



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.2026.85291>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

DESKXI: Design and Development of a Voice-First Desktop AI Mentor for Interactive Learning, Personalized Study Assistance, and Intelligent Task Management

Urmil Mamulkar¹, Deepak Kumar², Sumedh Dhoke³, Vijay Sharma⁴, Sejal Farkade⁵, Shrushti Bhosale⁶, Dr. Ruchi Kadwane⁷

Department of Computer Science and Engineering, G H Rasoni University, Amravati, India

Abstract: *Students who study alone rarely have a tutor, an interview coach and a study planner within reach, and screen-based assistants pull attention away from the work itself. DESKXI is a voice-first desktop AI mentor that combines these roles in one device: it teaches by conversation, runs mock interviews, sets spoken quizzes, keeps Pomodoro study cycles, delivers reminders and retains durable facts about the learner between sessions, all through a streaming cloud speech-to-speech model. This paper describes how DESKXI divides work between the cloud model, which carries the open-ended conversation, and deterministic local services, which hold exact state such as timers, reminders and quiz keys. It also describes the real-time client runtime that carries these functions: a bounded drop-oldest capture queue, a writer-thread playback path, a microphone mute gate, a three-tier recovery design and a SQLite store. We examine selected parts of this runtime by running the unmodified source code on a reference workstation with only the hardware and cloud modules stubbed. The measurements identified four limitations that matter for mentoring: speech that keeps playing for about 1.1 s after the learner interrupts, an unfiltered down-sampler that folds out-of-band energy into the speech band, reminders that fire up to 0.95 s early, and a quiz-answer parser that reached 44.6 % accuracy on our author-written stress set. We give a remedy for each. Learning outcomes were not studied.*

Keywords: AI mentor, voice interaction, desktop robot, speech-to-speech model, study assistant, real-time systems.

I. INTRODUCTION

Learners who work alone need three kinds of help that are normally supplied by different people or applications: an explainer who adapts to what they know, a practice partner who asks questions and gives feedback, and an organiser who keeps time and remembers commitments. Chat applications can supply all three, but only through a screen, which is the thing a student trying to concentrate would like to look at less. Robot tutors have shown learning benefits in classrooms and in one-to-one settings [20], [21], and spoken interaction removes the keyboard from the loop. What has changed recently is the cost of building one: speech-to-speech services now fold recognition, reasoning and synthesis into a single streaming session [5]–[7], built on multimodal foundation models [9]. DESKXI is our attempt at such a mentor. It is a desktop device that listens continuously, with no wake word, and offers a teaching mode, a mock-interview mode, spoken multiple-choice quizzes, Pomodoro study cycles with progress summaries, timed reminders, short web and weather look-ups, and a store of durable facts about the learner. The cloud model supplies the conversation; everything that must be exact, such as a timer or a quiz key, is kept in local deterministic services that the model reaches through function calls. Placing a mentor on a desk makes timing and reliability part of its teaching quality. Gaps between turns in everyday speech cluster around 200 ms [2], and one-way delays beyond 150 ms are treated as a degradation in telephony [3]. A learner who interrupts to say that an explanation went too fast expects the device to stop; a learner who sets a 25-minute focus timer expects it to survive a network drop. The model connection is itself a remote service with documented lifetime limits [8], and the device must feed it a continuous stream, play back replies through a wireless sink and restart cleanly if the session or the program ends. Studies of LLM-powered robots have examined interaction quality and user expectations [4]; this paper complements that work by describing the client-side runtime of a voice-based study assistant.

This paper therefore does two things. It describes how DESKXI allocates mentoring functions between the model and local services, and it measures selected parts of the real-time client that carries them.

The runtime mechanisms were added one at a time to cure symptoms seen during development, such as input overflows and silent crashes; this paper quantifies a subset of them. Our contributions are:

- C1 – A description of the DESKXI mentor and of the division of labour between a cloud speech model and deterministic local services (Section III).
- C2 – Closed-form relations for capture-queue staleness and loss, residual playback after an interruption, and reminder timing error; the last two are compared with measurement (Sections IV and VI).
- C3 – A white-box measurement study that runs the shipped Python and shell code with only the hardware and cloud modules stubbed, reporting four measured findings, all of them limitations that matter for mentoring (Sections V and VI).
- C4 – A test method that needs no robot hardware, so that the same measurements can be repeated on other voice-based study assistants.

