SOFTWARE DEVELOPMENT PRACTICE
SWE 530

FINAL DELIVERABLES

Student: Mehmet Oğuz Özüer
ID: 2022719249
Date: 20.05.2024

Project: Studyhole

Deployment URL: http://35.219.240.15:4200/

Git Repository: https://github.com/Oguzoz1/swe573-ozuer

Git Tag Version https://github.com/Oguzoz1/swe573-ozuer/releases/tag/v0.9

Wiki-Link: https://github.com/Oguzoz1/swe573-ozuer/wiki

Demonstration Video Link: https://www.youtube.com/watch?v=6KFqRHms8hI

HONOR CODE

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

Mehmet Oğuz Özüer

Username and Passwords for accounts

ID: ouz Password: ouz
ID: irem Password: irem

References

Referenced article to upload image:

https://medium.com/@kouomeukevin/how-to-upload-and-download-image-into-sql-database-with-spring-boot-c849ec5daec6

POSTING WITH POST TEMPLATE PROBLEM PLEASE READ:

The time this report is being written, v0.9 have no problem with creating and using post templates. However, after deployment, deployed application (although its entirely same with local application) is having problem with data conversion from input to string during post creation with created post template. Within the provided youtube link, it is showed that using post-template to create post is working. However, after deployment this issue came up. Hopefully, the problem will be resolved before checking the deployment link.

Table of Contents

PROJECT DETAILS

Project Name: Studyhole
Project Overview:

This project has been prepared with Spring Boot 3, Angular 17 and MySQL 8 with tools supporting Spring Boot Lombok, Mapstruct and Maven. The project intention is to create a community creation application to create community content management service. Specifically, studyhole, focuses on students or science enthusiasts who want to from and gather around communities and create study groups.

SOFTWARE REQUIREMENTS SPECIFICATIONS

Requirements listed below is extracted in a way from requirements elicitation during one of our class. Details exist within wiki:

https://github.com/Oguzoz1/swe573-ozuer/wiki/Elicitation-Questions-for-Study-Together-Community-Website

-System Functional Requirements-

Generalised System Functions

User Management

- The system shall have users, representing individuals who interact with the platform.
- The system shall have a registering and login system for individuals who want to be a user.
- Identity for a user shall be established through a unique username or email address along with optional profile information.

Community Management

- The system shall have communities, where users interact through creating a post, commenting on posts, giving positive and negative votes through up and down arrows.
- The system shall allow users to create communities by providing a name, description, and privacy setting (e.g., public or private).
- All communities shall have a unique name.
- The system shall not support communities within communities.

Content Management

- The system shall provide forms, templates, and interfaces when users intend to create posts, communities, and comments, with fields for contents, titles, and primitive data types.
- The system shall allow community creators and moderators to create custom post templates with predefined formats and fields.
- A post shall be a piece of content created by any type of user within a community, containing text, media, or links, which the post shaped by the community post template.

Reporting System

- The system shall have a reporting system that is intended for users to report any other user to the system.

Search Functionality

- The system shall have a search function to be used for searching users and communities.
- When any type of user searches another user, a new page shall list display names or used the display names.
- When any type of user searches a tag, a page shall list communities that have the tag.
- When any type of user searches a community by name, a new page shall list communities with that name or used the name within their name.

-User Requirements-

Registration and Login

- When users open the registration page, the registration system shall have fields for email, display name, and password.
- When an individual logs in as a user, the login system shall ask for email and password.
- User Types and Permissions
- The system shall support different types of users, including regular users, community owners, and moderators.

Community Interaction

- Any type of user shall have the ability to join any community and leave the communities they joined.
- Any type of user shall have the ability to create a community.
- Any type of user shall have the ability to create a post and send comments to posts within the community they joined.
- Any type of user shall have the ability to follow other users within the system.
- Any type of user shall see activities of other users within the system.

Profile Management

- Any type of user shall have a profile page containing personal/profile information and activity history.
- Personal/profile information in any user's profile page shall include profile picture, display name, stats (StuPoint, Number of posts and number of comments), field of study, and a list of joined communities.
- Any type of user shall have permission to change the field of study and profile picture in their profile settings.
- Profile settings for any type of user shall be available in their profile page, bottom right of their personal information.
- When any type of user forgets their password, the system shall provide a button in the login page to send provided email to change the password.

Moderator Permissions

- Moderator permissions shall allow permission owners to create templates for community posts.
- The system shall have moderator permissions to allow moderator-type users to regulate user-generated content through allowing moderator-type users to delete community posts, comments on posts, and ban users from their respective communities.
- Moderator-type users shall possess moderator permissions to regulate user-generated content within their respective communities.

Community/Post Requirements

Community Creation and Management

- When a user wants to create a community with an existing community name, the system shall give a pop-up that warns the user that the name is already taken.
- When a community owner wants to archive the community, the system shall provide a button within community settings which exists in the community home page as a button.
- When a community owner wants to grant moderator permissions to a user, the community owner shall find the designated user from the member list within the community settings page.
- While community owner type user is leaving their own community through leave community button in the community home page, the system shall give an error pop-up that warns the owner to transfer the community ownership before leaving the community.

