SWE 573 – Software Development Practice
Final Report – Community Information System Application

Cankut ER
ID: 2022719219
Date: 20.05.2024

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

3rd Party Software Usage:
During the project development, only open source and free libraries have been used. These are declared in more detail in the dependency lists, namely pom.xml for the API and package.json for the front end. These files can be found in the repository under directories “Communitter” and “communitter-frontend”.

GitHub: https://github.com/CankutER/SWE-573
Release URL: https://github.com/CankutER/SWE-573/releases/tag/v0.9
Deployment URL: http://ec2-3-127-222-202.eu-central-1.compute.amazonaws.com:3000
Demo Video URL: https://cloud-project-cankut.s3.eu-central-1.amazonaws.com/SWE-573-Final-Recording.mov

Table of Contents

1 Status of Build and Deployment

Project has been dockerized and deployed to an t2.medium EC2 instance provided by AWS.
All components of the project, including the database, is running in separate containers and can be ramped up via “docker compose up” command.

Building Locally:

Docker compose file has been prepared to run the application and the database in containers, thus docker is required. No further configuration is required.

Steps:

cd <repository path>
cd Communitter
docker compose up (“docker-compose up” for older docker versions)

Please note that ramping up will create “pgdata” directory under Communiter, this is for persistent volume binding and can be disregarded. When the application is ramped down (docker compose down) or process is stopped, application automatically removes all data created in the database.

Application will generate the static sql data at the startup. This is intended for data integrity and only entities that will stay the same during the application lifecycle are created. No user, community, template or post data is created initially. In order to test the application, a swagger interface, link of which is given above, is provided and example mock data to call these endpoints are provided in the report’s next section. This approach is intentionally adopted to allow users to test the application from scratch and create their own data for testing.

2 Overview

Project that is topic to this report is a Community Information Management System application. This application aims to provide a self-regulated environment to share information and data leveraging pre-defined structures defining how the data is shared. While still maintaining the flexibility and freedom, project proposes a solution to avoid data pollution in traditional social network applications.

Purpose of the project is not only developing code and implementing features but also put emphasize on the design, documentation and project management activities mainly utilizing

the mainstream version control systems. During the project lifecycle, wiki pages have been created for documentation and design in addition to the regular “Issues” created to keep track of the progress and remark requirements for any given task defined in “Issues”.

3 Software Requirement Specification

3.1 Glossary

Guest: Visiter that have not been authenticated.
User: Default role for a visiter that have been authenticated.
Creator: Role assigned for a community to a User who created that community.
Owner: Role assigned by Creator to a User within a community.
Moderator: Role assigned by Creator or Owner to a User within a community.
Post Template/Schema: Prescribed structure for data composed of data fields. Describes the format for information to be shared in a community.

3.2 Functional Requirements

3.2.1 Community Management

- F1: The system shall allow users to create communities by defining community name, post templates and description.
- F2: The system shall provide builders the mechanism to create information schemas for posts so that the builders can describe the type of information shared in their communities.
- F3: The system shall allow builders to create as many schemas as they need.
- F4: The system shall provide the ability to add data fields to schemas with the option of marking them as required or option, where the key is a chosen title and the value is the data type so that the builders can define the schemas as they please.
- F5: The system shall provide predefined data types for each data field value; such as geolocation, date, text, image, audio, video.
- F6: The system shall have no limit for the amount of data fields defined in a schema.
- F7: The system shall have mechanisms to create, update and delete schemas after a community has been created.
- F8: The system shall enforce users to choose a schema specifically created for the community they interact with before making a post.

3.2.2 Authorization, Roles and Permission

- F9: The system shall have "Owner", "Creator", "Moderator" and "User" roles, where "User" is a global default role and overwritten by the other roles inside a community as long as any other role than "User" is given.

