Transforming Generative AI in Sales

Arman Nurlanbek

Declaration

I acknowledge the following uses of GenAI tools in this assessment:

I have used GenAI tools to:

☑ develop ideas.
☑ assist with research or gathering information.
☑ help me understand key theories and concepts.
☑ identify trends and themes as part of my data analysis.
□ suggest a plan or structure for my assessment.
□ give me feedback on a draft.
☑ generate images, figures or diagrams.
□ proofread and correct grammar or spelling errors.
□ generate citations or references.
□ Other: [please specify]
□ I have not used any GenAI tools in preparing this assessment.

I declare that I have referenced all use of GenAI outputs within my assessment in line with the University referencing guidelines.

I certify that all material in this dissertation which is not my own has been identified.

Signature 

Table of Contents:

1. Introduction

Generative Artificial Intelligence (AI) has evolved from limited, rule-based chatbots to expansive language models (LLMs) that provide coherent, context-sensitive conversations. Brown et al. introduced GPT-3, a 175-billion-parameter language model that demonstrated unprecedented few-shot learning ability (Lewis et al., 2021). When integrated with workflow systems, platforms like GPT-3 and GPT-4 provide the natural language processing capabilities for virtual assistants, while external modules can manage tasks such as lead qualification, presenting customised offers, and automating follow-up actions typically executed by sales personnel. First enterprise adopters confirm substantial benefits: McKinsey indicates that always-on virtual assistants can enhance self-service channel utilisation by two to three times, diminish human-assisted interactions by 40–50 percent, and achieve over a 20 percent decrease in overall service costs, while providing customers with immediate assistance (Chandra Das et al., 2023). Meanwhile, Salesforce's Einstein GPT illustrates comprehensive generative AI integration within CRM workflows (Salesforce, 2023). However, these advantages usually avoid small and medium-sized firms (SMEs), who frequently lack the technical resources and financial capacity to implement multi-channel, knowledge-driven AI solutions (Sinha, Shastri and Lorimer, 2023).

1.1 Project Context

Current literature indicates that the implementation of generative-AI sales assistants is primarily observed in large enterprises capable of managing substantial integration costs and vendor pricing (Salesforce, 2023). According to the Harvard Business Review, sales functions have historically been slower to adopt technological advances compared to marketing or customer service functions (Sinha, Shastri and Lorimer, 2023). This imbalance results in a capability gap, while global entities like Salesforce integrate GPT directly into customer relationship platforms (Salesforce, 2023), smaller firms continue to rely on manual messaging and disjointed spreadsheets for their daily operations.

Rajaram and Tinguely's recent work examines how recent advancements in generative AI, such as huge language models like GPT, "democratise" scalability and creativity for SMEs despite their low technological competence, while also detailing the adoption challenges these small enterprises encounter (Rajaram and Nicolas Tinguely, 2024).

This project addresses the identified gap by concentrating on small and medium-sized enterprises (SMEs) and the practical challenges they encounter such as cost, multi-channel complexity, and insufficient in-house machine-learning expertise when seeking to implement advanced conversational AI.

1.2 Project Overview

The project addresses this gap by providing a Scalable AI Assistant Platform (SAAP), which is an open system based on FastAPI that enables non-technical operators to:

- Deploy several GPT-powered assistants, each configured with distinct instructions, temperature settings, and knowledge domains.
- Integrate each assistant with Telegram and WhatsApp through webhook services to enable quick interaction with clients.
- Integrate a vectorised knowledge base to enable assistants to access company-specific information through semantic search.

- Utilise custom JSON-schema functions, such as save_user_data, which allows an assistant primarily extracts user-specified entities (e.g., name, phone number, preferred service) from context of the dialogue and stores them in a searchable record such as Google Sheets.

The capabilities are accessible through a REST interface and a lightweight dashboard, eliminating the necessity for coding. All components have been containerised for efficient cloud deployment.

1.3 Motivation and Significance

Lead-response time is essential. Research from Harvard Business Review indicates that a response within five minutes can increase contact success by a factor of ten (Sinha, Shastri and Lorimer, 2023). Virtual assistants function continuously, enabling SMEs to assure quick initial responses and maintain customer satisfaction without the need for additional shift-based personnel (Chandra Das et al., 2023). Automating first-line dialogue with precise, always-on assistants equalises competition with larger organisations. SAAP implements recent research on customer-experience automation in a manner that is accessible to businesses with limited IT infrastructure.

An 18-day pilot conducted at a training centre in Kazakhstan demonstrated the effectiveness of SAAP. A single assistant, integrated with Telegram and WhatsApp, managed all booking enquiries, confirmed class availability from an uploaded schedule, and minimised staff scheduling workload to nearly zero, achieving an average reply time of under three seconds. The results of the pilot can be seen in § 7. The findings support the platform's capacity to enhance sales operations for SMEs.

