**Aim:** Integrating third party components into our application and XAML pages using shared resources.

**Description:**

The aim of this guide is to integrate third-party components into a React Native application, ensuring a more feature-rich and interactive user experience. By managing styles and themes using shared resources (such as centralized style files), we ensure consistency in the UI design and improve maintainability, especially in large-scale applications.

**Objective:**

 To integrate third-party components into a React Native application for enhanced functionality (such as charts, form elements, etc.).

 To centralize styles using shared resources for easy maintenance and consistent styling across the application.

 To ensure cross-platform functionality and performance through React Native's architecture and third-party packages.

The **Todo App** aims to provide users with a simple yet effective way of managing tasks. Key features include:

1. **Task Management**: Users can add, complete, and delete tasks.
2. **Task Progress Visualization**: A **Pie Chart** will show the progress of task completion (completed vs. pending).
3. **User Notifications**: **React Toastify** will display notifications on task additions, deletions, and updates.
4. **Modern UI**: The app uses **Material-UI** for a polished user interface with responsive design and interactive elements.

By integrating third-party components and libraries, the app not only provides essential features like task management but also enhances the user interface and experience.

***The objectives of the project are:***

1. **Integrate third-party components** such as **Pie Chart** and **React Toastify** into a React-based Todo app.
2. **Use Material-UI** to create a modern and responsive user interface for task management.
3. **Show task completion progress** dynamically in a Pie chart using **Chart.js**.
4. **Enable task notifications** for actions such as adding, completing, and deleting tasks, using **React Toastify**.
5. **Provide a user-friendly and efficient way** of managing and visualizing tasks, ensuring a smooth interaction for users.

**Tools Required:**

 **Node.js** for running the React Native CLI and JavaScript code.

 **React Native CLI** for managing the React Native development environment and dependencies.

 **Visual Studio Code** for writing JavaScript/JSX code.

 **Android Studio** or **Xcode** for running the Android or iOS emulator, respectively, or a physical device.

 **npm or yarn** for managing packages.

 **React Native third-party component libraries** such as react-native-paper, react-native-chart-kit, react-navigation, etc.

**Implementation:**

The following steps were followed for the implementation of the Todo App:

1. Setting Up the Project:

* Create a new React project using Create React App.
* Install necessary dependencies:

***npm install @mui/material @mui/icons-material react-chartjs-2 chart.js react-toastify***

* Create components and styles as needed.

2. Creating Core Features:

* Task Management:
  + Implemented the ability to add new tasks, mark them as completed, and delete them. This functionality is handled using React's useState for state management.
* React Toastify:
  + Used for displaying notifications when a task is added, completed, or deleted. Notifications are displayed with custom styles and messages based on the actions.
* Pie Chart for Progress:
  + Integrated a Pie chart using Chart.js and react-chartjs-2. The chart dynamically updates based on the number of completed tasks and pending tasks.
* Material-UI Components:
  + Used Material-UI for building a modern UI, including buttons, icons, and text fields for adding tasks.
* Styling:
  + Custom styles were written in CSS to enhance the user interface with hover effects, modern inputs, and buttons.

3. Components and Structure:

* TodoApp.js: Contains the main logic for adding, completing, deleting tasks, and rendering the pie chart for task progress.
* TodoApp.css: Styles the layout, including the task list, input fields, buttons, and pie chart container.

Example code:

* Task Management:
  + Added new tasks using a text input and updated the task state with the setTasks function.
* Task Completion Toggle:
  + Users can mark tasks as completed, which triggers a re-render and updates the task completion count.
* Pie Chart:
  + The chart reflects the number of completed vs pending tasks.

4. Notification System:

* Integrated React Toastify for task creation, deletion, and status change notifications.

5. Clear Completed Tasks:

* Added functionality to clear completed tasks via a button, which filters out completed tasks from the list.

