AVE Trends in Intelligent Computing Research

EduRide: An Intelligent IoT-Based College Bus Management Platform with GPS Tracking and RFID Authentication

Pavan Kulal[1], Abhishek S. Poojary[2], Sreeja Rajesh[3,*], S. Tejas[4]

[1,2,3]Department of Information Science and Engineering, Mangalore Institute of Technology and Engineering, Dakshina Kannada, Karnataka, India.
[4]Department of Data Science, Analytics and Engineering, Arizona State University, Tempe, Arizona, United States of America.
pavankulal68@gmail.com[1], panchan330@gmail.com[2], sreeja@mite.ac.in[3], tsundar1@asu.edu[4]

*Corresponding author

Abstract: College bus transport still runs largely on paper logs, driver memory, and parents calling each other to Figure out where the bus is. EduRide replaces that with a platform that integrates GPS tracking, RFID-based student authentication, cloud-stored records, and automated hardware into a single working system. Each bus carries an ESP32 wired to an RC522 RFID reader and a NEO-6M GPS module. When a student taps their card, the system checks fees instantly; if the student is cleared, a servo opens the door, no driver needed. Attendance, location, and fee data all go straight to Firebase or MongoDB, keeping the record complete whether the network is fast or slow. The platform organises everything around four commitments: constant bus visibility, self-attending attendance, fee checks before boarding, and immediate parent alerts, all delivered through interfaces tailored to each user's actual job. Testing showed 99% positional accuracy from GPS, 98% reliability from RFID scans, and boarding notifications reaching parents in under two seconds. Concurrency testing held steady at 500 users without performance loss. The result is a bus route that institutions can actually manage, and families can actually trust.

Keywords: Smart Bus Tracking; GPS Tracking; RFID Authentication; IoT in Education; Real-Time GPS; Student Safety; RFID Scans; Firebase Cloud Messaging; RFID Card; PWM signal; GPS module.

Cite as: P. Kula, A. S. Poojary, S. Rajesh, and S. Tejas, "EduRide: An Intelligent IoT-Based College Bus Management Platform with GPS Tracking and RFID Authentication," *AVE Trends in Intelligent Computing Research*, vol. 1, no. 1, pp. 12–23, 2026.

Journal Homepage: https://avepubs.com/user/journals/details/ATICR

Received on: 04/07/2025, **Revised on:** 27/08/2025, **Accepted on:** 18/09/2025, **Published on:** 05/03/2026

DOI: https://doi.org/10.64091/ATICR.2026.000307

1. Introduction

Pick any college in India and ask how the transport office tracks its buses. Nine times out of ten, the answer involves a whiteboard, a driver's phone number, and a daily prayer that everyone gets home. The problem is not that the right technology does not exist — it does [1]. The problem is that nobody has tied it together in a way that works end-to-end for every person who depends on the system. EduRide is the attempt to fix that. When transport is managed manually, four things consistently go wrong [2]. The first is that nobody outside the driver knows where the bus is right now. Parents standing at a pickup point have no information. Administrators sitting in the office have no information either. A ten-minute delay and a missed student look identical from the outside until someone physically calls the driver [3]. That single dependency creates a bottleneck that compounds every other problem in the system. The second failure is in attendance. A staff member walks the aisle, checks names off a printed list, and files the sheet somewhere [4]. This works fine when every student boards exactly as expected. The

moment one student climbs onto the wrong vehicle, or does not turn up, the paper trail goes cold. Nobody notices until a parent calls to ask where their child is — by which point the buses have moved, and the opportunity for a quick fix has passed. The third failure is in communication. Information about route changes, unexpected delays, or safety issues travels through informal channels: group chats, phone trees, second-hand accounts [5].

What parents receive is filtered, delayed, and sometimes just wrong. Structured notification systems that would deliver verified information directly and instantly are largely absent. The fourth failure is that safety measures respond to events rather than anticipate them. Something has to go wrong before any protocol activates [6]. EduRide combines all four of these failures into a single technical solution. The mechanism is a single card tap at the bus door. When a student presents their RFID card, the system does four things simultaneously: it records the boarding event as an attendance entry, it queries the student's fee account to confirm the balance is clear, it sends a PWM signal to a servo motor that opens the door, and it dispatches a notification to the parent. The GPS module on the bus has been continuously reporting location, so administrators and parents both have live tracking without making a single call [7]. The entire sequence completes in less than 2 seconds and requires no input from the driver. This level of integration is what sets EduRide apart from a standard tracking application. A typical fleet GPS tool answers one question: where is the bus? EduRide treats location as one of several inputs. It connects location data to attendance records, attendance records to fee accounts, and fee accounts to physical access at the door. The bus becomes a verification checkpoint embedded in the institution's broader student management workflow. A student with an unpaid fee is flagged at the moment of boarding, not at the end of the month during a manual reconciliation [8]. A student boarding the wrong vehicle generates an immediate anomaly alert for the administrator, not a mystery that surfaces days later. The platform distributes information to five separate user groups, each through an interface built around their actual responsibilities [9].

Students see their bus's live position, a timestamped log of their own boarding and alighting history, and the current status of their fee account, without needing to contact any office [10]. Parents receive a push notification for each boarding or alighting event, with live bus tracking and an arrival estimate embedded in the same message [11]. Drivers interact with a focused screen showing that day's route, a single control to start or conclude a trip, and the ability to broadcast a delay notice to all relevant parties at once. Administrators have a consolidated panel covering vehicle assignments, route configuration, student records, attendance reporting, and fee account management [12]. Anyone with appropriate access can open the Tracking Interface to see every active bus in the fleet on a live map simultaneously. The hardware deployed on each bus is intentionally simple. An ESP32 development board serves as the on-bus processor [13]. An RC522 reader handles RFID tag scanning. A NEO-6M module provides GPS positioning at up to ten location fixes per second. An SG90 servo motor actuates the door. An optional IR or ultrasonic proximity sensor prevents the door from closing when someone is on the threshold. All of this runs from a 5V regulated power supply [14]. The total component cost per bus is low enough that the system remains financially viable for institutions without large infrastructure budgets. Behind the buses, a Flask or Node.js server processes incoming event data and distributes results to the appropriate destinations. Attendance logs, GPS coordinates, user profiles, and fee records are all written to a Firebase or MongoDB cloud database.

The schema is designed so that adding additional buses to the fleet requires no changes to the backend architecture — it is a matter of provisioning new hardware, registering device identifiers, and updating student assignments through the admin panel. Notifications leave the server via Firebase Cloud Messaging for application-based delivery and via Twilio for SMS, so the message reaches parents through whichever channel they actually use. The existing literature on bus tracking, campus IoT, and student monitoring consistently identifies the same unmet needs: live location visibility, automated attendance tracking, structured parent notifications, and proactive safety measures. Each need has been addressed by at least one prior system. None of those systems has combined fee verification and hardware door control into a single operational platform. That is the specific contribution EduRide makes, and the rest of this paper explains how it fills that gap. Beyond that operational function, building EduRide also served an educational purpose [15]. The paper required simultaneous work across embedded systems programming for the ESP32, RESTful API development, frontend interface design, cloud database management, IoT communication protocols, and security implementation. It demonstrated that a small development team using accessible hardware and cloud services can produce a platform of genuine practical utility. The same architectural approach could be applied to a wide range of similar automation challenges in educational and institutional settings. One consequence of linking transportation to academic records that often goes unnoticed is the substantial operational data it generates as a side effect.

