Conversation Log 4/21/2023

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

Adding all 6 roles, and giving them each different views with different links (roles and links added, but all links lead to blank pages).

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

Change navigation based on role

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

Human: I would like you to make the navigation have different links depending on what role is logged in. I will paste the login.js, greeting.js (that accesses data) and Navigation so you can see how it works and you can tell me how to do it.

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 = () => {

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();

console.log('token should be empty');

console.log(token);

localStorage.setItem("token", token);

if(typeof token !== "undefined"){

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

} else {

navigate("/login-failed");

}

} 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;

Greeting.js:

import React, { useState, useEffect } from "react";

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

import styled from 'styled-components';

import jwt\_decode from "jwt-decode";

const HomeContainer = styled.div`

align-items: center;

justify-content: center;

height: 100%;

`;

const roles\_dict = {

1: 'a parking lot operator',

2: 'an advertiser'

}

const SubHeading = styled.h2`

margin-top:20vh;

font-size: 2rem;

width: fit-content;

color: white;

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

padding: 0.5rem 1rem;

`;

const SubSubHeading = styled.p`

font-size: 1.5rem;

width: fit-content;

color: white;

padding: 0.5rem 1rem;

`;

const WrapperStyle = styled.div`

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

padding-left: 0;

`;

const Greeting = () => {

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

const location = useLocation();

useEffect(() => {

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

const decodedToken = jwt\_decode(token);

setUser(decodedToken);

}, [location]);

console.log(user);

return (

<HomeContainer>

<WrapperStyle>

{user ? (

<>

<SubHeading>Welcome back, {user.data.first\_name}</SubHeading>

<SubSubHeading>You are logged in as {roles\_dict[user.data.role\_id]}</SubSubHeading>

</>

) : (

<SubHeading>Data loading...</SubHeading>

)}

</WrapperStyle>

</HomeContainer>

);

};

export default Greeting;

Navigation.js:

import React, { useState, useEffect } from 'react';

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

import logo from '../images/parkezlogosmall2.png'

import styled from 'styled-components';

import theme from '../theme';

const LogoContainer = styled.div`

display: flex;

align-items: center;

padding-left: 1.5rem;

`;

const StyledNav = styled.nav`

display: flex;

justify-content: space-between;

align-items: center;

background-color: ${({ scrolled }) => (scrolled ? theme.primary : 'transparent')};

text-shadow: 2px 2px 4px rgba(0, 0, 0, 1);

padding: 0rem;

position: fixed;

top: 0;

left: 0;

right: 0;

z-index: 100;

transition: background-color 0.3s;

`;

const StyledUl = styled.ul`

display: flex;

justify-content: space-between;

list-style-type: none;

margin: 0em;

padding: 0;

`;

const StyledLi = styled.li`

display: flex;

margin-right: 1rem;

height: 100%;

`;

const StyledNavLink = styled(NavLink)`

text-decoration: none;

color: ${theme.accent};

font-weight: 600;

display: flex;

align-items: center;

width: 100%;

height: 100%;

padding: 1rem 0.5rem;

&.active {

background-color: rgba(57,130,142,0.35);

}

`;

const StyledButton = styled.div`

display: flex;

align-items: center;

background-color: transparent;

height: 100%;

transition: background-color 0.3s;

&:hover {

background-color: ${theme.secondary};

}

`;

const Logo = styled.div`

color: ${theme.accent};

font-size: 1.2rem;

font-weight: bold;

padding-left: 1.5rem;

`;

const Navigation = () => {

const [scrolled, setScrolled] = useState(false);

const scrollToTop = () => {

window.scrollTo({

top: 0,

behavior: 'smooth',

});

};

useEffect(() => {

const handleScroll = () => {

const isScrolled = window.scrollY > 10;

if (isScrolled !== scrolled) {

setScrolled(isScrolled);

}

};

document.addEventListener('scroll', handleScroll, { passive: true });

return () => {

// Clean up the event listener

document.removeEventListener('scroll', handleScroll);

};

