SWE 573 – SOFTWARE DEVELOPMENT PRACTICE

Final Report

Ismail Enes Akkal

ID: 2022719006

Date: 19.05.2024

Github link ➔ https://github.com/EnesAkkal/SWE-573-Ismail-Enes-Akkal

Deployment link ➔ http://35.209.224.209:3000

Git tag version url ➔ https://github.com/EnesAkkal/SWE-573-Ismail-Enes-Akkal/releases/tag/v1.0.0

HONOR CODE

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

Ismail Enes Akkal

Contents

Project Details 3

Overview 3

Overview of implemented functionalities and Infrastructure of the Project 3

Implementation of the Project 4

Software Requirements Specification 5

Status of Deployment 7

User Manual 7

Test Results 10

User Tests 10

Unit Tests 12

Design Documents 13

Initial Database Design 13

Sequence Diagram 14

Use Case Diagram 15

Installation Instructions 16

Prerequisities 16

Frontend 16

Backend 16

Database 16

Docker 17

Login Credientials for Logging Into The System 17

Demo Video of The Application 17

Project Details

Overview

Overview of implemented functionalities and Infrastructure of the Project

1.1. Implemented Features

- Login / Register Feature for users
- Join Community Feature for users
- Create Community Feature for users
- User Information Feature for users
- Ability to create private communities
- Search community Feature for users
- Save customized community specific template feature for users
- Create a post within a community with default and customized templates.
- Ability to see community informations like name,description,owner and users.

1.2. **Infrastructure of the Project:** For the development of this project, I have chosen specific IDEs to meet to the needs of the frontend and backend development.

- **VS Code:** I've choosen Visual Studio Code for implementing the frontend. Its lightweight nature and the extensive library of extensions make it incredibly user-friendly for writing, debugging, and testing the frontend developments.

- **IntelliJ IDEA:** For the backend, I have choosen IntelliJ IDEA because I used to work with it before. Moreover, the ability to test API directly within IntelliJ IDE was another reason to choose this IDE for me. IntelliJ Idea's HTTP client has allowed me to quickly send requests and validate the responses from the backend.

- **React:** I decided to choose React as the frontend language because I used React before for frontend development.

- **Java Spring Boot:** I decided to use Java Spring Boot as the backend language. Although I do not have prior experience with Java Spring Boot, I used to work with java before so I am more familiar to Java than Python that's why I decided Java Spring Boot. I thought having no prior experience with Python would be hard for me If I had worked with Django.

- **MongoDB:** For the data storage needs of this project, I have selected MongoDB as the database.There were many practise videos over the internet for Spring Boot and MongoDB so I decided to go for MongoDB.

Implementation of the Project

- My project consists of 2 main folders. One of them is called “React” for the frontend files and “App” fo the backend files. As far as the backend implementation concerned, I decided to use Repository Pattern because I wanted to divide the responsibilities of the project as wide as possible so that any modification in one area would not affect others. I’d like to briefly mention the advantages of repository pattern. One of the advantages of using repository pattern is that it creates a abstraction layer over data access layer with the help of repositories so that we can perform CRUD operations through this layer instead of directly doing CRUD operations with data access layer. Another reason why I used repository pattern is that Java Spring Boot can automatically implement the repository interface with creating the CRUD operations behind the scenes.

  For the Backend Implementation, my structure is as the following;

```
.
├── config
|   └── WebConfig.java
├── controllers
|   ├── AuthController.java
|   ├── CommunityController.java
|   └── UserController.java
├── dtos
|   ├── CreateCommunityDto.java
|   ├── CredentialsDto.java
|   ├── ErrorDto.java
|   ├── SignUpDto.java
|   └── UserDto.java
├── exceptions
|   └── AppException.java
├── models
|   ├── community
|   |   └── Community.java
|   ├── post
|   |   ├── Comment.java
|   |   ├── Post.java
|   |   └── Template.java
|   └── user
|       └── User.java
├── repositories
|   ├── CommunityRepository.java
|   ├── PostRepository.java
|   └── UserRepository.java
└── services
    ├── CommentService.java
    ├── CommunityService.java
    ├── LoginService.java
    ├── PostService.java
    └── TemplateService.java
|   └── UserService.java
```

Controllers : Controllers are basically responsible for handing incoming and outgoing http requests between server and client. Right now, I have 3 controllers which are AuthController, CommunityController and UserController.

- AuthController is where my endpoints for register&login resides in. They basically handle login/register http requests that comes from the client.
- CommunityController is where my endpoints for join/create community resides in. They handle http requests for joining or creating new communities.
- UserController is not finished yet but it will be used for updating user information.

DTO's: DTO's are used to transfer the data between the layers of the application. I did not want to expose my entity tables to the users so I used dto's to encapsulate the data that I want to expose to the user. Morevoer, different functionalities within the application might lead to distinct API endpoints, each possibly requiring a unique set of data fields. This variability could lead to frequent changes in the underlying database entities if I were to use them directly for data transfer purpose.

Models: Models are basically representing my database schema for tables and it's attributes.I have 3 different folder under model folder and those are ; community,post and user. I don't have any other class under community and user.However, I have 3 different model under post which are; Comment,Post and Template.

Repositories: Repositories are basically responsible for CRUD operations for my database. There are 3 repository folders called; community,user and post. Apart from inheriting several methods for CRUD operations from Spring Data MongoDB, there are also other custom query methods for each repository. Respositories also helps to create an abstraction layer.

Services: Services act like a bridge between my controllers and repositories. It contains the bussiness logic of the application by carrying out operations. For instance, loginservice class is responsible for authentication of users. I have 6 different services right now which are "Comment Service Community Service,Login Service, Post Service, Template Service, UserService."

Software Requirements Specification

This area outlines the system requirements for a Community App designed to facilitate interaction, content sharing, and community management among users. The platform aims to provide a comprehensive environment where users can engage in discussions, share information, and collaborate within various communities based on their interests and needs. The functional requirements specified in this document are categorized into three main sections: Registration &

Authentication, Community Creation & Moderation, and User Profile & Interaction. Each category encompasses specific functionalities designed to improve the documentation quality of this app.

