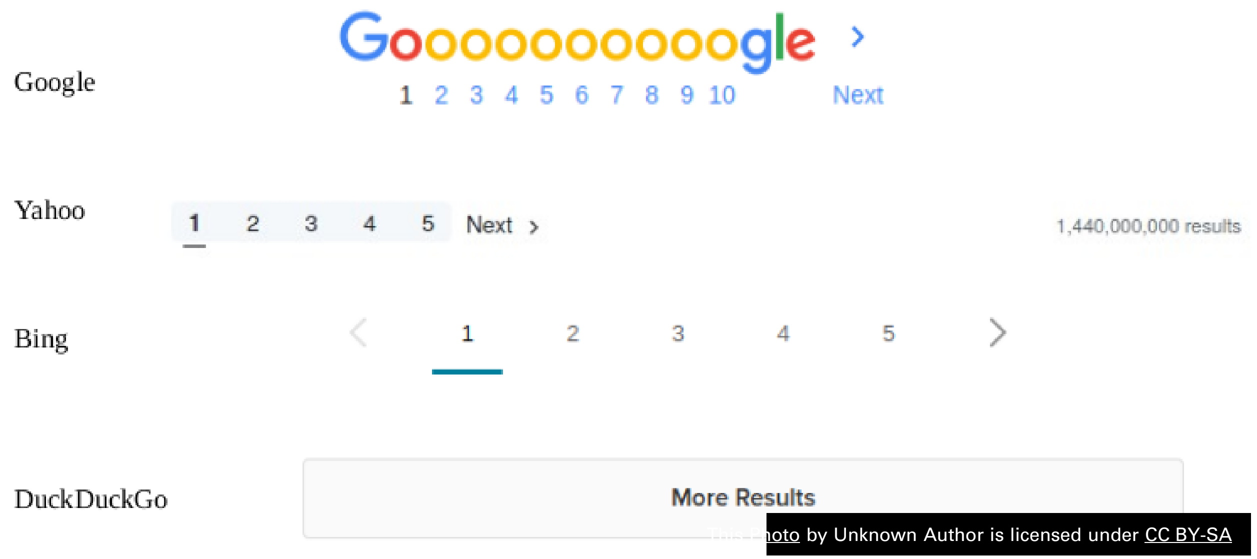
REACT APPS AND API INTEGRATION 1

Frank Walsh

Asside: Server Side Pagination

- Get Movies API:
  GET /api/movies
- Returns all Movies. Our DB is small but a real db will have thousands of movies
- Need Pagination
- Pagination means displaying a small number of results, by a page.
- If there are only few pages, we can fetch all items and paginate on the client
- For larger data sets, et the server do the work.
- Server side pagination is better for:
  - Large data set
  - Faster initial page load
  - Accessibility for those not running JavaScript
  - Complex view business logic



