Task management - take-home challenge

Overview

You are tasked with building a task management platform with two main routes:

1. **Table View Page:** A feature-rich table displaying tasks fetched from an API.
2. **Dashboard Page:** A page showcasing three graphs based on the task data.

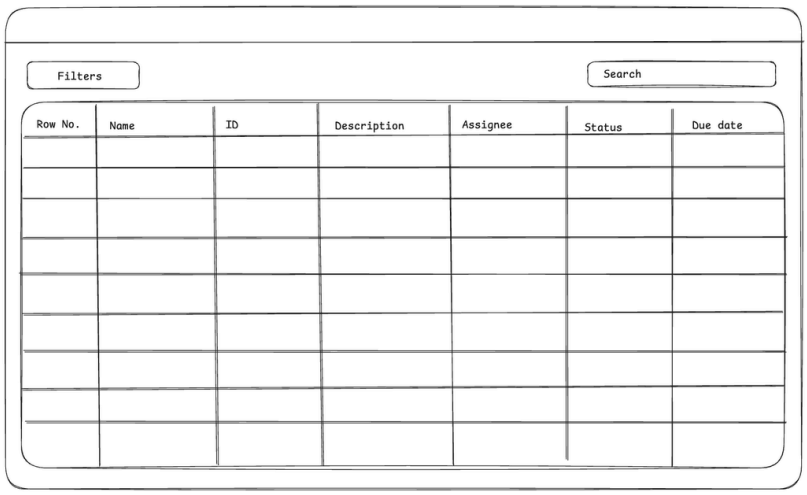
This project must be built using **ReactJS/NextJS**, **Redux** for state management, and **styled-components** for styling. You are free to use any UI component library **except for the Table component**. This ensures that you design and build the table UI yourself.

Technical Requirements

- **Framework:** Any ReactJS framework (e.g. CRA, NextJS, etc)
- **State Management:** Redux
- **Styling:** styled-components
- **UI Libraries (optional):** Any UI component library (e.g., Material-UI, Ant Design, Chakra UI, etc.) **except for the Table component**. You must build your own table.

Routes and Functionality

1. Table View Page



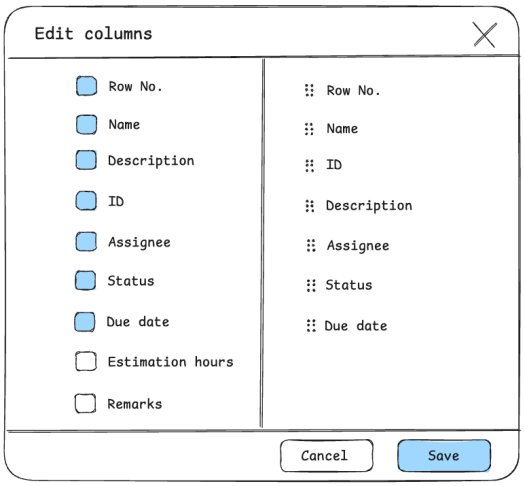
Route: `/table`

Features:

- **Data Fetching:**
  - Create a simple server and make an API that will give the list of tasks in the response.
  - The list will be dummy data that you can make/generate of yourself.
  - Fetch the task data from that API endpoint.
  - Handle loading and error states.
- **Table Implementation:**
  - Build your own table component (do not use a pre-built table component from a UI library).
  - Each row should display summary information of a task (e.g., title, status, due date, etc.).
- **Filtering, Sorting & Search:**
  - **Filtering:** Allow users to filter tasks by status, priority, and assignees. (add priority column also)
  - **Sorting:** Enable sorting for multiple columns.
  - **Search:** Implement a search bar to filter tasks based on keywords. The user should be able to search in the name and description.
- **Infinite Loading:**
  - As the user scrolls, load additional data.
- **Drawer Interaction:**
  - When a row is clicked, open a side drawer.
  - The drawer should slide in from the side and display all the details of the task.
  - Include a section for **comments** related to the task. This can be a display only, no need to add any input field.

Edit Columns:

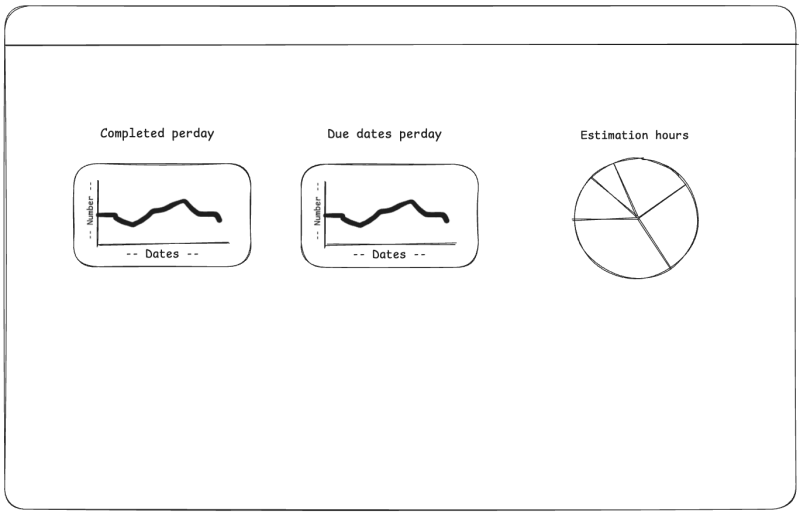
Add an ‘Edit Columns’ button on the table view page. The modal will allow show, hide, and re-arrange the columns. The config should persist on refresh. The design will be like the below:



UX/UI Considerations:

- Provide visual feedback for loading, errors, and empty states.
- The table should be easy to navigate with clear visual hierarchies and actions.
- Ensure accessibility (keyboard navigation for the table rows and drawer, appropriate ARIA roles, etc.).

2. Dashboard Page



Route: `/dashboard`

Features:

This must use the same dataset of the table view to generate the analytics.

- **Graph 1: Completed per Day (Line Chart)**
  - Display a line chart showing the number of tasks marked as completed each day.
  - X-axis: Days; Y-axis: Count of completed tasks.
- **Graph 2: Due Date per Day (Line Chart)**
  - Display a line chart showing the number of tasks with due dates on each day.
  - X-axis: Days; Y-axis: Count of tasks due.
- **Graph 3: Estimation Hours (Pie Chart)**
  - Display a pie chart illustrating the breakdown of tasks based on estimated hours.
  - Each slice of the pie represents a category or range of estimation hours.

Graph Data:

- You may use the same API data (processed accordingly) or assume separate endpoints for dashboard metrics.
- Provide error handling and loading states for the graphs.

Filters:

- Add filters in the dashboard.
- This should use the same filter component that is in the table view.

UX/UI Considerations:

- Ensure the graphs are responsive and interactive (hover states, tooltips, etc.).
- Maintain a consistent color palette and typography throughout the dashboard.

Evaluation Criteria

- Code quality
- Responsiveness
- Performance and optimizations
- Design and UI/UX

- Error handling and validations
- Robust (bug-free)
- Accessibility

Bonus Points

- **Unit & Integration Tests:** Write tests for components (using Jest, React Testing Library, etc.).
- **Theming:** Implement a theme for the whole platform.
- **Performance Optimization:** Optimize table rendering for large data sets (like virtualization).

Deliverables

- Source Code
- HLD (optional)
- Demo

Good luck with your implementation! This assignment is designed to assess your ability to convert your skills into deliverables and how you make a real-world product with the right practices. If you have any further questions or need clarification, please let us know.