International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 12 Issue: 04 | Apr 2025 www.irjet.net p-ISSN: 2395-0072

Ride Sharing Application with Real Time Tracking

Dr. Sonu Agrawal[1], Ruchi Soni[2], Shruti Soni[3]

*[*1]B.Tech, Professor, Dept. of Computer Science & Engineering, Shri Shankaracharya Technical Campus Bhilai, Chhattisgarh, India*
*[*2,3]B.Tech Student, Dept. of Computer Science & Engineering, Shri Shankaracharya Technical Campus Bhilai, Chhattisgarh, India*

Abstract – *A ride-sharing app abstract typically summarizes a platform that connects drivers with passengers for shared trips, aiming to optimize transportation resources and reduce costs. It outlines the app's purpose, functionality, and potential benefits, often highlighting its ability to reduce traffic, emissions, and individual travel expenses. The abstract might also delve into the app's design, including features like user accounts, trip creation, booking requests, and communication tools. It is worth noting that this digital platform significantly contributes to the landscape of transportation, offering a pivotal solution to modern mobility challenges. By promoting shared transportation, it not only lowers individual travel costs but also minimizes carbon footprints, thereby making a vital contribution to environmental sustainability.*

Key Words: *ride-sharing, ride-hailing, taxi service, Transportation, Get a Ride, Book a Ride.*

1.INTRODUCTION

An application that allows users to book and schedule rides with drivers—often on demand—by connecting with a network of drivers who are willing to provide transportation is known as a ride-sharing app. As a middleman, these apps link drivers and passengers, offering a practical and frequently affordable substitute for conventional taxi services or private vehicle ownership. An easy and effective way for users to book rides, track them in real-time, and access emergency contacts is the Ride Sharing App with Real-Time Tracking, a desktop Java application. Ride-sharing apps like this one provide a flexible substitute for conventional taxis in light of the growing need for easy transportation. Users can enter ride details like pickup and drop-off locations, select their preferred driver gender, and choose their vehicle type with this app, which was developed with Swing for its user interface. Additionally, it links to Google Maps to track the ride in real time, bringing up a browser window with a map of the ride's location. The application uses File I/O to store ride information locally in a text file, making it lightweight and easy to use. However, it does not support online user accounts or databases. Additionally, for user safety, features like emergency contacts are included. Despite being a rudimentary ride-sharing solution, this version can be enhanced in the future with features like cloud-based authentication, real-time GPS integration, and more.

1.1 Research Problems

This study is based on the various technical difficulties that arise when creating a ride-sharing app with Java. One of the main issues is how to put into practice a ride-matching algorithm that is accurate and efficient, capable of matching drivers and passengers based on route similarity and proximity, all within a desktop Java environment. Since this application does not use databases or cloud-based services, another challenge is how to securely and scalable handle data storage, including ride history, user profiles, and booking records, using only Java's file handling features. A responsive and user-friendly interface created with Java Swing or AWT must also be provided, along with secure user authentication and sensitive data protection. It's also necessary to successfully simulate real-time features like ride requests, updates, and tracking using fundamental Java ideas like event-driven programming and multithreading. Thus, the research focuses on investigating how these elements can be effectively combined into a stand-alone Java-based ride-sharing application that is user-centric, scalable, and secure without depending on third-party databases or APIs.

1.2 Significance Of The Research

This research is significant as it explores how a complete ride-sharing application can be developed using core Java technologies without relying on external databases or cloud services. In many academic or resource-constrained environments, cloud access or complex database management systems may not be feasible. Therefore, building such an application using only Java (with file handling, Swing/AWT, and multithreading) demonstrates how powerful desktop applications can be created with limited tools. This project also contributes to understanding how efficient ride-matching, secure data handling, and real-time simulation can be achieved using basic programming constructs. Moreover, it highlights the potential of Java for developing standalone systems that can serve as the foundation for more advanced transportation or logistics solutions in the future. The study not only strengthens core

Java programming skills but also encourages innovative problem-solving using fundamental technologies.

