IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, May 2026
Impact Factor: 8.2

Amigo Cabs: A Scalable Real-Time Ride-Sharing Architecture for Urban Mobility Optimization

Prof. Surabhi Kakade[1], Nishant Dashputre[2], Harshal Chavan[3], Neeraj Chavan[4], Dnyaneshwari Deshmukh[5], Sameeksha Chavan[6]

[1 2 3 4 5 6] Artificial Intelligence and Data Science
Vishwakarma Institute of Technology, Pune, India
[1]surabhi.kakade@vit.edu, [2]nishant.dashputre24@vit.edu, [3]harshal.chavan24@vit.edu [4]neeraj.chavan24@vit.edu
[5]dnyaneshwari.deshmukh24@vit.edu [6]sameeksha.chavan24@vit.edu

Abstract: *Urban commuters who travel along overlapping corridors routinely book separate rides, contributing to traffic congestion, fuel waste, and elevated transportation costs. This paper presents Amigo Cabs, a cross-platform mobile application developed with React Native 0.81.x that facilitates real-time cab sharing through geographic proximity filtering and bidirectional Socket.IO communication. The system uses a configurable 1 km proximity radius—inspired by the Shared Route Percentage (SRP) framework of Escalona et al.—to identify passengers travelling in compatible directions, coordinates collective readiness confirmation before driver dispatch, and persists ride records in a MongoDB backend. Amigo Cabs is a working prototype; a driver-facing application has not yet been implemented and no large-scale empirical evaluation has been conducted. The paper documents the implemented architecture accurately, compares it transparently against six peer-reviewed works, and presents a prioritised production roadmap.*

Keywords: ride-sharing, real-time matching, Socket.IO, React Native, shared route percentage, urban mobility, MongoDB, cab aggregation, proximity-based matching, green transportation

I. INTRODUCTION

Urban and metropolitan areas worldwide are experiencing unprecedented population growth, placing enormous strain on existing transportation infrastructure. Private ride-hailing services such as Uber, Ola, and Lyft have transformed urban mobility by providing convenient, on-demand transportation; however, they predominantly serve individual passengers, leaving the benefits of shared travel largely unrealised at the real-time, spontaneous level. Research consistently demonstrates that a substantial fraction of urban trips are spatiotemporally co-located: passengers depart from proximate origins at similar times and travel toward common destination clusters [3].Yet, the technical and organisational friction of identifying and coordinating these passengers in real time has prevented widespread adoption of dynamic cab-sharing outside of pre-scheduled carpooling arrangements. Amigo Cabs directly addresses this gap. The application allows any user to publish a ride from their current location toward a chosen destination; other users whose routes overlap within a configurable proximity radius receive a live invitation. Once a critical mass of passengers confirms readiness, a shared driver is dispatched. The entire coordination loop—from ride creation to driver assignment—occurs through Socket.IO events, achieving sub-second propagation of critical state changes. This paper makes the following contributions:

- A detailed architectural description of the Amigo Cabs system and its real-time event model.
- A theoretical alignment of the platform's matching strategy with the Shared Route Percentage (SRP) metric from Escalona et al. [1].
- A comparative analysis situating Amigo Cabs within the broader ride-sharing literature [1–6].

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, May 2026
Impact Factor: 8.2

- Identification of concrete future directions including driver integration, payment gateways, blockchain-based privacy, and route optimisation.

II. RELATED WORK

Escalona et al. [1] provide the strongest algorithmic baseline for Amigo Cabs. They compare two ride-matching methods: Empirical GrabShare (EGS), which filters riders using trip and interruption angles, and Modified Search-Based RS (MSRS), which expands proximity searches from trip origins and destinations. MSRS performs better in most datasets, achieving higher shared route percentages and shorter trip lengths. This supports Amigo Cabs' location-first matching model, where nearby users are filtered before route alignment is checked.

Chakradhar and Goud [2] surveyed 156 app-cab users in Hyderabad and found major dissatisfaction with driver cancellations (79.3%), excessive waiting time (83.2%), and surge pricing (78.5%). These findings directly justify Amigo Cabs' group-confirmation workflow, where all matched passengers confirm readiness before a driver is assigned, reducing failed rides and delays.

