Secure Movie API Development

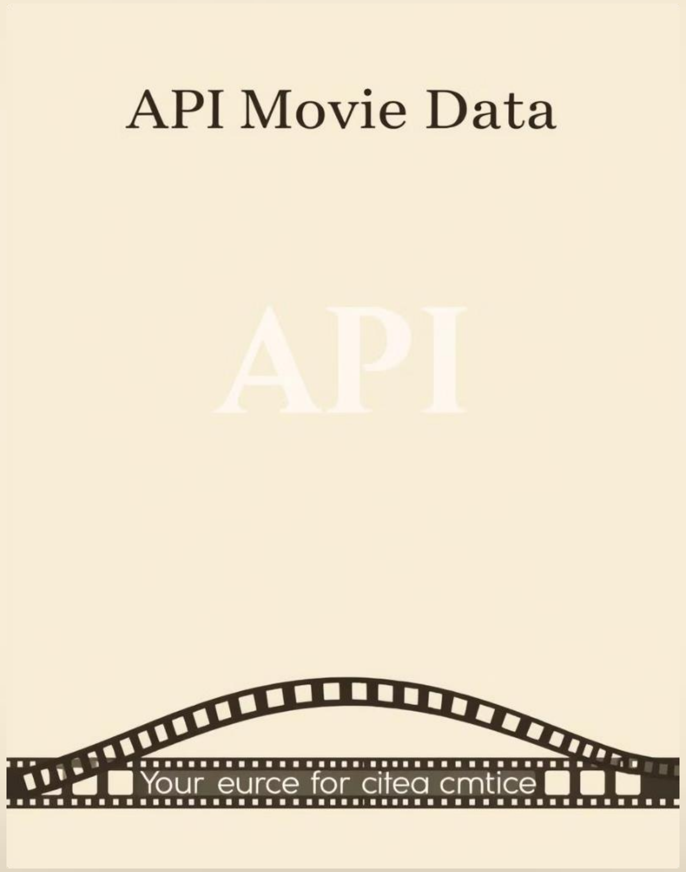
This report provides an overview of the development and implementation of Movie API. Designed to meet modern functionality, security, and performance, the API built using Node.js and MySQL serves as a robust, scalable solution for retrieving detailed movie data.

The main objective of this project is to create a RESTful API with a focus on efficient and graceful error handling, Security measures and perfect user experience .This includes resources such as HTTPS/SSL encryption, JWT-based authentication, and input validation. To meet these essential requirements, this API not only facilitate access to reliable data; But it also ensures protection of the privacy of user data and applications from potential threats.

This report delves into the application's technical aspects, architecture, challenges encountered during development. and the measures taken to overcome them. It also highlights the application testing process. Security considerations and limitations, providing a holistic view of the project development process. From this report, readers can gain information about the thought process, strategies, and technologies used to create a safe and effective movie API.

.

by Ebad Salehi



Features

Implemented

- User registration and authentication using JWT.
- HTTPS enforcement with SSL/TLS.
- Movie search functionality with pagination.
- Detailed movie information endpoint.
- Secure file upload system for movie posters.

Unimplemented

- Advanced search and filtering.
- Token revocation system.
- Refresh token mechanism.

Detailed Breakdown

1 **User Authentication**

Email and password-based registration and login, using `JWT` tokens for secure authentication.

2 **Movie Data Retrieval**

Search by title or Fetch detailed movie data by IMDb ID.

3 **Poster Upload and Retrieval**

PNG file uploads with size constraints. Token-protected retrieval of poster images.

4 **Error Handling**

Comprehensive error responses for various scenarios.

Technical Architecture & Challenges

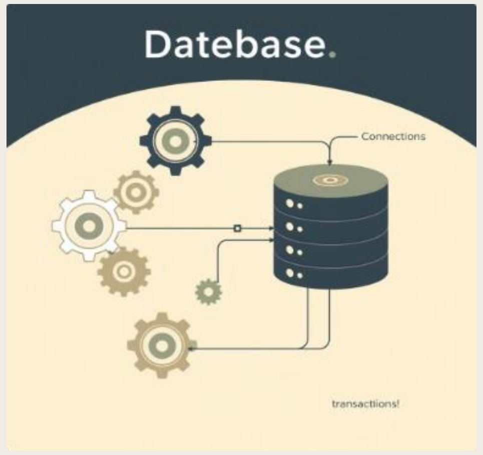
Architecture Overview

- **Routes:** Maps API endpoints to controllers.
- **Controllers:** Encapsulates logic for data handling and API responses.
- **Database:** Uses MySQL with Knex.js for query building.

