DECLARATION

We declare that this project report entitled "**School Bus Tracking and Student Attendance System**" has been prepared by us and is the record of the work done by us during the academic year 2024-2025 and, to the best of our knowledge and belief, it has not previously formed the basis for the award of any diploma or degree to us or any other candidate by this or any other University.

Date:

Anosh Chodankar	**Shreyash Vaigankar**
2202303	2202338
Aldous D'Souza	**Samuel D'Souza**
2202301	2202336
Murari Mhamal	**Stanislaus Borges**
2202323	2202341
Clyde D'Souza	**Ayush Maldar**
2202309	2202306

Mapusa - Goa

CERTIFICATE

This is to certify that this project entitled " **School Bus Tracking and Student Attendance System** " is the record of work done by **Mr. Aldous D'Souza, Mr. Anosh Chodankar, Mr. Ayush Maldar, Mr. Clyde D'Souza, Mr. Murari Mhamal, Mr. Samuel D'Souza, Mr. Shreyash Vaigankar and Mr. Stanislaus Borges,** students of T.Y.B.Sc. Computer Science, during the academic year 2024-2025 under my guidance, and to the best of my knowledge and belief, it has not formed the basis of any Degree or Diploma by Goa University or elsewhere.

Mr. Edwin D'Souza
(Head of Department)

Ms. Prajoti Chimulkar
(Project Guide)

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost, we would like to thank and praise God for all the grace and blessings received by us.

We would like to acknowledge, with much appreciation, the crucial role played by Mr. Edwin D'Souza, Head of the Department of Computer Science and Ms. Ursula Barreto, Acting Principal of St. Xavier's College for their constant support throughout the completion of our project work.

We are extremely grateful to our project guide Ms.. Prajoti Chimulkar for her guidance and support.

We would like to extend our thanks to Mr. Elston Dias, Computer Science Laboratory Assistant and Mr. Joaquim D'Souza, Computer Science Laboratory Attendant, for providing all the necessary tools and resources that were essential for the successful completion of our project.

We would be remiss if we forget our loving Parents whose love, care, support, and encouragement have brought us this far in our lives.

Lastly, we would like to express our gratitude to everyone who contributed to the success of this project. Their assistance and encouragement have been truly invaluable, and we are deeply thankful for their support.

Table of Contents

1. Introduction

The School Bus Tracking and Student Attendance System is a service designed to enhance the safety and efficiency of school transportation. It provides real-time tracking of school bus and ensures accurate attendance of students on the bus. By integrating hardware, software and cloud-based technologies, the system aims to address the common challenges faced by schools, parents and bus operator.

The app's functionality is made possible by its connection to a GPS sensor via an ESP32 sensor, which acts as a bridge using Bluetooth technology. This setup enables real-time location access and provides crucial information about its location. The app's interface is highly curated, ensuring a user-friendly experience.

The app's main feature of real time location access is due to the NEO 6M GPS module, this module uses satellite data to get its location, while consuming very less power of around 60 mW. The NEO 6M is responsible to get the location, the ESP32 sends the location data to a mobile phone using the Bluetooth Low Energy (BLE) protocol.

Our app connects to the ESP32 using mobile data to receive the location and transmits it to our server, the server in turn is responsible to provide the location data to the parents and school administration. The app caters to 3 kinds of users, the bus operator, the parents and the school administration. This ensures that a user gets the information that is useful to them.

The case that houses the ESP32 microcontroller, battery, and the GPS module is 3D printed, ensuring a lightweight and durable design. The casing is designed to be compact and unobtrusive while simultaneously ensuring easy access to the components inside.

By leveraging these basic technologies, we empower the school administration to manage the transport facility provided to the students while ensuring their safety by tracking the live location of the bus and keeping track of the number of students currently present on the bus. Parents can also track the bus and get the desired data about the live location of the bus, the lady staff present on the bus, the number of students presently on the bus, the time their child boarded the bus, the route followed by the bus and the approximate time when the child will reach the destination.

1.1. Objectives

The objective of our project is to ease the anxiety of parents and the school administration when the students are travelling between their homes and the school by providing the location of the school bus and a list of students present on the bus. The system combines GPS technology, mobile app interfaces and database management to provide a seamless solution.

The following are the key objectives of the project:

i. Real-Time Tracking: The system continuously tracks the school bus using a GPS module (NEO-6M) installed on the bus. Parents and school authorities can view the exact location of the bus on an interactive map within the app. This feature helps parents anticipate arrival times, reduces waiting time at bus stop, and enhances overall safety by providing a live location feed.

ii. Attendance Management: A QR code scanner integrated into the app will record student attendance when they board and exit the bus. Each student will have a unique QR code on their ID card that gets scanned by the school staff present on the bus.

iii. SOS Feature: In case of an emergency such as a bus breakdown, accident, or any safety concern, the SOS button present on the app operated by the school staff in-charge allows immediate alerts to be sent to parents, school authorities, and emergency services.

iv. Parent Notifications: The system will send automated notifications to parents whenever their child: boards the school bus, reaches school safely, departs from school, arrives at the designated bus stop. These notifications will be sent via push notifications through the app, keeping the parents informed and ensures better communication between school transport authorities and families.

v. Data Reporting: The app will generate detailed reports on student attendance records on the bus per trip and provide a monthly summary of the attendance. These reports can be accessed by school administrators and parents to enhance safety measures, and ensure compliance with school transport policies.

2. Literature Review

The safety and security of school children during transit a is critical concerns for parents, schools, and the school transport authorities. Traditional school transportation systems lack real-time monitoring of the bus, leading to inefficiencies and potential safety risks. IoT-based solutions incorporating GPS tracking, QR codes and mobile applications have emerged to enhance student monitoring and school bus tracking. This literature review explores existing research on these technologies, highlighting their methodologies, benefits, and limitations.

A. GPS-Based Tracking Systems

Several studies have investigated the application of GPS technology to track school buses in real-time, improving safety and operational efficiency. These systems utilize GPS modules integrated with communication networks to provide location updates.

Ruturaj Shelake, Reshma Chavan, Raju Rai and Mangesh Manake [1]. This study presents a system that integrates GPS tracking with a centralized monitoring server. The system enables real-time alerts, route tracking, and mobile access, ensuring the security of children during transportation. However, the study highlights a dependency on stable internet connectivity, which poses challenges in areas with poor network coverage.

Moechammad Sarosa, Mentari Tika Putri Ningrum and Putri Elfa Mas'udia [2]. This research develops a mobile application-based solution that provides real-time updates on school bus trips. The system allows parents to monitor their children's journey and receive estimated arrival times, reducing waiting periods at bus stops. While the solution enhances user convenience, it requires regular GPS updates, which may drain battery life on mobile devices.

Wendy Mwende Mbabu [3]. This study introduces a comprehensive tracking system that monitors vehicle parameters such as location, speed, and passenger lists. The research focuses on enhancing student safety by integrating multiple sensors to detect driver behavior. However, privacy concerns arise regarding the storage and transmission of student location data.

B. Attendance Tracking Solutions

Another crucial aspect of student monitoring is attendance verification, ensuring that children board and exit the correct bus.

Khang Jie Liew and Tee Hean Tan [4]. This study introduces a system that generates unique QR codes for each class session. Students scan the code upon entering the bus and the system records their attendance, enhancing accuracy and preventing proxy attendance. The research highlights how QR codes can streamline the attendance process and reduce administrative workload.

K Kiran Kumar, Pattan Firoze, K Ramesh Babu and Samarouthu Mounika [5]. This research describes a system that incorporates QR codes and internet-connected devices to take student attendance. The study demonstrates that QR codes can efficiently replace traditional roll-call methods and improve accuracy in student monitoring.

QR code technology offers a low-cost and easy-to-implement solution for school bus attendance tracking. Unlike RFID-based systems, QR codes do not require specialized hardware, making them a more accessible choice for schools. The primary challenge of QR-based attendance logging is the need to actively scan the code, which may lead to issues in cases of non-compliance or device malfunctions.

3. Hardware Description

3.1. Hardware Used

3.1.1. ESP32

The ESP32 is not only a budget-friendly SoC but also a powerhouse of features that make it an ideal choice for both hobbyists and professionals alike. In addition to its built-in Wi-Fi and Bluetooth capabilities, its dual-core processor and generous memory help it handle complex tasks with ease. Here's a deeper dive into its capabilities, with a special focus on its pin configuration and peripheral interfaces.

Key Features and Capabilities

- **Wireless Connectivity:** Built-in 2.4 GHz Wi-Fi (supporting 802.11 b/g/n) and Bluetooth (including BLE) enable a wide range of wireless communication, making it perfect for IoT applications such as remote monitoring and control.
- **Processing Power:** The dual-core Tensilica Xtensa LX6 processor (operating at up to 240 MHz) delivers high performance, allowing simultaneous multitasking and real-time data processing.
- **Memory and Storage:** With 520 KB of SRAM and additional flash memory (commonly 4 MB), the ESP32 can run complex applications and store large amounts of code and data.

- **Programming Flexibility:** Whether you prefer the Arduino IDE, the more advanced ESP-IDF, or even MicroPython, the ESP32's ecosystem supports multiple development environments.

3.1.2. USB A to Micro USB Data and Power Cable

The micro-USB cable is an essential accessory for the ESP32 microcontroller, serving multiple critical functions:

1. **Power Supply:** It provides the necessary power to the ESP32 board, enabling it to operate effectively.
2. **Firmware Flashing:** The cable facilitates the uploading of firmware and sketches from a computer to the ESP32, a process commonly known as flashing.
3. **Serial Communication:** It enables serial debugging and communication between the ESP32 and development environments like the Arduino IDE, allowing developers to monitor outputs and troubleshoot efficiently.

3.1.3. u-blox NEO-6M GPS Module

The ublox NEO6M is a widely used, high-performance GPS module engineered for precise positioning and tracking applications. It offers a robust set of features that make it ideal for integration into IoT projects—such as a school bus monitoring system—where Realtime location data is crucial. Below is a detailed look at its capabilities and benefits:

Detailed Overview

High Sensitivity and Accuracy:

• The NEO6M is designed to quickly acquire satellite signals even in challenging environments, delivering fast "time-to-first-fix" performance.
• Its high sensitivity and reception enable it to track multiple satellites simultaneously, ensuring an accurate fix for latitude, longitude, altitude, and speed.

Persistent Configuration Storage:

• The settings persist across power cycles, meaning the module resumes operation in your preferred configuration without needing to be re-programmed each time.

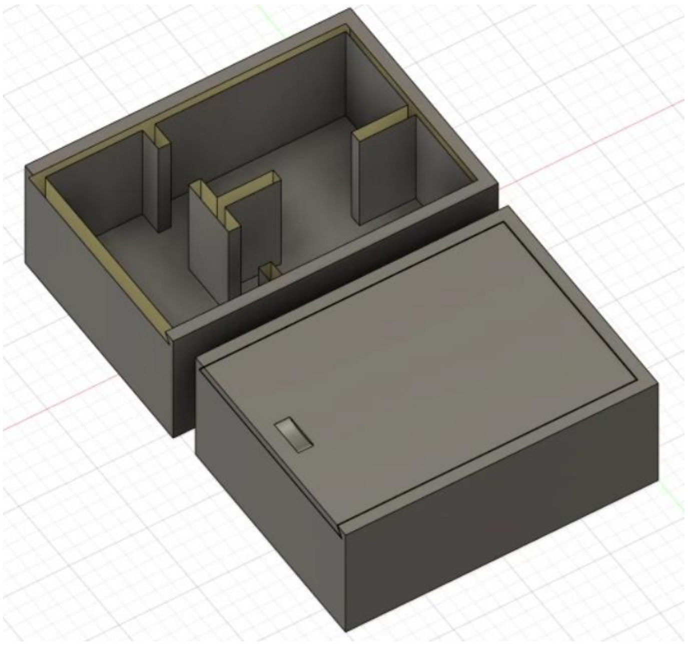
Communication Protocols:

• UART is the standard interface used for communication with microcontrollers like the ESP32, making integration straightforward through libraries such as TinyGPS++ or NeoGPS.

