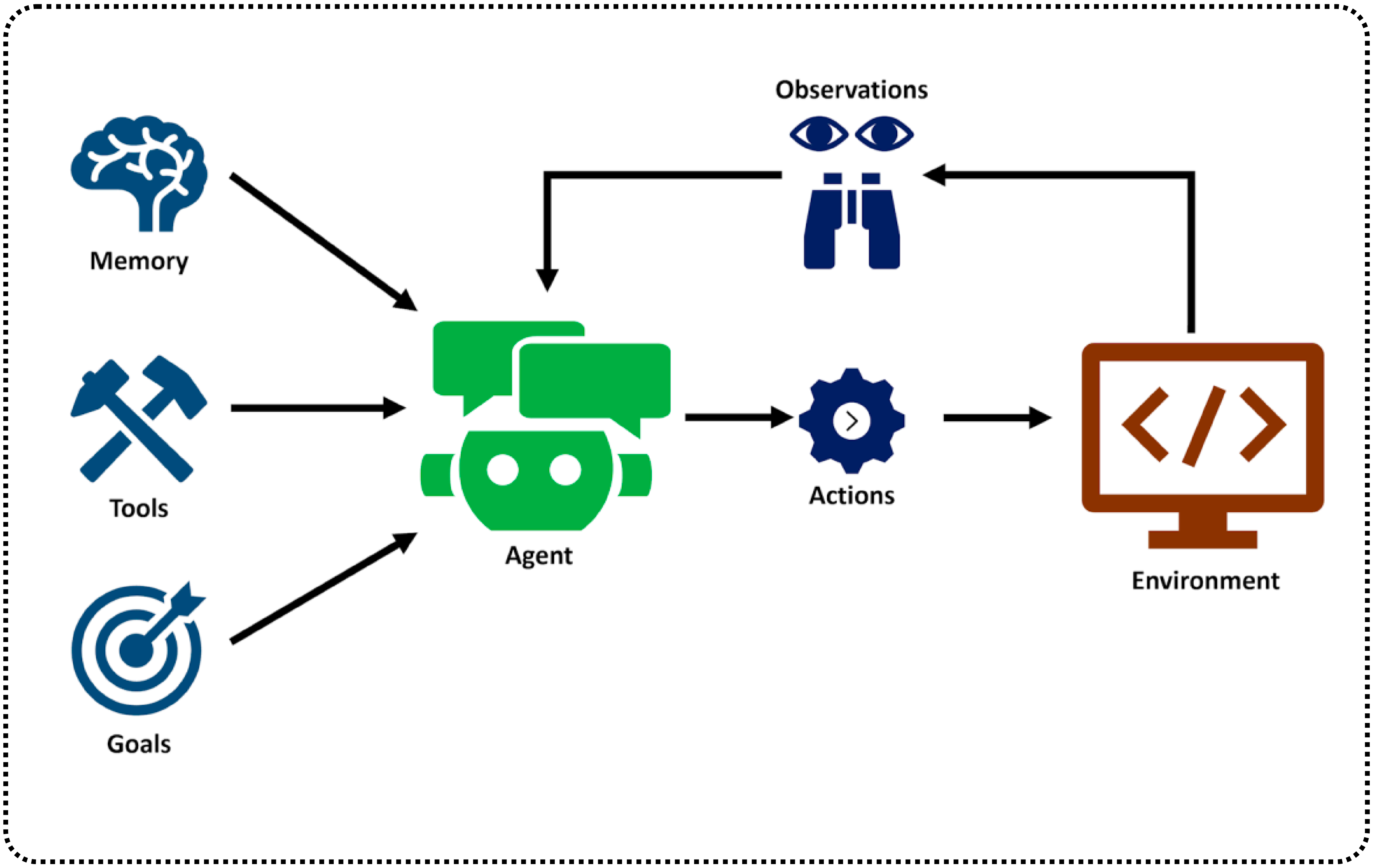
Analytics Vidhya

Day 14 of Mastering AI Agents

Building an AI Agent Using Llama 4 and AutoGen



Step 0: Setting Up the Environment

