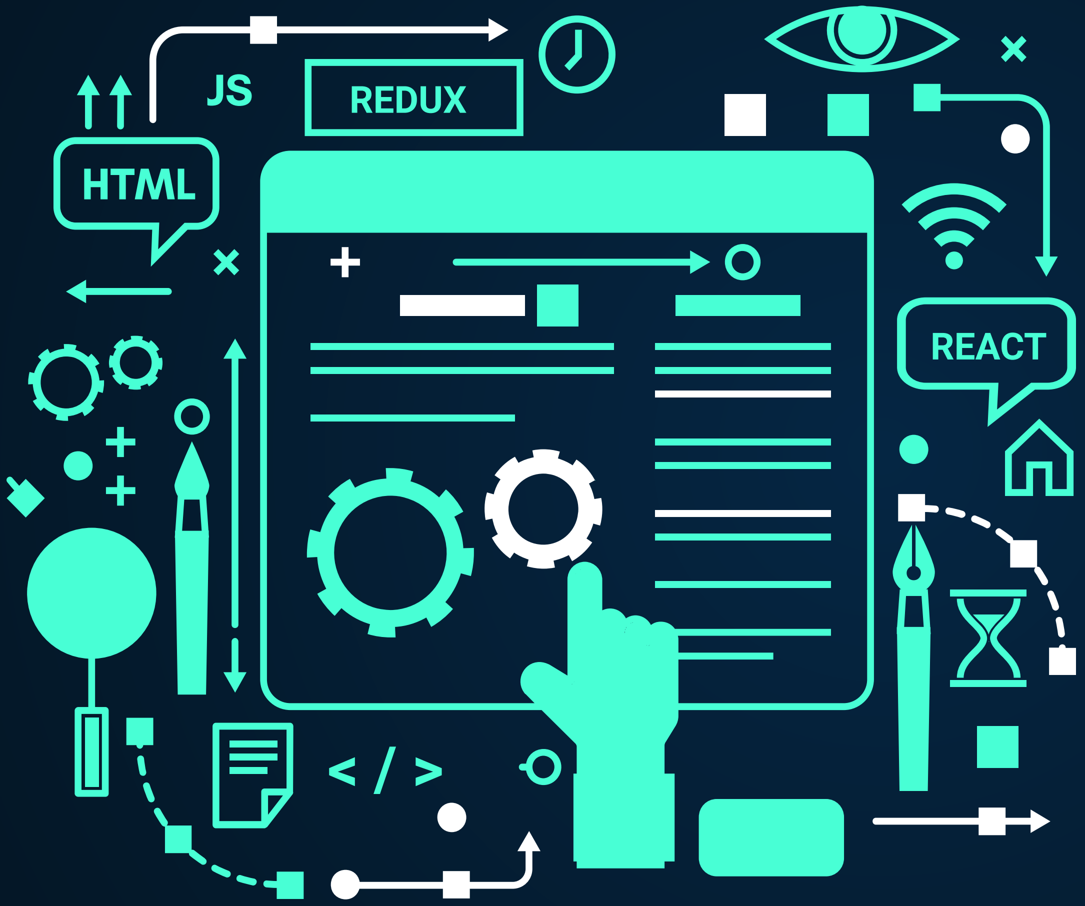
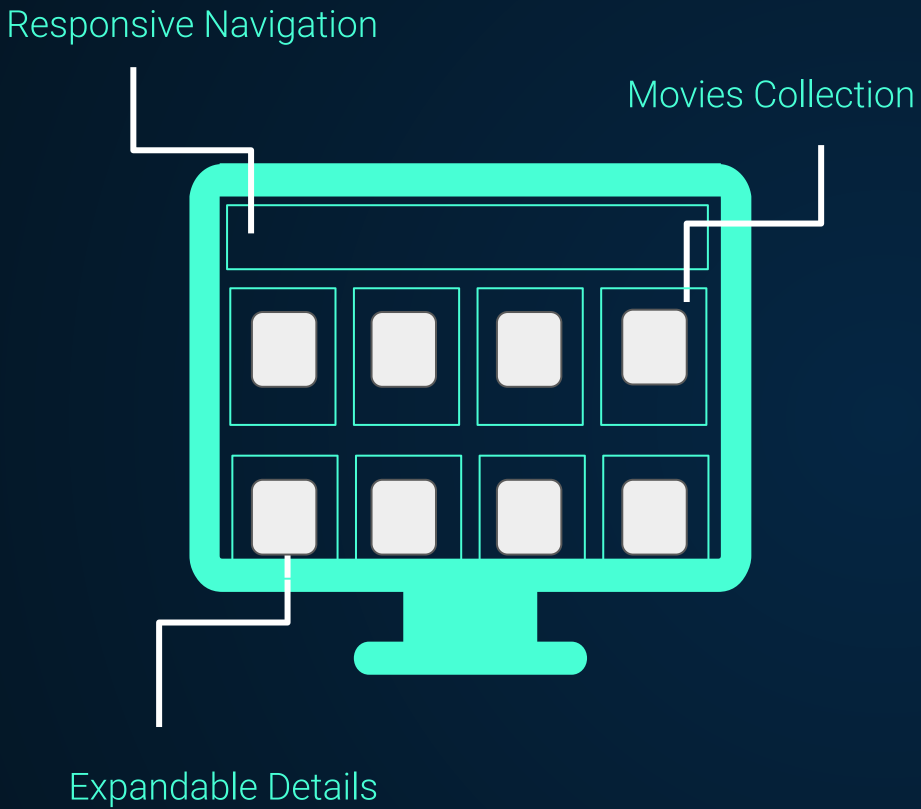


