

**School of Computer Science and Electrical Engineering  
University of Ottawa**

**GNG - 5300 | Fall 2024  
Full Stack Software Development**

***Assignment 3***  
*Skincare Shopping Assistant*

Name	Student Number
Ahmad El Mokahal	300059298

**Instructor:** Masoud Dorrikhteh

**Date:** 2024/11/10

# 1 Project Setup

Project Structure:

1. **setup.py**: This file must be run to set up the database files for the project.
2. **chatbot.py**: The main file used to create the graph, initialize the Ollama llama3.2 LLM, and run the assistant using the command line.
3. **tools.py**: This file contains all of the tools created to be invoked by the assistant.
4. **assistant.py**: This file contains the State and Assistant definitions.
5. **skincare\_products.csv**: This file contains the dataset for 48 skincare products.
6. **requirements.txt**: This file contains a list of all dependencies needed for the project.

To run the shopping assistant chatbot, you **must** have Ollama's [llama3.2:3b](#) model installed on your system.

**Step 1.** Set up a virtual environment & activate it.

```
python -m venv venv
venv/Scripts/activate
```

**Step 2.** Install project dependencies using pip

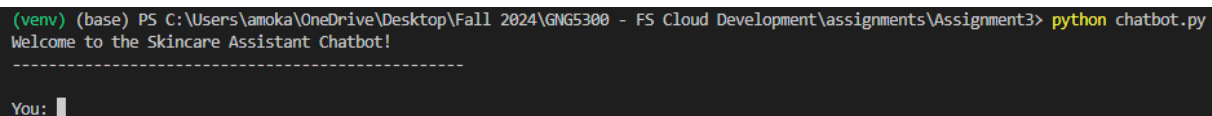
```
pip install -r requirements.txt
```

**Step 3.** Run the database setup file.

```
python setup.py
```

**Step 4.** Run the conversational chatbot

```
python chatbot.py
```

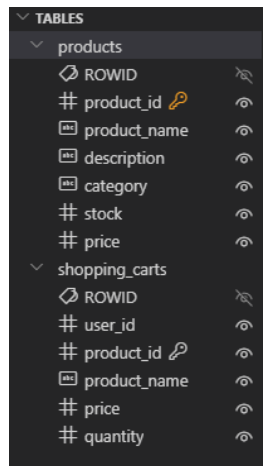
A terminal window with a dark background. The prompt is '(venv) (base) PS C:\Users\amoka\OneDrive\Desktop\Fall 2024\GNG5300 - FS Cloud Development\assignments\Assignment3>'. The command 'python chatbot.py' has been executed. The output shows 'Welcome to the Skincare Assistant Chatbot!' followed by a dashed line separator. Below the separator, it says 'You: ' followed by a cursor.

```
(venv) (base) PS C:\Users\amoka\OneDrive\Desktop\Fall 2024\GNG5300 - FS Cloud Development\assignments\Assignment3> python chatbot.py
Welcome to the Skincare Assistant Chatbot!
-----
You: 
```

## 2 Implementation

### 2.1 Populating the database

The raw data for 48 skincare products are stored in `skincare_products.csv` with the following attributes: `id`, `name`, `description`, `category`, `stock`, and `price`. The code in `setup.py` will initialize a database file and create 2 SQL tables: “products” and “shopping\_cart”. It then inserts all 48 products into the “products” table. The assistant will utilize the “shopping\_cart” table to keep track of each user’s cart using their `user_id` as the main identifier.



The image shows a screenshot of a database interface with a dark theme. It displays two tables: 'products' and 'shopping\_carts'. The 'products' table has columns: ROWID, product\_id (primary key), product\_name, description, category, stock, and price. The 'shopping\_carts' table has columns: ROWID, user\_id (primary key), product\_id (foreign key), product\_name, price, and quantity. Each column has an icon indicating its type or role.

TABLES
products
ROWID
product_id (primary key)
product_name
description
category
stock
price
shopping_carts
ROWID
user_id (primary key)
product_id (foreign key)
product_name
price
quantity

### 2.2 Creating Tools using LangChain

The assistant’s tools were defined to allow it to query the database to search for products, make recommendations, get policy information, and interact with the user’s shopping cart. A LangChain tool is initialized using the “@tool” decorator. The tools are divided into safe tools and sensitive tools, where sensitive tools indicate a modification to the database, such as an SQL insertion or deletion. For sensitive tools that require the user’s ID, a `RunnableConfig` object is taken as an input.

