

Smart School Bus: Integrated School Transportation Safety & Management System

Prof. M. K. Nichat¹, Chaitanya Ladole², Samiksha Chavan³, Gorakh Tapdiya⁴, Rajshri Dhumane⁵, Krutika Warthi⁶

Department of Computer Science and Engineering, P. R. Pote Patil College of Engineering and Management, Amravati, Maharashtra, India

Abstract --School transportation safety is a persistent concern for parents, school administrators, and transport staff, since conventional arrangements depend on manual attendance registers, verbal communication between drivers and school offices, and no continuous visibility of a bus's location once it leaves the school premises. This paper presents Smart School Bus, a role-based, cross-platform application built using Flutter and Dart on the client side and Firebase Authentication together with the Firebase Realtime Database on the backend, which is designed to centralize school transportation operations for four distinct user roles: Parent, Driver, Conductor, and Admin. The system is organized around an authentication and role-resolution pipeline that determines the screens presented to each signed-in user, a live bus-telemetry model that streams location and status data from the Firebase Realtime Database to tracking screens, a student attendance module that lets authorized staff record individual or bulk boarding status, and a role-aware notification system that keeps parents and staff informed of delays, arrivals, and alerts. Authorization is enforced not only in the user interface but also at the database layer through Firebase security rules, so that a modified client cannot bypass role restrictions. This paper describes the system's architecture, functional modules, real-time data flow, and security model exactly as implemented, and it explicitly separates functionality that has been implemented from functionality such as physical vehicle-telemetry hardware and production-grade monitoring that remains proposed future work. No quantitative performance results are reported, since formal load, accuracy, or field testing had not been completed at the time of writing; instead, the areas the system is designed to be validated against are identified. The paper situates this work relative to prior IoT-based school bus tracking literature and identifies a research gap around role-based, real-time, database-driven mobile applications for school transportation.

Keywords - School bus tracking, student safety, Flutter, Dart, Firebase Realtime Database, Firebase Authentication, role-based access control, IoT-oriented telemetry.

I. INTRODUCTION

Every school day, large numbers of students depend on school bus services to travel safely between home and school.

Parents want confirmation that their child has boarded and disembarked at the correct stop; drivers and conductors need a simple way to record this information without disrupting the trip; and school administrators need visibility across the entire fleet of buses, routes, and students. Conventional school transportation setups largely rely on manual roll calls, phone calls, and paper registers.

Smart School Bus is proposed and implemented as a response to this gap. It is a single, cross-platform Flutter application, distinguished from a generic tracking utility by its explicit separation of four user roles - Parent, Driver, Conductor, and Admin, each of which is presented with only the screens and actions relevant to it. The application authenticates users through Firebase Authentication, resolves each user's role and (where applicable) their assigned bus from the Firebase Realtime Database, and then streams live bus location, student boarding status, and notifications to the relevant screens as the underlying data changes.

This paper documents the system as designed and implemented, drawing on the project's own technical documentation. Where a capability described in this paper for example, physical vehicle-telemetry hardware integration or formal security testing has not yet been completed, this is stated explicitly rather than presented as a finished result, and is identified separately as future work.

II. MOTIVATION

The motivation for this work follows directly from the practical shortcomings of manual school transportation management: parents have no way to confirm boarding status other than waiting for their child to arrive; drivers and conductors have no digital record of who is on the bus at a given moment; and school administrators have no consolidated view of fleet status, routes, or student-to-parent relationships. A system that gives each of these stakeholders a focused, role-appropriate, real-time interface while keeping sensitive operations restricted to the correct roles can reduce this uncertainty and give school transportation a level of digital accountability comparable to other safety-critical services.



International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 10, October 2026)

III. PROBLEM STATEMENT

There is no single, role-aware, real-time platform that lets parents, drivers, conductors, and administrators interact with the same underlying school-transportation data set, each through an interface scoped to their own responsibilities. Existing manual processes provide no live bus location, no digitally recorded boarding or alighting status, and no structured, role-restricted administrative control over buses, routes, students, and user accounts. Where digital tracking does exist in isolated forms for example, driver-side location sharing alone [1], [2], [5], [7] it is rarely combined with authenticated, role-based access control enforced at the data layer, which is necessary if the system is to be trusted with student safety information.

