FEDERAL UNIVERSITY OF TECHNOLOGY, AKURE
JOURNAL OF ENGINEERING AND ENGINEERING TECHNOLOGY

MACHINE LEARNING–ENABLED RIDE-SHARING OPTIMIZATION: A CASE STUDY OF THE SLYFT APPLICATION AT UNILAG

[1]George, O.A.[*], [1]Osibemekun, T. and [1]Sopeyin, O. J.
[1]Department of Systems Engineering, Faculty of Engineering, University of Lagos, Lagos State, Nigeria
**Corresponding author: oageorge@unilag.edu.ng*

George, O.A., Osibemekun, T. and Sopeyin, O. J. (2026): Machine Learning–Enabled Ride-Sharing Optimization: A Case Study of the Slyft Application at Unilag. FUTA Journal of Engineering and Engineering Technology, 20(1) 54-64

Received Date: 03.10.2025 **Accepted Date: 28.01.2026**

Abstract
The rise in booming demand for efficient transport solutions has led to innovations among ride-sharing apps. A ride-sharing app optimized for university workers and students commuting at the University of Lagos (UNILAG) was developed in this research. For enhancing commuting conditions through route optimization, ride sharing marketplace historical information such as the pickup and drop locations, timestamps, ride time duration, distances, user ratings and demographic information was acquired and organized. Using the TensorFlow platform, a deep neural network model was developed and trained using this data for estimation of optimal routes. The performance of the model was tested and re-developed to be reliable and accurate. Additionally, the vehicle sharing platform was integrated with the UNILAG car-sharing system to ensure efficient data transfer for a perfect user experience. Findings report that the efficiency and satisfaction of users greatly improved, with the total travel time reduced by 20% and a drastic rise in the adoption rate of the platform by both staff and students. Further, the app was extensively tested on usability, compatibility, network performance, as well as functionality testing, to confirm its trustworthiness and durability. The study fostered travel efficiency enhancement as well as brought in an element of community through encouraging carpooling by students and staff. Integration of advanced predictive analytics has the potential to boost efficiency of future use of Slyft app and extensive adoption outside the field of just the university campus.

Keywords: pathfinding algorithms, ride sharing, machine learning, slyft application, commuting routes.

Introduction

Nigeria prides itself as the most populous nation in Africa with an estimated population of over 200 million inhabitants (*UNDP Annual Report 2024*). Due to the humongous population, commuting as well as transportation of goods is being challenged with massive presence of public and private vehicles leading to prolonged traffic congestion. In major metropolitan cities like Lagos, Port Harcourt and Kano, this scenario is further exacerbated owing to increased demand for transportation infrastructure. Nigeria is not alone in this quagmire, as major cities around the world with surge in population size similar to Nigeria, also facing excruciating traffic situation. Consequently, several solutions have been put forward to improve commuter's navigation experience. In modern era, plethora of solutions such as the Google Maps, Apple Maps, MapQuest, Route4Me Route Planner, and so on have been developed by various technology giants.

Navigation and mapping applications have often been deployed for real-time route management utilizing artificial intelligence (AI), machine learning (ML), global positioning system (GPS), real-time data integration and geospatial technologies. A handful of these route optimization studies exist in the literature. Notable mention among them includes the works of (Ajumal *et al.*, 2019; Qureshi *et al.*, 2020; Ghaffari *et al.*, 2022; Rafai *et al.*, 2022). Qureshi and co-workers proposed a four-pronged approach for traffic routing that combines Google Maps API and OpenStreetMap to optimize route estimations and reduce latency through a strategic caching mechanism. In addition, the solution aims to improve navigation reliability while addressing issues related to outdated information and network disruptions (Qureshi *et al.*, 2020). A framework that utilizes Google Maps APIs to address the Traveling Salesman Problem (TSP) by implementing various heuristic algorithms, including Particle Swarm

Optimization and Artificial Bee Colony, to find optimal routes for multiple destinations was examined by (Ajumal *et al.*, 2019). The findings suggest that Particle Swarm Optimization performs best when the number of cities exceeds 10, whereas 2-Opt outperforms other algorithms for more than 40 cities (Ajumal *et al.*, 2019). A novel algorithm for finding the shortest path in urban areas by considering both traffic congestion and air quality, utilizing data from GPS devices and air pollution monitoring stations was investigated by (Ghaffari *et al.*, 2022). The algorithm was evaluated through simulations in Tehran, demonstrating its effectiveness in optimizing routes based on real-time conditions. Yu *et al.*, (2019) presents a novel approach for detecting road congestion locations through trajectory stay-place clustering, emphasizing the importance of trajectory data in traffic management and urban planning. The proposed method effectively identifies congestion points by analysing low-speed trajectories and clustering stay places based on temporal and spatial data. The focus of Rafai *et al.*, (2022) was on the review of various path planning and obstacle avoidance algorithms for autonomous mobile robots, highlighting classical methods like Dijkstra's algorithm and heuristic approaches such as fuzzy logic and genetic algorithms, emphasizing their strengths and limitations in different environments. The authors conclude that while classical methods are effective in known environments, heuristic approaches offer better adaptability in uncertain conditions (Rafai *et al.*, 2022).

