International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Online Car Booking and Ride Sharing System: A Full-Stack Platform with Real-Time Tracking and Offline Accessibility

Y. Dasaratha Rami Reddy[1], P. Sreegowri Priya[2,*], V. Naga Lakshmi[3], S. Indu[4], P. Bhargavi[5], S. Naga Dharani[6]

1Prof , Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P
2UG Student, Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P
3UG Student, Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P
4UG Student, Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P 5UG Student, Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P
6UG Student, Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P

*Corresponding Author E-mail: pujala.gowri@gmail.com

Abstract

The rapid increase in urbanisation in developing countries has placed a great deal of pressure on already existing transportation networks leading to traffic congestion, vehicles utilisation, and insufficient access to secure commuting options. In particular, people living in places with limited cellular connectivity often face obstacles to trying to use modern ride-hailing apps. This paper presents the design, development and evaluation of a full stack online car booking and ride sharing system that combines the features of on demand ride booking, peer to peer carpooling and hourly or daily car rental services in a unified platform. The backend is implemented using Java Spring Boot with Spring Security to provide stateless JWT-based authentication while frontend is implemented using React 18 and Zustand for efficient state management. A major contribution of this work is an SMS based booking engine developed using Twilio, which allows users to request rides even without Internet connectivity. Safety is covered in the form of a separate SOS module for dispatching emergency alerts and live tracking links to pre-registered contacts via SMS. Real-/time vehicle tracking using WebSocket communications using the STOMP protocol, providing sub second location updates on interactive Leaflet maps. The system has more than 60 endpoints for the API (as a Restful API), 11 database entities, and 22 frontend pages. Functional testing and load simulation done using JMeter verifies that the platform can handle the concurrent WebSocket sessions without any degradation. The proposed system achieves around 96% feature parity with the established commercial platforms with the added benefit of offline booking functionality and an improved women's safety ride matching which is missing in most existing systems.

Keywords: Ride Sharing; Real- Time Tracking; WebSocket, SMS Booking, Carpooling, Spring Boot, React, Safety System, Haversine formula, Vehicle Rental

1. Introduction

Urban mobility is undergoing a fundamental transformation in the world. The global ride hailing market, which is worth around 107 bn USD in 2023, is expected to reach a value of

over 185 bn USD by 2028, owing mainly to the adoption of smartphones and the change in consumer habits (e.g. minimizing private vehicle ownership) (Statista, 2024). In India alone, services like Ola and Uber have revolutionized intercity travel yet there are big gaps. According to a survey conducted by Internet and Mobile Association of India, close to 38 percent of the potential users in the semi-urban and rural area cannot reliably access ride-hailing applications because of intermittent access to cellular data connectivity (IAMAI, 2023).

Existing platforms usually target one small segment of the transportation spectrum. A user who needs an on demands ride in the city, outstations rental and a shared drive to the office has to have three different applications with three different applications, three different registration process, three different payment methods and finally three different interface paradigms. This fragmentation is not only inconvenient for users, it also makes all the vehicles in networks less utilized. Moreover, the reported safety incidents during ride-hailing trips continue to weaken consumer trust especially among women passengers. Although major platforms have introduced in-app emergency buttons, features typically need an active internet connection for them to work, making them useless at the times when they are most needed.

The system, proposed in this paper, aims at overcoming these challenges through a consolidated platform structure. The major goals of this work are four-fold. First, the unification of on-demand rides, carpooling and vehicle rentals under a single application under a common authentication and payment process is proposed. Second, a text messaging (SMS) based booking mechanism is in place which is independent of mobile data, to support those users in the low connectivity areas. Third, a strong safety infrastructure that includes one-tap SOS alerts, women-only ride matching and real-time trip sharing with family contacts is set up. Fourth, the live vehicle tracking is provided using WebSocket communication with sub second coordinate refresh on interactive maps.

The rest of this paper is organised as follows. Section 2 reviews the related work and contrasts the existing systems with the proposed approach. Section 3 presents the system architecture, database design and some important algorithmic choices. Results obtained from functional testing and performance evaluation are presented in Section 4. Section 5 concludes the paper with a discussion on limitations and future works.