1. Account Management

Registration & Authentication

- Users shall be able to create an account with a unique username and password. ➔ Completed
- Users shall be able to delete their accounts. ➔ Not Completed
- Users shall be able to login to the website with their unique username and password. ➔ Completed

2. Community Management

Community Creation & Settings

- Users shall be able to create new communities with a unique name. ➔ Completed
- Community owners shall be able to assign users as moderators within their community. ➔ Not Completed
- Community ownership shall be transferred by the creator to other members of the community if the creator wants to leave the community. ➔ Not Completed
- The system shall allow community owners to set their community status as public or private. ➔ Completed

Moderation & Privacy

- Community owners and moderators shall be able to kick users from the community. ➔ Not Completed
- Moderator or community owners shall handle reports sent by the users and take proper action. ➔ Not Completed
- The system shall ensure that content and members within private communities are visible only within those specific private communities. ➔ Not Completed
- The system shall allow joining to a private community only through invitations sent by the community owner or moderator. ➔ Not Completed
- Users shall have the ability to report inappropriate content or behavior for review by the moderator and community owner. ➔ Not Completed

3. User Interaction & Profiles

User Profiles

- Users shall have profiles that can include information such as personal details, photos, and so on. ➔ Partially Completed
- Users shall see the total number of communities, posts, and members. ➔ Completed

Social Features & Notifications

- Users shall be able to mute another user. ➔ Not Completed
- Users shall be able to accept or decline invitations sent by the moderator. ➔ Not Completed
- Users shall receive notifications of new activities such as; invitation request, like, comment, and so on. ➔ Not Completed
- Users shall have the ability to like or comment on a post. ➔ Not Completed
- Users shall be able to comment on existing posts. ➔ Not Completed
- The system shall display a user's status to other members. ➔ Not Completed

Community Interaction & Discovery

- Users shall be able to join multiple communities. ➔ Completed
- Users shall see the list of most active communities. ➔ Completed
- Users shall see the list of most active members. ➔ Not Completed
- Users shall be able to view the list of recently active communities.
- The system shall provide search functionality to allow users to find posts and communities. ➔ Partially Completed
- Users shall be able to search posts and communities by name and tags. ➔ Partially Completed

Content Creation & Management

- Users shall have the capability to create posts, each with a title, associated tag and an image. ➔ Completed.
- Users shall be able to attach images or files upon preference when creating a post. ➔ Not Completed
- Users shall be able to edit or delete their own posts and comments. ➔ Not Completed
- The system shall denote mandatory fields with a (*) and highlight them in red if they are left empty. ➔ Completed

Status of Deployment

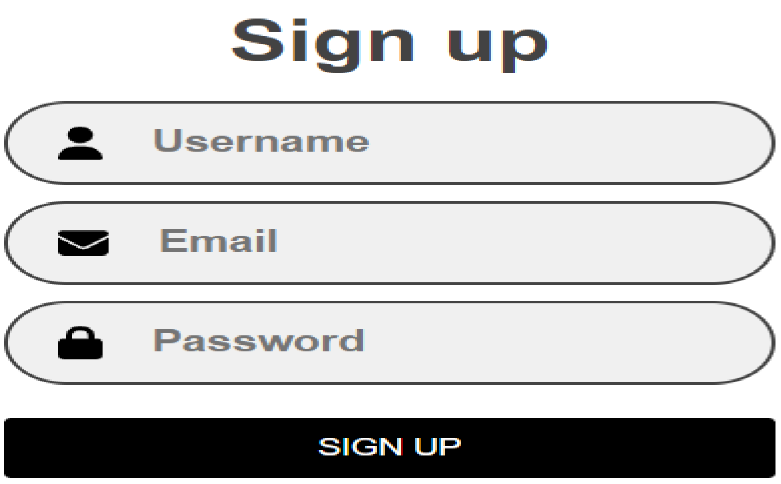
The application has been dockerized and deployed.

http://35.209.224.209:3000

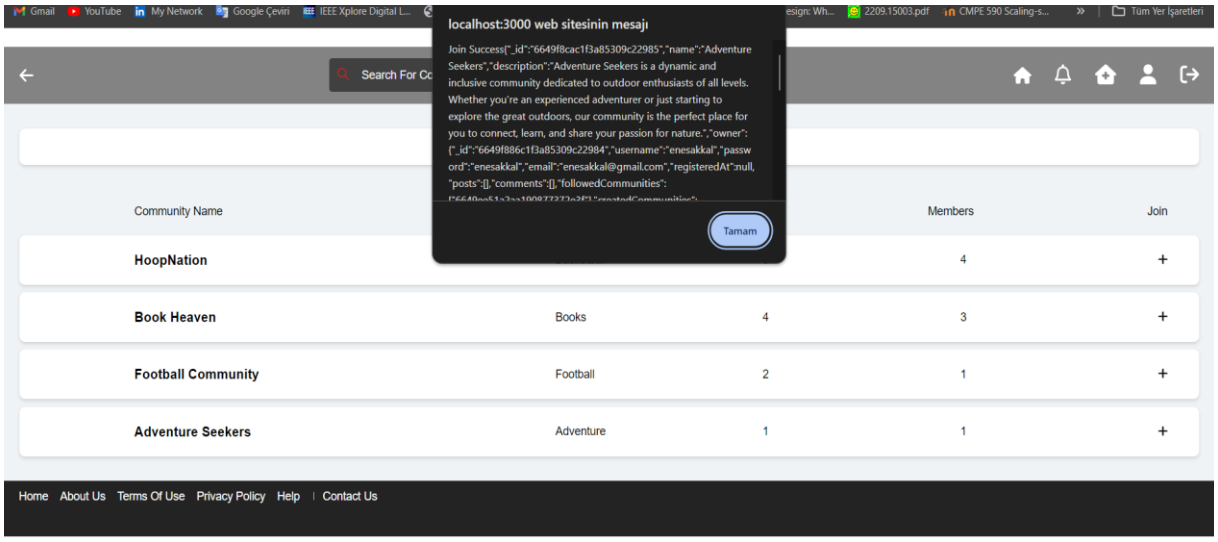
User Manual

Step 1: Run Docker and go to http://localhost:3000/login.

Step 2: Login to the system with given credientals or register to the system to determine your credientals.



Step 3: You can join a community to create a post.In this ss below, I've joined "Adventure Seekers " Community.

