International Journal of Information Technology & Computer Engineering

ISSN 2347–3657
Volume 14, Issue 2, 2026

Journey Connect

D Navaneetha[1],M Sanjana[2], L Shruthi[3],K Snehitha Guna[4],
[1]Associate Professor;Department Of Information Technology Bhoj Reddy Engineering College For Women Hyderabad India
[2,3,4]B.Tech Students; Department Of Information Technology Bhoj Reddy Engineering College For Women Hyderabad India
Mail Id; sanjana102304@gmail.com[2],snehithakaleru@gmail.com[4],lingannapetashruthishruthi@gmail.com[3]

Abstract
Urban transportation systems continue to face major challenges such as traffic congestion, increasing fuel prices, limited parking availability, and environmental pollution. Ride-sharing has emerged as an effective solution to improve transport efficiency while reducing travel costs and carbon emissions. This paper presents Journey Connect, a smart ride-sharing platform designed to connect passengers and drivers through a secure and user-friendly digital environment. The proposed system enables users to register, authenticate, and either request or provide rides according to their travel requirements. Key functionalities include source and destination selection, vehicle preference settings, ride scheduling, booking management, and real-time communication between users.The platform is developed to optimize vehicle occupancy by matching users with similar travel routes, thereby minimizing the number of single-occupancy vehicles on roads. Journey Connect also enhances convenience through instant notifications, route coordination, and transparent ride details. In addition, the system contributes to sustainable urban mobility by lowering fuel consumption and reducing greenhouse gas emissions. Experimental evaluation and functional testing indicate that the platform improves ride accessibility, decreases commuting expenses, and supports efficient transport resource utilization. The proposed solution demonstrates how intelligent ride-sharing applications can play a significant role in modern smart city ecosystems.

Keywords
Ride-Sharing, Smart Transportation, Carpooling, Urban Mobility, Sustainable Transport, Vehicle Optimization, Real-Time Booking, Smart City, Cost Reduction, Journey Connect

Introduction

Rapid urbanization has significantly increased the demand for reliable, affordable, and efficient transportation systems. Growing populations in metropolitan areas have intensified problems such as traffic congestion, higher fuel consumption, limited parking space, and longer commuting times. At the same time, environmental issues including air pollution and greenhouse gas emissions have created an urgent need for sustainable mobility solutions. Conventional transport methods often depend on single-occupancy vehicles, resulting in underutilized seating capacity, unnecessary fuel usage, and increased road congestion. These challenges highlight the importance of adopting smarter transportation models that optimize existing resources while improving commuter convenience.Ride-sharing has emerged as an effective solution for addressing many of these urban mobility concerns. By allowing multiple passengers traveling in similar directions to share a single vehicle, ride-sharing reduces the number of vehicles on the road, lowers travel expenses, and decreases environmental impact. Shared mobility also promotes better utilization of vehicle capacity and contributes to reduced traffic density in busy cities. In addition to economic and environmental benefits, ride-sharing provides flexibility for users who seek convenient alternatives to public transport or private vehicle ownership. However, many currently available platforms do not fully address user comfort, safety, and localized travel preferences, which can limit their adoption among daily commuters.To overcome these limitations, this paper presents Journey Connect, an Android-based ride-sharing application designed to create a secure, efficient, and user-friendly travel ecosystem for urban commuters. The platform enables users to register and authenticate securely, after which they can either offer rides as drivers or book rides as passengers. Drivers can publish travel details such as source, destination, schedule, available seats, and vehicle type, while passengers can search and reserve rides that match their travel needs. The system also includes location-based matching to improve route efficiency and minimize waiting time.A distinguishing feature of Journey Connect is its focus on user-centric customization and safety. The application provides options such as rider gender preference selection, helping users feel more comfortable and secure during travel. Real-time ride management, transparent booking details, and communication features further enhance trust between drivers and passengers. By integrating convenience with security-focused functionality, the platform encourages wider participation in shared mobility services.

