

Assignment 1 – CSD-3353 Web Applications in C#.NET

This project was created and run using Visual studio .Net 9.0. Further instructions and application functionality are for the same.

Steps to Run:

1. Press F5 to run the program. (Press ctrl-shift-B to build the program if needed).
2. Web application would have started in your default browser with address:
<https://localhost:7177/>

Web application functionality:

1. Index page: as you run the project you will be redirected to the default page i.e. Index page as mentioned below. Also, the list of pre included movies are visible by default.

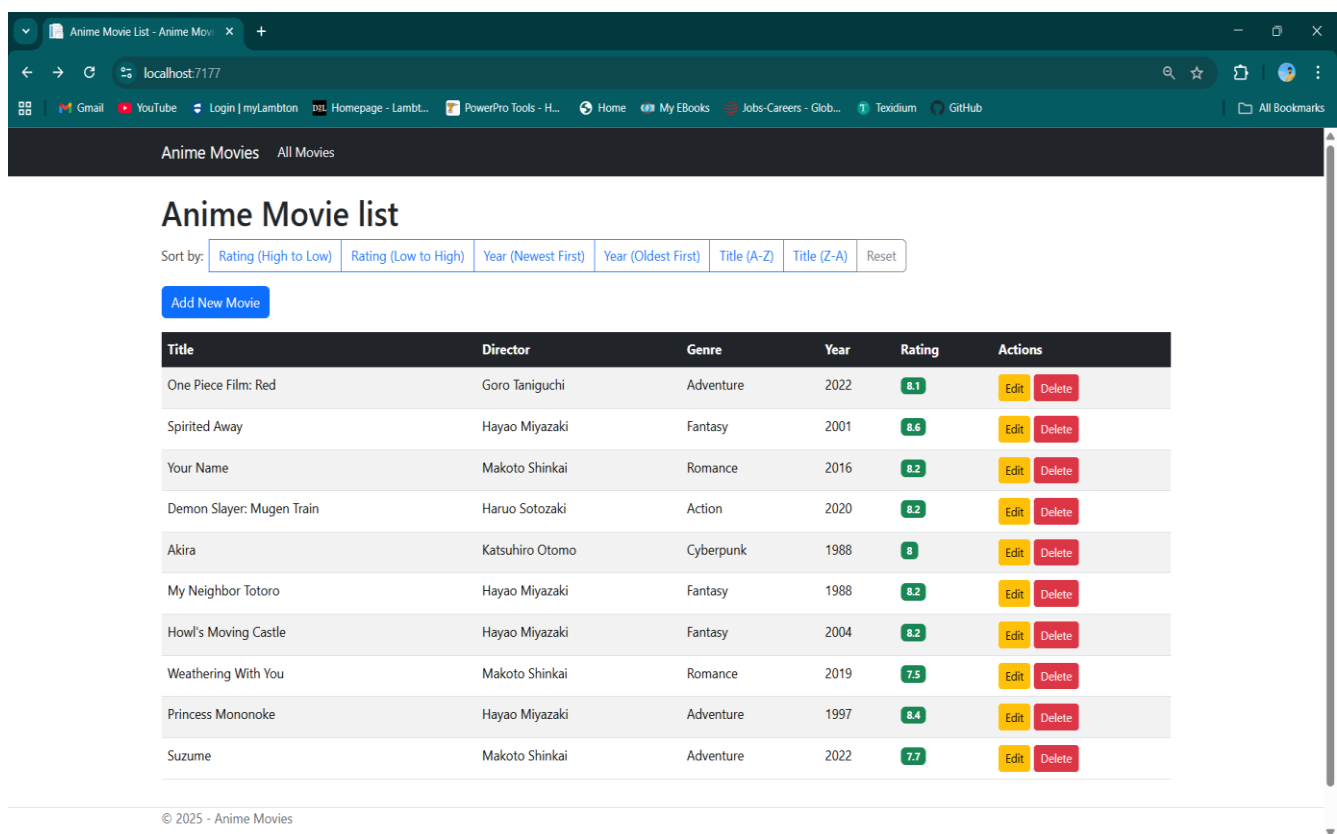


Figure 1 List of movies

2. Sorting: Additional feature is added in this project, for sorting the list as required. User can sort by rating, year, or title.
3. Create or add movie: By clicking on “Add new movie”, user will be directed to a new page where new movie details are required to add it.

Add a new Anime Movie - Anim

localhost:7177/Movies/Create

Gmail

YouTube

Login | myLambton

Homepage - Lambt...

PowerPro Tools - H...

Anime Movies

All Movies

Add a new Anime Movie

Title

Director

Genre

ReleaseYear

Rating

Create

Back to List

© 2025 - Anime Movies

Figure 2 Create page

Add a new Anime Movie - Anim

localhost:7177/Movies/Create

Gmail

YouTube

Login | myLambton

Homepage - Lambt...

PowerPro Tools - H...

Anime Movies

All Movies

Add a new Anime Movie

Title

New

Director

Unknown

Genre

Genre

ReleaseYear

2000

Rating

10.0

Create

Back to List

© 2025 - Anime Movies

Figure 3 Add data

Anime Movie list

Sort by: [Rating \(High to Low\)](#) [Rating \(Low to High\)](#) [Year \(Newest First\)](#) [Year \(Oldest First\)](#) [Title \(A-Z\)](#) [Title \(Z-A\)](#) [Reset](#)

[Add New Movie](#)

Title	Director	Genre	Year	Rating	Actions	
One Piece Film: Red	Goro Taniguchi	Adventure	2022	8.1	Edit	Delete
Spirited Away	Hayao Miyazaki	Fantasy	2001	8.6	Edit	Delete
Your Name	Makoto Shinkai	Romance	2016	8.2	Edit	Delete
Demon Slayer: Mugen Train	Haruo Sotozaki	Action	2020	8.2	Edit	Delete
Akira	Katsuhiro Otomo	Cyberpunk	1988	8	Edit	Delete
My Neighbor Totoro	Hayao Miyazaki	Fantasy	1988	8.2	Edit	Delete
Howl's Moving Castle	Hayao Miyazaki	Fantasy	2004	8.2	Edit	Delete
Weathering With You	Makoto Shinkai	Romance	2019	7.5	Edit	Delete
Princess Mononoke	Hayao Miyazaki	Adventure	1997	8.4	Edit	Delete
Suzume	Makoto Shinkai	Adventure	2022	7.7	Edit	Delete
New	Unknown	Genre	2000	10	Edit	Delete

© 2025 - Anime Movies

Figure 4 Result after adding new movie

4. Edit page: By clicking on “edit”, user will be directed to edit page where the selected movie data is prepopulated and can be edited.