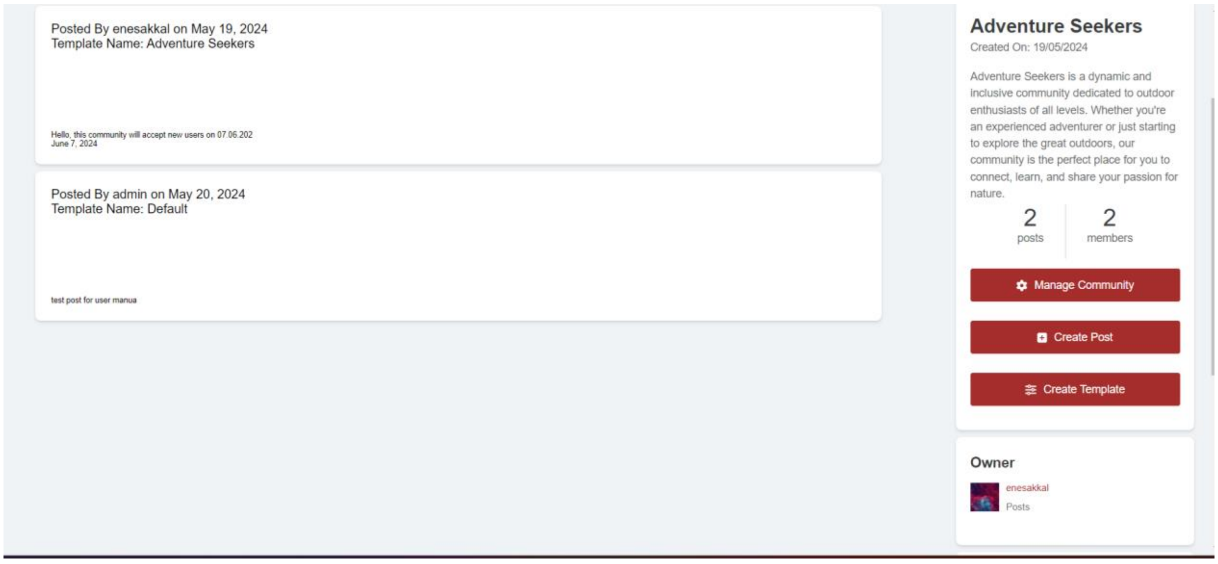


localhost:3000 web sitesinin mesajı

```
Join Success{"_id":"6649f8cac1f3a85309c22985","name":"Adventure
Seekers","description":"Adventure Seekers is a dynamic and
inclusive community dedicated to outdoor enthusiasts of all levels.
Whether you're an experienced adventurer or just starting to
explore the great outdoors, our community is the perfect place for
you to connect, learn, and share your passion for nature.","owner":
{"_id":"6649f886c1f3a85309c22984","username":"enesakkal","passw
ord":"enesakkal","email":"enesakkal@gmail.com","registeredAt":null,
"posts":[],"comments":[],"followedCommunities":
```

Community Name			Members	Join
HoopNation			4	+
Book Heaven	Books	4	3	+
Football Community	Football	2	1	+
Adventure Seekers	Adventure	1	1	+

Step 4: You can create a post by clicking on "Create Post"



Posted By enesakkal on May 19, 2024
Template Name: Adventure Seekers

Hello, this community will accept new users on 07.06.202
June 7, 2024

Posted By admin on May 20, 2024
Template Name: Default

test post for user manua

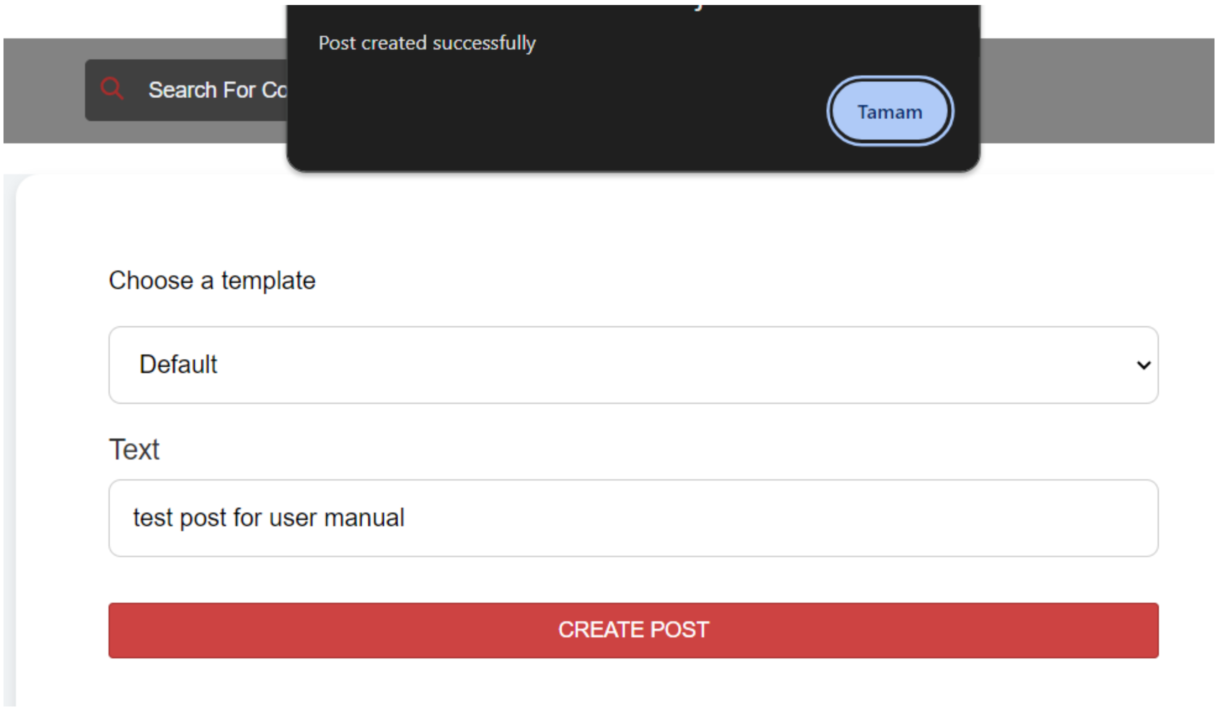
Adventure Seekers
Created On: 19/05/2024

Adventure Seekers is a dynamic and inclusive community dedicated to outdoor enthusiasts of all levels. Whether you're an experienced adventurer or just starting to explore the great outdoors, our community is the perfect place for you to connect, learn, and share your passion for nature.

