DEPARTMENT OF COMPUTING AND INFORMATION SYSTEMS
SCHOOL OF ENGINEERING AND TECHNOLOGY

PROGRAMMING PROJECT FOR:
Bachelor of Software Engineering (Hons);
BSc (Hons) Information Technology;
BSc (Hons) in Computer Science;
BSc (Hons) Information Technology (Computer Networking and Security);
BSC (HONS) Data Analytics;
BSc (Hons) Business Studies;

ACADEMIC SESSION: April 2023

CSC1024: PROGRAMMING PRINCIPLES

DEADLINE: 24 July 2023 (Monday), before 6.00 PM.

INSTRUCTIONS TO CANDIDATES

- This programming project will contribute **50%** to your final grade.
- This programming project is a final examination.
- This programming project is a **GROUP** assignment.

IMPORTANT

The University requires students to adhere to submission deadlines for any form of assessment. Penalties are applied in relation to unauthorized late submission of work.

- Coursework submitted after the deadline but within **1 week** will be accepted for a maximum mark of **40%.**
- Work handed in following the extension of **1 week** after the original deadline will be regarded as a non-submission and marked **ZERO**.

Student's Declaration:

(Name) (ID) (Signature)

We 1) Habiba Tarek Abdallah Mohamed Hassouna 21071097 [illegible]

2) Hajah Atikah Haziqah@Nuratikah Haziqah Binti Abd Jalil 22100259 [illegible]

3)Humayra Mohamed Esmail 22103535 [illegible]

received the assignment and read the comments.

Academic Honesty Acknowledgement

"We (names stated above) verify that this paper contains entirely my own work. I have not consulted with any outside person or materials other than what was specified (an interviewee, for example) in the assignment or the syllabus requirements. Further, I have not copied or inadvertently copied ideas, sentences, or paragraphs from another student. I realize the penalties *(refer to the student handbook)* for any kind of copying or collaboration on any assignment."

1) [illegible]

2) [illegible]

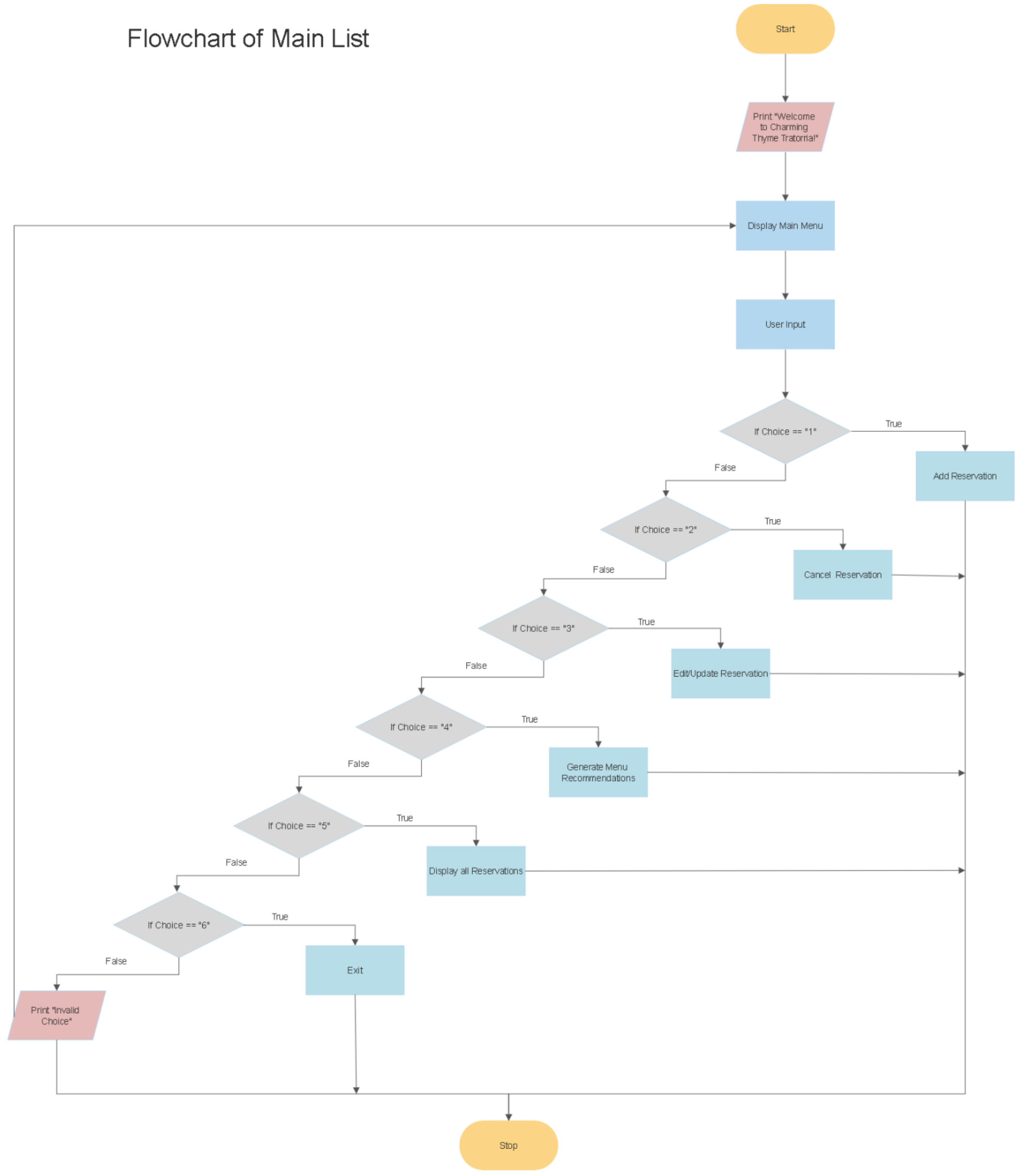
3) [illegible]

4)

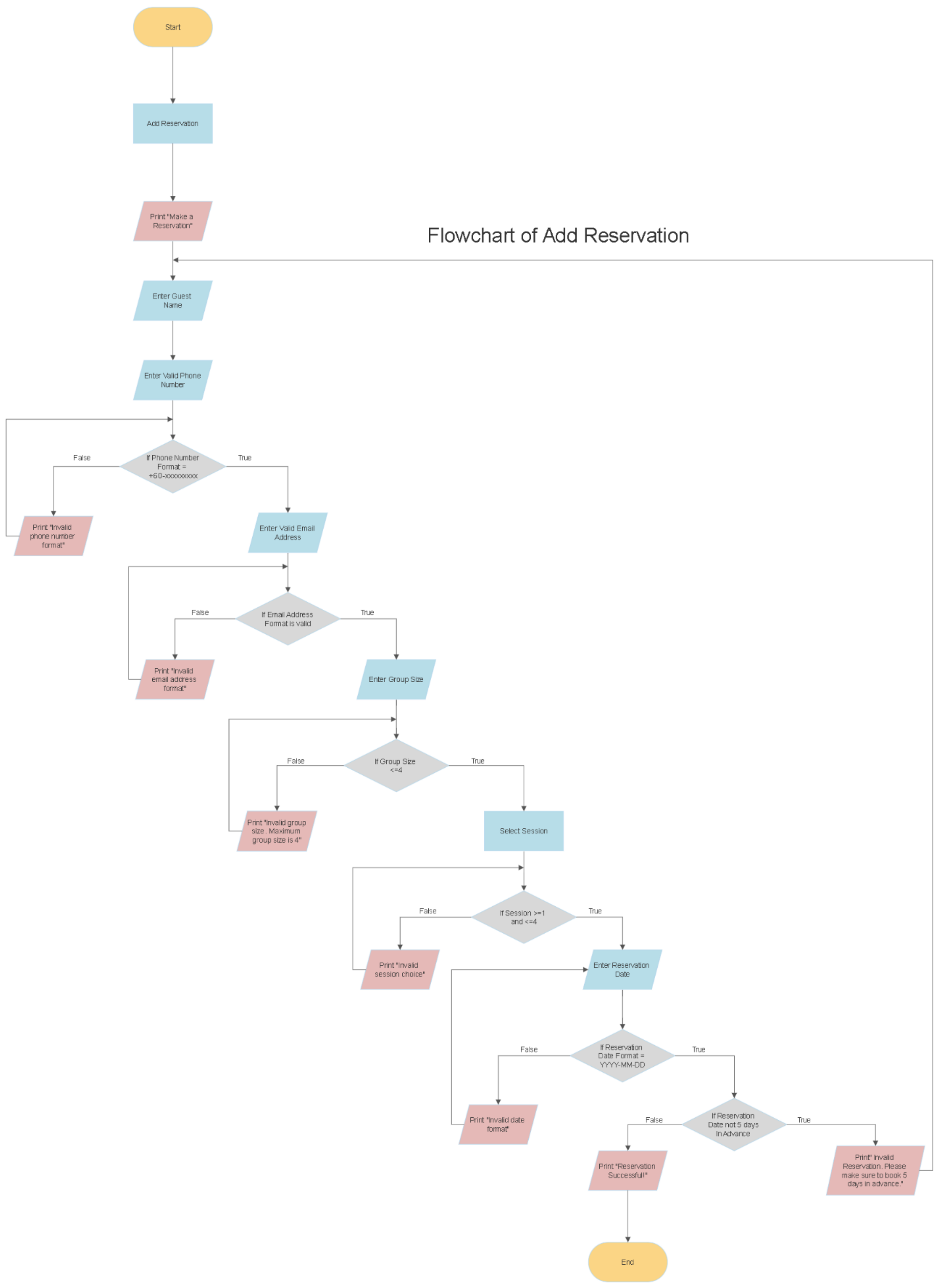
…………………..…………………………24/07/2023…………………………..……….........................

(Student's Signature and Date)

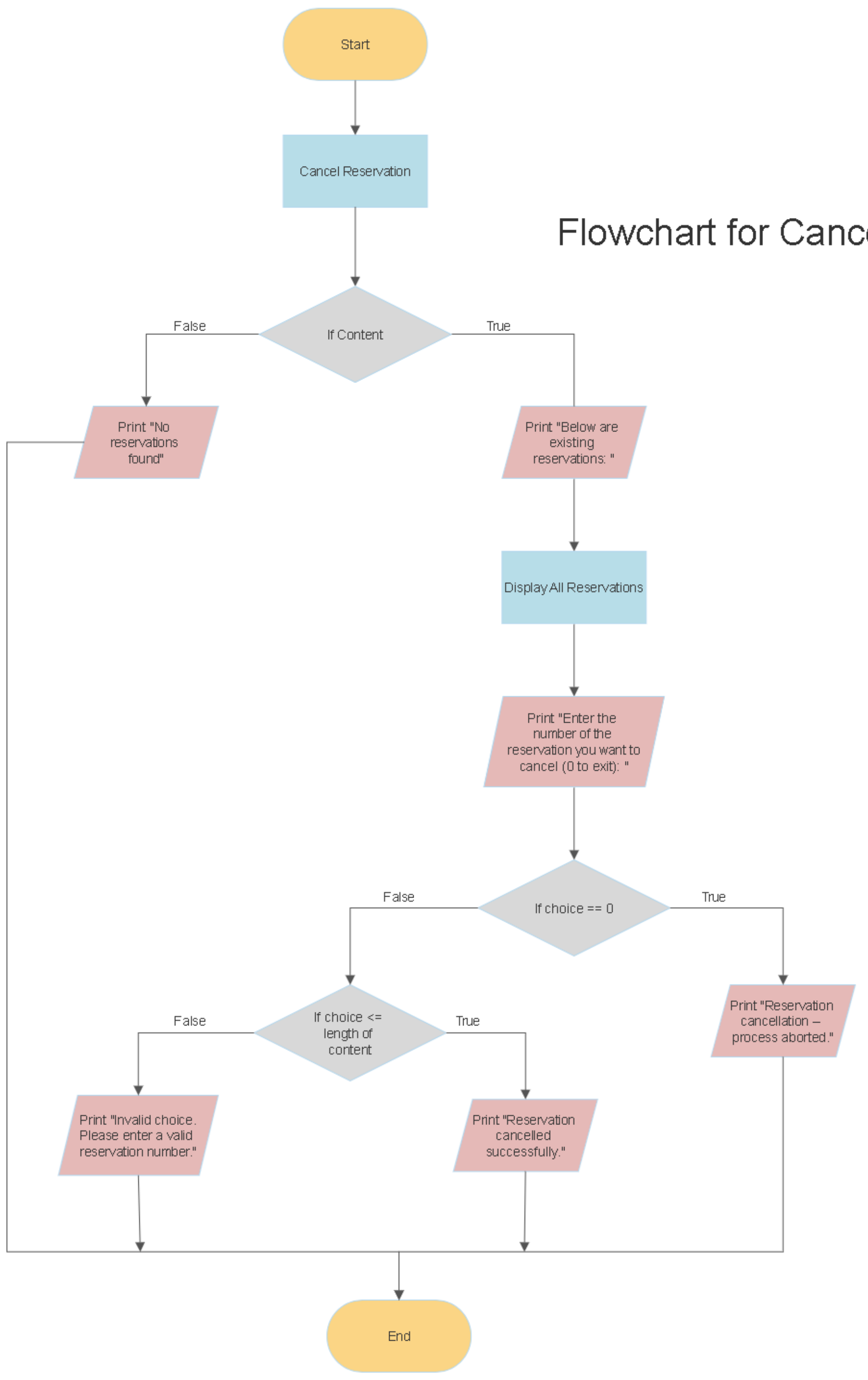
Flowchart:


Flowchart of Main List
Start
Print "Welcome to Charming Thyme Tratorria!"
Display Main Menu
User Input
If Choice == "1"
True
Add Reservation
False
If Choice == "2"
True
Cancel Reservation
False
If Choice == "3"
True
Edit/Update Reservation
False
If Choice == "4"
True
Generate Menu Recommendations
False
If Choice == "5"
True
Display all Reservations
False
If Choice == "6"
True
Exit
False
Print "Invalid Choice"
Stop

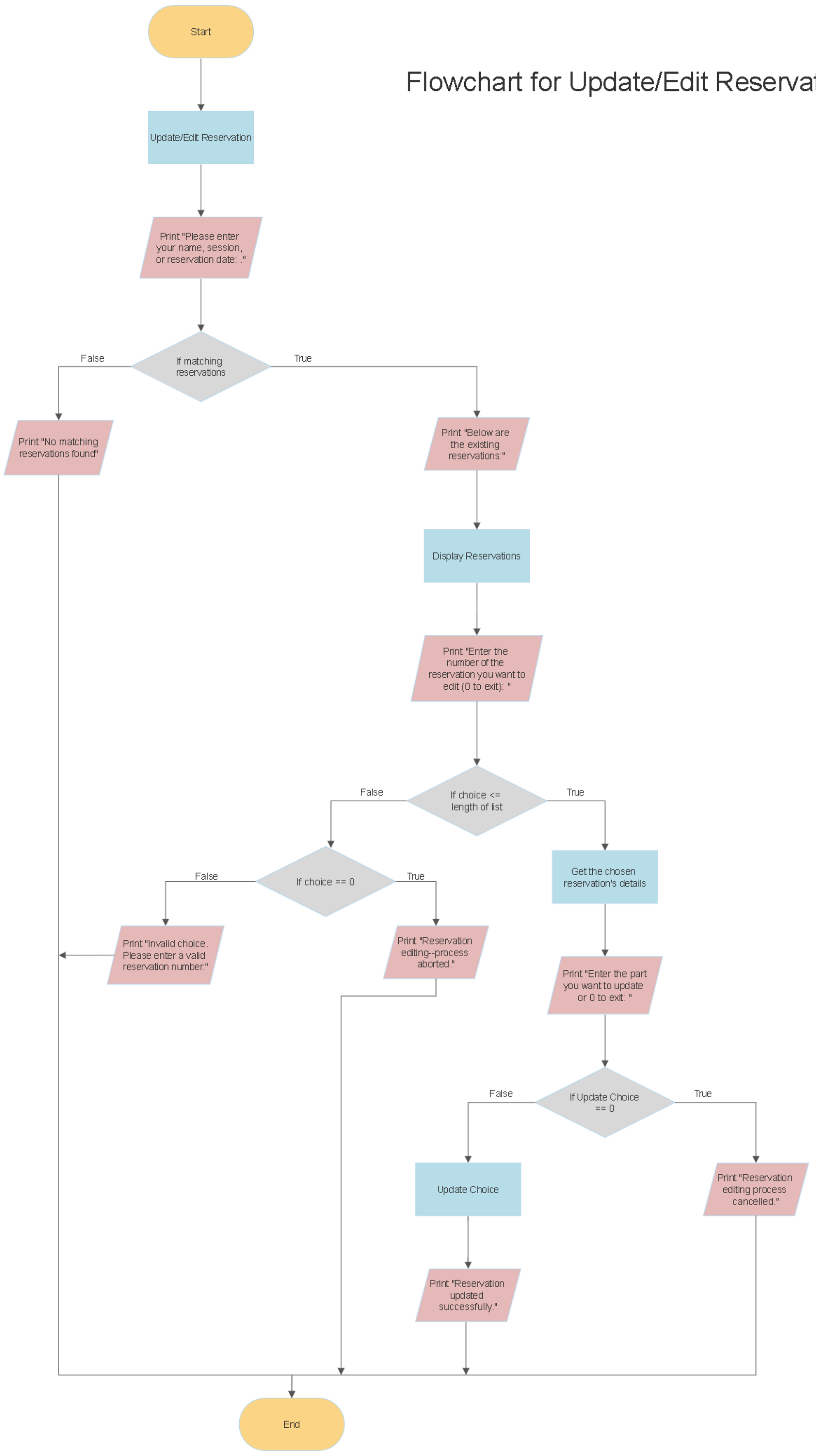
Flowchart of Add Reservation


Start
Add Reservation
Print "Make a Reservation"
Enter Guest Name
Enter Valid Phone Number
If Phone Number Format = +60-xxxxxxxxx
False
True
Print "Invalid phone number format"
Enter Valid Email Address
If Email Address Format is valid
False
True
Print "Invalid email address format"
Enter Group Size
If Group Size <=4
False
True
Print "Invalid group size. Maximum group size is 4"
Select Session
If Session >=1 and <=4
False
True
Print "Invalid session choice"
Enter Reservation Date
If Reservation Date Format = YYYY-MM-DD
False
True
Print "Invalid date format"
If Reservation Date not 5 days in Advance
False
True
Print "Reservation Successfull"
Print" Invalid Reservation. Please make sure to book 5 days in advance."
End

Flowchart for Cancel Reservation

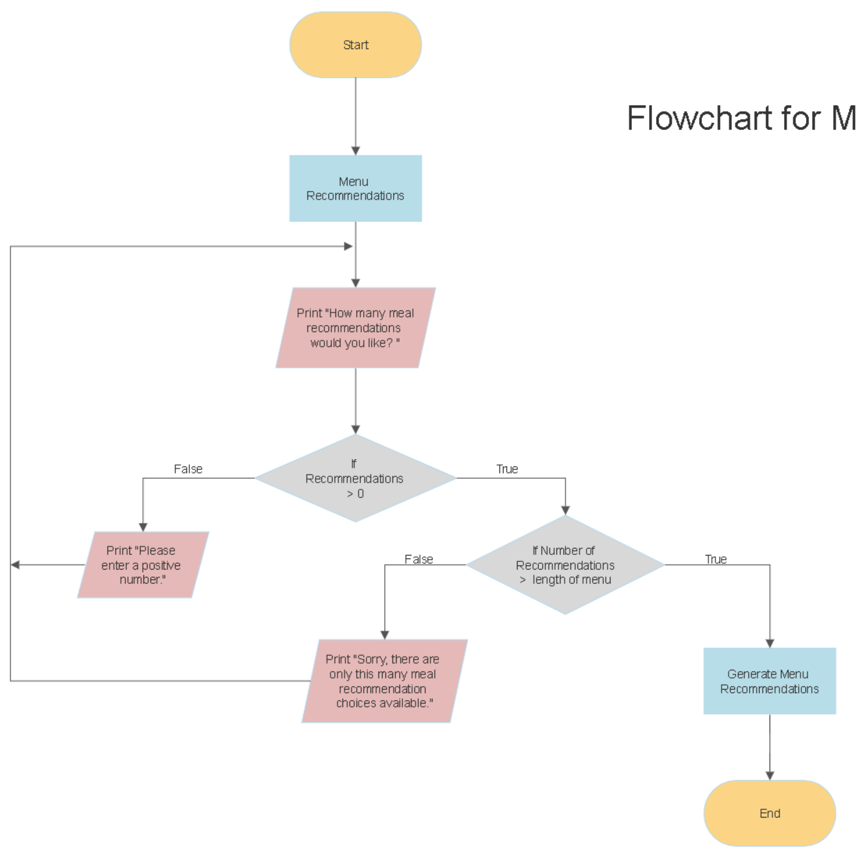

Start
Cancel Reservation
If Content
False
True
Print "No reservations found"
Print "Below are existing reservations: "
Display All Reservations
Print "Enter the number of the reservation you want to cancel (0 to exit): "
If choice == 0
False
True
If choice <= length of content
False
True
Print "Reservation cancellation – process aborted."
Print "Invalid choice. Please enter a valid reservation number."
Print "Reservation cancelled successfully."
End

Flowchart for Update/Edit Reservation



Start
Update/Edit Reservation
Print "Please enter your name, session, or reservation date: ."
If matching reservations
False
True
Print "No matching reservations found"
Print "Below are the existing reservations:"
Display Reservations
Print "Enter the number of the reservation you want to edit (0 to exit): "
If choice <= length of list
False
True
If choice == 0
False
True
Print "Invalid choice. Please enter a valid reservation number."
Print "Reservation editing--process aborted."
Get the chosen reservation's details
Print "Enter the part you want to update or 0 to exit: "
If Update Choice == 0
False
True
Update Choice
Print "Reservation editing process cancelled."
Print "Reservation updated successfully."
End

Flowchart for Meal Recommendations


Start
Menu Recommendations
Print "How many meal recommendations would you like? "
If Recommendations > 0
False
True
Print "Please enter a positive number."
If Number of Recommendations > length of menu
False
True
Print "Sorry, there are only this many meal recommendation choices available."
Generate Menu Recommendations
End

Techniques used:

Importing several modules:
from datetime import datetime, timedelta
The **'datetime'** module provides classes for manipulating dates and times. It is used in this code to handle reservation dates and calculate time differences using the **'timedelta'** class.

import re
The **'re'** module stands for "regular expressions" and provides support for pattern matching operations using regular expressions. It is used in this code to validate the phone numbers and email addresses of the customers.

import random
The **'random'** module provides functions to generate random numbers, sequences, and selections. It is used in this code to randomly generate the menu recommendations.

import os
The **'os'** module provides a way of using operating system-dependent functionality, like clearing the terminal screen. It is used in this code to clear the terminal screen when the user chooses to exit the program.

Classes and functions:
Classes: Classes are used to create a blueprint for objects that share common properties and behaviours. The **'Reservations'** and **'Menu'** classes are created to encapsulate released data and functions together. The **'Reservations'** class represents the reservation management system, and the **'Menu'** class represents the menu recommendations system. By using classes, the code becomes more organized, and it is easier to maintain and extend functionalities in the future. Each class contains its own methods (functions) that handle specific tasks related to reservations or menu recommendations.

Functions: Functions are used to define blocks or reusable code that can be called multiples times within the program. Functions in this code represent different actions that the user can take and the corresponding functionalities.
For example:

- **make_reservation()**: This function handles the process of making a reservation by gather guest information and validating the reservation details.
- **cancel_reservation()**: This function allows the user to cancel and existing reservation.
- **edit_reservation()**: This function enables the user to modify details of an existing reservation.
- **generate_recommendations()**: This function generates random menu recommendations for the user.
- **print_reservations()**: This function displays all existing reservations.
- Functions like **'get_valid_phone_number()'**, **'get_valid_email_address()'**, and more are functions that handle the validation of phone numbers, email addresses, and other input fields to ensure the data entered by the user is in correct format.

The **‘Reservations’** class has an **‘_ _ init_ _’** methods initializes two attributes: **‘self.reservations’** and **‘self.reservation_dates’**.

- **‘self.reservations’**: This attribute is a dictionary that stores reservations for different time sessions. The restaurant has four-time sessions during the day, each of these time sessions can have multiple reservations made by different guests. Therefore, **‘self.reservations’** is used to keep track of these reservations. It is structured as a dictionary, where the keys are the time sessions, and the values are lists of reservations made for each session. Each reservation is represented as a tuple containing guest information (name, phone number, email address, group size, session, and reservation date).
- **‘self.reservation_dates’**: This attribute used to store reservation dates categorized by session. Since each session can have multiple reservations, **‘self.reservation_dates’** is used to keep trach of the dates on which reservations are made for each session. It is structured as a dictionary, where the keys are the time sessions, and the values are lists of reservation dates for each session. This helps to program easily check for valid reservations and manage the reservation availability for each time session.

Main menu:
Def **main_menu(self)**: this function is responsible for displaying the main six options and handling the choices.

<u>User input and looping:</u>
The code uses a **‘while True’** loop to keep the main menu running until the user decides to exit. The ‘while True’ loop ensures that the main menu is displayed repeatedly, allowing the user to perform multiple actions without having to restart the program. The loop will continue running until option 6 has been chosen to exit.

<u>Display of Options:</u>
The **‘print’** statements display the available options to the user, presenting a clear list of actions they can perform within the reservation system. Each option is associated with a number, and the user can enter the corresponding number to select the desired action.

<u>User Choice Handling:</u>
The code uses a series of ‘**if-elif-else'** statements to handle the user’s choice.
After presenting the menu options, the code waits for the user to input their choice. Depending on the user’s input, the program executes the corresponding actions. For example, if [1], add reservation, is chosen, the **‘make_reservation()’** called method is called. Similarly, for the other options.

