IFQ716 Advanced Web Development
Assignment 2: Node.js and Express-based API

Jenny Guidetti
Queensland University of Technology

Introduction

A Node.js and express-based application programming interface (API) was developed and tested using modern methodologies (OpenJS Foundation, 2017). The RESTful API enables users to retrieve data from a structured query language (SQL) database ‘movies,’ which contains detailed movie information. Six different endpoints were implemented using Express’s routing and middleware features, based on the provided Swagger documentation (*Ratings and Streaming API,* n.d.). The report discusses the implemented elements, application architecture, testing outcomes and security aspects.

The six API endpoints were successfully implemented to function as specified in their Swagger documentation. Two of these endpoints are restricted to authorised users with bearer tokens. However, some endpoints exhibit error variations due to gaps in the documentation. Adjustments were made to improve error handling for service failures, undocumented edge cases, and instances with no results. The response headers sent also differ, resulting from the use of the ‘helmet’ module (Helmet, n.d.).

Several features were implemented to enhance the security of the API. The external SQL-related middleware Knex (Knex.js, n.d.) was used to avoid raw SQL queries. To simplify production, object-relational mapping (ORM) could be considered to eliminate the use of SQL altogether. Self-signed HTTPS was implemented to enable transport layer security (TLS) encryption. A certificate authority (CA) signed certificate should be obtained and implemented for verification purposes. Database credential information and JSON web token (JWT) secrets are stored securely in an environment variable (Auth0, n.d.).

Technical Description of the Application

Three routes have been implemented; movies, posters and users. Each API endpoint has a module within these routes, promoting separation of concerns and enhancing maintainability. The index.js file references these, ensuring that only the index router needs to be invoked in app.js. This approach keeps app.js relatively clean, facilitating long-term maintenance. Authorisation and validation logic is encapsulated and located in a dedicated middleware folder. Other middleware functions and modules remain in the entry point, due to their minimal code size. As the project grows, these could be moved to the middleware folder to maintain a clean app.js.

Five out of six endpoints use promises with the ‘.then()’ method to handle data, while one uses async/await with a try/catch block. All endpoints are designed to return a 500 status code in the event of a server or database error. More complex endpoints could be refactored into smaller, more focused functions to enhance readability and maintainability. Consistency could also be improved by implementing async/await with try/catch blocks to all routing functions.

‘/movies/search’ uses a GET request to allow users to search for movies based on three parameters and returns an array of movies.‘paginatedMovies’ handles the database query for fetching movies based on the provided title and year. It orders the results using ‘.orderBy’ and the ‘paginate()’ method sets the limit for items per page with ‘perPage = 100’. Finding the extension of knex’s query builder ‘knex-paginate’ was key to pagination of results. The routing function extracts parameters and sets page number default. Comprehensive error handling ensures validation of query parameters, mandatory parameters, year format, and existence of results. Ensuring the correct ordering of errors was vital for accurate error handling.

‘/movies/data/{imdbID}’ returns movie information based on the IMDb ID using a GET request. The ‘getMoviesDb’ function fetches movie data from the database by linking the ‘basics’ and ‘crew’ tables on the ‘tconst’ column. ‘.first()’ is used to select a single object rather than an array, and the promise is rejected if no results are found. The ‘getNames’ function combines directors and writers from the ‘names’ table into a single string, as the original function failed to return multiple names. Since the movie database lacks actors or ratings information, ‘getMoviesOmdb’ fetches this from the OMDB API (Fritz, B., 2014). The routing function validates the IMDb ID parameter, combines data from the database and OMDB API, and returns a formatted JSON response. Uploading OMDB API information to the database via a protected endpoint could be an improvement.

The ‘/posters/{imdbID}’ endpoint fetches movie posters from the database based on the provided IMDb ID using a GET request. Custom authorization middleware ensures the request is authenticated. A Knex query retrieves the row in the database, checks if results are found, and returns the poster image. Errors are returned if no poster is found or there’s a server error.

