

Project Report Of Shop Assist AI 2.0

In today's digital age, online shopping has become the go-to option for many consumers. However, the overwhelming number of choices and the lack of personalized assistance can make the shopping experience daunting. To address this, we have developed ShopAssist AI, a chatbot that combines the power of large language models and rule-based functions to ensure accurate and reliable information delivery.

Objective:

The key learning objectives of this project are:

- Understanding the multi-stage architecture behind the fully functioning chatbot system,
- Integrate OpenAI's Chat Completion APIs to provide a natural and contextually relevant conversation,
- Build an end-to-end chatbot solution that seamlessly interacts with users, understands their needs and delivers accurate and relevant recommendations.

Problem Statement:

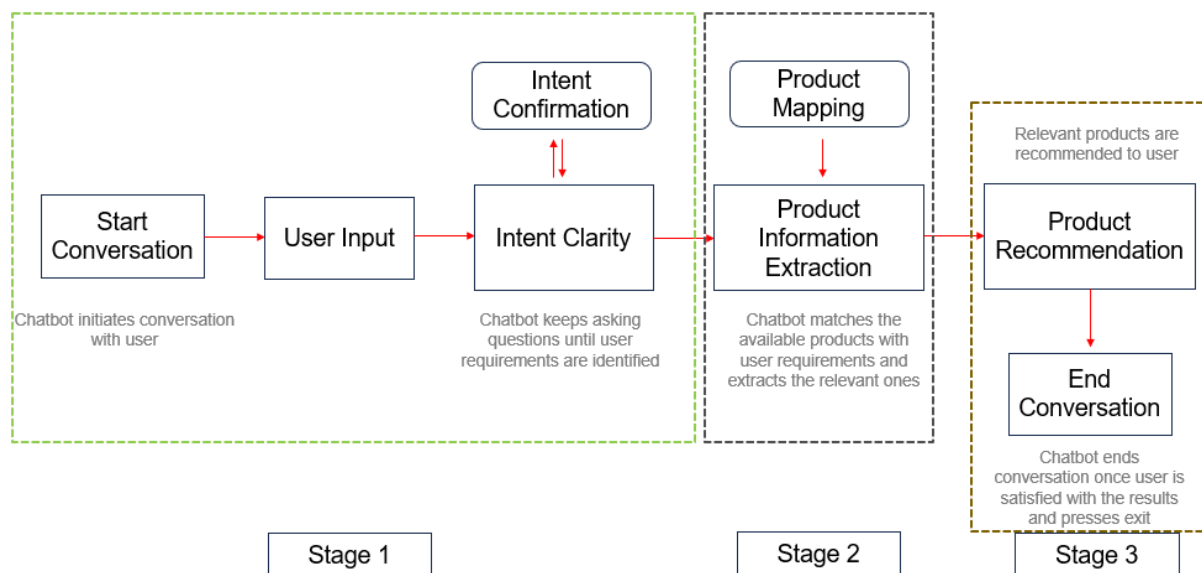
- Interact with users,
- Understand their laptop requirements and
- Recommend the most suitable laptops from a dataset based on their needs and preferences.

