Nilay Aydin

SWE 573 Software Development Practice Spring 2024

Californucation

May 20, 2024

Table of Contents

REPOSITORY AND DEPLOYMENT LINKS

Deployment url: http://16.170.162.125:3000

Backend Github url: https://github.com/niloaydin/SWE-573-HW

Frontend Github url: https://github.com/niloaydin/SWE-573-HW

Backend Tag Version: https://github.com/niloaydin/SWE-573-HW/releases/tag/v1.0.0

Frontend Tag Version: https://github.com/niloaydin/SWE-573-FE/releases/tag/v1.0.0

Video url: https://youtu.be/dFou9KRIAP8?si=S_5wro16p8SrJe7P

HONOR CODE

Related to the submission of all the project deliverables for the Swe573 2024 Spring semester project reported in this report, I Nilay Aydin declare that:

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

Nilay Aydin

OVERVIEW

Before going into project details, I would like to add some additional notes regarding my thought, design and implementation process. As I have stated in the milestone report, I chose to use Java, Spring Boot, and React for the project implementation. I will talk about the reasons behind those reasons, and the result of the implementation process. I wanted to learn Spring Boot for a very long time since it was on my todo list. Thanks to this class and project, I had to learn in a very detailed manner. My main focus was on the backend thinking frontend implementation would be easier. It's safe to say, this thought was completely wrong, I will list the reason for this statement. Starting backend development, my main goal was to use technologies that were not familiar to me; thus, Java, Spring boot, and Postgresql. I spent a lot of time understanding how Spring Boot works under the hood, what are the main characteristics, and what are the most optimized design choices. I think I spent too much time understanding the structure of the framework, but still I am leaving this class with an important tool in my toolbox which is learning how to learn something completely new, and create an application with those new technologies. I am content with what I learned through this semester and through this project, also I am very proud of myself.

At the very beginning, we created requirements according to the needs and scope of the application. Listing requirements was easy; nonetheless, completing them was not so much. I focused too much on the backend, I did not leave much time to complete the frontend thinking it would be easy. Creating the frontend was another challenge I had to face, main ones being learning Spring Boot and relational database structures. I did not maintain the backend development in parallel with frontend development. I finished the backend functionalities first, then implemented the frontend. This was a major mistake, because while implementing the frontend I faced lots of bugs and issues in the backend. Creating the backend solely does not give much of a perspective, I guess. I realized my mistakes during frontend development, and tried to solve the backend-related problems. This caused me to have a fully-functioning frontend. Even though I implemented nearly all of the backend features, there are some functionalities left to be implemented in the backend. So, it is safe to say I wanted to create a fully functioning backend with all the features I aimed at, and I created one at the cost of missing functionalities in the

frontend.

Another point in my implementation was the separation of frontend and backend repositories. There are two reasons for this decision: separation of concerns and not knowing how to handle frontend and backend in the same repository. I still kept using the backend repository as my main repository, creating frontend issues in the backend repository with the Frontend label. This decision provided me with easier navigation between repositories and easier deployment.

Towards the end of the deadline, I tried to make strategic decisions. I will mention in the later chapters; nevertheless, I would like to mention the selection of requirements to implement. As we had discussed in the classes, the main functionalities are chosen from all requirements.

At the beginning of the development, I did not make much use of issue creation. It was very overwhelming to keep up with the issues while dealing with the application creation process. Even though I had a roadmap, sticking to created issues was challenging. For example, I created an issue and started implementing what that issue requires. During the implementation, some unexpected errors or dependencies came up in order to complete the issue. It was very confusing how to handle those situations. Nevertheless, I realize that I was not very successful creating the issues, because if I were, those unexpected situations would be much easier to handle. At the end of the project, I got better at creating issues and sticking to them. At some point, I stopped thinking about the next step but followed the issues I created to show the roadmap. This is another tool I have in my toolbox from now on.

Additionally, I would like to highlight that, for authentication in the backend I code along to a youtube video created by Ali Bouali. It was the first feature I implemented in the backend, so I needed a guide.

