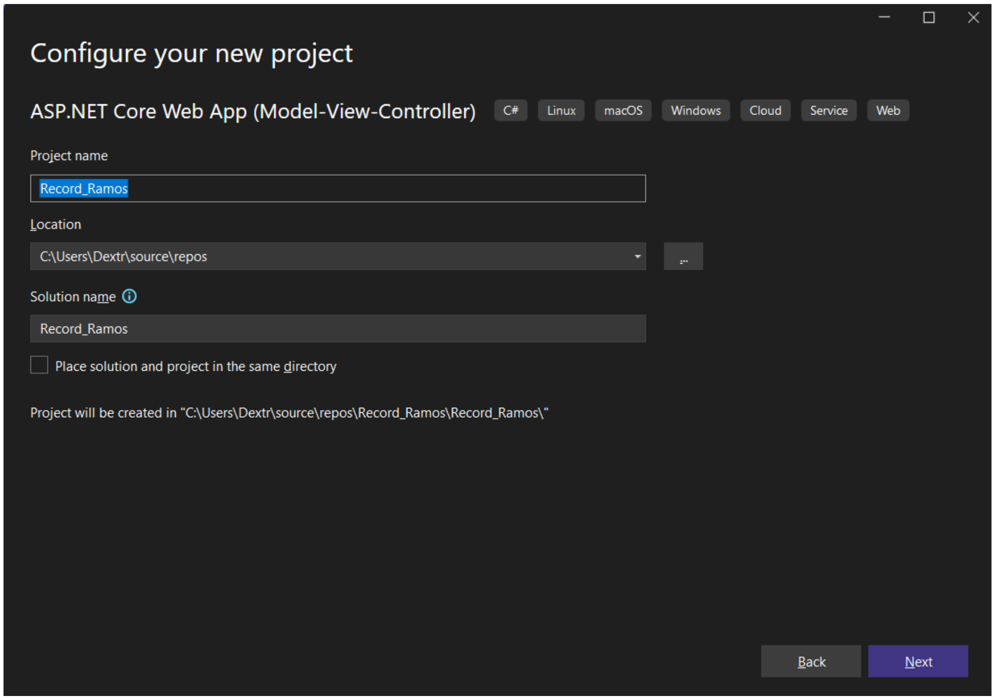
NAME: Dexter Ramos________________________ Subject Code:_8441_____ Time: 10:00-12:00____

Create a new ASP.NET Core Web Application:

- Use **ASP.NET Core MVC** template.
- Enable **Individual User Accounts** for authentication.
- Project Name: Record_LastName (Screenshot)



Configure your new project

ASP.NET Core Web App (Model-View-Controller)

