

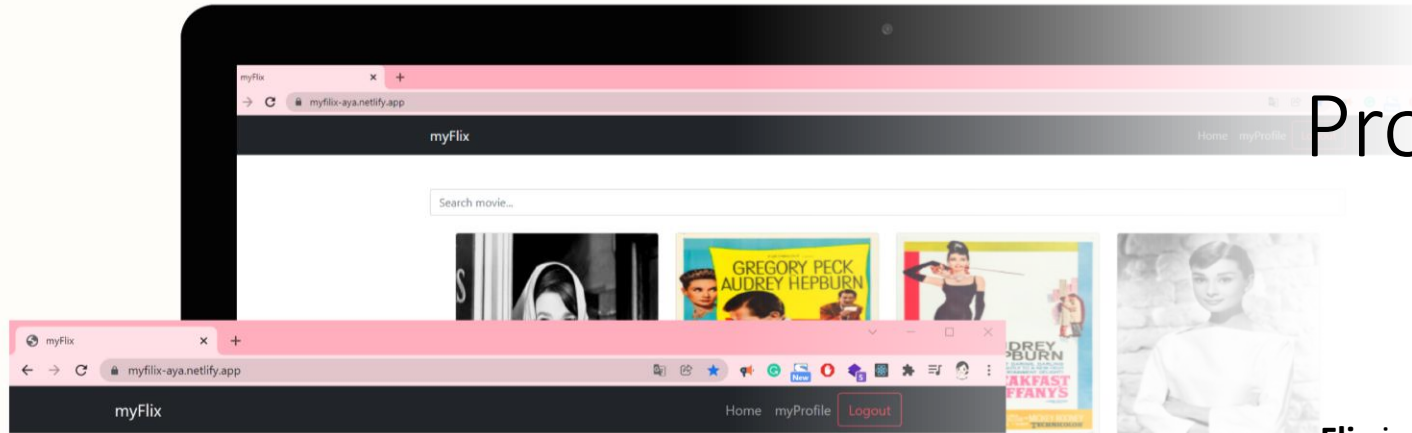
myFlix full-stack project

Case Study

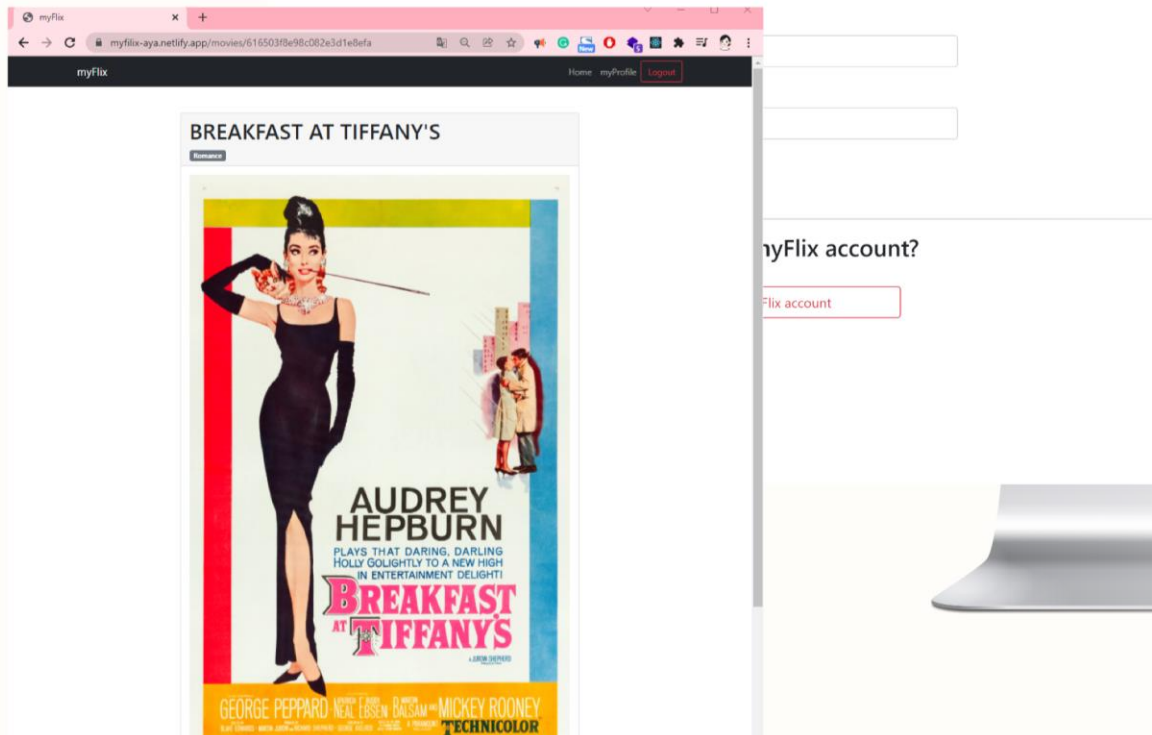
Project Overview and Objective

myFlix is a full-stack web application project adapting MERN (MongoDB, Express, React, and Node.js) stack. The application provides movies information (title, genre, director, description, etc....) and allows users to register, update their profile and add/delete their favourite movie lists.

My objective for the project is to build the complete both server-side and client-side for the application from scratch. The project will demonstrate my mastery of full-stack JavaScript development, including APIs, databases, business logic, authentication, and more.