The ‘/posters/add/{imdbID}’ endpoint handles uploading a poster image for an IMDb ID to the database using a POST request. Authorization middleware authorizes users, and Multer handles multipart/form-data to upload files (Multer, n.d.). Multer is configured with ‘multer.memoryStorage()’ to store uploaded files in memory as ‘Buffer’ objects. The IMDb ID and uploaded file are extracted using ‘req.params’ and ‘req.file’ respectively. A ‘posters’ table was created in the ‘database using ‘LONGBLOB’ to store bigger files. The data is inserted into this table, and a success message is returned. Error messages handle cases where a poster already exists for the IMDb ID or parameters are missing. The use of ‘try/catch’ blocks helps address headers error during production with ‘.then()’ statements.

The ‘/user/registration’ endpoint allows users to create a new account by providing an email and password using a POST request. The email and password are extracted from the request body, and the password is securely hashed using ‘bcrypt’ before being stored in the database (Bcrypt, n.d.). These are inserted into the ‘users’ table created in the database and a success message is returned. Errors handle missing information with a 400 status code, and an existing email with a 409 status code.

The ‘/user/login’ endpoint authenticates users by verifying their email and password against records in the ‘users’ table using a POST request. If authentication is successful, it generates a

JWT for the user to use for authenticated requests. The email and password are extracted from the request body and ‘bcrypt’ compares the provided password with the hashed password. If they match, a success message is logged, and a JWT with a 24-hour expiry time is generated and signed with a secret key. The response includes the token, token type and expiration time. The request body is validated, returning a 400 status code with an error message if missing information. Errors also handle cases where the user is not found or the passwords do not match.

Testing and Limitations

Each endpoint specified in the API documentation was thoroughly tested to ensure it performed as expected. All endpoints functioned correctly in normal conditions, returning the expected data and HTTP status codes (200, 201). Handling of edge cases included testing invalid query parameters, missing mandatory parameters, non-existent IMDb IDs, no results found, and error variations. These edge cases resulted in a mixture of HTTP status codes (400, 401, 409) and error messages. MySQL database connection errors were forced to verify the applications ability to respond with a 500 status code. The application demonstrates robust error handling through consistent use of HTTP status codes and informative error messages. This ensures users receive feedback on issues encountered. The reliance of the API on the external OMDB API for additional movie data introduces dependency risks.

Security

HTTPS is important in ensuring the confidentiality and integrity of data in transit. It provides authentication which assures clients of a legitimate server and prevents eavesdropping and man-in-the-middle attacks. Whilst the self-signed certificate used is suitable for developing and testing, a CA is trusted by browsers and clients, providing a higher level of trust and security in comparison. Let’s encrypt (Internet Security Research Group, n.d.) is a CA that offers free, automated certificates, making it a suitable choice for API deployment. Certbot (Electronic Frontier Foundation, n.d.) can be used to install and automatically renew certificates without manual intervention.

Although this application has implemented several security measures, there are areas for improvement. The risk of SQL injection attacks is reduced through the use of Knex as a query builder. Helmet helps secure the application by setting various HTTP headers to protect against common web vulnerabilities including cross-site scripting. Although bearer tokens are implemented which are harder for attackers to trick, implementing rate limiting on login attempts is necessary to prevent brute force attacks. A new MySQL user was created to reduce the risk of privilege escalation attacks but privileges need to be altered. Passwords are hashed and salted before storage to enhance password security. Using well-maintained and reputable npm packages reduces the risk of introducing malicious code; however several of the packages had vulnerabilities flagged. Overall security of the application can be strengthened by regularly updating dependencies, enhancing logging and monitoring and staying informed about the latest security advisories.

References

Auth0 (n.d.). *JWT.IO.* Retrieved June 11, from https://jwt.io/

Bcrypt (n.d.). Retrieved June 11, 2024, from https://www.npmjs.com/package/bcrypt

Electronic Frontier Foundation (n.d.). *Certbot*. Retrieved June 16, 2024, from https://certbot.eff.org/

Fritz, B. (2014). *The Open Movie Database*. OMDb API. Retrieved June 15, 2024, from https://www.omdbapi.com/

Helmet (n.d.). Retrieved June 11, 2024, from https://www.npmjs.com/package/helmet

Internet Security Research Group (n.d.). *Let's Encrypt*. Retrieved June 16, 2024 from https://letsencrypt.org/

Knex.js (n.d.). Retrieved June 10, 2024, from https://knexjs.org/

Kong (n.d.). *Insomnia* (Version 9.2.0)[Computer Software]. https://docs.insomnia.rest/insomnia/install

Movie information API (n.d.). Swagger. Retrieved June 10, 2024, from http://54.79.30.138:3001/

Multer (n.d.). Retrieved June 11, 2024, from https://www.npmjs.com/package/multer

OpenJS Foundation (2017). *Express*. Retrieved June 10, 2024, from https://expressjs.com/

Oracle Corporation (n.d.). MySQL Workbench (Version 8.0.36). [Computer Software]. https://dev.mysql.com/downloads/workbench/

Appendix

Installation Guide:
This guide will cover what is required to install this code on a new machine.

