iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: https://doi.org/10.22214/ijraset.2026.85130

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

VAANIX: A Real-Time Voice AI Agent Platform with Action-Enabled Agents, Automated Appointment Booking and a Business Workspace

Prof. Saba Chaughule[1], Atharv Bhosale[2], Srushti Phalke[3], Avdhooth Chaudhari[4], Srushti Magar[5]
[1]*Acting Head of Department, Department of Computer Engineering, P K Technical Campus, Chakan, Pune, Savitribai Phule Pune University (SPPU), Maharashtra, India*
[2, 3, 4, 5]*Final Year Students, Department of Computer Engineering, P K Technical Campus, Chakan, Pune, Savitribai Phule Pune University (SPPU), Maharashtra, India*

Abstract: ***Telephone support is still the main way many small businesses talk to their customers, yet it is also the channel they are least able to staff around the clock. Traditional IVR menus frustrate callers, and most of the newer AI voice products are closed, priced per minute and difficult to adapt to a specific business. This paper presents VAANIX, a self-hostable voice agent platform that connects ordinary phone calls to a streaming pipeline of speech recognition, a large language model and speech synthesis, so that a caller can simply talk and be understood. The first version of the platform focused on that conversational pipeline and a developer console. In this paper we describe a second, larger iteration. A single Agent Studio now holds everything about an agent: its prompt, voice, knowledge base, API settings, call logs, single calls and bulk calling. Agents can also be given actions at creation time. The most important of these is appointment booking, in which the agent checks availability while the caller is still on the line, books the slot, and confirms it by voice. Confirmed bookings appear immediately on a Business Workspace home screen that shows a calendar of the client's own appointments instead of generic call statistics. We explain the architecture, the action mechanism, the way double bookings are prevented, and the security design built on row level isolation. Manual testing across all modules showed that callers receive a spoken reply typically within one to two seconds and that booked slots are reflected on the calendar within moments, without the per-minute platform fees charged by managed voice agent services.***

Keywords: Voice AI Agents, Real-Time Speech Pipeline, Large Language Models, Tool Calling, Appointment Automation, Twilio Media Streams, Supabase, Business Workspace.

I. INTRODUCTION

Ask a clinic receptionist how their morning goes and the answer is usually the same: the phone rings while someone is standing at the desk, a second call arrives while the first is on hold, and a few patients give up and hang up. Most of those calls are not complicated. People want to know whether the doctor is available on Thursday, or they want a slot. A human can handle that in thirty seconds, but only when a human is free.

Phone automation has existed for decades, mostly as touch-tone menus. They are cheap to run and miserable to use, because the caller has to fit a real question into a menu someone else designed. Large language models have changed what is possible. A model can follow free-form speech, ask a clarifying question, and respond in a natural voice. Putting that into a live telephone call is harder than it sounds, because the delay between the caller finishing a sentence and hearing a reply decides whether the exchange feels like a conversation or like talking to a recorded message. Human turn-taking research suggests that the gap between speakers is only a few hundred milliseconds in most languages [11], and telecommunication standards treat one-way delays beyond roughly a hundred and fifty milliseconds as a quality concern [12]. A cascaded pipeline of streaming speech recognition, a language model and streaming synthesis cannot reach those figures, but recent practical work shows that a well-built cascade can respond within about a second [1].

VAANIX began as an attempt to build such a pipeline ourselves rather than rent it. The first version connected Twilio Media Streams to Deepgram, Claude and ElevenLabs over a WebSocket, and wrapped it in a React console where a developer could create an assistant, bind a phone number and read call logs. It worked as a talking agent, but it did not do anything. A caller could ask about an appointment and the agent could only describe how appointments usually work.

©IJRASET: All Rights are Reserved

This paper is about the step that followed. Two observations drove it. First, a voice agent that cannot act is a more pleasant IVR, and little else. Businesses want the call to end with something done, a booking made or a record created. Second, the console we had built showed generic analytics such as total calls and minutes, which a clinic owner does not care about. They care about tomorrow's appointments. The second iteration therefore adds three things: an Agent Studio that gathers every agent related task in one place, a per-agent action mechanism through which the language model can call real backend functions, and a Business Workspace that presents the results of those actions, starting with a calendar of booked and scheduled appointments.

The rest of the paper is organised as follows. Section II reviews related work. Section III describes the system architecture. Section IV explains the real-time voice pipeline. Section V covers the Agent Studio and the action mechanism. Section VI describes automated appointment booking and the Business Workspace. Section VII discusses security and data design. Section VIII reports testing and findings. Section IX lists limitations and challenges, followed by future scope, conclusions, acknowledgements and references.

