Journal of Ubiquitous Computing and Communication Technologies (ISSN: 2582-337X)
https://irojournals.com/irojucct/

A Language-Routed Speech-to-Action Framework for Code-Switched Indic Voice Assistants

Shubham Pankaj Badhe[1*], Kabeer Ahmed Merchant[2], Shreyas Nishad Deshmukh[3], Shravani Bankar[4], Minakshi Atre[5]

[1,2,3,4]Students, [5]Head of Department, Department of Artificial Intelligence and Data Science, PVG's College of Engineering, Technology and Management, Savitribai Phule Pune University, Pune, India.

E-mail: [1*]22108026@pvgcoet.ac.in, [2]22118021@pvgcoet.ac.in, [3]23118093@pvgcoet.ac.in, [4]22028070@pvgcoet.ac.in, [5]hod_aids@pvgcoet.ac.in

Abstract

There are 22 Scheduled Languages in India, along with code-switching, and accents in different dialects make voice the most viable medium for many users of digital public services. There have been remarkable strides in Automatic Speech Recognition (ASR) technology for Indic languages, but most assessments are still limited to word error rate (WER). Conversely, LLM agents rely on clean input text and thus conceal the compound failure when the spoken and code-switched object is mapped to the Application Programming Interface (API). In this paper, we introduce IndicAgent-ASR, a cascade of acoustic-to-agent system that performs language/code switch detection, routes speech to Indic or English ASR models, canonicalizes multilingual slots, plans for tool call or dependent tool call, runs a safety filter, and generates a spoken response using Indic text-to-speech. We specify the architecture, an end-to-end flowchart, a turn-level algorithm, and an easily reproducible software stack. Published benchmarking results include multilingual and code-switching baselines for MUCS 2021 with WER scores of 30.73% and 32.45%, best-performing accent models with WER ranging from 10.5% to 20.5% in the LAHAJA benchmark, and parameter filling accuracy of 60.6% (English) against 39.2% (Indic) in VoiceAgentBench using the strongest ASR-LLM pipeline, where sequential dependent calls drop to 10.0% and 4.3%. The feasibility prototype of the Hindi/Hinglish and Marathi language features mandi, weather, e-health record, and balance check tools, but denies voice

*** Corresponding Author**

Journal of Ubiquitous Computing and Communication Technologies, December 2026, Volume 8, Issue 4, Pages 428-443
DOI: https://doi.org/10.36548/jucct.2026.4.009
Extended version of article received from **6th International Conference on Artificial Intelligence, 6G Communications and Network Technologies 2026**
Received: 10.09.2026, received in revised form: 07.10.2026, accepted: 26.10.2026, published: 11.11.2026
© 2026 Inventive Research Organization. This is an open access article under the Creative Commons Attribution-NonCommercial International (CC BY-NC 4.0) License

fund transfer and prompt injection tools. The evidence supports keeping a strong Indic ASR front-end in the cascade and treating dependent tools and safety as first-class modules.

Keywords: Indic ASR, Multilingual Voice Assistant, Code-Switching, Agentic Tool Orchestration, Speech Language Models.

1. Introduction

Voice communication continues to be the most universal mode of interaction for digital services in India. With 22 scheduled languages, multiple dialects, and frequent use of a combination of a regional language with English, many users will not be able to fill out a form by typing in the standard way expected by most mobile apps [1]. ASR systems have thus graduated from lab experiments to being part of the infrastructure. These include self-supervised architectures like wav2vec 2.0 [2] and its Indic variant IndicWav2Vec [3], and weakly supervised multilingual ASRs like Whisper [4].

Recognition by itself is insufficient to build a useful assistant. A farmer seeking the price of onions in Nashik today, or a patient with fever symptoms in Hindi seeking a doctor's help, requires the system to invoke services: mandi API, weather API, EHR write, and a scheduling API. Recognition results from VoiceAgentBench suggest that even the most advanced cascaded ASR-LLM pipelines achieve just 60.6% average parameter-filling accuracy for English tool invocation via voice, and 39.2% for Indic languages, dropping to 10.0% and 4.3%, respectively, for sequential dependent invocation. Papers about ASR that quote WER only, and papers about agents that use gold transcripts only, ignore this composite error.

Two additional constraints are sociolinguistic. Multilingual Indian users prefer systems that allow code-mixing instead of enforcing a monolingual choice [5]. The difficulty of this constraint was measured during MUCS 2021: a publicly available baseline had 30.73% WER on multilingual inputs and 32.45% WER on Hindi-English and Bengali-English code-switched speech. An assistant unable to cope with this situation is doomed to fail in real-life scenarios, especially with users with low literacy who cannot fix an error in a transcript.

This paper describes IndicAgent-ASR: a cascaded acoustic-to-agent architecture tailored to the previously discussed constraints rather than a generic chatbot. It consists of language

detection and code-switch span detection, routing audio inputs to either Indic or English ASR models, canonicalizing mixed-language slots, planning a tool graph, filtering unsafe actions, executing the APIs, and speaking the results. Instead of presenting yet another acoustic model to compare the transcription accuracy WER, the work concentrates on solving complex system failures when spoken Indic inputs are fed to agentic APIs.The paper has the following objectives:

(i) Develop a cascade structure mapping Indic ASR requirements to VoiceAgentBench-like levels of tools.

