# Movies Case Study – Proof of Work Report

## Table of Contents



## 1. System Architecture Diagram

The following diagram shows the overall system design:

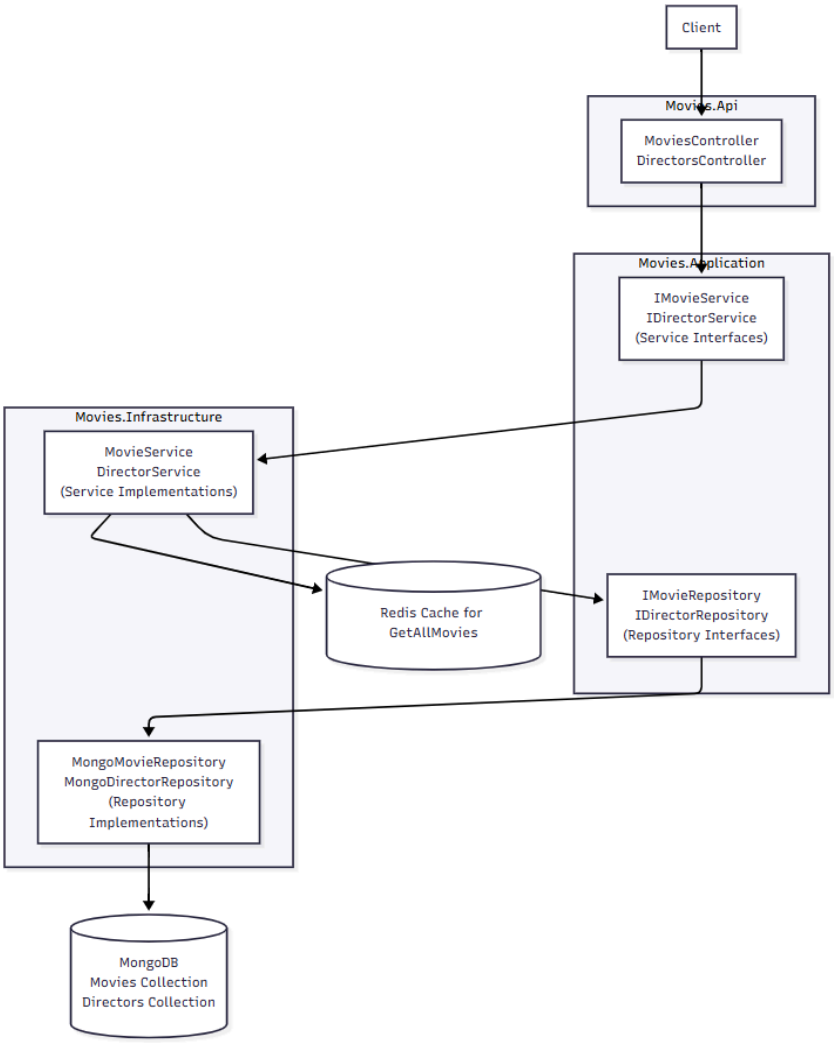


*Figure 1. System Architecture Diagram*

## 2. Executive Summary

This project is a Movie & Director Management API designed with clean architecture and containerization.

- REST API Movies & Directors
- MongoDB database with indexes
- Dockerized setup
- Global exception handling & FluentValidation
- Unit tests (Controllers & Services)
- Swagger interactive documentation

## 3. Database & Domain Design

Movies Collection Example:


```
{
  "id": "650f2e37c9a3b213ad8f91d2",
  "title": "Inception",
  "description": "A mind-bending thriller about dream invasion.",
  "releaseDate": "2010-07-16T00:00:00Z",
  "genre": "Science Fiction",
  "rating": 8.8,
  "imdbId": "tt1375666",
  "directorId": "650f2e12c9a3b213ad8f91b1"
}
```


Directors Collection Example:


```
{
  "id": "650f2e12c9a3b213ad8f91b1",
  "firstName": "Christopher",
  "lastName": "Nolan",
  "birthDate": "1970-07-30T00:00:00Z",
  "bio": "British-American director known for Inception and The Dark Knight trilogy."
}
```


Indexes:

- Unique index on imdbId – prevents duplicate movies.
- Index on directorId – faster queries by director.

## 4. API Endpoints Example

Below are selected examples of API operations and their results, covering both successful and failing scenarios. These are only sample cases, and additional situations can be tested depending on specific requirements.

