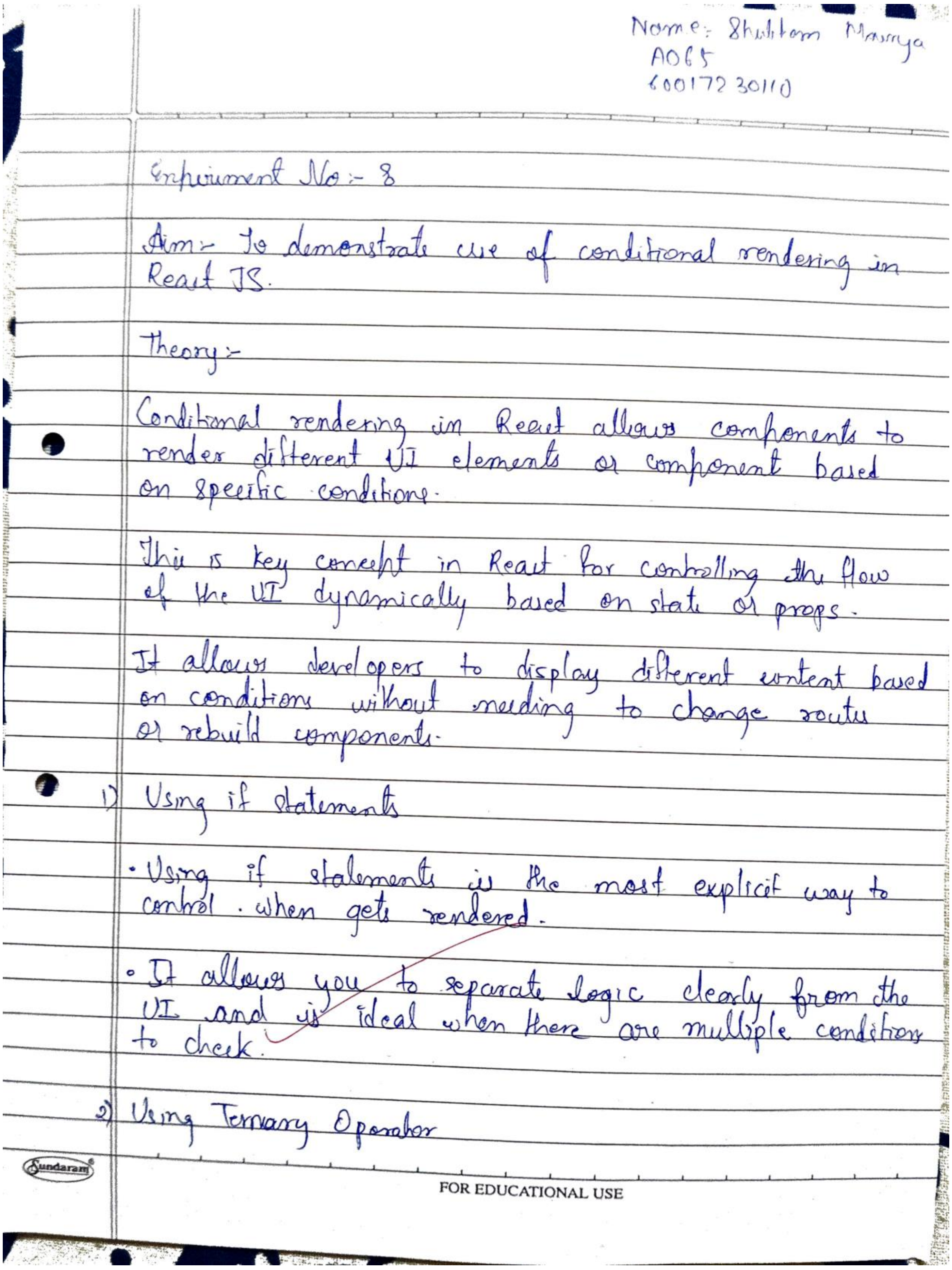
Department of Artificial Intelligence and Machine Learning
B.Tech. Sem: V Subject: Full Stack Development Laboratory (DJS22AML504)

Experiment 8

Name: Shubham Mourya **SAP ID: 60017230110**



Name:- Shubham Mourya
A065
60017230110

Experiment No :- 8

Aim:- To demonstrate use of conditional rendering in React JS.

