MovieDatabase

This Postman collection tests all award-related API endpoints, including creation, retrieval, editing, and deletion. It validates request formats, response codes, and authentication flows to ensure robust backend functionality and accurate data handling across the movie review platform.EndFragment

---

GET Login

http://localhost:5000/api/v1.0/login

User Login Endpoint

Overview

This endpoint allows users to authenticate and obtain a JWT (JSON Web Token) for accessing protected routes in the MovieDatabase API.

Request Details

- **Method**: GET
- **Endpoint**: `/api/v1.0/login`
- **Authentication**: No additional authentication required for this request

Response

- **Success Response**:
  - **Status Code**: 200 OK
  - **Response Body**:

```json
{
  "token": "JWT_TOKEN_STRING"
}
```

Post-Request Behavior

- Upon successful login, the JWT token is automatically stored in a global Postman variable `jwt_token`
- This token can be used for subsequent authenticated requests to the MovieDatabase API

Notes

- Ensure you have valid credentials before making this request
- The token is typically short-lived and may expire after a certain period

AUTHORIZATION Basic Auth

Username	<username>
Password	<password>

GET (FAIL) Login

http://localhost:5000/api/v1.0/login?

Login Endpoint

This endpoint authenticates a user and returns an access token for subsequent API requests.

Purpose

Validates user credentials and generates a JWT (JSON Web Token) that must be included in the `x-access-token` header for protected endpoints.

Expected Behavior

- **Success (200 OK)**: Returns a valid JWT token that should be stored and used for authenticated requests
- **Failure (401 Unauthorized)**: Invalid credentials or missing authentication information

Authentication

This is a public endpoint that does not require an existing token. However, it expects user credentials to be provided (typically via request body or query parameters).

Usage

1. Send a request with valid user credentials
2. Store the returned JWT token
3. Include the token in the `x-access-token` header for all subsequent authenticated requests

Response Format

- **Success**: JSON object containing the authentication token
- **Error**: JSON object with error message and status code

Notes

- The token has an expiration time and will need to be refreshed periodically
- Failed login attempts may be rate-limited for security purposes
- This is a test request configured to expect a 401 status (authentication failure scenario)

HEADERS

x-access-token	none

PARAMS

POST **Add New Movie**

http://localhost:5000/api/v1.0/movies

Add New Movie Endpoint

Purpose

This endpoint allows users to add a new movie to the movie database.

Request Details

- **HTTP Method**: POST
- **Endpoint**: `/api/v1.0/movies`
- **Authentication**: Requires a valid JWT token (x-access-token header)

Request Body (x-www-form-urlencoded)

Parameter	Type	Required	Description
Title	String	Yes	The title of the movie
Year	Integer	Yes	The release year of the movie
Runtime	String	Yes	Movie duration (e.g., '100 min')
Genre	String	Yes	Movie genre
Director	String	Yes	Name of the movie's director
Actors	String	Yes	Main actors in the movie
Plot	String	Yes	Brief description of the movie
IMDb Rating	Float	Yes	IMDb rating of the movie

Expected Response

- **Status Code**: 201 Created
- **Response Body**: JSON object containing the newly created movie's details and URL

Example

Plain Text

```
Title: Test Movie
Year: 2025
Runtime: 100 min
Genre: Comedy
Director: Declan Matthews
Actors: John Smyth
Plot: Test plot, test plot
IMDb Rating: 9
```

Notes

- Ensure you have a valid JWT token before making this request
- All fields are required
- The response will include a URL to the newly created movie resource

HEADERS

x-access-token	{{jwt_token}} JWT Token from global variables

Body urlencoded

Title	Test Movie A character string representing the Title of the movie
Year	2025 An integer representing the year the movie was released
Runtime	100 min A character string representing the length of the movie
Genre	Comedy A character string representing the Genre of the movie
Director	Declan Matthews A character string representing the Director/Directors of the movie
Actors	John Smyth

	A character string representing the Actors in the movie
Plot	Test plot, test plot A character string representing the plot of the movie
IMDb Rating	9 An integer representing the rating on IMDb

POST (FAIL) Add New Movie

http://localhost:5000/api/v1.0/movies

Add New Movie Endpoint

Purpose

This endpoint allows adding a new movie to the movie database.

Request Parameters

The request uses x-www-form-urlencoded body with the following fields:

- **Title** (required): Name of the movie
- **Year** (required): Year of release
- **Runtime** (optional): Movie duration in minutes
- **Genre** (optional): Movie genre
- **Director** (optional): Name of the movie director
- **Actors** (optional): List of main actors

Authentication

Requires a valid JWT token passed in the 'x-access-token' header.

Possible Responses

- **201 Created**: Movie successfully added
- **400 Bad Request**: Invalid or missing required parameters
- **401 Unauthorized**: Invalid or missing authentication token
- **404 Not Found**: Endpoint not found (as shown in the current test script)

Example Request Body

Plain Text

```
Title=The Simpsons Movie
```

```
Year=2007
Runtime=87

Genre=Comedy
Director=David Silverman
```

Notes

- Ensure all required fields are provided
- Token must be obtained via login endpoint before making this request

AUTHORIZATION Basic Auth

Username	<username>
Password	<password>

HEADERS

x-access-token	{{jwt_token}}

Body urlencoded

missing data	x
data missing	y

POST Add Movie Review

{{new_movie_url}}/reviews

Add Movie Review

Purpose

This endpoint allows authenticated users to submit a review for a specific movie. Users can provide a rating (in stars) and a written comment about their viewing experience.

Authentication

Requires a valid JWT token in the `x-access-token` header. Users must be logged in to submit reviews.

Request Parameters

Headers

- `x-access-token` (required): JWT authentication token obtained from the login endpoint

Body Parameters (URL-encoded)

- `username` (required, string): The name of the user submitting the review
- `comment` (required, string): The user's written review or feedback about the movie
- `stars` (required, integer): Rating given to the movie on a scale (typically 1-5 stars)

Path Variables

The movie ID is included in the URL path via the `{{new_movie_url}}` variable, which should contain the full URL to the specific movie resource.

