SWE 573 FINAL REPORT

COMMUNITY HUB

Deployment URL: http://ec2-18-153-90-187.eu-central-1.compute.amazonaws.com:8080

GitHub URL: https://github.com/Ozmus/SWE-573-Ahmet-Ozmus

Git Tag Version URL: https://github.com/Ozmus/SWE-573-Ahmet-Ozmus/releases/tag/v0.9

Ahmet Özmüş 2023719081

Advisor:

Suzan Üsküdarlı

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 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.

Ahmet Özmüş

Table of Contents

Overview

The project aimed to design and implement a cloud-based web application which serves as community information system. Spring boot is used in order to this purpose. The objective was to create a web application from scratch and manage the software processes.

Implementation of the project is provided. Java Spring Boot is used as a software platform of the application. Monolith approach is taken, so codebase contains the user interface, business logic, and data access layers. Also, Model View Controller and Data Access Object patterns is applied to the project. Spring Controller classes are used as bridge between view and model layers. In the view layer, thymeleaf used to bind data between html and java files. Service classes are used to access data access objects and achieve the business logic. Lastly, data access objects are used to access database and realize database related operations. MySQL is used as database language. It is one of the structured query languages which is used for managing and manipulating relational databases. Also as a cloud provider, Amazon Web Services is used. Dockerized application and database runned in EC2 machine.

The database schema plays a crucial role in the application's functionality, as it establishes the structure and relationships between various entities. The SQL script provided outlines the database schema for the application, creating and dropping tables as needed.

users Table: This table stores information about the users of the application, including their username, first name, last name, email, and encrypted password. It serves as the primary source of user information and contains a primary key (id) to uniquely identify each user.

communities Table: This table stores information about the different communities within the application, including their name, description, image URL, and archived status. Each community is uniquely identified by its primary key (id).

user roles Table: This table manages the roles of users within specific communities. Each record in the table includes a user ID, community ID, and the role of the user in the community (e.g., MEMBER, OWNER, MOD). This table uses composite primary keys (user id and community id) and maintains foreign key references to the users and communities tables.

content templates Table: This table defines content templates for communities, which allow for consistent structuring of content within a community. Each record contains

a template ID, name, and the community ID to which the template belongs.

contents Table: This table stores the content posted by users within the application. Each record includes an ID, title, user ID, and content template ID. The table maintains foreign key relationships with the users and content templates tables.

fields Table: This table stores metadata about the fields in content templates, such as the field name, data type, and associated content template ID. It supports the creation of customizable content structures within templates.

field values Table: This table stores the values of different fields in contents, associating a specific field value with a content ID and field ID. The table has composite primary keys (content id and field id) and foreign key relationships with the contents and fields tables. A monolithic architecture was chosen for this project to keep the codebase cohesive and straightforward. All components of the application, including the user interface, business logic, and data access layers, are integrated within a single codebase.

Monolithic architecture might create problems in large projects managed by many developers, and comparing to micro service architecture, it is less fault tolerant. However, monolithic architecture is a less complex and fast way to implement the project which consists of one developer in this case.

On the other hand, the project employs the Model-View-Controller (MVC) and Data Access Object (DAO) design patterns. MVC separates the application into logical layers, facilitating better organization and maintainability. DAO abstracts data access logic, and makes the codebase more modular.

Software Requirements Specification

Glossary

- System: The backend system that controls the application.
- User: Someone that uses the application.
- Community: Community is a concept that users come together around shared interests, goals, or activities to interact, collaborate, and support one another on the application.

Requirements

Functional Requirements

Log In Requirements

L1. The system shall provide registration to the application with email and password

L2. The system shall provide log in mechanism using email and password

Authentication and Authorisation

A1. Password shall be encrypted

A2. User shall be able to execute actions according to his/her user role

Community Requirements

C1. While a community created, the system shall provide a default content template

C2. When a user posts a content in that community, the system shall create the content in that community

C3. While a community's privacy policy is public, any user shall be able to enter the community.

C4. While a community's privacy policy is private, only users who is invited shall be able to enter the community.

C5. Community page shall include description and followers informations of that community.