}, [scrolled]);

return (

<StyledNav scrolled={scrolled}>

<LogoContainer>

<img

src={logo}

alt="ParkEzAI Logo"

style={{ height: '40px', width: '40px', marginRight: '0px' }}

/>

<Logo>ParkEz</Logo>

</LogoContainer>

<StyledUl>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/">Home</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/find-parking">Find Parking</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/about">About</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/signup">Sign Up</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

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

</StyledButton>

</StyledLi>

</StyledUl>

</StyledNav>

);

};

export default Navigation;

ChatGPT: You can achieve the desired functionality by adding a state for the user role in the Navigation component and updating the navigation links based on the role. You can use the useEffect hook to check for the token in localStorage and set the role state accordingly. Then, conditionally render navigation links based on the role. Here is an updated version of your Navigation.js:javascriptCopy codeimport React, { useState, useEffect } from 'react';

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

import logo from '../images/parkezlogosmall2.png'

import styled from 'styled-components';

import theme from '../theme';

import jwt\_decode from "jwt-decode";

// ... (Keep the existing styled components here)

const Navigation = () => {

const [scrolled, setScrolled] = useState(false);

const [userRole, setUserRole] = useState(null);

useEffect(() => {

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

if (token) {

const decodedToken = jwt\_decode(token);

setUserRole(decodedToken.data.role\_id);

}

}, []);

// ... (Keep the existing code for scrolling here)

const renderLinksByRole = () => {

const commonLinks = (

<>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/">Home</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/find-parking">Find Parking</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/about">About</StyledNavLink>

</StyledButton>

</StyledLi>

</>

);

if (userRole === 1) {

return (

<>

{commonLinks}

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/operator-dashboard">Operator Dashboard</StyledNavLink>

</StyledButton>

</StyledLi>

</>

);

} else if (userRole === 2) {

return (

<>

{commonLinks}

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/advertiser-dashboard">Advertiser Dashboard</StyledNavLink>

</StyledButton>

</StyledLi>

</>

);

} else {

return (

<>

{commonLinks}

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/signup">Sign Up</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

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

</StyledButton>

</StyledLi>

</>

);

}

};

return (

<StyledNav scrolled={scrolled}>

<LogoContainer>

<img

src={logo}

alt="ParkEzAI Logo"

style={{ height: '40px', width: '40px', marginRight: '0px' }}

/>

<Logo>ParkEz</Logo>

</LogoContainer>

<StyledUl>

{renderLinksByRole()}

</StyledUl>

</StyledNav>

);

};

export default Navigation;

In this code, I added a new state variable userRole and a new useEffect hook to set the role based on the token in localStorage. Then, I created a function renderLinksByRole that conditionally renders navigation links based on the user role. Finally, I replaced the previous navigation

Human: Navigate works and shows the correct Navigation menu but only after I refresh the page does it update the current links. How can I make it update the Navigation menu as soon as the token changes?

ChatGPT: You can update the Navigation menu as soon as the token changes by adding an event listener for the token update. You can create a custom event that will be triggered when the token changes and update the userRole accordingly. Here's how you can do that:Update the Login component to dispatch a custom event after setting the token in localStorage.Login.js:javascriptCopy code// ... (Keep the existing code)

const Login = () => {

// ... (Keep the existing code)

const handleSignInSubmit = async (e) => {

// ... (Keep the existing code)

if (response.ok) {

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

localStorage.setItem("token", token);

if(typeof token !== "undefined"){

const tokenUpdateEvent = new CustomEvent('tokenUpdate', { detail: token });

window.dispatchEvent(tokenUpdateEvent);

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

} else {

navigate("/login-failed");

}

} else {

navigate("/login-failed");

}

};

// ... (Keep the existing code)

};

export default Login;

Update the Navigation component to listen for the custom event and update the userRole accordingly.Navigation.js:javascriptCopy codeimport React, { useState, useEffect } from 'react';

// ... (Keep the existing imports)

// ... (Keep the existing styled components)

const Navigation = () => {

const [scrolled, setScrolled] = useState(false);

const [userRole, setUserRole] = useState(null);

useEffect(() => {

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

if (token) {

const decodedToken = jwt\_decode(token);

setUserRole(decodedToken.data.role\_id);

