**Frontend (Presentation Layer)**

* **Framework:** React
* **Components and Features**
* **User Roles:**

**1. Seller:**

* Registration Form: Form to register and agree to terms and conditions.
* Product Posting: Interface to post products for sale, including details such as name, description, price, and images.
* View Inquiries: View inquiries from potential buyers.
* Purchase Premium Listings: Interface to purchase premium listing options.

**2. Buyer:**

* Registration Form: Form to register and manage their account.
* Product Browsing: Interface to browse available products.
* Product Inquiry: Interface to contact sellers or make inquiries about products.
* Cart and Purchase History: Interface to add products to cart and view purchase history.

**3. Administrator:**

* Product Management: Interface to manage product categories and delete expired or inappropriate listings.
* Discount Management: Interface to configure service fee discounts for premium listings.
* User Management: Manage user accounts and handle reports.
* **Common Features:**
* Navigation and Layout: Uniform design across all pages.
* Authentication: Token-based authentication for users.
* Authorization: Role-based access control.
* Validation: Client-side validation for user input.

**Backend (Business Logic Layer)**

* Framework: ASP.NET Core REST API
* Key Services and Endpoints
* **Authentication and Authorization:**
  + Token Generation and Validation: Implement secure token-based authentication.
  + Role-Based Access Control: Ensure proper role-based access control.
* **Seller Services:**
* POST /api/sellers/register: Register seller.
* POST /api/products: Post product for sale.
* GET /api/products/{productId}/inquiries: View inquiries for a product.
* POST /api/products/{productId}/premium: Purchase premium listing for a product.
* **Buyer Services:**
* POST /api/buyers/register: Register buyer.
* GET /api/products: Browse available products.
* POST /api/products/{productId}/inquire: Make an inquiry about a product.
* POST /api/cart: Add product to cart.
* GET /api/buyers/{buyerId}/purchase-history: View purchase history.
* **Administrator Services:**
* PUT /api/products/{productId}/category: Manage product categories.
* DELETE /api/products/{productId}: Delete expired or inappropriate product listings.
* POST /api/discounts/configure: Configure service fee discounts.
* GET /api/users: Manage user accounts.
* **Common Services:**
* GET /api/company/profile: Get company profile information.
* GET /api/company/contact: Get company contact details.
* **Session Tracking:**
  + Middleware to track product posting and purchasing activities.
* **Payment Handling:**
  + Integration with payment services for purchasing premium listings.

**Database Layer (Data Access Layer)**

* Database Options: SQL Server or MongoDB
* Key Tables/Collections
* **Users:**
* Sellers: Store seller details.
* Buyers: Store buyer details.
* **Products:**
* Product Listings: Store product details.
* Categories: Store product categories.
* Inquiries: Store buyer inquiries about products.
* Premium Listings: Store information about premium listings.
* **Sessions:**
  + Activity Tracking: Track product posting and purchasing activities per session.
* **Payments:**
  + Premium Listings: Store information about premium listing purchases.
  + Discounts: Store discount configurations.
* **Configuration:**
  + Service Fee Discounts: Store service fee discount settings (using XML configuration files).

**Additional Considerations**

* API Documentation: Provide clear documentation for all API endpoints.
* Security: Ensure secure data handling and storage, especially for personal and payment information.
* Scalability: Design the system to handle increased load as the application grows.
* User Experience: Focus on intuitive design and easy navigation for all user roles.

**Architecture Diagram**

* **A typical 3-tier architecture with:**
* Presentation Layer (View): React-based frontend.
* Business Logic Layer: ASP.NET Core REST API handling the business logic.
* Data Access Layer: SQL Server or MongoDB for data storage and retrieval.

By implementing these features and functions across the frontend, backend, and database layers, the app will be well-equipped to facilitate the buying and selling of products in an OLX-like system.