Deployed project here: LINK HERE

Start

Installation

1. Clone or download the repo
2. Run 'npm i' in the CLI
3. Run 'npm start' in the CLI

Timeframe

6 days

Team

- Sonia Choudhury

@soniacweb

- Ustin Vaskin

@UstinVaskin

Overview

For FilmLovers

A Web Site- TV & Movie Database with custom-built filters using Third-party APIs.

Technologies used

Technologies: API, CSS, Git, GitHub, React,React Hooks, UX, Node.js, Heroku

Componets and Elements

Actor, Grid, Header, HeroImage, LoadMoreBtn, Modal, MovieInfo, MovieInfoBar, MovieThumb, Navigation, SearchBar, Spinner/Loading

Home Page

The homepage is divided into four sections:

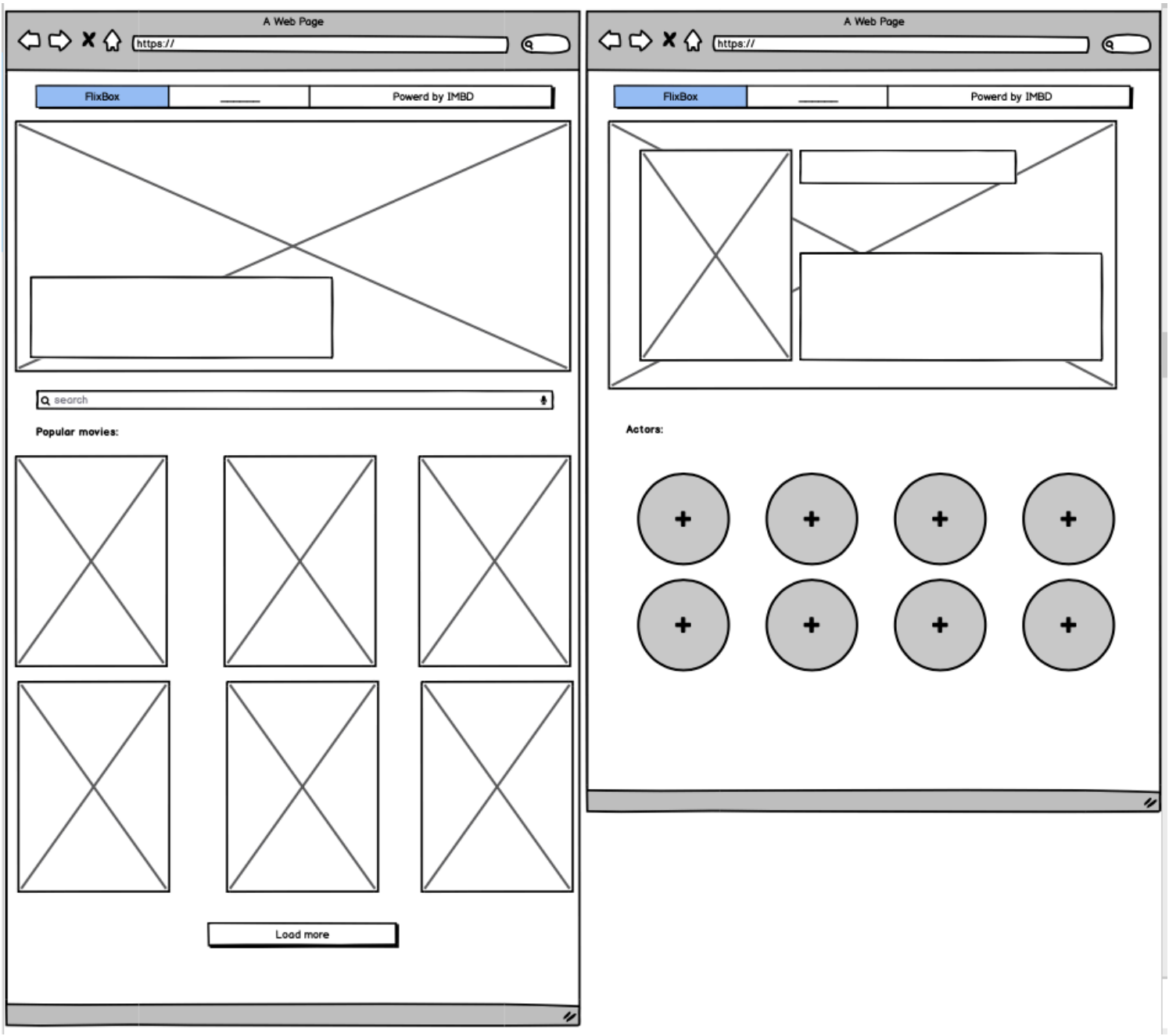
Header

The main hero

Search bar

Popular Movies

Wireframes



We used hooks to fetch movies

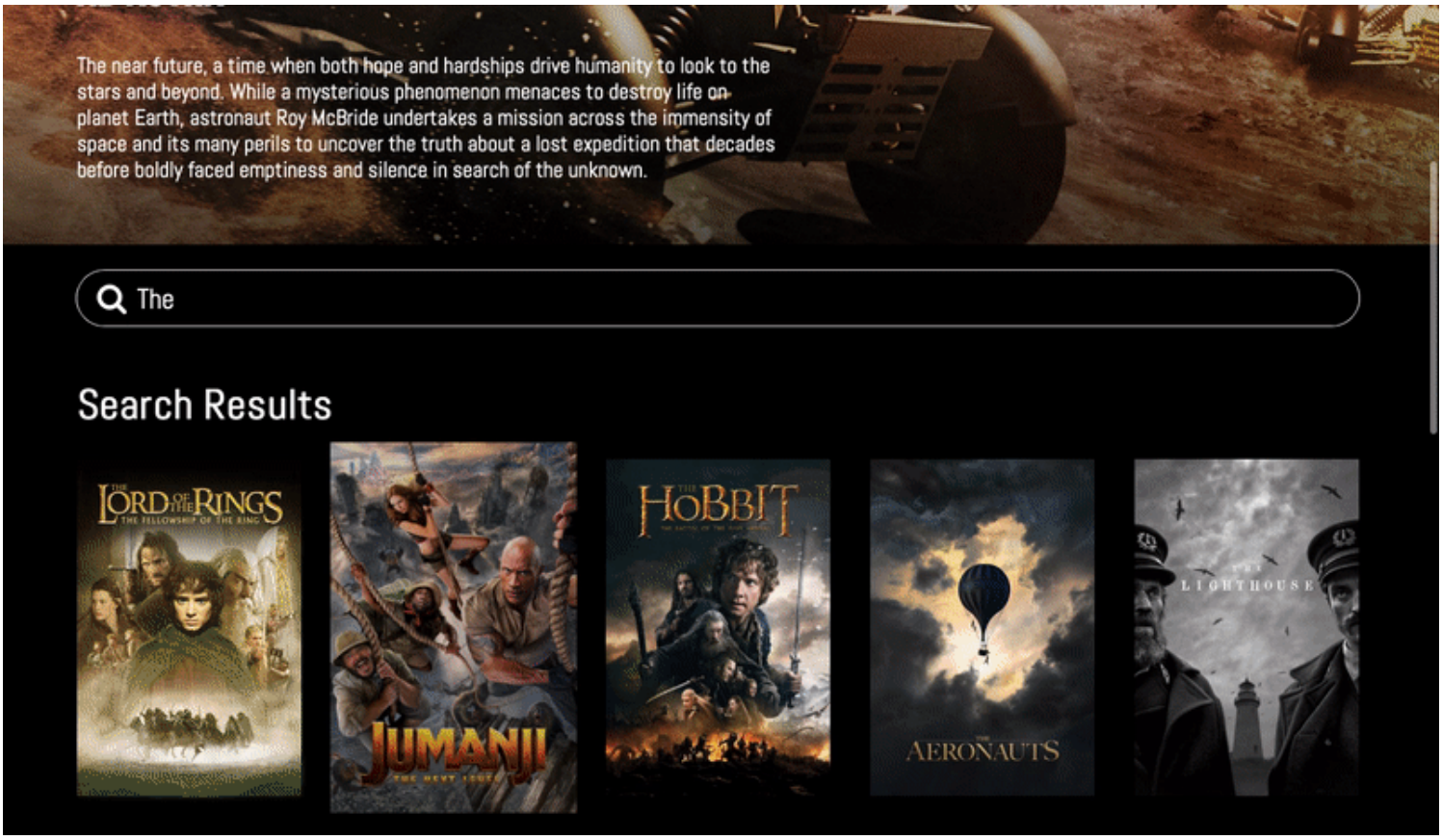
```
import { useState, useEffect } from 'react'
import { POPULAR_BASE_URL, } from '../../config'

//custom hook

 export const useHomeFetch = () => {
  const [state, setState] = useState({ movies: [] })
  const [loading, setLoading] =  useState(false) //default
  const [error, setError] = useState(false) //default
```

```
  // console.log(state)


  const fetchMovies = async endpoint => {
    setError(false)
    setLoading(true)

    const isLoadMore= endpoint.search('page')
  try {
  const result = await (await fetch(endpoint)).json()
  // console.log(result)

    setState(prev => ({
    ...prev,
  movies:
  isLoadMore !== -1
  ? [...prev.movies, ...result.results]
  : [...result.results],
  heroImage: prev.heroImage || result.results[0],
  currentPage: result.page,
  totalPages: result.total_pages,
      }))
  } catch (error) {
      setError(true)
        console.log(error)
  }
    setLoading(false)

  }

  useEffect(() => {
    fetchMovies(POPULAR_BASE_URL)
  }, [])

  return [{ state, loading, error }, fetchMovies]
 }
```

Search movie function

```
const searchMovies = search => {
const endpoint = search ? SEARCH_BASE_URL + search : POPULAR_BASE_URL

setSearchTerm(search)
fetchMovies(endpoint)
```

```
}
```