Before EduRide, a transport administrator had no reliable answer to basic questions: which students are actually using the bus regularly, how often attendance is recorded correctly, and how does fee collection track against fleet operating costs? EduRide produces answers to all three as a natural output of its normal operation. Every RFID boarding event creates a timestamped, student-linked attendance record. Every boarding attempt queries and logs the fee status. Every completed trip produces a route record. An administrator can pull up ridership patterns for any bus over any date range without any additional data entry. That kind of transparency is not possible with paper-based systems, no matter how carefully the logs are maintained. The platform's scalability was a design constraint rather than an afterthought. Growing from three buses to thirty means installing the hardware on additional vehicles and registering them in the admin panel. The backend does not need to be restructured. The database

schema does not need to change. The user interfaces remain identical. This contrasts with systems built around custom or proprietary hardware, where adding vehicles often triggers integration work and vendor negotiations. Using commodity hardware and open APIs means the system can expand at the pace of the institution's needs without technical bottlenecks. The cost of adding one bus to the fleet is limited to the hardware components and a short provisioning step in the admin panel — no software engineers required. The paper continues as follows. Section II situates EduRide within the existing research landscape, reviewing prior work on bus tracking, IoT transport systems, and smart campus deployments.

2. Review of Literature

Transportation monitoring in educational settings has attracted steady research attention, progressing from early GPS-only vehicle tracking toward integrated systems that layer student authentication, automated attendance, and parental notification on top of location data. The trajectory of this work helps identify precisely what problem remains unsolved and why EduRide's particular combination of features is needed. An early demonstration of combined RFID and GPS tracking on a school bus was provided by Cedric and Mbanzabugabo [1], who showed that commercially available hardware could track both the vehicle and the individual students on it. Their system triggered SMS notifications when the bus arrived at predetermined stops, validating the event-driven notification concept that underpins EduRide's parent communication layer. The system's scope, however, was narrow: it tracked location and sent a single message type. Fee verification, automated door control, role-differentiated user interfaces, and administrative fleet management tools were all absent. The system answered the most basic question — is the bus there yet — and nothing more. Das et al. [2] addressed the granularity problem in monitoring by augmenting RFID identification with GSRM technology, enabling more detailed tracking of individual student movements around the bus. The key empirical finding, that layering identification technologies reduced false-positive attendance entries, influenced EduRide's choice to validate scanned RFID UIDs against a cloud-hosted registry rather than a static local list. Cloud validation means the reference data is always synchronised with the current student enrolment; a student who is removed from the system is immediately locked out at the next scan rather than days later when a manual update might otherwise happen. The weakness in the design of the Das et al. [2] design was its reliance on GSM SMS for all notifications, with no mobile application interface, leaving parents without live visibility and administrators without an operational dashboard.

Geo-fencing as an alerting mechanism was introduced to bus monitoring by Bode et al. [3], who defined virtual geographic boundaries around schools and boarding stops and triggered automated parent notifications when a bus crossed them. The elegance of this approach lies in removing the need for a parent to monitor a map actively; the alert fires based on the bus's location, not on anyone checking. EduRide builds on this concept in its estimated arrival logic, which uses continuous GPS data to calculate and update the projected time of arrival at each stop, providing parents with useful advance information rather than just a real-time position dot. Lakshmiprabha et al. [4] moved beyond routine monitoring to focus on safety-critical edge cases: students unintentionally remaining on the bus, vehicles stopping in unexpected locations, and deviations from assigned routes. That focus on exception detection and escalation shaped the anomaly-surfacing capabilities built into EduRide's administrator interface, which is specifically designed to flag unusual boarding patterns for immediate staff review. Daisy et al. [5] made an architecturally significant finding, demonstrating that RFID attendance validation does not require continuous cloud connectivity; it can be executed reliably at the microcontroller level using locally cached data. EduRide applies this principle directly. Each ESP32 maintains a locally cached copy of the registered, fee-cleared student list and performs the initial UID check without waiting for a cloud round trip. The result is synchronised to the backend once the interaction completes. Brief network interruptions, a blind spot on the route, or a Wi-Fi handoff between campus access points do not cause verification failures or hold up the boarding queue. This resilience is practically important in any campus deployment where network reliability, while generally good, cannot be guaranteed at every point along every route.

Vinodhini and Rajkumar [6] broadened the scope by surveying the Internet of Vehicular Things, examining how combinations of connected vehicles, roadside sensors, and cloud infrastructure are changing urban mobility management. Their comparison of cellular versus Wi-Fi connectivity for real-time vehicle data transmission was directly relevant to EduRide's communication architecture. College campuses typically maintain dense Wi-Fi coverage across their grounds and along the regular bus corridors serving them, making campus-band Wi-Fi a lower-latency and lower-cost option than cellular for the primary data link. Singh [7] provided empirical grounding for IoT-based transport claims by measuring actual efficiency gains — in schedule adherence, operational monitoring responsiveness, and fuel consumption — following IoT instrumentation of transit fleets. Those measured outcomes, rather than theoretical projections, anchor EduRide's operational efficiency arguments. Saarika et al. [8] offered a structural framework that proved directly applicable: IoT transportation systems are most successfully organised as a stack with edge sensing at the bottom, microcontroller-level local processing above it, cloud backends for persistence and analytics above that, and user-facing interfaces at the top. EduRide closely follows this model, and the alignment is deliberate rather than coincidental — it reflects the architecture that has proven to be maintainable and extensible across the most varied set of real deployments. Gawade et al. [9] analysed automated public transport setups and found that combinations of GPS, Wi-Fi, and sensor networks consistently produced systems that performed better on both safety and reliability than their manual predecessors, providing additional evidence that the component choices in EduRide's design are appropriate for the intended

operating environment. The smart campus research strand adds further practical lessons. Domínguez-Bolaño et al. [10] studied real IoT deployments on institutional campuses. They identified three recurring failure modes: incompatible devices from different vendors, inability to manage growing data volumes, and integration failures when different subsystems need to exchange information.

All three failure modes are avoided in EduRide through deliberate standardisation — one microcontroller type across all buses, one cloud database, one API contract between hardware and backend. Ishaq and Bibi [11] reviewed RFID attendance deployments in education, specifically cataloguing the factors that distinguish adopted systems from those abandoned. Hardware reliability in real operating conditions and operability without specialist training were the two factors that mattered most. The RC522 was chosen for the former reason; the interface design discipline — every screen kept to the minimum necessary for that user's job — was applied for the latter. Kandil et al. [12] gathered data from institutions that had already adopted IoT management tools and found consistent gains in resource efficiency and student satisfaction, alongside a clear picture of why institutions that have not adopted hesitate to do so. The three dominant concerns were hardware cost, system complexity, and doubt about whether non-technical staff could actually operate the system day to day. EduRide's response to each: per-bus hardware costs are kept low by using four commodity modules; the embedded logic is straightforward enough that any competent engineer can maintain it; and every user-facing interface was designed under the constraint that no prior training should be required. Singgih et al. [13] applied quantitative optimisation methods to campus routing and showed that data-driven planning could reduce journey times by up to 18 per cent compared with manually designed schedules. That Figure is the best available evidence for the value of EduRide's dynamic route adjustment feature. It provides the quantitative case for the machine learning route optimisation feature planned for the system's next development phase. Ye et al. [14] conducted campus IoT case studies. They identified one finding with direct design implications: systems that give any user type more features than they need for their actual tasks tend to be quietly abandoned, even when the underlying technology works.