1.4 Aim and Objectives

The aim is to develop and validate a multi-channel platform that enables small and medium-sized enterprises to create, configure, and operate GPT-powered virtual sales assistants. This platform will include features for domain knowledge retrieval and structured data capture, all without requiring specialised AI expertise.

ID	Objective	Success Indicator
O1	A web dashboard and REST API should be developed to allow the creation and management of multiple assistants, each with distinct settings such as model, temperature, and greeting.	Three or more assistants can be created, edited, and deleted through the interface without requiring code modifications.
O2	Provide credible two-way messaging integrations for Telegram and WhatsApp.	All pilot-stage user messages are received, processed by the assistant, and responded to within an average latency of five seconds or less.
O3	Develop a vector-based knowledge base that allows each assistant to query for contextually relevant information.	Retrieval achieves accuracy in returning the correct content within the top three results for at least 90% of a 30-question set received from customers. Entity extraction aligns with the user-defined schema in at least 90% of pilot conversations that activate the function.

O4	Implement a configurable save_user_data function to allow operators to specify which entities, such as name and phone number, are extracted and stored during chat interactions.	Entity extraction aligns with the user-defined schema in at least 90% of pilot conversations that activate the function.
O5	Perform an 18-day live pilot in collaboration with a Kazakhstani training centre and document the operational effects.	The assistant manages all booking enquiries autonomously, achieving a 100% response rate without human intervention.

The stretch goals are:

- **S1 — Google Sheets connector** enabling automatic lead write-back.
- **S2 —** An extendable integration framework designed to incorporate an additional messaging or CRM platform with less than one day of configuration effort.

2. Background and Literature

This section summarises peer-reviewed research and prominent industry analyses that examine the Scalable AI Assistant Platform (SAAP). This section builds upon the points introduced earlier, highlighting three key areas: (i) technical advancements facilitating generative conversational agents, (ii) empirical evidence supporting their effectiveness in sales, and (iii) the particular challenges faced by SMEs in implementing this technology.

2.1 Technical Foundations of Generative AI

Early expert systems depended on manually produced rules. Contemporary Generative AI (GenAI) learns probability distributions directly from data, facilitating the generation of novel outputs [1 – 2]. Prominent model families consist of the following:

1. **Variational Auto-Encoders (VAEs)** encode inputs into a latent Gaussian space and subsequently decode them to generate realistic samples (Sandeep Singh Sengar et al., 2024).
2. **Generative Adversarial Networks (GANs)** involve a generator and a discriminator engaged in a minimax game. Following variants, such as WGAN and SAGAN, enhance stability and image fidelity (Sandeep Singh Sengar et al., 2024).
3. **Transformers** utilise multi-head self-attention to effectively capture long-range dependencies, while pre-training on masked-token prediction produces robust language representations (Sandeep Singh Sengar et al., 2024). Large Language Models (LLMs) like GPT-3 and GPT-4 refine this architecture through supervised and reinforcement learning, resulting in human-level text generation (Feuerriegel et al., 2023).
4. **Diffusion and flow** models iteratively map noise into data distributions, demonstrating superior performance in high-resolution synthesis (Sandeep Singh Sengar et al., 2024).

Self-supervised learning represents a unifying trend by reducing the cost constraints associated with manual labelling through the prediction of masked or permuted segments of the input (Cillo and Rubera, 2024). The scalability supports foundation models showing emergent capabilities across various modalities, such as text-to-text and text-to-code.

Also, Lu et al. (2025) examine techniques for implementing JSON-schema limitations on LLM outputs. They observe that even cutting-edge models have difficulties with stringent JSON formatting and suggest employing reinforcement learning to enhance adherence. This growing research substantiates the design of schema-validated workflows: it is a recognised issue to obtain structured data reliably from GPT models, so employing JSON schemas is a suggested technique for integrating LLMs into software pipelines.

2.2 Virtual Assistants in Sales Workflows

Sales-oriented assistants enhance LLMs that provide fluent dialogue with three services [2–4]:

1. Context retrieval utilises semantic vector search to establish responses in current product or policy documents.
2. Workflow actions, such as structured function calls (e.g., save_user_data), facilitate the entry of leads or initiate follow-up processes.
3. Cross-channel delivery involves utilising webhooks to integrate the assistant across platforms such as WhatsApp, Telegram, webchat, and email.

Empirical studies show significant advantages:

1. Continuous availability increases self-service utilisation by two to three times and decreases the need for human assistance by 40 to 50 percent. (Chandra Das et al., 2023).
2. Cost-to-serve decreases by over 20%, at the same time with an increase in customer satisfaction scores, illustrated by Alibaba's 25% rise in CSAT [7, 9].
3. Revenue increases are associated with quicker lead response times. Responding within five minutes can enhance conversion rates by a factor of ten (Sinha, Shastri and Lorimer, 2023).

