



CineDEX

A FULL STACK CASE STUDY

Jamie Tracy



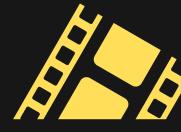
CineDEX

Overview

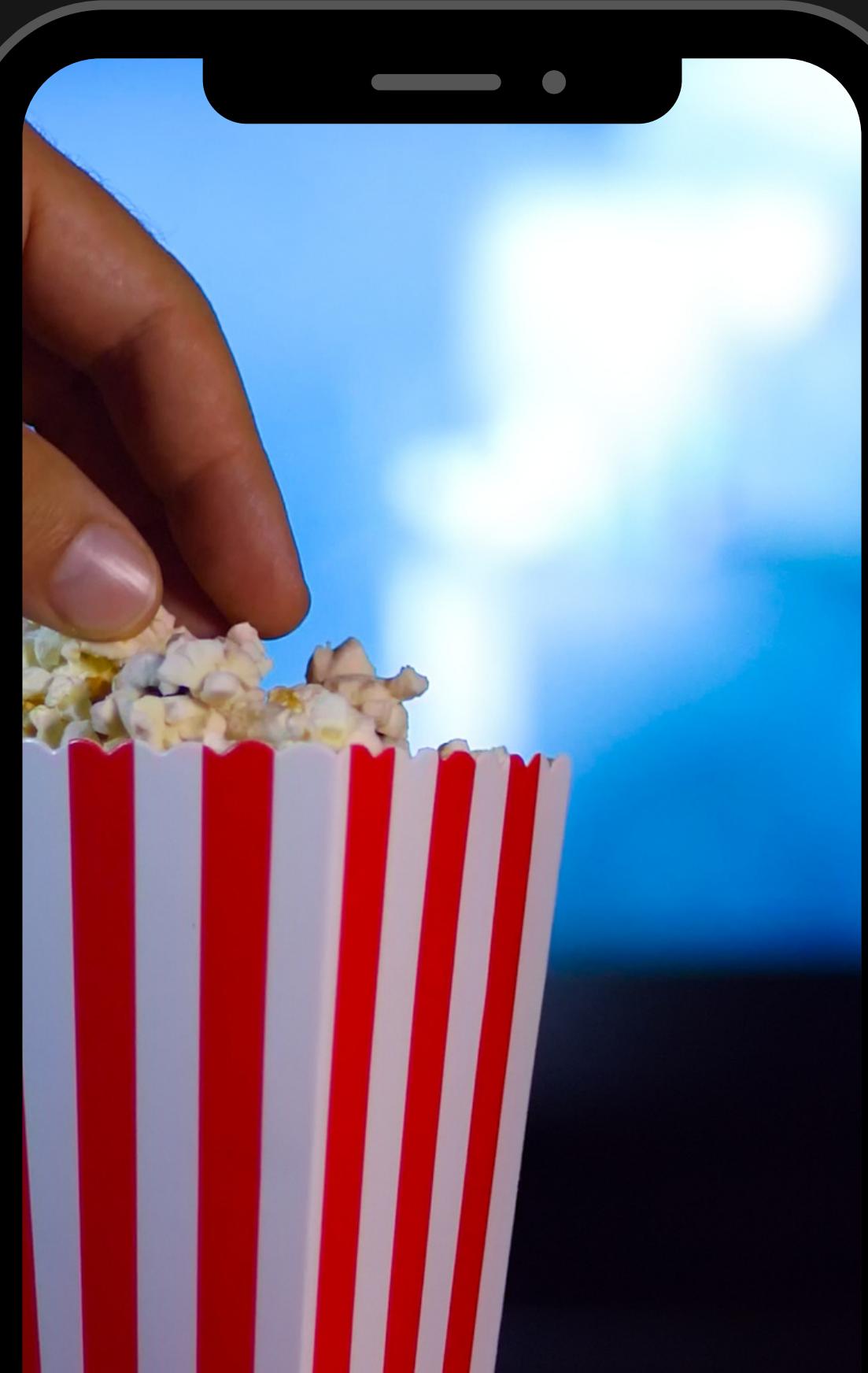
CineDEX is a single-paged responsive web application developed using the MERN stack and uses routing and multiple interface views to provide a number of features to film lovers. Users can register, login, update their profiles, search for movies by title, genre, or director and even create a favorite movie list where they can add or remove favorites. CineDEX also provides users with information about an assortment of movies, directors, and genres.