C6. A community shall include specific content types for itself.

User Requirements

U1. The user shall be able to create a community by a unique name, description, and a privacy policy (public or private)

U2. When a user creates a community the system shall assign the user as the creator of that community

U3. The user shall be able to follow communities whose privacy status is public.

U4. The user shall see the contents of communities that he/she follows on the home page.

U5. While a user does not have any community, the user shall see the contents of popular communities on the home page.

U6. The user shall be able to up vote or down vote the contents which he/she has access.

U7. The user shall be able to comment the contents which he/she has access.

U8. The user shall be able to follow the other users.

U9. The user shall be able to leave the community.

U10. The user shall be able to unfollow the users.

Managements Requirements

M1. When an owner or moderator assign a user as moderator the system shall assign moderator role to that user for that specific community

M2. When an owner assigns a user as owner the system shall assign owner role to that user for that specific community

M3. If an owner tries to kick another owner in that community then the system shall show an error message

M4. If a moderator tries to kick another moderator in that community then the system shall show an error message

M5. When an owner or moderator kick a user the system shall delete the user from that community

M6. When an owner kick a moderator in that community the system shall delete the user from that community

M7. When an owner retrieve a moderator's role the system shall delete the user's moderator role in that community

M8. A moderator or owner shall be able to create new content type with specifying its fields (name as text and value as w3c data types).

M9. A moderator or owner shall be able to delete a content type which already exist in the community.

M10. A moderator or owner shall be able to edit a content type with specifying its new fields (name as text and value as w3c data types) or deleting existing fields.

M11. While community's privacy status is private, a moderator or creator shall be able to invite users using their email addresses.

M12. If there is only one owner in a community then the system shall not allow the owner to leave the community.

M13. The creator shall be able to archive the community. Contents will be still available, yet no new posts will be allowed.

M14. The creator shall be able to unarchive an archived community.

M15. The creator or moderator shall be able to delete posts which posted in community.

M16. The creator or moderator shall be able to edit community description.

Search Requirements

S1. The user shall be able to search contents on home page with including at least one of the community name, content title, content type, and user name of the poster information.

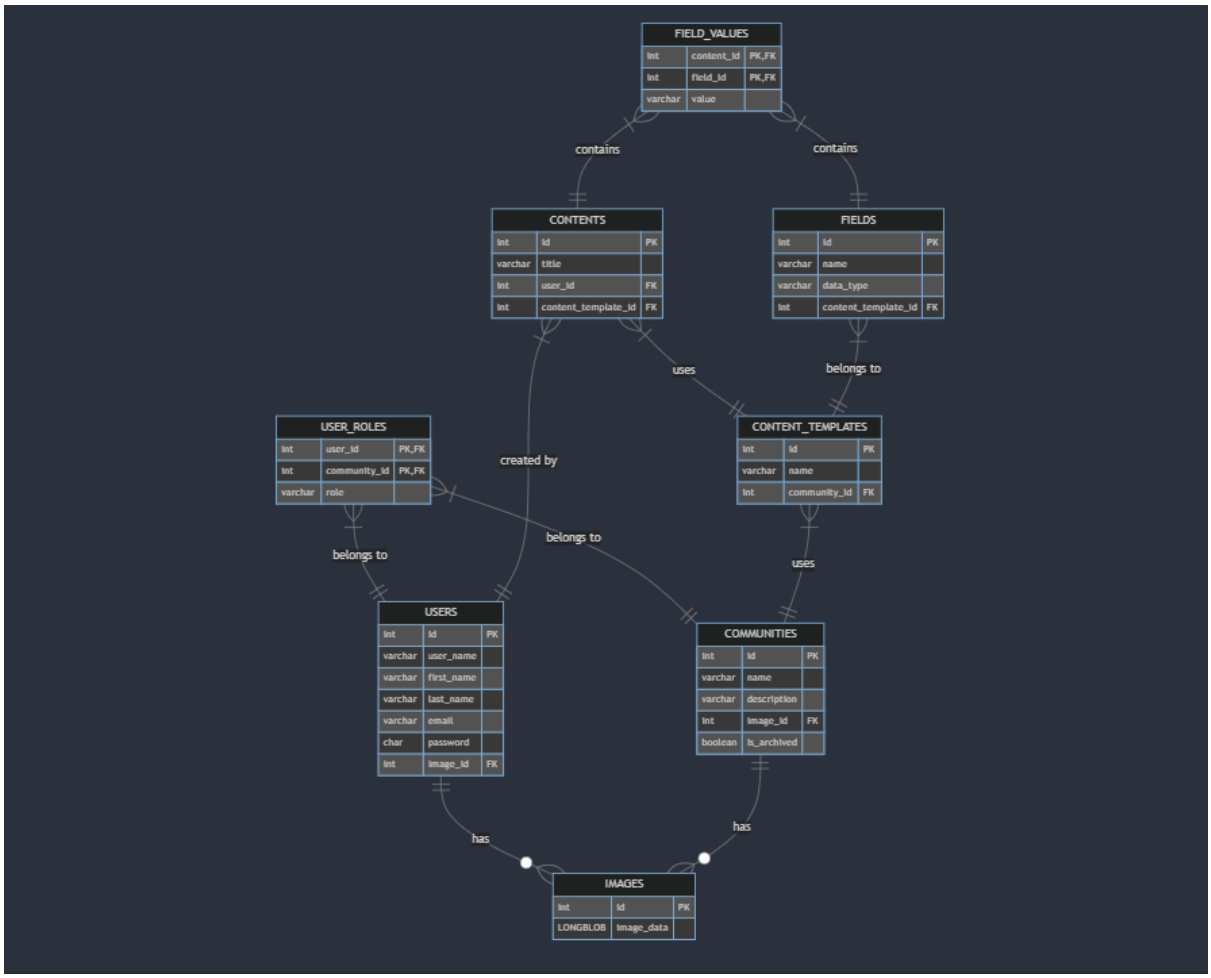
S2. The creator or moderator shall be able to search users on management page with including at least one of user email address, and user name.

