

SWE573 Software Development Practice Final Report

Project Name: “Communitism” a Social Community Application.

AUTHOR

FERİDUN BERK AKYOL 2022719192

DATE 19.05.2024

DEPLOYMENT [URL](#)

GITHUB REPOSITORY [URL](#)

GITHUB TAG VERSION [URL](#)

HONOR CODE

Related to the submission of all the project deliverables for the Swe573 2024 Spring semester project reported in this report, I **Feridun Berk Akyol** 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.

Feridun Berk Akyol
Signature

Table of Contents

1. Project Overview2
 Tech Stack2

2. Software Requirements Specification.....3
 Functional Requirements.....3
 Non-Functional Requirements.....4

3. Design Documents5
 ER Diagram5
 UML Diagrams.....6
 Sequence Diagrams10
 Use Case Diagrams.....12

4. Status of the project13

5. Status of the Deployment18

6. User Manual.....19

7. Test Results25

Project Overview

In SWE573 course our aim was to develop a community application from scratch. We started our design with requirements elicitation, requirements are based on our discussions during class hours.

Our application is about creating communities for users to interact with each other based on their interests. It's similar to platforms like Reddit, 4Chan and Hacker News. Users can create their own communities, join the available communities, and create posts within these communities. One of the most important features of our application is the post templates every community has a default post template and custom post templates can also be created.

Tech Stack

Backend

- JAVA 21*
- Spring Boot 3

Frontend

- JavaScript
- HTML, CSS, Bootstrap, Thymeleaf

Database

- MySQL 8.0.0

Deployment

- AWS as a CSP (Cloud Service Provider)
- Docker Desktop (for local containerization)

Version Control

- Git and GitHub

We built our application following the MVC(Model-View-Controller) pattern and utilizing DTOs. For the frontend we used the SSR(Server Side Rendering) where we render the html in backend send the rendered version to client.

We used the template engine "Thymeleaf" alongside Javascript and bootstrap.

Testing

You can use the given URL in the title page to use our application you can either create your own user or use the test user. Username: test, Password: test.

owners or admins. We will dive into details how our application works in User guide.

Software Requirements Specification

Function Requirements

User Management:

1. Users can register with unique usernames and passwords.
2. Users can edit their profiles (including profile pictures, avatars, bio, etc.).
3. Users have profile pages displaying their credentials and subscribed communities.
4. Users can view other user profiles.

Search Functions:

5. Users can search for communities, posts, and comments.
6. System provides recommendations based on user search results.
7. Users can list all communities.

User Interactions:

8. Users can create posts and comments.
9. Users can create communities.
10. Users can comment to posts.
11. Users can delete their posts.
12. Users can delete and edit their profiles.
13. Users can view and subscribe/unsubscribe to communities.
14. Users can create posts with custom templates provided by communities.

Moderation:

15. Communities have moderators can take action against offensive content and ban users.
16. Owners can delete communities and posts.
17. Communities can be private or public.
18. Community Specific Post Templates:
19. Communities have owners.
20. Owners are assigned during community creation.
21. Posts are listed under community pages.
22. Subscribers are listed under community pages.
23. Community owners can create custom post templates.

Non-Functional Requirements:

Security:

1. Sensitive data such as passwords are encrypted.
2. Secure authentication mechanisms prevent unauthorized access.

Performance and Reliability:

3. Ensure minimal downtime and data corruption.
4. Scalability to handle increasing user and content load.

Usability:

5. User-friendly interface for easy navigation and interaction.
6. Accessibility options for users with disabilities.

Design Documents

ER Diagram

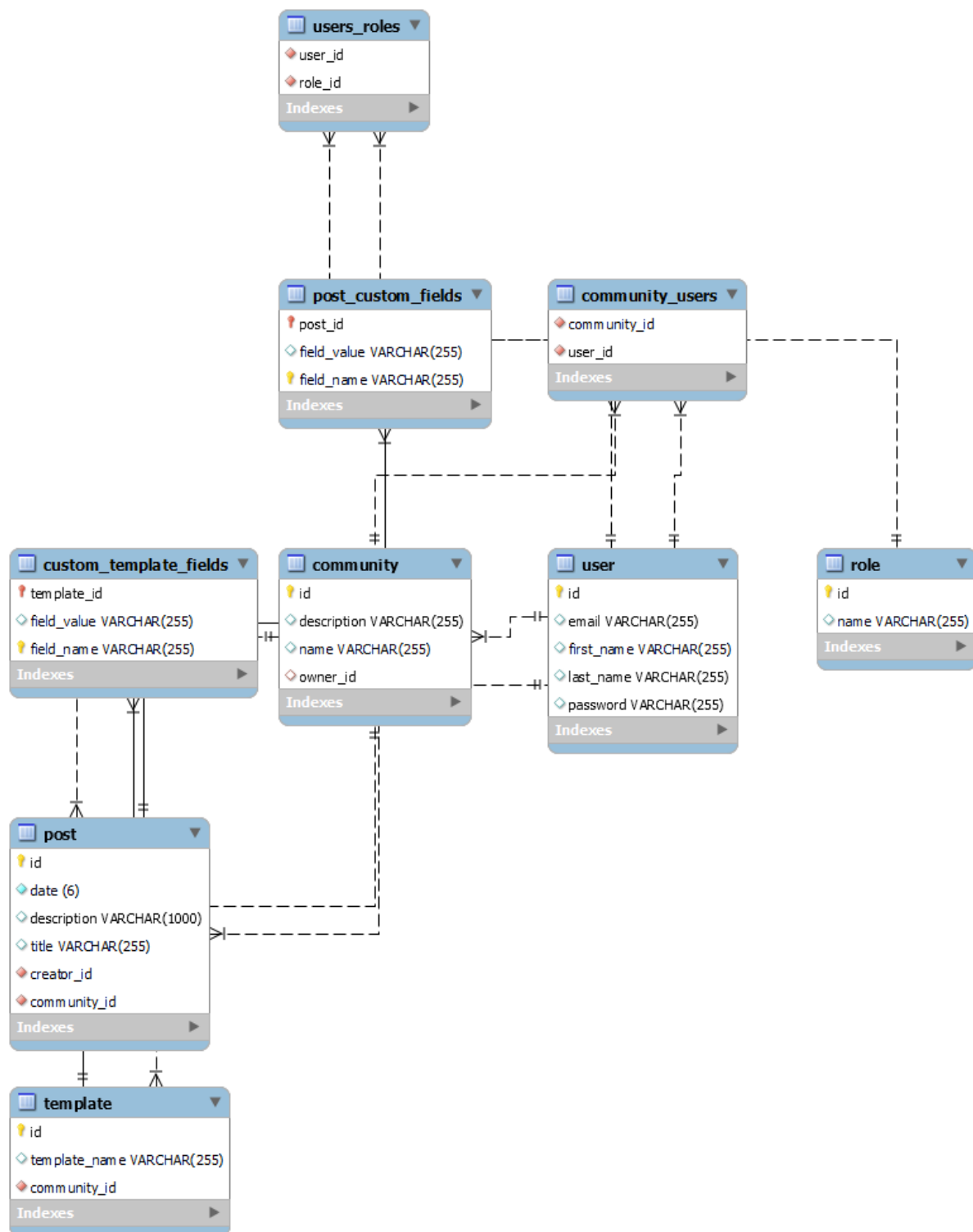


Figure 1. Er diagram of the database design in crow's foot notation.

We can see the Database design in Figure 1, The relations between entities are given in crow's foot notation.

Relationships Between Entities.

Some of the relationships between main entities are as follows.

- Post class has a one-to-many relationship with the community meaning one community can have multiple posts, but a post can only belong to a single community.
- Community class has many to many relationships with User meaning one user can subscribe to multiple communities and communities can have multiple subscribers.
- Community class has one to many relationships with community owner meaning a community owner can own multiple communities, but a community can only have one owner.
- Template has one to many relationships with the post class meaning a post can only have one template, but one template can be used for multiple posts.

All the relationships are represented in the ER diagram.