<u>Calling Methods from the Class:</u>
The code calls various methods defined within the same class: ‘**self.make_reservation()**’, **‘self.cancel_reservation()**’, **‘self.edit_reservation()’**, and **self.print_reservations()’**.
By using the keyword **‘self’**, the code can access and call other methods within the same class. This encapsulates related functionalities within the class, making the code more organized and modular.

Using Another Class:
The code uses the **'Menu'** class to generate menu recommendations when the user selects option 4.

Terminal Screen Clearing:
Whenever the user chooses to exit the program (option 6 is selected), the code clears the terminal screen using the **'os.system'** function. This provides a cleaner interface and avoids cluttering the terminal with previous outputs.

Add reservation:
def make_reservation(): this method utilizes a combination of user input handling, validation, data storage, file handling, and control flow techniques to ensure a smooth and accurate reservation process for the restaurant's customers.

User input and looping:
The code uses nested **'while True'** loop to ensure the reservation process continues until the user decides not to make any more reservations. The inner **'while True'** loop is used to handle the user input for the number of guests (group side) to ensure it is a valid value (not exceeding the maximum group size of 4).

User Input and Validation:
The code uses the **'input'** function to display prompts to the user and capture their input. The code then calls specific validation functions (**'get_valid_phone_number()'**, **'get_valid_email_address()'**, **'is_valid_group_size()'**) to ensure that the phone number, email address and group size provided by the user are in the correct format and meet the reservation criteria.

Method calls within the class:
The code calls other methods with the class: **'self.get_valid_phone_number()**', **'self.get_vald_email_address()'**, **'self.is_valid_group_size()'**, **'self.select_session()'**, **'self.get_reservation_date()'**, **self.is_valid_reservation()'**, **'self.is_available()'**, **'self.write_to_file()'**.
By using the keyword **'self'**, the code can access and call other methods within the same class. This encapsulates related functionalities within the class, making the code more organized and modular.

Conditional Statements:
The code uses **'if'** statements to check whether the selected session and reservation date are valid (at least 5 days in advance) and whether the selected session has available capacity for the specified group size. If any of these conditions are not met, appropriate error messages are displayed, and the user is prompted to enter valid information.

Data Storage and Manipulation:
The code stores reservation information in dictionaries (**'self.reservations'** and **'self.reservation_dates'**) and appends new reservations to the existing data.

The **‘self.reservations’** dictionary is used to store reservations based on the selected session, where each session has a list of reservations. The **‘self.reservation_dates’** dictionary is used to store reservation dates for each session. Whenever a new reservation is made, it is appended to the respective session’s list in **‘self.reservations’**, and the reservation date is appended to the corresponding session’s list in **‘self.reservation_dates’**.

<u>File Handling (Write to File):</u>
The code writes reservation details to a text file using the **‘write_to_file()**’ method.
The **‘write_to_file()’** method takes the reservation details (guest name, phone number, email address, group size, session, and reservation date) and appends them to a text file. This allows the reservation data to be saved and retrieved across different program runs, providing persistence for reservation records.

<u>Adding another reservation:</u>
After successfully making a reservation, the user is asked whether they want to add another reservation. The user’s response is checked using an **‘if’** statement. If the user answers “yes”, the reservation process continues, otherwise, the process exits the loop and returns to the main menu.

Cancel Reservation:
def ‘cancel_reservattion(): this method demonstrates file handling techniques to read and write reservation data, condition statements for decision-making, user input validation, and method calls with the class for updating the in-memory data. The code allows the users to cancel reservations while ensuring the reservation data is consistent between in-memory storage and file.

<u>File Handling (Read from File):</u>
The code reads reservation data from a text file using the **‘open()’** function and reads the content with **‘file.readline()’**.
The method starts by opening the specified file path in read mode (“**r**”) and reads all the lines from the file into the ‘content’ list. This allows the program to access and manipulate the existing reservation data.

<u>Conditional statement and User input:</u>
The code uses **‘if’** statements to handle various scenarios related to the existing reservations and user input.
The method first checks if there are any reservations (**‘if not content’**) and prints a message if the file is empty. If there are any reservations, it prints a list of existing reservations with their details. The method then enters a loop to ask the user for the reservation number they would like to cancel. It validates the user input to ensure it is a valid reservation number (**‘1<= choice <= len(content)’**) and provides the option to exit the cancellation process by entering 0.

<u>File Handling (Removal – Write to File):</u>
After obtaining the user’s choice of reservation to cancel, the method opens the file again, but this time in write mode (“**w**”). It iterates through the ‘**content**’ list, excluding the chosen

reservation (identified by its index ‘**i**’) and writes all other reservations back to the file, effectively removing the canceled reservation.

Calling Methods within the Class:
After canceling the reservation and updating the file, the code calls the **‘update reservation_data()’** method within the class. This method updates the **‘self.reservations’** dictionary based on the modified data on the file, ensuring the in-memory reservations data matches the file data.

Exception Handling:
The code uses exception handling with a **‘try-except’** block to handle potential file not found errors.
The **‘try’** block contains the code that might arise a **‘FileNotFoundError’** if the specified file is not found. The **‘except’** block handles this exception and prints an error message if the file is not found. This prevents the program from crashing and provides informative feedback to the user.

Edit/Update Reservation:
def **update_reservation_data** method ensures the **‘self.reservations’** dictionary accurately reflects the reservation data from the file, and the **‘edit_reservation’** method allows the user to interactively update specific parts of a reservation, ensuring the file is correctly updated with the changes made by the user. The techniques used in these methods contribute to a user-friendly reservation system.

File handling (Read from File – **update_reservation_data**):
The code reads the reservation data from a text file using the **‘open()’** function and reads the content with **‘file.readlines()’**.
The method starts by opening the specified file path in read mode (“**r**”) and reads all the lines from the file into the ‘content’ list. This allows the program to access the update the existing reservation data.

Data Storage and Manipulation (**update_reservation_data**):
The **‘self.reservations’** dictionary is initialized with four empty time sessions. Then, for each line in the file content, the method parses the reservation details (reservation_date, session, name, email_address, phone_number, group_size) and appends the reservation to the appropriate session’s list in **‘self.reservations’**.

File Handling (Write to File – **edit_reservation**):
After the user has selected a reseravtion to edit and made the necessary updates, the method opens the file again, but this time in write mode (“**w**”). It then writes the modified content (including the updated reservation details) back to the file, effectively saving the changes made by the user.

User Input and Validation (**edit_reservation**):
The user is guided through the process of updating a reservation by displaying the existing reservation details and allowing the user to choose which part (name, email, session, date, phone number or group size) they would like to update. The user’s input is validated to ensure it matches the available options and the input data is valid.

Looping and Conditional Statements (**edit_reservation**):
The code uses a **'while True'** loop to ensure that the user can continue editing the reservation until they choose to exit the process. It also uses an inner **'while True'** loop to validate the user's input for choosing the reservation to edit and part to update. If the user enters an invalid input, the loop continues until valid input is provided.

Calling Methods within the Class (**edit_reservation)**:
When the user chooses to update the session, data, phone number, or group size, the corresponding methods: '**self.select_session()'**, **'self.get_reservation_date()'**, **'self.get_vald_phone_number()'**, are called to handle the user input and validation. After successfully updating the reservation, the code calls **'self.update_reservation_date()'** to synchronize the in-memory **'self.reservations'** dictionary with the changes made in the file.

Exception handing:
The code uses exception handling with a **'try-except'** block to handle potential file not found errors.
The **'try'** block contains the code that might arise a **'FileNotFoundError'** if the specified file is not found. The **'except'** block handles this exception and prints an error message if the file is not found. This prevents the program from crashing and provides informative feedback to the user.

Generate Menu Recommendations:
def generate_recommendations method generates random meal recommendations from the available menu items, considering user input for the number of recommendations. The techniques used in this method provide flexibility and a user-friendly experience for obtaining menu suggestions.

File Handling (Read from File):
The code reads menu items from the file **'menuItems_21071097.txt'** using the **'open()'** function and reads the content with **'file.readlines()'**. The method opens the specified menu items file in read mode ("**r**") and reads all the lines from the file into the **'menu_items'** list. This allows the program to access the available menu items for generating recommendations.

Random Selection (Random Sampling):
The method prompts the user to input the number of meal recommendations that they would like. It then uses **'random.sample()'** to randomly select the specified number of recommendations from the **'menu_items'** list. This ensures that the recommendations are randomly chosen each time the method is called.

User Input and Validation:
The code requests the user to input the desired number of meal recommendations. It checks if the input is a positive number and if it does not exceed the total number of available menu items. If the user enters an invalid input, an appropriate error messages are displayed.

Looping and Enumerate:

After generating the meal recommendations, the method uses a **‘for’** loop with **‘enumerate()’** to display each recommendation along with its corresponding number. This provides a user-friendly way of presenting the menu recommendations.

<u>Exception Handling (File Not Found):</u>
The code uses exception handling with a **‘try-except’** block to handle the case when the **‘menuItems_21071097.txt’** file is not found.
The **‘try’** block contains the code that might arise a **‘FileNotFoundError’** if the specified file is not found. The **‘except’** block handles this exception and prints an error message if the file is not found.

Display:
def print_reservations method reads and displays the existing reservations stored in the text file in a structured format. It takes care of potential file-related errors.

<u>File Handling:</u>
The method uses file handling to read the contents of the reservations text file. It opens the file using **‘with open(file_path, “r”) as file:’** and reads its content using **‘content = file.readline()’**. The file is closed automatically once the block inside the **‘with’** statement is executed.

<u>Splitting:</u>
The method splits each line of the file content into separate fields using the ‘split(“**I**”)’ method.

<u>Displaying Reservation Details</u>:
After splitting the content, the method displays the reservation details using **‘print’** statements. It prints information such as reservation date, session, name, email address, phone number, and group size for each reservation in the file. It uses string formatting to present the details clearly.

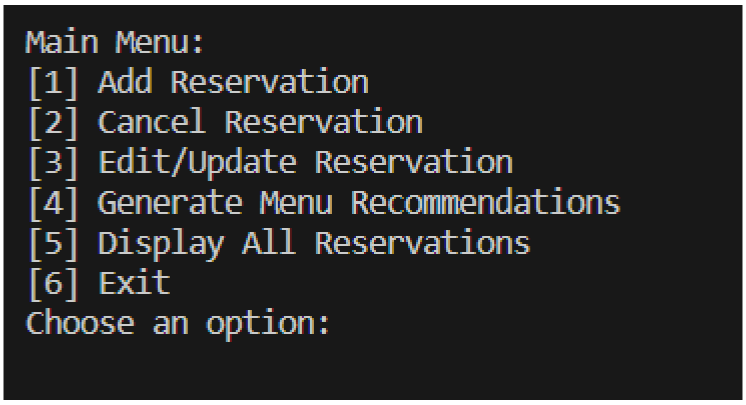
<u>Handling Empty File:</u>
The method checks if the content is empty (no reservations found) and displays a message accordingly.

<u>Exception Handling:</u>
The method uses a **‘try-except’** block to catch a **‘FileNotFoundError’** in case the reservations file does not exist or the provided file path is incorrect. It prints an appropriate error message in such cases.

Exit:
When the user selects the 6th option, the exit, the programs clears the terminal screen using **‘os.system(‘cls’ if os.name == ‘nt’ else ‘clear’)’**and prints “Exiting the program”.
The program then prompts the user to enter the word “charming” to start a new reservation.
If the user enters “charming”(case insensitive), the program clears the terminal screen again using **‘os.system (‘cls’ if os.name == ‘nt’ else ‘clear’)’**.

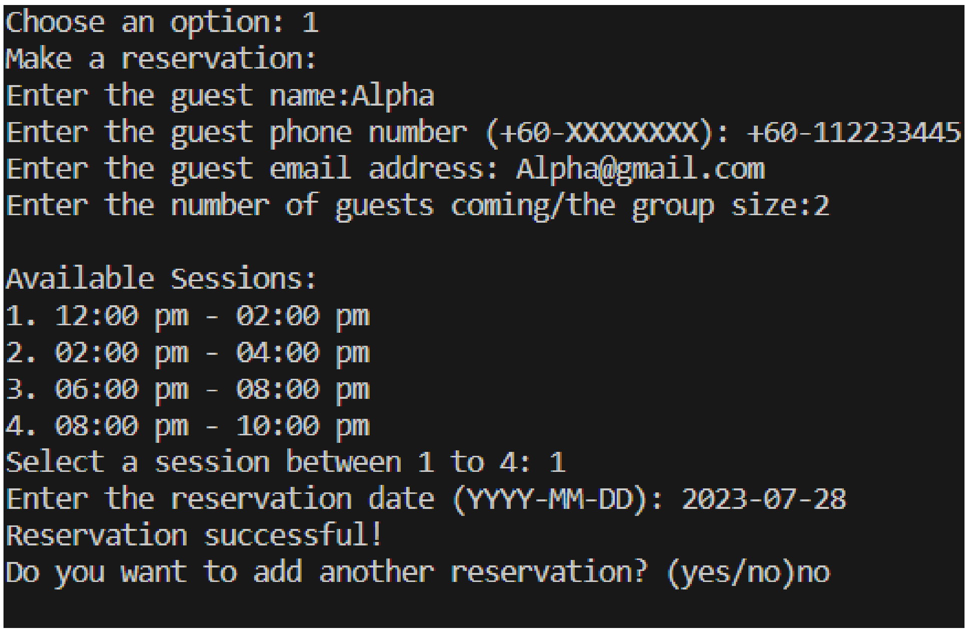
However, if the user enters anything other than “charming”, the code raises a **‘StopIteration’** exception using the **‘raise’** statement. This causes the program to exit the **‘main_menu’** loop abruptly and stops the program's execution.

