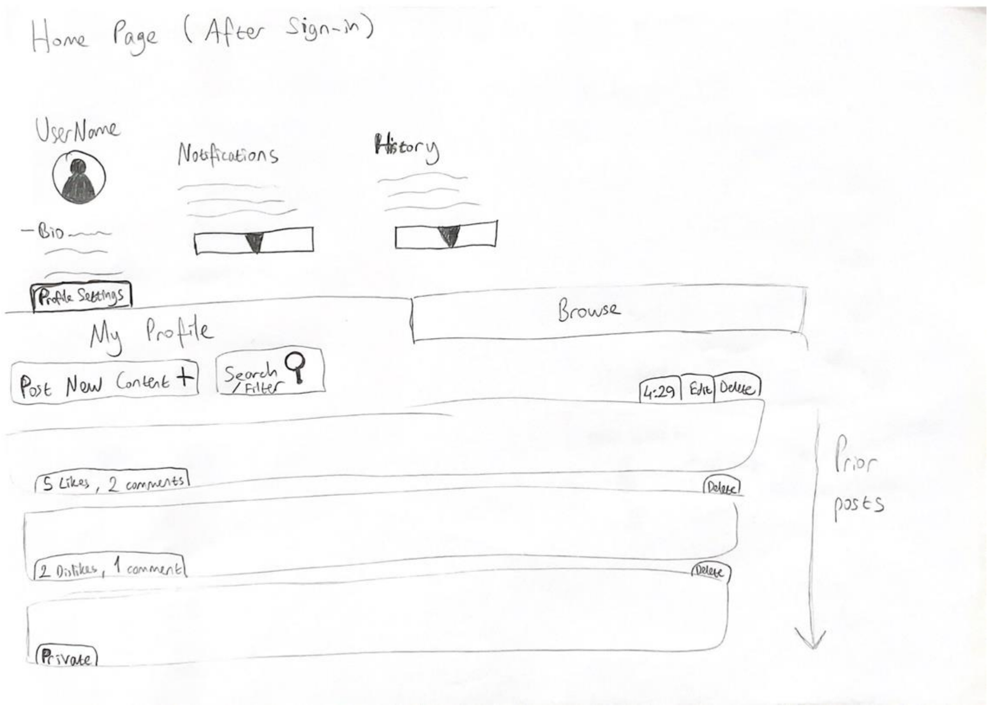


Figure 4. Mock up screen 3 - Home page screen

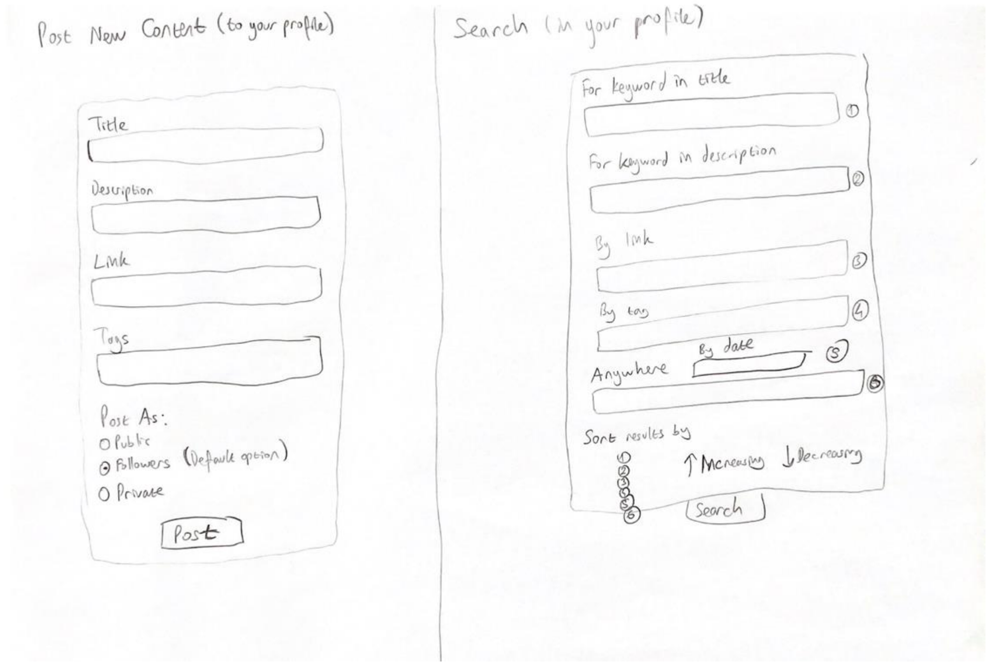


Figure 5. Mock up screen 4 - Content creation screen

Status of the Project

The status of each requirement

Creating a User Account

1. Implemented, documented, deployed, user tested – Not unit tested
2. Implemented, documented, deployed, user tested – Not unit tested
3. Implemented, documented, deployed, user tested – Not unit tested
4. Implemented, documented, deployed, user tested – Not unit tested
5. Not implemented
6. Implemented, documented, deployed, user tested – Not unit tested

Content Creation

7. Implemented, documented, deployed, user tested – Not unit tested
8. Implemented, documented, deployed, user tested – Not unit tested
9. Not implemented
10. Modification & Deletion functions → Implemented, documented, deployed, user tested – Not unit tested
    Duplication function → Not implemented
11. Requirement altered → Modification of content is not disallowed by a time limit but rather with the existence of interactions (likes or comments) → Implemented, documented, deployed, user tested – Not unit tested
12. Implemented, documented, deployed, user tested – Not unit tested
13. Implemented, documented, deployed, user tested – Not unit tested
14. Not implemented
15. Implemented, documented, deployed, user tested – Not unit tested

User Interactions

16. Not implemented
17. Implemented, documented, deployed, user tested – Not unit tested
18. Comment functionality → Implemented, documented, deployed, user tested – Not unit tested
    Tag functionality for other users' posts → Not implemented
19. Implemented, documented, deployed, user tested – Not unit tested
20. Not implemented
21. Implemented, documented, deployed, user tested – Not unit tested
22. Implemented, documented, deployed, user tested – Not unit tested
23. Visibility of shared contents by friends → Implemented, documented, deployed, user tested – Not unit tested
    Visibility of shared contents by criteria and for selected users → Not implemented

Other

24. Implemented, documented, deployed, user tested – Not unit tested