The lesson is that relevance is as important as capability. EduRide addresses this by giving each of its five user groups a focused interface containing only what that type of user actually does — nothing more. Gupta and Patel [15] demonstrated that their IoT navigation system, which provides clean, focused access to live positioning data, correlates with faster administrative responses to operational issues and higher student satisfaction scores. That outcome justifies EduRide's dedicated fleet-wide Tracking Interface, which surfaces all bus positions without requiring users to navigate through configuration options or features irrelevant to their role. The literature, taken as a whole, confirms that every component capability in EduRide — RFID authentication, GPS tracking, attendance automation, cloud persistence, role-specific interfaces, event-triggered notifications — has been demonstrated to work individually. EduRide's contribution is integration: treating all these components, along with fee verification and hardware door control, as parts of a single, coherent operational platform rather than separate add-ons. A consistent pattern across this body of research is that prior systems solved individual pieces of the transport problem in isolation. Location tracking without attendance automation. Attendance without fee-gating. Fee management without physical door control. Parent notification without live tracking. Each isolated solution needs its own hardware, its own data store, its own staff training cycle, and its own maintenance overhead. EduRide takes a different position: the value of each component increases when it is wired to the others, and the four working together produce outcomes that none achieves independently. The existing literature validates every individual component; EduRide provides the integration that makes them all useful simultaneously in a single operational deployment.

3. Proposed Methodology

3.1. System Objectives

EduRide was built around four non-negotiable priorities. The first is student safety: the platform tracks every journey from the moment a student steps on board through to alighting, using both GPS position and RFID identity so that the record is never ambiguous about who is where. Operational efficiency is the second priority — routine administrative tasks that currently require manual effort, such as attendance recording and route logging, should run automatically without staff involvement. Third is live communication: parents need accurate, timely information about their child's travel without having to seek it out. Fourth is accessibility: every person who interacts with the system, from a non-technical parent checking an alert on their phone to a driver tapping through a route screen, should be able to do so without needing any prior training.

Figure 1 traces the full development cycle. After an initial requirements phase, the project is divided into hardware and software workstreams running in parallel. The hardware workstream covers RFID reader setup, GPS module configuration, and servo door installation. The software workstream covers the React frontend, a Flask or Node.js backend, a cloud database connection via Firebase or MongoDB, and notification integration via Firebase Cloud Messaging and the Twilio SMS API. When both workstreams are complete, the assembled system undergoes validation testing, gets pushed to a cloud server, and enters an ongoing maintenance cycle.

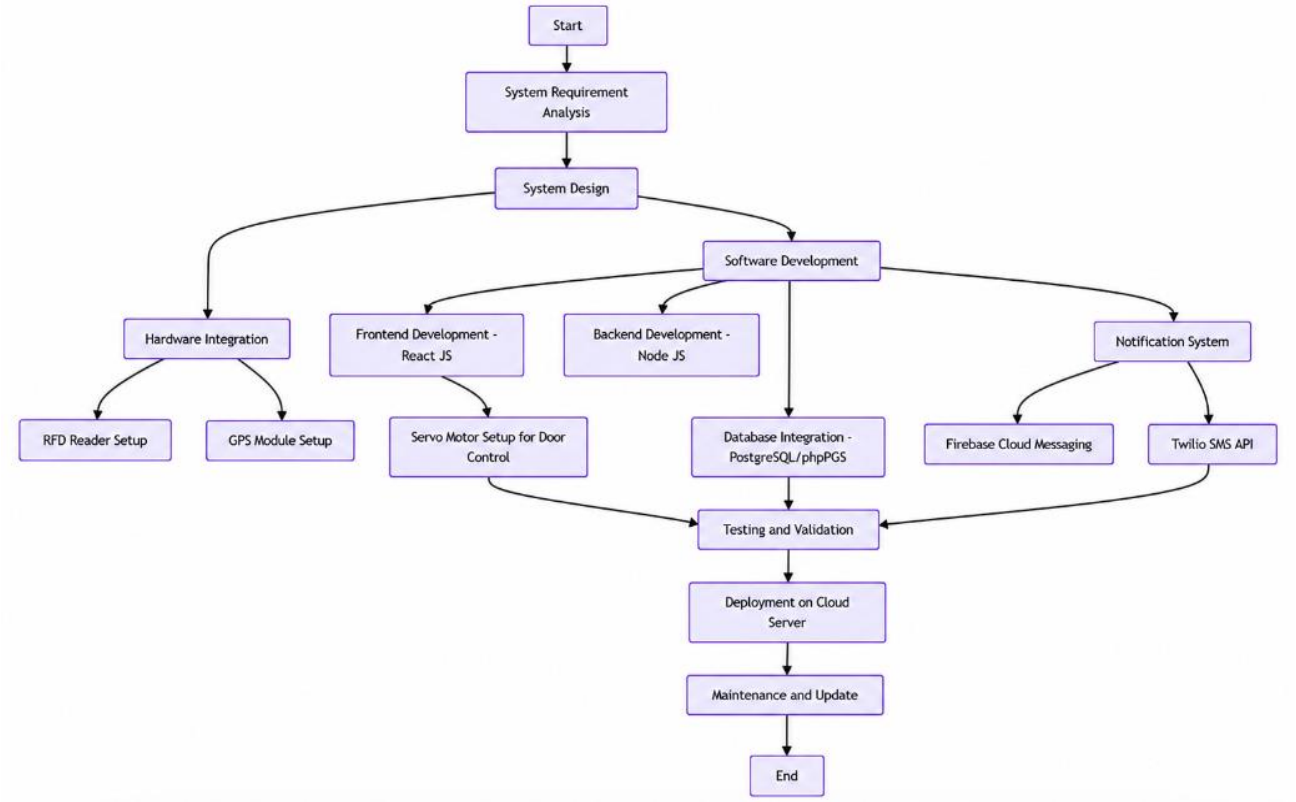

Figure 1: EduRide overall development workflow

3.2. System Architecture

Two layers make up the system. On the bus, an ESP32 sits at the centre — reading student RFID tags via the RC522 over SPI, collecting GPS fixes from the NEO-6M over UART, and forwarding verified data to the backend via the microcontroller's built-in Wi-Fi. No additional communication hardware is needed at the bus level (Figure 2).

