CHATBOT FOR STUDENT HELPDESK (RULE-BASED)

Project title: Chatbot for Student Helpdesk (Rule-based).

Project Details:

Intern Name:Thrisha sambari.

Submitted To: GKB Labs

Task: Task 2(SET B).

Development Environment: Google collab.

Project Description:

This chatbot system is designed to simulate a helpdesk assistant specifically tailored for educational institutions. By relying on keyword-based logic and conditional statements, it can interpret user input and respond appropriately based on a predefined set of rules.

The chatbot operates in the terminal/command-line environment and uses **simple string handling and if-else/match-case logic** to determine responses.

Problem Statement:

In most colleges and universities, students frequently encounter confusion or delays when trying to obtain basic information such as **fee deadlines, course offerings, faculty contact details, or academic schedules**. These queries are often repetitive and directed toward administrative staff or faculty, leading to inefficiencies and unnecessary workload.

Despite the growing use of technology in education, many institutions still lack a simple, automated system to handle such routine questions. Traditional solutions like physical notice boards or static websites are often outdated, difficult to navigate, or lack personalization.

There is a clear need for an **interactive, easy-to-use system** that can provide instant answers to student queries without the need for human intervention.

Solution:

To address the problem of repetitive student queries and reduce the workload on administrative and academic staff, we propose building a **console-based, rule-based chatbot** that functions as a virtual student helpdesk.

This chatbot will use **string matching and conditional logic (e.g., if-else or match-case)** to recognize specific keywords or patterns in user input and respond accordingly with predefined answers

Tools and technologies used:

Programming Language

- **Python 3.x**
    - Python is chosen for its simplicity, readability, and strong support for string handling and conditional logic, which are essential for a rule-based chatbot.

Development Environment

- **Console / Terminal**
    - The chatbot is designed as a **console-based application**, so no GUI framework or web interface is needed.
- **Code Editor / IDE**
        - Or any basic text editor(Google Editor).

Core Python Modules

- **No external libraries required**
    - The project runs using only built-in Python features like:
        - print(), input()
        - String methods (lower(), in, etc.)
        - Conditional statements (if-else, match-case if using Python 3.10+)
- **Optional: re module**
    - For basic pattern matching (regex), if more flexibility in keyword recognition is desired.

File Handling

- **Text File I/O**
    - Used for the **bonus feature**: saving chat logs using Python's built-in open() function.

Testing Tools (Manual)

- Manual testing using various user inputs in the console.
- No third-party testing libraries are required due to the simplicity of the project.

Version Control (Optional)

- **Git (optional)**
    - To track versions and changes during development, especially in team settings.

Requirements

- **Python 3.x** installed on your system
- Any **code editor or terminal** (like VS Code, IDLE, or command prompt)
- No external libraries needed (only built-in Python modules like input(), print())
- Optional: re module (for advanced keyword matching)

Function Descriptions

Function Name	Purpose
greeting()	Prints a welcome message when the chatbot starts.
get_response()	Takes user input and returns a response using rule-based logic.
save_chat_log()	Saves the chat conversation to a .txt file.
start_chat()	Starts the chatbot loop, takes input from user, and prints responses.

Script Execution Flow

1. The program starts and shows a greeting.
2. The user types a question (e.g., “When is the fee deadline?”).
3. The chatbot checks the input and gives the correct answer using if-else or match-case.
4. If input is not understood, a fallback message is shown.
5. User can type exit to end the chat.
6. Optionally, the user can save the chat log.

Example Usage:

You: What is the last date for fee payment?

Bot: The last date for fee payment is August 15, 2025.

You: Show faculty contacts for Computer Science

Bot: Dr. A. Rao: arao@college.edu

Prof. S. Mehta: smehta@college.edu

You: Save log

Bot: Chat log saved to 'chat_log.txt'

You: Exit

Bot: Goodbye! Have a great day!

Future Enhancements

- Add **GUI** using Tkinter or a web interface using Flask
- Connect with a **database** to fetch real-time information
- Improve input recognition using **regular expressions**
- Add support for **voice input/output**
- Build a **mobile version** of the chatbot