25. Implemented, documented, deployed, user tested – Not unit tested
26. Not implemented
27. Public contents can be seen by every user → Implemented, documented, deployed, user tested – Not unit tested

Specific recommendations for each user → Not implemented

28. Latest comments and content creations can be seen in the “Latest Activities” section of the website → Implemented, documented, deployed, user tested – Not unit tested
    Specific recommendations for the user → Not implemented
29. Basic search with a keyword → Implemented, documented, deployed, user tested – Not unit tested
    Advanced search based on various attributes → Not implemented

Status of Deployment

1) For the Milestone 2, I had carried out the following steps for deployment:
The project has been dockerized and the image has been pushed to AWS Elastic Container Registry. Link to the public AWS registry: AWS ECR - Lami Kaan Kosesoy public registry

Using the AWS Relational Database Service, an instance of Postgres SQL database has been created and configured to work with the project’s docker image. Project data that is stored on the local Postgres database has also been migrated to the AWS database on the cloud. Link to the AWS database: AWS RDS - Lami Kaan Kosesoy postgres database

Using the AWS Elastic Container Service, a service has been created where the docker image is connected to the database and run in a container on the cloud. The public URI for the deployed service is: Deployment Address for Milestone 2 - Primitive site

2) For the final deployment, I couldn’t make the above process work again. Always had issues with service and couldn’t view the site from the public IP address. Instead, another method has been carried out:

An AWS EC2 instance has been launched, an Ubuntu virtual server running on the cloud.

Instead of using the AWS RDS service, another container for a postgres database instance has been created and connected with Django application using docker compose.

The orchestrated containers are run on the EC2 instance and a public IP address has been assigned to it.

