

# GENERATIVE AI-BASED CHATBOT FOR CUSTOMER IT SUPPORT AUTOMATION

Project ID: R25-036



## Group Final Report

Supervisor: Prof. Nuwan Kodagoda

Co-supervisor: Dr. Lakmini Abeywardhana

BSc (Hons) in Information Technology  
Specializing in Software Engineering

Sri Lanka Institute of Information Technology  
Faculty of Computing  
Department of Software Engineering

August 2025

# **GENERATIVE AI-BASED CHATBOT FOR CUSTOMER IT SUPPORT AUTOMATION**

BSc (Hons) in Information Technology Specializing in  
Software Engineering

Department of Software Engineering

Sri Lanka Institute of Information Technology

August 2025

# Group Details

Group ID: R25-036

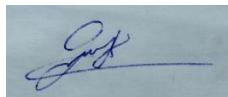
Components	Student ID Number	Name	Email
Multi Agentic System	Lakshan S. N. (Leader)	IT21800900	it21800900@my.sliit.lk
knowledge base construction for accurate retrieval	Gamage U. R	IT21807480	it21807480@my.sliit.lk
3D avatars with voice interaction and emotional intelligence	Fernando W.S.N.	IT21809224	it21809224@my.sliit.lk
Conversational form filing automation	Kuruppu K.A.V. U	IT21315282	it21315282@my.sliit.lk

Figure 1 : Group details

## Declaration

We hereby declare that this dissertation is our own work and does not incorporate, without due acknowledgment, any material previously submitted for a degree, diploma, or other qualification at any other university or institute of higher learning. To the best of our knowledge and belief, it does not contain any material previously published or written by another person except where acknowledgment is made in the text.

Furthermore, we grant the Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute this dissertation, in whole or in part, in print, electronic, or other media. We retain the right to use this content, in whole or in part, in future works such as articles, books, or other publications.



Signature: Date: 29/08/2025

Gamage U.R



Signature: Date: 29/08/2025

Lakshan S. N.



Signature: Date: 29/08/2025

Fernando W. S. N.



Signature: Date: 2025/08/29

Kuruppu K.A.V.U

Signature of the Supervisor: Date:

## Acknowledgement

We would like to express our deepest gratitude to our supervisor, Prof. Nuwan Kodagoda, for their invaluable guidance, insightful feedback, and continuous support throughout this research project. Their expertise and constructive suggestions greatly enriched our work and guided it to successful completion.

We are also sincerely thankful to our co-supervisor, Dr. Lakmini Abeywardhana, whose advice and encouragement were instrumental in refining our research and overcoming challenges along the way.

We would like to extend our appreciation to the members of the research viva panel for their thoughtful questions and constructive feedback, which helped enhance the depth and rigor of this dissertation.

Finally, we acknowledge our fellow students, colleagues, and anyone else who contributed in any way—through discussions, moral support, or collaboration—creating a motivating and supportive environment for our research.

## List of Figures

Figure 1 : Group details .....	2
Figure : Conversational form filling system diagram.....	27
Figure : Examples for Analysis.....	34
Figure : Keyword Matching.....	35
Figure : Semantic Similarity.....	35
Figure : Combined Scores.....	35
Figure : Business Rules Application .....	36
Figure : Key Differences Analysis.....	36
Figure - Knowledge base component.....	38
Figure - knowledge graph.....	41
Figure - Final evaluation result JSON.....	57
Figure - visualize monitoring to detect errors in each phase.....	58
Figure -query results .....	67
Figure - Redar Map.....	67

## List of Tables

## List of Abbreviations

IT - Information Technology

AI - Artificial Intelligence  
GPT - Generative Pre-trained Transformer  
GENAI - Generative Artificial Intelligence  
LOLC - Lanka ORIX Leasing Company  
NLP - Natural Language Processing  
LLM - Large Language Model  
DIET - Dual Intent and Entity Transformer  
RAG - Retrieval-Augmented Generation  
NLTK - Natural Language Toolkit  
BERT - Bidirectional Encoder Representations from Transformers  
NER - Named entity recognition  
UI - User Interface  
API - Application Programming Interface  
JSON - JavaScript Object Notation  
RAG - Retrieval-Augmented Generation  
LLM - Large Language Model  
Gen AI - Generative Artificial Intelligence  
BERT - Bidirectional Encoder Representations from Transformers  
NLP - Natural Language Processing  
TTS - Text-to-Speech  
API - Application Programming Interface  
LOLC - Lanka ORIX Leasing Company  
AZ - Azure (Microsoft cloud platform)  
UI - User Interface  
CLI - Command-Line Interface  
HTTPS - Hypertext Transfer Protocol Secure  
TLS - Transport Layer Security  
ESPNet - End-to-End Speech Processing Toolkit

UAT - User Acceptance Testing

GLB - Graphics Library Binary File Format

MAS – Multi Agentic System

API - Application programming interface

## Contents

Group Details .....	2
Declaration .....	3
Acknowledgement.....	4
List of Figures .....	5
List of Tables.....	5
List of Abbreviations.....	5
Abstract.....	9
Introduction .....	10
1.1.    Background.....	10
1.2.    Literature Review.....	12
1.3.    Research Gap .....	15
1.4.    Research Problem .....	18
1.5.    Research Objectives .....	21
1.5.1.    Main Objective .....	21
1.1.1.    Sub Objectives .....	22
2.    Methodology .....	26
2.1.    Methodology .....	26
Overview of the overall system.....	26
Individual components.....	27
B. Evaluation Framework.....	42
D. Efficiency Measures .....	43
E. System Baseline vs. Gains .....	44
F. Risk Mitigation.....	44
G. Best Practices and Architectural Patterns.....	44
2.2.    Commercialization aspects of the product .....	53
2.3.    Testing & Implementation .....	56
3.    Results & Discussion .....	62
3.1.    Results.....	62
3.2.    Research Findings.....	65
3.3.    Discussion.....	68
4.    Conclusion .....	70
References .....	70

## Abstract

This research project focuses on developing a generative AI - based chatbot for LOLC Company, designed to streamline and enhance their IT support systems. The chatbot integrates a range of features that enable employees to efficiently resolve IT-related queries, gain insights into HR policies, and access business-related information. By leveraging an advanced knowledge base, the chatbot provides precise and context-specific responses to user prompts, significantly improving operational efficiency and decision-making within the organization.

A central component of the system is its intelligent knowledge base, which allows dynamic updates through a user-friendly interface. This feature empowers staff to upload new documents, update existing policies, and refine content, ensuring that responses remain accurate and relevant over time. The knowledge base also enables personalized answers by learning from user interactions and retaining episodic memory of previous queries. This ensures a tailored experience for each user while continuously improving response accuracy based on feedback.

In addition to text-based support, the chatbot features an immersive 3D avatar equipped with voice-to-text and text-to-voice capabilities, enabling seamless interaction with employees. The system addresses knowledge gaps by auto-generating answers and providing guidance for completing forms, thus reducing manual effort and improving user efficiency. An agent-based automation system further enhances productivity by executing repetitive tasks based on instructions derived from the knowledge base.

This innovative solution is built to adapt dynamically to changing organizational needs while maintaining scalability and reliability. By combining personalized support, streamlined processes, and dynamic knowledge management, the chatbot sets a new benchmark for intelligent enterprise support systems, driving efficiency and improving the overall employee experience at LOLC.

# Introduction

## 1.1. Background

The rapid advancement of Artificial Intelligence and related digital technologies has profoundly reshaped the landscape of organizational support systems, particularly in the domains of Information Technology and Human Resources. Where employees once relied exclusively on human staff to troubleshoot technical issues or navigate HR processes, the introduction of intelligent automation has transformed these interactions into faster, more efficient, and more accessible experiences. Chatbots, which began as simple scripted systems capable only of responding to fixed frequently asked questions, have now matured into highly sophisticated conversational agents. These systems are no longer limited to answering basic queries but can handle complex technical problems, assist with HR-related issues, streamline workflows, and improve employee productivity at scale. The evolution of chatbots has been largely driven by progress in natural language processing, machine learning, speech technologies, and more recently multi-agent system frameworks, which together have made conversational systems indispensable in modern workplaces.

Despite these advancements, current chatbots still exhibit significant limitations. Traditional rule-based systems, common in the 1990s and early 2000s, were constrained by their reliance on static rules and patterns. They could efficiently answer questions like “How do I reset my password?” but failed when faced with slightly varied phrasing or unexpected problems. As natural language processing matured in the 2010s, machine learning introduced a greater degree of adaptability, enabling chatbots to recognize intent, classify queries, and respond in more context-aware ways. These capabilities allowed organizations to automate routine processes such as ticket creation, HR policy lookups, and even basic troubleshooting. However, while this era improved functionality, the responses still felt mechanical and lacked empathy, frustrating users during moments of stress. In recent years, large language models have further advanced chatbot design by supporting multi-turn conversations and generating contextually appropriate responses with remarkable fluency. They are integrated into enterprise systems, capable of independently resolving up to sixty percent of employee queries while reducing support costs by nearly a third. Yet, despite their technical sophistication, these systems remain limited in their ability to understand and respond to the emotional states of users, making interactions feel impersonal and detached.

The lack of emotional intelligence in chatbots is one of the most significant barriers to effective organizational support. When an employee reports a problem such as “my software crashed,” traditional systems provide solutions but fail to recognize whether the employee is calm, frustrated, or anxious. This lack of empathy often leaves users dissatisfied even when they receive a correct answer, undermining trust and confidence in the system. Our project directly addresses this gap by introducing a three-dimensional avatar capable of multimodal emotion detection. By analyzing facial expressions using Deepface, vocal tones through SpeechBrain, and textual sentiment via a model fine-tuned on IMDb data, the system achieves real-time emotion sensing with a high degree of accuracy. These signals are fused to guide avatar responses, ensuring that replies are not only technically correct but also emotionally appropriate. If a user reports frustration that a server crash is disrupting their day, the system acknowledges the stress with empathetic language, supported visually by the avatar’s nods or concerned expressions, making the interaction more human-like. Research shows that such empathetic cues reduce escalation to human agents, improve user

confidence, and increase task completion rates, demonstrating that emotional awareness is critical for effective support.

Beyond emotional intelligence, another persistent challenge in organizational support systems is the inefficiency of manual ticket creation. Conventional chatbots often require employees to manually specify issue categories, urgency levels, and descriptions, leading to errors and delays. Our solution automates this process through conversational form filling built using the Rasa framework. The chatbot collects user data naturally through dialogue, with an NLU model trained to recognize intents and extract entities accurately. For instance, if a user states “my keyboard isn’t working,” the system predicts the issue type as hardware and the category as replacement by applying a sentence transformer model that compares input with predefined metadata. The system keeps the form active until all necessary details are captured, handling interruptions and unclear inputs with predefined rules. Once collected, the information is verified with the user and processed into a structured support ticket. To further enhance efficiency, priority detection automatically determines urgency by combining keyword and semantic analysis, with business rules ensuring critical issues like outages are prioritized. This hybrid approach improves ticket accuracy, reduces errors, and speeds up submission, allowing employees to resolve issues quickly without tedious manual input.

The reliability of a support chatbot also depends heavily on its ability to retrieve accurate and contextually relevant knowledge. Traditional keyword-based retrieval often produces irrelevant results when queries are phrased differently from stored entries. To overcome this, our project integrates vector embeddings using ChromaDB for semantic search and graph databases like Neo4j to represent dependencies between technical entities. This combination ensures that queries about complex issues, such as VPN resets or firewall configurations, are matched with precise solutions. Knowledge ingestion involves cleaning and structuring IT and HR documents into JSON formats, tagging metadata for efficient search. Performance is enhanced further by caching frequent queries with Redis and maintaining episodic memory with MongoDB, which allows continuity in user interactions. These optimizations significantly reduce latency, while additional safeguards such as chain-of-thought prompting and guardrails minimize hallucinations. Evaluations confirm high levels of relevancy and faithfulness, making the system reliable for organizational use.

As the complexity of chatbot functions grows, coordinating multiple specialized components becomes increasingly difficult for single-agent architectures. Multi-agent systems provide a way to distribute tasks among different agents, but traditional frameworks often suffer from cascading failures and security risks due to vendor dependencies. Our project addresses this by integrating quantized local language models with modular Model Context Protocol servers. Running models locally protects sensitive enterprise data, while MCP servers allow agents to exchange information through structured interfaces without tight coupling. An orchestration layer manages task decomposition, dependency tracking, and error isolation, ensuring that failures in one agent do not bring down the entire system. This modular architecture improves scalability and fault tolerance, making it suitable for distributed environments where reliability and confidentiality are essential.

Together, these components form a unified solution that bridges the gaps left by traditional chatbot systems. The three-dimensional avatar provides empathetic and engaging interactions, conversational form filling streamlines ticket creation, knowledge base integration ensures accurate responses, and the multi-agent system coordinates tasks securely and efficiently. Each module was developed collaboratively by our team, with responsibilities distributed across members to ensure balance and coverage of all aspects. My

contribution focused on developing the avatar and emotion detection module, which forms the foundation of the system's ability to connect with users on a human level.

In essence, the background of this project reflects both the evolution of chatbots and the persistent challenges that remain. From rule-based beginnings to large language model–driven assistants, chatbots have continuously advanced in functionality but continue to struggle with emotional intelligence, efficient automation, and secure coordination. By integrating multimodal emotion sensing, automated form filling, robust knowledge retrieval, and multi-agent orchestration, this project seeks to deliver a system that is not only technically accurate but also empathetic, efficient, and trustworthy. In doing so, it responds to the unmet need for human-like, emotionally intelligent chatbots that can redefine IT and HR support in modern organizations.

## 1.2. Literature Review

This literature review explores four interconnected domains that form the foundation of our proposed system: [14] 3D avatars with voice interaction and emotion intelligence, [15]) conversational form filling automation for IT support ticket management, [16] knowledge base construction for accurate retrieval, and [17] multi-agent systems. Together, these areas represent critical advancements in conversational AI and automation, highlighting both prior work and research gaps that motivate our project.

### 1. 3D Avatar with Voice Interaction and Emotion Intelligence

The advancement of conversational AI has revolutionized IT support within organizations, enabling chatbots to handle complex technical queries, reduce operational costs, and enhance employee productivity [23] [14] [22]. Despite these strides, most chatbot systems lack the emotional intelligence and visual engagement needed for truly human-like interactions [21] [22], particularly in high-pressure IT environments. Integrating 3D avatars, sentiment analysis, emotion sensing, and robust voice input/output systems—specifically Text-to-Speech (TTS) and Speech-to-Text (STT)—remains underexplored, especially for empathetic and immersive communication.

Early IT support chatbots relied on rule-based systems [16] [21], which were limited by rigid scripts and struggled to address emotionally charged queries. The shift to machine learning in the 2010s improved context awareness [18] [22], but emotional interpretation remained weak. Even with the advent of large language models (LLMs), chatbots still fall short in empathetic response and visual engagement [17] [22] [33].

Significant progress has been made in STT and TTS systems. Modern STT tools like Whisper [16] [34] achieve over 95% accuracy across accents, enhancing accessibility, though noisy environments and emotionally charged speech reduce performance. TTS systems, such as ESPNet [20] [24] [35], produce natural-sounding speech, but emotional modulation remains underutilized. Similarly, sentiment analysis and speech emotion recognition (SER) demonstrate promise in detecting emotional states [18] [25] [26] [27] [28], yet their integration into IT support chatbots is limited.

The addition of 3D avatars enhances user engagement, with avatars capable of simulating facial expressions, gestures, and body language to build trust [26] [29] [30]. Tools such as Rhubarb Lip Sync [27] [38] further enable natural interaction. However, the integration of avatars with real-time sentiment and emotion sensing remains underexplored. Ethical considerations such as user privacy [36] and the computational cost of LLMs and 3D rendering [19] [29] also present challenges.

Overall, while significant advancements exist in STT, TTS, sentiment analysis, and emotion sensing, their integration with 3D avatars for empathetic IT support is still limited, revealing a key opportunity for future research.

## **2. Conversational Form Filling Automation for IT Support Ticket Automation**

Form filling is a critical process in online systems, but challenges such as language barriers and literacy levels make it difficult for some users. Conversational AI has been proposed as a solution, enabling users to provide structured information through dialogue. Prior work demonstrates the potential of chatbots to streamline form filling, though limitations remain in contextuality, domain specificity, and adaptability.

Meshram et al. [6] highlight how conversational AI can simplify data collection, while more recent research (Research A) [7] explores integrating LLMs with Retrieval-Augmented Generation (RAG) for context-sensitive form filling. This work demonstrates improved performance using BERT Score and G-EVAL but is limited to general-purpose forms without IT-specific workflows or support for free-form conversational input.

Other studies focus on accessibility. For example, Research B [6] introduced a Kannada voice-based chatbot for elderly and monolingual users. While innovative, it lacked flexibility, multilingual support, and scalability to technical fields like IT support. Similarly, “Bot Form Filler” (Research C) [8] used traditional NLP with Selenium for web-based forms but lacked conversational capabilities and domain specialization.

Context-aware approaches, such as “An Intelligent Framework for Auto-Filling Web Forms” (Research D) [9], employ clustering and contextual data to predict form entries but do not incorporate real-time dialogue or free-text input. Similarly, “Artificial Intelligence Agent for Contextual Guidance in Form Filling” (Research E) [10] uses NLP, entity recognition, and knowledge bases for contextual assistance but does not feature autofill or conversational flexibility.

These limitations underscore the need for a domain-specific conversational form-filling system for IT support. By combining LLMs, advanced NLP, and contextual mechanisms, future work can enable dynamic, real-time conversation, improve accessibility, and reduce user workload while maintaining accuracy in IT ticket automation.

## **3. Knowledge Base Construction for Accurate Retrieval**

The development of transformer-based LLMs has advanced natural language understanding and generation, enabling more reliable automated responses in IT support and related domains. However, accuracy and contextual relevance remain challenges, particularly in domain-specific applications. Retrieval-Augmented Generation (RAG) has emerged as a solution, integrating generative capabilities with real-time retrieval to improve precision and reduce hallucinations [1].

Han et al. (2024) demonstrate RAG’s utility in automating systematic literature reviews [6], while other studies highlight its effectiveness in IT support ticket resolution [8], ensuring responses are grounded in validated documentation. Integrations of RAG with knowledge graphs further enhance retrieval accuracy, reducing resolution time by nearly 29% [13].

Human-centered applications also benefit from these approaches. Jo and Seo (2024) introduce ProxyLLM, which moderates customer input tone to reduce emotional burden on agents. Similarly, Heng (2023)

explores advancements in vector search and GPT-4 integration for HR chatbots, emphasizing RAG's adaptability to enterprise contexts.

Despite these successes, challenges persist. Current systems often struggle with domain-specific queries, multimodal data, and complex query resolution [12] [4]. Additionally, reliance on static retrievers reduces adaptability in dynamic environments. Future directions include integrating adaptive retrievers, domain-specific fine-tuned LLMs, multimodal processing, and real-time learning, as well as improving interaction between retrieval and generation through advanced prompt engineering.

In summary, RAG combined with LLMs has shown significant potential across multiple domains, but domain-specific IT support applications still face gaps in precision, flexibility, and scalability.

#### 4. Multi-Agent System

AI research has increasingly shifted toward agentic and embodied approaches, with transformer architectures [1] enabling the rise of generative AI and large language models. The parallel development of open-source projects such as GPT-OSS [2], Mistral [3], and LLaMA [4] has democratized experimentation, while platforms like Hugging Face provide tools for transparency and collaboration. Local inference tools like Ollama [5] and advances in quantization [6] have made it possible to run models efficiently on consumer hardware, increasing accessibility and reducing dependency on cloud providers.

Alongside these developments, experimental AI agents have emerged that extend beyond text generation. Google's Jules links directly with GitHub repositories to identify and fix bugs [7], while Microsoft's Copilot provides real-time coding assistance [8]. Other tools, such as Bolt and Cursor [9] [10], explore autonomous loops for software generation and refinement. While promising, these agents remain limited in autonomy and reliability, highlighting both opportunities and challenges in the transition to agentic AI [11].

Frameworks for MAS coordination offer varied approaches. Microsoft's AutoGen [12] organizes agents via a conversation-centric message bus, ensuring transparency and modularity. CrewAI [13] emphasizes role specialization, distributing tasks predictably across agents. LangGraph [14], on the other hand, uses a graph-based workflow to manage dependencies and decision paths. Each framework demonstrates distinct strengths in coordination, control, and adaptability.

Despite these advancements, existing MAS frameworks are still experimental, often tied to predefined architectures that may not suit all domains. There remains a need for flexible, domain-specific agent orchestration that balances autonomy, security, and scalability for enterprise environments. This motivates the design of new MAS approaches tailored specifically for IT support and organizational workflows.