UML Diagram

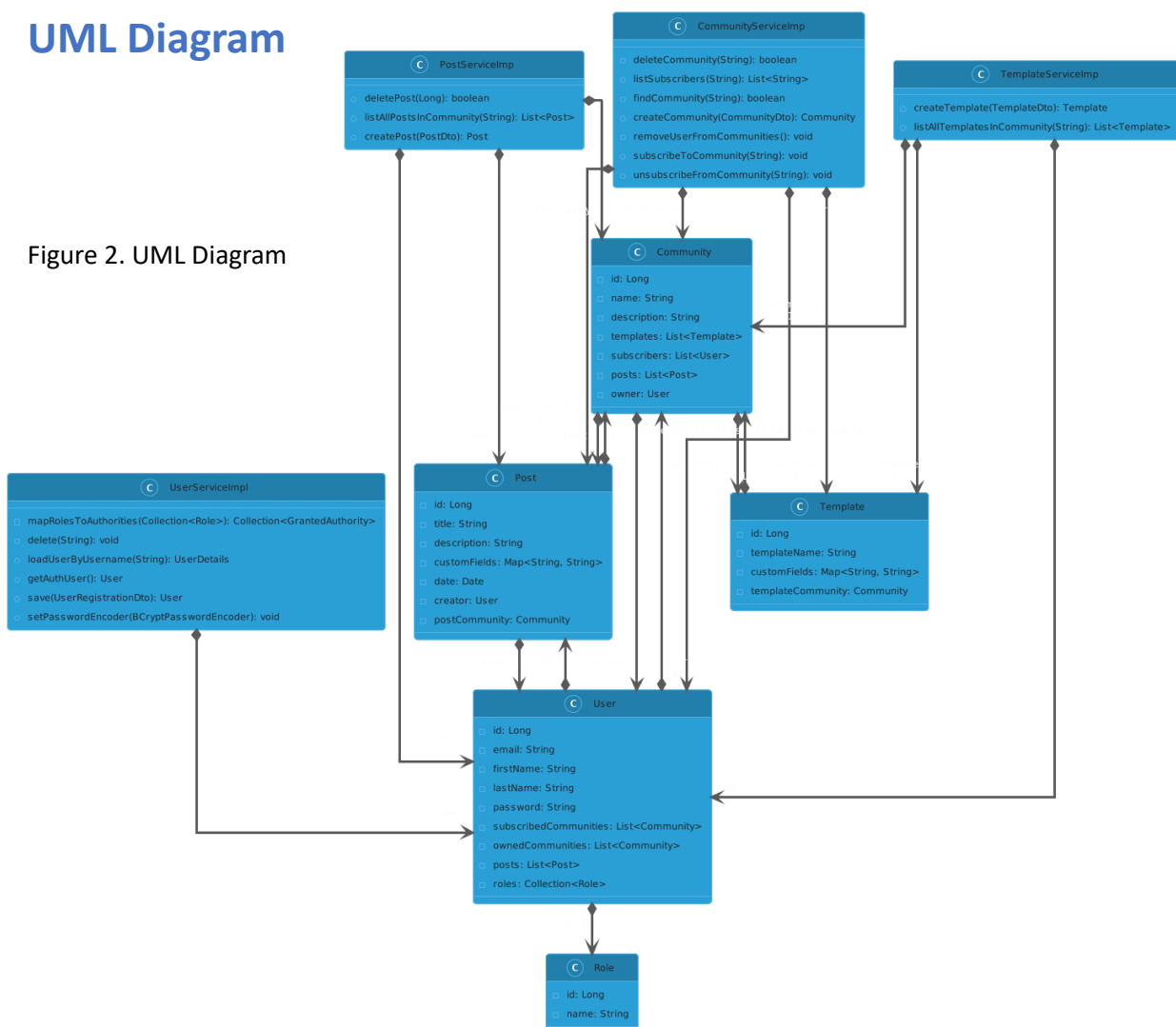


Figure 2. UML Diagram

The latest version of our UML diagram is given in the Figure 2. It's been changed and modified since the initial design of our Project as part of spiral design approach the first version is still available in the repository [wiki page](#).

Explanation of entities of our system.

Community

The Community entity represents a community within the system. It has several attributes: id (a unique identifier for the community), name (the name of the community), description (a brief description of the community), templates (a list of templates associated with the community), subscribers (a list of users subscribed to the community), posts (a list of posts made in the community), and owner (the user who owns the community). A community can host various templates that structure the posts within it and can manage multiple subscribers who can contribute posts and participate in community activities.

Post

The Post entity represents a post within a community. Its attributes include id (a unique identifier for the post), title (the title of the post), description (the content of the post), customFields (a map for additional data if custom Template is used), date (the creation date of the post), creator (the user who created the post), and postCommunity (the community to which the post belongs). A post serves as the primary means of content creation and interaction within a community, allowing users to share information and engage with each other.

Template

The Template entity defines a template for creating posts within a community. It has attributes such as id (a unique identifier for the template), templateName (the name of the template), customFields (a map for additional metadata fields), and templateCommunity (the community to which the template belongs). Templates provide a standardized structure for posts, ensuring consistency and facilitating the creation of posts with predefined fields and formats.

User

The User entity represents a user within the system. It includes attributes such as id (a unique identifier for the user), email (the user's email address),

firstName (the user's first name), lastName (the user's last name), password (the user's password), subscribedCommunities (a list of communities the user is subscribed to), ownedCommunities (a list of communities the user owns), posts (a list of posts created by the user), and roles (a collection of roles assigned to the user). Users interact with communities by subscribing to them, creating posts, and managing their own communities. The roles attribute defines the user's permissions and access levels within the system.

Role

The Role entity defines a role that can be assigned to a user. It includes attributes such as id (a unique identifier for the role) and name (the name of the role). Roles are used to manage permissions and access control within the system, allowing users to have different levels of access and functionality based on their assigned roles.

Service Classes

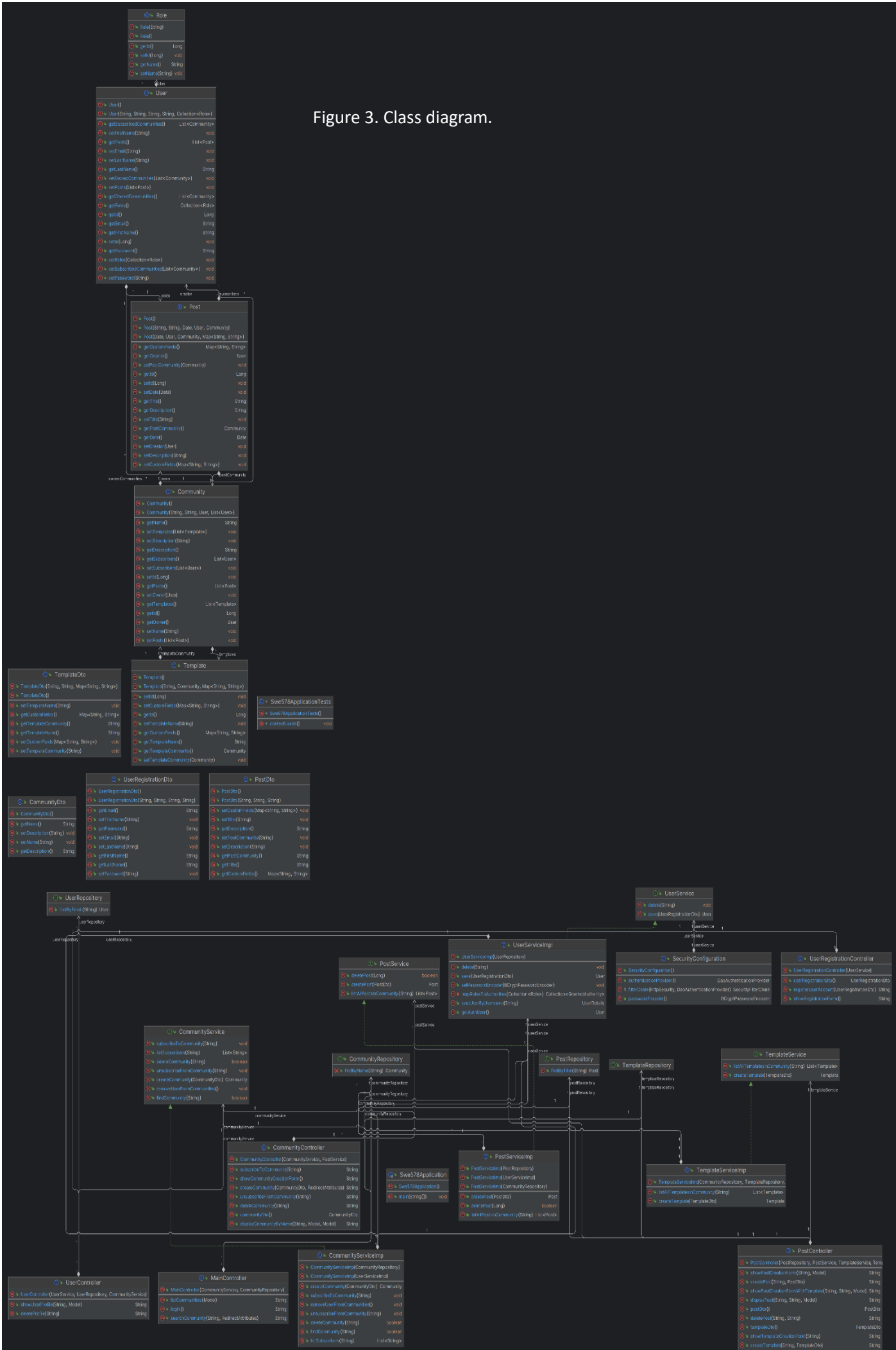
Service Classes are implementing the logic for managing classes. The service layer facilitates operations related to classes such as community, user, post and template, ensuring that the business logic is separate from the data access layer.

- CommunityService interacts with Community, User, Post, and Template.
- PostService interacts with Post, Community, and User.
- TemplateService interacts with Template, Community, and User.
- UserService interacts with User.

Every service is independent and has single responsibility although we have similarities with microservice architecture we can not say we fully utilized the microservice architecture model in our application but used it as a guide.

DTO Classes

We also used the DTO (Data Transfer Objects) in our application to transfer data between the layers. Mainly between controllers and services. All the classes including Database layer and DTOs can be seen in the Figure 3.



Sequence Diagrams

Updated versions of our sequence diagrams are given below, old versions are available and can be seen in the repository [wiki page](#).

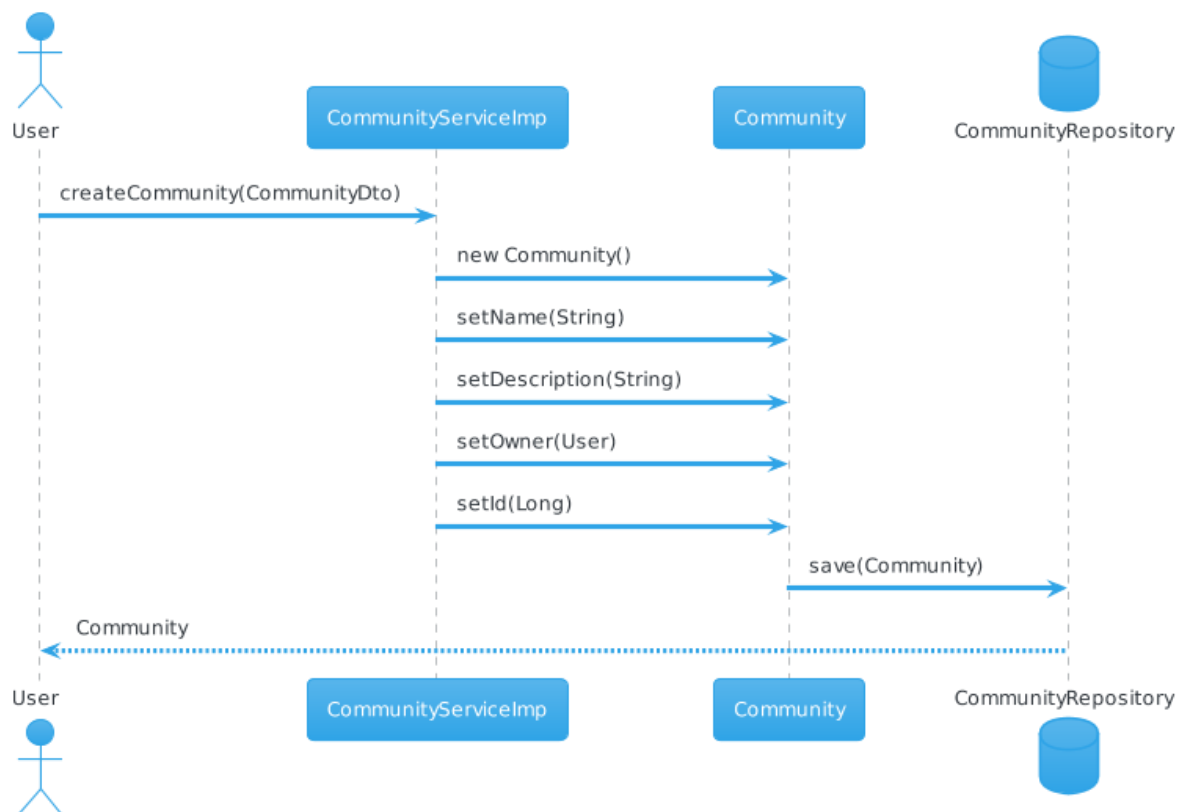


Figure 4. Sequence Diagram for Community Creation.

