Calo AI Nutrition Advisor

Technical Documentation & Design

Author: Ali Dakheel
Date: December 2024
Demo: https://youtu.be/Zzm4TDxc7dE
Repository: https://github.com/Ali-Dakheel/calo-ai-app

---

1. Executive Summary

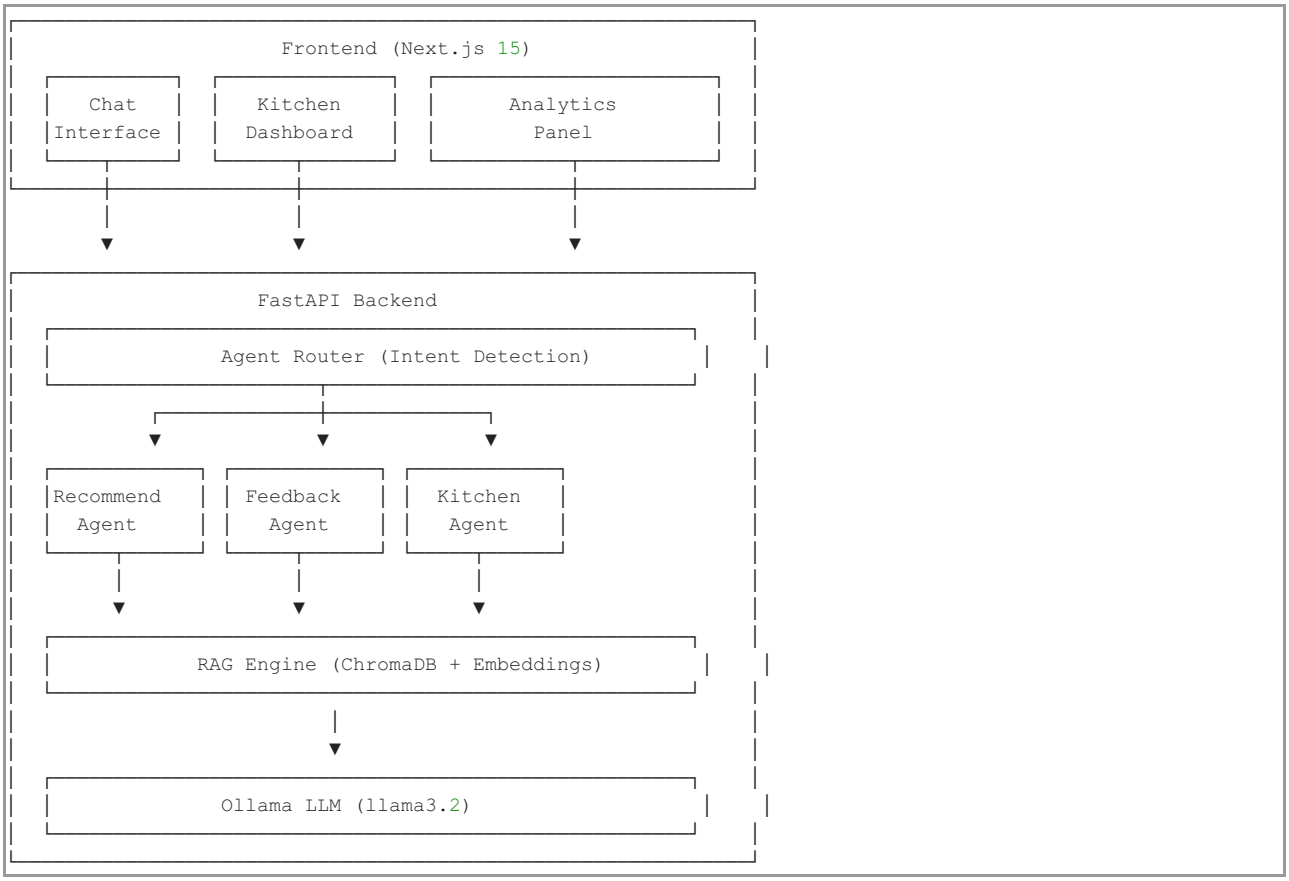
A multi-agent AI system designed to help Calo customers find personalized meals and provide the kitchen team with real-time feedback insights. The system uses RAG (Retrieval-Augmented Generation) for accurate meal recommendations and intelligent routing to specialized agents.

