Project Description :

Simple React Dashboard is a clean, responsive, and beginner-friendly web application built using **React.js**, designed to display user profile information, manage tasks with completion tracking, and show real-time task statistics. The dashboard layout follows modern UI/UX design principles with a vertically stacked grid structure and pastel color themes to enhance usability and visual appeal. This project demonstrates core React concepts such as component-based architecture, state management using hooks (useState), form handling, and conditional rendering. It is fully responsive across mobile, tablet, and desktop screens, ensuring an optimal user experience on any device.

Libraries Used :

1. React.js (Frontend Library)

- **Purpose:** Builds the user interface using component-based architecture.
- **Features Used:**
    - useState() for managing state.
    - JSX syntax for rendering HTML in JavaScript.
    - Functional components (App component).

2. Create React App (CRA) (React Build Tool)

- **Purpose:** Sets up the React project with sensible defaults and configuration.
- **Features:**
    - Webpack and Babel preconfigured.
    - Development server (npm start).
    - Build tools (npm run build).

3. CSS3 (Styling Language)

- **Purpose:** Styles the layout and UI components.
- **Techniques Used:**
    - CSS Grid for vertical layout.
    - Media queries for responsiveness.
    - Pastel color themes for modern UI.

4. Node.js + npm (Runtime & Package Manager)

- **Purpose:** Environment to run the app and manage dependencies.
- Installed automatically when you run :
- npx create-react-app simple-dashboard

Key Functions and Components from Code :

1. State Variables

- users:
  An array of user objects with properties like id, name, and email. Initially contains one user.
  Used to display, edit, and delete user profiles.
- editingUserId:
  Stores the ID of the user currently being edited. null means no user is being edited.
- tasks:
  An array of task objects, each with id, title, and completed status. Represents the user's to-do list.
- taskTitle:
  A string to keep track of the current input value when adding a new task.

2. Functions

- **updateUser(e, id)**
  Handles the form submission when editing a user profile.
  - Prevents default form submission behavior.
  - Updates the user in the users state whose id matches by replacing the name and email with the form values.
  - Resets editingUserId to null to exit edit mode.
- **deleteUser(id)**
  Removes the user with the given id from the users state.
- **toggleTask(id)**
  Toggles the completed status of the task with the given id.
- **deleteTask(id)**
  Removes the task with the given id from the tasks state.
- **addTask(e)**
  Handles form submission to add a new task:
  - Prevents default submission.

    - Checks if the input task title is not empty.
    - Creates a new task object with a unique ID (Date.now()), the task title, and completed: false.
    - Adds the new task to the tasks array and clears the input field.