Sample Test Cases and Outputs



```
Main Menu:
[1] Add Reservation
[2] Cancel Reservation
[3] Edit/Update Reservation
[4] Generate Menu Recommendations
[5] Display All Reservations
[6] Exit
Choose an option:
```

1. Add reservation – User enter ‘1’.

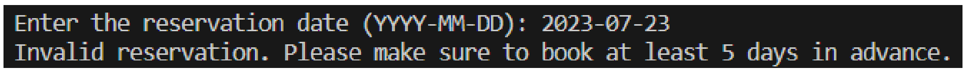


```
Choose an option: 1
Make a reservation:
Enter the guest name:Alpha
Enter the guest phone number (+60-XXXXXXXX): +60-112233445
Enter the guest email address: Alpha@gmail.com
Enter the number of guests coming/the group size:2

Available Sessions:
1. 12:00 pm - 02:00 pm
2. 02:00 pm - 04:00 pm
3. 06:00 pm - 08:00 pm
4. 08:00 pm - 10:00 pm
Select a session between 1 to 4: 1
Enter the reservation date (YYYY-MM-DD): 2023-07-28
Reservation successful!
Do you want to add another reservation? (yes/no)no
```

The output above shows when the user added a reservation that is within the booking conditions.

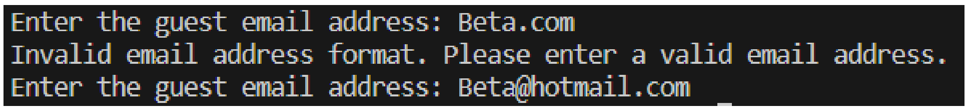
The outputs provided is the result of when a certain condition of a reservation is not met and how we provide a solution:

- Invalid Reservation Date: The program tells the user to only accept bookings that are 5 days ahead. It will then omit all the initial details of the customer and prompt the user to make a new booking instead.



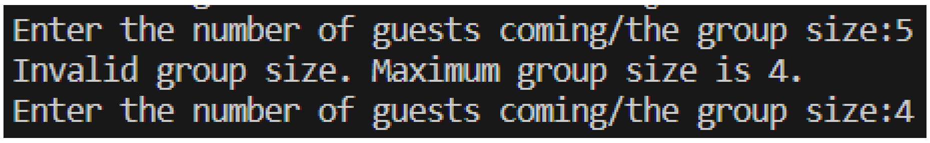
```
Enter the reservation date (YYYY-MM-DD): 2023-07-23
Invalid reservation. Please make sure to book at least 5 days in advance.
```

- Invalid Email Address: When an email address is invalid, the user must ask the guest to provide an email address that has at least one Atmar (@), and at least a period (.).



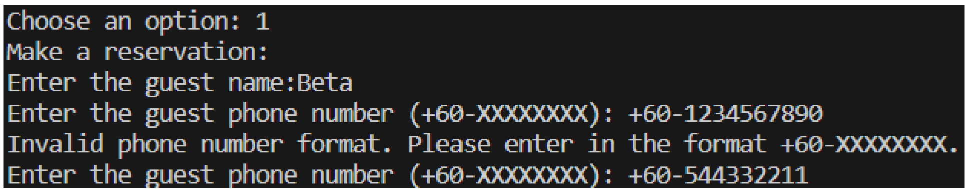
```
Enter the guest email address: Beta.com
Invalid email address format. Please enter a valid email address.
Enter the guest email address: Beta@hotmail.com
```

- Invalid Group Size: Ask the guest again for a valid group size when a maximum is exceeded. The maximum is 4 people per reservation as stated in the output.



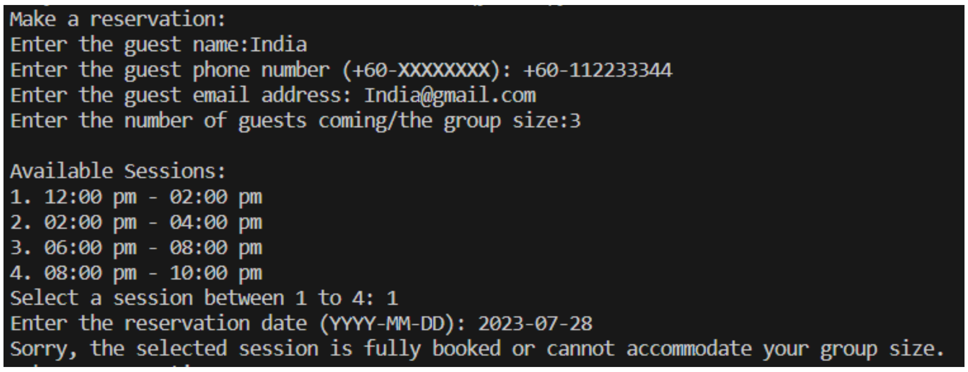
```
Enter the number of guests coming/the group size:5
Invalid group size. Maximum group size is 4.
Enter the number of guests coming/the group size:4
```

- Invalid Phone Number: The valid phone number is according to Malaysian phone number standard containing 9 digits total excluding the country code (+60). The user will have to ask the guest again for a valid phone number.



```
Choose an option: 1
Make a reservation:
Enter the guest name:Beta
Enter the guest phone number (+60-XXXXXXXX): +60-1234567890
Invalid phone number format. Please enter in the format +60-XXXXXXXX.
Enter the guest phone number (+60-XXXXXXXX): +60-544332211
```

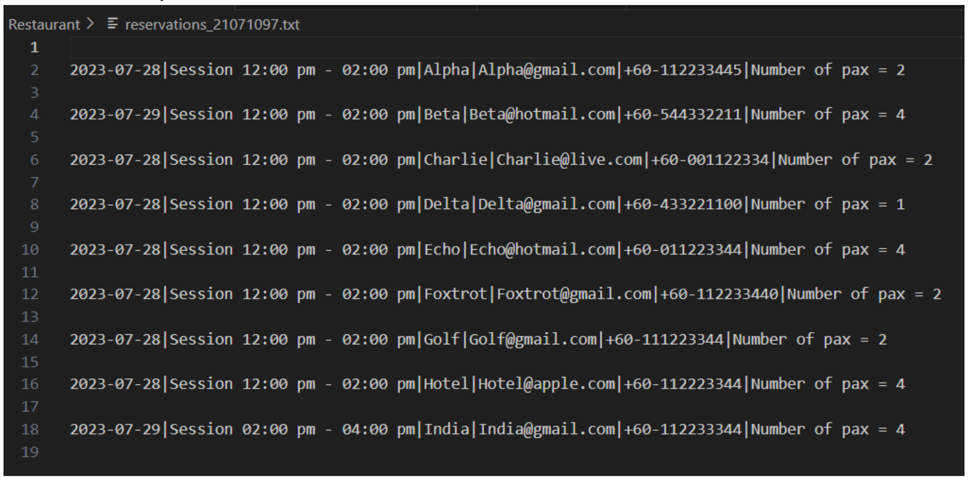
- Invalid Reservation Session: When a reservation is made on a session that has reached its maximum which is 8 reservations per session, the program will let the user know and prompts to make a new reservation.



```
Make a reservation:
Enter the guest name:India
Enter the guest phone number (+60-XXXXXXXX): +60-112233344
Enter the guest email address: India@gmail.com
Enter the number of guests coming/the group size:3

Available Sessions:
1. 12:00 pm - 02:00 pm
2. 02:00 pm - 04:00 pm
3. 06:00 pm - 08:00 pm
4. 08:00 pm - 10:00 pm
Select a session between 1 to 4: 1
Enter the reservation date (YYYY-MM-DD): 2023-07-28
Sorry, the selected session is fully booked or cannot accommodate your group size.
```

2. Cancel Reservation

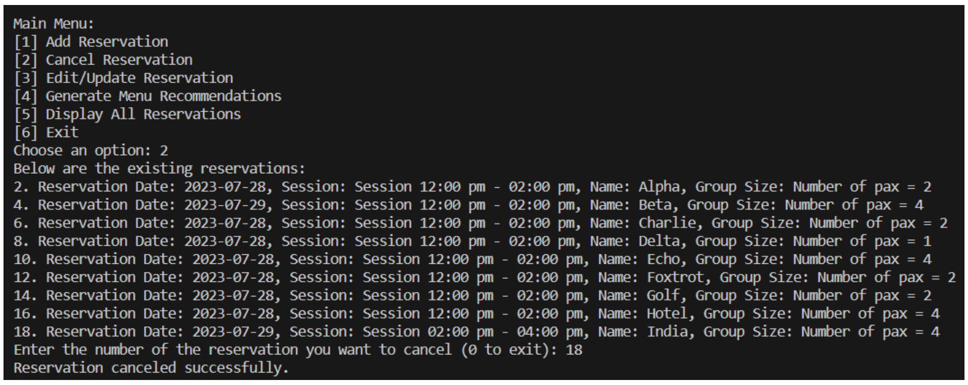
- The picture below shows the initial reservations in the text file for all reservations.



```
Restaurant > reservations_21071097.txt

2023-07-28|Session 12:00 pm - 02:00 pm|Alpha|Alpha@gmail.com|+60-112233445|Number of pax = 2

2023-07-29|Session 12:00 pm - 02:00 pm|Beta|Beta@hotmail.com|+60-544332211|Number of pax = 4

2023-07-28|Session 12:00 pm - 02:00 pm|Charlie|Charlie@live.com|+60-001122334|Number of pax = 2

2023-07-28|Session 12:00 pm - 02:00 pm|Delta|Delta@gmail.com|+60-433221100|Number of pax = 1

2023-07-28|Session 12:00 pm - 02:00 pm|Echo|Echo@hotmail.com|+60-011223344|Number of pax = 4

2023-07-28|Session 12:00 pm - 02:00 pm|Foxtrot|Foxtrot@gmail.com|+60-112233440|Number of pax = 2

2023-07-28|Session 12:00 pm - 02:00 pm|Golf|Golf@gmail.com|+60-111223344|Number of pax = 2

2023-07-28|Session 12:00 pm - 02:00 pm|Hotel|Hotel@apple.com|+60-112223344|Number of pax = 4

2023-07-29|Session 02:00 pm - 04:00 pm|India|India@gmail.com|+60-112233344|Number of pax = 4

```

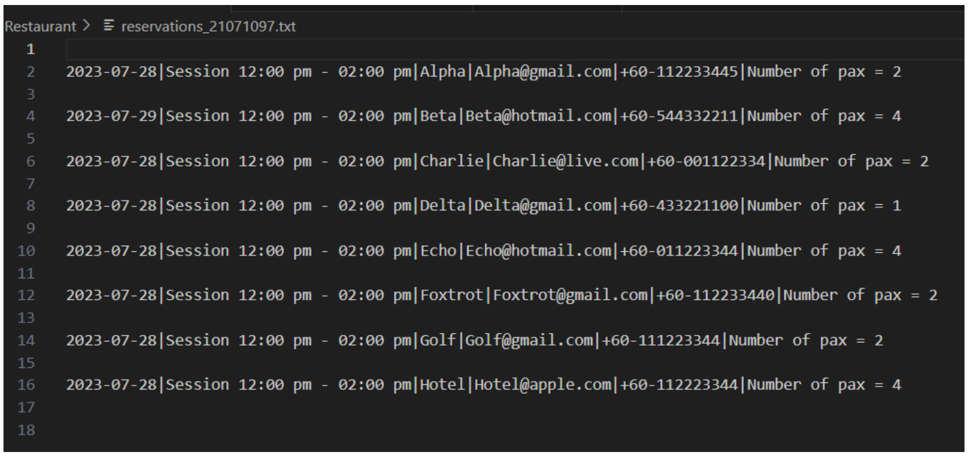
- When the user chooses option '2' at the main menu which is a flag to allow the process to cancel reservations, the program will prompt the user to choose a reservation to cancel by entering the number of the reservation as displayed.