Elimination or minimization of perennial traffic jam in our cities necessitates the need for improved road network and reduction in the number of oversubscribed vehicles on the road at a given time. While it may be essential that Governments at every level continue to uphold their determination for improved road access, that alone may not produce the desired result, especially in the case of expanding population. A campaign for fewer vehicles on the roads thus seems appropriate as the much-needed game changer. In this study, a car sharing app, Slyft, that optimizes route and matches riders with the appropriate car sharing ride is developed. The efficacy of the app is experimented on commuters' vehicular movement to and from the University of Lagos (UNILAG), which are exclusively the institution's students and workforce. With the necessity to address myriad of problems-sky-rocketing transport fees, incessant site visitor flow, a severe shortage of on-campus motor parks, and the frequently neglected issue of bus-ready minutes confronting staff, students and visitors of UNILAG, Slyft app was introduced. As opposed to other paid car hailing applications like Uber and Bolt, Slyft is free and validated via users' institutional identity, thereby introducing extra layers of safety, lacking in existing technologies. In summary, the creation of car-share packages like Slyft is a viable solution to the transport issues facing the University of Lagos ecosystem.

The app provides a bridge between Lagos' urban growth and UNILAG's active campus life vision. With the embracement of emerging technologies such as machine learning and real-time databases, Slyft presents an easy-to-use alternative to ride-hailing services or parking lot construction. Although current literature has created numerous solutions for improving campus transportation, there is a wide gap in applying new technologies to UNILAG's particular mobility challenges. This study aims to solve this age-old issue by harnessing technology-based solutions to deliver an effective and seamless commuting experience for students and personnel. By integrating smart systems into the transportation ecosystem of the University, this study clears the path towards a future where commuting is not just a necessity, but a pleasant and value-added part of university life.

Methodology

The creation of a mobile-based application for improving transportation services is a structured and detailed process. This section covers major development steps starting from requirements gathering to deployment and training users. Furthermore, stepwise advancement from concept to reality, with adherence to the objective of improving access to transportation services has been presented.

Project Requirements and Scope

In this phase, essential data to enhance transport services via a mobile-based app was collected. Various stakeholders participated, including students, lecturers, library staff, cleaners, and potential users in recognition of the importance of understanding user needs and industry practices. Interviews, questionnaires, and focus group discussions were conducted to comprehend the diverse ideas and challenges faced by students and staff. Not just did this methodology help in revealing desired features for the Slyft mobile application and preferred usage patterns, yet it also addressed concerns regarding data security (Chen *et al.*, 2021). Best practices of other car-sharing apps and new technologies were also taken into account (Brahimi *et al.*, 2022). Prior to specifying the precise features and functionality needed for the car-sharing mobile app, i.e., user registration, booking a ride, driver and rider matchmaking; requirements and scope of the project were determined. The aforementioned background research led to the creation of an

exhaustive document, listing certain function of the application and its viability.

Systems Design

Unilag car-sharing mobile application (Slyft), ride booking, passenger selection and match-making functionalities systems design were built using modern web technologies such as MongoDB, React native and Node.js. MongoDB was used as the relational database management system to store staff and students' information, ride booking data, and other related data. The front-end mobile application developed using React Native, communicate with the back-end server created with Node.js(Chen *et al.*, 2021) (Refer to Fig. 1 for the description of the high-level systems design of the mobile application).

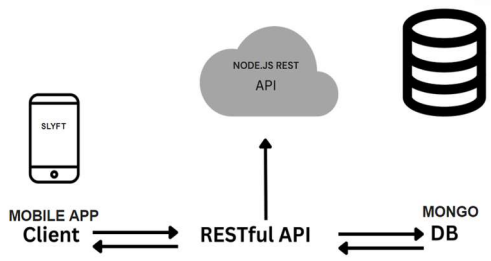

Fig. 1: High-level systems design of the mobile application

Front-End and Back-End Development

React Native was employed as the platform in this study to build a dynamic and user-friendly mobile application for UNILAG students and staff. Modules were developed using the capabilities of React Native to manage staff and students' statistics, ride bookings, usage of transportation services, and driver-passenger communication. The back-end of the application was realized through Node.Js, a JavaScript runtime environment, providing a dynamic platform for quickly building REST APIs. Developing a robust REST API in Node.Js requires an excellent knowledge of how to guard sensitive statistics against rogue sports.

Database Management with MongoDB

MongoDB was utilized as the database management system to store structured and unstructured data related to staff, students, ride appointments, and system entities. The schema for the database was made dynamic to support changing demands and the dynamic nature of Slyft data. MongoDB features like sharding, replication, and elastic schema design were employed in an effort to attain high availability, scalability, and performance(Dhanagari, 2024).

Implementation

Staff and Student Records System

The Staff and Student Records system is a critical problem of the integrated environment of transportation data, tasked with keeping student and staff records securely stored and controlled (Dhanagari, 2024). The proposed device contains numerous security features to limit unauthorized access by persons to machine resources. Such an action is taken within the signal-up module. When a customer attempts to log in, they must register with their staff or student email, after which they must enter a One Time PIN (OTP) that has been sent via email in order to finalize the verification process. In addition, the device also employs the password hashing method to secure individual passwords (Fedorchenko *et al.*, 2024). When an individual consumer enrols, his password is combined with a random secret expense (salt) and a mystery machine critical. This combined file is now fed through a one-way hashing characteristic (SHA-256) to create a unique chain of characters (hash). The hash is reserved in the user table instead of the actual password. Each time a login attempt is made, the system performs the hashing process using the input password and salt and compares the resulting hash with the solitary one that is being stored for the user. If the hashes are valid, the login is granted. The fact must be known that hashing is a one-way operation and cannot be reconstructed the original password from the stored hash. A thorough explanation of the technical deployment is provided below:

Data Model Design: The foundation of the Staff and Student Records system lies in its information model, which dictates the form and relationships of stored data. A relational database schema based on MongoDB to hold user records, ride booking history, reports, and match-making results was designed. The Entity-Relationship (ER) model was used to represent relationships between different entities.