[Check it out](#)





CineDEX



Objective



This project was part of the full-stack immersion course from the web development program at CareerFoundry to demonstrate mastery of full-stack JavaScript development. The goal was to create a complete web app (server-side and client-side) to showcase in my portfolio.



Details

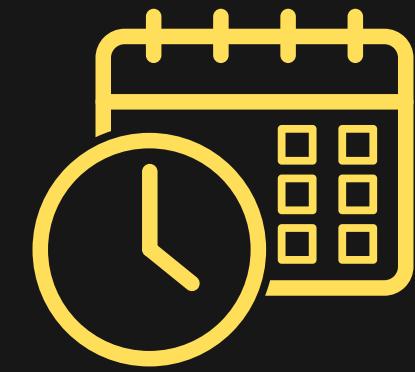


Roles

Role: Lead Developer

Tutor: Ebere Iweala

Mentor: Mahesh Rodrigo



Duration

Both server-side and client-side took me approximately 7-8 weeks to complete from start to finish, but client-side took significantly longer than server-side.



Methodologies

- MERN stack
- Postman
- Heroku
- React Bootstrap
- Redux



Server-Side

A screenshot of the Postman application interface. The top navigation bar shows 'Movie REST API Requests' and 'Allow new users to register'. The main area displays a POST request to 'https://cinedex.herokuapp.com/users'. The 'Body' tab shows a JSON payload:

```
1 "Username": "Pickles",
2 "Password": "$2$10g97TA7Ty6dF6SCNk2fmeJ.Fz/yTGVAtOnCjyBuoIRtz09sydgtI",
3 "Email": "imabigdilly@yahoo.com",
4 "Birthday": "2004-04-22T00:00:00Z",
5 "FavoriteMovies": [],
6 "_id": "5d65bd2857e83569ebd705",
7 "__v": 0
```

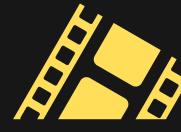
The response status is '201 Created' with a time of '517 ms'. The 'Pretty' tab shows the same JSON data.



A screenshot of the Postman application interface. The top navigation bar shows 'Movie REST API Requests' and 'Return list of all movies'. The main area displays a GET request to 'http://localhost:8080/movies'. The 'Body' tab shows the response:

```
1 [
2   {
3     "Genre": "Drama",
4     "Name": "Drama",
5     "Description": "The drama genre features stories with high stakes and many conflicts. They're plot-driven and demand that every char",
6     "Director": {
7       "Name": "Michel Gondry",
8       "Bio": "Michel Gondry is a French filmmaker noted for his inventive visual style and distinctive manipulation of mise en scène.",
9       "Birthyear": 1963
10    },
11    "Id": "646cc777f4f99837cd904",
12    "Title": "Eternal Sunshine of the Soggy Mind"
13  }
]
```

I created a REST API to interact with a non-relational database of movies I made using MongoDB. The API is accessed through HTTP methods, where requests can be sent to set endpoints to retrieve specific data from the database. I also implemented basic HTTP and JWT token-based authentication as well as password hashing and data validation. I was able to test the functionality of these endpoints using Postman during development. Last, I used MongoDB Atlas to host my database and Heroku to deploy my app.

