Conversation Log 4/20/2023

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Struggling to get Firebase JWT working for authentication. Much of ChatGPT’s advise is pretty questionable on this subject; I had to debug the code when I didn’t know the language… I’m probably would have saved time just using Google and boilerplate code instead.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Human: Well, it seems to correctly report if the username :and password are correct (I can see in the comments), but it always goes to the Greeting screen even if it fails, and none of the data gets to the Greeting screen. What could be the problem? Here are the three files again:

login.php:

<?php

header("Access-Control-Allow-Origin: \*");

header("Access-Control-Allow-Headers: Content-Type");

header("Content-Type: application/json");

// Replace these with your actual database credentials

$servername = "servername";

$username = "username";

$password = "Password";

$dbname = "databasename";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) {

die(json\_encode(["error" => "Connection failed: " . $conn->connect\_error]));

}

// Read JSON data from request body

$data = json\_decode(file\_get\_contents("php://input"), true);

$email = $data["email"];

$password = $data["password"];

$sql = "SELECT \* FROM users WHERE email = ?";

$stmt = $conn->prepare($sql);

$stmt->bind\_param("s", $email);

$stmt->execute();

$result = $stmt->get\_result();

$user = $result->fetch\_assoc();

if ($user) {

if (password\_verify($password, $user["password"])) {

session\_start();

$\_SESSION["user\_id"] = $user["id"];

$\_SESSION["email"] = $user["email"];

echo json\_encode(["success" => true]);

} else {

echo json\_encode(["error" => "Invalid email or password"]);

}

} else {

echo json\_encode(["error" => "Invalid email or password"]);

}

$stmt->close();

$conn->close();

?>

Login.js:

import React from 'react';

import styled from 'styled-components';

import heroImage from '../images/signin-hero.jpg';

import { useNavigate, Link } from 'react-router-dom';

const HomeContainer = styled.div`

align-items: center;

justify-content: center;

height: 100%;

`;

const Footer = styled.footer`

background-color: black;

color: white;

padding: 2rem;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

`;

const FooterItem = styled.p`

margin: 0.2rem;

`;

const Heading = styled.h1`

font-size: 2.5rem;

margin-bottom: 0; // Remove bottom margin

color: white;

width: fit-content;

margin-top:15vh;

background-color: rgba(0, 0, 0, 1); // No transparency

padding: 0.5rem 1rem;

`;

const SubHeading = styled.h2`

font-size: 2rem;

width: fit-content;

color: white;

background-color: rgba(0, 0, 0, 1); // No transparency

padding: 0.5rem 1rem;

`;

const HeroImage = styled.div`

width: 100%;

min-height: 100vh;

background-image: url(${heroImage});

background-position-y: top;

background-size: cover;

display: flex;

justify-content: center;

align-items: center;

display: inline-block;

font-size: 1.5rem;

margin-bottom: 2rem;

`;

const SignInForm = styled.form`

display: flex;

flex-direction: column;

align-items: center;

`;

const Input = styled.input`

font-size: 1rem;

padding: 0.5rem;

margin-bottom: 1rem;

width: 100%;

max-width: 300px;

`;

const LoginButton = styled.button`

font-size: 1rem;

padding: 0.5rem 1rem;

cursor: pointer;

margin-bottom: 1rem;

background-color: rgba(0, 0, 0, 1);

color: white;

transition: background-color 0.3s ease;

&:hover {

background-color: #0072ff;

}

`;

const ForgotPasswordLink = styled.a`

font-size: 0.8rem;

color: white;

text-decoration: underline;

cursor: pointer;

text-shadow: 0px 0px 3px rgba(0, 0, 0, 1),0px 0px 3px rgba(0, 0, 0, 1),0px 0px 3px rgba(0, 0, 0, 1);

&:hover {

color: #e7f1ff;

}

`;

const SignInOrganizer = styled.div`

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding-left: 0;

`;

const Login = () => {

const navigate = useNavigate();

const handleSignInSubmit = async (e) => {

e.preventDefault();

const email = e.target.elements[0].value;

const password = e.target.elements[1].value;

const response = await fetch("https://tomcookson.com/parkezaiphp/login.php", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email,

password,

}),

});

if (response.ok) {

const userData = await response.json();

console.log(userData);

navigate("/greeting", { state: { user: userData } });

} else {

console.log("Error: Invalid login credentials");

}

};

// const handleSignInSubmit = async (e) => {

// e.preventDefault();

// const email = e.target.elements[0].value;

// const password = e.target.elements[1].value;

// const response = await fetch("https://tomcookson.com/parkezaiphp/login.php", {

// method: "POST",

// headers: {

// "Content-Type": "application/json",

// },

// body: JSON.stringify({

// email,

// password,

// }),

// });

// if (response.ok) {

// const userData = await response.json();

// console.log(userData);

// navigate("/greeting", { state: { user: userData } });

// } else {

// // Handle error

// console.log("Error: Invalid login credentials");

// }

// };

return (

<HomeContainer>

<HeroImage>

<SignInOrganizer>

<Heading>Welcome back</Heading>

<SubHeading>Please sign in</SubHeading>

<SignInForm onSubmit={handleSignInSubmit}>

<Input type="email" placeholder="Email" required />

<Input type="password" placeholder="Password" required />

<LoginButton type="submit">Login</LoginButton>

<ForgotPasswordLink>Forgot my password</ForgotPasswordLink>

</SignInForm>

</SignInOrganizer>

</HeroImage>

<Footer>

<FooterItem>ParkEz Inc.</FooterItem>

<FooterItem>1234 Park Street, Suite 567</FooterItem>

<FooterItem>Stamford, CT 06902</FooterItem>

<FooterItem>Phone: (203) 123-4567</FooterItem>

<FooterItem>Email: support@parkez.ai</FooterItem>

</Footer>

</HomeContainer>

);

};

export default Login;

Greeting.js:

// src/components/Greeting.js

import React from "react";

import { useLocation } from "react-router-dom";

import styled from 'styled-components';

const Greeting = () => {

const location = useLocation();

const user = location.state.user;

const WrapperStyle = styled.div`

position: absolute;

color:white;

text-align: center;

`

return (

<div>

<WrapperStyle>

<h1>Welcome, {user.first\_name}!</h1>

<p>Your role: {user.role\_id === 1 ? "Lot Operator" : "Advertiser"}</p>

<p>Your company: {user.company\_name}</p>

<p>Your email: {user.email}</p>

{/\* Add more user-specific information as needed \*/}

</WrapperStyle>

</div>

);

};

export default Greeting;

ChatGPT: It seems there are a couple of issues with the code you provided.Firstly, the login.php script only returns {"success": true} when the login is successful, but it doesn't return the user data. You'll need to modify your PHP script to return the user data as well.Update your login.php file as follows:phpCopy code// ...

if ($user) {

if (password\_verify($password, $user["password"])) {

session\_start();

$\_SESSION["user\_id"] = $user["id"];

$\_SESSION["email"] = $user["email"];

unset($user["password"]); // Remove the password from the returned data

echo json\_encode(["success" => true, "user" => $user]);

} else {

echo json\_encode(["error" => "Invalid email or password"]);

}

} else {

echo json\_encode(["error" => "Invalid email or password"]);

}

// ...

