Conversation Log 4/18/2023

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

Creation of error/success pages, page that shows all users, login page back end.

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

Human: I added all that code, thanks. Can you write Success.js and Error.js components for me too?

ChatGPT: Certainly! Here are simple Success.js and Error.js components:Success.js:javascriptCopy codeimport React from 'react';

import styled from 'styled-components';

const Container = styled.div`

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

background-color: #e0f7fa;

`;

const Message = styled.h1`

font-size: 2rem;

margin-bottom: 1rem;

`;

const Success = () => {

return (

<Container>

<Message>Account created successfully!</Message>

<p>Your account has been created. You can now log in and start using the service.</p>

</Container>

);

};

export default Success;

Error.js:javascriptCopy codeimport React from 'react';

import styled from 'styled-components';

const Container = styled.div`

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

background-color: #ffcdd2;

`;

const Message = styled.h1`

font-size: 2rem;

margin-bottom: 1rem;

`;

const Error = () => {

return (

<Container>

<Message>Error creating account</Message>

<p>There was an error creating your account. Please try again later or contact support if the problem persists.</p>

</Container>

);

};

export default Error;

These components display a simple success or error message and some additional text. You can customize the styling and content as needed. Remember to import them in your main component or App.js file and add the appropriate routes, as shown in the previous answer.

Human: How can I send the error message from SignUp to Error?

ChatGPT: You can pass the error message from the SignUp component to the Error component using React Router's state prop. First, modify the Error component to receive the error message through props:javascriptCopy code// Error.js

import React from 'react';

import styled from 'styled-components';

const Container = styled.div`

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

background-color: #ffcdd2;

`;

const Message = styled.h1`

font-size: 2rem;

margin-bottom: 1rem;

`;

const Error = (props) => {

const { errorMessage } = props.location.state;

return (

<Container>

<Message>Error creating account</Message>

<p>{errorMessage}</p>

</Container>

);

};

export default Error;

Next, update the SignUp component to pass the error message when redirecting to the Error page:javascriptCopy code// SignUp.js

// ...

import { Redirect } from 'react-router-dom';

const Signup = () => {

// ...

const [error, setError] = useState(null);

const [redirectToError, setRedirectToError] = useState(false);

const handleSignUpSubmit = async (e) => {

e.preventDefault();

// ...

try {

const response = await fetch('signup.php', {

// ...

});

if (response.ok) {

setRedirectToSuccess(true);

} else {

const errorMessage = await response.text();

setError(errorMessage);

setRedirectToError(true);

}

} catch (error) {

setError('An unexpected error occurred.');

setRedirectToError(true);

}

};

if (redirectToSuccess) {

return <Redirect to="/success" />;

}

if (redirectToError) {

return <Redirect to={{ pathname: '/error', state: { errorMessage: error } }} />;

}

// ...

};

Now, when the error occurs, the SignUp component will redirect to the Error page and pass the error message through the state prop. The Error component will then display the received error message.

Human: This is my current code for Signup.js:

// Error.js

import React from 'react';

import styled from 'styled-components';

const Container = styled.div`

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

background-color: #ffcdd2;

`;

const Message = styled.h1`

font-size: 2rem;

margin-bottom: 1rem;

`;

const Error = (props) => {

const { errorMessage } = props.location.state;

return (

<Container>

<Message>Error creating account</Message>

<p>{errorMessage}</p>

</Container>

);

};

export default Error;

How do I get the error message to Error?

ChatGPT: I apologize for the confusion. It seems you have pasted the Error.js component code instead of Signup.js. I'll reiterate the steps to pass the error message from Signup.js to Error.js.First, make sure you have imported the required dependencies in Signup.js:javascriptCopy codeimport { useState } from 'react';

import { Redirect } from 'react-router-dom';

Next, add the following state variables in the Signup component:javascriptCopy codeconst [error, setError] = useState(null);

const [redirectToError, setRedirectToError] = useState(false);

Then, in your handleSignUpSubmit function, update the error handling logic as follows:javascriptCopy codeconst handleSignUpSubmit = async (e) => {

e.preventDefault();

// ...

try {

const response = await fetch('signup.php', {

// ...

});

if (response.ok) {

setRedirectToSuccess(true);

} else {

const errorMessage = await response.text();

setError(errorMessage);

setRedirectToError(true);

}

} catch (error) {

setError('An unexpected error occurred.');

setRedirectToError(true);

}

};

Finally, add the following conditional rendering logic at the beginning of the Signup component's return statement:javascriptCopy codeif (redirectToError) {

return <Redirect to={{ pathname: '/error', state: { errorMessage: error } }} />;

}

This will redirect the user to the Error page and pass the error message through the state prop when there's an error. The Error component will then display the received error message.