Post and Comment Interaction

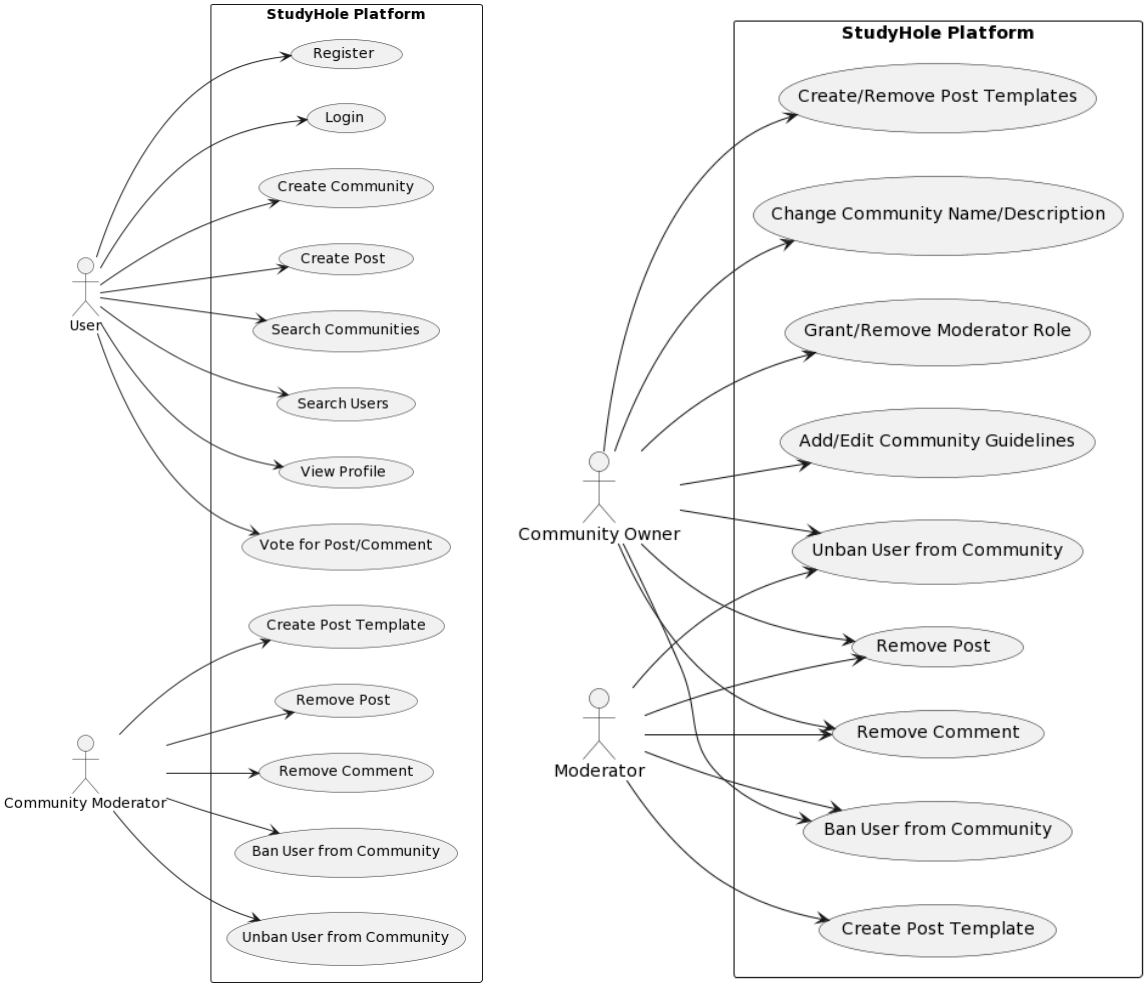
- When a user wants to vote a post or comment, the user shall press up and down (respectively: like, dislike) arrow button attached to the post or comment.
- When a user intends to edit their post or comment, the user shall press the "edit" button that is attached to their post or comment.
- When a post or comment is edited, the post or comment shall have a timestamp and edit mark.

Community Setting Page

- Community shall have a community setting page.
- Community setting page shall provide a section for moderator-permission owners to delete and add post templates.
- Community setting page shall provide a section for community owners to transfer community ownership to another user.
- Community setting page shall provide a section for community owners to change the name and description of their communities.
- Community setting page shall provide a section for moderator-permission owners to ban or unban users in the community.

