****

**PROGRAMMING FOR AI (LAB)**

**Lab Task No 10**

**Submitted To :- Sir Rasikh Ali**

**Submitted By :- Aiman Ijaz**

**Roll No :- SU92-BSAIM-F23-010**

**Department :- Software Engineering**

**Section :- BSAI-4A**

**Lab 10 task : Create any of the following Chatbot (GUI using Flask), develop its "Flask -> HTM:" based front-end**

**1. University Admission**

**2. Medical Center Information**

**3. Hospital Information**

**4. Hotel Information**

**5. Restaurant Information**

**6. Library Information**

**7. " Your Own Choice "**

**Medical Center Chatbot – Documentation**

**What Does It Do?**

This chatbot is a basic web-based assistant designed for a medical center. It interacts with users by answering frequently asked questions like appointment booking, doctor availability, visiting hours, emergency contact, and more. It simulates a human-like conversation using predefined patterns.

**How It Works – Step by Step**

**1. Libraries and Tools Used**

* **Flask**: A lightweight web framework used to serve the chatbot interface and handle user requests.
* **NLTK (Natural Language Toolkit)**: Used for natural language understanding and managing chatbot responses.
* **Random**: Used to randomly select one of the multiple possible replies to add variety in responses.

These libraries work together to create a simple, interactive chatbot system.

**2. Pattern-Based Chatbot Logic**

The chatbot is powered by predefined question-answer pairs using regular expressions. For example:

[r"what is your name?", ["I am a Medical Center Bot."]]

* The **left side** is a pattern the user might type (e.g., "What is your name?")
* The **right side** contains one or more responses the bot can return

This allows the chatbot to match user input with specific patterns and reply accordingly.

**3. Chat Object Creation**

Using the NLTK library, we create a Chat object:

chatbot = Chat(pairs, reflections)

* pairs is the list of questions and answers
* reflections is a helper that switches words like "I" to "you" for natural-sounding replies

This object handles matching the user input and generating appropriate responses.

**4. Backend Routing Using Flask**

Two main Flask routes are used:

* / route loads the chatbot webpage (HTML frontend).
* /get route handles user input and sends back the bot’s response using JSON.

This backend logic connects the frontend input box to the chatbot engine.

**5. Frontend Interaction**

The HTML page contains:

* A **chat window** to display conversation history
* An **input field** where the user types their message
* A **Send button** that triggers a JavaScript function

The JavaScript code:

1. Captures the user input.
2. Sends it to the /get route.
3. Receives the chatbot’s reply.
4. Updates the chat window dynamically without refreshing the page.

**6. Styling with CSS**

The chatbot interface is styled using:

* **Blue and grey background colors** to give a clean and calm appearance.
* **Black borders** to clearly separate messages and sections.
* Responsive and user-friendly layout for desktop and mobile views.

The styling improves the usability and makes the chat experience more engaging.