Project name: Record_Ramos

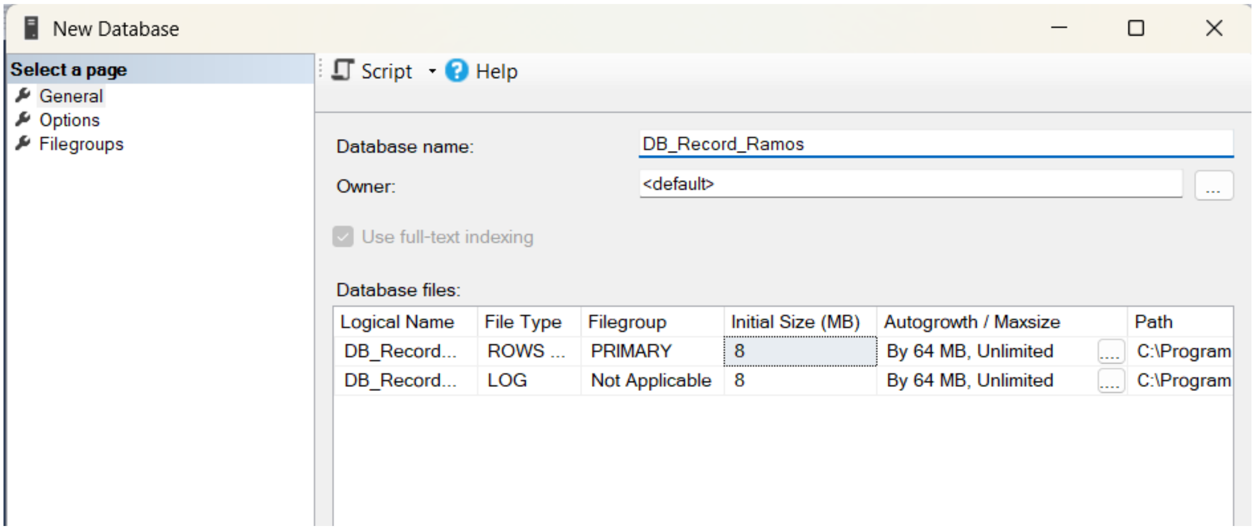
Location: C:\Users\Dextr\source\repos

Solution name: Record_Ramos

□ Place solution and project in the same directory

Project will be created in "C:\Users\Dextr\source\repos\Record_Ramos\Record_Ramos\"

- Set up a connection to **SQL Server (SSMS)** by configuring the `appsettings.json` file.
- Run migrations using `Add-Migration` and `Update-Database` to create Identity tables in SSMS.
- Database Name: DB_Record_LastName (Screenshot)



New Database

Database name: DB_Record_Ramos

Owner: <default>

☑ Use full-text indexing

Database files:

Logical Name	File Type	Filegroup	Initial Size (MB)	Autogrowth / Maxsize	Path
DB_Record...	ROWS ...	PRIMARY	8	By 64 MB, Unlimited	C:\Program
DB_Record...	LOG	Not Applicable	8	By 64 MB, Unlimited	C:\Program

Advanced Identity Page Customization

1. **Scaffold All Identity Pages**:
   - Scaffold all the Identity pages (Login, Register, Manage, Forgot Password, Reset Password).
   - Ensure that the following Identity views are scaffolded and available in the project:
     - `Login.cshtml`
     - `Register.cshtml`
     - `ForgotPassword.cshtml`
     - `ResetPassword.cshtml`
     - `Manage` pages for profile editing.
       - Layout
       - ChangePassword
       - PersonalData
       - Email
       - ResetAuthenticator
       - TwoFactorAuthentication
       - EnableAuthenticator
       - SetPassword

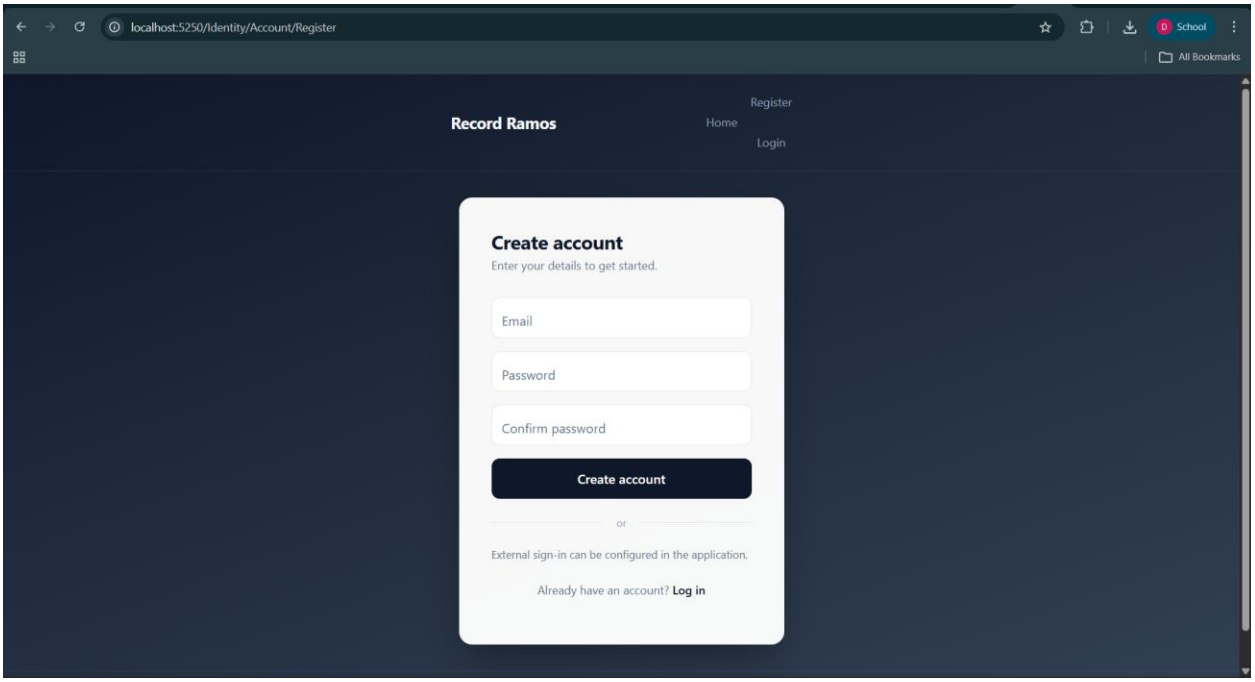
2. **Custom Layout for Identity Pages**:
   - Create a new **Identity-specific layout** to give these pages a unique look (separate from the main site layout if required).
   - You can name the layout something like `_IdentityLayout.cshtml` and apply it only to Identity-related views:

```
html
@{
    Layout = "~/Views/Shared/_IdentityLayout.cshtml";
}
```

