Abijith Sankar 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Campus Rideshare

Abijith Sankar, Amina Rashad, Belvin Thomas Cherian, Elona Elsa Thomas, Dr. Abin T. Abraham
Department of Computer Applications Saintgits College of Engineering Kerala, India

Abstract- College campuses often experience transportation challenges such as overcrowded shuttles, limited affordable commute options, traffic congestion, and unpredictable travel timings. Traditional campus travel systems lack real-time ride availability, secure identity verification, and a structured process for students and faculty to share rides. The Campus RideShare System addresses these issues by providing a secure, web-based platform that connects verified campus members traveling along similar routes. Through institution-based login, users can offer or request rides, match with nearby commuters, and share live location information for a safer and more convenient travel experience. Developed as a modern web application using Firebase and Leaflet for geolocation and mapping, the system incorporates features such as real-time route tracking, intelligent distance-based ride matching, and a rating mechanism to ensure trust and safety. By optimizing campus travel resources, improving coordination, reducing travel costs, and lowering carbon emissions, the proposed solution enhances overall campus mobility while promoting sustainable transportation.Keywords— Ride-sharing, Campus Transportation, Real-time Tracking, Firebase, Geolocation, Mobile Application, Smart Mobility.

Keywords— The Campus RideShare System focuses on enhancing campus transportation through ride-sharing, real-time tracking, and smart mobility solutions. It integrates geolocation and mapping technologies using Firebase and Leaflet to enable efficient route matching, secure identity verification, and live location sharing.

I. INTRODUCTION

Efficient and safe transportation within college campuses is a growing concern due to overcrowding, rising fuel costs, and the lack of coordinated commuting options. Students and faculty often depend on personal vehicles or irregular public transport systems, resulting in increased traffic congestion, unmanaged parking, and high travel expenses. Informal ride-sharing practices exist but lack verification systems, structured matching, or accountability, raising safety and reliability issues.

The Campus RideShare System introduces a digital, campus- exclusive ride-sharing platform that provides safe, cost- effective, and eco-friendly commuting. The system verifies each user via institutional credentials, enabling them to create ride offers, request rides, and view available matches based on their location and destination. With real-time tracking, OTP verification, in-app chat, and user-rating mechanisms, it ensures trust and accountability among all participants.

Built using Firebase for real-time data management, Google Maps for navigation, and Android for the user interface, this system delivers an optimized transportation experience tailored specifically for educational institutions.

II. RELATED WORK

Ride-Sharing Mobile Applications: Commercial platforms like UberPOOL, BlaBlaCar, and Lyft Shared Rides provide large-scale ride-sharing solutions using route-matching algorithms and GPS-based navigation. While effective for public commuting, these applications lack campus-specific verification, cost control, and internal community safety features. Additionally, these services expose users to unknown co-riders, making them unsuitable for a closed academic environment. Their fare structures and operational models are also not optimized for short-distance, low-budget student travel.

Carpooling Systems for Smart Cities: Researchers have proposed smart city carpooling systems using IoT, RFID, and location-based services to reduce traffic and pollution. These systems highlight the importance of optimizing routes and vehicle occupancy but are not tailored for campus environments requiring user authentication and closed- community access. Furthermore, such solutions often need significant infrastructure investment that colleges may not be able to support. They also lack personalized ride-matching designed for small, frequent, localized travel routes.

Campus Mobility Apps: Many institutions use campus mobility apps for shuttle tracking, timetable updates, and basic commute notifications. While useful, they lack peer-to-peer ride-sharing features, personalized route matching, and secure driver-rider verification systems. Most of these apps focus on managing institutional vehicles rather than offering flexible transportation options. They also do not provide advanced safety mechanisms such as rider authentication or OTP-based trip validation.

Sustainable Transportation Models: Sustainable transportation research emphasizes carpooling, public transit, and energy- efficient mobility to reduce carbon emissions. These models improve environmental outcomes but often overlook the need for community-specific trust, identity verification, and real- time coordination. In addition, many sustainability projects are theoretical and lack practical mobile-based implementation. They also do not address dynamic ride-matching for short-distance, high-frequency campus commutes.

