Lab 2: Create a Product Management App with Functional Components

1. Setup the Project:

1. **Open your terminal:**
   - This will be the command line interface where you can run commands to set up and manage your project.
2. **Create a new React project:**
   - Use `create-react-app` to set up a new React project. This command creates a new directory called `product-management-app` with all the necessary files and configurations.
   - `npx`: A package runner tool that comes with Node.js.
   - `create-react-app`: A command-line tool that sets up a new React project with a standard structure and configuration.

   ```
   npx create-react-app product-management-app
   ```
3. **Navigate to the project directory:**
   - Change your working directory to the newly created project directory.

   ```
   cd product-management-app
   ```

2. Create a Product List:

1. **Create a new file for product data:**
   - In the `src` directory, create a new file named `data.js`. This file will contain the product data for your app.
   - Open your code editor (VS Code) and create the file `src/data.js.`
2. **Add product data:**
   - Define an array of product objects, each containing properties like `id`, `name, description, price`, and `stock`.
   - This data will be imported and used in the components to display product information.

- Declare a constant variable products and assigns it an array of product objects. `const products = [ ... ];`

- Export the products array as the default export from this module, making it available for import in other files. `export default products;`

```
const products = [
  {
    id: 1,
    name: "React Book",
    description: "A high-performance book about React.",
    price: "$120",
    stock: 10
  },
  {
    id: 2,
    name: "Angular Book",
    description: "A latest-gen Angular book.",
    price: "$80",
    stock: 20
  }
  // Add more products here
];

export default products;
```

3. Create Product and ProductList Components:

1. **Create a directory for components:**

    - Inside the `src` directory, create a new directory named components. This directory will contain all the component files for better organization.

      ```
      mkdir src/components
      ```

2. **Create the Product component:**

    - Create a new file named `Product.js` inside the components directory. This component will display the details of a single product.

    - `import React from 'react';`

- Declare a functional component named `Product` that receives product and `onBack` as props: `function Product({ product, onBack }) { ... }`
- Render a button with an `onClick` event handler that calls the onBack function `<button onClick={onBack}>Back to list</button>`
- `export default Product;`
- Open `src/components/Product.js` and add the following code:

```
import React from 'react';

function Product({ product, onBack }) {
  return (
    <div>
      <h2>{product.name}</h2>
      <p><strong>Description:</strong>
{product.description}</p>
      <p><strong>Price:</strong> {product.price}</p>
      <p><strong>Stock:</strong> {product.stock}</p>
      <button onClick={onBack}>Back to list</button>
    </div>
  );
}

export default Product;
```

22. **Create the ProductList component:**

- Create a new file named `ProductList.js` inside the components directory. This component will display a list of products.
- `import React from 'react';`
- Import the products array from `data.js`: `import products from '../data';`
- Declare a functional component named ProductList that receives `onSelectProduct` as a prop. `function ProductList({ onSelectProduct }) { ... }`

- Iterate over the products array and returns a list item for each product. `products.map((product) => ( ... ))`
- Render a list item with an `onClick` event handler that calls the onSelectProduct function with the product as an argument. `<li key={product.id} onClick={() => onSelectProduct(product)}>{product.name}</li>`
- Exports the ProductList component as the default export from this module `export default ProductList;`
- Open `src/components/ProductList.js` and add the following code:

```
import React from 'react';
import products from '../data';

function ProductList({ onSelectProduct }) {
  return (
    <div>
      <h1>Product List</h1>
      <ul>
        {products.map((product) => (
          <li key={product.id} onClick={() =>
onSelectProduct(product)}>
            {product.name}
          </li>
        ))}
      </ul>
    </div>
  );
}

export default ProductList;
```

4. Create the Main App Component:

1. **Open the App.js file:**
   - Open the App.js file in the src directory. This file is the main component of your app.

- Import React and the useState hook from the 'react' package. `import React, { useState } from 'react';`
- Import the ProductList component. `import ProductList from './components/ProductList';`
- Import the Product component. `import Product from './components/Product';`
- Import the styles from App.css: `import './App.css';`
- Declare a functional component named App `function App() { ... }`
- Declare a state variable `selectedProduct` and a function `setSelectedProduct` to update it, initialized to null. `const [selectedProduct, setSelectedProduct] = useState(null);`
- Define a function to set the selected product: `const handleSelectProduct = (product) => { setSelectedProduct(product); };`
- Define a function to reset the selected product. `const handleBack = () => { setSelectedProduct(null); };`
- Return the JSX to render. `return ( ... );`
- Render a div with the class name App. `<div className="App"> ... </div>`
- Conditionally render either the `Product` component or the `ProductList` component based on the state. `{selectedProduct ? ( ... ) : ( ... )}`
- Export the App component as the default export from this module `export default App;`
- It should look like:

```
import React, { useState } from 'react';
import ProductList from './components/ProductList';
import Product from './components/Product';
import './App.css';

function App() {
  const [selectedProduct, setSelectedProduct] =
useState(null);

  const handleSelectProduct = (product) => {
    setSelectedProduct(product);
  };

  const handleBack = () => {
    setSelectedProduct(null);
  };

  return (
    <div className="App">
      {selectedProduct ? (
        <Product product={selectedProduct}
onBack={handleBack} />
      ) : (
        <ProductList
onSelectProduct={handleSelectProduct} />
      )}
    </div>
  );
}

export default App;
```

5. Style the App:

1. **Open the App.css file:**
    - Open the `App.css` file in the `src` directory. This file contains styles for your app.
    - Replace its content with the following styles:

```
.App {
  font-family: Arial, sans-serif;
  padding: 20px;
  text-align: center;
}

ul {
  list-style-type: none;
  padding: 0;
}

li {
  cursor: pointer;
  margin: 5px 0;
  padding: 10px;
  background-color: #f0f0f0;
  border: 1px solid #ccc;
  border-radius: 5px;
}

li:hover {
  background-color: #e0e0e0;
}

button {
  padding: 10px 20px;
  margin-top: 20px;
  border: none;
  background-color: #007bff;
  color: white;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}
```

  - These styles improve the appearance of the app, including the product list and buttons.

6. Running the App:

1. **Start the development server:**

    - In your terminal, run the following command to start the development server.

      ```
      npm start
      ```

2. **View the app:**
    - The app will automatically open in your default web browser. You should see a list of products.
    - Click on a product name to view its details. The Product component will display the selected product's information.
    - Click the "Back to list" button to return to the product list.