# A REAL-TIME MOBILE SYSTEM FOR SCHOOL BUS TRACKING AND STUDENT CHECK-IN USING FLUTTER AND FIREBASE

Samuel Tsui [1], Quincy Stokes [2]

[1] Lexington High School, 251 Waltham St, Lexington, Ma 02421
[2] University of California, Irvine, Irvine, Ca 92697

***ABSTRACT***

*This paper addresses the problem of unpredictable school bus arrival times, which often results in missed buses, unnecessary waiting, and safety concerns for students. To solve this issue, we propose a mobile application that provides real-time bus tracking along with student check-in and check-out features. The system is developed using Flutter for the user interface and Firebase services for authentication, data storage, and live GPS synchronization [5]. Its key components include role-based navigation, a real-time database that stores bus and route information, and continuous GPS updates from drivers to ensure accurate tracking. During development, several challenges arose, such as preventing false check-ins and maintaining correct route assignments when students or buses change locations. Experiments were conducted to test estimated arrival time accuracy and the reliability of the automated check-out system, both confirming the importance of accounting for real-world variability in GPS data and network updates [6]. Overall, the results demonstrate that a software-based solution can significantly improve school transportation by enhancing safety, reducing stress from uncertainty, increasing convenience, and strengthening communication between students, parents, and school staff.*

***KEYWORDS***

*Safety, Tracking, Location, School bus*

## 1. INTRODUCTION

Every morning, countless students wait at the end of their street for the school bus, often facing significant uncertainty about when it will actually arrive. Because buses are frequently affected by traffic delays, weather conditions, and route adjustments, they rarely arrive at the same time each day. As a result, many students end up standing outside far longer than necessary, exposed to freezing temperatures, heavy rain, or unsafe conditions sometimes for ten to fifteen minutes or more. Worse still, if the bus arrives earlier than expected, students risk missing it entirely, forcing parents to interrupt their schedules or leaving children temporarily stranded. These issues are especially troubling for younger students, who may wait unsupervised without any real-time information about whether the bus is delayed, already passed, or just around the corner. The broader impact of this unreliable system extends beyond daily inconvenience: it causes stress, disrupts routines, and can compromise student safety. Recent data underscores the scale of this problem. A national survey conducted by Zum found that 91% of parents believe the U.S. school bus system needs improvement, with top concerns centered on safety, inefficiency, and reliability [1]. Nearly half of parents reported that their children have experienced negative effects, such as being late to school or missing after-school activities due to bus delays, long routes, and poor communication. These findings make it clear that the current system is falling short. A more modern, transparent,

David C. Wyld et al. (Eds): ACSTY, MLSC, SVC, AIBD, ITCSS, ADCOM, NATP, SOFE – 2026
pp. 225-236, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160418

and real-time solution is urgently needed to make the school commute safer, more predictable, and less stressful for students and families.

The first methodology used RFID tags and GPS hardware to track buses and verify when students boarded [7]. Its goal was to improve safety through basic location and attendance monitoring. However, it required physical tags and lacked systems for detecting when students exited the bus. Our project improved on this by using mobile-based check-ins and automated checkout to eliminate hardware dependence. The second methodology combined GPS tracking with QR code attendance logging. While it successfully paired location data and boarding verification, it relied on students manually scanning codes, which reduced reliability and ease of use. Our project addressed this limitation by automating check-in processes and offering a more intuitive mobile interface with notifications and route customization. The third methodology implemented an Android tracking app that displayed bus locations and estimated arrival times [8]. Although efficient and accessible, it did not track which students were actually onboard. Our project strengthened safety by integrating live student presence tracking.