DESIGN DOCUMENTS

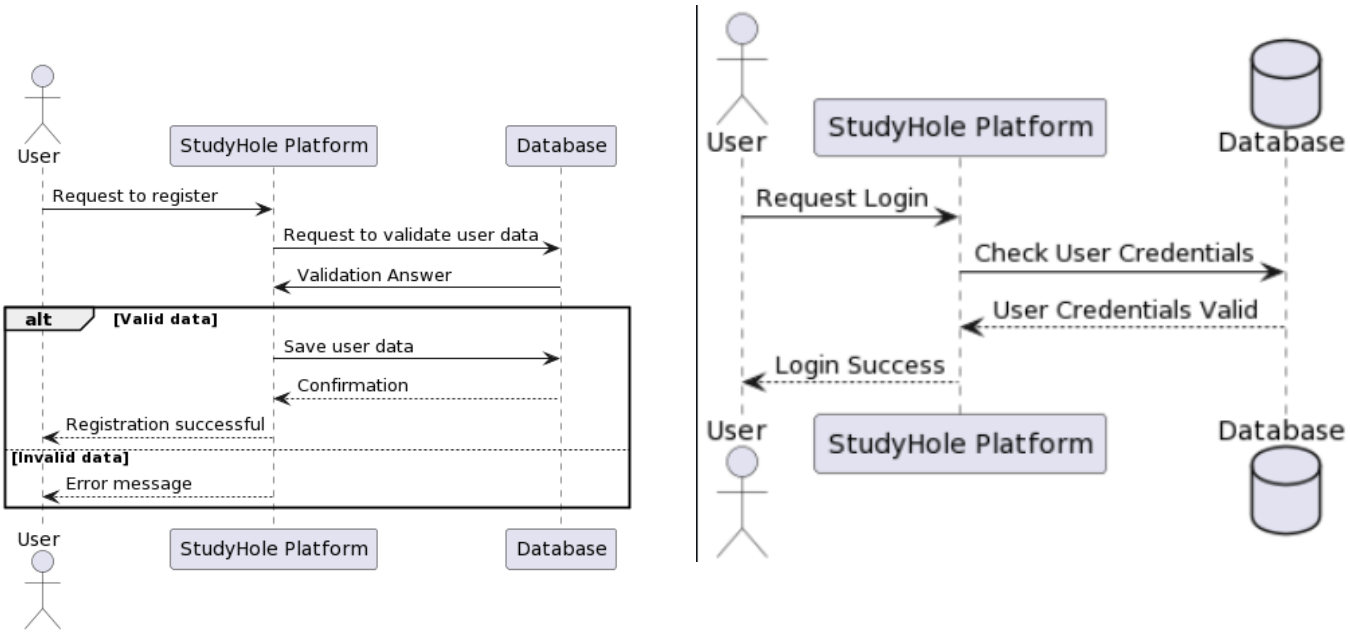
USER & COMMUNITY MODERATOR USE-CASE



These diagrams are displaying “Roles” of the users where each type is attended a role. Meanwhile, a user can do any simple tasks that community is providing, community management is responsibility of moderator and community owners.

In this sense, some of the permissions are shared across moderator and community owner but since the community owner has greater rights upon moderators, they have permission to grant roles, change community name and description, create and remove post templates.

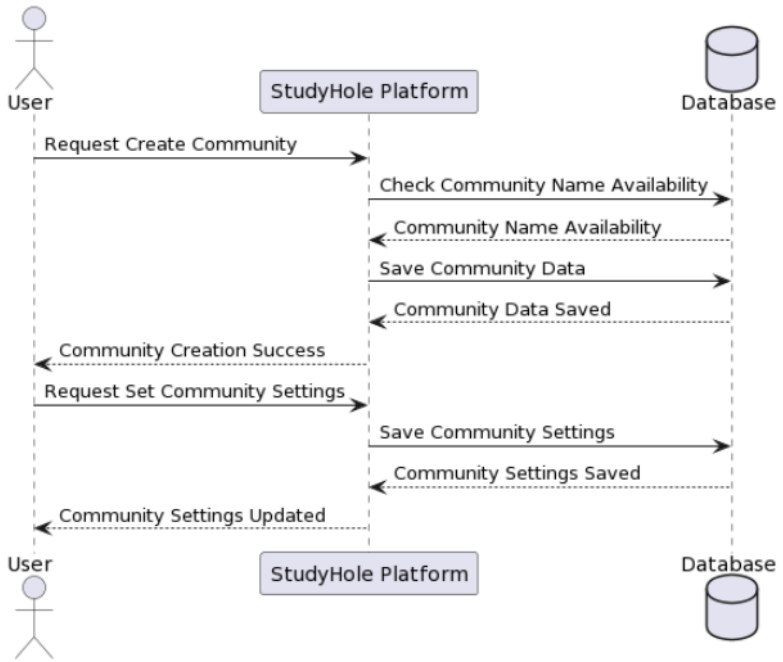
REGISTRATION AND LOGIN SEQUENCE DIAGRAM



This sequence diagram displays the interaction between user and database. User sends action to register and frontend makes an API call to request if it is correct. If the data provided is valid, then the database confirms and saves the user data. The confirmation that comes to front end sends a successful registration pop-up, else presents error pop-up.

