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 can understand and respond to reservation, inquiry, and other requests, enhancing the efficiency of customer service in restaurants.

1. System Architecture

The system is composed 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 users and coordinating the work of other modules. Intent\_Recognizer.py analyzes user input and recognizes the intent, booking\_process.py manages the restaurant reservation operations, qa\_system.py provides answers for information retrieval, and restaurant\_info.py handles basic restaurant-related information.

The system uses three datasets:

1. The Intents.xlsx file contains the primary intents recognized by the restaurant chatbot, common user dialogues, and some casual responses.
2. The restaurant\_info.xlsx file stores predefined restaurant information, including menus, addresses, contact details, operating hours, etc.
3. The CW1-Dataset.csv is an additional knowledge base provided by the university classroom, used for supplementing the chatbot's knowledge.

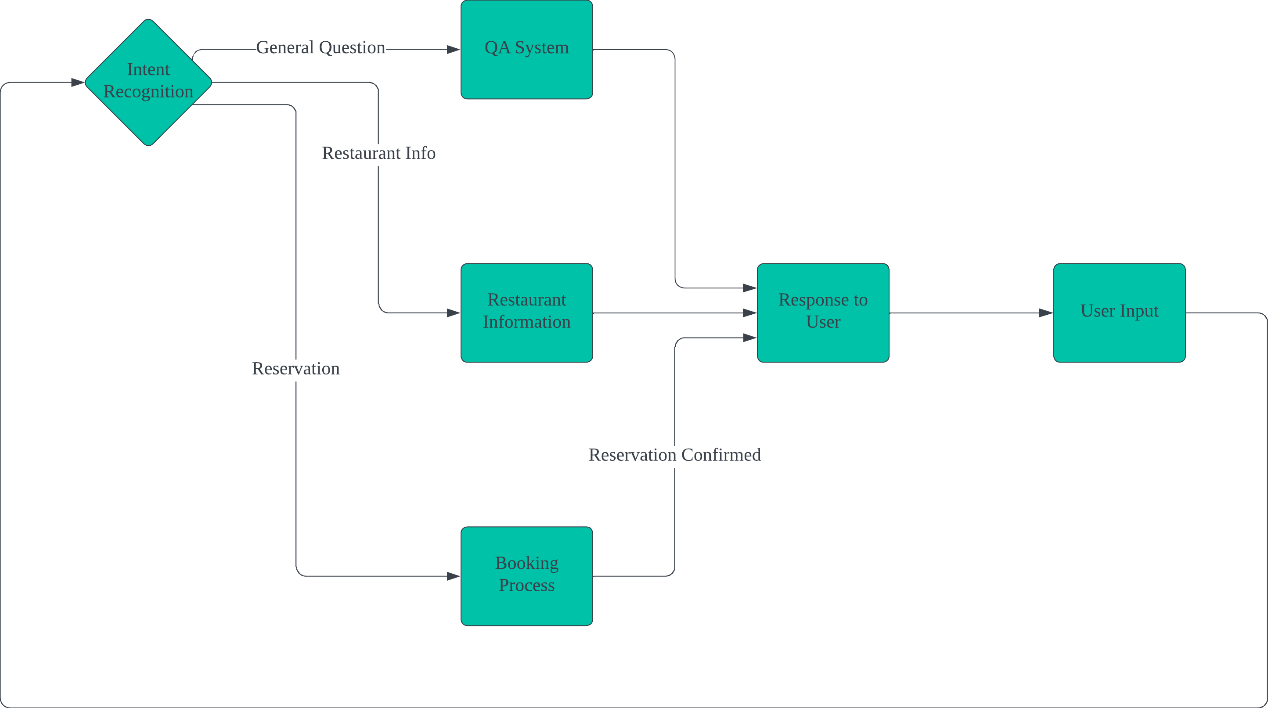
****

Figure 1: Brief System Flowchart

* 1. Main Control Module (chatbot.py)

The chatbot.py module provides interaction between the user and the chatbot, integrating intent recognition, restaurant information inquiry, reservation processing, and question-answering systems. Below is an analysis of each functionality:

It uses the time and datetime modules to obtain the current time. The predict\_intent and sentiment\_analysis functions in the Intent\_Recognizer module are used to identify the user's intent and analyze the sentiment of the input. The start\_reservation\_process and reservation\_history functions in the booking\_process module handle the user's restaurant reservation-related requests. The find\_answer function in the qa\_system module answers the user's open-ended questions. The load\_restaurant\_info and handle\_restaurant\_query functions in the restaurant\_info module process user queries related to restaurants.

The user's name is stored for personalized responses during the chat.

The load\_responses function loads responses corresponding to different intents from an Excel file and stores them in a dictionary for easy access when responding to user input.

The greet\_user function enables the chatbot to greet the user and ask for their name. Once the name is provided, the chatbot stores it and personalizes the greeting, while introducing available functional options.

The chatbot function is the main function of the chatbot, which handles the interactive flow with the user. It is divided into the following parts:

Users can input different commands, and the system will analyze the user's intent using the predict\_intent function and execute different operations based on the recognized intent.

If the user enters 'exit' or 'quit', the chatbot will exit.

If the user enters 'history', the chatbot will display the user's reservation history.

If the user's input relates to restaurants (e.g., restaurant name, address, menu, etc.), the chatbot will call the handle\_restaurant\_query function to provide relevant information.

If the user's intent is 'book', the chatbot will call the start\_reservation\_process function to begin the reservation process and handle the user's reservation request.

Users can query or change their name. If the user hasn't provided a name, the chatbot will prompt them to input one; if a name has already been provided, the chatbot will display the current name.

Users can change their name using the 'change' command. The chatbot will prompt the user to input a new name.

If the user inputs 'time', the chatbot will return the current date and time.

The chatbot will perform sentiment analysis on the user's input and provide an appropriate emotional response.

If the identified intent has a predefined response, the chatbot will randomly select one of the responses associated with that intent and reply to the user.

If the chatbot cannot identify the user's intent, it will call the find\_answer function to search for a suitable answer in the question-answering system. If no answer is found, the system will prompt the user to re-enter their query.

