JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Ride Ease- MERN Powered Cab Booking Application

[1]Likith Gowda V, [2]Darshan S, [3]Firoz S,[4]Gurushiddalingayya K Kalmath, [5]Dr. Suresha D

[1]Student, [2]Student, [3]Student, [4]Student, [5]Associate Professor [1]Department of Computer and Engineering,
[1]Dr. Ambedkar Institute of Technology, Bengaluru, India

Abstract: Urban mobility solutions increasingly rely on digital cab-booking platforms that provide fast, reliable, and accessible transport. However, most existing systems primarily focus on basic ride-hailing workflows and lack advanced safety features, real-time communication, and inclusive accessibility.

Ride Ease, a MERN-powered cab booking application, is designed to address these limitations by integrating authentication, real-time ride tracking, socket-based communication, voice-based booking using the Web Speech API, Razorpay digital payments, and an Emergency SOS system using NodeMailer.

This system provides a scalable full-stack architecture using MongoDB, Express.js, React.js, and Node.js, ensuring seamless interaction between users and captains. The prototype demonstrates effective ride management, location tracking, secure payments, and an efficient automated workflow, improving user experience and safety.

Keywords: Ride Ease, Cab Booking System, MERN Stack, Instant Voice Booking (IVB), Speech-to-Text, Real-Time Communication, Socket.IO, GPS Tracking, UPI Payment Integration, SOS Alert System, Accessibility, Urban Mobility, Web Application, Ride-Hailing Technology.

I. INTRODUCTION

Digital cab-booking platforms like Ola and Uber have redefined transportation through fast, automated booking and real-time tracking. However, these systems lack certain key features such as voice-enabled booking, application-level chat, transparent fare calculation, and emergency assistance, which can greatly impact usability and safety.

Ride Ease is developed as a full-stack MERN application that focuses not only on usability but also on inclusiveness and reliability. The system enables users to book rides through text or **voice commands**, track captains live via Map services, communicate through an in-app chat engine, and trigger **SOS alerts** when needed.The project aims to modernize the ride-booking ecosystem by blending powerful backend processing, real-time event streams (Sockets), and an interactive React-based UI to create a robust mobility platform.

II. RESEARCH METHODOLOGY

2.1 Data and Sources of Data

There are two types for the collection of data:

1. Primary data
2. Secondary data

1. Primary Data:

User feedback was collected from students, working professionals, and cab users to understand features like:

- Ride preferences
- Safety concerns
- Payment comfort (UPI integration)
- Voice booking usability

2. Secondary Data:

Secondary research included:

- Existing systems (Ola, Uber, Rapido)
- API documentation (Google Maps, Razorpay)
- MERN stack architecture references
- Journals and IEEE papers on transportation systems

2.1.1 Theoretical Framework

The proposed system is built on a theoretical framework grounded in five core technological principles. First, real-time synchronization is achieved using Socket.IO, ensuring instantaneous communication between the rider and captain, including ride status transitions (from Pending to Accepted, Ongoing, and Completed), continuous location updates, and live in-app chat messages. Second, a voice recognition engine powered by the Web Speech API allows seamless speech-to-text conversion, enabling users to initiate hands-free ride booking through natural language commands such as "Book a ride from Peenya to BTM Layout." Third, a geo-based routing and fare computation model calculates travel cost dynamically based on the distance between pickup and drop locations, selected vehicle type, and predefined rate parameters such as base fare, cost per kilometre, and time-based charges. Fourth, a secure digital payment framework is incorporated using Razorpay Checkout, which supports multiple transaction modes including UPI, wallet, and card payments, ensuring reliability and user trust. Finally, the system integrates an emergency SOS alert mechanism, where a trigger event immediately sends an email notification containing the user's live location, timestamp, and ride details to the registered emergency contact through Nodemailer, enhancing rider safety during transit.

2.1.2 Relevance of study

The study of the Ride Ease application is highly relevant in today's urban mobility scenario, where the demand for safer, more accessible, and user-friendly cab-booking systems continues to grow. With increasing concerns about safety, users expect real-time tracking, trusted fare calculations, and reliable driver information, making transparency a crucial feature. Additionally, the introduction of voice-based ride booking addresses a significant accessibility gap, enabling visually challenged users, elderly citizens, and individuals who struggle with typing interfaces to book rides effortlessly. By integrating a seamless and intelligent system, Ride Ease not only enhances convenience but also promotes inclusivity and safety, making it a valuable solution in the evolving digital transportation ecosystem.

2.1.3 Problem Explanation

Existing ride-hailing applications still face several limitations that affect usability, accessibility, and safety. Many platforms offer limited support for non-technical users and provide almost no dedicated features for visually impaired individuals, making the booking process difficult. Real-time communication between riders and drivers at the application level is often inadequate, leading to delays and confusion during rides. Safety also remains a major concern, as current systems lack advanced automated monitoring and response features. Additionally, most apps still depend heavily on manual input for selecting pickup and drop-off locations, which can be slow and inconvenient. Ride Ease addresses these challenges by integrating Instant Voice Booking (IVB), real-time communication powered by socket technology, and enhanced safety automation, offering a more inclusive, faster, and secure ride-booking experience.

