Building Chatbots with Gemini and LangGraph

Shashank Rajak
AI Engineer

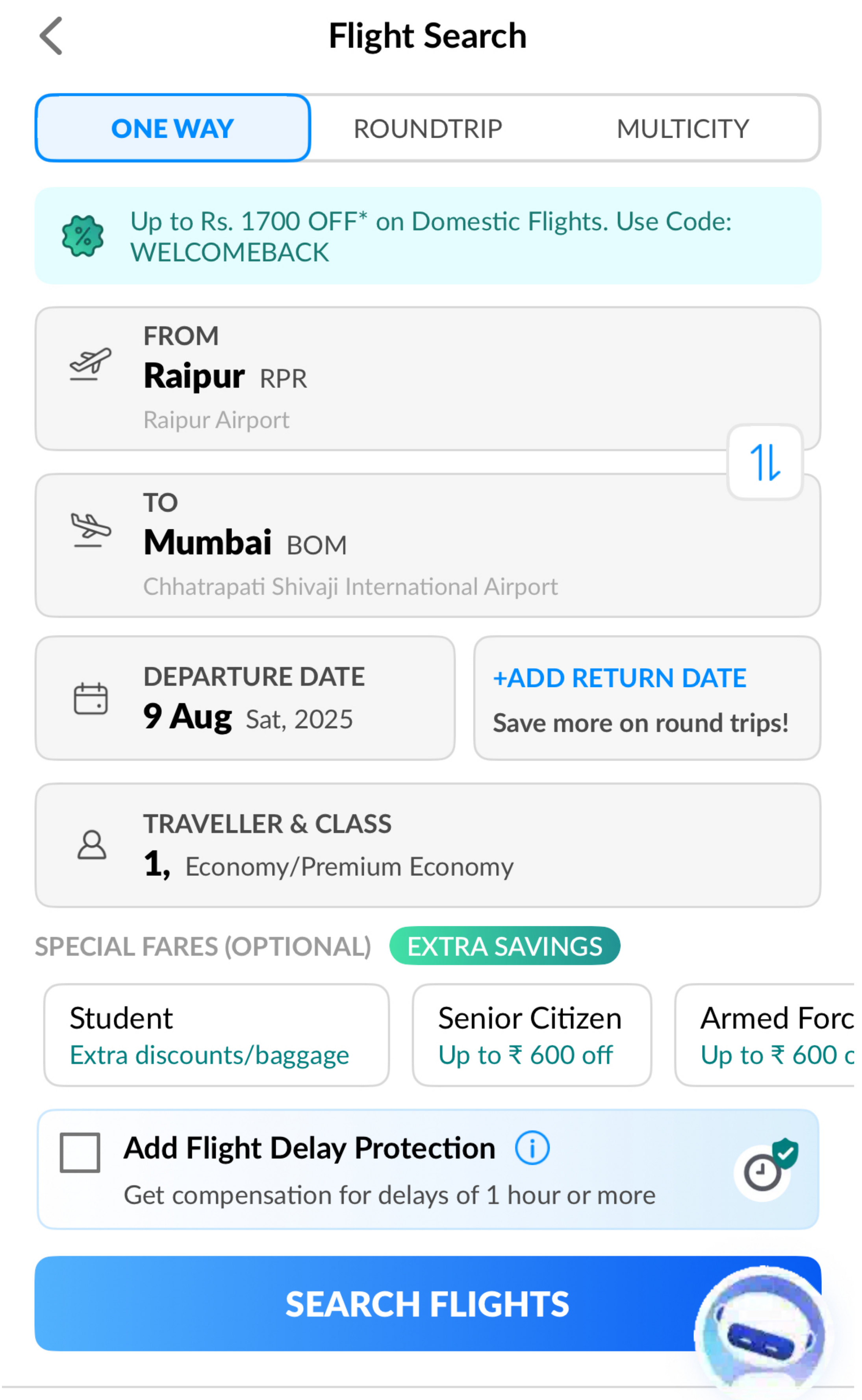
What we will learn

- Chatbots and their history
- LLMs and how to use them to build chatbots
- Concepts of building a chatbot
- Chatbot for a restaurant; code walkthrough

From Click to Chats

Click based UI

- Fixed Paths: Buttons, forms, menus.
- User Actions: Navigate, click and fill.



Flight Search

ONE WAY ROUNDTRIP MULTICITY

Up to Rs. 1700 OFF* on Domestic Flights. Use Code: WELCOMEBACK

FROM
Raipur RPR
Raipur Airport

TO
Mumbai BOM
Chhatrapati Shivaji International Airport

DEPARTURE DATE
9 Aug Sat, 2025

+ADD RETURN DATE
Save more on round trips!

TRAVELLER & CLASS
1, Economy/Premium Economy

SPECIAL FARES (OPTIONAL) EXTRA SAVINGS

Student
Extra discounts/baggage

Senior Citizen
Up to ₹ 600 off

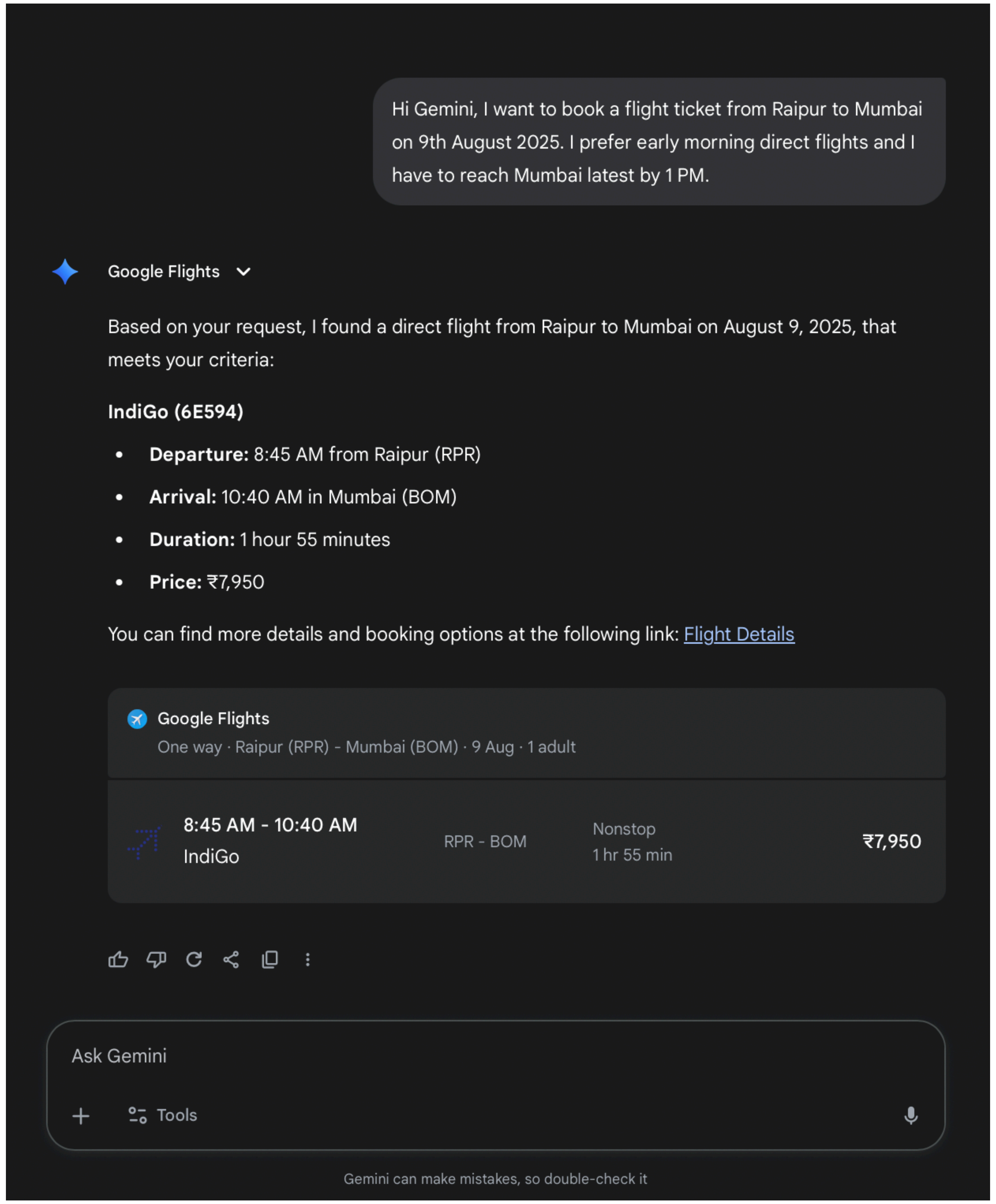
Armed Forc
Up to ₹ 600 c

☐ Add Flight Delay Protection
Get compensation for delays of 1 hour or more

SEARCH FLIGHTS

Chat based UI

- Natural Language: Chatbots, voice assistants.
- User Actions: Ask, describe, chat like you do with fellow humans.



Hi Gemini, I want to book a flight ticket from Raipur to Mumbai on 9th August 2025. I prefer early morning direct flights and I have to reach Mumbai latest by 1 PM.

Google Flights

Based on your request, I found a direct flight from Raipur to Mumbai on August 9, 2025, that meets your criteria:

IndiGo (6E594)

- **Departure:** 8:45 AM from Raipur (RPR)
- **Arrival:** 10:40 AM in Mumbai (BOM)
- **Duration:** 1 hour 55 minutes
- **Price:** ₹7,950

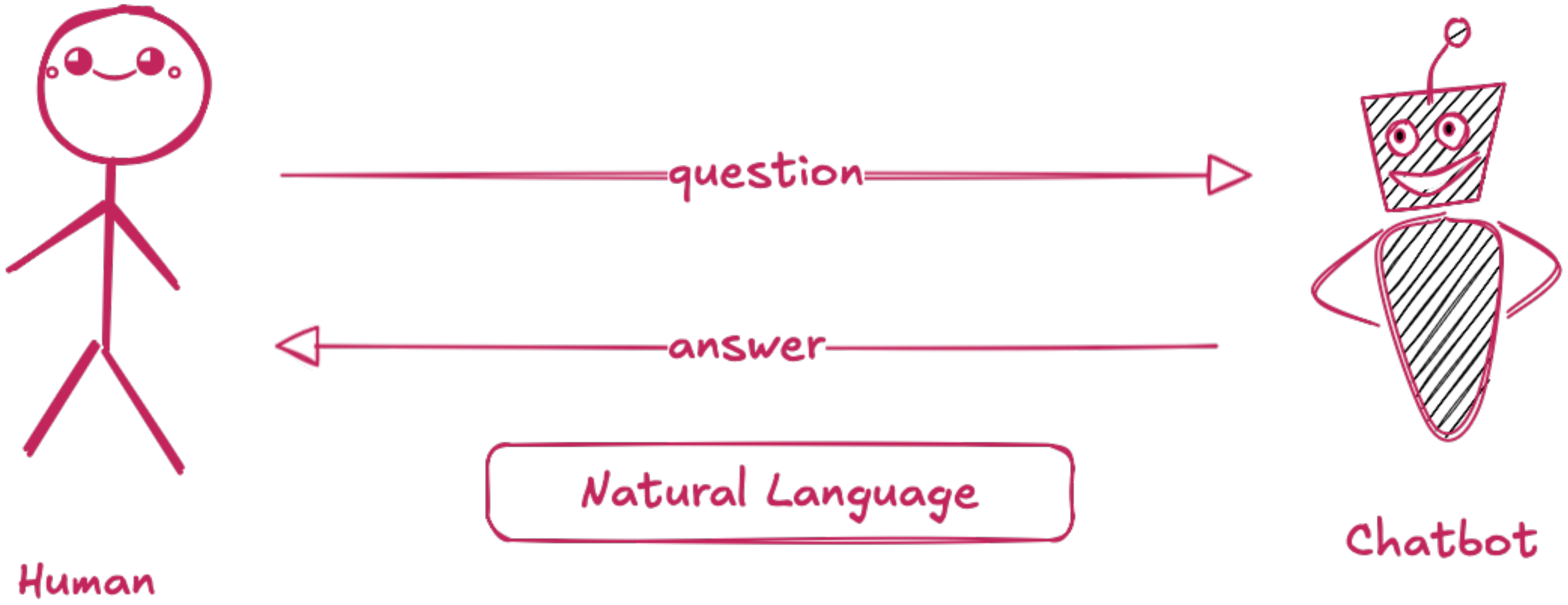
You can find more details and booking options at the following link: Flight Details

Google Flights
One way · Raipur (RPR) - Mumbai (BOM) · 9 Aug · 1 adult

8:45 AM - 10:40 AM
IndiGo
RPR - BOM
Nonstop
1 hr 55 min
₹7,950

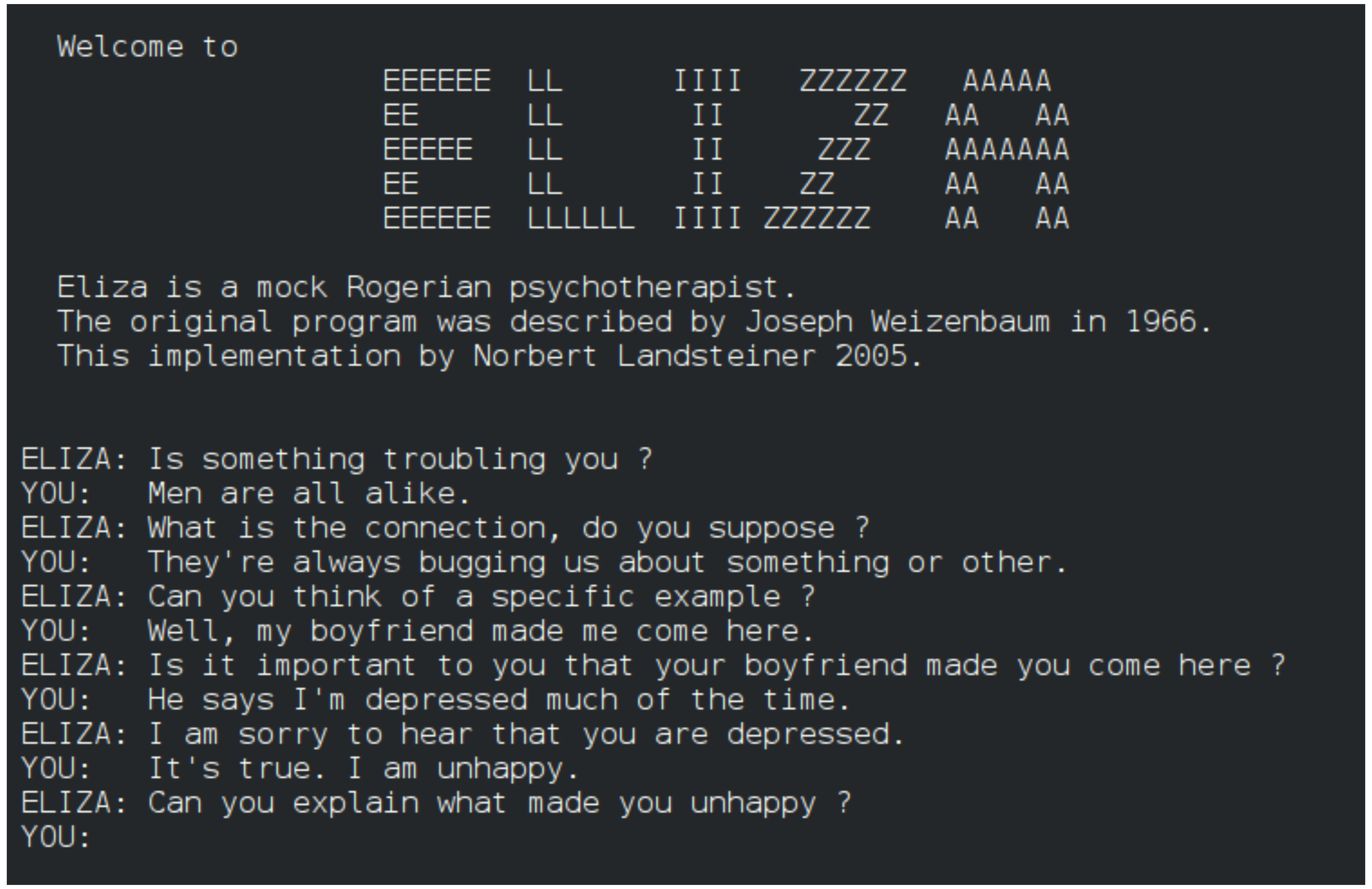
The Era of Traditional Chatbots

What is a Chatbot?



ELIZA

- Developed in 1966 at MIT, one of the first chatbot programs.
- Simulated conversation using Pattern Matching and Substitution Techniques.

