**Project Report**

* **Hotel Finders AI**

An AI Chatbot application that helps you book Hotel in 4 cities - Sarajevo, Zagreb, Belgrade, Ljubljana located in Europe. The User can provide some details such as the city , Number of rooms required, Budget in BAM currency and Hotel Star Ratings, and our chatbot will recommend the best suitable Hotel for the user from the available options extracted from a Kaggle dataset..

* **Data Link**-> <https://www.kaggle.com/datasets/djusdjus/hotel-rooms-prices-dataset>
* **Dataset Description**

We use Hotel Accomodation Prices dataset available on Kaggle.

This dataset contains 120 observations and 9 variables.

The variables are:

* ID
* Hotel name
* Price(BAM): Price for one night In currency of Bosnia and Herzegovina
* Hotel star rating: Number of stars (1-5)
* Distance: The distance of hotel from city centre in meters
* Customer rating: Rating on booking
* Rooms: Number of rooms
* Squares: m^2
* City: City in which hotel is

Majorly we have made use of variables : Hotel name,Price(BAM),Hotel star rating,Distance and City , in our application.

Project Flow and System Design

1. **Initialize\_conversation():**

The purpose of the function is to make the gpt model understand its role, its responsibilities as to what questions to ask and provide a context.

Also, the variables are introduced here and what values are expected to fill the keys(variables: Hotel name, Price(BAM), Hotel star rating, Distance and City)

There are 3 Thoughts as part of Chain of thoughts provided as prompt to the GPT Model as part of context.

After that we go for Few Short prompting where in we show a sample conversation between thee User and Assistant.

We create a variable debug\_conversation and store the call of our function initialize\_conversation in it.

1. **get\_chat\_model\_completions()**

This function takes in argument – messages and returns the message of assistant (the gpt bot)

We make use of ChatCompletionAPI from Open AI and specify the model as ‘gpt-3.5-turbo’ and temperature as 0 to avoid randomness in response, also the max tokens.

We then pass our debug\_conversation as message in this function which returns us the expected response:

*Assistant: "Hello! Welcome to our Hotel recommendation service. How can I assist you today?"*

Next, we check our function using a custom input :

*"Hi, I am Taran. I am looking for a hotel in Sarajevo."*

To which the bot replies

*That's wonderful, Taran! Sarajevo is a beautiful city. To help you find the perfect hotel, I'll need a bit more information. Do you wish to stay close to the city center or a bit further away?*

The response is as expected ,so we move ahead.

1. **moderation\_check(user\_input)**

It is a crucial part of a chatbot application so that user cannot misuse the chatbot application for any unethical expectations and so that the chatbot can strictly focus on its agenda I.e Hotel Recommendation

Here we make use of Moderation API by OpenAI and make sure that the prompt given as input by the user is Flagged or Not Flagged.

4. **intent\_confirmation\_layer**

This function takes a text response as input and evaluates whether the response contains a dictionary with specific keys ("City," "Number of Rooms," "Distance from city center," "Hotel Star Ratings," "Budget in BAM") and whether these keys have the values as expected. The function returns a one-word string, either "Yes" if the input meets the criteria or "No" if it doesn't.

We make sure that we can fill in all the values for our keys.

**5. dictionary\_present(response)**

The dictionary\_present function takes a text response as input and extracts a Python dictionary from the input text. The expected format of the dictionary matches our predefined dictionary user\_req. The function's task is to locate this dictionary within the input text and return it as a string.

**6. hotel\_map\_layer(df)**

The hotel\_map\_layer function takes a DataFrame df as input and processes it to create a new column named "Hotels" in the DataFrame. This new column contains dictionaries for each row, where each dictionary represents information about a hotel.

* 1. It defines a list of keys called keys representing the attributes of a hotel.
  2. It initializes an empty list called hotels\_list to store the dictionaries.
  3. It iterates through each row of the input DataFrame df.
  4. For each row, it creates a dictionary named hotel\_req with the following attributes:
  5. "City": It sets the city to the value from the "City" column if it matches one of the specified cities (Sarajevo, Zagreb, Belgrade, Ljubljana); otherwise, it sets it to None.
  6. "Number of Rooms": It retrieves the number of rooms from the "Rooms" column.
  7. "Distance from city center": It determines whether the distance from the city center is "high" or "low" based on a predefined threshold (2500 units).
  8. "Hotel Star Ratings": It takes the hotel star rating from the "Hotel star rating" column.
  9. "Budget in BAM": It retrieves the budget in BAM (Bosnia and Herzegovina Convertible Mark) from the "Price(BAM)" column.
  10. It appends each hotel\_req dictionary to the hotels\_list.
  11. Finally, it adds the "Hotels" column to the original DataFrame df, where each row contains the corresponding hotel dictionary.

**7. compare\_hotels\_with\_user**

The compare\_hotels\_with\_user function takes two parameters: user\_req\_string (a dictionary representing the user's hotel requirements) and the df . It filters the hotels in the DataFrame based on the user's requirements and returns the top 3 hotel recommendations that best match those requirements in JSON format.

* 1. It initializes a DataFrame df\_hotels by filtering hotels from the input DataFrame df based on the user's specified city requirement (user\_req\_string['city']).
  2. It further filters df\_hotels based on the number of rooms requirement (user\_req['number of rooms']).
  3. It continues to narrow down the list of hotels based on the budget requirement (user\_req['budget in bam']).
  4. The function then applies additional filters based on the distance from the city center and hotel star ratings specified in user\_req.
  5. It calculates a score for each hotel in the filtered list based on how well it matches the user's requirements.
  6. The function sorts the hotels in descending order of their scores.
  7. It selects the top 3 hotels with the highest scores.
  8. Finally, it formats the information of the top 3 recommended hotels, including the hotel name and details, into a JSON-formatted string and returns it.

**8. format\_recommendations\_to\_user(top\_hotels)**

The format\_recommendations\_to\_user function takes a JSON-formatted string of top hotel recommendations and formats it into a user-friendly message. It loads the JSON string into a Python list of dictionaries, iterates through the list, and creates a formatted message containing information about each recommended hotel, including its name, location, number of rooms, distance from the city center, hotel star ratings, and budget. The function then returns this formatted message as a string.

**9. dialogue\_system**

The dialogue\_system function serves as the core of our conversational system. It engages in a conversation with a user, captures their input, and processes it through a chat model. The conversation starts with an initial greeting and then iteratively captures user responses, checks for moderation issues, confirms intents, and extracts relevant information. Once the user provides the necessary information, such as hotel preferences, the function proceeds to fetch and recommend hotels based on the user's requirements. It then formats the hotel recommendations and presents them to the user. The function continues until the user decides to exit the conversation. This function effectively integrates various conversational components to provide a user-friendly and informative hotel recommendation system.