Optimizing Driver-Rider Matching In a Cab Management System :A Flutter and Firebase Implementation

Satyam Mishra[1], Aniket Nangare[2] ,Monika Meshram[3] , Prof. Deepali Patil[4]

Computer Engineering Department[1,2,3,4]
Nutan Maharashtra Institute of Engineering and Technology, Pune, Maharashtra[1,2,3,4]

Abstract**—**In today's dynamic urban environments, efficient cab management systems are crucial for seamless transportation and passenger satisfaction. This paper presents the design and implementation of a cab management system built with Flutter for a mobile frontend and Firebase for a robust backend. Our primary focus lies on optimizing the driver-rider matching process to ensure timely cab allocation and minimize wait times. The paper details the system architecture, outlining how Flutter's cross-platform capabilities create a user-friendly mobile application for both riders and drivers. Firebase's real-time functionality is leveraged to facilitate efficient communication and data exchange between app users and the backend system. We delve into the core aspects of our driver-rider matching algorithm, explaining how it considers various factors to optimize connections. This may include factors like driver location, rider destination, and real- time traffic conditions (if implemented). By implementing this cab management system, we aim to demonstrate the effectiveness of Flutter and Firebase in building a scalable and optimized solution for matching cab drivers with riders. The paper concludes by discussing the potential benefits of this system for transportation service providers and riders alike, emphasizing improved efficiency and user satisfaction.

Keywords—Cab management, Flutter, firebase,Driver-rider matching, Optimization, Real-time communication,alability, Efficiency, User satisfaction, Transportation

I. INTRODUCTION

The development of an Online Cab Booking Management System aims to streamline the cab hiring process, benefiting both customers and cab booking agencies. This system facilitates effortless online cab reservations, tracking of bookings, and viewing of available cabs. Designed for convenience, it enhances the overall customer experience within the vehicle rental industry. The widespread reliance on cab services for daily transportation makes this a valuable solution for optimizing the sector.

The Cab Management App project seeks to overcome the limitations of traditional cab services by offering a more streamlined, technology-driven solution. Its primary focus is on providing passengers with convenience, real-time information, and empowering drivers with the tools needed to make informed decisions. This project recognizes the importance of transportation in modern life and strives to improve the overall experience of cab services, creating a more efficient and connected urban environment. It's driven by the understanding that technology can transform traditional services, making them more responsive to the needs of both users and service providers.

The Cab Management App project has several key objectives aimed at delivering a comprehensive transportation solution. A primary goal is creating a user-friendly mobile app that allows for easy ride bookings, location selection, and vehicle preferences, enhancing customer experience through seamless interactions. Additionally, the project focuses on providing drivers with a dedicated app for managing ride requests, navigation, and communication, facilitating efficiency and potentially improving job satisfaction.

Furthermore, the implementation of sophisticated algorithms (such as Dynamic Ride Matching, Dijkstra's, and A*) is integral for optimizing routes, reducing passenger wait times, and making driver routes more efficient. Ensuring robust security measures is also crucial, prioritizing the protection of user data and fostering trust in the platform through privacy and reliability. In essence, the Cab Management App project aims to build a modern, customer-focused transportation solution that benefits both passengers and drivers while harnessing the power of technology and optimization algorithms.

II. LITERATURE SURVEY

The rise of on-demand mobility solutions has transformed urban transportation landscapes. Ride-sharing services offer convenient, affordable, and potentially more sustainable alternatives to traditional taxis and personal vehicle ownership. At the heart of these services lies the complex task of optimal driver-rider matching. Efficient matching algorithms, robust real-time location tracking, and secure user experiences are paramount for success. This literature survey delves into the key research advancements in driver-rider matching optimization, platform development with Flutter, and the use of Firebase for real-time data management within the ride-sharing domain.

