Report: E-commerce Product Listing and Detail Page

1. Implementation Steps

- Product Listing Page:
  - First, I created the product listing page which displays all products fetched from the API. This page includes product names, images price and add to cart button.
  - The page is responsive, meaning it looks good on both desktop and mobile devices.
- Individual Product Detail Page:
  - I set up dynamic routing to show individual product details. When a user clicks on a product, they are taken to a detailed page that displays more information about the product, like description, price, and images dimensions and add to cart button.
- Search Bar:
  - A search bar was added to allow users to search for products by name or category. This was integrated with the API to fetch search results based on the input.
- Filter:

  I added filters to the product listing page. These filters allow users to narrow down the products based on categories like "furniture" or "decor", and other attributes.

2. Challenges and Solutions

- *Challenge 1: API Data Fetching Issues:*
  - Initially, I faced some issues with fetching product data from the API. The API was not responding correctly or the data was being returned in an unexpected format.
  - **Solution:** I worked on debugging the API call, checked the response format, and added error handling to display a user-friendly message if the API failed.
- Challenge 2: Filter Functionality:

- The filters were not updating the products correctly based on user selection. Products were not being filtered as expected, and the page was not re-rendering with the correct filtered results.
- **Solution**: I adjusted the logic to handle filter changes properly by updating the API request based on the selected filter options. I also added state management to track the active filters and trigger a page update with the correct filtered data.

3. Best Practices

- Reusable Components:
  - I created reusable components such as ProductCard.js to display individual product details. This makes the code more organized and easier to maintain.

- API Integration:
  - I ensured that the API was integrated in a clean and efficient way, using async/await to handle asynchronous data fetching.

- Dynamic Routing:
  - I used dynamic routing for the product detail page so that each product has its own unique URL based on the product slug.

- Responsive Design:
  - I used Tailwind CSS to ensure that the pages look good on both desktop and mobile devices. The layout adapts to different screen sizes without breaking.

4. Conclusion

This project successfully implements an e-commerce platform with a product listing page, individual product detail pages, search functionality and filter. I faced a few challenges along the way but was able to solve them through debugging and implementing best practices like reusable components and dynamic routing.