Smart Campus Transportation Systems: Smart campus transportation systems typically integrate IoT sensors, GPS tracking, and centralized control centers to manage on-campus vehicles. While effective for monitoring shuttles and predicting arrival times, they do not support flexible, student-generated ride-sharing. These solutions prioritize campus fleet management rather than peer-to-peer transportation. Moreover, they do not include identity-verified matching processes or user-driven ride creation features.

Location-Based Social Matching Systems: Location-based social matching systems pair users based on proximity, similar destinations, and shared activities. These systems demonstrate the power of real-time GPS data in connecting individuals but do not include structured safety layers or institutional verification. Additionally, they are designed for social networking, not transportation, making them unsuitable for secure ride-sharing. They also lack features such as seat availability, route mapping, and driver-rider coordination.

Secure Peer-to-Peer Transportation Apps: Singh et al. proposed a peer-to-peer commuting platform with encrypted communication and identity verification to ensure safer travel. Although effective for public communities, the system lacks institution-specific authentication and campus-restricted access controls. It also does not support features like OTP-based ride validation or academic-domain login verification. Such limitations make it less suitable for closed and trusted environments like college campuses.

Machine Learning in Ride-Matching: Studies on machine learning-based ride-matching explore clustering, prediction models, and similarity algorithms to match riders and drivers accurately. These approaches improve efficiency but often require large datasets and complex computations not feasible for smaller campus setups. They also need continuous data collection and processing to maintain accuracy. This makes them difficult to implement in environments with limited infrastructure and user volume.

Safety-Centric Carpool Systems: Ramirez et al. introduced a safety-enhanced ride-sharing model featuring SOS alerts, emergency contact notifications, and user rating mechanisms. While effective for improving trust, the system does not implement institution-based verification or OTP validation for ride initiation. It also lacks campus-specific restrictions needed to limit access to verified students or staff. As a result, it may not provide the controlled environment required in academic communities.

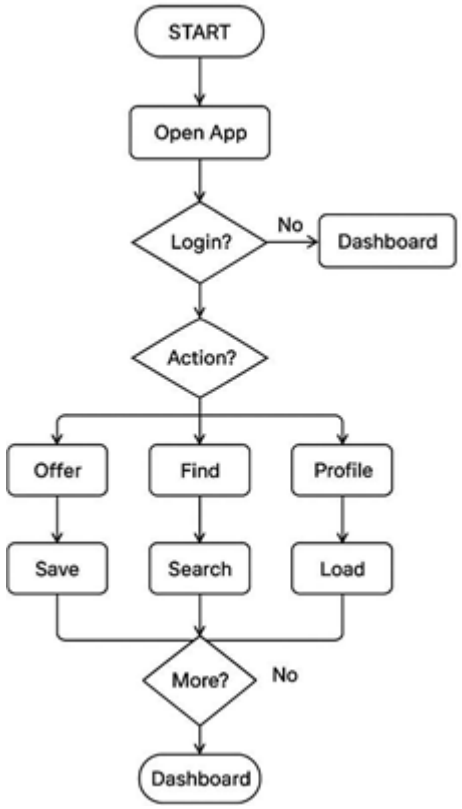
Real-Time Tracking Systems for Mobility Applications: Real- time GPS tracking systems focus on route visualization, accuracy improvement, and reducing latency in transportation services. These systems work well for public transport but do not address peer-to-peer ride coordination or driver-rider matching. They mainly monitor vehicle movement rather than facilitating ride-sharing participation. Additionally, they do not incorporate safety workflows or in-app communication mechanisms.

Shuttle Tracking Systems in Educational Institutions: Many universities use shuttle-tracking apps to help students monitor bus locations, predict arrival times, and manage crowd flow. While beneficial, these apps support only institutional vehicles and do not allow students to offer or request rides. Such systems lack flexibility and cannot address individual commute variations. Moreover, they do not integrate safety checks, verification processes, or ride-matching algorithms.

III. PROPOSED SYSTEM

The proposed Campus Ride Share System aims to create a secure, convenient, and cost-effective ride-sharing platform exclusively for students, faculty, and staff within a college campus. Unlike public ride-sharing services, this system operates entirely within a closed academic network, ensuring trusted interactions among verified users. The platform allows users to offer rides, request rides, and match with nearby drivers based on real-time location and destination preferences. By limiting access to authenticated campus members only, the system ensures a safer and more reliable commuting environment.