Backup Battery and Fast Reacquisition:

• Many NEO6M modules include an onboard backup battery (or coin cell) that powers the realtime clock (RTC) and preserves satellite data (ephemeris).
• This capability dramatically reduces the "hot start" time, allowing the module to reacquire satellite signals in as little as one second after a brief power interruption.

3.1.4. 3D Printed Case



A 3D-printed case is more than just a protective shell—it's an integral part of our design that enhances usability, durability, and aesthetics. In addition to its custom fit for components like the ESP32, GPS module, battery, and sensors.

By integrating these elements into our 3D-printed case design, we created a protective, functional, and visually appealing enclosure that not only safeguards our electronics from environmental hazards and mechanical shocks but also enhances the user experience and overall product quality.

4. Software Description

4.1. Software Used

4.1.1. Figma

Figma is a powerful cloud-based design tool used for creating user interfaces (UI), user experience (UX) prototypes, and vector-based graphics. Unlike traditional design software that requires local installation, Figma operates entirely within a web browser, allowing real-time collaboration between designers, developers, and stakeholders. It features an intuitive drag-and-drop interface with a wide range of tools for creating wireframes, mockups, and interactive prototypes.

4.1.2. Android Studio

Android Studio is the official integrated development environment (IDE) for Android app development, created by Google. It provides a comprehensive set of tools for building, testing, and debugging Android applications. Based on IntelliJ IDEA, Android Studio offers a rich environment tailored specifically for mobile development, including an advanced code editor, a powerful layout editor, and an emulator for testing apps across different Android devices.

Android Studio supports multiple programming languages, including Java, Kotlin, and C++, making it versatile for different development needs. Its integrated emulator allows developers to test applications on various Android devices without needing physical hardware, and its profiling tools help analyze app performance, memory usage, and network activity.

4.1.3. PostgreSQL (Postgres)

PostgreSQL, commonly known as Postgres, is a powerful open-source relational database management system (RDBMS) known for its robustness, scalability, and compliance with SQL standards. It is widely used in enterprise applications, data warehousing, and web applications due to its ability to handle large datasets and complex queries efficiently.

PostgreSQL is also optimized for concurrent connections, using MVCC (Multi-Version Concurrency Control) to handle multiple transactions without locking the database. Its high-availability features, including replication, clustering, and failover support, make it a preferred choice for mission-critical applications.

4.1.4. pgAdmin

pgAdmin is a graphical user interface (GUI) management tool for PostgreSQL, designed to make database administration easier. It provides a user-friendly interface for creating, modifying, and managing PostgreSQL databases without needing to write SQL queries manually.

pgAdmin offers a dashboard-style interface that displays database statistics, running queries, and performance insights. Developers and database administrators (DBAs) can use it to create, edit, and delete tables, indexes, functions, and schemas. It also features a SQL query editor with syntax highlighting and autocomplete, making it easier to run complex SQL queries and view results quickly.

4.1.5. Visual Studio Code (VS Code)

VS Code is a lightweight yet powerful source code editor developed by Microsoft. It supports a wide range of programming languages, including Python, JavaScript, Java, C++, and Go, making it one of the most versatile code editors available.

One of its biggest strengths is its extensible architecture—VS Code has a vast extension marketplace where developers can add support for linting, debugging, version control, and cloud integrations. It includes built-in support for Git, allowing users to manage source control directly within the editor.

4.1.6. Arduino IDE

The Arduino IDE is an open-source development environment used for programming Arduino microcontrollers and similar embedded systems. It features a simple code editor, serial monitor, and one-click compilation/upload functionality, making it accessible for beginners and experienced developers alike.

While the IDE is simple, it supports advanced features like external board support (ESP32, STM32, etc.), allowing it to program a variety of microcontrollers beyond just Arduino.

4.1.7. nRF Connect

nRF Connect is a suite of development tools for Bluetooth Low Energy (BLE) and IoT applications. Its primarily used for configuring, testing, and debugging BLE devices. It provides real-time packet inspection, making it invaluable for diagnosing connection issues and analysing BLE communication.

4.2. Libraries Used

4.2.1. BLEDevice.h

The BLEDevice.h header file is a core part of the ESP32 BLE stack, enabling the initialization and management of Bluetooth Low Energy functionalities. It provides the primary interface for setting up the BLE environment, including initializing the device as a BLE peripheral, configuring BLE services, and handling advertising processes. This header allows the ESP32 to act as a BLE device that can broadcast data to other devices such as smartphones or other microcontrollers. By including BLEDevice.h, the program gains access to functions that facilitate creating and managing BLE devices, setting up services, and ensuring that the ESP32 properly interacts with BLE clients. It serves as the foundation for BLE connectivity, ensuring that all further BLE functionalities can be structured on top of it.

4.2.2. BLEServer.h

The BLEServer.h file is another fundamental component of the ESP32's BLE stack, enabling the creation of a BLE server. Unlike BLE clients, which scan for and connect to available devices, a BLE server acts as the broadcaster, providing data to clients that request it. In this program, the ESP32 operates as a BLE server, constantly advertising its presence and allowing BLE clients to connect and retrieve GPS data. The BLEServer.h file contains methods for handling connections and disconnections, managing multiple connected clients, and defining callback functions that execute specific actions when a client interacts with the server. It is particularly useful in the context of IoT applications where a central device (such as a smartphone or another microcontroller) needs to collect data from a BLE-enabled peripheral like the ESP32. By using BLEServer.h, the ESP32 is capable of efficiently managing BLE connections and ensuring that incoming requests from clients are handled properly.

4.2.3. BLEUtils.h

The BLEUtils.h file is a supplementary utility library that provides additional functionality required for handling BLE operations. While the core BLE functionality is established using BLEDevice.h and BLEServer.h, this header enhances the experience by offering helper functions that facilitate BLE event management, UUID handling, and data formatting. It plays a crucial role in ensuring smooth interactions between the ESP32 and connected BLE devices, providing debugging tools and extended functionalities that make the implementation of BLE services more flexible and robust. In many cases, this file is useful for debugging BLE connections, helping developers analyze advertising packets, device discovery events, and connection statuses. It supports the seamless integration of BLE into the ESP32 environment by managing low-level details that might otherwise complicate the development process.

4.2.4. BLE2902.h

The BLE2902.h file is an essential component that enables the use of Bluetooth descriptors, specifically the Client Characteristic Configuration Descriptor (CCCD), which is necessary for BLE notifications. BLE notifications allow the ESP32 server to send updated data to a connected client without the client having to repeatedly request it. The BLE2902 descriptor ensures that a connected client can subscribe to characteristic updates and receive new data whenever the value changes. In this program, BLE2902.h is used to attach a descriptor to the GPS data characteristic, allowing BLE clients to enable notifications and receive GPS coordinates in real-time. This functionality is crucial for applications that require continuous updates, such as live location tracking, as it minimizes the need for constant polling from the client and instead pushes data whenever a new GPS reading is available.

4.2.5. TinyGPS++.h

The TinyGPS++.h header file is a widely used library designed for parsing and interpreting GPS data received from serial GPS modules. Unlike raw GPS output, which consists of complex NMEA (National Marine Electronics Association) sentences, TinyGPS++ simplifies the extraction of meaningful data such as latitude, longitude, speed, altitude, and time. This library continuously processes incoming GPS data, checking for valid updates and extracting only the latest and most reliable location information. In the given program, TinyGPS++ is used to decode the serial GPS output, ensuring that only the latest coordinates are sent over BLE. The library also includes additional features such as detecting whether a GPS fix is available, handling various GPS message formats, and filtering out invalid data to improve accuracy. It significantly reduces the complexity of working with GPS data, making it an invaluable tool for ESP32-based location tracking applications.

4.2.6. HardwareSerial.h

Finally, the HardwareSerial.h header file is an ESP32-specific library that provides advanced serial communication capabilities. The ESP32 has multiple hardware UART (Universal Asynchronous Receiver-Transmitter) interfaces, allowing it to communicate with external devices such as GPS modules, sensors, and other microcontrollers. By including HardwareSerial.h, the program is able to define and configure a dedicated hardware serial port (UART2) for reading GPS data without interfering with the primary USB serial communication used for debugging. This separation ensures efficient data handling, as GPS data is read from a dedicated serial interface while the main serial port remains available for logging and debugging purposes. The library allows developers to specify the UART pins, baud rate, and communication parameters, offering greater flexibility when working with multiple serial devices on the ESP32.

4.2.7. Osmdroid

The osmdroid library is a powerful, open-source alternative to Google Maps for Android that enables developers to integrate OpenStreetMap (OSM) into their applications. Unlike Google Maps, which requires an API key and relies on proprietary services, osmdroid is fully open-source and operates independently, allowing for offline maps, custom tile sources, and extensive customization options. This makes it particularly useful for projects that require full control over map data, work in offline scenarios, or need an alternative mapping solution that does not depend on Google services.

At its core, osmdroid provides a MapView widget that functions similarly to the Google Maps MapView, allowing developers to embed interactive maps within their applications. The MapView supports various features, including zooming, panning, rotating, and multi-touch gestures, ensuring a smooth user experience. Unlike Google Maps, which loads tiles dynamically from Google's servers, osmdroid allows developers to load map tiles from multiple sources, including local storage, custom tile servers, and OSM tile servers. This ability to use offline tiles makes osmdroid a preferred choice for applications that need to function in remote areas with limited or no internet access. The library automatically caches downloaded tiles, reducing bandwidth usage and improving performance in low-connectivity scenarios.

4.3. Languages Used

4.3.1. Java

Java is a popular object-oriented programming language. Java is used to develop mobile apps, web apps and much more. To create and run Java applications there is a set of tools that are available with Java JDK (Java Development Kit). The Java Development Kit (JDK) is designed to be platform-independent, meaning it can run on various operating systems like Windows, MacOS, and Linux. Platform independence is possible as Java uses a "write once, run anywhere" approach, which allows Java programs to be compiled into a format called bytecode. The bytecode is not specific to any one 22 platform but is instead designed to be interpreted by the Java Virtual Machine (JVM) on different platforms.

4.3.2. XML

XML, or Extensible Markup Language, is a versatile and structured format used for storing and exchanging data across different platforms and systems. Unlike HTML, which focuses on displaying information, XML is designed to define and transport data through custom, user-defined tags. It follows a hierarchical structure, making it easy to organize complex datasets in a readable format. XML is widely used in web services, configuration files, and document storage, serving as the foundation for technologies like SOAP and RSS feeds. Its strict syntax rules ensure consistency, while its platform independence allows seamless integration across various programming languages and applications.

