**Full Stack Development with MERN**

**Project Documentation format**

**1. Introduction**

**• Project Title:** ShopSmart: Your On-Demand Shopping Solution **• Team Members and Roles:**

| **Name** | **Role** |
| --- | --- |
| Hemanth | Full-Stack Developer (Frontend & Backend) |
| Venkatesh | Full-Stack Developer (Frontend & Backend) |
| Sai chandhan | Full-Stack Developer (Backend) |
| Manikanta | Full-Stack Developer (Frontend) |

**2. Project Overview  
• Purpose:**ShopSmart is a comprehensive multi-role e-commerce and seller management platform designed to:

* For Customers: Browse sellers, explore products, place orders, track delivery, and review products
* For Sellers: Manage profiles, products, staff, orders, subscriptions, and analytics
* For Staff: Process orders, update statuses, and assist with order fulfillment
* For Admins: Oversee the platform, approve sellers, manage users, and view analytics

**Goals**

1. Provide a seamless ordering experience for end-users
2. Empower sellers with tools for digital transformation
3. Enable efficient order fulfillment through staff workflows
4. Implement subscription-based monetization model
5. Ensure secure payments and data integrity
6. Scale to support thousands of concurrent users

**Features:  
User Features:**

* Email/phone-based registration and verification
* Browse sellers and product catalogs
* Add items to cart with quantity management
* Place orders with delivery address selection
* Razorpay payment integration with signature verification
* Real-time order status tracking
* Submit product reviews and ratings
* Profile management with image upload

**Seller Features:**

* Seller registration and admin approval workflow
* Product management (CRUD operations)
* Toggle product availability
* View incoming orders and update status
* Staff invite system with token-based registration
* Staff approval workflow
* Subscription plan management (FREE/BASIC/PREMIUM)
* Seller analytics and revenue tracking

**Staff Features:**

* Token-based registration via seller invite
* View assigned orders
* Request order status changes (subject to owner approval)
* Profile management

**Admin Features:**

* User management (view, suspend, delete)
* Seller approval/rejection workflow
* Order and product management across all sellers
* Revenue analytics by seller and subscription plan
* Order distribution and status analytics
* Staff management

**System Features:**

* Role-based access control (RBAC): CUSTOMER/SELLER/STAFF/ADMIN
* JWT-based authentication
* Email verification (SendGrid)
* SMS OTP verification (Twilio)
* Image upload for profiles and sellers (max 5MB)
* Subscription limits enforced by plan
* Order status lifecycle tracking
* Payment tracking and reconciliation

**3. Architecture**

**3.1 Frontend Architecture (Angular + Tailwind)  
Technology Stack:**

* Framework: Angular 15+
* Styling: Tailwind CSS
* HTTP Client: Axios
* State Management: Angular Services
* Routing: Angular Router

**Architecture Pattern:**

* Component-Based: Modular, reusable components
* Service-Based Auth: AuthService manages user authentication state globally
* Protected Routes: Route guards enforce role-based access
* API Layer: Centralized Axios instance for all backend calls

**Key Components:**frontend/  
├── src/  
│ ├── components/ # Reusable UI components  
│ ├── pages/ # Page components by role  
│ ├── services/ # Auth, API, and business logic  
│ ├── app.module.ts # Main app module  
│ └── main.ts # Entry point

**Data Flow:**User Input → Component → Axios API Call → Backend → MongoDB  
↓  
Auth Service (state)  
↓  
Route Guards (role check)

**3.2 Backend Architecture (Node.js + Express)  
Technology Stack:**

* Runtime: Node.js 18+
* Framework: Express.js
* Database: MongoDB with Mongoose ODM
* Authentication: JWT (JSON Web Tokens)
* Password Hashing: bcryptjs
* Email: SendGrid
* SMS: Twilio
* Payments: Razorpay
* File Upload: Express-based file handling

**Architecture Pattern:**

* MVC (Modified): Models, Controllers, Routes, Middleware layers
* Service-Oriented: Business logic in controllers and utilities
* Middleware Chain: Authentication, authorization, error handling
* Modular Routes: Separate route files by feature (auth, orders, sellers, etc.)