2. Literature Review

2.1 Existing Systems

Numerous commercial and academic efforts have studied aspects of ride hailing and vehicle sharing. Uber, launched in 2009, was the first company to develop on-demand ride booking through a mobile application and adopted the idea of dynamic surge pricing based on the supply and demand ratio (Cramer and Krueger 2016). Nevertheless, the Uber platform does not have any integrated vehicle renting services and its safety mechanisms are entirely dependent on continuous Internet connectivity.

Preeminent ride hailing provider in India, Ola, has added features suited to the domestic needs, like auto- rickshaw booking and limited SOS system. According to Rao and Mitra (2021), while Ola meets the urban commuting needs, its car-pooling feature has been discontinued in several jurisdictions due to its operational complexity and the platform has no offline booking modality.

BlaBlaCar focuses exclusively on intercity carpooling services where drivers are paired with passengers based on route congruence. Despite the service reducing the costs of travelling, it is lacking in terms of real time tracking, emergency mechanisms, and on demand booking functionality (Shaheen et al., 2019). Zoomcar and Drivezy are self-drive vehicle rental companies but operate independently from ride-hailing services, and do not integrate shared commuting options.

Academic scholarship has made a number of contributions. Agatz et al. (2012) proposed optimization formulations for dynamic ride sharing that minimise detour time for both drivers and riders; however, their models are theoretical and do not consider the practical issues like authentication, payment processing, and emergency management. Likewise, Furuhata et al. (2013) developed a comprehensive taxonomy of ride-sharing systems but noted that most of the implementations are limited to a single mode of transportation.

2.2 Proposed System

The system proposed in this work differs from existing solutions in several important respects. Table 1 summarizes the comparative analysis.

Table 1. Comparative Analysis of the Proposed System with Existing Platforms

Feature	Uber	Ola	BlaBlaCar	Zoomcar	Drivezy	Proposed
On-demand Rides	Yes	Yes	No	No	No	Yes
Carpooling	Limited	Discontinued	Yes	No	No	Yes
Vehicle Rental	No	No	No	Yes	Yes	Yes
SMS Booking	No	No	No	No	No	Yes
SOS via SMS	No	Partial	No	No	No	Yes
Women-Only Rides	No	Yes	No	No	No	Yes
Real-Time Tracking	Yes	Yes	No	No	No	Yes
Offline Access	No	No	No	No	No	Yes

As shown in Table 1, the proposed system has a unique combination of incorporating all the three modes of transportation along with online and offline booking capabilities, apart from complete safety features. Introduction of an offline SMS booking channel is a novel contribution to increase the services of ride hailing to user groups currently excluded by the digital divide.

