Wine Business Chatbot

The Wine Business Chatbot is a web-based chatbot application designed to assist users with inquiries related to the wine business. The chatbot is built using Flask for the backend and a simple HTML/CSS interface for the frontend.

Features

- User-friendly chat interface.

- Backend powered by Flask.

- Real-time responses to user queries.

- Modern, responsive design

Technologies Used

- Python: For the backend logic and server-side operations.
- Flask: A lightweight WSGI web application framework used for handling HTTP requests and responses.
- HTML/CSS: For structuring and styling the chat interface.
- JavaScript: For handling user interactions and sending requests to the server.

Setup and Installation

Prerequisites

- Python 3.x
- Flask

Installation

1. Clone the repository:

   ```bash

   git clone https://github.com/yourusername/wine-business-chatbot.git

   ```

2. Navigate to the project directory:

   ```bash

   cd wine-business-chatbot

   ```

3. Create a virtual environment:

   ```bash

   python -m venv venv

   ```

4. Activate the virtual environment:

- On Windows:

  ```bash
  venv\Scripts\activate
  ```

- On macOS/Linux:

  ```bash
  source venv/bin/activate
  ```

5. Install the required packages:

   ```bash
   pip install -r requirements.txt
   ```

6. Run the application:

   ```bash
   python app.py
   ```

7. Open your web browser and go to `http://127.0.0.1:5000`.

Overall Approach

1. Backend Setup: We used Flask to create a simple backend server that listens for incoming requests from the client-side chat interface.

2. Frontend Setup: The frontend was built using HTML and CSS for the structure and styling, respectively. JavaScript was used to handle user input and interact with the backend.

3. Integration: The backend and frontend were integrated to handle user queries and provide real-time responses.

Frameworks/Libraries/Tools Used

- Flask: Used for creating the backend server to handle HTTP requests and responses.

- Jinja2: Used for rendering HTML templates in Flask.

- Fetch API: Used in JavaScript to send asynchronous requests to the Flask backend.

-HTML/CSS: Used for creating and styling the chat interface.

- JavaScript: Used for handling user interactions and updating the chat interface dynamically.

Problems Faced and Solutions

1. Issue: The Flask server was not able to find the HTML template.
    - Solution: Ensured that the `templates` directory was correctly named and located in the same directory as `app.py`.
2. Issue: Styling inconsistencies across different browsers.
    - Solution: Used CSS reset and modern CSS techniques to ensure consistent styling across browsers.
3. Issue: Handling user input and displaying bot responses dynamically.
    - Solution: Used JavaScript to capture user input, send it to the Flask backend, and update the chat interface with the bot's responses.

Future Scope

-Natural Language Processing (NLP): Integrate NLP techniques to make the chatbot understand and respond more naturally.

- User Authentication: Add user authentication to provide personalized experiences.

- Database Integration: Store user queries and responses for analysis and improvement of the chatbot.

- Enhanced UI/UX: Improve the chat interface with more advanced CSS and JavaScript techniques, making it more interactive and visually appealing.

- Voice Integration: Add voice input and output capabilities to make the chatbot more accessible.

Contributing

Contributions are welcome! Please feel free to submit a pull request or open an issue to suggest improvements or report bugs.