SMART COLLEGE BUS TRACKER: REAL - TIME LOCATION AND ETA PREDICTOR

A PROJECT REPORT

Submitted by

RAHUL S	**510421104083**
SRICHANDIRAN S	**510421104095**
SRIRAM S	**510421104097**
YUVARAJ K	**510421104120**

in partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ARUNAI ENGINEERING COLLEGE,

(AUTONOMOUS)

TIRUVANNAMALAI - 606 603

ANNA UNIVERSITY :: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY :: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report titled **"SMART COLLEGE BUS TRACKER: REAL -TIME LOCATION AND ETA PREDICTOR"** is the bonafide work Done by **RAHUL S (510421104083), SRICHANDIRAN S (510421104095), SRIRAM S (510421104097), YUVARAJ K (510421104120),** who carried out the project work under my supervision.

SIGNATURE OF SUPERVISIOR
Ms. M. SOWNDHARYA, M.E.,
ASSISTANT PROFESSOR,
Computer Science & Engineering,
Arunai Engineering College,
Tiruvannamalai - 606603.

SIGNATURE OF HOD
Mrs. V. UMADEVI, M.E.,
ASP - HEAD OF THE DEPARTMENT,
Computer Science & Engineering,
Arunai Engineering College,
Tiruvannamalai -606603.

INTERNAL EXAMINER

EXTERNAL EXAMINER

CERTIFICATE OF EVALUATION

COLLEGE NAME : 5104-ARUNAI ENGINEERING COLLEGE
BRANCH : B.E COMPUTER SCIENCE AND ENGINEERING
SEMESTER : 8th SEMESTER (2021-2025)
SUBJECT CODE & NAME: CS3811 - PROJECT WORK
DATE OF EXAMINATION:

NAME OF THE STUDENT	REGISTER NUMBER	TITLE OF THE PROJECT	NAME OF GUIDE WITH DESIGNATION
RAHUL S	510421104083	SMART COLLEGE BUS TRACKER: REAL-TIME LOCATION AND ETA PREDICTOR	Ms. M. SOWNDHARYA, M.E, Assistant Professor, Computer Science & Engineering
SRICHANDIRAN S	510421104095		
SRIRAM S	510421104097		
YUVARAJ K	510421104120		

The report of the **PROJECT WORK** submitted for the fulfillment of Bachelor of Engineering degree in **COMPUTER SCIENCE AND ENGINEERING** of Anna University was evaluated and confirmed to be report of the work done by the above students

SIGNATURE OF SUPERVISIOR
Ms. M. SOWNDHARYA, M.E.,
ASSISTANT PROFESSOR,
Computer Science & Engineering,
Arunai Engineering College,
Tiruvannamalai-606603.

SIGNATURE OF HOD
Mrs. V. UMADEVI, M.E.,
ASP - HEAD OF THE DEPARTMENT,
Computer Science & Engineering,
Arunai Engineering College,
Tiruvannamalai-606603.

INTERNAL EXAMINER

EXTERNALEXAMINER

ACKNOWLEDGEMENT

It is our duty to thank the GOD Almighty and my beloved parents and teachers for their persistent blessing and constant encouragement to reach this level of what I am today.

A project of this nature needs co-operation and support from many for successful completion. In this regard, we are fortunate to express my heartfelt thanks to our beloved Founder Chairman **MR.E.V.VELU,** Chair Person **Mrs.SANKARI VELU** and **Er.E.V.KUMARAN, M.E.,** Vice Chairman, for providing necessary facilities with high-class environment throughout the course.

We take the privilege of expressing our sincere thanks to our beloved Registrar **Dr.R.SATHIYASEELAN,Ph.D.,** for granting permission to undertake the project and we also express my hearty thanks to principal, **Dr.C.ELANCHEZHIAN, Ph.D.,** for their advice in accomplishing this project work.

We are most grateful to **Mrs.V.UMADEVI.M.E**, ASP - Head of the Department, Department of Computer Science and Engineering, who has given me both moral and technical support adding experience to the job we have undertaken.

We are most grateful to our guide **Ms.M.SOWNDHARYA,M.E.,** Assistant Professor, for her astonishing guidance and encouragement for the successful completion of this project. They held us to the highest standards of quality and accuracy.

Last but not least we would like to thank my family members, friends, teaching and non-teaching faculties, who has assisted us directly or indirectly throughout this project.

ABSTRACT

SMART COLLEGE BUS TRACKER: REAL-TIME LOCATION AND ETA PREDICTOR

The increasing reliance on institutional transportation systems calls for innovative digital solutions to address inefficiencies in communication, punctuality, and safety. This paper introduces the Smart College Bus Tracker, a real-time cross-platform mobile application developed to enhance the student commuting experience by providing accurate bus location updates and Estimated Time of Arrival (ETA) information. Leveraging GPS technology and Google Maps API, the application continuously tracks the live location of college buses and dynamically calculates arrival times based on current speed, distance, and real-time traffic conditions.

Designed using the Flutter framework, the system ensures seamless deployment on both Android and iOS platforms, enabling broad accessibility. The application features two dedicated login modules—one for drivers and another for students—offering tailored functionalities such as route configuration, live location broadcasting, student-side bus selection, and ETA visualization. Firebase is employed for secure user authentication, real-time location updates, cloud-based route management, and data storage, ensuring low latency and high reliability.

In addition to reducing student wait times and eliminating uncertainty in bus arrivals, the application fosters accountability and improves safety by minimizing the need for unplanned phone communication or physical coordination. A key feature of the system is its modular and scalable architecture, which allows for easy integration of additional capabilities such as machine learning-based predictive analytics for demand forecasting, RFID-based student attendance tracking, push notification alerts, and data-driven decision-making for route optimization.

Preliminary testing within a controlled campus environment shows high system responsiveness, minimal lag in location updates, and strong user engagement, demonstrating the app's feasibility and value. The paper concludes by discussing deployment strategies, data privacy considerations, and potential collaborations with transport departments in other academic institutions for broader adoption and continuous improvement.

