React.js Component Creation Assignment

N V Sidhartha

20AK1A0174

AITS – TPT

Ph no: -7207105547

APP.JS Code: -

```
import React from "react";
import Card from "./Card";
import Button from "./Button";
import ImageDisplay from "./ImageDisplay";

const App = () => {
  const handleButtonClick = () => {
    alert("React.js Component Creation Assignment");
  };

  const imageUrl =
    "https://www.itl.cat/pngfile/big/311-3110208_civil-engineering-
work-hd.jpg";

  return (
    <div>
      <Card
        title="N V Sidhartha"
        content="B.Tech - Civil Engineering Student"
        color="#ff6f61"
      />
      <Button label="Click Me" onClick={handleButtonClick}
color="#66a7ff" />

      <div>
        <h1>Open Image</h1>
        <ImageDisplay imageUrl={imageUrl} />
      </div>
    </div>
  );
};

export default App;
```

Card.JS Code: -

```
// Card.js
import React from "react";
import "./Card.css";

const Card = ({ title, content, color }) => {
  return (
    <div className="card" style={{ backgroundColor: color }}>
      <h2>{title}</h2>
      <p>{content}</p>
    </div>
  );
};

export default Card;
```

Card.CSS Code: -

```
.card {
  border-radius: 10px;
  padding: 20px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  color: white;
}
```

Button.JS Code: -

```jsx
import React from "react";
import "./Button.css";

const Button = ({ label, onClick, color }) => {
  return (
    <button
      className="button"
      style={{ backgroundColor: color }}
      onClick={onClick}
    >
      {label}
    </button>
  );
};

export default Button;
```

Button.CSS Code: -

```css
/* Button.css */
.button {
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  color: white;
  font-weight: bold;
  cursor: pointer;
  transition: background-color 0.3s;
}

.button:hover {
  background-color: #555;
}
```

Image display.JS Code: -

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

const ImageDisplay = ({ imageUrl }) => {
  const [showImage, setShowImage] = useState(false);

  const toggleImage = () => {
    setShowImage(!showImage);
  };

  return (
    <div>
      <button onClick={toggleImage}>Toggle Image</button>
      {showImage && (
        <div>
          <img src={imageUrl} alt="Image" />
          <button onClick={toggleImage}>Close Image</button>
        </div>
      )}
    </div>
  );
};

export default ImageDisplay;
```

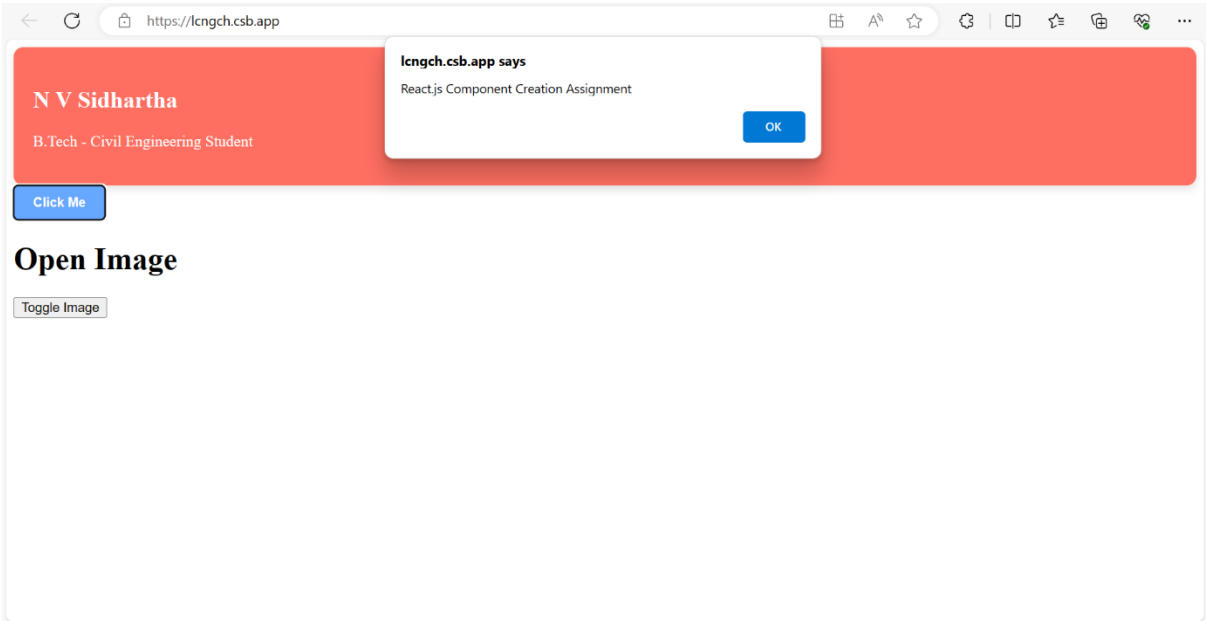
Style.CSS: -

```
.App {
  font-family: sans-serif;
  text-align: center;
}
```

Output: -

Open Image

Toggle Image



Assignment description: -

The way I approach to create a card and button component

1. Create a Card Component:

- Create a new file for your Card component, for example, Card.js.
- Define a functional component named Card in the file.
- Inside the Card component, structure the JSX for the card layout, including elements like title, description, image, etc.
- Use props to make the card component dynamic and reusable. You can pass props like title, description, image, etc., to customize the content of the card.
- Export the Card component at the end of the file.

2. Create a Button Component:

- Similarly, create a new file for your Button component, for example, Button.js.
- Define a functional component named Button in the file.
- Structure the JSX for the button element, including text and any other necessary attributes.
- Use props to customize the button component, such as onClick function, button text, styles, etc.
- Export the Button component at the end of the file.

3. Implementing the Components:

- Import the Card and Button components in your main application file.
- Use these components within your application by passing the required props.
- Customize the appearance and functionality of the components as needed.

By following these steps, you can create reusable and modular card and button components in React. This approach helps in maintaining a clean and organized code structure while building your React application.

When creating card and button components in React, you may encounter some challenges. Here are a few common challenges and how you can address them:

1. Styling and CSS: One challenge is ensuring consistent styling across different components and handling CSS conflicts. To address this, you can use CSS-in-JS libraries like styled-components or CSS modules to encapsulate styles and prevent conflicts.

2. Responsiveness: Making sure that your card and button components look good on various screen sizes can be a challenge. You can address this by using CSS media queries to adjust styles based on screen width or by using responsive design frameworks like Bootstrap.

3. Accessibility: Ensuring that your components are accessible to all users, including those using screen readers or keyboard navigation, can be a challenge. You can address this by adding appropriate ARIA attributes, focus management, and keyboard navigation support to your components.

4. State Management: Managing the state of your components, especially for interactive buttons, can be challenging. You can address this by using React's built-in state management or external libraries like Redux for more complex state management needs.

5. Reusability: Making your card and button components reusable and flexible for different use cases can be a challenge. You can address this by using props to customize the appearance and behavior of your components and by designing them with a clear API for easy integration into different parts of your application.