IV. OBJECTIVES

- To design and implement a centralized, cross-platform application that improves visibility of school buses through live location and route telemetry.
- To provide a clear digital record of student boarding status in place of manual, verbal communication.
- To give administrators centralized fleet, route, student, and user-management capabilities.
- To deliver real-time notifications and synchronize bus, student, and account information across devices using a shared real-time database.
- To build a responsive interface that functions correctly on both mobile devices and larger desktop or web screens.

V. EXISTING SYSTEM AND LIMITATIONS

Conventional school transportation relies on manual attendance registers maintained by conductors or drivers, verbal or phone-based communication between the bus staff and the school office, and no real-time visibility of the bus for parents beyond informal updates. This creates several limitations: attendance records are not centrally stored or auditable; there is no live confirmation of a bus's location or estimated arrival time; parents have no structured notification channel for delays, arrivals, or emergencies; and school administrators lack a single dashboard for managing buses, routes, students, and staff accounts.

These limitations motivate the shift toward a digital, rolebased, real-time system of the kind proposed in this paper.

VI. LITERATURE REVIEW

Recent research has increasingly focused on software-based school and educational transport systems that provide real-time tracking, mobile access, cloud synchronization, and student-safety features. Patil et al. presented an IoT based bus tracking system using GPS devices and an Android application, with real-time location and delay notifications for users [1]. Nunse et al. developed an Android application for school bus tracking and student monitoring, demonstrating the use of a mobile interface for transport visibility and monitoring [2].

More closely related to the present work, the EduBusGuru system combines GPS tracking from a driver's smartphone with the Firebase Realtime Database and Flutter to support real-time educational bus management, including administration of buses, routes, stops, and student assignments [3]. MyChildBus further demonstrates a parent-oriented, real-time school bus monitoring application using Firebase and Google Maps, with modules for live tracking, route navigation, communication, incident handling, and management of students, drivers, and buses [4].

Recent work also explores broader safety and transportation-management capabilities. Patwardhan et al. described a GPS-based school bus tracking system using IoT, cloud computing, and mobile technology for real-time location, estimated arrival information, route management, and emergency alerts [5]. These studies demonstrate the value of real-time digital transportation platforms, but they generally emphasize tracking, parent monitoring, or a specific management function rather than a single cross-platform application that combines four role-specific interfaces with database-enforced authorization and synchronized attendance information. The present Smart School Bus system addresses this software-integration gap through Flutter, Firebase Authentication, the Firebase Realtime Database, role resolution, role-specific navigation, student boarding status, notifications, and database-level security rules.

TABLE I. LITERATURE REVIEW SUMMARY

Ref.	Author / Year	Method / Technology	Key Findings	Limitation / Research Gap
[1]	Patil et al. (2024)	IoT + GPS + Android app	Real-time bus location tracking and delay notifications for educational transport.	Emphasizes tracking and notifications rather than a multi-role application architecture; does not describe database-level authorization.
[2]	Nunse et al. (2024)	Android mobile application	Supports school bus tracking and student monitoring through a mobile interface.	Does not describe a unified four-role architecture with database-level authorization.
[3]	EduBusGuru (2024)	GPS + Firebase Realtime Database + Flutter	Real-time bus tracking with administration of buses, routes, stops, and student assignments.	Presented in a college/campus transport context; does not present the same four-role attendance workflow.
[4]	Zulkaria & Darman (2025)	Mobile app + Firebase + Google Maps	Provides live school bus tracking, route navigation, communication, incident handling, and management modules.	User roles are more limited; database-enforced role permissions are not the main contribution.
[5]	Patwardhan et al. (2025)	GPS + IoT + cloud + mobile app	Real-time location, ETA, route information, and emergency alerts for school transportation.	Focuses mainly on tracking and alerts rather than integrated role-based application security and attendance.
[6]	Sharif et al. (2018)	GPS tracker + IoT counter + Firebase cloud DB + hybrid mobile app	Real-time campus bus location and estimated arrival streamed through a shared Firebase database.	Single-role (student-facing) campus application; no authentication, role separation, or attendance workflow.
[7]	Hasan & Hossen (2019)	Android + GPS + Google Maps (client-server, noncloud-DB)	Real-time Android bus location tracking for students without cloud-hosted real-time sync.	No cloud real-time database, no role-based access, no attendance or notification modules.
[8]	Kimm & Flynn (2022)	Android Fused Location Provider + Firebase Realtime Database	Demonstrates FirebaseRTDB-based real-time location streaming methodology for a mobile client.	General pedestrian-tracking use case; not applied to transportation safety, roles, or attendance.