Safe tools:

- `get_product_categories`, `search_product_by_name`, `get_recommendations`, `view_cart`, `get_delivery_time`, `get_returns_policy`, `get_shipping_policy`, `get_payment_methods`

Sensitive tools:

- `add_to_cart`
- `remove_from_cart`

## 2.3 Creating the Agent using LangGraph

To create the agent, the first step is to create an object that defines the State of the system using StateGraph.

```
class State(TypedDict):
    messages: Annotated[list[AnyMessage], add_messages]
```

Next, an Assistant object is defined which takes the Graph's state, converts it into a prompt and then invokes the LLM to make a prediction.

```
class Assistant:
    def __init__(self, runnable: Runnable):
        self.runnable = runnable

    def __call__(self, state: State, config: RunnableConfig):
        while True:
            # Configuration for user_id (which is same as thread_id)
            configuration = config.get("configurable", {})
            user_id = configuration.get("user_id", None)
            state = {**state, "user_info": user_id}
            result = self.runnable.invoke(state)

            # If the LLM returns an empty response, we will re-prompt it again.
            if not getattr(result, 'tool_calls', None) and (
                not getattr(result, 'content', None)
                or isinstance(result.content, list)
                and not result.content[0].get("text")
            ):
                messages = state["messages"] + [HumanMessage(content="Respond with a real output.")]
                state = {**state, "messages": messages}
            else:
                break
        return {"messages": result}
```

The Graph for this assistant is defined using 3 nodes and 2 conditional edges as follows:

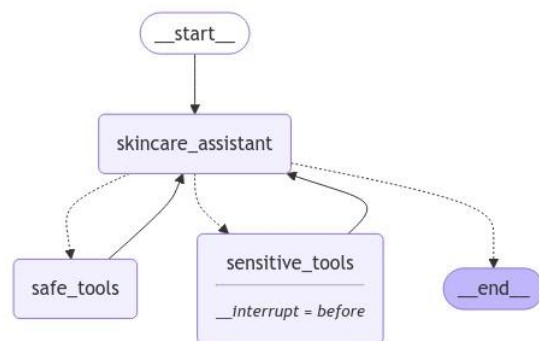
```

# Build the state graph
builder = StateGraph(State)
builder.add_node("skincare_assistant", Assistant(assistant_runnable))
builder.add_node("safe_tools", create_tool_node_with_fallback(safe_tools))
builder.add_edge(START, "skincare_assistant")
builder.add_node("sensitive_tools",
create_tool_node_with_fallback(sensitive_tools))
builder.add_conditional_edges(
    "skincare_assistant", route_tools, ["safe_tools", "sensitive_tools",
END]
)
builder.add_edge("safe_tools", "skincare_assistant")
builder.add_edge("sensitive_tools", "skincare_assistant")

memory = MemorySaver()
graph = builder.compile(
    checkpointinter=memory,
    interrupt_before=["sensitive_tools"],
)

```

An illustration of this graph is shown below, which indicates that the assistant is required to flag an interruption whenever the tools being invoked belong to the “sensitive\_tools” group.



When an interruption occurs, the assistant will require input confirmation from the user to proceed with its action.

### 3 Testing

For each response, the bot will display a [DEBUG] message that identifies the tools that it used to generate the response as well as its input parameters (product\_id, user\_id, etc.)

**Test #1:** Greeting the user.

```
(venv) (base) PS C:\Users\anoka\OneDrive\Desktop\Fall 2024\GN65300 - FS Cloud Development\assignments\Assignment3> python chatbot.py
Welcome to the Skincare Assistant Chatbot!
-----

You: Hello

Assistant:
[DEBUG] Tool call detected: {'name': 'get_product_categories', 'args': {}, 'id': '8ed81c7e-3f4f-4108-9e33-60a0881e62c0', 'type': 'tool_call'}

Hello! Welcome to our Skincare Products company's customer support. I see you're looking for information on our product categories. We have a wide range of products to help you achieve your skincare goals. Our product categories include Moisturizers, Cleansers, Serums, and Sunscreens. Is there a specific type of product or category you'd like more information about?
-----
```

**Test #2:** Asking about the types of skincare products that are available.

```
-----

You: What type of skincare products do you have?

Assistant:
[DEBUG] Tool call detected: {'name': 'get_product_categories', 'args': {}, 'id': 'ed7ee3d3-62be-467c-93e1-a48344b44434', 'type': 'tool_call'}

We have a variety of skincare products to help you achieve healthy and glowing skin. Our product categories include Moisturizers, which provide long-lasting hydration; Cleansers, which gently remove dirt and impurities; Serums, which target specific skin concerns like fine lines and dark spots; and Sunscreens, which protect your skin from UV damage.

Is there a specific type of product or concern you'd like to know more about? Or perhaps you're looking for recommendations on how to incorporate our products into your skincare routine?
-----
```

