NOTE: ANYWHERE YOU SEE RTWebApplication THAT IS THE NAME WE GAVE THE PROJECT AT THE POINT OF CREATING IT SO IF YOU NAMED YOURS DIFFERENT JUST REPLACE IT WITH THAT NAME YOU GAVE IT

01 Setup MVC Environment

Goal
Set up an ASP.NET Core MVC project with Entity Framework Core and connect it to SQL Server.

Key idea
Entity Framework Core maps C# classes to database tables.
ApplicationDbContext is the bridge between your code and the database.

Step 1. Install Entity Framework Core packages

In Visual Studio
Right click the project
Select Manage NuGet Packages
Install the following packages

- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.SqlServer
- Microsoft.EntityFrameworkCore.Tools

Rule
The package version must match your .NET version.

Step 2. Add a connection string in appsettings.json

What this does
This tells your application where the database is and how to connect.

Open appsettings.json and add

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost;Database=[Database Name];Trusted_Connection=true;TrustServerCertificate=true;MultipleActiveResultSets=true;"
  }
}

```

Explanation

DefaultConnection is the name used in Program.cs.
LocalDB is commonly used for college projects.
Database is created automatically when migrations run.

Step 3. Create ApplicationDbContext

Create a folder named Data.
Inside it, create ApplicationDbContext.cs

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

namespace RTWebApplication.DbServices
{
    public class ApplicationDbContext : DbContext
    {
        public ApplicationDbContext(DbContextOptions options) : base(options)
        {

        }

        // Tables will be added here later
    }
}
```

Explanation

DbContext is EF Core's base class.

DbContextOptions carries database configuration.

The constructor passes configuration into the base class

Step 4. Register DbContext in Program.cs

At the top of Program.cs add

```
1. using Microsoft.EntityFrameworkCore;
2. using RTWebApplication.Data;
```

Before `var app = builder.Build();` **add**

```
1. var connectionString =
builder.Configuration.GetConnectionString("DefaultConnection");
2.
3. builder.Services.AddDbContext<ApplicationDbContext>(options =>
4. {
5.     options.UseSqlServer(connectionString);
6. });
```

Explanation

GetConnectionString reads from appsettings.json.

AddDbContext enables dependency injection.

UseSqlServer sets SQL Server as the database provider

Step 5. Create the database using migrations

Open

Tools

NuGet Package Manager

Package Manager Console

Run

```
1. Add-Migration InitialCreate
2. Update-Database
```

Explanation
Add-Migration creates a migration file.
Update-Database applies it to SQL Server.

Common errors

No DbContext found
Check that ApplicationDbContext is public and registered.

Database not created
Ensure SQL Server Express localhost is installed or connected in the SSMS.

End result checklist

- Connection string exists
- ApplicationDbContext created
- DbContext registered
- Database appears in SQL Server Object Explorer