Name: Ang Swee Chow

Matric No: A0217005W

Final solution presented would be a browser-based todo manager with features implemented with the intent for general professional use.

Basic Use Cases

1. Todo manager is tailored for professional use as well as for students to track, plan and visualize their tasks
2. Todo manager can be used to track tasks to be done, with or without deadlines
3. Sub-tasks can be created under their respective main tasks, so as to track the various requirements for main task completion
4. Todo manager can be used to visualize tasks based on categories, user-prescribed priority as well as time left to deadlines
5. Todo manager has a search function for users to search for tasks by task names. Further development would be done to implement advanced search functions with multiple search requirements (eg searching both by priority, deadline and category)
6. Each user's various todo list will be automatically synced to the online database such that it can be retrieved, updated and modified on many devices, allowing for easy cross-device usage
7. Further development would be attempted to allow for organisation groups to be formed, to create common todo lists, allow for users to automatically notify others in the same organisation when their personal tasks are completed, request other users for follow-up, and to allow supervisors/superiors/teachers to automatically push tasks to their respective members.

Execution Plan

Development would be split into 3 separate milestones: Minimum product, push to production and additional development.

- Minimum Product

  Features:

  1. Basic todo manager with one todo list per user
  2. Users would be able to categorise and search/sort tasks based on task categories/names
  3. Basic username/encrypted password pair authentication
  4. Deadline setting for various tasks using time-to-task or date-time setting
  5. Responsive web design

  Plan:

  1. Learn various skills required: Ruby, HTML, CSS, Javascript, React and Git, Typescript

2. Model component to use Ruby on Rails, Controller component to use Ruby on Rails. View component to be React, with usage of HTML, CSS and TypeScript
3. Create database and tables required for basic todo list task creation, as well as user authentication

Table	Fields
Users	Username, Password, Email
Task Lists	Username, Task List ID, List Name
Tasks	Task List ID, Task Name, Category, Completion Status, Deadline
	(Primary keys are underlined)

4. Implement authentication landing page to allow for users to register with their email, username and a password that would be encrypted
   a. Model component to handle user creation and read
   b. Controller component to handle requests
   c. View component to handle client view of authentication page
5. Implement todo list page
   a. Model component to handle CRUD (Create, Read, Update, Delete) operations
   b. Controller component to handle requests
   c. View component to handle client view.
      Head would be a toolbar for logging out and future feature implementations
      Body would be a basic todo list listing task name, category and deadline, with a checkbox on the left of each task for users to indicate completion
      2 todo list, one for completed and one for uncompleted tasks. When user marks tasks as completed, the tasks will be sent to the completed tasks list after page refresh. A modify button would be on the right side of respective tasks.
      Button for creation of task would be at the upper right of the body component.
      Modal component would be used when users intend to modify and create new tasks.

- Push to production
  1. Implement Cron to allow for daily scheduled backups
  2. Publish solution to Heroku and debug should any issues occur when migrating to online realm
  3. Implement styling for view component using CSS
  4. Implement search component which is a circular button with a magnifying glass depicted in it next to the create task button. When clicked, a modal component would appear to allow users to search by task name/category

    5. OAuth authentication implementation for users to authenticate using Gmail accounts
    6. Priority component to be added for users to specify the priority of tasks and to highlight of priority levels by color
    7. Sorting feature for users to sort by category, task creation date, task name and time to deadline
    8. Implement subtasks
    9. Allow for multiple todo list per user. Each todo list will still have 2 separate lists, one for completed and one for incomplete tasks
    10. Testing to be done after every step to ensure that the product works in production. Also push solution to production after every step.
- Additional Development
    1. Attempt other given optional tasks before delving further
    2. Implement search feature to allow for advanced search based on multiple search terms (eg search based on task name, category and priority)
    3. Implement permissions to allow for other registered users to view, modify, complete, add tasks etc based on permissions given, and permission for other users to notify them on task completion
    4. Allow for the creation of notification groups by users, to be tagged to tasks to notify others on task completion for follow up
    5. Implement organisational groups to allow for common todo list
    6. Allow for supervisors within organisational groups to push new tasks to their respective sphere of influence
    7. Testing to be done after every step to ensure that the product works in production. Also push solution to production after every step.

Challenges Presented

Concepts such as the MVC framework and other conventions used in Ruby on Rails development posed as a difficulty in attempting this assignment. MVC was particularly hard to adapt to as previous work was done using the frontend/backend framework instead, where frontend handled user view and backend handled everything else and was not separated into view and controller components. This was the first time I used React as previous frontend consisted of HTML and CSS, which was manipulated by Python using Flask.