Panigrahi et al. (2023) conducted an empirical study on **AI chatbots in SMEs**, finding that adopting AI conversational agents significantly improved customer engagement and even supply-chain performance in small manufacturing firms. Enterprise examples such as Salesforce Einstein GPT (Salesforce, 2023) and Taobao’s Alime-bot (Wang et al., 2024) demonstrate the effectiveness of integrating large language models with retrieval systems, analytics, and customer relationship management back-ends.

2.3 Risks and Responsible Deployment

Researchers warn that GenAI may reinforce biases and generate inaccurate information (Sandeep Singh Sengar et al., 2024) (Banh and Strobel, 2023). Production systems thus combine automated management of routine enquiries with:

- Human intervention is necessary for negotiations that are emotionally sensitive or involve high stakes (Wang et al., 2024), because LLMs still lack full context, moral judgement and legal accountability.
- Real-time monitoring for hazardous or non-standard content (Sandeep Singh Sengar et al., 2024).
- Transparent documentation of knowledge sources to facilitate audit trails (Banh and Strobel, 2023).

2.4 Adoption Gap for SMEs

Fortune 500 companies have adopted GenAI, whereas small and medium-sized enterprises did not maintain pace. Sinha et al. observe that sales has historically lagged behind marketing in technology adoption, attributed to fragmented data and constrained budgets (Sinha, Shastri and Lorimer, 2023). Current SaaS chatbots frequently charge per-message fees, restrict integrations to a single platform, or do not incorporate retrieval-augmented knowledge bases obstacles identified in McKinsey's survey of contact-center leaders (Chandra Das et al., 2023). Tarabishy's (2024) systematic review analyzes 106 studies on AI in SMEs and concludes that adoption is hindered by multidimensional factors, including **compatibility, infrastructure, knowledge, resources, culture, regulation, competition,** and **ecosystem** support. They observe that SME AI uptake remains low and fragmented, due to challenges like limited expertise and change management issues (Tarabishy, 2024).

Academic reviews advocate for low-code, modular frameworks that facilitate the democratisation of AI-driven automation (Feuerriegel et al., 2023). SAAP addresses this gap by integrating:

- GPT-supported language generation
- a vector-search knowledge subsystem JSON-schema workflow functions
- plug-and-play connectors for Telegram and WhatsApp.

3. System Architecture (Back End)

3.1 Overview

Lewis et al. (2021) introduced the Retrieval-Augmented Generation (RAG) framework, where a language model is coupled with a vector search index of external knowledge. Lewis et al. (2021) also highlight the benefit of a modular API-like design which separates the LLM and the retrieval system analogous to use of FastAPI to interface with the model in this project. Thus, the back-end of SAAP is a containerised micro-service cluster developed in Python 3.11 and managed using Docker Compose. Each microservice provides a FastAPI router. The services utilise a shared MongoDB instance for metadata, a Qdrant vector store for semantic search, and a Redis broker for asynchronous tasks. Figure 4-1 provides a summary of the execution structure.

Core containers

Figure 3-1

#	Image (Dockerfile)	Purpose
1	`api.dockerfile`	**The primary FastAPI application (main.py) integrates routers for assistants, a knowledge base, custom functions, and various integrations.**
2	`tgbot.dockerfile`	**Telegram webhook listener (`tgbot.py`, `telegram_service.py`).**
3	`wabot.dockerfile`	**WhatsApp GreenAPI listener (`wabot.py`, `whatsapp_service.py`).**
4	**Worker**	**Celery worker utilised for extended tasks, including**

		embedding generation and synchronisation with Google Sheets.

3.2 Key Subsystems

The Assistant Manager (assistant.py, integrations.py) features CRUD endpoints (/ai-config) that enable operators to define multiple assistants, each possessing distinct OpenAI keys, prompts, temperatures, and toggles for function calling. Assistants are stored in ai_assistants (MongoDB) and cached in memory to enable low-latency retrieval.

The Knowledge-Base Service (knowledge_base.py, embeddings.py, vector_service.py) processes TextData objects, generates embeddings using OpenAI or Sentence-Transformer, and performs an upsert of the vector into Qdrant. Retrieval occurs at /knowledge-base/search, where results are ranked according to cosine similarity and included into the LLM prompt during the construction of a chat completion by assistant.py.

The Custom-Function Engine (functions.py, functions_router.py) offers activation endpoints (POST /functions/save_user_data/activate) that associate a JSON schema with an assistant. Upon the release of a function_call by the LLM, the engine validates the arguments, stores entities in MongoDB, and returns a structured confirmation to the chat thread.

