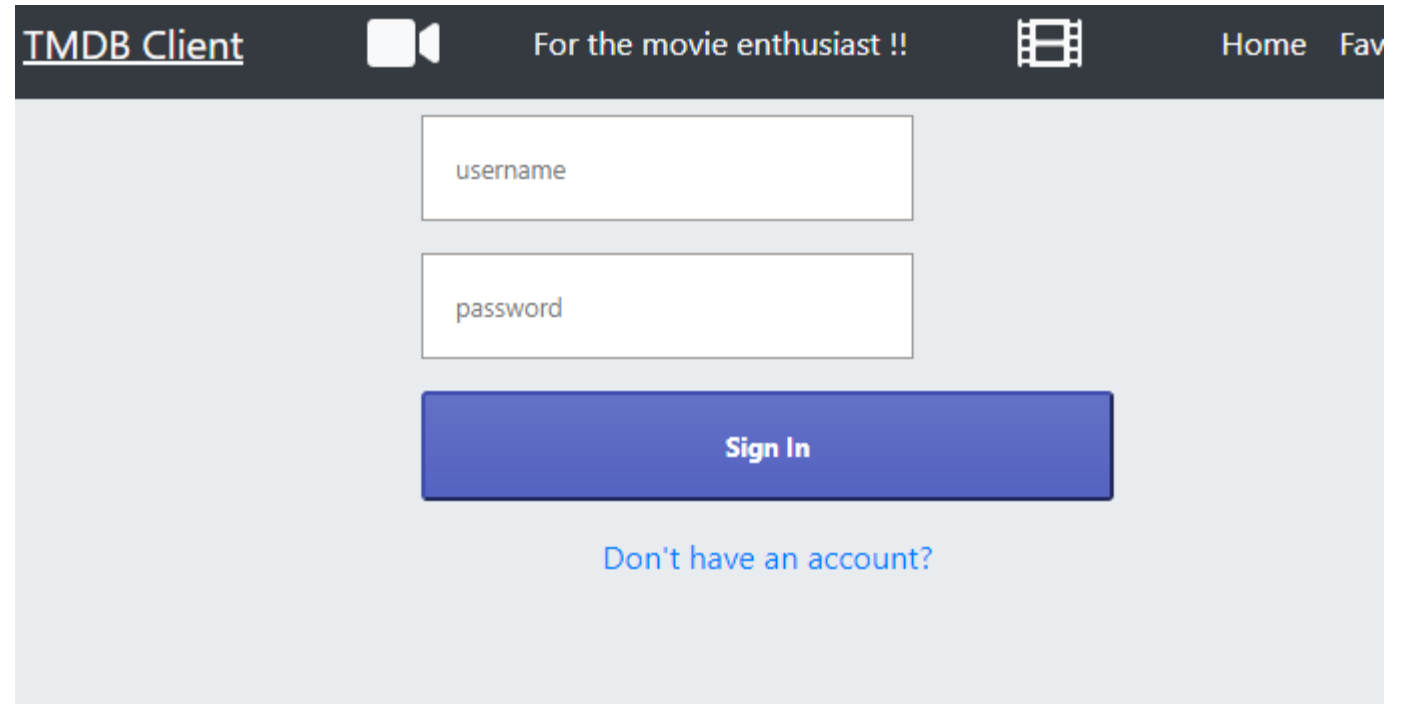




REACT APPS AND JWT

MovieDB 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



The screenshot shows a web application titled "TMDB Client" in the top left corner. To its right is a camera icon, followed by the text "For the movie enthusiast !!", a filmstrip icon, and navigation links "Home" and "Fav". The main content area has a light gray background and contains a login form. The form consists of two white input fields with gray borders: the top one is labeled "username" and the bottom one is labeled "password". Below these fields is a blue rectangular button with the text "Sign In" in white. Underneath the button is a blue hyperlink that reads "Don't have an account?".