Technical Challenges

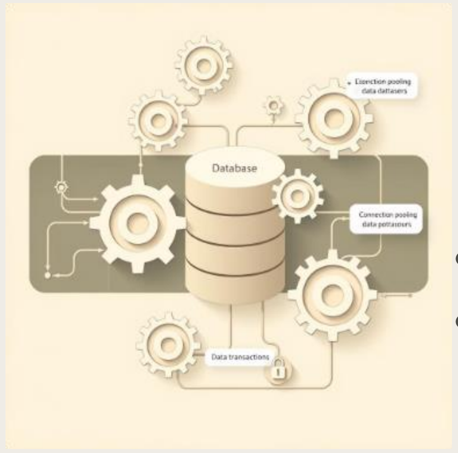
- **Database Setup:** Initial connection issues resolved using the `mysql2` driver.
- **JWT Implementation:** Securing token-based authentication with expiration management.
- **HTTPS Configuration:** Generated and configured SSL certificates for local development.

Database Details



Schema

- Users: Stores user credentials securely.
- Movies: Holds metadata like title, year, and IMDb ID.
- Posters: Manages poster file paths and links to movies.


Database
Esonction pooling data dattasers
Connection pooling data pottasours
Data transactions

Integration

- Connection pooling with `Knex.js.`
- Transactions for data integrity.

Testing and Limitations

Authentication

- Registered and logged-in users successfully.
- Invalid credentials return appropriate error messages.

Movie Search

Tested with valid and invalid titles and imdbID, ensuring accurate error handling.

Poster Uploads

Validated PNG format and size limits.

Error Scenarios

Simulated a wide range of possible errors messages such as database connection failure, token errors, …

Security & HTTPS

1 HTTPS Importance

- Encrypts sensitive data during transmission.
- Ensures server authenticity.
- Prevents data tampering.

2 Why a Certificate Authority?

- Trusted across browsers.
- Validates organizational legitimacy.
- Easy integration with web services.

3 Preferred CA

for production stage,using services such as *Let's Encrypt for free*, which is automated, and widely supported certificates is recommended.

4 Certificate Setup

For this project, running `generate-certificates.js` will generate two `.pem` files in `ssl` folder. For more details, please refer to `readme.md`

Robustness and Error Handling

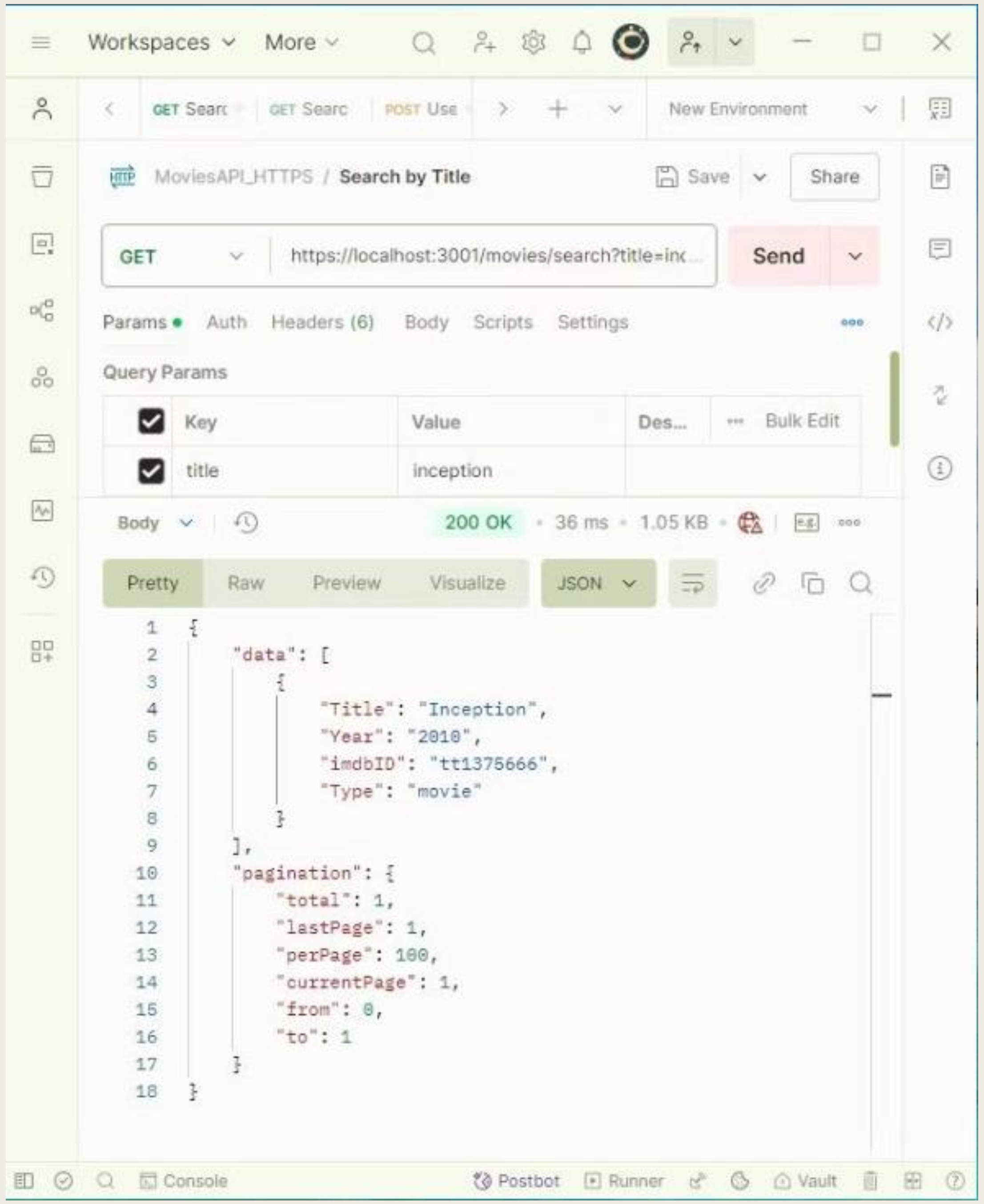
Error Handling Features

- Unified error middleware in Express.
- Custom error messages for user authentication.

