



CineDEX

A FULL STACK CASE STUDY

Jamie Tracy



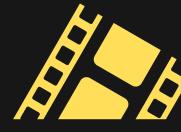
CineDEX

Overview

CineDEX is a single-paged responsive web application was 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

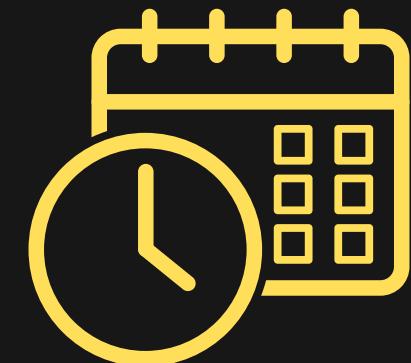


Roles

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 the current request is 'POST Allow new users to register'. The URL is <https://cinedex.herokuapp.com/users>. The 'Body' tab is selected, showing a JSON payload:

```
1
2   "Username": "Pickles",
3   "Password": "$2b$09$7fA7Ty6dF6SCNk2fmeJ.Fz/yTGVAtOnCjyBuoIRtZo9sydQti",
4   "Email": "imabigdilly@yahoo.com",
5   "Birthday": "2004-04-22T00:00:000Z",
6   "FavoriteMovies": [],
7   "_id": "6465bd2857e83569ebd705",
8   "__v": 0
```

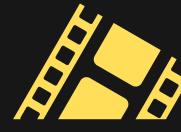
The response status is 201 Created, and the response body is a JSON object containing user information.



A screenshot of the Postman application interface. The top navigation bar shows 'Movie REST API Requests' and the current request is 'GET Return list of all movies'. The URL is <http://localhost:8080/movies>. The 'Body' tab is selected, showing a JSON response:

```
1
2   [
3     {
4       "Genre": [
5         "Name": "Drama",
6         "Description": "The drama genre features stories with high stakes and many conflicts. They're plot-driven and demand that every character has a clear motivation and goal."
7       ],
8       "Director": [
9         "Name": "Michel Gondry",
10        "Bio": "Michel Gondry is a French filmmaker noted for his inventive visual style and distinctive manipulation of mise en scène.",
11        "BirthYear": 1963
12      ],
13      "Id": "646cc77f746992877cd904",
14      "Title": "Eternal Sunshine of the Soggy Mind"
15    }
16  ]
```

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.

