**Date:29.10.25**

**TASK:12**

# Implementation of Chatbot for technical support

**CO4, CO5 S3**

## PROBLEM STATEMENT :

Modern businesses and organizations heavily rely on technology, leading to an increasing demand for efficient technical support. Traditional support channels, such as phone calls and email, often suffer from high response times, limited availability (especially outside business hours), and high operational costs due to the need for human agents. The challenge is to provide instant, 24/7, and cost-effective support for common technical queries, thereby improving user experience and optimizing support team efficiency.

## AIM:

The primary aim of this project is to design, develop, and implement an intelligent chatbot capable of providing instant, automated, and accurate technical support for common user queries and issues, thereby improving user satisfaction and optimizing support operations.

## OBJECTIVE:

To achieve the aim, the project will focus on the following key objectives:

1. Natural Language Understanding (NLU) Development: To enable the chatbot to accurately interpret and understand diverse user queries expressed in natural language, identifying key intents and entities related to technical problems.
2. Knowledge Base Integration: To build and integrate a comprehensive knowledge base containing solutions, FAQs, troubleshooting steps, and relevant documentation for common technical issues.
3. Conversational Flow Design: To design and implement intuitive and effective conversational flows that guide users through problem-solving, gather necessary information, and provide clear solutions or escalate issues appropriately.
4. Issue Resolution & Escalation: To develop mechanisms for the chatbot to autonomously resolve a predefined range of common technical problems

and, for more complex issues, seamlessly escalate to a human agent with relevant context.

5. User Interface (UI) Development: To create a user-friendly and accessible chat interface that can be easily integrated into existing platforms (e.g., website, internal portal).
6. Performance Evaluation: To evaluate the chatbot's effectiveness in terms of accuracy, resolution rate, response time, and user satisfaction through testing and metrics.

**DESCRIPTION :**

The "AI-Powered Chatbot for Technical Support" project will deliver an automated solution designed to streamline technical assistance processes. The core of the system will be built upon Natural Language Processing (NLP) and Machine Learning (ML) techniques, enabling the chatbot to understand user queries, even when phrased informally or ambiguously.

The project will involve constructing a robust knowledge base, populated with common technical FAQs, troubleshooting guides, step-by-step instructions, and relevant documentation specific to the target technical environment (e.g., IT support for a company's software, hardware issues for a product, etc.). The chatbot will leverage this knowledge base to provide instant and accurate responses.

**ALGORITHM:**

Input:

- USER_QUERY: Text input from the user (e.g., "My printer isn't working," "How do I reset my password?").
- KNOWLEDGE_BASE: A structured collection of FAQs, troubleshooting steps, and solutions, linked to specific intents.
- NLP_MODEL: A pre-trained Natural Language Processing model for Intent Recognition and Entity Extraction.

Output:

- CHATBOT_RESPONSE: Textual response to the user.
- Optional: ESCALATION_FLAG indicating if human intervention is needed.

**PROGRAM :**

```
# --- Configuration & Initialization ---

class NLUModel:

    def recognize_intent(self, text):

        # Placeholder: In reality, use a trained NLP model

        text_lower = text.lower()

        if "printer" in text_lower and ("not working" in text_lower or
"troubleshoot" in text_lower):

            return "printer_troubleshoot"

        elif "password" in text_lower and "reset" in text_lower:

            return "password_reset"

        elif "login" in text_lower and ("issue" in text_lower or "cannot access" in
text_lower):

            return "login_issue"

        elif "connect to agent" in text_lower or "talk to human" in text_lower:

            return "escalate_to_human"

        return "fallback"


    def extract_entities(self, text):

        # Placeholder: In reality, use NER or regex

        entities = {}

        if "printer" in text.lower():

            entities["device"] = "printer"

        # Add more entity extraction logic
```

```
    return entities


class KnowledgeBase:
    def __init__(self):
        # Example simple knowledge base
        self.solutions = {
            "printer_troubleshoot": {
                "general": "Please ensure your printer is turned on and connected to your computer. Try restarting both devices. If it's a network printer, check your Wi-Fi connection.",
                "offline": "Check if the printer is set to 'Offline' in your print queue. Right-click the printer icon, select 'See what's printing', then 'Printer' > 'Use Printer Offline' to uncheck it.",
                "no_ink": "Check ink cartridges. Replenish if empty."
            },
            "password_reset": "To reset your password, please visit our password recovery page at example.com/reset_password. Follow the on-screen instructions.",
            "login_issue": "Please verify your username and password. If you've recently reset your password, ensure you're using the new one. Clear your browser's cache and cookies."
        }
        self.resolvable_by_chatbot = {
            "printer_troubleshoot": True,
            "password_reset": True,
            "login_issue": True
        }
```