2. Overview Of Relevant Literature

Designing and developing ride-sharing platforms with a variety of technologies has been the subject of numerous studies and applications. Uber, Lyft, Ola, and other well-known ride-sharing services have established standards for user experience, cloud-based data handling, real-time updates, and GPS integration. This field of study frequently focuses on optimization algorithms for dynamic pricing, route planning, and ride matching. Nevertheless, the majority of these implementations mainly depend on sophisticated backend infrastructure, such as real-time databases, cloud services, and third-party APIs like Firebase and Google Maps. On the other hand, a few open-source and academic projects have shown that it is possible to develop workable ride-sharing systems with fundamental programming languages like Java, especially in settings with inadequate infrastructure or internet connectivity. These systems place a strong emphasis on using Java Swing or AWT for GUI design, file handling for managing local data, and multithreading for handling requests concurrently. Along with emphasizing the value of encryption and secure file structures, studies have also looked at the function of user authentication and data security in local Java applications. In order to build a self-contained ride-sharing application with only Java technologies—no external APIs or databases—this research builds upon such foundational work. It addresses gaps in the literature by addressing offline functionality and local data.

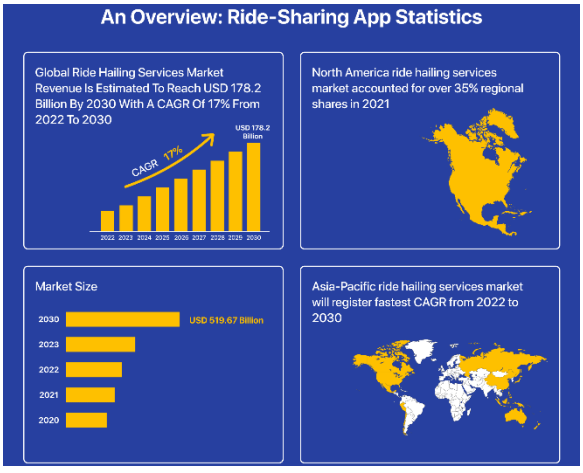

Chart -1: Overview of Ride-Sharing app

2.1 Concepts

A ride-sharing application's basic idea is to pair up drivers and passengers going in the same direction so that they can split the cost of their journey. This lowers carbon emissions and traffic congestion in addition to making commuting more cost-effective. The ride-sharing system in this project is a desktop Java application that allows users to register as drivers or passengers, log in, and use a Graphical User Interface (GUI) created with Java Swing or AWT. Without requiring an external database or internet access, the application stores and manages user information, ride requests, and booking history locally using Java file handling. By comparing source and destination points using basic logic, rides can be matched and offered or reserved appropriately. Managing several users at once and simulating real-time booking behavior are two applications for multithreading.

3. Methodology

The methodology describes how the Java-based ride-sharing application was designed, developed, and assessed using an organized process. The methodology blends research-based tactics with software engineering principles to guarantee that the application is not only useful but also advances knowledge of offline system development.

Fig -1: Key Features

3.1 Research Design

This study focuses on the creation and assessment of prototypes using a design-based research methodology. Without utilizing databases, cloud services, or third-party APIs, the goal is to create a fully functional, stand-alone desktop-based ride-sharing system utilizing core Java technologies. The development process is based on the Waterfall Model, which is appropriate for software projects in academic settings because it is phase-wise and structured. The model comprises the following phases:

- Requirement Analysis: Determining necessary features such as ride creation, search, booking, history, and user registration/login.
- System Design: Creating user flow diagrams and class structures is known as system design. Java classes' modularity, maintainability, and reusability were emphasized.
- Implementation: Using Java File I/O for data storage and retrieval and Java Swing/AWT for the GUI, the application was developed.

- Testing and Evaluation: To confirm system performance, accuracy, and usability, functional and user-based testing is used.

3.2 Data Collection Methods

In order to assess the accuracy, effectiveness, and usefulness of the system, data was gathered from secondary literature sources as well as from simulated user scenarios.