Related Work

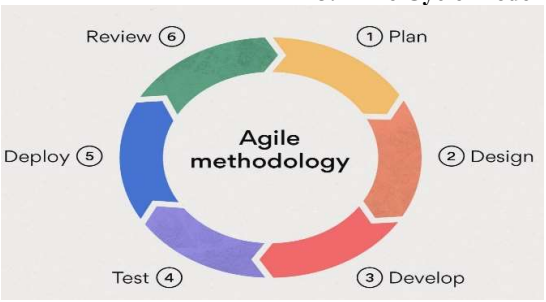
Literature Survey

The rapid growth of urbanization and population has placed significant pressure on existing transportation systems, leading to increased traffic congestion, higher fuel consumption, and rising environmental pollution. As cities expand, the reliance on private vehicles has increased, resulting in inefficient use of road space and vehicle capacity. This has created a strong demand for sustainable and intelligent transportation solutions that can improve mobility while reducing operational costs and environmental impact. Ride-sharing has emerged as one of the most effective approaches to address these challenges by enabling multiple users traveling in similar directions to share a single vehicle. This not only reduces the number of vehicles on the road but also improves fuel efficiency and lowers travel expenses for users.Existing commercial platforms such as Uber and Lyft have introduced advanced ride-hailing services that allow users to book rides conveniently through mobile applications. These systems provide features such as real-time tracking, digital payments, and route navigation. However, they primarily focus on individual ride bookings rather than shared mobility, which limits their effectiveness in reducing traffic congestion and environmental impact. Another widely known platform, BlaBlaCar, focuses mainly on long-distance travel, making it less suitable for daily urban commuting needs.Several research studies have highlighted the importance of intelligent ride-sharing systems in improving transportation efficiency. Agatz et al. (2012) demonstrated that dynamic ride-matching algorithms can significantly optimize shared mobility by considering route similarity, timing, and user constraints. Similarly, Shaheen et al. (2016) emphasized that shared mobility systems contribute to reduced traffic congestion and lower greenhouse gas emissions. Cici et al. (2014) further showed that mobile and social data can be effectively utilized to enhance ride-sharing performance through better demand prediction and matching strategies

Requirement Analysis

The requirement analysis phase defines the functional and non-functional needs of the proposed system, ensuring that the Journey Connect application effectively addresses user expectations and system objectives. The system is designed as a mobile-based ride-sharing platform that connects drivers and passengers for shared travel. It allows users to either offer rides or book rides based on their travel requirements, ensuring efficient utilization of vehicle capacity and reducing travel costs. The system also incorporates additional features such as source and destination selection, ride scheduling, and real-time ride management to improve usability and convenience.The functional requirements of the system include user registration and authentication, which ensures that only valid users can access the platform. Registered users can securely log in and access ride-sharing services. Drivers can post ride details such as source, destination, available seats, vehicle type, and route information. Passengers can search for available rides based on their travel preferences and book suitable options. The system also supports ride management features where drivers can update ride status, and passengers can track ride availability. Additionally, a key feature of the system is gender-based ride preference selection, which enhances user comfort and safety during travel.The non-functional requirements focus on system performance, scalability, usability, availability, and sustainability. The application must handle multiple users simultaneously without performance degradation. It should be scalable to support a growing number of users and ride requests. The user interface must be simple, intuitive, and easy to navigate for both drivers and passengers. High availability is essential to ensure uninterrupted access to services. Furthermore, the system promotes environmental sustainability by encouraging carpooling, which helps reduce fuel consumption and carbon emissions.

3.4 Life Cycle Model



Cycle Model

The development of the Journey Connect application follows a structured software development life cycle to ensure systematic planning, design, implementation, testing, and deployment. In the planning phase, the core idea of developing a secure and efficient ride-sharing platform was identified. Key requirements such as ride creation, ride booking, gender preferences, and map-based location selection were analyzed, and a project timeline was prepared.In the design phase, system architecture and user interface designs were created to ensure simplicity and usability.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026

Wireframes were developed for major screens, including login, home dashboard, ride posting, and ride searching interfaces. The database structure was designed using Supabase to manage user data, ride information, and booking details efficiently.During the development phase, the application was implemented using React Native for the frontend and Supabase for backend services. Core functionalities such as user authentication, ride posting, ride searching, and map integration were developed. Google Maps API was integrated to enable accurate location selection and route visualization.The testing phase involved verifying each module individually as well as in integration. Functional testing ensured that login, ride creation, and booking features worked correctly. User interface testing was performed to ensure smooth navigation and responsiveness across different devices. Bugs and errors identified during testing were resolved to improve system stability.

Design

Architecture

The architecture of the Journey Connect system is designed to support a seamless interaction between users, the mobile application, and backend services. The system follows a client–server model where the React Native mobile application acts as the client interface, and Supabase serves as the backend platform providing database, authentication, and real-time services. This layered architecture ensures modularity, scalability, and efficient communication between components.

