WWW.IJRPR.COM

International Journal of Research Publication and Reviews

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

AI-Powered Railway Accessibility Assistant for Differently Abled Passengers

Sushmita Nishandar[1], Preethi C[2], Roopashree M O[3], Dr. Santosh Kumar M[4]

[1-3] Department of Computer Science and Engineering, GM Institute of Technology, Davangere, Karnataka, India

[4]*Under the Guidance, Assistant Professor, Department of Computer Science and Engineering*

DOI : https://doi.org/10.55248/gengpi.07.1026.34044

A B S T R A C T:

Railway travel can be challenging for differently abled passengers when train information, accessibility settings, location assistance and emergency communication are spread across separate services. This paper presents RailSathi, an AI-powered web-based railway accessibility assistant that brings these functions into a common platform. The system provides secure registration and login with password hashing and optional time-based one-time password (TOTP) verification, adjustable text size, contrast, spacing and cursor settings, browser text-to-speech, multilingual interface support, an AI assistant using a Groq-hosted language model, and train information through the RailRadar API with a local fallback dataset. Location-based assistance uses browser geolocation, Leaflet and OpenStreetMap-based data to show nearby railway stations and hospitals. WebRTC and Socket.IO support browser-based voice/video communication with authorized emergency contacts, with a telephone fallback. Functional validation covered authentication, accessibility settings, live and fallback train-information paths, emergency calling between browser accounts, contact authorization and nearby assistance. The prototype demonstrates an integrated accessibility-oriented workflow; formal user studies and numerical performance benchmarks remain future work.

Keywords: Railway Accessibility, Assistive Technology, Artificial Intelligence, WebRTC, Emergency Communication, Multilingual Interface, RailRadar API, Inclusive Mobility.

1. INTRODUCTION

Railway transportation is an important part of public mobility, but passengers with visual, hearing, mobility or other accessibility requirements may need larger text, spoken information, simplified interaction, nearby-facility guidance or direct assistance. When these services are distributed across different applications, users must repeatedly change interfaces, which can increase effort during a journey. Existing railway services commonly focus on schedules, ticketing and running status, while accessibility and emergency workflows may require separate tools.

RailSathi addresses this fragmentation by combining railway information, conversational assistance, adaptable interface controls, multilingual interaction, location-based support and emergency communication in one web platform. The system is designed for use through a browser-enabled phone or computer and includes limited fallback paths for selected external-service failures. It is a prototype and is not intended to replace official railway information or emergency services.

2. LITERATURE SURVEY

Accessible digital services are commonly guided by the Web Content Accessibility Guidelines (WCAG), which emphasize perceivable, operable, understandable and robust interfaces. Human-computer interaction research also highlights the importance of reducing unnecessary navigation and making system feedback clear. WebRTC enables real-time audio/video communication between browsers, while location services can connect a user's position with nearby facilities. Mapping libraries such as Leaflet and open geographic data can support location-aware interfaces. Railway applications already provide schedules and train status, but RailSathi's emphasis is the integration of these capabilities with accessibility preferences and emergency contact workflows.

The identified gap is not the absence of individual technologies, but the lack of a single workflow that combines accessible presentation, railway data, AI-based guidance, nearby assistance and authorized communication. RailSathi uses modular components so that railway data, conversational assistance, mapping and communication can operate as separate services within one interface.

3. METHODOLOGY

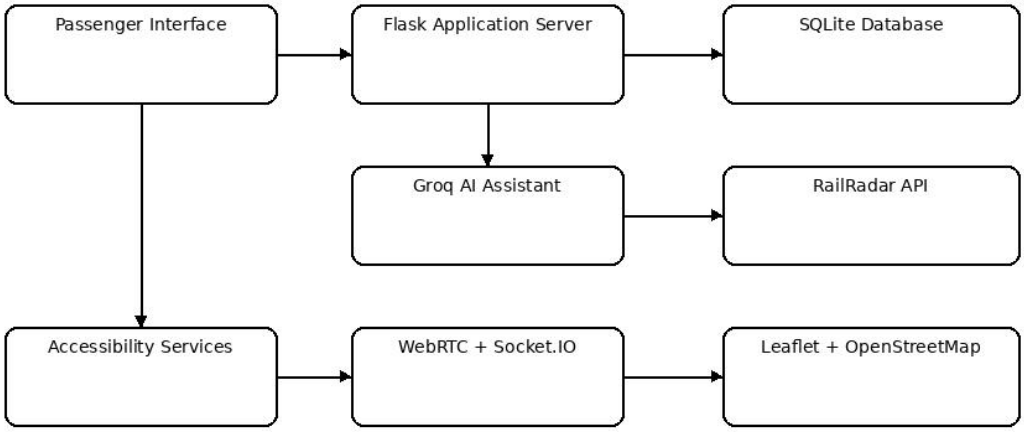
RailSathi follows an accessibility-first workflow. The passenger registers or logs in, configures interface preferences and then accesses railway information, the AI assistant, nearby assistance, language settings and emergency communication from a shared dashboard. The frontend is implemented using HTML, CSS and JavaScript; Python Flask handles backend requests and SQLite provides local persistence.