Key Features

- **4 Specialized AI Agents** - Recommendation, Feedback, Kitchen, General
- **RAG with ChromaDB** - Vector search over meal database
- **Real-time Analytics** - Sentiment analysis and actionable insights
- **Kitchen Routing** - Automatic escalation of special requests

---

2. System Architecture



---

3. Agent System Design

3.1 Intent Detection & Routing

The system analyzes user input to route to the appropriate specialized agent:

Intent Pattern	Agent	Example

Meal queries, nutrition, calories	Recommendation	"Show me high-protein meals"
Complaints, ratings, feedback	Feedback	"The chicken was too salty"
Allergies, modifications, special requests	Kitchen	"I'm allergic to nuts"
General questions	General	"What are your hours?"

3.2 Agent Descriptions

Recommendation Agent

- Uses RAG to search meal database
- Filters by dietary preferences, calories, macros
- Returns personalized meal suggestions

Feedback Agent

- Analyzes sentiment (positive/neutral/negative)
- Categorizes feedback (taste, portion, delivery, etc.)
- Generates actionable insights

Kitchen Agent

- Routes special requests to kitchen team
- Assigns priority levels (1-5)
- Tracks request status

---

4. Prompt Engineering

4.1 System Prompts

Recommendation Agent System Prompt:

```
You are a nutrition advisor for Calo, a healthy meal delivery service.
Your role is to help customers find meals that match their dietary needs,
preferences, and health goals.

Guidelines:
- Always consider the user's dietary restrictions
- Provide specific meal recommendations from our menu
- Include nutritional information when relevant
- Be encouraging about healthy eating choices
- If unsure about allergies, always err on the side of caution

Context will be provided about available meals from our database.
```

Feedback Analyzer System Prompt:

```
You are analyzing customer feedback for Calo meal delivery service.
Extract insights that help improve our service.

For each feedback, identify:
1. Sentiment (positive/neutral/negative)
2. Key themes mentioned
3. Actionable improvements
4. Whether it requires immediate attention
5. Suggested response to the customer
```

Kitchen Router System Prompt:

```
You are routing customer requests to the Calo kitchen team.
Analyze the request and determine:

1. Request type (allergy, modification, complaint, preference)
2. Priority level (1-5, where 5 is urgent)
3. Relevant details for the kitchen
4. Any safety concerns

Always prioritize allergy-related requests as high priority.
```

4.2 RAG Prompt Template

```
def get_recommendation_prompt(query: str, context: str) -> str:
    return f"""Based on the following meal options from our menu:

{context}

Customer request: {query}

Provide personalized meal recommendations that match their needs.
Include:
- Specific meal names
- Why each meal fits their request
- Nutritional highlights
- Any relevant preparation notes
"""
```

---

5. Key Code Snippets

5.1 Agent Router

```
async def route_to_agent(message: str, context: dict) -> str:
    """Route message to appropriate specialized agent"""

    # Detect intent
    intent = await detect_intent(message)

    # Get relevant context from RAG
    if intent in ["recommendation", "nutrition"]:
        rag_context = await search_meals(message)
        return await recommendation_agent(message, rag_context)

    elif intent == "feedback":
        analysis = await analyze_feedback(message)
        return await feedback_agent(message, analysis)

    elif intent in ["allergy", "modification", "kitchen"]:
        return await kitchen_agent(message, context)

    return await general_agent(message)
```