**Why This Approach?**

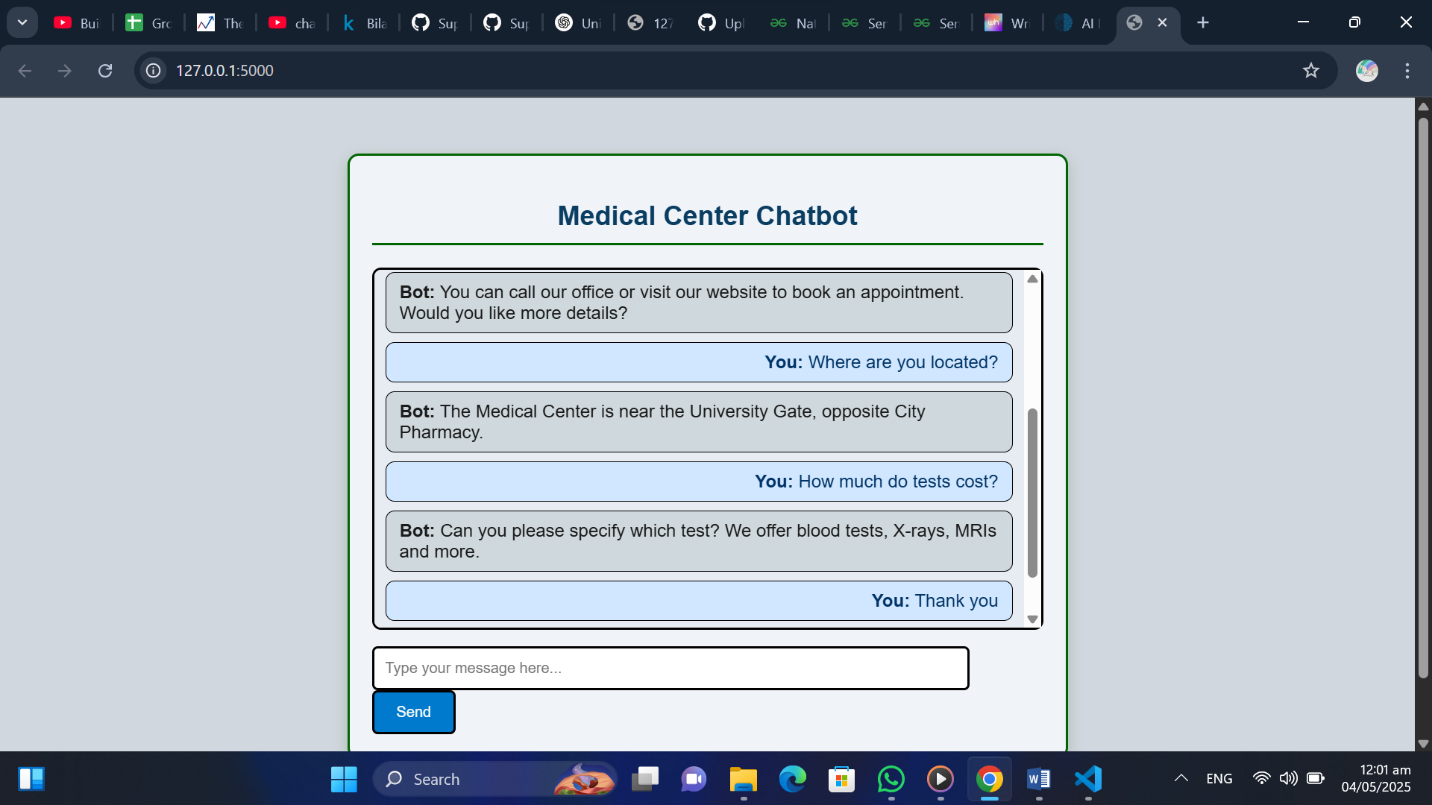
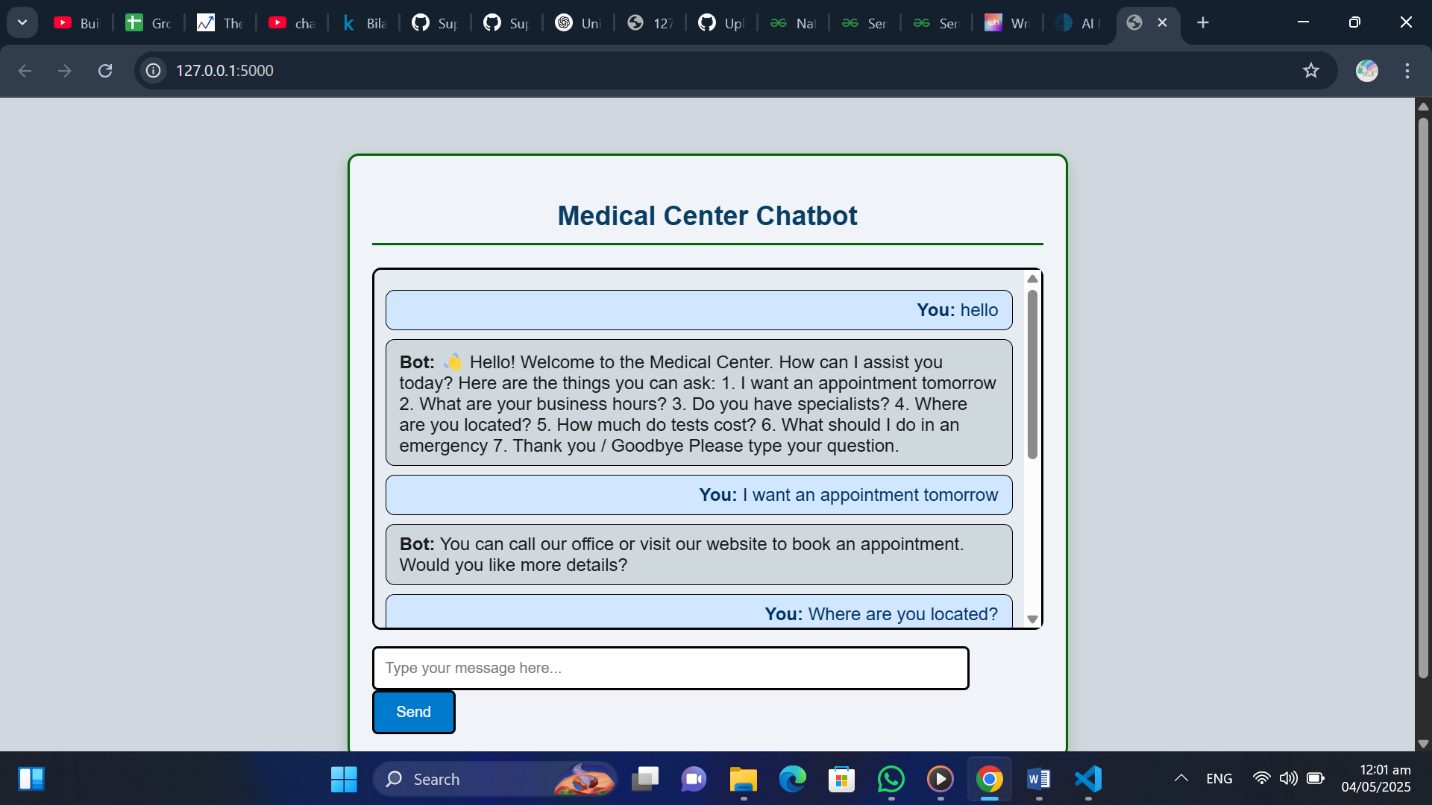
* **Simple and Effective**: Uses basic logic that’s easy to implement and understand.
* **No Training Required**: Since it’s rule-based, it doesn’t need machine learning or large datasets.
* **Easy to Customize**: New questions and answers can be added easily.
* **Real-Time Communication**: Thanks to JavaScript and Flask, responses are shown instantly.

**Conclusion**

This chatbot project demonstrates how to build a basic yet functional virtual assistant using Python, Flask, and NLTK. It teaches core concepts like:

* Backend and frontend communication
* Pattern matching with regular expressions
* Real-time interaction in a web application

**Output:**

****