SOFTWARE REQUIREMENTS SPECIFICATION AND STATUS

Below are the requirements elicited for the project based on class discussions and interview method.

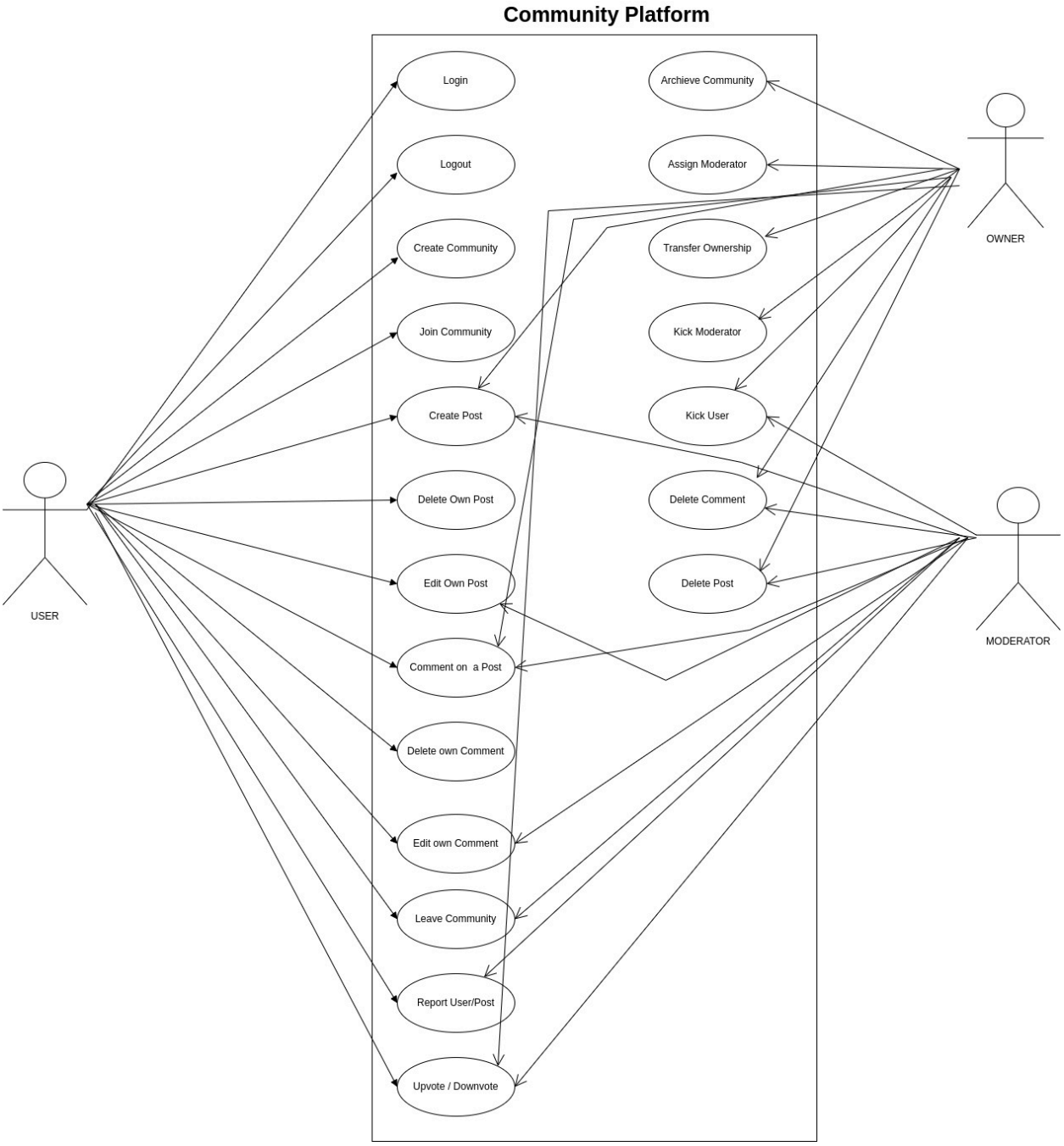
- **F1**: The system shall authenticate users with email and password. (Completed)
- **F2**: The system shall allow users to add an avatar and name to their profile. (Completed)
- **F3**: The system shall allow users to view their profile page. (Completed)
- **F4**: The system shall allow users to edit their profile information. (Completed)
- **F5**: The system shall support user roles such as Creator (Owner), Moderator, and User. (Partially Completed: There is no moderator)
- **F6**: The system shall assign the role of Owner to users when they create a community. (Completed)
- **F7**: The system shall designate the creator of the community as the Owner. (Completed)
- **F8**: The system shall provide options for specifying media types for posts during community creation, including image, geolocation, text, and date.(Partially Completed: There is no geolocation)
- **F9**: The system shall allow only the Creator to set the title, description, and rules for the community page. (Completed)
- **F10**: The system shall allow only the Creator to appoint or remove moderators. (Not Completed)
- **F11**: The system shall allow users to post content. (Completed)
- **F12**: The system shall allow users to comment on posts. (Not Completed)
- **F13**: The system shall allow users to edit their own posts. (Completed)
- **F14**: The system shall allow users to edit their own comments. (Not Completed)
- **F15**: The system shall allow users to delete their own posts. (Completed)
- **F16**: The system shall allow users to delete their own comments. (Not Completed)
- **F17**: The system shall allow users to upvote or downvote posts and comments. (Not Completed)

