Han Geng Ning
A0222055U

Basic Use Cases

Use Case Name	Use Case Description
Login to App	Existing user logs in to app
Sign up for App	New user creates an account
Forget password for App	User clicks forget password link on log in page and receives an email with a link to reset password
Sign out of App	User can sign out of app.
See all to-do items	User sees all their to-do items on home page
Mark to-do item as completed	User can toggle a checkbox to mark it as complete or incomplete
Delete to-do item	User can delete a to-do item. They will be alerted to confirm their deletion before the item is deleted.
Delete completed to-do items	Users can delete all completed to-do items. They will be alerted to confirm their deletion before the item is deleted.
Search for to-do item	User can search for a to-do item by category or by its task name.
Rearrange to-do item	User can view all to-do items, or view to-do items by category, by priority or by completion
Create new to-do item	User creates new to-do item. Each to-do item has these attributes: task, description, category, is_completed and is_priority
Edit one to-do item	User can see and edit all the attributes of one to-do item.

Extra Use Cases if time and ability permit

Use Case Name	Use Case Description
Schedule to-do item	User can include the time that a to-do item has to be done
View to-do items by datetime	User can view to-do items on a calendar based on its scheduled time
View Progress	User can see how many items are completed in a progress bar

Execution Plan

<u>Set up Ruby on Rails backend</u>

Han Geng Ning
A0222055U

First, I will set up the model and controllers in Rails. There are 2 models – 'User' and 'Todo'. The 'User' model is created using the gem 'devise'. Hence, the controller and CRUD methods to manage users are set up through 'devise'. Then, I will create a 'Todo' model with the following attributes: task, description, category, is_priority and is_complete. 'User' will have a has_many relationship with 'Todo' and 'Todo' will belong_to 'User'. The 'Todo' controller will have CRUD methods and a method to delete multiple items and will render json so that API requests can be handled. There will be a before_action to authenticate users before they can see or handle any to do items. I will use 'devise_token_auth' gem to help me generate tokens so that I can handle HTTP requests.

<u>Set up React front end</u>

I will install create-react-app in a separate client folder. All views will be rendered through React only. React hooks will be used to store the state and axios is used to make HTTP requests to carry out CRUD and fetch data from the Rails backend when necessary. CSS and some bootstrap will be used for basic styling. The main pages to be rendered are the Login page, Signup page, Forgot Password Page, Reset Password Page, Home page (displaying all the to do items, the search bar and navigation links), To-do-Edit page and a To-do-Create page. Other than CRUD, a filter method has to be implemented so that users can search by category or view by category. I will implement this on the front end as the database of to do items is not large in this app, and all the data will already be sent to the front end. The search should be dynamic and faster if done on the front end.

<u>Deploy</u>

The backend and frontend will be deployed separately on Heroku. After deploying the backend rails app, I need to ensure the paths in the HTTP requests are set correctly before deploying the frontend.

If time and ability permits, I will execute extra use cases.

Current Problems

By the mid-assignment I have executed most of the basic use cases, during which I faced many problems but managed to overcome most of them. Ruby, Ruby on Rails, React and HTTP requests were completely new to me.

My first problem was figuring how to learn them as I did not know where to start from in the documentation. Since I learn best by application, I watched several YouTube videos to see how the MVC framework works on rails and how the controller links the model and view. I also learnt React through video tutorials. From that, I was able to narrow down on specific parts of the documentation to find more information when necessary.

Next, I was confused about organization. At first, I thought that React could simply be installed and implemented within Rails JavaScript folder. However, upon further reading, I realized that it was best practice to develop the front end separately so that it was easier to scale, especially for bigger apps where many people work on them together. This made me aware of the importance of organization, be it in file structure or in code so I was more careful when executing the rest of my project. For example, I made React components when I had to repeat similar functionality across different pages.

I was most confused about integrating React and Rails. As I used devise for user authentication, I did not know how to override the devise controller such that the react frontend could send HTTP requests and execute the relevant methods. I also did not understand HTTP requests well and had trouble debugging the error messages. After some research, I realized that tokens could be passed to authorize actions and there was a gem called 'devise-token-auth' that helped to generate tokens and worked with devise. I also gained skills in debugging such as reading the console to see data and executing small sections of code at a time to see where the problem actually occurred.

Currently my app and most of the basic use cases seem to work locally but I am not sure if there are scenarios that I failed to consider or gaps in my logic. Doing comprehensive tests on a piece of software is something that I have yet to learn. There are probably better ways to organize my code that I am not aware of too.

My next hurdle will be learning how to deploy an app and figuring out the difference in production and development stages. I also need to set up my action mailer for production. After that, I will consider some of the extra use cases or see if there's another area I can learn about. Overall, it has been really interesting so far and it was really exciting to learn and make a full stack app.