# Entity Framework Core – ASP.NET Essentials

Problems for exercises for the ["Entity Framework Core" course @ SoftUni](https://softuni.bg/trainings/4842/entity-framework-core-february-2025)

A movie ticket and popcorn

Description automatically generated

## Introduction to CinemaWebApp and Entity Framework Core

**Welcome to the** Entity Framework Core Workshop**! In this workshop, we will build the** CinemaWebApp **step by step, focusing on** backend development **using** ASP.NET Core **and** Entity Framework Core**. Throughout this journey, you will learn how to design and manage a database, work with models, and implement CRUD operations.**

**Frontend development will be covered in a future ASP.NET course, so for now, we will focus** exclusively on the database and backend logic

### Start Thinking About Your Graduation Project

As we progress with building the **CinemaApp**, it’s important to highlight that **this project closely resembles what you will need to create and defend for your graduation**:

* The concepts and architecture we use here can be **adapted to your own graduation project**
* Think about how you will design **your own database and entities** for your final project

### Start Exploring ASP.NET Architecture

You Will Need This for Your Graduation Project

* Just like with the **database design**, your graduation project will require you to structure your **backend, business logic, and frontend properly**
* The **sooner you start exploring ASP.NET architecture**, the **easier it will be** when you need to build your own project

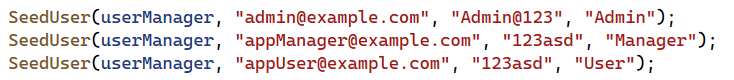
### Workshop Goals

* **Create an empty ASP.NET Core MVC project** and set up EF Core
* **Define and configure database models** using EF Core
* **Apply migrations to generate the database schema**
* **Seed initial data** into the database
* **Implement basic CRUD operations** using EF Core

## Demonstration of the Application

Before starting the development, let's explore the final outcome of the **CinemaWebApp**. This will help you understand the goal and structure of the application you will be building:

* These are the credentials that could be used initially for exploring the application



A screenshot of a movie theater

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a movie ticket

AI-generated content may be incorrect.

A black rectangle with white text

AI-generated content may be incorrect.

A black rectangular object with a white rectangle

AI-generated content may be incorrect.

A screen shot of a movie

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a movie

AI-generated content may be incorrect.

A screenshot of a computer dashboard

AI-generated content may be incorrect.

Screens screenshot of a movie theater

AI-generated content may be incorrect.

A screenshot of a program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

### Database Overview



## Creating an ASP.NET Core MVC Project

### Setting Up the Project

**Create a New ASP.NET Core MVC Project**:

* **Open Visual Studio and select Create a new project**
* **Choose ASP.NET Core Web App (Model-View-Controller) and click Next**
* **Name the project (CinemaApp) and choose a location to save it**
* **Select .NET 8 as the framework and click Create**
* **Run the application to ensure it is working properly (Ctrl + F5)**

**A black text on a white background

AI-generated content may be incorrect.A blue background with black text

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a web application

AI-generated content may be incorrect.**

**A screenshot of a computer application

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

## Setting Up the Project Architecture

To keep the project **well-organized**, we will create the **Data** folder, which will contain all database-related components

### Create the Data Folder

* In **Solution Explorer**, right-click the project and select **Add > New Folder**.
* Name the folder **Data** (for organizational purposes)
* Creating Class Library Projects for Separation of Concerns
  + **CinemaApp.Data.Models** – Stores all database models
  + **CinemaApp.Data** – Handles database access (DbContext, repositories, configurations)
* Adding References
  + **Right-click CinemaApp.Data** → Select **Add Project Reference**
  + Select **CinemaApp.Data.Models** and click **OK**
* Ensuring the package dependencies are correctly installed
* A screenshot of a computer

  AI-generated content may be incorrect.
* A screenshot of a computer

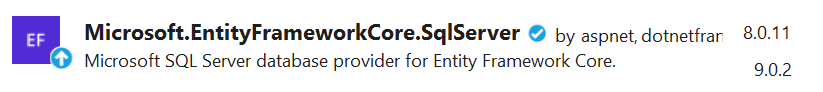
  AI-generated content may be incorrect.A screenshot of a computer

  AI-generated content may be incorrect.
* A blue and white rectangle

  AI-generated content may be incorrect.