Robustness

- Validation checks for all endpoints.
- Resilient to malformed requests.

Screenshots



MoviesAPI_HTTPS / Search by Title

GET https://localhost:3001/movies/search?title=inc

Query Params

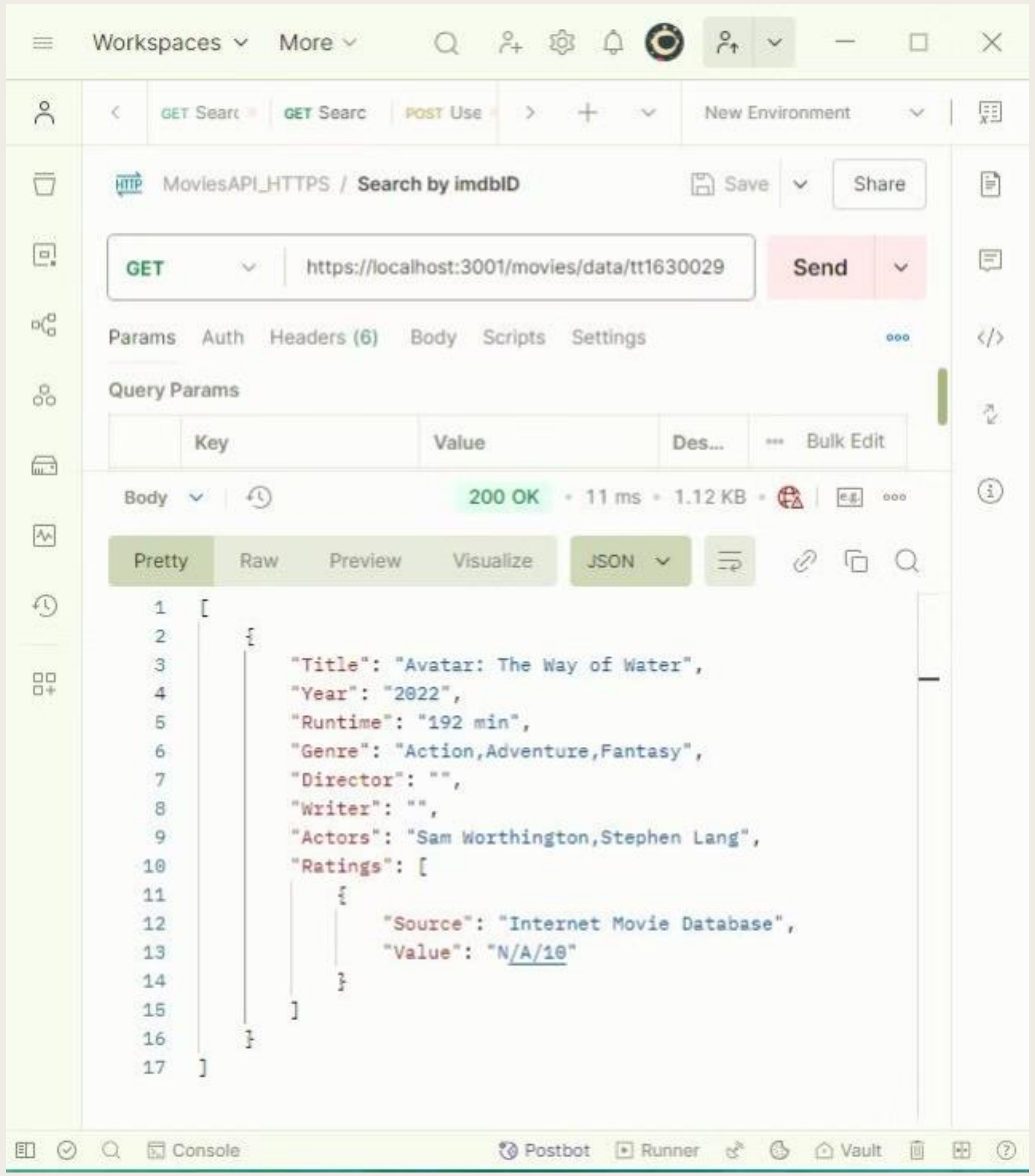
Key	Value	Des...
title	inception	

200 OK · 36 ms · 1.05 KB

JSON

```
{
    "data": [
        {
            "Title": "Inception",
            "Year": "2010",
            "imdbID": "tt1375666",
            "Type": "movie"
        }
    ],
    "pagination": {
        "total": 1,
        "lastPage": 1,
        "perPage": 100,
        "currentPage": 1,
        "from": 0,
        "to": 1
    }
}
```