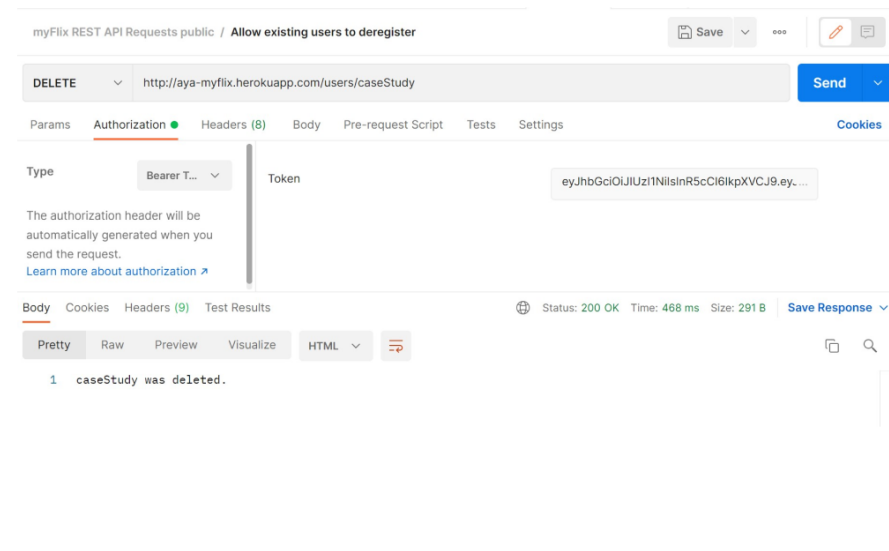
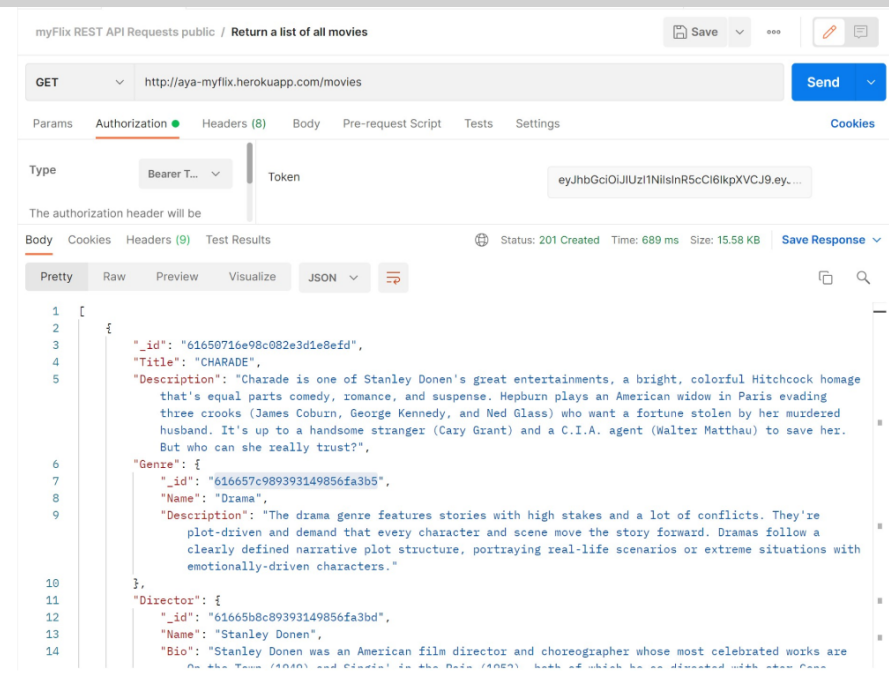
