



MOVIE_API

CASE STUDY

Laure Lincker

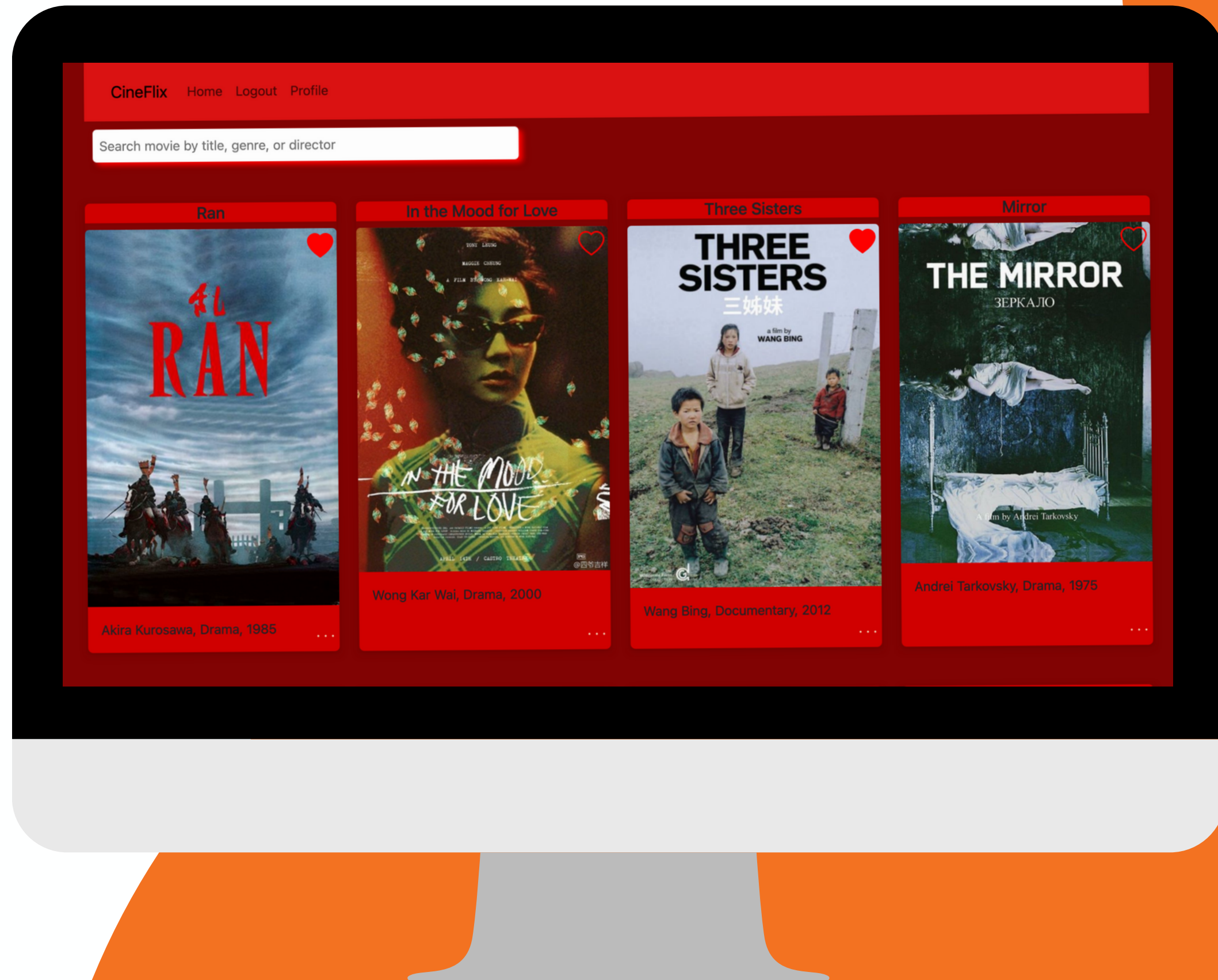
2 0 2 4

Overview

Movie_API is the server-side component of *Cineflix*. A responsive single-page application using the MERN stack, it allows users to find information about a collection of movies and bookmark them in a personal list after creating an account.

Purpose & Context

I have built this API as part of a full-stack project during my studies in web development with CareerFoundry in order to gain proficiency in Javascript development in both frontend and backend.