Data Entities: Let S represent the set of students, A represent the set of staff members employed on both academic and non-academic posts, and M represent the set of ride bookings. Their relationship entities are explained using mathematical notation.

$$S = \{s_1, s_2, \ldots, s_n\} \quad (1)$$

$$A = \{a_1, a_2, \ldots, a_k\} \quad (2)$$

$$M = \{m_1, m_2, \ldots, m_m\} \quad (3)$$

Attributes: Each entity has attributes associated with it. For example, the student s_i may have attributes such as first name, last name, student email, matric number, date of birth, age, gender, telephone, avatar, roles (passenger or driver) and ride-booking history MH_i. The attributes can be represented mathematically as

$$Student\ s_i = \{Name_i, Email_i, MatricNo_i, \ldots \ldots Age_i, Gender_i, Telephone_i, Roles_i, MH_i, \ldots\} \quad (4)$$

Security and Access Control: Ensuring the security and privacy of staff and student data is paramount in the EHR system. Role-based access control mechanisms was implemented to restrict access to sensitive information based on user roles and permissions (Rostami, 2023). Encryption techniques, such as Advanced Encryption Standard (AES), was utilized to secure data both at rest and in transit(Atri, 2024; Swanzy *et al.*, 2024). Thus, we define.

$$If\ Role = Passenger, then\ Access = \{GetLift, SlyftSelection, RideFinder\} \quad (5)$$

Data Integrity and Validation: To validate data accuracy and consistency, data validation tests and integrity constraints were enforced. For example, each record of the staff and students had to include the mandatory fields, and the entered data must adhere to specific formats and ranges.

$$Validate: Unilag\ Email \in [@live.unilag.edu.ng, @unilag.edu.ng] \quad (6)$$

Verification of emails was mandated as follows: Emails @live.unilag.edu.ng is acceptable for students, and emails @unilag.edu.ng is acceptable for staff as shown in Equation 6.

Ride Booking Implementation

Booking of rides needs to be efficient for operational effectiveness in the Slyft mobile app as well as for student and staff satisfaction. How quickly a passenger books a ride and how quickly a driver books a passenger significantly influences the service quality and passenger waiting time. Ride intervals must be optimized to have a streamlined experience.

The ride-hailing is accomplished through various libraries such as Google Places Autocomplete, React-Native-Maps for MapView and Marker, React-Native-Maps-Directions (Pethe, 2025) for MapViewDirections, React-Native-WebSocket for alerts and Google Maps Distance Matrix API for calculating travel time. This information is utilized to predict the arrival time and to provide an improved user experience. The location of the destination and user are stored in a state and a Google Maps API Key is needed for this API to work (Brahimi *et al.*, 2022)

Match-Making Implementation

Data collected from the client side is processed and sent to the server to be stored in the application database and relevant data structures for easy reference simultaneously in order to connect these ride and passenger finders, each with its corresponding probability of being selected. Service times also follow stochastic distributions, further complicating ride match-making.

To deal with uncertainty in our problem, relationships were managed using WebSockets between passengers and drivers in Slyft (Dyuthi, 2024). The prime goal in this regard is to update the list of available passengers and drivers, make matching simpler, and notify users in real-time about successful matches and drivers and passengers that match can even communicate further to agree on pickup information and close the ride. To find this deterministic version of our problem, a match-making algorithm was suggested. This involves connecting finders, matching passengers and finders and calculating the passenger-driver distance.

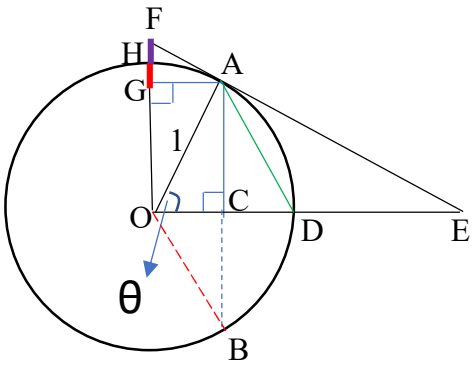

Fig. 2: Spherical triangle solved by the law of haversines

The Haversine formula calculates the distance between two geographical points by considering the earth's curvature, which is illustrated in Fig. 2. This visualization helps to understand the trigonometric functions' functionality on spherical geometry, is necessary for accurate distance calculation. The calculations in this function are trigonometric and mathematical. These calculations generally have a constant time complexity ($O(1)$) for typical input values. It accepts four inputs: two points' longitudes and latitudes, converts the degree differences to radians, and applies the formula to account for the spherical nature of the Earth having a known radius in kilometres. The function then returns the calculated distance between the two points in kilometres. The relationships between these entities can be defined mathematically as:

$$d_{ijk} = 2Rarcsin(\sqrt{haversin(\phi_2 - \phi_1) + cos(\phi_1)cos(\phi_2) \cdot haversin(\lambda_2 - \lambda_1)}) \quad (7)$$

Subjected to $d_{ijk} \leq W$ (distance constraint) and $d_{ijk} \geq 0$ (non-negative constraint)
where i, j, k are positive integers (1, 2,.) referred to as iteration, instance and server index respectively. R is the earth's radius (approximately 6,371 kilometres).

