Customer Management System – Frontend -User Guide

Frontend – React.js, Tailwind CSS with JWT Authentication

✅ React CMS Setup and Run Guide

◆ 1. Clone the Project from GitHub

Open a terminal and run:

```
git clone https://github.com/SheikGH/CMS_React_Frontend.git
```

Example:

```
git clone https://github.com/SheikGH/CMS_React_Frontend.git
```

◆ 2. Navigate to the Project Folder

```
cd CMS_React_Frontend
```

◆ 3. Create and Configure `.env` File

In the root of the project (inside `CMS_React_Frontend /`), create a file named `.env` if it doesn’t already exist.

Add the following line to `.env`:

```
REACT_APP_API_URL=https://localhost:7067/api
```

This will make sure your React 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`.

◆ 6. Start the React Development Server

```
npm start
```

This will start your React app at:

```
http://localhost:3000
```

◆ 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:

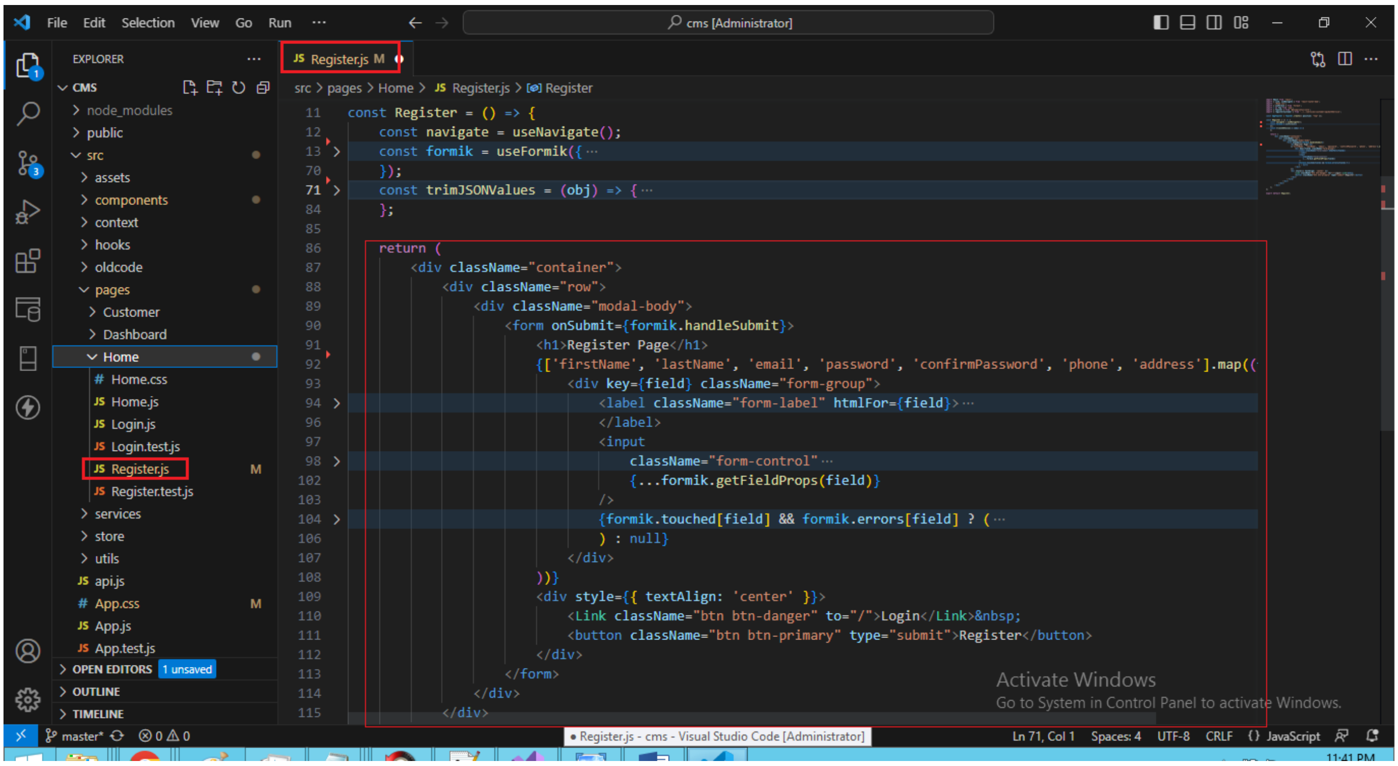
  ```
  services.AddCors(options =>
  {
      options.AddPolicy("AllowAll",
          builder =>
          {
              builder.AllowAnyOrigin()
                     .AllowAnyHeader()
                     .AllowAnyMethod();
          });
  });

  app.UseCors("AllowAll");
  ```

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

Frontend – React.js, Tailwind CSS with JWT Authentication

✓ 1. Register Page: http://localhost:3000/register



```
const Register = () => {
    const navigate = useNavigate();
    const formik = useFormik({ ···
    });
    const trimJSONValues = (obj) => { ···
    };

    return (
        <div className="container">
            <div className="row">
                <div className="modal-body">
                    <form onSubmit={formik.handleSubmit}>
                        <h1>Register Page</h1>
                        {['firstName', 'lastName', 'email', 'password', 'confirmPassword', 'phone', 'address'].map((
                            <div key={field} className="form-group">
                                <label className="form-label" htmlFor={field}> ···
                                </label>
                                <input
                                    className="form-control" ···
                                    {...formik.getFieldProps(field)}
                                />
                                {formik.touched[field] && formik.errors[field] ? ( ···
                                ) : null}
                            </div>
                        ))}
                        <div style={{ textAlign: 'center' }}>
                            <Link className="btn btn-danger" to="/">Login</Link>&nbsp;
                            <button className="btn btn-primary" type="submit">Register</button>
                        </div>
                    </form>
                </div>
            </div>
```

📌 Features:

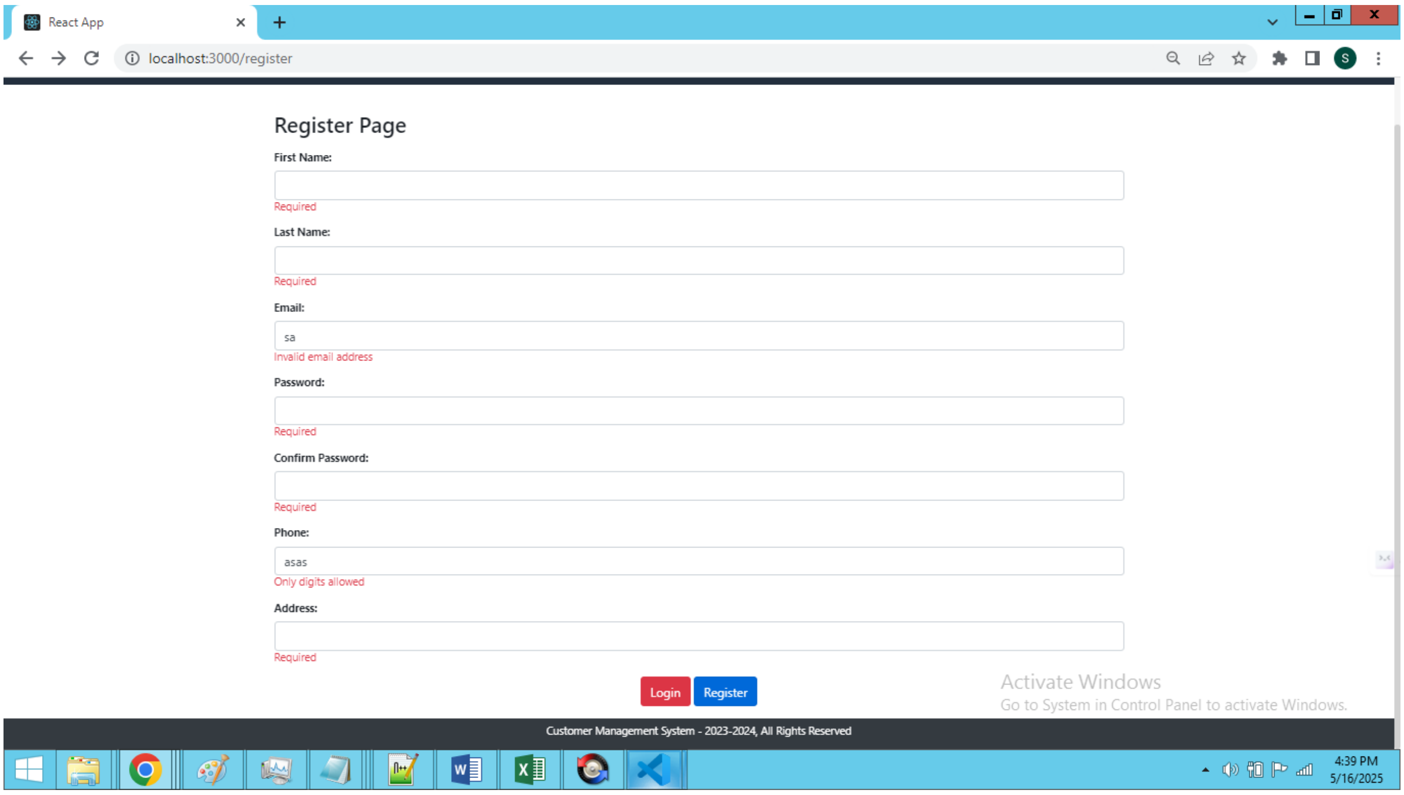
- Add a new customer with proper form validation.
- Submit valid customer data to the backend API and store it in the database.

