# **E-commerce Application High-Level Design (HLD)**

## **1. Introduction**

This document outlines the high-level design for the E-commerce application, covering the overall architecture, data flow, key components, and integration points. It is built using Angular, NgRx, Bootstrap, Node.js, and MongoDB to provide a scalable, secure, and user-friendly shopping experience.

## **2. System Overview**

The E-commerce application is a full-stack web application with separate frontend and backend services. It provides a seamless shopping experience, including product browsing, cart management, order processing, and user account management.

### **2.1 Key Features**

* User Authentication and Authorization
* Product Catalog and Search
* Shopping Cart Management
* Order Placement and Tracking
* Payment Integration
* Wishlist and Review System
* Admin Dashboard for Product and Order Management

## **3. Architecture**

The application follows a component-based architecture with a RESTful API backend and a reactive frontend using NgRx for state management.

### **3.1 Frontend Architecture (Angular + NgRx)**

* Component-Based Structure
* State Management with NgRx (Actions, Reducers, Effects, Selectors)
* Responsive Design with Bootstrap
* Lazy Loading for Module Optimization

### **3.2 Backend Architecture (Node.js + Express + MongoDB)**

* RESTful API Design
* Middleware for Authentication and Authorization
* Database Interaction with Mongoose
* Token-Based Authentication (JWT)

### **3.3 Data Flow**

1. User requests a product list -> API fetches data from MongoDB -> Data is stored in NgRx store -> UI components render the data.
2. Add to Cart -> Dispatch action to NgRx store -> Store updates the cart state.
3. Place Order -> API call to backend -> Order saved in MongoDB -> Confirmation email sent to user.

## **4. Database Design**

### **4.1 Schema**

* **Users:** User details, hashed passwords, roles (Admin, Customer)
* **Products:** Name, price, category, description, images, stock
* **Orders:** Order ID, user ID, product IDs, total amount, status, timestamps
* **Reviews:** User ID, product ID, rating, comments, timestamps
* **Carts:** User ID, product IDs, quantities, timestamps

## **5. Security and Authentication**

* Role-based access control (RBAC) for Admin and Customer roles
* JWT for secure API access
* Password hashing with bcrypt
* HTTPS and CORS configuration

## **6. API Design**

* **User Management:** Login, Signup, Password Reset
* **Product Management:** Add, Update, Delete, Fetch Products
* **Order Management:** Create Order, Track Order, Cancel Order
* **Review System:** Add Review, Fetch Reviews

## **7. Error Handling and Logging**

* Centralized error handling with middleware
* Logging with Winston or Morgan for HTTP requests

## **8. Testing Strategy**

* Unit testing with Jasmine/Karma (Frontend)
* Integration testing with Jest (Backend)
* API testing with Postman or Insomnia

## **9. Deployment and Scaling**

* CI/CD pipeline with GitHub Actions
* Docker for containerization
* Cloud deployment on AWS, Azure, or GCP

## **10. Future Enhancements**

* Mobile App Integration
* Real-time Chat for Customer Support
* AI-based Product Recommendations
* Multi-language and Multi-currency Support