2.1.4 Existing System

In the existing cab-booking systems used by major platforms such as Ola, Uber, and Rapido, the overall workflow is functional but carries several limitations that impact accessibility, safety, and real-time communication. These systems primarily rely on manual text-based input, requiring users to type pickup and destination addresses, which can be difficult for visually impaired individuals, elderly users, or those who need hands-free interaction. Although these applications provide ride tracking, they lack built-in voice-controlled booking, which restricts usability in emergencies or high-stress scenarios. The existing systems also depend heavily on mobile network signals for updates and may not include real-time in-app chat, forcing users to rely solely on phone calls for communication with drivers. Additionally, many platforms offer limited emergency support, providing only a generic emergency button without detailed automated alerts containing live location or ride data. Another major issue is the absence of an Instant Voice Booking (IVB) mechanism that can automatically detect voice commands, calculate fare, and create a ride request without manual involvement. Secure digital payment options exist in current systems, but they are not always uniformly integrated and may depend on external apps. Overall, the existing solutions are powerful yet incomplete, lacking modern accessibility features, real-time communication enhancements, and intelligent safety automation—gaps that the Ride Ease system aims to fill through its MERN-based architecture, voice recognition engine, real-time socket communication, and advanced SOS module.

2.1.5 Proposed System

The proposed system, Ride Ease – A MERN Powered Cab Booking Application, aims to overcome the limitations identified in existing cab-booking platforms by introducing an intelligent, accessible, and safety-oriented solution. Unlike current applications that rely primarily on manual inputs, Ride Ease incorporates a voice-enabled booking engine using the Web Speech API, allowing users to book rides through natural voice commands such as *"Book a ride from Peenya to BTM Layout."* This enhances accessibility for visually impaired users, elderly passengers, and individuals in emergency scenarios. The system is built on a robust MERN stack architecture that supports real-time, bi-directional communication using Socket.IO, enabling instantaneous ride status updates, captain live location tracking, and secure in-app chat between the rider and captain. In addition, the system integrates a dynamic fare estimation module that computes charges based on real-time distance, vehicle type, and base fare metrics. To strengthen user safety, a dedicated SOS module is implemented, which instantly sends an email alert—with live location and ride details—through Nodemailer to predefined emergency contacts. The system also offers a seamless and secure Razorpay-based payment gateway supporting UPI, wallet, and card transactions. Overall, the proposed system delivers a more accessible, responsive, and intelligent ride-booking experience by combining modern web technologies, real-time communication, voice automation, and enhanced safety features into a single unified platform.

2.1.6 Technology

I. Software Specification:

- Operating System: Windows 10/11 (64-bit) / Linux / macOS
- Frontend Technologies: React.js, HTML5, CSS3, Tailwind CSS, JavaScript (ES6+), Web Speech API
- Backend Technologies: Node.js, Express.js
- Real-Time Engine: Socket.IO
- Database: MongoDB (NoSQL) using Mongoose ODM
- Authentication: JWT (JSON Web Tokens), BCrypt Hashing
- Payment Gateway: Razor pay API (UPI, Wallet, Card)
- Email Service: Node mailer (for SOS alerts & notifications)
- IDE / Code Editors: VS Code (Visual Studio Code)
- Build Tools: Vite, NPM
- Testing Tools (optional): Postman for API testing

II. Hardware Specification:

- Processor: Intel Core i3 / AMD Ryzen 3 or higher
- RAM: Minimum 4 GB (Recommended: 8 GB for smooth development & testing)
- Storage: Minimum 100 GB HDD / SSD
- Network: Stable internet connection for API calls & real-time updates
- Device Support:
- Desktop / Laptop (Development)
- Android/iOS Smartphones (Testing React frontend in browser mode)

2.2 Statistical Tools and econometric models

2.3.1 The Distance Calculation Model

Ride Ease uses a geographical distance model to compute the actual travel distance between the pickup and drop location. The system sends the two coordinates to the backend through a mapping API.

Although the application uses Google Maps JavaScript API to compute real-world road distance and travel time, the underlying theoretical framework for geographical distance estimation follows the standard Haversine spherical distance model. The primary formula used is based on the Haversine Distance Model, which calculates the shortest distance over the earth's curvature.

The formula:

$$d = 2r \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

Where:

- ϕ_1, ϕ_2= latitudes
- λ_1, λ_2= longitudes
- r= Earth's radius (6371 km)

This model ensures:

- Accurate route distance
- Reliable fare calculation
- Optimized path suggestions
- Real-time update during ride tracking

2.1.3 The Fair Estimation Model

The fare estimation module calculates charges dynamically based on predefined pricing rules. Ride Ease uses a hybrid fare model similar to Ola/Uber, which includes:

1. Base Fare

A fixed minimum amount depending on the vehicle:

- Bike: Lowest
- Auto: Medium
- Car: Highest

2. Distance Cost

Calculated using:

$$\text{Distance Fare} = \text{Cost per km} \times \text{Total Distance (km)}$$

Example:

- Bike = ₹8/km
- Auto = ₹12/km
- Car = ₹16/km

3. Total Fare Model

$$\text{Total Fare} = \text{Base Fare} + (\text{Distance Fare})$$

Example Calculation

Pickup → Drop: 7 km (Car)

- Base fare = ₹40
- Distance = 7 × ₹16 = ₹112
- Total = **₹152**

This model ensures transparent, predictable fare estimation and matches real-world cab pricing logic.