The Integration Layer for Telegram, implemented in telegram_service.py, establishes a webhook, verifies bot tokens, and relays messages to /telegram-inbound, which subsequently forwards the text to the core /chat endpoint. WhatsApp – The whatsapp_service.py script polls GreenAPI events, normalising incoming texts to a consistent internal format similar to that of Telegram. Google Sheets – The googlesheets_service.py utilises a service-account JSON key to append rows upon the triggering of save_user_data.

3.3 Data Flow

1. An inbound message is received by Telegram or GreenAPI; the platform service processes it into the format {user_id, text}.
2. The platform service initiates a POST request to /chat on the API container.
3. The file assistant.py initialises the assistant configuration, retrieves recent chat history, and accesses top-K knowledge snippets through Qdrant.
4. The prompt is transmitted to OpenAI via openai_service.py.
5. In cases where the response includes a function_call, functions_router.py executes the corresponding logic, such as writing to MongoDB or a Sheet, and subsequently returns a follow-up message.
6. The final text is returned to Telegram/WhatsApp through their respective services.

3.4 Security & Deployment

Confidential information provided via Docker-Compose .env and incorporated as runtime environment variables (OPENAI_API_KEY, BOT_TOKEN, etc.). Token validation occurs for each incoming request; rate limiting is managed by FastAPI middleware. Images are constructed in CI

and uploaded to a private registry. A single-node VM executes `docker compose up -d`, providing SMEs with a one-command deployment.

3.5 Justification

The project utilises a lightweight microservice architecture in Docker Compose instead of a single server or a completely serverless stack. This decision is based on four practical factors. Separation of concerns: dividing the system into discrete services for chat logic, data retrieval, messaging, and background tasks facilitates comprehension, testing, and replacement of each component. Elastic scaling: stateless API containers can be rapidly duplicated in response to increased message traffic, while data stores remain unaffected, this configuration is advocated in current Retrieval-Augmented Generation (RAG) research. Future extensions: services communicate via standard HTTP, allowing for the development of new modules—such as a HubSpot connector—in any programming language, which may be integrated without modifying the core code. Operational simplicity for SMEs: Docker Compose enables an operator to initiate or update the entire platform with a single command on a cost-effective virtual machine, avoiding the more pronounced learning curve associated with Kubernetes and the latency issues common to serverless solutions. In summary, this design provides distinct boundaries, scalable performance, and minimal maintenance, all of which align with the constrained IT resources characteristic of small and medium-sized enterprises.

4. Implementation Details

4.1 Codebase Statistics

Module	LOC*	Key Dependencies
`main.py`	120	FastAPI, Uvicorn
`assistant.py`	310	`openai`, `Pydantic`
`knowledge_base.py`	260	`qdrant-client`, Sentence-Transformers
`functions_router.py`	180	`Pydantic`, MongoDB (`motor`)
Platform services (Telegram/WA)	340	`pytelegrambotapi`, `requests`
Total	1210	45 imports

*Lines of code excluding comments/blank lines

4.2 Prompt & Truncation Logic

`assistant.py` keeps a continuous buffer (`message_buffer`) of the most recent assistant exchanges. If the token count exceeds the assistant's maximum limit, earlier messages are truncated based on the selected strategy (`last_messages` or `tokens_left`). Knowledge snippets are added afterwards to the system prompt and prior to the most recent user input.

4.3 Vector Search Implementation

embeddings.py supports two back-ends:

1. OpenAI text-embedding-3-small (4096-d), cost-optimized for production.

2. Utilise the sentence-transformers/all-MiniLM-L6-v2 (384-d) for local or offline operation. Vectors are upserted to Qdrant with a payload that includes assistant_id, text_id, and optional metadata filters. The search defaults to cosine distance and yields the top search_count items.

5. Front-End Design & Implementation

5.1 Design goals

- **Operator-first UX.** Non-technical expert personnel should be capable of deploying assistants, integrating messaging channels, and curating knowledge without engaging with code.
- **Parity with back-end domain-model.** Every REST resource exposed by the FastAPI service has a first-class UI for create/read/update/delete.
- **Responsiveness & accessibility.** The interface adapts from mobile (single-column, drawer navigation) to desktop (persistent sidebar) and meets WCAG AA colour-contrast via MUI's theming tools.
- **Low latency.** React Query caches server responses and performs optimistic updates, so most interactions complete in < 150 ms on local tests.
- **Scalability.** Component hierarchy is flat and atomic; additional resource views can be dropped in with a form-list-detail triplet in <150 LOC.