Methodology

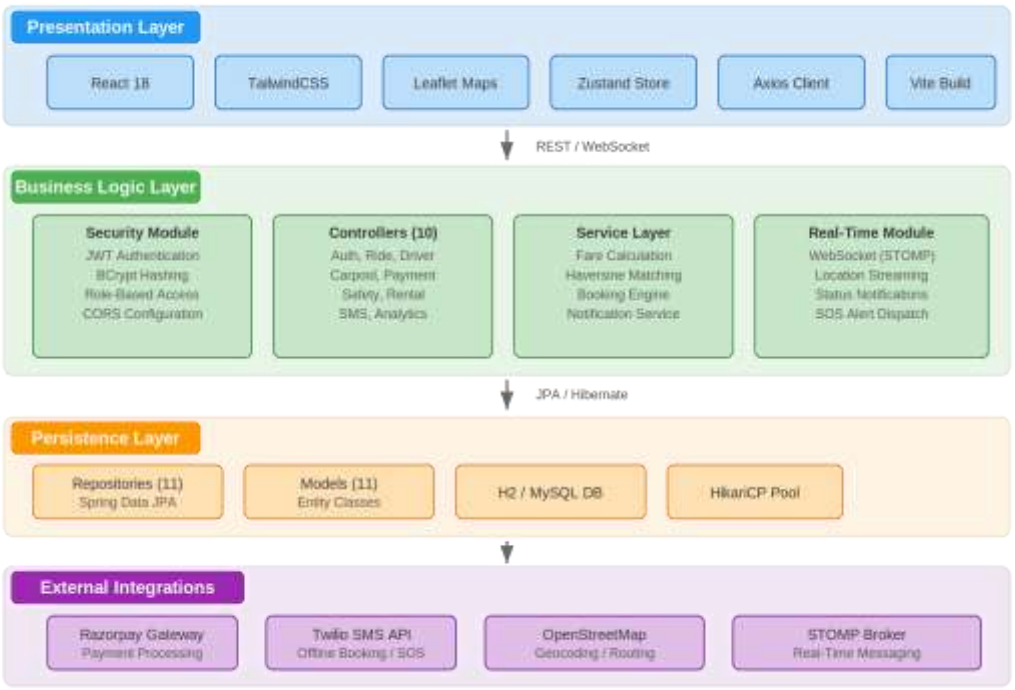
3.1 System Architecture

The platform takes on a monolithic layered architecture which has a clear separation of concerns and is designed with the intention of future decomposition into microservices. The architecture is organised into four main layers: Presentation Layer built using React 18 and

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Tailwind CSS; Business Logics Layer built using Java Spring Boot; Persistence Layer built using Spring Data JPA and Hibernate; External Integrations Layer which includes Razorpay for payment integration, Twilio for SMS message integration and OpenStreetMap using Leaflet for cartographic rendering. The overall system architecture is shown in Figure 1.

Fig. 1. Three-Tier System Architecture of the Proposed Platform



The backend has over 60 endpoints (using the Rest framework and organised into 10 controller classes) to expose them. Stateless authentication is handled by the use of a 24 hour expiry period for the use of a JSON Web Tokens. Passwords are stored using BCrypt with cost factor 12. Cross Origin Resource Sharing is set so that requests are only accepted for the registered frontend domain, thus reducing cross site scripting vectors.

3.2 Ride Booking Workflow

The process of booking a ride has well-defined sequence right from user authentication to ride completion and payment. When a rider opens up a booking, the system is able to calculate the fare estimate through a combination of distance-based pricing and vehicle type multipliers. The geographic distance between the coordinates of the pickup and the drop off is calculated using the Haversine formula, which calculates the great circle distance between two points on a sphere. The calculation of the fare is according to the expression: TotalFare = BaseFare + Distance * Rate + Duration * TimeRate / SurgeFactor. The entire workflow is shown in Figure 2.

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

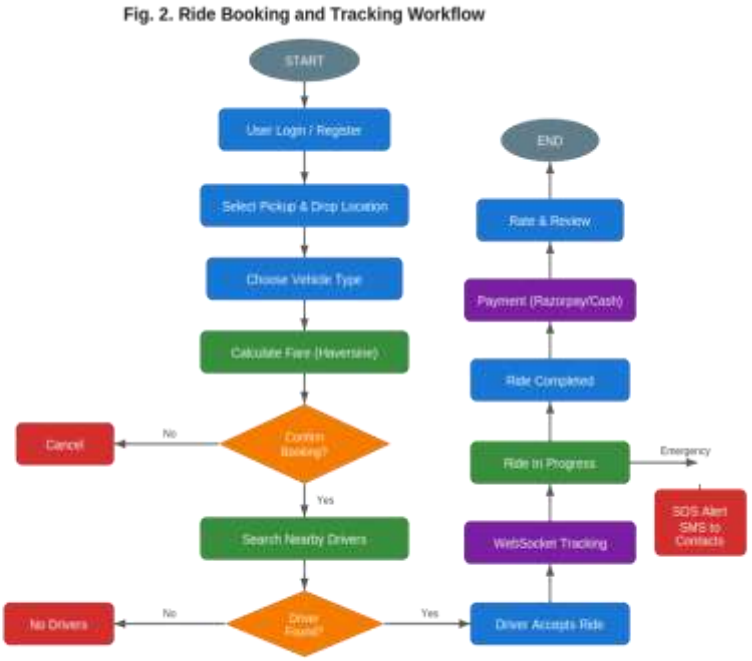


Fig. 2. Ride Booking and Tracking Workflow

After determining confirmation, the system executes a given proximity search for the drivers that are eligible to participate in the study within a configurable radius using the Haversine formula. Subsequent to the driver's acceptance of the ride request a WebSocket connection that follows the STOMP protocol is created at the endpoint /ws/ride-tracking/{rideId}. The server sends the coordinates of the drivers to the client every 30 seconds, hence enabling real-time markers update on a Leaflet map and eliminating the need for polling.