**Key Layers:**  
Request  
↓  
CORS & Body Parser Middleware  
↓  
Routes (authRoutes, cartRoutes, orderRoutes, etc.)  
↓  
Authentication Middleware (JWT verification)  
↓  
Authorization Middleware (Role-based check)  
↓  
Controllers (Business logic)  
↓  
Service Layer (Data queries, external API calls)  
↓  
Mongoose Models (Database schema & validation)  
↓  
MongoDB Database  
↓  
Response (JSON)

**Core Modules:**

| Module | Files | Purpose |
| --- | --- | --- |
| Auth | authController, authRoutes | User registration, login, JWT generation |
| Seller | sellerController, sellerRoutes | Seller CRUD, approvals, profiles |
| Product | productController, productRoutes | Product management, CRUD |
| Cart | cartController, cartRoutes | Cart operations (add, remove, view) |
| Order | orderController, orderRoutes | Order placement, status updates, tracking |
| Payment | paymentController, paymentRoutes | Razorpay integration, signature verification |
| Review | reviewController, reviewRoutes | Product reviews, ratings, aggregation |
| Staff | staffController, staffRoutes | Staff invites, approval, status requests |
| Subscription | subscriptionController, subscriptionRoutes | Plans, subscription management, limits |
| Admin | adminController, adminRoutes | User management, analytics, approvals |
| Verification | verificationController, verificationRoutes | Email/SMS OTP verification |

**Middleware:**

* authMiddleware.js – JWT verification and user extraction
* subscriptionMiddleware.js – Feature limit enforcement based on subscription plan

**3.3 Database Architecture (MongoDB + Mongoose)**  
Database Name: shopsmart  
Collections & Schema Overview:

| Collection | Purpose | Key Fields |
| --- | --- | --- |
| users | User accounts (customers, sellers, staff, admins) | email, password, userType, emailVerified, phoneVerified, profileImage |
| sellers | Seller profiles and metadata | ownerId, title, address, category, status (pending/approved), profileImage |
| products | Product items | sellerId, title, price, category, description, mainImg, isAvailable |
| carts | Shopping carts (user-based) | userId, sellerId, items[], totalAmount |
| orders | Customer orders | userId, sellerId, items[], totalAmount, status, paymentId, paymentStatus |
| payments | Payment records | userId, orderId, sellerId, amount, paymentMethod, gatewayPaymentId, paymentStatus |
| reviews | Product reviews and ratings | productId, userId, rating (1-5), comment, createdAt |
| staffinvites | Staff invitation tokens | email, sellerId, token, status (pending/accepted), expiresAt |
| subscriptions | Seller subscriptions | userId (seller owner), plan (FREE/BASIC/PREMIUM), status, billingCycleStart, billingCycleEnd |
| orderstatusrequests | Staff status change requests | orderId, staffId, requestedStatus, approvalStatus (pending/approved/rejected), createdAt |

**Data Relationships:**User (owner) ←→ Seller ←→ Product  
↓  
Order ← Cart  
↓  
Payment  
↓  
Review  
Seller ←→ StaffInvite → User (staff)  
↓  
Subscription  
↓  
OrderStatusRequest

**Indexes:**

* users.email – Unique index for login
* staffinvites.token – Unique index for invite validation
* reviews.(productId, userId) – Compound unique index to prevent duplicate reviews
* Order & payment records indexed by date for analytics queries

**4. Setup Instructions**

**4.1 Prerequisites**  
Ensure the following software is installed:

| Software | Version | Download |
| --- | --- | --- |
| Node.js | 18+ (LTS) | [https://nodejs.org](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html) |
| npm or yarn | 8+ | Included with Node.js |
| MongoDB Atlas | Cloud | [https://www.mongodb.com/atlas](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html) |
| Git | Latest | [https://git-scm.com](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html) |
| Postman (optional) | Latest | [https://postman.com](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html) |

**API Keys Required:**

* Razorpay: RAZORPAY\_KEY\_ID, RAZORPAY\_KEY\_SECRET
* SendGrid: SENDGRID\_API\_KEY, SENDGRID\_SENDER
* Twilio: TWILIO\_ACCOUNT\_SID, TWILIO\_AUTH\_TOKEN, TWILIO\_PHONE\_NUMBER

**4.2 Installation & Setup**  
**Step 1: Clone the Repository**  
git clone [https://github.com/yourusername/shopsmart.git](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)  
cd shopsmart

**Step 2: Backend Setup**cd backend  
npm install  
cp .env.example .env  
Edit .env with your configuration

**Backend .env Template:**

**vulnerability**

**Step 3: Frontend Setup**  
cd ../frontend  
npm install  
cp .env.example .env  
Edit .env

Frontend .env Template:

**Step 4: MongoDB Atlas Setup**

* Go to [https://www.mongodb.com/atlas](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Create a free cluster
* Create a database user
* Whitelist your IP
* Get connection string and add to backend .env

Step 5: Configure API Keys

* Razorpay: [https://razorpay.com](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* SendGrid: [https://sendgrid.com](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Twilio: [https://www.twilio.com](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

5. Folder Structure

5.1 Client (Frontend) Structure  
frontend/  
├── public/  
├── src/  
│ ├── api/  
│ ├── components/  
│ ├── services/  
│ ├── pages/  
│ ├── app.module.ts  
│ └── main.ts  
├── index.html  
├── package.json  
├── vite.config.js  
├── tailwind.config.js  
├── postcss.config.js  
└── [README.md](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