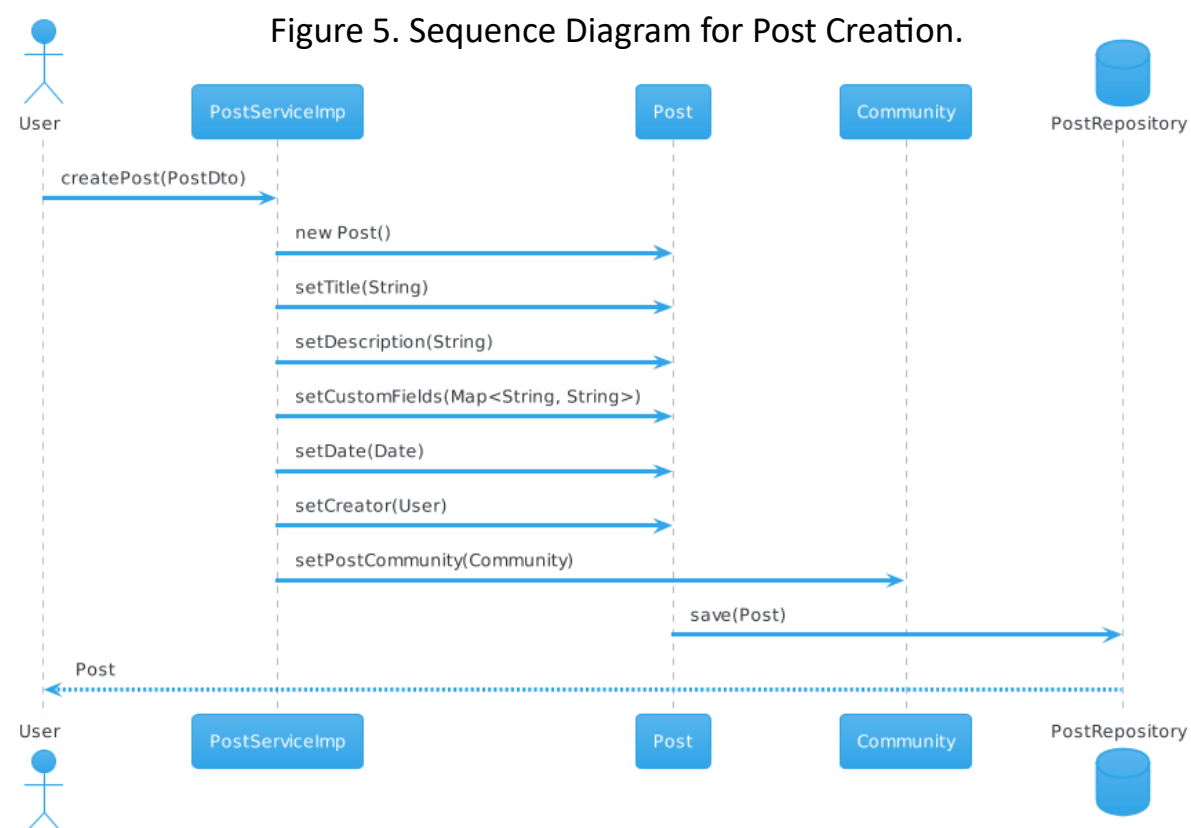


Figure 5. Sequence Diagram for Post Creation.

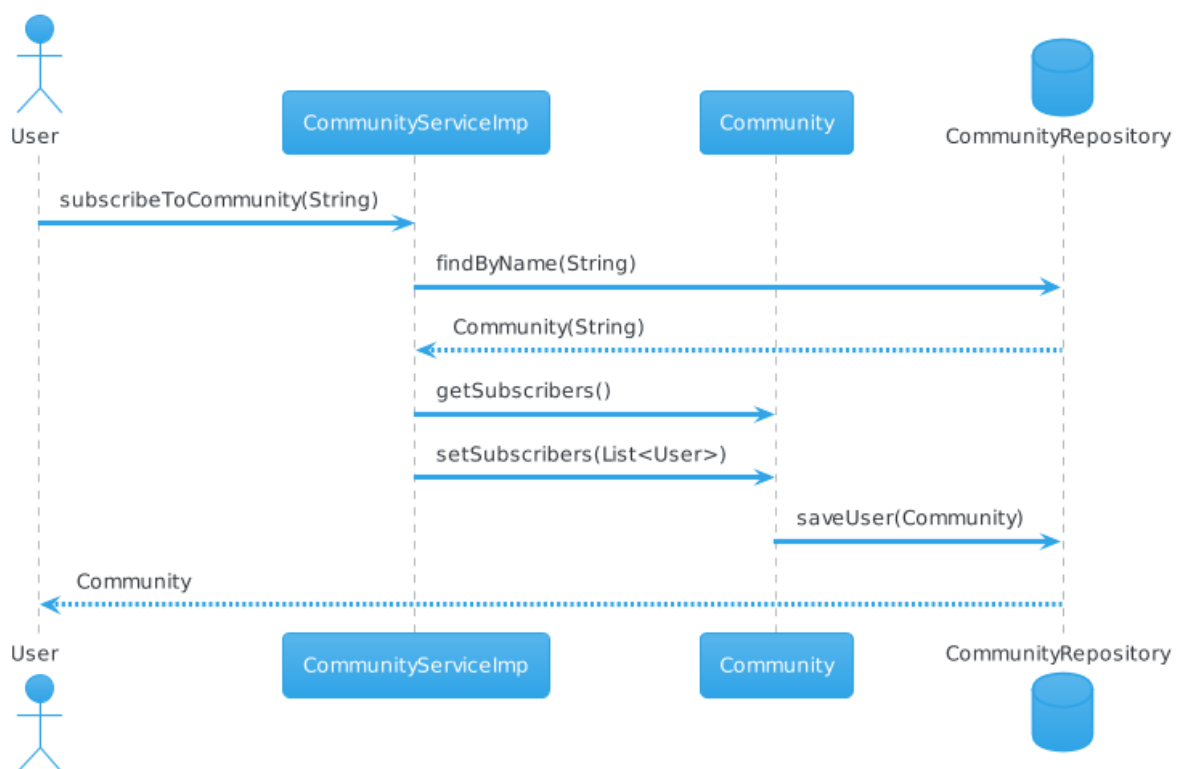
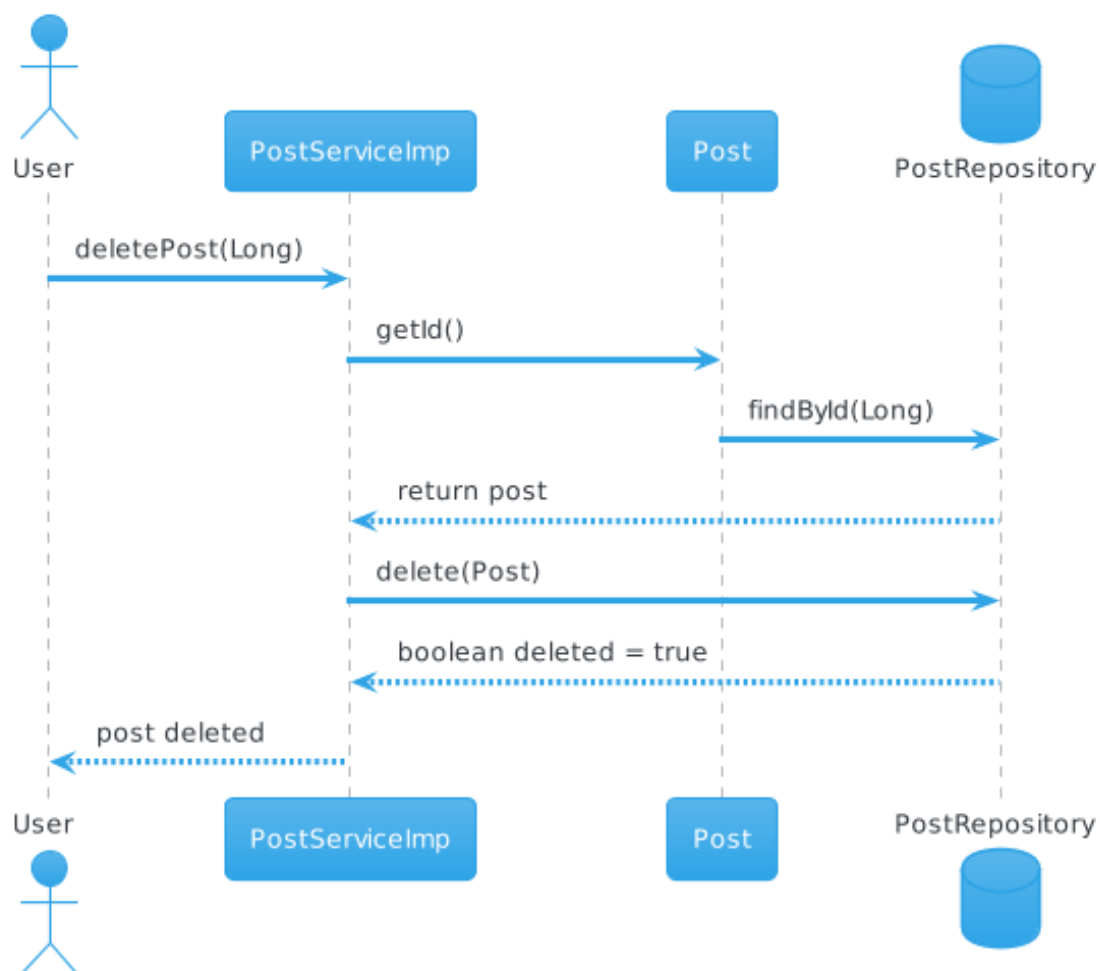


Figure 6. Sequence Diagram for Subscribing to a Community.

