**The Bucharest University of Economic Studies**

**Faculty of Business Administration**

**(in Foreign Languages)**

**Master’s Thesis**

**AI Models for Improving and Optimizing Customer Support Efficiency**

Scientific coordinator: Student:

Conf. Univ. Dr. BUȘU Mihail PANAIT Tudor-Alexandru

Bucharest

2025

Table of contents

[List of tables 4](#_Toc201159848)

[List of figures 4](#_Toc201159849)

[Abstract 5](#_Toc201159850)

[Introduction 1](#_Toc201159851)

[1. Literature review 2](#_Toc201159852)

[1.1. Evolution of Customer Support 4](#_Toc201159853)

[1.2. Evolution of Customer Support with AI Integration 6](#_Toc201159854)

[1.3. Current AI Models and Tools in Use 10](#_Toc201159855)

[2. Methodology 13](#_Toc201159856)

[3. Case Studies 15](#_Toc201159857)

[3.1. AI Powered Chatbots 15](#_Toc201159858)

[3.2. AI Enhanced Knowledge Management 18](#_Toc201159859)

[3.3. AI for Call Analysis and Voice Support 20](#_Toc201159860)

[3.4. AI in Returns and Refunds Automation 21](#_Toc201159861)

[4. Hands-On AI with Prototypes and Practical Applications 24](#_Toc201159862)

[4.1. Tech Stack 24](#_Toc201159863)

[4.2. Prototypes Presentation 27](#_Toc201159864)

[4.2.1. Sentiment Analysis 27](#_Toc201159865)

[4.2.2. Intent Analysis 28](#_Toc201159866)

[4.2.3. Rule-Based Chatbot 29](#_Toc201159867)

[4.2.4. Keyword-Based Chatbot 30](#_Toc201159868)

[4.2.5. Embeddings 31](#_Toc201159869)

[4.2.6. Knowledge Management with Search 33](#_Toc201159870)

[4.2.7. Conversational AI Chatbot 34](#_Toc201159871)

[4.2.8. Voice AI Chatbot 35](#_Toc201159872)

[4.2.9. Speech AI Assist 37](#_Toc201159873)

[4.2.10. ReACT & MRKL AI Agent 38](#_Toc201159874)

[4.3. Overview on the Prototypes Results 41](#_Toc201159875)

[5. Discussion 44](#_Toc201159876)

[5.1. What are the most common and high impact use cases of AI in customer support? 44](#_Toc201159877)

[5.2. How are different AI models (e.g., LLMs, embeddings, speech recognition) integrated into actual support workflows? 44](#_Toc201159878)

[5.3. What practical challenges and benefits have been observed in implementations? 45](#_Toc201159879)

[5.4. How can modular tools be assembled to support intelligent, multi-modal, and context-aware support systems? 45](#_Toc201159880)

[5.5. How can support operations progress from simple automation to more sophisticated agentic AI systems? 46](#_Toc201159881)

[6. Conclusion 47](#_Toc201159882)

[Bibliography 49](#_Toc201159883)

[Appendixes 59](#_Toc201159884)

[Appendix 1 – Sentiment Analysis Application Source Code 59](#_Toc201159885)

[Appendix 2 – Intent Analysis Application Source Code 61](#_Toc201159886)

[Appendix 3 – Rule-Based Chatbot Application Source Code 63](#_Toc201159887)

[Appendix 4 – Keyword-Based Chatbot Application Source Code 68](#_Toc201159888)

[Appendix 5 – Embeddings Application Source Code 73](#_Toc201159889)

[Appendix 6 – Knowledge Management with Search Application Source Code 77](#_Toc201159890)

[Appendix 7 – Conversational AI Chatbot Application Source Code 80](#_Toc201159891)

[Appendix 8 – Voice AI Chatbot Application Source Code 84](#_Toc201159892)

[Appendix 9 – Speech AI Assist Application Source Code 87](#_Toc201159893)

[Appendix 10 – ReACT & MRKL Application Source Code 91](#_Toc201159894)

# List of tables

[Table 1 - ReACT Agent Reasoning and Execution Flow; Source: own research 39](#_Toc201159895)

[Table 2 - ReACT & MRKL Agent Behavior in a Multi-Prompted User Query; Source: own research 41](#_Toc201159896)

[Table 3 - Summary of AI Prototypes for Customer Support; Source: own research 42](#_Toc201159897)

# List of figures

[Figure 1 - Graph showing the worldwide Google Search popularity of the term "AI"; Source: Google Trends 3](#_Toc201159898)

[Figure 2 - The Evolution of Customer Support; Source: own research 6](#_Toc201159899)

[Figure 3 - Example of a Decision Tree for a Rule-Based Chatbot; Source: own research 7](#_Toc201159900)

[Figure 4 - Timeline of Chatbot and AI Evolution in Customer Support; Source: own research 9](#_Toc201159901)

[Figure 5 - Project's folder structure; Source: own research 26](#_Toc201159902)

[Figure 6 - Application Flow for the Sentiment Analysis Prototype; Source: own research 27](#_Toc201159903)

[Figure 7 - Application Flow for the Intent Analysis Prototype; Source: own research 28](#_Toc201159904)

[Figure 8 - Application Flow for the Rule-Based Chatbot Prototype; Source: own research 29](#_Toc201159905)

[Figure 9 - Application Flow for the Keyword-Based Chatbot Prototype; Source: own research 30](#_Toc201159906)

[Figure 10 - Application Flow for the Embeddings Prototype; Source: own research 32](#_Toc201159907)

[Figure 11 - Application Flow for the Knowledge Management with Search Prototype; Source: own research 33](#_Toc201159908)

[Figure 12 - Application Flow for the Conversational Chatbot Prototype; Source: own research 35](#_Toc201159909)

[Figure 13 - Application Flow for the Voice AI Chatbot Prototype; Source: own research 36](#_Toc201159910)

[Figure 14 - Application Flow for the Speech AI Assist Prototype; Source: own research 38](#_Toc201159911)

# Abstract

*This dissertation investigates how Artificial Intelligence (AI) models can improve and optimize customer support efficiency. Through a combination of literature review, industry case studies, and ten functional prototypes, the research highlights how AI technologies, such as chatbots, voice assistants, sentiment analysis, and retrieval-augmented generation (RAG), can reduce operational costs, enhance scalability, and improve customer satisfaction.*

*The study adopts an exploratory, design-driven approach, demonstrating that even small and medium-sized enterprises can implement modular AI tools using accessible technologies. While AI shows strong potential in automating routine tasks and assisting agents, the research underscores the importance of hybrid systems where human oversight remains critical in complex or sensitive cases.*

Keywords: Artificial Intelligence, Customer Support, Operational Efficiency, Chatbots, Customer Satisfaction, Design-Driven Approach, Human-AI Collaboration

# Introduction

In recent years, Artificial Intelligence (AI) has emerged as one of the most transformative forces across industries, reshaping how businesses operate, make decisions, and engage with their customers. Among the many domains impacted by AI, customer support stands out as an area where the technology has not only introduced efficiency gains but has also redefined the expectations and experiences of both consumers and service providers.

Customer support traditionally relied on human agents to respond to queries, resolve issues, and maintain customer satisfaction. While this approach remains valuable, it also presents inherent limitations: high operational costs, limited availability, inconsistencies in service quality, and scalability challenges during periods of peak demand. As consumer expectations have evolved, driven by the need for instant, personalized, and seamless service, companies have turned to AI to augment or even automate parts of the support process. The proliferation of technologies such as chatbots, virtual assistants, sentiment analysis tools, voice AI, and generative language models has opened the door to new forms of customer interaction that are faster, more scalable, and increasingly human-like in their capabilities.

This transformation is not confined to large enterprises. Thanks to the democratization of AI technologies, through open-source frameworks, accessible APIs, and low-code platforms, organizations of all sizes can now experiment with and deploy AI-driven support solutions. Nevertheless, the successful integration of these tools requires a clear understanding of their capabilities, limitations, and the operational contexts in which they provide the most value.

This dissertation seeks to explore the central question: “How can AI models be used to improve and optimize customer support efficiency?”. In doing so, it also investigates other sub-questions.

To address these questions, this study employs a mixed-method approach. It begins with a literature review tracing the evolution of customer support, with a particular emphasis on the integration of AI over time. It then presents a series of case studies showcasing how leading organizations across banking, telecom, e-commerce, and retail have successfully implemented AI solutions to optimize service delivery. These are followed by a hands-on, prototype-driven exploration, in which ten AI-based modules were developed to demonstrate practical applications of various AI models, from sentiment detection and keyword-based bots to multimodal agents capable of reasoning and tool use.

The research is grounded in a descriptive, exploratory methodology, relying primarily on secondary data sources and functional system design to illustrate current trends, validate theoretical claims, and provide actionable insights for practitioners and researchers alike.

Ultimately, this dissertation argues that AI should not be viewed as a replacement for human customer service, but rather as a powerful ally that can enhance efficiency, consistency, and satisfaction when deployed responsibly. It further suggests that the future of customer support lies in hybrid systems, where AI handles routine or high-volume tasks, and human agents focus on complex, empathetic, and relationship-driven interactions.

By providing both strategic analysis and practical demonstrations, this research aims to contribute to a better understanding of the role AI can play in shaping the next generation of customer support, making it not only faster and cheaper, but also smarter and more human-centered.

# Literature review

Artificial Intelligence (AI) broadly refers to computer systems or machines that exhibit capabilities traditionally associated with human intelligence, such as learning, reasoning, problem-solving, and language understanding. One succinct definition is that AI is “the ability of a machine to perform cognitive functions we associate with human minds, such as perceiving, reasoning, learning, interacting with the environment, problem solving, and even exercising creativity” (Chui, Kamalnath, & McCarthy, 2020). In other words, AI enables machines to interpret data, draw conclusions, and make decisions in a way that mimics human cognitive processes. AI as a field encompasses sub-domains like machine learning (where algorithms improve through experience), natural language processing (enabling computers to understand and generate human language), computer vision (interpreting visual information), robotics, and more (Sheikh, Prins, & Schrijvers, 2023). The term “artificial intelligence” was first coined in the 1950s, and early AI research focused on rule-based symbolic reasoning (Gold, 2023). Modern AI, however, often centers on statistical and data-driven approaches, for example, deep learning neural networks that learn patterns from large datasets. In business contexts, AI is seen as a transformative technology that can automate or augment cognitive tasks. In customer service, AI systems aim to replicate or assist the problem-solving and communication abilities of human support agents, whether through text-based chatbots, voice assistants, or decision support tools (QuantumBlack, 2024).

Customer Experience (CX) refers to the overall perceptions and feelings a customer has when interacting with a company’s products, services, and brand across all touchpoints (Meyer & Schwager, 2007). One widely cited definition describes CX as “the internal and subjective response customers have to any direct or indirect contact with a company” (Lemon & Verhoef, 2016). This encompasses every aspect of a company’s offering – not only the quality of customer service, but also factors like advertising, ease of use of products, packaging, reliability, and so on (Meyer & Schwager, 2007). Customer experience is inherently holistic, involving cognitive, emotional, sensory, and social responses on the part of the customer (Lemon & Verhoef, 2016). For instance, a customer’s experience might include their satisfaction with how a support request was handled (a service interaction), their impression of a product’s performance, and even their feelings about the brand’s values. In the context of customer support, CX is directly influenced by how efficiently and empathetically issues are resolved (Lemon & Verhoef, 2016). A fast resolution with a friendly interaction contributes positively to CX, whereas long wait times or impersonal service detract from it. Customer experience management has become a strategic focus for businesses, recognizing that superior CX can drive loyalty, word-of-mouth, and competitive advantage (Meyer & Schwager, 2007). Therefore, when deploying AI in customer support, companies should carefully consider its impact on CX, ensuring AI tools not only increase efficiency, but also maintain or improve the quality of the customer’s experience (McLean & Wilson, 2016).

Customer support (or customer service) is the assistance and advice provided by a company to those people who buy or use its products or services (Khachina, 2025). This support can occur before purchase (answering inquiries), during use (helping with setup or issues), or after sales (handling complaints, returns, etc.) (Khachina, 2025). Customer support traditionally involves human service agents interacting with customers via channels such as phone, email, live chat, or in-person helpdesks.

Modern customer support encompasses a range of services across phone, email, live chat, and social media channels. Companies today handle an enormous volume of inquiries – platforms like Zendesk facilitate over 4.6 billion customer resolutions each year (OpenAI, 2025) – underscoring the scale and importance of support operations. In this landscape, customer expectations are higher than ever: consumers demand quick, effective, and around-the-clock help. Meeting these expectations is challenging, as it requires significant staffing and consistent quality. In fact, 96% of customers report they will switch brands after poor service (Shep, 2020), yet many businesses struggle to deliver uniformly high-quality support. Traditional support is costly and often inconsistent. As Intercom (a key player in the customer service and support industry) noted in their manifesto, online businesses simply “can’t afford the humans it would take to meet consumer expectations” (Intercom, 2023). Hiring and training enough agents to provide instant, 24/7 support is financially unsustainable, and even a well-trained human team can vary in responsiveness and expertise.

AI is emerging as a transformative force in customer support, promising to address these challenges through automation and intelligent assistance. Cutting-edge AI models such as OpenAI’s ChatGPT (a large language model capable of human-like text dialog) and Whisper (an automatic speech recognition model for transcribing voice) are enabling new support solutions. From AI chatbots that converse naturally with customers, to voice transcription services that analyze calls, to sentiment analysis tools that gauge customer satisfaction, AI is redefining how support can be delivered. For example, generative AI has been estimated to increase customer service productivity by 30–50% with greater consistency in service quality (Simon, Nicholas, Sukand, & Veronika, 2023). AI-driven support agents do not tire or deviate from best practices and can be available 24/7 at relatively low incremental cost, addressing both the cost and consistency issues.

Moreover, AI is no longer just a technical buzzword. It has become a mainstream concept that directly influences both consumer behavior and business strategy. As shown in Figure 1, interest in AI has surged dramatically since 2022, according to Google Trends. This surge is not limited to the tech community; it's a global shift driven by widespread awareness, media coverage, and tangible use cases that affect everyday life.

**Figure 1 - Graph showing the worldwide Google Search popularity of the term "AI"; Source: Google Trends**

This growing interest translates into real customer expectations. A recent survey revealed that 70% of consumers believe AI will improve their experience with brands, expecting faster, smarter, and more personalized interactions (Arora, et al., 2021). At the same time, 74% of businesses plan to increase their investments in AI for customer service, highlighting the pressure companies feel to keep up (Marti & Sinha, 2024).

From an industry perspective, this demand is matched by record-breaking investment flows. Global venture capital funding for AI companies reached over $100 billion in 2024, representing an 80% year-over-year increase or nearly 33% of all global venture funding (Glaser & Yao, 2025).This investor confidence is not misplaced as AI is projected to contribute to a 21% net increase in U.S. GDP by 2030, according to economic forecasts (Haan, 2024).

One critical performance dimension in customer support is efficiency which translates into how effectively and productively customer issues are resolved. Customer service efficiency refers to providing fast and accurate responses to customer problems while minimizing the effort and cost required (Khachina, 2025). High efficiency in support means customers get timely solutions (short wait times, quick resolution) and the company handles inquiries with optimal use of resources. Common metrics for support efficiency include average response time, first-contact resolution rate, average handle time per case, and the number of queries an agent (or automated system) can handle in a period. Efficiency is closely tied to customer satisfaction as well customers value quick solutions, so improving efficiency often boosts satisfaction and loyalty (Khachina, 2025).

## Evolution of Customer Support

Before the invention of the modern means of communication, customer service was based entirely on direct human interaction. Merchants, artisans, and tradespeople engaged with customers in person, whether in local markets, workshops, or small shops. Trust and reputation were essential, as most transactions relied on verbal agreements and personal relationships.

In ancient civilizations, traders knew that a bad experience could spread quickly by word of mouth, damaging their business. Complaints and feedback were delivered face-to-face, often requiring negotiation or compromise on the spot. An interesting find is that in ancient Mesopotamia, as early as 1750 BCE, King Hammurabi’s Code, included laws protecting buyers from dishonest practices, showing that customer protection was already seen as important in early commerce (The Editors of Encyclopædia Britannica, 2025) (King, 2008).

For long-distance communication, handwritten letters were used to raise concerns or request information. These were sent via messengers or early postal systems, but responses could take weeks or months (Brix, 2025). Patience was therefore a key part of early customer service.

During the Industrial Revolution, business began to scale, and customer expectations changed. This era. Some companies introduced more formal guarantees to ensure customer retention and satisfaction. For example, J.R. Watkins, an American entrepreneur in the 1860s, is credited with offering one of the first money-back guarantees, an early sign of structured customer care focused on satisfaction (The Watkins Co., n.d.).

The inception of the modern customer support can be traced back to the invention of the telephone in 1876 by Alexander Graham Bell, which revolutionized business communication (Hoory (Soft Construct LLC), 2025). Another steppingstone towards more digitized means of communications was the invention of the switchboard in 1894 that allowed businesses and customers to connect their phones to a wider network. By the 1960s, dedicated call centers emerged, utilizing technologies like the Private Automated Business Exchange (PABX) to manage customer interactions more efficiently (Fluido, 2023). The introduction of toll-free numbers by AT&T in 1967 democratized access to customer support, enabling broader reach without cost barriers. Early call centers focused on reactive voice support during business hours, often involving long wait times, limited availability, and manual case logging. These systems were often inefficient, costly, and inaccessible outside business hours. In the late 80s and 90s, with the advancement in computational power and the introduction of the computerized switching systems, a new trend emerged for outsourcing call centers to countries on other times zones, such as India, Pakistan or the Philippines. These were early signs of companies trying to achieve more cost-effective customer support solutions, while also adapting to the client’s expectations of 24/7 support (Fluido, 2023) (Rahaman, 2024).

The advent of the internet in the 1990s introduced new communication channels, notably email and live chat, allowing for more flexible and asynchronous customer interactions. This shift enabled businesses to handle a higher volume of inquiries and provided customers with alternative methods to seek support beyond traditional phone calls. These channels gave customers more flexibility and reduced wait times. However, they also created new challenges: asynchronous communication increased the volume and complexity of support tickets, and many companies lacked the tools to manage them effectively. To put it in perspective, in 1996, one year after Microsoft released Outlook, more than 10 million people worldwide were using Email services, reaching 500 million people by early 2000s (Ozcan & Guler, 2025). At this stage, CRM platforms like Zendesk and Salesforce started to gain traction, offering ticketing systems to track and streamline responses (Hoory (Soft Construct LLC), 2025).

As digital communication platforms proliferated, customer support expanded to include social media, SMS, and messaging apps. This approach ensures a seamless and consistent customer experience across various channels, allowing customers to interact with businesses through their preferred mediums (Customer Connect Expo, 2024). The "always-on" consumer emerged, expecting instant, personalized, 24/7 assistance. This shift led to what is now called omnichannel support, a unified approach to customer interactions across various platforms. However, this expansion also revealed limitations such as that the human support teams were stretched thin, response times lagged, and support quality became inconsistent (Connectys, 2025).

The latest evolution is the integration of Artificial Intelligence (AI). Tools like chatbots, voice assistants, sentiment analyzers, and Generative AI (e.g., ChatGPT, Whisper) can now automate complex tasks, deliver instant answers, and operate 24/7. AI also enables proactive support by analyzing data to anticipate issues, personalize recommendations, and measure satisfaction in real time. This marks a shift from reactive problem-solving to predictive, automated care (Das, et al., 2023).

**Figure 2 - The Evolution of Customer Support; Source: own research**

Customer support has undergone a remarkable transformation over the past century, evolving from basic, reactive assistance to sophisticated, AI-driven, omnichannel experiences. This progression mirrors technological advancements, shifting consumer expectations, and the increasing complexity of global markets.

## Evolution of Customer Support with AI Integration

The integration of AI into customer support has evolved over several decades, transforming how service is delivered. Early explorations date back to the 1960s with systems like ELIZA, an elementary chatbot that demonstrated the potential for machines to mimic human conversation (Rose, 2025). Though ELIZA was a simple pattern-matching program and not a practical support tool, it hinted at a future where automated agents could engage users in natural language.

By the 1980s, the first wave of automation arrived in call centers through Interactive Voice Response (IVR) systems (Rose, 2025). These telephone-based systems used pre-recorded prompts and touch-tone or speech inputs to route calls and answer basic queries. While IVRs were not “intelligent” in the modern AI sense, they paved the way for AI by familiarizing customers with automated self-service. The 1990s and early 2000s saw the emergence of rule-based online chatbots, which provided 24/7 support for simple FAQs (Verma, 2024) (Rose, 2025). These early chatbots were limited, following scripted decision trees, but they established the groundwork for more sophisticated AI-driven support. They could handle common questions via “if-then” logic (e.g. “If customer mentions ‘password’, respond with the account password reset link”)

**Figure 3 - Example of a Decision Tree for a Rule-Based Chatbot; Source: own research**

For instance, artificial conversational entities like ALICE (Artificial Linguistic Internet Computer Entity), developed in 1995, applied pattern-matching rules to simulate conversation, winning the Loebner Prize as an early chatbot 3 times (Wallace, 2009). However, such bots were brittle as they worked only for the specific prompts anticipated.