URI for the final deployment is: Final Deployment URI

System Manual

For the system to work, one should have python installed. Besides python, the following packages and the corresponding versions are required (as it can also be found in the “requirements.txt” file)
asgiref==3.5.2
Django==3.2.16

```
Pillow==8.2.0
psycopg2-binary==2.9.5
sqlparse==0.4.3
```

If docker is installed on their system, one can clone the repository from Github to their local and by running the “docker compose up” command in the project’s root directory (where dockerfile, docker-compose, manage.py etc. is located), the system shall be run.

User Manual

Description of how to use the system is provided below.

Step 1 – Sign Up/Registration:

- To use the system, one must be a registered user to the website.
- If the user is a first-timer on the website, then they should follow the sign up link which leads to the register page. By selecting an appropriate username and password, the user can register.
- Once the registration is complete, user can login with the selected credentials.

Step 2 – Viewing and Creating Contents:

- Once the user is logged in to the system, they are redirected to home page. On this page, the user can view public contents from all users, shared contents from friends and all the contents that belong to the user.
- By using the filters provided at the top of the contents section, user can filter contents to only see the public, shared or their own contents.
- By using the “Create new content” link below the filters, user can create a new content by providing input to the fields of the content creation form that shows upon clicking the link.

Step 3 – Liking and Commenting on Contents:

- By clicking to the header of the content, the user can go to the content page and comment on any content that is visible to them.
- The user can also like the contents that are visible to them, except if it’s their own content. Commenting is allowed but liking is prohibited for own contents.
- The user can revoke their likes by unliking, or delete their posted comments.

Step 4 – Updating and Deleting Contents

- Either on the home page or on the content page, the user may see links to updating or deleting the contents that belong to them.
- Deletion of contents is always possible.
- The contents can be updated as long as they have no interaction (likes or comments) from other users. If other users have already interacted with the content, then it can’t be modified but only deleted.

Step 5 – Profiles and Friend Relationships:
- On the left side of the home page user can see the preview of their profile. By clicking their username or using the link in the navbar, they can visit their profile.
- A user can also see another users profile by clicking on their username.
- In their own profiles, users can change their profile picture and bio using “edit profile” functionality.
- Users can follow and unfollow other users through profile pages.
- If two users follow each other, then they are considered as friends and can see each other’s shared contents in addition to public ones.

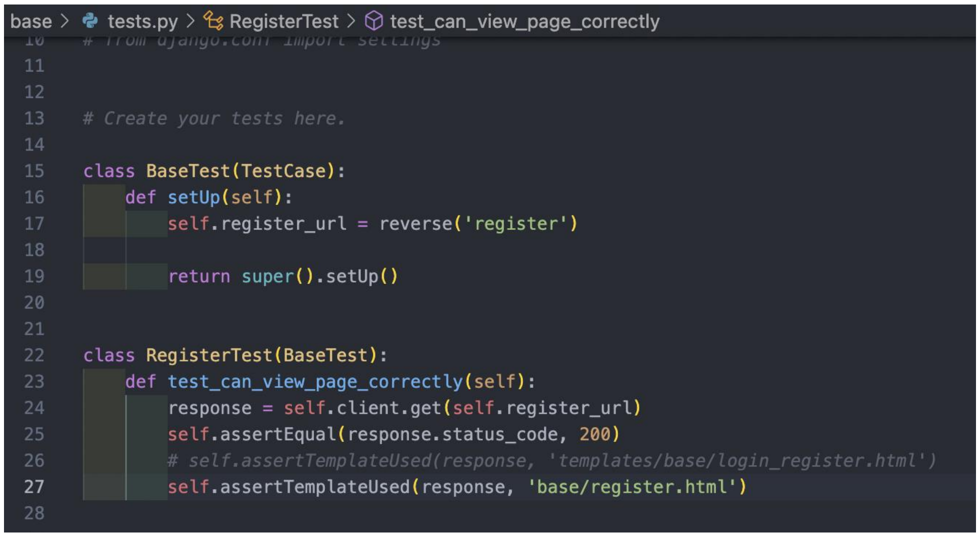
Step 6 – Search and Latest Activities
- In the right section of the home page, user can see up to 10 latest activities which is comprised of the comments or content creations that are visible to the user. By following the links, user can go to content or profile pages.
- In the navigation bar on any page, there exists a search box. It can be used to search for a keyword. Search function displays all the contents visible to the user and which include that keyword.

