**Customer Management System – Frontend - User Guide**

**Frontend – Angular, Angular Material UI Template, MatSnackBar Notification with JWT Authentication and Roles based Authorization**

**This is a standalone Angular application built using Angular CLI. The project includes features like JWT authentication, role-based authorization, form validations, lazy loading, reusable components, services, and more.**

**## 🔧 Features**

**- Angular – 18 Standalone App**

**- Material UI Template**

**- Enterprise Application – Folder Structure – app, core, shared, features**

**- JWT Authentication & Role-based Authorization**

**- MatSnackBar Notification**

**- Lazy Loading Modules**

**- Reactive Forms with Validation**

**- Angular Material UI**

**- CRUD Operations for Customer Entity**

**-** **List with Paging, Sorting, Searching**

**- Exception Handling**

**- Auth Guards, Role Guard, Resolver and Interceptors**

**- RxJS Observables, Services, Pipes**

**- State Management (via services)**

**- Fully Optimized for Performance**

**## 📦 Technologies Used**

**- Angular (latest version -18)**

**- Angular CLI**

**- CSS**

**- RxJS**

**- Angular Material**

**- Material Snack Bar**

## ✅ ****Angular CMS Setup and Run Guide****

## ✅ ****Step-by-Step Instructions****

### 📌 Prerequisites

Before starting, ensure you have the following installed:

* **Node.js** (v18 or later recommended) 👉 <https://nodejs.org>
* **Angular CLI** 👉 npm install -g @angular/cli
* **Git** 👉 <https://git-scm.com>
* **Code Editor** (e.g., VS Code) 👉 <https://code.visualstudio.com>

### 🔹 1. ****Clone the Project from GitHub****

Open a terminal and run:

git clone https://github.com/SheikGH/CMS\_Angular\_Frontend.git

Example:

git clone https://github.com/SheikGH/CMS\_Angular\_Frontend.git

### 🔹 2. ****Navigate to the Project Folder****

cd CMS\_React\_Frontend

### 🔹 3. ****Create and Configure**** environment ****File****

In the root of the project (inside CMS\_Angular\_Frontend /),

create a file named **environment.ts** if it doesn’t already exist.

**Change apiUrl in the environment.ts file in the path cms\src\environments\environment.ts:**

apiUrl=https://localhost:7067/api

This will make sure your Angular app communicates with the correct backend API.

### 🔹 4. ****Install Node.js (if not already installed)****

Make sure you have **Node.js (v14+ or v16+)** and **npm** installed.

To check:

node -v

npm -v

To install: <https://nodejs.org/>

### 🔹 5. ****Install Project Dependencies****

npm install

This will install all required packages listed in package.json.

### 🌐 Step 3: Serve the Angular App Locally

bash

CopyEdit

ng serve

Then open your browser and go to:

text

CopyEdit

http://localhost:4200/

### 

### 🔹 7. ****Make Sure Backend Is Running****

Ensure your .NET Core API project is running at:

https://localhost:7067/swagger/index.html

If you're using Visual Studio:

* Set the API project as **Startup Project**
* Press F5 or click **Start Debugging**

### 🔹 8. ****Common Pages and Actions****

| **Page** | **Path** | **Description** |
| --- | --- | --- |
| Register Page | /register | Add new customer (registers user) |
| Login Page | /login | Login with JWT token, saved in localStorage |
| Customer List Page | /customers | View, Edit, Delete customer records |
| Logout | - | Clears token, redirects to Login page |

### 🛠️ ****Troubleshooting Tips****

* If you get **CORS errors**, make sure your .NET backend has CORS enabled:

services.AddCors(options =>

{

options.AddPolicy("AllowAll",

builder =>

{

builder.AllowAnyOrigin()

.AllowAnyHeader()

.AllowAnyMethod();

});

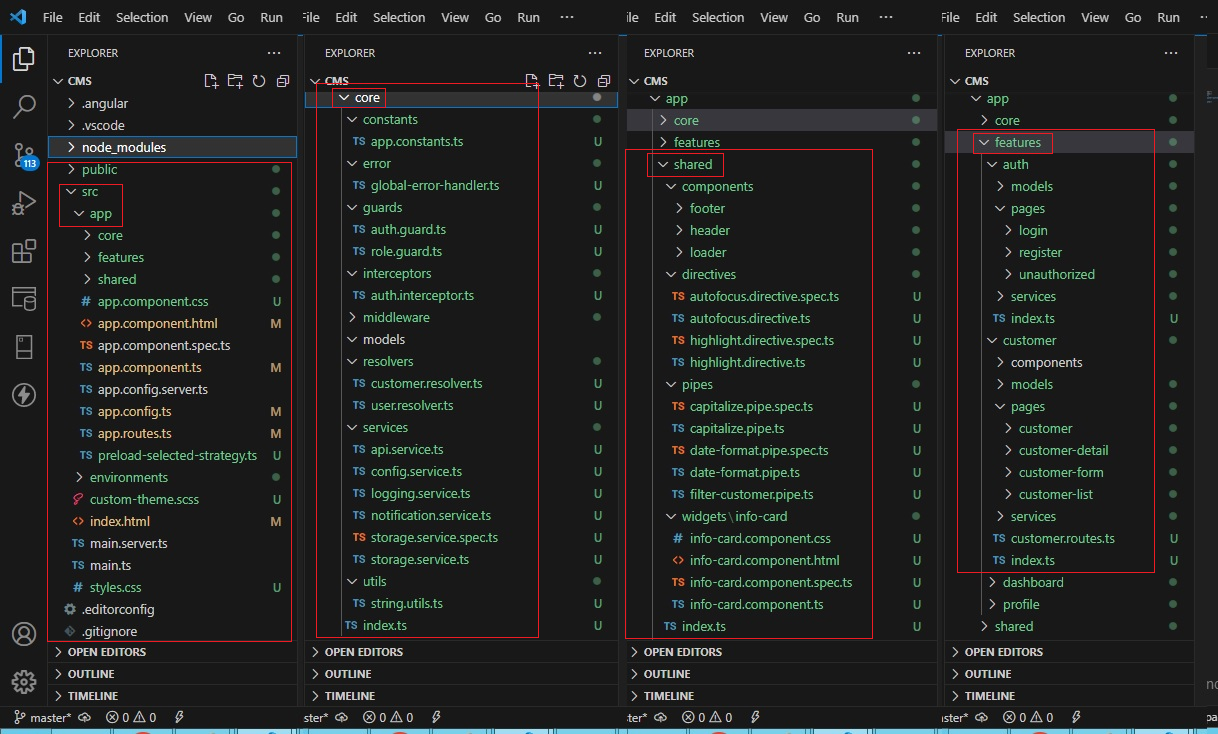
});

app.UseCors("AllowAll");