Expected Response

Success (201 Created)

Returns the newly created review resource with:

- Review details (username, comment, stars)
- URL to the created review resource
- Timestamp of creation
- Associated movie information

Error Responses

- **400 Bad Request**: Missing required fields or invalid data format
- **401 Unauthorized**: Missing or invalid authentication token
- **404 Not Found**: Movie resource does not exist
- **500 Internal Server Error**: Server-side error during review creation

Example Usage

1. Ensure you have a valid JWT token from the login endpoint
2. Set the `{{new_movie_url}}` variable to point to a valid movie resource
3. Provide username, comment, and star rating in the request body
4. The response will include a `url` field that is stored in the `{{new_review_url}}` global variable for subsequent requests

HEADERS

x-access-token	{{jwt_token}}

Body urlencoded

username	Test User

comment	This is a test comment
stars	3

POST (FAIL) Add Movie Review

{{new_movie_url}}/reviews

Purpose

This request is designed to test the API's error handling when attempting to add a movie review with missing required data. It intentionally sends incomplete data to validate that the server returns a 500 Internal Server Error.

Expected Input

- **Endpoint**: `{{new_movie_url}}/reviews`
- **Method**: POST
- **Authentication**: JWT token required in `x-access-token` header
- **Body Type**: URL-encoded form data
- **Required Fields**: The request intentionally sends incomplete data with a placeholder key "x" to trigger an error response

Expected Response

- **Status Code**: 500 Internal Server Error
- **Purpose**: Validates that the API properly handles requests with missing or invalid review data

Test Validation

The post-response script verifies that the status code is 500, confirming the API's error handling behavior for malformed review submissions.

Notes

This is a negative test case. For successful review creation, ensure all required fields (such as movie ID, rating, review text, etc.) are properly included in the request body.

HEADERS

x-access-token	{{jwt_token}}

Body urlencoded

missing data	x

POST **Add Award**

http://localhost:5000/api/v1.0/awards/add

Add Award Endpoint

Purpose

This endpoint allows adding a new award to the movie database.

Request Details

- **Method**: POST
- **Endpoint**: `/api/v1.0/awards/add`
- **Content-Type**: application/json

Request Body

Requires a JSON object with the following properties:

- `title` (string): The title of the movie associated with the award
- `award_name` (string): The name of the award
- `category` (string): The award category
- `year` (number): The year the award was given
- `won` (boolean): Whether the movie won the award

Example Request Body

json

```
{
  "title": "The Lion King",
  "award_name": "Postman Test Award",
  "category": "Postman Test Category",
  "year": 2019,
  "won": true
}
```

Expected Response

- **Status Code**: 201 Created
- Returns the URL of the newly created award resource

Notes

- Ensure all fields are provided

- The year should be a valid year
- The 'won' field indicates whether the movie won the specific award

HEADERS

Content-Type	application/json

Body raw (json)

```json
{
  "title": "The Lion King",
  "award_name": "Postman Test Award",
  "category": "Postman Test Category",
  "year": 2019,
  "won": true
}
```

PUT **Edit Movie**

{{new_movie_url}}

Edit Movie Endpoint

Purpose

This endpoint allows updating an existing movie's details in the movie database.

Request Details

- **HTTP Method**: PUT
- **Endpoint**: `{{new_movie_url}}`
- **Authentication**: Requires a valid JWT token (x-access-token)

Request Body

Provide the updated movie information using the following fields:

- `Title`: Name of the movie
- `Year`: Release year of the movie
- `IMDb Rating`: Movie's rating on IMDb
- `Runtime`: Movie's duration in minutes
- `Genre`: Movie's genre

- `Director`: Name of the movie's director
- `Actors`: Main actors in the movie

- `Plot`: Brief description of the movie

Authentication

- Ensure a valid JWT token is provided in the `x-access-token` header
- Token must have sufficient permissions to edit movie records

Expected Response

- **Success**: 200 OK status with updated movie details
- **Failure**: Appropriate error status with error message

Notes

- All fields are required
- Ensure all data is accurate before submission

HEADERS

x-access-token	{{jwt_token}}

Body urlencoded

Title	This is a second test movie
Year	1999
IMDb Rating	4
Runtime	160 min
Genre	Action
Director	Matthew Declans
Actors	Johnnie Smyth
Plot	Test plot test plot

PUT (FAIL) Edit Movie

{{new_movie_url}}

Edit Movie

This request updates an existing movie's information in the database.

Authentication

Requires a valid JWT token in the `x-access-token` header. Use the login endpoint to obtain a token first.

Request Details

- **Method**: PUT
- **URL**: `{{new_movie_url}}` - The URL should point to the specific movie resource (typically includes the movie ID)

Request Body (x-www-form-urlencoded)

The following fields can be updated:

- `Title` - The movie title
- `Year` - Release year
- `IMDb Rating` - Rating value
- `Runtime` - Duration (e.g., "160 min")
- `Actors` - Comma-separated list of actor names
- `Plot` - Movie plot summary

Expected Response

- **Success (200)**: Returns the updated movie object
- **Not Found (404)**: Movie with the specified ID does not exist
- **Unauthorized (401)**: Invalid or missing authentication token

Notes

- Only authenticated users can edit movies
- The movie ID must be included in the URL
- All fields are optional - only provided fields will be updated

HEADERS

x-access-token	{{jwt_token}}

Body urlencoded

Title	This is a second test movie
Year	1999
IMDb Rating	4
Runtime	160 min

Actors	Johnnie Smyth
Plot	Test plot test plot

PUT Edit Award

{{new_award_url}}

Edit Award Endpoint

Purpose

This endpoint allows authenticated users to update an existing award's information in the database.

