AI-Powered Personalized Recipe Maker

Introduction

In today's fast-paced world, especially among students, bachelors, and working professionals, meal preparation is often a challenge due to limited time, cooking skills, and ingredient availability. Traditional recipe platforms are static and generic, failing to adapt to individual preferences, dietary needs, or what ingredients are actually on hand. As a result, users are often overwhelmed or resort to unhealthy or repetitive food choices.

With the growing capabilities of Large Language Models (LLMs) and personalized AI assistants, there is a clear opportunity to reimagine how we cook and interact with recipes. This project proposes a Generative AI-based Recipe Maker—an intelligent chat-based assistant that recommends customized recipes based on the user's available ingredients, time constraints, dietary preferences, cooking history, and skill level.

Unlike typical recipe apps, our system leverages user-specific profiles and memory to provide dynamic, context-aware meal suggestions. It also records feedback after meals, allowing the model to adapt and evolve with each interaction. By integrating personalization, memory, and generative AI, this project aims to reduce decision fatigue, minimize food waste, and make healthy, satisfying cooking accessible to everyone, regardless of expertise.

Workflow

1. User

The user interacts through the UI, providing input such as available ingredients, cooking time, dietary restrictions, and occasionally feedback on past recipes. Each user is uniquely identified by a profile, allowing the system to tailor its suggestions to individual tastes and skill levels.

2. UI (User Interface)

The UI serves as the central communication hub between the user and backend services. It gathers inputs (ingredients, preferences), displays generated recipes, and captures user feedback. It also fetches user-specific memory from the database to present a personalized experience.

3. Profile & Memory

Each user has a profile storing cooking skill level, dietary choices, preferred cuisines, past meals, and feedback. This memory is fetched from and written back to a persistent database, enabling the system to continuously learn and adapt recipe suggestions over time.

4. Database

The database stores persistent data such as user profiles, feedback history, favorite meals, and usage patterns. It ensures that each session is personalized by maintaining memory across interactions. This data feeds into prompt generation and long-term personalization strategies.

5. Grocery List

The list of ingredients currently available to the user is either entered manually or pulled from an integrated smart fridge/inventory system. This list forms a core input to the prompt that is sent to the LLM, ensuring recipes are feasible with what's on hand.

6. Prompt Engineering

This module constructs optimized prompts by combining the grocery list, user profile, memory, and constraints (e.g. cooking time). It translates user context into a natural language query that guides the LLM to produce high-quality, personalized recipes.

7. LLM System

The language model (e.g., GPT-4o or similar) takes the engineered prompt and generates a recipe that fits the context. It can adapt instructions to user expertise, suggest substitutes, and offer cooking tips. The result is sent back to the UI for display.

8. Language Evaluation Module
Before showing the recipe to the user, it is passed through a post-processing step that evaluates the language quality. This module ensures:

- Clarity and readability (especially for beginners)
- Friendly and instructive tone
- Logical order of instructions
- Suitability for user's profile (e.g., avoids complex terms for novices)
  If the output doesn't meet the criteria, it triggers a prompt refinement for regeneration.

End-to-End Flow

Step 1: User Interaction

A user logs into the system. Their profile indicates they are vegetarian, prefer Indian cuisine, and are a beginner-level cook.

Step 2: Input via UI

The user provides the following:

- Grocery list: `potatoes, onions, tomatoes, paneer`
- Available time: `30 minutes`
- Dietary preference: `vegetarian`

Step 3: Profile & Memory Retrieval

The system fetches the user’s past preferences and feedback. It notes they disliked spicy dishes and previously rated a "paneer tikka" recipe 4/5.

Step 4: Prompt Engineering

Using the grocery list, time limit, user profile, and memory, a prompt is generated:

> "Suggest a vegetarian Indian recipe using potatoes, onions, tomatoes, and paneer that is mild in spice, takes under 30 minutes, and suitable for a beginner."

Step 5: LLM System Generation

The LLM responds with:

> “Paneer Bhurji – A quick, mildly spiced dish made by crumbling paneer and sautéing it with onions, tomatoes, and minimal spices. Total time: 25 mins. Serves: 2.”

It also provides step-by-step instructions tailored for a novice.

