Transportation Management System for FAST

Rabab Hussain 19I-0893

Mauazzama Aslam 19I-0892

Zayan Safi 19I-0850

Project Supervisor

Dr. Arshad Hassan

Department of Electrical Engineering

National University of Computer and Emerging Sciences, Islamabad

2023

Developer's Submission

"This report is being submitted to the Department of Electrical Engineering of the National University of Computer and Emerging Sciences in partial fulfillment of the requirements for the degree of BS in Electrical Engineering"

Developer's Declaration

"We take full responsibility of the project work conducted during the Final Year Project (FYP) titled **"Transportation Management System for FAST"**. We solemnly declare that the project work presented in the FYP report is done solely by us with no significant help from any other person; however, small help wherever taken is duly acknowledged. We have also written the complete FYP report by ourselves. Moreover, we have not presented this FYP (or substantially similar project work) or any part of the thesis previously to any other degree-awarding institution within Pakistan or abroad.

We understand that the management of the Department of Electrical Engineering of National University of Computer and Emerging Sciences has a zero-tolerance policy towards plagiarism. Therefore, we as an author of the above-mentioned FYP report solemnly declare that no portion of our report has been plagiarized and any material used in the report from other sources is properly referenced. Moreover, the report does not contain any literal citing of more than 70 words (total) even by giving a reference unless we have obtained the written permission of the publisher to do so. Furthermore, the work presented in the report is our own work and we have positively cited the related work of the other projects by clearly differentiating our work from their relevant work.

We further understand that if we are found guilty of any form of plagiarism in our FYP report even after our graduation, the University reserves the right to withdraw our BS degree. Moreover, the University will also have the right to publish our names on its website that keeps a record of the students who committed plagiarism in their FYP reports."

_____	_____	_____
Rabab Hussain	Mauazzama Aslam	Zayan Safi
BS(EE) 2019-0893	BS(EE) 2019-0892	BS(EE) 2019-0850

_____

Certified by Supervisor

_____

Verified by Plagiarism Cell Officer

Dated: _____

Abstract

Commuting is a growing necessity and due to the increasing number of students, it is becoming difficult to manage the university transportation system for administration and students alike. As the drivers and buses are updated daily each student has to individually contact the administration in order to avail the required information. At times students have to wait clueless for an unknown period of time not knowing whether a bus is arriving or not.

This manual coordination is inefficient and hectic for day to day user. Furthermore, the service quality promised by the administration is often not provided by the bus drivers. If not that, the bus driver might be using the bus for personal usage or unsafe driving. And if a stranger enters the bus there is ambiguity in verification of the identity of the commuter.

In order to tackle these problems, the Transportation Management System will entail data logging features using hardware implementation and app development for both users, the students and the admin, to keep track of the services and improve its efficiency.

The project aims to provide the following facilities:

1. Database having a record of students, payment status etc. for easy monitoring.
2. Admins can easily convey information to students instead of notifying each student individually.
3. Real-time bus location, route, eta, and miscellaneous information for passengers.
4. Fuel usage data to administrator.
5. Bus atmosphere monitoring through temperature and humidity sensor.
6. Barcode scanner to verify the commuter is a registered member.

Acknowledgments

We would like to express our gratitude to everyone who contributed to the completion of this project. Their constant input and sincere interest have been the key to achieving this milestone.

First of all, we would like to thank our project supervisor, Dr. Arshad Hassan, for his guidance throughout the project and crucial insights. We want to further extend our gratitude to the university faculty and workforce for accommodating our project needs throughout.

Contents

List of figures

1. Chapter 1 Introduction

1.1 Motivation

A digital survey from active users of the university transportation system was conducted. The result concluded that the majority of the commuters are facing various problems and more than 96% of them wanted a digitized transport management system.

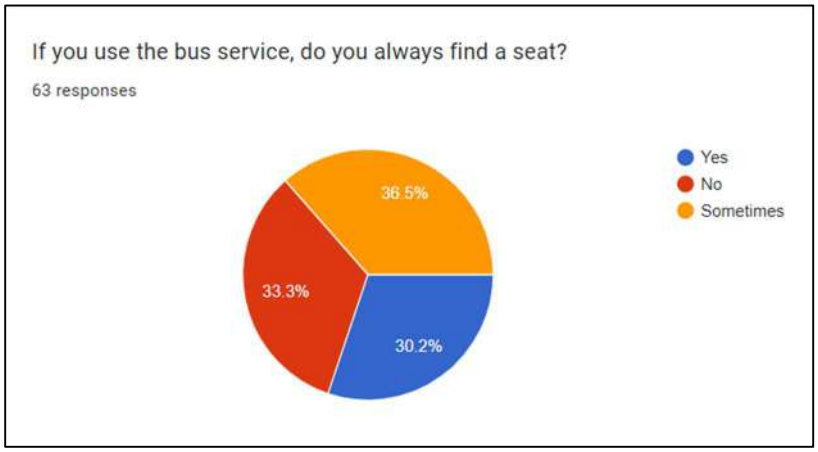


Figure 1 Snippet 1 survey form - Google forms

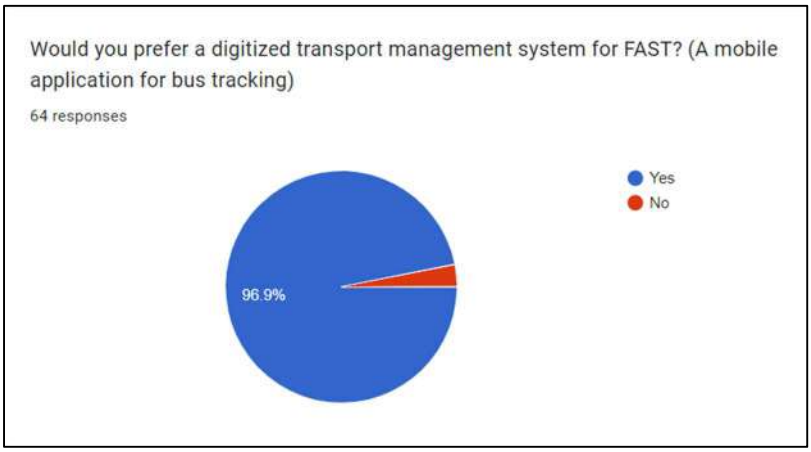


Figure 2 Snippet 2 survey form - Google forms

1.2 Introduction and Background

We have been using the FAST transportation service for the past three years. The experience has been quite hectic at times. The buses follow a pre-planned schedule where the pick-up and drop-off times are allotted, however, there are some variations in these due to external factors such as traffic. In such a scenario, the students have to call the driver, but the drivers are changed regularly. The students then have to contact the main admin, who sometimes cannot be reached due to many students calling in. Similarly, in the evening when the students have to get to their designated buses, the information is placed on a notice board. A large crowd of students gathers around it, and sometimes the information is unclear and needs to be verified by admins

individually. From the admin's perspective, another problem seen is in the verification of transportation fees. Drivers have to individually verify from each passenger whether the charges have been paid, which is both a time-consuming and unorganized method.

To cater to all these issues, our project proposes a smartphone-based application that can be accessed by both admin and commuters. The application will provide the following functionalities:

- Admin
  - Driver and bus allotment
  - Verification of transportation fee
  - Intimation of miscellaneous information to commuters
  - Planning and management of transportation routes
  - Fuel usage
  - Atmosphere conditions on the bus
- Commuters
  - Access information regarding designated bus, driver, route, etc.
  - Real-time bus location
  - Estimated time of arrival notifications
  - Atmosphere conditions on the bus

1.3 Problem Statement

At present, there is no digital management system for FAST transportation. This results in many issues for the students and the administration. Manually coordinating with each student is hectic and inefficient. Moreover, the fuel used by a bus on a single route is not precisely available to the admin. Making the entire process digital where both the student and admin receive real-time accurate information will help in resolving the problems faced.

1.4 Literature Review

The paper [1] introduces the idea of an intelligent transport and fleet management system. This system can help reduce the problems and mismanagement faced by the commuters. By using cost efficient five small modules the system is implemented. The modules are interconnected to each other, and the communication module sends the data remotely to the admin to track and observe the system.

The paper [2] introduces the idea of utilizing Database in the Advanced Transport Management System. The implementation of database is done by using the transport protocol HTTP and XML. The utilization of databases in transport services can make the system more effective. This system can also be extended to include GPS, so the user can track the route.

To regulate the flow of traffic and control traffic congestion, this paper [3] introduces the idea of an Intelligent Public Transportation and City Traffic Management System. This system has CCTV cameras to monitor the traffic flow and share it with the users automatically. The real-time traffic updates are shared to the user on his smartphone.

By using the technology of barcode, the paper [4] introduces the idea of an Information Student Management System. The students can swipe their identity cards on the barcode scanner in order to verify their information. This system is connected to the cloud and can be accessed and updated by the admin user. This system is secure and reduces human effort.

1.5 Project Scope

1.5.1 Project objective

To present software and hardware implemented demonstration of a digitized transportation management system for FAST-ISB by the end of spring 2023.

1.5.2 Technical requirements for the execution:

1. Usage of barcode data and scanners.
2. The admin end application should have access to all the data.
3. ETA and Real-time tracking should be provided to users.
4. Students should have access to the service quality tab.
5. New students should be able to register themselves with ease.

1.5.3 Limits and Exclusions:

1. The project will not be generic to all models of transportation, only for FAST ISB.
2. Online payment of transportation fees will not be included.
3. The project will always require active internet access.

1.6 Report Outline

This report contains multiple chapters, chapter 2 discusses the design methodology of the proposed solution and chapter 3 includes progress and recommendation for future enhancements.

2. Chapter 2 Design methodology, Scope, and Design Solution

The hardware and software design of the whole project is discussed in this section.

Section 2.1 covers the three main modules of the project. Database creation is discussed in Section 2.2. Section 2.3 covers the entire hardware setup of the project. The application is discussed in Section 2.4. The block diagram is covered in Section 2.5 followed by the flowcharts in Section 2.6. In the last section, 2.7 the design, experimentation and data collection are discussed.

2.1 Methodology:

Three main modules were required to design in this project:

- Database creation and implementation.
- Hardware design of the barcode scanner circuit and the sensor's circuit.
- Mobile application development for students and the administration.

2.2 Database

To implement a cloud-based real-time database for storing information of students and drivers and for ease of remote access, we have used Google Firebase. This Real-time Database allows users to build well defined, collaborative applications by providing client-side code accessing the database directly.

2.3 Hardware Setup

The hardware setup was divided into two parts:

- To implement the scanning of user ID cards (Barcodes), a barcode scanner implemented with a Wi-Fi module and Arduino was used.
- Multiple sensors were interfaced with ESP-32 to monitor the bus environment, track the bus location and count the passengers boarding in.

The setup is to be installed in the buses and is programmed to communicate with the database.

2.4 Application

The user-end application for both the fleet admin and commuters is designed using dart (an application development language). The platforms used are Visual Studio Code, Flutter, and Android studio. This application has a secure connection to the real-time database and utilizes the data available there.

2.5 Block diagram

The diagram below provides a concept of how each component will interact in the designed system.