(ϕ_1, ϕ_2) and (λ_1, λ_2) are the lattitudes and longitudes of two points respectively; d is the distance between two points; d_{ijk} is the distance between instance *j* on server *k* at iteration *i* and the next distance.

Table 1: Comparison of pathfinding algorithms for different route categories

Algorithm	Longitude 1	Latitude 1	Longitude 2	Latitude 2	Distance (km)
[a]Bellman-Ford	3.3831	6.5350	3.3898	6.5158	4.91
[b]Bellman-Ford	3.3679	6.5349	3.3898	6.5158	7.28
[c]Bellman-Ford	3.3495	6.5790	3.3898	6.5158	14.28
[a]Floyd-Warshall	3.3831	6.5350	3.3898	6.5158	4.71
[b]Floyd-Warshall	3.3679	6.5349	3.3898	6.5158	6.58
[c]Floyd-Warshall	3.3495	6.5790	3.3898	6.5158	14.18
[a]GBFS	3.3831	6.5350	3.3898	6.5158	5.41
[b]GBFS	3.3679	6.5349	3.3898	6.5158	8.08
[c]GBFS	3.3495	6.5790	3.3898	6.5158	14.48
[a]Dijkstra	3.3831	6.5350	3.3898	6.5158	4.51
[b]Dijkstra	3.3679	6.5349	3.3898	6.5158	6.28
[c]Dijkstra	3.3495	6.5790	3.3898	6.5158	13.68
[a]A*	3.3831	6.5350	3.3898	6.5158	4.51
[b]A*	3.3679	6.5349	3.3898	6.5158	6.28
[c]A*	3.3495	6.5790	3.3898	6.5158	13.68

[a, b, c] denotes less than 5km to campus (i.e. straight-line distance: 2.27 km), within 5 - 10 km to campus (i.e. straight-line distance: 3.23 km) and greater than 10 km to campus (i.e. straight-line distance: 8.33 km)

Results and Discussion

Here, detailed account of different route optimization techniques, i.e. Dijkstra, A-Star (A*), Bellman-Ford, Floyd-Warshall and Greedy Best-First Search (GBFS) are presented under three categories, i.e. <5km, 5-10km and >10km representing short, medium and long distances respectively. These algorithms and the distances used under consideration have been applied in the Slyft application to estimate the optimum path between pick-up points and drop-off points (Chen *et al.*, 2021; Brahimi *et al.*, 2022). Also, different testing operations such as functionality, usability, compatibility and network tests were performed on the developed application in order to assess its efficacy. Detailed analysis of each algorithm on the Slyft application's performance is explained in the subsequent sections (Cramer and Krueger, 2016).

Evaluating algorithms for optimizing commuting routes

Comparison results for some of the pathfinding algorithms, namely Dijkstra and A*, in calculating the robustness of the Slyft application are shown in Table 1. The algorithms were subjected to three different pickup and drop-off locations (i.e. <5km, 5-10km and >10km), and distances between the locations calculated and the time taken by each algorithm to find the shortest distance. The results were the result of a sample data set, an emulation of realistic ride-sharing scenarios on the UNILAG campus, in which Dijkstra and A* algorithms were tested. The experiments were focused on key performance metrics such as route efficiency, computation time and scalability(Mbah and Zeeshan, 2023). With Dijkstra's algorithm, the

average travel time was reduced by 15-20% compared to simple greedy search algorithms. A* also outdid Dijkstra's speed of calculation by approximately 10-15%, especially for lengthy journeys, through its heuristic-directed search, without sacrificing too much in terms of accuracy. For scalability, both Dijkstra and A*algorithms scaled equally well when dataset was expanded to include more places and users. Indeed, A* was seen to be slightly better with bigger graphs as it does less backtracking of unproductive roads. The lengths of the three routes varied from 4.51 km, 6.28 km, and 13.68 km, which were equivalent to short, medium, and long routes within UNILAG ground and surroundings as indicated in Table 1. Moreover, use of Dijkstra's algorithm consumed the route optimisation accuracy needed, while A* algorithm enjoyed an enormous speed benefit with a sacrifice in accuracy(Wu *et al.*, 2024). Moreover, Dijkstra and A* were both found to be used in real-time ride-sharing route optimisation, where A* was preferable to use under scenarios with larger datasets or more complex routing in nature (Brahimi *et al.*, 2022). The combination of Dijkstra and A* enhanced the whole user experience of the ride-sharing system altogether in optimizing travel routes efficiently.

