Assignment Document

This is a confidential document. This is not to be shared with anyone other than the candidate who is being evaluated for a role. The candidate must destroy their copy of this document after the intended purpose has been served.

Assignment - AI Engineer

We are seeking innovative AI engineers who can solve complex problems across diverse domains with a commitment to excellence, efficient execution, and ownership. This role emphasizes your ability to build cutting-edge AI solutions using the latest tools and frameworks, innovate with modern AI methodologies, and collaborate effectively.

As part of the AI team, we expect the same level of passion, commitment, and pursuit of excellence from you towards your role. This assignment focuses on your AI engineering skills and problem-solving abilities, applicable to any context.

We expect you to use AI-assisted tools **(Claude Code, Cursor AI, Windsurf, Lovable, Replit etc)** to aid your development process and to document how these tools helped expedite your work. The assignment is designed to be challenging yet achievable within the time limit when leveraging the latest AI tools and practices.

Background

You are tasked with designing and prototyping a Drone Security Analyst Agent for a docked drone that monitors a fixed property daily. The agent processes live telemetry data (e.g., drone position, altitude) and video feed in real-time to detect and analyze security events. It must identify objects or activities (e.g., “a blue Ford F150 entered twice today”) for future reference and generate immediate alerts (e.g., “person loitering at midnight near main gate”). Your goal is to build a functional prototype that showcases how this could work, using AI to assist you at every stage.

Problem Statement

Create a prototype for the Drone Security Analyst Agent. The system should:

1. Process simulated drone telemetry data and video frames.
2. Analyze video content to identify objects or events (e.g., vehicles, people) and log them with context.
3. Generate real-time security/safety alerts based on predefined rules.
4. Include a cross-domain element: explore indexing video frame-by-frame (e.g., using a database or search tool), even if this is new to you, and use AI to learn and implement it.

Requirements

1. Write a short feature spec defining the agent's value to property owners (e.g., "enhances security with automated monitoring") and 2–3 key requirements.
2. **Design/Architecture**: Propose a simple architecture for processing telemetry and video (e.g., data pipeline, storage, alert system). Use AI to suggest or validate it.
3. **Development**:
    - Implement a prototype in a language of your choice (e.g., Python).
    - Simulate video frames with text descriptions (e.g., "Frame 1: Blue truck at gate") and telemetry (e.g., "Time: 00:01, Location: Gate").
    - Use AI to generate at least one component (e.g., object detection logic, alert rules).
4. **Cross-Domain (Indexing)**: Build a basic frame-by-frame indexing system (e.g., store frames in a database with timestamps). Use AI to learn this if unfamiliar.
5. **QA**: Create test cases to verify functionality (e.g., "truck logged correctly," "alert triggered at midnight"). Use AI to assist.

Expected Sample Output

- **Logs**: "Blue Ford F150 spotted at garage, 12:00."
- **Alert**: "Person loitering at main gate, 00:01."
- **Indexed Frames**: Queryable by time or object (e.g., "show all truck events").

Submission Details:

- **Code**:
    - Submit your code through a private GitHub repository (GitHub provides them free of cost).
    - Create a repo on GitHub, upload your VLM scripts, LangChain agent code, context management implementation, and any test scripts—keep the repo private.
    - Add **assignments@flytbase.com** as a contributor to the repo.
- **Comprehensive Documentation:**
    - **README File:**
      Include detailed setup and running instructions.
        - Explain your design decisions and architectural choices.
        - Detail the AI tools you integrated, including their impact on your workflow.
    - **Design Artifacts:**
      Submit flowcharts, system architecture diagrams, or any other design artifacts that illustrate your problem-solving process and overall system design.
    - **Testing Documentation:**
      Explain how you validated the system's functionality, including test cases or scenarios for dynamic inputs and emergency responses.
- **Videos**:

    - Submit one or more videos (screen recordings) for each goal, showing video processing, context summaries, agent recommendations, scalability test, and innovative features (e.g., show frame descriptions, agent output, generated captions).
    - Ensure that your demo video has your **voiceover** explaining your solution. Videos without a voiceover will lead to incorrect analysis of the assignment submission.
- **Report**:
    - Submit a PDF report summarizing your approach to the problem.
    - Explain any assumptions you made (e.g., dataset choice, VLM selection).
    - Justify your choice of tools and configurations (e.g., Why CLIP vs. BLIP? Why this agent design?).
    - Show your results through examples (e.g., frame descriptions, agent recommendations) and reference your videos.
    - Discuss what could have been done better if submission time was not constrained (e.g., Could you add video summarization? Use a more advanced VLM?).
    - Detail how AI tools (e.g., Claude Code, Windsurf) assisted your work—e.g., "Claude Code generated LangChain agent code, but I customized the prompt."
    - Attach videos to the email or provide links to view them (e.g., Google Drive, ensure view access).

Additional Notes:

- **Environment Setup:** You are responsible for setting up your environment (e.g., Python, Hugging Face Transformers, LangChain, OpenCV) and sourcing the dataset and tools yourself. Include setup instructions in your GitHub README.
- **For Novices:** If you lack experience with tools (e.g., LangChain, VLMs), mention this in your report. The evaluation will consider your effort and learning approach.
- **Report Weightage:** The report carries equal weight in your assessment—spend sufficient time on it to explain your reasoning and analysis.

Please note: Any candidate found guilty of plagiarism will be disqualified from the assignment and will not be considered for further rounds of evaluation.

- Ensure appropriate viewing rights when sharing access to the repository and videos.
- Provide clear instructions in your README for environment setup and execution of the project.

Assessment Metrics:

- **Correctness & Model Performance (25%):** VLM generates meaningful frame descriptions, agent provides relevant recommendations.
- **Reasoning & Scalability (25%):** Agent uses context throughout the video

- **Experimentation & Innovation (30%):** Novel approach to problem solving, good documentation/report will help us evaluate this.
- **Documentation & Code Quality (20%):** Clear README, well-structured code, detailed report.

Bonus Enhancements (Bonus Points):

- Implement a video summarization feature (e.g., generate a 1-sentence summary of the video).
- Add an agent capability to answer follow-up questions (e.g., “What objects were in the video?”).
