**Project Title: AmazeStore**

**Objective:**

Develop an Amazon-powered online store with a sleek, user-friendly interface that mimics professional e-commerce platforms while showcasing technical depth.

**Core Features:**

1. **Dynamic Product Listings:**
   * Fetch and display live product data from the Amazon API.
   * Implement search and filter functionalities (e.g., by price, category, rating).
   * Use pagination or infinite scrolling for seamless navigation.
2. **Product Details Page:**
   * Display multiple images, detailed specifications, reviews, and pricing trends.
   * Add an interactive gallery for product images.
3. **Shopping Cart & Wishlist:**
   * Allow users to add/remove products.
   * Persist cart data using localStorage or Redux-Persist.
4. **Checkout Simulation:**
   * Include a checkout page with dummy payment and shipping forms.
   * Add order confirmation functionality.
5. **Personalized Recommendations:**
   * Based on user searches or API-provided related items.
6. **Responsive Design:**
   * Ensure the app works seamlessly on mobile, tablet, and desktop.
7. **Pricing Alerts:**
   * Enable users to set alerts for price drops and notify them via email (using a mock service or integration like Firebase).
8. **User Authentication:**
   * Add login/sign-up using OAuth (e.g., Google or Amazon).
   * Allow users to save their wishlist and past searches.

**Pages:**Home My account  
Cart Checkout  
Orders   
Product details   
Store Store search result  
Search result And more…

**Advanced Features (Optional):**

1. **Analytics Dashboard (Admin Only):**
   * Track user activity (most-searched items, wishlist trends, etc.).
   * Implement with tools like Recharts or D3.js.
2. **Multi-Language Support:**
   * Use i18n libraries like react-i18next.
3. **Dark Mode:**
   * Implement with useContext or Redux state.
4. **Accessibility Features:**
   * Ensure WCAG compliance with proper ARIA attributes and testing tools.

**Tech Stack:**

1. **Frontend Framework:**
   * **React + TypeScript**: For type-safe, maintainable code.
2. **State Management:**
   * **Redux Toolkit**: Manage API data, cart state, and user preferences.
3. **Styling:**
   * **I will use Bootstrap for responsiveness and mobile-first design**
4. **API Handling:**
   * Fetch API: To consume the Amazon API.
   * Use React Query for caching and efficient data fetching.
5. **Routing:**
   * **React Router**: For navigation (e.g., Home, Product Details, Cart).
6. **Form Handling:**
   * **Formik** or **React Hook Form**: For checkout forms.
7. **Testing:**
   * **Jest + React Testing Library**: Unit and integration tests.
8. **Deployment:**
   * **Vercel** or **Netlify** for quick, reliable deployment.

**Design Plan (Using Figma):**

* Create mockups for the home page, product details page, cart, and checkout flow.
* Design wireframes for mobile and desktop views.
* Add prototyping for navigation between pages.

**Why:**

1. **Real-World Use Case:**
   * Shows understanding of modern e-commerce patterns and API integration.
2. **Technical Depth:**
   * Highlights expertise in React, Redux, and TypeScript.
   * Demonstrates ability to implement advanced features like analytics and multi-language support.
3. **Design & UX:**
   * Incorporating Figma designs shows attention to detail and collaboration skills.
4. **Performance Optimization:**
   * Use of React Query and code splitting for improved performance.
5. **Future-Ready:**
   * Features like responsive design and accessibility show forward-thinking.

**Next Steps:**

1. **Plan & Design:**
   * Define project scope, draw Figma mockups, and finalize the tech stack.
2. **Develop Core Features:**
   * Implement basic product listing and details pages.
3. **Enhance Functionality:**
   * Add shopping cart, wishlist, and personalized recommendations.
4. **Polish & Test:**
   * Optimize performance, test accessibility, and conduct user testing.
5. **Deploy & Showcase:**
   * Use a custom domain, prepare documentation, and host a demo walkthrough.

With this approach, **AmazeStore** will be a standout project in your portfolio!

**Order:**

1. **Routing**: Set up React Router and define your pages first.
2. **Pages**: Define individual pages as components and link them to routes.
3. **Redux**: Set up Redux after the routing and pages are in place to manage global state.
4. **Components**: Build reusable components for shared UI elements.
5. **Testing**: Add tests for components, Redux actions, and reducers.

By following this structure, your app will be organized, and you’ll be able to easily scale it as new features and complexities are added.