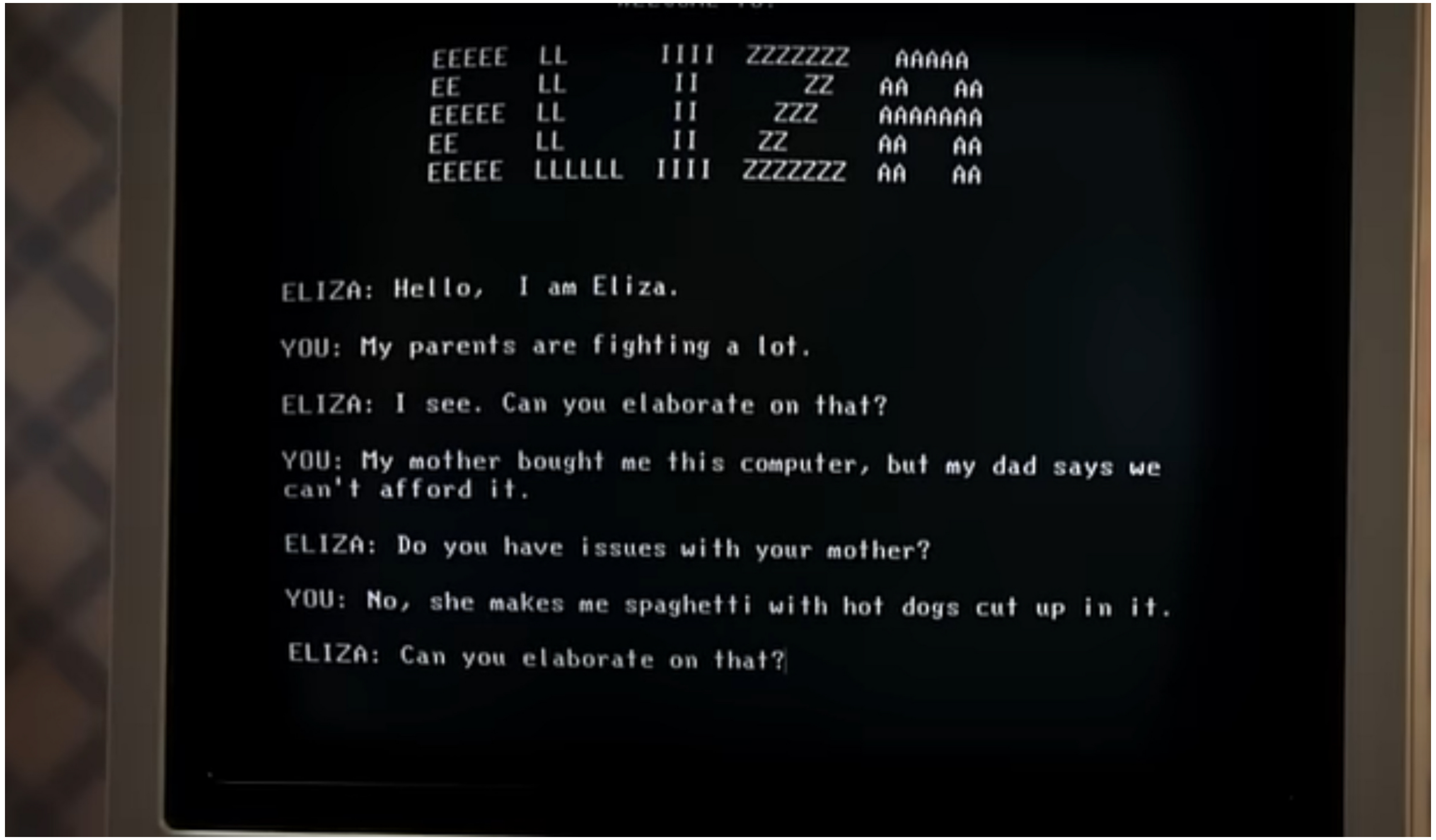


```
  Welcome to
                    EEEEEE  LL      IIII   ZZZZZZ   AAAAA
                    EE      LL       II       ZZ   AA   AA
                    EEEEE   LL       II     ZZZ    AAAAAAA
                    EE      LL       II    ZZ      AA   AA
                    EEEEEE  LLLLLL  IIII ZZZZZZ    AA   AA

  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

The Core Problem: Context is the King!

Source: Young Sheldon, S1E12, https://youtu.be/VdN6yXZ3cLM?si=tdr98JGWizQIOFvY

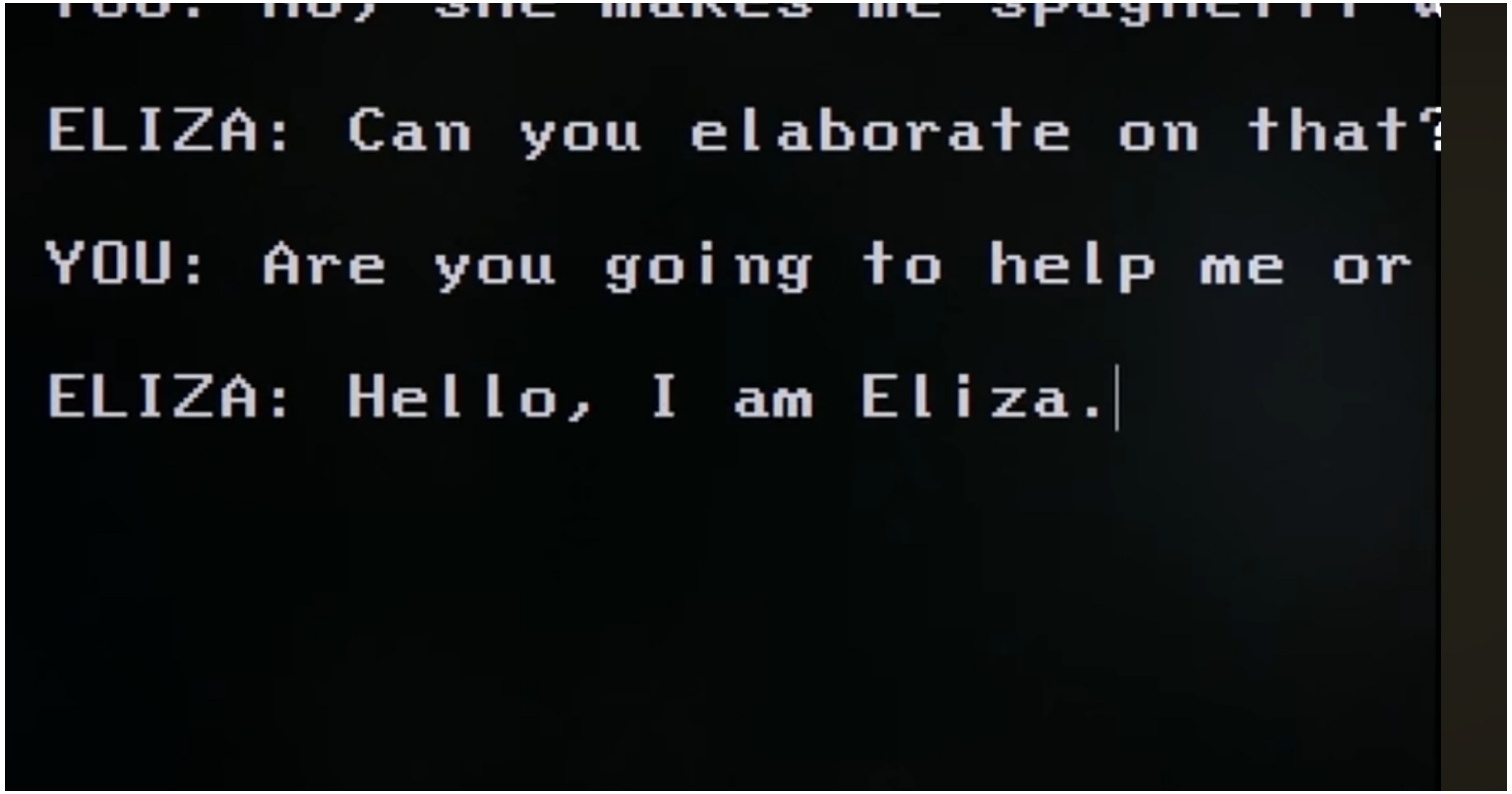
The Core Problem: Context is the King!



```
     EEEEE  LL      IIII  ZZZZZZZ   AAAAA
     EE     LL       II        ZZ  AA   AA
     EEEEE  LL       II      ZZZ   AAAAAAA
     EE     LL       II     ZZ     AA   AA
     EEEEE  LLLLLL  IIII  ZZZZZZZ  AA   AA


ELIZA: Hello,  I am Eliza.

YOU: My parents are fighting a lot.

ELIZA: I see. Can you elaborate on that?

YOU: My mother bought me this computer, but my dad says we
can't afford it.

ELIZA: Do you have issues with your mother?

YOU: No, she makes me spaghetti with hot dogs cut up in it.

ELIZA: Can you elaborate on that?
```

Source: Young Sheldon, S1E12, https://youtu.be/VdN6yXZ3cLM?si=tdr98JGWizQIOFvY

The Core Problem: Context is the King!

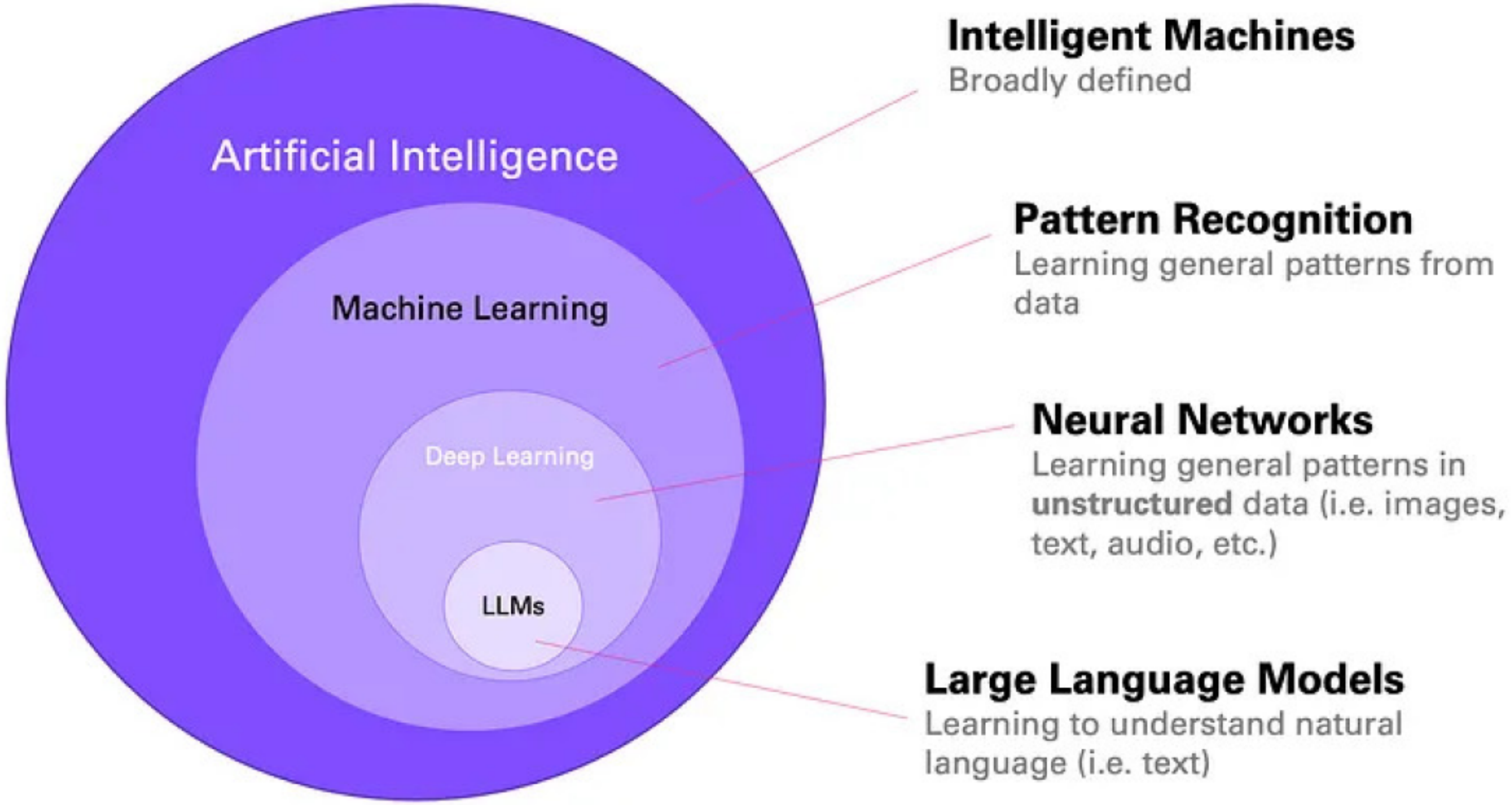


```
YOU: No, she makes me spaghetti w

ELIZA: Can you elaborate on that?