In this case, the user chose reservation 18.



```
Main Menu:
[1] Add Reservation
[2] Cancel Reservation
[3] Edit/Update Reservation
[4] Generate Menu Recommendations
[5] Display All Reservations
[6] Exit
Choose an option: 2
Below are the existing reservations:
2. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Alpha, Group Size: Number of pax = 2
4. Reservation Date: 2023-07-29, Session: Session 12:00 pm - 02:00 pm, Name: Beta, Group Size: Number of pax = 4
6. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Charlie, Group Size: Number of pax = 2
8. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Delta, Group Size: Number of pax = 1
10. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Echo, Group Size: Number of pax = 4
12. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Foxtrot, Group Size: Number of pax = 2
14. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Golf, Group Size: Number of pax = 2
16. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Hotel, Group Size: Number of pax = 4
18. Reservation Date: 2023-07-29, Session: Session 02:00 pm - 04:00 pm, Name: India, Group Size: Number of pax = 4
Enter the number of the reservation you want to cancel (0 to exit): 18
Reservation canceled successfully.
```

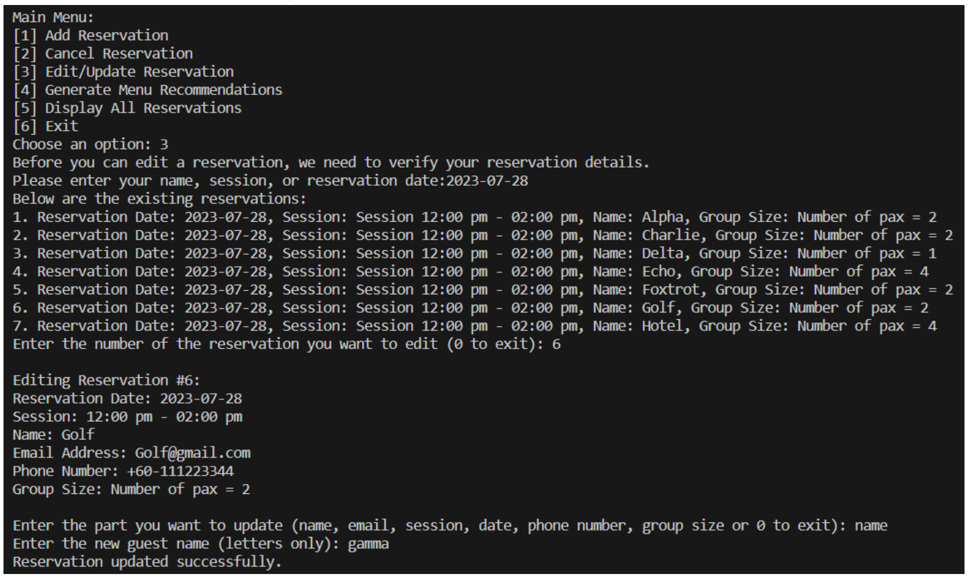
- In the picture below, the text file now shows that reservation 18 has been deleted from the file which shows that the program is successfully connected to the text file.



```
Restaurant > reservations_21071097.txt
1
2   2023-07-28|Session 12:00 pm - 02:00 pm|Alpha|Alpha@gmail.com|+60-112233445|Number of pax = 2
3
4   2023-07-29|Session 12:00 pm - 02:00 pm|Beta|Beta@hotmail.com|+60-544332211|Number of pax = 4
5
6   2023-07-28|Session 12:00 pm - 02:00 pm|Charlie|Charlie@live.com|+60-001122334|Number of pax = 2
7
8   2023-07-28|Session 12:00 pm - 02:00 pm|Delta|Delta@gmail.com|+60-433221100|Number of pax = 1
9
10  2023-07-28|Session 12:00 pm - 02:00 pm|Echo|Echo@hotmail.com|+60-011223344|Number of pax = 4
11
12  2023-07-28|Session 12:00 pm - 02:00 pm|Foxtrot|Foxtrot@gmail.com|+60-112233440|Number of pax = 2
13
14  2023-07-28|Session 12:00 pm - 02:00 pm|Golf|Golf@gmail.com|+60-111223344|Number of pax = 2
15
16  2023-07-28|Session 12:00 pm - 02:00 pm|Hotel|Hotel@apple.com|+60-112223344|Number of pax = 4
17
18
```

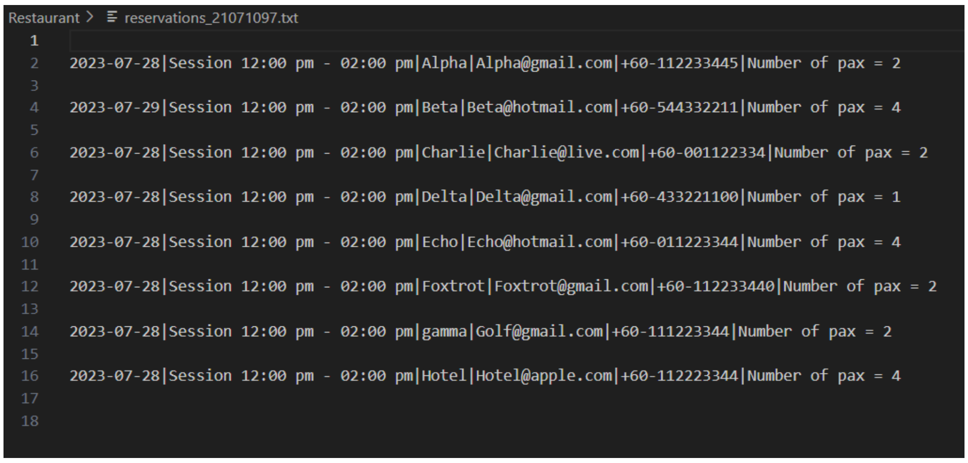
3. Update/Edit Reservation

- The picture below shows the user choosing option 3 which is to edit reservations. The program then asks the user to choose which reservation to edit similar to the operation to cancel a reservation. To improve accessibility, the chosen reservation is then displayed in a much easier format to understand so the user can choose which part of the reservation to edit.
- In this case, the user chose the name and changed the name of guest reservation 6 from ‘Golf’ to ‘gamma’.



```
Main Menu:
[1] Add Reservation
[2] Cancel Reservation
[3] Edit/Update Reservation
[4] Generate Menu Recommendations
[5] Display All Reservations
[6] Exit
Choose an option: 3
Before you can edit a reservation, we need to verify your reservation details.
Please enter your name, session, or reservation date:2023-07-28
Below are the existing reservations:
1. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Alpha, Group Size: Number of pax = 2
2. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Charlie, Group Size: Number of pax = 2
3. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Delta, Group Size: Number of pax = 1
4. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Echo, Group Size: Number of pax = 4
5. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Foxtrot, Group Size: Number of pax = 2
6. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Golf, Group Size: Number of pax = 2
7. Reservation Date: 2023-07-28, Session: Session 12:00 pm - 02:00 pm, Name: Hotel, Group Size: Number of pax = 4
Enter the number of the reservation you want to edit (0 to exit): 6

Editing Reservation #6:
Reservation Date: 2023-07-28
Session: 12:00 pm - 02:00 pm
Name: Golf
Email Address: Golf@gmail.com
Phone Number: +60-111223344
Group Size: Number of pax = 2

Enter the part you want to update (name, email, session, date, phone number, group size or 0 to exit): name
Enter the new guest name (letters only): gamma
Reservation updated successfully.
```

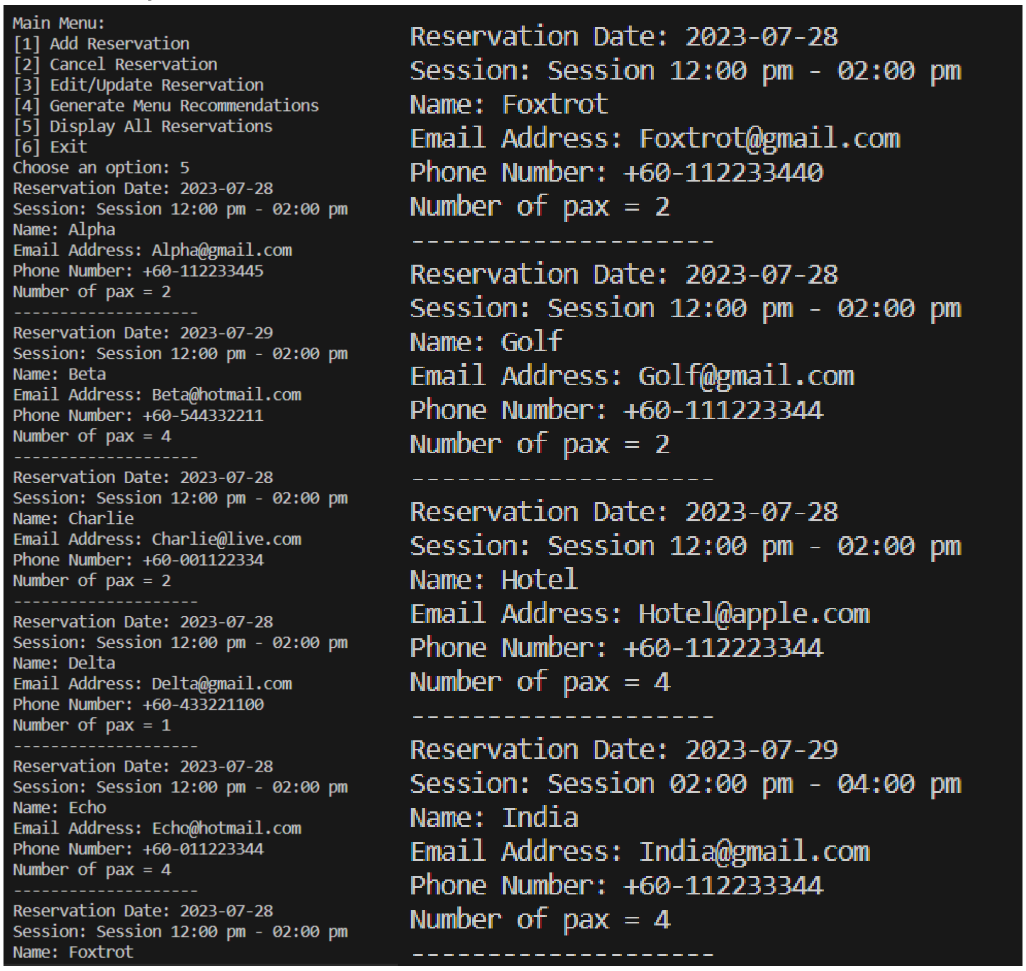
- Below the text file shows that the name 'Golf' changed to 'gamma' for reservation line 14.



```
Restaurant > reservations_21071097.txt
1
2   2023-07-28|Session 12:00 pm - 02:00 pm|Alpha|Alpha@gmail.com|+60-112233445|Number of pax = 2
3
4   2023-07-29|Session 12:00 pm - 02:00 pm|Beta|Beta@hotmail.com|+60-544332211|Number of pax = 4
5
6   2023-07-28|Session 12:00 pm - 02:00 pm|Charlie|Charlie@live.com|+60-001122334|Number of pax = 2
7
8   2023-07-28|Session 12:00 pm - 02:00 pm|Delta|Delta@gmail.com|+60-433221100|Number of pax = 1
9
10  2023-07-28|Session 12:00 pm - 02:00 pm|Echo|Echo@hotmail.com|+60-011223344|Number of pax = 4
11
12  2023-07-28|Session 12:00 pm - 02:00 pm|Foxtrot|Foxtrot@gmail.com|+60-112233440|Number of pax = 2
13
14  2023-07-28|Session 12:00 pm - 02:00 pm|gamma|Golf@gmail.com|+60-111223344|Number of pax = 2
15
16  2023-07-28|Session 12:00 pm - 02:00 pm|Hotel|Hotel@apple.com|+60-112223344|Number of pax = 4
17
18
```

