MERN Stack E-commerce Backend Development Prompt

I want to build a comprehensive e-commerce backend using the MERN stack (MongoDB, Express, React, Node.js) with TypeScript. The goal is to create a scalable platform similar to Shopify (not as large as Amazon) with all essential e-commerce functionalities.

Tech Stack Specifications

- **Backend**: Node.js with Express
- **Database**: MongoDB with Mongoose ODM
- **Language**: TypeScript
- **Module System**: ES Modules (use `"type": "module"` in package.json)
- **Authentication**: JWT with refresh tokens
- **Validation**: Zod for type validation
- **File Storage**: AWS S3 for product images
- **Payment Processing**: Stripe API
- **Email Service**: Nodemailer or SendGrid
- **Testing**: Jest with Supertest

Folder Structure

```
ecommerce-backend/
├── src/
│   ├── config/               # Configuration files
│   │   ├── database.ts       # MongoDB connection setup
│   │   ├── environment.ts    # Environment variables
│   │   └── constants.ts      # App constants
│   ├── controllers/          # Request handlers
│   │   ├── auth.controller.ts
│   │   ├── product.controller.ts
│   │   ├── order.controller.ts
│   │   ├── user.controller.ts
│   │   ├── cart.controller.ts
│   │   └── ...
│   ├── middleware/           # Custom middleware
│   │   ├── auth.middleware.ts  # Authentication middleware
│   │   ├── validation.middleware.ts  # Request validation
│   │   ├── error.middleware.ts  # Error handling
│   │   └── ...
│   ├── models/               # Database schemas
│   │   ├── User.model.ts
│   │   ├── Product.model.ts
│   │   ├── Order.model.ts
│   │   ├── Cart.model.ts
│   │   ├── Category.model.ts
│   │   ├── Review.model.ts
│   │   └── ...
│   ├── routes/               # API routes
│   │   ├── auth.routes.ts
│   │   ├── product.routes.ts
│   │   ├── order.routes.ts
│   │   ├── user.routes.ts
│   │   ├── cart.routes.ts
│   │   └── ...
│   ├── services/             # Business logic and external service integrations
│   │   ├── auth.service.ts
│   │   ├── product.service.ts
│   │   ├── payment.service.ts
│   │   ├── email.service.ts
│   │   └── ...
│   ├── types/                # TypeScript type definitions
│   │   ├── user.types.ts
│   │   ├── product.types.ts
│   │   ├── order.types.ts
│   │   └── ...
│   ├── utils/                # Helper functions
│   │   ├── logger.ts
│   │   ├── apiResponse.ts
│   │   ├── fileUpload.ts
│   │   └── ...
│   ├── app.ts                  # Express app configuration
│   ├── index.ts                # Main entry point
│   ├── server.ts               # Server setup
│   └── constants.ts            # Global constants
├── tests/                      # Test files mirroring src structure
├── .env                        # Environment variables (gitignored)
├── .env.example                # Example env vars for documentation
├── tsconfig.json               # TypeScript configuration
├── jest.config.js              # Jest configuration
├── package.json                # With "type": "module" for ES modules
└── README.md
```

Core Features to Implement

User Management

- User registration and authentication (JWT)
- Role-based access control (customer, admin, vendor)
- User profiles with avatar
- Address management (multiple shipping addresses)
- Password reset functionality
- Order history
- Wishlist functionality

Product Management

- Complete CRUD operations for products
- Product variants (size, color, etc.)
- Product categories and subcategories
- Product search with filters
- Product reviews and ratings
- Image upload and management
- Inventory tracking
- Related products functionality

Shopping Cart

- Add/remove items
- Update quantities

- Save for later
- Persistent cart (for logged-in users)
- Guest cart functionality

Order Processing

- Cart to order conversion
- Order status tracking (pending, processing, shipped, delivered)
- Order confirmation emails
- Invoice generation
- Order history
- Returns/refunds management