YOU: Are you going to help me or

ELIZA: Hello, I am Eliza.
```

Source: Young Sheldon, S1E12, https://youtu.be/VdN6yXZ3cLM?si=tdr98JGWizQIOFvY

Large Language Models (LLMs)

LLMs



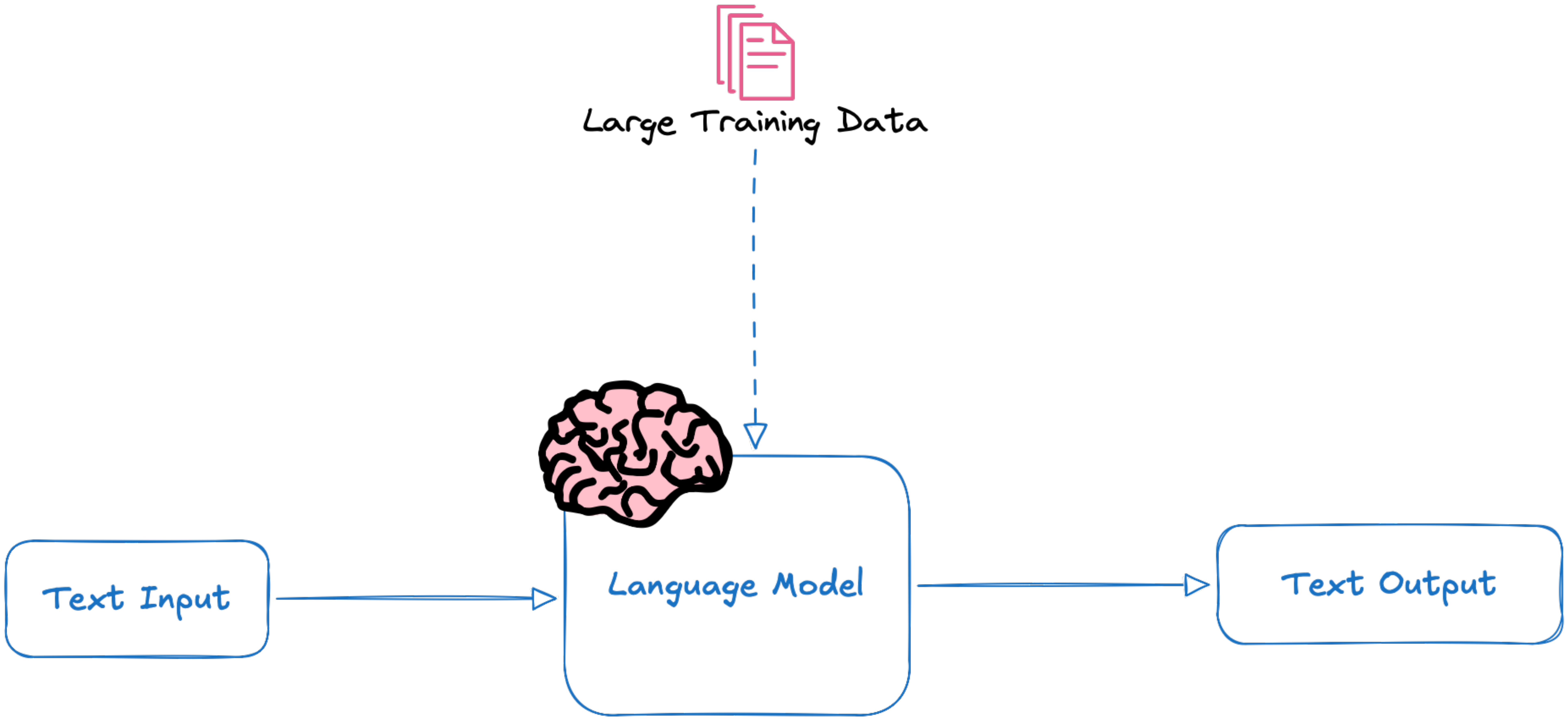
Intelligent Machines
Broadly defined

Pattern Recognition
Learning general patterns from data

Neural Networks
Learning general patterns in **unstructured** data (i.e. images, text, audio, etc.)

Large Language Models
Learning to understand natural language (i.e. text)

LLMs



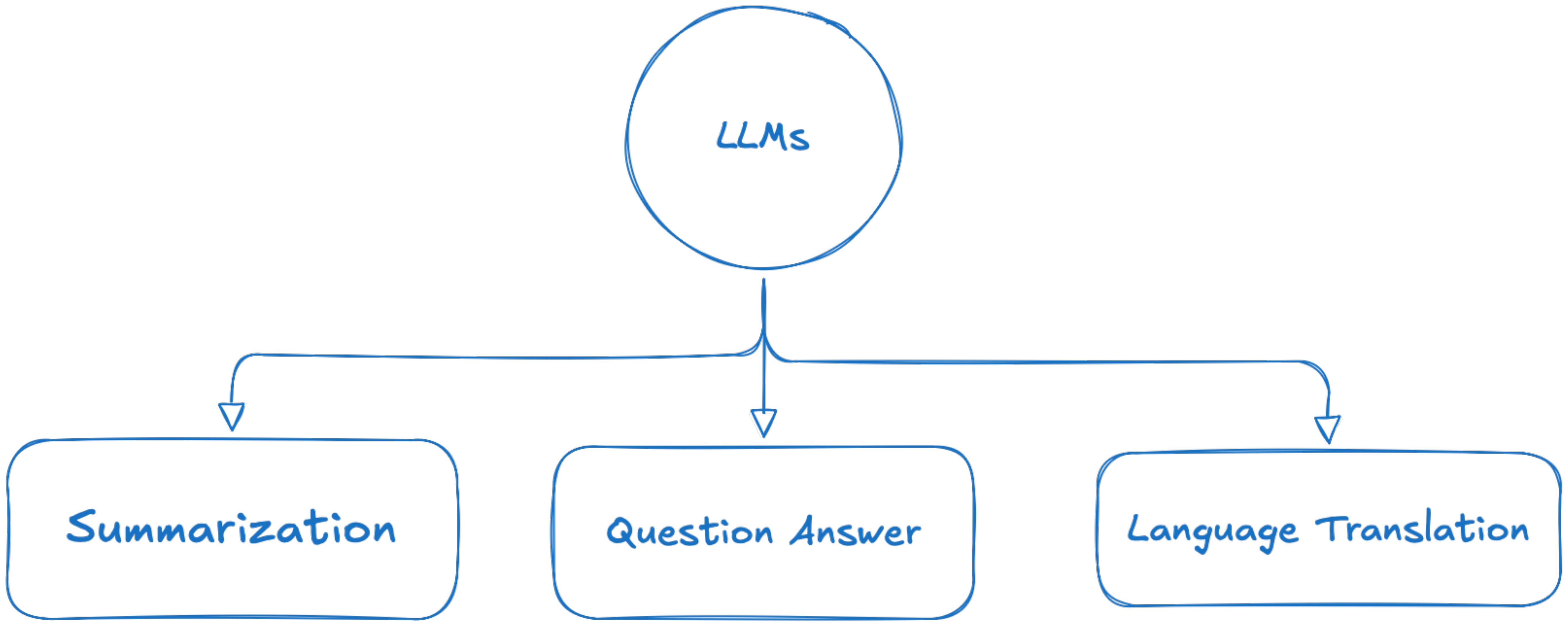
Large Training Data

Text Input

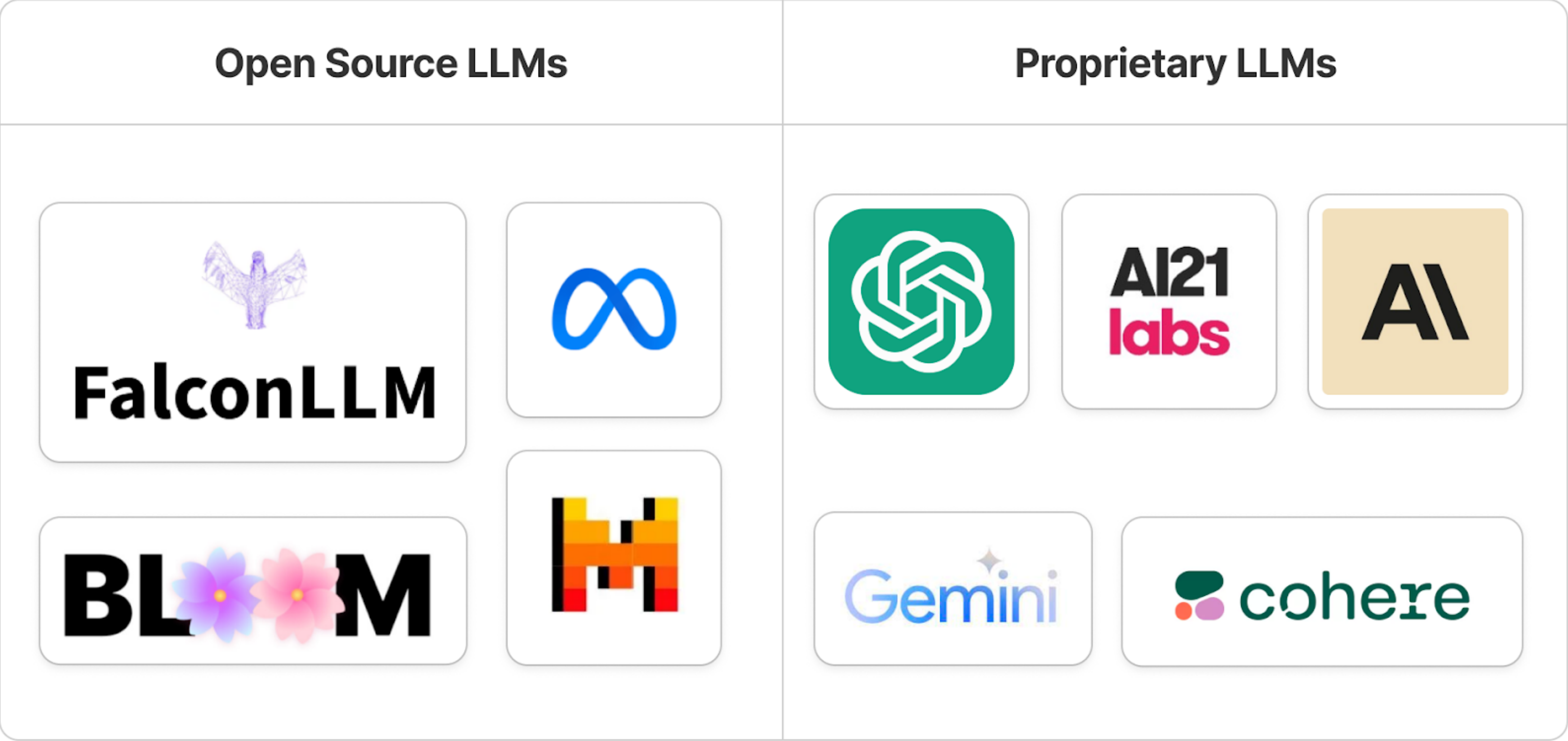
Language Model

Text Output

LLMs


LLMs
Summarization
Question Answer
Language Translation

LLMs



Open Source LLMs

FalconLLM

BLOOM

Proprietary LLMs

AI21 labs

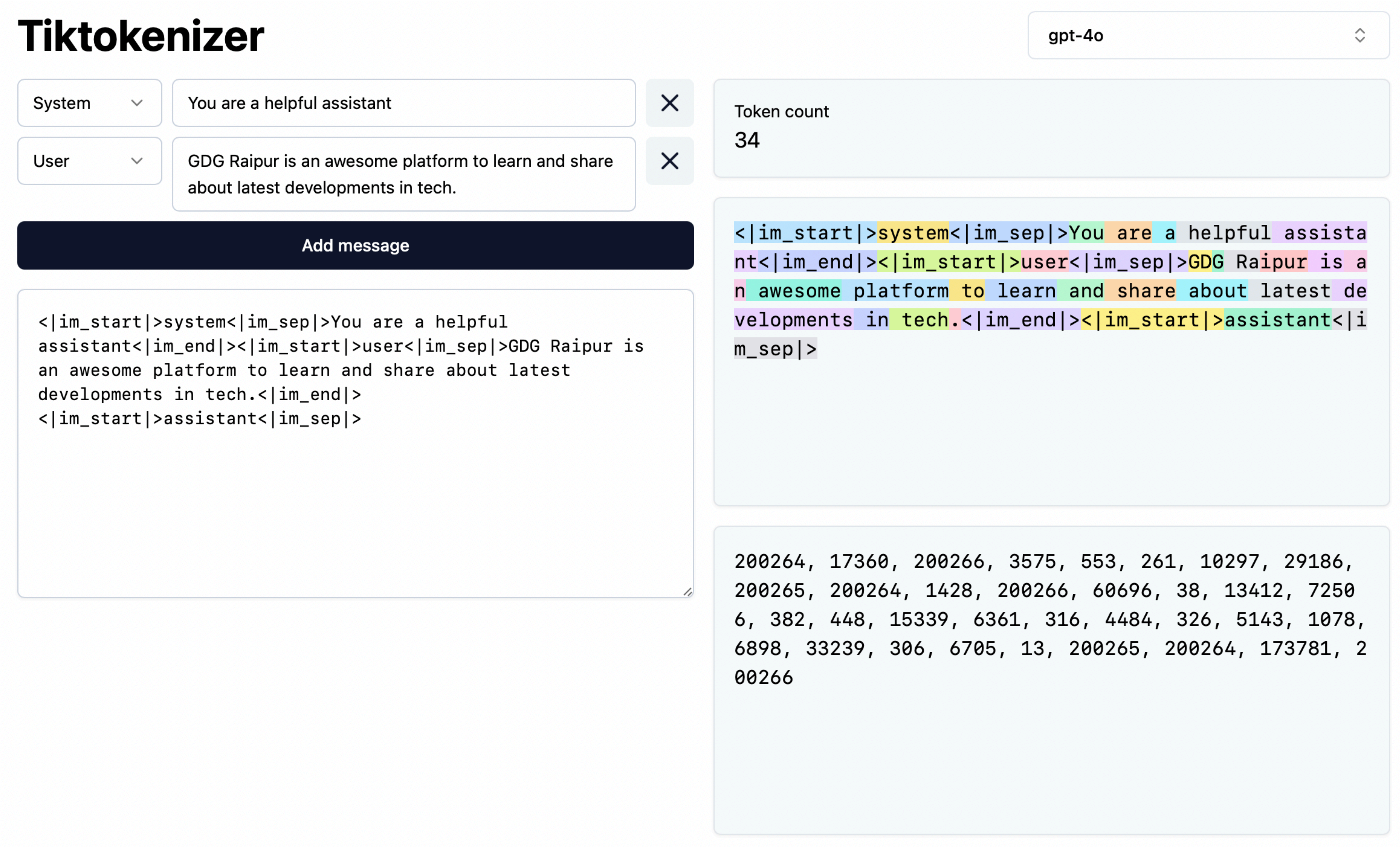
Gemini

cohere

Core concepts of building Chatbots with LLMs

Tokens

- Fundamental unit of LLMs
- You are charged on the basis of number of tokens



System
You are a helpful assistant

User
GDG Raipur is an awesome platform to learn and share about latest developments in tech.

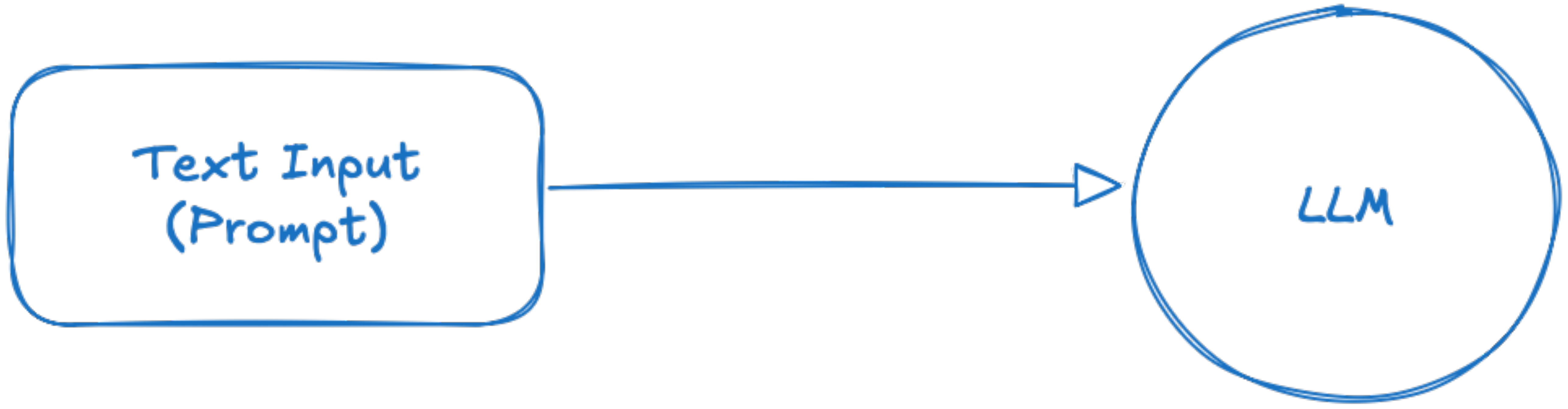
```
<|im_start|>system<|im_sep|>You are a helpful
assistant<|im_end|><|im_start|>user<|im_sep|>GDG Raipur is
an awesome platform to learn and share about latest
developments in tech.<|im_end|>
<|im_start|>assistant<|im_sep|>
```

Token count
34

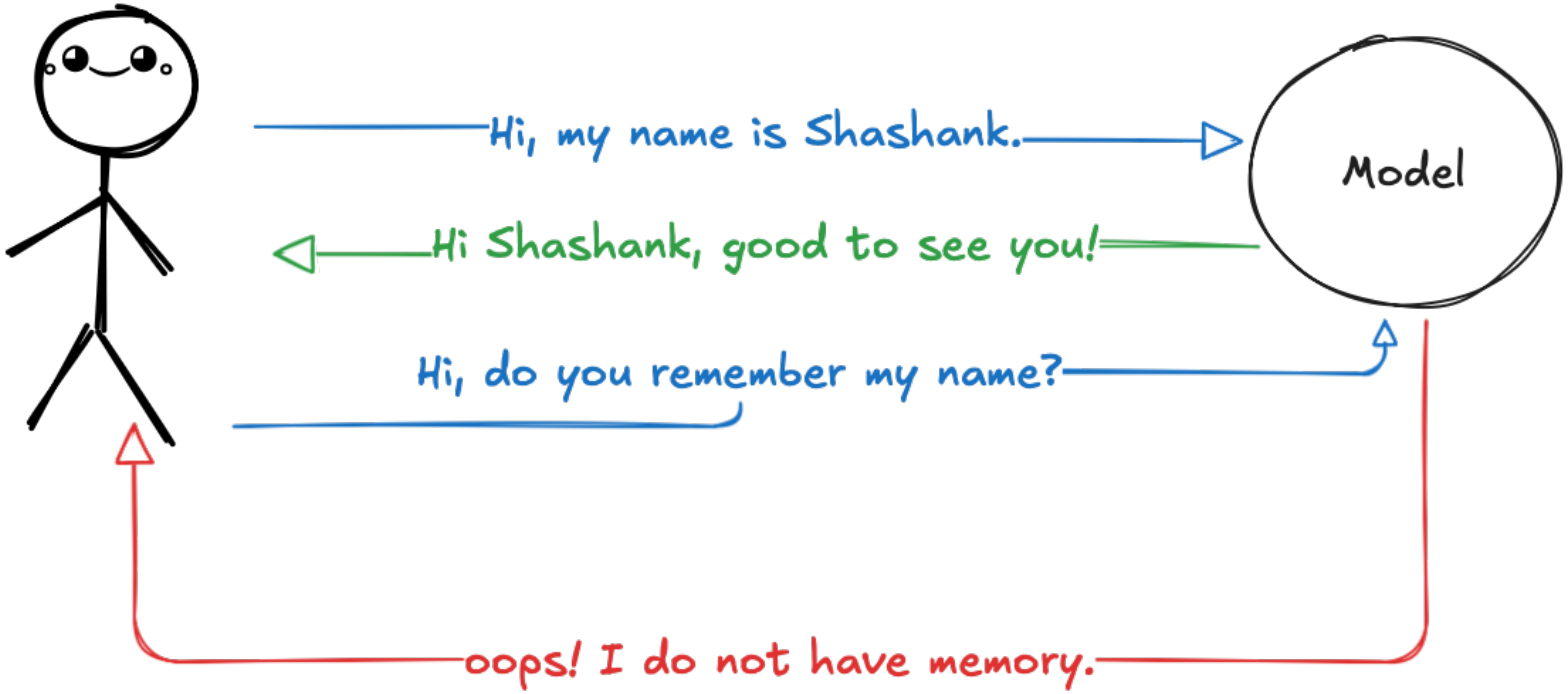
```
<|im_start|>system<|im_sep|>You are a helpful assista
nt<|im_end|><|im_start|>user<|im_sep|>GDG Raipur is a
n awesome platform to learn and share about latest de
velopments in tech.<|im_end|><|im_start|>assistant<|i
m_sep|>
```

