

Decision Log - Drone Operations AI Agent

Skylark Drones Assignment

10th February 2026

Executive Summary

Built an AI-powered conversational assistant for drone operations coordination using Python, Streamlit, LangChain, and Groq LLM. The agent manages pilot rosters, drone inventory, mission assignments, and automated conflict detection with real-time Google Sheets synchronization.

Key Technology Decisions

1. Python + Streamlit (Not React/Next.js)

Why: Rapid prototyping in single language, built-in chat UI, free deployment, 6-hour timeline constraint.

Trade-off: Less UI customization

2. Groq LLM (Initially OpenAI, Switched for Cost)

Initial Choice: OpenAI GPT-4o for best function calling reliability.

Final Choice: Groq `llama-3.3-70b-versatile` (100% FREE, no credit card).

Why Switch: Assignment demo doesn't justify API costs; Groq provides unlimited free tier.

Implementation: Simplified agent with keyword-based tool routing instead of complex LangChain agents due to API compatibility.

Trade-off: Less sophisticated reasoning, but ****\$0 cost**** for unlimited usage.

3. Google Sheets via gspread (Not Database)

Why: Required by assignment, Python library simpler than REST API.

Trade-off: Slower than database, but **saved 2 hours** on infrastructure setup.

4. Real-time Sync (Not Batched)

Why: Better UX for prototype demonstration, immediate feedback.

Trade-off: More API calls, but **better for evaluation**.

Critical Assumptions

1. Google Sheets Structure

Assumption: One spreadsheet with 3 worksheets: Pilots, Drones, Missions.

Rationale: Assignment specifies Google Sheets; single source of truth avoids database complexity.

2. Service Account Authentication

Assumption: Users can create Google Cloud Project and configure service account.

Rationale: Enables headless deployment to Streamlit Cloud without OAuth flows.

3. Comma-Separated Multi-Value Fields

Assumption: Skills/certifications stored as "Mapping, Survey, Inspection".

Impact: Parsing logic converts to Python lists for matching.

Urgent Reassignment Interpretation

Definition: When high-priority mission needs immediate resources, propose reallocating from lower-priority missions.

Implementation: `urgent_reassign` tool that:

1. Identifies resources assigned to source project
2. Analyzes target mission requirements (skills, location)
3. Generates impact report (delays, affected projects)
4. **Requires user confirmation** before execution

Why This Approach:

Safety: No automatic swapping without human approval

Transparency: Shows full impact before changes

Flexibility: Agent suggests, human decides

Alternative Rejected: Fully automated swapping → Too risky without human oversight.

Future Improvements (Given More Time)

Hungarian algorithm for optimal assignment - Automated optimization vs manual matching

WebSocket real-time updates - Multi-user collaboration without polling

pytest unit test suite - Regression prevention for conflict logic

Redis caching layer - 5x faster queries, reduced API quota

Audit log (separate Sheet) - Track changes, compliance, rollback

Better NLU with few-shot prompts - More reliable query parsing