Software Architecture

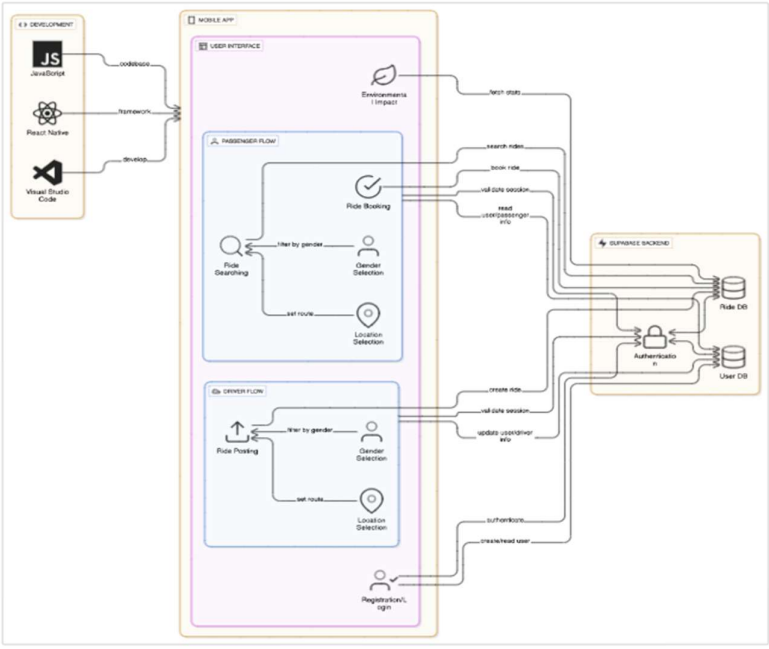


Fig 1 Software Architecture

The software architecture of Journey Connect defines the overall structure of the mobile application and its interaction with backend services. The application is developed using React Native with JavaScript, enabling cross-platform support for both Android and iOS devices. Visual Studio Code is used as the primary development environment for coding and testing.The system is divided into two main user flows: passenger and driver. The passenger flow includes functionalities such as ride search, ride booking, location selection, and optional gender-based filtering to enhance safety and comfort. Users can search for available rides based on source and destination and proceed with booking after reviewing ride details.The driver flow includes ride creation and management functionalities. Drivers can post ride details such as route, vehicle type, available seats, and travel schedule. They can also define intermediate stops along the route, which are visible to passengers during ride selection. Additionally, drivers can specify their gender to support filtering options for passengers.User authentication is integrated as a foundational module, allowing secure registration and login for all users before accessing system features. The overall architecture ensures smooth interaction between frontend components and backend services while maintaining data consistency and user experience.

Technical Architecture

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026

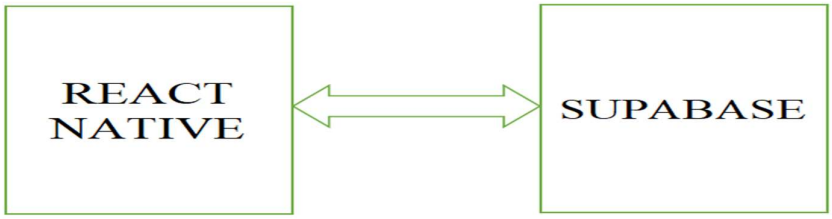


Fig 2 Technical Architecture

The technical architecture of the system is based on a client–server model where the React Native mobile application communicates directly with Supabase, which acts as a Backend-as-a-Service (BaaS). This eliminates the need for a separate custom backend server and simplifies system deployment and maintenance.The frontend layer is built using React Native, which handles user interactions such as ride search, booking requests, authentication, and ride creation. It communicates with Supabase using the Supabase SDK to perform operations such as fetching ride data, inserting bookings, and managing user authentication.Supabase provides core backend functionalities including a PostgreSQL database, authentication services, real-time data synchronization, and secure storage. It manages user records, ride details, and booking transactions. Security is enforced through row-level security policies, ensuring users can only access authorized data.The communication flow between the frontend and backend includes operations such as selecting rides (SELECT queries), booking rides (INSERT operations), and user authentication (sign-up and sign-in methods). Supabase also handles business logic through database functions, triggers, and security policies, ensuring efficient and secure system operation.