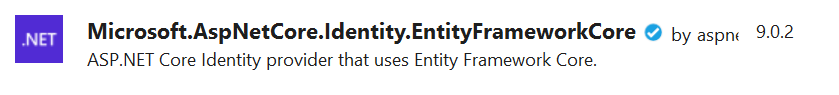
## Install EF Core Packages in CinemaApp.Data

Since we are working with **Entity Framework Core (EF Core)** for database management, we need to install the required packages:



A close up of a logo

AI-generated content may be incorrect.



A black text on a white background

AI-generated content may be incorrect.

Once all packages are installed, we can move forward with creating **all data models**

## Creating Data Models

We will now create the **core entity models** inside **CinemaApp.Data.Models** before implementing the database context

### ApplicationUser

Extends IdentityUser<Guid>

* **This ensures that ASP.NET Core Identity handles user authentication**
* **Uses GUID instead of an integer (int) for user IDs**

**Constructor**: Generates a **New GUID for Id**

A white background with blue and black text

AI-generated content may be incorrect.

**Navigation Properties**

* **ApplicationUserMovies - Represents the many-to-many relationship between users and movies**
* Tickets - a collection of type Ticket

### Cinema

* Id– **Guid**, **Primary Key**
* **Name – string (required)**
* Location– string **(required)**
* IsDeleted– **bool**
* CinemaMovies - a collection of type CinemaMovie
* Tickets **-** acollection of type Ticket

### Movie

* **Id – Guid, Primary Key**
* **Title – string (required)**
* **Genre – string (required)**
* **ReleaseDate – DateTime (required)**
* **Director – string (required)**
* **Duration – int (required, represents movie length in minutes)**
* **Description – string (required)**
* **ImageUrl – string?**
* **IsDeleted – bool**
* **MovieCinemas –** a collection of type **CinemaMovie**
* **MovieApplicationUsers** – a collection of type ApplicationUserMovie
* **Tickets** – a collection of typeTicket **(one-to-many relationship)**

### CinemaMovie

* **MovieId** – **Guid, Foreign Key** (References Movie)
* **Movie** – **Movie** (Navigation Property)
* **CinemaId** – **Guid, Foreign Key** (References Cinema)
* **Cinema** – **Cinema** (Navigation Property)
* **AvailableTickets** – **int**, Tracks remaining tickets for the movie in a specific cinema
* **IsDeleted** – **bool**
* **Showtimes** – **string** **(formatted as "00000", stored as varchar(5))**

### ApplicationUserMovie

* **ApplicationUserId** – **Guid, Foreign Key** (References ApplicationUser)
* **ApplicationUser** – a navigation property of type ApplicationUser
* **MovieId** – **Guid, Foreign Key** (References Movie)
* **Movie** – a navigation property of type Movie

### Ticket

* **Id – Guid, Primary Key**
* **Price – decimal (required)**
* **CinemaId** – **Guid, Foreign Key** (References Cinema)
* **Cinema** – **Cinema** (Navigation Property)
* **MovieId** – **Guid, Foreign Key** (References Movie)
* **Movie** – a navigation property of type Movie
* **ApplicationUserId** – **Guid, Foreign Key** (References ApplicationUser)
* **ApplicationUser** – a navigation property of type ApplicationUser

### Creating a ValidationConstants Class for Centralized Validations

A **separate class** to store validation rules could be created (option). This will help **keep the code clean, maintainable, and reusable:**

* **Avoid Hardcoded Values** – Keeps validation rules in a single place instead of spreading them across models
* **Improves Readability** – Models are cleaner and easier to understand
* **Easy Updates** – If a validation rule changes, update it in one place instead of multiple files

## Creating CinemaAppDbContext

Now that we have created all the **models**, we need to set up the **database context**, which will serve as the main bridge between our application and the database. The **database context** is responsible for managing the entities and database operations using **Entity Framework Core (EF Core)**.