Human: I'm using React v6 with this code for Signup:

import React from 'react';

import styled from 'styled-components';

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

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

const HomeContainer = styled.div`

background-color: black;

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 SignUpOrganizer = styled.div`

margin-top: 10vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding-left: 0;

`;

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;

background-color: black;

display: flex;

justify-content: center;

align-items: center;

display: inline-block;

font-size: 1.5rem;

margin-bottom: 0rem;

`;

const Input = styled.input`

font-size: 1rem;

padding: 0.5rem;

margin-bottom: 0.1rem;

width: 100%;

max-width: 300px;

`;

const LoginButton = styled.button`

font-size: 1rem;

padding: 0.5rem 1rem;

margin-top: 0.8rem;

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 SignUpForm = styled.form`

display: flex;

flex-direction: column;

align-items: center;

`;

const SignupButton = styled(LoginButton)`

margin-bottom: 0;

`;

const Select = styled.select`

font-size: 1rem;

padding: 0.5rem;

margin-bottom: 0.1rem;

width: 100%;

max-width: 300px;

`;

const Signup = () => {

const navigate = useNavigate();

const handleSignUpSubmit = async (e) => {

e.preventDefault();

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

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

const firstName = e.target.elements[2].value;

const lastName = e.target.elements[3].value;

const companyName = e.target.elements[4].value;

const companyAddress = e.target.elements[5].value;

const state = e.target.elements[6].value;

const city = e.target.elements[7].value;

const zip = e.target.elements[8].value;

const password = e.target.elements[9].value;

const role\_id = role === "parking\_lot\_owner" ? 1 : 2;

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

method: "POST",

headers: {

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

},

body: JSON.stringify({

role\_id,

email,

first\_name: firstName,

last\_name: lastName,

company\_name: companyName,

company\_address: companyAddress,

state,

city,

zip,

password,

}),

});

if (response.ok) {

const responseData = await response.json();

// Redirect to the success page

navigate('/success');

} else {

// Redirect to the error page

navigate('/error');

}

};

return (

<HomeContainer>

<HeroImage>

<SignUpOrganizer>

<SubHeading>Join us for parking lot monitoring or to advertise</SubHeading>

<SignUpForm onSubmit={handleSignUpSubmit}>

<Select required>

<option value="">Choose lot monitoring or advertising</option>

<option value="parking\_lot\_owner">Parking Lot Monitoring</option>

<option value="advertiser">Advertising</option>

</Select>

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

<Input type="text" placeholder="First Name" required />

<Input type="text" placeholder="Last Name" required />

<Input type="text" placeholder="Company Name" required />

<Input type="text" placeholder="Company Address" required />

<Select required>

<option value="">Select State...</option>

<option value="CT">CT</option>

<option value="NJ">NJ</option>

<option value="NY">NY</option>

</Select>

<Input type="text" placeholder="City" required />

<Input type="text" placeholder="ZIP" required />

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

<SignupButton type="submit">Sign Up</SignupButton>

</SignUpForm>

</SignUpOrganizer>

</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 Signup;

Does that change any part of the answer? It likes UseNavigate instead of useHistory

ChatGPT: Yes, since you are using React Router v6 and useNavigate, the process of passing the error message to the Error component changes slightly. Here's how to update the Signup.js component to pass the error message to the Error component using useNavigate:Add a state variable to store the error message:javascriptCopy codeimport { useState } from 'react';

// ...

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

Update the handleSignUpSubmit function to set the error message when there is an error:javascriptCopy codeif (response.ok) {

// Redirect to the success page

navigate('/success');

} else {

const errorText = await response.text();

setErrorMessage(errorText);

// Redirect to the error page

navigate('/error', { state: { errorMessage: errorText } });

}

In this case, when there's an error, we set the errorMessage state with the error text received from the server and pass it to the Error component using the state prop in the navigate function.Now, the Error component will receive the error message through the props.location.state.errorMessage property.

Human: Can you make it so the PHP page returns an error if a user is added and there are already 10 users in the system? There should never be more than 10 users.

I'm using your code right now:

<?php

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

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

header("Content-Type: application/json; charset=UTF-8");

header("Access-Control-Allow-Methods: POST");

// Replace these with your actual database credentials

$servername = "localhost";

$username = "your\_username";

$password = "your\_password";

$dbname = "your\_database\_name";

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

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$inputJSON = file\_get\_contents('php://input');

$input = json\_decode($inputJSON, TRUE);

$role\_id = $input["role\_id"];

$email = $input["email"];

$first\_name = $input["first\_name"];

$last\_name = $input["last\_name"];

$company\_name = $input["company\_name"];

$company\_address = $input["company\_address"];

$state = $input["state"];

$city = $input["city"];

$zip = $input["zip"];

$password = password\_hash($input["password"], PASSWORD\_DEFAULT);

$sql = "INSERT INTO users (role\_id, email, first\_name, last\_name, company\_name, company\_address, state, city, zip, password) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

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

$stmt->bind\_param("isssssssss", $role\_id, $email, $first\_name, $last\_name, $company\_name, $company\_address, $state, $city, $zip, $password);

if ($stmt->execute()) {

http\_response\_code(201);

echo json\_encode(["message" => "User created successfully"]);