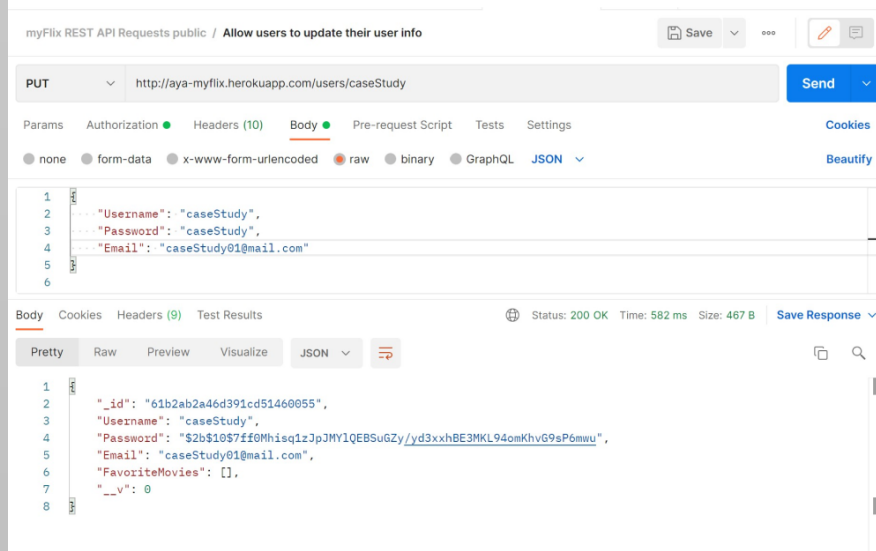
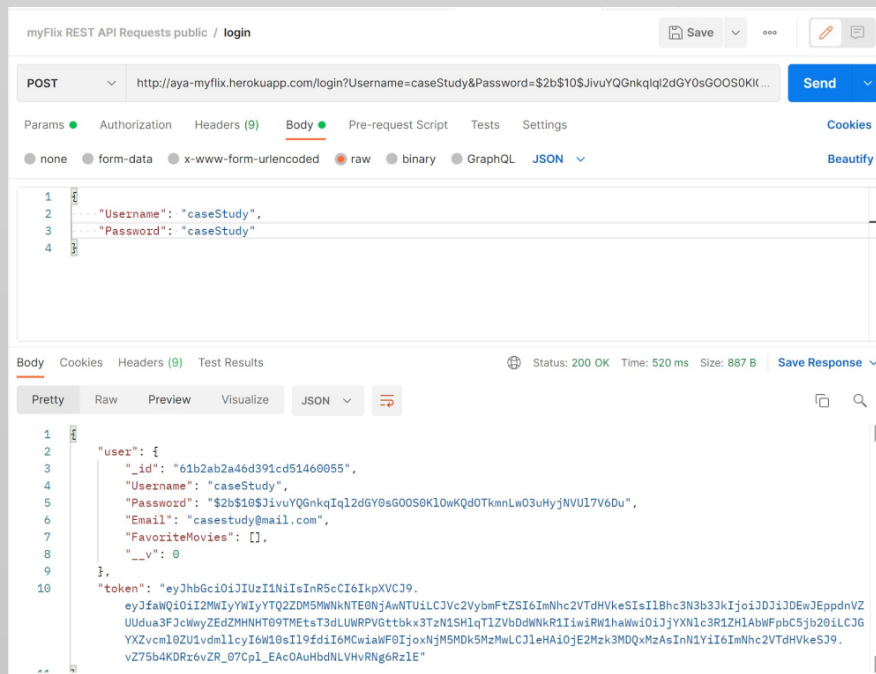
myFlix Login Page