The recent literature demonstrates growing use of GPS, cloud databases, mobile applications, and real-time notifications for educational transportation. However, the reviewed systems commonly focus on one or a smaller set of stakeholder views, such as parents, students, or transport administrators, and do not consistently describe a single cross-platform application with four distinct roles and database-level authorization. Smart School Bus responds to this gap by integrating Parent, Driver, Conductor, and Admin workflows in one Flutter application, using Firebase Authentication for identity, the Firebase Realtime Database for synchronized data, and security rules to restrict access at the data layer.

GPS-based school bus tracking has been studied extensively at the level of individual location-sensing and notification techniques [1],[2],[5],[7], and the Firebase Realtime Database has been used separately as a synchronization layer for vehicle- and pedestrian-tracking mobile applications [6],[8],[9],[10]. However, the reviewed systems generally address a single stakeholder view typically the parent or the vehicle operator rather than a shared platform with role-differentiated access for parents, drivers, conductors, and administrators, and few explicitly describe authorization enforced at the database layer rather than only in the client interface.

Smart School Bus responds to this gap by combining a Firebase-based real-time data layer with an authentication and role resolution pipeline that scopes both the user interface and the underlying database permissions to four distinct roles within a single cross-platform Flutter codebase.

VII. PROPOSED SYSTEM

Smart School Bus is proposed as a role-based Flutter application that connects school transport operations, live bus telemetry, student boarding status, route visibility, notifications, and administration through Firebase. Parents can see their child's boarding status and bus location; drivers can access route navigation and their profile; conductors can manage student check-in and check-out; and administrators can supervise buses, routes, tracking, and user and student data all from the same underlying Firebase Authentication and Realtime Database backend.

A. Proposed System Architecture

The application follows a layered architecture. Firebase is initialized before the Flutter application starts. An Auth Gate widget observes Firebase authentication state and decides whether to present the Login Screen or the authenticated application. Once a user is authenticated, a Role Resolution Shell loads the user's role and assigned bus ID, using locally cached values first for a fast start and then refreshing them from Firebase. A Main Navigation Shell then builds the navigation tabs dynamically based on the resolved role, so that a Parent, Driver, Conductor, or Admin sees only the screens relevant to their responsibilities. Firebase Realtime Database streams keep bus location, student status, and notifications synchronized across all connected clients. Session state - the resolved role, assigned bus, and last-selected navigation tab is persisted locally through Shared Preferences via a Session Service, so that the application can resume quickly on relaunch.