Request Details

- **HTTP Method**: PUT
- **Endpoint**: Award URL (stored in `{{new_award_url}}` variable)
- **Authentication**: Requires JWT token (x-access-token header)

Headers

Header	Value	Required	Description
x-access-token	{{jwt_token}}	Yes	JWT token for authentication

Request Body (x-www-form-urlencoded)

Parameter	Type	Required	Description
award_name	String	Yes	The name of the award
category	String	Yes	The award category
year	Integer	Yes	The year the award was given
won	Boolean	Yes	Whether the award was won (true/false)
movie_id	String	Yes	The MongoDB ID of the associated movie

Expected Response

- **Status Code**: 200 OK
- **Response Body**: JSON object containing the updated award details

Example

Plain Text

```
PUT {{new_award_url}}
Headers: x-access-token: {{jwt_token}}
Body:
  award_name: Postman Test Award
  category: Postman Test Category 2
  year: 2019
  won: true
  movie_id: 69026399a9f15b2a209c60cd
```

Notes

- Requires authentication
- All fields must be provided
- The award URL is typically obtained from the Add Award response
- Returns 404 if award not found

HEADERS

x-access-token	{{jwt_token}}

Body urlencoded

award_name	Postman Test Award
category	Postman Test Catergory 2
year	2019
won	true
movie_id	69026399a9f15b2a209c60cd

GET Logout

http://localhost:5000/api/v1.0/logout

Logout

This endpoint logs out the currently authenticated user by invalidating their JWT access token.

Purpose

Terminates the user's active session and ensures their access token can no longer be used for authenticated requests.

Authentication

Requires a valid JWT token to be provided in the `x-access-token` header.

Expected Behavior

- **Success (200)**: The user's token is invalidated and they are successfully logged out
- **Unauthorized (401)**: Invalid or missing token
- **Error (500)**: Server error during logout process

Important Details

- After logout, the JWT token becomes invalid and cannot be reused
- Users must log in again to obtain a new token for subsequent authenticated requests
- This is a standard user logout - for admin logout, use the Admin Logout endpoint
- The token variable `{{jwt_token}}` should be cleared after successful logout

Usage

Call this endpoint when a user wants to end their session or before switching accounts.

HEADERS

x-access-token	{{jwt_token}}

GET (FAIL) Logout

http://localhost:5000/api/v1.0/logout

Purpose

This endpoint logs out the currently authenticated user by invalidating their session token.

Authentication Required

- **Header**: `x-access-token`
- **Value**: Valid JWT token obtained from the login endpoint
- **Required**: Yes

Expected Behavior

- **Success (200)**: User is successfully logged out and the token is invalidated
- **Unauthorized (401)**: Missing or invalid authentication token

Usage Notes

- After logout, the token can no longer be used for authenticated requests
- The user must log in again to obtain a new valid token
- This is a standard logout operation that should be called when a user session ends

Test Validation

This request includes a test that validates a 401 status code response, which appears to be testing the failure case when no valid token is provided.

GET Get All Movies

http://localhost:5000/api/v1.0/movies

Get Movie Genre Count Endpoint

Overview

This endpoint retrieves the count of movies for each genre in the movie database.

Request Details

- **Method**: GET
- **Endpoint**: `/api/v1.0/movies/genre-count`

Response

Returns a JSON object where:

- Keys represent movie genres
- Values represent the number of movies in each genre

Example Response

```json
{
  "Action": 25,
  "Drama": 40,
  "Comedy": 30,
  "Sci-Fi": 15
}
```

Use Cases

- Generating genre distribution charts
- Analyzing movie catalog composition
- Providing genre statistics for reporting

GET (FAIL) Get All Movies

http://localhost:5000/api/v1.0/

Root Endpoint Request

Overview

This is a GET request to the root endpoint of the Movie Database API. It is used to check the basic connectivity and availability of the API.

availability of the API.

Expected Behavior

Currently configured to expect a 404 Not Found status code, which might indicate that the root endpoint does not have a specific implementation or response.

Potential Use Cases

- API health check
- Verifying server connectivity
- Debugging initial API interactions

Notes

- The request is made to http://localhost:5000/api/v1.0/
- A test script is in place to verify the 404 status code

GET Get One Movie

http://localhost:5000/api/v1.0/movies/69026399a9f15b2a209c607f

Get One Movie Endpoint

Description

Retrieves detailed information for a specific movie by its unique identifier.

Endpoint

`GET /api/v1.0/movies/{movieId}`

Parameters

- `movieId` (path parameter): A unique identifier for the specific movie to retrieve

Expected Response

- Status Code: 200 OK
- Response Body: JSON object containing movie details
  - Includes properties such as title, year, genre, director, actors, etc.

Example Use Case

Fetch comprehensive information about a specific movie when you know its unique identifier.

Potential Errors

- 404 Not Found: If the movie ID does not exist in the database

---

GET (FAIL) Get One Movie

http://localhost:5000/api/v1.0/movies/

Get One Movie Endpoint

Description

Retrieves details of a specific movie by its unique identifier.

Endpoint

`GET /api/v1.0/movies/`

Expected Behavior

- Returns detailed information about a single movie
- Requires a valid movie ID in the request

Possible Responses

- **404 Not Found**:
  - Occurs when no movie matches the provided identifier
  - Indicates the requested movie does not exist in the database

Notes

- Ensure a valid movie ID is provided
- Check authentication requirements if applicable
- Verify the correct endpoint URL before making the request

GET Get All Reviews

http://localhost:5000/api/v1.0/movies/69026399a9f15b2a209c60cd/reviews

Get All Reviews Endpoint

Overview

This endpoint retrieves all movie reviews from the database.

Request Details

- **Method**: GET
- **Endpoint**: `/api/v1.0/movies/{movieId}/reviews`
- **Base URL**: `http://localhost:5000`

Parameters

- `movieId` (path parameter): The unique identifier of the movie for which reviews are being retrieved

Response

- **Status Code**: 200 OK
- **Content Type**: JSON
- **Response Body**: An array of review objects, each containing details about a review

Example Response

```json
[
  {
    "_id": "reviewId",
    "movieId": "movieId",
    "reviewer": "Reviewer Name",
    "rating": 4.5,
    "comment": "Great movie!"
  }
]
```

Possible Errors

- 404 Not Found: If the specified movie ID does not exist

GET Admin Login

http://localhost:5000/api/v1.0/login

Admin Login

This endpoint authenticates an administrator user and returns an access token for subsequent API requests.