Effectively matching drivers and riders lies at the core of any ride-sharing platform. Researchers have extensively explored various optimization techniques to enhance matching efficiency, minimizing wait times, and maximizing system throughput. Tong et al. propose a real-time, dynamic dial-a-ride (DARP) algorithm that considers the joint matching of trips and vehicles [1]. This contrasts with traditional approaches that often perform trip and vehicle assignment sequentially, potentially leading to suboptimal outcomes. The complexity of the driver-rider matching problem necessitates the exploration of powerful optimization techniques. Bei and Zhang directly address the trip-vehicle assignment challenge, applying methods traditionally used for solving the Vehicle Routing Problem (VRP) in the context of ride-sharing [11]. Beyond efficiency metrics, Ma et al. consider the uncertainty inherent in ride-sharing scenarios, designing algorithms that aim for matching reliability in the face of fluctuating service requests and variable driver availability [12]. Addressing social aspects, Tewari et al. investigate ways to incorporate user preferences and social factors into the matching process, moving beyond purely geographic and time-based optimization [13].

Cross-platform development tools have become essential for ride-sharing companies seeking to deploy apps across multiple operating systems. Flutter, a framework by Google, has emerged as a popular choice due to its efficient code reusability and rapid development capabilities. Mishra et al. discuss the design and implementation of mobile apps with Flutter, highlighting its strengths in building visually appealing and responsive user interfaces [5]. While Flutter offers numerous advantages, it's crucial to address its challenges and optimize app performance. Attia et al. dissect common pitfalls and best practices when working with Flutter, emphasizing strategies for efficient rendering and state management [6].

The effectiveness of ride-sharing apps hinges on their ability to manage Firebase, a popular cloud-based Backend-as-a-Service (BaaS) from Google, which is widely used to build scalable ride-sharing platforms. Pandey et al. investigate strategies for scaling real-time mobile apps with Firebase, particularly focusing on its NoSQL database (Realtime Database) and push notification capabilities. Firebase's built-in synchronization and offline support facilitate the development of responsive and dependable mobile apps for ride-sharing services[7].
Ride-sharing platforms collect and manage sensitive user data, such as location, ride histories, and payment information. Ensuring strong security and data privacy measures is critical to building trust and protecting user information. Liu et al. examine security considerations for real-time applications built on Firebase, delving into best practices for data encryption and access control mechanisms [8]. Privacy-preserving techniques have also garnered attention within the ride-sharing space. Ma et al. propose a matching scheme that protects sensitive location data while facilitating efficient ride-sharing connections [9]. The implementation of such techniques addresses rising user concerns surrounding data privacy. Additionally, strict authentication and authorization controls are crucial. Butun et al. provide a detailed investigation of authentication and authorization mechanisms in the context of mobile applications [10].

While ride-sharing optimization has been the focus of substantial research, several areas offer opportunities for further exploration:

Multi-Modal Integration: Seamlessly incorporating ride-sharing with public transit systems can promote more sustainable transportation networks [14]. Stiglic et al. model and analyze the potential benefits and challenges associated with such integration. Developing optimization algorithms and interfaces that effectively bridge these forms of transportation is a promising research direction.
Carpooling and Ride-Sharing: Ride-sharing platforms could benefit from sophisticated algorithms specifically designed for carpooling scenarios [12, 13]. Matching multiple riders traveling along similar routes presents unique optimization challenges.
Incentive Mechanisms: Research into designing gamification features or dynamic pricing models that encourage ride-sharing during peak hours or in less-serviced areas could improve system performance. Wang et al. examine the concept of stable matching in dynamic ride-sharing, analyzing how pricing schemes can influence user and driver behaviors to optimize system-wide outcomes [15].

III. METHODOLOGY

The system architecture of the Cab Management App follows a modular design, divided into three distinct layers: the Presentation Layer, the Application Layer, and the Persistence Layer.

The Presentation Layer comprises the User Mobile App and the Driver Mobile App, developed using Flutter or a similar cross-platform framework. These apps interact with the Application Layer through RESTful APIs, enabling communication between the front-end and back-end components.

The Application Layer consists of the Ride Booking Module, which handles ride requests from users and manages the booking process. The Driver module is responsible for assigning rides to available drivers and facilitating parameters: The effectiveness of metaheuristics hinges on the proper tuning of parameters, which can require experimentation and domain knowledge.
Convergence speed: Metaheuristics may not always guarantee finding the absolute optimal solution within a reasonable time frame, especially for large-scale problems.
Potential Enhancements:

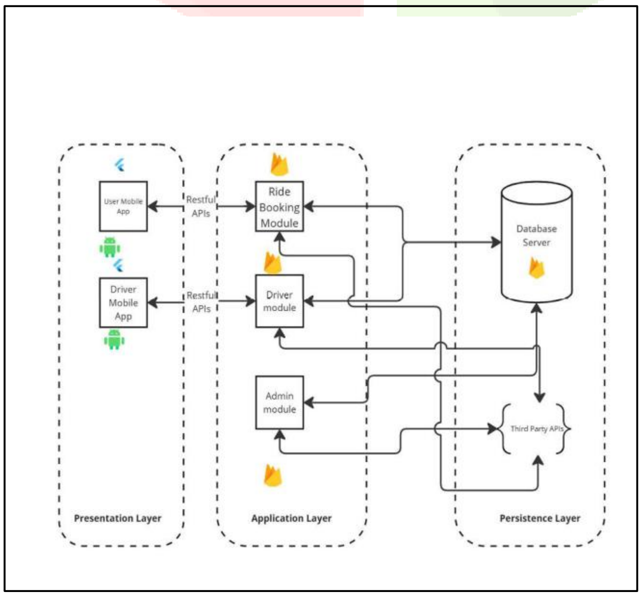
Hybrid metaheuristics: Combining different metaheuristics (ex: genetic algorithm with local search) can exploit the strengths of multiple methods.
Parallel computing: Metaheuristics lend themselves to parallel implementation, potentially speeding up the optimization process and enabling handling of larger-scale problems.
3. Traffic-Aware Route Optimization Algorithms
Concept: Traditional matching algorithms often rely on idealized distance or travel time estimations. Traffic-aware optimization algorithms aim to provide more realistic matches and routes by incorporating real-time traffic information.

Traffic Data Integration: These algorithms leverage real-time traffic information from sources like Google Maps, Waze, or other traffic monitoring systems. This data can be used to dynamically update travel time estimations based on traffic congestion, road incidents, or other factors.
Route Optimization: When calculating potential matches, rather than relying on static shortest-distance paths, traffic-aware algorithms attempt to identify routes that minimize travel time considering the current traffic situation. These algorithms often integrate with route-planning services that factor in real-time traffic.
Dynamic Rerouting: Even after a rider is matched with a driver, these algorithms can continue to monitor traffic conditions. If significant congestion develops along the initial planned route, the system can proactively suggest alternative routes to the driver to minimize delays and improve the rider's experience.
Strengths:

Realistic matching: Provides a more accurate picture of travel times, improving rider experience and reducing unexpected wait times.
Enhanced efficiency: Helps identify routes that avoid traffic jams, potentially leading to faster trips and improved utilization of vehicles.
Limitations:

Reliance on traffic data: The accuracy of traffic-aware optimization hinges on the availability and reliability of real-time traffic data.
Computation overhead: Incorporating real-time traffic data and dynamic rerouting can increase the computational complexity of the matching process.
Potential Enhancements:

Predictive modelling: Leverage machine learning techniques to forecast traffic patterns and integrate these forecasts into route planning for further improvement.



Multi-modal routing: The combination of ride-sharing with public transit options could be considered for more efficient routing, especially within congested areas.ting driver-rider matching. Additionally, the Admin module provides administrative functionalities for managing the system.

The Persistence Layer includes the Database Server, which serves as the central data storage component, likely utilizing a database management system like Firebase Realtime Database or a similar solution. The system integrates with Third-Party APIs, which could include services like Google Maps for location tracking, navigation, and potentially payment gateways for processing ride payments.

The architecture separates concerns and responsibilities into distinct components. The Presentation Layer handles the user interface and user interactions, while the Application Layer manages the core business logic and functionality. The Persistence Layer is responsible for data storage and integration with external services.

This modular design allows for scalability, maintainability, and flexibility, as each layer can be developed, tested, and deployed independently. The use of RESTful APIs facilitates seamless integration and data exchange between the different layers and components.

Overall, the architecture provides a solid foundation for the Cab Management App, enabling efficient ride booking, driver-rider matching, and system administration while leveraging the capabilities of third-party services and a robust database solution.

The system architecture follows a layered approach, promoting separation of concerns and enabling scalability. Each layer communicates with the others through well-defined interfaces, ensuring loose coupling and maintainability.

The Presentation Layer, consisting of the User Mobile App and Driver Mobile App, is responsible for providing a user-friendly interface for passengers and drivers. These apps are developed using Flutter, a cross-platform framework that allows for code reusability and rapid development across multiple platforms, including iOS and Android.

