Report for Assignment 1: Node.js-based API

This project involved building a RESTful API using Node.js, designed to handle movie-related data from third-party sources. The API allows clients to search for movies by title, retrieve movie details by IMDb ID, and upload or download custom movie posters.

The API was structured around the following main functionalities:

- Fetching movie data via third-party APIs such as OMDb.
- Supporting poster uploads and retrieval.
- Managing API keys securely using environment variables.

Technical Description of the Application

API Architecture

The API was built using native Node.js (without Express.js) and listens on port 5000. It provides the following endpoints, as documented in the provided Swagger interface:

1. **GET /movies/search/{title}**: Search for a movie by its title.
2. **GET /movies/data/{imdbID}**: Fetch detailed information about a movie using its IMDb ID.
3. **GET /posters/{imdbID}**: Retrieve the custom movie poster for a specific movie.
4. **POST /posters/add/{imdbID}**: Upload a custom movie poster for a specific movie.

Environment Variables

Sensitive information, like API keys, is stored in a .env file and accessed using the ".env" package. This helps ensure that keys are not hardcoded in the codebase.

```
OMDB_API_KEY=my_omdb_api_key
RAPID_API_KEY=my_rapidapi_key
```

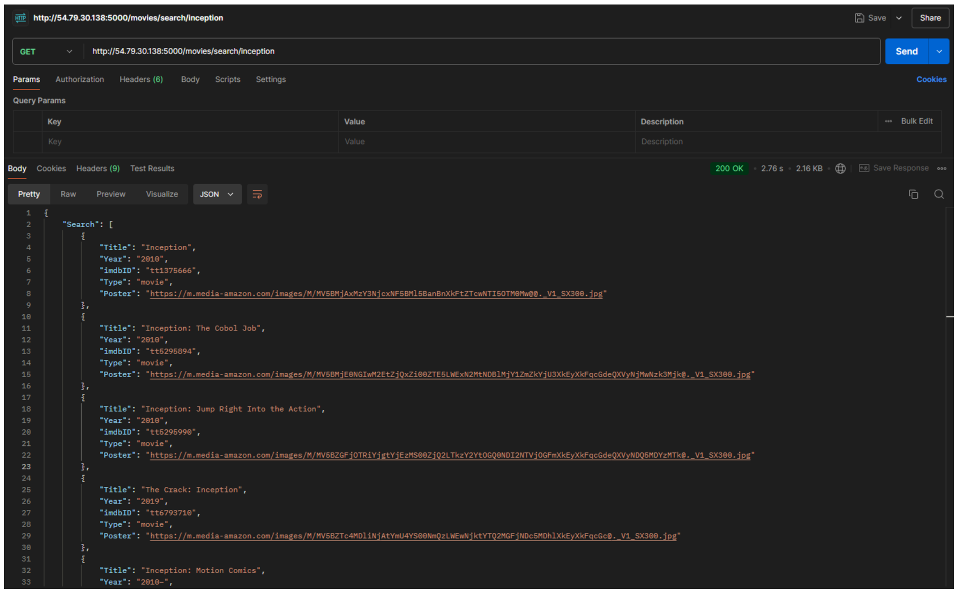
Error Handling

Error handling is implemented for missing query parameters, failed third-party requests, and file upload issues. Appropriate HTTP status codes are returned in each case (400, 404, or 500 depending on the error type).

API Endpoints and Their Implementation

1.1 GET /movies/search/{title} Successful Request

This endpoint searches for a movie by its title, returning relevant details from IMDb.



