FlashRide: A Scalable Web-Based Platform for Intelligent Ride Booking and Allocation

Gajanan K Solanke[1] Ajay Shiketod[2]
[1]Student [2]Assistant Professor
[1,2]Master of Computer Applications
[1,2]Anantrao Pawar College of Engineering & Research, Pune Affiliated to Savitribai Phule Pune University, India

Abstract — The growing need for effective urban transportation has shown the drawbacks of ride-booking (conventional) systems, such as lengthy wait times, lack of transparency, or wasteful resource use. This paper describes FlashRide, a novel web-based ride-booking platform that leverages a complex real-time trip-allocation system for scalable service delivery and ride booking. Modern web technologies and an automated backend system (which reduces manual operations) are the technologies employed in this system to improve efficiency and user experience. FlashRide eliminates the need for manual booking and creates a more dependable booking approach with features like user sessions, optimised driver match, dynamic fare estimate, and safe locking of rides. In the end, the suggested approach offers a scalable solution to satisfy the growing demand from metropolitan regions while facilitating seamless user and provider interaction. Results from this system's testing runs show increases in booking effectiveness, decreases in reaction times, and enhancements to the system as a whole. FlashRide offers people a simple, safe, and effective approach to handle contemporary transportation.

Keywords: Ride Booking System, Intelligent Allocation, Web-Based Platform, Real-Time Tracking, Scalability, Driver Matching, Ride Booking, Spring Boot, MySQL, Dynamic Ride Allocation, Scalable Mobility Platform, Real-Time Driver Matching, Intelligent Transport System, Adaptive Routing Strategy, On-Demand Ride Services

I. INTRODUCTION

Both urbanisation and digital change have grown significantly in recent years. The need for dependable and effective transportation systems has significantly increased as a result of these swift changes. Traditional ride-booking/dispatch systems have historically been inefficient due to delays, a lack of transparency, and an inefficient use of resources, all of which led to subpar service quality and a bad user experience. The creation of intelligent platforms that can generate real-time scaled and optimised transportation solutions is required due to the development of web technology and intelligent systems.

By offering an intelligent web-based ride booking platform that enables real-time ride allocation and effective service delivery, FlashRide resolves this issue. Unlike conventional manual despatch systems that follow pre-planned routes, FlashRide uses an automated back-end mechanism to minimise delays and maximise operational efficiency. FlashRide facilitates smooth user-provider interactions by utilising the most recent web technologies.

The FlashRide system will incorporate intelligent techniques for driver assignment and user validation, further automating procedures (such fare estimation and ride locking) that previously required a lot of human interaction in order to lessen reliance on people and increase system reliability. Additionally, the FlashRide technology is built to be scalable to accommodate rising ride demand in cities.

In order to reduce dependency on people and improve system dependability, the FlashRide system will include intelligent algorithms for driver assignment and user validation, further automating processes (such fare estimation and ride locking) that previously required a lot of human contact. Furthermore, the FlashRide technology is designed to be scalable in order to meet the growing demand for rides in urban areas

II. REVIEW OF LITERATURE:

The idea of digital transformation in terms of finding services and keeping safe records of them served as the foundation for FlashRide's development. This chapter examines the historical background of service discovery, the use of technology to rationalise service discovery, and the incorporation of more modern authentication techniques in web-based systems.

A. Evolution of Service Discovery Platforms:

In the past, local bar associations' publications or static paper directories like the Yellow Pages were used to find service providers. Because of the inherent limitations in their access to the end-user, the traditional methods of finding services are now considered inadequate and ineffective. They also fail to provide timely updates regarding a service practitioner who leaves the profession or for a service practitioner who relocates after holding a particular position. The necessity of "Public Discovery" platforms—which act as a bridge between providers and consumers—is heavily emphasised in recent literature. By developing a high-performance digital platform that addresses the issue of finding service providers, FlashRide resolves this issue.

B. Rationalisation of the Technology Stack:

Oracle SQL and Java Spring Boot Due to previous research's use of the PHP and MySQL stack for managing practitioner records and the ACID compliance offered by those technologies, the market has now demanded more scalable, high-performance enterprise solutions. As a result, practitioners have used larger architectures (in terms of both size and performance) to benefit from a more modular client-server architecture, which enhances the practitioner systems' reliability and ease of maintenance.

