### Report

### 1. Introduction

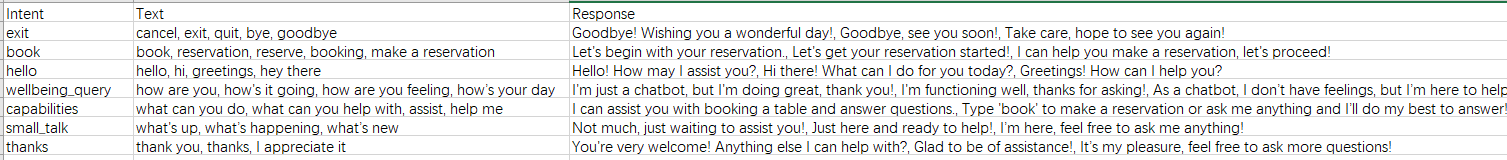
**This system is an interactive artificial intelligence (AI) system based on natural language processing (NLP), designed to provide intelligent reservation services for restaurants. Through conversations with users, the system is able to understand and respond to requests such as making reservations and inquiries, improving the restaurant’s customer service efficiency.**

### 2. System Architecture

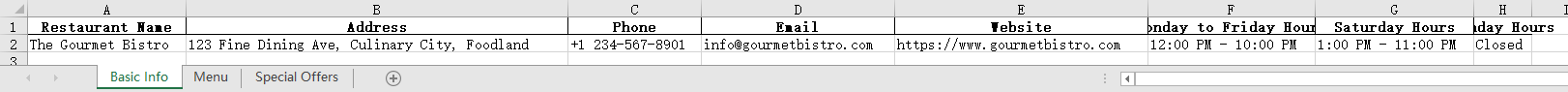
The system consists of five main modules: **chatbot.py**, **booking\_process.py**, **Intent\_Recognizer.py**, **qa\_system.py**, and **restaurant\_info.py**. Each module performs a specific task. **chatbot.py** is the core of the system, responsible for interacting with the user and coordinating the work of other modules. **Intent\_Recognizer.py** analyzes user input and identifies the user’s intent, while **booking\_process.py** manages restaurant reservation operations. **qa\_system.py** provides responses based on the knowledge base for information retrieval, and **restaurant\_info.py** provides answers related to basic restaurant information.

The system utilizes three datasets:

1. **Intents.xlsx**: This file contains the main intents recognized by the restaurant chatbot, along with common user dialogues and responses for casual conversations.
2. **restaurant\_info.xlsx**: This file stores predefined basic restaurant information, including the menu, address, contact details, operating hours, etc.
3. **CW1-Dataset.csv**: This dataset, provided by the university classroom, serves as an additional knowledge base for the chatbot.