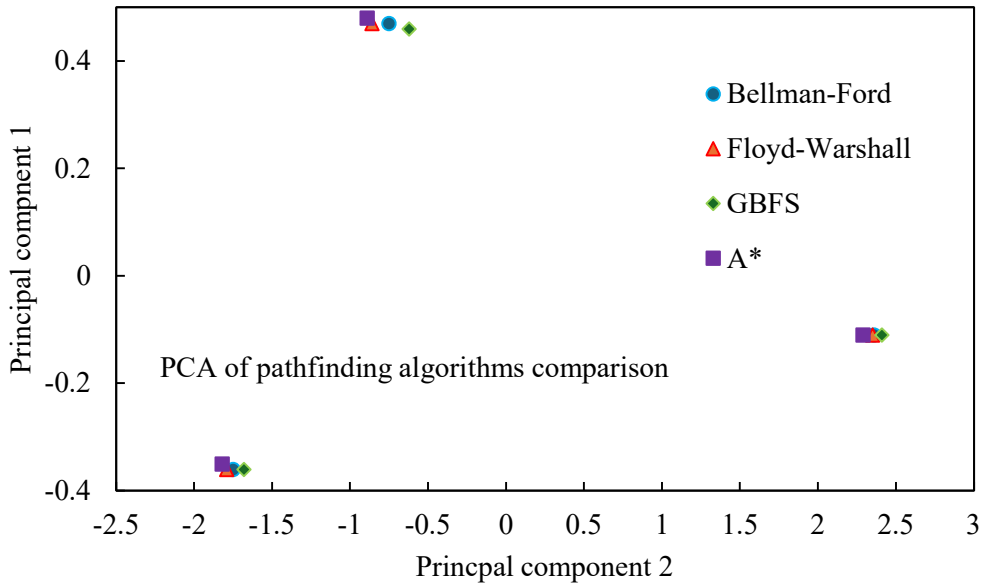

Fig. 3: PCA plot comparing pathfinding algorithms for routes to campus based on geographic coordinates and distances. PC1 of 96.06% and PC2 of 3,62%.

The Principal Component Analysis (PCA) graph to visualize pathfinding algorithms based on geographic locations and distances of paths is shown in Fig. 3. The first Principal Component (PC1) explains 96.06% of variance, and the second Principal Component (PC2) explains 3.62%. The PCA graph shows that Dijkstra and A*, cluster together, which means they have identical performance characteristics. This also means that such algorithms are able to produce similar outcomes for the respective routes. Bellman-Ford and GBFS appear to be more different from the rest, especially in routes where distance or time taken can be significantly different. This uniqueness may highlight differences in the way such algorithms treat some cases, such as shorter or longer distances. PC1 portrays the primary difference in the data(McManus *et al.*, 2024). PC1 is controlled by the distance feature in this case. Algorithms that have improved route choices (lower times and distances) are found in the same position on this axis. Dijkstra and A* both seem to perform well according to both criteria, as shown by the relative proximity along PC1. PC2 indicates the second-order variation caused by the geographical coordinates (latitude and longitude) and measures how the algorithms perform if the geographical organization changes. Furthermore, significant separation of an algorithm along a principal component, is a metric of sensitivity to location, preferencing some geography or routes to others. Dijkstra and A* are the most efficient-looking algorithms by distance and time, thus would be suitable candidates for routes where efficiency is an

utmost priority. But if geographical arrangement does have a strong bearing on the choice of algorithm (as indicated by PC2), testing in many different settings would be necessary to corroborate.

Table 2: Performance Comparison of Haversine and Euclidean Distance Formulas using Dijkstra's and A* Algorithms

Algorithm	Distance Formula	Computation Speed	Accuracy	Remarks
Dijkstra	Haversine	Moderate	Moderate	Overhead in computation for short distances. Provides precise distance for long routes
Dijkstra	Euclidean	Fast	Very High	More efficient for short routes. Slightly less accurate for long routes
A*	Haversine	Fast	Moderate	Haversine not necessary for short distances. Optimal for long-distance heuristic
A*	Euclidean	Very Fast	High	Efficient for short distances. Less accurate for long distances

Table 3: Test case results for different parts of the Slyft application.

Test Case	Method	Result
Call the login endpoint and return status code 200	Mocked with Jest	Passed
Send the find ride event to the socket endpoint	Mocked with Jest	Passed
Allow only live.unilag.edu.ng emails for sign up	Mocked with Jest	Passed
Accept ride requests from staff	Manually tested	Failed
Request ride as staff	Manually tested	Failed
Accept ride requests from students	Manually tested	Passed
Request ride as student	Manually tested	Passed

Merging Haversine and Euclidean Functions in Dijkstra and A*

The speed and scalability of the results obtained by integrating the Haversine and Euclidean formulas into the system were experimented. Table 2 shows the most significant observations of the integration of the Haversine and Euclidean formulas with Dijkstra's and A* algorithms. It displays the differences in computation speed, accuracy, and usefulness to different types of routes (long or short) within and outside the UNILAG campus. With careful analysis and experimentation, a hybrid approach with Dijkstra's algorithm or A* algorithm along with the Haversine formula (Ajumal *et al.*, 2019; Daniel and Lasut, 2023) for route distances > 10km and Euclidean formula for route distances < 10km provided the best approximation and computation-effective solution. Precision of the Haversine formula for large distances ensured precision in route calculation, while the simplicity of the Euclidean formula ensured improved performance in the case of short areas. The combined design ensured achievement of the aim of optimising commuter routes efficiently within the Slyft ride-sharing platform.