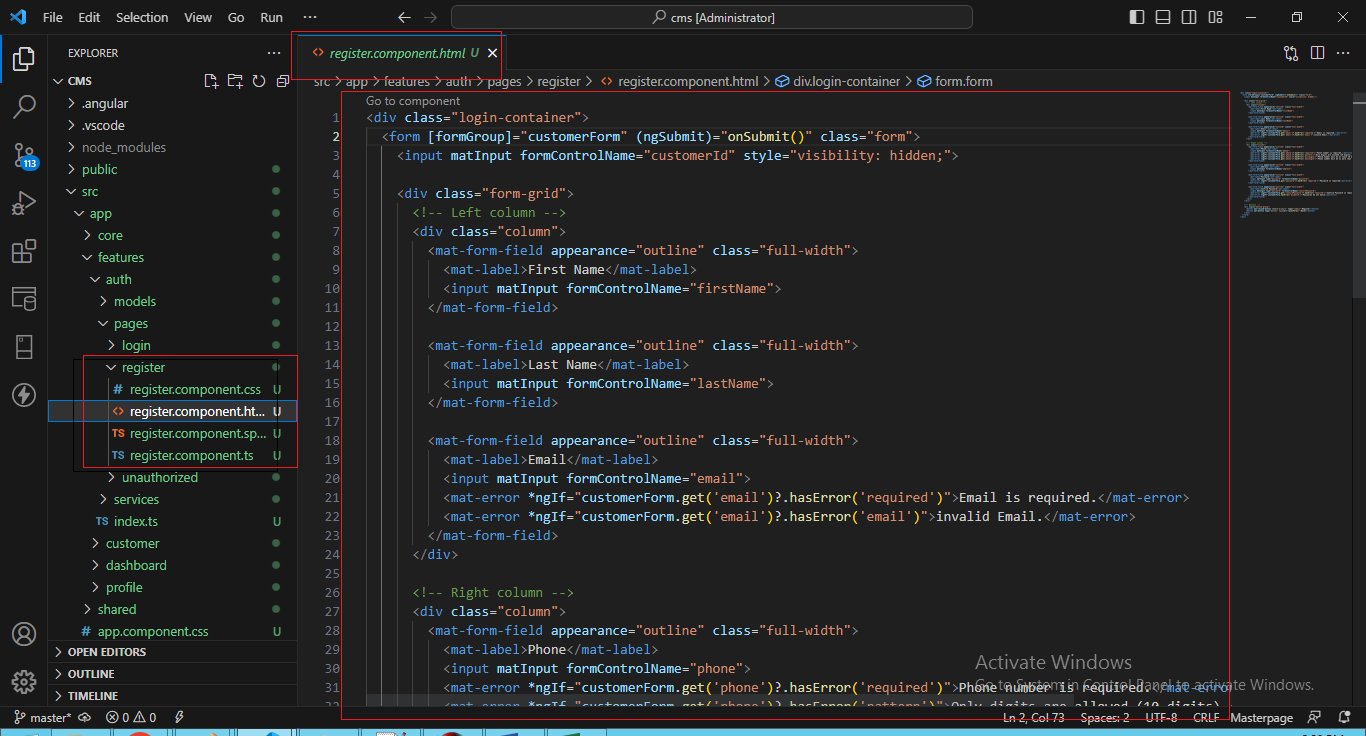
* If .env changes are not reflected, **restart the server** after editing the .env file.

**Frontend – Angular, Angular Material UI Template with JWT Authentication and Roles based Authorization**

**Folder Structure: App, Core, Shared and Features**

****

**✅ 1. Register Page:** <http://localhost:4200/register>

****

**📌 Features:**

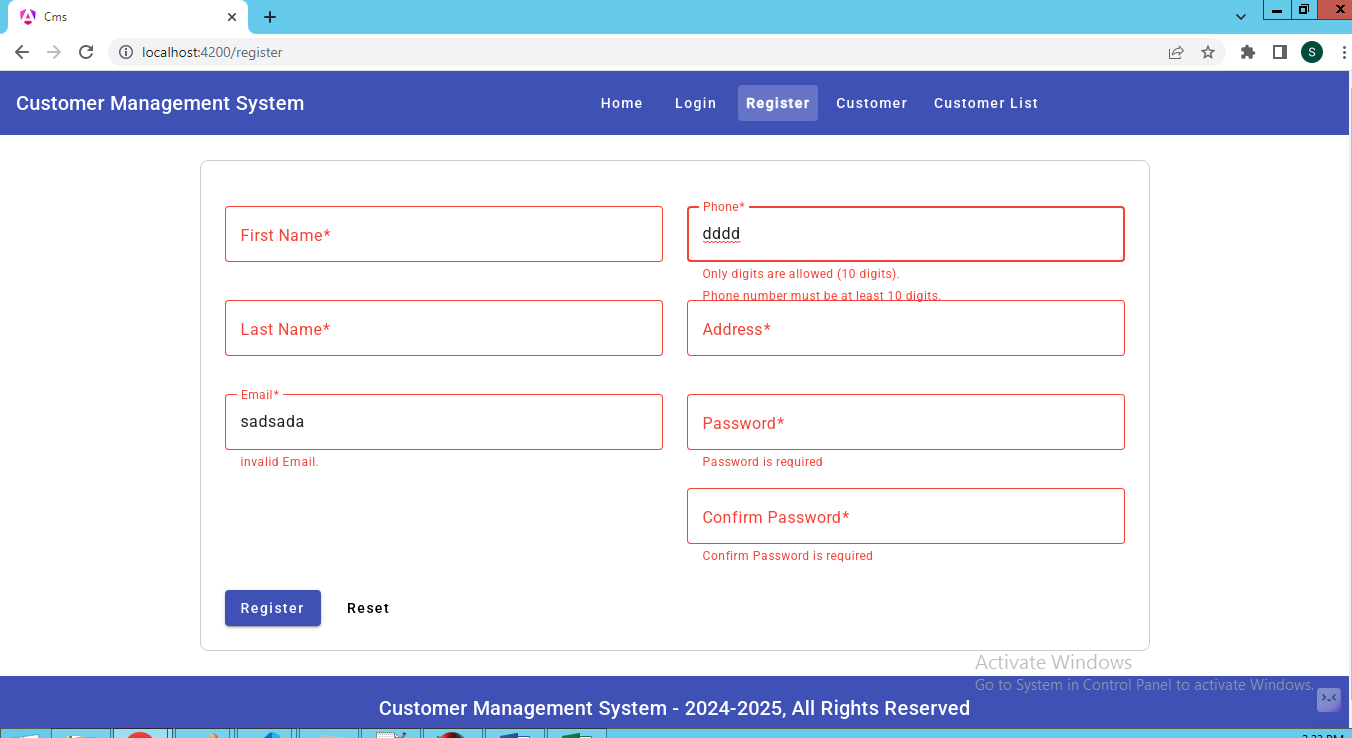
* Add a new customer with proper form validation.
* Submit valid customer data to the backend API and store it in the database.

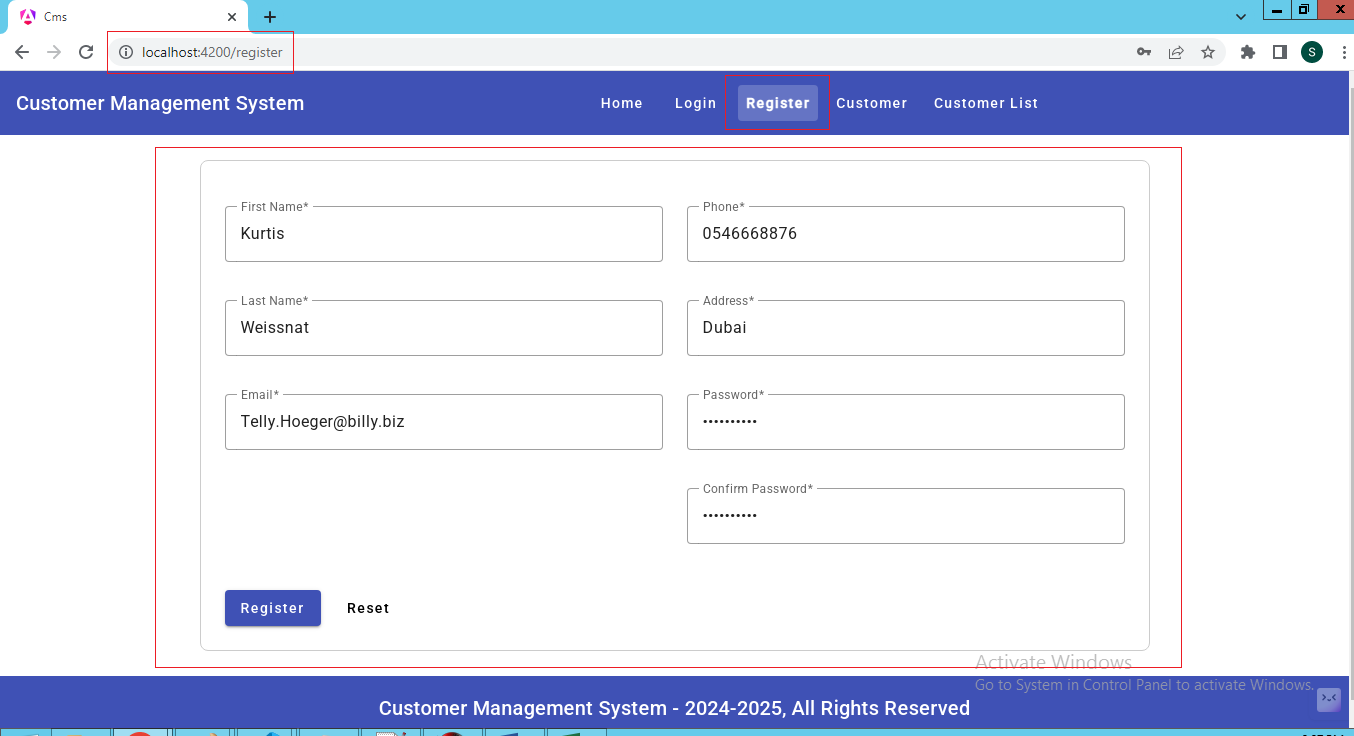
**✔️ Validations:**

* **Required fields:** First Name, Last Name, Email, Phone, Address, Password
* **Email format check**
* **Password strength:** Example: At least 6–8 characters, with uppercase, lowercase, number, and special character.