We make no claim about learning outcomes, conversational quality or user preference. Those need a controlled human study, which is outside this paper.

II. RELATED WORK

- 1) *Robot tutors and companions.* A review of social robots in education [20] reports positive but task-dependent effects on learning and stresses that physical presence and adaptive behaviour matter; an early field study had a robot act as a peer tutor for children [21]. These systems relied on scripted dialogue, so the engineering of an open-ended spoken mentor was not the issue.
- 2) *Turn-taking and delay.* Skantze's review [1] treats interruption handling, end-of-turn prediction and overlapping speech as core problems for spoken robots, and field studies of turn gaps [2] and delay recommendations [3] give the timing targets used here.
- 3) *Speech-to-speech systems.* Full-duplex models such as Moshi [5] and listen-while-speaking language models [6] move turn management inside the model; commercial streaming APIs [7] expose automatic voice-activity detection, an interruption flag, function calling and 16-bit PCM at 16 kHz in and 24 kHz out. They leave the client's buffering, echo handling and recovery to the integrator, which is where DESKXI's code sits.
- 4) *Dependability.* Crash-only design [10], recovery-oriented computing [11] and supervision trees [12] argue that restart should be the normal recovery path and that state should be durable. DESKXI follows this approach with a short shell supervisor and an embedded database.
- 5) *Queues and tails.* Excess buffering inflates latency without improving throughput [13], and tail percentiles rather than means govern perceived responsiveness [14]; both motivate the bounded capture queue described in Section IV-A.
- 6) *Signal handling.* Sample-rate reduction without a low-pass filter folds out-of-band energy into the passband [15], [16]; echo from a loudspeaker into a microphone is classically handled by cancellation [17], while DESKXI uses gating. We found no earlier report that examines these concerns together for a voice-based study assistant.

III. THE DESKXI MENTOR AND ITS RUNTIME

A. Mentoring Functions

The client runs on a single-board computer with a USB microphone and a Bluetooth speaker and opens a Live session at start-up. Table I lists what the learner can ask for and where each function is carried out. Teaching and interview behaviour is specified in the model's system prompt and runs entirely inside the live session; by design no second model is consulted for explanations, answer evaluation or follow-up questions, so that a turn costs one network round trip. Exact state lives outside the model: the model writes a four-option question and calls a local function that stores it and checks the learner's answer; Pomodoro sessions and reminders are rows in SQLite that a local worker polls every 250 ms and announces by injecting text into the live session.

TABLE I. MENTORING FUNCTIONS AND WHERE THEY RUN

Function	Reasoning	Local state	Sec.
Teaching mode	live model only	session context	–
Mock interview	live model only	turn log	–
Spoken quiz	model writes item	quiz key, parser	VI-D
Study coach (Pomodoro)	local state machine	study_sessions	VI-D
Reminders	local scheduler	reminders	VI-C
Durable facts, recent turns	model extracts facts at exit	facts, turns	VI-D
Weather and web look-up	network tools	none	–

B. Memory

Facts about the learner are extracted from the logged conversation at shutdown, stored by category, and injected in full, together with the last 20 turns, into the system prompt of every new or replacement session. We treat this only as a continuity mechanism and measure its cost; whether it improves mentoring is not evaluated here.