II. RELATED WORK AND BACKGROUND

A. Cascaded and end-to-end voice agents

Our design follows the cascaded approach analysed by Qiu et al., who compare an end-to-end speech model against a cascade of streaming speech to text, a language model with function calling, and streaming text to speech, and conclude that the cascade is still the practical choice for self-hosted real-time agents [1]. Their tutorial uses the same Deepgram and ElevenLabs services that we use, and that work served as the base reference for VAANIX. Where they concentrate on the latency of the pipeline itself, we concentrate on what happens after the pipeline understands the caller: how an action is selected, executed and shown to the business.

B. Tool use by language models

The idea that a language model can decide when to call an external function has a short but busy history. ReAct interleaves reasoning steps with actions so that the model can look something up before answering [7]. Toolformer showed that a model can learn when an API call helps [8]. Today the capability is available directly in commercial APIs through structured tool definitions, and the Claude interface that VAANIX uses accepts a list of tools with typed arguments and returns a structured request when the model wants one executed [5]. Retrieval augmented generation, which supplies the model with passages from a private store at answer time, is the other half of the picture [9], and the knowledge base in VAANIX is a simple application of it.

C. Telephony and speech services

Twilio Media Streams forwards raw call audio over a WebSocket and accepts audio in return [2]. Streaming recognition from Deepgram returns partial and final transcripts while the caller is still speaking [3], and ElevenLabs can begin returning synthesized audio before the full sentence has been generated [4]. Large scale weakly supervised models such as Whisper have raised the baseline for recognition accuracy [10], but they are usually run on complete audio segments, which is why a dedicated streaming service remains attractive for phone calls.

D. Gap addressed by this work

Existing managed voice agent platforms hide the prompt, the model and the action logic behind a dashboard, and charge by the minute. Open tutorials show the pipeline but stop at the conversation. What a small business needs sits between the two: an agent it can configure without writing code, that books real appointments, and a home screen that shows the outcome in the business's own terms. VAANIX targets that gap.

III. SYSTEM ARCHITECTURE

VAANIX is organised into four layers, shown in Fig. 1. The telephony layer deals with the phone network through Twilio, for both inbound calls and outbound or bulk dialling. The real-time voice engine is an Express server with a WebSocket handler that moves audio between Twilio and the speech and language services. The action and business logic layer, which is new in this iteration, contains the action registry, the availability checker, the booking and calendar service, and the question and answer knowledge base. The data and presentation layer consists of Supabase PostgreSQL and the React 19 front end, which is split into the Agent Studio and the Business Workspace.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

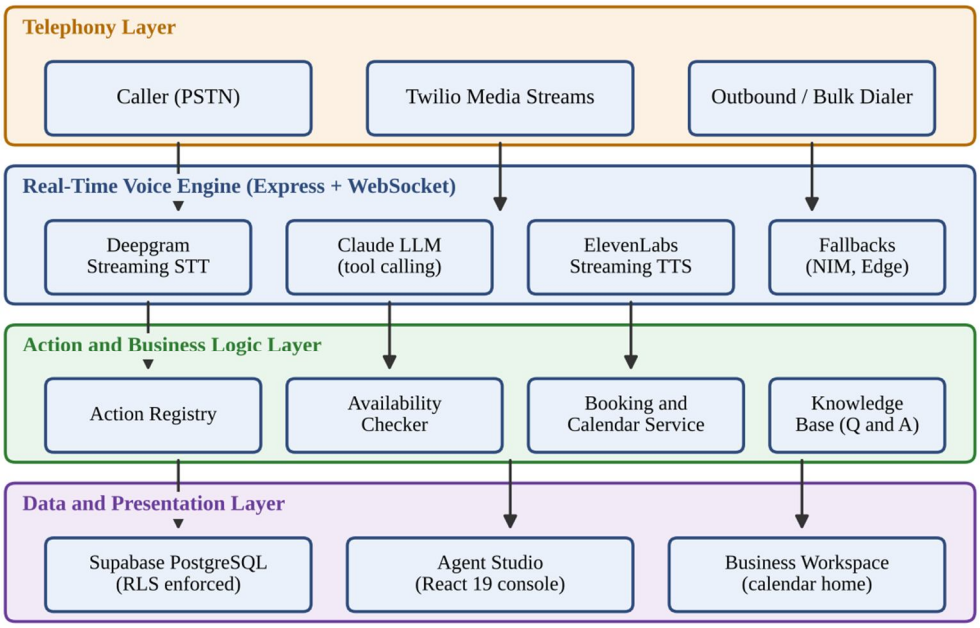


Fig. 1: VAANIX layered architecture with the action layer introduced in the second iteration

The layers were separated for a practical reason. The voice engine is latency sensitive and must never wait on slow work, so anything that is not needed to produce the next spoken sentence is kept out of it. Bookings, calendar updates and call log writes happen in the action layer and the data layer, and the voice engine only waits for them when the caller is actually asking for the result. The technology choices are summarised in Table 1.