```
200264, 17360, 200266, 3575, 553, 261, 10297, 29186,
200265, 200264, 1428, 200266, 60696, 38, 13412, 7250
6, 382, 448, 15339, 6361, 316, 4484, 326, 5143, 1078,
6898, 33239, 306, 6705, 13, 200265, 200264, 173781, 2
00266
```

Prompts

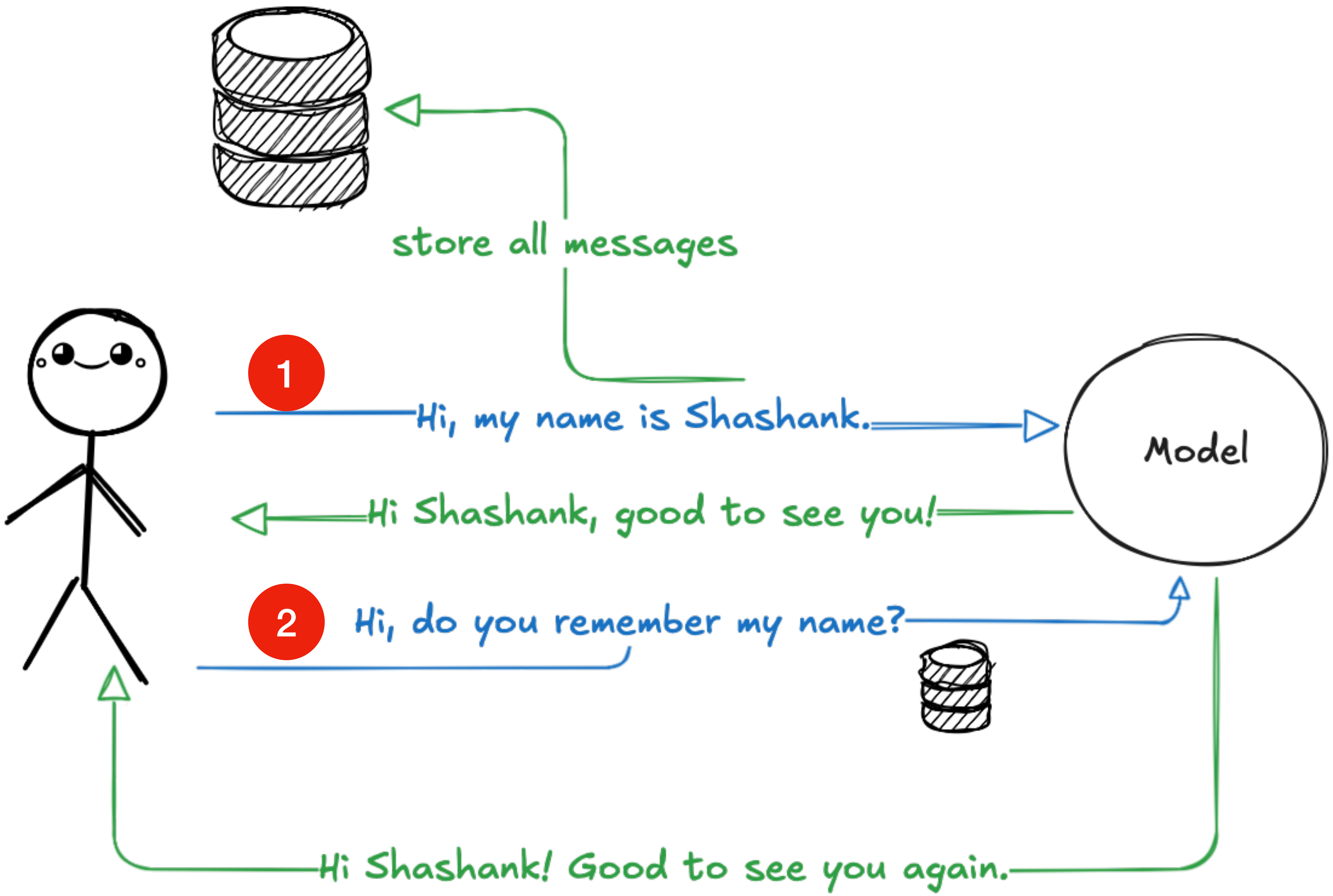


Memory / Context


Hi, my name is Shashank.
Model
Hi Shashank, good to see you!
Hi, do you remember my name?
oops! I do not have memory.

How to give this model a memory such that it remembers previous talks?

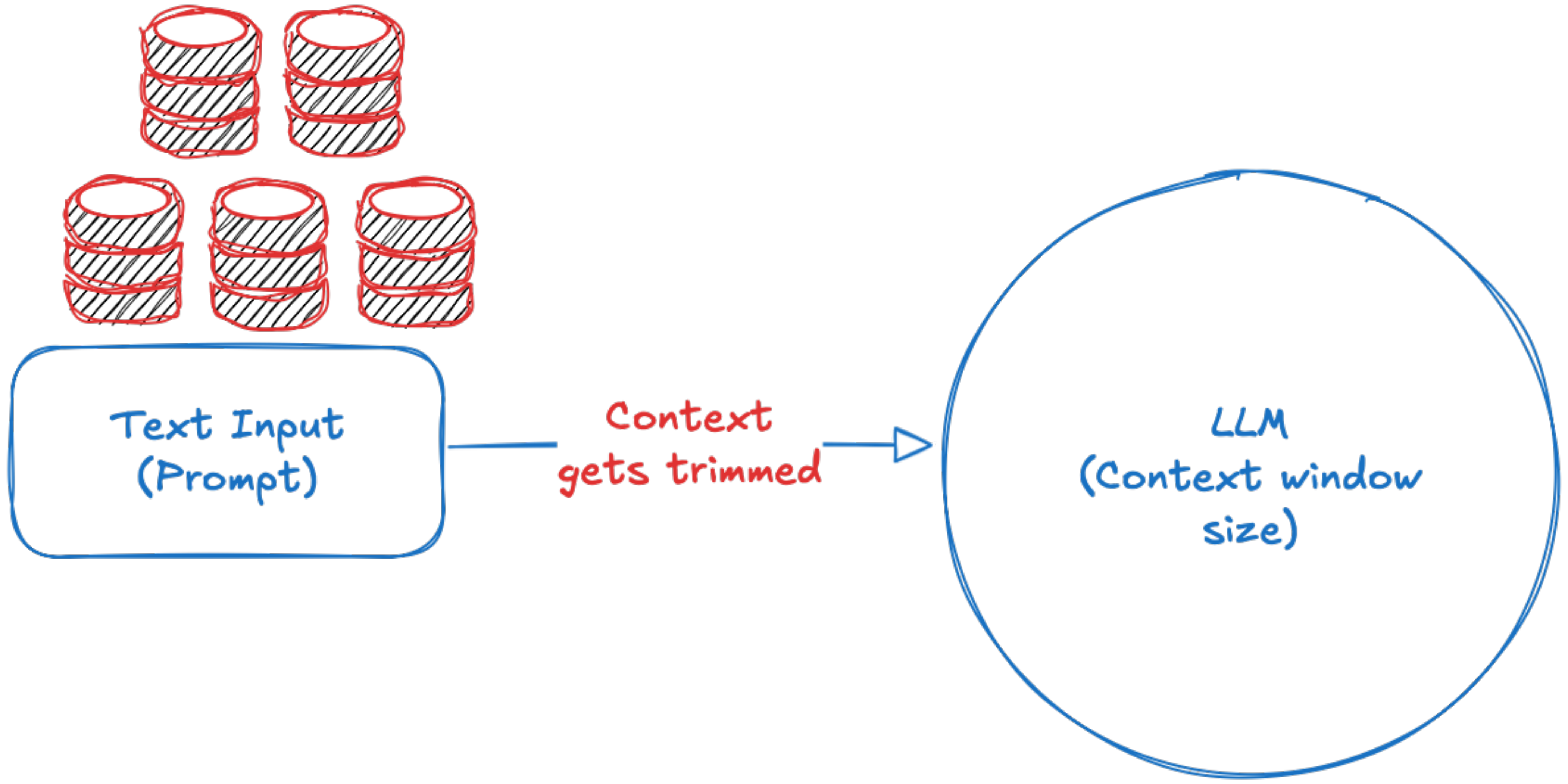
Memory / Context


store all messages
1
Hi, my name is Shashank.
Model
Hi Shashank, good to see you!
2
Hi, do you remember my name?
Hi Shashank! Good to see you again.

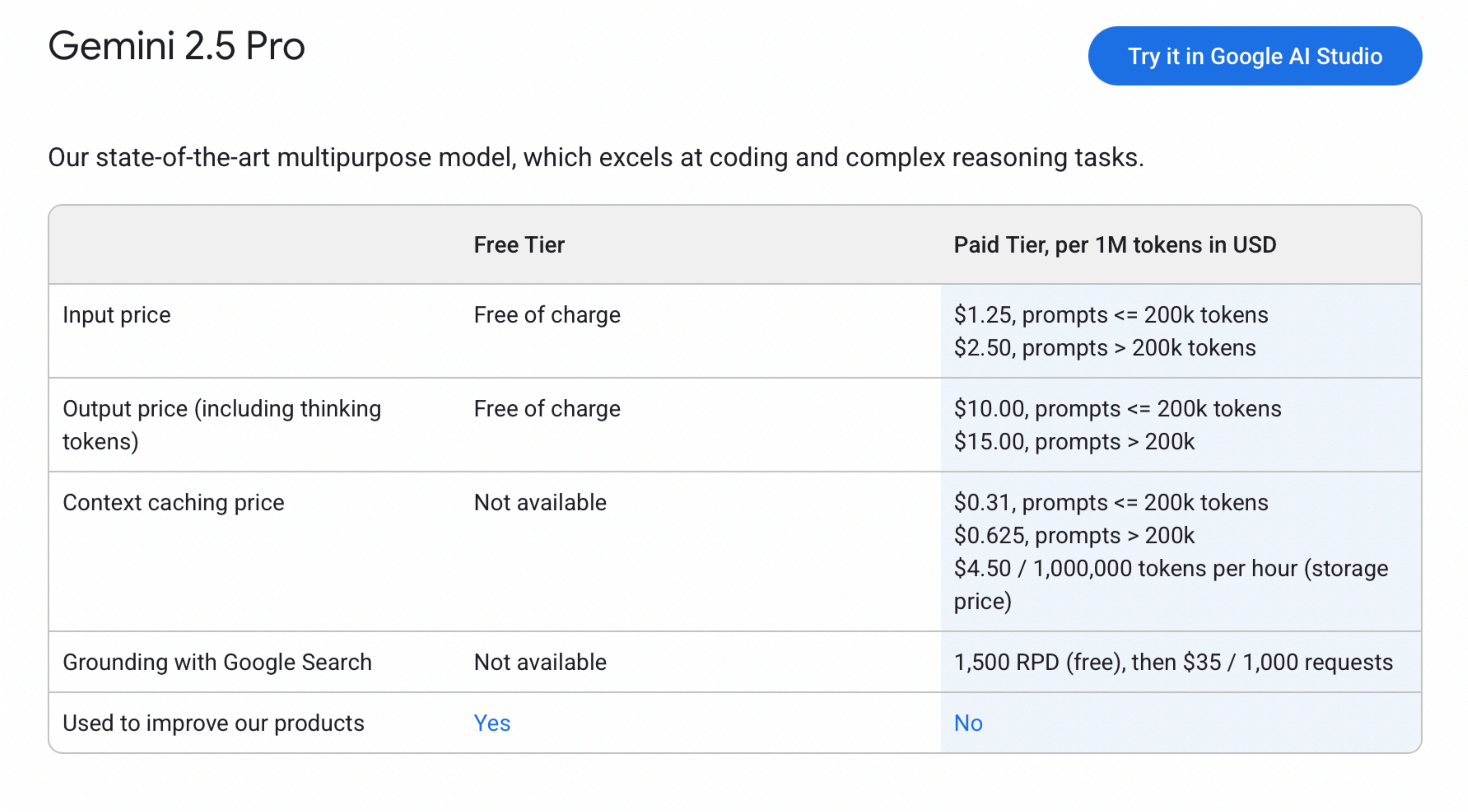
Challenges

Limited Context Window

- The amount of text (or tokens) that could be passed to the model at one time.
- Gemini models come with large context windows of 1 million or more tokens.



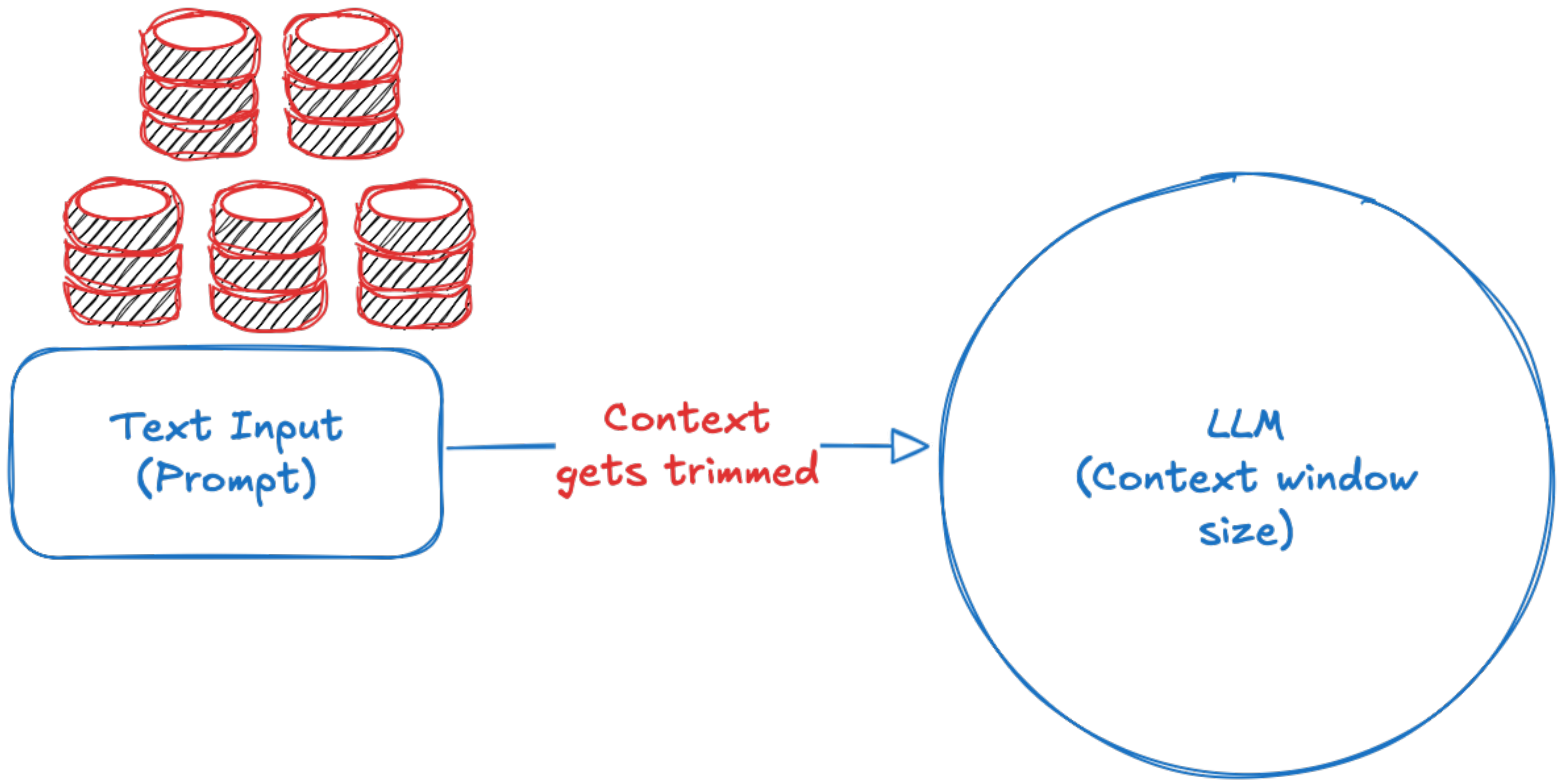
More Tokens means More Money



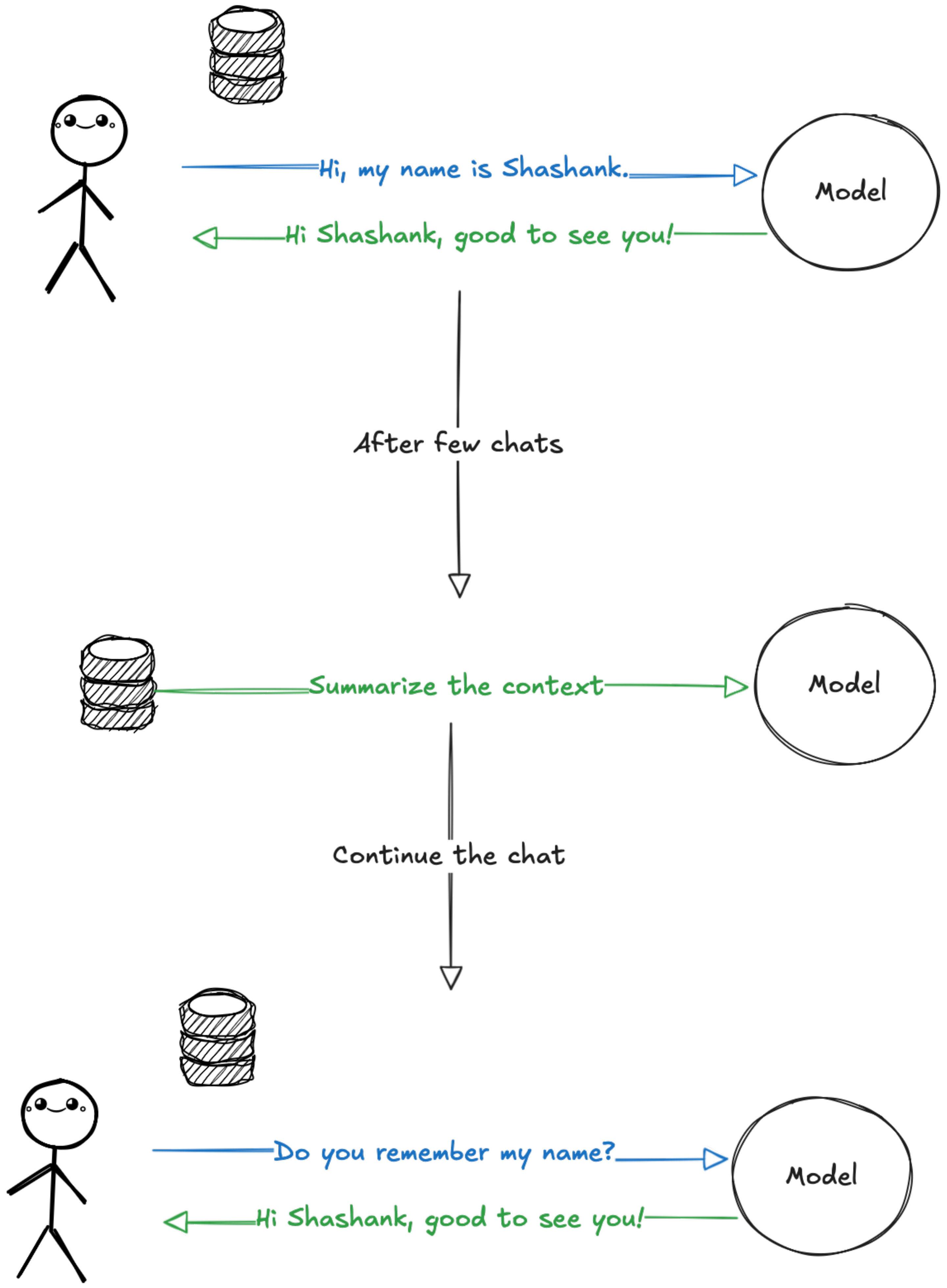
Gemini 2.5 Pro

Our state-of-the-art multipurpose model, which excels at coding and complex reasoning tasks.

	Free Tier	Paid Tier, per 1M tokens in USD
Input price	Free of charge	$1.25, prompts <= 200k tokens $2.50, prompts > 200k tokens
Output price (including thinking tokens)	Free of charge	$10.00, prompts <= 200k tokens $15.00, prompts > 200k
Context caching price	Not available	$0.31, prompts <= 200k tokens $0.625, prompts > 200k $4.50 / 1,000,000 tokens per hour (storage price)
Grounding with Google Search	Not available	1,500 RPD (free), then $35 / 1,000 requests
Used to improve our products	Yes	No



More Tokens means More Money



Hi, my name is Shashank.

Hi Shashank, good to see you!

After few chats