The Application Layer houses the core functionality of the Cab Management App. The Ride Booking Module acts as the central hub for handling ride requests, matching users with available drivers, and coordinating the overall ride process. The Driver module manages driver profiles, availability, and ride assignments, while the admin module offers administrative tools for monitoring system performance, generating reports, and managing user accounts.

The Persistence Layer leverages the power of Firebase Realtime Database, a cloud-hosted NoSQL database solution provided by Google. This database ensures real-time data synchronization and efficient storage of ride details, user information, and driver profiles.

Figure 1 System Architecture

To enhance the user experience and leverage external services, the system integrates with Third-Party APIs. These APIs may include popular mapping and navigation services like Google Maps for providing accurate location tracking, real-time traffic updates, and turn-by-turn navigation. Additionally, payment gateways like Stripe or PayPal can be integrated to facilitate secure in-app payments for completed rides.

The modular design of the system architecture allows for future enhancements and integrations with minimal disruption to existing components. Each layer can be independently scaled or upgraded as needed, ensuring the system's ability to handle increasing user demand and evolving requirements.

IV. ADVANTAGES AND DISADVANTAGES

A. *Advantages :*

1) *Enhanced User Experience:*
Optimized matching algorithms lead to faster ride assignments, reduced wait times, and improved overall satisfaction for riders. Real-time traffic integration enables more accurate route planning and travel time estimates, further improving reliability for users.

2) *Driver Efficiency:*
Efficient driver-rider matching and route optimization helps drivers maximize their time on the road and potentially increase earnings. Reduced detours and accurate travel time estimations streamline their workflow.

3) *Platform Scalability:*
Firebase's real-time database and cloud-based architecture are designed to handle large volumes of data and user requests. This allows the ride-sharing platform to potentially scale rapidly without requiring major infrastructure overhauls.

4) *Rapid Cross-Platform Development:*
This speeds up development, reduces costs, and ensures a consistent user experience across platforms.
Security: Firebase offers built-in security features, such as authentication, authorization, and data encryption, helping protect sensitive user and driver information.

5) *Potential for Innovation:*
The flexibility of Flutter and the real-time capabilities of Firebase create the foundation for incorporating advanced features in the future, such as carpooling, multi-modal transport integration, or personalized ride recommendations.

B. *Disadvantages :*

1) *Flutter Learning Curve:*
While Flutter is relatively easy to learn, developers unfamiliar with it might face an initial learning curve compared to purely native development approaches.
Performance Limitations: In certain performance-critical scenarios, Flutter might have minor overhead compared to fully native applications. Careful UI design and optimization techniques usually mitigate this issue.

2) *Dependence on Third-Party Services:*
Reliance on services like Firebase for backend functionality and Google Maps for navigation introduces an element of vendor lock-in and dependence on these external services.

3) *Algorithm Complexity:*
Developing sophisticated matching algorithms, especially those involving real-time traffic integration and optimization, can be complex. Thorough design and testing are crucial for ensuring accuracy and efficiency.

4) *Data Privacy Concerns:*
Ridesharing platforms handle sensitive user data. Implementing robust data privacy measures and transparently communicating these practices to users is essential to build trust.

5) *Important Considerations:*

a) Cost:
While Firebase has a free tier, scaling up to larger volumes of users and data can incur charges. It's essential to budget accordingly and explore cost-saving optimization strategies.

b) Maintenance:
Like any software system, a cab management app built with Flutter and Firebase will require ongoing maintenance, bug fixes, and feature updates to ensure continued performance and user satisfaction.

V. ALGORITHM

A. *Optimized Driver-Rider Matching with Real-time Traffic Integration*

1) *Request Handling*
A user submits a ride request, specifying origin, destination, and any preferences (e.g., car type, estimated travel time). The system performs geocoding to convert addresses to latitude/longitude coordinates.

2) *Initial Driver Filtering*
Availability and Proximity: The system identifies all drivers marked as available within a threshold radius around the rider's origin. This initial filtering reduces the pool of potential drivers.

3) *Vehicle Compatibility:*
If the user specified a preference (e.g., larger car for luggage), further filter the pool based on vehicle type.