The screenshot shows a web browser window with the URL `localhost:7177/Movies/Edit/12`. The page title is "Edit a Movie". The form contains the following fields:

- Title: New
- Director: Unknown
- Genre: Genre
- ReleaseYear: 2000
- Rating: 10

At the bottom of the form are two buttons: "Save" and "Back to List". The footer of the page reads "© 2025 - Anime Movies".

Figure 5 Pre-populated data of certain movie to edit

he Mov

+

st:7177/Movies/Edit/12

Login | myLambton

D1L Homepage - Lambt...

PowerPro Tools - H...

Home

My E

Anime Movies

All Movies

Edit a Movie

Title

New1

Director

Unknown

Genre

Genre

ReleaseYear

2000

Rating

10

Save

Back to List

© 2025 - Anime Movies

Figure 6 Editing data

77

Login | myLambton

D1L Homepage - Lambt...

PowerPro Tools - H...

Home

My EBooks

Jobs-Careers - Glob...

Texidium

GitHub

Anime Movies

All Movies

Anime Movie list

Sort by:

Rating (High to Low)

Rating (Low to High)

Year (Newest First)

Year (Oldest First)

Title (A-Z)

Title (Z-A)

Reset

Add New Movie

Title	Director	Genre	Year	Rating	Actions
One Piece Film: Red	Goro Taniguchi	Adventure	2022	8.1	<div>EditDelete</div>
Spirited Away	Hayao Miyazaki	Fantasy	2001	8.6	<div>EditDelete</div>
Your Name	Makoto Shinkai	Romance	2016	8.2	<div>EditDelete</div>
Demon Slayer: Mugen Train	Haruo Sotozaki	Action	2020	8.2	<div>EditDelete</div>
Akira	Katsuhiro Otomo	Cyberpunk	1988	8	<div>EditDelete</div>
My Neighbor Totoro	Hayao Miyazaki	Fantasy	1988	8.2	<div>EditDelete</div>
Howl's Moving Castle	Hayao Miyazaki	Fantasy	2004	8.2	<div>EditDelete</div>
Weathering With You	Makoto Shinkai	Romance	2019	7.5	<div>EditDelete</div>
Princess Mononoke	Hayao Miyazaki	Adventure	1997	8.4	<div>EditDelete</div>
Suzume	Makoto Shinkai	Adventure	2022	7.7	<div>EditDelete</div>
New1	Unknown	Genre	2000	10	<div>EditDelete</div>

© 2025 - Anime Movies

Figure 7 Result after editing data

5. Delete page: By clicking on” delete”, user will be directed to the delete page where confirmation will be asked to delete certain movie.