Figure 7. Sequence Diagram Deleting a Post.



Use Case Diagrams

Updated versions of our use case diagrams are given below, old versions are available and can be seen in the repository [wiki page](#).

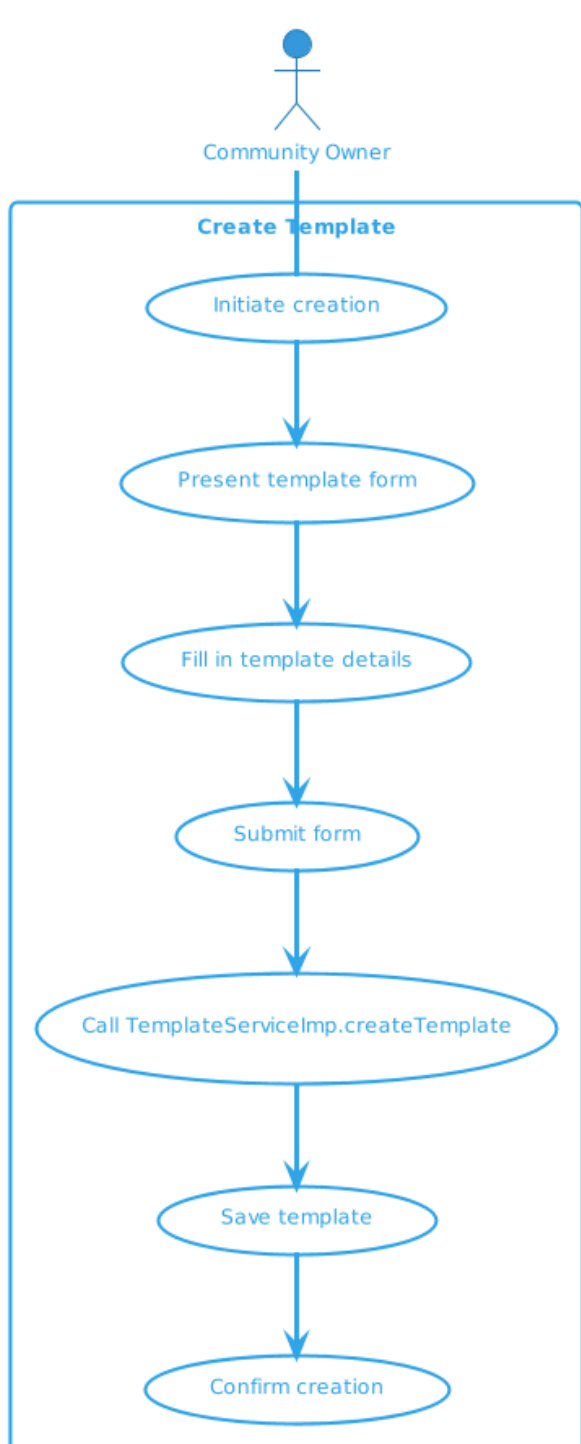


Figure 8. Use Case Diagram for Creating Template.

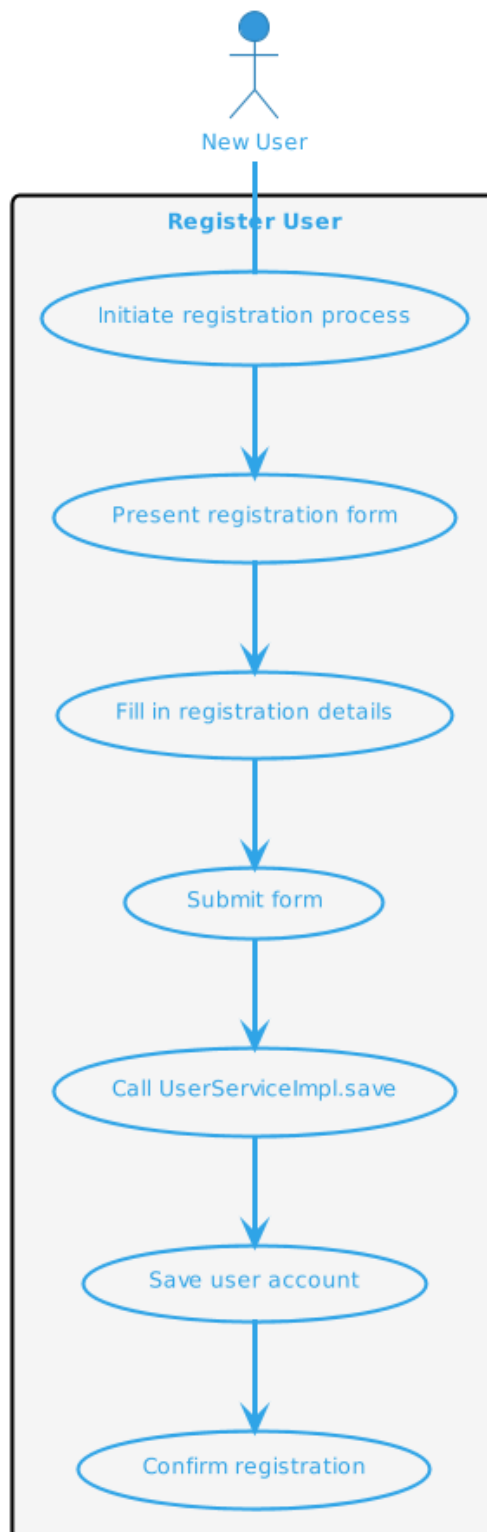


Figure 9. Use Case Diagram for User Registration.

Status of The Project

In this section we will talk about the status of every requirement. Is it completed? or Not completed? if its not completed fully what is the current status?

User Management:

- Requirement: Users can register with unique usernames and passwords.

Status: **Completed**. Users can register to system and saved to the database. Users can successfully login after registration.

- Requirement: Users can edit their profiles (including profile pictures, avatars, bio, etc.).

Status: **Not Completed**. Users can view their profiles and see their credentials but editing the credentials is currently unavailable. I wasn't prioritizing this requirement since user can already see credentials its easy to edit them but it was not prioritized.

- Requirement: Users have profile pages displaying their credentials and subscribed communities.

Status: **Completed**. Users can view their and other users profile page, also the subscribed communities are listed under the profile section. User can navigate to those communities from profile page.

- Users can view other user profiles. Status: **Completed**.

Search Functions:

- Requirement: Users can search for communities, posts, and comments.

Status: **Completed**. Users can search for communities for now but now for posts and comments since we haven't implemented the comments at this stage of the application. A basic search is available to search for communities.

- Requirement: System provides recommendations based on user search results.

Status: **Not Completed** Recommendation system is not implemented yet, it wasn't a priority based on the discussions during the class hours. I get the idea that it will be implemented in the next semester thus I didn't prioritize it.

- Requirement: Users can list all communities.

Status: **Completed**. Users are able to list all the communities in the main page along with the descriptions and owners. Users can also navigate to those communities choosing from the list or they can search for community name.

User Interactions:

- Requirement: Users can create posts.

Status: **Completed**. Users can create posts in communities of their choosing.

- Requirement: Users can comment on posts

Status: **Not Completed**. Users are yet not able to comment on posts.

- Requirement: Users can delete their posts.

Status: **Completed**. Users can delete their posts if they want to remove their content.

- Requirement: Users can delete their profiles.

Status: **Completed**. Users are able to delete their profiles. Once their profile is deleted, all their posts are also deleted. Its been discussed in the during the lecture in the next semester we will be implementing a feature to preserve the user data and maybe send it to user via mail after account deletion. This will be implemented alongside archiving.

- Requirement: Users can view and subscribe/unsubscribe to communities.

Status: **Completed**. Users can view the communities and they are able to subscribe or unsubscribe to communities.

- Requirement: Users can create posts with custom templates provided by communities.

Status: **Completed**. Users are able to create posts with custom templates or default templates. Custom templates are specific to communities meaning they are not shared among communities and custom templates are created by community owners or admins.

- Requirement: Users shall be able to create communities.

Status: **Completed**. Users can create communities providing a unique community name and a community description. Community descriptions are at least 188 and at max 255 characters. Purpose of this validation is to ensure that community descriptions are “descriptive.”

Moderation:

- Requirement: Communities have moderators can take action against offensive content and ban users.

Status: **Not Completed**. We have user roles for platform itself (for example platform admin) but not for communities since it's impossible to decide who will be able to ban who without any administration inside the community only community owners can take such actions.

- Requirement: Owners can delete communities and posts.

Status: **Completed**. Community owners are free to delete their communities along with the posts if they want. A user must own the community to be able to delete it.

- Requirement: Communities can be private or public.

Status: **Not Completed**. This requirement is also related to the hierarchy administration inside the community. Without the admins to accept or decline the subscription requests it's impossible to implement a

private/public community requirement. All the communities are behaving like public communities at the moment.

➤ Requirement: Community Specific Post Templates

Status: **Completed**. Community owners can create a custom templates in their community for the use of users. Users can view the custom templates in the post creation page and chose the template.

➤ Requirement: Communities have owners.

Status: **Completed**. Communities must have owners and community ownership cannot be transferred for now.

➤ Requirement: Owners are assigned during community creation.

Status: **Completed**. Since every community must have an owner. Creators are assigned as owners during the community creation.

➤ Requirement: Posts are listed under community pages.

Status: **Completed**. All the posts in community is listed under the community page by postdate. So newer posts will be on top and older posts will be down below.

➤ Requirement: Subscribers are listed under community pages.

Status: **Completed**. Subscribers of a community is listed under the community page users can see who is subscribed to the community.

➤ Requirement: Community owners can create custom post templates.

Status: **Completed**. Community owners can create custom templates with custom data fields and save those templates in the community so that users can use those templates to create posts if they don't want to use the default template.

Non-Functional Requirements:

Security:

➤ Requirement: Sensitive data such as passwords are encrypted.

Status: **Completed**. The user passwords are encrypted using "BCryptPasswordEncoder" Interface of Java Spring Security. It uses SHA256 algorithm to encode the passwords. User's passwords are stored

in the database as encoded hash strings. So even if database is leaked it's impossible to see users passwords.