B. *Real-Time Matching with Traffic-Aware Route Optimization*

1) *Traffic Data Integration:*

Fetch real-time traffic data from an external source (e.g., Google Maps Traffic API). Consider factors like congestion, accidents, and road closures.

2) *Candidate Route Generation:*

For each potential driver from the filtered pool, calculate several candidate routes to the rider's origin point and their
estimated travel times using a routing engine that incorporates traffic data (e.g., Google Maps Directions API or OpenStreetMap Routing Machine (OSRM)).

C. *Potential route variations could include:*

1) *Joint Matching and Route Scoring:*

Estimated travel time to the pickup location, considering traffic. Detour factor: Additional distance the driver must travel to pick up the rider compared to continuing their current activities. Rider preferences (weighting them if necessary).
Driver history (positive ratings could be prioritized). Route Score: Factor in the estimated travel time from the rider's origin to their destination, again considering traffic conditions.

2) *Optimization:*

Employ a suitable optimization technique to select the driver-route pair with the most favourable combination of matching score and route score: Greedy Approach: For faster matching, select the pair with the highest overall score. Metaheuristic-inspired Approach: If computational time allows, use a simplified genetic algorithm or particle swarm optimization variant to explore a wider search space, potentially leading to better matches over time.

D. *Matching Confirmation and Dynamic Rerouting*

1) *Notification:*

Dispatch the ride request to the selected driver and notify the rider of the estimated pickup time and vehicle details. Travel Monitoring: Begin monitoring the driver's progress toward the pickup location. Re-Routing: Periodically reevaluate traffic conditions along the route. If significant congestion arises, proactively suggest alternative routes to the driver using the route optimization process described in step 3.

2) *Key Improvements and Considerations:*

Combination of Techniques: Blends geographical proximity, traffic integration, and optimization approaches, enhancing the realism and efficiency of matches compared to using any in isolation. Rider Preferences: Incorporates user preferences to provide tailored matches.

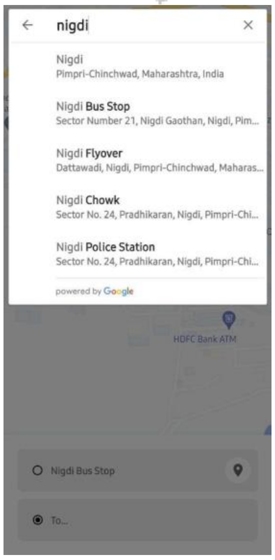
3) *Scalability:*

Starts with an initial filtering stage to
reduce the computational load for subsequent, more intensive calculations. Flexibility: The optimization step could be swapped with different algorithms to experiment and find the best performance for your specific use case. Data Dependency: Relies on accurate traffic data and GPS data.

VI. FUTURE SCOPE

One promising avenue is the integration with public transportation systems, enabling users to plan multi-modal journeys that combine ride-sharing services with buses and trains. This approach not only enhances convenience but also promotes eco-friendly travel options.
This could include EV charging station integration, range estimation, and incentives for using eco- transportation options, positioning the app as a leader in supporting sustainable transportation initiatives.
Leveraging advanced analytics and machine learning techniques could significantly improve various aspects of the app's operations. Expanding the app's functionality to include delivery and logistics services could open up new revenue streams. Users could request pickups and deliveries for packages, groceries, or other items, leveraging the existing driver network and infrastructure. As the app gains traction, international expansion could be considered to cater to a global user base. This would involve adapting the app to local languages, regulations, and transportation norms in different regions or countries, enabling the app to tap into new markets and user segments.

VII. RESULTS



nigdi

Nigdi
Pimpri-Chinchwad, Maharashtra, India

Nigdi Bus Stop
Sector Number 21, Nigdi Gaothan, Nigdi, Pim...

Nigdi Flyover
Dattawadi, Nigdi, Pimpri-Chinchwad, Maharas...

Nigdi Chowk
Sector No. 24, Pradhikaran, Nigdi, Pimpri-Chi...