   - Customize the navigation and branding for these Identity pages to give users a focused and consistent experience during authentication.

Deliverable:

- Screenshot of at least two Identity pages (Login and Register) with a unique layout.

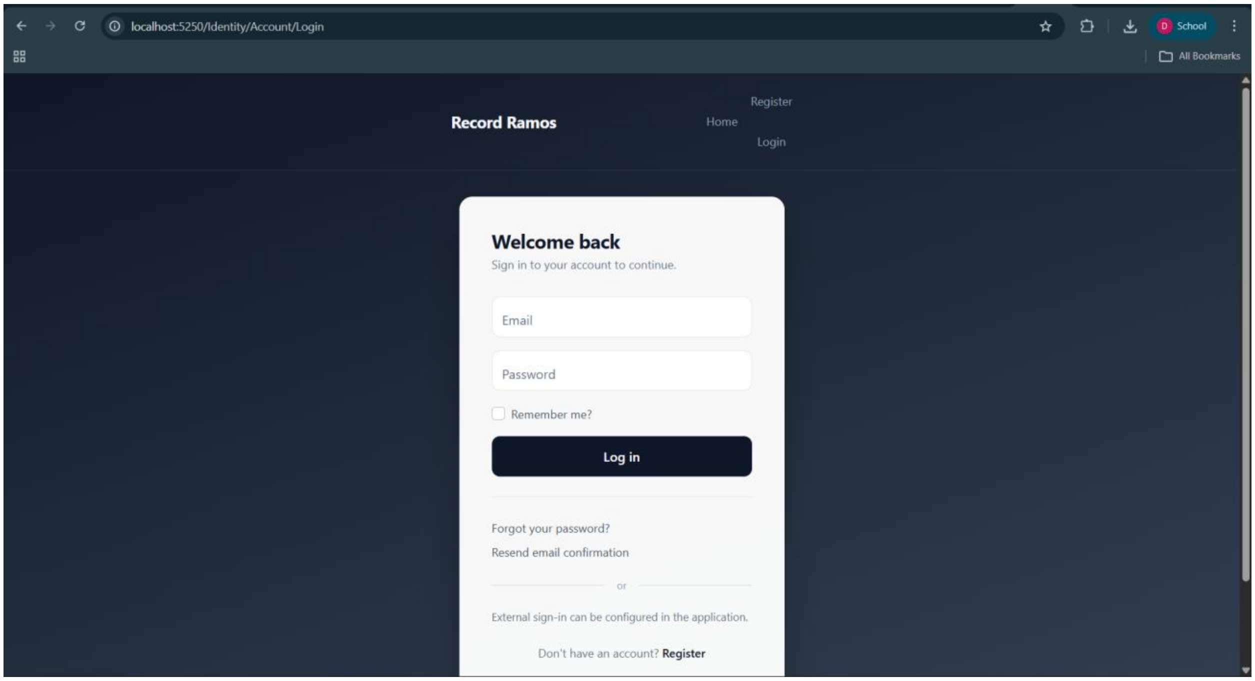


Create account

Enter your details to get started.

External sign-in can be configured in the application.

Already have an account? **Log in**



Welcome back

Sign in to your account to continue.

External sign-in can be configured in the application.

Don't have an account? **Register**

Part 2: Responsive Design and User Interface Improvements

3. **Enhance Forms with Bootstrap**:
   - Ensure that all forms in the Identity pages (e.g., Login, Register, Forgot Password) are styled using **Bootstrap** for responsiveness.
   - Add tooltips, validation messages, and icons to improve the user experience.

   Example for **Register.cshtml**:

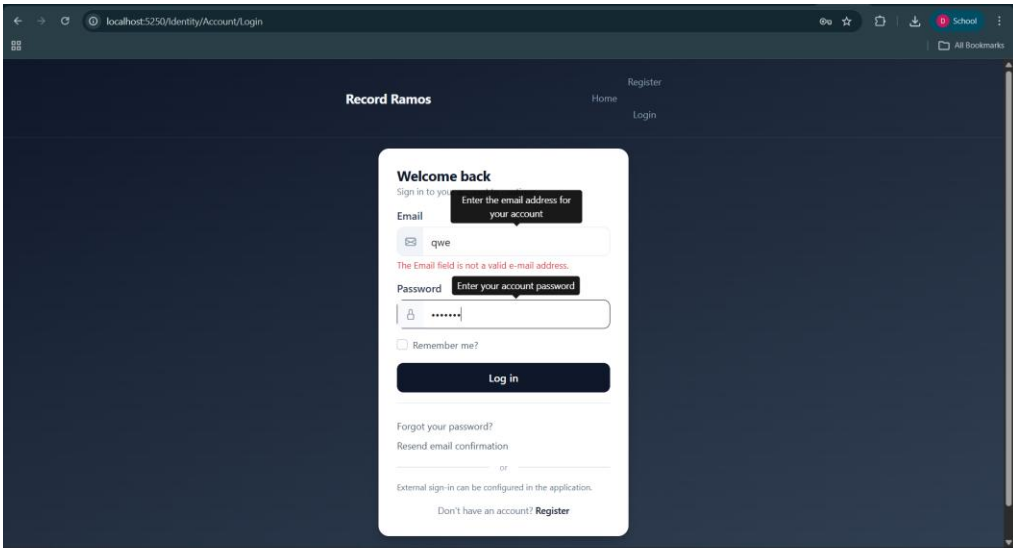
   html

```
<div class="row justify-content-center">
    <div class="col-md-6">
        <h2>Register</h2>
        <form asp-action="Register">
            <div class="form-group">
                <label asp-for="Input.Email"></label>
                <input asp-for="Input.Email" class="form-control" placeholder="Enter your email" />
                <span asp-validation-for="Input.Email" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Input.Password"></label>
                <input asp-for="Input.Password" class="form-control" placeholder="Enter your password" />
                <span asp-validation-for="Input.Password" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Input.ConfirmPassword"></label>
                <input asp-for="Input.ConfirmPassword" class="form-control" placeholder="Confirm your password" />
                <span asp-validation-for="Input.ConfirmPassword" class="text-danger"></span>
            </div>
            <button type="submit" class="btn btn-primary btn-block">Register</button>
        </form>
    </div>
</div>
```

   - Add **error messages**, **success notifications**, and **user feedback** elements (such as loading spinners during authentication).