ABSTRACT (TAMIL)

ஸ்மார்ட் கல்லூரி பேருந்து கண்காணிப்பாளர்: நிகழ்நேர இருப்பிடம் மற்றும் ETA முன்னறிவிப்பாளர்

கல்லூரி போக்குவரத்து அமைப்புகளின் மீதான அதிகரிக்கும் நம்பிக்கை தொடர்பு, நேர்த்தி மற்றும் பாதுகாப்பில் உள்ள செயலிழப்புகளை சரிசெய்ய புதிய டிஜிட்டல் தீர்வுகளை தேவைப்படுத்துகிறது. இக்கட்டுரை, Smart College Bus Tracker எனும் நேரடி நேரத்தில் செயல்படும் மொபைல் பயன்பாட்டை அறிமுகப்படுத்துகிறது. இது மாணவர்களின் பயண அனுபவத்தை மேம்படுத்தும் வகையில், துல்லியமான பேருந்து இருப்பிட தகவல்கள் மற்றும் கணிக்கப்பட்ட வருகை நேரத்தைக் (ETA) வழங்குகிறது.

GPS தொழில்நுட்பம் மற்றும் Google Maps API-ஐ பயன்படுத்தி, இந்த பயன்பாடு கல்லூரி பேருந்துகளின் நேரடி இருப்பிடத்தைக் கண்காணிக்கிறது மற்றும் தற்போதைய வேகம், தூரம் மற்றும் போக்குவரத்து நிலைமைகள் அடிப்படையில் ETA-ஐ தானாகவே கணிக்கிறது. Flutter ஃபிரேம்வொர்க்கை அடிப்படையாக கொண்டு வடிவமைக்கப்பட்ட இந்த அமைப்பு, Android மற்றும் iOS ஆகிய இரண்டிலும் எளிதில் நிறுவப்படும் வகையில் உருவாக்கப்பட்டுள்ளது, இதனால் பரந்த அளவில் அணுகல் உண்டு.

இந்த பயன்பாட்டில் இரண்டு தனி உள்நுழைவு தொகுதிகள் உள்ளன — ஒன்று டிரைவர்களுக்காகவும், மற்றொன்று மாணவர்களுக்காகவும் — ஒவ்வொன்றிலும் வழித்தட உள்ளமைவு, நேரடி இருப்பிட ஒளிபரப்பு, பேருந்து தேர்வு மற்றும் ETA காட்சி போன்ற தனிப்பட்ட அம்சங்கள் வழங்கப்படுகின்றன. Firebase தொழில்நுட்பம் பாதுகாப்பான உள்நுழைவு, நேரடி தரவு ஒத்திசைவு, மேகத்தில் வழித்தட நிர்வாகம் மற்றும் தரவுகளை சேமிப்பதற்காக பயன்படுத்தப்படுகிறது, இது குறைந்த தாமதத்தையும் உயர் நம்பகத்தையும் உறுதி செய்கிறது.

மாணவர்கள் பேருந்துகளை எதிர்பார்த்து நிற்கும் நேரத்தை குறைக்கும், பேருந்து வருகையில் ஏற்பட்ட குழப்பங்களை நீக்கும் இதன் திறனுடன், இந்த பயன்பாடு தொலைபேசிக் தொடர்பு மற்றும் நேரடி ஒத்திசைவைத் தவிர்த்து பாதுகாப்பையும் பொறுப்பையும் மேம்படுத்துகிறது. இந்த அமைப்பின் முக்கிய சிறப்பம்சம் என்பது அதன் தொகுதி அடிப்படையிலான, விரிவாக்கக்கூடிய கட்டமைப்பாகும். இதன் மூலம், எதிர்காலத்தில் இயந்திர கற்றல் அடிப்படையிலான முன்னறிவிப்பு பகுப்பாய்வுகள், RFID அடிப்படையிலான மாணவர் வருகை கண்காணிப்பு, தகவல் அறிவிப்புகள் மற்றும் வழித்தடங்கள் பற்றிய பகுப்பாய்வுகளுடன் முடிவெடுக்கும் திறன்களையும் இணைக்கலாம்.

கல்லூரி வளாகத்தில் கட்டுப்படுத்தப்பட்ட சூழலில் நடந்த ஆரம்ப சோதனைகள், இந்த பயன்பாட்டின் மிகுந்த எதிர்வினை திறன், குறைந்த நேரம் தாமதம் மற்றும் பயனாளர் திருப்தியை வெளிப்படுத்துகின்றன. இக்கட்டுரை, விரிவாக்கக் கண்ணோட்டங்கள், தரவுப் பாதுகாப்பு கவலைகள் மற்றும் இதைப் பிற கல்வி நிறுவனங்களுடன் இணைந்து செயல்படுத்தும் வாய்ப்புகள் குறித்த விவாதத்துடன் முடிகிறது.

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO	ABBREVIATIONS	FULL FORM
1	ETA	Estimated Time of Arrival
2	GPS	Global Positioning System
3	UI	User Interface
4	UX	User Experience
5	ID	Identification
6	API	Application Programming Interface
7	DB	Database
8	KM	Kilometer
9	MB	Megabyte
10	SDK	Software Development Kit
11	OTP	One-Time Password
12	HTTP	Hypertext Transfer Protocol
13	RTDB	Real-Time Database
14	UI/UX	User Interface/User Experience
15	JSON	JavaScript Object Notation

TABLE OF CONTENT

CHAPTER 1

1. INTRODUCTION

In today's fast-paced academic environment, punctuality and efficient communication are essential components of a smooth institutional transportation system. Traditional methods of tracking college buses—such as relying on fixed schedules or word-of-mouth updates—often lead to confusion, delays, and safety concerns among students and staff. These issues are further magnified during peak hours or adverse weather conditions when accurate and real-time information is critical.

To address these challenges, this project presents the Smart College Bus Tracker, a mobile application designed to provide students with real-time updates on the location and Estimated Time of Arrival (ETA) of their college buses. By integrating GPS technology and the Google Maps API, the application offers dynamic tracking of buses and adjusts ETA based on live traffic conditions, speed, and distance.