GET http://54.79.30.138:5000/movies/search/inception

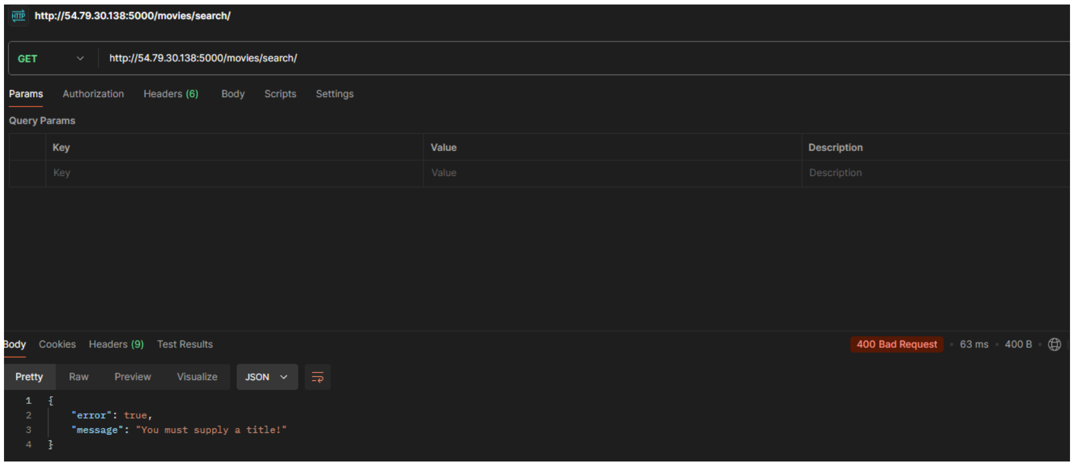
200 OK · 2.76 s · 2.16 KB

```
{
    "Search": [
        {
            "Title": "Inception",
            "Year": "2010",
            "imdbID": "tt1375666",
            "Type": "movie",
            "Poster": "https://m.media-amazon.com/images/M/MV5BMjAxMzY3NjcxNF5BMl5BanBnXkFtZTcwNTI5OTM0Mw@@._V1_SX300.jpg"
        },
        {
            "Title": "Inception: The Cobol Job",
            "Year": "2010",
            "imdbID": "tt5295894",
            "Type": "movie",
            "Poster": "https://m.media-amazon.com/images/M/MV5BMjE0NGIwM2EtZjQxZi00ZTE5LWExN2MtNDBlMjY1ZmZkYjU3XkEyXkFqcGdeQXVyNjMwNzk3Mjk@._V1_SX300.jpg"
        },
        {
            "Title": "Inception: Jump Right Into the Action",
            "Year": "2010",
            "imdbID": "tt5295990",
            "Type": "movie",
            "Poster": "https://m.media-amazon.com/images/M/MV5BZGFjOTRiYjgtYjEzMS00ZjQ2LTkzY2YtOGQ0NDI2NTVjOGFmXkEyXkFqcGdeQXVyNDQ5MDYzMTk@._V1_SX300.jpg"
        },
        {
            "Title": "The Crack: Inception",
            "Year": "2019",
            "imdbID": "tt6793710",
            "Type": "movie",
            "Poster": "https://m.media-amazon.com/images/M/MV5BZTc4MDliNjAtYmU4YS00NmQzLWEwNjktYTQ2MGFjNDc5MDhlXkEyXkFqcGc@._V1_SX300.jpg"
        },
        {
            "Title": "Inception: Motion Comics",
            "Year": "2010–",
```

