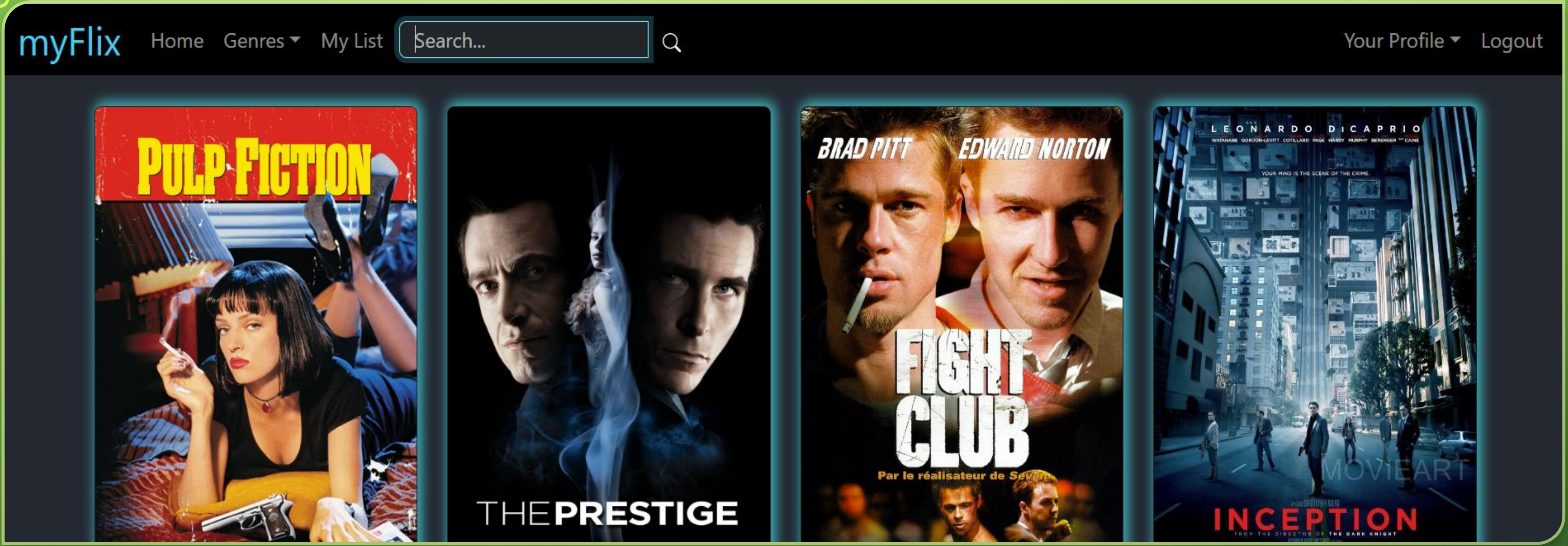


CASE STUDY – MOVIE_API

ALINA LEINWEBER

OVERVIEW – PART 1

PURPOSE AND CONTEXT

- The Movie_API was built using the MERN stack (MongoDB, Express) to serve as the back-end for the myFlix app, developed in both React and Angular versions. Together, they form a full-stack application.
- The challenge was to create the API and database from scratch while ensuring a seamless connection between them.
- I aimed to gain in-depth experience in building full-stack applications through this project.

Go To Project On GitHub

PROJECT DURATION

- It took about **four weeks** to complete this project.
- Each week focused on a different stage: setting up the server, creating the database, implementing API functionality, and finally, deploying to the cloud.
- Minor delays occurred during the authentication phase and with Postman tests.

OVERVIEW – PART 2

OBJECTIVE

- The specific goal of this project was to develop a robust RESTful API that provides movie fans with detailed film information, while keeping all user data safe.
- Users should be able to sign up, update their profiles, create lists of favorite movies, and get deep insights into the movies and directors they love.
- The focus was on creating a back-end that performs efficiently and meets security standards.

