WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

Clinora AI: A Voice-Enabled System for Reliable Hospital Appointment Management

Aditya Yadav[1] , Sumit Benke[2] , Kartik Jadhav[3] , Riya Godambe[4] , Prof. Prajkta Puranik [5]

Department of Artificial Intelligence and Machine Learning, ISBM College of Engineering, Pune Industry-sponsored live project in collaboration with City Hospital, Marunji, Pune, India

DOI : https://doi.org/10.55248/gengpi.07.1026.34039

ABSTRACT

Patients often need to complete several steps to arrange a hospital visit: identify a suitable doctor, check an available time, book the slot, and later retrieve or cancel the booking. A voice-based interface may make these tasks easier to start, but convenience alone is not enough. The system must also avoid incorrect bookings and explain when a request cannot be completed. This paper describes Clinora AI, a prototype design that connects a VAPI voice interface and a large language model (LLM) using predefined tool calls to a FastAPI backend and an SQLite database. The backend is responsible for validating requests and determining the final appointment state. The proposed workflow includes availability checks, booking, retrieval, cancellation, and rescheduling; each feature should be described as implemented only after it has been verified in the running application. This paper also sets out a test plan for intent handling, tool selection, transaction correctness, ambiguous requests, conflicts, and response time. Since measured results were not supplied, no performance figures are claimed. Clinora AI is intended for appointment administration, not diagnosis or treatment advice. Index Terms—Conversational AI, voice agent, appointment scheduling, large language model, tool calling, FastAPI, SQLite, healthcare administration.

I. INTRODUCTION

Arranging an appointment can involve more than choosing a date. Patients may need to identify a doctor by name or specialty, compare available times, or find details of an earlier booking. A request may also be incomplete: a patient might say they want an appointment next week without specifying a day or preferred time. In these cases, the system should ask a focused follow-up question rather than guess. A voice interface gives patients another way to interact with appointment services, but a natural-sounding response does not prove that the requested action took place. The model may misunderstand a date, choose an unsuitable operation, or respond before a database update has succeeded. In addition, a time slot that was free during an availability search may be taken before the patient confirms the booking. Clinora AI is designed around this distinction: the language model interprets the request, while the backend checks the data and controls appointment changes. A voice interface can offer an alternative to web forms or waiting for staff to answer a call. However, a fluent conversation does not guarantee a correct booking. A language model may misunderstand a date, select the wrong operation, or announce success before the database confirms the change. Availability can also change between a search and a booking attempt. Clinora AI addresses these risks by using the model to interpret the request while leaving validation and appointment-state changes to the backend.

A. PROBLEM STATEMENT

The project addresses how to make appointment tasks accessible through voice while ensuring that unclear requests, stale availability, unauthorized access, or overlapping booking attempts do not lead to incorrect appointment records.

B. OBJECTIVE AND CONTRUBUTION

Offer a voice-based way to request doctor or specialty information and check appointment availability, where supported by the implementation. Provide booking, retrieval, cancellation, and rescheduling operations only for features that are present and tested in the working application. Ask the patient for missing or ambiguous information instead of selecting a doctor, time, or record without sufficient evidence. Validate requests and apply scheduling rules in the backend before changing appointment data. Include notification and administrative dashboard functions only if these modules have actually been built and tested. Define repeatable tests that assess request handling, booking correctness, conflict responses, and end-to-end latency. The paper's central engineering idea is to keep two responsibilities separate. The language model interprets what the patient says and proposes a structured operation; the backend independently validates that operation and decides whether the appointment record can change. This separation is a design choice, not a claim that voice interfaces or LLM tool calling are new in themselves.

II. RELATED WORK

Laranjo et al. surveyed conversational agents used in healthcare and found differences in the dialogue approaches and evaluation methods reported across studies [1]. For an appointment system, a useful evaluation should go beyond whether the conversation sounds natural. It should also check whether the correct record is retrieved, access rules are followed, and the requested change is reflected accurately in the system. HL7 FHIR defines an Appointment resource for representing a planned healthcare meeting and its participants [2]. For a scheduling application, a successful availability search should be treated as a temporary view of the schedule rather than a reservation. The backend needs to recheck the slot when a booking request arrives. Mentioning FHIR does not mean Clinora AI conforms to the standard; that claim would require implementation and conformance testing. A tool call produced by an LLM should be treated as an input request, not as an instruction that can bypass application rules. The backend needs to validate the fields, check the caller's permissions, and enforce the constraints for the requested operation. The related-work section should be expanded with verified publications on tool calling, database transaction handling, healthcare privacy, and appointment scheduling before journal submission.