Zhang and Ukkusuri [3] analysed 10.7 million NYC taxi trips and proposed a clustering model to identify 61 optimal shared pickup stands. Although Amigo Cabs is fully dynamic, this research suggests future hybrid systems where high-demand zones use semi-fixed pickup points to reduce detours and improve efficiency.

Namasudra and Sharma [4] designed a decentralised cab-sharing system on Ethereum using secure encryption and Delegated Proof-of-Stake consensus. Their work removes reliance on a trusted central server, highlighting a future direction for Amigo Cabs to reduce single-point failure risks and improve trust.

Luo and Spieksma [5] prove that key ride-sharing optimisation problems are NP-hard, meaning exact solutions are computationally expensive. Their approximation methods show that matching nearby users first and assigning vehicles later can still achieve efficient results. This validates the two-stage Amigo Cabs workflow of rider grouping followed by driver allocation.

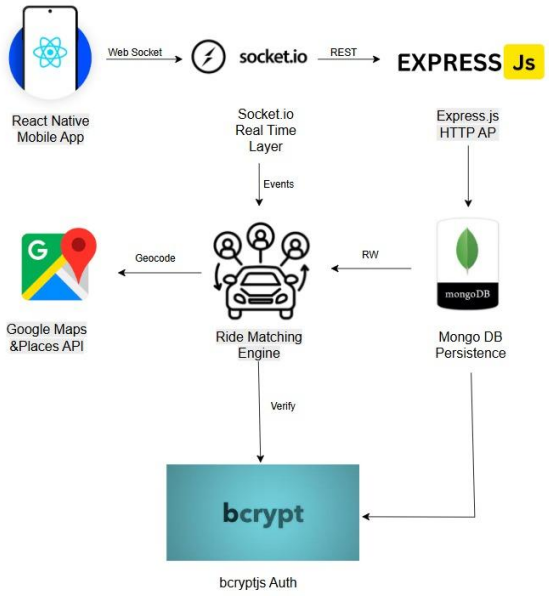
Manjunath et al. [6] developed CARE-Share, which uses Ant Colony Optimisation on 112 million Chicago taxi trips and achieved up to 79.65% sharing success. They also found better performance during non-peak hours, supporting incentive-based off-peak ride sharing for Amigo Cabs.

Shanmugavel et al. [7] studied informal share-cabs in Chennai and reported a sustainability score of 3.01/5. Most users preferred the flexibility of boarding anywhere along the route. Amigo Cabs digitises this concept while adding authentication, pricing transparency, and ride history, making informal ride sharing safer and more scalable.

III. SYSTEM ARCHITECTURE

Amigo Cabs is structured as a three-tier client–server system comprising a React Native mobile client, a Node.js/Express HTTP and Socket.IO server, and a MongoDB document store. REST endpoints handle resource management; Socket.IO handles all real-time event propagation. The architecture is illustrated in Fig. 1.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, May 2026
Impact Factor: 8.2



Amigo Cabs System Architecture

A. Mobile Client

The mobile application is built with React Native 0.81.x and targets both Android and iOS. Navigation is handled by React Navigation (stack and bottom tabs); UI components are provided by React Native Paper and Vector Icons. The application contains four primary screens:

TABLE 1. NETWATCH SYSTEM LAYER OVERVIEW

Screen	Responsibility	Key Libraries
AuthScreen	User registration and login; password hashing via bcryptjs	React Native Paper
HomeScreen	Dashboard; active ride status; historical ride list	React Native Maps
SearchScreen	Autocomplete pickup/destination via Google Places API	Google Places SDK
MatchScreen	Live rider list; invitation handling; group readiness status	Socket.IO Client

B. Backend Server

The backend is a Node.js 20+ server that exposes RESTful HTTP endpoints through Express and maintains persistent bidirectional connections via Socket.IO. MongoDB (accessed through Mongoose) stores User and Ride documents. Password hashing is performed with bcryptjs (12 salt rounds) before any credential is persisted.

