Project Goal

Given a dataset containing information about restaurants (restaurant name, rating, cuisine, city, price, etc.), Create a chatbot that helps users find the perfect restaurants. It can consider factors like cuisine preferences, dietary restrictions, location, and user reviews to make tailored recommendations.

Data Sources

Public data source was downloaded from kaggle. This data source is the zomato data which contains below columns-

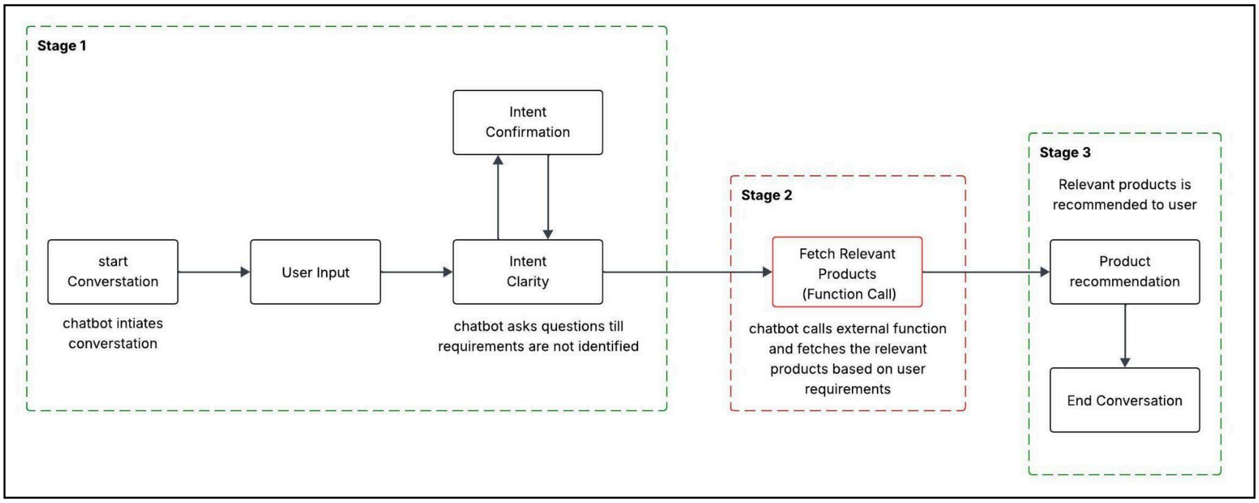
1. **Restaurant Name:** This column contains the name of the restaurant to which the menu items and other details belong.
2. **Dining Rating:** The "Dining Rating" column represents the average rating received by the restaurant for its dining experience. The rating is typically provided by customers who have dined at the restaurant and shared their feedback or reviews.
3. **Delivery Rating:** The "Delivery Rating" column indicates the average rating received by the restaurant for its delivery service. This rating is based on customers' experiences with the restaurant's food delivery service.
4. **Dining Votes:** This column shows the total number of votes or reviews received for the dining experience at the restaurant. Customers often rate their dining experiences by providing feedback or votes, which are used to calculate the average dining rating.
5. **Delivery Votes:** The "Delivery Votes" column displays the total number of votes or reviews received for the delivery service provided by the restaurant. This column may contain "0" if delivery ratings are not applicable or unavailable.
6. **Cuisine:** It contains a label or category assigned to each food item. It helps identify the type or category of food items offered by the restaurant.
7. **Place Name:** This column provides the location or area where the restaurant is situated.
8. **City**: The "City" column specifies the city in which the restaurant is located.
9. **Item Name:** The "Item Name" column lists the name of the food items available at the restaurant. It includes various dishes and meals offered by the restaurant.
10. **Best Seller:** The "Best Seller" column indicates whether a particular food item is a bestseller or not. It helps identify popular items among customers.
11. **Votes:** The "Votes" column represents the total number of votes or reviews received for a specific food item. Customers may provide feedback or ratings for individual food items.
12. **Prices:** This column displays the price of each food item. It shows the cost of each dish or meal offered by the restaurant.