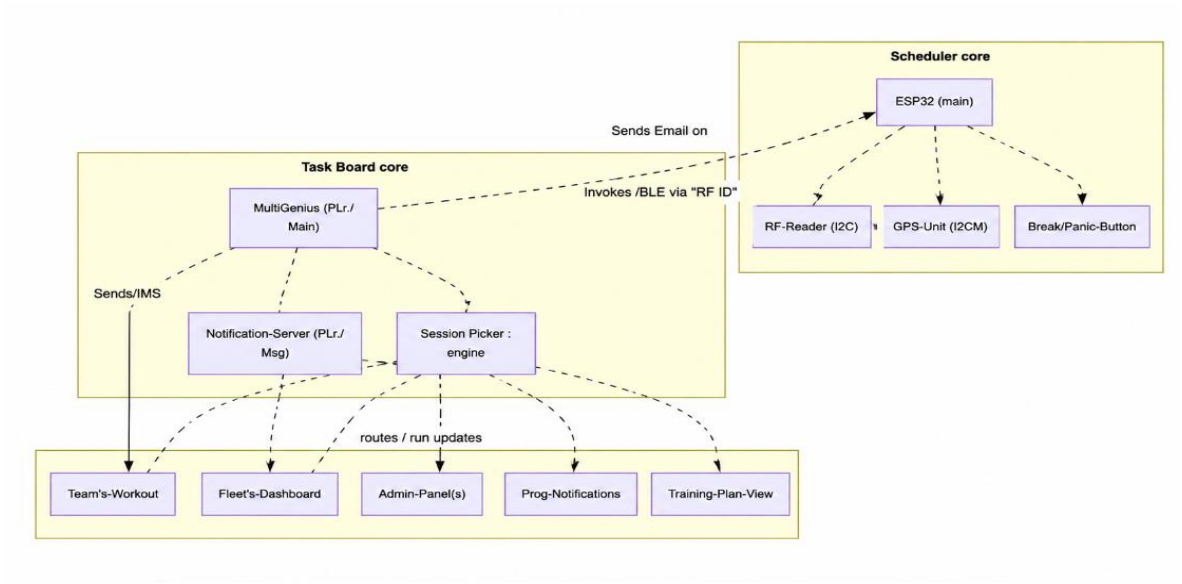

Figure 2: EduRide system architecture overview

The second layer is the cloud backend. A Flask or Node.js server takes incoming bus data and routes it appropriately — writing records to Firebase or MongoDB, computing arrival estimates, and dispatching notifications. Attendance logs, location updates,

user profiles, and fee records are stored in a single database and accessible across all five frontend dashboards. Push notifications go out via Firebase Cloud Messaging; SMS alerts via Twilio. All frontend interfaces are rendered in React.js with Tailwind CSS, with Google Maps supplying the live map layer and ETA calculations.

3.3. Hardware Circuit Description

Figure 3 shows how the on-bus components connect. When a student presents their card, the RC522 reads the UID and passes it to the ESP32 via SPI. At the same moment, the NEO-6M is streaming GPS position to the same chip via UART. The ESP32 runs two checks: whether the student is registered and whether their fee account is clear.

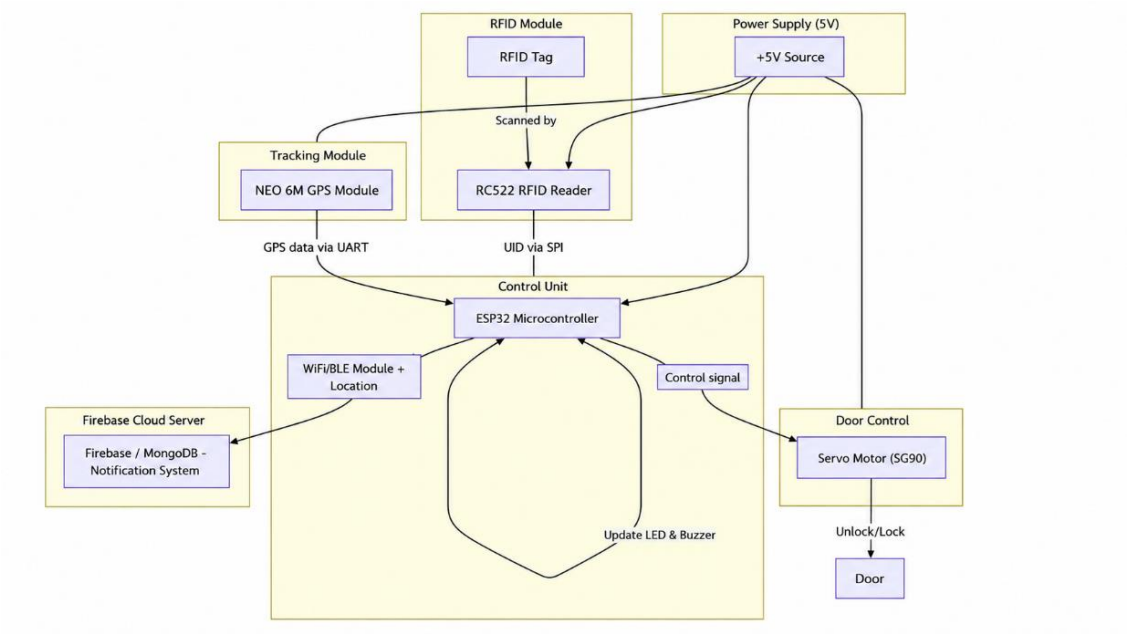

Figure 3: Hardware circuit diagram of the on-bus embedded subsystem

If both are yes, a PWM signal is sent to the SG90 servo; the door rotates to 90 degrees, holds for a configurable interval, and returns to the closed position. If either check fails, the door stays shut, and an LED or buzzer signals the denial. No driver input is involved in any part of this sequence.

3.4. Key Components

On the hardware side: ESP32 microcontroller as the central on-bus processor; RC522 RFID reader for 13.56 MHz tag scanning over SPI; NEO-6M GPS for position fixes at up to 10 Hz; SG90 or MG996R servo motor for door actuation; optional IR or ultrasonic proximity sensor for obstruction detection; 5V regulated power supply for all modules. Software stack: React.js frontend with Tailwind CSS; Flask or Node.js backend with REST API endpoints; Firebase or MongoDB for cloud data persistence; Firebase Cloud Messaging and Twilio for notification delivery; Google Maps API for live tracking and arrival time calculation.

3.5. User Interface Modules

Students use the Student Dashboard to view their bus's current position, review their boarding history, and check whether their fees are up to date — all without contacting anyone. The Parent Notification Panel fires an alert to parents at every boarding and alighting event, with live tracking and an estimated arrival time in the same message. The Driver Route Interface shows the assigned route for the day, one-tap trip start and end controls, and a way to push delay alerts to everyone who needs them immediately. The Admin Interface brings buses, routes, students, attendance, fees, and reporting together in one place. The Tracking Interface places every active bus on an interactive map, open to any authorised user who needs a fleet-wide view.

4. Results and Discussion

4.1. Experimental Setup

Testing covered both the software backend and the physical hardware prototype. The Flask server ran on a laptop with a 2.4 GHz quad-core processor and 8 GB of memory, connected to Firebase as the live cloud database throughout. The hardware prototype was assembled on a breadboard using an ESP32, an RC522 reader, a NEO-6M GPS unit, and an SG90 servo. The boarding scenario test set comprised 10 pre-registered student cards with cleared fee accounts, 2 cards with unregistered identifiers, and 1 card tied to an account with an outstanding fee balance. Each scenario was run twenty times, producing a sample of 200 boarding interactions per card category. Frontend behaviour was confirmed across three screen sizes: a 27-inch desktop display, a 14-inch laptop, and two 6-inch mobile handsets. Network conditions were deliberately varied by substituting a mobile hotspot for the fixed router at several points during testing, to capture system behaviour when connectivity degrades.