- **Requirement:** Secure authentication mechanisms prevent unauthorized access.

Status: **Completed.** Users are authenticated using the form-based authentication method. When user is logged in to the application the authentication information is stored in the cookies of the browser.

Performance and Reliability:

- Requirement: Ensure minimal downtime and data corruption.
Status: **Completed.** Haven't been encountered with any data corruption or downtime at the current state of the application.
- Requirement: Scalability to handle increasing user and content load.
Status: **Completed:** Our application is dockerized and deployed in a cloud platform AWS. Our application is running inside a container in an EC2 instance. In AWS ec2 instances can scale up and down based on the load.

Usability:

- Requirement: User-friendly interface for easy navigation and interaction.
Status: **Completed.** The application has a simple and user friendly interface built using html, CSS, bootstrap and java script. We have utilized the most used and simple frontend tools to provide a smooth user experience. I do believe we achieved it. I have shown my application to my family and friends the common comment on frontend was "Simple and Effective."
- Requirement: Accessibility options for users with disabilities.
Status. **Not Completed:** Unfortunately, I haven't been able to add accessibility functions to my application. Like voice overs for visually impaired people. Voice overs are complicated mechanisms and may have had take too much time to implements and I would have left with little to no time to implement other requirements such as templates and posts.

Status of Deployment

Our application is deployed and containerized using docker engine. Our Docker files are available in the repository our deployment URL is also available in the title page.

Here is the screenshot of our application running in container on both cloud as a deployed version and on docker engine running in a local container.

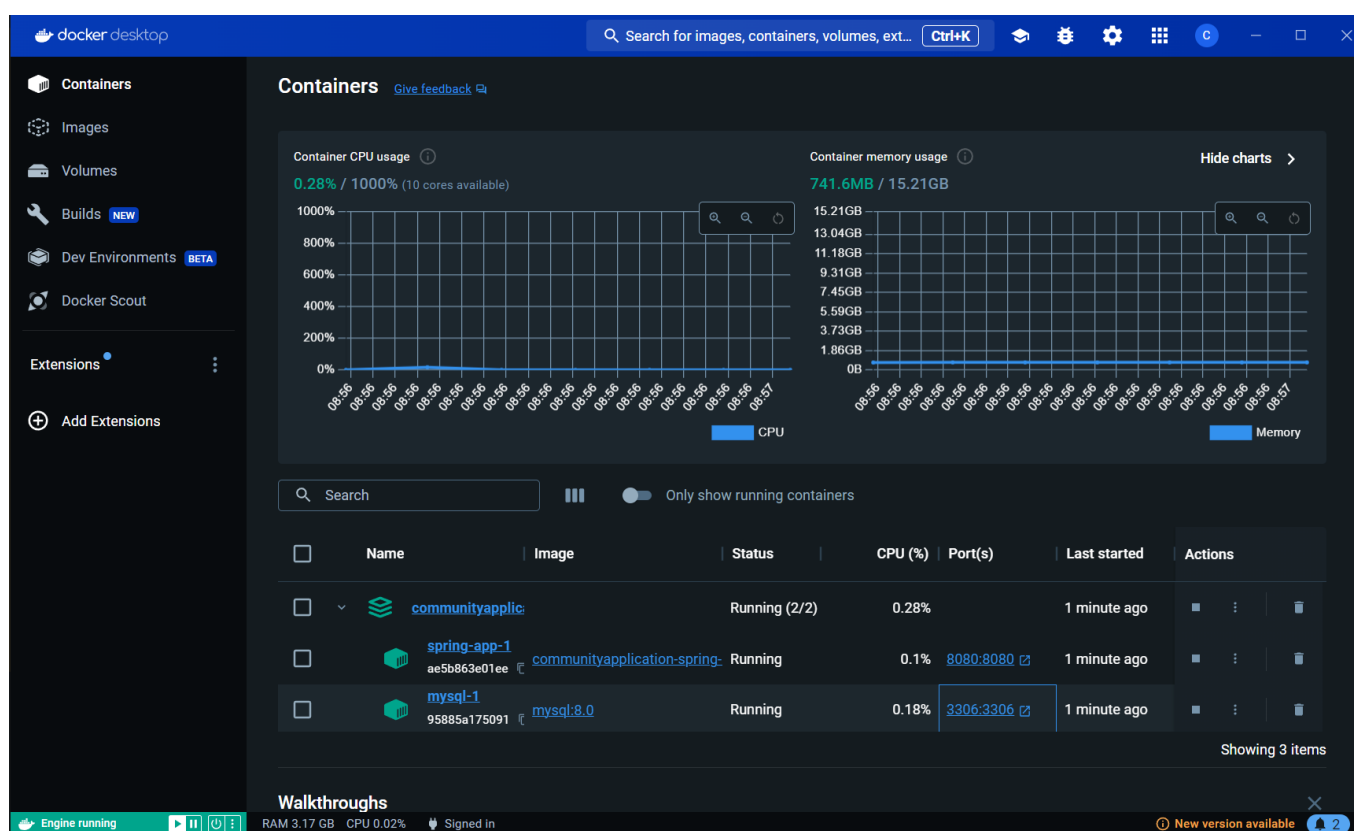


Figure 10. Dockerized application screenshot.

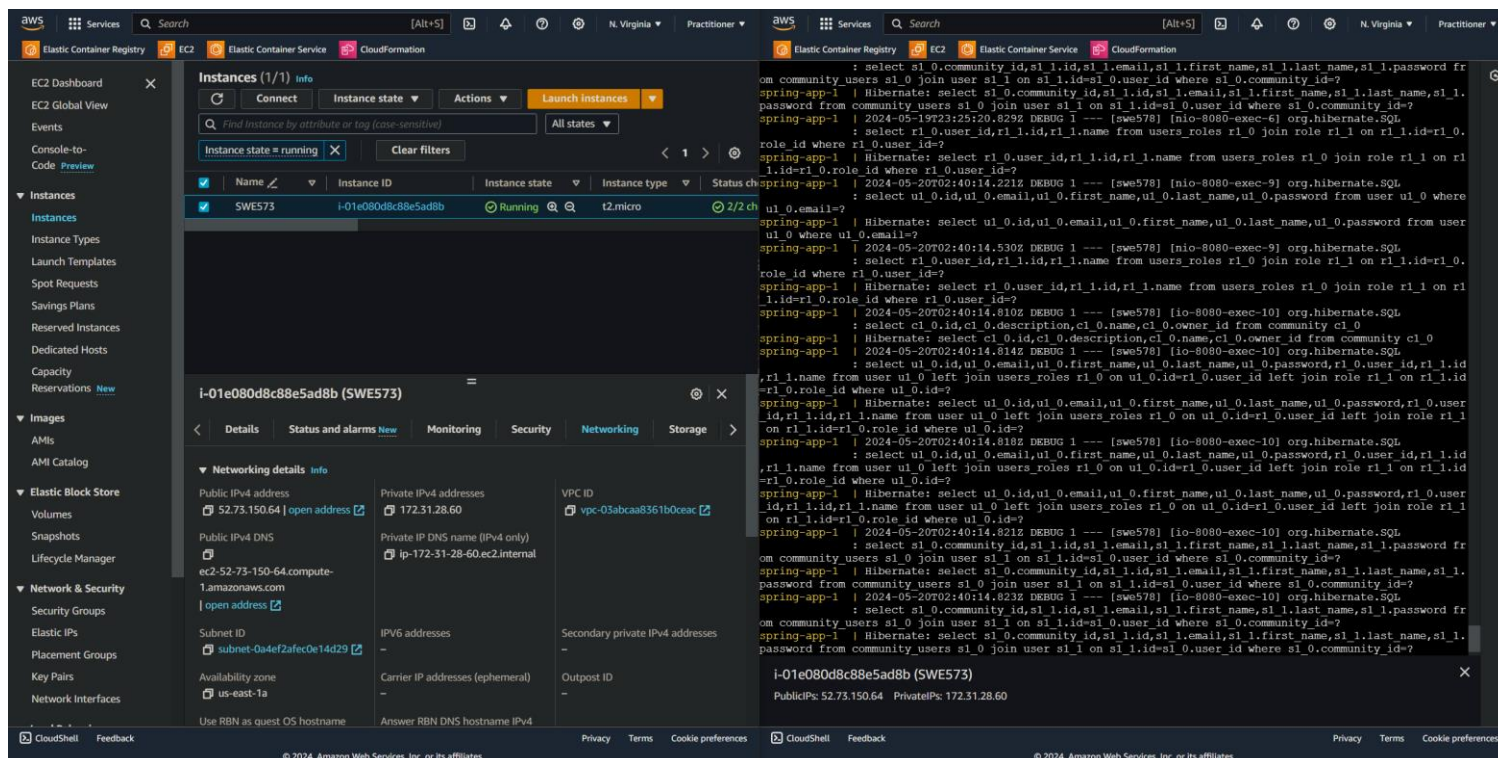


Figure 11. Deployed version of our application screenshot of AWS console and Command Line of our EC2 instance.

Installation Instructions.

Here is the installation guides for our application.

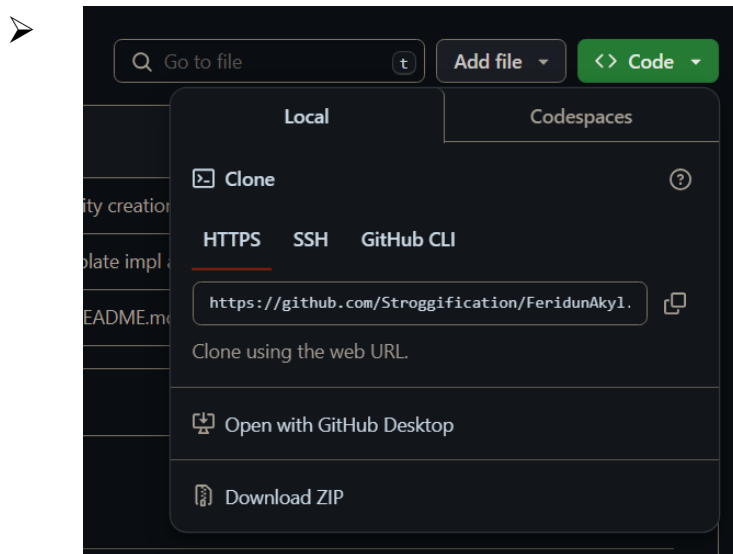
Local Installation.