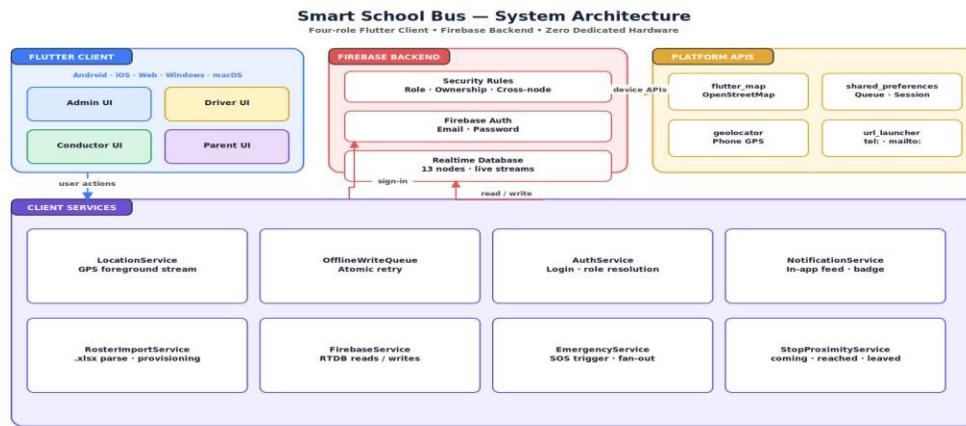


Fig. 1. System Architecture of Smart School Bus

B. System Workflow

- The application starts and initializes Firebase.
- AuthGate listens for the Firebase authentication state.
- Unauthenticated users are shown the Login screen; authenticated users proceed to role resolution.
- RoleResolutionShell loads cached role and bus data for a fast start, then refreshes it from Firebase.
- MainNavigationShell builds the correct set of screens for the resolved role: Parent, Driver, Conductor, or Admin.
- Screens subscribe to real-time Firebase streams and display current bus location, student status, and notification data.
- Authorized staff (Driver, Conductor, or Admin, as permitted) update boarding or operational data; Firebase persists the change and all connected clients receive the update through their active listeners.
- The user can sign out, which clears the authenticated session and any in-memory notifications.

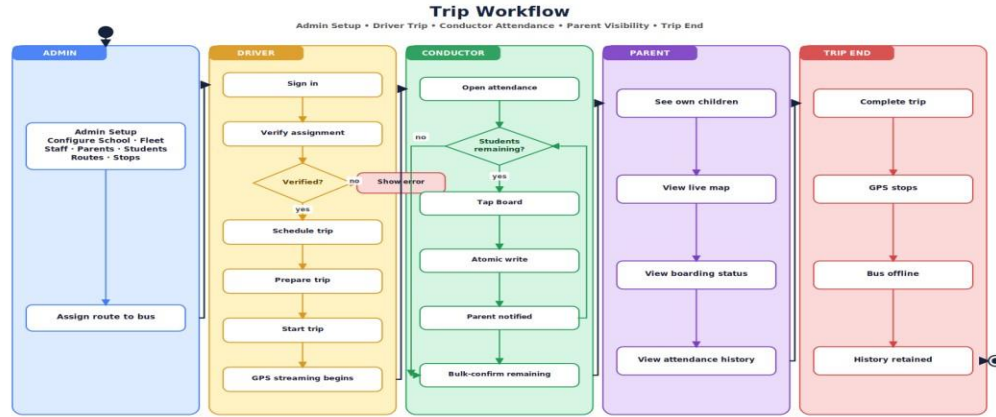


Fig. 2. System Workflow of Smart School Bus

C. System Modules

The system's functional modules, as implemented, are organized by role:

- *Authentication & User Access* - centralized in an AuthService; email/password sign-in, sign-out, role and bus-assignment lookup, and password reset via a dedicated ForgotPasswordScreen.
- *Parent module* - Child Boarding Status, Live Bus Tracking, Profile, and role-aware Notifications screens.
- *Driver module* - Bus Route Navigation and Driver Profile screens.
- *Conductor module* - Student Check-in/Check-out, Bus Route Map, and Profile screens.
- *Admin module* - FleetManagementScreen, RoutePlanningScreen, ManageStudentsScreen, PeopleDirectoryScreen, and CreateUserScreen, covering fleet status, live tracking, route planning, student records, the staff/parent directory, and account provisioning.
- *Tracking & Telemetry module* - a BusLocation model and FirebaseService that stream live vehicle data.
- *Student Attendance module* - student roster records with boarding status maintained through Firebase.
- *Notification module* - a NotificationService that generates, stores, and displays role-aware notifications.

TABLE II. ROLE-WISE SCREENS AND RESPONSIBILITIES

Role	Main Screens	Primary Responsibilities
Parent	Child Boarding Status; Live Bus Tracking; Profile; Notifications	View child's transport status, follow the assigned bus in real time, manage profile, and receive role-aware notifications.
Driver	Bus Route Navigation; Driver Profile	View assigned bus route and tracking data, and manage profile information.
Conductor	Student Check-in/Check-out; Bus Route Map; Profile	Manage boarding status and monitor the active bus route.
Admin	FleetManagementScreen; RoutePlanningScreen; ManageStudentsScreen; PeopleDirectoryScreen; CreateUserScreen	Supervise fleet and route operations and manage bus, route, student, and user account data.