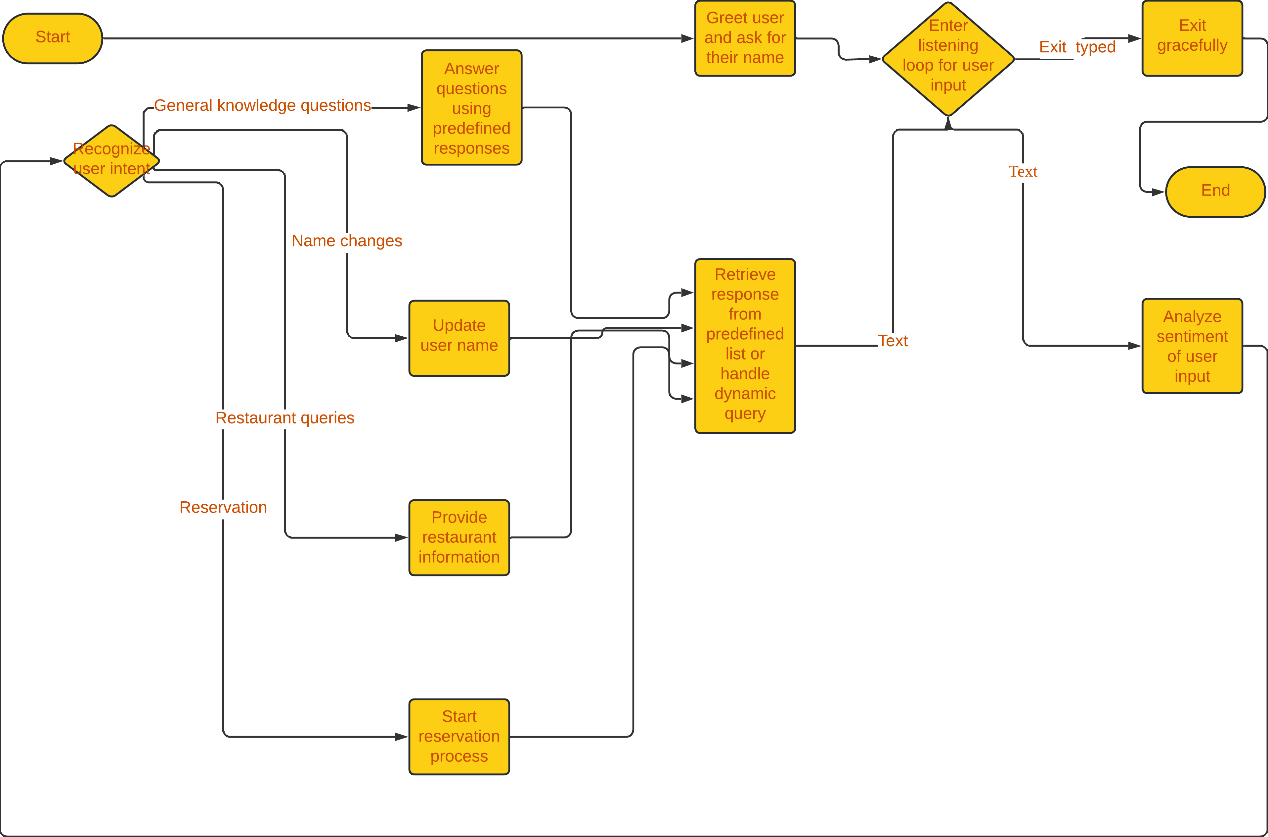


Figure 2: Main control analysis flow chart

* 1. Intent Recognition Module

The intent recognition module reads training data from an Excel file (data/Intents.xlsx), which contains intents and corresponding texts. Each text sentence is split (separated by commas), and each phrase is preprocessed before being stored in the all\_phrases and all\_intents lists, with the preprocessed phrases associated with their corresponding intents. The training phrases are then vectorized using TfidfVectorizer, converting each phrase into a numeric vector that represents the importance of the words in the phrase. The vectorization method uses n-grams from 1 to 3 (i.e., single words, two-word combinations, and three-word combinations).

Preprocess Text Function

The preprocess\_text function preprocesses the input text with the following steps:

Convert the text to lowercase for consistency.

Remove punctuation and special characters, keeping only letters and numbers.

Remove stop words (such as "the", "and", "is", etc.) to simplify the text.

Perform stemming to reduce words to their root forms.

Sentiment Analysis Function

The sentiment\_analysis function determines the sentiment of the input text by checking for positive or negative words.

If the text contains positive words (e.g., "happy", "good"), it returns a positive response.

If the text contains negative words (e.g., "sad", "angry"), it returns a negative response.

If neither positive nor negative words are found, it returns None, indicating that the sentiment cannot be determined.

Predict Intent Function

The predict\_intent function is used to predict the intent of the user's input:

The user's input is first preprocessed (using the preprocess\_text function).

The input is then transformed into a TF-IDF vector, and cosine similarity is calculated between this vector and all vectors of the training data.

The phrase with the highest similarity to the user input is found, and the corresponding intent is returned.

If the maximum similarity is below a set threshold (default is 0.4), the function returns "Other", indicating that no known intent could be matched.

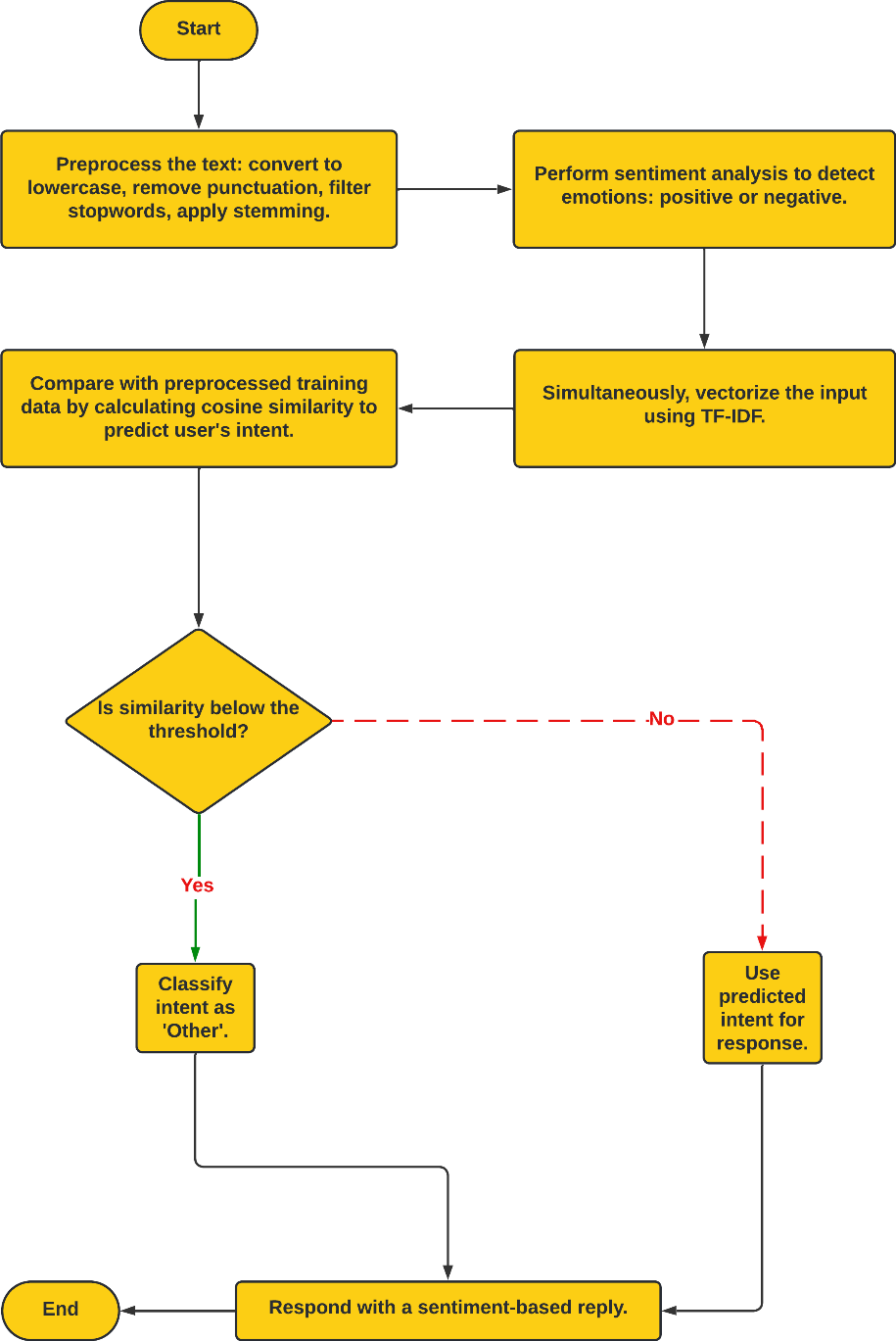


Figure 3: Intent recognition analysis diagram

* 1. Reservation Process Module

The restaurant\_hours is a dictionary that stores the restaurant's operating hours for each day of the week (e.g., Monday to Saturday from 12:00 PM to 10:00 PM, and "Closed" on Sunday, indicating the restaurant is closed).

The send\_confirmation\_email function is used to send a reservation confirmation email to the user's specified email address. The email includes reservation details such as the reservation time, number of people, contact information, etc. This function sends the email via NetEase's SMTP service and handles any failures in email delivery.

The is\_valid\_reservation\_time function validates the user's input reservation time to ensure it meets the required criteria:

Whether the input is in the correct date-time format (DD/MM/YYYY HH:MM AM/PM).

