Homework1

Huo Chuanrui 1155245577

Solution

1. Design a function called bill_agent, which is a unified interface for user calling.

```
def bill_agent(user_query):
    system_msg = SystemMessage(content="""You are a financial agent.
    1. If the user asks for total spending (Query 1), call 'calculate_actual_spent'.
    2. If the user asks for original price (Query 2), call 'calculate_original_price'.
    3. If the query is unrelated, return exactly: 'I cannot answer this question.'
    """)

    ai_msg = llm_with_tools.invoke([system_msg, HumanMessage(content=user_query)])

    if ai_msg.tool_calls:
        tool_name = ai_msg.tool_calls[0]["name"]
        tool_to_call = {
            "calculate_actual_spent": calculate_actual_spent,
            "calculate_original_price": calculate_original_price
        }[tool_name]
        return tool_to_call.invoke({})
    else:
        return ai_msg.content
```

   In this function, we design the prompt to tell llm that he is a financial agent that only answer such two questions. After receiving the command from user_query, it will decide which tool it should call.

2. Tool Introduction

   To solve these two questions, I design 2 tools for them, different questions for different prompts, First one is the calculation of actual spent, that's a easy one , I tell the llm to identify the amount with the words "Total" ,"Amount Due" and so on. Because of Gemini is a model supporting identifying the photo, so we didn't choose traditional OCR to extract the Text which may lower the accuracy.

```
@tool
def calculate_actual_spent():
    """Query 1: calculate the actual spent """
    prompt = """
Analyze each bill image.
    Find the FINAL 'Total', 'Amount Due', or 'Grand Total' for each bill.
    Ignore subtotals. Ignore savings.
    Extract these final numbers into a list.

Return STRICT JSON ONLY (no markdown, no extra text):
{
  "amounts": [480.20, 396.00, 160.10, ...]
}
Use exact numbers from the receipt. If cannot find for a bill, use 0.0.
"""

    data = analyze_bills_with_vision(prompt)

    amounts = data.get("amounts", [])
    total = sum(float(a) for a in amounts)

    return total
```

The second tool is calculation of original price which is more sophisticated than the first one. My solution for this is to add all discount value to the actual spent. So in this prompt, I tell the Gemini to identify all negative values with phrases like “OFF” above the “subtotal/小计”.

```
@tool
def calculate_original_price():
    """Query 2: calulate the original price (Paid + Discounts)"""
    prompt = """
Analyze each bill image independently in the order provided.

For each bill:
1. Identify the FINAL paid amount (the amount actually deducted, usually near 'OCTOPUS', 'VISA', 'Amount Deducted', 'Total Due'
   - Look for phrases like 'Amount Deducted', '扣除金額', 'OCTOPUS $xxx.xx'.
2. Identify EVERY discount/saving BEFORE the SUBTOTAL / 小計 line:
   - All negative amounts (e.g. -12.40)
   - Lines containing "Buy X Save", "% OFF", "5% OFF (CU-SCO)", "App upgrade", "MB upgrade", "包裝變形", "coupon"
   - Include ALL repeated negatives (e.g. multiple -12.40 for packaging deformation)
   - Do NOT include positive item prices or subtotals as discounts
   - Do NOT count rounding or change as discount

Return ONLY strict JSON (no markdown, no extra text):
{
  "bills": [
    {
      "bill_index": 1,
      "final_paid": 394.72,
      "discounts": [
        {"value": -12.40, "description": "包裝變形 -$12.40"},
        {"value": -12.80, "description": "Buy 2 Save $12.8 -$12.80"}
      ],
      "total_discount_absolute": 25.20
    },
```

3. After inputting different prompts for different questions, in the tools it will call the same function called analyze_bills_with_vision. In this function, it process the images first and then input the promopt and images to llm, then due to the possible error calculaion of llm, I use python code to sum the amount and return the final result.

```
def analyze_bills_with_vision(instruction, image_folder="/content"):
    if not os.path.exists(image_folder):
        print(f"Error: Folder {image_folder} not found.")
        return {"amounts": []}

    image_files = [os.path.join(image_folder, f) for f in os.listdir(image_folder)
                   if f.lower().endswith(('.png', '.jpg', '.jpeg'))]

    if not image_files:
        print("Error: No images found.")
        return {"amounts": []}

    json_instruction = instruction + """
    ------------------------------------------------------
    CRITICAL OUTPUT REQUIREMENT:
    Return ONLY valid JSON, no markdown (no ```json), no extra text outside the JSON.
    Follow the exact structure requested in the prompt.
    Be accurate and precise.
    ------------------------------------------------------
    """

    content = [{"type": "text", "text": json_instruction}]
    for img_path in image_files:
        content.append({
            "type": "image_url",
            "image_url": {"url": get_image_data_url(img_path)}
        })
```