Example Request - GET /movies/search/Inception

1.2 GET /movies/search/{no title entered}

If no title is provided, the API responds with a 400 status



GET http://54.79.30.138:5000/movies/search/

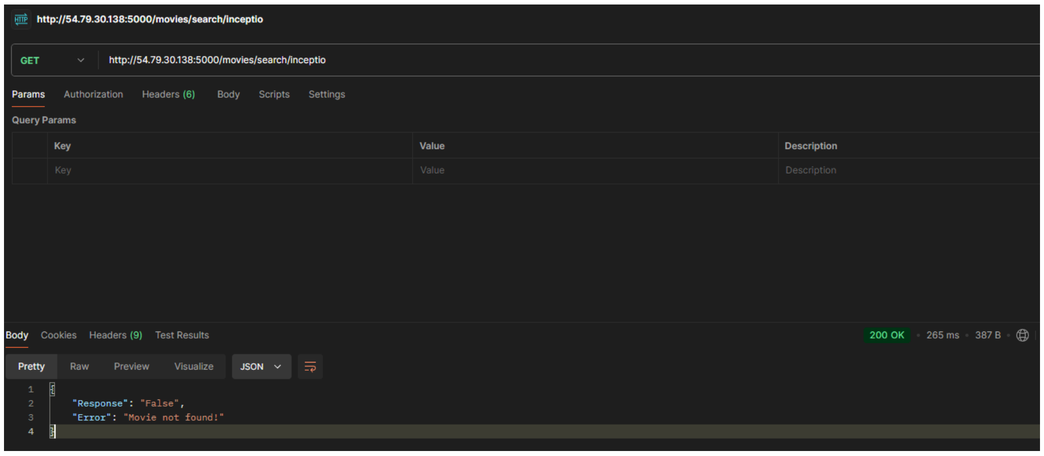
400 Bad Request · 63 ms · 400 B

```
{
    "error": true,
    "message": "You must supply a title!"
}
```

Example Request: GET /movies/search/

1.3 GET /movies/search/{incorrect title}

If the title is incorrect or no matches are found, the API returns **404**.



http://54.79.30.138:5000/movies/search/inceptio

GET http://54.79.30.138:5000/movies/search/inceptio

Params Authorization Headers (6) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (9) Test Results — 200 OK · 265 ms · 387 B

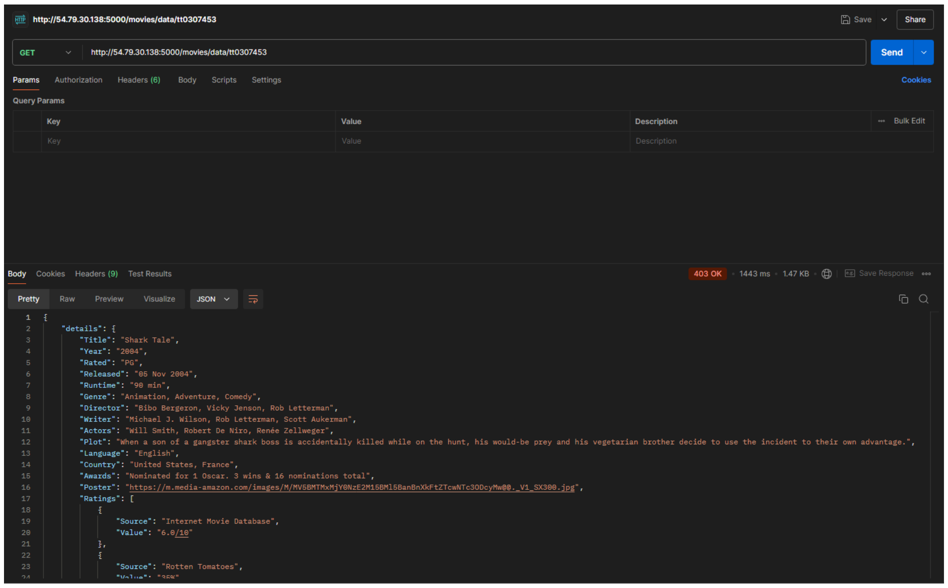
Pretty Raw Preview Visualize JSON

```
{
    "Response": "False",
    "Error": "Movie not found!"
}
```

Example Request: GET /movies/search/Inceptio

2.1 GET /movies/data/{imdbID} - Successful Request

This endpoint retrieves detailed data for a specific movie using its IMDb ID.



http://54.79.30.138:5000/movies/data/tt0307453

GET http://54.79.30.138:5000/movies/data/tt0307453 Send

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (9) Test Results — 403 OK · 1443 ms · 1.47 KB · Save Response