**✔️ Flow:**

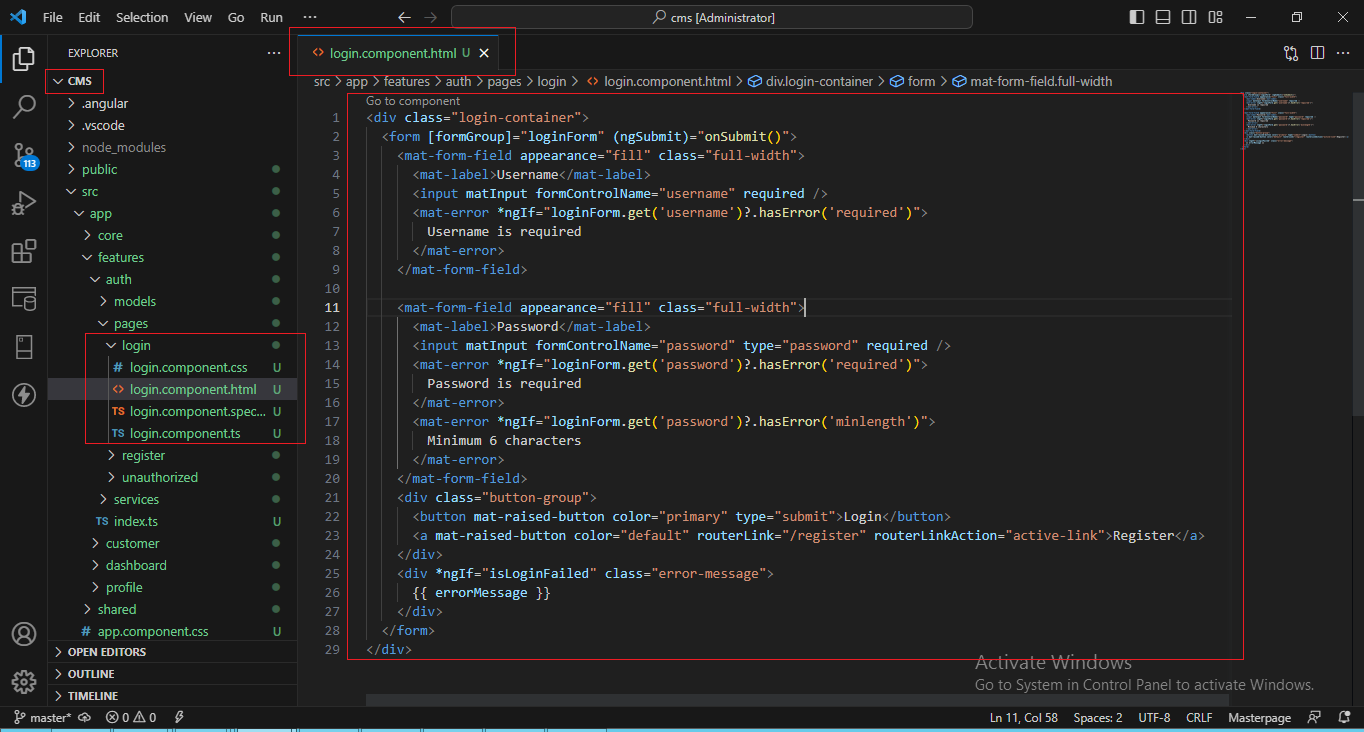
1. User fills out the form.
2. If any field is invalid → show validation errors.
3. If all fields are valid → POST request to API (e.g., /api/register).
4. On success → redirect to Login page or show success message.





Once submit Register page then redirect to Login page

**✅ 2. Login Page: http://localhost:4200/login**

****

**📌 Features:**

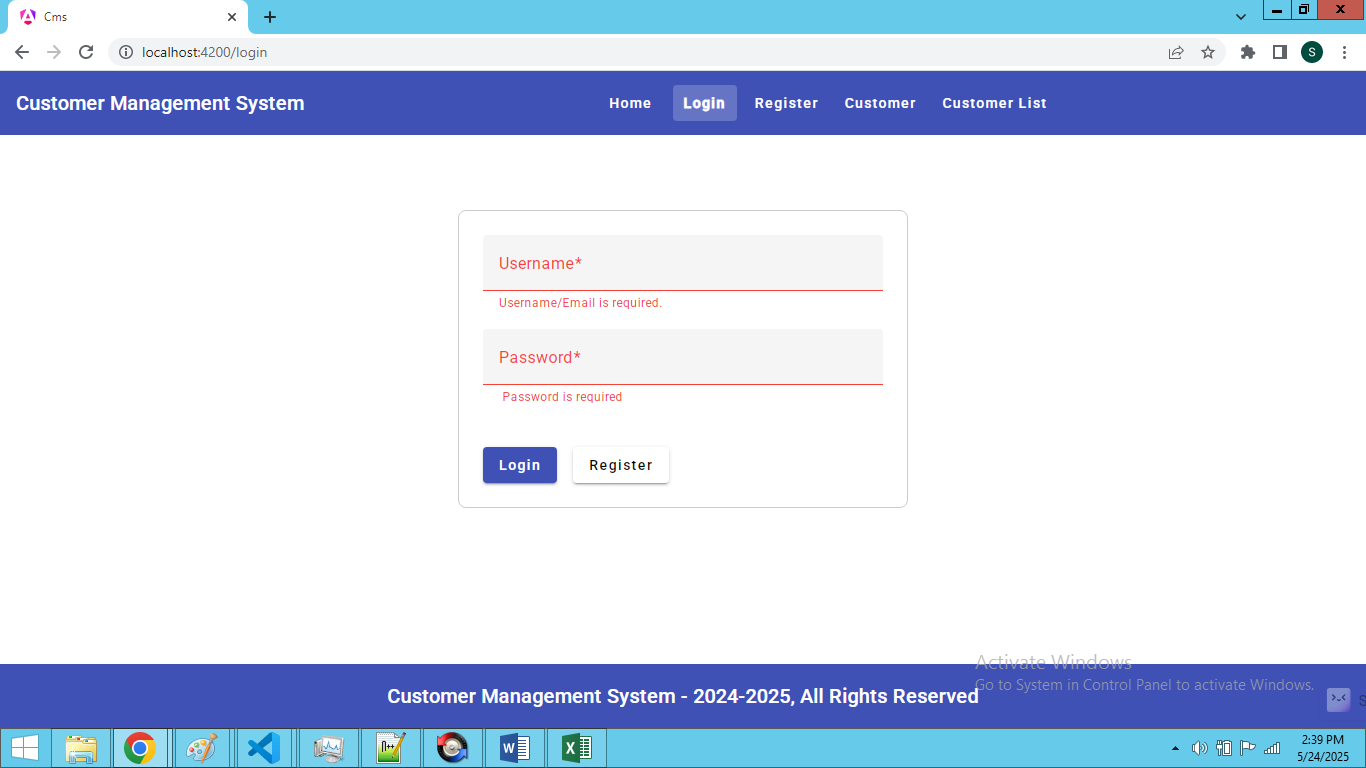
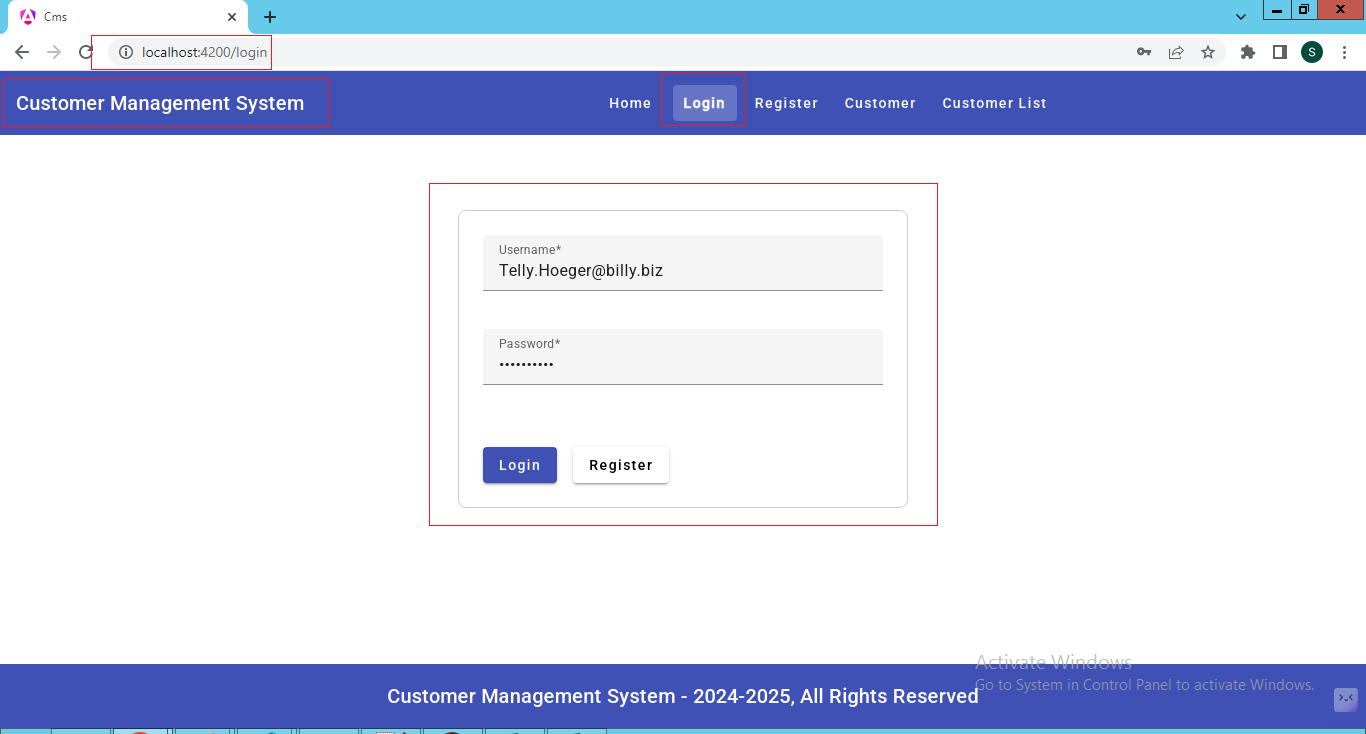
* Authenticate user using email and password.
* Receive JWT from API and store in localStorage.