- **F18**: The system shall allow users to report other users or posts to moderators. (Not Completed)
- **F19**: The system shall allow moderators to remove users from the community. (Not Completed)
- **F20**: The system shall not allow users who have been removed from a community to rejoin the same community.(Not Completed)
- **F21**: If the community is public, the system shall allow users to join without approval from moderators or the Owner.(Completed)
- **F23**: If the community is private, the system shall allow users to join by invitation only.(Not Completed)
- **F24**: The system shall provide a mechanism for generating invitation secrets for private communities.(Not Completed)
- **F25**: The system shall support multiple moderators in a community but only one Owner.(Not Completed)
- **F26**: The system shall not allow moderators to remove other moderators; only the Owner can do this.(Not Completed)
- **F27**: The system shall allow the Owner to relinquish their role without transferring it to another user.(Not Completed)
- **F28**: The system shall not allow users to edit posts or comments made by others.(Completed)
- **F29**: The system shall allow moderators and Owners to delete posts or comments made by users.(Partially Completed: There is no comment feature)
- **F30**: The system shall allow users to leave a community at any time.(Completed)
- **F31**: The system shall allow users to delete their accounts.(Not Completed)
- **F32**: The system shall provide advanced search for posts within a community.(Completed)
- **F33**: The system shall not allow the Creator to delete a community, but shall allow them to archive it.(Not Completed)
- **F34**: The system shall not impose a character limit on posts.(Completed)
- **F35**: The system shall allow Owners to create different templates for various types of posts.(Completed)