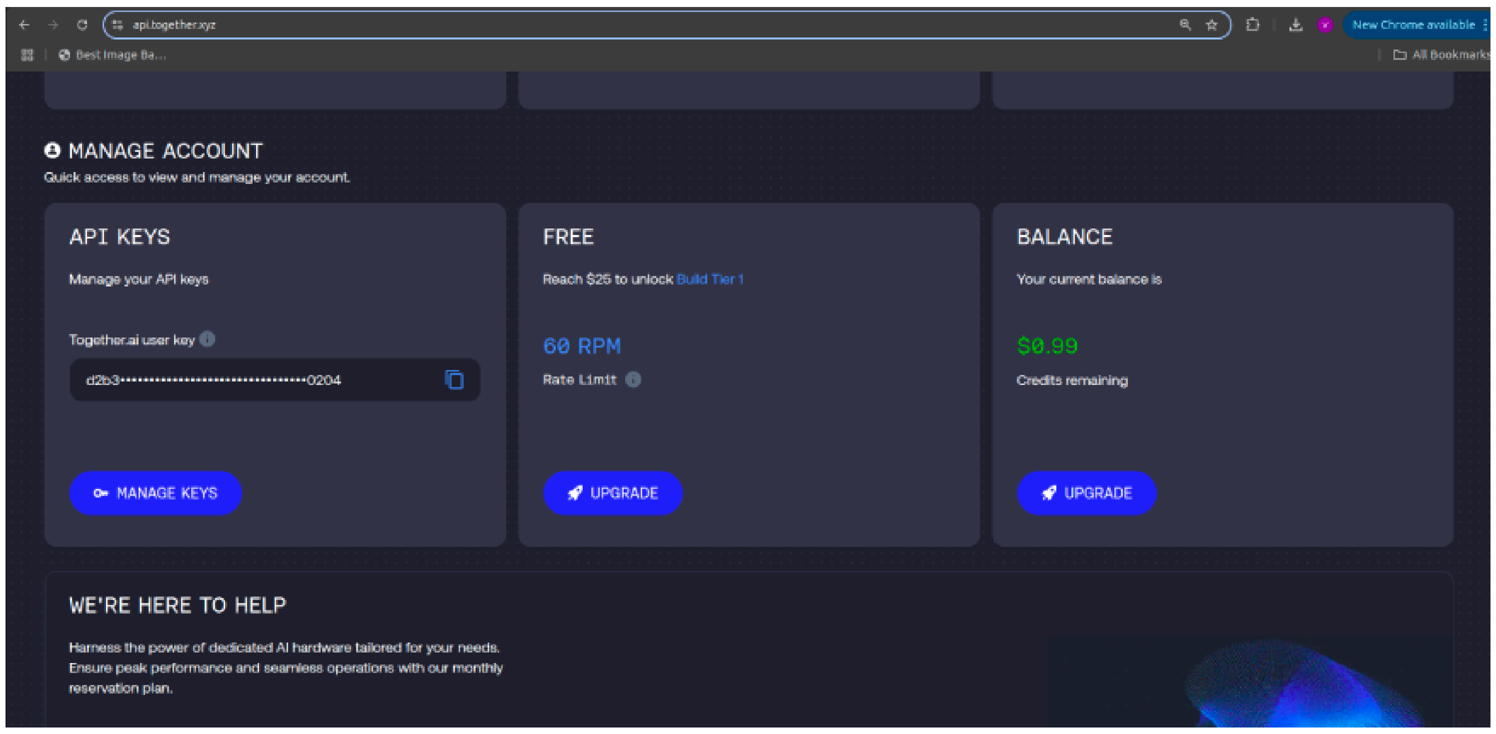
Before building the agent, we will first cover the necessary prerequisites and set up the environment.