**✔️ Validations:**

* Email is required and must be valid.
* Password is required.

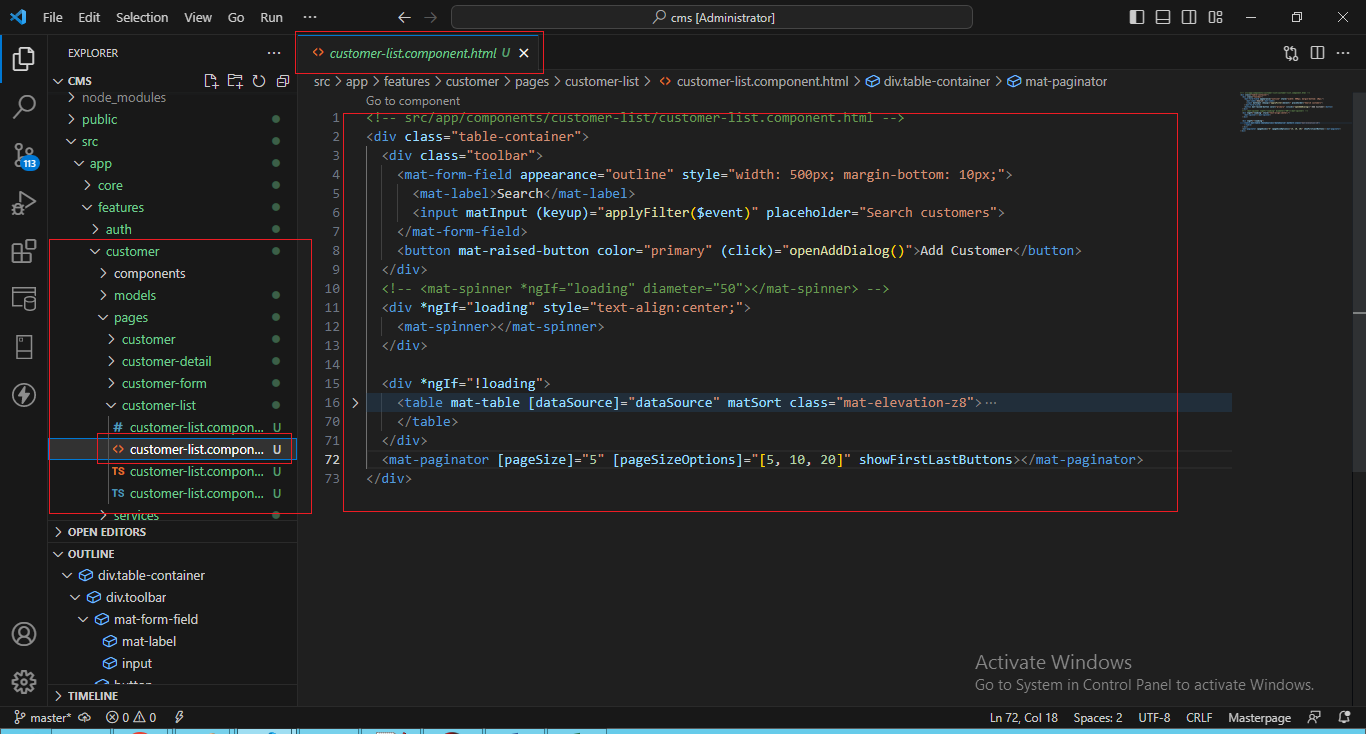
**✔️ Flow:**

1. User enters login credentials.
2. If invalid format → show client-side validation errors.
3. If wrong credentials → show API error message.
4. If credentials are correct:
   * Backend returns JWT token.
   * Store token in localStorage.
   * Redirect to /customer-list.

