Create a simple .NET 8 web application

Sunday, March 2, 2025 12:58 PM

Steps to Implement

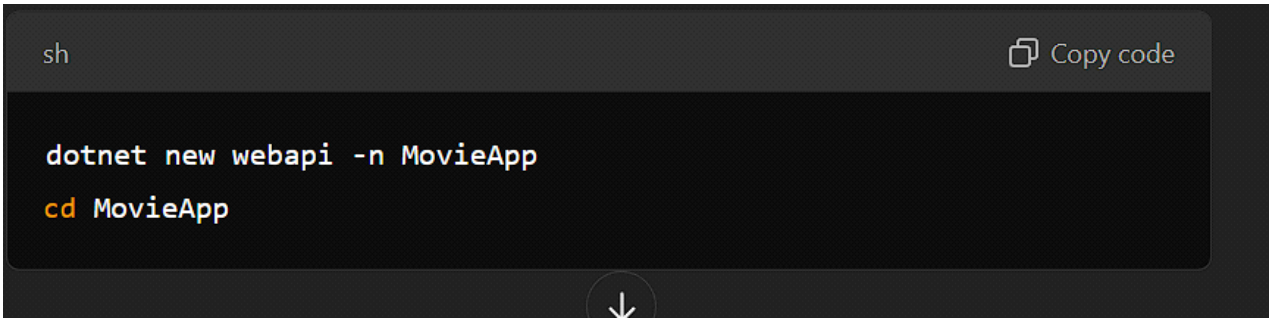
Create a new .NET Web API project

Steps to implement

1. Create a new .NET Web API project
2. Set up dependencies (EF Core,JWT,MySQL)
3. Configure database and authentication
4. Create User Authentication(Login/Register with JWT)
5. Implement CRUD operation for a Movie entity
6. Secure API endpoints with JWT authentication

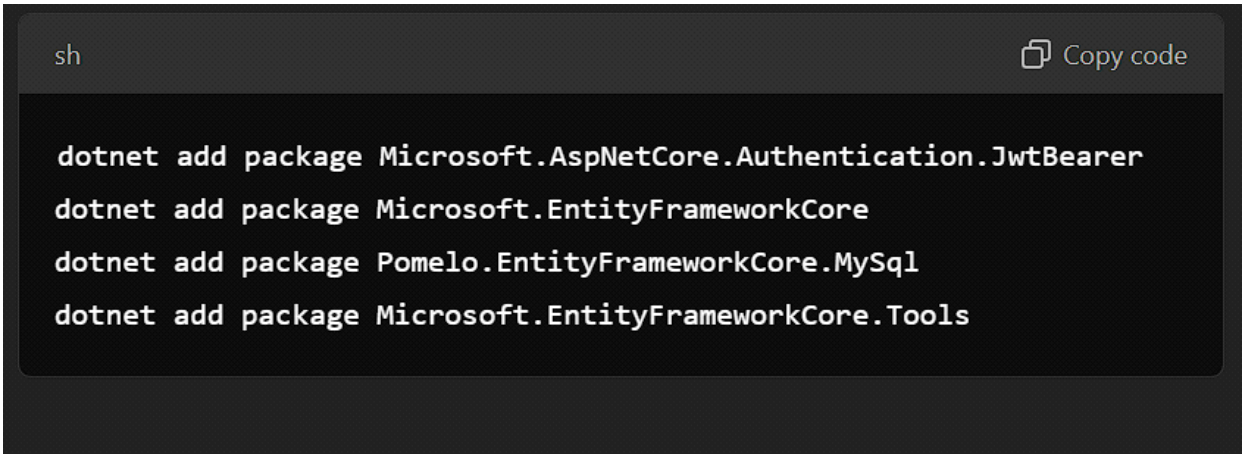
Step 1 : Create a New .NET web API Project