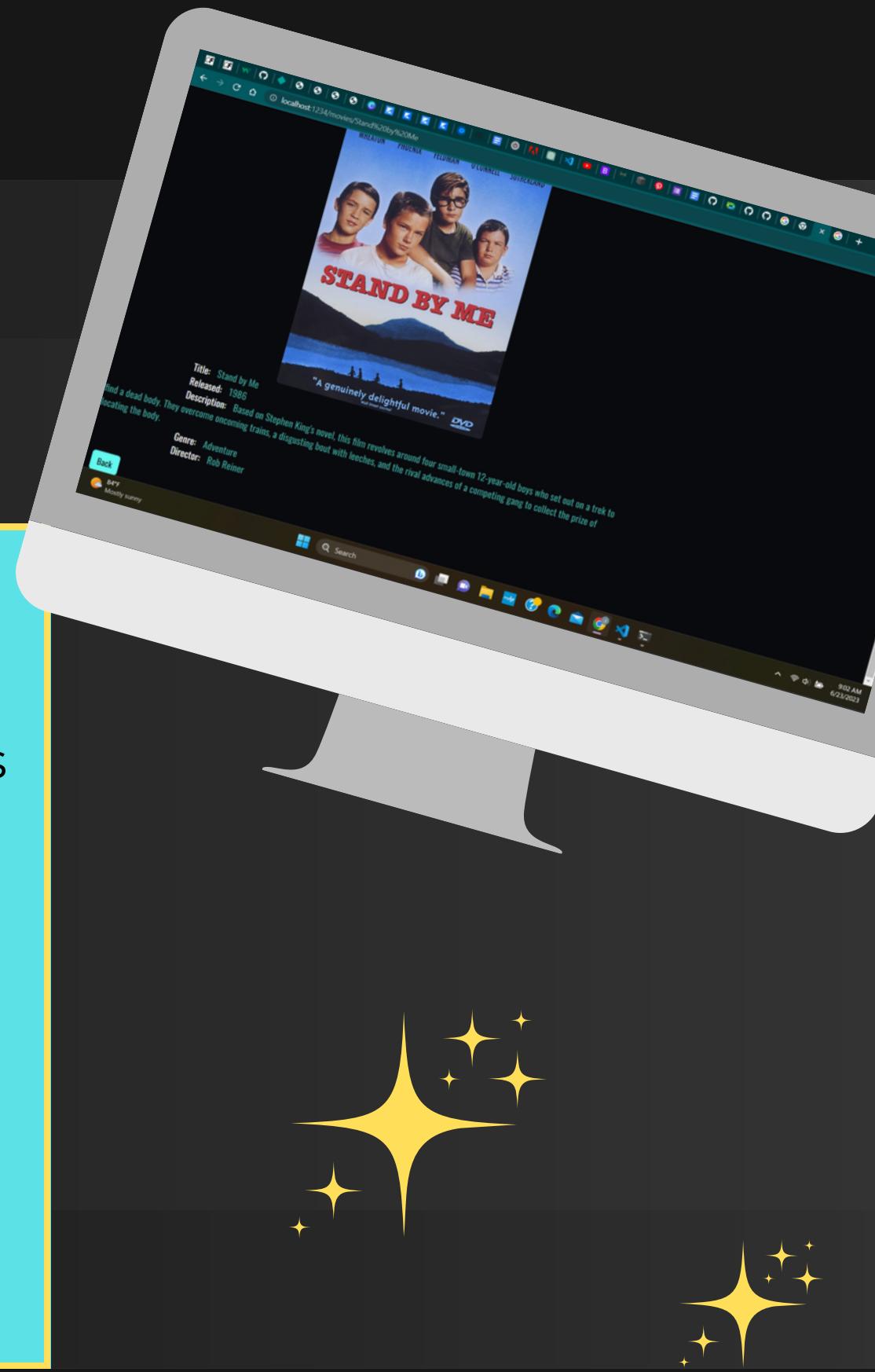
