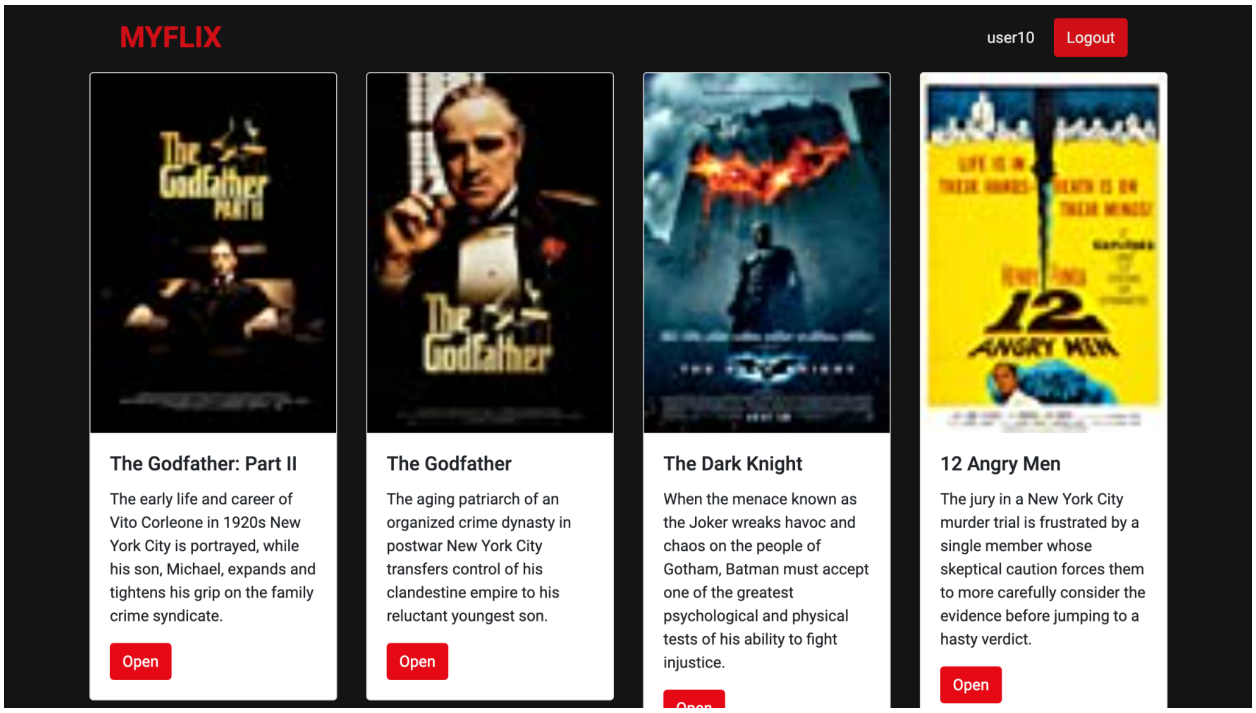
Case Study: MyFlix React Client

The Challenge

The objective of the MyFlix React Client project was to build the client-side for an application called myFlix based on its existing server-side code (REST API and database) using React. Users must be able to access information on movies, directors, and genres, be able to log in and log out, register, access their profile so they can view and edit their information, and add and remove movies from a list of favorites.



The Action

I started this project by first installing React, Axios, React Bootstrap, React Router, and Parcel.

```
{
  "name": "myflix-client",
  "version": "1.0.0",
  "description": "",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "parcel src/index"
```

```
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/jwho85/myFlix-client.git"
  },
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/jwho85/myFlix-client/issues"
  },
  "homepage": "https://github.com/jwho85/myFlix-client#readme",
  "dependencies": {
    "axios": "^0.27.2",
    "parcel": "^2.0.0-rc.0",
    "prop-types": "^15.8.1",
    "react": "^18.0.0",
    "react-bootstrap": "^2.3.1",
    "react-dom": "^18.0.0",
    "react-router-dom": "^5.2.1"
  },
  "devDependencies": {
    "@parcel/transformer-sass": "^2.0.0-rc.0"
  }
}
```

I then created the index.html page which connects to Bootstrap and Google Fonts. The <body> section also contains the app-container and a link to index.jsx.

```
<!DOCTYPE html>
<html>

<head>
    <title>myFlix</title>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css
" />
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link

href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300
;0,400;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&display=swap"
```

```
        rel="stylesheet">
</head>

<body>
    <div class="app-container"></div>
    <script type="module" src="index.jsx"></script>
</body>

</html>
```