Now, the PHP script will return the user data if the login is successful.Secondly, you should change the condition in your Login.js file to check if the success key in the response JSON is true, not the response's ok property. Update your handleSignInSubmit function as follows:javascriptCopy codeconst handleSignInSubmit = async (e) => {

e.preventDefault();

const email = e.target.elements[0].value;

const password = e.target.elements[1].value;

const response = await fetch("https://tomcookson.com/parkezaiphp/login.php", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email,

password,

}),

});

const data = await response.json();

if (data.success) {

console.log(data.user);

navigate("/greeting", { state: { user: data.user } });

} else {

console.log("Error: Invalid login credentials");

}

};

Now, it should only navigate to the Greeting screen if the login is successful, and the user data should be passed correctly to the Greeting component.

Human: Pasted it, I get this error mesage:

Compiled with problems:

×

ERROR

Cannot read properties of undefined (reading 'first\_name')

TypeError: Cannot read properties of undefined (reading 'first\_name')

at Greeting (http://localhost:3000/parkezai/static/js/bundle.js:1180:38)

at renderWithHooks (http://localhost:3000/parkezai/static/js/bundle.js:29441:22)

at mountIndeterminateComponent (http://localhost:3000/parkezai/static/js/bundle.js:32727:17)

at beginWork (http://localhost:3000/parkezai/static/js/bundle.js:34023:20)

at HTMLUnknownElement.callCallback (http://localhost:3000/parkezai/static/js/bundle.js:19033:18)

at Object.invokeGuardedCallbackDev (http://localhost:3000/parkezai/static/js/bundle.js:19077:20)

at invokeGuardedCallback (http://localhost:3000/parkezai/static/js/bundle.js:19134:35)

at beginWork$1 (http://localhost:3000/parkezai/static/js/bundle.js:39008:11)

at performUnitOfWork (http://localhost:3000/parkezai/static/js/bundle.js:38255:16)

at workLoopSync (http://localhost:3000/parkezai/static/js/bundle.js:38178:9)

ERROR

Cannot read properties of undefined (reading 'first\_name')

TypeError: Cannot read properties of undefined (reading 'first\_name')

at Greeting (http://localhost:3000/parkezai/static/js/bundle.js:1180:38)

at renderWithHooks (http://localhost:3000/parkezai/static/js/bundle.js:29441:22)

at mountIndeterminateComponent (http://localhost:3000/parkezai/static/js/bundle.js:32727:17)

at beginWork (http://localhost:3000/parkezai/static/js/bundle.js:34023:20)

at HTMLUnknownElement.callCallback (http://localhost:3000/parkezai/static/js/bundle.js:19033:18)

at Object.invokeGuardedCallbackDev (http://localhost:3000/parkezai/static/js/bundle.js:19077:20)

at invokeGuardedCallback (http://localhost:3000/parkezai/static/js/bundle.js:19134:35)

at beginWork$1 (http://localhost:3000/parkezai/static/js/bundle.js:39008:11)

at performUnitOfWork (http://localhost:3000/parkezai/static/js/bundle.js:38255:16)

at workLoopSync (http://localhost:3000/parkezai/static/js/bundle.js:38178:9)

ERROR

Cannot read properties of undefined (reading 'first\_name')

TypeError: Cannot read properties of undefined (reading 'first\_name')

at Greeting (http://localhost:3000/parkezai/static/js/bundle.js:1180:38)

at renderWithHooks (http://localhost:3000/parkezai/static/js/bundle.js:29441:22)

at mountIndeterminateComponent (http://localhost:3000/parkezai/static/js/bundle.js:32727:17)

at beginWork (http://localhost:3000/parkezai/static/js/bundle.js:34023:20)

at beginWork$1 (http://localhost:3000/parkezai/static/js/bundle.js:38986:18)

at performUnitOfWork (http://localhost:3000/parkezai/static/js/bundle.js:38255:16)

at workLoopSync (http://localhost:3000/parkezai/static/js/bundle.js:38178:9)

at renderRootSync (http://localhost:3000/parkezai/static/js/bundle.js:38151:11)

at recoverFromConcurrentError (http://localhost:3000/parkezai/static/js/bundle.js:37643:24)

at performConcurrentWorkOnRoot (http://localhost:3000/parkezai/static/js/bundle.js:37555:26)

ChatGPT: The error you are encountering is likely due to the Greeting component trying to access properties of an undefined object (user). It might be caused by the initial rendering of the Greeting component without a user object.To fix this issue, you can add a check for the user object in your Greeting component before trying to access its properties. Update your Greeting.js file as follows:javascriptCopy codeimport React from "react";

import { useLocation } from "react-router-dom";

import styled from 'styled-components';

const Greeting = () => {

const location = useLocation();

const user = location.state?.user;

if (!user) {

return <div>Loading...</div>;

}

const WrapperStyle = styled.div`

position: absolute;

color:white;

text-align: center;

`

return (

<div>

<WrapperStyle>

<h1>Welcome, {user.first\_name}!</h1>

<p>Your role: {user.role\_id === 1 ? "Lot Operator" : "Advertiser"}</p>

<p>Your company: {user.company\_name}</p>

<p>Your email: {user.email}</p>

{/\* Add more user-specific information as needed \*/}

</WrapperStyle>

</div>

);

};

export default Greeting;

This will render a "Loading..." message if the user object is not available, preventing the error. Once the user object is passed to the Greeting component, it will display the user's information as intended.

Human: How can I check if all the necessary data is coming from php in Login.js?