Run the following in VS Code terminal to create a new project



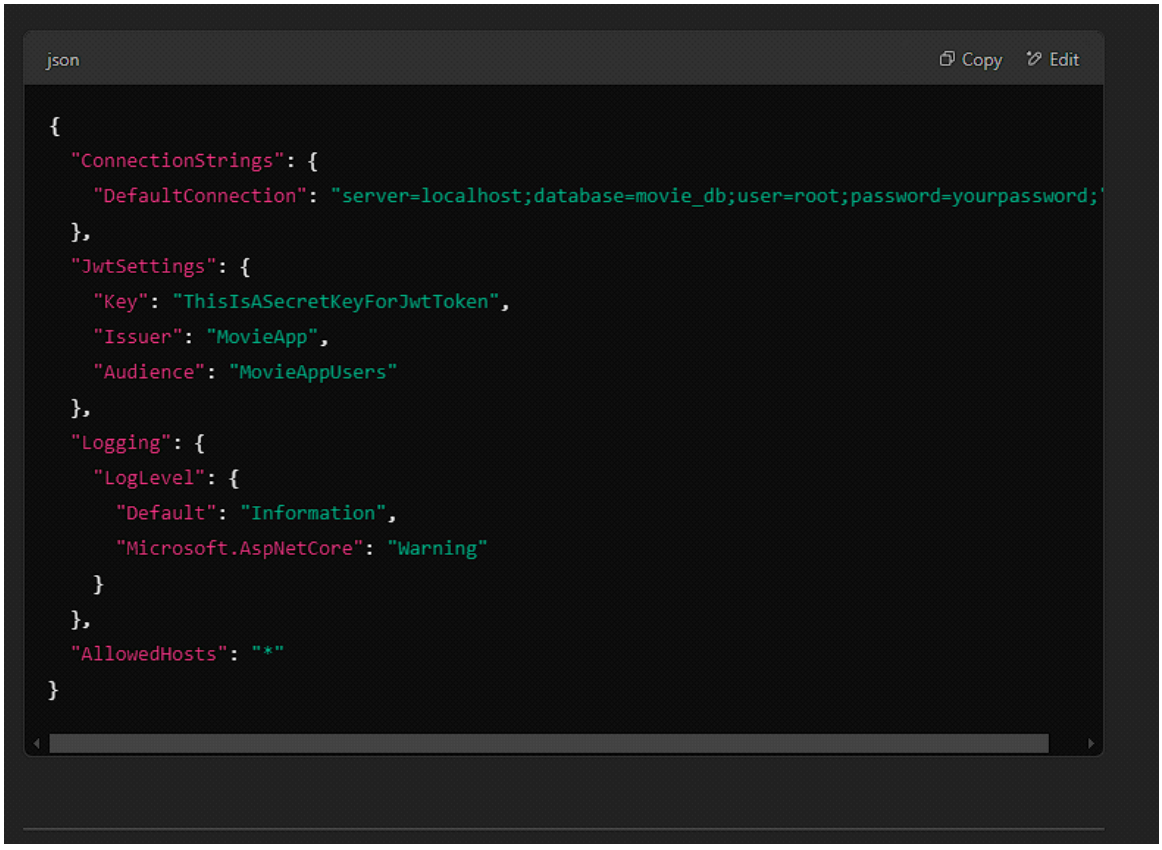
Step 2 : Install Required packages

Run the following command to install Entity Framework Core ,JWT authentication , and MySQL provider