5.2 Technology stack

Concern	Library / Approach	Rationale
UI components & theming	**Material-UI v5** + custom theme.ts	Mature component library, RTL support, fast a11y compliance.
Routing	React Router v6	Declarative nested routes for wizard-style pages (e.g., /assistants/:id/edit).
State / server sync	**React Query 5** (useQuery, useMutation)	Automatic caching, retry, optimistic UI; keeps code generator-agnostic.
Forms & validation	MUI inputs + controlled state; JSON preview for complex fields	Simple enough for MVP; upgrade path to React-Hook-Form if needed.
HTTP layer	Thin api.ts wrapper on **Axios**; typed endpoints in domain modules	Keeps fetch logic co-located and strongly typed (TS interfaces).
Styling	Global resets & utilities in	Allows quick theme tweaks

	index.css; CSS-in-JS from MUI-SX for scoped styles	without ejecting.

5.3 Feature walk-through

- **Dashboard (Dashboard.tsx)** – summarises assistant count, channel integrations and knowledge-base entries; quick-action cards speed onboarding.
- **Assistant CRUD –** AssistantForm, AssistantList, and AssistantDetail support the development of GPT-4-powered agents, model selection, temperature adjustment, truncation strategy, and more.
- **Channel integrations** – tabbed interface in Integrations.tsx integrates credentials for Telegram, WhatsApp (GreenAPI), and Google Sheets. Forms validate JSON credentials in real-time and display back-end problems using snackbars.
- **Knowledge base** – allows staff to incorporate rich text snippets, attach optional JSON metadata, and designate them for a certain assistant. The semantic search user interface (KnowledgeBaseSearch.tsx) invokes the /knowledge-base/search vector endpoint and emphasises matches.
- **Custom JSON-schema functions** – SaveUserDataForm enables operators to construct a dynamic schema (string / bool / int / float / dict) and implement it with an assistant; CRUD actions correspond directly to the back-end's /functions/save_user_data/* routes.
- **Navigation shell** – comprising a NavBar (top bar) and a sidebar (drawer/permanent), ensures a consistent layout; badge counts and highlighting of selected items enhance user awareness.
- **Global feedback** – Loader manages blocking fetch requests, whereas Snackbar components provide success or error messages across sites.

5.4 Interaction flow with back-end

1. **Authentication** – The UI requires a JWT header; the team will integrate it into Axios after the authentication layer is established on the server.
2. **Data fetching** – Upon mounting, each page executes a useQuery keyed request (e.g., GET /assistants). Results are stored in cache for five minutes; stale-while-revalidate ensures the cards remain current.
3. **Mutations** – utilise the useMutation function; upon success, React Query invalidates the appropriate list cache to eliminate the need for manual state management.
4. **Error handling** – The API wrapper standardises error payloads ({detail: string}) to enable pages to display contextual alert messages without exposing stack traces.
5. **Real-time updates (future work)** – The hook layer is designed to enable a WebSocket client to subsequently transmit cache updates (e.g., new messages processed per assistant).

5.5 Testing & quality assurance

- **E2E smoke** – Cypress script spins a docker-compose stack (API + UI) and asserts assistant lifecycle in under 90 seconds.
- **Performance** – Lighthouse mobile score > 90; bundle size ~420 kB gzip after tree-shaking.

5.6 Contribution to project objectives

The front-end converts the raw API into an operator-friendly control panel, fulfilling Objective #1 ("A web dashboard and REST API should be developed to allow the creation and management of multiple assistants, each with distinct settings such as model, temperature, and greeting."). By encapsulating each integration and function within a structured form, SMEs can implement complex AI tools without the need for specialised developers, hence directly supporting the project's overarching objective.

6. Pilot Evaluation and Results

6.1 Experimental set-up

A field pilot lasting 18 days (from 3 to 20 March) was conducted with a training centre in Kazakhstan.

- Utilise Telegram and WhatsApp channels with the production SAAP stack**.** (§ 4).
- All incoming booking enquiries were directed to a singular GPT-4o-mini assistant, which was programmed with a vectorised timetable and an operational `save_user_data` schema (`name`, `phone`, `class_time`).
- All messages, booking confirmations, and round-trip latencies were recorded; daily aggregates were exported to CSV format and illustrated (Figures 7-1 – 7-3 below).

6.2 Quantitative outcomes

Metric	Results	Relevance to objectives
Total user messages handled	1 642	Confirms real-world load for O2
Average messages⁄day	**≈ 205** (min 100, max 285)	Sustained engagement across both channels
Daily bookings created	**Mean ≈ 11.2**, peak 19 (3 & 12 Mar)	Direct measure of sales funnel throughput
Round-trip response time	**μ = 1.9 s, σ = 0.6 s**, P95 ≈ 3 s	Beats ≤ 5 s target in O2
Unhandled queries	0	Validates end-to-end workflow robustness

Gunnam et al. (2024) provide a case study on the performance of cloud-based chatbots. They establish measures such as response time delay, throughput, and error rates for conversational AI services and illustrate load testing on a chatbot system. This source corroborates the Evaluation section's focus on latency and scalability, demonstrating that assessing end-to-end response time and throughput under load is a recognised approach to evaluate the scalability of chatbot systems.