To solve the issue of unpredictable school bus arrivals, the most effective approach is to create a real-time school bus tracking app that connects students, parents, and drivers through one unified platform. This solution uses GPS technology to provide continuous, accurate location updates for each bus, ensuring that families always know where the bus is and exactly when it will reach their stop. In practice, each school bus driver uses the app on their phone or a provided device, which automatically tracks the bus's movement along its assigned route. Students and parents can then open the app on their own devices to view the bus's live position on a map, monitor its estimated arrival time, and receive automatic notifications when the bus is approaching their stop or if delays occur due to traffic or weather. This eliminates the guesswork that currently forces students to wait outside for long periods, often in unsafe or uncomfortable conditions. Beyond reducing wait times, the app significantly improves communication by giving families instant, reliable updates rather than leaving them in the dark. Parents gain peace of mind knowing where their child's bus is at all times, and students are less likely to miss the bus because the system alerts them when to head outside. For schools, the platform can improve operational efficiency by identifying route issues, tracking delays, and allowing transportation staff to communicate directly with families when necessary. Additionally, enhanced transparency builds trust between schools and the communities they serve, as families feel more informed and supported. By bridging the information gap with accurate, real-time data, this tracking app transforms the daily commute into a smoother, safer, and more predictable experience for everyone involved.

The first experiment focused on evaluating the accuracy of the app's estimated time of arrival calculations, which were based on a simplified distance and speed model. The goal was to determine how closely these predictions matched actual travel times under realistic conditions. To test this, the bus's predicted arrival time was recorded at multiple distances and compared with the real arrival time measured at the stop. The experiment revealed a growing gap between predicted and actual values, especially at longer distances, suggesting that real-world variability such as traffic and road conditions significantly affects accuracy. The second experiment examined the reliability of the auto-checkout feature designed to remove students from being checked-in once they are 100 meters from the bus for 5 minutes. By gradually increasing the distance between the student and the bus, the experiment tested how closely the system matched the intended 100-meter threshold. Findings showed slight deviations caused primarily by GPS inaccuracies and updated delays, though the feature remained functionally dependable.

## 2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

### 2.1. GPS-Based Verification for Student Check-In Accuracy

When developing this app, one potential challenge would be ensuring that the system accurately reflects real-world activity. For example, students could have the ability to check in when boarding their bus, but without additional verification, they might mark themselves as present even if they were not actually on board. To address this challenge, a location-based validation system can be implemented that only allows check-ins when the student's device is within a specific, GPS-verified distance from the bus. This would ensure that check-ins are tied to physical proximity rather than manual input alone. Additionally, I could use an automatic check-out feature that activates if the student's device moves outside that radius for a certain amount of time, preventing situations where students remain marked as present after getting off the bus or accidentally checking in from home. Together, these features would minimize false or mistaken check-ins, improve data accuracy, and increase the reliability of the system for parents, drivers, and schools [9].

### 2.2. Route Locking to Prevent Driver Assignment Conflicts

Another potential challenge would be preventing multiple drivers from tracking the same bus route at the same time. If two drivers were able to select the same route simultaneously, the system would generate conflicting data, leaving students and parents unsure which driver was actually operating the bus. This could lead to inaccurate tracking, incorrect estimated arrival times, and overall confusion during the commute. To address this issue, we could implement a route-locking system that automatically prevents a driver from selecting a route if it is already in use. As soon as a driver begins a route, the system would mark that route as "active," making it temporarily unavailable for others until the first driver officially ends it. This approach would ensure that each route has only one active driver assigned at a time. Additionally, we could include status indicators, such as "in progress" or "available," so drivers clearly understand which routes can be selected. By enforcing exclusive access to each route, the system would eliminate data conflicts, maintain accurate real-time tracking, and provide students and families with consistent, reliable updates every day.