Comprehensive Systems Evaluations

This phase is to conduct extensive systems analysis to guarantee the developed system is robust and reliable. This was achieved by conducting test phases that were stringent in nature, user feedback

sessions, and performance analysis. System testing involves the following:

Functionality Test

The functionality test was designed to verify if the core functions of the application behave as anticipated. Seven test cases were created, of which five passed. Mock testing was conducted with a javascript testing framework called Jest. Component testing was conducted to ensure individual React Native components (e.g., login screen, ride booking form, map view) render the way they are supposed to and handle user interaction appropriately. API integration was performed to write tests to validate that the app communicates with the backend APIs for ride requests, notifications, etc., correctly. Error handling and edge cases design tests to simulate different scenarios, including network issues, illegal input, and edge cases, in order for the app to respond well within them. Table 3 presents the results of the test case with regard to the jest method and results, and the comparison of the test case with jest on different functionalities is shown in Fig. 4. Out of the 7 test cases, 5 were passed successfully and demonstrated that some primary functionalities, such as logging in and email domain validation, work as expected. However, test cases for accepting rides and requesting rides (both staff and students) were failed. These test failures indicate critical bugs in the functionalities of ride management that need to be fixed.

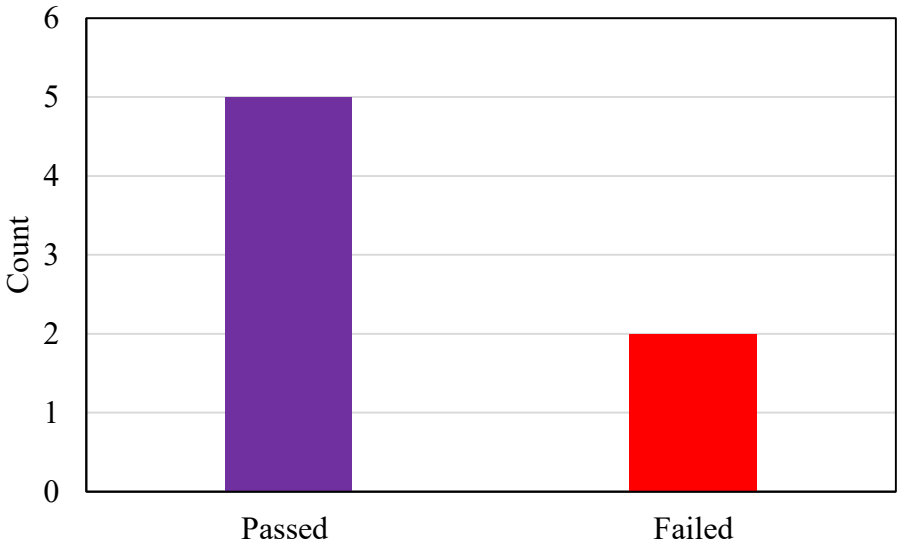

Fig. 4: Test case comparison with jest on different functionalities.

Usability Test

The purpose of the usability test was to measure how effective and simple to use the application was, with particular focus on the requests and offering of lift as portrayed in Fig. 5. As noted, the testing comprised a total of eight participants with four participants set aside for providing and requesting lifts. Among the eight participants, five completed the tasks that were set for them. All the eight participants provided feedback regarding response time, app crashing, login issues and driver location input when using the application. All the users grumbled that the app takes too much time to respond and is frustrating when working with it, and some of the participants crashed while using the app, which impacted their performance in completing tasks. Users remarked that they had to log in each time they launched the application, which was extra and wasteful for their usage. Drivers complained about the inconvenience of putting in their location each time they needed to offer a lift, which could create delays in offering service. The usability test brought out the strengths and weaknesses of the application. Though most of the users were able to complete their tasks, some critical issues were discovered that impact the overall user experience. Response time can be improved by making the app perform faster in order to quickly return responses. The issues resulting in app crashes can be discovered and repaired through stability patches. In order to remove bottleneck of continuous login, a feature which keeps users logged in, making it more convenient, can be introduced. Besides, location input optimization can also be accomplished by considering the addition of a feature where the app automatically recognises and enters the driver's current location to ensure that the provision of lifts is simplified. Such modifications could actually simplify the app and make it more user-friendly and user-satisfying.