Objective

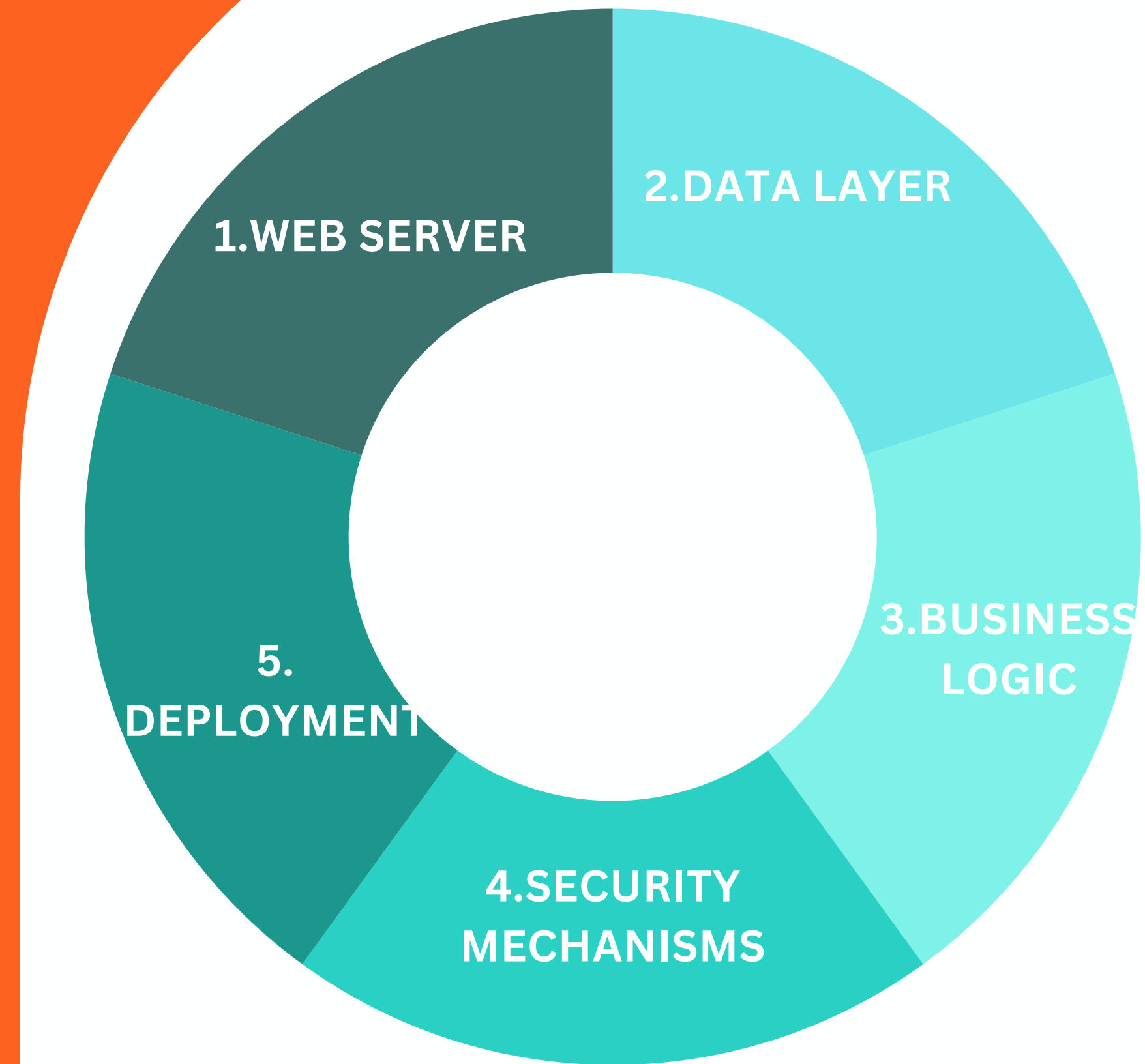
The project will demonstrate my ability to master back-end JavaScript development, including:

1. APIs
2. web server frameworks
3. databases
4. business logic
5. authentication
6. data security

Request	URL	HTTP Method	Request body data format	Response body data format
Get default text message	/	GET	none	String message
Add a new user	/users	POST	A JSON object holding user data, structured like this: { "Username": "Kim-Novak", "Password": "Vertigo123", "Email": "kim.novak@gmail.com", "Birthday": "13-02-1933" }	A JSON object holding user data plus an ID and an array for placing favourite movies: { "Username": "Kim-Novak", "Password": "\$2b\$10\$LUH.8iIEnXeWblWbrB.rd.LD/3rbkshj1Csdgek", "Email": "kim.novak@gmail.com", "Birthday": "13-02-1933", "FavouriteMovies": [], "_id": "d224bc3f-6174-481b-836f-e3a210fe6a45", "__v": 0 }
Login existing user	/users	GET	A JSON object holding user data, structured like this: { "Username": "Kim-Novak", "Password": "Vertigo123" }	A JSON object holding user data, an ID, a token and an array for placing favourite movies: "user": { "_id": "d224bc3f-6174-481b-836f-e3a210fe6a45", "Username": "Kim-Novak", "Password": "\$2b\$10\$LUH.8iIEnXeWblWbrB.rd.LD/3rbkshj1Csdgek", "Email": "kim.novak@gmail.com", "Birthday": "13-02-1933", "FavouriteMovies": [], "__v": 0 }, "token": "eyJBJVERJU6nbcjdhhHJGEFJBRJ.eyJFgarfKLNG4"
Get a list of ALL movies	/movies	GET	none	A JSON object holding an array with all movies.
Get data about a single movie by title	/movies/:Title	GET	None	A JSON object holding data about a single movie: "_id": "64f5d33c431c7b9fgh9plen34qq" "Title": "Ran", "Description": "A powerful but elderly warlord, decides to divide his kingdom among his three sons ultimately starting a fatal rivalry in their quest for power.", "ReleaseYear": 1985, "ImagePath": "www.critikat.com/wp-content/uploads/2016/04/artoff7462.jpg", "Featuredd": false, { "Genre": { "Name": "Drama", "Description": "Drama is a category or genre of narrative fiction (or semi-fiction) intended to be more serious than humorous in tone. " }, "Director": { "Name": "Akira Kurosawa", "Bio": "Kurosawa Akira was a Japanese filmmaker who directed 30 films in a career

Building Process

- 1**
 - Setup development using the terminal and Node.js syntax
 - Import Node modules
 - Create a “package.json” file
 - Import all necessary packages into the project directory
 - Define project dependencies
 - Route HTTP requests for the project using Express
 - Define the endpoints for the REST API
- 2**
 - Create a non-relational (NoSQL) database for storing movie and user data using MongoDB
- 3**
 - Model the business logic using Mongoose
- 4**
 - Implement authentication and authorization into the API using basic HTTP authentication and JWT (token-based) authentication
 - Incorporate data validation logic into the API
 - Implement data security and storage controls
- 5**
 - Host the project on the web using Netlify
 - Uploading the Database to MongoDB Atlas
 - Connecting the Database to the API



STEP 1: SERVER SIDE

The choice of Node



Rich library of JavaScript modules

These modules provide useful functionalities you can include when creating an API



Super Fast

Compiles and executes JavaScript at very fast speeds



Permissive Licence

Very limited restrictions on reuse and has, therefore, excellent licence compatibility

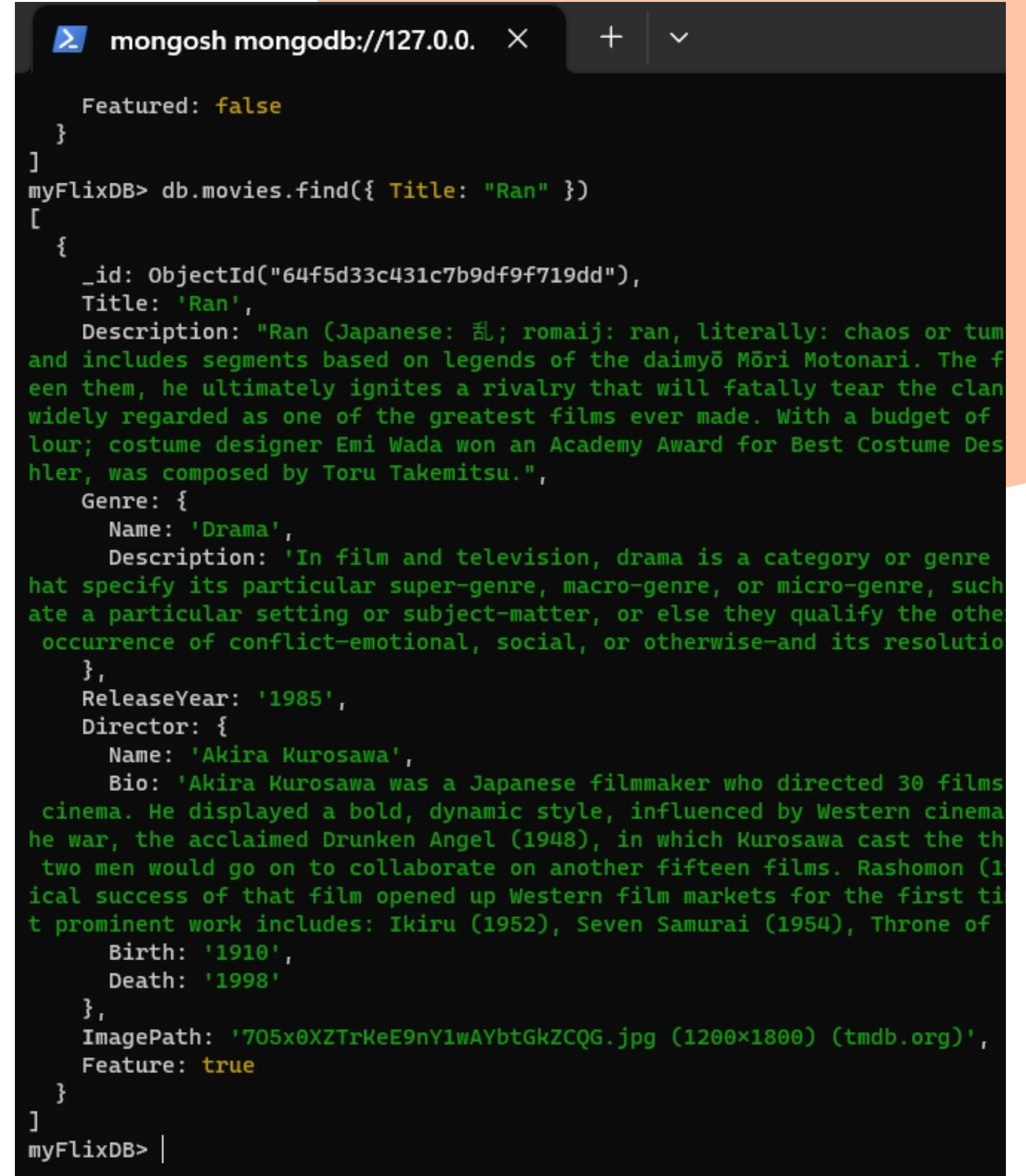
Node is well-suited for command line applications, single page applications, JSON-API-based applications such as my own project. It gives developers the advantage of being able to use a single language (JavaScript) for both the client- and server-side of web applications. For me, this meant to continue working with the coding language I mastered.

