****

* **SPEECH RECOGNITION TECHNIQUES**
* **VIRTUAL CUSTOMER SUPPORT SYSTEM**

**Project-2**

**Submitted by,**

**MONISHA. G**

**aut820722CS006**

* **Objective:**

**To build an AI-based virtual assistant that interacts with customers via voice or text, understands their queries, and provides automated responses based on a defined knowledge base or dynamic AI models.**

* **Key Features:**

**Voice or text-based chat interface**

**Predefined or AI-generated responses**

**Query classification (billing, technical, etc.)**

**PDF log of conversation (optional)**

**Live hand-off to human agent (optional)**

* **Tools & Technologies:**

**Function**

**Tool / Library**

**Voice Recognition speech\_recognition or Whisper**

**Text Response OpenAI GPT, Rasa, or custom intents**

**GUI (optional) Streamlit, Flask, or Tkinter**

**PDF Export fpdf or reportlab**

**Data Storage CSV, SQLite, or JSON**

* **Project Workflow:**

**User Input**

**Via voice or text**

**Speech converted to text (if voice)**

1. **Intent Recognition**

**Detect user intent (e.g., billing issue, refund request)**

**Match with predefined responses or use GPT for dynamic replies**

1. **Response Generation**

**Return a response from the knowledge base**

**Optionally show related help articles**

1. **Logging**

**Save user queries and responses to PDF or CSV**

**Sample Python Code (Text + PDF Logging)**

**Import datetime**

**From fpdf import FPDF**

**# Sample knowledge base**

**Responses = {**

**“refund”: “To process a refund, please fill out the refund form or contact support.”,**

**“billing”: “Your billing cycle starts every 1st of the month.”,**

**“support”: “You can reach our live support from 9AM to 6PM.”**

**}**

**Conversation\_log = []**

**Def get\_response(user\_input):**

**For keyword in responses:**

**If keyword in user\_input.lower():**

**Return responses[keyword]**

**Return “Sorry, I didn’t understand that. Can you rephrase?”**

**Def chat():**

**Print(“Virtual Customer Support Bot. Type ‘exit’ to quit.”)**

**While True:**

**User\_input = input(“You: “)**

**If user\_input.lower() == “exit”:**

**Break**

**Bot\_reply = get\_response(user\_input)**

**Print(“Bot:”, bot\_reply)**

**Conversation\_log.append((“You”, user\_input))**

**Conversation\_log.append((“Bot”, bot\_reply))**

**Def save\_conversation\_to\_pdf(filename=”chat\_log.pdf”):**

**Pdf = FPDF()**

**Pdf.add\_page()**

**Pdf.set\_font(“Arial”, size=12)**

**Pdf.cell(0, 10, f”Chat Log – {datetime.datetime.now()}”, ln=True)**

**For speaker, text in conversation\_log:**

**Pdf.multi\_cell(0, 10, f”{speaker}: {text}”)**

**Pdf.output(filename)**

**Print(f”Chat log saved to {filename}”)**

**If \_\_name\_\_ == “\_\_main\_\_”:**

**Chat()**

**Save\_conversation\_to\_pdf()**

* **Optional Extensions:**

**Voice Input & Output using speech\_recognition + pyttsx3**

**Live Chat GUI with Streamlit or Tkinter**

**Machine Learning intent classifier (e.g., using scikit-learn or GPT)**

**Multilingual Support**

**Human fallback option**