The system introduces a user-verification mechanism that requires login through institutional credentials such as college email IDs or student registration numbers. This feature ensures that only authorized campus members can create accounts and participate in ride-sharing activities. Additionally, the system generates a unique OTP (One-Time Password) for each ride, which both the rider and driver must confirm before starting the trip. This reduces the chances of impersonation and enhances overall user safety.



The proposed solution uses real-time GPS tracking to display available rides, live driver locations, estimated arrival times, and route details. This enables users to view nearby drivers heading in the same direction and book rides instantly. The system automatically updates ride status, seat availability, and route information to maintain transparency and avoid confusion. Real-time updates also help reduce waiting time and improve coordination between drivers and riders.

To enhance efficiency, the system integrates intelligent ride- matching logic that pairs riders with suitable drivers based on proximity, destination, departure time, and available seats. This ensures optimal vehicle utilization and reduces the number of individual trips, contributing to better traffic management on campus. Drivers can easily list rides by entering details such as starting point, destination, departure time, and available seats,

while riders can filter and select the most convenient option.

The platform additionally incorporates safety and communication features, including in-app chat, emergency SOS alerts, and post-ride rating systems to maintain trust among users. These features ensure that users can communicate effectively and report any concerns immediately. Ratings help maintain accountability and promote responsible behavior among participants. Overall, the proposed system provides a reliable, sustainable, and user-friendly solution tailored to the transportation needs of a modern academic campus.

A. Software and Hardware Requirements

Software Requirements: The software requirements for the Campus RideShare System include several technologies essential for frontend, backend, and mapping functionalities. HTML is used to structure the content of the web pages through elements such as headings, paragraphs, images, and links, while CSS manages the visual styling, layout, colors, and fonts to enhance user experience. JavaScript provides dynamic and interactive features such as form validation, content updates, and user interactions. Python serves as the backend programming language due to its readability, simplicity, and strong library support, and Flask is used as the lightweight framework to manage routing, templates, and server-side logic. Firebase provides cloud-based services including authentication and real-time database management, enabling secure and scalable backend operations. Additionally, the Leaflet JavaScript library is used for map rendering, allowing the application to display routes, markers, and interactive geographic elements. Together, these software components create a robust and efficient environment for developing and deploying the Campus RideShare System.

Hardware Requirements: The Campus RideShare System requires a computer equipped with an Intel Core i5 processor to ensure smooth development and execution of web- based and backend components. A minimum of 4 GB RAM is needed to support coding, testing, and running essential development tools, while 256 GB of storage is required to maintain project files, databases, and software dependencies without performance issues. These hardware specifications ensure that both the development environment and application services run efficiently throughout the project.

B. Functional Specifications

The functional specifications of the Campus RideShare System define the essential operations required to support secure, efficient, and user-friendly ride-sharing within a college environment. The system enables users to register using verified institutional credentials, log in securely, and maintain personalized profiles. Drivers can create ride listings by specifying pickup points, destinations, timing, and available seats, while riders can search, filter, and book rides based on real-time availability. The platform includes GPS-based location tracking to display nearby rides, provide estimated arrival times, and guide route navigation. An OTP-based ride confirmation process ensures that only the authenticated rider joins the assigned vehicle, enhancing safety. Additional core functions include in-app communication between driver and rider, cancellation options, ride history management, and a rating mechanism to maintain trust and accountability within the campus community.

C. System Architecture

The system architecture of the Campus RideShare System follows a simple three-layer structure designed to ensure smooth operation, clear data flow, and user-friendly interaction. The presentation layer provides the interface through which users register, log in, request rides, offer rides, and view available drivers using a web or mobile application. The application layer processes core operations such as user authentication, ride matching, ride creation, real-time location handling, and communication between users. The data layer manages storage and retrieval of ride details, user information, booking history, and location data using Firebase for secure cloud-based access. External services like Leaflet Map API and Firebase Authentication support map rendering, location tracking, and secure login,

ensuring reliability, scalability, and efficient performance for campus ride-sharing activities.