2 posts | 2 members

Owner
enesakkal
Posts

Step 5: Choose a template and click on "Create Post".

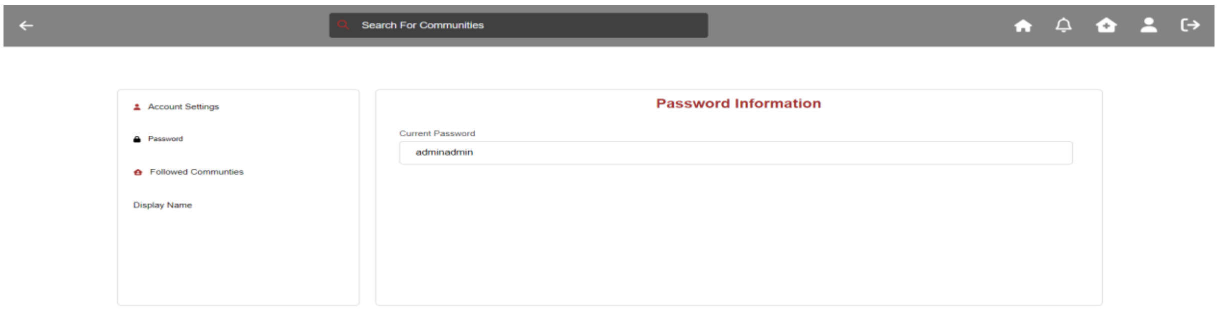


Post created successfully

Choose a template
Default

Text
test post for user manual

Step 6: Click on User button to see your account details



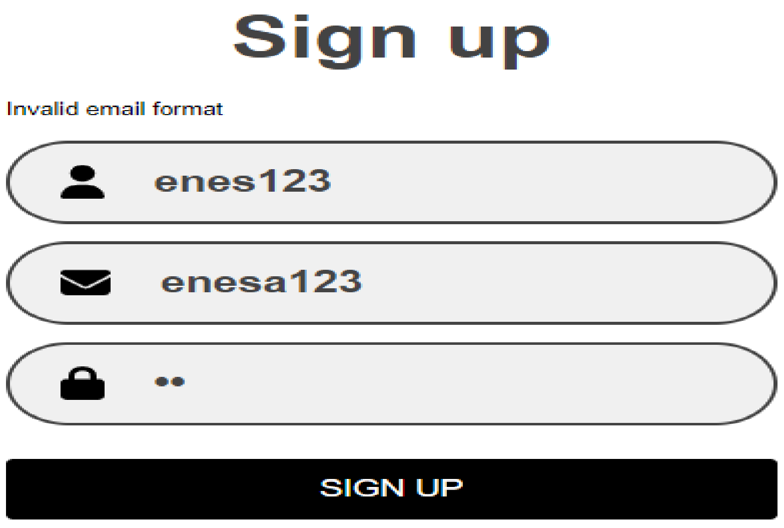
Password Information

Current Password
adminadmin

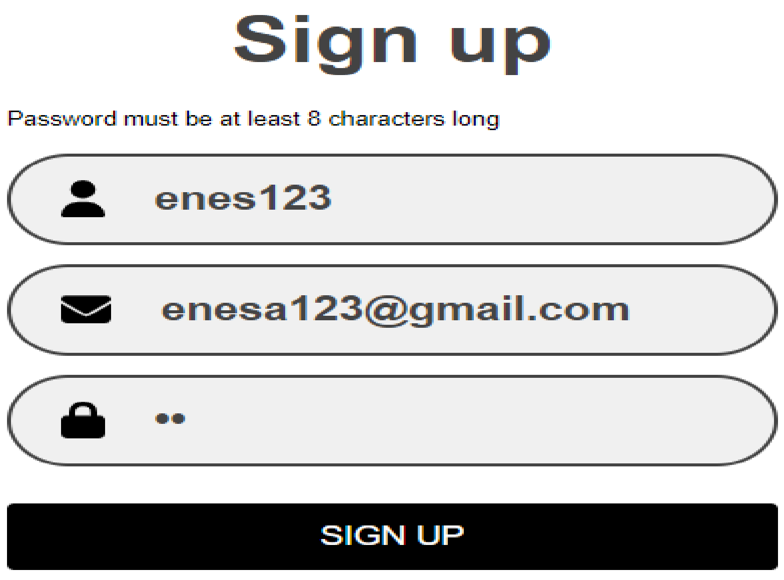
Test Results

User Tests

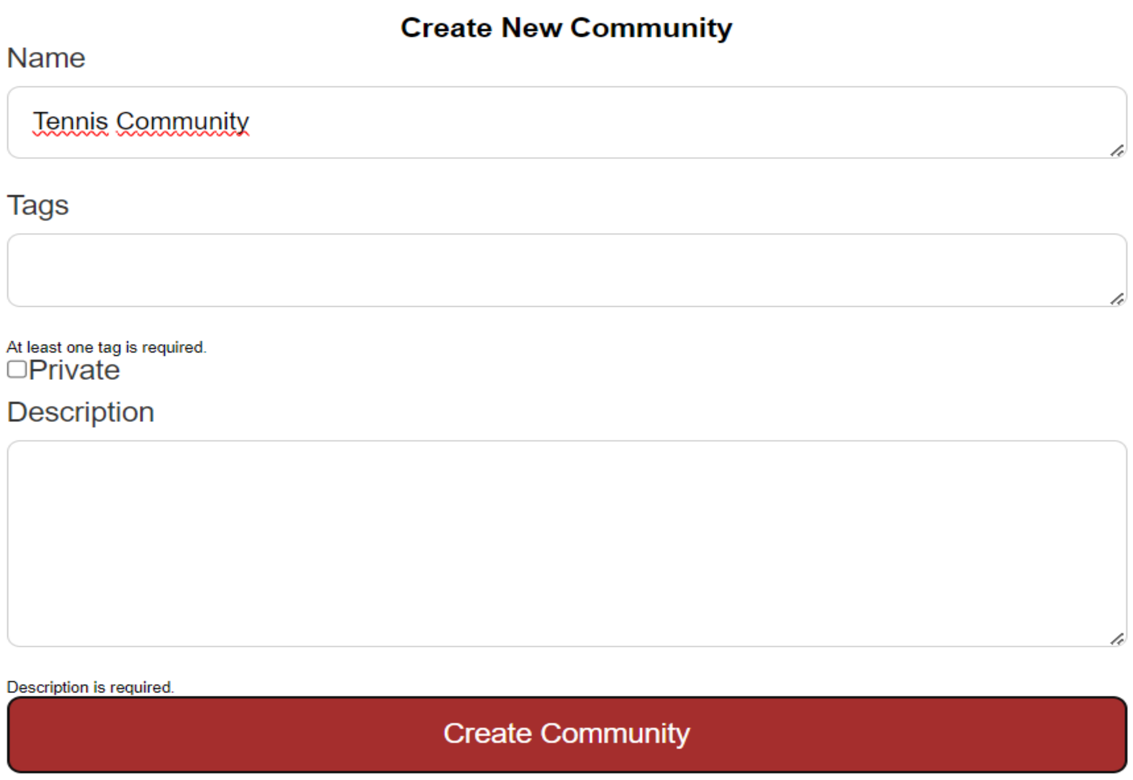
- Scenario 1: User tries to register to the system with a not valid email