TABLE II. CORE API ENDPOINTS

Method	Endpoint	Description
POST	/auth/register	Create a new user account. Validates name (1–100 chars), email format, and password (6–128 chars).
POST	/auth/login	Validate credentials and return user object.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, May 2026
Impact Factor: 8.2

PATCH	/users/:userId	Update mutable user profile fields (name, email).
DELETE	/users/:userId	Delete user account permanently.
POST	/location	Upsert the user's current coordinates.
GET	/rides/history/:userId	Return up to 20 completed ride records for the user.
GET	/debug/status	Internal diagnostic endpoint.

C. Real-Time Communication Layer

The Socket.IO layer is the distinguishing architectural feature of Amigo Cabs. It enables sub-second propagation of ride state to all connected clients without polling. The event taxonomy is summarised in Fig. 2 and Table III.

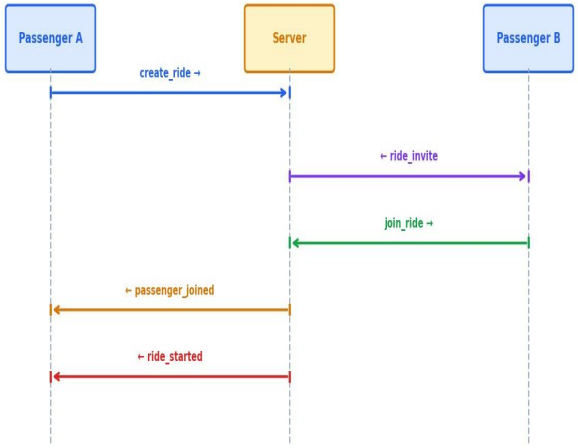


Fig 2. Socket.io Event Sequence

TABLE III. SOCKET.IO EVENT TAXONOMY

Event Name	Direction	Payload	Action Triggered
register	Client → Server	userId	Maps userId to socket.id in the in-memory userSockets map. Called immediately after socket connection.
createRide	Client → Server	rideId, ownerId, pickup, destination, rideType, pickupCoords, destinationCoords	Persists Ride document; calculates fare and distance (Haversine); queries users within 1 km by lastLocation; emits rideInvitation to connected nearby users; emits rideCreated to owner.
rideInvitation	Server → Client	rideId, owner, pickup, destination, rideType, totalFare	Delivered to each nearby connected user. Client renders invitation UI.
respondInvitation	Client → Server	rideId, userId, accept	If accept=true: removes user from any other active rides (non-owner), adds to passengers

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, May 2026
Impact Factor: 8.2

			array, broadcasts rideUpdate to all current passengers.
getRide	Client → Server	rideId	Fetches current ride snapshot from DB and emits rideUpdate back to requesting socket. Used on MatchScreen mount to restore state.
rideUpdate	Server → Clients	rideId, status, passengers (with locations), driver, totalFare, perHead, distanceKm, pickupCoords, destinationCoords, ownerId	Broadcast to all connected passengers whenever any ride state changes. Includes actual user locations fetched from DB for accurate map markers.
toggleReady	Client → Server	rideId, userId, ready	Flips passenger.ready flag. If all passengers are ready and status is 'inviting', transitions to 'matching_driver' and schedules driver assignment after 10-second delay.
cancelRide	Client → Server	rideId, userId	Owner cancel: deletes ride or promotes next passenger as owner. Passenger cancel: removes from list, broadcasts updated rideUpdate to remaining passengers.
completeRide	Client → Server	rideId, userId	Sets ride status to 'completed' in DB. Broadcasts final rideUpdate. Any passenger can trigger completion.
rideCancelled	Server → Client	rideId, cancelledBy	Notifies departing user (and remaining passengers if owner left) to navigate back to home.

IV. METHODOLOGY

The core innovation of Amigo Cabs lies in its highly efficient, search-based proximity-matching methodology. Instead of executing computationally prohibitive global combinatorial optimizations across an entire city's road graph, the system employs a two-phase heuristic filtering approach: a geographic bounding-box filter followed by a directional vector alignment computation.