CASE STUDY: myFlix

Kelsey Flynn
February 2021

TABLE OF CONTENTS



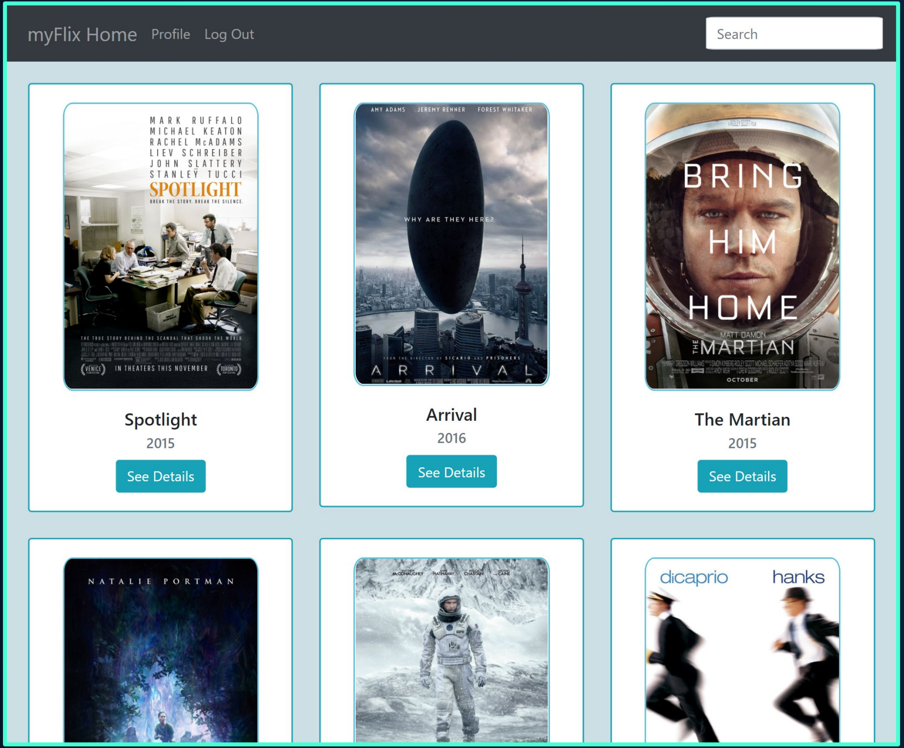
PROJECT OVERVIEW

myFlix is an movie database website, open for new users to view a collection of movies, and build up their profiles by adding their favorites.

myFlix's client-side application was created with React and Redux, and styled with React Bootstrap. It is connected to my existing server-side code (REST API and NoSQL database), which is stored on the cloud-based MongoDB Atlas.

myFlix is a MERN tech stack application, and the final product is hosted on Netlify.