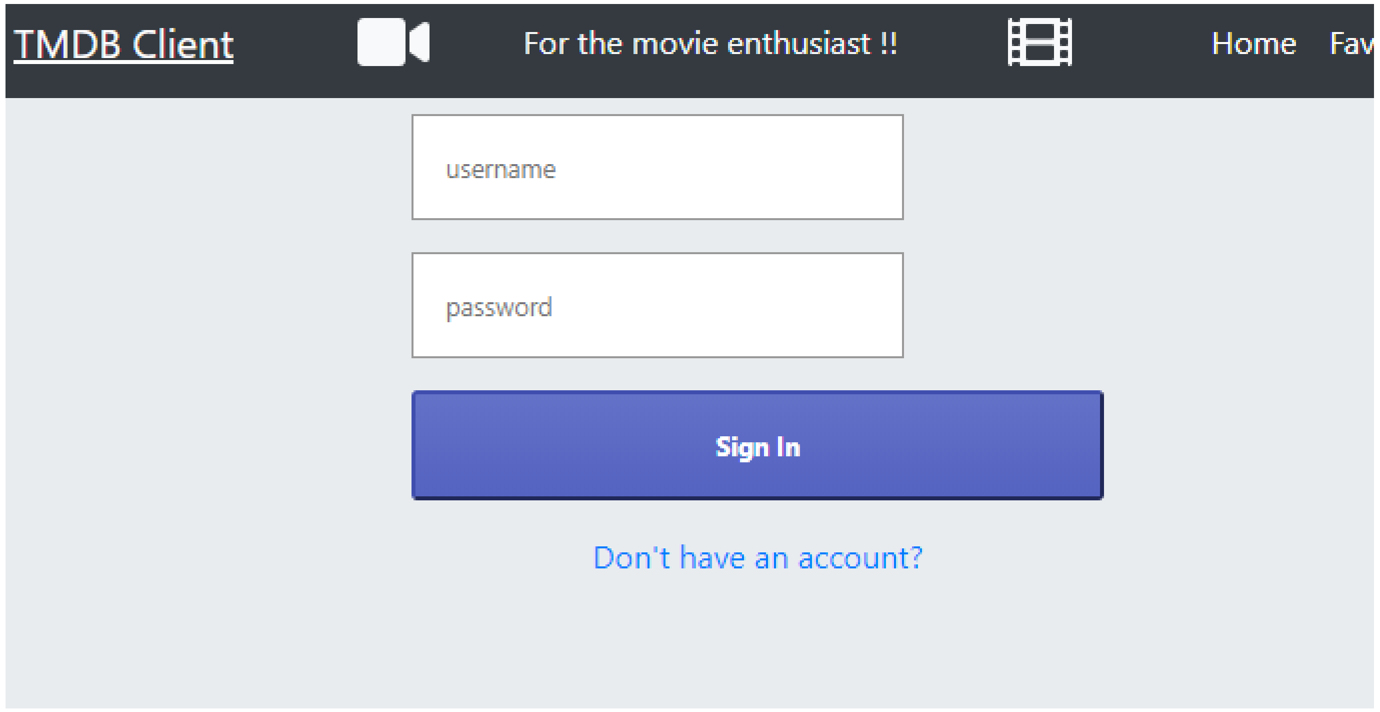
This Photo by Unknown Author is licensed under CC BY-SA

Pagination: Movies API

```javascript
router.get('/', asyncHandler(async (req, res) => {
    let { page = 1, limit = 4 } = req.query; // destructure page and limit and set default values
    [page, limit] = [+page, +limit]; //trick to convert to numeric (req.query will contain string values)

    const totalDocumentsPromise = movieModel.estimatedDocumentCount(); //Kick off async calls
    const moviesPromise = movieModel.find().limit(limit).skip((page - 1) * limit);

    const totalDocuments = await totalDocumentsPromise; //wait for the above promises to be fulfilled
    const movies = await moviesPromise;

    const returnObject = { page: page, total_pages: Math.ceil(totalDocuments / limit), total_results: totalDocuments, results: movies };//construct return Object and insert into response object

    res.status(200).json(returnObject);
}));
```

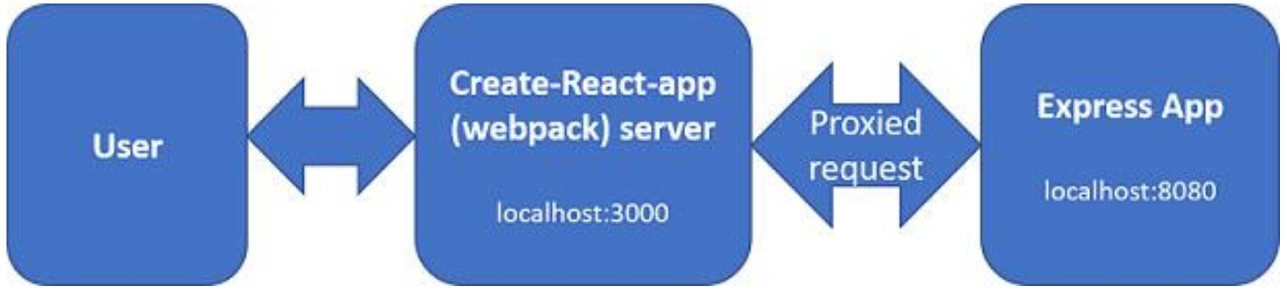
MovieDB React App

- We want to:
  - Replace with calls to Movie API
  - Provide login/signin capabilities.
  - Only allow signed in users to see Movies and add stuff



Proposed Architecture

- Create-React-app uses **Webpack development server**.
- Your Movie API is an Express.js app listening on port 8080 on your local host.
- Configure Webpack server to "proxy" any unknown requests to Express app
  - Just need "**proxy":"http://localhost:8080"** entry in package.json of **the React App.**
- Removes Cross-Origin-Resource-Sharing (CORS) issues with the browser