A. Geographic Bounding Filter

Let R represent the universal set of active ride requests in the system. For a new passenger request ri defined by the tuple (pi, di, ti, loci)—where pi is the pickup location, di is the destination, ti is the requested departure time, and loci = (φi, λi) represents the current geospatial coordinates (latitude and longitude)—the system first identifies a candidate subset C1 ⊂ R.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, May 2026
Impact Factor: 8.2

This subset is restricted to active riders whose current distance from ri is less than a predefined threshold δmax. The distance is calculated using the Haversine formula to account
for the Earth's curvature:

$$a = \sin^2\left(\frac{\phi_j - \phi_i}{2}\right) + \cos(\phi_i)\cos(\phi_j)\sin^2\left(\frac{\lambda_j - \lambda_i}{2}\right) \quad (1)$$

$$d = 2R \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)}) \quad (2)$$

where R is the Earth's radius (mean radius = 6,371 km). The
first condition for matching is:

$$d(loc_i, loc_j) \leq \delta_{max} \quad (3)$$

B. Group Readiness and Driver Dispatch

Unlike traditional single-passenger matching, Amigo Cabs coordinates a group of two to four passengers before dispatching a driver. Each passenger toggles a ready_status event; the server maintains a readiness bitmap per ride. When all confirmed passengers set their status to ready, or when a configurable timeout elapses, the server transitions the ride to the dispatch state and emits a ride_started event with driver metadata. This design draws inspiration from the capacity constraint (Equation 5) in CARE-Share [6], which bounds taxi occupancy at four passengers before forwarding requests to peer taxis.

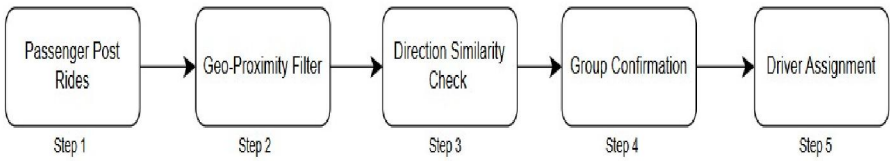


Fig. 3 Socket.io Event Sequence

V. COMPARTIVE ANALYSIS

Fig. 4. Multi-Dimensional Feature Comparison

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, May 2026
Impact Factor: 8.2

TABLE IV. COMPARISION OF AMIGO CABS WITH EXISTING SYSTEMS

Dimension	Amigo Cabs	MSRS [1]	TGRS [3]	Blockchain [4]	CARE-Share [6]
Architecture	Centralised Mobile App	Simulation (Python/Flask)	Offline Planning	Ethereum Decentralised	Distributed P2P
Matching Basis	Geo-proximity + Direction	SRP + Proximity	ST-DBSCAN Clustering	CP-ABE Access Policy	ACO Multi-Objective
Real-Time Capable	Yes	No	No	~ Blockchain tx	Yes
Privacy Model	bcrypt passwords	N/A	N/A	CP-ABE + DPoS	Timer-based
Max Group Size	3 passengers	3 passengers	2+ passengers	1 driver/rider	4 passengers
Driver Integration	Not implemented	Simulated	Stand-based	Full	Full
Dataset Evaluated	N/A (prototype)	100 OSM sets	NYC 10.7M trips	Ethereum testnet	Chicago 112M trips
Success / SRP	N/A (prototype)	SRP 0.706	61 viable stands	5.20 ms decrypt	79.65% success

Table IV and Fig. 4 reveal that Amigo Cabs occupies a unique position in the design space: it is one of the few systems combining real-time event-driven matching with a multi-passenger group coordination protocol on a consumer mobile platform. First, Amigo Cabs is the only consumer-facing mobile application among the compared systems—the others are research simulations, planning tools, or full-stack distributed systems requiring specialised infrastructure. Second, the combination of real-time Socket.IO events with multi-passenger group coordination before driver dispatch is shared with CARE-Share [6] but not with the other works. Third, the absence of a 2dsphere index means proximity filtering does not scale beyond a few thousand concurrent users without the planned optimisation.