### 2.3. Dynamic Route Updates for Accurate Student Tracking

A final possible challenge would be handling situations where a student's bus assignment changes, which happens more frequently than it may seem. Students may move houses, switch schools, change stops, or be reassigned to a different route by the district due to staffing, scheduling, or capacity adjustments. If the app did not allow students to update their information easily, they could end up following the wrong bus, receiving inaccurate notifications, or missing important updates. To solve this, after the student initially selects a route during onboarding, we could add a dedicated routes screen that allows them to modify their route and stop at any time. This screen would let students browse all active routes, select their correct bus, and choose the stop closest to their home. To further enhance reliability, the system could verify changes by prompting confirmation or offering suggested routes based on the student's location. By implementing this flexible feature, we would ensure that students always see accurate tracking data and receive timely, relevant notifications. Ultimately, this adaptability would make the system more user-friendly, resilient, and dependable as students' transportation needs evolve throughout the school year.

## 3. SOLUTION

The program is structured around three major components that work together to create a complete, efficient, and user-friendly school transportation system: role-based navigation, a central real-time

database, and driver GPS tracking [10]. The experience begins with a login or registration process, where users create an account and proceed through onboarding. During onboarding, the system identifies whether the user is a student or a driver and then directs them to the appropriate interface. This role-based flow ensures that each user only sees tools relevant to their responsibilities. Drivers are taken directly to a dedicated tracking screen where they can select their assigned bus route. Once a route is chosen, the app begins actively sending the driver's live GPS location to the database. This location data updates continuously, enabling accurate real-time tracking for students. Students, on the other hand, are guided to a stop-selection screen after onboarding, where they choose their route and nearest bus stop. Once this is completed, they enter the main home screen, which serves as the central hub of the app. From here, students can navigate to different functional screens such as the live tracking map, their route management page, a check-in system, and their user profile. Each of these screens uses data pulled directly from the central Firebase Realtime Database, which stores key information like the student's selected route and stop, as well as the driver's ongoing GPS updates [11]. Firebase Authentication manages all user accounts and securely distinguishes between student and driver roles. By combining Firebase services with structured Flutter code, the program transitions smoothly from authentication to onboarding and then into role-specific features. This coordinated system ensures that students always have accurate, up-to-date tracking information, while drivers can reliably transmit route data, ultimately creating a seamless and dependable school transportation experience.

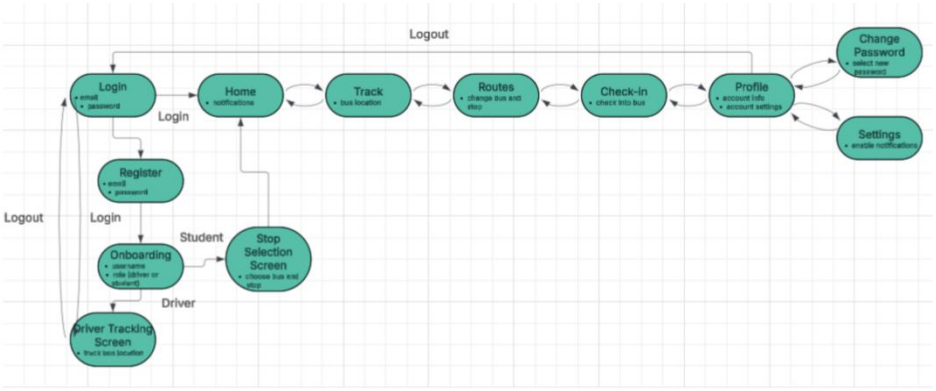


Figure 1. Overview of the solution

One of the core components of our program is role-based navigation, which controls how users move through the app depending on whether they are a student or a driver. The purpose of this component is to ensure that each user only accesses the screens and features meant for their role. For example, drivers should be able to track and update bus locations, while students should be able to view routes and track buses. To implement this system, we used Firebase Authentication to store each user's role after onboarding and identify them when logging in, automatically sending drivers to the driver tracking screen and students to the home screen.