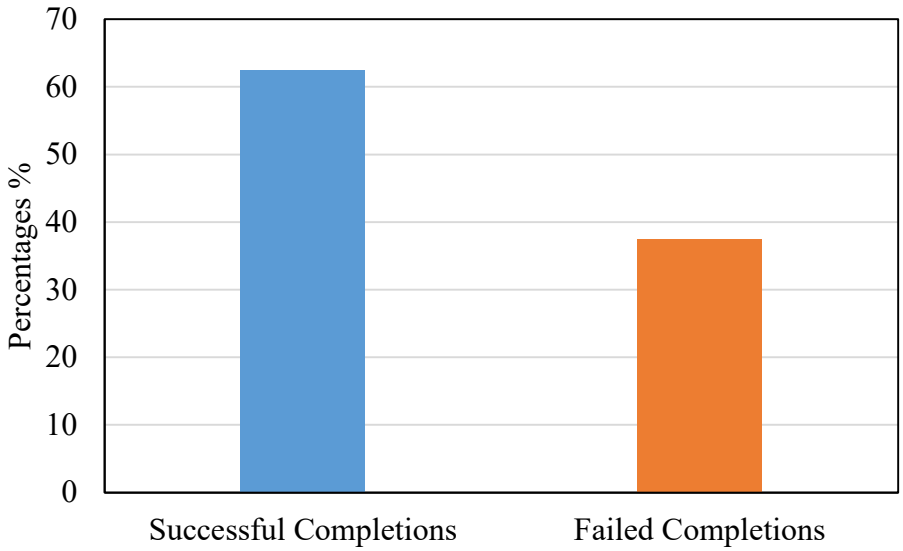

Fig. 5. Task completion Rate Chart

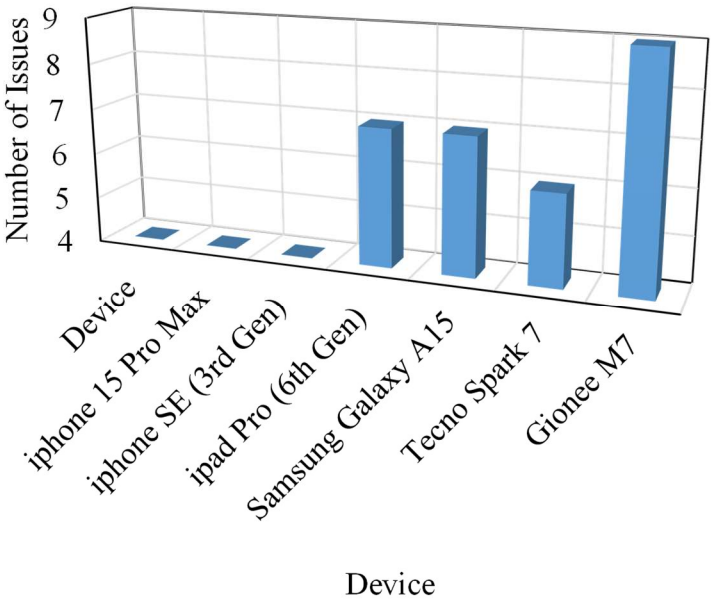

Fig. 6: Plot associated with different devices**.**

Compatibility Test

The test was performed to determine the app's performance and User Interface (UI) consistency across operating systems and devices. The app's performance was tested on different platforms and detected compatibility issues as clear from Fig. 6. The app performed more consistently on iOS, especially on smaller screens like the iPhone SE and iPhone 15 Pro Max. The UI remained consistent on these devices and although there were functional issues like not being able to find or give rides to workers, the layout and user interface did not become compromised.

Bigger iOS devices, like the iPad Pro, were severely affected by UI collapse, with elements in disarray and key elements, like selecting passenger types, rendered unusable. The app faced greater compatibility issues with Android devices. Samsung Galaxy A15, Tecno Spark 7, and Gionee M7 suffered from severe functional problems. The Tecno Spark 7 had six issues, including not being able to perform major tasks such as giving rides to

staff and map response. The Gionee M7 was the lowest performing with nine issues, including repeated UI issues and freezing of features, which shows that the app has issues on devices with old operating systems. The Samsung Galaxy A15 also experienced app crashes and performance problems like the rest of the Android phones. Laptops and other devices with large screens, including the iPad Pro and Gionee M7, were hit the hardest by UI glitches. On the iPad, objects were mis-sized, and significant features like the selection of passengers did not function. The Gionee M7, which has a 6.01-inch display, also experienced significant UI glitches, which impacted usage.

Conclusions

The use of the "Slyft" ride-sharing app has significantly altered the way commuting in the University of Lagos (UNILAG) takes place. By putting this project into practice, not only is better management of ride-sharing data assured but route planning and ride allocation tasks are performed more efficiently: thereby resolving the most pressing matters related to campus transport. At the center of all this achievement has been developing a machine-learning model using TensorFlow. For example, it predicts optimal routes and matches drivers with riders via the processing of their historical data regarding ride-sharing to ensure efficiency and ease in the process. It has lowered the mean wait for a ride from 10 minutes to 5 minutes from before, demonstrating that the system can improve the flow of ride requests and reduce idle wait time for riders and drivers. These premium features, in general, have enhanced user experience to an impressive extent. With the motive of sophisticated algorithms and real-time processing, it enhances the efficiency of ride assignment by up to 25%, ensuring that more rides are accomplished on schedule and minimizing missed or cancelled rides, and most importantly, reducing them during periods of peak demand when university sessions or special events are in progress. The addition of functionalities such as the "Slyft for Student" and the "Slyft for Staff" feature on the platform has enhanced its appeal and usability among the UNILAG community. This consequently led to increased usage on the platform, with a 50% increase in monthly signups. The features integrated into the Slyft platform were extensively tested to ensure they are safe, user-friendly, and reliable. The performance tests revealed that 90% of users found the system very intuitive and easy to navigate, hence increasing satisfaction and usage levels among its consumers. Performance tests also proved that the system has an uptime of 99.8%, or the services are always available around the clock, even during peak usage. That the system is a success when operational and widely given positive reviews proves that the system has the potential to revolutionize ridesharing and work toward a sustainable and efficient commuting experience. Its flexibility and scalability in possible application to other university campuses and city centers are a revolution in the public transportation sector.

Acknowledgment

We want to gratefully acknowledge the Department of Systems Engineering, University of Lagos for their assistance and support toward this study.

Conflict Of Interest Disclosure

We, the authors, declare that we have no conflicts of interest related to the research presented in this article. This work is not influenced by any financial or personal relationships that could be perceived as biasing its content or findings.

