Assignment 1:
Reflexive Web Agent with Tools Use

Student ID: 113524025, Name: 郁宸瑋

1. **(10%) Describe your Agentic AI application scenario, including target users, use cases, and problems to be solved.**
   - **Target Users**
     - **Consumers**: Shoppers who want to compare prices across different e-commerce platforms to find the best deal.
     - **E-commerce Analysts**: Professionals analyzing market trends and price fluctuations.
     - **Automated Shopping Bots**: AI-powered systems that assist users in purchasing products at the lowest possible price.
   - **Use Cases**

     1. **Automated Price Comparison**

        - The AI agent searches for a specific product across multiple e-commerce websites, extracts product names, and compares prices to find the cheapest option.

     2. **Real-Time Price Monitoring**

        - The agent can track the price fluctuations of a product over time, alerting the user when it reaches the lowest price.

     3. **E-commerce Market Research**

        - Businesses can use the AI agent to gather competitive pricing data and adjust their pricing strategies accordingly.

   - **Problems to be Solved**
     - **Time-consuming manual comparison**
       - Searching for the same product on multiple platforms is tedious; this agent automates the process.
     - **Inconsistent product listings**
       - Different websites may have varying product names and descriptions, making manual comparison difficult.
     - **Hidden discounts or misleading prices**
       - Some websites show discounts that are only applied at checkout. The AI ensures accurate price comparison.
2. **(10%) Analyze at least 2 potential technical challenges in implementation and propose preliminary solutions.**

   (1) Ensuring Accurate and Fair Price Comparisons Across Websites
   - **Problem**

- Different websites display product prices in varying ways:
  - Some include taxes, shipping fees, or discounts.
  - Others show misleading “list prices” with hidden discounts that appear at checkout.
  - Some websites load prices dynamically using JavaScript, making direct extraction difficult.

- **Solution**
  - Use Selenium to Ensure Full Page Load Before Extracting Prices，**def wait_for_page_load(driver, timeout=20)**
    - This ensures that prices appearing **after JavaScript execution** are captured.
  - **New Feature: "Refresh" to Handle Page Load Failures**
    - Some websites might **fail to load prices correctly** due to slow networks or anti-bot measures.
    - The newly implemented "Refresh" functionality allows the agent to reload the page if necessary，**exec_action_refresh(driver_task)**
    - **Impact**: Prevents missing or incomplete price data.
  - **Standardizing Extracted Prices Across Platforms**
    - Normalize prices (remove currency symbols, commas, and convert to float)，**clean_price(price_text)**
    - Helps ensure that **final price comparisons are accurate**.

(2) Handling UI Layout Variations for Consistent Price Extraction

- **Problem**
  - Websites have different UI structures, causing:
    - Prices becoming invisible due to improper scaling (e.g., mobile view or zoomed-in content).
- **Solution**
  - **Leverage Web Accessibility Tree for Consistent Element Detection**
    - **New Feature: "Zoom" to Handle Layout Scaling Issues**
      - **If product details are too small or cut off, the "Zoom" function dynamically adjusts the page: def exec_action_zoom(info, driver_task)**
      - **Impact**: Prevents UI rendering issues that might obscure the price.

(3) Ensuring Reliable Decision-Making Using Historical Responses

- **Problem**
  - The AI agent extracts multiple product listings, but selecting the most relevant product isn't just about finding the lowest price.
  - Some low-cost products may have lower quality, misleading descriptions, or be incorrectly classified.
  - Without considering previous extractions, each decision is made

in isolation, potentially leading to inconsistent or suboptimal recommendations.Prices becoming invisible due to improper scaling (e.g., mobile view or zoomed-in content).

- **Solution**
  - Store and Retrieve Past Responses for Better Decision-Making
    - Instead of making a decision per product, consolidate all options and use LLM analysis to determine the best choice.
    - When a product is extracted, save its details:
      - if product_result_temp:
      - product_results.append(product_result_temp)
      - logging.info(f"Website {website} result:
      - {product_result_temp}")
      - print(f"Website {website} result:",
      - product_result_temp)
      - else:
      - logging.info(f"No valid product result extracted
      - for website: {website}")
        - Ensures all extracted products are stored before selecting the best one.
  - Use LLM-Based Decision Making to Choose the Best Product
    - Instead of blindly selecting the lowest price, analyze products using an LLM-generated prompt:
      - if product_results:
      - decision_prompt = "Based on the following product
      - information, please choose the most appropriate
      - product and explain your reasoning process:\n"
      - for res in product_results:
      - decision_prompt += f"Product: {res['product']},
      - Website: {res['website']}, Price: ${res['price']}\n"
      - decision_prompt += "\nPlease answer in the
      - following format:\nProduct: <Product_Name>,
      - Website: <Website>, Price: $<Price>\n\nInclude your
      - full chain-of-thought in your answer."
      -
      - messages.append({
      - "role": "user",
      - "content": decision_prompt
      - })
        - This method ensures decisions are based on more than just price, taking into account product descriptions, brands, and quality.

3. **(20%) Explain how your system implements the complete cycle of environment perception, decision making, and action execution.**

(1) Perception

- The AI agent uses **Selenium** to execute the following steps:
  - Access the **target website**(e.g., Amazon, Target).
  - Identify **all interactive elements** (search bars, buttons, etc.) using **`get_web_element_rect()`**.
  - Capture a **screenshot of the webpage** using **encode_image()** and convert it into a format compatible with **GPT-4o-mini** for processing.
  - Example (Searching for PlayStation 5 on Target)
    - Action: Type [13]; playstation 5 console(At this stage, the AI agent has identified `tag[13]` as the search input field.