Search by Movie Title



MoviesAPI_HTTPS / Search by imdbID

GET https://localhost:3001/movies/data/tt1630029

Query Params

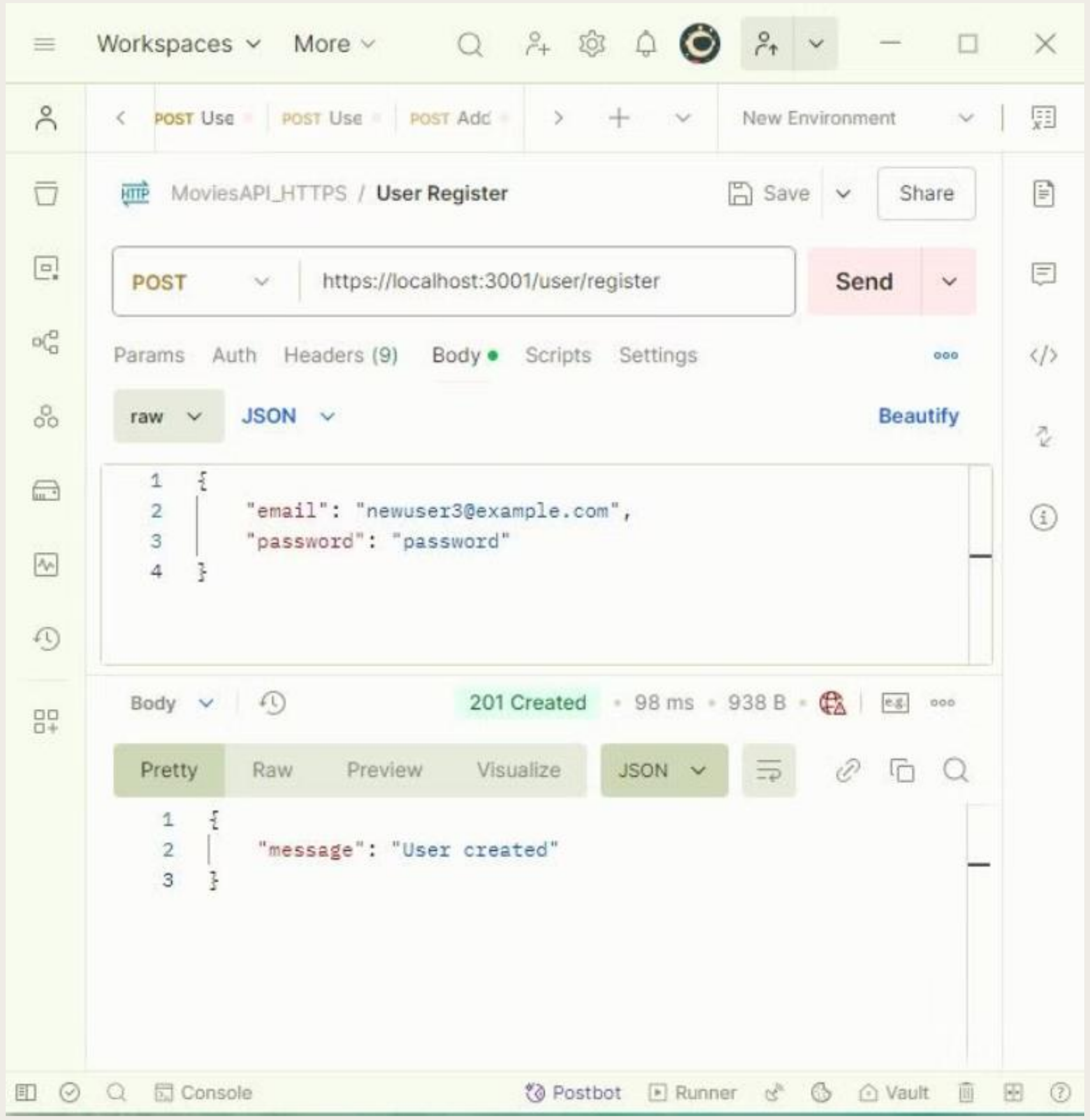
Key	Value	Des...

