# CVWO Project - Todo Manager

**Task 1 – Managing Tasks**

Basic use cases: To create, read, update and delete (CRUD) tasks from list of task

- CREATE: Add task
- READ: Read full title and description of task (likely in a pop-up)
- UPDATE: Edit title and/or description of task (could be done together with READ)
- DELETE: Delete task

Execution Plan:

1. Create a RESTFUL API server-side/backend for Ruby on Rails, and client-side/frontend for React
2. Setup CRUD using MVC (Model, View and Controller) for Rails, and change state in React to create single-page application
    a. MODEL: Include fields for database and their datatypes for Rails:
        i. title:string
        ii. (category:string) // for Task 2
        iii. description:text // can be empty
        iv. timeCreated:datetime // keep track of when it was first created
        v. timeUpdated:datetime // keep track of when it was last modified

    b. VIEW: Include Components for React part
        i. App.jsx – main component containing header, body and footer (as template)
        ii. body.jsx – body component containing all methods and controls the components and passed in props (includes DELETE method)
        iii. show.jsx – show all tasks *(Consideration: add new tasks to the top? or keep indexing? sorting method of tasks?)*
        iv. newtask.jsx – CREATE form to add new task to database
        v. task.jsx – show (READ) a particular task (likely using a modal) and can be edited (UPDATE) if necessary

    c. CONTROLLER: define CRUD methods for controlling/querying database/model
        i. def index – show all tasks in API
        ii. def create – add a task and show it in API
        iii. def update – update all attributes/parameters of task in API
        iv. def destroy – delete task in API

**Task 2 – Categorising Tasks**

Basic use cases: Filter through tasks into categories to find the task that the user wants easily.

Execution Plan:

1. Include the field "Category" in Model and View as part of the task (likely as a dropdown that allows addition)
2. Add filtering method of index with all tasks, to those of a certain category
3. (Optional) Add field for "Sub-categories" for multiple other categories for more concise searching
3. (Optional) Create other filtering/sorting system (ascending/descending order)
    a. Sort by Name (Alphabetical order)
    b. Sort by Last Modified
    c. Sort by Date Created
4. (Optional) Create search system to search for task
    a. Search by Title
    b. Search by Description
    c. Search by Category
    d. Search all
5. (Optional) Add a field for due date and time (deadline for tasks)
6. (Optional) Setup and run server in Heroku

**Suggestions and Problems Faced:**

- While the task looks simple and is the basis of CRUD applications, many details (perhaps features) were left out, such as how the tasks are supposed to be sorted, or whether the Todo Manager is supposed to run on a single route or use different routes (single-page application may rely more on React frontend, but it is also possible to have different routes using Rails and vanilla CSS/HTML, as React could just introduce more gems/packs and increase storage.)

- It was difficult to find online resources to integrate React with Rails, as most of them are either Rails-focused or React-focused. As I have a bit of exposure in React, it felt that it was most certainly possible to create a CRUD architecture easily with only React OR only in Rails (by generating a scaffold)

- In the end, I decided to use Rails as an API database and use GET/POST/PUT/DELETE AJAX requests by React to implement CRUD, so that the overall application will be more frontend-heavy (since I have more familiarity with React)

Additional dependencies that may be used:

- semantic-ui
- react_on_rails
- axios (for asynchronous HTTP calls)