DroneShield AI Chatbot Project

Objective

The goal of this project was to create a functional AI chatbot tailored to DroneShield's public-facing content, enabling efficient access to verified product information. It needed to be of production level quality that a company would consider embedding and deploying on their website. The chatbot serves as a proof of concept for integrating simple AI models into defence-focused customer and operations workflows.

Methodology

The chatbot was developed using a DroneShield flavoured chroma through the Streamlit framework. The process began by collecting public data from DroneShield's website and structuring it into a context retriever system. A lightweight Retrieval-Augmented Generation (RAG) method was used to ensure accurate, document-based responses. GPT-4 is used as a fallback as needed. Development occurred iteratively in Google Colab and Streamlit Cloud with assistance from Gemini, ChatGPT, and Claude for code debugging and refinement.

Trials, Challenges, and Key Learnings

One of the major challenges was adapting complex chatbot logic without formal coding experience or education. Initially, deployment errors in Streamlit and conflicts between API calls and context handling caused the app to fail repeatedly. Through collaboration with AI assistants and experimentation, I learned how Python functions interact within Streamlit's reactive framework, how caching affects API efficiency, and how to handle missing data gracefully through fallbacks.

Another challenge was balancing accurate information retrieval with clear, concise responses. The bot originally gave unoptimized answers which defeated the purpose of the project.

Through multiple testing phases, prompt engineering was refined to guide GPT-4's tone, scope, and accuracy. This process strengthened my understanding of prompt structures, dependencies, and iteration testing.

Results and Conclusion

The final chatbot successfully uses a custom retrieval component capable of referencing DroneShield's public content accurately. It features a branded interface, fallback responses for incomplete data, and strong contextual alignment. The project demonstrates how low-code AI solutions can be rapidly deployed with the aid of modern LLMs, even by non-programmers.

This experience provided a solid foundation in applying AI reasoning to real-world corporate use cases — particularly those relevant to defence and automation sectors.

Tech Stack

- OpenAI GPT-4 API
- Streamlit
- Python
- Google Colab
- RAG (Retrieval-Augmented Generation)
- HTML
- GitHub (Version Control)
- SiteGPT
- Claude