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>To-Do App</title>

</head>

<body>

  <div id="root"></div>

</body>

</html>

**ProgressChart.js**

import React from 'react';

import { Pie } from 'react-chartjs-2';

import { Chart as ChartJS, ArcElement, Tooltip, Legend } from 'chart.js';

// Register the chart components

ChartJS.register(ArcElement, Tooltip, Legend);

const ProgressChart = ({ completedTasks, totalTasks }) => {

  const remainingTasks = totalTasks - completedTasks;

  const data = {

    labels: ['Completed', 'Remaining'],

    datasets: [

      {

        data: [completedTasks, remainingTasks],

        backgroundColor: ['#4caf50', '#f44336'],

        hoverBackgroundColor: ['#66bb6a', '#e57373'],

      },

    ],

  };

  const options = {

    responsive: true,

    plugins: {

      legend: {

        display: true,

        position: 'bottom',

      },

    },

  };

  return (

    <div style={{ maxWidth: '300px', margin: '20px auto' }}>

      <Pie data={data} options={options} />

    </div>

  );

};

export default ProgressChart;

**TodoApp.css**

body {

    font-family: Arial, sans-serif;

    margin: 0;

    padding: 0;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    background-color: #f0f4f8;

  }

  .todo-container {

    max-width: 600px;

    margin: 40px auto;

    text-align: center;

    background-color: #fff;

    padding: 20px;

    border-radius: 8px;

    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

  }

  .input-container {

    display: flex;

    margin-bottom: 20px;

    background-color: #fafafa;

    padding: 10px;

    border-radius: 4px;

    box-shadow: inset 0 2px 4px rgba(0, 0, 0, 0.1);

  }

  .input-container input {

    flex: 1;

    padding: 8px;

    margin-right: 10px;

    border: 1px solid #ddd;

    border-radius: 4px;

  }

  .input-container button {

    padding: 8px;

    background-color: #4caf50;

    color: white;

    border: none;

    border-radius: 4px;

    cursor: pointer;

  }

  .input-container button:hover {

    background-color: #45a049;

  }

  .filter-buttons {

    margin: 20px 0;

  }

  ul {

    padding: 0;

    list-style: none;

  }

  li {

    padding: 12px;

    margin-bottom: 10px;

    background-color: #f9f9f9;

    border-radius: 4px;

    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

    display: flex;

    justify-content: space-between;

    align-items: center;

    cursor: pointer;

  }

  li.completed {

    text-decoration: line-through;

    color: gray;

  }

  li:hover {

    background-color: #f1f1f1;

  }

  li .MuiSvgIcon-root {

    color: #4caf50;

    margin-left: 10px;

  }

  li .MuiSvgIcon-root:hover {

    color: #45a049;

  }

  .progress-chart {

    margin-top: 20px;

  }

  .MuiIconButton-root {

    color: #e57373;

  }

  .MuiIconButton-root:hover {

    color: #f44336;

  }

**TodoApp.js**

import React, { useState } from 'react';

import { TextField, Button, IconButton, List, ListItem, ListItemText } from '@mui/material';

import { CheckCircle, Delete } from '@mui/icons-material';

import { Pie } from 'react-chartjs-2';

import Chart from 'chart.js/auto';

import { ToastContainer, toast } from 'react-toastify';  // Import Toastify

import 'react-toastify/dist/ReactToastify.css';  // Import styles for Toastify

import './TodoApp.css';

const TodoApp = () => {

  const [tasks, setTasks] = useState([]);

  const [taskInput, setTaskInput] = useState('');

  const [completedTasks, setCompletedTasks] = useState(0);

  const handleAddTask = () => {

    if (taskInput.trim() === '') return;

    const newTask = { id: Date.now(), text: taskInput, completed: false };

    setTasks([...tasks, newTask]);

    setTaskInput('');

    toast.success('Task added successfully!');  // Display success notification

  };

  const toggleTaskCompletion = (taskId) => {

    const updatedTasks = tasks.map((task) => {

      if (task.id === taskId) {

        task.completed = !task.completed;

      }

      return task;

    });

    setTasks(updatedTasks);

    updateCompletedTasks(updatedTasks);

    toast.info('Task status updated!');  // Notify task completion toggle

  };

  const deleteTask = (taskId) => {

    const updatedTasks = tasks.filter((task) => task.id !== taskId);

    setTasks(updatedTasks);

    updateCompletedTasks(updatedTasks);

    toast.error('Task deleted!');  // Notify task deletion

  };

  const updateCompletedTasks = (updatedTasks) => {

    const completedCount = updatedTasks.filter((task) => task.completed).length;

    setCompletedTasks(completedCount);