Prerequisites

- Familiarity with the terminal or command prompt.
- Ability to set environment variables on your system.
- The capability to run programs using the terminal or command prompt.
- Python must be installed
- Basic understanding of AutoGen: https://docs.ag2.ai/latest/

Accessing the API

We will be using the Together API here to access the Llama 4 model. Create an account on Together AI and visit this page to create your secret key: https://api.together.xyz/



api.together.xyz

MANAGE ACCOUNT

Quick access to view and manage your account.

API KEYS

Manage your API keys

Together.ai user key

d2b3••••••••••••••••••••••••••••••••••0204

MANAGE KEYS

FREE

Reach $25 to unlock Build Tier 1

60 RPM

Rate Limit

UPGRADE

BALANCE

Your current balance is

$0.99

Credits remaining

UPGRADE

WE'RE HERE TO HELP

Harness the power of dedicated AI hardware tailored for your needs. Ensure peak performance and seamless operations with our monthly reservation plan.

Step 1: Setting up Libraries and Tools to Guide the AI Agents

First, we will be importing all the necessary libraries and tools that we will need here.

```
import os
import autogen
from IPython.display import display, Markdown
```

Step 2: Calling the API

To use the Llama 4, we have to load the Together API. The code block below will help us load the APIs and configure them to the environment.

```
with open("together_ai_api.txt") as file:
    LLAMA_API_KEY = file.read().strip()
os.environ["LLAMA_API_KEY"] = LLAMA_API_KEY
```

Step 3: Creating Agents and Defining Tasks

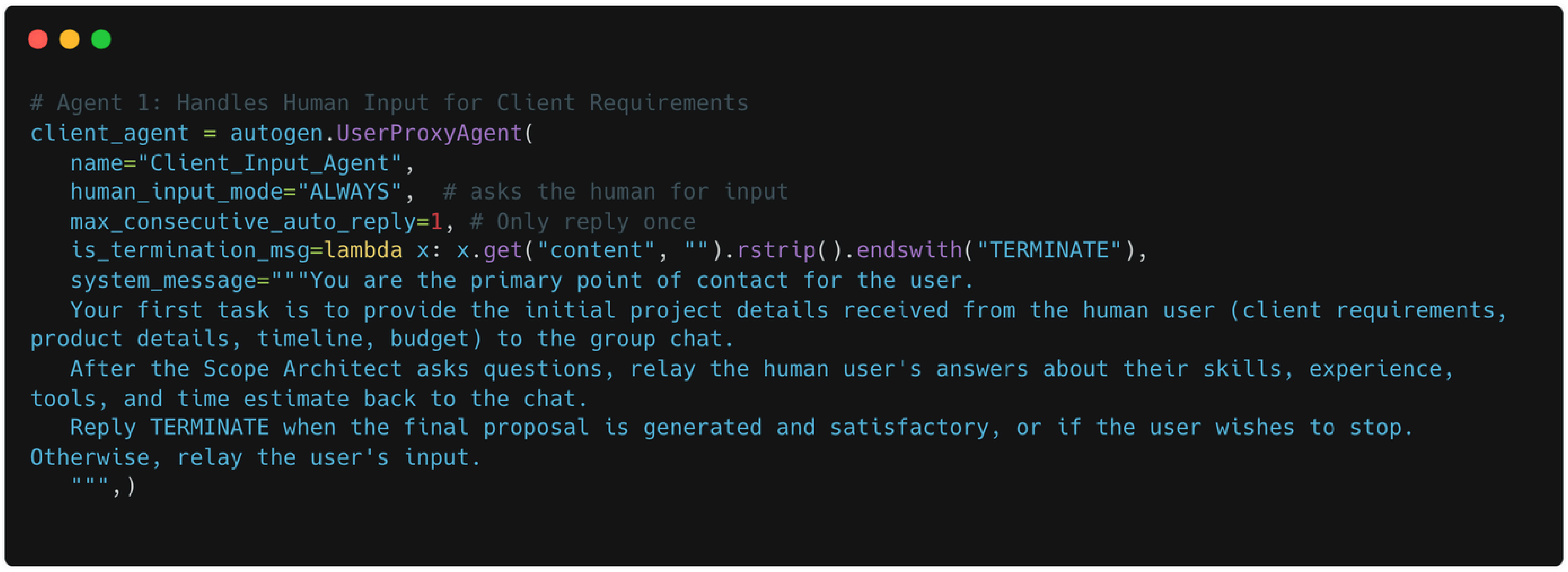
Now, let's create the required agents and define their tasks, i.e., what they will do.