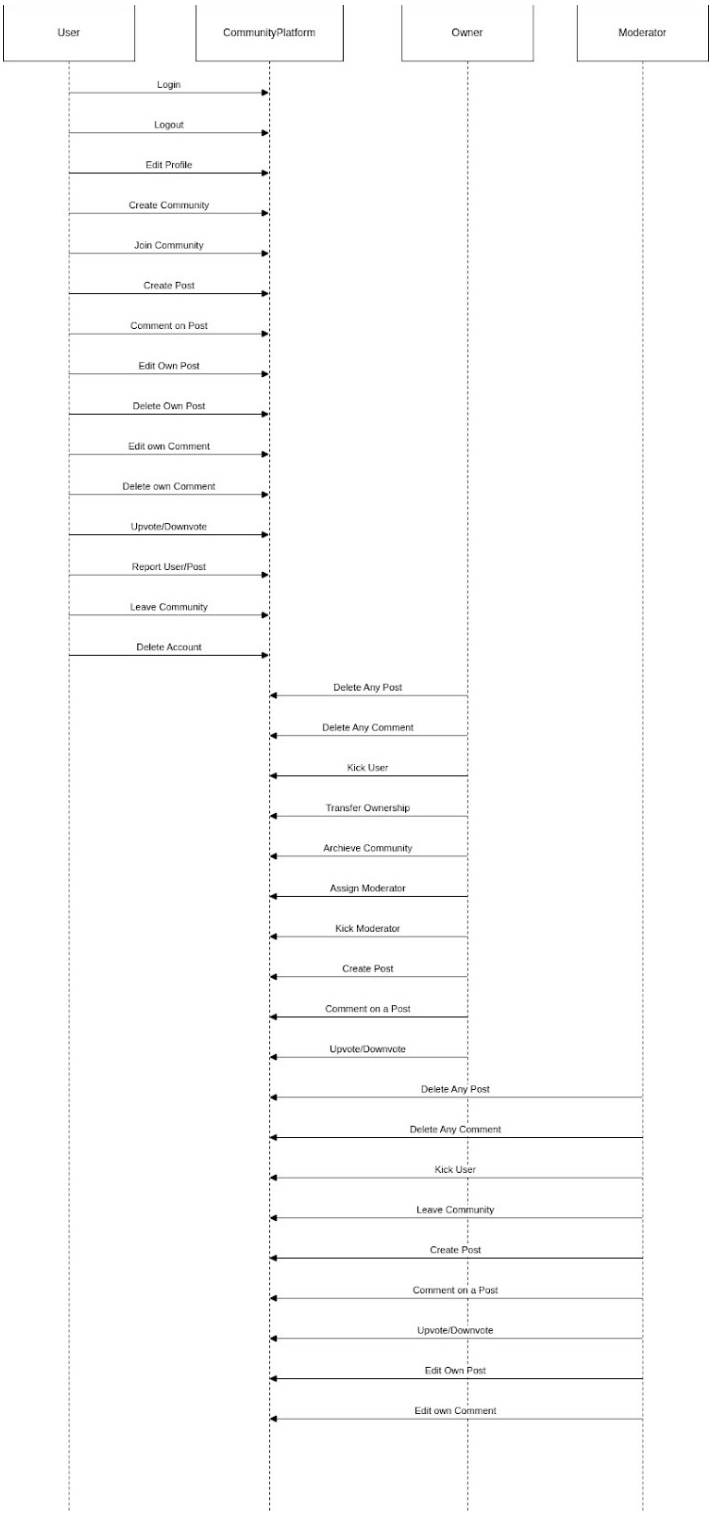
DESIGN DOCUMENTS

With the intent of creating a roadmap, design documents are created. Design documents include Use Case, Sequence, Class, and ER diagrams.