Deliverable:

- Screenshot of at least one form (Login, Register, or Forgot Password) with improved form styling, tooltips, and validation messages.



Welcome back

Sign in to you

Enter the email address for your account

Email: qwe

The Email field is not a valid e-mail address.

Password: Enter your account password

□ Remember me?

Forgot your password?

Resend email confirmation

or

External sign-in can be configured in the application.

Don't have an account? Register

Part 3: Dynamic Content Based on User Authentication and Roles

4. **Display Dynamic Navigation Based on User Authentication**:
   - Modify the shared layout (`_Layout.cshtml`) or the custom Identity layout to display different navigation items based on whether the user is authenticated or not.
   - Use the following Razor syntax to conditionally display links:

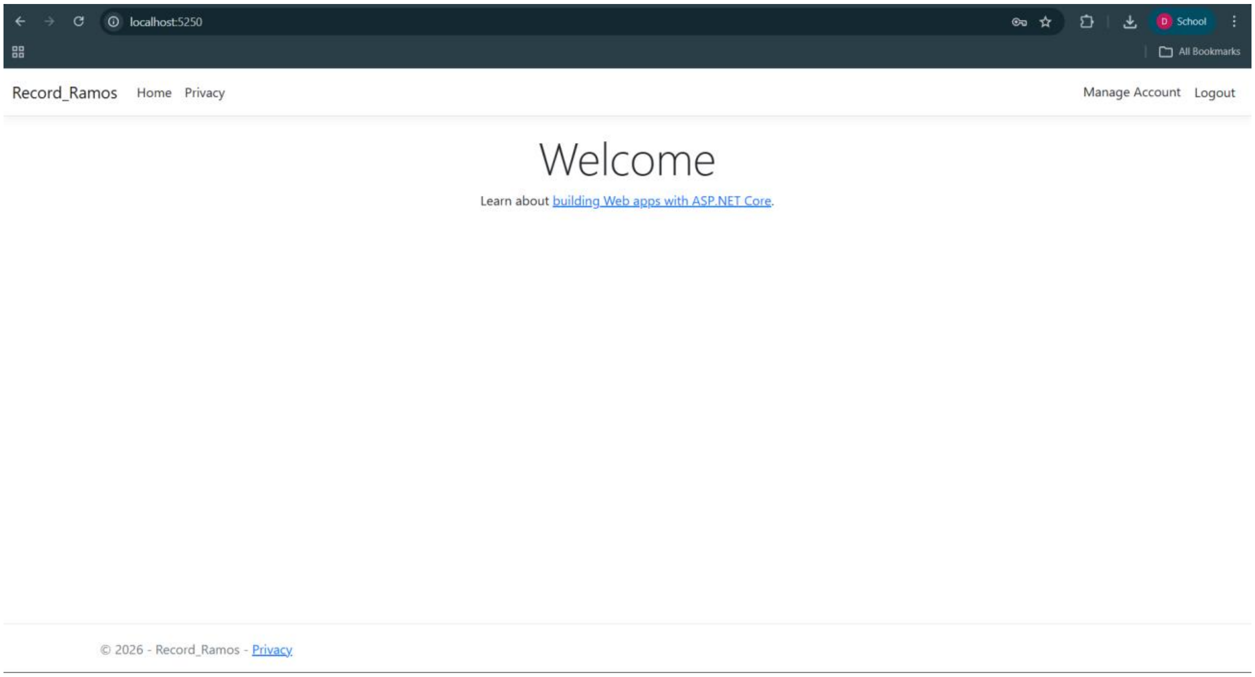
```
html

@if (User.Identity.IsAuthenticated)
{
    <ul class="navbar-nav ml-auto">
        <li class="nav-item">
            <a class="nav-link"
href="/Identity/Account/Manage">Manage Account</a>
        </li>
        <li class="nav-item">
            <a class="nav-link"
href="/Identity/Account/Logout">Logout</a>
        </li>
    </ul>
}
else
{
    <ul class="navbar-nav ml-auto">
        <li class="nav-item">
            <a class="nav-link"
href="/Identity/Account/Login">Login</a>
        </li>
        <li class="nav-item">
            <a class="nav-link"
href="/Identity/Account/Register">Register</a>
        </li>
    </ul>
}
```

Deliverable:

- ○ Screenshot showing different navigation menus for authenticated and unauthenticated users.

