Agentic RAG Project

Problem Statement

Develop an Agentic RAG (Retrieval-Augmented Generation) system designed for personalized nutrition and weight-loss diet planning. the system should generate tailored dietary recommendations based on patient-specific inputs, including name, age, gender, height, and weight. the solution should be implemented using Google ADK (Agent Development Kit) or a LangGraph-based architecture to enable structured orchestration, agentic workflows, and reliable retrieval-augmented reasoning.

Solution Overview

The solution is implemented as an Agentic RAG system using a LangGraph-based architecture, where the overall workflow is decomposed into specialized agents for intake processing, knowledge retrieval, diet planning, and safety validation.

A Retrieval-Augmented Generation (RAG) approach is used by storing domain-specific nutrition documents in ChromaDB, where text chunks are converted into vector embeddings using sentence-transformers (all-MiniLM-L6-v2). Relevant nutrition context is retrieved via similarity search and provided to the planning agent.

The diet plan is generated using a Groq-hosted LLaMA 3.1 large language model, while a dedicated safety agent enforces calorie limits, allergy constraints, and health risk checks. This design ensures the system produces personalized, grounded, and safe dietary recommendations.

Tools & Technologies Used

- **LangGraph** – Agent orchestration and workflow control
- **ChromaDB** – Vector database for nutrition knowledge
- **Sentence-Transformers (all-MiniLM-L6-v2)** – Text embeddings
- **Groq API (LLaMA 3.1)** – Low-latency LLM inference
- **Python** – Core implementation language

Agents

Agent - 1: **intake.py –** This agent Collects Data from the User and performs essential health calculations. (BMI, BMR, TDEE, Safe target calories, generates risk flags) produces a clean structured profile that becomes the input for all other agents.

Agent - 2: **ingest_kb.py -** Reads all text files from the KB. Breaks long documents into smaller chunks. Converts each chunk into embeddings. Stores all chunks + embeddings in a vector database (Chroma). the nutrition knowledge base so it can be searched efficiently. This agent runs only once or whenever KB is updated.

Agent – 3: **retriever.py -** This agent retrieves relevant nutrition information using a Retrieval-Augmented Generation (RAG) approach. It builds a semantic query from the structured intake data, embeds the query using the same embedding model (sentence-transformers/all-MiniLM-L6-v2) used during knowledge base ingestion, and performs similarity search against the Chroma vector database. The agent returns the most relevant nutrition context to support accurate and grounded diet planning.

Agent – 4: **planner.py** – This agent is responsible for generating the personalized diet plan. It combines structured user health data from Agent-1 with retrieved nutrition knowledge from Agent-3, and uses a Large Language Model (Groq-hosted LLaMA 3.1) to generate a realistic, safe, and personalized meal plan. The model is accessed via the Groq API, enabling fast, low-latency inference.

Agent – 5: **safety.py -** This agent validates the diet plan generated by the Planner Agent to ensure safety and compliance. It checks for minimum calorie thresholds, allergy violations, and risk-flag constraints (such as underweight or high-risk users). The agent adds safety warnings and a medical disclaimer when required, preventing unsafe or misleading dietary recommendations.

Orchestrator

Orchestrator.py: The orchestrator acts as the central controller of the Agentic RAG system. It manages the execution flow by invoking each agent in the correct order, passing structured outputs between agents, and handling user interaction. The orchestrator ensures seamless coordination between intake processing, knowledge retrieval, LLM-based planning, and safety validation, ultimately producing a reliable and end-to-end personalized nutrition plan.

Knowledge Base

guidelines.txt - Provides general weight-loss and nutrition guidelines that apply to everyone. Planner Agent uses this information to calculate safe calorie plans and justify decisions with citations.

