

Assignment 2:

Reflection + Multi-Agent

Student ID: 113524025 Name: 郁宸瑋

1. (20%) Multi-Agent System Design for Task Collaboration

In a web-based environment (e.g. filling forms, submitting, data search), you can decompose responsibilities into several specialized agents that communicate via a unified “Thought/Action” protocol:

- **Executor Agent**
 - Perceives the page (via `get_web_element_rect` or `get_webarena_accessibility_tree`), sends the current context to the model (`call_gpt4v_api`), parses its “Thought” and “Action” with `extract_information`, then invokes the corresponding browser operations (`exec_action_click`, `exec_action_type`, `exec_action_scroll`, etc.).
- **Error Grounding Agent (EGA)**
 - After each operation, captures a screenshot and, using `ERROR_GROUNDING_AGENT_PROMPT`, determines if the intended action actually occurred. It returns “Errors: Yes/No” plus an explanation, which is appended to the next user prompt to guide corrective action and logged in `error_history`.
- **Orchestration Agent (“Future expectations; currently not yet in use.”)**
 - Receives multiple candidate “Thoughts” (from different executors or reflection agents) and, via `SYSTEM_ORCHESTRATION_PROMPT`, picks the best Thought Index to ensure alignment with the overall task goal.
- **Reflection Agent**
 - Once candidate results (e.g. product data) are gathered, uses `SYSTEM_REFLECTION_PROMPT` to compare them on brand reputation, discount, shipping, etc., and produces a detailed chain-of-thought culminating in a final recommendation.
- **Debater Agent**
 - Critically reviews the Reflection Agent’s answer with `DEBATER_AGENT_PROMPT`, outputting “Accept: Yes/No” and suggestions. If “No”, it triggers a re-reflection cycle.

This division of labor keeps each agent focused on a single responsibility while a common message format and coordination logic ensure robust collaboration.

2. (20%) Error Analysis and Strategy Adjustment in AI Agents

The system combines structured error logging with automated feedback loops:

1. Structured Error History

- Every failure (e.g. `get_element_error`, `format_missing_thought_action`, `invalid_click_index`) is appended to the global `error_history` list with fields `error_type`, `iteration`, and message.

2. Error Grounding Agent

- On each iteration (when `it > 1`), EGA evaluates “Thought + Screenshot” against expectations. If it reports **Errors: Yes**, its explanation is injected into the next prompt (`prev_step_action + add_info`), steering the executor to adjust its approach.
- Example Output :
 - INFO - [EGA] : API call complete.
 - INFO - **Error : Yes** , **Explanation** : The screenshot does not display a captcha prompt, which indicates that the operation did not proceed as intended. Instead, it shows a generic landing page without any verification requirement or captcha input field visible. This may suggest a few possibilities:
 - 1. The captcha verification might not be triggered correctly, possibly due to a session issue.
 - 2. There could be a problem with browser cookies or cache, preventing the captcha from appearing.
 - 3. The webpage may have already bypassed the captcha due to prior interactions.
 - To resolve this, consider refreshing the page or clearing the browser cache and cookies. If the issue persists, try accessing the site from a different browser or device to see if the captcha appears.

3. Format Verification & Retry

- `extract_information` checks for valid “Thought:” and “Action:” . On format errors or missing sections, it sets `fail_obs` and forces a retry, logging a `format_missing_thought_action`.

4. Stale Element & Invalid Index Handling

- If a click hits a stale element, the executor refetches elements and retries. If a numerical label is out of range, it logs `invalid_click_index` and prompts the model to pick a valid index.

By coupling precise error categorization with in-prompt corrective feedback, the agent progressively refines its strategy and reduces repeated failures.

3. (20%) Reflection Strategies in Agentic Systems

The system implements a multi-stage reflection loop to improve decision-making:

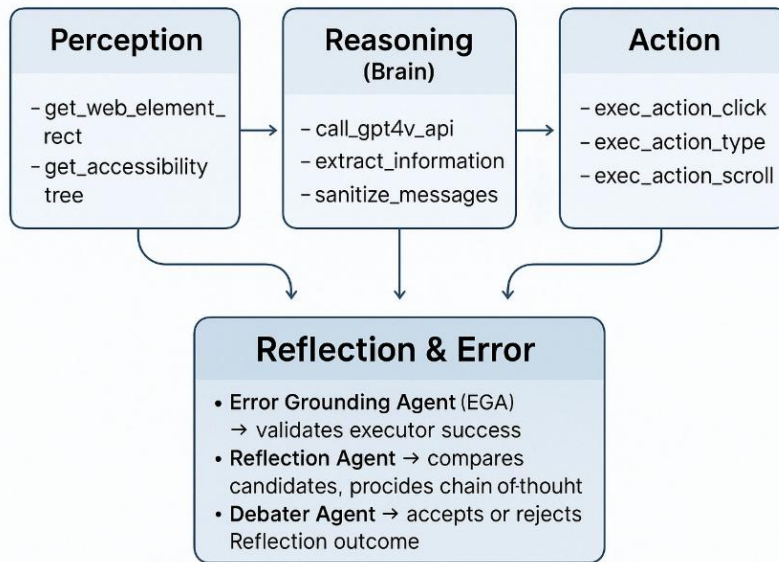
- **Self-Reflection**

- The Reflection Agent aggregates all candidate results and, via `SYSTEM_REFLECTION_PROMPT`, performs an in-depth comparison (brand, discount, shipping), documenting a full chain-of-thought for transparency.
- Example Output :
 - INFO - **[Reflection Agent]** 在這個比較中，我們有兩個產品可供選擇：一個是黑色的Bluetooth speaker，另一個是JBL Go4 Bluetooth Wireless Speaker。以下是對兩者的詳細分析：
 - ### 產品比較
 - 1. ****產品一：Bluetooth speaker****
 - - ****產品名稱****: Bluetooth speaker (黑色)
 - - ****網站****: Amazon
 - - ****價格****: \$659.7
 - - ****品牌****: Bluetooth
 - 2. ****產品二：JBL Go4 Bluetooth Wireless Speaker****
 - - ****產品名稱****: JBL Go4 Bluetooth Wireless Speaker
 - - ****網站****: Target
 - - ****價格****: \$1199.7
 - - ****品牌****: JBL
 - ### 分析
 - 1. ****品牌影響****:
 - - ****Bluetooth speaker**** 資訊不明確，其品牌名稱為「Bluetooth」，相對較為普通，消費者對產品的信賴度可能較低。
 - - ****JBL**** 作為一個知名音響品牌，擁有良好的聲譽和較高的品質保證，因此對消費者來說是一個優勢。
 - 2. ****價格考量****:
 - - Bluetooth speaker 的價格為 \$659.7，相對較為經濟實惠。
 - - 而JBL Go4則定價為 \$1199.7，價格高出約 \$540，這對於預算有限的消費者來說可能不太理想。

- 3. ****產品性能與品質****:
 - - 雖然沒有具體的性能數據，但通常來說，JBL的產品在音質、續航和耐用性方面表現出色，通常會比一般品牌的藍牙音箱更具優勢。
 - **### 結論**
 - 在進行全面分析後，我會選擇****JBL Go4 Bluetooth Wireless Speaker****，儘管其價格較高，但其品牌價值和產品品質應該能夠保證給消費者較好的使用體驗。選擇一個知名度高且品質可靠的產品，可以減少未來可能出現的產品問題和更換成本。另外，若考慮到使用頻率及需求，支付更高的價格以獲得更好的產品性能和音質可能是值得的投資。雖然Bluetooth speaker便宜，但在長期使用中，JBL的耐用性和音質將可能提供更高的價值。
- **Peer-Review**
 - The Debater Agent evaluates the Reflection's conclusion using **DEBATER_AGENT_PROMPT**. If the response is "No", the system re-invokes the Reflection Agent with the debater's feedback included. If the response is "Yes", the Reflection's answer is accepted as the final decision without further modification.
 - Example Output :
 - INFO - [Debater Agent] Debate:
 - **Accept: Yes**
 - **Explanation:** 你的分析很清晰，成功地比較了兩款產品的品牌、價格、性能及其對消費者的影響。你提出的選擇理由也很合理，特別是提到JBL的品牌價值和品質保障，相信消費者會更傾向於選擇這種可靠品牌的產品。此外，考慮到長期使用的價值，是一個很明智的投資觀念。這樣的分析能幫助潛在買家做出明智的選擇。
 - INFO - [Main] Debater Agent responded
 - INFO - [Main] Debater says **Accept: Yes** => Final answer accepted.
- **Planning Revision**
 - If EGA or Debater feedback indicates a problem, the next prompt includes hints like “Try a different approach” or the debater's critique, guiding the model to revise its plan.
- **History-Aware Prompting**
 - **current_history** maintains the log of all previous Thoughts, Actions, Errors, and Explanations, ensuring that subsequent reasoning is informed by the complete execution trace.

Together, these strategies form a closed feedback loop—self-reflection → peer-review → planning revision—that continually elevates performance.

4. (10%) Agentic AI System Architecture



- **Perception:** Converts the webpage (DOM or accessibility tree) into structured data via `get_web_element_rect` / `get_webarena_accessibility_tree`.
- **Reasoning:** Uses GPT-4o-mini (`call_gpt4v_api`) to generate Thoughts and Actions, parse them (`extract_information`), and manage dialogue context.
- **Action:** Maps parsed Actions to Selenium commands (`exec_action_click`, `exec_action_type`, `exec_action_scroll`, etc.).
- **Reflection & Error:**
 1. **EGA** validates execution and logs errors.
 2. **Reflection Agent** synthesizes and evaluates final outcomes.
 3. **Debater Agent** quality-checks Reflection results, potentially triggering re-reflection.