Page redirect to Customer List page

**✅ 3. Customer List Page: http://localhost:3000/customer-list**

****

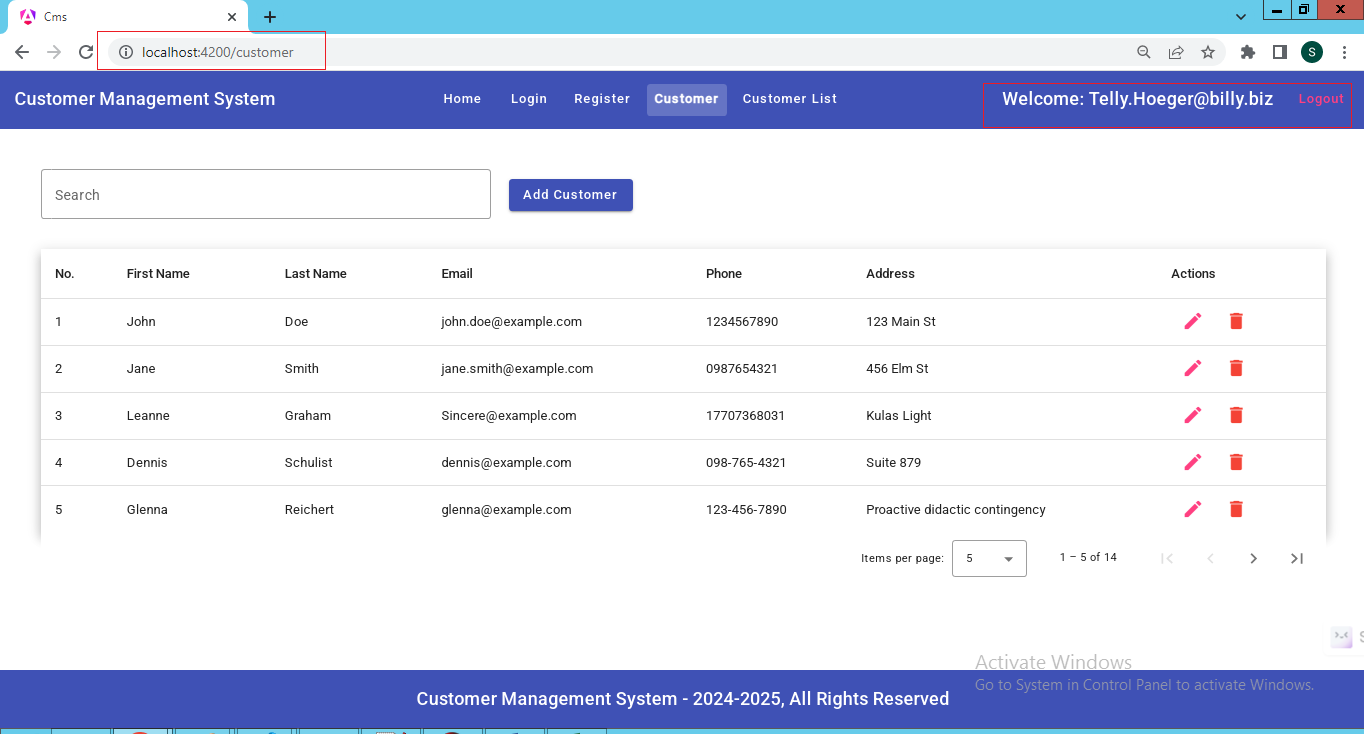
**📌 Features:**

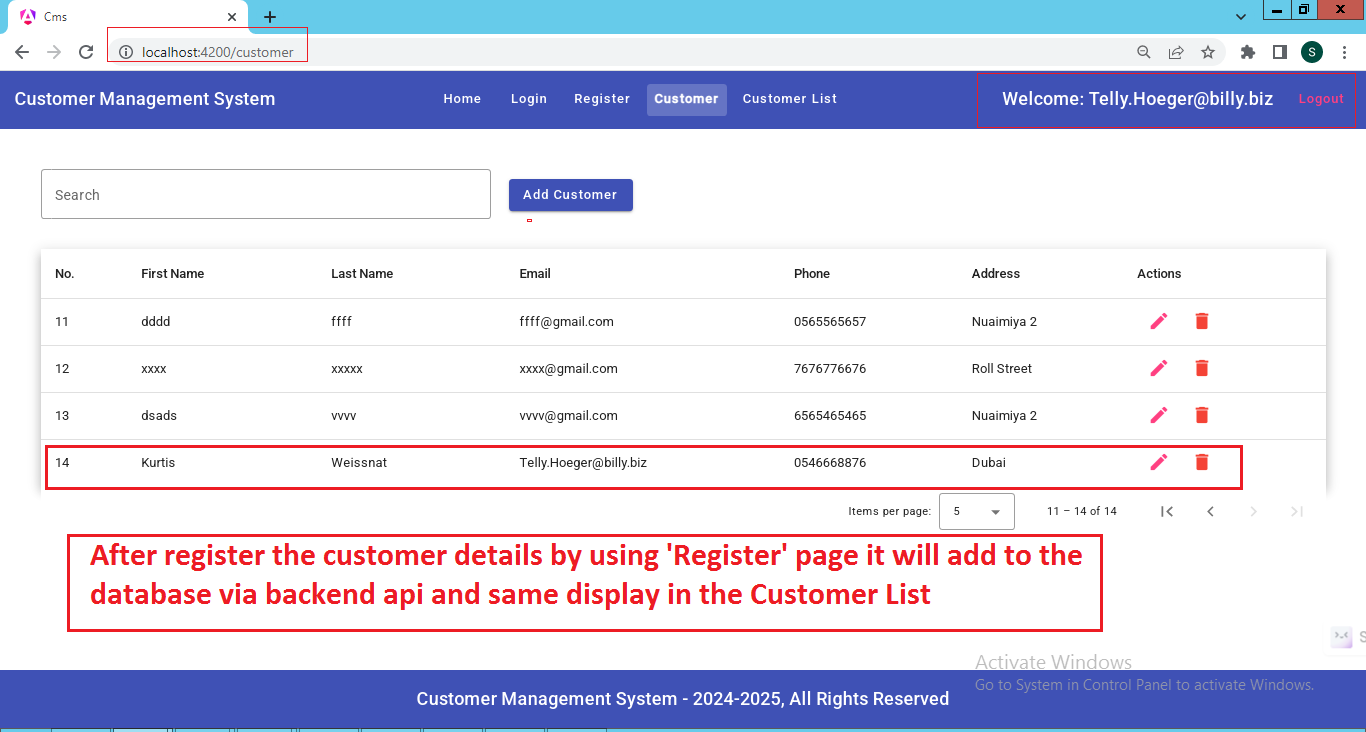
* View, edit, delete customer.
* Require JWT token to access.

**✔️ Functionality:**

**🟩 View Customer:**

* Fetch data from /api/customers using JWT token in headers.
* Display in table format.