Whether the reservation time is in the future (it cannot be in the past).

Checks if the restaurant is open on the requested day. If the restaurant is closed, the user is notified.

Ensures the input time is within the restaurant's operating hours (e.g., from 12:00 PM to 10:00 PM). If any of these conditions are not met, the function returns an error message.

The check\_for\_restaurant\_query function is used to detect if the user’s input is related to a restaurant query. It checks if the lowercase string of the user's input contains keywords related to the restaurant's information, menu, address, hours of operation, or promotions. If such keywords are found, it returns True, indicating that the user is likely asking about restaurant information.

The make\_reservation function is used to create a new restaurant reservation. By generating a unique reservation ID, the user's reservation details (including name, number of people, date/time, and contact information) are stored in the reservations dictionary. Each reservation is uniquely identified by its reservation ID. The function returns the reservation ID and the details of the reservation.

The get\_reservation\_details function returns the details of a reservation based on the given reservation ID. This includes the reserver's name, number of people, reservation time, status, and contact details. If the given ID is invalid (i.e., no matching reservation is found), it returns a "Reservation not found" message.

The check\_status function queries and returns the status of a reservation based on the given reservation ID (e.g., confirmed, canceled). If the reservation ID is invalid, it prompts with "Reservation not found."

The cancel\_reservation function is used to cancel an existing reservation. It looks up the reservation by its ID and updates its status to "Canceled." If the reservation ID is invalid, it returns a "Reservation not found" message.

The modify\_reservation function allows users to modify confirmed reservation details. Users can change the reserver's name, number of people, date/time, and contact information. The updated reservation details are saved, and the function returns the updated information. If the given reservation ID is invalid, it returns a "Reservation not found" message.

The add\_to\_reservation\_history function adds the reservation details to the history queue (with a maximum capacity of 10 records). The history is stored using a queue structure, ensuring that a maximum of 10 records are retained, and older records are automatically deleted when the limit is exceeded.

The get\_reservation\_history function is used to retrieve historical reservation records. If there are no historical records (i.e., the user has not made any reservations), it returns a prompt message. If there are historical records, the function returns all records listed in order.

The is\_valid\_email function validates the user's input email address to ensure it follows the standard email format. It checks the input using a regular expression to verify whether it is a valid email address. If the format is correct, it returns True; otherwise, it returns False.

The prompt\_for\_input function continuously prompts the user for input based on different requests and validates the input's correctness. The function checks if the user’s input (e.g., number of people, date/time, email, etc.) is valid and re-prompts the user if the input is invalid.

The start\_reservation\_process function is the main function of the reservation process. It guides the user through the entire reservation process:

1. First, it asks for the user's name and confirms the input.
2. Then, it sequentially asks for the number of people, date/time, and contact details.
3. After confirming all the information, the system asks the user to confirm the reservation.
4. If the user confirms, the system creates the reservation and sends a confirmation email.
5. The user can choose to modify, cancel, or check the status of their reservation.
6. If the reservation is confirmed, an email is sent to the user’s email address, and the reservation history is saved.

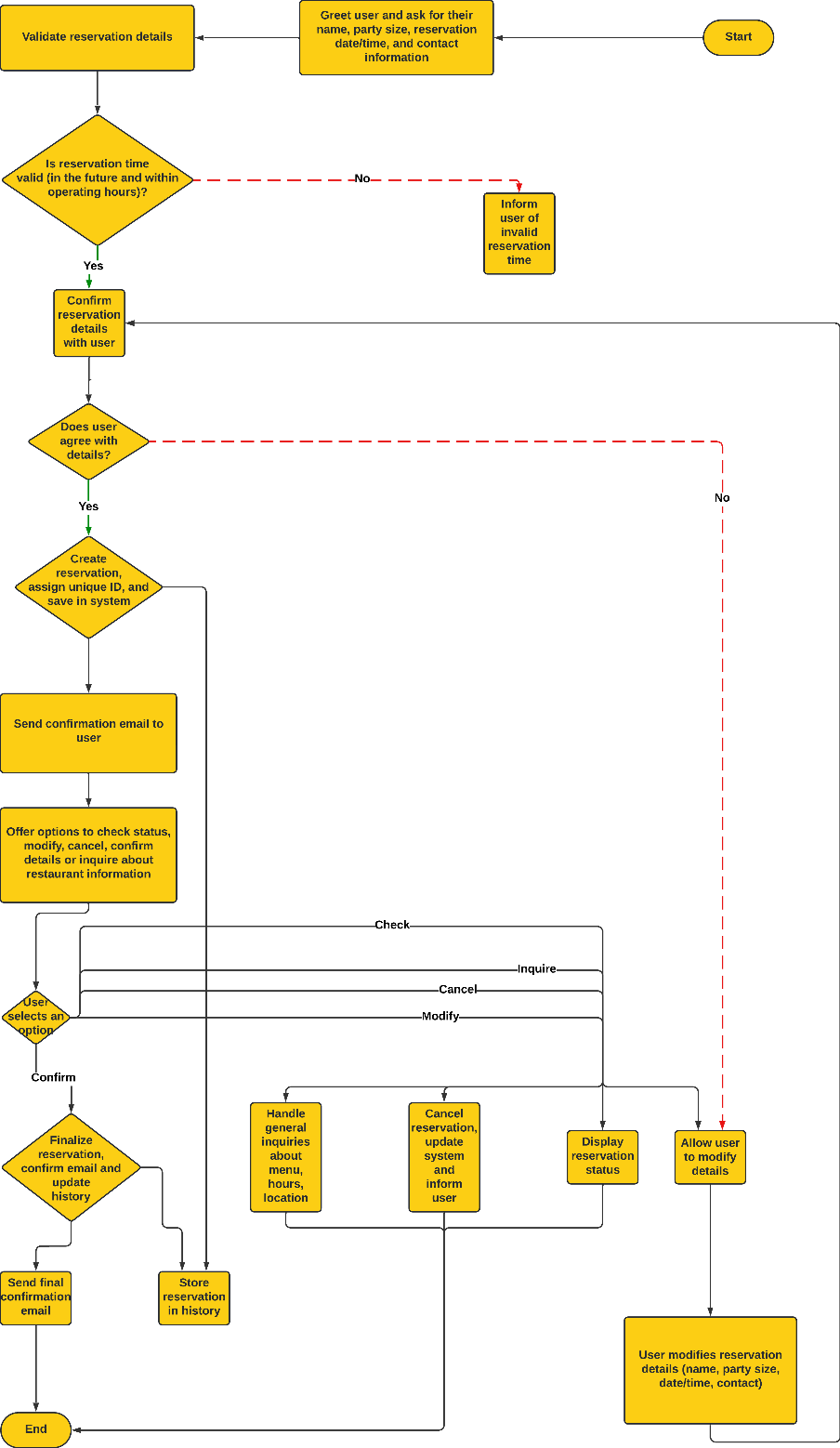


Figure 4: Booking Flowchart

* 1. Knowledge Base Q&A Module (qa\_system.py)

This module loads data from a CSV file, extracting the question and answer columns.

The TfidfVectorizer.fit\_transform function converts the collection of questions into TF-IDF vectors. Using the TfidfVectorizer, each question is transformed into a high-dimensional vector, where each dimension represents the TF-IDF weight of a word in that question.

The google\_search function is triggered when the system cannot find a matching answer in the dataset for the user's question. The function generates a Google search URL based on the user's query and opens the link in the browser, allowing the user to obtain more information through the search engine.

The find\_answer function is responsible for processing the user's input. It calculates the similarity between the input and the set of questions using the TF-IDF vectors. If the similarity exceeds a specified threshold (default 0.7), it returns the matching answer. If the similarity is insufficient, the system prompts the user to rephrase the question or choose another action (such as initiating a Google search or viewing help).