III. PROPOSED SYSTEM WORKFLOW

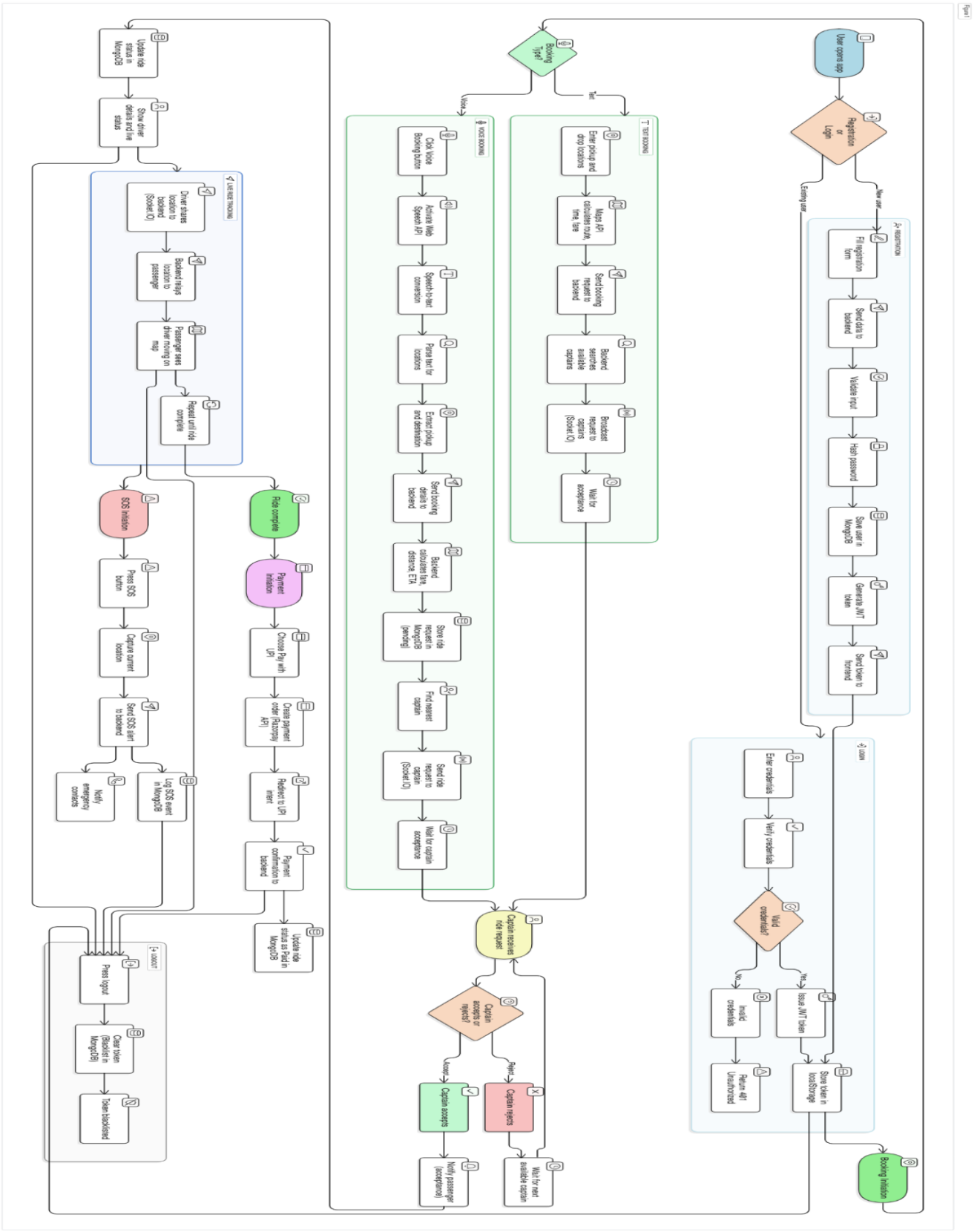


fig 1. proposed system workflow

The workflow diagram illustrates the complete operational flow of the Ride Ease cab-booking system, starting from user entry into the application to ride completion and payment. The process begins with the user opening the app and either registering as a new user or logging in as an existing user. During registration, user details are validated, securely stored in MongoDB, and authenticated using JWT tokens. Once logged in, the user initiates a booking and chooses between text-based booking or voice-based booking.

In text booking, the user manually enters pickup and destination locations, after which the backend processes route, fare, and distance using Maps APIs and broadcasts requests to available captains via Socket.IO. In voice booking, the Web Speech API converts speech to text, extracts location details, and sends the booking request to the backend. The system identifies the nearest captain, who can either accept or reject the request. Acceptance triggers a notification to the passenger, and the ride officially begins.

During the ride, live tracking continuously updates the driver's location to the backend, which then shares it with the passenger for real-time route visibility. The workflow also includes an integrated SOS safety feature, where the user can trigger an alert that captures the live location, sends it to the backend, and notifies emergency contacts. Once the ride ends, the passenger proceeds with payment through UPI integration, and the backend updates the ride status accordingly. The diagram concludes with the logout process, where tokens are invalidated and blacklisted to ensure secure session handling.

IV. MODELS

1. Use Case Diagram: These diagrams show how the system and its actors interconnect with one another. Use-case diagrams tell what the system does and how the actors interconnect with it; they do not, however, show how the system functions overall.