The index.jsx file contains the React Bootstrap <Container> and the <MainView /> component.

```
import React from 'react';
import ReactDOM from 'react-dom';
import MainView from './components/main-view/main-view';
import Container from 'react-bootstrap/Container';

// Import statement to indicate that you need to bundle `./index.scss`
import './index.css';

// Main component (will eventually use all the others)
class MyFlixApplication extends React.Component {
    render() {
        return (
            <Container>
                <MainView />
            </Container>
        );
    }
}

// Finds the root of your app
const container = document.getElementsByClassName('app-container')[0];

// Tells React to render your app in the root DOM element
ReactDOM.render(React.createElement(MyFlixApplication), container);
```

I then created a components folder with the following files:

- director-view
- footer
- genre-view

- login-view
- main-view
- movie-card
- movie-view
- navbar
- profile-view
- registration-view

The main-view contains the initial state of the movies, selectedMovie, and user variables. It also contains several functions.

```
export default class MainView extends React.Component {
    constructor() {
        super();
        // Initial state is set to null
        this.state = {
            movies: [],
            selectedMovie: null,
            user: null,
        };
    }
```

The getMovies function gets the movies from the server and sets the state of the movies variable to the response data.

```
getMovies(token) {
        axios.get("https://movie-api-hoover.herokuapp.com/movies", {
            headers: { Authorization: `Bearer ${token}` },
        })
            .then((response) => {
                // Assign the result to the state
                this.setState({
                    movies: response.data
                });
            })
            .catch(function (error) {
                console.log(error);
            });
    }
```

The componentDidMount function gets a token from local storage and if it exists, it sets the state of the user variable to the user item that is set in local storage. It then passes the token to the getMovies function.

```
componentDidMount() {
        let accessToken = localStorage.getItem("token");
        if (accessToken !== null) {
            this.setState({
                user: localStorage.getItem("user"),
            });
            this.getMovies(accessToken);
        }
    }
```

The setSelectedMovie function takes newSelectedMovie as a parameter and sets the state of the selectedMovie variable to newSelectedMovie.

```
setSelectedMovie(newSelectedMovie) {
        this.setState({
            selectedMovie: newSelectedMovie,
        });
    }
```

The onLoggedIn function takes authData as a parameter and sets the state of the user variable as authData.user.Username. It also sets the token and user in local storage.

```
onLoggedIn(authData) {
        console.log(authData);
        this.setState({
            user: authData.user.Username,
        });

        localStorage.setItem("token", authData.token);
        localStorage.setItem("user", authData.user.Username);
        this.getMovies(authData.token);
    }
```

The onLoggedOut function removes the token and user from local storage and sets the state of the user variable to null.

```
onLoggedOut() {
```

```
        localStorage.removeItem("token");
        localStorage.removeItem("user");
        this.setState({
            user: null,
        });
    }
```