Purpose

Allows administrators to log in to the MovieDatabase system and obtain a JWT token for accessing protected admin endpoints.

Authentication

No authentication required for this endpoint.

Request

- **Method**: GET
- **URL**: `http://localhost:5000/api/v1.0/login`
- **Headers**: None required
- **Body**: None

Expected Response

Success (200 OK)

```json
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

The response contains a JWT token that should be stored and included in the `x-access-token` header for all subsequent requests that require admin authentication.

Usage

1. Send a GET request to this endpoint
2. Store the returned token from the response
3. Include the token in the `x-access-token` header for authenticated requests

Notes

- The token is automatically saved to the global variable `admin_token` via the post-response script
- This token is required for admin operations such as adding, editing, or deleting movies and awards
- Token expiration and refresh policies should be handled according to your API's authentication strategy

AUTHORIZATION Basic Auth

Username	<username>
Password	<password>

GET (FAIL) Admin Login

http://localhost:5000/api/v1.0/login

Admin Login

This endpoint authenticates an administrator user and returns an access token for subsequent API requests.

Purpose

Validates admin credentials and generates a JWT token that must be included in the `x-access-token` header for all protected admin endpoints.

Expected Input

Method: GET
URL: `http://localhost:5000/api/v1.0/login`

Headers:

- `x-access-token`: Set to `"none"` for the initial login request

Authentication: Admin credentials should be provided via Basic Auth or as specified by the API implementation.

Responses

Success (200 OK)

Returns when authentication is successful.

```json
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "message": "Login successful"
}
```

Unauthorized (401)

Returns when:

- Invalid credentials are provided
- Missing authentication headers
- Account does not have admin privileges

```json
{
  "error": "Invalid credentials",
  "message": "Authentication failed"
}
```

Usage Notes

- Store the returned token securely
- Include the token in the `x-access-token` header for all subsequent admin requests
- Tokens may expire after a certain period and require re-authentication
- This endpoint is specifically for admin users; regular user login may use a different endpoint

DELETE Delete Movie

{{new_movie_url}}

Delete Movie Endpoint

Overview

This endpoint allows administrators to delete a specific movie from the movie database.

Authentication

- Requires an admin access token (`x-access-token`)
- Only authenticated admin users can delete movies

Request Details

- **Method**: DELETE
- **URL**: `{{new_movie_url}}`
- **Headers**:
  - `x-access-token` : Admin authentication token

Expected Behavior

- Successful deletion returns a 204 No Content status
- Attempting to delete a non-existent movie may result in an error
- Requires proper authorization to execute

Potential Error Scenarios

- 401 Unauthorized: Invalid or expired admin token
- 403 Forbidden: Insufficient permissions
- 404 Not Found: Movie does not exist

Usage Example

To delete a movie, ensure you have a valid admin token and the correct movie URL.

HEADERS

x-access-token	{{admin_token}}

DELETE Delete Award

{{new_award_url}}

Delete Award Endpoint

Purpose

This endpoint allows authenticated users to delete an award from the database.

Request Details

- **HTTP Method**: DELETE
- **Endpoint**: Award URL (stored in `{{new_award_url}}` variable)
- **Authentication**: Requires admin JWT token (x-access-token header)

Headers

Header	Value	Required	Description
x-access-token	{{admin_token}}	Yes	Admin JWT token for authentication

Expected Response

- **Status Code**: 204 No Content
- **Response Body**: Empty (successful deletion)

Example

Plain Text

```
DELETE {{new_award_url}}
Headers: x-access-token: {{admin_token}}
```

Notes

- Requires admin authentication
- The award URL is typically obtained from the Add Award response
- Returns 204 on successful deletion
- Returns 404 if award not found
- Returns 401/403 if not authenticated or not authorized

HEADERS

x-access-token {{admin_token}}

GET Admin Logout

http://localhost:5000/api/v1.0/logout

Admin Logout Endpoint

Purpose

This endpoint allows administrators to log out and invalidate their authentication token.

Request Details

- **HTTP Method**: GET
- **Endpoint**: `/api/v1.0/logout`
- **Authentication**: Requires admin JWT token (x-access-token header) and Basic Auth

Headers

Header	Value	Required	Description
x-access-token	{{admin_token}}	Yes	Admin JWT token for authentication

Authentication

- **Type**: Basic Auth
- **Username**: dmatthews959
- **Password**: adminpass

Expected Response

- **Status Code**: 200 OK
- **Response Body**: Confirmation message of successful logout

Example

Plain Text