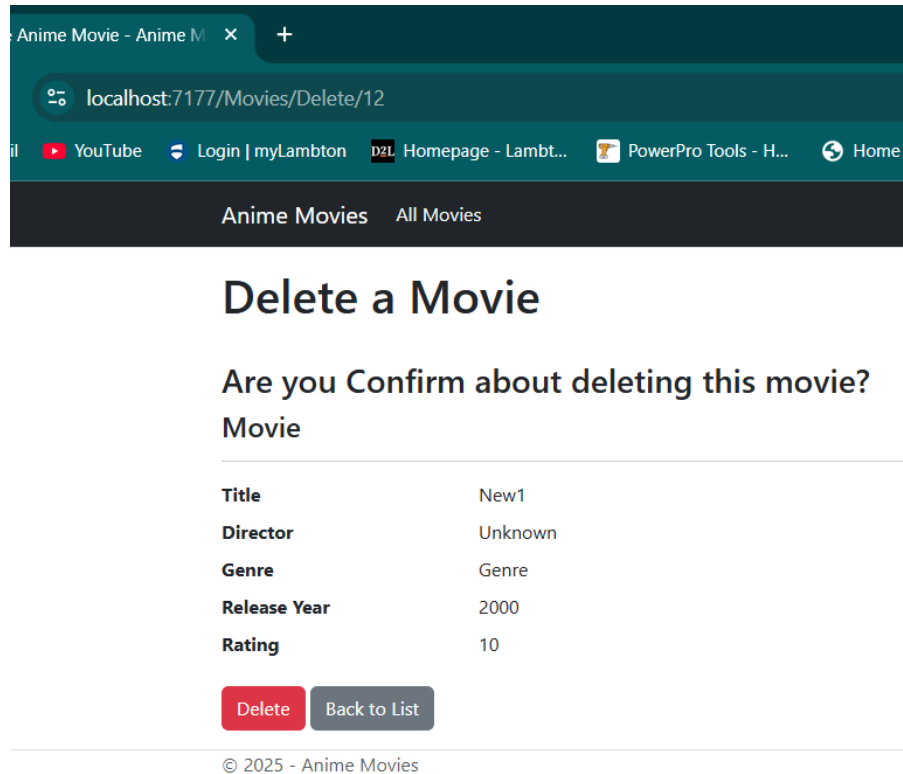


Figure 8 Confirmation to delete movie

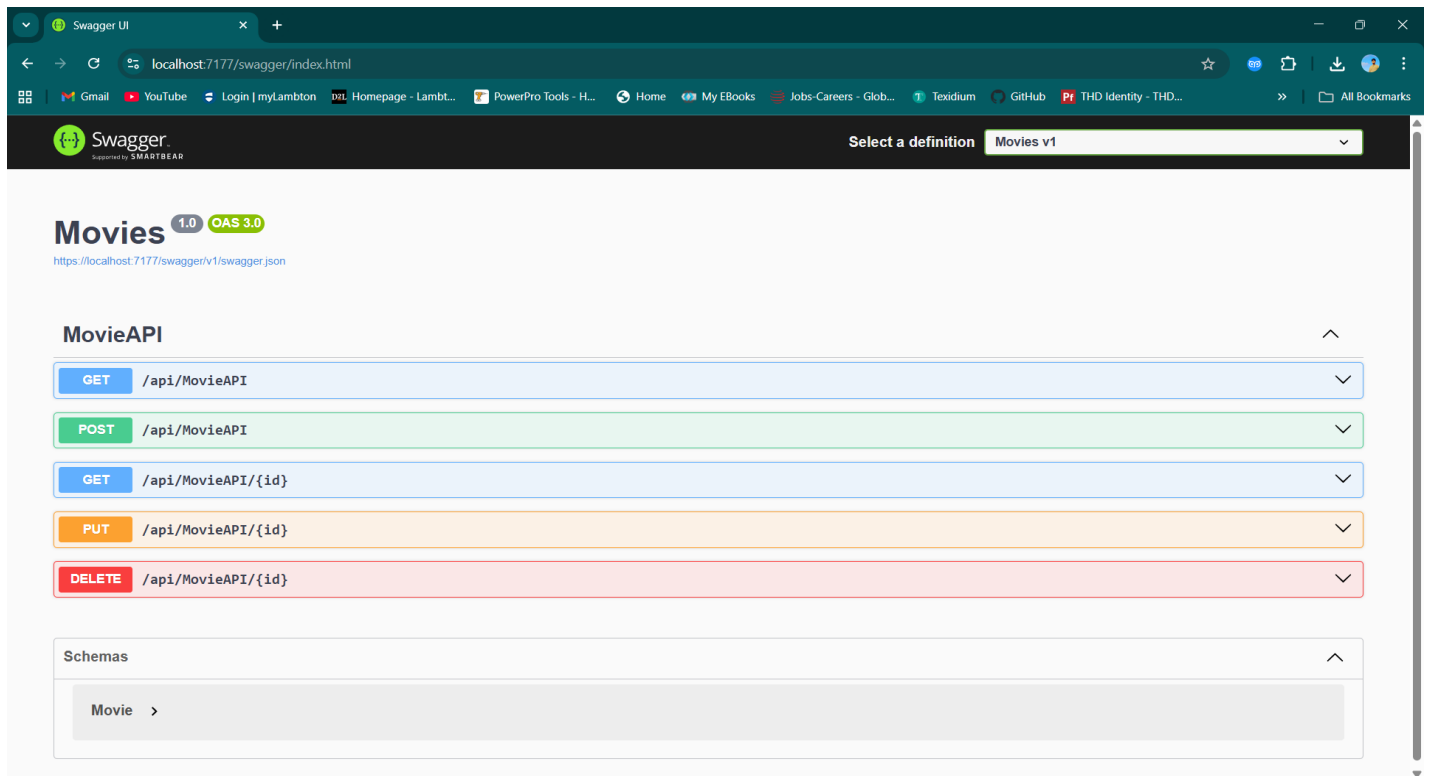
After pressing on delete the movie will be deleted from the list.

Assignment 2 – CSD-3353 Web Applications in C#.NET

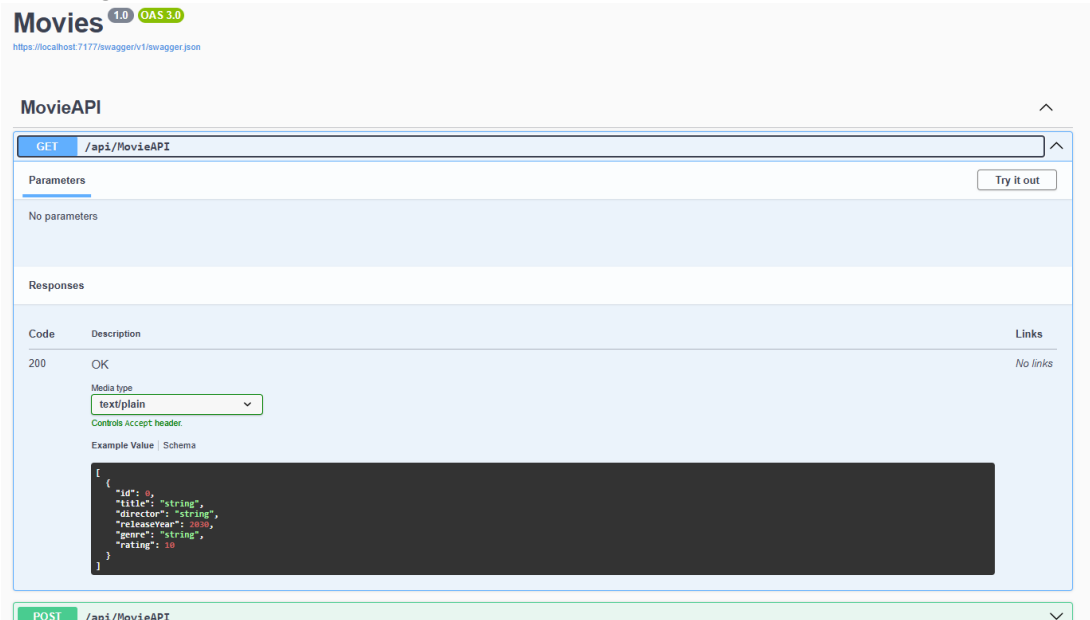
Enhanced Movies Manager – EF Core + API + Validation

Steps to run the program and Testing API:

- 1) Open project in Code editor (Extract if needed)
- 2) Press Ctrl+shift+B to build the project
- 3) Hot run the project or press 'F5'
- 4) If project does not run perfectly, there might be package or dependency problem you need to install or configure.
- 5) If project runs perfectly, a new web page will open. With the link of local host and port number. Add or write '/swagger/index.html' where we will test API working.



- 6) Testing GET for all movies: tap on it, Click on 'try it out'. Click on execute. If you see code 200, your test run fine. You will be able to see all the Movies.



Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7177/api/MovieAPI' \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7177/api/MovieAPI

Server response

Code	Details
200	<p>Response body</p> <pre>[{ "id": 1, "title": "One Piece Film: Red UPDATED", "director": "Goro Taniguchi", "releaseYear": 2022, "genre": "Adventure", "rating": 9 }, { "id": 2, "title": "Spirited Away", "director": "Hayao Miyazaki", "releaseYear": 2001, "genre": "Fantasy", "rating": 8.6 }, { "id": 3, "title": "Your Name", "director": "Makoto Shinkai", "releaseYear": 2016, "genre": "Romance", "rating": 8.2 }, { "id": 4, "title": "Demon Slayer: Mugen Train", "releaseYear": 2020, "genre": "Action", "rating": 8.7 }]</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Tue, 04 Nov 2025 03:34:09 GMT server: Kestrel</pre>

Responses

Code	Description	Links
200	OK	No links

- 7) Testing GET for single movie: tap on it, Click on 'try it out'. Add, 'ID' in the parameter. Click on execute. If you see code 200, your test run fine. You will be able to see all the selected Movie.