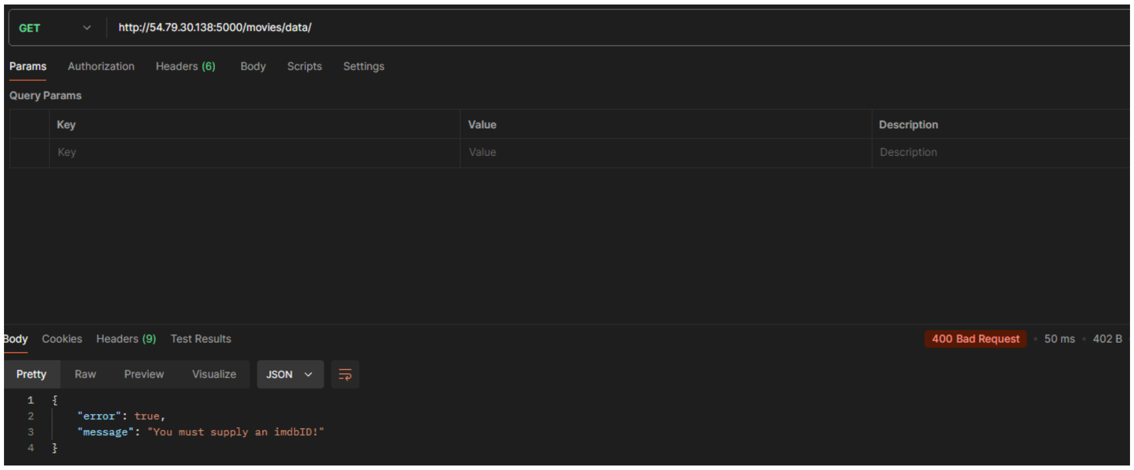
Pretty Raw Preview Visualize JSON

```
{
    "details": {
        "Title": "Shark Tale",
        "Year": "2004",
        "Rated": "PG",
        "Released": "05 Nov 2004",
        "Runtime": "90 min",
        "Genre": "Animation, Adventure, Comedy",
        "Director": "Bibo Bergeron, Vicky Jenson, Rob Letterman",
        "Writer": "Michael J. Wilson, Rob Letterman, Scott Aukerman",
        "Actors": "Will Smith, Robert De Niro, Renée Zellweger",
        "Plot": "When a son of a gangster shark boss is accidentally killed while on the hunt, his would-be prey and his vegetarian brother decide to use the incident to their own advantage.",
        "Language": "English",
        "Country": "United States, France",
        "Awards": "Nominated for 1 Oscar. 3 wins & 16 nominations total",
        "Poster": "https://m.media-amazon.com/images/M/MV5BMTMxMjY0NzE2M15BMl5BanBnXkFtZTcwNTc3ODcyMw@@._V1_SX300.jpg",
        "Ratings": [
            {
                "Source": "Internet Movie Database",
                "Value": "6.0/10"
            },
            {
                "Source": "Rotten Tomatoes",
                "Value": "35%"
```

Example Request - (Shark Tale): GET /movies/data/tt0307453

2.2 GET /movies/data/{imdbID} - No IMDb ID provided

This endpoint handles missing IMDb ID and responds with a 400 error.


GET http://54.79.30.138:5000/movies/data/

Query Params

Key	Value	Description

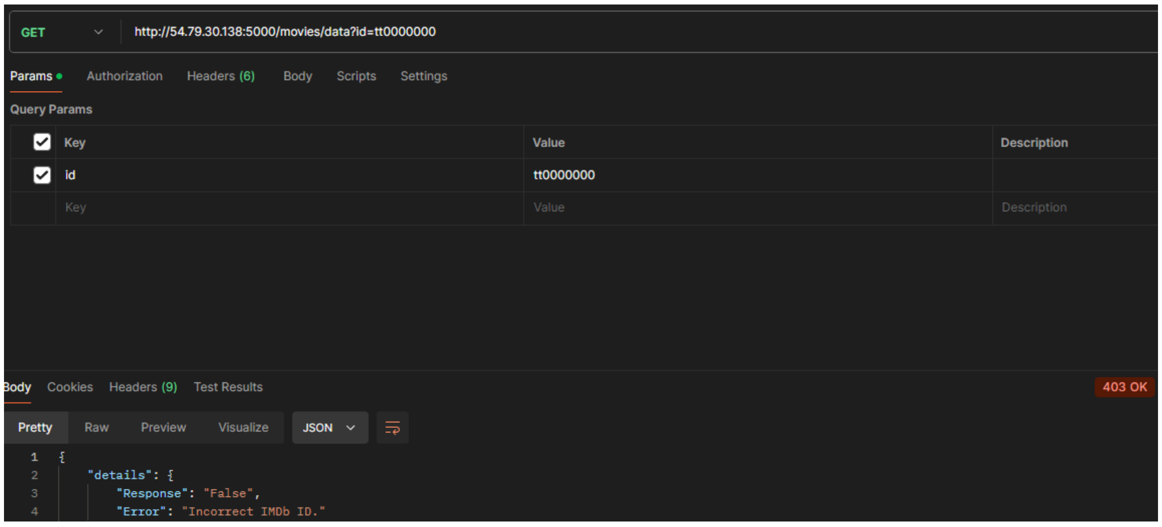
400 Bad Request · 50 ms · 402 B

```
{
    "error": true,
    "message": "You must supply an imdbID!"
}
```