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

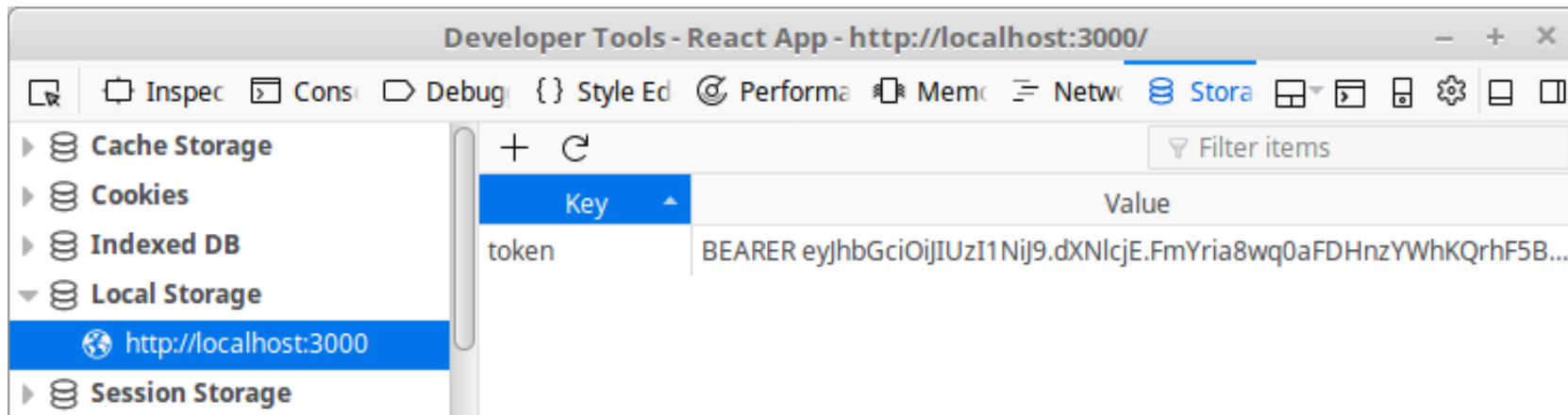


JavaWebToken Storage

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

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

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



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(false);

  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="/" component={AddMoviePage} />
              <Route exact path="/sign-in" component={SignInPage} />
              <Route path="/sign-out" component={SignOutPage} />
              <Route path="/review" component={ReviewPage} />
              <Route exact path="/favorites" component={FavoritesPage} />
              <Route path="/movie/:id" component={MoviePage} />
              <Route path="/" />
            </Switch>
            <Redirect from="/" to="/sign-in" />
          </AuthContextProvider>
        </GenresContextProvider>
      </MoviesContextProvider>
    </div>
  </div>
```

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

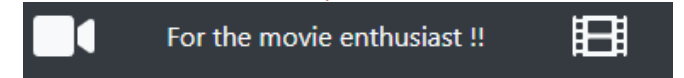
- Add to App router (in index.js)

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

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

