1. Technical Design Specification

The system, **NutriGraph**, is an Agentic Multimodal RAG application. It ingests visual data (menu/food images) and text, retrieves ground-truth nutrition data (USDA/OpenFoodFacts), and uses an agentic loop to refine accuracy through user interaction.

High-Level Architecture (GCP-Centric)

- **Frontend: Streamlit** (Python). Rapid prototyping for both "Diner" and "Restaurant" views.
- **Backend: FastAPI**. Serves the agent logic and retrieval endpoints.
- **LLM & Multimodal: Gemini 1.5 Flash** (via Vertex AI). Chosen for low latency, low cost, and native multimodal (image + text) capabilities.
- **Orchestration: LangGraph** (or LangChain). Critical for the "Agentic" loop (managing state between follow-up questions).
- **Data Layer:**
    - **Vector Store: ChromaDB** (local/containerized for speed) or **Vertex AI Vector Search**.
    - **Knowledge Base:** USDA FoodData Central & OpenFoodFacts (parquet/JSON dumps).

Module Breakdown

1. **Ingestion Agent (The "Eyes"):**
    - **Input:** Images of menus/dishes.
    - **Model:** Gemini 1.5 Flash with a prompt to extract structured ingredients and visual volume estimates.
    - **Output:** JSON list of potential ingredients.
2. **Retrieval Engine (The "Brain"):**
    - **RAG Pipeline:** Hybrid search (Keyword + Semantic).
    - **Source:** USDA Standard Reference (for raw ingredients) + Branded Foods (for specific products).
    - **Logic:** Matches extracted ingredients to database IDs.
3. **Clarification Agent (The "Interviewer"):**
    - **Trigger:** When confidence for an ingredient match or quantity is `< 90%`.
    - **Action:** Generates a specific follow-up question (e.g., "Is the chicken grilled or fried?", "What brand is the BBQ sauce?").
    - **State Management:** Uses LangGraph to maintain conversation history until convergence.
4. **Evaluation Engine:**
    - **Metric:** Mean Absolute Error (MAE) on Calorie/Protein/Carb/Fat counts.

    - **Ground Truth:** A "Golden Set" of 50 items with lab-verified data (as per proposal).

---

2. Weekly Project Plan (Team of 3)

Roles:

- **Student A (ML Architect):** Focus on Agentic Workflow (LangGraph), LLM Prompting, and Vision.
- **Student B (Data Engineer):** Focus on Data Pipelines (USDA/OpenFoodFacts), Vector DB, and Backend API.
- **Student C (Product Engineer):** Focus on Frontend (Streamlit), Evaluation Framework, and User Testing.

Week 1: Infrastructure, Data & "Hello World"

- **Goal:** A working pipeline where text input retrieves nutrition data.
- **Aryaman:** Create the "Gold Standard" CSV file with the first 10 ground-truth dishes. Build the basic **Gemini 1.5 Flash prompt** for identifying ingredients from food images.
- **Shivangi:** Download **USDA/OpenFoodFacts datasets**. Clean and chunk data. Set up **ChromaDB** and write a script to index the top 1000 most common ingredients.
- **Gautam:** ~~Initialize the **GitHub repo** and set up the **Streamlit** skeleton (tabs for Diner vs. Restaurant). Set up GCP Project (Vertex AI access).~~

Week 2: Core RAG & Multimodal Ingestion

- **Goal:** Upload an image, get a rough nutritional estimate (without follow-up questions).
- **Aryaman:** Implement the **Image-to-JSON** pipeline. Ensure the LLM correctly lists ingredients from a photo of a dish or menu.
- **Shivangi:** Build the **FastAPI retrieval endpoint**. It should take a list of text ingredients and return the closest matches from the Vector DB.
- **Gautam:** Connect Streamlit to the FastAPI backend. Build the **"Dish Detail View"** (displaying calories/macros). Implement a basic feedback form for users to flag wrong data.

Week 3: The "Agentic" Loop (The Complex Part)

- **Goal:** The system asks clarifying questions when uncertain.
- **Student A:** Implement **LangGraph state machine**. Define the logic: `if match_score < threshold -> generate_question -> wait_for_user_input`.

- **Student B:** Optimize retrieval. Implement **hybrid search** (e.g., if user types "Heinz Ketchup", match exact brand string before semantic vector search).
- **Student C:** Update UI to handle **chat-like interaction**. When the backend sends a "question", the UI must display it and let the user reply.

Week 4: Evaluation, Polish & Presentation

- **Goal:** Finalize metrics and slide deck.
- **Student A:** Run the **Agent Efficiency** metric (avg. # of questions to convergence). Fine-tune prompts to reduce "annoying" questions.
- **Student B:** Ensure the system scales to the full 50-item Gold Set. specific edge cases (e.g., "dressing on the side").
- **Student C:** Run the **MAE benchmarking script** against the Gold Standard. Create the final **slides/demo video**. Generate the "Restaurant Comparison" chart for the final report.

Summary of Deliverables by Member

Feature	Student A (ML/Agent)	Student B (Data/Backend)	Student C (UI/Eval)
Data	Prompt engineering for extraction	Ingestion scripts & Vector Indexing	"Gold Standard" dataset creation
Logic	LangGraph Agent & Vision Model	FastAPI Endpoints & Search Algo	Streamlit UI & State Management
Eval	Convergence Metrics (Agent speed)	Latency Optimization	Accuracy Metrics (MAE) & User Testing