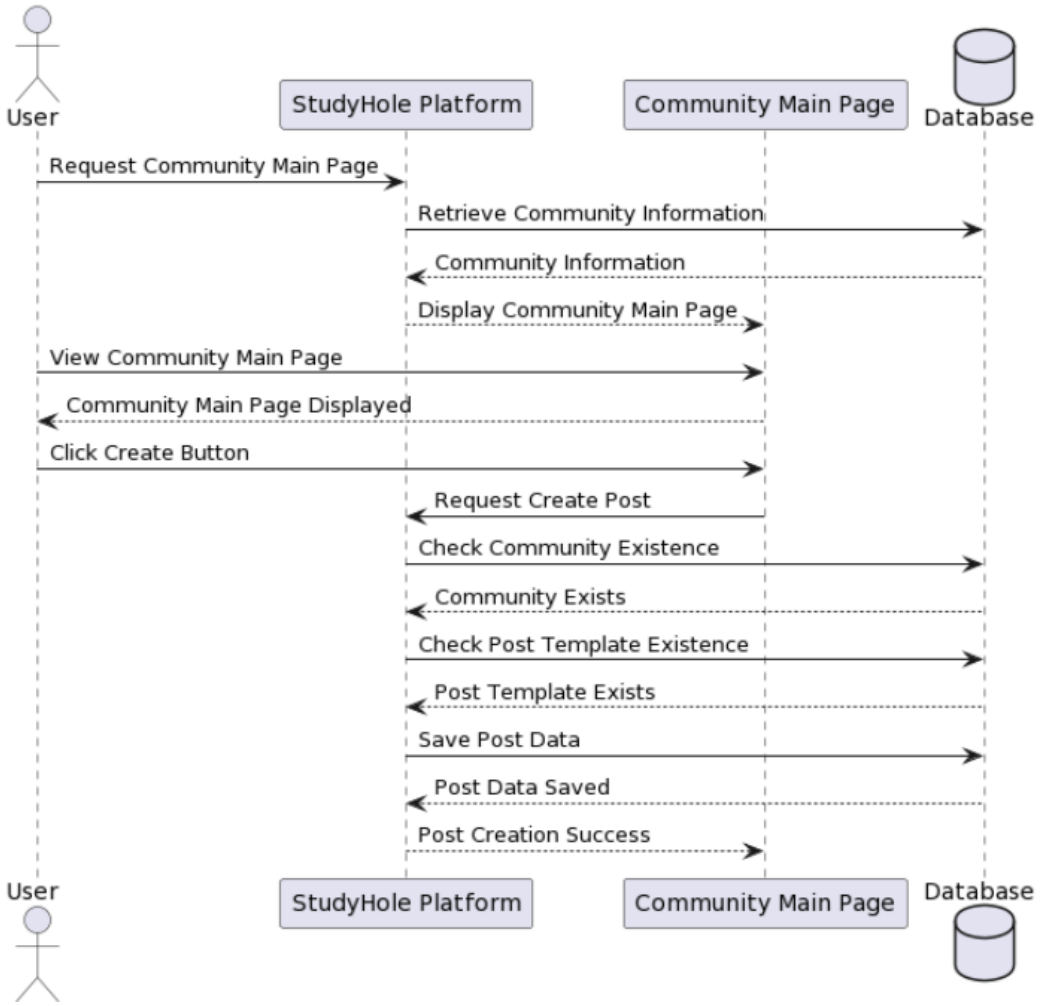
Login Sequence acts as similar to register. It gets confirmation from database and logs in the user.

COMMUNITY CREATION SEQUENCE DIAGRAM



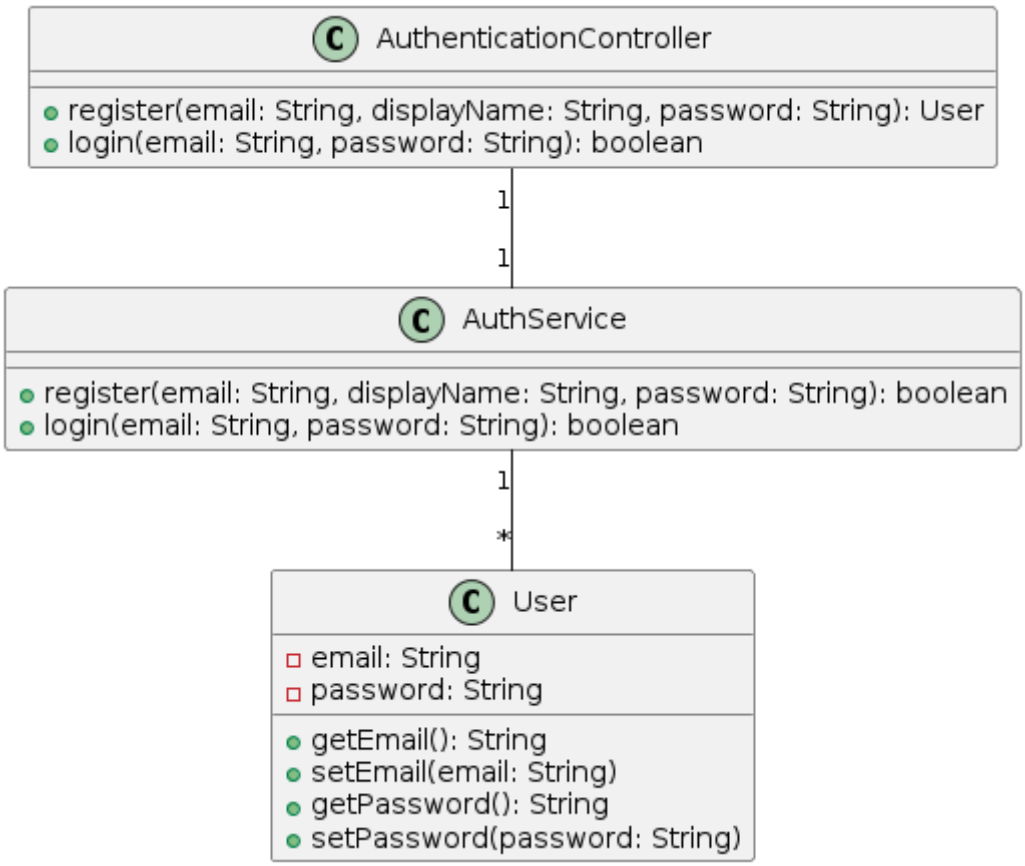
Community creation works as similar, it sends request to create a community and checks if the name is available, if it is available then it saves the community data and represents community creation success. However, in the application, community settings **do not require secondary call**, but user must have already given the description. Thus, it does it within one API call.