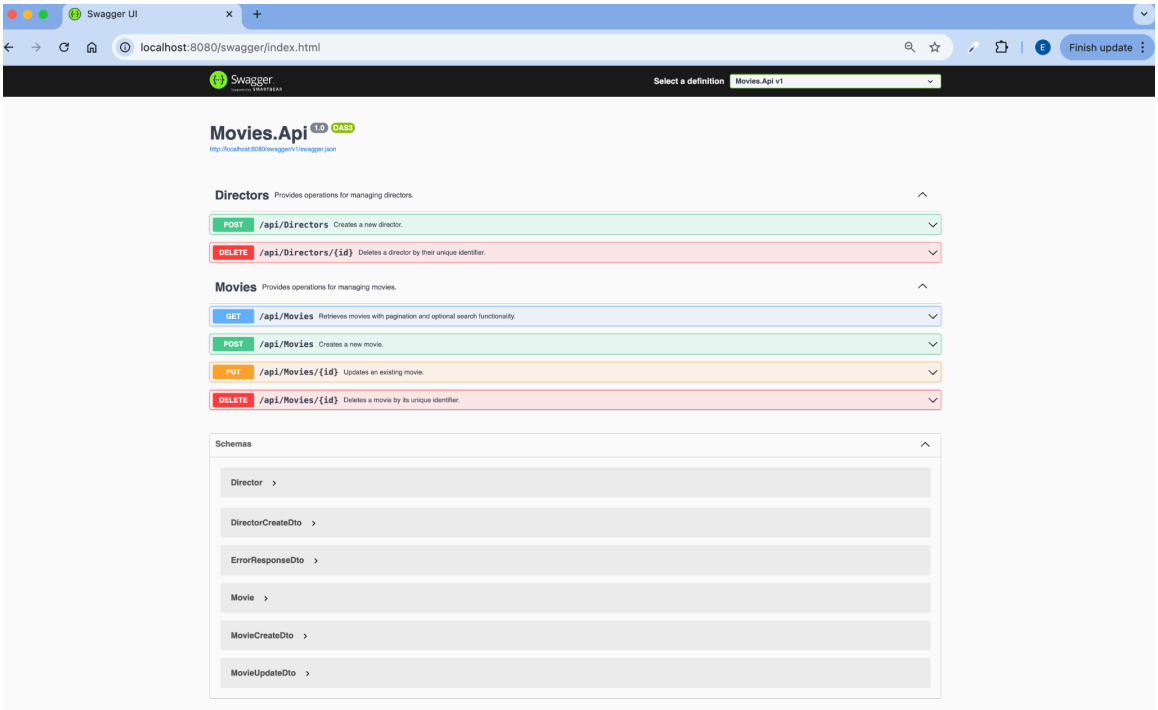


*Figure 2. Swagger UI – Movies & Directors endpoints*

**Curl**

```
curl -X 'POST' \
  'http://localhost:8080/api/Directors' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "firstName": "Christopher",
  "secondName": "Nolan",
  "birthDate": "1970-07-30T00:00:00Z",
  "bio": "British-American director known for Inception and The Dark Knight trilogy."
}'
```

**Request URL**

```
http://localhost:8080/api/Directors
```

**Server response**

| Code | Details |
|---|---|
| 201 | **Response body**<br>`{ "id": "68cee00bef5ad3158b5cd70e", "firstName": "Christopher", "secondName": "Nolan", "birthDate": "1970-07-30T00:00:00Z", "bio": "British-American director known for Inception and The Dark Knight trilogy." }`<br>**Response headers**<br>`content-type: application/json; charset=utf-8`<br>`date: Sat,20 Sep 2025 17:10:35 GMT`<br>`location: api/directors`<br>`server: Kestrel`<br>`transfer-encoding: chunked` |

**Responses**

| Code | Description |
|---|---|
| 201 | Created |

*Figure 3. Successful Director Creation*

**Curl**

```
curl -X 'POST' \
  'http://localhost:8080/api/Movies' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "title": "Inception",
  "description": "A mind-bending thriller about dream invasion.",
  "releaseDate": "2010-07-16T00:00:00Z",
  "genre": "Science Fiction",
  "rating": 8.8,
  "imdbId": "tt1375666",
  "directorId": "68cee00bef5ad3158b5cd70e"
}'
```

**Request URL**

```
http://localhost:8080/api/Movies
```

**Server response**

| Code | Details |
|---|---|
| 201 | **Response body**<br>`{ "id": "68cee063ef5ad3158b5cd710", "title": "Inception", "description": "A mind-bending thriller about dream invasion.", "releaseDate": "2010-07-16T00:00:00Z", "genre": "Science Fiction", "rating": 8.8, "imdbId": "tt1375666", "directorId": "68cee00bef5ad3158b5cd70e" }`<br>**Response headers**<br>`content-type: application/json; charset=utf-8`<br>`date: Sat,20 Sep 2025 17:12:03 GMT`<br>`location: api/movies`<br>`server: Kestrel`<br>`transfer-encoding: chunked` |

