Gan Wan Cheng Isaac (A0204718H)

CVWO Mid Assignment

Use Cases

Feature	Description
Create Task	Precondition: None Post condition: New Task is added to the list of tasks Basic Flow: - Use case begins when user clicks create button - User fills up form with Title, Content of the Task and submits form - Use case ends when model validates that title is not null and persists entry to database
Read Task	Precondition: None Post condition: User able to view task with its description Basic Flow: - Use case begins when user clicks view button - Use case ends when user is able to view task with its associated content
Update Task	Precondition: Task must already exist in the database Post condition: Title and contents of the task is altered using a PATCH request Basic Flow: - Use case begins when user clicks edit button - User is shown a form with the current entry in the database for alteration - Use case ends when model validates that title is not null and persists entry to database
Delete Task	Precondition: Task must already exist in the database Post condition: Title and contents of the Task is removed from the DB Basic Flow: - Use case begins when user clicks the delete button of the task - User is given a prompt to confirm action - Use case ends when user confirms prompt and entry is removed from DB
Check Task	Precondition: Task must already exist in the database Post condition : Task font is replaced with a strike through and task moves to the bottom Basic flow: - Use case begins begins when user clicks the check button next to the task - Boolean value of the status attribute of task is switched to True in the database - Use case ends when font of title is strikethrough and task moves to bottom of the list of tasks
Creating or	Precondition: There must at least be a task which exist

adding Tag to Task	Post condition: Task has a tag added to it Basic Flow: - Use case begins when the user clicks add tag button of each task - User adds a tag or chooses an existing tag to be added to task - Use case ends when model validates that tag is not null and persist tag to database
Delete Tag	Pre condition: Tag must exist Post condition: Tag is removed from all the post with the Tag Basic flow: - Use case begins when use clicks the x button of the tag - User given prompt to confirm action - Use case ends when user confirms prompt and tag is removed from DB
Query Tasks associated with Tags	Precondition: Tags and post must exists and they must have a relation Post condition: Users able to view list of post associated with tag Basic flow: - Use case begins when user clicks on particular tag in the tags view - Tasks with particular tag are listed

Execution Plan

1. Conceptualizing design and layout of the application
    - Sketch basic wire frame of the application to give an insight of the overall structure, functionalities and the flow between different components
    - This will also give me an idea as to what kind of data to persist to the database
2. Getting familiar with the rails framework and writing basic functionalities such as CRUD operations and task completion
    - This phase involves designing the DB schema and then writing basic functionalities and rendering it using html erb to figure out how they interact with the DB
3. Get familiar with react and build a rough mockup of the app
    - This phase involves creating a mockup with rough design elements and integrating existing libraries and components for styling
4. Learning how to create the API for the database and query API for data on React view
    - API would be used to funnel data from the database to the frontend
    - Some tinkering to learn how to tie together event triggers in react and controller functionalities
5. Deployment
    - Attempt to deploy the app so that if could be accessed from anywhere.
6. Attempt the other challenges if time permits

Challenges

- I have little experience with web development and have to pick up Rails and React.
- HABTM vs HMT for the joint table. Though I do no foresee any overlapping data between the tasks table and the tags table, I decide to err on the side of caution go with HMT since I also read that implementing an explicit model for the joint table is a bit more costly/complex than just modeling it out of the box as a model (using has_many_through) and