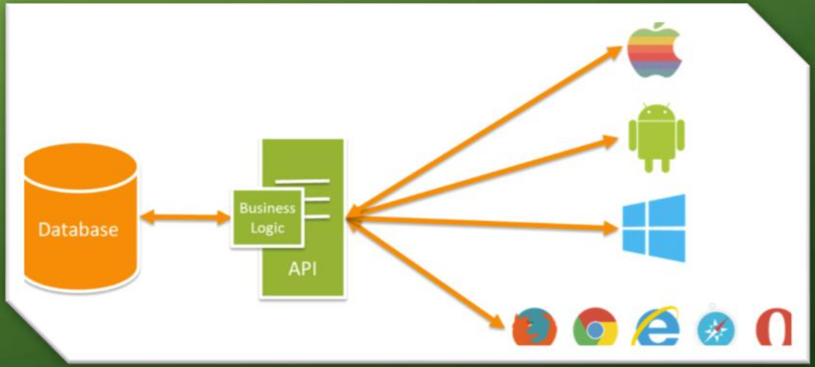
Go To Project On GitHub

TOOLS

- The code is written using Node.js and Express.js
- The application follows RESTful API design
- MongoDB is used as the database
- Authentication is implemented with JWT (JSON Web Token)
- Postman is used for testing

WHAT IS A REST API?

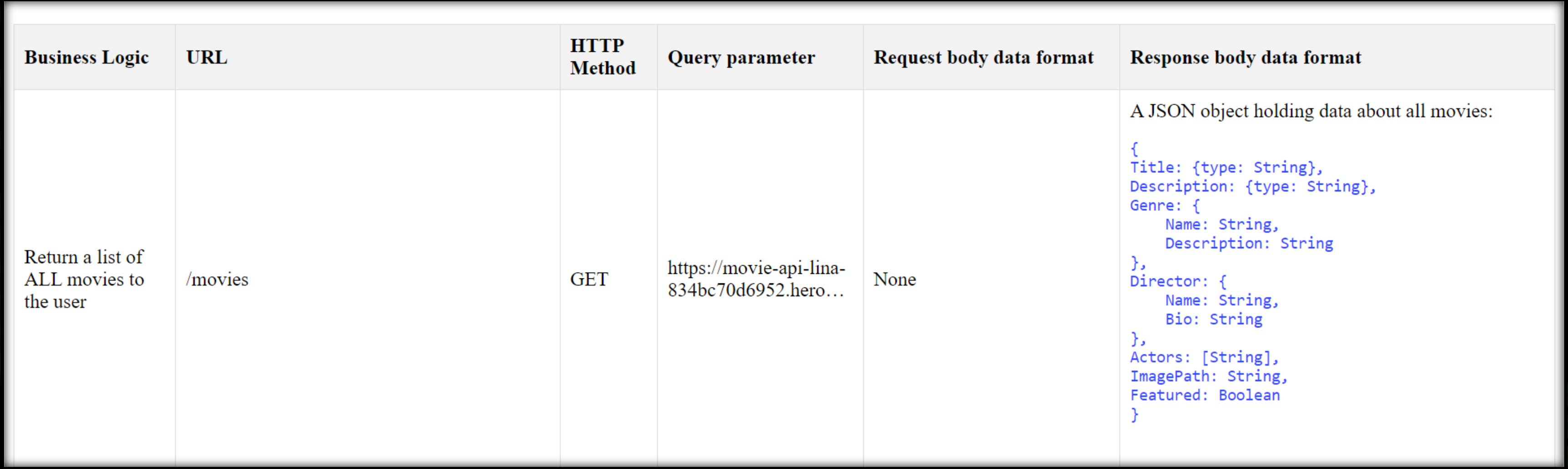
- A REST API is like a messenger between two different systems that lets them communicate.
- Example: When a user clicks "Show my favorite movies" in the app, the app requests the data from the API, which sends the data back in a format the app can display as the user's favorite movies.
- REST API endpoints often look like URLs starting with http:// or https://. Here the API was used to access and manage movie and user data.



CRUD

CRUD stands for Create, Read, Update, and Delete. These four operations allow data to be added, viewed, changed, or removed from the database in software applications. This is used when building API endpoints.

This API endpoint, that looks like an URL, allows users to list all the movies stored in the database by displaying them in a specific format. On the right side we see that the data will be displayed in JSON format.

