Military Asset Management System — Technical Brief

Deployment Links:

- **Frontend**: Vercel
- **Backend**: Render

1. Project Overview

- Description: Manage asset lifecycle (purchase, transfer, assignment, expenditure) across bases with a dashboard and role-based access.
- Assumptions: Node 20, MongoDB available, consistent role enums (`Admin`, `BaseCommander`, `LogisticsOfficer`, `Viewer`), JWT auth, time in ISO/UTC.
- Limitations: No real-time updates or audit trail, minimal server logging, no offline mode, single-tenant, no email notifications.

2. Tech Stack & Architecture

- Frontend: React + Vite + Tailwind for fast DX, routing, and utility-first styling.
- Backend: Express + Mongoose for simple REST APIs and schema modeling.
- Database: MongoDB for flexible event-style asset transactions.
- Architecture: Client calls REST; JWT stored in `localStorage`; Axios attaches token; server enforces auth + role guards; dashboard aggregates data server-side.

3. Data Models / Schema

- User: name, email (unique), password (hashed), role (enum), base (optional).
- Base: name (unique), location.
- Asset: name, type (enum), base, quantity.
- Purchase: asset, base, quantity, purchaseDate, addedBy.
- Transfer: asset, fromBase, toBase, quantity, transferDate, initiatedBy.
- Assignment: asset, base, quantity, assignedTo, assignmentDate, createdBy.
- Expenditure: asset, base, quantity, reason, dateExpended, createdBy.
- Relationships: Transactions reference `Asset`, `Base`, and `User`; dashboard computes balances via aggregate pipelines.

4. RBAC Explanation

- Roles: `Admin` (full), `BaseCommander` (base-wide manage), `LogisticsOfficer` (operational manage), `Viewer` (read-only).
- Enforcement (server): JWT verified in auth middleware; role middleware restricts routes by role.
- Enforcement (client): ProtectedRoute blocks unauthenticated; RoleGuard hides unauthorized UI/actions.

5. API Logging

- Current: Minimal console logs (server start, DB connect). No structured HTTP access logging.
- Morgan: Future planning to implement Morgan for HTTP access logging. (unable to do so due to time constraints)

6. Setup Instructions

- Backend:
  - Set `server/.env`: `PORT`, `MONGO_URI`, `JWT_SECRET`, `CORS_ORIGIN`.
  - `cd server && npm install && npm start`.
- Frontend:
  - Set `client/.env`: `VITE_API_BASE_URL` (e.g., `http://localhost:5000/api`).
  - `cd client && npm install && npm run dev`.
- Database:
  - Use a reachable MongoDB URI; models create indexes on first run.

7. API Endpoints (Key)

- Base URL: `http://localhost:5000/api`.
- Auth:
  - `POST /auth/register` → `{ name, email, password, role }`.
  - `POST /auth/login` → returns `{ token, user }`.
- Assets & Bases:
  - `GET /bases`, `POST /bases`, `GET /bases/:id`, `PUT /bases/:id`, `DELETE /bases/:id`.
  - `GET /assets`, `POST /assets`, `GET /assets/:id`, `PUT /assets/:id`, `DELETE /assets/:id`.
- Transactions:
  - `GET|POST /purchases`, `GET|PUT|DELETE /purchases/:id`.
  - `GET|POST /transfers`, `GET|PUT|DELETE /transfers/:id`.
  - `GET|POST /assignments`, `GET|PUT|DELETE /assignments/:id`.
  - `GET|POST /expenditures`, `GET|PUT|DELETE /expenditures/:id`.
- Dashboard:
  - `GET /dashboard` → filters: `base`, `type`, `startDate`, `endDate`; returns opening/closing balances and net movements.

Notes: Protected routes require `Authorization: Bearer <JWT>`. The client attaches it automatically.