ASP.NET Core project with EF Core (Code First), MSSQL, and MDI-style layout with login and CRUD for users and employees.

---

🔧 1. Create the Project

- Open Visual Studio → Create a new project

- Select ASP.NET Core Web App (Model-View-Controller) → Name it

- Select .NET 6 or .NET 7, enable Authentication: Individual Accounts (for login out-of-the-box)

---

🗃️ 2. Install EF Core Packages

Run this in Package Manager Console:

```bash

Install-Package Microsoft.EntityFrameworkCore.SqlServer

Install-Package Microsoft.EntityFrameworkCore.Tools

```

---

🧱 3. Create Models

User.cs

```csharp

public class User

{

public int Id { get; set; }

public string Username { get; set; }

public string Password { get; set; } // In real apps, hash this!

}

```

Employee.cs

```csharp

public class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public string Department { get; set; }

public string Email { get; set; }

}

```

---

🛠️ 4. Create DbContext

AppDbContext.cs

```csharp

public class AppDbContext : DbContext

{

public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

public DbSet<User> Users { get; set; }

public DbSet<Employee> Employees { get; set; }

}

```

---

⚙️ 5. Configure Connection String

In `appsettings.json`:

```json

"ConnectionStrings": {

"DefaultConnection": "Server=.;Database=MyAppDb;Trusted\_Connection=True;"

}

```

In `Program.cs`:

```csharp

builder.Services.AddDbContext<AppDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

```

---

🔄 6. Run Migrations

```bash

Add-Migration InitialCreate

Update-Database

```

---

🔐 7. Login Page (Username & Password)

- Create a `LoginController` and `LoginViewModel`.

LoginViewModel.cs

```csharp

public class LoginViewModel

{

public string Username { get; set; }

public string Password { get; set; }

}

```

LoginController.cs

```csharp

public class LoginController : Controller

{

private readonly AppDbContext \_context;

public LoginController(AppDbContext context)

{

\_context = context;

}

[HttpGet]

public IActionResult Index() => View();

[HttpPost]

public IActionResult Index(LoginViewModel model)

{

var user = \_context.Users.SingleOrDefault(u => u.Username == model.Username && u.Password == model.Password);

if (user != null)

{

return RedirectToAction("Dashboard", "Home");

}

ModelState.AddModelError("", "Invalid username or password");

return View(model);

}

}

```

Login View (`Index.cshtml`)

```html

<form asp-action="Index" method="post">

<input asp-for="Username" placeholder="Username" />

<input asp-for="Password" type="password" placeholder="Password" />

<button type="submit">Login</button>

</form>

```

---

🧭 8. HomeController: Dashboard & MDI Layout

HomeController.cs

```csharp

public class HomeController : Controller

{

public IActionResult Dashboard() => View();

}

```

Dashboard View (`Dashboard.cshtml`)

```html

<h2>Dashboard</h2>

<ul>

<li><a asp-controller="User" asp-action="Add">Add User</a></li>

<li><a asp-controller="Employee" asp-action="Add">Add Employee</a></li>

<li><a asp-controller="Employee" asp-action="List">View Employees</a></li>

</ul>

```

---

👥 9. User Management

- Controller: `UserController`

- Views: `Add`, `List`

UserController.cs

```csharp

public class UserController : Controller

{

private readonly AppDbContext \_context;

public UserController(AppDbContext context) => \_context = context;

public IActionResult Add() => View();

[HttpPost]

public IActionResult Add(User user)

{

\_context.Users.Add(user);

\_context.SaveChanges();

return RedirectToAction("List");

}

public IActionResult List() => View(\_context.Users.ToList());

}

```

---

👔 10. Employee Management (Add, Edit, Delete, List)

EmployeeController.cs

```csharp

public class EmployeeController : Controller

{

private readonly AppDbContext \_context;

public EmployeeController(AppDbContext context) => \_context = context;

public IActionResult Add() => View();

[HttpPost]

public IActionResult Add(Employee emp)

{

\_context.Employees.Add(emp);

\_context.SaveChanges();

return RedirectToAction("List");

}

public IActionResult List() => View(\_context.Employees.ToList());

public IActionResult Edit(int id) => View(\_context.Employees.Find(id));

[HttpPost]

public IActionResult Edit(Employee emp)

{

\_context.Employees.Update(emp);

\_context.SaveChanges();

return RedirectToAction("List");

}

public IActionResult Delete(int id)

{

var emp = \_context.Employees.Find(id);

if (emp != null)

{

\_context.Employees.Remove(emp);

\_context.SaveChanges();

}

return RedirectToAction("List");

}

}

```

---

✅ 11. MDI-Like Navigation

- The Dashboard view acts as the MDI main screen with links.

- Each menu redirects to Add or View pages in new screens.

---