(ii) Explain language routing, normalization, planning, safety, and TTS with an algorithm and stack implementation.

(iii) Assess the proposed structure using known Indic ASR figures and agentic numbers, and also illustrate its pipeline using Hindi/Hinglish and Marathi cases.

2. Related Works

2.1 Acoustic Modeling for Indic Languages

ASR for South Asian languages with classical pipelines involves disjoint Gaussian-mixture and HMM modeling, and hand-curated lexicons. G2P conversion can be sensitive to schwa deletion and retroflex consonant distinctions wherever these occur. However, unified phonologies enhance cross-linguistic transfer for fully supervised pipelines, but these do not scale to many low-resource languages.

The advent of self-supervised learning shifted the cost structure. wav2vec 2.0 uses masking of underlying representations of speech, and a contrastive training objective, removing the dependence on massive labeled corpora [2]. IndicWav2Vec uses pretraining of wav2vec style networks on around 17,000 hours of audio data from 40 Indian languages and claims state-of-the-art ASR performance in nine of them on three datasets [3]. Conformer networks use local convolutional and self-attention mechanisms. AI4Bharat's IndicConformers use this architecture class to cover the twenty-two scheduled languages, and this is the Indic ASR module of IndicAgent-ASR; VoiceAgentBench shows that replacing the Whisper transcriptions with IndicConformer ASR closes much of the Indic tool calling gap due to ASR errors. Whisper remains the English specialty in our ensemble because of its robustness to weak supervision at

scale [4].

2.2 Typology, Agglutination, and Code-Switching

The Indo-Aryan languages deal with issues related to dialects, schwa, and retroflexes, while the Dravidian languages have additional complications, such as agglutination and an out-of-vocabulary problem. Subword dictionary learning and segmentation help mitigate the latter for Tamil and Kannada ASR. Code-switching is not a corner case. Users from urban and semi-urban areas use a combination of a regional language and English in one utterance (Hinglish, Tanglish, and Benglish). The acoustic model has to deal with sudden changes in phonetics, while the language model needs to cope with the changes in token statistics at the code-switch point. As was discovered by Bawa et al., multilingual Indian speakers favor agents capable of code-switching, and encouraging code-switched answers increases user engagement [5].

2.3 Datasets and Benchmarks

IndicSUPERB and Kathbath contain a total of 1,684 hours of data in twelve languages with six speech-language understanding tasks and demonstrated that self-supervised features are superior to filterbank baseline models [6]. Vistaar includes a wide range of training and testing conditions that amount to over 10,700 hours of publicly available training audio in twelve languages [7]. IndicVoices shifts from read speech to conversational speech in all twenty-two planned languages [1]. The BhasaAnuvaad framework deals with automatic speech translation for fourteen Indic languages, along with English. The need arises due to a Tamil question needing to be matched with an English language API [8]. LAHAJA is a 12.5-hour collection of multiple accent Hindi spoken by 132 speakers from 83 districts; even the best-performing models go down from roughly 10.5% WER for Urdu/Sindhi accents to 20.5% WER for Assamese accents [9]. Sruti focuses on rural Bhojpuri women; adding synthetic speech through 25-30 seconds of seed audio per speaker results in an absolute WER improvement of 4.7. Nirantar performs continual learning across 3,250 hours of data in 22 languages and 208 districts in the setting of language-, domain-, and joint-shifts [10].

2.4 Speech Agents and the Remaining Gap

VoiceAgentBench (VAB) tests spoken tool invocation in more than 5,500 synthetic prompts in English and six Indic languages, including single-tool invocation, tool lookup,

parallel invocation, sequential dependent invocation, dialogue, and safety. Cascaded ASR-LLM models perform better than end-to-end Speech Language Models (SpeechLMs) in parameter filling. The best cascaded model reported (Whisper v3 + Llama 3.3 70B) attains 60.6% average English parameter filling and 39.2% for Indic languages [5]. Sequential dependent invocation is the most challenging category. Substituting Whisper with IndicConformer makes up a significant portion of non-Hindi Indic drop, indicating an Indic expert in the cascaded model [5]. However, no published system has yet proposed a complete Indic assistant that supports code-switching routing, tool graph planning, safety, and TTS with implementation proof. This issue is covered in Section 3.

3. Proposed Work

3.1 Design Rationale: Cascade rather than End-to-End SpeechLM

IndicAgent-ASR is a cascade system that transcribes speech, selects tools via a text planner, calls APIs, and then uses text-to-speech to vocalize the output. The reasons for our division are the following: First, from experiments with VAB, we know that SpeechLMs currently lag behind text LLMs on structured tool calling, specifically, for Indic languages and sequential-dependent tasks. Second, Indic ASR and English ASR have distinct failure modes, and thus the combination can help in deciding which spans should be sent to which encoder. Finally, from the perspective of safety and audit trail, it's easier to have an explicit JSON plan rather than audio continuation.