Server-Side

To complete the server-side of my web application, I used Node.js and Express to create RESTful API which is accessed via commonly used HTTP methods like GET and POST. The database was built using MongoDB (non-relational database) and hosted on MongoDB Atlas.

I used Postman to test all API endpoints and for data security reasons, I included user authentication and authorization by using basic HTTP authentication and JWT (token-based) authentication. The completed server-side application was deployed to Heroku.

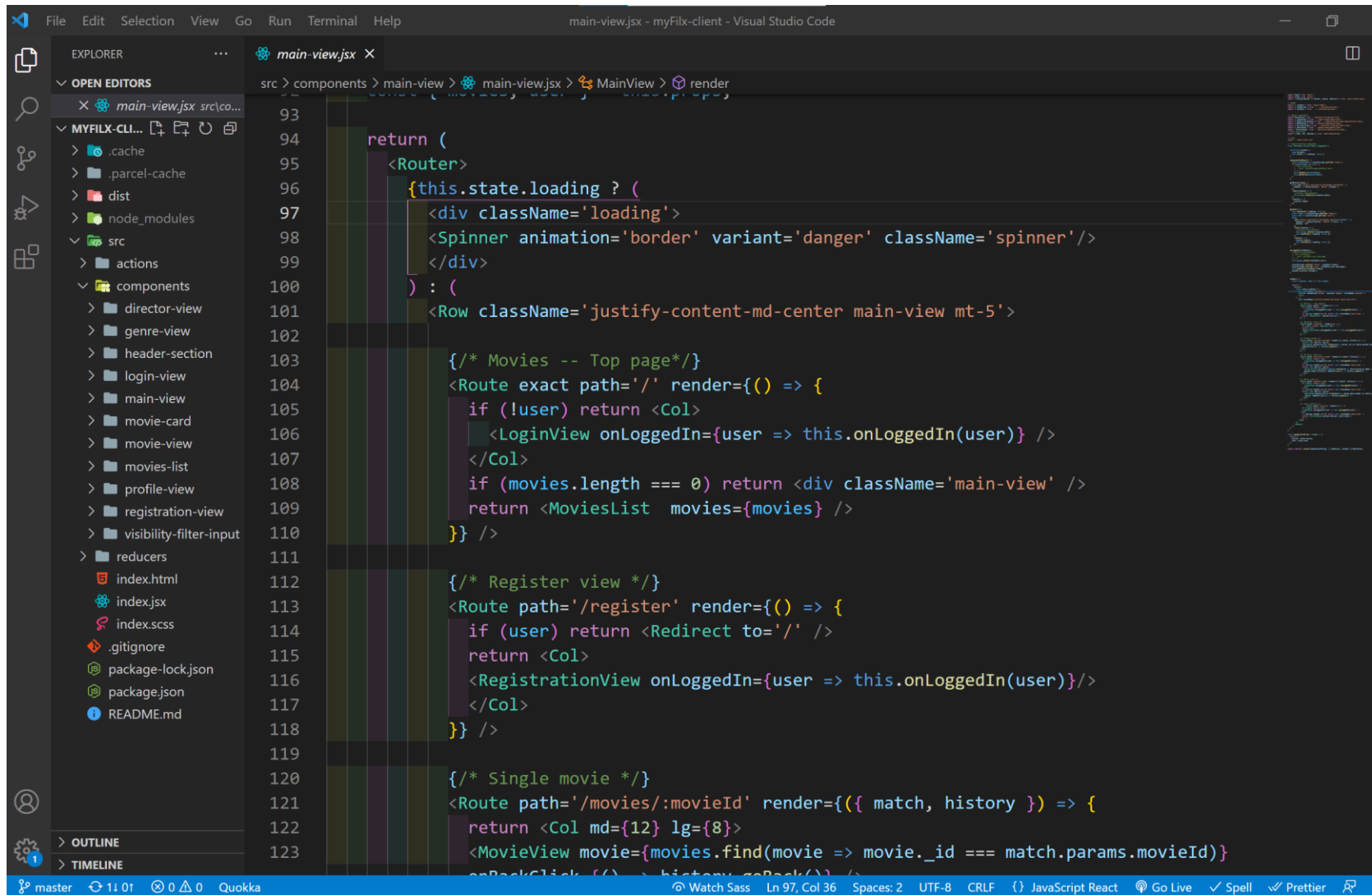


Client-Side

The client-side application is a single-page application (SPA) using React and React-Redux.

After the API and the database creation are completed, I start creating the interface which users would use when making requests and receiving responses from the server-side. I used React Bootstrap, the one of most popular UI libraries, to achieve a modern responsive appearance.

The application is hosted on Netlify (<https://myfilix-aya.netlify.app/>)



```
93
94   return (
95     <Router>
96       {this.state.loading ? (
97         <div className='loading'>
98           <Spinner animation='border' variant='danger' className='spinner' />
99         </div>
100       ) : (
101         <Row className='justify-content-md-center main-view mt-5'>
102
103           { /* Movies -- Top page */ }
104           <Route exact path='/' render={() => {