- F10: The system shall assign only one "Creator" to each community, which shall be the user who initially creates the community.
- F11: The system shall not allow changing the "Creator" of a community after creation.
- F12: The system shall allow the "Creator" to promote and demote "User"s to "Owner" and from "Owner".
- F13: The system shall not allow an "Owner" to kick or demote another "Owner".
- F14: The system shall allow the "Creator" and an "Owner" to promote/demote a "User" to/from "Moderator".
- F15: The system shall allow a "Moderator" to kick a "User".
- F16: The system shall allow a "Moderator" to edit/delete a post of another "User".
- F17: The system shall allow a "User" to report a post.
- F18: The system shall allow a "User" to report a "Moderator".

3.2.3 Identity Management

- F19: The system shall provide a profile page for each "User".
- F20: The system shall allow a "User" to have an avatar, description and a username.
- F21: The system shall allow a "User" to create/update/delete all identity information(avatar, description, username, email, password etc.) on the profile page.
- F22: The system shall allow a "User" to see all communities subscribed and roles assigned in those communities as a list on the profile page.
- F23: The system shall allow a "User" to manage community subscriptions (leave community, demote self etc.) on the profile page.

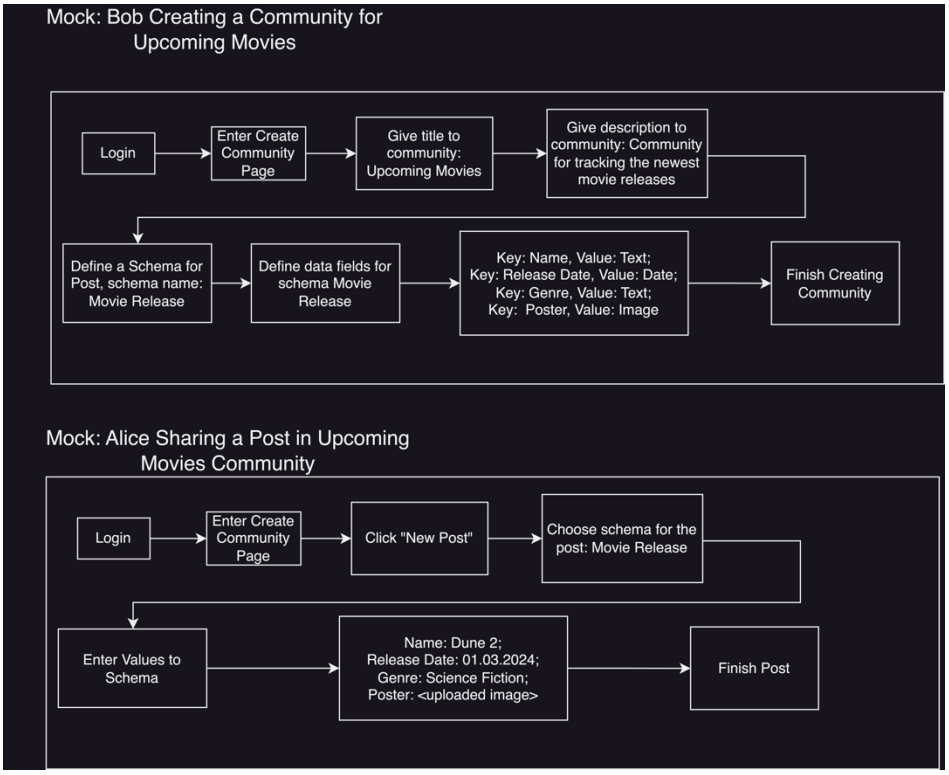
3.2.4 Security and Interaction

- F24: The system shall not allow direct messaging between "User"s.
- F25: The system shall allow a "User" to view other "User"s' avatar, description and community subscriptions on their profile page.
- F26: The system shall allow a "User" to like or upvote a post.
- F27: The system shall allow a "User" to comment on a post.
- F28: The system shall provide an authentication mechanism.
- F29: The system shall give most liked or upvoted posts priority in terms of order of display in a community's main page.
- F30: The system shall allow a "Creator" and an "Owner" to put tags or labels on the community they are responsible for.
- F31: The system shall be able to recommend communities to a "User" based on the tags of the communities the "User" is already subscribed.
- F32: The system shall allow for to make communities private.
- F33: The system shall only allow "Creator" to make a community private, either at creation or afterwards.