If no suitable matching answer is found, the system will prompt the user to rephrase the question or choose an action by entering a number. The user can choose from the following options:

1. Start a Google search to gather more information.
2. View system help to understand the services the chatbot can provide.

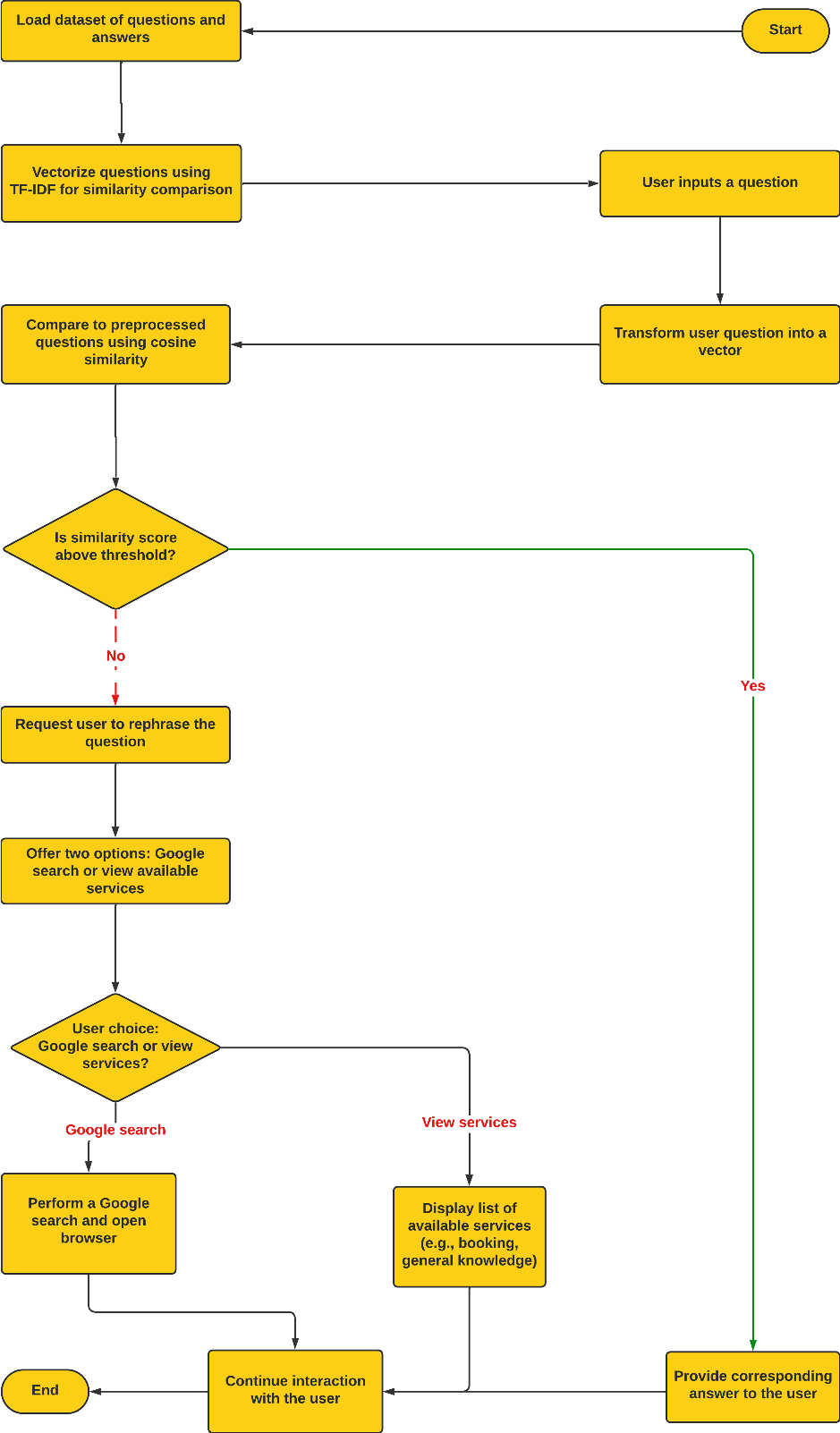


Figure 5: Knowledge base question and answer flow chart

* 1. Restaurant Information Response Module

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

load\_restaurant\_info Function:

This function loads the basic information, menu, and special offers of the restaurant from an Excel file named restaurant\_info.xlsx.

It organizes the restaurant's basic information (such as name, address, phone number, email, website, operating hours, etc.), menu, and special offer details into a dictionary and returns it.

handle\_restaurant\_query Function:

This function handles various queries from the user regarding the restaurant.

Based on the user's input, the function determines the type of query and returns the corresponding restaurant information. For example:

Restaurant Address: Provides the restaurant's address information.

Menu: Lists the restaurant's menu items, including the dish names, prices, and descriptions.

Operating Hours: Provides the restaurant's operating hours.

Special Offers: Lists current promotions or discounts.

Contact Information: Provides the restaurant's phone number or email.

Restaurant Name: Provides the restaurant's name.

If the user's query does not match any of the available categories, the chatbot will prompt the user with specific information categories they can inquire about (such as address, menu, operating hours, special offers, etc.).

