

Chatbot for the University of Poppleton

Project Overview

This Python script implements a basic chatbot for a virtual assistant at the University of Poppleton. The chatbot can respond to user inputs based on predefined responses stored in a JSON file and includes features such as memory logging, random disconnections, and dynamic responses. The chatbot's behavior includes handling user interactions, logging those interactions, and providing responses based on user input or random replies when no direct match is found.

Functions:

1. `read_json_file()`

- Description: Reads the `responses.json` file containing predefined responses for the chatbot. The responses are organized by categories and keywords.
- Returns: A dictionary containing responses categorized by keywords.
- Errors Handled: If the `responses.json` file is not found, an error message is displayed and an empty dictionary is returned.
- Usage Example:

```
responses = read_json_file()
```

2. `memory(log_file, sender, message)`

- Description: Logs the interaction between the user and the chatbot (sender and message) to a log file.
- Parameters:
 - `log_file`: Path to the log file where interactions are stored.
 - `sender`: The sender of the message (either user or chatbot).
 - `message`: The message being logged.
- Usage Example:

```
memory("chat_log.txt", "User", "Hello there!")
```

3. `random_disconnect(interaction_count, max_interactions, user_name, agent_name)`

- Description: Simulates a random disconnection after a certain number of interactions. The disconnection is triggered when the interaction count reaches the max interaction limit.

- Parameters:

- `interaction_count`: The current count of interactions.
- `max_interactions`: The maximum number of interactions before a disconnection occurs.
- `user_name`: The name of the user.
- `agent_name`: The name of the chatbot.

- Returns: `True` if a disconnection is triggered, `False` otherwise.

- **Usage Example:**

```
disconnected = random_disconnect(5, 10, "John", "Lia")
```

4. `find_response(user_input, responses)`

- Description: Finds a response based on the user's input. The response is looked up in the `responses` dictionary, which contains categories of keywords and their corresponding responses.

- Parameters:

- `user_input`: The input provided by the user (lowercase).
 - `responses`: A dictionary containing predefined responses.
- Returns: The corresponding response if found; otherwise, `None`.

- **Usage Example:**

```
response = find_response("What is the campus schedule?", responses)
```

5. `get_random_response()`

- Description: Returns a random response from a predefined list of general responses.

- Returns: A string containing a random response.

- **Usage Example:**

```
random_response = get_random_response()
```

6. `start_chatbot()`

- **Description:** Starts the chatbot interaction session. It handles user input, logs interactions, checks for disconnections, and responds to the user's queries.

- **Workflow:**

- Prompts the user for their name.

- Randomly selects a chatbot agent name from a list.

- Starts a session, logs the conversation in a log file.

- Responds to user input either with a predefined response or a random one.

- If a disconnection occurs (based on a random count of interactions), the session ends.

- **Usage Example:**

- start_chatbot()

File Structure:

- `responses.json`: A JSON file that stores predefined responses for different categories and keywords.

- **Example structure:**

```
json
{
  "greetings": {
    "hello": "Hi there! How can I help you?",
    "hi": "Hello! Welcome to the University of Poppleton."
  },
  "campus_info": {
    "schedule": "You can find the campus schedule on our website."
  }
}
```

- `chat_log.txt`: A dynamically generated log file that stores the conversation between the user and the chatbot.

Flow of Execution:

1. User Initiation:

- The chatbot asks for the user's name.
- A random agent name is selected from a list of names like "Lia", "Alex", "Jake", "Sally", or "Jay".
- The session is logged in a file named after the user.

2. Interaction:

- The chatbot listens for user inputs. If a matching response is found in the `responses.json` file, it is displayed.
- If no match is found, a random response is chosen from a predefined list.

3. Disconnection Simulation:

- After a random number of interactions (between 5 and 10), the chatbot simulates a disconnection with a random message.
- The conversation is logged, and the session ends.

4. Exit:

- If the user inputs "bye", "quit", or "exit", the chatbot sends a goodbye message and ends the session.

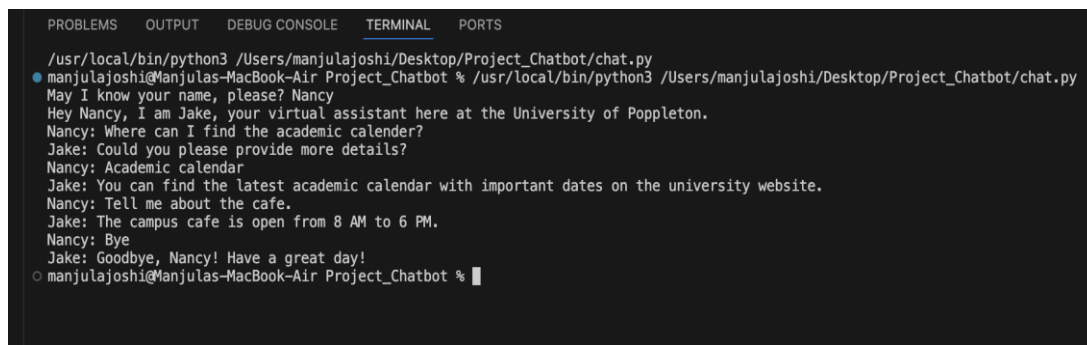
Error Handling:

- If the `responses.json` file is not found, an error message is displayed, and the chatbot continues with empty responses.
- If the user enters any input that doesn't match predefined keywords, a random response is given.

Dependencies:

- `random`: For generating random selections.
- `json`: For reading and writing JSON files.

Example Run:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
/usr/local/bin/python3 /Users/manjulajoshi/Desktop/Project_Chatbot/chat.py
manjulajoshi@Manjulas-MacBook-Air Project_Chatbot % /usr/local/bin/python3 /Users/manjulajoshi/Desktop/Project_Chatbot/chat.py
May I know your name, please? Nancy
Hey Nancy, I am Jake, your virtual assistant here at the University of Poppleton.
Nancy: Where can I find the academic calender?
Jake: Could you please provide more details?
Nancy: Academic calendar
Jake: You can find the latest academic calendar with important dates on the university website.
Nancy: Tell me about the cafe.
Jake: The campus cafe is open from 8 AM to 6 PM.
Nancy: Bye
Jake: Goodbye, Nancy! Have a great day!
manjulajoshi@Manjulas-MacBook-Air Project_Chatbot %
```

This documentation provides a comprehensive overview of the chatbot code's functionality, structure, and how it operates.

Submitted by: Manjula Joshi

University ID : 77576093