D. Database Design

The database design of the Campus Rideshare System is built around key entities such as Users, Rides, Bookings, and Payments, each storing essential details that support authentication, ride offering, and ride booking. These entities are connected through unique identifiers like email, ride_id, and booking_id, enabling one-to-many relationships between users and the rides or bookings they create. The design ensures data integrity, real-time updates, and secure access control using Firebase's structured collections and security rules.

E. Implementation Procedures

The implementation begins with setting up the development environment, integrating Firebase for authentication and real- time database operations, and building the Flask backend to handle routing, user actions, and ride management. The frontend interfaces are then developed using HTML, CSS, and JavaScript, followed by linking them with backend APIs to enable features like offering rides, booking rides, user profiles, and admin monitoring. Finally, all modules undergo iterative testing—including unit, integration, and UAT—to ensure smooth functionality before deployment.

IV. RESULTS AND DISCUSSION

A. Testing Methodology

The testing phase ensured the functionality, reliability, and security of the Campus Rideshare System using an incremental approach aligned with Agile methodology. Unit testing was performed on individual modules such as user authentication, ride posting, ride searching, booking functionality, and admin operations to verify that each component worked correctly. Integration testing validated the interaction between the Flask backend, Firebase Realtime Database, and the user interfaces, ensuring smooth data flow for operations such as offering a ride, booking a ride, and updating booking statuses. System testing evaluated the overall performance of the platform under realistic usage, verifying responsiveness, accuracy of real-time updates, and error handling.

B.Performance Results

Performance testing confirmed reliable functionality across all modules. Authentication and role-based access worked accurately, ensuring only verified users could log in, while admin access remained secure. Ride offering and ride booking operations performed as expected, with data consistently stored and updated in Firebase, including seat availability and booking records. Search operations returned precise ride matches based on pickup, drop, and date inputs. The admin dashboard displayed real-time data for rides, users, and bookings, with approve/cancel actions updating the database instantly. Ride history, earnings calculations, and user activity statistics were generated correctly, while system load testing showed stable performance across typical campus usage levels. Security testing validated that unauthorized access attempts were blocked and sensitive user information remained restricted.

Table 1
Performance Comparison of Visitor Management Systems

Criteria	Existing	Proposed	Cost	Performance
Speed	Slow	Fast	None	Better
Security	Low	High	None	Improved
Efficiency	Poor	Good	Low	Economical
Data	Manual	Digital	None	Accurate

C. System Advantages

The implemented rideshare system offers several advantages compared to manual ride coordination or external ride-sharing platforms. Automated ride posting and booking eliminate communication delays and manual coordination challenges. Secure authentication ensures that only verified college members participate, building trust and safety within the community. Real-time route matching and ride searches enhance convenience and reduce time spent finding suitable rides. Digital recordkeeping

provides complete, timestamped logs of rides, bookings, and cancellations, supporting transparency and easy reporting. The auto-update features for seats, booking status, and user profiles streamline user interactions and reduce repetitive data entry. The admin dashboard gives comprehensive oversight of users, rides, and system activity, enabling effective monitoring and decision-making.

D. Limitations and Challenges

Although the system operated successfully, some limitations were observed during testing. Real-time route matching relies on user-entered pickup and drop locations, and accuracy may vary depending on formatting or spelling. Since the system uses Firebase's free-tier services, large-scale usage may require upgraded plans or more robust backend solutions. The current version does not include features such as in-app navigation, automated fare calculation beyond fixed user input, or vehicle verification. Internet connectivity remains essential for accessing Firebase services, making the application dependent on stable network conditions. Additionally, integration with institutional services—such as campus ID systems or transport management—has not yet been implemented but may be important for broader deployment.

E. Scalability Considerations

Although designed with scalability in mind, expanding from a single college deployment to larger institutions requires technical enhancements. Heavier usage may require migrations from basic Firebase configurations to more advanced real-time database plans or Firestore for improved performance and indexing. Increased traffic would require optimized backend APIs, caching strategies, and potentially load balancing for the Flask server. Multi-institution deployment would require multi-tenant database structures and stricter access policies to isolate user groups. Despite these challenges, the modular architecture and use of widely adopted technologies make future scaling feasible.

