IronAI: Intelligent Fitness Agent

A Graph-Based LLM Architecture for Hybrid Fitness Coaching

Jhony Peñaherrera

December 4, 2025

Intelligent Agents Course - Final Project

The Problem: Analysis Paralysis in Fitness

Why Traditional Apps Fail

- Too rigid and formulaic
- No personalized motivation
- Poor natural language understanding

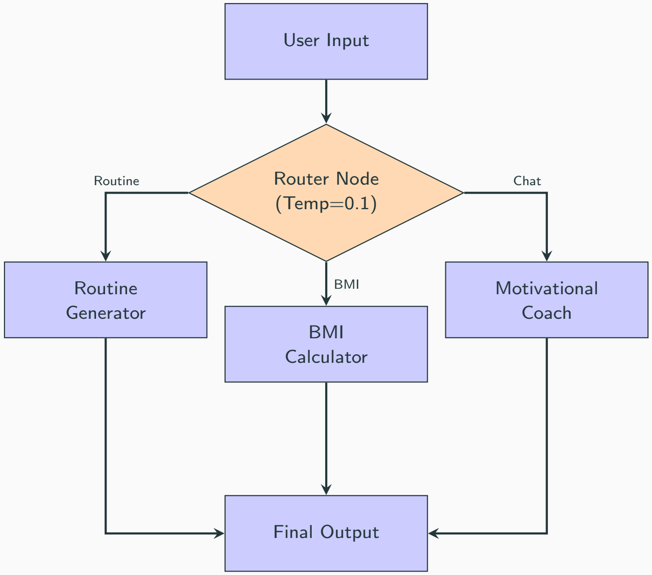
Why Generic LLMs Fail

- Hallucinate health calculations
- Inconsistent advice
- No specialization for fitness

IronAI Solution: Hybrid Intelligence

Combines **strict mathematical precision** (BMI, calories) with **creative motivational coaching** using temperature-controlled routing in a graph architecture.

System Architecture: Graph-Based Routing



Tech Stack: Python, LangChain, LangGraph, Ollama (Gemma 3:1b)

LangGraph: State Definition

Why State Management?
Multi-step workflows require persistent context across nodes

```python
from typing import TypedDict
from langgraph.graph import StateGraph

class AgentState(TypedDict):
    input_text: str          # User query
    intent: str              # Router classification
    user_data: dict          # Extracted parameters
    final_output: str        # Agent response
```

- **StateGraph**: Manages transitions between nodes
- **Conditional Edges**: Route based on `intent`
- **Persistence**: Each node reads/writes to shared state

The Router: Intent Classification with Precision

```python
def router_node(state: AgentState):
    prompt = """
    Classify intent as:
    - 'routine': workout plans
    - 'bmi': health calculations
    - 'chat': motivation/advice
    """

    llm = ChatOllama(
        model="gemma3-1b",
        temperature=0.1  # Precision!
    )

    intent = llm.invoke(prompt)
    return {"intent": intent}
```

Temperature Strategy

Router	`temp=0.1` *Deterministic classification*
Coach	`temp=0.7` *Creative responses*

Key LangGraph Feature: Conditional edges route to different nodes based on `state["intent"]`

Prompt Engineering: Few-Shot & Sequential Chains

Few-Shot Prompting for Personality (Chat Node)

- Define "IronCoach" persona: *Rude but motivating, like a drill sergeant*
- Provide 3-5 examples of typical exchanges in prompt template
- Result: Consistent personality across conversations

Sequential Chain (Routine Node)

Chain 1: Generate 5 exercises for target muscle group

$\downarrow$ *Output becomes input*

Chain 2: Recommend supplements based on exercises

Implemented using `LLMChain` with `PromptTemplate` from LangChain

All prompts isolated in `/prompts/templates.py` for maintainability

Tool Integration: BMI Calculator Node

Step 1: Extract Parameters

```python
from langchain.output_parsers
    import JsonOutputParser

parser = JsonOutputParser()
prompt = """
Extract weight and height:
Input: "I weigh 80kg and
        I'm 175cm tall"
Output: {"weight": 80,
         "height": 1.75}
"""
data = parser.parse(
    llm.invoke(prompt)
)
```

Step 2: Execute Tool

```python
# tools.py
def calculate_bmi_tool(
    weight: float,
    height: float
):
    bmi = weight / (height ** 2)

    if bmi < 18.5:
        return "Underweight"
    elif bmi < 25:
        return "Normal"
    else:
        return "Overweight"

# Call from node
result = calculate_bmi_tool(
    **data
)
```

Critical: `JsonOutputParser` ensures reliable extraction (no regex hacks!)

Production-Ready Code Structure

```
ironai/

          main.py
          requirements.txt

          src/
                nodes.py
                graph.py
                state.py
                tools.py

          prompts/
                templates.py

          logs/
              agent.log
```

Design Principles

- **Separation of Concerns**: Logic, prompts, and tools isolated
- **Testability**: Each node can be unit tested
- **Scalability**: Easy to add new nodes/intents
- **Logging**: Track routing decisions for debugging

Entry point (`main.py`) compiles graph and runs agent loop

Demonstration: Three Test Cases

1. **Routine Generation**
   - Input: *"Give me a chest routine for hypertrophy"*
   - Router → Routine Node (Sequential Chain)
   - Output: 5 exercises + supplement recommendations
2. **BMI Calculation**
   - Input: *"Calculate my BMI, I weigh 80kg and I'm 175cm"*
   - Router → BMI Node (Parameter Extraction + Tool)
   - Output: *"Your BMI is 26.1 (Overweight). Consider caloric deficit."*
3. **Motivational Coaching**
   - Input: *"I'm feeling tired today"*
   - Router → Chat Node (Few-Shot Personality)
   - Output: *"Tired? Your goals don't care! 10 push-ups NOW!"*

Key Success: Router correctly classifies intent in all cases (temp=0.1)

Conclusion: Why IronAI Works

Technical Achievements

- **LangChain**: Sequential chains, prompt templates, output parsers
- **LangGraph**: StateGraph with conditional routing
- **Prompt Engineering**: Few-shot for personality, structured extraction
- **Tool Integration**: Python function execution via LangChain
- **Temperature Control**: 0.1 (precision) vs 0.7 (creativity)

The Bigger Picture

Local Execution (Ollama) + **Graph Logic** (LangGraph) > Generic Chatbots

IronAI demonstrates that specialized agents with hybrid architectures
outperform general-purpose LLMs for domain-specific tasks.

[illegible]