Theory:-

Conditional rendering in React allows components to render different UI elements or component based on specific conditions.

This is key concept in React for controlling the flow of the UI dynamically based on state or props.

It allows developers to display different content based on conditions without needing to change routes or rebuild components.

1) Using if statements

• Using if statements is the most explicit way to control . when gets rendered.

• It allows you to separate logic clearly from the UI and is ideal when there are multiple conditions to check.

2) Using Ternary Operator

Sundaram

FOR EDUCATIONAL USE

- The ternary operator (condition ? true Expression : false Expression) is a more concise way to render different components based on a condition.

• It's best suited for simpler conditional renderings.

(3) Using Logical && Operator.

- The && operator is often used when you want to render a component only if a condition is true.

- This is a short hand way to render elements without an else branch.

(4) Using Switch Statements.

The switch statement is a good choice when you have multiple conditions or state that you want to render different components for.

(5) Rendering with short Circuiting

Short - circuiting takes advantage of Javascript is logical operator to render components based on a condition.

Conclusion:- These methods enable you to handle different scenarios in React component, providing a flexible way to manage UI

[illegible] 13/10

Department of Artificial Intelligence and Machine Learning
B.Tech. Sem: V Subject: Full Stack Development Laboratory (DJS22AML504)

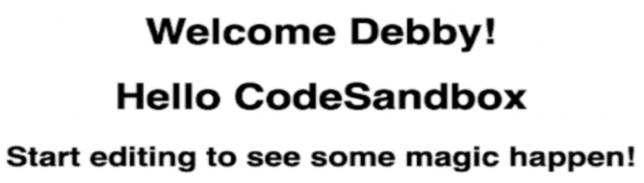
Date:	**01/10/2024 and 03/10/2024**
Aim	**Create an application to demonstrate use of Conditional rendering in React JS.**
Software	VS Code
Pre-requisite	Active internet connection

Theory	**Conditional Rendering**
	
	When developing applications, you must consider the experience of your application's users. You may want to show or deliver certain functionalities to your users based on their interaction with your application. In other words, depending on the state of your application, you may want your users to have access to specific content or functionalities built into your app. All of these are made possible by conditional rendering.
	
	In React, creating and rendering functional components in JSX is the order of the day, hence conditional rendering is the most feasible way of providing an easy user flow or customer experience based on certain events in your application.
	
	Using If-else
	Conditional rendering in React works similarly to the if-else statement in JavaScript, and each functional component returns a JSX value (which stands for JavaScript XML) that is rendered. The following example shows how to render JSX conditionally using the if-else syntax. You can accomplish this by using a variable or encapsulating the changing JSX in a wrapping function that is added to the return statement.
	
	First, you'll create two components: a HeaderComponent.js file that contains the JSX that will be rendered when the user logs in, as shown below:
	
	`export default function HeaderComponent(props) {`
	`  return (`
	`    <>`
	`      <h1> Welcome {props.username}! </h1>`
	`      <h1>Hello CodeSandbox</h1>`
	`      <h2>Start editing to see some magic happen!</h2>`
	`    </>`
	`  );`
	`}`
	
	Create a NotLoggedIn.js file to house the JSX that will be rendered when the user is not logged in, as shown below:
	
	`export default function NotLoggedIn(props) {`
	`  return <h2> No User Found </h2>;`
	`}`

Then, import the two components into the app.js file and add an if-else condition before the return statement that will assign either of the components to the {template} depending on whether the isLoggedIn state is true or false:

```
import HeaderComponent from "./HeaderComponent";
import NotLoggedIn from "./NotLoggedIn";
import React from "react";

import "./styles.css";

export default function App() {
  const [isloggedIn, setIsLoggedIn] = React.useState(false);
  let template;
  if (isloggedIn) {
    template = <HeaderComponent username="Debby" />;
  } else {
    template = <NotLoggedIn />;
  }
  return <div className="App">{template}</div>;
}
```

When a user logs in, the state is set to true, and the welcome message from the HeaderComponent is displayed to the user as follows:

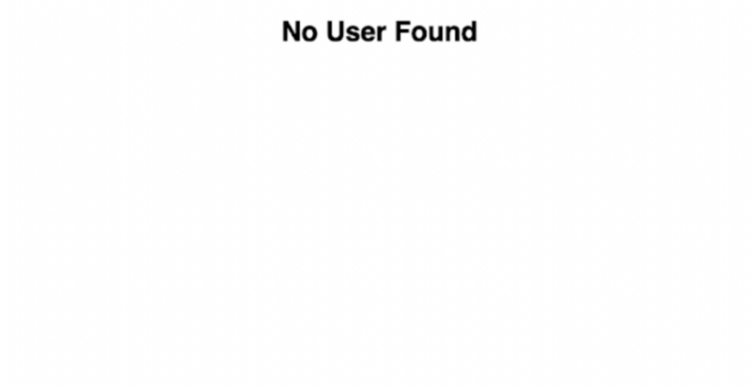


Welcome Debby!

Hello CodeSandbox

Start editing to see some magic happen!

If the user is not logged in, the state is set to false, and the following message is displayed from the NotloggedIn component:


No User Found

Rendering nothing with null
On occasion, you may want to limit the contents or pages that are rendered to your users based on their preferences, time, location, demography, and other interests in order to improve your application's user experience or personalization. In such a case, you would not want to bore or stress your users with irrelevant content.

When rendering JSX, conditional rendering in React gives you the option of not rendering a specific piece of content or anything to your users. A good example is the Paypal business site, where users are unable to access certain PayPal businesses because the template is not being rendered.

To implement such functionality in React, use 'null' as the rendered template. Using 'null' will result in nothing being rendered and will also prevent errors due to no template being returned. As an example, suppose you have a "food ordering" application that only renders a page to edit orders for specific users with permission rights, while users with none are rendered nothing.

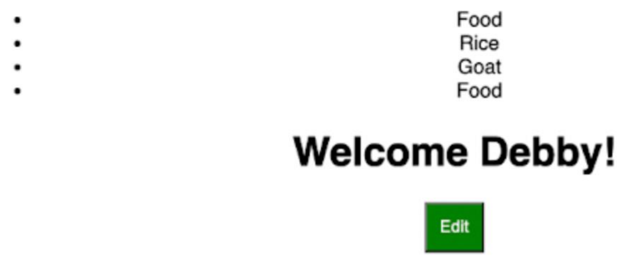
Then, create an EditComponent.js file that will contain a welcome message and a button interface for editing food orders:

```
export default function HeaderComponent(props) {
  return (
    <>
      <h1> Welcome {props.username}! </h1>
      <button style={{ padding: "10px", background: "green", color: "white"
}}>
        {" "}
        Edit{" "}
      </button>
    </>
  );
}
```

Import the EditComponent into the App.js file and add a state isHasPermission to check if the user has the permission to edit. Then, add a condition that checks if the user has editing permission and returns the EditComponent or null depending on whether the state is true or false:

```
import EditComponent from "./EditComponent";
import React from "react";

import "./styles.css";

export default function App() {
  const [isHasPermission, setIsHasPermission] = React.useState(true);
  let template;
  if (isHasPermission) {
    template = <EditComponent username="Debby" />;
  } else {
    template = null;
  }
  return (
    <div className="App">
      <div>
        <ul>
          <li> Food </li>
          <li> Rice </li>
          <li> Goat </li>
          <li> Food </li>
        </ul>
      </div>
      {template}
    </div>
  );
}
```

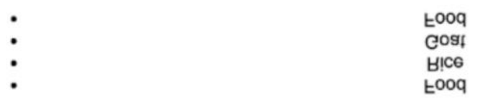
if the user has permission, the state is set to true and the EditComponent is rendered as follows:



- Food
- Rice
- Goat
- Food

Welcome Debby!

If the user has no permission, the state is set to false and null is rendered as the template:



- Food
- Goat
- Rice
- Food

Conditional rendering with switch statements

There are times when you may want to show a different UI to users based on the state of the application, such as the user's value. The JavaScript Switch statement is ideal for this functionality.

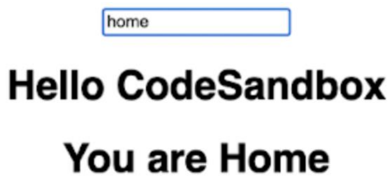
Switch statements are not part of the JSX syntax, and so they cannot be used directly within React. You can, however, use the Switch statements in a subcomponent before using the component in the main component.

In the following example, you will use Switch statements to render different content based on a specific case. You will create an input that takes the user's value and renders the components the user requests.

What is rendered to the user is determined by what the user enters as the case. Type the following code into your app.js file:

```
import "./styles.css";
import { useState } from "react";

function SwitchComponent(props) {
  switch (props.route) {
    case "home":
      return <h1> You are Home </h1>;
    case "about-us":
      return <h1> Check Us Out </h1>;
    case "learn":
      return <h1> Come and Learn the mind blowing stuffs </h1>;
    default:
      return null;
  }
}

export default function App() {
  const [path, setPath] = useState("");
  return (
    <div className="App">
      <input onChange={(e) => setPath(e.target.value)} />
      <h1>Hello CodeSandbox</h1>
      <SwitchComponent route={path} />
    </div>
  );
}
```

The component will render based on what the users enter as follows:


home

Hello CodeSandbox

You are Home

Using ternary operators

The ternary operator is synonymous with the 'if-else' operator. The only difference between the ternary operator and the 'if-else' statement is in the implementation, as JSX supports the use of ternary operators. That is, ternary operators can be easily added to the template to be rendered You can use the ternary operators to seamlessly render your components within the JSX syntax based on a specific condition.

Consider the following example, which renders different content based on the client's existence or state in the application:

```
import "./styles.css";

export default function App() {
  const isExistingClient = false;
  return (
    <div className="App">
      {isExistingClient ? (
        <>
          <h1>Hello CodeSandbox</h1>
          <h2>Start editing to see some magic happen!</h2>
        </>
      ) : (
        <>
          <h1> Hi!, New User </h1>
          <h3> Welcome to our palace </h3>
        </>
      )}
    </div>
  );
}
```

Using Logical AND (&&) and OR (||) operators (Short Circuit Evaluation)

Short-circuiting is how JavaScript handles logical expression evaluation, but the logical && and || operators work slightly differently in React. When the left-hand expression returns false, the right-hand expression is evaluated and returns true. If the left-hand expression is false, the evaluation of the second expression will be returned.

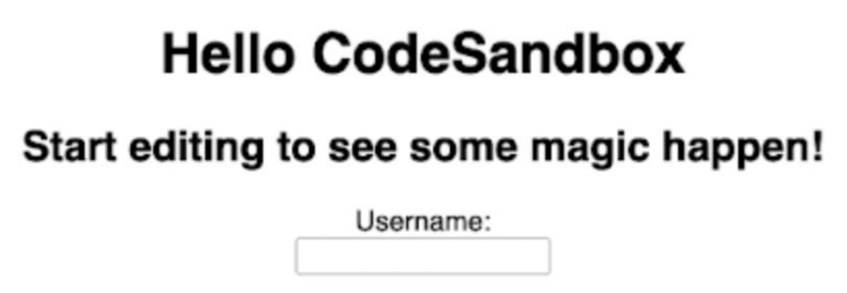
For example, suppose you have a store application; when the store is open, it should return true and render a specific message to users; when the store is closed, it should return false and render nothing. Also, when the AND (&&) operator is true or open, the right-hand-side expression is evaluated or rendered; if it is not true, it is rendered null.

```
import "./styles.css";

export default function App() {
  const isOpen = true;
  const isAvailable = false;
  return (
    <div className="App">
      {isOpen && (
        <>
          <h1>Hello CodeSandbox</h1>
      <h2>Start editing to see some magic happen!</h2>
     </>
    )}

    {isAvailable || (
     <>
      <h2> Sorry I am not available </h2>
      <button> Make Available </button>
     </>
    )}
   </div>
 );
}
```

Using IIFEs (Immediately Invoked Function Expressions)
IIFEs are self-invoking functions (functions that call themselves immediately after they have been created). They allow you to use your if...else and switch statements within the JSX you are returning. This opens up the possibility of using the previously mentioned switch or if-else method in the JSX.

In the following example, you have a state called isLoggedIn, and the content is rendered based on whether the user is logged in or not. In addition, based on the isLoggedIn state, an input field is rendered or a welcome user message is displayed.

```
import "./styles.css";

export default function App() {
  const isLoggedIn = false;
  const user = "Debby";

  return (
    <div className="App">
      <h1>Hello CodeSandbox</h1>
      <h2>Start editing to see some magic happen!</h2>

      {(() => {
        if (isLoggedIn) {
          return <h1> Welcome {user} </h1>;
        } else {
          return (
            <>
              <label style={{ textAlign: "left !important" }}>Username: </label>
              <br />
              <input />
            </>
          );
        }
      })()}
    </div>
  );
}
```



Hello CodeSandbox

Start editing to see some magic happen!

Username:

Code

Edit.jsx

```
import React from 'react';

const Edit = ({ isLogin, setIsLogin }) => {
    return (
        <div>
            <button onClick={() => setIsLogin(!isLogin)}>
                {isLogin ? 'Logout' : 'Login'}
            </button>
        </div>
    );
}

export default Edit;
```

Header.jsx

```
import React from 'react';

const Header = ({ username }) => {
    return (
        <div>
            <h1>Hello {username} bhai</h1>
        </div>
    );
};

export default Header;
```

Login.jsx

```
import React from 'react';

const Login = () => {
    return (
        <div>You are Logged In</div>
    );
};

export default Login;
```

NotLoggedIn.jsx

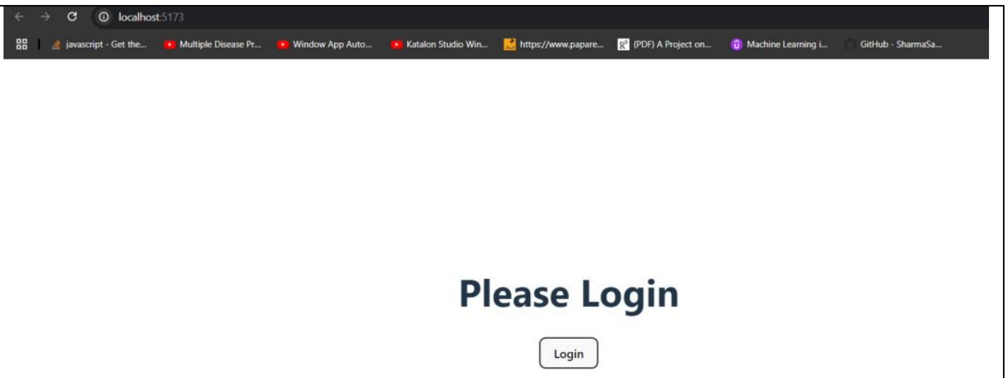
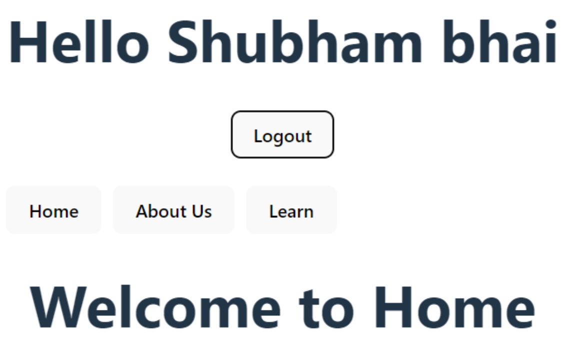
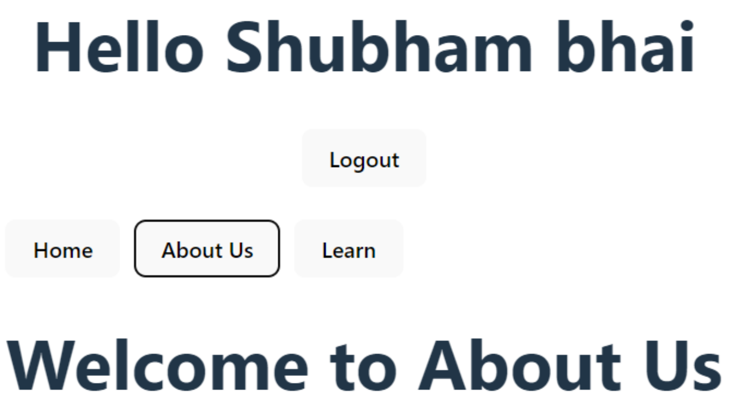
```
import React from 'react';

const NotLoggedIn = () => {
    return (
        <div>
            <h1>Please Login</h1>
        </div>
    );
};

export default NotLoggedIn;
```

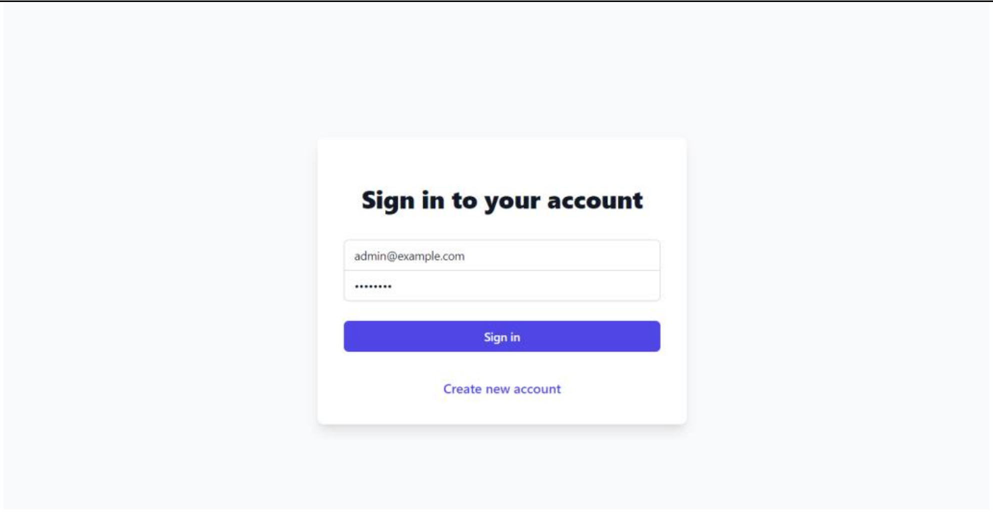
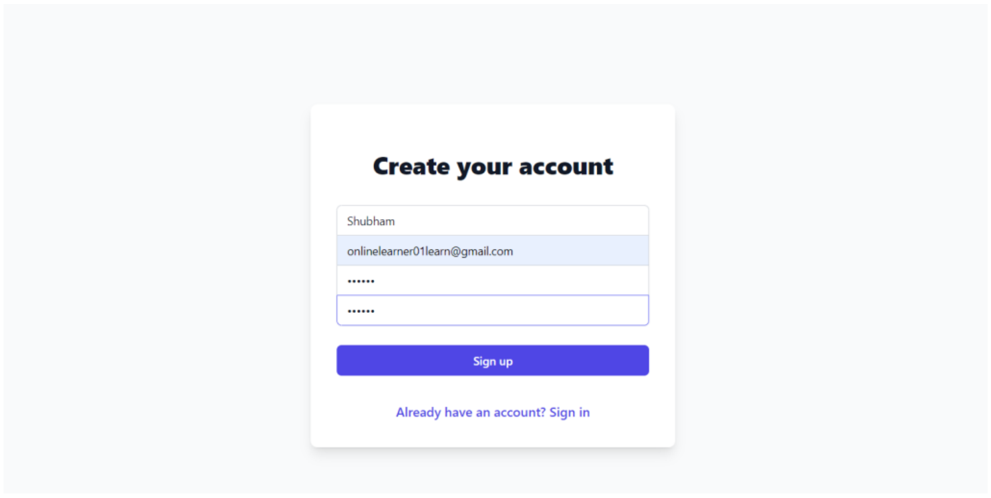
Switch.jsx

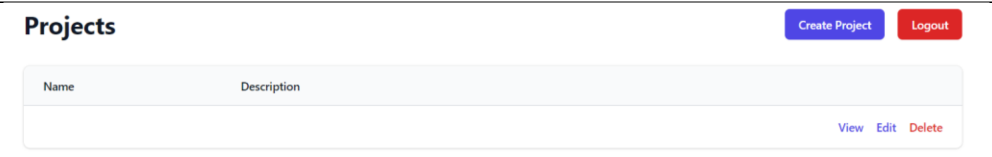
```
import React from 'react';

const SwitchComponent = ({ currentView }) => {
    let content;

    switch (currentView) {
        case 'home':
            content = <h1>Welcome to Home</h1>;
            break;
        case 'about':
            content = <h1>Welcome to About Us</h1>;
            break;
        case 'learn':
            content = <h1>Welcome to Learn</h1>;
            break;
        default:
            content = <h1>Welcome!</h1>;
    }

    return <div>{content}</div>;
};

export default SwitchComponent;
```

App.jsx

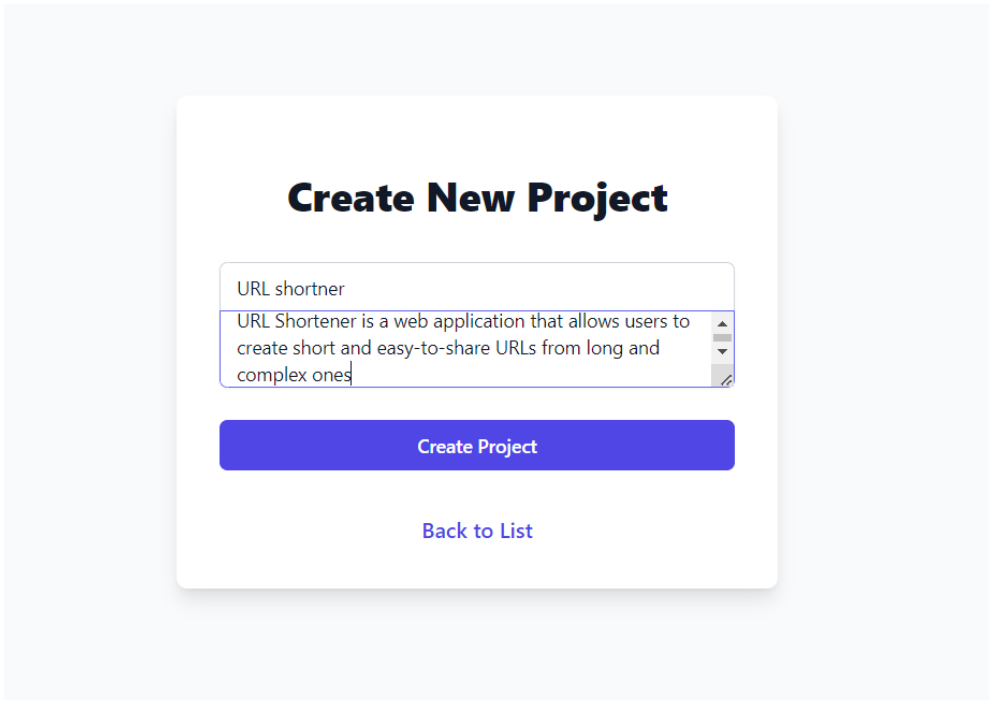
```
import { useState } from 'react';
import './App.css';
import Header from './components/Header';
import NotLoggedIn from './components/NotLoggedIn';
import Edit from './components/Edit';
import SwitchComponent from './components/Switch';

function App() {
  const [isLogin, setIsLogin] = useState(false);
  const [currentView, setCurrentView] = useState('home');

  return (
    <>
      {isLogin ? <Header username="Shubham" /> : <NotLoggedIn />}
      <Edit isLogin={isLogin} setIsLogin={setIsLogin} />
      <div>
        <br />
      </div>
      {isLogin ? (
        <>
          <div style={{ display: 'flex', gap: '10px' }}>
                                  <button onClick={() =>
setCurrentView('home')}>Home</button>
                <button onClick={() => setCurrentView('about')}>About
Us</button>
                                  <button onClick={() =>
setCurrentView('learn')}>Learn</button>
          </div>
          <SwitchComponent currentView={currentView} />
        </>
      ) : null}
    </>
  );
}

export default App;
```

Result	 localhost:5173 javascript - Get the... Multiple Disease Pr... Window App Auto... Katalon Studio Win... https://www.papare... (PDF) A Project on... Machine Learning i... GitHub - SharmaSa... Please Login Login  Hello Shubham bhai Logout Home About Us Learn Welcome to Home  Hello Shubham bhai Logout Home About Us Learn Welcome to About Us

Code	 Sign in to your account admin@example.com •••••••• Sign in Create new account  Create your account Shubham onlinelearner01learn@gmail.com •••••• •••••• Sign up Already have an account? Sign in