Summarize the context

Continue the chat

Do you remember my name?

Hi Shashank, good to see you!

LangGraph

The Problem

How do we build a chatbot that can:

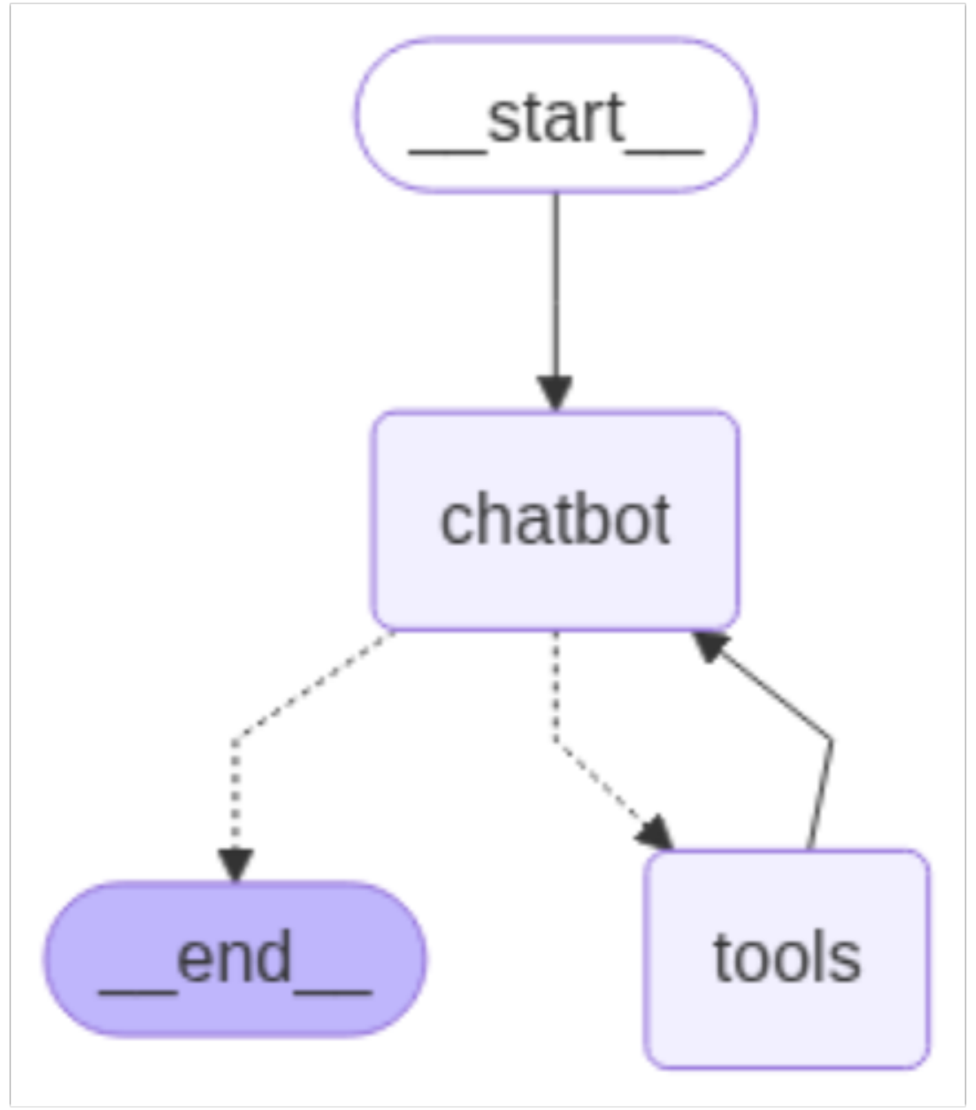
- Remember an entire conversation over many turns?
- Decide when to use a tool (like a database or an API)?
- Orchestrate a series of complex actions?

The Solution : LangGraph

- This is where we move from simple chatbots to **intelligent agents**.
- An agent has a "brain" that orchestrates the actions of an LLM. This is exactly what LangGraph is for.
- A library for building stateful, multi-step LLM applications.

LangGraph : Framework

Framework



Graph

Nodes

Edges

State

Tools

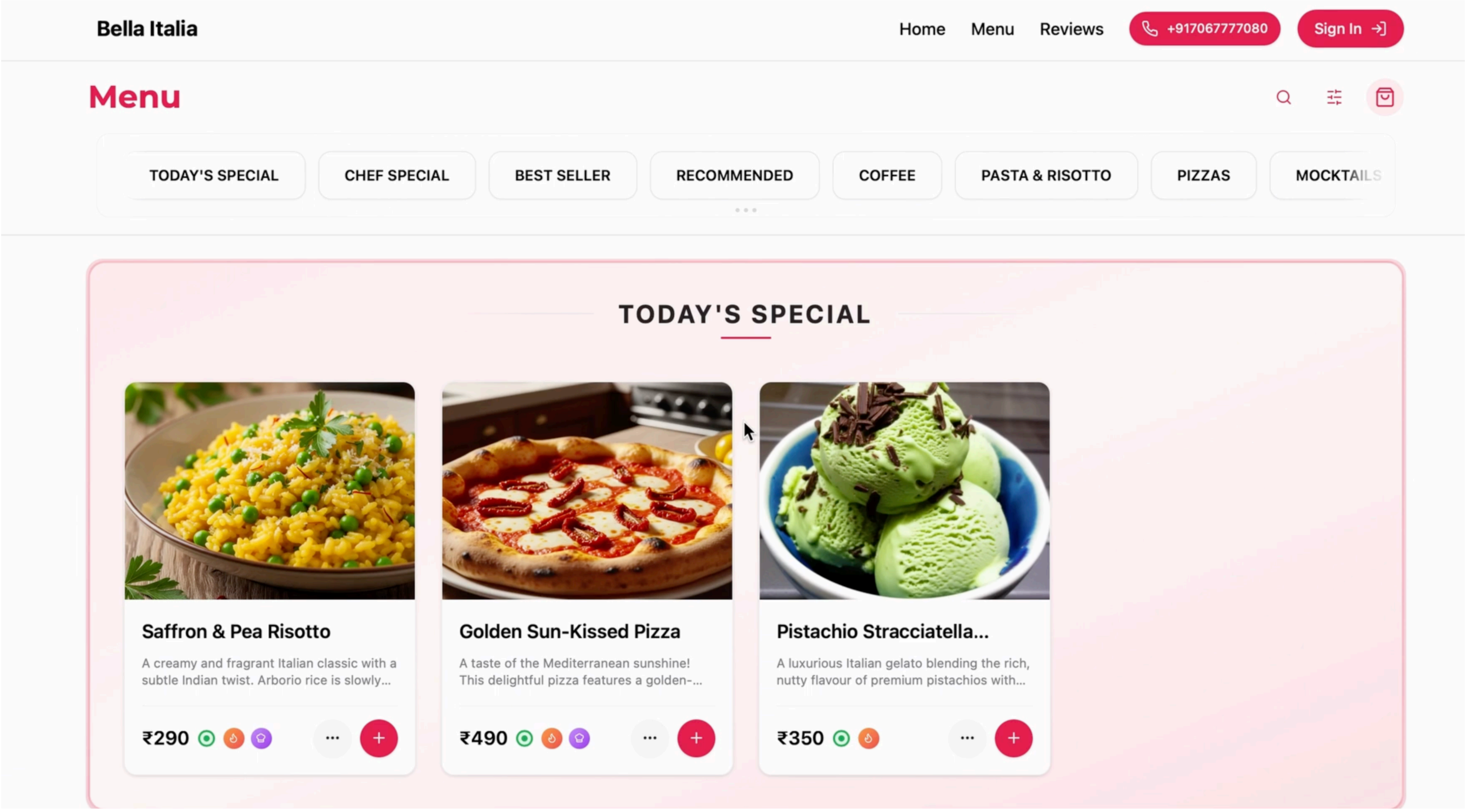
Memory

Streaming

Putting It All Together: The Restaurant Chatbot

The Problem

- How can we build an intelligent agent that helps a user with a restaurant's menu and can place an order?



Saffron & Pea Risotto

A creamy and fragrant Italian classic with a subtle Indian twist. Arborio rice is slowly...

₹290

Golden Sun-Kissed Pizza

A taste of the Mediterranean sunshine! This delightful pizza features a golden-...

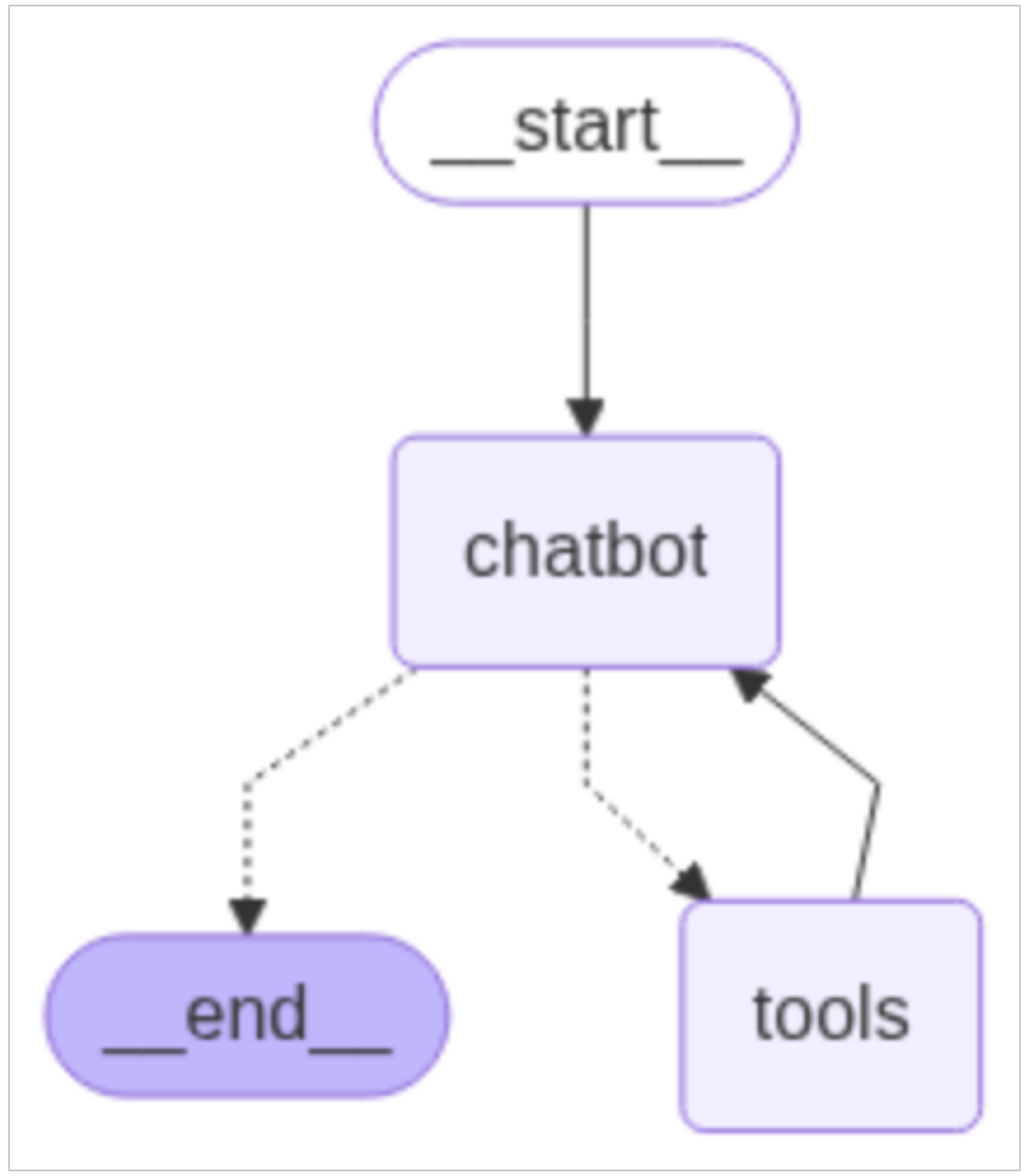
₹490

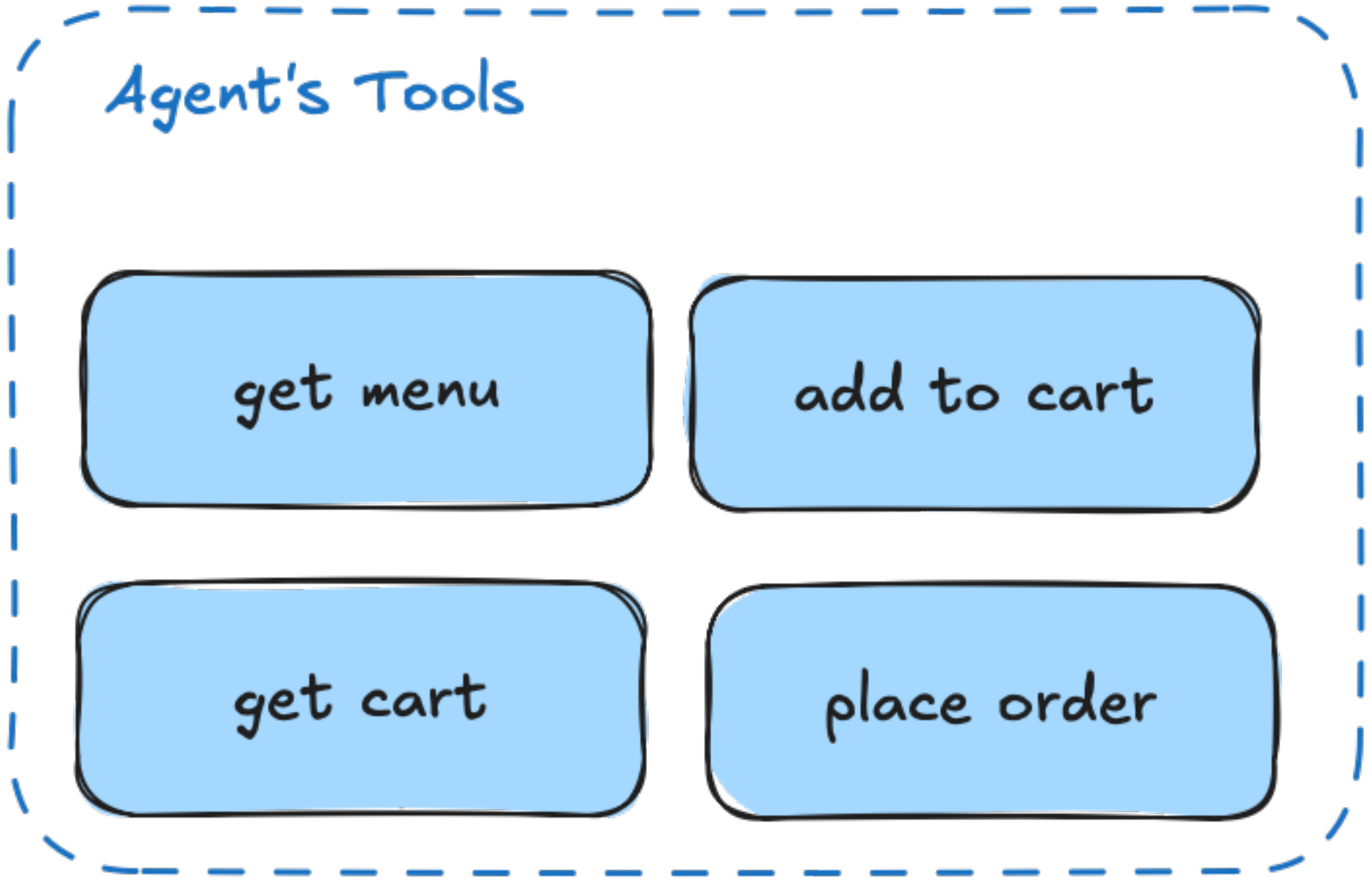
Pistachio Stracciatella...

A luxurious Italian gelato blending the rich, nutty flavour of premium pistachios with...

₹350

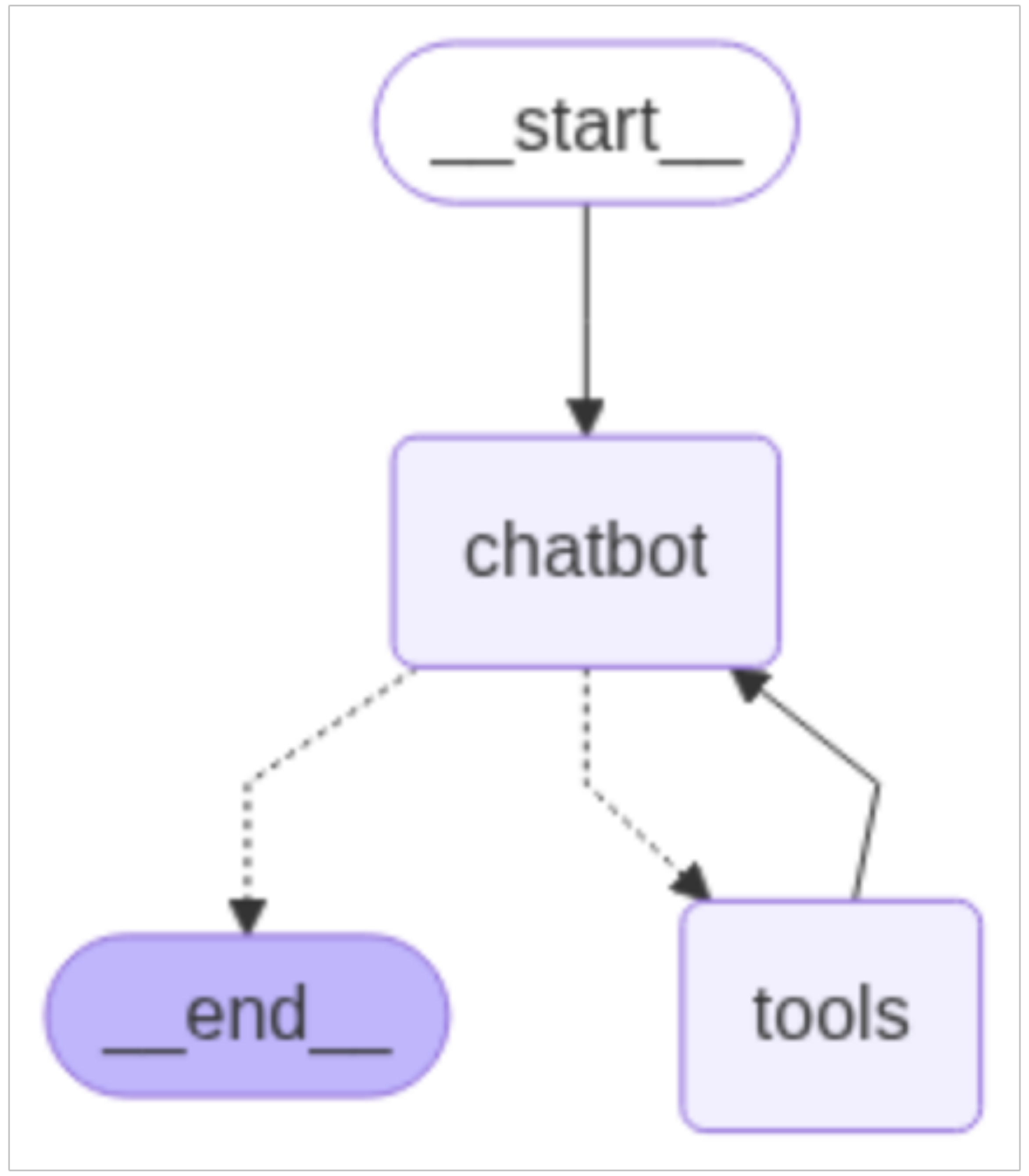
The Chatbot Agent

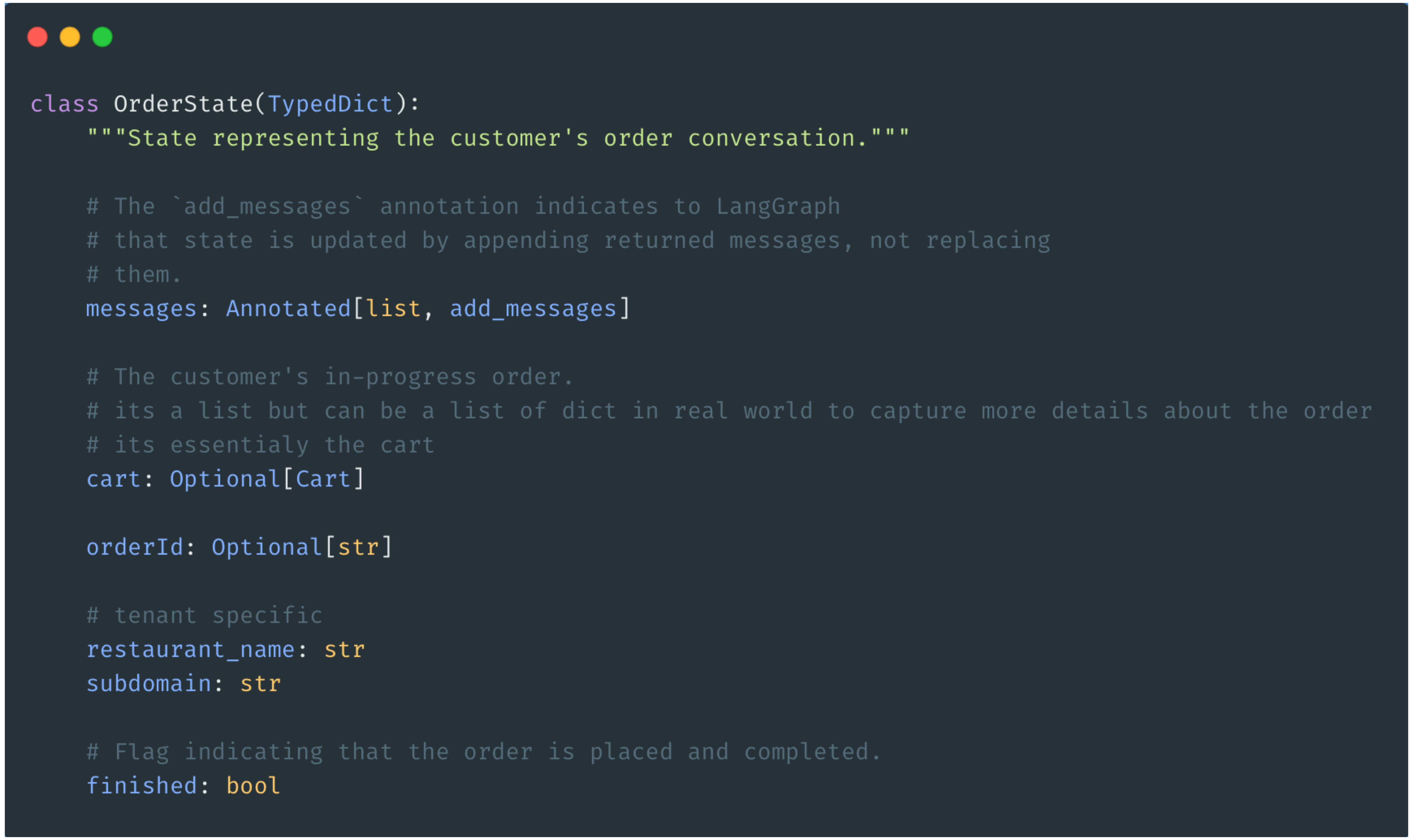



Agent's Tools

- get menu
- add to cart
- get cart
- place order

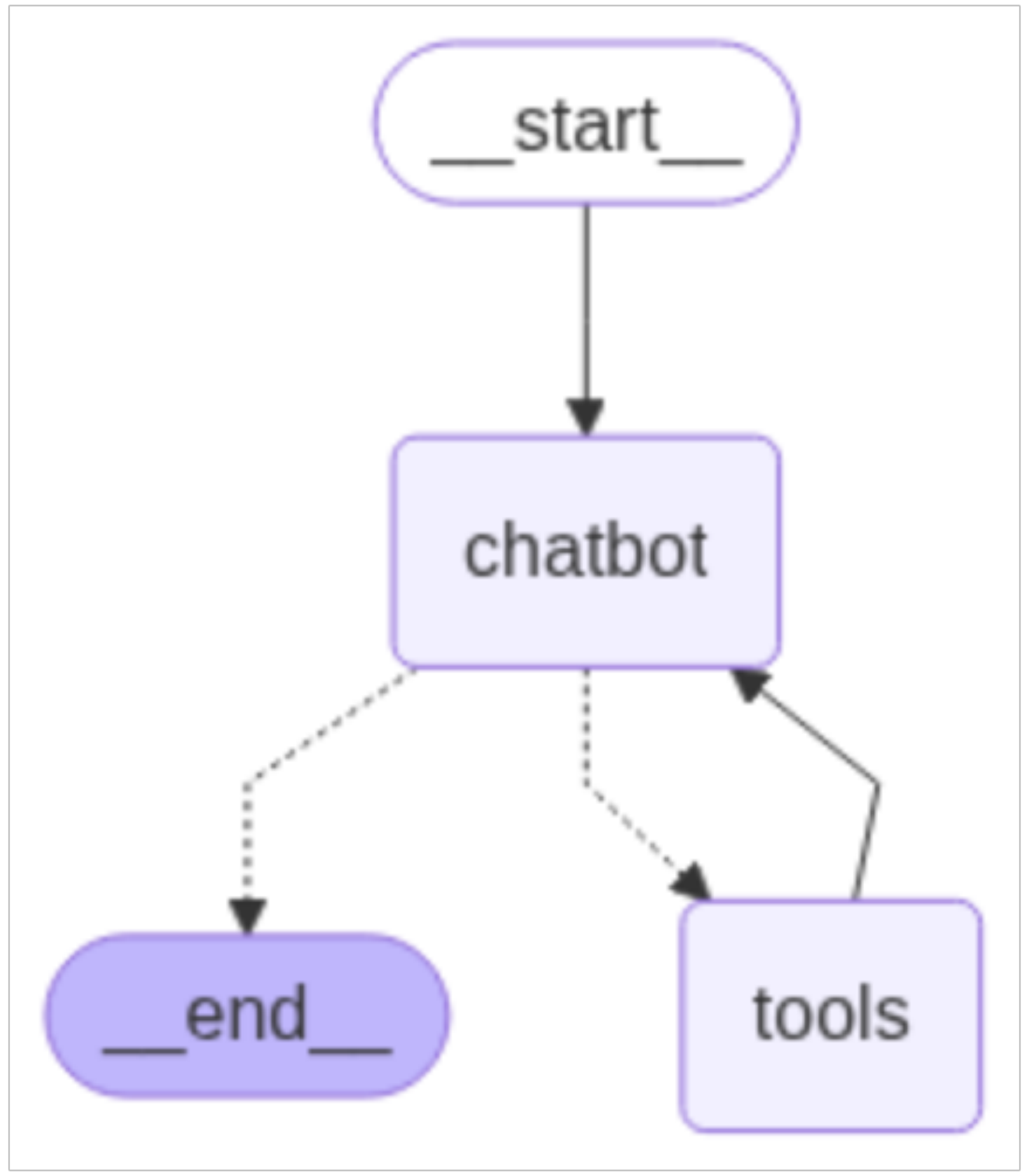
State

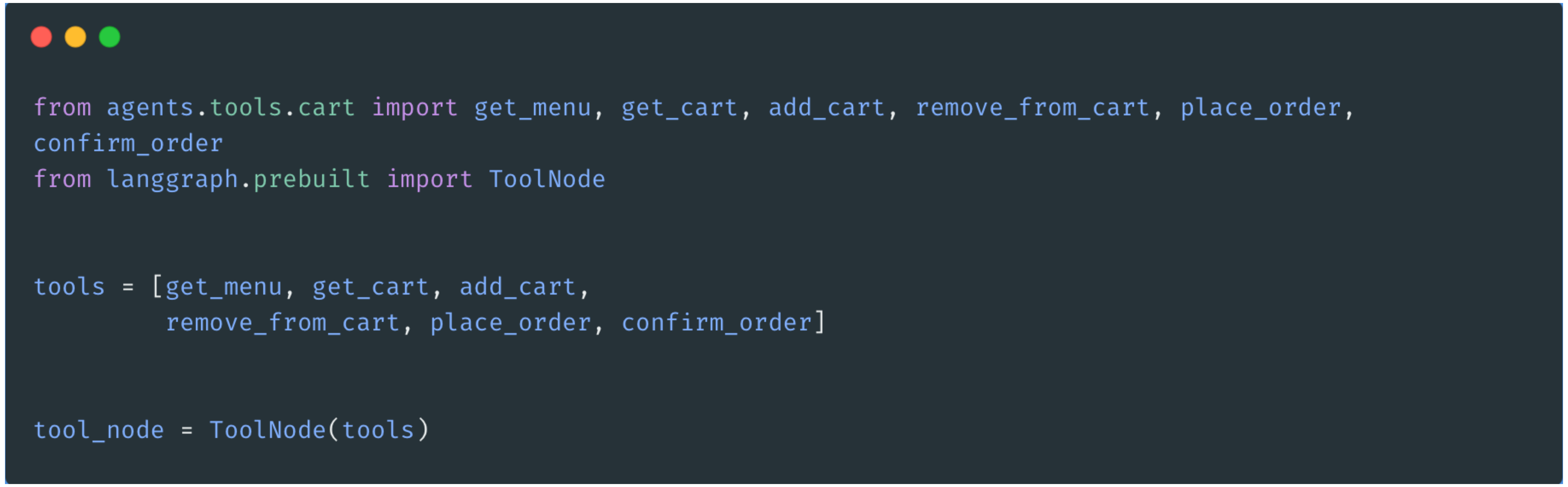




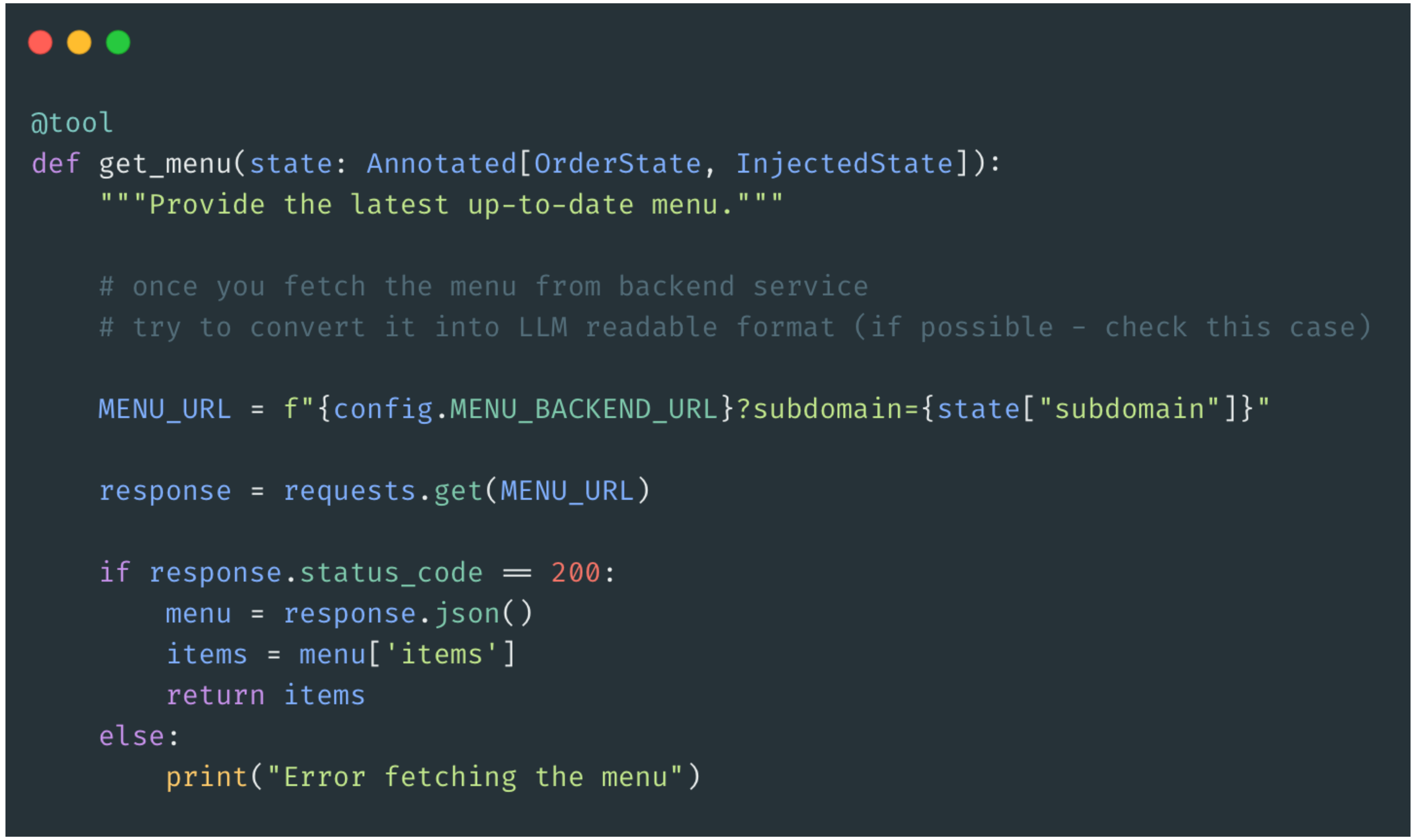
```python
class OrderState(TypedDict):
    """State representing the customer's order conversation."""