C. Scalable Architectures & Performance:

To ensure corporate scalability and performance, a modular approach to architectural design—which divides a system into logical layers—is essential. A practitioner system's

scalability and speed can be enhanced by separating the front-end and back-end components using Java Spring Boot.

D. Data Integrity:

This is achieved by expanding MySQL's established relational structure, which is based on an Oracle SQL relational model. Oracle SQL's relational structure uses a primary/foreign key relationship to link users to particular practitioner service areas. This relationship reduces data redundancy, which raises the degree of consistency of the data in the database.

E. Continuous Authentication and Security Procedures:

Researchers are concentrating on developing safe settings for real-time applications as the use of continuous authentication develops. Several research that supports this transition to continuous authentication for real-time systems have been published in the International Journal of Research in Applied Science & Engineering Technology (IJRASET). • Background Verification: Completing user identity forms on paper is not the same as continuous authentication using machine learning methods. The live verification procedure is no longer necessary for users to finish.

F. Security Through Authentication Using Tokens:

JWTs (Java Web Tokens) are being used as a session-less security method (removed from session-based security) in the context of FlashRide, and they will continue to replace older session-based methods (i.e., relying on MD5 for encryption) in high-volume transactions.

G. Proof of the Research Gap

My investigation into the many types of directory applications that are now in use indicates that most of them have some degree of dead space between those that have administrator control over their structure and those that do not. Although they don't all follow the same pattern, these applications are all attempting to build a Delegated Governance Model with some automated intelligent allocations.

H. Present-day applications are typically divided into two categories:

unregulated and strictly regulated by a single point of control (i.e., an administrator using a "long" approval process). Through the use of technology-based solutions, FlashRide's application combines the two forms mentioned above to provide people with access to public transit and justice in the Internet Age in a more effective and user-friendly manner.

III. Problem Statement

Due to their reliance on human and semi-automated processes, the present techniques for booking rides across urban areas have problems with speed and accuracy. Due to the fact that many providers will have outdated or no access to real-time data about their possible riders, this can result in considerable delays for consumers looking for a trip after making a request as well as decreased response time for drivers who will supply the service.

When consumers require instant assistance, the lengthy sign-up times and complexity of the current transportation booking systems might pose serious obstacles. Another significant problem with current ride-booking systems is security, as their present client authentication techniques are insufficiently secure and do not scale to match the demands of a contemporary-app.

Intelligent driver assignment systems are not used by most ride booking systems now in use, which can result in wasteful resource utilisation and longer customer wait times. The aforementioned issues present a chance for the creation of a ride-sharing service that delivers rides quickly, securely, and in real time while offering the best resource allocation techniques.

IV. Objective:

The goal of this research is to develop an intelligent and user-friendly ride-booking application by utilising cutting-edge web technology to enhance traditional transportation industry practices.

A. Particular Objectives:

Peak usage will be supported by a scalable system
Users can be accommodated by a three-tier architecture, which can grow as demand rises.

B. Smart Driver Assignment:

The closest available driver is found using an algorithm based on the user's location, making it possible to match Users in need quickly and effectively.

C. Improved Data Reliability and Security:

To guarantee correct data maintenance, secure user authentication (JWT) techniques and organised database systems are used.

D. Automated Calculation of Fares:

Transparency is achieved by dynamic fare estimation based on distance and traffic conditions.

E. Improved Accessibility for Users:

creation of an easy-to-use, responsive user interface to enable transportation reservations on any device.

F. Decrease Dependency on Manual Labour:

Key procedures (booking, allocation, and reporting) can be automated to boost productivity and reduce mistake.

G. Hypothesis of the study:

The purpose of the research hypothesis is to determine whether ride-booking systems' performance, security, and dependability may be greatly enhanced by integrating contemporary online technologies and mathematical models in comparison to more conventional methods.