Table 1: Technology stack

Component	Technology	Purpose
Front end	React 19, TanStack Router, Tailwind CSS 4	Agent Studio and Business Workspace
Animation and charts	Framer Motion, Recharts, Radix UI	Interface motion, widgets, accessible primitives
Server	Express 5, ws, TypeScript	HTTP routes and WebSocket audio handling
Telephony	Twilio Media Streams and REST API	Inbound, outbound and bulk calls, phone numbers
Speech to text	Deepgram streaming	Live transcription of mulaw audio
Language model	Claude (Anthropic SDK), NVIDIA NIM fallback	Dialogue and tool calling
Speech synthesis	ElevenLabs, Edge TTS fallback	Natural voice output and voice personas
Database	Supabase PostgreSQL	Users, agents, calls, bookings, knowledge base

IV. REAL-TIME VOICE PIPELINE

A. Call flow

When a call arrives, Twilio requests TwiML from the server, which answers by opening a media stream to a WebSocket endpoint. From then on the call is a loop (Fig. 2). Audio frames arrive from Twilio and are forwarded without buffering to Deepgram. As transcripts come back, the server waits for an end-of-utterance signal, assembles the dialogue so far together with the agent's system prompt, and sends it to Claude with streaming enabled. As soon as the first sentence of the reply has been generated, it is handed to ElevenLabs, and synthesized audio is streamed back to Twilio while the model is still producing the rest. The caller therefore hears the start of the answer long before the full answer exists.

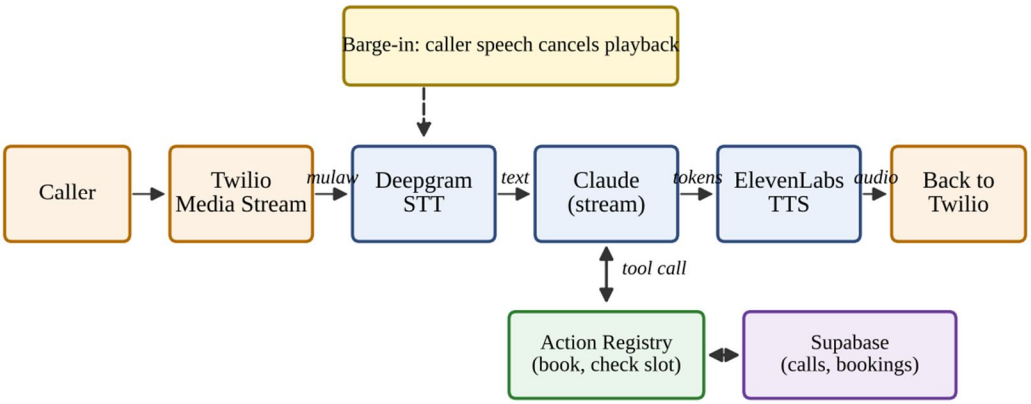


Fig. 2: Streaming voice pipeline with the action path and barge-in handling

B. Interruptions

People interrupt each other, and they do the same to an agent. If the caller starts talking while synthesized speech is being played, the server stops sending audio, clears Twilio's playback buffer and discards the unsent part of the reply, so that the next turn begins from what the caller said rather than from what the agent was in the middle of saying. Without this, the agent keeps talking over the caller, which is one of the fastest ways to make a voice product feel broken.

C. Fallbacks

External services fail in different ways at different times, so the pipeline is built from replaceable parts. If the primary synthesis service is unavailable, Edge TTS is used as a free fallback, with a plainer voice. If the primary language model endpoint cannot be reached, the NVIDIA NIM endpoint, which exposes an OpenAI compatible interface, can take over. The fallback is chosen per agent in its settings, and the choice is recorded with each call so that an unusual transcript can be traced back to the model that produced it.