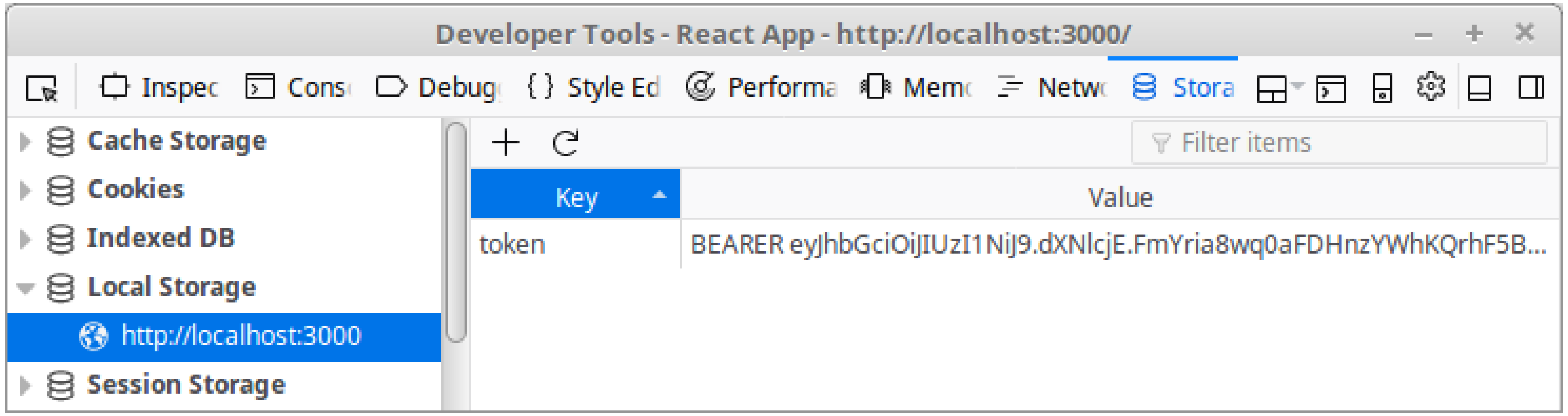


JavaWebToken Storage

- Most browsers/devices have **local storage** .Can access using **localStorage** object.

```
localStorage.setItem('token', token);
```