POST CREATION SEQUENCE DIAGRAM



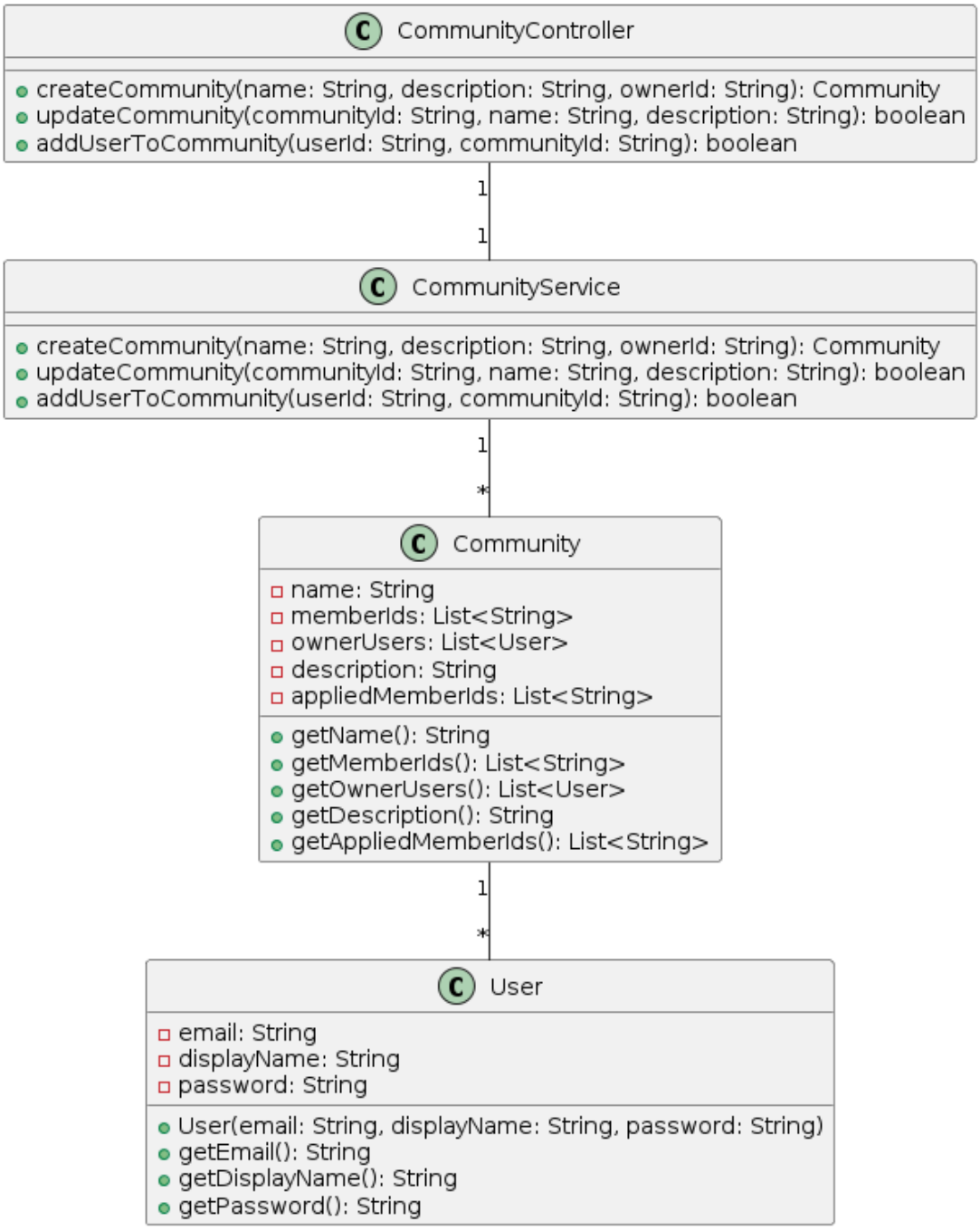
This diagram was an initial design document. However, through out programming the design, **it has changed.** Instead of confirming one-by-one, the community entity already holds information of post-template. During choosing post-template, the system only displays the existing **templates** therefore, we **do not require** to confirm the templates exists. Thus, the system checks the **community only**. And, receives the data related to that.

USER SERVICE CLASS DIAGRAM



This diagram is displaying how it executes login and creation of user. AuthController is responsible for API calls and service is executing logic. User is the entity that contains email, password and username. After Frontend makes a call AuthenticationController is being engaged and that triggers AuthService. Each of them returns user to the frontend.

COMMUNITY CREATION CLASS DIAGRAM



Communities include ownerUsers, memberusers, descriptions and such information. Community Service has more methods but in this case we describe updating, creating and adding user. In this diagram, we have relation that user could be many for Community. This is particularly true of ownerUsers since we contain members within an Id to avoid JSON recursion.

STATUS OF THE PROJECT

COMPLETED REQUIREMENTS

- User can register and login.
- User can create community with a description and set privacy (private or public).
- User can post to subscribed community within community profile.
- User can view posts, list of members, community picture, community description, member count within a community profile.
- Users can apply private community, cancel application, and leave community.
- User can create a post template with Text, Image, URL and Date Fields, and add as much data field as they want with template name field only appears once.
- Owner users can see applications and accept the users.
- Owner users can upload community profile pictures.
- Users have profiles and view their post and comment history within their profile.
- All users can see each other's profiles.
- Users can do simple search on "Community" and "Users" to find interested people or communities.
- Owner users can edit community guidelines.
- Users can vote posts.

NOT COMPLETED

- No moderation roles for users. Therefore, there is no way transferring community ownership, or any related stuff.
- No deleting comments, posts, or communities.
- No reporting system.
- No setting page and there is no way changing description of a community.
- No profile settings for users or they can not edit their profile.
- No geo-location for datafields.
- Users cannot vote comments.
- No advanced search.