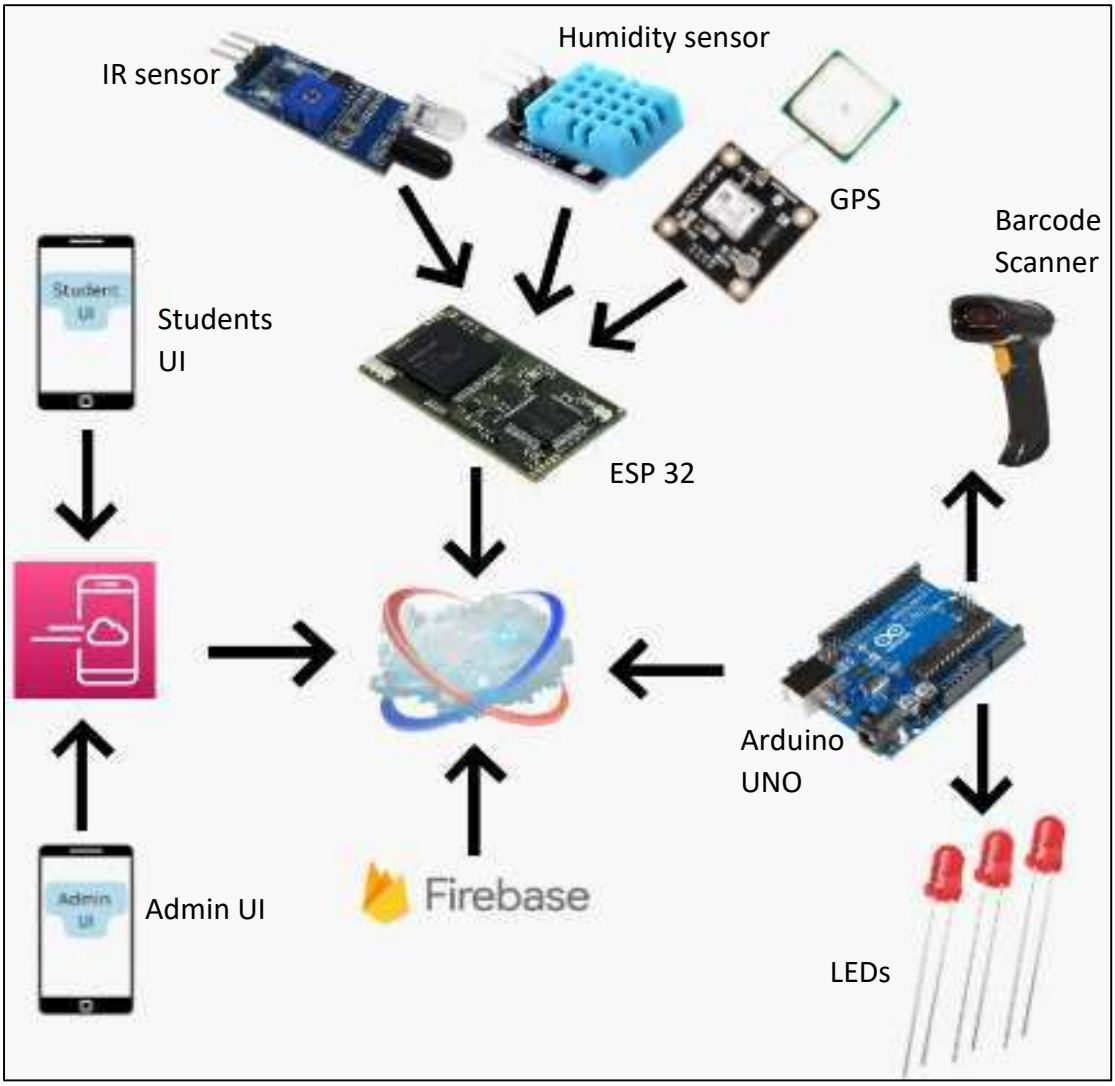


Figure 3 Block diagram of the system

The hardware components used are:

Arduino UNO:

Arduino UNO has been used as a microcontroller board. It plays an important role in determining the bus fee status of the students. As represented in the block diagram, the barcode scanner and the LEDs indicating the fee status are interfaced with Arduino.

LCD 16X2:

To display the fee status and the GPS location coordinates, LCD 16X2 is used. This LCD displays 16 characters per line, it utilizes two lines.

ESP-32:

ESP-32 is used as a WiFi module. It plays a significant part in sending the data from the microcontroller to the cloud-based database. All the four sensors; DHT11, Ultrasonic sensor, IR sensor and GPS sensor are interfaced with the ESP-32.

Barcode Scanner:

A Speed-X, 8500 2D Wire CMOS Handheld Barcode Scanner is used. The barcode scanner takes the student roll number as an input to display the fee status.

Arduino USB Host Shield Module:

Arduino USB Host Shield Module allows the connection of the barcode scanner with the microcontroller.

I2C Module:

To make the display easier, the I2C module is used. It is interfaced with the LCD 16X2 module.

LEDs:

The LEDs emit green and red light when current passes through them. They are connected to the barcode circuit.

GPS:

NEO-6M GPS module is used. It allows the real-time status monitoring of the buses.

Ultrasonic Sensor:

An ultrasonic sensor is used as a fuel level measurement sensor. It measures the distance of the current fuel with the threshold set and helps to efficiently detect the available fuel.

DHT-11 Temperature and Humidity Sensor:

DHT-11 Temperature and Humidity Sensor is used to monitor the bus temperature conditions.

IR Sensor:

IR Sensor is utilized to count the passengers entering the bus.

The software end utilizes as follows:

Firebase

Google Firebase services are used to create a real-time database where all the data from sensors and users is stored. The data is also retrieved as per requirement.

Android application

An application built using flutter libraries and VS code. Coded in dart, the application is accessible to both admin and the commuters.

2.6 Flow chart

The first flowchart is of the student's application user interface. The student has multiple options. The students can:

- Check their fee status.
- Turn on or off the location alerts.
- Enable or disable the location.

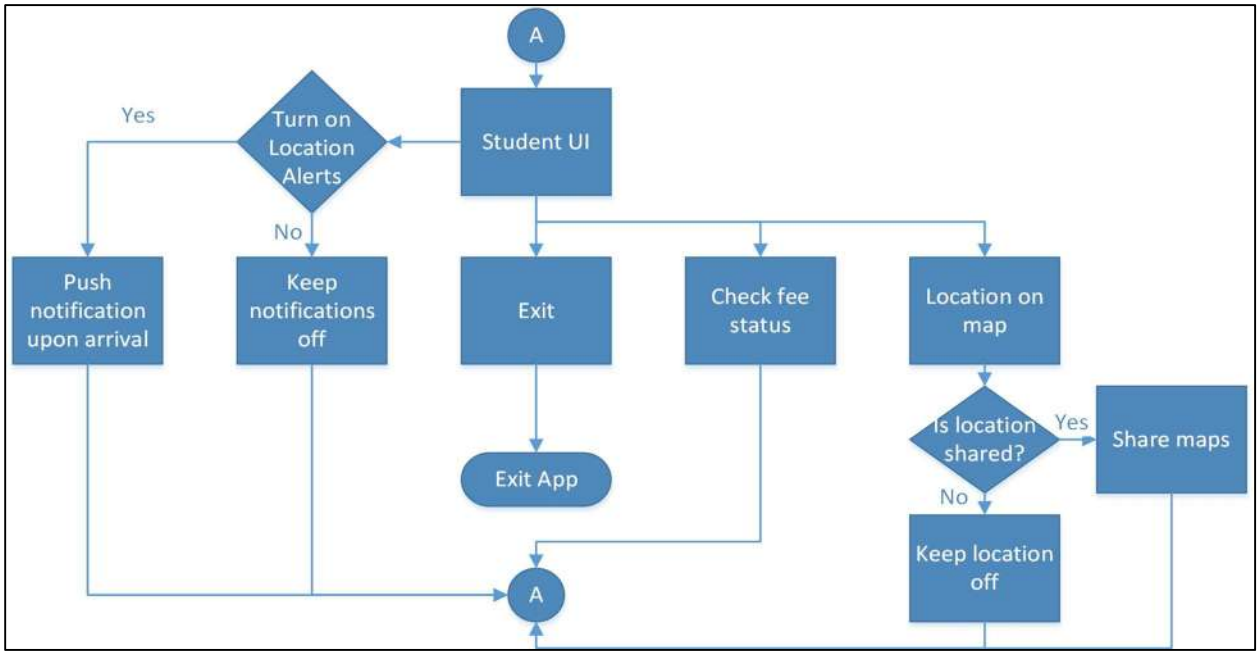


Figure 4 Student interface flow chart

The second flowchart is of the Administration's application interface. The bus administration has similarly multiple options as well:

- The admin can access and edit the driver's data.
- The admin can access and edit the student's data.
- The admin can access and edit the bus data.

The third flowchart above is of the barcode scanner interfacing. The barcode scanner takes the roll number of the student has an input. A check from the database is made whether the student has paid their dues or not, correspondingly a Green or Red LED glows; green indicating paid fee status and red indicating unpaid fee status.

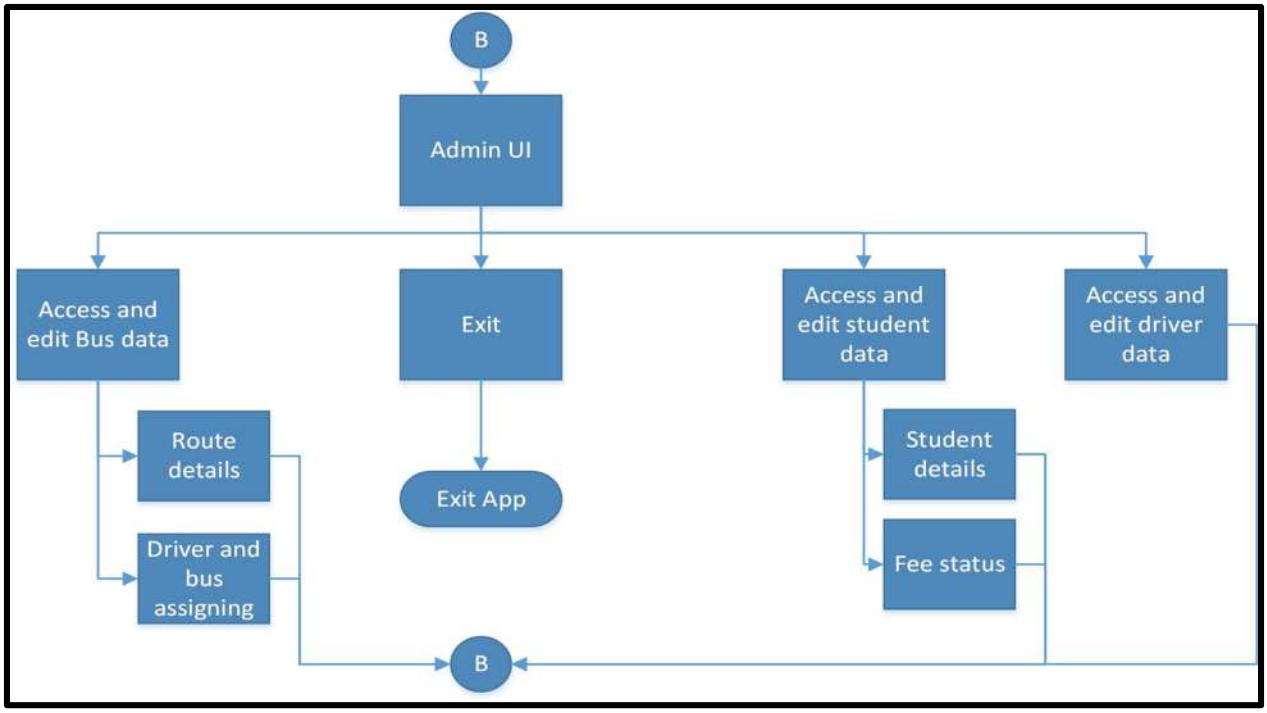


Figure 5 Administrator interface flow chart

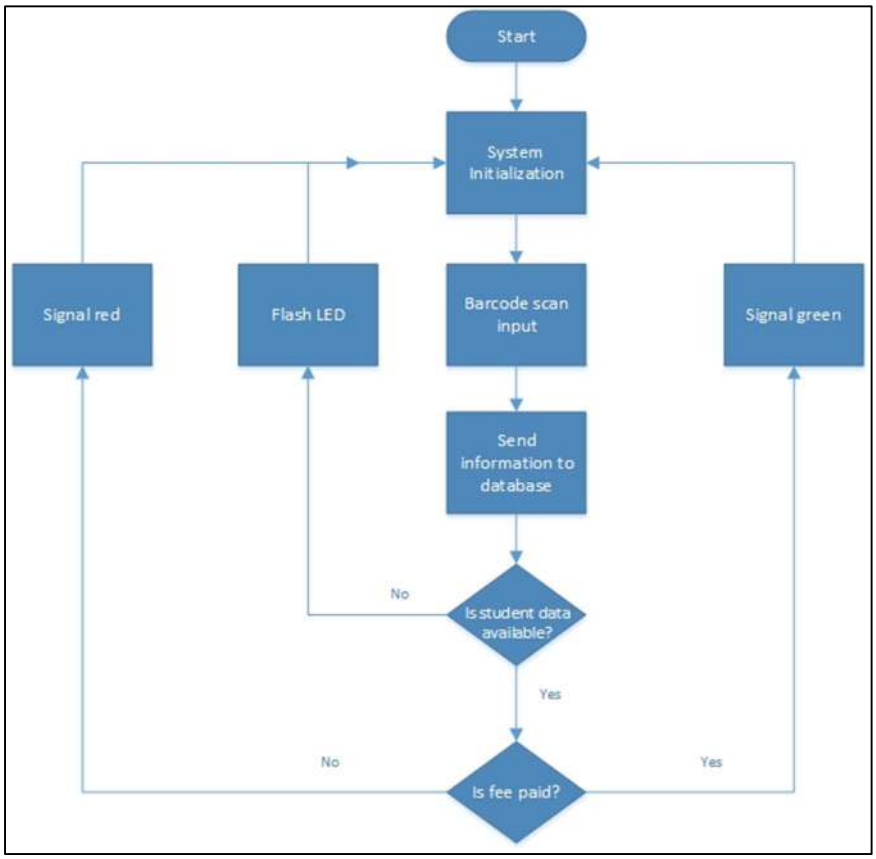


Figure 6 Barcode scanner interfacing flow chart

2.7 Design, Conduct Experiment, and Collect Data:

2.7.1 Database Implementation

To implement the database, the data was divided as follows:

- Students Data: The data collected was the Email, Name, Mobile number, Roll number, Route, and the fee status.
- Drivers Data: The data collected was the Driver's availability status, Name, Bus Number, Phone number and route.