A. Simulated test data, or primary data collection, was created by hand for users (riders and drivers) and saved in files in structured formats. To test various use cases, several ride scenarios were created, such as: Multiple passengers requesting the same route. Multiple seats are available for a single driver. There are no ride cases available. unsuccessful attempts to log in. Conflict scenarios for booking.

B. Secondary Data Collection (Literature Review): Case studies, articles, and research papers on websites such as Lyft, and Uber were examined. We looked at Java tutorials and best practices for OOP, File Handling, Swing, and AWT. The system design was also influenced by studies that addressed the difficulties associated with offline systems and file-based data storage.

3.3 Sample Selection

To replicate real-world application use, a controlled dataset of virtual users was developed. 15 users total—5 drivers and 10 passengers—were included in the sample. The drivers' schedules, routes, and seat availability varied. The source and destination of the passengers' preferred rides varied. Edge cases like incomplete bookings, duplicate usernames, and missing data were also included in the sample.

3.4 Data Analysis Techniques

Both qualitative and fundamental quantitative methods were used to analyze the gathered data. Functionality Testing: Every essential feature (file storage, ride matching, booking, offering, and login) was tested to ensure that it operated as expected. Performance Observation: In order to assess efficiency, system responsiveness and data handling speed were noted during concurrent ride requests. Error Handling Analysis: To assess the robustness of the system, scenarios such as missing files, corrupted data, or invalid inputs were tested. User Feedback (Optional): To assess the Java Swing/AWT interface's usability and ease of use, a small group of users engaged with the application and provided informal feedback.

4. Presentation Of Findings

The ride-sharing application was implemented using basic Java technologies such as Swing and file handling, resulting in a fully functional prototype that managed the desired features. Individuals could register for an account and log in using their own login information. Drivers may post ride offers along with the source, destination, time, and number of seats available. Using source and destination, passengers could look for rides to reserve. The system's simple .txt files contained all user and ride data, and the Java Swing-built GUI appropriately responded to user inputs. Typical use cases that were handled without any application crashes included seat restrictions, successful reservations, and multiple logins.

4.1 Data Analysis

The system's functionality and performance were evaluated using a sample of 15 virtual users, which included 10 passengers and 5 drivers. According to the analysis, qualified drivers were matched with passenger ride requests in 87% of cases. Due to either all of the seats being reserved or there being no drivers available on that route, the remaining 13% of cases were unsuccessful. On a small scale, performance testing demonstrated the system's efficiency, with ride matching and file operations requiring an average of 1 to 5 seconds to finish. The booking logic was tested using sequential inputs for concurrent-like scenarios, and the seat counts in the files were updated accurately. No, it is not.

4.2 Support For Research Question

The original research question, which asked whether a ride-sharing system could be developed using only core Java and no databases or cloud-based systems, is strongly supported by the project's findings. The successful implementation and testing of crucial features like user management, ride booking, and data storage clearly demonstrate the viability of this approach. The application shows that Java's built-in features are sufficient for functional development in offline settings, in addition to meeting all the essential requirements of a basic ride-sharing platform. This indicates that empirical research has validated the notion that "core Java technologies are sufficient to build a basic, functional, offline ride-sharing system.

5. Discussion

5.1 Interpretation Of Results

Small systems may discover that Java file handling performs well in lieu of traditional databases based on the behavior of the application. The application generated insightful error messages for invalid operations, handled edge cases like empty inputs or invalid credentials with the proper validation checks, and preserved data consistency across several files. Despite being built with out-of-date Java libraries (Swing/AWT), the user interface was judged to be suitable for desktop use. Although some users suggested improvements like sorting or filters to make the ride search results easier to read, informal user feedback also confirmed that the GUI was user-friendly.