At the same time, Natural Language Processing (NLP) techniques began to enter customer service. In the late 1990s and early 2000s, researchers and companies experimented with intent recognition and keyword extraction to classify customer requests. These early NLP models could “aim to understand customer input,” enabling rudimentary intent detection and routing of inquiries, though the capability was still very limited (Rose, 2025). Support automation in this era remained largely menu-driven (e.g. press ‘1’ for billing, etc.) or based on simple keyword triggers. Nonetheless, the period saw important milestones such as knowledge management systems and help desk software which became widespread, providing centralized FAQs and ticket tracking. By the mid-2000s, larger enterprises were using workflow automation for support, such as parsing incoming emails to route them to the appropriate department. Still, AI was far from seamless as these tools were “rigid” and often led to frustrating dead-ends when customers went off-script (Verma, 2024) (Rose, 2025). Toward the end of the 2000s, two developments hinted at the future: first, machine learning (ML) started improving classification tasks in support (e.g. learning from past tickets to predict ticket category). Second, consumer messaging platforms opened to bots – notably, China’s WeChat in 2009 allowed third party “Official Account” bots, popularizing more interactive customer-service bots in Asia (Ayyad, 2025).

These advances moved AI from front-end chatbots to behind-the-scenes assistants. Rather than trying (and often failing) to fully replace agents, AI started to support agents by providing insights and automating repetitive tasks. Key milestones of this era include IBM’s Watson (which, after its 2011 Jeopardy! win, was applied to customer service domains like call center decision-support) (IBM, n.d.) and virtual assistants like Apple’s Siri (2011) and Amazon’s Alexa (2014), which, while consumer-facing, accelerated NLP progress and user comfort with AI-driven dialogue. By the mid-2010s, vendors like Zendesk, Salesforce, and others integrated AI. For instance, Salesforce introduced Einstein AI features to recommend replies and knowledge articles to agents. A notable surge came around 2016–2017, when Facebook opened Messenger to chatbots, and many businesses rushed to deploy chat interfaces. This “chatbot boom” saw thousands of simple bots launched on messaging apps, some successes and many failures, illustrating that without genuine language understanding, bots often disappointed (Rapp, Curti, & Boldi, 2021). Nonetheless, by late 2010s the best systems combined machine learning NLP with human-designed conversation flows (a hybrid approach).

The field of NLP has enabled machines to understand and generate human language, while machine learning and deep learning techniques allow systems to improve with data. Modern AI support tools are far more sophisticated than early chatbots or IVRs. They leverage large language models (LLMs) which are AI models trained on massive text datasets to engage in free-form conversations and answer complex questions with a high degree of fluency. OpenAI’s ChatGPT, introduced in late 2022, is a prime example of an LLM that can “respond to prompts with human-like text and voice, answering complex questions with seeming ease” (Simon, Nicholas, Sukand, & Veronika, 2023). Such generative models have captured public imagination and spurred a wave of experimentation in customer service.

This generative leap dramatically expanded what AI could do in support. Instead of answering only pre-programmed questions, LLM-based chatbots can engage in free-form dialogue, answer novel questions by drawing on vast training data, and even draft personalized responses. Support teams began piloting AI “co-pilots” that draft full email or chat replies, which agents can then review and send (Ayyad, 2025).

AI can now summarize long customer conversations, pulling out key details for the next agent or for a follow-up email. Multilingual support became easier by leveraging LLMs’ translation abilities, enabling global service without hiring armies of translators (Rose, 2025). The COVID-19 pandemic accelerated digital support adoption, and AI was at the forefront as companies faced surges in online inquiries (World Bank Group, 2024). Techniques like reinforcement learning from human feedback (RLHF) were key to making these models more suitable for customer service. OpenAI’s InstructGPT models, for example, use RLHF to become “safer, more helpful, and more aligned” with user instructions (OpenAI, 2022). By 2023, major vendors (Microsoft, Google, Meta) all introduced LLM-based chat or support solutions.

Support-specific AI startups also emerged, offering retrieval-augmented generation (RAG) systems that combine LLMs with company knowledge bases for accurate answers. In summary, the 2020s have so far been defined by AI moving from automating back-end tasks to directly interacting with customers in natural language. This evolution has unlocked new possibilities such fully AI-driven chat agents, 24/7 self-service with a human-like touch, but also raised concerns around accuracy, consistency, and oversight (e.g., ensuring these powerful models remain factual and on-brand) (Rose, 2025) (Ayyad, 2025).

Alongside text-based AI, other AI modalities are being applied to support. Automatic speech recognition (ASR) models convert spoken language into text, enabling voice-based customer interactions to be transcribed and analyzed in real time. For instance, OpenAI’s Whisper model (2022) can transcribe conversations in multiple languages with high accuracy, and services like Google’s Cloud Speech-to-Text similarly power voicebots and call analytics. Sentiment analysis is another AI capability relevant to support as it uses NLP to detect emotional tone (positive, negative, neutral) in customer messages or call transcripts. This helps companies gauge customer satisfaction and urgency automatically. According to Microsoft’s documentation, sentiment analysis “mines text for clues about positive or negative sentiment” and can even pinpoint which aspects of a service are causing frustration (Microsoft, 2025).

**Figure 4 - Timeline of Chatbot and AI Evolution in Customer Support; Source: own research**

When discussing AI models improving customer support efficiency, we mean that AI can automate or streamline parts of the support process to handle more inquiries in less time or with fewer resources, all while maintaining accuracy and solution quality. For example, an AI chatbot might instantly answer common questions 24/7, thereby reducing the workload on human agents and shortening customer wait times (Andrade & Tumelero, 2021). A successful AI deployment can allow human support staff to focus on more complex tasks, improving overall service throughput. Indeed, a study at a Brazilian bank showed that integrating an AI virtual assistant (IBM Watson-based) led to 7.6 million customer service interactions handled via chatbot in one year, greatly increasing service availability and agility while freeing human agents for complicated cases (Andrade & Tumelero, 2021). Efficiency gains from AI can include qualities like agility, availability, accessibility, and predictability in service delivery (Andrade & Tumelero, 2021).

Lastly, Gartner suggests in their latest prediction for service report that our evolution will circle back from omnichannel to single channel as the multimodal LLMs will be able to dynamically process text, images, videos and files, all at once, thus enabling customers to interact with the AI agent through their preferred mean of communication (Gartner, 2025). OpenAI has recently showed how this future might look with the introduction of the GPT-4o vision capabilities in their conversational tool (OpenAI, 2024).

## Current AI Models and Tools in Use

AI technologies are being applied in customer support through various models and tools. Key applications include chatbots and virtual agents that handle customer conversations, natural language processing techniques for understanding customer queries (like intent recognition and sentiment analysis), predictive analytics for anticipating customer needs, and comprehensive conversational AI platforms that orchestrate interactions.

Chatbots are software agents that interact with users through natural language (text or voice), often serving as the first line of support in customer service. They are designed to simulate a human conversational partner, answering questions and performing simple actions (Shweta, 2022). Virtual agents (or virtual customer assistants) is a term often used interchangeably with chatbots, though it sometimes implies a more advanced system (potentially with a visual avatar or voice interface) that can handle a broader range of tasks (Lafferty, n.d.). In practice, both aim to automate customer interactions and provide instant assistance.

Modern chatbots in customer support use a combination of predefined conversation flows and AI algorithms. At their core, they rely on Natural Language Understanding (NLU) to parse what users are asking, and dialogue management to decide on the response or action. For example, if a customer types “I need to reset my password,” the chatbot’s NLU component will classify this as an “intent” to reset a password (OpenAI, 2025). The bot can then follow a scripted workflow or API call to guide the user through a password reset procedure. Many bots also incorporate retrieval mechanisms which are searching a knowledge base for relevant articles to present to the user (Zarecki, 2025).

Chatbots range from generative AI bots like Intercom’s Fin to domain-specific virtual assistants. Intercom’s Fin, launched in 2023 and powered byGPT-4, can converse with customers and resolve common queries. It was reported to successfully resolve up to 50% of customer support questions instantly, deflecting a large portion of routine workload from human agents (Intercom, n.d.). Unlike clunky scripted bots of the past, AI agents like Fin use LLMs and a company’s knowledge base to generate accurate, conversational answers, and seamlessly hand off more complex issues to humans when needed (Intercom, n.d.). Similarly, Zendesk introduced AI-powered “agents” that manage entire conversations autonomously. Early pilots of Zendesk’s AI agents (built on OpenAI models) showed promising results as they can plan multi-turn dialogues and execute tasks with an aim of achieving resolution in up to 80% of cases, far beyond the capabilities of traditional rule-based bots (OpenAI, 2025). These generative bots mark a shift from static decision trees to dynamic, context-aware assistance.

Efficiency gains from chatbots are significant as they can handle multiple queries simultaneously, never require breaks, and provide 24/7 service. Studies have shown that a well-implemented chatbot can resolve a large portion of routine inquiries without human intervention (some reports cite over 70% of common questions) (Chudleigh, 2025). This deflection of tickets reduces wait times and operational costs. For instance, the travel company Amtrak’s virtual assistant “Julie” reportedly answered over 5 million questions in a year and saved the company around $1 million in support costs annually, a concrete illustration of efficiency improvement (Nelson, 2017).

A critical aspect of chatbot design is the handoff to human agents. No matter how advanced a bot is, there will be queries it cannot fully handle (complex, unusual, or sensitive issues). Best practices dictate that the chatbot should gracefully transfer the conversation to a human agent in such cases, ideally along with context gathered so far (Intercom, n.d.) (Microsoft, 2024). This hybrid approach maintains efficiency while ensuring customer satisfaction for edge cases. Indeed, research finds that customers expect to be offered a human option if the bot falls short, and frustration grows when bots present a dead-end (Henderson, 2024). Therefore, successful chatbot deployments are integrated tightly with live support teams (Microsoft, 2024).

Another evolution in chatbot technology is the incorporation of voice. Voice-based virtual agents add speech recognition and text-to-speech on top of the chatbot’s NLU capabilities, allowing customers to talk to an AI as they would on a call (Rumley, Nguyen, & Neskovic, 2023). For example, some airlines employ voice bots that can understand spoken requests like “I want to change my flight” and execute the change or route the call appropriately (Billan, 2025). Voice virtual agents can improve efficiency by shortening call times and automating routine call types, though they also inherit the challenges of speech recognition errors and the need for natural-sounding synthesized voices (Rumley, Nguyen, & Neskovic, 2023). Another use for these AI systems, such as Whisper, in the context of call centers is that these speech recognition model achieve very high accuracy and produces a live text transcript of customer calls (Radford, et al., 2022). This transcript can then be analyzed by other AI components (for sentiment, keywords, etc.) (Dialpad, n.d.).

To improve CX, sentiment analysis tools are also increasingly integrated into support platforms to monitor customer mood and satisfaction (Reputation, 2025). For instance, Azure Cognitive Services (Emami, 2021) and IBM Watson Tone Analyzer (Kuan & Khan, 2022) can evaluate the sentiment of text in support tickets or chats. This technology lets support teams automatically flag angry or dissatisfied customers (Dialpad, n.d.).

Although the general tendency is to think of the role of AI in customer support of as just reacting to incoming queries, its role in predicting and preempting customer needs is also highly increasing. Predictive analytics uses data and machine learning algorithms to forecast future events or behaviors (NICE, n.d.). In customer support, predictive models can anticipate which customers might require help or identify issues before the customer is even aware of them, allowing the company to take proactive measures. This leads to higher efficiency by reducing the influx of support tickets and improving customer satisfaction by addressing problems early (NICE, n.d.).

One application is predictive customer issue detection. By analyzing patterns in product usage data, sensor readings, or system logs, AI can sometimes detect that a customer will experience a problem. For instance, an AI system might notice that a user’s car has potential problems (via telemetry data), thus predicting that the user is likely to contact support. The company could then send an automated message: “We noticed a problem with your car and have issued a software update,” thus solving the issue proactively (Steinle, 2025). This reduces the need for the customer to initiate a support request at all.

Another example is churn prediction and retention outreach. Machine learning models can predict which customers are at risk of dissatisfaction or churn based on their interaction history, support tickets, sentiment, and other factors. Support teams can use this insight to proactively reach out to those customers with offers of help, loyalty incentives, or personalized guidance, ideally before the customer decides to leave (Takyar, Customer churn prediction using machine learning: A comprehensive overview, 2025). This shifts support from a passive, reactive role to an active part of customer experience management.

Predictive analytics also helps in support resource optimization. AI models might predict spikes in support volume (for example, after a major product update or during holiday seasons for retail) (Steinle, 2025). Knowing this in advance allows managers to staff appropriately or deploy extra chatbot capacity. Predictive models could forecast the expected number of support cases of each type per day, so that workflows can be adjusted for efficiency (e.g., ensuring the knowledge base has updated answers for the most anticipated issues) (NICE, n.d.).

AI can also help in predictive routing where it predicts the best way to handle a customer’s query for optimal outcome. For instance, based on the customer’s profile and query, an AI might route high-value customers or urgent-sounding inquiries directly to senior agents, whereas straightforward questions go to a chatbot or a junior queue (Glia, n.d.). This maximizes the efficient use of skilled human agents and keeps resolution times low.

# Methodology

This paper follows an exploratory, descriptive research methodology focused on understanding and illustrating how AI models can improve customer support efficiency. Rather than conducting primary empirical research (such as surveys or experiments), I rely on secondary sources and practical examples to build insights (Templier & Paré, 2015).

The research was guided by several key exploratory questions:

1. What are the most common and high impact use cases of AI in customer support?
2. How are different AI models (e.g., LLMs, embeddings, speech recognition) integrated into actual support workflows?
3. What practical challenges and benefits have been observed in implementations?
4. How can modular tools be assembled to support intelligent, multi-modal, and context-aware support systems?
5. How can support operations progress from simple automation to more sophisticated agentic AI systems?

I gathered information from a variety of credible sources, including academic papers, industry whitepapers, product documentation, company blogs, and case studies. Academic literature (e.g. peer-reviewed articles and preprints) provided theoretical context on AI in customer service and known benefits and challenges. Industry sources such as technology vendor websites, AI product announcements, and consulting firm reports, provided up-to-date information on practical implementations and outcomes.

Using the collected sources, I identified key use cases of AI in customer support (for instance, AI chatbots, voice transcription, sentiment analysis, etc.). For each use case category, I selected representative real-world examples to examine in detail. These examples were drawn from reported case studies or product capabilities. By analyzing these concrete instances, I illustrate how theoretical benefits of AI translate into practice.

I qualitatively analyzed the content from sources to extract recurring themes such as common benefits (speed, cost reduction) and challenges (accuracy, bias) of AI in support. I cross-reference findings from multiple sources to ensure a reliable and balanced perspective. For instance, if an industry blog claimed a certain efficiency gain, I sought corroborating evidence or context from other analysts or research.

In parallel with the desk and case-based research, a suite of ten functional prototypes was developed to simulate key AI capabilities in customer support workflows. These prototypes were implemented using modern open-source tools, including Python, Streamlit, LangChain, FAISS, OpenAI APIs, and speech analysis libraries. Each tool illustrates a specific stage in the evolution of AI, starting from simple rule-based systems and keyword matchers to memory-augmented chatbots, voice assistants, and autonomous reasoning agents with tool access. These prototypes serve as design artifacts, intended not for testing model performance but for demonstrating feasibility and functionality.

This methodology, which is grounded in secondary research, descriptive case analysis, and iterative prototyping, is well suited for a topic as rapidly evolving as AI in customer service. It does not aim to generate statistically generalizable findings, but rather to map the state-of-the-art, synthesize expert perspectives, and offer concrete, interactive illustrations of how AI can be applied to real-world service challenges. This aligns with calls for more design and example-driven research in emerging technologies (Hevner, March, Park, & Ram, 2004).

# Case Studies

This chapter presents the findings from the analysis conducted to explore how companies today leverage AI to enhance customer support and experience. Building upon the literature review, which outlined the evolution and growing significance of AI in customer interactions, the following sections provide concrete, recent evidence of AI's measurable impact on business outcomes. Specifically, the results illustrate how AI-driven technologies have influenced customer satisfaction, operational efficiency, and overall performance metrics. The data includes real-world case studies, statistical improvements, and industry trends to offer a comprehensive understanding of AI’s current and practical benefits within customer support environments.

## AI Powered Chatbots

AI chatbots are now deployed by companies in banking, e-commerce, telecom, retail, and more to handle a large volume of routine service requests. These virtual assistants use natural language processing and machine learning to engage customers in human-like conversations, answer FAQs, troubleshoot issues, and even perform transactions – all available 24/7. By automating common inquiries, chatbots free human agents to focus on complex or sensitive cases, while improving response times and consistency.

In banking, Bank of America’s Erica exemplifies a successful virtual assistant. Since its launch in 2018 it has handled over 2 billion client interactions, with usage accelerating (it reached 1 billion interactions in 4 years and doubled to 2 billion 18 months later) (Bank of America, 2024). Erica can fulfill a range of requests, from providing account balances to offering personalized financial advice. Notably, 98% of users get what they need from Erica in under 44 seconds, reflecting a high rate of first-contact resolution; only the remaining 2% of queries are escalated to human agents via live chat (Bank of America, 2024). This speed and effectiveness have improved customer convenience and likely deflected millions of calls from service centers, thus improving the efficiency of the customer support department, allowing them to deliver better suited help to their customers (Bank of America, 2025). In April 2024, Bank of America reported that 20 million clients are regular Erica users, and the assistant had provided 800 million answers and 1.2 billion personalized insights (e.g. spend analysis, subscription reminders) to date (Bank of America, 2024). Such data-driven, proactive support has helped deepen customer engagement, while the always-on availability of AI service aligns with modern expectations for instant support. Nikki Kats, head of digital at Bank of America mentioned that Erica’s role is to become a personal concierge agent of each client, while also helping them control their finances, thus placing Erica as a centerpiece item in their bank-client relationship formula (Bank of America, 2025). Besides its role of helping customers with their inquiries, Erica managed to also fulfill jobs that would fall into the marketing department of the companies by delivering more than 7000 birthday wishes to clients and by also notifying clients of their new partnership with Starbucks (Bank of America, 2024). This shows that with a client-centric and humane approach, AI chatbots can not only serve the role of a support agent, but also become a key part of the customers life, thus increasing the trust of said customer in the company they are interacting with. The good implementation of Erica in Bank of America’s business and the increase of customer satisfaction reportedly increased their earnings by 19%, as multiple sources suggest (Mathews, 2024) (Aggarwal, 2023). Additionally, following the successful foundation of Erica, in 2020, Bank of America also launched and internal tool for employees called “Erica for Employees”, that is being used by more than 90% of the bank’s employees and reduced IT support calls by more than 50% as of April 2025 (Bank of America, 2025). Lastly, Bank of America is also using AI to elevate and personalize their exclusive services, such as the ones offered by their private banking division, through their “ask PRIVATE BANK” tool, but also to prepare the bank’s employees to prepare for client meetings (Bank of America, 2025). Bank of America continues to invest and thus strengthening their status as innovators and leaders in the financial services companies in the U.S.A., spending $13 billion annually on technology, $4 billion of which on new technologies in 2025 (Bank of America, 2025).

In the telecommunications sector, Vodafone deployed an AI chatbot named TOBi across its global markets to handle customer support queries (IBM, n.d.). Initially launched in 2018 on IBM Watson and later enhanced with Azure OpenAI and renamed as SuperTOBi, TOBi interacts via web and messaging channels, answering questions about bills, data plans, technical issues and many more (Hook, 2019) (Wood N. , 2024). The impact on operations has been noteworthy as TOBi is able to resolve about 70% of all customer inquiries on its own, significantly reducing the volume of issues that reach human call agents, which resulted in a 70% drop in cost-per-chat compared to prior live-agent chats (Vohra, 2024). In Germany in 2023, TOBi was able to handle 8 million customer requests, managing to resolve 65% of them without any human support (Dale, 2024). In other words, assisting customers via AI now costs less than one-third of a traditional chat, a substantial saving for an operator with over 600 million subscribers worldwide such as Vodafone (Vohra, 2024). Furthermore, in 2024, following the successful footsteps of TOBi, Vodafone introduced a revamped and more advanced generative AI chatbot dubbed “SuperTOBi” (Vodafone, 2024). SuperTOBi uses large language model capabilities to better understand complex customer requests (Vodafone, 2024). In Portugal (the pilot market), first-contact resolution rate jumped from 15% to 60% after SuperTOBi’s launch, meaning far more customers got their issue solved on the first try (Vodafone, 2024). Consequently, customer satisfaction rose as Vodafone’s online Net Promoter Score (NPS) increased by 14 points, reaching 64, indicating higher customer approval of the digital experience (Vodafone, 2024) (Vodafone, 2024). the back offices of Vodafone’s customer support, another tool built on the same technologies as SuperTOBi, called SuperAgent, is also assisting customer support agents by quickly finding references to customer’s inquiries or helping them navigate and answer complex queries (Jackson, 2024). These metrics and efforts underscore how a well-trained AI agent can both cut costs and improve service outcomes simultaneously, both on the front facing side and in the back offices. Moreover, Vodafone is keen on improving their processes through the use of AI as the company recently signed a ten-year partnership deal with Microsoft to make use of their cloud infrastructure and develop better costumer focused AI experiences and services (Noone, 2024).

E-commerce leaders have similarly embraced chatbots at scale. China’s Alibaba employs one of the world’s most advanced AI customer service systems to support both buyers and sellers on platforms they own like Aliexpress, Taobao and Tmall (Vohra, 2024). In busy periods, Alibaba’s chatbots handle over 10 million messages a day, resolving approximately 75% of all customer questions online and even about 40% of telephone hotline inquiries without human help (Wang, 2024). By offloading repetitive inquiries to AI, Alibaba reportedly saves over ¥1 billion (~$150 million) per year in customer service costs (Wang, 2024). Notably, this high automation has not compromised customer happiness, but it actually helped increase it as Alibaba saw a 25% increase in customer satisfaction scores after deploying the AI service bots (Wang, 2024). This suggests the bots are effectively assisting customers in a timely manner for routine needs, while agents can concentrate on complex issues. One key differentiator of Alibaba from other companies is that they are also a key player in the cloud computing industry, while also being one of the few big players that have successfully launched a commercially validated LLM (Alibaba, 2025). To sustain their business goals and further develop their services, Alibaba announced that they plan to invest $53 billion in cloud infrastructure and AI over the course of the next 3 years, hinting at the idea that AI is not only a novelty feature, but rather a main focus of the company (Wheeler, 2025).