S3. The creator or moderator shall be able to search content types on management page with including content type name.

S4. The user shall be able to search followers with username.

Design Documents

ERD



MOCKUPS

Status of Project

Completed Requirements

Log In Requirements

L1. The system shall provide registration to the application with email and password

L2. The system shall provide log in mechanism using email and password

Authentication and Authorisation

A1. Password shall be encrypted.

A2. User shall be able to execute actions according to his/her user role.

Community Requirements

C1. While a community created, the system shall provide a default content template.

C2. When a user posts a content in that community, the system shall create the content in that community.

C3. While a community's privacy policy is public, any user shall be able to enter the community.

C5. Community page shall include description and followers informations of that community.

C6. A community shall include specific content types for itself.

User Requirements

U1. The user shall be able to create a community by a unique name, description, and a privacy policy (public or private)

U2. When a user creates a community, the system shall assign the user as the creator of that community.

U3. The user shall be able to follow communities whose privacy status is public.

U4. The user shall see the contents of communities that he/she follows on the home page.

U9. The user shall be able to leave the community.

Managements Requirements

M1. When an owner or moderator assign a user as moderator the system shall assign moderator role to that user for that specific community

M2. When an owner assigns a user as owner the system shall assign owner role to that user for that specific community

M3. If an owner tries to kick another owner in that community then the system shall show an error message

M4. If a moderator tries to kick another moderator in that community then the system shall show an error message

M5. When an owner or moderator kick a user the system shall delete the user from that community

M6. When an owner kick a moderator in that community the system shall delete the user from that community

M7. When an owner retrieves a moderator's role the system shall delete the user's moderator role in that community

M8. A moderator or owner shall be able to create new content type with specifying its fields (name as text and value as w3c data types).

M9. A moderator or owner shall be able to delete a content type which already exist in the community.

M10. A moderator or owner shall be able to edit a content type with specifying its new fields (name as text and value as w3c data types) or deleting existing fields.

Search Requirements

S1. The user shall be able to search contents on home page with including at least one of the community name, content title, content type, and user name of the poster information.