The data was then stored in Google Firebase.

2.7.2 Database Design:

The below figure shows the entry of the student data in the database.

Figure 7 A snippet of Students's data from Google Firebase dashboard

Similarly, the data of more than 50 students has been saved.

The next figure shows the information of the driver's stored in the database.

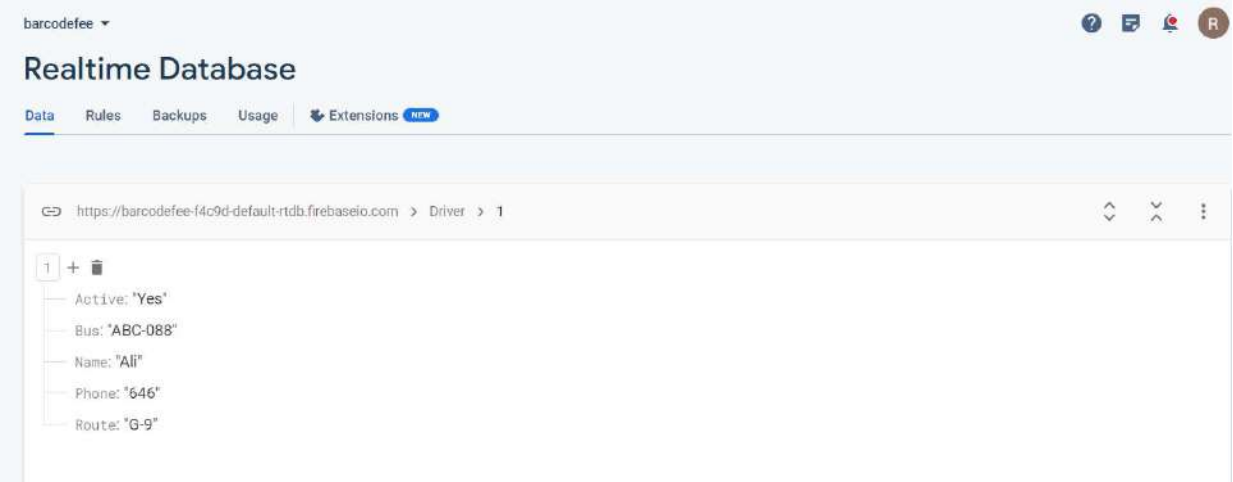


Figure 8 A snippet of Driver's data from Google Firebase dashboard

2.7.3 Database Schema:

The database schema of the students and the drivers is discussed. This is a way to organize the data in separate entities to make it easier to share a single schema within another database

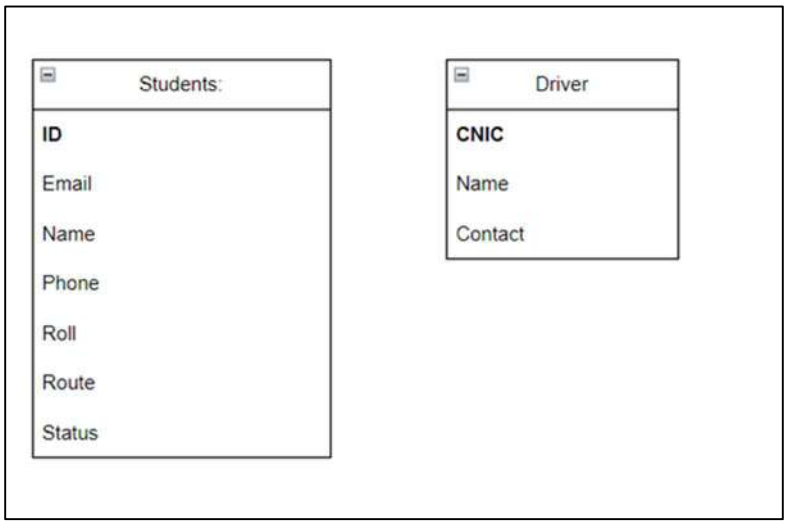


Figure 9 Schema design for the database

2.7.4 Database Entity Relationship Diagram:

This section discusses the database entity relationship diagram. Here the student, bus, driver, and admin are different entities and then all these entities have their respective attributes. Similarly, the relationships between different entities can be seen by the relationship box.

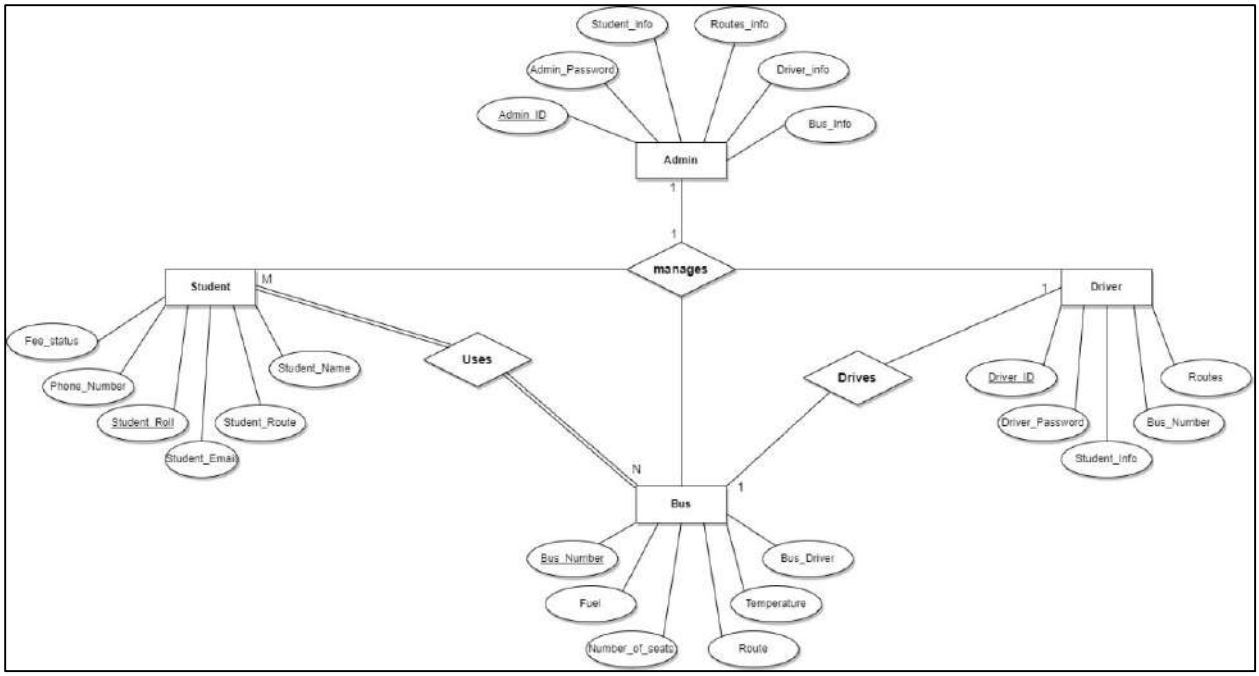


Figure 10 The Entity Relationship Diagram for designed for the management system

2.7.5 Interfacing Barcode Scanner with Arduino:

To interface the barcode scanner with the Arduino, Arduino USB host shield module was used. We sent the scanned data; roll number of the student to the serial monitor of Arduino via the USB Host Shield Rx (receiver) and Tx (transmitter) pins. This data was further sent to the cloud via ESP-8266.

2.7.6 Integrating Arduino, ESP-8266, Barcode Scanner, and Firebase:

Firstly, the barcode scanner scans the data, that is the roll number of the student. Secondly, the data is sent to the serial monitor via the USB Host Shield Rx and Tx pins. Furthermore, the data is sent to the firebase by using ESP-8266.

The code checks the “Feestatus” column in the student’s table. If the status is paid, the LCD displays “Paid”, correspondingly, the Green LED glows.

Similarly, when the fee status is not paid, LCD displays “Not Paid” and “Pending”. Correspondingly, Red LED glows.

2.7.7 Integrating GPS, DHT11, IR and Ultrasonic Sensor with ESP-32

The sensors used in the project were interfaced with the ESP-32. The sensors included:

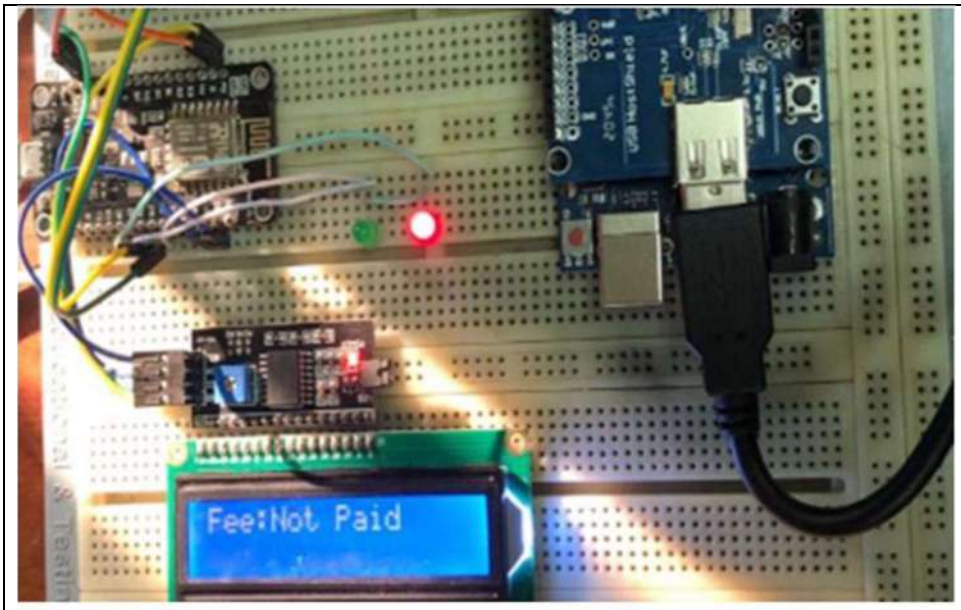
- DHT11; to monitor the temperature and humidity of the bus.
- IR Sensor; to count the passengers boarding in.
- Neo-6M GPS Sensor; to track the bus location.

- Ultrasonic Sensor; to track the fuel level measurement in the bus.

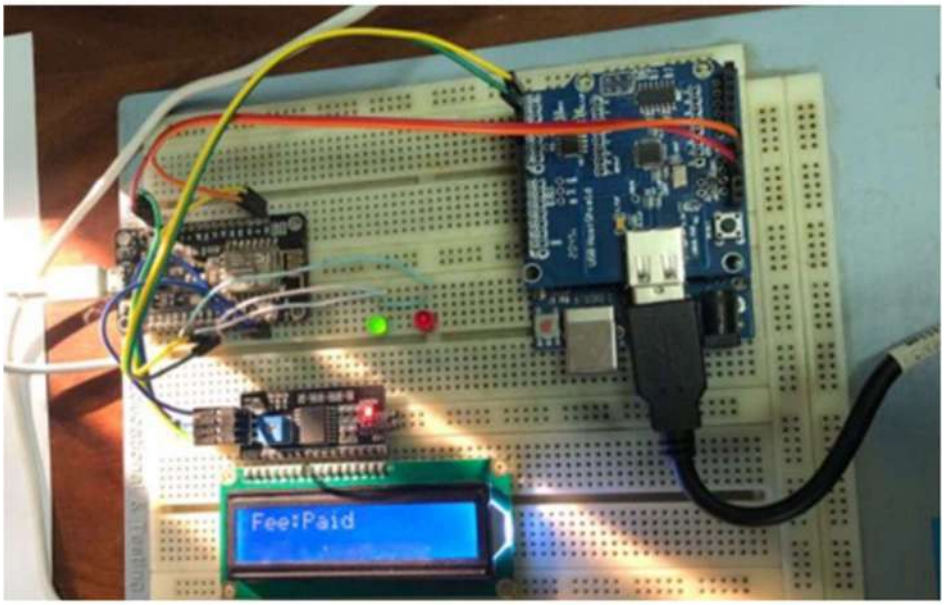
All the sensors were interfaced with ESP-32. The sensors reading were first read on the Arduino serial monitor and via the same ESP-32 were sent to the Google Firebase. From the Google Firebase, the application fetches the data.