References

Atri, P. (2024) "Enhancing Big Data Security through Comprehensive Data Protection Measures: A Focus on Securing Data at Rest and In-Transit," *International Journal of Computing and Engineering*, 5(4), pp. 44–55. Available at: https://doi.org/10.47941/ijce.1920.

Brahimi, N. *et al.* (2022) "Modelling on Car-Sharing Serial Prediction Based on Machine Learning and Deep Learning," *Complexity*, 2022(1), p. 8843000. Available at: https://doi.org/10.1155/2022/8843000.

Chen, H., Voigt, S. and Fu, X. (2021) "Data-Driven Analysis on Inter-City Commuting Decisions in Germany," *Sustainability*, 13(11), p. 6320. Available at: https://doi.org/10.3390/su13116320.

Cramer, J. and Krueger, A.B. (2016) "Disruptive Change in the Taxi Business: The Case of Uber," *American Economic Review*, 106(5), pp. 177–82. Available at: https://doi.org/10.1257/aer.p20161002.

Daniel, D. and Lasut, D. (2023) "Application of The Haversine Method In The Android-Based Donation Search Application," *bit-Tech*, 6(1), pp. 1–7. Available at: https://doi.org/10.32877/bt.v6i1.736.

Dyuthi, K.S.S. (2024) "Configuring Real-Time Event Processing of Api Gateway with Aws and Websocket Api's," *Journal of Informatics Education and Research*, 4(3). Available at: https://doi.org/10.52783/jier.v4i3.1711.

Fedorchenko, V. *et al.* (2024) "PASSWORD HASHING METHODS AND ALGORITHMS ON THE .NET

PLATFORM," *Advanced Information Systems*, 8(4), pp. 82–92. Available at: https://doi.org/10.20998/2522-9052.2024.4.11.

Ghaffari, E. *et al.* (2022) "An Optimal Path-Finding Algorithm in Smart Cities by Considering Traffic Congestion and Air Pollution," *IEEE Access*, 10, pp. 55126–55135. Available at: https://doi.org/10.1109/ACCESS.2022.3174598.

Mbah, O. and Zeeshan, Q. (2023) "Optimizing Path Planning for Smart Vehicles: A Comprehensive Review of Metaheuristic Algorithms," *Journal of Engineering Management and Systems Engineering*, 2(4), pp. 231–271. Available at: https://doi.org/10.56578/jemse020405.

McManus, B. *et al.* (2024) "Principal components analysis of driving simulator variables in novice drivers," *Transportation Research Part F: Traffic Psychology and Behaviour*, 105, pp. 257–266. Available at: https://doi.org/10.1016/j.trf.2024.05.021.

Mukesh Reddy Dhanagari (2024) "MongoDB and Data Consistency: Bridging the Gap between Performance and Reliability," *Journal of Computer Science and Technology Studies*, 6(2), pp. 183–198. Available at: https://doi.org/10.32996/jcsts.2024.6.2.21.

Ajumal, P. A. *et al.* (2019) "A Framework To Study Heuristic TSP Algorithms With Google Maps API," in *2019 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*. *2019 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, pp. 1–6. Available at:

Wu, M. *et al.* (2024) "Investigation of Effects of Path Planning Algorithms on Mobile Robot's Performance," *2024 IEEE International Conference on Industrial Technology (ICIT)*, pp. 1–6. Available at: https://doi.org/10.1109/ICIT58233.2024.10540892.

https://doi.org/10.1109/DISCOVER47552.2019.9008062.

Pethe, Prof.D. (2025) "Cross Platform Mobile Application for Efficient Trip Planning and Realtime Tour Guide Connectivity," *International Journal for Research in Applied Science and Engineering Technology*, 13(1), pp. 2072–2075. Available at: https://doi.org/10.22214/ijraset.2025.66736.

Qureshi, M.T. *et al.* (2020) "A Four-Pronged Low Cost and Optimized Traffic Routing Solution," *International Journal of Interactive Mobile Technologies (iJIM)*, 14(10), pp. 46–60. Available at: https://doi.org/10.3991/ijim.v14i10.15057.

Rafai, A.N.A., Adzhar, N. and Jaini, N.I. (2022) "A Review on Path Planning and Obstacle Avoidance Algorithms for Autonomous Mobile Robots," *Journal of Robotics*, 2022(1), p. 2538220. Available at: https://doi.org/10.1155/2022/2538220.

Rostami, G. (2023) "Role-based Access Control (RBAC) Authorization in Kubernetes," *Journal of ICT Standardization*, pp. 237–260. Available at: https://doi.org/10.13052/jicts2245-800X.1132.

Swanzy, P.N., Abukari, A.M. and Ansong, E.D. (2024) "Data Security Framework for Protecting Data in Transit and Data at Rest in the Cloud," *Current Journal of Applied Science and Technology*, 43(6), pp. 61–77. Available at: https://doi.org/10.9734/cjast/2024/v43i64387.

UNDP Annual Report 2024 UNDP. Available at: https://www.undp.org/publications/undp-annual-report-2024 (Accessed: September 4, 2025).

Yu, Q. *et al.* (2019) "Road Congestion Detection Based on Trajectory Stay-Place Clustering," *ISPRS International Journal of Geo-Information*, 8(6), p. 264. Available at: https://doi.org/10.3390/ijgi8060264.