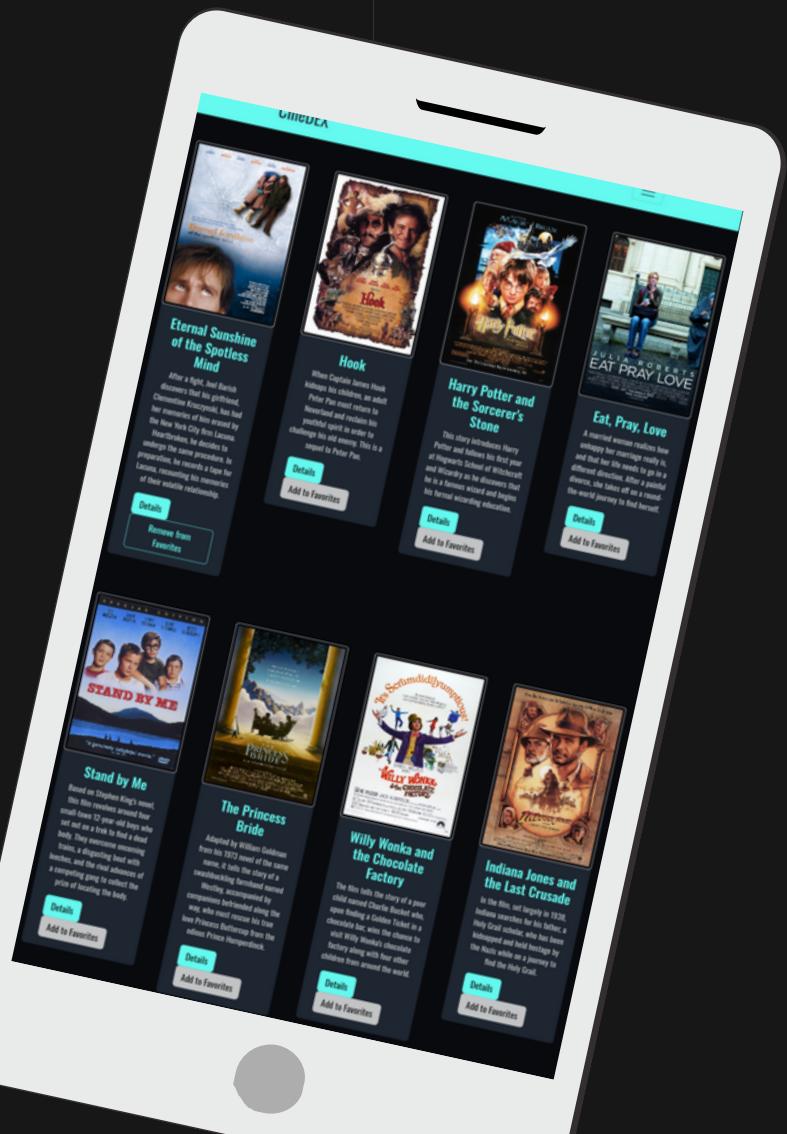