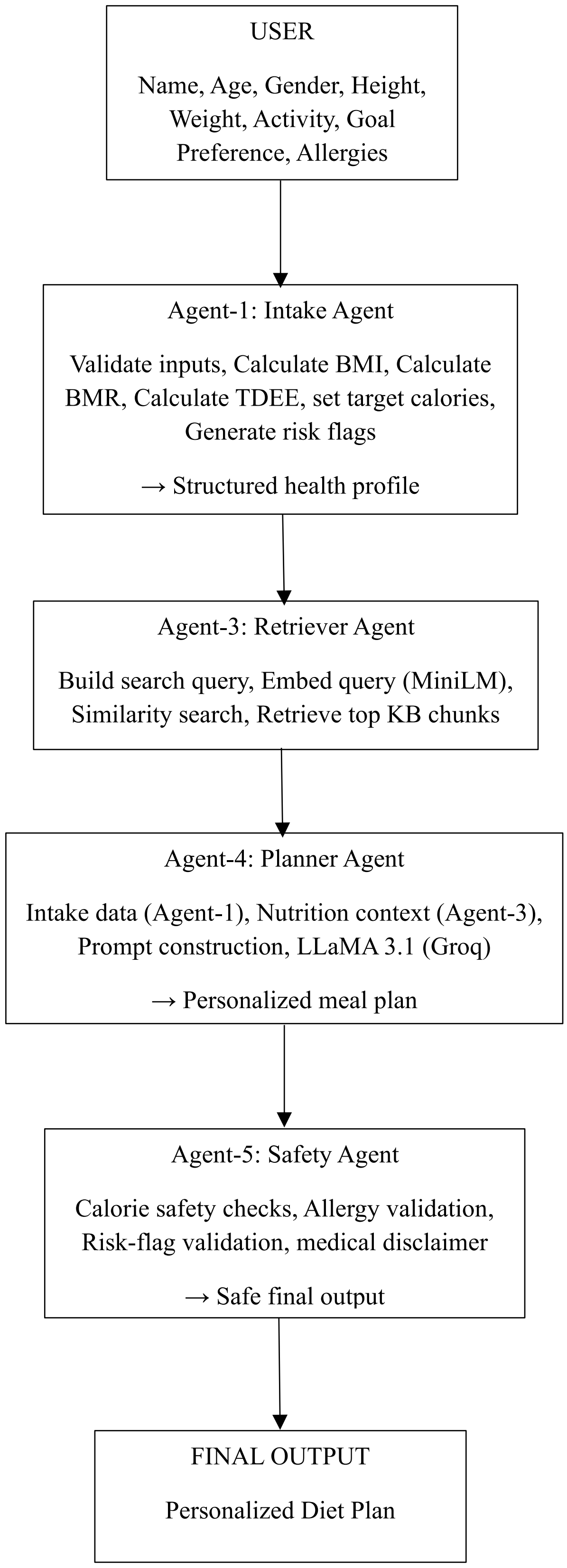
protein_info.txt - Provides detailed information about protein needs and high-protein food sources.

vegetarian_tips.txt - Provides nutrition advice specifically for vegetarian users.

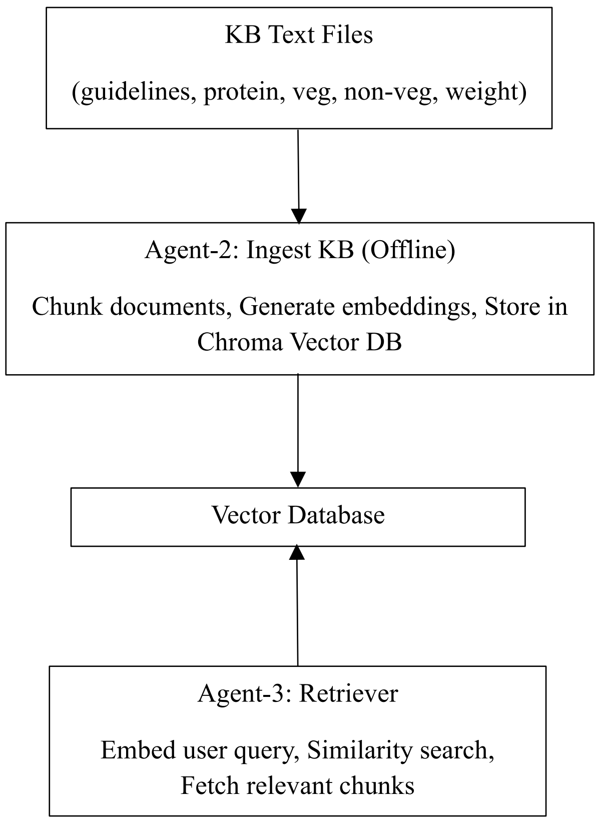
nonvegetarian_tips.txt - Provides guidance for non-vegetarian users.

weight_guidelines.txt – Provides for all the different weights.

Detailed Agent Flow


USER
Name, Age, Gender, Height, Weight, Activity, Goal Preference, Allergies
Agent-1: Intake Agent
Validate inputs, Calculate BMI, Calculate BMR, Calculate TDEE, set target calories, Generate risk flags
→ Structured health profile
Agent-3: Retriever Agent
Build search query, Embed query (MiniLM), Similarity search, Retrieve top KB chunks
Agent-4: Planner Agent
Intake data (Agent-1), Nutrition context (Agent-3), Prompt construction, LLaMA 3.1 (Groq)
→ Personalized meal plan
Agent-5: Safety Agent
Calorie safety checks, Allergy validation, Risk-flag validation, medical disclaimer
→ Safe final output
FINAL OUTPUT
Personalized Diet Plan

RAG Flow


KB Text Files
(guidelines, protein, veg, non-veg, weight)
Agent-2: Ingest KB (Offline)
Chunk documents, Generate embeddings, Store in Chroma Vector DB
Vector Database
Agent-3: Retriever
Embed user query, Similarity search, Fetch relevant chunks