- Scenario 2: User tries to fill the password area while registering with less than 8 character.



- Scenario 3: User tries to create a community without providing necessary fields like tags and description.



Create New Community

Name

Tennis Community

Tags

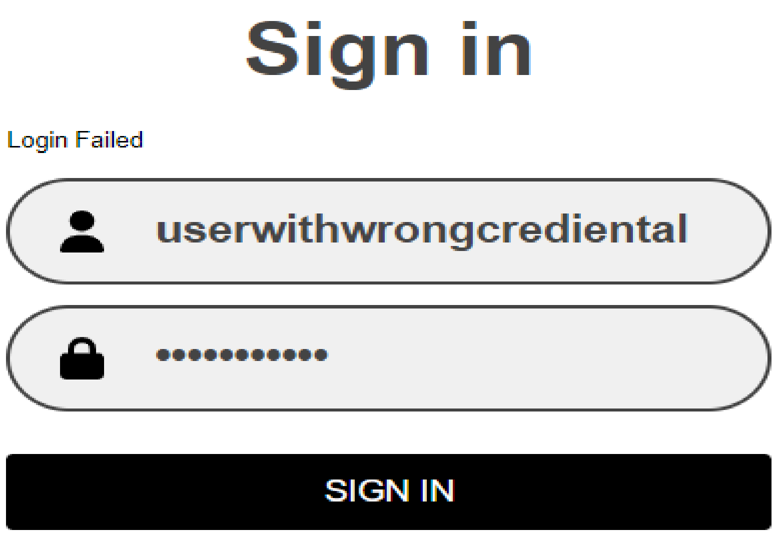
At least one tag is required.

☐ Private

Description

Description is required.

- Scenario 4: User tries to login with wrong credientals



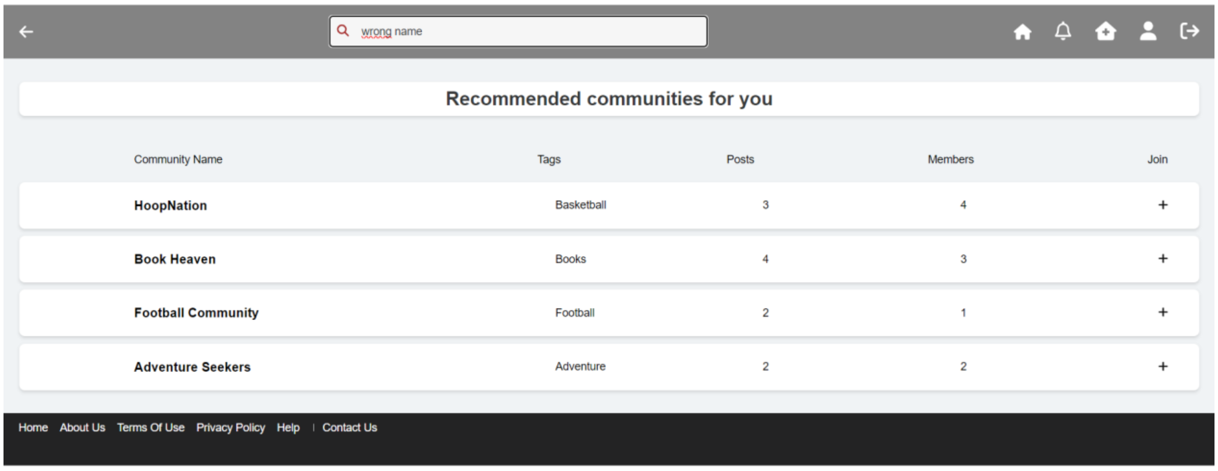
Sign in

Login Failed

userwithwrongcrediental

●●●●●●●●●●●

- Scenario 5: User tries to find a non-exist community through search bar.



