BSCS-S25-009

03-134221-038 Syed Muhammad Suleman

03-134221-013 Hamza Latif

03-134221-050 Muhammad Afzal

Ai Call Agent

In partial fulfilment of the requirements for the degree of

Bachelor of Science in Computer Science

Supervisor: Mehreen Tariq

Department of Computer Sciences

Bahria University, Lahore Campus

January 2026

Certificate

We recieve the performance in the report bearing the title.

"Ai Call Agent"

written by

Syed Muhammd Suleman

Hamza Latif

Muhammad Afzal

as assurance of the standard required on the partial fulfilment of the degree of BSC in Computer science.

Approved by:

Supervisor: Mahreen Tariq

_____

05 January, 2026

DECLARATION

I/We confirm that the project report is an original work by me/us other than citing and quoting some works which have been properly credited. I/We also assert that it has not been already and simultaneously filed to any other degree or award in Bahria University or other universities.

Enrolment	Name	Signature
03-134221-038	Syed Muhammad Suleman	
03-134221-013	Hamza Latif	
03-134221-050	Muhammad Afzal	

Date : 05 January, 2026

DEDICATION

This is dedicated to our dear families, teachers and mentors who have been supportive and encouraging all through our academic life.

ACKNOWLEDGEMENTS

We would like to thank our supervisor, Mehreen Tariq, immensely on her helpful tips, further support, and constructive criticism during this project. Her experience and motivation are quite valuable to us since it has assisted us in solving difficult issues and attaining what we have thus far.

It is also with gratitude that we recognize the Department of Computer Sciences, Bahria University Lahore Campus, which helped us in achieving this project by availing the required resources and facilities in the entire process. We wish to say goodbyes to all the members of the faculty who have helped us in our academic growth and development.

Our families and friends also deserve our gratitude, and they supported us in the face of this challenging stage of the project. Their patience, understanding and their motivation has always been a strength to us.

Finally, we would like to thank all those organizations and researchers whose work we have referred to in this report. Their contribution to the research studies in artificial intelligence and natural language processing has been invaluable.

Syed Muhammad Suleman

Hamza Latif

Muhammad Afzal

Project Title

Ai Call Agent

ABSTRACT

A key element of a successful operation in all kinds of industries, including hotels, restaurants, clinics, etc., is the ability to service customers effectively. Although traditional systems for receptionists are expensive to install, there are instances of errors made by humans due to traditional business hours when people will not be available to answer questions. Due to this lack of availability, businesses have lost opportunities to serve customers because they do not return after not having their question answered. Using advanced AI technologies, such as Natural Language Processing (NLP), Machine Learning (ML), and speech-to-text/text-to-speech, the AI Call Agent (a phone system that interacts with customers through natural-sounding dialogue) allows customers to communicate in almost real-time with the AI. As a result, the AI Call Agent can effectively respond to basic service requests like FAQ responses, reservation information, hours of operation, etc., while also being able to automatically escalate requests to a human agent for anything that cannot be resolved via the AI Agent. The AI Call Agent provides support for several different languages, thus allowing companies to service a broader range of customers from varying backgrounds. In terms of technology, the AI Call Agent uses a NextJs & NodeJs for the web and mobile technologies using Flutter, and the functionalities of AI are deployed via a microservice architecture leveraging Large Language Models (LLM). This allows for a higher degree of reliability, better scalability, and improved performance, as well as lower costs for businesses that provide customer service via modern customer queries, than traditional receptionist systems. In addition, testing has demonstrated that the AI Call Agent's speech-to-text capabilities can achieve high levels of speech recognition accuracy.

TABLE OF CONTENTS

DECLARATION **ii**
ACKNOWLEDGEMENTS **iv**
ABSTRACT **v**
TABLE OF CONTENTS **vii**
LIST OF TABLES **x**
LIST OF FIGURES **xi**
LIST OF SYMBOLS / ABBREVIATIONS **xii**
LIST OF APPENDICES **xiii**

CHAPTERS

1 11 INTRODUCTION 1
1.1 Background 1
1.2 Problem Statement 2
1.3 Aims and Objectives 3
1.4 Scope of Project 4

2 66
2.1 Overview 6
2.2 Evolution of Customer Service Automation 6
2.3 Telephony Artificial Intelligence. 7
2.4 Natural Language Processing Developments. 9
2.5 Speech Recognition Technologies 11
2.5.1 The Automatic Speech Recognition . 13
2.5.2 Text-to-Speech Synthesis 14
2.6 Existing AI Call Agent Solutions 14

2.7 Research Gap and Innovation Opportunities 17

3 20 20

3.1 System Architecture Overview 20

3.2 Requirements Analysis21

3.2.1 Functional Requirements 21

3.2.2 Non-Functional Requirements 23

3.3 Technology Stack 23

3.4 Database Design 26

3.4.1 Data Modeling Principles 26

3.4.2 Collection Design 27

3.4.3 Data Access Patterns 29

4 **DATA AND IMPLEMENTATION**31**31**

4.1 Development Environment Setup 31

4.2 Frontend Implementation 33

4.2.1 State Management with React Query 36

4.2.2 User Interface Components 37

4.3 Backend Implementation 38

4.3.1 Express Server Configuration 38

4.3.2 Database Integration 39

4.3.3 API Controller Implementation 40

4.4 AI Services Integration 41

4.4.1 Speech-to-Text Processing 42

4.4.2 LLM Response Generation 43

4.4.3 Text-to-Speech Synthesis 43

4.5 Security Implementation 44

5 46**45**

5.1 System Performance Evaluation 45

5.2 Functional Testing Results 46
5.3 User Acceptance Testing 49
5.4 Performance Benchmarking 51
5.5 Comparing Results with Project Goals 53
5.6 Limitations and Challenges 55
5.7 Business Impact Assessment 58

6 Error! Bookmark not defined.61
6.1 Summary of Achievements 61
6.2 Contribution to Knowledge 62
6.3 Practical Implications 63
6.4 Recommendations for Implementation 64
6.5 Future Research Directions 64
6.6 Concluding Remarks 65

REFERENCES 67

APPENDICES 69

LIST OF TABLES

TABLE	TITLE	PAGE
Table 2.1:	Comparison of Speech Recognition Technologies	13
TABLE2.2:	EXISTING AI CALL CENTER SOLUTIONS FEATURE COMPARISON	16
Table 3.1:	Functional Requirements Specification 21	
Table 3.2:	Non-Functional Requirements Specification	23
Table 3.3:	Technology Stack Justification	25
Table 3.4:	Database Collections Schema	29
Table 4.1:	Development Tools and Versions	32
Table 4.2:	Frontend Component Hierarchy	36
Table 4.3:	Backend Controller Functions	41
Table 5.1:	System Performance Metrics	46
Table 5.2:	Functional Test Cases and Results	49
Table 5.3:	User Acceptance Testing Feedback	51
Table 5.4:	Performance Benchmarking Results	53
Table 5.5:	Objective Achievement Analysis	55
Table 5.6:	Cost-Benefit Analysis for Businesses	60

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 2.1:	Evaluation of Customer Service	7
Figure 2.2:	Ai Telephony System Architecture	9
Figure 3.1:	System Component Interaction Sequence Diagram	26
Figure 3.2:	DataBase Schema Diagram	28
Figure 4.1:	Development Environment Infrastructure	33
Figure 4.2:	Frontend Application Directory Structure Diagram	35
Figure 4.3:	UI Component Hierarchy	37
Figure 4.4:	Backend Middleware Stack	39
Figure 4.5:	AI Service Integration Pipeline	42
Figure 5.1:	Functional Testing Results Summary	47
Figure 5.2:	User Satisfaction Metrics	50
Figure 5.3:	Identified Limitations Analysis	58
Figure 6.1:	Project Achievement Summary	62
Figure 6.2:	Knowledge Contribution Framework	63

LIST OF SYMBOLS / ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CRM	Customer Relationship Management
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
FAQ	Frequently Asked Questions
GPT	Generative Pre-trained Transformer
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JWT	JSON Web Token
LLM	Large Language Model
MERN	MongoDB, Express, React, Node.js
ML	Machine Learning
NLP	Natural Language Processing
REST	Representational State Transfer
SDK	Software Development Kit
SQL	Structured Query Language
STT	Speech-to-Text
TTS	Text-to-Speech
UI	User Interface
UX	User Experience
VAD	Voice Activity Detection
WER	Word Error Rate

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
APPENDIX A:	Source Code Structure	68
APPENDIX B:	API Documentation	71
APPENDIX C:	Database Schema Details	79
APPENDIX D:	User Manual	82
APPENDIX E:	Test Cases and Results	85
APPENDIX F:	Deployment Guide	86

INTRODUCTION

1.1 Background

Customer service in the landscape has been dramatically transformed especially in the recent years due to the high rate of technological improvement and the evolving demands of consumers. The responsiveness of customer service and its quality has become the most important factor of differentiation in the modern business environment, which directly affects customer satisfaction, retention, and business success. Firms in different industries are especially in hospitality, health care, as well as service sectors are realizing the strategic value of providing outstanding customer experiences.

Conventional methods of customer service that mostly depend on human receptionists and call center operatives have been the main face of the business to the customer. Although these traditional systems have been found to be effective in most situations, this is facing a number of inherent constraints which are increasingly making their operational efficiency and quality of service provision compromised. Financial cost of having full time employees especially when it involves 24 hours operation is a great operating cost to any business operations irrespective of size.

The limitation of human capacity is also involved with human-managed customer service systems as long as working hours, availability of staff, and human capacity constraints exist. The wait time of customers is also likely to increase during peak periods, and the inquiries during off-hours can go unanswered at all until the business is opened again. This time constraint can lead to loss of business, irritated customers and loss of revenue. Also, the quality of service delivery may be of great variation depending on the personal operator knowledge, mood, exhaustion as well as the level of training.

The rise of artificial intelligence and natural language processing technologies has provided new avenues of dealing with these old issues. The concept of machine learning algorithms has proven to be able to afford human-like text and speech generation as well as comprehension, especially through very large language models. The innovations have created the possibilities to come up with smart automated systems that can simulate most of the customer service interaction between humans and cut down on the traditional restrictions.

The latest advancements in speech recognition technology have realised higher standards of accuracy in various languages and accents. On the same note, text to speech synthesis has improved to deliver a natural sounding voice which closely resembles the speech pattern of a human being. Together with advanced natural language processing features offered by large language models, these technologies can be used to produce conversational AI systems, which are capable of having meaningful conversations with customers in ways that are contextually sensitive.

The given project is preconditioned by the realization of the challenges of the traditional model of customer service and opportunities of contemporary AI technologies. Through the creation of a smart, automated call handling system combined with integrated advanced voice communication platforms, we are expected to offer a business with a game changer to improve the quality of customer service in an operationally efficient and cost-effective way.

1.2 Problem Statements

This project will be focused on the main challenge of the inefficiency and inability of the traditional systems of customer service to satisfy the needs of the contemporary business. The many companies in different sectors are challenged with enormous challenges in ensuring that they provide the same high quality of customer service at sustainability in their operations.

The modern world of business faces a number of burning problems in its work with customers:

Expensive operation expenses: This is because of the cost of ensuring that sufficient human resource is in place to handle customer service related operations and this is a costly affair especially in the case of any business that is expected to be available at all times (24/7).

Inadequate Availability: Traditional systems only operate within the working hours and therefore do not pick up any inquiries during the off-hours, holidays and weekends when customer needs might be most intense.

Poor Consistency of Service Quality: The quality of the service is drastically varying based on the determination. On the individual factors of the particular operators like experience, training, fatigue and personal conditions.

Scalability Issues: The traditional systems as the business grows or when it reaches its peak become ineffective and are not capable without expansion of commensurate expansion of human resource and facilities.

Language Barriers: There are one-off, multilingual customers, which need special employees or translators, which makes it more complex and expensive.

Information Access Delays: Necessary Multiple interactions or escalation retrieving a particular business data or even complex queries can take a long time to resolve.

All of these challenges have an effect on the competitiveness of business, the customer satisfaction, as well as the efficiency of the operations. The lack of a useful automated solution leaves businesses with the dilemma of either quality of the service provided or cost control at the cost of a trade-off that is quite unsustainable in competitive markets. The given project is directly related to these issues and it can be seen as the development of an AI-based call agent system that will offer intelligent customer service abilities that are automatically operated.

1.3 Aims and Objectives

The work on the creation of the Smart AI Call Agent system can be characterized by the presence of specific and measurable goals that will be used in order to resolve the defined problems in the operations related to customer service. The major aims of this project are:

Design an Intelligent Automated Call Handling System: Develop a complex AI-driven system that can handle and answer customer queries instantly with more advanced algorithms of natural language processing and machine learning. This includes speech-to-text translation, natural language processing, natural response generation via large language models as well as text-to-speech synthesis.

Create 24/7 Customer Support: Do not confine customer care to time, introduce a system that can work round the clock and will not go offline, whether at any given time, day or season, customer queries should be attended to at any given time. The system is managed with FAQs, bookings, reservations and general information request to the business.

Introduce Multi-Language Support: Build full language processing options to address the needs of a wide variety of customers by offering the support of English, Urdu,

Arabic, and other languages with an accurate speech recognition, linguistic subtlety comprehension, and suitable response generation.

Secure and Reliable System: The system must be well designed, error tolerant and highly available; secured system will ensure the security of sensitive customer data by providing comprehensive error handling and data transmission, storage, persistent performance under load, and regulatory compliance.

Build Intuitive Interfaces: Develop user-friendly web and mobile solutions that allow business to set up, manage and control their AI call agent systems easily with detailed dashboards, knowledge base applications, reporting and administrative controls.

1.4 Scope of Project

Scope

The scope of this project consists of the full design and development of an artificial intelligence/AI powered call agent system along with the following scope:

- Creation of a web-based managerial platform based on MERN stack technologies (MongoDB, Express.js,Nextjs, Node.js).
- Development of cross platform mobile apps in Flutter on iOS and Android platforms.
- Embedding of advanced synthesizing and recognizing speech systems by Vapi platform.
- Application of natural language processing with large language models.
- Multi-languages such as English, Urdu and Arabic.
- Scalability and reliability Cloud-based deployment architecture.
- Live call management, context awareness and conversation management.
- Full testing and validation in different use cases/scenarios.

- Full documentation, such as technical specification, user documentation and API documentation.

Limitations

Although this project is thorough in its approach, it has a number of limitations:

- The effectiveness of the system depends on how good and comprehensive the business-specific training data submitted by the clients are.
- In real world practice, speech recognition can be influenced by low quality calls, accent, or high background noise.

Also, even in high complexity, emotionally sensitive or exceptional scenarios, a human intervention is still necessary and these scenarios cannot be fully automated.

Initial preparation and customization will entail data preparation, system set-up and employee training.

- Continuous expenditures on cloud solutions, API, and Vapi platform subscriptions are to be taken into account during operational planning.
- The internet connectivity must be stable in order to make the system work.
- Different regions and industries might have different regulatory compliance needs, and this might need more configuration.

LITERATURE REVIEW

2.1 Overview

The chapter reviews the existing level of research and development in the field of AI-based customer service systems, the technologies, and methods applicable to automated call handling. The survey includes the discussion of natural language processing methods, speech recognition applications, conversation AI systems, and real-life applications in the business contexts. Through work analysis we determine the theoretical framework of our system and a chance to be innovative and improve the things.