} else {

http\_response\_code(400);

echo json\_encode(["error" => "Unable to create user"]);

}

$stmt->close();

$conn->close();

}

?>

ChatGPT: To limit the number of users to 10 in the system, you can modify the PHP code as follows:Check the number of users in the database before inserting a new user.If there are already 10 users, return an error.If there are less than 10 users, proceed with the user insertion.Here's the updated PHP code:phpCopy code<?php

// ... (keep the existing headers and database connection code)

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

// ... (keep the existing input data processing code)

// Check the number of users in the database

$userCountSql = "SELECT COUNT(\*) AS user\_count FROM users";

$userCountResult = $conn->query($userCountSql);

$userCountRow = $userCountResult->fetch\_assoc();

$userCount = $userCountRow["user\_count"];

if ($userCount >= 10) {

http\_response\_code(400);

echo json\_encode(["error" => "Unable to create user: Maximum number of users (10) reached"]);

} else {

$sql = "INSERT INTO users (role\_id, email, first\_name, last\_name, company\_name, company\_address, state, city, zip, password) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

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

$stmt->bind\_param("isssssssss", $role\_id, $email, $first\_name, $last\_name, $company\_name, $company\_address, $state, $city, $zip, $password);

if ($stmt->execute()) {

http\_response\_code(201);

echo json\_encode(["message" => "User created successfully"]);

} else {

http\_response\_code(400);

echo json\_encode(["error" => "Unable to create user"]);

}

$stmt->close();

}

$conn->close();

}

?>

Now, the PHP script will first check the number of users in the system. If there are already 10 users, it will return an error message. Otherwise, it will proceed with the user insertion as before.

Human: If a message is sent to an error page this way:

const errorText = await response.text();

setErrorMessage(errorText);

navigate('/error', { state: { errorMessage: errorText } });

How would you write Error.js so it shows the message?

ChatGPT: You can use the useLocation hook from react-router-dom to access the state you passed while navigating to the Error component. Here's how you can modify the Error.js component to display the error message:javascriptCopy codeimport React from 'react';

import styled from 'styled-components';

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

const Container = styled.div`

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

background-color: #ffcdd2;

`;

const Message = styled.h1`

font-size: 2rem;

margin-bottom: 1rem;

`;

const Error = () => {

const location = useLocation();

const errorMessage = location.state.errorMessage;

return (

<Container>

<Message>Error creating account</Message>

<p>{errorMessage}</p>

</Container>

);

};

export default Error;

This code uses the useLocation hook to get the current location object, and then extracts the errorMessage from the state object. The error message is then displayed inside a paragraph element.

Human: How do I access the Message Text and the Error seporately in Error

ChatGPT: If you want to separate the message text and the error details in the Error component, you can modify the state object passed while navigating to include both pieces of information separately. In your Signup.js, you can modify the code as follows:javascriptCopy code// Inside handleSignUpSubmit function, where you're navigating to /error

const errorText = await response.text();

const messageText = "Unable to create user";

setErrorMessage(errorText);

navigate('/error', { state: { message: messageText, errorDetails: errorText } });

Then in your Error.js component, you can access both the message text and error details like this:javascriptCopy codeimport React from 'react';

import styled from 'styled-components';

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

const Container = styled.div`

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

background-color: #ffcdd2;

`;

const Message = styled.h1`

font-size: 2rem;

margin-bottom: 1rem;

`;

const Error = () => {

const location = useLocation();

const message = location.state.message;

const errorDetails = location.state.errorDetails;

return (

<Container>

<Message>{message}</Message>

<p>{errorDetails}</p>

</Container>

);

};

export default Error;

Now, the Error.js component will display the message text in the Message styled component and the error details in a paragraph element separately.

Human: I have the following string in react: "{"error":"Unable to create user: Maximum number of users (10) reached"}" how can I break this into 2 strings, the first string being: error the second string beng Unable to create user: Maximum number of users (10) reached

ChatGPT: You can parse the JSON string into a JavaScript object and then access the properties. Here's an example of how to do this:javascriptCopy codeconst jsonString = '{"error":"Unable to create user: Maximum number of users (10) reached"}';

const jsonObject = JSON.parse(jsonString);

const errorKey = Object.keys(jsonObject)[0];

const errorMessage = jsonObject[errorKey];

console.log(errorKey); // "error"

console.log(errorMessage); // "Unable to create user: Maximum number of users (10) reached"

In this example, we first parse the JSON string using JSON.parse() to convert it into a JavaScript object. Then we use Object.keys() to get an array of the object's keys, and we access the first key using [0]. Finally, we access the value associated with the key in the jsonObject.