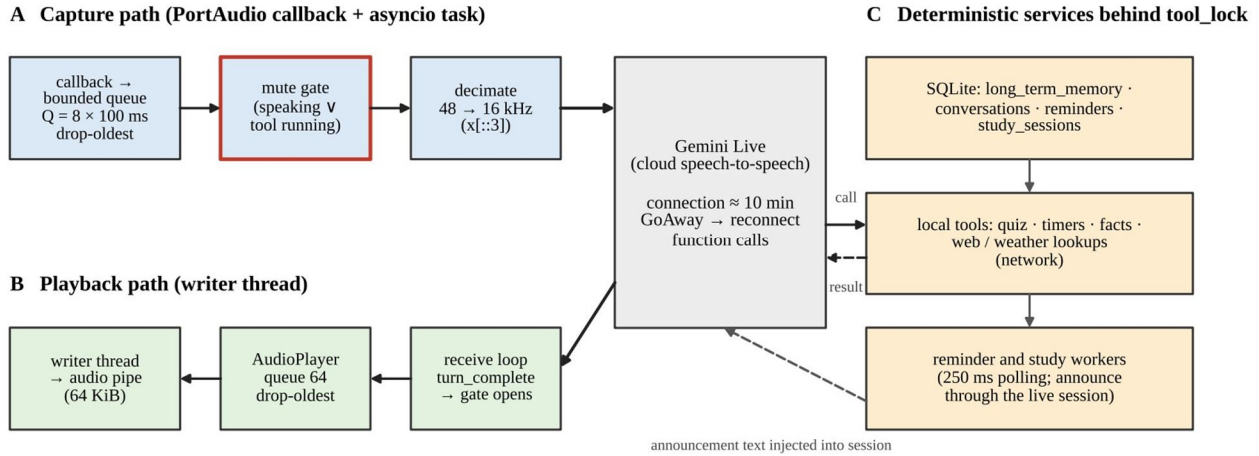


Fig. 1. DESKXI client runtime (software view). A: capture path with bounded queue and mute gate. B: playback path with writer thread. C: deterministic services reached through function calls; workers inject spoken announcements. Red outline marks the gate discussed in Section IV-B.

C. Data Paths

Fig. 1 shows the runtime. A sound-card callback copies each 100 ms block (4800 samples) into a queue of depth eight; an asyncio task removes blocks, drops two of every three samples to reach 16 kHz, and calls *send_realtime_input*. A gate discards blocks while a model reply is arriving or a tool is running. On the return path a receive loop hands audio to *AudioPlayer*, whose writer thread feeds a *pw-cat* pipe toward the Bluetooth sink; a 64-slot queue sits in front of the thread. Tool calls are serialised by a lock and answered from local functions.

D. Implementation Properties

Two properties come from reading the code. First, the active quiz item is process-local and is lost when the program restarts, whereas timers and reminders are rows in SQLite. Second, fact extraction runs at shutdown, so the facts of a session become available only after the program has exited cleanly, although the logged turns remain in the database.

IV. DESIGN ANALYSIS

A. Bounded-staleness capture

Let blocks arrive every $\tau = 100$ ms into a queue of depth $Q = 8$ that evicts its oldest entry when full. A block that waits in the queue is at most $Q\tau$ old when it is taken, so queued audio is bounded by

$$a_{\max} \approx Q\tau + t_{\text{send}} \approx 0.8 \text{ s} \quad (1)$$

plus one block already in flight when an upload stall begins, whose age equals the stall. If the send call blocks for a duration D , the queue then discards

$$L(D) \approx D/\tau - Q \text{ blocks} \quad (2)$$

per episode. The shipped design therefore trades completeness for freshness: the first words after a long stall survive, the audio from the middle of the stall does not. An unbounded queue would lose nothing but would deliver every block late. Equations (1) and (2) are analytical relations; they are not validated by experiment in this paper.

B. Playback, interruption and the mute gate

The writer thread keeps blocking pipe writes away from the event loop. The thread has a price: when the model streams faster than the sink drains, the 64-slot queue plus the pipe is the only buffer, and older chunks are dropped beyond it. When the user interrupts, the receiver calls *clear()*, which empties the 64-slot queue, but bytes already in the kernel pipe remain. With sink rate $\rho = 48\,000$ B/s (24 kHz, 16-bit, mono) the audio that still plays is

$$R = B_{\text{pipe}} / \rho \quad (3)$$

which is 1.37 s for the default 64 KiB pipe and 85 ms for a 4 KiB pipe. The mute gate adds a second constraint. It is closed while the model is generating or a tool runs, so the share of time the robot cannot hear is

$$DDC = (T_{\text{speak}} + T_{\text{tool}}) / T_{\text{session}} \quad (4)$$

which we call the deaf duty cycle. Because generation outruns playback, the gate opens when the model finishes, not when the speaker does; the unused *EchoGate* hangover class in the code base shows that the authors anticipated this window.