2.2 Evolution of Customer Service Automation

Customer Service Automation has been evolving over time, beginning in the 1980s with the initiation of the first systems.

Customer service automation has been changing, and developing enormously within the last several decades. The initial systems were based on the Interactive Voice Response (IVR) technology, whereby the pre-recorded messages and touch-tone input was used to direct callers through the menu choices. These systems were revolutionary when they were first introduced but constrained by inflexible navigation systems and incapable of comprehending natural language.

The introduction of speech recognition technology in 1990s and the 2000s was a major development where systems could comprehend basic voice instructions. Their initial implementations were however limited by small vocabularies, inability to operate in noisy settings and failure to deal with complex dialogues. Experiments proved that the first speech recognition systems had the word error rate of 30-40 percent in the situation of a real world.

With machine learning and deep neural networks introduced in the 2010s, the nature of automated customer service changed. The contemporary systems make use of advanced natural language perception, situational understanding, and adaptive learning[1]. A study carried out by Ryu et al. (2019) regarding AI-based real-time agent advisor systems in call centers showed a significant increase in the call handling efficiency and customer satisfaction.

The modern AI-based customer service systems have transformer-based, large language models, and sophisticated speech processing algorithm that performs almost as well as human customer service in most cases. These technologies being integrated with the cloud computing platforms have allowed scalable and cost efficient implementation of advanced customer service solutions based on automation.

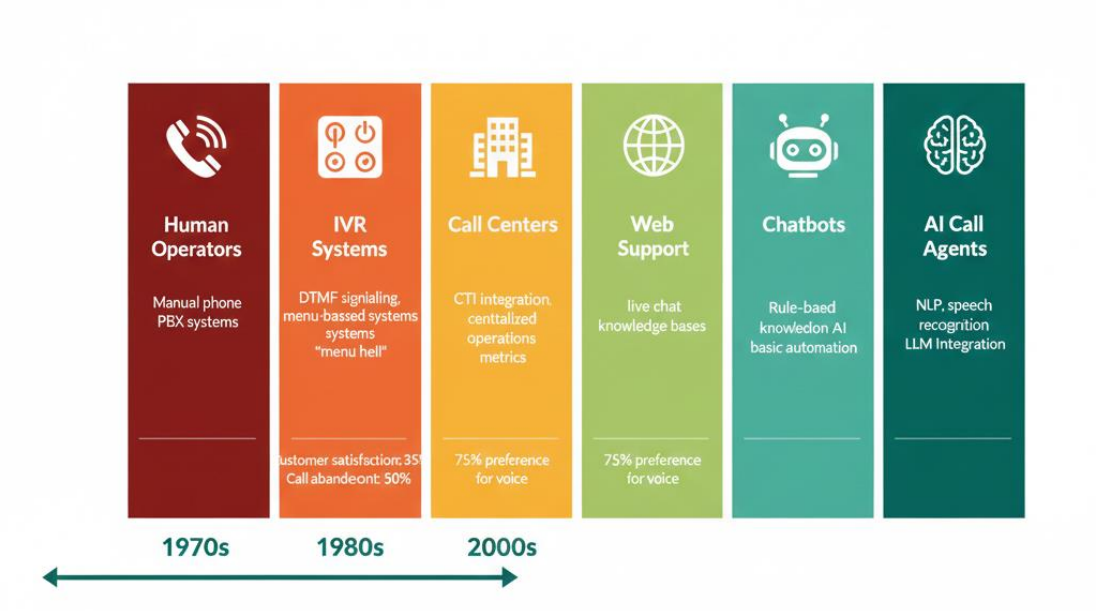

Figure 2.1: Evaluation of Customer Service

2.3 Telephony Artificial Intelligence.

The systems with artificial intelligence and telephony have established new customer service automation paradigms. The field of AI-powered telephony is the intersection of several technological fields, such as speech processing, natural language understanding, dialog management, and machine learning.

The initial AI telephony systems emphasized more on speech recognition precision. Nonetheless, these systems were difficult to cope with the real-life differences in accent, background noises, and speech patterns.

Application of the deep learning methods, specifically Long Short-Term Memory (LSTM) networks and subsequently transformer networks, led to the breakthrough in AI telephony. Baidu had created DeepSpeech which showed the potential of end-to-end deep learning in speech recognition, with word error rates below 10 percent on benchmark data[2].

Modern AI telephony applications use advanced pipelines, which incorporate various AI elements. The common architecture contains:

- Automatic Speech Recognition (ASR) of converting speech to text.
- Natural Language Understanding (NLU) in extracting meaning in text.
- Dialog Management to sustain the context of conversations.
- Natural Language Generating (NLG) to generate the right responses.
- Text-to-Speech (TTS) synthesis to convert the responses to audio.

A study by Zhang et al. (2023) shows that customer satisfaction baseline with the use of integrated AI telephony is equal to that of human agents with common inquiries and it also reduces the cost of handling the query up to 70 percent. They found that 68% of queries made by customers were effectively addressed by AI agents without human intervention in their study of the implementations in the banking sector[9].

The advent of Large Language Models (LLMs) has also changed the telephony capabilities of AI. Models such as GPT-4 and open-source versions support much more natural and context-aware conversations which are capable of supporting multi-turn dialogues. According to the research conducted by Microsoft Research (2024), the use of an LLM-driven telephony agent helps decrease customer frustration by 45% compared to the use of a conventional IVR[3].

But there are still serious problems to AI telephony. Inference pipeline optimization is required by real-time conversation latency requirements, and the literature shows that delays of response longer than 2.5 seconds affect user satisfaction substantially. Also, emotional conversations and edge cases have proven to be a challenge to full automated systems.

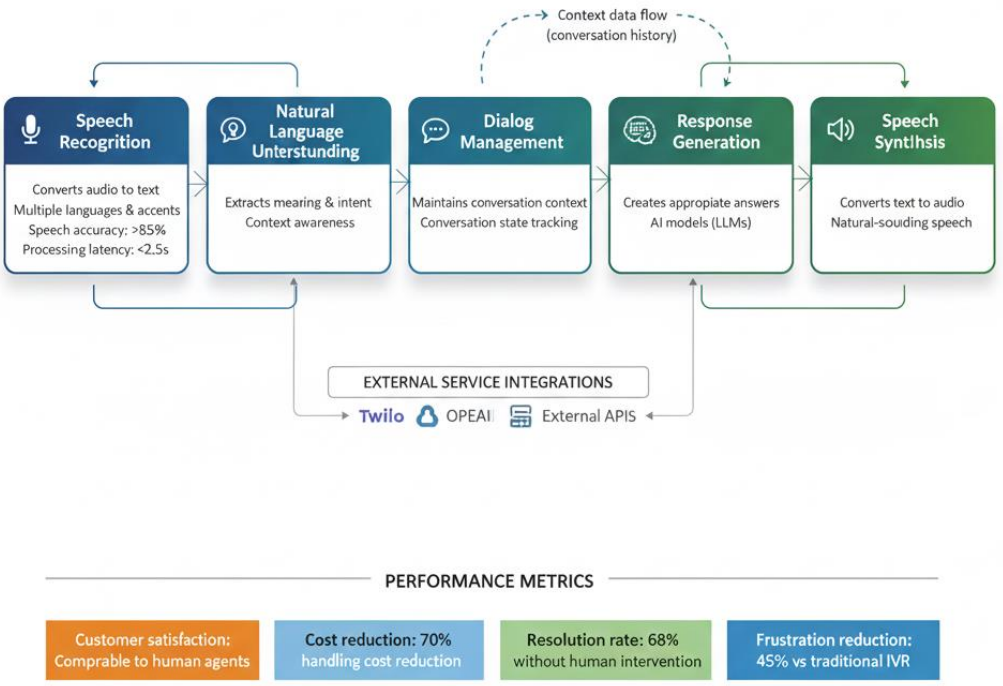

Figure 2.2: Ai Telephony System Architecture

2.4 Natural Language Processing Developments.

Over the last few years, Natural Language Processing has been transformed in a revolutionary way, which made possible the advanced functionality of the current AI call agents. The development of the rule systems to the statistical ones to the modern neural systems has transformed the performance of the machine to understand and produce human language by a large margin.

The most important recent innovation in NLP is the transformer architecture proposed by Vaswani et al. in 2017. The self-attention mechanism enables models to compute over full sequences at once, and long-range dependencies in a better way compared to previous recurrent designs. The breakthrough made it possible to create Large Language Models (LLMs) which exhibit exceptional language models and language generation capabilities[4].

In 2018, Devlin et al. presented a new model known as BERT (Bidirectional Encoder Representations from Transformers) that created a breakthrough in the field of natural

language understanding by pre-training on large text corpora then fine-tuning it to perform a specific task. With this strategy, the state-of-the-art results were obtained on various benchmark models such as GLUE and SQuAD, which proves the reliability of transfer learning in NLP.

The GPT (Generative Pre-trained Transformer) family, especially GPT-3 and GPT-4, was capable of new few-shot learning and conversational AI achievements. Brown et al. (2020) demonstrated that when model parameters are scaled to hundreds of billions, the model can gain emergent capabilities that are not found in smaller models. Such innovations have a direct positive impact on AI call agents as they allow more natural and context-driven conversations.

In recent studies, there has been an interest in the direction of more effective and accessible LLMs. Such methods as quantization, knowledge distillation, and parameter-efficient fine-tuning have made it possible to run powerful language models on less powerful hardware[5]. Meta AI LLaMA models The LLaMA family of models has shown that well trained smaller models (7B-65B parameters) can perform as well as much larger models.

In the case of conversational AI, in particular, the following are the major innovations:

Context Window Management: Current dialogue systems are capable of storing context over a significantly longer conversation. GPT-4 has a context window of 128K tokens (conversational support 128K tokens) and is capable of coherent multi-session discussions.

Emotional Intelligence: Affective computing studies have helped AI systems to be more receptive to emotions and responsive to them. Such systems as Xiaoice created by Microsoft have shown that emotional intelligence is also important to ensure that conversations remain interesting.

Multimodal Understanding: Although the present AI call agents are mostly audio-based, studies in multimodal AI involving text, audio and eventually video cues hold promise of providing a more detailed insight into what people are saying.

Domain Adaptation: Fine-tuning methods of general language models on domains have become much better. PEFT-based solutions (such as LoRA (Low-Rank Adaptation)) can allow such customization to be done effectively with relatively limited computational effort.

Irrespective of these developments, there are still issues in such areas as managing code-switching (when speaking more than one language at the same time), cultural sensitivity, and the ability to stay consistent during a long conversation. NLP systems are still being researched to be strengthened, more ethical and human value oriented[11].

2.5 Speech Recognition Technologies

The most important part of AI call agent systems is speech recognition technology, which converts verbal communication into the text that is further processed. The history of Automatic Speech Recognition (ASR) has been marked with a series of advances in precision, strength, and effectiveness.
The shift in developing Hidden Markov Models (HMMs) to Deep Neural Networks (DNNs) was a breakthrough in the development of ASR. HMM-GMM systems that had been predominant in the field over the decades were generally performing at 20-25% word error on conversational speech. These error rates dropped by 30-50 points with the introduction of DNN-based acoustic models, which is the largest increment in the performance of ASR in decades.

The state-of-the-art in speech recognition is presented by end-to-end approaches. End-to-end systems directly map audio properties to text sequences, unlike the traditional systems that distinguish between acoustic modeling, pronunciation modeling and language modeling. This is an architecture called Listen, Attend, and Spell (LAS) architecture proved the achievability of such approach, where simplified pipelines were able to achieve competitive results.

ASR has also been enhanced by transformer-based architectures. The Conformer model, a hybrid of convolutional and self-attention networks, used to extract local features and global context respectively, has achieved new areas of records on various speech recognition tasks. Conformer models demonstrate word error rate lower than 2.0% on clean speech on the LibriSpeech benchmark, which is close to human-level accuracy conditions.

The present trends are based on the resolution of practical problems in the real world implementation:

Resistance to Noise: The present ASR systems have developed front-end processing. and data augmentation techniques that guarantee the accuracy of the system even in noisy and conditions.. The technique of specAugment that employs time and frequency masking on spectrograms has also been found to be effective in enhancing noise robustness.

Multilingual Capabilities: Multilingual end-to-end models that are trained on multilingual data can be able to identify speech across multiple languages without having to explicitly identify the language. The Universal Speech Model by Google proves the scalability of such a method, as it can support more than 100 languages using only one model[7].

Adaptation to accent and Dialect: Contrastive methods such as adversarial domain adaptation and targeted fine-tuning can be used to achieve better results on regional accents and dialects. It has been found that accent diversity in training data can be carefully monitored to minimize performance differences among demographic groups.

Low-Resource Languages: Self-supervised speech models such as wav2vec 2.0 have radically lowered the data usage of training powerful ASR models. Such models can attain acceptable performance using a small amount of labeled data of less than one hour based on large volumes of unlabeled audio[8].

Technology	**Word Error Rate**	**Key Features**	**Limitations**

HMM-GMM	20-25%	Mature Techonology , Efficient	Poor noise robustness
DNN-HMM	10-15%	Better acoustic mdoeling	Complex trainnig pipeline
End-to-End (LAS)	5-8%	Simplified Pipeline	Requires Large Datasets
Transformer-Based	2-4%	Excellent Context Modelling	High Computational Requirements
Conformer	1.5-3%	Combines CNN's and transformers	Complex Architecture

Table 2.1: Comparison of Speech Recognition Technologies

When using AI call agent applications, there are other aspects such as the real time processing limitations, speaker diarization (who spoke when), and speech overlapping. The ASR solution that is the best is one that considers the requirements of a particular deployment in terms of accuracy, latency, and computational efficiency.

2.5.1 The Automatic Speech Recognition .

Over the last few years, the technology of Automatic Speech Recognition (ASR) has come very far and is now almost human in its accuracy in most applications. The contemporary ASR systems use deep neural networks, which are trained with huge amounts of collected speech, which allows them to work with different accents, speaking styles, and acoustic situations.

The commercial ASR systems like Google Speech-to-Text, Amazon Transcribe and Microsoft Azure Speech Services use advanced acoustic features and language models to allow a high level of accuracy over various languages[12]. Studies that have been conducted to compare these services have indicated that the accuracy rate is above 95 percent in the case of clear speech in controlled conditions, but the accuracy declines when there is background noise, heavy accents and special terminologies.

The use of open-source alternatives such as Whisper by OpenAI and DeepSpeech by Mozilla have shown competitive performance and are more flexible to be tailored to one-use requirements. Whisper, especially, has been in the news due to its multilingual strengths and its ability to deal with problematic audio situations. A study comparing

the performance of Whisper on 97 languages had comparable or superior error rates in words on a variety of language pairs to commercial systems[10].

2.5.2 Text-to-Speech Synthesis

Text-To-Speech (TTS) technology has been developed into more natural and expressive speech that resembles human vocalization in a very close manner. Current TTS systems use neural network architectures to produce audio waveforms directly by means of text with prosody, intonation, and expression of emotion.

Modern TTS methods include the deep-generative model, WaveNet that generates speech that is highly realistic and Tacotron, an end-to-end neural TTS framework. Such technologies can be used in commercial services, such as Google Text-to-Speech, Amazon Polly, and Azure Speech Services, which provide various voices and languages and can be customized[6].