```
GET /api/v1.0/logout
Headers: x-access-token: {{admin_token}}
Auth: Basic (username: dmatthews959, password: adminpass)
```

Notes

- Requires both JWT token and Basic Auth credentials
- Invalidates the current admin session
- After logout, the token should no longer be valid for authenticated requests

AUTHORIZATION Basic Auth

Username	<username>
Password	<password>

HEADERS

x-access-token	{{admin_token}}

GET (FAIL) Admin Logout

http://localhost:5000/api/v1.0/logout

Admin Logout Endpoint

Purpose

This endpoint is used to log out an administrative user from the system.

Request Details

- **HTTP Method**: GET
- **Endpoint**: `/api/v1.0/logout`

Authentication

Requires a valid admin token to be present in the request headers.

Expected Behavior

- Successfully logging out will invalidate the current admin session
- Returns a 401 Unauthorized status if the logout is unsuccessful or no valid token is provided

Possible Scenarios

- Successful logout: User is logged out and session is terminated
- Failed logout: Invalid or expired token results in a 401 error

Notes

- Ensure the `x-access-token` header is set with a valid admin token before making this request
- After logout, the admin will need to log in again to access protected endpoints

GET Get Reviews By Stars

GET Get Reviews By Stars

http://localhost:5000/api/v1.0/movies/reviews/stars?stars=5

Get Reviews by Star Rating

Overview

This endpoint retrieves movie reviews filtered by their star rating.

Request Parameters

- `stars` (query parameter, required): The number of stars to filter reviews by
  - Type: Integer
  - Possible values: 1-5

Authentication

- Requires a valid JWT token passed in the `x-access-token` header

Response

- Status Code: 200 OK
- Returns a list of movie reviews with the specified star rating

Example

- Request: `GET /api/v1.0/movies/reviews/stars?stars=5`
- This will return all movie reviews that have a 5-star rating

Potential Errors

- 401 Unauthorized: If the JWT token is invalid or missing
- 404 Not Found: If no reviews match the specified star rating

HEADERS

x-access-token	{{jwt_token}}

PARAMS

stars	5

GET (FAIL) Get Reviews By Stars

http://localhost:5000/api/v1.0/movies/reviews/stars

Get Movie Reviews by Star Rating

Overview

This endpoint retrieves movie reviews filtered by a specific star rating.

Request Details

- **Method:** GET
- **Endpoint:** `/api/v1.0/movies/reviews/stars`

Parameters

- `stars` (Header): The number of stars to filter reviews by

Expected Responses

- **404 Not Found:** No reviews found for the specified star rating

Notes

- Ensure the star rating is a valid input
- This request currently has a test expecting a 404 status code

Potential Use Cases

- Filtering reviews by quality
- Analyzing movie ratings
- Generating rating-based reports

HEADERS

stars

GET Search Movie and Filter Stars

http://localhost:5000/api/v1.0/movies/69026399a9f15b2a209c60cd/reviews/stars?stars=5

Search Movies by Star Rating

Overview

This endpoint allows users to filter movies based on their star ratings.

Request Details

- **Method**: GET
- **Endpoint**: `/api/v1.0/movies/69026399a9f15b2a209c60cd/reviews/stars`

Query Parameters

- `stars` (required): Number of stars to filter reviews by
  - Type: Integer
  - Possible Values: 1-5
  - Example: `?stars=5` retrieves all reviews with a 5-star rating

Expected Response

- **Status Code**: 200 OK
- **Response Body**: Array of movie reviews matching the specified star rating
- **Response Content**:
  - Review details including star rating, review text, reviewer information

Use Cases

- Find highly rated movies
- Analyze user satisfaction for a specific movie
- Filter reviews based on quality ratings

Example

Request: `GET /api/v1.0/movies/69026399a9f15b2a209c60cd/reviews/stars?stars=5` Response: Returns all 5-star reviews for the specified movie

PARAMS

stars	5

GET (FAIL) Search Movie and Filter Stars

http://localhost:5000/api/v1.0/movies/69026399a9f15b2a209c60cd/reviews/stars?stars=

Get Reviews by Star Rating

This endpoint retrieves all reviews for a specific movie filtered by star rating

This endpoint retrieves all reviews for a specific movie filtered by star rating.

Path Parameters

- `movieId` (string, required) - The unique identifier of the movie whose reviews you want to retrieve

Query Parameters

- `stars` (integer, required) - Filter reviews by star rating (typically 1-5)
  - Example: `?stars=4` returns all 4-star reviews for the movie

Expected Responses

Success (200 OK)

json

```
[
  {
    "reviewId": "string",
    "movieId": "string",
    "userId": "string",
    "rating": 4,
    "comment": "string",
    "createdAt": "timestamp"
  }
]
```

Bad Request (400)

- Missing or invalid `stars` parameter
- Invalid `movieId` format

Not Found (404)

- Movie with the specified ID does not exist

Usage Notes

- The `stars` query parameter must be provided with a valid integer value
- Returns an empty array if no reviews match the specified star rating
- Useful for analyzing movie ratings distribution or displaying filtered reviews to users

PARAMS

stars

GET Search Movie Title

http://localhost:5000/api/v1.0/movies/title?Title=The Lion King

Search Movies by Title

Endpoint Description

Retrieves movies that match the specified title in the database.

Request Parameters

- `Title` (query parameter, required): The title of the movie to search for
  - Type: String
  - Example: 'The Lion King'

Response

- Returns an array of movie objects matching the title
- Each movie object contains details such as title, year, genre, etc.

Example

- Request: `GET /api/v1.0/movies/title?Title=The Lion King`
- Successful Response: 200 OK
  - Returns matching movie(s) with title containing 'The Lion King'

Notes

- Search is case-insensitive
- Partial matches are supported
- Returns an empty array if no movies match the title

PARAMS

Title	The Lion King

GET (FAIL) Search Movie Title

http://localhost:5000/api/v1.0/movies/title?Title=

Search Movies by Title

Description

This endpoint allows searching for movies by their title in the movie database.

Endpoint Details

- **Method**: GET
- **URL**: `http://localhost:5000/api/v1.0/movies/title`

Query Parameters

- `Title` (required): The title or partial title of the movie to search for
  - Type: String
  - Example: `?Title=Inception`

Expected Responses

- **200 OK**: Returns a list of movies matching the title
- **400 Bad Request**: Occurs when no title is provided or search parameters are invalid

Usage Notes

- Partial matches are supported
- Search is case-insensitive
- Ensure the Title parameter is not empty when making the request

PARAMS

Title

GET Search by Director

http://localhost:5000/api/v1.0/movies/director?Director=Steven Spielberg

Search by Director Endpoint

Purpose

This endpoint allows users to search for movies by director name, returning all movies directed by the specified person.

Request Details

- **HTTP Method**: GET
- **Endpoint**: `/api/v1.0/movies/director`
- **Authentication**: Not required

Query Parameters

Parameter	Type	Required	Description
Director	String	Yes	The director's name to search for (e.g., "Steven Spielberg")

Expected Response

- **Status Code**: 200 OK
- **Response Body**: JSON array containing all movies directed by the specified director

Example

Plain Text