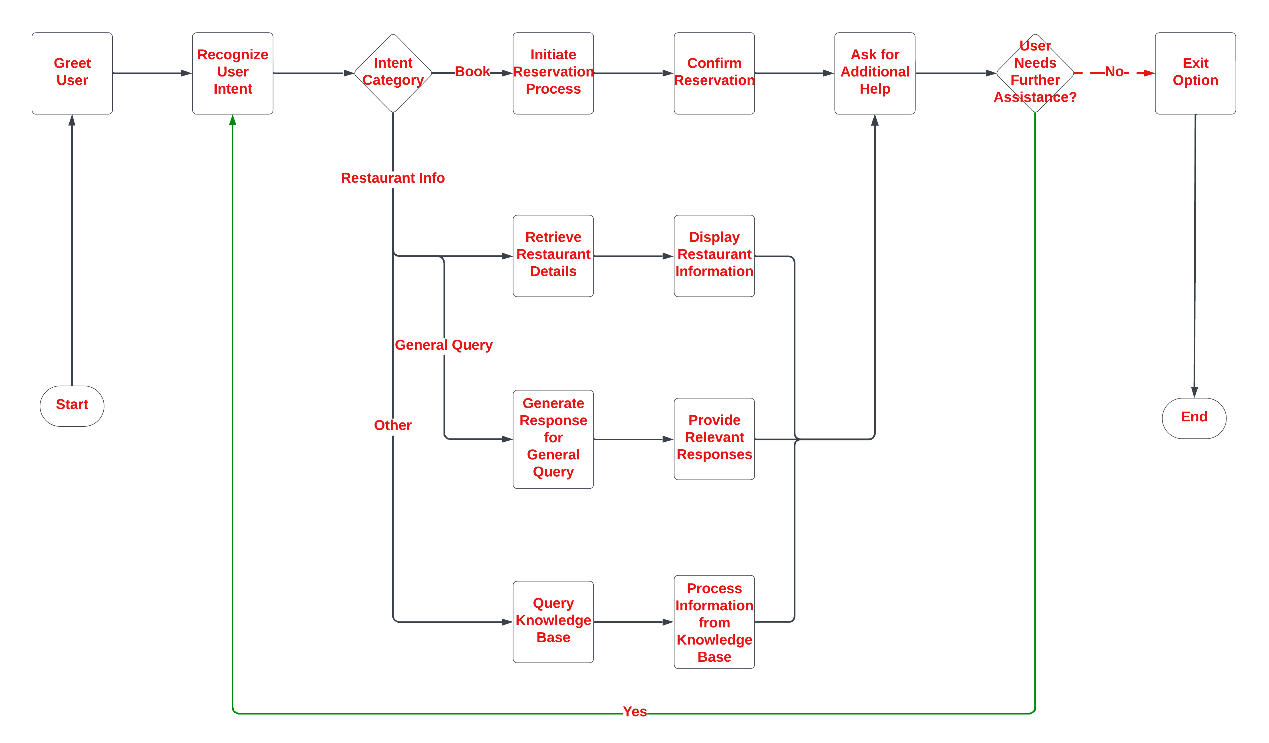
**Intents.xlsx**



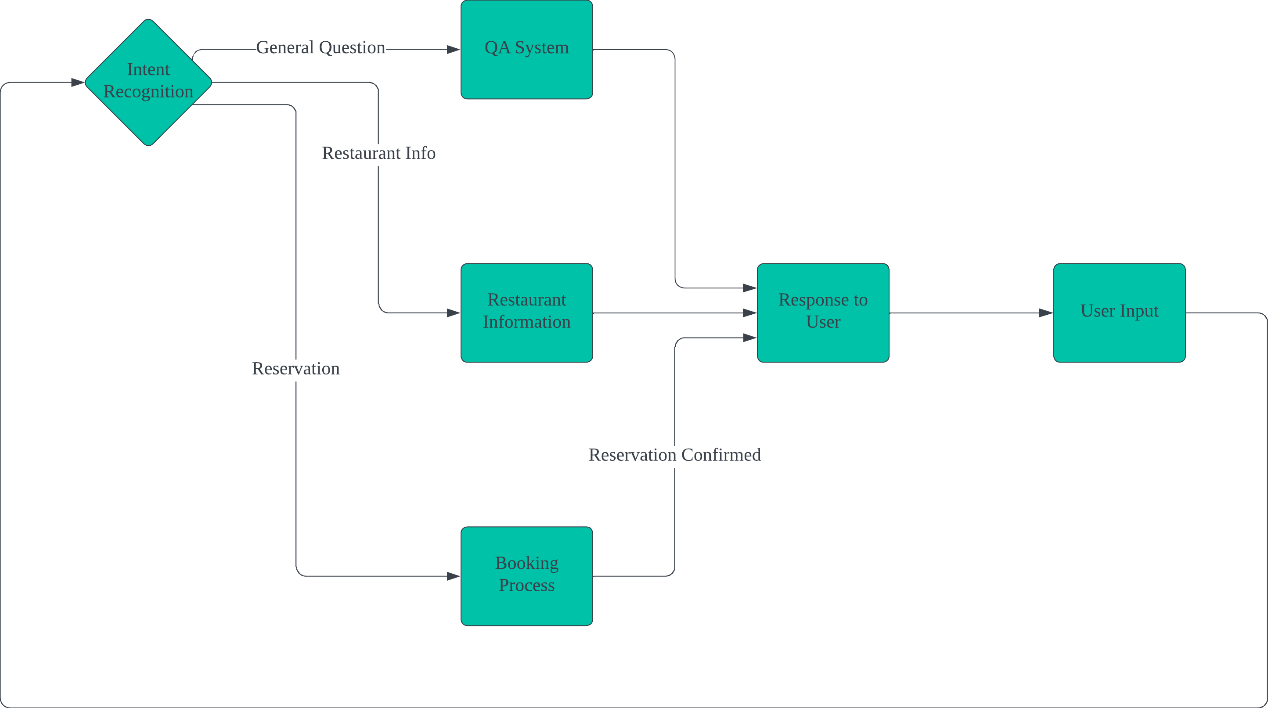
**restaurant\_info.xlsx**



### CW1-Dataset.csv



System Flowchart

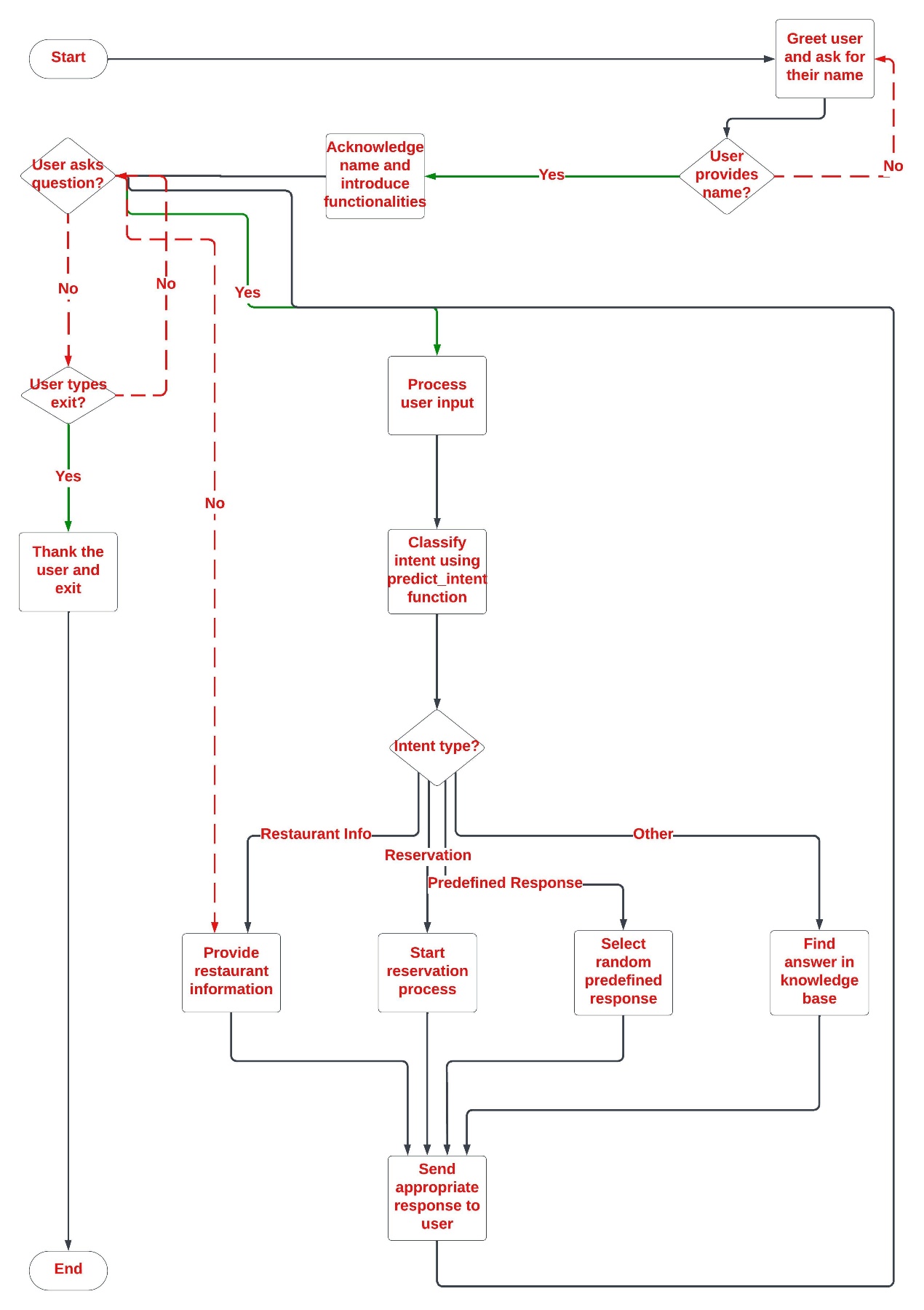
****

Data Flow Graph

### 2.1. Main Control Module (chatbot.py)

**chatbot.py** provides the interaction between the user and the chatbot, integrating intent recognition, restaurant information queries, reservation processing, and the question-answering system. Below is the analysis of each feature:

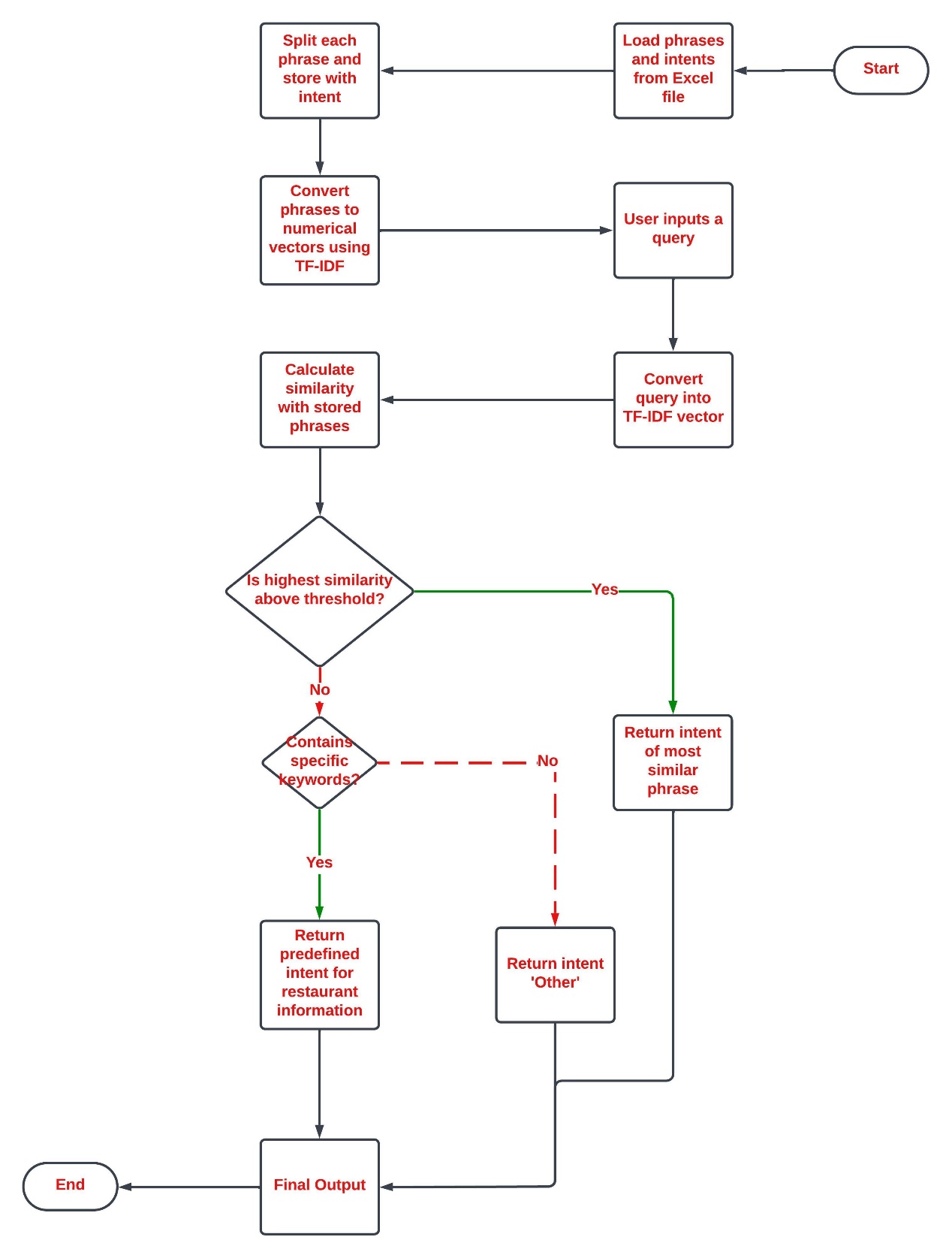
1. **User Identity Management and Guidance (greet\_user)** The variable user\_name is used to store the user's name, ensuring that the system can correctly address and display the user's name during the conversation, enhancing personalization and user-friendliness. When the system starts, the chatbot greets the user and guides them to provide their name via the greet\_user function. The chatbot also lists the actions the user can perform, such as making a reservation, inquiring about restaurant information, or engaging in casual conversation. This function manages user identity, provides basic guidance, and helps users understand the system’s functionalities and how to use them.
2. **Intent Recognition (predict\_intent)** The predict\_intent function is called to analyze the user input and determine their intent. For example, the user’s input could relate to restaurant reservations, inquiries about restaurant information, or casual conversation. Through the intent recognition system, the chatbot can effectively understand the user's needs and respond accordingly through different functional modules.
3. **Restaurant Reservation Process (start\_reservation\_process)** When the user's intent is to make a reservation, the chatbot calls the start\_reservation\_process function to initiate the reservation process. This function handles the user's restaurant reservation request, including seat confirmation, time scheduling, and other related tasks.
4. **Restaurant Information Query (handle\_restaurant\_query)** If the user asks for information about the restaurant (such as the address, operating hours, or menu), the chatbot invokes the handle\_restaurant\_query function to provide the relevant information.
5. **Question-Answer System (find\_answer)** When the chatbot cannot recognize the user's intent or when the user's query is unclear, it calls the find\_answer function. This function matches the user input with predefined question-answer pairs from the knowledge base using similarity scoring to find an appropriate response.
6. **Response Management (load\_responses)** The load\_responses function loads predefined responses for different intents from an external Excel file and randomly selects a suitable response. If the intent recognition system identifies an intent with predefined responses, the chatbot randomly selects one and outputs it.
7. **Conversation Management** The system maintains a continuous conversation loop with the user through the chatbot() function. This loop allows the chatbot to continuously listen for user input and process the input according to the identified intent. The user can exit the conversation at any time by entering "exit" or "quit."
8. **Error Handling** In the chatbot() function, if the user's input does not match any predefined intents, the chatbot returns a message saying, "I don't understand," prompting the user to rephrase their input. This ensures that the system can handle unclear or ambiguous inputs without crashing.



