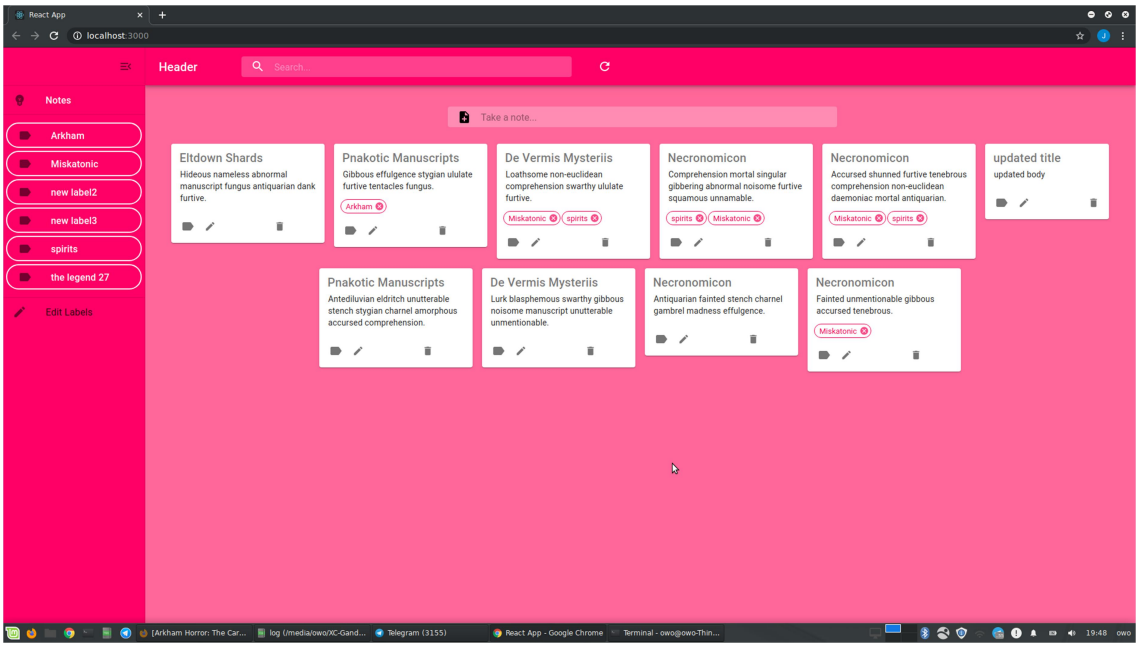
CVWO MidAssignment Submission
Ong Yin Ming Jonas – A0155237E

Execution Plan

I plan to build this ToDo manager in a form of a single page application.

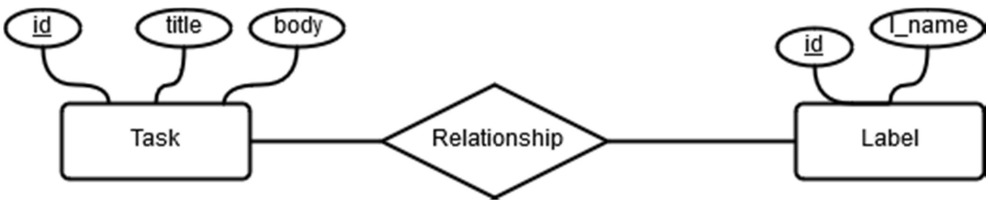
Frontend

Will be done in React, and I will be using Material-UI to simplify the design work of styling using prebuilt components. I will also be using the node module Axios for http requests to the Rails Server. A mockup is attached below.



Backend

The Rails server will be built to serve JSON API to the frontend to display. A current view of the Postgres database would involve three tables, the Task table (id, title, body of the note), the Label table (id, name of the label/tag) and a Task-Label association table (task id, label id of the association). I believe this structure would suit the application allowing a task to be labelled with more multiple labels. The following is a diagram representing the tables in Postgres. Label name (l_name) is unique and enforced by Rails Model.



Task2. Categorizing Tasks

I intend implement the application such that to tag a task, the user has to create a new tag/label (see use case UC15 extension) or choose from a list of existing tags to tag the task. While this adds additional steps to tagging a note, it would prevent issues such as spelling mistakes from causing a tag to be lost. This would also allow easier management of tags to see all categories of tasks available.

Rough Timeline

1st December – 15th December : Do some guides on Ruby

15th December – 30th December : Work on assignment

31st December : Mid Assignment Submission

2nd Jan – 12th Jan: Finish up assignment

25th Jan: Final Submission

Use Case

The following are some basic use cases the application will feature

System: Notes

Actor: User

Use Case: UC01 - View notes

1. User launches webapp
2. All notes are displayed

Use case ends.

Extensions:

2a. User requests to see labels used on a note

2b. System displays all labels on specific note

System: Notes

Actor: User

Use Case: UC02 - View Labels

1. User launches webapp
2. All labels are displayed

Use case ends.

System: Notes

Actor: User

Use Case: UC03 - View notes by filtering by label

1. User selects label to filter notes by
2. All notes with related tags are displayed

Use case ends.

System: Notes
Actor: User
Use Case: UC11 - Add notes

1. User chooses to add a new note
2. System requests for details of new note
3. System stores new note and displays all existing notes

Use case ends.

Extensions:

2a. System detects missing value in entered note details
2a1. System requests for correct data
2a2. User enters new data
Steps 2a1-2a2 are repeated until the data entered are correct.
Use case resumes from step 3.

System: Notes
Actor: User
Use Case: UC15 - tag notes

1. User chooses note to tag
2. System provides available tags for user to choose from
3. User chooses tag
4. System stores tag note relationship and updates display

Extensions:

2a. User wants to tag note with a tag that is not available
2a1. User creates a new tag (UC21)

System: Notes
Actor: User
Use Case: UC21 - create tag

1. User chooses to create a new tag
2. System requests for tag text
3. User provides tag
4. System adds tag and updates display

Problems currently being faced:

Currently there is some issue regarding CORS (Cross Origin Resource Sharing), which a current workaround has been to use foreman to run the Rails server on a specific port eg 3001. And configure the React side to accept the port as a proxy during development.