ChatGPT: You can use console.log() to output the received data in the browser's console, which will allow you to inspect the data coming from your PHP script.In your Login.js, you already have a console.log(userData) statement, which logs the response data after a successful login. If you want to see the raw response as well, you can add another console.log() right after receiving the response.Update the handleSignInSubmit function in Login.js like this:javascriptCopy codeconst handleSignInSubmit = async (e) => {

e.preventDefault();

const email = e.target.elements[0].value;

const password = e.target.elements[1].value;

const response = await fetch("https://tomcookson.com/parkezaiphp/login.php", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email,

password,

}),

});

// Log the raw response

console.log("Raw response:", response);

if (response.ok) {

const userData = await response.json();

// Log the JSON data

console.log("JSON data:", userData);

navigate("/greeting", { state: { user: userData } });

} else {

console.log("Error: Invalid login credentials");

}

};

Now, when you perform a login attempt, you'll see the raw response and the parsed JSON data in the browser's console. To view the console, you can use the following keyboard shortcuts:Chrome: Ctrl + Shift + J (Windows) or Cmd + Option + J (Mac)Firefox: Ctrl + Shift + K (Windows) or Cmd + Option + K (Mac)Safari: Cmd + Option + C (Mac)Edge: Ctrl + Shift + J (Windows)Remember to remove or comment out the console.log() statements when you're ready to deploy your application to production, as it's generally not a good practice to expose sensitive information in the console.

Human: Make it so when it login it is recorded in state who is logged in in this session, make it so it goes to greeting.js if it sucess and make it go to a login failed page if login fails. Write the unsucessful login page for me

ChatGPT: To achieve this, you can use the useState hook in the App.js file to store the user's data and conditionally render the components based on the login status. You also need to create a new component for the unsuccessful login page.First, create a new file named LoginFailed.js in your src directory and add the following code to create a simple "Login Failed" page:javascriptCopy codeimport React from "react";

const LoginFailed = () => {

return (

<div>

<h1>Login Failed</h1>

<p>Please check your email and password and try again.</p>

</div>

);

};

export default LoginFailed;

In your App.js file, import useState from 'react', LoginFailed component, and update the file as follows:javascriptCopy codeimport React, { useState } from "react";

import { Router, Link } from "@reach/router";

import Login from "./Login";

import Greeting from "./Greeting";

import LoginFailed from "./LoginFailed";

function App() {

const [user, setUser] = useState(null);

const handleLogin = (userData) => {

setUser(userData);

};

const handleLogout = () => {

setUser(null);

};

return (

<div>

<nav>

<Link to="/">Home</Link>

{user ? (

<button onClick={handleLogout}>Logout</button>

) : (

<Link to="/login">Login</Link>

)}

</nav>

<Router>

<Login path="/login" onLogin={handleLogin} />

{user && <Greeting path="/greeting" user={user} />}

<LoginFailed path="/login-failed" />

</Router>

</div>

);

}

export default App;

Update the Login.js file to use the onLogin prop and navigate to the appropriate route based on the login status:javascriptCopy codeimport React from "react";

import { useNavigate } from "@reach/router";

const Login = ({ onLogin }) => {

const navigate = useNavigate();

const handleSignInSubmit = async (e) => {

e.preventDefault();

const email = e.target.elements[0].value;

const password = e.target.elements[1].value;

const response = await fetch("https://tomcookson.com/parkezaiphp/login.php", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email,

password,

}),

});

if (response.ok) {

const userData = await response.json();

onLogin(userData);

navigate("/greeting");

} else {

navigate("/login-failed");

}

};

return (

<div>

<h1>Login</h1>

<form onSubmit={handleSignInSubmit}>

<input type="email" placeholder="Email" />

<input type="password" placeholder="Password" />

<button type="submit">Sign In</button>

</form>

</div>

);

};

export default Login;

Now, when a user logs in successfully, they will be redirected to the /greeting page, and if the login fails, they will be redirected to the /login-failed page. The user's data will be stored in the state of the App component, and you can use the handleLogout function to log the user out.

Human: Can you add your App.js code above so it works my my current App.js code:

import React from 'react';

import { HashRouter as Router, Route, Routes } from 'react-router-dom';

import Navigation from './components/Navigation';

import Home from './components/Home';

import FindParking from './components/FindParking';

import About from './components/About';

import HelpCenter from './components/HelpCenter';

import Login from './components/Login';

import Signup from './components/Signup';

import styled from 'styled-components';

import UsersList from './components/UsersList';

import Success from './components/Success';

import Error from './components/Error';

import Greeting from "./components/Greeting";

const ContentWrapper = styled.div`

display: flex;

flex-direction: column;

height: calc(100vh - 4rem); // Adjust this value based on the height of the navigation menu

`;

const App = () => {

return (

<Router>

<Navigation />

<ContentWrapper>

<Routes>

<Route path="/" exact element={<Home/>} />

<Route path="/find-parking" element={<FindParking/>} />

<Route path="/about" element={<About/>} />

<Route path="/help-center" element={<HelpCenter/>} />

<Route path="/login" element={<Login/>} />

<Route path="/signup" element={<Signup/>} />

<Route path="/success" element={<Success/>} />

<Route path="/error" element={<Error/>} />

<Route path="/users" element={<UsersList />} />

<Route path="/greeting" element={<Greeting />} />

</Routes>

</ContentWrapper>

</Router>

);

};

export default App;

ChatGPT: To integrate the new functionality with your current App.js code, you can follow these steps:Import the necessary dependencies and components at the top of your App.js file:javascriptCopy codeimport React, { useState } from 'react';

import { Route, Routes, Navigate } from 'react-router-dom';

import LoginFailed from './components/LoginFailed';

Add the user state and the handleLogin and handleLogout functions inside the App component:javascriptCopy codeconst App = () => {

const [user, setUser] = useState(null);

const handleLogin = (userData) => {

setUser(userData);

};

const handleLogout = () => {

setUser(null);

};

// ...