✓ 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.


React App

localhost:3000/register

Register Page

First Name:

Required

Last Name:

Required

Email:

sa

Invalid email address

Password:

Required

Confirm Password:

Required

Phone:

asas

Only digits allowed

Address:

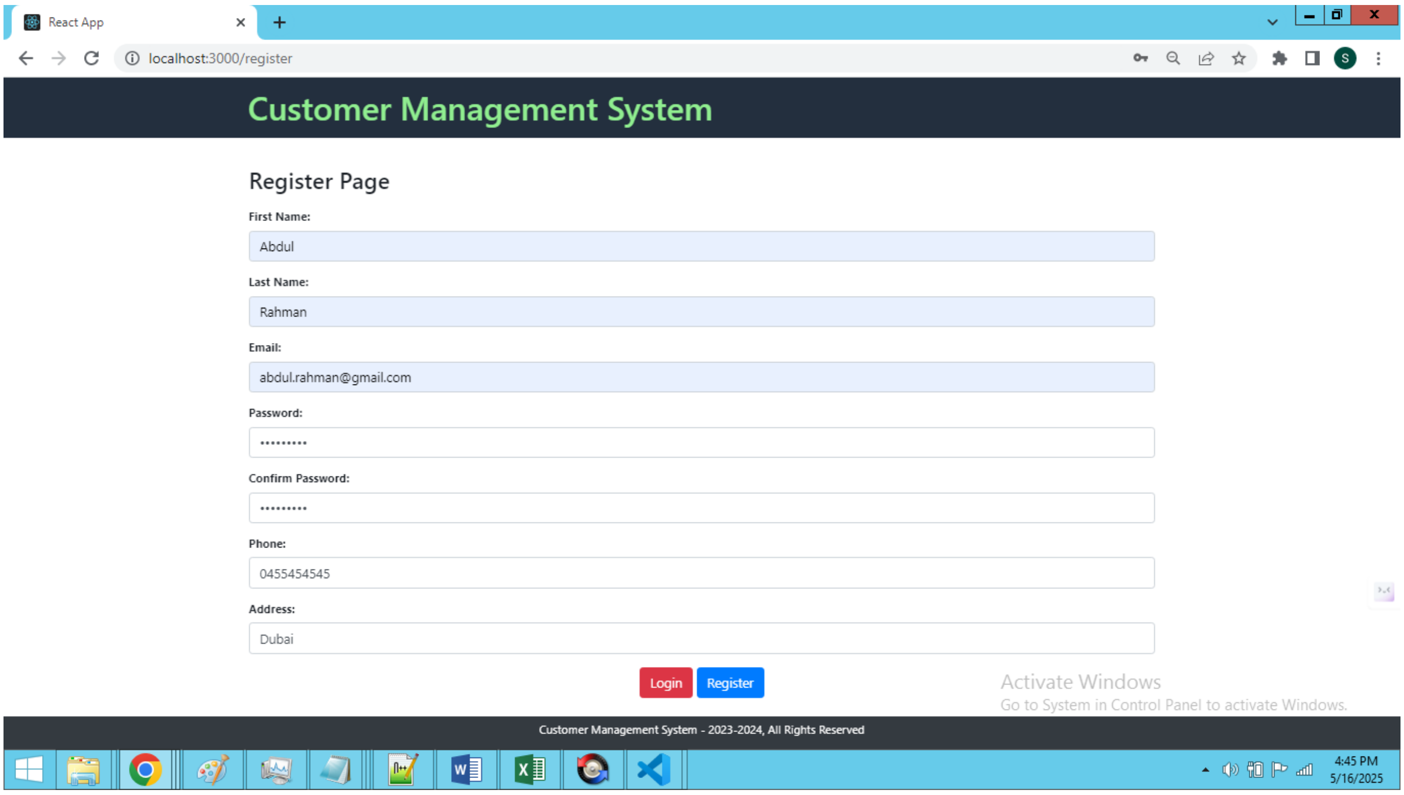
Required

Login Register

Activate Windows
Go to System in Control Panel to activate Windows.

Customer Management System - 2023-2024, All Rights Reserved

4:39 PM
5/16/2025


React App

localhost:3000/register

Customer Management System

Register Page

First Name:

Abdul

Last Name:

Rahman

Email:

abdul.rahman@gmail.com

Password:

•••••••••

Confirm Password:

•••••••••

Phone:

0455454545

Address:

Dubai

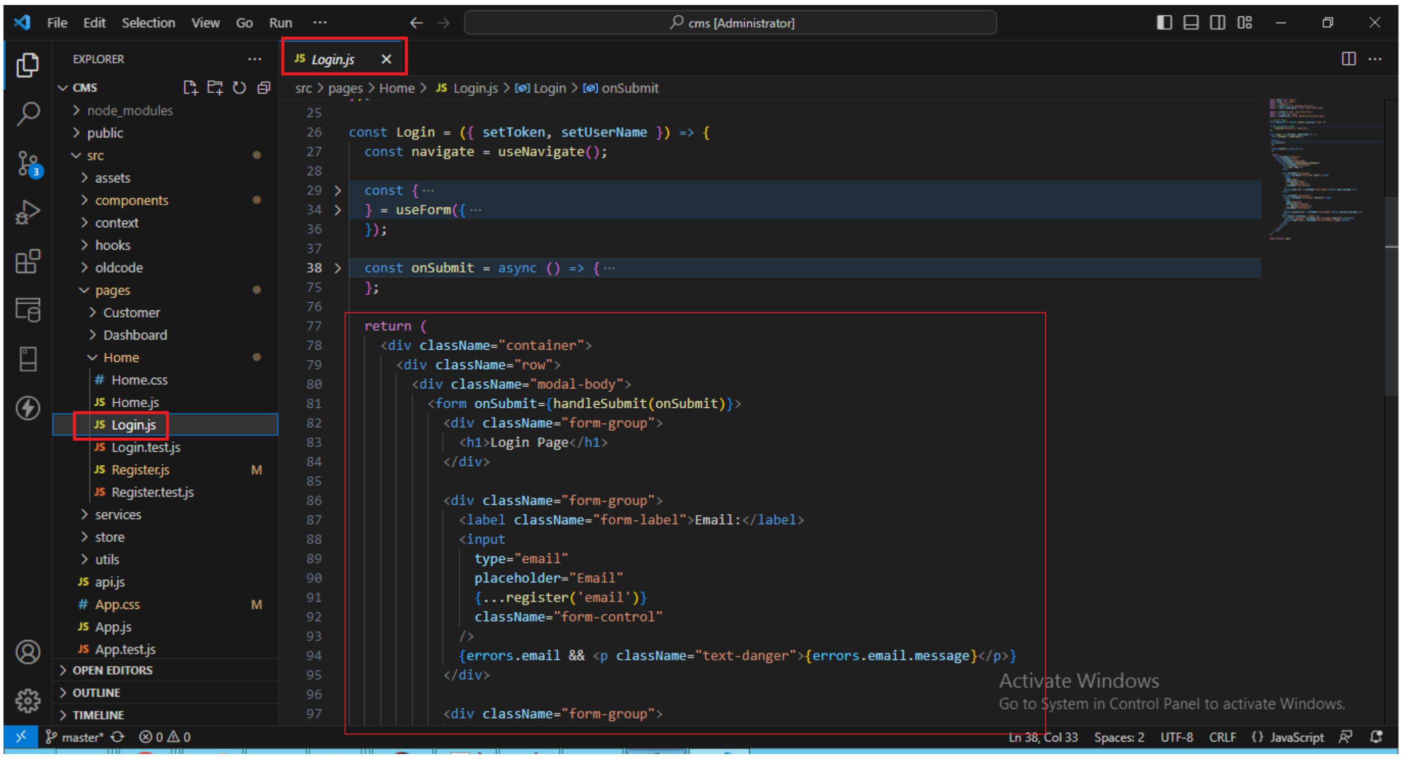
Login Register

Activate Windows
Go to System in Control Panel to activate Windows.