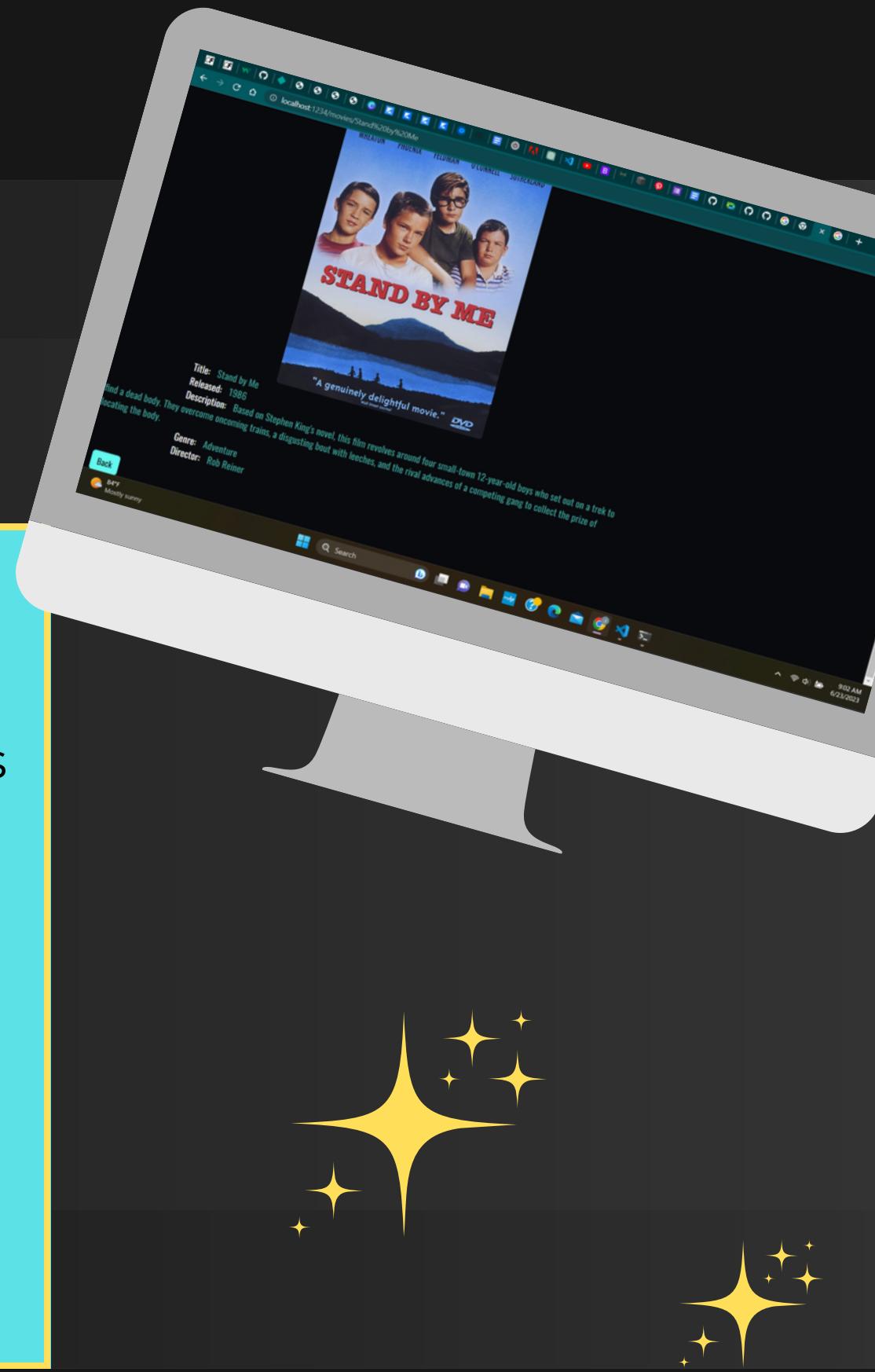
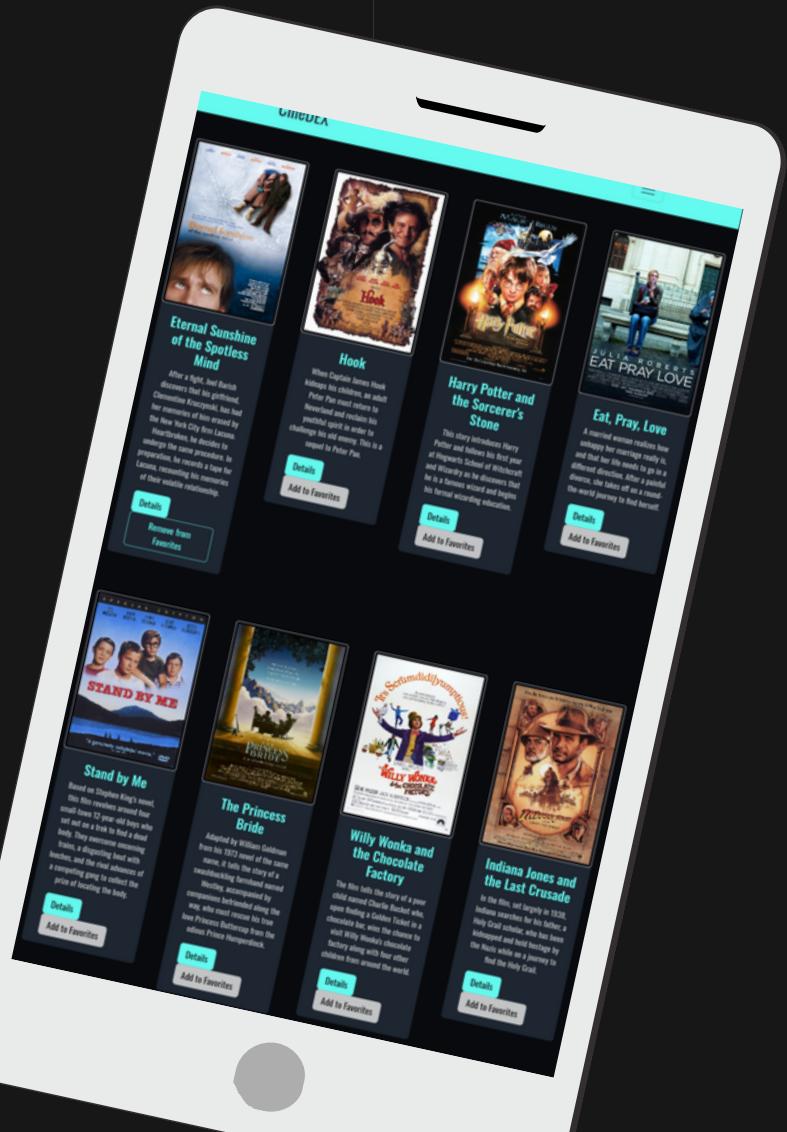