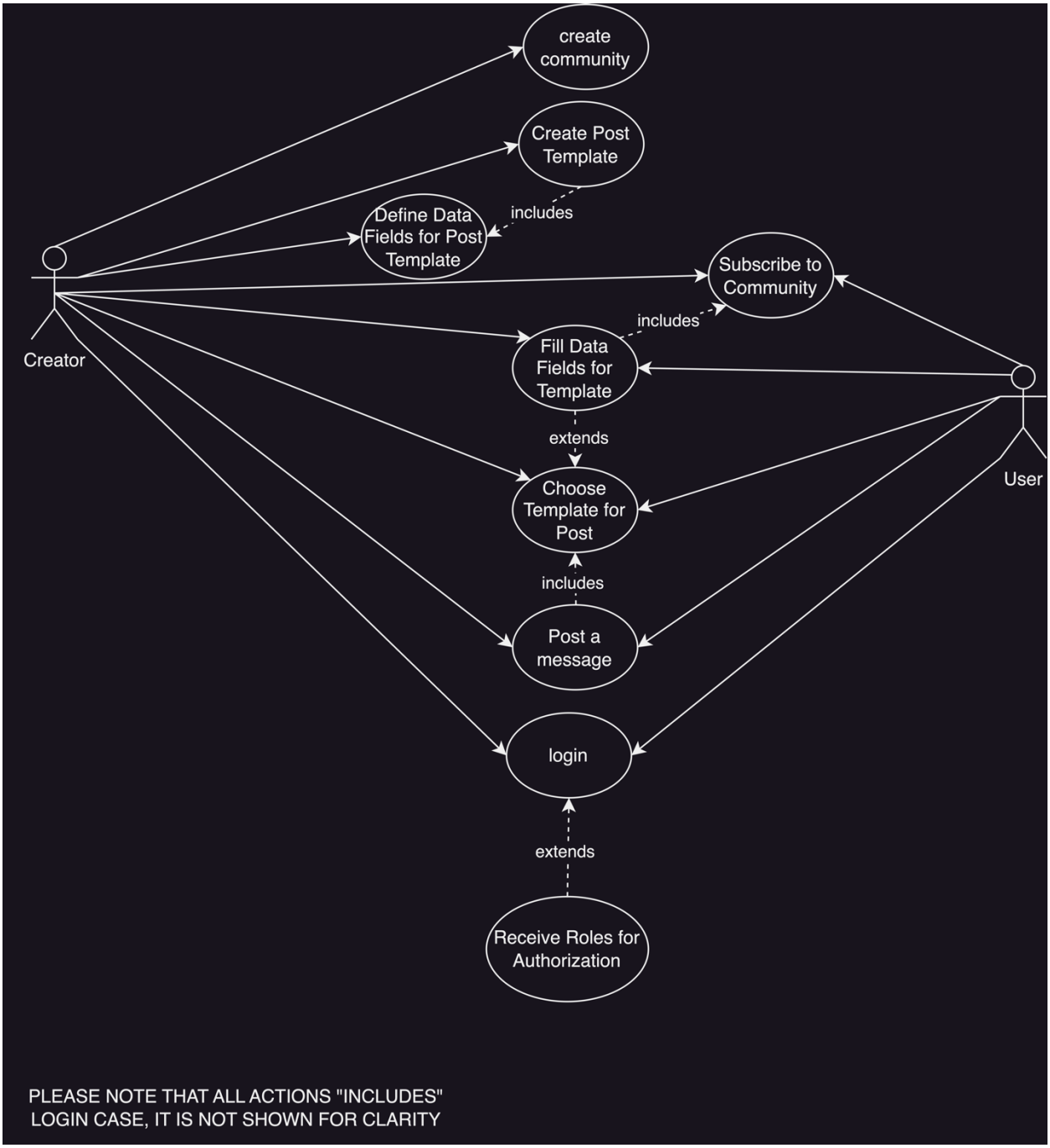
- F34: The system shall allow recruitment of a new "User" to a private community only via invitation.
- F35: The system shall allow any "User" of a private community to send invitation to another "User".
- F36: The system shall provide search mechanisms for finding communities and posts in communities.