Example Request: GET /movies/data/

2.3 GET /movies/data/{invalid imdbID} - Invalid IMDb ID

This endpoint handles an invalid IMDb ID and responds with a 403 OK error.


GET http://54.79.30.138:5000/movies/data?id=tt0000000

Query Params

Key	Value	Description
id	tt0000000	

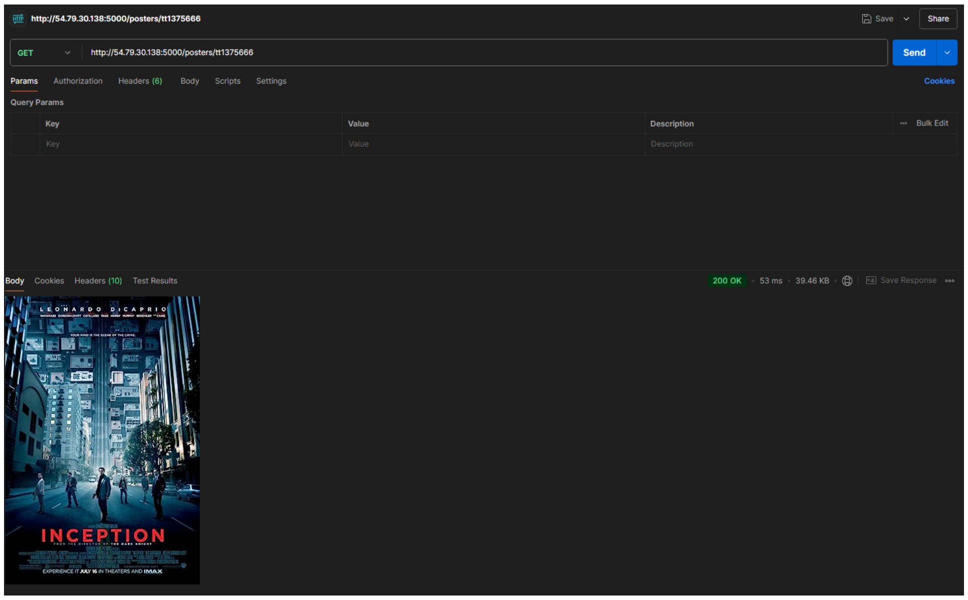
403 OK

```
{
    "details": {
        "Response": "False",
        "Error": "Incorrect IMDb ID."
```