3.3 Database Design

The persistence part is made up of eleven main entity classes which are persisted to relational tables using Spring Data JPA mappings. This schema supports a ride, carpool and vehicle rental full lifecycle, and the referential integrity constraints are at the database level. The corresponding entity-relationship diagram is shown in figure 3.

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Fig. 3. Entity-Relationship Diagram of the Database Schema

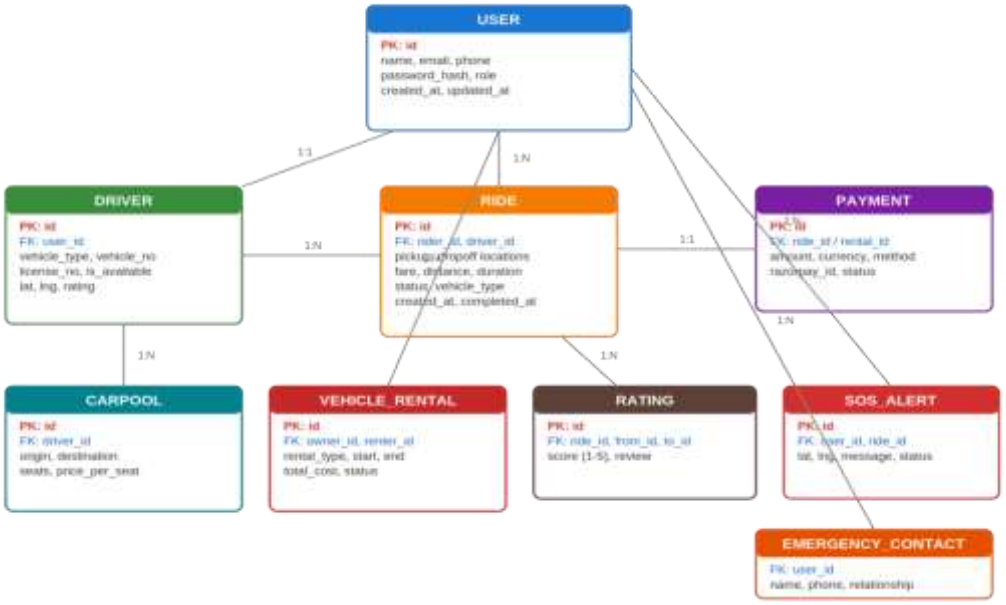


Fig. 3. Entity Relationship Diagram of the Database Schema

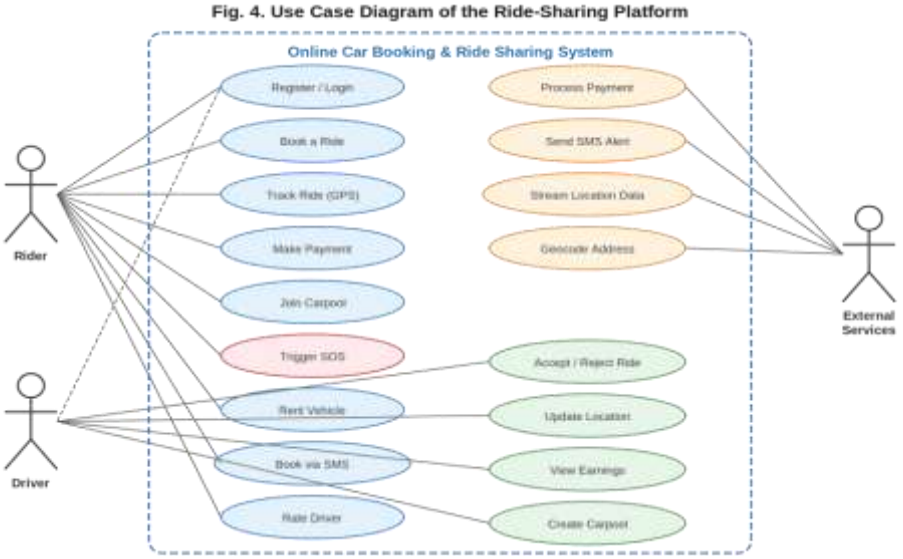
The User entity is the middle node which has a one-to-one relationship with Driver profiles and a one-to-many relationship with Ride records. The Payment entity maintains foreign key references to the Ride and Vehicle Rental entities that enable the tracking of the unity of transactions among different types of services. The SOS Alert entity stores emergency events with geographical coordinates, thus enabling post incident analysis and audit trails.