V. AGENT STUDIO AND THE ACTION MECHANISM

A. Why a single studio

In the first iteration, the pieces that make up an agent were scattered. Prompts lived on one page, phone numbers on another and call history on a third. A person setting up an agent for a client had to move between all three to test one change. The Agent Studio replaces this with one tab, organised around the agent. Inside it, a user can create and edit the agent, write its system prompt, choose a voice, maintain the knowledge base, set API related options, read the call logs for that agent, place a single test call, and start a bulk calling run. The main tab of the application is deliberately left free for something else, which is described in Section VI.

B. Actions

An action is a named capability that the agent may use during a call. While creating an agent, the user picks the actions it needs from a list, for example appointment booking. Each selected action is stored as part of the agent's configuration, and at the start of every call the server converts that list into tool definitions for the language model. A definition has a name, a plain description of when the tool should be used, and a typed argument schema. An agent without the booking action is not shown a booking tool, and therefore cannot attempt to book. This keeps the model's decision space small and its behaviour easier to predict.

When the model asks for a tool, the server does not trust the request as it stands. It validates the arguments against the schema, adds the identifiers the model should never choose for itself, namely the agent, the owner and the call, runs the corresponding handler, and returns the handler's result to the model as the tool result. Only then does the model compose its spoken answer. Two properties follow from this. The model can never write to a record that belongs to another business, because the owner is attached by the server. And every action leaves a trace, because the handler writes it to the database together with the call it came from.

C. Describing actions to the model

We found early on that the description of a tool matters as much as its code. A vague description such as "books appointments" led the model to call the tool as soon as a caller mentioned the word appointment, before a date had been given. The final descriptions therefore state the preconditions plainly: availability must be checked first, a date and a time window must be known, and the caller's name must be collected before a booking is attempted. Each action in the registry carries this text next to its schema, so changing the behaviour of the agent is a matter of editing one description, not the pipeline.

VI. AUTOMATED APPOINTMENT BOOKING AND BUSINESS WORKSPACE

A. The booking conversation

The booking action is the one we use to show what the new architecture is for. Consider a clinic. A caller asks for an appointment with the doctor on Thursday afternoon. The language model identifies the date and a rough time window and calls the availability tool. The tool reads the clinic's configured working days, hours and slot length, subtracts the slots that are already booked, and returns the free slots in that window. The model offers two or three of them in a natural sentence. When the caller agrees to one, the model asks for the caller's name if it does not have it, and then calls the booking tool with the chosen slot. The result comes back as a confirmation or as a conflict, and the agent either confirms the appointment by voice or offers the nearest alternatives (Fig. 3).

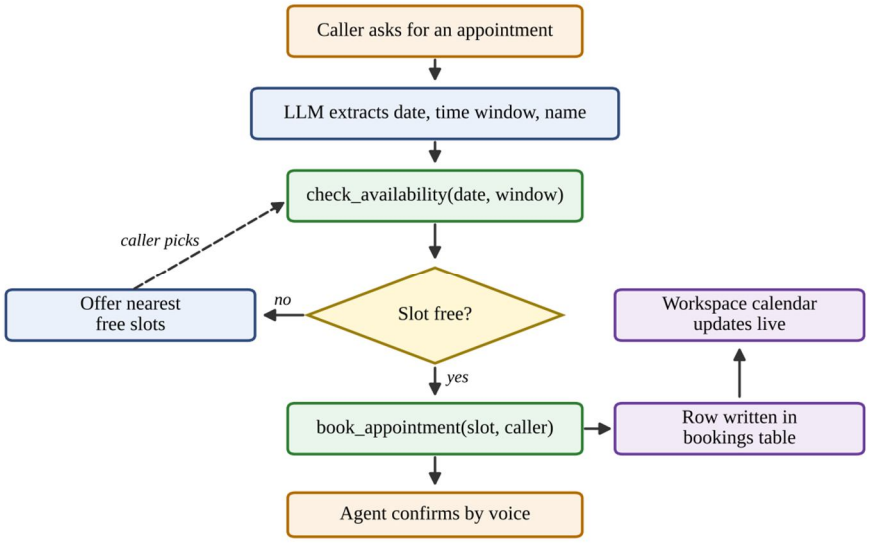


Fig. 3: Appointment booking flow from the caller's request to the calendar update

B. Preventing double bookings

Two callers can ask for the same slot at the same moment, particularly when an agent handles many calls in parallel. Checking availability and then inserting a row is not safe on its own, because both checks can pass before either insert happens. For this reason the final guarantee is placed in the database. Each booking stores its start and end as a time range, and a PostgreSQL exclusion constraint on that range, scoped to the same agent and ignoring cancelled bookings, rejects any overlapping insert. When the second caller's insert fails, the booking tool returns a conflict result, and the agent apologises and offers the next free slot. The availability check is therefore a convenience for the conversation and the constraint is the actual rule.