Implementation

The implementation of the Journey Connect system focuses on developing a cross-platform mobile application using modern web and mobile technologies. The system is designed to ensure scalability, maintainability, and efficient performance while providing a smooth user experience for both drivers and passengers.

Technologies Used

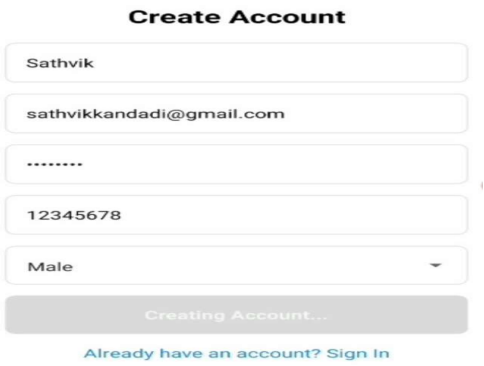
The application is developed using React Native, an open-source framework that enables cross-platform mobile development using JavaScript. It allows a single codebase to be used for both Android and iOS applications, significantly reducing development effort and ensuring consistent performance across devices. React Native also provides reusable components, hot reloading, and native-like performance, making it suitable for real-time ride-sharing applications.Supabase is used as the backend platform, providing essential services such as authentication, database management, real-time updates, and secure storage. It is built on PostgreSQL and offers a scalable backend infrastructure without requiring a separate server setup. Supabase ensures secure data handling using row-level security policies and supports real-time synchronization for ride updates and bookings.The development environment includes JavaScript as the primary programming language and Visual Studio Code as the code editor. VS Code provides useful extensions such as ESLint for code quality, Prettier for formatting, and React Native tools for debugging. Git and GitHub are used for version control, enabling collaborative development and efficient code management through branching strategies.

Implementation Overview (Code Logic)

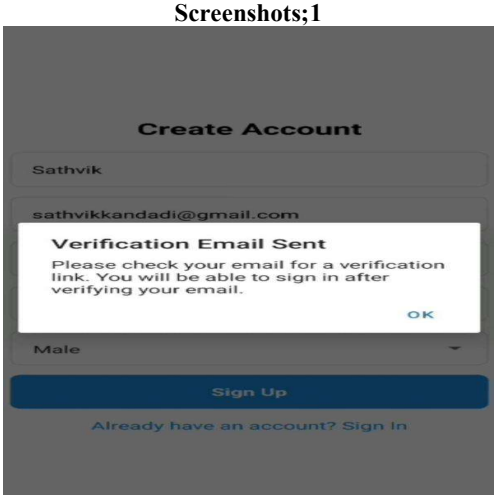
The system includes multiple functional modules such as user profile management, ride creation, and ride booking. The profile module retrieves authenticated user details such as email and user ID and displays them in a structured format.The “Give a Ride” module allows drivers to create ride entries by selecting vehicle type, available seats, fare, source, and destination using map integration. The system validates all inputs before storing ride data in the Supabase database.The “Take a Ride” module enables passengers to view available rides, select a preferred ride, and proceed with booking. It includes validation checks for ride availability, user authentication, and seat capacity. Once a booking is confirmed, the system updates ride availability and stores booking records securely in the database.Utility functions support map interactions, data fetching, and real-time updates, ensuring smooth system performance. The overall implementation ensures that both driver and passenger workflows operate efficiently with minimal delay and high reliability.

Screenshots

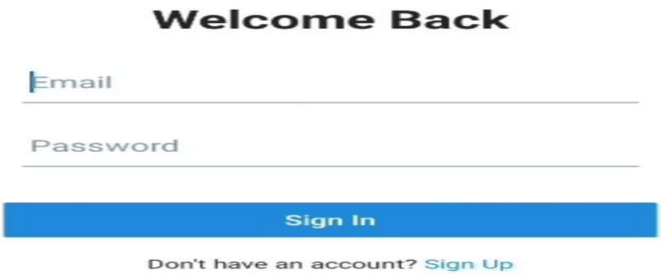
International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026