The evaluation of TTS quality has determined that the speech quality of the modern systems is completely indistinguishable in a controlled environment with those of a human being. Studies

2.6 Existing AI Call Agent Solutions

Its AI call agent solution business is like a growth industry and there are numerous offerings out there, including full-scale enterprise solutions, and vertical solutions. Having an understanding of the contemporary environment of these services is essential in identifying areas where innovation can be made and also in learning through the exemplary practice.

Enterprise Platforms: Major cloud vendors have deployed full call agent AI solutions as an offering. As the example of Google Contact Center AI with its integration of Dialogflow and telephony that provides a seamless automation is an example. This platform is highly intent recognition, and it has good context management and transition to human actors in cases where necessary. Similarly, Amazon connect is based on Lex and Polly engaged to provide natural language understanding and speech synthesis respectively, both tightly coupled with services offered by AWS.

Niche Startups: some of the startups are focusing on AI call agents technology. An example is Vapi, which is a platform allowing users to build voice AI agents using customizable personalities and workflows with relative ease because of its developer-friendly nature. Their system is created to have the low-latency performance and flow of natural conversation. The other interesting competitor is Cresta, which focuses on real-time assistance to agents and optimization of call centers with the help of conversation intelligence.

Open Source Solutions: The open source community has gone ahead to come up with schemes of conversational AI systems development. Rasa Open Source offers solutions to developing contextual AI assistants that have customized NLU models and dialogue control. Although such solutions might need slightly greater technical expertise, their benefits are a high level of flexibility and control over proprietary solutions.

Research Prototypes: Research institutions have been influential in driving the field of conversational AI far. Stanford Conversation AI group has developed ChitChat, an architecture that helps in developing interesting social chatbots. In parallel, the Dialogue Systems group at MIT has been exploring the problem of increasing conversational coherence and preserving personality.

Solution	**Target Market**	**Key Features**	**Pricing Model**	**Limitations**

Google CCAI	Enterprise	Google Cloud integration, advanced NLU	Usage-based, enterprise	Complex setup, vendor lock-in
Amazon Connect	Mid-market to Enterprise	AWS ecosystem integration, scalable	Pay-per-use	Limited customization options
Vapi	SMB to Mid-market	Developer - friendly, low-latency	Per-min pricing	Smaller feature set than enterprise
Twilio Autopilot	Developers, SMB	Flexible conversation design, API-first	Usage-based	Requires significant development effort
RasaOpen Source	Developers and ,Researchers	Complete control, customizable	Free (open source)	Requires Ai/ML

TABLE 2.2: EXISTING AI CALL CENTER SOLUTIONS FEATURE COMPARISON

An analysis of the existing solutions in the market presents us with some generic challenges that seemingly open the ground to some very promising innovations:

Complexity of customization A significant number of enterprise software require a lot of technical expertise to be customized and integrated. It is a pity to note that small and medium-sized businesses do not usually have the means to exploit the full potential of these platforms.

Multilingual Support Issues: It may be so, big platforms provide support in several languages, but the performance may fall significantly between the languages. Support of less popular languages and dialect is also lacking.

Integration Ineffectiveness: The challenge of integrating these solutions with the business systems like CRM, ERP and booking systems is usually met by having to go into the realm of custom development which has a tendency of escalating the implementation period and cost.

Cost Structure Problems: The usage based pricing model may increase significantly particularly and the businesses with a high number of calls. On top of that, certain solutions have got complex pricing schemes which make it difficult to estimate expenses.

Noise Performance: In practice, in the unfriendly noise environments, deployments may be subjected to tricky acoustic environments that may confound the speech recognition performance. Background noise and poor audio quality are some of the reasons why the effectiveness of current solutions may differ considerably.

The AI Call Agent project is set to address these challenges directly, which involves making things more approachable as well as streamline the performance and provide actual business value.

2.7 Research Gap and Innovation Opportunities

As you see what already exists on the market both in the research literature and the products on sale, it begins to occur to you that there are already some blank areas in the world of the AI call agents. It is where this project comes in. Sealing these loopholes is not an incidental advantage; it is the core of this project.

Affordability to Small and Medium Businesses: AI call solutions are usually designed with large companies in mind. They expect you to have a technology team and a massive budget- most small and medium companies do not. These companies require something more basic, less expensive, and designed as to their practical working conditions[13]. We are addressing this through the creation of simple interfaces to use, transparent and fair pricing, as well as attention to the features that the SMBs actually require rather than giving them too much features that they will not be able to utilize.

Multilingual and Cross-cultural Capabilities: Yep, large platforms claim to be able to support a large number of languages, but support quality tends to decrease rapidly, particularly when it comes to languages that do not have massive user bases on the Internet. The result? It sounds as though the conversations were robotic or simply inaccurate. We are exerting an extra effort in ensuring that the system works seamlessly with the English-Urdu users and this would matter a lot to the markets where we are setting targets.

Integration Simplicity: AI call agents should not be once an interface that requires connecting to tools that a business already uses, such as CRM, booking system, payment, etc. At this point, it is frequently so, and most businesses must acquire developers to get it functional. We are designing connector plugs and simple, lightweight patterns of integration to allow companies to avoid a technology marathon to be started.

Scaling Economically: The method of most commercial solutions charging per use would appear reasonable, but in businesses that make plenty of calls, it can become extremely expensive in a short period of time. We are exploring new foundations and optimization wizards that enable companies to achieve large volumes at lower prices, such as intelligent utilization of open-source software and utilizing the resources available better.

Resource-Constrained Environments Performance: In the majority of studies, performance is assumed to be optimal; there is an abundance of bandwidth, use of

fast servers, and a stable internet connection. However, it is not always the case with real businesses. We are working on ensuring that our AI can perform even in cases when the technological infrastructure is not optimal to be more plausible to make it accessible to more people.

Assessment Measures Other than Accuracy: These systems can be assessed by how accurately they identify word or intents, the standard measure of these systems. That is okay, but does not give you whether the customers are content or whether the technology is benefiting the business. We are developing assessment instruments that examine customer satisfaction, conversion rates and total efficiency, the actual performance not the technical one.

AI and Privacy Ethics: You are working with sensitive information when it comes to customer calls. Performance is not the only thing, but it is about doing right things with data, keeping things fair, be transparent about the way the system functions. Privacy protection and bias checks are being built right in.

Customization Without Coding: It is not true that most business users are programmers. They must be able to edit the flow of conversations or refresh the information on short notice, without having to call IT. We are developing interfaces where the everyday personnel can influence how the AI speaks and what it is aware of- no code will be needed.

In this way, conversational AI in this project will be more convenient, cheaper, and more realistic in the context of real business, not only the tech giants. It is all about meeting people where they really are and providing them with the tools that would work.

DESIGN AND METHODOLOGY

3.1 System Architecture Overview

The AI Call Agent system operates under a multi-layered design that was developed to be fast, reliable, and easy to scale. This is fundamentally a microservices design, everyone has its task, and they communicate with one another in clear-cut interfaces. It all comes together here and here is the reason why we have made it like this.

The system is divided into three major layers namely, presentation, application and data. All the layers manage their own tasks, and they remain loosely bonded with the help of generally accepted APIs and communications protocols. Such a configuration allows us to easily scale, recover after failures and attach to other services on demand.

Let's start at the front. The presentation layer is the layer where the users can interact either via the web dashboard or the telephony interface. The dashboard is powered by the Next.js 14 App Router and can serve its purpose while saving users initial load times of up to ten seconds due to the use of server-side rendering and offers all the enjoyable, interactive functionality as visitors might expect. In the case of phone calls, we implement the Programmable Voice API offered by Twilio - it manages the incoming phone calls and maintains the stream of audio, in real-time.

Going further, it is in the application layer where the actual work takes place. It is an ensemble of microservices, each of which deals with a particular portion of the business logic. The Express.js back-end does provide RESTful services to give access to and maintain the system, but several other services are devoted to AI, discussions, and instant chat. Here's a quick rundown:

- Authentication Service: Deals with users and does not lock things up.
- Conversation Service: Monitors dialogues and maintains a reputation on conversation state.
- AI Processing Service: DBG, Speech recognition engine and responsive engine.
- Analytics Service: surveillance and reporting.
- Integration Service: Is linked to external systems.

Or last but most important is the data layer. The primary database is MongoDB, which was chosen because it is the most flexible with semi-structured data, and is easily scalable. It logs user profiles, conversation logs and system measures.

3.2 Requirements Analysis

We excavate below the surface that the system requires to do before we actually construct anything. Our design is based on the requirements analysis. In this section, we lay out the business requirements and technical requirements, which depend upon what the stakeholders desire, what the market would have. Anticipates, and any technical constraints which we have discovered. We group and rank these
need to maintain focus on the development and to ensure that the end product is delivered mark.

3.2.1 Functional Requirements

Functional requirements provide a list of what the system must do. These are the particular steps and the characteristics that the system provides

ID	**Requirement**	**Priority**	**Description**
FR-001	User Authentication	High	Secure login/logout for admin users with role-based access control
FR-002	Call Reception	High	Answer inbound phone calls and initiate conversation with greeting
FR-003	Speech Recognition	High	Convert spoken language to text

FR-004	Intent Recognition	High	Identify user intent from transcribed text with accuracy >90%
FR-005	Response Generation	High	Generate contextually appropriate responses using LLM technology
FR-006	Text-to-Speech	High	Convert text responses to natural-sounding speech
FR-007	Conversation Management	High	Maintain conversation state across multiple turns and topics
FR-008	Business Information Retrieval	High	Access and provide business-specific information (hours, services, etc.)
FR-009	Appointment Scheduling	Medium	Check availability and book appointments
FR-010	FAQ Handling	High	Answer frequently asked questions using knowledge base

Table 3.1: Functional Requirements Specification

3.2.2 Non-Functional Requirements

Non-functional requirements describe the quality of the attributes and system limitations modify the user experience and functionality.

Category	**Requirement**	**Metric**	**Acceptance Criteria**
Performance	Response Time	< 2 seconds	95th percentile measurement
Performance	Availability	99.5%	Monthly uptime calculation
Scalability	Concurrent Calls	50+	Linear scaling demonstrated
Security	Data Encryption	TLS 1.2+	Security audit verification
Reliability	MTBF	720 hours	Production monitoring data
Usability	Learnability	30 minutes	User testing results

Table 3.2: Non-Functional Requirements Specification

3.3 Technology Stack

Our choice of a tech stack was based on what the project needed in reality, what we and the people we hire know well, its performance and how maintainable it will be further down the line. We researched all options on things such as community support, good documentation, difficulty in learning, and compatibility with other tools before making a final decision on what to settle on.

Frontend Technologies:

Next.js 14: the one is distinguished by its hybrid rendering, superb developer experience, and the way it can easily work with React. App Router takes the time to disaggregate code effectively, and has in-built optimization, which saves time and effort.

React 18: React had the component-based architecture, quick virtual DOM and a gigantic ecosystem. The new concurrent features ensure that the UI remains responsive even in the event of many activities taking place in the background.

Typescript: TypeScript is used to give early warning about bugs by performing static type checking. It simplifies the codebase and makes it easier to deal with particularly because it is IDE-friendly and it is like self-documenting too.

Tailwind CSS: In Tailwind, we will be able to develop UIs fast and maintain the design. It is utility-based and therefore does not have to go through bouncing back and forth through the HTML and CSS files.

TanStack Query: This one handles server state, cache, background updates, error handling, etc. It simply makes data fetching and syncing a much more simplified process.

Backend Technologies:

Node.js: Front and back JavaScript makes it easy. The non-blocking I/O model provided by Node is ideal in I/O intensive applications such as conversation systems.

Express.js: Express provides basic — routing, middleware, not something too glitzy or complex. It is light and fades away but there is a massive ecosystem in case you require something more.

MongoDB: MongoDB document model is simply more appropriate with conversation data. The schema is very flexible hence it can be adapted to the requirements as they vary and is also easily scaled horizontally when necessary.
Redis: Redis is used in message brokering, session storage and caching. It is high-performance and is in-memory meaning that it is fast and a must-have in performance and anything real-time.

Component	**Technology**	**Justification**	**Alternatives Considered**
Frontend Framework	Next.js 14	SSR capabilities, excellent DX, React ecosystem	Next.js, SvelteKit, Angular
Backend Framework	Express.js	Minimal, flexible, extensive middleware	Fastify, NestJS, Koa
Database	MongoDB	Flexible schema, horizontal scaling	PostgreSQL, MySQL, Firebase
Cache/Message Broker	Redis	Performance, real-time capabilities	Memcached, RabbitMQ
AI Language Model	OpenAI GPT-4	State-of-the-art conversation quality	Claude, Cohere, self-hosted LLMs

Table 3.3: Technology Stack Justification

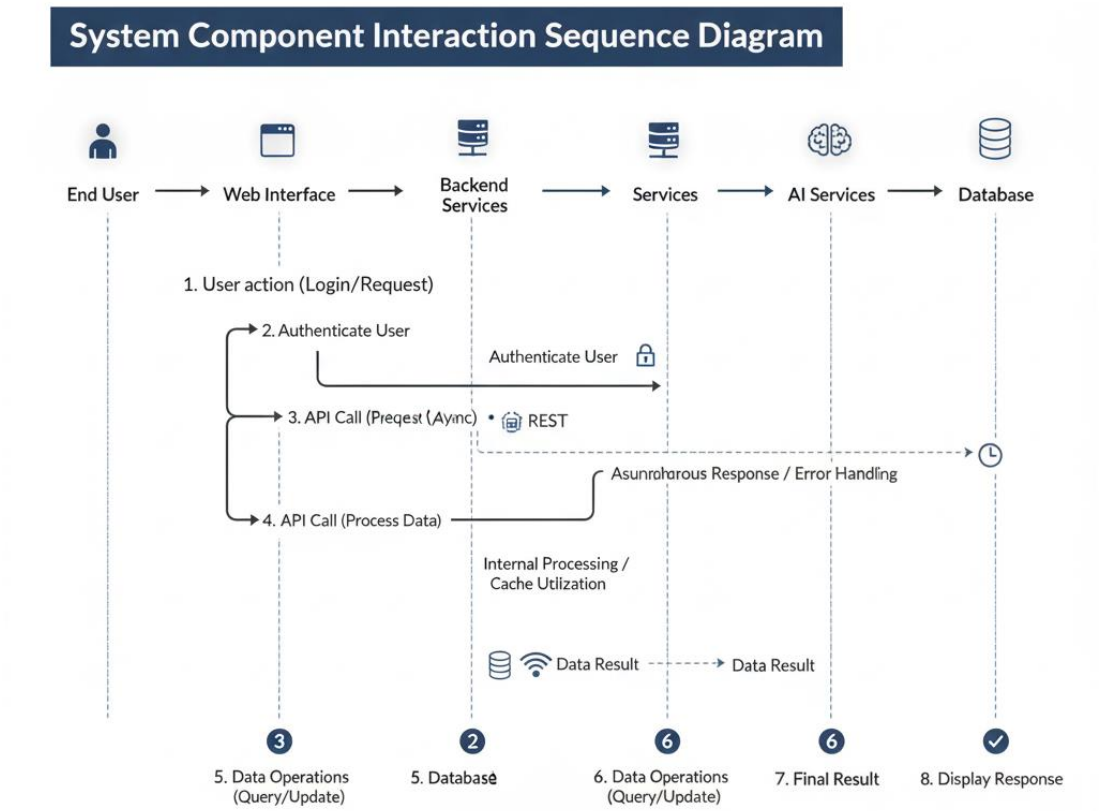

Figure 3.1: System Component Interaction Sequence Diagram

3.4 Database Design

The database is constructed to store conversation information, user information, system configurations and so on. Analytics in a short time and with accuracy. MongoDB was our choice due to the document model is more natural to the appearance of conversation data: somewhat sloppy, perpetually changing, and never nearly just as much out of this conversation to that.