Authenticated User

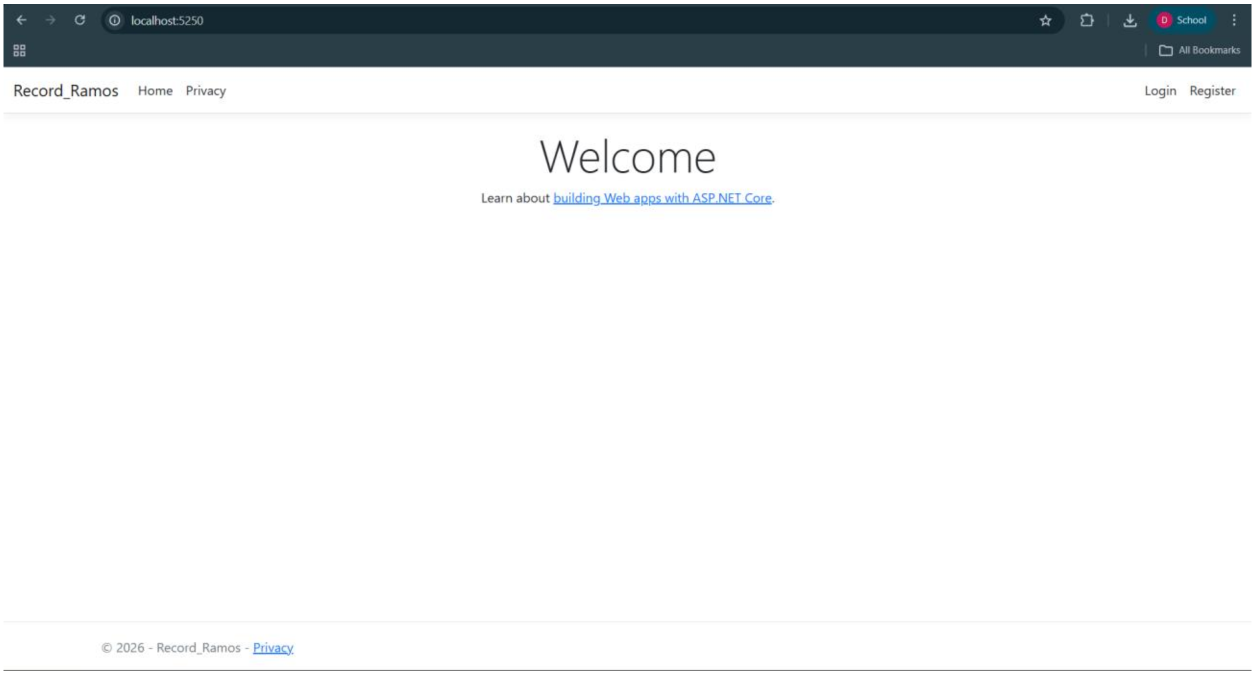


Welcome

Learn about building Web apps with ASP.NET Core.

© 2026 - Record_Ramos - Privacy

Unauthenticated User



Welcome

Learn about building Web apps with ASP.NET Core.

