Application: To-Do-Manager

In our busy lives, it is easy for chores to pile up while we quickly lose track of them. Among the many facets of technology, unfortunately even distraction is counted. Given this world, the need for an application that helps manage and plan out the tasks one has is imperative. This application would serve to help users store and track their tasks and also provide syncing and sharing facilities to ensure optimal productivity. Beyond the basic CRUD (Create-Read-Update-Delete) paradigm, effective user-friendly features (search, tags, scheduling, syncing etc.) supplement these efforts and seek to render this app an essential utility tool for those conscious about their productivity.

Use cases and features

Basic Layer

- CRUD functionality – standard implementation on React at the frontend and Ruby-on-Rails at the backend,
- *Tags* - (think Categories)
  - Depending on user discretion, may be added to a *Task* for easier classification. Selecting a *Tag* would let the user view all the *Tasks* tagged with it.
  - Grouping occurs through classifying the relevant database records similarly and collating their addresses under the tag (done using Ruby-on-Rails).
  - A *Task* can have multiple *tags*, for example: "Buying bread" is a *Task* that can be associated both to "*Groceries" and "Making a Toast"*.

Advanced Layer[1]

- *Tags*[2]
  - Hierarchical tags, i.e., categories within categories
  - Analogous to file explorer – folders within folders exist. Similarly, all subtasks of a subtask of a *Task* could be grouped under the parent subtask header.
- Schedule / *Time-tracking*
  - Enable users to assign deadline times for specific tasks
  - Syncing with event calendars using .ics event files, as well as linking to Google/Outlook/Apple Calendar
  - Setting up reminders (frequency and timing adjusted by the user)
  - Tracking time spent by a user on one Task
- Sort
  - Sorting *Tasks* by
    1. Title (Alphabetically ascending or descending)
    2. Deadline (Earliest to Latest or vice-versa)
    3. User provided *Ranks*
- Search
  - Case-insensitive search on
    1. Titles

[1] These features, while conceptualized for demonstrative purposes, may not be included in the final build
[2] Continuation of the Tags feature

2. Text-Search on the contents of the Task, keyword based[3]

- *Progress Bar*
  - Percentage of tasks completed, as well as number of tasks completed out of total tasks – all represented numerically as well as through a graphical bar.

Execution Plan

- Building on a Ruby-on-Rails framework using TypeScript – utilizing the React library for interface design.
- Implementing automated actions using Cron[4]
  - Giving reminders to users as specified (for deadlines etc)
  - Backup and syncing processes automated at a frequency set by the user
- Hosting
  - Using Heroku to create a completely web-based instance of the application to be accessed by browsers.
- RESTful API structure[5] to bridge the TypeScript built interface on the Ruby-on-Rails framework.

Progress Report

- Developing the frontend forms and interfaces required using React
  - Considering different design templates for the skin of the app
  - Experimenting with Material UI
- Parallelly developing the backend on Rails
  - Had issues integrating React to Rails, resolved using nonstandard edits to `webpacker.yml` file used to read .js files on the Puma Browser
  - Have created a demo of Active Records that are analogous to Tasks in the To-Do List application. However, the Updating (U of CRUD) functionality is not adequately rendered yet, and instead creates a mirrored record with the changes recorded.
- Having issues integrating the two – Initially attempted to start from the backend and build to the frontend but hit many technical snags which led to consideration of independent building.

[3] Possibly difficult to implement using basic techniques within Ruby-on-rails and Typescript, based on the database size.

[4] Attempting to make it platform independent, and browser-sufficient

[5] Implementation contingent on requirement – may use alternate methods to integrate backend and frontend.