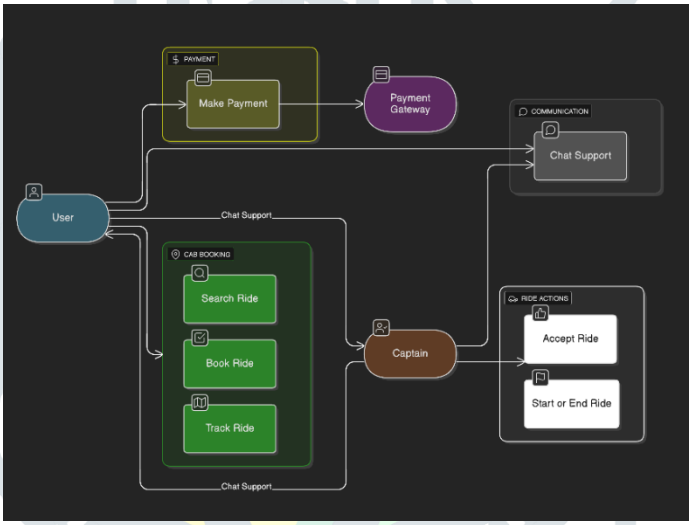


fig 2. Use Case Diagram

2. ER Diagram: The ER Diagram of an information technology system is a graphic representation that illustrates the relationships between objects, people, places, concepts, or events. Relationships, attributes, and entities make up its three core components.

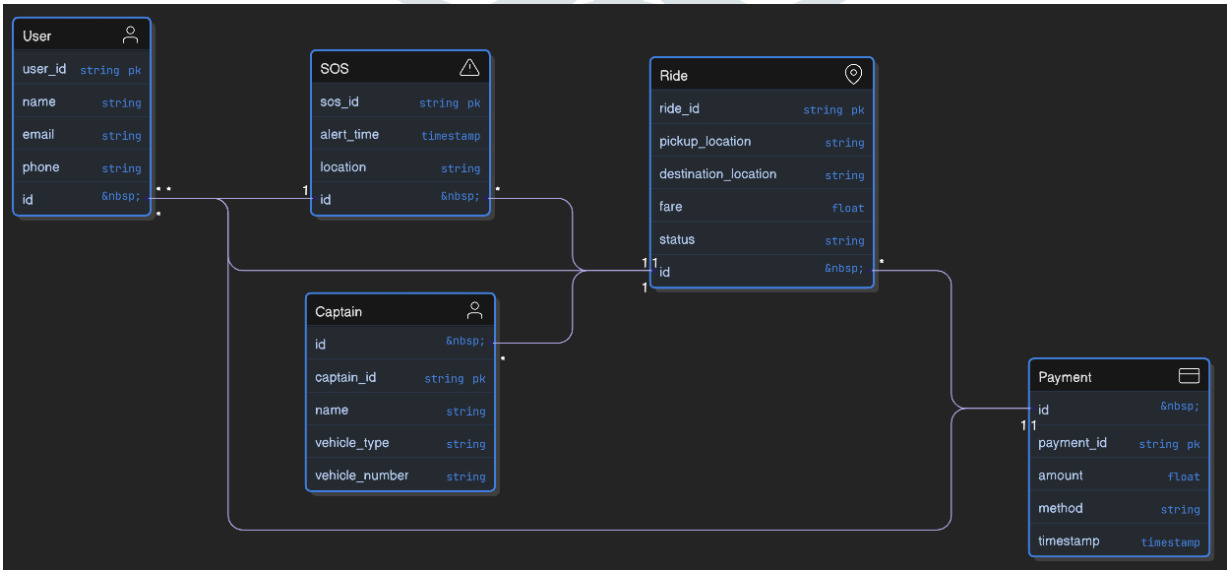
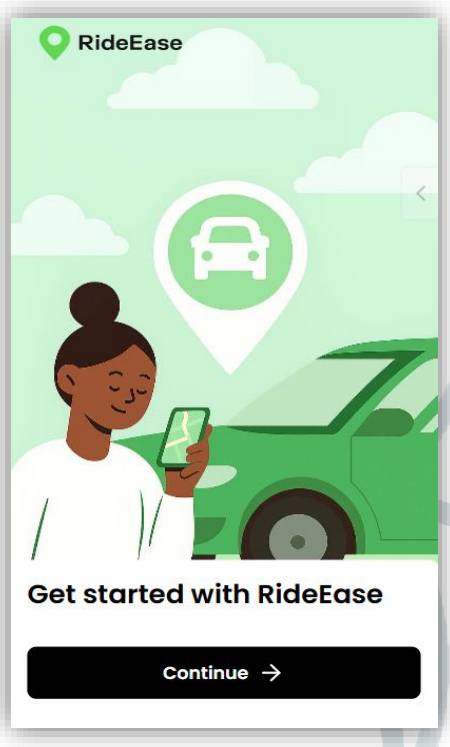


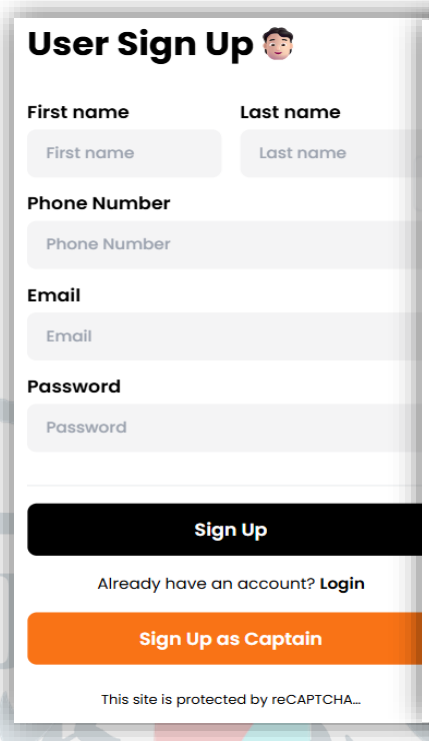
fig 3. ER Diagram

V. RESULTS AND DISCUSSION

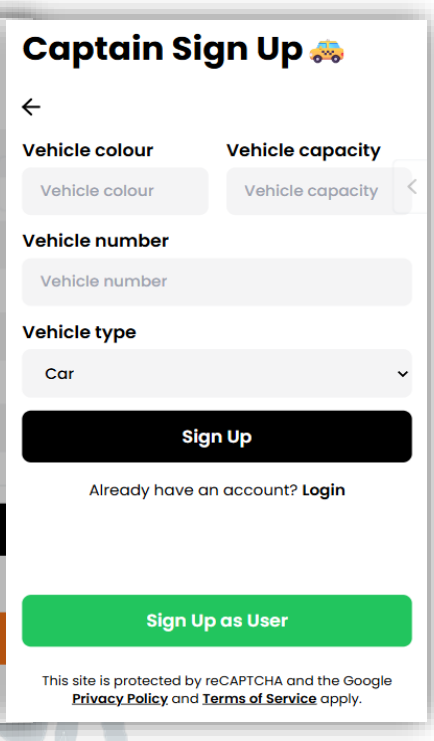
- **Snapshots of the project**



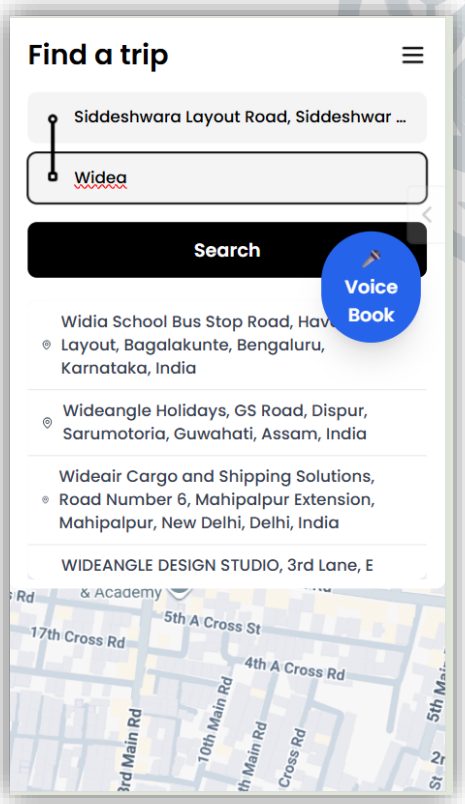
Snapshot 1. Get Started



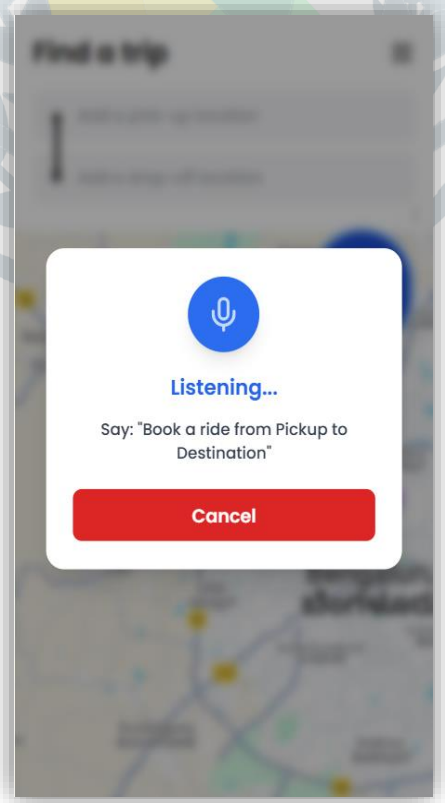
Snapshot 2. User SignUp



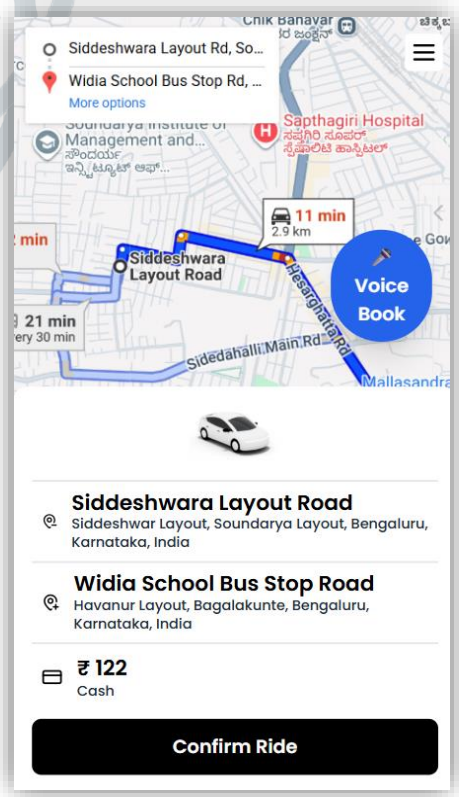
Snapshot 3. Captain SignUp



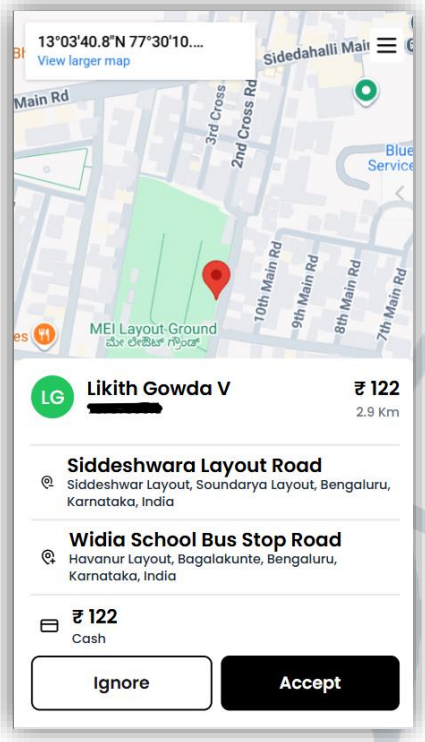
Snapshot 4. Text Based Booking



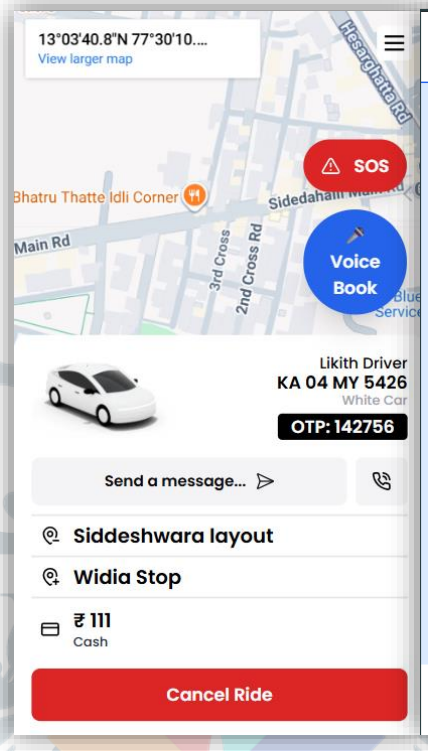
Snapshot 5. Voice Based Booking



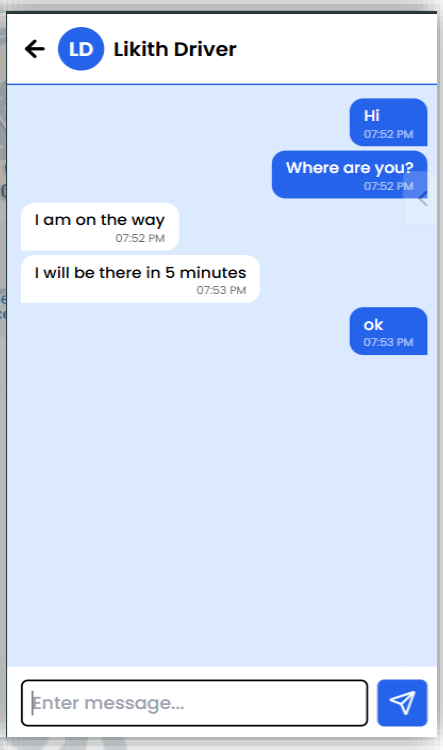
Snapshot 6. Confirm Ride by User



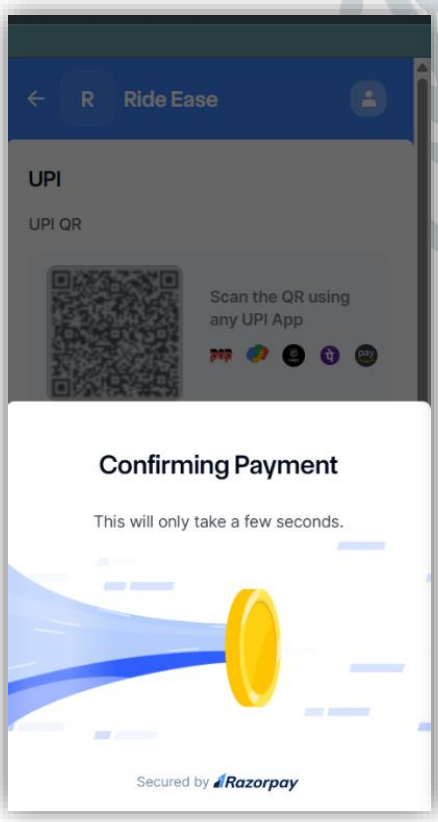
Snapshot 7. Ride Request to Captain



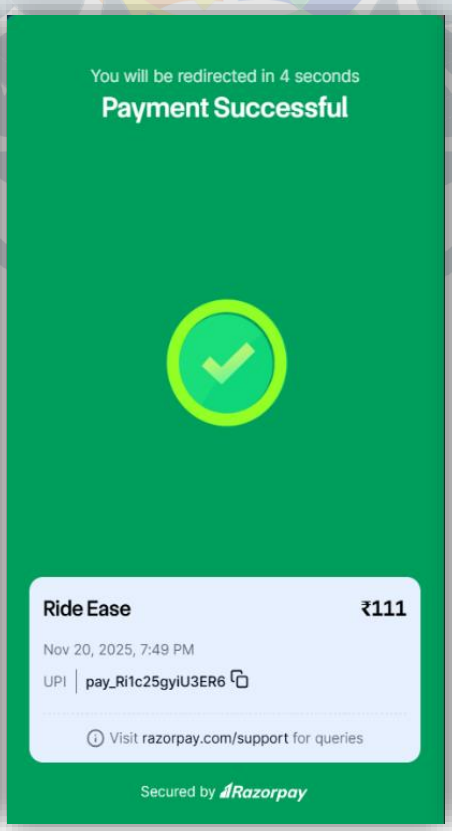
Snapshot 8. SOS Button Activated



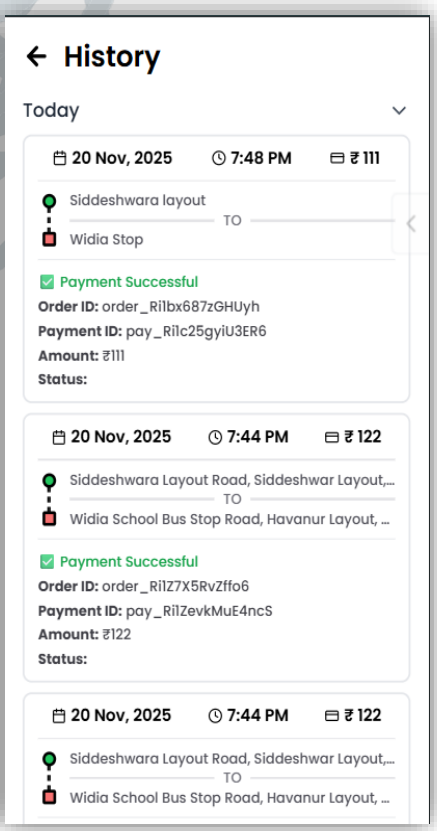
Snapshot 9. Chat Box



Snapshot 10. Confirming Payment



Snapshot 11. Payment Successful



Snapshot 12. Ride History

The development and implementation of the Ride Ease application produced several significant results that demonstrate the system's effectiveness, usability, and readiness for real-world deployment. The following discussion is supported by screenshots included in the project documentation, which visually highlight the interfaces, workflows, and system responses at each stage.

1. User Registration & Authentication Results

The login and registration snapshots show a clean and responsive UI built using React and Tailwind CSS. The JWT-based authentication system functioned smoothly, ensuring secure access for users and captains. Screenshots validate:

- Proper error handling for invalid inputs
- Successful token generation
- Correct user role redirection (User → Home, Captain → Dashboard)