SNEAK PEEK

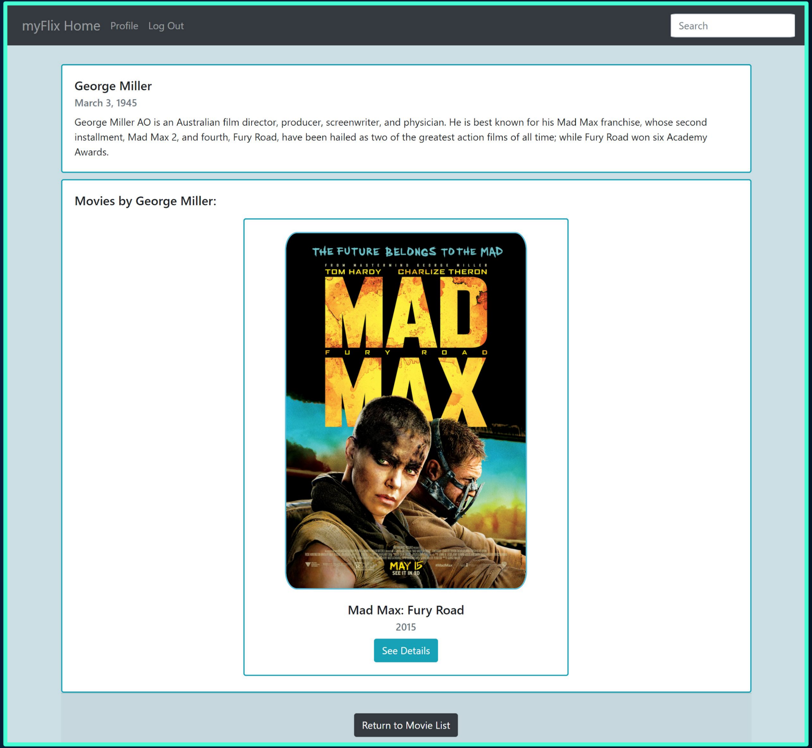


myFlix Home Profile Log Out Search

Spotlight
2015
See Details

Arrival
2016
See Details

The Martian
2015
See Details



myFlix Home Profile Log Out Search

George Miller
March 3, 1945
George Miller AO is an Australian film director, producer, screenwriter, and physician. He is best known for his Mad Max franchise, whose second installment, Mad Max 2, and fourth, Fury Road, have been hailed as two of the greatest action films of all time; while Fury Road won six Academy Awards.

Movies by George Miller:

Mad Max: Fury Road
2015
See Details

Return to Movie List

PERSONAL PROJECT GOALS

PORTFOLIO READY

One of my goals was to have this project meet the specifications of the assigned task, and also be something I personally wanted to showcase on my portfolio.

CONCEPTUAL KNOWLEDGE

This meant that I spent extra time solidifying the concepts of each stage (React, Redux, Bootstrap, etc). As I worked on this application solo, I used example repositories and video tutorials to be confident in my ability to explain what I'd steps I'd taken.

USER EXPERIENCE

I asked the key question "Would I actually use this website?" when developing **myFlix**. While I kept the code as clear and concise as possible, I also focused on having a modern and functional application.

