Name : Mayuri Yerande
Div : D15B
Roll No.: 70

Experiment No. 9

Aim: Experiment to study the Advanced React.

Program:

Build an online shopping Website. Make use of the router, at least one Hook,ref, forms etc.

Theory:

Router:

React Router is a standard library for routing in React. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.

Let us create a simple application to React to understand how the React Router works. The application will contain three components: home component, about a component, and contact component. We will use React Router to navigate between these components.

Hooks:

Hooks are the new feature introduced in the React 16.8 version. It allows you to use state and other React features without writing a class. Hooks are the functions which "hook into" React state and lifecycle features from function components. It does not work inside classes.

Hooks are backward-compatible, which means it does not contain any breaking changes. Also, it does not replace your knowledge of React concepts.

Ref:

Creating Refs

Refs are created using React.createRef() and attached to React elements via the ref attribute. Refs are commonly assigned to an instance property when a component is constructed so they can be referenced throughout the component.

There are a few good use cases for refs:

- Managing focus, text selection, or media playback.

- Triggering imperative animations.
- Integrating with third-party DOM libraries.

Avoid using refs for anything that can be done declaratively.

Forms:

This form has the default HTML form behavior of browsing to a new page when the user submits the form. If you want this behavior in React, it just works. But in most cases, it's convenient to have a JavaScript function that handles the submission of the form and has access to the data that the user entered into the form. The standard way to achieve this is with a technique called "controlled components".

Controlled Components

In HTML, form elements such as <input>, <textarea>, and <select> typically maintain their own state and update it based on user input. In React, mutable state is typically kept in the state property of components, and only updated with setState().

Code:

Router:

```
const App = () => {
 const device = useMobileDetect();


 return (
   <Router>
     <BasketContextProvider>
       <div className={clsx(device.type === "mobile" &&
styles.paddingForMobile, styles.container)}>
         <Header />
         <main className={styles.main}>
           <Switch>
             <Route path="/" exact>
               <Home />
             </Route>
             <Route path="/product/:slug">
               <Detail />
             </Route>
             <Route path="/category/:slug">
               <Category />
             </Route>
           </Switch>
```

```
          </main>
          <Footer />
        </div>
        <BasketSidebar />
        {device.type === "mobile" && <MobileBottomNav />}
      </BasketContextProvider>
    </Router>
  );
};


export default App;
```

Home Component:

```
const Home = () => {
 const result =
useMakeRequest("https://fakestoreapi.com/products/");


 if (!result.data) {
   if (result.error) {
     return (
       <div style={{ width: "100%", display: "flex",
justifyContent: "center", marginTop: "30px" }}>
         <Title txt={result.error} size={25}
transform="uppercase" />
       </div>
     );
   } else {
     return (
       <div style={{ width: "100%", display: "flex",
justifyContent: "center", marginTop: "30px" }}>
         <Title txt="Loading..." size={25} transform="uppercase"
/>
       </div>
     );
   }
 } else {
   return (
     <section className={styles.home}>
       <div className={styles.container}>
```

```
          <div className={styles.row}>
            {result.data && (
              <div className={styles.title}>
                <Title txt="all products" color="#171717"
size={22} transform="uppercase" />
              </div>
            )}
          </div>
          <div className={styles.row}>
            {result.data ? (
              result.data.map((product, key) => <Card
product={product} key={key} />)
            ) : (
              <div style={{ width: "100%", display: "flex",
justifyContent: "center" }}>
                <Title txt={result.error} size={25}
transform="uppercase" />
              </div>
            )}
          </div>
        </div>
      </section>
    );
  }
};

export default Home;
```

Details component:

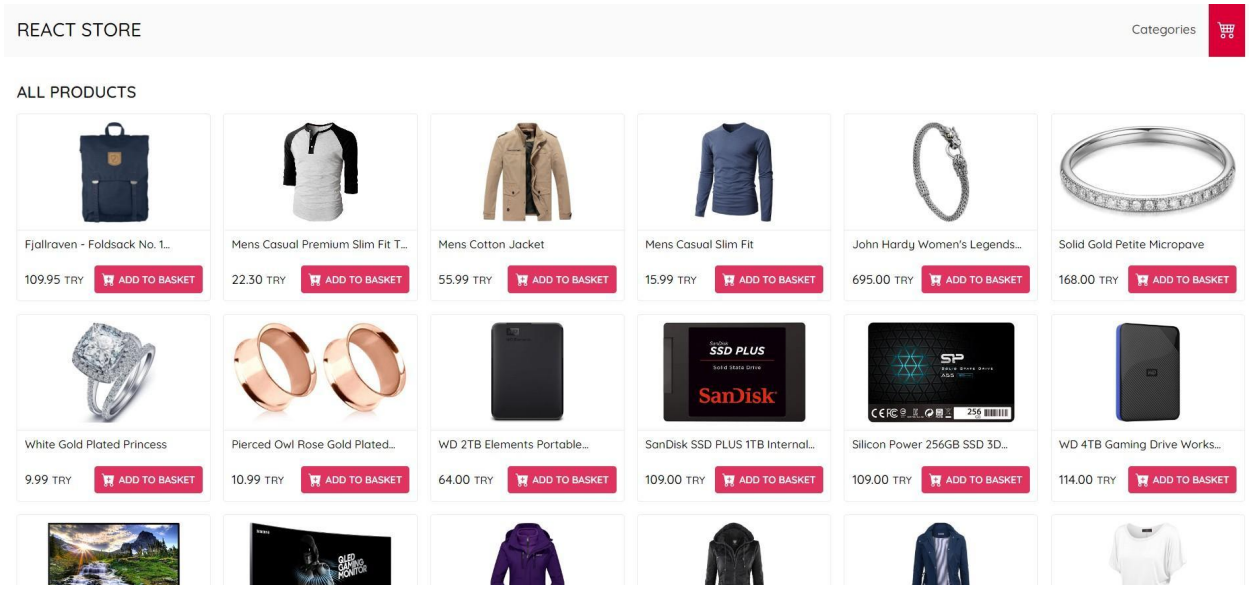
```
const Detail = () => {
  const { slug } = useParams();
  let id = slug.split("-");
  id = id[id.length - 1];

  const result =
useMakeRequest(`https://fakestoreapi.com/products/${id}`);
  const { basketItems } = useContext(BasketContext);

  const setStars = (rate) => {
    let elements = [];
    let controlNumber = 0;
```

```
    for (let i = 1; i <= 5; i++) {
      if (i <= parseInt(rate)) {
        controlNumber = parseInt(rate) - i;
        elements.push(<GetIcon icon="BsFillStarFill"
color="#F0A500" size={20} key={i} />);
      } else if (controlNumber === 0) {
        controlNumber = 1;
        elements.push(<GetIcon icon="BsStarHalf" color="#F0A500"
size={20} key={i} />);
      } else {
        elements.push(<GetIcon icon="BsStar" color="#F0A500"
size={20} key={i} />);
      }
    }

    return elements;
  };

  const getItemFromBasket = (data) => {
    let filter = basketItems.length > 0 &&
basketItems.filter((item) => item.id === data.id)[0];
    if (filter) {
      return filter;
    } else {
      return data;
    }
  };

  return (
    <section className={styles.detail}>
      {!result.data ? (
        <div style={{ width: "100%", display: "flex",
justifyContent: "center" }}>
          <Title txt="Loading..." size={25}
transform="uppercase" />
        </div>
      ) : (
        <div className={styles.content}>
          <div className={styles.top}>
            <div className={styles.img}>
              <img src={result.data.image} alt="" />
```

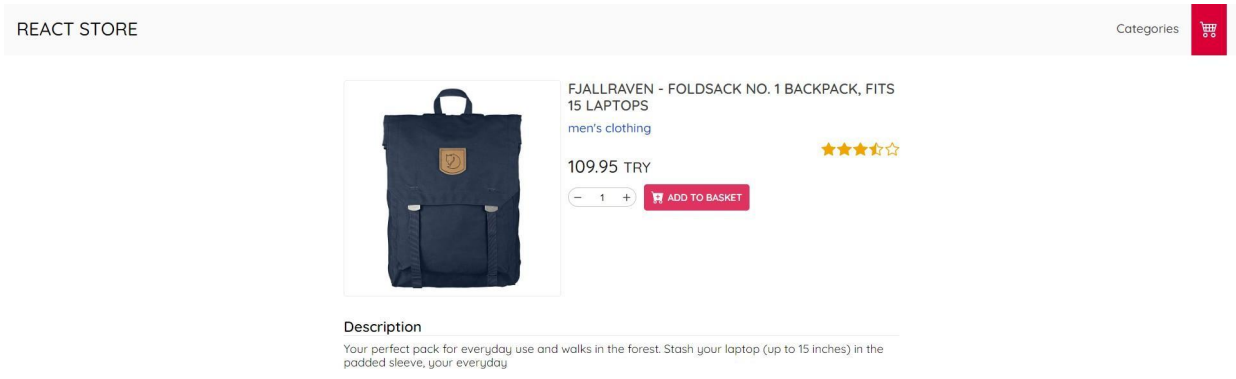
```
            </div>
            <div className={styles.info}>
              <div className={styles.title}>
                <Title txt={result.data.title}
transform="uppercase" size={20} />
              </div>
              <div className={styles.category}>
                <Link to={`/category/${result.data.category}`}
style={{ color: "#0E3EDA" }}>
                  {result.data.category}
                </Link>
              </div>
              <div className={styles.rating}>
                <div
className={styles.stars}>{setStars(result.data.rating.rate)}</di
v>
              </div>
              <div className={styles.price}>
                <p>
                  {result.data.price.toFixed(2)}
<small>TRY</small>
                </p>
              </div>
              <div className={styles.addToBasketAndQuantity}>
                <div className={styles.quantityBox}>
                  <Quantity
data={getItemFromBasket(result.data)} />
                </div>
                <AddToBasketBtn data={result.data} />
              </div>
            </div>
          </div>
          <div className={styles.bottom}>
            <Title txt="Description" size={20}
transform="capitalize" />
            <p
className={styles.desc}>{result.data.description}</p>
          </div>
        </div>
      )}
    </section>
```

```
  );
};

export default Detail;
```

Output:
Home page:



Details:



Conclusion: An online shopping website using react has been successfully created.