200 OK · 11 ms · 1.12 KB

JSON

```
[
    {
        "Title": "Avatar: The Way of Water",
        "Year": "2022",
        "Runtime": "192 min",
        "Genre": "Action,Adventure,Fantasy",
        "Director": "",
        "Writer": "",
        "Actors": "Sam Worthington,Stephen Lang",
        "Ratings": [
            {
                "Source": "Internet Movie Database",
                "Value": "N/A/10"
            }
        ]
    }
]
```

Search by imdbID



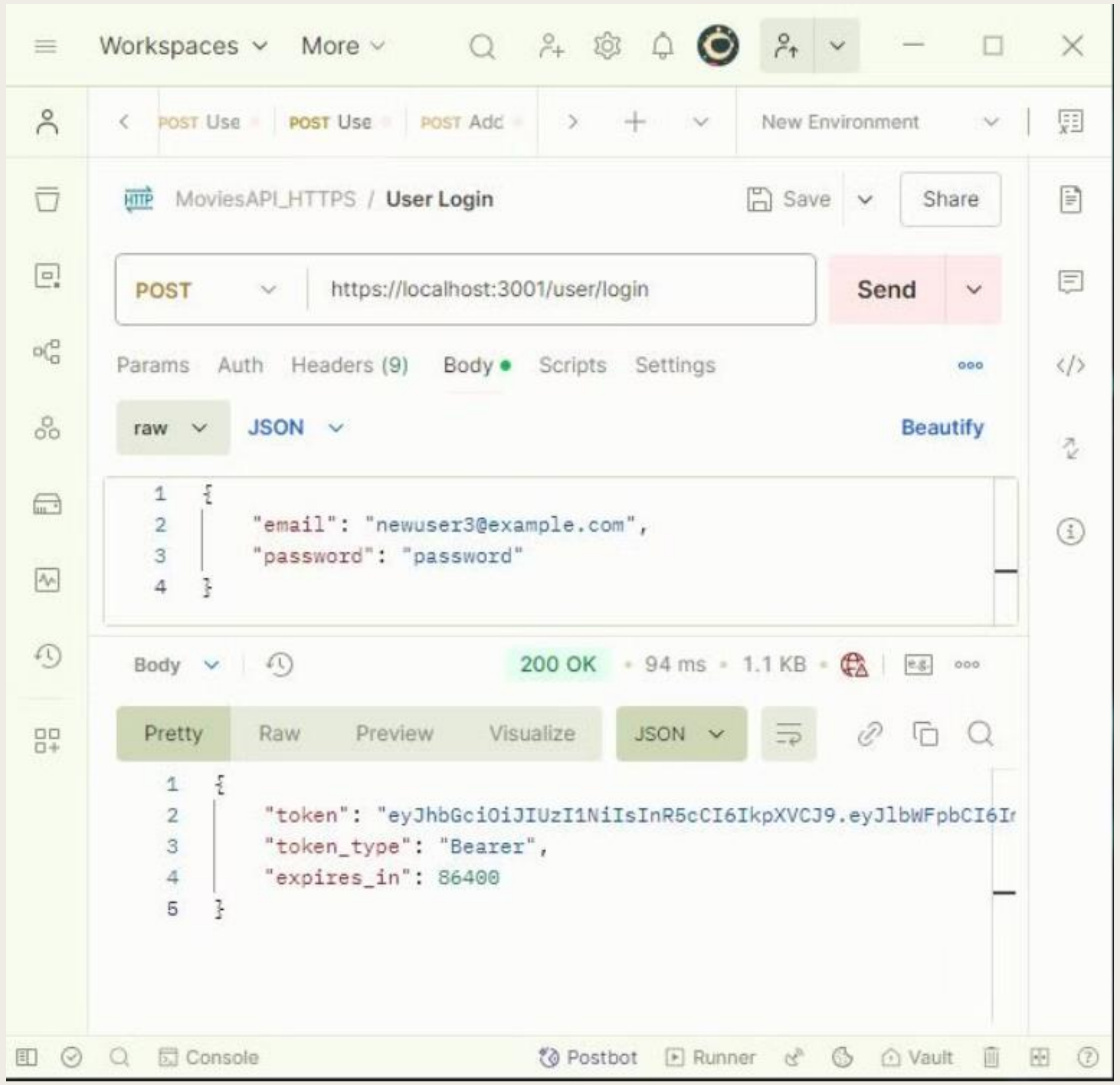
POST https://localhost:3001/user/register

```
{
    "email": "newuser3@example.com",
    "password": "password"
}
```