  };

  const pieChartData = {

    labels: ['Completed', 'Pending'],

    datasets: [

      {

        data: [completedTasks, tasks.length - completedTasks],

        backgroundColor: ['#4caf50', '#f44336'],

        hoverBackgroundColor: ['#388e3c', '#d32f2f'],

      },

    ],

  };

  return (

    <div className="todo-container">

      <h1>Todo App</h1>

      <div className="input-container">

        <TextField

          variant="outlined"

          label="New Task"

          value={taskInput}

          onChange={(e) => setTaskInput(e.target.value)}

          fullWidth

        />

        <Button onClick={handleAddTask} variant="contained" color="primary">

          Add

        </Button>

      </div>

      <List>

        {tasks.map((task) => (

          <ListItem key={task.id} className={`task-item ${task.completed ? 'completed' : ''}`}>

            <ListItemText primary={task.text} />

            <div>

              <IconButton onClick={() => toggleTaskCompletion(task.id)}>

                <CheckCircle />

              </IconButton>

              <IconButton onClick={() => deleteTask(task.id)}>

                <Delete />

              </IconButton>

            </div>

          </ListItem>

        ))}

      </List>

      <div className="progress-chart">

        <h3>Task Completion Progress</h3>

        <Pie data={pieChartData} />

      </div>

      <Button

        variant="contained"

        color="secondary"

        onClick={() => setTasks(tasks.filter((task) => !task.completed))}

        fullWidth

      >

        Clear Completed Tasks

      </Button>

      {/\* Toast Container for Notifications \*/}

      <ToastContainer />

    </div>

  );

};

export default TodoApp;

**App.cs**

.App {

  text-align: center;

}

.App-logo {

  height: 40vmin;

  pointer-events: none;

}

@media (prefers-reduced-motion: no-preference) {

  .App-logo {

    animation: App-logo-spin infinite 20s linear;

  }

}

.App-header {

  background-color: #282c34;

  min-height: 100vh;

  display: flex;

  flex-direction: column;

  align-items: center;

  justify-content: center;

  font-size: calc(10px + 2vmin);

  color: white;

}

.App-link {

  color: #61dafb;

}

@keyframes App-logo-spin {

  from {

    transform: rotate(0deg);

  }

  to {

    transform: rotate(360deg);

  }

}

**App.js**

import React from 'react';

import TodoApp from './components/TodoApp';