Main control analysis flow chart

### 2.2. Intent Recognition Module（Intent\_Recognizer.py）

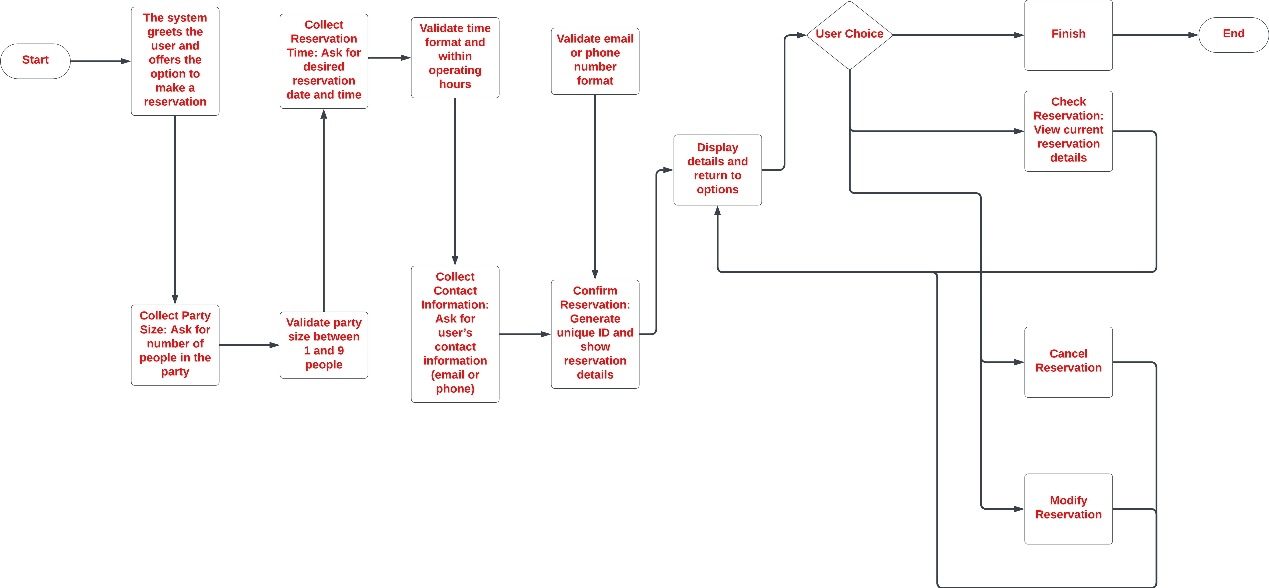
1. **Data Preparation**:  
   The dataset is loaded from an Excel file (Intents.xlsx), which contains different intents and their corresponding phrases. Each entry in the "Text" column contains multiple phrases, separated by commas, which are associated with a specific intent. All phrases and their corresponding intents are stored in two separate lists: all\_phrases and all\_intents.
2. **TF-IDF Vectorization**:  
   The TfidfVectorizer is used to convert all phrases into numerical vectors (TF-IDF features). The vectorizer is configured with an n-gram range of 1 to 3 words (including unigrams, bigrams, and trigrams), which helps capture contextual information of the phrases. The resulting TF-IDF matrix X represents the vectorized form of all the phrases.
3. **Intent Prediction**:  
   The predict\_intent() function takes the user's input and transforms it into a TF-IDF vector using the same vectorizer that was used for the phrases. It then calculates the cosine similarity between the user input vector and all predefined phrase vectors (matrix X).  
   If the highest similarity exceeds a certain threshold, the function returns the intent associated with the most similar phrase. If the similarity is below the threshold, it returns "Other", indicating that the intent could not be recognized.  
   Additionally, the function includes a keyword-based rule: if the user input contains keywords related to the restaurant (such as "menu", "location", "offers", etc.), it will return the intent "AskAboutRestaurant".



Intent recognition analysis diagram

### 2.3. Reservation Process Module（booking\_process.py）

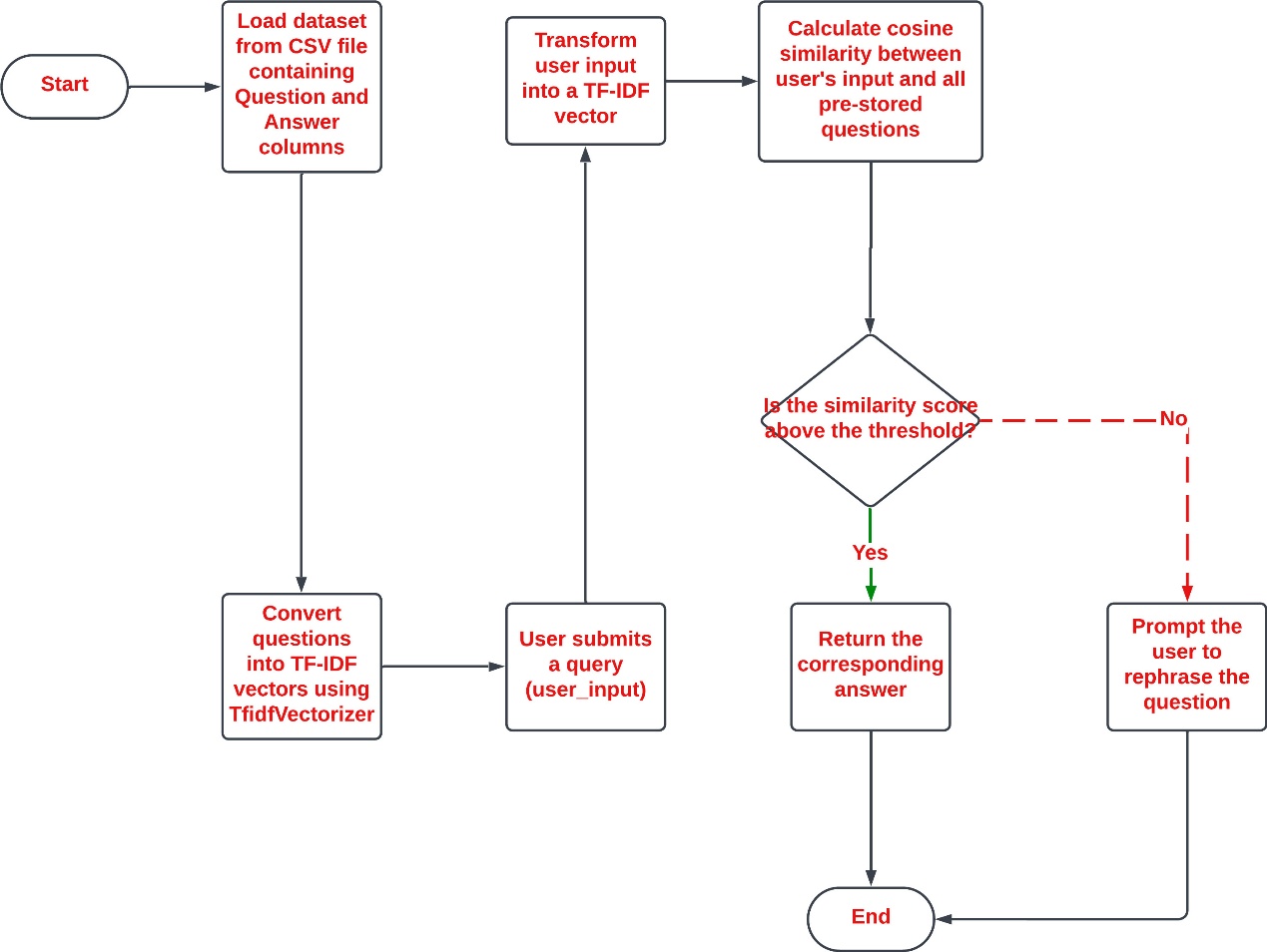
1. **Load Restaurant Information and Functions**:  
   A restaurant\_info module is introduced to load basic restaurant information, and the handle\_restaurant\_query function is used to handle queries related to the restaurant.
2. **Handling User-Provided Time (is\_valid\_reservation\_time)**:  
   The is\_valid\_reservation\_time function verifies whether the user-provided time is in the correct format (MM/DD/YYYY HH:MM AM/PM). It also checks whether the time falls within the restaurant's operating hours (based on the current day of the week). If the time is either in an incorrect format or outside of operating hours, the system returns an appropriate error message.
3. **Making a Reservation (make\_reservation)**:  
   After the user provides a valid time and other reservation details, the make\_reservation function generates a unique reservation ID and stores the relevant reservation information (such as name, party size, time, and contact details) in a reservations dictionary. Upon successful reservation, the system returns a confirmation message and displays the reservation details.
4. **Display Reservation Details (get\_reservation\_details)**:  
   When the user queries or confirms their reservation, the get\_reservation\_details function displays the reservation details, including the reservation ID, name, party size, time, status, and contact information.
5. **Modify Reservation (modify\_reservation)**:  
   The user can modify their reservation by selecting the appropriate option. The modifiable items include:
   1. Party size
   2. Date and time
   3. Contact information  
      The modified information is updated in the reservation record, and the system returns a confirmation message.
6. **Cancel Reservation (cancel\_reservation)**:  
   If the user chooses to cancel the reservation, the system changes the reservation status to "canceled" and displays a cancellation message.
7. **Check Reservation Status (check\_status)**:  
   The user can check the status of their reservation by entering the appropriate option. The system will return the current reservation status (e.g., confirmed, canceled).
8. **Interactive Input Process (prompt\_for\_input)**:  
   The system prompts the user to input various reservation details using the prompt\_for\_input function. The user inputs the following information step by step:
   1. Party size
   2. Reservation date and time
   3. Contact details (email/phone)  
      After each input, the system verifies the validity of the information to ensure it meets the requirements (e.g., time format, party size range).
9. **Main Process (start\_reservation\_process)**:  
   The start\_reservation\_process function serves as the main entry point for the reservation process. The user first provides party size, date and time, and contact details, all of which are used to create a reservation.  
   After completing the reservation, the user can choose to:
   1. Check reservation status
   2. Modify reservation
   3. Cancel reservation
   4. The process then concludes.