Human: Okay, so I love the way that PHP returns an error if over 10 users are created. Can you take the below code and make it so it also returns an error -- using the same method -- if a user enters an email address that has already been entered. This is the current signup.php code:

<?php

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

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

header("Content-Type: application/json; charset=UTF-8");

header("Access-Control-Allow-Methods: POST");

// Replace these with your actual database credentials

$servername = "localhost";

$username = "your\_username";

$password = "your\_password";

$dbname = "your\_database\_name";

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

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$inputJSON = file\_get\_contents('php://input');

$input = json\_decode($inputJSON, TRUE);

$role\_id = $input["role\_id"];

$email = $input["email"];

$first\_name = $input["first\_name"];

$last\_name = $input["last\_name"];

$company\_name = $input["company\_name"];

$company\_address = $input["company\_address"];

$state = $input["state"];

$city = $input["city"];

$zip = $input["zip"];

$password = password\_hash($input["password"], PASSWORD\_DEFAULT);

$userCountSql = "SELECT COUNT(\*) AS user\_count FROM users";

$userCountResult = $conn->query($userCountSql);

$userCountRow = $userCountResult->fetch\_assoc();

$userCount = $userCountRow["user\_count"];

if ($userCount >= 10) {

http\_response\_code(400);

echo json\_encode(["error" => "Unable to create user: Maximum number of users (10) reached"]);

} else {

$sql = "INSERT INTO users (role\_id, email, first\_name, last\_name, company\_name, company\_address, state, city, zip, password) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

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

$stmt->bind\_param("isssssssss", $role\_id, $email, $first\_name, $last\_name, $company\_name, $company\_address, $state, $city, $zip, $password);

if ($stmt->execute()) {

http\_response\_code(201);

echo json\_encode(["message" => "User created successfully"]);

} else {

http\_response\_code(400);

echo json\_encode(["error" => "Unable to create user"]);

}

$stmt->close();

}

$conn->close();

}

?>

ChatGPT: You can modify the existing code to check if the email already exists in the database before attempting to create the user. Here's the modified signup.php code:phpCopy code<?php

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

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

header("Content-Type: application/json; charset=UTF-8");

header("Access-Control-Allow-Methods: POST");

// Replace these with your actual database credentials

$servername = "localhost";

$username = "your\_username";

$password = "your\_password";

$dbname = "your\_database\_name";

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

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

if ($\_SERVER["REQUEST\_METHOD"] == "POST") {

$inputJSON = file\_get\_contents('php://input');

$input = json\_decode($inputJSON, TRUE);

$role\_id = $input["role\_id"];

$email = $input["email"];

$first\_name = $input["first\_name"];

$last\_name = $input["last\_name"];

$company\_name = $input["company\_name"];

$company\_address = $input["company\_address"];

$state = $input["state"];

$city = $input["city"];

$zip = $input["zip"];

$password = password\_hash($input["password"], PASSWORD\_DEFAULT);

$userCountSql = "SELECT COUNT(\*) AS user\_count FROM users";

$userCountResult = $conn->query($userCountSql);

$userCountRow = $userCountResult->fetch\_assoc();

$userCount = $userCountRow["user\_count"];

$emailCheckSql = "SELECT COUNT(\*) AS email\_count FROM users WHERE email = ?";

$emailCheckStmt = $conn->prepare($emailCheckSql);

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

$emailCheckStmt->execute();

$emailCheckResult = $emailCheckStmt->get\_result();

$emailCheckRow = $emailCheckResult->fetch\_assoc();

$emailCount = $emailCheckRow["email\_count"];

$emailCheckStmt->close();

if ($userCount >= 10) {

http\_response\_code(400);

echo json\_encode(["error" => "Unable to create user: Maximum number of users (10) reached"]);

} elseif ($emailCount > 0) {

http\_response\_code(400);

echo json\_encode(["error" => "Unable to create user: Email already exists"]);

} else {

$sql = "INSERT INTO users (role\_id, email, first\_name, last\_name, company\_name, company\_address, state, city, zip, password) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

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

$stmt->bind\_param("isssssssss", $role\_id, $email, $first\_name, $last\_name, $company\_name, $company\_address, $state, $city, $zip, $password);

if ($stmt->execute()) {

http\_response\_code(201);

echo json\_encode(["message" => "User created successfully"]);

} else {

http\_response\_code(400);

echo json\_encode(["error" => "Unable to create user"]);

}

$stmt->close();

}

$conn->close();

}

?>

This modification adds a new SQL query to check if the email address is already in the database before attempting to create the user. If the email address is found, an error message is sent back to the client, indicating that the email already exists.

Human: This is a school project, so I want a button to be on the signup page that resets the users table of the database to be prepopulated with a good mix of users for demonstration service.

The button should be in the signup page, it should trigger php to at 8 fake accounts with funny fake data. Make sure we can see what the passwords are since it is just a demo.