Time handling is a smaller but real source of bugs. Slots are stored as timestamps with time zone, while working hours are interpreted in the business's own zone, which for our clients is Indian Standard Time. The agent speaks in local time, the database stores absolute time, and conversion happens in exactly one place, in the availability module.

C. Business Workspace

The main tab of the application is now the Business Workspace. Instead of counters for total calls and minutes, it shows whatever matters to the type of client that is logged in. For a doctor it is a calendar with day, week and month views, in which each booked appointment appears as a block with the caller's name, phone number and the call it came from.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

The calendar is subscribed to changes in the bookings table, so a slot booked during a call shows up without anyone refreshing the page. Selecting an appointment opens the transcript of the call that produced it, which lets staff check what the caller was actually told. The business can also review a day's appointments before it starts.

The Workspace is designed as a home for client specific logic and not as a clinic calendar only. A different business type would place a different view there, for instance a table of table reservations or a list of captured leads, drawing on the same action and data layers. Appointment booking is the first view built, because it is the commonest request we heard from clients.

D. Bulk and outbound calling

Outbound calling lives inside the Agent Studio, next to the agent that will place the calls. A user can start a single call to a number to test an agent, or upload a list and start a bulk run, for example to remind patients of tomorrow's appointments. The dialler places calls at a limited rate through the Twilio REST API, passes the target agent and any per-contact context into the call, and records the outcome of each attempt in the call log so that a run can be reviewed afterwards.

VII. DATA DESIGN AND SECURITY

Persistent state is held in Supabase PostgreSQL. The main tables are users, assistants, calls, phone numbers, knowledge base question and answer pairs, and, in this iteration, bookings together with the per-agent availability settings. Every row that belongs to a business carries an owner identifier. Row level security policies make each authenticated user able to read and change only rows that carry their own identifier, so that one client's calendar can never appear in another client's console, even if a front end bug asked for it [6].

Secrets are kept in environment variables on the server and are never shipped to the browser. The service role key used for server side writes is used only inside the Express process. Voice and language provider keys are read at call time. Phone numbers of callers and call transcripts are personal data, so they are visible only to the owning business, and the booking record stores only what is needed to run the appointment: a name, a number and the time. Authentication follows the Supabase session model with token based access, and all browser traffic uses HTTPS.

VIII. TESTING AND FINDINGS

A. Test approach

Testing was manual and module by module. Each page of the console and each action in the pipeline was exercised with prepared scenarios, including the unhappy paths: an unavailable slot, a caller who changes the date halfway through, a caller who stays silent, a caller who interrupts, and a request for something outside the agent's knowledge. Two team members tested in parallel on separate agents to try to produce a double booking deliberately. Telephone tests were made with real calls from mobile phones on ordinary broadband and mobile data connections. The areas covered are listed in Table 2.

Table 2: Functional areas tested

Area	Scenario examined	Result
Authentication	Sign up, sign in, session persistence, protected pages	Pass
Agent Studio	Create, edit and delete agents; voice and prompt changes	Pass
Knowledge base	Adding question and answer pairs, answering from them	Pass
Inbound calls	Greeting, free conversation, interruption, silence	Pass
Outbound and bulk	Single test call, list upload, outcome recording	Pass
Availability	Free slots, fully booked days, outside working hours	Pass
Booking	Normal booking, conflict, change of mind, missing name	Pass
Workspace calendar	Live appearance of bookings, day and week views	Pass
Call logs	Transcript, duration, linked booking	Pass
Data isolation	Access across two business accounts	Pass

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

B. Observed behaviour

Table 3 summarises what we observed. We deliberately report ranges and not single figures, because call quality depends on the caller's network, on the providers' load at that moment and on the length of the reply. The values are in line with those reported for comparable cascaded agents [1].

Table 3: Indicative behaviour observed during testing

Measure	Observation	Assessment
Gap before the agent starts speaking	About one to two seconds	Acceptable
Extra delay when a tool is called	Usually under a second	Acceptable
Booking visible on calendar	Within a few seconds	Good
Overlapping booking attempts	Rejected every time by the database	Good
Recovery after a provider fault	Fallback voice or model used	Good
Interruption handling	Playback stops, next turn starts cleanly	Good

C. Comparison with other approaches

Table 4 compares VAANIX with the two things clients usually have today. The comparison is qualitative and concerns capabilities, since pricing of managed services changes frequently.