function App() {

return (

<div className="App">

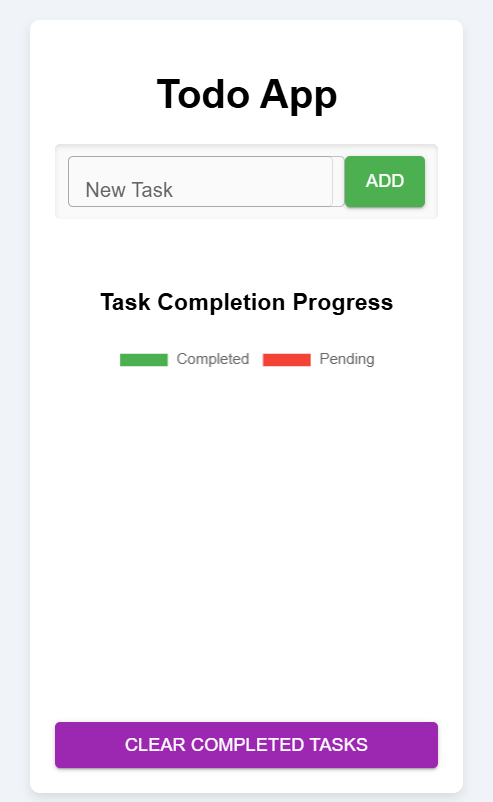
<TodoApp />