CineDEX

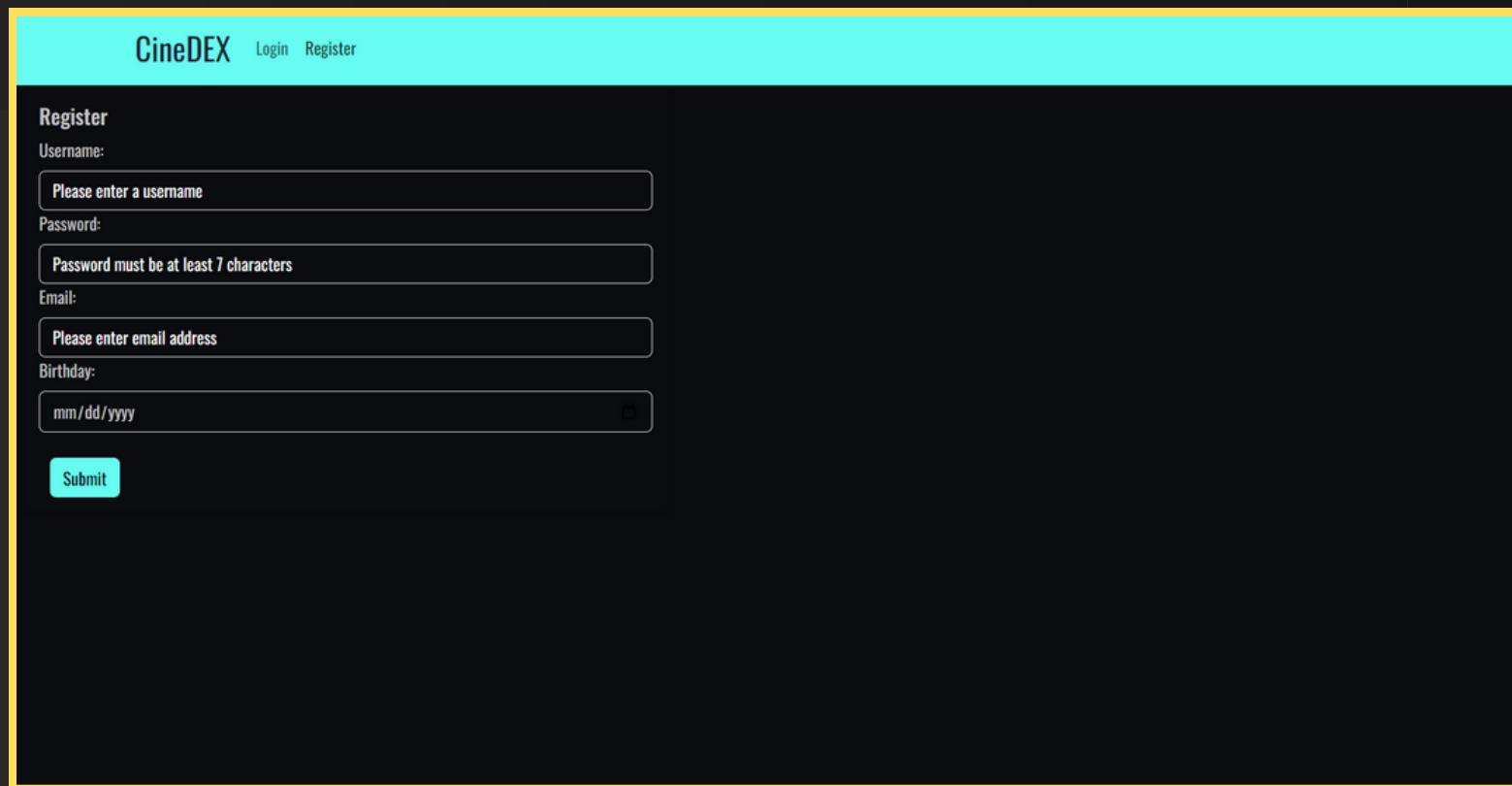
Client-Side

After ensuring my server-side code was fully functional, I needed to create the interface to be used by users to make requests and get responses from the server-side. Using React and Redux I created components for each of the interface views of the application.

After developing the logic to support the requests from users and handle data through the REST API endpoints, I used React Bootstrap to create aesthetically appealing and consistent styling.