Step 1. Set up Node and Visual Studio (VS) Code:
Download and install applications:
https://nodejs.org/en/
https://code.visualstudio.com/Download

Step 2. Install express and express-generator globally:
Open VS Code and run the following in the terminal

```
npm install express express-generator -g
```

Step 3. Extract the API files to the desired directory where files were extracted:

```
cd path/to/extracted/project
```

Step 3. Install project dependencies:

```
npm install
```

Step 4. Install Mysql and Knex in the project directory:

```
npm install knex mysql2 --save
```

Step 5. Install additional modules:

```
npm install cors

npm install morgan

npm install swagger-ui-express

npm install jsonwebtoken

npm install dotenv --save

npm install bcrypt

npm install helmet

npm install knex knex-paginate --save

npm install --save multer
```

Step 6. Set up a MySQL Server and MySQL Workbench
Download and install these from the following links:
https://dev.mysql.com/downloads/installer/

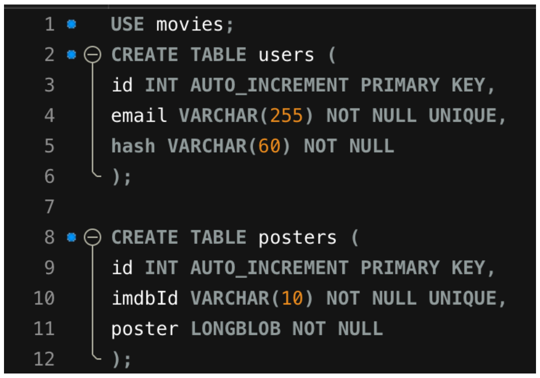
https://dev.mysql.com/downloads/workbench/

Step 7. MySQL Workbench setup

1. Download the database:
   https://canvas.qutonline.edu.au/courses/1604/assignments/7021
2. Open MySQLWorkbench and set up a new MySQL connection by pressing the '+' button
3. Under the File tab, select 'Open new SQL script' and open 'movie_small.sql'
4. Run the code by pressing the lightning bolt

Step 8. Add tables to MySQL Workbench

1. Open a new SQL query tab
2. To create a users table and posters table run the following code and refresh the schemas:

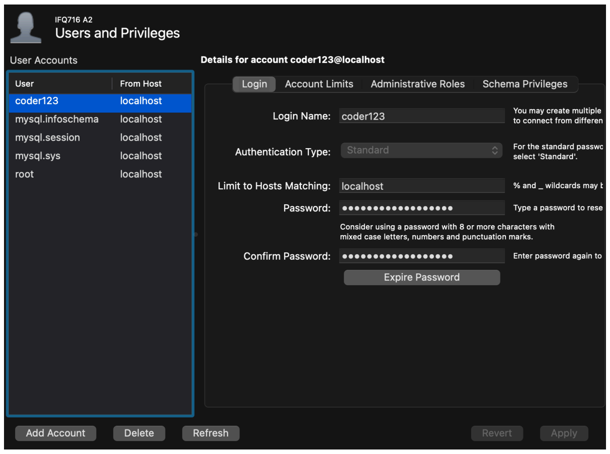


```sql
USE movies;
CREATE TABLE users (
id INT AUTO_INCREMENT PRIMARY KEY,
email VARCHAR(255) NOT NULL UNIQUE,
hash VARCHAR(60) NOT NULL
);

CREATE TABLE posters (
id INT AUTO_INCREMENT PRIMARY KEY,
imdbId VARCHAR(10) NOT NULL UNIQUE,
poster LONGBLOB NOT NULL
);
```

Step 9. Add new user

1. Navigate to ‘Users and Privileges’ under the Administration tab
2. Add account and enter login name, host name and password:



IFQ716 A2
Users and Privileges

User Accounts

User	From Host
coder123	localhost
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost

Details for account coder123@localhost

Login Name: coder123 — You may create multiple to connect from differen

Authentication Type: Standard — For the standard passwo select 'Standard'.