};

Update the Login route to pass the handleLogin function as a prop:javascriptCopy code<Route path="/login" element={<Login onLogin={handleLogin} />} />

Update the Greeting route to pass the user data as a prop and conditionally render the route based on the user state:javascriptCopy code{user && <Route path="/greeting" element={<Greeting user={user} />} />}

Add the LoginFailed route:javascriptCopy code<Route path="/login-failed" element={<LoginFailed />} />

Update your Navigation component to accept the handleLogout function as a prop and pass it from App.js. Also, conditionally render the "Logout" button based on the user state:javascriptCopy code<Navigation user={user} onLogout={handleLogout} />

Your updated App.js file should look like this:javascriptCopy codeimport React, { useState } from 'react';

import { HashRouter as Router, Route, Routes } from 'react-router-dom';

import Navigation from './components/Navigation';

import Home from './components/Home';

import FindParking from './components/FindParking';

import About from './components/About';

import HelpCenter from './components/HelpCenter';

import Login from './components/Login';

import Signup from './components/Signup';

import styled from 'styled-components';

import UsersList from './components/UsersList';

import Success from './components/Success';

import Error from './components/Error';

import Greeting from './components/Greeting';

import LoginFailed from './components/LoginFailed';

const ContentWrapper = styled.div`

display: flex;

flex-direction: column;

height: calc(100vh - 4rem); // Adjust this value based on the height of the navigation menu

`;

const App = () => {

const [user, setUser] = useState(null);

const handleLogin = (userData) => {

setUser(userData);

};

const handleLogout = () => {

setUser(null);

};

return (

<Router>

<Navigation user={user} onLogout={handleLogout} />

<ContentWrapper>

<Routes>

<Route path="/" exact element={<Home />} />

<Route path="/find-parking" element={<FindParking />} />

<Route path="/about" element={<About />} />

<Route path="/help-center" element={<HelpCenter />} />

<Route path="/login" element={<Login onLogin={handleLogin} />} />

<Route path="/signup" element={<Signup />} />

<Route path="/success" element={<Success />} />

<Route path="/error" element={<Error />} />

<Route path="/users" element={<UsersList />} />

{user && <Route path="/greeting" element={<Greeting user={user} />} />}

<Route path="/login-f

Human: Can you just show App.js in the above post again since it got cut off:

ChatGPT: Apologies for the cut-off code. Here's the complete App.js:javascriptCopy codeimport React, { useState } from 'react';

import { HashRouter as Router, Route, Routes } from 'react-router-dom';

import Navigation from './components/Navigation';

import Home from './components/Home';

import FindParking from './components/FindParking';

import About from './components/About';

import HelpCenter from './components/HelpCenter';

import Login from './components/Login';

import Signup from './components/Signup';

import styled from 'styled-components';

import UsersList from './components/UsersList';

import Success from './components/Success';

import Error from './components/Error';

import Greeting from './components/Greeting';

import LoginFailed from './components/LoginFailed';

const ContentWrapper = styled.div`

display: flex;

flex-direction: column;

height: calc(100vh - 4rem); // Adjust this value based on the height of the navigation menu

`;

const App = () => {

const [user, setUser] = useState(null);

const handleLogin = (userData) => {

setUser(userData);

};

const handleLogout = () => {

setUser(null);

};

return (

<Router>

<Navigation user={user} onLogout={handleLogout} />

<ContentWrapper>

<Routes>

<Route path="/" exact element={<Home />} />

<Route path="/find-parking" element={<FindParking />} />

<Route path="/about" element={<About />} />

<Route path="/help-center" element={<HelpCenter />} />

<Route path="/login" element={<Login onLogin={handleLogin} />} />

<Route path="/signup" element={<Signup />} />

<Route path="/success" element={<Success />} />

<Route path="/error" element={<Error />} />

<Route path="/users" element={<UsersList />} />

{user && <Route path="/greeting" element={<Greeting user={user} />} />}

<Route path="/login-failed" element={<LoginFailed />} />

</Routes>

</ContentWrapper>

</Router>

);

};

export default App;

This updated App.js should work with the changes we made to the Login.js and Greeting.js components, as well as the new LoginFailed component.

Human: I get this error message:

ERROR in ./src/components/Login.js 6:0-44

Module not found: Error: Can't resolve '@reach/router' in '/home/tom/ParkEzAI/parkezai/src/components'

With this code:

import React from "react";

import { useNavigate } from "@reach/router";

const Login = ({ onLogin }) => {

const navigate = useNavigate();

const handleSignInSubmit = async (e) => {

e.preventDefault();

const email = e.target.elements[0].value;

const password = e.target.elements[1].value;

const response = await fetch("https://tomcookson.com/parkezaiphp/login.php", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email,

password,

}),

});

if (response.ok) {

const userData = await response.json();

onLogin(userData);

navigate("/greeting");

} else {

navigate("/login-failed");

}

};

return (

<div>

<h1>Login</h1>

<form onSubmit={handleSignInSubmit}>

<input type="email" placeholder="Email" />

<input type="password" placeholder="Password" />

<button type="submit">Sign In</button>

</form>

</div>

);

};

export default Login;

How can it be fixed?

ChatGPT: I apologize for the confusion. You are using React Router v6, and I mistakenly provided you with a solution using @reach/router. Let's update the Login.js file to use the useNavigate hook from react-router-dom instead:javascriptCopy codeimport React from "react";

import { useNavigate } from "react-router-dom";

const Login = ({ onLogin }) => {

const navigate = useNavigate();

const handleSignInSubmit = async (e) => {

e.preventDefault();

const email = e.target.elements[0].value;

const password = e.target.elements[1].value;

const response = await fetch("https://tomcookson.com/parkezaiphp/login.php", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email,

password,

}),

});

if (response.ok) {

const userData = await response.json();

onLogin(userData);

navigate("/greeting");

} else {

navigate("/login-failed");

}

};

return (

<div>

<h1>Login</h1>

<form onSubmit={handleSignInSubmit}>

<input type="email" placeholder="Email" />

<input type="password" placeholder="Password" />

<button type="submit">Sign In</button>

</form>

</div>

);

};

export default Login;

This updated code imports useNavigate from react-router-dom and should work with your current setup.

Human: Can you re-wrtie greeting.js to work with this code?

It currently looks like this:

import React from "react";

import { useLocation } from "react-router-dom";

import styled from 'styled-components';

const Greeting = () => {

const location = useLocation();

const user = location.state?.user;

if (!user) {

return <div>Loading...</div>;

}

const WrapperStyle = styled.div`

position: absolute;

color:white;

text-align: center;

`

return (

<div>

<WrapperStyle>

<h1>Welcome, {user.first\_name}!</h1>

<p>Your role: {user.role\_id === 1 ? "Lot Operator" : "Advertiser"}</p>

<p>Your company: {user.company\_name}</p>

<p>Your email: {user.email}</p>

{/\* Add more user-specific information as needed \*/}

</WrapperStyle>

</div>

);

};

export default Greeting;

ChatGPT: In order to make the Greeting.js work with the updated Login.js code, we need to change how it receives the user data. We will pass the user data as a prop instead of using the useLocation hook. Here's the updated Greeting.js code:javascriptCopy codeimport React from "react";

import styled from 'styled-components';

const Greeting = ({ user }) => {

if (!user) {

return <div>Loading...</div>;

}

const WrapperStyle = styled.div`

position: absolute;

color:white;

text-align: center;

`

return (

<WrapperStyle>

<h1>Welcome, {user.first\_name}!</h1>

<p>Your role: {user.role\_id === 1 ? "Lot Operator" : "Advertiser"}</p>

<p>Your company: {user.company\_name}</p>

<p>Your email: {user.email}</p>

{/\* Add more user-specific information as needed \*/}

</WrapperStyle>

);

};

export default Greeting;

Now, when you render the Greeting component in the App.js, you need to pass the user prop. I assume you have added a state for the logged-in user in App.js. Let's say you have added a state named loggedInUser, then you can pass it as a prop like this:javascriptCopy code<Route path="/greeting" element={<Greeting user={loggedInUser} />} />

Make sure to update the state name if you used a different name for the logged-in user state in App.js.