Authentication stores passwords using hashing rather than plain text. TOTP-based two-factor authentication provides an additional verification step when enabled. The accessibility module offers text-size, contrast, spacing and cursor adjustments, as well as browser-based text-to-speech. Interface language options include English, Hindi, Kannada, Telugu, Tamil and Marathi.

For railway queries, the backend attempts to retrieve data through the RailRadar API when configured. A local fallback dataset supports demonstration of the workflow when live data is unavailable. The AI assistant forwards supported questions to a Groq-hosted language model and uses predefined rule-based responses for selected common queries if the service is unavailable. Generated AI responses are treated as assistance, not as an authoritative source of live train status.

The location module requests browser geolocation permission and displays nearby railway stations and hospitals using Leaflet and OpenStreetMap-based data. For emergency communication, WebRTC provides browser media connections while Socket.IO and Flask-SocketIO coordinate signaling. The system checks the configured emergency-contact workflow and provides a telephone fallback when browser-based communication cannot be used.

Fig. 1. Overall architecture of the proposed RailSathi system.



The main service flow is summarized in Table 1.

Table 1. Main modules and implementation technologies

Module	**Technology / Service**	**Purpose**
Web interface	HTML, CSS, JavaScript	Common accessible dashboard
Backend and storage	Python Flask, SQLite	Request handling and local persistence
Authentication	Password hashing, TOTP	Account protection and extra verification
AI assistance	Groq-hosted language model	Conversational railway guidance
Train information	RailRadar API, local fallback	Retrieve or demonstrate train data
Location assistance	Geolocation, Leaflet, OpenStreetMap	Nearby station and hospital map
Emergency communication	WebRTC, Socket.IO	Browser voice/video signaling

4. SYSTEM DESIGN AND WORKFLOW

The architecture separates user-facing functions from external services. Passenger requests are handled by the Flask application, which uses SQLite for local data and routes railway queries to RailRadar or the local fallback dataset. AI requests are sent to the configured language-model service, while location and emergency features use their dedicated browser and communication interfaces. This separation helps keep AI-generated guidance distinct from live railway information.

The principal use cases include account registration and login, accessibility preference management, train-information requests, AI-assisted questions, language selection, nearby-facility search, emergency contact management and voice/video calls. Selected fallback responses and the local train dataset allow core demonstrations to continue when external services are unavailable, but they do not remove the need for network connectivity in real deployment.

5. RESULTS AND DISCUSSION

The prototype was evaluated using functional scenarios rather than a statistical machine-learning benchmark. The project validation covered registration and login, TOTP verification when enabled, accessibility settings, live and fallback train-information paths, emergency calling between two browser accounts, emergency-contact authorization and location-based assistance. These checks provide functional evidence that the major modules are integrated in the application.

Table 2. Functional validation summary

Test area	Observed validation	Status
Authentication	Registration/login and configured TOTP flow checked	Functionally tested
Accessibility	Text size, contrast, spacing, cursor and Read Page options checked	Functionally tested
Railway information	Live API path and local fallback workflow checked	Functionally tested
Emergency call	Voice/video call between two browser accounts checked	Functionally tested
Contact authorization	Emergency contact workflow checked	Functionally tested
Nearby assistance	Location permission and station/hospital map checked	Functionally tested

No numerical accuracy, response-time or usability score is claimed because the project material does not include a controlled measurement dataset or a standardized accessibility audit.