ChatGPT: To achieve this, you'll need to create a new PHP script to handle the reset and prepopulate the database with the demo users. Then, add a button on the signup page to call the script.Create a new PHP script called reset\_and\_prepopulate.php and place it in your server directory. This script will reset and prepopulate the users table with demo data:phpCopy code<?php

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

header("Content-Type: application/json; charset=UTF-8");

header("Access-Control-Allow-Methods: POST");

// Replace these with your actual database credentials

$servername = "localhost";

$username = "your\_username";

$password = "your\_password";

$dbname = "your\_database\_name";

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

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

// Truncate the users table

$truncateSql = "TRUNCATE TABLE users";

$conn->query($truncateSql);

// Demo users data

$demoUsers = [

[

"role\_id" => 1,

"email" => "john.doe@example.com",

"first\_name" => "John",

"last\_name" => "Doe",

"company\_name" => "John's Company",

"company\_address" => "123 Main St",

"state" => "CA",

"city" => "Los Angeles",

"zip" => "90001",

"password" => "john123"

],

// Add more demo users here

];

// Insert demo users into the users table

$insertSql = "INSERT INTO users (role\_id, email, first\_name, last\_name, company\_name, company\_address, state, city, zip, password) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

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

foreach ($demoUsers as $user) {

$passwordHash = password\_hash($user["password"], PASSWORD\_DEFAULT);

$stmt->bind\_param(

"isssssssss",

$user["role\_id"],

$user["email"],

$user["first\_name"],

$user["last\_name"],

$user["company\_name"],

$user["company\_address"],

$user["state"],

$user["city"],

$user["zip"],

$passwordHash

);

$stmt->execute();

}

$stmt->close();

$conn->close();

http\_response\_code(200);

echo json\_encode(["message" => "Users table reset and prepopulated with demo users"]);

?>

Add the following button to your signup page:htmlCopy code<button id="reset-and-prepopulate" type="button">Reset and Prepopulate Users</button>

Add the following JavaScript code to handle the button click and call the reset\_and\_prepopulate.php script:javascriptCopy codedocument.getElementById("reset-and-prepopulate").addEventListener("click", async function () {

const response = await fetch("path/to/reset\_and\_prepopulate.php", { method: "POST" });

if (response.ok) {

const data = await response.json();

alert(data.message);

} else {

alert("Error resetting and prepopulating users");

}

});

Replace path/to/reset\_and\_prepopulate.php with the actual path to the reset\_and\_prepopulate.php script on your server.When you click the "Reset and Prepopulate Users" button on the signup page, it will truncate the users table and prepopulate it with the demo users specified in the reset\_and\_prepopulate.php script. The demo users' passwords are stored in plain

Human: Can you generate 7 pretend users for reset and prepopulate. Can you make the fake values funny and be sure the states are NY, CT or NJ

ChatGPT: Sure! Here are seven pretend users with funny names, fake addresses in NY, CT, or NJ, and their passwords:phpCopy code// Demo users data

$demoUsers = [

[

"role\_id" => 1,

"email" => "funky.chicken@example.com",

"first\_name" => "Funky",

"last\_name" => "Chicken",

"company\_name" => "Cluckin' Good",

"company\_address" => "123 Cluck St",

"state" => "NY",

"city" => "New York",

"zip" => "10001",

"password" => "funky123"

],

[

"role\_id" => 1,

"email" => "jolly.giraffe@example.com",

"first\_name" => "Jolly",

"last\_name" => "Giraffe",

"company\_name" => "High Heads",

"company\_address" => "456 Tall Ave",

"state" => "CT",

"city" => "Hartford",

"zip" => "06103",

"password" => "jolly123"

],

[

"role\_id" => 1,

"email" => "curious.cat@example.com",

"first\_name" => "Curious",

"last\_name" => "Cat",

"company\_name" => "Whisker Whispers",

"company\_address" => "789 Purr St",

"state" => "NJ",

"city" => "Newark",

"zip" => "07102",

"password" => "curious123"

],

[

"role\_id" => 1,

"email" => "chatty.penguin@example.com",

"first\_name" => "Chatty",

"last\_name" => "Penguin",

"company\_name" => "Ice Breakers",

"company\_address" => "321 Waddle Ave",

"state" => "NY",

"city" => "Buffalo",

"zip" => "14201",

"password" => "chatty123"

],

[

"role\_id" => 1,

"email" => "happy.hippo@example.com",

"first\_name" => "Happy",

"last\_name" => "Hippo",

"company\_name" => "River Riders",

"company\_address" => "654 Splash St",

"state" => "CT",

"city" => "Bridgeport",

"zip" => "06604",

"password" => "happy123"

],

[

"role\_id" => 1,

"email" => "lively.lemur@example.com",

"first\_name" => "Lively",

"last\_name" => "Lemur",

"company\_name" => "Tree Jumpers",

"company\_address" => "987 Leap Ln",

"state" => "NJ",

"city" => "Jersey City",

"zip" => "07302",

"password" => "lively123"