1) The main theory (H_1) :

With a proximity-based allocation model and a Spring Boot-based backend, the suggested FlashRide system dramatically cuts ride allocation time (goal < 1 second) and enhances system responsiveness under heavy concurrent user demands.

2) Supplementary Theory

- H_2 (Security): By lowering server-side overhead when compared to conventional session-based techniques, stateless JWT-based authentication enhances system security and scalability.

- H_3 (Pricing Transparency): Pricing accuracy and user confidence are improved by using a dynamic fare estimating approach based on traffic and distance considerations.
- H_4 (Data Integrity): By using a normalised relational database, redundancy between user, driver, and booking records is removed and data consistency is ensured.
- H^2 (System Efficiency): Resource utilisation is enhanced and user waiting times are decreased by intelligent driver allocation based on real-time proximity.

3) Hypothesis is Null (H_0):

The FlashRide system does not significantly outperform current ride-booking systems in terms of response time, allocation accuracy, security, or system efficiency.

a) Significance of the Study:

An antiquated method of delivering and overseeing urban transportation services is being replaced with the FlashRide System. The FlashRide System can enhance the efficiency of current urban transit services by utilising an efficient, scalable, and automated digital platform. The study's findings will support technological advancements, social equity related to existing technology, and the integration of these technologies into operations.

1) Advances in Technology

- Upgraded System Architecture:
  This study's primary contribution is the transition from conventional stacking web designs to a more contemporary system architecture utilising relational databases and Spring Boot, which will improve the user experience for concurrent users engaging with the FlashRide System.
- Enhanced Security Systems:
  A more secure stateless method of data protection than a session-based method will be offered by the usage of JSON Web Tokens (JWT) for user-based authentication.
- Better Algorithm Architecture:
  The FlashRide System's coding complexity has increased and its reaction time has decreased due to the development of algorithmic methods for distributing trips based on driver availability and offering the capacity to dynamically estimate fares.

2) Social Effects via Access

- Boost Access:
  Ride-booking services will be more accessible to all users thanks to adaptable designs that work on a variety of devices.

V. Methodology / System Architecture:

In order to achieve efficient ride booking, speedy ride request processing, and secure communication, FlashRide's system design follows a logical methodology. Using UML representations, this design approach focuses on both the system's operation and the allocation logic for the intelligent distribution of resources, such as drivers.

A. How the system works

The system's entire functioning follows this method, which operates sequentially

1) User Communication:

The user enters the requested ride details (pick-up and drop-off location) into the web-based front-end application.

2) Verification

By using secure token-based authentication, the system verifies that the user is permitted to access the web-based front-end application.

3) Processing Requests:

The user's ride request is processed by the system's back end, which also determines whether any drivers are accessible based on real-time availability (i.e., driving distance to pick up the user).

4) Driver Assignment:

Depending on the driver's availability, the back-office system will assign the closest driver from the user's pick-up location once it has identified the most suitable drivers.

5) Calculating Fares:

Using dynamic procedures depending on the user's ride request and anticipated distance travelled, the fare is calculated at the back-end of the FlashRide system.

B. Allocation of drivers:

A simple proximity-based selection process is used for driver allocation. The following are the steps involved in driver allocation:

1) Getting a list of drivers who are currently in operation.
2) Sorting the list of active drivers according to their availability.
3) Calculating the user's distance from each possible driver.
4) Using the distance between each driver and the user to sort the list of available drivers.

1) Frontend Development with the Technology Stack

To guarantee that the website can be properly shown on a variety of devices, the frontend was created with mobile first in mind.

- Markup and Style: React is used to produce an appropriately organised and aesthetically consistent website.
- Framework: To make our website responsive, Bootstrap 5 was utilised.

2) Development of Backends

The backend is in charge of handling the site's business logic, responding to user requests, and guaranteeing the security of every transaction. • Framework: To offer business logic and enable multiple users to request data simultaneously, Java Spring Boot was utilised.

3) Building Databases

To guarantee data consistency and speedy query processing, the database was created as a relational database.

- Database System: Scalable and dependable data storage was provided using Oracle SQL.
- Tables: To relate Users, Drivers, and Bookings, each table included main and foreign keys.
- Normalisation: To get rid of redundant data and preserve data integrity, the database was successfully normalised.