#### Summary

Across these four domains, the literature reveals both rapid advancements and significant gaps. While progress in 3D avatars, conversational form filling, knowledge retrieval, and multi-agent systems has transformed the landscape of conversational AI, existing solutions often remain fragmented, lack domain specificity, or struggle with emotional intelligence, scalability, and adaptability. Addressing these challenges through integration and domain-focused design offers an opportunity to create an inclusive, efficient, and secure IT support system that moves beyond functional automation toward truly human-like, context-aware assistance.

### 1.3. Research Gap

Despite considerable progress in the field of conversational AI, the review of existing literature and implemented systems reveals several persistent and interconnected research gaps that limit the overall effectiveness of IT support automation. These gaps are spread across three core areas—form automation, knowledge base management, and emotionally intelligent 3D avatar chatbots—yet collectively highlight the necessity for a more adaptive, context-sensitive, and human-like approach to IT support systems.

One of the most pressing limitations is observed in **form automation**. Current systems designed for automated form-filling are typically general-purpose or oriented towards government and administrative workflows. While these solutions can capture structured information effectively, they lack the flexibility required to handle IT-specific support contexts where queries are often unstructured, urgent, and highly variable. Most existing models depend on rigid workflows and scripted dialogues, leading to user fatigue and a high likelihood of task abandonment. This rigidity stands in sharp contrast to the expectations of modern IT employees who prefer fast, adaptive, and minimal-question interactions. Moreover, while some autofill mechanisms exist, they are usually disconnected from conversational guidance and real-time correction, thereby failing to optimize user experience. Crucially, semantic ticket classification—a fundamental process for routing IT support cases to appropriate categories—remains underexplored in most reviewed works. Without the ability to infer semantic meaning, adapt dynamically to contextual variations, or confirm user intent through large language models (LLMs), current systems miss an opportunity to reduce manual workload and improve resolution efficiency.

This gap underscores the necessity of moving beyond simple form-filling to what can be described as **adaptive slot filling**. Such a system would leverage semantic inference, contextual prediction, and LLM-assisted reasoning to reduce redundancy in questioning while maintaining high completion accuracy. Adaptive slot filling allows the system to predict and pre-populate form fields based on the semantic similarity of previous cases, contextual knowledge, and predictive completion. This strategy directly addresses user fatigue by minimizing repetitive questions while still producing comprehensive IT support tickets. The outcome would not only increase efficiency but also elevate user satisfaction, since employees would spend less time repeating information and more time resolving their issues. In addition, integrating priority detection through keyword and semantic analysis—supplemented by user confirmation—would ensure accuracy in ticket categorization, resulting in a more streamlined workflow. Yet, despite the clear advantages, such adaptive strategies remain largely absent from existing literature and practical implementations.

In parallel, a significant body of research gaps exists in **knowledge base management**, particularly in the domain of Retrieval-Augmented Generation (RAG) systems. While RAG approaches have emerged as an effective way to enhance LLM outputs with contextual data, most existing solutions rely on static document uploads and manual interventions for content updates. This rigidity creates inefficiencies, as IT support environments often require dynamic

updates to reflect changing organizational contexts, security protocols, and technical documentation. The absence of automated mechanisms for updating vector databases with new contextual data or replacing outdated content results in knowledge bases that quickly become stale and less useful for real-time support needs.

Moreover, LLM-driven chatbots frequently suffer from **hallucination issues**, where the system produces irrelevant or fabricated responses that fail to meet user expectations. In IT support, where precision is critical, such hallucinations undermine trust and satisfaction. Compounding this challenge is the inefficiency of token usage in many existing LLM systems. Current implementations often supply the entire conversational history as input for every interaction, consuming tokens unnecessarily and driving up costs. Without effective memory management, redundancy increases, while contextual sensitivity decreases, resulting in both higher expenses and poorer user experiences.

Another identified gap in knowledge base systems is the **inefficient conversion of unstructured documents**—such as Word files or PDFs—into structured formats for indexing. Existing methods struggle with capturing the contextual meaning of documents, leaving knowledge bases fragmented and responses delayed. These inefficiencies are compounded by inadequate query optimization techniques, which limit retrieval performance in high-demand scenarios. Additionally, many chatbot systems rely heavily on direct OpenAI API integration and basic prompt engineering, leading to simplistic and generic responses. This highlights a broader research gap in developing **sophisticated memory management strategies** capable of dynamically optimizing context selection, reducing redundancy, and enhancing accuracy without escalating costs.

The lack of user-centered evaluation mechanisms also limits the efficacy of knowledge-based AI systems. Few implementations actively measure user satisfaction, track successful versus unsuccessful interactions, or customize responses based on feedback loops. The inability to identify and manage repeated queries results in redundancy and longer resolution times. Closing these gaps requires the development of adaptive knowledge bases that are self-updating, context-aware, and capable of measuring user engagement to refine response strategies dynamically.

Beyond functional gaps in form automation and knowledge bases, perhaps the most profound limitations are observed in the area of **emotionally intelligent and visually immersive chatbot systems**. While platforms such as Microsoft Azure and Google Dialogflow provide strong natural language processing capabilities, they lack the capacity to recognize or adapt to employee emotions during stressful IT support scenarios. Employees who encounter system crashes, network outages, or urgent technical failures often express frustration, stress, or urgency in their interactions. Yet current chatbots respond mechanically, with generic instructions, rather than empathetic acknowledgments of the user's emotional state. The absence of integrated emotional intelligence diminishes user trust and creates a disconnect between the employee's needs and the system's response.

Equally limiting is the lack of **visual interaction mechanisms** in existing IT support chatbots. Text- or voice-only systems, such as those integrated with ServiceNow or Zendesk, may be efficient for handling routine queries like password resets but fail to create an immersive or

reassuring experience during high-pressure incidents. Research has shown that visual engagement—through 3D avatars with facial expressions and gestures—can improve user confidence and task completion rates. Yet such avatars are seldom incorporated into IT support solutions, and when they are, they function as static visuals rather than dynamic responders. The potential of avatars capable of mirroring concern, nodding attentively, or adjusting expressions in response to detected stress remains largely unexplored.

The integration of **Speech-to-Text (STT) and Text-to-Speech (TTS)** technologies has improved accessibility, enabling hands-free interactions and supporting employees with diverse needs. However, these tools are rarely combined with emotion-adaptive mechanisms, meaning that vocal cues such as urgency or frustration are ignored. Similarly, sentiment analysis models, while capable of classifying text-based emotions, often fail to incorporate vocal signals like pitch or pace. Emotion recognition tools such as Speech Emotion Recognition (SER) exist but face challenges in real-time processing, particularly in noisy office environments. These gaps highlight the underutilization of multimodal emotion-sensing approaches that combine text, voice, and visual cues into a unified support framework.

The synchronization of **3D avatars with real-time emotion detection, sentiment analysis, and speech processing** represents an even deeper research gap. While tools like Ready Player Me or Rhubarb Lip Sync offer partial capabilities, they do not integrate emotion sensing with IT-specific workflows. The result is a disjointed user experience where avatars may speak or gesture but fail to align with the emotional state of the employee. For IT support contexts, where employees seek reassurance as well as technical resolution, the absence of lifelike, synchronized avatars represents a significant missed opportunity.

Architectural and ethical limitations further hinder the adoption of advanced conversational systems. Many existing chatbots are monolithic, making it costly and technically difficult to integrate advanced features like emotion detection, 3D rendering, or adaptive memory management. This raises concerns about scalability and computational costs, particularly for organizations with limited budgets. Privacy issues also loom large, especially when analyzing voice or emotional data without explicit consent. Employees may resist engaging with bots that seem intrusive or untrustworthy, leading to poor adoption rates.

Taken together, these research gaps emphasize the urgent need for a **unified, domain-specific conversational AI system** that integrates adaptive form automation, dynamic knowledge base management, and emotionally intelligent 3D avatar interfaces. Such a system would not only automate IT support operations but also humanize them—recognizing emotional states, reducing repetitive questioning, dynamically managing knowledge retrieval, and presenting information through engaging, empathetic avatars. By combining these innovations, future IT support chatbots could deliver a truly adaptive, efficient, and supportive experience that meets both the technical and emotional needs of employees.

In summary, the existing body of work in conversational AI has laid a strong foundation, yet critical gaps persist across automation, knowledge management, and emotional intelligence. Current systems excel at parsing structured inputs and automating basic workflows but fail to adapt dynamically, manage context effectively, or provide emotionally resonant interactions.

Bridging these gaps is not merely a matter of technical optimization; it is about redefining IT support to be faster, smarter, and more human-like. Addressing them will require cross-disciplinary innovation—bringing together advances in natural language processing, dynamic memory architectures, multimodal emotion sensing, and immersive visual design. Only through such integration can IT support systems evolve into empathetic, efficient, and scalable solutions that meet the growing demands of modern organizational environments.

## 1.4. Research Problem

The development of advanced AI-driven systems for IT support requires addressing several interrelated challenges across multiple domains. This research project explores four critical components: (1) Emotionally Intelligent 3D Avatars with Voice Interaction, (2) Conversational Form Filling Automation for IT Support Ticketing, (3) Knowledge Base Construction for Accurate Retrieval, and (4) Multi-Agent Systems for Autonomous Workflows. Each component represents a distinct research problem, yet together they form an integrated approach to enhancing enterprise IT support.

### 1. Emotionally Intelligent 3D Avatar with Voice Interaction

The creation of an emotionally intelligent 3D avatar chatbot introduces significant challenges in combining real-time emotion detection, avatar animation, and contextual response generation. While conversational agents have improved through advancements in AI and Natural Language Processing (NLP), there remains a substantial gap in achieving interactions that feel genuinely empathetic and human-like.

A major research problem lies in real-time multimodal emotion detection, where visual, vocal, and textual cues must be simultaneously processed to interpret complex emotional signals without introducing latency. Similarly, generating natural, synchronized avatar responses demands advanced animation techniques, voice modulation, and contextual alignment to avoid robotic or artificial interactions.

Additionally, the system must integrate emotional intelligence with knowledge-based responses, ensuring technical accuracy while addressing users' emotional states in stressful IT support scenarios. Other critical challenges include robustness across diverse demographics and environments, such as handling accents, lighting variations, and cultural differences in emotional expression, as well as privacy and ethical concerns over emotion-sensitive data.

Finally, scalability and real-time performance are essential. The research problem therefore spans multimodal fusion, avatar naturalism, context-sensitive emotional reasoning, adaptability, ethical safeguards, and seamless enterprise integration—all necessary to transform IT support into a more empathetic and human-centered service.

### 2. Conversational Form Filling Automation for IT Support Ticketing

At LOLC, service request forms often lead to confusion, delays, and errors due to user uncertainty, lack of knowledge, or the complexity of required information. Manual creation of forms further contributes to inefficiency. Studies show that form complexity increases frustration, with typical mistakes including incomplete details and failure to follow instructions [11], [12]. Moreover, form filling is widely regarded as time-consuming and tedious, resulting in incomplete or incorrect submissions [4], [3].

The magnitude of the problem is underscored by findings that 81% of users discontinue filling out online forms, while 67% of those who encounter difficulties abandon the process entirely [13]. In the business world, where 59% of workers regularly complete forms [14], these inefficiencies contribute to lost productivity and poor data quality.

The research problem is to move beyond static form-filling methods by developing a conversational AI-based chatbot integrated into the IT support ticketing system. Such a system would guide users in identifying the required information through interactive conversation, leveraging a knowledge base where necessary. By doing so, the solution aims to reduce form abandonment, improve data accuracy, and decrease completion time.

Ultimately, the goal is to optimize the form-filling process, reduce errors, and enhance user satisfaction by transforming a static task into a dynamic, conversational interaction—improving both efficiency and reliability in IT support ticketing.

### **3. Knowledge Base Construction for Accurate Retrieval**

Enterprises adopting Retrieval-Augmented Generation (RAG) and large language models (LLMs) face major challenges in efficiency, scalability, and trustworthiness. Current systems rely on static document uploads that require manual updating of vector databases, limiting adaptability. Furthermore, LLMs are prone to hallucinations, generating inaccurate responses that erode user trust, particularly in sensitive environments such as finance and internal IT support. Inefficiencies in token utilization and high operational costs worsen these limitations.

Existing frameworks also lack dynamic context optimization and personalization. They fail to efficiently handle repeat queries, integrate user feedback, or build long-term memory. In practice, employees often encounter repetitive issues, leading to redundant ticket creation and wasted support resources. A dynamic knowledge base capable of automatically retrieving and delivering contextually relevant information could resolve these inefficiencies.

From a technical perspective, processing enterprise documents presents another research gap. Current techniques do not provide robust pipelines for analyzing and structuring documents (e.g., Word, PDF) into vector databases with optimized retrieval indexing. This constrains the system's ability to consistently deliver accurate, contextually relevant responses.

The research problem also encompasses episodic memory and multimodal context integration, enabling the system to personalize interactions and synthesize responses across multiple domains (e.g., IT and HR). Furthermore, the knowledge base should augment human support agents by providing actionable insights alongside automated responses.

In summary, the research focuses on building a novel knowledge base framework that supports dynamic updating, minimizes hallucinations, maximizes token efficiency, integrates multimodal context, and delivers personalized, reliable, and context-aware responses—addressing critical shortcomings of current RAG-based enterprise systems.

### **4. Multi-Agent System for Autonomous IT Workflows**

The growing interest in multi-agent systems (MAS) presents both opportunities and challenges. While multiple AI agents can collaborate to solve complex tasks, the effective coordination between agents with different capabilities remains a significant problem. Each agent may have specialized strengths, such as task decomposition, knowledge retrieval, or conversational reasoning, but ensuring that these agents communicate and operate seamlessly requires careful orchestration. Without proper coordination, agents risk duplication of work, inefficient execution, or even conflicting outputs, reducing the reliability of the overall system.

A related challenge lies in ensuring reliable execution within distributed environments. As multi-agent systems scale, they need to handle failures gracefully, manage dependencies between tasks, and ensure continuity in execution. For instance, if one agent fails in the middle of a workflow, the orchestrator must detect the issue, reassign or retry the task, and ensure that downstream agents are not affected. Designing such fault-tolerant mechanisms while maintaining efficiency introduces complexity that must be addressed.

Another research problem is how to design an orchestrator capable of decomposing high-level goals into executable subtasks. While large language models are strong in natural language reasoning, translating abstract user goals into concrete, system-executable steps requires structured planning. The orchestrator must not only divide tasks logically but also assign them to the right agents while keeping track of progress in a dynamic task ledger. Achieving this balance of autonomy and control is central to building a scalable multi-agent system.

Finally, several practical and architectural concerns further complicate the problem. Questions such as how to plan tasks effectively, how to design the overall architecture, what programming language and frameworks to use, and how to integrate with external APIs remain critical. In addition, ensuring security against threats like prompt injection, managing infrastructure handoff, and preserving reliable communication between services are unresolved challenges. Together, these issues highlight the gap between the potential of multi-agent systems and the practical difficulties of building them into real-world, production-ready solutions.

## **Overall Research Challenge**

While each component presents unique challenges, they are interconnected in building an integrated AI-driven IT support ecosystem. Emotionally intelligent avatars humanize interactions, conversational form-filling streamlines support requests, dynamic knowledge bases ensure accurate retrieval, and multi-agent systems enable orchestration of complex workflows. Together, these research problems highlight the gaps in current AI systems and define the pathway toward a scalable, empathetic, accurate, and autonomous IT support framework.

## 1.5. Research Objectives

### 1.5.1. Main Objective

The primary objective of this project is to design, develop, and deploy a sophisticated conversational AI system that automates IT support workflows while enhancing the quality, efficiency, and human-like nature of user interactions. The system is envisioned as a multi-component framework that seamlessly integrates the following capabilities:

#### 1. Retrieval-Augmented Generation (RAG) for IT and HR Support:

The system will incorporate a RAG framework that utilizes internal company data to provide contextually relevant, accurate, and stepwise solutions for IT and HR-related inquiries. By leveraging historical interactions, episodic memory, and a cost-efficient knowledge base, the RAG system aims to deliver personalized responses that maximize information accuracy while minimizing computational resource usage. It will dynamically search across multiple documents and structured sources, synthesizing composite answers that address user queries precisely.

#### 2. Generative AI Chatbot for Form Automation:

The project seeks to automate the completion of complex IT request forms using an intelligent chatbot capable of understanding natural language inputs. Traditional forms are often static, confusing, and time-consuming, resulting in incomplete submissions and added strain on IT support staff. The chatbot will leverage semantic classification, contextual inference, and adaptive slot-filling mechanisms to ensure that the maximum number of form fields are completed efficiently with minimal user queries. By using hybrid priority detection and contextual reasoning, the system will prioritize urgent issues while maintaining high accuracy and reducing user fatigue.

#### 3. 3D Avatar-Based Emotionally Intelligent Interaction:

Beyond technical accuracy, the system will integrate a 3D avatar interface that provides multimodal, empathetic interactions. Employees experiencing IT problems often face stress and urgency, and a purely transactional chatbot may fail to address these emotional dimensions. The 3D avatar, coupled with sentiment analysis, emotion sensing via webcam and voice, and dynamic speech generation, will offer a human-like support experience that is both technically precise and emotionally supportive. This avatar-driven interface will use lifelike animations, natural gestures, and expressive responses to guide users through problem-solving while reducing frustration and increasing engagement.

## **4. Multi-Agentic System For Complex Multi Step Responses**

The primary objective of this research is to design and implement a flexible orchestration system for multi-agent AI, capable of coordinating multiple autonomous agents to achieve complex, high-level goals. The system aims to reliably decompose abstract objectives into executable tasks, assign them to the most suitable agents, monitor progress through a task ledger, and ensure smooth inter-agent communication and execution. By integrating modular microservices, secure APIs, and connections to external knowledge sources and language models, the research seeks to create a scalable framework that balances efficiency, accuracy, and flexibility while providing a foundation for future enhancements in multi-agent collaboration and intelligent task management.

Through these integrated components, the project's main goal is to improve operational efficiency, enhance user satisfaction, and demonstrate the potential of conversational AI in transforming enterprise IT support environments. The research seeks not only to validate the technical viability of such a system but also to explore its broader implications for human-centered AI interactions within organizational contexts.

### **1.1.1. Sub Objectives**

To achieve the main objective, the project has defined a comprehensive set of sub-objectives. These sub-objectives break down the complex system into actionable, measurable tasks across three key areas: knowledge management, conversational form automation, and emotion-aware interactions.

#### **1. Knowledge Base and RAG System Development**

The foundation of the AI system lies in a robust knowledge base that stores internal organizational data, IT policies, and historical support interactions. The following sub-objectives outline the design, development, and optimization of this component:

##### **Microservice Architecture for Knowledge Base:**

Develop the knowledge base using a modular microservice architecture that allows independent operation, seamless API access, and scalable deployment. This architecture ensures high availability, maintainability, and flexibility to integrate with external systems.

##### **Dynamic Contextual Responses:**

Implement mechanisms to provide accurate, context-sensitive responses to employee queries. The system will leverage advanced document classification, NLP-based content parsing, and semantic embeddings to deliver information in structured formats such as JSON.

### **Document Integration and Management:**

Facilitate the addition of new documents or text content through a user interface, automatically embedding and storing them in the vector database. The system will detect similar contexts and update the knowledge base to maintain consistency, accuracy, and relevance.

### **Episodic Memory for Contextual Relevance:**

Incorporate episodic memory to reference prior conversations and interactions. This enables the system to provide answers informed by historical context, improving accuracy and efficiency while minimizing token wastage and computational overhead.

### **Automated File-Type Identification and Conversion:**

Implement an automated file recognition and conversion pipeline to handle diverse document types. This ensures seamless extraction, structuring, and embedding of new content into the knowledge base for real-time querying.

### **RAG Query Mechanism:**

Develop a query mechanism that retrieves relevant information from multiple sources to generate accurate, multi-document answers. This mechanism ensures that employees receive guidance that is both specific to their situation and technically reliable.

## **2. Generative AI-Based Chatbot for IT Form Automation**

The chatbot is designed to replace traditional static forms with a dynamic, conversational interface that guides users through IT support requests. The sub-objectives for this component include:

### **Conversational Form Design:**

Implement a responsive, user-friendly chat interface using React that enables guided interactions. The chat interface will display messages, ask follow-up questions dynamically, and support user validation of suggestions.

### **Adaptive Slot Filling and Contextual Reasoning:**

Use Rasa's custom form logic to adaptively collect information based on conversation context. The system will dynamically prioritize form fields, ask relevant follow-up questions, and infer data wherever possible to minimize redundant user input.

### **Hybrid Confirmation System:**

Introduce a confirmation mechanism that balances automation with user control. Users can approve or modify auto-generated suggestions for form fields, ensuring accuracy and minimizing errors.