2.7.8 Hardware Design of the Barcode Scanner:

In the below figure, the interfacing of the barcode scanner with Arduino via the USB host shield module can be seen. The LCD 16X2 has been interfaced with the I2C module. The LEDS are connected to the pins of ESP-8266 declared as the output pins.



Red LED glows when the student's fee status is not paid.



Green LED glows when the student's fee status is paid.

Figure 11 LCD, ESP, Arduino, Arduino USB host shield and LEDs. Hardware set up captured.

2.7.9 Hardware Schematic of the Sensor's Circuit:

The hardware schematic diagram of the sensor's circuit was made on Fritzing software.

From left to right in the schematic, the first sensor is DHT11. The second sensor is the IR sensor. Above ESP32, an ultrasonic sensor is connected and lastly below the LCD, GPS sensor is attached.

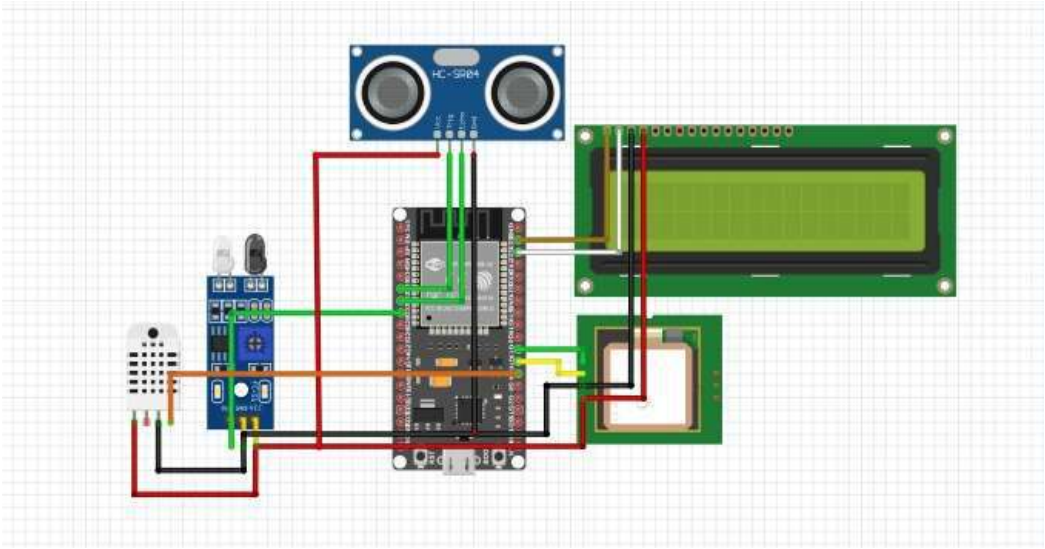


Figure 12 Hardware Schematic Diagram of Sensor's Circuit

2.7.10 Hardware Design of the Sensor's Circuit:

The hardware design of the sensor's circuit is shown below.

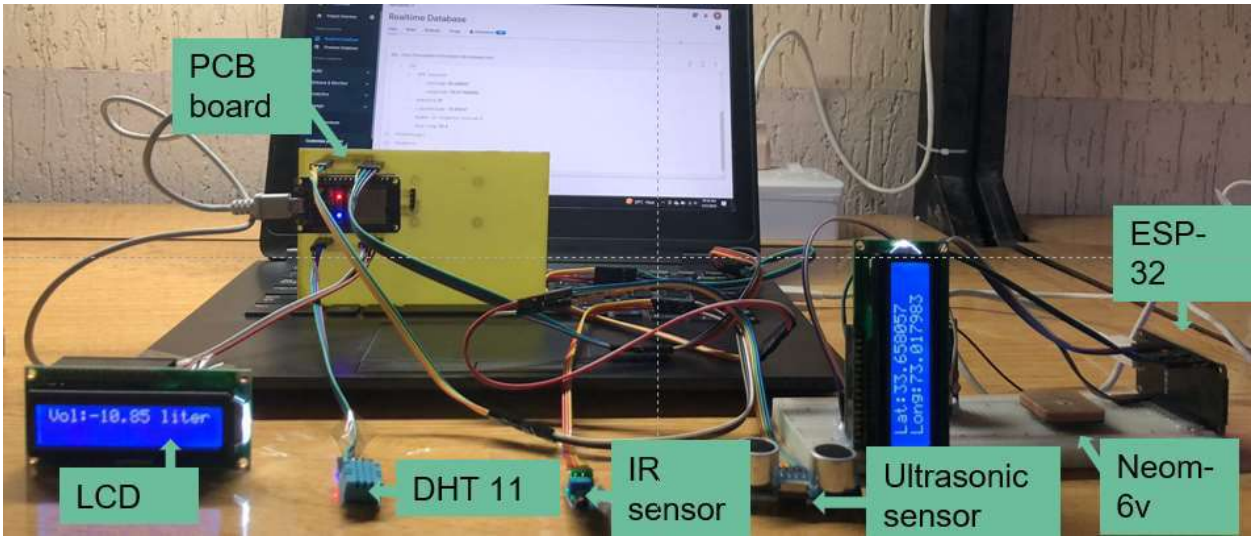


Figure 13 Hardware Design of the Sensor's Circuit

Initially, the sensors were tested using two different ESP-32. One ESP-32 was connected with the GPS Sensor and the other ESP32 was connected with the remaining sensors.

After testing all the sensors, they were connected to one ESP-32 to optimize the resources. The final hardware design was implemented as shown in Figure 12; the schematic diagram.

2.7.11 Hardware Casing Schematic

For the first step, the case was designed on CorelDraw software. The dimensions were chosen after calculating the initial design measurements of the hardware on the breadboard.

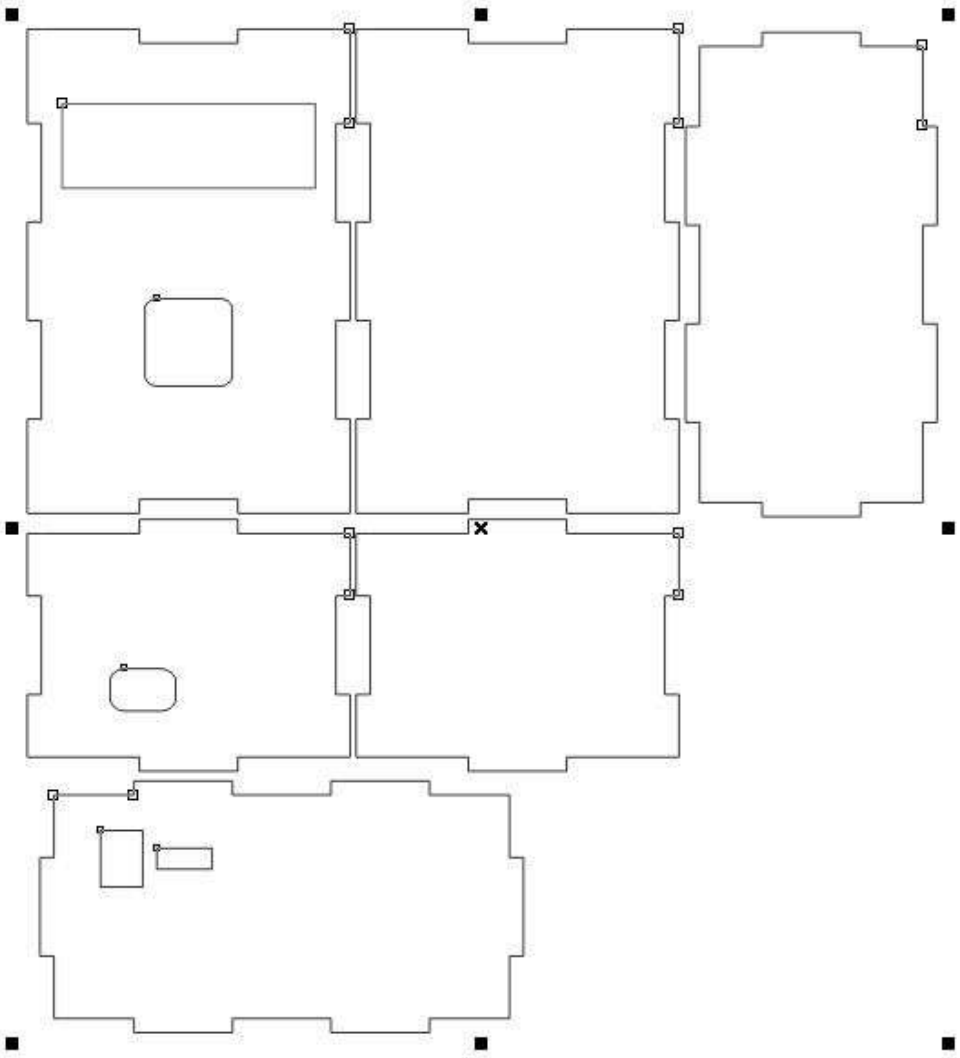


Figure 14 Schematic of the Hardware Casing

2.7.12 Hardware Casing of the Barcode Scanner

The final hardware casing of the barcode scanner circuit was as below:

Figure 15 Hardware Casing of the barcode scanner- paid fee status

Figure 16 Hardware Casing of the barcode scanner -unpaid/pending fee status

Figure 15 shows the blinking of the Green LED when the fee status is paid. Similarly, Figure 16 shows the blinking of Red LED as Pending fee status of student is displayed on the LCD.

2.7.13 Hardware Casing of the Sensor's circuit:

The final hardware casing of the sensor's circuit was as below:

Figure 17 Hardware Casing of the Sensor's Circuit

On the 16x2 LED, the volume of the fuel and the longitude and latitude were displayed. This data was also displayed on the Google firebase and the data from DHT11 and IR sensor was also displayed on the Google Firebase.

2.7.14 Sensor's Data Display on Google Firebase:

All the data read was displayed on Google Firebase by using ESP-32.

Figure 18 shows the real-time data read from the sensors to the Google Firebase. The first cell is the GPS location that can be seen. It changes in real-time as the location changes. The second cell is of Humidity. The third cell is of the fuel level; a numeric value can easily be used to monitor the availability of the fuel in vehicle. The fourth cell shows the data read from the IR sensor; incrementing the number of passengers boarding in. The last cell shows the temperature within the bus.

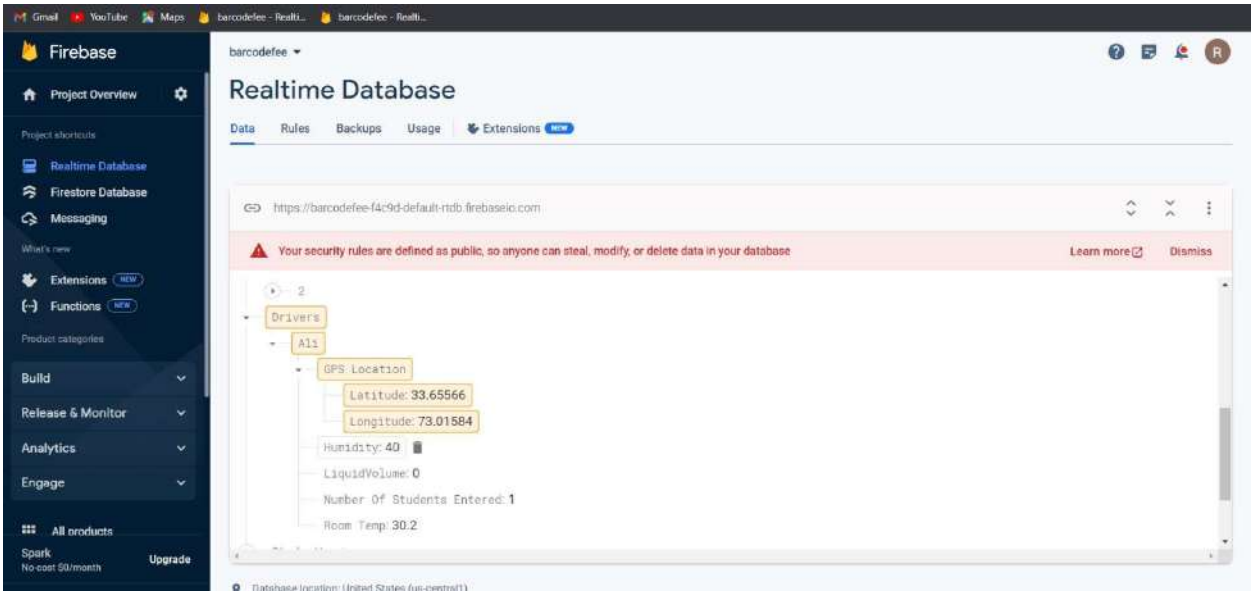


Figure 18 Data from sensors being displayed on Google Firebase

2.7.15 Application Implementation:

To implement the application, the tools used were the following:

- Flutter
- Visual Studio Code
- Android Studio
- Dart language

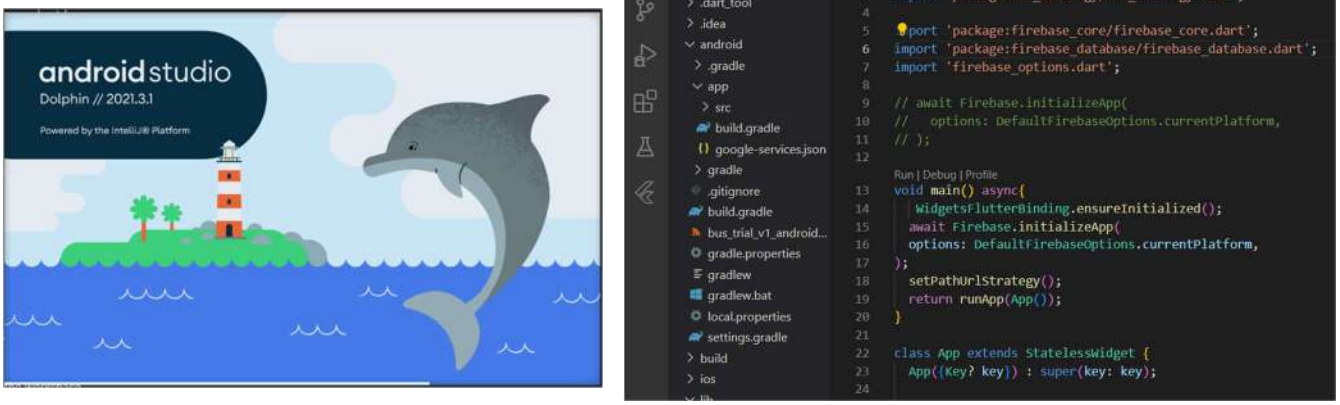


Figure 19 Android studio version used displayed on launch page and Visual Studio Code snippet on the right

Packages imported for flutter library, firebase, Google API and more.

- App title set for the app bar
- Routing for pages done
- Pages formed
- Buttons coded
- Data loaded from firebase
- Data added to a map variable
- Data loaded to an array
- Desired content displayed

```dart
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'package:url_strategy/url_strategy.dart';

import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_database/firebase_database.dart';
import 'firebase_options.dart';

```

Figure 20 Import packages code in Visual Studio Code

```dart
@override
Widget build(BuildContext context) => MaterialApp.router(
  routerDelegate: _router.routerDelegate,
  routeInformationParser: _router.routeInformationParser,
  routeInformationProvider: _router.routeInformationProvider,
);
```

Figure 21 Page routing code for the application and snippet of the emulator used on the right

```
79              ElevatedButton(
80                onPressed: () => context.go('/page2'),
81                child: const Text('Drivers Data'),
82              ), // ElevatedButton
83              const SizedBox(height: 10,),
84              ElevatedButton(
85                onPressed: () => context.go('/page3'),
86                child: const Text('Students Data'),
```

Figure 22 Code for buttons created in the application

```
129   loaddata() async{
130     final ref = FirebaseDatabase.instance.ref();
131     print("before");
132     final snapshot = await ref.child('Students:/').get();
133     print("hi");
134     if (snapshot.exists) {
135       a = snapshot.value;
136       List all = List.from(a as List);
137
138       for (var i = 0; i < all.length; i++) {
139         try {
140           Map<String, dynamic> _post = Map<String, dynamic>.from(all[i] as Map);
141           array.add(_post);
```

Figure 23 Code for getting data from the database to the application

```
181           child: Row(
182             children: [
183               Text(index.toString()+"-"),
184               Text('\t\t\tName:\t' + array[index]['Name'],style:TextStyle(color:array[index]['Status']=="Paid"?Colors.green:Colors.red)),
185               Text('\t\t\tRoll#:\t' + array[index]['Roll'],style:TextStyle(color:array[index]['Status']=="Paid"?Colors.green:Colors.red)),
186               Text('\t\t\tPhone:\t' + array[index]['Phone'],style:TextStyle(color:array[index]['Status']=="Paid"?Colors.green:Colors.red)),
187             ],
188           ), // Row
189         ) // Container
```

Figure 24 Code for displaying the students information in the application page

Google cloud services are utilized for push notifications and google maps API is installed for ETA, polylines and map display.

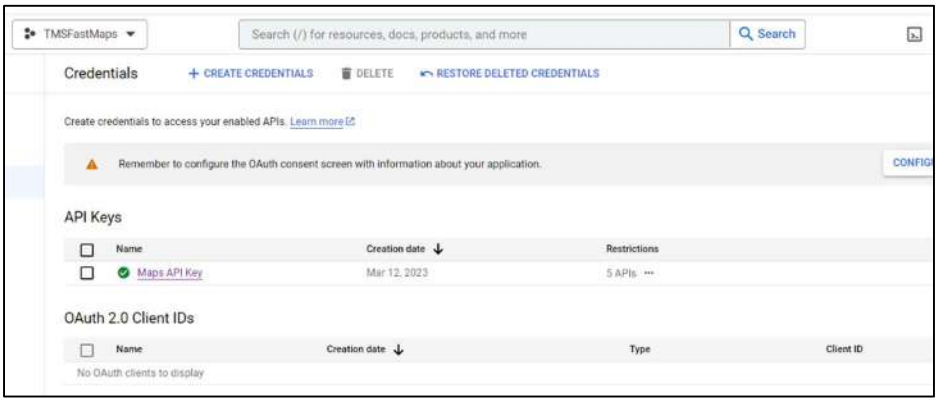


Figure 25 Cloud services API keys tab for maps API

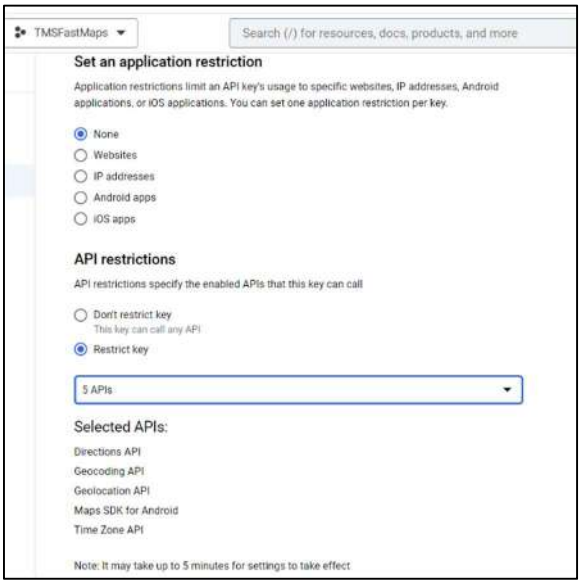


Figure 26 List of APIs selected to utilize in the application

2.7.16 Android application screenshots

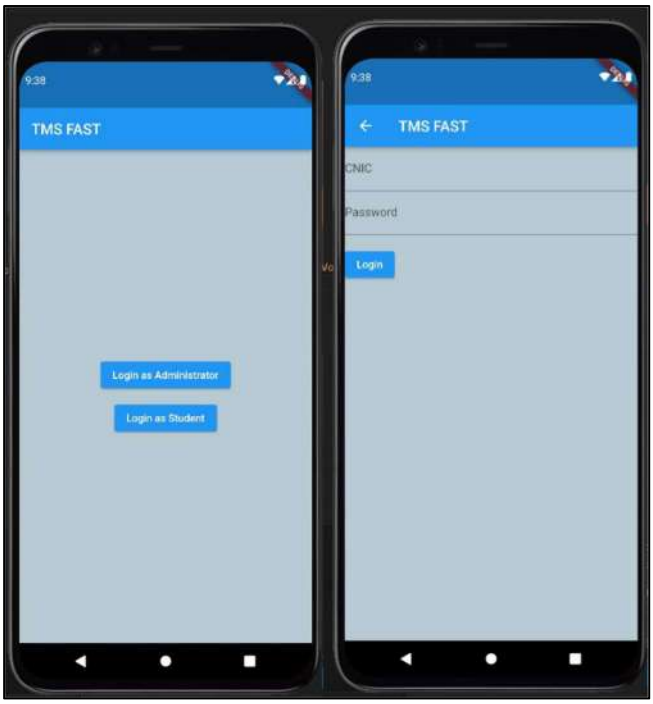


Figure 27 Initial landing page and login page for Administrator

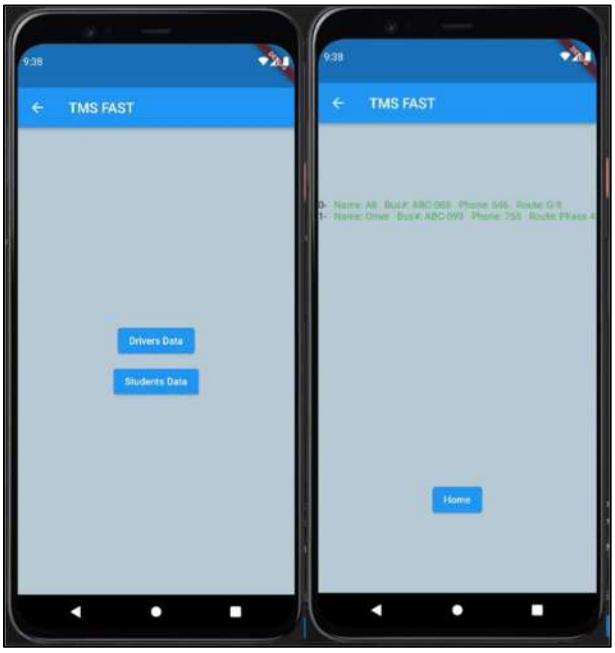


Figure 28 Administrator landing page and page under drivers data tab

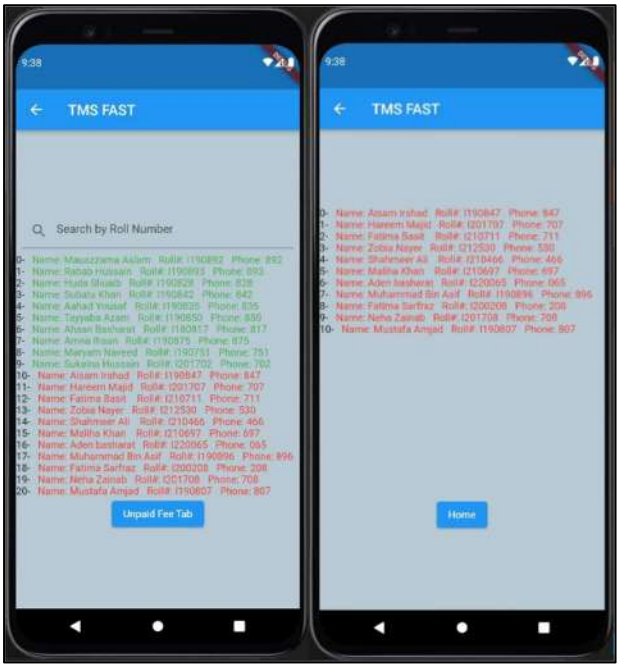


Figure 29 Student data tab and page of unpaid fee students

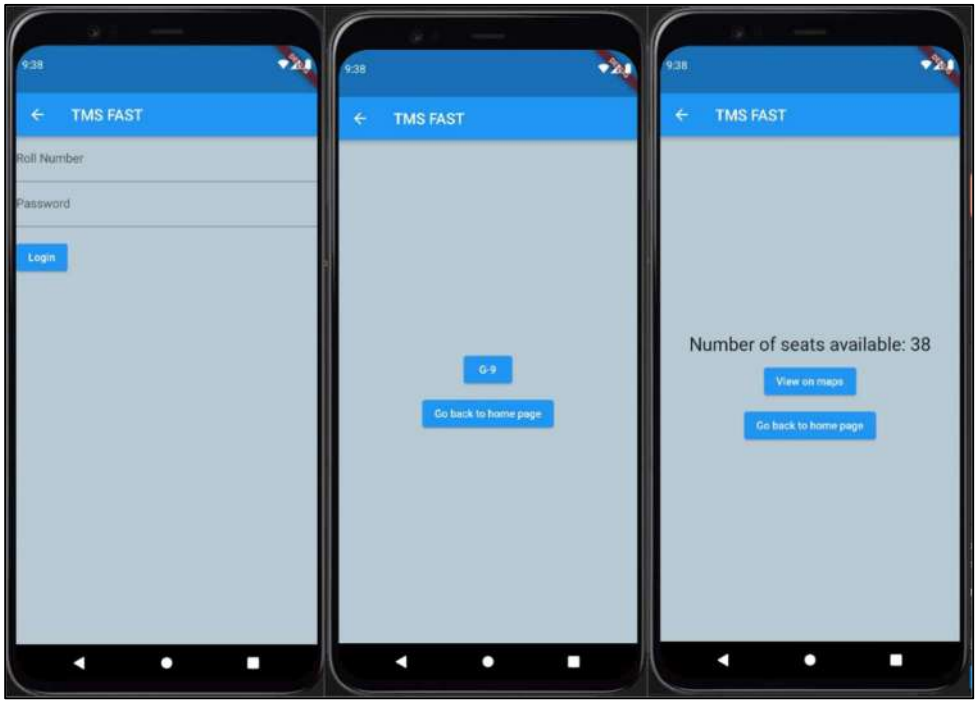


Figure 30 Student login page, landing page and route page displaying seat availability