Example Request: GET /movies/data/tt0000000

3.1 GET /posters/{imdbID}

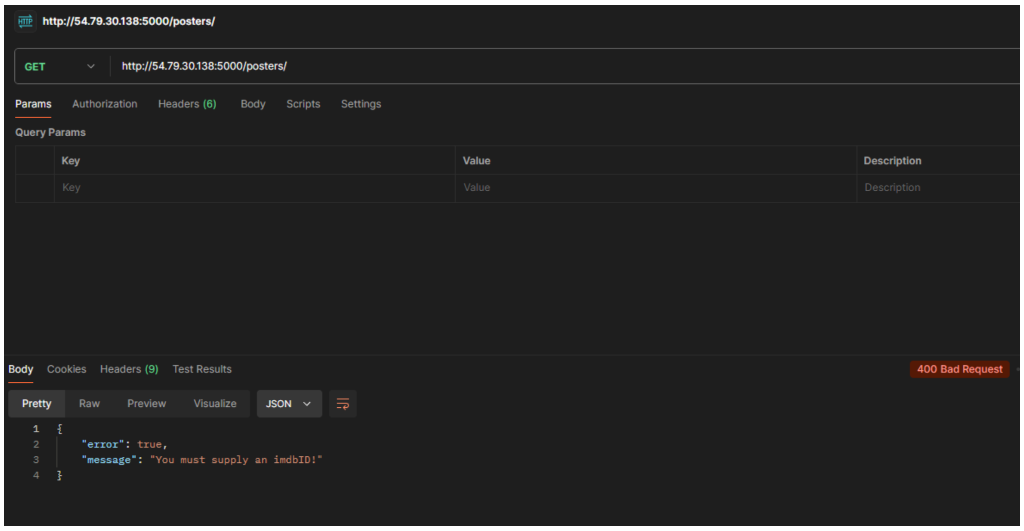
This endpoint retrieves a custom poster for a specific movie, identified by its IMDb ID.



Example Request: GET /posters/tt1375666

3.2 GET /posters/ - no IMDb ID

This endpoint handles missing IMDb ID and returns an error.

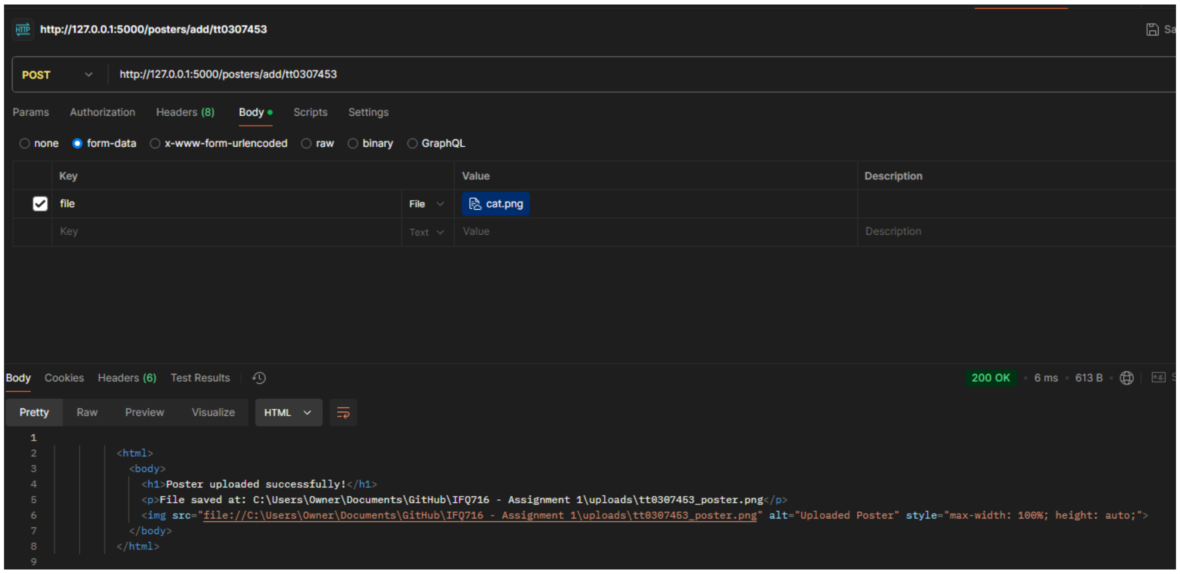


```
{
    "error": true,
    "message": "You must supply an imdbID!"
}
```

Example Request: GET /posters/

4.1 POST /posters/add/{imdbID}

This endpoint allows users to upload a custom movie poster for a specific movie, identified by its IMDb ID. The poster file is uploaded as form-data.