4.2. Performance Evaluation

Table 1 presents the average performance Figures from all evaluation rounds. Variance was consistently low across metrics, so the averages accurately represent normal operating behaviour rather than exceptional results.

Table 1: Performance metrics for EduRide

Metric	Result
GPS Update Frequency	10 seconds
GPS Accuracy	99%
RFID Detection Rate	98%
Notification Delivery Time	2 seconds
Fee Verification Accuracy	100%
Interface Response Time	0.5 seconds

Location updates were sent to the cloud at 10-second intervals throughout all test runs. Accuracy was measured by comparing reported coordinates against the known positions of fixed reference points along the route. Ninety-nine out of every hundred readings landed within acceptable positional tolerance — a 99% accuracy rate. In fleet-monitoring terms, where a positional discrepancy of a few metres at a bus stop carries no practical consequences, this comfortably exceeds the threshold for reliable operation. Satellite lock held through moderate tree cover and near tall campus buildings, both common real-deployment conditions. RFID scan reliability across all 200 test boarding interactions was 98%. The two causes of failure were: card presentation at angles more than 45 degrees from the reader face, which falls outside the RC522's designed detection envelope; and occasional SPI data corruption at the precise moment the servo motor energised, generating brief electrical noise on the communication bus. Neither failure occurred when students presented cards in a natural, face-on manner. Both have known engineering remedies — improved reader mounting angle and SPI line shielding or software debounce — neither of which was necessary to demonstrate a working prototype. The complete notification delivery chain — UID scan at the ESP32, attendance record creation, event dispatch to the backend, notification processing and queuing, Firebase Cloud Messaging push delivery, and receipt confirmation on the parent's device — completed in under two seconds for every test run.

This matters because below two seconds, a boarding notification functions as a real-time alert. Above that threshold, it feels like a log entry that arrives after the fact. Every run, clearing in two seconds means parents consistently receive information while the bus is still nearby, not after it has driven away. Fee verification returned 100% accuracy across all 200 runs. Registered, cleared-account cards were admitted every time. Pending balance cards were blocked every time. Unregistered cards were blocked every time. No card received the wrong outcome on any of the 200 tested interactions. Given that fee access control is the mechanism most directly tied to safety and institutional policy enforcement, a clean result across the full test set is the most meaningful single indicator that the system is performing as intended. Routine interface operations such as attendance history retrieval, live map refresh, fee record lookup, and driver route display all returned within 0.5 seconds. The one scenario that occasionally pushed past that mark was the Tracking Interface loading five or more buses simultaneously, which stretched to around 0.8 seconds when the map canvas rendered multiple concurrent GPS streams. This is a client-side rendering issue rather than a backend capacity problem and is scheduled for resolution through pre-loading and incremental rendering before any larger operational rollout.

4.3. Security Evaluation

Data travelling between the bus hardware, the application server, and end-user devices is encrypted under AES-256. This was confirmed by running a packet capture on the network interface during a live test: every intercepted packet was unreadable without the decryption key. JWT-based role permissions govern access to system functions. To stress-test the access boundaries, administrator-level API endpoints were called using tokens belonging to student and parent accounts. Every request returned HTTP 403 without exception. A parent token reached no fee management data. A driver token reached no student personal records. The permission model was held in every scenario tested. Data minimisation governs how personal information is

stored. Student names, RFID identifiers, and attendance records sit in the main database, all encrypted at rest. Parent contact details are in a separate Table 1, which is joined to the attendance data only at notification generation time; the join result is discarded immediately after the message is sent and is never written back to the database. This structure limits the blast radius of a potential breach. It aligns with GDPR data minimisation requirements and Indian personal data protection obligations for systems that hold records relating to students under 18.

4.4. Usability Evaluation

Five participants evaluated the interfaces: two students, one parent, one bus driver, and one college administrator. Nobody was briefed, trained, or shown the system before the session started. Each person received access to their role-specific dashboard and a list of five tasks to complete independently. The students needed to retrieve attendance records from the previous week, locate their assigned bus on the live map, and confirm their current fee balance. The parent needed to find a boarding alert from a specified date in the past and follow the bus in live tracking mode. The driver needed to initiate a trip, submit a mid-route status update, and broadcast a delay notification. The administrator needed to create a new student account, assign the student to a bus, and export an attendance report for the current week. Every participant finished their complete task list without asking for guidance at any point. The average completion time across all tasks and all participants was under 45 seconds. The slowest individual task was the administrator's attendance report export, which took 38 seconds on the first attempt. No one expressed confusion about where any feature was located or what any interactive element would do. The parent observed that the boarding alert message contained all the relevant information, such as which child and which bus, without needing the app to be opened. The driver commented that the single-tap trip start was far simpler than the paper-based sign-off they had been completing at the start of every shift.

4.5. Automatic Door Mechanism

Figure 4 shows the prototype door control assembly. The boarding sequence runs without any driver action. A card is presented; the RC522 captures the UID; the ESP32 cross-references it against the student registry and confirms the fee account is active. Both checks clear, the servo rotates from 0 to 90 degrees, and the door opens. After six seconds, the servo returns to zero, and the door closes. If the proximity sensor detects an object within the threshold during closure, it holds the door open until the path is clear. Every interaction, whether access is granted or denied, generates a timestamped record in the cloud database, triggers an attendance or denial update, and, if a boarding event occurred, fires a parent notification.

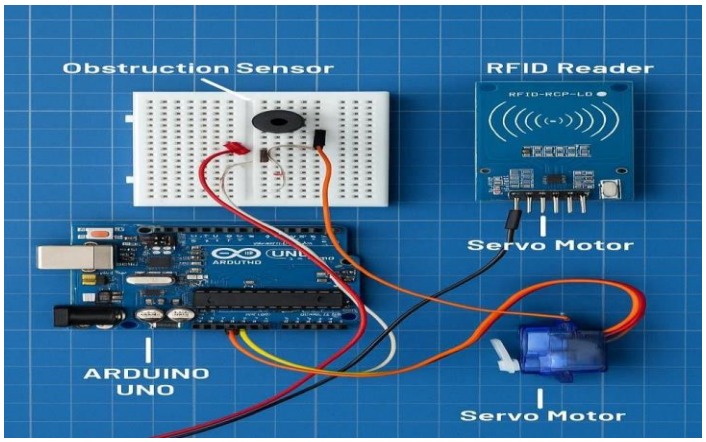

Figure 4: Physical prototype demonstrating RFID-triggered automatic door control

Mechanical performance was reliable throughout testing. The complete open-hold-close cycle averaged 7.2 seconds, keeping the boarding queue moving even at busy stops. The door reached the fully open position within 1.5 seconds of the PWM signal in every run. The proximity sensor held the door correctly in all 20 obstacle-present tests. It closed normally in all 20 obstacle-absent tests, achieving a perfect score across the full set of 40 door-safety trials.

4.6. Scalability

Concurrent load was tested using Apache JMeter configured to generate 500 simultaneous API requests against the three highest-traffic endpoints: GPS location update, attendance event logging, and dashboard data retrieval (Figure 5).

