## Contents

# Restaurant Reservation and Ordering System Documentation

## Overview

This system is designed to manage restaurant reservations and orders via a conversational chatbot. The chatbot uses OpenAI's GPT-3.5-turbo model to interact with users, gather necessary information, and process orders and reservations. The system integrates with SQLite database to store and manage reservations and orders.

This high-level documentation provides an overview of the key components and functionality of the restaurant reservation and ordering system. For further details, refer to the system's code and implementation.

## Components

1. **Database**
    - **SQLite Database**: The system uses SQLite database named restaurant.db to store reservation and order information.
    - **Tables**: The primary table is reservations, which holds data such as reservation date, time, name, phone number, email address, number of guests, reservation type, address, and delivery time.
2. **Chatbot Integration**
    - **OpenAI GPT-3.5-turbo**: The chatbot uses OpenAI's GPT-3.5-turbo model to understand and respond to user queries.
    - **Function Descriptions**: The system defines several functions to handle different user requests, such as placing an order, making a reservation, canceling a reservation, and retrieving the menu. These functions are described in a way that the GPT model can call them when necessary.
3. **Functions and Features**
    - **Order Placement**: Users can place delivery orders, specifying items, customizations, address, and delivery time.
    - **Reservations**: Users can make dine-in reservations, providing details such as the number of guests and reservation time.

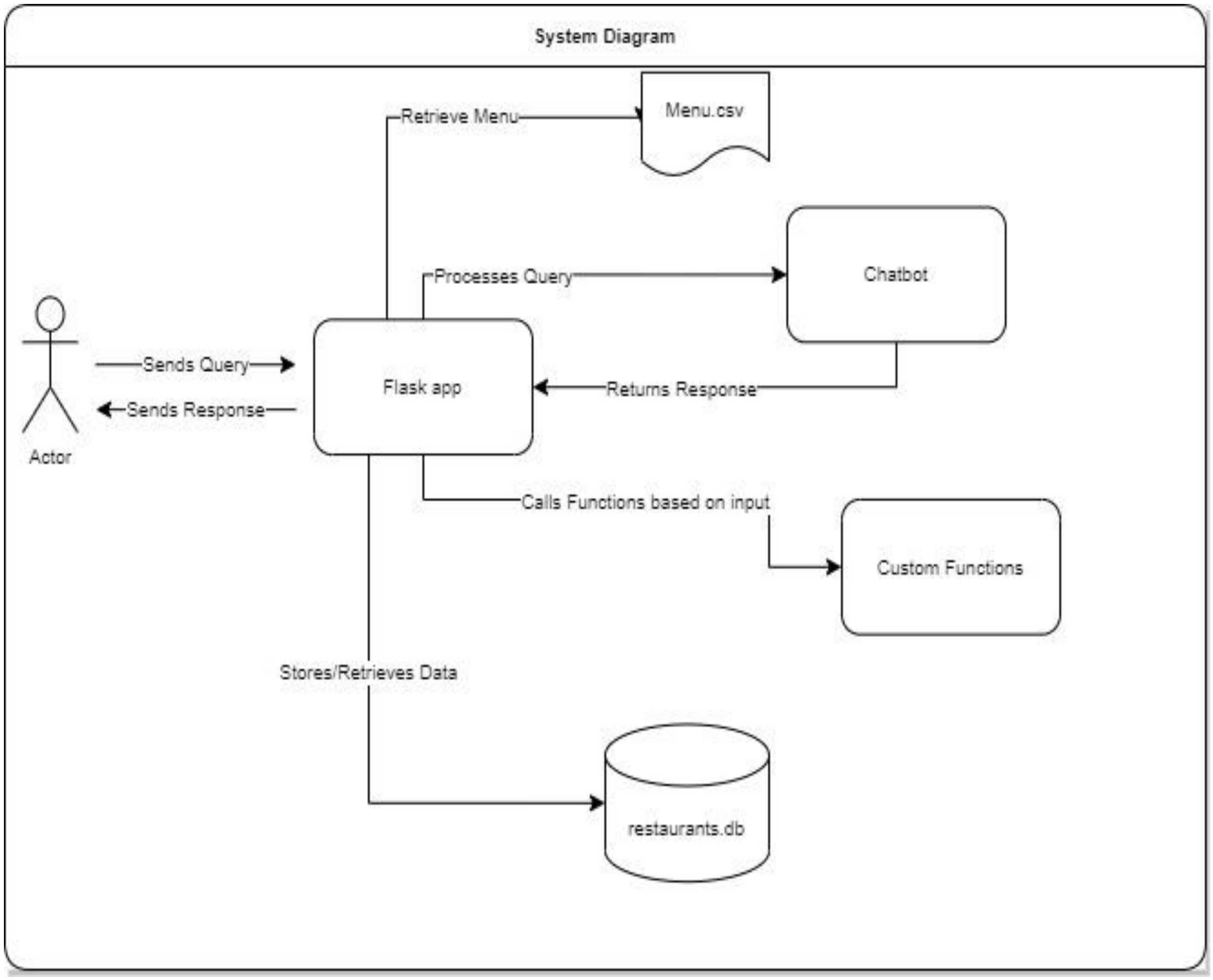
- **Cancellation**: Users can cancel existing reservations or orders by providing a reservation number or phone number.
- **Information Retrieval**: The system can provide information on operating hours, special offers, and the restaurant's location.
- **Menu Retrieval**: Users can request the menu, which is stored in a CSV file and categorized by sections.