Another interesting story comes from the fintech giant Klarna, a global payments provider, which built an AI assistant to scale customer support (Klarna). Klarna’s bot now manages two-thirds of all customer chats, handling over 2.3 million conversations and effectively performing the workload of 700 full-time agents (Klarna, 2024). Through cost savings and increased capacity, the solution improved operational efficiency by an estimated $40 million in 2024 (Klarna, 2024). Customer experience metrics also improved as the average query resolution time dropped from 11 minutes to under 2 minutes thanks to instant AI responses , and the AI bot’s customer satisfaction ratings match those of human agents while actually exceeding humans in accuracy, leading to 25% fewer follow-up contacts (Vohra, 2024). However, Klarna's overreliance on AI soon exposed the limits of automation, as AI struggled to handle complex, emotionally nuanced cases that required empathy and discretion (The Economic Times, 2025). While AI-driven efficiency initially seemed promising, the lack of human agents led to a decline in service quality, frustrating customers with generic, unhelpful responses. Recognizing the shortcomings of full automation, Klarna is now rehiring human workers to restore trust and service reliability (The Economic Times, 2025). The case highlights two interesting facts: one, the risks of excessive automation, where cost savings and speed may come at the expense of meaningful customer interactions and brand reputation, and second, the fact that AI, at least at the moment, should not entirely replace humans, but rather assist them, like a copilot (The Economic Times, 2025).

In the retail sector, H&M launched a chatbot to assist with online customer service and saw notable benefits (Ahern, n.d.). The AI chatbot operates on H&M’s website, mobile app, and social media, helping customers with product discovery, checking store inventory, order status, sizing advice, and even handling returns or exchange queries (Filipsson, n.d.) (Shetty, 2024). By automating these frequent interactions, H&M’s implementation led to a 35% rise in customer satisfaction ratings and a 20% cut in support operating costs, according to reports (Shetty, 2024). The chatbot’s ability to provide instant answers and personalized recommendations, drawing on customers’ browsing and purchase history, has created a smoother shopping experience (Goyal, 2023) (Shetty, 2024). Importantly, it also serves as a buffer during high-volume periods. For example, during holiday sales, H&M’s chatbot can manage thousands of simultaneous inquiries about promotions, stock availability, and shipping, ensuring customers receive timely information without overwhelming human staff (Filipsson, n.d.). This scalability preserves service quality during traffic spikes.

Overall, the case studies clearly demonstrate that AI-powered chatbots, when properly trained on a company’s knowledge base and seamlessly integrated into customer channels, can lead to significant improvements in response speed, consistency, and overall service availability. This results in measurable gains such as reduced contact volumes and lower support costs.

However, while these examples highlight the impressive utility of AI in customer support, it is equally important to recognize current limitations in their design and implementation. Many of the AI bots deployed are not truly “intelligent” in the sense that they lack advanced generative capabilities (Talkative, 2023). They operate primarily on pre-determined scripts or answer pools, which means they are not as creative or flexible as human agents when faced with complex, emotionally nuanced, or non-standard customer queries (Tongco, 2025). In essence, these systems excel at handling straightforward or routine tasks but fall short when more personalized or intricate problem-solving is required (Tongco, 2025).

A compelling example of the risks of over-reliance on AI is illustrated by the experience of Klarna. The fintech giant once replaced approximately 700 customer support personnel with AI-driven chatbots (Dobkin, 2025). Initially, this move was intended to streamline operations and drive efficiency. However, the outcome was a notable decline in customer service quality, which the company itself acknowledged by later hiring human agents to address the shortcomings (Dobkin, 2025). This case indicates that while AI chatbots can serve as effective copilots for routine functions, they should not replace human judgment altogether. Instead, advanced generative AI or a hybrid system, where AI supports human agents, may be better suited for scenarios that require a higher degree of empathy, creativity, or complex decision-making (Tongco, 2025).

In conclusion, the integration of AI in customer support is certainly on the right track, as it enables faster and more efficient handling of basic tasks. Yet, for a truly high-caliber customer experience, especially in emotionally charged or complex interactions, a balanced approach that combines AI’s strengths with human expertise is essential. This strategy leverages the speed and consistency of AI for streamlined processes while retaining the uniquely human ability to understand nuanced customer situations and respond creatively when needed (Tongco, 2025).

## AI Enhanced Knowledge Management

Beyond customer-facing chatbots, AI is also being deployed behind the scenes to empower support agents with better knowledge and tools. AI-powered knowledge management systems help by intelligently retrieving relevant information, guiding agents through troubleshooting steps, and even automating certain manual tasks (WorkAI, 2023). By serving as a real-time coach or research assistant, these tools enable human agents to resolve issues more accurately and quickly. This section examines how companies have improved internal support operations using AI for knowledge bases and agent assistance.

One example comes from Banca Transilvania (BT), a major Romanian bank and one of the most recognizable Romanian brands (Brand Finance, 2024). BT implemented an AI based internal virtual agent named “David” to support its operations helpdesk teams (Banca Transilvania, 2020) (DRUID, n.d.). Built on a conversational AI platform, David interfaces with employees, allowing them to query the bot for information or assistance with internal processes. The AI agent provides instant answers from the bank’s extensive knowledge bases and can perform automated tasks such as interest rate calculations, generating routine reports, and performing verification checks across the bank’s applications (Banca Transilvania, 2020) (DRUID, n.d.). By doing so, it reduces the workload on IT support and back-office staff and minimizes human error in routine operations. The primary goal was to improve employees’ access to correct information and expedite troubleshooting when they encounter system issues (Banca Transilvania, 2020) (DRUID, n.d.). Early results indicate the tool has succeeded in increasing productivity of internal teams and ultimately translating to better service for customers (since employees can resolve customer requests faster) (Banca Transilvania, 2020) (DRUID, n.d.). Reportedly, in the first two months after its launch, David helped BT to save over 600.000 RON (Banca Transilvania, 2020). BT’s digital director noted the vision is for this AI agent to become “our most important tool in daily operational activities,” enabling more efficient, high-quality service delivery to customers and colleagues (DRUID, n.d.). David is not BT’s first experiment with AI, as they have also successfully launched two other internal bots, Aida and Raul (Banca Transilvania, 2020). These bots collectively saved hundreds of workhours and freed employees of mundane work (Banca Transilvania, 2020). These deployments highlight how AI-driven knowledge bases can ensure every support representative (or internal user) gets consistent, instant answers drawn from a centralized repository, rather than manually searching through documents or escalating simple queries.

Another knowledge-focused implementation is by British Telecom in its UK contact centers. After a series of mergers, British Telecom had multiple disparate knowledge systems for its 10,000 service agents, which led to inconsistent answers and high repeat call rates (Young S. , 2022). British Telecom introduced an AI-powered guided knowledge solution called “Albert” to unify and streamline agent knowledge access (Young S. , 2022). Unlike a traditional keyword search, Albert presents agents with a dynamic decision tree or “guided help” workflow, then the AI asks context questions and then walks the agent through the compliant steps to diagnose and answer the customer’s issue (Young S. , 2022). This prescriptive approach ensures that regardless of which agent handles the call, the customer receives the same accurate solution, thereby standardizing service quality (Young S. , 2022). The system contains tens of thousands of distilled knowledge articles and is tailored for different departments (including product-specific info and compliance scripts) (Young S. , 2022). It even prefills customer and device data from British Telecom’s CRM into the guided process, saving the agent time during the call (Young S. , 2022). The impact has been positive: British Telecom reports an 8% increase in first-contact resolution (fewer customers needing to call back) and a 5% increase in NPS after rolling out the AI knowledge system (Young S. , 2022). Notably, agent training time was cut roughly in half, as new hires ramp up faster by relying on Albert’s step-by step guidance (Young S. , 2022). This case shows how an AI-enhanced knowledge base can reduce error rates, improve consistency, and speed up onboarding, benefiting both customers and support teams.

In summary, AI-driven knowledge management solutions, whether in the form of an internal chatbot, guided search, or agent assist dashboard, have proven effective at boosting support operations. By delivering the right information at the right time (either to employees or directly to customers), these systems increase first-contact resolution and consistency (Banca Transilvania, 2020) (DRUID, n.d.) (Young S. , 2022). The results include measurable improvements such as higher resolution rates and NPS and huge reductions in internal workloads (Banca Transilvania, 2020) (DRUID, n.d.) (Young S. , 2022). Ultimately, this leads to faster, more accurate service for customers and a more empowered support workforce

## AI for Call Analysis and Voice Support

Regardless of a user’s generation or other demographic aspects, voice calls remain a critical support channel for many industries, and AI technologies are enhancing these traditional call center interactions in multiple ways (Gartner, 2025). Speech recognition and natural language understanding (NLU) allow AI to interpret live customer calls, transcribing conversations in real time and analyzing them for intent or sentiment (Microsoft, 2025). Sentiment analysis on calls can alert supervisors or prompt agents when a customer is frustrated, enabling timely intervention (Subhashis & George, 2024). AI can also provide real-time recommendations during calls (agent assist) or automate parts of calls via interactive voice response (Teneo, n.d.). Post-call, machine learning models can summarize call logs and extract insights (Kane, et al., 2021). These capabilities help improve quality, reduce handling time, and ensure consistency in voice support (Teneo, n.d.).

Amazon has applied AI to its own customer service and also offers these tools through AWS (Amazon, n.d.). Amazon developed neural network models for its customer support that could both handle common issues end-to-end and assist human agents with suggested replies (Amazon, n.d.). In internal trials, neural-network chatbots outperformed Amazon’s older rule-based bots in “automation rate” which measures if the automated agent was able to successfully resolve the issue without referring it to a human agent and if the customer contacted customer service again within 24 hours of the initial resolution (Kramer, 2020). Simultaneously, Amazon tested an AI agent assist system that generates response suggestions for human agents, which has been shown to save agents time on typing and research (Kramer, 2020). These AI models, trained on millions of past customer interactions, allowed Amazon to support new regions and languages more easily by automating routine text chats and guiding agents on voice calls (Kramer, 2020). Amazon’s cloud AI, Amazon Lex, further exemplifies voice innovations as it can be used to build voicebots that understand natural speech, analyze caller sentiment, summarize conversations, and even trigger back-end workflows for the agent (Amazon, n.d.) (Amazon, n.d.). By leveraging such AI, a voice call can be immediately transcribed and important details (like account info or complaint type) extracted for the agent’s console. Additionally, the AI can monitor the caller’s tone or words to gauge sentiment. For example, flagging if a customer sounds angry, enabling supervisors to step in or the system to prioritize that call (Amazon, n.d.) (Amazon, n.d.). Early adopters report meaningful efficiency gains as a McKinsey study found that blending AI bots into contact centers can double agent productivity while cutting cost-per-call in half, by automating routine work and shortening call durations (Blackader, et al., 2025).

Enterprises in various sectors have deployed AI voice agents to augment their call centers. Camping World, a U.S. retailer of RVs and camping gear, faced surging call volumes and implemented an IBM Watson voice assistant named “Arvee” to handle after-hours and overflow calls (IBM, n.d.) (Wood C. X., 2024). Arvee would answer customer calls 24/7, provide information or record messages, and then hand off to live agents when necessary (with a “warm transfer” including a summary of the caller’s issue) (IBM, n.d.) (Wood C. X., 2024). This ensured no customer inquiry was missed due to call center hours or hold times. The results included a 40% increase in customer engagement (more inquiries handled) and a reduction in average wait time by 33 seconds due to the AI screening and addressing many questions up front (IBM, n.d.) (Wood C. X., 2024).

Another example is in telecom. Vodafone has integrated AI-driven voice capabilities into its contact center solutions to enhance call routing efficiency and customer authentication. For instance, Vodafone New Zealand partnered with Convai to launch Vodafone Voice Concierge, which uses AI, machine learning, and speech recognition to analyze customer requests and direct calls accordingly. This solution has been shown to reduce average call times by 15–30 seconds (One New Zealand, 2020).

In the healthcare industry, Regina Maria, a leading private hospital network in Romania, has integrated an AI speech recognition system into its call centers to radically improve patient interactions (DRUID, n.d.) (Regina Maria, 2022). When a patient calls, the system immediately greets them and carefully listens to their needs to accurately route the call to the most appropriate assistant (DRUID, n.d.). This not only spares patients from repeatedly explaining their situation to different agents but also equips the human operators with immediate context about the caller’s issue (DRUID, n.d.). As a result, patients feel as if they are actively engaging in conversation rather than waiting in a queue, which improves their overall experience. Moreover, by effectively filtering out less critical calls, this intelligent system helps prioritize patients with urgent or complex needs, thereby boosting the efficiency and output of the contact center operations (DRUID, n.d.).

More broadly, speech analytics AI is being used by banks, insurers, and telecoms to automatically monitor calls for quality and compliance, something not feasible with human auditors (Spitch, n.d.). These AI systems can detect if agents are using the correct greeting, if required legal disclaimers were read, or if the customer’s sentiment turned negative during the call (Spitch, n.d.).

AI is also aiding real-time agent coaching on calls (Convin, 2025). Startups and software like Balto and Cogito provide live guidance by analyzing ongoing conversations (Balto, n.d.) (Cogito, n.d.). They might display cues like “Slow down speech” or suggest a relevant solution article when they recognize a certain problem description (Convin, 2025). By analyzing voice and language patterns, such tools help agents adjust their behavior immediately (e.g., to show more empathy if customer sounds upset) (Convin, 2025).

Companies using these have reported improvements in call outcomes; for instance, one case study showed a 9% rise in customer satisfaction for Motel Rocks after sentiment-aware assistance was introduced, as agents could intervene more empathetically when AI flagged an unhappy caller (Zendesk, n.d.).

Overall, AI in voice support is driving faster resolutions and more consistent service. Automation of simple calls through conversational IVR or voicebots means customers get quick answers (for example, account balance inquiries or order status can be given by a bot in seconds) (Zendesk, n.d.). Meanwhile, human agents benefit from AI driven transcription, sentiment alerts, and suggested answers, all of which shorten call duration and improve quality (Convin, 2025) (Balto, n.d.) (Cogito, n.d.). Gadi Shamia, CEO of Replicant, a conversational AI solution provider, noted that enabling AI in their contact centers achieved a 50% reduction in cost per call while maintaining service levels (Blackader, et al., 2025).

The continuing trend is a hybrid model as AI handles the straightforward interactions or parts of a call, and seamlessly hands off to humans for complex, emotional, or high-value conversations, with the AI still assisting in the background (Convin, 2025) (Balto, n.d.) (Cogito, n.d.).

## AI in Returns and Refunds Automation

Handling product returns and refunds is a significant aspect of customer support in retail and e-commerce, and AI is now streamlining these processes as well (Takyar, n.d.). Traditional returns handling can be costly and slow involving multiple back-and-forth communications and manual approvals (Takyar, n.d.). AI is helping by automating return eligibility decisions, providing instant refunds under certain conditions, and detecting fraudulent behavior (Relevance AI, n.d.). This results in faster resolution for customers and lower operational burden for companies (Beam, 2024).

A notable example is Temu, a rapidly growing global e-commerce marketplace (Temu, n.d.). Temu has leveraged AI driven automation in its customer service chat to simplify the returns experience. Through the Temu app or website, customers can initiate return or refund requests via a chatbot interface (Temu, 2025). The system uses predefined rules and machine learning to decide if a return needs physical inspection or if an instant refund can be granted (ProductScope, 2024). In practice, many Temu customers receive immediate refunds for low-cost items simply by requesting via the app, without needing to ship the item back (Young K. , 2024). User reports indicate that for items below a certain price, the Temu bot will often issue a “returnless refund” crediting the customer’s account right away and saving them the hassle of packaging and mailing the product (Young K. , 2024). In one forum discussion, a customer described getting about $120 worth of refunds for a few orders within minutes through the chatbot, all without providing photos or detailed explanations, as the AI approved the requests automatically (Choice Cheapies, 2024). This ultra-fast resolution delights customers and encourages loyalty (they get problems resolved with minimal effort). From Temu’s perspective, using AI to automate these refund decisions likely reduces support labor and may be economically sensible for inexpensive goods (Young K. , 2024). It aligns with a broader industry trend of “returnless refunds” for cheap or hard-to-resell items (Hadero, 2024).

Major retailers like Amazon have also implemented algorithmic return automation. Amazon, Walmart, Target, and others, will sometimes proactively give customers a refund and let them keep the product, if the cost of return shipping outweighs the item’s value (Hadero, 2024). These decisions are powered by backend algorithms that consider factors such as item price, customer purchase history, and return shipping logistics. For example, a $10 item might cost $8 to ship back and then still need processing, so the system may choose to refund without return to save overall cost and provide a better customer experience (Santos & Koromyslova, 2020). The Associated Press reports that this “returnless refund” approach is used sporadically for low-cost or bulky to-ship products as a tool to keep online shoppers satisfied while cutting reverse logistics expenses (Hadero, 2024). Customers appreciate the convenience (essentially a free item and no trip to the post or courier office), which can improve brand sentiment. At scale, such AI-driven policies have to be carefully managed to prevent abuse, but when combined with fraud detection models (flagging serial return abusers), companies can minimize losses (Santos & Koromyslova, 2020).

AI also assists in the workflow of returns. H&M, for instance, integrates its chatbot to guide customers through the return process. The chatbot can automatically generate digital return labels and instructions for the customer, based on the order details it pulls up (Filipsson, n.d.). This reduces the need for a human agent to manually handle each return request. Customers get a quick, self-serve solution to process returns or exchanges, and the business ensures the correct data (like return merchandise authorization numbers) are captured consistently by the AI (Filipsson, n.d.). Similarly, in other retail settings, AI vision technology is being piloted to assess returned items (e.g., using photos to judge condition) to automate refund approvals (Mitzner, 2025) (Drenik, 2025). Although still emerging, these AI applications aim to cut down the time from return initiation to refund issuance dramatically.

Another critical area is returns fraud prevention, where AI models analyze return patterns to flag suspicious behavior (like someone who frequently claims high-value items “never arrived” or returns bricks in the box) (Mitzner, 2025). A recent study suggested about 15% of returns in e-commerce could be fraudulent or abusive (PYMNTS, 2024). AI can cross-reference data (frequency of returns, customer profile, product value) and decide to require extra verification for some returns. For example, beauty retailer 100% PURE adjusted its returns policy to combat abuse by using data to identify serial abusers and then enforce stricter return conditions on those accounts (PYMNTS, 2024). As one expert put it, “the key is to teach your AI models to ‘think’ the way fraudsters do and ask the right questions at the right time” (PYMNTS, 2024). By doing so, AI not only smooths the return experience for honest customers but also saves companies money by intercepting fraudulent refunds.

In summary, AI-powered automation in returns and refunds is improving efficiency and customer goodwill. E-commerce innovators like Temu demonstrate how a largely AI-driven returns process can make an online retailer stand out by offering hassle-free, near-instant refunds that drive customer satisfaction (Temu, 2025). Traditional retail giants are also quietly using AI to optimize returns, balancing cost savings with keeping customers happy (Hadero, 2024). The business impact is significant: billions in potential cost reduction through lower shipping and handling, fewer support tickets (since bots handle the interaction), and improved retention as customers perceive the returns process as easy and fair. As returns are an inevitable part of commerce, especially with the rise of online shopping, AI’s role in managing them intelligently will only grow, contributing to leaner operations and better customer support outcomes (Mitzner, 2025) (Drenik, 2025).

# Hands-On AI with Prototypes and Practical Applications

As we’ve seen so far in the chapters of this paper, AI technologies have greatly developed in the last 5 years and customer support processes adapted alongside it. This accelerated growth has also sparked a fierce competition among legacy players and startups for a slice of the $757 billion market (Precedence Research, 2025).

Although the paper mostly focused on big enterprises and how they’ve used AI to improve their customer support efforts, a great advantage of this fast paced evolution of AI is the democratization of the technology through plenty of choices, reduced cost, ease of use and even open-source or free solutions (Costa, Aparicio, Aparicio, & Aparicio, 2024).

In this chapter I will take a deep dive into the available software, APIs, SDKs and frameworks, both paid and open source, to showcase, through interactive applications, how even small to medium companies can use AI to improve their customer support and experience processes.

## Tech Stack

The way I envisioned this application was as a trip through time, taking the user from the most rudimentary forms of customer support to the most advanced, AI and ML driven, commercially available solutions. Each prototype represents a steppingstone in the evolution of AI in service operations, starting from basic rule-based logic and culminating in multi-modal, reasoning-enabled agents.

To reflect this journey, the application is structured around ten distinct modules, each showcasing a different integration of AI into the customer support workflow:

1. Sentiment Analysis
   1. Analyses the sentiment of a customer message to better understand customer emotions and improve support responses.
2. Intent Analysis
   1. Detects the underlying intent behind customer messages to route and prioritize support requests efficiently.
3. Rule-Based Chatbot
   1. Demonstrate a rule-based chatbot that follows predefined rules to assist customers with common issues.
4. Keyword-Based Chatbot
   1. A more advanced rule-based chatbot that is able to handle text as an input, with basic understanding of words and typos.
5. Embeddings
   1. Creates embeddings from text files for semantic search and similarity analysis.
6. Knowledge Management with Search
   1. Allows users to search through the articles of the knowledge base using either keyword similarity or semantic search with embeddings.
7. Conversation AI Chatbot
   1. Engage with a conversational AI chatbot that can remember past interactions and access tools for enhanced support.
8. Voice AI Chatbot
   1. Interact with a voice AI chatbot that can transcribe audio input and respond with synthesized speech.
9. Speech AI Assist
   1. Evaluates speech in real time and offers feedback on pace, tone, or filler words, helping support agents refine their communication during or after calls.
10. ReACT & MRKL AI Agent
    1. Showcases an advanced React-style agent using the MRKL (Modular Reasoning, Knowledge, and Language) paradigm to combine multiple tools and reasoning steps for complex customer support tasks.