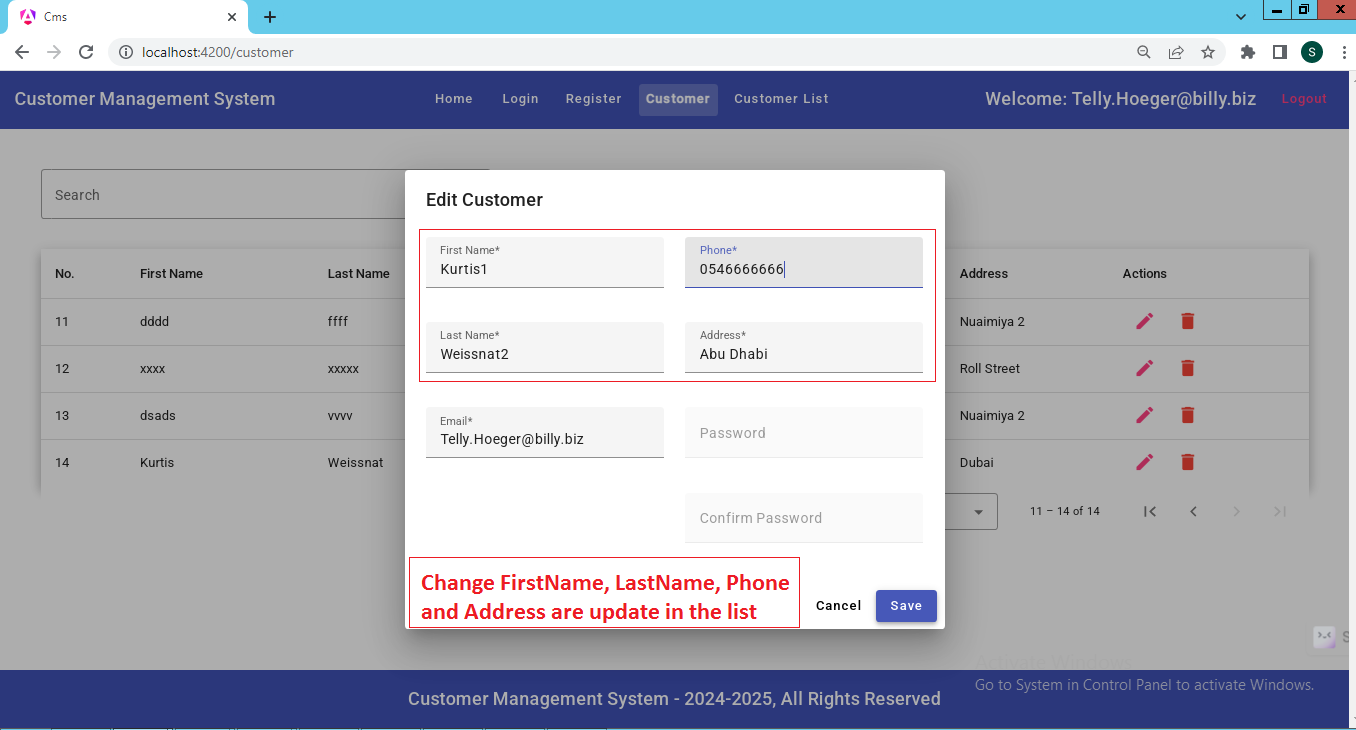


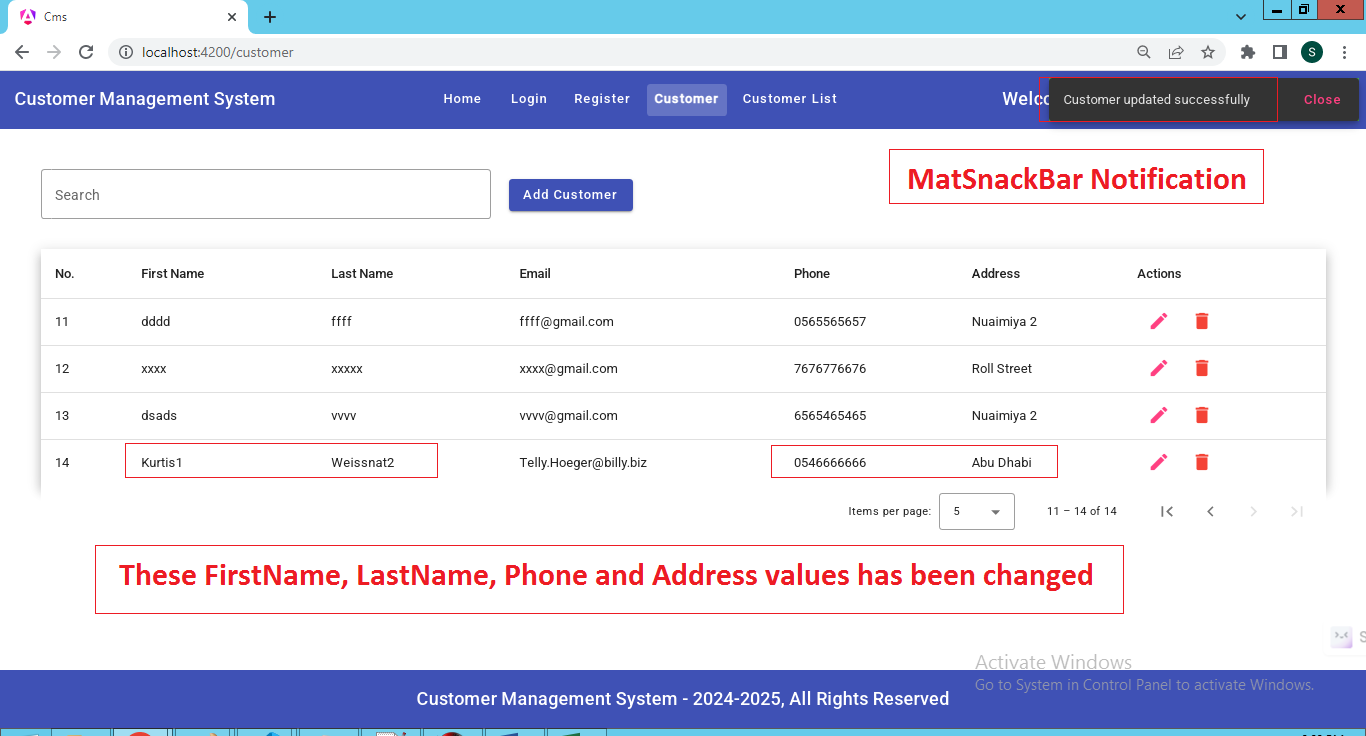


**🟨 Edit Customer:**

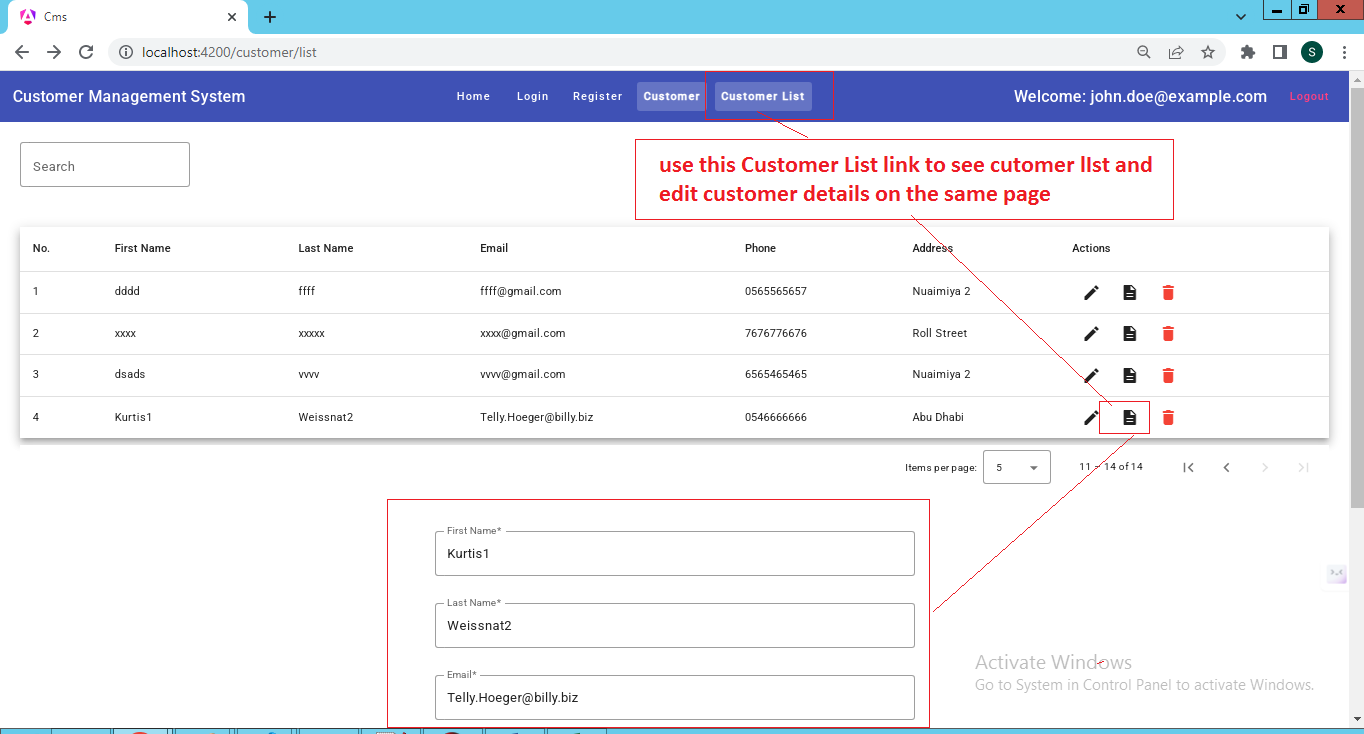
* Click on "Edit" → Inline form or Modal.
* Validate fields (same as Register).
* On valid input, PUT request to API.
* On success, reload or refresh list.