**5.2 Server (Backend) Structure  
backend/**  
├── src/  
│ ├── app.js  
│ ├── config/  
│ ├── controllers/  
│ ├── middlewares/  
│ ├── models/  
│ ├── routes/  
│ ├── seeds/  
│ └── utils/  
├── uploads/  
│ └── profiles/  
├── server.js  
├── package.json  
├── .env.example  
└── [README.md](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

**6. Running the Application**

**6.1 Start Backend Server**  
cd backend  
npm run dev  
npm start  
**Expected Output:**  
Server running on [http://localhost:5000](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)  
MongoDB connected successfully

**6.2 Start Frontend Server**cd frontend  
npm run dev  
npm run build  
npm run preview  
Expected Output:  
Angular ready in 500 ms  
Local: [http://localhost:4200/](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

**6.3 Verify Setup**  
Backend Health Check:  
curl [http://localhost:5000/api/health](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)  
Frontend Access: Open [http://localhost:4200](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html) in your browser  
Seed Sample Data (Optional):  
cd backend  
npm run seed:products

**7. API Documentation**

**API Base URL**

* Development: [http://localhost:5000](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Production: [Your deployed backend URL]

Authentication  
All protected endpoints require:  
Authorization: Bearer <JWT\_TOKEN>

Core API Endpoints  
(Adapted for ShopSmart: see original for endpoint details)

**8. Authentication & Authorization**

**Authentication Strategy  
Method: JWT (JSON Web Tokens)**  
**Flow:**

1. User Registration/Login
2. Backend validates credentials
3. If valid, generates JWT token with user info & role
4. Token sent to frontend (stored in localStorage/sessionStorage)
5. Frontend includes token in Authorization header for all protected requests
6. Backend middleware verifies token on every protected route

**Token Structure  
JWT Payload:**

* Token Expiry: 24 hours (configurable in backend
* Role-Based Access Control (RBAC)  
  **Roles**:
* CUSTOMER – End users
* SELLER – Seller owners
* STAFF – Seller staff members
* ADMIN – Platform administrators

Session Management

* Frontend: JWT token stored in browser localStorage
* Token Refresh: On login, token valid for 24 hours; no refresh token implemented (MVP)
* Logout: Clear token from localStorage on frontend

**9. User Interface**

**Key UI Features**

1. Homepage / Landing Page

* Seller discovery with filters
* Search functionality
* Quick access to login/register

1. User Dashboard

* Order history
* Active order tracking with status updates
* User profile management
* Address book

1. Seller Dashboard

* Product management
* Real-time order queue
* Order status updates
* Staff management
* Analytics dashboard
* Subscription management

1. Admin Dashboard

* User management (view, suspend)
* Seller approvals
* Order oversight
* Revenue analytics (charts, graphs)
* Product management across sellers

1. Responsive Design

* Mobile-first approach using Tailwind CSS
* Works on iOS (Safari) and Android (Chrome)
* Desktop optimized for 1920x1080+

**Color Scheme**

* Primary: Orange (#FF6B35)
* Secondary: Blue (#004E89)
* Accent: Green (#1FB578)
* Neutral: Gray/White (#F5F5F5, #FFFFFF)

**UI Components**

* Navigation bar with role-based menu
* Modal dialogs for confirmations
* Toast notifications for actions
* Loading spinners
* Product cards with ratings
* Order status badges
* Pagination for lists

**10. Testing**

**Testing Strategy  
Levels:**

1. Unit Testing: Individual functions (future enhancement)
2. Integration Testing: API endpoints with database
3. End-to-End Testing: Complete user workflows
4. UAT Testing: User acceptance by stakeholders

Testing Tools

| Tool | Purpose | Status |
| --- | --- | --- |
| Postman | API endpoint testing | ✅ In Use |
| Jest | Unit testing (planned) | 📋 Backlog |
| Angular Testing | Component testing | 📋 Backlog |
| Selenium/Playwright | E2E testing (planned) | 📋 Backlog |
| JMeter | Load testing (planned) | 📋 Backlog |

Current Test Coverage  
**Manual Testing Completed:**

* ✅ 101 test cases across 12 feature modules
* ✅ 95% pass rate (96 passed, 3 failed post-fix, 2 not tested)
* ✅ All critical/high-severity bugs resolved
* ✅ Security testing (CORS, JWT, XSS, SQL injection)
* ✅ Payment integration testing with Razorpay sandbox

**Test Case Breakdown:**

* Authentication: 15/15 ✅
* Seller Management: 10/10 ✅
* Product Management: 11/12 (1 edge case pending)
* Cart & Checkout: 8/8 ✅
* Order Management: 9/10 (real-time sync v1.1 feature)
* Payments: 6/6 ✅
* Reviews: 6/6 ✅
* Staff: 6/6 ✅
* Subscriptions: 6/6 ✅
* Admin: 7/8 (analytics with large dataset v1.1)
* Notifications: 5/6 (SMS under high load fixed)
* Security & Performance: 8/8 ✅