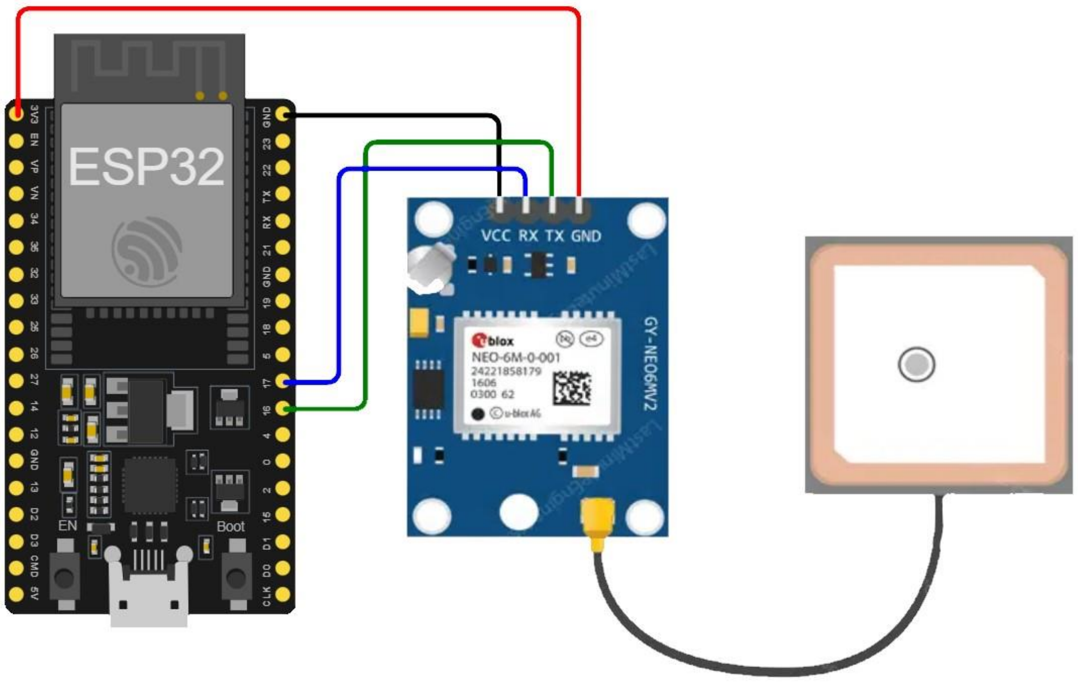
4.3.3. Arduino IDE Language

The Arduino IDE language is based on C and C++, designed to simplify programming for Arduino microcontrollers. It provides a streamlined environment with built-in functions that make it easy to interface with hardware components like sensors, motors, and LEDs. The language includes specialized functions for handling input and output, timing, and communication, allowing users to write code that interacts directly with electronic circuits. With a structure based on a simple setup and loop function, Arduino programs, known as sketches, run continuously, making it ideal for real-time applications. The Arduino IDE also includes a vast library system, enabling users to extend functionality effortlessly, whether for robotics, IoT, or embedded systems.

5. Location Tracking

Our project integrates an ESP32 microcontroller with a NEO-6M GPS module to acquire real-time location data and transmit it wirelessly via Bluetooth Low Energy (BLE). The ESP32 serves as both a GPS data processor and a BLE server, allowing our app to receive GPS coordinates. The system is designed for low-power, real-time tracking, making it suitable for applications such as asset tracking, vehicle monitoring, and outdoor navigation.

5.1. Circuit Diagram



The NEO-6M GPS module is powered directly from the ESP32, which itself receives power through its micro USB port. This setup ensures a compact and integrated design without requiring an additional power source for the GPS module. The ESP32 can be conveniently powered using a standard USB A to micro USB cable, making it adaptable for various power sources, including vehicle USB ports. Most music systems feature USB ports intended for connecting pen drives, providing a stable 5V power output. We use this port to power our system. This setup is advantageous as it allows the system to be powered near the bus operator, ensuring the device remains secure and out of reach of children, minimizing potential tampering or damage.

The ESP32 and the NEO-6M GPS module communicate over UART (Universal Asynchronous Receiver-Transmitter) protocol, which enables efficient and reliable serial communication. The TX (transmit) pin of the GPS module is connected to the RX (receive) pin of the ESP32, while the RX pin of the GPS module is connected to the TX pin of the ESP32. This bidirectional connection ensures seamless data exchange, allowing the ESP32 to continuously receive real-time location data from the GPS module.

5.2. Working

Initializing UART for GPS Communication

```
#include <TinyGPS++.h>
#include <HardwareSerial.h>

TinyGPSPlus gps;
HardwareSerial GPS_Serial(2); // Using UART2 for GPS communication

void setup() {
   Serial.begin(115200); // Debugging serial monitor
   GPS_Serial.begin(9600, SERIAL_8N1, 16, 17); // RX = GPIO16, TX = GPIO17
}
```

The above snippet initializes the ESP32's communication with the NEO-6M GPS module using the TinyGPS++ library. This library helps parse GPS data effectively. A HardwareSerial instance is created to facilitate communication over UART2, using GPIO16 and GPIO17 as the RX and TX pins, respectively. The baud rate is set to 9600 bps, which is the default for the NEO-6M GPS module. The Serial.begin(115200); function initializes serial communication for debugging, ensuring that real-time data logs can be viewed in the serial monitor. This configuration ensures the ESP32 can continuously receive GPS data, making it essential for location tracking applications.

Setting up BLE on ESP32

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

#define SERVICE_UUID "12345678-1234-1234-1234-123456789abc"
#define CHARACTERISTIC_UUID "abcd1234-5678-1234-5678-abcdef123456"

BLECharacteristic *pCharacteristic;
BLEAdvertising *pAdvertising;

void setup() {
  BLEDevice::init("ESP32 GPS");
  BLEServer *pServer = BLEDevice::createServer();
  BLEService *pService = pServer->createService(SERVICE_UUID);
  pCharacteristic = pService->createCharacteristic(
    CHARACTERISTIC_UUID,
                                BLECharacteristic::PROPERTY_READ        |
BLECharacteristic::PROPERTY_NOTIFY
  );
  pCharacteristic->addDescriptor(new BLE2902());
  pService->start();
  pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->start();
}
```

This snippet initializes Bluetooth Low Energy (BLE) functionality on the ESP32. The ESP32 is configured as a BLE server with a specific service UUID and a characteristic UUID, allowing it to communicate with BLE-enabled devices. The characteristic is assigned the READ and NOTIFY properties, meaning connected devices can read the GPS data and receive real-time notifications when new data is available. A BLE2902 descriptor is added to handle BLE notifications properly. Finally, the BLE advertising process is started, allowing external devices to discover and connect to the ESP32. This setup ensures seamless wireless data transmission of GPS coordinates to the mobile app

Handling BLE Connection and Disconnection

```
class MyServerCallbacks : public BLEServerCallbacks {
  void onConnect(BLEServer* pServer) {
    Serial.println("Client connected");
  }
  void onDisconnect(BLEServer* pServer) {
    Serial.println("Client disconnected, restarting advertising");
    pAdvertising->start();
  }
};
```

The MyServerCallbacks class is responsible for handling BLE connection events. When a client connects, the onConnect() function logs a message to indicate successful pairing. If the client disconnects, the onDisconnect() function automatically restarts BLE advertising, ensuring that the ESP32 remains discoverable to new devices. This functionality is crucial for maintaining uninterrupted BLE communication, especially in dynamic environments where connections may drop frequently.

Implementing FreeRTOS for GPS and BLE Processing

```
TaskHandle_t GPSTaskHandle;
TaskHandle_t BLETaskHandle;
```

```
void GPSTask(void *parameter) {
  while (true) {
    bool locationSent = false;

    while (!locationSent) {
      while (GPS_Serial.available() > 0) {
        char c = GPS_Serial.read();
        gps.encode(c);

