Name: Gandham Bhanu Sri

Roll Number: 20NN1A05D5

Vignan's Nirula Institute of Technology and Science for Women

ASSIGNMENT-2

Aim: To create reusable components like card components and buttons using React.js

- Here we used two components named “Button.jsx” and “Card.jsx” and integrated them in “page.js”

Code in **Button.jsx**

```
import React from 'react';
import PropTypes from 'prop-types';
import styled from 'styled-components';

const ButtonElement = styled.button`
  background-color: ${(props) => props.backgroundColor || 'blue'};
  color: ${(props) => props.color || '#fff'};
  border: none;
  border-radius: 4px;
  padding: 8px 16px;
  cursor: pointer;
  transition: background-color 0.3s;

  &:hover {
    background-color: ${(props) => props.hoverBackgroundColor || 'blue'};
  }
`;

const Button = ({ onClick, text, backgroundColor, color, hoverBackgroundColor
}) => {
  return (
    <ButtonElement
      onClick={onClick}
      backgroundColor={backgroundColor}
      color={color}
      hoverBackgroundColor={hoverBackgroundColor}
    >
      {text}
    </ButtonElement>
  );
};
```

```
Button.propTypes = {
  onClick: PropTypes.func.isRequired,
  text: PropTypes.string.isRequired,
  backgroundColor: PropTypes.string,
  color: PropTypes.string,
  hoverBackgroundColor: PropTypes.string,
};

export default Button;
```

Code in **Card.jsx**

```
import React from 'react';
import PropTypes from 'prop-types';
import styled from 'styled-components';

const CardContainer = styled.div`
  border: 1px solid #ccc;
  border-radius: 8px;
  padding: 16px;
  margin: 16px;
  width: 300px;
`;

const CardTitle = styled.h2`
  margin-bottom: 8px;
`;

const CardDescription = styled.p`
  color: #666;
`;

const CardImage = styled.img`
  width: 100%;
  height: auto;
  border-radius: 8px;
`;

const Card = ({ title, description, imageUrl }) => {
  return (
    <CardContainer>
      <CardTitle>{title}</CardTitle>
      {imageUrl && <CardImage src={imageUrl} alt={title} />}
      <CardDescription>{description}</CardDescription>
    </CardContainer>
  );
```

```
};

Card.propTypes = {
  title: PropTypes.string.isRequired,
  description: PropTypes.string.isRequired,
  imageUrl: PropTypes.string,
};

export default Card;
```

Code in **page.js**

```
"use client"
import React from 'react';
import Card from './components/Card';
import Button from './components/Button';

const App = () => {
  return (
    <div>
      <h1>Sample React Application</h1>
      <div style={{ display: 'flex', justifyContent: 'center' }}>
        <Card
          title="Example Card 1"
          description="This is a sample card component with a button."
          imageUrl="https://cdni.autocarindia.com/Utils/ImageResizer.ashx?n=ht
tps://cms.haymarketindia.net/model/uploads/modelimages/Mercedes-Benz-GLC-
110820231043.jpg"
        >
          <Button text="Click me" onClick={() => alert('Button clicked!')} />
        </Card>
        <Card
          title="Example Card 2"
          description="This is another sample card component with a button."
          imageUrl="https://imgd-ct.aeplcdn.com/664x415/n/cw/ec/129907/q3-
right-front-three-quarter.jpeg"
        >
          <Button
            text="Click me too"
            onClick={() => alert('Another button clicked!')}
            backgroundColor="#28a745"
            hoverBackgroundColor="#218838"
          />
        </Card>
      </div>
    </div>
  );
```

```
};

export default App;
```

This is the code where we integrated the card and button components into page.jsx.

Sample output:

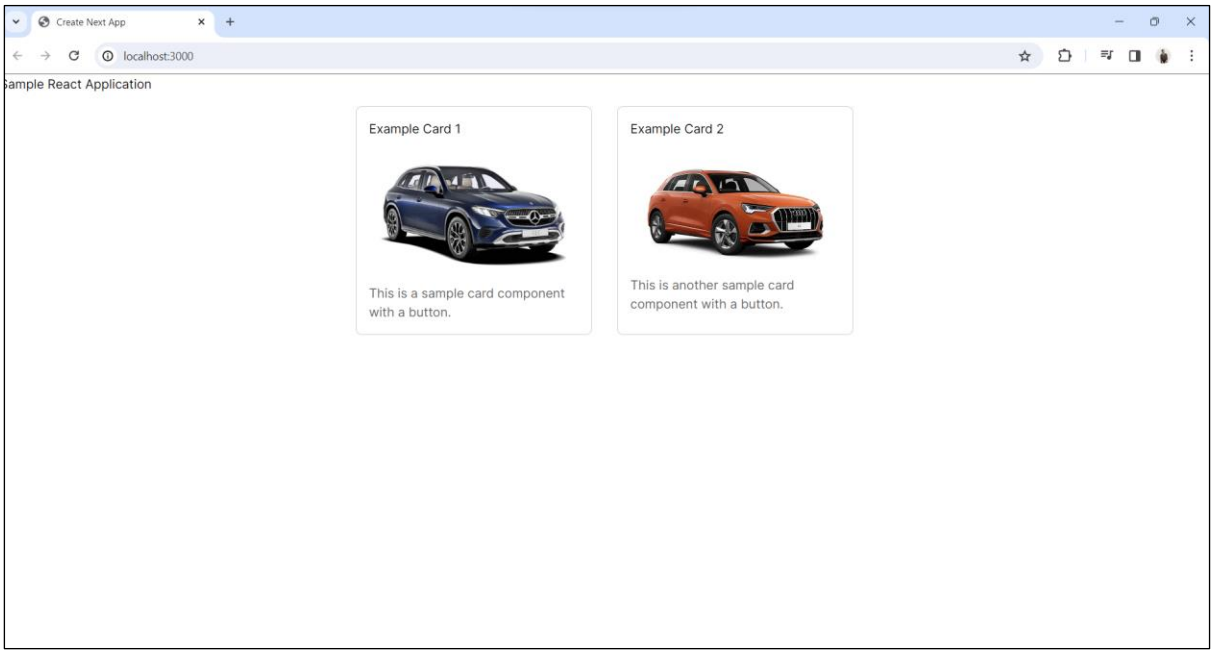

Fig: Output of the integration of Card and Button components

Component Reusability:

The card and button components are designed to be highly reusable and customizable. Both components accept props to pass data and event handlers, allowing for easy customization with different content, styles, and functionality. This ensures that the components can be efficiently utilized across various parts of the application without the need for extensive modifications.

Component Structure:

The component structure is well-organized and intuitive, adhering to React.js best practices. Each component is encapsulated within its own file, promoting modularity and separation of concerns. The code is structured in a clear and concise manner, making it easy to understand and navigate. This organized structure enhances maintainability and facilitates collaboration among developers.

Styling:

The components are styled appropriately using the styled-components library, ensuring a visually appealing appearance. The styling is consistent, with defined styles for various elements such as borders, backgrounds, text colors, and hover effects. The use of styled-

components also promotes encapsulation of styles within components, preventing style conflicts and enhancing code maintainability.

npm install prop-types

The PropTypes package provides a mechanism for type-checking React component props, ensuring that the correct data types are passed to components. It helps catch potential errors and improves code reliability by defining the expected props' shapes and types.

npm install styled-components

The styled-components is a popular CSS-in-JS library for styling React components. It allows developers to write CSS directly within their JavaScript files, using tagged template literals. This approach enables component-level styling, scoped styles, and dynamic styling based on props, leading to more maintainable and encapsulated code.

Both packages are essential tools in modern React development, enhancing code quality, readability, and maintainability. They streamline the development process by providing standardized solutions for common challenges such as prop validation and component styling. By using PropTypes, we can ensure that their components receive the correct data types, reducing the likelihood of runtime errors and improving code robustness. styled-components, on the other hand, empowers developers to create visually appealing and customizable UI components, leveraging the flexibility and power of JavaScript for styling.

Functionality:

The components function as expected, fulfilling their respective purposes effectively. The button component is clickable and provides visual feedback when hovered over or clicked, enhancing user interaction. Additionally, the card component displays data such as title, description, and an optional image, providing a clear and informative representation of content. Both components handle click events and data rendering seamlessly, contributing to a smooth user experience.

Conclusion:

In conclusion, this assignment has provided practical experience in creating reusable and customizable components using React.js. By implementing the card and button components, I had gained a deeper understanding of React.js fundamentals and how to apply them to real-world projects. This exercise has reinforced concepts such as component reusability, structure organization, styling techniques, and functionality implementation, empowering developers to build robust and user-friendly interfaces in React.js applications.