To bring this vision to life, I developed a modular, interactive web application hosted on a public GitHub repository under the name “dissertation-paper-ai-models-for-improving-and-optimizing-customer-support-efficiency” (Panait, 2025).

In order to support the wide range of functionalities explored through the AI prototypes presented in this chapter, a modular, Python-based architecture was developed (Python, n.d.). The system was designed with three key principles in mind: technical accessibility, practical applicability, and flexibility for extension. The chosen technology stack reflects these principles, enabling seamless integration of state-of-the-art AI tools while maintaining ease of use for both technical and non-technical users.

At the foundation of this system lie the OpenAI large language models (LLMs), accessed via the openai Python library (OpenAI, n.d.). These models are orchestrated through the LangChain and LangGraph frameworks, which facilitate the construction of modular reasoning pipelines, memory-based conversation management, and tool augmentation (LangChain, n.d.) (LangGraph, n.d.). These components are essential in enabling advanced functionalities such as context retention, multi-step reasoning, and tool-based interactions, which are explored in prototypes such as the Conversational AI Chatbot and the React & MRKL Agent.

Complementing the generative AI components are several traditional natural language processing (NLP) libraries, including nltk, textblob, scikit-learn, and rake-nltk. These tools are employed for sentiment detection, intent classification, keyword extraction, and text preprocessing, particularly in the early prototypes that simulate earlier stages of AI adoption in customer service (TextBlob, n.d.) (scikit learn, n.d.) (Vishwas B, n.d.).

For semantic search and information retrieval tasks, FAISS (Facebook AI Similarity Search) is used as the primary vector indexing library. FAISS enables the implementation of RAG workflows, allowing the application to search, match, and retrieve relevant documents or knowledge base entries based on semantic similarity rather than exact keyword matching (Facebook Research, n.d.).

To support voice-based interactions and speech processing capabilities, the system incorporates pydub and praat-parselmouth for audio manipulation, alongside ffmpeg as a media processing backend (Jadoul, Thompson, & de Boer, 2018) (Boersma & Weenink, n.d) (FFmpeg, n.d.). These tools are instrumental in enabling prototypes such as the Voice AI Chatbot and the Speech Assist module, where real-time audio input, transcription, and feedback are required.

The application interface is developed using Streamlit, an open-source Python framework for building interactive data applications (Streamlit, n.d.). Streamlit was selected due to its minimal setup requirements, rapid development capabilities, and its ability to support a wide range of media inputs and outputs, including text, audio, and dynamic visualizations. Additional libraries such as plotly, matplotlib, and pandas are integrated to visualize embeddings, similarity graphs, sentiment scores, and user interactions in a user-friendly manner (Plotly, n.d.) (Matplotlib, n.d.) (pandas, n.d.).

Environmental variables, such as API keys, are managed using python-dotenv to ensure security and separation of configuration from the core application logic (Saurabh, n.d.).

The project follows a modular file and folder structure to ensure clarity, maintainability, and ease of extension. The structure is organized as follows:

**Figure 5 - Project's folder structure; Source: own research**

Each prototype is implemented as an independent Streamlit page, accessible via the sidebar navigation. This modular design allows users to interact with each AI model individually while preserving a consistent user experience across the application. It also supports extensibility, as new prototypes or tools can be added without requiring changes to the existing architecture.

To facilitate replication and experimentation, the project is fully containerized and requires minimal setup. It can be deployed in any environment that supports Python 3.8 or later, Git and pip (Python, n.d.) (Python Package Index, n.d.) (Git, n.d.). The setup process is as follows:

1. Clone the repository
   1. git clone https://github.com/TudorPanait/dissertation-paper-ai-models-for-improving-and-optimizing-customer-support-efficiency.git
2. Create and activate a virtual environment
   1. On Windows
      1. python -m venv .venv
      2. .venv\Scripts\activate
   2. On macOS/Linux
      1. python3 -m venv .venv
      2. source .venv/bin/activate
3. Install the required dependencies
   1. pip install -r requirements.txt
4. Define environment variables (OpenAI API key) in a .env file.
5. Launch the application
   1. streamlit run app/Home.py

This deployment process ensures reproducibility for other researchers and practitioners who wish to build upon this work or use it as a foundation for their own AI-driven support tools.

## Prototypes Presentation

This section presents a series of ten progressively complex AI prototypes developed to address various aspects of customer support automation. Each prototype illustrates a different method of integrating artificial intelligence into the support workflow. The intent is not only to showcase functional implementations but also to demonstrate the evolution of AI sophistication and the increasing capacity of these systems to replicate and enhance human-like decision-making in support scenarios.

### Sentiment Analysis

This prototype demonstrates the use of AI techniques to evaluate the emotional tone of customer text inputs. The goal is to assist support teams in identifying customer dissatisfaction, urgency, or positivity to enable more effective, timely, and empathetic responses. Automated sentiment analysis can be particularly impactful in prioritizing tickets, managing support queues, and identifying potential churn risks.

The application is implemented using Python and Streamlit, with two selectable sentiment analysis engines: TextBlob, a lexicon-based NLP tool, and OpenAI GPT, a generative large language model accessed via API. This dual approach allows for a comparative understanding of rule-based versus generative sentiment classification systems.

Upon execution, the user is presented with a text input field and can choose the analysis method from a dropdown menu. Once a message is submitted, the system either:

* Uses TextBlob to return a polarity score (ranging from -1.0 to 1.0) and a subjectivity score (ranging from 0.0 for objective to 1.0 for subjective language), or
* Calls the OpenAI API (gpt-4.1-nano) using a prompt engineered to elicit structured output including the sentiment label (positive/neutral/negative), polarity score, subjectivity classification (low/medium/high), and a natural language explanation of the result.

This implementation provides both analytical and explanatory feedback, which can be valuable in customer service contexts where transparency and reasoning are needed to support decision-making.

The application follows a simple user-driven flow:

**Figure 6 - Application Flow for the Sentiment Analysis Prototype; Source: own research**

Automated sentiment detection is increasingly integrated into modern support platforms. Companies such as Zendesk and Salesforce offer similar features within their AI-powered dashboards. In practice, this functionality can be used to:

* Route urgent issues to senior agents.
* Detect trends in customer dissatisfaction.
* Trigger alerts for reputational risk or crisis management.

Furthermore, the implementation of both lightweight (TextBlob) and advanced (GPT) approaches demonstrates the range of AI complexity that can be applied depending on organizational needs and resource availability.

### Intent Analysis

The demonstration is designed to identify and classify the intent behind customer messages. In the context of customer support, recognizing user intent is essential for accurate ticket routing, effective query resolution, and the delivery of context-appropriate responses. Automating intent classification improves response times and enables personalized assistance, particularly in high-volume support environments.

The prototype utilizes a large language model (LLM) to determine whether a customer input expresses an action, information need, or question. This classification aids in deciding whether a message requires execution of a task (e.g., “I want to cancel my subscription”), provision of factual content (e.g., “I need more details about your pricing”), or further dialog (e.g., “Can I speak to someone about my order?”).

The application is implemented using Streamlit for the user interface and the ChatOpenAI module from LangChain to access OpenAI's GPT model (gpt-4.1-nano). After entering a text sample, the user clicks "Analyze," and the system streams back the model’s structured output, which includes both the detected intent type and a short natural-language explanation justifying the classification.

The model is primed with a system prompt that ensures consistency and interpretability in the model’s output. The LLM then evaluates the semantic structure and context of the input message to classify the user’s goal accordingly.

The logic flow is as follows:

**Figure 7 - Application Flow for the Intent Analysis Prototype; Source: own research**

This pattern ensures both functional output and educational feedback, which can be useful for quality assurance and agent training.

Intent detection is a foundational task in AI-powered support systems. Its application spans various domains:

* In chatbots, intent classification is used to trigger workflows such as refund processing, product lookups, or escalation.
* In ticketing systems, it allows for automatic routing to the appropriate department or escalation tier.
* In voice assistants, it underpins decision-making and dialog flow.

Leading platforms such as Salesforce Einstein, Google Dialogflow, and IBM Watson Assistant rely heavily on intent classification as a core feature. For instance, Dialogflow supports custom intent mappings for hundreds of use cases, from technical support to appointment scheduling.

This prototype simplifies these complex systems by focusing on a minimal yet functional classification framework (Action, Information, Question), which can be extended to multi-label or hierarchical intent classification schemes as needed.

### Rule-Based Chatbot

This module introduces a rule-based chatbot designed to simulate the early, pre-AI era of customer support automation. The goal is to illustrate how structured, decision-tree logic can be applied to handle frequently encountered customer queries such as account issues, order status updates, or the need to escalate a request to a human representative. Rule-based systems, while limited in flexibility, remain a foundational concept in the evolution of intelligent customer service systems.

The chatbot operates using a static set of predetermined conversational paths triggered by user selections. Upon launching the module, the chatbot presents a set of initial options (e.g., *Account Issues*, *Order Status*, or *Talk to a Human Agent*). Each selection dynamically updates the conversation with new prompts and follow-up buttons or input fields, guiding the user through a scripted resolution process.

The application is implemented using the Streamlit framework, leveraging session state to maintain conversation history and to render context-sensitive UI elements such as buttons, text fields, and messages. No natural language understanding is involved; instead, responses are hardcoded and conditionally displayed based on user interaction.

The conversation is governed by a finite-state system. Each button press by the user advances the state of the dialogue, which in turn determines the next set of messages or inputs presented.

The underlying logic follows a strictly deterministic flow:

**Figure 8 - Application Flow for the Rule-Based Chatbot Prototype; Source: own research**

This mimics the experience of interacting with early-generation support systems, where the user was restricted to choosing from a limited menu of options and could not deviate from scripted pathways.

Notably, the chatbot avoids any reliance on AI models or external APIs. This highlights the contrast between deterministic and probabilistic approaches to customer support automation, a comparison that is central to the broader narrative of this dissertation.

Rule-based chatbots were among the first automated solutions deployed in customer support environments. Their simplicity made them relatively easy to implement and control, especially in industries with high volumes of repetitive queries such as telecommunications, e-commerce, and banking. However, their rigidity also poses significant limitations. These systems lacked contextual understanding and often led to user frustration when interactions deviated even slightly from expected input.

In real-world scenarios, early implementations by companies such as Comcast or AT&T in the 2000s exemplified this design, using IVR systems and decision trees to guide users through multi-step menus. While these systems were cost-efficient, they often scored low on customer satisfaction due to their inability to handle ambiguous or non-standard queries.

The prototype developed here serves both as a functioning application and as a historical reference point. It contextualizes the subsequent development of more adaptive and intelligent systems showcased in later modules of this project.

### Keyword-Based Chatbot

This prototype presents a chatbot designed to recognize and respond to predefined keywords or their close variations within user messages. It aims to simulate the transition from rigid, rule-bound automation to systems capable of handling greater variability in user language. In contrast to the previous module, which offered fully deterministic paths, this chatbot introduces a lightweight form of text understanding through approximate matching and optional AI-assisted classification.

Keyword-based systems have historically been used to power early-generation web chat interfaces, offering a more natural feel than button-only workflows while still operating without deep language comprehension. They represent an intermediate step in the evolution of AI-enhanced customer support.

The chatbot uses a curated dictionary of keywords and corresponding responses. When a user types a message, the system attempts to match the input to one of several predefined categories, including common customer support domains such as *account*, *order*, *payment*, and *shipping*. If a clear match is found, the chatbot replies with a canned response associated with that category. If not, it falls back on a default reply that prompts the user to rephrase or select a relevant topic.

The system supports two operational modes:

* Basic Matching Mode: The user input is converted to lowercase and compared against a set of predefined keyword variants using difflib.SequenceMatcher. If a close enough match (similarity ratio > 0.9) is found, the chatbot selects the corresponding response.
* AI-Assisted Mode: If enabled via a toggle, the system sends the user input to a GPT-based model (gpt-4.1-nano) using LangChain. The model is instructed to classify the message into one of ten predefined categories without generating explanations or freeform responses. This hybrid design blends deterministic keyword classification with model-guided input normalization.

The underlying logic of the application follows this generalized flow:

**Figure 9 - Application Flow for the Keyword-Based Chatbot Prototype; Source: own research**

A single message might contain several candidate keywords, but only the best match is selected for response generation. The session state maintains a record of the ongoing dialogue to preserve conversational continuity across turns.

A typical user input such as: “Hey, can you help me with a refund?” will be processed in the following ways:

* Without AI assistance, the system checks the input against similarity thresholds and, detecting “help,” returns: “Sure, I'm here to help. Please describe your issue.”
* With AI assistance enabled, the GPT model receives the message and returns the category “help,” which then maps to the same response.

If no match is found (e.g. the user types a vague or irrelevant message) the system replies with:

* “I'm sorry, I didn't understand that. I can help you with account, order, shipping or payment issues. Can you please rephrase?”

This fallback ensures the interaction remains constructive even in the absence of strong keyword alignment.

Keyword-based systems have long been used in production environments due to their simplicity and low computational cost. However, their brittleness is well-documented. Minor typos, synonyms, or uncommon phrasings can easily lead to misclassification or failure to match. This prototype addresses some of those limitations by incorporating fuzzy string matching and optional AI support to enhance resilience.

Retail companies, customer service bots on e-commerce platforms, and FAQ systems often rely on similar keyword-based logic, particularly in resource-constrained settings or as fallback mechanisms within larger conversational systems.

The significance of this prototype lies in its demonstration of how simple enhancements to legacy logic can bridge the gap toward more intelligent systems without requiring full LLM integration from the outset.

### Embeddings

This prototype demonstrates the creation and visualization of text embeddings, a foundational concept in modern NLP and a critical component of many AI-powered customer support systems. The objective of this module is twofold: first, to explain how textual information can be transformed into mathematical representations suitable for machine learning tasks; and second, to show how this transformation enables semantic similarity, document clustering, and intelligent retrieval capabilities.

Embedding models serve as the bridge between unstructured human language and the structured, numerical format required by most computational systems. By encoding words, sentences, or entire documents into vector form, they allow machines to perform tasks such as similarity search, intent detection, and document classification with high accuracy and contextual awareness.

An embedding is a numerical representation of a text unit (e.g., word, sentence, or paragraph) in a continuous vector space, where semantically similar texts are located near each other. Unlike simple keyword matching, embeddings capture deeper meaning, context, and linguistic relationships. For example, the vectors for "refund" and "money back" would be close in a well-trained embedding space, despite no shared vocabulary.

Embeddings are typically produced by large pre-trained models that have learned statistical patterns from massive text corpora. Models such as OpenAI's text-embedding-3-small or Sentence-BERT are capable of generating fixed-length vectors from variable-length input texts, making them useful for downstream applications like question answering, document search, and recommendation engines.

This module allows the user to generate embeddings from a set of local .txt files stored in the application’s resources/articles directory. The process involves several key steps:

1. **Document Ingestion and Chunking**
   1. Each file is read and then split into overlapping text segments using the RecursiveCharacterTextSplitter. This method ensures that long documents are broken down into manageable chunks, which is important because embedding models typically have a maximum token limit per input.
2. **Vectorization**
   1. Each text chunk is passed through OpenAI's text-embedding-3-small model via the OpenAIEmbeddings class from LangChain. The result is a high-dimensional vector representing the semantic content of the text.
3. **Vector Indexing**
   1. The resulting embeddings are stored in a FAISS (Facebook AI Similarity Search) index, which is optimized for efficient similarity search in large vector datasets. The index is persisted locally in the resources/vectorstore directory.
4. **Statistical Reporting**
   1. A bar chart is rendered to show how many vector chunks were created per source document, providing transparency into the granularity and distribution of the data.
5. **Visualization**
   1. To make the embedding space more interpretable, the vectors are projected into three dimensions using t-SNE (t-distributed stochastic neighbor embedding) and displayed via an interactive 3D scatter plot using Plotly. This allows users to visually explore the clustering of semantically related chunks.

This pipeline offers a complete overview of how raw textual data can be transformed and operationalized in an AI workflow:

**Figure 10 - Application Flow for the Embeddings Prototype; Source: own research**

Embeddings are increasingly central to intelligent search and retrieval functions in customer support environments. For example, if a user asks “How can I change my delivery address?”, a traditional keyword-based system might miss the match if the support document uses “update shipping information.” An embedding-based retrieval system, however, would correctly identify the semantic similarity and return the relevant help article.

Companies such as Amazon, Zendesk, and Google Cloud have integrated embedding-based knowledge systems into their support platforms. These systems support not only document lookup but also agent-assist tools, enabling representatives to access contextual information faster and more accurately.

This prototype showcases the underlying mechanics of such systems in a simplified but fully functional form. It sets the stage for the following modules, which will build on this embedding foundation to deliver dynamic, retrieval-augmented conversation and search functionalities.

### Knowledge Management with Search

The application implements a practical knowledge management system designed to demonstrate how organizations can enable intelligent self-service and agent-assist features through search functionality. At its core, this module enables users to query a document repository using either keyword matching or embedding-based semantic retrieval.

The system simulates a common customer support use case in which users (or support agents) need to locate relevant information across a corpus of support articles, policy documents, or product guides. The ability to retrieve accurate and contextually relevant content from internal knowledge bases significantly enhances both efficiency and customer satisfaction.

The knowledge base in this prototype consists of plain-text files stored in a predefined local directory. Users interact with the system by entering a search query and selecting one of two retrieval methods:

1. Keyword-Based Search
   1. This method performs a literal string match. It reads each document and returns any file that contains the exact search term (case-insensitive). The matched keyword is then visually highlighted within the result to enhance readability. While this approach is computationally inexpensive and interpretable, it suffers from lexical rigidity as synonyms or related phrases will not yield results unless they match the query text exactly.
2. Embedding-Based Semantic Search
   1. This method leverages vector-based similarity by comparing the semantic representation of the search query against an indexed vector store built from the documents. The system uses OpenAI’s text-embedding-3-small model to embed the query, and then uses FAISS to identify the top-k closest matches in the embedding space.

The logic flow of the prototype is as follows:

**Figure 11 - Application Flow for the Knowledge Management with Search Prototype; Source: own research**

The embedding-based approach supports nuanced understanding of language, returning results that may not share vocabulary with the original query but express similar intent or meaning.

For example, a user typing “how do I get a refund” might retrieve articles containing “return policy” or “money-back guarantee”, an outcome keyword matching alone would not achieve.

Embedding-powered document retrieval is now a standard feature in many enterprise customer support platforms. Tools like Intercom's Fin, Salesforce Einstein Search, and Zendesk AI use similar architectures to deliver contextually relevant knowledge base articles during live chats or email responses.

In smaller organizations, embedding-based document search can be used to reduce the burden on human agents, especially in ticket triage or escalation workflows. Instead of manually searching through documentation, AI systems can suggest helpful articles in real time.

By including both retrieval strategies in this prototype, the application underscores the contrast between deterministic and semantic retrieval. This makes the tool especially valuable for illustrating to business stakeholders how more advanced search systems can improve the accuracy and relevance of internal knowledge access.

### Conversational AI Chatbot

We now advance to a fully-fledged conversational AI assistant that integrates memory, context awareness, RAG, and a defined personality. It represents a significant leap forward from earlier rule or keyword-based systems by providing an adaptable, natural dialogue interface designed to simulate human-like customer service. The assistant can retain context across multiple exchanges, incorporate external knowledge dynamically, and respond with tone and nuance, all of which contribute to a more satisfying user experience.

In the evolution of AI for customer support, this chatbot illustrates the point where automation becomes not only functional but also interpersonal, reflecting a maturity in AI systems that allows for scalable yet personalized service interactions.

The chatbot is implemented using the ChatOpenAI LLM interface from LangChain and incorporates the following architectural features:

1. **Session Memory**
   1. Chat history is maintained across turns via Streamlit’s session state. This enables the system to recall previous user messages and model responses, forming the basis for multi-turn dialogue. The model is explicitly prompted with this conversational context at each turn to simulate memory retention.
2. **Retrieval-Augmented Generation (RAG)**
   1. When a user query is received, the system performs a similarity search over a vectorized knowledge base using FAISS and OpenAI’s text-embedding-3-small model. The retrieved documents are appended to the conversation history and provided as contextual background for the model's next response. This allows the chatbot to respond with information it was not trained on, while ensuring up-to-date and company-specific relevance.
3. **Prompt Engineering and Personality Conditioning**
   1. The assistant is guided by a detailed system prompt (AI\_PROMPT) that defines its tone, capabilities, and limitations. It is framed as *Tudor*, a helpful, and professional support AI for a fictional online electronics store. The prompt includes behavior instructions, fallback guidelines, and examples of keyword themes without enforcing rigid intent detection.

Each user message is processed through the following logic chain:

**Figure 12 - Application Flow for the Conversational Chatbot Prototype; Source: own research**

Conversational AI systems with context awareness and retrieval mechanisms are rapidly becoming the backbone of modern customer service. Enterprise solutions like ChatGPT Enterprise, Klarna’s AI assistant, and Google Cloud's Agent Assist employ similar architectures to offer not only real-time help but also real-time learning where new information can be injected into the system without retraining the underlying model.

This prototype demonstrates that such capabilities can be achieved without proprietary infrastructure, using open tools and APIs. It also highlights how modular components (embedding engines, vector stores, LLMs, and UI frameworks) can be composed into cohesive, production-ready systems.