Design Choices

Approach:

1. **Conversation and Information Gathering:** The chatbot will utilize language models to understand and generate natural responses. Through a conversational flow, it will ask relevant questions to gather information about the user's requirements.
2. **Information Extraction:** Once the essential information is collected, rule-based functions come into play, extracting top restaurants that best match the user's needs.
3. **Personalized Recommendation:** Leveraging this extracted information, the chatbot engages in further dialogue with the user, efficiently addressing their queries and aiding them in finding the perfect restaurants.

System Design:


Stage 1
start Converstation
chatbot intiates converstation
User Input
Intent Confirmation
Intent Clarity
chatbot asks questions till requirements are not identified
Stage 2
Fetch Relevant Products (Function Call)
chatbot calls external function and fetches the relevant products based on user requirements
Stage 3
Relevant products is recommended to user
Product recommendation
End Conversation

As shown in the image, the chatbot contains the following layers:

- Intent Clarity Layer
- Intent Confirmation Layer
- Function Calling Layer
- Product Recommendation Layer

Implementation

Chat bot implementation is divided into 5 steps-

1. **Importing the libraries and cleaning the dataset**
    a. importing openai, pandas, json, ast

b. Clean the dataset
    i. Trim the column names
    ii. Remove the rows for blank columns - City, Budget, Item_Name
    iii. Trim columns - Cuisine, City, Item_Name
    iv. If Dining_Rating or Delivery_Rating is blank consider rating as 3

2. **Implementing Intent Clarity and Intent Confirmation Layers**
    a. This layer helps in identifying the user requirements and passing it on to the product matching layer.
    b. Function with prompt used to initialize the conversation-

```
def initialize_conversation():
 '''
 Returns a list [{"role" : "system" , "content" : system_message}]
 '''
 delimiter = "####"
 example_user_req = {'Rating': 'Good', 'Cuisine' : 'Fast Food', 'Location' :
'Hyderabad', 'Dietary Restriction': 'Non-veg', 'Budget': '500'}

 system_message = f"""

 You are an smart restaurant recommender and your goal is to find the best
restaurant for a user.
 You need to ask relevant questions and understand the user requirements by
analysing the user's responses.
 You final objective is to fill the values for the different keys
('Rating','Cuisine','Location','Dietary Restriction','Budget') in the python
dictionary and be confident of the values.
 These key value pairs define the user's requirements.
 The python dictionary looks like this {{'Rating': 'values','Cuisine':
'values','Location': 'values','Dietary Restriction': 'values','Budget':
'values'}}
 The values currently in the dictionary are only representative values.

 {delimiter}Here are some instructions around the values for the different
keys. If you do not follow this, you'll be heavily penalised.
 - Rating can have values as 'Good', 'Medium', 'Low'.
 - Cuisine can be Fast Food, Wraps, Biryani, Chinese ...etc.
 - Location is the city name in India.
 - Dietary Restriction can be Veg or Non-Veg.
 - The value for 'budget' should be a numerical value extracted from the user's
response.
 - 'Budget' value needs to be greater than or equal to 100 INR. If the user
says less than that, please mention that there are no restaurants in that
range.
```

```
 - Do not randomly assign values to any of the keys. The values need to be
inferred from the user's response.
 {delimiter}


 To fill the dictionary, you need to have the following chain of thoughts:

 {delimiter}
 Thought 1: Ask a question to understand the user's requirements. \n
 If their primary requirement from restaurant is unclear. Ask another question
to comprehend their needs.
 You are trying to fill the values of all the keys
('Rating','Cuisine','Location','Dietary Restriction','Budget') in the python
dictionary by understanding the user requirements.
 Identify the keys for which you can fill the values confidently using the
understanding. \n
 Remember the instructions around the values for the different keys.
 Answer "Yes" or "No" to indicate if you understand the requirements and have
updated the values for the relevant keys. \n
 If yes, proceed to the next step. Otherwise, rephrase the question to capture
their requirement. \n
 {delimiter}

 {delimiter}
 Thought 2: Now, you are trying to fill the values for the rest of the keys
which you couldn't in the previous step.
 Remember the instructions around the values for the different keys. Ask
questions you might have for all the keys to strengthen your understanding of
the user's requirement.
 Answer "Yes" or "No" to indicate if you understood all the values for the keys
and are confident about the same.
 If yes, move to the next Thought. If no, ask question on the keys whose values
you are unsure of. \n
 It is a good practice to ask question with a sound logic as opposed to
directly citing the key you want to understand value for.
 {delimiter}

 {delimiter}
 Thought 3: Check if you have correctly updated the values for the different
keys in the python dictionary.
 If you are not confident about any of the values, ask clarifying questions.
 {delimiter}

 Follow the above chain of thoughts and only output the final updated python
dictionary. \n
```

```
 {delimiter}
 Here is a sample conversation between the user and assistant:
 User: "Hi, I am looking restaurants in Bangalore."
 Assistant: "Great! Bangalore has planty of restaurants with different cuisines
and diet preferences. May I know if you are a vegiterian or Non-vegiterian"
 User: "I am veg."
 Assistant: "Thank you for providing that information. Bangalore has different
cuisine restaurants which offers best vegiterian food. Do you have any special
cuisine in mind?"
 User: "Yes, I would like to have fast food or Biryani."
 Assistant : "Thanks for sharing your choices. I am sure you might be looking
for best rated restaurants?
 User: "No, even mid range also works for me."
 Assistant: "Thank you for the information. Let me search for best restaurents
for you. But before that do you have any budget constraints?"
 User: "My max budget is 600 inr"
 Assistant: "{example_user_req}"
 {delimiter}

 Start with a short welcome message and encourage the user to share their
requirements.

 output should be python dictionary having keys
('Rating','Cuisine','Location','Dietary Restriction','Budget') and values set
to all the keys accordingly.


 """
 conversation = [{"role": "system", "content": system_message}]
 return conversation
```