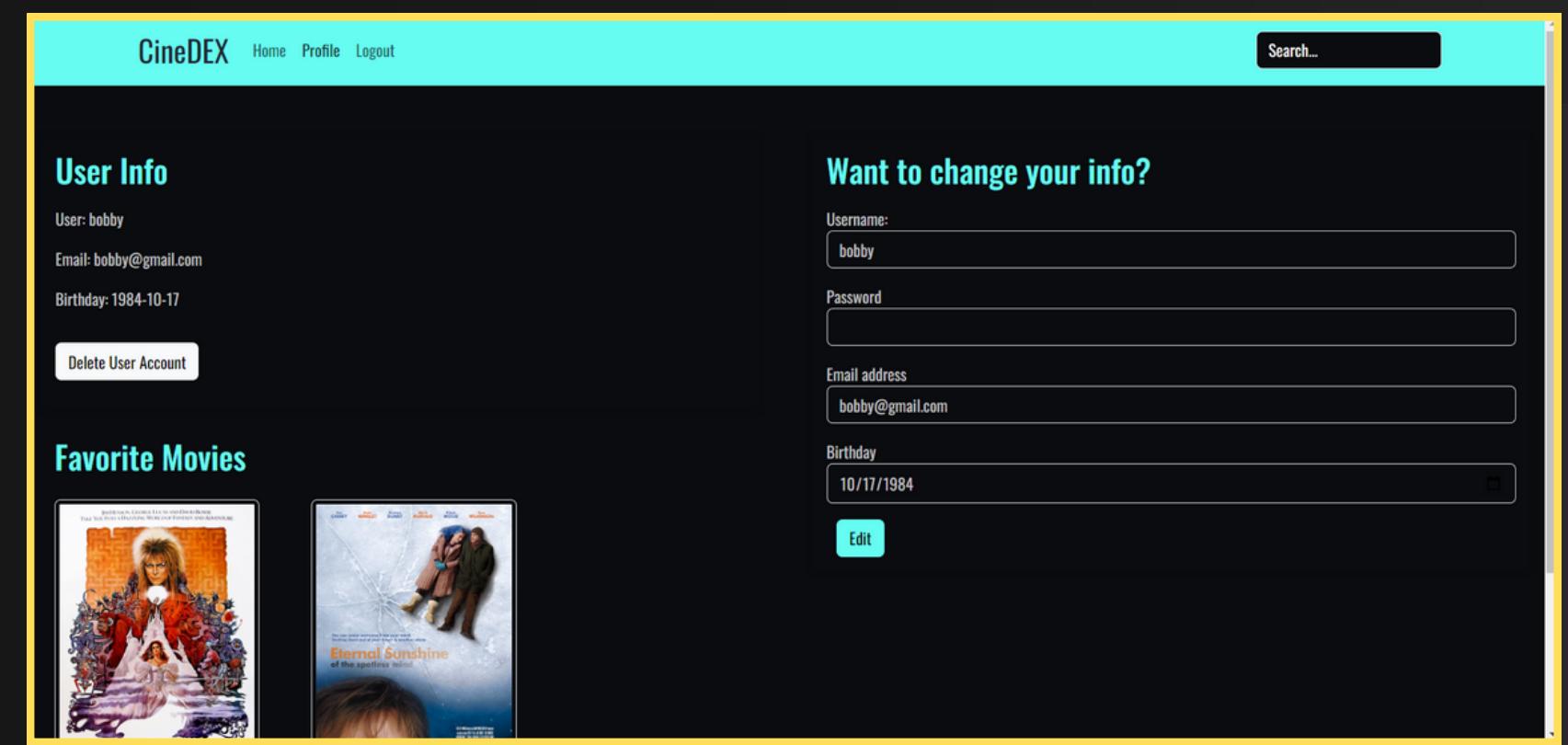


Interface Views



The registration view shows a form titled "Register". It includes fields for "Username" (with placeholder "Please enter a username"), "Password" (with placeholder "Password must be at least 7 characters"), "Email" (with placeholder "Please enter email address"), and "Birthday" (with placeholder "mm/dd/yyyy"). A "Submit" button is located at the bottom left.