4. Display Reservation

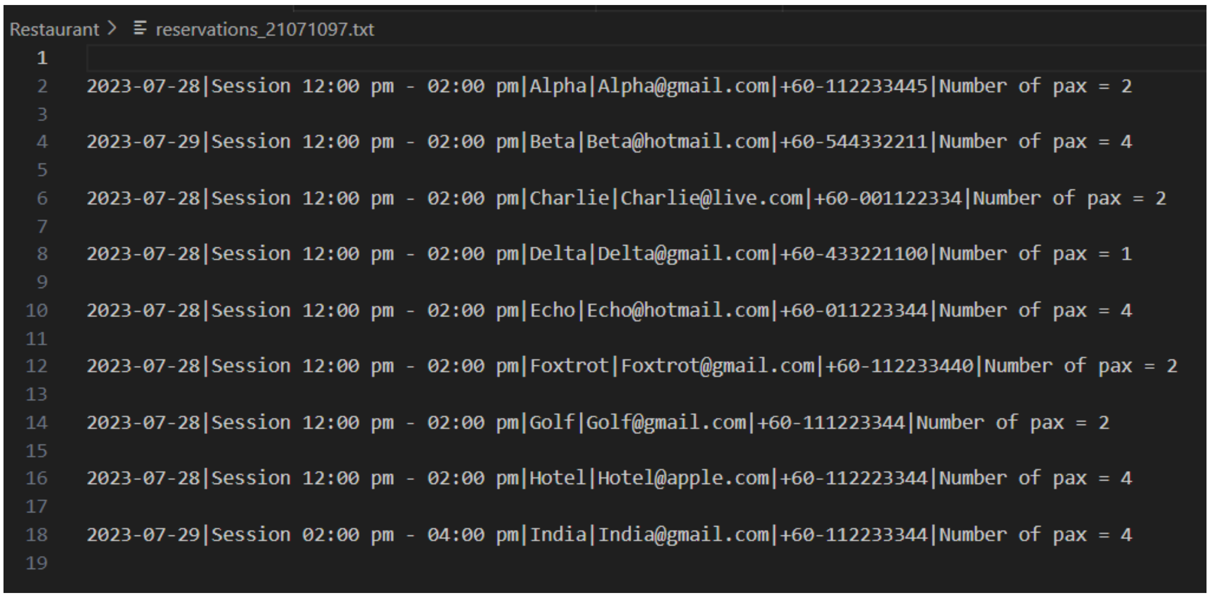
- To display all reservations, the user chooses option 4 on the main menu. The reservations are then displayed to the user in an easily understandable format so its easy for the user to take notice of the reservation's details.



```
Main Menu:
[1] Add Reservation
[2] Cancel Reservation
[3] Edit/Update Reservation
[4] Generate Menu Recommendations
[5] Display All Reservations
[6] Exit
Choose an option: 5
Reservation Date: 2023-07-28
Session: Session 12:00 pm - 02:00 pm
Name: Alpha
Email Address: Alpha@gmail.com
Phone Number: +60-112233445
Number of pax = 2
--------------------
Reservation Date: 2023-07-29
Session: Session 12:00 pm - 02:00 pm
Name: Beta
Email Address: Beta@hotmail.com
Phone Number: +60-544332211
Number of pax = 4
--------------------
Reservation Date: 2023-07-28
Session: Session 12:00 pm - 02:00 pm
Name: Charlie
Email Address: Charlie@live.com
Phone Number: +60-001122334
Number of pax = 2
--------------------
Reservation Date: 2023-07-28
Session: Session 12:00 pm - 02:00 pm
Name: Delta
Email Address: Delta@gmail.com
Phone Number: +60-433221100
Number of pax = 1
--------------------
Reservation Date: 2023-07-28
Session: Session 12:00 pm - 02:00 pm
Name: Echo
Email Address: Echo@hotmail.com
Phone Number: +60-011223344
Number of pax = 4
--------------------
Reservation Date: 2023-07-28
Session: Session 12:00 pm - 02:00 pm
Name: Foxtrot
```

```
Reservation Date: 2023-07-28
Session: Session 12:00 pm - 02:00 pm
Name: Foxtrot
Email Address: Foxtrot@gmail.com
Phone Number: +60-112233440
Number of pax = 2
--------------------
Reservation Date: 2023-07-28
Session: Session 12:00 pm - 02:00 pm
Name: Golf
Email Address: Golf@gmail.com
Phone Number: +60-111223344
Number of pax = 2
--------------------
Reservation Date: 2023-07-28
Session: Session 12:00 pm - 02:00 pm
Name: Hotel
Email Address: Hotel@apple.com
Phone Number: +60-112223344
Number of pax = 4
--------------------
Reservation Date: 2023-07-29
Session: Session 02:00 pm - 04:00 pm
Name: India
Email Address: India@gmail.com
Phone Number: +60-112233344
Number of pax = 4
--------------------
```

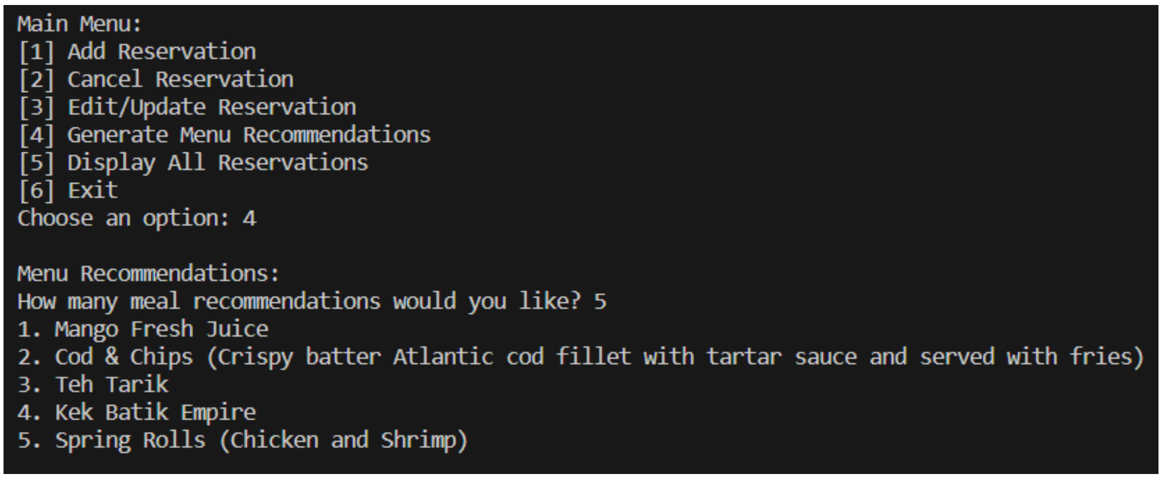
- In the text file below shows that the reservations displayed matched with all of the reservations in the text file.



```
Restaurant > reservations_21071097.txt

2023-07-28|Session 12:00 pm - 02:00 pm|Alpha|Alpha@gmail.com|+60-112233445|Number of pax = 2

2023-07-29|Session 12:00 pm - 02:00 pm|Beta|Beta@hotmail.com|+60-544332211|Number of pax = 4

2023-07-28|Session 12:00 pm - 02:00 pm|Charlie|Charlie@live.com|+60-001122334|Number of pax = 2

2023-07-28|Session 12:00 pm - 02:00 pm|Delta|Delta@gmail.com|+60-433221100|Number of pax = 1

2023-07-28|Session 12:00 pm - 02:00 pm|Echo|Echo@hotmail.com|+60-011223344|Number of pax = 4

2023-07-28|Session 12:00 pm - 02:00 pm|Foxtrot|Foxtrot@gmail.com|+60-112233440|Number of pax = 2

2023-07-28|Session 12:00 pm - 02:00 pm|Golf|Golf@gmail.com|+60-111223344|Number of pax = 2

2023-07-28|Session 12:00 pm - 02:00 pm|Hotel|Hotel@apple.com|+60-112223344|Number of pax = 4

2023-07-29|Session 02:00 pm - 04:00 pm|India|India@gmail.com|+60-112233344|Number of pax = 4

```

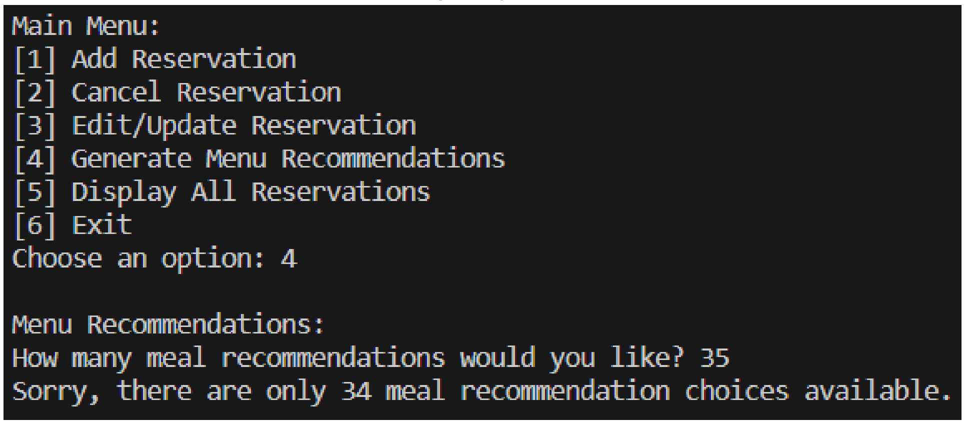
5. Generate Meal Recommendation

- To generate random meal recommendation when the guest asks for it, the user just has to choose option 5 and input the number of meal recommendations they would like.



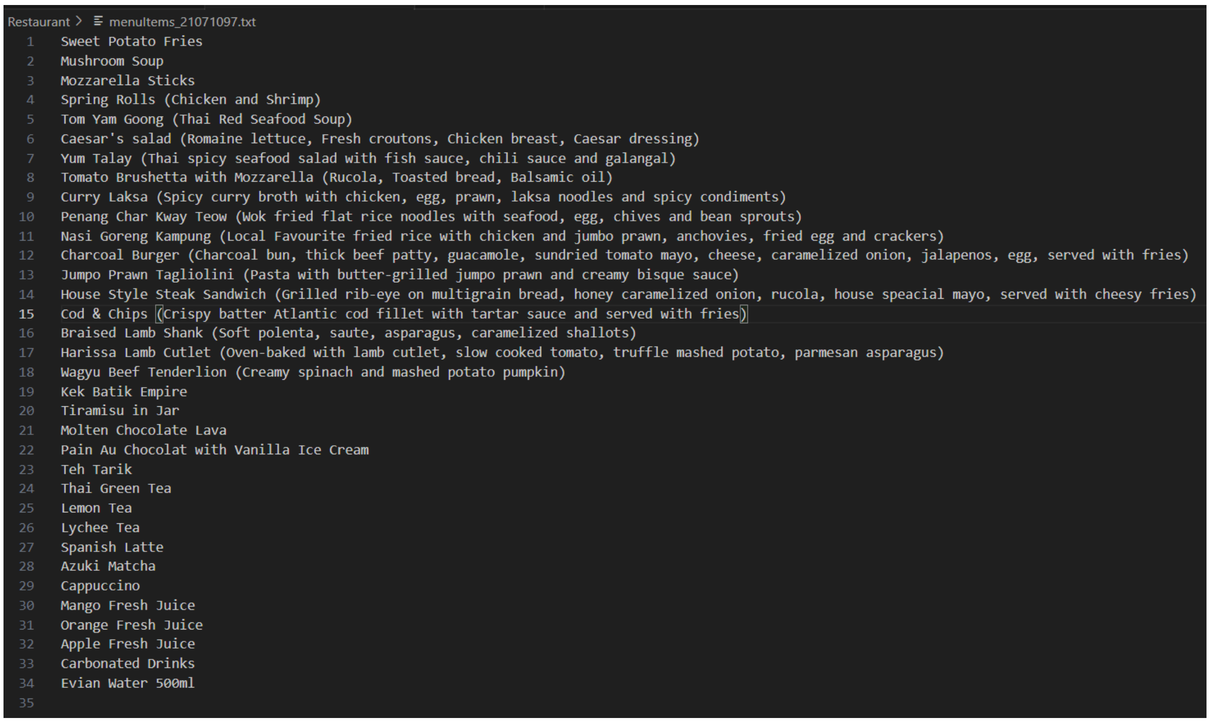
```
Main Menu:
[1] Add Reservation
[2] Cancel Reservation
[3] Edit/Update Reservation
[4] Generate Menu Recommendations
[5] Display All Reservations
[6] Exit
Choose an option: 4

Menu Recommendations:
How many meal recommendations would you like? 5
1. Mango Fresh Juice
2. Cod & Chips (Crispy batter Atlantic cod fillet with tartar sauce and served with fries)
3. Teh Tarik
4. Kek Batik Empire
5. Spring Rolls (Chicken and Shrimp)
```

- If the user input at the amount of meal recommendations is more than the available meal in the menu text file, the program will reject the input and ask the user to input a valid amount within the capacity of the menu, which in this case is 34.



```
Main Menu:
[1] Add Reservation
[2] Cancel Reservation
[3] Edit/Update Reservation
[4] Generate Menu Recommendations
[5] Display All Reservations
[6] Exit
Choose an option: 4

Menu Recommendations:
How many meal recommendations would you like? 35
Sorry, there are only 34 meal recommendation choices available.
```

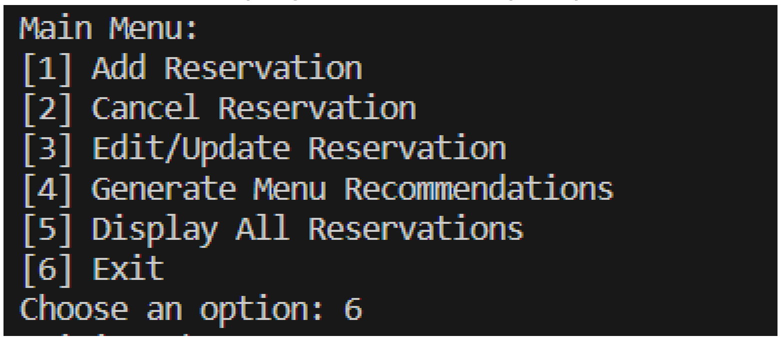
- The picture below is evidence that there are indeed only 34 items available in the menu. The restaurant may choose to add more items and the warning will match the item number to the item available in the text file.



```
Restaurant > menuItems_21071097.txt
 1  Sweet Potato Fries
 2  Mushroom Soup
 3  Mozzarella Sticks
 4  Spring Rolls (Chicken and Shrimp)
 5  Tom Yam Goong (Thai Red Seafood Soup)
 6  Caesar's salad (Romaine lettuce, Fresh croutons, Chicken breast, Caesar dressing)
 7  Yum Talay (Thai spicy seafood salad with fish sauce, chili sauce and galangal)
 8  Tomato Brushetta with Mozzarella (Rucola, Toasted bread, Balsamic oil)
 9  Curry Laksa (Spicy curry broth with chicken, egg, prawn, laksa noodles and spicy condiments)
10  Penang Char Kway Teow (Wok fried flat rice noodles with seafood, egg, chives and bean sprouts)
11  Nasi Goreng Kampung (Local Favourite fried rice with chicken and jumbo prawn, anchovies, fried egg and crackers)
12  Charcoal Burger (Charcoal bun, thick beef patty, guacamole, sundried tomato mayo, cheese, caramelized onion, jalapenos, egg, served with fries)
13  Jumpo Prawn Tagliolini (Pasta with butter-grilled jumpo prawn and creamy bisque sauce)
14  House Style Steak Sandwich (Grilled rib-eye on multigrain bread, honey caramelized onion, rucola, house speacial mayo, served with cheesy fries)
15  Cod & Chips (Crispy batter Atlantic cod fillet with tartar sauce and served with fries)
16  Braised Lamb Shank (Soft polenta, saute, asparagus, caramelized shallots)
17  Harissa Lamb Cutlet (Oven-baked with lamb cutlet, slow cooked tomato, truffle mashed potato, parmesan asparagus)
18  Wagyu Beef Tenderlion (Creamy spinach and mashed potato pumpkin)
19  Kek Batik Empire
20  Tiramisu in Jar
21  Molten Chocolate Lava
22  Pain Au Chocolat with Vanilla Ice Cream
23  Teh Tarik
24  Thai Green Tea
25  Lemon Tea
26  Lychee Tea
27  Spanish Latte
28  Azuki Matcha
29  Cappuccino
30  Mango Fresh Juice
31  Orange Fresh Juice
32  Apple Fresh Juice
33  Carbonated Drinks
34  Evian Water 500ml
35
```