### Booking Flowchart

### 2.4 Knowledge Base Q&A Module (qa\_system.py)

1. **Data Loading and Preprocessing**:  
   The data is loaded from a CSV file, which contains two columns: "Question" (the question) and "Answer" (the corresponding answer).
2. **TF-IDF Vectorization**:  
   The TfidfVectorizer is used to convert all the questions (from the "Question" column) into TF-IDF vectors, enabling similarity calculations.
3. **Find the Most Matching Answer Function (find\_answer)**:  
   When a user inputs a question (user\_input), it is first converted into a TF-IDF vector.  
   The cosine similarity between the user's input and all the stored questions is then calculated.  
   The system identifies the question with the highest similarity to the user's input and returns the corresponding answer (if the similarity exceeds a predefined threshold).  
   If the similarity is below the threshold, the system returns the message: "I don't understand. Can you please rephrase or say something else?"



Knowledge base question and answer flow chart

### 2.5 Restaurant Information Response Module（restaurant\_info.py）

In the restaurant\_info.py, there are two functions: load\_restaurant\_info and handle\_restaurant\_query.

1. **load\_restaurant\_info Function**:  
   This function loads the basic restaurant information, menu, and special offers from an Excel file named restaurant\_info.xlsx.  
   It organizes the restaurant's basic details (such as the restaurant name, address, phone number, email, website, and business hours) along with the menu and special offer information into a dictionary and returns it.
2. **handle\_restaurant\_query Function**:  
   This function handles various user queries regarding the restaurant.  
   Based on the user's input, it determines the type of query and returns the corresponding restaurant information, such as:
   * **Restaurant Address**: Provides the address of the restaurant.
   * **Menu**: Lists the restaurant's menu items, including dish names, prices, and descriptions.
   * **Business Hours**: Provides the restaurant's operating hours.
   * **Special Offers**: Lists current specials or promotions.
   * **Contact Information**: Provides the restaurant's phone number or email.
   * **Restaurant Name**: Provides the name of the restaurant.

If the user's query does not match any predefined category, the chatbot will prompt the user with a list of information types they can inquire about, such as the address, menu, business hours, or special offers.

### 3. Conversational Design

#### 3.1. Prompt Design

Dialogue Tags and Key Phrases: My program includes clear prompts such as:

“Chatbot: Hello, liyu! How can I assist you today?”

“Chatbot: Here are some things I can help you with:

1. You can type 'book' to start the restaurant reservation process.
2. Ask about the restaurant, such as its name, opening hours, address, contact information, menu, or special offers.
3. Chat with me casually, for example, 'How are you?'
4. Ask me some general knowledge questions, like 'How much is 1 tablespoon of water?'”

