Name: Daniel Lim Wei En
Matriculation Number: A0233730N

1. Use Cases for Application

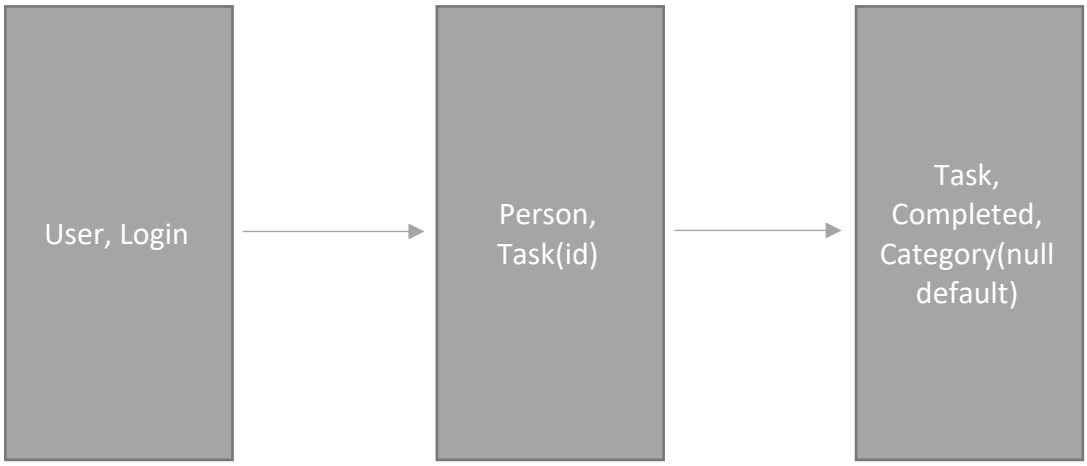
Purpose of the Application:
- A task management web application for individual users to manage tasks that they may have.

What the application does:
(To-do List)
- At its core, the web application serves to manage tasks. Therefore, the most basic feature that it will have, is a general to-do list feature that displays all the tasks of the current user in added order.
- The to-do list feature will support basic CRUD (Create, Read, Update and Delete) operations.
    - (Create) It will allow users to add new tasks to their to do list.
    - (Read) The current users' tasks are by default, read and displayed to the user in the order that they added the tasks.
    - (Update) It will allow users to update existing tasks and change the contents of the task or to mark tasks as complete.
    - (Delete) It will allow users to delete their existing tasks once they are complete or to remove tasks that they would like removed.
- I intend to use PostGreSQL / SQLite3 as the relational database for this application to handle database queries and storing the necessary data for the to-do list.

(Categorization Options)
- On top of just being a simple to-do list, the web application must also have the option for users to organize their tasks into different categories. This is a basic requirement expected of most task management applications today.
- We can handle categorization using relational databases. The to-do lists are keys to separate table holding all the users' tasks and each task can be categorized.
- Separate tables can be used in the web application on the frontend for displaying these tasks in different categories so that users can more easily search for their tasks from their own categories.
- And at the most basic, tasks will be categorized into completed and existing tasks with options for users to further categorize tasks.

(Login System) (Optional, should I have time)
- The website should have some user authentication system. This is to allow different users to make use of the application and so that they can each have their own personal to-do lists on the web application.
- Users can register for accounts and login to see their own personal to-do lists and manage their own tasks and only their own personal tasks, allowing different users to make use of the app on their own.
- Ruby on Rails has functionality that will allow us to do this.

Below is a mock-up of the relational databases to be used in this application.
(Placeholder, Very Simplified Relations)



2. Technologies to Utilize

Frontend: React.js
- I intend to learn and utilize React.js for the frontend as it is a popular framework for building User Interfaces.

Backend: Ruby on Rails
- I intend to use Ruby on Rails for the backend/server/controller portion of the web application due to the added functionality that comes with Rails for building web applications and due to the time and difficulty constraints of using Go for the backend.

Database: PostGreSQL/SQLite3
- I intend to learn and use PostGreSQL/SQLite3 as the database/model for the web application.

3. Timeline for Implementation

Level 1: Planning and Preparing (1 ~ 2 Days)
- Drafted a plan for learning the required technologies and for implementing the actual web application.

Level 2: Getting Started (1 ~ 2 Weeks)
- Learning the required technologies takes time and getting used to the new frameworks and technologies requires some effort.
- The order that I intend to learn these technologies in is as follows.
    - Git and GitHub (Necessary for collaboration, good to start early, especially if I have minimal knowledge. Version Control)
    - React.js (Start with the Frontend framework as I have come from CS1101S so JavaScript is the most familiar language to me and React.js is in JavaScript. Frontend Framework, most tangible results)
    - JavaScript/TypeScript (Already possess knowledge of JavaScript so adding TypeScript and learning it can help reduce type errors as well as improve code.)

    - Ruby/Ruby on Rails (Move on to the backend framework. Learn Ruby first by getting used to the syntax and its intricacies, then move on to learn Ruby on Rails. Backend Framework)
    - Relational Databases (Learn SQL, how databases work and PostGreSQL/SQLite3. Model)
    - HTML/CSS and RESTful APIs (These are generally easier so they can be learnt as I am learning React.js for the frontend)
    - Testing knowledge on Sample React application.

Level 3/4: Implementing and Deploying the Web Application (2 ~ 4 Weeks)

- This is where the actual implementation takes place. This happens concurrently with learning the new frameworks as the Level 2 above, while good for introducing me to the technologies, will not guarantee me mastery over it. So, this is where the skills will be put to the test and where more learning takes place.
- Prepare the necessary documents for submissions.