Not Completed Requirements

Community Requirements

C4. While a community's privacy policy is private, only users who is invited shall be able to enter the community.

User Requirements

U5. While a user does not have any community, the user shall see the contents of popular communities on the home page.

U6. The user shall be able to up vote or down vote the contents which he/she has access.

U7. The user shall be able to comment the contents which he/she has access.

U8. The user shall be able to follow the other users.

U10. The user shall be able to unfollow the users.

Managements Requirements

M11. While community's privacy status is private, a moderator or creator shall be able to invite users using their email addresses.

M12. If there is only one owner in a community then the system shall not allow the owner to leave the community.

M13. The creator shall be able to archive the community. Contents will be still available, yet no new posts will be allowed.

M14. The creator shall be able to unarchive an archived community.

M15. The creator or moderator shall be able to delete posts which posted in community.

M16. The creator or moderator shall be able to edit community description.

Search Requirements

S2. The creator or moderator shall be able to search users on management page with including at least one of user email address, and user name.

S3. The creator or moderator shall be able to search content types on management page with including content type name.

S4. The user shall be able to search followers with username.

Status of Deployment

The application is deployed in the Amazon Web Services. It can be accessible with the following URL: http://ec2-18-153-90-187.eu-central-1.compute.amazonaws.com:8080 . Dockerized application is deployed with EC2 machine.

Firstly, application is dockerized using the Dockerfile (https://github.com/Ozmus/SWE-573-Ahmet-Ozmus/blob/main/Dockerfile) , and created an docker image. The image pushed to hub.

Secondly, docker-compose.yml file (https://github.com/Ozmus/SWE-573-Ahmet-Ozmus/blob/main/docker-compose.yml) formed. This file uses the image created in the first phase. Also, with this file, the database is formed. MySQL image is used for this purpose. With sql script (https://github.com/Ozmus/SWE-573-Ahmet-Ozmus/blob/main/sql_scripts/create.sql), the database and tables are formed in this docker-compose.yml file.

Lastly, I created a EC2 machine. “sql_scripts” folder that includes “create.sq”l and “docker-compose.yml” files are manually added to the EC2 machine. Then, docker is installed in the EC2 machine. After login in the docker, the image (099000/community-application:latest) that pushed to docker hub is pulled in the EC2 machine. After installing docker-compose in EC2 machine docker-compose up -d command is runned. With these processes the containers for dockerized application and database are created and started. After defining a security group that allows traffic on EC2 from 8080 port, the application is became accessible from internet.

Instructions for Installation

Option 1: Docker

1- Go to Git Hub Repo : https://github.com/Ozmus/SWE-573-Ahmet-Ozmus
2- Download the code
3- Prepare docker and docker-compose in your environment
4- Run docker-compose up -d command in the terminal (git bash for windows)

Option 2: Manual

Open the project and prepare your local environment for the project. Use java 17 (amazon-coretto). Download mysql 8.3.0 and configure src/main/resources/application.properties file according to your datasource username, password, and url. Run “sql_scripts/create.sql” file in your MySQL server. Also, docker's latest mysql image can be used, which I prefered. In this case do not forget to set MYSQL_ROOT_PASSWORD environment variable and the port if 3306 port is already in use. After these steps application is ready to be ran in the local environment. Access the application in 8080 port from your localhost.

User Manual

1- Register
2- Log in
3- Go to community tab and create community
4- After creating community form does not appears. It is a known bug. Use back button on browser.
5- Go home page
6- Click post and create content
7- Go management page
8- Edit content template and add new content template
9- Go home page and create new content with new template
10- Search using keyword
11- Use advanced search (advanced search uses exact match. If you left one field empty it will not search for it. Only not empty fields took in consideration. Without community content template cannot be searched.
12- Log out
13- Create another user
14- Join created community using community tab and clicking on the community
15- Log out
16- Login in previous user who created the community
17- In management tab promote depromote and kick the second user
18- In profile page edit user fields

Test Results

Tests are executed as unit tests and manually. All unit tests are passed.

Video

https://drive.google.com/file/d/1-rCaK5sHPju4N4RcihSCb86Q1v5ME7Bd/view?usp=sharing