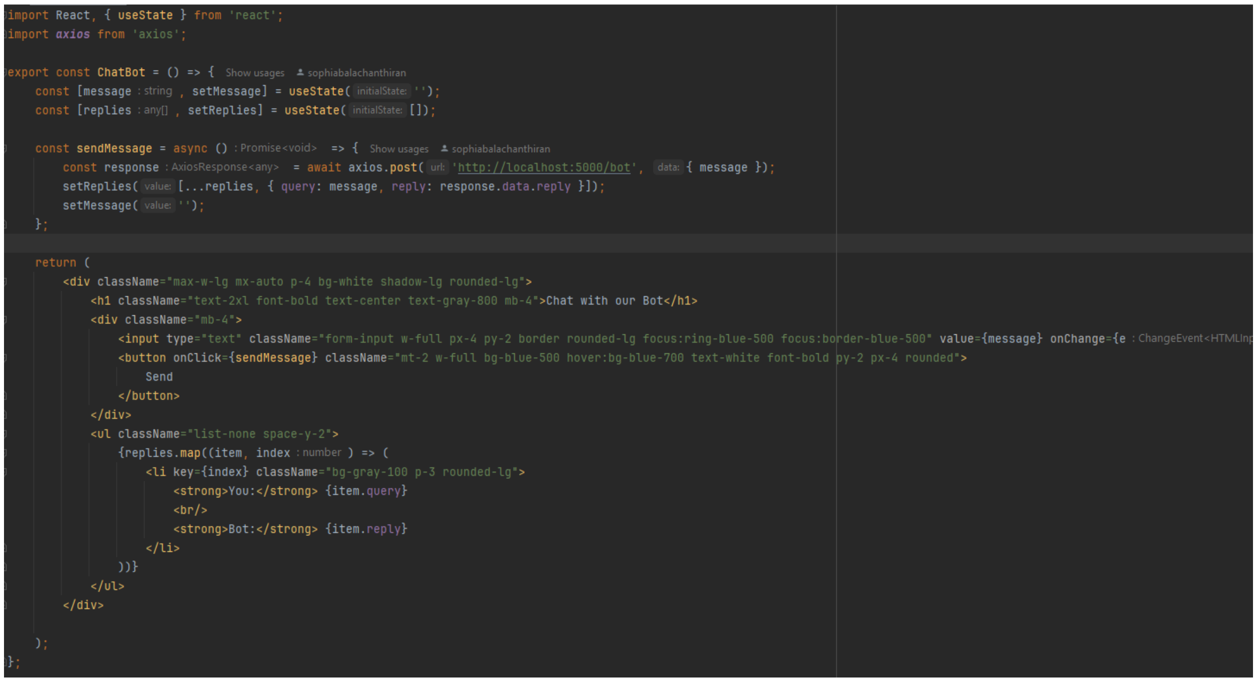
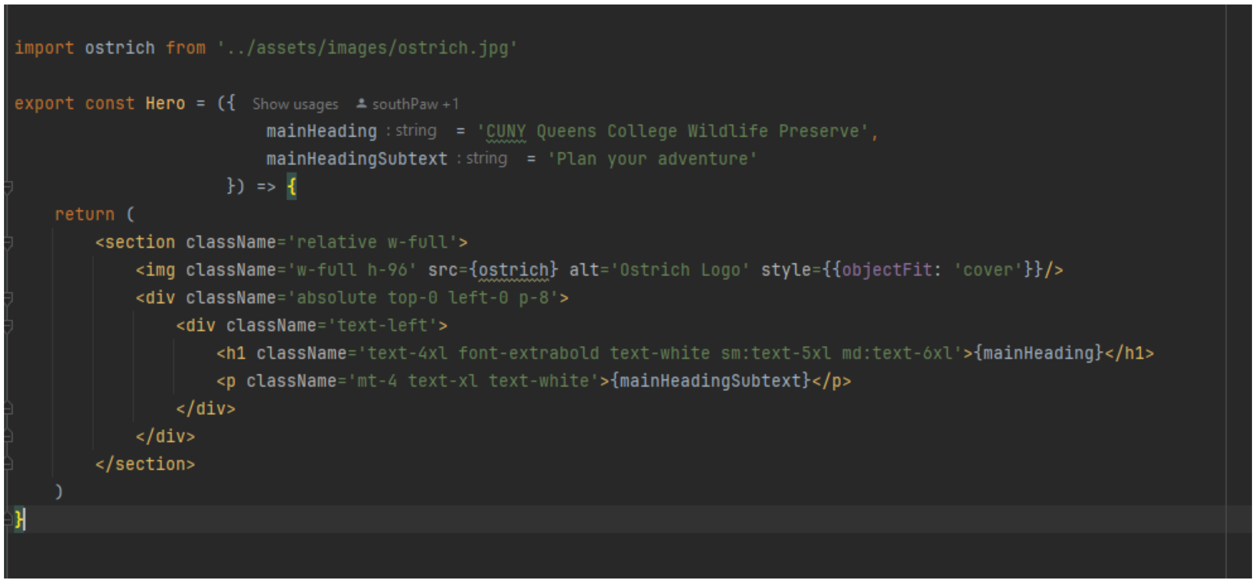
Sophia Balachanthiran, Ardit Hoxhaj, Jaskiran Parmar, Rayan Ahlouche, Daniel Gargiullo
CSCI 355
Professor Fried
Gray Paper

Home Page:

1. 