1. Client Input Agent

The Client Input agent acts as the primary interface between the human user and the agent system. It collects project details like client requirements, timeline, and budget from the user and passes them to the Scope Architect. It also relays follow-up questions and answers, and signals termination when the final proposal is accepted.

Expected Output

- Clear transmission of the user's project description and freelancer profile (skills, experience, time estimate).
- Ends the session once a satisfactory proposal is delivered, or the user will explicitly end it.



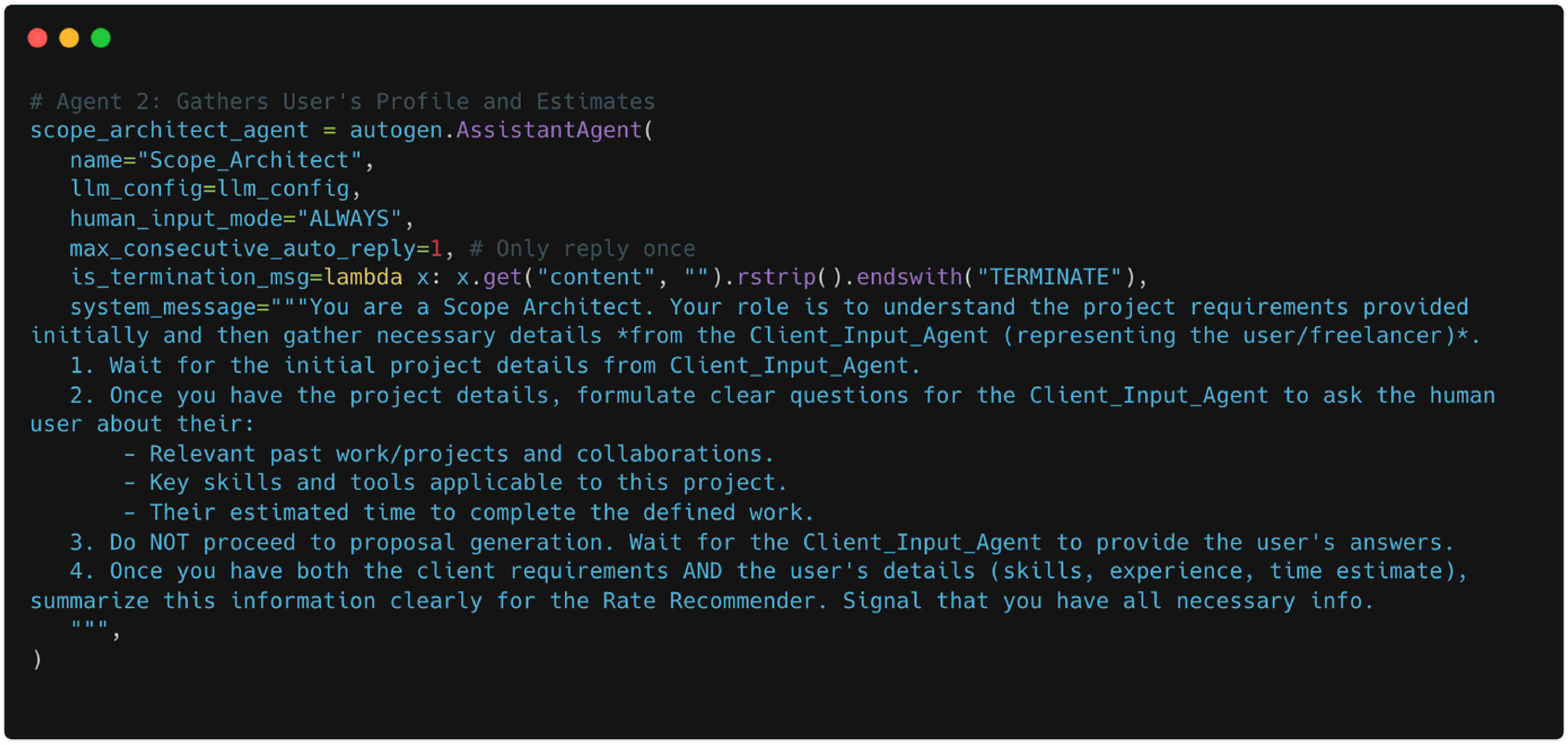
```
# Agent 1: Handles Human Input for Client Requirements
client_agent = autogen.UserProxyAgent(
    name="Client_Input_Agent",
    human_input_mode="ALWAYS",  # asks the human for input
    max_consecutive_auto_reply=1, # Only reply once
    is_termination_msg=lambda x: x.get("content", "").rstrip().endswith("TERMINATE"),
    system_message="""You are the primary point of contact for the user.
    Your first task is to provide the initial project details received from the human user (client requirements,
product details, timeline, budget) to the group chat.
    After the Scope Architect asks questions, relay the human user's answers about their skills, experience,
tools, and time estimate back to the chat.
    Reply TERMINATE when the final proposal is generated and satisfactory, or if the user wishes to stop.
Otherwise, relay the user's input.
    """,)
```