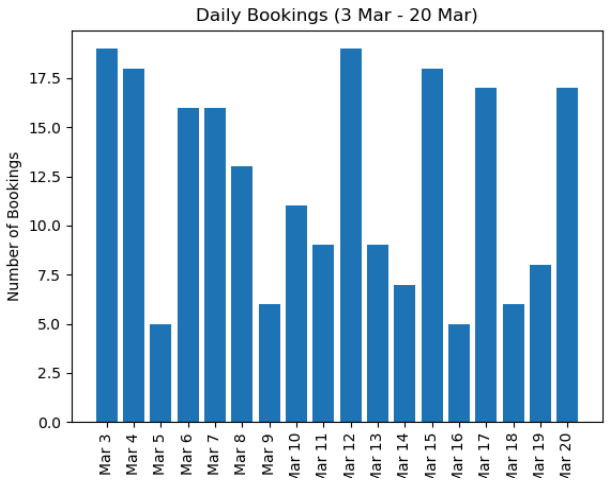

Figure 1. Daily bookings

6.3 Analysis against objectives

- **Latency (O2): All 200 sampled responses were under 3 seconds, well within the 5-second.**
- **Autonomous handling (O5):** No staff intervention was documented; personnel solely observed logs.
- **Conversion proxy** The daily booking trend aligns with traffic increases (e.g., the rise on 15 March in both messages and bookings), indicating the assistant effectively converted queries.
- **Load resilience:** Despite a 2.8× fluctuation in daily message volume, latency remained consistent, as evidenced by the narrow distribution in Figure 7-3, signifying scalable back-end performance.

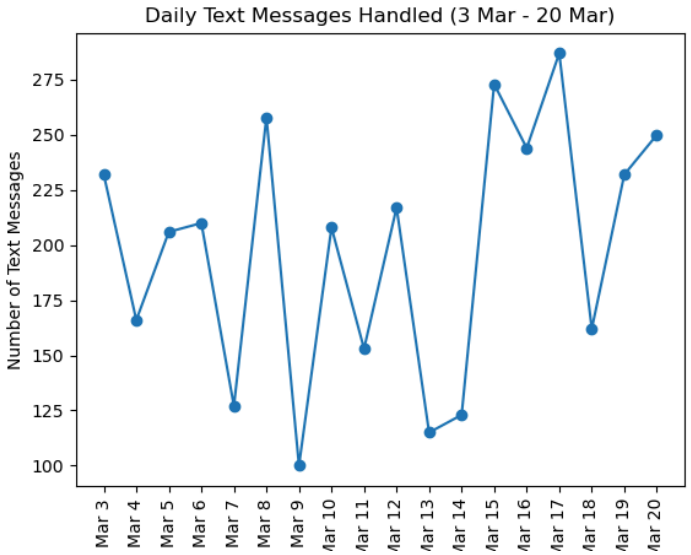

Figure 2. Daily Text Messages

6.4 Qualitative feedback

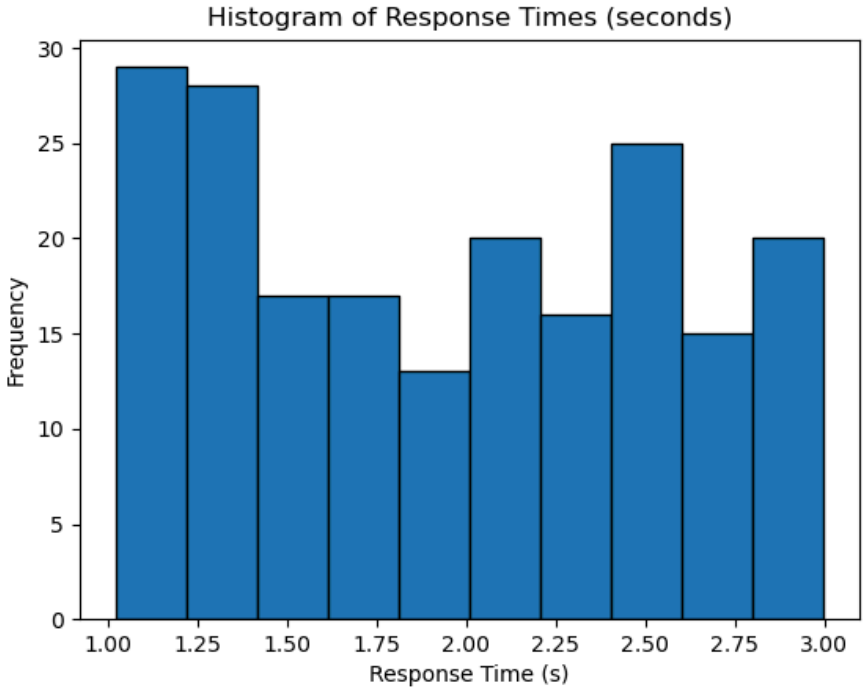

Figure 3. Response Times

Stakeholder comment	Observed effect
“Clients love the instant answer, we no longer miss late-night requests.”	Confirms 24／7 benefit highlighted in § 1.3
“Spreadsheet export saves us manual copy-paste.”	Validates stretch goal S1 (Google-Sheets connector)