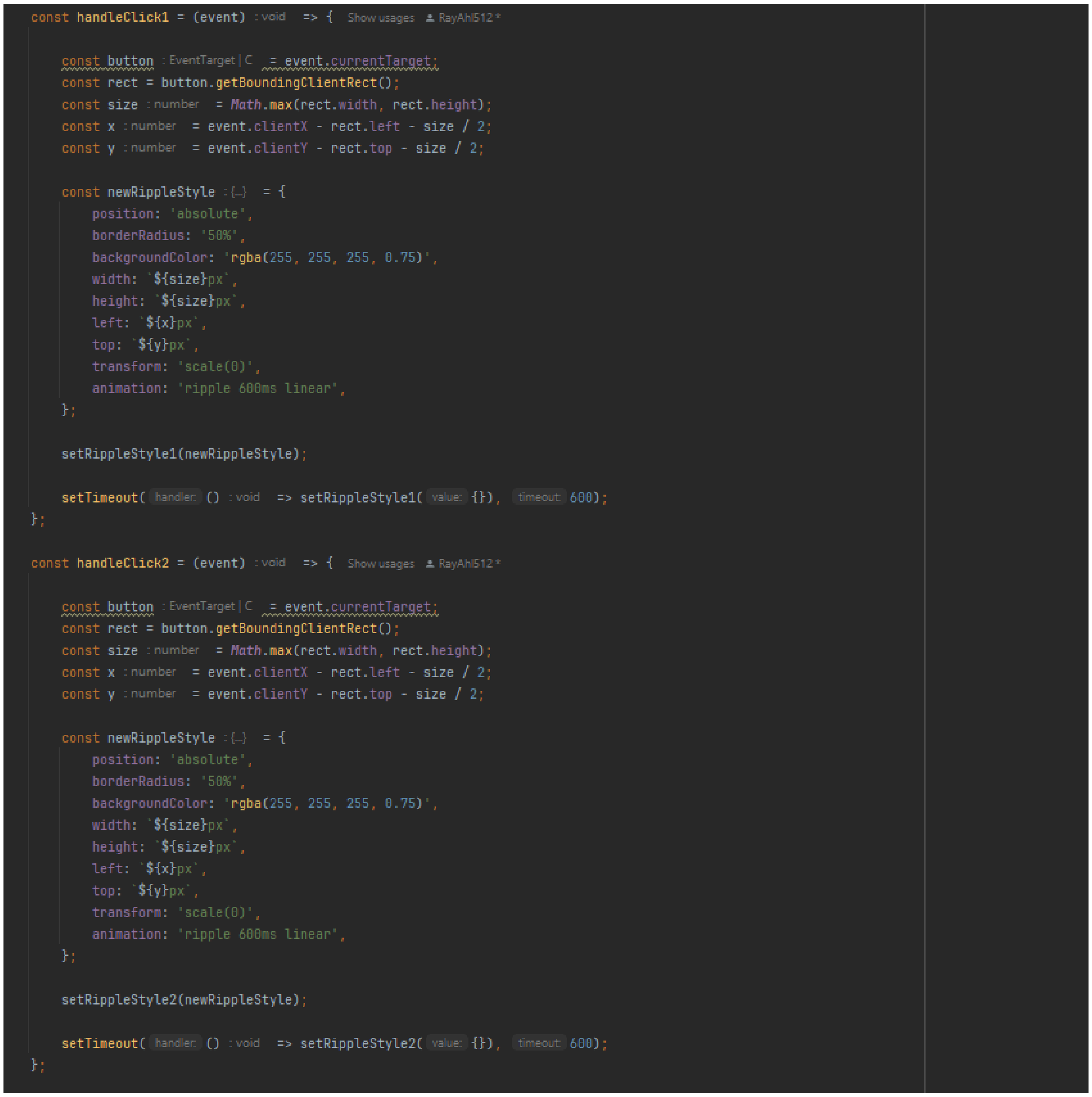
```
import React, { useState } from 'react';
import axios from 'axios';

export const ChatBot = () => {  Show usages  sophiabalachanthiran
    const [message : string , setMessage] = useState( initialState: '');
    const [replies : any[] , setReplies] = useState( initialState: []);

    const sendMessage = async () : Promise<void>  => {  Show usages  sophiabalachanthiran
        const response : AxiosResponse<any>  = await axios.post( url: 'http://localhost:5000/bot',  data: { message });
        setReplies( value: [...replies, { query: message, reply: response.data.reply }]);
        setMessage( value: '');
    };

    return (
        <div className="max-w-lg mx-auto p-4 bg-white shadow-lg rounded-lg">
            <h1 className="text-2xl font-bold text-center text-gray-800 mb-4">Chat with our Bot</h1>
            <div className="mb-4">
                <input type="text" className="form-input w-full px-4 py-2 border rounded-lg focus:ring-blue-500 focus:border-blue-500" value={message} onChange={e : ChangeEvent<HTMLInp
                <button onClick={sendMessage} className="mt-2 w-full bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4 rounded">
                    Send
                </button>
            </div>
            <ul className="list-none space-y-2">
                {replies.map((item, index : number ) => (
                    <li key={index} className="bg-gray-100 p-3 rounded-lg">
                        <strong>You:</strong> {item.query}
                        <br/>
                        <strong>Bot:</strong> {item.reply}
                    </li>
                ))}
            </ul>
        </div>

    );
};
```

Code defines a React component called ChatBot. There is state management using the ‘useState’ hook from React to manage the variables ‘message’ and ‘replies’. The function ‘sendMessage’ is called when the user sends a message, using a POST request to the local server with the user’s message. When receiving a response, the ‘replies’ state updates with the user’s question and the bot’s reply, then clearing the state of ‘message’. The UI consists of a heading called “Chat with our Bot”, an input field, a button labeled ‘Send’ and a list that displays the conversation history. There are multiple event handlers, ‘onChange’ updates the ‘message’ state as the user types into the input field, ‘onKeyPress’ triggers the ‘sendMessage’ function when the user presses the enter key, and the ‘onClick’ triggers the ‘sendMessage’ function. There are Tailwind CSS classes on the layout, input field, button, and list items.

