Instruction Sheet: Building a Simple Shoe Showcase Website with React

Objective

Create a simple shoe showcase website using React. You will utilize components, state management with `useState`, and JSX while applying CSS for styling. This project will help you understand the core concepts of React, including passing props, managing state, and using children props effectively.

Project Structure

Your application should consist of the following components:

1. **App.jsx** (Main component)
2. **Header.jsx** (Displays the website title)
3. **ShoeList.jsx** (Displays a list of shoes)
4. **ShoeItem.jsx** (Displays individual shoe details)
5. **Footer.jsx** (Displays footer information)

Steps to Follow

1. Set Up Your React App

Use Create React App to set up your project:
bash
Copy code

```
npm create vite@latest
cd shoe-showcase
npm install
npm run dev
```

2. Create Components

Create the following components in the `src` folder:

- **Header.jsx**
- **ShoeList.jsx**
- **ShoeItem.jsx**
- **Footer.jsx**

3. Implementing the App Component

In `App.jsx`, implement the main structure of your app.

jsx
Copy code

```jsx
import React, { useState } from 'react';
import Header from './Header.jsx';
import ShoeList from './ShoeList.jsx';
import Footer from './Footer.jsx';

function App() {
  const [shoes, setShoes] = useState([
    { id: 1, name: 'Sneaker', price: 50 },
    { id: 2, name: 'Boot', price: 80 },
    { id: 3, name: 'Sandals', price: 30 },
    { id: 4, name: 'Loafers', price: 60 },
  ]);

  return (
    <div className="app">
      <Header />
      <ShoeList shoes={shoes} />
      <Footer />
    </div>
  );
}

export default App;
```

4. Implement the Header Component

In `Header.jsx`, create a simple header.

jsx
Copy code

```jsx
import React from 'react';

function Header() {
  return <h1>Shoe Showcase</h1>;
}

export default Header;
```

5. Implement the ShoeList Component

In `ShoeList.jsx`, use the `ShoeItem` component to display each shoe.

jsx
Copy code

```jsx
import React from 'react';
import ShoeItem from './ShoeItem.jsx';

function ShoeList({ shoes }) {
  return (
    <div className="shoe-list">
      {shoes.map((shoe) => (
        <ShoeItem key={shoe.id} shoe={shoe} />
      ))}
    </div>
  );
}

export default ShoeList;
```

6. Implement the ShoeItem Component

In `ShoeItem.jsx`, display the details of each shoe and use a dynamic CSS class.

jsx
Copy code

```jsx
import React from 'react';

function ShoeItem({ shoe }) {
  const priceClass = shoe.price > 60 ? 'expensive' : 'affordable';

  return (
    <div className={`shoe-item ${priceClass}`}>
      <h2>{shoe.name}</h2>
      <p>Price: ${shoe.price}</p>
    </div>
  );
}

export default ShoeItem;
```

7. Implement the Footer Component

In `Footer.jsx`, create a simple footer.

jsx
Copy code

```jsx
import React from 'react';

function Footer() {
  return <footer>© 2024 Shoe Showcase</footer>;
}

export default Footer;
```

8. Custom CSS Styling

Create a `styles.css` file in the `src` directory and add your custom styles. Make sure to apply dynamic classes in the `ShoeItem` component.

css
Copy code

```css
.shoe-item {
  border: 1px solid #ccc;
  padding: 10px;
  margin: 10px;
}

.expensive {
  background-color: #f8d7da;
}

.affordable {
  background-color: #d4edda;
}

.shoe-list {
  display: flex;
  flex-wrap: wrap;
  justify-content: space-around;
}
```

Key Concepts to Cover

- **Passing Props**: Pass the `shoes` array to the `ShoeList` and `shoe` prop to `ShoeItem`.
- **State Management**: Use `useState` to manage the shoes data.
- **Children Prop**: (Optional) You can modify your `ShoeItem` to accept additional children if needed.
- **Dynamic CSS Classes**: Use conditions to apply different styles based on the shoe price.

Additional Notes

- Feel free to add more styling and functionality as you become more comfortable with React.
- Explore additional features like sorting or filtering shoes based on price or name.
- Addition and deletion of shoes can also be implemented

Good luck with your project!