105             if (!user) return <Col>
106               <LoginView onLoggedIn={user => this.onLoggedIn(user)} />
107             </Col>
108             if (movies.length === 0) return <div className='main-view' />
109             return <MoviesList movies={movies} />
110           }} />
111
112           { /* Register view */ }
113           <Route path='/register' render={() => {
114             if (user) return <Redirect to='/' />
115             return <Col>
116               <RegistrationView onLoggedIn={user => this.onLoggedIn(user)} />
117             </Col>
118           }} />
119
120           { /* Single movie */ }
121           <Route path='/movies/:movieId' render={({ match, history }) => {
122             return <Col md={12} lg={8}>
123               <MovieView movie={movies.find(movie => movie._id === match.params.movieId)}
124                 onClick={() => history.goBack()} />
125             </Col>
126           }} />
127         </Row>
128       )
129     }
130   )
131 }
```

Challenges

The challenging part for the server-side is to become comfortable with using the CLI (command-line interface) but the database creation phase helped me to get used to using it.

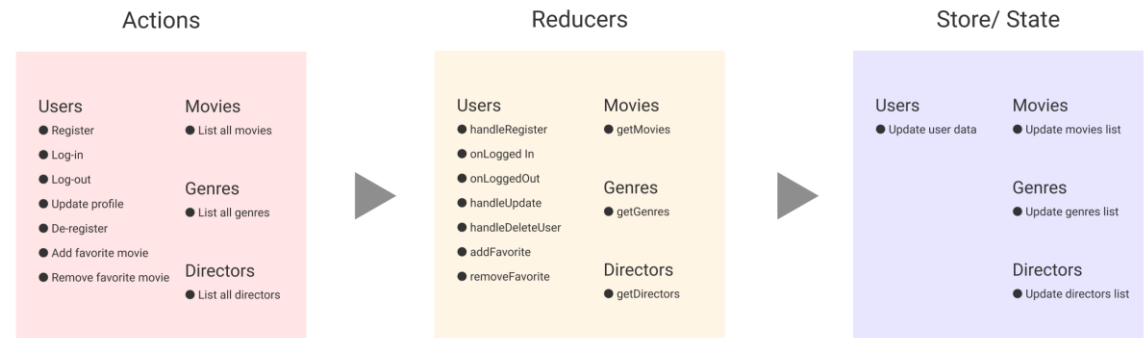
For the client-side, it took me time to understand the Redux, and drawing an architectural diagram for the application helped me to see the Redux fundamentals.