2. 

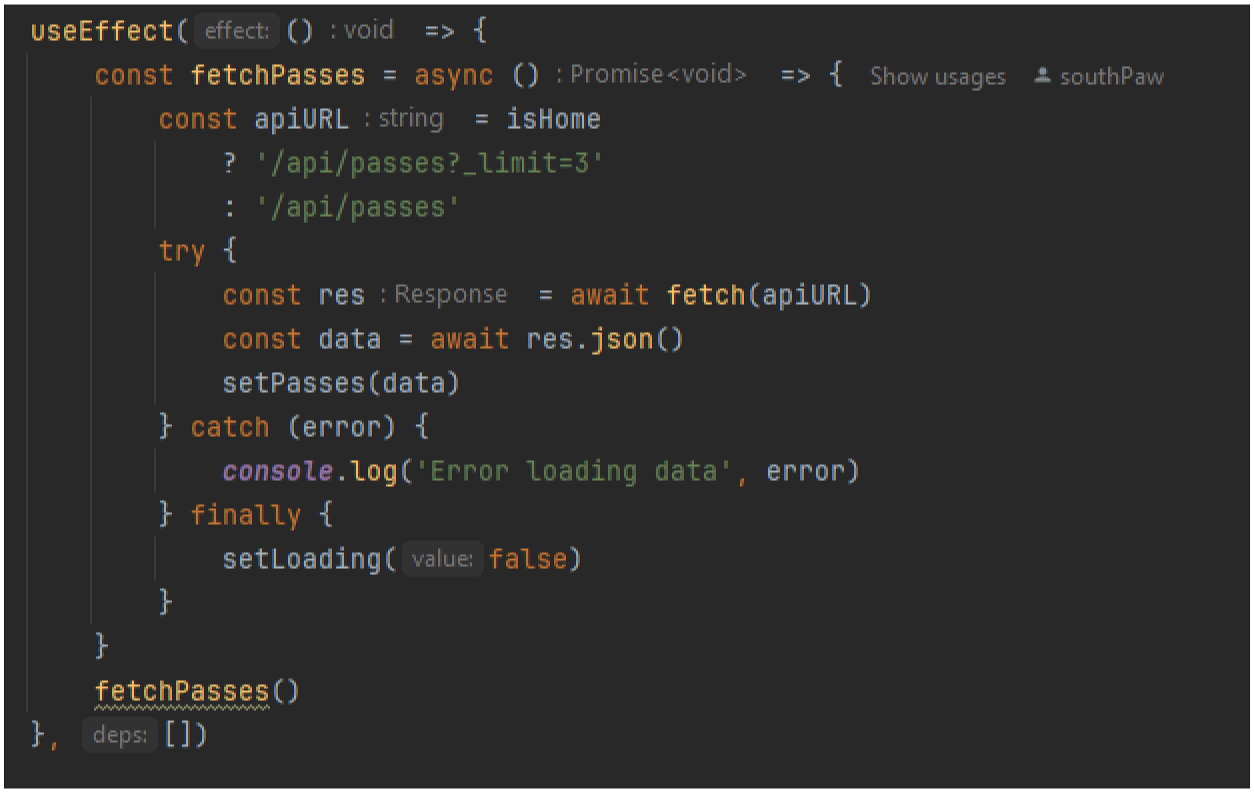
```
import ostrich from '../assets/images/ostrich.jpg'

export const Hero = ({  Show usages  southPaw +1
                         mainHeading : string  = 'CUNY Queens College Wildlife Preserve',
                         mainHeadingSubtext : string  = 'Plan your adventure'
                     }) => {
    return (
        <section className='relative w-full'>
            <img className='w-full h-96' src={ostrich} alt='Ostrich Logo' style={{objectFit: 'cover'}}/>
            <div className='absolute top-0 left-0 p-8'>
                <div className='text-left'>
                    <h1 className='text-4xl font-extrabold text-white sm:text-5xl md:text-6xl'>{mainHeading}</h1>
                    <p className='mt-4 text-xl text-white'>{mainHeadingSubtext}</p>
                </div>
            </div>
        </section>
    )
}
```

Code defines a React component called ‘Hero’, accepting two props, ‘mainHeading’ which is the main heading text displayed, and ‘mainHeadingSubtext’ which displays subText below the main heading. Within the ‘section’ there are ‘img’ and ‘div’ elements, ‘img’ displays an image covering the entire width of its container, and ‘div’ is absolutely positioned containing ‘mainHeading’ and ‘mainHeadingSubtext’. There are Tailwind CSS classes for layout, typography, and positioning.

3. 

