

|  |
| --- |
| **RAJALAKSHMI INSTITUTE OF TECHNOLOGY** |
| (An Autonomous Institution, Affiliated to Anna University, Chennai) |

**DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)**

**ACADEMIC YEAR 2025 - 2026**

**SEMESTER III**

**ARTIFICIAL INTELLIGENCE LABORATORY**

**MINI PROJECT REPORT**

|  |  |
| --- | --- |
| **REGISTER NUMBER** | 2117240030048 |
| **NAME** | Inas Husayn K |
| **PROJECT TITLE** | AI Chat Bot |
| **DATE OF SUBMISSION** |  |
| **FACULTY IN-CHARGE** | **Mrs. M. Divya** |

**Signature of Faculty In-charge**

**🧠 INTRODUCTION**

**Brief Overview of Artificial Intelligence Concepts**

**Artificial Intelligence (AI) is the branch of computer science that focuses on creating intelligent systems capable of performing tasks that normally require human intelligence. These tasks include understanding natural language, recognizing patterns, making decisions, and learning from experience. AI applications are widely seen in speech recognition, recommendation systems, robotics, and conversational agents such as chatbots.**

**Topic Introduction and Background Context**

**A chatbot is an AI-based application that simulates human conversation through text or voice interactions. It processes user input, understands the context (in basic or advanced form), and responds accordingly. Chatbots have become an essential part of websites and mobile applications, offering instant responses, 24/7 availability, and efficient customer service.**

**Why the Problem Matters**

**With the increasing demand for digital communication, organizations and individuals often need simple chatbot systems to automate common queries. However, complex AI chatbots can be costly or difficult to implement. Hence, developing a lightweight, rule-based chatbot using Python Flask and web technologies offers a practical and educational solution for learning and small-scale automation.**

**Project Aim**

**This project aims to design and implement a simple web-based chatbot that can communicate with users in real-time, provide automated responses, and demonstrate client-server communication using Python Flask, HTML, CSS, and JavaScript.**

**⚙️ PROBLEM STATEMENT**

**To develop a simple, web-based chatbot system that can process user input, generate relevant text-based responses, and display them dynamically through an interactive web interface using Python (Flask) for backend processing and HTML/CSS/JS for frontend interaction.**

**🎯 GOAL**

**The main goal of the project is to build a functional chatbot interface that:**

* **Understands basic user queries using keyword-based logic.**
* **Provides real-time responses without page reloads.**
* **Demonstrates integration between frontend and backend technologies.**

**Expected Result:  
A fully functional chatbot application that can engage in simple text-based conversations and act as a prototype for intelligent communication systems.**

**📖 THEORETICAL BACKGROUND**

**1. About the Problem**

**Chatbots are conversational software systems designed to emulate human interaction. They can be divided into:**

* **Rule-Based Chatbots: Work on pre-defined rules, patterns, and keywords.**
* **AI-Based Chatbots: Use Natural Language Processing (NLP) and Machine Learning to understand intent.**

**This project uses the Rule-Based approach for simplicity, where responses are generated based on keyword matching within user messages.**

**2. Algorithm Background**

**The chatbot follows a simple pattern-matching algorithm.  
Steps involved:**

1. **Receive user input.**
2. **Convert input to lowercase for uniformity.**
3. **Match input with predefined keywords or phrases.**
4. **Return the corresponding response.**
5. **If no match is found, display a default fallback message.**

**3. Literature Survey**

* **Early chatbots like ELIZA (1966) used rule-based pattern matching for medical counseling simulation.**
* **ALICE (1995) introduced AIML (Artificial Intelligence Markup Language) for better pattern definition.**
* **Modern chatbots like Dialogflow or IBM Watson use AI and NLP models for advanced conversations.  
  For educational and introductory purposes, a rule-based chatbot remains simple, transparent, and easy to implement.**

**4. Justification for Choosing the Algorithm**

**The rule-based algorithm is chosen because:**

* **It is simple, efficient, and does not require large datasets or training.**
* **Suitable for small projects and web integration using Flask.**
* **Provides predictable, controlled responses ideal for demonstration and learning purposes.**

**🔍 ALGORITHM EXPLANATION WITH EXAMPLE**

**Algorithm Steps:**

1. **Start**
2. **Input: User message (msg)**
3. **Convert message to lowercase**
4. **Check if the message contains specific keywords:**
   * **If contains “hello” → respond “Hi there! How can I help you?”**
   * **If contains “your name” → respond “I’m ChatBot, your friendly assistant.”**
   * **If contains “bye” → respond “Goodbye! Have a nice day!”**
   * **Else → respond “I’m not sure about that. Could you please rephrase?”**
5. **Display response on the chat interface.**
6. **End**