If you want to run our application locally without any containerization technology switch to Docker instructions below if you want to containerize your application after getting the code. (Getting the code is same for both)

- Docker is recommended. **NOTE:** Do NOT change any application properties after you clone the repository if you are going to use docker.

1. Get the code from the repository.

- You can either download the code manually from the repository or you can use git to clone the repository.
- For manual download.
- Go to [repository](#).
- Click on the green <> Code button.
- Download ZIP



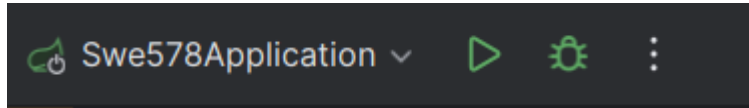
- Get the Code Using git
- If you have git installed locally you can use CLI to get the code from repository
- For Windows open PowerShell or CMD, for Linux open terminal.
- Clone the repository using: `git clone <repository url>`
- - git clone <https://github.com/Stroggification/FeridunAkyl>

Since you have successfully get the code let's get to installation.

NOTE: Continue to docker installation if you want to run the application in a container

- **Prerequisites:** You should have JAVA 21* and JAVA JDK(Java development kit) installed on your system along with maven. You should also have MySQL installed on your local system since our application uses MySQL database and application properties are configured for MySQL DB.
- You should also have maven installed to provide a runtime environment and build our spring boot 3 application
- You can check your java version by using cli type `java -version`
- Go to `application.properties` and change the database name of your choosing
- `#spring.datasource.url= jdbc:mysql://localhost:3306/YOURDBNAME`
- Create a schema with the same name in mysql
- `CREATE SCHEMA YOURDBNAME`

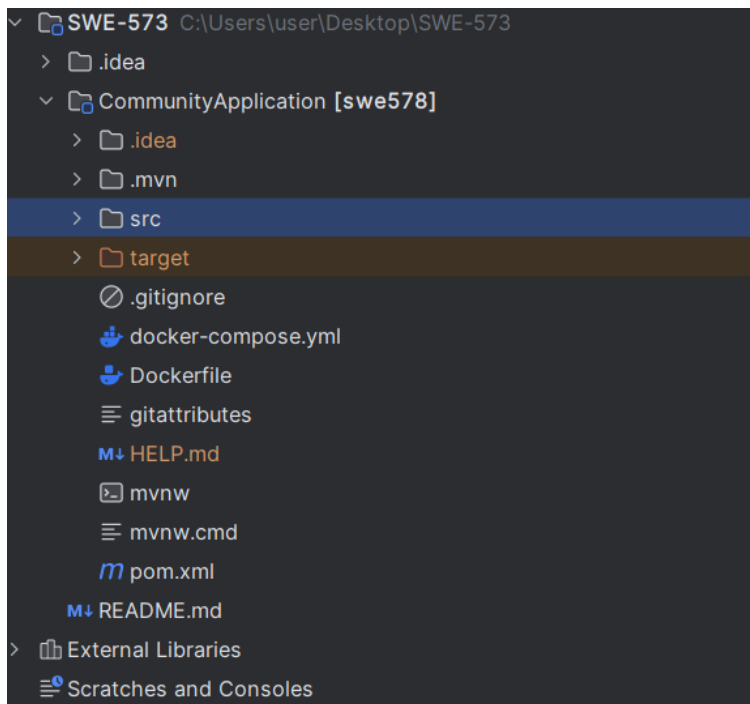
- Open the project folder in you favorite ide
- IDEs like intelliye or Eclipse will automatically fetch your project and dependencies using maven
- You can manually run you project without any ide by CLI
- If you are on IDE click run



- If you are using CLI type cd to your project directory
- Build the application using maven type mvn package
- Run the packaged application type
- java -cp target/appname-1.0-SNAPSHOT.jar appname.App
- Application will be running on localhost:8080

Docker Installation.

- **Prerequisites:** Make sure you have Docker engine installed on your machine both the docker compose and docker must be installed on your machine.
- You can check docker version by typing docker --version
- Cd to project directory where the docker-compose file resides.
- You must be at the same level as the docker-compose file



- Type “docker-compose up --build” in the CLI

- You application will be running on container after docker engine builds the application.

User Manual.

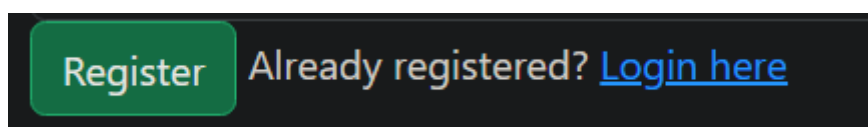
This is the user manual section where we talk about how to use our application.

Registration

To be able to use the application users must register using the registration form.

You may register using your credentials.

Fill the registration form and click on “register” button.



If you have registered successfully there will be a pop-up message indicating the registration is successful.

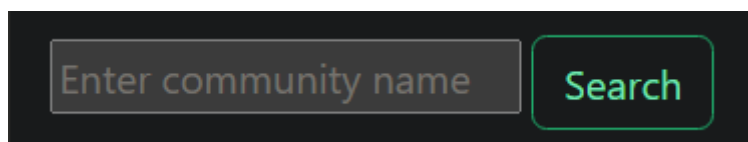
Login

Login to system using your username and password and click login.

If your credentials are right you will be redirected to mainpage. If your credentials are incorrect a pop-up message will tel you that your credentials are not correct.

Search Function

You can search for communities using search bar on the top right corner of the main page. Search is case insensitive its enough to write the community’s name you want to search for.



If the community is found you will be redirected to community page if the community is not available “Community not found” message will appear. Which means there is no community with that name.

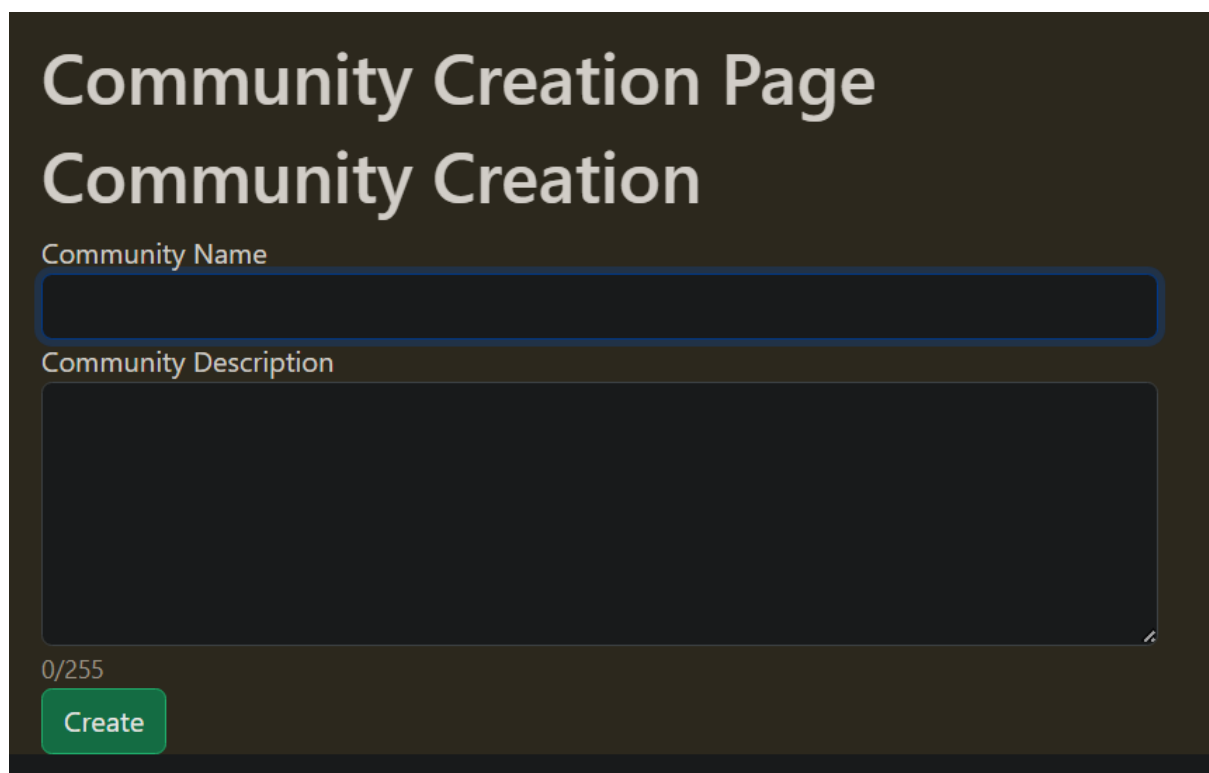
Creating a Community

You can create a community by clicking the “Create a Community” button you will be forwarded to Community creation page.

Here you will be giving your community a name and a description.

Community descriptions must be at least 188 characters and at most 255 characters long. The reasoning is we want community description to be descriptive and also not too long. Short and effective descriptions for your community is encouraged.

The system won't let you create a community without a descriptive community description and will display error messages. Saying your description is too short or too long.

A screenshot of a web form titled "Community Creation Page" and "Community Creation". The form has a dark background with light-colored text. It contains two input fields: "Community Name" (a single-line text box) and "Community Description" (a multi-line text area). Below the description field, there is a character count "0/255". At the bottom left of the form is a green button labeled "Create".

Community Creation Page

Community Creation

Community Name

Community Description

0/255

Create

Joining a Community

If you want to join an already existing community. First you must navigate to community page. You can either navigate to community page by searching or choosing the community from community list in the main page.

After navigating to community page you can use subscribe and unsubscribe buttons to join and leave the community.

Creating a Post

You can create post in any community by clicking the create post button in the community page. You will be redirected to post creation page.

You can either create a post using the default post template with “Title” and “Description” or you can use a custom post template.

Custom templates are created by community owners. If the owner has created any custom template, you will see them below the default template. If you want to use any of the custom templates just click on the one you want to use.

For both the custom templates and default template you must fill all the necessary fields to create a post otherwise system won't let you create your post and prompt a message telling you fill all the necessary fields.

Deleting a Post

If you are not happy with your post don't worry you can always delete it by navigating to your post and click delete.

Creating a Post Template

One of the most groundbreaking features of our application is post templates. Community owners can create custom post templates with custom data fields.

