# Assignment: Build a Movie REST API

### Objective

Create a Spring Boot REST API for managing movies. The API should allow performing all CRUD operations (Create, Read, Update, Delete) and also support search features.

---

### Entity: Movie

Each movie will have the following fields:

\* movieId (Long)

\* movieName (String)

\* directorName (String)

\* budget (Double)

---

Create a movie REST endpoint with the following URI's

http://localhost:9090/movies

With the above URI, we should be able to perform all the CRUD operation on movies

Movies

movieId

movieName

directorName

budget

MoviesController

Test it via postman

http://localhost:9090/movies - should return all the movies

http://localhost:9090/movies/1911 - should return movie with movie id 1911

http://localhost:9090/movies/1911 -DELETE should delete movie with movie id 1911

http://localhost:9090/movies - POST - should be able to save a movie

http://localhost:9090/movies - PUT - should be able to update a movie

http://localhost:9090/movies/findMovie/Ironman - GET - should be able to search and return a movie

http://localhost:9090/movies/findMovie/29000/988000 - GET - should be able to search and return a movie based on budget range

### Tasks

1. \*\*Create a REST Controller → MoviesController\*\*

The controller should expose endpoints under the base URI:

```

http://localhost:9090/movies

```

2. \*\*Implement the following REST Endpoints\*\*

\* \*\*Fetch all movies\*\*

\* Method: GET

\* URL: `http://localhost:9090/movies`

\* Returns the list of all movies.

\* \*\*Fetch a movie by ID\*\*

\* Method: GET

\* URL: `http://localhost:9090/movies/{id}`

\* Example: `http://localhost:9090/movies/1911` → returns movie with id = 1911.

\* \*\*Delete a movie by ID\*\*

\* Method: DELETE

\* URL: `http://localhost:9090/movies/{id}`

\* Example: `http://localhost:9090/movies/1911` → deletes movie with id = 1911.

\* \*\*Save a new movie\*\*

\* Method: POST

\* URL: `http://localhost:9090/movies`

\* Request body (JSON):

```json

{

"movieId": 1911,

"movieName": "Ironman",

"directorName": "Jon Favreau",

"budget": 140000000

}

```

\* \*\*Update an existing movie\*\*

\* Method: PUT

\* URL: `http://localhost:9090/movies`

\* Request body should contain updated movie details.

\* \*\*Find movie by name\*\*

\* Method: GET

\* URL: `http://localhost:9090/movies/findMovie/{movieName}`

\* Example: `http://localhost:9090/movies/findMovie/Ironman`

\* \*\*Find movies by budget range\*\*

\* Method: GET

\* URL: `http://localhost:9090/movies/findMovie/{min}/{max}`

\* Example: `http://localhost:9090/movies/findMovie/29000/988000`

---

### Deliverables

1. A Spring Boot project with `MoviesController`.

2. A simple Movie model class.

3. Use either \*\*HashMap (in-memory DB)\*\* or \*\*Spring Data JPA with H2/MySQL\*\*.

4. Test all endpoints using Postman.

---

### Expected Outcome

\* Participants should be able to design REST endpoints.

\* Participants should understand CRUD operations in REST.

\* Participants will gain practice with Postman API testing.