Design: there are three stages of the chatbot, which are as follows

Stage 1: Intent Clarity and Intent Confirmation

Stage 2: Product Extraction and Product Mapping

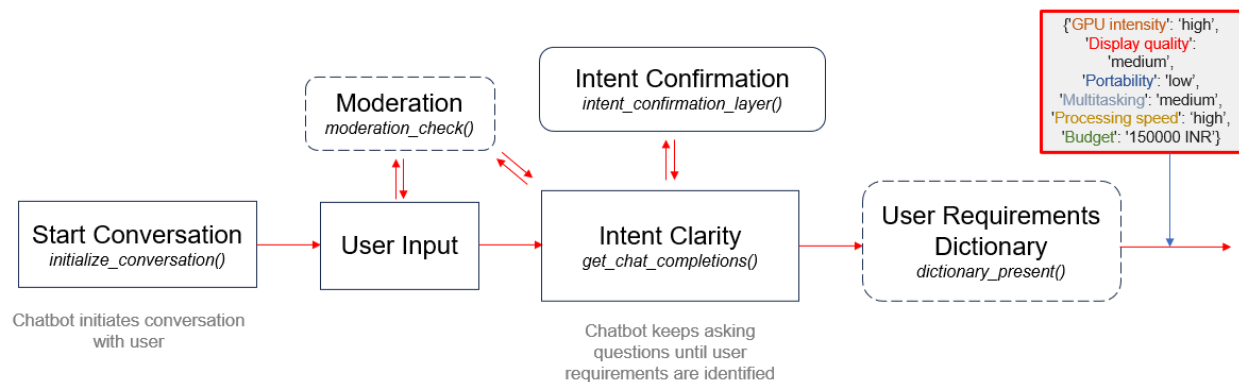
Stage 3: Product Recommendation



Stage 1 - Intent Clarity and Intent Confirmation

The first stage involves a conversation between the user and the AI system. Python functions such as `initialize_conversation()`, trigger the conversation, and those such as `get_chat_completions()` allows the conversation to continue with each conversation via LLM calls.

The stage includes an additional layer called `moderation_check()` to flag and discontinue conversations that contain unsafe or sensitive content.



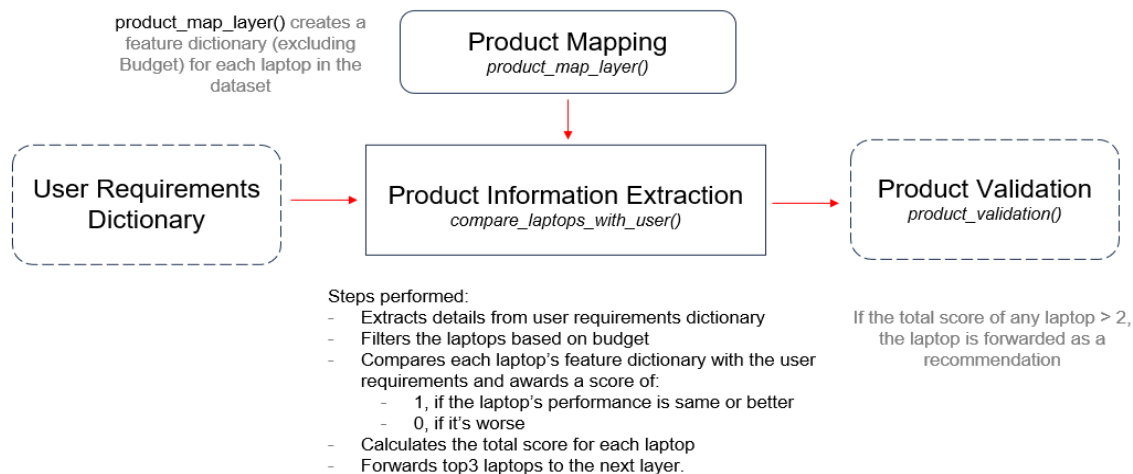
Initialising conversation and chat model completion: The conversation starts with the 'initialize_conversation()' function, where the AI system introduces itself and asks for the user's requirements. The subsequent messages from the user and the AI system are attributed to 'chat_model_completion()' function where the OpenAI's chat completion API is utilized for continuing the user conversation, until the requirements have been completely identified.

Intent confirmation: The AI system uses an 'Intent Confirmation layer' (intent_confirmation_layer()) as a flag (yes or no) to indicate whether all requirements of the user have been captured. During the conversation, if the system receives 'no,' it understands that further questions are needed to capture all the requirements.

User requirements dictionary: Once the 'Intent Confirmation' layer confirms that all requirements of the user have been captured (i.e. 'yes' flag), it passes the requirements to the 'User Requirements Dictionary' (dictionary_present() function) to be converted into a dictionary/dictionary-like object to the user_req variable.

Stage 2 - Product Mapping and Information Extraction

The second stage of the system is the 'Product Mapping and Information Extraction' stage. This stage filters the laptops as per the user requirements dictionary captured in the previous stage and uses it to present the top three laptop recommendations to the user.