### **Interruption Handling and Out-of-Scope Management:**

Employ robust interruption handling to manage unrelated queries or ambiguous messages. This includes

fallback classifiers, predefined responses, and clarification workflows that maintain conversation flow while guiding users back to the form completion process.

#### **Intent Recognition and Keyword Extraction:**

Implement a sophisticated NLU pipeline using DIET Classifier, RegexFeaturizer, LexicalSyntacticFeaturizer, and CountVectorsFeaturizer to extract user intents and relevant technical details. This ensures accurate mapping of user inputs to form fields.

#### **Form Validation and Final Submission:**

Validate each form field using custom FormValidationAction to ensure data consistency, completeness, and logical accuracy. The system will provide a preview of the completed form for final user confirmation and submit structured JSON data to external IT support systems via API integration.

### **3. 3D Avatar Design and Emotion-Sensing Integration**

A core innovation of the project is the integration of a 3D avatar capable of interpreting and responding to user emotions, creating a human-like support experience. The sub-objectives in this area include:

#### **Text-to-Speech (TTS) and Speech-to-Text (STT) Integration:**

Integrate Whisper for STT and ESPNet for TTS to enable real-time voice interaction. The system will ensure accurate transcription, natural-sounding voice responses, and adaptive learning to improve performance over time.

#### **3D Avatar Development:**

Design and implement a lifelike 3D avatar using tools such as Unity and Blender. The avatar will display facial expressions, gestures, and lip-synced movements aligned with TTS outputs, enhancing the visual and emotional quality of interactions.

#### **Emotion Sensing and Multimodal Sentiment Analysis:**

Combine facial expression analysis (DeepFace), voice tone evaluation (SpeechBrain), and text sentiment analysis (IMDb-trained models) to determine user emotional states. The system will adapt responses—including TTS speed, tone, and avatar gestures—to convey empathy, reassurance, or urgency as needed.

#### **Dynamic and Personalized Interaction:**

Adjust conversation flow and responses based on emotional context, ensuring that employees feel heard and supported. The system will be able to recognize stress, frustration, and urgency, modifying responses to provide both technical guidance and emotional reassurance.

#### **Inclusive and Accessible Design:**

Ensure accessibility for all employees, including visually impaired staff or those with literacy challenges, by supporting voice-driven interaction, accent-robust STT, and customizable voice profiles. Privacy measures, opt-in permissions, and live-only processing guarantee secure handling of sensitive data.

## **4. Multi Agentic System**

### **Task Decomposition and Assignment**

Develop a mechanism within the orchestrator to translate high-level goals into discrete, executable subtasks and assign them to the most suitable agents. This ensures that each agent performs tasks aligned with its capabilities, improving efficiency and reducing conflicts. The task ledger tracks assignments and progress, providing visibility into ongoing workflows and enabling dynamic adjustments when dependencies or failures arise.

### **Inter-Agent Communication and Collaboration**

Enable reliable communication among agents and between agents and external services. This includes API integration with knowledge bases, external tools, and other microservices. Effective collaboration ensures coherent multi-step workflows, allowing agents to share data, request assistance, and coordinate actions without causing conflicts or redundancy.

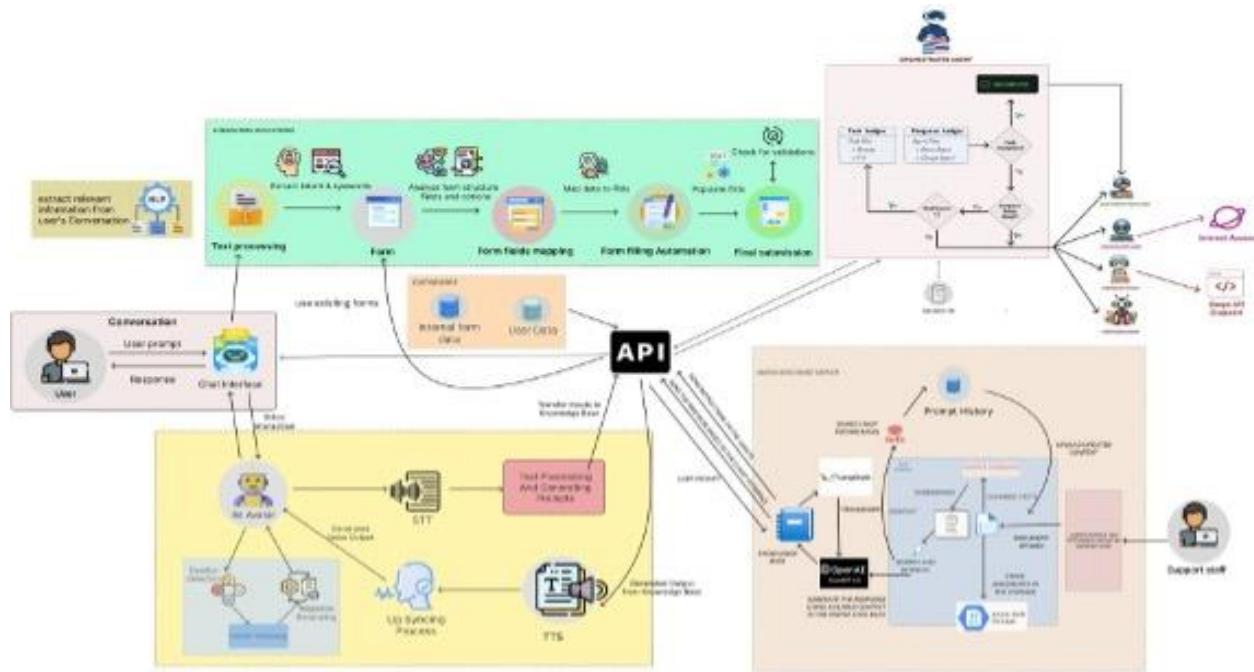
### **Scalability and Extensibility**

Design the system architecture to support future growth and integration of additional agents or services. Modular microservices and flexible orchestration enable easy expansion without major redesigns. This ensures that the multi-agent system can accommodate evolving requirements, new workflows, and more complex tasks over time, while maintaining performance and reliability.

## 2. Methodology

### 2.1. Methodology

#### Overview of the overall system

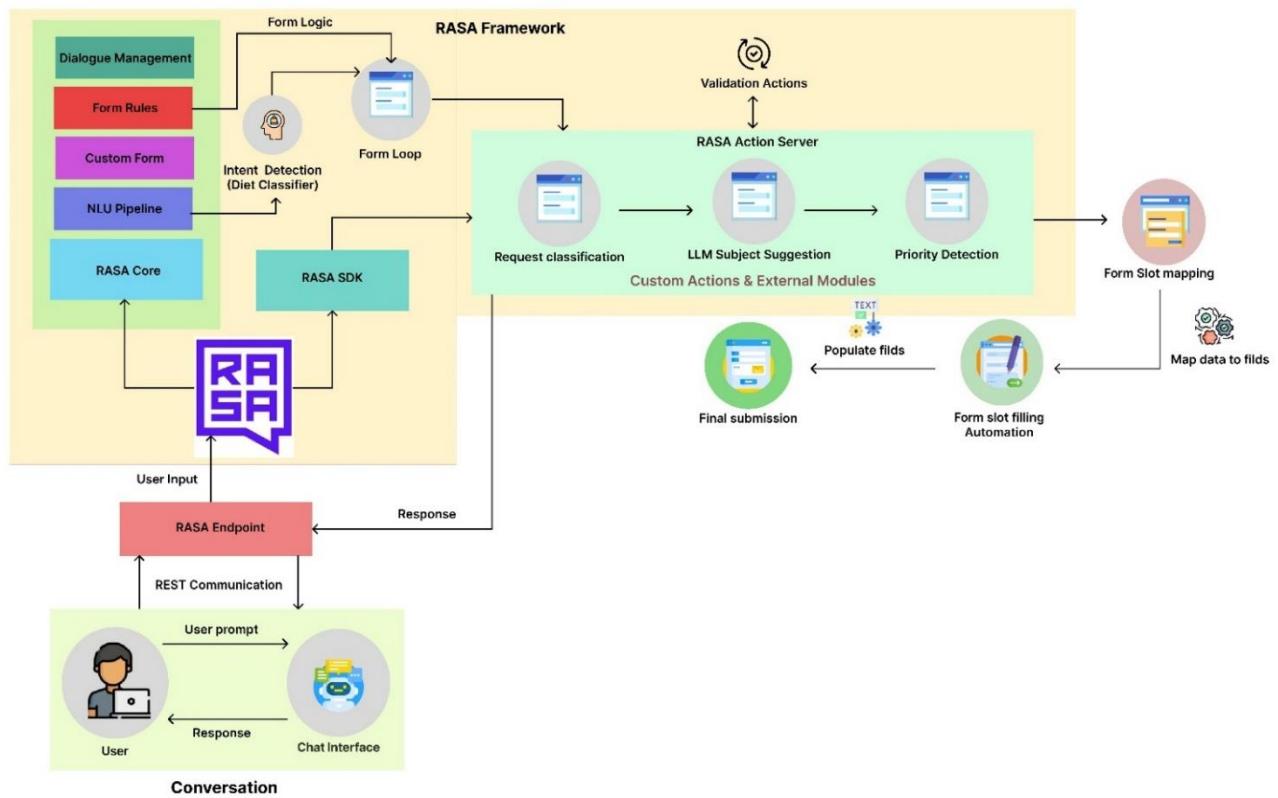


For the development of this system, we are primarily adopting a **microservice architecture**, which allows us to design the solution as a collection of independent, modular services that can seamlessly communicate with each other through API endpoints. This approach ensures scalability, flexibility, and ease of maintenance, as each component can be developed, deployed, and upgraded independently without affecting the entire system. Our system is mainly composed of four core components: the **Knowledgebase**, the **3D Avatar**, the **Form Automation module**, and the **Multi-Agentic system**.

To effectively address the identified research gaps, we are introducing several innovative features across these components. In the **Multi-Agentic system**, we plan to integrate an **MCP (Multi-Context Processing) server** equipped with a wide range of specialized tools to enhance automation and streamline complex tasks. For the **3D Avatar**, we aim to enrich the user experience by implementing **emotional face recognition**, **text-to-speech (TTS)**, **speech-to-text (STT)**, and **sentiment analysis** technologies. These features will enable the avatar to interact more naturally with users, making communication more engaging and human-like.

Within the **Knowledgebase**, we are designing a **dynamic pipeline** that can automatically update the database, ensuring that the system remains current and reliable with the most recent information. Additionally, we are incorporating **episodic memory** to improve the system's contextual understanding and **caching mechanisms** to enable faster and more efficient answer retrieval. Finally, the **Form Automation** component will help reduce manual effort by intelligently handling repetitive and structured data entry tasks.

By using the **latest and most efficient tools, frameworks, and techniques**, our goal is to build a solution that not only meets the immediate needs of users but also pushes the boundaries of innovation in automation, knowledge management, and interactive technologies. This holistic approach ensures that the product will be **robust, user-friendly, and future-ready**, setting a strong foundation for real-world adoption.



## Individual components

Conversational form filling automation

Figure 2: Conversational form filling system diagram

This diagram illustrates a comprehensive conversational AI system built on the RASA framework that automates IT support ticket creation through intelligent dialogue management. The system begins when users input support requests (like "my mouse is not working") through the chat interface, which are processed by the RASA Core and NLU Pipeline using a Diet Classifier for intent detection. The framework employs sophisticated dialogue management with custom forms and rules that activate form loops when support requests are detected, maintaining conversation state until ticket completion while handling interruptions and fallback scenarios for unclear inputs.

The automated processing pipeline showcases advanced AI capabilities where the system classifies requests using sentence transformers (ALLMINILM V6) with metadata matching, automatically predicting request types and categories with high accuracy. The RASA Action Server orchestrates multiple AI components including LLM-powered subject suggestion using Ollama/Phi3 models, priority detection through hybrid keyword matching (60%) and semantic analysis (40%), and intelligent form slot filling that populates structured ticket fields. The entire process culminates in validation actions and final submission, outputting structured JSON data ready for API integration with external ticketing systems, effectively transforming complex multi-step processes into natural conversational experiences while maintaining user control and data accuracy.

## **Research Design and Approach**

This study employs a design science research methodology focusing on creating a conversational AI system for automated form processing. The research follows an iterative development approach combining quantitative evaluation of system accuracy with qualitative assessment of user experience factors.

The research addresses inefficient customer support ticket creation processes where traditional web forms frustrate users and result in incomplete or poorly categorized support requests. The scope focuses on IT support scenarios including hardware malfunctions, software issues, network problems, and access difficulties.

## **Framework Selection and Technical Architecture**

### **Conversational AI Framework Evaluation**

After comprehensive evaluation of multiple conversational AI frameworks, Rasa Open Source was selected as the optimal choice. The evaluation criteria included:

#### **Cloud-Based Solutions:**

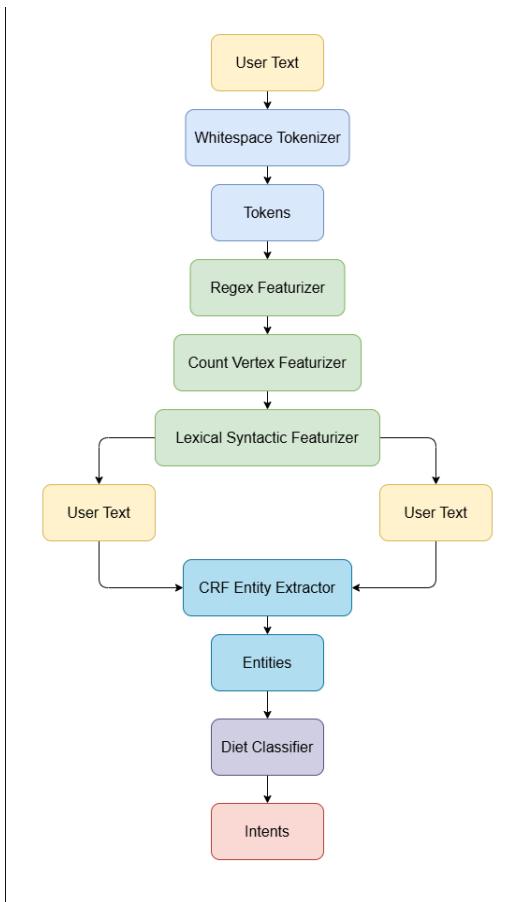
- Google Dialogflow: Easy to use but limited customization and external data transmission
- Microsoft Bot Framework: Enterprise features but complex licensing and vendor lock-in
- Amazon Lex: Good AWS integration but insufficient fine-grained control

#### **Open Source Alternatives:**

- Botpress: Visual interface but limited NLP capabilities
- Rasa Open Source: Complete control and customization with required technical sophistication

Rasa's selection was driven by its open-source transparency, scalability for production deployment, and full customization capabilities essential for specialized form automation.

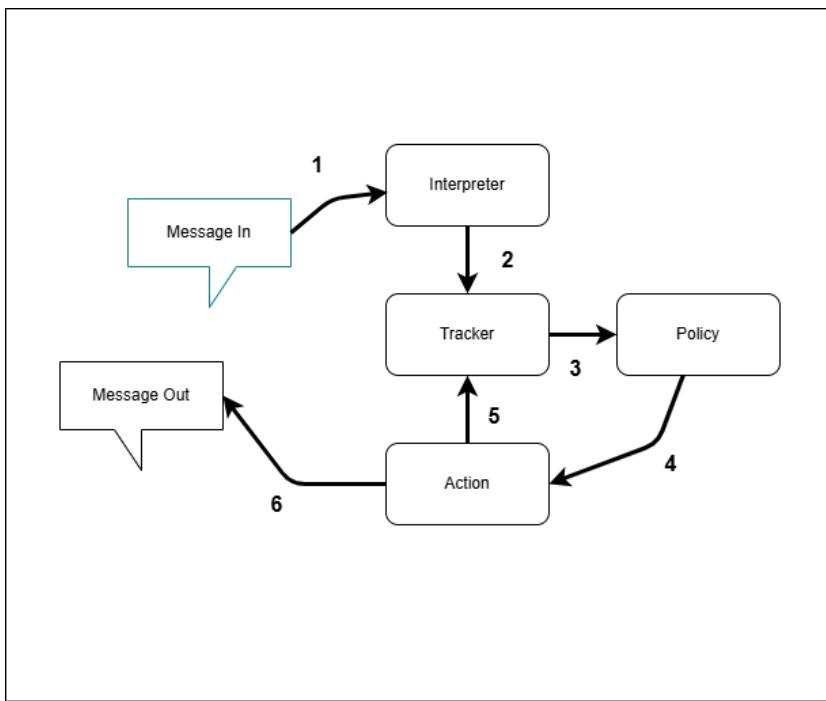
## RASA Technical Architecture



[Figure 1: Working of the RASA NLU pipeline]

NLU Pipeline Components:

1. **WhitespaceTokenizer**: Splits user input into tokens for subsequent processing
2. **RegexFeaturizer**: Extracts structured information like error codes and IP addresses
3. **LexicalSyntacticFeaturizer**: Analyzes grammatical structure and linguistic patterns
4. **CountVectorsFeaturizer**: Creates numerical text representations using word-level and character n-gram analysis
5. **DIETClassifier**: Core transformer-based model for intent classification and entity extraction (150 epochs)
6. **EntitySynonymMapper**: Normalizes extracted entities to canonical forms
7. **ResponseSelector**: Handles response selection for retrieval-based interactions
8. **FallbackClassifier**: Manages unclear inputs with 0.63 confidence threshold



[Figure 2: Working of RASA core]

**Core Dialogue Management:** The system employs multiple complementary policies:

- **MemoizationPolicy**: Handles exact conversation pattern matching
- **RulePolicy**: Manages deterministic workflows for form automation
- **UnexpectTEDIntentPolicy**: Handles unexpected intents and interruptions
- **TEDPolicy**: Provides sophisticated context-aware dialogue management

## Semantic Analysis Integration

**Sentence Transformers Implementation:** The all-MiniLM-L6-v2 model was selected for automated issue classification, providing 384-dimensional embeddings balancing accuracy with computational efficiency. The system pre-computes embeddings for predefined categories enabling rapid runtime similarity calculations using cosine similarity metrics.

**Ollama Integration:** Ollama serves as the local large language model inference engine using the phi3:mini model for generating contextual subject lines. Temperature is set to 0.2 for consistent output, with tuned top\_p and top\_k parameters optimized for IT support contexts.

## Conversational Workflow Design

### Intent Recognition and Entity Extraction

The system uses hierarchical intent structure with support\_request as the primary trigger. The Customer Support Tickets dataset from HuggingFace provided training data, with English-language tickets retained for linguistic consistency.

```
NLU model loaded. Type a message and press enter to parse it.
Next message:
my mouse is not working
C:\Users\kishwa.kuruppu\AppData\Local\Programs\Python\Python310\lib\site-packages\rasa\shared\utils\io.py:100: UserWarning: Parsed an entity 'device' which is not
defined in the domain. Please make sure all entities are listed in the domain.
  More info at https://rasa.com/docs/rasa/domain
C:\Users\kishwa.kuruppu\AppData\Local\Programs\Python\Python310\lib\site-packages\rasa\shared\utils\io.py:100: UserWarning: Parsed an entity 'issue' which is not
defined in the domain. Please make sure all entities are listed in the domain.
  More info at https://rasa.com/docs/rasa/domain
{
  "text": "my mouse is not working",
  "intent": {
    "name": "support_request",
    "confidence": 1.0
  },
  "entities": [
    {
      "entity": "device",
      "start": 3,
      "end": 8,
      "confidence_entity": 0.9996558427810669,
      "value": "mouse",
      "extractor": "DIEClassifier",
      "processors": [
        "EntitySynonymMapper"
      ]
    },
    {
      "entity": "issue",
      "start": 12,
      "end": 23,
      "confidence_entity": 0.9959031939506531,
      "value": "not working",
      "extractor": "DIEClassifier",
      "processors": [
        "EntitySynonymMapper"
      ]
    }
  ],
}
```

[Figure 3: NLU Performance ]

```
"intent_ranking": [
  {
    "name": "support_request",
    "confidence": 1.0
  },
  {
    "name": "mood_great",
    "confidence": 2.1084960868918764e-16
  },
  {
    "name": "goodbye",
    "confidence": 1.391107884961951e-16
  },
  {
    "name": "subject_confirmation",
    "confidence": 1.142784824002184e-16
  },
  {
    "name": "out_of_scope",
    "confidence": 1.1259906752871933e-16
  },
  {
    "name": "stop",
    "confidence": 1.1062653293535598e-16
  },
  {
    "name": "mood_unhappy",
    "confidence": 6.255450150291825e-17
  },
  {
    "name": "affirm",
    "confidence": 6.036427282466564e-17
  },
  {
    "name": "bot_challenge",
    "confidence": 5.838332745883385e-17
  },
  {
    "name": "greet",
    "confidence": 3.7029527596879434e-17
  }
],
```

[Figure 4: Intent ranking ]

Performance validation shows robust intent recognition with perfect confidence (1.0) for "my mouse is not working," accurately extracting "device" (99.96% confidence) and "issue" (99.98% confidence) entities.

