1. **Add Connection String**

**"ConnectionStrings": {**

**"NZWalksConnectionString": "Server=DESKTOP-PUB10M6; Database= NZWalksDb; Trusted\_Connection=True;TrustServerCertificate=True;",**

**"NZWalksAuthConnectionString": "Server=DESKTOP-PUB10M6; Database= NZWalksAuthDb; Trusted\_Connection=True;TrustServerCertificate=True;"**

**},**

1. **Add DbContext**

Create a DbContext file named **NZWalksAuthDbContext.cs** inside the Data folder.

**AppDbContext – (already created)**

using Microsoft.EntityFrameworkCore;

public class NZWalksDbContext: DbContext

{

public NZWalksDbContext(DbContextOptions<NZWalksDbContext> options) : base(options)

{

}

public DbSet<Walk> Walks { get; set; }

public DbSet<Region> Regions { get; set; }

public DbSet<Difficulty> Difficulties { get; set; }

}

**AppAuthDbContext**

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

public class NZWalksAuthDbContext : IdentityDbContext<IdentityUser>

public class NZWalksAuthDbContext: IdentityDbContext //if identityUser class is default

{

public NZWalksAuthDbContext(DbContextOptions<NZWalksAuthDbContext> options) : base(options)

{

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

//seed data for roles like roles

}

}

1. **Register the connection string and AppAuthDbContext in the program.cs file**

// Register the authentication database context with dependency injection

var authConnectionString = builder.Configuration.GetConnectionString("NZWalksAuthConnectionString");

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

options.UseSqlServer(authConnectionString));

1. **Seeding roles data**

public class NZWalksAuthDbContext: IdentityDbContext

{

public NZWalksAuthDbContext(DbContextOptions<NZWalksAuthDbContext> options) : base(options)

{

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

//seed data for roles

var readerRoleId = "3d878464-81c8-481f-92cc-48d3efc8ee0b".ToString();

var writerRoleId = "c5715846-7a07-4fba-bc3e-bfd9dcdd67a9".ToString();

var roles = new List<IdentityRole>()

{

new IdentityRole() { Id= readerRoleId, ConcurrencyStamp=readerRoleId, Name = "Reader", NormalizedName="Reader.ToUpper()" },

new IdentityRole() { Id = writerRoleId, ConcurrencyStamp=writerRoleId, Name="Writer", NormalizedName="Writer.ToUpper()" }

};

modelBuilder.Entity<IdentityRole>().HasData(roles);

}

}

1. **Migration and update the database**

add-migration addingIdentity or add-migration "creatingAuthDB"

Note: if the migration message has a space, then we can use “” double quotes

If we have multiple DbContext classes, then

add-migration MigrationName -context YourDbContextName

add-migration AddIdentitySchema -context NZWalksAuthDbContext

Har DbContext ka apna alag Migrations folder banana achha hota hai for clarity

add-migration AddIdentitySchema -context NZWalksAuthDbContext -output-dir Migrations/Auth

Working in visual Studio

PM> add-migration createAuthDb -Context NZWalksAuthDbContext -outputDir Migrations/Auth

**🔁 PMC vs .NET CLI Comparison:**

| **Console** | **Context Parameter** | **OutputDir Parameter** |
| --- | --- | --- |
| **PMC (Visual Studio)** | -Context | -OutputDir |
| **.NET CLI (terminal)** | --context | --output-dir |

Update database

**PMC (Package Manager Console) me syntax:**

**Update-Database -Context NZWalksAuthDbContext**