FE Home Assignment

`Note: The app should work as expected and it shouldnt have any issues.`

Best Practices for Web Development

Do's

1. **Code Structure and Naming Conventions**
    - Ensure proper code structuring by adhering to a consistent folder and file naming convention.
2. **Modular Design**
    - Separate concerns by breaking down code into reusable components and modules.
3. **Testing**
    - Thoroughly test the application across various scenarios and edge cases to identify and rectify bugs.
4. **Responsive Design**
    - Implement responsive design principles using CSS media queries or CSS-in-JS libraries such as styled-components.
5. **Optimized Data Fetching**
    - Optimize data fetching by minimizing unnecessary requests and implementing caching mechanisms.
6. **Effective use of `useEffect`**
    - Refactor the use of `useEffect` hooks to reduce unnecessary re-renders and enhance performance.
7. **State Management**
    - Don't use too many `useState` hooks and avoid using unnecessary states.
8. **TypeScript Typing**
    - Define TypeScript types for all data structures, including products and categories.
9. **UI Libraries**
    - Consider utilizing UI component libraries like Material-UI or Bootstrap for improved consistency and responsiveness.
10. **State and Types Management**
    - Avoid using multiple `useState` hooks unnecessarily.

- Ensure TypeScript types are defined accurately and used consistently throughout the application.

Don'ts

1. **Lack of Structure**
   - Avoid a lack of clear structure and organization in the code.
2. **Neglecting Bugs**
   - Resolve bugs and functionality issues promptly.
3. **Non-Responsive Design**
   - Ensure responsiveness by testing on different screen sizes and devices.
4. **Inefficient Data Handling**
   - Refrain from unnecessary data fetching or inefficient use of `useEffect` hooks.
5. **Inconsistent Typing**
   - Ensure consistency in TypeScript type definitions for different data structures.
6. **Poor CSS Practices**
   - Avoid poor or inconsistent CSS implementation leading to responsiveness issues.
7. **Overcomplication**
   - Don't unnecessarily complicate the application with state management libraries like Redux when simpler solutions suffice.
8. **Maintainability**
   - Maintain code quality and ensure maintainability.