CVWO Mid-Submission Write-up

Gao Tianrun
A0252114W

I will be making the web forum with a ReactJS frontend website and a Ruby on Rails backend API, and the frontend will communicate with the backend using API requests.

Use Cases

The web forum will support basic CRUD functions (post, read, edit, delete) for thread posts and comments, and a tagging system to categorise threads. It will also have a basic authentication system using username and password, a search feature that searches for tags, users and posts, and a sorting feature to order threads.

Posting a new thread	**Primary actors:** Logged-in users **Flow:** 1. User goes to the new thread page 2. User fills in the title and body of the thread 3. User can select tags from a dropdown list of popular tags or create their own tags to add to the thread (Optional) 4. User clicks on the “Post” button 5. Thread will be added to the database
Editing a thread	**Primary actors:** User who posted the thread **Flow:** 1. User clicks the edit button of the thread they want to edit 2. User edits the title, tags and body of the thread in the same format as posting a new thread 3. User clicks on the “Post” button 4. Thread will be updated
Deleting a thread	**Primary actors:** User who posted the thread **Flow:** 1. User clicks on the delete button of the thread they want to delete 2. Thread will be deleted from the database
Posting a new comment on a thread	**Primary actors:** Logged-in users **Flow:** 1. User goes to the thread post they want to comment on 2. User clicks on the “Comment” button 3. User fills in the body of what they want to comment 4. User clicks on the “Post” button 5. Comment will be added to the database and is linked to the thread post

Editing a comment	**Primary actors:** User who posted the comment **Flow:** 1. User goes to the comment they want to edit under the thread post 2. User clicks on the “Edit” button 3. User edits the body of the comment 4. User clicks on the “Post” button 5. Comment will be updated
Deleting a comment	**Primary actors:** User who posted the comment **Flow:** 1. User goes to the comment they want to delete under the thread post 2. User clicks on the “Delete” button 3. Comment will be deleted from the database
Searching content	**Primary actors:** All users **Flow:** 1. User enters text into the search bar 2. User can click on the options in the dropdown list which shows tags or users which matches the inputted text 3. User will be brought to the tag or user page OR 1. User enters text into the search bar 2. User presses enter 3. Threads with titles or bodies matching the inputted text will be displayed as well as matching tags and users
Liking threads	**Primary actors:** All users **Flow:** 1. User goes to the post thread they want to like 2. User clicks on the “Like” icon button 3. Likes of the thread will be updated
Sorting threads	**Primary actors:** All users **Flow:** 1. User clicks on the “Sort by” button 2. User selects the order to sort the threads (e.g. by most likes, newest, oldest) 3. Page will be updated to show the sorted list