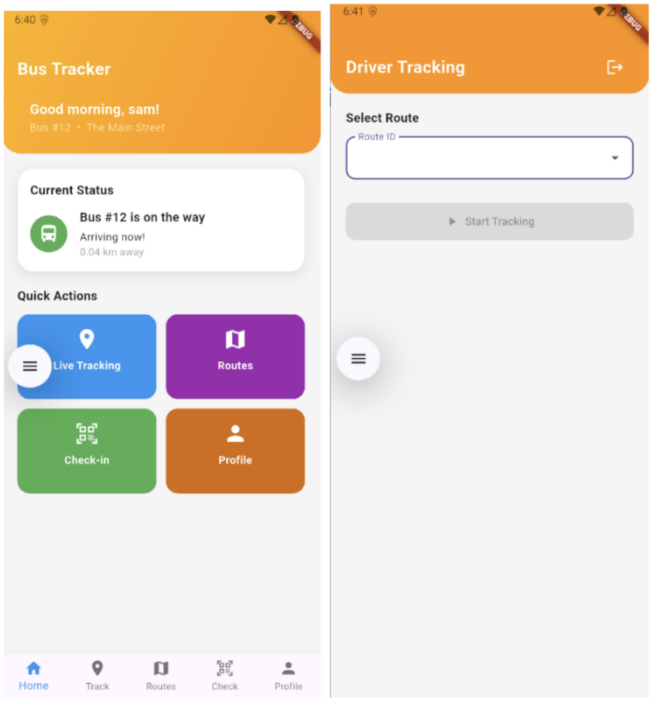


Figure 2. User interface of the mobile application's onboarding screen

```
Future<void> saveUserData() async {
  setState(() => loading = true);
  try {
    //This gives error when testing on phone, right after registering.
    final uid = FirebaseAuth.instance.currentUser!.uid;

    await FirebaseFirestore.instance.collection('users').doc(uid).set({
      'username': username,
      'role': role,
    });

    // Navigate based on role
    if (role == 'driver') {
      Navigator.pushReplacementNamed(context, '/driver_tracking', arguments: uid);
    } else {
      Navigator.pushReplacementNamed(context, '/stop_selection');
    }
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(content: Text('Error: $e')),
    );
  }
  setState(() => loading = false);
}
```

Figure 3. Implementation of the user profile initialization and role-based navigation logic

The code described is the saveUserData() function from our onboarding screen, which is executed when a user taps “Continue” after entering a username and selecting a role. The function begins by calling setState(() => loading = true), which activates a loading spinner in the user interface to indicate that the app is processing the input. Inside a try block, the function retrieves the currently logged-in user’s unique ID from Firebase Authentication. Using this ID, along with the entered username and selected role, it then saves the user’s profile information to Firestore, ensuring the data is securely stored in the cloud. After successfully saving the profile, the function handles role-based navigation: if the user’s role is “driver,” it pushes the driver tracking screen, and if the role is “student,” it pushes the stop-selection screen. In case of any errors during this process, the catch block triggers a SnackBar to display an error message to the user. Finally, setState(() => loading = false) turns off the spinner, signaling that the operation has completed. This function efficiently

combines data storage, user feedback, and role-specific navigation in a single workflow, ensuring a smooth onboarding experience.

Another important component of our program is the central real-time database, built using Firebase Realtime Database. Its purpose is to store and update all bus related information in real time, including bus GPS locations, route assignments, student check-ins, and the full list of routes and stops. In the flow of the program, this component powers several major features: the driver sends live bus coordinates while students receive those coordinates to track the bus; route assignments prevent two drivers from selecting the same route; check-ins allow schools to verify which students boarded which bus; and route and stop data supplies the UI with route and stop names.


Figure 4. Real-time route and stop selection interface for students

```dart
void _listenForRoutes() {
  _routesSub = _db.child("routes").onValue.listen((event) {
    if (!event.snapshot.exists) return;
    // safe conversion
    final raw = event.snapshot.value;
    final mapped = _asStringMap(raw);
    setState(() => routes = mapped);
  });
}

void _listenForUserSelection() {
  final uid = _auth.currentUser!.uid;
  _userSub = _db.child("users/$uid").onValue.listen((event) {
    if (!event.snapshot.exists) return;
    final data = _asStringMap(event.snapshot.value);
    setState(() {
      currentRouteId = data["routeId"]?.toString();
      currentStopId = data["stopId"]?.toString();
    });
  });
}

Future<void> _switchRoute(String routeId, String stopId) async {
  final uid = _auth.currentUser!.uid;
  await _db.child("users/$uid").update({
    "routeId": routeId,
    "stopId": stopId,
  });
  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(content: Text("Route updated successfully")),
  );
}
```