The application is developed using Flutter, ensuring compatibility across both Android and iOS platforms. It features two user-specific modules: one for drivers, which allows them to broadcast live location and configure routes; and another for students, enabling them to select their bus and view accurate ETA. Firebase is used for secure authentication, real-time data synchronization, and cloud data storage.

This system not only reduces wait times and improves travel planning but also enhances safety by eliminating the need for unnecessary calls or manual updates. With a scalable and modular architecture, the Smart College Bus Tracker is future-ready, capable of integrating advanced features like predictive analytics, push notifications, and RFID-based attendance systems. The project aims to revolutionize how college transport systems function, making commuting more reliable, transparent, and efficient for all stakeholders.

CHAPTER 2

2. LITREATURE SURVEY

1.Title: Revolutionizing Bus Tracking and Arrival Prediction in Real Time for Both Public and Private Sectors, Powered by Firebase

Authors: Mohammed Shafiulla, Abdul Lateef Haroon P S, Shaik Mohammed, Suhail Ahmed, Mahika Dhar M, Utkarsh Kumar in 2024

Description:

Abstract— Public transport systems often face challenges related to reliability and efficiency, leading to inconvenience and uncertainty for passengers. Existing solutions for real-time bus monitoring typically rely on expensive Internet of Things (IoT) components. This paper presents Track N Go, a cost-effective alternative for real-time monitoring of public buses and taxis using mobile phone location data. The system utilizes a robust backend system built with Java and React.Js for the user interface. Integration of GPS via the Map Box API and Firebase database ensures accurate location tracking. Track N Go provides accurate real-time monitoring, addressing challenges associated with public transport and improving overall urban mobility. The system enhances convenience and peace of mind for passengers by delivering reliable and user-centric solutions. The evaluation of Track N Go demonstrates its effectiveness in improving urban mobility and transit services. By offering a reliable and cost-effective solution for real-time bus monitoring, Track N Go aims to streamline daily commutes and make public transport more efficient and predictable.

2.Title: Real-Time Campus University Bus Tracking Mobile Application

Authors: Shafri A. Sharif, Mohd S. Suhaimi, Nurul N. Jamal, Irsyad K. Riadz, Ikmal F. Amran, Dayang N. A. Jawawi in 2018

Description:

Real-Time Campus University Bus Tracking Mobile Application is a mobile application to help campus members detect the current location of the bus in real-time. Real-Time Campus University Bus Tracking Mobile Application is a hybrid mobile application. However, for this development, it is developed for Android user only. It can show updated estimation time arrival and the number of persons inside the bus. This project using two devices embedded inside the bus, which are GPS Tracker device

and IoT people counter device. All devices will transmit the data into cloud database which is Firebase. Real-Time Campus University Bus Tracking Mobile Application is developed as a platform for user to receive the data transmitted from database. Other than that, Student will know the time arrival of the bus and the current quantity of people inside the bus to lead them avoid wasted time knowing that they wait for the bus that pack of passenger. The student also able to make complaint and feedback via the platform. Furthermore, this project using PhoneGap as a tool platform to develop the application. The GPS Tracker device using Arduino and IoT people counter using Raspberry PI to transmit data. Overall this project using the reusability techniques and Agile method to complete all the system which it is involved four interactions to make it full system work as expected.

3. Title: SmartCommute: A Cloud-Integrated Real-Time Bus Tracking and ETA Prediction System Using Flutter and Firebase

Authors: R. Meenakshi, A. Tharun Kumar, V. Pranav Raj, K. Sowmya, M. Dinesh, P. Harini – 2024

Description:

Urban commuters frequently encounter issues such as uncertainty in bus arrivals, long wait times, and inefficient route tracking. Existing systems often lack scalability and demand high-end hardware. SmartCommute introduces an affordable, scalable, and mobile-centric solution for real-time bus tracking and Estimated Time of Arrival (ETA) predictions. Built using Flutter for cross-platform compatibility, the application leverages Google Maps API for route visualization and GPS tracking, while Firebase Firestore serves as the cloud database backend to store and sync location data efficiently. The system includes distinct interfaces for drivers and passengers, enabling real-time communication and accurate bus location updates. SmartCommute's modular design also allows integration with route scheduling algorithms to dynamically adjust ETAs based on traffic conditions. Experimental deployment across select educational institutions demonstrated improved punctuality and user satisfaction. By eliminating the need for dedicated tracking hardware and relying on mobile GPS, SmartCommute provides an effective and budget-friendly alternative for both public and institutional transport monitoring.

CHAPTER 3

3. OBJECTIVE

The **Smart College Bus Tracker** project has been conceptualized with a clear set of technical, functional, and user-centered objectives aimed at transforming traditional college bus transportation systems through modern technology. The specific objectives are:

1. **To Develop a Real-Time Mobile Application**
    - Design and implement a cross-platform mobile application using the **Flutter framework**, ensuring accessibility on both Android and iOS devices.
    - Integrate **GPS technology** and the **Google Maps API** to display real-time location data of college buses with high accuracy.
2. **To Provide Live ETA and Route Details**
    - Utilize GPS data in combination with live traffic conditions, current bus speed, and route distance to calculate and display an accurate **Estimated Time of Arrival (ETA)** for each stop.
    - Continuously update ETA as the bus moves or if traffic conditions change.
3. **To Implement Role-Based Login Modules**
    - Develop **separate, secure login interfaces** for students and drivers.
        - **Student Module**: Allows selection of assigned bus, real-time tracking, and receiving alerts.
        - **Driver Module**: Enables route configuration, live location updates, and communication features.
    - Ensure that the user experience is customized based on login type.
4. **To Integrate Firebase for Secure and Real-Time Data Management**
    - Use **Firebase Authentication** for secure sign-in and role identification.
    - Employ **Firebase Realtime Database or Firestore** to ensure seamless data synchronization across all users and devices.
5. **To Improve Safety, Punctuality, and Communication**
    - Reduce the **idle wait time** at bus stops by providing accurate location and ETA.
    - Ensure **better communication** between students, drivers, and administrators through integrated real-time updates.
    - Enhance **student safety** through constant visibility of bus movements and location

.