STEP 2 DATA LAYER

After building and experimenting on both a relational (SQL) database using PostgreSQL and a non-relational using MongoDB, I concluded that the flexibility of a non-relational database was more suitable to my project.

STEP 3 BUSINESS LOGIC

To ensure the data in the non-relational database remains consistent, I have used Mongoose which allows to enforce uniformity in data from the application-side. To do so I have created specific formats using models to follow. Mongoose then acts as a translator between the Node.js application and the MongoDB database layer.

A screenshot of a MongoDB terminal window. The title bar shows 'mongosh mongodb://127.0.0.1'. The terminal content shows a query 'db.movies.find({ Title: "Ran" })' being executed, returning a single document for the movie 'Ran'. The document includes fields like _id, Title, Description, Genre, ReleaseYear, Director, ImagePath, and Feature.

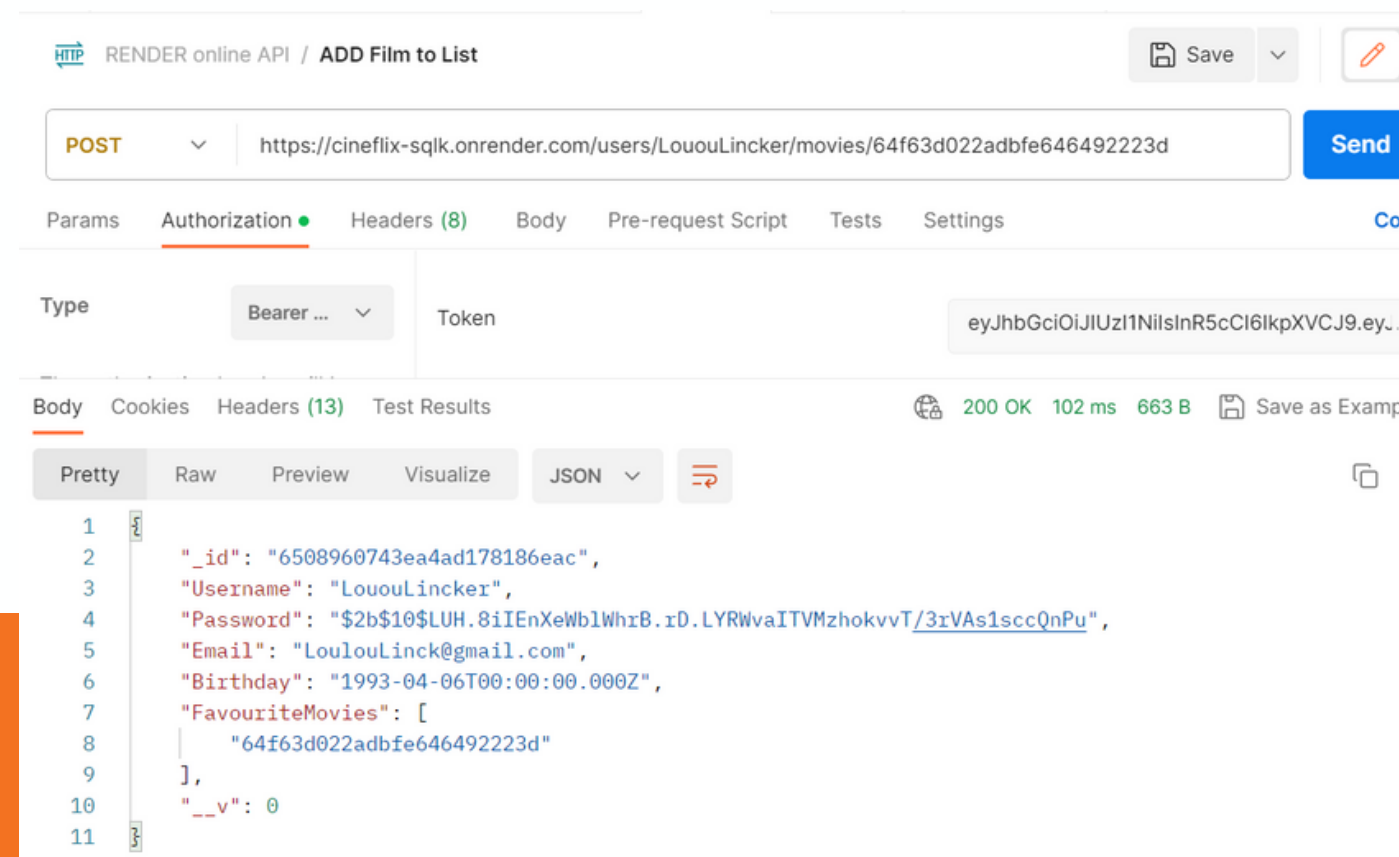
```
Featured: false
}
]
myFlixDB> db.movies.find({ Title: "Ran" })
[
  {
    _id: ObjectId("64f5d33c431c7b9df9f719dd"),
    Title: 'Ran',
    Description: "Ran (Japanese: 乱; romaij: ran, literally: chaos or tum
and includes segments based on legends of the daimyō Mōri Motonari. The f
een them, he ultimately ignites a rivalry that will fatally tear the clan
widely regarded as one of the greatest films ever made. With a budget of
lour; costume designer Emi Wada won an Academy Award for Best Costume Des
hler, was composed by Toru Takemitsu.",
    Genre: {
      Name: 'Drama',
      Description: 'In film and television, drama is a category or genre
hat specify its particular super-genre, macro-genre, or micro-genre, such
ate a particular setting or subject-matter, or else they qualify the othe
occurrence of conflict-emotional, social, or otherwise-and its resolutio
},
    ReleaseYear: '1985',
    Director: {
      Name: 'Akira Kurosawa',
      Bio: 'Akira Kurosawa was a Japanese filmmaker who directed 30 films
cinema. He displayed a bold, dynamic style, influenced by Western cinema
he war, the acclaimed Drunken Angel (1948), in which Kurosawa cast the th
two men would go on to collaborate on another fifteen films. Rashomon (1
ical success of that film opened up Western film markets for the first ti
t prominent work includes: Ikiru (1952), Seven Samurai (1954), Throne of
      Birth: '1910',
      Death: '1998'
    },
    ImagePath: '705x0XZTrKeE9nY1wAYbtGkZCQG.jpg (1200x1800) (tmdb.org)',
    Feature: true
  }
]
myFlixDB> |
```