Customer Management System - 2023-2024, All Rights Reserved

4:45 PM
5/16/2025

✓ 2. Login Page: `http://localhost:3000/login`



```
const Login = ({ setToken, setUserName }) => {
  const navigate = useNavigate();

  const {⋯
  } = useForm({⋯
  });

  const onSubmit = async () => {⋯
  };

  return (
    <div className="container">
      <div className="row">
        <div className="modal-body">
          <form onSubmit={handleSubmit(onSubmit)}>
            <div className="form-group">
              <h1>Login Page</h1>
            </div>

            <div className="form-group">
              <label className="form-label">Email:</label>
              <input
                type="email"
                placeholder="Email"
                {...register('email')}
                className="form-control"
              />
              {errors.email && <p className="text-danger">{errors.email.message}</p>}
            </div>

            <div className="form-group">
```

📌 Features:

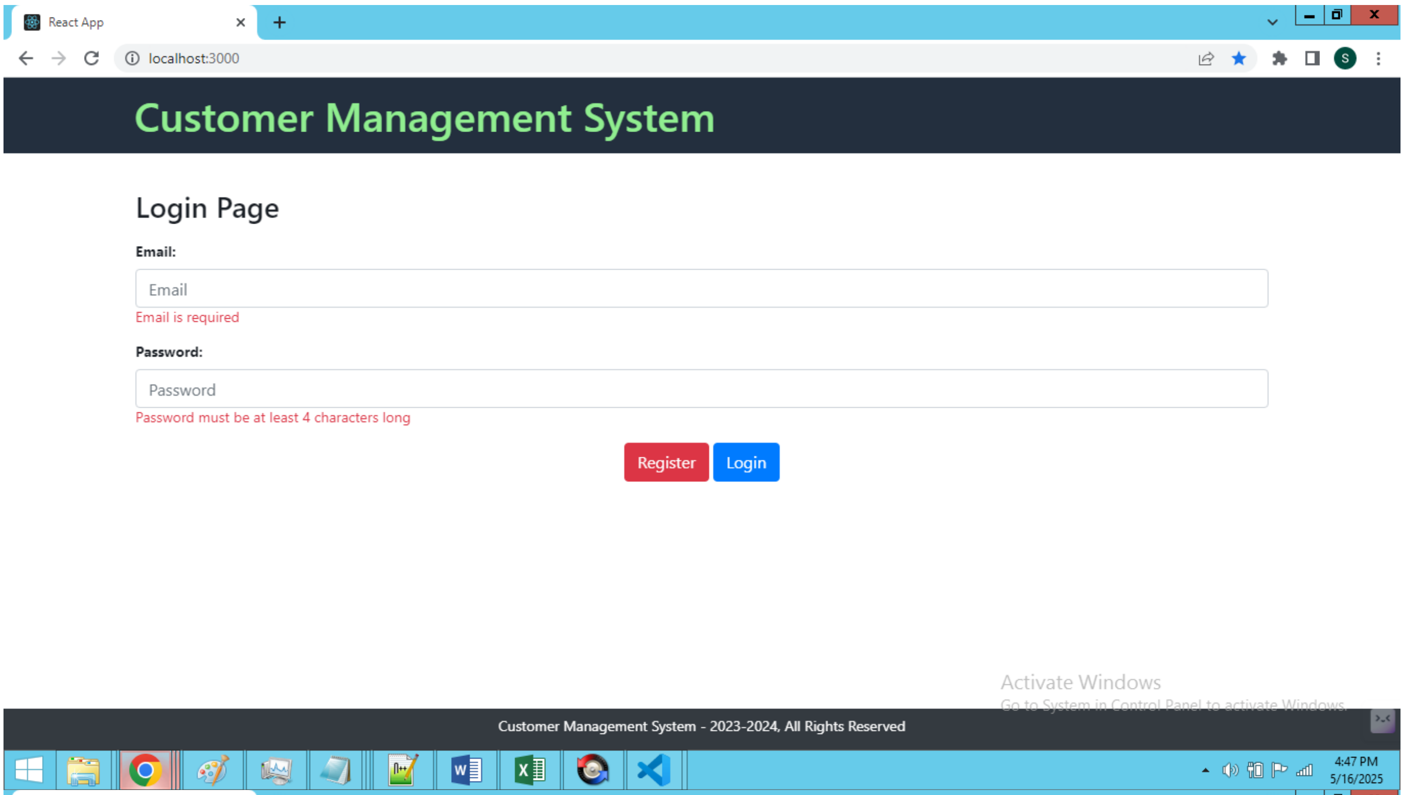
- Authenticate user using email and password.
- Receive JWT from API and store in localStorage.

✔️ 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`.



React App

localhost:3000

Customer Management System

Login Page

Email:

Email

Email is required

Password:

Password

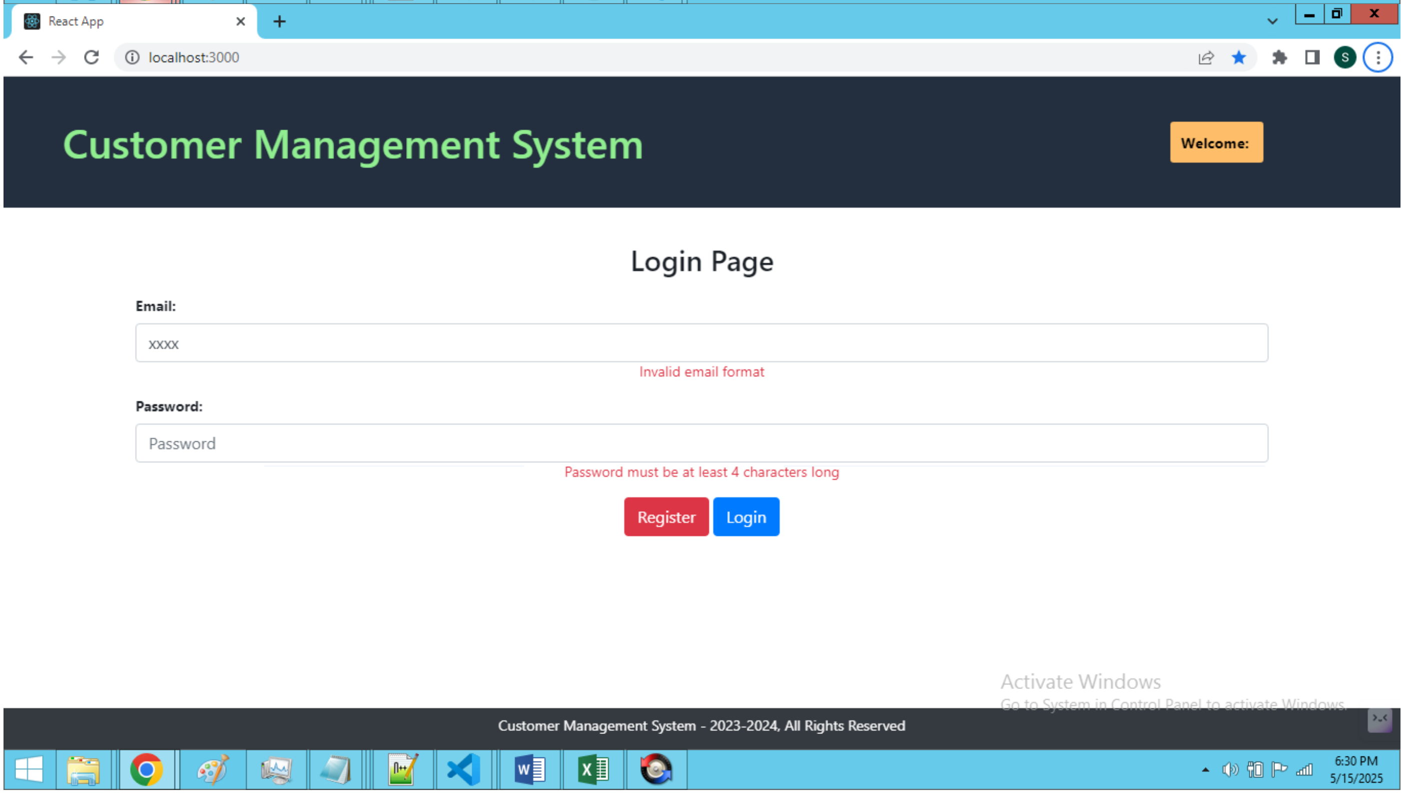
Password must be at least 4 characters long

Register Login

Activate Windows
Go to System in Control Panel to activate Windows.

Customer Management System - 2023-2024, All Rights Reserved

4:47 PM
5/16/2025



React App

localhost:3000

Customer Management System

Welcome:

Login Page

Email:

xxxx

Invalid email format

Password:

Password

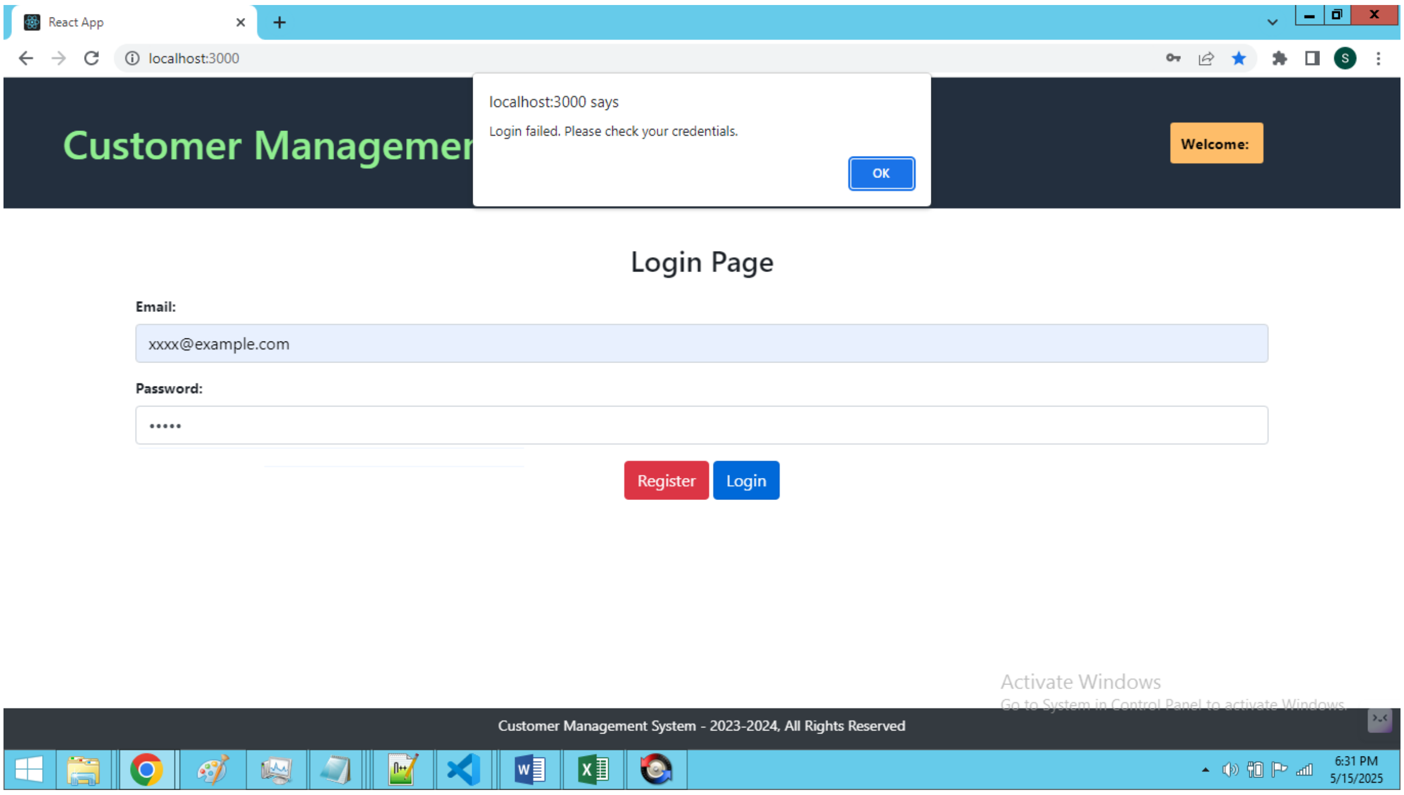
Password must be at least 4 characters long

Register Login

Activate Windows
Go to System in Control Panel to activate Windows.

Customer Management System - 2023-2024, All Rights Reserved

6:30 PM
5/15/2025



React App

localhost:3000

localhost:3000 says

Login failed. Please check your credentials.

OK

Customer Managemen

Welcome:

Login Page

Email:

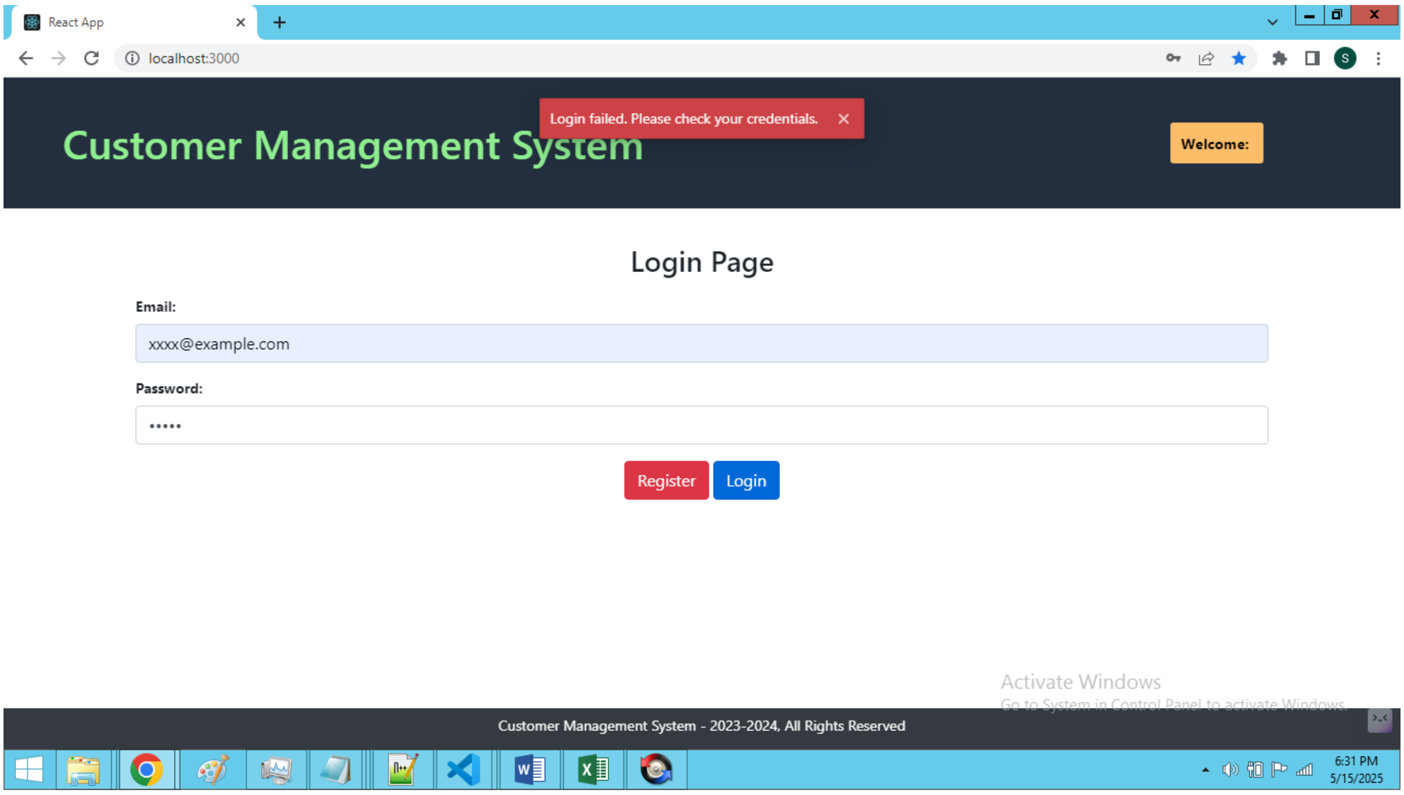
xxxx@example.com

Password:

•••••

Register Login

Customer Management System - 2023-2024, All Rights Reserved



React App

localhost:3000

Login failed. Please check your credentials.

Customer Management System

Welcome:

Login Page

Email:

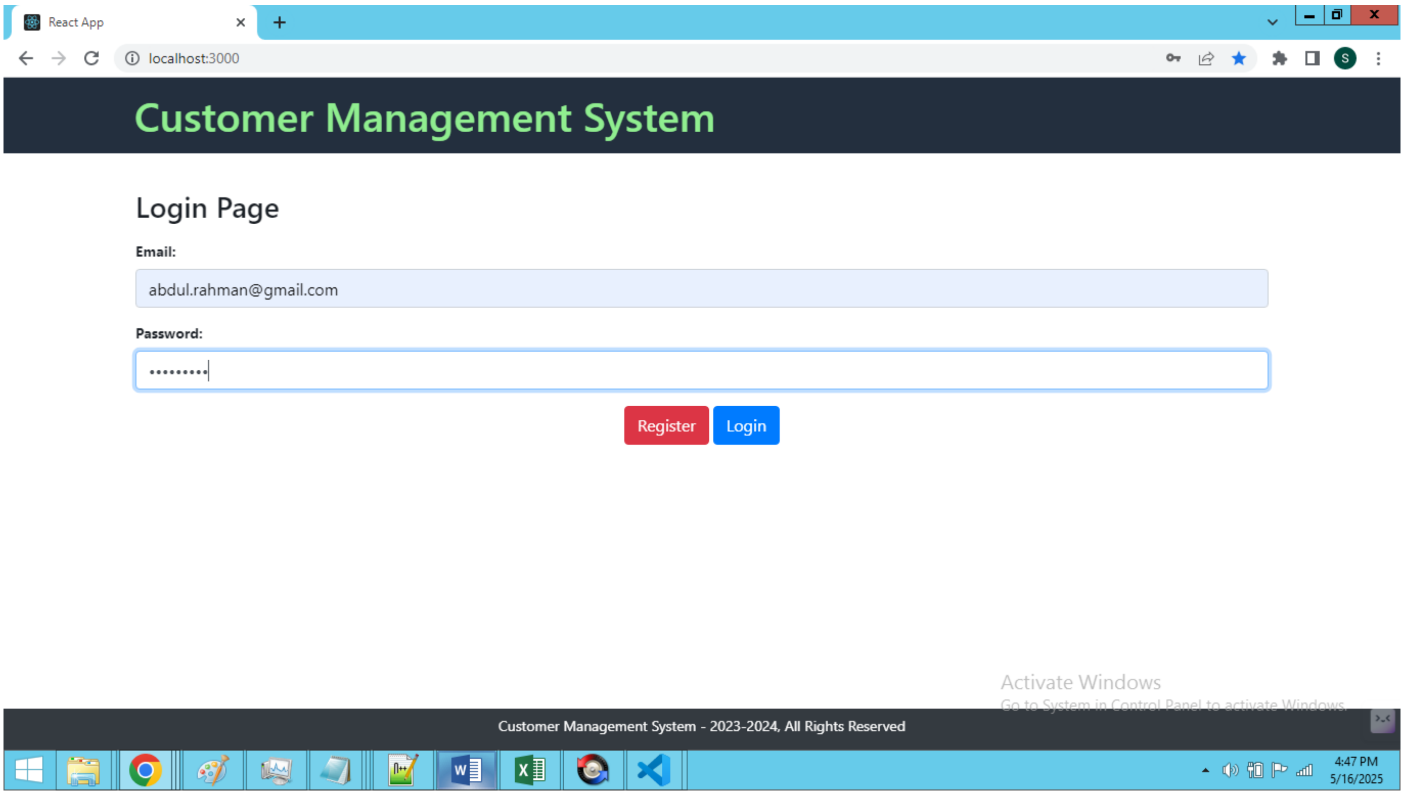
xxxx@example.com

Password:

•••••

Register Login

Customer Management System - 2023-2024, All Rights Reserved


React App

localhost:3000

Customer Management System

Login Page

Email:

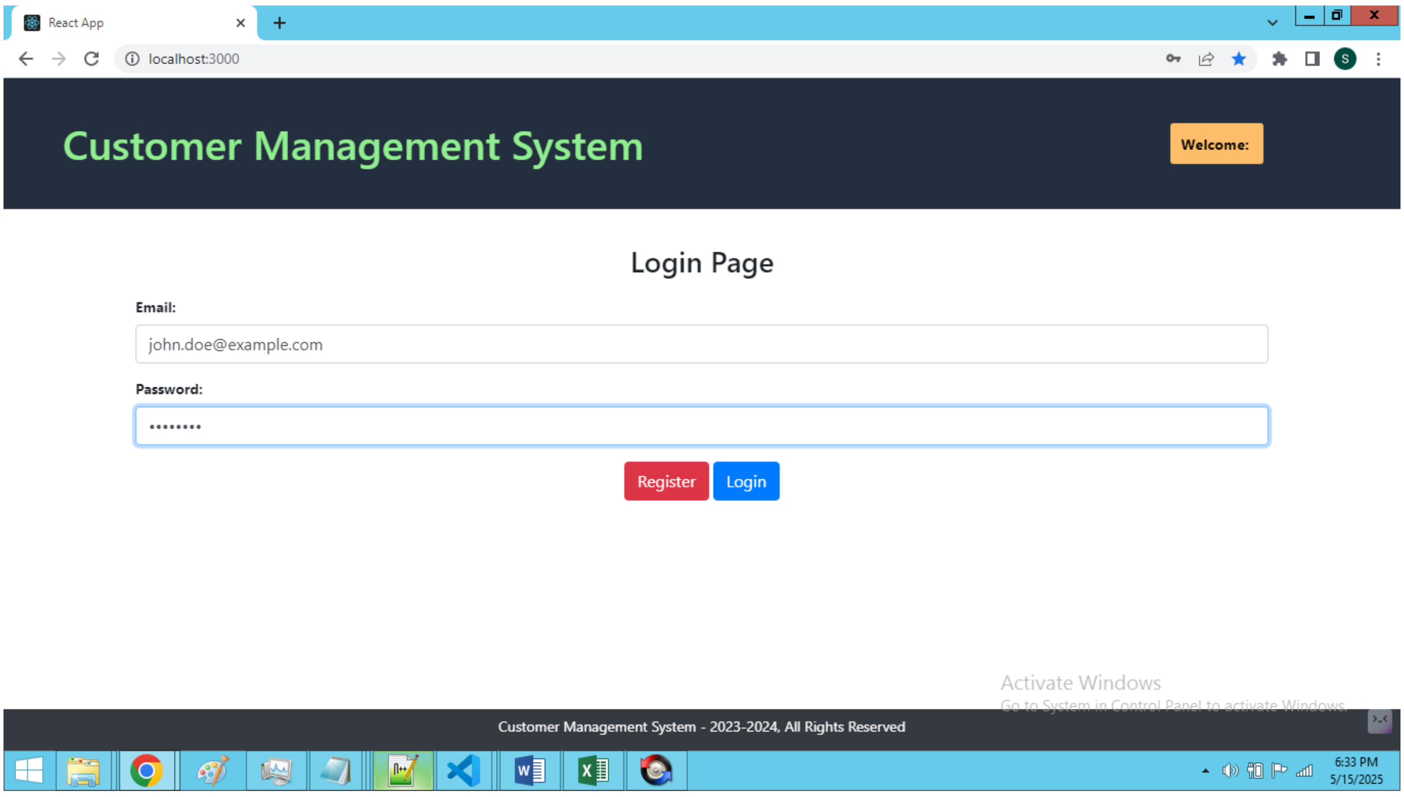
abdul.rahman@gmail.com

Password:

•••••••••

Register Login

Customer Management System - 2023-2024, All Rights Reserved


React App

localhost:3000

Customer Management System

Welcome:

Login Page

Email:

john.doe@example.com

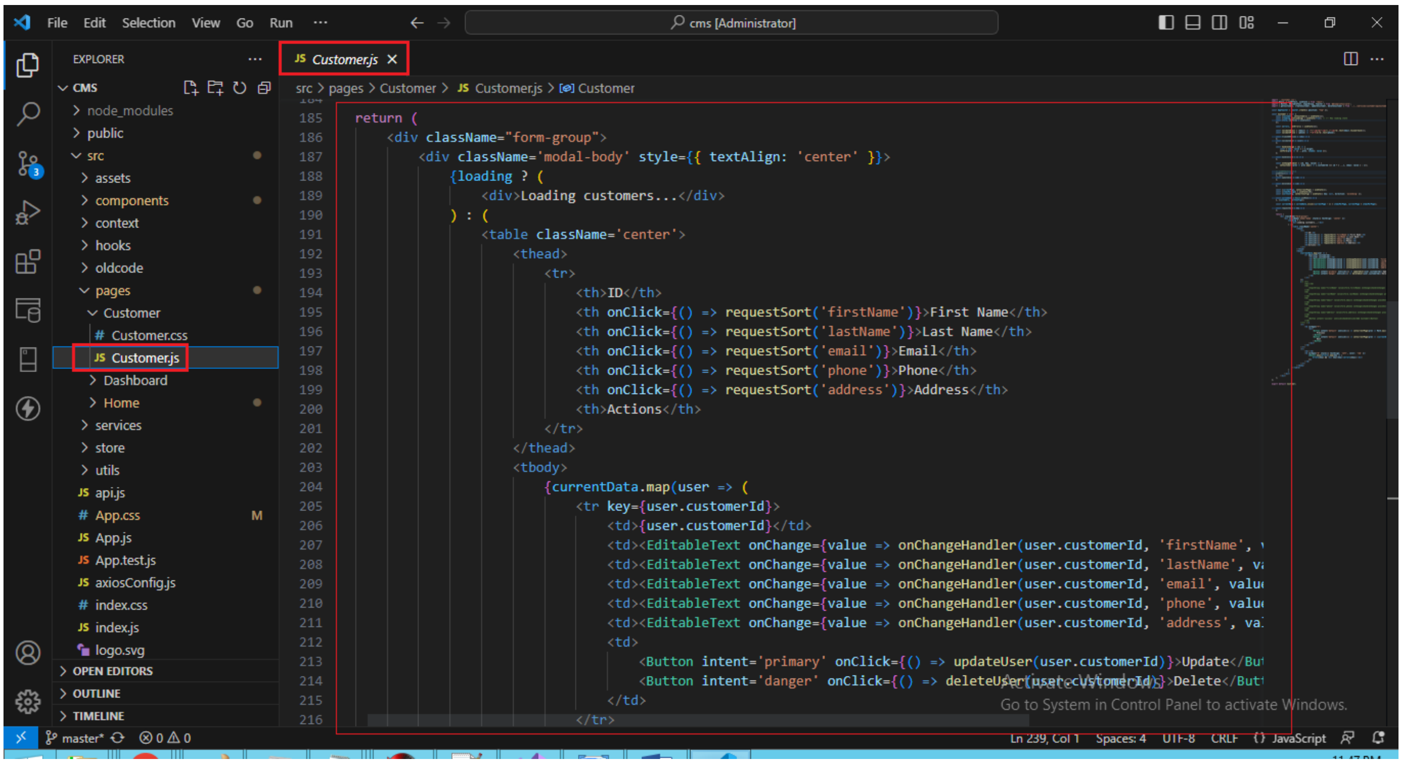
Password:

••••••••

Register Login

Customer Management System - 2023-2024, All Rights Reserved

✓ 3. Customer List Page: `http://localhost:3000/customer-list`



```
return (
    <div className="form-group">
        <div className='modal-body' style={{ textAlign: 'center' }}>
            {loading ? (
                <div>Loading customers...</div>
            ) : (
                <table className='center'>
                    <thead>
                        <tr>
                            <th>ID</th>
                            <th onClick={() => requestSort('firstName')}>First Name</th>
                            <th onClick={() => requestSort('lastName')}>Last Name</th>
                            <th onClick={() => requestSort('email')}>Email</th>
                            <th onClick={() => requestSort('phone')}>Phone</th>
                            <th onClick={() => requestSort('address')}>Address</th>
                            <th>Actions</th>
                        </tr>
                    </thead>
                    <tbody>
                        {currentData.map(user => (
                            <tr key={user.customerId}>
                                <td>{user.customerId}</td>
                                <td><EditableText onChange={value => onChangeHandler(user.customerId, 'firstName', 
                                <td><EditableText onChange={value => onChangeHandler(user.customerId, 'lastName', v
                                <td><EditableText onChange={value => onChangeHandler(user.customerId, 'email', valu
                                <td><EditableText onChange={value => onChangeHandler(user.customerId, 'phone', valu
                                <td><EditableText onChange={value => onChangeHandler(user.customerId, 'address', va
                                <td>
                                    <Button intent='primary' onClick={() => updateUser(user.customerId)}>Update</Bu
                                    <Button intent='danger' onClick={() => deleteUser(user.customerId)}>Delete</But
                                </td>
                            </tr>
```

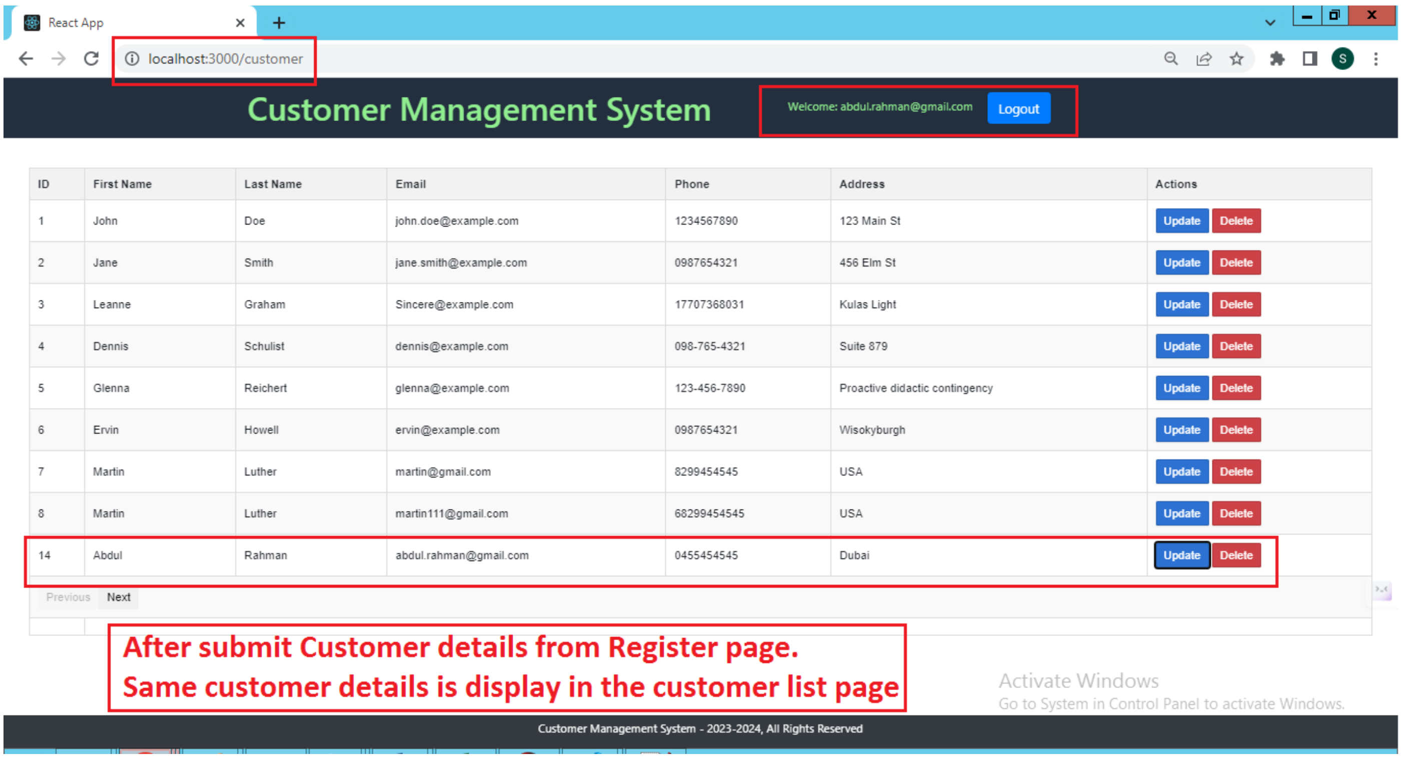
📌 Features:

- 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.



localhost:3000/customer

Customer Management System

Welcome: abdul.rahman@gmail.com Logout

ID	First Name	Last Name	Email	Phone	Address	Actions
1	John	Doe	john.doe@example.com	1234567890	123 Main St	Update Delete
2	Jane	Smith	jane.smith@example.com	0987654321	456 Elm St	Update Delete
3	Leanne	Graham	Sincere@example.com	17707368031	Kulas Light	Update Delete
4	Dennis	Schulist	dennis@example.com	098-765-4321	Suite 879	Update Delete
5	Glenna	Reichert	glenna@example.com	123-456-7890	Proactive didactic contingency	Update Delete
6	Ervin	Howell	ervin@example.com	0987654321	Wisokyburgh	Update Delete
7	Martin	Luther	martin@gmail.com	8299454545	USA	Update Delete
8	Martin	Luther	martin111@gmail.com	68299454545	USA	Update Delete
14	Abdul	Rahman	abdul.rahman@gmail.com	0455454545	Dubai	Update Delete

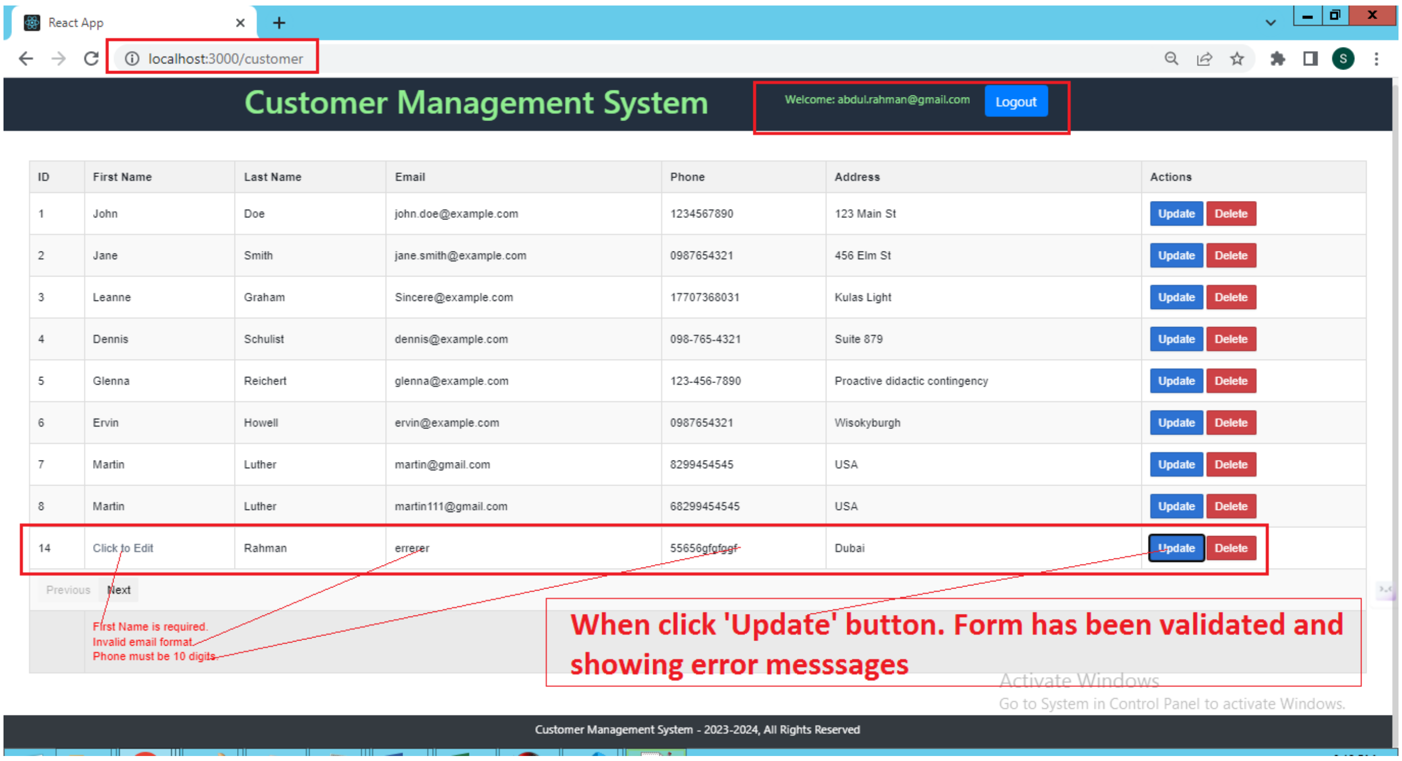
Previous Next

After submit Customer details from Register page.
Same customer details is display in the customer list page

Customer Management System - 2023-2024, All Rights Reserved

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.