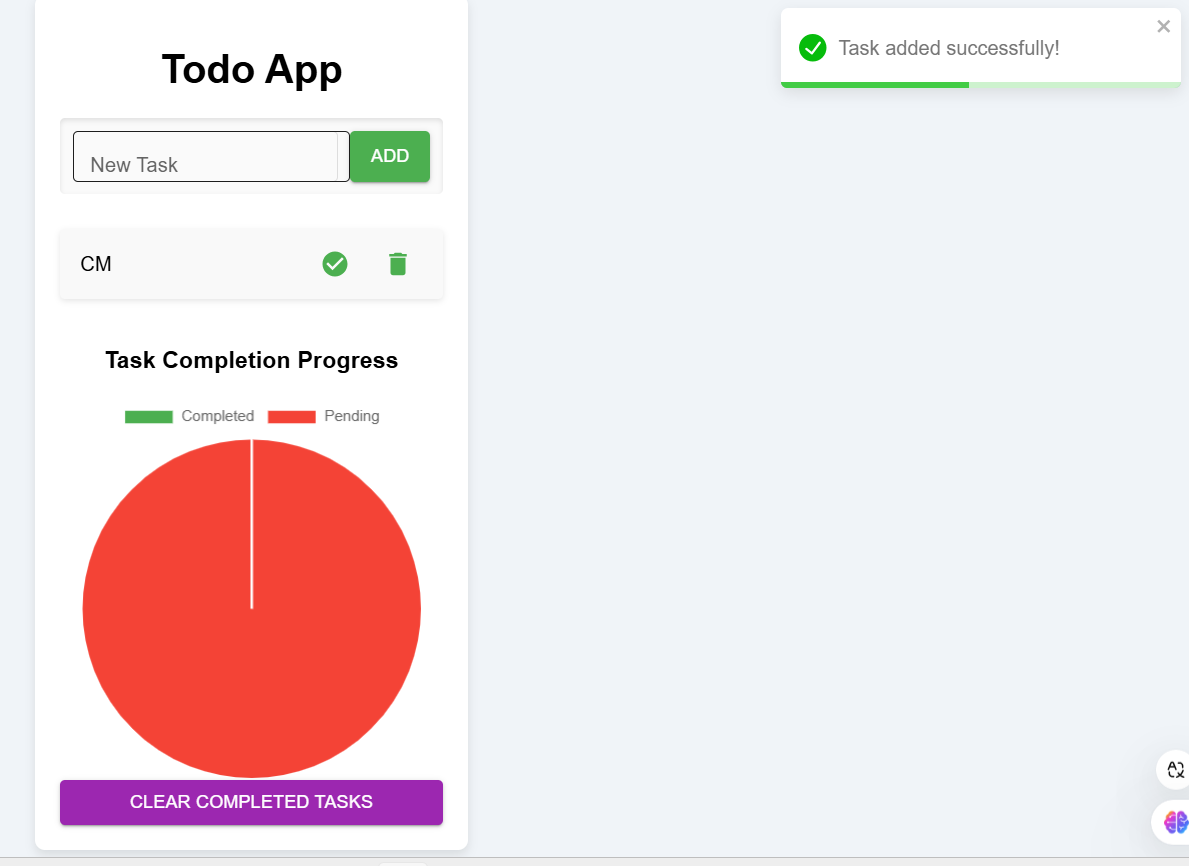
</div>

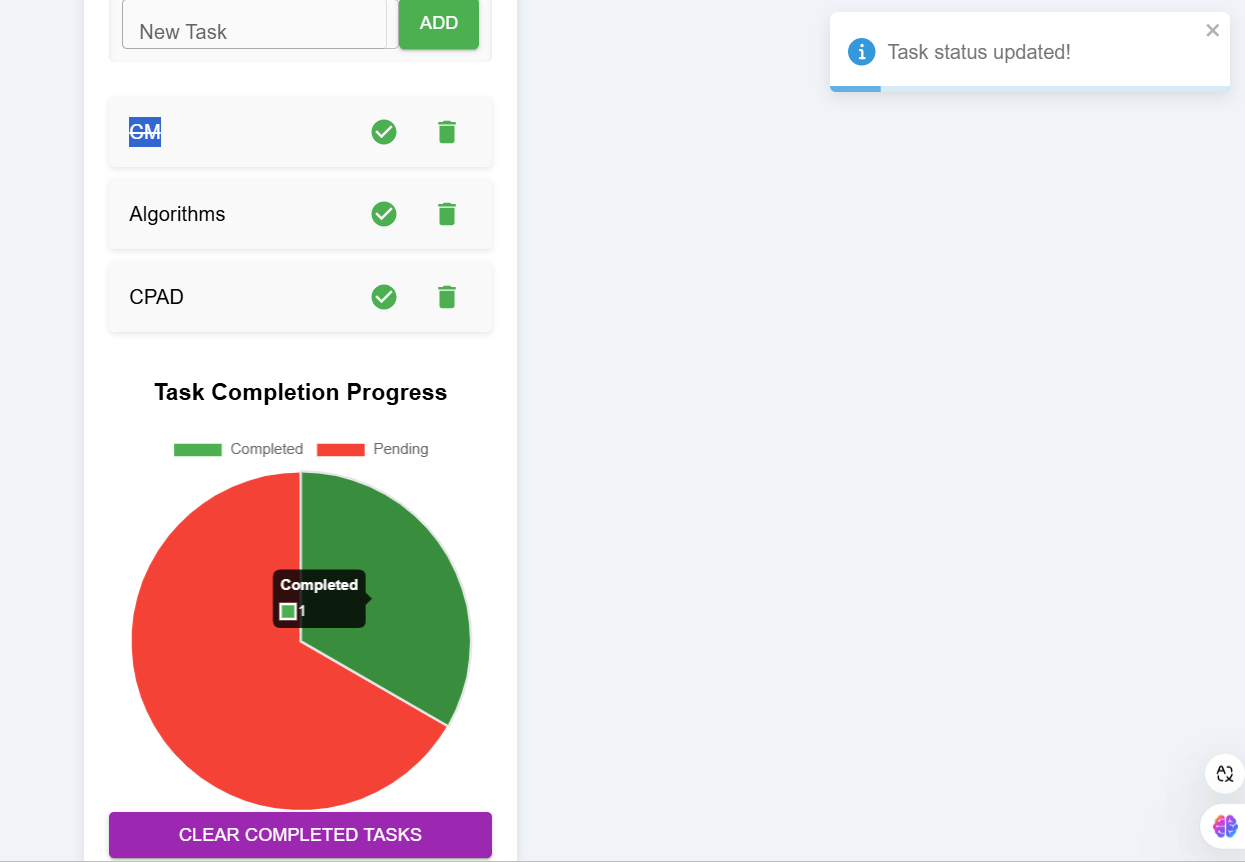
);