C. Recovery stack and deterministic services

Fig. 2 shows three tiers. T1 re-opens the SDK receive iterator after each turn, because the iterator ends at turn completion. T2 treats GoAway, close codes 1006/1011 and keep-alive failure as retirement of the model connection, which the provider documents as occurring after roughly 10 min for a connection and 15 min for audio-only sessions without compression [8]. On retirement it flushes the in-memory transcript to SQLite, rebuilds the system prompt from all stored facts plus the last 20 turns, drops queued audio, and reconnects. The provider also offers resumption handles valid for two hours [8]; DESKXI instead re-primes from its own store. T3 is the supervisor script, which relaunched the program after any exit code other than 0 or 130.

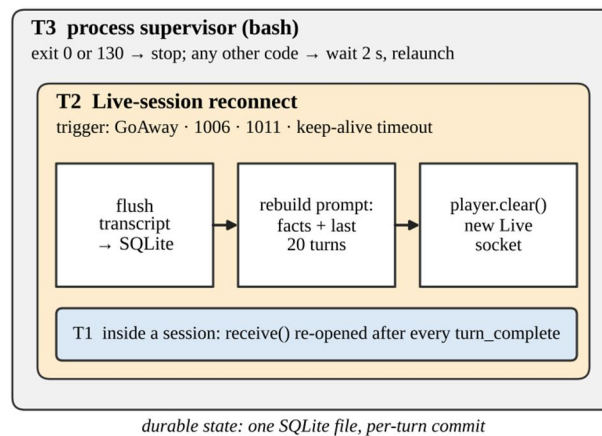


Fig. 2. Three-tier recovery design. Durable state lives in a single SQLite file.

Each logged turn is committed to SQLite individually; SQLite’s atomic-commit design [18] is the relevant durability mechanism, although this paper does not test it. Reminders and study timers are rows in the same file, whereas the active quiz item is process-local (Section III). The cost of the reminder design is at-least-once delivery [19]: the worker sends text to the model and only then marks the row delivered, so a restart between those two steps would announce the reminder a second time. Reminder times are stored truncated to whole seconds, which gives

$$e = -\text{frac}(t_0 + d) + w, \quad w \in [0, 0.25] \text{ s} \quad (5)$$

where t_0 is the creation time, d the requested delay, and w the polling offset of the 250 ms worker. The mean error is about -0.375 s: reminders are early on average.

V. EXPERIMENTAL METHOD

The harness imports DESKXI’s files unchanged and replaces only the hardware and cloud modules (PortAudio, Tk, google-genai, Tavily, requests) with stubs; database paths are redirected to a temporary directory. The experiments call the real *AudioPlayer* against a rate-limited stand-in for *pw-cat*, and the real *create_reminder* and *reminder_worker*. The 4 KiB pipe configuration is a counterfactual reconstructed in the harness and is labelled as such. All numbers come from one reference workstation (2 vCPU Intel Xeon 2.1 GHz, 8 GB, Linux 6.18, Python 3.13), not from the target single-board computer.

E1 (playback and interruption) produces a 12 s reply at 1×, 2× and 4× real time into a 48 kB/s sink, calls *clear()* after 3 s and records how much speech is still played. E2 (down-sampling) synthesises tones, white noise and speech-like harmonics and passes them through the shipped decimation step and through a polyphase resampler for comparison. E3 (reminder timing) creates 150 reminders with delays of 2–6 s and records when each is announced. E4 (local services) times the local tools and parses 56 hand-written quiz answers.

VI. RESULTS AND FINDINGS

A. Playback and interruption (E1)

Finding 1 – audio outlives the interruption. After *clear()*, 1.0–1.2 s of speech remained in the pipe at 2× and 4× generation speed, consistent with Eq. (3); a 4 KiB pipe reduced this to 20–65 ms (Fig. 3). Since a human turn gap is about 200 ms [2], the default leaves the robot talking over the user for roughly five gap lengths. Compounding this, the mute gate drops microphone blocks while the model speaks, so a spoken interruption may not reach the model at all. Chunk sizes and the real generation-to-playback ratio of the live service were not measured here.

