AI Customer Support Agent

A PROJECT REPORT

Submitted by

Lakshay Gupta (24BCS10786)
Karan Juneja (24BCS10791)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

October, 2025

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1. Introduction to Project

This project, titled “AI Customer Support Agent”, demonstrates the practical use of Python for automating customer service operations through a voice-assisted application. The system uses Python-based frameworks and libraries to make and manage customer calls, analyze real-time speech, and generate structured summaries after the conversation.

1.2. Identification of Problem

Traditional customer service systems depend heavily on human operators to make and analyze support calls. This results in:

- Increased operational cost and human error,
- Delay in response time,
- Inconsistent reporting and documentation.
- By using Python-based automation, the project aims to:
- Reduce human dependency,
- Improve accuracy of call summaries, and
- Enable scalable and intelligent call management using AI-driven solutions.

CHAPTER 2: Background Study

2.1. Existing Solutions

Existing systems such as CRM voice integration tools (like Twilio, Zendesk Voice) focus mainly on call management and recording. However, they lack automated analysis and AI-driven summarization of the conversation. Most solutions require manual tagging or post-call data entry.

2.2. Problem Definition

To develop an automated voice-assisted calling system that can:

- Initiate customer calls automatically,
- Listen to conversations in real-time,
- Store conversation data in a structured database, and
- Generate summarized reports with issue and resolution details.

2.3. Goals/Objectives

- Automate customer call initiation using voice assistance.
- Record and analyze real-time call data.
- Store structured information (product, issue, resolution) in SQLite.
- Generate natural-language summaries for quick review.
- Provide a user-friendly React-based web interface for interaction.

CHAPTER 3. DESIGN FLOW/PROCESS

3.1. Evaluation & Selection of Specifications/Features

- Frontend: React.js for responsive UI and real-time event listening (call-start, speech-start, volume-level, call-end).
- Backend: Flask for handling API routes, Vapi integration, and analysis workflow.
- Database: SQLite with SQLAlchemy ORM for smooth integration and schema management.
- Speech & NLP: Automated speech recognition for transcribing and analyzing calls.

3.2. Analysis of Features and Finalization Subject to Constraints

- Constraints: Limited computational resources for real-time processing, network latency, and accuracy of speech-to-text models.
- Database Tables:
    - Calls: Stores call metadata (call_id, timestamp, duration, customer_name, email, phone, summary, resolution status, product).
- Finalized Features: Secure API-based communication, automated transcription, database logging, and post-call report generation.

3.3. Design Flow

1. User initiates a call via frontend (React).
2. React interacts with Flask backend using REST API.
3. Flask communicates with Vapi to start and monitor the call.
4. Real-time events (speech, call start/end) are tracked and stored.
5. Speech data is processed for transcription and structured analysis.
6. Results are saved in SQLite through SQLAlchemy ORM models.
7. The summary and analysis are displayed on the frontend dashboard.

CHAPTER 4. RESULTS ANALYSIS AND VALIDATION

4.1. Implementation of Solution

The implementation integrates all components:

- Frontend: React app displays live call status and summaries.
- Backend: Flask APIs manage call flow, using SQLAlchemy models to store and retrieve call data.
- Database: SQLite holds all structured records for persistence.
  Testing confirmed successful automation of call initiation, transcription, and result generation. Sample outputs included structured JSONs with key fields and human-readable summaries validated for accuracy.

CHAPTER 5. CONCLUSION AND FUTURE WORK

5.1. Conclusion

The Voice-Assisted Customer Call Application successfully demonstrates automation in voice-based customer support. It combines modern web technologies, natural language processing, and database management to provide structured post-call insights. SQLite and SQLAlchemy ensure efficient data storage and retrieval for analytical purposes.

5.2. Future Work

- Integration of AI sentiment analysis for deeper conversation understanding.
- Migration to PostgreSQL for scalability.
- Dashboard analytics for call statistics and performance tracking.
- Integration with CRM systems for enterprise-level deployment