We already know how to create a community now lets navigate to our community and create a template.

Click on create a template button. You will be directed to template creation page. First you will need to name your template for example if its going to be a “Location” template where people enter the location coordinates you might want to name it as a “Template for Sharing Locations”

Then you can add fields to your template by using “Add custom Field” button

You will be prompted to enter a field name. You may also fill the fields for with examples or you may leave them blank its up to you.

Click “Create” button.

Deleting a Community

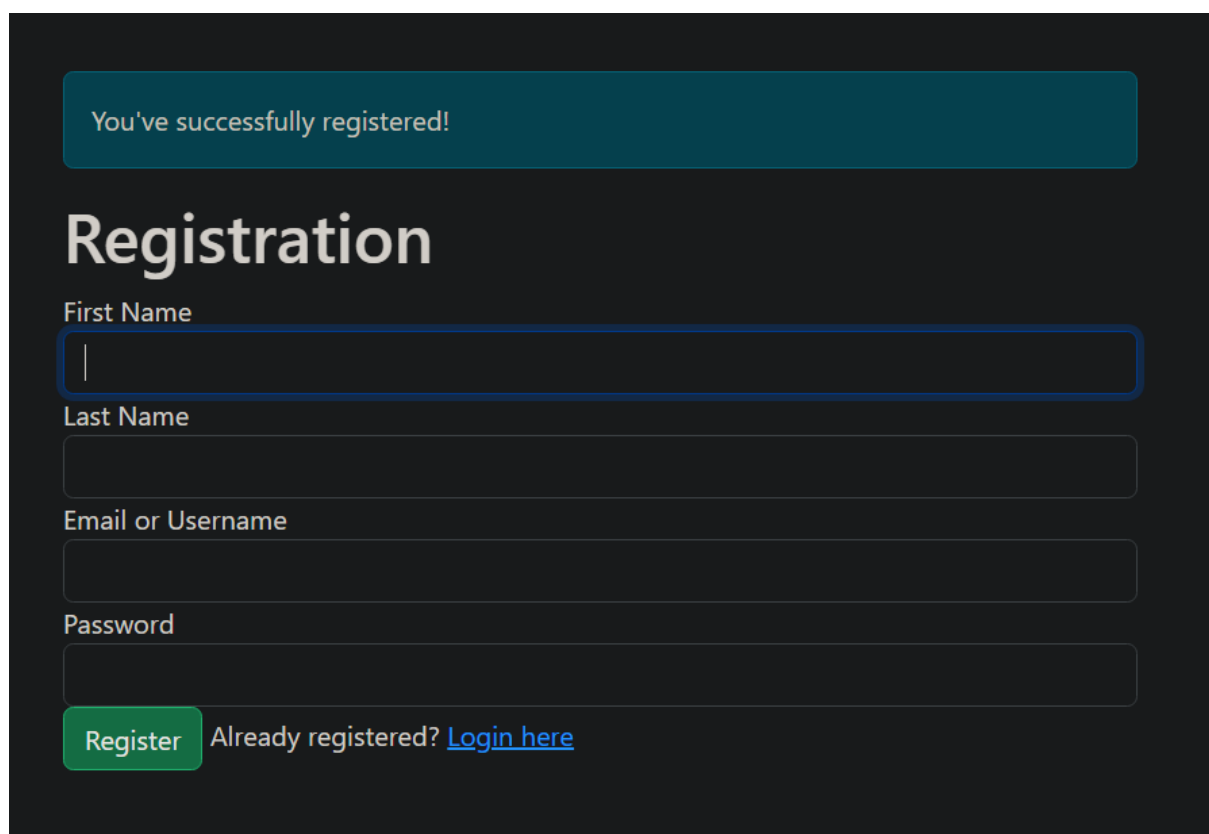
Only community owners are allowed to delete their communities. You can delete your community by clicking on delete this community button. Once you delete your community all the posts and templates will be gone so be careful!!

Test Results

User Tests.

Login/Registration

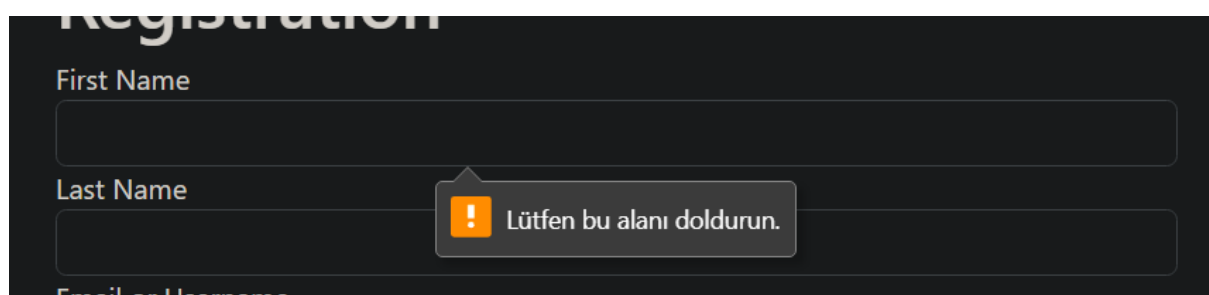
If user is successfully registered a success message will be displayed.



This screenshot shows a registration form on a dark background. At the top, a teal banner displays the message "You've successfully registered!". Below this, the word "Registration" is written in large white text. The form includes four input fields: "First Name", "Last Name", "Email or Username", and "Password". The "First Name" field is currently active, indicated by a blue border and a cursor. At the bottom left, there is a green "Register" button. To its right, the text "Already registered?" is followed by a blue link "Login here".

Users won't be able to submit the registration form without filling all the details.

Users won't be able to register with an already existing username since it's a unique constraint user will be forwarded to login page if he is already a user.



This screenshot shows the same registration form as above, but with a validation error. The "First Name" field is empty. A grey tooltip with an orange exclamation mark icon is positioned over the "Last Name" field, displaying the message "Lütfen bu alanı doldurun." (Please fill this field). The "Email or Username" field is also visible below it.

Community Creation

Users will be able to create communities.

Communities cannot have same names. Community name is a unique constraint so the application will throw an error if user tries to create a community with same name.

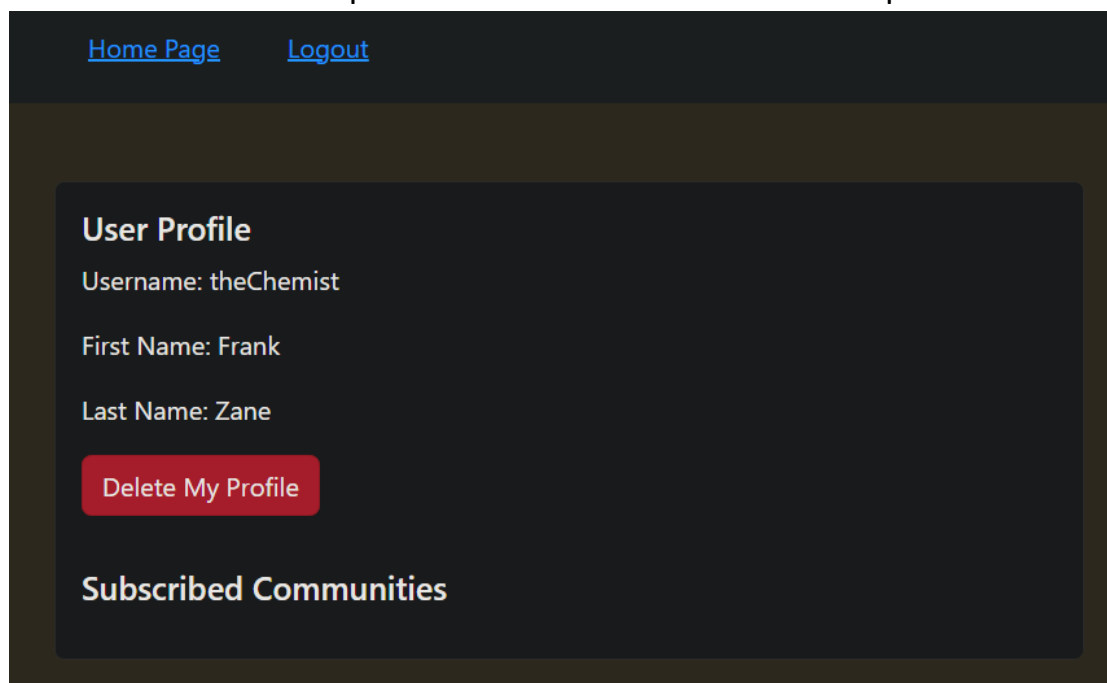
The application won't let users to submit a form without filling all the fields or if the user is violating the validations.



