**Amazon Clone Project Overview**

This project is a full-stack e-commerce application that replicates core Amazon functionality using modern web technologies. Here's a comprehensive explanation you can use for interviews:

**Project Architecture**

This Amazon clone uses a **Node.js and Express backend** with a **SQLite database** for data persistence, and a **vanilla JavaScript frontend** with responsive HTML/CSS. The application follows the **MVC (Model-View-Controller)** pattern for clean code organization.

**Key Features**

**1. User Authentication & Authorization**

* **Secure Registration System**: Users can create accounts with email verification
* **Login System**: Password authentication with bcrypt encryption for security
* **Session Management**: Using express-session with SQLite storage
* **Role-based Authorization**: Separate user and admin privileges
* **User Profiles**: Personal information management and order history

**2. Database Design**

* **SQLite Database**: Lightweight, serverless database solution
* **Structured Schema**: Separate tables for users, products, orders, etc.
* **Data Relationships**: Foreign key relationships between tables
* **Password Security**: Bcrypt hashing for secure password storage

**3. Admin Dashboard**

* **User Management**: View and manage user accounts
* **Analytics**: Track statistics like new registrations and active users
* **Admin-only Access**: Protected routes with middleware security

**4. Frontend Implementation**

* **Responsive Design**: Fully responsive UI across devices
* **Modern UI**: Replicates Amazon's familiar interface
* **Dynamic Content**: JavaScript for interactive elements
* **State Management**: Client-side session state with authentication status

**5. Security Features**

* **Authentication Middleware**: Protects private routes
* **Password Hashing**: Secure password storage
* **Session Management**: Secure cookie-based sessions
* **Validation**: Input validation for all user-submitted data

**Technical Challenges & Solutions**

**Authentication Flow**

I implemented a complete authentication system where:

* Users register with email/password
* Passwords are securely hashed with bcrypt
* Login creates a session stored in SQLite
* Navigation dynamically changes based on login status
* Admin users get additional access

**Dynamic UI Components**

The UI intelligently adapts to the user's authentication state:

* When logged out: Shows login/register options
* When logged in: Shows profile and logout options
* For admin users: Shows additional admin dashboard access

**Database Security**

I prioritized security by:

* Using prepared statements to prevent SQL injection
* Implementing proper password hashing
* Creating unique constraints on email to prevent duplicates
* Storing session data securely

**Future Enhancements**

* **Product Management**: Complete CRUD for products in admin panel
* **Shopping Cart Functionality**: Full implementation of cart and checkout
* **Payment Integration**: Adding payment processing options
* **Order Tracking**: Order history and status updates
* **User Reviews**: Product rating and review system
* **Enhanced Security**: Adding 2FA for admin accounts

**Technologies Used**

* **Backend**: Node.js, Express.js
* **Database**: SQLite with sqlite3 driver
* **Authentication**: bcrypt, express-session
* **Frontend**: HTML5, CSS3, JavaScript (ES6+)
* **UI Components**: Custom CSS, Font Awesome for icons
* **API Architecture**: RESTful API endpoints

This project demonstrates my ability to build a complete full-stack application with secure authentication, database management, and responsive UI design—core skills that are essential for modern web development.