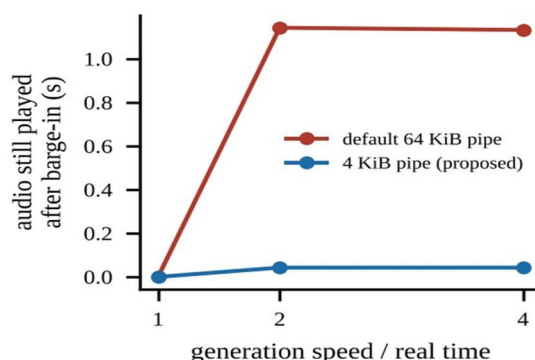


Fig. 3. Speech still audible after an interruption versus generation speed, default and 4 KiB pipe.

B. Down-sampling (E2)

Finding 2 – aliasing. The step *samples[:,3]* applies no low-pass filter. A tone at 9–22 kHz reappeared in the 0–8 kHz band at 0 dB gain; a 9–20 kHz noise burst passed with −0.01 dB loss. Polyphase resampling rejected the tones by 31–95 dB and the burst by 29.6 dB. For a white noise floor the naive step lowers the signal-to-noise ratio by 4.9–5.0 dB (theory 4.77 dB) at floors from −70 to −40 dBFS. The filter costs a median 150 μs per 100 ms block against 1.1 μs here, 0.15 % of the block time; the cost on the target board was not measured.

C. Reminder timing (E3)

TABLE II. REMINDER DELIVERY TIMING

Measure	Result
Delivery error, n = 150 (s)	mean −0.37, SD 0.30, range −0.95...+0.18
Early	85.3 %
Early by more than 0.5 s	36.7 %

Finding 3 – early reminders. The mean error of −0.37 s agrees with Eq. (5); a reminder asked for in 5 s can be spoken after 4.05 s.

D. Local services and quiz parsing (E4)

TABLE III. LOCAL-SERVICE MEASUREMENTS

Measure	Workstation
Quiz start / check, p50 (ms)	0.001 / 0.001
Save fact / create reminder, p50 (ms)	1.2 / 1.9
Study status, p50 / p99 (ms)	0.17 / 0.39
Prompt build, 1 000 / 10 000 facts (ms)	0.8 / 11

Local tools answered within about 2 ms at p50. The memory prompt grows linearly because every stored fact is injected: 10 000 facts produce a 579 k-character prompt (about 145 k tokens at four characters per token), so context use rises with the size of the store.

Finding 4 – answer parsing. *check_quiz_answer* accepted bare forms (16 of 16 correct) but missed all 12 hedged phrases and all 8 ordinal variants, and chose a wrong option in 7 of 8 sentences beginning with the letters A–D, such as “actually it’s C”. Overall accuracy was 44.6 % (25 of 56; 40.0 % on the 50 answerable cases). These utterances were written by us, so the figure measures robustness, not field accuracy; it holds only if the model forwards raw speech instead of a clean letter. Constraining the argument to an enumeration would remove the dependence.

VII. DISCUSSION AND LIMITATIONS

- 1) *What the findings mean for a mentor.* A learner interrupts to slow an explanation down, so audio that outlasts the interruption by about a second (Finding 1) makes the device appear not to listen; the mute gate compounds this by discarding microphone blocks while the model is speaking. Timers and reminders carry the study-coach function, so early firing (Finding 3) is visible to the learner. The quiz parser (Finding 4) matters because a correct answer judged wrong is the one error a tutor cannot afford; it also shows why exact state belongs in local code but needs strict argument contracts from the model. Because the active quiz item is process-local, an interrupted quiz item would have to be re-asked after a restart.
- 2) *Recommendations.* (i) Insert a polyphase low-pass filter before decimation; the cost is negligible. (ii) Shrink the pipe with *F_SETPIPE_SZ* and let the gate follow playback state instead of generation state. (iii) Store reminder times with sub-second precision, or round up. (iv) Write an idempotency key with the delivery, or mark first and accept at-most-once delivery for non-critical reminders. (v) Declare the quiz answer as an enumeration and persist the active item. Items (i) and (ii) were evaluated offline only; they are not part of the shipped build.
- 3) *Threats to validity.* The numbers come from a workstation with stubs for the cloud and the sound card, so they test logic, queueing and operating-system behaviour but not target-board timing, Bluetooth latency or acoustic echo; these remain open. Four items were not measured: end-to-end response latency, the deaf duty cycle of Eq. (4), behaviour during a live session reconnect, and processor load. The capture-queue relations of Eqs. (1) and (2), the supervisor and reconnect tiers of Fig. 2, and the durability of the SQLite store are described as design but were not evaluated experimentally in this paper. The 56-utterance parsing set is author-written.

