Assignment: Multi-Tenant Inventory Management System

Build a SaaS platform where multiple businesses (tenants) manage inventory, suppliers, and orders independently with complete data isolation. Must handle real-world complexities: product variants, concurrent operations, purchase orders, and smart stock alerts.

Core Requirements

1. Multi-Tenant Architecture - Each tenant has isolated data, multiple users with roles (Owner/Manager/Staff), and their own inventory/suppliers/orders.

YOUR DECISION: Document your tenant isolation approach (separate DBs, schema-based, or row-level) with pros/cons in ARCHITECTURE.md.

2. Complex Inventory -

- Products with variants (e.g., T-shirt: 3 sizes × 3 colors = 9 SKUs), each tracking its own stock
- Track all stock movements (purchase/sale/return/adjustment) with timestamps
- Smart low-stock alerts: Don't alert if the pending Purchase Order will replenish stock
- Order processing: Handle concurrent orders, insufficient stock, cancellations, and partial fulfillment

YOUR DECISIONS: (1) How to model variants in MongoDB? (2) How to prevent race conditions when two users order the last item?

3. Suppliers & Purchase Orders - Manage suppliers with pricing, create Purchase Orders with multiple items, track status (Draft→Sent→Confirmed→Received), handle partial deliveries and price variances, and auto-update stock on receipt.

4. Dashboard & Analytics - Show inventory value, low-stock items (considering pending Purchase Orders), top 5 sellers (30 days), and stock movement graph (7 days). **Must load <2 seconds with 10,000+ products.**

Technical Stack & Requirements

- Frontend: React (hooks), Context API/Redux, React Router, responsive UI, real-time updates (Socket.io)
- Backend: Node.js + Express, MongoDB + Mongoose, JWT auth, role-based access, proper error handling
- Must Handle: Concurrent operations, data integrity (stock never negative), MongoDB transactions, proper indexing
- Bonus: TypeScript, deployed demo (Vercel, Render, etc.), API documentation

Deliverables

1. GitHub Repository: Clean commits, .env.example, seed script with 2+ tenants

2. ARCHITECTURE.md (MANDATORY): Document your multi-tenancy approach, data modeling decisions, concurrency handling, performance optimization strategy, scalability considerations, and trade-offs made. Explain WHY you chose each approach.

3. README.md: Setup instructions, test credentials (2 tenants, different roles), features implemented, assumptions, known limitations, time breakdown

Submission

GitHub link (public or shared) | Live demo (optional) | Setup instructions

Important: We value clear thinking and decisions over perfect code. Document your reasoning. Handle edge cases. If you can't finish everything in 16-20 hours, prioritize core features and document what you'd improve.

Good luck! We look forward to reviewing your work.