TECHNICAL OBJECTIVES

SERVER & CLIENT-SIDE CONNECTIONS

Connected API endpoints made in server-side code to the client-side views. Endpoints included:

- Return all movies (& movie filtering)
- Return single movie
- Return genre & director info
- Create new user
- Login existing user

SINGLE PAGE REACT APP (SPA)

Learned and developed first React application with best practices in mind. Utilization of both class and functional components, included state routing SPA endpoints.

RESPONSIVE UI

myFlix required React Bootstrap for styling, and needed additional UI features such as filtering of movie titles, which was done with Redux.

TIMELINE

PROJECT BRIEF

JANUARY 20 - 25
React foundations

JANUARY 26 - FEBRUARY 2
Component creation, SPA routing

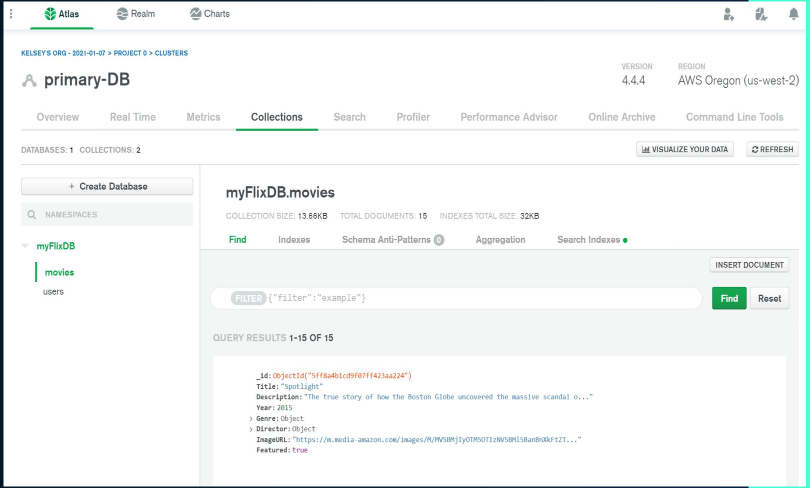
FEBRUARY 3 - 8
Bootstrap styling/UI

FEBRUARY 9 - 14
Redux implementation, final edits

DEPLOYMENT

Server-side Recap

A previous project dealt with the creation of the database, enacting CRUD on API endpoints, authentication and JWT authorization.

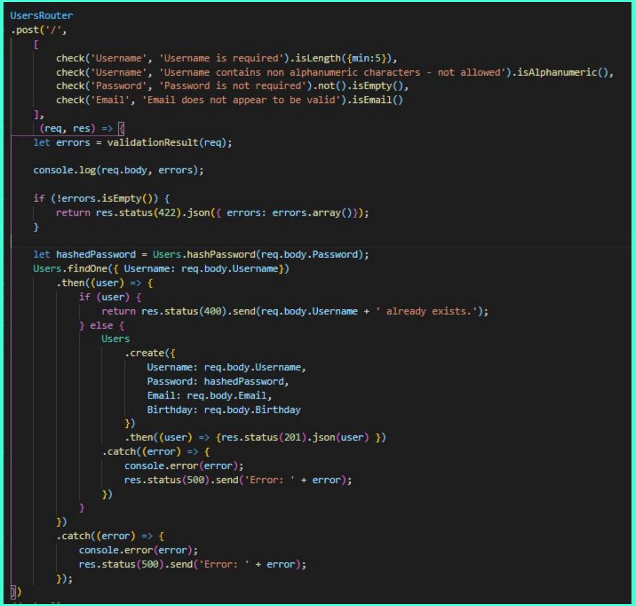


```
_id: ObjectId("5ff8a4b1cd9f07ff423aa224")
Title: "Spotlight"
Description: "The true story of how the Boston Globe uncovered the massive scandal o..."
Year: 2015
> Genre: Object
> Director: Object
ImageURL: "https://m.media-amazon.com/images/M/MV5BMjIyOTM5OTIzNV5BMl5BanBnXkFtZT..."
Featured: true
```

NoSQL collections for movies and users were created in a previous project, and hosted on MongoDB Atlas.