From an implementation roadmap standpoint, this prototype serves as a blueprint for SMEs interested in deploying AI assistants that go beyond transactional automation. The use of RAG allows the system to remain lightweight while being deeply informative. Meanwhile, conversational memory contributes to a more fluid, human-like interaction that can drive customer satisfaction and retention.

As such, this chatbot marks the beginning of AI systems that listen, remember, and adapt, a milestone in the evolution explored throughout this chapter.

### Voice AI Chatbot

Building upon the capabilities demonstrated in the previous modules, this prototype explores the integration of voice input and output into a conversational AI agent. The goal is to simulate a more human-like, multimodal interaction that mirrors real-world call center experiences or virtual voice assistants.

Voice interfaces represent the next frontier in AI-mediated customer service, allowing for hands-free, natural language interactions that improve accessibility and efficiency. While the current prototype is designed for demonstration purposes using standard transcription and text-to-speech pipelines, it is conceptually aligned with the architecture of real-time AI voice platforms such as Retell, Vapi, and emerging offerings like OpenAI’s Realtime API for gpt-4o.

This chatbot extends the conversational model by replacing the traditional text input-output loop with speech-to-text (STT) for capturing user input and text-to-speech (TTS) for delivering AI-generated responses. It does not retain memory or utilize retrieval-augmented generation (RAG), thereby functioning as a stateless assistant capable of single-turn interactions.

The process is as follows:

1. Voice Capture
   1. The user records a short audio message directly within the Streamlit interface using the st.audio\_input() component.
2. Transcription
   1. The audio is transcribed into text using OpenAI's gpt-4o-mini-transcribe model, which provides a fast, lightweight transcription service suitable for short messages.
3. Response Generation
   1. The transcribed query is sent to an OpenAI language model (gpt-4.1-nano) to generate a natural language response appropriate to the inferred intent and content of the original message.
4. Speech Synthesis
   1. The model’s text output is passed to OpenAI’s gpt-4o-mini-tts service, which returns a synthesized voice file in .mp3 format. The assistant's voice is configured with expressive tone controls, matching the sentiment of the response.
5. Playback
   1. The generated audio response is automatically played back in the user interface, completing the voice-to-voice interaction loop.

**Figure 13 - Application Flow for the Voice AI Chatbot Prototype; Source: own research**

This process highlights the integration of multiple AI modalities (audio input processing, language understanding, and voice synthesis) into a cohesive, real-time user experience.

This prototype operates in a stateless mode, meaning it does not remember previous turns of the conversation. This design choice simplifies the implementation and aligns with use cases where responses are limited to one-off queries (e.g., checking shipping status, asking for store hours). However, it also restricts the chatbot’s ability to handle complex, multi-step requests or personalize its responses based on conversation history.

In production settings, real-time streaming capabilities, now available via OpenAI’s Realtime API, allow for sub-300ms latency in transcription and speech synthesis, enabling full-duplex conversations. Platforms like Retell and Vapi have already begun offering commercial solutions with streaming ASR and TTS, integrated memory, and back-end orchestration layers. These services are revolutionizing voice-based automation in industries ranging from e-commerce to insurance.

Although this prototype does not replicate real-time streaming, it serves as a compelling demonstration of the feasibility of voice-AI integration using readily available models and frameworks. It also opens the door to future modules with bidirectional voice memory, emotion detection, and voice-personalized recommendations.

### Speech AI Assist

The prototype introduces a speech analysis and coaching tool designed to assist customer support agents in improving the clarity, tone, and pacing of their verbal communication. Effective spoken delivery plays a pivotal role in customer satisfaction, particularly in voice-based support channels such as phone calls, live chat escalations, and voicebots. This tool enables agents to receive immediate, objective feedback on their vocal performance, which is traditionally available only through post-call reviews or manual coaching.

Unlike the previous prototypes focused on customer-facing automation, this module addresses agent enablement, offering a practical example of AI-powered internal tooling for quality assurance and skills development.

The Speech Assist tool evaluates the acoustic and prosodic features of a user's voice recording and offers performance feedback based on key vocal metrics. These include pitch variation, speech rate, intensity, harmonic clarity, and pause frequency, factors that significantly influence how speech is perceived in professional support environments.

Upon recording, the user's voice is automatically transcribed and analyzed. The system then generates structured feedback, flagging potential areas for improvement in vocal delivery.

The workflow consists of four main stages:

1. Audio Recording and Preprocessing
   1. The user provides a voice sample using the Streamlit st.audio\_input() component. The audio is temporarily stored as a .wav file using Python’s tempfile module.
2. Acoustic Feature Extraction
   1. The file is processed using parselmouth, a Python wrapper for the Praat speech analysis toolkit. Key acoustic features extracted include:
      1. Mean pitch and pitch range (intonation)
      2. Mean intensity (loudness)
      3. Harmonic-to-noise ratio (HNR) (vocal clarity)
      4. Voiced segment ratio (speaking vs. silent time)
      5. Pause frequency (smoothness of delivery)
3. Speech Rate Estimation
   1. The recorded sample is transcribed using OpenAI’s gpt-4o-mini-transcribe model. The total word count is then divided by the audio duration to estimate words per minute (WPM).
4. Feedback Generation
   1. Based on pre-established heuristics, the system evaluates the extracted features and generates personalized recommendations. For example:
      1. Monotone delivery is flagged if pitch range is too narrow.
      2. Overly fast speech is discouraged if WPM exceeds 170.
      3. High pause frequency is flagged as a disruption to fluency.

This end-to-end loop produces an interactive coaching session in under one minute, providing agents with actionable, data-driven insights through the following application logic:

**Figure 14 - Application Flow for the Speech AI Assist Prototype; Source: own research**

### ReACT & MRKL AI Agent

The final prototype in this chapter showcases a state-of-the-art conversational AI agent that integrates both reasoning and tool execution, enabling it to dynamically plan, act, and respond in a structured, interpretable manner. Known as a ReACT + MRKL agent, this system goes beyond static chat capabilities by using tools to fetch information, process external queries, and refine its answers iteratively.

This prototype serves as the culmination of the AI evolution journey presented throughout the chapter, demonstrating how language models can now think and act in a modular, intelligent way—unlocking new levels of automation in customer support.

ReACT (Reasoning and Acting) is an emerging paradigm in AI agent development in which large language models (LLMs) are prompted to display intermediate reasoning steps before taking actions. Rather than outputting final answers directly, the agent first explains its thinking (“Thought”), chooses an appropriate action (“Action”), provides an input to that action (“Action Input”), and then processes the result (“Observation”). This structured chain is repeated until the model concludes that it has sufficient information to return a final response.

This approach enhances transparency and trust, as users can observe the AI’s logic step-by-step. It also increases reliability, as intermediate observations can be used to verify or correct assumptions before a final answer is given.

MRKL (pronounced "miracle") stands for Modular Reasoning and Knowledge Learning. This architecture involves breaking down a chatbot into interchangeable “modules” or tools, each capable of handling a specific kind of task (e.g., web search, calculations, database queries, document retrieval). The agent routes parts of a user’s query to the relevant tools as needed.

By integrating ReACT with MRKL, an AI system is able to reason about a user’s question, call on tools to supplement its capabilities, and synthesize an answer using both pre-trained knowledge and external, up-to-date information.

In this prototype, the ReACT + MRKL agent is powered by OpenAI’s gpt-4.1-mini model and orchestrated using LangChain’s agent framework. The architecture includes the following components:

* **LLM Reasoning Core**
  + The language model is instructed with a prompt template that guides it through the ReACT framework, including steps for thinking, acting, and finalizing its output.
* **Tool Integration**
  + The agent is equipped with a single tool: search\_knowledge\_base, a RAG-style function that performs vector similarity search over a FAISS index of customer support articles. While only one tool is used here for demonstration, the system is built to scale as tools like web search, API calls, or calculators could be easily added.
* **Streaming Response and Error Handling**
  + Responses are streamed live into the Streamlit interface. Errors in tool parsing or output are handled gracefully to maintain continuity.

The agent is prompted to work through the following structured logic:

**Table 1 - ReACT Agent Reasoning and Execution Flow; Source: own research**

|  |  |
| --- | --- |
| Step | Description |
| Question | The original user input, possibly containing multiple sub-questions. |
| Thought | The agent’s internal reasoning about what information or tools it needs. |
| Action | The specific tool the agent decides to use (e.g., a search or API call). |
| Action Input | The argument or query provided to the selected tool. |
| Observation | The output returned by the tool, used as evidence or context. |
| Thought (updated) | The agent reassesses what it knows after reviewing the tool's output. |
| Final Answer | A comprehensive, markdown-formatted response delivered to the user. |

In ReACT-based agents, the Action–Action Input–Observation cycle is repeatable up to three times per prompt. This design enables the agent to perform multi-step reasoning and retrieval when the initial tool call does not yield sufficient information. For instance, the agent may:

* Search multiple knowledge segments separately
* Compare results from different queries
* Refine its query based on intermediate findings

After the third cycle, the model is instructed to finalize its answer regardless of the completeness of the information, ensuring that users always receive a response without indefinite computation.

This looping capability allows the agent to simulate a human-like problem-solving process by thinking, acting, gathering information, and iterating, before presenting a well-grounded answer.

ReACT & MRKL agents represent a paradigm shift in AI-driven customer service. Compared to legacy systems, they offer a range of benefits:

* Compositional Understanding
  + The agent can accurately handle multi-intent queries by segmenting and addressing each part individually, a task traditional chatbots cannot execute reliably.
* Tool-Aware Actionability
  + The model does not rely solely on its training data but can invoke specific tools such as search, databases, or APIs—to supplement or verify its answers.
* Self-Explanation
  + The visible “Thought–Action–Observation” chain fosters interpretability and user trust, allowing even non-technical stakeholders to audit the decision-making process.
* Fail-Safe Structure
  + Even when the first action does not return relevant results, the model can retry with a different approach, ensuring resilience in complex workflows.

Traditional “one-shot” chatbots struggle with complex or multi-intent prompts, often responding only to the most salient part of the message or ignoring sub-questions entirely. By contrast, the ReACT agent is capable of decomposing such inputs into manageable sub-questions, processing each independently (including taking actions), and then synthesizing a cohesive answer.

Consider the following complex user input:

*"I am very sad. The courier broke my product during delivery, and I have a bad product. Can I still return it? Or is it included under warranty? I would like to send you a picture so you can investigate — is there an email address for these kinds of issues?"*

This query contains at least four distinct intents:

1. An emotional expression (“I am very sad”) suggesting a tone-aware or empathetic response.
2. A factual inquiry about return eligibility given a damaged product.
3. A warranty-related follow-up to the same issue.
4. A procedural request about sending photographic evidence via email.

The ReACT agent would approach this in a structured fashion:

**Table 2 - ReACT & MRKL Agent Behavior in a Multi-Prompted User Query; Source: own research**

|  |  |
| --- | --- |
| Step | Agent Behavior |
| Thought | The user is upset and is asking about a damaged product. Multiple questions are present. |
| Question 1 | Can the product be returned if damaged during delivery? |
| Action | Search knowledge base for “return damaged product delivery” |
| Observation | Returns are accepted within 30 days for items damaged in transit, with photo proof. |
| Question 2 | Is it also covered under warranty? |
| Action | Search knowledge base for “warranty for damaged product” |
| Observation | Warranty does not cover shipping damage, only manufacturer defects. |
| Question 3 | Is there an email address where the customer can send photos? |
| Action | Search knowledge base for “email for support photos” |
| Observation | support@abcstore.com is listed for escalated claims and image submissions. |
| Final Answer | The agent now responds with: - Expressing empathy - Clarifying return policy - Explaining warranty limitations - Providing the correct email address. |

This level of decomposition and targeted action would not be possible in earlier rule-based or static models. Importantly, the ReACT framework allows the agent to pause, think, gather, and respond accurately across multiple knowledge areas, even when the user input is emotionally charged, informal, or poorly structured.

In commercial applications, similar agents are already being employed by platforms like Klarna’s AI assistant, Perplexity AI, and OpenAI GPT-4o, where reasoning transparency and multi-turn planning are central to performance. This prototype demonstrates how such state-of-the-art design patterns can be replicated using open-source frameworks, modular prompts, and scalable APIs, making them accessible even to SMEs.

## Overview on the Prototypes Results

This chapter has presented a progressive, hands-on exploration of artificial intelligence in customer support, from early-stage deterministic systems to contemporary, reasoning-enabled AI agents. The ten prototypes showcased here represent distinct stages in the evolution of support automation, mapped not only to technological advances but also to increasing degrees of intelligence, flexibility, and personalization.

What began with rule-based logic and keyword detection evolved into capabilities such as semantic understanding, memory retention, tool orchestration, and multimodal voice interaction. Each prototype was designed to be modular, transparent, and practically replicable using open tools and APIs, offering a blueprint for AI adoption, especially for resource-constrained SMEs looking to modernize their support infrastructure.

The summary table below provides a comparative overview of each prototype based on its functionality, underlying technology, intended use case, and its general feasibility for implementation in small to medium-sized enterprises.

**Table 3 - Summary of AI Prototypes for Customer Support; Source: own research**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Prototype | Functionality | Core Tools | Primary Use Case |
| 1 | Sentiment Analysis | Detect emotional tone in customer messages | TextBlob, OpenAI GPT, NLTK | Escalation triage, emotion-aware routing |
| 2 | Intent Analysis | Classify user intent (action, info, question) | OpenAI GPT, LangChain | Ticket routing, auto-tagging |
| 3 | Rule-Based Chatbot | Fixed logic paths via button/menu navigation | Streamlit session state | FAQs, basic support automation |
| 4 | Keyword-Based Chatbot | Flexible keyword matching with fuzzy logic | difflib, OpenAI GPT | General inquiries, fallback logic |
| 5 | Embeddings | Semantic vectorization of support articles | OpenAI Embeddings, FAISS, LangChain | Similarity search, document clustering |
| 6 | Knowledge Search | Keyword and embedding-based knowledge retrieval | FAISS, OpenAI, file I/O | Self-service portals, agent-assist |
| 7 | Conversational AI Chatbot | Multi-turn dialogue with memory + RAG | ChatOpenAI, FAISS, LangChain | Dynamic chat, case-based reasoning |
| 8 | Voice AI Chatbot | Voice input/output via STT and TTS | gpt-4o-mini-transcribe, TTS, Streamlit audio | Voice support simulation |
| 9 | Speech Assist | Analyze agent speech for clarity and pace | parselmouth, OpenAI STT | Agent coaching, QA feedback |
| 10 | ReACT + MRKL Agent | Tool-using agent with reasoning and RAG integration | LangChain, FAISS, ChatOpenAI, Custom Tools | Complex queries, autonomous task resolution |

Lower-complexity prototypes (1–4) are easy to implement and require minimal infrastructure, making them ideal entry points for small businesses. However, their functionality is limited in adaptability and personalization.

Embedding-based tools (5–6) offer significantly improved contextual understanding without requiring the full architecture of LLMs or agents. These provide a realistic upgrade path for SMEs seeking more intelligent automation.

Prototypes 7–9 introduce conversational memory, speech synthesis, and performance coaching, features that emulate human interaction and are increasingly expected in modern CX environments. These are more resource-intensive but deliver high experiential value.

The final prototype (10) showcases the emerging frontier of AI in customer service, which is represented by modular, reasoning-capable agents that can independently decompose problems, use tools, and formulate complete answers. While promising, these systems are better suited for larger organizations or pilot programs, given their technical complexity and operational requirements.

This chapter has emphasized that meaningful AI adoption in customer support is not exclusive to large enterprises. By modularizing functionalities and leveraging public APIs, even small organizations can begin integrating AI into their workflows at a controlled pace. Whether through automated sentiment monitoring, knowledge-based chatbots, or real-time speech feedback, each of these prototypes provides a viable, scalable entry point into modernizing customer engagement.

More importantly, this progression illustrates that AI adoption is not a binary leap but a strategic continuum where each stage prepares the foundation for more advanced, impactful systems to follow.

# Discussion

This chapter reflects on the findings obtained from both the secondary research and the practical prototype development presented throughout this paper. It aims to bridge the gap between theoretical insights, real-world AI deployments in customer support, and hands-on technological experimentation. Specifically, it revisits the guiding research questions and offers evidence-based perspectives informed by the case studies and the functional applications developed. By doing so, it draws a cohesive picture of how AI is actively reshaping the customer support landscape, both strategically and operationally.

## What are the most common and high impact use cases of AI in customer support?

The analysis of case studies revealed that AI applications in customer support tend to cluster around a few recurring, high-impact categories. AI-powered chatbots represent the most prominent use case, acting as the front line for handling large volumes of customer queries. Solutions such as Bank of America’s Erica, Vodafone’s TOBi, and Klarna’s AI assistant show that conversational agents can resolve up to 70% of support requests independently, significantly reducing response times and operational costs.

Another impactful domain is internal knowledge management, where AI systems assist human agents by surfacing relevant information at the right time. Implementations such as British Telecom’s Albert or Banca Transilvania’s David demonstrate tangible improvements in agent productivity, first-contact resolution, and consistency of service. Additionally, AI has proven effective in voice-based support settings, enabling transcription, sentiment detection, and real-time agent guidance. In return and refund operations, companies like Temu and Amazon have embraced AI to automate eligibility assessments and streamline customer interactions, showing substantial efficiency gains and customer satisfaction improvements.

All of these use cases were intentionally mirrored in the prototype suite developed during this research, confirming their practical relevance and adaptability even in resource-constrained environments. From keyword-triggered bots to memory-augmented chat agents and voice-enabled support tools, the prototypes provide a tangible demonstration of how each of these use cases can be implemented and extended.

## How are different AI models (e.g., LLMs, embeddings, speech recognition) integrated into actual support workflows?

The real-world case studies and prototype development show a clear evolution in how AI models are being operationalized within customer service ecosystems. LLMs, such as OpenAI’s GPT, are now widely deployed for generating human-like responses, performing intent classification, and guiding multi-turn dialogues. Embedding models are equally central, particularly in RAG systems that power intelligent document search and contextual responses. These models have found commercial application in platforms like Intercom’s Fin, Zendesk’s AI agents, and Salesforce’s Einstein assistant.

In the prototypes developed as part of this paper, embeddings and vector databases (e.g., FAISS) were used to semantically index and retrieve knowledge base content, enabling chatbots to answer questions based on actual documentation rather than relying solely on pre-trained knowledge. Similarly, OpenAI’s transcription and text-to-speech APIs were used to prototype voice-based assistants, showcasing a streamlined integration of ASR and generative language models into the support workflow.

These technical components are no longer siloed; they are orchestrated together to form dynamic support systems capable of handling multiple modalities and interaction types. This layered integration is becoming the new standard in modern customer support automation.

## What practical challenges and benefits have been observed in implementations?

The practical benefits of AI in customer support are evident and quantifiable. Companies reported up to 50–70% cost savings in chat operations, significant improvements in customer satisfaction (as reflected in Net Promoter Scores), and notable gains in operational efficiency. For instance, Bank of America’s Erica handled over 2 billion interactions with a resolution time of under 44 seconds for 98% of cases. Vodafone’s SuperTOBi achieved a 45% increase in first-contact resolution post-implementation.

However, these successes do not come without challenges. One common limitation is the inability of current AI systems to consistently handle emotionally nuanced or contextually complex situations. Klarna’s experience, where over-automation led to a noticeable decline in service quality, illustrates the risks of relying solely on AI. Customers expect empathy, flexibility, and the option to interact with human agents—factors that AI cannot always deliver. The lack of a seamless handoff mechanism between AI agents and humans was also a recurring pain point.

These insights were further reinforced through the prototypes. Simpler bots (e.g., rule-based or keyword-triggered) were quick to implement but easily confused by ambiguity or spelling errors. Generative chatbots showed higher flexibility and fluidity, but also required careful prompt engineering, model tuning, and error handling to avoid incorrect or irrelevant answers. Moreover, voice-enabled assistants added another layer of complexity, such as transcription inaccuracies and latency in response generation.

The combined evidence suggests that while AI introduces clear advantages, it must be implemented thoughtfully, with appropriate guardrails and fallback strategies to prevent quality erosion in sensitive cases.

## How can modular tools be assembled to support intelligent, multi-modal, and context-aware support systems?

Both the prototypes and case studies reveal a strong preference for modular AI systems. Rather than relying on monolithic platforms, modern support solutions are built by combining specialized tools: LLMs for reasoning and response generation, vector databases for semantic retrieval, ASR models for voice input, and external APIs for system actions.

The prototypes in this research reflect this modularity. Using Python and frameworks like LangChain and Streamlit, AI functionalities were split into reusable components each capable of addressing a specific part of the support journey. For example, a sentiment analysis module could be used as a standalone escalation filter or integrated into a larger conversation flow. Similarly, embeddings could power both direct search tools and retrieval-augmented chatbots.

This architecture mirrors what leading platforms are doing. Klarna, Intercom, and Zendesk all use combinations of internal LLMs, customer knowledge graphs, intent classifiers, and escalation APIs to construct comprehensive, context-aware support systems. These systems are not rigid scripts, but rather fluid orchestration layers where AI agents can pull in the right tools at the right time.

This modularity enables both scalability and flexibility. Companies can adopt AI step-by-step, starting with small interventions (e.g., automated FAQs or agent-assist tools) and later evolving into more autonomous, tool-using agents like the one demonstrated in the ReACT + MRKL prototype.

## How can support operations progress from simple automation to more sophisticated agentic AI systems?

The evolution path uncovered in this research is both technical and strategic. Organizations typically begin with rule-based or keyword-driven bots that automate repetitive tasks. These systems provide immediate efficiency gains but are limited in scope and adaptability. As needs grow, teams often adopt vector-based search and retrieval to improve the quality and depth of responses, followed by LLMs for more fluid dialogue and personalization.