D. Authentication and Role Management

Authentication is centralized in AuthService. Users sign in with email and password through Firebase Authentication. On successful sign-in, the service reads the authenticated user's UID, retrieves the corresponding role from `/users/{uid}/role`, and, where applicable, retrieves the bus assignment from `/users/{uid}/busId`. Role resolution supports four roles: Parent, Driver, Conductor, and Admin. Administrators can provision new users by supplying a name, email, role, optional bus ID, and phone number through the `CreateUserScreen`, which uses the same centralized authentication service. User sessions and the currently selected navigation tab are persisted locally so that the correct role-specific view is restored on the next launch.

E. Real-Time Bus Tracking

The `BusLocation` model captures live telemetry for each bus, including latitude, longitude, speed, run status, `lastupdate` timestamp, current stop, total stops, stop label, estimated time of arrival, and bus number. `FirebaseService` listens at the `/buses/{busId}` path in the Realtime Database and converts incoming records into strongly typed `BusLocation` objects for the tracking screens. The model also performs staleness detection: if the most recent update is older than a configured two-minute threshold, the telemetry is treated as stale, and the interface can reflect this rather than showing an outdated position as if it were current.

H. Database Design

F. Student Boarding/Attendance Management

Student records include an ID, name, grade, seat, stop, parent UID, boarding status, and boarding timestamp. Supported status values are *pending*, *boarded*, and *alert*. Authorized bus staff (Driver, Conductor, and Admin) can update an individual student's status or mark all students on a roster with a status in bulk, which is useful for start-of-route or end-of-route resets. Student rosters are synchronized through the Firebase Realtime Database at `/studentRosters/{busId}`, so that boarding-status changes made by a conductor are immediately visible in the corresponding parent's app.

G. Notification System

Notifications are stored per user under `/notifications/{uid}`. Supported, validated notification kinds are *delay*, *arrival*, *boarding*, *alert*, *info*, *departure*, and *emergency*. Each notification record captures a title, message, timestamp, and read state, with optional bus or student context attached where relevant. `NotificationService` converts these records into typed `AppNotification` objects, sorts them newest-first, and exposes an unread count; it also includes short-window duplicate-prevention logic so that the same event is not reported to the user multiple times in quick succession. In-memory notifications are cleared on sign-out, and the listener is re-established on the next authenticated session.

TABLE III. FIREBASE REALTIME DATABASE STRUCTURE

Path	Purpose	Typical Data
<code>/users/{uid}</code>	Identity and role profile	name, email, role, busId, phone
<code>/buses/{busId}</code>	Live bus telemetry	lat, lng, speedKmph, status, ETA, stop data
<code>/studentRosters/{busId}</code>	Student manifest	student details, status, boardedAt, parentUid
<code>/parentChildIndex/{parentUid}</code>	Parent-child linkage	bus/student relationships
<code>/notifications/{uid}</code>	User notification inbox	kind, title, message, timestamp, read state