201 Created • 98 ms • 938 B

```
{
    "message": "User created"
}
```

New User Registration



POST https://localhost:3001/user/login

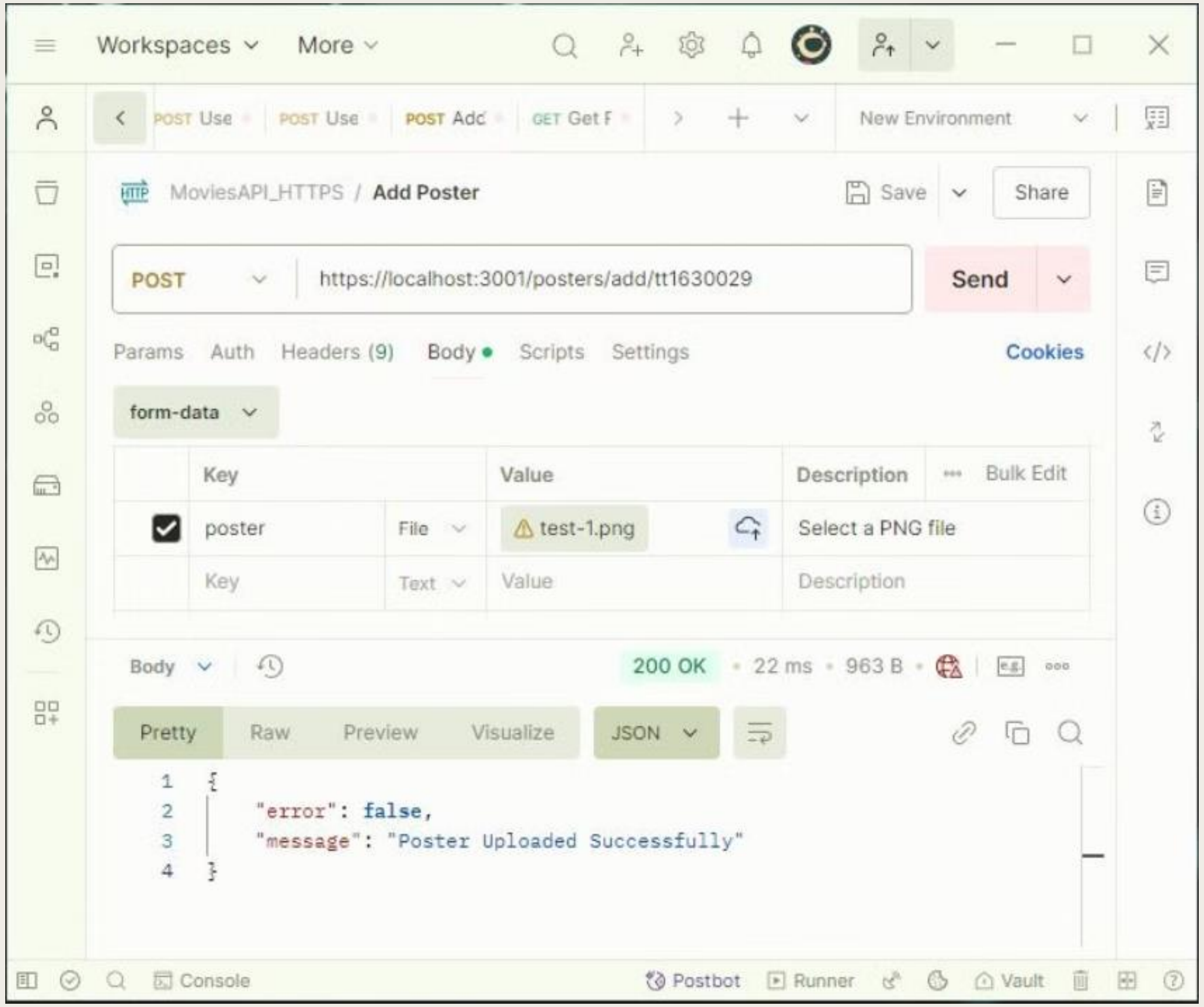
```
{
    "email": "newuser3@example.com",
    "password": "password"
}
```