```
GET /api/v1.0/movies/director?Director=Steven Spielberg
```

Notes

- The search is case-sensitive
- Returns an empty array if no movies match the director
- Multiple movies by the same director will be returned

PARAMS

Director	Steven Spielberg

GET Search Movie by Year

http://localhost:5000/api/v1.0/movies/searchyear?Year=1999

Search Movie by Year Endpoint

Purpose

This endpoint allows users to search for movies released in a specific year.

Request Details

- **HTTP Method**: GET
- **Endpoint**: `/api/v1.0/movies/searchyear`
- **Authentication**: Not required

Query Parameters

Parameter	Type	Required	Description
Year	Integer	Yes	The release year to search for (e.g., 1999, 2020)

Expected Response

- **Status Code**: 200 OK
- **Response Body**: JSON array containing all movies released in the specified year

Example

Plain Text

```
GET /api/v1.0/movies/searchyear?Year=1999
```

Notes

- Year must be a valid integer
- Returns an empty array if no movies match the year
- Multiple movies from the same year will be returned

PARAMS

Year	1999

GET (FAIL) Search Movie by Year

http://localhost:5000/api/v1.0/movies/searchyear?Year=

Search Movies by Year Endpoint

Overview

This endpoint allows searching for movies released in a specific year.

Request Parameters

- `Year` (Query Parameter, Required):
  - The specific year to search movies for
  - Example: `?Year=2020`

Expected Responses

- **200 OK**: Returns a list of movies released in the specified year
- **404 Not Found**: No movies found for the given year
    - Current test script expects a 404 status code

Usage Notes

- Ensure the year is a valid four-digit year
- The endpoint is case-sensitive
- Empty or invalid year parameters may result in an error

Example

GET /api/v1.0/movies/searchyear?Year=2019

PARAMS

Year

GET Search Movie by Year Range

http://localhost:5000/api/v1.0/movies/year-range?start=1990&end=2000

Search Movies by Year Range Endpoint

Description

This endpoint allows users to retrieve a list of movies released within a specified year range.

Request Parameters

- `start` (required): The starting year of the movie release range
- `end` (required): The ending year of the movie release range

Example Usage

- Retrieve movies released between 1990 and 2000
    - GET /api/v1.0/movies/year-range?start=1990&end=2000

Response

Returns an array of movie objects containing details of movies released within the specified year range.

Possible Response Codes

- 200 OK: Successfully retrieved movies
- 400 Bad Request: Invalid year range parameters

PARAMS

start	1990
end	2000

GET Search Movie by Genre

http://localhost:5000/api/v1.0/movies/genre?Genre=Comedy

Search Movie by Genre Endpoint

Purpose

This endpoint allows users to search for movies by their genre, returning all movies that match the specified genre.

Request Details

- **HTTP Method**: GET
- **Endpoint**: `/api/v1.0/movies/genre`
- **Authentication**: Not required

Query Parameters

Parameter	Type	Required	Description
Genre	String	Yes	The genre to search for (e.g., Comedy, Drama, Action)

Expected Response

- **Status Code**: 200 OK
- **Response Body**: JSON array containing all movies matching the specified genre

Example

Plain Text

```
GET /api/v1.0/movies/genre?Genre=Comedy
```

Notes

- The search is case-sensitive
- Returns an empty array if no movies match the genre
- Multiple movies can be returned in the response

PARAMS

Genre	Comedy

GET (FAIL) Search Movie by Genre

http://localhost:5000/api/v1.0/movies/genre?Genre

Purpose

This request is designed to test the error handling behavior when searching for movies by genre with an incomplete or invalid query parameter.

Endpoint

`GET http://localhost:5000/api/v1.0/movies/genre`

Query Parameters

- `Genre` (required): The genre name to search for (e.g., "Action", "Comedy", "Drama", "Thriller")
  - **Note**: In this test case, the parameter is intentionally left empty or incomplete to validate error handling

Expected Behavior

This is a **failure test case** that validates the API's error handling when:

- The `Genre` query parameter is missing a value
- The genre parameter is malformed or incomplete
- Invalid genre data is provided

Expected Response

- **Status Code**: `404 Not Found`
- **Response**: Error message indicating that the genre was not found or the request was invalid

Test Validation

The post-response script validates that:

javascript

```javascript
pm.test("Status code is 404", function () {
    pm.response.to.have.status(404);
});
```

Usage Notes

- This request is part of the negative testing suite
- It ensures the API properly handles invalid genre search requests
- For successful genre searches, use the "Search Movie by Genre" request with a valid genre value

PARAMS

Genre

GET Genre Count

http://localhost:5000/api/v1.0/movies/genre-count

Genre Count Endpoint

Overview

Retrieves a count of movies grouped by their genres from the movie database.

Endpoint Details

- **Method:** GET
- **Path:** `/api/v1.0/movies/genre-count`

Response

Returns a JSON object where:

- Keys represent unique movie genres
- Values represent the number of movies in each genre

Example Response

json

```json
{
  "Action": 25,
  "Drama": 40,
```

```
  "Comedy": 30,
  "Sci-Fi": 15
}
```

Use Cases

- Analyze movie genre distribution
- Generate genre-based statistics
- Create genre-wise movie dashboards

Notes

- No query parameters required
- Useful for data visualization and reporting

GET Director Count

http://localhost:5000/api/v1.0/movies/director-count

Director Count Endpoint

Overview

This GET request retrieves the count of unique directors in the movie database.

Endpoint Details

- **Method**: GET
- **URL**: `http://localhost:5000/api/v1.0/movies/director-count`

Response

- **Content Type**: JSON
- **Expected Response**:
  - A JSON object containing the total number of unique directors
  - Status Code: 200 OK

Example Response

```json
{
  "total_directors": 150
}
```

Notes

- This endpoint provides a quick way to get an overview of the diversity of directors in the movie database
- No query parameters or request body is required

GET Actor Count

http://localhost:5000/api/v1.0/movies/actor-count

Actor Count Endpoint

Overview

This GET request retrieves the total count of actors in the movie database.

Endpoint Details

- **Method**: GET
- **URL**: `http://localhost:5000/api/v1.0/movies/actor-count`

Response

- Returns a numerical value representing the total number of unique actors in the database.

Usage

Use this endpoint to quickly get an overview of the actor diversity in the movie database.

Example Response