The final frontier lies in agentic AI which are systems that not only respond but actively reason, decide, and act. The ReACT + MRKL prototype developed in this research illustrates this potential. Unlike traditional bots, it can deconstruct a complex user message, retrieve relevant documents from a knowledge base, and synthesize a coherent, well-reasoned response. This mirrors the capabilities of modern AI copilots being deployed in high-performing service teams.

However, this sophistication comes with increased responsibility. As AI gains autonomy, issues such as interpretability, hallucination risks, data privacy, and brand alignment become more critical. Therefore, transitioning from basic automation to agentic intelligence requires not only technical expertise, but also governance frameworks and cross-functional collaboration between support, product, legal, and AI teams.

What is encouraging is that the tools and methodologies to make this transition are increasingly accessible. As demonstrated in this dissertation, even a small, modular tech stack can support the design and deployment of AI-driven support agents with advanced capabilities.

# Conclusion

The goal of this dissertation was to explore how AI models can be leveraged to improve and optimize customer support efficiency. In an increasingly digital and customer-driven world, the expectations for fast, consistent, and personalized service have never been higher. At the same time, businesses are under growing pressure to reduce operational costs and scale service operations without compromising quality. Against this backdrop, AI has emerged not only as a viable solution, but as a transformative force, capable of reshaping how companies interact with their customers.

Through a combination of literature review, real-world case studies, and hands-on prototype development, this research has mapped the evolution of AI in customer service, identified key use cases, and demonstrated the practical feasibility of implementing such technologies across different scales of operation. By adopting a narrative and exploratory methodology, the study has provided both strategic insights and technical illustrations of how AI models, ranging from rule-based systems to reasoning-enabled agents, can be used to streamline, enhance, and augment support workflows.

The findings show that AI applications in customer support generally fall into five impactful domains: AI chatbots for high-volume interaction handling; knowledge management systems that empower agents with real-time access to contextual information; voice support and transcription tools that optimize live conversations; automation in returns and refund management; and predictive analytics that enable proactive customer engagement. Case studies from organizations such as Bank of America, Vodafone, Klarna, Alibaba, and H&M illustrated that these technologies, when properly integrated, can significantly improve efficiency, reduce costs, and elevate customer satisfaction metrics.

However, this dissertation also surfaced the limitations of AI in emotionally complex or ambiguous situations, where human intuition, empathy, and discretion are still indispensable. Over-reliance on automation without a fallback to human support can degrade customer experience, as highlighted in the case of Klarna’s temporary service dip. Therefore, the study advocates for a hybrid model where AI acts as an intelligent copilot, augmenting human capabilities rather than replacing them.

The ten AI prototypes developed as part of this research served a dual purpose. First, they validated that the use cases identified in the literature and case studies can be practically implemented using modern AI tools. Second, they showcased a clear technological progression from simple rule-based automation to advanced multi-modal and tool-using agents. These prototypes were built using accessible, open-source, and modular technologies, highlighting that even small and medium-sized businesses can experiment with and adopt AI solutions without needing massive resources.

In addition to confirming the benefits of AI adoption, the research emphasizes that success depends not just on technology, but also on thoughtful orchestration, ethical safeguards, and continuous monitoring. As AI systems become more autonomous and human-like in their behavior, ensuring alignment with organizational values, compliance standards, and customer expectations becomes even more critical.

Looking ahead, the direction of AI in customer support is unmistakably moving toward agentic systems capable of understanding, reasoning, acting, and interacting across multiple modalities. Tools like GPT-4o, multimodal LLMs, and advanced orchestration frameworks such as LangChain and MRKL are already shaping the future of how businesses will handle customer interactions. As these systems evolve, companies must stay agile, adopting a modular, iterative, and human-centric approach to innovation.

This dissertation demonstrates that AI is not a distant technological aspiration but a practical, scalable asset that can be integrated today. Whether the goal is to automate FAQs, assist live agents, analyze speech patterns, or resolve customer queries autonomously, AI offers a spectrum of solutions that align with different organizational needs and maturity levels. More importantly, this research underscores that meaningful AI implementation is not a one-time initiative, but a continuous journey, where organizations must balance efficiency with empathy, automation with authenticity, and innovation with impact.

Ultimately, AI’s role in customer support is not to replace the human experience, but to enhance it, by making service faster, smarter, and more aligned with the needs of the modern customer.

# Bibliography

Aggarwal, A. (2023, October 31). *How ‘Erica- A Conversational AI Agent’ helped power a 19% spike in earnings at Bank of America.* Retrieved from FluidAI: https://www.fluid.ai/blog/how-erica-a-conversational-ai-agent-helped-power-a-19-spike-in-earnings-at-bank-of-america

Ahern, G. (n.d.). *The 2022 Complete Guide to Chatbots.* Retrieved 2025, from Ometrics: https://www.ometrics.com/blog/complete-guide-chatbots-case-studies/

Alibaba. (2025, February 20). *Alibaba: Scale Up AI Investments in Three Areas Over the Next Three Years.* Retrieved from Alibaba: https://www.alibabagroup.com/en-US/document-1829422257118642176

Amazon. (n.d.). *Amazon Lex - AI Chat Builder.* Retrieved 2025, from Amazon AWS: https://aws.amazon.com/lex/

Amazon. (n.d.). *Unleash AI to transform every customer interaction.* Retrieved 2025, from Amazon AWS: https://aws.amazon.com/connect/

Andrade, I. M., & Tumelero, C. (2021). Increasing customer service efficiency through artificial intelligence chatbot. *Revista de Gestão*.

Arora, N., Ensslen, D., Fiedler, L., Liu, W. W., Robinson, K., Stein, E., & Schüler, G. (2021, November 12). *The value of getting personalization right—or wrong—is multiplying.* Retrieved from McKinsey & Company: https://www.mckinsey.com/capabilities/growth-marketing-and-sales/our-insights/the-value-of-getting-personalization-right-or-wrong-is-multiplying

Ayyad, K. (2025, March 10). *A Brief History of AI in Customer Support.* Retrieved from Teammates: https://teammates.ai/blog/history-of-ai-in-customer-support

Balto. (n.d.). *Contact Center AI Software.* Retrieved 2025, from Balto: https://www.balto.ai/

Banca Transilvania. (2020, September). *Chatbots of BT.* Retrieved from Banca Transilvania: https://www.bancatransilvania.ro/news-files/chatbots-of-bt-2020/whitepaper\_chatbots\_of\_bt\_septembrie\_2020.pdf

Bank of America. (2024, April 08). *BofA’s Erica Surpasses 2 Billion Interactions, Helping 42 Million Clients Since Launch.* Retrieved from Bank of America: https://newsroom.bankofamerica.com/content/newsroom/press-releases/2024/04/bofa-s-erica-surpasses-2-billion-interactions--helping-42-millio.html

Bank of America. (2025, April 08). *AI Adoption by BofA’s Global Workforce Improves Productivity, Client Service.* Retrieved from Bank of America: https://newsroom.bankofamerica.com/content/newsroom/press-releases/2025/04/ai-adoption-by-bofa-s-global-workforce-improves-productivity--cl.html

Bank of America. (2025, February 24). *Digital Interactions by BofA Clients Surge to Over 26 Billion, up 12% Year-Over-Year.* Retrieved from Bank of America: https://newsroom.bankofamerica.com/content/newsroom/press-releases/2025/02/digital-interactions-by-bofa-clients-surge-to-over-26-billion--u.html

Beam. (2024, April 04). *Automating Returns, Refunds, and Replacements Using AI Agents.* Retrieved from Beam: https://beam.ai/articles/returns-refunds-replacements

Billan, M. (2025, May 20). *Airline Chatbot: Benefits, Use Cases & Examples.* Retrieved from Master of Code: https://masterofcode.com/blog/airline-chatbot

Blackader, B., Buesing, E., Amar, J., Raabe, J., Mehndiratta, M., & Gupta, V. (2025, March 19). *The contact center crossroads: Finding the right mix of humans and AI.* Retrieved from McKinsey: https://www.mckinsey.com/capabilities/operations/our-insights/the-contact-center-crossroads-finding-the-right-mix-of-humans-and-ai

Boersma, P., & Weenink, D. (n.d). *Praat: doing phonetics by computer.* Retrieved 2025, from Praat: https://praat.org/

Brand Finance. (2024). *Banking - The annual report on the most valuable and strongest banking brands.* Retrieved from Brand Finance: https://brandirectory.com/reports/banking/2024

Brix, A. C. (2025, May 13). *Postal System.* Retrieved from Britannica: https://www.britannica.com/topic/postal-system

Choice Cheapies. (2024, April 26). *Temu refund tip.* Retrieved from Choice Cheapies: https://www.cheapies.nz/node/46321

Chudleigh, S. (2025, May 14). *Key Chatbot Statistics for 2025: Perceptions, Market Growth, Trends.* Retrieved from botpress: https://botpress.com/blog/key-chatbot-statistics

Chui, M., Kamalnath, V., & McCarthy, B. (2020). *An executive’s guide to AI.* Retrieved from McKinsey & Company: https://www.mckinsey.com/capabilities/quantumblack/our-insights/an-executives-guide-to-ai

Cogito. (n.d.). *Powering the Teams Behind Great Service.* Retrieved 2025, from Cogito: https://cogitocorp.com/

Connectys. (2025, May 1). *Scaling Customer Support: How to Prepare Your CX Team for Peak Season Challenges.* Retrieved from Connectys: https://www.conectys.com/blog/posts/scaling-customer-support-how-to-prepare-your-cx-team-for-peak-season-challenges/

Convin. (2025, January 13). *AI Coaching Tools for Real-Time Call Center Agent Training.* Retrieved from Convin: https://convin.ai/blog/ai-coaching

Costa, C., Aparicio, M., Aparicio, S., & Aparicio, J. (2024). The Democratization of Artificial Intelligence: Theoretical Framework. *Applied Sciences*.

Customer Connect Expo. (2024, June 10). *Call Center Evolution: How Customer Interaction Has Changed Over the Years.* Retrieved from Customer Connect Expo: https://www.customerconnectexpo.com/blogs/call-center-evolution-customer-interaction-changed-over-years

Dale, O. (2024, July 04). *Vodafone Embraces AI with €140 Investment in Chatbot Technology.* Retrieved from Blockonomi: https://blockonomi.com/vodafone-embraces-ai-with-e140-investment-in-chatbot-technology/

Das, A. C., Phalin, G., Patidar, I. L., Gomes, M., Sawhney, R., & Thomas, R. (2023, March 27). *The next frontier of customer engagement: AI-enabled customer service.* Retrieved from McKinsey & Company: https://www.mckinsey.com/capabilities/operations/our-insights/the-next-frontier-of-customer-engagement-ai-enabled-customer-service

Dialpad. (n.d.). *Contact center Ai.* Retrieved 2025, from Dialpad: https://www.dialpad.com/features/contact-center-ai/

Dobkin, R. (2025, May 22). *Klarna’s AI replaced 700 workers. It now wants some of them back to improve customer service.* Retrieved from Independent: https://www.independent.co.uk/news/business/klarna-ceo-sebastian-siemiatkowski-ai-job-cuts-hiring-b2755580.html

Drenik, G. (2025, February 13). *AI And Innovation Are Redefining Retail Returns And Logistics In 2025.* Retrieved from Forbes: https://www.forbes.com/sites/garydrenik/2025/02/13/ai-and-innovation-are-redefining-retail-returns-and-logistics-in-2025/

DRUID. (n.d.). *Banca Transilvania uses AI technology to boost operational processes and employee productivity.* Retrieved from DRUID: https://info.druidai.com/hubfs/cs/druid-cs-banking-operations-support.pdf

DRUID. (n.d.). *Major healthcare clinic boosts patient engagement with DRUID AI Agents.* Retrieved 2025, from DRUID: https://www.druidai.com/case-studies/conversational-ai-healthcare-regina-maria

Emami, A. (2021, November 021). *Introducing Azure Cognitive Service for Language.* Retrieved from Microsoft: https://techcommunity.microsoft.com/blog/azure-ai-services-blog/introducing-azure-cognitive-service-for-language/2910735

Facebook Research. (n.d.). *faiss*. Retrieved 2025, from GitHub: https://github.com/facebookresearch/faiss

FFmpeg. (n.d.). *About FFmpeg.* Retrieved 2025, from FFmpeg: https://ffmpeg.org/about.html

Filipsson, F. (n.d.). *How H&M Uses AI-Powered Chatbots to Improve Customer Service.* Retrieved 2025, from Redress Compliance: https://redresscompliance.com/how-hm-uses-ai-powered-chatbots-to-improve-customer-service/

Fluido. (2023). *The evolution of customer service.* Fluido. Retrieved from https://fluidogroup.com/wp-content/uploads/2023/01/FINAL-VERSION-CUSTOMER-SERVICE-EBOOK.pdf

Gartner. (2025). *Strategy and Leadership Predictions for Service Leaders in 2025.* Retrieved from Gartner: https://emt.gartnerweb.com/ngw/globalassets/en/customer-service-support/documents/trends/customer-service-and-support-2025-leadership-strategy-predicts.pdf

Git. (n.d.). *About.* Retrieved 2025, from Git: https://git-scm.com/about/

Glaser, J. D., & Yao, K. M. (2025, March 10). *The State of the Funding Market for AI Companies: A 2024 - 2025 Outlook.* Retrieved from Mintz: https://www.mintz.com/insights-center/viewpoints/2166/2025-03-10-state-funding-market-ai-companies-2024-2025-outlook

Glia. (n.d.). *Predictive AI Routing - Route to the Right Agent, Right Away.* Retrieved 2025, from Glia: https://www.glia.com/ai-routing

Gold, E. (2023, April 10). *The History of Artificial Intelligence from the 1950s to Today.* Retrieved from freeCodeCamp: https://www.freecodecamp.org/news/the-history-of-ai/

Goyal, A. (2023, August 19). *H&M integrates Virtual Assistant, Chatbot and Live Chat with Google Services: Know Why.* Retrieved from Safalta: https://www.safalta.com/online-digital-marketing/projects-case-studies/h-m-integrates-virtual-assistant-chatbot-and-live-chat-with-google-services

Haan, K. (2024, October 16). *22 Top AI Statistics And Trends.* Retrieved from Forbes: https://www.forbes.com/advisor/business/ai-statistics/

Hadero, H. (2024, November 04). *Many retailers offer ‘returnless refunds.’ Just don’t expect them to say for which products.* Retrieved from Associated Press: https://apnews.com/article/customer-returns-refunds-keep-items-amazon-54d94df2f40866020300685db6dd32f9

Henderson, L. (2024, September 09). *How to build a seamless chatbot to human handoff experience in customer support.* Retrieved from EBI: https://ebi.ai/blog/chatbot-to-human-handoff/

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *Management Information Systems Quarterly*.

Hook, D. (2019, August 16). *TOBI the chatbot: What makes him tick?* Retrieved from Vodacom: https://now.vodacom.co.za/article/tobi-the-chatbot-what-makes-him-tick

Hoory (Soft Construct LLC). (2025, March 25). *History of Customer Service: How Did It All Begin?* Retrieved from Hoory: https://www.hoory.com/blog/i/history-of-customer-service-how-did-it-all-begin

IBM. (n.d.). *Driving a reimagined customer experience with an AI-powered virtual assistant.* Retrieved 2025, from IBM: https://www.ibm.com/case-studies/camping-world

IBM. (n.d.). *Enhancing digital assistant TOBi with generative AI.* Retrieved 2025, from IBM: https://www.ibm.com/case-studies/vodafone-tobi

IBM. (n.d.). *Watson, ‘Jeopardy!’ champion.* Retrieved 2025, from IBM: https://www.ibm.com/history/watson-jeopardy

Intercom. (2023, November). *Believe.* Retrieved from Intercom: https://www.intercom.com/believe

Intercom. (n.d.). *How you can use AI for customer service to help teams and customers.* Retrieved 2025, from Intercom: https://www.intercom.com/learning-center/ai-customer-service

Jackson, M. (2024, July 05). *Vodafone Invests £120m to Deploy New SuperTOBi AI Chatbot.* Retrieved from ISPreview: https://www.ispreview.co.uk/index.php/2024/07/vodafone-invests-120m-to-deploy-new-supertobi-ai-chatbot.html

Jadoul, Y., Thompson, B., & de Boer, B. (2018). Introducing Parselmouth: A Python interface to Praat. *Journal of Phonetics*.

Kane, A., Kirkpatrick, C., Lott, C., Dhamodaran, K., Rezabek, F., Engledow, S., & Strahan, B. (2021, December 17). *Post call analytics for your contact center with Amazon language AI services.* Retrieved from Amazon AWS: https://aws.amazon.com/blogs/machine-learning/post-call-analytics-for-your-contact-center-with-amazon-language-ai-services/

Khachina, E. O. (2025, January 30). *8 Strategies to Provide Efficient Customer Service.* Retrieved from BoldDesk: https://www.bolddesk.com/blogs/strategies-for-efficient-customer-service

King, L. W. (2008). *The Code of Hammurabi.* Retrieved from Yale Law School - The Avalon Project: https://avalon.law.yale.edu/ancient/hamcode.asp

Klarna. (2024, February 27). *Klarna AI assistant handles two-thirds of customer service chats in its first month.* Retrieved from Klarna: https://www.klarna.com/international/press/klarna-ai-assistant-handles-two-thirds-of-customer-service-chats-in-its-first-month/

Klarna. (n.d.). *Hej, we’re Klarna.* Retrieved 2025, from n.d.: https://www.klarna.com/us/about-us/

Kramer, J. (2020, February 25). *Amazon.com tests customer service chatbots.* Retrieved from Amazon Science: https://www.amazon.science/blog/amazon-com-tests-customer-service-chatbots

Kuan, S., & Khan, W. (2022, Ocrober 31). *Improving Watson NLP performance in IBM products through Intel optimizations.* Retrieved from IBM: https://research.ibm.com/blog/watson-nlp-intel-improvements

Lafferty, J. (n.d.). *What Is a Virtual Agent?* Retrieved 2025, from Salesforce: https://www.salesforce.com/uk/agentforce/what-are-virtual-agents/

LangChain. (n.d.). *Introduction*. Retrieved 2025, from LangChain: https://python.langchain.com/docs/introduction/

LangGraph. (n.d.). *Home*. Retrieved 2025, from LangGraph: https://langchain-ai.github.io/langgraph/

Lemon, K. N., & Verhoef, P. C. (2016). Understanding Customer Experience Throughout the Customer Journey. *Journal of Marketing*.

Marti, S., & Sinha, D. (2024, October 10). *New Accenture Research Finds that Companies with AI-Led Processes Outperform Peers.* Retrieved from Accenture: https://newsroom.accenture.com/news/2024/new-accenture-research-finds-that-companies-with-ai-led-processes-outperform-peers

Mathews, A. (2024, July 30). *How Bank Of America’s Erica Boosted Earnings by 19% and What’s Coming Next.* Retrieved from AIM Research: https://aimresearch.co/market-industry/how-bank-of-americas-erica-boosted-earnings-by-19-and-whats-coming-next

Matplotlib. (n.d.). *Matplotlib: Visualization with Python.* Retrieved 2025, from Matplotlib: https://matplotlib.org/

McLean, G., & Wilson, A. (2016). Evolving the online customer experience … is there a role for online customer support? *Computers in Human Behavior*, 602-610.

Meyer, C., & Schwager, A. (2007). Understanding Customer Experience. *Harvard Business Review*.