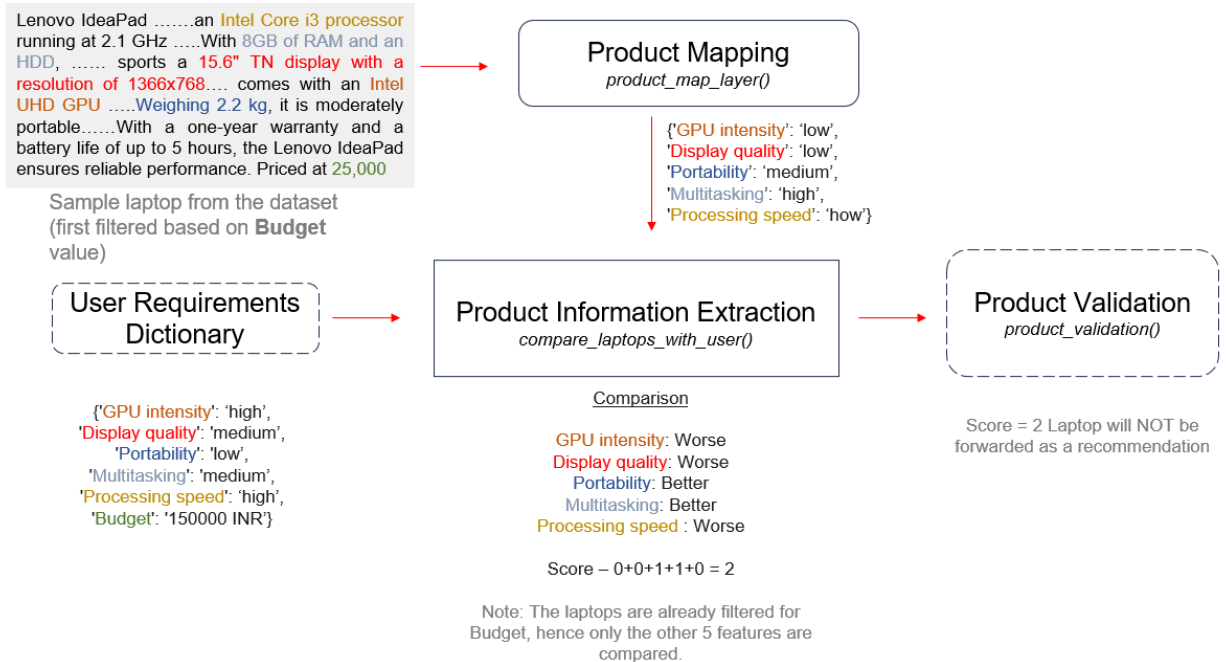
Part 1 (Product Mapping): You create a feature dictionary for each laptop from its given description. This is done using the `product_map_layer()` function, which extracts key features and criteria from laptop descriptions.

- This function extracts the primary laptop features from the detailed description of any laptop (such as GPU intensity, display quality, portability, multitasking, processing speed, etc.). These entries are stored as key-value pairs of a dictionary `lap_spec`.
- Once these values are extracted, they are mapped with the appropriate classification value (low, medium or high) defined inside rules.
- To perform this mapping, you need a detailed set of rules from your heuristics, which the LLM model is not privy to.
- Since this operation is independent from any user input, you need to execute this function once for all the laptops.

Part 2 (Information Extraction): So, till this stage, you have two dictionaries: `lap_spec` and `user_req`. All you need to do now is to determine how similar a laptop's features (stored in `lap_spec`) are to the user's requirements (stored in `user_req`). But before you compare these two dictionaries, they need to be converted from a string of dictionaries to just a dictionary. This is done using the `dictionary_present()` function.