As a NoSQL database, Mongoose schemas were used to enforce uniformity. Other security measures included were input validation, CORS, and password hashing (bcrypt).

Endpoints were tested in Postman for accuracy before moving on to the client-side code.



```
UsersRouter
.post('/',
    [
        check('Username', 'Username is required').isLength({min:5}),
        check('Username', 'Username contains non alphanumeric characters - not allowed').isAlphanumeric(),
        check('Password', 'Password is not required').not().isEmpty(),
        check('Email', 'Email does not appear to be valid').isEmail()
    ],
     (req, res) => {
    let errors = validationResult(req);

    console.log(req.body, errors);

    if (!errors.isEmpty()) {
        return res.status(422).json({ errors: errors.array()});
    }

    let hashedPassword = Users.hashPassword(req.body.Password);
    Users.findOne({ Username: req.body.Username})
        .then((user) => {
            if (user) {
                return res.status(400).send(req.body.Username + ' already exists.');
            } else {
                Users
                    .create({
                        Username: req.body.Username,
                        Password: hashedPassword,
                        Email: req.body.Email,
                        Birthday: req.body.Birthday
                    })
                    .then((user) => {res.status(201).json(user) })
                .catch((error) => {
                    console.error(error);
                    res.status(500).send('Error: ' + error);
                })
            }
        })
        .catch((error) => {
            console.error(error);
            res.status(500).send('Error: ' + error);
        });
})
```

STAGE 1 : REACT-APP

IMPORTANCE

Working with a framework like React gave me insight of what to expect from others, such as Angular and Vue, and what the differences are between each one.

DECISIONS

I used both functional and class components, based on their usage and relation to their parent. This gave me more experience on how to troubleshoot each type.

SUCCESSES

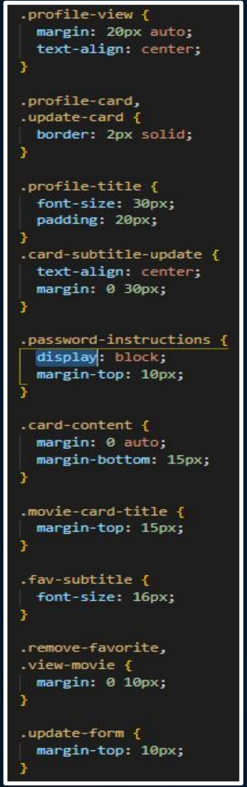
Improved skills: HTML, JavaScript, CSS..

New skills: Building React app framework from scratch, without use of cra-template.

CHALLENGES

Learning and utilizing React for the first time was an intense learning curve, so I spent extra time building my foundational knowledge before diving in.

STAGE 2 : BOOTSTRAP STYLING

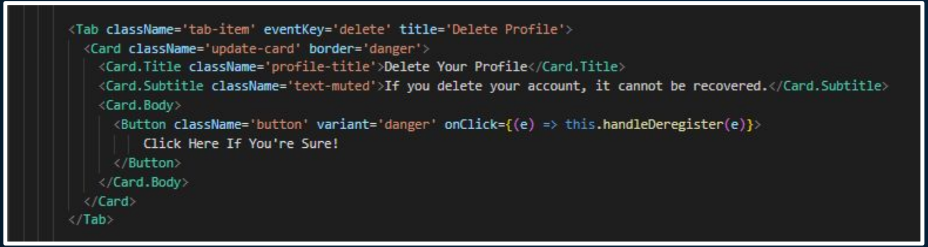


```scss
.profile-view {
  margin: 20px auto;
  text-align: center;
}

.profile-card,
.update-card {
  border: 2px solid;
}

.profile-title {
  font-size: 30px;
  padding: 20px;
}
.card-subtitle-update {
  text-align: center;
  margin: 0 30px;
}

.password-instructions {
  display: block;
  margin-top: 10px;
}

.card-content {
  margin: 0 auto;
  margin-bottom: 15px;
}

.movie-card-title {
  margin-top: 15px;
}

.fav-subtitle {
  font-size: 16px;
}

.remove-favorite,
.view-movie {
  margin: 0 10px;
}

.update-form {
  margin-top: 10px;
}
```