**Customer list value are edited by popup model form**

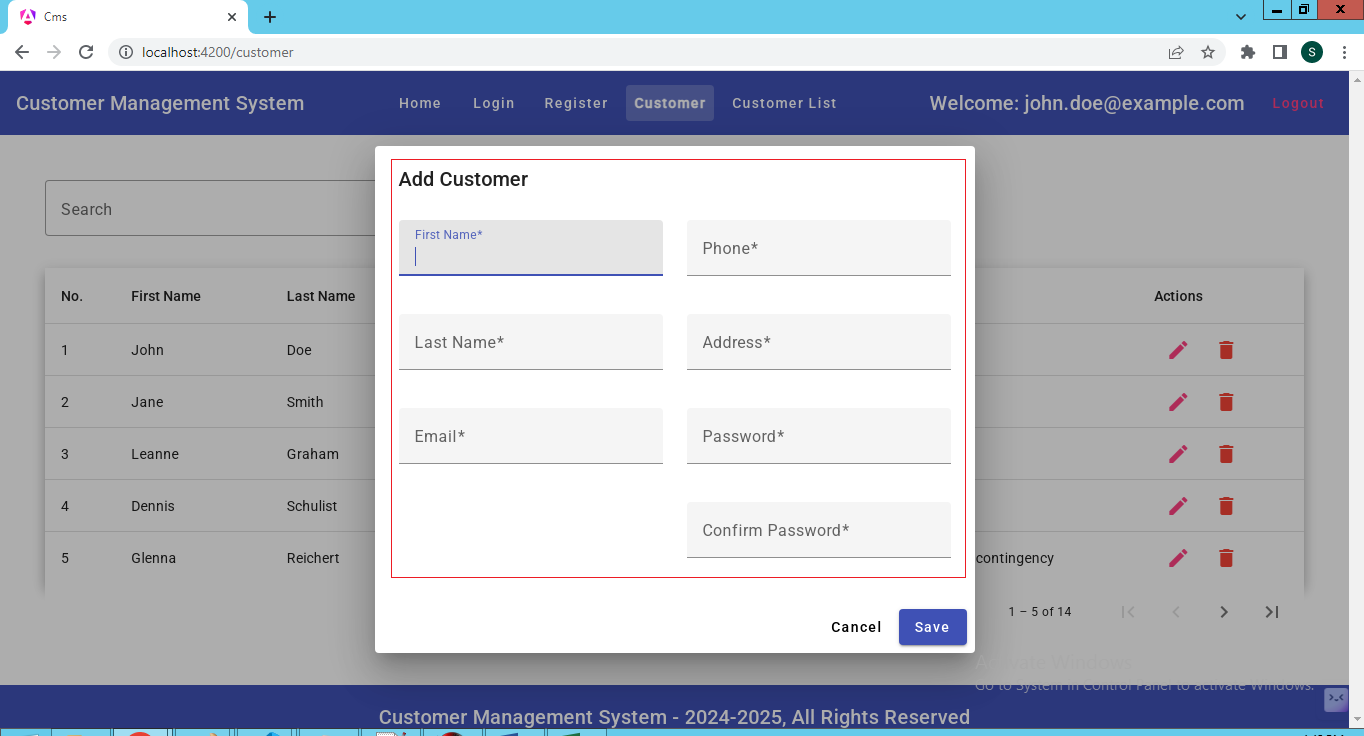
****

****

**Customer list value are edited by same page form**

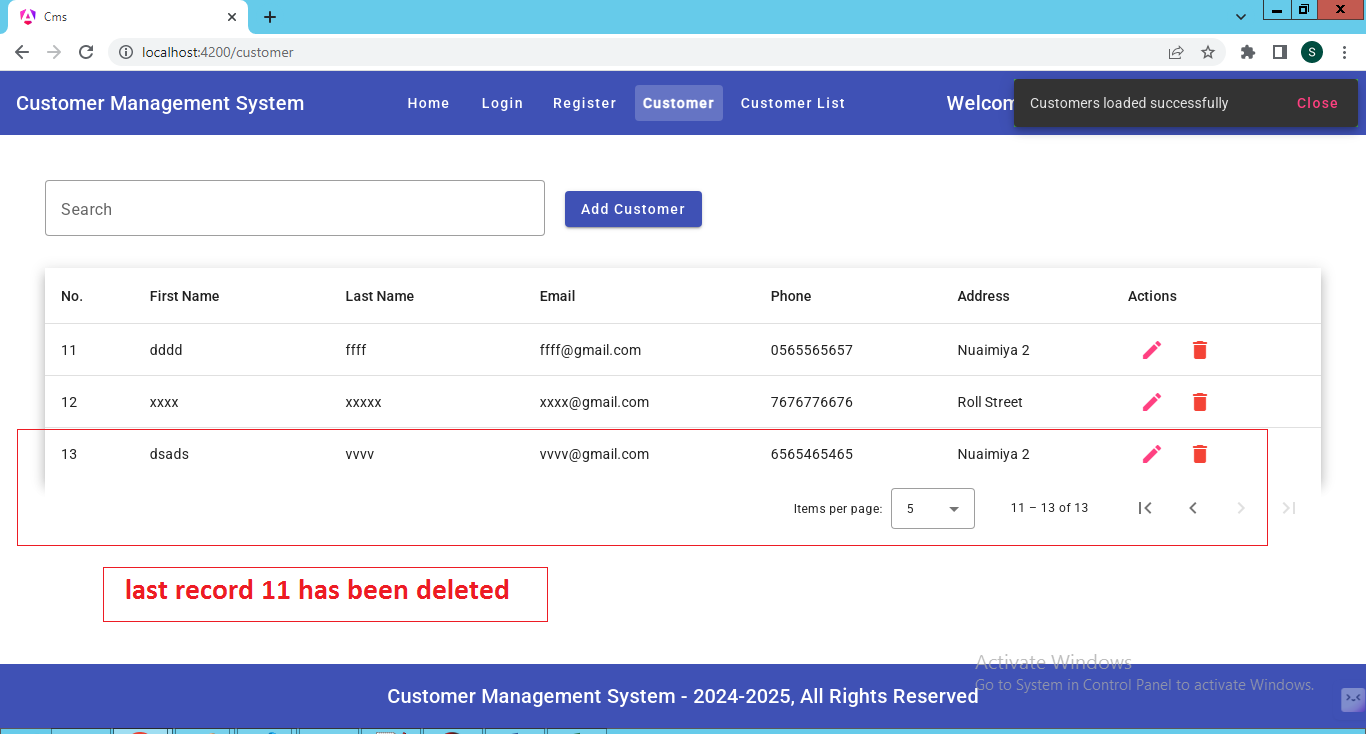
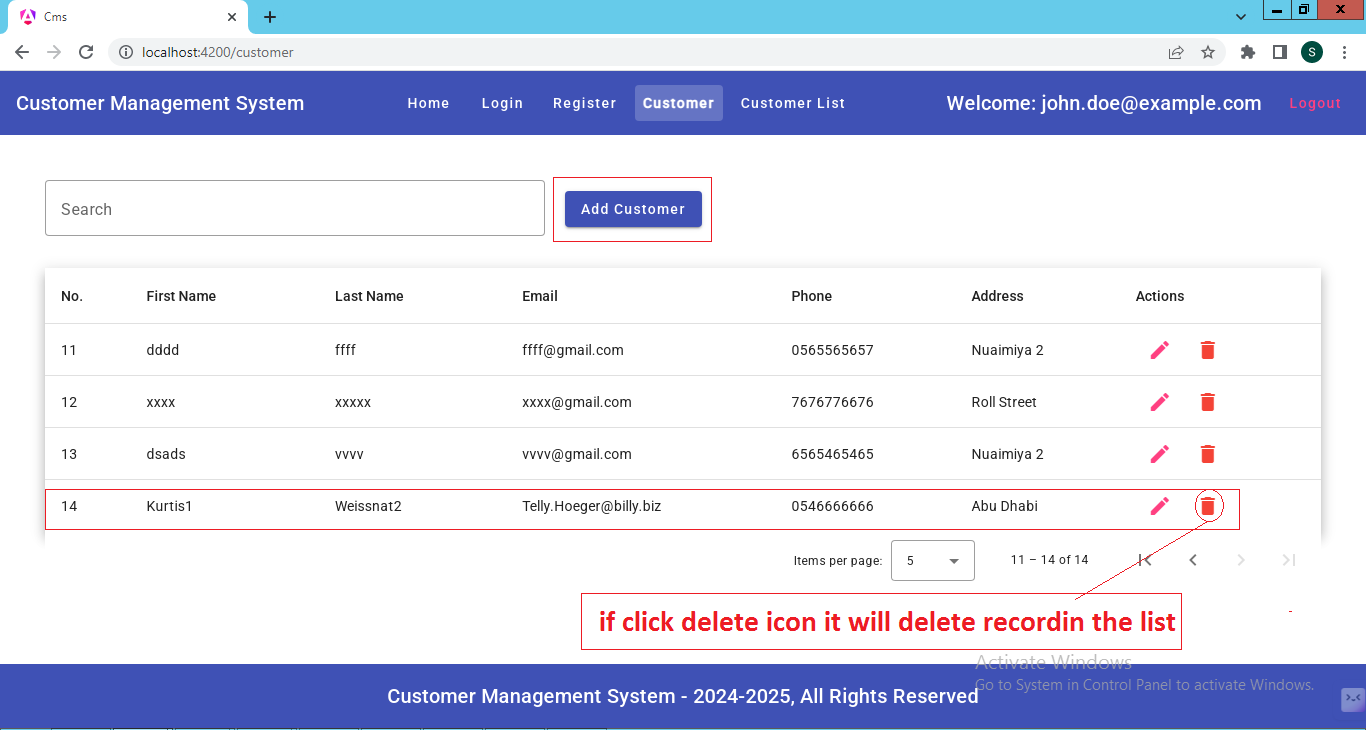
****