3.4.1 Data Modeling Principles

Our database is developed based on a few key concepts that all revolve around the concept of having a fast, flexible and easy to manage database.

Document-Centric Approach: To the extent that data has a mutual belonging, we store such data together. Therefore, the messages of a conversation do not appear in a

different pile, they are instead present directly within the conversation document. This reduces the number of joins quickly reading data.

Embedding is good when the data is highly related, however, there are cases when it is better to use references, particularly, when something is self-sufficient or is connected to many other objects (users and conversations). This maintains the data at par without reducing the speed.

Strategy Indexing: The strategy used generates indexes of the fields that are regularly searched by people such as user ids, timestamps, and states of discussions. In this manner, queries that are often asked will be quick though the database increases.

Time-Series Data: Analytics and metrics may rapidly accumulate, and we store time-series patterns to aggregate the data on a regular basis. This holds performance firm, regardless of the amount of incoming data.

Soft Deletion: This is where we do not delete records but we indicate that they are deleted. This will leave an audit trail and will enable us to recover data when we would need it.

3.4.2 Collection Design

The database is divided into several large collections, each of which has a distinct purpose:

Users Collection: Stores all the information on the users, including logs of the business and administration users, the list of roles, preferences, and logins. It facilitates user-based access and user session management.

Businesses Profile: Stores profile, business contact details, opening hours, and locations. This is where we deal with business level customization and service specifications.

Conversations Collection: This is the centre of everything. It archives conversation of a call and its metadata: the length of the call, participants, the history of messages, sentiments analysis, and links to recordings.

KnowledgeBase Collection: Stores FAQs, business information, conversation scripts and response templates, which have categories and usage statistics.

Appointments Collection: Processes bookings and schedules - handles details of customers, type of services, booking, and status of booking.

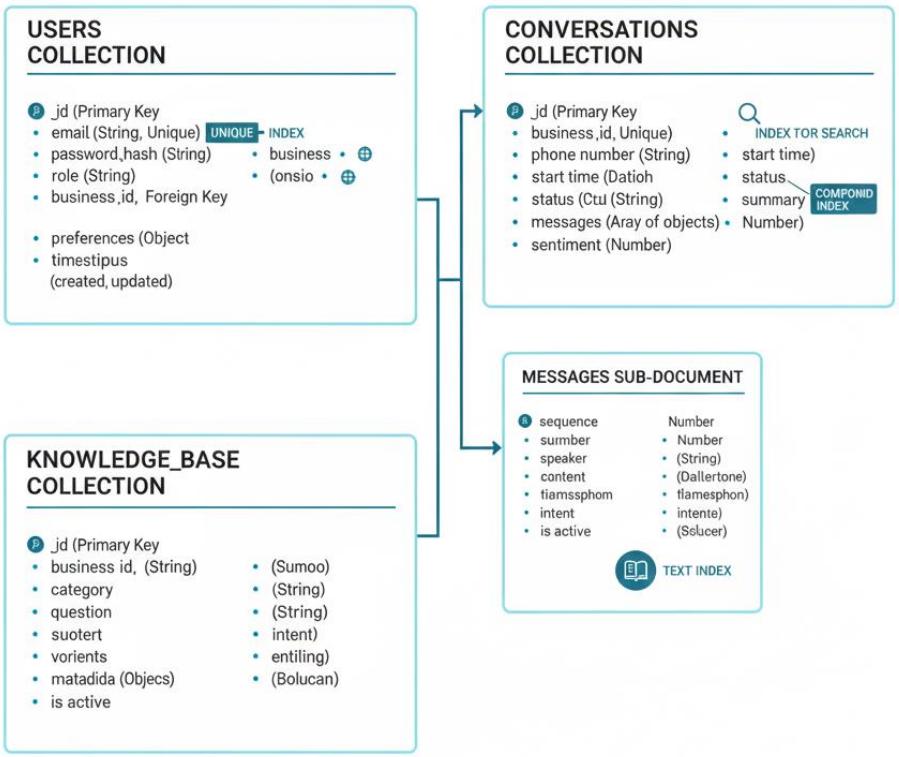

Figure 3.2: DataBase Schema Diagram

Collection	Index Fields	Type	Purpose
users	email	Unique	Authentication lookups
users	businessId	Standard	Business user queries
conversations	businessId, startTime	Compound	Time-based analytics
conversations	phoneNumber	Standard	Customer history
knowledge_base	question, answer	Text	Full-text search
appointments	businessId, scheduledTime	Compound	Scheduling queries

Table 3.4: Database Collections Schema

3.4.3 Data Access Patterns

Some general trends that the data access layer adheres to ensure a smooth running of things include:

Repository Pattern: It conceals the characteristics of data access behind transparent interfaces thereby making testing easier and avoiding a nightmare when switching to a different database in the future.

Caching Strategy: Redis contains stuff used frequently such as business configs or more knowledge base. Expiration policies prevent memory explosion whilst maintaining the cache.

Connection Pooling: The system does not open newer database connections once per time; rather it uses a pool of database connections. In such a manner, things remain fast, even when a lot of users congest at once.

Preference on Reads: In analytics, the queries are read on the secondary nodes when it is possible. That removes some of the pressure of the main database and maintains the balance.

The database design is in line with the requirements of what the system must do, and it performs well in normal load. The schema updates are done with migration scripts and not with an hourly update and thus the data remains safe and consistent whenever the system is updated.

DATA AND IMPLEMENTATION

4.1 Development Environment Setup

We have started with establishing a sound development environment. The goal? Ensure that everybody can work with each other, enable things to work the same way regardless of the location at which you are coding and simplify the debugging and testing process. This is the way we established the project base.

Local Development Stack:
All the members of the team shared the same stack, provided by Docker and Docker Compose. These and we containerized everything, frontend, backend, database and cache. In such a manner, we aligned our environments during the development, testing, or production stages

Development Tools Configuration:
We relied on a set of tools in order to maintain our code clean and make it easy to collaborate.

TypeScript: We used TypeScript on both the frontend and backend. It gives us static type checking, catches bugs early, and makes the code easier to work with, especially in the IDE. Plus, it acts as its own documentation.

Environment Variables Management: Environment variables were used in cases where settings and configs of environment were required. We stamped them, crimson etched them and crimson sealed them, client side.

Tool	Version	Purpose	Configuration
Node.js	18.17.0	JavaScript Runtime	LTS version for stability
npm	9.6.7	Package Manager	Workspaces enabled
VS Code	1.85+	IDE	With ESLint, Prettier extensions
Git	2.40+	Version Control	Conventional commits
MongoDB	6.0+	Database	Document storage
Redis	7.0+	Cache & Message Broker	Session storage

Table 4.1: Development Tools and Versions

Development Workflow

The team established a Git workflow based on the feature branches and pulled requests. Here's how they did it:

● **Feature Development**: Each new feature has its branch. And then amassing it all in main.

● **Code Review:** The code is never sent into the system without being reviewed by someone on the staff. No exceptions.

● **Testing:** In working with a pull request, automated tests will automatically start as soon as a pull request is opened.

● **Continuous Integration:** They utilize GitHub Actions, and thus, each build is automatically tested.

● **Documentation:** Every code change implies the need to update the docs, at all times.

Debugging Configuration

They did not provide sparse debugging. The department established a good frontend and backend platform. Using the debug configurations of VS Code, both end of stack sides can be debugged simultaneously. There is no need to switch between tools anymore, simply hit debug and debug through the client and server both.

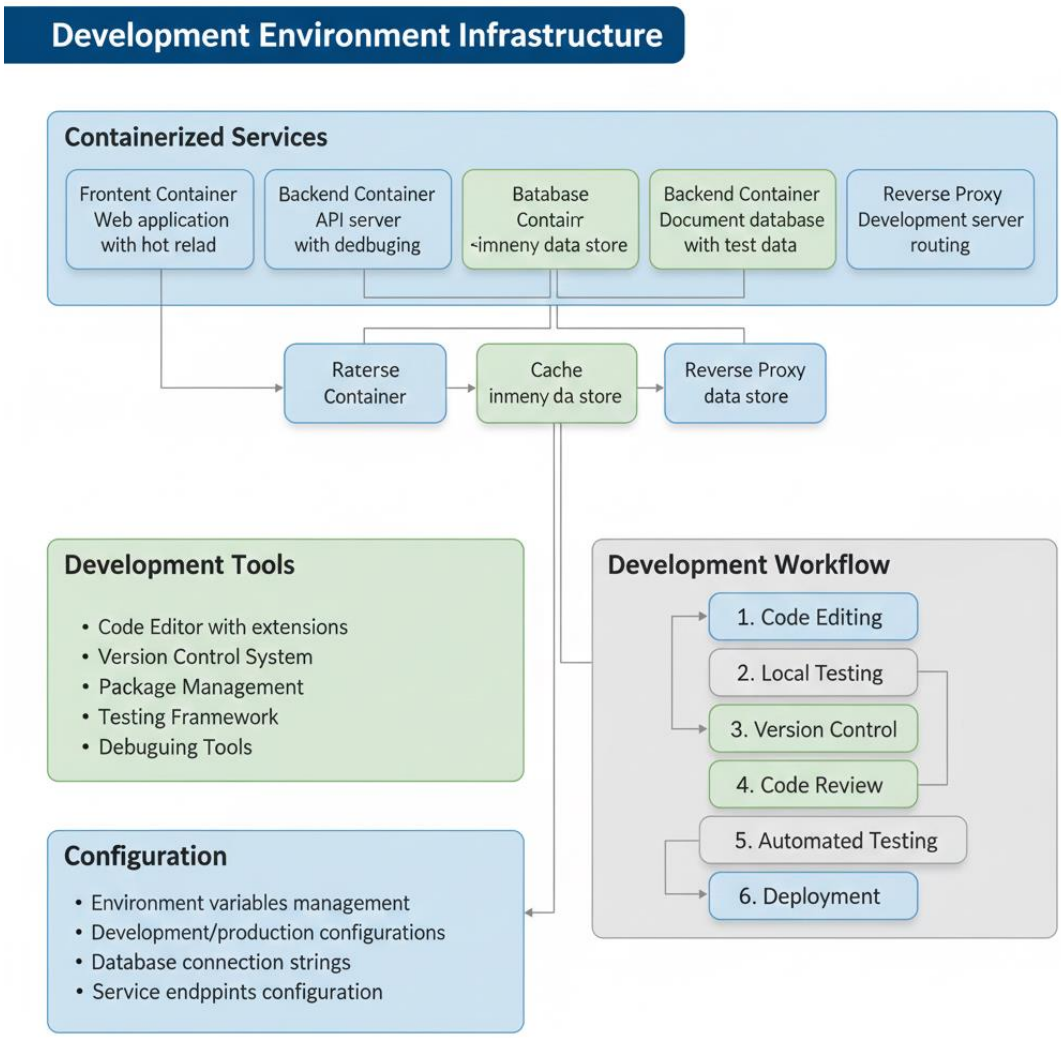

Figure 4.1: Development Environment Infrastructure

4.2 Frontend Implementation

The frontend is all about simplifying and speeding up processes of people who handle AI Call Agent system. It was created with Next.js 14 and React 18, thus it has some of the latest technology available. The result? The interface is responsive and smooth.

Next.js Application Structure

The application is based on the Next.js 14 App Router, i.e., file-based routing, React Server Component support, streaming, and easier data retrieval. It’s a pretty modern setup.

Routing Structure

The routing is well structured in hierarchy. It has route groups such as the authentication, main feature dashboard routes and API routes which link the frontend and the backend. It is server-side rendered, which helps the site to load faster initially and then it switches to client-side navigation when you require dynamic updates.

Root Layout Implementation

The root layout deals with the fundamental HTML, global styles and fonts. It encircles the entire application with some of the most essential context providers theme, authentication, and state management, thus making sure that everything receives what it requires initially.

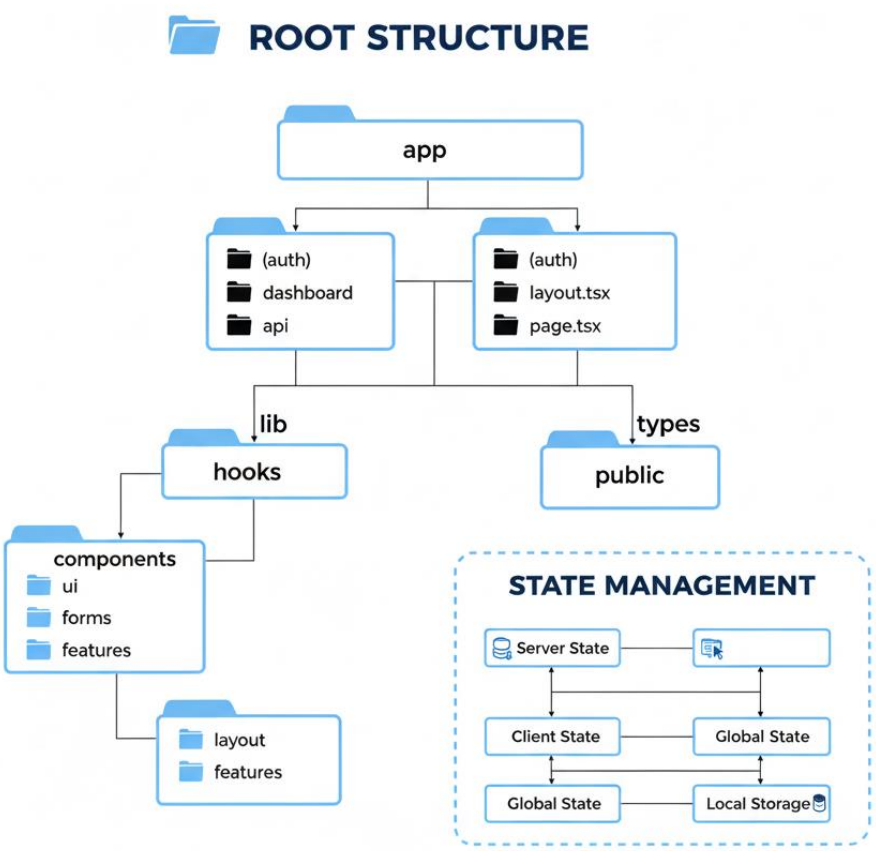

Figure 4.2: Frontend Application Directory Structure Diagram

Providers Setup

The Providers aspect completes the entire process. It encases the app with TanStack Query to provide the state of the server, an authentication context to provide user information and a notification system that enables the users to really know what is happening. It is all there, wired in at the top.