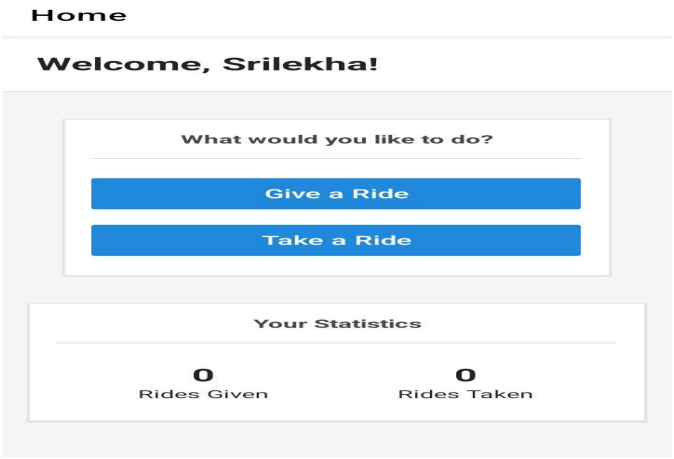
Screenshots;1



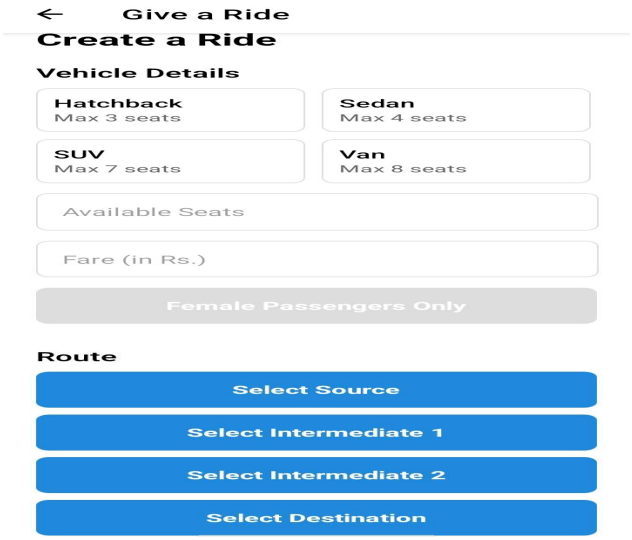
Screenshots;2



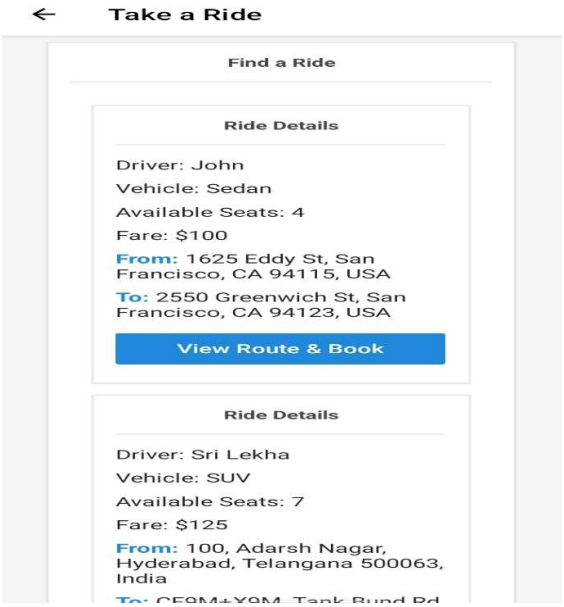
Screenshots;3



Screenshots;4

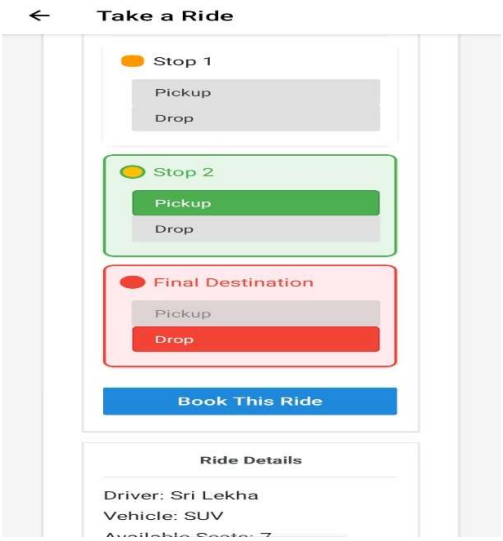


Screenshots;5



Screenshots;6

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026



Screenshots;7

Testing and Validation

Overview of Testing

The testing phase of the Journey Connect system ensures that the application functions correctly, meets user requirements, and performs efficiently under different conditions. A combination of manual testing and structured validation techniques is applied to verify the correctness, reliability, and usability of the system. Testing is carried out at different levels, including functional, performance, and positive testing, to ensure comprehensive evaluation of the application.Functional testing is performed to verify that all core features of the system operate as intended. This includes user authentication, where the system is tested for secure registration, login, and logout processes with proper session handling. The ride posting functionality is validated by ensuring that drivers can successfully create ride entries with details such as source, destination, intermediate stops, vehicle type, and gender preferences, and that this data is accurately stored in the backend database. Similarly, ride searching is tested to confirm that passengers can filter and find rides based on route preferences and optional constraints such as gender selection. Map integration is also tested to ensure that users can correctly select locations, view routes, and interact with map-based inputs without errors. Additionally, booking confirmation is validated to ensure that once a ride is selected, the system updates the booking status and provides proper confirmation to the user.Performance testing is conducted to evaluate the responsiveness and stability of the application under different usage conditions. Load testing ensures that the system can handle multiple users performing ride searches, postings, and bookings simultaneously without performance degradation. Response time is measured for key operations such as login, ride search, and ride creation to ensure quick system feedback. Map rendering performance is also evaluated to confirm smooth loading and navigation across different devices, ensuring that real-time location updates and route visualization function efficiently without delays or crashes.

Test Cases

The test cases for the Journey Connect system are designed to validate each functional module of the application, including authentication, ride creation, ride search, booking, and map-based interactions. Each test case checks whether the system produces the expected output for given input conditions. For example, valid user credentials should successfully allow login, while invalid credentials should trigger appropriate error messages. Similarly, when a driver posts a ride with complete details, the system should store the information correctly in the database and make it available for passengers to search.In ride booking scenarios, the system is tested to ensure that when a passenger selects a valid ride and confirms booking, the available seat count is updated correctly and a confirmation message is generated. Map-related test cases validate whether users can successfully select source and destination points and whether the system accurately displays routes between selected locations. Additional test cases ensure that filtering options such as gender-based ride selection work as expected and return correct results.