Microsoft. (2024, October 10). *Transition conversations from bot to human.* Retrieved from Microsoft: https://learn.microsoft.com/en-us/azure/bot-service/bot-service-design-pattern-handoff-human?view=azure-bot-service-4.0

Microsoft. (2025, May 22). *Call center overview.* Retrieved from Microsoft: https://learn.microsoft.com/en-us/azure/ai-services/speech-service/call-center-overview

Microsoft. (2025, February 02). *What is sentiment analysis and opinion mining?* Retrieved from Microsoft: https://learn.microsoft.com/en-us/azure/ai-services/language-service/sentiment-opinion-mining/overview

Mitzner, D. (2025, February 11). *How AI Is Transforming E-Commerce Returns, Cutting Costs And Improving Efficiency.* Retrieved from Forbes: https://www.forbes.com/sites/dennismitzner/2025/02/11/ai-is-transforming-e-commerce-returns-cutting-costs-and-improving-efficiency/

Nelson, R. (2017, November 07). *10 Case Studies on Chatbots.* Retrieved from Overthink Group: https://overthinkgroup.com/chatbot-case-studies/

NICE. (n.d.). *Predictive Analytics for Customer Support.* Retrieved 2025, from NICE: https://www.nice.com/info/top-benefits-of-predictive-analytics-for-customer-support

NICE. (n.d.). *What is Predictive Analytics for Customer Support?* Retrieved 2025, from NICE: https://www.nice.com/glossary/predictive-analytics-for-customer-support

Noone, G. (2024, January 16). *Vodafone inks new digital alliance with Microsoft.* Retrieved from Tech Monitor: https://www.techmonitor.ai/digital-economy/vodafone-microsoft-partnership

One New Zealand. (2020, September 01). *Vodafone NZ signs strategic partnership to bring AI into contact centre solutions.* Retrieved from One New Zealand: https://media.one.nz/business/vodafone-nz-signs-strategic-partnership-to-bring-ai-into-contact-centre-solutions

OpenAI. (2022, January 27). *Aligning language models to follow instructions.* Retrieved from OpenAI: https://openai.com/index/instruction-following/

OpenAI. (2024, May 13). *Live demo of GPT-4o vision capabilities.* Retrieved from OpenAI: https://www.youtube.com/watch?v=RI-BxtCx32s

OpenAI. (2025, March 27). *Zendesk uses OpenAI to build adaptive service agents focused on resolutions.* Retrieved from OpenAI: https://openai.com/index/zendesk/

OpenAI. (n.d.). *OpenAI developer platform*. Retrieved 2025, from OpenAI: https://platform.openai.com/docs/overview

Ozcan, Z., & Guler, U. D. (2025, May 14). *Evolution of Email and What the Future Holds.* Retrieved from Tabular: https://tabular.email/blog/past-and-future-of-email

Panait, T. A. (2025, June 15). *AI Models for Improving and Optimizing Customer Support Efficiency.* Retrieved from GitHub: https://github.com/TudorPanait/dissertation-paper-ai-models-for-improving-and-optimizing-customer-support-efficiency

pandas. (n.d.). *About pandas.* Retrieved 2025, from pandas: https://pandas.pydata.org/about/

Plotly. (n.d.). *Plotly Open Source Graphing Library for Python.* Retrieved 2025, from Plotly: https://plotly.com/python/

Precedence Research. (2025, June 02). *Artificial Intelligence (AI) Market Size, Share, and Trends 2025 to 2034.* Retrieved from Precedence Research: https://www.precedenceresearch.com/artificial-intelligence-market

ProductScope. (2024, December 04). *Temu Return Policy: A Design-Driven Disruption.* Retrieved from Design Disruptors: https://designdisruptors.com/blog/temu-return-policy/

PYMNTS. (2024, March 28). *AI Proves Effective in Stopping eCommerce Returns Fraud.* Retrieved from PYMNTS: https://www.pymnts.com/news/ecommerce/2024/ai-proves-effective-in-stopping-e-commerce-returns-fraud

Python. (n.d.). *About Python*. Retrieved 2025, from Python: https://www.python.org/about/

Python Package Index. (n.d.). *Installing Packages.* Retrieved 2025, from Python Packaging User Guide: https://packaging.python.org/en/latest/tutorials/installing-packages/

QuantumBlack. (2024, April 03). *What is AI (artificial intelligence)?* Retrieved from McKinsey & Company: https://www.mckinsey.com/featured-insights/mckinsey-explainers/what-is-ai

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022). Robust Speech Recognition via Large-Scale Weak Supervision. *40th International Conference on Machine Learning.* Retrieved from https://cdn.openai.com/papers/whisper.pdf

Rahaman, S. (2024, October 21). *A History Of Customer Service.* Retrieved from Fullview: https://www.fullview.io/blog/history-of-customer-service

Rapp, A., Curti, L., & Boldi, A. (2021). The human side of human-chatbot interaction: A systematic literature review of ten years of research on text-based chatbots. *International Journal of Human-Computer Studies*.

Regina Maria. (2022, July 28). *Regina Maria lanseaza, in premiera in Romania, primul Asistent Virtual care evalueaza online simptomele pacientilor.* Retrieved from Regina Maria: https://www.reginamaria.ro/stiri/regina-maria-lanseaza-premiera-romania-primul-asistent-virtual-care-evalueaza-online

Relevance AI. (n.d.). *Returns and Refunds Process Optimization AI Agents.* Retrieved 2025, from Relevance AI: https://relevanceai.com/agent-templates-tasks/returns-and-refunds-process-optimization-ai-agents

Reputation. (2025). *Unlocking the Power of Customer Sentiment Analysis: A Beginner’s Guide.* Retrieved from Reputation: https://reputation.com/resources/articles/customer-sentiment-analysis-101/

Rose, W. (2025, April 16). *A brief history of AI in customer support.* Retrieved from Assembled: https://www.assembled.com/blog/a-brief-history-of-ai-in-customer-support

Rumley, K., Nguyen, J., & Neskovic, G. (2023, May 02). *How Speech Recognition Improves Customer Service in Telecommunications.* Retrieved from Nvidia: https://developer.nvidia.com/blog/how-speech-recognition-improves-customer-service-in-telecommunications/

Santos, G., & Koromyslova, E. (2020). A Review on Returnless Refunds in E-Commerce Retailing: Sales through Leniency. *iBusiness*, 69-80.

Saurabh, K. (n.d.). *python-dotenv 1.1.0.* Retrieved 2025, from Python Package Index: https://pypi.org/project/python-dotenv/

scikit learn. (n.d.). *Getting Started*. Retrieved 2025, from scikit learn: https://scikit-learn.org/stable/getting\_started.html

Sheikh, H., Prins, C., & Schrijvers, E. (2023). *Mission AI.* Springer, Cham.

Shep, H. (2020, July 12). *Ninety-Six Percent Of Customers Will Leave You For Bad Customer Service.* Retrieved from Forbes: https://www.forbes.com/sites/shephyken/2020/07/12/ninety-six-percent-of-customers-will-leave-you-for-bad-customer-service/

Shetty, S. (2024, November 09). *The Digital CX Revolution: Leveraging AI and Emerging Technologies to Enhance Customer Engagement.* Retrieved from CXPA: https://www.cxpa.org/blogs/sahil-shetty/2024/11/09/the-digital-cx-revolution-leveraging-ai

Shweta. (2022, August 21). *What Is A Chatbot? Everything You Need To Know.* Retrieved from Forbes: https://www.forbes.com/advisor/business/software/what-is-a-chatbot/

Simon, B., Nicholas, C., Sukand, R., & Veronika, S. (2023, July 06). *How Generative AI Is Already Transforming Customer Service.* Retrieved from https://www.bcg.com/publications/2023/how-generative-ai-transforms-customer-service: https://www.bcg.com/publications/2023/how-generative-ai-transforms-customer-service

Spitch. (n.d.). *Powerful Speech Analytics. Exceptional Contact Center Performance.* Retrieved 2025, from Spitch: https://spitch.ai/speech-analytics/

Steinle, S. (2025, May 8). *Six AI-Accelerated, Proactive, and Preventative Methods Transforming SAP’s Customer Support.* Retrieved from SAP: https://news.sap.com/2025/05/six-ai-proactive-preventative-methods-transforming-customer-support/

Streamlit. (n.d.). *Streamlit documentation.* Retrieved 2025, from Streamlit: https://docs.streamlit.io/

Subhashis, J., & George, R. V. (2024, January 31). *What is Call Center Sentiment Analysis?* Retrieved from Sprinklr: https://www.sprinklr.com/cxm/call-center-sentiment-analysis/

Takyar, A. (2025). *Customer churn prediction using machine learning: A comprehensive overview.* Retrieved from LeewayHertz: https://www.leewayhertz.com/ai-and-ml-in-customer-churn-prediction/

Takyar, A. (n.d.). *AI in complaints and returns management: Scope, integration, use cases, challenges and future outlook.* Retrieved 2025, from ZBrain: https://zbrain.ai/ai-in-complaints-and-returns-management/

Talkative. (2023, May 02). *The Limitations of Chatbots (And How to Overcome Them).* Retrieved from Talkative: https://gettalkative.com/info/limitations-of-chatbot

Templier, M., & Paré, G. (2015). A Framework for Guiding and Evaluating Literature. *Communications of the Association for Information Systems*.

Temu. (2025, January 16). *Temu | Return and Refund Policy.* Retrieved from Temu: https://www.temu.com/return-and-refund-policy.html

Temu. (n.d.). *About Temu.* Retrieved 2025, from Temu: https://www.temu.com/about-temu.html

Teneo. (n.d.). *Artificial Intelligence in Call Centers: The Complete Guide.* Retrieved Teneo, from Teneo: https://www.teneo.ai/blog/artificial-intelligence-in-call-centers

TextBlob. (n.d.). *TextBlob: Simplified Text Processing*. Retrieved 2025, from TextBlob: https://textblob.readthedocs.io/en/dev/

The Economic Times. (2025, June 09). *Company that sacked 700 workers with AI now regrets it — scrambles to rehire as automation goes horribly wrong.* Retrieved from The Economic Times: https://economictimes.indiatimes.com/news/international/us/company-that-sacked-700-workers-with-ai-now-regrets-it-scrambles-to-rehire-as-automation-goes-horribly-wrong/articleshow/121732999.cms

The Editors of Encyclopædia Britannica. (2025, April 25). *Code of Hammurabi.* Retrieved from Britannica: https://www.britannica.com/topic/Code-of-Hammurabi

The Watkins Co. (n.d.). *A Natural History.* Retrieved 2025, from Watkins: https://www.watkins1868.com/pages/our-history

Tongco, H. K. (2025, April 02). *The Human Touch Isn’t Optional: AI’s Limits in Handling Complex Customer Needs.* Retrieved from Superstaff: https://www.superstaff.com/blog/ai-limitations-in-customer-service/

Verma, K. (2024, September 26). *The Evolution of Chatbots: From Rule-Based to AI-Powered.* Retrieved from Toronto Digital: https://torontodigital.ca/blog/the-evolution-of-chatbots-from-rule-based-to-ai-powered/

Vishwas B, S. (n.d.). *rake-nltk 1.0.6*. Retrieved 2025, from Python Package Index: https://pypi.org/project/rake-nltk/

Vodafone. (2024, May). *FY24 Results.* Retrieved from Vodafone: https://reports.investors.vodafone.com/view/131231154/22/

Vodafone. (2024, July 04). *Meet SuperTOBI - Vodafone’s new Generative AI virtual assistant now serving customers in multiple countries.* Retrieved from Vodafone: https://www.vodafone.com/news/technology/meet-super-tobi-vodafone-s-new-generative-ai-virtual-assistant-now-serving-customers-in-multiple-countries

Vohra, D. K. (2024, October 01). *How AI and RAG Chatbots Cut Customer Service Costs by Millions.* Retrieved from Nexgen Cloud: https://www.nexgencloud.com/blog/case-studies/how-ai-and-rag-chatbots-cut-customer-service-costs-by-millions

Wallace, R. (2009). The Anatomy of A.L.I.C.E. In R. R. Epstein, *Parsing the Turing Test.*

Wang, Y. (2024, February 14). *Case Study: How Alibaba Uses AI Chatbots to Serve a Billion Customers.* Retrieved from AI Business: https://aibusiness.com/ml/the-alibaba-challenge-how-to-effectively-engage-with-a-billion-customers-

Wheeler, K. (2025, February 26). *How Alibaba’s Investment Surges AI and Cloud Expansion.* Retrieved from Technology Magazine: https://technologymagazine.com/articles/how-alibabas-investment-surges-ai-and-cloud-expansion

Wood, C. X. (2024, July 24). *5 AI Case Studies in Customer Service and Support.* Retrieved from VKTR: https://www.vktr.com/ai-disruption/5-ai-case-studies-in-customer-service-and-support/

Wood, N. (2024, July 04). *Blow to IBM as OpenAI gives Vodafone's chatbot the superhero treatment.* Retrieved from Telecoms: https://www.telecoms.com/ai/blow-to-ibm-as-openai-gives-vodafone-s-chatbot-the-superhero-treatment

WorkAI. (2023, April 28). *Why do you need an AI-powered knowledge management system?* Retrieved from WorkAI: https://workai.com/insights/ai-powered-knowledge-management-system/

World Bank Group. (2024, March 05). *Accelerated by COVID and AI, Global Digital Landscape Remains Uneven.* Retrieved from World Bank Group: https://www.worldbank.org/en/news/press-release/2024/03/05/accelerated-by-covid-and-ai-global-digital-landscape-remains-uneven

Young, K. (2024, March 25). *A Comprehensive Guide to Temu’s Refund Without Return Policy.* Retrieved from THunt: https://thunt.ai/blog/a-comprehensive-guide-to-temus-refund-without-return-policy

Young, S. (2022, December 22). *How ‘Albert’ transformed our knowledge-based contact centre solution.* Retrieved from British Telecom : https://business.bt.com/insights/how-ai-transformed-our-contact-centre-solution/

Zarecki, I. (2025, April 21). *RAG Chatbot: What’s it All a Bot?* Retrieved from K2view: https://www.k2view.com/blog/rag-chatbot/

Zendesk. (n.d.). *Motel sees self-service climb by 3X and faster ticket handling with Zendesk AI.* Retrieved 2025, from Zendesk: https://www.zendesk.com/customer/motel-rocks/

# Appendixes

## Appendix 1 – Sentiment Analysis Application Source Code

import os

from dotenv import load\_dotenv

import streamlit as st

from textblob import TextBlob

from langchain\_openai import ChatOpenAI

def header():

st.set\_page\_config(page\_title="Sentiment Analysis", page\_icon=":neutral\_face:")

st.title("Sentiment Analysis")

st.write("This page is dedicated to sentiment analysis tasks.")

load\_dotenv()

def main():

st.subheader("Analyze the sentiment of your text")

user\_input = st.text\_area("Enter text for sentiment analysis:", value="I don't know what to write here, but I want to analyze the sentiment of this text.")

user\_choice = st.selectbox("Choose analysis method:", ["TextBlob", "OpenAI"])

if st.button("Analyze"):

if user\_input:

if user\_choice == "TextBlob":

textblob\_sentiment\_analysis(user\_input)

else:

st.write\_stream(openai\_sentiment\_analysis(user\_input))

else:

st.warning("Please enter some text to analyze.")

def textblob\_sentiment\_analysis(user\_input):

analysis = TextBlob(user\_input)

sentiment = analysis.sentiment

st.write(f"Polarity: {sentiment.polarity}\n\n Subjectivity: {sentiment.subjectivity}")

def openai\_sentiment\_analysis(user\_input):

llm = ChatOpenAI(model="gpt-4.1-nano")

for chunk in llm.stream(

[

(

"system",

"You are a sentiment analysis assistant. Your purpose is to analyse the appended text for sentiment, polarity, subjectivity and to explain the reasoning behind your output. The output should follow the parttern - Sentiment (Negative/Neutral/Positive)\nPolarity(-1.0,1.0)\nSubjectivity(Low/Medium/High)\nExplanation"),

(

"human",

user\_input

)

]

):

yield chunk.content

header()

main()

## Appendix 2 – Intent Analysis Application Source Code

import os

from dotenv import load\_dotenv

import streamlit as st

from langchain\_openai import ChatOpenAI

def header():

st.set\_page\_config(page\_title="Intent Analysis", page\_icon=":left\_right\_arrow:")

st.title("Intent Analysis")

st.write("This page is dedicated to intent analysis tasks.")

load\_dotenv()

def main():

st.subheader("Analyze the intent of your text")

user\_input = st.text\_area("Enter text for intent analysis:", value="I don't know what to write here, but I want to analyze the intent of this text.")

if st.button("Analyze"):

if user\_input:

st.write\_stream(openai\_intent\_analysis(user\_input))

else:

st.warning("Please enter some text to analyze.")

def openai\_intent\_analysis(user\_input):

llm = ChatOpenAI(model="gpt-4.1-nano")

for chunk in llm.stream(

[

(

"system",

"You are an intent analysis assistant. Your purpose is to analyse the appended text for intent, and to explain the reasoning behind your output. The output should follow the pattern - Intent (Action/Information/Question)\nExplanation"

),

(

"human",

user\_input

)

]

):

yield chunk.content

header()

main()

## Appendix 3 – Rule-Based Chatbot Application Source Code

import os

from dotenv import load\_dotenv

import streamlit as st

def header():

st.set\_page\_config(page\_title="Rule Based Chatbot", page\_icon=":memo:")

st.title("Rule Based Chatbot")

st.write("This page is dedicated to the rule based chatbot.")

load\_dotenv()

def main():

st.subheader("Interact with the rule based chatbot")

init()

# Level 1: Initial message

if st.session\_state.last\_button == "Account Issues":

write\_message("user", "text", "I have an account issue, please help.")

write\_message("assistant", "text" , "Please select one of the following options.")

write\_message("assistant", "button", "Reset Password")

write\_message("assistant", "button", "Update Info")

elif st.session\_state.last\_button == "Order Status":

write\_message("user", "text", "I want to check my order status, please.")

write\_message("assistant", "text", "Please select one of the following options.")

write\_message("assistant", "button", "Track Order")

write\_message("assistant", "button", "Cancel Order")

elif st.session\_state.last\_button == "Talk to a Human Agent":

write\_message("user", "text", "I want to talk to a Human Agent, please.")

write\_message("assistant", "text", "Please wait while we connect you to a human agent.")

# Level 2: Account Issues

if st.session\_state.last\_button == "Reset Password":

write\_message("user", "text", "I want to reset my password.")

write\_message("assistant", "text", "Please follow this link to reset your password: [Reset Password Link]")

elif st.session\_state.last\_button == "Update Info":

write\_message("user", "text", "I want to update my account information.")

write\_message("assistant", "text", "Please follow this link to update your information: [Update Info Link]")

# Level 2: Order Status

if st.session\_state.last\_button == "Track Order":

if st.session\_state.user\_input == "":

write\_message("user", "text", "I want to track my order.")

write\_message("assistant", "input", "Please enter your order ID to track your order.")

elif st.session\_state.user\_input:

write\_message("user", "text", f"My order ID: {st.session\_state.user\_input}")

write\_message("assistant", "input", "Tracking your order...")

write\_message("assistant", "text", f"Your order status for {st.session\_state.user\_input} is:\n\n - In Transit\n - Estimated Delivery: 3 days")

elif st.session\_state.last\_button == "Cancel Order":

if st.session\_state.user\_input == "":

write\_message("user", "text", "I want to cancel my order.")

write\_message("assistant", "input", "Please enter your order ID to cancel your order.")

elif st.session\_state.user\_input:

write\_message("user", "text", f"My order ID: {st.session\_state.user\_input}")

write\_message("assistant", "input", "Processing your cancellation request...")

write\_message("assistant", "text", f"Your order with ID {st.session\_state.user\_input} has been submitted for cancellation. You will receive a confirmation soon.")

def init():

if "rule\_based\_messages" not in st.session\_state:

st.session\_state.rule\_based\_messages = [

{

"role": "assistant",

"type": "text",

"content": "How may I assist you today?"

},

{

"role": "assistant",

"type": "button",

"content": "Account Issues"

},

{

"role": "assistant",

"type": "button",

"content": "Order Status"

},

{

"role": "assistant",

"type": "button",

"content": "Talk to a Human Agent"

}

]

st.session\_state.last\_button = None

st.session\_state.user\_input = ""

messages = st.session\_state.get("rule\_based\_messages", [])

i = 0

while i < len(messages):

current\_role = messages[i]["role"]

group = []

while i < len(messages) and messages[i]["role"] == current\_role:

group.append(messages[i])

i += 1

with st.chat\_message(current\_role):

for msg in group:

if msg["type"] == "button":

if st.button(msg["content"]):

st.session\_state.last\_button = msg["content"]

else:

st.write(msg["content"])

def write\_message(role, type, content):

message = {

"role": role,

"type": type,

"content": content

}

with st.chat\_message(role):

if type == "button":

if st.button(content):

st.session\_state.last\_button = content

elif type == "input":

if st.session\_state.user\_input == "":

st.session\_state.user\_input = st.text\_input(content)

if st.session\_state.user\_input:

st.rerun()

elif content == "Tracking your order...":

st.write(f"Tracking your order with ID {st.session\_state.user\_input}...")

elif content == "Processing your cancellation request...":

st.write(f"Processing cancellation for order ID {st.session\_state.user\_input}...")

else:

st.write(content)

st.session\_state.rule\_based\_messages.append(message)

header()

main()

## Appendix 4 – Keyword-Based Chatbot Application Source Code

import os

from dotenv import load\_dotenv

import streamlit as st

import difflib

from langchain\_openai import ChatOpenAI

KEYWORD\_RESPONSES = {

"hello": "Hello! How can I assist you today?",

"help": "Sure, I'm here to help. Please describe your issue.",

"price": "Our pricing information can be found on our website.",

"support": "You can contact support at support@example.com.",

"bye": "Goodbye! Have a great day!",

"order": "For order-related inquiries, please visit our order help page.",

"account": "For account-related issues, please visit our account help page.",

"payment": "For payment issues, please check our payment help section.",

"shipping": "Shipping information can be found on our shipping page.",

"default": "I'm sorry, I didn't understand that.\n\n I can help you with account, order, shipping or payment issues.\n\nCan you please rephrase?",

}

SIMILAR\_KEYWORDS = {

"hello": ["hello", "helo", "hey", "hi", "hullo"],

"help": ["help", "halp", "assist", "support"],

"price": ["price", "cost", "fee", "prize", "pricing"],

"support": ["support", "assistance", "helpdesk"],

"bye": ["bye", "goodbye", "see ya", "cya"],

"order": ["order", "purchase", "buy", "place order"],

"account": ["account", "profile", "user account", "my account"],

"payment": ["payment", "pay", "billing", "charge"],

"shipping": ["shipping", "delivery", "ship", "postage"],

}

def header():

st.set\_page\_config(page\_title="Keyword Based Chatbot", page\_icon=":abc:")

st.title("Keyword Based Chatbot")

st.write("This page is dedicated to the keyword based chatbot.")

st.session\_state.use\_ai = st.selectbox(

"Enable AI for smarter keyword matching? (e.g., if 'Yes', typing 'Holla' will use AI to find the closest keyword match)",

options=["No", "Yes"],

index=0,

)

load\_dotenv()

def main():

st.subheader("Interact with the keyword based chatbot")

init()

def init():

if "use\_ai" not in st.session\_state:

st.session\_state.use\_ai = "No"

if "keyword\_based\_messages" not in st.session\_state:

st.session\_state.keyword\_based\_messages = [

("assistant", "How may I assist you today?\n\nI can help you with account, order, shipping or payment issues."),

]

user\_input = st.chat\_input("Type your message here...")