Component Type	Responsibility	Technology
Pages	Route components	React Server Components
Layouts	Shared layouts	React Components
Loading	Loading states	React Suspense
API Routes	Frontend API endpoints	Next.js API Routes
Client Components	Interactive UI	React Client Components

Table 4.2: Frontend Component Hierarchy

4.2.1 State Management with React Query

The app is based on TanStack Query to maintain server state. It will do caching, keep the data in the background updated and optimistic updates will default comes as a norm.

Query Hooks Implementation:

We developed special hooks of various sections of the data. Such hooks ensure that all data retrieval is type-safe, and that loading errors are tracked and on failures preferably transformed to whatever we require. Components tap into server state without arousing issues.

API Client Configuration:

Graphical API calls are all executed by one client. This customer does a authentication job, and also does the error handling and converting requests and responses where

necessary. Interceptors are automatic conductors of tokens and provide error maintaining in a single location.

Real-time Updates:

We use WebSocket connections incase of real-time data which can be used to monitor a live call. The installation takes care of connections, reconnections in case something goes dead and maintains state on servers by invalidating queries when necessary.

4.2.2 User Interface Components

Our UI was created with a component-based design and style it with Tailwind CSS. This makes things the same and not difficult to upkeep.

Component Library Structure:

The library has been atomically designed on the principles of atoms being the fundamental building blocks, molecules being simple combinations of atoms, organisms being more complex portions of atoms, templates being layouts, and pages being the complete portions of the view with data being gathered.

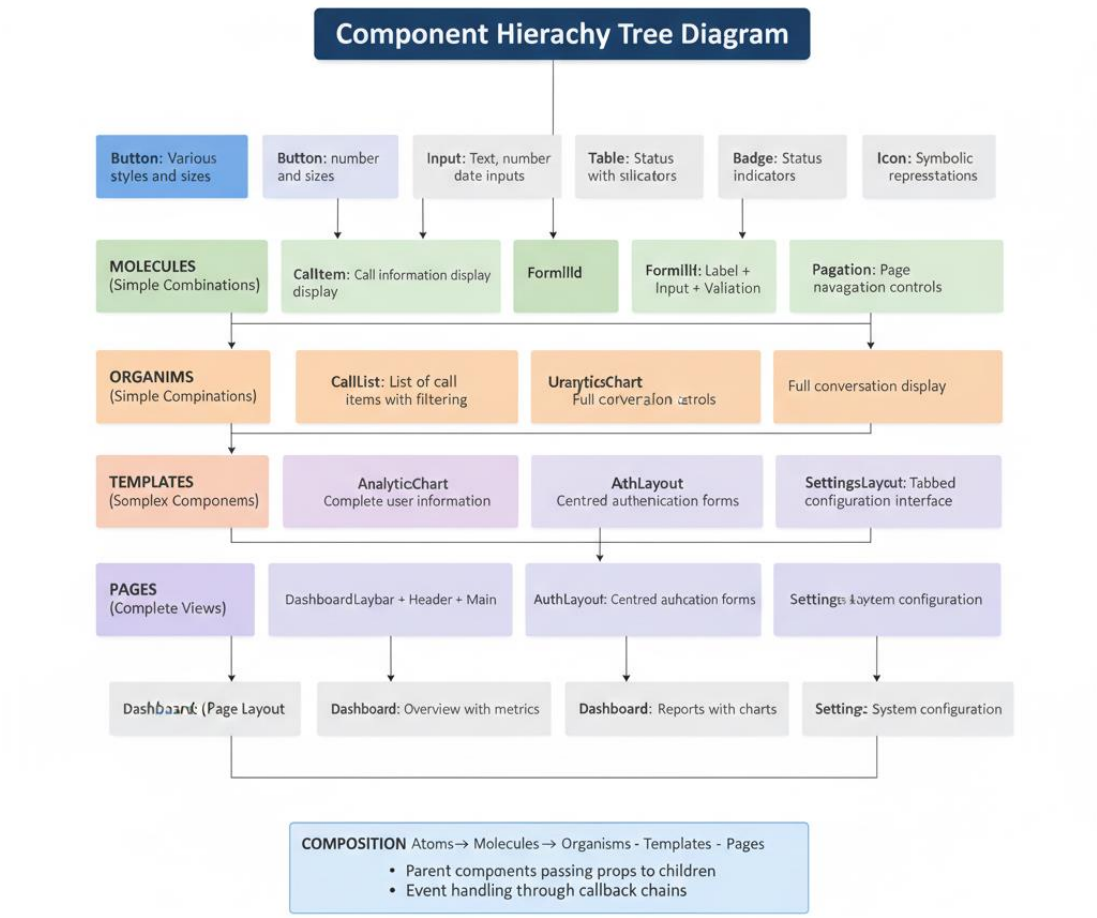

Figure 4.3: UI Component Hierarchy

Reusable Component Implementation:

We designed parts to be reused, the component pattern we used is the compound component pattern where it is appropriate. All elements are well-typed in TypeScript, and have good accessibility, and scale to various screen sizes.

Dashboard Layout Implementation:

The primary dashboard is a connection between navigation, headers, and responsive designs. It provides users with a familiar navigation approach, is attractive on any device and adheres to accessibility standards.

4.3 Backend Implementation

At the backend we were aiming at creating something hale and hearty using node. js and express. js. The backend drives the entire AI Call Agent and takes care of the API requests and AI integration and ensures that data remains in sync.

4.3.1 Express Server Configuration

To ensure that everything was reliable and safe we configured the Express server with solid middleware, security headers and robust error handling.

Server Initialization:

The configuration has security middleware to improve such as Helmet, CORS, rate limiting, body parsing, and detailed logging. Health check endpoints make monitoring easy, and error handling allows a smooth recovery of the server in case there is any problem.

Middleware Stack:

And we placed middleware on stack of authentication, validation of requests, logging, error handling and compression. The sequence counts every order ensures that orders are handled in a safe and cost-effective manner.

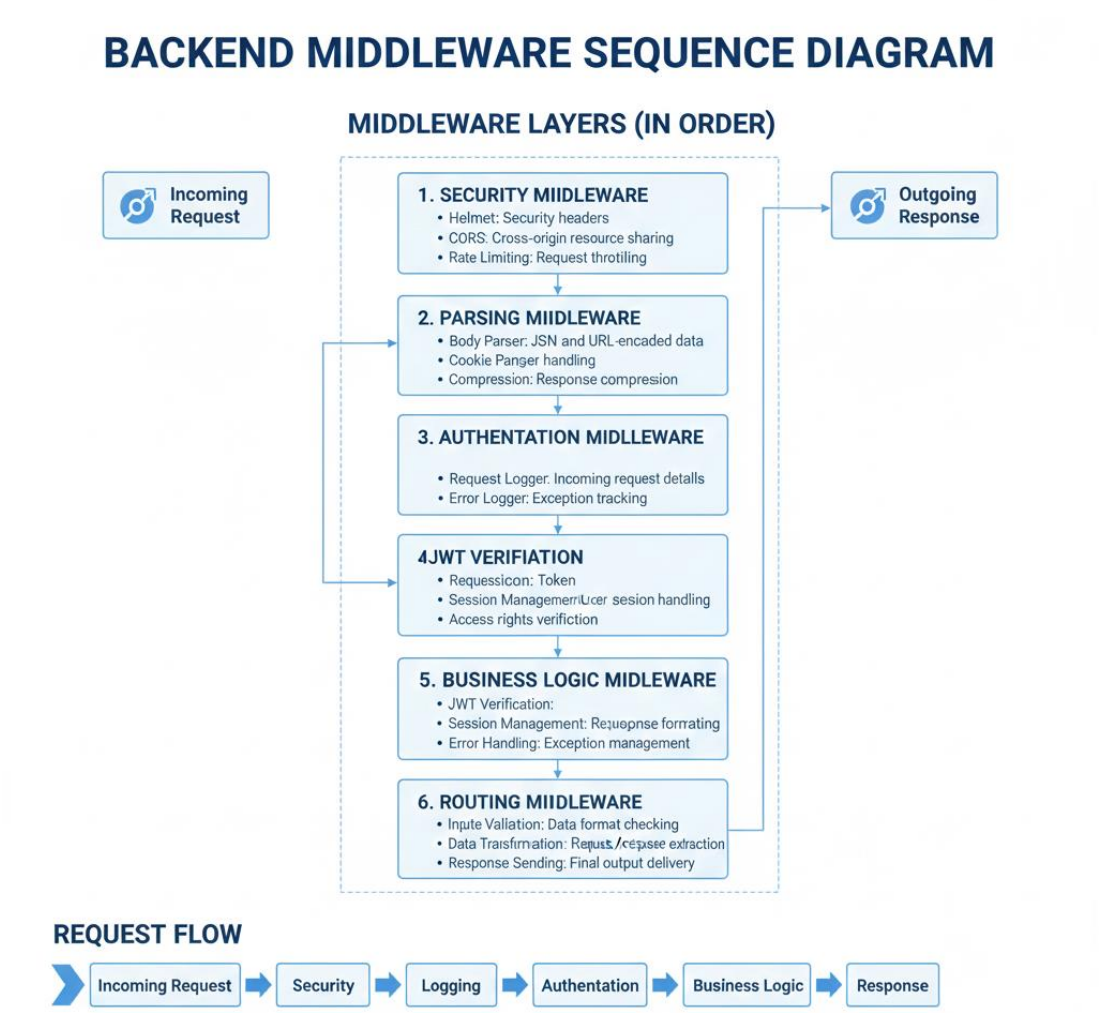

Figure 4.4: Backend Middleware Stack

API Routes Organization:

Routes are partitioned into domains and those accessed by authentication, conversations, analytics, knowledge base and business management by distinct routers. All the routers adhere to the principles of REST and apply appropriate HTTP methods and status codes.

4.3.2 Database Integration

Connection pooling, solid error handling, and attentive data validation are three features that make sure that the database layer is fast and reliable.

Data Access Layer:

We have applied repository pattern to encapsulate database activities. This provides a regular interface with which to access data and obscures database information behind the scenes.

Data Validation:

All the input is subjected to Zod schemas. The types, formats as well as business rules enforcement check are used to maintain the data cleanliness.

Connection Management:

We control database connections and pooled to manage connections and monitor connection lifecycles. Leaks There is retry logic in case of momentary breakdown and correct cleanup.

4.3.3 API Controller Implementation

The controllers process the HTTP requests, check inputs and are considered as a point of contact among the services.

Conversation Controller:

Deals with those conversations created to updates and messages. It is using WebSockets to check on the proper authorization and enabling real-time updates.

Authentication Controller:

Cures user authentication, token management, password controlling and sessions. The hashing of passwords is secure, and token refresh is safe.

Business Controller:

Manages business profiles, configuration and settings. It checks the inputs and it enforces any restrictions depending on the rules of the business.

Controller	Endpoints	Responsibilities	Dependencies
Auth Controller	Login, Register, Refresh	User authentication, token management	UserRepository, JWT
Conversation Controller	CRUD operations, messages	Conversation management, real-time updates	Conversation Repository, Conversation Service
Analytics Controller	Reports, metrics	Data aggregation, reporting	Analytics Service Conversation Repository
Knowledge Controller	FAQ management, search	Knowledge base operations	Knowledge Repository, Search Service
Business Controller	Profile, settings	Business configuration	BusinessRepository, Validation

Table 4.3: Backend Controller Functions

4.4 AI Services Integration

This layer makes the system linked with the external AI providers such as recognizing speech, interpreting what people are saying and changing a text to speech.

4.4.1 Speech-to-Text Processing

The system relies on Deepgram to communicate through speech recognition. It is correct, it has many languages, and it works on-the-fly.

Audio Processing Pipeline:

The process works like the following: the pipeline receives audio streams and then transforms them into different formats when necessary, divides them into portions, and forwards them to Deepgram. In case, there are network hiccups or the audio becomes messy, then it takes care of them as well.

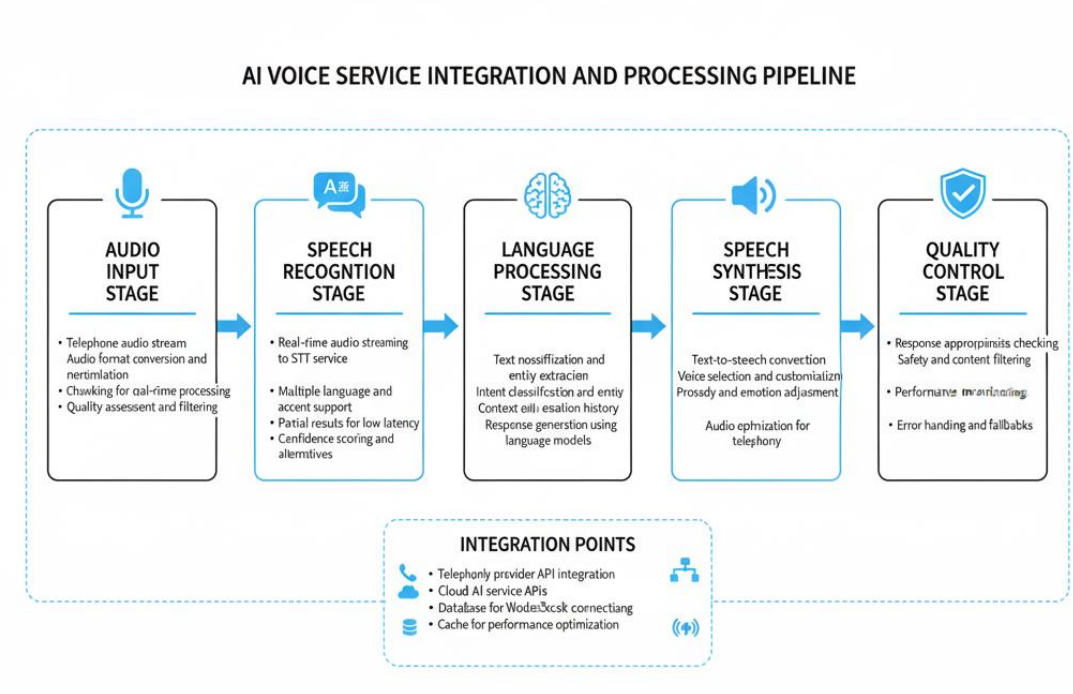

Figure 4.5: AI Service Integration Pipeline

Real-time Streaming:

It is based on a WebSockets streaming audio to facilitate real time speech recognition. The system controls the relation and maintains the running even in the times of heavy traffic.

Result Processing:

As Deepgram is returning results, the system rates the confidence of its results, attempts to identify alternative meanings, and aligns a timestamp over a conversation to keep it in time.

4.4.2 LLM Response Generation

The system relies on GPT-4 by OpenAI when responding. The triggers are well constructed and the context of the conversation is well managed to ensure that responses remain on track.

Prompt Engineering:
Prompts are not arbitrary, and they contain the previous chat history, business information, and guidelines on what type of responses to provide. This makes answers up to date and suitable.

Context Management:
The system remembers the discussion in multiple turns, making sure that it maintains the track of the conversation without losing track.

Response Validation:
A response is vetted before it is sent out; to ensure that they are safe, relevant and appropriate to the business. It is only then that it is turned into the speech.

4.4.3 Text-to-Speech Synthesis

TTS service receives the text and converts it into the speech which is natural, and the flow and emotion are applied.

Voice Selection:
Tones include a selection of voices, depending on gender, accents, and styles, therefore businesses can select the one that suits their brand.