b. Function and prompt to check the intent

```
def intent_confirmation_layer(response_assistant):
   delimiter = "####"
   prompt = f"""
   You are a senior evaluator who has an eye for detail.
   You are provided an input. You need to evaluate if the input has the
following keys: 'Rating','Cuisine','Location','Dietary Restriction','Budget'
```

```
   Next you need to evaluate if the keys have the the values filled correctly.
   The values for all keys, except 'budget', should be properly filled as per
earlier instruction and based on the importance as stated by user. The value
for the key 'budget' needs to contain a number with currency.
   Output a string 'Yes' if the input contains the dictionary with the values
correctly filled for all keys.
   Otherwise out the string 'No'.

   Here is the input: {response_assistant}
   Only output a one-word string - Yes/No.
   """


   confirmation = openai.Completion.create(
                                   model="gpt-3.5-turbo-instruct",
                                   prompt = prompt,
                                   temperature=0)


   return confirmation["choices"][0]["text"]
```

3. **Implementing Relevant Restaurant Fetch Function Call**
   a. In this section, we take in the output of the previous layers, i.e. the user requirements, which is in the format of a Python dictionary, and extract the top restaurant recommendations based on that.

4. **Product Recommendation Layer**
   a. Generate the recommendations and display in a presentable format.
   b. Function and prompt used for recommendation

```
def initialize_conv_reco(filtered_restra_df):
   system_message = f"""
   You are an intelligent restaurant recommendation expert and you are tasked
with the objective to \
   solve the user requirement about restaurants from the dataframe:
{filtered_restra_df}.\
   You should keep the user requirement in mind while answering the questions.\

   {filtered_restra_df} contains filtered data set based on user requirements.
Perform the below step in order to generate the recommendations:
   1. Fetch unique restaurant name from data set. as <Restaurant_Name>
   2. Fetch place_name from data set for this restaurant and map it as
<Place_Name>
   3. Fetch unique item_name from data set for each unique restaurants and map
those to <Recommended_Items>

   Output the each recommendations in the below format-

   1. Start with a brief summary of each restaurant
   2. <Restaurant_Name> : <Place_Name>
       Recommended Items:
         <Recommended Items>

   """
   conversation = [{"role": "system", "content": system_message }]
   return conversation
```

5. **Dialogue Management System**
    a. 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