}

}, []);

useEffect(() => {

const handleTokenUpdate = (event) => {

const updatedToken = event.detail;

const decodedToken = jwt\_decode(updatedToken);

setUserRole(decodedToken.data.role\_id);

};

window.addEventListener('tokenUpdate', handleTokenUpdate);

return () => {

// Clean up the event listener

window.removeEventListener('tokenUpdate', handleTokenUpdate);

};

}, []);

// ... (Keep the existing code for scrolling and rendering links by role)

};

export default Navigation;

In this code, I added a custom event tokenUpdate in the Login component which is dispatched after setting the token in localStorage. Then, in the Navigation component, I added a new useEffect hook to listen for the tokenUpdate event and update the userRole accordingly.Now, the Navigation menu should update as soon as the token changes, without requiring a page refresh.

Human: Can you implement a functional logout button and add it to Navigation.js?

ChatGPT: Sure, I can help you implement a functional logout button and add it to the Navigation.js. Here's the updated Navigation.js:javascriptCopy codeimport React, { useState, useEffect } from 'react';

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

// ... (Keep the existing imports)

// ... (Keep the existing styled components)

const Navigation = () => {

const [scrolled, setScrolled] = useState(false);

const [userRole, setUserRole] = useState(null);

const navigate = useNavigate();

const handleLogout = () => {

localStorage.removeItem("token");

setUserRole(null);

navigate("/login");

};

useEffect(() => {

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

if (token) {

const decodedToken = jwt\_decode(token);

setUserRole(decodedToken.data.role\_id);

}

}, []);

useEffect(() => {

const handleTokenUpdate = (event) => {

const updatedToken = event.detail;

const decodedToken = jwt\_decode(updatedToken);

setUserRole(decodedToken.data.role\_id);

};

window.addEventListener('tokenUpdate', handleTokenUpdate);

return () => {

// Clean up the event listener

window.removeEventListener('tokenUpdate', handleTokenUpdate);

};

}, []);

// ... (Keep the existing code for scrolling and rendering links by role)

return (

<StyledNav scrolled={scrolled}>

{/\* ... (Keep the existing code for Logo and other links) \*/}

<StyledUl>

{/\* ... (Keep the existing code for other links) \*/}

{userRole && (

<StyledLi>

<StyledButton onClick={handleLogout}>

<StyledNavLink to="/login">Logout</StyledNavLink>

</StyledButton>

</StyledLi>

)}

</StyledUl>

</StyledNav>

);

};

export default Navigation;

In this code, I've added a handleLogout function to remove the token from localStorage, set the userRole to null, and navigate to the "/login" page. The logout button is rendered conditionally based on the presence of userRole, so it will only be displayed when the user is logged in.Now, the Navigation component should have a functional logout button that appears when the user is logged in.

Human: Can you create a table in a MariaDB 10 database that is a lookup table for these:

1, Lot Operator, customer

2, Advertiser, paying customer

3, Customer Support, employee

4, Lot Specialist, employee

5, Advertising Specialist, employee

6, Accountant, employee

Make the employee / customer column a True or False value since there's only 2 possibilities

ChatGPT: To create a table in a MariaDB 10 database that represents the lookup table for the given data, you can use the following SQL statement:sqlCopy codeCREATE TABLE roles (

id INT PRIMARY KEY,

role\_name VARCHAR(50) NOT NULL,

user\_type VARCHAR(50) NOT NULL,

is\_employee BOOLEAN NOT NULL

);

Next, insert the given data into the table:sqlCopy codeINSERT INTO roles (id, role\_name, user\_type, is\_employee)

VALUES

(1, 'Lot Operator', 'customer', FALSE),

(2, 'Advertiser', 'paying customer', FALSE),

(3, 'Customer Support', 'employee', TRUE),

(4, 'Lot Specialist', 'employee', TRUE),

(5, 'Advertising Specialist', 'employee', TRUE),

(6, 'Accountant', 'employee', TRUE);

This SQL statement will create a roles table with four columns: id, role\_name, user\_type, and is\_employee. The is\_employee column will store TRUE for employees and FALSE for customers. The INSERT INTO statement will then populate the table with the provided data.Please note that you need to have access to a MariaDB server and an existing database to execute these SQL statements. You can use a tool like phpMyAdmin, MySQL Workbench, or the command line to run these statements.