GET

/api/MovieAPI/{id}

Parameters

Cancel

Name	Description
id * required integer(\$int32) (path)	<div>1</div>

Execute

Responses

Code	Description	Links
200	<div>OK</div> <div>Media type text/plain</div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <div><pre>{ "id": 0, "title": "string", "director": "string", "releaseYear": 2030, "genre": "string", "rating": 10}</pre></div>	No links

GET

/api/MovieAPI/{id}

Parameters

Cancel

Name	Description
id * required integer(\$int32) (path)	<div>1</div>

Execute

Clear

Responses

Curl

curl -X 'GET' \ 'https://localhost:7177/api/MovieAPI/1' \ -H 'accept: text/plain'

Request URL

https://localhost:7177/api/MovieAPI/1

Server response

Code	Details
200	<div>Response body</div> <div><pre>{ "id": 1, "title": "One Piece Film: Red UPDATED", "director": "Goro Taniguchi", "releaseYear": 2022, "genre": "Adventure", "rating": 9}</pre></div> <div>Response headers</div> <div>content-type: application/json; charset=utf-8 date: Tue,04 Nov 2025 03:35:06 GMT server: Kestrel</div>

Responses

Code	Description	Links
200	OK	No links

- 8) Testing POST to add a movie: tap on it, Click on 'try it out'. Add a new movie detail in JSON. Click on execute. If you see code 201, your test run fine. You will be able to see the added movie.

The screenshot shows a REST client interface with the following sections:

- POST /api/MovieAPI**: The method and endpoint.
- Parameters**: A section with "No parameters" and buttons for "Cancel" and "Reset".
- Request body**: A dropdown menu set to "application/json".
- Edit Value | Schema**: A text area containing a JSON object:

```
{  "title": "string1",  "director": "string2",  "releaseYear": 2025,  "genre": "string3",  "rating": 10}
```
- Execute**: A large blue button to execute the request.
- Responses**: A table showing the response details.

Code	Description	Links
200	OK	No links
- Media type**: A dropdown menu set to "text/plain".
- Controls Accept header**: A green message indicating the header is controlled.
- Example Value | Schema**: A text area showing a partial JSON response:

```
{  "id": 0,  "title": "string",
```

The screenshot shows a REST client interface with the following sections:

- Curl**: A text area containing a curl command:

```
curl -X 'POST' \  "https://localhost:7177/api/MovieAPI" \  -H 'accept: text/plain' \  -H 'content-type: application/json' \  -d '{  "title": "string1",  "director": "string2",  "releaseYear": 2025,  "genre": "string3",  "rating": 10  }'
```
- Request URL**: A text area containing the URL "https://localhost:7177/api/MovieAPI".
- Server response**: A section showing the response details.

Code	Details
201	Response body
- Response body**: A text area containing a JSON object:

```
{  "id": 13,  "title": "string1",  "director": "string2",  "releaseYear": 2025,  "genre": "string3",  "rating": 10}
```
- Response headers**: A text area containing headers:

```
content-type: application/json; charset=utf-8  date: Tue, 04 Nov 2025 03:36:39 GMT  location: https://localhost:7177/api/MovieAPI/13  server: Kestrel
```
- Responses**: A section at the bottom.

- 9) Testing PUT to update a movie: tap on it, Click on 'try it out'. Update a movie detail in populated JSON. Click on execute. If you see code 204, your test run fine. You will be able to see the updated movie.

PUT

/api/MovieAPI/{id}

Parameters

Cancel

Reset

Name	Description
id * required	
integer(\$int32)	1
(path)	

Request body

application/json

Edit Value | Schema

```
{
  "id": 1,
  "title": "1string",
  "director": "2string",
  "releaseYear": 2024,
  "genre": "3string",
  "rating": 9
}
```

Execute

Responses

Curl

```
curl -X 'PUT' \
  'https://localhost:7177/api/MovieAPI/1' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 1,
    "title": "1string",
    "director": "2string",
    "releaseYear": 2024,
    "genre": "3string",
    "rating": 9
  }'
```

Request URL

https://localhost:7177/api/MovieAPI/1

Server response

Code	Details
204	
Undocumented	<div>Response headers<div>date: Tue, 04 Nov 2025 03:39:06 GMT server: Kestrel</div></div>

Responses

Code	Description	Links
200	OK	No links

10) Testing DELETE to delete a movie: tap on it, Click on 'try it out'. Enter 'id' number in parameter. Click on execute. If you see code 204, your test run fine. You would have deleted the assigned movie.

DELETE /api/MovieAPI/{id}

Parameters

Cancel

Name	Description
id * required	
integer(\$int32)	1
(path)	

Execute

Responses

Code	Description	Links
200	OK	No links

Name	Description
id * required	
integer(\$int32)	1
(path)	

Execute

Clear

Responses

Curl

```
curl -X 'DELETE' \
  'https://localhost:7177/api/MovieAPI/1' \
  -H 'accept: */*'
```

Request URL

```
https://localhost:7177/api/MovieAPI/1
```

Server response

Code	Details
204	

Undocumented

Response headers

```
date: Tue, 04 Nov 2025 03:39:52 GMT
server: Kestrel
```

Responses

11) After all the testing try step 6 again. There you would be able to see all the updating that were performed during the testing. Also, same updates would be visible in the home page or index page.