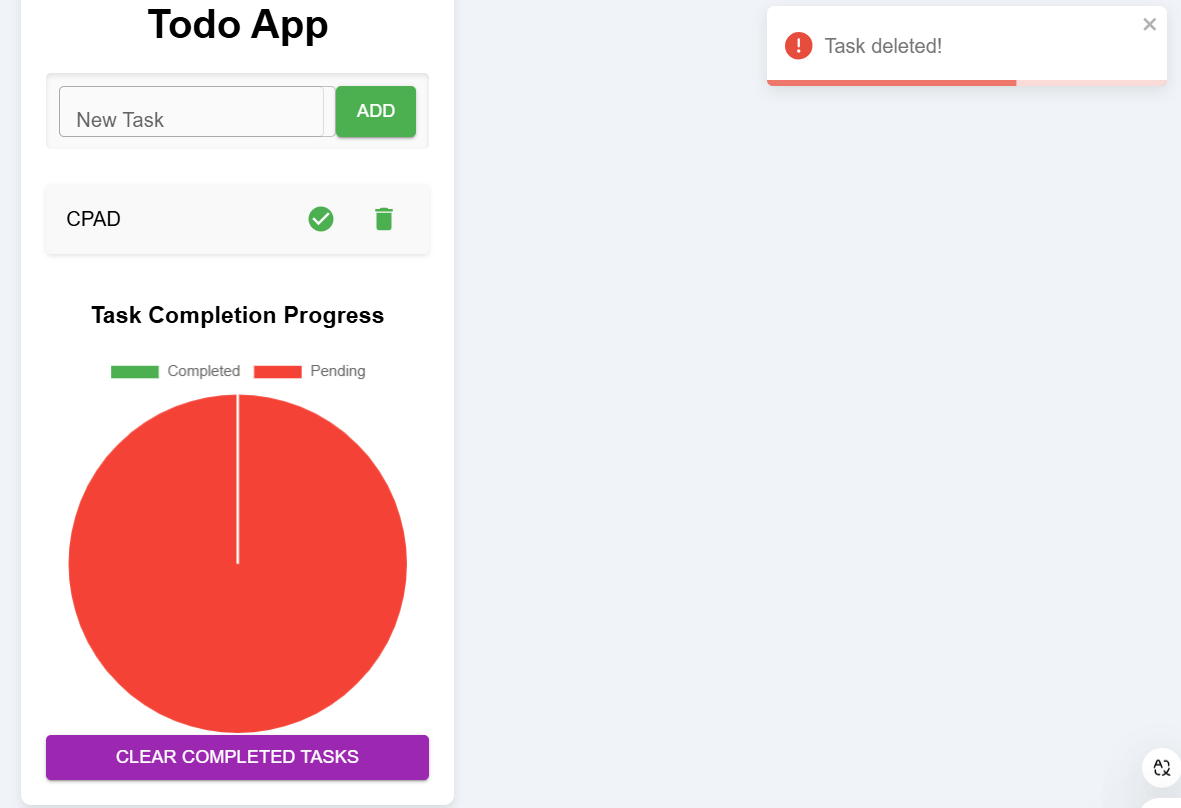
}

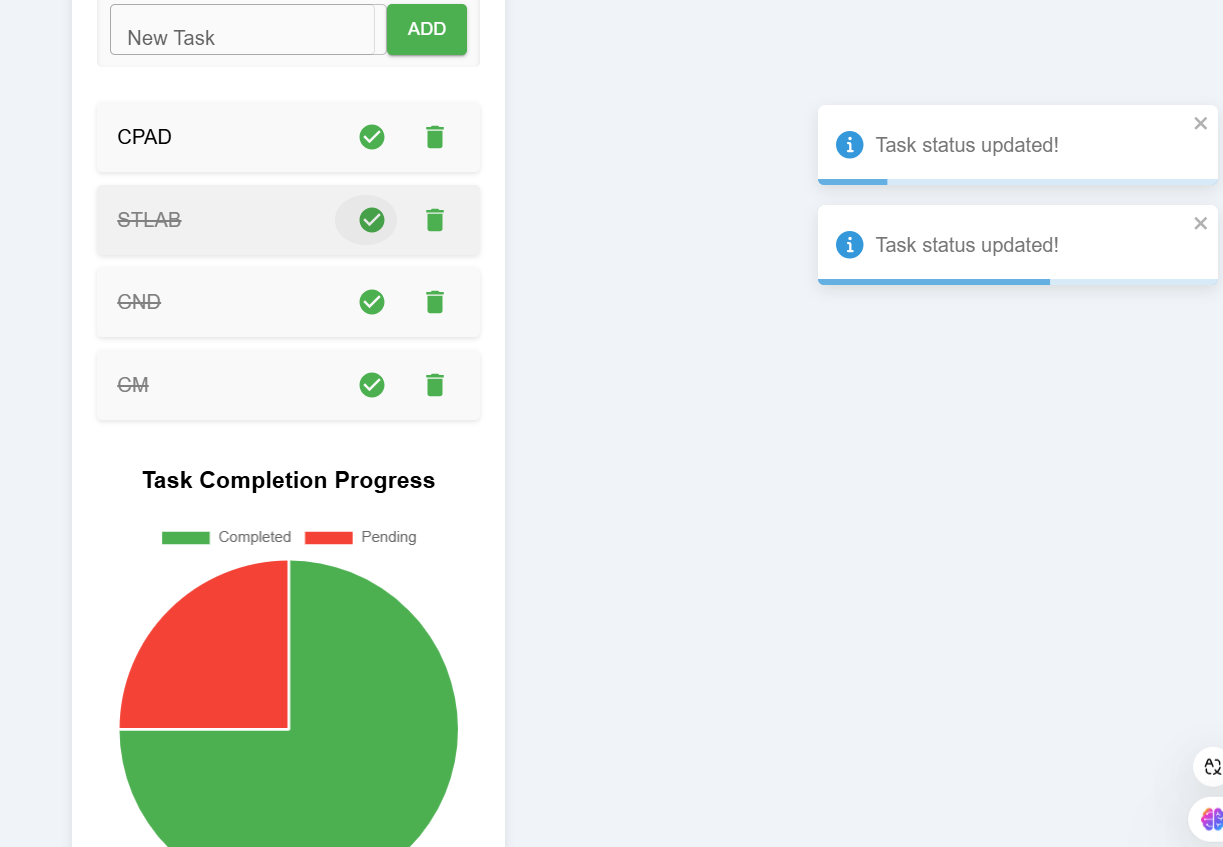
export default App;











**Conclusion:**

The **Todo App** successfully integrates third-party components to enhance its functionality and user experience. By combining task management, dynamic progress visualization via a pie chart, and real-time notifications, the app delivers an efficient solution for task management. The use of modern technologies like **React**, **Material-UI**, **Chart.js**, and **React Toastify** ensures the app is user-friendly and visually appealing, while maintaining ease of maintenance and scalability.

The project demonstrates how third-party components and libraries can be seamlessly integrated into a React app to provide a richer user experience and functional features. With these enhancements, the Todo App offers an effective and modern solution for users to organize and track their tasks.