CVWO Assignment 22/23 Final Submission

Ng Rui Jie, Solomon A0253046L

Learning Points

Rails API (JSON, routes, validating ApplicationRecord, schema structure), React (Props, useEffect, React Router Dom), MUI, Typescript, postgresql (querying database), user authentication (JWT tokens, bcrypt to encrypt passwords, caching user information and token), deployment using render and netlify.

Process

This project has been very fulfilling. Starting from ground zero, there was much I had to learn. Each challenge was another "Dunning-Kruger Effect" cycle, that took some googling and reading to upskill and much resilience to problem solve issues. Despite the rather uphill climb at the start, much technical skills were gained during the cycle of problem solving.

Knowing how much learning I had to do, I adopted a strategy of scaling the project a step at a time and constantly checking for issues. For the same reason, user authentication was the most challenging part, as many implementations in rails and react had to come together at once, before the JWT token and user caching could be tested, making it more difficult to pinpoint the location of potential bugs. Although I initially made the mistake of paying too much attention to CSS and styling of the frontend, focusing more on the functionalities helped me make greater leaps towards project completion.

Functionality of Web Application

ChitChat is an online forum that allows users to share their thoughts online by creating a thread online to start a conversation or commenting in response to other's threads.

Frontend (React)

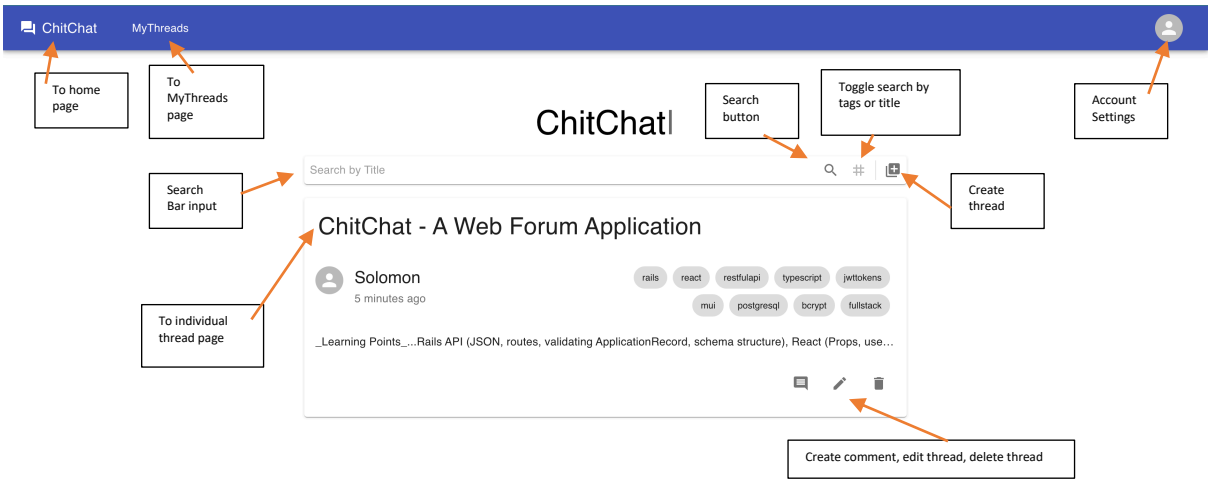
- CRUD functions for threads and comments (create, view, edit, delete)
- Dialog (MUI element) for create/edit/delete threads and comments (user can stay on current page and fill in form on "pop up" page)
- User accounts (username and password, change password, create account)
- User authentication (caching JWT token and user information, login, logout, auto login whenever page refreshes)
- Search bar (toggle search by title or tags, dropdown autocomplete)
- Navigation bar (home page, mythreads page and account settings)
- Tagging (give users control over categorizing their threads, search by tags)
- Global messages and alert user elements (notify users of success / error messages)
- Error checking (reject empty fields for create/edit threads and comments, trim whitespaces on username, attempt to create thread and comment when not logged in etc)
- Date and time (usage of moments library to print date/time as days/hours/minutes/seconds ago for more intuitive and concise view)
- MyThreads page (view all threads created by current user)