6. **To Build a Scalable and Future-Ready Platform**
    - Design a **modular architecture** to allow easy integration of future features such as:
        - **RFID or QR-based attendance systems**
        - **Emergency/SOS features**
        - **Parental access portal**
        - **Machine Learning models** for predicting ETA and identifying transport inefficiencies.
    - Make the application scalable to support **multi-campus institutions** or **district-level education boards**.

3.1 Technical Objectives

- To build the mobile application using **Flutter** for cross-platform support (Android and iOS).
- To implement **Google Maps API** for real-time location display and tracking.
- To use **Firebase** or another suitable backend (like Node.js with MongoDB or PostgreSQL) for:
    - **Authentication** (login/signup for students and drivers).
    - **Storing user data**, bus routes, and live coordinates.
    - **Real-time database or Firestore** for continuous location updates.
- To use **GeoLocation and Background Location services** to capture and transmit the bus driver's location even when the app is minimized.
- To apply **ETA calculation algorithms** using distance and average speed for dynamic time updates.To ensure **scalability and security** in handling user data and bus tracking logic.

CHAPTER 4

4. SYSTEM ANALYSES

4.1 Existing System

The traditional transportation system followed in many university campuses and public or private sectors often relies on manual and outdated methods of communication and tracking. Students or passengers must depend on:

- Printed or static timetables.
- Physical inquiry with transport staff or administration.
- Waiting at designated stops with no real knowledge of whether the bus is on time or delayed.

There is no mechanism for users to view the current position of a bus or receive updates in real-time. Additionally, administrators have no dashboard or centralized control system for managing and monitoring the buses, drivers, or route performance.

In the case of universities, this causes significant problems during peak hours when multiple students depend on timely transportation. Absence of real-time information affects not just convenience, but also punctuality, attendance, and academic performance.

4.1.1 Disadvantages of Existing System

1. **Lack of Real-Time Information**
   Students have no visibility into the actual location of the bus. They may arrive too early or too late at stops.
2. **Increased Waiting Time**
   Due to unpredictable arrival, students often wait long periods at bus stops, resulting in frustration and reduced productivity.
3. **Poor Communication**
   There is no direct or automated communication between the transport department and students regarding delays, cancellations, or route changes.
4. **No Feedback Loop**
   Existing systems do not allow passengers to report issues or provide feedback, making system improvements difficult.

5. **Static Scheduling**
   Timings are fixed, not accounting for dynamic variables like traffic, weather, or special events.
6. **Inefficient Resource Allocation**
   Administrators have no insights into peak demand times, underused routes, or driver behavior, leading to inefficiency.

4.2 Proposed System

The proposed solution is a mobile-based real-time bus tracking and arrival prediction system using GPS, Google Maps API, and Firebase cloud services. This system is designed for both students (users) and transport administrators (admins), offering accurate location updates, notification services, and operational management.

The mobile app helps users to:

- Track live bus movement.
- View estimated time of arrival (ETA).
- Receive alerts and notifications about delays or changes.

For the admin, the system allows:

- Addition and management of buses, drivers, and routes.
- Real-time data analytics and monitoring.
- Communication with users via push notifications.

4.2.1 Advantages of Proposed System

- **Real-Time Tracking and Visibility**
  Users can view live bus locations on a map, making planning more efficient.
- **ETA and Route Updates**
  By integrating with Google Maps API, the system calculates arrival times based on distance and traffic conditions.
- **Improved Student Experience**
  Users can plan accordingly, reducing stress and ensuring punctuality for classes or exams.

- **Admin Control Panel**

  The admin dashboard allows complete control over transportation logistics, improving service quality.
- **Data Analytics**

  Firebase database collects usage and route data, enabling insights for better decision-making.
- **Scalable and Cloud-Based**

  Firebase ensures reliability, scalability, and seamless synchronization across devices.
- **Notification and Alert Mechanism**

  Students are notified of any unexpected changes or delays, improving transparency.

4.2.2 Algorithm

The system uses a combination of **location-based services, cloud storage, and logical conditions** to determine the bus location and estimate its arrival time.

Steps

1. **GPS Location Acquisition:**
   - The driver's app fetches real-time latitude and longitude using the device's GPS.
2. **Location Push to Firebase:**
   - Coordinates are uploaded continuously to Firebase Realtime Database.
3. **Location Retrieval (User Side):**
   - The student app listens for location changes in the database.
   - Coordinates are plotted on Google Maps.
4. **Distance Calculation:**
   - Google Maps API is used to compute the distance and estimated time from bus to the selected stop.
5. **ETA Display:**
   - ETA is shown to the user along with the bus icon on the map.
6. **Notification Trigger:**
   - If the bus is within a certain radius (e.g., 1 km or 5 mins), a notification is sent to the

CHAPTER 5

5. SYSTEM REQUIREMENTS

System requirements define the necessary software and hardware components needed to develop, deploy, and run the application efficiently. This section outlines both **software** and **hardware** prerequisites for the successful execution of the project.

5.1 Software Requirements

This subsection includes all the software tools, frameworks, operating systems, and services required during the development and execution phases of the project.