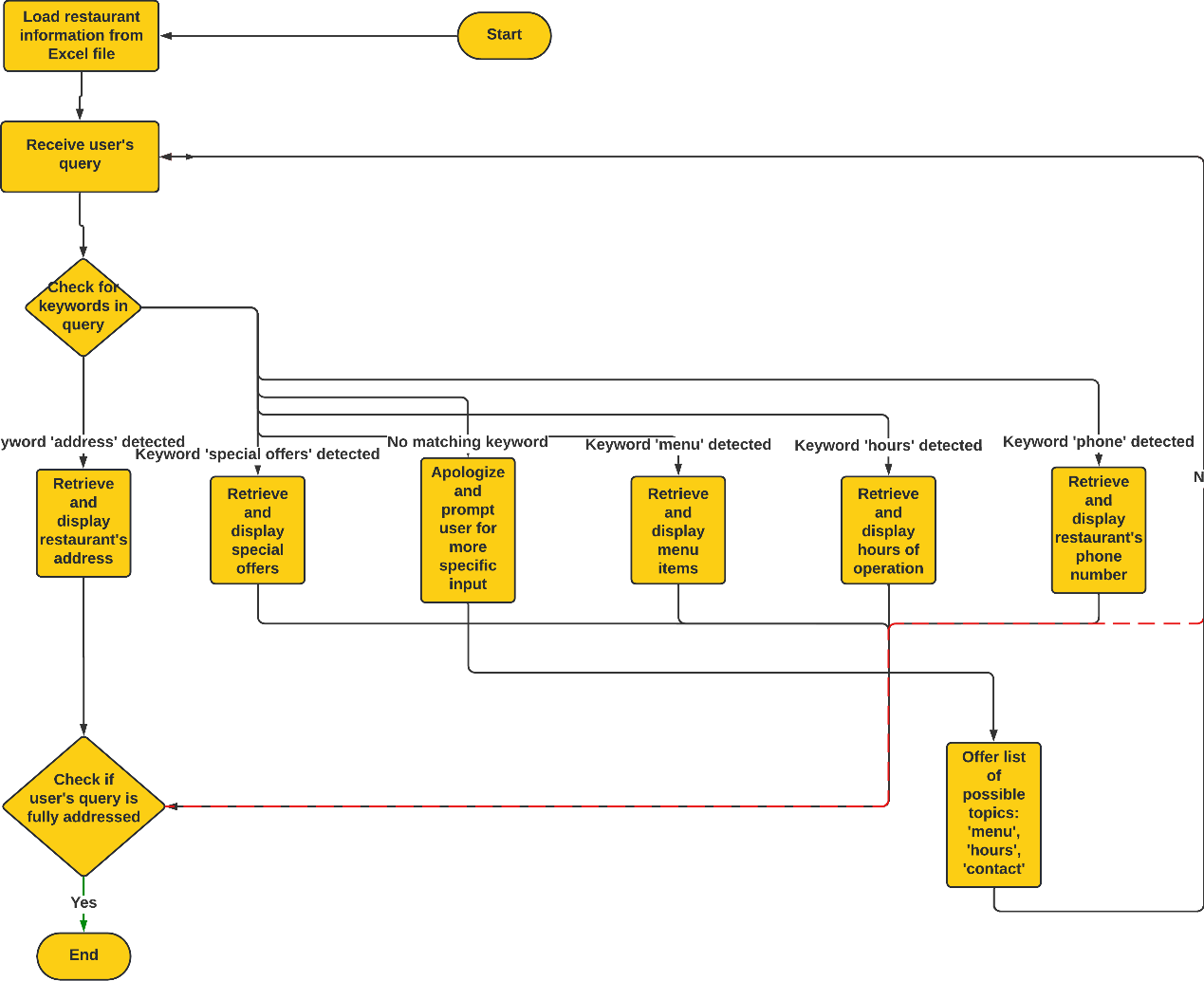


Figure 6: Restaurant information answer flow chart

1. Conversational Design
   1. 3.1 Prompt Design

Conversation Markers and Key Phrases: The 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:"

"You can type 'book' to start the restaurant reservation process."

"Ask about the restaurant, such as its name, opening hours, address, contact information, menu, or special offers."

"Chat with me casually, for example, 'How are you?'"

"Type 'history' to ask about your booking history."

"Ask me some general knowledge questions, like 'How much is 1 tablespoon of water?'"

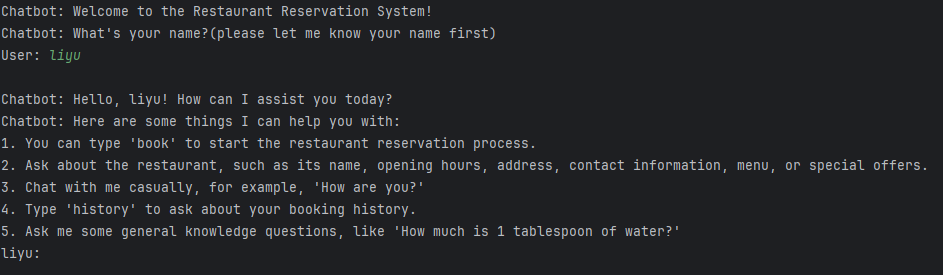
In the reservation process, the prompts might include:

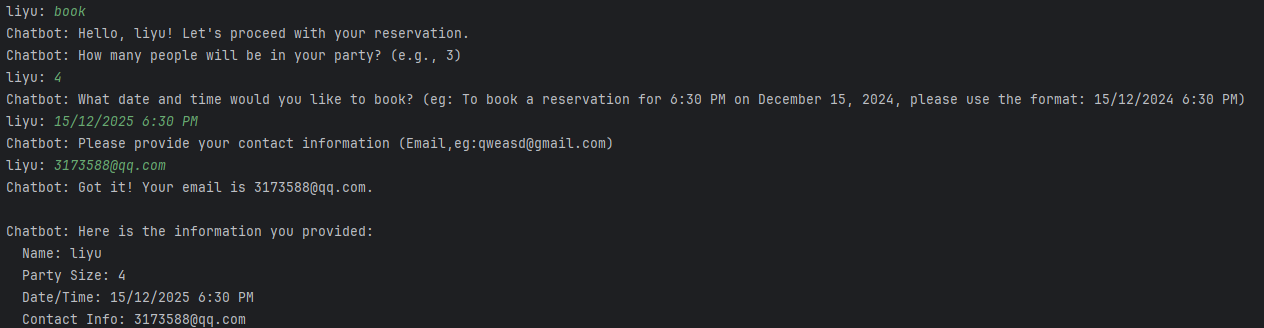
Chatbot: "How many people will be in your party?"

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

Clarity and Conciseness: The prompts are designed to be clear and concise, directly informing users of the ways they can interact with the chatbot. In the reservation process, the chatbot directly asks for required information (e.g., number of people, date/time, contact details), making it easier for users to understand and follow the steps.

Supporting Users and Task Structure: The structure supports multiple types of interaction: casual conversation, restaurant queries, reservation tasks, and other general inquiries. Each prompt is designed to guide the user through a specific task step by step. For instance, the process starts by asking for the number of people, followed by asking for the date/time, and then the contact information. This structured approach helps the user focus on completing each step in the process.





* 1. Discoverability

Contextual Help: The program provides guidance for the next steps, such as:

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

There are also prompts to assist users in providing correct input, such as:

Chatbot: "Please enter a valid number. For example: 3." This helps users input valid information, ensuring smooth interaction.

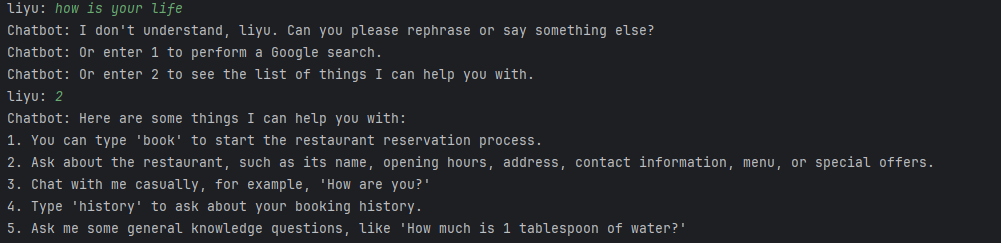
Discovering Features and Content: The program’s prompts explain the different actions users can take (e.g., modify reservation, check reservation status), guiding users to discover the various functions available.

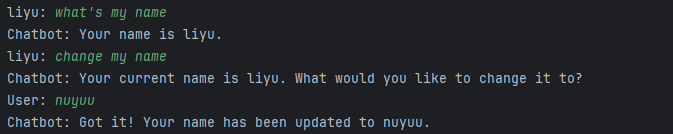












* 1. Error Handling

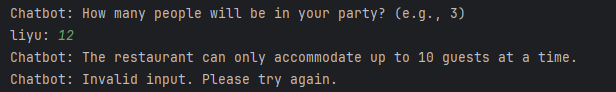
Strategy: The program uses error handling to ensure the validity of user inputs. If the user provides an input the chatbot doesn't understand, the chatbot will say, "I don't understand. Can you please rephrase or say something else?" It also validates input such as party size (1 <= size <= 10), valid email/phone number formats, and date/time formats.

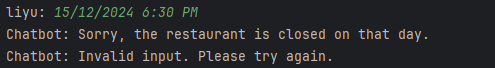
If the user enters invalid information, the system provides an error message and asks for re-input, such as:

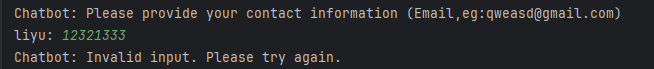
Chatbot: "Please enter a valid number."

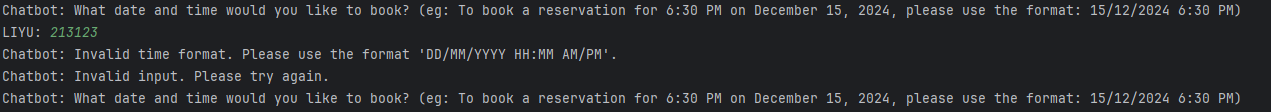
Chatbot: "Invalid time format. Please use the format 'DD/MM/YYYY HH:MM AM/PM'."

Effectiveness: Error messages are clear and instructive, informing the user what went wrong and providing suggestions on how to correct it. This reduces user confusion and frustration.