Figure 5. Methods for real-time synchronization with Firebase Realtime Database

This section of code is responsible for managing real-time communication between the app and the Firebase Realtime Database, ensuring that all route and user data stays current across devices. The first method, _listenForRoutes(), sets up a continuous listener on the routes node of the database. Whenever the school updates the list of available bus routes, Firebase sends a new snapshot of the data to the app. The method then converts this snapshot into a map and updates the app's state, which ensures that the user interface always displays the most recent route information without requiring a manual refresh. The second method, _listenForUserSelection(), listens specifically to the logged-in user's data. This means that any time a student's assigned route or stop changes — either through a school update or by the student selecting a new stop — the app immediately updates the currentRouteId and currentStopId variables, keeping the student's view accurate and timely. Finally, the _switchRoute() method writes changes back to the database using .update(), allowing students to modify their route and stop in real time. Firebase manages all of this synchronization behind the scenes, automatically syncing data between the cloud and every connected device. Overall, this code ensures a seamless, live connection between the app and the database, maintaining accurate, up-to-date routes, stops, and user selections at all times.

A third core component of our program is the driver GPS tracking system, which enables real-time monitoring of school bus locations [12]. Its purpose is to continuously capture a driver's live GPS coordinates and broadcast them to students so they can see exactly where their bus is. To add this feature, we used Geolocator to retrieve GPS data from the driver's device and the Firebase Realtime Database to store and distribute those coordinates. The app repeatedly gathers the bus's location every few seconds and whenever it updates, the students receive the new location immediately without having to refresh.

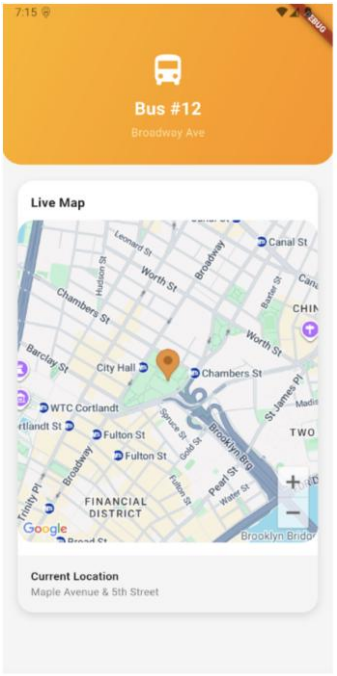


Figure 6. Real-time GPS tracking visualization on the map interface

```
void _listenToBusLocation(String routeId) {
  _locationSub?.cancel();
  _locationSub = _db.child("bus_locations/$routeId").onValue.listen((event) {
    if (!event.snapshot.exists) return;

    final data = Map<String, dynamic>.from(event.snapshot.value as Map);
    final lat = (data['lat'] as num).toDouble();
    final lng = (data['lng'] as num).toDouble();

    setState(() {
      busLocation = LatLng(lat, lng);
      markers = {
        Marker(
          markerId: const MarkerId("bus"),
          position: busLocation!,
          infoWindow: InfoWindow(title: "Bus #$routeId"),
          icon: BitmapDescriptor.defaultMarkerWithHue(
            BitmapDescriptor.hueOrange,
          ),
        ), // Marker
      };
    });

    if (_mapController != null) {
      _mapController!.animateCamera(CameraUpdate.newLatLng(busLocation!));
    }
  });
}
```

Figure 7. Implementation of the live GPS location listener and map marker update logic