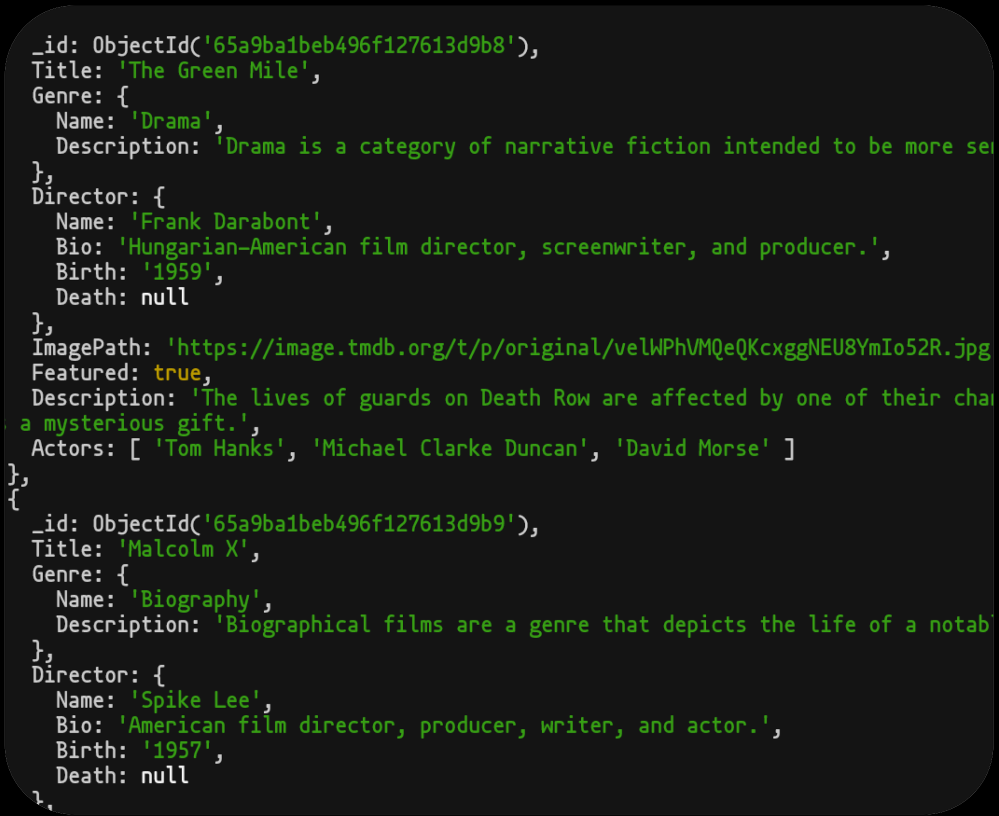


Business Logic	URL	HTTP Method	Query parameter	Request body data format	Response body data format

Return a list of ALL movies to the user	/movies	GET	https://movie-api-lina-834bc70d6952.hero…	None	A JSON object holding data about all movies:
					`{`
					`Title: {type: String},`
					`Description: {type: String},`
					`Genre: {`
					`    Name: String,`
					`    Description: String`
					`},`
					`Director: {`
					`    Name: String,`
					`    Bio: String`
					`},`
					`Actors: [String],`
					`ImagePath: String,`
					`Featured: Boolean`
					`}`

MONGODB DATABASE

- In a software application, a database is used to store all kinds of information, like product catalogs.
- A MongoDB database stores this data in an easy-to-read format, similar to a list, but slightly more complex.
- In this application, the MongoDB database was used to store information about different movies as well as user login data.



```
  _id: ObjectId('65a9ba1beb496f127613d9b8'),
  Title: 'The Green Mile',
  Genre: {
    Name: 'Drama',
    Description: 'Drama is a category of narrative fiction intended to be more se
  },
  Director: {
    Name: 'Frank Darabont',
    Bio: 'Hungarian-American film director, screenwriter, and producer.',
    Birth: '1959',
    Death: null
  },
  ImagePath: 'https://image.tmdb.org/t/p/original/velWPhVMQeQKcxggNEU8YmIo52R.jpg
  Featured: true,
  Description: 'The lives of guards on Death Row are affected by one of their cha
 a mysterious gift.',
  Actors: [ 'Tom Hanks', 'Michael Clarke Duncan', 'David Morse' ]
},
{
  _id: ObjectId('65a9ba1beb496f127613d9b9'),
  Title: 'Malcolm X',
  Genre: {
    Name: 'Biography',
    Description: 'Biographical films are a genre that depicts the life of a notab
  },
  Director: {
    Name: 'Spike Lee',
    Bio: 'American film director, producer, writer, and actor.',
    Birth: '1957',
    Death: null
  },
```

AUTHENTICATION & AUTHORIZATION

AUTHENTICATION

- For the API, I used basic HTTP authentication. This means the app sends the username and password securely in the HTTP header.
- The API then checks if the details are correct in the database, and if they match, the user gets logged in successfully.

AUTHORIZATION

- If the username and password are correct, the app creates a token.
- This token is sent automatically with every request, and only with the correct token can the user access the app.
- Tokens are long and complex, making them extremely difficult for hackers to find out.

