E-Commerce Backend Development Documentation

Prepared for: Rohan Reddy
Stack: Node.js, Express.js, MongoDB, WebSockets, AI-powered search (Vector Search)

---

1. Project Setup

Objective: Establish a solid foundation for the project.

Steps: 1. Initialize Node.js project using `npm init`. 2. Install required packages: Express.js, Mongoose, bcrypt, jsonwebtoken, cors, dotenv, socket.io, OpenAI/vector search library. 3. Setup `.env` for environment variables. 4. Project folder structure:

```
/project-root
│
├── /controllers
├── /models
├── /routes
├── /middleware
├── /utils
├── /config
└── server.js
```

---

2. Database Schemas

2.1 User Schema

```
const UserSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: { type: String, enum: ['user', 'admin'], default: 'user' },
  createdAt: { type: Date, default: Date.now }
});
```

2.2 Product Schema

```
const ProductSchema = new mongoose.Schema({
  name: String,
  description: String,
  price: Number,
  stock: Number,
  category: String,
  images: [String],
  embedding: [Number], // For AI search
  createdAt: { type: Date, default: Date.now }
});
```

2.3 Order Schema

```
const OrderSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  products: [{ productId: mongoose.Schema.Types.ObjectId, quantity: Number }],
  totalAmount: Number,
  status: { type: String, enum: ['Pending', 'Shipped', 'Delivered',
'Cancelled'], default: 'Pending' },
  createdAt: { type: Date, default: Date.now }
});
```

2.4 Wishlist Schema

```
const WishlistSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  products: [{ type: mongoose.Schema.Types.ObjectId, ref: 'Product' }]
});
```

2.5 Review Schema

```
const ReviewSchema = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  productId: { type: mongoose.Schema.Types.ObjectId, ref: 'Product' },
  rating: { type: Number, min: 1, max: 5 },
  comment: String,
  createdAt: { type: Date, default: Date.now }
});
```

---

3. API Endpoints

3.1 User Routes

Method	Route	Description
POST	/api/users/register	Register new user
POST	/api/users/login	User login and get JWT
GET	/api/users/me	Get current user profile (JWT required)

3.2 Product Routes

Method	Route	Description
POST	/api/products	Admin: create product
GET	/api/products	Get all products
GET	/api/products/:id	Get product by ID
PUT	/api/products/:id	Admin: update product
DELETE	/api/products/:id	Admin: delete product

3.3 Order Routes

Method	Route	Description
POST	/api/orders	User: create order
GET	/api/orders	User: get user orders
GET	/api/orders/all	Admin: get all orders
PUT	/api/orders/:id	Admin: update order status

3.4 Wishlist Routes

Method	Route	Description
POST	/api/wishlist	Add product to wishlist
GET	/api/wishlist	Get user wishlist
DELETE	/api/wishlist/:productId	Remove product from wishlist

3.5 Review Routes

Method	Route	Description
POST	/api/reviews	Add review for product
GET	/api/reviews/:productId	Get reviews for product

3.6 Search Route

Method	Route	Description
GET	/api/search?q=keyword	Semantic product search using vector embeddings

3.7 Chat (WebSocket)

- Users connect via Socket.io client.
- Private room is created per user-support interaction.
- Messages exchanged in real-time.
- Optional: store chat history in MongoDB.

---

4. Security & Middleware

- JWT Authentication & Role-based Authorization
- Input validation using `express-validator`
- Rate limiting middleware
- CORS configuration
- Global error handling middleware
- Password hashing with bcrypt

---

5. AI-Powered Search

1. Generate embeddings for each product using OpenAI or similar API.
2. Store embeddings in MongoDB.
3. Search endpoint converts query into embedding and returns top matches.

---

6. Example Frontend Integration

WebSocket:

```
const socket = io('http://localhost:5000');
socket.emit('joinRoom', { userId });
socket.on('message', msg => console.log(msg));
```

Vector Search:

```
fetch('/api/search?q=shoes')
  .then(res => res.json())
  .then(data => console.log(data));
```

---

7. Suggested Timeline

Phase	Duration	Tasks
Setup & Auth	3-4 days	Project setup, User model, Registration/Login, JWT
Product Management	3 days	CRUD, images, stock
Order Management	2-3 days	Orders, status updates, user order history
Chat Integration	2 days	Socket.io setup, message handling
AI Search	2-3 days	Embeddings, vector search API
Optional Features	3-4 days	Wishlist, Reviews, Emails
Testing & Deployment	2 days	Postman tests, frontend integration, deploy

---

8. Deliverables

1. Fully functional backend with all APIs.
2. Organized folder structure.
3. Database schemas and documentation.
4. Example frontend integrations for chat and search.
5. Security best practices and input validation implemented.
6. Professional PDF documentation for reference.