Military Asset Management System – Project Overview

1. Project Overview

The Military Asset Management System (MAMS) enables commanders and logistics personnel to manage movement, assignment, and expenditure of mission-critical assets such as vehicles, weapons, and ammunition across multiple bases. The system ensures transparency, accountability, and operational efficiency.

Assumptions: Each base maintains independent inventory; authorized personnel operate the system; assets have unique identifiers. Limitations: Requires stable network for real-time updates; depends on correct user input for accuracy.

2. Tech Stack & Architecture

Backend

Node.js + Express chosen for scalability, high performance, and easy integration with RBAC middleware. RESTful APIs ensure interoperability.

Frontend

React selected for responsiveness, modular UI components, mobile-friendly design, and fast client-side rendering.

Database

PostgreSQL chosen for relational consistency, strong ACID compliance, and suitability for asset tracking, transfers, and auditing logs.

Architecture

Architecture includes: React frontend → Secure REST API Layer → RBAC Middleware → PostgreSQL DB. All transaction logs stored in a dedicated audit table.

3. Data Models / Schema

Core Entities: - Bases: BaseID, Name, Location - Assets: AssetID, EquipmentType, Description - Inventory: Tracks Opening Balance, Purchases, Transfers In/Out, Closing Balance - Transfers: Records movement between bases - Assignments: Tracks allocation to personnel - Expenditures: Records asset usage/expenditure - Users: Login credentials, role, assigned base - Audit Logs: All transactions with timestamps Relationships: One-to-many between bases → inventory, transfers, users.

4. RBAC Explanation

Roles: - Admin: Full system access, all bases. - Base Commander: Access restricted to specific base inventory, assignments, transfers. - Logistics Officer: Can record purchases, manage transfers, limited view access. Enforcement: Middleware validates JWT token → checks role → checks base association → grants/denies access.

5. API Logging

All sensitive actions (purchases, transfers, assignments, expenditures) are logged. Each log entry stores: UserID, ActionType, PayloadSnapshot, Timestamp, BaseID. Logs support audits and compliance requirements.

6. Setup Instructions

Backend: 1. Install Node.js 2. Run: npm install 3. Configure .env (DB URI, JWT Secret) 4. Run: npm start Database: 1. Install PostgreSQL 2. Import schema.sql 3. Configure user permissions Frontend: 1. Install Node 18+ 2. Run: npm install 3. Configure API URL 4. Run: npm run dev

7. API Endpoints (Key Examples)

POST /auth/login – User login GET /dashboard – View KPIs POST /purchases – Record a purchase POST /transfers – Transfer assets between bases GET /transfers/history – Complete movement logs POST /assignments – Assign assets to personnel POST /expenditures – Record expended assets