**Problem Statement:**

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 Fashion-Finder AI, a chatbot that combines the power of large language models and rule-based functions to ensure accurate and reliable information delivery.

**Goal:**

Given a dataset containing information about fashion products (product names, specifications, descriptions, etc.), build a chatbot that parses the dataset and provides accurate fashion recommendations based on user requirements.

**System Architecture**

The chatbot should ask a series of questions to understand the user's needs better. To simplify this process, we have identified four key features that represent the user's preferences:

- Price

- Colour

- Rating

- Gender

At the end, confirm whether the user's requirements have been accurately captured. After that, the chatbot will present the top three products that best match the user's criteria and continue the conversation to assist in finding the ideal option.

A diagram of a flowchart

Description automatically generated

**Stage 1**

- Intent Clarity

- Intent Confirmation

**Stage 2**

- Product Mapping

- Product Information Extraction

**Stage 3**

- Product Recommendation

**Key functions of the Chatbot**

Here’s a quick overview of the key functions that make up the chatbot. We’ll explore each of them in detail later.

**initialize\_conversation():** This sets up the variable "conversation" with the system's initial message.

**get\_chat\_completions():** This takes the ongoing conversation as the input and returns the response by the assistant

**moderation\_check():** This checks whether the user's or the assistant's message can be considered inappropriate. If any one of these is inappropriate, the conversation is terminated.

**intent\_confirmation\_layer():** This feature assesses if the chatbot has accurately recorded the user's profile based on the assistant's response. This specifically determines whether the user's properties listed below have been recorded.

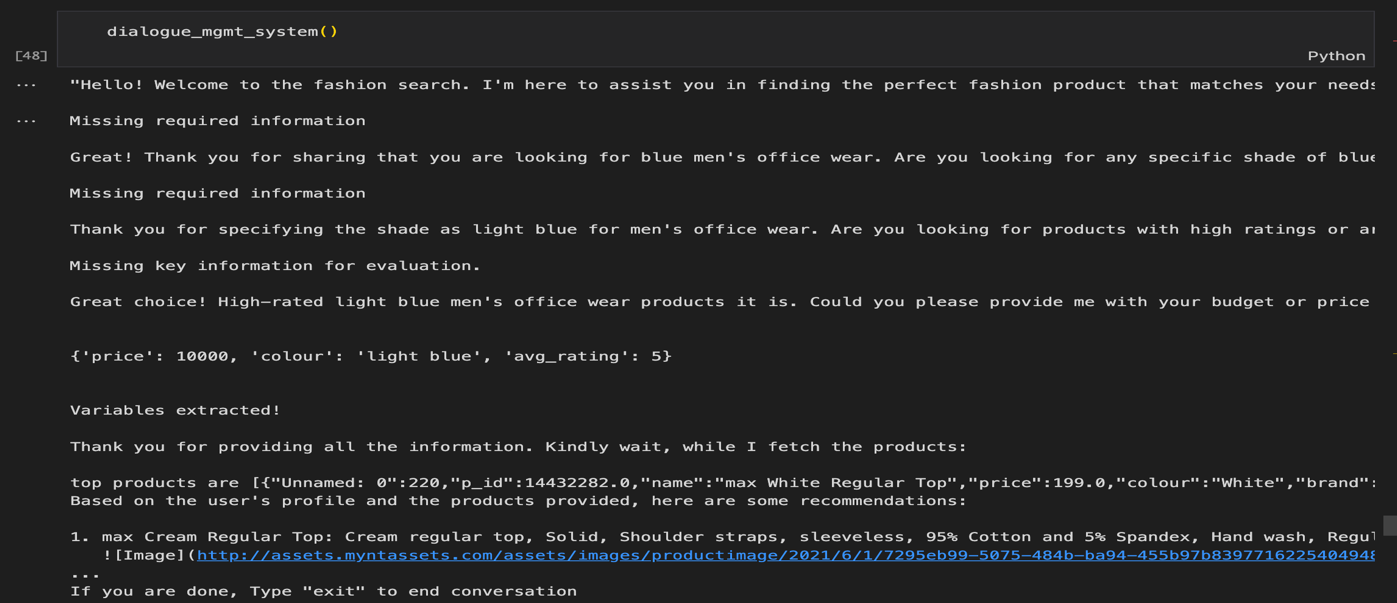
**dictionary\_present():** This function determines whether or not the chatbot returns the user's profile's final understanding as a Python dictionary. The information is extracted as a Python dictionary if one exists.

**compare\_laptops\_with\_user():** This function returns the top three suggestions after comparing the user's profile with the various products.

**initialize\_conv\_reco():** Starts the discussion about recommendations.

**Dialogue management system:**

Bringing everything together, we create a `diagloue\_mgmt\_system()` function that contains the logic of how the different layers would interact with each other. This will be the function that we'll call to initiate the chatbot. Below are some of the queries tested with the dialogue management system.



**Challenges Faced**

The fashion dataset has issues with data quality and consistency, including missing values, inaccurate entries, and mislabeled products. It also had inconsistent price and rating formats and inadequate descriptions. Preprocessing and data cleaning procedures were put into place to deal with missing numbers, fix discrepancies, and standardise formats. employed data validation methods to guarantee the dataset's accuracy and integrity.

Because fashion choices are subjective and user requirements vary, it was challenging to effectively capture the user's fashion preferences.The intent layer was not functioning as expected, and the conversation outcomes were erratic. For better outcomes, the prompts were improved, implemented methods for user feedback to gradually enhance and strengthen the chatbot's intelligence of user preferences.

When receiving inputs, the model became confused and kept asking the same questions without making any recommendations. simplified the questions by giving thorough guidance and including the thought process.

Since creating indexes using Chromadb required a lot of effort and resources, it was decided to forego it.

It was challenging to make sure the chatbot offered precise and pertinent suggestions depending on customer needs, particularly when dealing with a wide range of products. analysed user preferences and paired them with appropriate items using sophisticated machine learning algorithms.

**Future Scope & Recommendations**

**Enhanced Data Preprocessing:** The model performs better with more data. In order to create a strong chatbot system that can more accurately capture intent, combine data from several sources.

**Natural Language Processing (NLP) Advanced**: Use text embeddings (like FastText or BERT) to improve similarity matching. Use sentiment analysis to comprehend user reviews.

**Enhanced Algorithms for Recommendations:** Apply collaborative filtering according to user actions. Utilise content-based filtering to find comparable items. For strong suggestions, combine content-based and collaborative filtering.

**Interaction with Users and Feedback Loop:** Gather user input and utilise it to improve suggestions. Use a dialogue management system to make chatbots more interactive.

**Performance optimisation and scalability:** To handle massive datasets effectively, use batch processing. Use parallel computing to process data more quickly. Put caching in place for data that is accessed frequently.

**User Experience and Interface:** Make sure the chatbot has an easy-to-use UI. For increased engagement, incorporate visual components (such as product photos) into your recommendations.