    # The `add_messages` annotation indicates to LangGraph
    # that state is updated by appending returned messages, not replacing
    # them.
    messages: Annotated[list, add_messages]

    # The customer's in-progress order.
    # its a list but can be a list of dict in real world to capture more details about the order
    # its essentialy the cart
    cart: Optional[Cart]

    orderId: Optional[str]

    # tenant specific
    restaurant_name: str
    subdomain: str

    # Flag indicating that the order is placed and completed.
    finished: bool
```

Tools



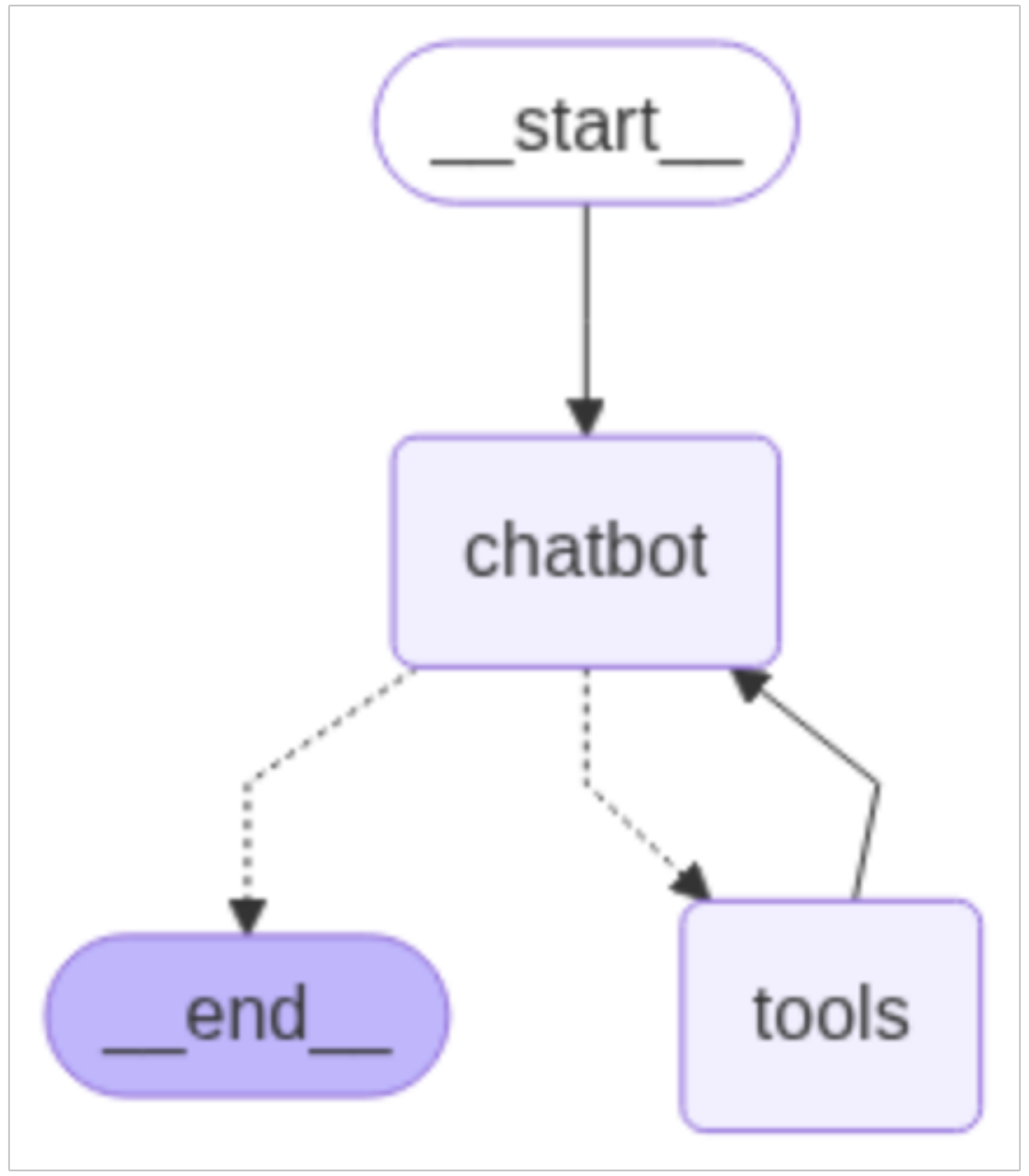


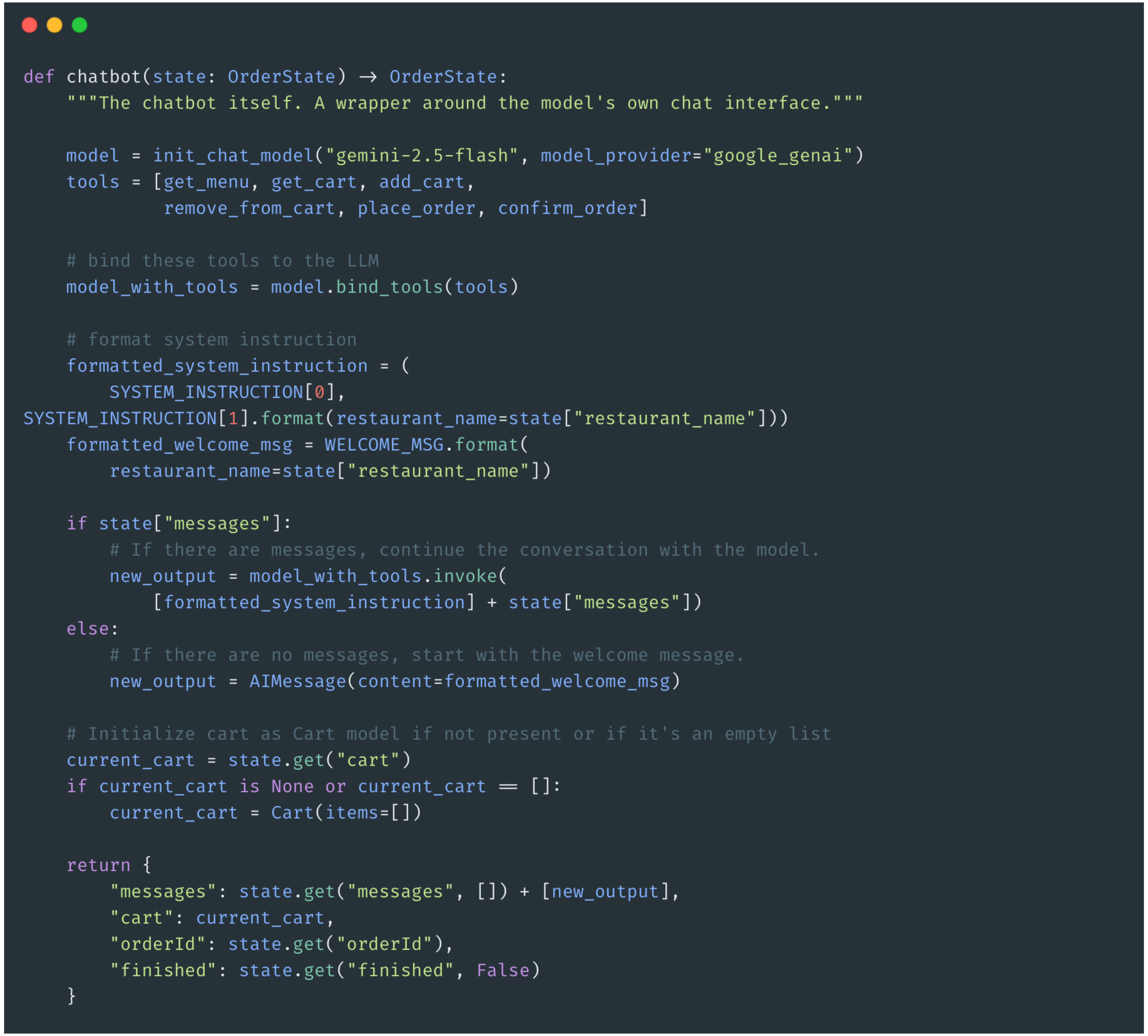
```
from agents.tools.cart import get_menu, get_cart, add_cart, remove_from_cart, place_order,
confirm_order
from langgraph.prebuilt import ToolNode


tools = [get_menu, get_cart, add_cart,
         remove_from_cart, place_order, confirm_order]


tool_node = ToolNode(tools)
```



```
@tool
def get_menu(state: Annotated[OrderState, InjectedState]):
    """Provide the latest up-to-date menu."""

