



Hands-Free Railway Booking: Designing and Audio-Only Reservation Interface for Blind and Partially Sighted Users

Mohammed Irfan M¹, Pranesh A², Adharsh G S³, Prof. Vani M P⁴

E-mails: mohammedirfan.h2025@vitstudent.ac.in¹

E-mails: pranesh.a2025@vitstudent.ac.in²

E-mails: adharsh.gs2025@vitstudent.ac.in³

E-mails: mpvani@vit.ac.in⁴

ABSTRACT

Buying a train ticket is one of those tasks most people do without a second thought, yet for a person who cannot see, the standard online journey—scanning timetables, clicking seat maps, reading confirmation screens—can be genuinely impossible to complete alone. This paper documents the design of a fully audio-based reservation system intended to close that gap. Instead of adding accessibility features to an existing visual interface, the system was built from the ground up with voice as its only interaction channel: users speak to it, and it speaks back. No screen interaction is required at any stage.

The technical foundation consists of three well-established components—automatic speech recognition to capture what the user says, a language understanding layer to work out what they mean, and a speech synthesizer to deliver responses—connected to a standard railway booking back end. The harder engineering challenge is not assembling these parts but making the assembled system behave consistently in the kinds of imperfect real-world conditions that a blind traveler actually encounters: background noise, regional pronunciation, patchy mobile coverage, and the occasional wrong word. This paper documents the design decisions made in response to those challenges, describes how the system is structured, and sets out clearly what it cannot yet do.

Keywords: audio-only interface, speech recognition, rail ticket booking, assistive system, visual disability, natural language understanding, spoken confirmation

Introduction

Accessibility in digital services is often discussed in terms of guidelines and compliance checklists. This project started somewhere else: with the observation that a blind person attempting to book a train ticket on most Indian railway platforms will hit obstacles within the first few seconds that no set of WCAG guidelines has fully solved. Dynamic seat-availability tables do not render predictably in screen readers. Seat-selection graphics cannot be navigated without vision. Session timeouts discard work in progress without warning. The cumulative effect is that many visually impaired travelers either ask a sighted person to handle the booking or simply do not travel independently.

The project described here adopts a more direct approach: remove the visual interface entirely and replace it with conversation. A user opens the application, hears a brief prompt, and from that point onward the interaction proceeds entirely through spoken language. The user says what they want; the system responds in speech; the ticket gets booked. No screen, no keyboard, no touchscreen gestures.

Three technologies make this possible. Automatic speech recognition (ASR) converts the user's spoken words into text. A natural language processing (NLP) layer interprets the resulting text to determine both what the user wants to do and the specific details—origin, destination, travel date, class—needed to act on that request. A text-to-speech (TTS) engine converts the system's responses back into audio for delivery. Each technology is commercially mature; the design work here lies in connecting them into a pipeline that handles the variability and imprecision of real speech reliably enough to be trusted with a financial transaction.

The paper is structured as follows. The methodology section explains the design principles that shaped the interaction model. The architecture section describes how the system is layered. The modules section covers each component in detail. Diagrams illustrate the data flow, the use cases, and the session activity. The limitations section is deliberately candid about where the current design falls short. The future scope section identifies the improvements most likely to increase real-world impact.

Methodology

The design process was guided by a single governing constraint: every part of the user experience has to work without sight. That constraint turns out to be surprisingly demanding, because most software design—even software with accessibility features—still assumes the user can glance at the screen to orient themselves. When that assumption is removed, even small decisions about how prompts are worded or how

errors are communicated become critical.

The methodology began with a set of representative scenarios written out as transcripts of how a session should actually go. One scenario involved a user who knew their destination but had not yet decided on a date. Another involved a user who gave a station name that the ASR engine might plausibly mishear. A third involved a user who got most of the way through a booking and then asked to change the class. Writing these out before touching any code forced early decisions about how the system should handle incomplete input, low-confidence transcriptions, and mid-flow changes of mind.

Four principles emerged from this exercise. The first is that the system must never leave the user uncertain about what to do next. Every prompt ends with a clear, specific invitation: “Please say the name of your destination station.” The second is that no consequential action is irreversible on the first attempt. Any booking or cancellation triggers a spoken summary that the user must explicitly confirm before the transaction proceeds. The third is that when the system is uncertain, it says so—once, plainly—and asks a single follow-up question rather than generating an error message or presenting a numbered menu that the user would have to remember. The fourth is that confirmation prompts are slow and deliberate; the system does not rush through a summary of a financial transaction.

On the technical side, these principles translate into a processing pipeline connecting a microphone to an ASR service, the transcription output to an NLP classifier, the classified intent and extracted entities to a reservation back end, and the back end’s response to a TTS engine. The pipeline was validated by running scripted sessions covering all major use cases and evaluating each for both correctness of outcome and naturalness of interaction.

DETAILED SYSTEM ARCHITECTURE

The system is divided into four layers. Each layer has a clearly bounded responsibility and communicates with the layers immediately above and below it through defined interfaces. This structure was chosen specifically to allow individual components to be upgraded—a better ASR model, a different railway API—without requiring changes elsewhere in the system.

Spoken Input and Output Layer

This is the layer the user encounters. It has two components: a microphone input path that captures the user’s voice and a speaker output path that plays the system’s responses. By design, this layer contains no processing logic. It does not interpret what the user says; it simply records audio and plays back whatever it receives from above. Keeping this layer thin ensures that it can be adapted for different hardware—a phone microphone, a smart speaker, a Bluetooth headset—without affecting the application’s behaviour.

The prompts delivered through this layer follow a consistent structure throughout every session. They are short, use plain language, and always specify what kind of input the system is waiting for. Research on voice interfaces consistently shows that users lose confidence quickly when they are not sure whether the system is listening or what it expects them to say; the prompt design addresses this directly.

Language Understanding Layer

Audio captured at the input layer is passed to an ASR engine, which produces a text transcript. That transcript is then processed by an NLP component, which assigns the utterance an intent label—one of a small fixed set of things the system can do—and extracts the named entities relevant to that intent. For a booking request, the relevant entities are origin station, destination station, travel date, travel class, and passenger count.

Both the ASR and NLP stages have to cope with imprecision. The ASR stage deals with acoustic imprecision: background noise, accented speech, unusual station names. The NLP stage deals with linguistic imprecision: the same request phrased in many different ways, partial information, and phrases that could belong to more than one intent. Both stages return confidence scores alongside their primary outputs; the application layer uses these scores to decide whether to act on a result or ask for clarification.

Reservation Logic Layer

This layer is the decision-making core of the system. It receives a structured representation of the user’s intent and entities from the language understanding layer, determines what operation is needed, and coordinates the steps to carry it out. For a ticket booking, this involves searching for matching trains, presenting options to the user, collecting any missing details, assembling a transaction summary, and waiting for spoken confirmation before committing.

Error handling is also the responsibility of this layer. When the back end reports that no trains are available, or that a seat has just been taken by another transaction, the layer translates that outcome into a spoken explanation and, where possible, offers an alternative course of action. The layer also enforces the confirmation requirement: it will not submit a booking or cancellation without receiving an explicit spoken yes from the user first.

Data Access Layer

The data access layer manages the connection to external systems: the railway schedule and availability database in test deployments, and a live railway operator API (such as the IRCTC API in production) when real-time data is required. It is responsible for constructing correctly formatted API requests, parsing responses, managing authentication credentials and session tokens, and ensuring that sensitive

information—passenger details, payment data—is handled according to applicable security and privacy requirements.

This layer also manages the seat-hold mechanism that keeps a chosen berth reserved during the time between the user selecting it and the payment being confirmed. If the payment fails or the user abandons the session, the hold is released so that the seat returns to the available pool

End-to-End Data Flow

User speaks → microphone captures audio → ASR produces transcript → NLP extracts intent and entities → reservation logic executes the requested operation → data access layer queries or updates the back end → reservation logic composes a response → TTS converts the response to speech → speaker delivers the audio to the user. This cycle repeats for each turn of the conversation.

IMPLEMENTATION OVERVIEW

Selecting and configuring the right tools at each layer requires balancing accuracy, cost, connectivity requirements, and the specific characteristics of Indian railway vocabulary.

At the ASR layer, the primary challenge is station names. India has several thousand railway stations, many with names that are long, regionally specific, and difficult for models trained predominantly on European or North American English. Visakhapatnam, Nagercoil, and Lokmanya Tilak Terminus are examples of names that standard ASR models frequently misrecognize. The most effective mitigation is to supply the ASR engine with a custom vocabulary or phrase list containing station names and common railway terms (class codes like SL, 3A, 2A; terms like PNR, berth, quota). Commercial speech APIs from Google, Amazon, and Microsoft all support this kind of domain adaptation. For deployments where internet access is unreliable, on-device models such as Vosk or OpenAI Whisper provide a fallback, accepting lower accuracy in exchange for independence from connectivity.

The NLP layer requires a classifier trained on a dataset of railway booking utterances. Because the same intent can be expressed in dozens of different ways in everyday speech, the training set must cover that variation. It also needs to handle the fact that users frequently mix English with Hindi or a regional language mid-sentence: “Mujhe Mumbai se ek sleeper ticket chahiye to Delhi” should parse the same as “I need one sleeper ticket from Mumbai to Delhi.” Named-entity recognition for station names benefits from a reference list (gazetteer) of official station names and their common variants and abbreviations.

On the output side, the TTS engine needs careful configuration for railway-specific number formats. The five-digit train number 12301 should be spoken as individual digits (“one two three zero one”), not as a large number. Fare amounts should be spoken with the currency unit: “two hundred and forty rupees.” Speaking rate and pitch should be user-adjustable, stored in a profile, and applied automatically at session start—some users find slower speech easier to follow; others find it frustrating.

For deployment, a mobile application is the practical starting point. Users carry their phones everywhere, the microphone and speaker quality is sufficient for voice interaction, and mobile data provides the connectivity needed for live schedule queries. A lightweight web version extends access to users on shared computers or low-storage devices.

SYSTEM MODULE

1: Voice Capture and Transcription

This module is active throughout a session, opening the microphone after each system prompt and closing it once the user has finished speaking. It applies a noise-reduction filter before passing audio to the ASR engine, which improves transcription quality in busy environments. When a transcript is returned, the module plays a short acknowledgement sound—a single tone—so the user knows their input was captured even before the system has finished processing it.

If the ASR engine returns a result with a confidence score below a configurable threshold, the module does not pass it downstream. Instead it generates a re-prompt: “I’m sorry, I didn’t quite follow that. Could you say it again?” This happens before any application logic runs, which prevents a misheard word from producing a confusing or incorrect search result. The re-prompt fires at most twice per turn; after a second failed attempt, the module passes the best available transcript downstream with a low-confidence flag attached, so the application layer can decide how to handle it.

Module 2: Intent Classification and Entity Extraction

This module receives a text transcript and returns two things: the user’s intent (what they want to do) and the entities embedded in the utterance (the specific values needed to do it). Intent categories include: search for trains, check seat availability, book a ticket, cancel a booking, look up a PNR, and end the session. Entities for a booking intent include origin station, destination station, travel date, travel class, and number of passengers.

The module handles entity extraction pragmatically. Date expressions are normalized: “next Tuesday,” “the fourteenth,” and “14th March” all map to a concrete date. Station names are matched against the reference gazetteer using fuzzy matching to account for slight mispronunciations or variant spellings. When the intent is clear but one or more entities are missing, the module returns the partial result with a flag indicating which entities are absent, allowing the application layer to ask one focused question rather than prompting the user to repeat the whole request.

Module 3: Booking and Data Management

This module sits between the application logic and the external railway data. It receives structured operation requests—find trains satisfying given criteria, check availability for a specific service, reserve a seat, cancel a booking, retrieve PNR status—and translates each into the appropriate API call or database query. Responses are parsed and returned in a normalized format that the application layer can work with regardless of which back end is in use.

The module distinguishes between infrastructure failures (the API is unreachable, the response is malformed) and business-logic outcomes (no matching trains, chosen class fully booked, cancellation window expired). Each type of failure produces a structured response that the application layer converts into an appropriate spoken message. For multi-step bookings, the module maintains the seat hold between the selection step and the payment confirmation, and releases it automatically if the session times out or the payment fails.

Module 4: Response Synthesis and Confirmation Management

This module converts text responses from the application layer into spoken audio and manages the timing and sequencing of the system's side of the conversation. When multiple pieces of information need to be delivered together—for example, a list of three matching trains with their departure times—the module reads them in a consistent format with a short pause between items, giving the user time to absorb each one before the next.

Confirmation management is the module's most critical responsibility. Whenever the application layer indicates that a transaction is ready to execute, this module takes over: it reads the full transaction summary at a deliberately slow pace, naming every detail (train, date, class, passenger name, fare, refund policy where relevant), then pauses and waits for a spoken yes. A yes triggers execution. Any other response—including silence beyond a timeout threshold—causes the module to ask a clarifying question: "Should I go ahead and confirm this booking, or would you like to change something

ARCHITECTURE DIAGRAM

The architecture diagram (Figure 1, shown above) maps the component structure described in the preceding sections. Several design properties visible in the diagram are worth drawing out explicitly.

1. The input/output layer is strictly a conduit. It does not process or interpret anything; it only moves audio in and out. This makes it easy to substitute different hardware without touching any application code.
2. The ASR and NLP components run sequentially, not in parallel. NLP never sees raw audio; it only sees text produced by ASR. This separation keeps each component's scope well defined.
3. All back-end communication is funnelled through the data access layer. No other component holds credentials or makes API calls. This concentration of data access responsibility simplifies security auditing and back-end migration.
4. The TTS component is a terminal node in the processing graph. It receives finished output and speaks it. It does not make any decisions about content.

The consequence of this structure is that the system can evolve component by component. Switching from one ASR engine to another affects only that component's adapter code. Integrating a new railway API affects only the data access layer. Adding support for a new language affects primarily the NLP and TTS

Proposed System Architecture for Voice Based Train Reservation System

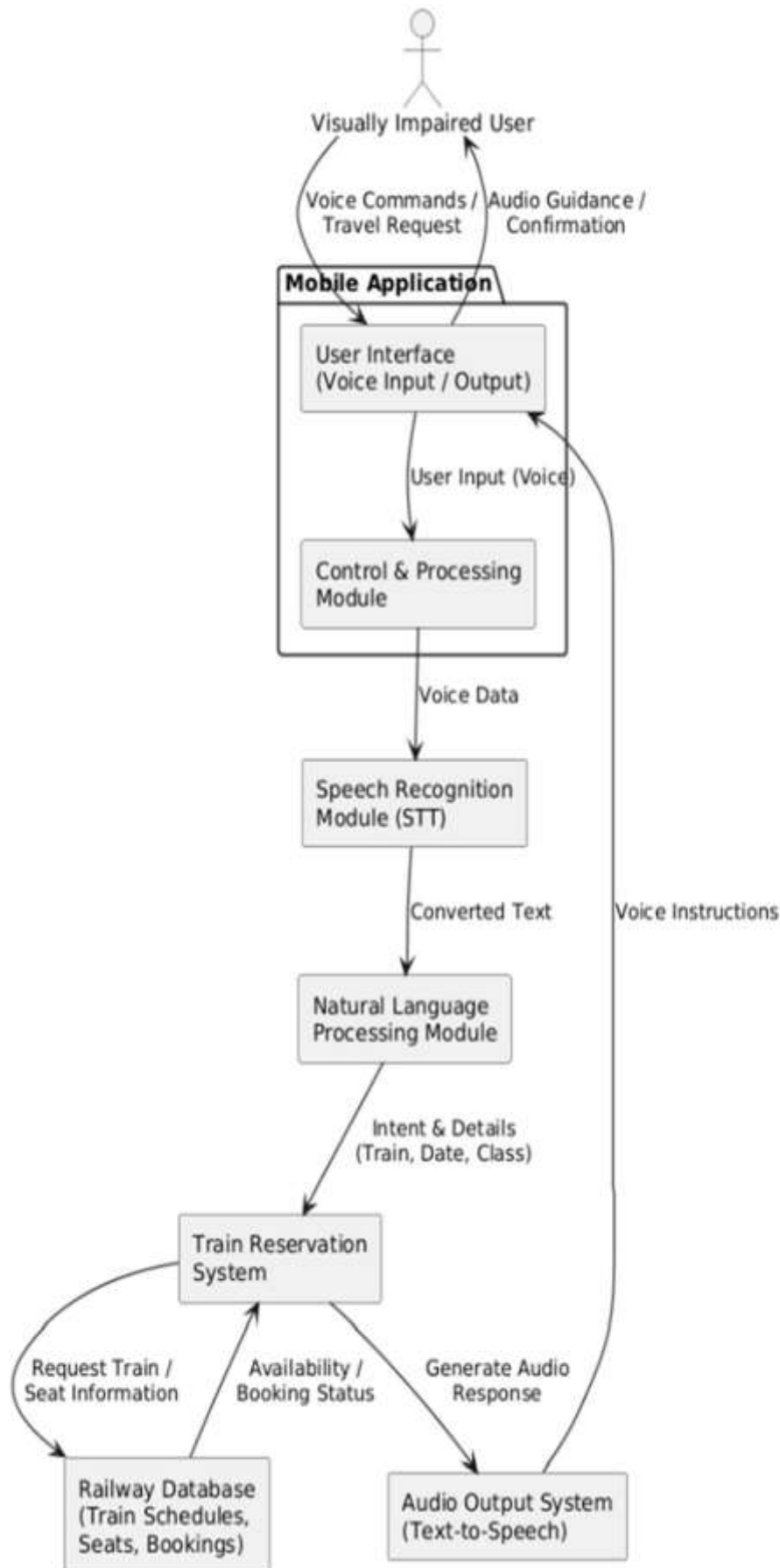
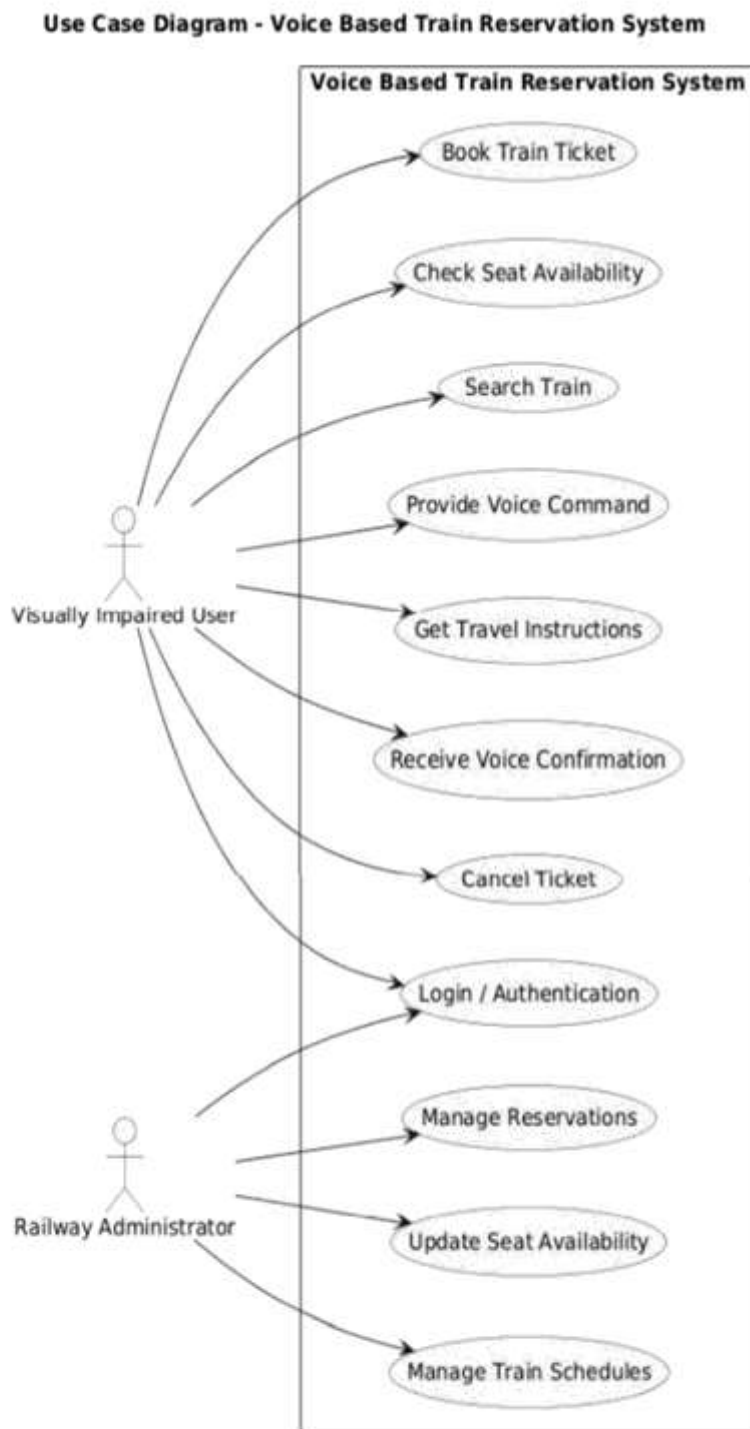


Figure 2: Use case diagram for the voice-based rail reservation sys



System Actors

The system has two actors, operating in completely separate contexts. The blind or partially sighted traveler is the primary actor. Every interaction they have with the system occurs through voice; they never touch a screen or operate a keyboard. The railway administrator is the secondary actor, responsible for maintaining the data that the system depends on—train schedules, seat inventory, and booking records. The two actors do not interact through the system; their use cases are entirely independent.

User Capabilities

- **Voice login:** The user identifies themselves using a spoken personal identification number or a registered passphrase; no typing is involved at any stage of authentication.
 - **Train search:** The user states an origin station, destination station, and travel date; the system reads back a list of matching services with departure times and estimated journey durations.
 - **Availability check:** For any service in the search results, the user can request a breakdown of seat availability and current fare by class.
 - **Ticket booking:** The user selects a service and class, provides the required passenger details, hears a complete transaction summary, confirms it verbally, and receives a spoken booking reference number.
 - **Ticket cancellation:** The user provides a PNR or booking reference, hears the applicable cancellation policy and refund amount, confirms the cancellation verbally, and receives spoken confirmation of the outcome.
 - **Outcome confirmation:** Following any completed action, the system reads a full spoken summary; the user does not need to check a screen to know whether the operation succeeded.
- Administrator Capabilities
 - **Reservation oversight:** Review active bookings, handle disputes, and correct anomalies in the booking records.
 - **Inventory management:** Adjust seat counts manually to reflect coach substitutions, rake changes, or technical discrepancies.
 - **Schedule administration:** Add and update service information, mark trains as cancelled or delayed, and adjust timetable data.

ACTIVITY DIAGRAM

The activity diagram in Figure 3 traces the path through a complete session. The session opens with a welcome prompt; the system speaks first so the user knows it is ready. The user then speaks, triggering the transcription and intent-classification cycle. The first decision point checks whether the classified intent is actionable: if the intent is clear, the system proceeds; if the transcript is too ambiguous to classify confidently, the system asks a single clarifying question and loops back.

Once an intent and its required entities are confirmed, the relevant operation is executed. Informational requests—schedule lookups, availability checks, PNR status queries—return a spoken answer immediately. Transactional requests—bookings and cancellations—enter the confirmation sub-flow: the system reads a complete summary, waits for an explicit spoken yes, and only then submits the transaction. After any completed operation, the system asks whether the user needs anything else; a negative response—or silence beyond the session timeout—closes the session.

Algorithm for the Session Activity Flow

5. Session starts; system delivers welcome prompt and opens microphone.
6. User speaks; system records audio and passes it to the ASR engine.
7. ASR produces text transcript; confidence score is evaluated.
8. If confidence is low, issue re-prompt (maximum two attempts per turn).
9. NLP classifies intent and extracts named entities from transcript.
10. If entities are incomplete, issue targeted clarifying prompt and return to step 2.
11. Application layer determines operation type (informational or transactional).
12. Data access layer executes required query or API call.
13. If transactional: read full summary aloud; wait for spoken yes; submit only on confirmation.
14. Deliver response via TTS; prompt for next input or close session.

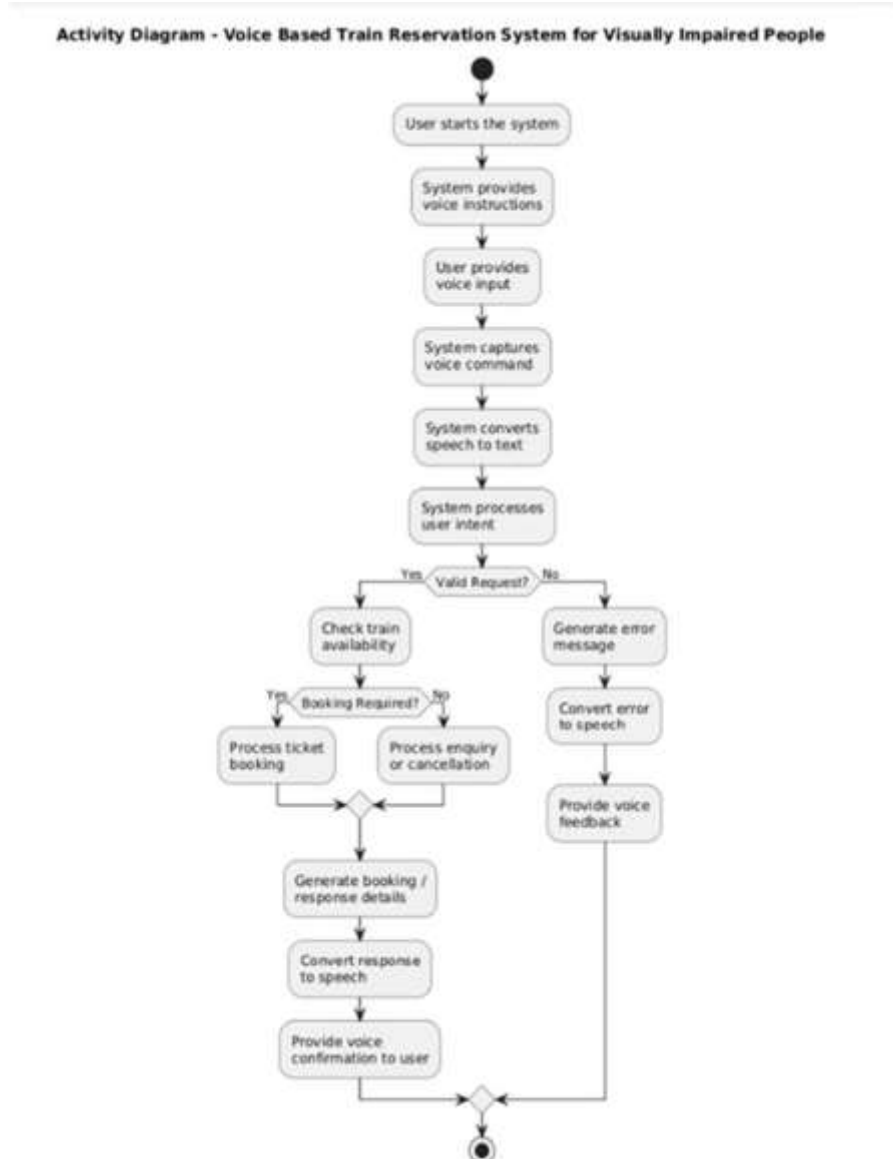


Figure 3: Activity diagram for a session in the voice-based rail reservation sys

LIMITATIONS

A candid account of what the current system cannot do well is more useful to future developers than a list of its strengths. Five limitations stand out.

Recognition accuracy in demanding acoustic conditions. The system was designed and initially tested in relatively quiet environments. At a busy station—with platform announcement loudspeakers, crowds, and diesel locomotives—transcription accuracy drops in ways that are hard to fully compensate for through noise suppression alone. Long station names with limited phonetic contrast to similar names (Secunderabad and Secundrabad, for example) are particularly vulnerable. The practical effect is that users in the noisiest environments are precisely those who most need the system to work, yet find it least reliable.

Single-language operation. The system currently operates in English. The majority of India's visually impaired population is more comfortable in Hindi, Tamil, Telugu, Kannada, Marathi, or Bengali. Designing for English-only access means the system is, in practice, only accessible to a specific demographic slice of its intended beneficiaries.

Dependency on network connectivity. Cloud-based ASR and live schedule queries both require a stable internet connection. Coverage gaps are common at many of the rural stations where the need for an accessible booking system is arguably greatest. On-device ASR models reduce this dependency but at a cost in accuracy.

Security of voice-based authentication. A spoken PIN or passphrase is easier to use than a typed password but is also more susceptible to recording-based replay attacks. Mitigations exist (liveness detection, one-time spoken phrases, device-bound authentication) but none of them is fully frictionless for a blind user in a public space.

The journey ends at the ticket. Once a booking is confirmed, the system's involvement stops. The user still needs to navigate to the correct platform, find their coach, and identify their berth—tasks that remain largely unsupported for visually impaired passengers. The reservation system solves one segment of what is actually an end-to-end accessibility problem

FUTURE SCOPE

Each limitation above suggests a corresponding line of future work.

- **Station-noise-adapted ASR:** Collecting labelled audio recordings specifically from Indian railway environments—with real ambient noise profiles, real station-name pronunciations across regional accents—would produce a far more robust acoustic model than anything currently available. This is a community-level resource problem as much as a technology problem.
- **Full Indian-language coverage:** Hindi, Tamil, Telugu, Kannada, Bengali, and Marathi together cover the overwhelming majority of the population. Adding full-pipeline support—ASR, NLP, and natural-sounding TTS—for these six languages would transform the system from a product for English-speaking blind users to a genuinely national service.
- **Offline-capable architecture:** Separating the functions that require live data (real-time availability) from those that do not (timetable lookup, session management, confirmation flow) would allow much of the interaction to proceed on-device. Prefetching and caching timetable data on a daily basis would cover most queries without a network call.
- **Wearable integration:** A smartwatch with a microphone and bone-conduction audio output would allow the booking interaction to take place without the user holding a phone to their ear in a crowded public space. This is both a convenience improvement and a safety improvement for users navigating busy concourses.
- **Station navigation extension:** Indoor positioning using Bluetooth Low Energy beacons—already piloted in a small number of Indian airports—could extend the system's reach from booking to boarding: "Your platform is Platform 6. Walk forward approximately 80 metres and turn left at the footbridge."
- **Live service information:** Integrating real-time train running data would allow the system to alert a waiting passenger proactively when their service is delayed, cancelled, or diverted to a different platform—without requiring them to find and navigate an information board.
- **Multi-turn conversational reasoning:** The current NLP layer handles one intent per turn. A large language model backbone would allow the system to carry conversational context across multiple turns: asking "What's the earliest I can leave for Pune?" and then "And what about the return journey on Sunday?" without resetting the context each time.
- **Stronger authentication design:** Voiceprint verification with randomized challenge phrases (to prevent replay) combined with a device-bound credential would deliver authentication security comparable to standard banking applications, without requiring any text input.

Conclusion

The starting point for this project was a specific, concrete problem: a person who cannot see should be able to buy a train ticket without asking someone else to do it for them. That problem turns out to require solving several interconnected technical challenges simultaneously, and it demands a design philosophy quite different from the one that produced the visual interfaces most people currently use.

The system described in this paper addresses the problem by replacing the visual interface with a conversational one. Its four-layer architecture separates the concerns of audio capture, language understanding, reservation logic, and data access into components that can each be improved independently. Its interaction model is built around two non-negotiable principles: the user is never left uncertain about what to say next, and the user is never committed to a financial transaction they did not explicitly approve. The limitations documented here are not incidental. They reflect the actual difficulty of the problem: making speech recognition reliable in genuinely noisy environments, extending language support across India's linguistic diversity, and ensuring that authentication by voice is as secure as authentication by text. Each of these is a tractable engineering and research challenge, not a fundamental barrier. The broader point is about what independent access to public transport means for a person with a visual impairment. It is not a quality-of-life improvement on top of an otherwise fully functional situation. It is, for many people, a precondition for employment, education, and social participation. A voice-based reservation system that functions reliably enough to be trusted is a contribution to something more important than a usability metric—it is a contribution to a more navigable world. Future scope Collecting labelled audio recordings specifically from Indian railway environments. Full Indian-language coverage: Hindi, Tamil, Telugu, Kannada, Bengali, and Marathi together cover the overwhelming majority of the population. Separating the functions that require live data (real-time availability) from those that do not (timetable lookup, session management, confirmation flow) would allow much of the interaction to proceed on-device. A smartwatch with a microphone and bone-conduction audio output would allow the booking interaction to take place

without the user holding a phone to their ear.

The image shows two web interfaces. The top interface is ClarityCheck's 'Plagiarism Results' page. It displays an analyzed text snippet about a hands-free railway booking system. The results show 1% plagiarism, with 0% exact match, 0% partial match, and 99% unique content. Two sources are listed: biostealth.ai (57% match) and medium.com (50% match). The bottom interface is QuillBot's AI Detector. It shows the same text snippet and reports that 3% of the text is likely AI-generated. The breakdown shows 3% AI-generated, 0% human-written & AI-refined, and 97% human-written. It also lists 'Main AI Contributors' as 'AI-generated'.

ClarityCheck Plagiarism Results

Analyzed Text: 830 Words, 11707 Characters

Hands-Free Railway Booking: Designing an Audio-Only Reservation Interface for Blind and Partially Sighted Users ABSTRACT Buying a train ticket is one of those tasks most people do without a second thought, yet for a person who cannot see, the standard online journey—scanning timetables, clicking seat maps, reading confirmation screens—can be genuinely impossible to complete alone. This paper documents the design of a fully audio-based reservation system intended to close that gap. Instead of adding accessibility features to an existing visual interface, the system was built from the ground up with voice as its only interaction channel: users speak to it, and it speaks back. No screen interaction is required at any stage. The technical foundation consists of three well-established components—automatic speech recognition to capture what the user says, a language understanding layer to work out what they mean, and a speech synthesizer to deliver responses—connected to a standard railway booking back end. The harder engineering challenge is not assembling these parts but making the assembled system behave consistently in the kinds of imperfect real-world conditions that a

Plagiarism Results

1% Plagiarism

- Exact match: 0%
- Partial match: 0%
- Unique content: 99%

Sources (3)

- biostealth.ai: 57% (1/1) The architecture section describes how the system is layered.
- medium.com: 50% (1/1) Error handling is also the responsibility of this layer.

Check Another Text

QuillBot AI Detector

English

ABSTRACT

Buying a train ticket is one of those tasks most people do without a second thought, yet for a person who cannot see, the standard online journey—scanning timetables, clicking seat maps, reading confirmation screens—can be genuinely impossible to complete alone. This paper documents the design of a fully audio-based reservation system intended to close that gap. Instead of adding accessibility features to an existing visual interface, the system was built from the ground up with voice as its only interaction channel: users speak to it, and it speaks back. No screen interaction is required at any stage. The technical foundation consists of three well-established components—automatic speech recognition to capture what the user says, a language understanding layer to work out what they mean, and a speech synthesizer to deliver responses—connected to a standard railway booking back end. The harder engineering challenge is not assembling these parts but making the assembled system behave consistently in the kinds of imperfect real-world conditions that a blind traveler actually encounters: background noise, regional pronunciation, patchy mobile coverage, and the occasional wrong word. This paper documents the design decisions made in response to those challenges, describes how the system is structured, and sets out clearly what it cannot yet do.

1,132 Words

Detect AI again

3% of text is likely AI

AI-generated: 3%
Human-written & AI-refined: 0%
Human-written: 97%

Main AI Contributors

- AI-generated: 1
- AI-generated: 1

Want your text to sound more authentic? Refine with Paraphraser