Human: There is a prop user going to greeting... can you make it so i can reach that prop:

import React from "react";

import { useActionData, useLocation } from "react-router-dom";

import styled from 'styled-components';

const Greeting = () => {

const location = useLocation();

const user = location.state?.user;

console.log(user)

if (!user) {

return <div>Loading...</div>;

}

const WrapperStyle = styled.div`

position: absolute;

color:white;

text-align: center;

barckground-color:white;

`

return (

<div>

<WrapperStyle>

<h1>Welcome, {user.first\_name}!</h1>

<p>Your role: {user.role\_id === 1 ? "Lot Operator" : "Advertiser"}</p>

<p>Your company: {user.company\_name}</p>

<p>Your email: {user.email}</p>

</WrapperStyle>

</div>

);

};

export default Greeting;

ChatGPT: You have a small typo in your code. Instead of useActionData, it should be useLocation. Here's the corrected version:javascriptCopy codeimport React from "react";

import { useLocation } from "react-router-dom";

import styled from 'styled-components';

const Greeting = () => {

const location = useLocation();

const user = location.state?.user;

console.log(user)

if (!user) {

return <div>Loading...</div>;

}

const WrapperStyle = styled.div`

position: absolute;

color:white;

text-align: center;

background-color:white;

`

return (

<div>

<WrapperStyle>

<h1>Welcome, {user.first\_name}!</h1>

<p>Your role: {user.role\_id === 1 ? "Lot Operator" : "Advertiser"}</p>

<p>Your company: {user.company\_name}</p>

<p>Your email: {user.email}</p>

</WrapperStyle>

</div>

);

};

export default Greeting;

With this code, you should be able to access the user prop passed from the parent component. The console.log(user) line will print the user object in the browser console, and you should see the user data displayed in the Greeting component.

Human: I want to use JWT session management with my project. What must I change

ChatGPT: To use JWT (JSON Web Tokens) for session management, you'll need to make the following changes to your project:Update the PHP login script to create and return a JWT when the user logs in.Update the Login component in React to store the JWT in localStorage.Create a utility function in your React app to check the JWT's validity.Protect your routes using the utility function to check JWT's validity.Update the PHP API endpoints to accept and validate the JWT in the request headers.Here's a high-level overview of how to make these changes:Update the PHP login script:You'll need to use a library like firebase/php-jwt to create and handle JWTs in your PHP scripts. After validating the user's credentials, generate a JWT and return it in the response. The JWT should contain the user's ID, role, and other relevant information.Update the Login component in React:When the login is successful, store the received JWT in the browser's localStorage. You can do this in the handleSignInSubmit function of the Login component:javascriptCopy codeif (response.ok) {

const { token } = await response.json();

localStorage.setItem("jwt", token);

onLogin(token);

navigate("/greeting");

}

Create a utility function to check the JWT's validity:Create a utility function to decode and validate the JWT. You can use the jsonwebtoken library to handle JWTs in JavaScript. Install the library using npm install jsonwebtoken.Then, create a utility function like this:javascriptCopy codeimport jwtDecode from "jsonwebtoken/decode";

export const getUserFromToken = () => {

const token = localStorage.getItem("jwt");

if (!token) return null;

try {

const user = jwtDecode(token);

return user;

} catch (error) {

return null;

}

};

Protect your routes:Update your routes to require authentication. You can create a wrapper component that checks if the user is authenticated and redirects them to the login page if they're not.Update PHP API endpoints:In your PHP scripts, check for the presence of the JWT in the request headers. Use the JWT library you added earlier to decode and validate the JWT. If the token is valid, proceed with processing the request. Otherwise, return an error.That's a high-level overview of how to implement JWT-based session management in your project. Depending on your project's specific requirements, you may need to make additional changes or adjustments.

Human: Okay, I'm using PHP 8.1... can I import JWT libaries to that for my project?

ChatGPT: Yes, you can use JWT libraries with PHP 8.1 for your project. One popular library for handling JWT in PHP is firebase/php-jwt. This library is compatible with PHP 8.1.To get started, you'll need to install the library using Composer, which is a dependency manager for PHP.First, make sure you have Composer installed. If you don't have it installed, you can follow the instructions on the official Composer website to set it up.Once Composer is installed, navigate to your project's root directory and run the following command:bashCopy codecomposer require firebase/php-jwt

This will add the firebase/php-jwt library to your project.After the library is installed, you can use it in your PHP scripts to create, decode, and verify JWTs. Here's an example of how to create a JWT:phpCopy code<?php

require 'vendor/autoload.php';

use Firebase\JWT\JWT;

$key = "your\_secret\_key"; // Replace this with your own secret key

$issuedAt = time();

$expirationTime = $issuedAt + 3600; // Token will expire in 1 hour

$payload = array(

"iss" => "http://yourdomain.com",

"iat" => $issuedAt,

"exp" => $expirationTime,

"userId" => "user\_id", // Replace this with the actual user ID

"userRole" => "user\_role" // Replace this with the actual user role

);

$jwt = JWT::encode($payload, $key);

echo "JWT: " . $jwt . PHP\_EOL;

?>

And here's an example of how to decode and verify a JWT:phpCopy code<?php

require 'vendor/autoload.php';

use Firebase\JWT\JWT;

$key = "your\_secret\_key"; // Replace this with your own secret key

$jwt = "your\_jwt"; // Replace this with the actual JWT received from the client

try {

$decoded = JWT::decode($jwt, $key, array('HS256'));

print\_r($decoded);

} catch (\Firebase\JWT\ExpiredException $e) {

echo "Error: Token expired." . PHP\_EOL;

} catch (\Firebase\JWT\BeforeValidException $e) {

echo "Error: Before valid exception." . PHP\_EOL;

} catch (\Firebase\JWT\SignatureInvalidException $e) {

echo "Error: Invalid signature." . PHP\_EOL;

} catch (\Exception $e) {

echo "Error: " . $e->getMessage() . PHP\_EOL;

}

?>

Remember to replace the placeholders with actual values in your project.