III. SYSTEM ARCHITECTURE

The proposed design has six main layers: the patient's voice interface, VAPI conversation handling, LLM-based interpretation of the request, a predefined tool interface, FastAPI business logic, and SQLite persistence. Notifications and an administrative dashboard can be separate components if they are part of the deployed system. The final architecture figure must show the actual system rather than an aspirational design. [Replace this note with an architecture diagram based on the working code. Use a different label or line style for any module that is planned but not implemented.]

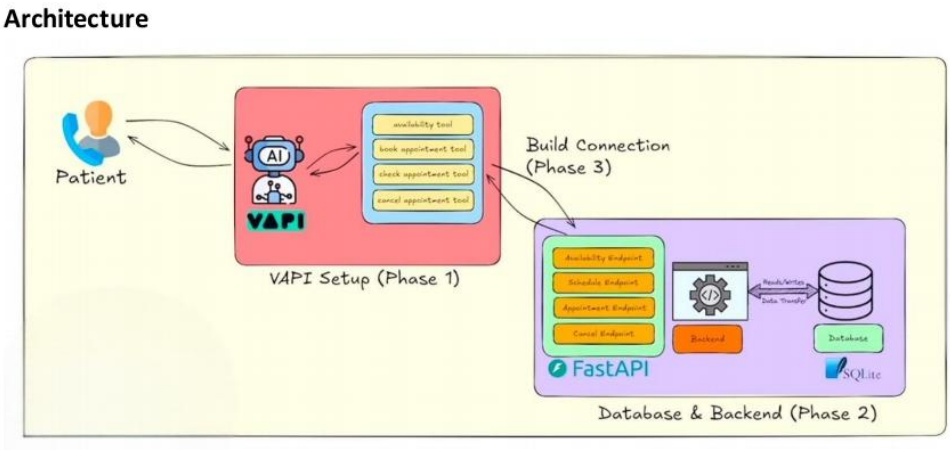


Fig. 1. Reference architecture illustrating the voice agent, tool layer, API backend, and database. The final figure should be redrawn or adapted to match the implemented Clinora AI modules.

A. Voice and Language Layer VAPI manages the voice conversation. The LLM interprets the patient's request and extracts relevant fields, such as the requested doctor or specialty, date, time, and appointment reference. If the patient describes a specialty rather than naming a doctor, the system can search a doctor directory if that capability exists. When more than one result fits, it should ask the patient to choose instead of making an unsupported selection. Details from an earlier appointment must not be disclosed until identity and access checks have passed.

B. Backend and Database FastAPI receives the structured operation and should validate required fields, date and time formats, permissions, current appointment status, and slot availability. The database may contain fields such as appointment ID, patient ID, doctor ID, start and end times, status, and creation time, but this description must be checked against the actual schema. Database constraints and transaction handling are important when two requests attempt to reserve overlapping times. Do not claim that these safeguards are implemented unless the code and tests demonstrate them.

C. Notifications and Administration If email or SMS notifications are part of the implementation, they should be triggered after the backend confirms the appointment change. Notification delivery and booking status should be tracked separately: a message can fail even though the booking was saved successfully. An administrative dashboard could present schedules, booking totals, cancellations, and workload summaries. Access to such information should be restricted by role and organizational scope

IV. WORKFLOW AND METHODOLOGY

A. Availability and Doctor Discovery The system should identify the requested doctor, specialty, or facility and query the schedule currently stored by the backend. If the patient asks about a previous visit, the system must verify identity and authorization before returning appointment information. Availability results can become outdated, so the booking operation needs its own final availability check.

B. Booking The agent collects the information needed for a booking and, where appropriate, repeats the doctor, date, time, and facility so the patient can confirm the request. The backend then validates the input and checks the slot again before saving the appointment. If another request has already

taken the slot, the backend should return a conflict rather than create an overlapping booking. The voice agent should say that the booking succeeded only after it receives a successful result from the backend.

C. Retrieval, Cancellation, and Rescheduling Before retrieving an appointment, the backend should confirm that the caller is allowed to access it. A cancellation should be accepted only when the record exists and its current status permits cancellation. Rescheduling requires additional care: the system should validate the replacement slot and apply the change in a way that does not accidentally leave the patient without a valid appointment. The meanings of statuses such as confirmed, cancelled, and rescheduled should be defined consistently. These are required behaviours; report them as implemented only after checking and testing the corresponding code.

D. Doctor Unavailability If a doctor becomes unavailable after an appointment has been booked, the system should flag the case for review and present alternatives allowed by hospital policy. It should not silently move the appointment to a different time. Any replacement should be confirmed, recorded, and communicated to the patient and relevant staff. Unless the running system already supports this workflow, describe it as a recommendation rather than an existing feature.