The render function uses React Router to display different views based on whether a user is logged in or not. If the user is logged in, it displays the main view with all the movies. The render function also includes routes and views for <MovieView />, <RegistrationView />, <GenreView />, <DirectorView />, <ProfileView />, <Menubar />, and <FooterView />.

```
render() {
        const { movies, selectedMovie, user } = this.state;

        /* MovieView shows one movie, MoveCard shows all movies */

        return (
            <Router>
                <Menubar user={user} />
                <Row className="justify-content-md-center">
                    <Route
                        exact
                        path="/"
                        render={() => {
                            if (!user) {
                                return <LoginView onLoggedIn={(user) =>
this.onLoggedIn(user)} />;
                            }
                            if (movies.length === 0) {
                                return <div className="main-view" />;
                            }
                            return movies.map((m) => (

                                <Col md={3} key={m._id}>
                                    <MovieCard movie={m} />
                                </Col>
                            ));
                        }}
                    />
                </Row>
```

```
                <Route
                    exact
                    path="/movies/:movieId"
                    render={({ match, history }) => {
                        if (!user) {
                            return <LoginView OnLoggedIn={(user) =>
this.onLoggedIn(user)} />;
                        }
                        if (movies.length === 0) {
                            return <div className="main-view" />;
                        }
                        return (
                            <Row className="justify-content-md-center">
                                <Col md={8}>
                                    <MovieView
                                        movie={movies.find((movie) =>
movie._id === match.params.movieId)}
                                        onBackClick={() =>
history.goBack()}
                                    />
                                </Col>
                            </Row>
                        );
                    }}
                />

                <Route
                    path="/register"
                    render={() => {
                        if (user) {
                            return <Redirect to="/" />;
                        }
                        return (
                            <Row className="justify-content-md-center">
                                <Col>
                                    <RegistrationView />
                                </Col>
                            </Row>
                        );
                    }}
                />

                <Route
```

```
                    exact
                    path="/genres/:name"
                    render={({ match, history }) => {
                        if (!user) {
                            return <LoginView onLoggedIn={(user) =>
this.onLoggedIn(user)} />;
                        }
                        if (movies.length === 0) {
                            return <div className="main-view" />;
                        }
                        return (
                            <Row className="justify-content-md-center">
                                <Col md={8}>
                                    <GenreView
                                        genre={
                                            movies.find((m) => m.Genre.Name
=== match.params.name).Genre
                                        }
                                        onBackClick={() =>
history.goBack()}
                                    />
                                </Col>
                            </Row>
                        );
                    }}
                />

                <Route
                    exact
                    path="/directors/:name"
                    render={({ match, history }) => {
                        if (!user) {
                            return <LoginView OnLoggedIn={(user) =>
this.onLoggedIn(user)} />;
                        }
                        if (movies.length === 0) {
                            return <div className="main-view" />;
                        }
                        return (
                            <Row className="justify-content-md-center">
                                <Col md={8}>
                                    <DirectorView
                                        director={
```

```
                                        movies.find((m) =>
m.Director.Name === match.params.name).Director
                                    }
                                    onBackClick={() =>
history.goBack()}
                                />
                            </Col>
                        </Row>
                    );
                }}
            />


            <Route
                path="/users/:username"
                render={({ history, match }) => {
                    if (!user) {
                        return <LoginView onLoggedIn={(user) =>
this.onLoggedIn(user)} />;
                    }
                    if (movies.length === 0) {
                        return <div className="main-view" />;
                    }
                    return (
                        <ProfileView
                            history={history}
                            movies={movies}
                            user={user === match.params.username}
                        />
                    );
                }}
            />
            <FooterView />
        </Router>
    );
  }
}
```

DirectorView contains the logic for rendering the director name, director bio, director birth, director death, and a back button.

```
import React from 'react';
```

```
import PropTypes from 'prop-types';
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';
import { Link } from "react-router-dom";
import "./director-view.scss";

export class DirectorView extends React.Component {

    render() {
        const { director, onBackClick } = this.props;

        return (
            <Card>
                <Card.Body>
                    <Card.Title>{director.Name}</Card.Title>
                    <Card.Text>{director.Bio}</Card.Text>
                    <Card.Text>Born: {director.Birth}</Card.Text>
                    <Card.Text>Died: {director.Death}</Card.Text>
                    <Button onClick={() => { onBackClick();
}}>Back</Button>
                </Card.Body>
            </Card>
        );
    }
}

DirectorView.propTypes = {
    onBackClick: PropTypes.func.isRequired
};
```

FooterView gets the current date and displays it in a footer bar.

```
import React from 'react';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';
import Container from 'react-bootstrap/Container';
import "./footer.scss";

export class FooterView extends React.Component {
```

```
    render() {

        let year = new Date().getFullYear();

        return (
            <footer className="footer fixed-bottom">
                <p>MYFLIX (c) {year}</p>
            </footer>
        )
    }
}
```

GenreView contains the logic for rendering the genre name, genre description, and a back button.

```
import React from 'react';
import PropTypes from 'prop-types';
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';
import { Link } from "react-router-dom";
import "./genre-view.scss";

export class GenreView extends React.Component {

    render() {
        const { genre, onBackClick } = this.props;

        return (
            <Card>
                <Card.Body>
                    <Card.Title>{genre.Name}</Card.Title>
                    <Card.Text>{genre.Description}</Card.Text>
                    <Button onClick={() => { onBackClick();
}}>Back</Button>
                </Card.Body>
            </Card>
        );
    }
}
```

```
GenreView.propTypes = {
    onBackClick: PropTypes.func.isRequired
};
```

LoginView contains several functions and a login form.