```json
{
  "total_actors": 1234
}
```

Notes

- No authentication required
- No query parameters needed
- Useful for dashboard or statistical purposes

GET Average Review Rating per Movie

http://localhost:5000/api/v1.0/movies/average-review-rating

Get Average Review Rating

Overview

This endpoint retrieves the average review rating across all movies in the database.

Endpoint Details

- **Method:** GET
- **URL:** `/api/v1.0/movies/average-review-rating`

Response

- **Content Type:** JSON
- **Expected Response:**

```json
```

```
{
  "averageRating": number
}
```

Notes

- Returns the overall average rating for all movie reviews
- Useful for getting a quick insight into the general movie rating in the database
- No query parameters or request body required

GET Movies by Actor

http://localhost:5000/api/v1.0/movies/by-actor/Leonardo DiCaprio

Get Movies by Actor

Endpoint Description

Retrieves a list of movies featuring a specific actor.

Parameters

- `actor` (path parameter): The name of the actor to search movies for
  - In this case: Leonardo DiCaprio

Response

- Returns an array of movie objects
- Each movie object typically includes details such as:
  - Movie title
  - Release year
  - Genre
  - Other actor information

Example Use Case

Fetch all movies in which Leonardo DiCaprio has acted.

Possible Responses

- 200 OK: Successfully retrieved movies
- 404 Not Found: No movies found for the specified actor

GET (FAIL) Movies by Actor

http://localhost:5000/api/v1.0/movies/by-actor/

Movies by Actor Endpoint

Description

Retrieves a list of movies featuring a specific actor.

Endpoint Details

- **Method:** GET
- **URL:** `http://localhost:5000/api/v1.0/movies/by-actor/`

Parameters

- Actor name should be passed as a path parameter (not shown in the current request)

Expected Responses

- **200 OK:** Returns an array of movies featuring the specified actor
- **404 Not Found:** No movies found for the given actor

Usage Example

To use this endpoint, replace the actor name in the URL path. Example:
`http://localhost:5000/api/v1.0/movies/by-actor/Tom Hanks`

Notes

- Ensure the actor name is URL-encoded if it contains spaces
- The endpoint is case-sensitive

GET Movie by Minimum IMDb Rating

http://localhost:5000/api/v1.0/movies/minrating/7.5

Get Movies by Minimum IMDb Rating

This endpoint retrieves all movies that have an IMDb rating equal to or greater than the specified minimum rating value.

Endpoint

`GET /api/v1.0/movies/minrating/{rating}`

Path Parameters

Parameter	Type	Required	Description
`rating`	number	Yes	The minimum IMDb rating threshold (e.g., 7.5). Returns all movies with ratings >= this value

Query Parameters

None

Request Headers

- `Content-Type` : application/json (optional)

Authentication

May require authentication depending on API configuration. Check collection-level auth settings.

Response

Returns a JSON array of movie objects that meet the minimum rating criteria.

Success Response (200 OK):

```json
[
  {
    "id": "string",
    "title": "string",
    "year": number,
    "director": "string",
    "genre": "string",
    "imdb_rating": number,
    "actors": ["string"],
    ...
  }
```

Example Usage

- Get all movies with rating >= 7.5: `/api/v1.0/movies/minrating/7.5`
- Get all movies with rating >= 8.0: `/api/v1.0/movies/minrating/8.0`
- Get all movies with rating >= 9.0: `/api/v1.0/movies/minrating/9.0`

Possible Response Codes

- **200 OK**: Successfully retrieved movies matching the criteria
- **400 Bad Request**: Invalid rating parameter format
- **404 Not Found**: No movies found matching the criteria

- **500 Internal Server Error**: Server error occurred

Notes

- The rating parameter accepts decimal values (e.g., 7.5, 8.2)
- Results include movies with ratings exactly equal to or higher than the specified value
- An empty array is returned if no movies meet the criteria

GET (FAIL) Movie by Minimum IMDb Rating

http://localhost:5000/api/v1.0/movies/minrating/

Get Movies by Minimum IMDb Rating

Description

Retrieve a list of movies that have an IMDb rating equal to or above a specified minimum rating.

Endpoint

`GET /api/v1.0/movies/minrating/`

Parameters

- No query parameters are required for this endpoint

Expected Response

- **Success Response (200 OK)**: Returns an array of movie objects with IMDb ratings at or above the minimum threshold
- **Error Response (404 Not Found)**: Indicates no movies meet the rating criteria or an issue with the request

Example Use Case

- Fetching all high-rated movies in the database
- Filtering movies for a top-rated movies section

Notes

- Ensure the minimum rating threshold is configured on the server-side
- The current test expects a 404 response, which might indicate a configuration or implementation issue

---

GET (FAIL) Search Movie by Year

http://localhost:5000/api/v1.0/movies/searchyear?Year=test

Search Movies by Year Endpoint

Overview

This endpoint allows searching for movies based on a specific year.

Request Details

- **Method:** GET
- **Endpoint:** `/api/v1.0/movies/searchyear`

Query Parameters

- `Year` (required): The year to search movies for
  - Type: String
  - Example: Use a valid year like '2020' or a specific year value

Possible Responses

- **200 OK:** Returns a list of movies from the specified year
- **404 Not Found:**
  - Occurs when no movies are found for the given year
  - Possible reasons:
    1. Invalid year format
    2. No movies exist for the specified year
    3. Incorrect year parameter

Example Requests

- Valid request: `http://localhost:5000/api/v1.0/movies/searchyear?Year=2020`
- Current test request: `http://localhost:5000/api/v1.0/movies/searchyear?Year=test` (returns 404)

Troubleshooting

- Ensure the 'Year' parameter is a valid year
- Check the API documentation for supported year formats
- Verify the backend implementation of the year search functionality

PARAMS

Year	test

GET Get All Awards

http://localhost:5000/api/v1.0/awards/all

Get All Awards Endpoint

Purpose

This endpoint retrieves all awards stored in the database.

Request Details

- **HTTP Method**: GET
- **Endpoint**: `/api/v1.0/awards/all`
- **Authentication**: Not required

Expected Response

- **Status Code**: 200 OK
- **Response Body**: JSON array containing all awards with their details (award name, category, year, winner status, associated movie)

Example