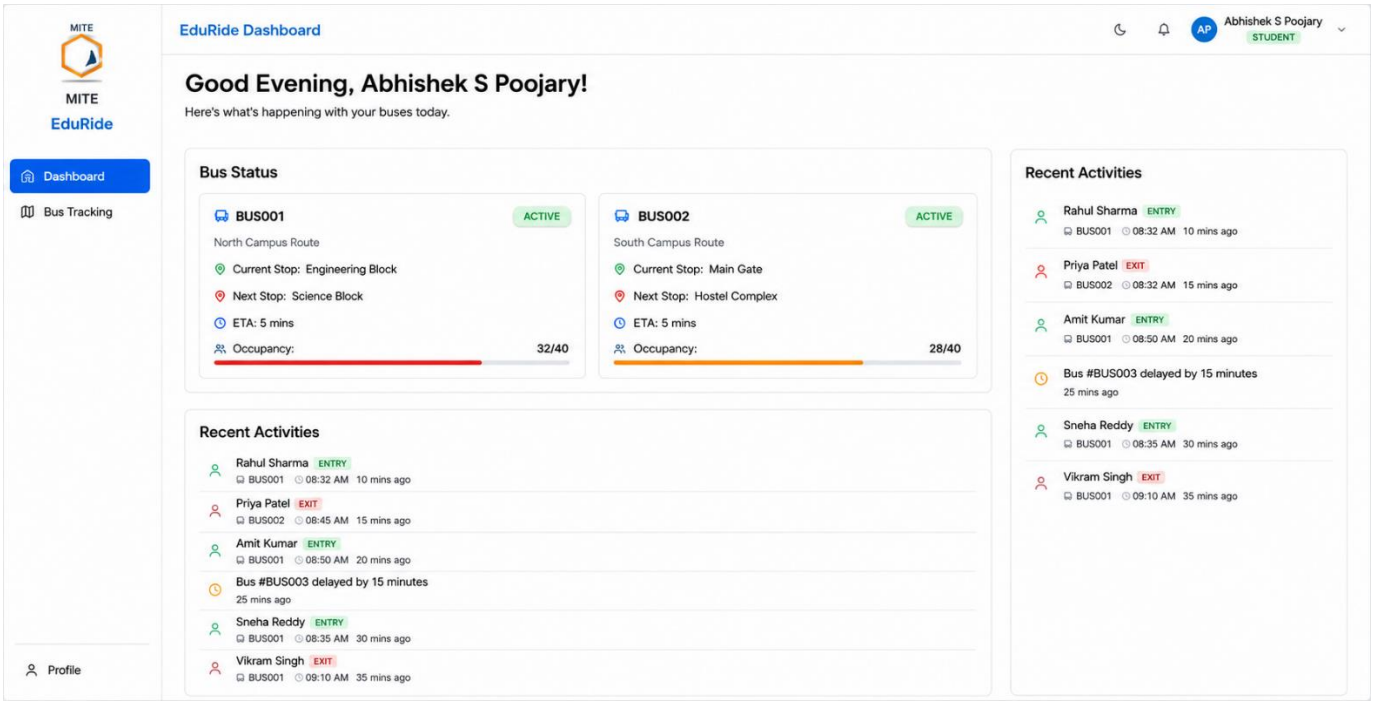

Figure 5: Student dashboard showing live bus location and attendance history

Average response time under this load was 312 milliseconds. The same endpoints under single-user conditions responded in 87 milliseconds (Figure 6).

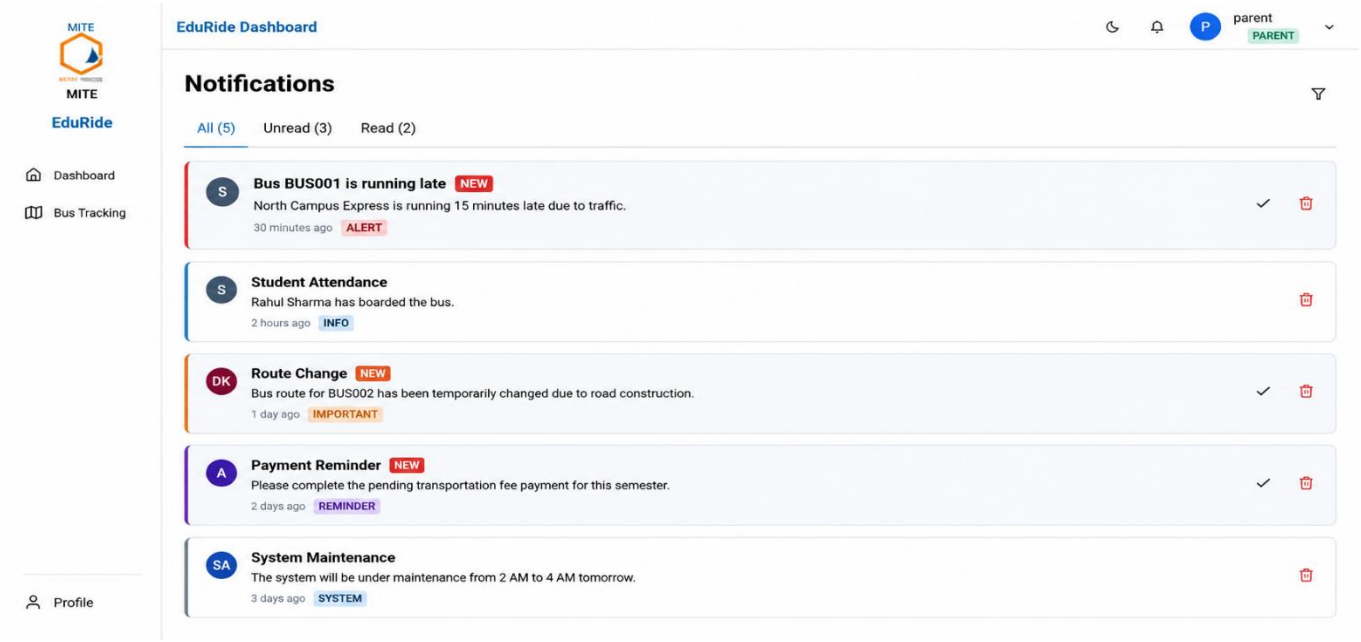

Figure 6: Parent notification panel with real-time boarding alerts

The ratio, 3.6 times slower under 10 times the load, sits well within acceptable bounds for an institutional deployment where response times in the hundreds of milliseconds are indistinguishable from instantaneous for end users. No request returned a time-out or error during the full load test run (Figure 7).