The validate function validates user input from the form and displays an error message if the information is invalid.

```
const validate = () => {
        let isReq = true;
        if (!username) {
            setUsernameErr('Username Required');
            isReq = false;
        } else if (username.length < 2) {
            setUsernameErr('Username must be 2 characters long');
            isReq = false;
        }
        if (!password) {
            setPasswordErr('Password Required');
            isReq = false;
        } else if (password.length < 6) {
            setPassword('Password must be 6 characters long');
            isReq = false;
        }

        return isReq;
    }
```

The handleSubmit function posts the username to the server, and if there is a response, it sets the data variable to the response.data.

```
const handleSubmit = (e) => {
        e.preventDefault();
        const isReq = validate();
        if (isReq) {
            /* Send a request to the server for authentication */
            axios.post('https://movie-api-hoover.herokuapp.com/login', {
                Username: username,
                Password: password
            })
```

```
                .then(response => {
                    const data = response.data;
                    props.onLoggedIn(data);
                })
                .catch(e => {
                    console.log('no such user')
                });
        }
    };
```

MovieCard contains the logic for rendering the movie image, movie title, movie description, and link to open the individual movie.

```
import React from 'react';
import PropTypes from 'prop-types';
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';
import { Link } from "react-router-dom";
import "./movie-card.scss";

export class MovieCard extends React.Component {
    render() {
        const { movie } = this.props;

        return (
            <Card>
                <Card.Img variant="top" src={movie.ImagePath}
crossorigin="anonymous" />
                <Card.Body>
                    <Card.Title>{movie.Title}</Card.Title>
                    <Card.Text>{movie.Description}</Card.Text>
                    <Link to={`/movies/${movie._id}`}>
                        <Button>Open</Button>
                    </Link>
                </Card.Body>
            </Card>
        );
    }
}
```

```
MovieCard.propTypes = {
    movie: PropTypes.shape({
        Title: PropTypes.string.isRequired,
        Description: PropTypes.string.isRequired,
        ImagePath: PropTypes.string.isRequired
    })
};
```

Menubar contains several functions and the navbar.

The onLoggedOut function clears local storage and opens the login page.

```
const onLoggedOut = () => {
        localStorage.clear();
        window.open("/", "_self");
    }
```

The isAuth function checks to see whether the user is authorized by checking to see if a token is available in local storage and returning false if it is not.

```
const isAuth = () => {
        if (typeof window == "undefined") {
            return false;
        }
        if (localStorage.getItem("token")) {
            return localStorage.getItem("token");
        } else {
            return false;
        }
    }
```

ProfileView contains several functions, the logic for rendering a favorite movies section, and a form where the user can update their username, password, email, and birthday or choose to delete their profile.

I first set the following variables to null.

```
constructor() {
        super();

        this.state = {
```

```
            Username: null,
            Password: null,
            Email: null,
            Birthday: null,
            FavoriteMovies: [],
        };
    }
```

The componentDidMount function calls the getUserDetails function.

```
componentDidMount() {
        this.getUserDetails();
    }
```

This function sets the Username variable to the user item in local storage and sets the token variable to the token item in local storage. It then gets the user details from the server and sets the state of Username, Password, Email, Birthday, and FavoriteMovies to what is returned from the server.

```
getUserDetails() {
        const Username = localStorage.getItem("user");
        let token = localStorage.getItem("token");

        axios

.get(`https://movie-api-hoover.herokuapp.com/users/${Username}`, {
                headers: { Authorization: `Bearer ${token}` },
            })
            .then((response) => {
                this.setState({
                    Username: response.data.Username,
                    Password: response.data.Password,
                    Email: response.data.Email,
                    Birthday: response.data.Birthday.split('T')[0],
                    FavoriteMovies: response.data.FavoriteMovies,
                });
            })
            .catch(function (error) {
                console.log(error);
            });
    }
```

The functions setUsername, setPassword, setEmail, and setBirthday take input from the form and set the state of the Username, Password, Email, and Birthday variables to the input value when they are changed.

```
setUsername(input) {
        this.setState({ Username: input });
    }

    setPassword(input) {
        this.setState({ Password: input });
    }

    setEmail(input) {
        this.setState({ Email: input });
    }

    setBirthday(input) {
        this.setState({ Birthday: input });
    }
```

The editUserDetails function gets the user and token from local storage and updates the server with the current state of Username, Password, Email, and Birthday when the “Update Information” button is clicked. If the username has been updated, it sets the user item in local storage to the new username, displays an alert, and calls the getUserDetails function again.

```
editUserDetails = (e) => {
        e.preventDefault();

        const Username = localStorage.getItem("user");
        let token = localStorage.getItem("token");

        axios
            .put(
                `https://movie-api-hoover.herokuapp.com/users/${Username}`,
                {
                    Username: this.state.Username,
                    Password: this.state.Password,
                    Email: this.state.Email,
                    Birthday: this.state.Birthday,
                },
                {
```

```
                    headers: { Authorization: `Bearer ${token}` },
                }
            )
            .then((response) => {
                this.setState({
                    Username: response.data.Username,
                    Password: response.data.Password,
                    Email: response.data.Email,
                    Birthday: response.data.Birthday,
                });
                localStorage.setItem("user", this.state.Username);
                alert(`Profile has been updated.`);
                this.getUserDetails();
            })
            .catch((response) => {
                console.error(response);
            });
    };
```