4. **Utility Functions**
    - **Time Parsing and Validation:** Functions to parse and validate user-provided times, ensuring they fall within the restaurant's operating hours.
    - **Email Confirmation (Placeholder):** A placeholder function for sending email confirmations for reservations and orders (not fully implemented in the code).

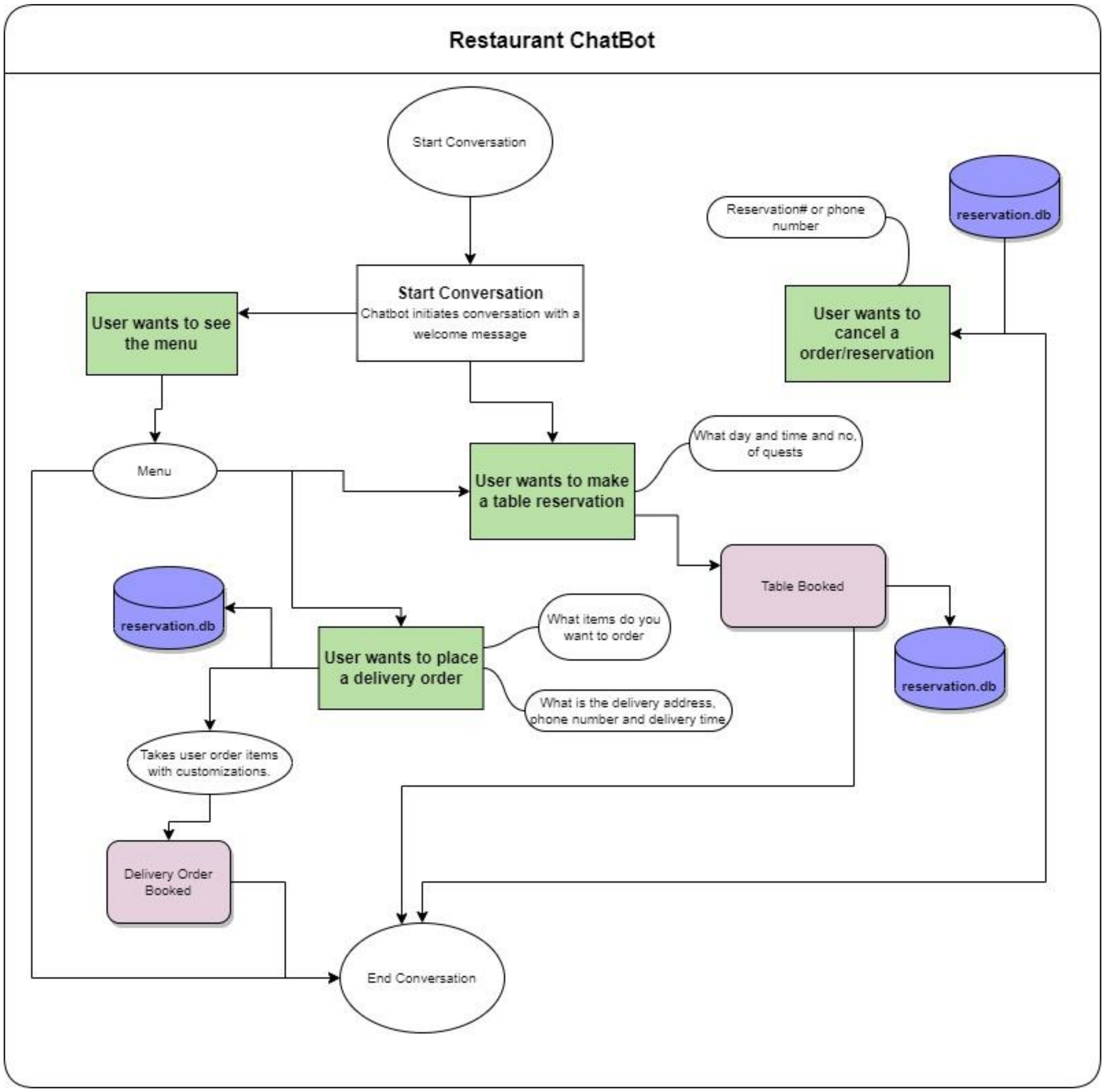
5. **Thread-Safe Database Connection**
    - **Thread-Safe Storage**: The system uses thread-local storage to manage database connections, ensuring that each thread uses its own connection to avoid conflicts.