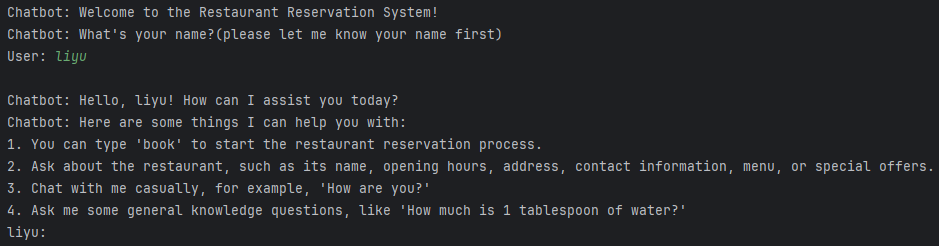
For prompts in the reservation process:

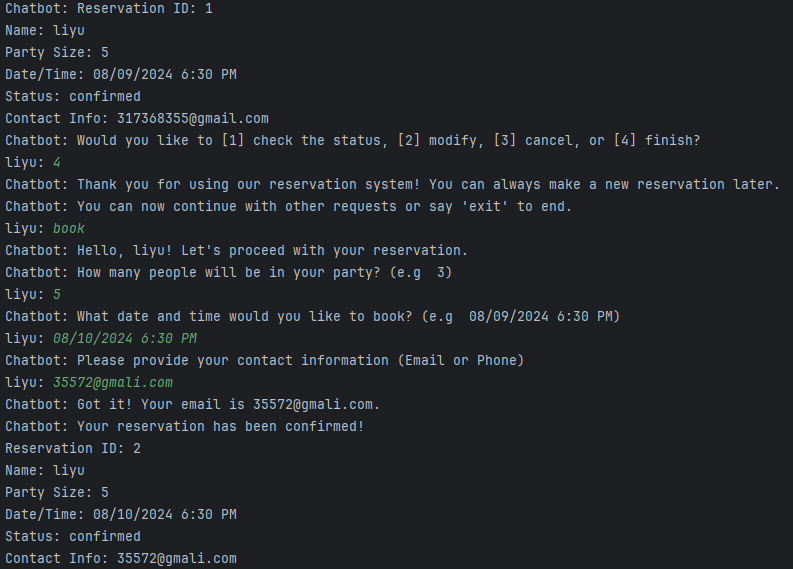
"Chatbot: How many people will be in your party?"

"Sorry, our operating hours today are from 12:00 PM to 10:00 PM."

Clarity and Conciseness: Prompts are simple and direct, asking the user for the required information (such as party size, date/time, contact details, etc.).

Supporting User/Task Structure: Each prompt is designed to guide users step-by-step through the process (e.g., asking for party size first, then asking for date/time), helping the user focus on one step at a time.







2. Discoverability

Contextual Assistance: The program provides guidance for the next step, such as:

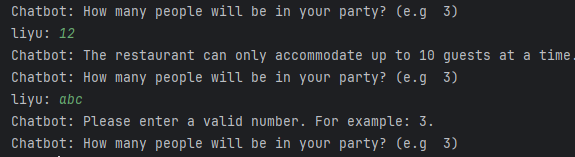
"Chatbot: Would you like to [1] check the status, [2] modify, [3] cancel, or [4] finish?" This helps users understand what they can do next after completing a step.

There are also prompts to help users enter valid information, such as:

"Please enter a valid number. For example: 3."

"Chatbot: The restaurant can only accommodate up to 10 guests at a time."

Discovering Features and Content: The program provides explanations of the different actions users can take (e.g., modify a reservation, check reservation status), which helps users discover available features.



3. Error Handling

Strategy: The program uses error handling to ensure valid user input. If a user enters something that the bot doesn’t understand, it will say:

"I don't understand. Can you please rephrase or say something else?"

It also validates ranges (1 <= size <= 10), and checks the format of email/phone numbers and date/time inputs.

If the user inputs something invalid, the system will provide an error message and prompt them to re-enter the information, such as:

"Chatbot: Please enter a valid number."

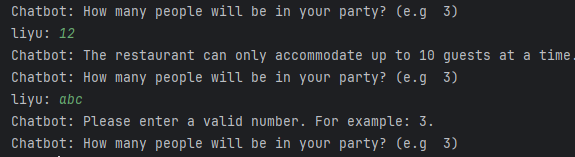
"Chatbot: Invalid time format. Please use the format 'MM/DD/YYYY HH:MM AM/PM.'"

Effectiveness: Error messages are clear and provide guidance, telling the user what went wrong and offering suggestions for correction, which helps reduce user confusion and frustration.









4. Personalization

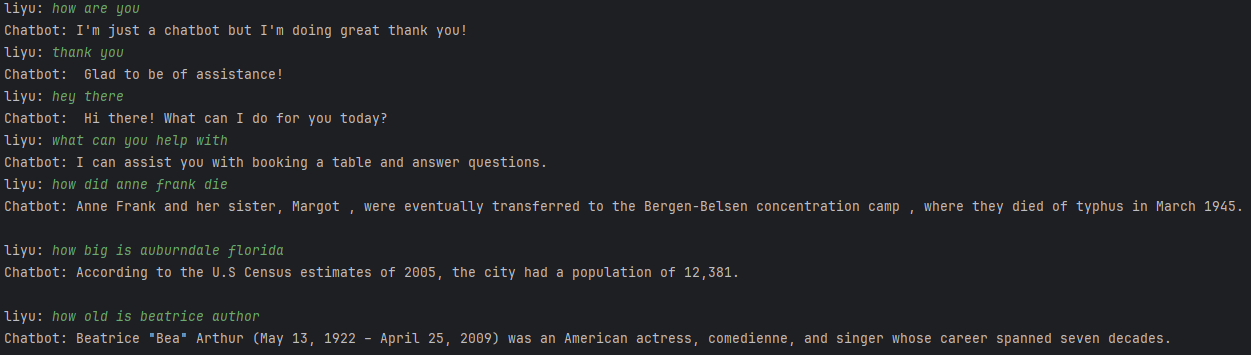
Personalized Experience: The program provides a personalized experience through small chats and knowledge-based interactions with users, such as:

"Chatbot: I'm just a chatbot but I'm doing great, thank you!"

"liyu: how old is Beatrice Arthur?"

"Chatbot: Beatrice 'Bea' Arthur (May 13, 1922 – April 25, 2009) was an American actress, comedienne, and singer whose career spanned seven decades."

These interactions help personalize the experience and make the conversation more engaging.



5. Confirmation

Strategy: After users submit their reservation information, the system provides confirmation, such as:

"Chatbot: Your reservation has been confirmed!"

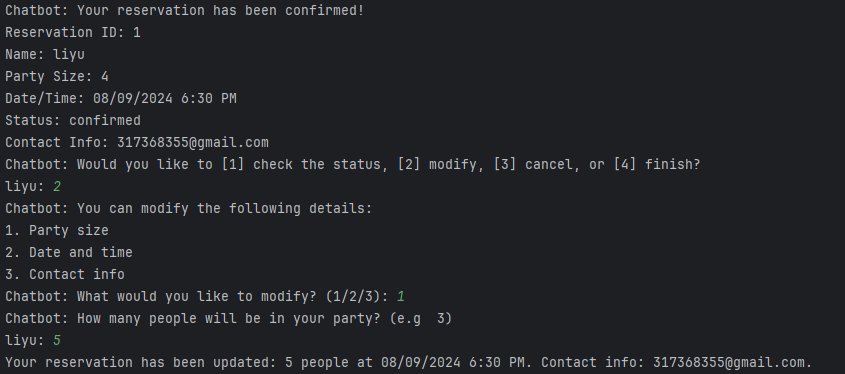
It also includes the reservation details to ensure users know their request has been successfully processed.