4 Design Documents

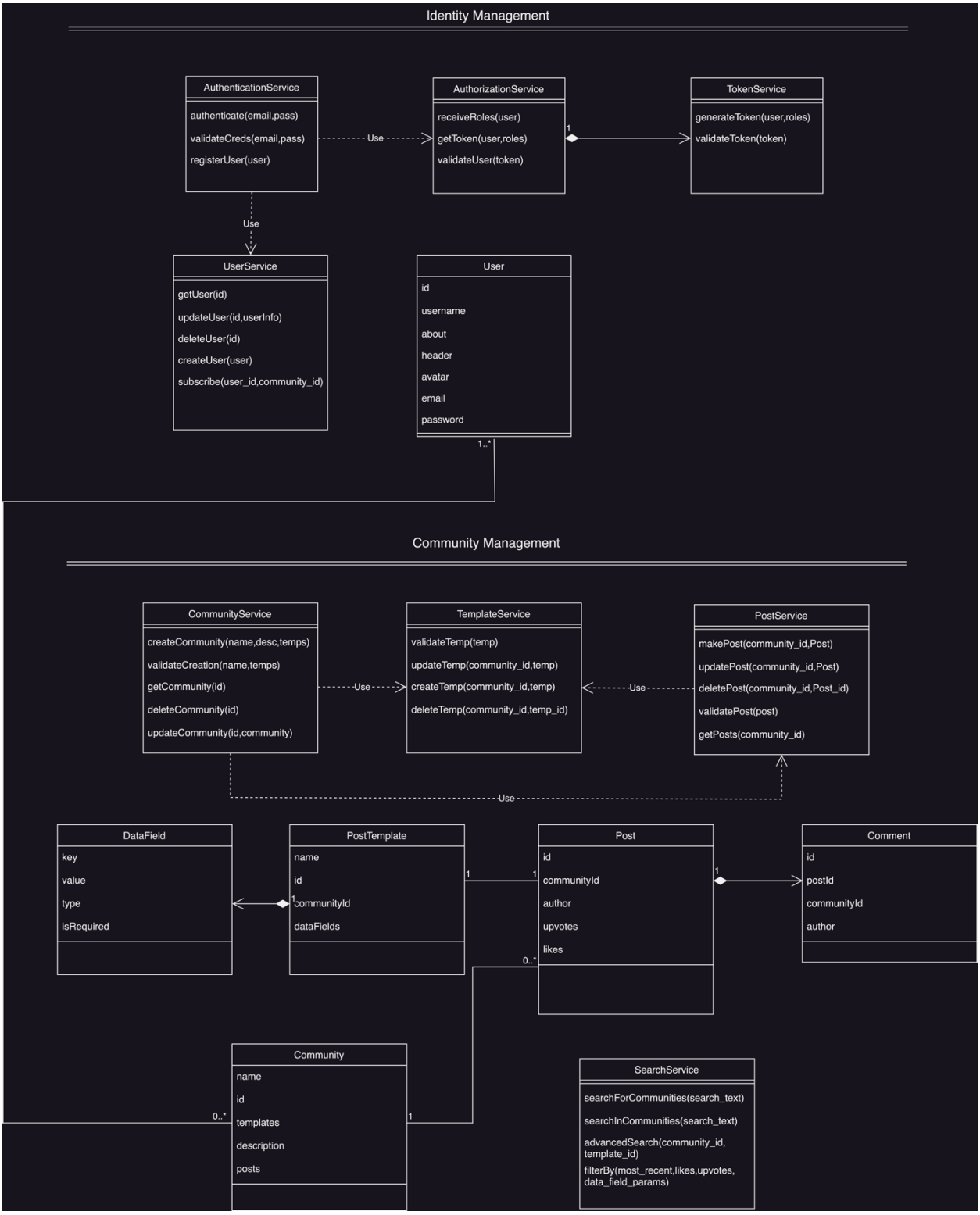
Mocks for Use Cases:



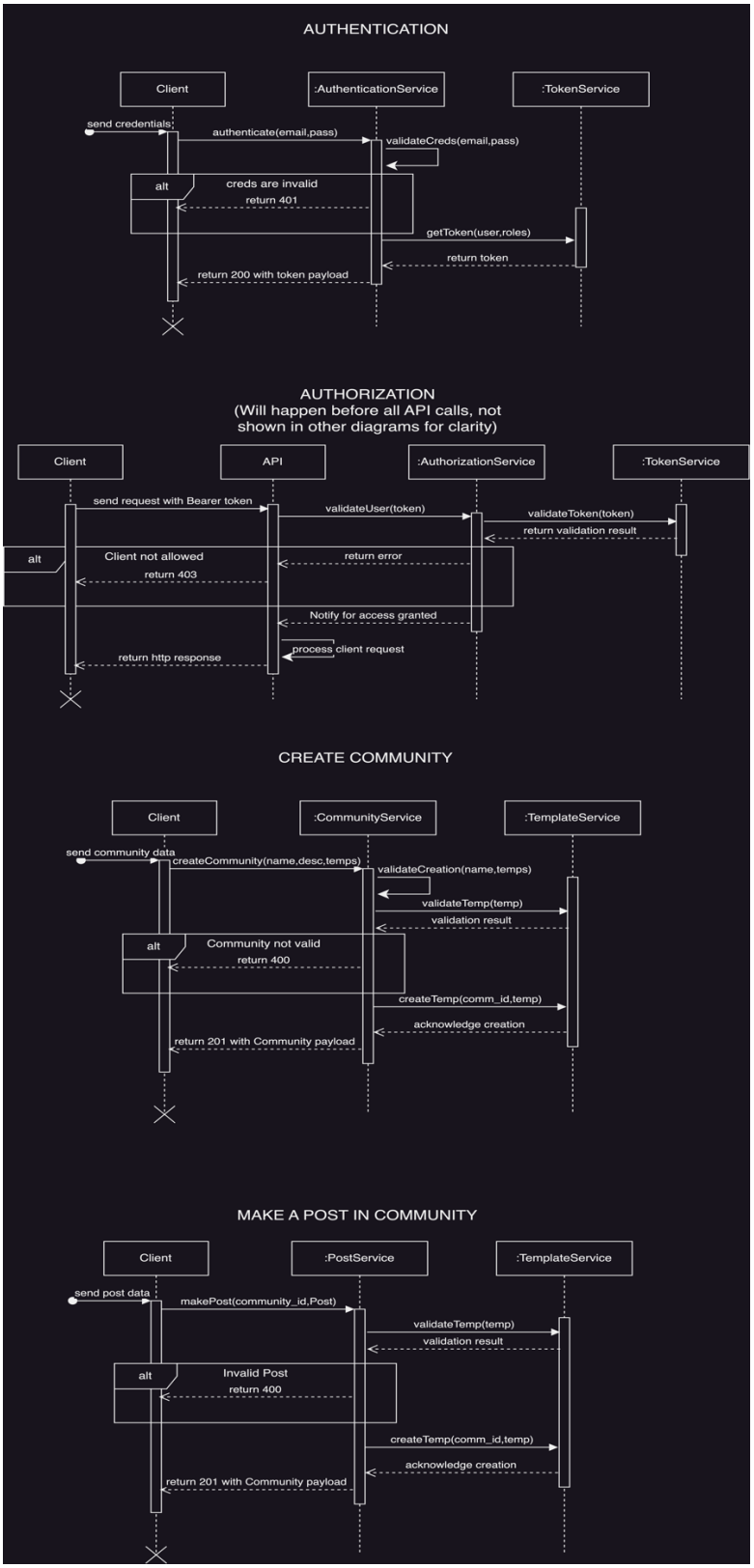
Use Case Diagrams:



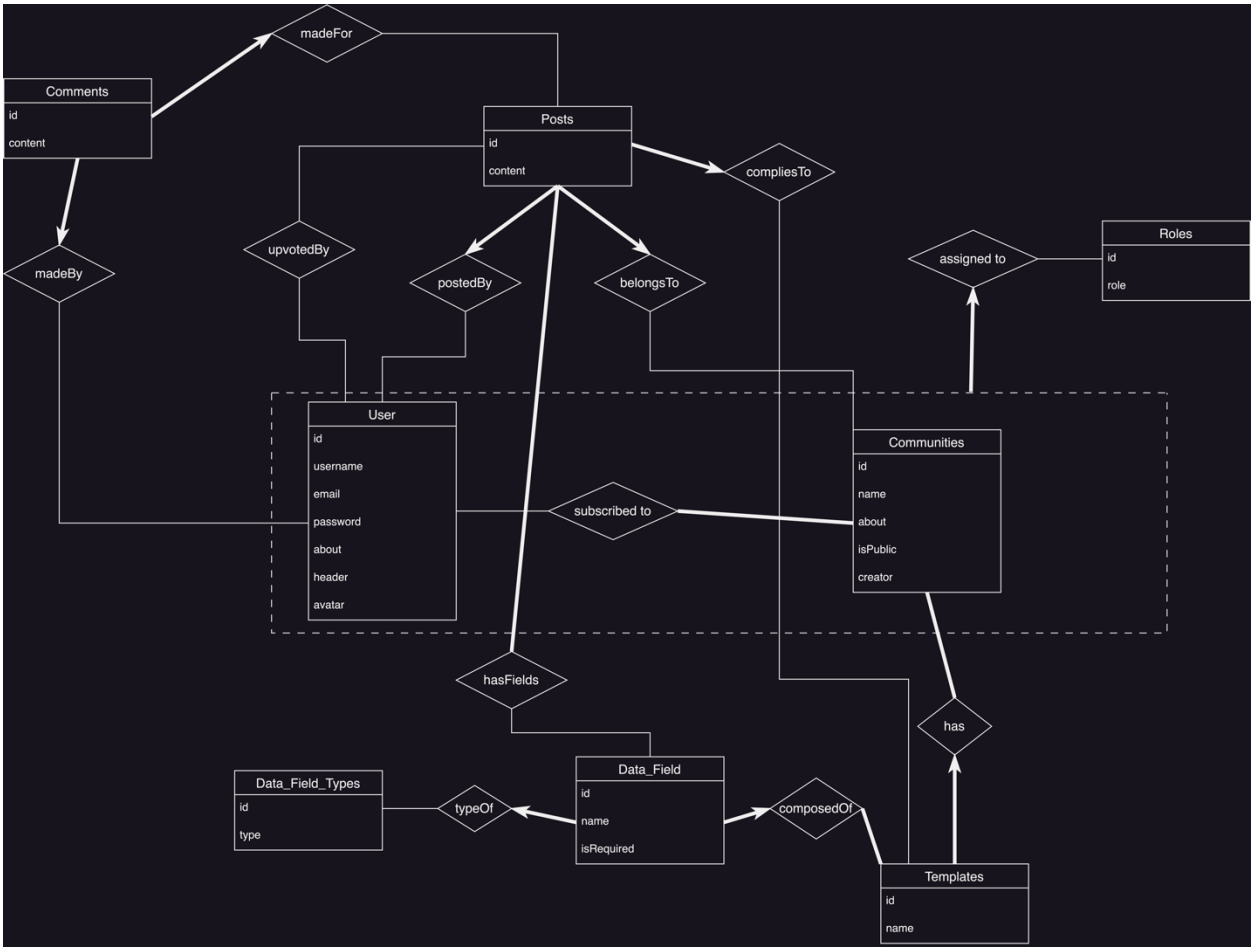
Class Diagrams:



Sequence Diagrams:



Entity-Relationship Diagram:



5 Completed Tasks

Below are the completed tasks in the project.