**Responses**

| Code | Description |
|---|---|
| 201 | Created |

*Figure 4. Successful Movie Creation*

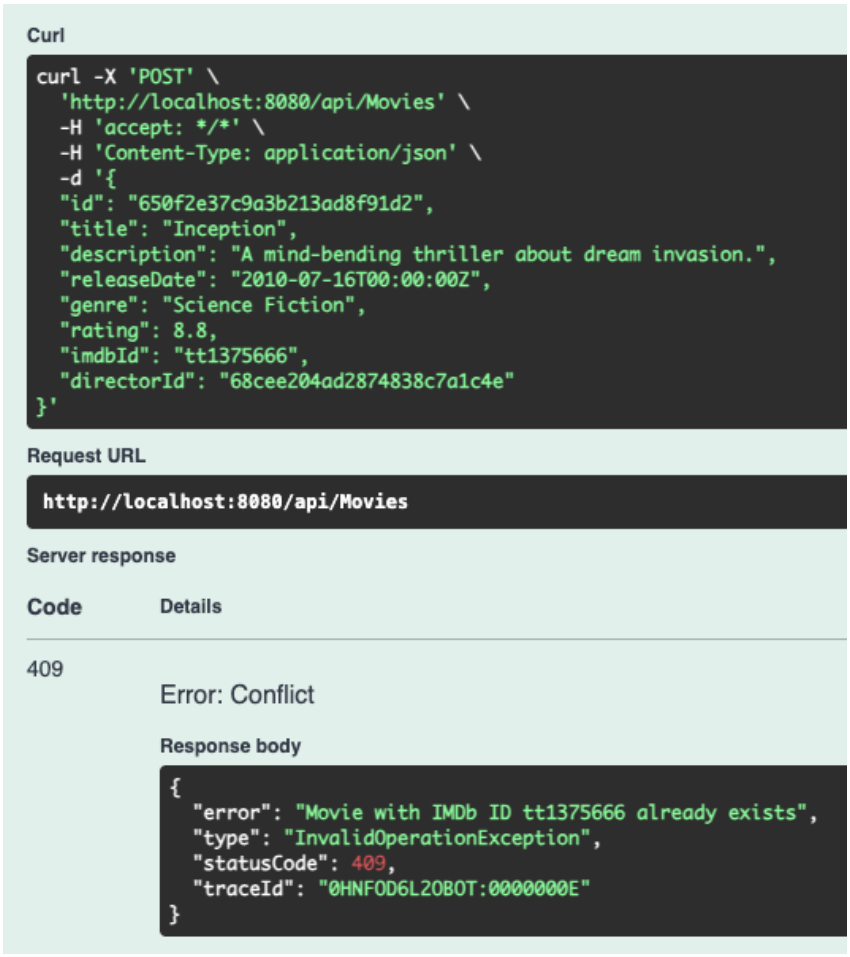


*Figure 5. Movie Conflict Error (duplicate IMDb)*

Curl

```
curl -X 'DELETE' \
  'http://localhost:8080/api/Directors/68cee204ad2874838c7a1c4e' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/api/Directors/68cee204ad2874838c7a1c4e
```

Server response

| Code | Details |
| --- | --- |
| 409 | Error: Conflict<br>Response body |

```
{
  "error": "Cannot delete director with existing movies",
  "type": "InvalidOperationException",
  "statusCode": 409,
  "traceId": "0HNFOD6L2OBOT:0000000B"
}
```

*Figure 6. Delete Director Error (existing movies)*

Curl

```
curl -X 'GET' \
  'http://localhost:8080/api/Movies?Page=1&Size=20' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/api/Movies?Page=1&Size=20
```

Server response

| Code | Details |
|---|---|
| 200 | Response body |


```
{
  "items": [
    {
      "id": "68ce9880f677b4be9d8355ea",
      "title": "Inception",
      "description": "A mind-bending thriller about dream invasion.",
      "releaseDate": "2010-07-16T00:00:00Z",
      "genre": "Science Fiction",
      "rating": 8.8,
      "imdbId": "tt1375666",
      "directorId": "68ce980af677b4be9d8355e8"
    }
  ],
  "page": 1,
  "size": 20,
  "searchText": null,
  "total": 1,
  "totalPages": 1,
  "hasNext": false,
  "hasPrevious": false
}
```


*Figure 7. Get Movie List with Pagination*

## 5. Unit Tests

Created controller and service tests using xUnit, Moq, and FluentAssertions.

These tests cover success and error scenarios for both Movies and Directors.

