**API Documentation for movie API**

**Base URL** http://localhost:3000

### **Endpoints**

#### **1. GET /movies**

* **Description**: Fetches all movies from the database.
* **Query Parameters**:
  + q: (optional) A search query to filter movies by title or genre.
* **Responses**:
  + **200 OK**:
    - **Description**: A list of movies.

**Response Body**:  
 {

"status": "success",

"results": 2,

"data": [

{

"id": "60a7fefbba3e4b0ad6e9d798",

"title": "Movie 1",

"genre": "Action",

"rating": 5,

"streamingLink": "https://example.com/movie1"

},

{

"id": "60a7fefbba3e4b0ad6e9d799",

"title": "Movie 2",

"genre": "Comedy",

"rating": 4,

"streamingLink": "https://example.com/movie2"

}

]

}

#### **2. POST /movies**

* **Description**: Creates a new movie entry.

**Request Body**:  
 {

"title": "New Movie",

"genre": "Action",

"rating": 5,

"streamingLink": "https://example.com/new-movie"

}

* **Responses**:
  + **201 Created**:
    - **Description**: The movie has been successfully created.

**Response Body**:  
 {

"status": "success",

"data": {

"title": "New Movie",

"genre": "Action",

"rating": 5,

"streamingLink": "https://example.com/new-movie"

}

}

#### **3. PUT /movies/{id}**

* **Description**: Updates a movie by its ID.
* **URL Parameters**:
  + id: The ID of the movie to update.

**Request Body**:  
 {

"title": "Updated Movie",

"genre": "Action",

"rating": 4,

"streamingLink": "https://example.com/updated-movie"

}

* **Responses**:
  + **200 OK**:
    - **Description**: The movie was updated successfully.

**Response Body**:  
 {

"status": "success",

"data": {

"title": "Updated Movie",

"genre": "Action",

"rating": 4,

"streamingLink": "https://example.com/updated-movie"

}

}

#### **4. DELETE /movies/{id}**

* **Description**: Deletes a movie by its ID.
* **URL Parameters**:
  + id: The ID of the movie to delete.
* **Responses**:
  + **204 No Content**:
    - **Description**: The movie was successfully deleted.
  + **404 Not Found**:
    - **Description**: Movie not found for the provided ID.

**Response Body**:  
 {

"status": "error",

"message": "No movie found with that ID"

}

### 

### 

### **Test Cases Covered**

#### 1. GET /movies

### should return all movies when no query is provided

### Verifies that all movies are fetched when no query parameter is passed in the request.

### should return filtered movies when query is provided

### Verifies that only movies matching the query are returned when a search query is passed in the request.

#### 2. POST /movies

### should create a new movie

### Verifies that a new movie is successfully created when the required fields are passed in the request body.

### should return error if not admin

### Verifies that the API returns a 401 Unauthorized error when the user does not have the necessary 'admin' role.

#### 3. PUT /movies/:id

### should update a movie

### Verifies that a movie is updated successfully when the movie ID is valid and the request body contains valid movie data.

### should return error if movie not found

### Verifies that the API returns a 404 Not Found error if the movie ID provided in the request does not exist.

#### 4. DELETE /movies/:id

### should delete a movie

### Verifies that the movie is successfully deleted when the movie ID exists and the request is valid.

### should return error if movie not found for deletion

### Verifies that the API returns a 404 Not Found error when attempting to delete a movie that does not exist.

### should return error if not admin

### Verifies that the API returns a 401 Unauthorized error when the user does not have the necessary 'admin' role to perform the deletion.

### 

### 