Component	Description
Operating System	Windows 10 or above / macOS / Linux (Ubuntu 20.04 or later)
IDE/Editor	Visual Studio Code, Android Studio, or any Flutter-supported IDE
Programming Language	Dart (Flutter SDK), optionally Java/Kotlin for Android or Swift for iOS
Backend Platform	Firebase (Authentication, Firestore, Realtime Database, Cloud Functions)
Database	Firebase Firestore or Realtime Database
Maps Integration	Google Maps API for location tracking
API Services	RESTful APIs (Google Maps API, Geolocation API, etc.)
Version Control	Git (GitHub or GitLab for repository hosting)
Testing Tools	Flutter Test, Android Emulator, Firebase Test Lab (Optional)
Deployment	Google Play Store (for Android), Firebase Hosting (if web version needed)

5.2 Hardware Requirements

This subsection covers the necessary physical equipment and minimum specifications required to develop, test, and use the system.

a) ***Development Environment (for developers)***

Hardware Component	Minimum Requirement
Processor	Intel i5 or equivalent quad-core processor
RAM	8 GB or higher
Storage	256 GB SSD (for faster build and load times)
Graphics	Integrated or dedicated GPU with OpenGL 3.2+ support
Display	13” or larger screen with 1920x1080 resolution
Internet	Stable broadband connection for API access, Firebase, and testing

b) ***User Devices (for end-users like students and drivers)***

Device Type	Requirements
Smartphone	Android 8.0+ / iOS 12+ with GPS and internet connectivity
Browser Support (Optional)	Chrome, Firefox, or Safari for accessing any web interface if available
Battery Backup	Sufficient to support location sharing and tracking during commute time

CHAPTER 6

6. TECHNOLOGY STACK

6.1 Frontend

- **Flutter SDK**: Chosen for its cross-platform capabilities, allowing the app to run seamlessly on both Android and iOS devices from a single codebase.
- **Dart Language**: The primary programming language used with Flutter, offering fast performance and reactive UI components.
- **Google Maps Flutter Plugin**: Used to embed maps and render the bus's live location within the app.
- **UI Frameworks and Widgets**:
  - Custom widgets for bus selection, ETA display, and login.
  - Material Design for consistent and intuitive UX.
- **Figma (for prototyping)**: Used during the design phase to visualize and test UI/UX before implementation.

6.2 Backend

- **Firebase Realtime Database / Firestore**:
  - Stores live bus coordinates, student-bus mappings, and route information.
  - Supports real-time syncing of location data.
- **Firebase Authentication**:
  - Handles secure login/signup for both student and driver roles.
  - Role-based access ensures that students and drivers view only relevant data.
- **Cloud Functions (optional)**:
  - Can be used for calculating ETA, sending notifications, or managing route logic asynchronously.
- **Node.js with Express.js**:
  - For building RESTful APIs and managing business logic.
- **Socket.IO**:
  - For real-time location updates using WebSockets.

6.3 Database

- **Firebase Firestore** *(preferred for real-time apps)*:
  - NoSQL cloud-hosted database with real-time sync support.
  - Stores user data, bus metadata, location logs, and route configurations.
- **Alternative: PostgreSQL (for relational structure)**:
  - If switching to a traditional backend, used to handle structured data like users, routes, stops, and travel history.

Example Collections/Tables:

- users (userID, name, email, role)
- buses (busID, routeID, currentLocation, ETA)
- routes (routeID, stops[], timings)
- locations (timestamp, latitude, longitude, busID)

6.4 APIs and Integrations

- **Google Maps API**:
  - Core for live map rendering and marker movement.
  - Provides route paths, directions, and distance calculations.
- **Google Directions API**:
  - For calculating the best path between two points and estimating time of arrival.
- **GeoLocation Plugin**:
  - Captures driver's current location using GPS.
- **Background Location Plugin**:
  - Ensures location updates even when the app is running in the background.
- **Push Notification Service** (e.g., Firebase Cloud Messaging):
  - Sends alerts to users about bus arrival, delays, or emergencies.

6.5 Security Layer

- **Role-Based Access Control (RBAC)**:
  - Restricts data access based on user role (Student, Driver, Admin).
- **Firebase Authentication Security Rules**:
  - Ensures that only authorized users can read/write data in the Firestore database.

- **Encrypted Data Transmission**:
  - All communication between app and backend is done over **HTTPS** to prevent data interception.
- **Data Validation and Sanitization**:
  - Input validation to avoid injection attacks or malformed data entries.
- **Secure API Key Storage**:
  - API keys (e.g., for Google Maps) are stored securely and not hardcoded in the frontend.
- **Session Timeout and Token Expiry**:
  - Auto logout mechanisms to prevent misuse from unattended sessions.

CHAPTER 7

7. Architecture

The Smart College Bus Tracker system is designed to provide real-time location tracking of college buses, enabling students to view accurate arrival times and locations while offering drivers and administrators streamlined control and monitoring capabilities. The system is based on a client-server architecture using cloud-based backend services (Firebase) for real-time communication and data storage.

7.1 System Architecture

The system follows a **three-tier architecture** comprising the **Presentation Layer (Frontend)**, **Application Logic Layer (Backend)**, and **Database Layer**. Here's a breakdown of each layer:

1. Presentation Layer (Frontend)

- Developed using **Flutter** for cross-platform compatibility (Android/iOS).
- Separate interfaces for **Students**, **Drivers**, and **Admin** users.
- Integrates **Google Maps SDK** to display live bus location.
- Interfaces with backend services via REST APIs or Firebase SDK.

2. Application Logic Layer (Backend)

- Built using **Firebase Functions** and **Realtime Database/Firestore**.
- Handles:
    - Location updates from drivers.
    - ETA calculations using distance and speed.
    - Authentication and role-based access (Admin/Driver/Student).
    - Notifications and alerts for delays or arrivals.