When users modify or cancel their reservation, the system provides confirmation messages like:

"Chatbot: Your reservation has been updated."

"Chatbot: Your reservation has been canceled."

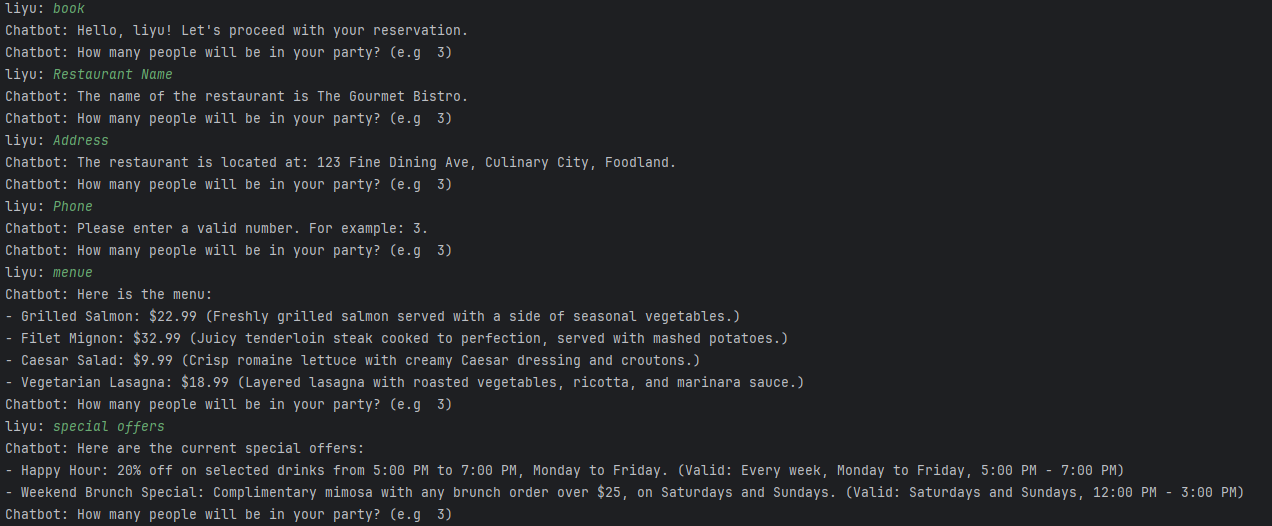
Effectiveness: This confirmation strategy is effective because it reassures users that their request has been successfully completed, reducing any possible misunderstandings.



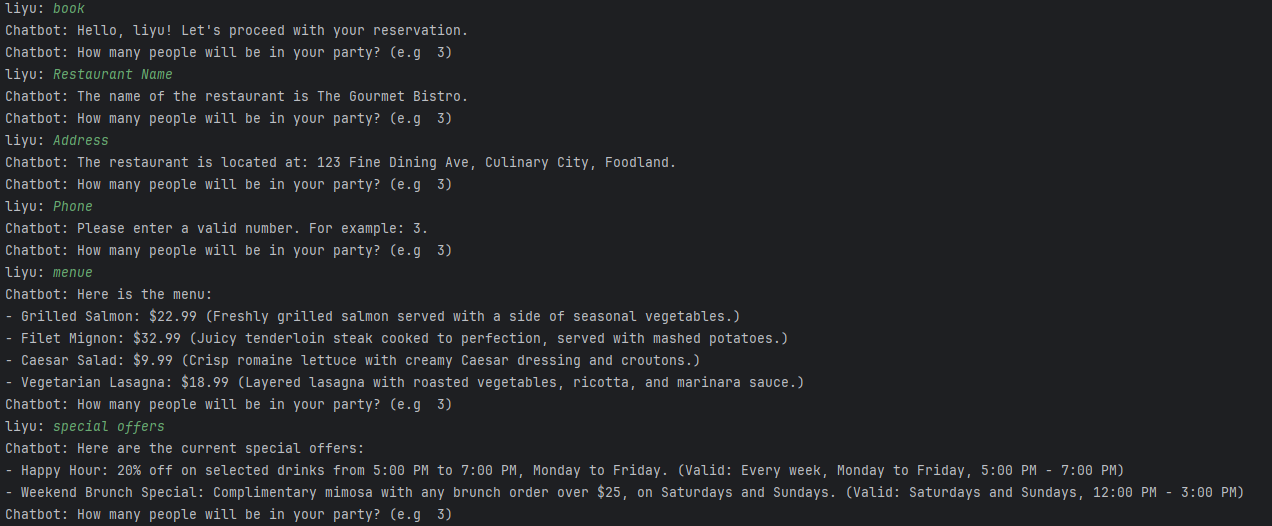
6. Context

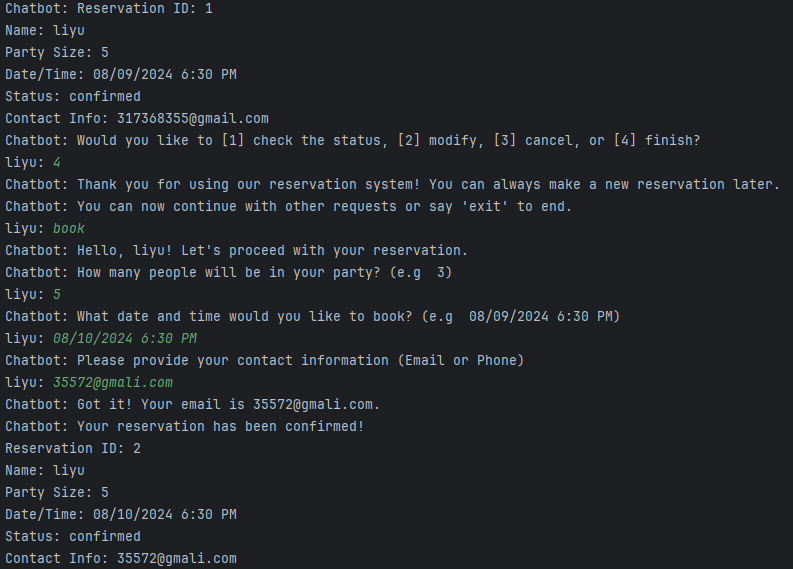
Context Tracking: The system effectively tracks context throughout the reservation process. For example, once users provide their party size, date/time, and contact information, the chatbot uses this information to continue processing the reservation. Additionally, users can ask the chatbot for information about the restaurant at any time during the process, and then continue with the reservation.

