# POC- Customer Dashboard Application

**Objective**

This **Proof of Concept (POC)** is designed to manage **customer data efficiently**, allowing users to perform operations such as **adding, editing, deleting, and viewing** customer records. The system includes **authentication & role-based access control** to differentiate between **Admin** and **Customer** users.

**Backend (ASP.NET Core)**

**✅ Tech Stack & Components**

* **Framework:** ASP.NET Core Web API
* **Database:** Microsoft SQL Server
* **Authentication:** JWT-based authentication
* **Data Access:** Entity Framework Core (EF Core)
* **Logging & Error Handling:** Built-in Exception Handling

**✅ Key Backend Features**

* **User Authentication (**Login API **with JWT tokens)**
* **Role-Based Access Control (**Admin **vs** Customer**)**
* **Customer CRUD Operations (**Create, Read, Update, Delete**)**
* **Secure API Access (**Using Bearer Token for Authentication**)**

**✅ Important Backend APIs**

**1. User Login API (POST /api/login)**

* Validates user credentials
* Generates a **JWT token** containing user role
* Returns token, username, and role

**2. Customer Management APIs**

* **Create Customer:** POST /api/customer
* **Get Customers:** GET /api/customer
* **Update Customer:** PUT /api/customer/{id}
* **Delete Customer:** DELETE /api/customer/{id}

**3. Authentication & Role Management**

* JWT Authentication **validates user identity** before API access.
* Role-based access ensures **Admin can manage customers**, while **Customers** have limited privileges.

**🖥️ Frontend (Angular)**

**✅ Tech Stack & Components**

* **Framework:** Angular
* **Styling:** Bootstrap for UI enhancements
* **State Management:** LocalStorage-based session handling
* **API Communication:** HTTP Client in Angular

**✅ Key UI Features**

* **Login Screen (**JWT authentication**)**
* **Dynamic Navbar (**Admin & Customer Role Display**)**
* **Customer Management (**Add, Edit, Delete & Display**)**
* **Bootstrap Modal (**Used for Adding & Editing Customers**)**
* **User-Friendly UI (**Card-styled Table for Customer Records**)**

**✅ Important Components**

**1. HeaderComponent**

* Displays **navbar**, user role, and logout option
* Dynamically updates **UserRole & UserName** after login/logout

**2. CustomerDetailsComponent**

* Displays customer data in a **Card-Styled Table**
* Allows **editing & adding customers** via Bootstrap modals
* Calls the backend APIs for **CRUD operations**

**3. Authentication Handling (AuthService)**

* Stores **token, username, role** in LocalStorage
* Uses **BehaviorSubject** for real-time role updates

**Additional Notes:**

* **Video Demo Reference:** I have included a video demo showcasing how the application works locally.
* **Video Link:** <https://github.com/puriJagannadhK/ValueLabsDemo/blob/main/Demo/demo.mp4>
* **Connection String & API URLs:** The connection string has been commented out, and the API URLs have been manually added for seamless integration.
* **Controller Structure:** I have added only the **controller structure for time being**, but we can extend this further by implementing a **service layer and repository structure**

**Database Table Structure:** CR\_Users

A screenshot of a computer program

AI-generated content may be incorrect.