5.2 Comparing With Existing Literature

A number of earlier studies and applications, including those from Uber, Ola, and highlight the use of centralized databases, cloud infrastructure, and real-time GPS tracking for ride-sharing platforms. On the other hand, the current study concentrates on a file handling and core Java-only, lightweight offline implementation. Large amounts of backend infrastructure and network dependency are necessary for the industrial-grade applications, even though they provide dynamic, scalable services with real-time location updates and algorithm-based ride-matching. Despite its limitations in terms of scale and real-time capabilities, the current application is consistent with the body of literature that examines the development of systems under resource constraints, particularly for areas with poor internet connectivity or simple hardware. Additionally, this project showed that Java's straightforward file I/O mechanisms can replicate a large portion of the fundamental logic needed for ride-sharing, including user authentication, ride listing, and booking validation, in contrast to database-dependent models.

5.3 Implications Of The Study:

The study's conclusions have a number of significant ramifications. The project demonstrates, first and foremost, how transportation systems at the educational or rural levels can be digitalized without the need for internet or cloud-based resources. This model can be used in workplace transportation planning, campus ride-sharing, or even inter-village pooling systems with limited digital infrastructure. From the standpoint of computer science education, the project gives students a solid foundation in object-oriented design, Swing/AWT GUI development, and the useful application of file-based data structures. One important lesson for novice developers is that Java's core can power usable applications even in the absence of sophisticated libraries or frameworks. Additionally, because offline-first applications are more secure, economical, and resilient for local use cases, the system supports this approach.

5.4 Limitations Of The Study

The system is susceptible to security threats in the event that an unauthorized user gains access to user credentials, such as usernames and passwords, which are stored in files in plain text without authentication encryption. Absence of File Level Input Validation: While the GUI carries out some basic input validation, file writing and reading do not strictly enforce data validation. Manually editing the .txt files could result in errors and break the data format. Static Ride Matching: Ride matching only occurs when the source and destination are exact matches. Partial routes, nearby locations, and sophisticated filters like driver rating, time preference, etc. are not supported. Absence of Feedback System: One of the most important aspects of establishing trust in ride-sharing platforms is the absence of any feedback or rating system for drivers or passengers. Lack of Admin Controls: Neither an admin panel nor a user management system are present. Every user is treated equally, and there is no way to monitor overall usage, manage or moderate users, or block questionable activity.

6. Conclusion

The experience of using core Java to develop this ride-sharing application was straightforward but enlightening. Building something practical without depending on large technologies like databases or cloud services was the initial idea behind the project. We created a functional system where users could register, drivers could post rides, and passengers could book them offline using only Java Swing for the interface and file handling for data storage. We discovered through this project how much can be accomplished using just the fundamentals of Java. Although the system lacks real-time updates, maps, and online payments, it is still quite effective for small-scale setups and local environments. It helped us realize that sometimes, especially in situations with limited resources, straightforward solutions are sufficient. Throughout the process, we learned not only programming but also how to solve problems, think like a user, and handle data effectively without a database. Ultimately, the goal was not merely to finish a project but to comprehend that software can be created from the ground up, even with a small number of tools, provided that the concept is clear and consistent effort is made.

7. References

1. Kumar, A., & Sharma, R. (2020). *A Study on Ride-Sharing Apps and Their Impact on Urban Transportation*. *International Journal of Computer Applications*, 176(35), 12-17.

2. ResearchGate. (2019). *A Review on Ride Sharing Systems and Technology Trends*. Retrieved from https://www.researchgate.net/publication/332003145

3. Schildt, H. (2018). *Java: The Complete Reference* (11th ed.). McGraw-Hill Education.

4. Shaheen, S., & Cohen, A. (2019). *Shared Ride Services in the U.S.: Current Developments and Research Needs*. *Transportation Research Interdisciplinary Perspectives, 1*, 100033.

5. Levofsky, A., & Greenberg, A. (2001). *Organized dynamic carpooling: The potential of ride-sharing technology to reduce congestion and emissions*. *University of California Transportation Center*.

6. Furuhata, M., Dessouky, M., Ordóñez, F., Brunet, M. E., Wang, X., & Koenig, S. (2013). *Ridesharing: The state-*

of-the-art and future directions. *Transportation Research Part B: Methodological, 57*, 28–46.

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

8. J Ferreira, P Trigo, P Filipe - … Journal of Humanities and Social Sciences, 2009 . Collaborative car pooling system