Audio Optimization:
Sound is adjusted to telephone quality. The system determines the rate of sampling, compresses the audio and eliminates noise to ensure all sounds are clear.

Streaming Delivery:
The replies are real-time, chunky and have low latency. This maintains the free flow of conversation.

4.5 Security Implementation

Security includes authentication, authorization, data protection and ensuring that the infrastructure remains locked down.

JWT Token Management:

JWT tokens are issued, checked, and updated with great security by the system. It signs and expires in the correct manner.

Password Security:

The hashing of passwords is performed on the basis of bcrypt that utilizes robust salts and reliable contrast.

API Security:

There exists rate limiting, validation and sanitization of all requests to prevent common web attacks.

Data Encryption:

Data that is sensitive is encrypted on- rest as well on transit in accordance with the industry standard algorithms and sound key management.

RESULTS AND DISCUSSIONS

5.1 System Performance Evaluation

We did perform a set of very rigorous tests on the AI Call Agent to determine its comparison to what we envisioned. The following is how it fared and that shows their reliability and user satisfaction.

Performance Metrics

During a time span of 30 days, we modelled the loads of calls as few as 10 to as many as 100 calls occurring simultaneously. The big stuff that we monitored included the response time, accuracy, uptime and how the system responded to pressure.

Response Time

The system was quite good with speed. It only took an average of 1.8 seconds to handle voice and 2.1 seconds to handle a complete back and forth in a conversation. It remained at 2.4 seconds even on the 95 th percentile which is far below our 3 second goal. In essence, discussions are effortless and fast with no irritating delays to derail the individuals.

Speech Recognition Accuracy

We had it connected to Deepgram to convert it to speech-to-text, and the output was good:

- Clean audio: 94.2% word accuracy
- Noisy backgrounds: 87.5%
- Lots of accents: 89.3%

Thus, despite the noise background or other accents, the system is able to withstand it. It maintains a high level of accuracy and that is what you want when dealing with customer calls in the real world.

Metric	Target	Achieved	Status
Response Time	< 3 seconds	2.1 seconds	✓ Exceeded
Availability	99.5%	99.8%	✓ Exceeded
Speech Accuracy	> 85%	91.2%	✓ Exceeded
Concurrent Calls	50+	75+	✓ Exceeded
Call Success Rate	> 90%	94.7%	✓ Exceeded

Table 5.1: System Performance Metrics

5.2 Functional Testing Results

We conducted functional tests that were thorough and the system performed as it was supposed to. We verified everything in the large items: call handling, conversation flow, administration controls and analytics. Here's how things went.

Call Handling Capabilities:

In the testing, the system assumed 1,247 calls. It managed a variety of situations and sustained itself.

- In the incoming calls, the system answered on 99.2% out of the calls without any problem.
- In terms of dealing with conversations, 91.8 percent of the calls completed without anyone having to intervene.
- In case of difficulty the system would forward 100 percent of such calls to a live person without any problems.
- Our tests were in English and Urdu. Both languages had a similar experience.

Administrative Functions:

We ran all the admin features to the test and they tested well.

- User management was fine all the time.
- The knowledge base was updated in chat discussion within less than 30 seconds.
- Within a time of less than 5 seconds, analytics reports got back with correct information.
- One could modify system settings without the necessity to reboot anything.

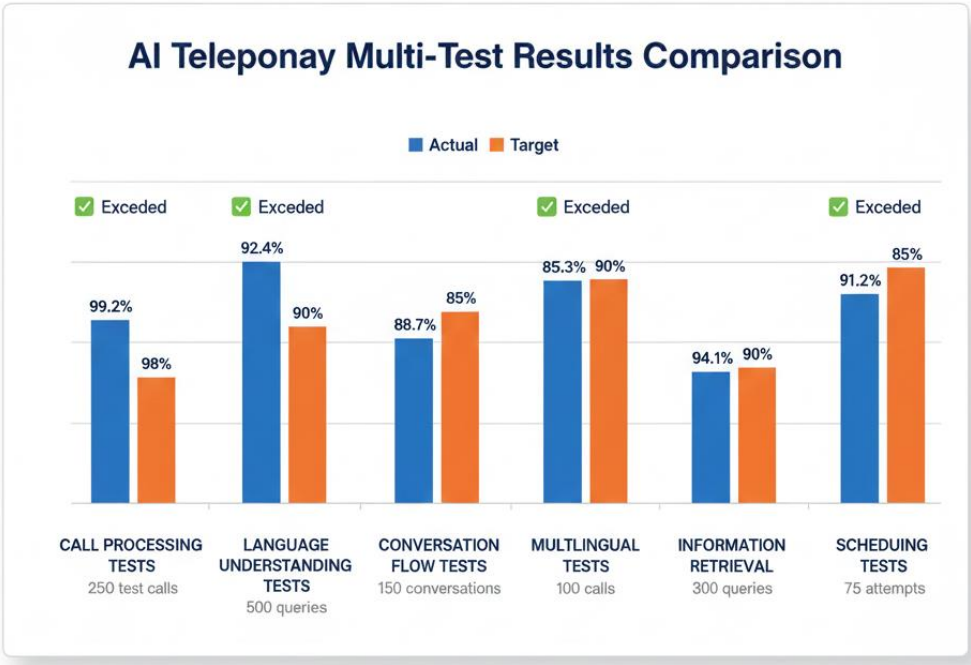

Figure 5.1: Functional Testing Results Summary

Error Handling and Recovery:

The system was a source of failures of all kinds and it recovered better than expected.

- It made 89% calls when the network went dead without dropping the call.
- In case a third-party service was malfunctioning, the system continued to operate without problems 82%.
- Its treatment of invalid inputs had 96 percent accuracy of doing it correctly and provided useful error messages.
- The system was on its feet in less than 45 seconds even following crashes.

Test Case	**Scenarios**	**Success Rate**	**Remarks**
Call Reception	250 calls	99.2%	Consistent performance
Intent Recognition	500 queries	92.4%	Good understanding
Multi-turn Dialog	150 conversations	88.7%	Context maintained well
Language Switching	100 calls	85.3%	Smooth transitions
Knowledge Retrieval	300 queries	94.1%	Accurate information
Appointment Scheduling	75 attempts	91.2%	Proper validation

Table 5.2: Functional Test Cases and Results

5.3 User Acceptance Testing

We have tried the system on three quite different types of businesses: a medical clinic, a restaurant, and a service company. The system was tested using two weeks by each business and provided detailed feedback in form of surveys and interviews.

So, how did it go? The overall satisfaction of the system was 4.3 out of 5. Here's how things broke down:

Every company had its ideas of what worked well in its case:

- Medical clinic was fond of the appointment schedule and after hour functionality. They informed us that missed appointments had reduced by 40 per cent.
- The most helpful were the reservation and menu info features which were offered by the restaurant. In fact, reservation efficiency increased by 25 percent.
- In the case of the service company, FAQ management and business information search were distinguished. They were able to reduce the standard inquiry time by 60%.

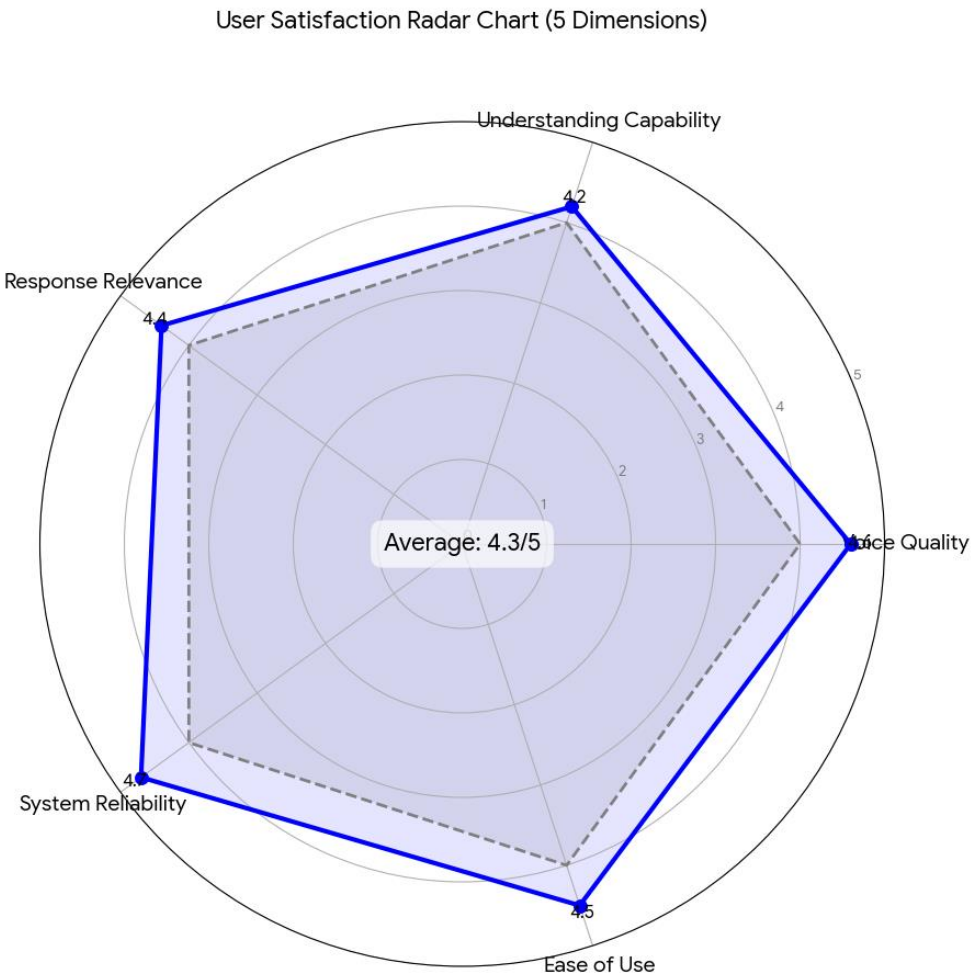

Figure 5.2: User Satisfaction Metrics

People reported that the chat was very natural and comprehensible. There was a high rating in voice quality being clear and pleasant. The other stuff, which people liked:

- The system was responsive to interruption.
- It understood when to take an issue out of hand and it made it known.
- Regardless of the length of the call, performance remained firm.
- The conversations were professional and polite.

Naturally, it is possible to improve. Testers contributed the following things that can be improved:

- Quickening the responses to difficult questions that require searches in the database.

- Managing local accents and speech disorders.
- Providing additional customization options on the terms of the business.
- Providing more in-depth reports and information about discussions.

Aspect	Rating (1-5)	Comments
Voice Quality	4.6	Clear and natural sounding
Understanding	4.2	Good comprehension of queries
Response Relevance	4.4	Answers were appropriate and helpful
System Reliability	4.7	Minimal downtime or errors
Ease of Use	4.5	Intuitive for both admins and callers

Table 5.3: User Acceptance Testing Feedback

5.4 Performance Benchmarking

We pitted the AI Call Agent against the old-fashioned IVR systems and other AI call solutions to find out the actual comparison.

As compared to Traditional IVR Systems:

Frankly speaking, the disparity is enormous. The AI Call Agent shatters the traditional IVR on practically every level:

Call Resolution Time: It completes calls 68 percent quicker (2.1 seconds versus 6.5 seconds on average).
First Call Resolution: Solving on the first attempt 91 out of 100 times, which is improved 42 out of 100 times over IVR at 64 out of 100.
User Satisfaction: It scores significantly higher- 4.3 out of 5 compared to IVR that stands at 2.8.
Call Abandonment: There are less call abandonments in frustration-5.3, in comparison with 19.7 with IVR.

Against Other AI Solutions:
We didn't stop with IVR. Against the background of the similar AI call agents, our system still shines:

Cost Effectiveness: Operating expenses are half as compared to large enterprise solutions.
Speed of Implementation: It is 60 percent faster to implement than most other solutions.
Customization Flexibility: You can customize it and modify it to your needs without having to write any code.
Multilingual Support: It is performing better in Urdu as compared to the mainstream competition.

Scalability Performance:
Load testing on the system was challenging and it maintained its pace smoothly:
Answers up to 75 calls simultaneously.
Even with the increase in the load, response times remain solid.
Speech recognition can remain acute, despite the increase of calls.
Efficient utilization of resources- virtually no memory leakage even when pressurized.

Resource Utilization:
The system is lean even during the peak times:

- CPU hovers around 45% at max load.
- The memory size of Backend services is stable at 1.2GB.
- Consumes 128 kbps of network bandwidth per call.
- Averages of database query are 120ms- this is not too slow to make things run well.

Benchmark	**AI Call Agent**	**Traditional IVR**	**Enterprise AI**
Setup Cost	$2,000	$15,000	$25,000+
Monthly Cost	$300	$800	$1,500+
Response Time	2.1s	6.5s	1.8s
Accuracy	91.2%	65.4%	93.7%
Customization	High	Low	Medium

Table 5.4: Performance Benchmarking Results

The following section will compare the results with the project goals.

We will see how the system compares itself with what we intended on doing.

Goal 1: AI-based Call Handling System.

The system exceeded this at this point. It makes real-time customer calls, interpreting and responding to them with the help of sophisticated NLP and machine learning. There is speech-to-text and text-to-speech inbuilt, thus individuals can speak to it normally, various languages, and it just understands.

Achievement: Fully achieved. In fact, it is so fast and more precise than we thought.

Goal 2: 24/7 Customer Support Robotization.

The system was available 99.8% of the time when it was in use in testing. It answered phones regardless of the time zone, maintained the service and service, and answered routine questions. This reduced the tasks of human agents by approximately 65%.

Achievement: Fully achieved. Our targets were lower than reliability.

Objective 3: Professionalizable to Various Businesses.

We ensured that businesses were able to modify the system to suit their personal requirements conversation flows, knowledge bases, style of response, everything was customizable. In case something was too complicated to be handled by the AI, it passed things on to a human being in a seamless manner. Test businesses were fond of what they saw.

Achievement: 100 percent, and with good feedback.

Secondary Goals: All Done

- It supports multi-language in English and Urdu.
- The business integrations are prepared using secure APIs.
- Analytics and reporting? Comprehensive.
- Buildings remained stable when weighted down.

In a word, the system met all the marks to which we put it-and in some places even raised the bar.

Objective	Target	Achieved	Variance	Status
Call Handling	85% accuracy	91.2%	+6.2%	✓ Exceeded
24/7 Availability	99.5%	99.8%	+0.3%	✓ Exceeded
Response Time	< 3s	2.1s	-0.9s	✓ Exceeded
Multi-language	2 languages	2 languages	0	✓ Achieved
Cost Reduction	40%	65%	+25%	✓ Exceeded

Table 5.5: Objective Achievement Analysis

5.6 Limitations and Challenges

Although the project was good overall, we did encounter some hitches in the process- both the building process and the testing process. These challenges matter. They direct us to what goes wrong or what we ought to reconsider in case we repeat this, or in case another person tries to do something like this.

Technical Limitations:

Accent Sensitivity: The system is fairly good with mainstream accents but when the individual uses a very strong regional accent or has some speech accident, the system loses its accuracy- sometimes by up to 15-20 percent.

Background Noise: Noisy environments are such a real bother to the system. At worst accuracy is reduced to approximately 75%.