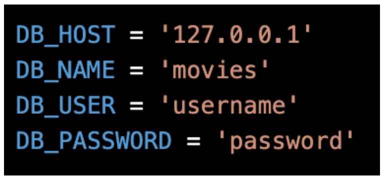
Limit to Hosts Matching: localhost — % and _ wildcards may b

Password: •••••••••••••••••• — Type a password to rese

Consider using a password with 8 or more characters with mixed case letters, numbers and punctuation marks.

Confirm Password: •••••••••••••••••• — Enter password again to

3. Update the environmental variable with these:



```
DB_HOST = '127.0.0.1'
DB_NAME = 'movies'
DB_USER = 'username'
DB_PASSWORD = 'password'
```

Step 10. Run the server: `npm start`

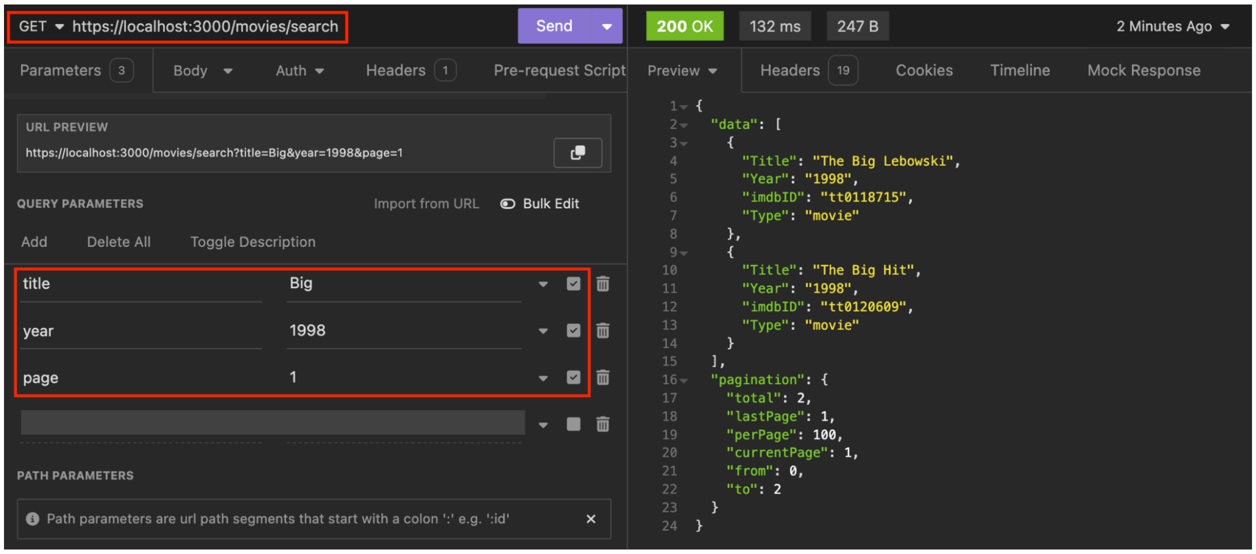
Step 11. Navigate to the following link in your browser to access the swagger documentation: https://localhost:3000/docs

User Guide:
A basic guide on how to use the API using Insomnia

1. Download and install Insomnia from https://insomnia.rest/download
2. Create a new request collection and then a new request within the project, select the HTTP request option

Search for movies by title:

1. Select GET request and enter the following URL: https://localhost:3000/movies/search
2. Enter a movie title and optionally the year and/or page number under parameters (no other parameters are allowed)
3. Click send for list of results