Step 7 – Logout
- By using the link to the right of the navigation bar, users can logout of the system.

A visual demonstration to the usage of the website can be seen in more detail by checking the presentation video link shared at the end of the report.

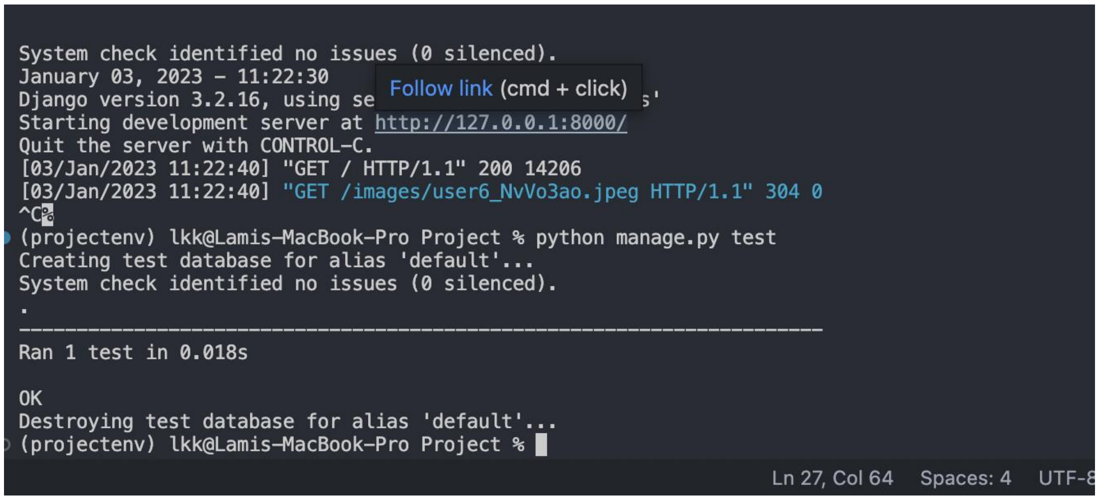
Test Results

Unit Tests
I have only 1 unit test which checks if the register page can be displayed properly or not. When “python manage.py test” is run on local, the test passes.


base > tests.py > RegisterTest > test_can_view_page_correctly

```
# from django.conf import settings


# Create your tests here.

class BaseTest(TestCase):
    def setUp(self):
        self.register_url = reverse('register')

        return super().setUp()


class RegisterTest(BaseTest):
    def test_can_view_page_correctly(self):
        response = self.client.get(self.register_url)
        self.assertEqual(response.status_code, 200)
        # self.assertTemplateUsed(response, 'templates/base/login_register.html')
        self.assertTemplateUsed(response, 'base/register.html')

```

Figure 6. Unit test for displaying the register page



```
System check identified no issues (0 silenced).
January 03, 2023 - 11:22:30
Django version 3.2.16, using se                          s'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[03/Jan/2023 11:22:40] "GET / HTTP/1.1" 200 14206
[03/Jan/2023 11:22:40] "GET /images/user6_NvVo3ao.jpeg HTTP/1.1" 304 0
^C
(projectenv) lkk@Lamis-MacBook-Pro Project % python manage.py test
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
.
----------------------------------------------------------------------
Ran 1 test in 0.018s

OK
Destroying test database for alias 'default'...
(projectenv) lkk@Lamis-MacBook-Pro Project %
```

Figure 7. Result of the test

User Tests
The functionalities of the website has been showed to be working during the demonstrations in the class. It can also be observed by following the link to the video demonstration which is provided below.

Link to Video Demonstration

https://youtu.be/11vzwgn5Gh0