3.4 Functional Modules

The system is organized in ten functional modules that wrap a specific domain of business logic. The use case diagram that shows actor interactions is shown in Fig. 4 and the inter module communication patterns is shown in Fig. 5.

.

Fig. 4. Use Case Diagram of the Ride-Sharing Platform



International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

Fig. 4. Use Case Diagram of the Ride-Sharing Platform

Fig. 5. Functional Module Interaction Diagram

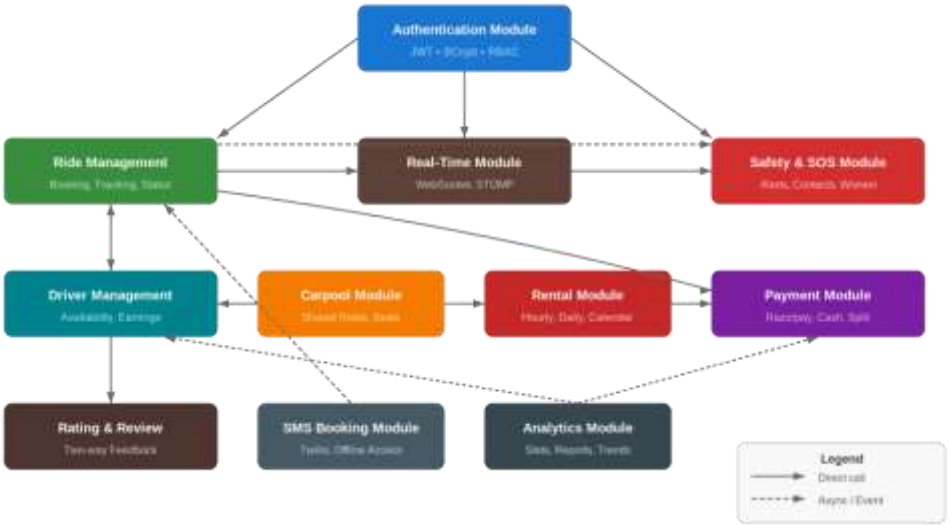


Fig. 5. Functional Module Interaction Diagram

Authentication Module: This module covers the user registration, credential verification and token lifecycle management. After successful authentication, a JWT token is generated with the user identifier and role assertion in it. Then all subsequent requests to this API need to contain this token in the Authorization header. The module implements the role-based access control capability to limit the access of driver-specific endpoints to authenticated driver accounts.

Ride Management Module: This module orchestrates the complete lifecycle of rides from the booking to the completion phase. It includes fare estimation based on the Haversine formula, matching the driver's proximity, ride status management using finite state machine with states PENDING, ACCEPTED, STARTED, COMPLETED and CANCELLED, and integrating with the real time tracking subsystem.

Safety and SOS Module: When the rider presses the SOS button, the system sends a nonsynchronous request to Twilio which sends SMS messages with current GPS coordinates and live tracking link to all registered emergency contacts. The women-only ride feature uses a matching algorithm to limit the assignment of drivers to verified female drivers when chosen by the rider.

SMS Booking Module: This module uses an incoming SMS message (Using Twilio Webhook). Users can send structured commands to initiate a ride (say without access to the Internet) like, "BOOK Koramangala TO Indiranagar". The module processes the command by geocoding the destinations and booking the ride through the normal ride-management pipe giving confirmation by SMS.