The screenshot shows a web browser's developer tools interface. At the top, the 'Curl' tab is active, displaying the command: `curl -X 'GET' \ 'https://localhost:7177/api/MovieAPI' \ -H 'accept: text/plain'`. Below this, the 'Request URL' is `https://localhost:7177/api/MovieAPI`. The 'Server response' section shows a status code of 200. The 'Response body' is a JSON array of movie data:

```
[
  {
    "id": 2,
    "title": "Spirited Away",
    "director": "Hayao Miyazaki",
    "releaseYear": 2001,
    "genre": "Fantasy",
    "rating": 8.6
  },
  {
    "id": 3,
    "title": "Your Name",
    "director": "Makoto Shinkai",
    "releaseYear": 2016,
    "genre": "Romance",
    "rating": 8.2
  },
  {
    "id": 4,
    "title": "Demon Slayer: Mugen Train",
    "director": "Haruo Sotozaki",
    "releaseYear": 2020,
    "genre": "Action",
    "rating": 8.2
  },
  {
    "id": 5,
    "title": "Akira",
    "director": "Katsuhiro Motoki",
    "releaseYear": 1988,
    "genre": "Action",
    "rating": 8.0
  }
]
```

Below the response body, the 'Response headers' are shown:

```
content-type: application/json; charset=utf-8
date: Tue, 04 Nov 2025 03:40:29 GMT
server: Kestrel
```

At the bottom, the 'Responses' tab is active, showing a table with columns 'Code', 'Description', and 'Links'.

Validation: The following picture displays the validation that was added to the form for individual inputs.

[Anime Movies](#) [All Movies](#)

Add a new Anime Movie

Title

Movie title is mandatory

Director

Director name is mandatory

Genre

Genre is mandatory

ReleaseYear

Release year is mandatory

Rating

Rating must be between 0.0 and 10.0

[Create](#)[Back to List](#)

Assignment 3 – CSD-3353 Web Applications in C#.NET

Part 1: MongoDB Atlas Setup screenshots

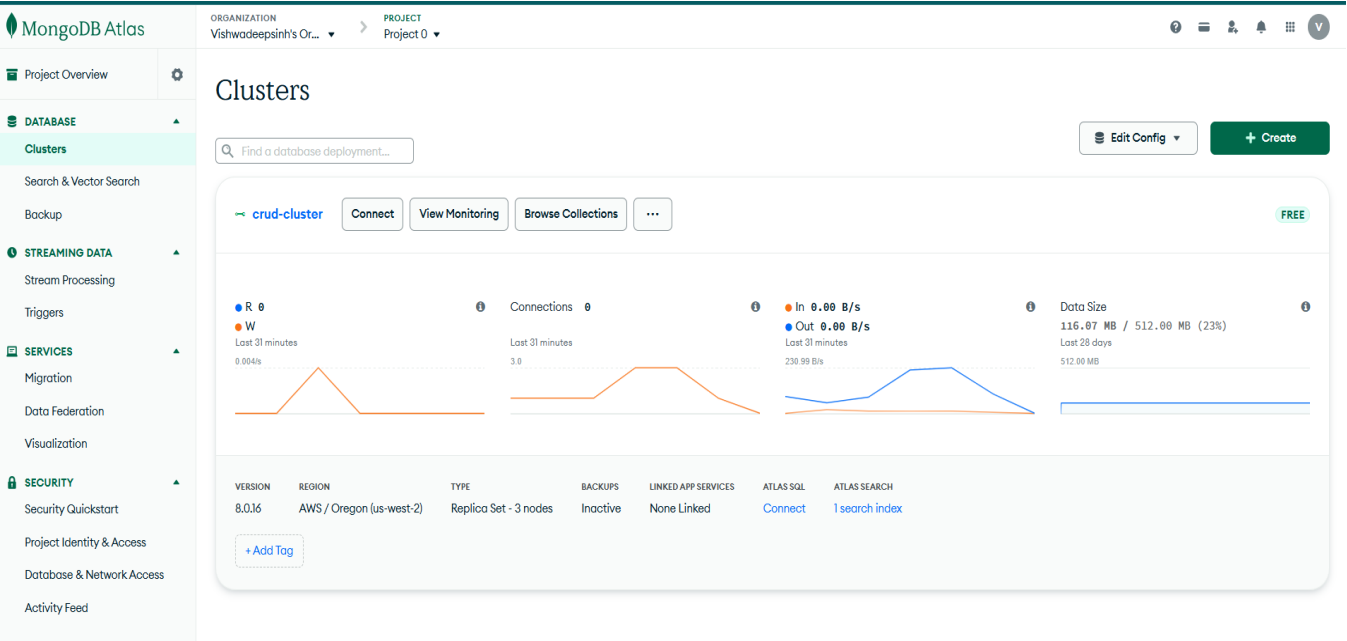


Figure 9 Cluster

Authentication Method

Password

Certificate

AWS IAM

Federated Auth (MongoDB 7.0 and up)

MongoDB uses SCRAM as its default authentication method.

Password Authentication

Edit Password

User Description

Add an optional description to your user.

Database User Privileges

Configure role based access control by assigning database user a mix of one built-in role, multiple custom roles, and multiple specific privileges. A user will gain access to all actions within the roles assigned to them, not just the actions those roles share in common. You must choose at least one role or privilege. [Learn more about roles.](#)

Built-in Role

Select one built-in role for this user.

Read and write to any database

Custom Roles

Select your pre-defined custom role(s). Create a custom role in the Custom Roles tab.

Specific Privileges

Select multiple privileges and what database and collection they are associated with. Leaving collection blank will grant this role for all collections in the database.

Restrict Access to Specific Clusters/Federated Database Instances/Stream Processing Workspaces

Enable to specify the resources this user can access. By default, all resources in this project are accessible.

Cancel

Update User

Figure 10 Privileges read/write permissions



Edit IP Access List Entry

Atlas only allows client connections to a cluster from entries in the project's IP Access List. Each entry should either be a single IP address or a CIDR-notated range of addresses. [Learn more](#)

ADD CURRENT IP ADDRESS

Access List Entry:

0.0.0.0/0

Comment:

Optional comment describing this entry

Cancel

Confirm

Figure 11 Network access

Steps to setup MongoDB:

- Created MongoDB account and a free cluster
- Create a database and gives read/write privileges
- Edit IP address to 0.0.0.0/0 to allow access from anywhere
- Connect the cluster and select application(C#/.NET) and copy the connection string

Part 2: Token generation & Usage

Steps:

1. Build and run the program. If runs perfectly localhost:7177 page will open
2. Move to /swagger/index page address.
3. Now we will check each function with authentication and authorisation
4. First of register a user. Click on POST: /api/Auth/register => Try it out => fill unique details => execute.