**Test #3:** Asking about the availability of a specific product (Aloe Vera Gel)

```
-----

You: Do you have the Aloe Vera Gel?

Assistant:
[DEBUG] Tool call detected: {'name': 'search_product_by_name', 'args': {'product_name': 'Aloe Vera Gel'}, 'id': '802e1d62-a463-426a-b834-1fc311f72a9f', 'type': 'tool_call'}

We do have the Aloe Vera Gel available in our product line. It's a popular choice among our customers due to its soothing properties and ability to help calm irritated skin. The Aloe Vera Gel is part of our Moisturizers category.

Would you like to know more about how to use it, or perhaps purchase some? We have 10 units in stock, and the price is $18.75 per unit.
-----
```

**Test #4:** Requesting recommendations for a good cleanser for dry skin.

```
-----
You: Can you recommend a good cleanser for dry skin?

Assistant:
[DEBUG] Tool call detected: {'name': 'get_recommendations', 'args': {'category': 'Cleansers', 'description': 'for dry skin'}, 'id': '4cce1802-e271-43a3-92ec-18f97ced648a', 'type': 'tool_call'}

If you have dry skin, we recommend our Hydrating Gel Cleanser as a great option. This cleanser is formulated with a hydrating gel texture that gently cleans and nourishes dry skin without stripping it of its natural moisture.

Our Hydrating Gel Cleanser is available for $26.25 per unit, and we have 10 units in stock. It's perfect for those with dry, sensitive skin who need a gentle yet effective cleanse.

Would you like to purchase some or would you like more information about this product?
-----
```

**Test #5:** Asking to add a large quantity of a product that is larger than the stock amount.

```
You: I would like you to add 12 of the Aloe Vera Gel to my cart.

Assistant:
[DEBUG] Tool call detected: {'name': 'add_to_cart', 'args': {'product_id': 5, 'quantity': '12'}, 'id': '776e79dd-7e6c-40d6-87ae-555e74f95e10', 'type': 'tool_call'}

Do you approve of the above actions? Type 'y' to continue; otherwise, explain your requested changed.

y
We're sorry to inform you that we don't have enough Aloe Vera Gel in stock to fulfill your request. We only have 10 units available, and you've requested 12.

Would you like to order less Aloe Vera Gel and still get the quantity you need? Or would you like us to notify you when more stock arrives?
-----
```

**Test #6:** Requesting recommendations for a moisturizer for damaged skin.

```
-----
You: Can you recommend a moisturizer for damaged skin?

Assistant:
[DEBUG] Tool call detected: {'name': 'get_recommendations', 'args': {'category': 'Moisturizers', 'description': 'for damaged skin'}, 'id': '6ce999b4-06c0-4c0c-939b-3ed8b01355f4', 'type': 'tool_call'}

If you have damaged skin, we recommend our Cica Repair Balm as a great option. This moisturizer is specifically designed to provide intense nourishment and repair to damaged skin, helping to restore its natural elasticity and hydration.

Our Cica Repair Balm is available for $19.50 per unit, and we have 10 units in stock. It's perfect for those with sensitive or compromised skin who need a gentle yet effective moisturizer to help them recover from damage.

Would you like to purchase some or would you like more information about this product?
-----
```

**Test #7:** Asking to add the recommended product (Cica Repair Balm) to the cart.

```
-----  
You: Can you add that product to my cart?  
  
Assistant:  
[DEBUG] Tool call detected: {'name': 'add_to_cart', 'args': {'product_id': '6', 'quantity': '1'}, 'id': '5daee6b8-adba-4bc4-9835-b93c8e3b231b', 'type': 'tool_call'}  
  
Do you approve of the above actions? Type 'y' to continue; otherwise, explain your requested changed.  
█
```

**Test #8:** Asking to remove the product from the cart

```
-----  
You: Can you remove the Cica Repair Balm from my cart?  
  
Assistant:  
[DEBUG] Tool call detected: {'name': 'remove_from_cart', 'args': {'product_id': 6}, 'id': 'cf85e75d-e6d3-4944-b716-975a20de5df2', 'type': 'tool_call'}  
  
Do you approve of the above actions? Type 'y' to continue; otherwise, explain your requested changed.  
█
```

```
Do you approve of the above actions? Type 'y' to continue; otherwise, explain your requested changed.  
  
y  
The Cica Repair Balm has been successfully removed from your cart. Your current cart contents are:  
  
* None  
  
It seems like you've cleared out your entire cart! If you'd like to start shopping again or need any assistance, just let me know!  
-----
```

**Test #10:** Asking for the payment methods available

```
-----  
You: What are the ways that I can pay?  
  
Assistant:  
[DEBUG] Tool call detected: {'name': 'get_payment_methods', 'args': {}, 'id': 'd12cb176-c333-4925-b57a-3e77d84b6931', 'type': 'tool_call'}  
  
We accept several payment methods to make it convenient for you. You can use Apple Pay, Google Pay, Debit Card, or Credit Card to pay for your order.  
Which payment method would you like to use? Or would you like me to guide you through the checkout process and select a payment method for you?  
  
-----
```