ProfileView.scss

IMPORTANCE

Without styling, the render statements of the React components give a very telegraphic representation in the virtual DOM.



```jsx
<Tab className='tab-item' eventKey='delete' title='Delete Profile'>
  <Card className='update-card' border='danger'>
    <Card.Title className='profile-title'>Delete Your Profile</Card.Title>
    <Card.Subtitle className='text-muted'>If you delete your account, it cannot be recovered.</Card.Subtitle>
    <Card.Body>
      <Button className='button' variant='danger' onClick={(e) => this.handleDeregister(e)}>
        Click Here If You're Sure!
      </Button>
    </Card.Body>
  </Card>
</Tab>
```

ProfileView.jsx

SUCCESSES

I was very pleased with the end result of the styling, my experience with using basic Bootstrap on previous work was a great stepping stone to applying it to a React app.

CHALLENGES

By using React Bootstrap, rather than basic CSS/SCSS, the styling was often “split” between a component’s .jsx file and its .scss file, such as the snippets to the side. Due to this, and the fact parent components had control over their children, sometimes it was difficult to locate exactly where a styling feature was coming from.

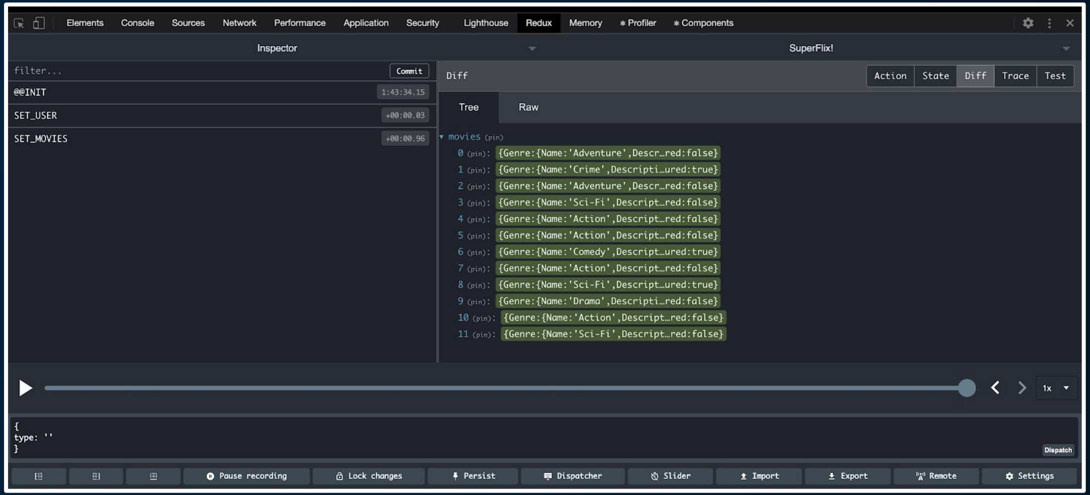
STAGE 3 : REDUX

IMPORTANCE

Implementing Redux at this stage was valuable as it allows scaling up to be easier the app grows and more components are made.

SUCCESSES

I had an easier time with Redux than I did initially with React, and was especially grateful to the Redux Chrome Extension tools, as I most of my troubleshooting through it.