The deleteProfile function displays a confirmation message before allowing users to delete their profile. If they click yes, it retrieves the user and token from local storage, and deletes the user from the server. Then it displays an alert that the profile has been deleted, clears local storage, and opens the login page.

```
deleteProfile = (e) => {
        e.preventDefault();

        let result = confirm(`Are you sure you want to delete your
profile?`);

        if (result) {
            const Username = localStorage.getItem("user");
            let token = localStorage.getItem("token");

            axios

.delete(`https://movie-api-hoover.herokuapp.com/users/${Username}`, {
                    headers: { Authorization: `Bearer ${token}` },
                })
                .then((response) => {
                    alert(`Profile has been deleted.`);
                    localStorage.clear();
```

```
                    window.open("/", "_self"); // So the page will open in
the current tab
                })
                .catch(function (error) {
                    console.log(error);
                });
        }
    };
```

The removeFavorite function takes movie as a parameter, gets the user and token from local storage, and deletes the movie from the server. It then calls compenentDidMount function, which in turn calls the getUserDetails function.

```
removeFavorite = (movie) => {
        const Username = localStorage.getItem("user");
        let token = localStorage.getItem("token");

        axios
            .delete(

`https://movie-api-hoover.herokuapp.com/users/${Username}/movies/${movie._i
d}`,
                {
                    headers: { Authorization: `Bearer ${token}` },
                }
            )
            .then((response) => {
                console.log(response);
                this.componentDidMount();
            })
            .catch(function (error) {
                console.log(error);
            });
    };
```

The render function displays the favorite movies section and the form where users can update their profile information or delete their profile.

```
render() {

        const { Username, Password, Email, Birthday, FavoriteMovies } =
```

```
this.state
        const { movies } = this.props;

        return (
            <>
                <Row className="justify-content-md-center">
                    {FavoriteMovies.length === 0 ? (
                        <p className="white-text">You have no favorite
movies.</p>
                    ) : (
                        movies
                            .filter((movie) =>
FavoriteMovies.includes(movie._id))
                            .map((movie) => (

                                <Col md={3}>
                                    <Card className="favorite-movie
card-content" key={movie._id}>
                                        <Card.Img
                                            className="fav-poster"
                                            variant="top"
                                            src={movie.ImagePath}
                                            crossorigin="anonymous"
                                        />
                                        <Card.Body>
                                            <Card.Title
className="movie_title">{movie.Title}</Card.Title>
                                            <Button
                                                size="sm"
                                                variant="danger"
                                                value={movie._id}
                                                onClick={(e) =>
this.removeFavorite(movie)}
                                            >
                                                Remove
                                            </Button>
                                        </Card.Body>
                                    </Card>
                                </Col>
                            ))}
                </Row>

                <Row className="justify-content-md-center">
```

```
                    <Col md={6}>
                        <Form>
                            <Form.Group controlId="formUsername">
                                <Form.Label>Username:</Form.Label>
                                <Form.Control
                                    type="text"
                                    placeholder="Enter new username"
                                    value={Username}
                                    onChange={(e) =>
this.setUsername(e.target.value)}
                                />
                            </Form.Group>
                            <Form.Group controlId="formPassword">
                                <Form.Label>Password:</Form.Label>
                                <Form.Control
                                    type="password"
                                    placeholder="Enter new password"
                                    onChange={(e) =>
this.setPassword(e.target.value)}
                                />
                            </Form.Group>
                            <Form.Group controlId="formEmail">
                                <Form.Label>Email:</Form.Label>
                                <Form.Control
                                    type="email"
                                    placeholder="Enter new email"
                                    value={Email}
                                    onChange={(e) =>
this.setEmail(e.target.value)}
                                />
                            </Form.Group>
                            <Form.Group controlId="formBirthday">
                                <Form.Label>Birthday:</Form.Label>
                                <Form.Control
                                    type="date"
                                    value={Birthday}
                                    onChange={(e) =>
this.setBirthday(e.target.value)}
                                />
                            </Form.Group>
                            <div className="buttons">
                                <Button
                                    variant="primary"
```

```
                                    type="submit"
                                    onClick={this.editUserDetails}
                                >
                                    Update Information
                                </Button>
                                <Button
                                    variant="danger"
                                    type="submit"
                                    onClick={(e) => this.deleteProfile(e)}
                                >
                                    Delete Profile
                                </Button>
                            </div>
                        </Form>
                    </Col>
                </Row>
            </>
        );
    }
}
```

RegistrationView contains two functions and the logic and the logic for rendering the registration form.