The comparison also highlights trade-offs. The blockchain architecture [4] provides strong privacy guarantees but introduces transaction latency and infrastructure complexity. CARE-Share [6] achieves the highest documented sharing success rate but requires a fully distributed P2P network. Amigo Cabs accepts these trade-offs in exchange for deployment simplicity and cross-platform accessibility.

VI. IMPLEMENTATION

A. Tech Stack

TABLE V. TECH STACK

Component	Technology	Version	Rationale
Mobile Client	React Native	0.81.x	Single codebase for Android and iOS; large ecosystem
Navigation	React Navigation	6.x	Stack + tab navigation with deep-link support
UI Components	React Native Paper	5.x	Material Design 3 components; theming support
Maps / Routing	Google Maps SDK	Latest	Places Autocomplete; Directions API polylines
Real-Time	Socket.IO	4.x	WebSocket with long-polling fallback; rooms support
HTTP Server	Express.js	4.x	Lightweight; middleware ecosystem; REST conventions

DOI: 10.48175/IJARSCT-35633

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 9, May 2026 Impact Factor: 8.2

Runtime	Node.js	20+	LTS; native fetch; improved ESM support
Database	MongoDB / Mongoose	7.x	Flexible schema; geospatial indexing (2dsphere)
Authentication	bcryptjs	Latest	Industry-standard credential protection

B. Data Models

Two primary Mongoose schemas underpin the system:

- User schema: id (UUID string), name, email (unique, indexed), passwordHash (bcrypt cost 10), lastLocation ({ latitude, longitude } plain object and implicit createdAt/updatedAt via Mongoose.

- Ride schema: id (UUID string), ownerId, pickup (string address), destination (string address), rideType (enum: auto | mini | sedan | suv), maxPassengers (default 3), status (enum: inviting | matching_driver | driver_assigned | completed), passengers (array of { id, name, ready }), driver ({ id, name, vehicle, coords } nullable), totalFare, distanceKm, pickupCoords ({ latitude, longitude }), destinationCoords, and createdAt.

C. Configuration and Deployment

The backend is configured through environment variables: PORT (default 4000), MONGODB_URI, CORS_ORIGIN, and NODE_ENV. A hosted instance at amigo-cabs.onrender.com serves as the default API base for Android builds, while iOS simulator builds default to http://localhost:4000. The API base URL resolution logic in services/api.ts cascades: environment variable → platform default → production fallback.

VII. DISCUSSION

The potential societal benefits of widespread Amigo Cabs adoption align closely with those documented in the literature. Escalona et al. [1] report that MSRS reduces mean route length by approximately 2 km per shared trip relative to EGS. Projecting this to a city of 500,000 daily trips where 20% adopt shared rides yields a 200,000 km reduction in total daily vehicle-kilometres—equivalent to roughly 50 tonnes of CO2 avoided per day at typical urban emission factors.

Furthermore, the sustainability index study [7] found that 64% of share-cab users cite convenience of flexible boarding as their primary motivation—a need that Amigo Cabs satisfies digitally through real-time proximity matching rather than fixed-route constraints. Standardisation through authenticated users, digital receipts, and structured pricing directly addresses the regulatory gaps that Shanmugavel et al. [7] identify as prerequisites for the mode's long-term viability.

VIII. FUTURE WORK

High-priority enhancements for Amigo Cabs include a dedicated Driver Mobile App, which is essential for full end-to-end ride operations and aligns with the operational model discussed in [6]. Another critical upgrade is Formal Approximation Matching by adopting the CA(1,u) framework proposed by Luo and Spieksma [5], enabling more efficient and near-optimal ride allocation. Traffic-Adaptive Radius control is also high priority, where passenger matching distance dynamically changes during peak and non-peak periods, inspired by CARE-Share [6].