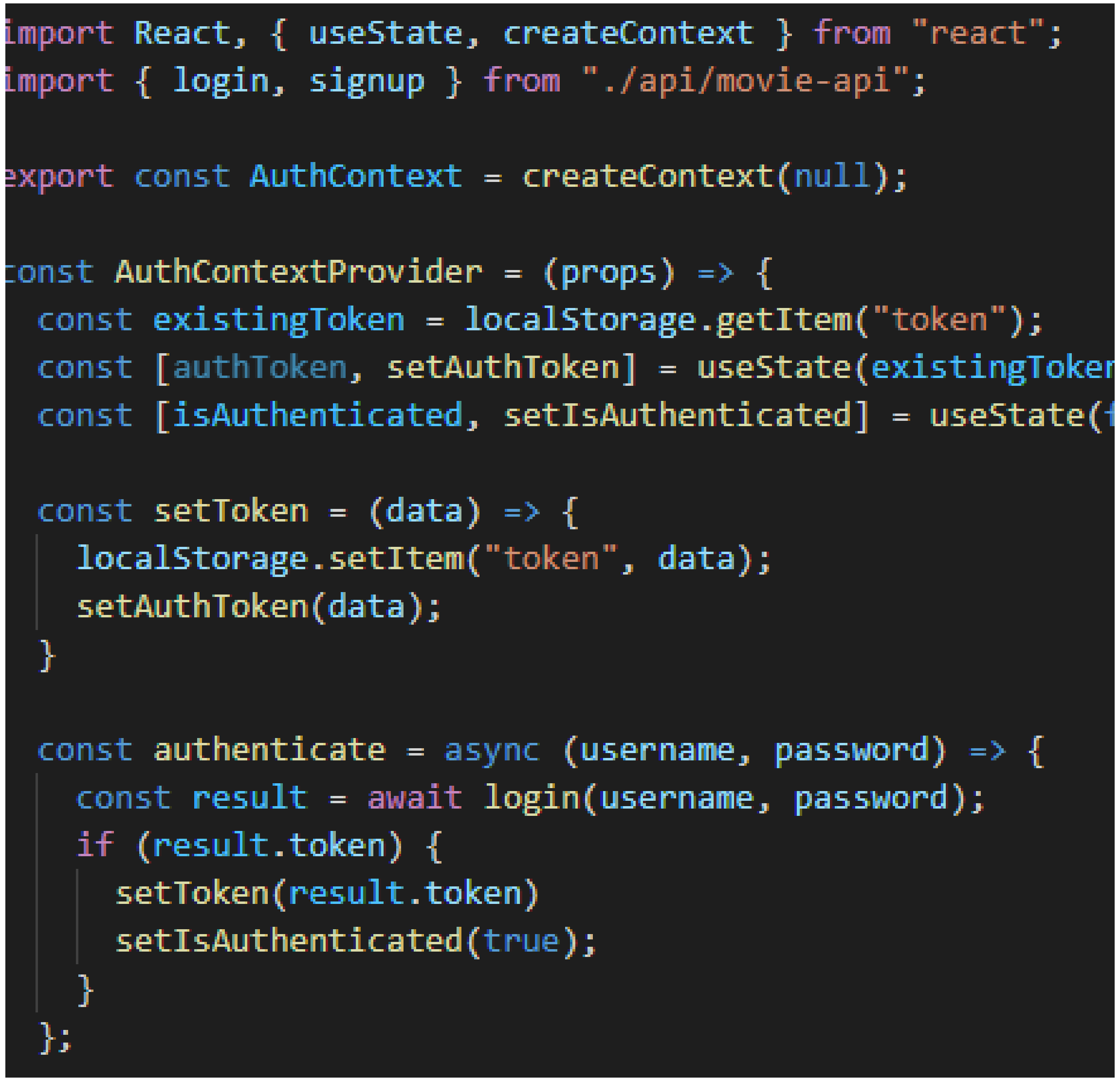
```
 const token =     localStorage.getItem('token');
```



Key	Value
token	BEARER eyJhbGciOiJIUzI1NiJ9.dXNlcjE.FmYria8wq0aFDHnzYWhKQrhF5B...

