Backend framework:

According to our requirements to build an e-commerce system using React, we should choose using **Node.js with Express.js** as the backend technology. Here are some reasons why this is a solid choice:

# Node.js with Express.js:

1. **Unified Language (JavaScript):**
   * Using the same language (JavaScript) for both the frontend (React) and backend simplifies the development process and allows for code reuse and a consistent development experience.
2. **Performance:**
   * Node.js is built on Chrome's V8 JavaScript engine and is known for its non-blocking, event-driven architecture, which makes it ideal for building high-performance, scalable network applications.
3. **Rich Ecosystem:**
   * With npm, the Node.js package manager, you have access to a vast ecosystem of libraries and tools that can accelerate development and add powerful features to your application.
4. **Real-Time Capabilities:**
   * Node.js excels in handling real-time data and can efficiently manage WebSockets, making it a great choice for features like live notifications, real-time order updates, and chat functionalities.
5. **Scalability:**

* Node.js can handle a large number of simultaneous connections with high throughput, making it suitable for e-commerce platforms that require scalability.

Frontend framework:

**React.js:**

1. **Component-Based Architecture**
   * React's modular design allows for the creation of reusable and isolated components, such as **Profile Management**, **Product Listings**, and **Order Tracking**.
   * Each feature, like Artist Profiles, Customization Requests, or Auction Listings, can be implemented as separate components, making development and maintenance efficient.
2. **Interactive User Interface (UI)**
   * React provides a seamless way to create dynamic and responsive UIs. Features like **real-time auction bidding**, **customization interactions**, and **visitor counters** require efficient state management and updates, which React handles gracefully.
3. **State Management**
   * Using tools like React's Context API or external libraries like Redux, you can manage complex state efficiently. Features such as **real-time order tracking**, **sales analytics**, and **bid updates** require precise and scalable state management.
4. **React Hooks**
   * React hooks like useState and useEffect allow for managing component state and lifecycle events, which are essential for implementing features like **notifications**, **real-time updates**, and **profile customization options**.
5. **Seamless Integration with APIs**
   * React works well with APIs, making it an ideal choice for features like **fetching product details**, **handling auction data**, and **loading artist profiles dynamically**.
6. **Performance Optimization**
   * React’s Virtual DOM optimizes rendering performance, especially for frequently updated elements such as **order statuses**, **auction timers**, and **real-time sales tracking charts**.

# Database Framework:

In our application, we have decided to use **two databases: PostgreSQL** and **Firebase Realtime Database**. This decision aligns with the specific requirements of the system and ensures efficient handling of different types of data.

1. **PostgreSQL:**

PostgreSQL is a powerful, open-source relational database system known for its advanced features, extensibility, and reliability. It is designed to handle structured data and supports complex queries, and data integrity constraints.

**Relational Data Management**:

The application requires a robust database to manage structured data such as user accounts, products, orders, and payment details.

PostgreSQL provides a strong schema enforcement mechanism, ensuring consistency and integrity for critical data.

**Support for Complex Queries**:

PostgreSQL supports advanced querying, indexing, and transactions, making it ideal for generating reports, managing relationships between entities, and handling large datasets.

1. **Firebase Realtime Database:**

Firebase Realtime Database is a cloud-hosted NoSQL database provided by Google. It enables real-time data synchronization across devices and supports a serverless approach.

**Real-Time Capabilities**:

The auction feature in the application requires instant updates for bids and auction status. Firebase’s real-time synchronization ensures a seamless user experience.

**Ease of Integration**:

Firebase provides SDKs for multiple platforms, simplifying the implementation of real-time functionality.

**Minimal Setup and Maintenance**:

Being a serverless database, Firebase eliminates the need for manual server management, reducing development effort.

# 