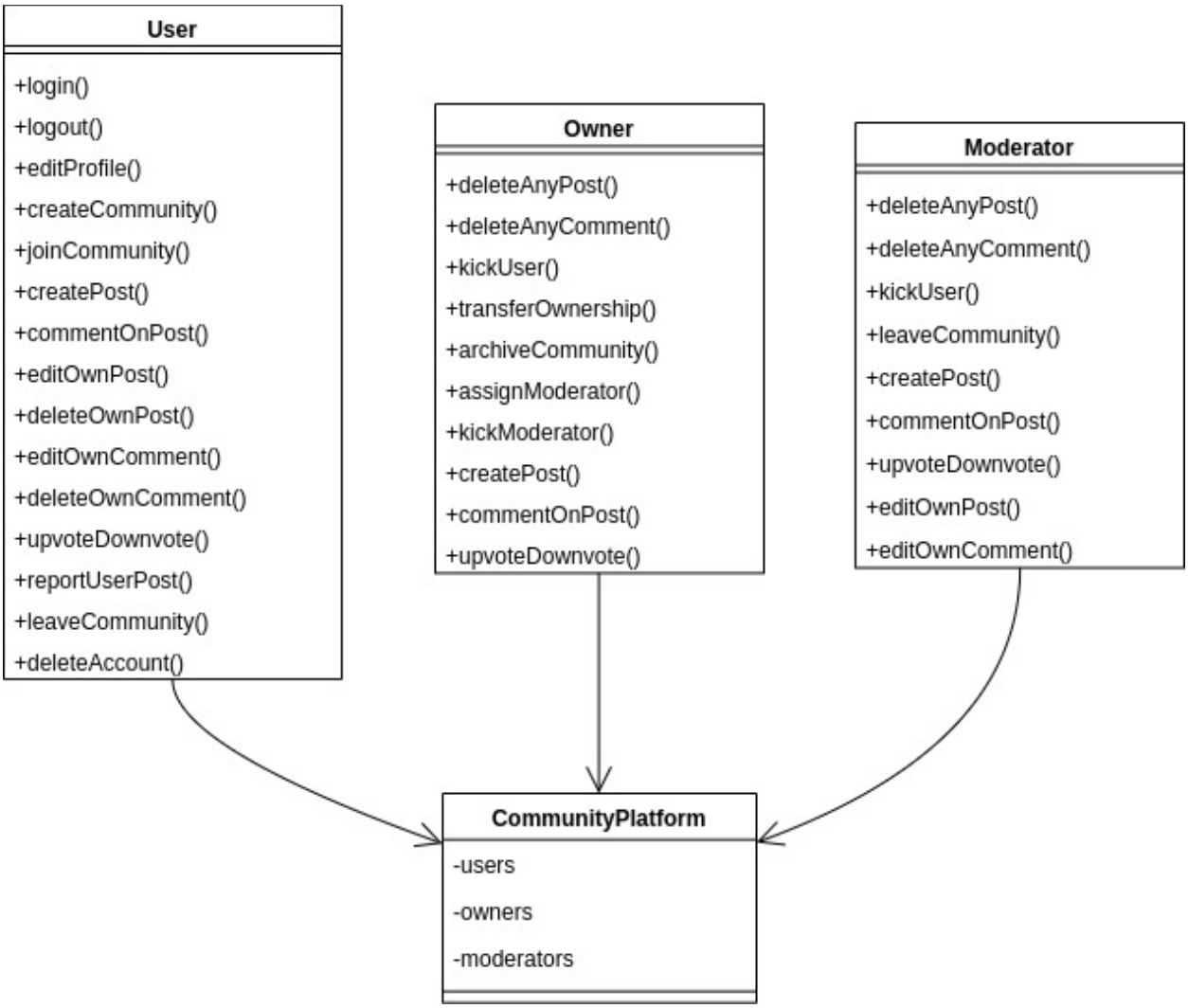
Use Case Diagram



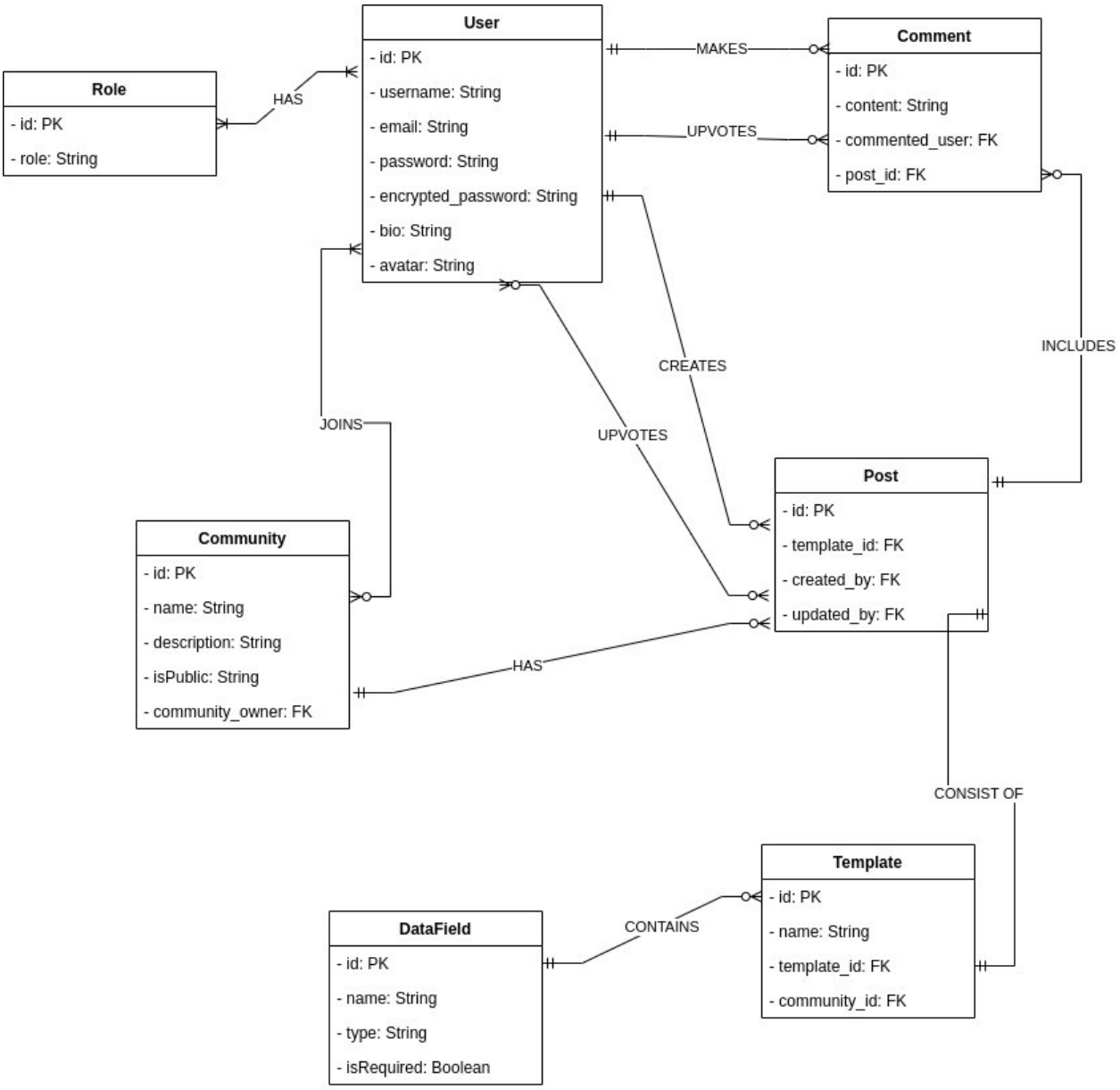
Sequence Diagram



Class Diagram


User
+login()
+logout()
+editProfile()
+createCommunity()
+joinCommunity()
+createPost()
+commentOnPost()
+editOwnPost()
+deleteOwnPost()
+editOwnComment()
+deleteOwnComment()
+upvoteDownvote()
+reportUserPost()
+leaveCommunity()
+deleteAccount()
Owner
+deleteAnyPost()
+deleteAnyComment()
+kickUser()
+transferOwnership()
+archiveCommunity()
+assignModerator()
+kickModerator()
+createPost()
+commentOnPost()
+upvoteDownvote()
Moderator
+deleteAnyPost()
+deleteAnyComment()
+kickUser()
+leaveCommunity()
+createPost()
+commentOnPost()
+upvoteDownvote()
+editOwnPost()
+editOwnComment()
CommunityPlatform
-users
-owners
-moderators

ER Diagram



DEPLOYMENT

The project is divided into two repositories: Backend and Frontend. Both repositories have their own Docker file. Backend docker file also includes the database. I used AWS for deployment.