STATUS OF DEPLOYMENT

Deployment URL: http://35.219.240.15:4200/

Project is deployed on GCP. Github repository contains Dockerfiles for angular and spring, as well as docker-compose for mysql-8 jdk17, spring and angular.

Issue regarding to using post-template to create post has been addressed which occurred on deployment in youtube-video and title page. The issue might be resolved during the time of checking the deployment.

INSTALLATION

Please pull repository from my github profile.

This is installation guide for windows and Linux. Please add PATH location of the downloaded files within your windows system.

Java Installation:

Windows:
Install Java 17 from:

Java Downloads | Oracle India

Linux:

"sudo apt update"
"sudo apt install wget"

Download JDK 17:

"wget https://download.oracle.com/java/17/latest/jdk-17_linux-x64_bin.tar.gz"

Extract:

"tar -xvf jdk-17_linux-x64_bin.tar.gz"

Move Directory:

"sudo mv jdk-17 /usr/lib/jvm/"

Set Java Home:

"export JAVA_HOME=/usr/lib/jvm/jdk-17"

"export PATH=$JAVA_HOME/bin:$PATH"

Spring Boot Installation:

Maven download: Maven – Download Apache Maven

Linux:

"wget -qO- https://apache.osuosl.org/maven/maven-3/3.8.4/binaries/apache-maven-3.8.4-bin.tar.gz | tar -xz -C /opt"

Angular 17:

Linux/Windows:

"npm install -g @angular/cli"

MYSQL 8:

For Windows, please download mysql 8 workbench and create a database called "studyhole".

https://dev.mysql.com/downloads/mysql/

For linux, please create studyhole:

"sudo apt install mysql-server"

Secure Installation:

“sudo mysql_secure_installation”

Docker

Make sure you have docker in your system. Move to the directory to “docker-compose” file exists. Then type in CLI:

“docker-compose –up build”

Application Properties and Keys for Local Run:

Create “resources/application.properties” Example content:

```
############## Database Properties ##############################################
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/studyhole
# spring.datasource.url=jdbc:mysql://mysql:3306/studyhole
spring.datasource.username=root
spring.datasource.password=123456
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
spring.jpa.hibernate.ddl-auto=update
spring.datasource.initialize=true
spring.jpa.show-sql=true

############# Mail Properties ###################################################
spring.mail.host=sandbox.smtp.mailtrap.io
spring.mail.port=2525
spring.mail.protocol=smtp

############# Domain Properties #################################################
app.url=http://localhost:8080

############# Public Key ########################################################
jwt.public.key=classpath:app.pub
jwt.private.key=classpath:app.key
jwt.expiration.time=900000
```

Create “resources/app.pub”

-----BEGIN PUBLIC KEY-----

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAsEUhXr6cHgKkpftNVih0

XmFZNlwPoiCGzQRucI9Rhvcios98Tm5XDhimbdhBaTSuuuucwvEHDTzCkdHhUfwD

oKbvS/FwX5Wzgxc12W3ahihkf0Ejw1+fWIaONbAKzKBMfo5Ih4K7enDHq+Rf90Bf

x+uPqQOe9L0XfxoQU3E5iOQF4JgmHrau6RwMVyELe8GXGVUCw1txjPNqe66a9sUP

l5CJ7Qi98nm8w0ALydlqgyUxqdKKvpa4pYiG7rThFuhxRPBrZP/Y2AouiIgr3MR0

paAGkbw0AXlxPDBIBC4ZPYNdjyhAW51C3yVbj07Q7zA2JWBsFz7h47LBArcH1qAP

VQIDAQAB

-----END PUBLIC KEY-----

Create “resources/app.key”