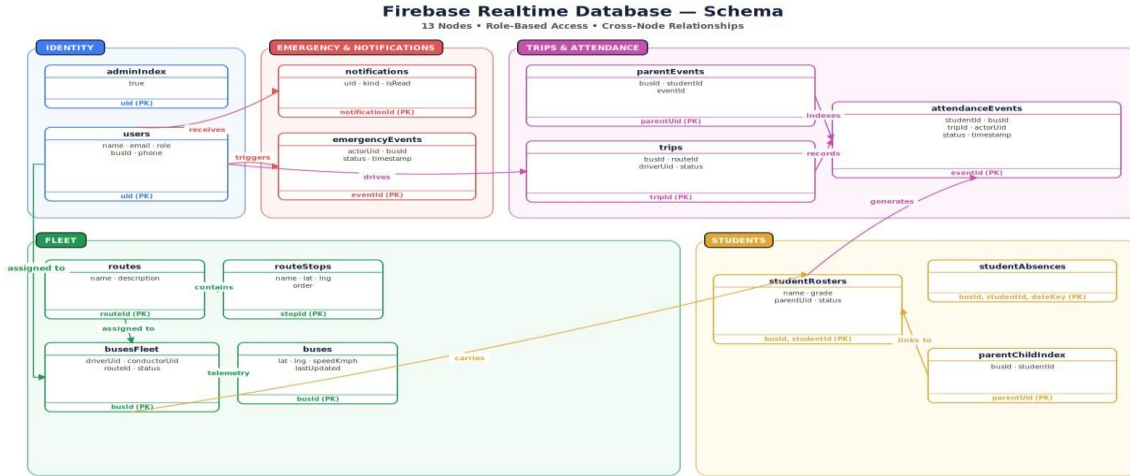


Fig. 3. Firestore Realtime Database Structure

I. Security and Access Control

The application enforces authorization at two levels. At the interface level, MainNavigationShell only builds the screens appropriate to the resolved role, so a Parent is never shown administrative screens, and so on. At the data level, the supplied database.rules.json defines authenticated, role-aware access rules directly on the Firestore Realtime Database: the root of the database is closed by default; user records can be read or updated only by the account owner or an Admin; bus records can be read by an Admin, the assigned

Driver or Conductor, or a parent linked to that bus, while writes to bus records are restricted to an Admin or the Driver assigned to that specific bus; and writes to student status fields are restricted to an Admin or the assigned Driver/Conductor.

This two-layer design reflects the project's stated security principle: hiding a screen in the user interface alone is not a sufficient security boundary. A modified or custom client could otherwise still issue unauthorized reads or writes directly against the database, so the same restrictions are enforced again at the data layer.

J. Technology Used

TABLE IV. TECHNOLOGY STACK

Layer	Technology / Implementation	Purpose
UI	Flutter + Dart	Cross-platform interface and responsive layouts across Android, Web/Chrome, and Windows targets.
Authentication	Firebase Authentication	Email/password sign-in, sign-out, and password reset.
Database	Firebase Realtime Database	Live bus telemetry, student rosters, user profiles, and notifications, synchronized in real time.
Session	SharedPreferences / SessionService	Persists role, bus assignment, and the active navigation tab across app launches.
Design	Custom AppTheme + Manrope/Inter typefaces	Consistent, safety-oriented visual system (Safety Blue and Safety Orange).
Utilities	url_launcher	Opening external links and actions where required.



Flutter and Dart provide a single codebase capable of targeting Android, Web/Chrome, and Windows, which is well suited to a system that must reach parents on mobile devices while giving administrators the option of a larger desktop or browser-based view. Firebase Authentication removes the need to build and secure a custom credential store, while the Firebase Realtime Database's onValue listener model gives the application an event-driven data layer: rather than polling for updates, screens are notified automatically whenever the underlying bus, student, or notification data changes. This combination keeps the client logic focused on presentation and role-based navigation while the backend handles authentication, storage, and real-time synchronization.

K. System Implementation

Application initialization begins with Firebase being configured before the Flutter widget tree is built, ensuring authentication and database services are available from the first frame. AuthGate then observes Firebase's authentication-state stream and renders either the LoginScreen or the authenticated application shell accordingly. On successful authentication, role resolution is handled by RoleResolutionShell, which first applies a cached role and bus ID for a fast, responsive start, and then refreshes both values from the /users/{uid} node in the Realtime

Database, so that any role change made by an Admin takes effect the next time the user's session is refreshed. Role based navigation is then constructed by MainNavigationShell, which reads the resolved role and dynamically assembles the corresponding tab set - for example, FleetManagementScreen, RoutePlanningScreen, ManageStudentsScreen, PeopleDirectoryScreen, and CreateUserScreen for an Admin, versus Child Boarding Status, Live Bus Tracking, Profile, and Notifications for a Parent.

Firebase synchronization is handled by two dedicated services: FirebaseService listens at /buses/{busId} and /studentRosters/{busId} and converts incoming JSON payloads into strongly typed BusLocation and Student model objects, while NotificationService performs the equivalent conversion for /notifications/{uid} into AppNotification objects.

Because these listeners are attached using Firebase's onValue mechanism, the application is event-driven rather than dependent on periodic manual refreshes: a status update written by a conductor, for instance, is pushed to every subscribed client - including the linked parent's device without any client having to poll the server.

