Coding Assignment

Purpose:

Purpose of this coding assignment is to understand candidates' problem solving capabilities and to evaluate his/her front-end development capabilities.

Technology Stack:

React.js/Next.js with Typescript

Guidelines:

- Disclosure of this assignment and your submission is prohibited.
- There is no time limit for completing this assignment.
- Send your assignment in a compressed file (archived by ZIP, tar.gz etc.) by email or share it via a file sharing service such as Google Drive.
  - In case of use of a file sharing service, make sure you set the correct access rights so that it can be downloaded.
  - Please make sure to delete the node modules folder before compressing the file.

Note:

Simple utility libraries can be used to solve specific problems, however, please refrain from using frameworks that provide full or partial solutions to this challenge.

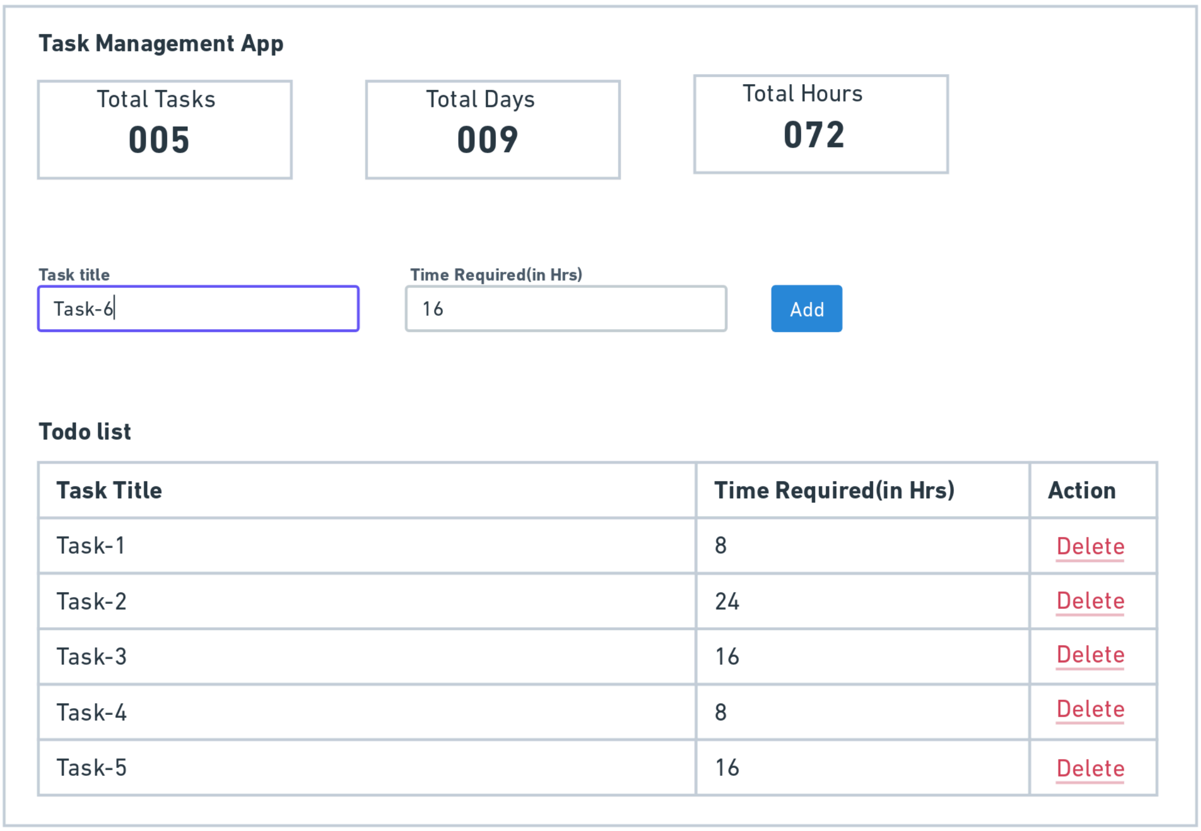
Problem Definition:

In the bustling town of Moneyville, a team of ambitious developers and designers were working tirelessly to create innovative solutions for task management. They understood the importance of efficient task tracking and collaboration, which led them to envision a cutting-edge Task Management System (TMS). The TMS aimed to simplify the lives of individuals and teams by providing a seamless and intuitive platform to manage their daily tasks. With a strong focus on user experience, the team set out to build a user-friendly UI that would revolutionize the way people organized their work.

Despite the team's determination and vision, they faced a significant challenge in developing the TMS. Due to budget constraints and time limitations, they were unable to allocate resources to build a backend system for data storage and retrieval. This posed a problem as they needed to store and manage task-related information such as titles, remaining hours. However, the team's determination to deliver a functional product remained unshaken.

To overcome this hurdle, the team decided to adopt an innovative approach. They would leverage the capabilities of React.js/Next.js with Typescript to create an in-memory data storage solution. By storing all the task data within the UI itself, they would eliminate the need for a separate backend system, enabling users to enjoy a seamless task management experience.

As a frontend engineer, you need to implement the UI of the task management app based on the following requirements.
An example of what the UI should look like is as follows.



Task Management App

Total Tasks	Total Days	Total Hours
005	009	072

Task title: Task-6

Time Required(in Hrs): 16

Todo list

Task Title	Time Required(in Hrs)	Action
Task-1	8	Delete
Task-2	24	Delete
Task-3	16	Delete
Task-4	8	Delete
Task-5	16	Delete

Requirements for Task Management App:

1. On **Add** button click
   - Validate : Task title
     - Task Title and time required should not be empty
     - Character length should not exceed 128 characters.
   - Validate : Time Required
     - Allow only numeric values input
     - Rage should be between 0-24
   - On validate failure : Display appropriate error message in modal dialog
   - On validation success
     - Add task in the Todo list
     - Update dashboard
       - Increase “Total Task” counter by 1
       - Recalculate “Total Days” counter based on total of Time Required for each task (1 day = 8hrs, total days can be in decimal points up to 2 decimal digits)
       - Recalculate “Total Hours” counter to include hours of newly added task

2. On **Delete** button click
    - Display a confirmation modal dialog with appropriate message
        - On OK button click
            - Delete the selected task from “Todo list”
            - Update dashboard
                - Reduce “Total Task” by 1
                - Recalculate “Total Days” counter to reduce the hours for deleted task
                - Recalculate “Total Hours” counter to reduce the hours for deleted task