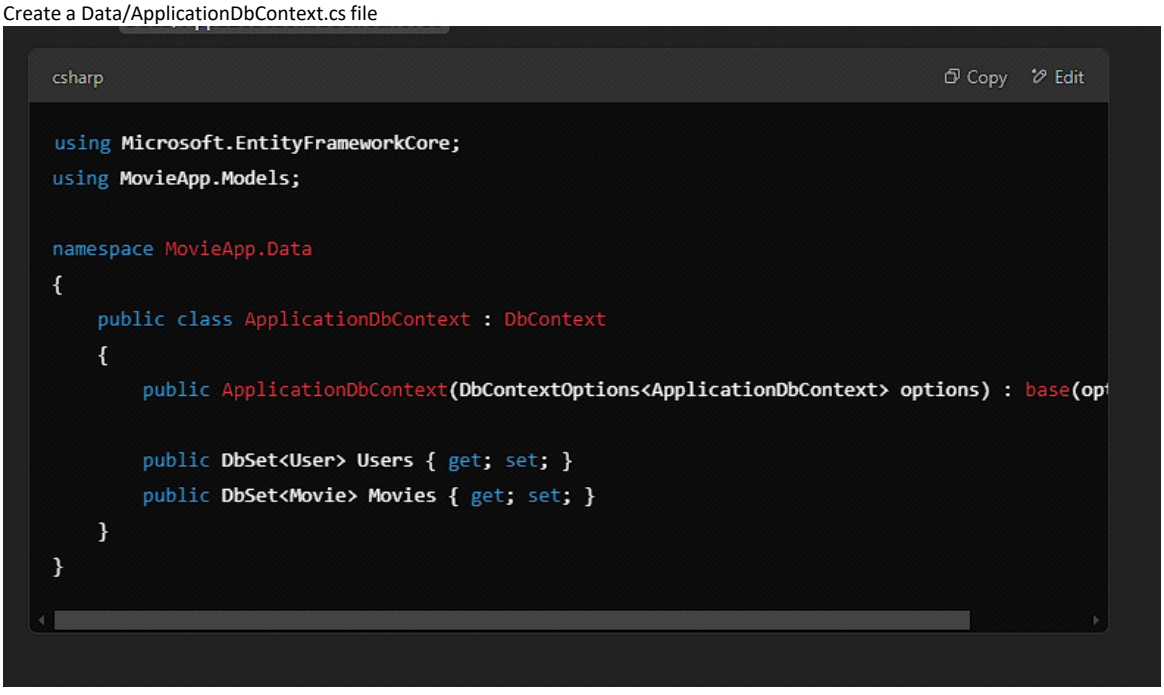
Step 3 : Configure appsettings.json

Edit appsettings.json to add the database connection string and JWT settings



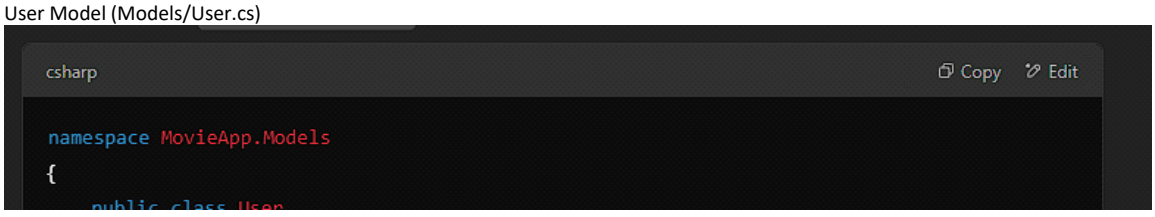
Step 4 : Set up Database Context

Create a Data/ApplicationDbContext.cs file



Step 5 : Create User & Movie Models

User Model (Models/User.cs)



```csharp
namespace MovieApp.Models
{
    public class User
    {
        public int Id { get; set; }
        public string Username { get; set; }
        public string Email { get; set; }
        public string PasswordHash { get; set; }
    }
}
```

Movie Model (`Models/Movie.cs`)

```csharp
namespace MovieApp.Models
{
    public class Movie
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public string Genre { get; set; }
        public string Description { get; set; }
    }
}
```

Step 6 : Implement Authentication Service

Create **Services/AuthService.cs**:

```csharp
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Text;
using Microsoft.IdentityModel.Tokens;
using MovieApp.Models;

namespace MovieApp.Services
{
    public class AuthService
    {
        private readonly IConfiguration _config;

        public AuthService(IConfiguration config)
        {
            _config = config;
        }

        public string GenerateJwtToken(User user)
        {
            var key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_config["JwtSettings:Ke
            var creds = new SigningCredentials(key, SecurityAlgorithms.HmacSha256);

            var claims = new[]
            {
                new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),
                new Claim(ClaimTypes.Name, user.Username)
            };

            var token = new JwtSecurityToken(
                issuer: _config["JwtSettings:Issuer"],
                audience: _config["JwtSettings:Audience"],
                claims: claims,
                expires: DateTime.UtcNow.AddHours(2),
                signingCredentials: creds
            );

            return new JwtSecurityTokenHandler().WriteToken(token);
        }
    }
}
```