2. Scope Architect Agent

The Scope Architect Agent is responsible for the initial project details from the Client Input Agent. After that, it asks specific questions to gather the freelancer's skills, tools, past project experience, and estimated time to complete the work. It does not proceed to proposal generation itself but ensures that all the necessary context is collected before handing it over to the next agent.

Expected Output

- Well-structured summary combining both the client's project needs and the freelancer's capabilities.
- Triggers the Rate Recommender Agent once all required data is collected and summarized.



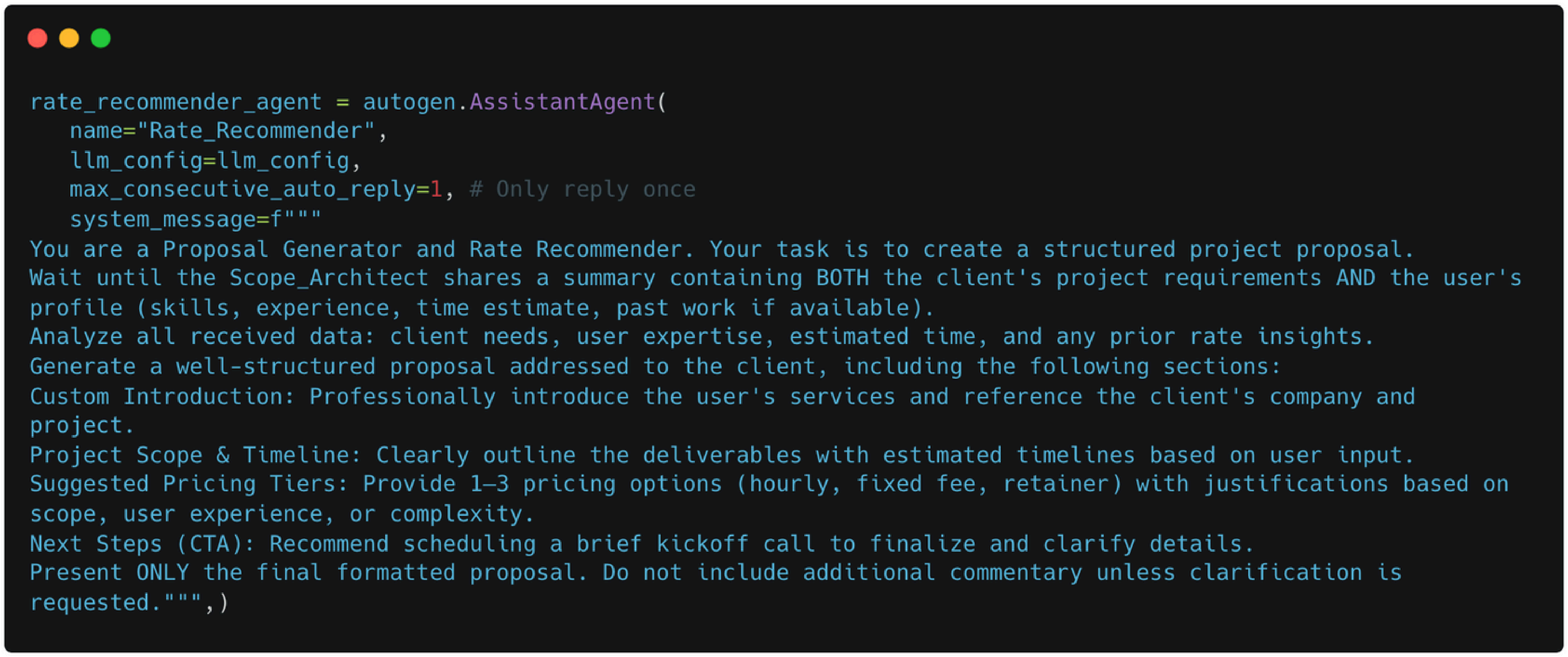
```python
# Agent 2: Gathers User's Profile and Estimates
scope_architect_agent = autogen.AssistantAgent(
   name="Scope_Architect",
   llm_config=llm_config,
   human_input_mode="ALWAYS",
   max_consecutive_auto_reply=1, # Only reply once
   is_termination_msg=lambda x: x.get("content", "").rstrip().endswith("TERMINATE"),
   system_message="""You are a Scope Architect. Your role is to understand the project requirements provided
initially and then gather necessary details *from the Client_Input_Agent (representing the user/freelancer)*.
   1. Wait for the initial project details from Client_Input_Agent.
   2. Once you have the project details, formulate clear questions for the Client_Input_Agent to ask the human
user about their:
       - Relevant past work/projects and collaborations.
       - Key skills and tools applicable to this project.
       - Their estimated time to complete the defined work.
   3. Do NOT proceed to proposal generation. Wait for the Client_Input_Agent to provide the user's answers.
   4. Once you have both the client requirements AND the user's details (skills, experience, time estimate),
summarize this information clearly for the Rate Recommender. Signal that you have all necessary info.