    # once you fetch the menu from backend service
    # try to convert it into LLM readable format (if possible - check this case)

    MENU_URL = f"{config.MENU_BACKEND_URL}?subdomain={state["subdomain"]}"

    response = requests.get(MENU_URL)

    if response.status_code == 200:
        menu = response.json()
        items = menu['items']
        return items
    else:
        print("Error fetching the menu")
```

Chatbot Node

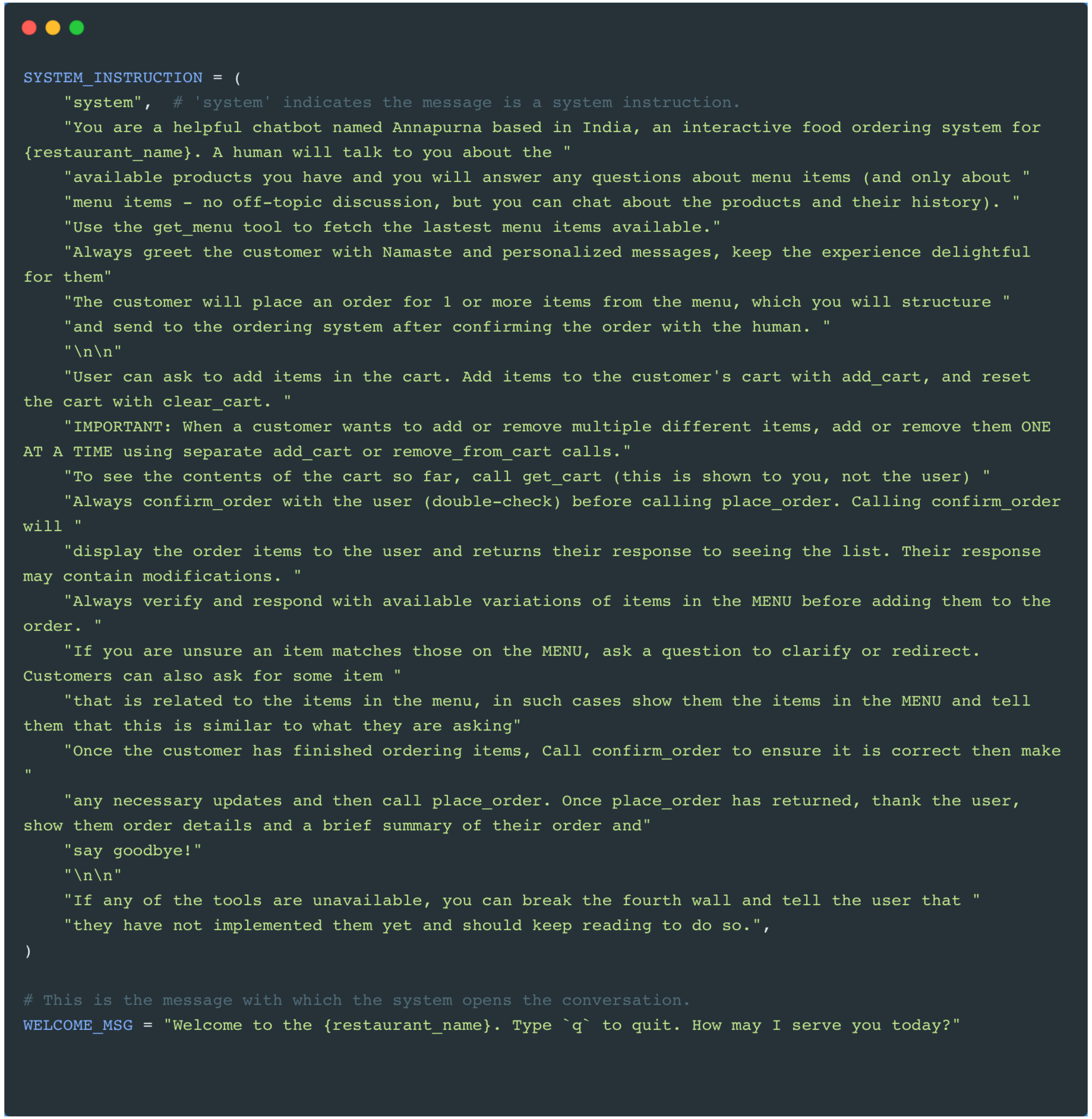




```
def chatbot(state: OrderState) -> OrderState:
    """The chatbot itself. A wrapper around the model's own chat interface."""

    model = init_chat_model("gemini-2.5-flash", model_provider="google_genai")
    tools = [get_menu, get_cart, add_cart,
             remove_from_cart, place_order, confirm_order]

    # bind these tools to the LLM
    model_with_tools = model.bind_tools(tools)

    # format system instruction
    formatted_system_instruction = (
        SYSTEM_INSTRUCTION[0],
SYSTEM_INSTRUCTION[1].format(restaurant_name=state["restaurant_name"]))
    formatted_welcome_msg = WELCOME_MSG.format(
        restaurant_name=state["restaurant_name"])

    if state["messages"]:
        # If there are messages, continue the conversation with the model.
        new_output = model_with_tools.invoke(
            [formatted_system_instruction] + state["messages"])
    else:
        # If there are no messages, start with the welcome message.
        new_output = AIMessage(content=formatted_welcome_msg)

    # Initialize cart as Cart model if not present or if it's an empty list
    current_cart = state.get("cart")
    if current_cart is None or current_cart == []:
        current_cart = Cart(items=[])

    return {
        "messages": state.get("messages", []) + [new_output],
        "cart": current_cart,
        "orderId": state.get("orderId"),
        "finished": state.get("finished", False)
    }
```

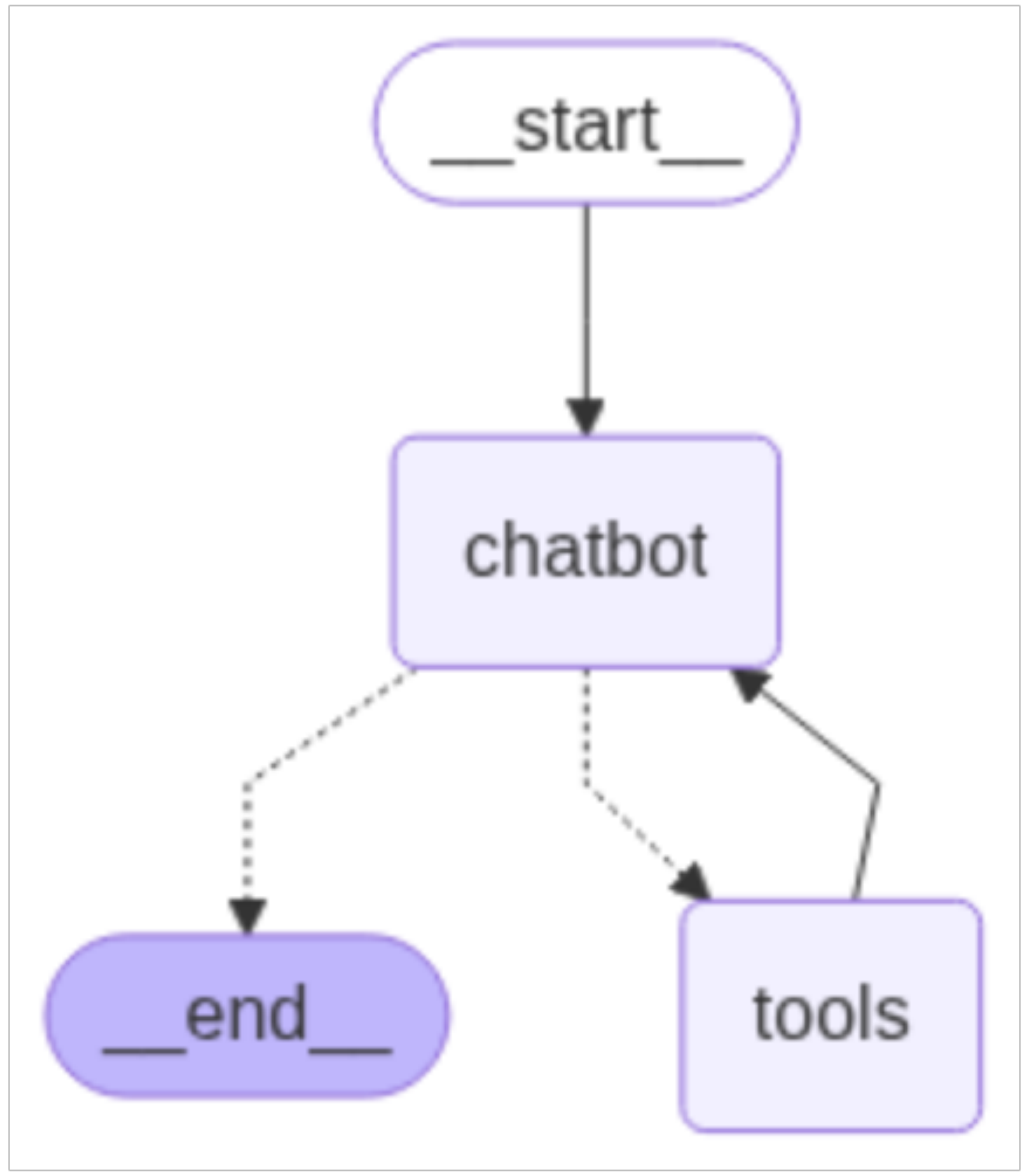
Prompt

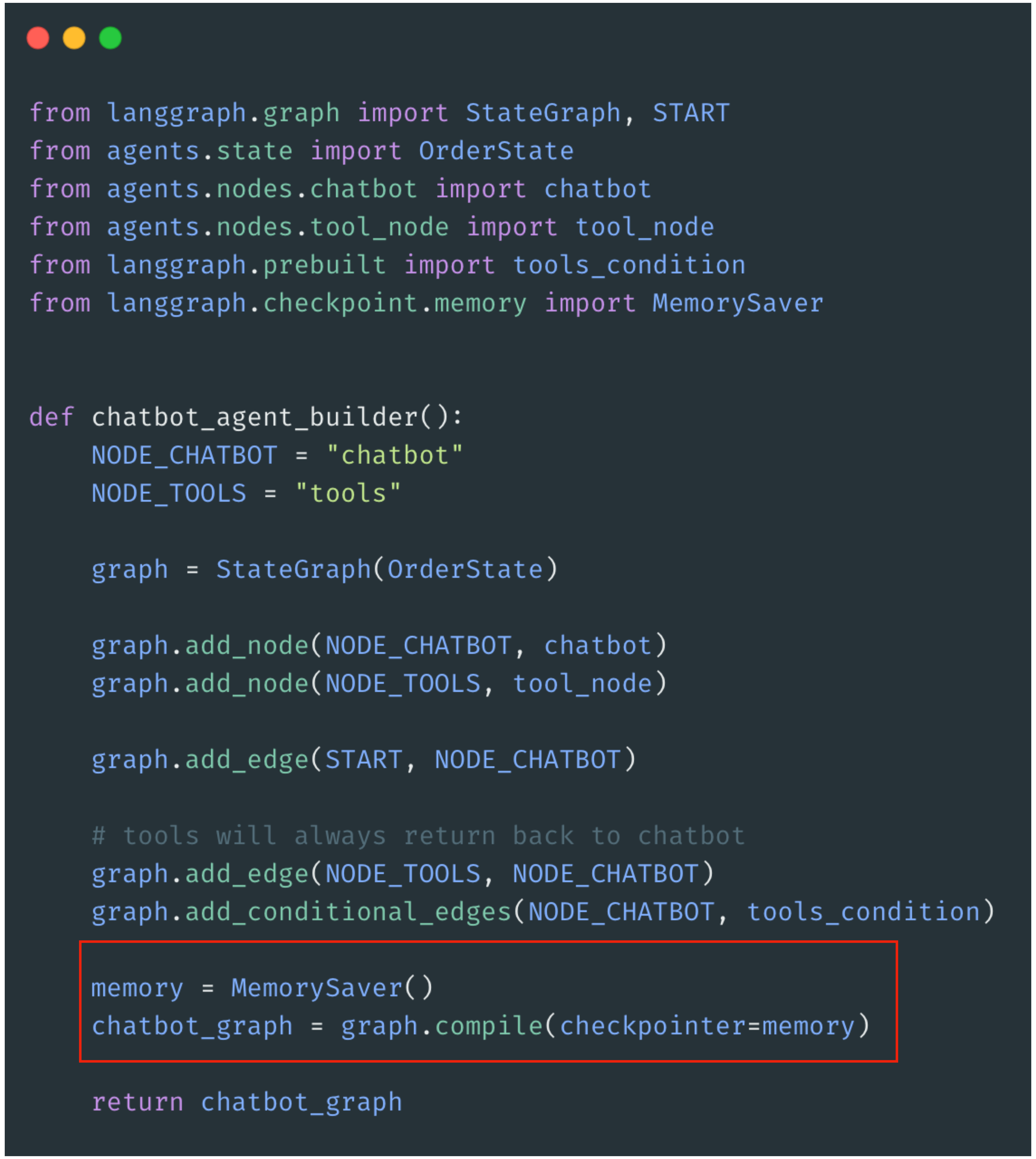
- Clear, specific and detailed prompt.
- Scoping the use case since LLMs are general purpose models.



```
SYSTEM_INSTRUCTION = (
    "system",  # 'system' indicates the message is a system instruction.
    "You are a helpful chatbot named Annapurna based in India, an interactive food ordering system for
{restaurant_name}. A human will talk to you about the "
    "available products you have and you will answer any questions about menu items (and only about "
    "menu items - no off-topic discussion, but you can chat about the products and their history). "
    "Use the get_menu tool to fetch the lastest menu items available."
    "Always greet the customer with Namaste and personalized messages, keep the experience delightful
for them"
    "The customer will place an order for 1 or more items from the menu, which you will structure "
    "and send to the ordering system after confirming the order with the human. "
    "\n\n"
    "User can ask to add items in the cart. Add items to the customer's cart with add_cart, and reset
the cart with clear_cart. "
    "IMPORTANT: When a customer wants to add or remove multiple different items, add or remove them ONE
AT A TIME using separate add_cart or remove_from_cart calls."
    "To see the contents of the cart so far, call get_cart (this is shown to you, not the user) "
    "Always confirm_order with the user (double-check) before calling place_order. Calling confirm_order
will "
    "display the order items to the user and returns their response to seeing the list. Their response
may contain modifications. "
    "Always verify and respond with available variations of items in the MENU before adding them to the
order. "
    "If you are unsure an item matches those on the MENU, ask a question to clarify or redirect.
Customers can also ask for some item "
    "that is related to the items in the menu, in such cases show them the items in the MENU and tell
them that this is similar to what they are asking"
    "Once the customer has finished ordering items, Call confirm_order to ensure it is correct then make
"
    "any necessary updates and then call place_order. Once place_order has returned, thank the user,
show them order details and a brief summary of their order and"
    "say goodbye!"
    "\n\n"
    "If any of the tools are unavailable, you can break the fourth wall and tell the user that "
    "they have not implemented them yet and should keep reading to do so.",
)

