PROJECT REPORT

Report: Todo List Application Design and Implementation

1. Overview

This project is a simple **Todo List application** built using **React.js**. The app allows users to add, edit, delete, and mark tasks as completed. The data persists using **localStorage**, so tasks are saved even when the page is refreshed. The app uses **Tailwind CSS** for styling and leverages a modern, minimalist design inspired by the **dark theme** shown in the referenced image.

2. Core Features

2.1 Add New Tasks

- Users can add new tasks by typing into the input box and clicking the "Submit" button.
- Validation is implemented to disable submission if the input length is less than or equal to 3 characters.

2.2 Edit Existing Tasks

- Users can edit any task by clicking the **edit icon** (using **react-icons**).
- The task is removed from the list and placed back into the input field, where it can be modified and resubmitted.

2.3 Delete Tasks

- Tasks can be deleted using the **delete icon**.
- Once deleted, tasks are removed both from the UI and **localStorage**.

2.4 Mark as Completed

- Users can mark tasks as completed using a checkbox. Completed tasks will have a **strikethrough** effect.
- Completed tasks remain in the list but can be filtered to show only active tasks using the "Show Finished" checkbox.

2.5 Persistent Data Storage

- The app uses **localStorage** to persist task data. This means tasks remain available even after refreshing the page or closing the browser.

3. Design and Styling

3.1 General Layout

- The app is designed with **responsive, mobile-first** principles.
- It is centered both horizontally and vertically using **flexbox** to ensure it looks good on any screen size.
- The entire UI has a minimalist look with **rounded corners**, consistent **padding**, and **hover effects** for better user interaction.

3.2 Color Palette

- The design uses a **blue-based dark theme**, similar to the provided reference image.
    - **Background color:** bg-blue-950 (very dark blue, almost black).
    - **Primary container color:** bg-blue-400 for the card.
    - **Todo items:** bg-blue-600 with text in white (text-white).
    - **Button hover effects** are implemented to change colors when hovered to provide a visual response to user actions.

3.3 Components and Styling

- **Header**: A bold, white text Todo List header is centered at the top to provide context.
- **Search Bar**: Styled as a rounded input box, similar to the reference design, though it is non-functional in this version.
- **Todo Items**: Displayed as rows in a card-like format, with checkboxes to mark them as completed.
- **Form Section**: Includes the input field for adding tasks, which also has rounded corners and a shadow effect.

4. Technical Details

4.1 React Hooks

- **useState**: This is used to manage the todo list (todos), the input field value (todo), and the visibility toggle (showFinished).
- **useEffect**: This is used to load todos from localStorage when the app first renders.

4.2 Data Persistence with localStorage

- Tasks are saved and loaded from **localStorage**.
- When a task is added, edited, or deleted, the updated list is saved back to localStorage.

4.3 Icons

- **react-icons** is used for the edit (FaEdit) and delete (AiFillDelete) icons, providing a clean and consistent design.

4.4 Tailwind CSS

- The app is styled entirely using **Tailwind CSS**, allowing for fast and efficient styling without needing external stylesheets.
  - **Responsive Design**: Tailwind's responsive utilities are used to ensure the layout works across devices.
  - **Custom Styling**: Classes like rounded-lg, bg-blue-950, shadow-lg, etc., are used to give a modern, clean design.

5. Code Structure

The main functionality is encapsulated in the App component. Below is the high-level structure:

- **State Management**: Handles the todos array and the current input value.
- **Effect Hook**: Loads tasks from localStorage on initial render.
- **Event Handlers**:
  - handleAdd: Adds a new task to the list.
  - handleEdit: Edits an existing task.
  - handleDelete: Removes a task from the list.
  - handleCheckbox: Marks tasks as complete or incomplete.
  - toggleFinished: Toggles visibility of completed tasks.

6. Improvements & Future Enhancements

6.1 Search Functionality

- The current implementation has a search bar for aesthetic purposes only. A future enhancement could involve implementing real-time filtering of tasks based on user input in the search bar.

6.2 Drag-and-Drop for Task Reordering

- Implementing drag-and-drop functionality would allow users to reorder their tasks dynamically.

6.3 Subtasks

- Users could benefit from adding subtasks to each main task for better task management.

6.4 Animations

- Adding subtle animations (e.g., when tasks are added, edited, or removed) would enhance the user experience.

7. Conclusion

This **Todo List React App** serves as a functional and user-friendly task manager with a clean, modern design. It effectively combines **React's state management** with **localStorage** for persistent data storage and utilizes **Tailwind CSS** to ensure a responsive and aesthetically pleasing user interface.