See: TESTING\_PLAN\_REPORT.md for detailed test case documentation

**11. Screenshots & Demo**

**Application Flow Screenshots  
User Journey:**

1. Registration → Email Verification → Login
2. Browse Sellers → View Products → Add to Cart
3. Checkout → Razorpay Payment → Order Confirmation
4. Track Order Status → Delivery → Leave Review

**Seller Journey:**

1. Register → Admin Approval → Add Products
2. View Orders → Update Status → Staffing
3. View Analytics → Manage Subscription

**Features Showcase**

| Feature | Screenshot | Description |
| --- | --- | --- |
| Product Discovery | [screenshot] | Browse sellers and products with ratings |
| Cart Management | [screenshot] | Add/remove items, view total, checkout |
| Order Tracking | [screenshot] | Real-time order status (PLACED → DELIVERED) |
| Seller Dashboard | [screenshot] | Manage products, view orders, staff mgmt |
| Admin Analytics | [screenshot] | Revenue charts, order distribution, trends |
| Payment Gateway | [screenshot] | Razorpay modal, secure checkout |

**Live Demo**

* Frontend: [https://shopsmart-frontend.netlify.app/](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Backend API: [https://shopsmart-backend.onrender.com](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Postman Collection: [Link to Postman workspace]

12. Known Issues & Limitations

Known Issues

| Issue | Severity | Status | Workaround |
| --- | --- | --- | --- |
| Real-time order status updates require refresh | Medium | Backlog | Manual refresh or polling |
| SMS delivery delay during peak load | Low | Fixed | Message queue with retry logic |
| Admin analytics slow with 10,000+ orders | Low | Monitor | Pagination/filtering in v1.1 |
| Image upload validation edge case | Low | Closed | Proper validation added |

Limitations (MVP Scope)

* No WebSocket Implementation: Real-time updates use polling
* Single Currency: Only INR supported (Razorpay default)
* No Multi-Seller Cart: Can only order from single seller per transaction
* Limited Payment Methods: Only Razorpay and COD
* No Delivery Partner Integration: Manual delivery assignment by seller staff
* No Refund Management: UI ready, refund logic to be implemented in v1.1
* Basic Analytics: Limited to revenue, orders, subscriptions

13. Future Enhancements

Version 1.1 (Q2 2026)  
Priority: High

* WebSocket implementation for real-time order status sync
* Refund management system with payment reversal
* Email notification templates customization
* Seller-specific delivery zones and delivery fees
* Advance order scheduling
* Coupon and discount code system
* Advanced admin analytics

Priority: Medium

* Multi-language support
* Dark mode for UI
* Delivery partner app
* Order rating/feedback system
* Seller search by location/category
* Wishlist/favorites for users
* Bulk SMS campaign for sellers

Version 2.0 (Q4 2026)

* Mobile app (Angular Native or Flutter)
* Push notifications (Firebase Cloud Messaging)
* Advanced ML-based recommendations
* Social features (referral program, friend ordering)
* Aggregated seller data (OpenStreetMap integration)
* Multi-city expansion with regional language support
* In-app customer support chat
* Subscription auto-renewal management for sellers
* Automated invoice generation (GST compliance)
* Payment webhook for reconciliation

Version 3.0 (2027+)

* Expand to international markets
* Cloud warehouse integration
* AI-driven demand forecasting
* Dynamic pricing based on demand
* Supplier management system
* Product waste analytics
* Carbon footprint tracking
* Blockchain-based order verification (optional)

Additional Resources

Documentation Files

* [README.md](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html) – Quick start guide
* SETUP\_INSTALLATION\_GUIDE.md – Detailed installation steps
* API\_DOCUMENTATION.md – Complete API reference
* DATABASE\_SCHEMA.md – Database design details
* FEATURE\_DOCUMENTATION.md – Feature descriptions
* TESTING\_PLAN\_REPORT.md – Test cases and UAT results
* ADMIN\_SETUP\_GUIDE.md – Admin initialization
* SUBSCRIPTION\_MODEL\_DOCUMENTATION.md – Subscription details

External Links

* MongoDB Documentation: [https://docs.mongodb.com](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Express.js Guide: [https://expressjs.com](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Angular Documentation: [https://angular.dev](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Vite Documentation: [https://vitejs.dev](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Tailwind CSS: [https://tailwindcss.com](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Razorpay Integration: [https://razorpay.com/docs](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* SendGrid API: [https://docs.sendgrid.com](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)
* Twilio API: [https://www.twilio.com/docs](vscode-file://vscode-app/c:/Users/thath/AppData/Local/Programs/Microsoft%20VS%20Code/072586267e/resources/app/out/vs/code/electron-browser/workbench/workbench.html)