200 OK • 94 ms • 1.1 KB

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6I
    "token_type": "Bearer",
    "expires_in": 86400
}
```

User Login

Protected Endpoints



MoviesAPI_HTTPS / Add Poster

POST https://localhost:3001/posters/add/tt1630029

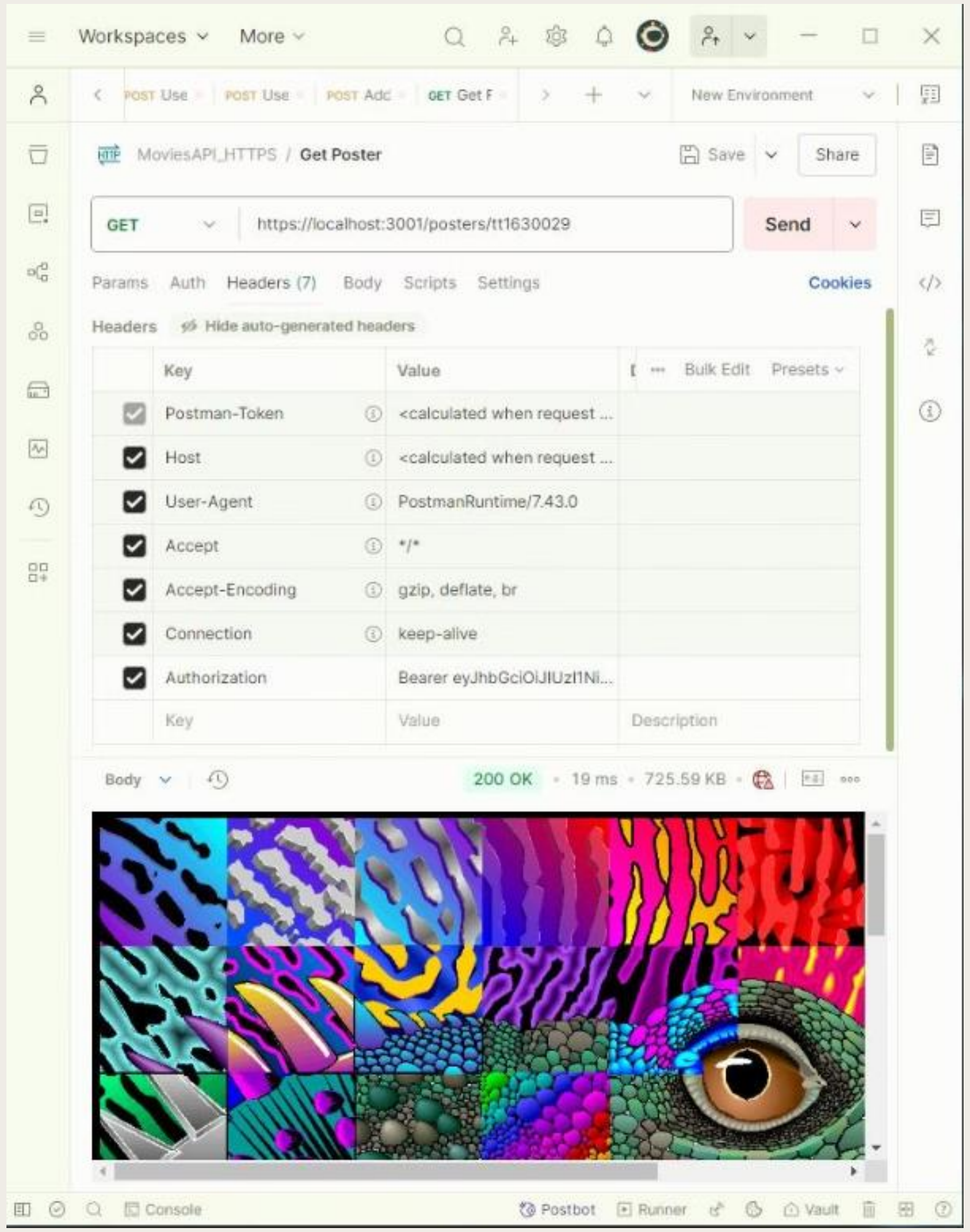
Body: form-data

Key		Value	Description
✓ poster	File	test-1.png	Select a PNG file
Key	Text	Value	Description

Body — 200 OK • 22 ms • 963 B — JSON