Registration View

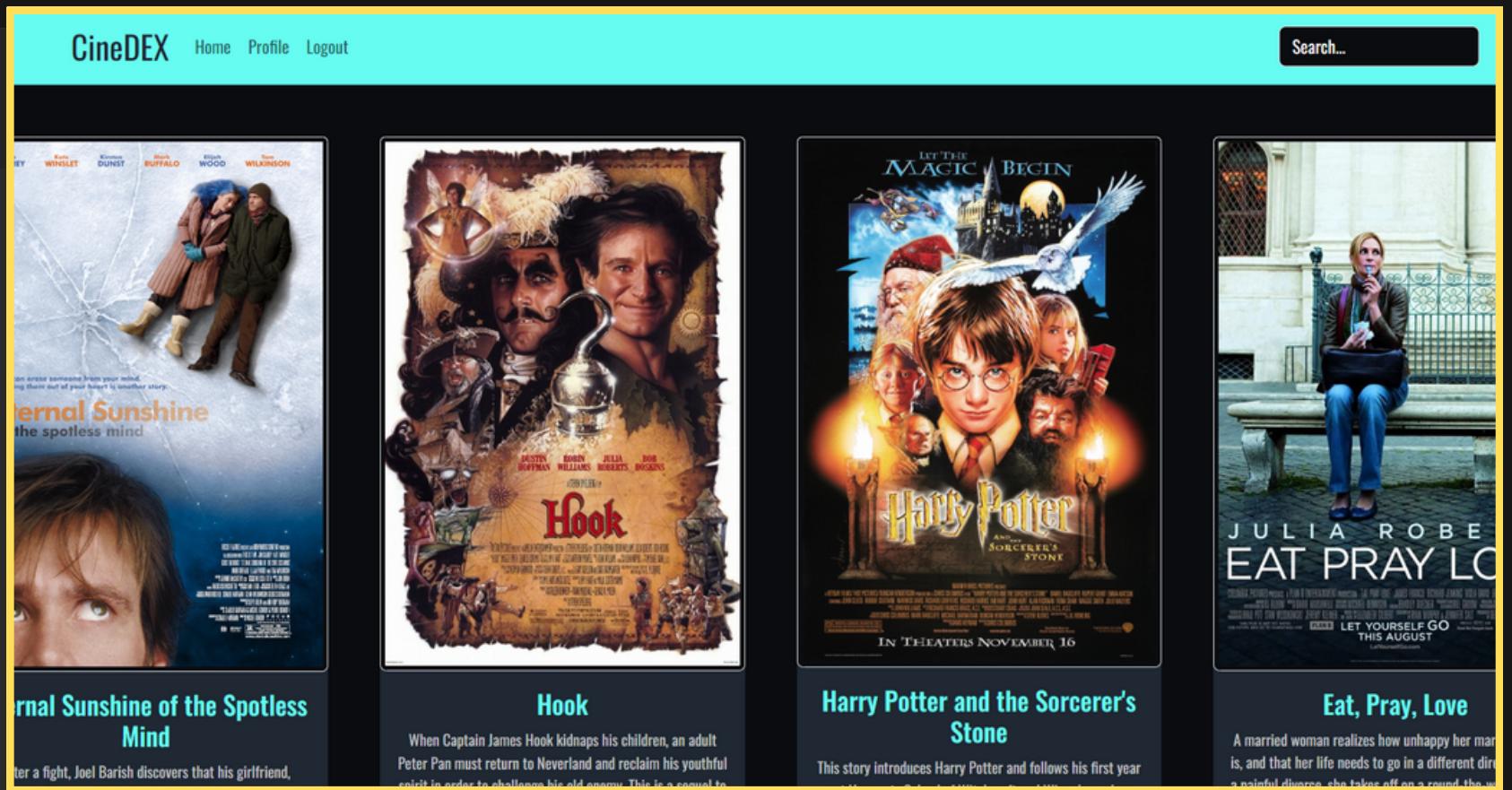


The profile view shows a "User Info" section with details: User: bobby, Email: bobby@gmail.com, Birthday: 1984-10-17, and a "Delete User Account" button. Below this is a "Favorite Movies" section featuring two movie posters: "Labyrinth" and "Eternal Sunshine of the Spotless Mind". To the right, there's a "Want to change your info?" section with fields for "Username" (bobby), "Password", "Email address" (bobby@gmail.com), and "Birthday" (10/17/1984). An "Edit" button is located at the bottom right of this section.