```
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>  
)  
;
```





username

password

password again

Sign Up

Already have an account?





INTEGRATING API INTO REACT APP

Agenda

- Review API Design
- Develop front end API Module
 - Endpoint
 - Routes
- React Authentication Context
- React Login/Registration Page

Review API Design

Fully understand the API

- Examine Routes
- Examine Request & Response Requirements
 - Body (e.g. properties)
 - Format (e.g. json)
- Examine Authentication
 - JWT/Sessions
- **TEST WITH POSTMAN**
 - Even better, write unit tests!

default

GET /api/movies List Movies

GET /api/movies/{id} Get a movie by id

GET /api/genres Get genres

GET /api/users Get list of uses (Only for Admins)

POST /api/users Authenticate/Register users

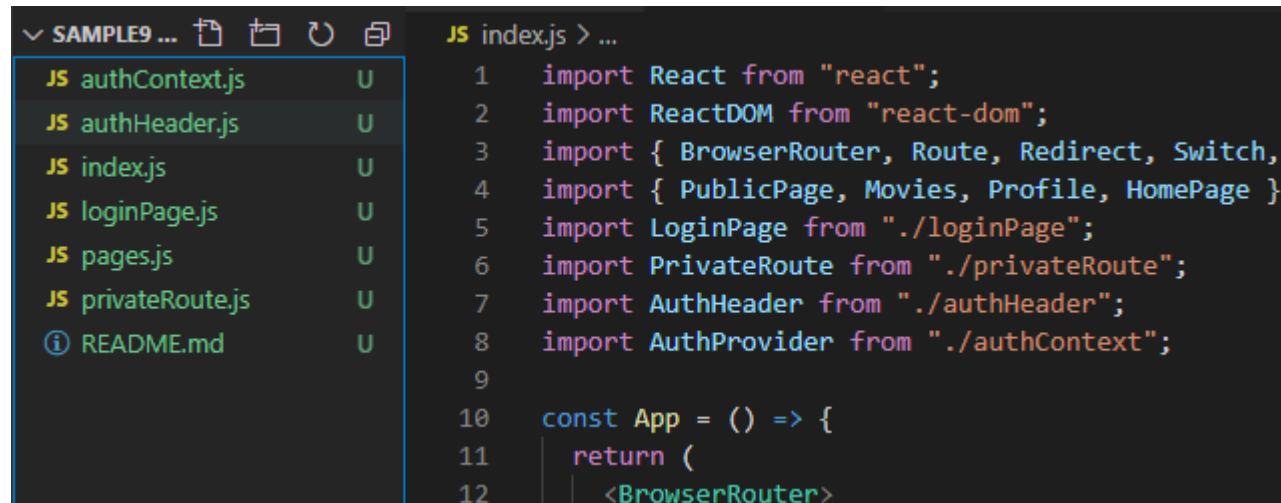
GET /api/users/{userid}/favourites Get User Favourites

POST /api/users/{userid}/favourites Add a favourite

```
[
  {
    "genre_ids": [
      28,
      14,
      878
    ],
    "_id": "5fd73334e4700a102472d5b7",
    "adult": false,
    "backdrop_path": "/jeAQdDX9nguP6YOX6QSWKDPkbBo.jpg",
    "id": 590706,
    "original_language": "en",
    "original_title": "Jiu Jitsu",
    "overview": "Every six years, an ancient order of jiu-jits  
mankind hangs in the balance.",
  }
]
```

Design React App

- This can be your Movies App from assignment 1
- Use stub API/TMDB up to now
- Can update to include authentication(login/register) pages and routes



The screenshot shows a code editor with a dark theme. On the left is a file explorer for a project named 'SAMPLE9 ...'. It lists several files: authContext.js, authHeader.js, index.js, loginPage.js, pages.js, privateRoute.js, and README.md. Each file has a 'U' icon next to it. The 'index.js' file is selected. On the right, the code for 'index.js' is displayed. It starts with imports for React, ReactDOM, and various components from 'react-router-dom' and local files. The code defines a functional component 'App' that returns a 'BrowserRouter'.

```
JS index.js > ...
1  import React from "react";
2  import ReactDOM from "react-dom";
3  import { BrowserRouter, Route, Redirect, Switch,
4  import { PublicPage, Movies, Profile, HomePage }
5  import LoginPage from "../loginPage";
6  import PrivateRoute from "../privateRoute";
7  import AuthHeader from "../authHeader";
8  import AuthProvider from "../authContext";
9
10 const App = () => {
11   return (
12     <BrowserRouter>
```

Write/Update API Module

- Abstract API calls to exported functions.

```
export const getMovies = [()] => {  
  ⚡ return fetch(  
    '/api/movies',{headers: {  
      'Authorization': window.localStorage.getItem('token')  
    }}  
  ).then(res => res.json());  
};  
  
export const getMovie = id => {  
  return fetch(  
    `/api/movies/${id}`, {headers: {  
      'Authorization': window.localStorage.getItem('token')  
    }}  
  ).then(res => res.json());  
};
```

Update pages to use Contexts

- Login Page:

```
return (  
<>  
  <h2>Login page</h2>  
  <p>You must log in to view the protected pages </p>  
  <input id="username" placeholder="user name" onChange={e => {  
    setUsername(e.target.value);  
  }}></input><br />  
  <input id="password" type="password" placeholder="password" onChange={e => {  
    setPassword(e.target.value);  
  }}></input><br />  
  { /* Login web form */}  
  <button onClick={login}>Log in</button>  
  <p>Not Registered?  
  <Link to="/signup">Sign Up!</Link></p>  
</>  
>);
```