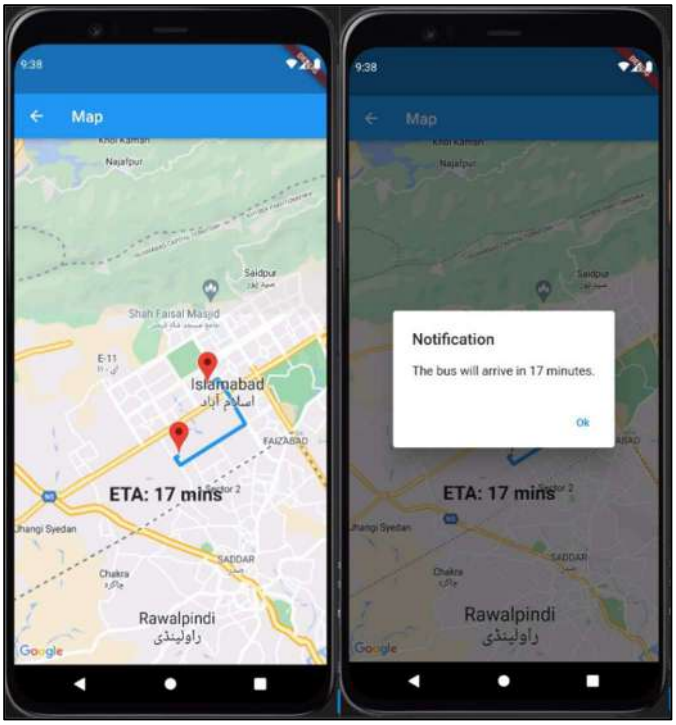


Figure 31 Maps route displayed with ETA of bus with a push notification

3. Chapter 3 Progress and Recommendations

3.1 Project Progress

3.1.1 Deliverables set

1. A functioning and developed application connected to a database.
2. Tested hardware modules with stored mock data.
3. A week of reported data on fuel and service quality.
4. Real-time tracking and Estimated Time Arrival(ETA) on the application.

3.1.2 Milestones till project closing

1.	Project proposal documentation and approval	13/09/2022
2.	Schema design	30/09/2022
3.	Basic app development learning and implementation	16/10/2022
4.	Database learning and implementation	05/11/2022
5.	Controller and hardware programming	30/11/2022
6.	ETA and Real-time tracking	01/04/2022
7.	Final project report	30/05/2023

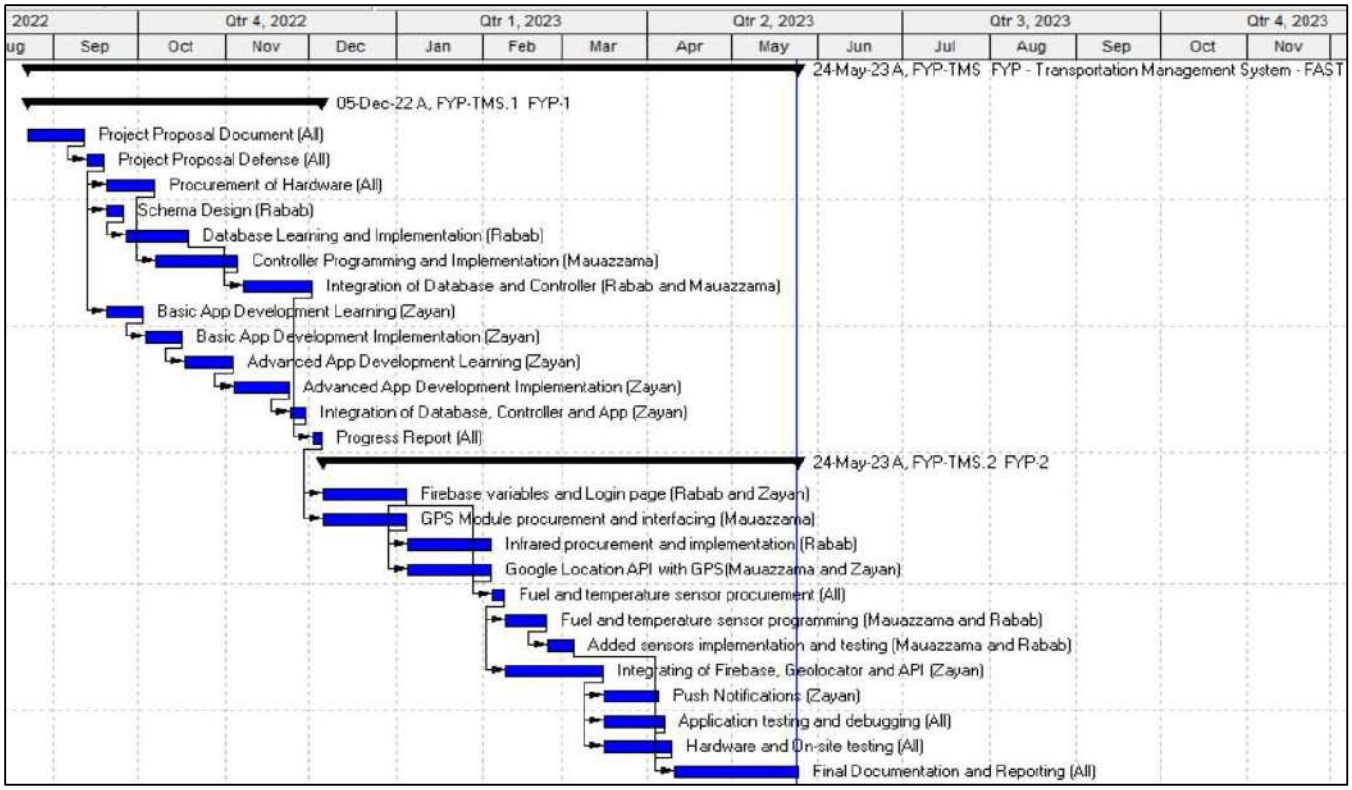


Figure 32 Project schedule using Primavera P6 in form of a Gantt chart

The Gantt chart entails the project divided in to its activities, starting from project proposal to all the modules fulfilled as per deliverables and ending with the conclusion of this final report.

3.2 Future recommendations

During the duration of this project we have concluded that the scope defined can be scaled vastly. With ample time, research and API modules this project can be generalized to all transportation systems around the world rather than just of a university.

Improved hardware, however costly can improve the results and administrative management largely. A fuel flow rate sensor, accelerometer and monitoring AI enhanced cameras can further add to the ease and accuracy of this management system. The application can be developed in to further platforms for ease of commuter's access.

3.3 Conclusion

The project followed all its defined milestones and a displayable prototype was made. Hardware was encased properly and ready to be deployed, similarly the software was ready to use.

One of the objectives for this project was to gain insights with respect to this project regarding sustainable development goals defined by the United Nations General Assembly.

- SDG 09: Industry, innovation and infrastructure
- SDG 11: Sustainable cities and communities
- SDG 12: Responsible consumption and production

The project proposes a new digitization of the transportation system used on a daily basis. The solution to economic fuel consumption complements the global goals responsible consumption and the elimination of paper receipts to mobile application advances in the sustainable cities target of the UN.

Further during this project our team has gained a learning experience on how to use new modules, programming languages and development platforms. The transportation management system for FAST provides an engineering solution to tackle a real life problem.

Appendix A – Codes utilized

Android application

The android application code has been uploaded on GitHub for further use by the community and better development in the future. The program can be found here: https://github.com/ZayanSafi/TMSFAST

However, some snippets from the code are provided below.

main:

```
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'package:url_strategy/url_strategy.dart';

import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_database/firebase_database.dart';
import 'firebase_options.dart';

import 'admin_side.dart';
import 'driver_data.dart';
import 'student_fee_data.dart';
import 'unpaid_students.dart';
import 'login.dart';
import 'student_side.dart';
import 'student_login.dart';
import 'admin_login.dart';
import 'route_1.dart';
import 'package:firebase_messaging/firebase_messaging.dart';
import 'g_9.dart';

// await Firebase.initializeApp(
//   options: DefaultFirebaseOptions.currentPlatform,
// );
  Future<void> _messageHandler(RemoteMessage message) async {
    print('background message ${message.notification!.body}');
  }

void main() async{
   WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
  options: DefaultFirebaseOptions.currentPlatform,
);
  setPathUrlStrategy();
  FirebaseMessaging.onBackgroundMessage(_messageHandler);
  return runApp(App());
}

class App extends StatelessWidget {
  App({Key? key}) : super(key: key);

  static const String title = 'TMS FAST';

  // @override
  // Widget build(BuildContext context) => MaterialApp.router(
  //   routerDelegate: _router.routerDelegate,
  //   routeInformationParser: _router.routeInformationParser,
  //   routeInformationProvider: _router.routeInformationProvider,

  // );
  @override
```

```
Widget build(BuildContext context) {
  return MaterialApp.router(
    routerDelegate: _router.routerDelegate,
    routeInformationParser: _router.routeInformationParser,
    routeInformationProvider: _router.routeInformationProvider,
    theme: ThemeData(
      scaffoldBackgroundColor: Color.fromARGB(255, 184, 203, 213),
    ),
    // builder: (context, child) {
    //   return Container(
    //     decoration: BoxDecoration(
    //       image: DecorationImage(
    //         image: AssetImage('C:\Games\bgimage.jpg'), // Replace with your image path
    //         fit: BoxFit.cover,
    //       ),
    //     ),
    //     child: child,
    //   );
    // },
  );
}

final GoRouter _router = GoRouter(
  errorBuilder: (context, state) => ErrorScreen(error:state.error),
  routes: <GoRoute>[
    GoRoute(

      routes: <GoRoute>[
        GoRoute(
          path: 'page2',
          builder: (BuildContext context, GoRouterState state) =>
          const Page2Screen(),
        ),
        GoRoute(
          path: 'page3',
          builder: (BuildContext context, GoRouterState state) =>
          const Page3Screen(),

        ),
        GoRoute(
          path: 'page4',
          builder: (BuildContext context, GoRouterState state) =>
          const Page4Screen(),

        ),
        GoRoute(
```

```
            path: 'page5',
            builder: (BuildContext context, GoRouterState state) =>
            const Page5Screen(),

          ),
           GoRoute(
            path: 'page6',
            builder: (BuildContext context, GoRouterState state) =>
            const Page6Screen(),

          ),GoRoute(
            path: 'page7',
            builder: (BuildContext context, GoRouterState state) =>
            Page7Screen(),

          ),GoRoute(
            path: 'page8',
            builder: (BuildContext context, GoRouterState state) =>
            Page8Screen(),

          ),
          GoRoute(
            path: 'page9',
            builder: (BuildContext context, GoRouterState state) =>
            MapScreen(),

          ),
          GoRoute(
            path: 'page10',
            builder: (BuildContext context, GoRouterState state) =>
            G9Screen(),

          ),
        ],
        path: '/',
          builder: (BuildContext context, GoRouterState state) =>
          const Page1Screen(),
          //const MapScreen(),
      ),
    ],

  );
}

class ErrorScreen extends StatelessWidget {
  final Exception? error;
  const ErrorScreen( {Key? key, required this.error}) : super(key: key);

  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Error"),

      ),
      body: Center(
        child: Text(
            error.toString()
        ),
      ),
    );
  }
}

adminlogin:
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'main.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_database/firebase_database.dart';
import 'firebase_options.dart';
import 'admin_side.dart';


class Page7Screen extends StatefulWidget {
  @override
  _Page7ScreenState createState() => _Page7ScreenState();
}

class _Page7ScreenState extends State<Page7Screen> {
  final _formKey = GlobalKey<FormState>();
  final _emailController = TextEditingController();
  final _passwordController = TextEditingController();
  FirebaseDatabase database = FirebaseDatabase.instance;
  late Object? a ;
  var array = [];


loaddata() async {
  final ref = FirebaseDatabase.instance.ref();
  print("before");
  final snapshot = await ref.child('AdminLogin/').get();
  print("hi");
  if (snapshot.exists) {
    a = snapshot.value;
    List all = List.from(a as List);

    for (var i = 0; i < all.length; i++) {
      try {
        Map<String, dynamic> _post =
            Map<String, dynamic>.from(all[i] as Map);
        array.add(_post);
      } catch (e) {
        print('null error');
      }
    }
    setState(() {});
    print('array: $array');
  } else {
    print('No data available.');
  }
}

@override
void initState() {
  super.initState();
```

```
  loaddata();
}

@override
Widget build(BuildContext context) => Scaffold(
      appBar: AppBar(title: const Text(App.title)),
      body: Form(
        key: _formKey,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            TextFormField(
              controller: _emailController,
              decoration: InputDecoration(labelText: 'CNIC'),
              validator: (value) {
                if (value!.isEmpty) {
                  return 'Please enter your 13-digit CNIC number (without dashes or spaces)';
                }
                return null;
              },
            ),
            TextFormField(
              controller: _passwordController,
              obscureText: true,
              decoration: InputDecoration(labelText: 'Password'),
              validator: (value) {
                if (value!.isEmpty) {
                  return 'Please enter your password';
                }
                return null;
              },
            ),
            Padding(
              padding: const EdgeInsets.symmetric(vertical: 16.0),
              child: ElevatedButton(
                onPressed: () {
                  if (_formKey.currentState!.validate()) {
                    final email = _emailController.text.trim();
                    final password = _passwordController.text.trim();
                    print('email: $email');
                    print('password: $password');
                    final match = array.any(
                        (map) => map['CNIC'].toString() == email && map['Password'] == password);
                    print('match: $match');
                    if (match) {
                      Navigator.push(
                        context,
                        MaterialPageRoute(builder: (context) => Page5Screen()),
                      );
                    } else {
                      print('Invalid cnic or password');
                    }
                  }
                },
                child: Text('Login'),
              ),
            ),
          ],
```

```
/// The screen of the fifth page.
class Page5Screen extends StatelessWidget {
  /// Creates a [Page5Screen].
  const Page5Screen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) => Scaffold(
    appBar: AppBar(title: const Text(App.title)),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          ElevatedButton(
            onPressed: () => context.go('/page2'),
            child: const Text('Drivers Data'),
          ),
          const SizedBox(height: 10,),
          ElevatedButton(
            onPressed: () => context.go('/page3'),
            child: const Text('Students Data'),
          ),
        ],
      ),
    ),
  );
}
```

```
driverdata:
import 'package:flutter/material.dart';
import 'package:go_router/go_router.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_database/firebase_database.dart';
import 'firebase_options.dart';
import 'main.dart';


class Page2Screen extends StatefulWidget {
  const Page2Screen({super.key});

  @override
  State<Page2Screen> createState() => _Page2ScreenState();
}

class _Page2ScreenState extends State<Page2Screen> {
  FirebaseDatabase database = FirebaseDatabase.instance;
  late Object? a ;
  var array = [];

  loaddata() async{
```

```
    final ref = FirebaseDatabase.instance.ref();
  print("before");
  final snapshot = await ref.child('Driver/').get();
  print("hi");
  if (snapshot.exists) {
      a = snapshot.value;
      List all = List.from(a as List);
      for (var i = 0; i < all.length; i++) {
        try {
          Map<String, dynamic> _post = Map<String, dynamic>.from(all
          if(_post['Active']=='Yes' || _post['Active']=='yes')
          {
              array.add(_post);
          }


        } catch (e) {
          print('null error');
        }

      }


      setState(() {

      });
  } else {
      print('No data available.');
  }
}

@override
void initState() {
  // TODO: implement initState
  super.initState();
  loaddata();
}

@override
Widget build(BuildContext context) {
  Size size = MediaQuery.of(context).size;
  return Scaffold(
  appBar: AppBar(title: const Text(App.title)),
  body: Center(
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children:  <Widget>[
        Container(
          height: 400,
          width: size.width,
          child:ListView.builder(
        itemCount: array.length,
        itemBuilder: (BuildContext context, int index) {
          return (
            Container(
              child: Row(
                children: [
```