CineDEX

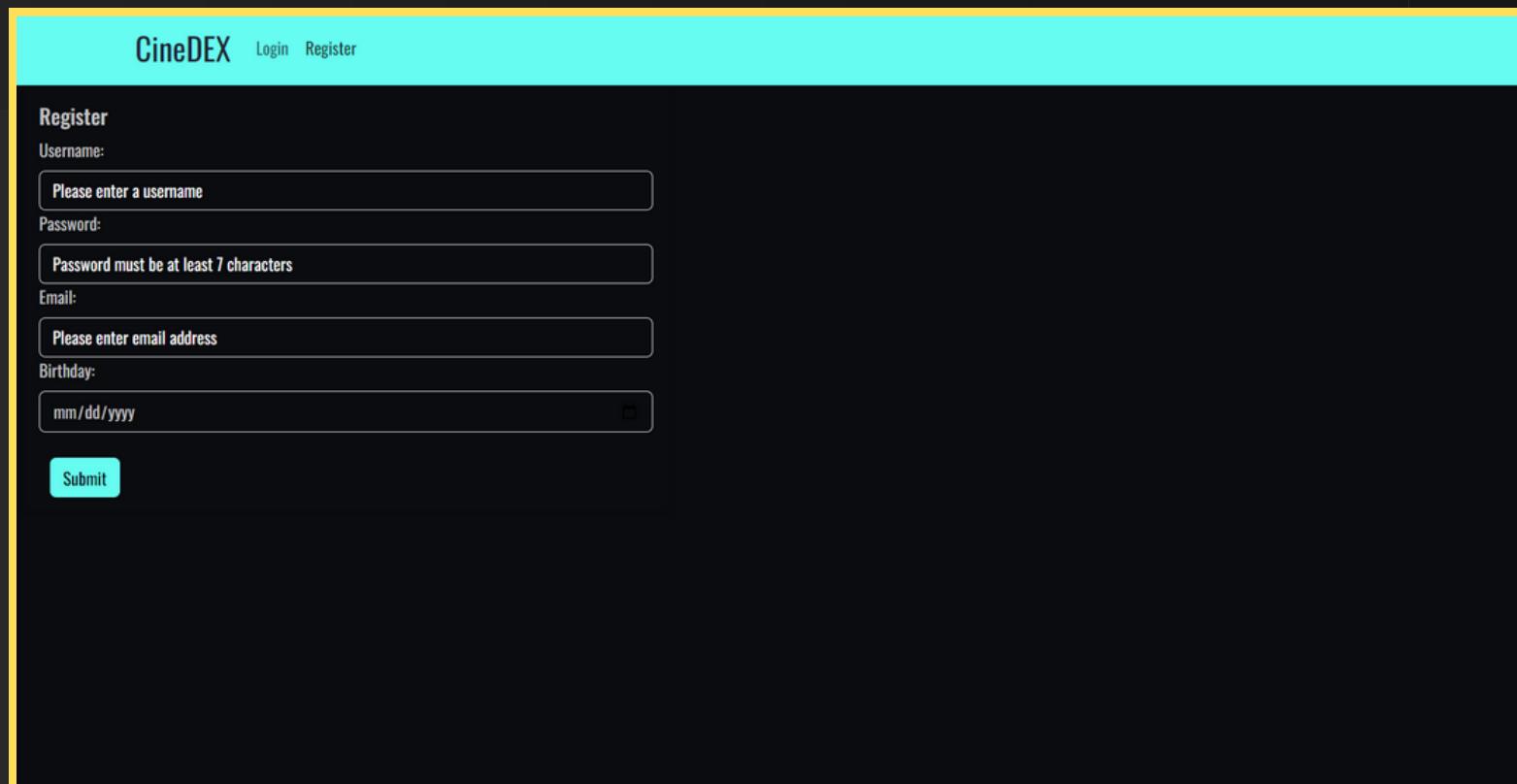
Client-Side

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.