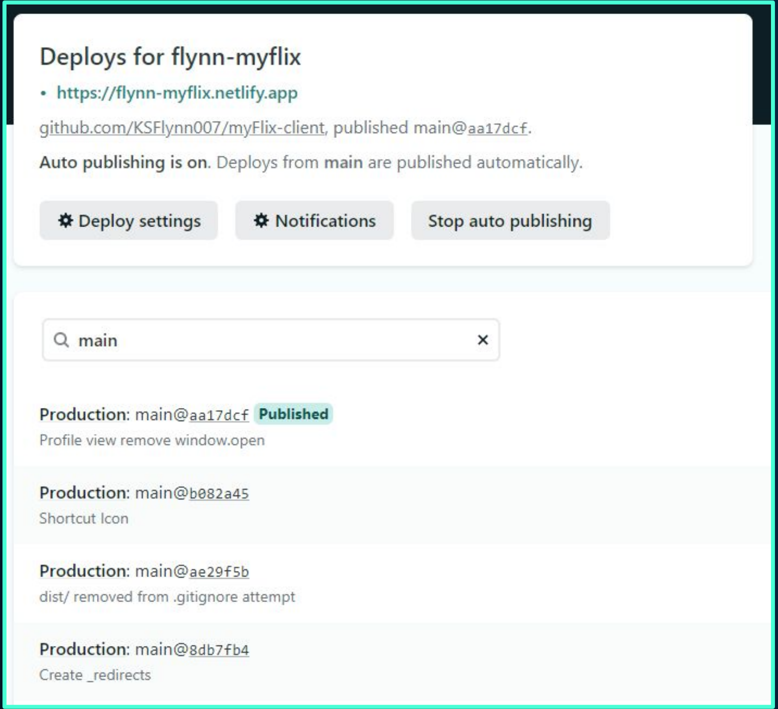


```
@@INIT        1:43:34.15
SET_USER      +00:00.03
SET_MOVIES    +00:00.96
```

```
▾ movies (pin)
  0 (pin): {Genre:{Name:'Adventure',Descr…red:false}
  1 (pin): {Genre:{Name:'Crime',Descripti…ured:true}
  2 (pin): {Genre:{Name:'Adventure',Descr…red:false}
  3 (pin): {Genre:{Name:'Sci-Fi',Descript…red:false}
  4 (pin): {Genre:{Name:'Action',Descript…red:false}
  5 (pin): {Genre:{Name:'Action',Descript…red:false}
  6 (pin): {Genre:{Name:'Comedy',Descript…ured:true}
  7 (pin): {Genre:{Name:'Action',Descript…red:false}
  8 (pin): {Genre:{Name:'Sci-Fi',Descript…ured:true}
  9 (pin): {Genre:{Name:'Drama',Descripti…red:false}
  10 (pin): {Genre:{Name:'Action',Descript…red:false}
  11 (pin): {Genre:{Name:'Sci-Fi',Descript…red:false}
```

```
{
type: ''
}
```

STAGE 4 : DEPLOYMENT



Deploys for flynn-myflix

- https://flynn-myflix.netlify.app

github.com/KSFlynn007/myFlix-client, published main@aa17dcf.

Auto publishing is on. Deploys from **main** are published automatically.

Production: main@aa17dcf **Published**
Profile view remove window.open

Production: main@b082a45
Shortcut Icon

Production: main@ae29f5b
dist/ removed from .gitignore attempt

Production: main@8db7fb4
Create _redirects

SUCCESSES

New skills: Netlify hosting; experiencing alternative hosting methods is valuable to understand what platform is best for certain projects.

So far Netlify has been my favorite hosting site, simply because of its link to Github, so that when a repo is updated, a new version is deployed automatically.

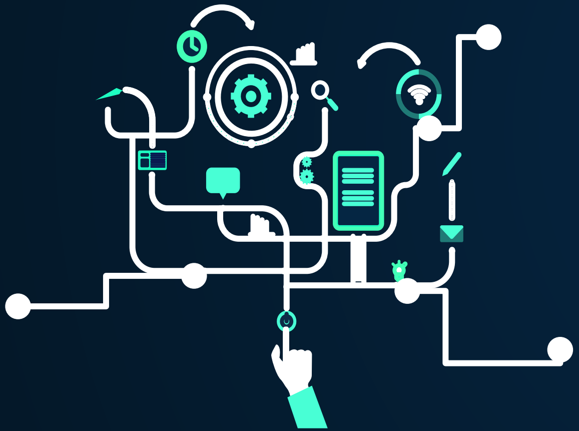
CHALLENGES

The SPA format of React caused some confusion in Netlify, and my first deployed version couldn't find the route to the endpoint '*/register* ', though had no problem with any others.