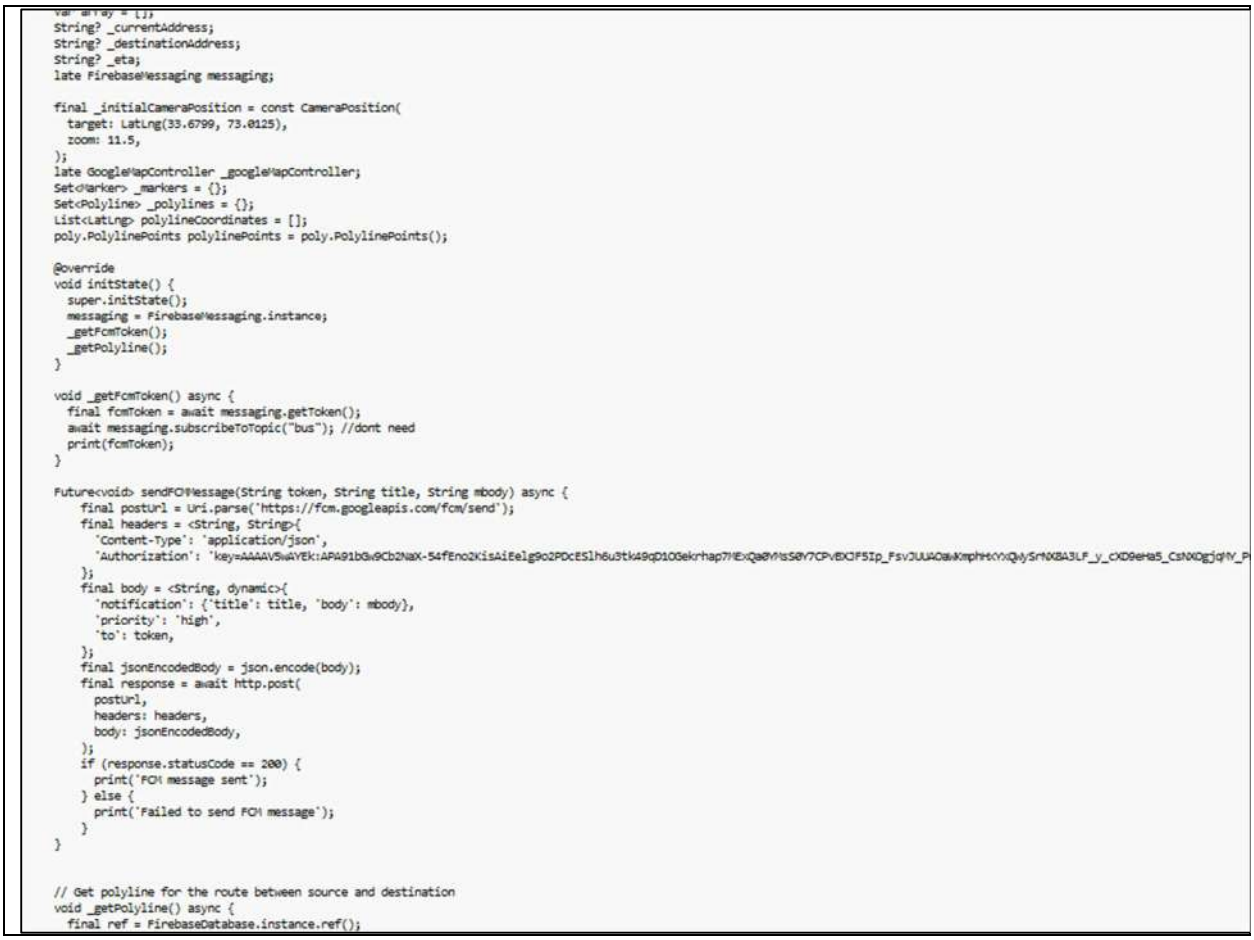
```
                children: [
                  Text(index.toString()+"-"),
                  Text('\t\t\tName:\t' + array[index]['Name'],style:TextStyle(color:array[index]['Active']=="Yes"?Colors.green:Colors.red)),
                  Text('\t\t\tBus#:\t' + array[index]['Bus'],style:TextStyle(color:array[index]['Active']=="Yes"?Colors.green:Colors.red)),
                  Text('\t\t\tPhone:\t' + array[index]['Phone'],style:TextStyle(color:array[index]['Active']=="Yes"?Colors.green:Colors.red)),
                  Text('\t\t\tRoute:\t' + array[index]['Route'],style:TextStyle(color:array[index]['Active']=="Yes"?Colors.green:Colors.red)),
                ],
              ),
            )
          );
        }),
    ),
          ElevatedButton(
              onPressed: () => context.go('/'),
              child: const Text('Home'))
        ],
      ),
    ),
  );
  }
}
```

```
class G9Screen extends StatefulWidget {
  const G9Screen({Key? key}) : super(key: key);

  @override
  _G9ScreenState createState() => _G9ScreenState();
}

class _G9ScreenState extends State<G9Screen> {
  late DatabaseReference _databaseReference;
  int? firebaseVariable;
  int availableSeats = 40;

  @override
  void initState() {
    super.initState();
    _databaseReference =
        FirebaseDatabase.instance.reference().child('/Drivers/Ali/Number Of Students Entered');
    _databaseReference.onValue.listen((DatabaseEvent databaseEvent) {
      setState(() {
        firebaseVariable = databaseEvent.snapshot.value as int?;
      });
    });
  }

  @override
  Widget build(BuildContext context) => Scaffold(
        appBar: AppBar(title: const Text(App.title)),
        body: Center(
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Text(
                'Number of seats available: ${availableSeats - (firebaseVariable ?? 0)}',
                style: TextStyle(fontSize: 24),
              ),
              const SizedBox(height: 10),
              ElevatedButton(
                onPressed: () => context.go('/page9'),
                child: const Text('View on maps'),
              ),
              const SizedBox(height: 10),
              ElevatedButton(
                onPressed: () => context.go('/'),
                child: const Text('Go back to home page'),
              ),
            ],
          ),
        ),
      );
}
```

```
/// The screen of the first page.
class Page1Screen extends StatelessWidget {
  /// Creates a [Page1Screen].
  const Page1Screen({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) => Scaffold(
    appBar: AppBar(title: const Text(App.title)),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          ElevatedButton(
            onPressed: () => context.go('/page7'),
            child: const Text('Login as Administrator'),
          ),
          const SizedBox(height: 10,),
          ElevatedButton(
            onPressed: () => context.go('/page8'), //page8 change
            child: const Text('Login as Student'),
          ),
        ],
      ),
    ),
  );
}
```