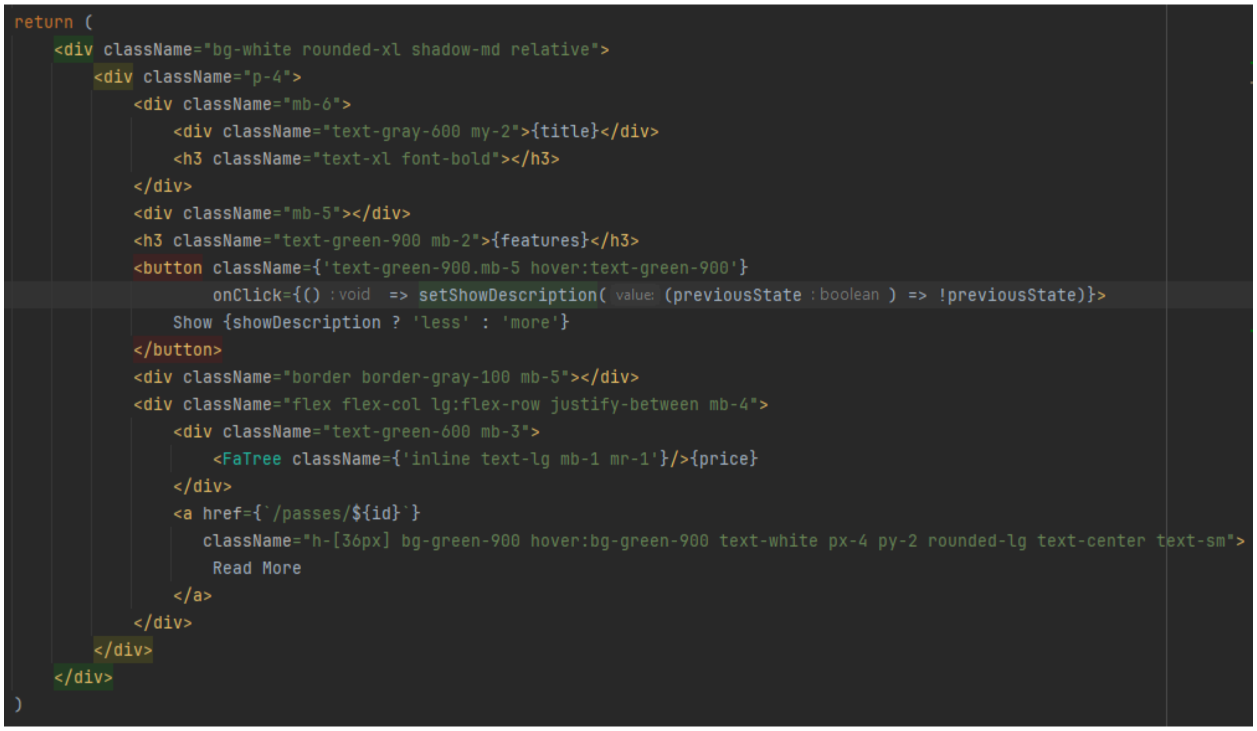
```
const handleClick1 = (event) :void  => {  Show usages  RayAhl512 *

    const button :EventTarget | C  = event.currentTarget;
    const rect = button.getBoundingClientRect();
    const size :number  = Math.max(rect.width, rect.height);
    const x :number  = event.clientX - rect.left - size / 2;
    const y :number  = event.clientY - rect.top - size / 2;

    const newRippleStyle :{...}  = {
        position: 'absolute',
        borderRadius: '50%',
        backgroundColor: 'rgba(255, 255, 255, 0.75)',
        width: `${size}px`,
        height: `${size}px`,
        left: `${x}px`,
        top: `${y}px`,
        transform: 'scale(0)',
        animation: 'ripple 600ms linear',
    };

    setRippleStyle1(newRippleStyle);

    setTimeout( handler: () :void  => setRippleStyle1( value: {}),  timeout: 600);
};

const handleClick2 = (event) :void  => {  Show usages  RayAhl512 *

    const button :EventTarget | C  = event.currentTarget;
    const rect = button.getBoundingClientRect();
    const size :number  = Math.max(rect.width, rect.height);
    const x :number  = event.clientX - rect.left - size / 2;
    const y :number  = event.clientY - rect.top - size / 2;

    const newRippleStyle :{...}  = {
        position: 'absolute',
        borderRadius: '50%',
        backgroundColor: 'rgba(255, 255, 255, 0.75)',
        width: `${size}px`,
        height: `${size}px`,
        left: `${x}px`,
        top: `${y}px`,
        transform: 'scale(0)',
        animation: 'ripple 600ms linear',
    };

    setRippleStyle2(newRippleStyle);

    setTimeout( handler: () :void  => setRippleStyle2( value: {}),  timeout: 600);
};
```

Code defines two functions acting as event handlers handling clicks on buttons, ‘handleClick1’ and ‘handleClick2’ within the ‘HomeCards’ component. Creates a visual ripple effect at the position where the user clicks. The ‘handleClick1’ function calculates the position of the click relative to the button that triggered the event. It creates a new objected ‘newRippleStyle’ for the ripple effect, which includes properties such as position, borderRadius, backgroundColor, width, height, left, top, transform, and animation. The ‘newRippleStyle’ object uses the ‘setRippleStyle1’ function to update the state variable controlling the style of the ripple effect for the first button.