        if (gps.location.isUpdated()) {
         String gpsData = String(gps.location.lat(), 6) + "," + String(gps.location.lng(),
6);
          Serial.println("Sending GPS Data: " + gpsData);
          pCharacteristic->setValue(gpsData.c_str());
          pCharacteristic->notify();
          locationSent = true;
        }
      }
      vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
     vTaskDelay(10000 / portTICK_PERIOD_MS); // Wait 10 seconds before next
update
  }
}
```

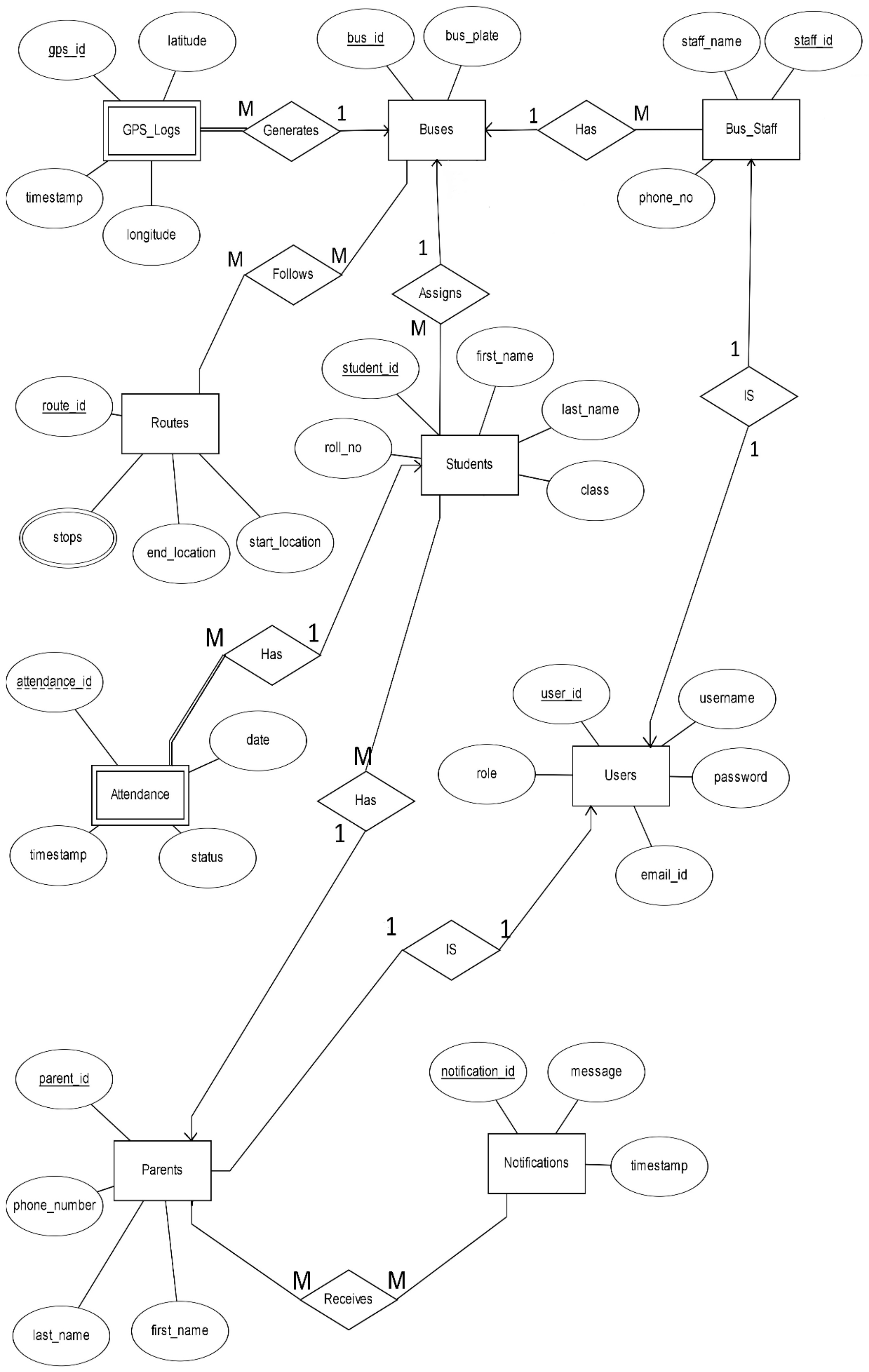
The GPSTask() function runs as a FreeRTOS task, enabling real-time GPS data processing while the ESP32 handles BLE communication in parallel. This function continuously checks if new GPS data is available. Once a valid latitude and longitude reading is obtained, it is formatted as a string and sent over BLE using pCharacteristic->setValue() and pCharacteristic->notify(). The use of vTaskDelay() ensures that the task does not consume excessive processing power, optimizing the ESP32's performance and energy efficiency.

In conclusion the ESP32 reads location data from the NEO-6M GPS module, processes it using TinyGPS++, and transmits it via Bluetooth Low Energy (BLE) to a remote device. Using FreeRTOS tasks, the system ensures that GPS data processing and BLE communication operate efficiently in parallel. The modular and power-efficient design makes this project highly adaptable for real-world applications such as tracking and geolocation-based systems.

6. Database

Our server hosts a robust relational database that forms the backbone of our school management system. The database schema is composed of nine interrelated tables—Users, Parents, Students, Routes, Buses, Bus_Staff, Attendance, GPS_Logs, and Notifications each designed to store a specific category of information while maintaining strong relationships among data points. This meticulously structured schema not only enhances data integrity through foreign key constraints but also ensures efficient retrieval, update, and reporting of information critical to running a school effectively.

6.1. Entity Relationship Diagram



6.2. Schema Design

Table Name: users

Field	**Data Type**	**Description**
user_id	serial	Unique identifier for each user.
first_name	character varying(50)	User’s first name.
last_name	character varying(50)	User’s last name.
phone_number	character varying(15)	User’s phone number; must be unique.
email	character varying (100)	User’s email address; must be unique.
password	text	User’s password (ideally stored in hashed format).
role	character varying(20)	The role assigned to the user (e.g., admin, parent, staff).

Primary Key: user_id

Purpose: Manages system user accounts by storing contact details, credentials, and assigned roles (such as admin, parent, or staff).

Table Name : parents

Field	**Data Type**	**Description**
parent_id	serial	Unique identifier for each parent record.
user_id	integer	References the user account (FK to the users table).

Primary Key: parent_id

Foreign Key: user_id → users(user_id)

Purpose: Links parent-specific details to a user account, enabling parent-related communication and tracking.

Table Name: students

Field	Data Type	Description
student_id	serial	Unique identifier for each student.
first_name	character varying(50)	Student’s first name.
last_name	character varying(50)	Student’s last name.
class	character varying(20)	Class or grade level of the student.
parent_id	integer	References the parent (FK to the parents table).
bus_id	integer	References the assigned bus (FK to the buses table).
roll_no	integer	The roll number assigned to the student.

Primary Key: student_id

Foreign Keys: parent_id → parents(parent_id), bus_id → buses(bus_id)

Purpose: Stores student details including names, class, roll number, and associations with a parent and bus for transportation.

Table Name: attendance

Field	Data Type	Description
attendance_id	serial	Unique identifier for each attendance record.
student_id	integer	References the student (FK to the students table).
date	date	The date on which the attendance is recorded.
status	character varying(10)	Indicates attendance status (e.g., present, absent).
timestamp	timestamp without time zone	Automatically records when the record was created.

Primary Key: attendance_id

Foreign Key: student_id → students(student_id)

Purpose: Records daily attendance for students including date, status (e.g., present/absent), and a creation timestamp.

Table Name: buses

Field	Data Type	Description
bus_id	serial	Unique identifier for each bus.
bus_plate	character varying(20)	Unique plate number assigned to the bus.