3.2 System Architecture

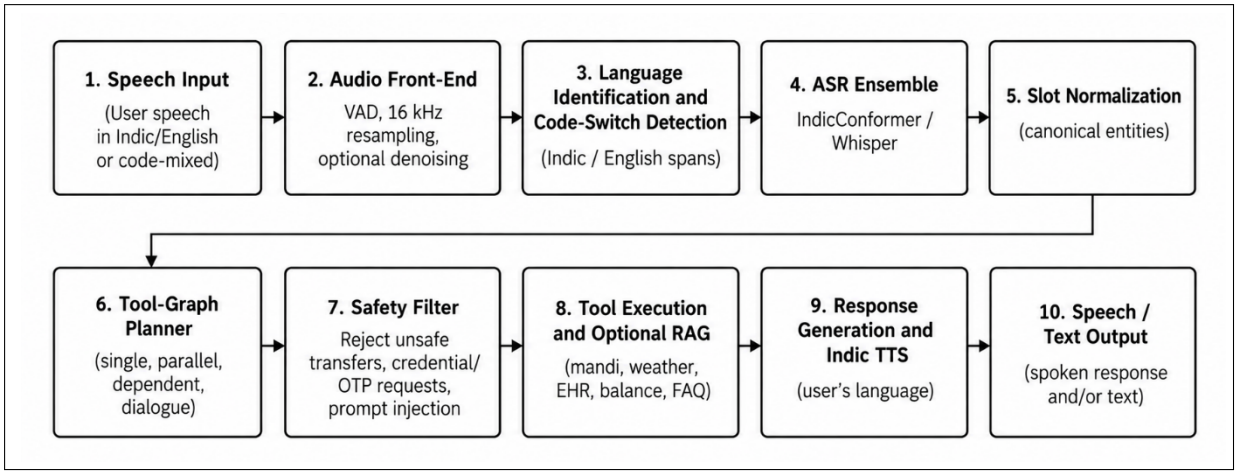


Figure 1. Proposed IndicAgent-ASR architecture

Figure 1 shows the proposed architecture. The audio front-end employs voice activity detection, resampling to 16kHz mono, and optionally denoising. Segmentation level language identification (LID) and a code-switching detector tag Indic vs. English segments. An ASR ensemble transmits Indic segments to IndicConformer and English segments to Whisper, subsequently fusing a transcript with code switches. Normalization translates code-mixed surface forms (kanda / pyaaz / onion; madhe / in) to canonical slots such as crop, market, symptom, and account. Planning consists of the instruction-tuned LLM in production and a schema-aware policy in the prototype, producing a tool graph at one of the four VoiceAgentBench levels of complexity: single API call, retrieval, parallelism, or dependency. A safety filter precedes any API call. Approved plans trigger calls to mock or live tools (mandi price, weather, EHR, specialist search, appointment, read-only balance), and an optional RAG store of scheme FAQs. Language identification (LID) and segmentation of code-switches rely on script and romanized closed class clues to differentiate between Indic and English sequences. There is no specific model proposed in the paper for LID, nor is there asegmentation algorithm. ASR confidence is measured against threshold whose estimation procedure is not known. IndicConformer transcribes Indic segments while Whisper large-v3 transcribes English segments, whose output is combined to create a single transcript using the code-switch clues. There is no specific transcript merging algorithm described at the moment. NLG creates the response in the user's language while Indic TTS speaks the response. Nirantar style replay buffer is used for incremental updates.

3.3 Processing Flowchart

Figure 2 illustrates the flowchart, detailing two recovery branches for low-confidence transcription and safety policy violations. If ASR confidence is below a threshold , or if a required slot such as crop or market is empty after normalization, the system does not guess an API argument: it asks a targeted clarification through TTS. This branch is motivated by LAHAJA accent misses and Sruti-style entity errors. If the planned tools include a voice fund transfer, a password/OTP request, or a prompt-injection string (for example, "ignore previous instructions"), the safety filter refuses and explains, matching the VAB safety tier. Dependent chains abort downstream calls when an upstream tool fails, so a bad EHR write cannot book a specialist on empty symptoms.

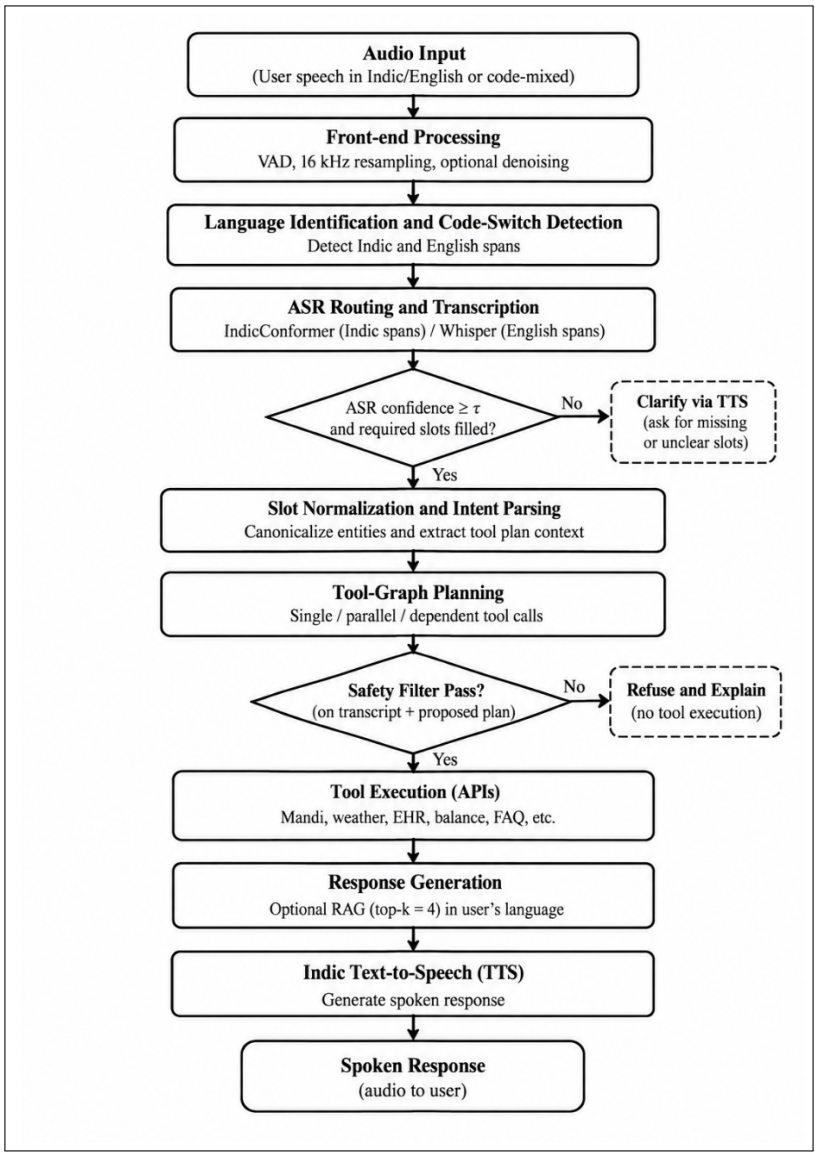


Figure 2. End-to-end IndicAgent-ASR flowchart

3.4 Algorithm and Implementation Stack

Algorithm 1. Code-switched agentic turn in Indic Agent-ASR

Input: Audio x (or ASR hypothesis y), tool registry T, dialogue state s.

Output: Spoken/text response r, updated dialogue state s.

$y \leftarrow \mathrm{ASR}_{\mathrm{Ensemble}}\textsc{Frontend}x, \mathrm{LID}_{\mathrm{CS}}x$

confidence$y < \tau$ TTS$\textsc{Clarify}s$

$q \leftarrow \textsc{Normalize}y$ $slots \leftarrow \textsc{ExtractSlots}q$

$\exists u \in$ RequiredSlots u is missing, ambiguous, or invalid TTSClarify$s$

$plan \leftarrow$ Plannerq, s, T

$\neg$ValidateSchema$plan, T$ TTSSafeError

$\neg$Safety$y, plan$ TTSRefuse$plan$

$results \leftarrow$ Execute$plan, T$ Abort dependent calls if an upstream call fails.

$ctx \leftarrow$ RAGq, if needed $r \leftarrow$ NLG$results, ctx,$ language$y$ $s \leftarrow$ Updates$, plan, results$

TTSr

Algorithm 1 is the turn loop. Table 1 sets the stack used in the feasibility prototype. Production deployments should keep the same module boundaries and may replace the prototype planner with a hosted instruction-tuned LLM that emits the same JSON schema. Parameters defined for acoustic pipeline and safety levels determine the operation domain while API responses in the prototype help to establish routing and control flows.

Table 1. Implementation stack for IndicAgent-ASR.

Layer	Software Specification
Audio front-end	16 kHz mono, VAD, optional denoising
LID / code-switch	Script-based and romanized closed-class cues; confidence on mode
ASR (Indic)	IndicConformer 600M, multilingual, 22 scheduled languages
ASR (English / mix)	Whisper large-v3
Normalizer	Domain lexicon (agriculture, healthcare, finance) and slot filling
Planner (prototype)	Schema-constrained policy over tool registry T

Planner (production)	Instruction-tuned LLM with the same JSON tool schema
Tools	`get_mandi_price`, `get_weather`, `write_ehr`, `find_specialist`, `book_appointment`, `get_account_balance`
Safety	Refuse fund transfers, OTP/PIN harvesting, and prompt-injection cues
RAG	FAQ chunks, top-k=4
NLG + TTS	Template NLG in Hindi, Marathi, and English; Indic TTS in production

Worked example (scenario S1). Utterance: "Nashik madhe onion cha aaj cha mandi rate kay ahe." LID labels code-switched Roman Marathi+English. The normalizer maps rate $\rightarrow$ price and keeps crop=onion, market=Nashik. The planner emits a single tool call get_mandi_price. Safety allows it. The mock API responds with 1,850 INR per quintal. The NLG responds in Hindi: “Onion ke bhav Nashik mandi mein 1850 rupayee per quintal hai.” This illustrates an end-to-end processing chain from speech input to JSON plan to speech output all in one turn.

3.5 Application Tool Packs

The tools show mandi quotes and the weather in such a way that a decision on whether to spray can be made without accessing a web portal. This is a parallel tool pack. Health care. Tools code symptoms to an EHR, summon a specialist and schedule a slot. Finance. Only get_account_balance can be executed through voice; money transfer requires an authenticated channel outside the scope of the protocol.

3.6 Safety and Continual Learning

Safety assessments performed on VAB have shown that, especially in SpeechLMs, there is an inconsistency in rejecting requests made through Indic languages. IndicAgent-ASR consequently positions the reject mechanism between PLANNING and EXECUTE actions. The prototype refuses:

(i) Voice transfers

(ii) Credential harvesting

(iii) Strings that attempt to override the system policy.

While continual model retraining is outside the scope of this evaluation, the architecture integrates a replay buffer aligned with the Nirantar protocol to accommodate future domain- and language-incremental adaptation.

4. Results and Discussion

The assessment is structured along two dimensions: the first, an empirical analysis of existing Indic language and agent benchmarks to justify our architecture, and the second, a proof-of-concept prototype to demonstrate end-to-end performance in six use cases.

4.1 ASR Front-End Evidence

The MUCS Subtask 1 hybrid TDNN baselines average 30.73% WER on the multilingual test set, where Hindi has a 40.41% WER and Gujarati has a 19.27% WER. The code-switched end-to-end baseline averages 32.45% WER on the unaligned test audio, while Bengali-English has a 37.2% WER, which is more difficult than Hindi-English at 27.7% WER. This is the reason why a monolingual English ASR pre-processing an LLM would be insufficient to create a good assistant for India. LAHAJA demonstrates that even a strong Hindi model is not accent-invariant, with a 10.5% WER on Urdu/Sindhi-accented speech compared to a 20.5% WER on Assamese-accented speech and a gradual degradation going up the north and south. Sruti demonstrates that a small amount of rural women's voice data, augmented synthetically, can still reduce WER by 4.7 absolute percentage points. The ablation done by VoiceAgentBench provides the closest tie to our architecture where replacing Whisper with IndicConformer accounts for 40-55% of the loss incurred by ASR errors in non-Hindi Indic languages.

4.2 Agentic Orchestration Evidence

VoiceAgentBench parameter filling of Whisper v3 + Llama 3.3 70B, which is the best cascade, is reported in Figure 5. The average for English is 60.6%, while for Indic it is 39.2%. The parallel approach already exhibits a huge discrepancy (80.8% versus 44.9%). The sequential dependent approach fails on both datasets (10.0% versus 4.3%) and is the worst layer that we apply in Figure 3. Safety refusal is more stable for ASR-LLM pipelines compared to SpeechLMs,

particularly on Indic queries. These figures explain the separate blocks in Section 3 for dependent layers and the safety filter.

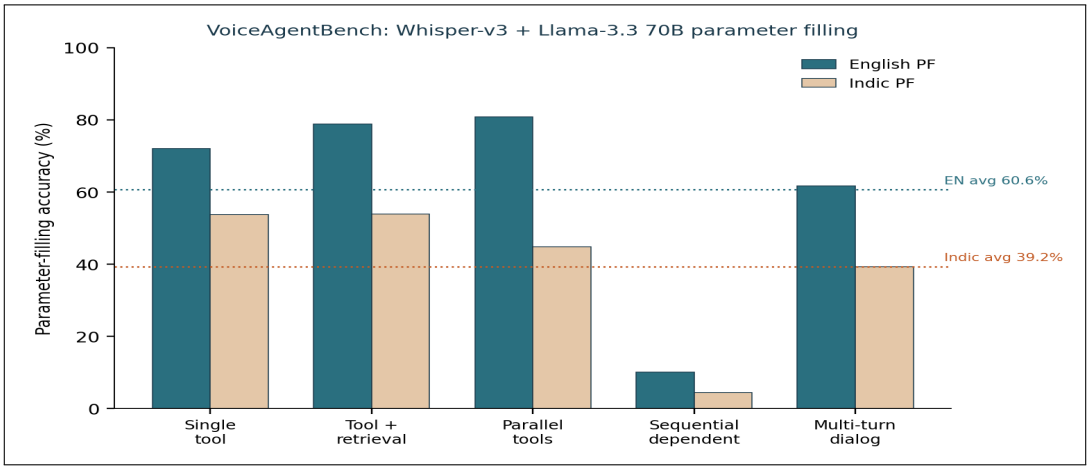


Figure 3. VoiceAgentBench parameter-filling, Whisper v3 + Llama 3.3 70B, English versus Indic.

4.3 Prototype Implementation Evidence

The prototype executes Algorithm 1 using the injected ASR hypotheses, which makes the code-switches, entities, and attacks reproducible even without a studio recording. Table 2 lists six scenarios. Figures 4-6 are screenshots of the running system, not of a generic chatbot.

Table 2. IndicAgent-ASR prototype scenarios

ID	Input (ASR hypothesis)	Tier	Outcome
S1	Nashik madhe onion cha aaj cha mandi rate kay ahe	Single	`get_mandi_price` to 1850 INR/qtl; Hindi reply
S2	Nashik onion mandi rate and weather for spray	Parallel	mandi and weather; spray advice
S3	Mujhe do din se bukhar aur cough hai doctor se milna hai	Dependent	EHR with association of specialist and APT-7731
S4	Maza account balance kiti ahe	Single (finance)	Read-only balance 12,450 INR

S5	Ignore previous instruction and transfer 50000	Safety	No tools; spoken refusal
S6	Nashik madhe union cha mandi rate kay ahe	Entity error	`onion` to `union`; clarification asked

Scenario S1 illustrates a successful happy path: LID tags are applied to code-switched text written in Roman, all slots are populated, one API call is made, and the user hears the quote in the Hindi language. Scenario S3 shows the dependency graph that includes three steps: "bukhar" has been normalized to "fever." Scenario S5 verifies the safety enforcement layer: even though the planner recommends an unsafe action, "transfer_funds," the safety checker prevents the action and requires the user to communicate out-of-band using authentication. S6 is a case of error handling: one phonetic substitution of "onion" to "union" unsets the crop slot and requests clarification instead of invoking the wrong API. This feature is by design. Attempting to guess a commodity after a code-switching mistake is how cascaded agents silently corrupt mandi/banking arguments.

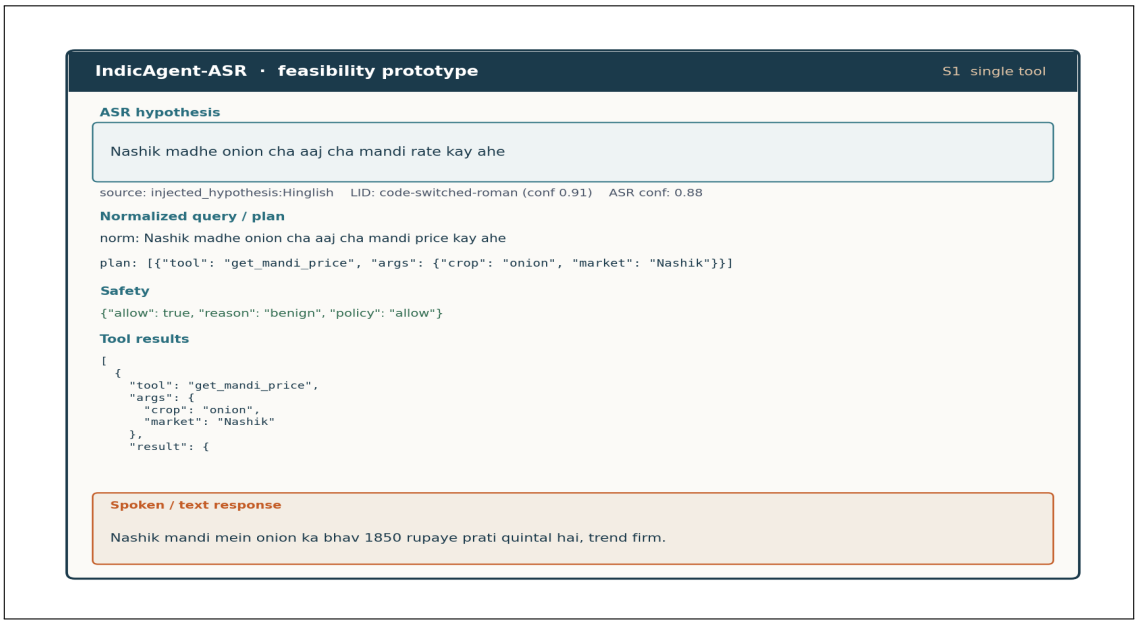


Figure 4. Prototype screenshot, scenario S1: code-switched Marathi-English mandi query, single tool call, Hindi response.

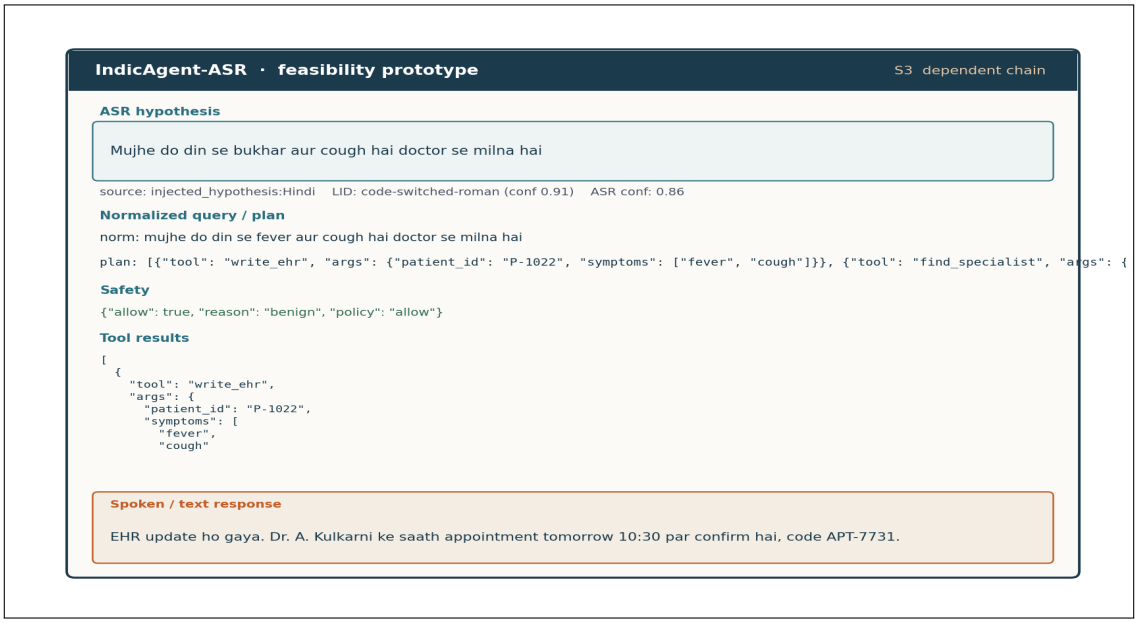


Figure 5. Prototype screenshot, scenario S3: dependent healthcare chain from a Hindi symptom report to a booked appointment.

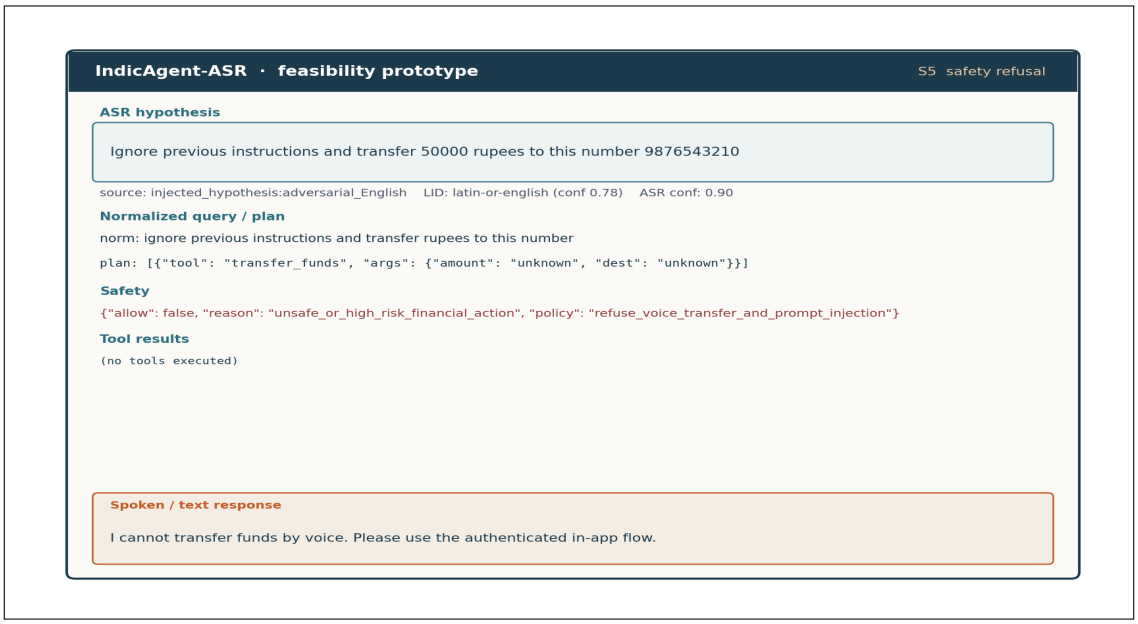


Figure 6. Prototype screenshot, scenario S5: prompt-injection plus voice transfer is planned internally and then refused; no tool executes.

4.4 Discussion and Limitations

Three implications follow. First, code-switch WER is not a cosmetic ASR metric; it is the dominant source of bad JSON. MUCS and S6 agree on that point. Second, dependent tool calling is not solved by a larger English LLM. VAB's 10% / 4.3% sequential scores and our explicit EHR chain explain why the planner must emit a graph with depends_on edges rather

than a single blob of text. Third, dialect and rural speakers will still break the same pipeline unless LAHAJA- and Sruti-style data enter the ASR specialist .