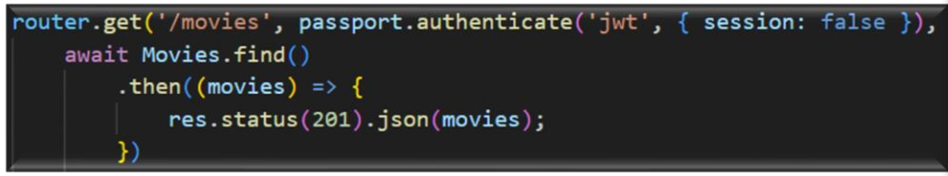
SECURITY

- The movie API uses password hashing. This means the password is changed into a secret code before being saved in the database.
- So, even if someone hacks the database, they can't see the real password—only the user knows it.

TESTING AND DEBUGGING WITH POSTMAN

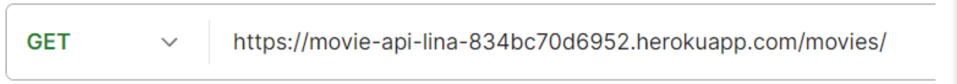
- Postman is a tool that helps developers test how their web servers handle requests and responses.
- It's like a digital assistant that helps you make sure everything is communicating properly.
- I used Postman to ensure that all endpoints worked correctly and fetched/stored the right data in the corresponding database, which hosts all user and movie information.

Snippet of code for the movies endpoint



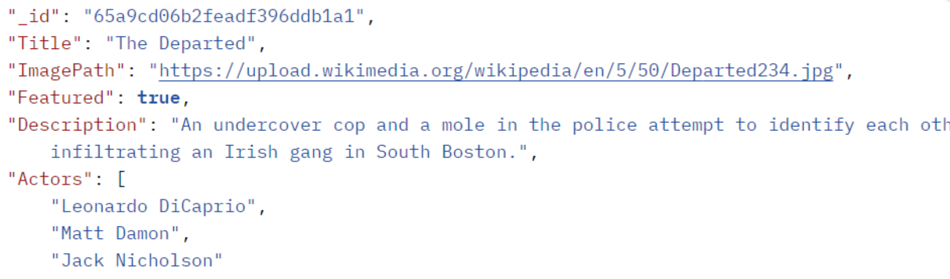
```
router.get('/movies', passport.authenticate('jwt', { session: false }),
    await Movies.find()
        .then((movies) => {
            res.status(201).json(movies);
        })
```

Endpoint URL to list all movies from the database



```
GET    https://movie-api-lina-834bc70d6952.herokuapp.com/movies/
```

Snippet of the returned data from the database



```
"_id": "65a9cd06b2feadf396ddb1a1",
"Title": "The Departed",
"ImagePath": "https://upload.wikimedia.org/wikipedia/en/5/50/Departed234.jpg",
"Featured": true,
"Description": "An undercover cop and a mole in the police attempt to identify each oth
    infiltrating an Irish gang in South Boston.",
"Actors": [
    "Leonardo DiCaprio",
    "Matt Damon",
    "Jack Nicholson"
```

CHALLENGES, SOLUTIONS, BEST PRACTICES

When I started testing the endpoints with Postman, it became clear that my endpoints needed further adjustments to ensure they functioned as intended, which initiated the debugging process. A lot of the bugs been related to authentication, authorization and hashing the password.

For debugging, the development console and the console.log command became my best friends. I implemented several statements that showcased through an endpoint exactly what was working and where the process got stuck. The try/catch approach was also very helpful in identifying and resolving issues.

I learned to consider and address specific use cases, such as when a user tries to change details for another user. This was resolved by implementing a check: if(req.user.Username !== req.params.Username){...}

When connecting the API to my Angular front end, I had to adjust some of the responses sent to Angular. Responses that worked for React caused errors in Angular, requiring modifications.

Another valuable lesson was ensuring that when users update their details, there is a mechanism in place to prevent empty fields from being added to the database. Instead, fields that are left blank should be ignored.

CORS and connection between the database and the endpoints on the other hand, went smoothly.

Looking back, I learned a lot about creating secure and reliable APIs during this project.

The biggest lesson? Good error handling is a game-changer!

Additionally, paying attention to the smallest details helps overall avoid bugs in the first place.

On top of that, I gained a deep understanding of the overall process of building an API from scratch, and I really enjoyed the experience, including problem-solving and debugging, reminds me of detective work.

In the future, I'd love to add a feature for users to reset their passwords, as this is common practice nowadays.

I'd also like to allow users to log in with OAuth, as most of us have a Facebook or Google account, which would make the overall user experience smoother.

Additionally, I would add features that allow users to leave comments on movies and add ratings, so users can see the overall ratings and rate the movies themselves.

FINAL CONCLUSIONS