Conclusion

The Journey Connect ride-sharing application successfully provides a convenient, secure, and efficient platform for urban transportation. The system enables users to either offer rides or book available rides, thereby improving vehicle

utilization and reducing overall travel costs. By integrating features such as user authentication, ride management, location-based search, and gender preference options, the application enhances both usability and user safety.

Future Scope

The Journey Connect system can be further enhanced with advanced features to improve functionality, scalability, and user experience. One important enhancement is the integration of real-time GPS tracking, which would allow passengers to monitor the live location of drivers during the ride. This would significantly improve transparency and user trust.Another potential improvement is the integration of secure in-app payment gateways, enabling cashless transactions and digital wallet support for seamless fare payments. A rating and review system can also be introduced to allow users to evaluate drivers and passengers, thereby improving accountability and service quality within the platform.The system can be further expanded by implementing AI-based ride-matching algorithms that suggest optimal ride-sharing options based on route similarity, timing, and user preferences. Push notification services can also be added to provide real-time updates regarding ride status, booking confirmations, cancellations, and driver arrivals.Additional enhancements may include ride history tracking, analytics dashboards for users to view cost savings and environmental impact, and support for multi-stop routes to increase flexibility. Expanding user roles to include administrators or moderators can further improve system management and security. Accessibility features such as voice assistance and screen reader compatibility can also be introduced to make the application more inclusive.

References

[1] Agatz, N. A., Erera, A. L., Savelsbergh, M. W., & Wang, X. (2012). Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2), 295–303.

[2] Shaheen, S., Chan, N., & Gaynor, T. (2016). Casual carpooling and ride-sharing: A review of the literature. University of California Transportation Center.

[3] Cici, B., Markopoulou, A., & Frias-Martinez, E. (2014). Assessing the potential of ride-sharing using mobile and social data: A study across multiple cities. *ACM International Conference on Embedded Networked Sensor Systems (SenSys)*.

[4] Google Maps Platform Documentation. Official documentation for integrating maps, navigation, and real-time location services in mobile applications.

[5] Android Developers Documentation. Official guidelines for Android application development, including UI design, navigation, and system architecture best practices.

[6] Gurumurthy, K. M., & Kockelman, K. M. (2016). Analyzing dynamic ride-sharing potential for shared autonomous vehicle systems using agent-based simulation. *Transportation Research Record.*