E. Failure and Retry Handling The application should distinguish network timeouts, database errors, and notification failures because they have different effects. A timeout does not always mean that a booking failed; the database may have committed the change before the response was lost. Before retrying, the backend should check the original operation or use an idempotency mechanism to avoid creating a second booking. Logs should help developers diagnose failures without retaining unnecessary personal or medical information. .

V. IMPLEMENTATION DESIGN

Component	Technology / module	Responsibility
Voice interaction	VAPI	Manage voice conversation and handoff
Language understanding	LLM with tool calling	Identify intent and extract structured fields
API backend	Python / FastAPI	Validate requests and apply scheduling rules
Persistent storage	SQLite	Store appointment and related records
Notifications	Email/SMS provider [specify]	Notify patients and doctors after state changes
Admin dashboard	Web interface [specify]	Display authorized operational summaries

A. Illustrative API Operations

Operation	Illustrative endpoint	Expected behavior
Availability	GET /availability	Return current slots for a doctor and date
Booking	POST /appointments	Validate and create a booking
Retrieval	GET /appointments/{id}	Return an authorized appointment record
Cancellation	POST /appointments/{id}/cancel	Cancel if the current state permits it
Rescheduling	PATCH /appointments/{id}	Validate and apply a safe schedule change
Doctor directory	GET /doctors	Search authorized doctor and specialty information

These routes are examples, not claims about the current codebase. Replace them with the actual implemented endpoints, request schemas, response codes, and database constraints before submission.

VI. SECURITY, PRIVACY, AND SAFETY

Appointment records can reveal sensitive personal information. A production deployment should authenticate users, check authorization on every read and write, restrict administrator access by role and hospital, protect API keys and provider credentials, and use encrypted connections. The project should also define how long records and logs are retained and how they can be deleted. Voice responses should disclose only the information needed for the task, especially when other people may be nearby. Use synthetic data for prototype testing unless appropriate approvals and safeguards are in place. Clinora AI is intended to support administration and should not provide diagnosis or treatment recommendations.

VII. EVALUATION PLAN

A meaningful evaluation should use a fixed set of test requests covering normal use as well as incomplete requests, ambiguous names or dates, invalid input, duplicate attempts, simultaneousbookings,

unauthorized access, and simulated service failures. Record the number of tests, software and model versions, and relevant hardware and network conditions so another team can understand how the

results were obtained. Intent accuracy = correctly classified intents / total intent tests × 100. Tool-selection accuracy = correct tool choices / total tool-selection tests × 100.

Valid transaction success rate = valid requests completed correctly / total valid requests × 100. Conflict-prevention rate = conflicting booking attempts correctly rejected / total conflict tests × 100.

Clarification rate = ambiguous requests that receive an appropriate clarification / total ambiguous tests × 100. Notification delivery rate = notifications confirmed delivered / notification attempts × 100.

Latency is the time from the start of a user request to the final response; report median, mean, and 95th percentile. A. Results Reporting The current draft does not provide measured test results.

Fill the results table only after running the tests and recording the observations. Do not substitute expected outcomes, estimates, or invented values for experimental results.

Test ID	Scenario	Expected outcome
T1	Check availability for a known doctor	Current available slots returned
T2	Book a free slot	Exactly one valid appointment created
T3	Book an occupied slot	Conflict returned; no duplicate booking
T4	Two simultaneous requests for the same slot	At most one incompatible booking succeeds
T5	Patient omits date or time	Agent asks a focused clarification
T6	Patient asks about a previous visit	Identity and authorization checked before disclosure
T7	Retrieve unknown appointment	Clear not-found response
T8	Cancel an active appointment	Status changes and result is confirmed
T9	Reschedule to an occupied slot	Original appointment remains valid; conflict reported
T10	Doctor becomes unavailable	Affected booking flagged and policy-driven alternatives offered
T11	Notification provider fails	Booking state remains accurate; notification failure tracked
T12	Backend times out after booking request	Original outcome checked before retry; duplicate avoided
T13	Unauthorized patient requests another record	Access denied
T14	Database/backend error	No false success confirmation

A. Results Reporting Template

No measured results were supplied for this draft. Complete the table below only after running the tests; do not replace the placeholders with estimates or expected values.

Metric	Measured result
Intent accuracy	[Enter measured value and test count]
Tool-selection accuracy	[Enter measured value and test count]
Valid transaction success rate	[Enter measured value and test count]
Conflicting bookings accepted	[Enter observed count / total conflict tests]
Clarification rate	[Enter measured value and test count]
Notification delivery rate	[Enter measured value and test count]
Median / mean / P95 latency	[Enter measured values and units]

VIII. DISCUSSION

Separating language interpretation from transaction handling gives the application a clear place to verify whether an appointment change actually occurred. The evaluation should therefore include failure cases, not just demonstrations where everything works: two users requesting the same slot, incomplete patient details, outdated availability, failed rescheduling, and notification outages. Conversation quality and transaction correctness are different properties and should be measured separately. Whether Clinora AI improves on a manual or web-based process remains an open question until it is compared with a defined baseline.

