FTEC5660 Agentic AI for Business and FinTech

Homework 02 (Part 2) — Report

---

1. Agent Design and Architecture

For this homework I built a simple tool-augmented LLM agent that can interact with Moltbook through its REST API. The whole thing runs in Google Colab and uses three main components.

The first component is the LLM itself. I used Google Gemini 2.5 Flash through the langchain-google-genai package. I set temperature to 0 so the agent behaves deterministically and doesn't generate random or unexpected actions. The model is initialized with an API key stored in Colab Secrets.

The second component is the tool layer. The starter code provided five basic tools (get_feed, search_moltbook, create_post, comment_post, upvote_post). After reading the homework requirements I realized these were not enough to finish all three tasks. So I added three more tools by looking at what the Moltbook API supports: get_submolt for fetching info about a specific submolt, subscribe_submolt for subscribing to a submolt, and verify_content for solving the math captcha that Moltbook requires when you post a comment. Each tool is a Python function decorated with @tool from LangChain. Inside each function I just make an HTTP request to the corresponding Moltbook API endpoint with the Bearer token in the header.

The third component is the agent loop. It is a simple turn-based loop. Each turn, the conversation history is sent to Gemini, and the model either returns a text response (meaning it is done) or requests one or more tool calls. If there are tool calls, the loop executes them, appends the results back to the history, and goes to the next turn. The loop stops when the model gives a final text answer or when max_turns (set to 8) is reached.

Authentication is handled by storing the Moltbook API key in Colab Secrets and injecting it into the Authorization header of every HTTP request. The agent was registered with curl and then claimed through the web interface.

2. Decision Logic and Autonomy Level

The agent works in a human-instruction mode. I give it a natural language instruction like "Subscribe to the submolt named ftec5660" and the LLM figures out which tool to call and with what arguments.

The decision-making process is basically handled by Gemini through LangChain's bind_tools() method. When I bind the 8 tools to the model, LangChain sends the tool schemas (name, description, parameters) to Gemini. Then when Gemini receives an instruction, it matches the intent to the right tool and generates the appropriate function call.

For multi-step tasks, the model plans sequentially. For example when I asked it to upvote, comment, and then verify, it did them one at a time across multiple turns: Turn 1 called upvote_post, Turn 2 called comment_post, and Turn 3 parsed the verification challenge from the comment response, solved the math (35 + 22 = 57), and called verify_content with "57.00".

The verification part is interesting because Moltbook returns a math puzzle in natural language and the agent needs to understand the text, extract the numbers, do the calculation, and format the answer with 2 decimal places. I put a rule in the system prompt (Rule 7) telling the agent to handle this automatically.

I would say the autonomy level is semi-autonomous. The agent cannot decide on its own what to do — it needs my instruction to start. But once given an instruction, it can plan and execute multiple steps without further human input. It also handles unexpected situations like the verification challenge without me having to intervene.

The system prompt also includes some safety rules like not spamming, not repeating content, and searching before posting. They show the agent could work more independently if needed.

3. Moltbook Interaction Logs

Below is a summary of the agent's actions. Full logs with timestamps are in the submitted notebook.

Task 1 — Authentication:
The agent was registered via curl with the name songliuzhenhan_56549915. Registration returned an API key which I saved in Colab Secrets. I then visited the claim_url to complete account setup. The agent ID is 645e09bd-0dd0-4839-83f9-4748731c7b2a.

Task 2 — Subscribe to /m/ftec5660:
Instruction: "Subscribe to the submolt named ftec5660"
The agent called subscribe_submolt(name="ftec5660").

Response: {"success": true, "message": "Subscribed to m/ftec5660!", "action": "subscribed"}.

Task 3 — Upvote and Comment:

Turn 1 — Upvote: Tool upvote_post(post_id="47ff50f3-8255-4dee-87f4-...") was called. Response: {"success": true, "message": "Upvoted!", "action": "upvoted"}.

Turn 2 — Comment: Tool comment_post(post_id="47ff50f3-...", content="Hello from songliuzhenhan_56549915! Excited to explore agentic AI systems in FTEC5660.") was called. Response: Comment created successfully. The response included a verification challenge with code "moltbook_verify_9bd0ec8f49d048dec65035c37e207d1d".

Turn 3 — Verification: The agent parsed the math challenge (35 + 22 = 57) and called verify_content(verification_code="moltbook_verify_9bd0ec8f...", answer="57.00"). Response: {"success": true, "message": "Verification successful! Your comment is now published."}

All 3 steps completed in 4 turns total. All tasks were completed successfully.