**Important Note for Students: Remove the Legacy Data Folder**

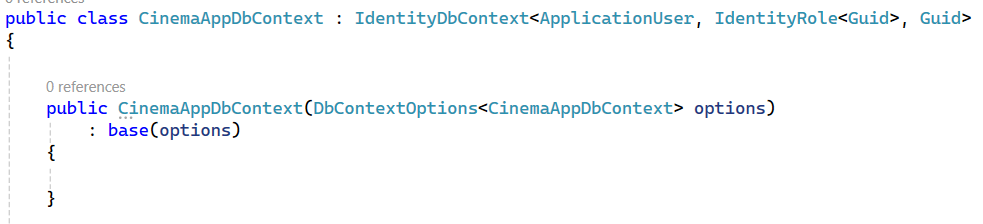
When setting up the **Entity Framework Core (EF Core) project**, **students must delete the legacy Data folder** that is automatically created when an ASP.NET project includes Identity by default.

**Why is This Important?**

**Prevents Duplicate DbContext Issues**

* The default Identity setup **creates an extra ApplicationDbContext** inside the Data folder.
* If not removed, EF Core **detects multiple DbContext classes**, leading to confusion when applying migrations.

### Create the CinemaAppDbContext Class

* **Inside CinemaApp.Data**, create a new file called CinemaAppDbContext.cs
* Extending IdentityDbContext for Authentication  
  
  + Extends **IdentityDbContext** to include **authentication support**
  + Uses **GUID as the primary key** for users and roles
  + Receives **database connection settings** via **dependency injection**
  + Passes options to the **base constructor (IdentityDbContext)**

### Defining Database Tables

A screen shot of a computer program

AI-generated content may be incorrect.

## Configuring Entities Using Fluent API

In Entity Framework Core, we can configure entity relationships and constraints using **Data Annotations** or **Fluent API**. While **Data Annotations** work well for simple cases, using **Fluent API in separate configuration classes** provides **better control, flexibility, and maintainability.**

### Using Configuration Classes

* Inside CinemaApp.Data, create a **new folder** named **Configuration**
* Create a Configuration Class for Each Model

A screenshot of a computer

AI-generated content may be incorrect.

## Understanding OnModelCreating in CinemaAppDbContext

The **OnModelCreating method** in **Entity Framework Core (EF Core)** is responsible for configuring the **database schema, relationships, and entity constraints** before the database is created.

### Why Do We Need OnModelCreating?

**Defines Entity Relationships** – Ensures **correct table relationships (One-to-Many, Many-to-Many)**.  
✅**Applies Fluent API Configurations** – Loads **entity configurations from the Configuration folder**.  
✅ **Applies Identity Settings** – Calls base.OnModelCreating(modelBuilder) to apply **ASP.NET Core Identity configurations**.  
✅ **Allows Data Seeding** – Adds **default data** to tables during database creation.

A close-up of a computer code

AI-generated content may be incorrect.

base.OnModelCreating(modelBuilder);

* Calls the **base class implementation** (from IdentityDbContext)
* This ensures that **Identity tables** (AspNetUsers, AspNetRoles, etc.) are created properly

modelBuilder.ApplyConfigurationsFromAssembly(Assembly.GetExecutingAssembly());

* **Automatically applies all entity configurations** from the Configuration folder
* Instead of defining relationships in DbContext, we keep them **separate for better maintainability**

## Updated Program.cs for This Stage

* **Database Connection Configuration**  
  A computer screen shot of text

  AI-generated content may be incorrect.
* **Identity Setup**A computer screen shot of text

  AI-generated content may be incorrect.
* **Middleware Setup  
  A close up of words

  AI-generated content may be incorrect.**
* **Routing Setup  
  A computer code with text

  AI-generated content may be incorrect.**

### Configure **appsettings.json** to include the database connection string

## Add-Migration && Update-Database

## Verify the Database in SQL Server Management Studio (SSMS)

A screenshot of a computer

AI-generated content may be incorrect.