QuickTick
Simple Todo App with Undo Functionality

By
Amaan
BML Munjal University

Abstract

This project presents a Simple Todo App with Undo Functionality, built using React and JavaScript . It allows users to add, remove, and complete tasks, with an undo option to revert actions within five seconds . A two-column layout organizes tasks, and a confetti effect enhances user engagement. The app leverages React's useState hook or state management and react-confetti or animations. Key challenges include efficient state handling and seamless task movement. Future improvements include persistent storage, drag-and-drop support, and mobile optimization. This project emphasizes interactive and user-friendly task management.

Introduction

For this assignment, I developed a Simple Todo App with Undo Functionality using React and JavaScript. The app allows users to add, remove, and mark tasks as completed. Additionally, it includes an undo feature that lets users revert the removal or completion of a task within 5 seconds. To make the app more engaging, I added a confetti celebration effect whenever a task is marked as completed.

Design and Features

1. Task Management

The app provides the following core functionalities:

- Add a Task : Users can add new tasks by typing in the input field and clicking the "Add Task" button.
- Remove a Task : Users can remove tasks by clicking the "Remove" button next to each task.
- Mark as Completed : Users can mark tasks as completed by checking the checkbox next to each task. Completed tasks are moved to the "Completed Tasks" column.

2. Undo Functionality

The app includes an undo feature that allows users to revert the most recent action (removal or completion) within 5 seconds. A notification appears at the bottom-right corner of the screen with an "Undo" button. If the user clicks the button within 5 seconds, the action is reverted.

3. Two-Column Layout (Additional)

The app uses a two-column layout :

- Add a Task: This column displays tasks that are yet to be completed.

- Completed Tasks: This column displays tasks that have been marked as completed.

This layout makes it easy for users to visualize their progress and manage their tasks effectively.

4. Confetti Celebration (Additional)
To make the app more engaging, I added a confetti effect that appears for 3 seconds whenever a task is marked as completed. This provides positive feedback and makes the experience more enjoyable.

Methodology

1. Technologies Used
- React: A JavaScript library for building user interfaces. React's component-based architecture made it easy to organize the app into reusable components.
- JavaScript: Used for implementing the app's logic, such as adding, removing, and completing tasks.
- CSS: Used for styling the app and creating a modern, intuitive user interface.
- react-confetti: A library used to add the confetti effect when a task is completed.

2. Key Components
The app is divided into three main components:
- App.js: The main component that manages the state and logic for the app. It handles adding, removing, and completing tasks, as well as the undo functionality.
- TodoList.js: Displays the list of tasks and allows users to add new tasks.
- Task.js: Represents a single task with options to remove or mark it as completed.

3. State Management
The app uses React's `useState` hook to manage the following states:
- pendingTasks : An array of tasks that are yet to be completed.
- completedTasks : An array of tasks that have been marked as completed.
- lastAction : Tracks the most recent action (removal or completion) for the undo feature.
- showConfetti : Controls the visibility of the confetti effect.

4. Undo Functionality
- The undo feature is implemented using a combination of `useState` and `setTimeout`:
- When a task is removed or completed, the action is stored in the `lastAction` state.
- A notification with an "Undo" button is displayed for 5 seconds.
- If the user clicks the "Undo" button within 5 seconds, the action is reverted, and the task is restored to its previous state.

5. Confetti Effect
The confetti effect is implemented using the `react-confetti` library:
- When a task is marked as completed, the `showConfetti` state is set to `true`, and the confetti effect is displayed.

- After 3 seconds, the `showConfetti` state is set to `false`, and the confetti effect disappears.

Challenges and Solutions

1. Managing State for Undo Functionality
One of the main challenges was implementing the undo feature. I had to ensure that the app could track the most recent action and revert it within 5 seconds. To solve this, I used the `lastAction` state to store the type of action (remove or complete) and the task involved. A `setTimeout` function was used to clear the `lastAction` state after 5 seconds.
2. Moving Tasks Between Columns
Another challenge was moving tasks between the "Add a Task" and "Completed Tasks" columns. I solved this by maintaining two separate arrays (`pendingTasks` and `completedTasks`) and updating them whenever a task was marked as completed or undone.
3. Adding Confetti Effect
Adding the confetti effect was fun but required careful timing. I used the `react-confetti` library and controlled its visibility using the `showConfetti` state. The effect is displayed for 3 seconds and then automatically hidden.

User Interface and Experience

The app has a modern and intuitive user interface designed to provide a seamless user experience. Key design elements include:

- Two-Column Layout: Makes it easy to visualize pending and completed tasks.
- Dark Blue and Teal Green Color Scheme: Provides a sleek and professional look.
- Light Mint Background: Adds a subtle and calming touch to the app.
- Confetti Effect: Adds a celebratory element to task completion, making the app more engaging.

Conclusion

This assignment was a great learning experience. I was able to implement all the required features, including task management, undo functionality, and a confetti celebration effect. The app is functional, user-friendly, and visually appealing. I enjoyed working on this project and learned a lot about React, state management, and creating engaging user interfaces.

Future Improvements

- Persistent Storage - Store tasks in local storage or a database so they are preserved even after the app is closed or refreshed.
- Drag-and-Drop Functionality - Allow users to drag tasks between the "Add a Task" and "Completed Tasks" columns.

- AI Integration - Use an AI to tell a goal that someone likes to achieve and then it provides the various tasks to do for achieving this goal.

Assignment: Simple Todo App with Undo Functionality
Objective:
Create a simple Todo App that allows users to add, remove, and mark tasks as completed. The app should also include an undo feature, enabling users to revert the removal or completion of a task within 5 seconds.

Requirements:
Task Management:
● Users can add new tasks.
● Users can remove existing tasks.
● Users can mark tasks as completed.
Undo Functionality:
● Users can undo the removal of a task within 5 seconds.
● Users can undo the completion of a task within 5 seconds.
● The undo feature should only be available for the most recent removal or completion action.
In-Memory Storage:
● The app should store tasks in-memory, using a data structure such as an array or object.
● Tasks should be preserved during the app's runtime but will be lost when the app is closed or refreshed.
User Interface:
● The app should have a simple and intuitive UI.
● Tasks should be displayed in a list format.
● Each task should have a checkbox to mark it as completed.
● A notification or alert should be displayed when a task is removed or completed, with an option to undo the action.

Technical Requirements:
● The app should be built using a programming language of your choice (e.g., JavaScript, Python, Java).
● The app should use a suitable framework or library for building the UI (e.g., React, Angular, Vue.js).

Constraints:
● The undo feature should only be available for 5 seconds after removing or completing a task.
● The app should not use any external libraries or APIs for implementing the undo feature.

Deliverables:
● A working Todo App with undo functionality.
● Source code for the app.
● A brief report (1-2 pages) explaining the design and implementation of the undo feature.
Evaluation Criteria:

● Correctness and functionality of the app.
● Implementation of the undo feature for removing and completing tasks.
● Code quality, organization, and readability.
● User interface and user experience.

Submission Guidelines:
Please submit your assignment as a zip file containing the source code, report, and any other relevant files.