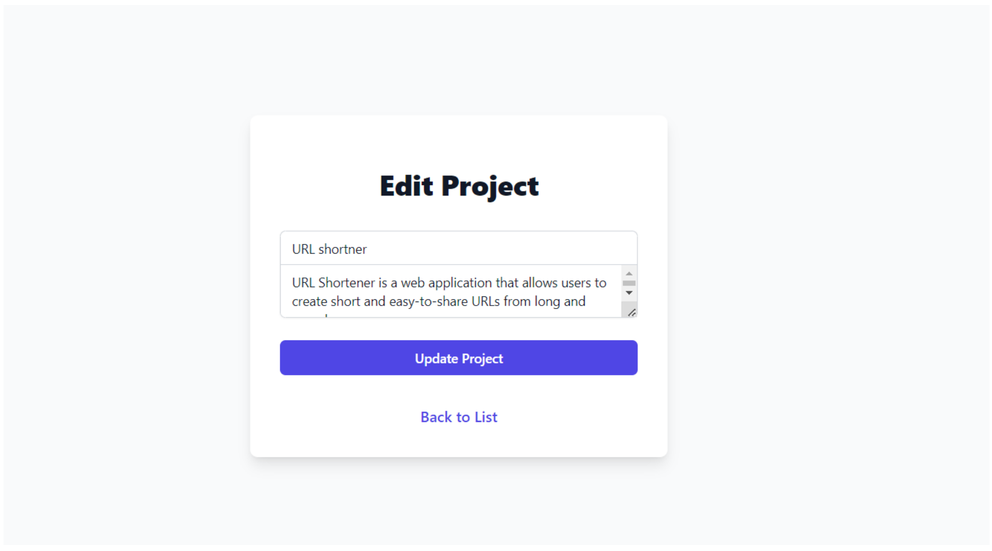

Projects

Name	Description	
		View Edit Delete


Create New Project

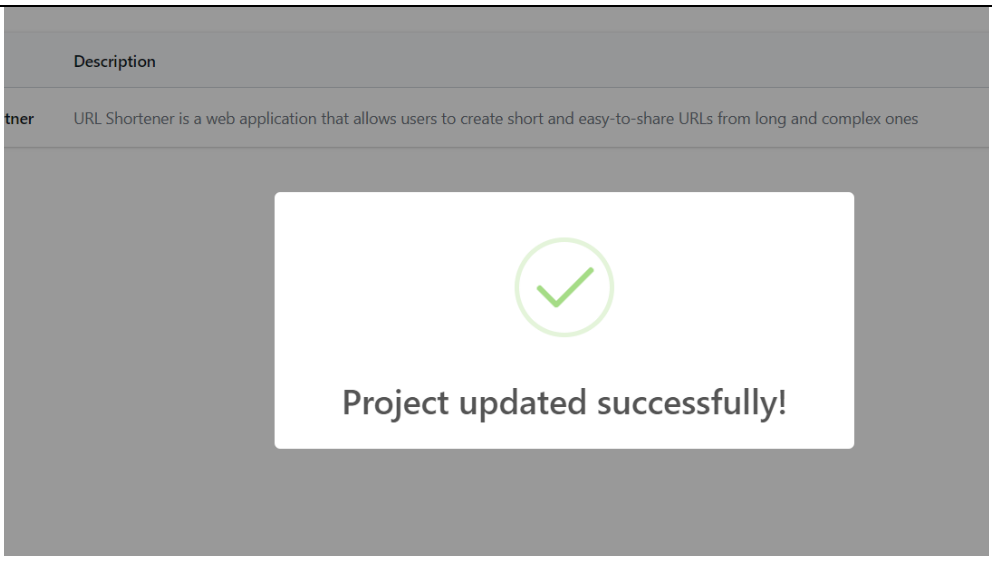
URL shortner

URL Shortener is a web application that allows users to create short and easy-to-share URLs from long and complex ones


Edit Project

URL shortner

URL Shortener is a web application that allows users to create short and easy-to-share URLs from long and

	 Description tner URL Shortener is a web application that allows users to create short and easy-to-share URLs from long and complex ones Project updated successfully!
Conclusion	Conditional Rendering is in React, as well as several methods of Conditional Rendering in React, such as using if...else, rendering nothing with null, conditional rendering with switch statements, using ternary operators, using logical AND (&&) and OR (||) operators (Short Circuit Evaluation), and Using IIFEs. We also learned about some conditional rendering use cases.