VIII. CONCLUSION

DESKXI assigns open-ended teaching and interview talk to a cloud speech model and exact state to local services. A bounded drop-oldest capture queue, a writer thread for playback and a restart-based supervisor over a durable SQLite store form a runtime designed to keep capture latency bounded and to preserve study state. Workstation measurements of selected parts of that runtime exposed four limitations: speech that outlasts an interruption, an unfiltered down-sampler, early reminders and a fragile quiz-answer parser. The method needs no robot hardware, only the code, and it found defects that conversation tests would not. Next steps are replication on the target board, evaluation of the capture, reconnect and restart mechanisms there, a gate that tracks playback, and a user study of whether the improved runtime changes how people judge the robot.

IX. ACKNOWLEDGMENT

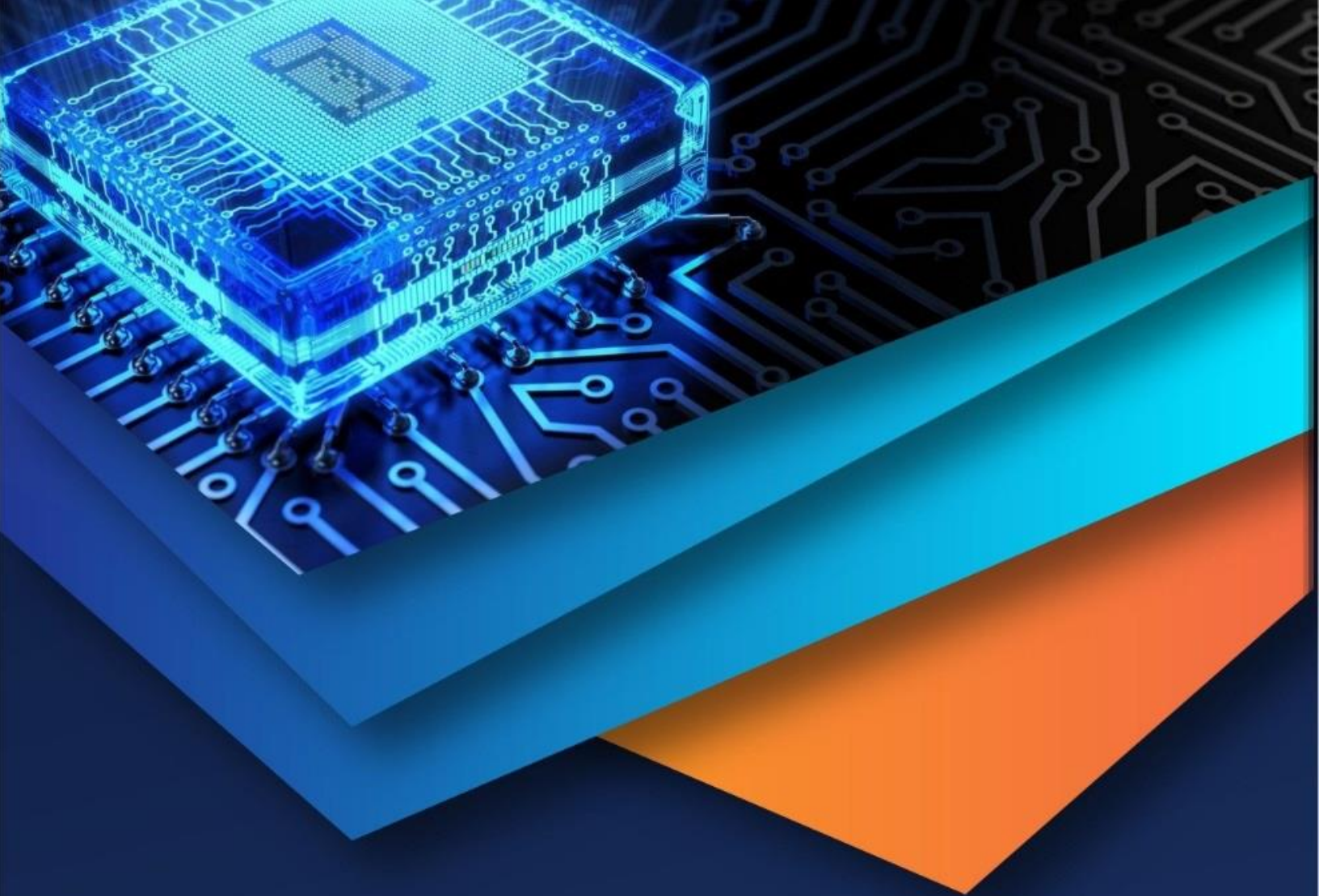
The authors thank Dr. Ruchi Kadwane, Department of Computer Science and Engineering, G H Raisonni University, Amravati, for guidance.