Bus tracking is implemented through the BusLocation model's staleness check, which compares the current time against the telemetry's last-update timestamp and flags data older than two minutes as stale, so that tracking screens can distinguish a genuinely live bus from one whose telemetry feed has stopped updating. Student status management is implemented through authorized writes to /studentRosters/{busId}, restricted at the database-rules layer to Admin and the assigned Driver/Conductor, supporting both individual updates and bulk status changes across an entire roster. Notification handling combines the per-user /notifications/{uid} node, the seven supported notification kinds, and short-window duplicate suppression, so that repeated writes for the same event do not generate redundant alerts. Admin management is implemented through the FleetManagementScreen, RoutePlanningScreen, ManageStudentsScreen, PeopleDirectoryScreen, and CreateUserScreen, which together give an administrator centralized control over buses, routes, students, the user directory, and account provisioning, all backed by the same Firebase Authentication and Realtime Database layer used by the other roles.

VIII. RESULTS AND DISCUSSION

At the current stage of the project, the contribution reported in this paper is the system design, architecture, and functional implementation described in Sections VIII and X. No formal load testing, field trial, or accuracy measurement had been completed at the time of writing. This paper therefore does not report quantitative accuracy, response-time, or performance figures, since doing so without completed testing would misrepresent the project's current status. What can be reported truthfully is the set of functional capabilities that have been implemented: role-based authentication and navigation, live bus-location streaming with staleness detection, individual and bulk student status updates, a role-aware notification system, and Firebase-rules-based access control and the areas that the project documentation identifies as requiring structured validation before production deployment.

TABLE V. IDENTIFIED TESTING AND VALIDATION AREAS (NOT YET EXECUTED AS FORMAL TESTS)

Area	What Should Be Verified
Authentication	Valid/invalid login, logout, password reset, and session restoration.
Role access	Each role receives only its intended navigation tabs and functions.
Live tracking	Valid telemetry renders correctly; missing or stale telemetry shows an appropriate state.
Attendance	Individual and bulk status updates persist and appear consistently across clients.
Notifications	Creation, read/unread state, ordering, duplicate prevention, and sign-out behaviour.
Admin	User creation, student management, parent linking, and fleet/route screens.
Responsive UI	No overflow on mobile/desktop; cards, navigation, and text reflow correctly.
Security	Firebase rules reject unauthorized reads/writes even if a client is modified.

These areas represent expected outcomes to be verified once formal testing is undertaken, not results already obtained. Framing them this way is consistent with the project documentation, which notes that production deployment should include stronger operational monitoring, backup/recovery procedures, privacy controls, and formal security testing before the system is relied upon in an operational school setting.

IX. ADVANTAGES

- Consolidates transportation operations for four stakeholder roles into a single application and backend, reducing the need for separate manual processes for each role.
- Provides parents with real-time visibility of their child's boarding status and the bus's live location instead of relying on informal updates.
- Enforces authorization at both the interface and database layers, so that a modified client cannot bypass role restrictions.
- Runs from a single Flutter codebase across Android, Web/Chrome, and Windows, reducing duplicated development effort across platforms.

The hardware and analytics related extensions considered during planning - such as dedicated GPS/cellular telemetry hardware and geofencing are not part of the current software implementation described in Sections VIII through X and are treated separately as future work in Section XIV.

X. CHALLENGES AND LIMITATIONS

- The current implementation depends entirely on software-side telemetry and status data written into the Firebase Realtime Database; it does not yet integrate any physical vehicle-mounted hardware, so live bus data must be supplied by a data source such as a driver-side device or a manually configured feed rather than by dedicated onboard sensors.
- The attendance interface contains scanner-oriented UI components, but the specific hardware or identification method for a final deployment has not yet been finalized and should be validated once that method is chosen.
- No formal load testing, field trial, or security penetration testing had been completed at the time of writing.
- Production deployment would require stronger operational monitoring, backup and recovery procedures, and privacy controls beyond what has currently been implemented.

XI. FUTURE SCOPE

- Improve security and maintain detailed audit logs for administrative activities and system changes.
- Add automated emergency alerts to quickly notify parents and authorities during critical events.

- Enhance route, delay, attendance, and performance analytics for better monitoring and decision-making.
- Implement automated testing and stronger role-based security to improve system reliability and access control.

XII. CONCLUSION

This paper has presented Smart School Bus, a role-based, cross-platform school transportation application built with Flutter, Dart, Firebase Authentication, and the Firebase Realtime Database. By resolving each authenticated user into one of four roles - Parent, Driver, Conductor, or Admin and scoping both navigation and the underlying database permissions to those roles, the system centralizes live bus tracking, student boarding-status management, notifications, and administrative control within a single, event-driven backend.