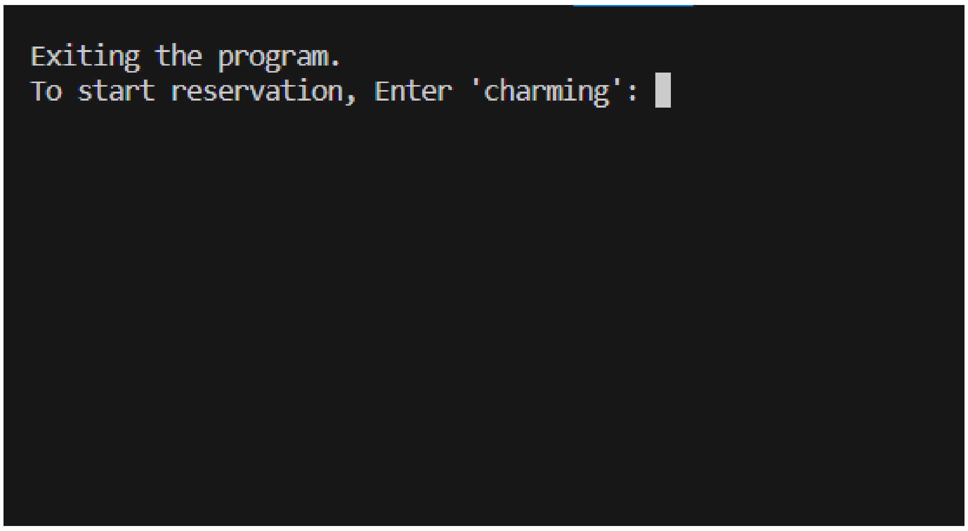
6. Exit

- To exit the program, the user input option 6.



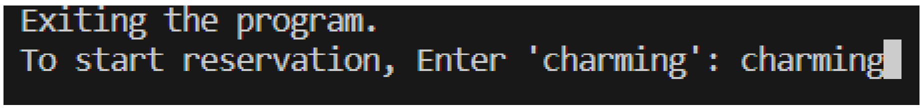
```
Main Menu:
[1] Add Reservation
[2] Cancel Reservation
[3] Edit/Update Reservation
[4] Generate Menu Recommendations
[5] Display All Reservations
[6] Exit
Choose an option: 6
```

- When option 6 is chosen, the terminal display will clear out showing that the reservation operation has be exited.



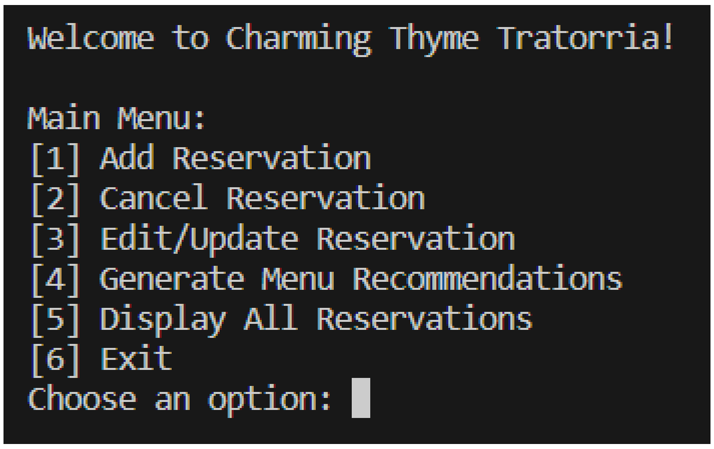
```
Exiting the program.
To start reservation, Enter 'charming':
```

- To start reservations again, the user may enter the word ‘charming’ or if the restaurant wishes to have security over the reservation system, they may replace the word ‘charming’ to a safe word and omit the word from printing to display.



```
Exiting the program.
To start reservation, Enter 'charming': charming
```

- When the program starts again, the terminal display will clear out again to ensure the user knows the program has just started.



```
Welcome to Charming Thyme Tratorria!

Main Menu:
[1] Add Reservation
[2] Cancel Reservation
[3] Edit/Update Reservation
[4] Generate Menu Recommendations
[5] Display All Reservations
[6] Exit
Choose an option: 
```

Python Code

```python
from datetime import datetime, timedelta
import re
import random
import os



class Reservations:
    def __init__(self):
        self.reservations = { # Initialize a dictionary to store reservations
for each session.
            '12:00 pm - 02:00 pm': [],
            '02:00 pm - 04:00 pm': [],
            '06:00 pm - 08:00 pm': [],
            '08:00 pm - 10:00 pm': []
        }
        self.reservation_dates = {}  # Store reservation dates by session


    def main_menu(self):
        # Display the main options and handle user's choice
        while True:
            print ("Welcome to Charming Thyme Tratorria!")
            print("\nMain Menu:")
            print("[1] Add Reservation")
            print("[2] Cancel Reservation")
            print("[3] Edit/Update Reservation")
            print("[4] Generate Menu Recommendations")
            print("[5] Display All Reservations")
            print("[6] Exit")

            choice = input("Choose an option: ")

            if choice == "1":
                self.make_reservation()
            elif choice == "2":
                self.cancel_reservation()
            elif choice == "3":
                self.edit_reservation()
            elif choice == "4":
                menu = Menu()
                menu.generate_recommendations()
            elif choice == "5":
                self.print_reservations()
            elif choice == "6":
                os.system('cls' if os.name == 'nt' else 'clear') # Clear the
terminal screen for a cleaner look
                print("Exiting the program.")
                choice = input("To start reservation, Enter 'charming': ") #
Enter safe word to start again
                if choice.lower() == "charming":
                    os.system('cls' if os.name == 'nt' else 'clear')  # Clear
the terminal screen for a cleaner look
                else:
                    raise StopIteration # Stop reservation process
            else:
                print("Invalid choice.")

    def make_reservation(self):
        """To make a reservation.
        Prompts the user for necessary information to make the reservation
valid and adds it to
        the reservations vocabulary."""
        while True:
            print("Make a reservation:")
            name = str(input("Enter the guest name:"))
            phone_number = self.get_valid_phone_number()  # prompts user to
enter a valid phone number
            email_address = self.get_valid_email_address()  # prompts user to
enter a valid email address


            while True:
                try:
                    group_size = int(input("Enter the number of guests
coming/the group size:"))
                    if self.is_valid_group_size(group_size):  # ensures only
4pax per table/reservation
                        break
                    else:
                        print("Invalid group size. Maximum group size is 4.")
                except ValueError:
                    print("Invalid input. Please enter a valid group size.")


            session = self.select_session()
            reservation_date = self.get_reservation_date()


            if not self.is_valid_reservation(session, reservation_date): #
error counter
                print("Invalid reservation. Please make sure to book at least
5 days in advance.")
```

```
                continue
            if not self.is_available(session, group_size): # error counter
                print("Sorry, the selected session is fully booked or cannot
accommodate your group size.")
                continue
# Add reservation details to the respective session's reservation list.
            self.reservations[session].append(
                (name, phone_number, email_address, group_size, session,
reservation_date))
            self.reservation_dates.setdefault(session,
[]).append(reservation_date)


            print("Reservation successful!")
            self.write_to_file(name, phone_number, email_address, group_size,
session, reservation_date)
            # append guest details to reservation text file
            choice = input("Do you want to add another reservation? (yes/no)")
            if choice.lower() != "yes":
                break


    def write_to_file(self, name, phone_number, email_address, group_size,
session, reservation_date, file_path =
'Restaurant/reservations_21071097.txt'):
        #Write the guest reservation details to the text file from add
reservations
        with open(file_path, "a") as file:
            file.write(f"\n{reservation_date}|Session
{session}|{name}|{email_address}|{phone_number}|Number of pax =
{group_size}\n")
            file.flush() # for fast response



    def get_valid_phone_number(self):
        """Ensures the phone number entered is valid and in the format of +60-
XXXXXXXXX
        (the Malaysian phone number format). Returns the valid phone number"""
        while True:
            phone_number = input("Enter the guest phone number (+60-XXXXXXXX):
")
            if re.match(r'^\+60-\d{9}$', phone_number):
                return phone_number
            else:
                print("Invalid phone number format. Please enter in the format
+60-XXXXXXXX.")
```

```python
    def get_valid_email_address(self):
        """Ensures the email address entered is valid then returns the email
address."""
        while True:
            email = input("Enter the guest email address: ")
            if re.match(r'^[\w\.-]+@[\w\.-]+\.\w+$', email):
                return email
            else:
                print("Invalid email address format. Please enter a valid
email address.")


    def select_session(self):
        """Method to prompt the user to select a session from the available
session options provided.
        Returns the selected session"""
        print("\nAvailable Sessions:")
        session_options = list(self.reservations.keys())
        for i, session in enumerate(session_options, start=1):
            print(f"{i}. {session}")
        while True:
            session_choice = input("Select a session between 1 to 4: ")
            if session_choice.isdigit() and 1 <= int(session_choice) <=
len(session_options):
                return session_options[int(session_choice) - 1]
            else:
                print("Invalid session choice.")


    def get_reservation_date(self):
        """Prompts the user to enter a date in the format of YYYY-MM-DD.
        Returns the parsed reservation date."""
        while True:
            try:
                date_input = input("Enter the reservation date (YYYY-MM-DD):
")
                reservation_date = datetime.strptime(date_input,
"%Y-%m-%d").date()
                return reservation_date
            except ValueError:
                print("Invalid date format. Please enter in the format YYYY-
MM-DD.")


    def is_valid_reservation(self, session, reservation_date):
        """Checks if the reservation is valid.
        Returns True if the reservation is valid (at least 5 days in advance),
False otherwise."""
        current_date = datetime.now().date()
        min_reservation_date = current_date + timedelta(days=5)
        return reservation_date >= min_reservation_date


    def is_valid_group_size(self, group_size):
        """Checks if the group size is valid (maximum 4 guests per table).
        Returns True if the group size is within the allowed limit, False
otherwise."""
        return group_size <= 4


    def is_available(self, session, group_size):
        """Checks if the selected session is available for the given group
size and
        does not exceed the maximum number of reservations.
        Returns True if the session is available, False otherwise."""
        session_reservations = self.reservations[session]
        if len(session_reservations) >= 8:  # Maximum of 8 reservations per
session
            return False
        total_group_size = sum(group_size for _, _, _, group_size, _, _ in
session_reservations)
        return total_group_size + group_size <= 4 * 8 - 1



    def print_reservations(self,
file_path='Restaurant/reservations_21071097.txt'):
        # Prints the existing reservations from the text file in an easily
readable format.
        try:
            with open(file_path, "r") as file:
                content = file.readlines()


            if not content:
                print("No reservations found.")
            else:
                for line in content:
                    reservation_details = line.strip().split("|")
                    if len(reservation_details) == 6:
                        reservation_date, session, name, email_address,
phone_number, group_size = reservation_details
                        print(f"Reservation Date: {reservation_date}")
                        print(f"Session: {session}")
                    print(f"Name: {name}")
                    print(f"Email Address: {email_address}")
                    print(f"Phone Number: {phone_number}")
                    print(f"{group_size}")
                    print("-" * 20)
        except FileNotFoundError:
            print(f"Error:{file_path} file not found.")


    def cancel_reservation(self,
file_path='Restaurant/reservations_21071097.txt'):
        # For the user to cancel a reservation chosen from the text file
according to number of reservation
        try:
            with open(file_path, "r") as file:
                content = file.readlines()


            if not content:
                print("No reservations found.")
                return


            print("Below are the existing reservations:")
            for i, line in enumerate(content, start=1):
                reservation_details = line.strip().split("|")
                if len(reservation_details) == 6:
                    reservation_date, session, name, email_address,
phone_number, group_size = reservation_details
                    print(f"{i}. Reservation Date: {reservation_date},
Session: {session}, Name: {name}, Group Size: {group_size}")


            while True:
                try:
                    choice = int(input("Enter the number of the reservation
you want to cancel (0 to exit): "))
                    if choice == 0:
                        print("Reservation cancellation--process aborted.")
                        return
                    elif 1 <= choice <= len(content):
                        break
                    else:
                        print("Invalid choice. Please enter a valid
reservation number.")
                except ValueError:
                    print("Invalid input. Please enter a valid reservation
number.")
```

```
            # Remove the cancelled reservation from text file
            with open(file_path, "w") as file:
                for i, line in enumerate(content, start=1):
                    if i != choice:
                        file.write(line)


            print("Reservation canceled successfully.")
            self.update_reservation_data()
        except FileNotFoundError:
            print(f"Error: {file_path} file not found.")


    def update_reservation_data(self,
file_path='Restaurant/reservations_21071097.txt'):
        # Update the in-memory reservations dictionary based on the latest the
text file.
        self.reservations = {
            '12:00 pm - 02:00 pm': [],
            '02:00 pm - 04:00 pm': [],
            '06:00 pm - 08:00 pm': [],
            '08:00 pm - 10:00 pm': []
        }
        try:
            with open(file_path, "r") as file:
                content = file.readlines()


            for line in content:
                reservation_details = line.strip().split("|")
                if len(reservation_details) == 6:
                    reservation_date, session, name, email_address,
phone_number, group_size = reservation_details
                    if session in self.reservations:
                        self.reservations[session].append((name, phone_number,
email_address, int(group_size), session, reservation_date))
        except FileNotFoundError:
            print(f"Error: {file_path} file not found.")



    def edit_reservation(self,
file_path='Restaurant/reservations_21071097.txt'):
        """Allows the user to edit a specific part of a reservation."""
        try:
            with open(file_path, "r") as file:
                content = file.readlines()

            if not content:
                print("No reservations found.")
                return
            print("Before you can edit a reservation, we need to verify your
reservation details.")
            check_value = input("Please enter your name, session, or
reservation date:")
            matching_reservations = [(i, line) for i, line in
enumerate(content) if check_value in line] # allow user to find reservations
according to details.
            if not matching_reservations:
                print("No matching reservations found.")
                return
            print("Below are the existing reservations:")
            for i,  (_, line) in enumerate(matching_reservations, start=1):
                reservation_details = line.strip().split("|")
                if len(reservation_details) == 6:
                    reservation_date, session, name, email_address,
phone_number, group_size = reservation_details
                    print(f"{i}. Reservation Date: {reservation_date},
Session: {session}, Name: {name}, Group Size: {group_size}")

            while True:
                try:
                    choice = int(input("Enter the number of the reservation
you want to edit (0 to exit): "))
                    if choice == 0:
                        print("Reservation editing--process aborted.")
                        return
                    elif 1 <= choice <= len(matching_reservations):
                        break
                    else:
                        print("Invalid choice. Please enter a valid
reservation number.")
                except ValueError:
                    print("Invalid input. Please enter a valid reservation
number.")

            # Get the chosen reservation's details.
            chosen_index, chosen_reservation = matching_reservations[choice -
1]
            reservation_details = chosen_reservation.strip().split("|")
            if len(reservation_details) == 6:
                reservation_date, session, name, email_address, phone_number,
group_size = reservation_details
                session = session.replace('Session ', '')
                print(f"\nEditing Reservation #{choice}:")
                print(f"Reservation Date: {reservation_date}")
                print(f"Session: {session}")
                print(f"Name: {name}")
                print(f"Email Address: {email_address}")
                print(f"Phone Number: {phone_number}")
                print(f"Group Size: {group_size}")
                    # Display reservation in a readable format to easily
choose a part to edit.
                while True:
                    update_choice = input("\nEnter the part you want to update
(name, email, session, date, phone number, group size or 0 to exit): ")
                    update_choice = update_choice.lower()
                    if update_choice == '0':
                        print("Reservation editing process cancelled.")
                        return
                    elif update_choice in ['name', 'email', 'session', 'date',
'phone number', 'group size']:
                        break
                    else:
                        print("Invalid choice. Please enter a valid option.")

                # Update the chosen detail with the new detail updated.
                if update_choice == 'name':
                    name = input("Enter the new guest name (letters only): ")
                elif update_choice == 'email':
                    email_address = input("Enter the new email address: ")
                elif update_choice == 'session':
                    session = self.select_session()
                elif update_choice == 'date':
                    reservation_date = self.get_reservation_date()
                elif update_choice == 'phone number':
                    phone_number = self.get_valid_phone_number()
                elif update_choice == 'group size':
                    group_size = input("Enter the new group size: ")
                    group_size = 'Number of pax = ' + group_size

                # Update the reservation details in the content list.
                content[chosen_index] = f"{reservation_date}|Session
{session}|{name}|{email_address}|{phone_number}|{group_size}\n"

                # Write the updated reservation details back to the file.
                with open(file_path, "w") as file:
                    file.writelines(content)

                print("Reservation updated successfully.")
                self.update_reservation_data()
        except FileNotFoundError:
            print(f"Error: {file_path} file not found.")


class Menu:
    def generate_recommendations(self):
        # Generate random menu recommendations from text file content list.
        try:
            with open("Restaurant/menuItems_21071097.txt", "r") as file:
                menu_items = file.readlines()

            print("\nMenu Recommendations:")
            num_recommendations = int(input("How many meal recommendations
would you like? "))
            if num_recommendations <= 0:
                print (f"Please enter a positive number.")
            elif num_recommendations > len(menu_items):
                print (f"Sorry, there are only {len(menu_items)} meal
recommendation choices available.") # error counter when meal recommendation >
meal available.
            else:
                recommendations = random.sample(menu_items,
num_recommendations)
                for i, item in enumerate(recommendations, start=1):
                    print(f"{i}. {item.strip()}")
        except FileNotFoundError:
            print("Error: 'menuItems_21071097.txt' file not found.")
if __name__ == "__main__":
    reservations = Reservations()
    reservations.main_menu()


reservations = Reservations()
reservations.make_reservation()
reservations.print_reservations()
```