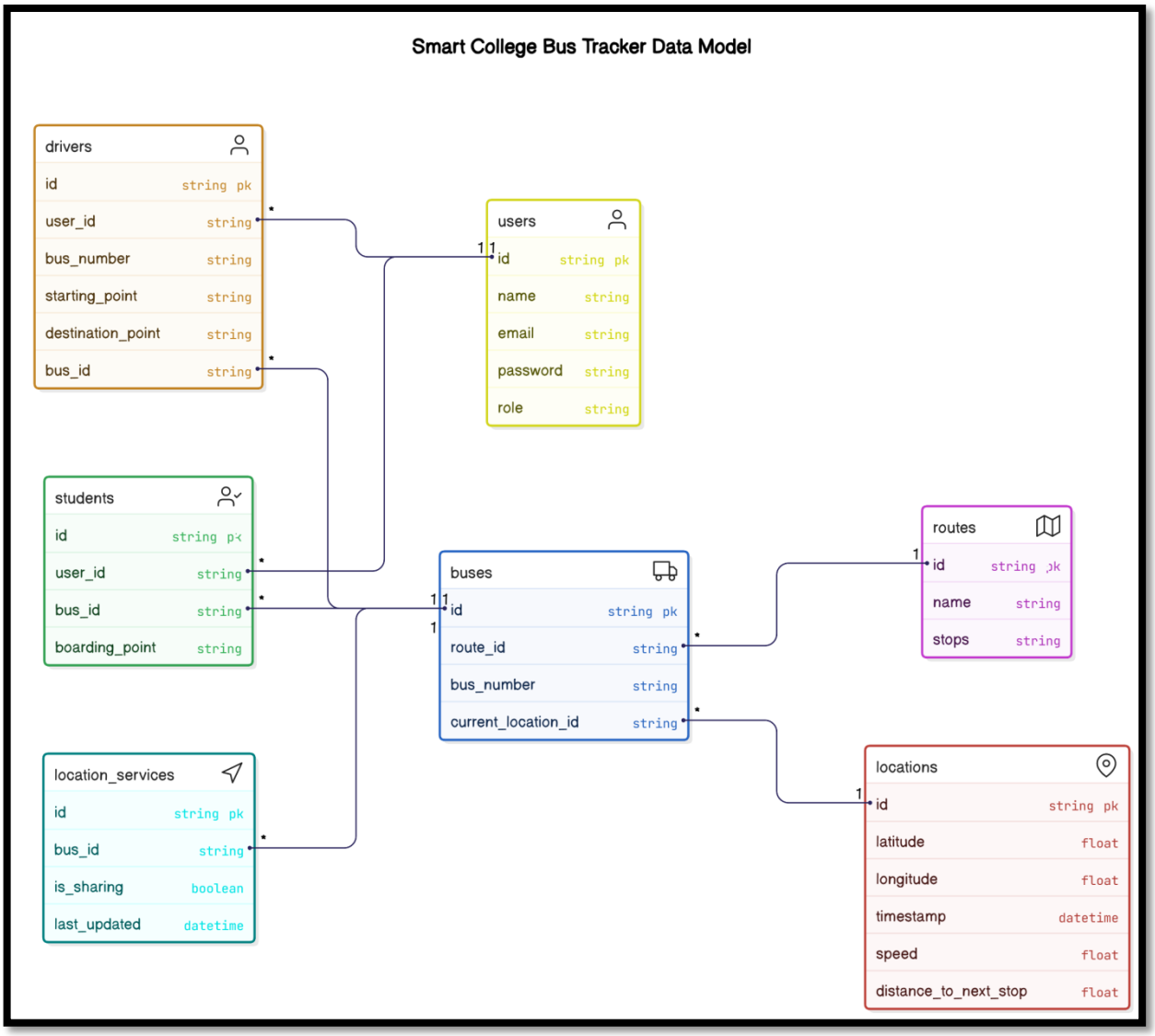
3. Database Layer

- **Firebase Realtime Database** / **Cloud Firestore**:
    - Stores user profiles, route data, bus info, location coordinates.
    - Real-time updates to push changes instantly to connected devices.

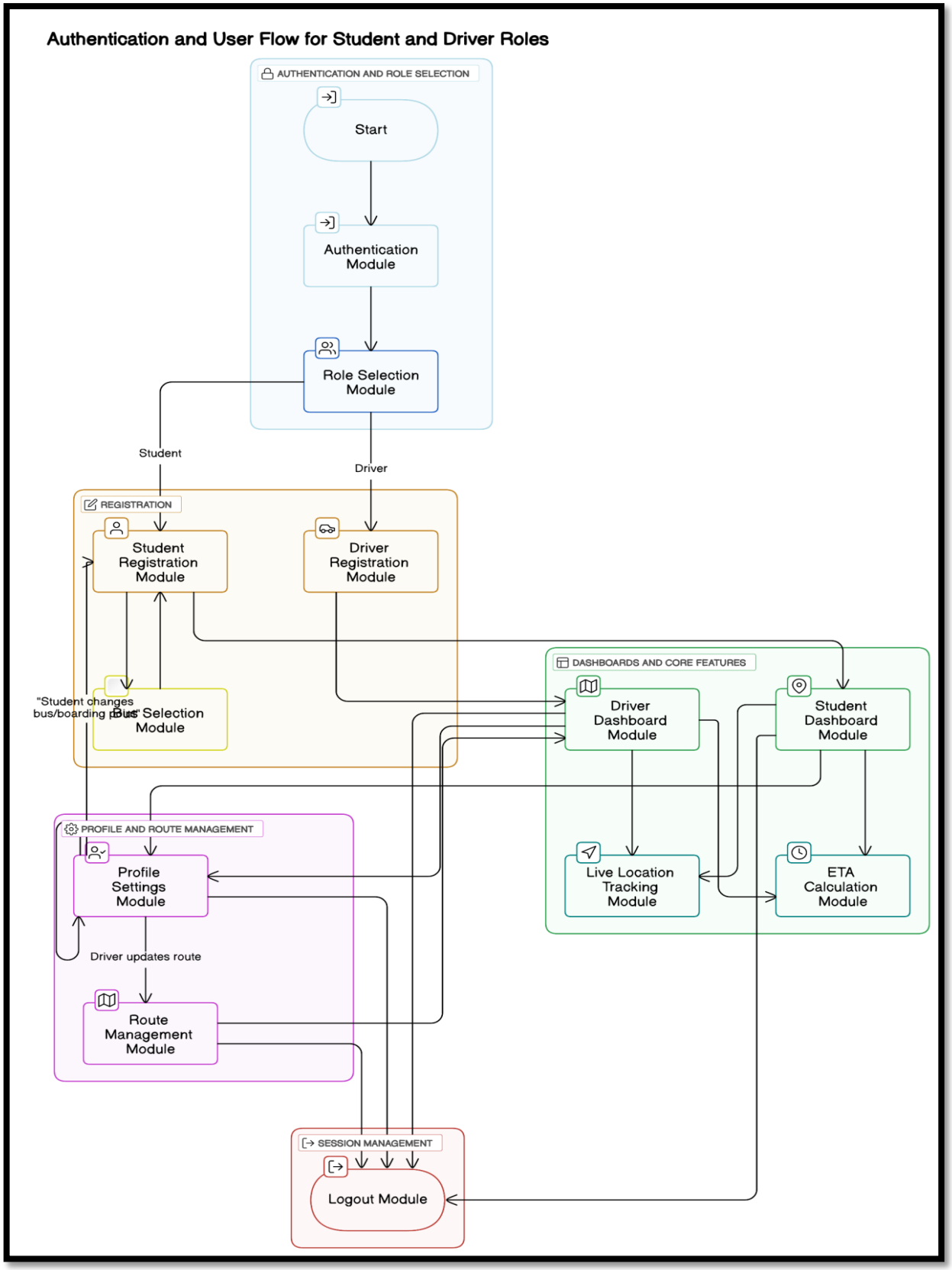
System Communication Flow:

1. Driver logs in and starts sharing GPS coordinates.
2. Location is pushed to Firebase in real-time.
3. Students log in, select their bus, and track its live location.
4. Admin can monitor all buses, add routes, assign drivers, and manage users.
5. All modules are synced with the cloud for consistent performance.

Class diagram



Architecture Diagram



CHAPTER 8

8. METHODOLOGY

The development methodology of the **Smart College Bus Tracker** project is based on a **systematic and iterative software development life cycle (SDLC)**, blending elements from both the **Agile** and **Prototype-based** models to ensure user-focused design, flexible enhancements, and early validation.

1. Requirements Gathering and Analysis

The process began with a **comprehensive requirement analysis** through:

- **Surveys** and **interviews** conducted among students, bus drivers, and college transport staff to understand their current pain points and expectations.
- Identification of core functional needs such as:
    - Real-time bus tracking.
    - Estimated Time of Arrival (ETA).
    - Role-based login systems.
    - Route management and location broadcasting.
    - Secure login and data privacy.

Insights from this phase provided a **clear vision** of what features were essential for a minimum viable product (MVP) and what could be planned as future enhancements.

2. System Design and Planning

The **UI/UX design** and architectural planning phase focused on:

- Creating **low-fidelity wireframes** and **interactive prototypes** using **Figma** to visualize user journeys and interfaces for both student and driver roles.
- Designing a **modular architecture** using the Model-View-Controller (MVC) approach to separate the application's business logic from its UI, ensuring scalability and ease of maintenance.

Key Design Considerations:

- Platform-independence using **Flutter**, enabling a single codebase for both Android and iOS.
- Use of **Firebase** for cloud-based, scalable backend services with real-time data synchronization.

- Selection of **Google Maps API** and **GPS sensors** to provide reliable and accurate bus location and route data.

3. Technology Stack Selection

The chosen tools and technologies ensured fast, reliable, and secure mobile application development:

- **Flutter** for cross-platform mobile app development.
- **Firebase Authentication** for secure and role-specific login (Student, Driver).
- **Firebase Firestore** for storing live location data and route details.
- **Google Maps API** for real-time map display and route visualization.
- **Device GPS Integration** for fetching live bus location coordinates.

4. Implementation and Development

The development process was executed in modules:

- **User Authentication**:
  - Implemented secure login with Firebase for both students and drivers.
  - Role-based redirection ensured appropriate dashboard access for each user type.
- **Student Module**:
  - Option to select their assigned bus.
  - Live map display of the selected bus location.
  - ETA calculation based on speed, traffic conditions, and GPS distance.
- **Driver Module**:
  - Capability to start a trip and broadcast real-time location.
  - Route configuration based on predefined stops.
  - Auto location update intervals to optimize battery and bandwidth usage.
- **Admin (Future Module)**:
  - Design blueprint for an admin dashboard for route, user, and fleet management.

5. Testing and Validation

After development, rigorous **testing phases** were conducted:

- **Functional Testing**: Verified individual features such as login, map rendering, route updates, and real-time location sync.

- **Integration Testing**: Ensured that GPS, Maps API, and Firebase worked smoothly together.
- **Usability Testing**: Conducted trial runs with actual students and drivers on campus to gather user feedback on UI clarity and performance.
- **Performance Testing**: Measured app responsiveness under different network conditions.

Bug reports and improvement suggestions from users were documented, and necessary fixes were applied to improve system stability and usability.

6. Deployment and Pilot Testing

Following successful testing:

- The application was **packaged and deployed** on both **Android (via APK/Play Store)** and **iOS (via TestFlight)** platforms for pilot testing.
- A selected group of students and drivers were involved in **real-world usage** over a defined time period to monitor:
    - Location accuracy.
    - Server load handling.
    - Login and sync reliability.
    - User experience in live conditions.

7. Post-Deployment Review and Future Scope

- Data logs and user interactions were analyzed post-deployment to evaluate app efficiency.
- Feedback helped identify potential enhancements such as:
    - **Push notifications** for delays or rerouting.
    - **RFID integration** for student attendance.
    - **Machine learning models** for predictive ETA based on historical traffic patterns.
    - **Admin web portal** to monitor buses across multiple campuses.

Summary of Methodology in Points:

1. **Requirement Analysis**: Surveys, interviews, and problem analysis.
2. **Design Phase**: UI wireframes in Figma, modular architecture planning.
3. **Tech Stack Finalization**: Flutter, Firebase, Google Maps API, GPS.
4. **Module-Wise Development**: Separate flows for student and driver users.

5. **Real-Time Integration**: Location broadcasting, map updates, ETA logic.
6. **Testing**: Functional, integration, performance, and usability tests.
7. **Pilot Deployment**: Testing in a live environment with feedback collection.
8. **Continuous Improvement**: Planning enhancements based on real usage data.