POST http://127.0.0.1:5000/posters/add/tt0307453

Body: form-data

Key		Value	Description
✓ file	File	cat.png	

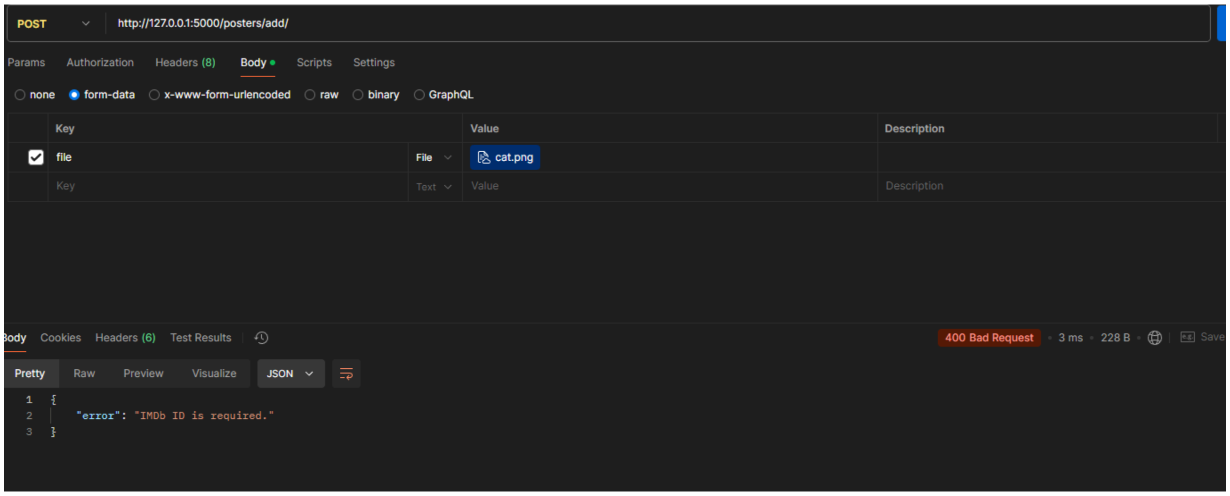
200 OK · 6 ms · 613 B

```
<html>
  <body>
    <h1>Poster uploaded successfully!</h1>
    <p>File saved at: C:\Users\Owner\Documents\GitHub\IFQ716 - Assignment 1\uploads\tt0307453_poster.png</p>
    <img src="file://C:\Users\Owner\Documents\GitHub\IFQ716 - Assignment 1\uploads\tt0307453_poster.png" alt="Uploaded Poster" style="max-width: 100%; height: auto;">
  </body>
</html>
```

Example Request: POST /posters/add/tt0307453, with an image file uploaded as form-data under the key file.

4.2 POST /posters/add/ - No IMDb ID provided

This endpoint handles missing IMDb ID and responds with a 400 error.