Collections in MongoDB using CRUD

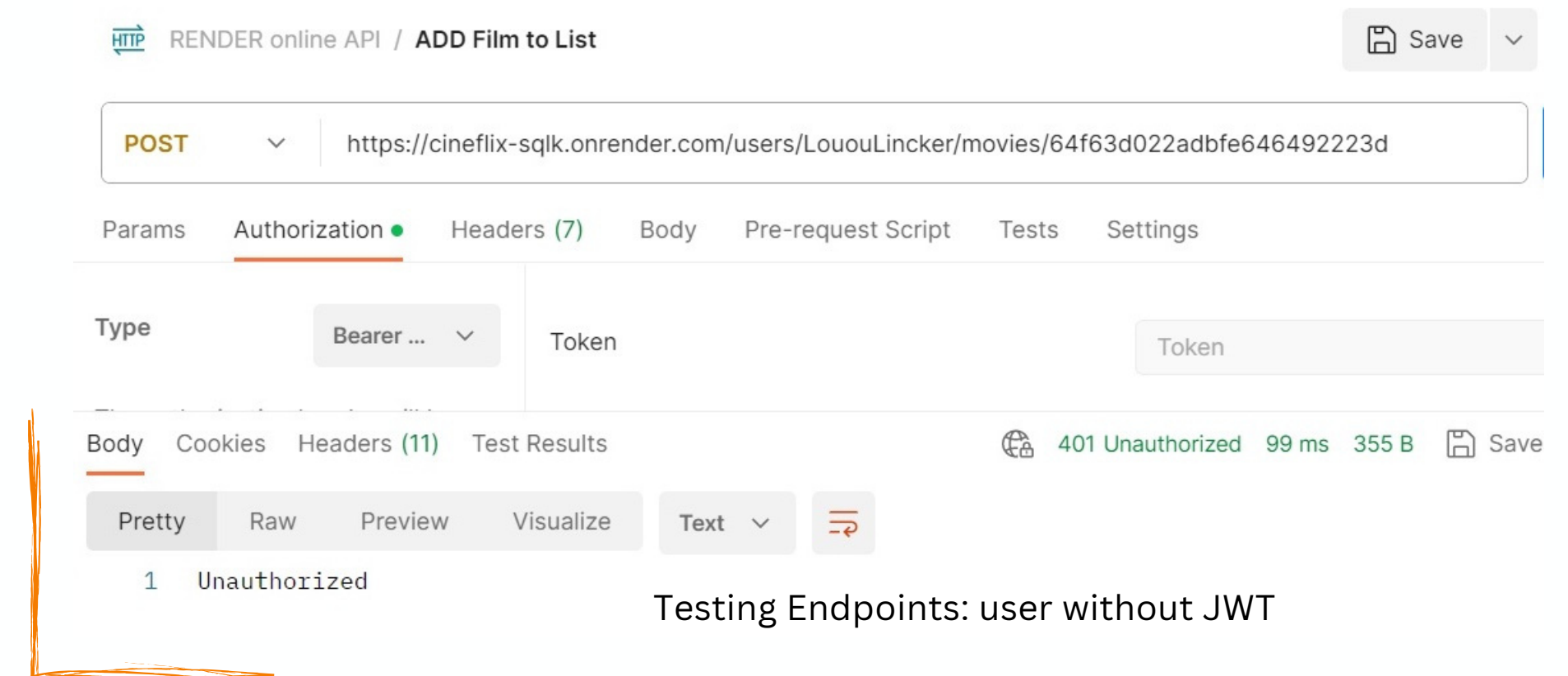
STEP 4 SECURITY MECHANISMS

Authentication & Authorisation

After exploring authentication and authorisation methods, I implemented basic HTTP authentication for initial login requests and JWT authentication for future API requests using the Passport middleware. Lastly, I tested these methods using Postman.



Testing Endpoints: user with JWT