<u>Sample Text File</u>

menuItems_21071097.txt

```
Sweet Potato Fries
Mushroom Soup
Mozzarella Sticks
Spring Rolls (Chicken and Shrimp)
Tom Yam Goong (Thai Red Seafood Soup)
Caesar's salad (Romaine lettuce, Fresh croutons, Chicken breast, Caesar dressing)
Yum Talay (Thai spicy seafood salad with fish sauce, chili sauce and galangal)
Tomato Brushetta with Mozzarella (Rucola, Toasted bread, Balsamic oil)
Curry Laksa (Spicy curry broth with chicken, egg, prawn, laksa noodles and spicy condiments)
Penang Char Kway Teow (Wok fried flat rice noodles with seafood, egg, chives and bean sprouts)
Nasi Goreng Kampung (Local Favourite fried rice with chicken and jumbo prawn, anchovies, fried egg and crackers)
Charcoal Burger (Charcoal bun, thick beef patty, guacamole, sundried tomato mayo, cheese, caramelized onion, jalapenos, egg, served with fries)
Jumpo Prawn Tagliolini (Pasta with butter-grilled jumpo prawn and creamy bisque sauce)
House Style Steak Sandwich (Grilled rib-eye on multigrain bread, honey caramelized onion, rucola, house speacial mayo, served with cheesy fries)
Cod & Chips (Crispy batter Atlantic cod fillet with tartar sauce and served with fries)
Braised Lamb Shank (Soft polenta, saute, asparagus, caramelized shallots)
Harissa Lamb Cutlet (Oven-baked with lamb cutlet, slow cooked tomato, truffle mashed potato, parmesan asparagus)
Wagyu Beef Tenderlion (Creamy spinach and mashed potato pumpkin)
Kek Batik Empire
Tiramisu in Jar
Molten Chocolate Lava
Pain Au Chocolat with Vanilla Ice Cream
Teh Tarik
Thai Green Tea
Lemon Tea
Lychee Tea
Spanish Latte
Azuki Matcha
Cappuccino
Mango Fresh Juice
Orange Fresh Juice
Apple Fresh Juice
Carbonated Drinks
Evian Water 500ml
```

reservations_21071097.txt

```
2023-07-28|Session 12:00 pm - 02:00 pm|Alpha|Alpha@gmail.com|+60-112233445|Number of pax = 2

2023-07-29|Session 12:00 pm - 02:00 pm|Beta|Beta@hotmail.com|+60-544332211|Number of pax = 4

2023-07-28|Session 12:00 pm - 02:00 pm|Charlie|Charlie@live.com|+60-001122334|Number of pax = 2

2023-07-28|Session 12:00 pm - 02:00 pm|Delta|Delta@gmail.com|+60-433221100|Number of pax = 1

2023-07-28|Session 12:00 pm - 02:00 pm|Echo|Echo@hotmail.com|+60-011223344|Number of pax = 4

2023-07-28|Session 12:00 pm - 02:00 pm|Foxtrot|Foxtrot@gmail.com|+60-112233440|Number of pax = 2

2023-07-28|Session 12:00 pm - 02:00 pm|gamma|Golf@gmail.com|+60-111223344|Number of pax = 2

2023-07-28|Session 12:00 pm - 02:00 pm|Hotel|Hotel@apple.com|+60-112223344|Number of pax = 4

2023-08-01|Session 06:00 pm - 08:00 pm|Juliet|Juliet@gmail.com|+60-112233445|Number of pax = 1

2023-07-27|Session 08:00 pm - 10:00 pm|Kilo|Kilo@hotmail.com|+60-135702468|Number of pax = 4

2023-08-01|Session 06:00 pm - 08:00 pm|Lima|Lima@apple-id.com|+60-086421357|Number of pax = 2

2023-07-30|Session 08:00 pm - 10:00 pm|Mike|Mike@magicmike.com|+60-136221320|Number of pax = 2

2023-07-31|Session 06:00 pm - 08:00 pm|November|Nova@november.com|+60-122376520|Number of pax = 1

2023-07-31|Session 02:00 pm - 04:00 pm|Oscar|Oscar@desert-co.com|+60-427639291|Number of pax = 3

2023-08-10|Session 08:00 pm - 10:00 pm|Papa|Papa@email.com|+60-283925439|Number of pax = 2

2023-08-15|Session 06:00 pm - 08:00 pm|Quebec|Quebec@imail.sunway.edu.com.my|+60-147267220|Number of pax = 4

2023-07-31|Session 08:00 pm - 10:00 pm|Romeo|RomeolovesJuliet@gmail.com|+60-132220879|Number of pax = 4

2023-08-02|Session 02:00 pm - 04:00 pm|Sierra|SierraSong@mail.com|+60-123870574|Number of pax = 2

2023-07-27|Session 06:00 pm - 08:00 pm|Tango|Tangolala@hotmail.com|+60-112222280|Number of pax = 2

2023-08-03|Session 02:00 pm - 04:00 pm|Uniform|UniformFIxed@help-desk.com|+60-668268900|Number of pax = 4

2023-07-31|Session 06:00 pm - 08:00 pm|Victor|Vict0r@gmail.com|+60-111103211|Number of pax = 2
```

Task List

No.	Assigned member	Date finished	Task	Description
1	Habiba	06/07/2023	Edit the initial draft of Python code	Continue editing based on the initial code focusing more on add, cancel and edit reservation. Study the uses of different modules that are allowed in the project.
2	Habiba	12/07/2023	Limitation Counter and Main Menu Code	Include coding for limitations within the reservation, for example; the limit of reserving 5 days-in-advance, 8 reservation per session and more. Further improve the accessibility of the Main Menu button where the user can be directed to certain functions.
3	Habiba	23/07/2023	Report Editing	Explain techniques used in the codes.
4	Hajah Atikah	20/07/2023	Menu recommendation Code	Create a menu list containing 34 initial items to generate from using imported module and make the choice to be readily utilized through the Main Menu. Create an error counter for users that want to generate too many menus' recommendations than what the restaurant has.
5	Hajah Atikah	21/07/2023	Add, Display, Cancel Reservation.	Improve the overall coding for the add, display and cancel reservation so it aligns with the content of their respective text file.

6	Hajah Atikah	23/07/2023	Report Editing	Provide different cases to be tested and present an output. Create the text file the project prompted. Proofread code comments.
7	Humayra	25/06/2023	Create Initial draft of Python code	Create an outline for how the entire coding flows with the use of class and user defined function. Requires the member to study the use of class and self-function more in depth.
8	Humayra	15/07/2023	Improve Coding for Reservation	Add or edit necessary coding to connect all the different functionality of reserving a slot (add, cancel and edit). Allows the whole code to take reservation on loop until the user wants to exit.
9	Humayra	22/07/2023	Report Editing	Create the Flow Chart of the overall Python code.
10	Habiba, Hajah Atikah, Humayra	24/07/2023	Report Proofread	Last session for editing and re-reading of the whole report. Importing necessary outputs, codes and finishing up on cover page and task list.