Table 4: Comparison with existing approaches

Capability	Touch-tone IVR	Managed voice agent services	VAANIX
Understands free speech	No	Yes	Yes
Model and prompt control	Not applicable	Limited	Full
Real backend actions	Fixed menus	Varies	Per-agent actions
Appointment calendar for the business	No	Usually external	Built-in Workspace
Outbound and bulk calls	Rare	Often extra	Built-in
Cost model	Fixed	Per minute fee	Provider usage costs only
Self-hostable	Some	No	Yes

D. Feedback from testers

The people who tried the system were mostly classmates and a few acquaintances who run small practices. Their comments were consistent. The booking conversation felt natural once the agent stopped asking for information the caller had already given. The calendar was the part that non-technical testers liked most, because it showed them something they could recognise. The most common requests were to let a caller reschedule or cancel by phone, and to send a text message confirmation after a booking.

IX. CHALLENGES AND LIMITATIONS

A. Models acting too early

The first booking prompts produced agents that tried to book before they had a date. The cure was not a cleverer model but better tool descriptions and a server side check that rejects incomplete arguments with a message the model can act on, such as asking the caller for the missing date. Returning a readable error to the model, and not throwing, let it recover inside the same turn.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

B. Spoken dates and times

Callers say "day after tomorrow", "next Monday", "around five". The model resolves these correctly only if it knows today's date and the business's time zone, so both are inserted into the prompt at the start of each call. We also found it safer to make the agent repeat the full date and time back to the caller before booking.

C. Latency with tool calls

A tool call adds a round trip to the database before the model can answer. Keeping the availability query narrow, limited to the requested window and an indexed time range, kept the added delay small. While a tool is running, the agent can say a short filler phrase so that the line is not silent, which callers found far more tolerable than a pause.

D. Telephony in India

Connecting a pipeline like this to a local Indian number is bound by telecom rules on local voice numbers, which are harder to obtain through global providers. For demonstrations we used a number from outside India. A deployment for Indian clients would need an India based telephony provider for the carrier leg, with the rest of the system unchanged.

E. Scope

The current actions are availability lookup and booking. The agent cannot yet change or cancel an existing appointment, it does not send confirmation messages, and the pipeline has been tested mainly in English. Testing was manual, with a small number of users, so the findings in Section VIII should be read as indicative and not as a statistical evaluation.

X. CONCLUSION

VAANIX started as a talking pipeline and has become a platform that gets things done on a phone call. The Agent Studio puts every part of an agent in one place, the action mechanism gives each agent only the capabilities it needs, and the Business Workspace shows the client the outcome in the form they care about, a calendar of real appointments. Two design decisions turned out to matter most. The first was to enforce the no-double-booking rule in the database and not in the conversation. The second was to attach identity and ownership on the server and never leave them to the language model. Both are cheap to implement and make the system far easier to trust. The work also showed that most of the effort in a voice agent goes into the edges of the conversation, such as interruptions, spoken dates and silent moments, and not into the model itself. We believe a small, self-hosted platform of this kind is a realistic option for clinics, salons, coaching classes and other appointment driven businesses that cannot afford a full call centre but do not want to lose calls.

XI. FUTURE SCOPE

A. Near term

Rescheduling and cancelling appointments by phone. Text message and WhatsApp confirmations and reminders sent automatically after a booking. Lead capture and message taking as additional actions. Calendar synchronisation with external calendars through the iCalendar format [13].

B. Medium term

Support for Hindi and Marathi calls, since many of the clients we spoke to serve local customers. Additional Workspace views for other business types, such as reservations and order status. Analytics on missed calls and booking conversion, presented in business terms. Migration of the telephony leg to an India based provider.

C. Long term

Sentiment aware hand-off to a human when a caller is upset or the request is outside the agent's scope. Customer relationship tool integration. Fine-tuning for domain vocabulary such as medical specialties. A visual builder for call flows and a public API for institutions that want to embed the agent in their own systems.