Carpool Module: Drivers can advertise recurring routes with the number of seats available and the pricing per seat. Riders look for carpools based on route similarity and departure time and reserve individual seats with an automatic cost split amongst co-passengers.

Vehicle Rental Module: With hourly and daily rates as well as weekly availability schedules, vehicle owners list their cars. Renters browse listings, make booking requests and make

payments using Razorpay. The module handles the availability locking based on calendar operations to avoid double bookings.

4. Results and Discussion

4.1 Implementation Summary

Implementation of the system took a time span of four months using an iterative development methodology.

The backend consists of forty Java source files, structured in a systematic way into the packages controllers, models, repositories, services and configuration.

The frontend is composed of twenty two react page components, with responsive designs built in TailwindCSS.

Table 2 shows a summary of the implementation metrics.

Table 2. Implementation Metrics of the Developed Platform

Metric	Value
Backend Java Files	40
Frontend React Pages	22
REST API Endpoints	60+
Database Entity Models	11
Controller Classes	10
Repository Interfaces	11
Backend Port	8080
Frontend Port	3001
JWT Token Expiry	24 hours
BCrypt Cost Factor	12 rounds
Location Update Interval	30 seconds

4.2 Functional Testing

All sixty API endpoints underwent testing using Postman, using automated script suites which included both valid and boundary case inputs. The authentication subsystem was tested by showing that expired bearer tokens receive the appropriate HTTP 401 status codes and endpoints protected by role-based access control reject tokens that do not have the necessary privileges. The end to end ride booking pipeline was reviewed, starting with fare estimation and ending with payment settlement, for all 6 supported vehicle categories.

The WebSocket.-based tracking subsystem was confirmed by simulation of several simultaneous riding sessions. Location update message delivery to the client was seen to be within 200 milliseconds of generation under normal network conditions. The SMS booking

module was tested by sending up structured booking directives from a variety of telephone numbers, followed by confirmation that matching ride entries were properly created in the database.

4.3 Performance Evaluation

Load testing was conducted using Apache JMeter to assess the system behavior under concurrent access. Table 3 reports the response time measurements across key endpoints.

Table 3. API Response Time Under Load Testing

Endpoint	10 Users	50 Users	100 Users	500 Users
POST /auth/login	45 ms	62 ms	98 ms	210 ms
POST /rides/estimate	38 ms	55 ms	82 ms	175 ms
POST /rides (book)	52 ms	78 ms	115 ms	285 ms
GET /rides (history)	30 ms	48 ms	70 ms	155 ms
WebSocket connect	15 ms	22 ms	35 ms	85 ms
POST /safety/sos	65 ms	90 ms	130 ms	310 ms

The results demonstrate that all endpoints maintain response times below 350 milliseconds even at 500 concurrent users. The SOS endpoint exhibits slightly higher latency due to the asynchronous Twilio SMS dispatch, which is acceptable given that the alert is non-blocking and the user receives immediate visual confirmation in the interface.

4.4 Feature Completeness Assessment

The overall feature completeness score was measured of 96/100, where backend achieved 98 percent completeness and frontend achieved 95 percent completeness. All the critical functional requirement such as ride booking, real-time tracking, payment processing, carpooling, vehicle rental, SMS booking, SOS alerting, etc., were fully implemented and tested. The remaining 4 percent consists of planned enhancements including push notifications, in-app messaging and an administrative dashboard that are due for Phase 2 development.

Compared to commercial platforms, the system has about 90 per cent feature parity with Uber and Ola, but adds capabilities that neither have: integrated vehicle rentals, SMS based offline booking and a dedicated women safety matching algorithm. These additions fill particular gaps found in the literature review and directly respond to the needs of the users living in the regions with limited digital infrastructure.