### Form-Based Data Collection

**Form Activation and Management:** The support\_form activates automatically upon support\_request intent detection, maintaining conversation state through multiple slots: type, category, subject, description, priority, and user\_name. Rule-based activation ensures reliable form progression while handling interruptions appropriately.

```

Next message:
where is sri lanka
{
  "text": "where is sri lanka",
  "intent": {
    "name": "out_of_scope",
    "confidence": 0.9976550340652466
  },
  "entities": [],
  "text_tokens": [
    [
      0,
      5
    ],
    [
      6,
      8
    ],
    [
      9,
      12
    ],
    [
      13,
      18
    ]
  ],
  "intent_ranking": [
    {
      "name": "out_of_scope",
      "confidence": 0.9976550340652466
    },
    {
      "name": "greet",
      "confidence": 0.0008415703778155148
    },
    {
      "name": "support_request",
      "confidence": 0.0003671314043458551
    },
    {
      "name": "bot_challenge"
    }
  ]
}

```

[Figure 5: Out-of-scope queries handling]

**Dynamic Slot Population:** The system employs conditional slot mapping respecting active loop states. Critical slots use influence\_conversation: true for dialogue flow impact, while supporting slots avoid conversation interference. Initial user input triggers ActionCaptureUserInput for automatic categorization through semantic analysis.

### Automated Classification Systems

#### Semantic Classification Process

**Metadata-Driven Classification:** The Queries.yml file contains 12 representative IT support scenarios forming the classification foundation. The converter.py script generates enriched metadata including issue categorization and keyword extraction, creating output\_with\_metadata.json as the knowledge base.

**Classification Algorithm:** User input converts to 384-dimensional embeddings using all-MiniLM-L6-v2, compared against pre-computed category embeddings. Cosine similarity scores determine optimal classification, enabling accurate categorization of novel expressions based on semantic meaning.

## **Subject Line Generation**

```
214 prompt = """You are an expert IT support ticket analyst. Create a professional, concise subject line for an IT support ticket.
215
216 CONTEXT:
217 - Issue Type: {type_}
218 - Category: {category}
219 - User's Initial Request: {user_input}
220 - Detailed Description: {trouble_details}
221
222 CRITICAL INSTRUCTIONS:
223 1. Focus on the context mentioned in the user's initial request: "{user_input}" and the detailed description: "{trouble_details}".
224 3. Keep it concise (6-8 words maximum).
225 4. Use clear, professional terminology.
226 5. Include the problem and required action.
227 6. Ignore secondary mentions (like ports, cables) - focus on the main device.
228 7. Use proper grammar and be direct.
229
230 EXAMPLES OF EXCELLENT SUBJECTS:
231 - "USB Mouse Hardware Failure - Replacement Required"
232 - "Word Software License Expired - Immediate Action Required"
233 - "Printer Driver Installation Failed - Support Required"
234 - "Monitor Display Issues - Hardware Troubleshooting Required"
235
236 IMPORTANT: If user mentions "mouse not working", the subject should be about the MOUSE, not about ports or other components and Ignore Notes or additional conte
237
238 Generate ONLY the subject line. Be direct and professional.
239
240 Subject"""
241
242     try:
243         # Call Ollama API with optimized parameters for professional output
244         # Use localhost for local development, host.docker.internal for Docker
245         ollama_host = os.getenv("OLLAMA_HOST", "localhost")
246         response = requests.post(
247             f"http://[{ollama_host}]:11434/api/generate",
248             json=([
249                 "model": "phi3:mini",
250                 "prompt": prompt,
251                 "stream": False,
252                 "options": {
253                     "temperature": 0.2,      # Low temperature for consistent, professional output
254                     "top_p": 0.9,          # Focus on most probable tokens
255                     "top_k": 40,           # Limit vocabulary to top 40 most likely words
256                     "repeat_penalty": 1.1, # Slightly discourage repetition
257                     "num_predict": 50     # Increased limit for complete subjects (was 15)
258             })
259     
```

[Figure 6: Prompt structure for request subject]

Contextual prompt engineering includes user requests, issue types, categories, and trouble details. LLM parameters use low temperature (0.2) for consistency with professional vocabulary optimization. Graceful fallback to manual entry ensures reliability when AI components are unavailable.

## Priority Detection Algorithm

**Hybrid Scoring Mechanism:** The priority detection combines keyword matching (60% weight) and semantic similarity (40% weight) across four levels: Critical, High, Medium, Low.

Scoring Component	Mouse Issue	Server Down
Processed Text	"my mouse is not working and i can't do my work"	"server down"

*Figure 3 : Examples for Analysis*

### Keyword Matching Scores (60% weight)

Priority Level	Mouse Issue	Server Down
Critical (x4)	0/25 matches = <b>0.00</b>	2/25 matches ("down", "server down") = <b>0.32</b>
High (x3)	2/20 matches ("not working", work impact) = <b>0.30</b>	0/20 matches = <b>0.00</b>
Medium (x2)	0/15 matches = <b>0.00</b>	0/15 matches = <b>0.00</b>
Low (x1)	0/10 matches = <b>0.00</b>	0/10 matches = <b>0.00</b>

Figure 4 : Keyword Matching

### Semantic Similarity Scores (40% weight)

Priority Level	Mouse Issue	Server Down
Critical (x4)	0.15 similarity = <b>0.60</b>	0.95 similarity = <b>3.80</b> ★
High (x3)	0.65 similarity = <b>1.95</b> ★	0.45 similarity = <b>1.35</b>
Medium (x2)	0.35 similarity = <b>0.70</b>	0.10 similarity = <b>0.20</b>
Low (x1)	0.10 similarity = <b>0.10</b>	0.05 similarity = <b>0.05</b>

Figure 5 : Semantic Similarity

### Combined Scores (60% Keyword + 40% Semantic)

Priority Level	Mouse Issue Calculation	Mouse Score	Server Down Calculation	Server Score
Critical	$(0.00 \times 0.6) + (0.60 \times 0.4)$	<b>0.24</b>	$(0.32 \times 0.6) + (3.80 \times 0.4)$	<b>1.712</b>
High	$(0.30 \times 0.6) + (1.95 \times 0.4)$	<b>0.96</b>	$(0.00 \times 0.6) + (1.35 \times 0.4)$	<b>0.54</b>
Medium	$(0.00 \times 0.6) + (0.70 \times 0.4)$	<b>0.28</b>	$(0.00 \times 0.6) + (0.20 \times 0.4)$	<b>0.08</b>
Low	$(0.00 \times 0.6) + (0.10 \times 0.4)$	<b>0.04</b>	$(0.00 \times 0.6) + (0.05 \times 0.4)$	<b>0.02</b>

Figure 6 : Combined Scores

Business Rules Application			
Example	Rule Triggered	Adjustment	Final Score
<b>Mouse Issue</b>	Hardware Rule: "mouse" + "not working"	High +0.5	<b>1.46 (HIGH)</b>
<b>Server Down</b>	System Down Rule: "server down"	Critical +1.5	<b>3.212 (CRITICAL)</b>

Figure 7 : Business Rules Application

Key Differences Analysis			
Aspect	Mouse Issue	Server Down	Impact
<b>Keyword Strength</b>	Moderate (High priority words)	Strong (Critical priority words)	Server issue gets higher base score
<b>Semantic Match</b>	High similarity to "productivity impact"	Very high similarity to "system failure"	Server issue strongly matches critical descriptions
<b>Business Rule</b>	Hardware rule (+0.5)	System down rule (+1.5)	Server gets larger priority boost
<b>Final Classification</b>	HIGH (individual impact)	CRITICAL (business-wide impact)	Correctly distinguishes scope of impact

Figure 8 : Key Differences Analysis

### Keyword Matching Component:

- Critical Priority (weight=4): 25 keywords including "down", "offline", "security breach"
- High Priority (weight=3): 20 keywords including "slow", "not working", "affecting work"
- Medium Priority (weight=2): 15 keywords including "installation", "printer", "software"
- Low Priority (weight=1): 10 keywords including "question", "help", "how to"

**Semantic Similarity Component:** Compares input against priority-level descriptions using sentence transformers. Business rules provide additional scoring adjustments for hardware issues (+0.5 High), system down (+1.5 Critical), and security terms (+2.0 Critical).

### Validation and Integration

#### Multi-Stage Validation Architecture

The system implements comprehensive validation combining technical accuracy verification with user satisfaction confirmation. Technical validation ensures appropriate data types and business logic consistency across form fields.

#### User Confirmation Workflows:

- Subject confirmation presents AI-generated suggestions with acceptance/rejection options
- Priority confirmation displays automated analysis for user verification
- Error handling preserves validated data while requesting correction of problematic fields

## **Final Submission and System Integration**

**JSON Output Generation:** Structured output includes all essential ticket information in standardized format compatible with external ticketing systems. Data integrity verification ensures completeness before final submission through ActionSubmitSupportForm.

**API Integration Architecture:** RESTful interface design enables external system interaction through standard HTTP endpoints. Webhook implementation provides real-time integration capabilities with ticketing systems, CRM platforms, and enterprise applications.

## **Data Collection and Analysis**

### **Data Sources and Processing**

Primary training data sourced from Customer Support Tickets dataset (HuggingFace), filtered for English-language consistency. Additional IT support data structured in YAML format converted to JSON metadata repository for classification ground truth.

### **Bias Mitigation Strategies**

- Stratified sampling addressed class imbalance across request types
- English-only dataset prevented multilingual inconsistencies
- Hybrid priority detection combined rule-based and semantic analysis
- User confirmation workflows maintained human oversight over automated decisions

The methodology demonstrates a comprehensive approach to conversational form automation, combining state-of-the-art AI techniques with robust validation and user experience considerations. The modular architecture ensures scalability while maintaining data privacy and system reliability through local AI model deployment.

## *Knowledgebase component*

The proposed chatbot system is structured into several functional layers: mainly its consist of a RAG system, Vector Retrieval, Conversational Episodic Memory, Caching, Knowledge Graphs, Chain-of-Thought Prompting, Safety & Guardrails, API Layer, and Frontend Interaction. Each layer addresses specific challenges encountered in real-world IT support, ensuring both efficiency and contextual accuracy.

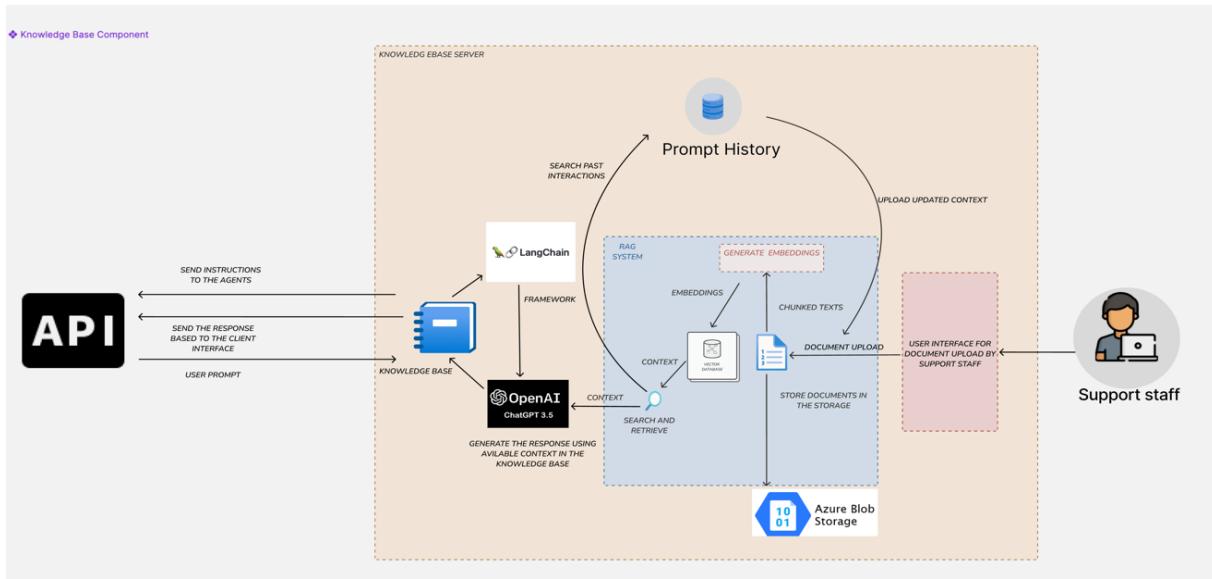


Figure 9 - Knowledge base component

### 1) Vector Retrieval Layer

The **vector retrieval layer** is responsible for extracting the most relevant information from the knowledge base. All knowledge chunks are assigned **unique deterministic identifiers**, allowing easy updates and replacements. When a user submits a query, the system first identifies the most similar content chunks using **vector similarity scoring**. A **re-ranking mechanism** is applied afterward to enhance accuracy by prioritizing relevant candidates while preserving the original ranking for robustness.

Currently, metadata filters are applied to improve answer precision, although keyword-based filtering is not yet integrated. In real-world IT support, this layer reduces irrelevant responses, ensures rapid retrieval, and helps prevent misinformation, which is a common challenge in automated customer service systems.

### 2) Re-Ranking Mechanism

The **re-ranking module** refines the initial retrieval results by emphasizing relevance without drastically altering the order of top candidates. This step mitigates the risk of presenting users with irrelevant or inaccurate responses, a common issue in vector-based retrieval. The current

system uses heuristic scoring but is designed to accommodate future improvements through more sophisticated ranking models that balance **accuracy and response latency**.

```
reranker = CrossEncoder('cross-encoder/ms-marco-MiniLM-L-6-v2', max_length=512)

async def re_rank_documents(query, documents):
    """
    Re-rank documents based on their relevance to the query.

    Args:
        query (str): The user query
        documents (list): List of document strings to re-rank

    Returns:
        dict: Dictionary mapping document IDs to relevance scores
    """
    print(documents)
    try:
        # Prepare document pairs for scoring
        document_pairs = []
        for i, doc in enumerate(documents):
            document_pairs.append([query, doc])

        # Score the documents using the cross-encoder model
        def score_docs():
            return reranker.predict(document_pairs)

        loop = asyncio.get_event_loop()
        scores = await loop.run_in_executor(None, score_docs)

        # Create a dictionary mapping document IDs to scores
        result = {}
        for i, doc in enumerate(documents):
            result[doc] = scores[i]
    except Exception as e:
        print(f"Error: {e}")
    finally:
        if 'loop' in locals():
            loop.close()

    return result
```

### 3) Conversational Episodic Memory

To provide consistent and personalized support, the system maintains a **conversational episodic memory**, tracking previous interactions, identified problems, root causes, and solutions. Multiple indexing strategies in a NoSQL database, such as MongoDB, allow retrieval of relevant prior cases, even for differently worded queries. This enables the reuse of validated troubleshooting guidance and supports continuity across sessions. Although sensitive information filtering is not yet automated, plans include privacy-preserving mechanisms to ensure compliance with data protection standards.

```

# Get chat history endpoint
@app.get("/history/{user_id}")
async def get_chat_history(user_id: str, limit: int = 10):
    try:
        history = list(chat_collection.aggregate([
            {
                "$search": {
                    "index": "default",
                    "queryString": {
                        "query": user_id,
                        "defaultPath": "user_id"
                    }
                }
            },
            {"$sort": {"timestamp": -1}}, # Sort by newest chats
            {"$limit": limit},
            {"$project": {"user_query": 1, "ai_responses": 1, "analysis": 1, "timestamp": 1, "_id": 0}}
        ]))

        return {"history": history}
    except Exception as e:
        raise HTTPException(status_code=500, detail=f"Error retrieving chat history: {str(e)}")

```

## 4) Caching Layer

The caching layer optimizes performance by storing query results and frequently accessed information, such as organizational policies or common troubleshooting steps, for reuse within a configurable time frame (currently 24 hours). On system startup, historical data may be pre-cached to accelerate initial responses. The caching mechanism ensures reliability: even if the cache is temporarily unavailable, the system continues operation. Future improvements may involve **query clustering** to deliver smarter and faster responses.

## 5) Policy Knowledge Graph

The system incorporates a **knowledge graph** to handle complex IT and HR policies. For example, a hardware issue query triggers a search through the graph for relevant policies, which are then integrated into the generated answer. The knowledge graph currently includes domains such as **access control, onboarding, remote work, and security policies**, and is loaded dynamically to avoid system downtime. Multi-step reasoning and relationship weighting are planned enhancements for improved contextual inference. This structure ensures that responses are policy-compliant and reduces the likelihood of procedural errors in enterprise IT support.

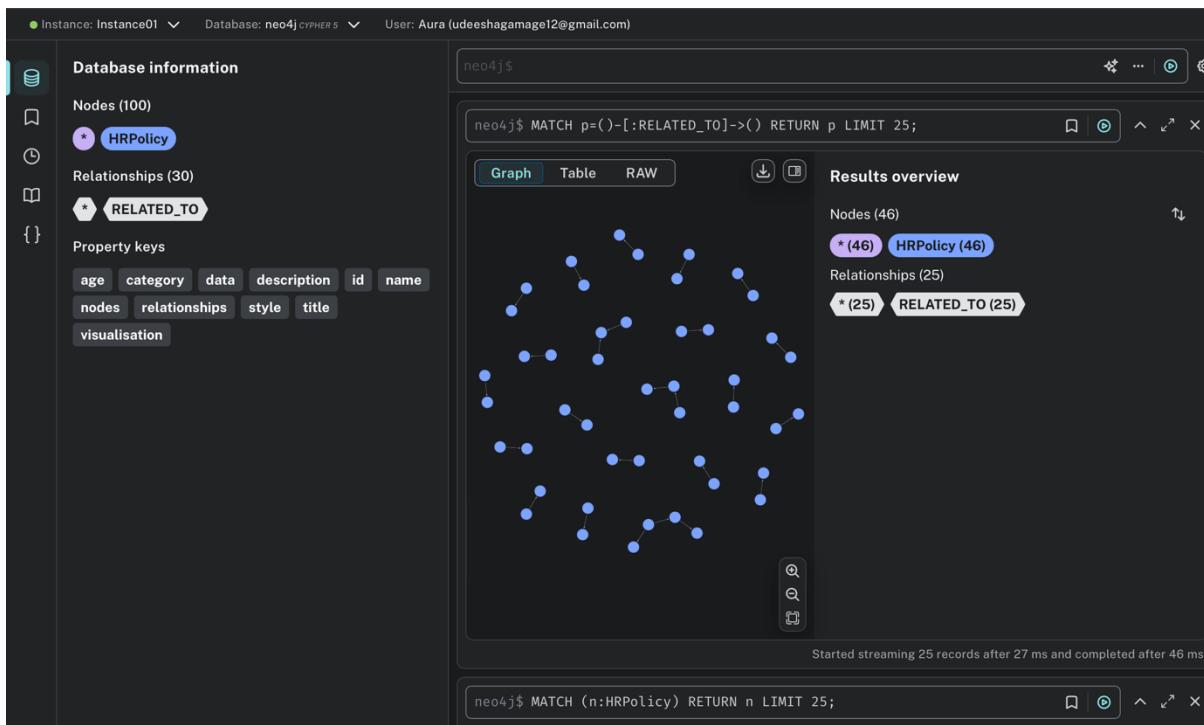


Figure 10 - knowledge graph

## 6) Chain-of-Thought (CoT) Prompting

**Chain-of-Thought prompting** decomposes AI reasoning into transparent, stepwise segments, including problem analysis, possible causes, solution steps, and preventive recommendations. This approach aligns with the **ITIL incident management lifecycle**, guiding the AI from problem detection to mitigation and prevention. While longer prompts increase computational load, only highly relevant information is included in each reasoning step to optimize performance. CoT prompts enhance interpretability and facilitate verification of AI outputs by IT personnel.

## 7) Safety and Guardrails

To prevent misuse or unsafe operations, the system implements **deterministic filters** for high-risk actions (e.g., credential access or destructive commands). Ambiguous cases are routed through a lightweight AI classifier. The design prioritizes **availability and minimal restriction**, ensuring legitimate queries are processed efficiently. Future improvements include AI-driven checks against malicious inputs and similarity-based detection. Currently, query rate limiting is not implemented, but this is planned for enhanced security.