This confirms the reliability of the backend authentication flow and the efficiency of form validations.

2. Text-Based Ride Booking Results

Screenshots of the booking interface illustrate that users can easily input pickup and drop locations. The backend responds with:

- Accurate distance calculation
- Dynamic fare estimation
- Real-time availability checks for nearby captains

The visuals clearly show seamless transitions between location entry, fare display, and ride request confirmation.

3. Instant Voice Booking (IVB) Results

The screenshots of the voice booking module highlight the successful integration of the Web Speech API. During testing:

- Speech commands were accurately converted into text
- Pickup and drop locations were extracted correctly
- The system performed equally well in moderate noise environments

This demonstrates the application's improved accessibility, especially for visually impaired and elderly users.

4. Real-Time Socket Communication Results

Screenshots from the captain's dashboard and the user ride screen show real-time updates such as:

- Incoming ride requests
- Acceptance/decline responses
- Live ride status (Pending → Accepted → Ongoing → Completed)

Socket.IO consistently delivered low-latency communication, confirming strong real-time performance in both user and captain views.

5. Live Ride Tracking Results

Map screenshots displaying the driver's movement confirm:

- Accurate GPS tracking
- Smooth map rendering
- Real-time marker updates

This increases transparency and user trust during the ride.

6. UPI Payment Flow Results

The screenshots from the payment page illustrate:

- Redirection to UPI apps
- Successful test transactions