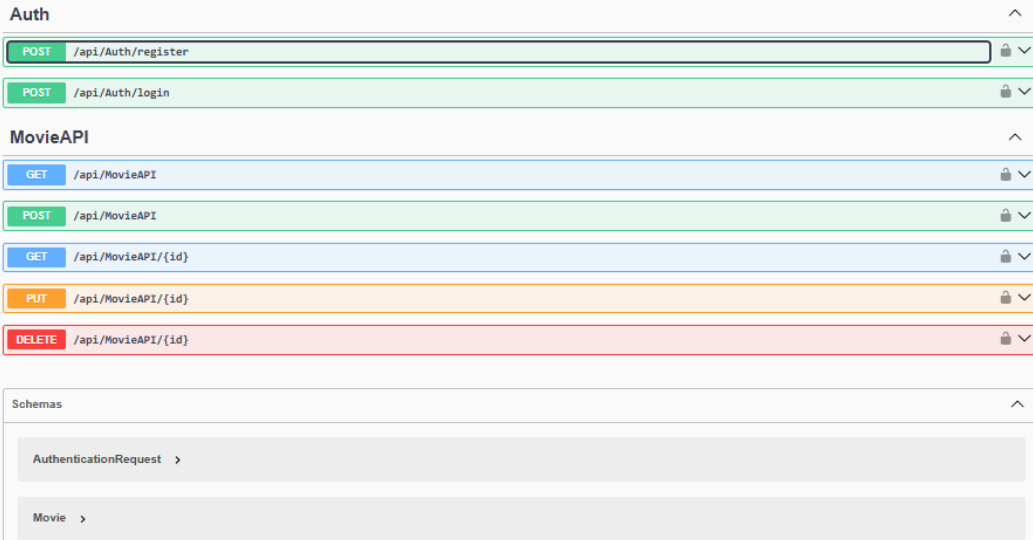


Figure 12 Swagger/Index page

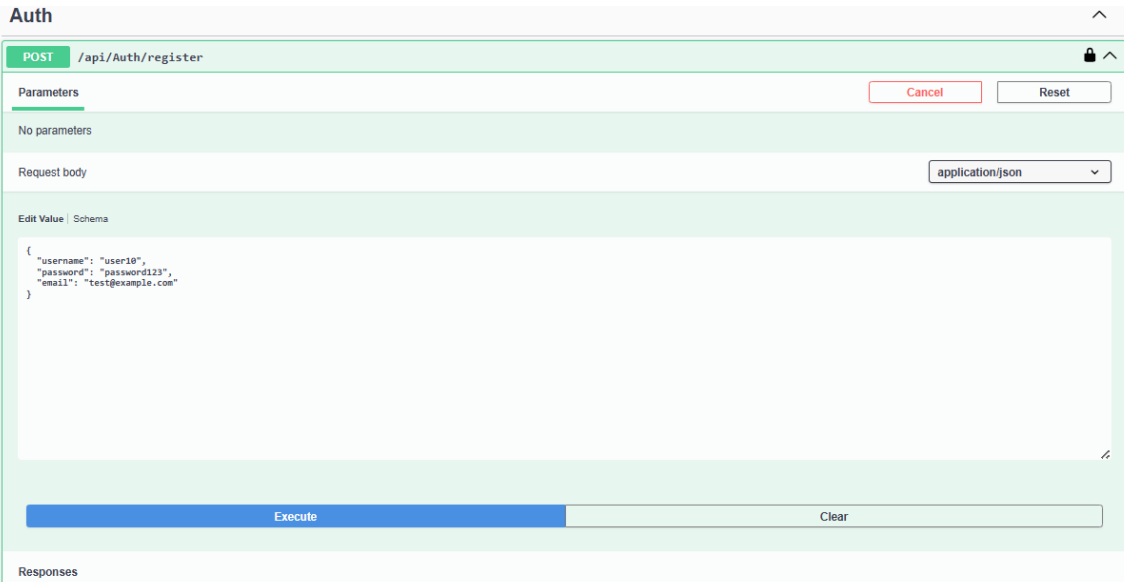


Figure 13 Register a new user

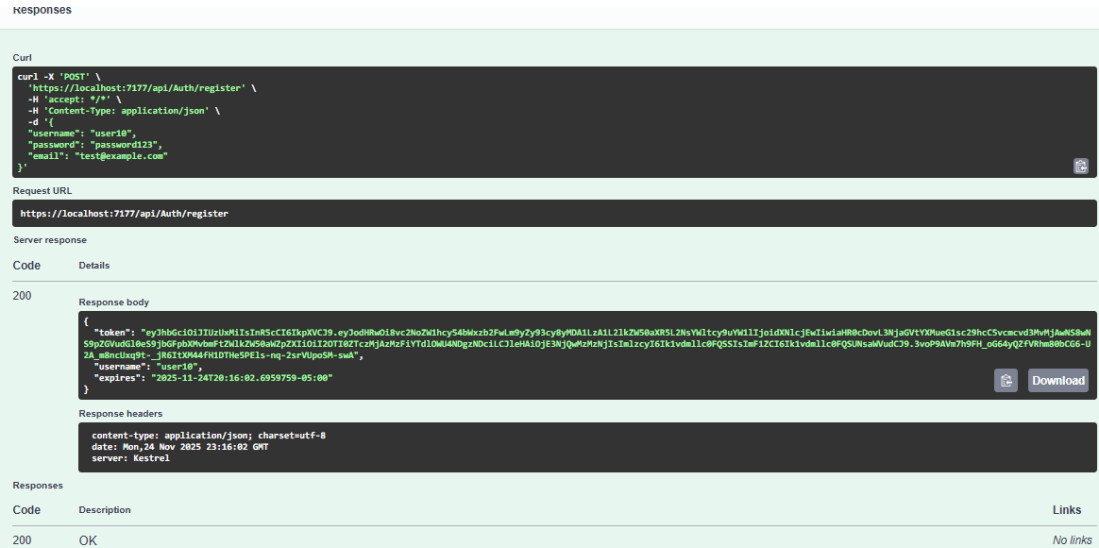




Figure 14 Confirmation after executing

POST

/api/Auth/login



Parameters

Cancel

Reset

No parameters

Request body

application/json

Edit Value

Schema

```
{
  "username": "user10",
  "password": "password123"
}
```

Execute

Clear

Responses

Figure 15 Login using Username and password

Responses

Curl

```
curl -X 'POST' \
'https://localhost:7177/api/Auth/login' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
  "username": "user10",
  "password": "password123"
}'
```