**Example:**

**User Input: “Hello, bot!”  
Processing: Keyword “hello” detected.  
Output: “Hi there! How can I help you today?”**

**User Input: “What’s your name?”  
Processing: Keyword “name” detected.  
Output: “I’m ChatBot, your friendly assistant.”**

**User Input: “See you later.”  
Processing: Keyword “bye” detected.  
Output: “Goodbye! Have a great day 😊”**

**Implementation and Code:**

from flask import Flask, request, jsonify

app = Flask(\_\_name\_\_)

# HTML + CSS + JS all in one

page = """

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Simple ChatBot</title>

<style>

body {

font-family: Arial, sans-serif;

background: #f8f4f9;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.chat-container {

width: 400px;

background: #ffffff;

border-radius: 12px;

box-shadow: 0 4px 12px rgba(0,0,0,0.2);

overflow: hidden;

display: flex;

flex-direction: column;

}

h2 {

background: #ffb6c1;

color: #fff;

margin: 0;

padding: 12px;

text-align: center;

font-weight: bold;

}

.chat-box {

flex: 1;

padding: 10px;

overflow-y: auto;

background: #fafafa;

display: flex;

flex-direction: column;

}

.message {

margin: 10px 0;

padding: 8px 12px;

border-radius: 10px;

max-width: 80%;

word-wrap: break-word;

}

.user {

background: #ffb6c1;

align-self: flex-end;

color: white;

}

.bot {

background: #e0e0e0;

align-self: flex-start;

}

.input-area {

display: flex;

padding: 10px;

background: #f1f1f1;

}

.input-area input {

flex: 1;

padding: 10px;

border: none;

border-radius: 8px;

outline: none;

}

.input-area button {

background: #ffb6c1;

border: none;

padding: 10px 15px;

margin-left: 8px;

border-radius: 8px;

cursor: pointer;

color: white;

font-weight: bold;

transition: 0.3s;

}

.input-area button:hover {

background: #ff8da1;

}

</style>

</head>

<body>

<div class="chat-container">

<h2>💬 ChatBot</h2>

<div id="chat-box" class="chat-box"></div>

<div class="input-area">

<input type="text" id="user-input" placeholder="Type a message..." autocomplete="off" onkeypress="if(event.key === 'Enter') sendMessage()">

<button onclick="sendMessage()">Send</button>

</div>

</div>

<script>

function sendMessage() {

const userInput = document.getElementById("user-input");

const message = userInput.value.trim();

if (message === "") return;

appendMessage("You", message);

userInput.value = "";

fetch("/get", {

method: "POST",

body: new URLSearchParams({ msg: message }),

headers: { "Content-Type": "application/x-www-form-urlencoded" }

})

.then(response => response.json())

.then(data => {

appendMessage("Bot", data.response);

});

}

function appendMessage(sender, message) {

const chatBox = document.getElementById("chat-box");

const msgDiv = document.createElement("div");

msgDiv.classList.add("message", sender === "You" ? "user" : "bot");

msgDiv.innerHTML = `<strong>${sender}:</strong> ${message}`;

chatBox.appendChild(msgDiv);

chatBox.scrollTop = chatBox.scrollHeight;

}

</script>

</body>

</html>

"""

# Chatbot logic

def chatbot\_response(user\_input):

user\_input = user\_input.lower()

if "hello" in user\_input:

return "Hi there! How can I help you today?"

elif "your name" in user\_input:

return "I'm ChatBot, your friendly assistant."

elif "how are you" in user\_input:

return "I'm just a bot, but I'm doing great! How about you?"

elif "help" in user\_input:

return "Sure! Try asking me about my name or say hello."

elif "bye" in user\_input:

return "Goodbye! Have a wonderful day 😊"

else:

return "I'm not sure about that. Could you please rephrase?"