6. **Deployment**
    - **Flask app –** The main application
    - **Start the flask web server** to make the application available.

7. **User Interaction Flow**
    - **Initialization:** The conversation starts with a system message that sets the context for the chatbot.
    - **User Input**: Users interact with the chatbot by typing their requests.
    - **Function Invocation**: Based on the user's input, the chatbot determines if a specific function should be called to handle the request.
    - **Response Generation:** The chatbot processes the request, calls the appropriate function if necessary, and generates a response for the user.
    - **Conversation Management:** The conversation history is maintained to provide context for ongoing interactions.


System Diagram
Retrieve Menu
Menu.csv
Processes Query
Chatbot
Sends Query
Sends Response
Actor
Flask app
Returns Response
Calls Functions based on input
Custom Functions
Stores/Retrieves Data
restaurants.db

## Processes


Restaurant ChatBot
Start Conversation
Start Conversation
Chatbot initiates conversation with a welcome message
User wants to see the menu
Menu
Reservation# or phone number
reservation.db
User wants to cancel a order/reservation
What day and time and no. of guests
User wants to make a table reservation
Table Booked
reservation.db
reservation.db
User wants to place a delivery order
What items do you want to order
What is the delivery address, phone number and delivery time
Takes user order items with customizations.
Delivery Order Booked
End Conversation

1. **Placing an Order**
    - User specifies items, address, and delivery time.
    - System validates the delivery time and processes the order.

  - Order details are stored in the database, and a confirmation message with an order number is provided.

2. **Making a Reservation**
   - User provides details such as date, time, number of guests, and contact information.
   - System stores the reservation details in the database.
   - A confirmation message with a reservation number is provided.

3. **Canceling a Reservation**
   - User provides a reservation number or phone number.
   - System checks the database and deletes the corresponding reservation.
   - A cancellation confirmation message is provided.

4. **Retrieving Information**
   - User requests information such as operating hours, special offers, or location.
   - System retrieves and provides the requested information.

## Restaurant Business Rules and Considerations

- **Operating Hours**: The restaurant operates from 10 AM to 10 PM daily. Delivery orders can be placed between 10 AM and 7:30 PM.
- **Delivery Time Validation**: Delivery times must be at least 30 minutes after the order is placed and within the operating hours.
- **Menu Management:** The menu is stored in a CSV file, and the chatbot ensures that only available items are ordered.
- **Thread Safety**: The use of thread-local storage for database connections ensures that the system can handle concurrent user interactions without conflicts.

## Challenges

- Learning Graph was high. It took me a week at least to get the model working.
- Functions invocation was difficult to implement.
- There is still lot of enhancements required on this chatbot.
- I am still not able to figure out how to show formatted menu.
- Sometimes the chatbot hallucinates on the menu. Not confident enough to handle that.

## Open Points

- Moderator module has not been incorporated.
- Time is not local. Also 12/24 AM/PM time format is not reliable.
- No module to check if the item is the menu or not. It is relying on the instructions provided in the prompts.