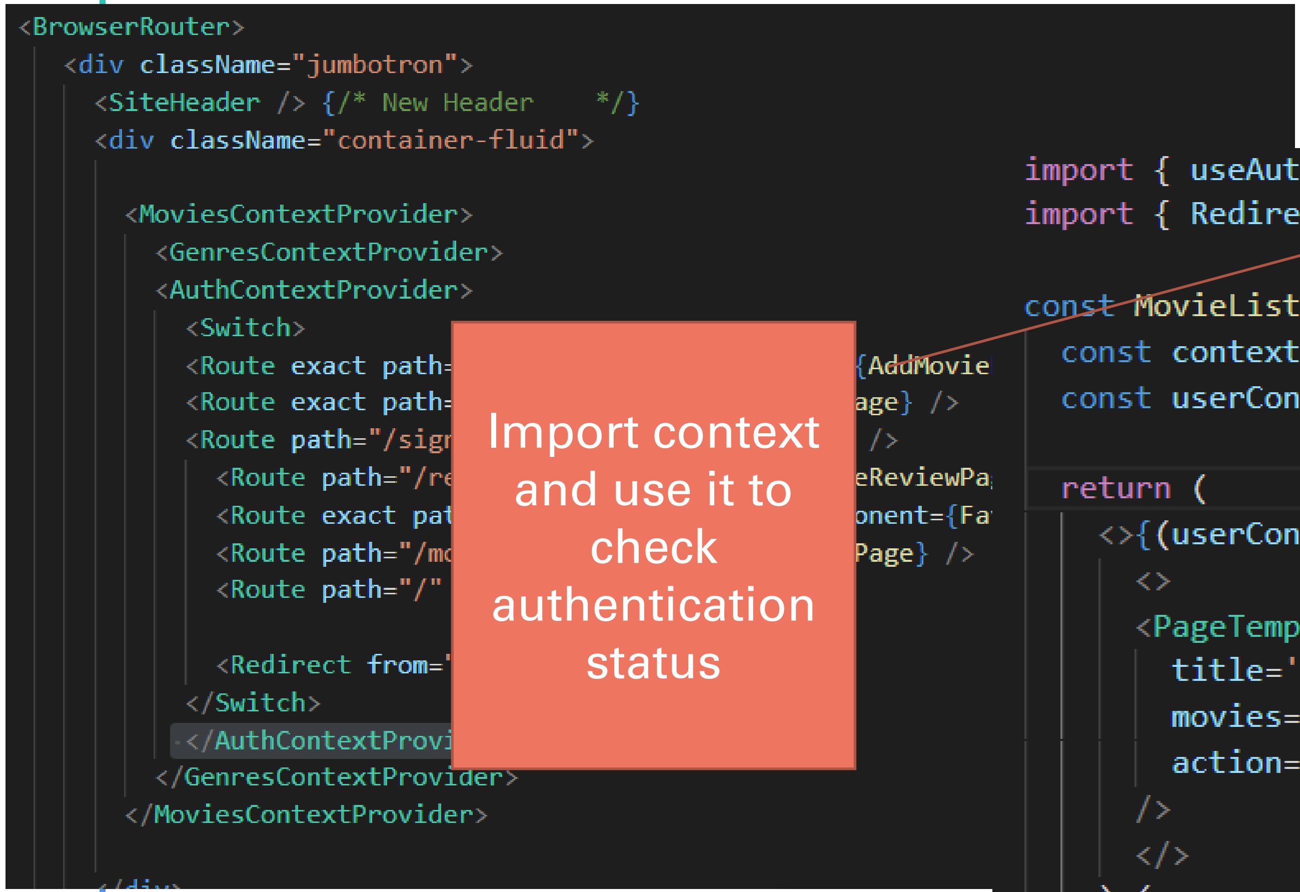
Contexts

- Create a Authentication Context in MovieDB React App.
- As with Movie context, use it to pass data through the component tree
- Share authentication details between components



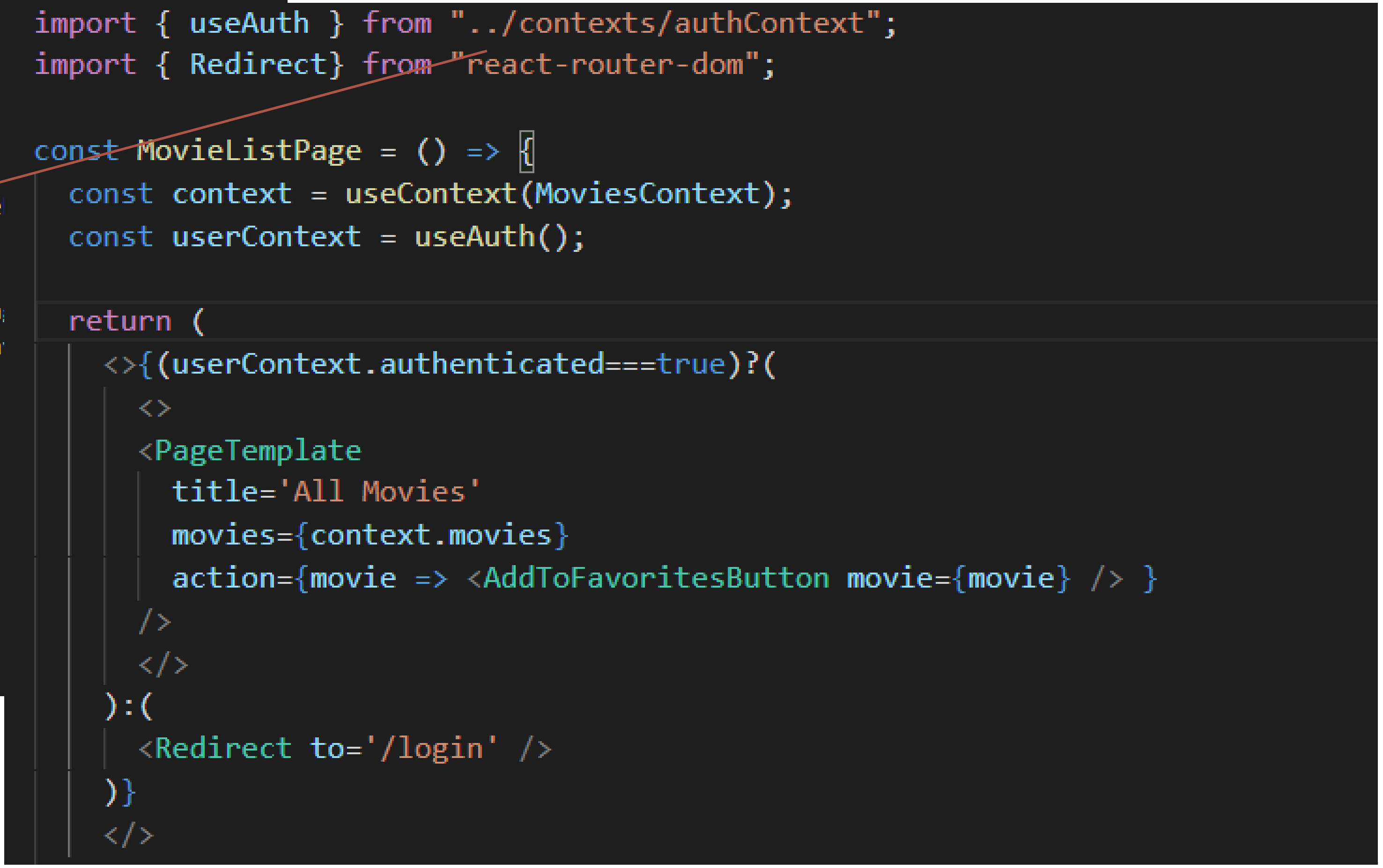
```
import React, { useState, createContext } from "react";
import { login, signup } from "./api/movie-api";

export const AuthContext = createContext(null);

const AuthContextProvider = (props) => {
  const existingToken = localStorage.getItem("token");
  const [authToken, setAuthToken] = useState(existingToken
  const [isAuthenticated, setIsAuthenticated] = useState(

  const setToken = (data) => {
    localStorage.setItem("token", data);
    setAuthToken(data);
  }

  const authenticate = async (username, password) => {
    const result = await login(username, password);
    if (result.token) {
      setToken(result.token)
      setIsAuthenticated(true);
    }
  };
```