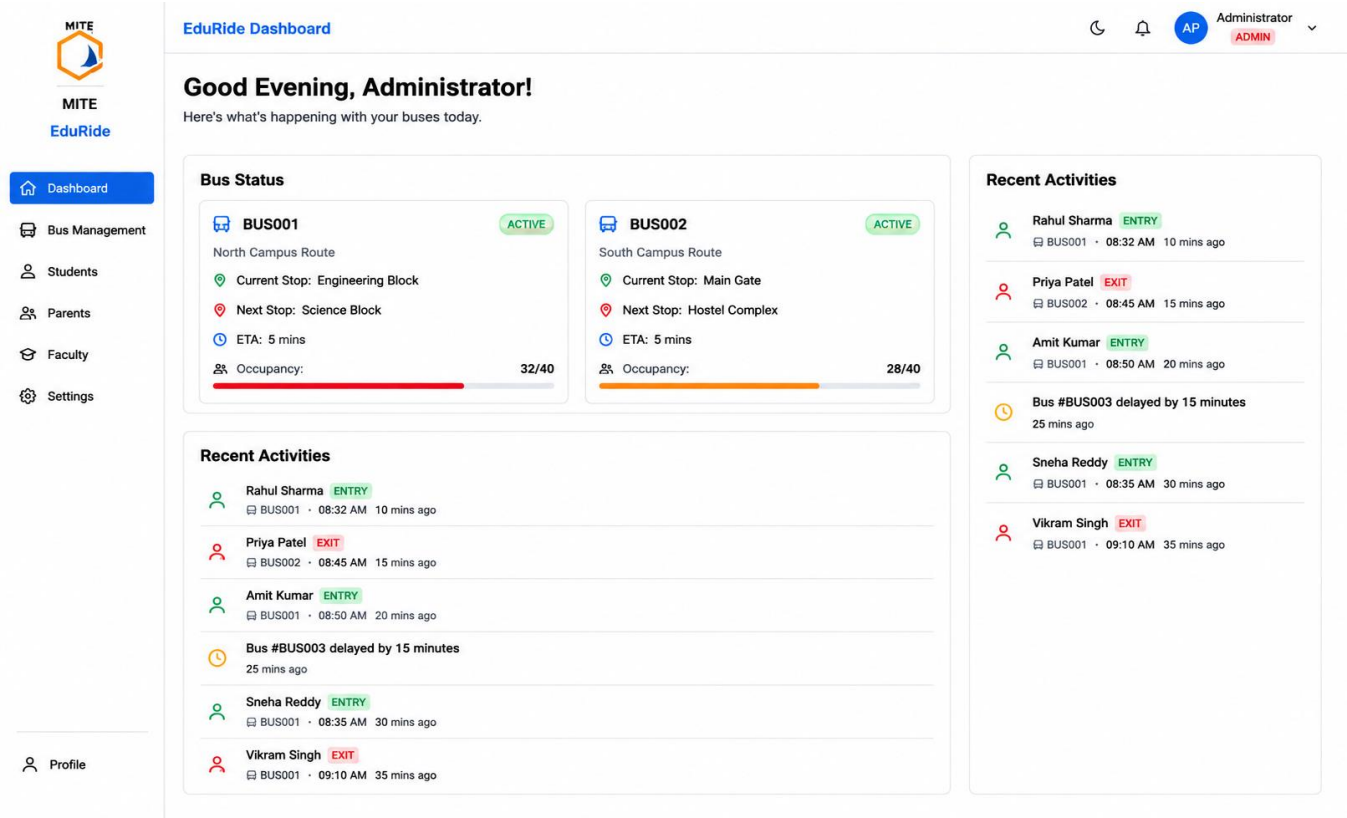

Figure 7: Admin interface for centralised route and fee management

Hardware endurance testing confirmed that the RC522 reader and the SIM808 GSM module both operated without fault across simulated outdoor conditions: temperatures ranging from 18 to 42 degrees Celsius, continuous vibration representative of a moving vehicle, and the electromagnetic interference levels typical of a busy campus (Figure 8).

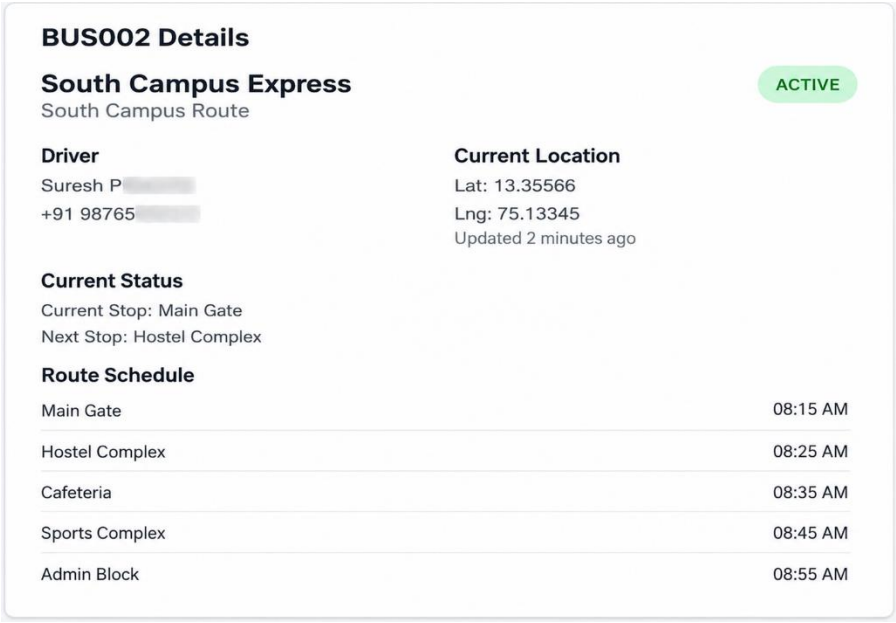

Figure 8: Driver route interface for trip management

The ESP32's Wi-Fi connection remained active across all mobile test routes and recovered within 3 seconds each time it needed to hand off between campus access points (Figure 9).