* 1. Personalization

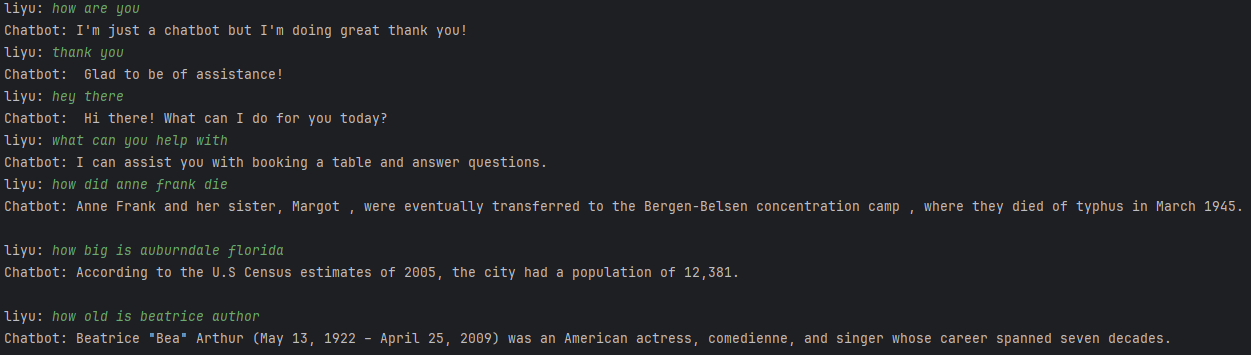
Personalized Experience: The program personalizes the experience by analyzing the user's emotions. For example, when the user's input contains words like "excited" or "joy," the chatbot responds with, "I'm glad you're feeling good!" If the input contains words like "sad" or "bad," the response will be, "I'm sorry you're feeling this way. How can I help?" Users can also change their name, and the chatbot can ask for and remember their name. Additionally, the chatbot can engage in small talk and general knowledge Q&A to offer a more personalized experience, such as:

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

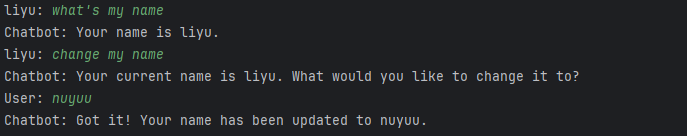
User (liyu): "How old is Beatrice Arthur?"

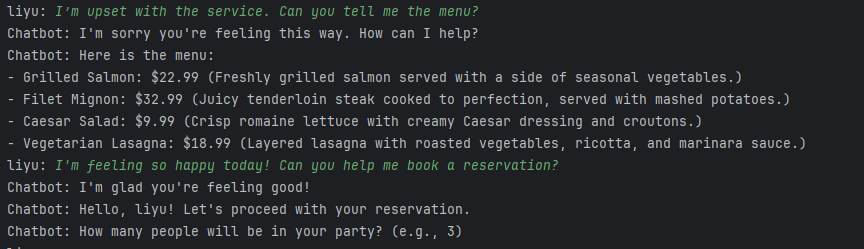
Chatbot: "Beatrice 'Bea' Arthur (May 13, 1922 – April 25, 2009) was an American actress, comedienne, and singer whose career spanned seven decades."

This adds more personalized interaction and engagement to the conversation.









* 1. Confirmation

Conversation Markers and Key Phrases: The program includes clear prompts such as:

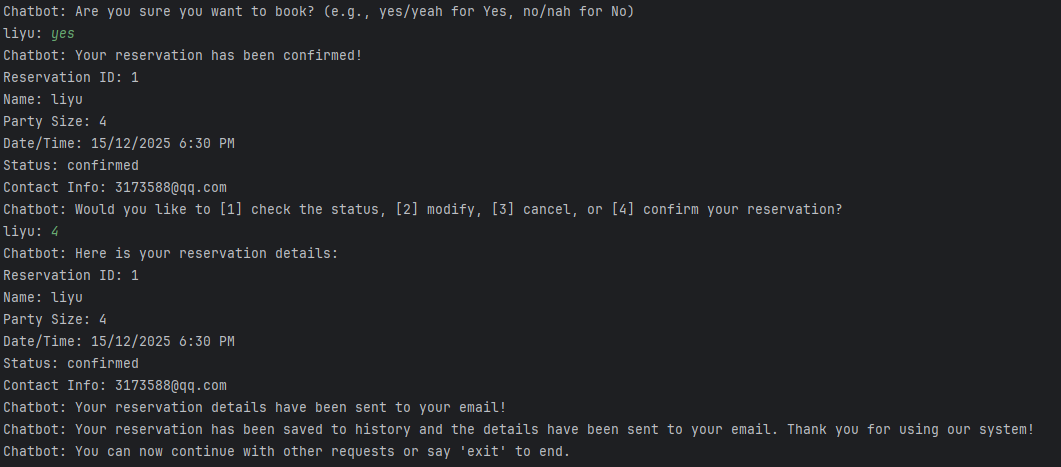
Chatbot: "Your reservation has been confirmed!" The chatbot also includes reservation details and sends a confirmation email to the user. This ensures that users know their request has been successfully processed.

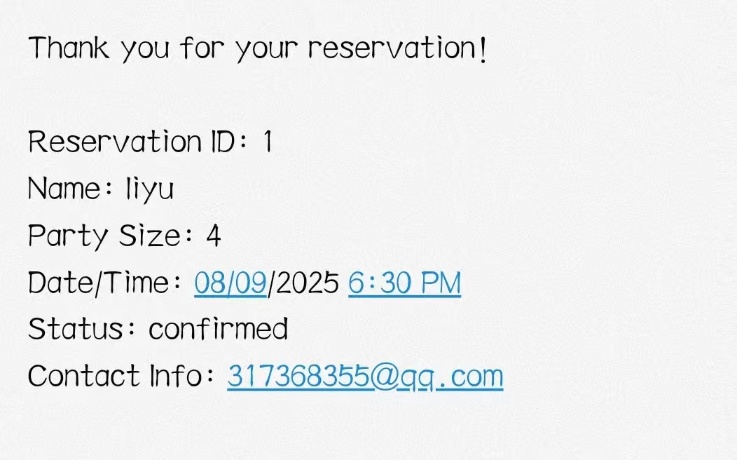
When the user modifies or cancels a reservation, the system provides a confirmation message, such as:

Chatbot: "Your reservation has been updated..."

Chatbot: "Your reservation has been canceled."

Effectiveness: This confirmation strategy is highly effective because it reassures users that their requests have been successfully completed, reducing any potential misunderstandings.