XII. CONFLICT OF INTEREST

The authors declare that there is no conflict of interest regarding the publication of this research paper.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

XIII. ACKNOWLEDGEMENTS

The authors thank the Department of Computer Engineering at P K Technical Campus, Chakan, Pune, and Savitribai Phule Pune University (SPPU) for the academic environment in which this work was carried out. We are especially grateful to our guide, Prof. Saba Chaughule, for her steady guidance and for the time she gave to reviewing our design decisions at every stage of the project. Thanks are also due to the classmates and acquaintances who tested the platform and told us plainly what did not work. Finally, we acknowledge the open source communities behind React, Supabase, Tailwind CSS and the other tools on which the platform was built.

REFERENCES

[1] J. Qiu, Z. Chen, L. Yang, M. Zhu, Z. Liu, J. Tan, W. Zhao, R. Murthy, R. Ram, A. Prabhakar, S. Heinecke, C. Xiong, S. Savarese, and H. Wang, "Building Enterprise Realtime Voice Agents from Scratch: A Technical Tutorial," arXiv:2603.05413, Mar. 2026.
[2] Twilio, "Media Streams," Twilio Voice Documentation. [Online]. Available: https://www.twilio.com/docs/voice/media-streams
[3] Deepgram, "Streaming Speech-to-Text," Deepgram Developer Documentation. [Online]. Available: https://developers.deepgram.com/docs
[4] ElevenLabs, "Text to Speech," ElevenLabs Documentation. [Online]. Available: https://elevenlabs.io/docs
[5] Anthropic, "Tool use with Claude," Claude API Documentation. [Online]. Available: https://docs.claude.com
[6] Supabase, "Row Level Security," Supabase Documentation. [Online]. Available: https://supabase.com/docs/guides/database/postgres/row-level-security
[7] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in Proc. International Conference on Learning Representations (ICLR), 2023.
[8] T. Schick, J. Dwivedi-Yu, R. Dessi, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, "Toolformer: Language Models Can Teach Themselves to Use Tools," in Advances in Neural Information Processing Systems (NeurIPS), 2023.
[9] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Advances in Neural Information Processing Systems (NeurIPS), 2020.
[10] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust Speech Recognition via Large-Scale Weak Supervision," in Proc. International Conference on Machine Learning (ICML), 2023.
[11] T. Stivers et al., "Universals and cultural variation in turn-taking in conversation," Proceedings of the National Academy of Sciences, vol. 106, no. 26, pp. 10587-10592, 2009.
[12] ITU-T, "Recommendation G.114: One-way transmission time," International Telecommunication Union, 2003.
[13] B. Desruisseaux, Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)," RFC 5545, Internet Engineering Task Force, Sep. 2009.

AUTHORS INFORMATION

Project Guide

Prof. Saba Chaughule is a Master of Engineering (ME) graduate in Computer Engineering and is currently the Acting Head of the Department of Computer Engineering at P K Technical Campus, Chakan, Pune, affiliated with Savitribai Phule Pune University (SPPU). She has strong expertise in full stack development, and her teaching and mentoring cover web application design, databases and software engineering practice. As the guide of the VAANIX project she helped the team at every stage, from choosing the problem and shaping the architecture to reviewing the module designs, the testing approach and the documentation. Her regular feedback and her insistence on clear reasoning behind every design decision improved the quality of the work considerably.