5.1 ACCESSIBILITY AND AI RESPONSE ANALYSIS

The accessibility dashboard provides controls intended to make information easier to perceive and navigate. Browser-based text-to-speech offers an additional way to consume page content, while language selection is intended to reduce language-related barriers. These functions were checked as application features; their effectiveness for different disability groups requires evaluation with representative users.

The AI assistant provides a natural-language entry point for railway-related questions and platform guidance. A rule-based fallback supports selected common queries when the external AI service is unavailable. However, the assistant is not a substitute for official train status or emergency response. Live railway information should come from the configured data service, and urgent cases should use the emergency communication workflow.

5.2 EMERGENCY AND LOCATION ASSISTANCE

The emergency module combines authorized contacts with browser-based voice/video communication. WebRTC handles media connections, while Socket.IO coordinates signaling. A telephone fallback provides another route when browser communication is unsuitable. The location module uses browser geolocation permission and an interactive map to show nearby railway stations and hospitals. Location and communication services depend on browser permissions, network conditions and external service availability.

5.3 CHALLENGES AND LIMITATIONS

Table 3. Identified challenges and current mitigation

Challenge	Current approach
External API or AI service unavailable	Local train dataset and limited rule-based AI responses
Browser media or location permission denied	Provide clear permission-dependent workflow and telephone fallback for calls
Network interruption	Fallback supports selected demonstrations but does not guarantee live service
Accessibility effectiveness not measured	Future testing with users and a structured accessibility audit

6. CONCLUSION

RailSathi demonstrates an accessibility-oriented railway assistance platform that integrates AI-based conversation, train information, adaptive interface controls, multilingual interaction, location-based assistance and emergency communication within one web application. Functional validation covered the principal workflows and demonstrated how the services can be brought together without requiring passengers to navigate separate applications for every task.

The current system is a research prototype. Future work should include usability studies with differently abled passengers, standardized accessibility audits, broader device and network-failure testing, stronger security assessment and verified station-level accessibility data such as lifts, ramps and assistance desks. Further deployment would require careful attention to privacy, consent and the reliability of external railway services.

REFERENCES

[1] World Wide Web Consortium, “Web Content Accessibility Guidelines (WCAG) 2.2,” W3C Recommendation, 2023.

[2] A. Cooper, R. Reimann, D. Cronin, and C. Noessel, About Face: The Essentials of Interaction Design, 4th ed. Wiley, 2014.

[3] B. Shneiderman et al., Designing the User Interface: Strategies for Effective Human-Computer Interaction, 6th ed. Pearson, 2016.

[4] A. J. Dix, J. Finlay, G. D. Abowd, and R. Beale, Human-Computer Interaction, 3rd ed. Pearson, 2004.

[5] J. Rosenberg et al., “WebRTC 1.0: Real-Time Communication Between Browsers,” W3C Recommendation, 2021.

[6] M. Jones, B. Bradley, and N. Sakimura, “JSON Web Token (JWT),” RFC 7519, Internet Engineering Task Force, 2015.

[7] M. M’Raihi et al., “TOTP: Time-Based One-Time Password Algorithm,” RFC 6238, IETF, 2011.

[8] M. Haklay and P. Weber, “OpenStreetMap: User-Generated Street Maps,” IEEE Pervasive Computing, vol. 7, no. 4, pp. 12–18, 2008.

[9] OpenStreetMap Foundation, “OpenStreetMap Data and Documentation,” project documentation.

[10] V. Agafonkin, “Leaflet: An Open-Source JavaScript Library for Interactive Maps,” project documentation.

[11] Pallets Projects, “Flask Documentation,” Python web application framework documentation.

[12] SQLite Consortium, “SQLite Documentation,” embedded SQL database documentation.

[13] WebRTC Working Group, “WebRTC API and Real-Time Communication,” W3C specifications and documentation.

[14] Groq, “Groq API Documentation,” developer documentation.

[15] RailRadar, “Railway Data API,” project/service documentation.

[16] Socket.IO, “Socket.IO Documentation,” real-time bidirectional event-based communication documentation.

[17] United Nations, “Convention on the Rights of Persons with Disabilities,” United Nations Treaty Series, 2006.

[18] United Nations, “Transforming Our World: The 2030 Agenda for Sustainable Development,” A/RES/70/1, 2015.