**Objectives** Travel Chat Bot: Travel chat bot is an application built using openAI API to interact with the users and fetch details for their travel plan like: Travel\_From, Travel\_To, Travel\_Mode, and Class.

**Data** A synthetic data containing 1000 records was created using chatGPT prompt to have above fields along with Price.( Please find the csv file )

The cities under scope of this application are:

* + 'Bangalore',
  + 'Goa',
  + 'Hyderabad',
  + 'Delhi',
  + 'Pune',
  + 'Mumbai',
  + 'Kolkata',
  + 'Chennai',
  + 'Jaipur',
  + 'Ahmedabad'

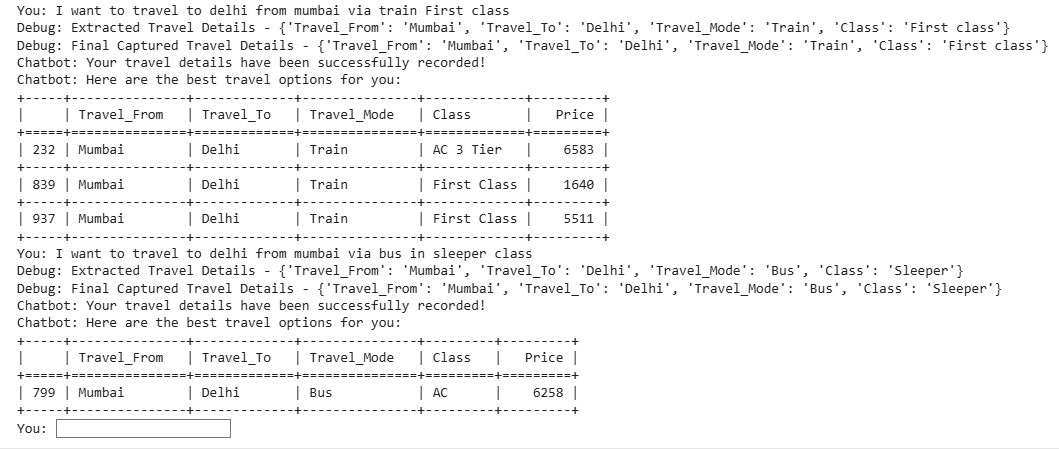
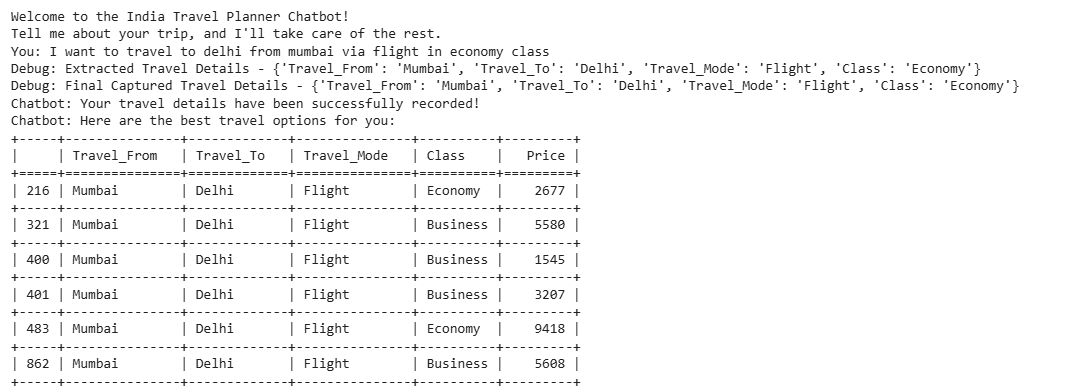
The Mode of Travel under scope of this application are:

* ‘Bus’,
* ’Flight’ &
* ‘Train’

Class applicable as per the Mode of Travel are:

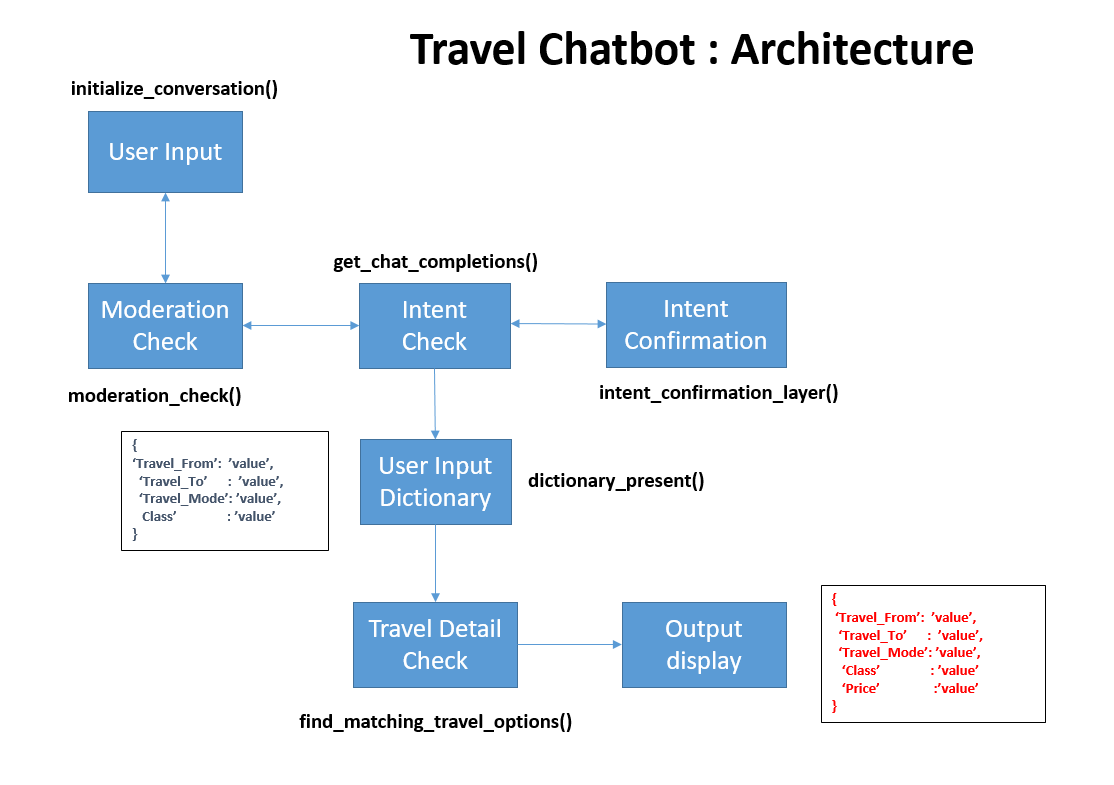
* Bus: AC, Non AC, Sleeper
* Train: First Class, AC2 Tier, AC3 Tier, Sleeper
* Flight: Business, Economy

A Travel chat bot enables the user to interact with openAI APIs to capture the useful details in the form of a python dictionary which is then matched against the data present in the csv file to display in tabular format the travel details to the users.

**Output**: Screenshot of the results is shown below 

Given below is the functional details for each function call.

1. **Architecture Design Implementation**



**2. Function Details**

**a. initialize\_conversation ()**

* Initializes the chatbot with a system message explaining how the user should provide travel details.
* Returns a conversation history list with the system message.

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

* Extracts structured travel details (Travel\_From, Travel\_To, Travel\_Mode, Class) from user input.
* Uses OpenAI's GPT to parse the response and return it in JSON format.
* If parsing fails, it returns an empty dictionary.

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

* Uses OpenAI’s Moderation API to check if the input contains inappropriate content.
* Returns "Flagged" if the content is inappropriate, otherwise returns "Not Flagged".

**d. get\_chat\_completions (conversation, json\_format=True)**

* Calls OpenAI's GPT model to generate a chatbot response.
* Uses a conversation history as input and returns a formatted response.

**e. intent\_confirmation\_layer (travel\_details, conversation)**

* Checks if all required details (Travel\_From, Travel\_To, Travel\_Mode, and Class) are present.
* If details are missing, prompts the user for clarification.
* Returns a confirmation result: "Yes" if all details are present, otherwise "No" with a clarification request.

**f. find\_matching\_travel\_options(travel\_details, csv\_file)**

* Reads a CSV file containing travel options.
* Matches rows that have the same Travel\_From, Travel\_To, and Travel\_Mode as extracted from the user input.
* If a match is found, returns a formatted table of travel options using the tabulate library.

**g. main()**

* Runs the chatbot loop:
  1. Welcomes the user and initializes the conversation.
  2. Waits for user input.
  3. If the user enters "exit" or "bye", the chatbot terminates.
  4. Checks if the input is flagged by the moderation system.
  5. Extracts structured travel details.
  6. If details are missing, asks for clarification.
  7. Once all details are present, records the travel information.
  8. Retrieves matching travel options from the CSV file and displays them in a tabular format.

**3. Flow of Events**

1. **User Input:** The user provides their travel details in natural language.
2. **Moderation Check:** Ensures that the input is appropriate.
3. **Extract Travel Details:** The chatbot processes the input and extracts details like source, destination, mode, and class.
4. **Confirm Missing Details:** If any details are missing, the chatbot asks for clarification.
5. **Find Matching Travel Options:** Once details are confirmed, the chatbot searches the CSV file for available travel options.
6. **Display Results:** The chatbot presents the best-matching travel options in a table.
7. **Repeat or Exit:** The user can enter another query or exit the chatbot.

**4. Challenges:** Faced a lot of challenge while implementing the intent confirmation block as I am NOT from a coding background. A lot of Iterations went in ensuring that the chat bot confirms the user details in a Json format. Also I faced challenge in ensuring that the user input is captured appropriately delineating the major Input field needed ( Travel From, Travel To, Travel Mode, Class)

**5. Learning –** Many times when the code becomes bulky, its natural to lose sight of bigger objective like we need a dictionary capturing the user input. I realized that I have not imported Json module in google collab file. Eventually I was able to create my first Chatbot!! ☺ Thank You IITB !!