route_1:

```
import 'dart:async';
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:flutter_polyline_points/flutter_polyline_points.dart' as poly;
import 'package:google_directions_api/google_directions_api.dart' as directions;
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_database/firebase_database.dart';
import 'firebase_options.dart';
import 'package:geolocator/geolocator.dart';
import 'package:geocoding/geocoding.dart';
import 'package:firebase_messaging/firebase_messaging.dart';
import 'dart:convert';
import 'package:http/http.dart' as http;


class MapScreen extends StatefulWidget {
  const MapScreen({Key? key}) : super(key: key);

  @override
  _MapScreenState createState() => _MapScreenState();
}
class _MapScreenState extends State<MapScreen> {
  FirebaseDatabase database = FirebaseDatabase.instance;
  late Object? a ;
  var array = [];
```



```
var array = [];
String? _currentAddress;
String? _destinationAddress;
String? _eta;
late FirebaseMessaging messaging;

final _initialCameraPosition = const CameraPosition(
  target: LatLng(33.6799, 73.0125),
  zoom: 11.5,
);
late GoogleMapController _googleMapController;
Set<Marker> _markers = {};
Set<Polyline> _polylines = {};
List<LatLng> polylineCoordinates = [];
poly.PolylinePoints polylinePoints = poly.PolylinePoints();

@override
void initState() {
  super.initState();
  messaging = FirebaseMessaging.instance;
  _getFcmToken();
  _getPolyline();
}

void _getFcmToken() async {
  final fcmToken = await messaging.getToken();
  await messaging.subscribeToTopic("bus"); //dont need
  print(fcmToken);
}

Future<void> sendFCMMessage(String token, String title, String mbody) async {
    final postUrl = Uri.parse('https://fcm.googleapis.com/fcm/send');
    final headers = <String, String>{
      'Content-Type': 'application/json',
      'Authorization': 'key=AAAAV5WAYEk:APA91bGw9Cb2NaX-54fEno2KisAiEelg9o2PDcESlh6u3tk49qD1OGekrhap7HExQa0MtS0V7CPvBXJF5Ip_FsvJUAOBwKmpHHcYxQwySrNXBA3LF_y_cXD9eHa5_CsNXDgjqfV_P
    };
    final body = <String, dynamic>{
      'notification': {'title': title, 'body': mbody},
      'priority': 'high',
      'to': token,
    };
    final jsonEncodedBody = json.encode(body);
    final response = await http.post(
      postUrl,
      headers: headers,
      body: jsonEncodedBody,
    );
    if (response.statusCode == 200) {
      print('FCM message sent');
    } else {
      print('Failed to send FCM message');
    }
}


// Get polyline for the route between source and destination
void _getPolyline() async {
  final ref = FirebaseDatabase.instance.ref();
```

```
directions.DirectionsService.init('AIzaSyAJbdPy3UF-DvZDatNnAU3CevDVX6Hdye8');
final _directionsApi = directions.DirectionsService();

print("before");
final snapshot = await ref.child('Bus').get();
print("hi");
if (snapshot.exists) {
    a = snapshot.value;
    List all = List.from(a as List);
    for (var i = 0; i < all.length; i++) {
      try {
        Map<String, dynamic> _post = Map<String, dynamic>.from(all[i] as Map);
        array.add(_post);
        } catch (e) {
        print('null error');
      }
    }
    setState(() {
    });
print(array[0]['Long']);
print("End");
print(array[0]['Lat']);
}
// LatLng _sourceCoordinates = LatLng(33.6528, 73.0177);
LatLng _sourceCoordinates = LatLng(array[0]['Lat'],array[0]['Long']);
LatLng _destinationCoordinates = LatLng(33.6882, 73.0351);

_markers.add(Marker(
  markerId: MarkerId('source'),
  position: _sourceCoordinates,
  infoWindow: InfoWindow(
    title: 'Source',
    snippet: 'This is the source location',
  ),
));
_markers.add(Marker(
  markerId: MarkerId('destination'),
  position: _destinationCoordinates,
  infoWindow: InfoWindow(
    title: 'Destination',
    snippet: 'This is the destination location',
  ),
));

poly.PolylineResult result = await polylinePoints.getRouteBetweenCoordinates(
  'AIzaSyAJbdPy3UF-DvZDatNnAU3CevDVX6Hdye8',
  poly.PointLatLng(_sourceCoordinates.latitude, _sourceCoordinates.longitude),
  poly.PointLatLng(
      _destinationCoordinates.latitude, _destinationCoordinates.longitude),
  travelMode: poly.TravelMode.driving,
);
if (result.points.isNotEmpty) {
  result.points.forEach((poly.PointLatLng point) {
    polylineCoordinates.add(LatLng(point.latitude, point.longitude));
  });
}
```

```
if (result.points.isNotEmpty) {
  result.points.forEach((poly.PointLatLng point) {
    polylineCoordinates.add(LatLng(point.latitude, point.longitude));
  });
}
else{
  print("empty poly");
}

await placemarkFromCoordinates(_sourceCoordinates.latitude, _sourceCoordinates.longitude)
    .then((List<Placemark> placemarks) {
  Placemark place = placemarks[0];
  setState(() {
    _currentAddress =
        '${place.street}, ${place.subLocality}, ${place.subAdministrativeArea}, ${place.postalCode}';
  });
}).catchError((e) {
  debugPrint(e);
});

await placemarkFromCoordinates(_destinationCoordinates.latitude, _destinationCoordinates.longitude)
    .then((List<Placemark> placemarks) {
  Placemark place = placemarks[0];
  setState(() {
    _destinationAddress =
        '${place.street}, ${place.subLocality}, ${place.subAdministrativeArea}, ${place.postalCode}';
  });
}).catchError((e) {
  debugPrint(e);
});



final source = _currentAddress;
final destination = _destinationAddress;


// Get the route between the source and destination
final request = directions.DirectionsRequest(
  origin: source,
  destination: destination,
  travelMode: directions.TravelMode.driving,
);

print(source);
print(destination);

_directionsApi.route(request,
  (directions.DirectionsResult response, directions.DirectionsStatus? status) {
if (status == directions.DirectionsStatus.ok) {
  print("Directions request successful");
  // Extract the ETA from the response object
  final duration = response.routes?.first.legs?.first.duration!.text;
  print("ETA: $duration");
  setState(() {
    _eta = 'ETA: $duration';
  });
  // Check if ETA is 5 minutes or less
  if (duration != null && duration == '5 mins') {
    print("Sending notification...");
    sendFCMessage(
    'fBii-1grTFOr6us82PxWWO:APA91bHBV8d1jd6_naS_zDA2KEJpN6HepbbmET3JhQXAHP-1TMS7Qg6PrTmN0cbWphtTYOJAXe7oHYHyf-RXm4eF6G7cAPVosW36U1P2n9C-I1IlySNws52d7xf5uzqtYKnhTpI2d-CH',
```

Hardware

Barcode scanner

```
#include <SoftwareSerial.h>
#define rxPin 14
#define txPin 12
#define red 15
#define green 13
SoftwareSerial mySerial = SoftwareSerial(rxPin, txPin);
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Set the LCD address to 0x27 for a 16 chars and 2 line display
LiquidCrystal_I2C lcd(0x27, 16, 2);

#include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#include <addons/RTDBHelper.h>

#define WIFI_SSID "Hexagon"
#define WIFI_PASSWORD "twentythree"
#define DATABASE_URL "barcodefee-f4c9d-default-rtdb.firebaseio.com"  //<databaseName>.firebaseio.com or <databaseName>.<region>.firebasedatabase.app
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseJson json;  // format for storing and transporting data.
FirebaseConfig config;

byte inData;
char inChar;
String BuildINString;
String Feestatus;
String Name;


void setup()
{
  Serial.begin(115200);
  mySerial.begin(115200);
  pinMode(red, OUTPUT);
  pinMode(green, OUTPUT);

  lcd.begin();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Connecting To");
  lcd.setCursor(0, 1);
  lcd.print("      Wifi");
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
```

```
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  config.database_url = DATABASE_URL;
  config.signer.test_mode = true;
  Firebase.reconnectWiFi(true);
  Firebase.begin(&config, &auth);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Barcode Scanner");
  lcd.setCursor(0, 1);
  lcd.print("   Ready");
}

void loop() {
  ReadData();
  delay(200);
  if (BuildINString != "")
  {
    lcd.clear();
    BuildINString.trim();
    Serial.print(">");
    Serial.print(BuildINString);
    Serial.println("<");
    Firebase.getString(fbdo, "/status_of_students/" + BuildINString + '/' + "Name");
    Name = fbdo.to<String>();
    Serial.println(Name);
    lcd.print("Name:");
    lcd.setCursor(0, 1);
    lcd.print(Name);
    delay(2000);
    lcd.clear();
    Firebase.getString(fbdo, "/status_of_students/" + BuildINString + '/' + "Fees");
    Feestatus = fbdo.to<String>();
    Serial.println(Feestatus);
    lcd.print("Fee:");
    lcd.print(Feestatus);
    if (Feestatus == "Paid")
    {
      digitalWrite(green, HIGH);
      digitalWrite(red, LOW);
    }
    else if (Feestatus == "Pending")
    {
      digitalWrite(green, LOW);
      digitalWrite(red, HIGH);
    }
    else if (Feestatus == "Not Paid")
    {
      digitalWrite(green, LOW);
      digitalWrite(red, HIGH);
    }
    BuildINString = "";
  }
}

void ReadData() {
  while (mySerial.available() > 0) {
    inData = 0;
    inChar = 0;
    inData = mySerial.read();
    inChar = char(inData);
    BuildINString = BuildINString + inChar;
  }
}
```

GPS and remaining sensors

```
//---------GPS----------
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
static const int RXPin = 16, TXPin = 17;
static const uint32_t GPSBaud = 9600;
TinyGPSPlus gps;
SoftwareSerial ss(RXPin, TXPin);

//-------Variables-------
#include <math.h>
float h;
float t;
float liquid_volume;
const int IR_PIN = 33; // Change this to the pin connected to your IR module
int count = 0;
int lastState = HIGH;

//--------firebase-----
#include <WiFi.h>
#include <FirebaseESP32.h>
#include <addons/RTDBHelper.h>
#define WIFI_SSID "Hexagon"
#define WIFI_PASSWORD "twentythree"
#define DATABASE_URL "barcodefee-f4c9d-default-rtdb.firebaseio.com"  //<databaseName>.firebaseio.com or <databaseName>.<region>.firebasedatabase.app
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

//-------I2C Lcd--------
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display

//------Ultrasonic------
#define TRIG_PIN 32
#define ECHO_PIN 35

//------DHT------
#include "DHT.h"
#define DHTPIN 4
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
float latt;
float lngg;
//-------Setup------
void setup() {

  Serial.begin(115200);
  ss.begin(GPSBaud);
```

```
  Wire.begin();
  lcd.begin();
  lcd.backlight();
  dht.begin();

  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  pinMode(2, OUTPUT);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  digitalWrite(2, HIGH);
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());
  Serial.println();
  Serial.printf("Firebase Client v%s\n\n", FIREBASE_CLIENT_VERSION);
  config.database_url = DATABASE_URL;
  config.signer.test_mode = true;
  Firebase.reconnectWiFi(true);
  Firebase.begin(&config, &auth);

  pinMode(IR_PIN, INPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  Serial.println(F("DHTxx test!"));
}
void loop()
{
  while (ss.available() > 0) {
    if (gps.encode(ss.read()))
    {
      Serial.print(F("Location: "));
      latt = gps.location.lat();
      lngg = gps.location.lng();
      Serial.print("Lattitude:");
      Serial.print(latt, 6);
      Serial.print(" Longitude:");
      Serial.print(lngg, 6);
      lcd.setCursor(0, 1);
      lcd.print(latt);
      lcd.print(",");
```

```
      lcd.print(lngg);
      digitalWrite(TRIG_PIN, LOW);
      delayMicroseconds(2);
      digitalWrite(TRIG_PIN, HIGH);
      delayMicroseconds(10);
      digitalWrite(TRIG_PIN, LOW);
      unsigned long pulse_width = pulseIn(ECHO_PIN, HIGH);
      float distance = pulse_width / 58.0;
      float jar_radius = 8.75; // radius of the jar in centimeters
      float jar_height = 20.0; // height of the jar in centimeters
      float liquid_height = jar_height - distance; // height of the liquid in the jar in centimeters
      liquid_volume = 3.14159 * jar_radius * jar_radius * liquid_height / 1000.0; // volume of the liquid in the jar in liters
      if (liquid_volume < 0)
      {
        liquid_volume = 0;
      }
      Serial.print("Liquid volume: ");
      Serial.print(liquid_volume);
      Serial.println(" liters");
      lcd.setCursor(0, 0);
      lcd.print("V:");
      lcd.print(liquid_volume);
      lcd.println(" liters   ");

      int currentState = digitalRead(IR_PIN);
      if (currentState == LOW && lastState == HIGH) {
        count++;
        Serial.println("Person detected. Count: " + String(count));
      }
      lastState = currentState;
      h = dht.readHumidity();
      t = dht.readTemperature();
      Serial.print(F("Humidity: "));
      Serial.print(h);
      Serial.print(F("%  Temperature: "));
      Serial.print(t);
      Serial.print(F("°C "));
      Serial.print(" counting");
      Serial.println(count);

      Firebase.setInt(fbdo, "/Drivers/Ali/GPS Location/Latitude",  latt);
      Firebase.setInt(fbdo, "/Drivers/Ali/GPS Location/Longitude",  lngg);
      Firebase.setInt(fbdo, "/Drivers/Ali/Room Temp", t);
      Firebase.setInt(fbdo, "/Drivers/Ali/Humidity", h);
      Firebase.setInt(fbdo, "/Drivers/Ali/LiquidVolume", liquid_volume);
      Firebase.setInt(fbdo, "/Drivers/Ali/Number Of Students Entered",  count);
    }
  }
}
```

Bibliography

[1] M. Iqbal, S. Noreen, T. Sabir, S. Afghani. (2014, January). “Intelligent Transport Fleet Management System,” IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE)

e-ISSN: 2278-1676,p-ISSN: 2320-3331, Volume 9, Issue 1 Ver. I, PP 68-75, Jan 2014.

[2] C. Maria, I. Petrescu. “Use in Database in Advanced Transport Management System – Interface Using XML - Study Of Case,” [Online], Feb 2011.

[3] S. Mulay, C. Dhekne, R. Bapat, T. Budukh. “Intelligent City Traffic Management and Public Transportation System”, JCSI International Journal of Computer Science Issues, Vol. 10, Issue 3, No 1, May 2013

[4] Raj Kiran T, T Abhinav, V Nafeez, Adithya H B, Amulya S, R Meghana, Sunil MP. “Student database management and inquiry system using barcode scanner”, Department of Electronics and Communication Engineering, School of Engineering and Technology, Jain University, Bangalore, Vol-1 Issue-5, 2016.