GenAI-Powered Data Pipeline Project Report

Date: March 22, 2025

Student: Rusheek Reddy (Assumed from screenshots)

Course: ALY-6050

# 1. Project Overview

The goal of this project was to build a cloud-based GenAI-powered data pipeline using Langflow and AstraDB.   
The system ingests an unstructured FAQ dataset, vectorizes it using OpenAI embeddings, and enables a Retrieval-Augmented Generation (RAG) workflow to answer user queries based on the ingested data.

# 2. Architecture & Components

The architecture consists of two main flows in Langflow:  
  
- Load Data Flow: Ingests CSV data, splits text into chunks, generates embeddings using OpenAI, and stores vectors in AstraDB.  
  
- Retriever Flow: Accepts user questions, embeds the query, retrieves relevant chunks from AstraDB, constructs a prompt, and uses GPT-3.5 to generate an answer.

# 3. Tools & Technologies

- Langflow (Cloud version)  
- AstraDB with vector search enabled  
- OpenAI Embeddings (text-embedding-3-small)  
- OpenAI GPT-3.5 Turbo LLM  
- Dataset: faq\_bulk\_150\_records.csv

# 4. Key Design Decisions

- Chose text chunking (100 size, 20 overlap) to optimize context retrieval.  
- Used `text-embedding-3-small` for faster embeddings.  
- Limited retrieval results to 5 chunks to reduce LLM context size and speed up response time.  
- Chose GPT-3.5 for balanced speed and quality.  
- Integrated prompt templating with variables for dynamic RAG prompting.

# 5. Challenges & Solutions

- Saving Flow Issue (504 Timeout): Solved by retrying and exporting JSON manually.  
- Slow response time: Optimized by reducing retrieved chunks and simplifying prompt.  
- Langflow component connection bugs: Solved via careful step-by-step testing and visual flow debugging.

# 6. Results & Screenshots

The system successfully answers user queries based on the ingested FAQ data. Latency was kept under 2–3 seconds per full query cycle.  
Screenshots are available in the GitHub repository under /assets/screenshots showing the Load and Retrieval Flows in action.

# 7. Future Improvements

- Add feedback loop to improve answers over time.  
- Deploy as a public chatbot using Streamlit or Gradio.  
- Integrate LangSmith for observability.  
- Use fine-tuned LLM for domain-specific support.  
- Explore Hugging Face open-source LLMs for cost reduction.