Human: I think I installed jwt correctly, whats a very simple way I can test it with my project?

ChatGPT: To test the JWT functionality with your project, let's create two simple PHP scripts: one to generate the JWT token and another to decode and verify the token.Create a file called generate\_jwt.php in your project directory and add the following code:phpCopy code<?php

require 'vendor/autoload.php';

use Firebase\JWT\JWT;

$key = "your\_secret\_key"; // Replace this with your own secret key

$payload = array(

"userId" => "user\_id", // Replace this with the actual user ID

"userRole" => "user\_role" // Replace this with the actual user role

);

$jwt = JWT::encode($payload, $key);

echo "JWT: " . $jwt;

?>

Create another file called decode\_jwt.php in your project directory and add the following code:phpCopy code<?php

require 'vendor/autoload.php';

use Firebase\JWT\JWT;

$key = "your\_secret\_key"; // Replace this with your own secret key

$jwt = isset($\_GET['jwt']) ? $\_GET['jwt'] : ''; // Get the JWT from the URL query string

if (!$jwt) {

echo "Error: No JWT provided.";

exit;

}

try {

$decoded = JWT::decode($jwt, $key, array('HS256'));

print\_r($decoded);

} catch (\Firebase\JWT\ExpiredException $e) {

echo "Error: Token expired.";

} catch (\Firebase\JWT\BeforeValidException $e) {

echo "Error: Before valid exception.";

} catch (\Firebase\JWT\SignatureInvalidException $e) {

echo "Error: Invalid signature.";

} catch (\Exception $e) {

echo "Error: " . $e->getMessage();

}

?>

Now, you can test the JWT functionality:Open a terminal, navigate to your project directory, and start a local PHP server:shCopy codephp -S localhost:8000

In your web browser, navigate to http://localhost:8000/generate\_jwt.php. This will display a JWT token. Copy the token.In your web browser, navigate to http://localhost:8000/decode\_jwt.php?jwt=YOUR\_JWT\_TOKEN\_HERE, replacing YOUR\_JWT\_TOKEN\_HERE with the token you copied in the previous step. This should display the decoded payload information.If you see the decoded payload information, your JWT installation and implementation are working correctly.

Human: The firebase/jwt library know works on my server -- I was able to test it with your code but I had to change a few things since its version 6.4.

ChatGPT: I'm glad to hear that the firebase/jwt library is working on your server. I apologize for any confusion with the code provided earlier. Since you're using version 6.4 of the library, some adjustments might be necessary.If you need any further assistance or have questions about implementing JWT in your specific project, please feel free to ask, and I'll be happy to help.

Human: Can you please make a Login.js page and Login.php work with firebase/jwt. I want to keep the token and use it to keep a session going for someone using the website. You will probably have to rewrite Login.php entirely and much of Login.js. This is what they look like now:

Login.js

import React from 'react';

import { useNavigate } from "react-router-dom";

import styled from 'styled-components';

import heroImage from '../images/signin-hero.jpg';

const HomeContainer = styled.div`

align-items: center;

justify-content: center;

height: 100%;

`;

const Footer = styled.footer`

background-color: black;

color: white;

padding: 2rem;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

`;

const FooterItem = styled.p`

margin: 0.2rem;

`;

const Heading = styled.h1`

font-size: 2.5rem;

margin-bottom: 0; // Remove bottom margin

color: white;

width: fit-content;

margin-top:15vh;

background-color: rgba(0, 0, 0, 1); // No transparency

padding: 0.5rem 1rem;

`;

const SubHeading = styled.h2`

font-size: 2rem;

width: fit-content;

color: white;

background-color: rgba(0, 0, 0, 1); // No transparency

padding: 0.5rem 1rem;

`;

const HeroImage = styled.div`

width: 100%;

min-height: 100vh;

background-image: url(${heroImage});

background-position-y: top;

background-size: cover;

display: flex;

justify-content: center;

align-items: center;

display: inline-block;

font-size: 1.5rem;

margin-bottom: 2rem;

`;

const SignInForm = styled.form`

display: flex;

flex-direction: column;

align-items: center;

`;

const Input = styled.input`

font-size: 1rem;

padding: 0.5rem;

margin-bottom: 1rem;

width: 100%;

max-width: 300px;

`;

const LoginButton = styled.button`

font-size: 1rem;

padding: 0.5rem 1rem;

cursor: pointer;

margin-bottom: 1rem;

background-color: rgba(0, 0, 0, 1);

color: white;

transition: background-color 0.3s ease;

&:hover {

background-color: #0072ff;

}

`;

const ForgotPasswordLink = styled.a`

font-size: 0.8rem;

color: white;

text-decoration: underline;

cursor: pointer;

text-shadow: 0px 0px 3px rgba(0, 0, 0, 1),0px 0px 3px rgba(0, 0, 0, 1),0px 0px 3px rgba(0, 0, 0, 1);

&:hover {

color: #e7f1ff;

}

`;

const SignInOrganizer = styled.div`

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding-left: 0;

`;

const Login = ({ onLogin }) => {

const navigate = useNavigate();

const handleSignInSubmit = async (e) => {

e.preventDefault();

const email = e.target.elements[0].value;

const password = e.target.elements[1].value;

const response = await fetch("https://tomcookson.com/parkezaiphp/login.php", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email,

password,

}),

});

if (response.ok) {

const userData = await response.json();

onLogin(userData);

navigate("/greeting");

} else {

navigate("/login-failed");

}

};

return (

<HomeContainer>

<HeroImage>

<SignInOrganizer>

<Heading>Welcome back</Heading>

<SubHeading>Please sign in</SubHeading>

<SignInForm onSubmit={handleSignInSubmit}>

<Input type="email" placeholder="Email" required />

<Input type="password" placeholder="Password" required />

<LoginButton type="submit">Login</LoginButton>

<ForgotPasswordLink>Forgot my password</ForgotPasswordLink>

</SignInForm>

</SignInOrganizer>

</HeroImage>

<Footer>

<FooterItem>ParkEz Inc.</FooterItem>

<FooterItem>1234 Park Street, Suite 567</FooterItem>

<FooterItem>Stamford, CT 06902</FooterItem>

<FooterItem>Phone: (203) 123-4567</FooterItem>

<FooterItem>Email: support@parkez.ai</FooterItem>

</Footer>

</HomeContainer>

);

};

export default Login;