Request URL

```
https://localhost:7177/api/auth/login
```

Server response

Code	Details
200	<div><div>Response body</div><div><pre>{ "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IWRkcSIsImVudCI6IjEwMjYyOTQxMTI0OTI0ZmZlcnJhdGEzF1VTd1OWI0NDgzNDc1CjEhHAJOEjNjQWtKzDkScmlzcjE1Ikdldmllc0FQSStS1mf1ZCTG16Iklvdml1c0FQSUNscmVudC9yL043_30d35fuce8d513bszbR-PANALYHzpgrt4Gbu_y318FrAHSISvNvDTFS9YT2Z28odABN1OKIEB-TbMGtIQ", "username": "user10", "expires": "2025-11-24T20:16:29.0662156-05:00" }</pre></div><div><div>Download</div></div></div>
	<div><div>Response headers</div><div><pre>content-type: application/json; charset=utf-8 date: Mon, 24 Nov 2025 23:16:28 GMT server: Kestrel</pre></div></div>

Responses

Code	Description	Links
200	OK	No links

Figure 16 Login Confirmation

- Another way to login using token number: Search for Authorize button on index page click on that. It will ask for token, Enter the token and you will be authorized for further mentioned function accessibility.

Available authorizations

Bearer (http, Bearer)

Please enter a valid token

Value:

Authorize

Close

Figure 17 Insert token for authorization



- Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7177/api/MovieAPI' \
  -H 'accept: text/plain' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZWlkeWZlLnR5cy8yMDAxLzA1LzI1LzI1ZS80XzUyM1tscy9uYm11IjoiaXN1c2IwIiwiaWF0IjEwIjoiZWlkaHR8CDoVt3NjJaGTVXQWUeG1sc29hcC5vcmcvd3RwJGJwM58'
```

Request URL

https://localhost:7177/api/MovieAPI

Server response

CodeDetails

200

Response body

[]

Download

Response headers

content-type: application/json; charset=utf-8
date: Mon, 24 Nov 2025 23:26:03 GMT
server: Kestrel

Responses

CodeDescriptionLinks



200OK

No links

Figure 19 Shows [], when no data found

- POST

/api/MovieAPI



Parameters

Cancel

Reset

No parameters


Request body

application/json

▼

Edit Value | Schema

```
{
  "title": "The Matrix",
  "director": "Lana Wachowski, Lilly Wachowski",
  "releaseYear": 1999,
  "genre": "Sci-Fi",
  "rating": 8.7
}
```



Execute

Clear



Responses

Figure 20 Post a movie



- PUT

/api/MovieAPI/{id}



Parameters

Cancel

Reset

Name	Description
id ^{* required}	
string	<input type="text" value="6924ec02835ffd0b206369b9"/>
(path)	

Request body

application/json

▼

Edit Value | Schema

```
{
  "id": "string",
  "title": "string",
  "director": "string",
  "releaseYear": 2030,
  "genre": "string",
  "rating": 10,
  "createdAt": "2025-11-24T23:43:29.877Z",
  "createdBy": "string"
}
```

↕

Execute

Figure 22 Enter ID and edit value information

The screenshot shows a REST client interface with a blue 'Execute' button and a 'Clear' button. Below the buttons, the 'Responses' tab is active, displaying a successful PUT request. The 'Curl' section shows the command: `curl -X 'PUT' \ 'https://localhost:7177/api/MovieAPI/6924ec02835ffdb0b206369b9' \ -H 'accept: */*' \ -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy44bWxzZ2Fubm9yZy93cy8yMDA1LzA1L2lkZS50aXRSL2NsYWltcy9uVWllIjoiJ01jZWljbWVlIiwiaHR0cDovL3NjaGV0YXN0eG1sc29' \ -H 'Content-Type: application/json' \ -d '{ \"id\": \"6924ec02835ffdb0b206369b9\", \"title\": \"One Piece Film: Red\", \"director\": \"Gorō Taniguchi\", \"releaseYear\": 2022, \"genre\": \"Adventure\", \"rating\": 8.1, \"createdAt\": \"2025-11-24T23:36:34.323Z\", \"createdBy\": \"user10\" }'` . The 'Request URL' is `https://localhost:7177/api/MovieAPI/6924ec02835ffdb0b206369b9`. The 'Server response' shows a 204 status code with headers: `date: Mon, 24 Nov 2025 23:47:23 GMT` and `server: Kestrel`. The 'Responses' table shows a 200 OK status with no links.

Execute Clear

Responses

Curl

```
curl -X 'PUT' \
'https://localhost:7177/api/MovieAPI/6924ec02835ffdb0b206369b9' \
-H 'accept: */*' \
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy44bWxzZ2Fubm9yZy93cy8yMDA1LzA1L2lkZS50aXRSL2NsYWltcy9uVWllIjoiJ01jZWljbWVlIiwiaHR0cDovL3NjaGV0YXN0eG1sc29' \
-H 'Content-Type: application/json' \
-d '{
  "id": "6924ec02835ffdb0b206369b9",
  "title": "One Piece Film: Red",
  "director": "Gorō Taniguchi",
  "releaseYear": 2022,
  "genre": "Adventure",
  "rating": 8.1,
  "createdAt": "2025-11-24T23:36:34.323Z",
  "createdBy": "user10"
}'
```

Request URL

https://localhost:7177/api/MovieAPI/6924ec02835ffdb0b206369b9

Server response

Code Details

204

Response headers