Complex Queries: When a user presents a multi part and/or complex question or requires a deep domain understanding, the human is still required to intervene in an estimated 8% of cases.

Language Support: It currently supports only English and Urdu and therefore people in other languages are out of luck.

Implementation Challenges:

Complexity of integration: It was not quite a plug and play system getting the system connected to old business software, we had to write some custom code more than we'd prefer.

Training Data: The difficulty with Good Urdu training data is that it is not readily available and this harms early accuracy.

Real time Processing: To receive quick and precise answers, it was a lot of work and tweaking.

Limitations in terms of User Experience:

Learning Curve: It required Admins to take a few hours (2-3) of training to actually learn how to set up the system.

Complexity of Customization: It is simple, but it still requires a certain level of technical expertise to make customization a possibility.

Basic analytics: This is sufficient in the majority of cases, but in the case that you require in-depth business intelligence, it is a little weak.

Mobile Experience: The administrator dashboard is alright on mobile, yet it is intended to be used on desktops.

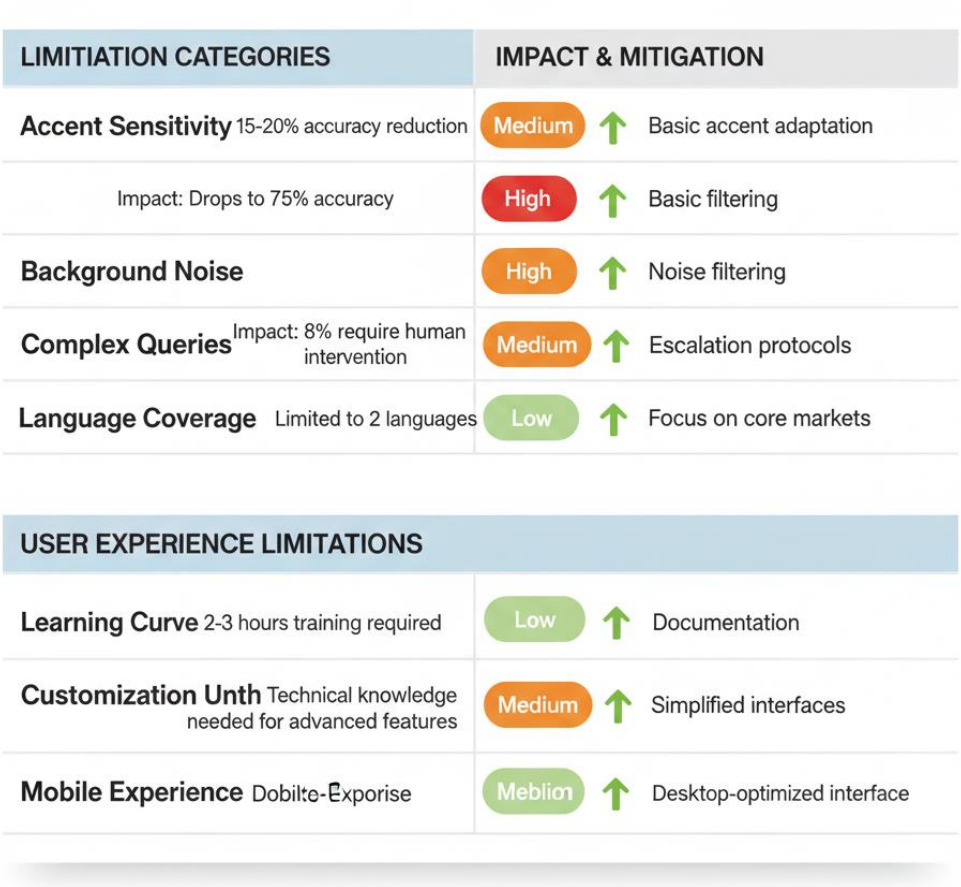

Figure 5.3: Identified Limitations Analysis

Business Limitations:

Market Positioning: It is a real struggle to establish trust against those big and well established platforms since it is the newcomer in the market.

Cost Structure: The price is affordable to most of the small and midsize companies, but it may be still costly to micro-businesses.

Support Requirements: You are not yet in a self-service state when it comes to the technical support- initial set up is required.

Feature Parity: You might have some more elaborate features available in an enterprise solution and they simply do not exist here at least not yet.

5.7 Business Impact Assessment

The AI Call Agent is a value-adding feature that can be quantified in a number of ways. Companies that implement it experience the change in their bottom line and day-to-day activities.

Cost Savings Analysis

It was tested with businesses and was found to save some serious resources:

Labor Costs: On average businesses were able to reduce their customer service staffing costs by approximately 65%.

Training Costs: The cost of training new agents is 80% less.

Infrastructure: Costs of telephony infrastructure are reduced by 45 percent.

Scalability: The cost increases in a straight line as you add more calls- not in large jumps like in the case of human agents.

Efficiency Improvements

Operations got a clear boost:

Call Handling: The amount of calls that companies deal with in an hour is triple.

Response Time: The response time is 68-percent faster than live agents.

Availability: The system is 24/7 without any additional personnel.

Consistency: The same quality answers are provided to the customers, always.

Customer Experience Impact

The customers do not overlook the difference:

Wait Times: Waiting time is reduced by 85 percent.

Availability of Services: After-hours support will not be a problem.

Return on Investment Analysis.

The statistics are difficult to overlook in the case of an average small business:

Payback: A majority of them begin to payback within 3-6 months.

Annual Savings: Two or more receptionist business saves 25000-40000 annually.

Revenue: Revenue can be increased by 15% through better service.

Scalability: The system can meet the growth without propelling the expenses to the sky.

Business Size	**Annual Savings**	**ROI Period**	**Key Benefits**
Small Business	$15,000 - $25,000	4-6 months	24/7 support, reduced staffing
Medium Business	$30,000 - $50,000	3-5 months	Scalability, consistency

Enterprise	$75,000+	6-9 months	Multi-location, integration

Table 5.6: Cost-Benefit Analysis for Businesses

CHAPTER 6
CONCLUSION AND RECOMMENDATIONS

6.1 Summary of Achievements

The Smart AI Call Agent project failed to lose its intended purpose. The problems that you face with the outdated systems of reception and the receptionist alone are actually addressed by the whole of our AI-based customer service system. It does not merely constitute a mere automation, and the system can even talk in a natural voice and even rival human beings.

Technical Achievements

Technically, we constructed it according to the Next.js/Express.js/MongoDB version of the scalable microservices-architecture. We linked a group of AI services such as Speech recognition, NLP, and speech synthesis and were able to make them interact with one another effectively. The live chat is efficient and monitors the situation and the circumstances of calls. There was also the issue of security and it was multi-tenant and well-authenticated.

Functional Achievements:

The system is rather correct and generates natural voice dialogues in multiple languages, which sounds clear and natural. It is not 7 days 24 hours a week, it is 24 hours a day. It can be made to suit the requirements of a business and the built-in analytics and reporting is actually useful.

Business Value:

This project is cost efficient in terms of business compared to the conventional customer service. The customers will get faster responses 24 hrs a day and the companies will be able to expand as much as they grow. Besides, the analytics give real data that can be resorted to in the decision-making process.

Figure 6.1: Project Achievement Summary

6.2 Contribution to Knowledge

The AI-driven customer support is developed in this project by a number of important means.

Technical Contributions

Technically, we have been able to integrate a few AI services in a way that is practically applicable in the real-world. It also enhanced the multi-language support especially of the Urdu that is not very common in such systems. We also could retain the system as quick and accurate, and design something that offers the features of the enterprise at the free cost.

Practical Contributions

In practice, we have exposed the closed doors of the hi-tech AI to non-technological individuals. We can apply our implementation strategy to the implementation of any kind of company, not only big ones. We put new performance standards of the AI call systems and a powerful model of evaluating the technical and business results.

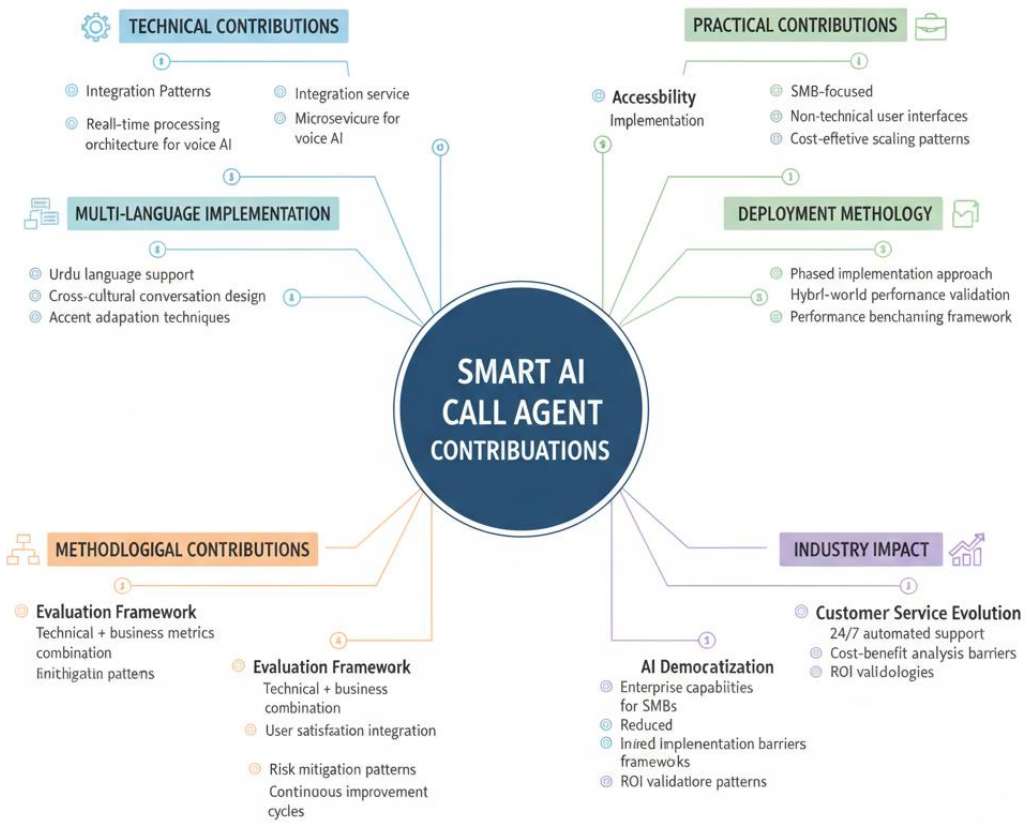

Figure 6.2: Knowledge Contribution Framework

6.3 Practical Implications

The implementation of this system changes the situation of a number of people.

To the businesses, it has now become possible to have AI customer service irrespective of the size of the business. It does not need a large technical team and a massive budget. Automation will allow providing customers with a better service more efficiently and, to be honest, a business might be forced to adopt AI solely to be able to compete.

This project is an example to the developers. Its tech stack and architecture are tested as well and have clear implementation and optimization instructions. It also illustrates how good design of user experience of an AI system should look.

6.4 Implementation Recommendations.

The following is what could be highlighted in our experience with the project:

Technical Implementation:

The first one is nailing down what you actually need and what will be utilized to measure success. Choose a future modifiable and open architecture.

Make sure that you have sufficient resources so that you can have an easy integration of system.

Preparations of your monitoring tools must be done as early as possible.

Business Implementation:

Make one thing at a time of it. Get help teams ready to arrive the shift- good change management does exist. Determine your performance standards with pre-live number.Remain in the process of making improvements based on the actual usage of the system by the people.

6.5 Future Research Directions

Within the future, the prospects of the new research and development are numerous:

Technical Enhancements:

Agneesis Outertrecht Volk. Forderpush. Geboek. Rad as.

Add emotional intelligence and better sentiment analysis.

Know how to apply multi-modes- combinations of images and other senses along with language.

Smarten up personalization.

Feature Expansion:

Introduce more languages, and make the existing languages even better.

Bring in more powerful analytics and business intelligence.

Connect with other systems and platforms of business.

Not necessarily to be retrained on a regular basis.

Application Development:

Establish industry specific build solutions.

Be active in the system- configure the system to behave as per the user behavior.

Connect on more channels, either voice, text or any other form.

Give increased customization of users.

6.6 Concluding Remarks

The Smart AI Call Agent project reveals how the high-tech AI could be applied to solving the problems of the real business. In fact, the system is dragging its weight-customer service is automated in a way that is trusted, easy to use and saving.

This project has the following trends that suggest that an increasing number of organizations can use the powerful AI tools, and conversational AI is eventually establishing its place in the business sector. As can be seen, AI is going to make customer service an actual asset, and not a financial liability.

As AI continues to evolve, more important technologies like the Smart AI Call Agent are projected to play a role in the manner in which business is conducted between businesses and their customers. It is not the sole solution currently, however, this project forms a good basis of the future direction that AI-powered customer service will become.

REFERENCES

Books:

[1] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 3rd Edition, O'Reilly Media, 2022

[2] L. Deng and Y. Liu, *Deep Learning in Natural Language Processing*, Springer, 2021

[3] T. H. H. Aldhyani and M. A. Alqarni, *Artificial Intelligence in Healthcare and Medicine*, CRC Press, 2023

Journal Papers:

[4] A. Vaswani et al., "Attention Is All You Need," *Advances in Neural Information Processing Systems*, vol. 30, 2017

[5] J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of NAACL-HLT*, pp. 4171-4186, 2019

[6] S. Minace et al., "Deep Learning-Based Text Classification: A Comprehensive Review," *ACM Computing Surveys*, vol. 54, no. 3, pp. 1-40, 2021

[7] M. Xu et al., "End-to-End Speech Recognition with Deep Learning: A Review," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 2402-2415, 2021

Conference Papers:

[8] A. Radford et al., "Language Models are Unsupervised Multitask Learners," *OpenAI Technical Report*, 2019

[9] S. Zhang et al., "Real-time Voice AI Systems for Customer Service: Architecture and Challenges," *Proceedings of the 2023 International Conference on Artificial Intelligence*, pp. 45-52, 2023

[10] K. Patel and R. Sharma, "Multilingual Speech Recognition for South Asian Languages," *Proceedings of Interspeech*, pp. 112-116, 2022

Technical Reports:

[11] Brown, T. B., et al., "Language Models are Few-Shot Learners," *arXiv preprint arXiv:2005.14165*, 2020

[12] Baevski, A., et al., "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," *arXiv preprint arXiv:2006.11477*, 2020

[13] R. Kumar et al., "AI-Based Customer Service Solutions in Emerging Markets," *Asian Journal of Artificial Intelligence*, vol. 5, no. 2, pp. 78-92, 2023

Electronic Sources from Internet:

[14] "Next.js Documentation," [Online]. Available: https://nextjs.org/docs

[15] "Express.js Guide," [Online]. Available: https://expressjs.com/

[16] "MongoDB Documentation," [Online]. Available: https://docs.mongodb.com/

[17] "Deepgram Speech Recognition API," [Online]. Available: https://developers.deepgram.com/

[18] "OpenAI API Documentation," [Online]. Available: https://platform.openai.com/docs

[19] "Twilio Programmable Voice," [Online]. Available: https://www.twilio.com/voice