Use Context Provider in React App



```
<BrowserRouter>
  <div className="jumbotron">
    <SiteHeader /> {/* New Header    */}
    <div className="container-fluid">

      <MoviesContextProvider>
        <GenresContextProvider>
        <AuthContextProvider>
          <Switch>
          <Route exact path=                      {AddMovie
          <Route exact path=                      age} />
          <Route path="/sign                      />
            <Route path="/re                      eReviewPa
            <Route exact pat                      onent={Fa
            <Route path="/mo                      Page} />
            <Route path="/"

            <Redirect from="
          </Switch>
          </AuthContextProvi
        </GenresContextProvider>
      </MoviesContextProvider>
```

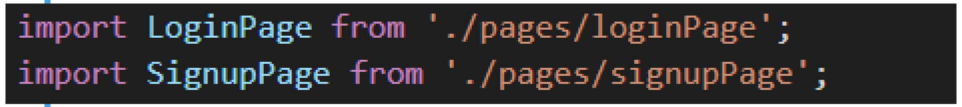
Import context and use it to check authentication status



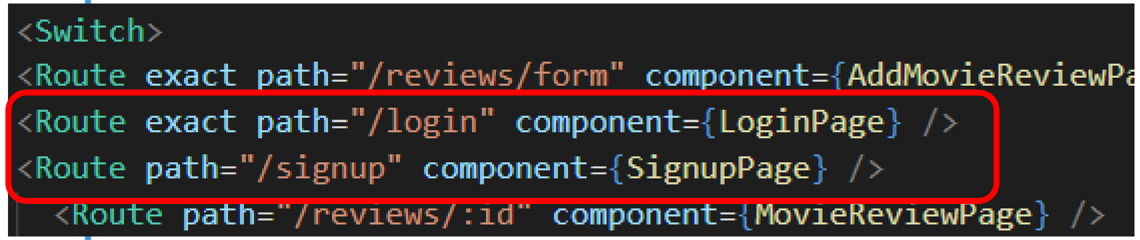
```
import { useAuth } from "../contexts/authContext";
import { Redirect} from "react-router-dom";

const MovieListPage = () => {
  const context = useContext(MoviesContext);
  const userContext = useAuth();

  return (
    <>{(userContext.authenticated===true)?(
      <>
      <PageTemplate
        title='All Movies'
        movies={context.movies}
        action={movie => <AddToFavoritesButton movie={movie} /> }
      />
      </>
    ):(
      <Redirect to='/login' />
    )}
    </>
```