Students

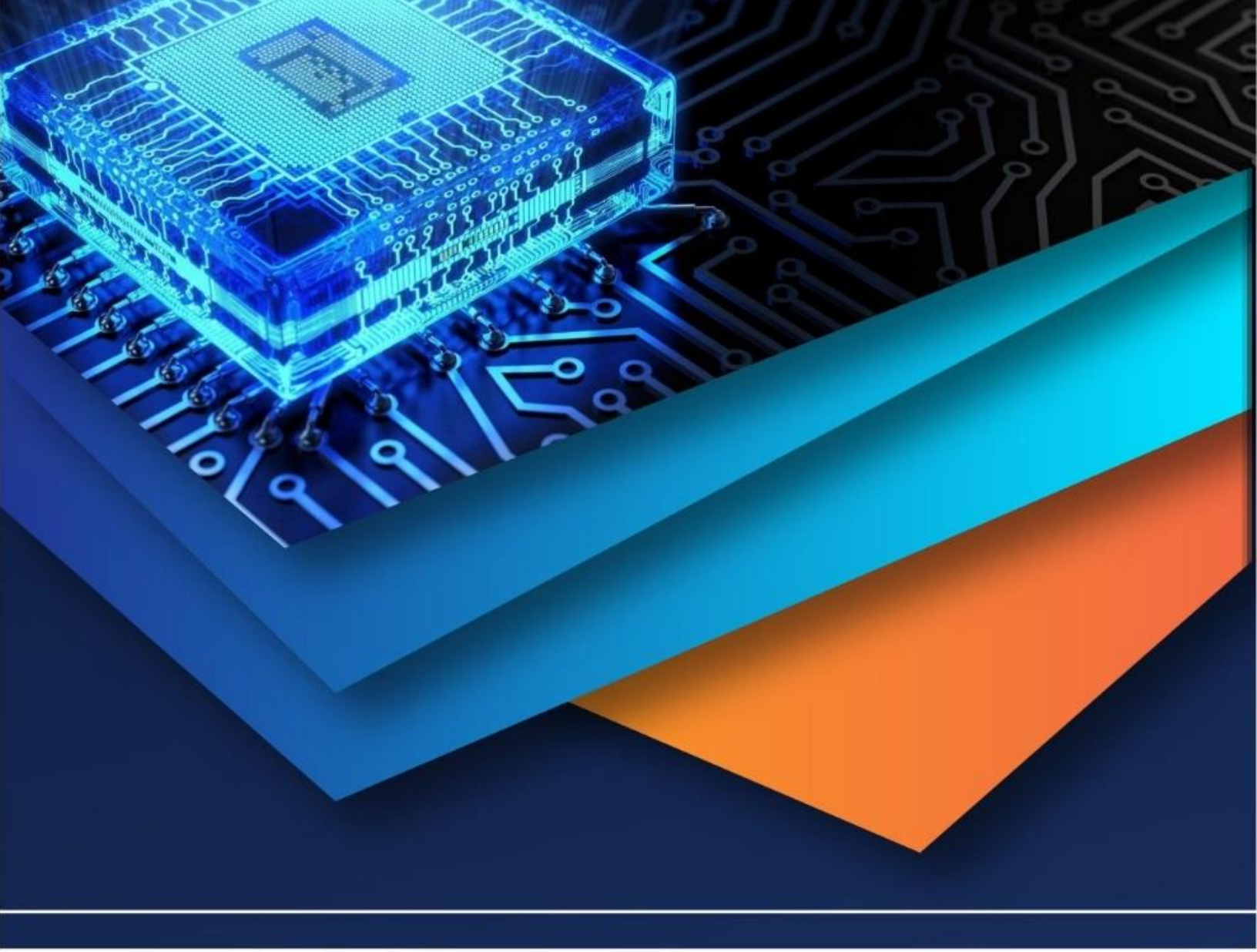
1. Mr. Atharv Bhosale is a final year student of Bachelor of Engineering in Computer Engineering at P K Technical Campus, Chakan, Pune, affiliated with SPPU. He is skilled in full stack development and in the design and integration of AI based systems, and has worked with real-time communication, speech and language model services in several academic and personal projects. He has also completed a two month internship as an Associate AI Developer, which gave him hands-on experience of building AI features for production software. In VAANIX he led the system architecture, the real-time voice pipeline, the action mechanism and the front end implementation, including the Agent Studio and the Business Workspace. He intends to continue working on AI driven products and real-time systems.
2. Ms. Srushti Phalke is a final year student of Bachelor of Engineering in Computer Engineering at P K Technical Campus, Chakan, Pune, affiliated with SPPU. She completed a diploma in engineering before joining the degree programme. She is proficient in SQL and database design and has experience in Android application development. In VAANIX she was responsible for the database design, including the schema, relationships and constraints behind assistants, calls and bookings, and for the data access and integrity rules that keep each client's information separate.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

3. Mr. Avdhooth Chaudhari is a final year student of Bachelor of Engineering in Computer Engineering at P K Technical Campus, Chakan, Pune, affiliated with SPPU. He holds a diploma in engineering and has professional experience as a Customer Relations Executive at Just Dial, where he dealt daily with customers and business owners. This experience gave him a strong ability to communicate with clients, draw out their real requirements and carry out ground research. In VAANIX he handled requirement gathering from prospective clients, such as clinics, the study of how appointments are handled in practice, and the preparation of realistic call scenarios. He is also strong in software testing and contributed to test design and verification.

4. Ms. Srushti Magar is a final year student of Bachelor of Engineering in Computer Engineering at P K Technical Campus, Chakan, Pune, affiliated with SPPU. She is particularly strong in software testing and carried out the manual testing of every module of the platform, from authentication and the Agent Studio to calling, booking and the calendar. She also contributed to background research and to the front end work. Her systematic test scenarios, covering both normal and unhappy paths, uncovered several problems before they could reach users.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)