Human: Can you make a statement that inserts this data:

1,funky.chicken@example.com,Funky,Chicken,Cluckin' Good,123 Cluck St,NY,New York,10001

2,jolly.giraffe@example.com,Jolly,Giraffe,High Heads,456 Tall Ave,CT,Hartford,06103

3,curious.cat@parkez.com,Curious,Cat,Whisker Whispers,789 Purr St,NJ,Newark,07102

4,chatty.penguin@parkez.com,Chatty,Penguin,Ice Breakers,321 Waddle Ave,NY,Buffalo,14201

5,happy.hippo@parkez.com,Happy,Hippo,River Riders,654 Splash St,CT,Bridgeport,06604

6,lively.lemur@parkez.com,Lively,Lemur,Tree Jumpers,987 Leap Ln,NJ,Jersey City,07302

Into this table:

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

role\_id INT,

email VARCHAR(255) NOT NULL UNIQUE,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

company\_name VARCHAR(255) NOT NULL,

company\_address VARCHAR(255) NOT NULL,

state VARCHAR(2) NOT NULL,

city VARCHAR(255) NOT NULL,

zip VARCHAR(10) NOT NULL,

password VARCHAR(255) NOT NULL,

FOREIGN KEY (role\_id) REFERENCES roles(id)

);

ChatGPT: You can use the following SQL statement to insert the provided data into the users table:sqlCopy codeINSERT INTO users (

role\_id, email, first\_name, last\_name, company\_name,

company\_address, state, city, zip, password

) VALUES

(1, 'funky.chicken@example.com', 'Funky', 'Chicken', 'Cluckin\' Good', '123 Cluck St', 'NY', 'New York', '10001', ''),

(2, 'jolly.giraffe@example.com', 'Jolly', 'Giraffe', 'High Heads', '456 Tall Ave', 'CT', 'Hartford', '06103', ''),

(3, 'curious.cat@parkez.com', 'Curious', 'Cat', 'Whisker Whispers', '789 Purr St', 'NJ', 'Newark', '07102', ''),

(4, 'chatty.penguin@parkez.com', 'Chatty', 'Penguin', 'Ice Breakers', '321 Waddle Ave', 'NY', 'Buffalo', '14201', ''),

(5, 'happy.hippo@parkez.com', 'Happy', 'Hippo', 'River Riders', '654 Splash St', 'CT', 'Bridgeport', '06604', ''),

(6, 'lively.lemur@parkez.com', 'Lively', 'Lemur', 'Tree Jumpers', '987 Leap Ln', 'NJ', 'Jersey City', '07302', '');

Please note that I've left the password field empty in this statement. You should replace the empty strings with the hashed password for each user before inserting the data into your database. Storing plain-text passwords in the database is not secure, so be sure to hash them using a secure hashing algorithm like bcrypt before saving them.

Human: This is is a school project so the passwords are publically available. Can you make the password for each account the first\_name but all lowercase with 123 added to the end?

ChatGPT: Sure! You can use the following SQL statement to insert the data with the desired password format:sqlCopy codeINSERT INTO users (

role\_id, email, first\_name, last\_name, company\_name,

company\_address, state, city, zip, password

) VALUES

(1, 'funky.chicken@example.com', 'Funky', 'Chicken', 'Cluckin\' Good', '123 Cluck St', 'NY', 'New York', '10001', 'funky123'),

(2, 'jolly.giraffe@example.com', 'Jolly', 'Giraffe', 'High Heads', '456 Tall Ave', 'CT', 'Hartford', '06103', 'jolly123'),

(3, 'curious.cat@parkez.com', 'Curious', 'Cat', 'Whisker Whispers', '789 Purr St', 'NJ', 'Newark', '07102', 'curious123'),

(4, 'chatty.penguin@parkez.com', 'Chatty', 'Penguin', 'Ice Breakers', '321 Waddle Ave', 'NY', 'Buffalo', '14201', 'chatty123'),

(5, 'happy.hippo@parkez.com', 'Happy', 'Hippo', 'River Riders', '654 Splash St', 'CT', 'Bridgeport', '06604', 'happy123'),

(6, 'lively.lemur@parkez.com', 'Lively', 'Lemur', 'Tree Jumpers', '987 Leap Ln', 'NJ', 'Jersey City', '07302', 'lively123');

This statement sets the password for each account to the lowercase first name followed by '123'. However, please remember that storing plain-text passwords is not recommended for real-world applications due to security concerns.

