Reflection on overall project:

As I reached the end of the project deadline, reflecting back on the entire process I felt like I have gained a whole new understanding for web development as well as how web applications work in general. I decided to use a React-on-Rails framework for this project.

Through learning to create a full web application for the project, I have gained a deeper understanding of how to properly utilise React components to dynamically display content. Somewhere down the line, I also learned to integrate Bootstrap to make generating html component easier and look nicer as well. One area for improvement for sure could have been the CSS styling for the entire application. The different divisions and tags in the components are all references by class names which resulted in a long list of arbitrary sounding class names. This could definitely be done better and is at it's current state not very possible to maintain and extend. More could have been done to organise the css code or to use another implementation of CSS such as tailwindCSS or SASS in order to make it more maintainable.

While I was learning Reactjs, I was mainly using a static json server as my endpoint to fetch data for my react components. Only when I started digging further into understanding what are RESTFUL APIs and how are they used in everyday applications, did I start understanding how frontend frameworks, such as Reactjs or Angular, are linked to backend which holds all the data required. I also managed to learn more about the benefits of a relational database such as POSGRESQL or SQLITE3. Having a database where entries can be linked to each other with various relationships allows for complex API calls that is required in modern applications. This was a stark contrast to what I was using initially which was not relational and every end point only returns the data in json format that was hardcoded into the server. At the same time, I also manage to pick up useful tools to help test APIs such as Postman which allows us to send requests directly to the API and read the return value which was exceptionally useful. I was also pleasantly surprised at how Rails was able to generate SQL queries for you through the controller which saved quite a lot of time as I did not have to figure out how to generate proper SQL commands in order to obtain the data that I needed.

User manual:

The web application is not completely integrated even though it is in the same repository. Therefore, two separate commands must be run in the terminal to start up the web application.

Front-end: npm run start
Back-end: rails s -p 8000

DB schema: