AI Hallucination Detection System - Integration Guide

This document explains how to integrate the individual modules developed by each team member into a single working prototype of the AI Hallucination Detection System. The final system will use Gemini Pro as the base LLM, a retrieval pipeline (Wikipedia + ChromaDB), a detection and correction layer, and a frontend for interaction.

# Step 1: Setup Common Environment

- Ensure all team members use Python 3.9+.  
- Create a shared `requirements.txt` with all dependencies:  
 datasets, wikipedia, chromadb, sentence-transformers, google-generativeai,  
 langchain, flask, streamlit, scikit-learn, matplotlib, sqlite3.  
- One member creates a GitHub repo; others clone it and push their modules.

# Step 2: Connect Retrieval (Member 1) with Detection (Member 2)

- Member 1 provides retrieval function: `retrieve\_evidence(question)` that returns top documents.  
- Member 2 integrates it inside hallucination detection pipeline.  
- Workflow:  
 1. User question → Gemini Pro generates raw answer.  
 2. Retrieval function fetches supporting docs.  
 3. Detection compares raw answer with evidence embeddings.

# Step 3: Add Correction Module (Member 3)

- Member 3 provides `correct\_answer(raw\_answer, evidence)` function.  
- This uses Gemini Pro again with retrieved evidence to regenerate a fact-grounded answer.  
- The corrected answer is stored with citations + confidence in SQLite.  
- Now pipeline looks like:  
 Question → Raw Answer → Detection → If Hallucination → Correction → Final Answer.

# Step 4: Frontend Integration (Member 4)

- CLI connects directly to the pipeline.  
- Flask provides REST API endpoints:  
 `/ask` → accepts question, returns raw + corrected answer, citations, confidence.  
- Streamlit calls Flask API to display interactive Q&A interface with highlighted hallucinations.

# Step 5: Testing & Deployment (Member 5)

- Member 5 integrates testing script:  
 1. Run TruthfulQA dataset through both raw Gemini and corrected pipeline.  
 2. Save results in SQLite + generate graphs.  
- Deploy Flask backend (e.g., Render/Heroku).  
- Deploy Streamlit app (Streamlit Cloud).  
- Share public demo link.

# Final System Workflow

1. User asks question via CLI or Streamlit frontend.  
2. Backend (Flask + LangChain) sends query to Gemini Pro.  
3. Retrieval module fetches evidence from Wikipedia/ChromaDB.  
4. Detection module checks hallucination risk.  
5. If hallucination → correction module regenerates grounded answer.  
6. SQLite logs store raw + corrected outputs with citations.  
7. User sees both answers, hallucination flag, confidence score, and citations.