Load more movies function

```
const loadMoreMovies = () => {
const searchEndpoint = `${SEARCH_BASE_URL}${searchTerm}&page=${currentPage + 1}`
const popularEndpoint = `${POPULAR_BASE_URL}&page=${currentPage + 1}`

const endPoint = searchTerm ? searchEndpoint : popularEndpoint

fetchMovies(endPoint)
}
```

Movie Show page

- Our approach was to first get the key information about the movie displayed on the page. (image, genre, year etc), and then render the information on the page;

Render

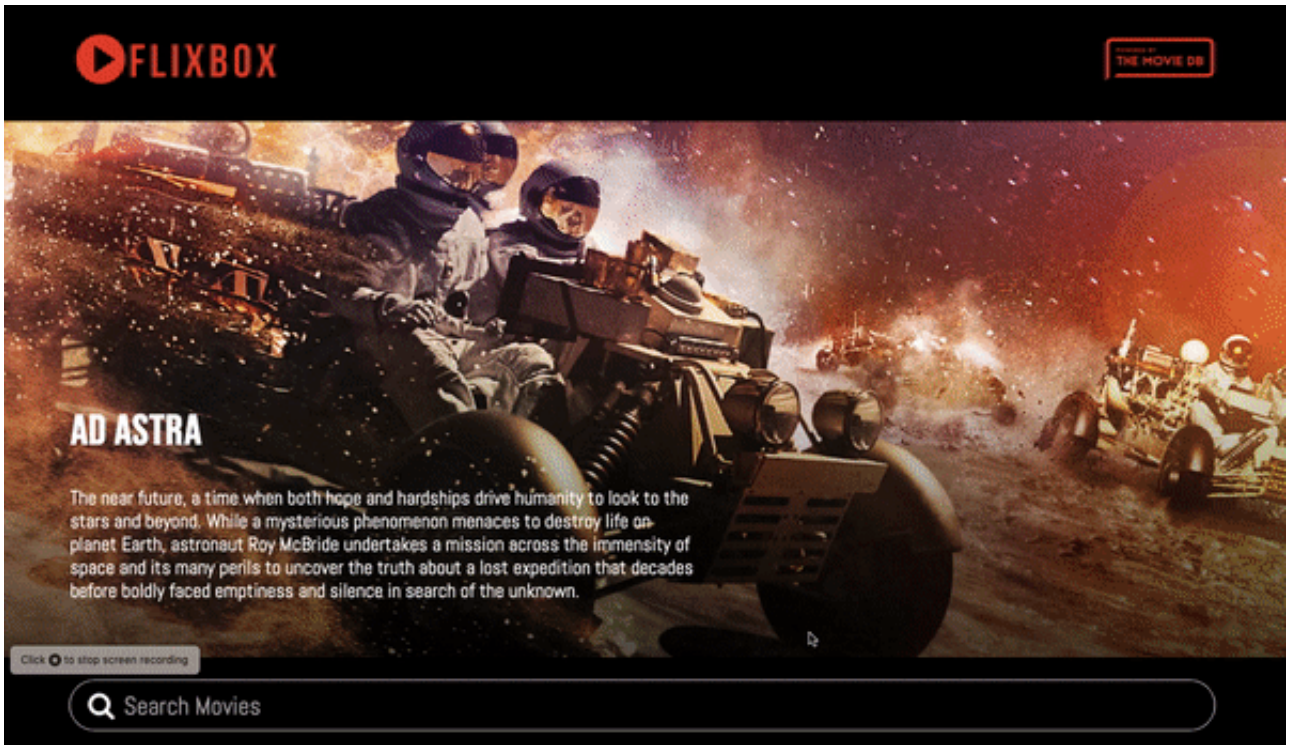
```
<Navigation movie={movie.original_title} />
<MovieInfo movie={movie} />
<MovieInfoBar
time={movie.runtime}
budget={movie.budget}
revenue={movie.revenue}
/>
<Grid header="Actors">
  {movie.actors.map(actor => (
      <Actor key={actor.credit_id} actor={actor} />
  ))}
</Grid>
```

Fetching a movie

```
export const useMovieFetch = movieId => {
const [state, setState]= useState({})
const [loading, setLoading]= useState(true)
const [error, setError]= useState(false)

const fetchData= useCallback(async () => {
setError(false)
setLoading(true)

try {
const endpoint= `${API_URL}movie/${movieId}${API_KEY}`
const result= await (await fetch(endpoint)).json()
// console.log(result)

const creditsEndpoint=`${API_URL}movie/${movieId}/credits${API_KEY}`
const creditsResult= await (await fetch(creditsEndpoint)).json()
// console.log(creditsResult)
const directors = creditsResult.crew.filter(
  member => member.job === 'Director'
)
  setState({
    ...result,
    actors: creditsResult.cast,
    directors,
  })
```

```
} catch (error) {
 setError(true)
}
setLoading(false)
}, [movieId])

useEffect(() => {
  fetchData()
}, [fetchData])

return [state, loading, error]
}

export default useMovieFetch
```

Final Product



Modifications:

Filtering, Trailer, Comments/ Logins, Refactoring, Accessibility