```
date: Mon, 24 Nov 2025 23:47:23 GMT
server: Kestrel
```

Responses

Code	Description	Links
200	OK	No links

DELETE /api/MovieAPI/{id}

Figure 23 Confirmation message for updating information

The screenshot shows a REST client interface with a blue 'Execute' button and a 'Cancel' button. Below the buttons, the 'Responses' tab is active, displaying a successful GET request. The 'Curl' section shows the command: `curl -X 'GET' \ 'https://localhost:7177/api/MovieAPI' \ -H 'accept: text/plain' \ -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy44bWxzZ2Fubm9yZy93cy8yMDA1LzA1L2lkZS50aXRSL2NsYWltcy9uVWllIjoiJ01jZWljbWVlIiwiaHR0cDovL3NjaGV0YXN0eG1sc29'` . The 'Request URL' is `https://localhost:7177/api/MovieAPI`. The 'Server response' shows a 200 status code with a response body containing three movie objects: `{ \"id\": \"6924ec02835ffdb0b206369b9\", \"title\": \"One Piece Film: Red\", \"director\": \"Gorō Taniguchi\", \"releaseYear\": 2022, \"genre\": \"Adventure\", \"rating\": 8.1, \"createdAt\": \"2025-11-24T23:36:34.323Z\", \"createdBy\": \"user10\" }, { \"id\": \"6924ec5f835ffdb0b206369ba\", \"title\": \"One Piece Film: Red\", \"director\": \"Gorō Taniguchi\", \"releaseYear\": 2022, \"genre\": \"Adventure\", \"rating\": 7.3, \"createdAt\": \"2025-11-24T23:38:07.963Z\", \"createdBy\": \"user10\" }, { \"id\": \"6924eca6835ffdb0b206369bc\", \"title\": \"Demon Slayer: Mugen Train\" }`.

Responses

Curl

```
curl -X 'GET' \
'https://localhost:7177/api/MovieAPI' \
-H 'accept: text/plain' \
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy44bWxzZ2Fubm9yZy93cy8yMDA1LzA1L2lkZS50aXRSL2NsYWltcy9uVWllIjoiJ01jZWljbWVlIiwiaHR0cDovL3NjaGV0YXN0eG1sc29' 
```

Request URL

https://localhost:7177/api/MovieAPI

Server response

Code Details

200

Response body

```
{
  "id": "6924ec02835ffdb0b206369b9",
  "title": "One Piece Film: Red",
  "director": "Gorō Taniguchi",
  "releaseYear": 2022,
  "genre": "Adventure",
  "rating": 8.1,
  "createdAt": "2025-11-24T23:36:34.323Z",
  "createdBy": "user10"
},
{
  "id": "6924ec5f835ffdb0b206369ba",
  "title": "One Piece Film: Red",
  "director": "Gorō Taniguchi",
  "releaseYear": 2022,
  "genre": "Adventure",
  "rating": 7.3,
  "createdAt": "2025-11-24T23:38:07.963Z",
  "createdBy": "user10"
},
{
  "id": "6924eca6835ffdb0b206369bc",
  "title": "Demon Slayer: Mugen Train"
}
```

Figure 24 Verifying through GET movies name

9. Delete a movie DELETE /api/MovieAPI/: Click on DELETE => Try it out => enter id => execute.

The screenshot shows a REST client interface with a red 'DELETE' button and a 'Cancel' button. Below the buttons, the 'Parameters' tab is active, showing a required string parameter 'id' with the value `6924ec9a835ffdb0b206369bbt`. The 'Execute' button is visible. The 'Responses' tab is also visible, showing a table with columns 'Code', 'Description', and 'Links'.

DELETE /api/MovieAPI/{id}

Parameters

Cancel

Name	Description
id * required string (path)	6924ec9a835ffdb0b206369bbt

Execute

Responses

Code	Description	Links
------	-------------	-------

Figure 25 Enter ID of the movie

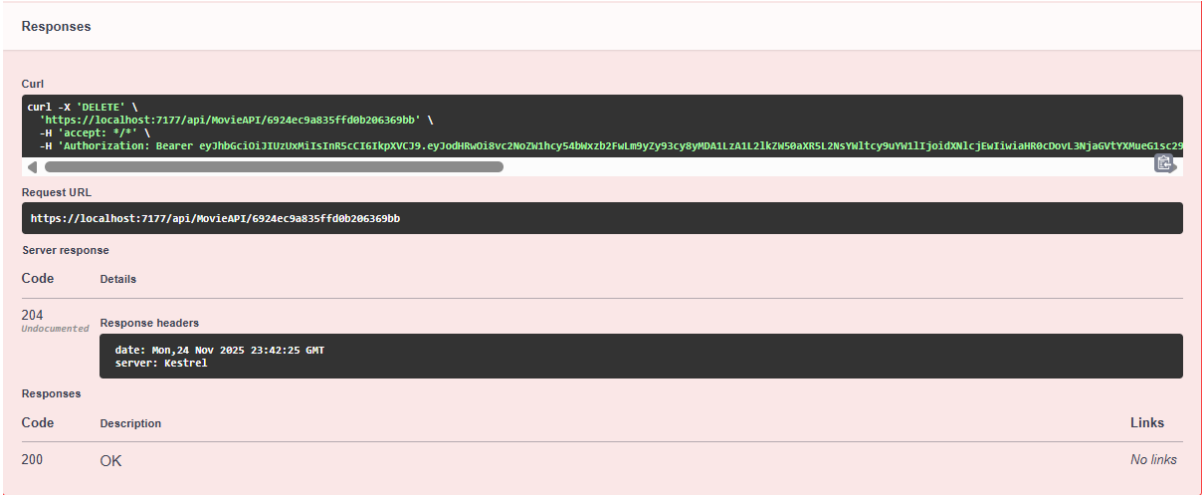


Figure 26 Message of Delete confirmation

Anime Movies

All Movies

Anime Movie list

Sort by:

Rating (High to Low)

Rating (Low to High)

Year (Newest First)

Year (Oldest First)

Title (A-Z)

Title (Z-A)

Reset

Add New Movie

Title	Director	Genre	Year	Rating	Actions
One Piece Film: Red	Gorō Taniguchi	Adventure	2022	8.1	<div>EditDelete</div>
One Piece Film: Red	Gorō Taniguchi	Adventure	2022	7.3	<div>EditDelete</div>
Demon Slayer: Mugen Train	Haruo Sotozaki	Action	2020	8.2	<div>EditDelete</div>
Spirited Away	Hayao Miyazaki	Fantasy	2001	8.6	<div>EditDelete</div>
Your Name	Makoto Shinkai	Romance	2016	8.2	<div>EditDelete</div>

© 2025 - Anime Movies

Figure 27 Movie/ Index page final output

References, credits or tools used:

- 1. <https://getbootstrap.com/docs/5.3/getting-started/introduction/> for styling the web application.
- 2. Pre-installed GitHub co-pilot in visual studio to help while coding and solving issues.