Military Asset Management System Documentation

1. Project Overview

Description:
The Military Asset Management System is a web-based application designed to manage military assets, assignments, purchases, and transfers across different bases. It provides secure authentication, role-based access control (RBAC), and a dashboard for tracking assets and user activities.

Assumptions:

- Users are military personnel with assigned roles (admin, base commander, logistics officer).
- Each user is assigned to a base (except admins).
- All asset transactions are logged for audit purposes.

Limitations:

- The system is designed for demonstration and may require enhancements for production use (e.g., advanced security, scalability).
- Only basic error handling and validation are implemented.

---

2. Tech Stack & Architecture

- **Backend:** Node.js, Express.js
    - Chosen for its scalability, asynchronous I/O, and large ecosystem.
- **Frontend:** React.js
    - Chosen for its component-based architecture and responsive UI capabilities.
- **Database:** MongoDB (with Mongoose ODM)
    - Chosen for its flexibility in handling complex, evolving data models.

Architecture:

- RESTful API backend
- React SPA frontend
- MongoDB for persistent storage

---

3. Data Models / Schema

- **User**: username, email, password, role, assignedBase, firstName, lastName, rank, isActive
- **Asset**: name, type, serialNumber, status, base, purchaseDate
- **Assignment**: assetId, userId, assignedDate, returnDate
- **Purchase**: assetId, purchaseDate, vendor, cost
- **Transfer**: assetId, fromBase, toBase, transferDate

Relationships:

- Users can be assigned to assets (Assignment)
- Assets can be transferred between bases (Transfer)
- Purchases are linked to assets

---

4. RBAC Explanation

- **Roles:**
    - Admin: Full access to all features and data
    - Base Commander: Manage assets and users at their base
    - Logistics Officer: Handle asset assignments, purchases, and transfers at their base
- **Enforcement:**
    - Middleware checks JWT and user role before allowing access to protected routes

---

5. API Logging

- All critical transactions (login, asset assignment, purchase, transfer) are logged to the database or server logs for audit and traceability.
- Errors and failed login attempts are also logged.

---

6. Setup Instructions

1. **Backend:**
    - Navigate to `server/`
    - Run `npm install`
    - Set up `.env` with MongoDB URI and JWT secret
    - Run `node server.js`
2. **Database:**
    - Ensure MongoDB is running (local or cloud)

    - Use provided seed scripts if needed

3. **Frontend:**
    - Navigate to `client/`
    - Run `npm install`
    - Run `npm start` to launch the React app

---

7. API Endpoints (Sample)

- `POST /api/auth/register` - Register a new user
- `POST /api/auth/login` - User login
- `GET /api/assets` - List all assets
- `POST /api/assignments` - Assign asset to user
- `POST /api/purchases` - Record asset purchase
- `POST /api/transfers` - Transfer asset between bases

For more details, refer to the codebase and README files.