wrong name

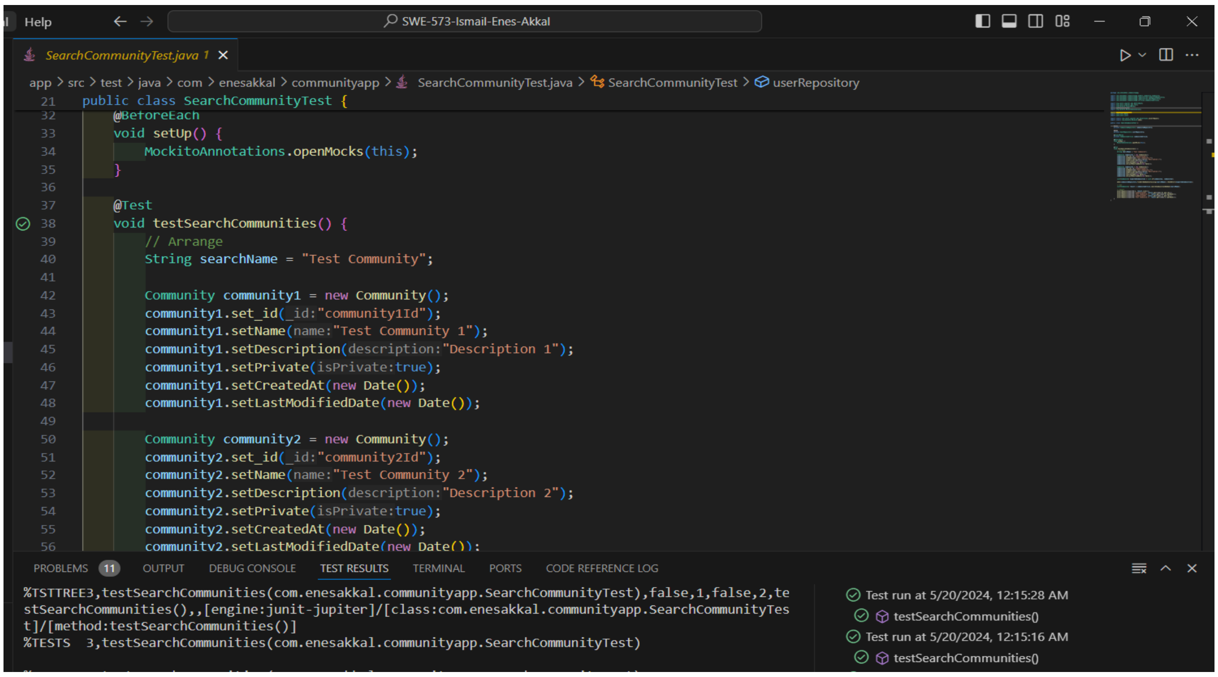
Recommended communities for you

Community Name	Tags	Posts	Members	Join
HoopNation	Basketball	3	4	+
Book Heaven	Books	4	3	+
Football Community	Football	2	1	+
Adventure Seekers	Adventure	2	2	+

Home About Us Terms Of Use Privacy Policy Help | Contact Us

Unit Tests

- Unit test for searching Community



SWE-573-Ismail-Enes-Akkal

SearchCommunityTest.java 1

app > src > test > java > com > enesakkal > communityapp > SearchCommunityTest.java > SearchCommunityTest > userRepository

```java
public class SearchCommunityTest {
    @BeforeEach
    void setUp() {
        MockitoAnnotations.openMocks(this);
    }

    @Test
    void testSearchCommunities() {
        // Arrange
        String searchName = "Test Community";

        Community community1 = new Community();
        community1.set_id(_id:"community1Id");
        community1.setName(name:"Test Community 1");
        community1.setDescription(description:"Description 1");
        community1.setPrivate(isPrivate:true);
        community1.setCreatedAt(new Date());
        community1.setLastModifiedDate(new Date());

        Community community2 = new Community();
        community2.set_id(_id:"community2Id");
        community2.setName(name:"Test Community 2");
        community2.setDescription(description:"Description 2");
        community2.setPrivate(isPrivate:true);
        community2.setCreatedAt(new Date());
        community2.setLastModifiedDate(new Date());
```

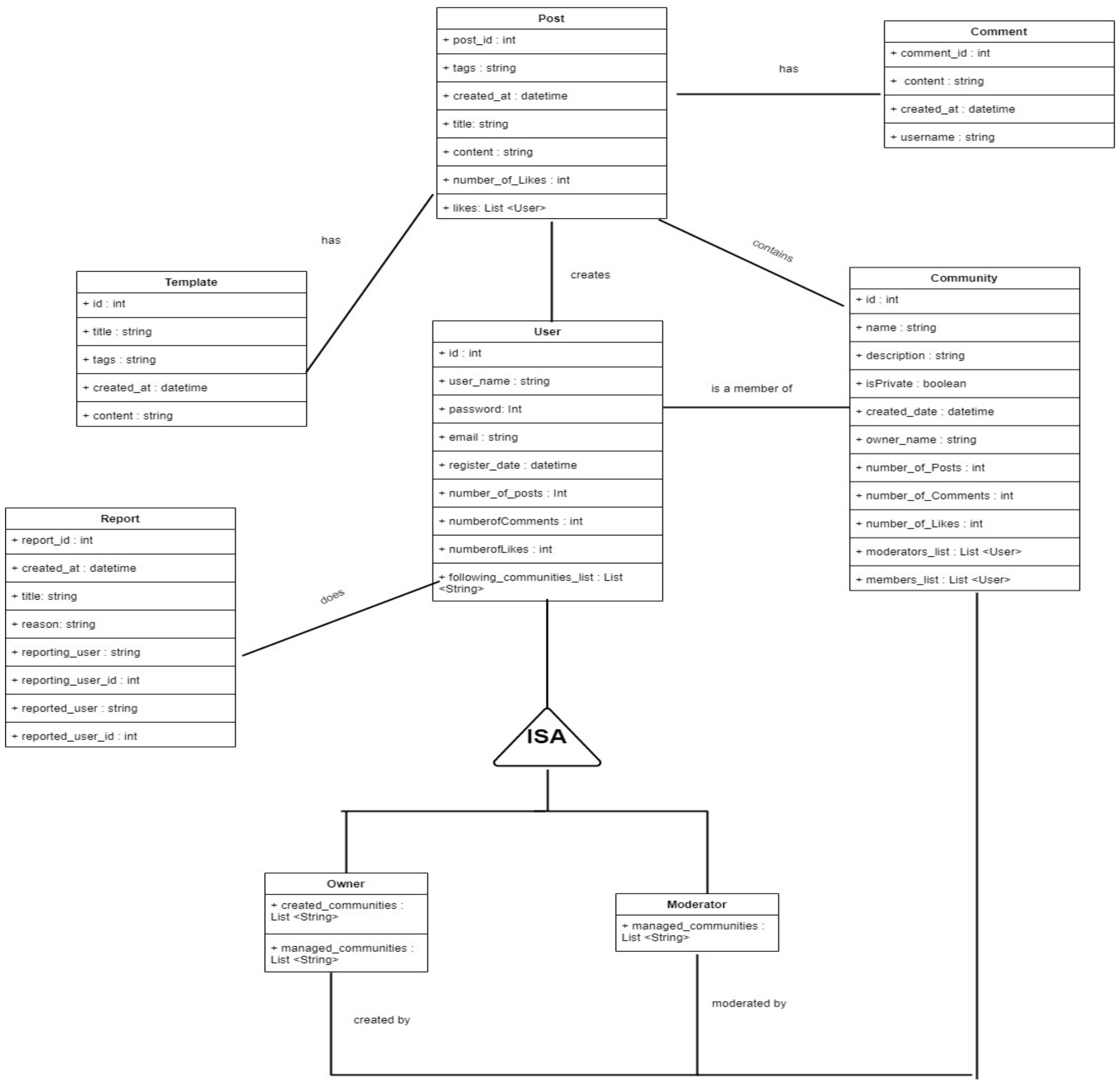
PROBLEMS 11 OUTPUT DEBUG CONSOLE TEST RESULTS TERMINAL PORTS CODE REFERENCE LOG

```
%TSTTREE3,testSearchCommunities(com.enesakkal.communityapp.SearchCommunityTest),false,1,false,2,te
stSearchCommunities(),,[engine:junit-jupiter]/[class:com.enesakkal.communityapp.SearchCommunityTes
t]/[method:testSearchCommunities()]
%TESTS  3,testSearchCommunities(com.enesakkal.communityapp.SearchCommunityTest)
```