**🟥 Add Customer:**

****

**🟥 Delete Customer:**

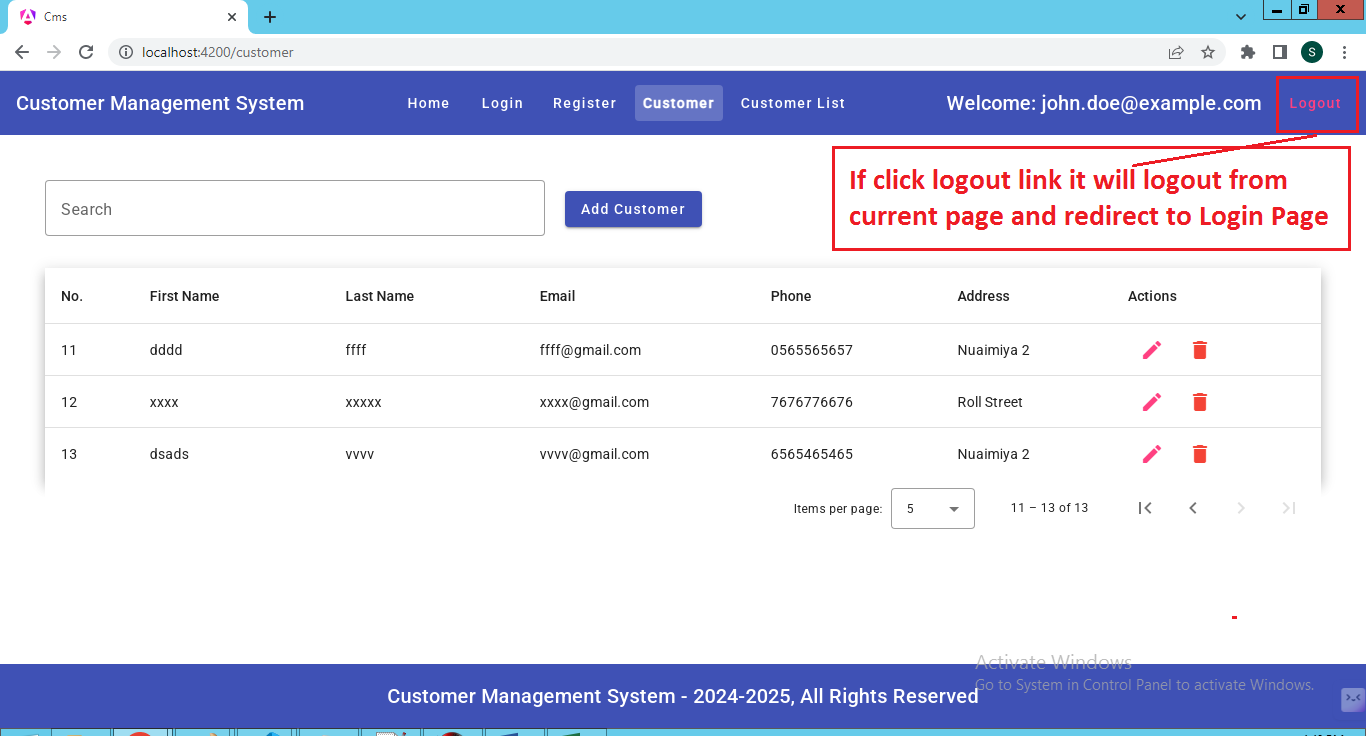
* Click "Delete" → confirmation prompt.
* Send DELETE request with customer ID.
* On success, remove entry from table.

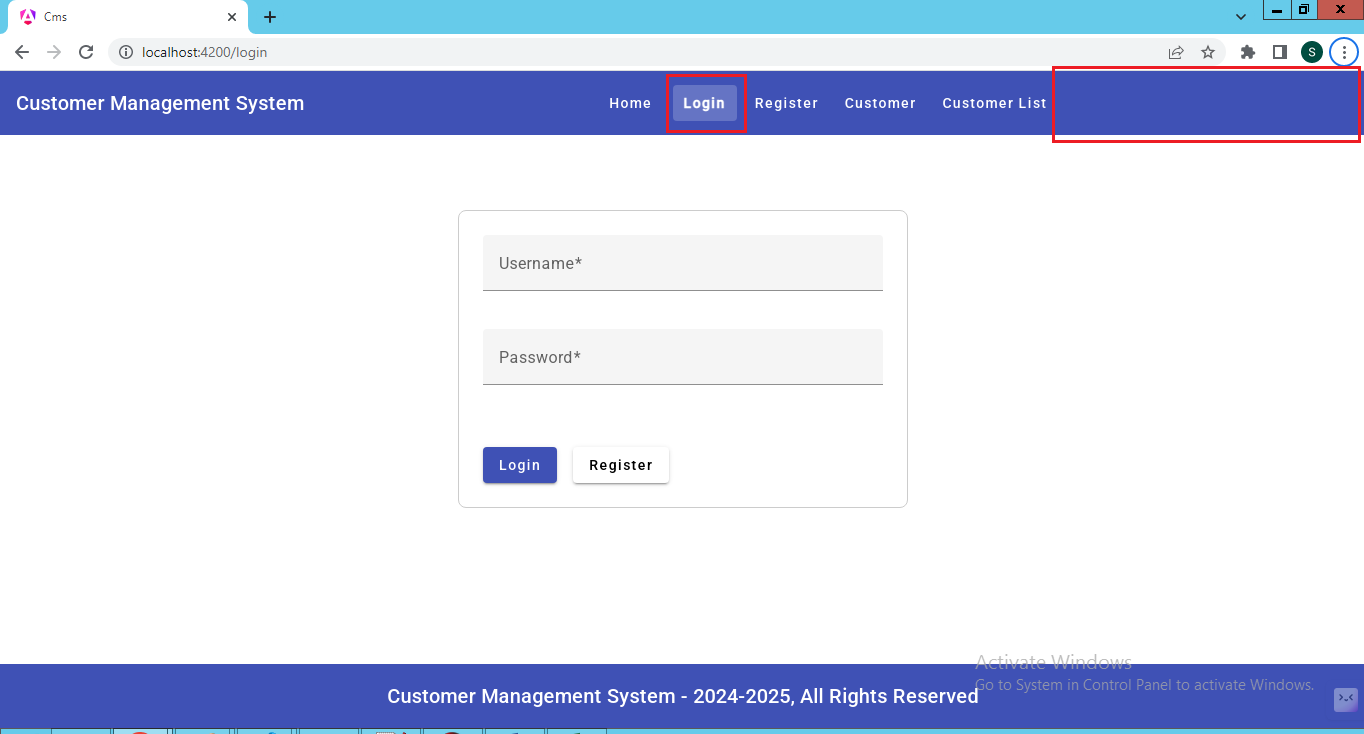


**✅ 4. Logout**

**✔️ Flow:**

* Click Logout → localStorage.removeItem('token')
* Redirect to /login





**🧪 Sample API Interaction Flow**

**Register API:**

POST /api/register

{

"firstName": "Abdul",

"lastName": "Rahman",

"email": "abdul.rahman@gmail.com",

"password": "Abdul@123",

"phone": "9876543210",

"address": "Sharjah"

}

**Login API:**

POST /api/login

{

"email": "abdul.rahman@gmail.com",

"password": "Abdul@123"

}

Response:

{

"token": "eyJhbGciOiJIUzI1NiIsInR..."

}

**Authenticated Request:**

GET /api/customers

Headers: { Authorization: `Bearer ${token}` }

**✅ Summary Table**

| **Page** | **Validations** | **API Call** | **On Success** |
| --- | --- | --- | --- |
| Register | Required fields, email, phone, password | POST /register | Show success, go to Login |
| Login | Required fields, invalid credentials | POST /login | Store JWT, go to Customer List |
| Customer List | Protected route via token | GET /customers | Render customer data in table |
| Edit Customer | Same validations as register | PUT /customers/:id | Update data in UI |
| Delete Customer | Click delete → confirm | DELETE /customers/:id | Remove from list |
| Logout | Clear token, redirect | N/A | Go to Login page |