Project Url = http://16.170.162.125:3000

PROJECT INSTALLATION

Requirements for Backend:

Docker

IntelliJ or Eclipse

Postgresql server

Backend Installation:

1. Clone the repository from this github link.
2. Open the project with IntelliJ or Eclipse.

If you want to run the project locally:

1. Change the database connection properties in properties.yml file.
2. Run the project through ide
3. If the project runs without error, you can navigate to `http://localhost:8080`

If you want to run the project with Docker:

1. Open the ide terminal
2. Run `docker compose up -d`
3. In order to double check, run `docker ps`
4. If you see your containers running, you can navigate to `http://localhost:8080`

Requirements for Frontend:

Docker

Vscode or any other IDE

Node Js

Npm

Frontend Installation:

1. Clone the repository from this github link.
2. Open the project with VScode or the IDE of your choice

If you want to run the project locally:

1. Open the terminal
2. Run `npm install` to download dependencies

3. Change `base_url.js` file for the backend connection url
4. Run `npm start`
5. One the application is running without an error, you can navigate to `http://localhost:3000`

If you want to run the project with Docker:

5. Open the ide terminal
6. Run `docker build -t community-application-frontend .` to create the image
7. In order to double check, run `docker images`
8. If you see your image there, run `docker run -p 3000:3000 community-application-frontend`
9. In order to double check, run `docker ps`
10. If you see your container running, you can navigate to `http://localhost:3000`

USER MANUAL

Authentication

1. Upon accessing the application via the URL, users not logged in are redirected to the login page.
2. Existing users can log in using their credentials.
3. New users have the option to register and provide their information.
4. After successful registration, the application automatically redirects the user to the login page for authentication.
5. If the user's credentials are correct, the application navigates the user to the home page.
6. If the user's credentials are not correct, the application shows an error.

Homepage

1. On the homepage, users can view a list of all communities and recently published posts.
2. Users can click on the "Visit" button next to communities to navigate to the community detail page.

3. Similarly, users can click on "See details" on posts to be directed to the post details page.
4. Additionally, users have the option to perform a basic search for communities.

Header

1. The header includes tabs for "Communities", "Home", "Profile", "Create Community", and "Logout".
2. Clicking on the "Communities" tab directs users to the community list page where they can view all communities.
3. Clicking on the "Home" tab directs users to the homepage.
4. Clicking on the "Profile" tab directs users to their profile page where they can view their information.
5. Clicking on the "Create Community" tab directs users to the create community page where they can create a new community.
6. Clicking on the "Logout" tab logs users out from the application and directs them to the login page.

Create Community Page

1. When users click on the "Create Community" tab, they will be directed to the create community page.
2. On the create community page, users can fill in the community name, description, and specify whether it's private or not.
3. After filling in the necessary information, users can click on the "Create" button.
4. Upon clicking the "Create" button, users will be directed to the newly created community details page.

Community Detail Page

1. The community detail page offers various functionalities:

a. Users can view the posts created within the community.
b. They can see the list of members belonging to the community.
c. Information about the owner of the community is displayed.
d. Users have the ability to create posts within the community.

e. They can join or leave the community based on their preference.

f. Post templates specific to the community are also visible.

g. They can conduct template based specific post searches.

Functionalities Reserved for Owners:

2. They can create a template specific to that community.

3. Clicking on "Create Template" allows the owner to access the template creation page.

4. On the template creation page, the owner can specify:

a. Template name.

b. Dynamic data fields, with the option to add as many as desired.

c. For each data field, they can choose from the following data types: text, number, date, and URL.

5. After creating the template, the owner has the ability to delete the template at a later time if desired.

Create Post Page

1. On the community detail page, users click on "Create Post".
2. In the create post page, users first choose a template for the post.
3. After selecting the template, users see the fields associated with the chosen template.
4. Users fill in the required fields.
5. Upon clicking "Submit", if the post field validation fails (i.e., if users haven't filled in the required fields or if their input values do not match the template's input values), an error is displayed, and the post is not created.
6.
7. If the validation passes, the post is successfully created, and users are redirected back to the community detail page.