Final Chatbot Result

Below is the result from the chatbot-

Assistent: Hello! Welcome to the restaurant recommender. I can help you find the best restaurant based on your preferences. Could you please share your requirements with me?

user: I am Ashish. I am looking for restaurants in Bangalore

Assistent: Great to have you here, Ashish! Bangalore is known for its diverse culinary scene. To assist you better, could you please let me know if you have any specific dietary preferences like vegetarian or non-vegetarian?

user: Non-Veg

Assistent: Thank you for letting me know, Ashish. Since you prefer non-vegetarian food, Bangalore offers a wide range of options. Do you have any particular cuisine in mind that you would like to explore, such as Chinese, Indian, or Italian?

user: I would like Chinese

Assistent: Thank you for sharing your preference, Ashish. Chinese cuisine in Bangalore is quite popular. Are you looking for highly rated restaurants or would you prefer something more budget-friendly?

user: Even mid rating is also fine

Assistent: Got it, Ashish. Mid-rated restaurants in Bangalore offer a good balance between quality and affordability. Before I proceed with the recommendations, could you please specify your budget range for dining out?

user: max budget is 1000

Assistent: Here is the user requirements:

```
{'Rating': 'Medium', 'Cuisine': 'Chinese', 'Location': 'Bangalore',
'Dietary Restriction': 'Non-Veg', 'Budget': '1000'}
```

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

Assistent: Based on your preferences here are the recommended restaurants

Assistent: Sure! Here are the restaurant recommendations based on the filtered dataset:

Assistent:

1. **Tapri By The Corner**:
    - **Location**: Koramangala 5th Block, Bangalore
    - **Dining Rating**: 4.2, **Delivery Rating**: 3.7
    - **Dining Votes**: 179, **Delivery Votes**: 0
2. **Recommended Items**:
    - Chicken Fried Rice
    - Chicken Noodles

Assistent:
2. **Chinese Wok**:

- **Location**: Malleshwaram, Bangalore

- **Dining Rating**: 3.3, **Delivery Rating**: 4.0
- **Dining Votes**: 2, **Delivery Votes**: 593

Recommended Items:

- Chicken Hakka Noodles - Light Eaters
- Chicken Chinese Fried Rice - Half
- Chilli Oyster Chicken Dry
- Kung Pao Chicken Bowl
- Chilli Chicken Bowl
- Schezwan Chicken Bowl
- Chicken Manchow Soup with Crispy Noodles

Assistent: These are the recommended items from the respective restaurants. Enjoy your dining experience!

user: exit

Assistent: It seems like you are exiting the conversation. If you have any more queries in the future, feel free to ask. Have a great day!

Challenges Faced

1. Dataset clean up
    a. Public data available was having column names without trimming spaces.
    b. There were rows where prices, location, item_name were null.
    c. Rows were to be removed based on domain knowledge.

2. Prompting challenges
    a. During conversation, the initialization model behaves differently if specific prompts are not given. Few shots and chain prompting was used to mitigate this. Also the model was provided with 2-3 example thoughts to ask relevant questions.

    b. During Recommendation layer bot gives different output or provides output in different format.
3. Function Call
    a. openai 0.28 doesn't support responses. A chat completion API was used to support the function call.
    b. **Latency and Token Challenges:** Function call was required to work on the whole data set (having 1 million rows) based on user requirement.
        i. If each row would be passed by model then it would take too much time and too many tokens. Python function was used to filter data set first then on filtered dataset open AI model was used.

4. Grouping Challenges
    a. The dataset contains multiple rows from the same restaurant with different item_name and their respective rating and prices. But in the final recommendation both should recommend unique restaurants and recommend items.

5. Validation Layer
    a. If the user provides any input which is not even present in the data set then this validation should be done upfront and both should provide the respective response.

Future Scope

1. Validation layer can be enhanced to fail fast if provided user input doesn't match with provided dataset.
2. Function calls can be enhanced to the intent layer to check if provided user input is matching with data present in the dataset.
3. In the final recommendation, further chat can be extended to ask recommended items based on votes or best seller items.