@app.route("/")

def index():

return page

@app.route("/get", methods=["POST"])

def get\_response():

user\_msg = request.form["msg"]

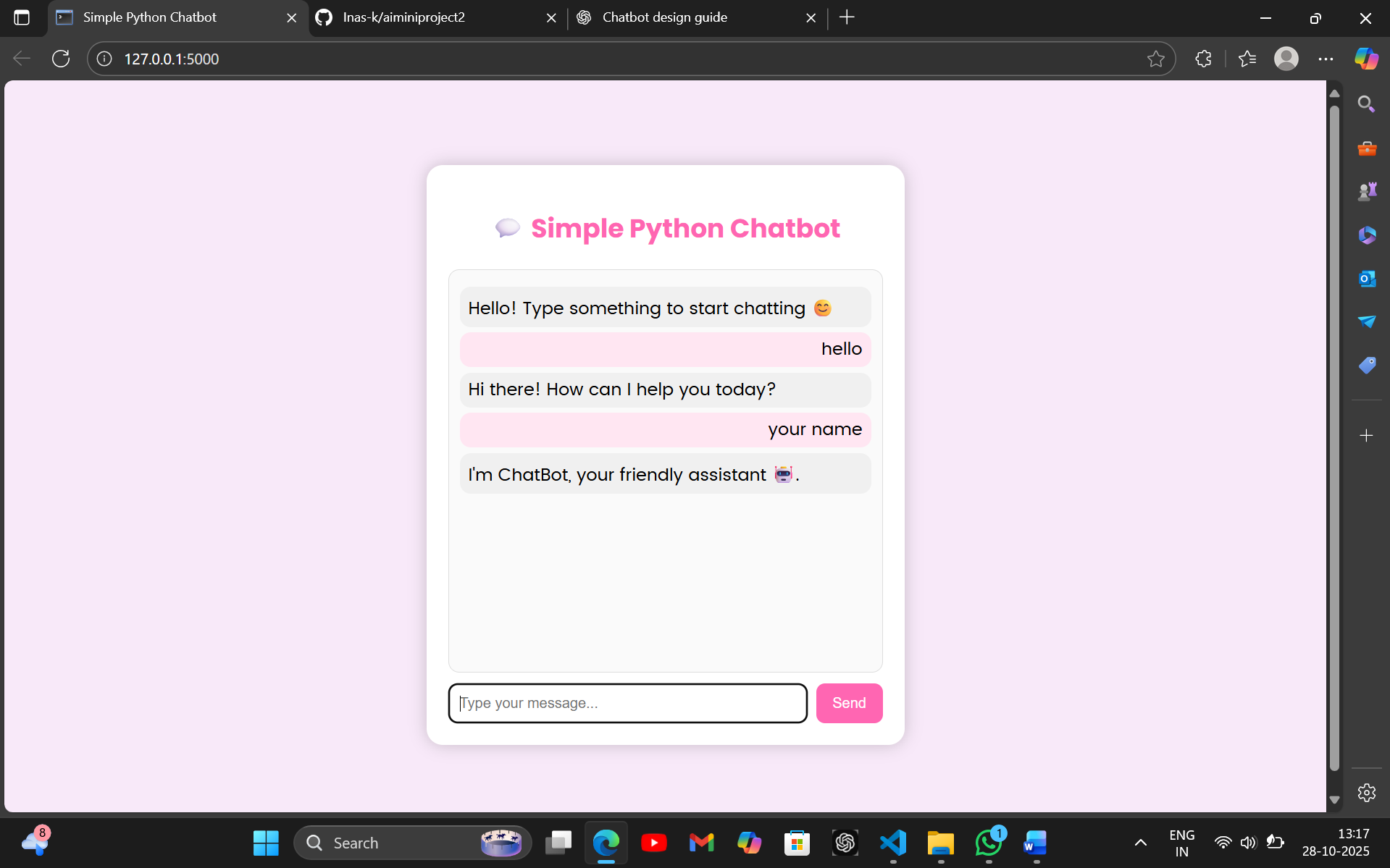
bot\_reply = chatbot\_response(user\_msg)

return jsonify({"response": bot\_reply})

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

**OUTPUT:**

****

**RESULTS AND FUTURE ENHANCEMENT:**

The chatbot web application was successfully developed using Python (Flask), HTML, and CSS. It responds to user queries in real-time through an interactive and user-friendly chat interface.

|  |  |
| --- | --- |
| **Git Hub Link of the project and report** | **https://github.com/Inas-k/aiminiproject1** |

**REFERENCES**

* **📚 References**
* Mark Lutz, *Learning Python*, 5th Edition, O’Reilly Media, 2013.
* Miguel Grinberg, *Flask Web Development: Developing Web Applications with Python*, O’Reilly Media, 2018.
* Jon Duckett, *HTML and CSS: Design and Build Websites*, Wiley Publishing, 2011.
* Flask Official Documentation — *Pallets Projects Team*
* W3Schools Online Web Tutorials — *HTML, CSS, and JavaScript References*