GenAI Based ShopAssist Application

Introduction

In this project, task is to build ShopAssist AI, which is a laptop recommendation chat-bot that can:

- Interact with users interactively,
- Understand the user's laptop requirements, and,
- Recommend the most suitable laptops based on their needs and preferences.

Project Background

In today's digital age, online shopping has become the preferred option for many consumers. However, the vast array of choices and the lack of personalised assistance can make the shopping experience overwhelming and challenging. This chat-bot combines the power of LLMs and rule-based functions to provide accurate and reliable recommendations during the online laptop shopping experience.

Problem Statement

Given a dataset containing laptop information (product names, specifications, descriptions, etc.), build a chat-bot that parses the dataset and provides accurate laptop recommendations based on user requirements. This chat-bot, named ShopAssist AI, will

- Interact with users,
- Understand their laptop requirements,
- Recommend the most suitable laptops from a dataset based on their needs and preferences.

Primarily, this chat-bot will analyse the 'Description' column for each laptop, understand whether the user's requirements match the laptop's specifications and then forward a relevant laptop as a recommendation.

System Design / Architecture

This entire project will be divided into 3 stages:

Stage 1: Understanding User Requirement

This stage will actually interact with the user proactively and understand the user's requirements. It is very much necessary that user provides all the necessary information which are needed to filter out products from the database. Therefore, this stage will keep the conversation alive with the user until all the required information about the product is received. Once all the required information is extracted from the user, the same will be provided to the next stage in appropriate format.

Stage 2: Product Mapping & Extraction

All the details about the products (which are in database) are first gathered and information is extracted from these products in the same format which is provided as an output of Stage 1 Strict comparison are made to filter out only those products from the database which matches the user requirement provided by Stage 1 Hereby, only the filtered & top 3 products will be sent out to the next stage for recommending to the user

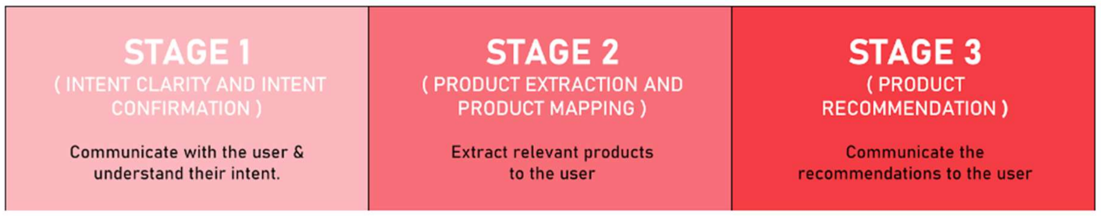
Stage 3: Product Recommendation

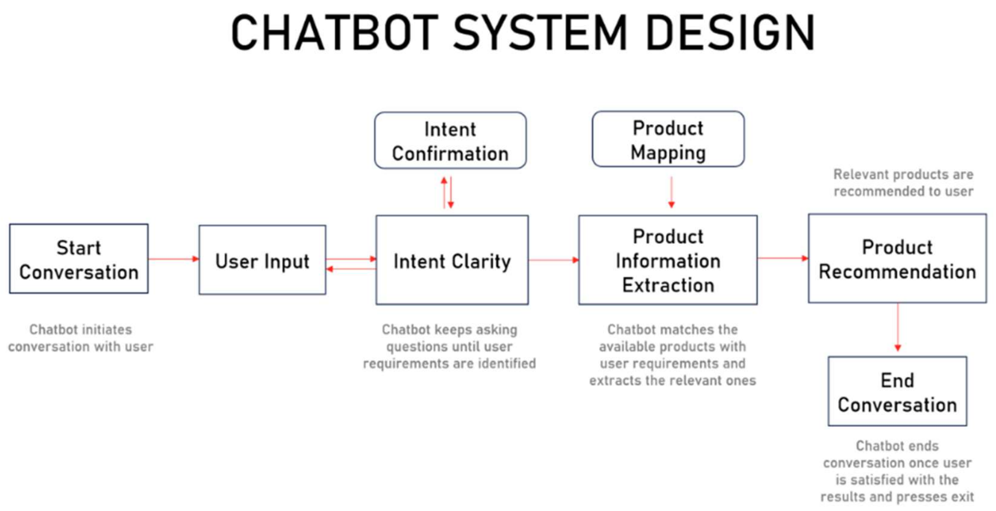
Here, the chat-bot indulges itself as a good sales representative to elaborate on the product which are provided by previous stage. This chat-bot needs to explain the filtered products to the user from the user perspective / profile by carefully detailing all the information accurately.

Finally, all the stages are combined to form an interactive chat-bot which can be deployed on web

System Architecture

ShopAssistAI follows a client-server architecture. Users interact with the web interface hosted on a server running the Flask application. The application interacts with OpenAI's API for conversation generation and moderation and retrieves and compares laptop data from an external database.


STAGE 1
(INTENT CLARITY AND INTENT CONFIRMATION)
Communicate with the user & understand their intent.
STAGE 2
(PRODUCT EXTRACTION AND PRODUCT MAPPING)
Extract relevant products to the user
STAGE 3
(PRODUCT RECOMMENDATION)
Communicate the recommendations to the user


CHATBOT SYSTEM DESIGN
Start Conversation
Chatbot initiates conversation with user
User Input
Intent Confirmation
Intent Clarity
Chatbot keeps asking questions until user requirements are identified
Product Mapping
Product Information Extraction
Chatbot matches the available products with user requirements and extracts the relevant ones
Relevant products are recommended to user
Product Recommendation
End Conversation
Chatbot ends conversation once user is satisfied with the results and presses exit

Implementation Details

The Flask application utilizes various functionalities:

- **Routing:** Maps user requests to appropriate functions based on URLs.
- **Conversation Management:** Handles conversation initiation, response generation through OpenAI's chat model, and conversation history maintenance.
- **User Input Processing:** Captures user input, performs moderation checks, and extracts user profiles from conversation history (converting user input string to JSON using OpenAI Function calling).
- **Recommendation Logic:** Compares user profiles with laptop data, validates recommendations, and generates recommendation text.

Major Functions

- `initialize_conversation()`: Initializes the variable conversation with the system message.
- `get_chat_completions()`: Takes the ongoing conversation as the input and returns the response by the assistant.
- `moderation_check()`: Checks if the user's or the assistant's message is inappropriate. If any of these is inappropriate, it ends the conversation.
- `intent_confirmation_layer()`: Evaluates if the chatbot has captured the user's profile clearly.

- `dictionary_present()`: Checks if the final understanding of the user's profile is returned by the chatbot as a Python dictionary.
- `compare_laptops_with_user()`: Compares the user's profile with the different laptops and comes back with the top 3 recommendations.
- `initialize_conv_reco()`: Initializes the recommendations conversation.

Function Calling API's

The Shop Assistance Project aims to enhance the shopping experience by integrating function calling APIs that streamline various customer interactions and backend operations

Benefits:

- **Enhanced User Experience:** By providing fast, accurate, and personalized responses, customers enjoy a more efficient and enjoyable shopping experience.
- **Operational Efficiency:** Streamlines backend operations and inventory management, reducing manual errors and improving overall efficiency.
- **Improved Support:** Facilitates better customer support through real-time communication and efficient issue resolution.

Implementing function calling APIs in the Shop Assistance Project optimizes customer interactions, backend processes, and support functions, resulting in a more integrated and user-friendly shopping experience.