## 6. Proof of Work

```
elifdemirel@macbook-pro movies-case-study % docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS                                                NAMES
f56adf2738e7   movies-case-study-movies-api    "dotnet Movies.Api.d…"   5 seconds ago    Up 4 seconds    0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp          movies-api
896bd7564b20   mongo-express:1.0.0-alpha.4    "tini -- /docker-ent…"   3 minutes ago    Up 3 seconds    0.0.0.0:8081->8081/tcp, [::]:8081->8081/tcp          mongo-express
57d725d4cb0b   mongo:7                        "docker-entrypoint.s…"   3 minutes ago    Up 4 seconds    0.0.0.0:27017->27017/tcp, [::]:27017->27017/tcp      mongo
e496c01db07b   redis:7-alpine                 "docker-entrypoint.s…"   3 minutes ago    Up 4 seconds    0.0.0.0:6379->6379/tcp, [::]:6379->6379/tcp          redis
```

*Figure 8. Running containers with Docker*

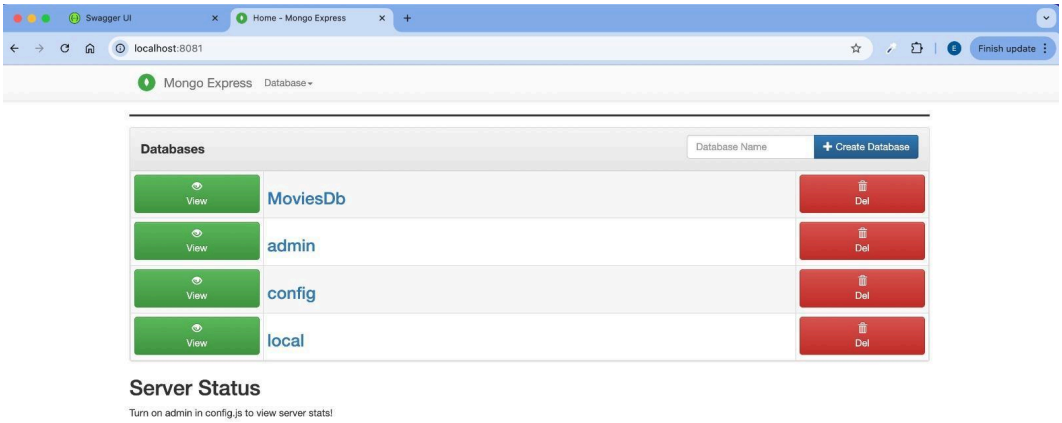


*Figure 9. Mongo Express – MoviesDb database*

```
elifdemirel@macbook-pro movies-case-study % dotnet test
Restore complete (0,6s)
  Movies.Domain succeeded (0,1s) → src/Movies.Domain/bin/Debug/net9.0/Movies.Domain.dll
  Movies.Application succeeded (0,1s) → src/Movies.Application/bin/Debug/net9.0/Movies.Application.dll
  Movies.Infrastructure succeeded (0,1s) → src/Movies.Infrastructure/bin/Debug/net9.0/Movies.Infrastructure.dll
  Movies.Api succeeded (0,2s) → src/Movies.Api/bin/Debug/net9.0/Movies.Api.dll
  Movies.Tests succeeded (0,2s) → tests/Movies.Tests/bin/Debug/net9.0/Movies.Tests.dll
[xUnit.net 00:00:00.00] xUnit.net VSTest Adapter v3.1.4+50e68bbb8b (64-bit .NET 9.0.8)
[xUnit.net 00:00:00.04]   Discovering: Movies.Tests
[xUnit.net 00:00:00.06]   Discovered:  Movies.Tests
[xUnit.net 00:00:00.06]   Starting:    Movies.Tests
     Warning:
     The component "Fluent Assertions" is governed by the rules defined in the Xceed License Agreement and
     the Xceed Fluent Assertions Community License. You may use Fluent Assertions free of charge for
     non-commercial use only. An active subscription is required to use Fluent Assertions for commercial use.
     Please contact Xceed Sales mailto:sales@xceed.com to acquire a subscription at a very low cost.
     A paid commercial license supports the development and continued increasing support of
     Fluent Assertions users under both commercial and community licenses. Help us
     keep Fluent Assertions at the forefront of unit testing.
     For more information, visit https://xceed.com/products/unit-testing/fluent-assertions/
[xUnit.net 00:00:00.15]   Finished:    Movies.Tests
  Movies.Tests test succeeded (0,5s)

Test summary: total: 12, failed: 0, succeeded: 12, skipped: 0, duration: 0,5s
Build succeeded in 1,9s
```

*Figure 10. All unit tests passing (12/12)*

## 7. Conclusion

This project demonstrates my ability to build clean, testable backend systems following industry best practices including clean architecture, containerization, automated testing, and proper error handling.

The Movies API showcases my experience in key backend development areas:

- RESTful API design with clear structure
- Unit testing for controllers and services
- Docker and MongoDB integration
- API documentation with Swagger

This case study reflects my approach to backend development: simple, structured, maintainable, and production-ready solutions.