◇ Step 7: Create Authentication Controller

Create **Controllers/AuthController.cs**:

Create `Controllers/AuthController.cs`:

```csharp
using Microsoft.AspNetCore.Mvc;
using MovieApp.Data;
using MovieApp.Models;
using MovieApp.Services;
using System.Security.Cryptography;
using System.Text;

namespace MovieApp.Controllers
{
    [Route("api/auth")]
    [ApiController]
    public class AuthController : ControllerBase
    {
```

```
public class AuthController : ControllerBase
{
    private readonly ApplicationDbContext _context;
    private readonly AuthService _authService;

    public AuthController(ApplicationDbContext context, AuthService authService)
    {
        _context = context;
        _authService = authService;
    }
```


```
    [HttpPost("register")]
    public IActionResult Register(User user)
    {
        user.PasswordHash = HashPassword(user.PasswordHash);
        _context.Users.Add(user);
        _context.SaveChanges();
        return Ok(new { message = "User registered successfully" });
    }

    [HttpPost("login")]
    public IActionResult Login(User loginUser)
    {
        var user = _context.Users.FirstOrDefault(u => u.Email == loginUser.Email);
        if (user == null || user.PasswordHash != HashPassword(loginUser.PasswordHash))
            return Unauthorized("Invalid credentials");

        var token = _authService.GenerateJwtToken(user);
        return Ok(new { token });
    }

    private string HashPassword(string password)
    {
        using var sha256 = SHA256.Create();
        return Convert.ToBase64String(sha256.ComputeHash(Encoding.UTF8.GetBytes(password)
    }
  }
}
```

Step 8: Implement CRUD Operations for Movies

Create **Controllers/MovieController.cs**:

csharp Copy Edit

```csharp
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using MovieApp.Data;
using MovieApp.Models;

namespace MovieApp.Controllers
{
    [Route("api/movies")]
    [ApiController]
    [Authorize]
    public class MovieController : ControllerBase
    {
        private readonly ApplicationDbContext _context;

        public MovieController(ApplicationDbContext context)
        {
            _context = context;
        }
```

Copy Edit

```csharp
        [HttpGet]
        public IActionResult GetAllMovies() => Ok(_context.Movies.ToList());

        [HttpGet("{id}")]
        public IActionResult GetMovie(int id) => Ok(_context.Movies.Find(id));

        [HttpPost]
        public IActionResult AddMovie(Movie movie)
        {
            _context.Movies.Add(movie);
            _context.SaveChanges();
            return Ok(movie);
        }

        [HttpPut("{id}")]
        public IActionResult UpdateMovie(int id, Movie movie)
        {
            var existing = _context.Movies.Find(id);
            if (existing == null) return NotFound();

            existing.Title = movie.Title;
            existing.Genre = movie.Genre;
            existing.Description = movie.Description;
            _context.SaveChanges();
            return Ok(existing);
        }
```

```
        [HttpDelete("{id}")]
        public IActionResult DeleteMovie(int id)
        {
            var movie = _context.Movies.Find(id);
            if (movie == null) return NotFound();

            _context.Movies.Remove(movie);
            _context.SaveChanges();
            return Ok(new { message = "Movie deleted" });
        }
    }
}
```

◇ Step 9: Configure JWT Authentication

Modify **Program.cs**:

csharp Copy Edit

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseMySql(builder.Configuration.GetConnectionString("DefaultConnection"),
    new MySqlServerVersion(new Version(8, 0, 33))));

builder.Services.AddSingleton<AuthService>();
builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)
    .AddJwtBearer(options =>
    {
        options.TokenValidationParameters = new TokenValidationParameters
        {
            ValidateIssuer = true,
            ValidateAudience = true,
            ValidateLifetime = true,
            ValidateIssuerSigningKey = true,
            ValidIssuer = builder.Configuration["JwtSettings:Issuer"],
            ValidAudience = builder.Configuration["JwtSettings:Audience"],
            IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Config
        };
    });

builder.Services.AddControllers();
builder.Services.AddAuthorization();
var app = builder.Build();

app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();
app.Run();
```