# **Recipe Generator - Architecture Document**

## **1. Overview**

### ****Project Name:**** Recipe Generator

### ****Objective:****

A web application that allows users to enter a list of ingredients and get the top n matching recipes based on ingredient similarity. Users can also filter results based on dietary preferences and cuisine type. If no close match is found in the database, a query is sent to an LLM for potential recipes.

## User Story

﻿﻿As a user, I want to enter a list of ingredients, and get the top n recipes I can make using these ingredients. I want the closest match to my list of ingredients id the exact match is not available. I would also like to set my dietary preferences and the type of cuisine for my ingredients I input.

## **2. System Architecture**

The system follows a **modular architecture** consisting of the following components:

### ****2.1 Frontend (User Interface)****

* **Technology:** Streamlit
* **Purpose:**
  + Accepts user input (ingredients, preferences, cuisine type)
  + Displays recipe recommendations
  + Allows users to bookmark recipes and view history
  + Handles user authentication

### ****2.2 Backend****

* **Technology:** Python (FastAPI for API endpoints)
* **Components:**
  1. **Recipe Search Engine** – Uses FAISS for fast similarity search
  2. **Recipe Database** – Stores structured recipe metadata
  3. **User Management System** – Handles authentication and bookmarks
  4. **LLM Integration** – Queries an external LLM if no match is found in FAISS

### ****2.3 Database Layer****

* **Primary Storage:** PostgreSQL (for metadata, user data, bookmarks)
* **Vector Search:** FAISS (stores recipe embeddings for similarity search)
* **Storage Strategy:**
  + FAISS embeddings persist on disk (recipe\_faiss.index)
  + Metadata stored in PostgreSQL
  + User bookmarks and preferences stored in PostgreSQL

## **3. Data Flow**

1. **User Input:** User enters ingredients, selects cuisine type and dietary preferences
2. **FAISS Search:** Queries FAISS for the closest recipe matches
3. **Database Lookup:** Retrieves full recipe details from PostgreSQL
4. **LLM Fallback:** If no close match is found, fetches a recipe suggestion from an LLM
5. **Response to UI:** Displays results, allowing users to bookmark recipes

## **4. Data Storage & Schema**

### ****4.1 Recipes Table (PostgreSQL)****

CREATE TABLE recipes (

recipe\_id SERIAL PRIMARY KEY,

name TEXT,

ingredients TEXT,

instructions TEXT,

cuisine\_type TEXT,

dietary\_info TEXT,

url TEXT,

created\_at TIMESTAMP DEFAULT NOW()

);

### ****4.2 Recipe Embeddings (FAISS Stored on Disk)****

CREATE TABLE recipe\_embeddings (

recipe\_id INT PRIMARY KEY REFERENCES recipes(recipe\_id),

embedding VECTOR(768)

);

### ****4.3 User Table (For Authentication & Bookmarks)****

CREATE TABLE users (

user\_id SERIAL PRIMARY KEY,

username TEXT UNIQUE,

password\_hash TEXT,

email TEXT UNIQUE

);

### ****4.4 Bookmarks Table****

CREATE TABLE bookmarks (

bookmark\_id SERIAL PRIMARY KEY,

user\_id INT REFERENCES users(user\_id),

recipe\_id INT REFERENCES recipes(recipe\_id),

created\_at TIMESTAMP DEFAULT NOW()

);

## **5. Technology Stack**

| **Component** | **Technology** |
| --- | --- |
| Frontend | Streamlit |
| Backend | Python (FastAPI) |
| Database | PostgreSQL |
| Vector Search | FAISS |
| Scraping | Selenium |
| LLM API | OpenAI API (or local LLM) |

## **6. Scalability Considerations**

* **FAISS on Disk:** Avoids RAM overload when scaling beyond 5K recipes
* **PostgreSQL Indexing:** Optimized queries for fast retrieval
* **Asynchronous API Calls:** To prevent blocking operations
* **Batch Processing for Scraping:** Efficient data storage and updates

## **7. Deployment Plan**

* **Local Development:** Dockerized PostgreSQL & FAISS
* **Cloud Deployment:** Host on AWS/GCP with managed PostgreSQL
* **API Hosting:** FastAPI on EC2 or Google Cloud Run
* **CI/CD:** GitHub Actions for automated deployment

## **8. Future Enhancements**

* Add **more vector dimensions** for improved similarity search
* Implement **user rating system** for recipe ranking
* Introduce **collaborative filtering** for personalized recommendations
* Improve **LLM integration** for generating missing recipes

**Author:** Zorawar Jaiswal  
**Date:** 02/26/2025  
**Version:** 1.0