

User Management Dashboard

A **responsive ReactJS dashboard** to display and manage user information. Supports **searching users, adding users, deleting users**, and **light/dark theme toggle** using Tailwind CSS and Redux Toolkit.

Features

- **Responsive Design:** Works seamlessly on desktop, tablet, and mobile devices.
- **User Listing:** Fetches users from [JSONPlaceholder API](#) and displays them in cards.
- **Search/Filter:** Search users by name, email, or company.
- **Add User:** Add new users through a modal form with validation.
- **Delete User:** Remove users directly from the list.
- **Theme Toggle:** Light and dark themes with global context.
- **State Management:** Uses Redux Toolkit to manage user data.
- **Error Handling & Fallback:** Displays loading states and handles API errors gracefully.

Tech Stack

- **Frontend:** React (Functional Components + Hooks)
- **State Management:** Redux Toolkit
- **Styling:** Tailwind CSS with class-based dark mode
- **API:** Axios / Fetch API
- **Build Tool:** Vite

Project Structure

```
src/
  └── components/
    ├── common/
    │   ├── Button.jsx
    │   ├── Card.jsx
    │   └── Modal.jsx
    └── users/
        ├── UserCard.jsx
        ├── userList.jsx
        └── AddUserForm.jsx
  └── store/
    ├── index.js
    └── slices/
        └── userSlice.js
  └── themes/
    └── ThemeProvider.jsx
  └── utils/
    └── constants.js
```

```
└── App.jsx  
└── main.jsx
```

Installation

1. Clone the repository

```
git clone <your-repo-url>  
cd user management dashboard
```

2. Install dependencies

```
npm install
```

3. Start the development server

```
npm run dev
```

Visit <http://localhost:5173> to view the app.

Usage

- **View Users:** Displays all users in a responsive card grid.
- **Search Users:** Filter users by typing in the search bar.
- **Add User:** Click [Add User](#) to open a modal form, fill details, and submit.
- **Delete User:** Click the delete button on any user card to remove it.
- **Toggle Theme:** Click [Dark](#) or [Light](#) to switch themes dynamically.

Approach

1. **Layout & Design:** Used **CSS Grid/Flexbox** for responsive layout and Tailwind CSS for consistent styling.
2. **API Integration:** Fetched user data from **JSONPlaceholder API** with loading and error handling.
3. **Component Reusability:** Created reusable components for **Button, Card, Modal, UserCard**.
4. **Redux:** Managed user data globally, including actions to **fetch, add, and delete users**.
5. **Theme Management:** Implemented a **ThemeProvider** to toggle between light and dark mode.
6. **Fallback:** Used static local data if API fails.

Challenges & Solutions

- **Dynamic Dark Mode:** Initially, UI didn't update on theme toggle.
 - **Solution:** Applied `.dark` class on `<html>` and added `transition-colors` for smooth updates.
 - **API Failure Handling:** Ensured app works even if API is down using **fallback static data**.
 - **Responsive Layout:** Adjusted Grid and Flex layouts for mobile and tablet screens.
-

Author

Nitesh Sharma Email: niteshkumarsharma831@gmail.com

Phone: +91 9572861917 LinkedIn: <https://www.linkedin.com/in/nitesh-kumar-sharma-2894a1185/>

Portfolio: <https://devcraftnitesh.vercel.app/>

Previous Projects

- **E-Commerce Website:** [Shopizo Online](#)
- **Job Seeking Platform:** [HireOn WorkBridge](#)