Nigdi Police Station
Sector No. 24, Pradhikaran, Nigdi, Pimpri-Chi...

powered by Google

Figure 2 Location Search

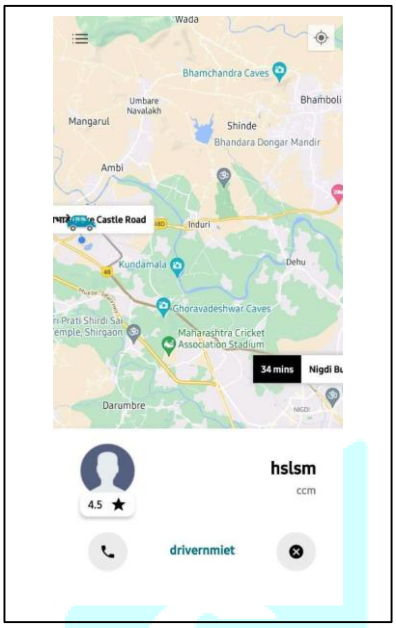


Figure 3 Driver Details

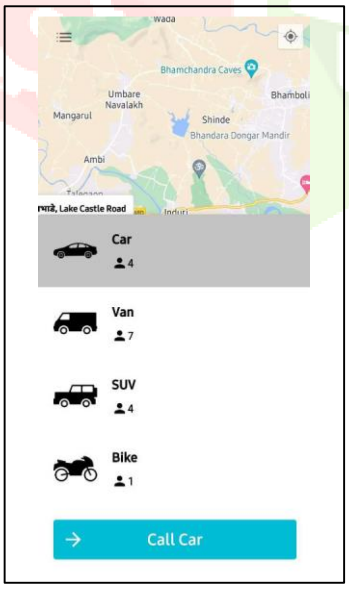
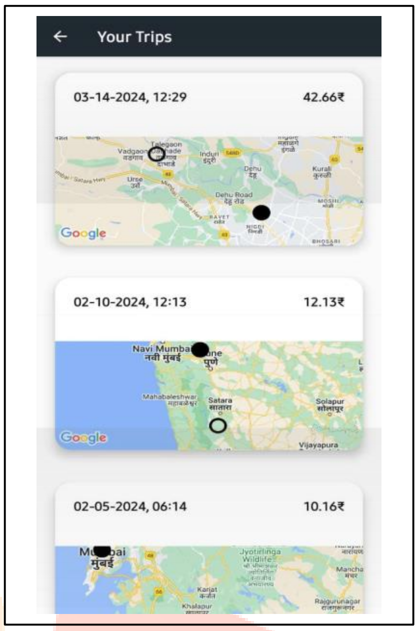


Figure 4 Types of Vehicles


Your Trips

03-14-2024, 12:29 42.66₹

02-10-2024, 12:13 12.13₹

02-05-2024, 06:14 10.16₹

Figure 5 Ride History

VIII.CONCLUSION

The development of the Cab Management App signifies a substantial advancement in the modernization and optimization of the transportation industry. By implementing innovative technologies and carefully designed algorithms, the project has effectively tackled key challenges faced by passengers and drivers within traditional transportation systems. The focus on providing an intuitive platform for ride bookings, vehicle selection, and optimized route planning has significantly enhanced the user experience, resulting in greater convenience and satisfaction.

Moreover, the incorporation of powerful algorithms like Dynamic Ride Matching, Dijkstra's, and A* has been instrumental in boosting system efficiency and reliability. These algorithms facilitate real-time route calculations, minimize wait times, and streamline driver routes, directly contributing to improved service quality. The strong emphasis placed on data security and privacy measures builds confidence among users, assuring them that their sensitive information is protected.

The Cab Management App demonstrates significant potential for continued growth and expansion. With ongoing enhancements, it has the capacity to transform how people travel, fostering a more accessible, efficient, and sustainable transportation landscape. As technology rapidly evolves, the Cab Management App is well-positioned to integrate future advancements, maintaining its role as a leader in transportation innovation and ushering in a new era of mobility for users across the globe.

ACKNOWLEDGMENTS

We wish to thank our parents and associates for their valuable support and encouragement throughout the development of the project work and we would also like to thank our guide Prof. Deepali Patil for guiding us throughout the project work.

REFERENCES

[1] Y. Tong, Y. Zeng, L. Zhou, L. Chen, and K. Ye, "Real-time Ridesharing: A Dynamic Dial-a-Ride Algorithm Considering Joint Matching of Trips and Vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 6, pp. 3584-3596, 2021.