VI. SYSTEM ARCHITECTURE:

Fig. 1 illustrates the system architecture of the FlashRide platform. The process begins with the rider, who interacts with the web application (frontend) to request a ride. The frontend sends the request to the backend server, where the core processing and allocation logic are executed.

The backend server communicates with the database to retrieve and store relevant data such as user details, driver availability, and ride information. Based on this data, the system identifies and assigns the most suitable driver. The assigned driver then receives the ride request and proceeds with the service.

This architecture ensures efficient communication between components, supports real-time data processing, and enables scalable and reliable ride booking operations.

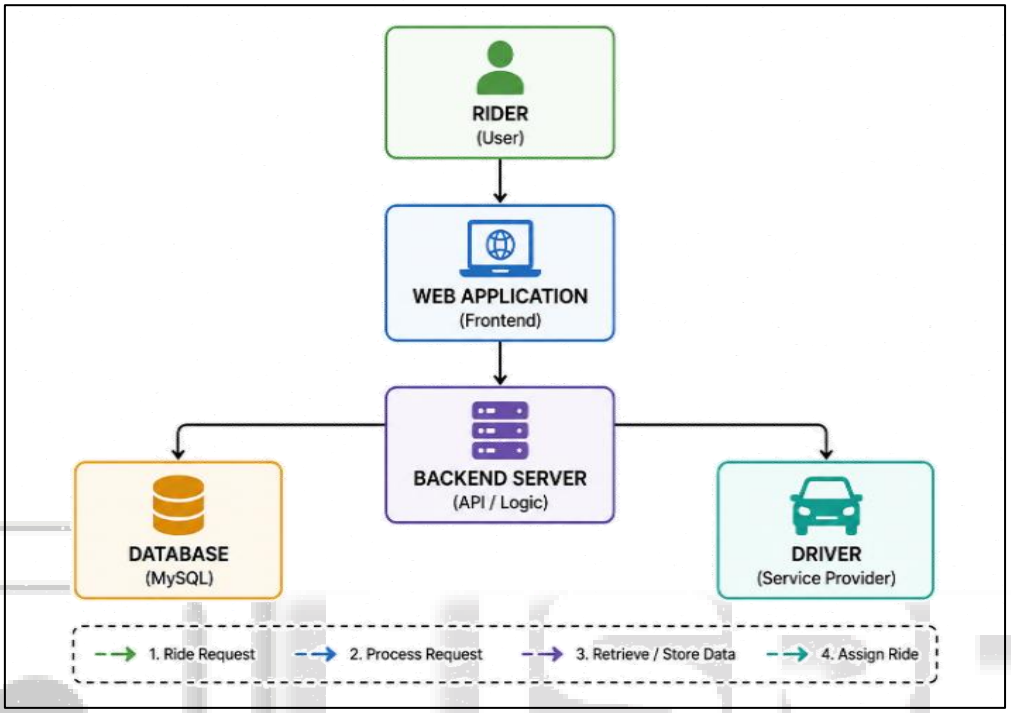

Fig. 1: System Architecture of FlashRide

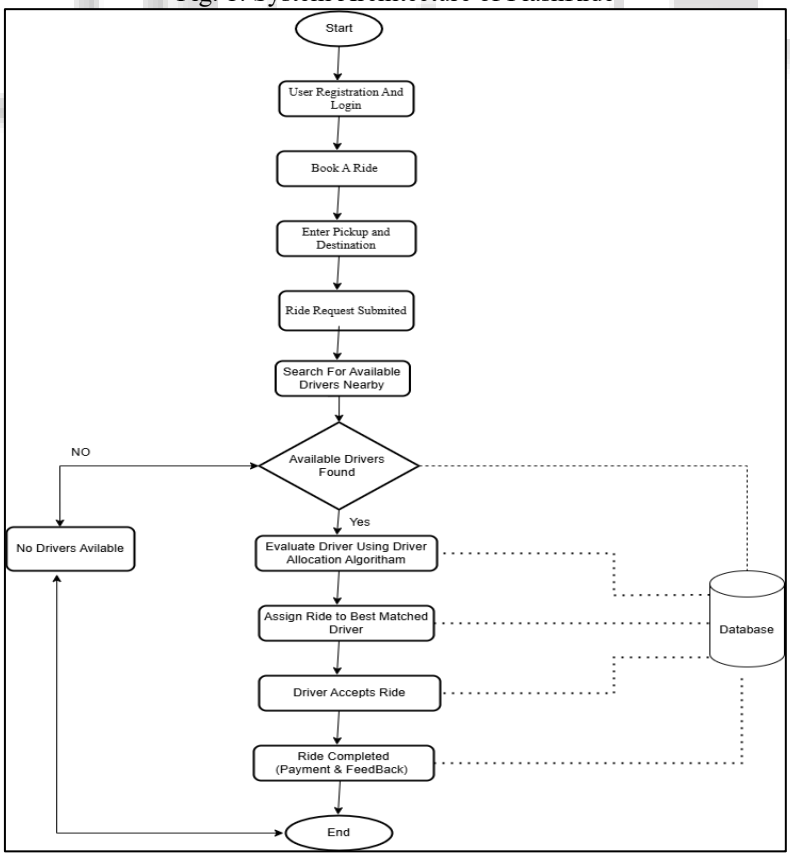

Fig. 2: Flowchart Of Working of FlashRide Application

Fig. 2 illustrates the workflow of the FlashRide system for intelligent ride booking and allocation. The process begins with user registration and authentication, where the rider logs into the platform. After successful login, the user enters the pickup and destination locations to request a ride.

The system then processes the request and retrieves real-time data of available drivers from the database. Based on this data, an intelligent allocation algorithm is executed to identify the most suitable driver. The algorithm considers parameters such as distance, driver availability, and estimated time of arrival.

Once the optimal driver is selected, the ride request is assigned, and both the rider and driver are notified. Real-time tracking is then initiated, allowing the user to monitor the driver's movement. The ride continues until the destination is reached, after which the ride is completed and stored in the system for future reference.

This workflow ensures efficient ride allocation, reduced waiting time, and improved system performance.

