**✅ Admin Dashboard (React + TypeScript + Vite) — Progress Documentation**

**🔧 Technologies Used**

* **React** with **TypeScript**
* **Vite** (for fast bundling and development)
* **React Router** for routing (/login, /dashboard)
* **Chart.js** for statistics visualization
* **Axios** for HTTP requests
* **Bootstrap 5** for layout
* **Custom CSS** (dashboard.css) for the chic sidebar and layout

**✅ Implemented Features**

**1. Authentication**

* /login page built with:
  + Email & Password inputs
  + Axios POST to /auth/login
  + Token saved to localStorage
  + Auto-redirect to /dashboard if logged in
* Session persistence using !!localStorage.getItem('token')

**2. Dashboard Layout**

* Clean layout with:
  + A fixed **left sidebar** (Admin, Dashboard, Projects, Users, Settings)
  + Top-right **Logout** button
  + Responsive central content

**3. Dashboard Content**

* Stats boxes:
  + Total Emails
  + Total Projects
  + Total Users
* User list table (fetched via /dashboard/users)
* Two bar/line charts showing monthly activity

**4. Error Handling & Debugging**

* Fixed Chart.js re-init issues (canvas reuse)
* Type safety using a shared User interface
* TS errors like type 'any' not assignable resolved

**✅ File Structure (Key)**

pgsql

CopierModifier

📁 src/

├── api/

│ └── axiosInstance.ts

├── assets/

│ └── dashboard.css

├── components/

│ ├── Dashboard.tsx

│ ├── LoginPage.tsx

│ └── UserTable.tsx

├── types/

│ └── types.ts // contains interface User

├── App.tsx

├── main.tsx

The great works began :

## ✅ Project Documentation – Admin Dashboard

### 📁 1. Frontend (React + Vite + TypeScript)

#### ✔️ Implemented Features:

* ✅ **Login Page**
  + Email/password login with token saved to localStorage
  + Error feedback for invalid credentials
* ✅ **Protected Routing**
  + /dashboard is protected and only accessible with valid token
  + Redirects unauthenticated users to /login
* ✅ **Dashboard Page**
  + Beautiful layout with sidebar navigation
  + Top bar with logout button
  + Responsive design using dashboard.css
  + Displays:
    - Total emails, projects, users
    - User list from backend
    - Monthly activity charts using Chart.js
* ✅ **Sidebar Menu**
  + Fixed left-side navigation
  + Routes: Dashboard, Projects, Users, Settings (placeholders for now)
* ✅ **Logout**
  + Clears token and navigates to /login

#### 🔗 Axios Setup:

* Custom instance in axiosInstance.ts
* Automatically includes Authorization headers if token exists

#### 🧠 Types:

* Defined User type in types.ts and reused across app

### 🖥️ 2. Backend (ASP.NET Core Web API)

#### ✔️ Implemented API Endpoints:

| **Route** | **Method** | **Description** |
| --- | --- | --- |
| /auth/login | POST | Authenticates user and returns JWT token |
| /dashboard/stats | GET | Returns emails, projects, users count |
| /dashboard/users | GET | Returns list of registered users |

#### 🔐 Security:

* JWT authentication implemented
* All dashboard endpoints are protected by token middleware

#### 📦 Controllers:

* AuthController: Handles login and JWT generation
* DashboardController: Provides stats and user list

#### 🗃️ Models:

* User, LoginRequest, DashboardStats etc.

#### 🧪 Tested:

* All APIs tested successfully with Swagger and Axios from frontend

## 🔜 Recommended Next Steps

### 🚀 Phase 2 – Improvements & New Features

#### 1. ✅ Refactor & Polish

* Improve mobile responsiveness
* Make logout button more elegant (icon with tooltip)
* Add a profile picture/icon at the top right
* Add theme toggle (light/dark)

#### 2. 🔐 User Profile Section

* /profile page to view/edit user data
* Upload avatar image
* Add user role and profile settings

#### 3. 📊 Analytics

* Dynamic data for charts (from database)
* Live updates with SignalR (optional)

#### 4. 🧑‍💼 Admin Management

* Create/update/delete users
* Role-based access (admin vs user)

#### 5. 🗃️ Data Management

* CRUD for projects, settings, emails, etc.
* Paginate and filter user list

#### 6. 🛠 Deployment

* Deploy backend on Azure App Service
* Deploy frontend on Vercel or Netlify
* Use environment variables for API URLs

**🛠️ Next Recommended Steps**

**🔹 1. Profile Page**

* Add a route like /profile
* Allow editing name, password, and avatar
* Reuse layout with sidebar

**🔹 2. Routing Logic Refactor**

* Use <PrivateRoute> pattern to protect authenticated pages more cleanly

**🔹 3. Better Chart Data**

* Replace hardcoded datasets with backend-driven data

**🔹 4. Sidebar Improvements**

* Add icons from Unicons (already linked)
* Add collapsible sidebar (mobile responsive)

**🔹 5. Theme Switcher**

* Light/dark mode toggle using useContext or useState

**🔹 6. API Error Feedback**

* Show toast messages or alerts on login failure or API errors

**🔹 7. Export User Data**

* Add a button to export user list to CSV or PDF

**🔹 8. Role-Based Access**

* If your app will have admin/editor roles, use token-based role validation