The 404 error was fixed by adding a '*_redirects*' file into my dist folder to preserve the virtual DOM.

LOOKING BACK

Figuring out how my components, their props and states interacted with each other sometimes felt like this....



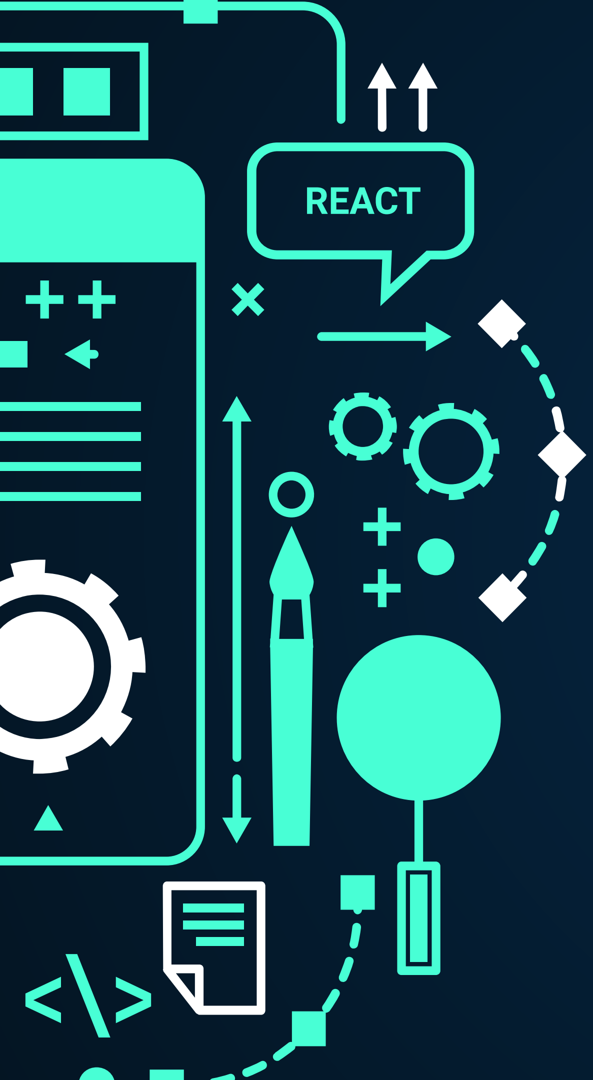
So I kept copious in-code notes and made big changes within new branches on Git and merged into the main branch when finished. These techniques enabled me to keep the flow of the app straight.

The most time consuming errors I encountered were not from functions, where I assumed I was going awry. They were either small typos or outstanding connection issues in my server-side code.

Lesson learned:
Looking at both smallest and biggest picture is as important as the individual components.

If I could do anything different in this project, I would have created more child components under ProfileView.jsx to separate out major CRUD actions, like updating user information, rather than keeping it all in one file.

This project was my most difficult to date, and as I continue to improve **myFlix**, I will work on building a more robust collection of movies and developing new visuals to allow for a more seamless layout of movie cards.



Thank you!

- To visit the live application, click this link:

  https://flynn-myflix.netlify.app/

- For any questions, please reach out to me at:

  kelseyflynnn@gmail.com
  +1.403.808.2573

- Or check out out what else I'm working on at:

  https://www.linkedin.com/in/kelsey-flynn-429464a6/

  https://github.com/KSFlynn007/

CREDITS

- Presentation template by Slidesgo
- Icons by Flaticon
- Infographics by Freepik
- Images created by Freepik
- Author introduction slide photo created by Freepik
- Text & Image slide photo created by Freepik