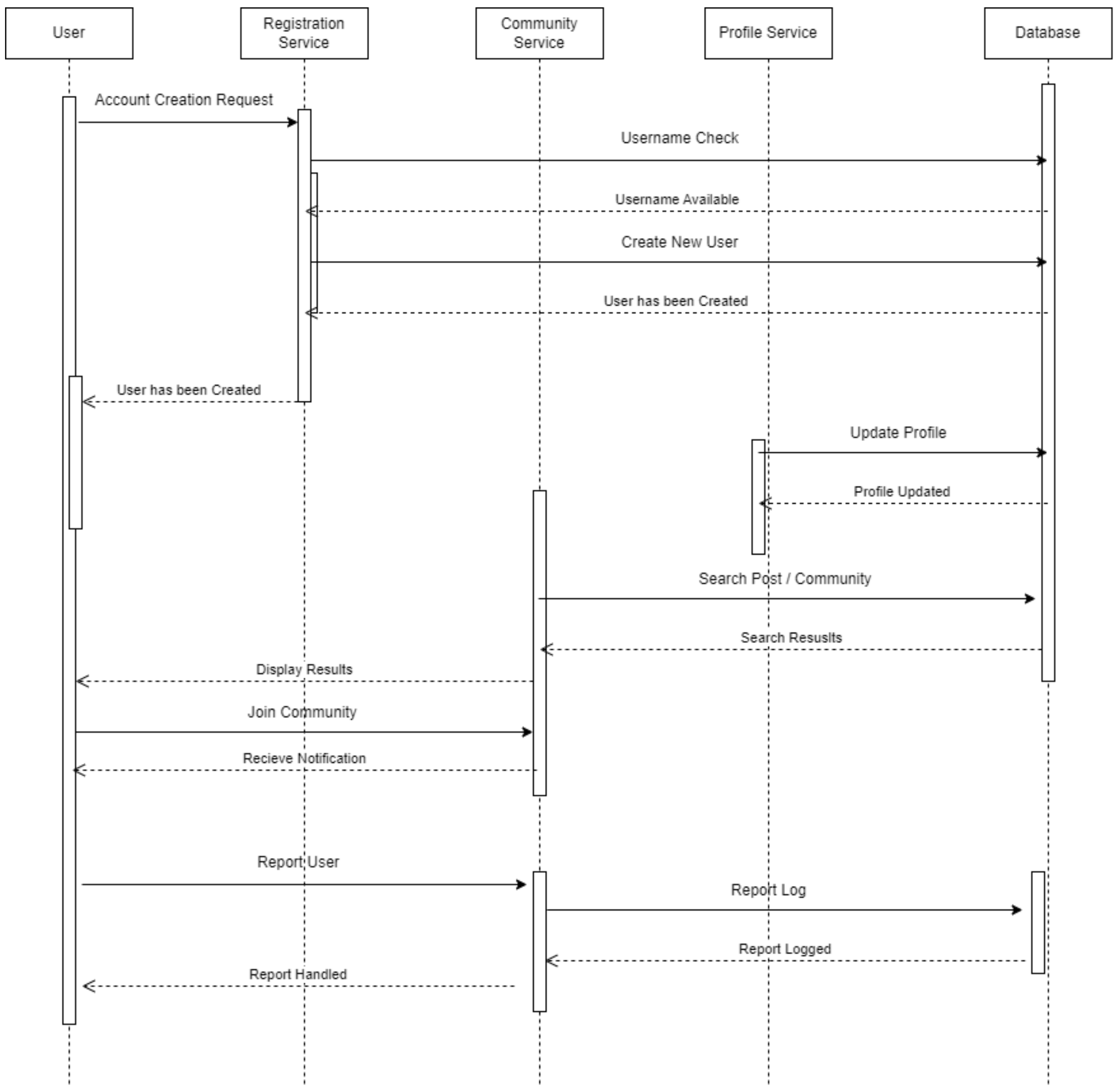
```
✓ Test run at 5/20/2024, 12:15:28 AM
  ✓ testSearchCommunities()
✓ Test run at 5/20/2024, 12:15:16 AM
  ✓ testSearchCommunities()
```