If users want to modify or cancel their reservation, the system can remember the previous reservation details (stored in reservations) and update or query based on this information.



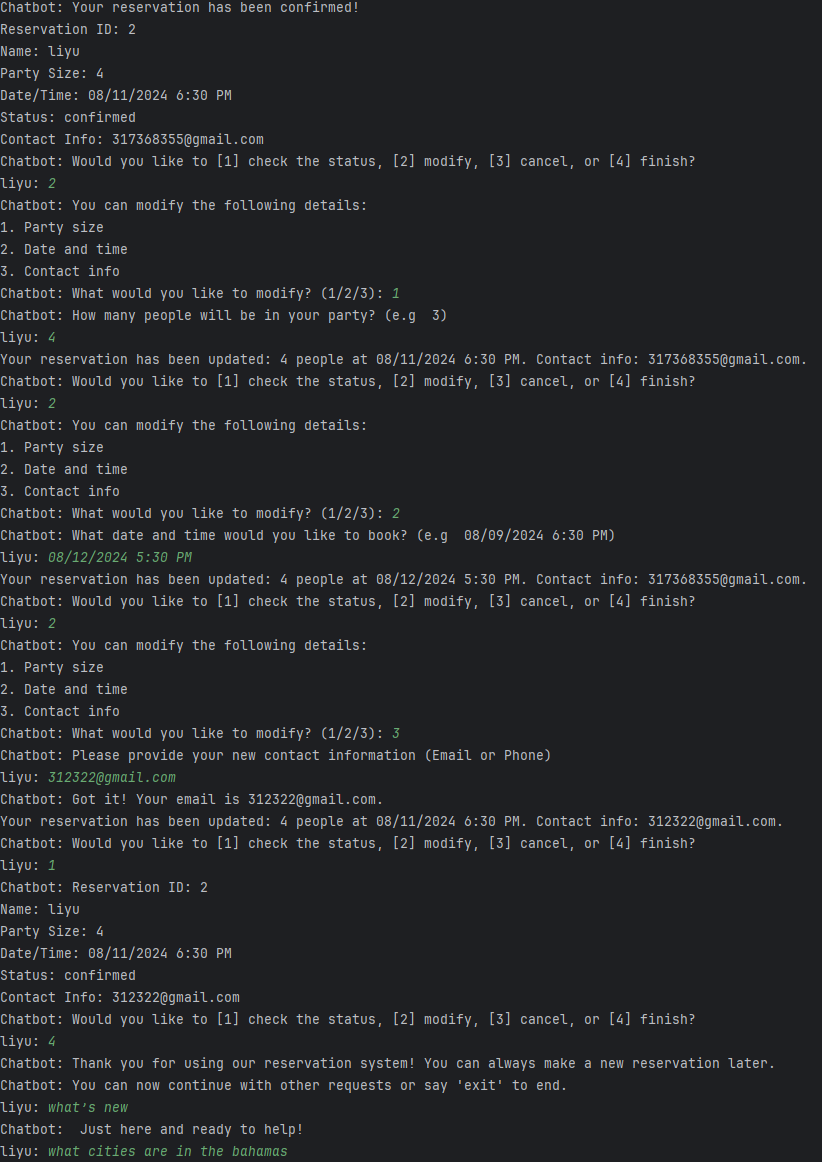
Effectiveness: Context management is handled well, as the chatbot provides relevant prompts based on the current interaction stage. However, the system could further extend context tracking (e.g., remembering user preferences or past queries).





### 4. Usability Testing Evaluation

I first ran a series of tests using each dataset from the spreadsheet to assess different aspects of the system. The final results showed that the chatbot met the expected outcomes, demonstrating high reliability and accuracy in user intent recognition and response generation. Below are some screenshots from the tests I conducted:



I then selected 5 test users and observed how participants interacted with the system, recording their actions, feedback, and difficulties. I also inquired about their opinions on the interface design, prompt information, and interaction flow, to verify whether the system met user needs: ensuring users could interact with the bot and successfully complete restaurant reservations, queries, and modifications. The usability of the system was assessed by testing whether users could quickly understand and use the system, especially when filling out information and completing reservations. Potential interaction issues were identified by observing the user process, noting any confusing or unclear prompts.

The following are the summarized issues identified:

1. **Some simple questions were not answered**: The knowledge base was too small, and the recognition of questions related to the knowledge base was not accurate enough.
2. **Unclear prompts**: Some users reported that the prompts were unclear, especially when input errors occurred. The system did not clearly inform users on how to correct errors (e.g., when the date/time format was incorrect).
3. **Unfriendly interface**: Some users were unsure about the next steps, particularly when modifying or canceling reservations. The design of the prompts and buttons needs to be more explicit.
4. **Form input issues**: Some users encountered difficulties entering phone numbers or emails, requiring better input format guidance.

### 5. Discussion

Through a series of self-tests and user feedback, I gained insights into the strengths and weaknesses of the project. The following is a discussion and reflection on the project:

#### Strengths:

1. **Core Functionality**: The chatbot successfully engages in simple conversations with users and performs basic tasks such as reserving a table, querying the menu, and checking operating hours, meeting the primary business requirements.
2. **Conversational AI**: By integrating conversational AI, users can interact naturally with the system, providing a convenient way to engage with restaurant services.
3. **Data-Driven**: The system dynamically retrieves restaurant information (such as the menu, special offers, etc.) from Excel files and provides accurate answers based on real-time data.
4. **Query Handling**: While the system handles basic queries well, the user experience declines when handling more complex requests, such as detailed reservation modifications.
5. **Scalability**: The system is designed to easily accommodate additional functional modules.

#### Next Steps for Improvement:

1. **Personalization**: Future improvements could include saving user preferences, recording historical reservations, and offering personalized suggestions to better meet user needs.
2. **Advanced NLP Capabilities**: Enhance the application of natural language processing techniques to improve the system’s ability to handle more complex queries by recognizing user intent more accurately.
3. **Expanding Use Cases**:
   1. **Multi-Restaurant Support**: The system could be designed to support multiple restaurant configurations, with each restaurant having its own menu, special offers, and operating hours. This would expand the system’s scope and increase its applicability across various dining establishments.