Human: this the below 2 databases, is there any way to indicate that role\_id in users, references the id in roles?

CREATE TABLE roles (

id INT PRIMARY KEY,

role\_name VARCHAR(50) NOT NULL,

user\_type VARCHAR(50) NOT NULL,

is\_employee BOOLEAN NOT NULL

);

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

role\_id INT,

email VARCHAR(255) NOT NULL UNIQUE,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

company\_name VARCHAR(255) NOT NULL,

company\_address VARCHAR(255) NOT NULL,

state VARCHAR(2) NOT NULL,

city VARCHAR(255) NOT NULL,

zip VARCHAR(10) NOT NULL,

password VARCHAR(255) NOT NULL,

FOREIGN KEY (role\_id) REFERENCES roles(id)

);

ChatGPT: Yes, you have already indicated that role\_id in the users table references the id in the roles table using the FOREIGN KEY constraint:sqlCopy codeFOREIGN KEY (role\_id) REFERENCES roles(id)

This line establishes a relationship between the users and roles tables, where each role\_id in the users table must match an id in the roles table. This enforces referential integrity and ensures that you cannot insert a user with a non-existent role.

Human: Great! Thanks

ChatGPT: You're welcome! If you have any more questions or need further assistance, feel free to ask. Good luck with your school project!

Human: Can you update this PHP code:

<?php

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

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

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

$servername = "";

$username = "";

$password = "";

$dbname = "";

$conn = new mysqli($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" => "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" => 2,

"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" => 2,

"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" => 2,

"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" => "daring123"

]

];

// 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"]);

?>

So $demoUsers matches the data in the insert statement above:

INSERT INTO users (

role\_id, email, first\_name, last\_name, company\_name,

company\_address, state, city, zip, password

) VALUES

(1, 'funky.chicken@example.com', 'Funky', 'Chicken', 'Cluckin\' Good', '123 Cluck St', 'NY', 'New York', '10001', 'funky123'),

(2, 'jolly.giraffe@example.com', 'Jolly', 'Giraffe', 'High Heads', '456 Tall Ave', 'CT', 'Hartford', '06103', 'jolly123'),

(3, 'curious.cat@parkez.com', 'Curious', 'Cat', 'Whisker Whispers', '789 Purr St', 'NJ', 'Newark', '07102', 'curious123'),

(4, 'chatty.penguin@parkez.com', 'Chatty', 'Penguin', 'Ice Breakers', '321 Waddle Ave', 'NY', 'Buffalo', '14201', 'chatty123'),

(5, 'happy.hippo@parkez.com', 'Happy', 'Hippo', 'River Riders', '654 Splash St', 'CT', 'Bridgeport', '06604', 'happy123'),

(6, 'lively.lemur@parkez.com', 'Lively', 'Lemur', 'Tree Jumpers', '987 Leap Ln', 'NJ', 'Jersey City', '07302', 'lively123');

ChatGPT: I've updated the $demoUsers array in the PHP code to match the data in the INSERT statement you provided:phpCopy code<?php

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

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

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

$servername = "";

$username = "";

$password = "";

$dbname = "";

$conn = new mysqli($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" => "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" => 2,

"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" => 3,

"email" => "curious.cat@parkez.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" => 4,

"email" => "chatty.penguin@parkez.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" => 5,

"email" => "happy.hippo@parkez.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" => 6,

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

"first\_name" => "Lively",

"last\_name" => "Lemur",

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

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

"state" => "NJ",

"city" => "Jersey

Human: I have a menu that responds to roles 1 and 2, but now we have roles 1, 2, 3, 4, 5 and 6. The buttons Home, Find Parking, About, Sign Up and Login only make sense for someone who hasn't logged in yet, so remove those options for all roles (except when user is logged out). The roles correspond to the roles table above.