Payment Integration

- Stripe integration for card payments
- PayPal integration (optional)
- Multiple payment methods
- Payment status tracking
- Refund processing
- Payment security

Admin Features

- Dashboard with sales analytics
- Order management
- Product management
- User management
- Inventory management
- Sales reports

Additional Features

- Coupon and discount system
- Wishlist functionality
- Newsletter subscription
- Email notifications
- SEO-friendly product URLs
- Recently viewed products

API Endpoints Structure

Authentication

- POST `/api/auth/register` - Register new user
- POST `/api/auth/login` - User login
- POST `/api/auth/refresh-token` - Refresh access token
- POST `/api/auth/forgot-password` - Password reset request
- POST `/api/auth/reset-password` - Reset password with token
- GET `/api/auth/me` - Get current user

Products

- GET `/api/products` - List all products with pagination
- GET `/api/products/:id` - Get product details
- POST `/api/products` - Create new product (admin)
- PUT `/api/products/:id` - Update product (admin)
- DELETE `/api/products/:id` - Delete product (admin)
- GET `/api/products/search` - Search products
- GET `/api/products/categories` - List all categories
- GET `/api/products/category/:slug` - Get products by category

Orders

- GET `/api/orders` - List user orders
- GET `/api/orders/:id` - Get order details
- POST `/api/orders` - Create new order
- PUT `/api/orders/:id/status` - Update order status (admin)
- GET `/api/orders/:id/invoice` - Generate order invoice

Cart

- GET `/api/cart` - Get user cart
- POST `/api/cart` - Add item to cart
- PUT `/api/cart/:itemId` - Update cart item
- DELETE `/api/cart/:itemId` - Remove item from cart
- DELETE `/api/cart` - Clear cart

Users

- GET `/api/users/profile` - Get user profile
- PUT `/api/users/profile` - Update user profile
- GET `/api/users/addresses` - Get user addresses
- POST `/api/users/addresses` - Add new address
- PUT `/api/users/addresses/:id` - Update address
- DELETE `/api/users/addresses/:id` - Delete address

Reviews

- GET `/api/products/:id/reviews` - Get product reviews
- POST `/api/products/:id/reviews` - Add product review
- PUT `/api/reviews/:id` - Update review
- DELETE `/api/reviews/:id` - Delete review

Development Challenges to Address

1. **Database Design** - Creating efficient schemas for product variants, orders, and inventory
2. **Authentication & Security** - Implementing secure user authentication and authorization
3. **Payment Processing** - Securely handling payments and meeting compliance requirements
4. **Performance Optimization** - Ensuring fast responses for product listings and searches
5. **Image Management** - Handling product image uploads, storage, and optimization
6. **Order Workflow** - Creating a robust order processing pipeline
7. **Testing Strategy** - Implementing comprehensive unit and integration tests
8. **Deployment Strategy** - Setting up CI/CD and production environment

Implementation Priorities

1. **Phase 1: Core Infrastructure** - Database setup, authentication, basic product management
2. **Phase 2: Product System** - Complete product CRUD, categories, search
3. **Phase 3: User & Cart System** - User profiles, cart functionality
4. **Phase 4: Order Processing** - Order creation, payment integration
5. **Phase 5: Advanced Features** - Reviews, discounts, wishlist, etc.

Implementation Requirements

- Use the latest technologies and packages available as of 2025
- Follow industry-standard coding practices and patterns

- Implement proper error handling and logging
- Write clean, maintainable code with comprehensive comments
- Use ES Modules throughout the project
- Apply proper TypeScript types and interfaces
- Implement complete input validation
- Follow RESTful API best practices
- Add comprehensive API documentation (Swagger/OpenAPI)
- Use environment variables for all sensitive configurations
- Implement proper database indexing for performance

Please help me design and implement this e-commerce backend, focusing on best practices for scalability, security, and maintainability. I'm particularly interested in proper TypeScript implementation, clean architecture, and robust error handling.