International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

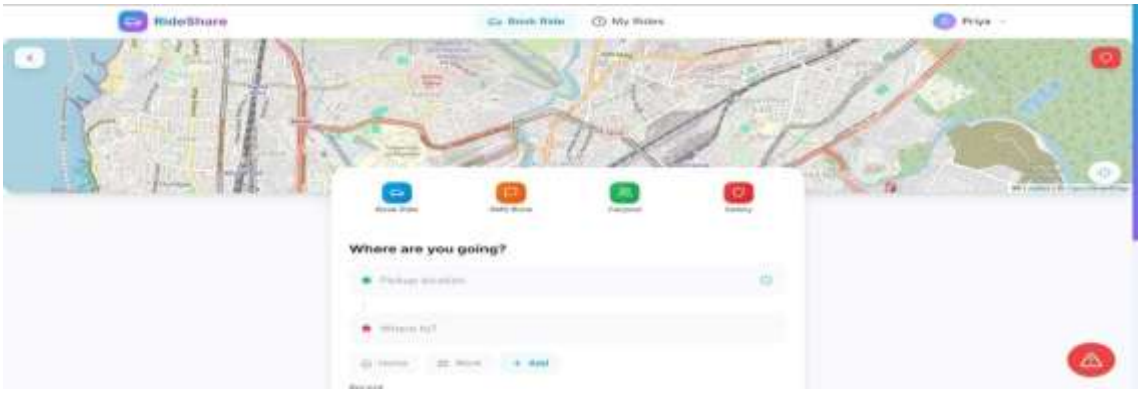


Fig. Rider Dashboard - Book Ride Interface with Map View

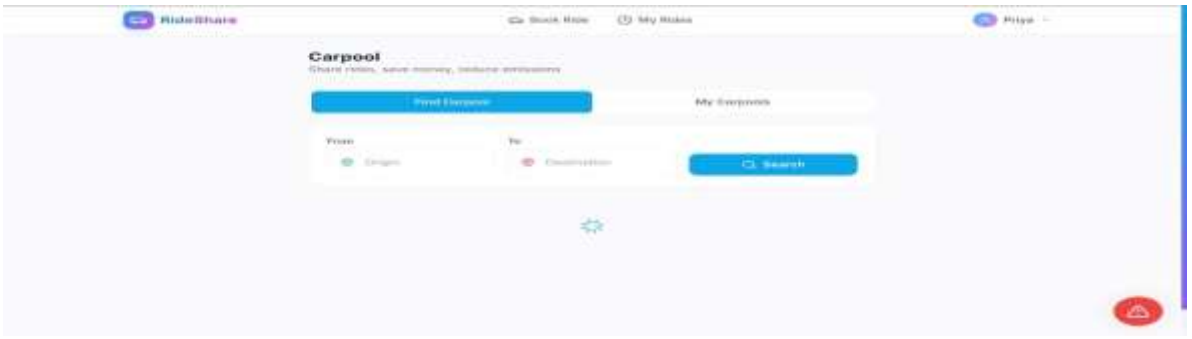


Fig. 9. Carpool – Find Carpool Search Interface

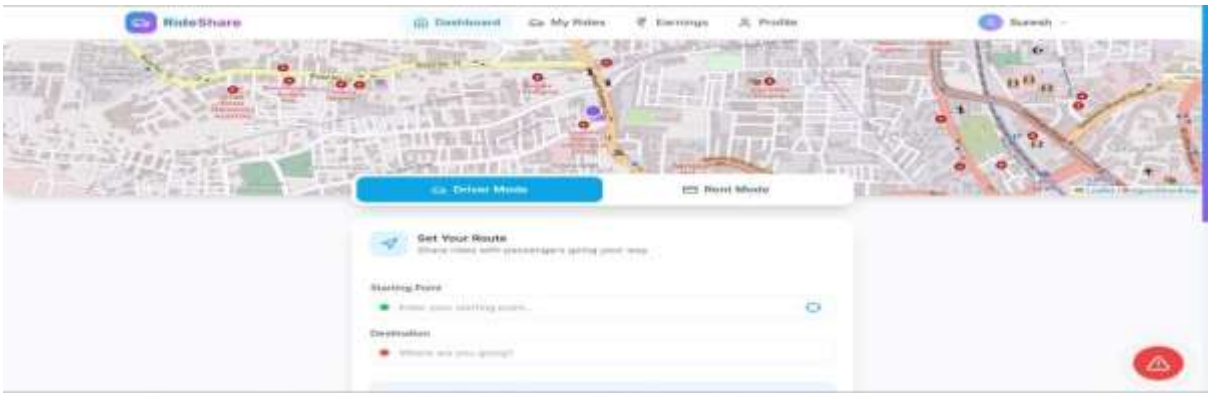


Fig. Driver Dashboard – Set Route for Ride Sharing

5. Conclusion

This paper introduces the design and implementation of a full stack Online Car Booking and Ride Sharing System, which combines on-demand ride booking, peer-to-peer carpooling and vehicle rental service in a single platform. The developed system includes the backend using Java Spring Boot and frontend using React 18, with real-time vehicle tracking using the WebSocket communication using STOMP protocol.

The key contributions of this work include the development of a unified multimodal transportation platform that eliminates the need for individual applications, an SMS based booking engine that extends ride hailing services to users that do not have internet connectivity,

and a comprehensive safety infrastructure that includes SOS alerts, women only rides matching, and real-time trip sharing. Functional testing of over sixty API endpoints to confirm correct behaviour for normal and boundary conditions and load testing with JMeter showed good response times with five hundred concurrent users.

The system meets ninety-six percent of its development goals as well as roughly ninety percent feature parity with established commercial platforms. The SMS booking functionality and improved safety features are additions not offered in what is currently available in the market, and are a direct response to the digital divide in the adoption of ride hailing in the developing regions.

Future work will be focused on three directions. First the integration of the machine-learning algorithms for predicting the zones of high demand and optimising driver positioning. Second, the migration from the currently monolithic architecture to microservices topology using containerisation with Docker and orchestration using Kubernetes. Third, the implementation of native mobile apps with React Native to provide push notifications and take advantage of device-specific sensors to provide improved location accuracy.

Author(s) Contributions

