Applied AI Build Challenge – Project Report

Title: Personal Knowledge Assistant Using Retrieval-Augmented Generation (RAG) with Flask + ChromaDB + Perplexity API

1. Objective of the Project

The goal of this project is to build a functional, end-to-end AI-powered Retrieval-Augmented Generation (RAG) system that can answer questions using my own personal text data. The system uses a Flask web interface, a local vector database (ChromaDB) for storing embeddings, and the Perplexity API for generating context-aware responses.

2. Tools & Frameworks Used

- Flask 3.0

- ChromaDB 0.5

- SentenceTransformers (all-MiniLM-L6-v2)

- Perplexity API

- Python 3.10

- Torch, Transformers, dotenv, requests

3. Project Summary (Workflow)

Personal .txt files → Chunking → Embeddings → Vector Database (ChromaDB)

User Query → Query Embedding → Semantic Search → Retrieve Chunks

LLM (Perplexity API) → Final Answer

4. Key Implementation Steps

- Created Python 3.10 virtual environment

- Installed dependencies from requirements.txt

- Implemented rag.py with embedding, chunking, retrieval, and Perplexity integration

- Built Flask app with routes for homepage and question answering

- Loaded all personal data from /data on startup

5. Results & Observations

- Successfully loaded and embedded text data

- Retrieval accuracy high

- Perplexity produced contextual answers

- Flask app works end-to-end

6. Learnings

Learned how RAG systems, vector DBs, embeddings, and LLM APIs integrate into a real project. Understood version compatibility issues and AI workflow structuring.

7. Future Improvements

- Add PDF & OCR ingestion

- Add authentication

- Deployment on cloud

- Add chat history