IX. LIMITATIONS

The system's behaviour may be affected by speech-recognition errors, accents, background noise, network delays, model settings, and errors in the schedule data. SQLite may be adequate for a small prototype, but its suitability for a larger deployment depends on workload and concurrency requirements. Any dashboard or notification function also depends on the quality of the underlying records and the availability of external services. These limitations should be reassessed using results from the implemented system.

X. FUTURE SCOPES

The following ideas are potential extensions. Keep them in this section unless the features have already been implemented and evaluated:

1. Multilingual voice interaction: add support for additional languages and regional accents, then test the system with speakers and recording conditions that reflect expected use. 2. Doctor discovery: allow patients to search by specialty, availability, and preferred appointment time, with clear handling when several doctors match.

3. Rescheduling and reminders: add a safe rescheduling workflow and automated reminders through communication channels approved by the organization.

4. Conflict handling: test and strengthen database transactions and concurrency controls for requests that target the same slot.

5. Security and privacy: improve authentication, role-based permissions, encryption, audit logging, and controls for protecting patient information.

6. Hospital integration: investigate integration with hospital management systems or electronic health record platforms, using standards such as HL7 FHIR where appropriate.

7. Performance and usability studies: measure task completion, tool-selection accuracy, booking conflicts, clarification behaviour, response latency, and user experience. These extensions should be introduced and assessed incrementally. Any statement that an enhancement improves reliability, accessibility, or performance should be supported by measured evidence.

XI. CONCLUSION

Clinora AI explores voice-based support for hospital appointment administration by connecting VAPI, an LLM that uses predefined tool calls, a FastAPI backend, and SQLite storage. The planned workflow covers availability checks, booking, retrieval, cancellation, and rescheduling; notification and dashboard functions should be included as current capabilities only when they are implemented. The design assigns validation, authorization, scheduling rules, and final transaction status to the backend rather than relying on the model's conversational response. Clear clarification questions, conflict handling, safe retries, and truthful booking confirmations are important requirements for this type of system. Since the supplied draft contains no measured evaluation results, it is not yet possible to draw conclusions about the system's reliability or effectiveness. The next step is to test the implemented functions, report reproducible results, and compare them with a clearly defined baseline.

REFERENCES

[1] A. Laranjo et al., “Conversational agents in healthcare: a systematic review,” Journal of the American Medical Informatics Association, [1 2 3 4 5 6]vol. 25, no. 9, pp. 1248–1258, 2018, doi: 10.1093/jamia/ocy072.

[2] HL7 International, “Appointment—FHIR Release 5,” HL7 FHIR Specification. Available: https://hl7.org/fhir/R5/appointment.html. Verify the version and access date before submission

. [3] M. Milne-Ives et al., “The effectiveness of artificial intelligence conversational agents in health care: systematic review,” Journal of Medical Internet Research, vol. 22, no. 10, article e20346, 2020. Verify complete metadata and DOI.

[4] H. Ding et al., “Evaluation framework for conversational agents with artificial intelligence in health interventions: a systematic scoping review,” Journal of the American Medical Informatics Association, [7]vol. 31, no. 3, pp. 746–761, 2024. Verify complete metadata and DOI.

[5] H.-D. Xu et al., “AgentTOD: A Task-Oriented Dialogue Agent with a Flexible and Adaptive API Calling Paradigm,” ACM Transactions on Information Systems, vol. 43, no. 5, article 136, 2025. Verify publication details and relevance.

[6] T. Labruna, G. Bonetta, and B. Magnini, "Task-Oriented Dialogue Systems through Function Calling," in Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing, pp. 614–622, 2025. Verify the proceedings record and complete citation.

[7] J. M. Ruiz Mejia and D. B. Rawat, "MedScrubCrew: A Medical Multi-Agent Framework for Automating Appointment Scheduling Based on Patient-Provider Profile Resource Matching," Healthcare, vol. 13, no. 14, article 1649, 2025. Verify DOI and bibliographic details.

[8] P. Lagakis et al., "ClinApp: A Microservices-Based Platform for Efficient Medical Visit Scheduling," Studies in Health Technology and Informatics, vol. 305, pp. 339–340, 2023. Verify complete citation. Editorial note: These references were carried forward from the supplied draft and have not all been independently verified in this rewrite. Confirm each source, DOI, page range, and claim it supports. Remove any reference that cannot be verified, and add checked sources on LLM tool calling, transaction safety, healthcare privacy, and appointment scheduling.