Login.php

<?php

header("Access-Control-Allow-Origin: \*");

header("Access-Control-Allow-Headers: Content-Type");

header("Content-Type: application/json");

// Replace these with your actual database credentials

$servername = "servername";

$username = "username";

$password = "Password";

$dbname = "databasename";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) {

die(json\_encode(["error" => "Connection failed: " . $conn->connect\_error]));

}

// Read JSON data from request body

$data = json\_decode(file\_get\_contents("php://input"), true);

$email = $data["email"];

$password = $data["password"];

$sql = "SELECT \* FROM users WHERE email = ?";

$stmt = $conn->prepare($sql);

$stmt->bind\_param("s", $email);

$stmt->execute();

$result = $stmt->get\_result();

$user = $result->fetch\_assoc();

if ($user) {

if (password\_verify($password, $user["password"])) {

session\_start();

$\_SESSION["user\_id"] = $user["id"];

$\_SESSION["email"] = $user["email"];

unset($user["password"]); // Remove the password from the returned data

echo json\_encode(["success" => true, "user" => $user]);

} else {

echo json\_encode(["error" => "Invalid email or password"]);

}

} else {

echo json\_encode(["error" => "Invalid email or password"]);

}

$stmt->close();

$conn->close();

?>

ChatGPT: To implement JWT-based session management, you'll need to make changes to both Login.js and Login.php files. First, update your Login.php file to generate a JWT token upon successful login. Then, modify the Login.js file to store the JWT token in localStorage and include it in the Authorization header for future requests.Here's how to update your code:Update Login.php:Install the firebase/php-jwt library if you haven't done it already:bashCopy codecomposer require firebase/php-jwt

Now, modify the Login.php code:phpCopy code<?php

require \_\_DIR\_\_ . '/vendor/autoload.php'; // Make sure to adjust the path to your vendor/autoload.php

use Firebase\JWT\JWT;

// ... (Keep the existing code for database connection and headers)

if ($user) {

if (password\_verify($password, $user["password"])) {

// Generate JWT token

$key = "your\_secret\_key"; // Replace this with your own secret key

$payload = [

"iss" => "https://yourwebsite.com", // Replace this with your own website URL

"aud" => "https://yourwebsite.com", // Replace this with your own website URL

"iat" => time(),

"nbf" => time(),

"exp" => time() + (60 \* 60), // Token expires in 1 hour

"data" => [

"user\_id" => $user["id"],

"email" => $user["email"]

]

];

$jwt = JWT::encode($payload, $key);

unset($user["password"]); // Remove the password from the returned data

echo json\_encode(["success" => true, "user" => $user, "token" => $jwt]);

} else {

echo json\_encode(["error" => "Invalid email or password"]);

}

} else {

echo json\_encode(["error" => "Invalid email or password"]);

}

$stmt->close();

$conn->close();

?>

Update Login.js:javascriptCopy codeimport React, { useState } from "react";

// ... (Keep the rest of the import statements)

const Login = ({ onLogin }) => {

const navigate = useNavigate();

const [errorMessage, setErrorMessage] = useState("");

const handleSignInSubmit = async (e) => {

e.preventDefault();

const email = e.target.elements[0].value;

const password = e.target.elements[1].value;

const response = await fetch("https://tomcookson.com/parkezaiphp/login.php", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email,

password,

}),

});

const data = await response.json();

if (response.ok) {

localStorage.setItem("token", data.token);

onLogin(data.user);

navigate("/greeting");

} else {

setErrorMessage(data.error || "Login failed");

navigate("/login-failed");

}

};

// ... (Keep the rest of the component unchanged)

};

export default Login;

These changes will generate a JWT token on successful login, send it to the client, and store it in localStorage. Remember to include the JWT token in the Authorization header for any future API requests that require authentication.

Human: Can you explain these lines to me:

"iss" => "https://yourwebsite.com", // Replace this with your own website URL

"aud" => "https://yourwebsite.com", // Replace this with your own website URL

ChatGPT: Certainly! These lines are part of the JWT payload and are standard JWT claims that help provide additional information and validation for the token."iss" (Issuer): This claim identifies the principal that issued the JWT. In this case, it's your website URL. This information is used by the recipient of the JWT to validate that the token was indeed issued by a trusted source (your website). When you validate a JWT token, you would typically check if the "iss" claim matches your expected issuer."aud" (Audience): This claim identifies the recipients that the JWT is intended for. In this case, it's also set to your website URL. This helps ensure that the token is only used by the intended audience. When validating the token, the recipient checks if their identifier is included in the "aud" claim.For both claims, you should replace "https://yourwebsite.com" with your actual website URL. This helps ensure that the JWT token is used only within your application and by the intended audience.

Human: If the project is hosted on "tomcookson.com/parkezai" and the php file is hosted on "tomcookson.com/php2/" then which of those URLs should I make iss and aud?

ChatGPT: In this case, you can set both "iss" and "aud" to the base URL of your website, which is "https://tomcookson.com". This will help ensure that the JWT token is used only within your application and by the intended audience.Here's how it should look:phpCopy code$payload = [

"iat" => $issuedAt,

"nbf" => $notBefore,

"exp" => $expiration,

"iss" => "https://tomcookson.com",

"aud" => "https://tomcookson.com",

"data" => [

"user\_id" => $user["id"],

"email" => $user["email"],

"first\_name" => $user["first\_name"],

"last\_name" => $user["last\_name"],

"role\_id" => $user["role\_id"],

"company\_name" => $user["company\_name"]

]

];

The "iss" and "aud" claims don't have to be the exact URL of the script, but rather the base URL of the website that is responsible for issuing and consuming the JWT tokens.