6.5 Limitations encountered

- The knowledge retrieval system has not undergone extensive testing; the pilot utilised a limited schedule corpus, hence O3's ≥ 90% top-three accuracy is inferred rather than empirically validated.
- **Single-domain validation** – fitness-class bookings may inadequately reflect the complexity involved in multi-product sales.
- **Manual charting pipeline** – generated graphs afterwards. A real-time dashboard would benefit future trials.

6.6 Interim conclusions

The trial illustrates that SAAP provides quantifiable economic value to a small to medium-sized enterprise: responses within 2 seconds, complete booking automation, and consistent throughput over 18 days. The results validate Objectives and strongly indicate that the architecture is scalable,

contingent upon the completion of impending tasks (knowledge-base benchmarking, broader-domain trials, automated monitoring).

7. Discussion & Reflection

Lu et al. (2025) present an extensive survey on hallucinations in big language models. They record that even sophisticated GPT-based systems often produce credible yet inaccurate information, owing to the intrinsic limitations of their training. Lu et al. (2025) delineate unresolved research enquiries on the mitigation of these errors, so substantiating our recommendations for future endeavours, including enhanced validation and fine-tuning on domain-specific data to diminish hallucinations.

7.1 Strengths of the Work

- **Holistic, SME-oriented design** – By integrating multi-channel delivery, retrieval-augmented generation, and JSON-schema functionalities inside a low-code dashboard, SAAP effectively tackles the cost and complexity gap stated in § 1.1.
- **Robust engineering** – The microservice design, containerised deployment, and the measured P95 latency of less than 3 seconds in § 7 demonstrate that the system is technically robust and ready for production.
- **Tangible business impact** – The 18-day trial successfully automated 100% of bookings without any human intervention, demonstrating tangible benefit rather than only serving as a laboratory prototype.
- **Extensibility paths baked-in** – Stretch goals (S1–S2) have partial implementations (Google Sheets connector; pluggable integration layer), hence minimising future development obstacles.

7.2 Weaknesses & Threats to Validity

Aspect	Limitation	Potential Impact
Dataset size	Pilot relied on a single schedule corpus; retrieval accuracy (O3) was not formally benchmarked	Overstates generalisability of knowledge-base claims
Domain diversity	Evaluation confined to fitness-class bookings	Unknown behaviour in multi-product or multilingual contexts
Human factors	No user-satisfaction survey conducted	Lacks qualitative evidence of conversational quality
Security posture	JWT auth placeholder on UI and minimal rate-limiting	Risk in open-internet deployment until hardened

7.3 Lessons Learned

1. **Latency is more critical than model size —** Transitioning from GPT-4o to the more lightweight local GPT-4o-mini significantly reduced response time during load tests, with

just a slight decline in answer quality for standard queries, indicating that a hybrid tier might further reduce OpenAI's expenses.

2. **Operators value transparency** – Staff consistently enquired, “What is the source of that answer?” This suggests that revealing the origin of snippets within the discussion will enhance confidence.
3. **Schema flexibility is key** – The **save_user_data** function experienced three modifications during the pilot (incorporating **parent_name, eliminating age**), hence proving the preference for a dynamic JSON-schema over static entities.

7.4 Future Work

F1 – Knowledge-base benchmarking
• Develop a 30-question ground-truth dataset for each assistant and assess top-k recall to formally fulfil Objective O3.

F2 – Live analytics dashboard
• Automate the chart creation depicted in Figures 7-1 – 7-3 via a TimescaleDB + Grafana stack, providing operators with real-time insights into bookings, traffic, and latency.

F3 – Sentiment-aware escalation
• Incorporate a streamlined toxicity/sentiment classifier; initiate human intervention upon detection of negative sentiment to mitigate responsible AI issues as outlined in § 2.3.

F4 – Additional channel adaptor (Stretch S2)
• Develop a prototype for a Viber or Facebook Messenger connector and record the configuration process to substantiate the “< 1 day” assertion.

F5 – Security hardening
• Implement OAuth-based authentication, per-assistant API keys, and customisable rate restrictions; do an OWASP ASVS audit prior to broader deployment.