],

[

"role\_id" => 1,

"email" => "daring.dolphin@example.com",

"first\_name" => "Daring",

"last\_name" => "Dolphin",

"company\_name" => "Ocean Acrobats",

"company\_address" => "147 Flipper Ct",

"state" => "NY",

"city" => "Rochester",

"zip" => "14604",

"password

Human: How can I add a button that runs reset-and-prepopulate in the footer of the following file:

import React from 'react';

import styled from 'styled-components';

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

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

import { useState } from 'react';

const HomeContainer = styled.div`

background-color: black;

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 SignUpOrganizer = styled.div`

margin-top: 10vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding-left: 0;

`;

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;

background-color: black;

display: flex;

justify-content: center;

align-items: center;

display: inline-block;

font-size: 1.5rem;

margin-bottom: 0rem;

`;

const Input = styled.input`

font-size: 1rem;

padding: 0.5rem;

margin-bottom: 0.1rem;

width: 100%;

max-width: 300px;

`;

const LoginButton = styled.button`

font-size: 1rem;

padding: 0.5rem 1rem;

margin-top: 0.8rem;

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 SignUpForm = styled.form`

display: flex;

flex-direction: column;

align-items: center;

`;

const SignupButton = styled(LoginButton)`

margin-bottom: 0;

`;

const Select = styled.select`

font-size: 1rem;

padding: 0.5rem;

margin-bottom: 0.1rem;

width: 100%;

max-width: 300px;

`;

const Signup = () => {

const navigate = useNavigate();

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

const handleSignUpSubmit = async (e) => {

e.preventDefault();

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

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

const firstName = e.target.elements[2].value;

const lastName = e.target.elements[3].value;

const companyName = e.target.elements[4].value;

const companyAddress = e.target.elements[5].value;

const state = e.target.elements[6].value;

const city = e.target.elements[7].value;

const zip = e.target.elements[8].value;

const password = e.target.elements[9].value;

const role\_id = role === "parking\_lot\_owner" ? 1 : 2;

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

method: "POST",

headers: {

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

},

body: JSON.stringify({

role\_id,

email,

first\_name: firstName,

last\_name: lastName,

company\_name: companyName,

company\_address: companyAddress,

state,

city,

zip,

password,

}),

});

if (response.ok) {

navigate('/success');

} else {

const errorText = await response.text();

setErrorMessage(errorText);

navigate('/error', { state: { errorMessage: errorText } });

}

};

return (

<HomeContainer>

<HeroImage>

<SignUpOrganizer>

<SubHeading>Join us for parking lot monitoring or to advertise</SubHeading>

<SignUpForm onSubmit={handleSignUpSubmit}>

<Select required>

<option value="">Choose lot monitoring or advertising</option>

<option value="parking\_lot\_owner">Parking Lot Monitoring</option>

<option value="advertiser">Advertising</option>

</Select>

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

<Input type="text" placeholder="First Name" required />

<Input type="text" placeholder="Last Name" required />

<Input type="text" placeholder="Company Name" required />

<Input type="text" placeholder="Company Address" required />

<Select required>

<option value="">Select State...</option>

<option value="CT">CT</option>

<option value="NJ">NJ</option>

<option value="NY">NY</option>

</Select>

<Input type="text" placeholder="City" required />

<Input type="text" placeholder="ZIP" required />

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

<SignupButton type="submit">Sign Up</SignupButton>

</SignUpForm>

</SignUpOrganizer>

</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>

<h2>Demonstrations Features (not for production)</h2>

<button id="reset-and-prepopulate" type="button">Reset and Prepopulate Users</button>

</Footer>

</HomeContainer>

);

};

export default Signup;

ChatGPT: You'll need to create a function to handle the reset and prepopulate button click and call the reset\_and\_prepopulate.php script from your React component. Here's how you can do it:Add a new async function called resetAndPrepopulate to the Signup component:javascriptCopy codeconst resetAndPrepopulate = async () => {

const response = await fetch("path/to/reset\_and\_prepopulate.php", { method: "POST" });

if (response.ok) {

const data = await response.json();

alert(data.message);

} else {

alert("Error resetting and prepopulating users");

}

};

Replace path/to/reset\_and\_prepopulate.php with the actual path to the reset\_and\_prepopulate.php script on your server.Update the "Reset and Prepopulate Users" button in the Footer component to call the resetAndPrepopulate function when clicked:htmlCopy code<button id="reset-and-prepopulate" type="button" onClick={resetAndPrepopulate}>Reset and Prepopulate Users</button>

Now, when you click the "Reset and Prepopulate Users" button on the signup page, it will call the reset\_and\_prepopulate.php script to truncate the users table and prepopulate it with the demo users specified in the script.

Human: Can you create an component that shows all current existing users of the database and all their information in one table?