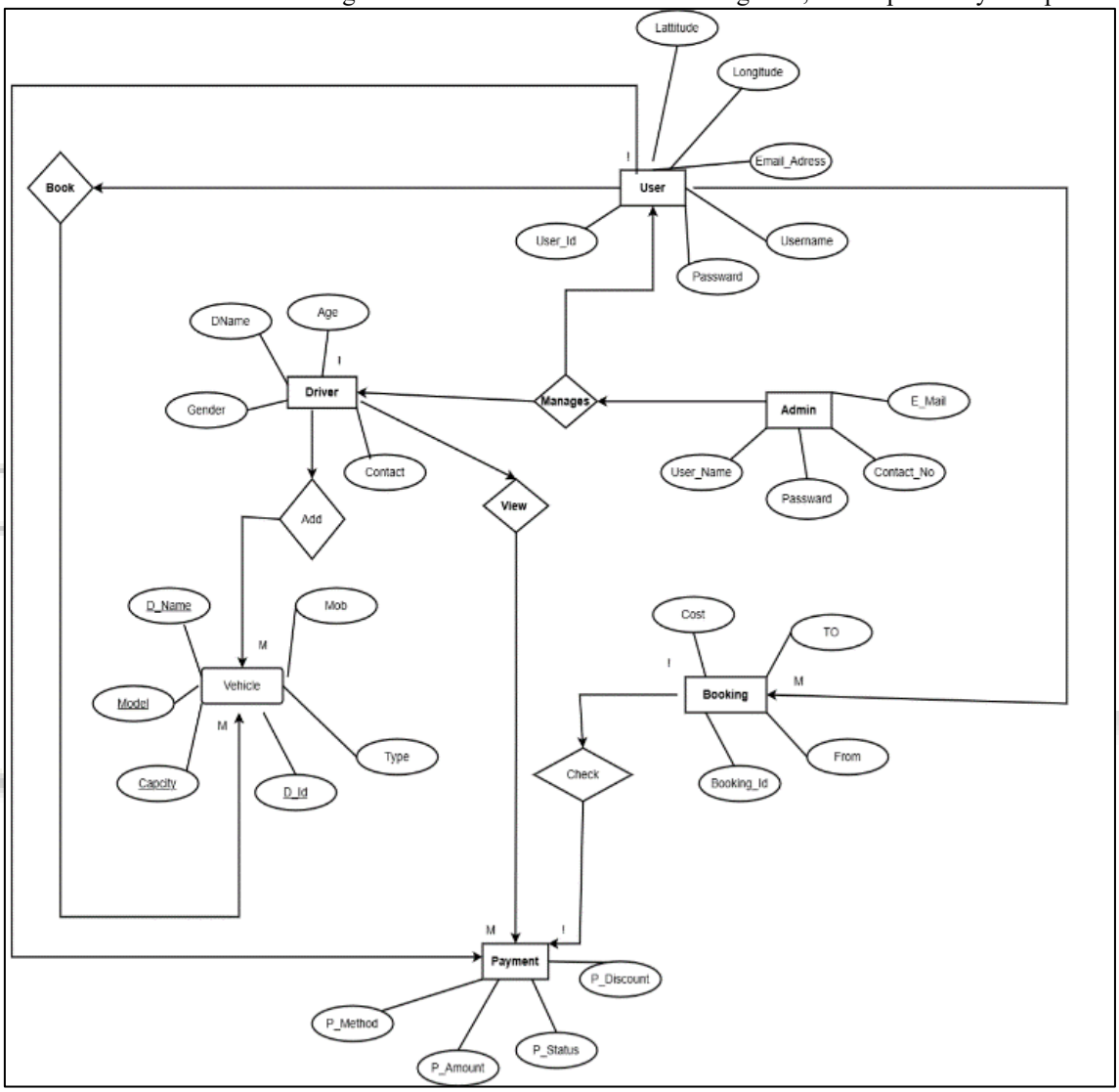

Fig. 3: Entity-Relationship Diagram of FlashRide System

Fig. 3 shows the Entity-Relationship (ER) diagram of the FlashRide system. It includes key entities such as User, Driver, Ride, and Booking. The User entity stores rider information, while the Driver entity maintains driver and vehicle details. The Ride entity contains ride-related data such as pickup location, destination, and status, and the Booking entity manages ride requests.

A user can create multiple bookings, and a driver can complete multiple rides over time, representing one-to-many relationships. This ER diagram helps in organizing data efficiently and supports smooth ride booking and allocation in the system.

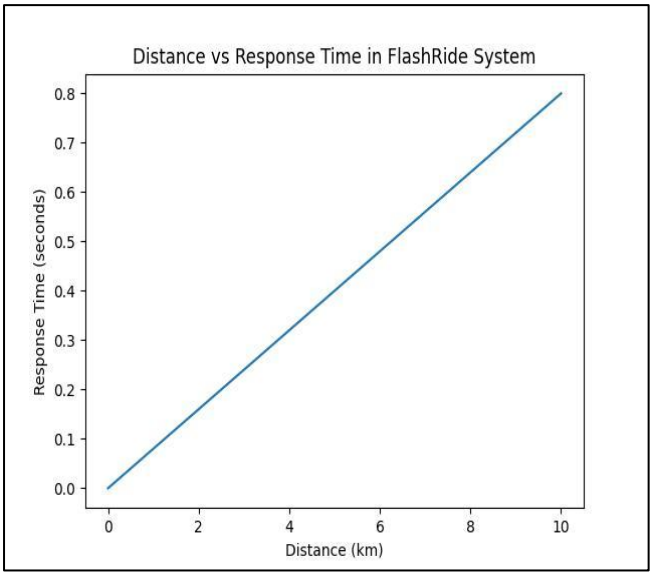

Fig. 4: Distance vs Response Time in FlashRide System

Fig. 4 illustrates the relationship between distance and response time in the FlashRide system. It can be observed that the response time increases linearly as the distance between the rider and driver increases. This is because longer distances require more processing and travel estimation, leading to slightly higher response times.

