1. Introduction

Piebot is an intelligent chatbot designed to assist users in selecting and enrolling in various courses. Utilizing Dialogflow for natural language processing and a Flask web application for backend operations, Piebot efficiently collects user details and preferences, storing them in a MySQL database. The chatbot can also offer personalized recommendations based on user interactions.

2. Objectives

The primary objectives of Piebot are:

- To provide a seamless course selection and enrollment process.
- To collect and store user details such as name, email, phone number, and selected course.
- To offer personalized course recommendations based on user preferences.

3. System Architecture

Piebot consists of the following key components:

- Dialogflow: For understanding and processing user inputs.
- Flask Web Application: For handling backend operations.
- MySQL Database: For storing user details.

4. How Piebot Works

4.1 Interaction Flow

- User Inquiry: The user initiates a conversation with Piebot, expressing interest in enrolling in a course.
- Course Selection: Piebot presents a list of available courses using buttons for easy selection.
- User Details Collection: Upon selecting a course, the user is prompted to provide their name, email, and phone number.
- Data Storage: The collected details are stored in a MySQL database.
- Confirmation Message: Piebot confirms the successful enrollment of the user.

4.2 Detailed Process

- Course Selection Intent:
    - The *want.course* intent in Dialogflow is triggered by user phrases indicating interest in courses.
    - Piebot responds with buttons for each course using a custom payload.

- User Details Intent:
    - The *course_selection* intent is triggered when the user selects a course.
    - Piebot collects the user's name, email, and phone number and stores these details in the MySQL database.

5. Code Explanation

Imports and App Initialization:

```
from flask import Flask, request, jsonify
import mysql.connector
app = Flask(_name_)
```

Flask: Imported to create the web application instance.

request: Used to access incoming request data.

jsonify: Used to create JSON responses.

mysql.connector: Provides the connector to interact with MySQL.

MySQL Configuration:

```
db_config = {
    'user': 'root',
    'password': 'Principie!@123',
    'host': 'localhost',
    'database': 'users'
}
```

db_config: A dictionary holding MySQL database connection parameters such as user, password, host, and database name.

Database Connection Function:

```
def get_db_connection():
    return mysql.connector.connect(**db_config)
```

get_db_connection(): A helper function that establishes and returns a connection to the MySQL database using the configuration in db_config.

Route and Intent Handling:

```
@app.route('/', methods=['POST'])
def index():
    data = request.get_json()
    print("Received JSON data:", data)
 intent_name = data['queryResult']['intent']['displayName']


    print(f"Received intent: {intent_name}")


    if intent_name == 'course_selection':
        try:
            course_name = data['queryResult']['parameters']['course_name'][0]
            name = data['queryResult']['parameters']['cust_name']
            email = data['queryResult']['parameters']['cust_mail']
            phone_number = data['queryResult']['parameters']['cust_contact']


            print(f"Course Name: {course_name}, Name: {name}, Email: {email}, Phone Number:
{phone_number}")


            success = store_user_details(course_name, name, email, phone_number)
            if success:
                response = {
                    'fulfillmentText': f"User {name} with email {email} and phone {phone_number} enrolled in
{course_name} added successfully."
                }
            else:
                response = {
                    'fulfillmentText': "Failed to add user. Please try again later."
                }
        except KeyError as e:
            print(f"KeyError: {e}")
            response = {
```

```
            'fulfillmentText': f"Missing parameter: {e}. Please ensure all required information is provided."
        }
    return jsonify(response)
  else:
    response = {
      'fulfillmentText': "No matching intent handler found."
    }
    return jsonify(response)
```

@app.route('/', methods=['POST']): Defines a route for handling POST requests at the root URL (/).

data = request.get_json(): Parses the incoming JSON request data.

intent_name = data['queryResult']['intent']['displayName']: Extracts the intent name from the JSON data.

if intent_name == 'course_selection': Checks if the intent is course_selection.

Extracts parameters (course_name, name, email, phone_number) from the JSON data.

Calls store_user_details to insert the user details into the database.

Constructs a response message based on the success or failure of the database

operation.

Database Operation Function:

```
def store_user_details(course_name, name, email, phone_number):
  try:
    connection = get_db_connection()
    cursor = connection.cursor()
    query = "INSERT INTO user_details (course_name, name, email, phone_number) VALUES (%s, %s, %s, %s)"
    cursor.execute(query, (course_name, name, email, phone_number))
    connection.commit()
    cursor.close()
```

```
        connection.close()
        return True
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        return False
```

store_user_details(): Inserts user details into the user_details table in the database.

Establishes a connection using get_db_connection().

Executes an SQL INSERT query to add the user details.

Commits the transaction and closes the connection.

Returns True if the operation is successful, otherwise catches and prints any errors and returns False.

Run the Application:

```
if _name_ == "_main_":
    app.run(debug=True)
```

Starts the Flask application in debug mode.

7. Enhancements and Future Work

To improve Piebot, the following features can be added:

- Personalized Recommendations: Utilize user preferences and past interactions to suggest courses.
- User Authentication: Add authentication mechanisms for secure user data handling.
- Real-Time Notifications: Integrate with messaging services to send real-time alerts and updates.
- Feedback Mechanism: Allow users to provide feedback on courses and the chatbot itself to improve services.

8. Conclusion

Piebot demonstrates a practical implementation of a chatbot for course enrollment, leveraging modern technologies such as Dialogflow, Flask, and MySQL. It provides a user-friendly interface for course selection and effectively manages user data. With additional features and improvements, Piebot can be enhanced to offer even more value to users.

Github link - https://github.com/principie/PieBot