➤ Example:

Stage 2: Example 2



Stage 3 - Product Recommendation

Finally, you have reached the product recommendation layer. It takes the output from the 'compare_laptops_with_user' function in the previous layer and provides the recommendations to the user. The broader process happening in this stage can be summarized as follows:

- The product validation layer will recommend a maximum of three laptops to the product recommendation layer. This ensures that the user is presented with a manageable number of laptop options to choose from.
- It is possible that no laptops meet the score threshold of three, in which case the product validation layer will feed 'None' or 'No laptops matched' to the product recommendation layer. In such cases, the AI system will be instructed to connect the user to a human expert.



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.

Conversation of chatbot flow:

Chatbot Assistant: Hello! I am an intelligent laptop gadget expert and I am here to help you find the best laptop for your needs. Please let me know what you are looking for in a laptop and any specific requirements you may have. **User:** I need a laptop for coding.

Chatbot Assistant: Great! As a coder, having a laptop with good processing speed is crucial for running multiple development tools and applications smoothly. Do you typically work on large codebases or complex algorithms that require intense processing power? Understanding the intensity of your coding tasks will help me recommend the most suitable laptop for you. **User:** I work with image processing and deep learning applications.

Chatbot Assistant: Thank you for providing that information. Working with image processing and deep learning applications requires a laptop with high processing speed

and GPU intensity. These tasks can be computationally demanding, so a strong GPU is essential for efficient performance. Do you also require a high-resolution display for better visualization of images and data? Understanding your requirements will help me tailor my recommendations accordingly. **User**: Yes. A high-resolution display would be ideal.

Chatbot Assistant: Great to know that a high-resolution display is important to you. Along with high GPU intensity and processing speed, having high display quality is crucial for working with image processing and deep learning applications. To better understand your needs, do you prefer a laptop that is highly portable and easy to carry around, or do you primarily work from a fixed location? Understanding your preference for portability will help me recommend the most suitable laptop for you. **User**: Portability is not a priority but ideally the laptop should weigh around 2 kgs.

Chatbot Assistant: Got it, portability is not a top priority but you prefer a laptop that weighs around 2 kgs. This indicates that you need a balance between portability and performance. Given your focus on image processing and deep learning, high GPU intensity, processing speed, and display quality are still crucial factors. Lastly, could you please specify your budget for the laptop? This information will help me narrow down the options within your price range while meeting your specified requirements. **User**: around 150000

{'GPU intensity': 'high', 'Display quality': 'high', 'Portability': 'medium', 'Multitasking': 'medium', 'Processing speed': 'high', 'Budget': 150000}

Variables extracted!

Chatbot Assistant:

Thank you for providing all the information. Kindly wait, while I fetch the products:

1. Acer Predator: Core i7 processor, 16GB RAM, SSD storage, NVIDIA GTX graphics, 17.3" IPS display (1920x1080), Windows 10, 3.2 kg, Dual Cooling Fans, 5 hours battery life, 1-year warranty, Price: Rs. 80,000

2. MSI GL65: Core i7 processor, 16GB RAM, HDD+SSD storage, NVIDIA GTX graphics, 15.6" IPS display (1920x1080), Windows 10, 2.3 kg, RGB Keyboard, 4 hours battery life, 2-year warranty, Price: Rs. 55,000

3. Lenovo ThinkPad: Ryzen 7 processor, 16GB RAM, SSD storage, NVIDIA GTX graphics, 14" IPS display (2560x1440), Linux, 1.6 kg, Backlit Keyboard, 6 hours battery life, 3-year warranty, Price: Rs. 60,000

Based on your user profile and the laptops available, the Acer Predator seems to align well with your requirements. It offers high GPU intensity and processing speed with a high-quality display. The laptop also provides decent portability and multitasking capabilities within your budget. **User**: what GPU does it use?

Chatbot Assistant: The Acer Predator laptop uses an NVIDIA GTX graphics processor. **User**: exit

Chatbot Assistant: If you have any more questions in the future, feel free to ask. Have a great day!

Conclusion: While ShopAssist AI is a robust system, there's always room for improvement. A notable challenge we've encountered is the complexity of managing multiple layers and components to achieve seamless and efficient interactions between the chatbot and users. Hence, we present this project, where you have multiple opportunities to enhance the customer experience by crafting a superior and more intelligent chatbot.