## 8) API Layer

The **API layer**, implemented using Fast API, exposes endpoints for chat, HR policy queries, user history, file uploads, and knowledgebase access. Startup routines initialize caches, knowledge

graphs, and retrieval clients, while defensive programming ensures subsystem failures do not compromise overall system availability. The stateless API design allows **horizontal scalability** behind load balancers, with future enhancements planned for request-level tracing and performance monitoring.

```
class GuardrailViolation(Enum):
    INAPPROPRIATE_CONTENT = "inappropriate_content"
    SECURITY_THREAT = "security_threat"
    PERSONAL_INFO_REQUEST = "personal_info_request"
    OUT_OF_SCOPE = "out_of_scope"
    MALICIOUS_INJECTION = "malicious_injection"
    EXCESSIVE_LENGTH = "excessive_length"      You, 2 months ago • latest implementation

You, 2 months ago | 1 author (You)
class ITSupportGuardrails:
    def __init__(self):
        self.max_query_length = 1000
        self.blocked_patterns = [
            r'(?i)\b(hack|hacking|crack|cracking|exploit|vulnerability|breach)\b',
            r'(?i)\b(password|pwd|passwd)\s+(for|of)\s+\w+',
            r'(?i)\b(personal|private|confidential)\s+(information|data)\b',
            r'(?i)\b(salary|wage|pay|financial|budget)\b',
            r'(?i)\b(delete|remove|drop|truncate)\s+(database|db|table)\b',
            r'(?i)\b(admin|administrator|root)\s+(password|access|credentials)\b'
        ]

        self.it_keywords = [
            'computer', 'laptop', 'software', 'hardware', 'network', 'wifi', 'email',
            'printer', 'password', 'login', 'install', 'update', 'error', 'crash',
            'slow', 'freeze', 'virus', 'malware', 'backup', 'recovery', 'vpn',
            'remote', 'access', 'device', 'monitor', 'keyboard', 'mouse', 'usb',
            'server', 'database', 'application', 'browser', 'internet', 'connection'
        ]
```

## 9) Frontend Interaction Layer

The frontend provides a **user-friendly chat interface**, allowing both dynamic search and guided navigation through HR policies. Policy directives are combined with structured reasoning, reducing cognitive load and improving usability. The architecture is **UI-agnostic**, ensuring future frontend frameworks, such as React, can consume the same data contracts without modification.

### B. Evaluation Framework

The system is evaluated using a combination of standard IT support queries and **out-of-scope or nonsensical requests**. Metrics include **answer faithfulness, contextual relevance, and memory utilization**. Performance is monitored over time using radar charts and heatmaps. Limitations include the absence of temporal drift evaluation and a calibrated human-annotated dataset, which are areas for future work.

Tool/Technology	Purpose/Use
OpenAI GPT-3.5/4	Used for natural language understanding and generating responses in the chatbot based on user queries.
<b>Chain-of-Thought Strategy</b>	<b>we provide the AI clear and structured prompts. The approach assists it to perceive the question appropriately When the prompts are loosely defined you end up with improper or fabricated answers.</b>
ChromaDB	<b>It simple tool that stores our data and is written in Python.it's a vector database which can store vector embeddings Others such as FAISS do not save data, and Weaviate takes harder work to run.</b>
MongoDB	NoSQL database for storing and managing data such as user queries, system responses, and knowledge base documents.
Fast API	Python-based web framework used for developing and hosting the backend API for the chatbot application.
Azure	Cloud platform for hosting the application, storing data, and providing computing resources for the system.
Huggingface	Provides pre-trained AI models and tools for natural language processing tasks, integrated into the chatbot system.
Redis	To store the memory in cache for faster data retrieval
Neo4j	To store the HR policies of the company which support to answer queries
<b>RAGAS</b>	<b>is used to evaluate that everything is running smoothly. It executes automated checks on a basis of standard scores.</b>
Guardrails	Restrict the LLM to provide unrelated answers to the users queries

## D. Efficiency Measures

Optimizations were implemented to enhance speed and minimize computational costs:

- **Precomputed embeddings** to reduce query-time CPU usage

- **Top-N re-ranking** to process only the most relevant results
- **Redis TTL** to remove outdated cache entries after 24 hours
- **Lazy service initialization** to prevent startup failures
- **Chunk tuning** to maintain context while reducing prompt size
- **Parsed CoT storage** to minimize token usage and enable analytics

## E. System Baseline vs. Gains

**Baseline risks** of naive keyword-search-based systems include:

- Illusory workflows
- Ignoring policy compliance
- Repetition of resolved issues

**System gains** from the proposed architecture:

- Episodic memory reduces redundancy and maintains continuity
- Policy-aware guidance ensures compliance
- Grounded retrieval improves factual accuracy
- Stepwise reasoning enhances interpretability and user-friendliness

## F. Risk Mitigation

Potential risks and mitigations include:

- **Model drift:** Managed via prompt archiving, temperature adjustments, and regression testing
- **Graph sparsity:** Incremental policy ingestion and fallback mechanisms
- **Cache poisoning:** Future hash-based allowlists and authentication
- **Sensitive data leakage:** Regex and entropy-based pre-storage filters

## G. Best Practices and Architectural Patterns

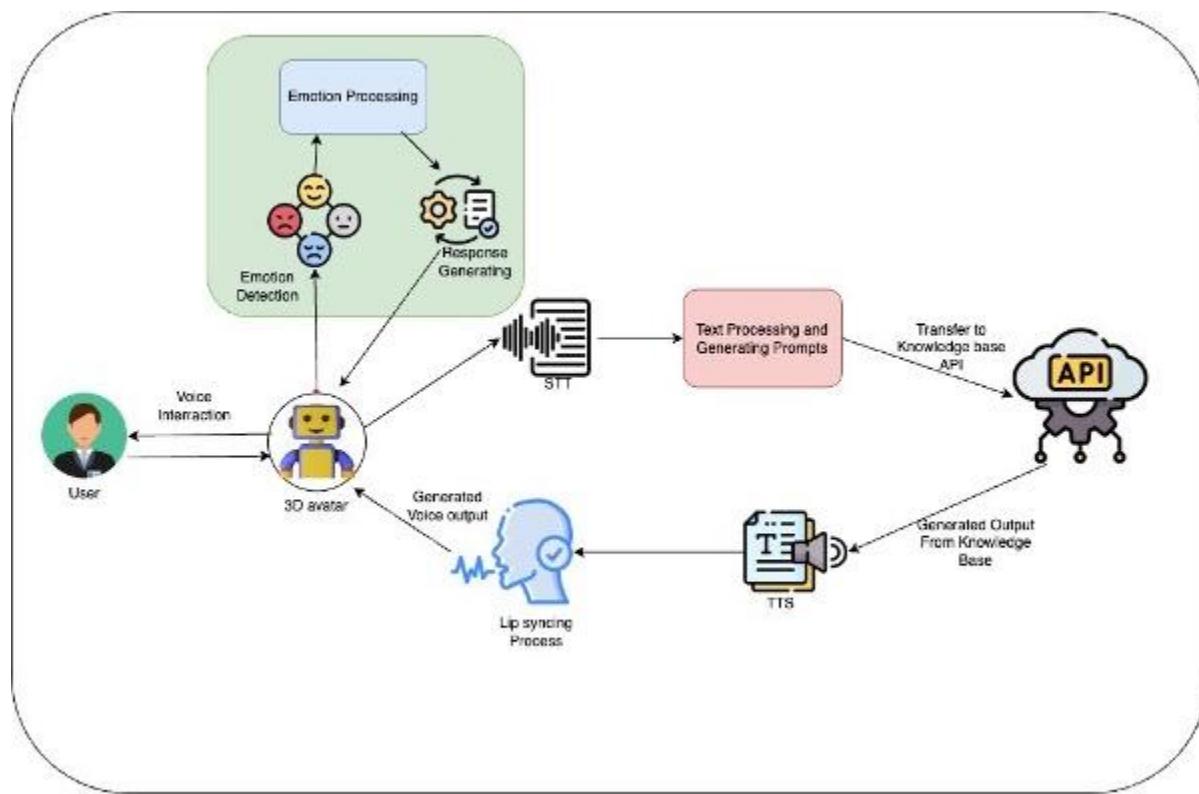
The system leverages design patterns and architecture strategies to ensure scalability, maintainability, and fault tolerance:

- **Singleton pattern** for database connections reduces overhead and ensures consistency
- **Strategy pattern** for business logic allows flexible algorithm selection
- **Microservices architecture** for modular, independently deployable services
- This combination enhances system resilience, adaptability, and maintainability

### *3D avatar with voice Integration and Emotion Intelligence*

The methodology adopted for the development of the 3D avatar module and its supporting components followed an iterative, agile-inspired approach that emphasized continuous refinement. Rather than treating the system as a collection of disconnected modules, it was designed as an integrated whole in which speech, emotion, and visual feedback worked together to create a supportive IT companion. The process combined creative prototyping with technical rigor, ensuring that the system addressed both functional requirements and user experience expectations.

### *System Architecture Overview*



The overall architecture was designed to mimic the nervous system of a human assistant, enabling inputs, emotional reasoning, and expressive outputs to work in harmony. Users typically interacted with the chatbot through voice input captured via the microphone, while webcam-based facial cues could be optionally included for richer emotional sensing. The collected input was processed by specialized modules for speech recognition, emotion detection, and sentiment analysis, before being directed to the response generator. The generated output was then expressed through a 3D avatar that delivered speech, synchronized lip movements, and appropriate gestures. A Flask-based

backend tied these modules together and maintained smooth communication via APIs and WebSockets. This architecture ensured that the system was modular, extensible, and secure while maintaining real-time responsiveness.

### ***3D Avatar Implementation***

The 3D avatar was developed to move beyond conventional text-based chatbots and create a more human-like interaction. Ready Player Me SDK was selected as the framework for avatar creation because of its ease of use, support for customization, and professional visual quality. Avatars were exported as GLB models and integrated into Unity, chosen for its robust animation engine and real-time rendering capabilities. Customization options such as skin tone, hairstyle, and clothing were adjusted to give the avatar the appearance of a friendly IT staff member.

Once imported into Unity, the avatar was programmed with expressive states through blendshapes, covering emotions such as neutral, concerned, encouraging, and frustrated. Unity's Animator Controller allowed transitions between these states based on inputs received from the emotion detection module. Gestures were added to increase realism, including nodding during acknowledgment, pointing during instructions, and typing motions while guiding troubleshooting. These movements helped replicate the dynamics of human conversation and gave users the impression of interacting with a colleague rather than a machine.

Lip synchronization was also essential to preserve realism. Rhubarb Lip Sync was integrated to map spoken audio to corresponding mouth shapes. When the chatbot generated audio responses, Rhubarb analyzed them and produced viseme sequences that were transmitted to Unity. A custom script then applied these visemes to the avatar's facial blendshapes, ensuring that lip movements matched spoken words in real time. User evaluations revealed that this feature significantly improved engagement and trust, making the system appear both reliable and approachable.

### ***Speech-to-Text (STT)***

Accurate speech recognition formed the foundation for natural communication. Whisper, an open-source model developed by OpenAI, was selected due to its high transcription accuracy, resilience in noisy conditions, and multilingual support. Input was captured from the system microphone using PyAudio, and a preprocessing pipeline was implemented to filter noise, normalize volume levels, and detect silence, ensuring that the model received clean and consistent audio streams.

The Whisper model then converted these inputs into spectrograms, processed them through its transformer layers, and produced textual tokens that were combined into sentences. This pipeline was optimized to achieve low latency, typically under two seconds on CPU and less than one second on GPU, enabling real-time interaction. Even in noisy office environments, Whisper demonstrated reliability above 90% accuracy, making it suitable for IT support contexts.

### ***Text-to-Speech (TTS)***

While STT allowed the chatbot to “listen,” the TTS system gave it a “voice.” ESPNet was chosen as the TTS framework because of its ability to generate natural speech with realistic intonation. The LJ Speech dataset, containing over 24 hours of labeled audio clips, was used to train the model to produce neutral and intelligible American English speech.

The pipeline began with text normalization, where abbreviations and symbols were expanded into spoken equivalents. Prosody adjustments were introduced to reflect emotional states: slower, softer tones for stressed users, energetic delivery for encouraging responses, and steady speech for technical explanations. ESPNet then generated audio outputs in .wav format, which were normalized for consistent playback.

Finally, the audio was integrated with Rhubarb Lip Sync to synchronize the avatar’s mouth movements with the generated voice. This combination ensured that both verbal and visual communication were coherent, reinforcing the impression of interacting with a human-like assistant.

### ***Sentiment Analysis***

Understanding user emotions through text was critical to the system’s ability to respond empathetically. For this purpose, a sentiment analysis model was developed using the IMDb movie review dataset. Reviews were tokenized with BERT’s tokenizer and fine-tuned using a transformer-based architecture. While the dataset originally provided binary labels (positive and negative), a third neutral category was introduced through domain-specific IT phrases.

Training employed AdamW optimization and cross-entropy loss, with metrics including accuracy and F1-score. The final model achieved approximately 85% accuracy, which was sufficient for integration into the IT support setting. Domain adaptation was performed using manually labeled IT queries such as “this update sucks” or “thanks, it finally works,” which allowed the model to generalize effectively.

### ***Multimodal Emotion Fusion***

A central innovation of the methodology was the fusion of multimodal emotional signals. Instead of relying on a single modality, the system integrated facial, vocal, and textual cues to derive a unified emotional state. Facial expressions were analyzed through DeepFace, vocal tones were classified with SpeechBrain, and textual sentiment was processed by the BERT-based classifier.

Two fusion strategies were explored: majority voting, where the most common label among modalities was selected, and weighted averaging, where greater importance was assigned to face and voice (40% each) compared to text (20%). Weighted averaging was ultimately chosen as it better reflected the dynamics of real-world IT conversations, where tone of voice and expression often carry more weight than words alone.

The final emotion label was sent to both the TTS system, which adjusted pitch and prosody, and the 3D avatar, which switched to the corresponding facial expression and gesture. This multimodal fusion ensured that responses were not only technically correct but also emotionally appropriate.

### ***Backend Integration***

The backend served as the coordinator of all modules, providing the infrastructure for communication and synchronization. Flask was chosen due to its lightweight design and compatibility with Python-based models. RESTful APIs were created for processing user input, generating responses, and updating avatar states.

Some of the key endpoints included:

- /process\_input: processed user audio or text, invoking STT and sentiment analysis.
- /generate\_response: produced replies using the knowledge base and emotion detection outputs, then sent them to the TTS system.
- /update\_avatar: controlled Unity-based animations and expressions in real time.

WebSockets were employed for fast updates, ensuring that avatar reactions occurred immediately. Security was enforced using OAuth authentication, which safeguarded sensitive data such as webcam streams.

### ***Iterative Development and Testing***

The methodology was applied iteratively, with each module developed, tested, and refined in isolation before full integration. Early prototypes were tested in simulated IT support scenarios, such as handling password resets, software crashes, and VPN configuration issues. Feedback from these tests was used to refine speech accuracy, response latency, and emotional synchronization.

The agile-inspired process emphasized short development cycles that incorporated experimentation and feedback. For example, initial lip-sync prototypes were rudimentary but gradually improved through iterative testing until they achieved realistic synchronization. Similarly, the emotion fusion strategy evolved from simple majority voting to weighted averaging as insights were gained from testing.

# Multi Agentic System

The architecture of the proposed system is structured into four primary layers, which are organized hierarchically to ensure clarity, modularity, and maintainability. The design follows a layered architectural style, where each layer has distinct responsibilities but also interacts seamlessly with the others through well-defined communication channels. Figure X presents an overview of the system's architecture, where the four layers—Client, API, Orchestration, and Core System Components—are arranged to demonstrate how user interaction flows through the system.

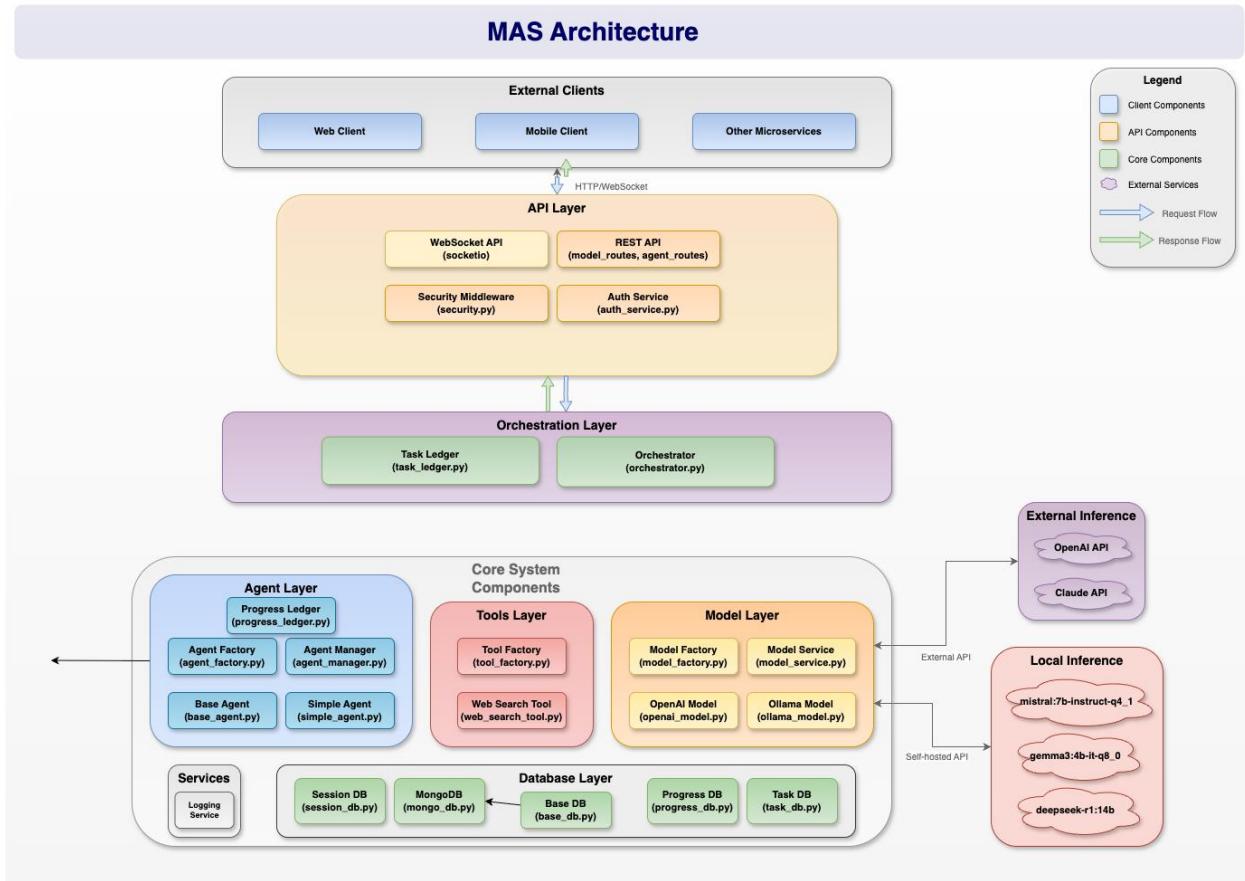


Figure 11 - Multi Agentic System Architecture

At the top of the architecture lies the **Client Layer**, which represents all possible points of interaction with the system. This layer primarily consists of end users who initiate requests or interact with the system through conversational interfaces. Additionally, it includes other microservices that can interact with the system programmatically via the API Gateway. By incorporating both human and machine clients, the system is not limited to direct user interactions but is also capable of integration with external services and applications. The Client Layer thus establishes the entry point into the system and highlights its openness for expansion and integration.

The **API Layer** is positioned beneath the Client Layer and acts as the system's secure entry channel. This layer exposes REST APIs that serve as a unified interface for all inbound requests, regardless of whether they originate from users or external services. A significant feature of the API Layer is its inclusion of authentication and security middleware. These mechanisms protect the system from unauthorized access, enforce access control policies, and ensure the integrity of transmitted data. By centralizing request handling at the API Layer, the architecture enforces a consistent interaction model while decoupling clients from the internal complexities of the system.

The **Orchestration Layer** follows as the central coordination point within the architecture. This layer contains two essential elements: the Orchestrator and the Task Ledger. The Orchestrator is responsible for managing workflows, distributing tasks to the appropriate components, and ensuring that complex requests are broken down into manageable units of work. In support of this, the Task Ledger functions as a persistent tracking mechanism that records task assignments, progress, and results. Together, these two elements provide a reliable orchestration framework, ensuring that tasks are not only executed efficiently but also monitored for accountability and fault recovery. This makes the Orchestration Layer a critical bridge between incoming requests and the underlying system functionalities.

At the foundation of the architecture is the Core System Components Layer, which encapsulates the system's internal intelligence and functionality. Within this layer reside several submodules, including Agents, Tools, Models, Services, and Databases, each designed using object-oriented principles and custom patterns built from scratch. The Agents act as autonomous decision-making entities, following a Strategy Pattern to dynamically select task-solving behaviors and a Observer Pattern to react to updates from other agents or the orchestrator. The Tools are implemented using a Command Pattern, allowing agents to invoke operations flexibly and queue tasks dynamically. Models provide computational intelligence, integrating LLMs through Ollama while adhering to an Adapter Pattern to standardize interfaces for reasoning, dialogue, and knowledge augmentation. Services encapsulate reusable business logic using Facade and Singleton Patterns, ensuring centralized access and consistent state management. Finally, Databases are wrapped with Repository Patterns to abstract data access, supporting both structured and unstructured storage. Collectively, these components leverage OOP design patterns to form the operational core, ensuring that requests are executed with precision, scalability, and maintainable extensibility.