© 2026 - Record_Ramos - Privacy

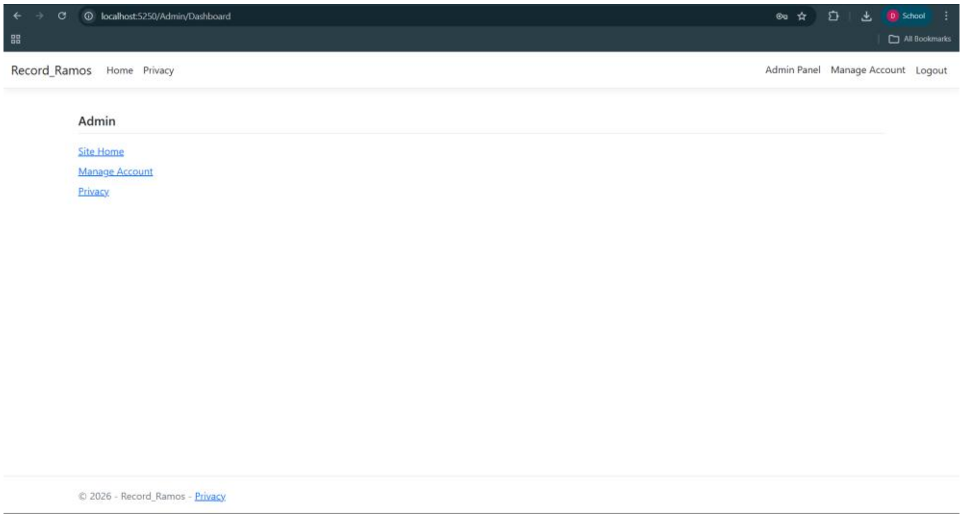
5. **Role-Based Layout Customization**:
   - Customize the layout further to display different navigation elements for users with specific roles (e.g., Admins vs. regular users).
   - For example, display an **Admin Panel** link for users with the "Admin" role:

```
html
@if (User.IsInRole("Admin"))
{
    <li class="nav-item">
        <a class="nav-link" href="/Admin/Dashboard">Admin
Panel</a>
    </li>
}
```

Deliverable:

- Screenshot showing different links for Admin users versus regular users.