REFERENCES

- [1] G. Skantze, “Turn-taking in conversational systems and human–robot interaction: A review,” *Comput. Speech Lang.*, vol. 67, Art. no. 101178, 2021.
- [2] T. Stivers et al., “Universals and cultural variation in turn-taking in conversation,” *Proc. Natl. Acad. Sci. USA*, vol. 106, no. 26, pp. 10587–10592, 2009.
- [3] ITU-T, “One-way transmission time,” Recommendation G.114, 2003.
- [4] C. Y. Kim, C. P. Lee, and B. Mutlu, “Understanding large-language model (LLM)-powered human-robot interaction,” in *Proc. ACM/IEEE Int. Conf. Human-Robot Interaction*, 2024.
- [5] A. Défossez et al., “Moshi: A speech-text foundation model for real-time dialogue,” *arXiv:2410.00037*, 2024.
- [6] Z. Ma et al., “Language model can listen while speaking,” in *Proc. AAAI Conf. Artif. Intell.*, 2025, *arXiv:2408.02622*.
- [7] Google, “Live API capabilities guide,” Gemini API documentation. [Online]. Available: <https://ai.google.dev/gemini-api/docs/live-api/capabilities> (accessed Oct. 2026).



- [8] Google, "Live API session management," Gemini API documentation. [Online]. Available: <https://ai.google.dev/gemini-api/docs/live-api/session-management> (accessed Oct. 2026).
- [9] Gemini Team, Google, "Gemini: A family of highly capable multimodal models," arXiv:2312.11805, 2023.
- [10] G. Candea and A. Fox, "Crash-only software," in Proc. 9th Workshop Hot Topics Operating Syst. (HotOS IX), 2003.
- [11] D. Patterson et al., "Recovery-oriented computing (ROC): Motivation, definition, techniques, and case studies," Univ. California, Berkeley, Tech. Rep. UCB//CSD-02-1175, 2002.
- [12] J. Armstrong, "Making reliable distributed systems in the presence of software errors," Ph.D. dissertation, Royal Inst. Technol. (KTH), Stockholm, Sweden, 2003.
- [13] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the Internet," Commun. ACM, vol. 55, no. 1, pp. 57–65, 2012.
- [14] J. Dean and L. A. Barroso, "The tail at scale," Commun. ACM, vol. 56, no. 2, pp. 74–80, 2013.
- [15] R. E. Crochiere and L. R. Rabiner, Multirate Digital Signal Processing. Englewood Cliffs, NJ, USA: Prentice-Hall, 1983.
- [16] A. V. Oppenheim and R. W. Schaffer, Discrete-Time Signal Processing, 3rd ed. Upper Saddle River, NJ, USA: Pearson, 2010.
- [17] J. Benesty et al., Advances in Network and Acoustic Echo Cancellation. Berlin, Germany: Springer, 2001.
- [18] SQLite Consortium, "Atomic commit in SQLite." [Online]. Available: <https://www.sqlite.org/atomiccommit.html> (accessed Oct. 2026).
- [19] M. Kleppmann, Designing Data-Intensive Applications. Sebastopol, CA, USA: O'Reilly, 2017.
- [20] T. Belpaeme, J. Kennedy, A. Ramachandran, B. Scassellati, and F. Tanaka, "Social robots for education: A review," Sci. Robot., vol. 3, no. 21, Art. no. eaat5954, 2018.
- [21] T. Kanda, T. Hirano, D. Eaton, and H. Ishiguro, "Interactive robots as social partners and peer tutors for children: A field trial," Hum.-Comput. Interact., vol. 19, no. 1–2, pp. 61–84, 2004.



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)