ChatGPT: Sure! First, create a new PHP file called get\_users.php on your server to fetch all users and their information from the database:phpCopy code<?php

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

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

header("Content-Type: application/json; charset=UTF-8");

header("Access-Control-Allow-Methods: GET");

$servername = "localhost";

$username = "your\_username";

$password = "your\_password";

$dbname = "your\_database\_name";

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

if ($conn->connect\_error) {

die("Connection failed: " . $conn->connect\_error);

}

if ($\_SERVER["REQUEST\_METHOD"] == "GET") {

$sql = "SELECT \* FROM users";

$result = $conn->query($sql);

$users = array();

while ($row = $result->fetch\_assoc()) {

$users[] = $row;

}

http\_response\_code(200);

echo json\_encode($users);

$conn->close();

}

?>

Now, create a new React component called UsersList.js to display the users in a table:jsxCopy codeimport React, { useState, useEffect } from 'react';

import styled from 'styled-components';

const UsersTable = styled.table`

width: 100%;

border-collapse: collapse;

`;

const TableHeader = styled.th`

padding: 1rem;

border: 1px solid black;

`;

const TableCell = styled.td`

padding: 1rem;

border: 1px solid black;

`;

const UsersList = () => {

const [users, setUsers] = useState([]);

useEffect(() => {

const fetchUsers = async () => {

const response = await fetch("path/to/get\_users.php", { method: "GET" });

if (response.ok) {

const data = await response.json();

setUsers(data);

} else {

alert("Error fetching users");

}

};

fetchUsers();

}, []);

return (

<div>

<h1>All Users</h1>

<UsersTable>

<thead>

<tr>

<TableHeader>ID</TableHeader>

<TableHeader>Email</TableHeader>

<TableHeader>First Name</TableHeader>

<TableHeader>Last Name</TableHeader>

<TableHeader>Company Name</TableHeader>

<TableHeader>Company Address</TableHeader>

<TableHeader>State</TableHeader>

<TableHeader>City</TableHeader>

<TableHeader>ZIP</TableHeader>

</tr>

</thead>

<tbody>

{users.map((user) => (

<tr key={user.id}>

<TableCell>{user.id}</TableCell>

<TableCell>{user.email}</TableCell>

<TableCell>{user.first\_name}</TableCell>

<TableCell>{user.last\_name}</TableCell>

<TableCell>{user.company\_name}</TableCell>

<TableCell>{user.company\_address}</TableCell>

<TableCell>{user.state}</TableCell>

<TableCell>{user.city}</TableCell>

<TableCell>{user.zip}</TableCell>

</tr>

))}

</tbody>

</UsersTable>

</div>

);

};

export default UsersList;

Replace path/to/get\_users.php with the actual path to the get\_users.php script on your server.Now you can import and use the UsersList component in your application. The component will fetch and display all users and their information in a table.Remember to add a route for the UsersList component in your `App.js

Human: Can you add a button that leads to the UsersList in the footer of Signup?

ChatGPT: Certainly! You can add a button in the Footer of the Signup component to navigate to the UsersList component. First, make sure to add a route for the UsersList component in your App.js file (if you haven't done so already):jsxCopy codeimport UsersList from './components/UsersList';

// Inside your Routes component

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

Now, update the Signup component's Footer to include a button that navigates to the UsersList page:jsxCopy code// At the beginning of the Signup component

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

// Inside the Footer component

<Link to="/users">

<button type="button">View All Users</button>

</Link>

This will add a "View All Users" button to the Footer of the Signup component, which will navigate to the UsersList component when clicked.

Human: Can you make a ternery javascript operator where if a number is 1 it returns "parking lot operator" or 2 returns "advertiser"

ChatGPT: Sure! You can use a ternary operator to check if a number is 1 or 2 and return the corresponding string. Here's an example:javascriptCopy codeconst number = 1; // Replace with the actual number you want to check

const result = number === 1 ? "parking lot operator" : number === 2 ? "advertiser" : "unknown";

console.log(result); // Output: "parking lot operator"

In this example, the result variable will hold the string "parking lot operator" if the number is 1, "advertiser" if the number is 2, or "unknown" if the number is neither 1 nor 2. You can replace the number variable with the actual value you want to check.

Human: Make it so if role is 1 it shows "Lot Operator" and if roles is 2 it shows "Advertiser"

Here is the code:

{users.map((user) => (

<tr key={user.id}>

<TableCell>{user.id}</TableCell>

<TableCell>{user.role\_id}</TableCell>

<TableCell>{user.email}</TableCell>

<TableCell>{user.first\_name}</TableCell>

<TableCell>{user.last\_name}</TableCell>

<TableCell>{user.company\_name}</TableCell>

<TableCell>{user.company\_address}</TableCell>

<TableCell>{user.state}</TableCell>

<TableCell>{user.city}</TableCell>

<TableCell>{user.zip}</TableCell>

</tr>

))}

ChatGPT: You can use a ternary operator to display "Lot Operator" or "Advertiser" based on the value of user.role\_id. Update the second TableCell in the map function like this:jsxCopy code{users.map((user) => (

