CVWO Mid-submission Write Up

Ngo Hoang Nhat Minh

Use Cases

Description

NUSForum is a user-friendly web platform designed for discussions about all things NUS. The forum is organized into communities tailored to specific interests or groups, such as dedicated pages for different courses. Within these communities, users can initiate discussion threads about relevant topics, while others contribute by leaving comments to further the conversation. Additionally, a voting system allows users to rank posts based on popularity and relevance.

Basic Create, Read, Update, Delete (CRUD) Operations

The home page indexes the trending posts which users can click on to read. To create new posts and comments, users have to log in first. Logged-in users can click on their profile page to view their history - which includes their posts and comments. They can then click on these posts and comments to have the option to edit or delete them.

Upvote and downvote post

Each post includes an upvote and downvote button. Users can upvote helpful posts, ignore posts they find neutral, or downvote those they consider low-quality.

Categorize post

When creating a post, users must choose a category for the post. This will help later users to sort and search for posts more easily and accurately.

Search for threads

On top of the forum page, on the header bar, there is a search bar that allows the user to easily and conveniently search for any post. When the user type in specific keywords, relevant options matching the keywords will drop down for users to choose.

Sort posts by topics

On top of the list of posts and comments, there is a sorting menu with different sorting options. The user can choose to sort the post by ”Most recent”, ”Most upvote”, or ”Trending”. For more customization, users can also choose the timeframe to sort: ”day”, ”week”, ”month” or ”all time”.

Create and joining communities

Users can create communities about a specific topic. They will be granted admin rights to the communities and have the authority to manage any posts and comments within the communities. They also have the option to kick out users who violate the community’s guidelines.

Use Cases Diagram

Figure 1: Use Case Diagram

Database Diagram

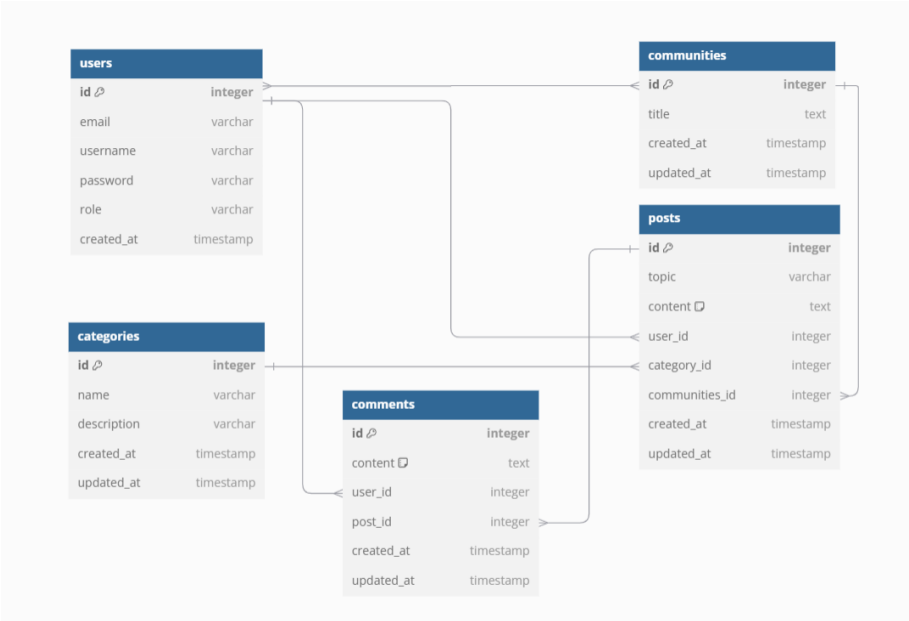

Figure 2: Database Diagram

Execution Plan

Backend

The backend will use the MVC architectural pattern with Ruby on Rails and a SQLite database. It will communicate with the frontend by sending JSON data through API endpoints, enabling independent development and testing of the backend and frontend while ensuring seamless integration. The database schema includes a User table, a Post table, a Category table, a Community table, and a Comment table. Associations and parameters of each table have been mapped out according to Figure 2.

Authentication will be handled using the Devise gem to include user registration, login, and password recovery.

Frontend

After setting up the backend, I will integrate it with the frontend using React, which will communicate with the Rails backend through API calls. Cross-Origin Resource Sharing (CORS) will be configured on the Rails backend to enable secure data fetching from the frontend application.

First, I will begin with implementing the CRUD operations for viewing, creating, updating, and deleting posts. Next, CRUD operations for comments will also be added, including creating a comment section under each post and options to update and delete comments. I will use React's state management to create a dynamic user experience.

After finishing with the skeleton of the forum, I will implement a category system and community feature to sort and organize posts. Voting functionality for upvoting or downvoting posts and comments will also be introduced to engage users. The homepage will display posts sorted by votes to highlight popular content.

Design and User Experience

To ensure a user-friendly interface, Bootstrap will be used for styling and responsive design. I will also adhere to accessibility guidelines to make the forum inclusive for all users. Mobile responsiveness will also be thoroughly developed, as forums are often accessed on smartphones.

After meeting the minimum viable product standard, I will focus on developing additional functionalities to enhance user experience. A basic moderation system will be implemented, enabling community admins to maintain quality and standards. Features such as reporting posts and comments will also be added. On the backend, additional security measures, such as rate-limiting API endpoints and encrypting sensitive data, will be implemented to ensure user data protection and prevent abuse. I will also set up a Cron Job scheduler to regularly back up the database.

If I still have time, I will learn to use Redux to manage my React application state, and Docker to package the software into standardized Containers.

Testing and Deployment

Thorough testing will be conducted for both the backend and frontend. RSpec will be used for backend testing, and Jest will be employed for frontend unit and integration testing.

The backend will be hosted on Heroku, and the frontend will be deployed on Netlify.