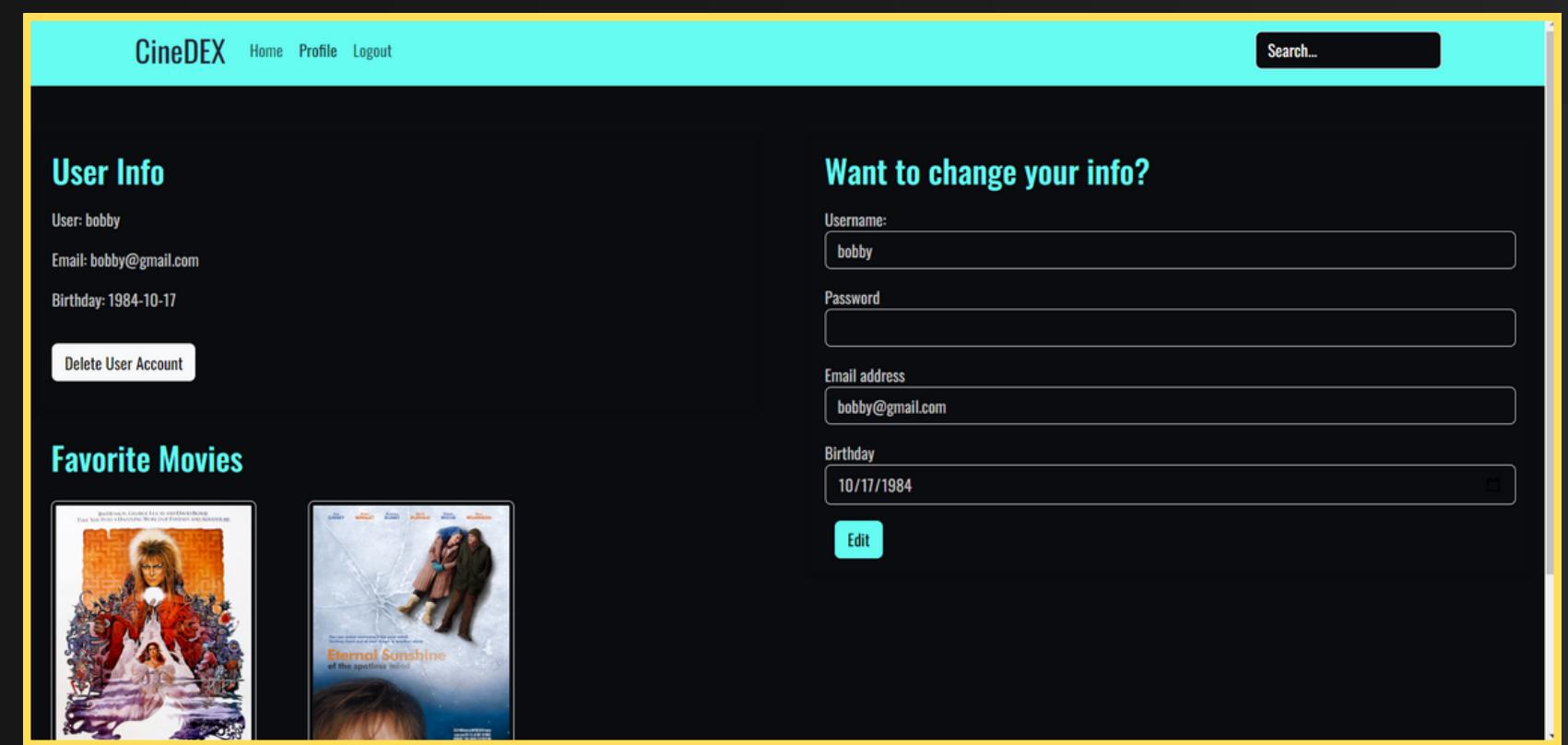


Interface Views



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

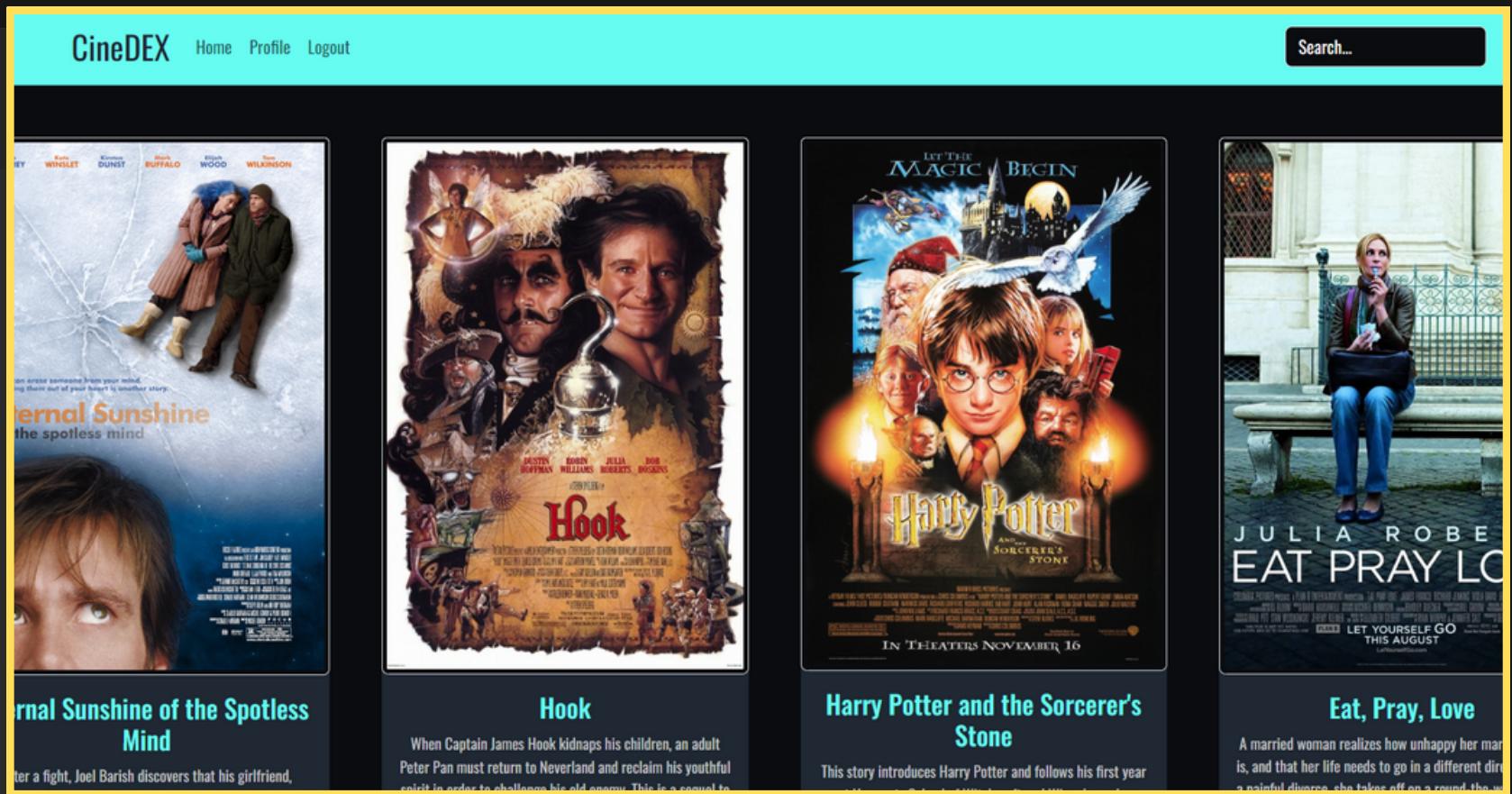


The profile view shows a "User Info" section with details: User: bobby, Email: bobby@gmail.com, Birthday: 1984-10-17, and a "Delete User Account" button. Below this is a "Want to change your info?" section with input fields for "Username" (bobby), "Password", "Email address" (bobby@gmail.com), and "Birthday"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