Admin



localhost:5250/Admin/Dashboard

Record_Ramos Home Privacy Admin Panel Manage Account Logout

Admin

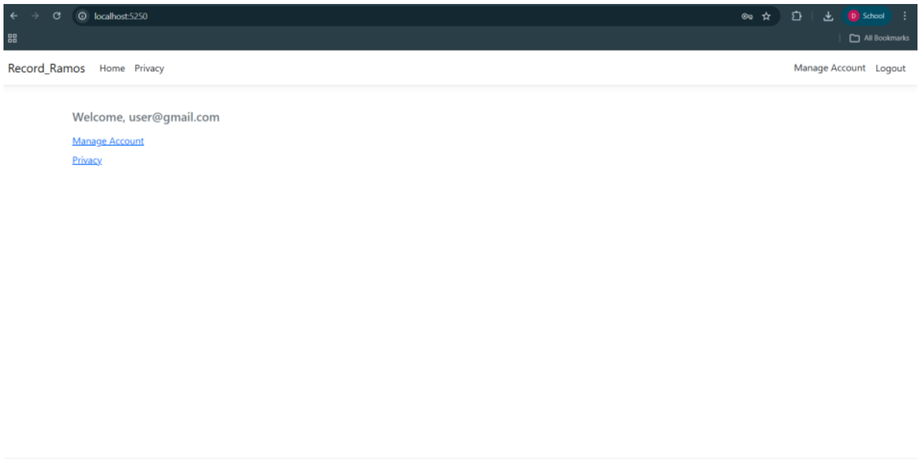
Site Home

Manage Account

Privacy

© 2026 - Record_Ramos - Privacy

Regular User



localhost:5250

Record_Ramos Home Privacy Manage Account Logout

Welcome, user@gmail.com

Manage Account

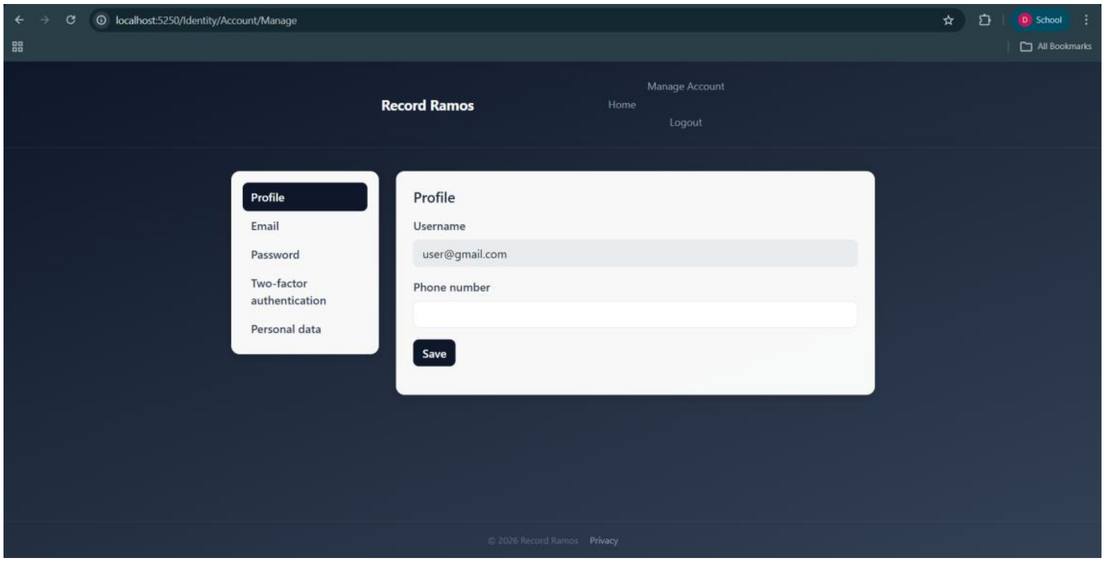
Privacy

Part 4: Profile Management Customization

6. **Customize Profile Management Pages**: IF NEEDED
   - Scaffold and customize the **Manage** section under Identity (Account Management).
   - Ensure the user can update their profile, change passwords, and manage account settings.
   - Add Bootstrap styling to the `Manage` pages and apply consistent UI elements across these pages.

Deliverable:

- Screenshot of the customized **Manage** profile page with a consistent design.

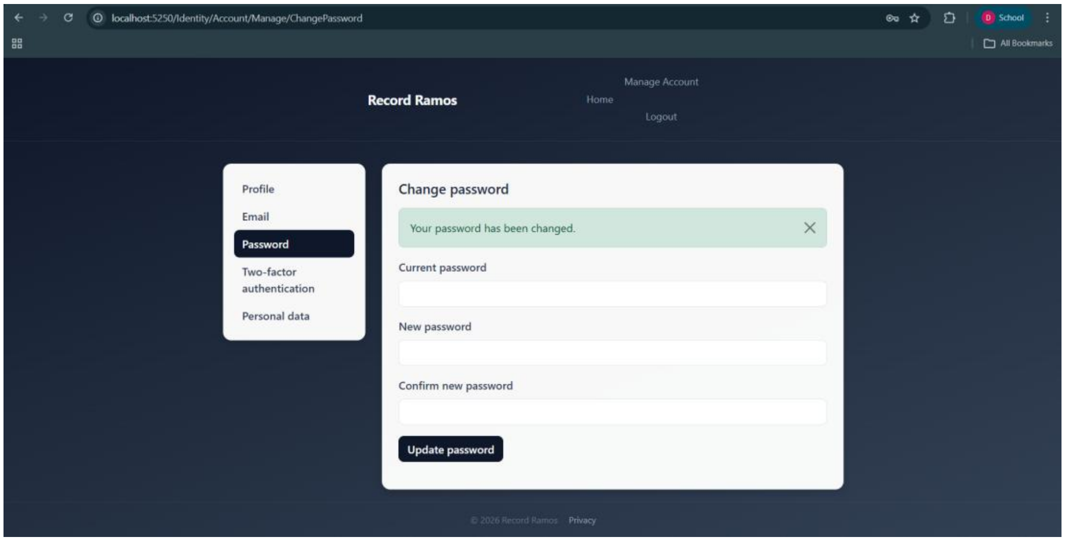


Profile

Username

user@gmail.com

Phone number



Change password

Your password has been changed.

Current password

New password

Confirm new password