Kelven Nathanael

E1528447@u.nus.edu

A0325540M

SIMPLE WEB FORUM

Theme: A Minimalist Social CRUD Application

Architecture: Model-View-Controller (MVC)

1. Overview

This project is designed to apply core web development principles in a minimalist way to create a functional "Social CRUD" application. To achieve this, I utilized the MVC (Model-View-Controller) architecture. This pattern allows for a clear separation of concerns, structuring the code so that it is significantly easier to maintain, scale, and debug.

2. User Stories

User stories define the requirements of the application from the perspective of the end-user. For this minimalist web forum, I prioritize the "Social CRUD" experience:

- As a Guest: I want to view a feed of all posts so that I can see what the community is discussing without needing an account.
- As a Registered User: I want to create text-based posts and comments so that I can share my thoughts with others.
- As an Author: I want to edit or delete my own posts so that I can correct mistakes or remove content I no longer wish to share.
- As a Reader: I want to comment on a post so that I can engage in an active conversation.

3. The Model

The Model represents the data layer. If I were to put it in analogy, it is like the kitchen inside a restaurant. It handles the "cooking" (querying) and the storage of the ingredients (data). I chose PostgreSQL for this project because it is fast, reliable, and handles relational data (like connecting a comment to a specific post) very efficiently. For the data scheme, I use as the one below.

1. User Entity:
    - id (Primary Key)
    - Username (Text)
2. Post Entity:
    - id (Primary Key)

    - title (String, limited to 100 characters for UI consistency).
    - content (Text-based body).
    - username (Foreign Key linked to the User).
    - timestamp (To track when the post was created).
3. Comment Entity:
    - id (Primary Key).
    - post_id (Foreign Key linked to the parent Post).
    - username (Foreign Key linked to the User).
    - body (Text).

4. The View

The View is the presentation layer. For this project, I used React.js with TypeScript and Material UI (MUI).

To keep the website minimalist, I made a specific design choice regarding the login. Since the application only requires a username, a dedicated login page felt unnecessary and inconvenient. Instead, I designed a small login box directly in the header. This makes the application more accessible, allowing users to "log in" and start posting without ever leaving the main feed.

5. The Controller

The Controller acts as the bridge connecting the Frontend (View) to the Backend (Model).

- Language: I used Go (Golang) Go was chosen because it is one of the fastest languages for web servers, ensuring that the forum remains responsive even as the number of posts grows. Additionally, I use gin framework to ease the coding.
- Communication: The Controller passes objects and variables using JSON. This makes it easy for the React frontend to parse the data and update the UI instantly without needing a full page reload.
- Security: The Controller is responsible for checking the username before an Update or Delete command is sent to the database, ensuring users can only modify their own content.

6. Conclusion

Using the MVC architecture provided a clear roadmap for this project. It helped clarify which technical libraries were needed and ensured that each component had a specific job. By defining the Model first and understanding exactly what variables needed to pass through the Controller, the design of the View became much more intuitive. This structure allows the "Minimalist Forum" to be both simple for the user and professional in its code organization.