route_id	integer	References the route (FK to the routes table).
driver_id	integer	References the bus driver (FK to bus_staff table).
attendant_id	integer	References the bus attendant (FK to bus_staff table).

Primary Key: bus_id

Foreign Keys: route_id → routes(route_id), driver_id → bus_staff(staff_id), attendant_id → bus_staff(staff_id)

Purpose: Maintains details for each bus, including its license plate, associated route, assigned driver, and attendant.

Table Name: routes

Field	Data Type	Description
route_id	serial	Unique identifier for each route.
route_name	character varying(100)	The name of the route.
start_location	character varying(255)	The starting point of the route.
end_location	character varying(255)	The ending point of the route.
stops	jsonb	A JSON object containing details of stops along the route.

Primary Key: route_id

Purpose: Defines bus routes with details such as route name, starting and ending locations, and a JSONB field for stop details.

Table Name: bus_staff

Field	Data Type	Description
staff_id	serial	Unique identifier for each bus staff member.
user_id	integer	References the associated user (FK to the users table).

Primary Key: staff_id

Foreign Key: user_id → users(user_id)

Purpose: Stores information about bus staff (drivers and attendants) and links them to user accounts.

Table Name: gps_logs

Field	Data Type	Description
gps_id	serial	Unique identifier for each GPS log entry.
bus_id	integer	References the bus (FK to the buses table).
latitude	numeric(10,7)	Latitude coordinate of the bus's location.
longitude	numeric(10,7)	Longitude coordinate of the bus's location.
timestamp	timestamp without time zone	Automatically records when the GPS data was logged.

Primary Key: gps_id

Foreign Key: bus_id → buses(bus_id)

Purpose: Logs GPS coordinates (latitude and longitude) and timestamps for each bus to track its location over time.

Table Name: notifications

Field	Data Type	Description
notification_id	serial	Unique identifier for each notification record.
parent_id	integer	References the parent receiving the notification (FK to parents table).
message	text	The content of the notification message.
timestamp	timestamp without time zone	Automatically records when the notification was created.

Primary Key: notification_id

Foreign Key: parent_id → parents(parent_id)

Purpose: Manages notifications sent to parents, recording message content and the time of notification.

Key Benefits and Functional Areas of the Database

- **Comprehensive System Integration:**
  The database is designed to serve as a robust backbone for a school management system by integrating various functions—from user management to real-time bus tracking—into a cohesive structure.

- **Enhanced Security and Access Control:**
  Role-based access control is implemented across the system, ensuring that administrators, parents, and bus staff have access to features and data relevant to their responsibilities, thereby enhancing security and user experience.

- **Seamless Family and Student Record Management:**
  By structuring parent and student data effectively, the database maintains clear and accurate relationships between students and their guardians. This ensures that communication and record-keeping are efficient and reliable.

- **Optimized Transportation Management:**
  The system facilitates precise tracking of bus assignments and routes, ensuring that students are connected with the appropriate transportation services. This optimization supports effective scheduling and resource allocation.

- **Robust Attendance and Operational Monitoring:**
  Integrated attendance monitoring on school buses provides accountability, while real-time tracking of bus locations improves operational safety and efficiency.

- **Timely Communication and Alerts:**
  A dedicated notifications mechanism ensures that important updates—such as bus status alerts—are communicated promptly to parents, supporting proactive engagement and responsiveness.

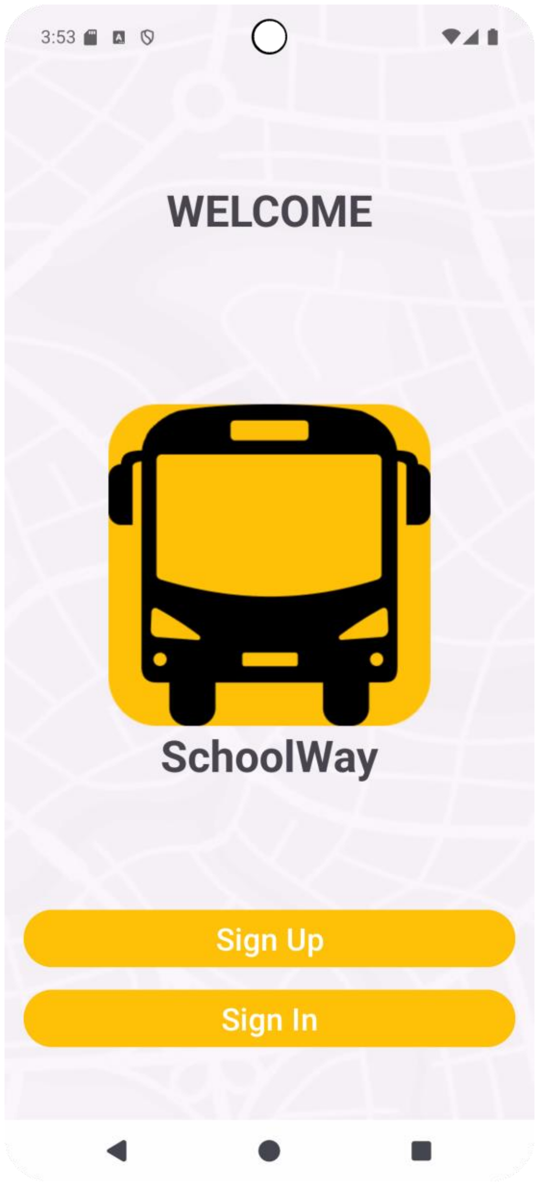
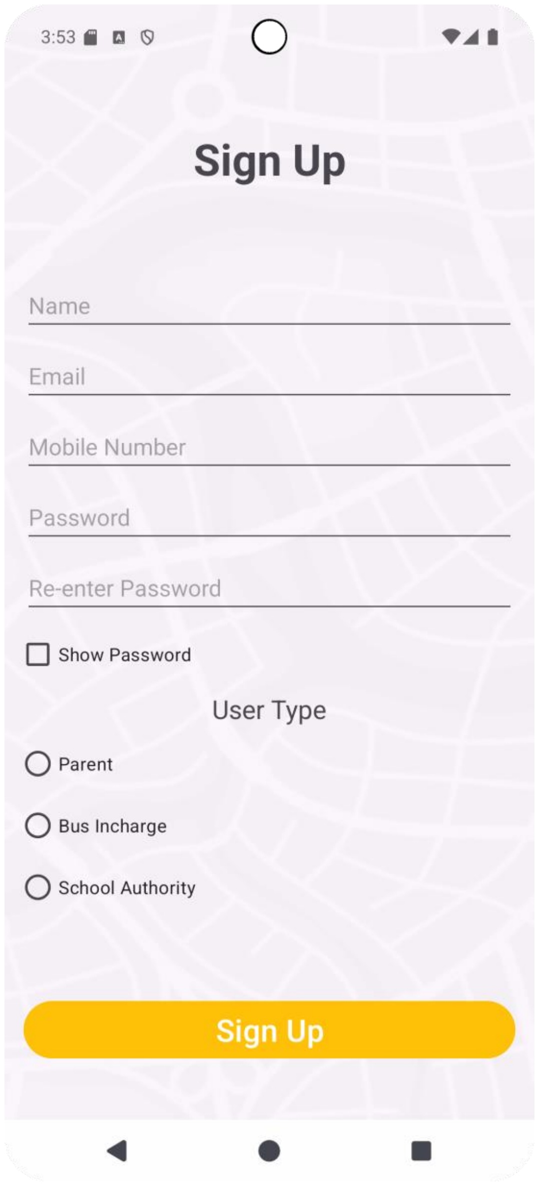
- **Data Integrity and Scalability:**
  The database is built with stringent data integrity constraints, including primary keys and foreign key relationships, ensuring that information remains accurate and reliable as the system scales.

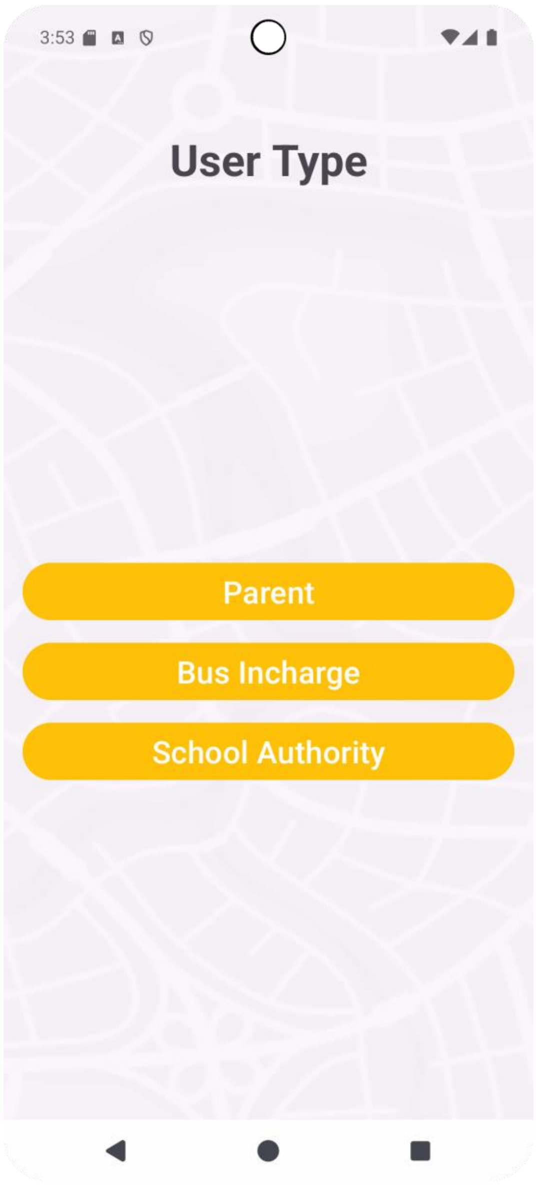
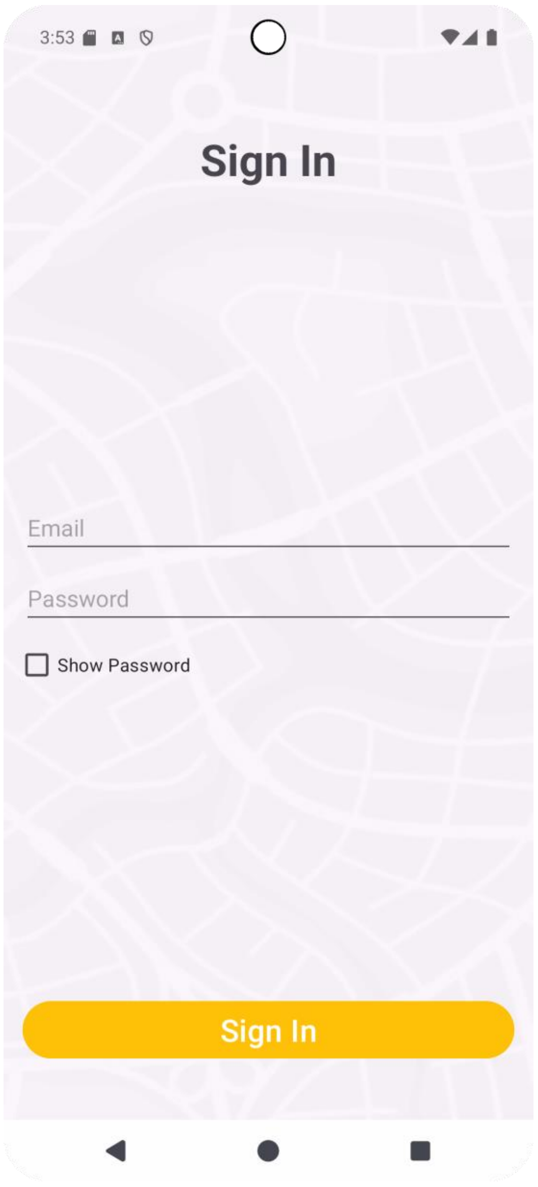
7. App