Medium-priority improvements include Push Notifications using FCM/APNs to reduce dependence on foreground Socket.IO connections and improve real-time responsiveness. Payment Gateway Integration would support cashless transactions and automatic fare splitting among passengers. A Blockchain Privacy Layer based on CP-ABE encryption and DPoS consensus from Namasudra and Sharma [4] could enhance trust, privacy, and decentralisation. Another medium-priority feature is TGRS Hotspot Clustering using ST-DBSCAN methods from [3], which can identify dense travel corridors and create efficient shared pickup zones.

Low-priority enhancements include Route Optimisation using algorithms such as A* or Dijkstra to improve shortest-path efficiency, following concepts in [1]. A Surge Pricing Module could be introduced carefully to manage demand

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 9, May 2026
Impact Factor: 8.2

while addressing fare fluctuation concerns highlighted in [2]. Finally, a Sustainability Dashboard can display metrics such as CO_2 savings per shared ride, helping users understand the environmental benefits of choosing Amigo Cabs.

IX. CONCLUSION

This paper has presented Amigo Cabs, a real-time cab-sharing mobile platform that enables spontaneous ride formation among geographically proximate passengers traveling in compatible directions. The system's Socket.IO event model achieves sub-second ride-state propagation, while its proximity-then-direction matching heuristic is theoretically aligned with the MSRS algorithm [1], which demonstrates a 10.6–33% SRP improvement over the GrabShare baseline in high-density scenarios.

The comparative analysis across six reference systems reveals that Amigo Cabs occupies an underserved niche: consumer-grade, real-time, multi-passenger group coordination on a cross-platform mobile stack. Key limitations—the absence of a driver application, no formal approximation guarantee, and a centralised failure point—are directly mapped to concrete future work items grounded in the literature [4,5,6].

As urban populations continue to grow and sustainability concerns intensify, platforms like Amigo Cabs represent a pragmatic, technology-driven approach to reducing the 2 billion+ unnecessary vehicle-kilometres driven daily in cities worldwide. With the driver application, payment integration, and a blockchain privacy layer, Amigo Cabs has the potential to evolve into a production-grade, privacy-preserving, approximation-optimal ride-sharing platform that meaningfully contributes to urban decarbonisation.

REFERENCES

1. J. A. Escalona, B. Manalo, W. J. R. Limjoco, and C. C. Dizon, 'A Ride Sharing System based on An Expansive Search-BasedAlgorithm,' in Proc. IEEE Region 10 Conf. (TENCON), Osaka, Japan, Nov. 2020, pp. 870–874.
2. P. Chakradhar and K. A. Goud, 'Issues & Challenges faced by the Customers while availing the Cab Services through theapp-based – A study on Ola & Uber App Based Services,' Empirical Economics Letters, vol. 17, Special Issue 1, pp. 1–16, Jan. 2024.
3. W. Zhang and S. V. Ukkusuri, 'Share-a-Cab: Scalable Clustering Taxi Group Ride Stand From Huge Geolocation Data,' IEEE Access, vol. 9, pp. 9771–9776, 2021, doi: 10.1109/ACCESS.2021.3050299.
4. S. Namasudra and P. Sharma, 'Achieving a Decentralized and Secure Cab Sharing System Using Blockchain Technology,'IEEE Trans. Intell. Transp. Syst., 2022, doi: 10.1109/TITS.2022.3186361.
5. K. Luo and F. C. R. Spieksma, 'Approximation algorithms for car-sharing problems,' arXiv:2007.03057, Jul. 2020. [Online]. Available: https://arxiv.org/abs/2007.03057.
6. A. Manjunath, V. Raychoudhury, S. Saha, S. Kar, and A. Kamath, 'CARE-Share: A Cooperative and Adaptive Strategy forDistributed Taxi Ride Sharing,' IEEE Trans. Intell. Transp. Syst., 2021, doi: 10.1109/TITS.2021.3066439.
7. L. Shanmugavel, C. Parsuvanathan, and S. Nail, 'Perceptions of Share-Cabs as a Sustainable Transportation Mode: Stakeholder Perspectives and the Path to Standardization,' J. Balkan Tribological Assoc., vol. 29, no. 5, pp. 861–870, 2023.