Design Documents

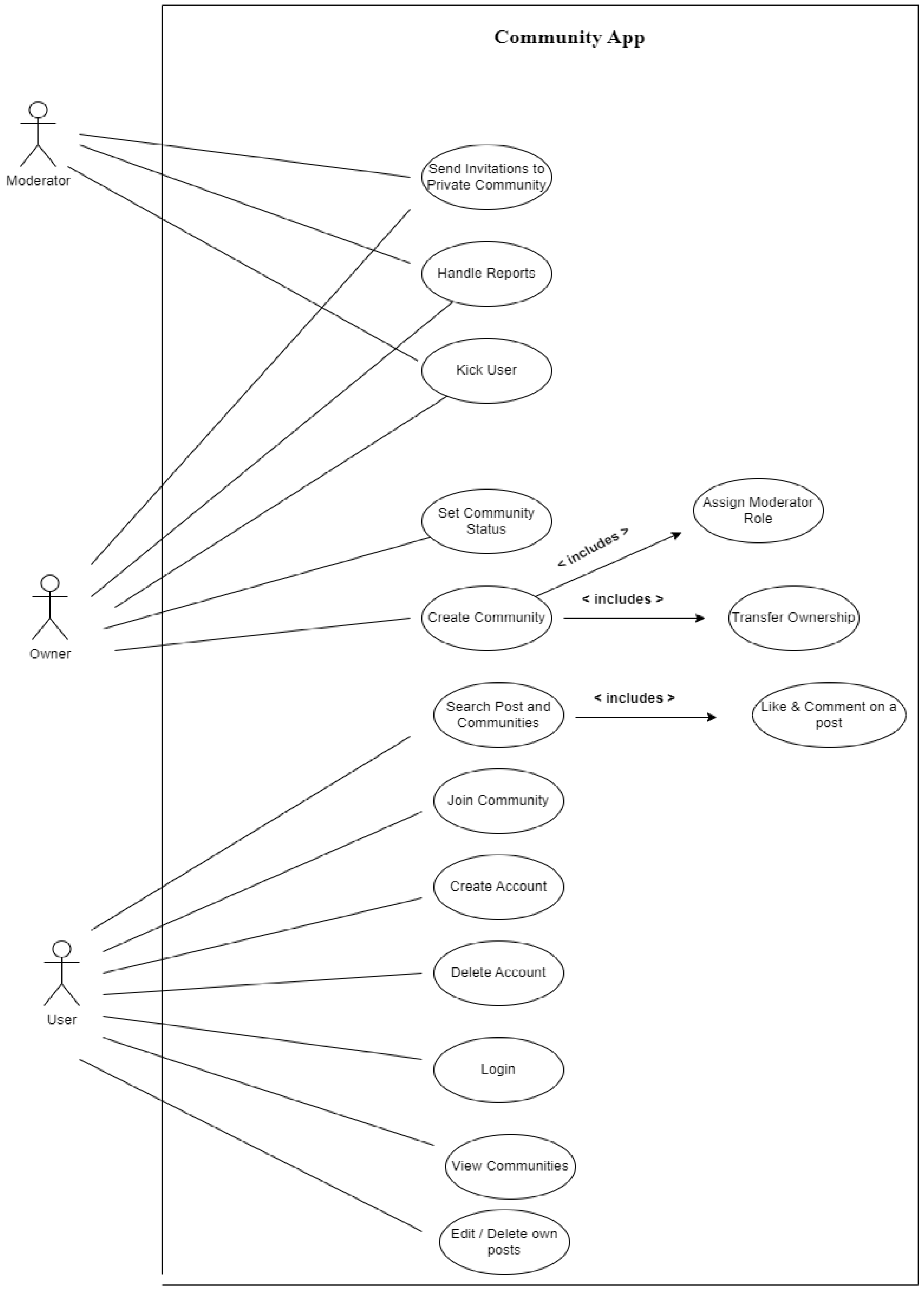
Initial Database Design



Sequence Diagram



Use Case Diagram



Installation Instructions

Prerequisities

1. Java 17

2. Node.js and Npm

3. IntellijIDEA or any preffered UDE

4. MongoDB Compass

5. Docker Desktop

Frontend

1. You need to download node.js if you dont have one by clicking the link below.

2. If you dont have node.js first download node.js from

https://nodejs.org/en/download/current

3. cd React

4. run npm i

5. npm start (for every start) in react folder.

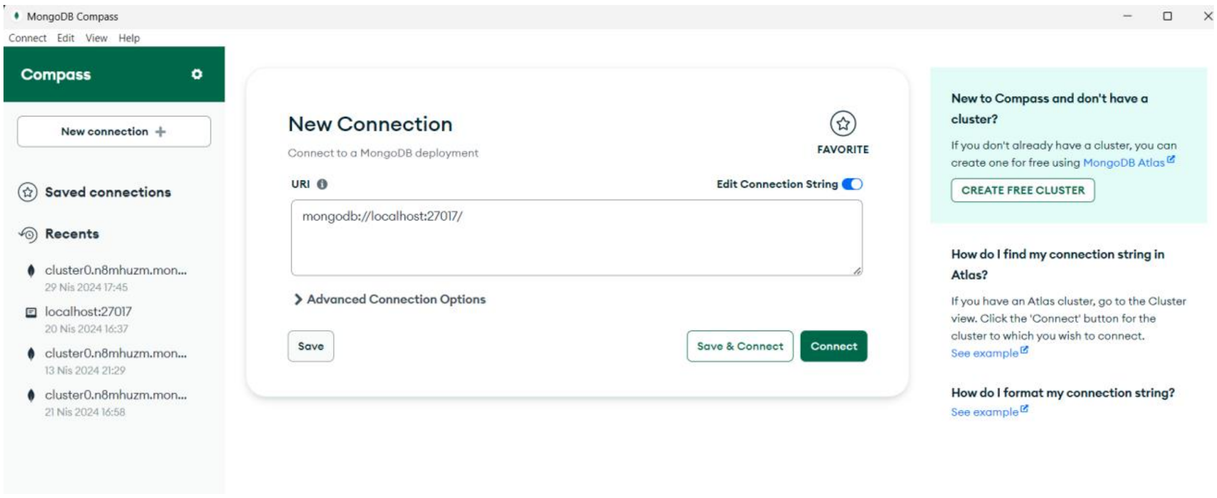
Backend

1. cd to app folder

2. Run ./gradlew bootRun

Database

1. You should create an account on MongoDB Atlas if you don’t have one
   ⇨ https://account.mongodb.com/account/login?nds=true&_ga=2.3314995.269306997.17144014871639680084.1713906131&_gac=1.116928884.1714401978.Cj0KCQjwir2xBhC_ARIsAMTXk86YJNzEoSEkastryJqKNIRZ6c7_FPg8kZ4TVscmxz2XhXBy-wjfmt4aArtXEALw_wcB
2. The github repo already contains the MongoDB_URI with credentials under application.yaml file so cloning the repo will be enough to access the database.
3. Download mongoDB Compass from here (https://www.mongodb.com/try/download/compass) and click on “New Connection”.



mongodb://localhost:27017/

New to Compass and don't have a cluster?
If you don't already have a cluster, you can create one for free using MongoDB Atlas

How do I find my connection string in Atlas?
If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.

How do I format my connection string?

4. uri:mongodb+srv://enesakkal:121416f9@cluster0.n8mhuzm.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0
   Use the above URI to connect the database.

5. Username: enesakkal
   Password:121416f9

Docker

- You need to download docker from https://www.docker.com/products/docker-desktop/
- cd to App folder
- Type "./gradlew build"
- Cd back
- Type "docker-compose up" in the main foler.

Login Credentials for Logging Into The System

You can either login with the following credentials or create a new user.

Username: testuser22

Password: testuser22

Demo Video of The Application

Demo video of the Application can be reached below with the following link

https://youtu.be/4w74e6aAHFs