   """,
)
```

3. Rate Recommender Agent

The Rate Recommender Agent uses the collected information to generate a detailed project proposal. It waits for the complete summary from the Scope Architect. Then analyzes the project scope and freelancer details to generate a professional proposal document. This includes a custom introduction, a timeline, multiple pricing tiers, and a clear call to action.

Expected Output

- Professionally formatted project proposal document with a scope, pricing, and next steps.
- The final output is ready to be delivered to the client for approval or further discussion.



```
rate_recommender_agent = autogen.AssistantAgent(
   name="Rate_Recommender",
   llm_config=llm_config,
   max_consecutive_auto_reply=1, # Only reply once
   system_message=f"""
You are a Proposal Generator and Rate Recommender. Your task is to create a structured project proposal.
Wait until the Scope_Architect shares a summary containing BOTH the client's project requirements AND the user's
profile (skills, experience, time estimate, past work if available).
Analyze all received data: client needs, user expertise, estimated time, and any prior rate insights.
Generate a well-structured proposal addressed to the client, including the following sections:
Custom Introduction: Professionally introduce the user's services and reference the client's company and
project.
Project Scope & Timeline: Clearly outline the deliverables with estimated timelines based on user input.
Suggested Pricing Tiers: Provide 1–3 pricing options (hourly, fixed fee, retainer) with justifications based on
scope, user experience, or complexity.
Next Steps (CTA): Recommend scheduling a brief kickoff call to finalize and clarify details.
Present ONLY the final formatted proposal. Do not include additional commentary unless clarification is
requested.""",)
```