POST http://127.0.0.1:5000/posters/add/

Body: form-data

Key		Value	Description
✓ file	File	cat.png	

400 Bad Request · 3 ms · 228 B

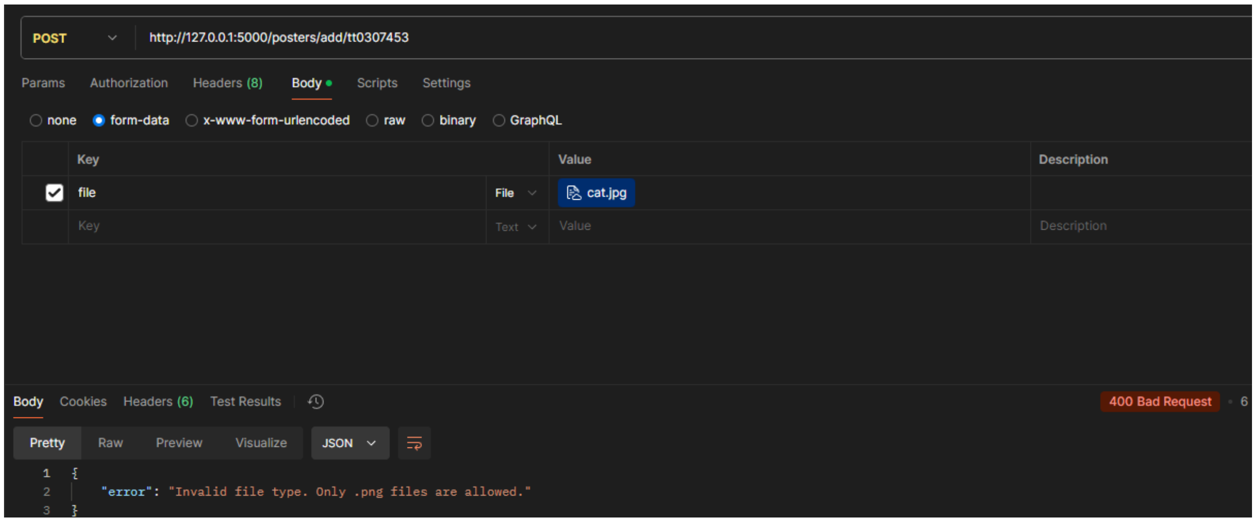
```
{
    "error": "IMDb ID is required."
}
```

Example Request: POST /posters/add/, with an image file uploaded as form-data under the key file.

4.3 POST /posters/add/{imdbID} - No .png provided

This endpoint handles the incorrect file uploaded with a 400 error message.

Example Request: POST /posters/add/tt0307453, with an image file uploaded as form-data under the key file that is not a .png file


POST http://127.0.0.1:5000/posters/add/tt0307453

Body: form-data

Key		Value	Description
file	File	cat.jpg	
Key	Text	Value	Description

400 Bad Request

```
{
    "error": "Invalid file type. Only .png files are allowed."
}
```

Testing and Errors Encountered

Error 1: 500 Internal Server Error for Movie Search

- **What happened**: While testing the /movies/search/{title} endpoint with the movie *Inception*, I received a 500 Internal Server Error.
- **Cause**: The error was due to an invalid OMDb API key being used. The .env file wasn't set up correctly, causing the API key to be undefined.
- **Solution**: After fixing the .env file and ensuring the API key was properly loaded, the issue was resolved.

Error 2: 401 Unauthorized for Movie Details by IMDb ID

- **What happened**: When testing the /movies/data/{imdbID} endpoint, I encountered a 401 Unauthorized error from OMDb.
- **Cause**: This was due to using an expired or invalid API key for OMDb.

- **Solution**: After generating a new OMDb API key and updating the .env file, the error was resolved.

Error 3: Missing Parameter in Poster Upload

- **What happened**: During the testing of /posters/add/{imdbID}, the server returned a 400 Bad Request error when I forgot to include the required IMDb ID in the URL.
- **Solution**: I updated the error handling to properly catch missing parameters and return a meaningful error message to the client, explaining what was missing.

Error 4: Unsupported File Type for Poster Upload

- **What happened**: While testing the /posters/add/{imdbID} endpoint, I attempted to upload a file with an unsupported format (e.g., .txt or .gif) instead of an image.
- **Cause**: The server was not validating the file type, so it accepted all file types and failed silently.
- **Solution**: I added file type validation to ensure that only .png files could be uploaded. If an invalid file type was uploaded, the server responded with errors.

Error 5: File Upload Size Limitation

- **What happened**: When trying to upload a large movie poster, the upload failed because the server wasn't configured to handle large file uploads.
- **Solution**: I configured the fs module to increase the buffer size and handle larger uploads successfully.

Error 6: CORS Issues

- **What happened**: While testing from a different client application, I ran into CORS issues that blocked the requests.
- **Solution**: I added the necessary CORS headers to the server responses to allow cross-origin requests

Appendix: Installation Guide

Prerequisites:

- Node.js (v14 or higher)
- npm (Node Package Manager)
- API keys for OMDb and RapidAPI