Until MVP:

- CRUD operations for Users (API) (Register, update profile, delete account, get user profile)
- Method level authorization based on user roles defined for a community (RBAC) (API)
- Creating communities, subscribing and unsubscribing (API)
- Creating templates for communities (API)
- Creating and deleting posts for a given template (API)
- Post validation based on data types (API)

After MVP:

- Creating templates in front end
- Creating posts based on chosen templates on front end
- Home page where users can browse existing communities
- Profile and user pages to view user information including community memberships
- Community page to view details about the visited community including member information and posts
- Basic search in the navbar to quickly search for users and communities with a specific input
- Advanced search mechanism within communities for chosen templates and filters for all data fields except image types. Application also supports geolocation proximity filters in kilometers.
- Geolocation functionalities while creating and viewing posts. Users are able to interactively pick a location on a map interface
- Ability to Join and Leave Communities
- Interactively navigate through users and communities via clickable links in user and community pages
- Image uploads for posts

6 Incomplete Work

Below are the work that has been intended or attempted but could not be completed. Please kindly note that I omitted certain requirements defined in the specification as they are discussed to be completed in the continuation of this course. I still made some investments such as RBAC and advanced filtering including geolocation to better prepare for the remaining requirements.

- Update and delete operations for user profile.
- Update and delete operations for community info (name & about).
- Template deletion is not implemented intentionally for data integrity regarding the old posts. I plan to add a “deprecated” field to the database and continue with that approach in next semester.
- Even though data is incorporated for public/private communities, it does not have any impact on how the application works. It will make more sense as soon as community invitation system is implemented, so again data is included only for preparing the infrastructure for next semester.
- Community deletion is not implemented intentionally.
- No unit tests has been implemented.

7 User Manual

Most of the use cases are quite straightforward and easy for any person that has minimal computer literature. However, nuances in the use cases such as template creation, making posts and advanced filtering are explained below.

Template Creation:

- On the community page, click on “Create Template” button.
- In the creation menu, type the name of the template you wish to create.
- Click on the select dropdown menu and choose the data type for the field you wish to add to the template and finally click “Add Field” button.
- Enter the name of the data field in corresponding box and choose if that particular field will be required to create a post or not.
- Finally click “Create Template” button and exit.

Post Creation:

- On the community page click on “Make a Post” button.
- Choose your template in “Select Template” dropdown.
- Fill the fields for the template with the correct data type described for the field.
- For the required fields, there is “*” marker at the start of the field label.
- Make sure to comply with the format described for the Date field type.
- For geolocation, choose the location you wish on the map, it is possible to move and zoom around the map.

Post Filtering:

- On the community page, click on “Filter Posts” button.
- Choose the template you wish to filter for.
- Fill the filter parameters you wish to use, leave the ones you do not want to use at default settings as they have no filtering effect at those settings.
- Please note that for Date and Number data types, there are min-max and start-end values. It is possible to fill only one of them not to define a range but only a greater/smaller relationship.
- Click on “Filter” button to view filtered posts.
- Click on “Reset Filters” to view all the posts in a community.

8 User Tests

No unit test has been implemented as previously mentioned. However, I conducted several user tests that has been successful. List of those tests are as follows:

- Login, register and logout, redirections occur as a result of these operations on the frontend
- Preventing creating templates as a user that does not have the “creator” role for the community
- Preventing users from making post to a community which they are not subscribed to
- Preventing users from making posts that do not have all the required fields
- Preventing users from making posts that do not comply with the input requirements such as date, number and geolocation formats
- Preventing users from creating templates that has the same name
- Preventing users with “creator” role withing a community from leaving that particular community

8.1 Test Cases and Data

Users:
There has been two users configured to test the application. Communities created with both users to test access controls.