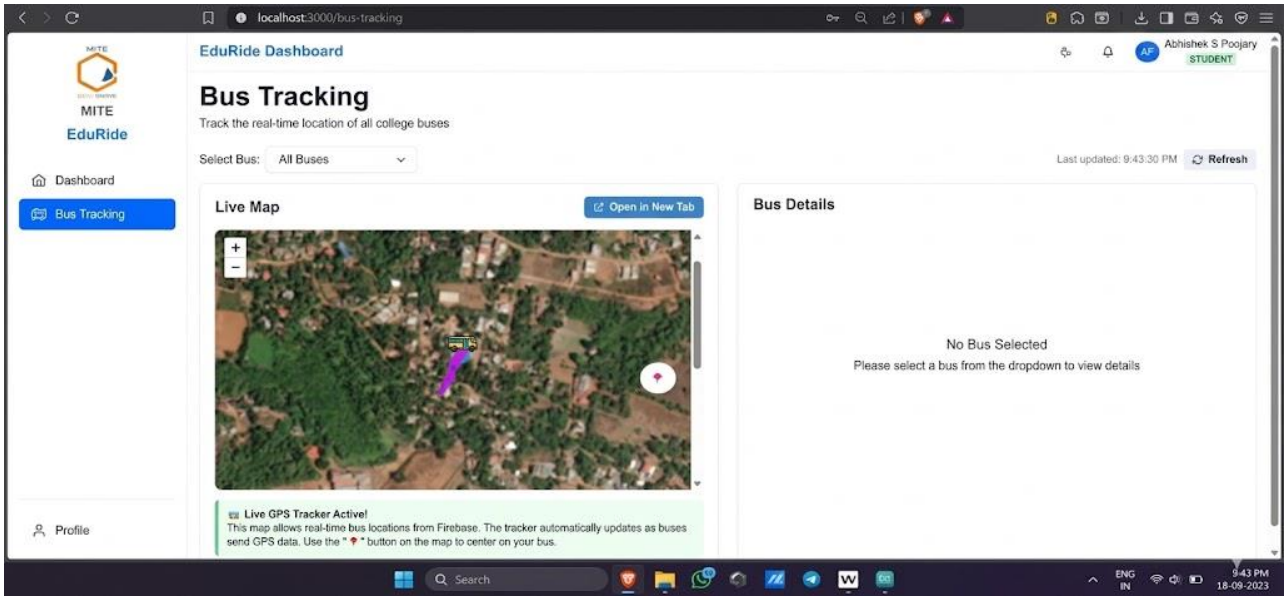

Figure 9: Tracking interface displaying live fleet positions

5. Conclusion and Future Scope

5.1. Conclusion

EduRide's argument is a practical one: college transport does not have to run on paper logs and phone calls. Connecting an ESP32, RFID reader, and GPS unit on each bus to a cloud backend produces a system where attendance records itself, doors open and close without driver involvement, parents receive boarding confirmation in real time, and administrators see the full fleet on a live map. The test results reinforce this argument with numbers: 99% GPS accuracy, 98% RFID scan reliability, boarding notifications consistently under two seconds, and 100% accuracy on fee access decisions across 200 test interactions. None of these Figures was aspirational targets set before testing began — they are the outcomes the system actually produced when run against controlled scenarios on physical hardware. EduRide proves that the technology required to close the gap between what college bus systems currently offer and what students and families actually need is available, affordable, and works. The platform demonstrates that thoughtful integration makes a meaningful difference.

5.2. Future Scope

Multiple directions for further development have been identified. Machine learning applied to historical trip data and live traffic feeds could generate route recommendations that measurably reduce travel time and fuel consumption beyond what static scheduling achieves. IoT sensors mounted on bus engines and drivetrain components could support predictive maintenance workflows, allowing mechanical problems to be addressed before they cause service disruptions. Computer vision-based facial recognition could eventually replace RFID cards as the primary boarding authentication mechanism, removing the need for physical cards while maintaining identity verification integrity. An emergency detection module could trigger automatic SOS notifications to parents, drivers, and emergency services upon detecting collision-level deceleration events or significant unplanned route deviations. Payment gateway integration within the application would allow parents to settle outstanding fee balances directly, eliminating the gap between fee status checks and fee payments. A dedicated analytics module would allow administrators to examine ridership trends, punctuality data, and fleet utilisation patterns and use that information to make better decisions about route design, vehicle allocation, and scheduling.

Acknowledgement: The authors sincerely acknowledge the academic support, research facilities, and collaborative opportunities provided by Mangalore Institute of Technology and Engineering and Arizona State University through its Analytics and Engineering programs, which significantly contributed to the successful completion of this research work.

Data Availability Statement: The data supporting this study are available from the corresponding author and will be shared upon reasonable request, where applicable.

Funding Statement: The authors collectively confirm that no external funding, sponsorship, grant support, or financial assistance was received for conducting this research or preparing the manuscript.

Conflicts of Interest Statement: All authors declare that they have no competing financial, professional, or personal interests that could have influenced the design, execution, analysis, or publication of this research. All sources, datasets, citations, and references have been properly acknowledged and documented.

Ethics and Consent Statement: The authors affirm that the research was conducted in compliance with recognised ethical standards and institutional requirements. Necessary permissions, organisational approvals, and informed consent from participants were obtained before data collection, and appropriate measures were implemented to ensure confidentiality, privacy, and the responsible handling of research data.

References

1. U. Cedric and J. B. Mbanzabugabo, "Smart school bus monitoring and notification system using RFID and GPS," *Research Square*, 2021. [Accessed by 12/05/2025].
2. R. R. Das, A. C. J. Malathi, M. Aarthy, M. L. Moses, and D. R. Sona, "IoT-based school bus and student monitoring system using RFID and GSRM technologies," *Int. J. Intell. Syst. Appl. Eng. (IJISAE)*, vol. 12, no. 3, pp. 164–173, 2024.
3. N. Bode, S. Wadhai, H. Raghatate, T. Naitam, and A. Seloker, "Advance Geo-Fencing Bus Tracking and Attendance System with Real-Time Location Alert," *Int. J. Adv. Comput. Eng. Commun. Technol.*, vol. 14, no. 1, pp. 210–215, 2025.
4. L. Lakshmiprabha, S. Jadhav, S. Rawani, and S. Kumbhar, "IoT based child security system for school bus," *Int. J. Eng. Res. Technol. (IJERT)*, vol. 14, no. 1, pp. 1–6, 2025.
5. S. J. S. Daisy, B. Gayathri, and S. Induja, "IoT enabled school bus tracking and student monitoring system," *Int. J. Adv. Res. Comput. Commun. Eng. (IJARCCE)*, vol. 14, no. 5, pp. 65–71, 2025.
6. M. Vinodhini and S. Rajkumar, "Intelligent smart transport system using Internet of Vehicular Things — A review," in Futuristic Communication and Network Technologies, *Springer*, Singapore, 2023.
7. A. Singh, "Transportation management using IoT," in Deep Learning Technologies for the Sustainable Development Goals, Advanced Technologies and Societal Change, *Springer*, Singapore, 2023.
8. P. S. Saarika, K. Sandhya, and T. Sudha, "Smart transportation system using IoT," in *Proc. International Conference on Smart Technologies for Smart Nation (SmartTechCon)*, Bengaluru, India, 2017.
9. N. R. Gawade, S. R. Buchade, K. N. Aadeni, S. P. Jirage, and S. A. Naik, "Review on automatic IoT based smart public transport bus and station system," *Int. J. Res. Appl. Sci. Eng. Technol. (IJRASET)*, vol. 12, no. 11, pp. 2000–2003, 2024.
10. T. Domínguez-Bolaño, V. Barral, C. J. Escudero, and J. A. García-Naya, "An IoT system for a smart campus: Challenges and solutions illustrated over several real-world use cases," *Internet of Things*, vol. 25, no. 4, p. 101099, 2024.
11. K. Ishaq and S. Bibi, "IoT-based smart attendance system using RFID: A systematic literature review," *arxiv.org arXiv:2308.02591*, 2023. [Accessed by 17/05/2025].
12. O. Kandil, R. Rosillo, R. A. El Aziz, and D. De La Fuente, "Investigating the impact of the Internet of Things on higher education: A systematic literature review," *J. Appl. Res. Higher Educ.*, vol. 17, no. 1, pp. 254–273, 2025.
13. I. K. Singgih, A. R. Prabowo, S. Soegiharto, M. L. Singgih, and F. P. Dharma, "Smart campus applications: A literature review on operations research and big data," *Int. J. Technol.*, vol. 16, no. 3, pp. 796–824, 2025.
14. X. Ye, H. Peng, and H. Liao, "Research and practice of IoT technology based on smart campus," *Frontiers in Computing and Intelligent Systems*, vol. 10, no. 1, pp. 59 –62, 2024.
15. P. Gupta and V. Patel, "Smart campus navigation: An IoT-based approach for intelligent administration," *Int. J. Sci. Res. Dev. (IJSART)*, vol. 11, no. 3, pp. 460–463, 2025.