[Arena]

Take-Home Test: Extend UAV Logger with an Agentic Chatbot Backend

Objective: Fork the UAV Logger repository and extend its functionality by integrating a **chatbot backend**. This task tests your ability to work with real-world UAV telemetry data, backend development, and LLM integration.

There's no time limit or limitation on the tools you should be using; you are encouraged to use all the tools, documentation and AI tools available. We believe that spending 8 hours should give us enough signal, however you can choose to spend as much time as possible.

We want to see **creativity** in features/scope, ability to build fast by using **development tools**, interest in **understanding a new domain** such as flight.

Requirements:

1. **Fork and Setup:**
    - Fork the UAV Logger repository to your own GitHub account.
        i. Website: https://plot.ardupilot.org/#/
        ii. Code: https://github.com/ArduPilot/UAVLogViewer
    - Set up the project locally and verify that you can run the existing code.
2. **Backend Extension:**
    - Add a new backend module **in Python** *(or optionally extend the existing Node.js backend)*.
    - Integrate a **chatbot feature** that interacts with users through the backend.
    - You can use **any LLM API** (OpenAI, Anthropic, Mistral, Groq, etc.).
    - Required: Stick to Full Stack, make sure that the BE apis are functioning.
3. **Tooling:**
    - Required: use an AI programming tool like **Windsurf**, **Cursor.dev**, **Continue**, or a similar AI assistant to support your development process.

New Feature 1: Chatbot

The chatbot will allow users to ask **questions about MAVLink protocol data** parsed from .bin flight logs.

- The chat screen should be incorporated into the existing app. When a user uploads the bin file, the existing app should work as expected.
- The user should be able to upload any file and use the chatbot. Please do not store the data in github.
- The chatbot should behave **agentically**, meaning it maintains conversation state and can proactively ask for clarifications when needed.
- It should retrieve parsed telemetry information dynamically.
- LLM can be prompted to use the documentation here: https://ardupilot.org/plane/docs/logmessages.html
- Upload a bin file (e.g.this bin file)
- Attention to UI/UX is appreciated.
- Example questions users should be able to ask - **these are just example questions, we don't expect you to write specific code blocks to answer each question**.
    - "What was the highest altitude reached during the flight?"
    - "When did the GPS signal first get lost?"
    - "What was the maximum battery temperature?"
    - "How long was the total flight time?"
    - "List all critical errors that happened mid-flight."
    - "When was the first instance of RC signal loss?"

New Feature 2: Make the Chatbot Flight aware

Extend the chatbot to also **detect and reason about flight anomalies** based on MAVLink data.

- Users should be able to ask **high-level, investigative questions**, such as (again these are just example questions, we don't expect you to write specific code blocks to answer each question**)**
    - "Are there any anomalies in this flight?"
    - "Can you spot any issues in the GPS data?"

- Your system should design a mechanism for handling these broader queries:

    - Either **send structured telemetry data to the LLM** and let the LLM infer and detect anomalies.
    - Or **detect and reason about anomalies server-side** before presenting results.

- **Agentic vs Hardcoded Behavior:**
    - Instead of rigid, rule-based checks (e.g., "altitude must not drop more than 10 meters in 1 second"), aim to **let the agent reason** about patterns, thresholds, and inconsistencies dynamically.
    - You may **hint or suggest strategies** to the LLM (e.g., "look for sudden changes in altitude, battery voltage, or inconsistent GPS lock") instead of explicitly coding every rule.
    - Design your prompts, context injection, or intermediate analysis to encourage **flexible reasoning** rather than binary rule evaluation.

What are we looking for?

As part of our evaluation process, we'll be reviewing your submission with a focus on identifying areas where your work stands out. Specifically, we're looking for spikes or differentiated output in any of the following areas:

- Full Stack development: full integration to the existing code base, and thoughtful APIs
- UX and design thinking: ease of use, additional visualizations besides text.
- Agent building: following agentic standards, use of state of the art tools and packages.
- End-to-End (ETE) intelligence: Going beyond rule based, hard coded prompts.

We do not expect that candidates will have exceptional output in all of the above areas, so please do not hesitate to submit your effort even if you feel it is "lopsided" (e.g. stronger in fullstack and UX design, weaker use of agent building). Show us where you shine! Lopsided submissions are not only acceptable, they are encouraged.

What to expect next

- When you are done, please do the following;

    - Share your fork in github with us, with instructions of how to run it.
    - Take a video of the new functionality and share with us,
        - You can use any file sharing service such as dropbox / google drive etc. Send us the link. We will get back to you as fast as possible, and arrange the next steps.
        - The video should only include the app, please avoid sending us your dev setup, code, backend APIs in the video, or additional slides/presentation.. **Anything that's not the demo, we will not review.**
- We want to hear how you dealt with the project, in a 30 minute session.