Execution plan

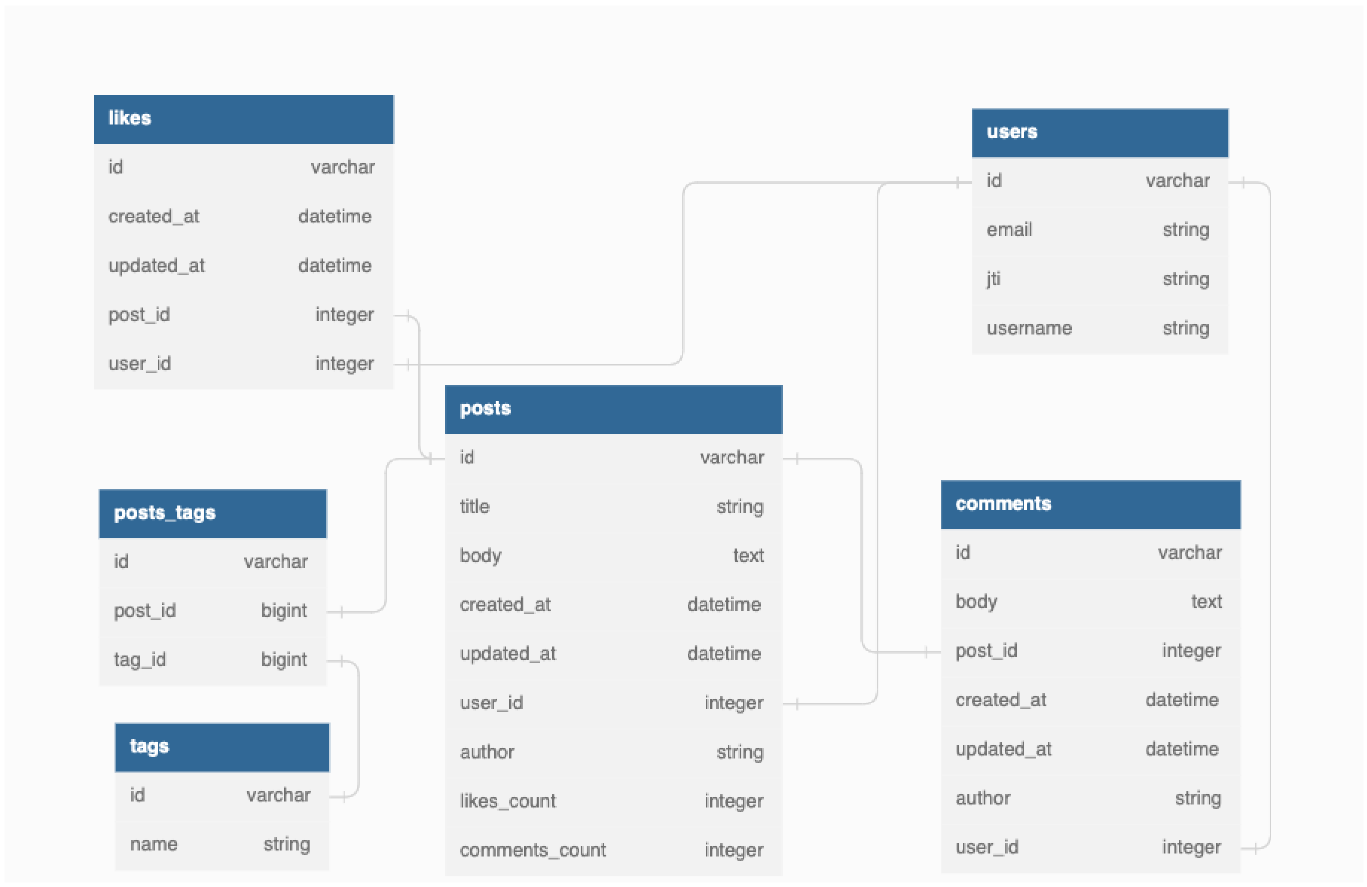
I have decided to create 2 separate apps for the frontend and backend. Since this is my first time learning how to use ReactJS and Ruby on Rails, doing this will create a clear distinction and separation between them and prevent any confusion for me.

First, I will create the Ruby on Rails backend as well as create the database to store the thread posts and comments of the web forum. This will ensure that the backbone of the website will be completed and set its foundation. I will be using PostgresQL for the database as hopefully I will be able to deploy the app on Heroku, which only has support for PostgresQL. Controllers will also be created in the Ruby on Rails app to handle the API data requests from the frontend.

Then, I will move on to create the basic barebones framework for the ReactJS website, ensuring that the basic CRUD functionalities work before adding extra features such as user authentication and tagging of the threads. It will be using axios to make ajax requests to the backend to perform the CRUD functionalities.

Finally, I will focus on integrating MaterialUI and using CSS to make the website user-friendly, more aesthetic and intuitive to use. Any extra time left after finishing all of this will be used to learn and incorporate extra features or tools from the optional level.

Database Schema



likes
id varchar
created_at datetime
updated_at datetime
post_id integer
user_id integer
users
id varchar
email string
jti string
username string
posts
id varchar
title string
body text
created_at datetime
updated_at datetime
user_id integer
author string
likes_count integer
comments_count integer
posts_tags
id varchar
post_id bigint
tag_id bigint
tags
id varchar
name string
comments
id varchar
body text
post_id integer
created_at datetime
updated_at datetime
author string
user_id integer