F. Future Enhancement Opportunities

Several improvements could further enhance system performance and user experience. Integrating Google Maps or a navigation API could provide real-time route guidance, distance calculation, and dynamic ride matching. Introducing a rating and feedback system would help ensure safety and quality of service. Adding digital payment gateways (UPI, card, wallets) would eliminate manual payment handling. Developing a mobile application for Android/iOS would make the platform more accessible and user friendly. Advanced features such as AI-based ride suggestions, carbon-saving analytics, and real-time GPS tracking could significantly improve intelligence and user engagement. Offline support and deeper integration with institutional systems could further improve reliability and operational convenience.

V. CONCLUSION

The Campus Rideshare System successfully demonstrates how technology can be used to solve transportation challenges within a campus environment. By enabling students and staff to offer and book rides through a secure, real-time platform, the system reduces traffic congestion, improves commuting convenience, and promotes sustainable mobility. The integration of Firebase for authentication and data storage, together with a Flask backend and a user-friendly frontend, ensures smooth functionality across all modules.

The system meets all major project objectives, including secure login, verified user access, reliable ride posting, accurate ride searching, and seamless booking processes. Real-time updates allow users to view available rides instantly, while the admin dashboard ensures effective oversight of rides, users, and booking activities. Comprehensive testing—unit, integration, system, and UAT—validated the performance, accuracy, and reliability of each component under real-world conditions.

While the system is highly functional, certain limitations were identified. The platform depends on stable internet connectivity for database operations, and the current version offers only basic ride

matching without advanced features like real-time tracking or automated route optimization. Additionally, its scalability is limited by Firebase's free-tier constraints, and integration with external systems such as campus transport services has not yet been implemented.

Despite these constraints, the system provides a strong foundation for future improvements. Features such as AI-based ride recommendations, digital payment integration, enhanced security mechanisms, real-time GPS tracking, and mobile application support can significantly expand its capabilities. Overall, the Campus Rideshare System delivers an efficient, secure, and eco-friendly solution that enhances transportation within the campus and sets the stage for advanced, scalable development in the future.

REFERENCES

1. Pressman, R. S. & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach (9th ed.). McGraw-Hill Education.
2. Sommerville, I. (2016). Software Engineering (10th ed.). Pearson Education.
3. Fowler, M. (2004). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.). Addison- Wesley.
4. IEEE Standard for Software Test Documentation, IEEE Std 829-2008.
5. IEEE Standard for Software Verification and Validation,
   IEEE Std 1012-2016.
6. FirebaseDocumentation-
   https://firebase.google.com/docs
7. Flutter Official Documentation – https://flutter.dev/docs
8. W3Schools HTML, CSS, and JavaScript Tutorials – https://www.w3schools.com
9. Python Flask Framework Documentation – https://flask.palletsprojects.com
10. Google Maps Platform Documentation – https://developers.google.com/maps
11. Atlassian Agile Coach – Agile Methodology Overview – https://www.atlassian.com/agile
12. Sharma, P. & Gupta, S. (2019). "Smart Ride Sharing Application for College Students." International Journal of Computer Applications (IJCA), Vol. 182, No. 38, pp. 10–14.
13. Singh, R. & Kaur, P. (2020). "An Efficient Carpooling System Using Real-Time Database." International Journal of Scientific & Engineering Research, Vol. 11, No. 5.
14. Kaur, J. & Verma, A. (2021). "Design and Implementation of a Campus Carpooling Application." International Research Journal of Engineering and Technology (IRJET), Vol. 8, Issue 6.
15. ISO/IEC 25010:2011 – Systems and Software Quality Models.
16. GitHub Documentation – https://docs.github.com
17. Stack Overflow Developer Community – https://stackoverflow.com
18. Google Cloud Platform Documentation – https://cloud.google.com/docs
19. MDN Web Docs – HTML, CSS, and JavaScript Reference – https://developer.mozilla.org
20. Hussain, M. & Ahmed, Z. (2022). "Optimized Ride- Sharing Systems for Smart Cities." Journal of Advanced Research in Computer Science and Software Engineering, Vol. 12, Issue 3.
21. Khan, A. & Patel, R. (2020). "A Study on Carpooling Apps for Sustainable Transportation." International Journal of Innovative Research in Technology (IJIRT), Vol. 7, Issue 12.
22. FlutterFire Documentation – https://firebase.flutter.d