[2] X. Li, W. Wu, Y. Huang, Y. Peng, H. Fu, and W. Sun, "Optimizing Driver-Rider Matching for Ride-Sharing Services Using Metaheuristics," in Proc.

[3] F. Rossi, R. Zhang, Y. Hindy, and M. Pavone, "Dynamic Ride-Sharing System with Real-Time Vehicle Dispatching and Route Optimization,"20, no. 8, pp. 3064-3077, 2019.

[4] W. Luo, P. Sun, D. Mu, and H. Zhao, "A Traffic-Aware Route Planning Algorithm for Ride-Sharing Services," in Proc. of 2018 IEEE International Conference on Intelligent Transportation Systems (ITSC), 2018, pp. 2666-2671.

[5] T. Mishra, A. Kumar, and S. Satapathy, "Designing and Implementing Mobile Applications with Flutter," in Proc. of 2021 International Conference on Computing and Communication Systems (IC3S), 2021, pp. 72–77.

[6] A. Attia, M. Catasta, and E. T. S. El-Masri, "Cross-Platform Development with Flutter: Challenges and Best Practices," IEEE Software, vol. 37, no. 4, pp. 20-28, 2020.

[7] S. Pandey, P. Gunasekaran, S. Mallidi, K. Brohi, and V. Ramasamy, "Scaling Real-Time Mobile Applications with Firebase," IEEE Cloud Computing, vol. 6, no. 6, pp. 26–33, 2019.

[8] H. Liu, H. Jin, C. Wu, and W. Yang, "Security Considerations for Real-time Applications using Firebase," in Proc. of 2020 IEEE 6th International Conference on Cloud and Big Data Computing (CBDCom), 2020, pp. 104-108.

[9] X. Ma, H. Ma, Z. Zhao, S. Chen, and J. Wang, "Privacy-Preserving Ride-Sharing Matching," IEEE Transactions on Dependable and Secure Computing, DOI: 10.1109/TDSC.2022.3208337 (Early access)

[10] I. Butun, I. E. Morrar, and R. Sankar, "Authentication and Authorization Mechanisms for Mobile Applications," IEEE Access, vol. 8, pp. 140696-140713, 2020.

[11] X. Bei and S. Zhang. "Algorithms for Trip-Vehicle Assignment in Ride-Sharing." AAAI Conference on Artificial Intelligence, vol. 32, no. 1., 2018.

[12] Y. Ma, Y. Xu, and Z. Ding, "Optimizing Reliable Ride-sharing Matching for User Requests with Multi-dimensional Uncertainty," IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 14353-14364, 2020.

[13] G. S. Tewari, B. S. Nithya, and S. G. Asha, "Optimization Technique for Ride-Sharing Recommendation Systems," in Proc. of 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), 2017, pp. 999-1004

[14] M. Stiglic, N. Agatz, M. Savelsbergh, and M. Gradisar, "Enhancing Urban Mobility: Integrating Ride-Sharing and Public Transit," Computers & Operations Research, vol. 90, pp. 12-21. 2018.

[15] X. Wang, J. Agatz, and S. Erera, "Stable Matching for Dynamic Ride-Sharing Systems," Transportation Science, vol. 52, no. 4, pp. 850-867, 2018.

[16] Y. Zhan, J. Zhao, and Y. Zhu, "A Spatial-temporal Route Planning Approach for Ridesharing Considering Time-dependent Traffic Conditions and User Preferences," in Proc. of 2021 China Automation Congress (CAC), 2021, pp. 7101-7106.

[17] R. G. Kula, N. P. Rougier, and G. D. Schmitter, "Implementing Dynamic User Interfaces with Flutter," IEEE Computer Graphics and Applications, vol. 41, no. 2, pp. 118-125, 2021.

[18] X. Yang, W. Song, and X. Ran, "Flutter App Performance Optimization on Android Platform," in Proc. of 12th International Conference on Communication Software and Networks (ICCSN), 2020, pp. 53-59.

[19] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "A Study of the Energy Consumption Characteristics of Cryptographic Algorithms and Security Protocols," IEEE Transactions on Mobile Computing, vol. 5, no. 2, pp. 128-143, 2006.

[20] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke. "Middleware for Internet of Things: A Survey." IEEE Internet of Things Journal, vol. 3, no. 1, pp. 70-95, 2016.