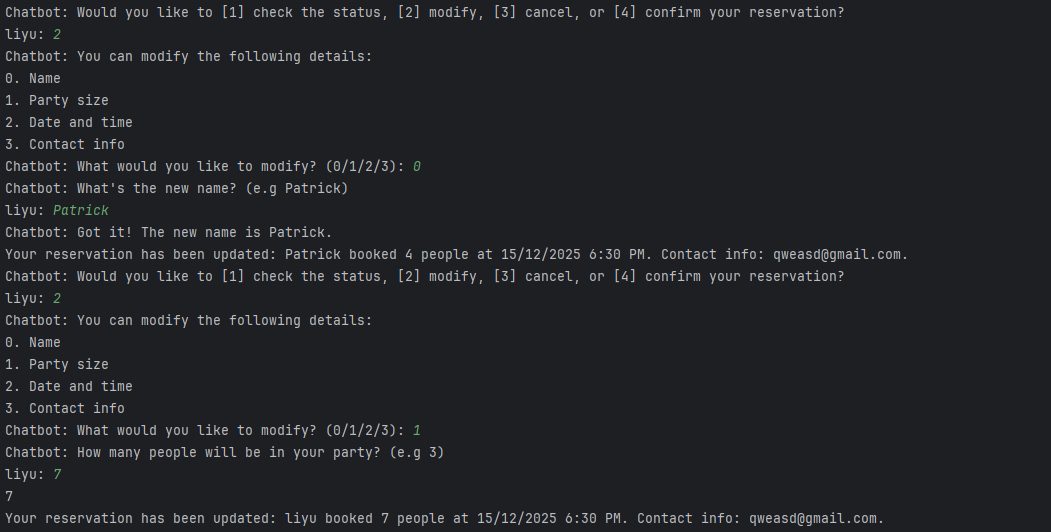


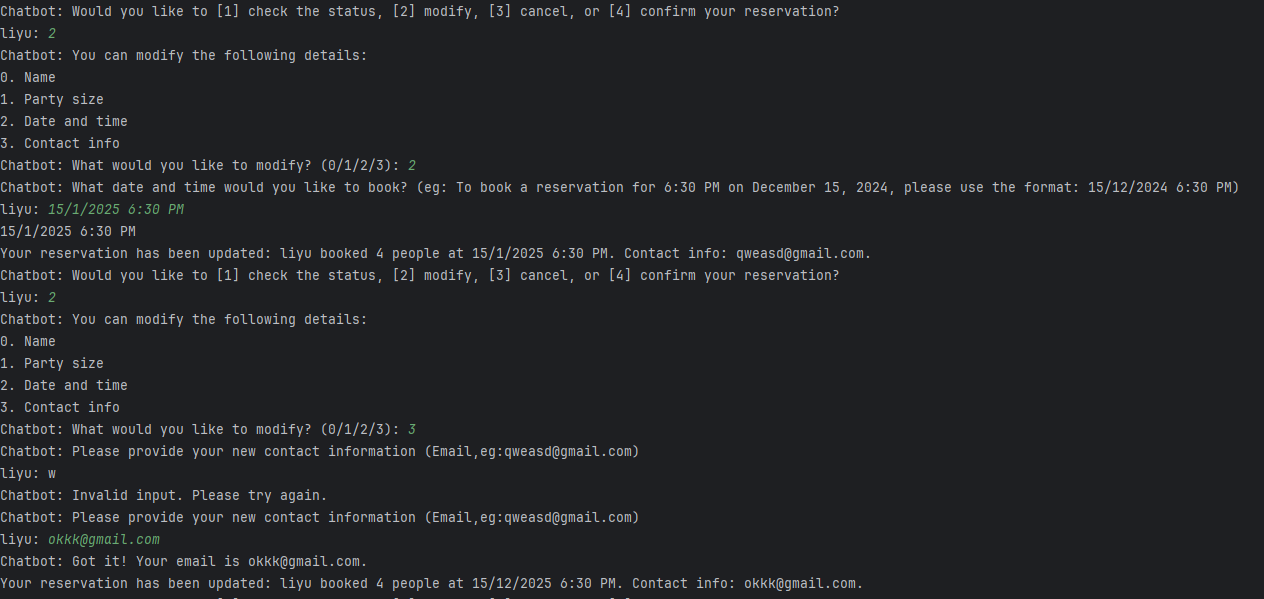
* 1. Context

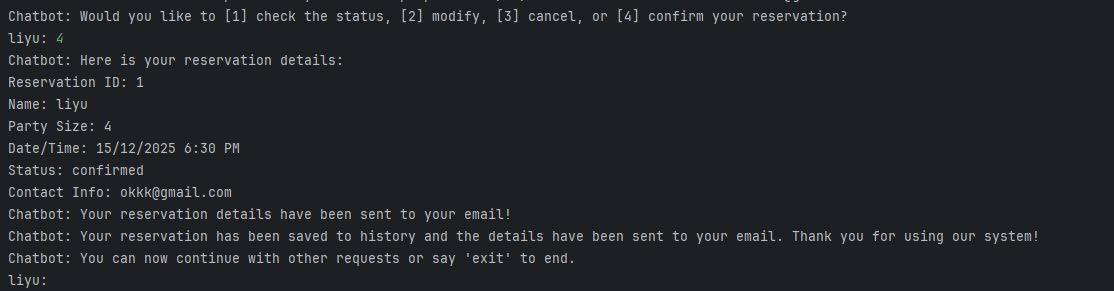
Context Tracking: The system effectively tracks context throughout the reservation process. For example, once the user provides details like party size, date/time, and contact information, the chatbot uses this information to continue processing the reservation. During the process, the user can also ask the chatbot for information about the restaurant and continue with the reservation.

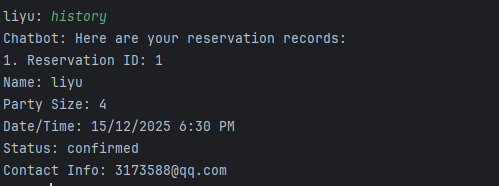
If the user wants to modify or cancel a reservation, the system can recall the details of the previous reservation (stored in reservations) and update or query the information accordingly.

Effectiveness: Context is well managed, with the chatbot providing relevant prompts based on the current interaction stage. Additionally, if the user has an existing reservation, they can view the details of their reservation, ensuring smooth and efficient interaction.



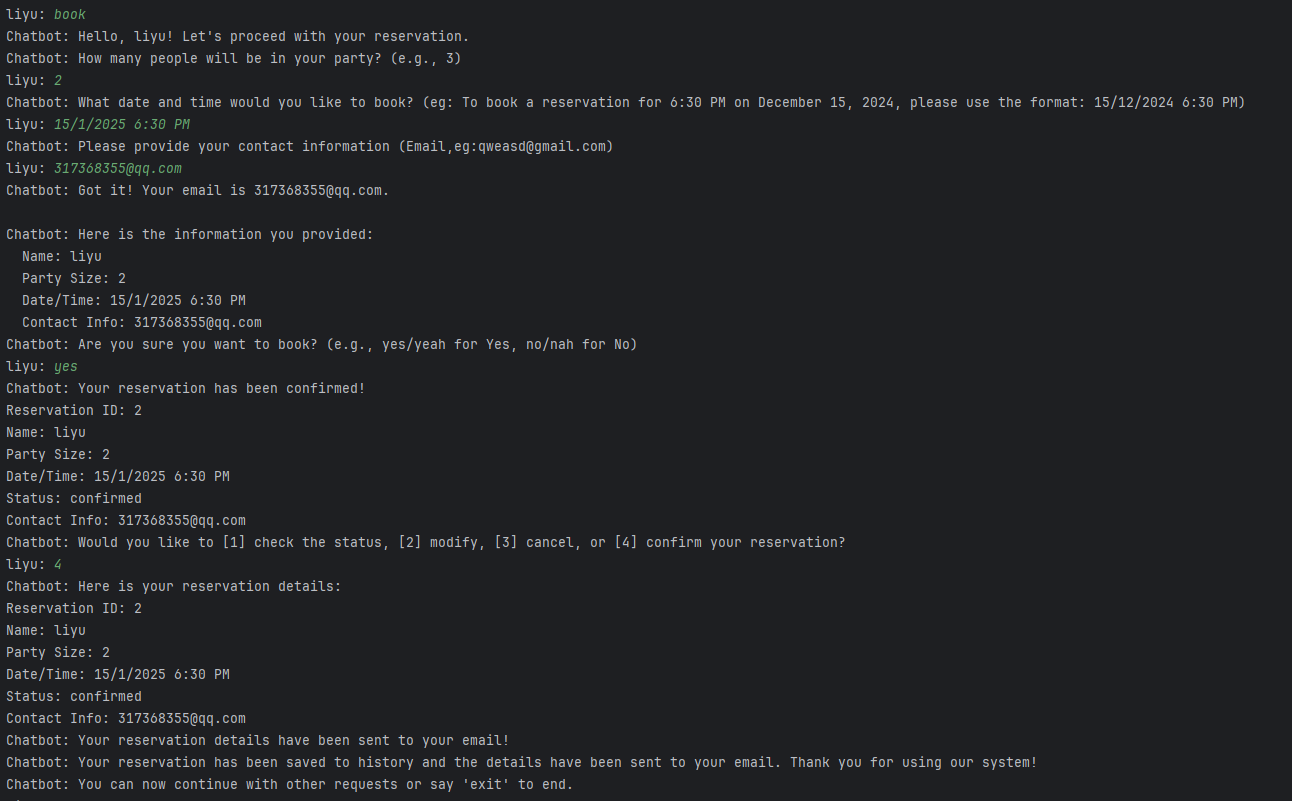






1. Usability Testing Evaluation

I first ran a series of tests on each dataset from the spreadsheet to evaluate different aspects of the system. The final results indicate that the chatbot meets the expected outcomes, showing high reliability and accuracy in intent recognition and response generation. Below are some screenshots from the chat that I tested:



Next, I wrote some questions that users might ask and set corresponding intent labels. I compared the predicted intent with the actual intent by performing a string comparison. If they matched, it indicated that the intent recognition passed; if they were different, it indicated a recognition error. I then assessed the overall accuracy. The results showed that when the user's input was varied, the current intent classifier did not perform well in recognizing the user's intent. However, the critical intent of "book" for restaurant reservation was recognized quite accurately. Currently, the intent classifier only calculates string similarity, which has limitations and is restricted by the predefined intent texts.