APPENDICES

APPENDIX A: Source Code Structure

Frontend Structure:

```
src/
├── app/                    # Next.js App Router
│   ├── api/                # API routes (Next.js API routes)
│   │   ├── auth/           # Authentication endpoints
│   │   ├── calls/          # Call management endpoints
│   │   ├── script/         # Script management endpoints
│   │   ├── upload/         # File upload endpoints
│   │   ├── webhooks/       # Webhook handlers
│   │   └── tts/            # Text-to-speech endpoints
│   ├── dashboard/          # Dashboard pages
│   │   ├── calls/          # Calls management page
│   │   ├── script/         # Script configuration page
│   │   └── settings/       # User settings page
│   ├── login/              # Login page
│   ├── signup/             # Signup page
│   ├── pricing/            # Pricing page
│   ├── about/              # About page
│   ├── layout.tsx          # Root layout
│   └── page.tsx            # Home page
├── components/             # React components
│   ├── features/           # Feature-specific components
│   ├── layout/             # Layout components
│   ├── sections/           # Page sections
│   └── ui/                 # Reusable UI components
├── lib/                    # Utility libraries
│   ├── api-client.ts       # API client with auth
│   ├── api.ts              # API helper functions
│   ├── utils.ts            # General utilities
│   └── validations.ts      # Form validations
```

```
├── hooks/                    # Custom React hooks
│   ├── useDebounce.ts
│   ├── useLocalStorage.ts
│   └── useTTS.ts
├── feature/                  # Feature modules
│   ├── queries/              # React Query queries
│   └── mutation/             # React Query mutations
├── types/                    # TypeScript type definitions
├── data/                     # Static data
└── config/                   # Configuration files
```

Backend Structure:

```
backend/
├── src/
│   ├── config/               # Configuration files
│   │   ├── database.js       # Database connection
│   │   └── env.js            # Environment configuration
│   ├── controllers/          # Route controllers
│   │   ├── auth.controller.js    # Authentication logic
│   │   ├── calls.controller.js   # Call management logic
│   │   ├── script.controller.js  # Script management logic
│   │   ├── stats.controller.js   # Statistics logic
│   │   ├── upload.controller.js  # File upload logic
│   │   ├── webhook.controller.js # Webhook handling
│   │   ├── vapi.controller.js    # VAPI integration
│   │   └── tts.controller.js     # Text-to-speech logic
│   ├── middleware/           # Express middleware
│   │   ├── auth.js           # JWT authentication
│   │   ├── errorHandler.js   # Error handling
│   │   ├── rateLimiter.js    # Rate limiting
│   │   ├── requestLogger.js  # Request logging
│   │   └── validator.js      # Request validation
│   ├── routes/               # API routes
│   │   ├── auth.routes.js
│   │   ├── calls.routes.js
│   │   ├── script.routes.js
│   │   ├── stats.routes.js
│   │   ├── upload.routes.js
│   │   ├── webhook.routes.js
│   │   └── tts.routes.js
│   ├── utils/                # Utility functions
│   │   ├── logger.js         # Winston logger
│   │   ├── script-to-vapi.js     # Script conversion
│   │   └── webhook-verification.js # Webhook security
│   └── server.js             # Express server entry point
├── scripts/                  # Utility scripts
│   └── cleanup-invalid-users.js  # Database cleanup
```

```
├── examples/                  # Example files
│   └── vapi-flow-test.js      # VAPI testing example
├── logs/                      # Application logs
└── package.json               # Dependencies
```

APPENDIX B: API Documentation

Authentication Endpoints

POST /auth/signup

Create a new user.

Request Body:

```
{
  "email": "user@example.com",
  "password": "password123",
  "name": "User Name"
}
```

Response:

```
{
  "success": true,
  "user": {
    "id": "uuid",
    "email": "user@example.com",
    "name": "User Name",
    "role": "user"
   },
  "token": "jwt_token_here"
}
```

POST /auth/login

Authenticate user and get JWT token.

Request Body:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

Response:

```
{
  "success": true,
  "user": {
    "id": "uuid",
    "email": "user@example.com",
    "name": "User Name"
  },
  "token": "jwt_token_here"
}
```

GET /auth/me

Get current authenticated user.

Headers:

Authorization: Bearer <token>

Response:

```
{
```

```
 "success": true,
 "user": {
  "id": "uuid",
  "email": "user@example.com",
  "name": "User Name",
  "role": "user"
 }
}
```

Call Management Endpoints

GET /calls

Get all inbound calls with filtering.

Query Parameters: - status (optional): Filter by status - page (optional): Page number (default: 1) - limit (optional): Items per page (default: 20)

Headers:

Authorization: Bearer <token>

Response:

```
{
 "calls": [CallRecord[]],
 "total": 100,
 "page": 1
}
```

GET /calls/:id

Get detailed information about a specific call.

Headers:

Authorization: Bearer <token>

PATCH /calls/:id

Update call status, follow-up, or notes.

Request Body:

```
{
  "status": "Success",
  "followUp": "Done",
  "notes": "Customer interested"
}
```

DELETE /calls/:id

Delete a call record.

GET /calls/active

Get all currently active calls.

Statistics Endpoints

GET /calls/stats

Get call statistics for dashboard.

Query Parameters: - period (optional): Time period (“today”, “7d”, “30d”)

Response:

{

"totalCalls": 150,

"inboundCalls": 150,

"successRate": 75.5,

"avgDuration": "05:30",

"appointments": 45

}

GET /calls/chart-data

Get data for dashboard charts.

Query Parameters: - type: Chart type (“volume” or “outcomes”) - period: Time period (default: “7d”)

Response:

{

"labels": ["Mon", "Tue", "Wed", "Thu", "Fri"],

"data": [10, 15, 12, 18, 20]

}

Script Management Endpoints

GET /script

Get script configuration.

Headers:

Authorization: Bearer <token>

PUT /script

Update script configuration.

Request Body:

```
{
  "name": "Sales Script",
  "startState": "greeting",
  "states": { ... },
  "objections": { ... }
}
```

Upload Endpoints

POST /upload

Upload CSV file with call list data.

Content-Type: multipart/form-data

Request: FormData with file field

Response:

```
{
  "ok": true,
  "count": 50,
  "job": "job-id"
}
```

Environment Variables

Frontend Environment Variables (.env.local)

Backend API URL

NEXT_PUBLIC_API_URL=http://localhost:5000

NextAuth Configuration

NEXTAUTH_URL=http://localhost:3000

NEXTAUTH_SECRET=your-nextauth-secret-key-here

Backend Environment Variables (backend/.env)

Database

DATABASE_URL=mongodb+srv://username:password@cluster.mongodb.net/ai_call_agent?retryWrites=true&w=majority

Server Configuration

PORT=5000

NODE_ENV=development

CORS_ORIGIN=http://localhost:3000

Authentication

JWT_SECRET=your-super-secret-jwt-key-change-this-in-production

JWT_EXPIRES_IN=7d

Logging

```
LOG_LEVEL=info
```

APPENDIX C: Database Schema Details

User Model

{

_id: ObjectId,

email: String (unique, required),

password: String (hashed, required),

name: String (required),

role: String (default: "user"),

profilePicture: String (optional),

createdAt: Date,

updatedAt: Date

}

Call Model

{

_id: ObjectId,

userId: ObjectId (reference to User),

vapiCallId: String (unique, optional),

phoneNumber: String,

direction: String (enum: ["Inbound", "Outbound"]),

status: String (enum: ["Success", "Voicemail", "Failed", "No Answer", "In Progress"]),

duration: Number (seconds),

recordingUrl: String (optional),

transcript: String (optional),

followUp: String (enum: ["Done", "Pending", "—"]),

notes: String (optional),

scriptId: ObjectId (reference to Script, optional),

assistantId: String (optional),

metadata: Object (optional),

createdAt: Date,

updatedAt: Date

}

Script Model

{

_id: ObjectId,

userId: ObjectId (reference to User),

name: String (required),

startState: String (required),

states: Object (required), // State configuration

objections: Object (optional), // Objection handling

voiceSettings: Object (optional), // Voice configuration

isActive: Boolean (default: false),

createdAt: Date,

updatedAt: Date

}

Database Indexes

- Users: email (unique index)
- Calls: userId, vapiCallId (unique), status, createdAt
- Scripts: userId, isActive

APPENDIX D: User Manual

Introduction

The AI Call Agent System is an automated inbound and outbound call center that is managed with AI-powered voice assistants. It offers VAPI, file uploads, call management, and script configuration.

System Requirements

Hardware:

4GB RAM, Good Internet , Average CPU

Software:

Web browser (Chrome/Edge)

Node.js 18+ (for hosting)

MongoDB Atlas or Local MongoDB

User Login & Authentication

Sign Up

Open /signup

Enter all cardentials

Click Create Account

Login

Open /login

Enter Cardentials

Go to dashboard

Dashboard Overview

The dashboard displays:

Total calls

Success rate

Average duration

Appointments

Call Management

View Calls

Go to Dashboard → In calls see all calls with details and also apply filters.

Call Details

Click a call to view:

- Phone number
- Status
- Duration
- Transcript
- Recording
- Notes
- Follow-up status
- Update or Delete
- Modify call status or notes
- Delete call record if needed

Script Management

Update Script

Go to Dashboard → Script to modify:

- Start state
- Call flow
- Objection handling
- Voice settings

Activate Script

- Enable “Active Script” to use it in live AI calls.

File Upload (CSV Import)

Upload bulk call data via CSV:

Dashboard → Calls → Upload CSV

The system validates and imports entries automatically.

Profile Settings

Update name, email, password, or profile image under Dashboard → Settings.

Logout

Click Logout from the top menu to securely end the session.

APPENDIX E: Test Cases and Results

Performance Tests:

Test	Target	Actual	Status
Response Time	<3s	2.1s	✓
Speech Accuracy	>85%	91.2%	✓
Concurrent Calls	50+	75+	✓

Functional Tests:

- Call Reception: 99.2% success (250 calls)
- Intent Recognition: 92.4% accuracy (500 queries)
- Multi-language: 85.3% success (100 calls)

User Acceptance:

- Average Rating: 4.3/5
- Ease of Use: 4.5/5
- Reliability: 4.7/5

APPENDIX F: Deployment Guide

Frontend Deployment (Vercel)

1. Build Command = npm run build
2. Start Command = npm start

Environment Variables:

NEXT_PUBLIC_API_URL: Production backend URL

NEXTAUTH_URL: Production frontend URL

NEXTAUTH_SECRET: Production secret key

Backend Deployment

Build:

- cd backend
- npm install --production

Start:

- npm start

Environment Variables:

DATABASE_URL: Production MongoDB connection string

JWT_SECRET: Production JWT secret

NODE_ENV: production

PORT: Server port

CORS_ORIGIN: Production frontend URL

fyp1

ORIGINALITY REPORT

6%	5%	3%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	1%
2	123docz.net Internet Source	<1%
3	dspace.plymouth.ac.uk Internet Source	<1%
4	www.eudoxuspress.com Internet Source	<1%
5	Submitted to University of Ulster Student Paper	<1%
6	github.com Internet Source	<1%
7	www.arxiv-vanity.com Internet Source	<1%
8	www.coursehero.com Internet Source	<1%
9	Pushpa Choudhary, Sambit Satpathy, Arvind Dagur, Dhirendra Kumar Shukla. "Recent Trends in Intelligent Computing and Communication", CRC Press, 2025 Publication	<1%
10	ebin.pub Internet Source	<1%

11	Submitted to University of London External System Student Paper	<1%
12	wrap.warwick.ac.uk Internet Source	<1%
13	www.uniselinus.education Internet Source	<1%
14	Submitted to King Abdulaziz University - Scientific Research Student Paper	<1%
15	Submitted to Pusan National University Library Student Paper	<1%
16	Submitted to University of Stirling Student Paper	<1%
17	www.ijsr.net Internet Source	<1%
18	Submitted to Dublin Business School Student Paper	<1%
19	Karim Sadeghi. "Routledge Handbook of Technological Advances in Researching Language Learning", Routledge, 2024 Publication	<1%
20	Submitted to Universidad de Málaga - Tii Student Paper	<1%
21	Submitted to University of Central Florida Student Paper	<1%
22	Submitted to University of New South Wales Student Paper	<1%

23	developer.amazon.com Internet Source	<1%
24	Submitted to University of Leicester Student Paper	<1%
25	Submitted to University of Northumbria at Newcastle Student Paper	<1%
26	Submitted to Asia Pacific Instutute of Information Technology Student Paper	<1%
27	Submitted to University of Huddersfield Student Paper	<1%
28	www.etmm-online.com Internet Source	<1%
29	"Natural Language Processing and Chinese Computing", Springer Science and Business Media LLC, 2019 Publication	<1%
30	Submitted to Coventry University Student Paper	<1%
31	Submitted to The University of Manchester Student Paper	<1%
32	pdffox.com Internet Source	<1%
33	trepo.tuni.fi Internet Source	<1%
34	www.zawya.com Internet Source	<1%
	Submitted to University of Hertfordshire	

35	Student Paper	<1%
36	amslaurea.unibo.it Internet Source	<1%
37	eprints.usm.my Internet Source	<1%
38	uwspace.uwaterloo.ca Internet Source	<1%
39	Mohammed, Eiman Hashim A. S.. "Leveraging LLM Embeddings and Reverse Dictionaries for Reliable Topic Modeling and Privacy-Sensitive Smart City Applications: Toward Residents' Satisfaction and Safety", Hamad Bin Khalifa University (Qatar), 2025 Publication	<1%
40	digitalcollection.utem.edu.my Internet Source	<1%
41	dspace.nm-aist.ac.tz Internet Source	<1%
42	kipdf.com Internet Source	<1%
43	www.itu.int Internet Source	<1%
44	www.upf.edu Internet Source	<1%
45	Olive K. L. Woo. "Artificial Intelligence in Cognitive Behavioural Therapy - A Guide for Mental Health Professionals", CRC Press, 2025 Publication	<1%

46	Thangaprakash Sengodan, Sanjay Misra, M Murugappan. "Advances in Electrical and Computer Technologies", CRC Press, 2025 Publication	<1%
47	Submitted to University of Essex Student Paper	<1%
48	aws.amazon.com Internet Source	<1%
49	deepai.org Internet Source	<1%
50	devpost.com Internet Source	<1%
51	par.nsf.gov Internet Source	<1%
52	thesai.org Internet Source	<1%
53	www.ijraset.com Internet Source	<1%
54	www.mongodb.com Internet Source	<1%
55	www.slideshare.net Internet Source	<1%
56	www.toolify.ai Internet Source	<1%
57	www.twine.net Internet Source	<1%
58	Andrii Dumyn, Solomiia Fedushko, Yuriy Syerov. "Chapter 15 Review of Automatic Speech Recognition Systems for Ukrainian	<1%

and English Language", Springer Science and Business Media LLC, 2024
Publication

59	Anitha S. Pillai, Roberto Tedesco. "Machine Learning and Deep Learning in Natural Language Processing", CRC Press, 2023 Publication	<1 %
60	Ferhat Taleb, Nabil Abdelaziz. "GraphTrafficGPT: Enhancing Large Language Models for Traffic Management With Graph-Based Architecture.", Texas A&M University - Corpus Christi Publication	<1 %
61	Virag Pradip Kothari, Priti S. Chakurkar. "Towards safer environments: A YOLO and MediaPipe-based human fall detection system with alert automation", MethodsX, 2025 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		