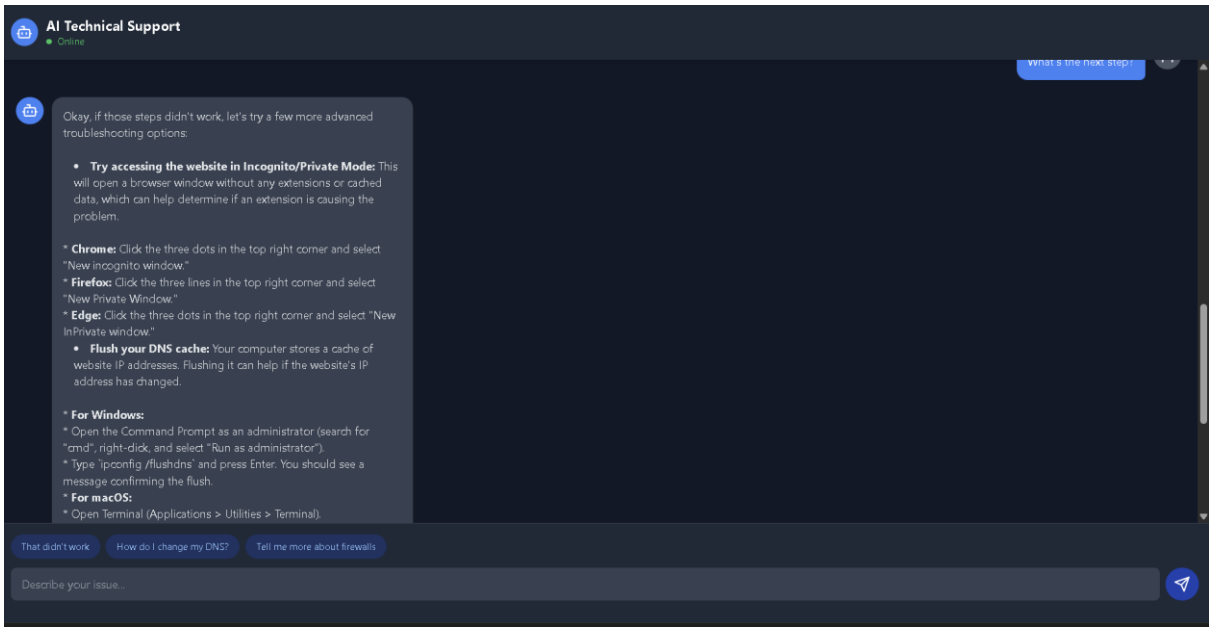
```
    def get_solution(self, intent, entities):
        if intent == "printer_troubleshoot":
            if "status" in entities and entities["status"] == "offline":
                return self.solutions["printer_troubleshoot"]["offline"]
            elif "problem" in entities and entities["problem"] == "no_ink": # More
complex entity detection
                return self.solutions["printer_troubleshoot"]["no_ink"]
            return self.solutions["printer_troubleshoot"]["general"]
        return self.solutions.get(intent, None)

    def is_resolvable_by_chatbot(self, intent):
        return self.resolvable_by_chatbot.get(intent, False)

# --- Chatbot Core ---
class TechSupportChatbot:
    def __init__(self):
        self.nlp_model = NLUModel()
        self.kb = KnowledgeBase()
        self.session_context = {} # Stores context for ongoing conversations

    def process_user_query(self, user_query, session_id):
        intent = self.nlp_model.recognize_intent(user_query)
        entities = self.nlp_model.extract_entities(user_query)
```

```
        response_text = ""
        escalation_flag = False

        # Update session context (simplified)
        self.session_context[session_id] = {"last_intent": intent, "entities": entities}

        if intent == "escalate_to_human":
            response_text = "I'm connecting you to a human agent. Please wait a moment."
            escalation_flag = True
        elif intent == "fallback":
            response_text = "I'm not sure I understand. Can you rephrase your question or tell me more? You can also type 'connect to agent' for human support."
        else: # Recognized intent
            solution = self.kb.get_solution(intent, entities)
            if solution and self.kb.is_resolvable_by_chatbot(intent):
                response_text = solution
            else: # Intent recognized but chatbot can't fully resolve or needs more info
                response_text = "I have an idea of what you need, but I might require more information or the issue is complex. Would you like to try rephrasing, or should I connect you to a human agent?"
                # In a real bot, you'd ask clarifying questions here instead of direct escalation
```

```
            # Example: if intent is 'printer_troubleshoot' but no specific problem entity, ask 'Is your printer offline, out of ink, or something else?'


        return response_text, escalation_flag


    def chat(self):
        print("Welcome to Tech Support Chatbot! How can I help you today? (Type 'exit' to quit, 'connect to agent' for human help)")
        session_id = "user123" # Simple single session for this example

        while True:
            user_input = input("You: ").strip()
            if user_input.lower() == 'exit':
                print("Chatbot: Goodbye!")
                break

            response, escalate = self.process_user_query(user_input, session_id)
            print(f"Chatbot: {response}")

            if escalate:
                print("Chatbot: Human agent has been notified.")
                break

# --- Main Execution ---
if __name__ == "__main__":
```

```
chatbot = TechSupportChatbot()

chatbot.chat()
```

## OUTPUT :




## CONCLUSION:

The "AI-Powered Chatbot for Technical Support" project aims to revolutionize the delivery of technical assistance by leveraging advanced natural language processing and machine learning. Through the meticulous development of an NLU engine for accurate intent recognition and entity extraction, coupled with a comprehensive and intelligently structured knowledge base, the chatbot will provide instant, 24/7, and precise solutions for common technical queries.