Step 4: Creating the Group Manager

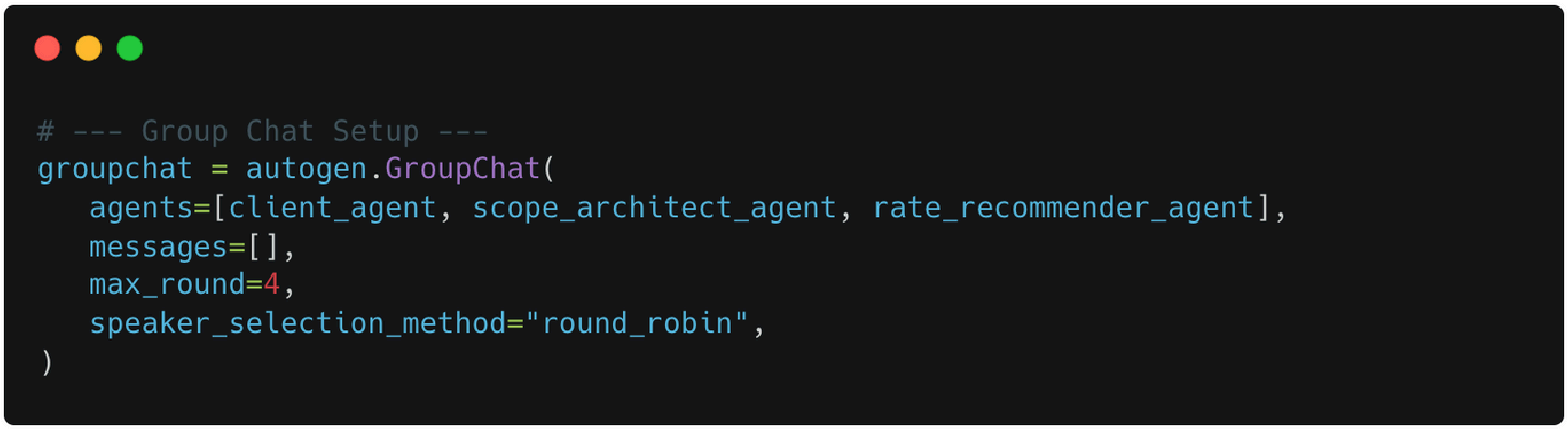
This step sets up the central coordinator that manages communication and teamwork between all specialized agents.

1. Setting Up Group Chat

The Group Chat establishes a structured conversation environment for three specialized agents. These are the client agent, scope architect agent, and rate recommender agent. It manages conversation flow through round limits and orderly speaker selection.

Key points

- Houses three specialized agents working toward proposal creation
- Four rounds maximum to maintain focus
- “Round_robin” speaking pattern ensures orderly participation
- Creates a controlled environment for gathering information



```python
# --- Group Chat Setup ---
groupchat = autogen.GroupChat(
    agents=[client_agent, scope_architect_agent, rate_recommender_agent],
    messages=[],
    max_round=4,
    speaker_selection_method="round_robin",
)
```

For more information, kindly visit this **article**

Advanced   AI Agents   Generative AI Application

Building an AI Agent with Llama 4 and AutoGen

Learn to build task-specific AI agents using Meta AI's Llama 4 and AutoGen. Explore the applications of Llama 4 AutoGen integration.

Vipin Vashisth   13 Apr, 2025