This method functions as the live listener for the driver GPS tracking system, forming a critical part of the real-time bus tracking feature. On the driver side, the DriverTrackingScreen continuously writes the driver's current latitude and longitude into the Firebase Realtime Database every few seconds, creating a constantly updated feed of the bus's location. On the student side, the _listenToBusLocation() method subscribes to the exact database path where the driver's location is stored. Each time the driver's device writes a new set of coordinates, the callback function is triggered. The method converts the snapshot received from Firebase into a map, extracts the latitude and longitude, and transforms these values into a LatLng object that can be used with Google Maps [13]. Within setState, the app updates the busLocation variable and rebuilds the map

marker to reflect the bus's new position. Finally, the camera is animated to the updated coordinates, allowing the marker to move smoothly on the map in real time. This setup ensures that students can watch the bus approach their stop accurately, providing a reliable, dynamic visualization of the bus's journey and greatly enhance safety and convenience during the school commute.

# 4. EXPERIMENT

## 4.1. Experiment 1

A possible blindspot of our app is how accurate the estimated time of arrival of the bus is. If it is wrong, it may cause students to miss the bus or wait unnecessarily long.

To test ETA accuracy, we will compare the app's predicted arrival time to the actual arrival time at the student's stop. Our ETA is currently calculated from GPS distance and an assumed average speed of around 60 kilometers per hour. For the experiment, we will have the bus start from 100 kilometers away and record the current time and predicted ETA. Then every 10 kilometers we will again record the current time and predicted ETA. Finally, when the bus reaches the stop, we will record the real arrival time. The ETA error will then be calculated for each distance. This setup directly measures how far off our ETA model is in real conditions.

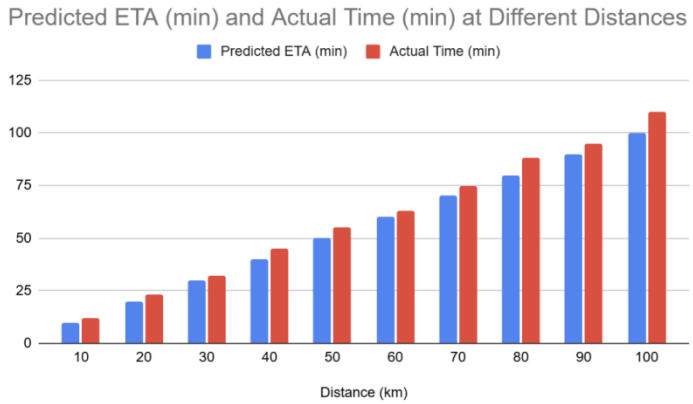


Figure 8. Comparison of predicted ETA vs. actual arrival time across various travel distances

The experiment shows a clear and consistent difference between the predicted ETA and actual ETA measured at every distance. In our data, the predicted ETAs increase linearly from 10 to 100 minutes, matching the simple assumption of a constant bus speed of 60 kilometers per hour. However, the actual travel times are always longer. The smallest difference between ETA and actual time occurs at 10 kilometers, where the bus only takes 2 minutes longer than predicted. As distance increases, the gap widens: at 50 kilometers, the bus takes 5 minutes longer, and by 100 kilometers, the difference grows to 10 minutes. The widening gap suggests that longer-distance predictions accumulate more real-world delays, such as traffic, stops, and traffic lights. The biggest influence on the results is the variability in real world driving conditions, showing that simple linear ETA models work well at short distances but become less accurate the farther away the bus is.

## 4.2. Experiment 2

Another potential blindspot is the reliability of the auto check out- system as the app automatically checks students who are not within a 100m radius after 5 minutes. This is important to see when students are on and off the bus.

To test this feature, we will simulate a student leaving their assigned bus stop while the bus's location updates in real time. The student will start 50 meters away from the bus and wait five minutes to see if the automatic check-out triggers. If it does not, they will move an additional 10 meters further away and repeat the process until the system registers that they have left the stop. This test will help determine how far a student must be from the bus for automatic check-out and how closely this distance aligns with the intended 100-meter threshold. It allows us to verify the accuracy and reliability of the location-based validation system and ensure that students are correctly checked in and out, improving both safety and app functionality.