The app brings together all key aspects of the system, making it a one-stop solution for parents, school authorities, and bus operators. Each user role has a customized interface, ensuring that the information presented is relevant to their needs. Parents can view real-time bus locations, receive notifications about delays or arrivals, and stay updated on their child's attendance. School administrators can track student attendance, manage buses and routes, and communicate with parents efficiently. Bus operators have access to route details, vehicle status, and GPS data for safe and timely transportation.

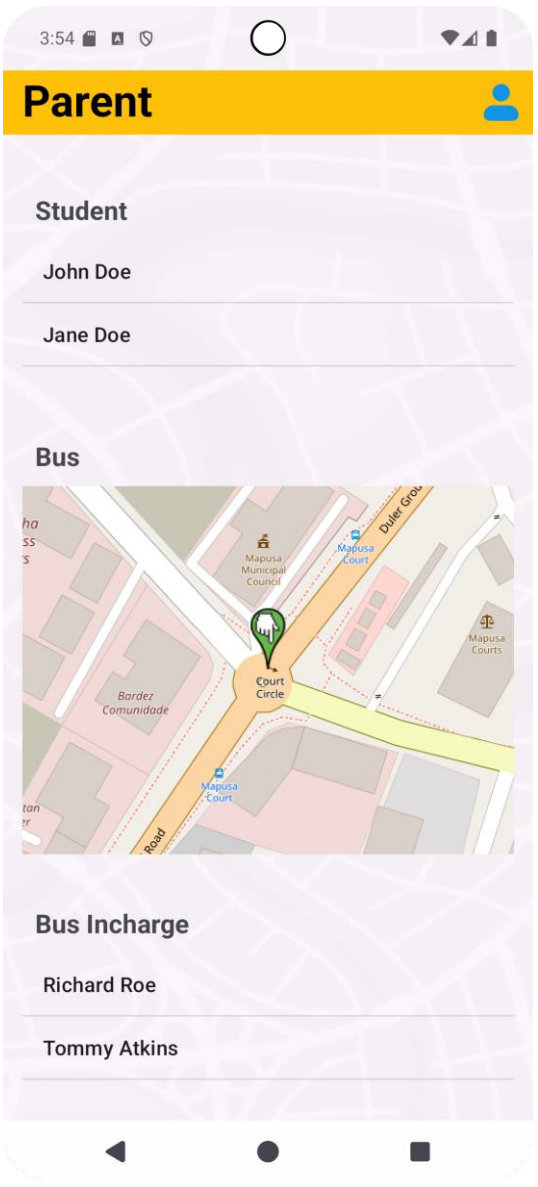
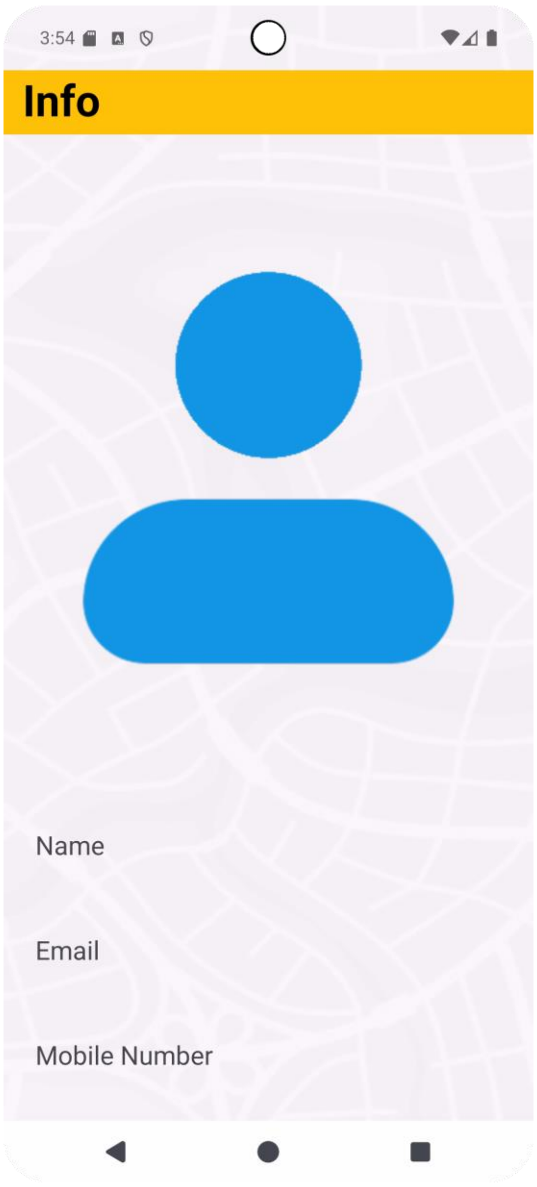
The app strikes a balance between functionality and simplicity, making it easy to navigate while offering powerful features. Whether checking bus locations, receiving alerts, or managing routes, the app ensures users can do everything they need without hassle. The clean design and intuitive navigation make it accessible for people with varying levels of technical knowledge, ensuring that no one feels overwhelmed while using it. With its role-based access and easy-to-understand interface, the app enhances communication, safety, and operational efficiency in the school transportation system.

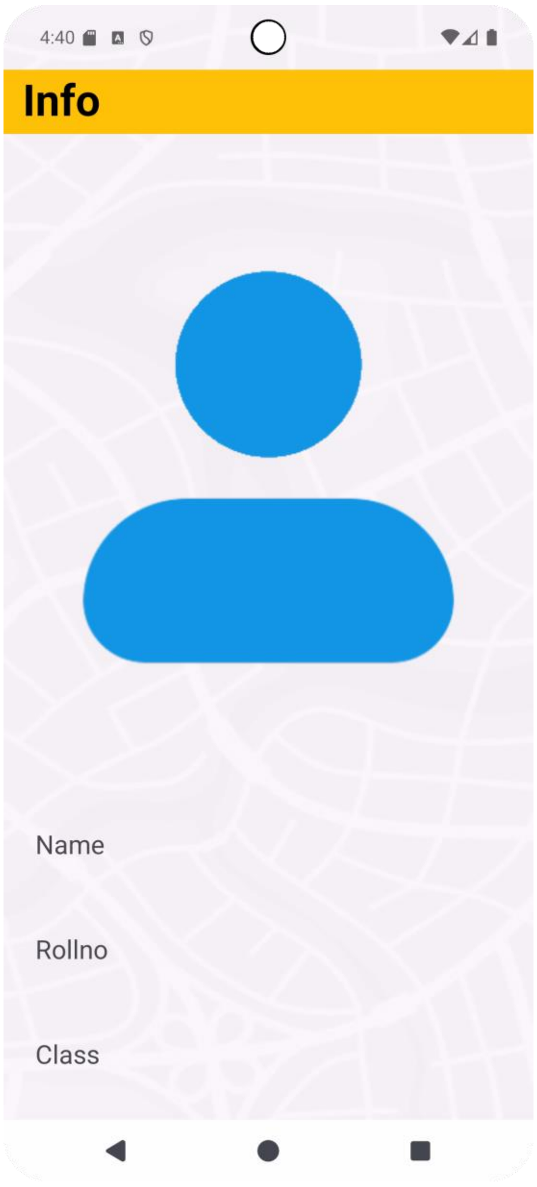
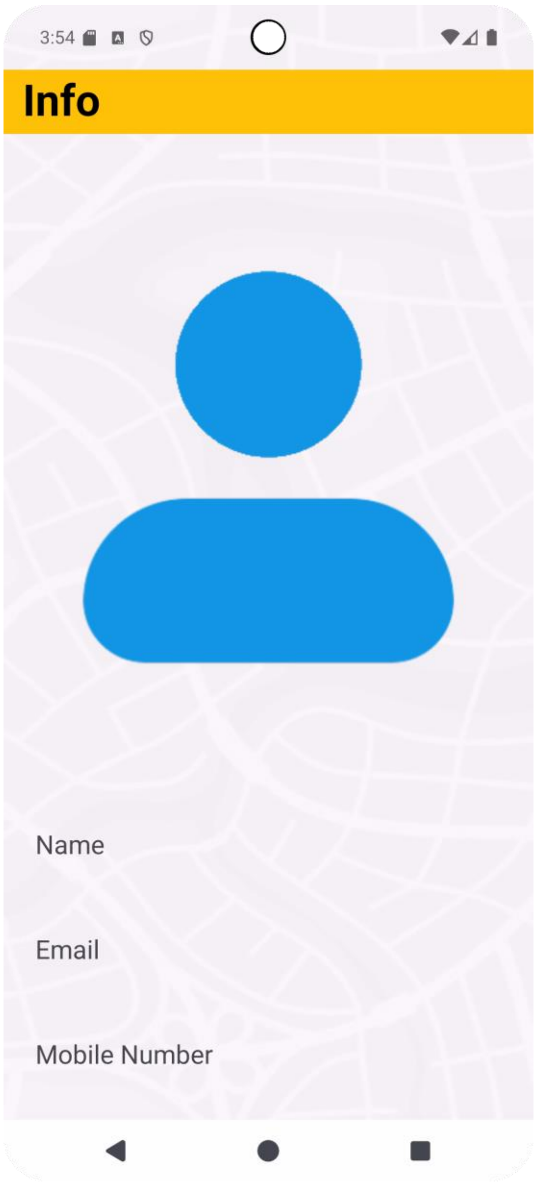
7.1. Sign Up / Login

	 3:53 Sign Up Name Email Mobile Number Password Re-enter Password Show Password User Type Parent Bus Incharge School Authority Sign Up
The start screen of the app allows users to sign up for a new account or sign in into an existing account.	If the user chooses to create a new account, they are provided the standard options to enter their name, email, mobile number and password. The user must choose what type of user they are: Parent, Bus Incharge or School Authority.

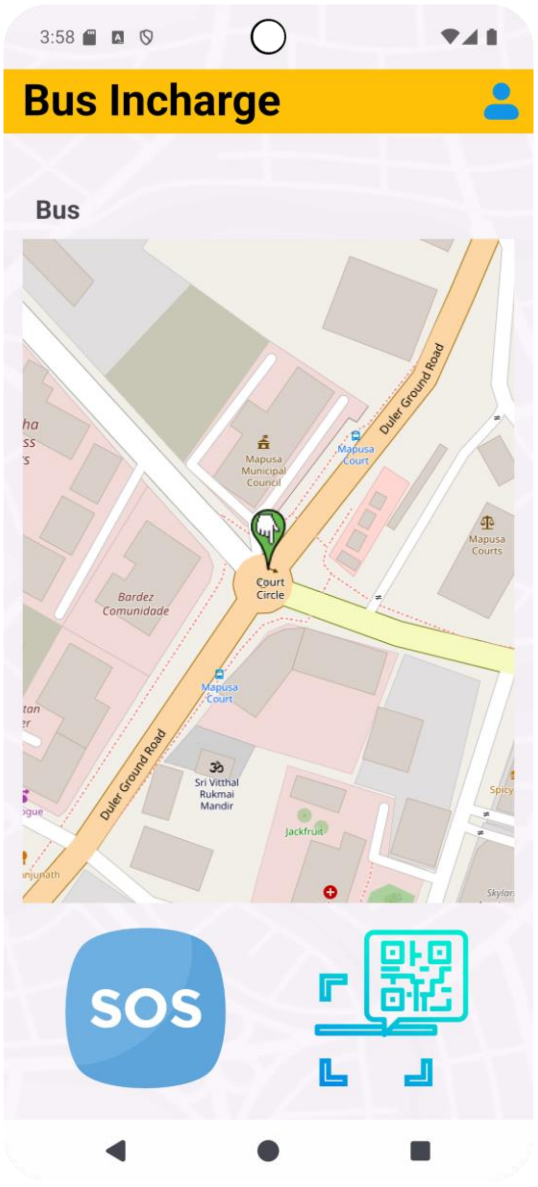
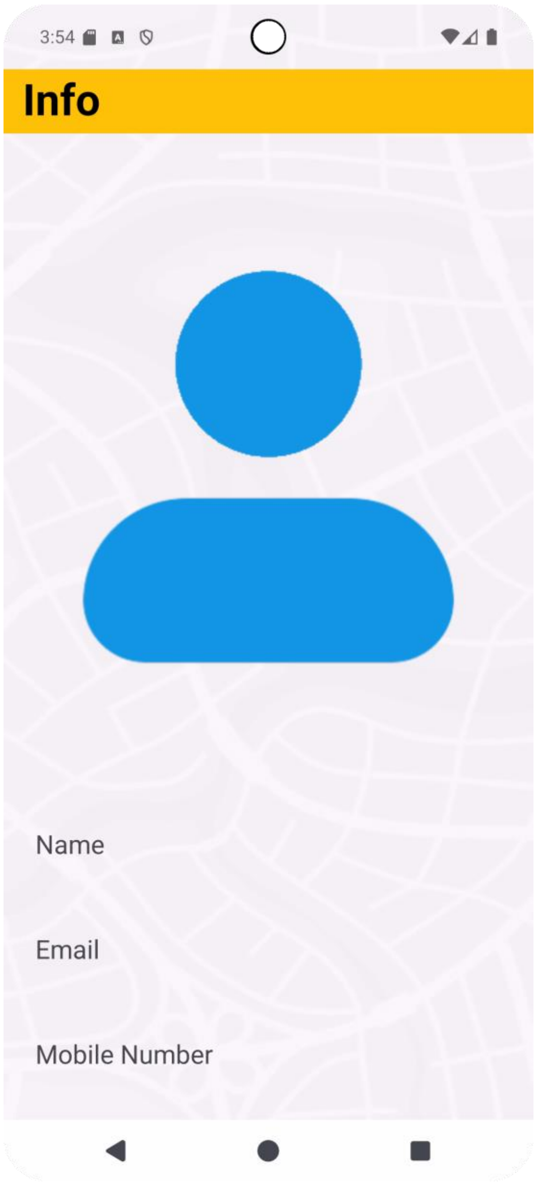
 3:53 User Type Parent Bus Incharge School Authority	 3:53 Sign In Email Password Show Password Sign In
If the user chooses to sign in, they are first given the choice to choose what kind of user they are.	Then user can then enter their credentials, this step is the same for all the 3 types of users.

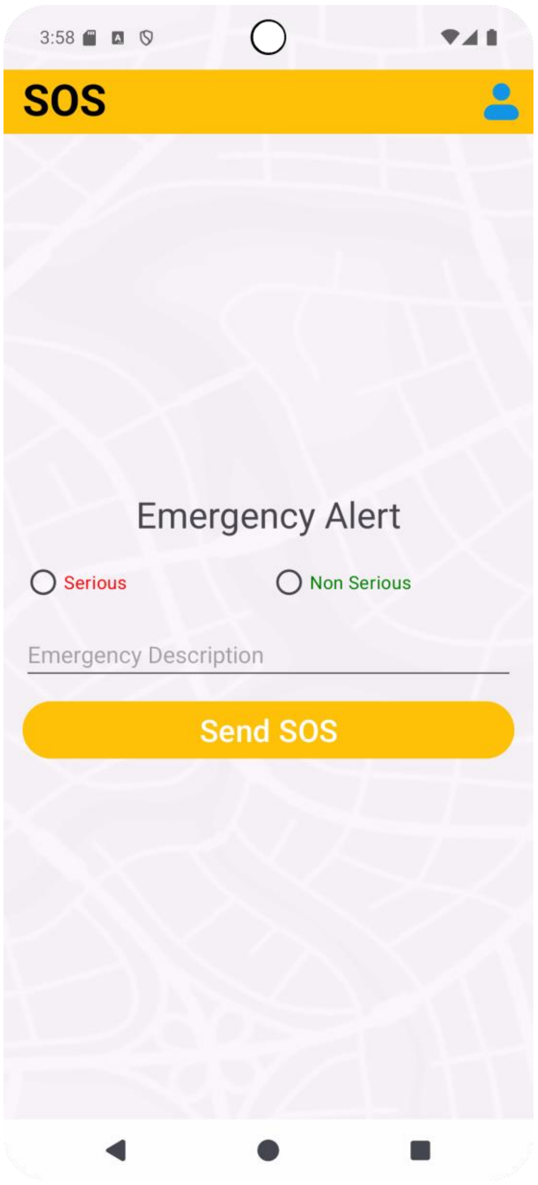
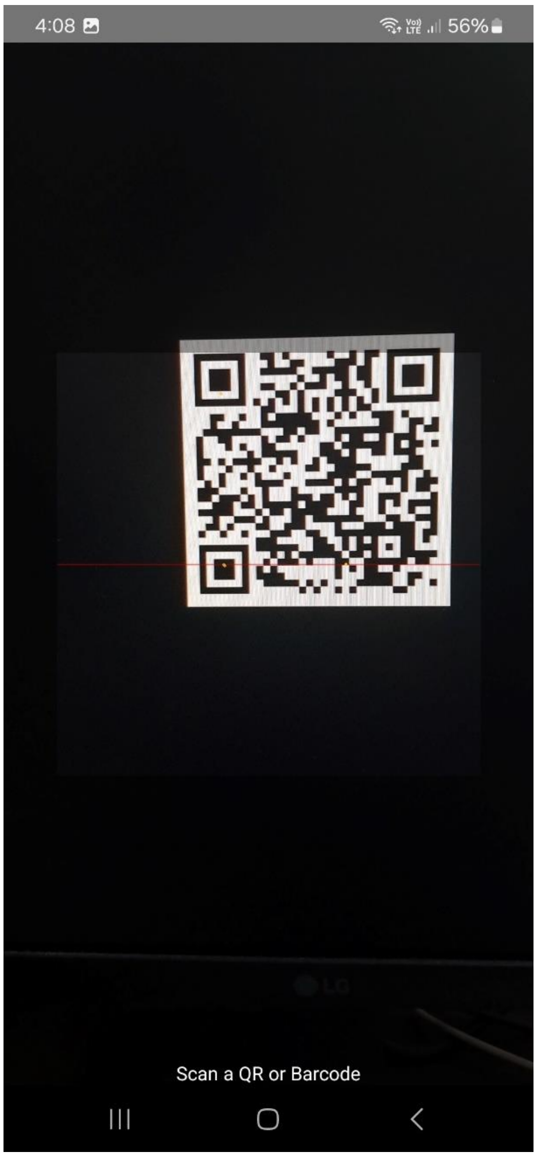
7.2. Parents

 3:54 **Parent** **Student** John Doe Jane Doe **Bus** Mapusa Municipal Council Mapusa Court Duler Grou Mapusa Courts Court Circle Bardez Comunidade Road **Bus Incharge** Richard Roe Tommy Atkins	 3:54 **Info** Name Email Mobile Number
The Parent home screen features a quick over view of all the features available to a Parent	Clicking the user icon at the top right corner of the Parent screen leads you to this screen which has more detailed information about the user using the app.

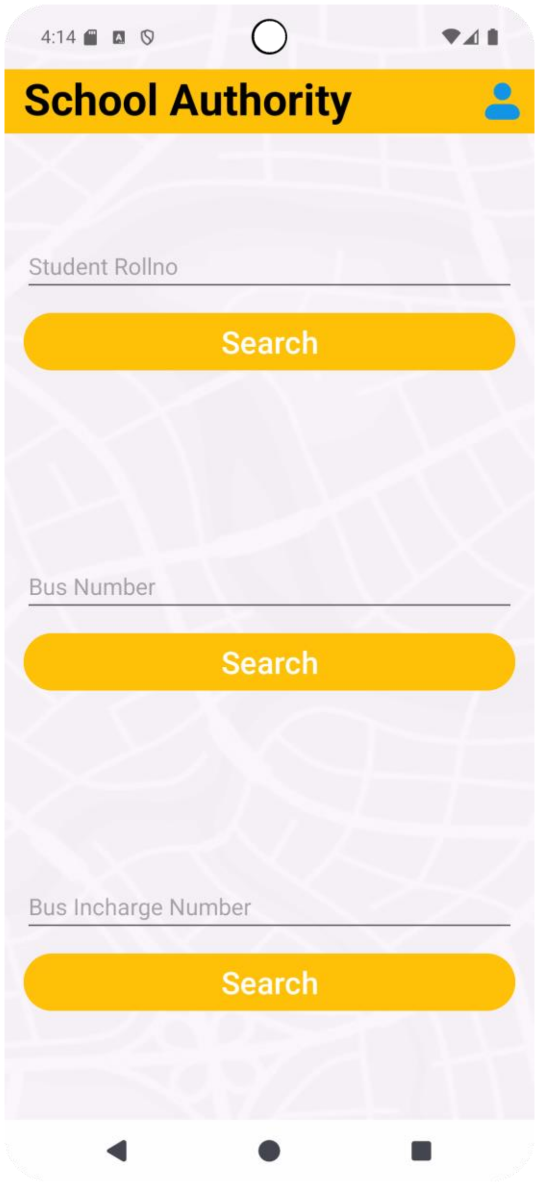
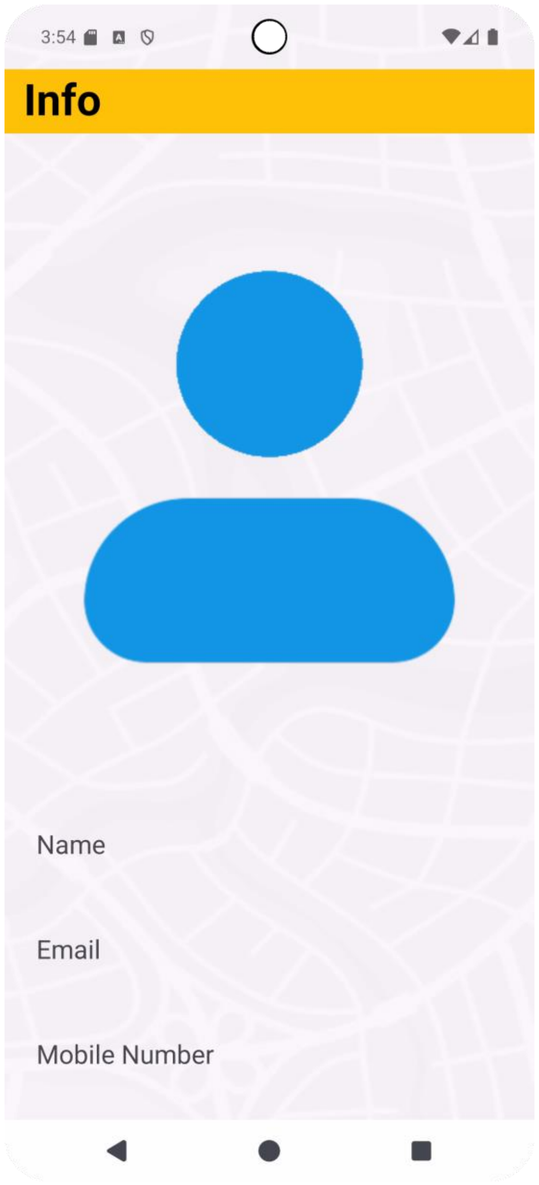
 Name Rollno Class	 Name Email Mobile Number
Clicking on any Student from the Parent screen leads you to this screen which has more detailed information about the Student.	Clicking on any Bus Incharge from the Parent screen leads you to this screen which has more detailed information about the Bus Incharge.

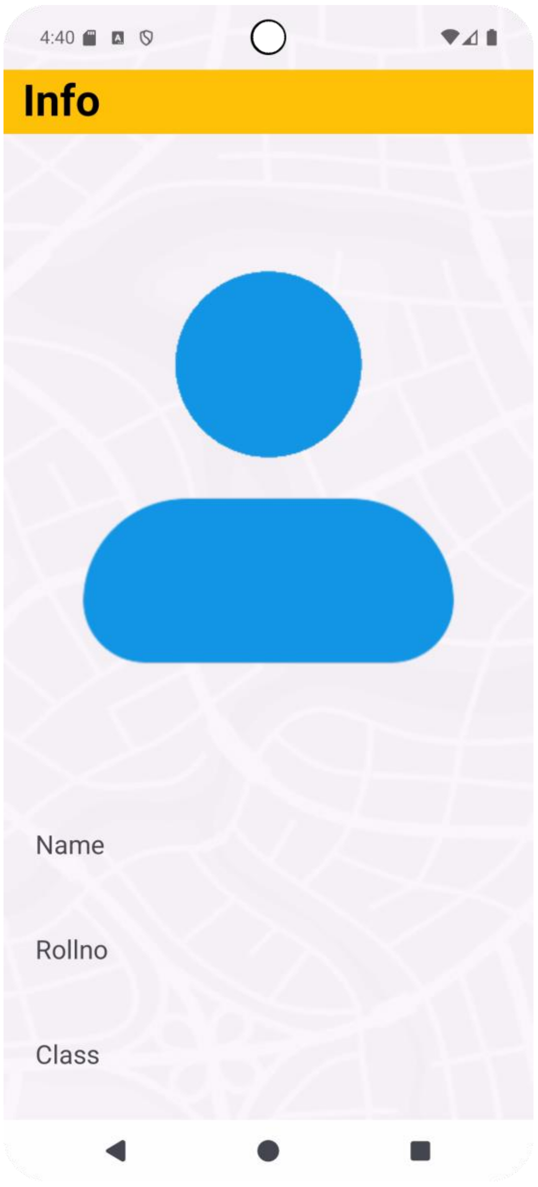
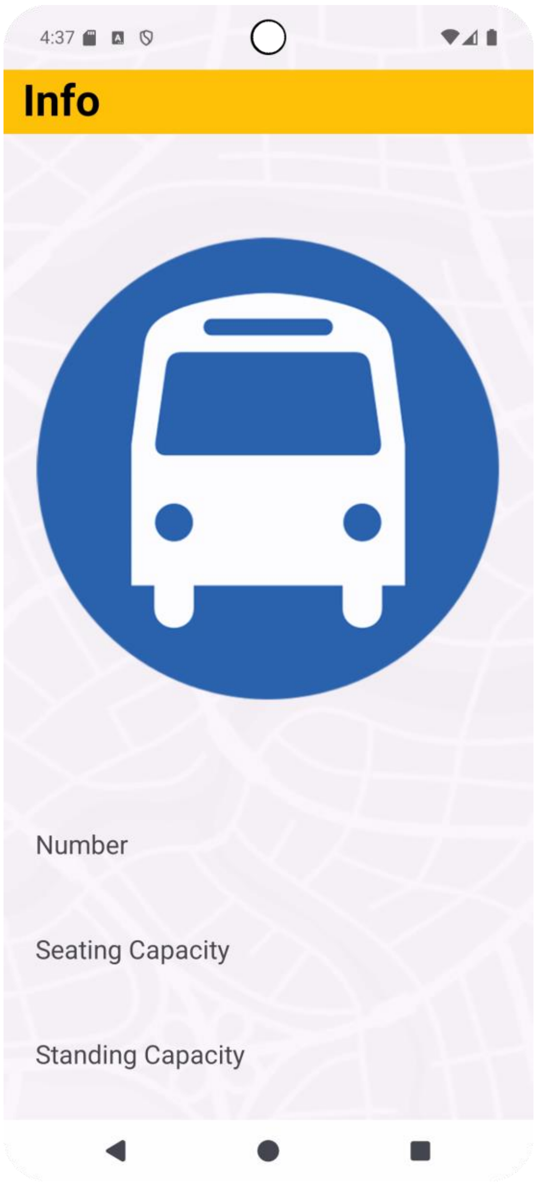
7.3. Bus Incharge

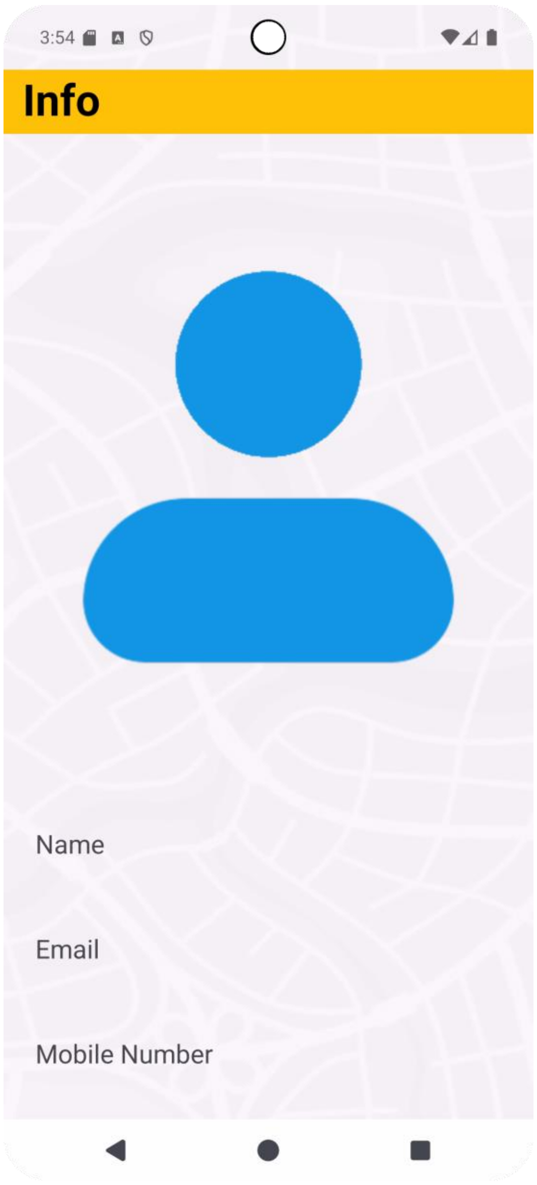
 3:58 Bus Incharge Bus Duler Ground Road Mapusa Municipal Council Mapusa Court Mapusa Courts Court Circle Bardez Comunidade Sri Vitthal Rukmai Mandir Jackfruit SOS	
The Bus Incharge home screen features a quick over view of all the available features available to a Bus Incharge.	Clicking the user icon at the top right corner of the Bus Incharge screen leads you to this screen which has more detailed information about the user using the app.

 SOS Emergency Alert Serious Non Serious Emergency Description Send SOS	 Scan a QR or Barcode
The SOS tab provides the functionality to alert the parents and school authorities if there is a problem. The bus incharge can choose the severity of the problem and add a description.	The Scan tab starts the camera and begins scanning the identification cards of the students boarding the bus. This is how the parents and school authorities get the attendance of Students on the buses.

7.4. School Authority Tab

 4:14 School Authority Student Rollno Search Bus Number Search Bus Incharge Number Search	
The School Authority home screen features a quick over view of all the available features available to a School Authority	Clicking the user icon at the top right corner of the School Authority screen leads you to this screen which has more detailed information about the user using the app.

	 4:37 Info Number Seating Capacity Standing Capacity
Searching any Student Rollno on the previous screen leads you to this screen which has more detailed information about the Student.	Searching any Bus Number on the previous screen leads you to this screen which has more detailed information about the Bus.