(2) Decision-Making

- The AI agent interacts with **GPT-4o-mini** via **call_gpt4v_api()** to analyze webpage content and determine:
  1. **Whether the search results are correct** (e.g., a list of PlayStation 5 consoles).
  2. **Which product is the cheapest** (e.g., the Astro Bot bundle priced at $399.99).
  3. **How to extract the product name, website, and price** in the required format.
- Example (Amazon Search Results)
  - Thought: The cheapest PlayStation 5 console listed is "Sony PS5 PlayStation 5" at $429.99.
  - Action: Answer; Product: Sony PS5 PlayStation 5, Website: Amazon, Price: $429.99

(3) Action Execution

- Based on GPT-4o-mini response, the AI agent performs the appropriate action:
  - **If the search results are incorrect**, it re-enters the search query:
    - Action: Type [13]; playstation 5 console
  - If the correct products are displayed but require further scrolling, it moves down:
    - Action: Scroll [WINDOW]; down
  - Once the lowest price is identified, it finalizes the answer:
    - Action: Answer; Product: PlayStation 5 Console (Astro Bot Bundle), Website: Target, Price: $399.99

(4) **Final AI Agent Decision**

- Once the AI agent retrieves pricing information from multiple websites, it compares all options using **print_message()** and **sanitize_messages()**. **GPT-4V then determines the final choice.**
- **Example: PlayStation 5**

Product Name	Website	Price
Sony PS5 PlayStation 5	Amazon	$429.99
PlayStation 5 Console (Astro Bot Bundle)	Target	$399.99

- **Final AI Decision:**
  - Thought: The PlayStation 5 Console (Astro Bot Bundle) at Target is priced at $399.99, which is lower than Amazon's option ($429.99). Additionally, the bundle may offer extra value with the included game.
  - Action: Product: PlayStation 5 Console (Astro Bot Bundle), Website: Target, Price: $399.99

4. **(30%) Design and execute 3 test tasks, analyze the results, and propose potential improvements based on the current implementation.**

Test Case 1: PlayStation 5 Console Price Comparison

Search Criteria:

Product: PlayStation 5 Console

Comparison Websites:

Amazon: https://www.amazon.com/

Target: https://www.target.com/

Result Analysis:

Product	Website	Price
Sony PS5 PlayStation 5	Amazon	$429.99
PlayStation 5 Console (Astro Bot Bundle)	Target	**$399.99**

Final Selection: Target ($399.99), **$30 cheaper than Amazon**.

Advantages:

1. The AI agent successfully retrieved PlayStation 5 prices from both websites and selected the cheapest option.
2. The AI noted that the Astro Bot Bundle might provide additional value but prioritized price in decision-making.

Issues:

- There might be **a discrepancy between listed prices and checkout prices** (e.g., discounts or hidden fees).
- The AI **did not attempt to use filtering options** (e.g., sorting by lowest price), which could improve efficiency.

Test Case 2: Portable Bluetooth Speaker Price Comparison

Search Criteria:

Product: A portable Bluetooth speaker with a water-resistant design

Comparison Websites:

Amazon: https://www.amazon.com/

Target: https://www.target.com/

Result Analysis:

Product	Website	Price

Portable Bluetooth Speaker (Unbranded)	Amazon	$19.99
JBL Go4 Bluetooth Wireless Speaker	Target	**$39.99**

Final Selection: Target($39.99), JBL Go4 Bluetooth Wireless Speaker ($39.99)
Advantages:

1. AI did not simply compare prices but also considered brand value and quality, ultimately selecting JBL over the cheaper but unbranded Amazon product.
2. Brand Reputation Consideration: JBL is a well-known brand, whereas the Amazon product lacks brand information, which may indicate lower quality.
3. Long-Term Value Consideration: JBL speakers are known for their reliability, potentially offering a better user experience.

Issues:

- Brand vs. Price Priority Should Be User-Selectable – Some users might still prefer the lowest-priced option.

Improvement Suggestions:

- Introduce User Preference Selection – Allow users to choose between "Lowest Price First" or "Brand Reputation First" so the AI can tailor its decision-making accordingly.
- Provide a More Detailed Product Specification Comparison – If AI can extract more detailed product specifications, it could help users make more informed decisions.

Test Case 3: Men's Running Shoes Price Comparison

1. Enhance Filtering Options

- **Currently, the AI agent doesn't use filtering options, such as:**
  - **"Sort by Lowest Price"**
  - **"Show Discounted Products"**
  - **"Filter by 4-star Ratings & Above"**
- **Suggested Improvement: Enable AI to sort results by lowest price first, reducing search time.**

2. Ensure Price Accuracy

- **The AI does not verify checkout prices, meaning it might select a product with a higher actual purchase price.**
- **Suggested Solution:**
  - **AI can attempt to "Add to Cart" and check the final checkout price.**
  -

3. Provide Multiple Price Comparison Modes

Mode	Key Considerations
Lowest Price Priority	**Simply selects the cheapest product**
Brand Reputation Priority	**Prioritizes well-known brands if the price difference is small**
Best Value (Cost-Performance)	**Balances price, brand, and customer ratings**