```
{
    "error": false,
    "message": "Poster Uploaded Successfully"
}
```

Upload a movie poster



MoviesAPI_HTTPS / Get Poster

GET https://localhost:3001/posters/tt1630029

Headers

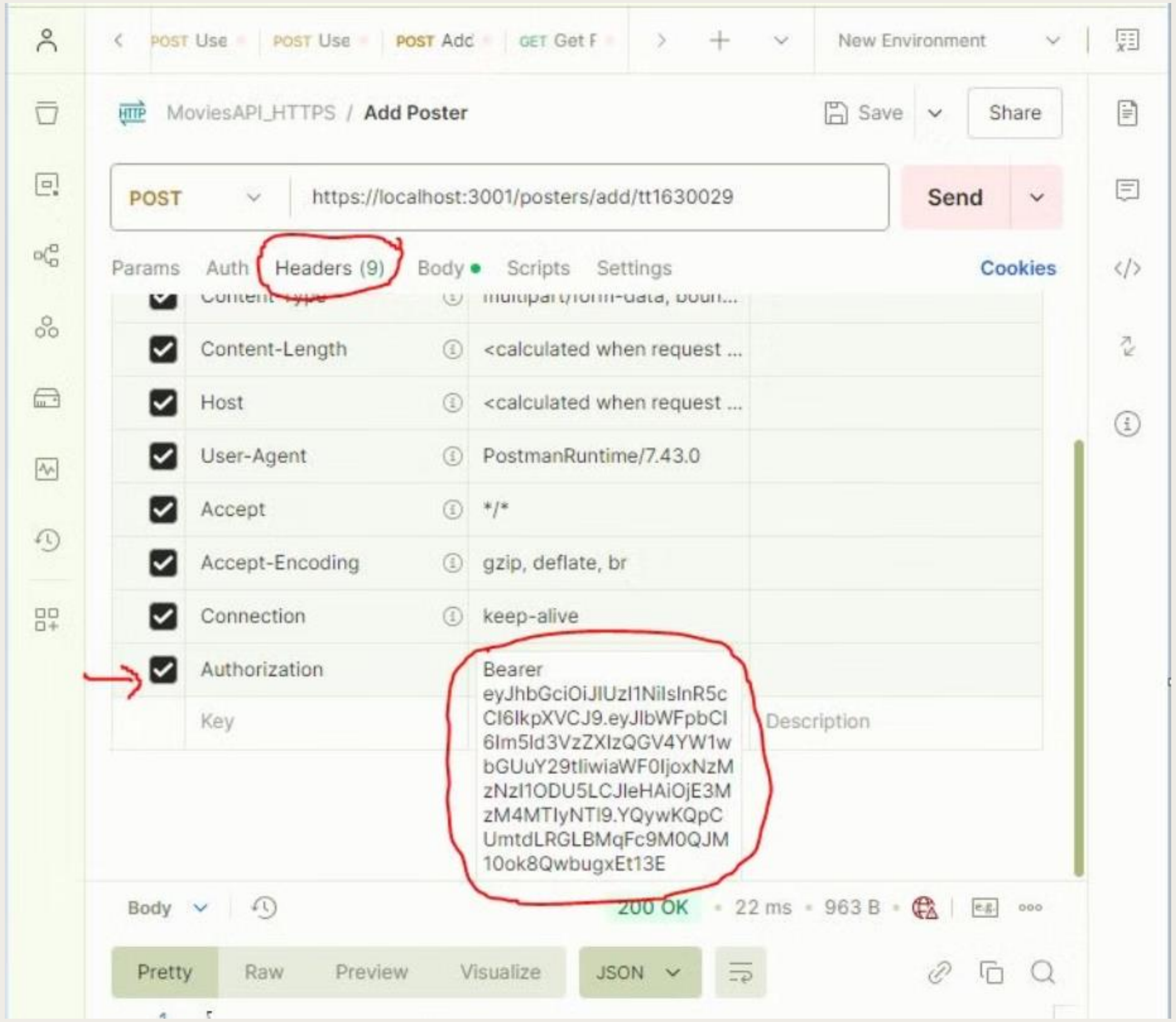
	Key	Value	Description
✓	Postman-Token	<calculated when request ...	
✓	Host	<calculated when request ...	
✓	User-Agent	PostmanRuntime/7.43.0	
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	Authorization	Bearer eyJhbGciOiJIUzI1Ni...	
	Key	Value	Description

Body — 200 OK • 19 ms • 725.59 KB

Retrieve a poster

Using JWT Tokens to interact With Protected Endpoints

a JWT token generated during the login, must be passed in the header of the request for authentication.



MoviesAPI_HTTPS / Add Poster

POST https://localhost:3001/posters/add/tt1630029

Params Auth Headers (9) Body Scripts Settings Cookies

	Key	Value	Description
✓	Content-Type	multipart/form-data; boun...	
✓	Content-Length	<calculated when request ...	
✓	Host	<calculated when request ...	
✓	User-Agent	PostmanRuntime/7.43.0	
✓	Accept	*/*	
✓	Accept-Encoding	gzip, deflate, br	
✓	Connection	keep-alive	
✓	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6Im5ld3VzZXIzQGV4YW1wbGUuY29tIiwiaWF0IjoxNzMzNzI1ODU5LCJleHAiOjE3MzM4MTIyNTI9.YQywKQpCUmtdLRGLBMqFc9M0QJM10ok8QwbugxEt13E	
	Key		Description

200 OK • 22 ms • 963 B

References

- ChatGPT. (2024). Responses and insights on API development, security practices, and troubleshooting. Retrieved from OpenAI.
- Knex.js Documentation. (n.d.). *Knex.js: A SQL query builder for Node.js*. Retrieved December 6, 2024, from https://knexjs.org
- Let's Encrypt. (n.d.). *Free SSL/TLS certificates*. Retrieved December 6, 2024, from https://letsencrypt.org
- Multer Documentation. (n.d.). *Multer: Middleware for handling multipart/form-data*. Retrieved December 6, 2024, from https://github.com/expressjs/multer
- Node.js Documentation. (n.d.). *Node.js: JavaScript runtime built on Chrome's V8 JavaScript engine*. Retrieved December 6, 2024, from https://nodejs.org
- Node.js HTTPS Tutorial. (n.d.). *Node.js HTTPS setup*. Retrieved December 6, 2024, from https://www.youtube.com/watch?v=Oe421EPraWQ
- Postman API Testing. (n.d.). *Postman: API testing for developers and teams*. Retrieved December 6, 2024, from https://postman.com
- test-1.png. (n.d.). *Camera illustration*. Retrieved December 6, 2024, fromhttps://www.vecteezy.com/png/1198768-camera
- test-2.png. (n.d.). *Camera PNG clipart*. Retrieved December 6, 2024, from https://www.vecteezy.com/png/1198753-camera
- test-3.png. (n.d.). *Camera film roll clipart design*. Retrieved December 6, 2024, from https://www.vecteezy.com/png/9306133-camera-film-roll-clipart-design-illustration
- JWT Documentation. (n.d.). *JSON Web Tokens for authentication*. Retrieved December 6, 2024, from https://github.com/auth0/node-jsonwebtoken

Appendix

Requirements

Node.js

(v14 or higher).

MySQL

(v5.7 or higher).

SSL certificates

(Generated Locally).

Installation Steps

25%

Clone the repository:

```
git clone https://github.com/Ebad-S/Movies-API
```

50%

Install dependencies:

```
npm install
```

75%

Configure `.env` file to reflect your local setup and also generate SSL certificates.

100

Run the server:

```
npm run dev
```

Testing Instructions:

- Use ***Postman*** to import API collection and interact with the endpoints
- ***Test endpoints***: For a comprehensive documentation, setup guide, and example 'query path', please refer to `readme.md` in the project's GitHub repo.
- ***Runtime issues***: refer to `logs/errors.log.`