Table 1: Classification Report

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score | Support |
| exit | 0.81 | 0.45 | 0.58 | 29 |
| book | 0.9 | 0.97 | 0.94 | 38 |
| hello | 1.0 | 0.32 | 0.49 | 28 |
| wellbeing\_query | 0.29 | 0.5 | 0.37 | 20 |
| capabilities | 0.73 | 0.4 | 0.52 | 20 |
| thanks | 0.95 | 0.9 | 0.92 | 20 |
| positive\_responses | 0.78 | 0.35 | 0.48 | 20 |
| negative\_responses | 0.8 | 0.4 | 0.53 | 20 |
| name | 0.65 | 0.61 | 0.63 | 18 |
| change | 1.0 | 0.68 | 0.81 | 19 |
| time | 0.92 | 0.65 | 0.76 | 17 |
| AskAboutRestaurant | 0.85 | 0.82 | 0.84 | 28 |
| Other | 0.13 | 0.41 | 0.19 | 27 |
| Accuracy | 0.59 |  |  | 304 |
| Macro avg | 0.75 | 0.57 | 0.62 | 304 |
| Weighted avg | 0.76 | 0.59 | 0.63 | 304 |

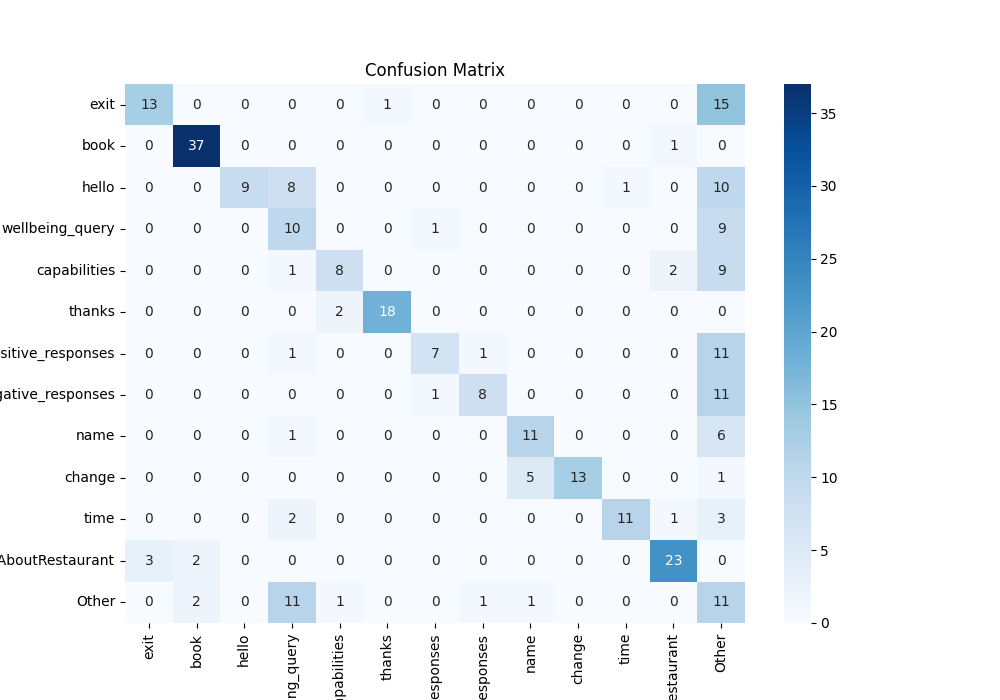


Figure 6: Confusion Matrix

I then selected 10 test users, observed how they interacted with the system, recorded their actions, feedback, and difficulties encountered.

Below is the survey questionnaire for them:

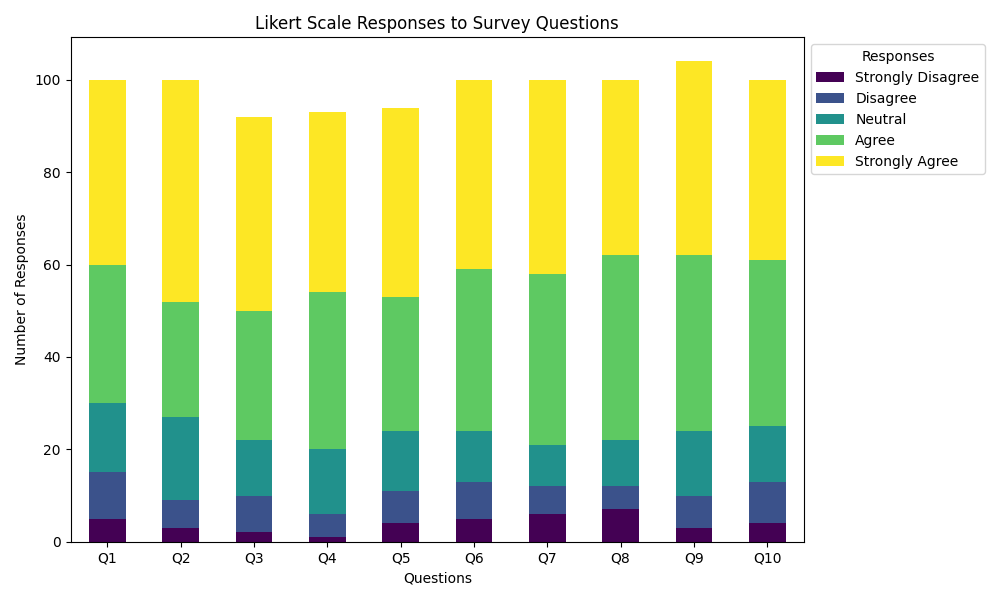


Figure 7: Result analysis chart

The results of the survey show that most people agree with and are satisfied with the system.

1. Discussion

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

Successes:

Core Functionality: The chatbot is capable of engaging in simple conversations with users and performs basic functions such as restaurant reservations, querying menus, and checking operating hours. It meets the essential business requirements.

Conversational AI: By integrating conversational AI, the system allows users to interact naturally, providing a convenient way to engage with restaurant services.

Data-Driven: The system can dynamically retrieve restaurant information (e.g., menu, specials) from Excel spreadsheets and provide accurate answers based on real-time data.

Query Handling: While the system handles basic queries well, user experience declines when processing more complex requests (e.g., detailed reservation modifications).

Scalability: The system can easily accommodate additional functional modules, allowing for expansion.

Next Steps for Improvement:

Advanced NLP Capabilities: Utilize large pre-trained models to enhance the application of natural language processing technologies, improving the system’s ability to handle more complex queries and accurately recognize user intent.

Expand Use Cases:

Multi-Restaurant Support: The system can be designed to support multiple restaurant configurations, with each restaurant having its own menu, specials, and operating hours. This would allow the system to be expanded to support multiple restaurants, further enhancing its applicability.