The validate function displays various errors if the username, password, and email are either not entered or entered incorrectly.

```
const validate = () => {
        let isReq = true;
        if (!username) {
            setUsernameErr('Username Required');
            isReq = false;
        } else if (username.length < 2) {
            setUsernameErr('Username must be 2 characters long');
            isReq = false;
        }
        if (!password) {
            setPasswordErr('Password Required');
            isReq = false;
        } else if (password.length < 6) {
            setPassword('Password must be 6 characters long');
            isReq = false;
```

```
        }
        if (!email) {
            setEmailErr('Email Required');
            isReq = false;
        } else if (email.indexOf('@') === -1) {
            setEmailErr('Email is invalid');
            isReq = false;
        }

        return isReq;
    }
```

The handleSubmit function first makes sure all the required form fields have been entered correctly and then posts a new user to the server. It then returns the user data from the server, alerts the user that registration was successful, and opens the login page.

```
const handleSubmit = (e) => {
        e.preventDefault();
        const isReq = validate();
        if (isReq) {
            /* Send a request to the server for authentication */
            axios.post('https://movie-api-hoover.herokuapp.com/users', {
                Username: username,
                Password: password,
                Email: email,
                Birthday: birthday
            })
                .then(response => {
                    const data = response.data;
                    console.log(data);
                    alert('Registration successful, please login!');
                    window.open('/', '_self'); // So the page will open in
the current tab
                })
                .catch(response => {
                    console.error(response);
                    alert('Unable to register');
                });
        }
    };
```

The return function displays the registration form and a link to the login page if users are already registered.

```
 return (
        <Row className="registration-view justify-content-md-center">
            <Col md={6}>
                <Form>
                    <Form.Group controlId="formUsername">
                        <Form.Label>Username:</Form.Label>
                        <Form.Control type="text" placeholder="Enter
username" value={username} onChange={e => setUsername(e.target.value)} />
                        {/* code added here to display validation error */}
                        {usernameErr && <p
className="login-error">{usernameErr}</p>}
                    </Form.Group>
                    <Form.Group controlId="formPassword">
                        <Form.Label>Password:</Form.Label>
                        <Form.Control type="password"
placeholder="Password" value={password} onChange={e =>
setPassword(e.target.value)} />
                        {/* code added here to display validation error */}
                        {passwordErr && <p
className="login-error">{passwordErr}</p>}
                    </Form.Group>
                    <Form.Group controlId="formEmail">
                        <Form.Label>Email:</Form.Label>
                        <Form.Control type="email" placeholder="Email"
value={email} onChange={e => setEmail(e.target.value)} />
                        {/* code added here to display validation error */}
                        {emailErr && <p
className="login-error">{emailErr}</p>}
                    </Form.Group>
                    <Form.Group controlId="formBirthday">
                        <Form.Label>Birthday:</Form.Label>
                        <Form.Control type="date" onChange={e =>
setBirthday(e.target.value)} />
                    </Form.Group>
                    <div className="buttons">
                        <Button variant="primary" type="submit"
onClick={handleSubmit}>
                            Register
                        </Button>
                        <p></p>
```

```
                        <p className="white-text">Already registered? <Link
to={'/'}><span className="link-underline">Sign in</span></Link> here.</p>
                    </div>
                </Form>
            </Col>
        </Row>
    );
}
```

The Results

This project was one of the hardest ones to complete during my time at CareerFoundry. There were many different functions and components to keep track of and lots of testing to make it work. When I got stuck, I used console log to help me display data, turned to Google for answers, reached out to the CareerFoundry slack channel for help, and reached out to my tutor. I learned a multitude of new things from this project, including how to use React, how to use Axios, how to use React Bootstrap, and how to use React Router. I also learned how to use conditional rendering, how to process data that is input into a form, and how to connect the front end to the back end. This was my first project using React, but I'm sure it won't be my last, and I look forward to mastering this framework.