For role 1 (Lot Operator) have a Parking Lot Dashboard, an Archive, a support,a Anaylsis, and Billing button.

For role 2 (Advertisor), have a Advertisements Dashboard, an Archive, a support,a Anaylsis, and Billing button.

For role 3 (Customer Support) have a Support tickets, a Lot Admin, a Ad Admin, billing admin, and User Account Admin buttons.

For role 4 (Lot Specialist) have a Support tickets, Lot Admin, Lot billing admin, and Lot Account Admin buttons.

For role 5 (Advertising Specialist) have a Support tickets, Ad Admin, Adbilling admin, and Ad Account Admin buttons.

For role 6 (Accountant) have Billing, Employee Admin)

All logged in accounts should have an account button as well.

This is what the navigation currently looks like

import React, { useState, useEffect } from 'react';

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

import logo from '../images/parkezlogosmall2.png'

import styled from 'styled-components';

import theme from '../theme';

import jwt\_decode from "jwt-decode";

const LogoContainer = styled.div`

display: flex;

align-items: center;

padding-left: 1.5rem;

`;

const StyledNav = styled.nav`

display: flex;

justify-content: space-between;

align-items: center;

background-color: ${({ scrolled }) => (scrolled ? theme.primary : 'transparent')};

text-shadow: 2px 2px 4px rgba(0, 0, 0, 1);

padding: 0rem;

position: fixed;

top: 0;

left: 0;

right: 0;

z-index: 100;

transition: background-color 0.3s;

`;

const StyledUl = styled.ul`

display: flex;

justify-content: space-between;

list-style-type: none;

margin: 0em;

padding: 0;

`;

const StyledLi = styled.li`

display: flex;

margin-right: 1rem;

height: 100%;

`;

const StyledNavLink = styled(NavLink)`

text-decoration: none;

color: ${theme.accent};

font-weight: 600;

display: flex;

align-items: center;

width: 100%;

height: 100%;

padding: 1rem 0.5rem;

&.active {

background-color: rgba(57,130,142,0.35);

}

`;

const StyledButton = styled.div`

display: flex;

align-items: center;

background-color: transparent;

height: 100%;

transition: background-color 0.3s;

&:hover {

background-color: ${theme.secondary};

}

`;

const Logo = styled.div`

color: ${theme.accent};

font-size: 1.2rem;

font-weight: bold;

padding-left: 1.5rem;

`;

const Navigation = () => {

const [scrolled, setScrolled] = useState(false);

const [userRole, setUserRole] = useState(null);

const navigate = useNavigate();

const handleLogout = () => {

localStorage.removeItem("token");

setUserRole(null);

navigate("/login");

};

const scrollToTop = () => {

window.scrollTo({

top: 0,

behavior: 'smooth',

});

};

useEffect(() => {

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

if (token) {

const decodedToken = jwt\_decode(token);

setUserRole(decodedToken.data.role\_id);

}

}, []);

useEffect(() => {

const handleTokenUpdate = (event) => {

const updatedToken = event.detail;

const decodedToken = jwt\_decode(updatedToken);

setUserRole(decodedToken.data.role\_id);

};

window.addEventListener('tokenUpdate', handleTokenUpdate);

return () => {

window.removeEventListener('tokenUpdate', handleTokenUpdate);

};

}, []);

useEffect(() => {

const handleScroll = () => {

const isScrolled = window.scrollY > 10;

if (isScrolled !== scrolled) {

setScrolled(isScrolled);

}

};

document.addEventListener('scroll', handleScroll, { passive: true });

return () => {

// Clean up the event listener

document.removeEventListener('scroll', handleScroll);

};

}, [scrolled]);

const renderLinksByRole = () => {

const commonLinks = (

<>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/">Home</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/find-parking">Find Parking</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/about">About</StyledNavLink>

</StyledButton>

</StyledLi>

</>

);

if (userRole === 1) {

return (

<>

{commonLinks}

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/operator-dashboard">Operator Dashboard</StyledNavLink>

</StyledButton>

</StyledLi>

</>

);

} else if (userRole === 2) {

return (

<>

{commonLinks}

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/advertiser-dashboard">Advertiser Dashboard</StyledNavLink>