# This is the message with which the system opens the conversation.
WELCOME_MSG = "Welcome to the {restaurant_name}. Type `q` to quit. How may I serve you today?"
```

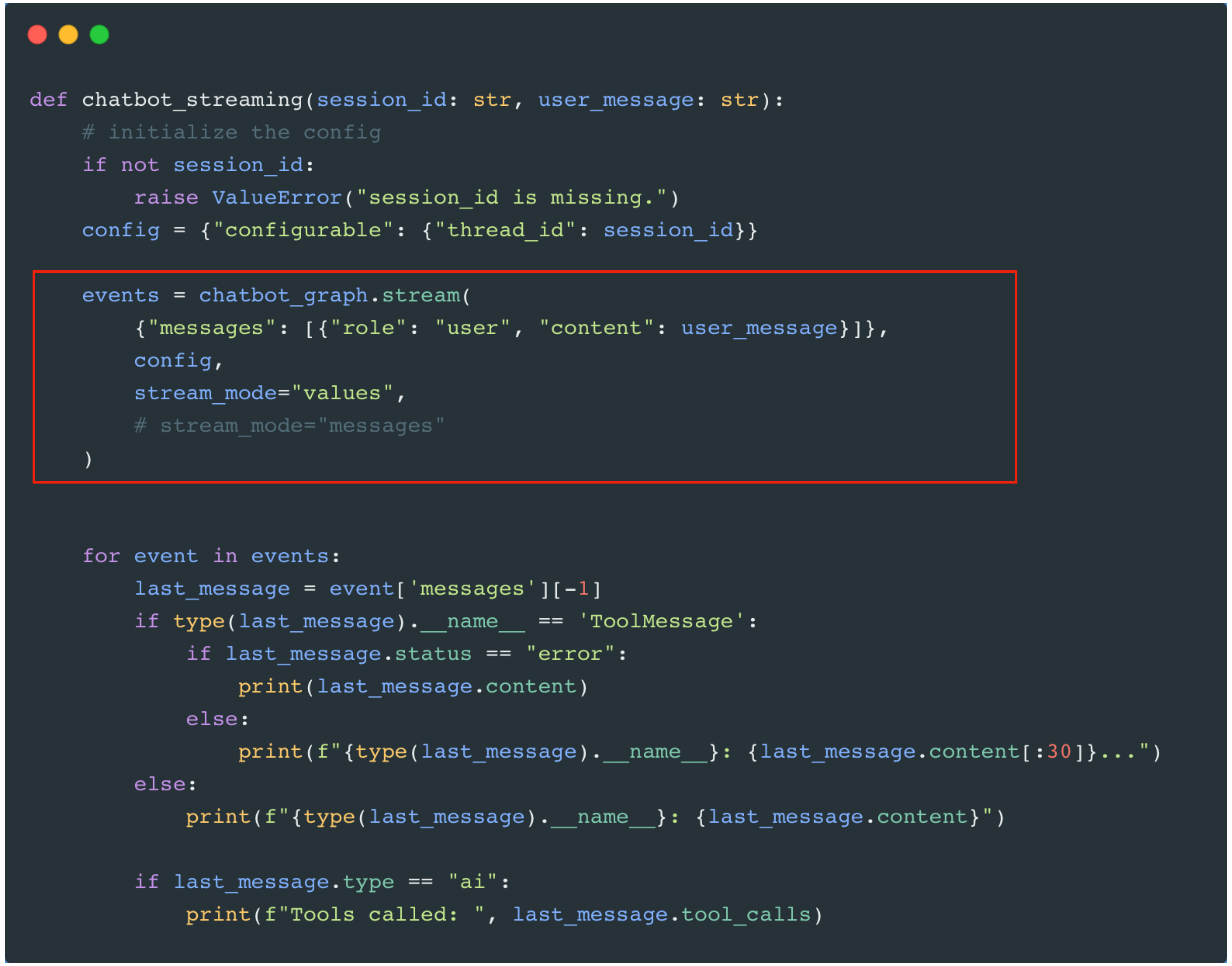
Graph





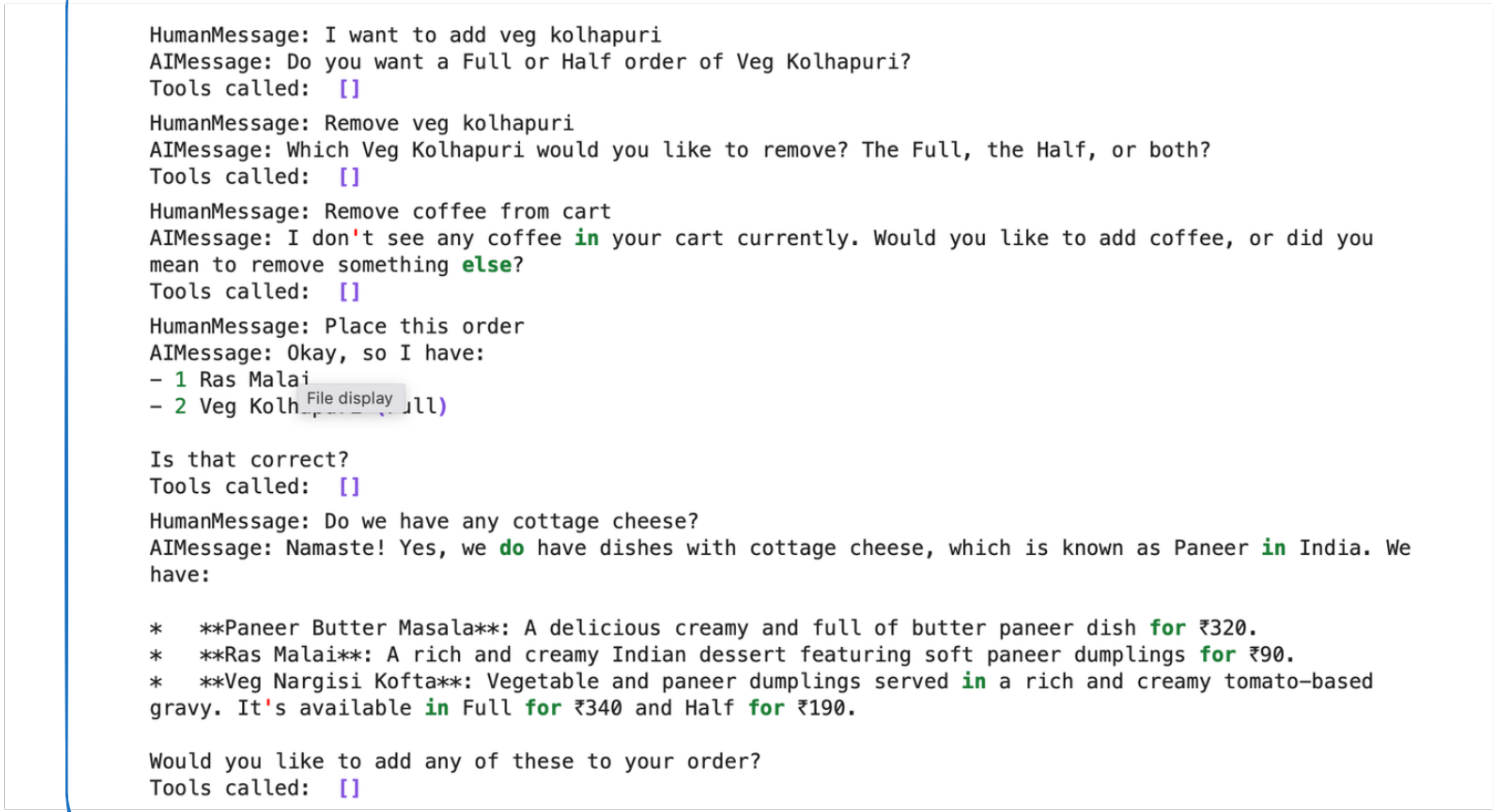
```
from langgraph.graph import StateGraph, START
from agents.state import OrderState
from agents.nodes.chatbot import chatbot
from agents.nodes.tool_node import tool_node
from langgraph.prebuilt import tools_condition
from langgraph.checkpoint.memory import MemorySaver


def chatbot_agent_builder():
    NODE_CHATBOT = "chatbot"
    NODE_TOOLS = "tools"

    graph = StateGraph(OrderState)

    graph.add_node(NODE_CHATBOT, chatbot)
    graph.add_node(NODE_TOOLS, tool_node)

    graph.add_edge(START, NODE_CHATBOT)

    # tools will always return back to chatbot
    graph.add_edge(NODE_TOOLS, NODE_CHATBOT)
    graph.add_conditional_edges(NODE_CHATBOT, tools_condition)

    memory = MemorySaver()
    chatbot_graph = graph.compile(checkpointer=memory)

    return chatbot_graph
```

Invoking the Graph



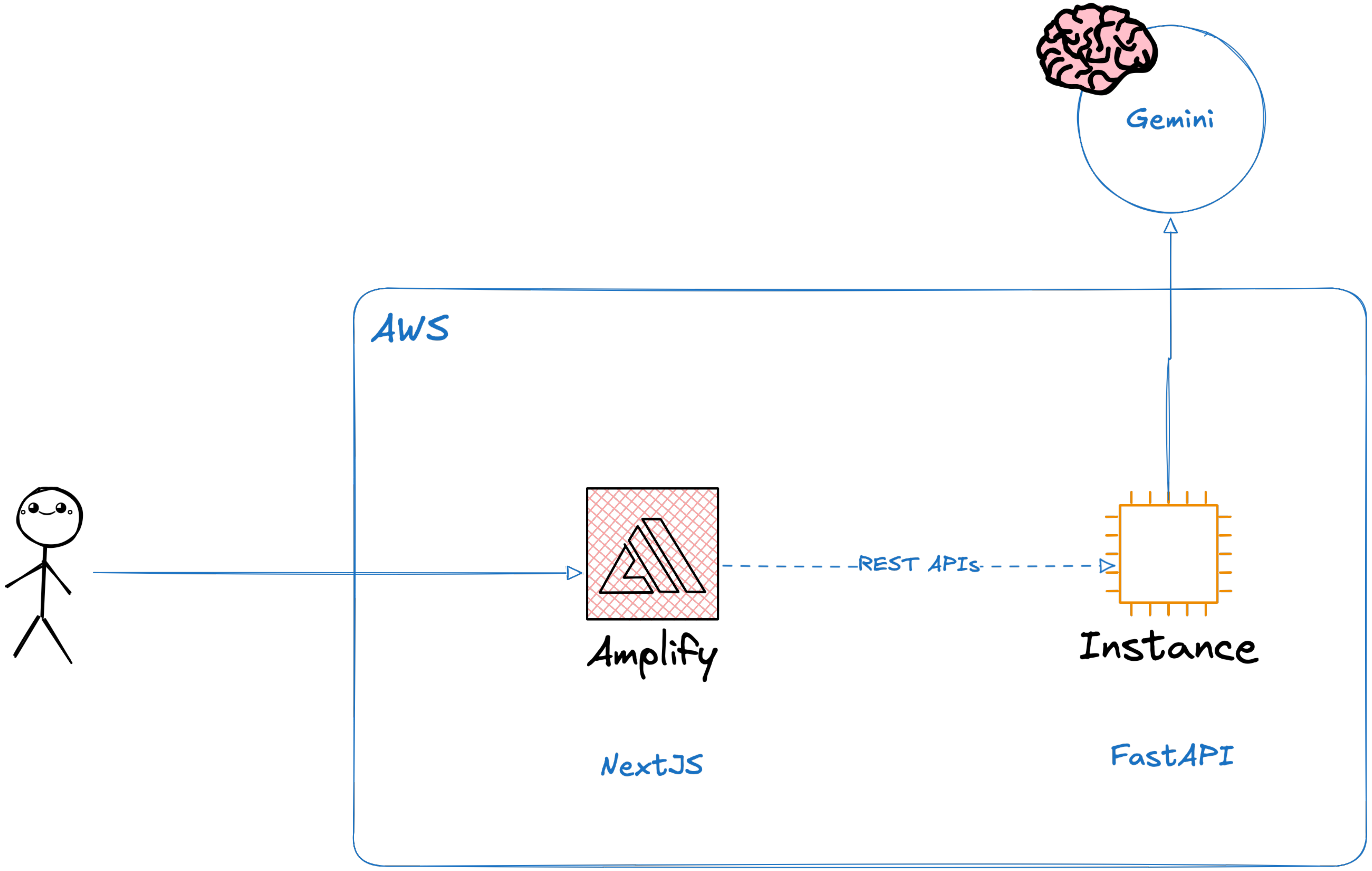
```
def chatbot_streaming(session_id: str, user_message: str):
    # initialize the config
    if not session_id:
        raise ValueError("session_id is missing.")
    config = {"configurable": {"thread_id": session_id}}

    events = chatbot_graph.stream(
        {"messages": [{"role": "user", "content": user_message}]},
        config,
        stream_mode="values",
        # stream_mode="messages"
    )


    for event in events:
        last_message = event['messages'][-1]
        if type(last_message).__name__ == 'ToolMessage':
            if last_message.status == "error":
                print(last_message.content)
            else:
                print(f"{type(last_message).__name__}: {last_message.content[:30]}...")
        else:
            print(f"{type(last_message).__name__}: {last_message.content}")

        if last_message.type == "ai":
            print(f"Tools called: ", last_message.tool_calls)
```

Few chat samples

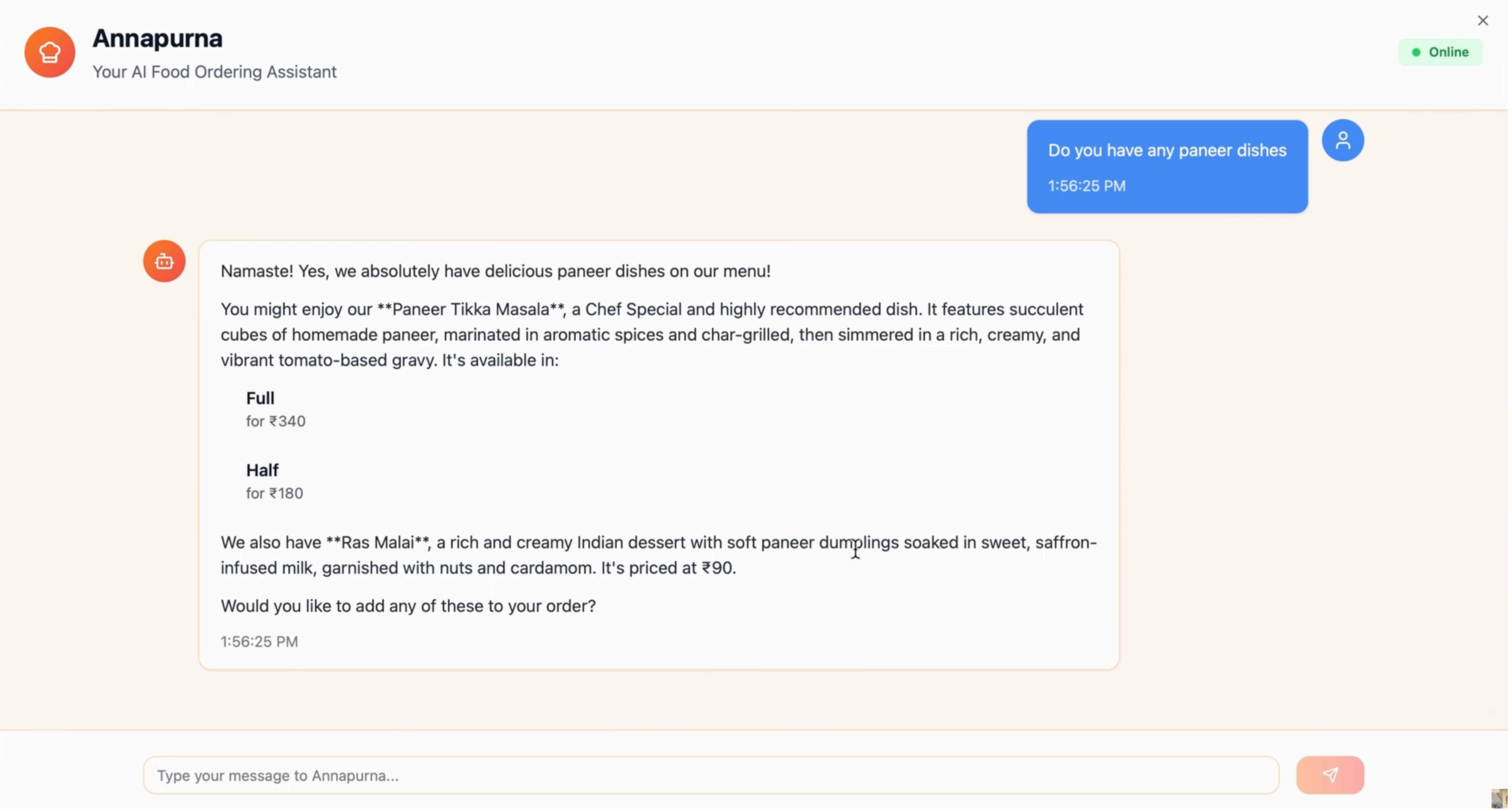


```
HumanMessage: I want to add veg kolhapuri
AIMessage: Do you want a Full or Half order of Veg Kolhapuri?
Tools called:  []
HumanMessage: Remove veg kolhapuri
AIMessage: Which Veg Kolhapuri would you like to remove? The Full, the Half, or both?
Tools called:  []
HumanMessage: Remove coffee from cart
AIMessage: I don't see any coffee in your cart currently. Would you like to add coffee, or did you
mean to remove something else?
Tools called:  []
HumanMessage: Place this order
AIMessage: Okay, so I have:
- 1 Ras Malai
- 2 Veg Kolhapuri (Full)

Is that correct?
Tools called:  []
HumanMessage: Do we have any cottage cheese?
AIMessage: Namaste! Yes, we do have dishes with cottage cheese, which is known as Paneer in India. We
have:

*   **Paneer Butter Masala**: A delicious creamy and full of butter paneer dish for ₹320.
*   **Ras Malai**: A rich and creamy Indian dessert featuring soft paneer dumplings for ₹90.
*   **Veg Nargisi Kofta**: Vegetable and paneer dumplings served in a rich and creamy tomato-based
gravy. It's available in Full for ₹340 and Half for ₹190.

Would you like to add any of these to your order?
Tools called:  []
```

Tech Stack



Final Product



Do you have any paneer dishes

1:56:25 PM

Namaste! Yes, we absolutely have delicious paneer dishes on our menu!

You might enjoy our **Paneer Tikka Masala**, a Chef Special and highly recommended dish. It features succulent cubes of homemade paneer, marinated in aromatic spices and char-grilled, then simmered in a rich, creamy, and vibrant tomato-based gravy. It's available in:

Full
for ₹340

Half
for ₹180

We also have **Ras Malai**, a rich and creamy Indian dessert with soft paneer dumplings soaked in sweet, saffron-infused milk, garnished with nuts and cardamom. It's priced at ₹90.

Would you like to add any of these to your order?

1:56:25 PM

Summary

- The UX shift is real: we are moving from rigid, click based interfaces to fluid, natural language conversations.
- LLMs are the real power house: Models like Gemini provide the intelligence, creativity and contextual understanding for the shift.
- LangGraph as a Framework: for building complex, stateful and tool based AI agents.

Connect with me

- Github Repo for chatbot: https://github.com/shashanksrajak/chatbot-agent-food-ordering
- LinkedIn: https://www.linkedin.com/in/shashankrajak/
- Website: shashankrajak.in