Recent educational transportation studies demonstrate the usefulness of real-time tracking, mobile applications, cloud synchronization, and digital transport management; however, the reviewed work does not consistently combine these capabilities with four-role navigation and database-enforced authorization in one cross-platform application. A separate line of Firebase- and GPS-based real-time tracking research establishes the underlying synchronization architecture event-driven cloud databases feeding live location and status to mobile clients that this system applies to the school-transportation setting. GPS-based school bus safety systems and Firebase-based real-time tracking applications have each been studied independently, but a role-based platform combining authenticated, database-enforced access control across four stakeholder roles within one real-time system has not been commonly reported. The contribution of the present work is therefore its integrated software architecture, role-aware workflow, real-time synchronization model, and security design, all realized in software on top of Firebase. Formal performance and field validation, along with future physical vehicle-telemetry hardware integration and the AI-assisted safety extensions noted in Section XIV, remain identified future work rather than completed results.

REFERENCES

- [1] P. Patil, D. Patil, V. Bhandare, D. Malpathak, and J. Thakur, "IoT based Bus Tracking System," *International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences*, vol. 12, no. 2, 2024.
- [2] N. Nunse, T. Jadhav, P. Nirgude, S. Pangare, and A. D. Talole, "An Android Application for School Bus Tracking and Student Monitoring System," *International Journal of Creative Research Thoughts*, vol. 12, no. 3, pp. b26-b32, 2024.
- [3] "EduBusGuru: Enhancing College Bus Tracking and Management with Real-Time Mobile App," in *Proc. 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024, doi: 10.1109/ICCCNT61001.2024.10724760.
- [4] F. N. Zulkaria and R. Darman, "MyChildBus: Real-Time School Bus Monitoring and Rides Tracking Mobile Application for Parents," *Applied Information Technology and Computer Science*, vol. 6, no. 2, pp. 110-129, 2025, doi: 10.30880/aitcs.2025.06.02.007.
- [5] S. N. Patwardhan, S. Patil, A. Koli, and O. Atole, "School Bus Tracking System," *IJSREM*, 2025.
- [6] S. A. Sharif, M. S. A. Suhaimi, N. N. Jamal, I. K. Riadz, I. F. Amran, and D. N. Abang Jawawi, "Real-Time Campus University Bus Tracking Mobile Application," in *Proc. 2018 Seventh ICT International Student Project Conference (ICT-ISPC)*, 2018, pp. 1-6, doi: 10.1109/ICT-ISPC.2018.8523915.
- [7] M. N. Hasan and Md. S. Hossen, "Development of An Android Based Real Time Bus Tracking System," in *Proc. 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, Dhaka, Bangladesh, 2019, pp. 1-5, doi: 10.1109/ICASERT.2019.8934621.
- [8] H. Kimm and D. Flynn, "Android Application for Tracking Pedestrian Movements in Realtime with Firebase," in *Proc. 2022 IEEE Cloud Summit*, Fairfax, VA, USA, 2022, pp. 91-96.
- [9] W. J. Li, C. Yen, Y. S. Lin, S. C. Tung, and S. M. Huang, "Just IoT Internet of Things based on the Firebase Real-time Database," in *Proc. 2018 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE)*, Hsinchu, Taiwan, 2018, pp. 43-47.
- [10] R. Mallik, D. Sing, and R. Bandyopadhyay, "GPS Tracking App for Police to Track Ambulances Carrying COVID-19 Patients for Ensuring Safe Distancing," *Transactions of the Indian National Academy of Engineering*, vol. 5, no. 2, pp. 181-185, 2020, doi: 10.1007/s41403-020-00116-8.
- [11] H. Hossam, R. Tamer, M. Mohsen, O. Ahmed, M. Alaa, J. Mourad, R. Adel, A. Hatem, and S. Sherif, "Enhanced smart commuting with artificial intelligence for intelligent health and safety monitoring in school buses," *Scientific Reports*, vol. 16, Art. no. 9665, 2026, doi: 10.1038/s41598-026-41628-7.