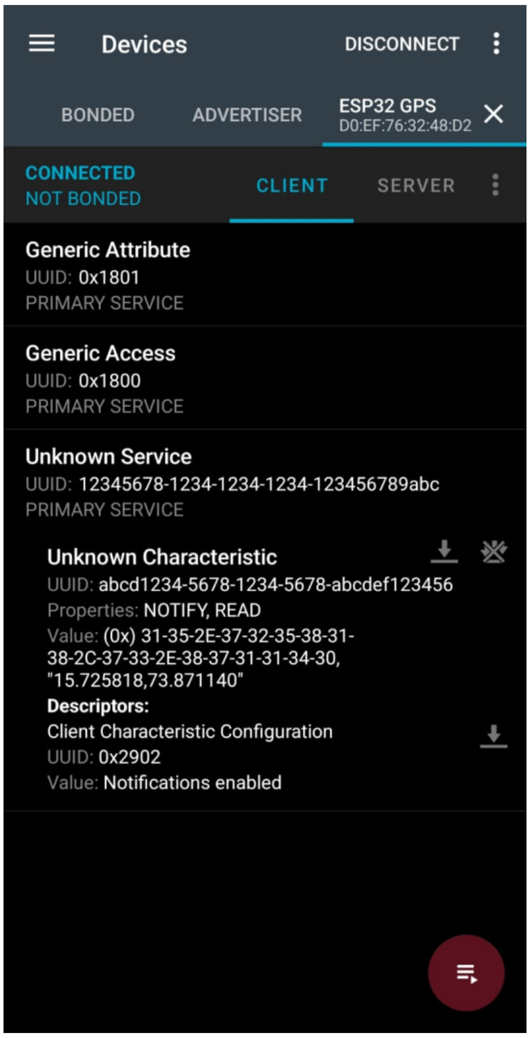


Searching for Bus Incharge(driver, helper) name on the previous screen leads you to this screen which has more detailed information about them.

8. Testing

The GPS module was tested by placing it just behind the windshield and its power cable was connected to pendrive port of the music entertainment system. This ensured the best signal quality while ensuring easy access.

To connect to the module, we used the app nRF Connect. As soon as the module acquired a GPS signal it began sending the coordinates over BLE which were captured by the app. Once subscribed the coordinates could be continuously captured and accessed through the log section of the app.

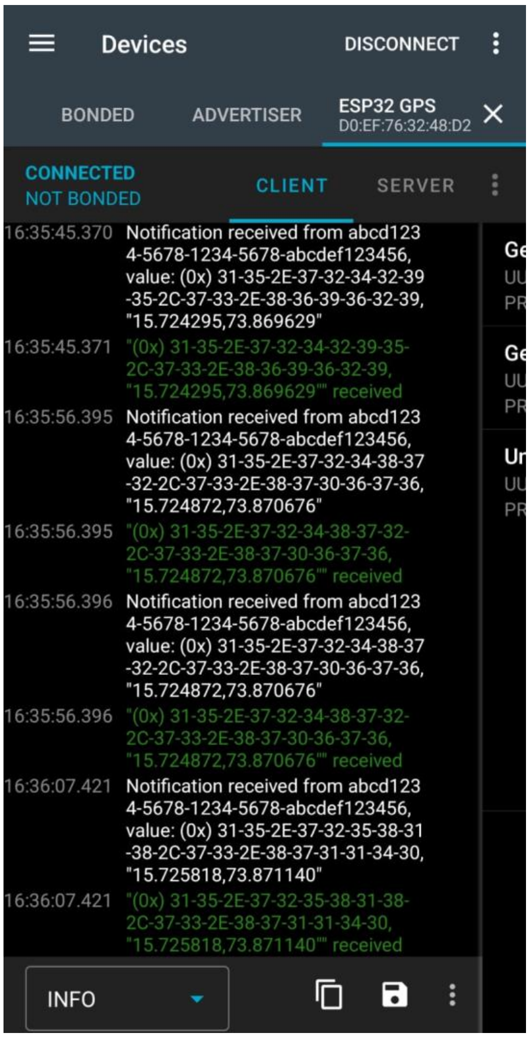


Generic Attribute
UUID: 0x1801
PRIMARY SERVICE

Generic Access
UUID: 0x1800
PRIMARY SERVICE

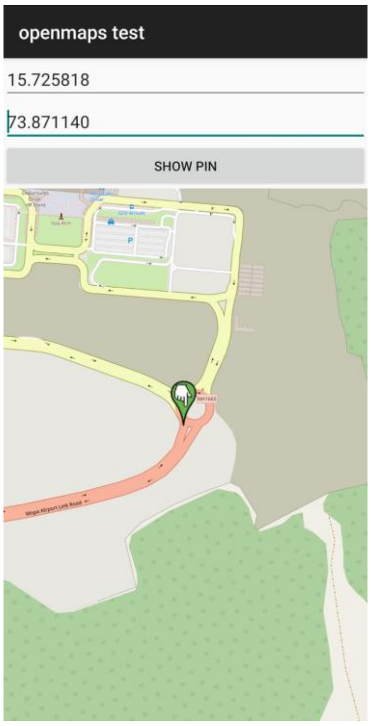
Unknown Service
UUID: 12345678-1234-1234-1234-123456789abc
PRIMARY SERVICE

Unknown Characteristic
UUID: abcd1234-5678-1234-5678-abcdef123456
Properties: NOTIFY, READ
Value: (0x) 31-35-2E-37-32-35-38-31-38-2C-37-33-2E-38-37-31-31-34-30, "15.725818,73.871140"
Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Value: Notifications enabled



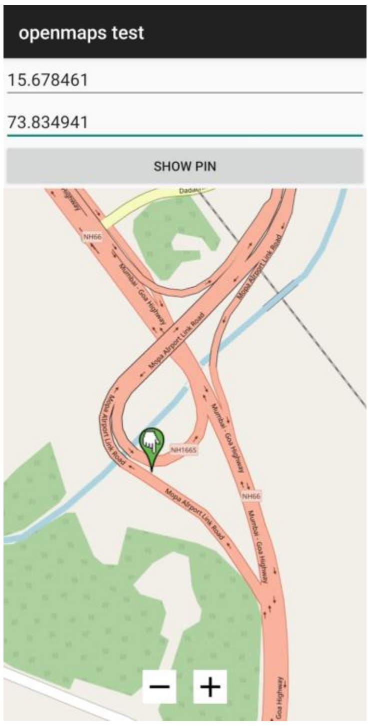
```
16:35:45.370 Notification received from abcd123
             4-5678-1234-5678-abcdef123456,
             value: (0x) 31-35-2E-37-32-34-32-39
             -35-2C-37-33-2E-38-36-39-36-32-39,
             "15.724295,73.869629"
16:35:45.371 "(0x) 31-35-2E-37-32-34-32-39-35-
             2C-37-33-2E-38-36-39-36-32-39,
             "15.724295,73.869629"" received
16:35:56.395 Notification received from abcd123
             4-5678-1234-5678-abcdef123456,
             value: (0x) 31-35-2E-37-32-34-38-37
             -32-2C-37-33-2E-38-37-30-36-37-36,
             "15.724872,73.870676"
16:35:56.395 "(0x) 31-35-2E-37-32-34-38-37-32-
             2C-37-33-2E-38-37-30-36-37-36,
             "15.724872,73.870676"" received
16:35:56.396 Notification received from abcd123
             4-5678-1234-5678-abcdef123456,
             value: (0x) 31-35-2E-37-32-34-38-37
             -32-2C-37-33-2E-38-37-30-36-37-36,
             "15.724872,73.870676"
16:35:56.396 "(0x) 31-35-2E-37-32-34-38-37-32-
             2C-37-33-2E-38-37-30-36-37-36,
             "15.724872,73.870676"" received
16:36:07.421 Notification received from abcd123
             4-5678-1234-5678-abcdef123456,
             value: (0x) 31-35-2E-37-32-35-38-31
             -38-2C-37-33-2E-38-37-31-31-34-30,
             "15.725818,73.871140"
16:36:07.421 "(0x) 31-35-2E-37-32-35-38-31-38-
             2C-37-33-2E-38-37-31-31-34-30,
             "15.725818,73.871140"" received
```

To visualize the coordinates on a map we used a custom app that accepted coordinates and displayed the pins on the map.


openmaps test

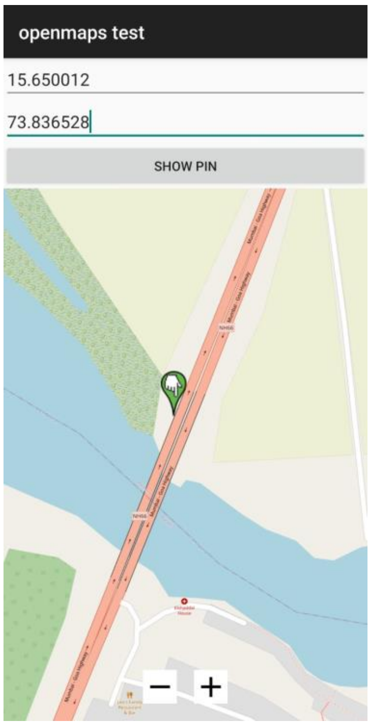
15.725818

73.871140


openmaps test

15.678461

73.834941


openmaps test

15.650012

73.836528

9. Benefits

- Real-Time Tracking: Parents and school authorities can monitor the bus's live location, reducing waiting times and enhancing safety.
- Accurate Attendance Management: QR code scanning ensures precise student attendance records, minimizing proxy attendance and administrative workload.
- Cost-Effective Hardware: The use of affordable components (ESP32, NEO-6M) and open-source software (Arduino IDE, PostgreSQL) lowers implementation costs.
- Multi-User Accessibility: Role-based interfaces cater to parents, school authorities, and bus operators, ensuring relevant data is accessible to each stakeholder.
- Enhanced Safety Features: The SOS button enables immediate alerts during emergencies, while real-time notifications keep parents informed about their child's status.
- Scalable Database: The PostgreSQL database efficiently manages student, bus, and attendance data, supporting scalability for larger institutions.
- Lightweight and Durable Design: The 3D-printed case ensures the hardware is compact, weather-resistant, and easy to install in buses.

10. Conclusion

In conclusion our service enhances the safety and efficiency of school transportation. It provides real-time tracking of school buses and ensures accurate student attendance monitoring. By integrating hardware, software and cloud-based technologies, the system aims to address the common challenges faced by schools, parents and school transport authorities.