However, the system maintains a consistent and predictable increase, indicating efficient handling of distance-based calculations. The linear trend demonstrates that the FlashRide platform performs reliably and scales well even as the distance grows, ensuring stable and efficient ride allocation.

VII. Proposed System:

The FlashRide system is a high-performance web-based platform designed to improve urban transportation. Scalability and security are made possible by the three-tier design of the FlashRide system, which also uses a centralised database with real-time ride lookup and driver allocation to replace fragmented and manual operations.

A. The system framework.

The Spring Boot framework, which powers the backend processing and manages many user requests concurrently, is used in the development of FlashRide. Additionally, FlashRide uses a relational database to guarantee data consistency, ride and user dependability from a storage standpoint, and a safe environment for storing ride and user data. Furthermore, compared to alternative forms or conventional systems, FlashRide's three-tier architecture offers better speed and scalability.

B. Primary Features

1) Smart Drive Assignment:

This method reduces customer wait times by assigning drivers based on their availability and present location. • Secure Authentication: JWT is used to enable stateless, secure client-server communication. • Modular Architecture: The system is divided into three modules: admin, driver, and user. Access is dependent on roles for each. • Responsive Interface: Made for mobile devices, this feature makes it simple for users to reserve a ride without complicated procedures. • Automated Reporting: The administrator can keep an eye on system utilisation and performance in real time thanks to real-time reporting.

C. Workflow

The steps in the system's process are as follows:

1) The user requests a ride and enters the ride's location and other details.
2) Verify user identity using a secure token.
3) Responding to requests and locating drivers who are available.
4) Once the ride information have been processed, the user's screen will display the confirmed ride.

VIII. Limitations of The System:

While the FlashRide System has many benefits there are a number of limitations as to what can be done presently:

A. Reliance on Internet Connectivity:

The system operates only with a connection to the internet in real-time; if there is poor connectivity or latency the performance of the System will be diminished.

B. Admin/Driver Verification Involves Manual Process:

Currently both Admin and Driver verification require manual work to be completed; therefore the manual processing will slow down both the scale of the system as well as how quickly drivers can be brought on to the System.

C. Limited Geographic Use:

The System has been built for optimal performance for urban geographical locations; it is expected that performance will vary for rural or less populated areas as there will likely be less number of drivers in those locations.

D. No real-time Communication:

The Platform currently does not allow for chat or voice communications to be done between Users and Drivers in real-time.

E. Hardware Dependency:

The systems performance will depend heavily upon the type of Server hardware that is being used,therefore if lower class/hardware specification will affect the Response time of the System.

F. External Factors:

The accuracy of the allocation will be impacted by many external dynamic influences (Traffic variation and GPS coordinates inaccurately.

IX. Conclusion

FlashRide is a complete digital solution designed to solve today's ride-booking system issues. It is based on web technologies and Intelligent Processing. In addition to enabling real-time trip assignment, secure user verification, and increased dependability while offering transportation services, FlashRide offers a single, centralised digital solution to do away with conventional manual operations.

In order to guarantee quick response times, data integrity, and secure communication, the system was built utilising Spring Boot, a relational database, and JWT-based security. Algorithms with optimal design offer faster response times and better driver assignments.

All things considered, the new system will provide sufficient size for urban transportation systems in the real world, increase operational efficiency, and enable more consumers to access services. FlashRide's design serves as an example of how well-thought-out intelligent online applications may greatly enhance existing mobility services.