```
Plain Text

GET /api/v1.0/awards/all
```

Notes

- Returns all awards in the database
- No pagination is implemented
- Each award includes information about the associated movie
- Returns an empty array if no awards exist

GET Get awards by Movie ID

http://localhost:5000/api/v1.0/movies/69026399a9f15b2a209c6169/awards

Get Awards by Movie ID

Overview

Retrieves all awards associated with a specific movie using its unique identifier

Retrieves all awards associated with a specific movie using its unique identifier.

Endpoint Details

- **Method:** GET
- **Path:** `/api/v1.0/movies/{movieId}/awards`

Parameters

- `movieId` (path parameter, required): The unique identifier of the movie
  - Example: `69026399a9f15b2a209c6169`

Response

Returns an array of award objects for the specified movie. Each award object typically includes:

- Award name
- Year of award
- Category
- Recipient details

Possible Status Codes

- `200 OK`: Successfully retrieved movie awards
- `404 Not Found`: Movie ID does not exist

Example Use Case

Retrieve all awards won by a specific movie to showcase its critical acclaim or achievements.

GET (FAIL) Get awards by Movie ID

http://localhost:5000/api/v1.0/movies/69026399a9f15b2a209c6092/awards

Get Awards by Movie ID

This endpoint retrieves all awards associated with a specific movie using the movie's unique identifier.

Purpose

Fetch a list of awards (nominations and wins) for a particular movie, including details about the award ceremony, category, year, and whether the movie won.

Parameters

Path Parameters:

- `movieId` (string, required) - The unique identifier of the movie. This should be a valid MongoDB ObjectId format.

Expected Responses

Success (200 OK) Returns an array of award objects for the specified movie:

```json
[
  {
    "_id": "award_id",
    "movieId": "movie_id",
    "awardCeremony": "Academy Awards",
    "category": "Best Picture",
    "year": 2020,
    "won": true
  }
]
```

Not Found (404) Returns when the movie ID doesn't exist in the database:

```json
{
  "error": "Movie not found"
}
```

Bad Request (400) Returns when the movie ID format is invalid:

```json
{
  "error": "Invalid movie ID format"
}
```

Notes

- The endpoint returns an empty array if the movie exists but has no awards
- Award data includes both nominations and wins (check the `won` field)
- Results are typically sorted by year in descending order

GET Get Award Winners by Year

http://localhost:5000/api/v1.0/awards/2023/winners

Get Award Winners by Year

Overview

Retrieves a list of award winners for a specific year.

Endpoint Details

- **Method:** GET
- **URL:** `/api/v1.0/awards/2023/winners`
- **Base URL:** `http://localhost:5000`

Response

- **Status Code:** 200 OK
- **Content Type:** application/json

Purpose

This endpoint allows users to fetch all award winners for the year 2023.

Example Use Case

Retrieve a comprehensive list of award-winning movies or individuals for the specified year.

Notes

- Ensure the server is running locally
- No authentication required for this endpoint
- Returns an array of award winners

GET (FAIL) Get Award Winners by Year

http://localhost:5000/api/v1.0/awards/1999/winners

Get Award Winners by Year

Overview

Retrieves a list of award winners for a specific year.

Endpoint Details

- **Method**: GET
- **URL**: `/api/v1.0/awards/1999/winners`
- **Base URL**: `http://localhost:5000`

Parameters

- **Year** (path parameter): The specific year for which award winners are being retrieved (in this example, 1999)

Expected Responses

- **200 OK**: Returns a list of award winners for the specified year
- **404 Not Found**: No award winners found for the given year (current test scenario)

Possible Use Cases

- Retrieving historical award winners
- Generating year-specific award reports
- Analyzing award-winning movies or individuals

Notes

- Ensure the year parameter is a valid year in the database
- The endpoint is case-sensitive and expects a numeric year

Current Test Status

The request is currently configured to expect a 404 status code, which might indicate:

- The endpoint is not fully implemented
- No data exists for the specified year
- Potential testing or development phase of the API

GET Awards by Award Ceremony

http://localhost:5000/api/v1.0/awards/awardcompany/Academy Award

Get Awards by Award Ceremony

Overview

Retrieves a list of awards for a specific award ceremony.

Endpoint Details

- **Method:** GET
- **URL:** `http://localhost:5000/api/v1.0/awards/awardcompany/Academy Award`

Parameters

- `awardcompany` (path parameter): The name of the award ceremony (in this case, 'Academy Award')

Response

Returns a JSON array of awards associated with the specified award ceremony.

Expected Response

- **Status Code:** 200 OK
- **Response Body:**

```json
[
  {
    "movieId": "string",
    "movieTitle": "string",
    "awardYear": "number",
    "awardCategory": "string",
    "winner": "boolean"
  }
]
```

Example

Fetches all Academy Award nominations and winners for movies in the database.

GET (FAIL) Awards by Award Ceremony

http://localhost:5000/api/v1.0/awards/awardcompany/Test Fail

Purpose

This endpoint retrieves awards associated with a specific award ceremony or company. It's designed to fail with a 404 status code when the specified award ceremony does not exist in the database.

Expected Input

- **Path Parameter**: `awardcompany` (string) - The name of the award ceremony/company to search for
  - Example: "Test Fail" (non-existent ceremony for testing)
  - Example valid values might include: "Academy Awards", "Golden Globes", etc.

Expected Responses

404 Not Found

When the specified award ceremony does not exist in the database:

```json
{
  "error": "Award ceremony not found"
}
```

200 OK (for valid ceremonies)

200 OK (for valid ceremonies)

When the award ceremony exists, returns an array of awards:

```json
[
  {
    "id": 1,
    "movie_id": 123,
    "award_name": "Best Picture",
    "award_company": "Academy Awards",
    "year": 2023,
    "won": true
  }
]
```

Test Validation

This request includes a post-response script that validates the response status code is 404, confirming the expected failure scenario for non-existent award ceremonies.