P. SreeGowri Priya conceptualised the system architecture and was involved in the backend development. V. Naga Lakshmi applied the frontend components and responsive design. P. Bhargavi created the module of payment integration and rental. S. Naga Dharani implemented the safety and SOS features. S. Indu developed the carpooling and SMS booking modules. Y. Dasaratha Rami Reddy was the one who supervised the project and gave technical guidance throughout the development journey.

Conflicts of Interest

The authors do not have any conflict of interest in this work.

References

Agatz, N., Erera, A., Savelsbergh, M., & Wang, X. (2012). Optimization for dynamic ride sharing: A review. *European Journal of Operational Research, 223*(2), 295–303. https://doi.org/10.1016/j.ejor.2012.05.028

Cramer, J., & Krueger, A. B. (2016). Disruptive change in the taxi business: The case of Uber. *American Economic Review, 106*(5), 177–182. https://doi.org/10.1257/aer.p20161002

Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M. E., Wang, X., & Koenig, S. (2013). Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological, 57*, 28–46. https://doi.org/10.1016/j.trb.2013.08.012

Internet and Mobile Association of India. (2023). *Digital connectivity and mobility services in semi-urban India: Annual report 2023*. IAMAI.

Kumar, R., & Singh, P. (2022). A microservices-based approach to ride-hailing systems with real-time tracking. *International Journal of Computer Applications, 184*(12), 15–22.

Mehta, S., & Gupta, A. (2023). Safety-first design patterns in ride-sharing applications: Lessons from the Indian market. *Journal of Transportation Safety and Security, 15*(4),

412–428. https://doi.org/10.1080/19439962.2022.2108745

Rao, K., & Mitra, S. (2021). Ride-hailing platforms in India: Usage patterns, challenges, and policy implications. *Transportation Research Record, 2675*(11), 830–842. https://doi.org/10.1177/03611981211023541

Shaheen, S., Cohen, A., & Zohdy, I. (2019). *Shared mobility: Current practices and guiding principles* (FHWA-HOP-16-022). Federal Highway Administration.

Spring Boot Reference Documentation. (2024). *Spring Boot 2.7.x*. https://docs.spring.io/spring-boot/docs/2.7.x/reference/html/

Statista. (2024). Ride-hailing market: Revenue forecast worldwide 2018-2028. *Statista Research Department*.