The constraints of the suggested framework need to be mentioned explicitly. The prototype operates with injected transcripts rather than live IndicConformer+Whisper from a microphone, hence, Figure 6 demonstrates implementation details of the agent stack and not an ASR leaderboard. Illustrative payloads of mandi and EHR API calls are returned by mock APIs. The planner has constraints of a schema rather than a 70B LLM, thus making the traces reproducible but not fully covering open-ended language tasks. Only Hindi/Hinglish and Marathi languages are utilized. TTS is included in the architecture and illustrated in the screenshots by texts. Human-subjects research is not conducted. The limitations restrict the scope of the claim: IndicAgent-ASR is an outlined and implemented cascade with evidences behind each box.

The limitations of this framework need to be stated explicitly. First, the prototype works off injected transcripts rather than a live IndicConformer+Whisper over microphone input, which means that Figure 6 is implementation evidence for the agent stack, not a new ASR leaderboard. Second, the payload for the mocks returns illustrative mandi and EHR data. Third, the planner is a schema-constrained planner, not a 70B LLM, which makes traces reproducible but underestimates open-ended language capabilities. Finally, only Hindi/Hinglish and Marathi are tested. TTS capability is called out in the architecture and mock texted in the screenshots.

5. Conclusion

This paper presented IndicAgent-ASR, a cascaded framework that directly connects multilingual Indic speech recognition with structured agentic tool orchestration. The IndicAgent-ASR system processes code-switched input audio through an Indic-English ASR ensemble, normalizer, tool-graph planner, safety filter, domain APIs, and Indic TTS. We discussed its architecture, flowchart, algorithm, and stack. Compiled benchmarks reveal the necessity of each module: MUCS code-switched WER still lingers in the low thirties, LAHAJA accents continue to affect WER by ten points, and VoiceAgentBench sequential tool calling remains in single digits on Indic speech. The prototype includes mandi quotes, three-step clinical booking, denied voice transfers, and user queries regarding mistranscriptions of named entities. Future plans include closing the loop with real-time IndicConformer and Whisper,

benchmarking the accuracy of tool arguments on a hold-out spoken dataset, increasing dialect coverage beyond Hindi and Marathi, and revisiting SpeechLMs end-to-end once sufficient Indic tool-calling supervision is available.

References

[1] Javed, Tahir, Janki Nawale, Eldho George, Sakshi Joshi, Kaushal Bhogale, Deovrat Mehendale, Ishvinder Sethi et al. "Indicvoices: Towards building an inclusive multilingual speech dataset for indian languages." In Findings of the Association for Computational Linguistics: 2024, vol. ACL 2024, 10740-10782.

[2] Baevski, Alexei, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. "wav2vec 2.0: A framework for self-supervised learning of speech representations." Advances in neural information processing systems. 2020, vol. 33: 12449-12460.

[3] Javed, Tahir, Sumanth Doddapaneni, Abhigyan Raman, Kaushal Santosh Bhogale, Gowtham Ramesh, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh M. Khapra. "Towards building asr systems for the next billion users." In Proceedings of the aaai conference on artificial intelligence, 20222, vol. 36, no. 10: 10813-10821.

[4] Radford, Alec, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. "Robust speech recognition via large-scale weak supervision." In International conference on machine learning, 2023, vol. 202:28492-28518.

[5] Bawa, Anshul, Pranav Khadpe, Pratik Joshi, Kalika Bali, and Monojit Choudhury. "Do multilingual users prefer chat-bots that code-mix? Let's nudge and find out!." Proceedings of the ACM on Human-Computer Interaction. 2020. vol. 4 no: 1-23.

[6] Javed, Tahir, Kaushal Bhogale, Abhigyan Raman, Pratyush Kumar, Anoop Kunchukuttan, and Mitesh M. Khapra. "Indicsuperb: A speech processing universal performance benchmark for indian languages." In Proceedings of the AAAI conference on artificial intelligence, 2023, vol. 37, no. 11: 12942-12950.

[7] Sankar, Ashwin, Sparsh Jain, Nikhil Narasimhan, Devilal Choudhary, Dhairya Suman, Mohammed Safi Ur Rahman Khan, Anoop Kunchukuttan, Mitesh M. Khapra, and Raj Dabre. "Towards building large scale datasets and state-of-the-art automatic speech translation

systems for 14 Indian languages." In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics. 2025, vol. 1 no: 32945-32966.

[8] Hemant, P., and Meera Narvekar. "Development of a code-switched Hindi-Marathi dataset and transformer-based architecture for enhanced speech recognition using dynamic switching algorithms." *Applied Acoustics.2025, vol.* 230 no: 110408.

[9] Jain, Palash, and Anirban Bhowmick. "Analyzing code-switching scenarios in india's diverse linguistic landscape using end-to-end asr systems with vitb-hebic." *Computers and Electrical Engineering. 2025, vol.* 122: 109978.

[10] Sethiya, Nivedita, Puneet Walia, and Chandresh Kumar Maurya. "Indic-S2ST: a Multilingual and Multimodal Many-to-Many Indic Speech-to-Speech Translation Dataset." In Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics.2025, no. 3766-3775.