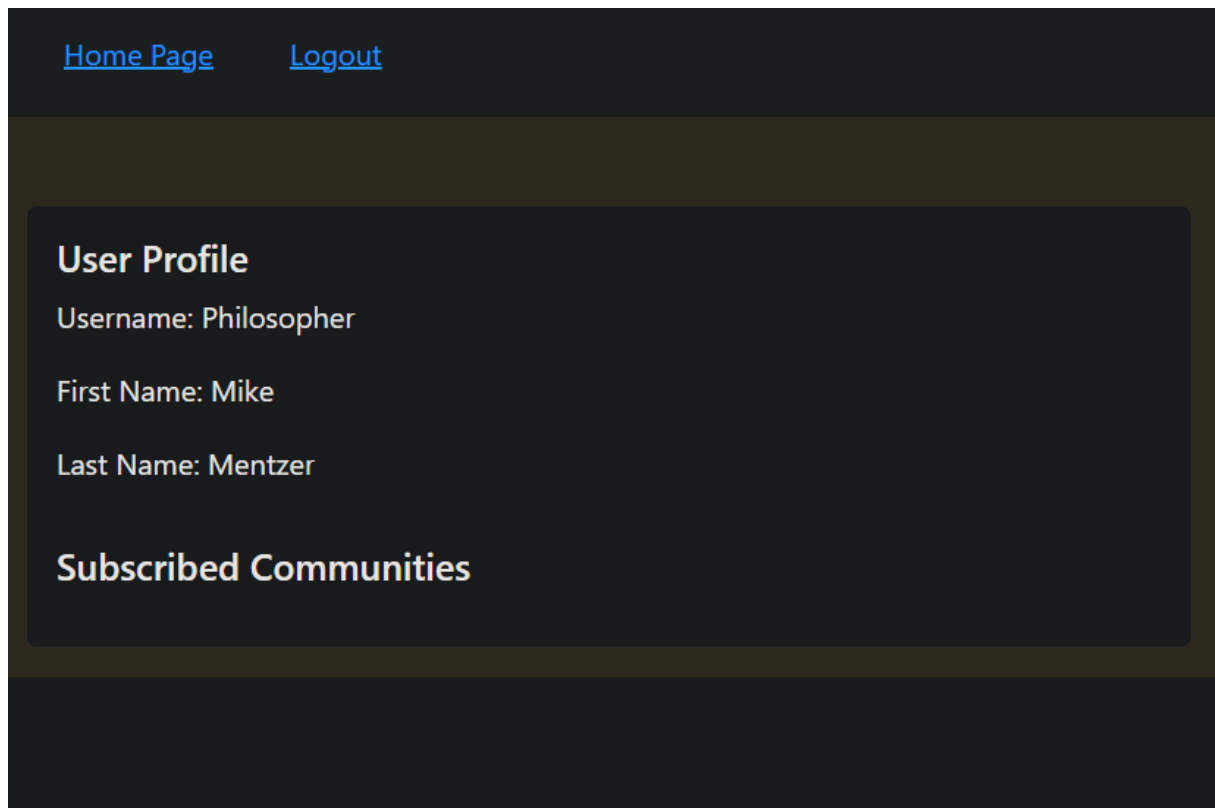
The screenshot shows a 'Community Creation Page' with a dark theme. The title 'Community Creation' is prominently displayed. Below it, there are two input fields: 'Community Name' and 'Community Description'. The 'Community Name' field contains the text 'Coding Community'. The 'Community Description' field is empty. A green 'Create' button is located below the 'Community Name' field. A validation error message is displayed in a grey box with an orange exclamation mark icon: 'Community description must be at least 188 characters long.' The character count '0/255' is visible next to the 'Community Description' field.

User Profiles

Users can delete their profiles but not delete other user's profiles.



The screenshot shows a 'User Profile' page with a dark theme. At the top, there are two links: 'Home Page' and 'Logout'. The main content area is titled 'User Profile' and contains the following information: 'Username: theChemist', 'First Name: Frank', and 'Last Name: Zane'. Below this information is a red button labeled 'Delete My Profile'. At the bottom of the page, there is a section titled 'Subscribed Communities'.



As you can see if a user is at someone else's profile page they won't be able to delete it.

Creating a Post

User can choose between default or custom templates while creating a post.

A screenshot of a post creation page with a light beige background. The title "Post Creation Page" is at the top, followed by the subtitle "Create Post with Default Template". Below the subtitle, there is a "Post Title" label and a text input field. Underneath that is a "Post Description" label and a larger text area. A character count "0/1000" is visible below the text area. A green "Create" button is positioned below the character count. Below the button, the text "You may also choose a custom template!" is displayed. Underneath this text is a box titled "Custom Post Template for Location" which contains two bullet points: "Longitude:" and "Latitude:". The form is clean and modern with a light color palette.

Title
Custom Post Template for Location

Custom Fields
Longitude
34.52
Latitude
41.015137

Submit

After post is successfully created with a custom template it will be displayed in the posts under the community

Community GYM COmmunity

SubscribeUnsubscribeDelete This CommunityCreate PostCreate Template

Subscribers

theChemist

Posts

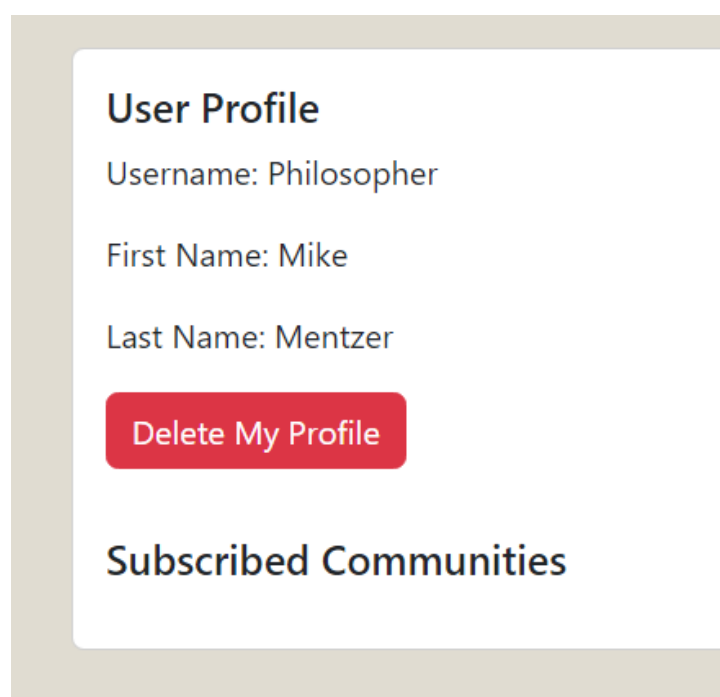
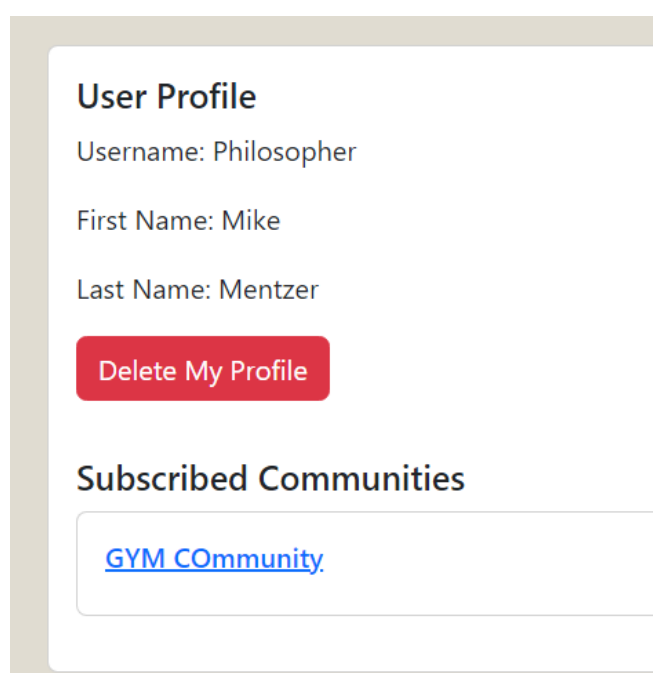
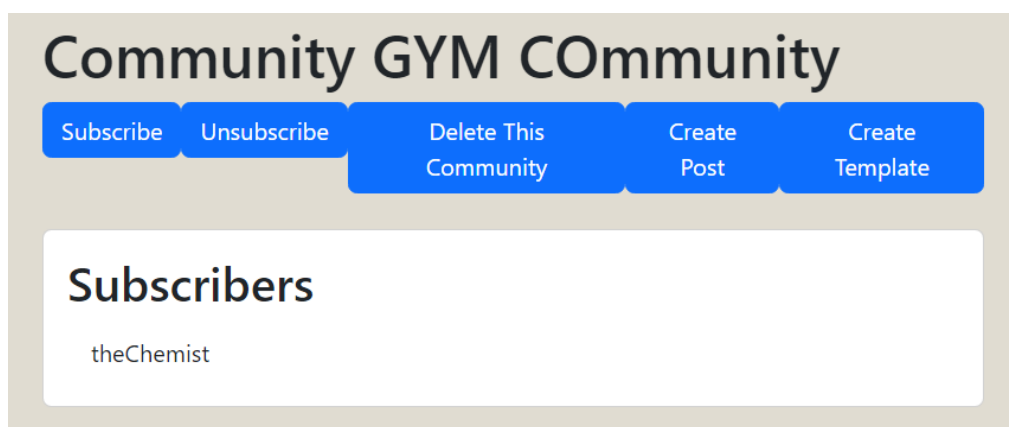
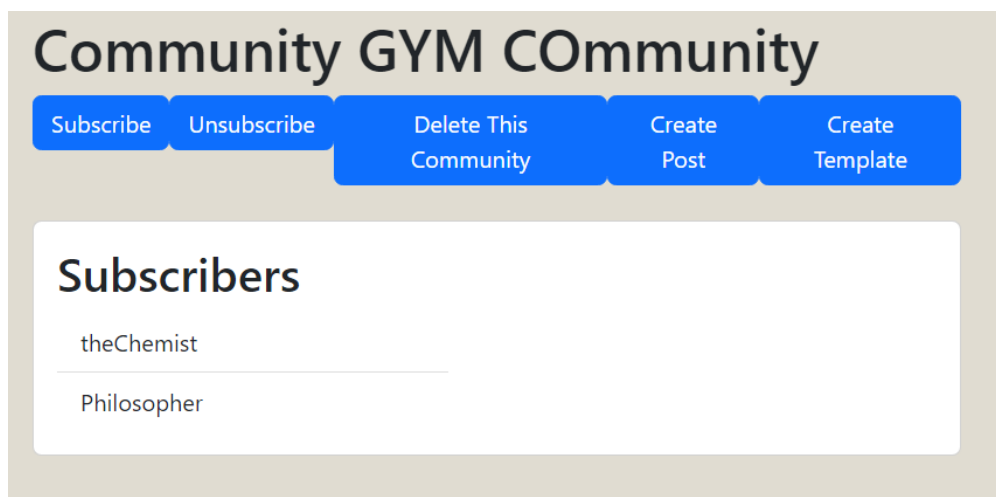
[Go To Post](#)

Post date: 2024-05-20 08:35:06.401
Posted by: theChemist

- Longitude: 34.52
- Latitude: 41.015137

Unsubscribe/Subscribe

If user is unsubscribed from a community they will be removed from the subscribers list of that community and users wont be able to see the community in their profiles any more.



Creating a Template

Community owners can navigate to template creation page and create a custom template. After the template is created its added to the template list for the community and displayed in the post creation page

Template Creation Page

Template Creation

Template Name

Create

Add Custom Field

Template Creation Page

Template Creation

Template Name

Custom Template for Sharing Images

Image URL

Create

Add Custom Field

Create

You may also choose a custom template!

Custom Post Template for Location

- Longitude:
- Latitude:

Custom Post Template for Sharing Images

- Image URL: