Scaffolding a Simple e-Commerce Website Project with React

This guide outlines the steps for building a simple e-commerce website using React, based on the project files provided and the `playlist.txt`. This guide aims to help students learn React through a hands-on project, building component by component.

High-Level Steps:

1. **Project Setup:**
    - **Initialize:** Create a new React app using Create React App (`npx create-react-app my-eshop`).
    - **Install Dependencies:** Install necessary dependencies (`npm install @material-ui/core @material-ui/icons react-router-dom react-currency-format firebase`).
    - **Structure Folders:** Create the `src/components` directory to hold individual React component files. Move `Header.js`, `Home.js`, `Checkout.js`, `Product.js`, `Subtotal.js`, `Login.js`, `CheckoutProduct.js` into `src/components`.
    - **Clean Up:** Remove unnecessary files generated by Create React App (`src/App.test.js`, `src/logo.svg`, `src/setupTests.js`).
2. **Building the User Interface (UI) Components:**
    - **Header Component:**
        - Create `src/components/Header.css` for styling the header.
        - Implement the Header component in `src/components/Header.js`:
            - Use Material UI icons for the shopping basket, storefront, and search icons.
            - Implement routing with `react-router-dom` to link the logo to the home page and the login button to the login page.
            - Display the basket item count dynamically based on the basket state.
    - **Product Component:**
        - Create `src/components/Product.css` for styling individual products.
        - Implement the Product component in `src/components/Product.js`:
            - Receive product props (id, title, image, price, rating) to dynamically display product information.
            - Implement the "Add to Basket" button functionality using the context API to update the basket state.
    - **Home Component:**
        - Create `src/components/Home.css` for styling the home page.
        - Implement the Home component in `src/components/Home.js`:
            - Include a hero banner image.
            - Display a grid of Product components, passing relevant data to each product.
    - **CheckoutProduct Component:**
        - Create `src/components/CheckoutProduct.css` for styling product listings on the checkout page.
        - Implement the CheckoutProduct component in `src/components/CheckoutProduct.js`:
            - Display individual product details (image, title, price, rating) from the basket.

        - Implement the "Remove from Basket" button to allow users to remove items.

    - **Checkout Component:**
        - Create `src/components/Checkout.css` for styling the checkout page layout.
        - Implement the Checkout component in `src/components/Checkout.js`:
            - Display the list of items in the basket using the CheckoutProduct component.
            - Include the Subtotal component to display the total price of basket items.

    - **Subtotal Component:**
        - Create `src/components/Subtotal.css` for styling the subtotal section.
        - Implement the Subtotal component in `src/components/Subtotal.js`:
            - Calculate and display the total price of items in the basket.
            - Include a "Proceed to Checkout" button (non-functional in this simple version).

    - **Login Component:**
        - Create `src/components/Login.css` for styling the login page.
        - Implement the Login component in `src/components/Login.js`:
            - Create a form with email and password input fields.
            - Implement Firebase authentication for user sign-in and registration.
            - Redirect users to the home page upon successful login.

3. **State Management with Context API:**

    - **Create Context:** Create `src/StateProvider.js` to define the context object and reducer.
    - **Initial State:** Define the initial application state in the reducer (`src/reducer.js`) – start with an empty basket.
    - **Reducer Function:** Implement the reducer function to handle state updates based on dispatched actions.
    - **Wrap App Component:** Wrap the main App component in `src/index.js` with the StateProvider, making the state accessible throughout the application.

4. **Connecting UI to State and Actions:**

    - **Product Component:**
        - Access the dispatch function from the context in `src/components/Product.js` to update the basket state when the "Add to Basket" button is clicked.
    - **CheckoutProduct Component:**
        - Similarly, access the dispatch function in `src/components/CheckoutProduct.js` to update the basket state when an item is removed.
    - **Header Component:**
        - Access the basket state from the context to display the dynamic count of items in the basket.
    - **Subtotal Component:**
        - Access the basket state to calculate and display the total price.

5. **Implementing Routing:**

    - **Setup Routing:**

        - In `src/App.js`, set up routing using `react-router-dom` to define routes for the home page, checkout page, and login page.
        - Render the corresponding components (Home, Checkout, Login) based on the current route.

6. **Setting up Firebase:**
    - **Create Firebase Project:** Create a new Firebase project and configure it for web development.
    - **Enable Authentication:** Enable email/password authentication in the Firebase console.
    - **Integrate Firebase:**
        - Create `src/firebase.js` to initialize Firebase with your project credentials.
        - Export the Firebase authentication instance for use in the Login component.

7. **Deploying the App:**
    - **Build Project:** Build the React project for production (`npm run build`).
    - **Deploy to Firebase:** Use Firebase Hosting to deploy your built application.

Breakdown of Components:

- **App.js:** The main component that renders the application and handles routing.
- **Header.js:** Displays the website header with navigation, search bar, and basket information.
- **Product.js:** Represents a single product, displaying its details and an "Add to Basket" button.
- **Home.js:** Displays the home page content, including a list of products.
- **CheckoutProduct.js:** Represents a product in the checkout page, allowing removal from the basket.
- **Checkout.js:** Displays the checkout page with the list of selected products and the subtotal.
- **Subtotal.js:** Calculates and displays the total price of items in the basket.
- **Login.js:** Handles user authentication using Firebase.
- **StateProvider.js:** Sets up the context API for state management.
- **reducer.js:** Contains the reducer function for updating the application state.
- **firebase.js:** Initializes Firebase and exports the authentication instance.

This scaffolding guide provides a structured approach to building the e-commerce website project. By following these steps, students can gain practical experience with React concepts like components, props, state management, context API, routing, and integrating external services like Firebase.## Step 1: Project Setup

This guide will walk you through building a simplified e-commerce website using React. We'll be utilizing functional components, hooks, and the Context API for state management. The goal is to give you hands-on experience with building dynamic user interfaces and handling data flow in a React application.

Objectives

- **Initialize a new React project using Create React App.**
- **Install the necessary project dependencies.**
- **Structure the project directories for organizing components.**
- **Clean up unnecessary files generated by Create React App.**

Let's get started!

1.1 Initializing a New React Project

We will use Create React App, a convenient tool that sets up the basic structure of our React application.

1. **Open your terminal or command prompt.**
2. **Navigate to your desired project directory.**
3. **Run the following command to create a new React app:**

```
npx create-react-app my-eshop
```

Replace `my-eshop` with your preferred project name. This command will generate a new folder with your project name containing the basic file structure of a React app.

4. **Navigate into your new project directory:**

```
cd my-eshop
```

1.2 Installing Project Dependencies

Our project requires several third-party libraries to implement specific features like UI components, routing, currency formatting, and Firebase integration.

1. **Make sure you are still inside your project directory in the terminal.**
2. **Run the following command to install all the required dependencies:**

```
npm install @material-ui/core @material-ui/icons react-router-dom react-currency-format firebase
```

This command will install the following libraries:

- **@material-ui/core & @material-ui/icons:** For using pre-built UI components from Material-UI.
- **react-router-dom:** For implementing routing and navigation between different pages (Home, Checkout, Login).
- **react-currency-format:** For formatting currency values.
- **firebase:** For backend services, including user authentication.

1.3 Structuring Project Folders

Organizing our project files is essential for maintainability. We'll create a dedicated folder for our React components.

1. **Inside your `src` folder, create a new folder named `components`.**
2. **Create the following files within the `src/components` directory:**
    - `Header.js`
    - `Home.js`
    - `Checkout.js`
    - `Product.js`
    - `Subtotal.js`
    - `Login.js`
    - `CheckoutProduct.js`

1.4 Cleaning Up Unnecessary Files

Create React App generates some files we won't be using in this project. Let's remove them.

1. **Delete the following files from the `src` folder:**
    - `App.test.js`
    - `logo.svg`
    - `setupTests.js`

You have now successfully set up the foundation for your React e-commerce application! You've initialized a new project, included all the necessary libraries, organized your project structure, and are ready to move on to building the UI components.

Step 2: Building the User Interface (UI) Components

In this step, we'll bring our e-commerce site to life by creating the UI components using React. We'll be using Material-UI for pre-built components to speed up the styling process and focus on functionality.

Objectives

- Create functional React components for each part of the UI (Header, Product, Home, CheckoutProduct, Checkout, Subtotal, Login).
- Implement basic styling for each component using CSS.
- Integrate Material UI icons for visual elements.
- Set up basic routing using `react-router-dom` to navigate between Home, Login, and Checkout pages.

Let's break down the process for each component:

2.1 Header Component

The Header will contain our logo, a search bar, navigation links, and the shopping basket.

1. **Create `src/components/Header.css` for styling:**

```
.header {
    height: 60px;
    display: flex;
    align-items: center;
    background-color: #131921;
    position: sticky;
    top: 0;
    z-index: 100;
}

.header__logo {
    display: flex;
    align-items: center;
    color: #ff9f00;
    margin: 0 25px;
}

.header__logoImage {
    margin-right: 10px;
}

.header__logoTitle {
```

```
    text-decoration: none;
    border: 0;
    color: white;
}

.header__search {
    display: flex;
    flex: 1;
    align-items: center;
    border-radius: 24px;
    margin: 0 10px;
}

.header__searchInput {
    height: 12px;
    padding: 10px;
    border: none;
    width: 100%;
}

.header__searchIcon {
    padding: 5px;
    height: 22px !important;
    background-color: #ff9f00;
}

.header__nav {
    display: flex;
    justify-content: space-evenly;
}

.nav__item {
    display: flex;
    flex-direction: column;
    margin-left: 10px;
    margin-right: 10px;
    color: white;
}

.nav__itemLineOne {
    font-size: 10px;
}

.nav__itemLineTwo {
    font-size: 13px;
    font-weight: 800;
}

.nav__itemBasket {
    display: flex;
    align-items: center;
    color: white;
```

```
    margin-right: 50px;
    margin-left: 10px;
}

.nav__basketCount {
    margin-left: 10px;
    margin-right: 10px;
}
```

2. **Implement the Header component in `src/components/Header.js` :**

```
import React from "react";
import "./Header.css";
import ShoppingBasketIcon from '@material-ui/icons/ShoppingBasket';
import StorefrontIcon from '@material-ui/icons/Storefront';
import SearchIcon from '@material-ui/icons/Search';
import { Link } from "react-router-dom";
import { useStateValue } from "./StateProvider";

function Header() {
    const [{basket}, dispatch] = useStateValue();
    return (
        <div className="header">
          <Link to="/" style={{ textDecoration:"none" }}>
            <div className="header__logo">
                <StorefrontIcon className="header__logoImage" fontSize="large"/>
                <h2 className="header__logoTitle">eShop</h2>
            </div>
          </Link>


      <div className="header__search">
        <input type="text" className="header__searchInput" />
        <SearchIcon className="header__searchIcon" />
      </div>

      <div className="header__nav">
        <Link to="/login" style={{ textDecoration:"none" }}>
          <div className="nav__item">
            <span className="nav__itemLineOne">Hello Guest</span>
            <span className="nav__itemLineTwo">Sign In</span>
          </div>
        </Link>
        <div className="nav__item">
          <span className="nav__itemLineOne">Your</span>
          <span className="nav__itemLineTwo">Shop</span>
        </div>
        <Link to="/checkout" style={{ textDecoration: "none" }}>
          <div className="nav__itemBasket">
            <ShoppingBasketIcon/>
            <span className="nav__itemLineTwo nav__basketCount">{basket.length}
```

```
</span>
          </div>
        </Link>
      </div>
        </div>
    )
}

export default Header
```

Explanation:

- We import necessary components from `react-router-dom` for routing.
- The `Link` component is used to create links to the home page ("/") and the login page ("/login").
- We use the `useStateValue` hook to access the `basket` from the context API to display the number of items in the basket dynamically.

2.2 Product Component

This component will represent a single product in our shop.

1. **Create `src/components/Product.css` for styling:**

```
.product {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: flex-end;
    margin: 10px;
    padding: 20px;
    width: 100%;
    max-height: 400px;
    min-width: 100px;
    background-color: white;
    z-index: 1;
  }

  .product__rating {
    display: flex;
  }

  .product > img {
    max-height: 200px;
    width: 100%;
    object-fit: contain;
    margin: 20px 0 0 15px;
  }

  .product > button {
    background: #ff9f00;
    border: 1px solid;
    margin-top: 10px;
```

```
    border-color: #a88734 #9c7e31 #846a29;
    color: #111;
  }

  .product__price {
    margin-top: 5px;
  }

  .product__info {
    height: 100px;
    margin-bottom: 15px;
  }
```

2. **Implement the Product component in `src/components/Product.js` :**

```
import React from "react";
import "./Product.css";
import { useStateValue } from "./StateProvider";

function Product({id, title, image, price, rating}) {

    const [state, dispatch] = useStateValue();

    const addToBasket = () => {
        dispatch({
            type: "ADD_TO_BASKET",
            item: {
                id: id,
                image: image,
                price: price,
                rating: rating,
            },
        });
    };

    return (
        <div className="product">
            <div className="product__info">
                <p>{title}</p>
                <p className="product__price">
                    <small>$</small>
                    <strong>{price}</strong>
                </p>
                <div className="product__rating">
                    {Array(rating)
                    .fill()
                    .map((_, i) => (
                        <p>⭐</p>
                    ))}
                </div>
            </div>
```

```
            <img src={image} />

            <button onClick={addToBasket}>Add to Basket</button>
        </div>
    )
}

export default Product
```

Explanation:

- The `Product` component receives product information (id, title, image, price, rating) as props.
- It uses `useStateValue` to access the dispatch function from the context.
- The `addToBasket` function dispatches an "ADD_TO_BASKET" action with the product information when the "Add to Basket" button is clicked, which will be handled by the reducer to update the basket state.

2.3 Home Component

This component will display a list of products and a hero banner.

1. **Create `src/components/Home.css` for styling:**

```
.home {
    display: flex;
    justify-content: center;
    margin-left: auto;
    margin-right: auto;
    max-width: 1500px;
}
.home__image {
    width: 100%;
    height: 400px;
    z-index: -1;
    margin-bottom: -50px;
    mask-image: linear-gradient(to bottom, rgba(0, 0, 0, 1), rgba(0, 0, 0, 0));
}

.home__row {
    display: flex;
    z-index: 1;
    margin-left: 5px;
    margin-right: 5px;
}
```

2. **Implement the Home component in `src/components/Home.js` :**

```
import React from "react";
import "./Home.css";
import Product from "./Product";
```

```
function Home() {
    return (
        <div className="home">
            <div className="home__container">

                <img src="https://www.x-cart.com/wp-
content/uploads/2019/01/ecommerce-768x278.jpg" alt="" className="home__image" />

                <div className="home__row">
                    <Product
                        id="12321341"
                        title="Bennett Mystic 15.6 inch Laptop Shoulder Messenger
Sling Office Bag, Water Repellent Fabric for Men and Women (Blue)"
                        price={11.96}
                        rating={5}
                        image="https://images-na.ssl-images-
amazon.com/images/I/71mEsHyzSCL._SL1000_.jpg"
                    />
                    <Product
                        id="49538094"
                        title="IFB 30 L Convection Microwave Oven (30BRC2, Black,
With Starter Kit)"
                        price={239.0}
                        rating={4}
                        image="https://images-na.ssl-images-
amazon.com/images/I/81D8pNFmWzL._SL1500_.jpg"
                    />
                </div>

                <div className="home__row">
                    <Product
                        id="4903850"
                        title="All the Light we Cannot See: The Breathtaking World
Wide Bestseller Paperback"
                        price={199.99}
                        rating={3}
                        image="https://images-eu.ssl-images-
amazon.com/images/I/514kNkerQ0L._SY264_BO1,204,203,200_QL40_FMwebp_.jpg"
                    />
                    <Product
                        id="23445930"
                        title="Amazon Echo (3rd generation) | Smart speaker with
Alexa, Charcoal Fabric"
                        price={98.99}
                        rating={5}

image="https://media.very.co.uk/i/very/P6LTG_SQ1_0000000071_CHARCOAL_SLf?
$300x400_retinamobilex2$"
                    />
                    <Product
                        id="3254354345"
                        title="New Apple iPad Pro (12.9-inch, Wi-Fi, 128GB) - Silver
```

```
(4th Generation)"
                        price={598.99}
                        rating={4}
                        image="https://images-na.ssl-images-
amazon.com/images/I/816ctt5WV5L._AC_SX385_.jpg"
                    />
                </div>

                <div className="home__row">
                    <Product
                        id="90829332"
                        title="Samsung LC49RG90SSUXEN 49' Curved LED Gaming Monitor
- Super Ultra Wide Dual WQHD 5120 x 1440"
                        price={1094.98}
                        rating={4}
                        image="https://images-na.ssl-images-
amazon.com/images/I/6125mFrzr6L._AC_SX355_.jpg"
                    />
                </div>
            </div>
        </div>
    )
}

export default Home
```

Explanation:

- The `Home` component renders a hero banner image and a grid of `Product` components.
- You can replace the placeholder product information with your own.

2.4 CheckoutProduct Component

This component will be used to display a product on the Checkout page.

1. **Create `src/components/CheckoutProduct.css` for styling:**

```
.checkoutProduct {
    display: flex;
    margin-top: 20px;
    margin-bottom: 20px;
}

.checkoutProduct__info {
    padding-left: 20px
}

.checkoutProduct__info > button {
    background: #ff9f00;
    border: 1px solid;
    margin-top: 10px;
    border-color: #a88734 #9c7e31 #846a29;
```

```
    color: #111;
}

.checkoutProduct__image {
    object-fit: contain;
    width: 180px;
    height: 180px;
  }

.checkoutProduct__rating {
    display: flex;
  }

.checkoutProduct__title {
    font-size: 17px;
    font-weight: 800;
}
```

2. **Implement the CheckoutProduct component in `src/components/CheckoutProduct.js` :**

```
import React from 'react'
import "./CheckoutProduct.css";
import { useStateValue } from "./StateProvider";

function CheckoutProduct({ id, image, title, price, rating }) {
    const [{basket}, dispatch] = useStateValue();

    const removeFromBasket = () => {
        dispatch({
            type: "REMOVE_FROM_BASKET",
            id: id,
        })
    }
    return (
        <div className="checkoutProduct">
            <img src={image} alt="" className="checkoutProduct__image" />

            <div className="checkoutProduct__info">
                <p className="checkoutProduct__title">{title}
                </p>
                <p className="checkoutProduct__price">
                    <small>$</small>
                    <strong>{price}</strong>
                </p>
                <div className="checkoutProduct__rating">
                    {Array(rating)
                    .fill()
                    .map((_, i) => (
                        <p>⭐</p>
                    ))}
                </div>
```

```
                <button onClick={removeFromBasket}>Remove from Basket</button>
            </div>
        </div>
    )
}

export default CheckoutProduct
```

Explanation:

- Similar to the `Product` component, `CheckoutProduct` displays product information.
- It also has a "Remove from Basket" button that dispatches a "REMOVE_FROM_BASKET" action to remove the item from the basket state.

2.5 Checkout Component

The Checkout component will display the items in the basket and the order subtotal.

1. **Create `src/components/Checkout.css` for styling:**

```
.checkout {
    display: flex;
    padding: 20px;
    background-color: white;
    height: max-content;
}

.checkout__ad {
    width: 100%;
    margin-bottom: 10px;
}

.checkout__title {
    margin-right: 10px;
    padding: 10px;
    border-bottom: 1px solid lightgray;
}

.checkout__left {
    display: flex;
    flex-direction: column;
}
```

2. **Implement the Checkout component in `src/components/Checkout.js` :**

```
import { SportsBasketball } from "@material-ui/icons";
import React from "react";
import "./Checkout.css";
import CheckoutProduct from "./CheckoutProduct";
import Subtotal from "./Subtotal.js";
import { useStateValue } from "./StateProvider";
```

```
function Checkout() {
    const [{basket}, dispatch] = useStateValue();
    return (
        <div className="checkout">
            <div className="checkout__left">
                <img src="https://images-na.ssl-images-
amazon.com/images/G/02/UK_CCMP/TM/OCC_Amazon1._CB423492668_.jpg" alt=""
className="checkout__ad" />
                <div>
                    <h2 className="checkout__title">
                        Your Shopping Basket
                    </h2>
                    {basket.map(item => (
                        <CheckoutProduct
                            id = {item.id}
                            title = {item.title}
                            image = {item.image}
                            price = {item.price}
                            rating = {item.rating}
                        />
                    ))}
                </div>
            </div>

            <div className="checkout__right">
                <Subtotal />
            </div>
        </div>
    )
}

export default Checkout
```

Explanation:

- It accesses the `basket` from the context API.
- It maps over the `basket` array to render a `CheckoutProduct` component for each item.
- It includes the `Subtotal` component to display the total price.

2.6 Subtotal Component

This component will calculate and display the total price of all items in the basket.

1. **Create `src/components/Subtotal.css` for styling:**

```
.subtotal {
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    width: 300px;
    height: 100px;
    padding: 20px;
```

```
    background-color: #f3f3f3;
    border: 1px solid #dddddd;
    border-radius: 3px;
}

.subtotal__gift {
    display: flex;
    align-items: center;
}

.subtotal__gift > input {
    margin-right: 5px;
}

.subtotal > button {
    background: #ff9f00;
    border-radius: 2px;
    width: 100%;
    height: 30px;
    border: 1px solid;
    margin-top: 10px;
    border-color: #a88734 #9c7e31 #846a29;
    color: #111;
}
```

2. **Implement the Subtotal component in `src/components/Subtotal.js` :**

```
import React from "react";
import "./Subtotal.css";
import CurrencyFormat from "react-currency-format";
import { useStateValue } from "./StateProvider";
import { getBasketTotal } from "./reducer";


function Subtotal() {
    const [{basket}, dispatch] = useStateValue();
    return (
        <div className="subtotal">
            <CurrencyFormat
                renderText = {(value) => (
                    <>
                        <p>
                            Subtotal ({basket.length} items): <strong>${value}
</strong>
                        </p>
                        <small className="subtotal__gift">
                            <input type="checkbox" /> This order contains a gift
                        </small>
                    </>
                )}
```

```
                decimalScale={2}
                value={getBasketTotal(basket)}
                displayType={"text"}
                thousandSeparator={true}

            />

            <button>Proceed to Checkout</button>
        </div>
    )
}

export default Subtotal
```

Explanation:

- It utilizes the `CurrencyFormat` component from `react-currency-format` for formatting the price.
- It imports and uses the `getBasketTotal` selector function from the reducer to calculate the total price of all items in the basket.

2.7 Login component

The Login component will handle user authentication using Firebase.

1. **Create `src/components/Login.css` for styling:**

```
.login {
    background-color: white;
    height: 100vh;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.login__logo {
    display: flex;
    margin: 30px 0;
}

.login__logoImage {
    color: #ff9f00;
}

.login__logoTitle {
    color: black;
}

.login__container {
    width: 300px;
    height: fit-content;
    display: flex;
```

```
    flex-direction: column;
    border-radius: 5px;
    border: 1px solid lightgray;
    padding: 20px;
}

.login__container > h1 {
    font-weight: 500;
    margin-bottom: 20px;
  }

.login__container > form > h5 {
    margin-bottom: 5px;
}

.login__container > form > input {
    height: 30px;
    margin-bottom: 10px;
    background-color: white;
    width: 98%;
  }

.login__container > p {
    margin-top: 15px;
    font-size: 12px;
}

.login__signInButton {
    background: #ff9f00;
    border-radius: 2px;
    width: 100%;
    height: 30px;
    border: 1px solid;
    margin-top: 10px;
    border-color: #a88734 #9c7e31 #846a29;
  }

  .login__registerButton {
    border-radius: 2px;
    width: 100%;
    height: 30px;
    border: 1px solid;
    margin-top: 10px;
    border-color: darkgray;
  }
```

2. **Implement the Login component in `src/components/Login.js` :**

```
import React, { useState } from 'react';
import './Login.css'
import { Link, useHistory } from "react-router-dom";
```

```
import StorefrontIcon from '@material-ui/icons/Storefront';
import { auth } from "./firebase";

function Login() {
    const history = useHistory();
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');

    const signIn = e => {
        e.preventDefault();

        auth
            .signInWithEmailAndPassword(email, password)
            .then(auth => {
                history.push('/');
            })
            .catch(error => alert(error.message))

    }

    const register = e => {
        e.preventDefault();

        auth
            .createUserWithEmailAndPassword(email, password)
            .then((auth) => {
                if (auth) {
                    history.push('/');
                }
            })
            .catch(error => alert(error.message))

    }

    return (
        <div className='login'>
            <Link to='/' style={{ textDecoration: "none" }}>
                <div className="login__logo">
                    <StorefrontIcon className="login__logoImage" fontSize="large" />
                    <h2 className="login__logoTitle">eSHOP</h2>
                </div>
            </Link>

            <div className='login__container'>
                <h1>Sign-in</h1>

                <form>
                    <h5>E-mail</h5>
                    <input type='text' value={email} onChange={e =>
setEmail(e.target.value)} />

                    <h5>Password</h5>
```

```
                    <input type='password' value={password} onChange={e =>
setPassword(e.target.value)} />

                    <button type='submit' className='login__signInButton' onClick=
{signIn}>Sign In</button>
                </form>

                <p>
                    By signing-in you agree to the eShop Website Conditions of Use &
Sale. Please
                    see our Privacy Notice, our Cookies Notice and our Interest-
Based Ads Notice.
                </p>

                <button className='login__registerButton' onClick={register}>Create
your eShop Account</button>
            </div>
        </div>
    )
}

export default Login;
```

Explanation:

- This component includes a simple form for email and password input.
- The `signIn` function handles user sign-in with Firebase authentication.
- The `register` function handles user registration.
- Upon successful sign-in/registration, it should redirect users to the home page.

After completing this step, you'll have the basic UI structure of your React e-commerce application. In the next step, you will connect these UI components to the state and actions using the Context API to manage the application's state efficiently.

Step 3: State Management with Context API

In this step, we'll implement state management in our e-commerce application using the Context API. This will allow us to manage the application's state, particularly the shopping basket, in a centralized and efficient way, making it accessible to different components without prop drilling.

Objectives

- Understand the concept of state management in React and why it is crucial.
- Learn how to use the Context API in a React application.
- Implement a global state for managing the shopping basket.
- Create a reducer function to handle state updates based on specific actions.

3.1 Creating the Context and Initial State

1. **Create `src/StateProvider.js` :** This file will hold our context object and the logic to provide the state to our application.

```
import React, { createContext, useContext, useReducer } from "react";

// Create the context object
export const StateContext = createContext();

// Create the StateProvider component that will wrap our app
export const StateProvider = ({ reducer, initialState, children }) => (
  <StateContext.Provider value={useReducer(reducer, initialState)}>
    {children}
  </StateContext.Provider>
);

// Export the useStateValue hook for accessing state and dispatch
export const useStateValue = () => useContext(StateContext);
```

Explanation:

- We create a context object using `createContext()`.
- The `StateProvider` component uses the `useReducer` hook. This hook manages our state and updates it based on the actions dispatched.
- The `useStateValue` hook is a custom hook that allows components to easily access the context's value, which includes both the state and the dispatch function.

2. **Define the initial state in `src/reducer.js`:**

```
// src/reducer.js

export const initialState = {
  basket: [], // The basket will start as an empty array
};

// ... (Rest of the reducer.js file, we'll implement the reducer function next)
```

3.2 Implementing the Reducer Function

The reducer function is the heart of our state management. It takes the current state and an action as arguments and returns a new state based on the action.

Update `src/reducer.js` with the following:

```
// src/reducer.js

export const initialState = {
    basket: [],
};

//Selector
export const getBasketTotal = (basket) => {
    return(basket?.reduce((amount, item) => item.price + amount, 0));
}
```

```
const reducer = (state, action) => {
  console.log(action);
  switch (action.type) {
    case "ADD_TO_BASKET":
      return {
        ...state, // Keep the existing state
        basket: [...state.basket, action.item], // Add the new item to the basket
      };

    case "REMOVE_FROM_BASKET":
        // Logic for removing item from basket (we will implement this in step 4)
        const index = state.basket.findIndex(
            (basketItem) => basketItem.id === action.id
        );

        let newBasket = [...state.basket];

        if (index>=0) {
            newBasket.splice(index, 1);
        } else {
            console.warn(
                `Can't remove product(id: ${action.id}) as its not in the basket!`
            )
        }

        return {
            ...state,
            basket: newBasket
        }
    default:
      return state; // If action type doesn't match, return the current state
  }
};

export default reducer;
```

Explanation:

- **`ADD_TO_BASKET`** : This action will be dispatched when a user clicks the "Add to Basket" button on a product. The reducer adds the new item to the basket while keeping the existing items.
- **`REMOVE_FROM_BASKET`** : (We'll implement the logic for this action in a later step) - This action will be dispatched when the user clicks "Remove from Basket."
- **`default`** : It's essential to have a default case in the reducer. If the reducer encounters an action it doesn't recognize, it should return the current state without any modifications.

3.3 Wrapping the App Component with the Provider

To make our state accessible throughout the application, we need to wrap our main `App` component with the `StateProvider` .

Update `src/index.js` :

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reducer, { initialState } from "./reducer";
import { StateProvider } from "./StateProvider"; // Import StateProvider

ReactDOM.render(
  <React.StrictMode>
    <StateProvider initialState={initialState} reducer={reducer}>
      <App />
    </StateProvider>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Now, any component within our `App` component can access the state and dispatch function provided by the context.

By completing this step, we've set up the foundation for global state management in our application. In the next step, we'll connect our UI components to this state, allowing them to read from and update the state based on user interactions.

Step 4: Connecting UI to State and Actions

We have successfully set up our application's state management using the Context API and a reducer. In this step, we'll connect our UI components to this state, allowing them to both dispatch actions that modify the state and access the state to display updated information dynamically.

Objectives

- Understand how to dispatch actions from UI components to update the global state.
- Learn how to access the state from UI components to render dynamic content.
- Implement the "Add to Basket" functionality for the `Product` component.
- Implement the "Remove from Basket" functionality for the `CheckoutProduct` component.
- Update the `Header` component to display the item count dynamically.
- Connect the `Subtotal` component to the basket state for total price calculation.

4.1 Dispatching Actions from UI Components

We'll start by connecting our `Product` component's "Add to Basket" button to the global state.

1. **Open `src/components/Product.js` and import `useStateValue`:**

```
import React from "react";
import "./Product.css";
import { useStateValue } from "./StateProvider"; // Import useStateValue

// ... (rest of the Product component code)
```

2. **Access the dispatch function within the component:**

```
function Product({ id, title, image, price, rating }) {
  const [{ basket }, dispatch] = useStateValue(); // Destructure dispatch

  const addToBasket = () => {
    // Dispatch the ADD_TO_BASKET action when the button is clicked
    dispatch({
      type: "ADD_TO_BASKET",
      item: {
        id: id,
        title: title,
        image: image,
        price: price,
        rating: rating,
      },
    });
  };

  return (
    // ... (rest of the Product component JSX)
  );
}
```

Explanation:

- We use the `useStateValue()` hook to get access to the state and the dispatch function provided by our Context API.
- When the "Add to Basket" button is clicked, the `addToBasket` function is triggered.
- The `addToBasket` function dispatches an action of type `ADD_TO_BASKET` along with the product's information. This action will be sent to our reducer.

4.2 Removing Items from the Basket

We will implement a similar approach for removing items from the basket in the `CheckoutProduct` component.

1. **Open `src/components/CheckoutProduct.js`:**
2. **Implement the `removeFromBasket` function:**

```
import React from 'react';
import "./CheckoutProduct.css";
import { useStateValue } from "./StateProvider";

function CheckoutProduct({ id, image, title, price, rating }) {
  const [{ basket }, dispatch] = useStateValue();

  const removeFromBasket = () => {
    // Dispatch the REMOVE_FROM_BASKET action when the button is clicked
    dispatch({
      type: "REMOVE_FROM_BASKET",
      id: id,
```

```
    });
  };

  // ... (rest of your CheckoutProduct component)
}
```

Explanation:

- Similar to the `Product` component, we access the `dispatch` function from `useStateValue`.
- The `removeFromBasket` function dispatches the `REMOVE_FROM_BASKET` action with the `id` of the product to be removed. Our reducer already has the logic to handle this action type.

4.3 Accessing State for Dynamic Content

Now, let's update the `Header` component to display the number of items in the basket dynamically.

1. **Open `src/components/Header.js`:**

2. **Access the basket state:**

```
import React from "react";
import "./Header.css";
// ... (other imports)
import { useStateValue } from "./StateProvider";

function Header() {
  const [{ basket }, dispatch] = useStateValue(); // Get the basket from the state

  return (
    // ... (rest of the Header component JSX)
    <Link to="/checkout" style={{ textDecoration: "none" }}>
          <div className="nav__itemBasket">
            <ShoppingBasketIcon/>
            <span className="nav__itemLineTwo nav__basketCount">
                {basket?.length} {/* Display the length of the basket */}
            </span>
          </div>
    </Link>
    // ... (rest of the Header component JSX)

  );
}
```

Explanation:

- We access the `basket` array directly from the state using the `useStateValue` hook.
- The length of the `basket` array (representing the number of items) is then displayed in the header.

4.4 Connecting the Subtotal Component

Finally, let's connect the `Subtotal` component to calculate and display the total price of the items in the basket.

1. **Open `src/components/Subtotal.js`:**
2. **Import the `getBasketTotal` selector and use it:**

```
import React from "react";
import "./Subtotal.css";
import CurrencyFormat from "react-currency-format";
import { useStateValue } from "./StateProvider";
import { getBasketTotal } from "./reducer"; // Import the selector

function Subtotal() {
    const [{ basket }, dispatch] = useStateValue();

    return (
        <div className="subtotal">
            {/* ... other JSX ... */}
            <CurrencyFormat
                // ...
                value={getBasketTotal(basket)} // Use the selector here
                // ...
            />
            {/* ... other JSX ... */}
        </div>
    );
}
```

Explanation:

- We import the `getBasketTotal` selector that we defined in `reducer.js`.
- We pass the `basket` array to the selector, which calculates the total price and returns it.

By completing this step, we've successfully connected our UI components to the centralized state. This connection enables us to build a dynamic and interactive user experience, where user actions directly impact the application's state, and the UI reflects those changes in real-time.

Step 5: Implementing Routing with `react-router-dom`

In this step, we will set up routing in our e-commerce application using the `react-router-dom` library. This will allow users to navigate between different pages (Home, Checkout, Login) seamlessly, creating a more intuitive and user-friendly experience.

Objectives

- Install the `react-router-dom` library.
- Understand the concepts of routing, routes, and navigation in a single-page application.
- Define routes for the Home, Checkout, and Login pages in our application.
- Implement conditional rendering of components based on the active route.

5.1 Installing `react-router-dom`

We'll start by installing the necessary dependency. If you haven't already, run the following command in your project's terminal:

```
npm install react-router-dom
```

5.2 Setting Up the Router

Now, we need to wrap our application with the `BrowserRouter` component provided by `react-router-dom`. This component provides the necessary context for routing to work correctly.

1. **Open `src/index.js`:**
2. **Import `BrowserRouter` and wrap the `<App />` component:**

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reducer, { initialState } from "./reducer";
import { StateProvider } from "./StateProvider";
import { BrowserRouter } from 'react-router-dom'; // Import BrowserRouter

ReactDOM.render(
  <React.StrictMode>
    <StateProvider initialState={initialState} reducer={reducer}>
      <BrowserRouter> {/* Wrap the App component */}
        <App />
      </BrowserRouter>
    </StateProvider>
  </React.StrictMode>,
  document.getElementById('root')
);
```

Explanation:

- The `BrowserRouter` component now manages the routing for our entire application.

5.3 Defining Routes

Let's define the routes for our different pages within the `App.js` file:

1. **Open `src/App.js`:**
2. **Import necessary components from `react-router-dom`:**

```
import './App.css';
import Home from "./Home";
import Header from "./Header";
import Checkout from "./Checkout";
import Login from "./Login";
import { BrowserRouter as Router, Route, Switch } from "react-router-dom"; // Import
components
```

3. **Set up the `Switch` and `Route` components:**

```
function App() {
  return (
    <div className="App">
      <Router> {/* Now Router instead of BrowserRouter */}

        <Switch> {/* Use Switch to render only one route at a time */}

          <Route path="/login">
            <Login />
          </Route>

          <Route path="/checkout">
            <Header />
            <Checkout />
          </Route>

          <Route path="/"> {/* Default route for the home page */}
            <Header />
            <Home />
          </Route>

        </Switch>

      </Router>
    </div>
  );
}
```

Explanation:

- We import `Switch`, `Route`, and optionally rename `BrowserRouter` to `Router` for brevity.
- The `Switch` component ensures that only one route is rendered at a time.
- Each `Route` component defines a specific path and the component to be rendered when that path matches the current URL.
- The `path="/"` route acts as our default route, rendering the `Home` component when no other route matches.

With routing implemented, your e-commerce application is now more user-friendly and efficient. Users can navigate between pages smoothly, enhancing their browsing and shopping experience!

Step 6: Setting up Firebase for User Authentication

In this crucial step, we will integrate Firebase into our e-commerce application. Firebase offers a suite of powerful tools for web development, and we'll be utilizing its authentication service to manage user sign-in and registration securely.

Objectives

- Create a new Firebase project specifically for our e-commerce website.
- Enable email/password authentication within our Firebase project.
- Initialize Firebase within our React application.
- Export the necessary Firebase authentication instance to be used within our `Login` component.

6.1 Creating a Firebase Project

1. **Go to the Firebase Console:** Navigate to https://console.firebase.google.com/
2. **Create a Project:** Click on "Add project" and follow the on-screen instructions to create a new Firebase project. Choose a descriptive name for your project.
3. **Configure Project:** Once your project is created, select the "Web" option (denoted by the `< >` icon) to configure it for web development.
4. **Register App:** You'll be prompted to register your app. Provide a nickname for your app. You can skip setting up Firebase Hosting for now.
5. **Copy Configuration:** You'll be presented with your Firebase project's configuration object. This object contains sensitive information, including your API keys. **Copy this configuration object to your clipboard; we'll use it shortly.**

6.2 Enabling Email/Password Authentication

1. **Navigate to Authentication:** From your Firebase project's dashboard, click on "Authentication" in the left-hand sidebar.
2. **Enable Method:** Select "Email/Password" from the list of sign-in methods. Toggle it to "Enabled" and save your changes.

6.3 Initializing Firebase in Your React App

1. **Create `firebase.js`:** In the `src` directory of your React project, create a new file named `firebase.js`.
2. **Paste Configuration and Initialize:** Paste the Firebase configuration object you copied earlier into `firebase.js`. Then, add the following code below it to initialize Firebase in your app:

```
import firebase from "firebase/compat/app";
import "firebase/compat/auth";
import "firebase/compat/firestore";

const firebaseConfig = {
    // ... your Firebase config object here
};

// Initialize Firebase
const firebaseApp = firebase.initializeApp(firebaseConfig);

// Export Authentication and Database instances
const db = firebaseApp.firestore(); // For potential future use with Firestore
const auth = firebase.auth();

export { db, auth };
```

Explanation:

- We import the necessary modules from Firebase.
- The `firebaseConfig` object is used to initialize your Firebase app.
- We initialize Firebase using `firebase.initializeApp(firebaseConfig)`.

- For convenience, we export the `auth` instance, which we'll use in our `Login` component.
- We also export `db` for potential future use with Cloud Firestore if you choose to expand the project.

6.4 Integrating Firebase with the Login Component

We've laid the groundwork for user authentication. In the next step, we will connect the `Login` component to our Firebase project, enabling users to create accounts and sign in securely.

Step 7: Connecting Everything Together

This step focuses on integrating the components built in the previous steps and bringing our e-commerce website to life. We'll implement product display on the home page, add-to-basket functionality, a functional checkout page, and user authentication using Firebase.

Objectives

- Render product information dynamically on the home page using the `Product` component.
- Implement the "Add to Basket" functionality to update the basket state using context API.
- Configure routing to navigate between the home page, checkout page, and login page.
- Integrate Firebase authentication into the login page for user sign-in and registration.

7.1 Rendering Products on the Home Page

1. **Import Components:** In `src/components/Home.js`, import the `Product` component:

```
import Product from "./Product";
```

2. **Create Product Data:** For simplicity, we'll hardcode product data within the `Home` component. In a real-world scenario, this data would likely be fetched from an API or database. Add the following product data inside the `Home` component's `return` statement before the divs with className `home__row`:

```
const products = [
  {
    id: "12321341",
    title:
      "Bennett Mystic 15.6 inch Laptop Shoulder Messenger Sling Office Bag,
Water Repellent Fabric for Men and Women (Blue)",
    price: 11.96,
    rating: 5,
    image:
      "https://images-na.ssl-images-
amazon.com/images/I/71mEsHyzSCL._SL1000_.jpg",
  },
  {
    id: "49538094",
    title: "IFB 30 L Convection Microwave Oven (30BRC2, Black, With Starter
Kit)",
    price: 239.0,
    rating: 4,
    image:
      "https://images-na.ssl-images-
```

```
amazon.com/images/I/81D8pNFmWzL._SL1500_.jpg",
  },
  // Add more products here...
];
```

3. **Map Product Data to Components:** Utilize JavaScript's `map` function to dynamically create `Product` components for each product in the `products` array. Replace the existing code within the `home__row` divs with:

```
<div className="home__row">
  {products.slice(0, 2).map((product) => (
    <Product
      key={product.id}
      id={product.id}
      title={product.title}
      price={product.price}
      rating={product.rating}
      image={product.image}
    />
  ))}
</div>
<div className="home__row">
  {products.slice(2, 5).map((product) => (
    <Product
      key={product.id}
      id={product.id}
      title={product.title}
      price={product.price}
      rating={product.rating}
      image={product.image}
    />
  ))}
</div>
<div className="home__row">
  {products.slice(5).map((product) => (
    <Product
      key={product.id}
      id={product.id}
      title={product.title}
      price={product.price}
      rating={product.rating}
      image={product.image}
    />
  ))}
</div>
```

4. **Add Key Prop:** Ensure each `Product` component has a unique `key` prop (we use `product.id`) for React's efficient rendering.

7.2 Implementing Add to Basket Functionality

1. **Import `useStateValue`:** In `Product.js`, import `useStateValue` to access the global state:

```
import { useStateValue } from "./StateProvider";
```

2. **Access Dispatch Function:** Destructure the `dispatch` function from `useStateValue`:

```
const [{ basket }, dispatch] = useStateValue();
```

3. **Create `addToBasket` Function:** Define the `addToBasket` function within the `Product` component to handle adding items to the basket.

```
const addToBasket = () => {
  // Dispatch the action to the reducer
  dispatch({
    type: "ADD_TO_BASKET",
    item: {
      id: id,
      title: title,
      image: image,
      price: price,
      rating: rating,
    },
  });
};
```

4. **Attach `addToBasket` to Button:** Attach this function to the "Add to Basket" button within the `Product` component:

```
<button onClick={addToBasket}>Add to Basket</button>
```

7.3 Setting Up Routing

1. **Import Routing Components:** In `src/App.js`, import the necessary components from `react-router-dom`:

```
import { BrowserRouter as Router, Route, Switch } from "react-router-dom";
```

2. **Wrap with Router:** Wrap the main `App` component with the `BrowserRouter` component.

```
 return (
     <div className="App">
         <Router>
             {/* ...rest of your application... */}
         </Router>
     </div>
 )
```

3. **Define Routes:** Define the routes for the home page (`/`), checkout page (`/checkout`), and login page (`/login`):

```
<Router>
  <Switch> {/* Ensure only one route is matched at a time */}
    <Route path="/login">
      <Login />
    </Route>
    <Route path="/checkout">
      <Header />
      <Checkout />
    </Route>
    <Route path="/"> {/* Default route for the home page */}
      <Header />
      <Home />
    </Route>
  </Switch>
</Router>
```

7.4 Implementing Firebase Authentication in Login

1. **Import Firebase Authentication:** In `src/components/Login.js`, import `auth` from your Firebase configuration file:

```
import { auth } from "./firebase";
```

2. **Implement `signIn` Function:** Implement the `signIn` function that will be triggered when the user clicks the "Sign In" button:

```
const signIn = (e) => {
  e.preventDefault(); // Prevent the default form submission

  auth
    .signInWithEmailAndPassword(email, password)
    .then((auth) => {
      // Successfully signed in
      history.push("/"); // Redirect to the home page
    })
    .catch((error) => alert(error.message));
};
```

3. **Implement `register` Function:** Implement the `register` function for new user registration:

```
const register = (e) => {
  e.preventDefault();

  auth
    .createUserWithEmailAndPassword(email, password)
    .then((auth) => {
      // Successfully created a new user
      if (auth) {
        history.push("/");
      }
```

```
    })
    .catch((error) => alert(error.message));
};
```

4. **Attach Functions to Buttons:** In the `Login` component's JSX, attach the `signIn` function to the "Sign In" button and the `register` function to the "Create your eShop Account" button.

```
<button type="submit" onClick={signIn} className="login__signInButton">
  Sign In
</button>

<button onClick={register} className="login__registerButton">
  Create your eShop Account
</button>
```

You have now successfully implemented the core functionalities of your e-commerce application! You can start the application by running `npm start`. Test the user authentication, add products to your basket, and proceed to checkout. In the next steps, you can explore additional features like payment integration, order management, and more.