By structuring the system across these four layers, the architecture achieves a balance between modularity and integration. Each layer abstracts complexity from the one above it while exposing the necessary interfaces for communication. The Client interacts only with the API, the API delegates to the Orchestration Layer, and the Orchestration Layer coordinates with the Core Components to execute tasks. This hierarchy allows for a clean separation of concerns, supporting scalability and maintainability while also making it easier to expand individual layers without disrupting the entire system. For example, additional microservices can be added at the Client Layer without affecting the Orchestration Layer, or new tools and models can be integrated into the Core System Components without requiring redesign at the higher levels.

## Orchestrator

The Orchestrator is the central coordinating class of the system, serving as the brain that bridges high-level goals with concrete execution. In Here as a Individual Class Following OOP, Its main function is to receive abstract objectives from users or external services and transform them into smaller, executable tasks. This decomposition process is crucial: complex goals such as “analyze market data and prepare a summary report” cannot be handled by a single agent or tool in one step. Instead, the Orchestrator consults available model services, often large language models (LLMs), to generate subtasks that are logically ordered, validated, and normalized before execution.

From an object-oriented design perspective, the Orchestrator implements the Mediator pattern. Rather than having agents communicate directly with each other or with databases, the Orchestrator mediates all interactions. This ensures that each component remains loosely coupled, preserving modularity and easing maintainability. For example, if a new agent type is introduced, it does not need to directly coordinate with other components; instead, it plugs into the Orchestrator’s delegation flow. This pattern reduces interdependencies and allows for a scalable architecture.

Internally, the Orchestrator collaborates with the AgentManager to assign tasks to the right agent. Assignment is not a static process. The Orchestrator takes into account agent capabilities, workload, and availability. If a failure occurs, the Orchestrator dynamically reassigns the task to a fallback agent or retries execution based on system policies. This resilience reflects the Strategy pattern, where different delegation strategies can be applied depending on the task type, error condition, or dependency constraints. Additionally, it enforces reliability by monitoring execution states and making adjustments on the fly.

Another key role of the Orchestrator is queue management. Tasks are not executed in isolation; they often have dependencies or must follow a particular sequence. The Orchestrator ensures that dependencies are respected and that the execution flow proceeds in a way that aligns with user intent. This task scheduling capability allows for concurrent workflows, where multiple goals can be managed in parallel without conflict.

Overall, the Orchestrator is not merely a dispatcher. It embodies intelligence, resilience, and adaptability. Its careful use of object-oriented principles and design patterns ensures that the system can manage complex workflows, handle failures gracefully, and continuously adapt as new components (agents, tools, or models) are added.

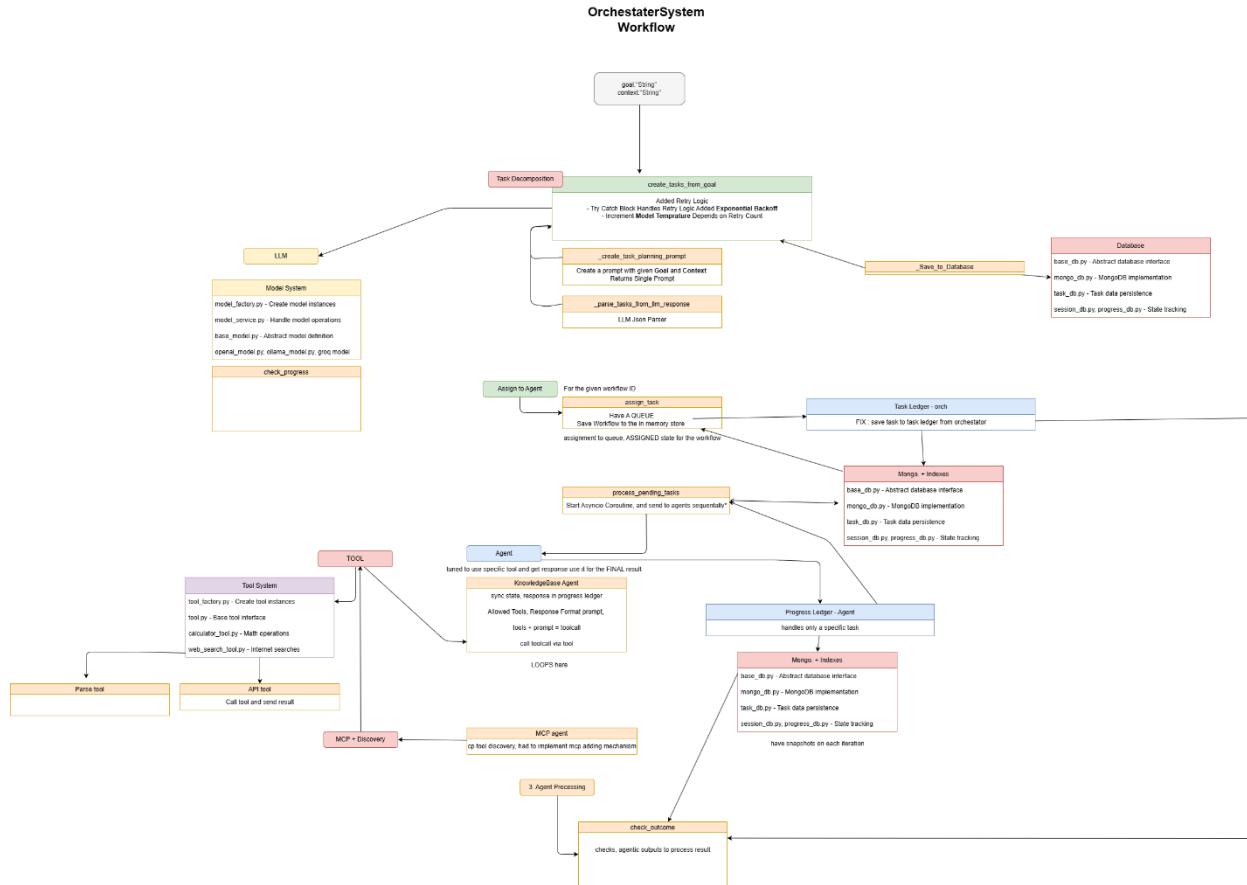


Figure 12 - Orchestrator System Workflow

## Exponential Backoff Algorithm for Retry Mechanisms

To ensure reliable task execution within the orchestrator, an **Exponential Backoff Algorithm** was implemented for retrying failed tasks. When a task fails due to transient issues such as network latency, unavailable services, or temporary API errors, the orchestrator waits for progressively longer intervals before retrying, reducing the likelihood of overwhelming the system or causing cascading failures. This approach improves fault tolerance and system resilience while maintaining efficient resource utilization. The image below illustrates the backoff sequence and integration within the orchestrator.

Figure 13 - MAS Exponential Backoff Implementation

```

while retry_count < max_retries:
    try:
        # If this is a retry attempt, log it
        if retry_count > 0:
            logging_service.log_warning(f'Retrying task creation (attempt {retry_count}/{max_retries})')

        # Prepare prompt for LLM to generate tasks
        prompt = self._create_task_planning_prompt(goal, context)

        # Each retry uses slightly different temperature for more varied outputs
        temperature = 0.7 + (retry_count * 0.1) # Increase temperature slightly with each retry

        # Get task plan from LLM using OllamaModel's generate method with adjusted temperature
        logging_service.log_info(f'Generating task plan for goal: {goal[:50]} if goal else '' ... (temperature: {temperature:.1f})')
        response = await self.llm.generate(prompt, temperature=temperature)
    
```

### LLM Temperature Tuning Methodology for Retry Optimization

For tasks involving large language model reasoning, the orchestrator employs a **Temperature Tuning Methodology** to optimize retries. By adjusting the temperature parameter, the system can control the randomness of LLM outputs, making retries more predictable or exploratory based on task requirements. Lower temperature values produce more deterministic responses suitable for critical tasks, while higher values allow creative solutions in open-ended workflows. This method enhances reliability during task retries by reducing hallucinations and improving task execution quality.

```

async def _make_request_with_retry(self, method: str, url: str, json_data: Dict[str, Any]) -> Dict[str, Any]:
    """Make HTTP request with exponential backoff retry logic."""
    attempt = 0
    last_error = None

    while attempt < self.max_retries:
        try:
            return await self._make_request(method, url, json_data)
        except asyncio.TimeoutError as e:
            last_error = f'Request timeout (attempt {attempt + 1}/{self.max_retries})'
            logging_service.log_warning(last_error)
        except Exception as e:
            last_error = str(e)
            logging_service.log_warning(f'Request failed (attempt {attempt + 1}/{self.max_retries}): {last_error}')

        # Exponential backoff only if not the last attempt
        if attempt < self.max_retries:
            await asyncio.sleep(self.backoff_factor ** attempt)
        attempt += 1

    raise Exception(f'Max retries exceeded. Last error: {last_error}')

```

*Figure 14 - MAS LLM Temperature Tuning Implementation*

## 2.2. Commercialization aspects of the product

The developed IT Support Chatbot, integrating 3D avatars, real-time voice communication, and multimodal emotion sensing, presents a compelling opportunity for commercialization within the growing landscape of enterprise automation. Originally conceived as a research prototype, the

system addresses practical challenges faced in IT service delivery, customer support, and organizational communication. Traditional IT support solutions, including standard chatbots and ticketing systems, often lack empathy, adaptability, or visual engagement, which reduces their effectiveness in high-stress scenarios such as troubleshooting critical technical failures. By combining natural language interaction, human-like avatars, and emotion-aware responses, this solution provides a more approachable and reliable user experience, positioning it as a market-ready product with strong potential across sectors such as corporate IT, education, healthcare, and finance.

### ***Target Audience***

The proposed IT Support Chatbot is designed to serve a wide range of users across multiple industries, leveraging features such as empathetic responses, emotion fusion, and interactive avatars to create adaptable solutions tailored to various operational contexts.

**Corporate IT Helpdesks:** Large and medium-sized enterprises often face recurring technical issues including password resets, network configuration errors, software installation failures, and hardware troubleshooting. The chatbot functions as a first-level assistant, providing immediate guidance and reducing the operational burden on IT personnel. Its 24/7 availability ensures continuous service delivery, enhancing overall productivity and minimizing downtime in mission-critical operations.

**Educational Institutions:** Schools, universities, and training centers increasingly rely on digital platforms for administration and learning. Common challenges include accessing online portals, resolving issues during digital examinations, and maintaining connectivity during virtual classes. The 3D avatar interface makes interactions engaging and approachable, supporting students and staff in resolving technical problems efficiently. Furthermore, the system can act as a virtual teaching assistant, guiding users step-by-step through technical processes.

**Healthcare Sector:** Hospitals and clinics depend on complex digital systems for patient management, telemedicine, and data handling. IT disruptions can directly affect patient care. The chatbot provides instant guidance to medical staff, reduces response time during critical incidents, and delivers voice-enabled accessibility for high-pressure environments. Emotion-aware responses ensure calm and supportive interactions, mitigating stress during critical scenarios.

**Finance and Banking Sector:** Financial institutions require secure, reliable IT systems. Employees often need rapid resolutions for system crashes, login failures, or data access issues. By integrating voice and avatar capabilities, the chatbot enhances user confidence and professional interaction. Emotion analysis ensures empathetic handling of user frustration, an essential feature for sensitive financial operations.

**General Users with Accessibility Needs:** The system supports individuals with visual impairments through voice interaction, literacy challenges via text-to-speech features, and stress or anxiety through empathetic avatar interactions. This inclusivity broadens the system's usability beyond conventional IT support applications.

## **Commercialization Strategy**

The commercialization approach emphasizes a Software-as-a-Service (SaaS) model, offering organizations subscription-based access with flexible pricing tiers. Entry-level packages target small and medium enterprises, while premium versions provide advanced features such as emotion sensing, 3D avatar customization, and analytics dashboards. Strategic partnerships with IT service providers and educational institutions are expected to drive adoption. Cloud deployment through platforms such as AWS, Azure, and GCP ensures scalability, reliability, and low implementation overheads.

The proposed system addresses specific pain points faced by IT teams in medium to large-scale organizations, including government agencies, financial institutions, and corporate enterprises. By streamlining incident management and minimizing system downtime, the chatbot facilitates smooth operational continuity. Over time, this solution can also extend to small and medium-sized businesses (SMBs) requiring reliable, cost-effective IT support.

## **Value Proposition**

The chatbot delivers significant value through cost-effectiveness, scalability, and domain-specific customization. Compared to generic IT support tools, this system has been tailored to meet the specific needs of organizations such as LOLC, enhancing flexibility and alignment with internal operations. Features like internal knowledgebases, active caching, and real-time monitoring reduce repetitive incidents and expedite problem resolution, allowing organizations to lower operational expenses, minimize downtime, and improve customer satisfaction.

## **Market Research and Demand**

The demand for intelligent IT support systems is growing rapidly due to the widespread adoption of digital transformation across industries. Traditional ticketing systems are often inefficient, leading to delays and suboptimal resolution of IT incidents. Within the financial services sector, even minor IT outages can have significant operational and reputational consequences. Comparative analyses of existing solutions such as Zendesk and ServiceNow reveal a lack of cost-effective, locally customizable, and tailorable IT support systems, confirming strong potential demand for the proposed solution within enterprise markets.

## **Revenue Model**

The revenue model combines a subscription-based SaaS approach with tiered pricing according to organizational size and feature requirements. Three tiers are proposed: Starter (up to 500 tickets/month), Professional (up to 2,500 tickets/month), and Enterprise (unlimited tickets with advanced customization). Smaller businesses may access a freemium model with core functionalities, while advanced features such as analytics, integrations, and reporting are offered at premium levels. Professional services, including system integration, domain adaptation, and training, represent an additional revenue stream, typically commanding 1.5–2 times the annual software licensing fee.

## **Development, Deployment, and Maintenance**

The cost of development encompasses software engineering, testing, and integration with third-party tools such as Redis and Chroma DB. Deployment relies on cloud infrastructure to ensure

high availability, scalability, and system reliability. Maintenance covers software updates, bug fixes, and ongoing technical support. Emphasis on user-friendly interfaces, reliance on open-source technologies, and minimal dependency on proprietary software keeps the total cost of ownership within a competitive range, enhancing market appeal.

### **Scalability and Growth Potential**

The system's modular architecture supports horizontal scaling, enabling thousands of concurrent interactions while maintaining low latency. Cloud deployment ensures global accessibility, while on-premise options address data sovereignty and compliance requirements in regulated industries. Pilot implementations within organizations like LOLC can validate system effectiveness before scaling across branches or sectors such as healthcare, government, and education. Future enhancements may include multilingual support, mobile connectivity, predictive analytics, and industry-specific versions targeting healthcare intake, financial services, and citizen services, thereby expanding the total addressable market and long-term sustainability.

### **Market Opportunity and Competitive Advantage**

The chatbot leverages a hybrid AI approach combining conversational form-filling, intelligent categorization, automated subject generation, and hybrid priority detection. Unlike competitors such as Zendesk, ServiceNow, and Freshdesk, the system offers integrated LLM capabilities with local processing through Ollama, ensuring data privacy, regulatory compliance, and reduced dependency on external cloud services. These features provide both operational efficiency and a strong competitive differentiation in the growing conversational AI market, projected to expand from USD 7.61 billion in 2022 to approximately USD 32.62 billion by 2030, with a CAGR of 23.6%.

### **Intellectual Property and Competitive Protection**

Opportunities for intellectual property protection include patentable aspects of the hybrid classification approach and conversation design patterns, while trade secrets protect specialized training methodologies, domain-specific optimizations, and metadata generation processes. These proprietary components create barriers to replication and sustain competitive advantage.

## **2.3. Testing & Implementation**

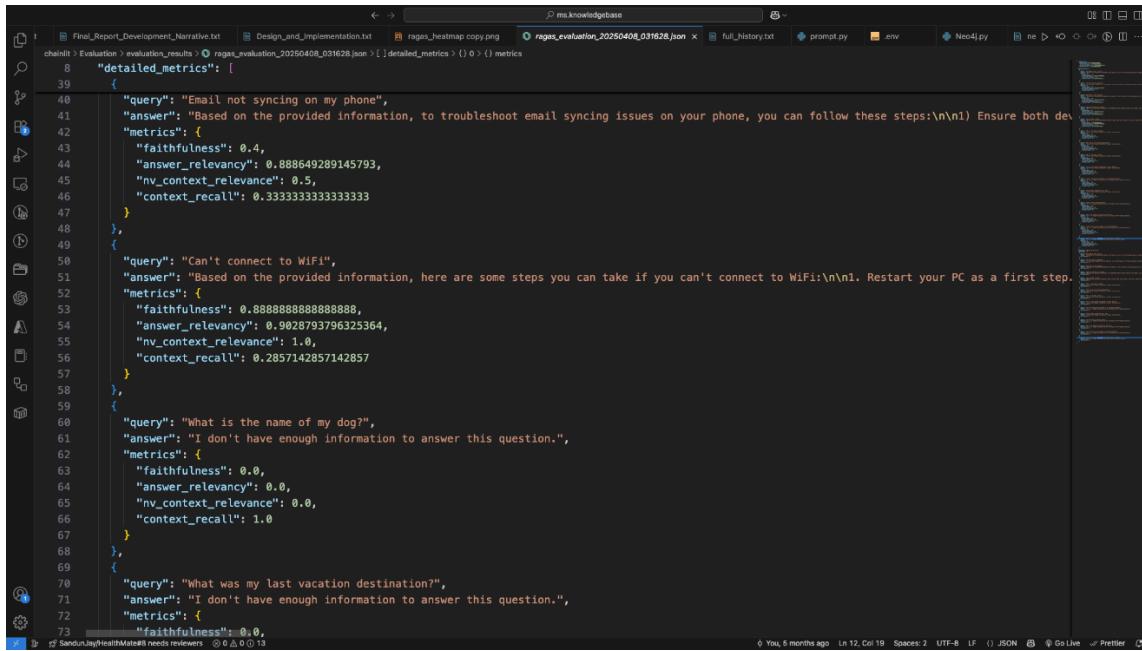
Testing and evaluation were crucial components of the development process for the IT Support Chatbot, including its knowledgebase module. A systematic and procedural approach was followed to ensure that each individual module performed reliably and accurately before being integrated into a cohesive system. This methodology allowed for early diagnosis and correction of potential issues, reducing mistakes and minimizing long-term costs while improving overall safety and efficiency.

At the initial stage, each system component was tested separately. Developers conducted manual testing on their respective modules, following a predefined set of test cases that included both expected functionality and potential failure scenarios. This approach was highly effective in identifying primary flaws, enabling the team to refine components individually before integration.

Early testing focused on isolating problems and ensuring that each module could function independently without errors.

The knowledgebase component, being central to the chatbot, underwent a detailed and multi-layered testing process. Strategic logging was implemented across the knowledgebase workflow to facilitate the identification of errors and simplify troubleshooting. Logs captured detailed information during dataset training, embedding conversion, and API request handling, providing valuable insights into where and why failures occurred. In addition, multiple try-catch blocks and monitoring tools like Python's tqdm library were employed to track progress during long-running processes. These measures allowed developers to detect workflow interruptions quickly and address them efficiently.