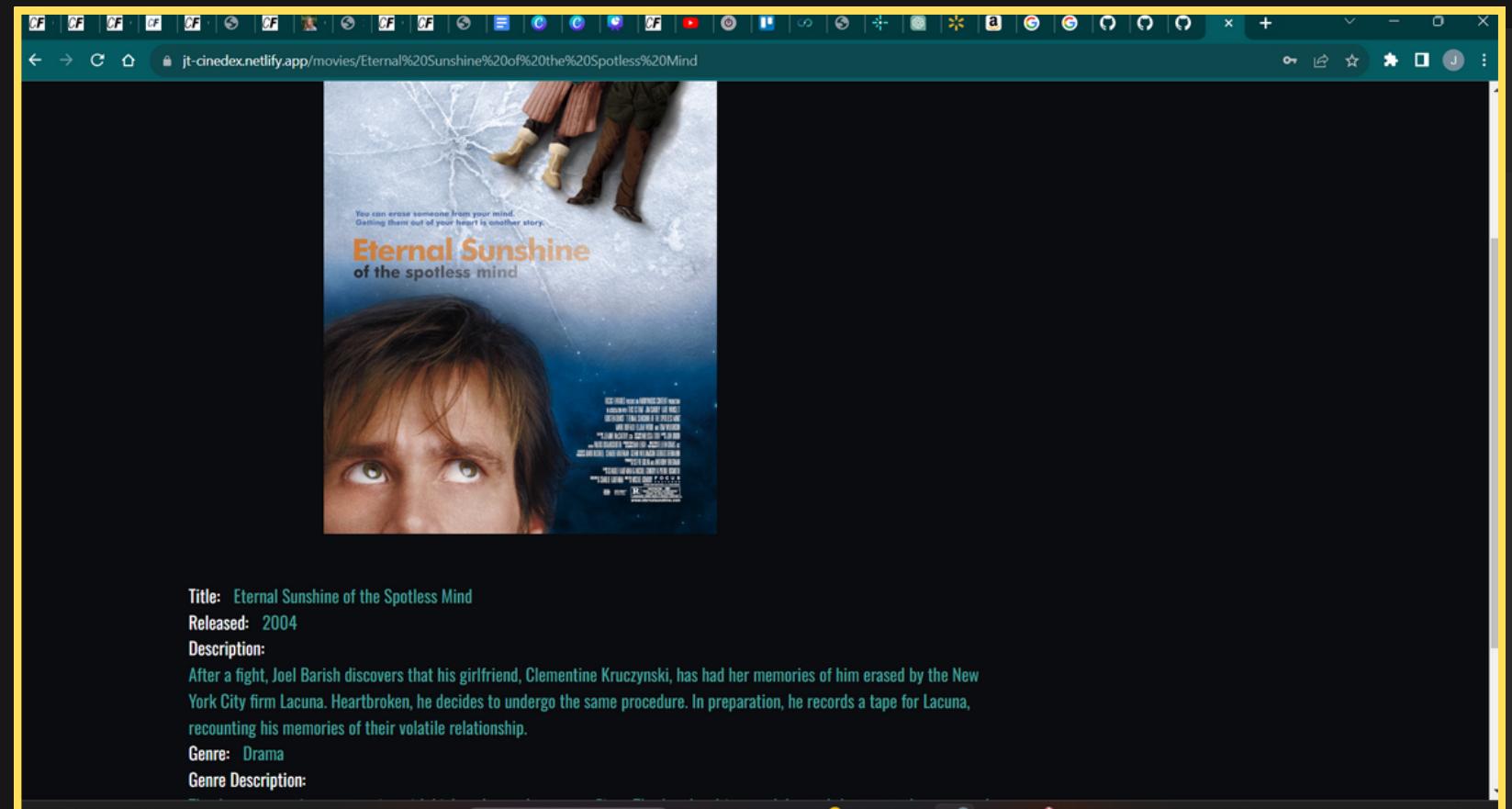
Profile View

Users can register, log in, create a profile and browse movies.

Interface Views



Main View



Movie View

Upon clicking the details button on a particular movie, the user can access release date, a movie synopsis, and genre and director information.

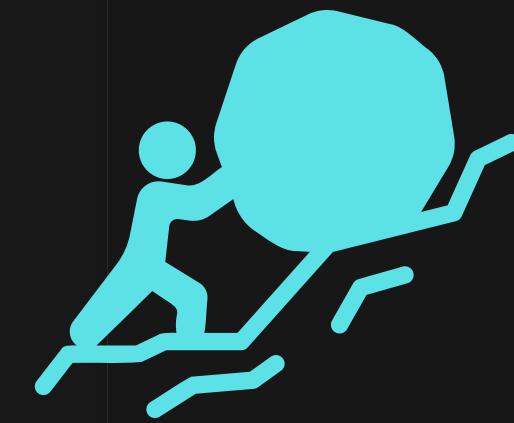


Retrospective



What went well

As a very organized person and analytical thinker, I really enjoyed working on the server-side portion and database and felt I was able to complete it quickly and efficiently.



Challenges

While I struggled a bit at first with the logic behind the interaction between components, I tried to take more time to grasp the new concepts I learned. Finding the time to do this was also a challenge.



Final thoughts

I believe I would have benefitted from creating wireframes of each view so I could get a better feel for exactly what logic was required for each component and how it would interact with the other components as well as with styling.