Military Asset Management System - Project Report

1. Project Overview

Description:

The Military Asset Management System is a full-stack web application for tracking, managing, and auditing military assets (vehicles, weapons, equipment, etc.) across multiple bases. It supports asset purchases, transfers, assignments, role-based access control (RBAC), and comprehensive audit logging.

Assumptions:

- Users are authenticated and assigned roles (admin, commander, officer).
- Each asset is uniquely identified and associated with a base.
- All critical actions are logged for audit and compliance.

Limitations:

- No real-time notifications.
- No mobile app (web only).
- Assumes a single organization (no multi-tenancy).

2. Tech Stack & Architecture

Frontend: React + Tailwind CSS

Backend: Node.js + Express

Database: MongoDB (Mongoose ODM)

Authentication: JWT (JSON Web Tokens)

RBAC: Custom middleware

Audit Logging: MongoDB-based, via middleware

Why chosen:

- React/Tailwind: Rapid UI, modern UX
- Node/Express: Scalable, async I/O
- MongoDB: Flexible schema, easy scaling
- JWT: Stateless, secure

- Middleware: Centralized enforcement

3. Data Models / Schema

User: name (unique), email (unique), password (hashed), role, rank, base (ref), department, contact, permissions, status
Base: name (unique), location, commander (ref), status, capacity
Asset: assetId (unique), name, type, category, specifications, currentBase (ref), status, assignedTo (ref), purchaseInfo, maintenanceHistory, location
Purchase: purchaseId (unique), base (ref), items, supplier, purchaseOrder, totalAmount, status, deliveryDate, notes
Transfer: transferId (unique), fromBase (ref), toBase (ref), assets, requestedBy (ref), approvedBy (ref), status, priority, transportDetails, securityClearance, reason, notes, timeline
Assignment: assignmentId (unique), asset (ref), assignedTo (ref), assignedBy (ref), base (ref), assignmentDate, expectedReturnDate, status, purpose, mission, condition, notes, expenditure, maintenance
AuditLog: timestamp, user (ref), action, resource, resourceId, details, ipAddress, userAgent, sessionId, base (ref), severity, notes

Relationships:

- Users are assigned to Bases.
- Assets belong to Bases and can be assigned to Users.
- Purchases, Transfers, and Assignments reference Assets, Users, and Bases.
- AuditLogs reference Users, Bases, and resources.

4. RBAC Explanation

Roles:

- Admin: Full access
- Commander: Manage assets, purchases, transfers for their base
- Officer: View/create assignments, view assets, submit requests

Access Levels:

- Enforced via middleware (hasPermission, requireRole)
- Permissions set per role, stored in User model

- Example: Only users with create_purchases can create a purchase

Enforcement:

- Middleware checks JWT, extracts user, verifies permissions before controller logic runs

5. API Logging

How:

- All sensitive/critical API actions are logged via middleware (auditMiddleware.js)
- Logs stored in AuditLog collection in MongoDB

What is logged:

- User, action, resource, resourceId, details, IP, user agent, session, base, severity, notes

Why:

- Ensures traceability, accountability, and compliance

6. Setup Instructions

Backend:

1. cd backend
2. npm install
3. Set up .env with MongoDB URI and JWT secret
4. node seed.js (optional: seed demo data)
5. npm start

Frontend:

1. cd frontend
2. npm install
3. npm start (dev) or npm run build (prod)

Database:

- Use MongoDB Atlas or local MongoDB
- Update connection string in backend .env

7. API Endpoints (Sample)

Auth:

POST /api/auth/register - Register new user

POST /api/auth/login - Login

Users:

GET /api/assignments/users/available - List users for assignment (RBAC protected)

Assets:

GET /api/assignments/assets/available - List available assets for assignment

Assignments:

POST /api/assignments - Create assignment

GET /api/assignments - List assignments

PATCH /api/assignments/:id/return - Return asset

PATCH /api/assignments/:id/expend - Mark asset as expended

Purchases:

POST /api/purchases - Create purchase

GET /api/purchases - List purchases

PATCH /api/purchases/:id/approve - Approve purchase

Transfers:

POST /api/transfers - Create transfer

GET /api/transfers - List transfers

PATCH /api/transfers/:id/approve - Approve transfer

Dashboard:

GET /api/dashboard/metrics - Dashboard metrics

GET /api/dashboard/movement-breakdown - Net movement breakdown