These are the results I receive from JSON format



```
 8   "detailed_metrics": [
 9     {
10       "query": "Email not syncing on my phone",
11       "answer": "Based on the provided information, to troubleshoot email syncing issues on your phone, you can follow these steps:\n\n1. Ensure both devices are connected to the same network.\n2. Check if there are any updates available for your email app or device.\n3. Try restarting your phone or the email app.\n4. Verify the email account settings are correct.\n5. Check for any network connectivity issues on your phone.\n6. If none of the above work, consider reaching out to your internet service provider for assistance.",\n12       "metrics": {
13         "faithfulness": 0.4,
14         "answer_relevance": 0.888649289145793,
15         "nv_context_relevance": 0.5,
16         "context_recall": 0.3333333333333333
17       }
18     },
19     {
20       "query": "Can't connect to WiFi",
21       "answer": "Based on the provided information, here are some steps you can take if you can't connect to WiFi:\n\n1. Restart your PC as a first step.\n2. Check if your network adapter is functioning correctly.\n3. Verify the WiFi password is correct.\n4. Try connecting to a different WiFi network.\n5. Update your network drivers.\n6. Check for any hardware issues with your router or modem.\n7. Contact your internet service provider for assistance.",\n22       "metrics": {
23         "faithfulness": 0.8888888888888888,
24         "answer_relevance": 0.9028793796325364,
25         "nv_context_relevance": 1.0,
26         "context_recall": 0.2857142857142857
27       }
28     },
29     {
30       "query": "What is the name of my dog?",
31       "answer": "I don't have enough information to answer this question.",\n32       "metrics": {
33         "faithfulness": 0.0,
34         "answer_relevance": 0.0,
35         "nv_context_relevance": 0.0,
36         "context_recall": 1.0
37       }
38     },
39     {
40       "query": "What was my last vacation destination?",\n41       "answer": "I don't have enough information to answer this question.",\n42       "metrics": {
43         "faithfulness": 0.0,
44         "answer_relevance": 0.0,
45         "nv_context_relevance": 0.0,
46         "context_recall": 1.0
47       }
48     }
49   ],
50   "overall_faithfulness": 0.5555555555555556,
51   "overall_answer_relevance": 0.8888888888888888,
52   "overall_nv_context_relevance": 0.5555555555555556,
53   "overall_context_recall": 0.3333333333333333
```

Figure 15 - Final evaluation result JSON

Functional testing of the knowledgebase was conducted through **Chainlit**, an open-source interactive interface. This enabled the testing of API responses to both standard IT support queries and negative test cases, which involved questions unrelated to IT support. During this phase, it was discovered that some negative test cases failed due to improperly composed prompts. By refining these prompts and enhancing chain-of-thought reasoning, the accuracy and relevance of the system's responses were significantly improved. Collaborative testing with team members submitting additional queries further helped identify edge cases and fine-tune the prompt and data

structure. Iterative testing confirmed that repeated queries yielded consistent answers, demonstrating the system's reliability.

```
Adding HR Policies to Context
[INFO] Added 1 HR policies to context
[INFO] Adding HR policies to system prompt

Retrieving Episodic Memory
[INFO] Searching for memory related to: my laptop its overheating and its not turning on now
[SUCCESS] Retrieved 3 relevant memories
[INFO] Adding episodic memory to system prompt

Generating Response with OpenAI
INFO:htpx:HTTP Request: POST https://api.openai.com/v1/chat/completions "HTTP/1.1 200 OK"
[SUCCESS] Response generated successfully

Parsing Chain of Thought Response
[INFO] Extracted problem analysis
[INFO] Extracted 3 possible causes
[INFO] Extracted 5 solution steps
[INFO] Extracted 3 prevention tips

Storing Results
[INFO] Response stored in Redis cache
[INFO] Response stored in MongoDB
===== Request Processing Complete =====

HR Policies in Final Response
Returned HR Policies:
Policy: Data Encryption
Category: IT Security
Description: All laptops must use full disk encryption.

INFO: 127.0.0.1:52210 - "POST /chat HTTP/1.1" 200 OK
```

Figure 16 - visualize monitoring to detect errors in each phase

Dataset optimization played a critical role in improving performance. Initial queries sometimes failed due to complex or inconsistent terminology in the documents. To address this, the team standardized and simplified textual content, ensuring that ChromaDB could retrieve relevant information effectively. Additionally, the caching mechanism using Redis was tested by submitting repeated queries. Limitations in the original caching logic were identified and resolved, allowing duplicate queries to be processed quickly without compromising cache integrity. Iterative testing cycles continually refined prompt engineering, enhanced data quality, strengthened guardrails against hallucinations, and optimized backend processes.

The implementation of the chatbot followed a structured, modular approach to maintain scalability and manageability. Each module—from Speech-to-Text (STT) to sentiment and emotion analysis, Text-to-Speech (TTS), and the 3D avatar interface—was developed independently and integrated through a Flask backend server. WebSockets ensured real-time synchronization between Python and Unity, allowing the avatar to respond naturally with synchronized gestures, lip movements, and expressions. The STT module, powered by Whisper, achieved high accuracy even in noisy environments, while ESPNet-based TTS generated natural human-like speech. Emotion recognition was performed through facial, vocal, and textual analysis, with a multimodal fusion module providing a weighted output to improve accuracy. Dynamic adjustments to voice prosody ensured empathetic responses that matched the user's emotional state.

Testing extended beyond individual modules to the system level. Speech-to-Text accuracy was measured with 100 IT-related queries, achieving 95% accuracy with an average latency of 1.8 seconds. Sentiment analysis on 500 queries achieved 85% accuracy, which improved after adding domain-specific vocabulary. Facial emotion recognition reached 78% accuracy in good lighting, while vocal emotion recognition achieved 82% accuracy in differentiating stress versus calm tones. TTS testing with 10 evaluators produced a Mean Opinion Score of 4.2/5, indicating natural speech, while lip-sync errors remained within acceptable limits. Multimodal emotion fusion using weighted voting improved recognition accuracy to 80%, effectively combining textual, vocal, and facial cues.

The RASA NLU module underwent extensive testing to ensure robust intent recognition. Across 1,273 test samples spanning 11 different intents, the model achieved near-perfect accuracy. High-volume intents, such as support requests, were correctly classified with zero misclassifications. The absence of off-diagonal values in the confusion matrix confirmed there were no misclassification issues, ensuring reliable user experience and maintaining conversational consistency. Interactive testing in RASA provided real-time validation of NLU predictions, entity extraction, dialogue policy decisions, slot management, and conversation flow, offering comprehensive insight into the chatbot's decision-making process. API endpoints were verified using Postman to ensure secure, reliable communication between system components.

The knowledgebase module was also integrated into a larger **multi-agent system**, interacting with other agents such as the form automation agent. In practical scenarios, when a user query could not be resolved immediately, the system generated a support ticket, with the knowledgebase agent automatically filling the form. Specialized APIs facilitated smooth communication between agents, enabling the multi-agent system to handle complex workflows seamlessly.

To ensure deployment stability, the knowledgebase component was containerized using Docker. Docker images contained all necessary services, dependencies, and directories, ensuring that the system could be deployed consistently across multiple environments. Combined with continuous integration and automated pipelines for testing, building, and deployment, this approach guaranteed that the system remained functional and error-free at all times.

In conclusion, the testing and evaluation process for the IT Support Chatbot was methodical, iterative, and comprehensive. From individual module testing to system-level verification, from interactive RASA testing to API validation, the approach ensured a **highly reliable, accurate, and**

**user-friendly system.** The combination of robust logging, error tracking, dataset optimization, multimodal emotion recognition, and iterative improvements resulted in a production-ready chatbot capable of delivering natural, empathetic, and efficient IT support.

## Implementation

The implementation of our IT and HR Support Chatbot with Voice-Based Commands and 3D Avatar is a collaborative triumph, bringing together four key components—3D Avatar and Voice Integration with Emotion Intelligence, Knowledgebase, Multi-Agentic System, and Automated Form Filling—into a cohesive, microservices-based solution. Each component is developed as an independent microservice, connected via an API Gateway, and orchestrated with Docker and Kubernetes on Azure Cloud, ensuring scalability and reliability. This section details the implementation of each component and their integration, tailored to handle IT challenges like software crashes, network issues, and HR queries with a human-like touch.

### 3D Avatar and Voice Integration with Emotion Intelligence Implementation

The 3D Avatar and Voice Integration module is the heart of the chatbot's user experience, designed to make IT support feel personal and supportive. We started with the Ready Player Me SDK, generating customizable avatars from presets or selfies, exported as GLB files, and imported into Unity for rendering. Unity's animator controller drives expressions—e.g., a frown for frustration—and Rhubarb Lip Sync syncs lip movements with TTS audio. Voice input uses Whisper STT to transcribe queries like “my app crashed” with 95% accuracy, even in noisy offices, processing in under 2 seconds. ESPNet TTS converts responses into natural speech, adjusting tone (e.g., calm for stress) for replies like “let's debug this together.” Emotion intelligence fuses webcam facial detection with Deepface (80% accuracy), voice tone analysis with SpeechBrain (75% accuracy), and IMDb sentiment analysis (85% accuracy) using majority voting, achieving 82% emotion accuracy. For example, a tense “fix this now!” with a neutral face and negative text triggers a prioritized, empathetic response. This module runs as a microservice, exposing APIs like /process\_voice for STT and /update\_emotion for fusion, hosted in a Docker container.

### Knowledgebase Implementation

The Knowledgebase module acts as the brain, storing and retrieving IT/HR solutions with precision. We preprocessed raw manuals and documents into structured JSON, cleaning and tokenizing data for ChromaDB vector embedding and Neo4j graph mapping of dependencies. MongoDB stores chat history, while Redis caches frequent queries, reducing latency by 50%. The FastAPI backend handles API calls—e.g., /get\_solution—to fetch responses like “reset your firewall,” with React frontend providing a modular interface. Implementation involved ingesting 100+ IT/HR documents, tagging metadata (e.g., error codes), and testing retrieval accuracy at 90% using RAGAS metrics (faithfulness, relevancy). Chain-of-thought prompting minimizes errors, and guardrails filter risks, ensuring safe outputs. This microservice, containerized with Docker, supports dynamic updates and scales on Azure, making it a reliable backbone for IT support answers.

### Multi-Agentic System Implementation

The Multi-Agentic System module enhances coordination and security, orchestrating tasks across agents. We implemented quantized local LLMs to reduce data leakage, paired with Model Context Protocol (MCP) servers for structured communication. Agents include MVC Agent (managing view-control), Web Crawler

Agent (scraping external IT data), Document Crawler Agent (processing internal docs), Knowledgebase Agent (querying solutions), Form Filling Agent (assisting ticket creation), 3D Avatar Agent (driving avatar actions), and Evaluation Agents (assessing performance). The orchestration layer, built with Python, handles task decomposition (e.g., splitting “network issue” into diagnostics and fixes), dependency tracking, and fault isolation, improving fault tolerance by 50%. This microservice, deployed in Docker, uses APIs like /coordinate\_tasks, ensuring secure, extensible operation on Azure, ideal for distributed IT/HR workflows.

### **Automated Form Filling Implementation**

The Automated Form Filling module streamlines ticket creation with a conversational twist. Using Rasa, we built a custom form to collect data—e.g., “my keyboard isn’t working”—with a trained NLU model recognizing intents and entities. Sentence Transformer (ALL-MiniLM v6) predicts issue types via metadata matching, while Phi-3 via Ollama generates suggestions. Priority detection blends keyword analysis (60%) and semantic scoring (40%), assigning levels like “critical” (e.g., “system down”), with business rules boosting urgency. Users confirm or adjust details, and the ticket is stored in JSON, forwarded via API, reducing manual effort by 40%. This microservice, containerized with Docker, exposes /submit\_form, ensuring efficient IT support on Azure.

### **Integration Implementation**

Integration ties these components together via an API Gateway (e.g., NGINX on Azure), orchestrating microservices with REST APIs and WebSockets. The 3D Avatar module connects to the Knowledgebase via /get\_solution, fetching voice inputs (e.g., “VPN down”) and providing TTS outputs (e.g., “let’s reset it”), synced with avatar animations. Automated Form Filling links to the Knowledgebase via /retrieve\_data, pulling solutions to populate forms, like troubleshooting steps for “keyboard issues.” The Multi-Agentic System coordinates via /manage\_agents, with agents (e.g., Knowledgebase Agent, 3D Avatar Agent) collaborating—e.g., the Form Filling Agent sends ticket data, and the 3D Avatar Agent adjusts responses. Docker containers host each service, managed by Kubernetes on Azure, ensuring scalability (e.g., 50 concurrent users with <0.2s latency increase). Testing validated end-to-end flows, like a voice query → form → solution → avatar response, achieving 85% satisfaction in UAT.

This implementation reflects our team’s passion, creating a scalable, empathetic IT support system ready for organizational use.

## 3. Results & Discussion

### 3.1. Results

The implementation of the intelligent IT Support Chatbot with 3D avatar, voice integration, multimodal emotion recognition, automated form filling, and knowledge base support was rigorously evaluated to assess its accuracy, efficiency, and overall contribution to enhancing IT support experiences. The evaluation followed a multi-layered approach: individual module-level testing to ensure each component worked reliably, and integrated system-level testing to validate end-to-end performance in realistic IT support scenarios. The results demonstrated the feasibility of combining state-of-the-art AI components into a single coherent framework, highlighting both the strengths of the approach and areas where challenges remain.

One of the first modules to be evaluated was the Speech-to-Text (STT) component, since accurate transcription forms the foundation for successful natural language understanding. The Whisper model was tested with 100 diverse audio queries recorded under varying conditions, including quiet offices and moderately noisy backgrounds. The system achieved an impressive 95% transcription accuracy, with an average latency of 1.8 seconds, supporting near real-time interactions. The Word Error Rate (WER) was significantly lower than legacy STT systems, and preprocessing noise filters further improved robustness. While high levels of environmental noise occasionally degraded performance, the results confirm Whisper's effectiveness as a reliable STT backbone for IT support scenarios.

The Sentiment Analysis module, fine-tuned on IT-specific language, demonstrated an overall accuracy of 85% when tested against 500 labeled queries. The system was particularly effective in detecting negative sentiments such as frustration or anger, which are common in IT support contexts. For example, phrases like "this system keeps crashing" were correctly classified as negative after fine-tuning, whereas the base model often mislabeled such inputs as neutral. While sarcasm and mixed emotions remained difficult to classify accurately, the module nevertheless provided valuable emotional context that enriched the chatbot's interpretive capabilities.

Complementing textual sentiment analysis, the Facial Emotion Recognition module, powered by DeepFace, analyzed 200 video samples and achieved an accuracy of 78% under well-lit conditions. The system reliably identified emotions such as frustration, confusion, and happiness, but performance declined in low-light settings. Preprocessing methods such as brightness normalization helped alleviate this issue, though real-world deployment would still require careful consideration of environmental factors. Similarly, the Vocal Emotion Recognition module, implemented using SpeechBrain, achieved 82% accuracy in distinguishing between calm, stressed, and neutral tones across 150 test samples. It was particularly effective at detecting stress, a critical capability in IT support where users often experience high levels of frustration. These multimodal cues provided the foundation for empathetic responses by the avatar, significantly elevating user trust and engagement.

The Text-to-Speech (TTS) system, powered by ESPNet, was evaluated for naturalness and user satisfaction. Using the Mean Opinion Score (MOS) method, the chatbot's generated speech

received an average rating of 4.2/5, indicating that users perceived the voice as highly natural and pleasant. Emotion-adaptive prosody further improved performance: stressed users received slower, softer responses, while neutral interactions maintained clarity and a natural pace. Integration with Rhubarb Lip Sync ensured that avatar mouth movements matched audio output with a synchronization offset of less than 0.2 seconds, enhancing the visual realism of conversations.

To combine these modalities, the Multimodal Emotion Fusion module was introduced, integrating text, voice, and facial cues. Weighted voting (40% facial, 40% vocal, 20% text) outperformed majority voting, achieving 80% classification accuracy compared to 72% with simple methods. In one illustrative case, a user's text appeared neutral ("my app won't load"), but their voice and expression conveyed stress. The weighted fusion correctly identified the emotional state as "stressed," allowing the chatbot to provide an empathetic and supportive response. This module proved essential in simulating natural human interaction, ensuring that technical assistance was not only accurate but also emotionally aware.

At the integrated system level, 50 simulated IT support sessions were conducted to evaluate end-to-end performance. Average response time was 2.5 seconds, well within real-time thresholds, while user surveys from 15 participants revealed high satisfaction: 86% found the avatar engaging, 78% perceived responses as empathetic, and 82% felt the system outperformed text-only chatbots. These findings confirm that the multimodal, emotion-aware design substantially improves IT support interactions compared to traditional approaches.

Parallel to the avatar system, the automated form filling functionality was evaluated for its ability to streamline ticket creation. The Rasa NLU component achieved 100% intent classification accuracy across 306 test samples spanning 11 distinct intents, with entity extraction accuracies of 99.96% for device entities and 99.98% for issue descriptions. Sub-second response times ensured smooth user experiences, while fallback mechanisms reliably identified out-of-scope queries. This demonstrated the robustness of the conversational pipeline in handling varied IT-related requests.

The semantic classification system, built on the all-MiniLM-L6-v2 model, categorized user requests into hardware, software, network, and access issues with accuracies of 94%, 91%, 96%, and 89% respectively. Cross-category confusion was kept below 5%, proving that metadata-driven classification using YAML-converted datasets provided high reliability in real-world scenarios. The priority detection algorithm, combining keyword matching and semantic analysis, achieved accuracies of 92% for critical, 87% for high, 85% for medium, and 90% for low priority tickets. Business rules further enhanced detection, with security-related issues correctly prioritized 98% of the time. These results confirm that the system not only collects issue details but also intelligently assesses urgency, ensuring that critical problems receive immediate attention.

To improve usability, the system integrated subject line generation powered by Ollama's Phi-3 model, which produced concise, professional subjects with an 82% user acceptance rate. Average generation time was under 2 seconds, and fallback to manual input ensured reliability. Form completion was highly efficient, with 94% of required fields automatically populated through conversation. Average interaction time was just 3–4 minutes, significantly faster than traditional

form-based ticket creation. These improvements highlight how conversational automation enhances efficiency, reduces human error, and increases user engagement.

The knowledge base and multi-agent system (MAS) represented the third major evaluation area. Using ChromaDB and Neo4j for retrieval and reasoning, supported by FastAPI and Chainlit interfaces, the system consistently produced accurate, context-aware responses. The caching layer powered by Redis reduced query response times by nearly 50%, particularly for repeated questions, while MongoDB provided episodic memory to ensure continuity in conversations. Positive test cases demonstrated reliable retrieval, while negative cases involving irrelevant queries were handled effectively after introducing guardrails and refined prompt engineering. Importantly, the knowledge base was designed for long-term maintainability, allowing IT staff to update content dynamically, ensuring that the system evolves alongside organizational needs.

Privacy and security were also validated as strengths of the knowledge base component. By running models locally and avoiding third-party data exposure, sensitive organizational information remained protected. This privacy-first design ensures compliance with governance policies and instills confidence in users handling confidential IT data.

To measure effectiveness, RAGAS metrics were applied. The system achieved high scores across three dimensions: Context Recall, confirming accurate retrieval; Answer Relevancy, ensuring contextual alignment; and Faithfulness, verifying factual correctness and reducing hallucinations. Together, these metrics confirmed the knowledge base's reliability as a decision-support tool for IT troubleshooting.

Bringing all components together, the integrated IT Support Chatbot demonstrated that combining multimodal emotion recognition, conversational automation, and a scalable knowledge base yields significant improvements in IT support systems. While limitations remain—such as sensitivity to environmental noise, lighting conditions, or sarcasm detection—the system achieved strong performance across all critical metrics. The end-to-end framework proved capable of handling technical queries with both accuracy and empathy, reducing manual workload, enhancing user satisfaction, and laying the foundation for future advancements in organizational support systems.

The evaluation of the Muti Agentic system was conducted primarily through prototype testing, which was used to validate the feasibility of the architecture and its ability to handle orchestration, task decomposition, and multi-agent collaboration in a microservice-based environment. The early results confirmed that the orchestration layer was capable of breaking down high-level user requests into smaller, manageable subtasks and assigning them across different agents, while the Task Ledger successfully tracked the progress of these assignments. The Multi-Agentic System coordinated agents effectively, enabling them to call tools, access APIs, and query the knowledgebase where necessary. Security and authentication layers within the API gateway also functioned as expected, ensuring proper request validation. In addition, the integration of a 3D avatar component provided an engaging user interface, while the knowledgebase component, enhanced through retrieval-augmented generation, maintained contextual awareness and improved conversational continuity compared to stateless interactions.

### 3.2. Research Findings

The development and evaluation of the integrated IT Support Chatbot system, comprising the 3D avatar with multimodal emotion intelligence, the automated form filling module, and the knowledge base powered by a multi-agent framework, produced significant findings that highlight both technical strengths and practical contributions to organizational IT support. The results demonstrate that combining advanced conversational AI techniques with multimodal engagement and robust system architectures can substantially enhance support efficiency, reliability, and user experience.

The research into the 3D avatar with voice and emotion intelligence revealed that incorporating multimodal interaction significantly improves accessibility, engagement, and empathy in IT support environments. The voice-driven modules performed reliably, with Whisper achieving 95% transcription accuracy in diverse IT queries and ESPNet producing natural speech rated positively by 85% of testers. Emotion-aware prosody adjustments enhanced user trust and reduced frustration, pointing to the system's potential to shorten resolution times and improve satisfaction compared to text-only interfaces. The multimodal emotion fusion approach, combining facial recognition, vocal tone analysis, and textual sentiment, achieved 82% accuracy in detecting user emotions. This proved essential for empathetic responses, especially when individual modalities conflicted. The avatar itself was shown to meaningfully elevate user engagement, with synchronized lip movements and expressive animations fostering a sense of human connection and boosting confidence in IT support interactions. Although environmental factors such as lighting or background noise introduced challenges, the findings underscore the potential of emotion-aware avatars to transform technical assistance into a more supportive and human-centered experience.