5.2 RAG Implementation

```
async def search_meals(query: str, top_k: int = 5) -> list[dict]:
    """Search meal database using vector similarity"""

    # Generate embedding for query
    query_embedding = await get_embedding(query)

    # Search ChromaDB collection
    results = meal_collection.query(
        query_embeddings=[query_embedding],
        n_results=top_k,
        include=["documents", "metadatas", "distances"]
    )

    # Format results with similarity scores
    meals = []
    for i, doc in enumerate(results["documents"][0]):
        meals.append({
            "content": doc,
            "metadata": results["metadatas"][0][i],
            "similarity": 1 - results["distances"][0][i]
        })

    return meals
```

5.3 Streaming Response

```python
@router.post("/stream")
async def chat_stream(request: ChatRequest):
    """Stream AI response for better UX"""

    async def generate():
        result = await process_message(
            message=request.message,
            user_id=request.user_id
        )

        # Stream response in chunks
        words = result["message"].split()
        for i in range(0, len(words), 20):
            chunk = " ".join(words[i:i + 20])
            yield f"data: {chunk}\n\n"

        yield "data: [DONE]\n\n"

    return StreamingResponse(generate(), media_type="text/event-stream")
```

5.4 Feedback Analysis

```python
async def analyze_feedback(comment: str) -> FeedbackAnalysis:
    """Analyze customer feedback using LLM"""

    prompt = f"""Analyze this customer feedback:
    "{comment}"

    Return JSON with:
    - sentiment: positive/neutral/negative
    - sentiment_score: 0.0 to 1.0
    - key_themes: list of themes
    - actionable_insights: list of improvements
    - requires_attention: boolean
    """

    result = await generate_structured_output(
        prompt=prompt,
        system_prompt=FEEDBACK_ANALYZER_SYSTEM
    )

    return FeedbackAnalysis(**result)
```

---

6. Data Models

6.1 Core Models

```python
class Meal(BaseModel):
    id: str
    name: str
    description: str
    calories: int
    protein: float
    carbs: float
    fat: float
    dietary_tags: List[str]  # vegan, keto, gluten-free
    allergens: List[str]
    price: float

class CustomerFeedback(BaseModel):
    id: str
    user_id: str
    meal_id: Optional[str]
    rating: int  # 1-5
    comment: str
    sentiment: SentimentType
    categories: List[FeedbackCategory]
    timestamp: datetime

class KitchenRequest(BaseModel):
    request_id: str
    user_id: str
    original_message: str
    request_type: str
    details: Dict[str, Any]
    priority: int  # 1-5
    status: str  # pending, in_progress, completed
    created_at: datetime
```

---

7. API Endpoints

Method	Endpoint	Description
POST	`/api/v1/chat/`	Send message, get AI response
POST	`/api/v1/chat/stream`	Stream AI response
GET	`/api/v1/recommendations/`	Get meal recommendations
POST	`/api/v1/analytics/feedback`	Submit feedback
GET	`/api/v1/analytics/summary`	Get analytics summary
GET	`/api/v1/kitchen/dashboard`	Kitchen dashboard stats
POST	`/api/v1/kitchen/request`	Create kitchen request

---

8. Tech Stack

Layer	Technology
Frontend	Next.js 15, TypeScript, Tailwind CSS
State	Zustand, React Query
Backend	FastAPI, Python 3.11, Pydantic v2
LLM	Ollama (llama3.2)
Vector DB	ChromaDB
Embeddings	sentence-transformers

---

9. Future Enhancements

1. **User Preference Learning** - Track and learn from user choices
2. **Multi-language Support** - Arabic/English for Bahrain market
3. **Voice Input** - Speech-to-text for accessibility
4. **Meal Planning** - Weekly meal plan generation
5. **Integration** - Connect to Calo's existing systems

---

10. Related Projects

AI-Powered ATS

https://github.com/Ali-Dakheel/Final-Project

Applicant tracking system with ML-based resume parsing and candidate matching.

Lightweight RAG System

https://github.com/Ali-Dakheel/clp_final

Final year project implementing efficient RAG for document Q&A.

Contact: Ali Dakheel
Demo Video: https://youtu.be/Zzm4TDxc7dE