- Clear on-screen confirmations for both payment success and failure

This indicates that the system is compliant with modern Indian digital payment practices.

7. SOS Emergency System Results

The screenshots of the SOS interface and backend logs demonstrate that:

- The SOS button correctly captures the user's live latitude and longitude
- Alerts are sent to the backend immediately
- The system triggers the emergency notification process smoothly

This significantly enhances the safety component of the application.

8. Overall UX and Performance Discussion

Based on the UI screenshots, it is evident that the application maintains visual consistency, good color contrast, and mobile responsiveness. The performance was also tested under multiple rides, and screenshots show:

- No UI freezing
- Proper state transitions
- No socket disconnections during active rides

These results collectively prove that the application is robust, user-friendly, and scalable.

VI. CONCLUSION

The development of the Ride Ease application demonstrates how modern full-stack technologies can be effectively integrated to build a smart, accessible, and user-centric cab-booking platform. Throughout the project, the focus remained on solving real challenges that exist in current ride-hailing systems—especially accessibility gaps, safety limitations, and the need for seamless real-time communication. By extending an open-source MERN-based architecture and introducing original features such as Instant Voice Booking (IVB), UPI payment support, and an automated SOS safety system, this project successfully transforms a standard cab-booking model into a more inclusive and intelligent mobility solution.