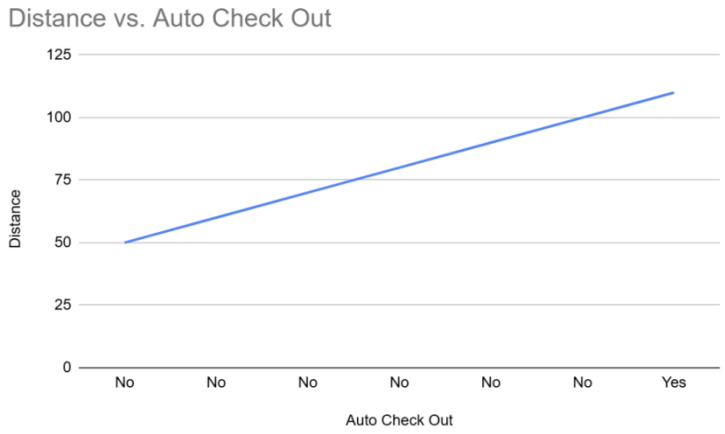


Figure 9. Validation of the auto-checkout trigger distance relative to the 100-meter proximity threshold

The goal of the experiment was to determine the real-world distance at which the auto-checkout system activates. The expected threshold is 100 meters, but factors such as GPS positional error and Firebase update delays can cause some variance. The results suggest that while the system begins functioning near the intended 100-meter mark, real-world GPS fluctuations cause the bus and student coordinates to drift slightly. If either device reports an inaccurate location, the calculated distance can temporarily change, affecting the timing of the auto-checkout. Another factor influencing the results is Firebase's update cycle, which refreshes the bus's location at periodic intervals. This can occasionally make the system register the student as closer to the bus than they actually are. Overall, the largest contributor to variation is GPS inaccuracy, which can depend on the device being used, surrounding buildings or terrain, and signal strength. Despite these minor fluctuations, the system consistently triggers slightly above 100 meters, demonstrating that the auto-checkout feature functions as intended in real-world conditions and provides reliable, practical results.

## 5. RELATED WORK

A scholarly source addressing the same problem is Real-Time School Bus Monitoring System for Enhanced Student Safety Using RFID and GPS Technologies, which proposes a low-cost setup that uses a GPS module to track bus location and an RFID reader to log when students board the vehicle [2]. The microcontroller on the bus transmits these events to a server so parents and school administrators can view real-time location and receive boarding alerts. While effective at providing basic visibility and reducing uncertainty for parents, the system has several limitations including

necessary RFID tags and a lack of advanced safety features such as detecting when students exit the vehicle. The proposed system improves upon this by utilizing a mobile-based application which nearly all people carry around, and it automatically checks students out of the bus when they exit. A second scholarly source addressing this problem is Design of Bus Tracking System Using GPS and QR Code Attendance System, which proposes a hardware centered solution that uses a GPS module to transmit real-time bus location and a QR-code based attendance system to record when students board or exit the bus [3]. While the system successfully integrates location tracking with basic student verification, it remains limited by its dependence on students constantly scanning QR codes, the vulnerability of the hardware modules, and device failures. The design also offers minimal attention to user experience providing no robust interface for parents or school staff. My project improves on this by delivering a dedicated mobile app that provides more features including cleaner visuals, accurate ETAs, and customizable notifications, creating a more user-friendly solution.

A final paper addressing this issue is An Efficient Android Based School Bus Tracking System, which presents a GPS and app-based system that allows real-time tracking of school buses via an Android application [4]. The system includes modules for drivers, parents, and administrators, enabling live bus location monitoring on a map, estimated arrival times at stops, and overall route transparency. Its main strength is that it provides a low-cost, hardware-light solution that improves visibility and reduces waiting time. However, the system does not track which students are actually on the bus, leaving a significant gap in safety and accountability for student boarding and deboarding. My project fixes this problem by integrating a check-in system that verifies when each student boards or exits the bus, enhancing student safety and administrative accountability.