4. 

```
useEffect( effect: () :void  => {
    const fetchPasses = async () :Promise<void>  => {  Show usages  southPaw
        const apiURL :string  = isHome
            ? '/api/passes?_limit=3'
            : '/api/passes'
        try {
            const res :Response  = await fetch(apiURL)
            const data = await res.json()
            setPasses(data)
        } catch (error) {
            console.log('Error loading data', error)
        } finally {
            setLoading( value: false)
        }
    }
    fetchPasses()
}, deps: [])
```

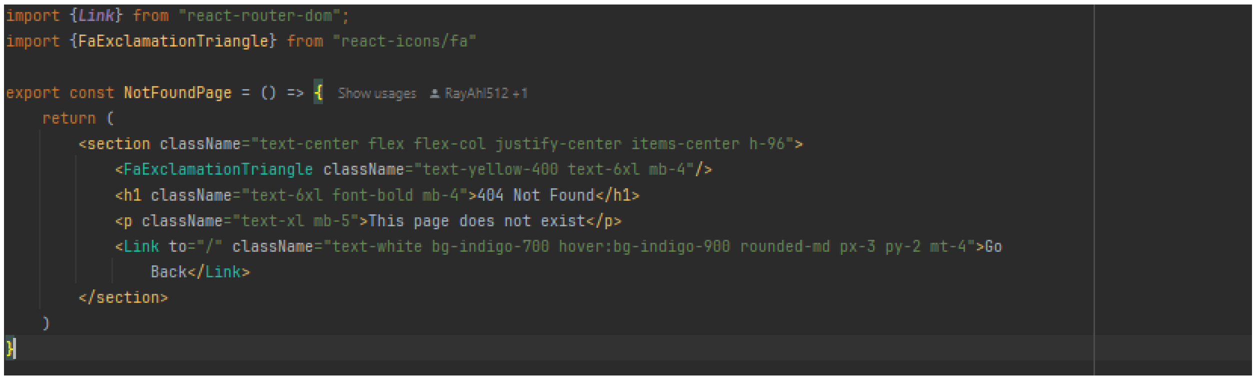
This code snippet from the PassListing file uses the 'useEffect' hook to fetch data from an API when the component mounts. It updates the component's state with the fetched data and handles any error that occur during the fetching process.

5. 

```
return (
    <div className="bg-white rounded-xl shadow-md relative">
        <div className="p-4">
            <div className="mb-6">
                <div className="text-gray-600 my-2">{title}</div>
                <h3 className="text-xl font-bold"></h3>
            </div>
            <div className="mb-5"></div>
            <h3 className="text-green-900 mb-2">{features}</h3>
            <button className={'text-green-900.mb-5 hover:text-green-900'}
                    onClick={() : void  => setShowDescription( value: (previousState : boolean ) => !previousState)}>
                Show {showDescription ? 'less' : 'more'}
            </button>
            <div className="border border-gray-100 mb-5"></div>
            <div className="flex flex-col lg:flex-row justify-between mb-4">
                <div className="text-green-600 mb-3">
                    <FaTree className={'inline text-lg mb-1 mr-1'}/>{price}
                </div>
                <a href={`/passes/${id}`}
                   className="h-[36px] bg-green-900 hover:bg-green-900 text-white px-4 py-2 rounded-lg text-center text-sm">
                    Read More
                </a>
            </div>
        </div>
    </div>
)
```

Code is a React component that renders a card-like structure with dynamic content. REpresents a reusable card component that can display various types of content, titles, features, prices, and more.

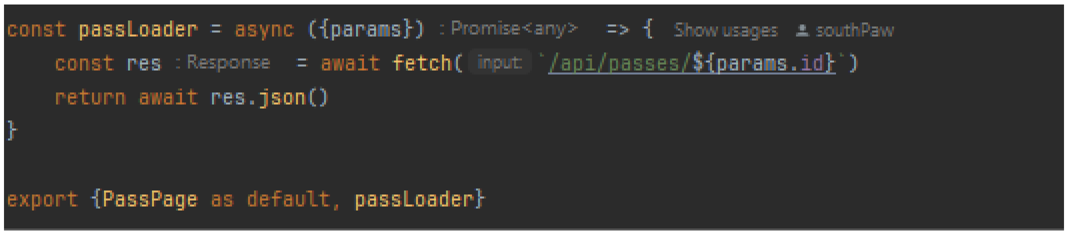
Not Found Page:

1. 

```
import {Link} from "react-router-dom";
import {FaExclamationTriangle} from "react-icons/fa"

export const NotFoundPage = () => {  Show usages  RayAhl512 +1
    return (
        <section className="text-center flex flex-col justify-center items-center h-96">
            <FaExclamationTriangle className="text-yellow-400 text-6xl mb-4"/>
            <h1 className="text-6xl font-bold mb-4">404 Not Found</h1>
            <p className="text-xl mb-5">This page does not exist</p>
            <Link to="/" className="text-white bg-indigo-700 hover:bg-indigo-900 rounded-md px-3 py-2 mt-4">Go
                Back</Link>
        </section>
    )
}
```

Code defines a React component called ‘NotFoundPage’ which represents a page displayed when a user tries to access a route that doesn’t exist within the application. There are two imports, one is ‘Link’ from ‘react-router-dom’ showing that the application uses React Router for navigation, and the other is ‘FaExclamationTriangle’ component from ‘react-cions/fa’, which renders an exclamation triangle icon.