Info for the First User:
Email: cnkt.er@gmail.com
Password: cankut1234
Username: Cankut
Creator for the Community: Sell your House

Info for the Second User:
Email: ecem@gmail.com
Password: ecem1234
Username: Ecem
Creator for the Community: Travelers Hub

Access Control and Data Integrity Test Cases:

- First test is to make a post as “Ecem” in the “Sell your House” community without joining. Expectedly received “Access Denied” error.

- Second test is to create a template in “Sell your House” community as “Ecem”. Since “Ecem” is not the creator of the community and only a user, expectedly received “Access Denied”.

- Third test is to create a template in “Sell your House” community named “Apartment”. Since there is already a template with the same name, expectedly received “Duplicate Entry” error.

- Fourth test is to try register as a user with username “Cankut”, since there is already user with that username register attempt has failed successfully.

Test Cases for Template and Post Creation, Advanced Search

“Sell your House” community has been primarily used for testing these cases as community allows sufficiently complex post templates and more importantly ability to test all the data fields in advanced filtering menu.

A template called “Apartment” has been created in the community. Template has fields “For sale until(date)”, “Description(string)”, “Size in m2(number)”, “Picture(image)” and finally “Location(geolocation)”. Only the “Picture” field is not required to make a post using this template.

First test was to attempt to make a post that does not have all the required fields. Leaving only one required field empty successfully returned an error message “Post does not have all the required fields”.

Second test is to attempt to make a post for Date and Number fields that does not comply with the required format. For the date field, “2024-15-15” is used as an input. Even though this input complies to the format, it has successfully not been accepted as there is no such date. For the number field, “130m2” is used as an input. As the input cannot be parsed as a number, successfully retrieved the error message “Post fields does not comply with the required format”.

Final test was to check advanced filtering menu. For this purpose several posts has been created with diverse fields to make use of all the parameters in filtering menu. Filtering works as a filter chain so a post must conform to the all the parameters defined in the filtering menu. If no input has been defined or left at default settings, filter for those particular fields will not apply. Posts created and filter parameters to test advanced filtering are as follows:

Filter parameters:
Start Date:2024-07-15
End Date: 2024-10-10
A location close to Goztepe, Istanbul chosen on the map, proximity is 5km
Min m2:110
Max m2:135

These filters effectively look for an apartment that is for sale between start and end dates, in a location that is maximum 5k far away from Goztepe and has a size in m2 between 110 and 135.

Posts:

Post1:
Description: Luxury in Umraniye
Size in m2: 120
For sale until: 2024-10-15
Location: Umraniye picked on map

Post2:
Description: Compact place in Nisantasi
Size in m2: 85
For sale until: 2024-10-01
Location: Nisantasi picked on map

Post3:
Description: Cozy place in Basaksehir
Size in m2: 130
For sale until: 2024-09-01
Location: Basaksehir picked on map

Post4:
Description: Apartment in the centre of Kadikoy
Size in m2: 110
For sale until: 2024-08-01
Location: Kadikoy picked on map

Post5:
Description: Nice place in Maslak
Size in m2: 140
For sale until: 2024-07-01
Location: Maslak picked on map

When these filter parameters are applied, apartments in Maslak and Nisantasi are disqualified as per m2 filters. Afterwards, apartment in Umraniye is filtered with date requirements. Among the remaining apartments, the ones Basaksehir and Kadikoy, choice is made based on proximity to Goztepe, as there is a requirement stating that filtered apartment cannot be more than 5km far away to Goztepe location. Thus, Kadikoy apartment has been chosen successfully.

It should also be mentioned that it is also possible to only filter for template names in the filtering menu.