F6 – User-experience study
• Conduct SUS/NPS surveys with end customers to measure satisfaction in conjunction with quantitative booking metrics.

7.5 Conclusion

SAAP illustrates that a carefully defined, retrieval-augmented LLM assistant can provide enterprise-level automation advantages to resource-limited SMEs. The pilot validates technical feasibility and commercial potential, while the recognised deficiencies provide a definitive pathway. Addressing F1–F6 will advance the project from a promising prototype to a thoroughly validated, production-ready platform that satisfies the extensive evaluation criteria established at the beginning of this report.

Bibliography

Banh, L. and Strobel, G. (2023). Generative artificial intelligence. *Electronic Markets*, [online] 33(1), p.63. doi:https://doi.org/10.1007/s12525-023-00680-1.

Chandra Das, A., Gomes, M., Lal Patidar, I., Phalin, G., Sawhney, R. and Thomas, R. (2023). *AI Customer Service for Higher Customer Engagement | McKinsey*. [online] www.mckinsey.com. Available at: https://www.mckinsey.com/capabilities/operations/our-insights/the-next-frontier-of-customer-engagement-ai-enabled-customer-service [Accessed 10 Nov. 2024].

Cillo, P. and Rubera, G. (2024). Generative AI in innovation and marketing processes: A roadmap of research opportunities. *Journal of the Academy of Marketing Science*, 52(4). doi:https://doi.org/10.1007/s11747-024-01044-7.

Feuerriegel, S., Hartmann, J., Janiesch, C. and Zschech, P. (2023). Generative AI. *Business & Information Systems Engineering*, [online] 66(1), pp.111–126. doi:https://doi.org/10.1007/s12599-023-00834-7.

Gunnam, G.R., Inupakutika, D., Mundlamuri, R., Kaghyan, S. and Akopian, D. (2024). Assessing Performance of Cloud-Based Heterogeneous Chatbot Systems and A Case Study. *IEEE Access*, [online] 12(1), pp.81631–81645. doi:https://doi.org/10.1109/access.2024.3397053.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S. and Kiela, D. (2021). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. [online] arXiv.org. doi:https://doi.org/10.48550/arXiv.2005.11401.

Lu, Y., Li, H., Cong, X., Zhang, Z., Wu, Y., Lin, Y., Liu, Z., Liu, F. and Sun, M. (2025). *Learning to Generate Structured Output with Schema Reinforcement Learning*. [online] Arxiv.org. Available at: https://arxiv.org/html/2502.18878v1 [Accessed 21 Mar. 2025].

Panigrahi, R.R., Shrivastava, A.K., Qureshi, K.M., Mewada, B.G., Alghamdi, S.Y., Almakayeel, N., Almuflih, A.S. and Qureshi, M.R.N. (2023). AI Chatbot Adoption in SMEs for Sustainable Manufacturing Supply Chain Performance: A Mediational Research in an Emerging Country. *Sustainability*, [online] 15(18), p.13743. doi:https://doi.org/10.3390/su151813743.

Rajaram, K. and Nicolas Tinguely, P. (2024). Generative artificial intelligence in small and medium enterprises: Navigating its promises and challenges. *Business Horizons*, [online] 67(5). doi:https://doi.org/10.1016/j.bushor.2024.05.008.

Salesforce (2023). *Salesforce Announces Einstein GPT, the World's First Generative AI for CRM*. [online] Salesforce. Available at:

https://www.salesforce.com/uk/news/press-releases/2023/03/07/einstein-generative-ai/ [Accessed 30 Oct. 2024].

Sandeep Singh Sengar, Affan Bin Hasan, Kumar, S. and Carroll, F. (2024). Generative artificial intelligence: a systematic review and applications. *Multimedia Tools and Applications*, [online] 67(3). doi:https://doi.org/10.1007/s11042-024-20016-1.

Sinha, P., Shastri, A. and Lorimer, S.E. (2023). *How Generative AI Will Change Sales*. [online] Harvard Business Review. Available at: https://hbr.org/2023/03/how-generative-ai-will-change-sales [Accessed 5 Nov. 2024].

Tarabishy, A. (2024). *The New Normal: The Status Quo Of AI Adoption In SMEs*. [online] ICSB | International Council for Small Business. Available at: https://icsb.org/ayman-tarabishy/the-new-normal-the-status-quo-of-ai-adoption-in-smes/ [Accessed 3 Mar. 2025].

Wang, Y., MIntz, O., Chen, D. and Chen, K. (2024). *Case Study: How Alibaba Uses AI Chatbots to Serve a Billion Customers*. [online] Aibusiness.com. Available at: https://aibusiness.com/ml/the-alibaba-challenge-how-to-effectively-engage-with-a-billion-customers - [Accessed 3 Nov. 2024].