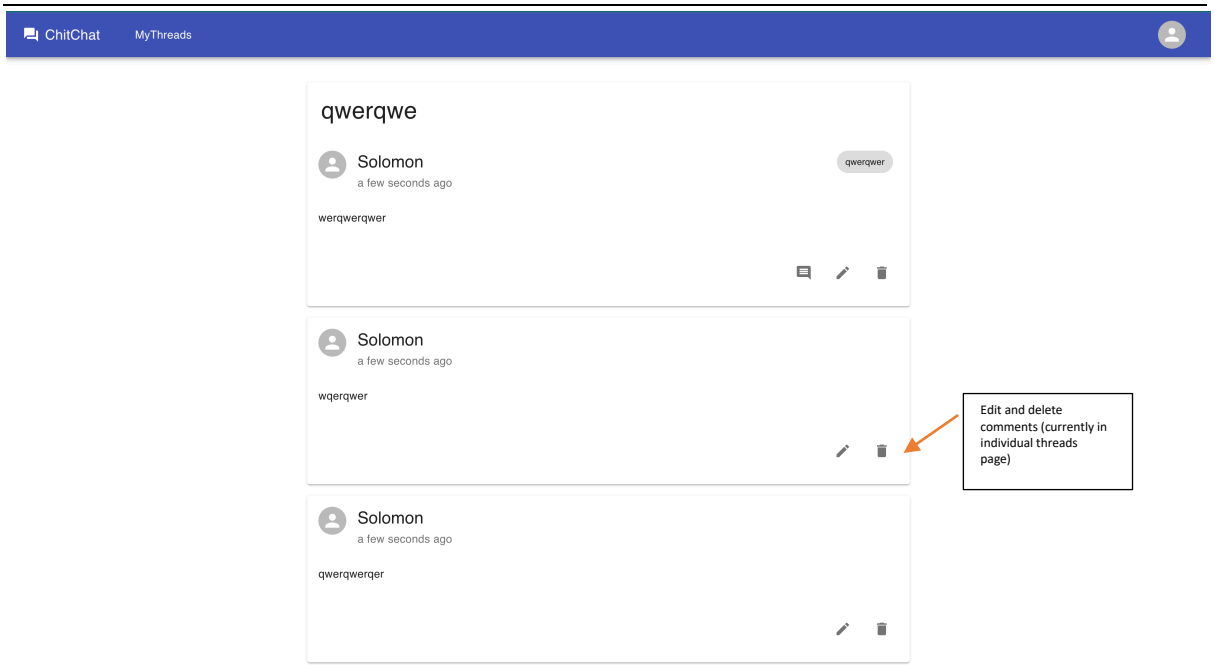
Backend (Rails API)

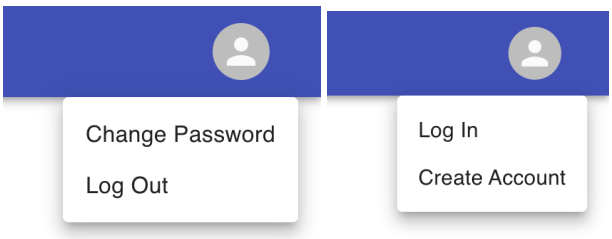
- User authentication (login, auto login by checking authorization header)

- User accounts (create accounts, change password)
- CRUD routes/methods for threads and comments, and additional routes/methods for search bar and autocomplete
- Password encryption (using bcrypt)
- JSON nesting and errors (in string array form)
- Comments routes nested under thread routes (by nesting resources in routes.rb)
- Error checking data fields (using “validates” in application record under model directory), especially user’s username and passwords (length, unique etc)

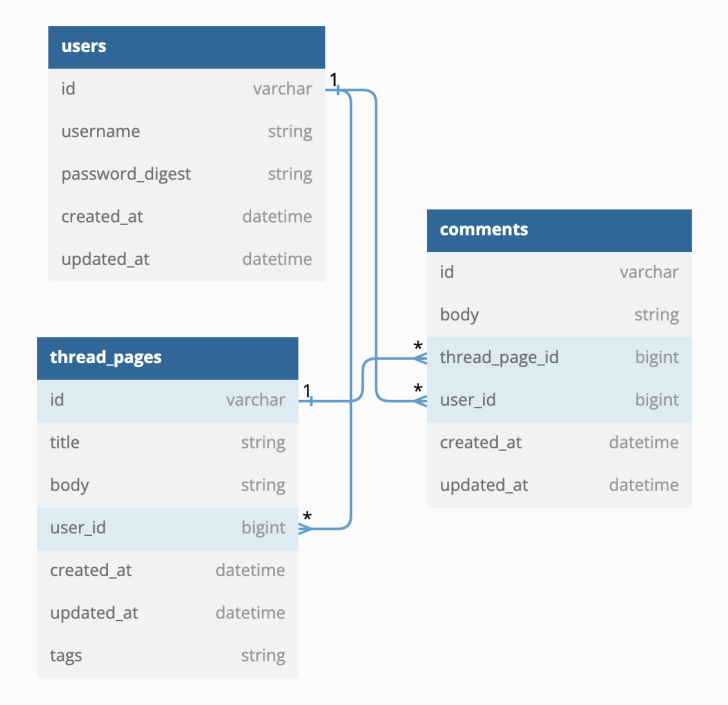
User Manual







Schema



Future Improvements

As passing props in react from component to component (especially “global” props) was tedious and inefficient, it would be useful to learn redux to better manage such variables. Moreover, I would read up more on the security aspects of web application, to better secure user information. Lastly, if I were to do this assignment again, I would first focus on the functionality to create a backbone web application, before adding CSS and styling, which would be more efficient, and set priorities right.

Overall, this has been a genuinely rewarding project, with many skills gained.