React App

localhost:3000/customer

Customer Management System

Welcome: abdul.rahman@gmail.com Logout

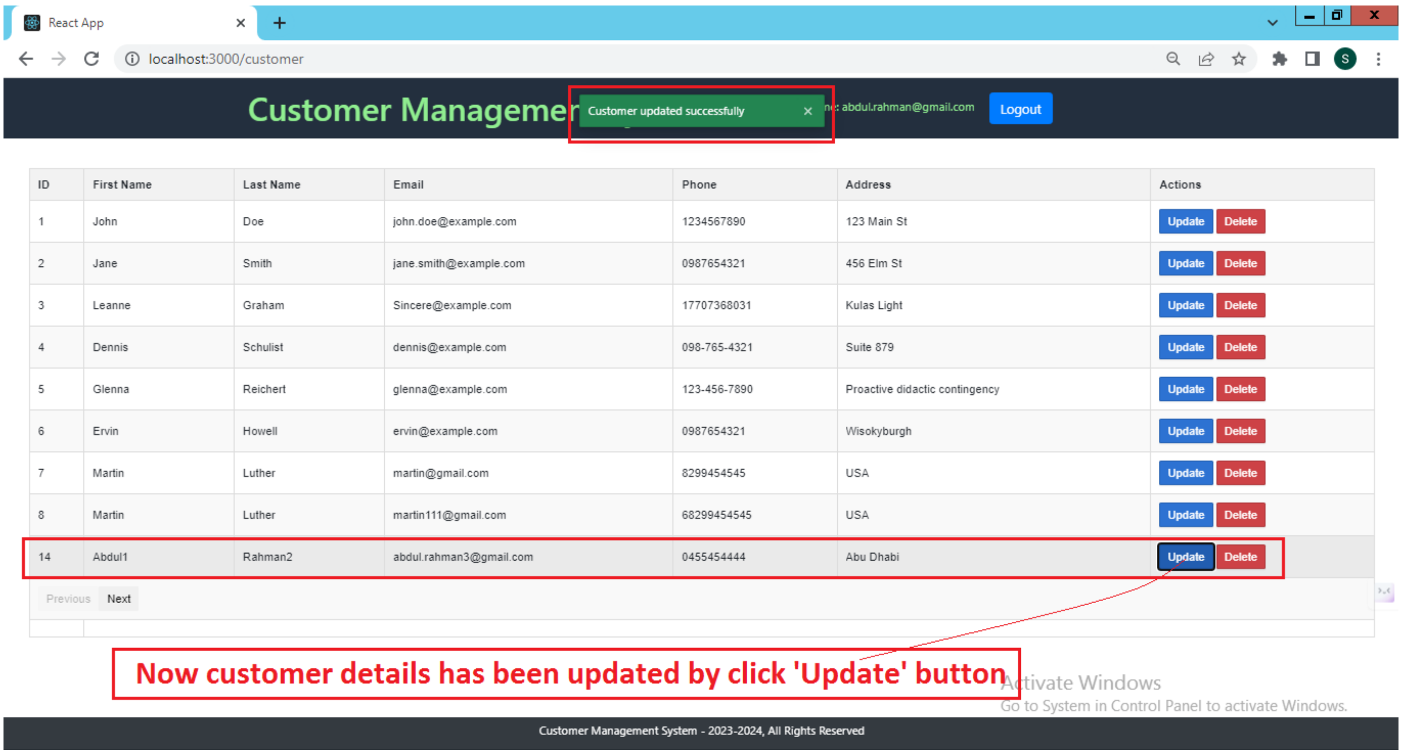
ID	First Name	Last Name	Email	Phone	Address	Actions
1	John	Doe	john.doe@example.com	1234567890	123 Main St	Update Delete
2	Jane	Smith	jane.smith@example.com	0987654321	456 Elm St	Update Delete
3	Leanne	Graham	Sincere@example.com	17707368031	Kulas Light	Update Delete
4	Dennis	Schulist	dennis@example.com	098-765-4321	Suite 879	Update Delete
5	Glenna	Reichert	glenna@example.com	123-456-7890	Proactive didactic contingency	Update Delete
6	Ervin	Howell	ervin@example.com	0987654321	Wisokyburgh	Update Delete
7	Martin	Luther	martin@gmail.com	8299454545	USA	Update Delete
8	Martin	Luther	martin111@gmail.com	68299454545	USA	Update Delete
14	Click to Edit	Rahman	errerer	55656gfgfggf	Dubai	Update Delete

Previous Next

First Name is required.
Invalid email format.
Phone must be 10 digits.

When click 'Update' button. Form has been validated and showing error messsages

Customer Management System - 2023-2024, All Rights Reserved


React App

localhost:3000/customer

Customer Managemen

Customer updated successfully ×

abdul.rahman@gmail.com Logout

ID	First Name	Last Name	Email	Phone	Address	Actions
1	John	Doe	john.doe@example.com	1234567890	123 Main St	Update Delete
2	Jane	Smith	jane.smith@example.com	0987654321	456 Elm St	Update Delete
3	Leanne	Graham	Sincere@example.com	17707368031	Kulas Light	Update Delete
4	Dennis	Schulist	dennis@example.com	098-765-4321	Suite 879	Update Delete
5	Glenna	Reichert	glenna@example.com	123-456-7890	Proactive didactic contingency	Update Delete
6	Ervin	Howell	ervin@example.com	0987654321	Wisokyburgh	Update Delete
7	Martin	Luther	martin@gmail.com	8299454545	USA	Update Delete
8	Martin	Luther	martin111@gmail.com	68299454545	USA	Update Delete
14	Abdul1	Rahman2	abdul.rahman3@gmail.com	0455454444	Abu Dhabi	Update Delete

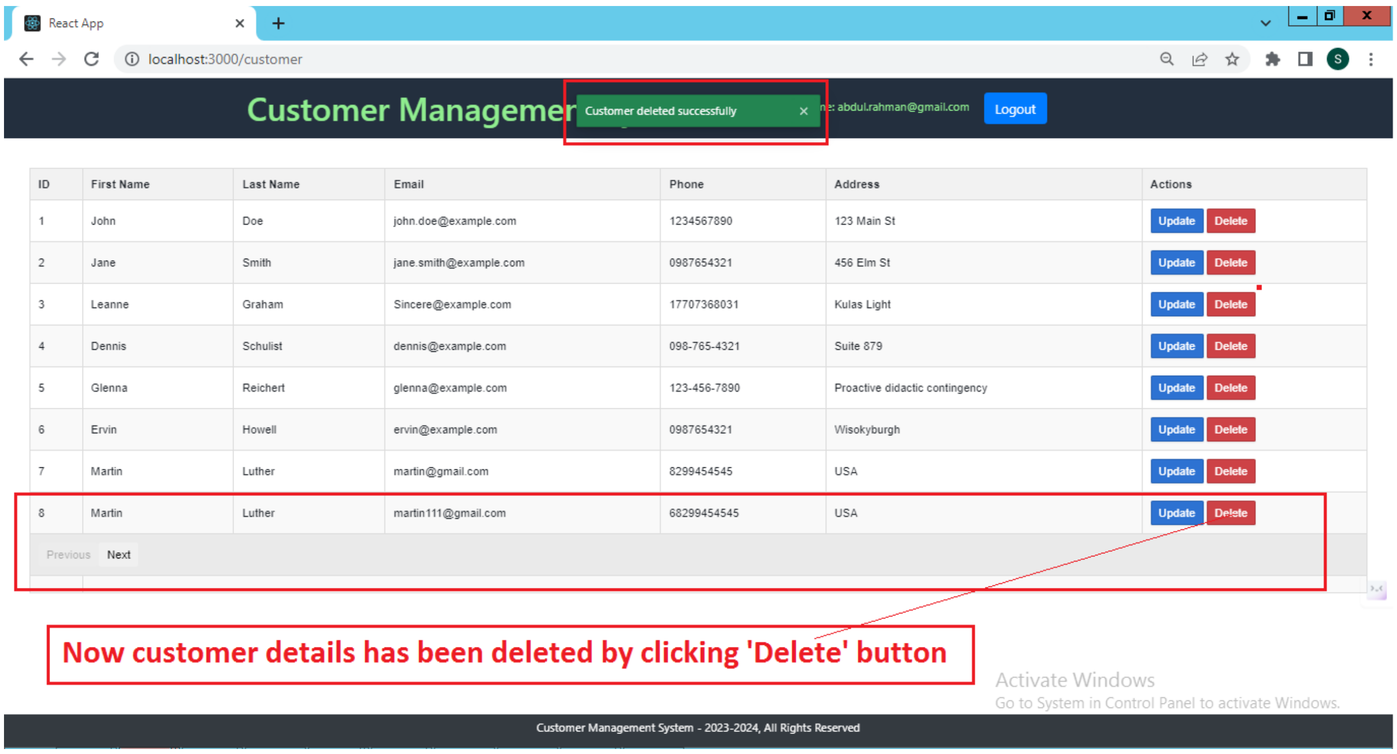
Previous Next

Now customer details has been updated by click 'Update' button

Customer Management System - 2023-2024, All Rights Reserved

Delete Customer:

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



ID	First Name	Last Name	Email	Phone	Address	Actions
1	John	Doe	john.doe@example.com	1234567890	123 Main St	Update Delete
2	Jane	Smith	jane.smith@example.com	0987654321	456 Elm St	Update Delete
3	Leanne	Graham	Sincere@example.com	17707368031	Kulas Light	Update Delete
4	Dennis	Schulist	dennis@example.com	098-765-4321	Suite 879	Update Delete
5	Glenna	Reichert	glenna@example.com	123-456-7890	Proactive didactic contingency	Update Delete
6	Ervin	Howell	ervin@example.com	0987654321	Wisokyburgh	Update Delete
7	Martin	Luther	martin@gmail.com	8299454545	USA	Update Delete
8	Martin	Luther	martin111@gmail.com	68299454545	USA	Update Delete

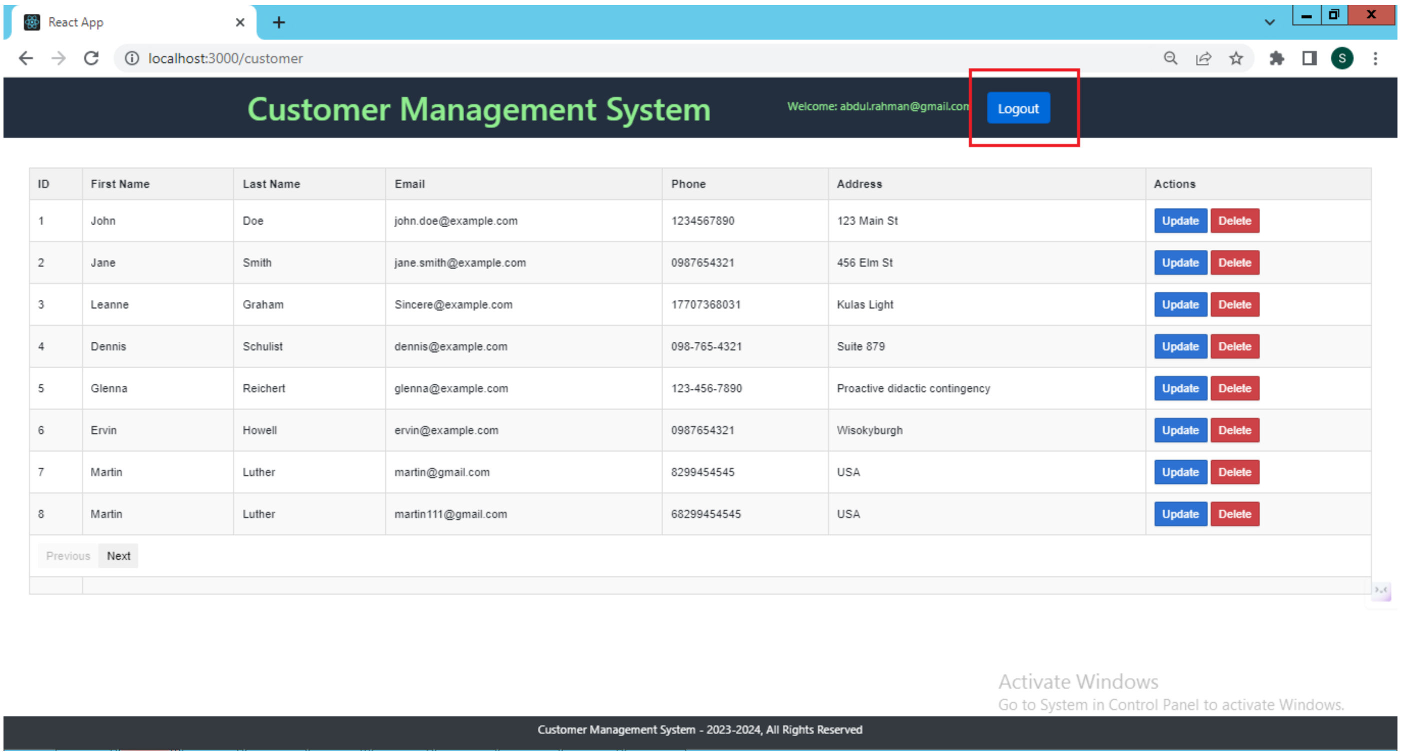
Now customer details has been deleted by clicking 'Delete' button

Customer Management System - 2023-2024, All Rights Reserved

✅ 4. Logout

✔️ Flow:

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



Customer Management System

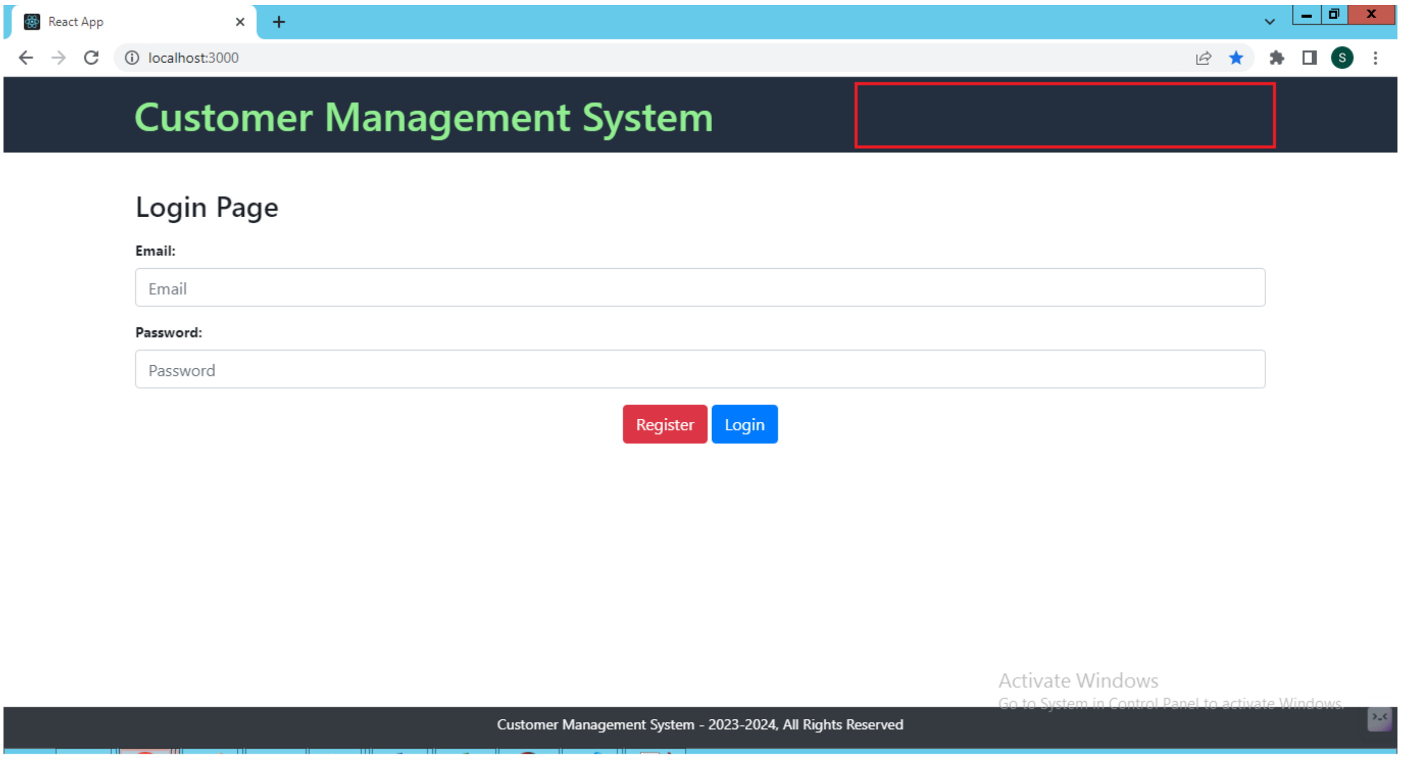
Welcome: abdul.rahman@gmail.com Logout

ID	First Name	Last Name	Email	Phone	Address	Actions
1	John	Doe	john.doe@example.com	1234567890	123 Main St	Update Delete
2	Jane	Smith	jane.smith@example.com	0987654321	456 Elm St	Update Delete
3	Leanne	Graham	Sincere@example.com	17707368031	Kulas Light	Update Delete
4	Dennis	Schulist	dennis@example.com	098-765-4321	Suite 879	Update Delete
5	Glenna	Reichert	glenna@example.com	123-456-7890	Proactive didactic contingency	Update Delete
6	Ervin	Howell	ervin@example.com	0987654321	Wisokyburgh	Update Delete
7	Martin	Luther	martin@gmail.com	8299454545	USA	Update Delete
8	Martin	Luther	martin111@gmail.com	68299454545	USA	Update Delete

Previous Next

Activate Windows
Go to System in Control Panel to activate Windows.

Customer Management System - 2023-2024, All Rights Reserved



Customer Management System

Login Page

Email:

Email

Password:

Password

Register Login

Activate Windows
Go to System in Control Panel to activate Windows.

Customer Management System - 2023-2024, All Rights Reserved

□ 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