GET https://localhost:3000/movies/search

URL PREVIEW
https://localhost:3000/movies/search?title=Big&year=1998&page=1

QUERY PARAMETERS

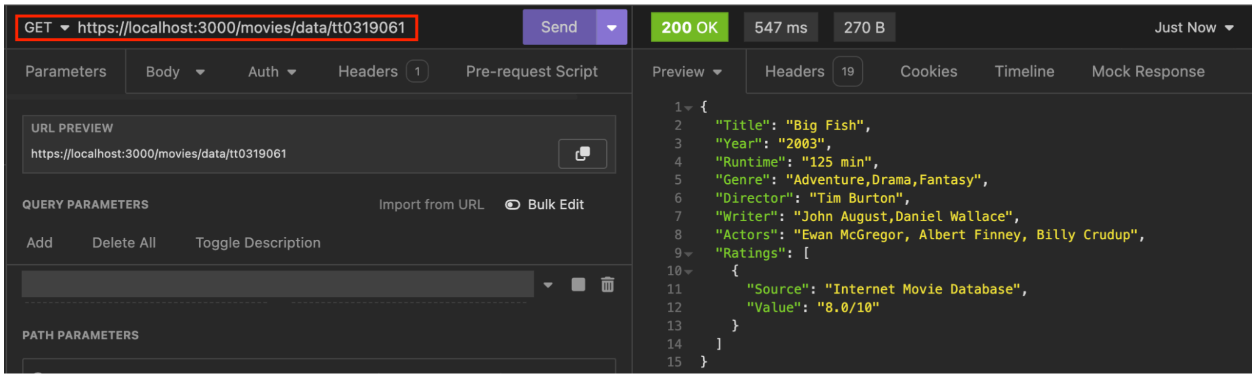
title	Big
year	1998
page	1

PATH PARAMETERS
Path parameters are url path segments that start with a colon ':' e.g. ':id'

```
{
  "data": [
    {
      "Title": "The Big Lebowski",
      "Year": "1998",
      "imdbID": "tt0118715",
      "Type": "movie"
    },
    {
      "Title": "The Big Hit",
      "Year": "1998",
      "imdbID": "tt0120609",
      "Type": "movie"
    }
  ],
  "pagination": {
    "total": 2,
    "lastPage": 1,
    "perPage": 100,
    "currentPage": 1,
    "from": 0,
    "to": 2
  }
}
```

Retrieve movie data:

1. Select GET request and enter the following URL with the movie IMDb ID: https://localhost:3000/movies/data/{imdbID}
2. Click send for detailed movie information



GET https://localhost:3000/movies/data/tt0319061

URL PREVIEW
https://localhost:3000/movies/data/tt0319061

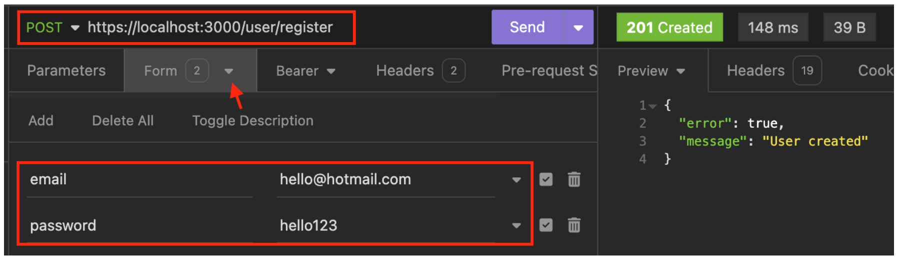
QUERY PARAMETERS

PATH PARAMETERS

```
{
  "Title": "Big Fish",
  "Year": "2003",
  "Runtime": "125 min",
  "Genre": "Adventure,Drama,Fantasy",
  "Director": "Tim Burton",
  "Writer": "John August,Daniel Wallace",
  "Actors": "Ewan McGregor, Albert Finney, Billy Crudup",
  "Ratings": [
    {
      "Source": "Internet Movie Database",
      "Value": "8.0/10"
    }
  ]
}
```

Register a new user:

1. Select POST request and enter the following URL: https://localhost:3000/user/register
2. Under ‘Form’ select ‘Form URL Encoded’ and enter email and password

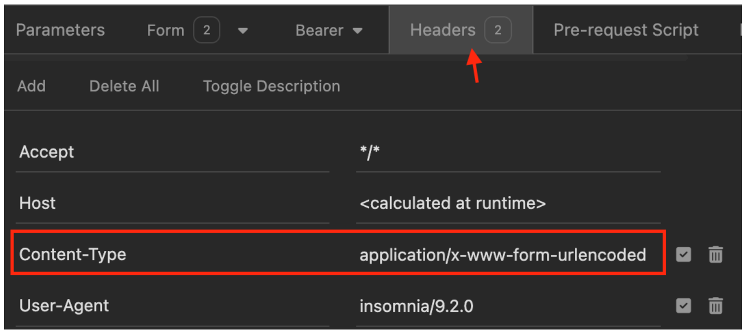


POST https://localhost:3000/user/register — 201 Created, 148 ms, 39 B

email	hello@hotmail.com
password	hello123

```
{
  "error": true,
  "message": "User created"
}
```