Testing Endpoints: user without JWT

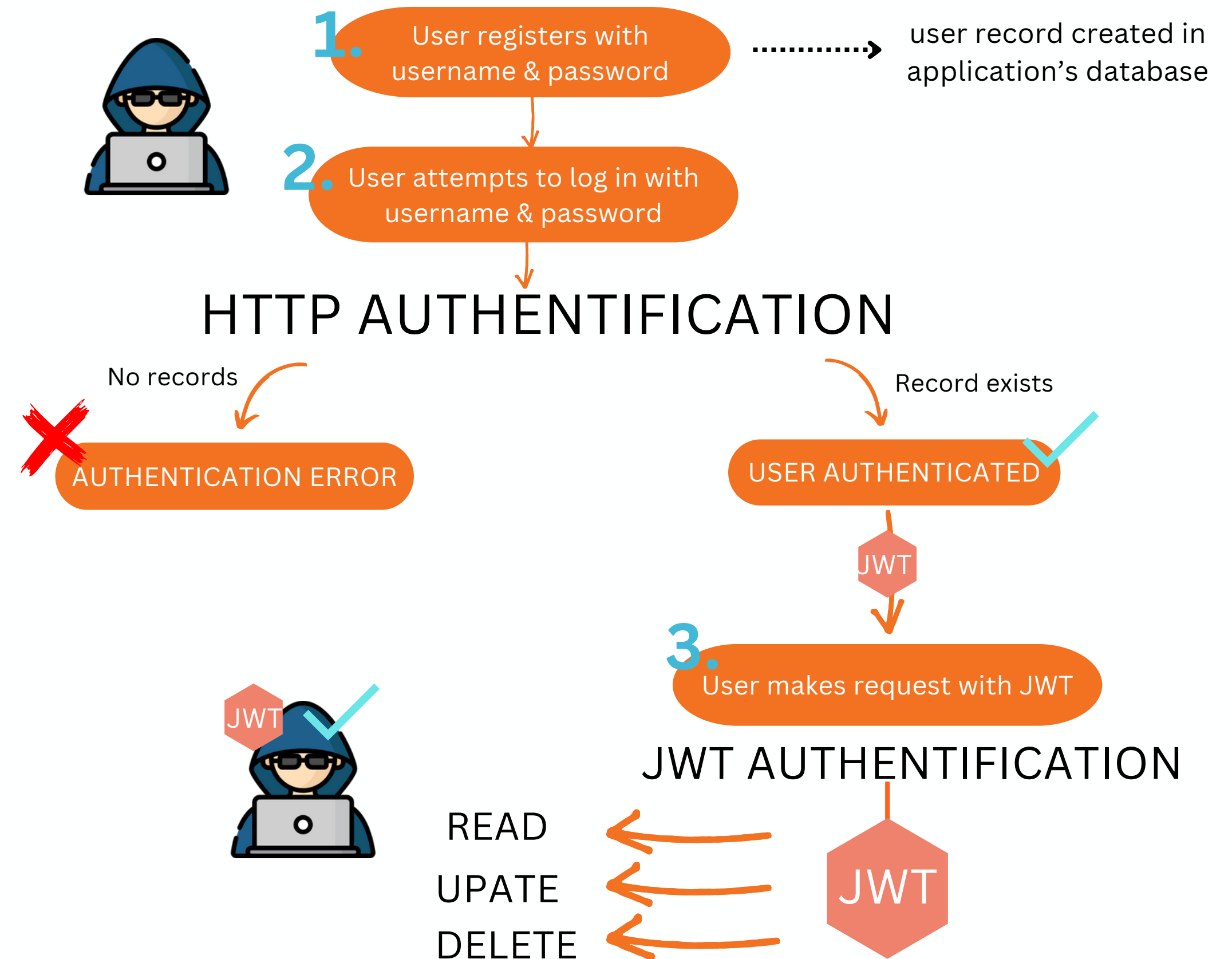
Authentication is the process of verifying a client's identity, whereas authorisation is the process of verifying that a client is allowed to take a certain action. I wanted to create a solution where only registered users can READ data about movies, UPDATE their profile, or DELETE their profile, but any anonymous client can CREATE a new user.