Post Detail Page

1. When users click on "See Details" for a post, they are directed to the post detail page.

2. On the post detail page, users can view the fields and their corresponding values for the post.
3. If users wish to edit the post, they can click on the "Edit Post" button.
4. Clicking on "Edit Post" reveals the input fields where users can modify the values of the fields.
5. After completing the editing process, if the validation passes (i.e., all required fields are filled), the post is successfully edited.
6. Upon successful editing, users are redirected back to the post detail page to view the updated post information.
7. If the user is not the creator of the post, trying to edit the post will not be authorized.

Edit Community Functionality

1. If a user is the owner of the community, they can access the "Edit Community" and "Create Template" buttons.
2. Clicking on "Edit Community" allows the owner to view and modify the community information.
3. After editing the community details, the owner can submit the changes.
4. If the validation passes (e.g., ensuring the community name is not empty), the community is successfully updated.
5. Upon successful editing, the owner is redirected to the community detail page to view the updated community information.

Profile Page

1. When users click on the profile page, they are directed to their own profile.
2. On the profile page, users can view the following information:

a. Name
b. Last name
c. Username
d. Email
e. Avatar
f. Communities they have joined

g. Communities they own

Edit Profile Functionality

1. When a user clicks on the "Edit" button on their profile page, they are presented with their current information.
2. Users can modify the inputs they wish to change.
3. Upon clicking "Submit", if there are no errors, the user's profile information is updated.
4. After successful submission, the user is redirected to their profile page to view the updated information.

USER TESTS

Login Request

URL: **http://13.60.88.103:3000/login**
Correct Input:

- Email: nilo@gmail.com
- Password: nilo

Output:

- User is successfully logged in.
- Token is saved to local storage.
- Redirected to the homepage.

Wrong Input:

- Email: dont_have_this_user@gmail.com
- Password: dont_have_this_user

Output:

- Error: Bad credentials.

Create Community

URL: http://13.60.88.103:3000/create-community

Correct Input:

- Community Name: User Test Community
- Description: This is a community for user testing.
- Is Public: true

Output:

- Community successfully created.
- User is redirected to the community detail page of the newly created community.

Wrong Input:

- Community Name: null
- Description: Description doesn't matter
- Is Public: true

Output:

- Error: Validation error. Community isn't created.

Create Post in a Community
URL: http://13.60.88.103:3000/create-post
Correct Input:

- Template: Template A
    - Name: Nilay
    - Age: 26

Output:

- Post is successfully created.
- User is redirected to the community detail page.

Wrong Input:

- Template: Template A
    - Name: null
    - Age: yirmialti

Output:

- Error: Validation error. Post isn't created.

Update Post

URL: http://13.60.88.103:3000/community/{communityId}/post-details/{postId}

Post Template: Template A

Owner Status: You are the owner of the post

Correct Input:

- Name: Neyzen

- Age: 32

Output:

- Post is successfully updated.
- User sees the updated post.

Wrong Input:

- Owner Status: You are NOT the owner of the post

Correct Input:

- Name: Neyzen
- Age: 32

Output:

- Error: Authorization error. Post isn't updated.

Advanced Search in Community

URL: http://13.60.88.103:3000/community/{communityId}

Post Template: Post Template A

Correct Input 1:

- Name: Neyzen

Output:

- Posts created with Post Template A and containing "Neyzen" in their name are displayed.

Correct Input 2:

- Name: Neyzen
- Age: 32

Output:

- Posts created with Post Template A and containing "Neyzen" in their name and "32" in their age are displayed.

Note To Instructor

This version of the project might contain some bugs. The most common one; if the token in the localstorage is not correct, there will be a lot of alerts in the homepage since it sends a lot of requests and shows each alert. If that's the case, please clear cache & localstorage, and try refreshing the page. It will redirect you to the login page. Even though this version might not be the stable one, I had an amazon learning experience through developing the application. I will continue to enhance the project even further as a self-project. I believe I have achieved my goal that I set at the beginning of this course. Thank you!