REFERENCES

[1] S. Banerjee, R. Johari, and C. Riquelme, "Pricing in Ride-Sharing Platforms: A Queueing-Theoretic Approach," in Proc. ACM Conf. Econ. Comput., 2015.
[2] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani, "AdWords and Generalized Online Matching," J. ACM, vol. 54, no. 5, Oct. 2007.
[3] K. Ghoseiri and R. Ghannadpour, "Multi-Objective Vehicle Routing Problem with Time Windows using Genetic Algorithm," Transportation Center, 2010.
[4] A. Syed, P. Reddy, and M. Ganesan, "Adaptive Matching Algorithms for High-Density Urban Ride-Hailing," in Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC), 2019.
[5] A. Karnik, "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.
[6] R. Jain, A. Durresi, and G. Babic, "Throughput Fairness Index: An Improved Measure of Scalability and Fairness," Proc. IEEE, 1999.
[7] M. Chowdhury, "Improved Hungarian Algorithm for Optimal Assignment in Transportation Systems," Indian J. Sci. Technol., 2022.
[8] S. Rathore and S. K. Ghosh, "Deep Reinforcement Learning-Based Dynamic Ride-Hailing Matching," in Proc. IEEE Int. Conf. Adv. Comput. (ICAC), 2023.
[9] V. Pandey and S. Boyles, "Dynamic Ride-sharing with Meeting Points: A Stochastic Optimization Approach," J. Indian Inst. Sci., vol. 101, no. 2, pp. 245–261, 2021.
[10] R. K. Gupta and P. Singh, "A Genetic Algorithm Approach for Multi-Objective Ride-Sharing in Indian Metropolises," Int. J. Intell. Transp. Syst. Res., 2024.
[11] A. Singh and M. Misra, "Microservices Architecture for High-Concurrency Transportation Systems," in Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD), 2022.
[12] P. Verma and S. Taneja, "Load Balancing Strategies for On-Demand Mobility Services in Cloud Environments," IEEE Trans. Cloud Comput., vol. 10, no. 3, pp. 1580–1592, 2023.
[13] J. Alonso-Mora et al., "On-Demand High-Capacity Ride-Sharing via Dynamic Trip-Vehicle Assignment," Proc. IEEE, 2017.
[14] A. Mehta et al., "Online Matching Algorithms for Ride Allocation," J. ACM, 2007.
[15] G. Wang et al., "On-Demand Ride Matching in Spatial Models," 2019.
[16] A. Syed et al., "Adaptive Matching Algorithms for Ride-Hailing," in Proc. IEEE ITSC, 2019.
[17] J. Kuhn, "Hungarian Method for Assignment Problem," Nav. Res. Logist. Q., 1955.
[18] G. S. Kumar, "Real-Time Geospatial Indexing for Scalable Ride-Booking Platforms," Indian J. Comput. Sci. Eng., vol. 14, no. 1, pp. 88–97, 2024.
[19] X. Long, J. Mao, H. Yang, and J. Li, "Dispatch Strategy for Multimodal Ride-Hailing Services," Transp. Res., 2026.
[20] J. Xu et al., "Taxi Dispatch System Based on Demand Prediction," J. Parallel Distrib. Comput., 2021.
[21] G. Liberona et al., "Driver Allocation in Ride-Hailing Systems Using Game Theory," 2021.
[22] Y. Duan et al., "Optimizing Order Dispatch in Ride-Sharing Systems," in Proc. IEEE ICCCN, 2019.
[23] N. K. Meena and A. K. Jain, "Spatio-Temporal Demand Forecasting for Ride-Hailing Services using Deep Learning," in Proc. IEEE 6th Conf. Inf. Commun. Technol. (CICT), 2022.
[24] R. Sharma, "Impact of Dynamic Pricing on Passenger Demand in Urban India: A Machine Learning Perspective," Transp. Dev. Econ., vol. 9, no. 4, 2023.
[25] S. Rathore and S. K. Ghosh, "Deep Reinforcement Learning-Based Dynamic Ride-Hailing Matching," in Proc. IEEE Int. Conf. Adv. Comput. (ICAC), 2023. (Focuses on matching logic for Indian urban centers).
[26] V. Pandey and S. Boyles, "Dynamic Ride-sharing with Meeting Points: A Stochastic Optimization Approach," J. Indian Inst. Sci., vol. 101, no. 2, pp. 245–261, 2021. (IISc Bangalore research on "Meeting Points" to improve ride efficiency).
[27] R. K. Gupta and P. Singh, "A Genetic Algorithm Approach for Multi-Objective Ride-Sharing in Indian Metropolises," Int. J. Intell. Transp. Syst. Res., 2024. (Addresses balancing wait time vs. fuel cost in Indian