STEP 4 SECURITY MECHANISMS

Authentication & Authorisation

HTTP & JWT: ADVANTAGES

- Basic HTTP authentication doesn't require any extra storage mechanism for the username and password as they're transmitted from the client to the API directly in the HTTP header.
- As it's risky to use HTTP authentication as a general authentication scheme beyond an initial login request, I paired it with a JWT token based authorisation. The initial user authentication will be handled by basic HTTP authentication. As a result the application generates a JWT for the user, allowing subsequent requests to be authenticated and authorised with JWT-based authentication.



STEP 4 SECURITY MECHANISMS

Data Security, Validation, & Ethics

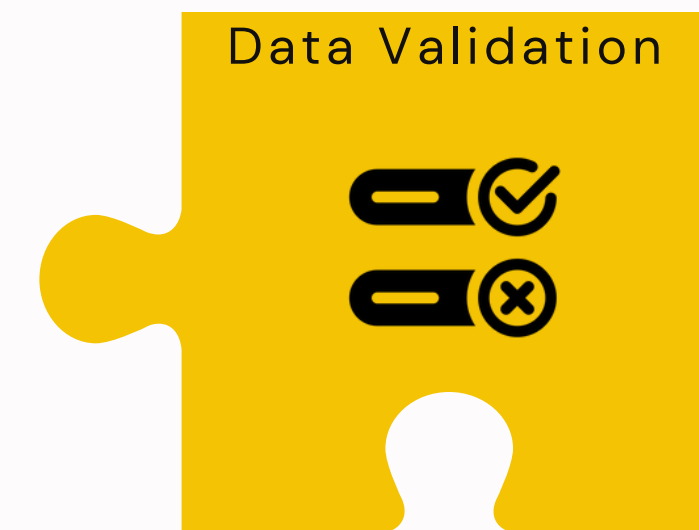
Ensuring data and web security considerations have been incorporated into the product and that I am fulfilling ethical responsibilities as a web developer.



Controls which domains have access to API's server, keeping it protected from malicious entities. The frontend will still be able to make requests to the API even if hosted at different domains.



Used the *bcrypt* module to hash users' passwords and compare hashed passwords every time users log in to ensure a secured login authentication process.

