**DOCUMENT 7**
**INSTALL IDENTITY AND ADD APPLICATIONUSER**

## Goal

Add ASP.NET Core Identity so your app supports user accounts, login, logout, and roles.
Then extend IdentityUser with extra fields like FirstName, LastName, Address.

## What Identity gives you

Tables for users, roles, claims, logins, tokens
Secure password hashing
Built in login and registration pages if you scaffold
UserManager and SignInManager services

## Step 1. Install Identity packages

In NuGet, install

- Microsoft.AspNetCore.Identity.EntityFrameworkCore

## Step 2. Create ApplicationUser

Inside the folder named Models Create ApplicationUser.cs right click on folder then click on add and then class the name the class ApplicationUser.cs

```
using Microsoft.AspNetCore.Identity;
using System.ComponentModel.DataAnnotations;

namespace RTWebApplication.Models
{
    public class ApplicationUser : IdentityUser
    {
        [Required]
        [MaxLength(50)]
        public string FirstName { get; set; } = "";

        [Required]
        [MaxLength(50)]
        public string LastName { get; set; } = "";

        [Required]
        [MaxLength(200)]
        public string Address { get; set; } = "";

        public DateTime CreatedAt { get; set; } = DateTime.UtcNow;
    }
}

```

**Explanation**

IdentityUser already contains Email, UserName, PasswordHash, PhoneNumber.

You extend it to store extra profile fields.
Required and MaxLength support validation and sensible database types.

### Step 3. Change ApplicationDbContext to IdentityDbContext

Open ApplicationDbContext.cs and change it.

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using RTWebApplication.Models;

namespace RTWebApplication.Data
{
    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext>
options)
            : base(options)
        {
        }

        public DbSet<Product> Products { get; set; }
    }
}

```

**Explanation**

IdentityDbContext adds Identity tables into your database schema. <ApplicationUser> tells Identity which user type to store.

### Step 4. Register Identity in Program.cs

Open Program.cs and add the needed usings.

```
using Microsoft.AspNetCore.Identity;
using RTWebApplication.Data;
using RTWebApplication.Models;

Then add Identity services after AddDbContext.

builder.Services
    .AddIdentity<ApplicationUser, IdentityRole>(options =>
    {
        options.Password.RequiredLength = 8;
        options.Password.RequireNonAlphanumeric = false;
        options.Password.RequireUppercase = true;
        options.User.RequireUniqueEmail = true;
    })
    .AddEntityFrameworkStores<ApplicationDbContext>()
```

**Explanation**

AddIdentity registers UserManager, SignInManager, RoleManager.
AddEntityFrameworkStores tells Identity to store data in your DbContext.

**Step 5. Create migration and update database**

```
1. Add-Migration AddIdentityAndApplicationUser
2. Update-Database
3.
```

What you should see in the database

AspNetUsers
AspNetRoles
AspNetUserRoles
AspNetUserClaims
AspNetRoleClaims
AspNetUserLogins
AspNetUserTokens

**If your new columns do not appear in AspNetUsers**

**Cause**

Migration not generated after editing ApplicationUser

**Fix**

Create a new migration and update database

```
1. Add-Migration AddCustomFieldsToApplicationUser
2. Update-Database
3.
```

**If it still does not appear**

**Cause**

You edited ApplicationUser but DbContext still uses IdentityDbContext without ApplicationUser generic

**Fix**

## Ensure ApplicationDbContext inherits from IdentityDbContext<ApplicationUser>