-----BEGIN PRIVATE KEY-----

MIIEvwIBADANBgkqhkiG9w0BAQEFAASCBKkwggSlAgEAAoIBAQCwRSFevpweAqSl
+01WKHReYVk2XA+iIIbNBG5wj1GG9yKiz3xOblcOGKZt2EFpNK6665zC8QcNPMKR
0eFR/AOgpu9L8XBflbODFzXZbdqGKGR/QSPDX59Yho41sArMoEx+jkiHgrt6cMer
5F/3QF/H64+pA570vRd/GhBTcTmI5AXgmCYetq7pHAxXIQt7wZcZVQLDW3GM82p7
rpr2xQ+XkIntCL3yebzDQAvJ2WqDJTGp0oq+lriliIbutOEW6HFE8Gtk/9jYCi6I
iCvcxHSloAaRvDQBeXE8MEgELhk9g12PKEBbnULfJVuPTtDvMDYlYGwXPuHjssEC
twfWoA9VAgMBAAECggEBAJPM43Y2DWF8heJHHUmpEioxJkTWXKVs8JrnpFLtWUb5
4ijyISEClCxVrvdb5Tz516kT9pXe1DLdR6hNe8jqr3/9eZkMEVWUiLjatWRjIGBt
bhbl377qZworiQKrYeLCZmGXnGlDzT+VYVdYXr9LUwSnGAZ36sj75L/GcvoEbmHA
qDUz0IkvvNawHKfcvld6Mcg5Or1PPJx2HJEGiMKRMv1wWvWtTBBbI3NRsT9UMscx
OwNKlyEIVMby921w3sWBsqzGlMQ+Jq/nMqViMEqT26Y21BUkBGO407UyiujokDuB
KkJvRsVTPslPHie1Ngwk1h7TaUC6vQLOzD4xTsVwa+ECgYEA+FSfGLa+chMt+Ad1
X03SrtuQKD/SX30OxeQo4pYotIP4ZJNpjVvM26icJ+HprRn4zoLyH2Qs10ne+djH
OH/m0UYD9COIjKlpYPXhKE91ZT3pShyCrMZC1xeGKSLSrX7qE8LEZPxnVoeP1hfD
GTbF3fqHlWVuhOvAboOPMOZo40cCgYEAtbbGrOTIK4NRrvorU670pAcn8A/hgCym
674BtJusPmLGbpe16mwUp8qsqhjh5z98888B/MJu/xCRErFWlo8tfVfEUe+4w++n
UC3RSaHvvA/vHQDL1rogd5EgCfTVIsDKh+irhPYQ0k6mOGZ/iotOfnZ1XhxFxUus
CZzezoV7LoMCgYEApZzx2a6ORk/KOX7n5R5mopzlgCJIL8ycXJe0OKECFplov9Kb
811Euz2wntyIVhxbFYa46PyK9y1Z2nCbNHAARJMYMeq8t2qIT9TLS3HqjhGDExz+
fTFgBEi4/ZguvuxEOdeL2PmIBWo1IAgK4jboMQlTAOx7j4QWet2cUnVngCMCgYEA
lMp/7gw7d53EodFI7vHgcftGEYT/cbOeToQ740ZhCIha0S26kKRFWYtUTMepGnBb
AUxs5SnUN1i6YaMCMrA0YGEgaLMxCIdQ6g9KQs+tfMFTWt1PC5lGgoE7yctHRni0
ngq/f1T5OyuZYa9UGf6tnFGlExV1b/LhgNGJg3O7BL8CgYASLd8MfI40qaljuGSw
0hulB3SHls8eoOBzSuWo6z+6phlESTBIAKOLfVi/8GZfzGP445dvJc4t5W0CY0zj
A8IMstZRDe5Vf3AKhXWSfT9y3Ab8ZA3UF+kdJeul8Un3RF5WUtDl94cAQlPb9Owo
95H/obKrFQMIs3F2i4dCxjPxCw==

-----END PRIVATE KEY-----

USER MANUAL

Please proceed to register and use all the systems.

As a user you should:

1- **To Register**: Press sign-up button to fill required fields. Then, please press sign-up button.
2- **To Login**: Please login with your credentials by clicking on top-right screen “login” button.
3- **To Create Post**: First, you should join a community. First, in home page, look at browse communities, or find in post history on main-page, or search in the search-bar. Then, on top right, of the community view page, there is a green button says subscribe. Then refresh and you will be able to see create post button just above create post template. Then proceed to choose post template or default template.
4- **To Create Community**: In homepage, press create community button and fill required places.
5- **To Upload Image to Community**: In community view-page, press settings button to reveal image upload and then, please proceed to upload 20-30 kb worth JPG image.
6- **To Edit Community Guidelines**, if you are the owner, press edit button on the view community page, edit the guidelines and save.
7- **To create post template**: press create post template, name the template and press plus button to add extra data fields. You can select one of the four data types and field name respective to that. Then press create template button.
8- **To Comment:** Please view a post then you will see a section to be able to comment.
9- **To Vote:** In homepage, post-view or community-view you will be display upvote and downvote buttons to vote.
10- **To apply community:** If community is private, then you can press apply.
11- **To accept community:** If you are owner, and you see applied members on the community homepage. You will have accept button right next to each user.

USER TEST: Creating a Post Template

Prerequisites:

- Registered Account
- Subscribed to any community

Test Steps:

1. **Login**:
- Click on the "Login" button in the top-right corner of the screen.
- Enter your registered credentials (username/email and password).
- Click the "Login" button.
- Verify that you are successfully logged in and redirected to the homepage.
2. **Join a Community**:
- On the homepage, browse communities or use the search bar to find a community to join.
- Click on the community name to go to the community view page.
- Click the "Subscribe" button on the top right of the community view page.
- Refresh the page and verify that you are now subscribed to the community and the "Create Post" button is visible.

3. **Create a Post Template:**
- On the community view page, click on the "Create Post Template" button.
- Enter a name for the template in the "Template Name" field.

- Click the "Plus" button to add extra data fields.
- Select one of the four data types (e.g., TextField, ImageField, UrlField, DateSField) for each field.
- Enter a name for each field respective to the data type selected.
- Repeat the above two steps to add multiple fields as needed.
- Click the "Create Template" button.

Expected Results:

Step 1: User should be able to login successfully and receive a confirmation message.

Step 2: User shall be able to join a community which is browsered through main page.

Step 3: User shall be able to create a post template by adding data fields, and the create template should appear in the list of available post template during post creation.