<tr key={user.id}>

<TableCell>{user.id}</TableCell>

<TableCell>{user.role\_id === 1 ? "Lot Operator" : user.role\_id === 2 ? "Advertiser" : "Unknown"}</TableCell>

<TableCell>{user.email}</TableCell>

<TableCell>{user.first\_name}</TableCell>

<TableCell>{user.last\_name}</TableCell>

<TableCell>{user.company\_name}</TableCell>

<TableCell>{user.company\_address}</TableCell>

<TableCell>{user.state}</TableCell>

<TableCell>{user.city}</TableCell>

<TableCell>{user.zip}</TableCell>

</tr>

))}

Now, the role\_id column will show "Lot Operator" if the role\_id is 1, "Advertiser" if the role\_id is 2, and "Unknown" for other values.

Human: Okay, you know what the database looks like and you know I'm using php. Everything you've had me do so far works great...

Now something more tricky... Can you help me make the Login page work, use PHP and MariaDB and React to do authentication.

Here is what the Login page looks like:

import React from 'react';

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 = () => {

const handleSignInSubmit = (e) => {

e.preventDefault();

console.log('Sign In form submitted');

};

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;

ChatGPT: To implement authentication with PHP, MariaDB, and React, follow these steps:Create a PHP script to handle login requests.Modify the React component to send a request to the PHP script with the user's email and password.Update the PHP script to query the database and validate the user's credentials.If the credentials are valid, start a session and store the user's information in the session.Here's a step-by-step guide:Step 1: Create a PHP script to handle login requestsCreate a new PHP file called login.php in your PHP backend folder:phpCopy code<?php

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

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

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

// Replace these with your database credentials

$servername = "localhost";

$username = "your\_username";

$password = "your\_password";

$dbname = "your\_db\_name";

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

if ($conn->connect\_error) {

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

}

$email = $\_POST["email"];

$password = $\_POST["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();

?>

Step 2: Modify the React component to send a request to the PHP scriptUpdate the handleSignInSubmit function in the Login component to send the user's email and password to the login.php script: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://yourwebsite.com/backend/login.php", {

method: "POST",

headers: {

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

},

body: JSON.stringify({

email,

password,

}),

});

const result = await response.json();

if (result.success) {

// Redirect the user to the dashboard or another protected page

console.log("Logged in successfully");

} else {

console.log(result.error);

}

};

Step 3: Update the PHP script to query the database and validate the user's credentialsThe login.php script already includes code to query the database and validate the user's credentials.Step 4: Start a session and store the user's information in the sessionThe login.php script also includes code to start a session and store the user's information in the session.Now, when a user submits the login form, the React component sends a request to the login.php script, which checks the database for a matching email and password. If the credentials are valid, a session is started, and the user's information is stored in the session.Make sure to replace the database credentials in the login.php file with your own. Also, update the URL in the fetch

Human: for some reason it doesn't isn't recognizing correct logins.

ChatGPT: I apologize for the confusion. I see that there is an issue with the code. The issue lies in sending the data as JSON from the React component, but trying to access it using $\_POST in the PHP script. We need to modify both the React component and the PHP script to handle the data correctly.Please follow these changes:Step 1: Update the PHP scriptUpdate the login.php script to read JSON data from the request body:phpCopy code<?php

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

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

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

// Replace these with your database credentials

$servername = "localhost";

$username = "your\_username";

$password = "your\_password";

$dbname = "your\_db\_name";

$conn = new mysqli($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();

?>

Step 2: Update the React componentNo changes are needed in the React component, as it already sends JSON data.Now, the login.php script should be able to read the JSON data sent from the React component and authenticate the user correctly. If you are still having issues, please double-check your database connection and credentials, as well as the structure of your users table.

Human: It works now thanks

ChatGPT: You're welcome! I'm glad it's working now. If you have any more questions or need further assistance, please don't hesitate to ask. Good luck with your project!

Human: How can I have a custom page that greets a user when they login with custom information about their account

ChatGPT: To create a custom greeting page for a user when they log in, you'll need to fetch the user's information from the database after they've been authenticated and then display that information on a new page.Here's a simple example to get you started:First, let's modify your handleSignInSubmit function to send a request to your PHP script, which will return user data if the login credentials are correct.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,

}),

});

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");

}

};

Next, create a new Greeting component in your React app. This component will display the custom user information.javascriptCopy code// src/components/Greeting.js

import React from "react";

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

const Greeting = () => {

const location = useLocation();

const user = location.state.user;

return (

<div>

<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 \*/}

</div>

);

};

export default Greeting;

Add a route for the Greeting component in your App.js:javascriptCopy code// ...

import Greeting from "./components/Greeting";

function App() {

return (

<div className="App">

<Router>

<Routes>

{/\* ... \*/}

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

</Routes>

</Router>

</div>

);

}

// ...

Now, when a user logs in with valid credentials, they'll be redirected to the /greeting page, which will display their custom information.