Human: Thanks... can you give me the whole login.php with all your updated code

ChatGPT: Here's the updated login.php code with the JWT implementation:phpCopy code<?php

require 'vendor/autoload.php';

use Firebase\JWT\JWT;

header("Access-Control-Allow-Origin: \*");

header("Access-Control-Allow-Headers: Content-Type");

header("Content-Type: application/json");

// Replace these with your actual database credentials

$servername = "servername";

$username = "username";

$password = "Password";

$dbname = "databasename";

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) {

die(json\_encode(["error" => "Connection failed: " . $conn->connect\_error]));

}

// Read JSON data from request body

$data = json\_decode(file\_get\_contents("php://input"), true);

$email = $data["email"];

$password = $data["password"];

$sql = "SELECT \* FROM users WHERE email = ?";

$stmt = $conn->prepare($sql);

$stmt->bind\_param("s", $email);

$stmt->execute();

$result = $stmt->get\_result();

$user = $result->fetch\_assoc();

if ($user) {

if (password\_verify($password, $user["password"])) {

$secretKey = "your-secret-key"; // Replace this with your own secret key

$issuedAt = time();

$notBefore = $issuedAt;

$expiration = $issuedAt + 3600; // Token valid for 1 hour

$payload = [

"iat" => $issuedAt,

"nbf" => $notBefore,

"exp" => $expiration,

"iss" => "https://tomcookson.com",

"aud" => "https://tomcookson.com",

"data" => [

"user\_id" => $user["id"],

"email" => $user["email"],

"first\_name" => $user["first\_name"],

"last\_name" => $user["last\_name"],

"role\_id" => $user["role\_id"],

"company\_name" => $user["company\_name"]

]

];

$jwt = JWT::encode($payload, $secretKey, "HS256");

echo json\_encode(["success" => true, "token" => $jwt]);

} else {

echo json\_encode(["error" => "Invalid email or password"]);

}

} else {

echo json\_encode(["error" => "Invalid email or password"]);

}

$stmt->close();

$conn->close();

?>

Make sure to replace "your-secret-key" with your own secret key. This updated code generates a JWT token upon successful authentication and includes the user data in the token payload. The token is then sent to the client.

Human: Thanks, can you also give me the whole login.js?

ChatGPT: Here's the updated Login.js code with JWT implementation:javascriptCopy codeimport React from 'react';

import { useNavigate } from "react-router-dom";

import styled from 'styled-components';

import heroImage from '../images/signin-hero.jpg';

// ... (all styled components definitions)

const Login = ({ onLogin }) => {

const navigate = useNavigate();

const handleSignInSubmit = async (e) => {

e.preventDefault();

const email = e.target.elements[0].value;

const password = e.target.elements[1].value;

const response = await fetch("https://tomcookson.com/php2/login.php", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email,

password,

}),

});

if (response.ok) {

const { success, token } = await response.json();

if (success) {

localStorage.setItem("token", token);

onLogin(token);

navigate("/greeting");

} else {

navigate("/login-failed");

}

} else {

navigate("/login-failed");

}

};

return (

// ... (all the JSX code for the component)

);

};

export default Login;

In this updated Login.js, after a successful login, the JWT token is stored in the browser's local storage and passed to the onLogin callback. This allows the user to maintain their session, and you can use the token for subsequent API calls that require authentication.

Human: Greeting.js isn't showing any user data correctly... it's just stuck on Waiting.

Can you fix it so it works with the new Login.js and Login.php:

Here it is:

import React from "react";

import { useLocation } from "react-router-dom";

import styled from 'styled-components';

const Greeting = (props) => {

const location = useLocation();

const user = location.state?.user;

console.log(props);

if (!user) {

return <div>Loading...</div>;

}

const WrapperStyle = styled.div`

position: absolute;

color:white;

text-align: center;

background-color:white;

`

return (

<div>

<WrapperStyle>

<h1>Welcome, {user.first\_name}!</h1>

<p>Your role: {user.role\_id === 1 ? "Lot Operator" : "Advertiser"}</p>

<p>Your company: {user.company\_name}</p>

<p>Your email: {user.email}</p>

</WrapperStyle>

</div>

);

};

export default Greeting;

ChatGPT: Since the user data is now embedded in the JWT token, you need to decode the token to retrieve the user data. Here's how you can modify the Greeting.js file to work with the new Login.js and Login.php:First, install the jwt-decode package:bashCopy codenpm install jwt-decode

Then, update your Greeting.js file:javascriptCopy codeimport React, { useState, useEffect } from "react";

import { useLocation } from "react-router-dom";

import styled from 'styled-components';

import jwt\_decode from "jwt-decode";

const Greeting = () => {

const [user, setUser] = useState(null);

const location = useLocation();

useEffect(() => {

const token = localStorage.getItem("token");

if (token) {

const decodedToken = jwt\_decode(token);

setUser(decodedToken.user);

}

}, [location]);

if (!user) {

return <div>Loading...</div>;

}

// ... (WrapperStyle styled-component definition)

return (

<div>

<WrapperStyle>

<h1>Welcome, {user.first\_name}!</h1>

<p>Your role: {user.role\_id === 1 ? "Lot Operator" : "Advertiser"}</p>

<p>Your company: {user.company\_name}</p>

<p>Your email: {user.email}</p>

</WrapperStyle>

</div>

);

};

export default Greeting;

In the updated Greeting.js, we use the useState and useEffect hooks to set and update the user data. We get the JWT token from the local storage and use the jwt-decode package to decode the token and retrieve the user data, which we then set as the state. This ensures that the user data is displayed correctly in the Greeting component.

Human: Hmmm it still gets stuck on Loading... although I see an encrypted data exists from user from the console.log in login... so the data is getting in

ChatGPT: Apologies for the confusion. It seems the issue is related to the token not being passed to the Greeting.js component via location. Let's update the Login.js file to pass the token using location state when navigating to the greeting page.Modify the Login.js file as follows:javascriptCopy code// ... (Other imports and styled-components definitions)

const Login = () => {

const navigate = useNavigate();

const handleSignInSubmit = async (e) => {

e.preventDefault();

const email = e.target.elements[0].value;

const password = e.target.elements[1].value;

const response = await fetch("https://tomcookson.com/php2/login.php", {

method: "POST",

headers: {

"Content-Type": "application/json",

},

body: JSON.stringify({

email,

password,

}),

});

if (response.ok) {

const { token } = await response.json();

localStorage.setItem("token", token);

navigate("/greeting", { state: { token } }); // Pass the token via location state

} else {

navigate("/login-failed");

}

};

// ... (The rest of the Login component)

return (

// ... (JSX code)

);

};

export default Login;

Now, the token will be passed to the Greeting.js component using location state, and it should work as expected.

Human: I've got a useState that can be null or can be text and changes between those two things live. How can I render in the JSX code so it does crash React?

ChatGPT: You can conditionally render the content based on whether the state variable is null or not. Let's say your state variable is called text. You can use a ternary operator to check if text is null and render the content accordingly:javascriptCopy codeimport React, { useState } from "react";

const YourComponent = () => {

const [text, setText] = useState(null);

// ... (Other component logic)

return (

<div>

{text ? (

// Render this JSX if text is not null

<p>{text}</p>

) : (

// Render this JSX if text is null

<p>No text available</p>

)}

</div>

);

};

export default YourComponent;

In this example, if text is not null, it will display the value of text within a <p> element. If text is null, it will display "No text available" instead. This approach ensures that React doesn't crash due to the state variable being null.