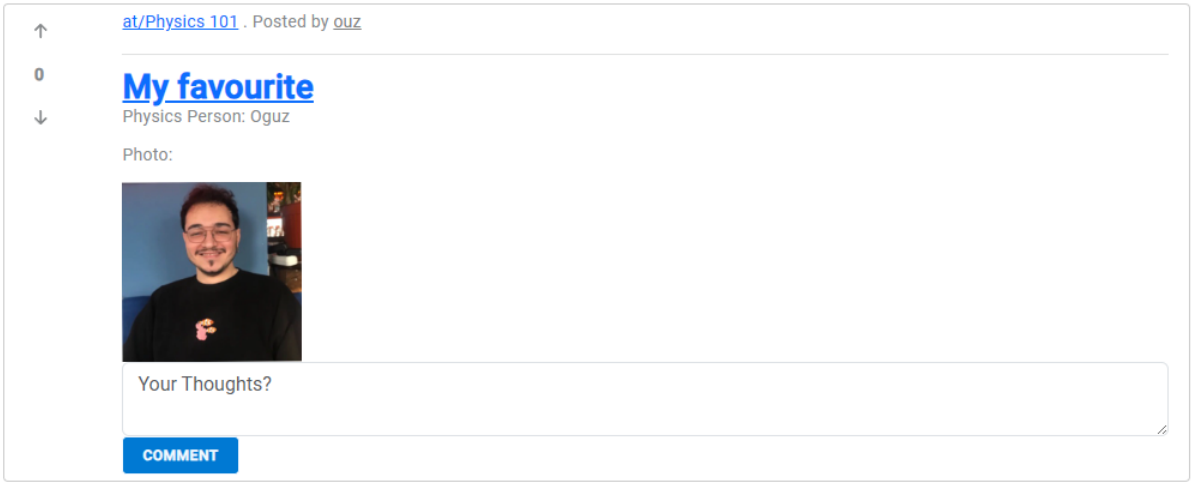
Notes:

- Test with different data type combinations.
- Test with creating a post with new post template.

RESULTS:

1- User is successfully registered and logged in.
2- User could subscribe to a community.
3- User could create a post template.
4- User could create a post with post template.

Screen Shot of the fourth result:



USER TEST: CREATING COMMUNITY

Prerequisites:

- You should have a registered account and be logged in.

Test Steps:

1. **Navigate to Create Community:**
- On the homepage, locate the "Create Community" button.
- Click on the "Create Community" button.
- Fill in Community Details:
- Enter a name for the community in the "Community Name" field.
- Enter a description for the community in the "Description" field.
- Check the box Private or Public.
- Click the "Create Community" button.
2. **Verify Community Creation:**
- Verify that you receive a success message indicating that the community has been created.
- Verify that you are redirected to the new community's view page.
- Verify that the community name, description, and other details are displayed correctly.
3. **Additional Verification:**
- On the new community's view page, verify that you see an "Edit" button or “Settings” button indicating you have ownership.
- Check if you can upload an image to the community by clicking on the "Settings" button and using the image upload feature.
- Verify that you can edit community guidelines by clicking on the "Edit" button, making changes, and saving them.

Expected Results:

Step 1: You should be able to locate and click on the "Create Community" button.

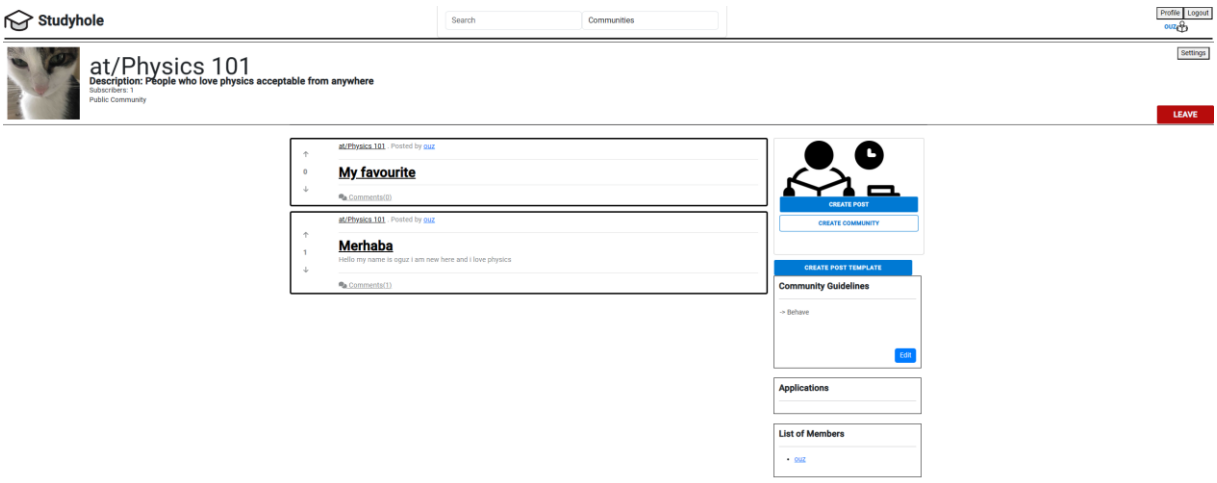
Step 2: You should be able to fill in the required community details and successfully create the community. You should receive a success message, be redirected to the new community's view page, and see all the entered details correctly displayed.

Step 3: You should have admin rights to the new community, be able to upload an image, and edit the community guidelines successfully.

RESULTS:

1- Successfully found create button and pressed.
2- Successfully filled required section with public community marked.
3- Successfully created community with ownership rights (can see edit and settings button)

Screen shot of the third result:



PROJECT REPOSITORY

Repository wiki-link will be directing you to a wiki-home page which has table of contents to all wiki content.

Repository Link: https://github.com/Oguzoz1/swe573-ozuer

Repository Wiki-Link: https://github.com/Oguzoz1/swe573-ozuer/wiki

Demonstration Link: https://www.youtube.com/watch?v=6KFqRHms8hI