Contents

[Project plan 1](#_Toc195287676)

[Api tokens 1](#_Toc195287677)

[Read access token 1](#_Toc195287678)

[Key 1](#_Toc195287679)

[Backend (WebApi) 1](#_Toc195287680)

[Added required packages 2](#_Toc195287681)

[Service api test 3](#_Toc195287682)

[Unit tests 4](#_Toc195287683)

[Authentication 4](#_Toc195287684)

[Test login 5](#_Toc195287685)

[Front-end (React.js app) 6](#_Toc195287686)

[Steps 6](#_Toc195287687)

[Login with admin / password 6](#_Toc195287688)

# Project plan

**Backend – ASP.NET Core Web API**

* Implements a code‑first Entity Framework Core (EF Core) model for local data (e.g., user comments).
* Uses dependency injection (built‑in in ASP.NET Core) and asynchronous programming using async/await.
* Exposes endpoints that (a) proxy queries to themoviedb.org for the latest/top movies, search operations, and movie details and (b) handle local functionalities such as storing user comments.

**Frontend – React SPA**

* Created using create-react-app (optionally with TypeScript for stricter typing).
* Organized in a modular structure (components, services, hooks, etc.).
* Implements Inversion of Control (IoC) by separating business logic into services and injecting these into components.
* Consumes the backend API for asynchronous calls, using fetch/axios and async/await.
* Implements unit tests (using Jest + React Testing Library) for front-end components and use xUnit/MSTest for backend testing.

# Api tokens

## Read access token

eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJhYzEyNjU2ZDNiNjg0ZmUwNjg2M2I1NjFjMzY2Yzc5YSIsIm5iZiI6MTc0NDM2MTkzNi4wNzUsInN1YiI6IjY3ZjhkOWQwZDNhYjdkN2E4YmFkN2U5ZCIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ2ZXJzaW9uIjoxfQ.Usl5WDq9ujpiy1sl5kdJZAuqX-nZRtEXo8K4AZLEwsw

## Key

ac12656d3b684fe06863b561c366c79a

# Backend (WebApi)

dotnet new webapi -n MovieCommentsApi

Created Models

Created Data context

Register services & dependency injection

Implement Service classes

Implement Controller clases

Created unit tests

SqlLocalDb installed  
sqllocaldb info already installed

DummySqlDbInstance

MSSQLLocalDB

Added required packages  
dotnet add package Microsoft.EntityFrameworkCore

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package Microsoft.EntityFrameworkCore.Tools

dotnet add package Microsoft.EntityFrameworkCore

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package xunit

dotnet add package xunit.runner.visualstudio

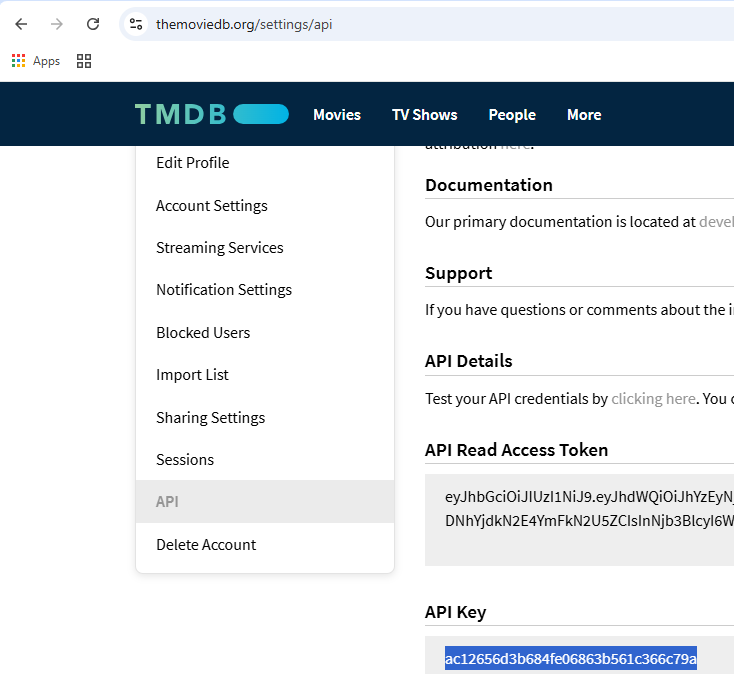
dotnet add package Microsoft.NET.Test.Sdk

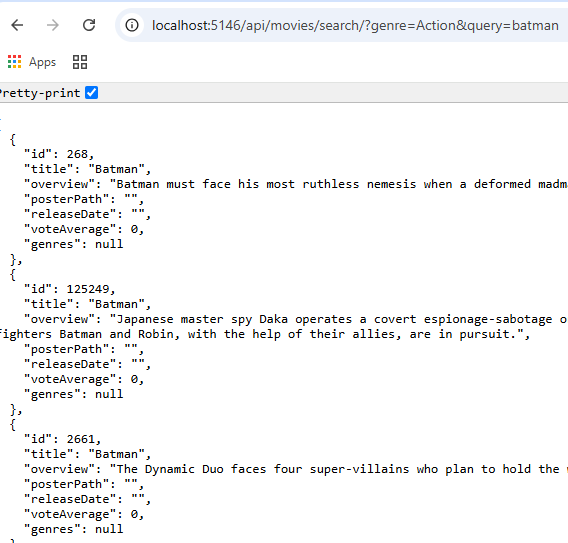
dotnet add package Swashbuckle.AspNetCore

dotnet add package Microsoft.EntityFrameworkCore.InMemory

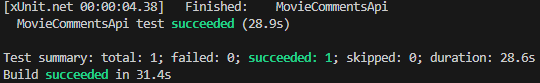
test service

## Service api test

  
<http://localhost:5146/api/movies/search/?genre=Action&query=batman>



Unit tests  
dotnet test



## Authentication

dotnet add package Microsoft.AspNetCore.Authentication.JwtBearer

Setup (Program.cs)

Protect endpoints

Add cors  
Microsoft.AspNetCore.Cors

AccountController.cs

Test login  
fetch('http://localhost:5146/api/account/login', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({

username: 'admin',

password: 'password'

})

})

.then(response => {

if (!response.ok) {

throw new Error('Login failed: ' + response.statusText);

}

return response.json();

})

.then(data => {

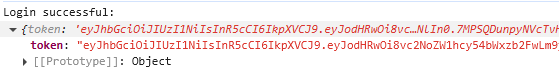
console.log('Login successful:', data);

})

.catch(error => {

console.error('Error:', error);

});



# Front-end (React.js app)

## Steps

Install Axios

Open a terminal and start the web api with: dotnet run

Open a terminal and start the react app with: dotnet start

## Login with admin / password