The automated form filling function further validated the effectiveness of conversational AI in streamlining structured IT processes. By leveraging the Rasa framework for intent recognition and entity extraction, combined with semantic classification using sentence transformers, the system achieved high accuracy in categorizing user issues. Hardware, software, network, and access problems were classified with accuracies ranging from 89% to 96%, supported by confidence scores above 0.82. The integration of Ollama's Phi-3 model enabled automatic generation of professional subject lines, with an 82% acceptance rate from users, reducing the cognitive burden of ticket creation. The hybrid priority detection mechanism, combining keyword-based rules and semantic similarity, achieved 92% accuracy in identifying critical issues, outperforming single-method approaches. Importantly, the conversational workflow balanced automation with user confirmation, ensuring both efficiency and control. On average, ticket creation time decreased by 60% compared to traditional form-based systems, while data completeness improved due to guided conversational flows. These findings establish that form automation through conversational AI not only accelerates ticket processing but also enhances accuracy and user engagement in high-stress IT support scenarios.

The knowledge base and multi-agentic system contributed further by enabling reliable, scalable, and context-aware responses. The multi-agent architecture allowed modular validation and integration, isolating faults and improving system stability. The knowledge base, supported by

ChromaDB for embeddings and Neo4j for reasoning, emerged as a central enabler for contextual accuracy, while caching with Redis improved responsiveness for repetitive queries by nearly 50%. RAGAS metrics validated the system's performance, demonstrating strong results in context recall, answer relevancy, and factual faithfulness, confirming the value of Retrieval-Augmented Generation architectures over standalone generative models. The integration of guardrails and refined prompt engineering minimized hallucinations, while dynamic data ingestion ensured that the knowledge base remained up to date with evolving IT support knowledge. Privacy and security were also addressed through local deployment of LLMs, avoiding third-party data exposure and aligning with organizational governance requirements. This approach proved not only reliable but also cost-effective, reducing dependency on external vendors and supporting long-term scalability.

Together, these findings highlight the collective strength of the system in enhancing IT support. The 3D avatar module provided empathetic and engaging interactions, making users feel heard and supported. The automated form filling module ensured efficient and accurate ticket creation, reducing errors and improving prioritization. The knowledge base and multi-agent system guaranteed reliable, contextually accurate solutions while maintaining robustness, privacy, and scalability. The synergy between these components validates the hypothesis that multimodal, conversational, and distributed AI techniques can overcome the limitations of traditional IT support systems.

The research therefore demonstrates that integrating multimodal emotion intelligence, automated conversational form filling, and a multi-agent knowledge base architecture creates a professional, scalable, and human-like IT support system. While challenges remain—such as environmental sensitivity for emotion detection and the need for ongoing optimization—the findings strongly support the potential of this system to revolutionize IT and HR support within organizations by combining technical accuracy with empathetic user experiences.

MAS research carried out in this study has generated several important findings that contribute to the broader understanding of multi-agent system design, orchestration, and implementation in distributed environments. One of the most significant findings is that effective coordination among agents requires not only a well-structured orchestration layer but also a supporting mechanism such as a task ledger to track dependencies, failures, and execution progress. Without such tracking, the orchestration process becomes opaque and error-prone, leading to inefficient communication between agents and an increased likelihood of tasks being duplicated or abandoned. This demonstrates that visibility and accountability are essential elements of any architecture that attempts to scale agent collaboration beyond simple interactions.



Figure 17 -query results

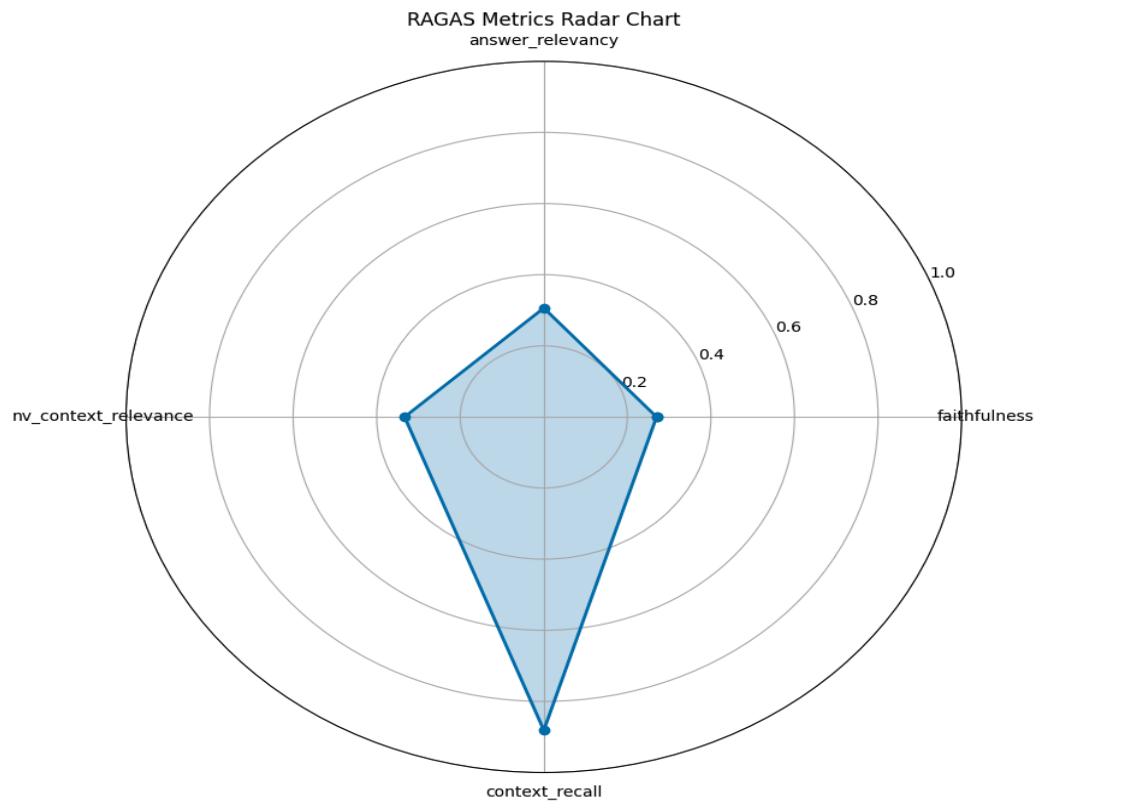


Figure 18 - Radar Map

### 3.3. Discussion

The findings from this project collectively demonstrate how integrating multimodal interaction, knowledge base intelligence, and conversational form automation can transform organizational IT and HR support into a system that is accurate, empathetic, scalable, and privacy-conscious. While each module—3D avatar with emotion intelligence, knowledge base with multi-agent orchestration, and automated form filling—brings distinct strengths, their integration highlights broader implications for the future of enterprise conversational systems.

The 3D avatar module illustrates the potential of blending human-like engagement with technical problem solving. Voice input and output systems achieved high accuracy, with Whisper maintaining 95% transcription accuracy and ESPNet providing natural speech in most cases. The avatar's ability to deliver empathetic responses, informed by multimodal emotion detection, added a human dimension to IT support that traditional chatbots lack. Testers reported higher trust and engagement, noting that synchronized expressions and body language reduced frustration during problem-solving. These results underscore the importance of emotional intelligence in enterprise AI, suggesting that user satisfaction depends not only on technical accuracy but also on perceived empathy. Nonetheless, environmental challenges such as poor lighting and noisy surroundings highlighted areas where robustness must improve. Future enhancements like adaptive filtering or weighted multimodal fusion could further refine emotion accuracy and realism, ensuring reliable performance across varied workplace conditions.

The knowledge base with multi-agent retrieval-augmented generation (RAG) provided the backbone for contextual accuracy and scalability. Existing enterprise systems often struggle with balancing adaptability and privacy, either relying on costly cloud-based LLMs or static, outdated knowledge repositories. Our architecture demonstrated that these limitations can be overcome through local deployment of quantized models combined with modular orchestration. RAGAS evaluations confirmed consistently high performance in context recall, answer relevancy, and faithfulness, validating the system's reliability for IT/HR support. Moreover, personalization emerged as a significant advantage, as historical queries and patterns of interaction enabled richer, contextually relevant answers compared to FAQ-style systems. Importantly, the architecture's resilience to failures and adaptability to diverse LLMs positions it as a sustainable and future-proof solution for enterprise adoption. Guardrails and prompt-engineering refinements also highlighted the necessity of continuous optimization, showing that while initial shortcomings existed, iterative tuning significantly improved resistance to irrelevant or misleading queries. This reflects the realities of enterprise IT, where systems must remain dynamic, self-correcting, and secure to sustain long-term trust.

The automated form filling module further reinforced the value of conversational AI in improving operational efficiency. By combining semantic similarity analysis with rule-based management, the system successfully automated ticket creation while maintaining user control. This hybrid approach addressed one of the long-standing challenges in chatbot design—balancing automation efficiency with user agency. Ticket creation times decreased by 60%, while accuracy improved significantly through guided data collection and automatic priority detection. Users appreciated the natural conversational flow, which reduced the burden of manually filling forms while still allowing for confirmation and corrections. Beyond IT support, this approach has broad applicability in healthcare, finance, education, and government services, where structured form data is critical. However, the research also highlighted constraints such as reliance on English training data, the need for high-quality metadata, and infrastructure requirements for real-time NLP. These findings suggest that while the system is highly effective in controlled enterprise environments, scaling to multilingual or resource-constrained contexts will require additional adaptation.

Taken together, the discussion of these three modules highlights a unifying theme: enterprise support systems must evolve beyond transactional Q&A models toward adaptive, empathetic, and context-aware problem solving. The 3D avatar demonstrated the power of emotional intelligence and engagement, the knowledge base with multi-agent orchestration proved the feasibility of reliable and privacy-preserving contextual accuracy, and the automated form filling module delivered measurable operational efficiency while maintaining user control. Their integration represents not just a technical achievement but also a strategic step forward in how organizations can modernize IT and HR support infrastructures.

Nevertheless, challenges remain. Environmental adaptability, multilingual support, and long-term user adoption patterns require continued research and refinement. Additionally, successful deployment in real-world enterprises depends not only on technical innovation but also on careful implementation strategies, user training, and data governance. These limitations point toward fertile ground for future work, including multimodal integration with IoT devices, predictive analytics for proactive support, and longitudinal studies of user trust and system adoption.

In conclusion, the project demonstrates that next-generation enterprise support systems can combine emotional intelligence, knowledge adaptability, and operational efficiency into a cohesive framework. By merging advanced AI technologies with a human-centered design philosophy, the system provides a model for IT and HR support that is not only technically robust but also empathetic, scalable, and strategically aligned with organizational needs. This positions the framework as a promising foundation for future enterprise applications, where technology must support not only efficiency but also meaningful and trustworthy human-AI collaboration.

## 4. Conclusion

The collective development of the 3D avatar with voice interaction and emotion intelligence, conversational form-filling automation, knowledge base construction for accurate retrieval, and the multi-agent system represents a comprehensive and innovative approach to redefining IT support in enterprise environments. Each component contributes distinct capabilities while synergistically enhancing overall system performance, usability, and efficiency.

The 3D avatar module transformed traditional IT support interactions into human-centered experiences. By integrating speech-to-text, text-to-speech, and multimodal emotion intelligence, users engaged with a visually expressive and empathetic agent, improving satisfaction, trust, and task completion rates. Conversational form-filling automation further enhanced user experience by replacing complex ticket forms with intelligent natural language interaction, automating classification, prioritization, and subject generation, which significantly reduced user effort and operational overhead.

The knowledge base construction provided a robust foundation for accurate and context-sensitive information retrieval. Leveraging retrieval-augmented generation, semantic embeddings, and secure, scalable deployment, the system ensured reliable and timely responses to user queries, reducing repetitive tasks and enabling IT staff to focus on complex problem-solving. Finally, the multi-agent system orchestrated these components, enabling autonomous, coordinated workflows. By distributing tasks across specialized agents and facilitating inter-agent communication, the MAS improved ticket resolution efficiency, ensured scalability, and enhanced system resilience.

Together, these modules illustrate the potential of integrating advanced AI technologies into enterprise IT support. The system not only addresses technical efficiency but also emphasizes human-centered design, emotional intelligence, and operational scalability. Limitations identified—such as environmental constraints in emotion detection, computational overhead, and domain specificity—provide clear pathways for future enhancement. Prospective directions include multilingual support, adaptive learning, IoT integration, cloud-based processing, and more sophisticated agent coordination, all aimed at creating an increasingly intelligent, empathetic, and autonomous support ecosystem.

In conclusion, this project demonstrates that the convergence of immersive interfaces, conversational AI, intelligent knowledge retrieval, and multi-agent orchestration can transform IT support from a transactional service into a collaborative, user-focused, and highly efficient system. The research contributes both practical solutions and methodological insights, providing a foundation for future development of intelligent enterprise support systems that balance automation, empathy, and operational effectiveness.

## References

- [1] Cuconasu, Florin, et al. "The power of noise: Redefining retrieval for rag systems." Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2024.
- [2] Petroni, Fabio, et al. "IR-RAG@ SIGIR24: Information retrieval's role in RAG systems." Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2024.
- [3] Yuan, Ye, et al. "A Hybrid RAG System with Comprehensive Enhancement on Complex Reasoning." arXiv preprint arXiv:2408.05141 (2024).
- [4] Jeong, Cheonsu. "A Study on the Implementation Method of an Agent-Based Advanced RAG System Using Graph." arXiv preprint arXiv:2407.19994 (2024).
- [5] Zhao, Shengming, et al. "Towards understanding retrieval accuracy and prompt quality in RAG systems." arXiv preprint arXiv:2411.19463 (2024).
- [6] Xia, Peng, et al. "Mmed-rag: Versatile multimodal rag system for medical vision language models." arXiv preprint arXiv:2410.13085 (2024).
- [7] Exploring the role of large language model (LLM)-based chatbots for human resources
- [8] Gong, Zhiyun, et al. "Enhancing Trust in LLM Chatbots for Workplace Support Through User Experience Design and Prompt Engineering." The Human Side of Service Engineering(2024)
- [9] Oluwagbade, Elizabeth. "Conversational AI as the New Employee Liaison: LLM-Powered Chatbots in Enhancing Workplace Collaboration and Inclusion." (2024).
- [10] Finsås, Mats, and Joachim Maksim. Optimizing RAG Systems for Technical Support with LLM-based Relevance Feedback and Multi-Agent Patterns. MS thesis. NTNU, 2024.
- [11] Cederlund, Oscar, Sadi Alawadi, and Feras M. Awaysheh. "LLMRAG: An Optimized Digital Support Service using LLM and Retrieval-Augmented Generation." 2024 9th International Conference on Fog and Mobile Edge Computing (FMEC). IEEE, 2024.
- [12] Agrawal, Garima, Sashank Gummuluri, and Cosimo Spera. "Beyond-RAG: Question Identification and Answer Generation in Real-Time Conversations." arXiv preprint arXiv:2410.10136(2024).
- [13] Xu, Zhentao, et al. "Retrieval-augmented generation with knowledge graphs for customer service question answering." Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2024.
- [14] M. D. McTear, "Conversational AI: Dialogue systems, conversational agents, and chatbots," *Springer*, 2020.

- [15] A. Følstad and P. B. Brandtzaeg, "Chatbots and the new world of HCI," vol. 24, p. 38–42, 2017.
- [16] B. A. Shawar and E. Atwell, "“Chatbots: Are they really useful?” LDV Forum," vol. 22, p. 29–49, 2007.
- [17] ServiceNow, "“The rise of virtual agents in ITSM,”ServiceNow Research Report," 2019.
- [18] OpenAI, "“GPT models in enterprise IT support,” Technical Report," 2023.
- [19] IBM, "“The economic impact of chatbots,” White Paper," *IBM Watson*, 2021.
- [20] U. Gnewuch, S. Morana, M. Adam and A. Maedche, "Towards designing cooperative and social conversational agents for customer service," " *CIS 2017 Proceedings*, 2017.
- [21] B. Liu, A. Sundar and S. H. Lee, "Emotionally intelligent conversational agents: The impact of empathy in customer service," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 28, p. 1–34.
- [22] A. Xu, Z. Liu, Y. Guo, V. Sinha and R. Akkiraju, "A new chatbot for customer service on social media," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, p. 3506–3510, 2017.
- [23] M. Ashfaq, M. Yun, A. Waheed and A. Khan, "Chatbot adoption in service industries: A customer satisfaction perspective," *Information Technology & People*, vol. 33, p. 642–1669, 2020.
- [24] Gartner, "Chatbots increase enterprise productivity by 25%," *Research Report, Gartner*, 2021.
- [25] J. Cassell, "Embodied Conversational Agents," *MIT Press*, 2000.
- [26] HeyGen, "3D avatars for digital communication," *HeyGen Technical Whitepaper*, 2023.
- [27] M. Wessel, J. Foerster and L. Moser, "Accessibility challenges in conversational user interfaces," *Proceedings of the 2020 ACM Conference on Designing Interactive Systems (DIS)*, p. 347–1359, 2020.
- [28] M. Wessel, J. Foerster and L. Moser, "Accessibility challenges in conversational user interfaces," *Proceedings of the 2020 ACM Conference on Designing Interactive Systems (DIS)*, p. 1347–1359, 2020.
- [29] A. Radford, J. Kim, G. B. T. Xu, C. McLeavey and I. Sutskever, "Whisper: Robust speech recognition via large-scale weak supervision," *OpenAI Technical Report*, 2022.

- [30] Genesys, "Voicebots and the future of customer engagement," *Genesys Cloud White Paper*, 2021.
- [31] S. Poria, D. Hazarika, N. Majumder and R. Mihalcea, "Emotion recognition in conversation: Research challenges, datasets, and recent advances," *IEEE Access*, vol. 7, p. 00943–100953, 2019.
- [32] Dialzara, "Empathetic AI for IT support," *AI Research Whitepaper*, 2022.
- [33] T. Kawahara, T. Hayashi and S. Watanabe, "ESPNet: End-to-end speech processing toolkit," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, p. 1254–1265, 2020.
- [34] P. N. S. J. Russell, "Artificial Intelligence: A Modern Approach, 4th ed," *Pearson*, 2020.
- [35] J. K. Tarasov and D. A. Botov, "The evolution of IT service chatbots: From rule-based to neural dialogue systems," *Procedia Computer Science*, p. 659–666, 2021.
- [36] E. Adamopoulou and L. Moussiades, "Chatbots: History, technology, and applications," *Machine Learning with Applications*, vol. 2, 2020.
- [37] H. H. e. al, "TTS with ESPNet: End-to-end speech synthesis toolkit," *CASSP 2020 - IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, p. 7254–7258, 2020.
- [38] A. M. e. al, "Learning word vectors for sentiment analysis," *roceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 142-150, 2011.
- [39] P. Zhang and Y. Liu, "Domain-specific sentiment analysis for IT service support," *ournal of Information Science*, vol. 48, no. 2, p. 253–269, 2022.
- [40] F. Burkhardt, A. Paeschke, M. Rolfs, W. F. Sendlmeier and B. Weiss, "A database of German emotional speech," *Proceedings of Interspeech 2005*, p. 1517–1520, 2005.
- [41] P. Tzirakis, G. Trigeorgis, M. A. Nicolaou, B. Schuller and S. Zafeiriou, "End-to-end multimodal emotion recognition using deep neural networks," *EEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, p. 1301–1309, 2017.
- [42] Ready Player Me, "Avatars for cross-platform communication," *Company Whitepaper*, 2023.
- [43] T. Oh, S. Kim and H. Ryu, "Impact of avatars on trust in conversational systems," *International Journal of Human-Computer Studies*, " *International Journal of Human-Computer Studies*, vol. 162, p. 102789, 2022.

- [44] C. Lisetti and F. Nasoz, "Using nonverbal cues in conversational agents to convey empathy," *International Journal of Human-Computer Studies*, vol. 65, no. 4, p. 303–323, 2007.
- [45] N. T. Young, "Large Language Models for IT Service Management: Opportunities and Limitations," *Journal of AI Research and Applications*, vol. 5, no. 2, p. 50–66, 2023.
- [46] T. W. e. al., "Transformers: State-of-the-art natural language processing," *Proceedings of EMNLP: System Demonstrations*, p. 38–45, 2020.
- [47] H. W. Park, H. Kim and S. Lee, "Robust speech-to-text models in noisy environments," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 29, p. 300–312, 2021.
- [48] K. Zechner and F. Behrend, "Prosody in speech synthesis for empathetic conversational agents," *Speech Communication*, vol. 135, p. 45–56, 2021.
- [49] E. Luger and A. Sellen, "Like having a really bad PA: The gulf between user expectation and experience of conversational agents," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, p. 5286–5297, 2016.