if user\_input:

st.session\_state.keyword\_based\_messages.append(("user", user\_input))

response = get\_response(user\_input)

st.session\_state.keyword\_based\_messages.append(("assistant", response))

for sender, message in st.session\_state.keyword\_based\_messages:

with st.chat\_message(sender):

st.markdown(message)

def get\_response(user\_input):

if st.session\_state.use\_ai == "Yes":

user\_input = ai\_keyword\_similarity(user\_input)

else:

user\_input = user\_input.lower()

words = user\_input.split()

best\_match = None

best\_ratio = 0.0

best\_keyword = None

for keyword, variants in SIMILAR\_KEYWORDS.items():

for variant in variants:

for word in words:

ratio = difflib.SequenceMatcher(None, word, variant).ratio()

if ratio > best\_ratio and ratio > 0.9:

best\_ratio = ratio

best\_keyword = keyword

if variant in user\_input:

best\_keyword = keyword

best\_ratio = 1.0

if best\_keyword:

return KEYWORD\_RESPONSES[best\_keyword]

return KEYWORD\_RESPONSES["default"]

def ai\_keyword\_similarity(user\_input):

llm = ChatOpenAI(model="gpt-4.1-nano")

message = [

(

"system",

"You are a keyword-based chatbot assistant. Your task is to classify the user's message into one of the following categories: hello, help, price, support, bye, order, account, payment, shipping, or default. "

"Respond with only the single category word (e.g., 'hello', 'help', etc.) that best matches the user's input. "

"If the input does not fit any category, respond with 'default'. Do not provide any explanation or additional text. Just return the category word.",

),

(

"human",

user\_input

)

]

matched\_keyword = llm.invoke(message).content.strip()

return matched\_keyword if matched\_keyword in KEYWORD\_RESPONSES else "default"

header()

main()

## Appendix 5 – Embeddings Application Source Code

import os

from dotenv import load\_dotenv

import streamlit as st

import pandas as pd

import faiss

import pickle

from sklearn.manifold import TSNE

import plotly.express as px

from langchain\_openai import OpenAIEmbeddings

from langchain\_community.vectorstores import FAISS

from langchain.docstore.document import Document

from langchain.text\_splitter import RecursiveCharacterTextSplitter

def header():

st.set\_page\_config(page\_title="Embedding", page\_icon=":link:")

st.title("Embedding")

st.write("This page is dedicated to the embedding functionality.")

load\_dotenv()

def main():

st.subheader("Create Embeddings from Text Files")

st.write("This functionality allows you to create embeddings from text files stored in the `resources/articles`.")

st.write("\*\*These are the texts that will be used to create embeddings:\*\*")

for file in os.listdir("resources/articles"):

if file.endswith(".txt"):

with st.expander(file):

with open(os.path.join("resources/articles", file), "r", encoding="utf-8") as f:

content = f.read()

st.write(content)

if st.button("Create Embeddings"):

docs = []

script\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

articles\_dir = os.path.join(script\_dir, "..", "..", "resources", "articles")

for file in os.listdir(articles\_dir):

if file.endswith(".txt"):

with open(os.path.join("resources/articles", file), "r", encoding="utf-8") as f:

content = f.read()

docs.append(Document(page\_content=content, metadata={"source": file}))

chunks = chunk\_documents(docs)

create\_embeddings(chunks)

def chunk\_documents(docs, chunk\_size=500, chunk\_overlap=200):

text\_splitter = RecursiveCharacterTextSplitter(

chunk\_size=chunk\_size,

chunk\_overlap=chunk\_overlap,

length\_function=len

)

return text\_splitter.split\_documents(docs)

def create\_embeddings(chunks):

with st.spinner("Creating embeddings..."):

if chunks:

embedding\_model = OpenAIEmbeddings(model="text-embedding-3-small")

db = FAISS.from\_documents(chunks, embedding\_model)

db.save\_local("resources/vectorstore")

st.balloons()

st.success("Embeddings created and saved successfully in the `resources/vectorstore` directory! Use them for search functionality available in the next page.")

chunk\_data = [{"Source": chunk.metadata["source"], "Chunk Size": len(chunk.page\_content)} for chunk in chunks]

df = pd.DataFrame(chunk\_data)

chunk\_counts = df.groupby("Source").size().reset\_index(name="Chunks")

st.write("\*\*Number of Chunks per Document:\*\*")

st.bar\_chart(chunk\_counts.set\_index("Source"))

visualize\_embeddings("resources/vectorstore/index.faiss", "resources/vectorstore/index.pkl")

else:

st.error("No documents found to create embeddings. Please upload a text file or ensure the sample texts are available.")

def visualize\_embeddings(faiss\_path, meta\_path):

if os.path.exists(faiss\_path) and os.path.exists(meta\_path):

index = faiss.read\_index(faiss\_path)

with open(meta\_path, 'rb') as f:

metadata = pickle.load(f)

else:

st.error("❌ FAISS index or metadata not found.")

return

num\_vectors = index.ntotal

vectors = index.reconstruct\_n(0, num\_vectors)

reduced\_vectors = TSNE(n\_components=3, random\_state=42).fit\_transform(vectors)

if isinstance(metadata, list):

labels = [m.get('text', f"Chunk {i}") if isinstance(m, dict) else str(m) for i, m in enumerate(metadata)]

else:

labels = [f"Chunk {i}" for i in range(num\_vectors)]

fig = px.scatter\_3d(

x=reduced\_vectors[:, 0],

y=reduced\_vectors[:, 1],

z=reduced\_vectors[:, 2],

hover\_name=labels,

title="FAISS 3D Embedding Visualization"

)

fig.update\_layout(margin=dict(l=0, r=0, t=50, b=0))

st.plotly\_chart(fig, use\_container\_width=True)

header()

main()

## Appendix 6 – Knowledge Management with Search Application Source Code

import os

from dotenv import load\_dotenv

import streamlit as st

from langchain\_openai import OpenAIEmbeddings

from langchain\_community.vectorstores import FAISS

from langchain.docstore.document import Document

def header():

st.set\_page\_config(page\_title="Knowledge Management with Search", page\_icon=":mag:")

st.title("Knowledge Management with Search")

load\_dotenv()

st.write("This page is dedicated to knowledge management and search functionality.")

st.write("It uses the articles present in the `resources/articles` directory.")

st.write("If searching with embeddings, it will search through the vector store in `resources/vectorstore`.")

def main():

st.subheader("Search for Information")

st.write("This feature allows you to search for information within a knowledge base.")

use\_embeddings = st.selectbox(

"Select the method for search:",

["Use Keywords", "Use Embeddings"]

)

search\_query = st.text\_input("Enter your search query:")

if search\_query:

if use\_embeddings == "Use Keywords":

search\_with\_keywords(search\_query)

else:

search\_with\_embeddings(search\_query)

def highlight\_keywords(content, query):

highlighted\_content = content.replace(query, f" \*\*\*`{query}`\*\*\* " )

return highlighted\_content

def search\_with\_keywords(query):

with st.spinner("Searching the knowledge base..."):

script\_dir = os.path.dirname(os.path.abspath(\_\_file\_\_))

articles\_dir = os.path.join(script\_dir, "..", "..", "resources", "articles")

results = []

for file in os.listdir(articles\_dir):

if file.endswith(".txt"):

with open(os.path.join(articles\_dir, file), "r", encoding="utf-8") as f:

content = f.read()

if query.lower() in content.lower():

highlighted\_content = highlight\_keywords(content, query)

results.append(Document(page\_content=highlighted\_content, metadata={"source": file}))

if results:

st.write("\*\*Search Results:\*\*")

for result in results:

st.write(f"\*\*Source:\*\* {result.metadata['source']}")

st.write(result.page\_content)

st.write("---")

else:

st.warning("No results found for your query.")

def search\_with\_embeddings(query):

with st.spinner("Searching the vector store..."):

embedding\_model = OpenAIEmbeddings(model="text-embedding-3-small")

db = FAISS.load\_local("resources/vectorstore", embedding\_model, allow\_dangerous\_deserialization=True)

results = db.similarity\_search(query, k=3)

if results:

st.write("\*\*Search Results:\*\*")

for result in results:

st.write(f"\*\*Source:\*\* {result.metadata['source']}")

st.write(result.page\_content)

st.write("---")

else:

st.warning("No results found for your query.")

header()

main()

## Appendix 7 – Conversational AI Chatbot Application Source Code

import os

from dotenv import load\_dotenv

import streamlit as st

from langchain\_openai import ChatOpenAI

from langchain\_openai import OpenAIEmbeddings

from langchain\_community.vectorstores import FAISS

AI\_PROMPT = (

"You are Tudor, an intelligent and friendly AI assistant for a fictional online electronics retailer called ABC Store. "

"You're here to help customers with anything from account issues to shipping questions, or just guide them through using the store. "

"Your responses should sound natural, conversational, and human-like — not robotic or scripted. "

"You can be creative in how you respond, as long as your answers are relevant, helpful, and professional.\n\n"

"Use the following keywords as inspiration for what the user might be asking about — you don’t need to repeat the exact phrasing, "

"just understand the theme and respond naturally:\n\n"

"- 'hello': Start a warm, friendly conversation. Introduce yourself as an AI assistant and invite them to ask anything.\n"

"- 'help': Offer support and encourage them to describe what they need. Let them know you're here to assist.\n"

"- 'price': Guide them to product pricing or deals, maybe even suggest popular items if relevant.\n"

"- 'support': Let them know how to reach human support, but try to help first if you can.\n"

"- 'bye': End the conversation kindly and leave the door open for them to return anytime.\n"

"- 'order': Help with tracking orders, modifying them, or understanding the process.\n"

"- 'account': Answer questions about login, profiles, or settings.\n"

"- 'payment': Address billing issues, accepted methods, and common problems.\n"

"- 'shipping': Explain shipping options, delivery times, and how to track packages.\n\n"

"If you’re unsure what the user means, don’t be afraid to ask for clarification. If something is outside your scope, say so politely, and offer to connect them to a human support agent.\n\n"

"Your personality should be helpful, patient, and a little witty when appropriate. You’re not just answering questions — you’re creating a great customer experience. "

"Think like a real AI assistant designed to impress."

"You can not take any actions on the user's behalf, but you can guide them through the process."

"If you need to take any actions, you can guide the user to do it themselves by providing step-by-step instructions or by asking if they would like to be connected to a human support agent."

)

def header():

st.set\_page\_config(page\_title="Conversational AI Chatbot", page\_icon=":speech\_balloon:")

st.title("Conversational AI Chatbot")

st.write("This page is dedicated to the conversational AI chatbot with memory and tool access.")

load\_dotenv()

def main():

st.subheader("Interact with the conversational AI chatbot")

init()

def init():

if "conversational\_ai\_messages" not in st.session\_state:

st.session\_state.conversational\_ai\_messages = [

("intstructions", AI\_PROMPT),

("assistant", "Hello! I am Tudor, your virtual AI assistant for ABC Store. How may I assist you today?"),

]

user\_input = st.chat\_input("Type your message here...")

if user\_input:

st.session\_state.conversational\_ai\_messages.append(("user", user\_input))

knowledge\_base\_info = search\_with\_embeddings(user\_input)

llm = ChatOpenAI(model="gpt-4.1-nano")

query = "History of the chat: " + str(st.session\_state.conversational\_ai\_messages) + "\nUser's latest request: " + user\_input + "Information from our knowledge base: " + str(knowledge\_base\_info)

response = llm.invoke(query)

st.session\_state.conversational\_ai\_messages.append(("assistant", response.content))

for role, message in st.session\_state.conversational\_ai\_messages:

if role == "user":

st.chat\_message("user").write(message)

elif role == "assistant":

st.chat\_message("assistant").write(message)

def search\_with\_embeddings(query: str) -> list:

"""

Search the internal knowledge base when in need of more information.

Args:

query (str): The user's query to search for relevant information.

Returns:

list: A list of relevant documents from the knowledge base.

"""

embedding\_model = OpenAIEmbeddings(model="text-embedding-3-small")

db = FAISS.load\_local("resources/vectorstore", embedding\_model, allow\_dangerous\_deserialization=True)

results = db.similarity\_search(query, k=2)

return results

header()

main()

## Appendix 8 – Voice AI Chatbot Application Source Code

import os

from dotenv import load\_dotenv

import streamlit as st

import tempfile

from openai import OpenAI

def header():

st.set\_page\_config(page\_title="Voice AI Chatbot", page\_icon=":microphone:")

st.title("Voice AI Chatbot")

st.write("This page is dedicated to the voice AI agent that has no memory or tool access.")

load\_dotenv()

def main():

st.subheader("Interact with the Voice AI Chatbot")

client = OpenAI()

audio = st.audio\_input("Record a voice message")

response = None

text\_response = None

if audio:

with tempfile.NamedTemporaryFile(delete=False, suffix=".wav") as tmp\_user\_audio:

tmp\_user\_audio.write(audio.getvalue())

user\_audio\_path = tmp\_user\_audio.name

with st.spinner("Processing your voice input..."):

response = client.audio.transcriptions.create(

file=open(user\_audio\_path, "rb"),

model="gpt-4o-mini-transcribe",

response\_format="text"

)

st.info(response)

st.write("---")

with st.spinner("Generating response..."):

text\_response = client.responses.create(

model="gpt-4.1-nano",

input="Answer the following question from the user: " + str(response)

).output\_text

with tempfile.NamedTemporaryFile(delete=False, suffix=".mp3") as tmp\_assistant\_audio:

assistant\_audio\_path = tmp\_assistant\_audio.name

with client.audio.speech.with\_streaming\_response.create(

model="gpt-4o-mini-tts",

voice="coral",

input=str(text\_response),

instructions="Use a tone that resembles the sentiment of the text.",

) as audio\_response:

audio\_response.stream\_to\_file(assistant\_audio\_path)

st.audio(assistant\_audio\_path, autoplay=True)

st.success(text\_response)

def init():

if "voice\_ai\_messages" not in st.session\_state:

st.session\_state.voice\_ai\_messages = [

("intstructions", AI\_PROMPT),

("assistant", "Hello! I am Tudor, your virtual AI assistant for ABC Store. How may I assist you today?"),

]

user\_input = st.chat\_input("Type your message here...")

if user\_input:

st.session\_state.voice\_ai\_messages.append(("user", user\_input))

knowledge\_base\_info = search\_with\_embeddings(user\_input)

llm = ChatOpenAI(model="gpt-4.1-nano")

query = "History of the chat: " + str(st.session\_state.voice\_ai\_messages) + "\nUser's latest request: " + user\_input + "Information from our knowledge base: " + str(knowledge\_base\_info)

response = llm.invoke(query)

st.session\_state.voice\_ai\_messages.append(("assistant", response.content))

for role, message in st.session\_state.voice\_ai\_messages:

if role == "user":

st.chat\_message("user").write(message)

elif role == "assistant":

st.chat\_message("assistant").write(message)

header()

main()

## Appendix 9 – Speech AI Assist Application Source Code

import os

from dotenv import load\_dotenv

import streamlit as st

import parselmouth

import tempfile

import numpy as np

from openai import OpenAI

import matplotlib.pyplot as plt

from scipy.signal import find\_peaks

from datetime import timedelta

client = OpenAI(api\_key=os.getenv("OPENAI\_API\_KEY"))

def header():

st.set\_page\_config(page\_title="Speech Assist", page\_icon=":microphone:")

st.title("Speech Assist")

st.write("This page is dedicated to analyzing vocal profiles and providing feedback for call-center agents.")

load\_dotenv()

def analyze\_audio(wav\_path):

sound = parselmouth.Sound(wav\_path)

duration = sound.duration

pitch = sound.to\_pitch()

intensity = sound.to\_intensity()

harmonicity = sound.to\_harmonicity\_cc()

formant = sound.to\_formant\_burg()

pitch\_values = pitch.selected\_array['frequency']

voiced\_pitch = pitch\_values[pitch\_values > 0]

mean\_pitch = voiced\_pitch.mean() if len(voiced\_pitch) > 0 else 0

min\_pitch = voiced\_pitch.min() if len(voiced\_pitch) > 0 else 0

max\_pitch = voiced\_pitch.max() if len(voiced\_pitch) > 0 else 0

pitch\_range = max\_pitch - min\_pitch

mean\_intensity = intensity.values.mean()

hnr = harmonicity.values[harmonicity.values > 0].mean() if np.any(harmonicity.values > 0) else 0

voiced\_ratio = len(voiced\_pitch) / len(pitch\_values) if len(pitch\_values) > 0 else 0

pauses = np.where(intensity.values[0] < 25)[0]

pause\_ratio = len(pauses) / len(intensity.values[0])

return {

"Duration (s)": duration,

"Mean Pitch (Hz)": mean\_pitch,

"Pitch Range (Hz)": pitch\_range,

"Mean Intensity (dB)": mean\_intensity,

"HNR (dB)": hnr,

"Voiced Time (%)": voiced\_ratio \* 100,

"Pause Ratio (%)": pause\_ratio \* 100

}

def transcribe\_audio(file\_path):

with open(file\_path, "rb") as audio\_file:

response = client.audio.transcriptions.create(

file=audio\_file,

model="gpt-4o-mini-transcribe",

response\_format="text"

)

return response

def give\_feedback(results, wpm):

feedback = []

if results["Mean Pitch (Hz)"] < 120:

feedback.append("🟡 Try raising your pitch — you may sound flat or monotone.")

if results["Pitch Range (Hz)"] < 30:

feedback.append("🟡 Vary your tone more to sound engaging.")

if wpm > 170:

feedback.append("🔴 You're speaking too fast — aim for 130–160 wpm.")

elif wpm < 110:

feedback.append("🟡 You're speaking slowly — try to pick up the pace.")

if results["HNR (dB)"] < 10:

feedback.append("🔴 Low HNR — your voice may sound breathy or unclear.")

if results["Pause Ratio (%)"] > 20:

feedback.append("🟡 There are a lot of pauses — try to smooth your delivery.")

return feedback

def main():

st.subheader("Analyze Your Speech")

audio\_data = st.audio\_input("Record your voice")

if audio\_data:

with tempfile.NamedTemporaryFile(delete=False, suffix=".wav") as tmp\_wav:

tmp\_wav.write(audio\_data.getvalue())

wav\_path = tmp\_wav.name

with st.spinner("Analyzing your speech..."):

results = analyze\_audio(wav\_path)

transcript = transcribe\_audio(wav\_path)

word\_count = len(transcript.split())

duration\_min = results["Duration (s)"] / 60

wpm = word\_count / duration\_min if duration\_min > 0 else 0

results["Estimated Speaking Rate (wpm)"] = wpm

st.subheader("💬 Transcript")

st.info(transcript)

st.subheader("📈 Acoustic Metrics")

for key, value in results.items():

st.write(f"\*\*{key}:\*\* {round(value, 2)}")

st.subheader("🧠 Feedback")

feedback = give\_feedback(results, wpm)

if feedback:

for tip in feedback:

st.write(f"{tip}")

else:

st.success("✅ Great job! Your vocal delivery and quality look excellent.")

header()

main()

## Appendix 10 – ReACT & MRKL Application Source Code

import os

from dotenv import load\_dotenv

import streamlit as st

from langchain.agents import AgentExecutor, create\_react\_agent

from langchain\_openai import ChatOpenAI

from langchain.callbacks.streamlit import StreamlitCallbackHandler

from langchain\_core.prompts import PromptTemplate

from langchain\_openai import OpenAIEmbeddings

from langchain\_community.vectorstores import FAISS

from langchain\_core.tools import tool

@tool

def search\_knowledge\_base(query: str) -> list:

"""

Search the internal knowledge base when in need of more information about products, services, or general inquiries.

Search one word at a time, or multiple terms separated by commas.

Searching the same word will return the same results.

Args:

query (str): The user's query to search for relevant information.

Returns:

list: A list of relevant documents from the knowledge base.

"""

embedding\_model = OpenAIEmbeddings(model="text-embedding-3-small")

if "," in query:

items = [item.strip() for item in query.split(",")]

elif " " in query:

items = [item.strip() for item in query.split(" ")]

else:

items = [query]

results = []

for item in items:

db = FAISS.load\_local("resources/vectorstore", embedding\_model, allow\_dangerous\_deserialization=True)

results.append(db.similarity\_search(item, k=3))

return results

PROMPT = PromptTemplate.from\_template("""

You are Tudor, an intelligent and friendly AI assistant for a fictional online electronics retailer called ABC Store.

Answer the following questions.

You have access to the following tools:

{tools}

Use the following format:

Question: the input question or questions you must answer; always split the input into multiple questions if it contains multiple questions

Thought: your reasoning

Action: the action to take, should be one of [{tool\_names}]

Action Input: the input to the action

Observation: the result of the action

... (this Thought/Action/Action Input/Observation can repeat a maximum of 3 times in order to answer the question or questions; after the 3rd time answer the question and skip the Action/Action Input/Observation steps and go straight to the Final Answer)

Thought: I now know enough to answer the question

Final Answer: the final output formatted nicely and clearly using markdown

Start by thinking about the input question or questions, then use the tools to gather information if needed, and finally provide a comprehensive answer in a timely manner.

Question: {input}

{agent\_scratchpad}

""")

def header():

st.set\_page\_config(page\_title="ReACT & MRKL", page\_icon=":robot:")

st.title("ReACT & MRKL")

st.write("This page demonstrates the ReACT (Reasoning and Acting) and MRKL (Modular Reasoning and Knowledge Learning) agent capabilities.")

load\_dotenv()

def main():

st.subheader("Interact with the ReACT & MRKL agent")

init()

def init():

llm = ChatOpenAI(model="gpt-4.1-mini", streaming=True)

tools = [search\_knowledge\_base]

agent = create\_react\_agent(llm, tools, PROMPT)

agent\_executor = AgentExecutor(agent=agent, tools=tools, handle\_parsing\_errors=True)

if prompt := st.chat\_input():

st.chat\_message("user").write(prompt)

with st.chat\_message("assistant"):

st\_callback = StreamlitCallbackHandler(st.container())

response = agent\_executor.invoke(

{"input": prompt}, {"callbacks": [st\_callback]}

)

st.write(response["output"])

header()

main()