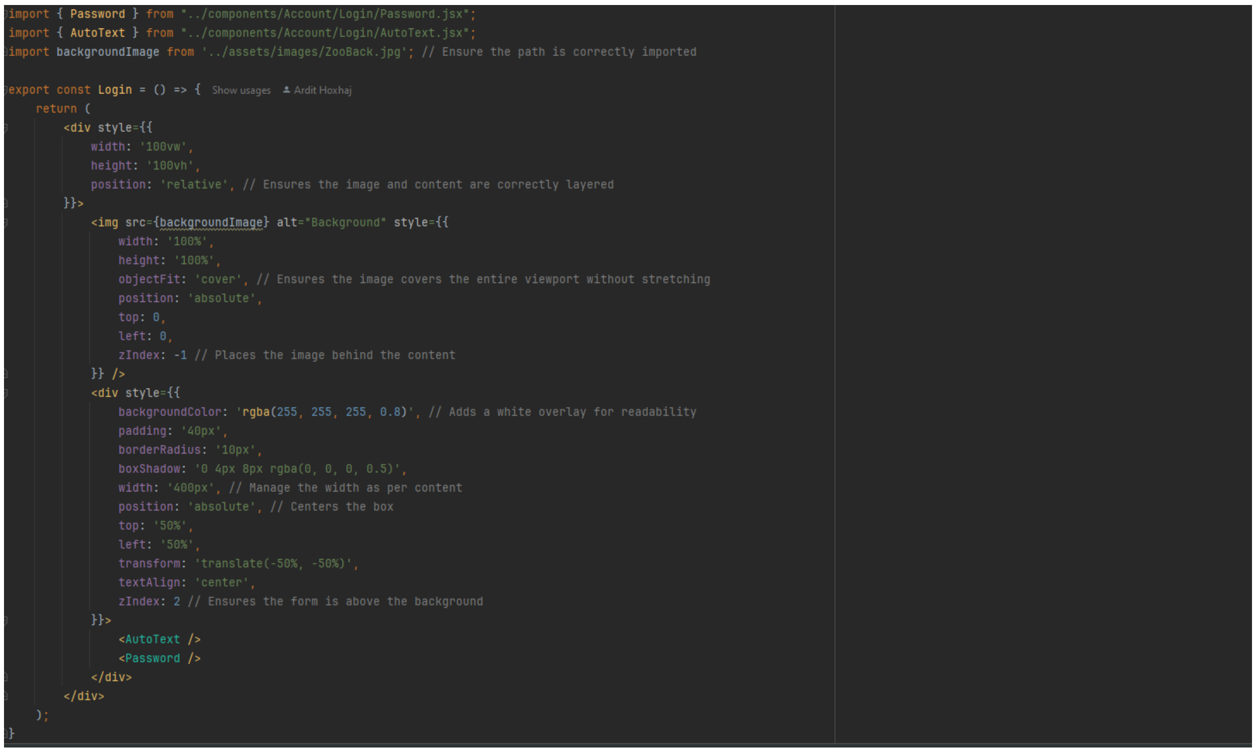
Pass Page:

1. 

```
const passLoader = async ({params}) : Promise<any>  => {  Show usages  southPaw
    const res : Response  = await fetch( input: `/api/passes/${params.id}`)
    return await res.json()
}

export {PassPage as default, passLoader}
```

Code defines an asynchronous function named ‘passloader’ which consists of a function signature that takes an object as a parameter, which has a route parameter ‘params’. The function body uses the ‘fetch’ function to make an HTTP GET request to the specific API endpoint. The URL for the endpoint is constructed using template literals where ‘/api/passes/’ is the base URL and ‘${params.id} is appended to it.

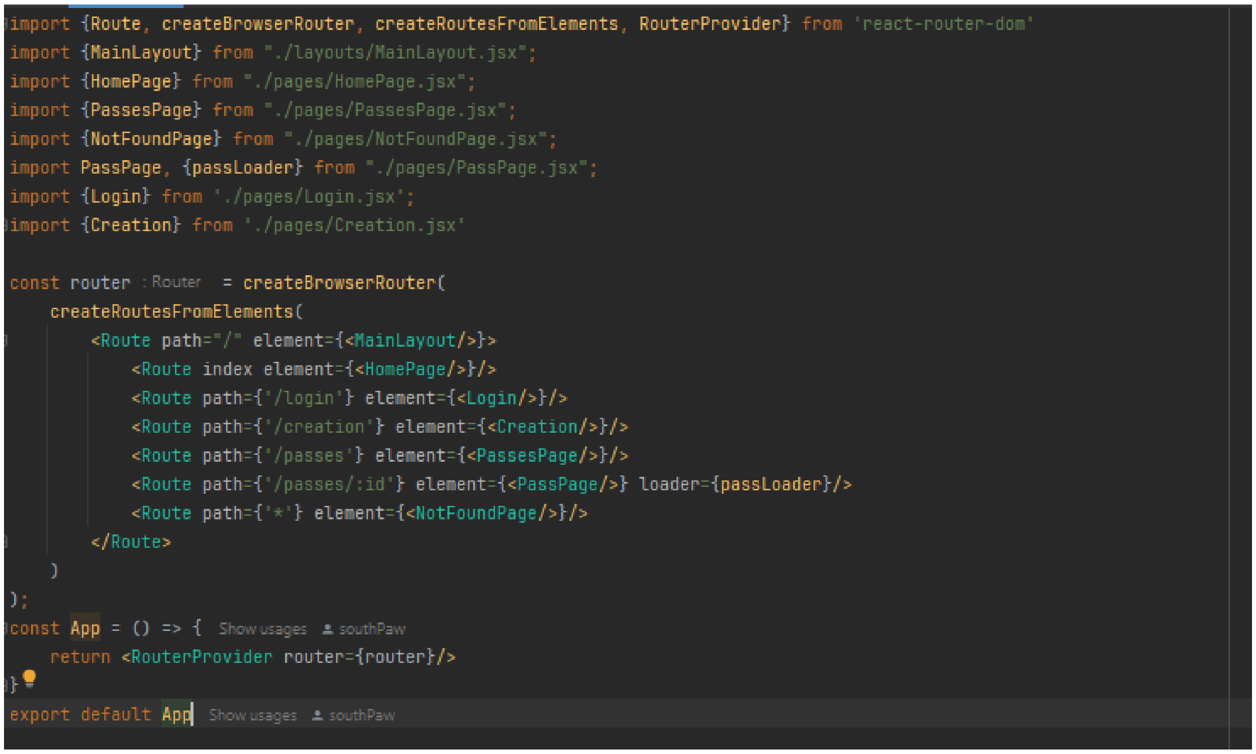
Login Page:

1. 

```
import { Password } from "../components/Account/Login/Password.jsx";
import { AutoText } from "../components/Account/Login/AutoText.jsx";
import backgroundImage from '../assets/images/ZooBack.jpg'; // Ensure the path is correctly imported

export const Login = () => {  Show usages  Ardit Hoxhaj
    return (
        <div style={{
            width: '100vw',
            height: '100vh',
            position: 'relative', // Ensures the image and content are correctly layered
        }}>
            <img src={backgroundImage} alt="Background" style={{
                width: '100%',
                height: '100%',
                objectFit: 'cover', // Ensures the image covers the entire viewport without stretching
                position: 'absolute',
                top: 0,
                left: 0,
                zIndex: -1 // Places the image behind the content
            }} />
            <div style={{
                backgroundColor: 'rgba(255, 255, 255, 0.8)', // Adds a white overlay for readability
                padding: '40px',
                borderRadius: '10px',
                boxShadow: '0 4px 8px rgba(0, 0, 0, 0.5)',
                width: '400px', // Manage the width as per content
                position: 'absolute', // Centers the box
                top: '50%',
                left: '50%',
                transform: 'translate(-50%, -50%)',
                textAlign: 'center',
                zIndex: 2 // Ensures the form is above the background
            }}>
                <AutoText />
                <Password />
            </div>
        </div>
    );
}
```

Code defines a React component called ‘Login’. There are two imports of components, one being the ‘Password’ and the other being ‘AutoText’. Within the ‘div’ element with ‘position: relative’, there are two main elements being ‘img’ and another’div’. The ‘img’ element displays a background image positioned absolutely to cover the entire area and placed behind the content using ‘zIndex: -1’. The other ‘div’ element contains the login form consisting of a white overlay with reduced opacity and positioned absolutely at the center with a specific width and rounded corners. Inline styling was used on the layout, positioning, and appearance of the background image and login form.

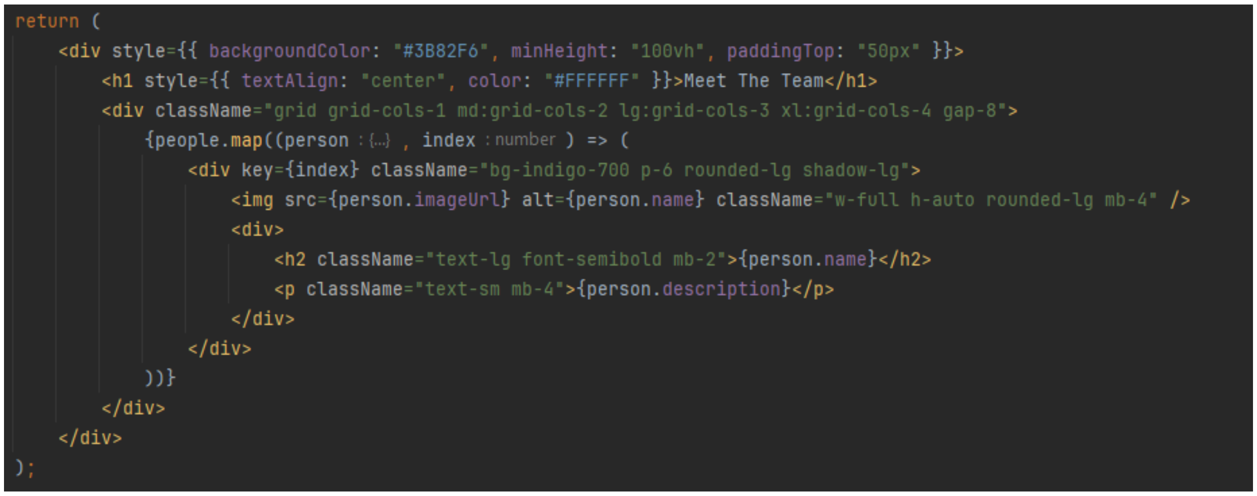
App.jsx:

1. 

```
import {Route, createBrowserRouter, createRoutesFromElements, RouterProvider} from 'react-router-dom'
import {MainLayout} from "./layouts/MainLayout.jsx";
import {HomePage} from "./pages/HomePage.jsx";
import {PassesPage} from "./pages/PassesPage.jsx";
import {NotFoundPage} from "./pages/NotFoundPage.jsx";
import PassPage, {passLoader} from "./pages/PassPage.jsx";
import {Login} from './pages/Login.jsx';
import {Creation} from './pages/Creation.jsx'

const router :Router  = createBrowserRouter(
    createRoutesFromElements(
        <Route path="/" element={<MainLayout/>}>
            <Route index element={<HomePage/>}/>
            <Route path={'/login'} element={<Login/>}/>
            <Route path={'/creation'} element={<Creation/>}/>
            <Route path={'/passes'} element={<PassesPage/>}/>
            <Route path={'/passes/:id'} element={<PassPage/>} loader={passLoader}/>
            <Route path={'*'} element={<NotFoundPage/>}/>
        </Route>
    )
);
const App = () => {  Show usages  southPaw
    return <RouterProvider router={router}/>
}

export default App  Show usages  southPaw
```

Code defines the routing for the React application using React Router with multiple imports for necessary modules and components from the ‘react-router-dom’ package, such as ‘Route’, ‘createBrowserRoute’, ‘createRoutesFromElements’, ‘RouterProvides’, as well as the various page components and layouts. The code also includes router configuration, which creates a browser router instance using ‘createBrowserRouter’.