Our service at its core is about location tracking, which is made possible through the NEO 6M GPS module using the ESP32 to interface with the GPS module. The ESP32 with its vast array of libraries allow quick parsing of the GPS data to get the most relevant information, the coordinates. The Bluetooth functionality of the ESP32 is then used to send the data to the mobile app.

Upon launching the app, users are guided through permission prompts, ensuring that the app can provide its full suite of services. The mobile app is versatile, allowing for the parents, school authorities and bus operators to easily access the information concerning them.

The parents can get information like the attendance of their children on the bus and routes the bus follows, the parents can also get information about the bus incharge(driver and helper).

The school authorities get information about all the buses and routes their school provides along with the respective bus incharge. The school authorities also get information of the students availing the bus facility.

The bus operators on the other hand can use the app to scan the identity cards of the students when they are boarding the bus, ensuring their attendance is logged. This attendance data along with the GPS data is sent to the server so that the parents and school authorities can benefit for it.

Our server is equipped with a simple yet strong database that handles the student, parent, GPS, routes and bus incharge data which ensures quick access for anyone using our service.

11. Limitations

- GPS Signal Sensitivity: The NEO-6M GPS module's antenna requires placement near the windshield for optimal signal reception. In older buses or vehicles with obstructed views, signal acquisition may be slower or less accurate.
- Route Optimization: The system currently lacks automated route optimization algorithms. Integrating such features would allow the system to adapt to traffic conditions and suggest more efficient routes to bus drivers.
- Automated Emergency Services: As the app relies on manual SOS alerts, incorporating automated emergency services (e.g., direct alerts to hospitals or police) could significantly enhance overall safety.

12. Scope For Further Development

- Custom Board: using a NEO 6M board with an ESP32 takes more space which brings up the need for a bulky 3d printed case with a complex design in order to house each board in its own compartment. Using a custom designed board will mitigate these problems by integrating the features of both these boards into one.
- Better Antenna: the antenna that's provided with the NEO 6M board is suitable only for prototyping and not for professional use, substituting this antenna with a bigger and stronger antenna will allow the module to be placed anywhere without the need for it to be placed near the wind shield.
- Battery operated: the ESP32 is powered using a USB cable connected to the pendrive port on the music entertainment system of the bus. Some older bus models lack this feature, adding a battery pack to the GPS module will make the module standalone.
- More Feature: the app lacks quality of life features like route optimization and automatic emergency service, in its current state it relies on the bus operators to make these decisions.

13. Bibliography

13.1. Literature Review

[1] Ruturaj Shelake, Reshma Chavan, Raju Rai and Mangesh Manake, "Intelligent Transport System for Real-Time School Bus Tracking for Safety and Security of Children Using GPS," ResearchGate, 2022. [Online]. Available: https://www.researchgate.net/publication/358039360_Intelligent_Transport_System_for_Real_Time_School_Bus_Tracking_For_Safety_and_Security_of_Child_Using_GPS

[2] Moechammad Sarosa, Mentari Tika Putri Ningrum and Putri Elfa Mas'udia, "Design and Implementation of School Bus Information and Tracking System," ResearchGate, 2023. [Online]. Available: https://www.researchgate.net/publication/370430447_Design_and_implementation_of_school_bus_information_and_tracking_system_application

[3] Wendy Mwende Mbabu, "A GPS Tracking System for School Transportation Services," Strathmore University, 2021. [Online]. Available: https://su-plus.strathmore.edu/bitstreams/1783c4e4-2b65-4334-8287-d5357b4b0922/download

[4] Khang Jie Liew and Tee Hean Tan, "QR Code-Based Student Attendance System," IEEE Xplore, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9681472

[5] K Kiran Kumar, Pattan Firoze, K Ramesh Babu and Samarouthu Mounika, "Implementing a QR-Based Attendance System for University Students," Journal of Computing and Digital Research, 2023. [Online]. Available: https://jcdronline.org/admin/Uploads/Files/66192d27b4c493.11311587.pdf

13.2. References

ESP32 Bluetooth Architecture en

https://www.espressif.com/sites/default/files/documentation/esp32_bluetooth_architecture_en.pdf