Step 6: Language Evaluation Module:

Evaluates the output of the LLM

Step 7: Display via UI

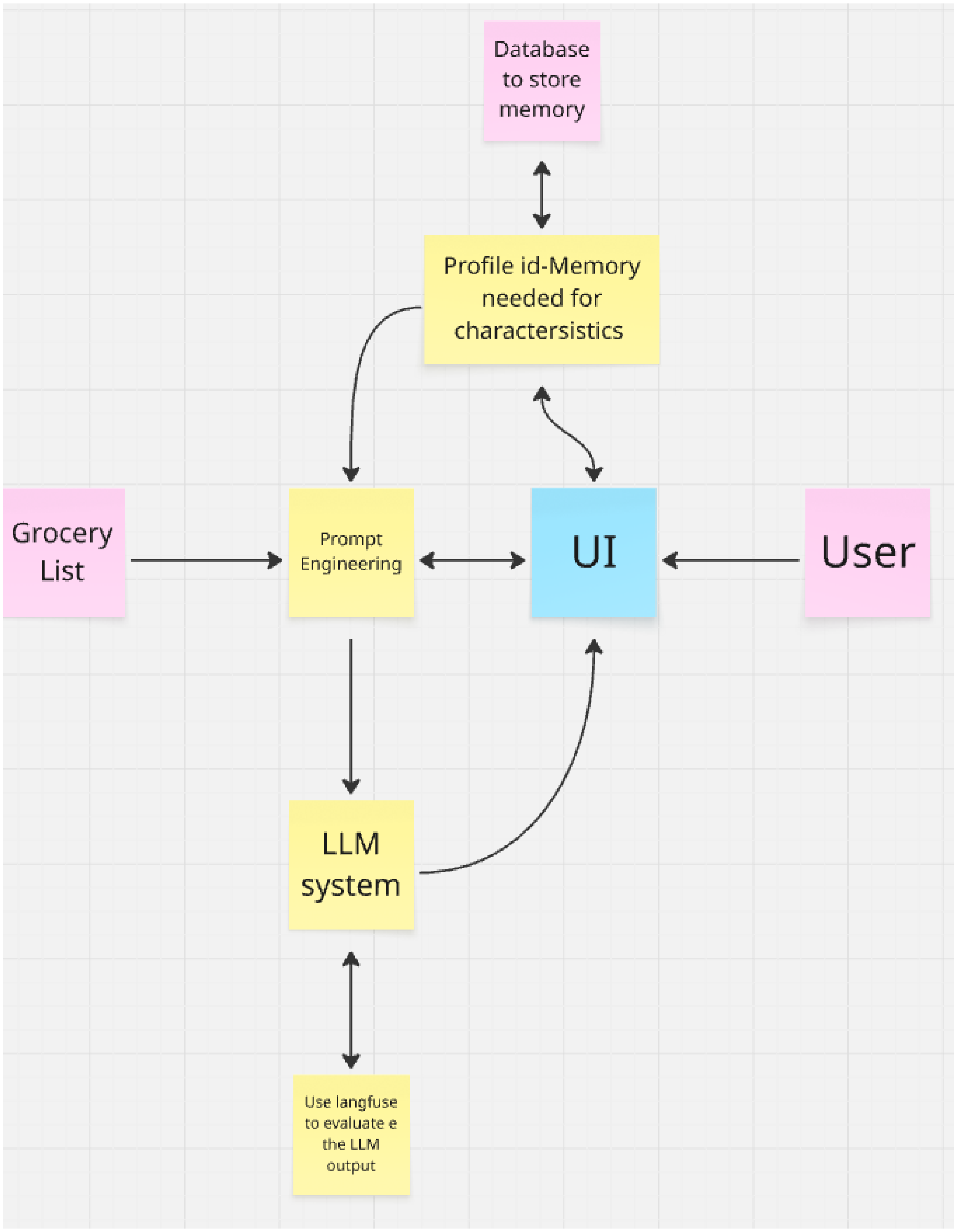
The UI displays the recipe, prep time, instructions, and a prompt asking:

“Did you like this recipe?”

Step 8: Feedback & Memory Update

User rates the recipe 5/5 and says, "Loved the simplicity." This feedback is stored in the database, updating the user’s profile and memory.

Architecture


Database to store memory
Profile id-Memory needed for charactersistics
Grocery List
Prompt Engineering
UI
User
LLM system
Use langfuse to evaluate e the LLM output