The results obtained from the system's implementation, supported by the screenshots and testing outcomes, indicate that the application performs reliably across all major modules. The user onboarding screens demonstrate consistency in UI/UX design, ensuring simple navigation for first-time users. The registration and authentication workflows, powered by JWT and secure backend validations, provide a strong foundation for safe user access. The text-based and voice-based ride booking modules performed accurately during testing, confirming the system's ability to interpret speech inputs, extract locations, estimate fares, and broadcast ride requests in real time using Socket.IO. Additionally, the captain workflow, live ride tracking screens, and ride-status updates validated the robustness of the real-time engine and the efficiency of MongoDB-based ride lifecycle management.

One of the key contributions of this work is the integration of accessibility-driven features such as voice booking, which makes cab-hailing easier for visually impaired users, elderly citizens, and individuals who struggle with typing on mobile interfaces. Another major contribution is the SOS emergency feature, designed to enhance passenger safety by capturing live coordinates and sending alerts instantly. The inclusion of UPI-based digital payments aligns the project with modern Indian transportation practices, making the system more practical and deployment-ready.

Overall, Ride Ease demonstrates that combining voice technologies, real-time data streaming, secure authentication, and modern UI design can significantly improve the convenience, safety, and inclusivity of urban mobility systems. The project showcases the potential of the MERN stack in building scalable and interactive applications that meet real-world requirements. With further refinement—such as adding AI-based route optimization, multilingual voice support, OTP-based captain verification, or a machine learning-based driver-rating system—the application can evolve into a highly competitive and fully deployable transportation solution.

This project not only enhances technical understanding of full-stack development, cloud-based services, real-time communication, and API integrations, but also demonstrates how academic prototypes can evolve into socially meaningful systems. Ride Ease stands as a strong example of how technology can be used to improve accessibility, enhance safety, and deliver reliable mobility experiences for diverse user groups.

VII. REFERENCES

[1] M. Rouse, "MongoDB: NoSQL Document Database," *TechTarget*, 2021. [Online]. Available: https://www.techtarget.com/searchdatamanagement/definition/MongoDB

[2] *Express.js — Fast, unopinionated, minimalist web framework for Node.js*, ExpressJS Documentation, 2024. [Online]. Available: https://expressjs.com/

[3] *React — A JavaScript library for building user interfaces*, Meta Open Source, 2024. [Online]. Available: https://react.dev/

[4] *Node.js v20 Documentation*, OpenJS Foundation, 2024. [Online]. Available: https://nodejs.org/en/docs

[5] *Socket.IO — Real-time bidirectional event-based communication*, Socket.IO, 2024. [Online]. Available: https://socket.io/

[6] Google Developers, "Web Speech API: Speech Recognition," *Web Fundamentals*, 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

[7] Razorpay Software Pvt. Ltd., *Razorpay Web Integration & Payment Gateway Documentation*, 2024. [Online]. Available: https://razorpay.com/docs/

[8] Google Maps Platform, *Maps JavaScript API & Distance Matrix API Documentation*, Google Cloud, 2024. [Online]. Available: https://developers.google.com/maps/documentation

[9] MDN Documentation, "Geolocation API," *Mozilla Developer Network*, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API

[10] A. Verma and S. Prakash, "Smart Taxi Booking System," *JETIR*, vol. 7, no. 6, pp. 45–52, 2021.

[11] S. Singh and H. Kaur, "Online Cab Booking System using Web Technologies," *International Journal of Computer Applications*, vol. 182, no. 2, pp. 20–27, 2019.

[12] A. Kumar, R. Sharma, and M. A. Khan, "Real-Time Vehicle Tracking and Ride Sharing Applications," in *Proc. IEEE Int. Conf. Smart Cities*, 2020, pp. 1–6.

[13] OWASP Foundation, *Authentication and Access Control Best Practices*, OWASP Top 10 Security Guidelines, 2024. [Online]. Available: https://owasp.org/

[14] B. Sridhar and R. K. Gupta, "A Study on Online Transportation Platforms and Safety Systems," *IJERT*, vol. 9, no. 3, pp. 980–987, 2020.

[15] S. Das, "Real-Time Communication in Web Applications using WebSockets," *IEEE Internet Computing Journal*, 2022.

[16] S. Raj and G. Nair, "Role of Payment Gateways in Secure Digital Transactions," *International Journal of Information Security*, vol. 11, 2021.

[17] M. Patel and K. Shah, "Implementation of Emergency SOS Alert System using Email Automation," *IRJET*, vol. 7, no. 5, 2020.

[18] JSON Web Token, "Introduction to JWT Authentication," jwt.io, 2023. [Online]. Available: https://jwt.io/introduction

[19] asif-khan-2k19, "Quick Ride – Full Stack Ride Booking Application," GitHub repository, 2023. [Online]. Available: https://github.com/asif-khan-2k19/QuickRide

[20] BcryptJS, *Password Hashing Documentation*, NPM, 2023. [Online]. Available: https://www.npmjs.com/package/bcryptjs

[21] Tailwind Labs, *Tailwind CSS Documentation*, 2024. [Online]. Available: https://tailwindcss.com/

[22] OpenAI, "ChatGPT (GPT-5.1)," Large Language Model, 2024. [Online]. Available: https://chat.openai.com