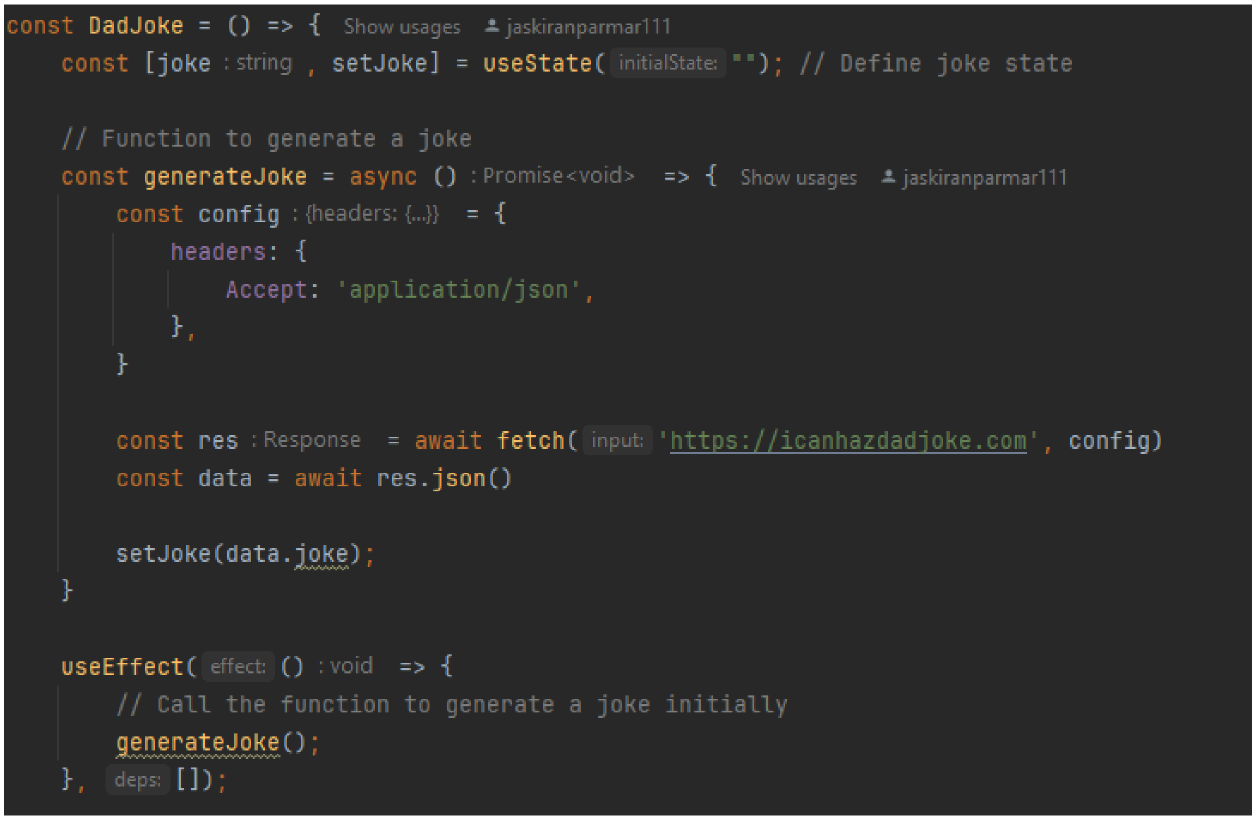
Contact Page:

1. 

```
return (
    <div style={{ backgroundColor: "#3B82F6", minHeight: "100vh", paddingTop: "50px" }}>
        <h1 style={{ textAlign: "center", color: "#FFFFFF" }}>Meet The Team</h1>
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 xl:grid-cols-4 gap-8">
            {people.map((person :{...} , index :number ) => (
                <div key={index} className="bg-indigo-700 p-6 rounded-lg shadow-lg">
                    <img src={person.imageUrl} alt={person.name} className="w-full h-auto rounded-lg mb-4" />
                    <div>
                        <h2 className="text-lg font-semibold mb-2">{person.name}</h2>
                        <p className="text-sm mb-4">{person.description}</p>
                    </div>
                </div>
            ))}
        </div>
    </div>
);
```

This code is a React component that renders a “Greetings” section with a grid layout containing the team members’ information. Generates a visually appealing section with a responsive grid layout, displaying team members’ images, names, and descriptions.

Animal Page:

1. 

```
const DadJoke = () => {  Show usages  jaskiranparmar111
    const [joke : string , setJoke] = useState( initialState: ""); // Define joke state

    // Function to generate a joke
    const generateJoke = async () : Promise<void>  => {  Show usages  jaskiranparmar111
        const config : {headers: {...}}  = {
            headers: {
                Accept: 'application/json',
            },
        }

        const res : Response  = await fetch( input: 'https://icanhazdadjoke.com', config)
        const data = await res.json()

        setJoke(data.joke);
    }

    useEffect( effect: () : void  => {
        // Call the function to generate a joke initially
        generateJoke();
    },  deps: []);
```

Code is defined as a React component that generates dad jokes. This component fetches a dad joke from an external API when it mounts and stores it in the component's state. Allowing for displays of the joke in the user interface when needed.