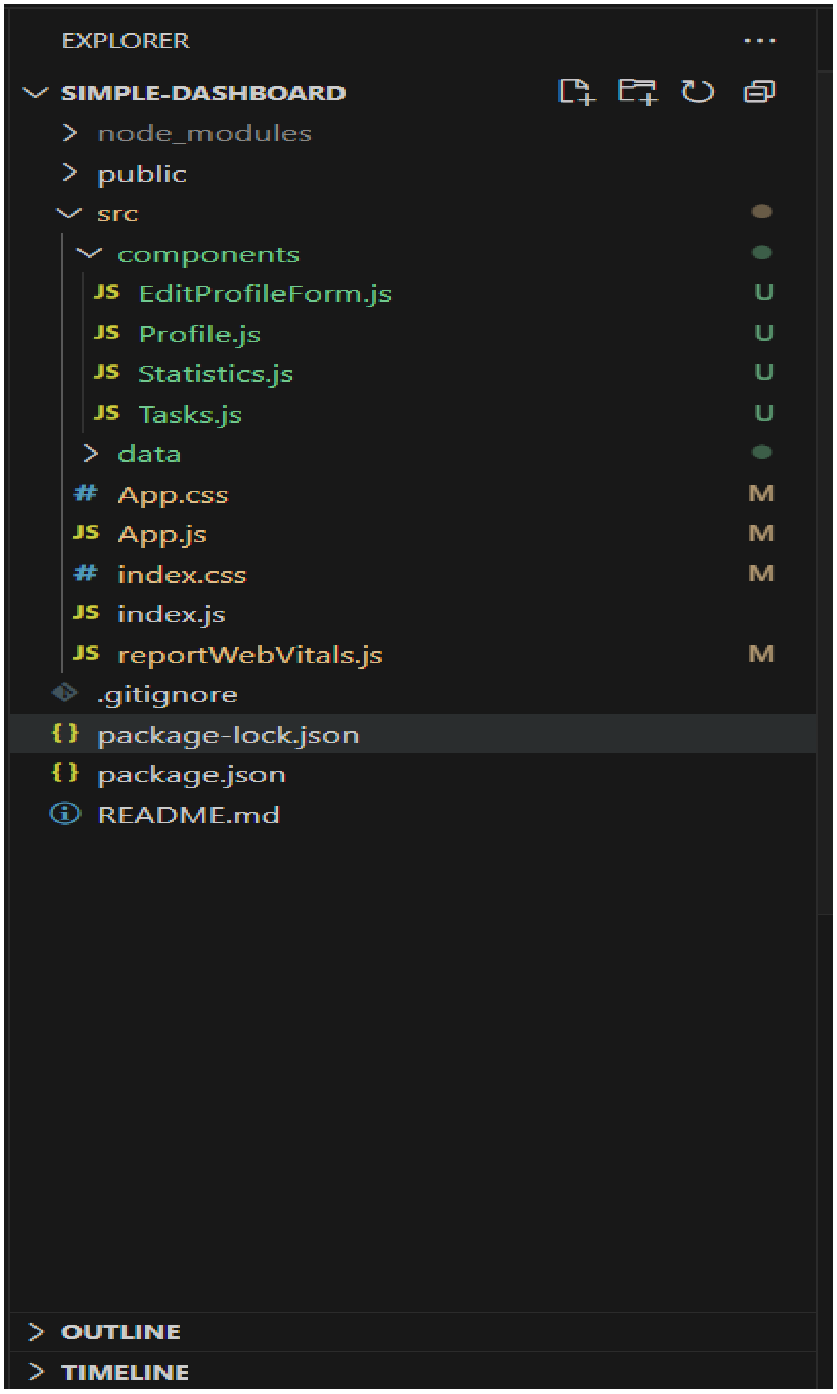
3. Derived Data

- **completed:**
  Counts how many tasks are completed by filtering the tasks array.
- **pending:**
  The number of tasks not yet completed (tasks.length - completed).

4. JSX Structure

- **Profile Section:**
  Lists users with options to edit or delete. If a user is in edit mode (editingUserId matches), it shows a form with inputs for name and email.
- **Task Manager Section:**
  Allows adding new tasks via an input and button. Displays the task list with checkboxes to toggle completion and buttons to delete tasks.
- **Statistics Section:**
  Shows total tasks, how many are completed, and how many are pending.

Overview of the Project



```
EXPLORER
SIMPLE-DASHBOARD
  > node_modules
  > public
  v src
    v components
      EditProfileForm.js        U
      Profile.js                U
      Statistics.js             U
      Tasks.js                  U
    > data
    App.css                     M
    App.js                      M
    index.css                   M
    index.js
    reportWebVitals.js          M
  .gitignore
  package-lock.json
  package.json
  README.md
> OUTLINE
> TIMELINE
```

App.js

```
import React, { useState } from 'react';
import './App.css';

function App() {
  const [users, setUsers] = useState([
    { id: 1, name: 'Aditi Sankpal', email: 'aditi@gmail.com' }
  ]);
  const [editingUserId, setEditingUserId] = useState(null);

  const [tasks, setTasks] = useState([
    { id: 1, title: 'drink water', completed: false },
    { id: 2, title: 'Submit assignment', completed: true },
    { id: 3, title: 'plant trees', completed: false }
  ]);
  const [taskTitle, setTaskTitle] = useState('');

  const updateUser = (e, id) => {
    e.preventDefault();
    const form = e.target;
    setUsers(users.map(user =>
      user.id === id ? { ...user, name: form.name.value, email: form.email.value
} : user
    ));
    setEditingUserId(null);
  };

  const deleteUser = (id) => setUsers(users.filter(user => user.id !== id));

  const toggleTask = (id) => {
    setTasks(tasks.map(task =>
      task.id === id ? { ...task, completed: !task.completed } : task
    ));
  };

  const deleteTask = (id) => setTasks(tasks.filter(task => task.id !== id));

  const addTask = (e) => {
    e.preventDefault();
    if (taskTitle.trim()) {
      setTasks([...tasks, {
        id: Date.now(),
        title: taskTitle,
        completed: false
```

```
   }]);
   setTaskTitle('');
  }
 };

 const completed = tasks.filter(t => t.completed).length;
 const pending = tasks.length - completed;

 return (
  <div className="dashboard-container">
   <h1 className="dashboard-title">Aditi's Dashboard</h1>

   <div className="dashboard-grid">
    {/* Profile Section */}
    <section className="card profile-section">
     <h2>Profile</h2>
     <ul className="profile-list">
      {users.map(user => (
       <li key={user.id}>
        {editingUserId === user.id ? (
         <form onSubmit={(e) => updateUser(e, user.id)} className="edit-
form">
          <input name="name" defaultValue={user.name} required />
          <input name="email" defaultValue={user.email} required />
          <button type="submit">Save</button>
         </form>
        ) : (
         <>
          <p><strong>Name:</strong> {user.name}</p>
          <p><strong>Email:</strong> {user.email}</p>
          <button onClick={() =>
setEditingUserId(user.id)}>Edit</button>
          <button onClick={() => deleteUser(user.id)}>Delete</button>
         </>
        )}
       </li>
      ))}
     </ul>
    </section>

    {/* Task Manager Section */}
    <section className="card task-section">
     <h2>Task Manager</h2>
     <form onSubmit={addTask} className="form">
      <input
```

```
              type="text"
              placeholder="New Task"
              value={taskTitle}
              onChange={(e) => setTaskTitle(e.target.value)}
              required
            />
            <button type="submit">Add Task</button>
          </form>
          <ul className="task-list">
            {tasks.map(task => (
              <li key={task.id}>
                <input
                  type="checkbox"
                  checked={task.completed}
                  onChange={() => toggleTask(task.id)}
                />
                <span className={task.completed ? 'completed' :
''}>{task.title}</span>
                <button onClick={() => deleteTask(task.id)}>Delete</button>
              </li>
            ))}
          </ul>
        </section>

        {/* Statistics Section */}
        <section className="card stats-section">
          <h2>Statistics</h2>
          <p><strong>Total Tasks:</strong> {tasks.length}</p>
          <p><strong>Completed:</strong> {completed}</p>
          <p><strong>Pending:</strong> {pending}</p>
        </section>
      </div>
    </div>
  );
}

export default App;
```