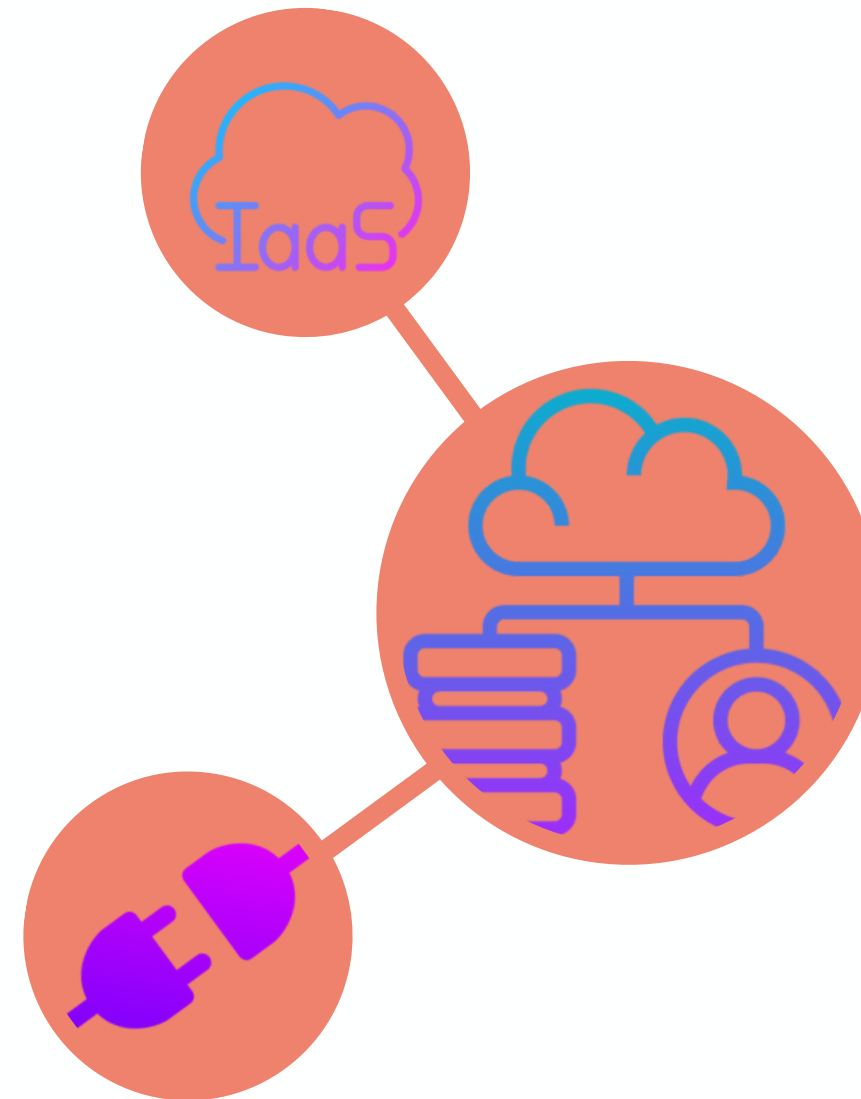


Against frontend attacks, *Express validator* library ensures that user inputs follow the correct format, minimising the risk of those containing malicious scripts. Also prevents bugs, as it ensures only expected types of data are stored in the database.

STEP 5 DEPLOYMENT

I have chosen to host my API on a PaaS provider called Netlify. A PaaS model offers not only servers and storage on an as-needed basis, but also an operating system and tools to help you build and deploy your application.

Once online, I was able to connect my database to my API on Netlify, ensuring that the entire backend of the project was online and ready to be used in tandem with a client app.



HOSTING MODELS

- Shared hosting
- Dedicated hosting
- Infrastructure as a Service (IaaS) aka “cloud infrastructure”
- Platform as a Service (PaaS)

I then imported the JSON files of my local MongoDB database onto a MongoDB Atlas “cluster” in order to host my database online.

RETROSPECTIVE



WHAT WENT WELL?

Building a database was my absolute favourite project of the entire course. I enjoyed the coding part itself, how to organise data as well as learning more about data storage and how to secure a web app. I gained substantial understandings of the different layers that compose the backend, which was still a mystery at this stage and loved the learning journey.



CHALLENGES

It all started pretty bad as I installed the wrong version of Node and found myself stuck for a long time. After seeking for help from more advanced developers on forums I eventually was set on the right path. This prepared me for the inevitable compatibility issues I was going to witness all along future projects and forced me to get out of my shell to ask for some help.



FUTURE FEATURES

Increasing the movies collection, adding a sorting option by country together with text input for users to write reviews and ratings are additional features I'd like to work on. Next I'd like to build an API with my favourite recipes and be able to search for them based on ingredients.



FINAL THOUGHTS

I have come a long way to complete my first backend project and am thrilled about all the new knowledge I created in the process. This project got me thinking a lot about data security and would like to further investigate on that topic. Also, how much more complicated is it to build a streaming platform? Will be looking into that too in my free time.

CREDITS

Library icons, Fast icons, Magnifying glass, Password icons,
Cloud service & Challenge icons created by Eucalyp
Acceptance criterion icons created by Danteee82
Computer icons created by Freepik
Api icons created by Grand Iconic
Free icons created by JunGSa
laas icons created by bsd

<https://www.flaticon.com>