</StyledButton>

</StyledLi>

</>

);

} else {

return (

<>

{commonLinks}

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/signup">Sign Up</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

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

</StyledButton>

</StyledLi>

</>

);

}

};

return (

<StyledNav scrolled={scrolled}>

<LogoContainer>

<img

src={logo}

alt="ParkEzAI Logo"

style={{ height: '40px', width: '40px', marginRight: '0px' }}

/>

<Logo>ParkEz</Logo>

</LogoContainer>

<StyledUl>

{renderLinksByRole()}

</StyledUl>

<StyledUl>

{/\* ... (Keep the existing code for other links) \*/}

{userRole && (

<StyledLi>

<StyledButton onClick={handleLogout}>

<StyledNavLink to="/login">Logout</StyledNavLink>

</StyledButton>

</StyledLi>

)}

</StyledUl>

</StyledNav>

);

};

export default Navigation;

ChatGPT: I have updated the Navigation component to account for the new roles and the desired menu items for each role:jsxCopy codeimport React, { useState, useEffect } from 'react';

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

import logo from '../images/parkezlogosmall2.png';

import styled from 'styled-components';

import theme from '../theme';

import jwt\_decode from 'jwt-decode';

// (Keep the existing styled components)

const Navigation = () => {

// (Keep the existing state, functions, and useEffect hooks)

const renderLinksByRole = () => {

const loggedInLinks = (

<StyledLi>

<StyledButton onClick={handleLogout}>

<StyledNavLink to="/login">Logout</StyledNavLink>

</StyledButton>

</StyledLi>

);

if (!userRole) {

return (

<>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/signup">Sign Up</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

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

</StyledButton>

</StyledLi>

</>

);

} else {

const roleLinks = {

1: (

<>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/operator-dashboard">Parking Lot Dashboard</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/archive">Archive</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/support">Support</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/analysis">Analysis</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/billing">Billing</StyledNavLink>

</StyledButton>

</StyledLi>

</>

),

2: (

<>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/advertiser-dashboard">Advertisements Dashboard</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/archive">Archive</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/support">Support</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/analysis">Analysis</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/billing">Billing</StyledNavLink>

</StyledButton>

</StyledLi>

</>

),

3: (

<>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/support-tickets">Support tickets</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/lot-admin">Lot Admin</StyledNavLink>

</StyledButton>

</

Human: Can you finish the previous response?

ChatGPT: Certainly, here's the continuation of the previous response with the Navigation component updated to include menu items for all roles:jsxCopy code </StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/ad-admin">Ad Admin</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/billing-admin">Billing Admin</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/user-account-admin">User Account Admin</StyledNavLink>

</StyledButton>

</StyledLi>

</>

),

4: (

<>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/support-tickets">Support tickets</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/lot-admin">Lot Admin</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/lot-billing-admin">Lot Billing Admin</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/lot-account-admin">Lot Account Admin</StyledNavLink>

</StyledButton>

</StyledLi>

</>

),

5: (

<>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/support-tickets">Support tickets</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/ad-admin">Ad Admin</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/ad-billing-admin">Ad Billing Admin</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/ad-account-admin">Ad Account Admin</StyledNavLink>

</StyledButton>

</StyledLi>

</>

),

6: (

<>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/billing">Billing</StyledNavLink>

</StyledButton>

</StyledLi>

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/employee-admin">Employee Admin</StyledNavLink>

</StyledButton>

</StyledLi>

</>

),

};

return (

<>

{roleLinks[userRole]}

<StyledLi>

<StyledButton onClick={scrollToTop}>

<StyledNavLink to="/account">Account</StyledNavLink>

</StyledButton>

</StyledLi>

{loggedInLinks}

</>

);

}

};

return (

<StyledNav scrolled={scrolled}>

<LogoContainer>

<img

src={logo}

alt="ParkEzAI Logo"

style={{ height: '40px', width: '40px', marginRight: '0px' }}

/>

<Logo>ParkEz</Logo>

</LogoContainer>

<StyledUl>{renderLinksByRole()}</StyledUl>

</StyledNav>

);

};

export default Navigation;

``