Login/Register Component

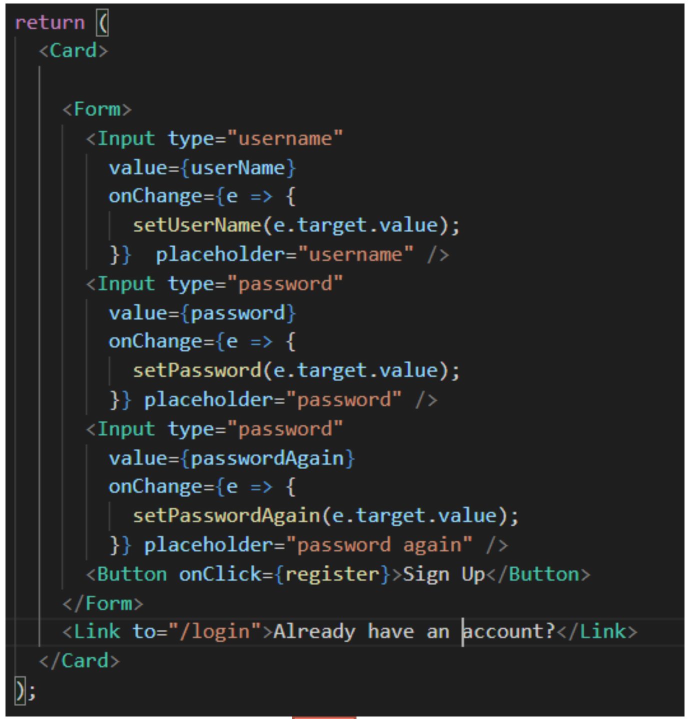
- Add to App router (in index.js)



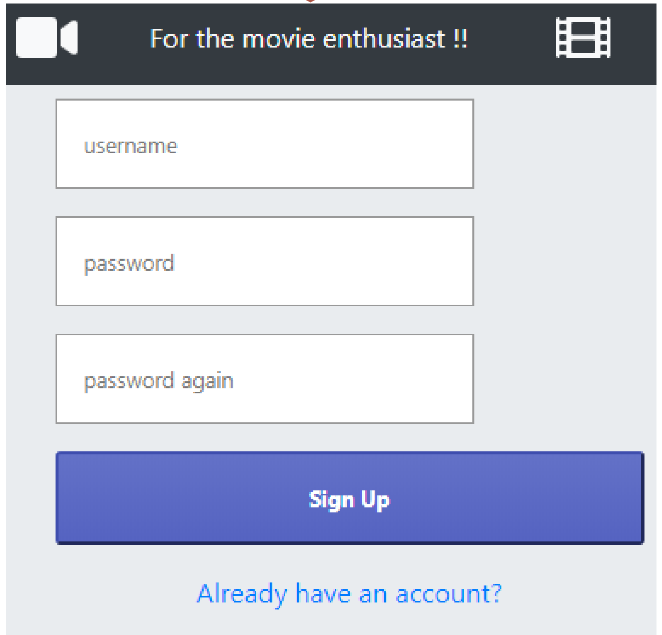
```
import LoginPage from './pages/loginPage';
import SignupPage from './pages/signupPage';
```



```
<Switch>
<Route exact path="/reviews/form" component={AddMovieReviewPa
<Route exact path="/login" component={LoginPage} />
<Route path="/signup" component={SignupPage} />
  <Route path="/reviews/:id" component={MovieReviewPage} />
```



```
return (
  <Card>

    <Form>
      <Input type="username"
        value={userName}
        onChange={e => {
          setUserName(e.target.value);
        }}  placeholder="username" />
      <Input type="password"
        value={password}
        onChange={e => {
          setPassword(e.target.value);
        }} placeholder="password" />
      <Input type="password"
        value={passwordAgain}
        onChange={e => {
          setPasswordAgain(e.target.value);
        }} placeholder="password again" />
      <Button onClick={register}>Sign Up</Button>
    </Form>
    <Link to="/login">Already have an account?</Link>
  </Card>
);
```


For the movie enthusiast !!
username
password
password again
Sign Up
Already have an account?