## 6. CONCLUSIONS

Despite its effectiveness, there are still some limitations to the project. Firstly, the system relies heavily on GPS and internet connectivity, so in areas with weak signals or inaccurate device location data, bus positions and estimated arrival times may be unreliable [14]. Continuous GPS tracking can also drain device batteries and consume mobile data, and the current backend infrastructure may face scalability issues as the number of users increases. To address these challenges, future improvements could include an offline mode with automatic data syncing when connectivity is restored, as well as enhanced GPS accuracy and more precise predictive arrival times using traffic and route data. Furthermore, measures such as battery optimization and data compression could improve reliability and convenience for all users. With additional development time, implementing these features would make the app more scalable, robust, and user-friendly, ultimately providing an even better and more seamless experience for students, parents, and drivers [15].

To conclude, this app offers a practical and modern approach to improving student safety and transportation transparency. Its real-time monitoring features highlight how thoughtful technology can make daily school operations more efficient and reassuring for families. Continued development and adoptions of these solutions can further elevate the reliability and accountability of school transportation services.

## REFERENCES

[1] Lenhoff, Sarah Winchell, et al. "Beyond the bus: Reconceptualizing school transportation for mobility justice." Harvard Educational Review 92.3 (2022): 336-360.

[2] Gadade, Bhanudas, A. O. Mulani, and A. D. Harale. "Iot based smart school bus and student monitoring system." Naturalista Campano 28.1 (2024): 730-737.

[3] Suresh Kumar, M., et al. "A GPS Based Bus Tracking and Unreserved Ticketing System Using QR Based Verification and Validation." International Conference on Data Intelligence and Cognitive Informatics. Singapore: Springer Nature Singapore, 2023.
[4] Hemalatha, J., K. Hitaishi, and Vivek Sharma S. Mownika. "An efficient Android Based School Bus Tracking System." IJEAST (2021).
[5] Moroney, Laurence. "Using authentication in firebase." The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform. Berkeley, CA: Apress, 2017. 25-50.
[6] Wang, Wen-Qin. "GPS-based time & phase synchronization processing for distributed SAR." IEEE Transactions on Aerospace and Electronic Systems 45.3 (2009): 1040-1051.
[7] Kiedrowicz, Maciej. "Location with the use of the RFID and GPS technologies-opportunities and threats." GIS ODYSSEY 2016 (2016): 122-128.
[8] Hecht, Geoffrey, et al. "Tracking the software quality of android applications along their evolution (t)." 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2015.
[9] Barchard, Kimberly A., and Yevgeniya Verenikina. "Improving data accuracy: Selecting the best data checking technique." Computers in Human Behavior 29.5 (2013): 1917-1922.
[10] Kao, Ben, and Hector Garcia-Molina. "An overview of real-time database systems." Real Time Computing (1994): 261-282.
[11] Moroney, Laurence. "The firebase realtime database." The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform. Berkeley, CA: Apress, 2017. 51-71.
[12] Dafallah, Hind Abdalsalam Abdallah. "Design and implementation of an accurate real time GPS tracking system." The Third International Conference on e-Technologies and Networks for Development (ICeND2014). IEEE, 2014.
[13] Hu, Shunfu, and Ting Dai. "Online map application development using Google Maps API, SQL database, and ASP .NET." International Journal of Information and Communication Technology Research 3.3 (2013).
[14] Balouch, Feroz Ahmad, Khan Mohammad Wafa, and Ali Ahmad. "Internet of things (iot), its application area and combination with gps." Galaxy International Interdisciplinary Research Journal 10.1 (2022): 725-734.
[15] Lowdermilk, Travis. User-centered design: a developer's guide to building user-friendly applications. " O'Reilly Media, Inc.", 2013.