Profile View

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

Interface Views



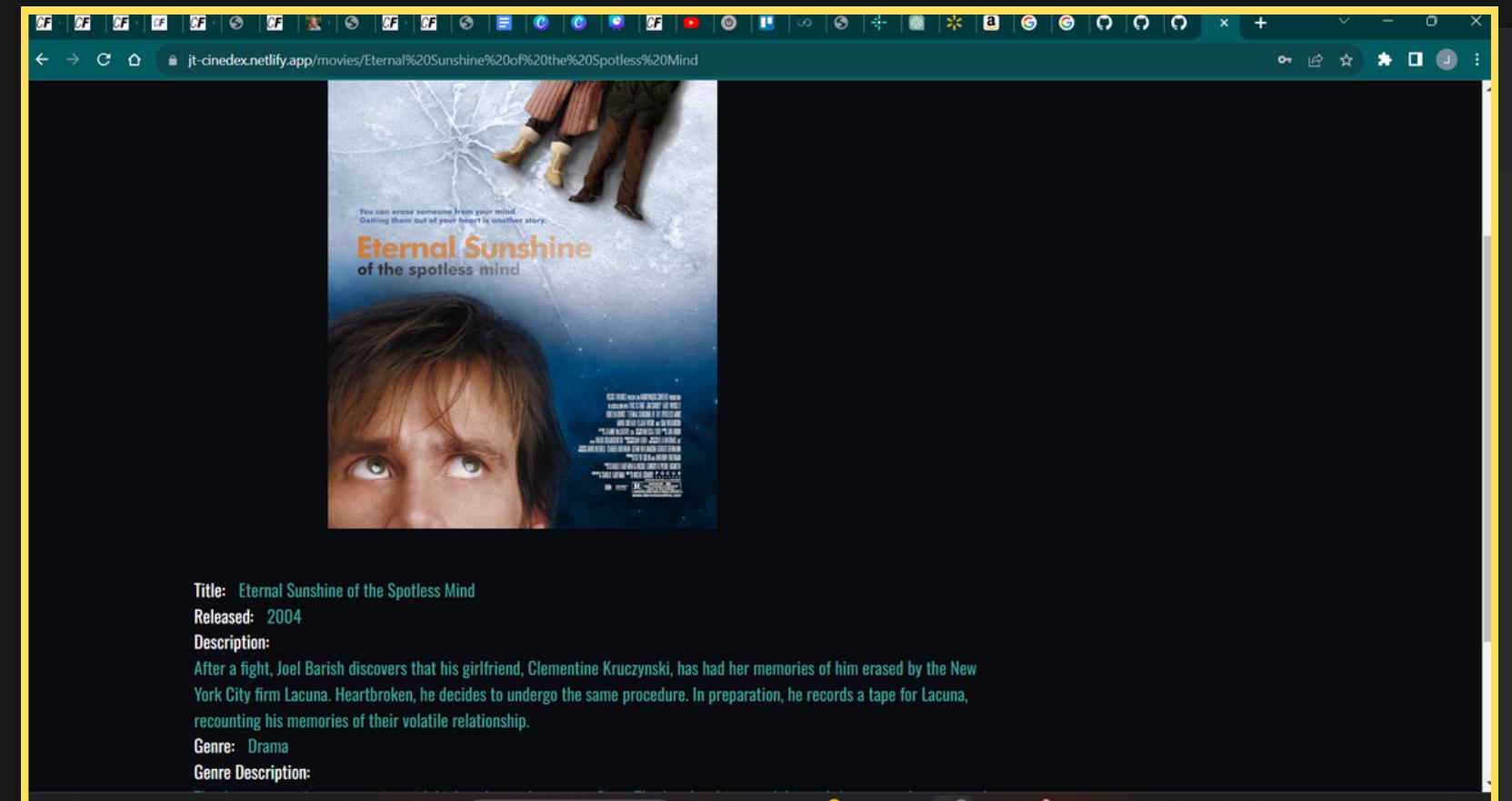
Eternal Sunshine of the Spotless Mind
After a fight, Joel Barish discovers that his girlfriend, Clementine Kruczynski, has had her memories of him erased by the New York City firm Lacuna.

Hook
When Captain James Hook kidnaps his children, an adult Peter Pan must return to Neverland and reclaim his youthful spirit in order to challenge his old enemy. This is a sequel to

Harry Potter and the Sorcerer's Stone
This story introduces Harry Potter and follows his first year

Eat, Pray, Love
A married woman realizes how unhappy her marriage is, and that her life needs to go in a different direction. After a painful divorce, she takes off on a round-the-world trip.

Main View



Eternal Sunshine of the Spotless Mind
Title: Eternal Sunshine of the Spotless Mind
Released: 2004
Description:
After a fight, Joel Barish discovers that his girlfriend, Clementine Kruczynski, has had her memories of him erased by the New York City firm Lacuna. Heartbroken, he decides to undergo the same procedure. In preparation, he records a tape for Lacuna, recounting his memories of their volatile relationship.
Genre: Drama
Genre Description:

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.