Duration

The project period was from 4th October to 5th November 2021.

- Server-side: 4 October to 17 October
- Client-side: 18 October to 5 November

It felt like a long time to complete the project, but I managed to finish it in about a month.



The screenshot shows the Visual Studio Code editor with two files open: `actions.js` and `reducers.js`.

actions.js:

```
1 export const SET_MOVIES = 'SET_MOVIES';
2 export const SET_FILTER = 'SET_FILTER';
3 export const SET_USER = 'SET_USER';
4
5 export function setMovies(value) {
6   //console.log('SET_MOVIES action triggered');
7   return {
8     type: SET_MOVIES,
9     value
10  };
11 }
12
13 export function setFilter(value) {
14   //console.log('SET_FILTER action triggered');
15   return {
16     type: SET_FILTER,
17     value
18  };
19 }
20
21 export function setUser(value) {
22   //console.log('SET_USER action triggered');
23   return {
24     type: SET_USER,
25     value
26  };
27 }
```

reducers.js:

```
1 import { combineReducers } from "redux";
2
3 import { SET_MOVIES, SET_FILTER, SET_USER } from "../actions/actions";
4
5 function visibilityFilter(state = '', action) {
6   switch (action.type) {
7     case SET_FILTER:
8       return action.value;
9     default:
10      return state;
11   }
12 }
13
14 function movies(state = [], action) {
15   switch (action.type) {
16     case SET_MOVIES:
17       //console.log('SET_MOVIES reducer reached');
18       return action.value;
19     default:
20       return state;
21   }
22 }
23
24 function user(state = '', action) {
25   switch (action.type) {
26     case SET_USER:
27       //console.log('SET_USERS reducer reached');
28       return action.value;
```

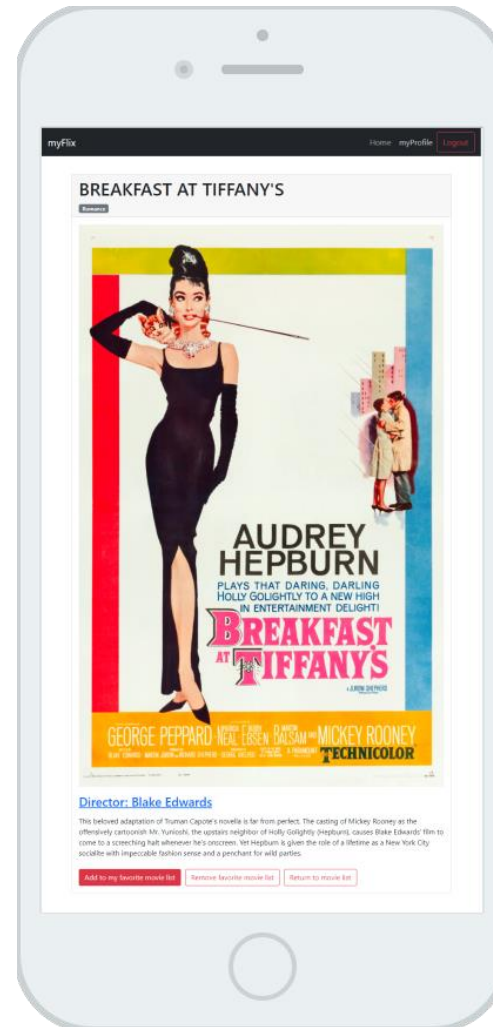
GitHub Repo URLs

myFlix RESTful API

https://github.com/Aya-Ogihara/movie_api

myFlix Client

<https://github.com/Aya-Ogihara/myFlix-client>



Credits

Role: Full stack developer

Tutor: Glen Coffey

Mentor: Theran Brigowatz