3. Select the ‘Header’ tab and add the following header:
   Content-Type: application/x-www-form-urlencoded

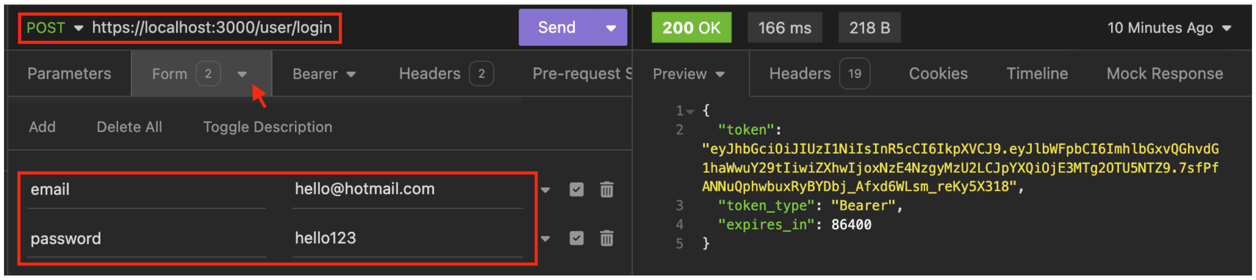


Accept	*/*
Host	<calculated at runtime>
Content-Type	application/x-www-form-urlencoded
User-Agent	insomnia/9.2.0

4. Click send for confirmation user has been created

Login to API for access to protected endpoints:

1. Select POST request and enter the following URL: https://localhost:3000/user/login
2. Enter same email and password in Form URL Encoded



POST https://localhost:3000/user/login — 200 OK, 166 ms, 218 B, 10 Minutes Ago

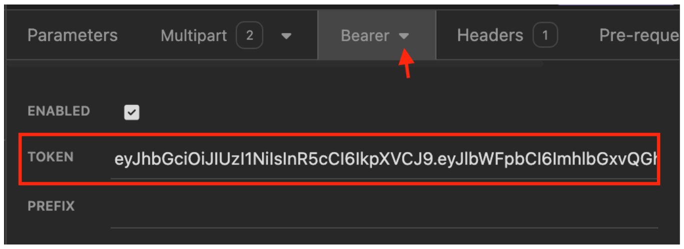
email	hello@hotmail.com
password	hello123

```
{
  "token":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImhlbGxvQGhvdG1haWwuY29tIiwiZXhwIjoxNzE4NzgyMzU2LCJpYXQiOjE3MTg2OTU5NTZ9.7sfPfANNuQphwbuxRyBYDbj_Afxd6WLsm_reKy5X318",
  "token_type": "Bearer",
  "expires_in": 86400
}
```

3. Ensure content-type is added to headers as shown in registration route:
   Content-Type: application/x-www-form-urlencoded
4. Click send to authenticate and receive JWT token

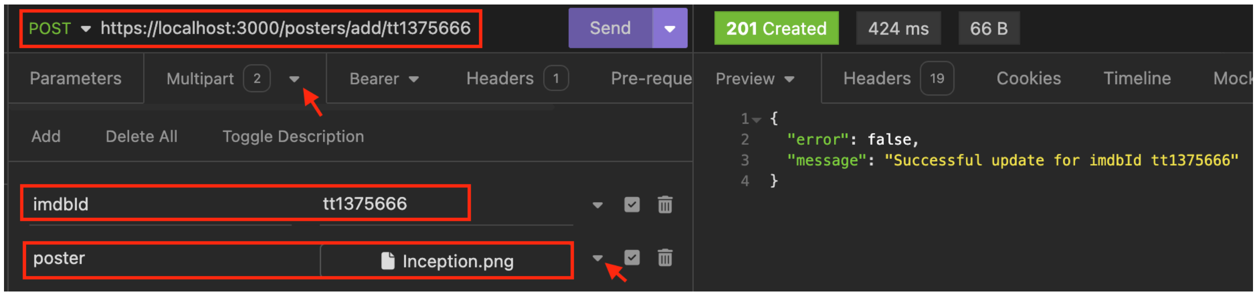
Upload poster to the database:

1. Select POST request and enter the following URL with the movie IMDb ID:
   https://localhost:3000/posters/add/{imdbID}
2. Select ‘Bearer Token’ and paste the token received from login route



ENABLED ✓
TOKEN eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImhlbGxvQGh
PREFIX

3. Select 'Multipart form', enter imdbID and select poster file (ensure file is selected instead of text)



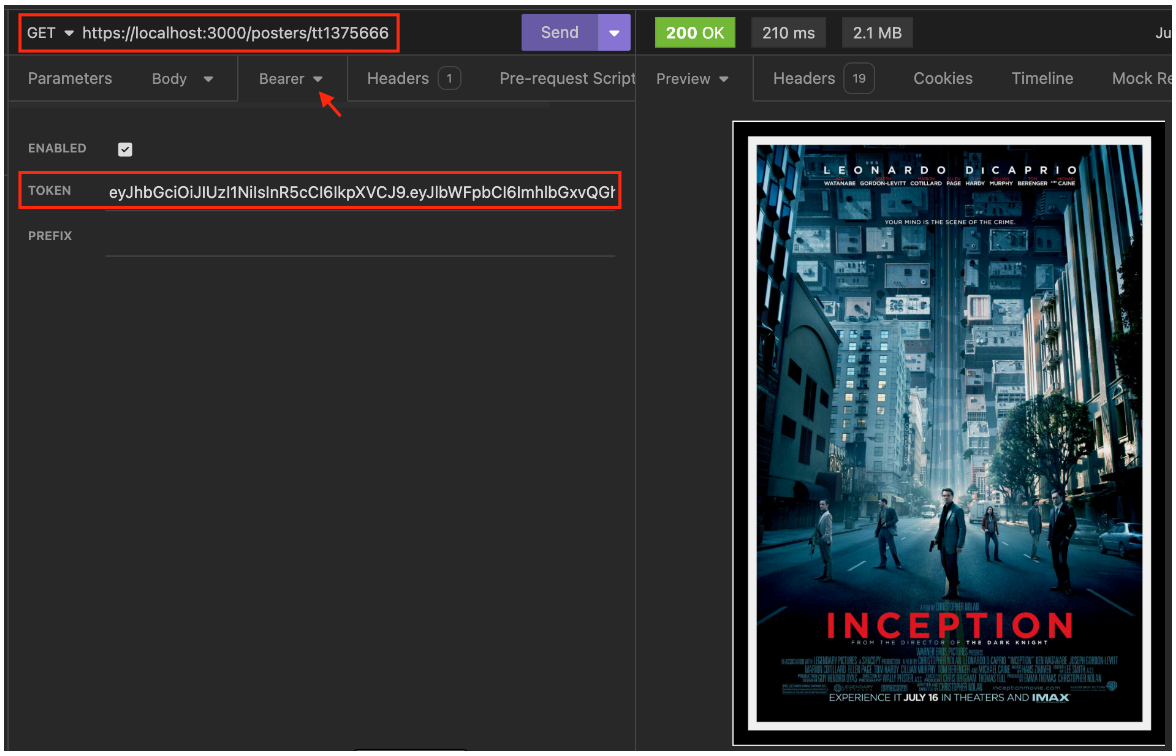
imdbId	tt1375666
poster	Inception.png

```
{
  "error": false,
  "message": "Successful update for imdbId tt1375666"
}
```

4. Click send for confirmation poster has been added to the database

Retrieve movie poster:

1. Select GET request and enter the following URL with the movie IMDb ID: https://localhost:3000/posters/{imdbID}


GET https://localhost:3000/posters/tt1375666
Send
200 OK
210 ms
2.1 MB
Parameters
Body
Bearer
Headers 1
Pre-request Script
Preview
Headers 19
Cookies
Timeline
LEONARDO DICAPRIO
YOUR MIND IS THE SCENE OF THE CRIME.
INCEPTION
FROM THE DIRECTOR OF THE DARK KNIGHT
EXPERIENCE IT JULY 16 IN THEATERS AND IMAX

ENABLED ✓
TOKEN eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImhlbGxvQGh
PREFIX

2. Select 'Bearer Token' and paste the token received from login route
3. Click send for the poster image