CHAPTER 9

9. Module Description

The system is divided into three main modules based on user roles: **Student**, **Driver**, and **Admin**. Each module has specific functionalities tailored to the needs of its users.

9.1 Student Module

This module is designed for students who want to track the real-time location and estimated arrival time (ETA) of their respective college buses.

Key Features:

- **User Registration/Login**: Students sign up using college email ID or phone number.
- **Bus Selection**: Students choose their assigned bus or route.
- **Live Location Tracking**: Integrated Google Maps view showing current bus position.
- **ETA Display**: Shows time remaining for the bus to reach the student's stop.
- **Notification Alerts**: Alerts for bus arrival, delays, route changes, or emergencies.
- **Favorites/Recent Routes**: Saves the most used routes or buses for quick access.
- **Feedback System**: Students can rate bus services or report issues.

Benefits:

- Saves time by eliminating unnecessary waiting.
- Enhances safety by knowing bus status and estimated arrival time.
- Improves transparency in transportation services.

9.2 Driver Module

This module is for bus drivers to share their real-time location and communicate route or status updates to both students and administrators.

Key Features:

- **Secure Login**: Driver authentication to access the tracking interface.
- **Start/Stop Tracking**: Option to begin and end location sharing.
- **Live GPS Location Upload**: Sends continuous coordinates to Firebase backend.

- **Route Assignment**: Drivers view and follow assigned routes set by admin.
- **Emergency Button**: Sends alerts in case of breakdowns or incidents.
- **Notifications Panel**: Receives messages or updates from admin (e.g., route change).

Benefits:

- Easy-to-use interface for real-time sharing.
- Improves communication with admin and students.
- Helps maintain route discipline and transparency.

9.3 Admin Module

This module provides a comprehensive dashboard for administrators to manage the entire transport system efficiently.

Key Features:

- **Admin Login Panel**: Secure login using admin credentials.
- **User Management**: Add, remove, or update student and driver accounts.
- **Bus Management**: Add new buses, update details (e.g., capacity, vehicle number).
- **Route Management**: Create, edit, and assign routes to specific buses/drivers.
- **Real-Time Monitoring**: Dashboard showing all active buses on map in real-time.
- **Notification System**: Send alerts to students/drivers (maintenance, delays).
- **Reports & Analytics**: Generate daily/weekly reports on bus usage, punctuality, and feedback.
- **Incident Reporting**: View logs of emergency button triggers and take appropriate action.

Benefits:

- Centralized control of transport operations.
- Ensures student safety and driver accountability.
- Reduces manual errors and increases operational efficiency.

CHAPTER 10

10. TESTING

10.1 Test Plan

The test plan outlines the strategy to validate the functionality, performance, usability, and security of the Smart College Bus Tracker app. The goal is to ensure a stable, bug-free, and user-friendly application that performs efficiently across different devices and user roles (Students, Drivers, Admin).

Testing Objectives:

- To verify that real-time location updates are accurately displayed.
- To ensure seamless login functionality and role-based navigation.
- To validate ETA calculations and route mapping.
- To check responsiveness, map integration, and background location tracking.

Testing Types:

- **Unit Testing**: For individual components such as login logic, map loading, and route selection.
- **Integration Testing**: To verify coordination between Firebase backend, Google Maps API, and UI layers.
- **UI/UX Testing**: Ensures smooth and intuitive interaction for both drivers and students.
- **Real-time Testing**: Assesses performance of location updates and synchronization in real-world moving scenarios.
- **Security Testing**: Verifies authentication, data privacy, and access control.
- **Cross-Platform Testing**: Ensures compatibility on both Android and iOS platforms.

10.2 Test Cases

Test Case ID	Test Case Description	Input	Expected Output	Status
TC001	Login as Student	Email & Password	Redirect to Student Dashboard	Pass

TC002	Login as Driver	Email & Password	Redirect to Driver Dashboard	Pass
TC003	View Live Location	Select Bus	Bus marker appears on map and updates in real-time	Pass
TC004	ETA Calculation	Location + Destination	ETA displayed correctly	Pass
TC005	Invalid Login	Wrong credentials	Display error message	Pass
TC006	Bus Selection	Choose from dropdown	Save student-bus mapping	Pass
TC007	Background Location Update	Minimize driver app	Location updates continue to backend	Pass
TC008	Notification Trigger	Admin sends delay alert	Notification received by students	Pass
TC009	Route Configuration	Driver sets route	Route gets saved and followed	Pass
TC010	Role-Based Access	Student tries to access driver panel	Access denied or redirected	Pass

10.3 Bug Fixes

Throughout development and testing, several bugs were identified and resolved to ensure smooth functioning. Some examples include:

Bug ID	Bug Description	Cause	Fix Implemented
B001	Location not updating in real-time	Incorrect Firestore path	Corrected document reference and enabled real-time sync
B002	App crashes on background tracking	Background permissions not granted	Implemented permission checks and fallback logic
B003	ETA not calculating	Missing API response from Directions API	Fixed API key and added error handling
B004	Login screen freezes	Asynchronous call not awaited	Added proper async/await handling
B005	Student sees wrong bus	Incorrect user-bus mapping logic	Added filter based on user ID and selected bus
B006	Driver cannot save route	Data not stored properly	Validated input and ensured correct Firestore write syntax

CHAPTER 11

11. SOURCE CODE:

Main file (main.dart)

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

import 'package:firebase_core/firebase_core.dart';

import 'package:google_maps_flutter/google_maps_flutter.dart';

import 'package:cloud_firestore/cloud_firestore.dart';

import 'package:geolocator/geolocator.dart';

import 'package:geocoding/geocoding.dart';

import 'bus_selection_screen.dart';

import 'bus_tracking_screen.dart';

import 'DriverDashboardScreen.dart';

import 'student_tracking_screen.dart';

import 'TrackingModeSelectorScreen.dart';

import 'landing_page.dart';

import 'login_screen.dart';

void main() async {

  WidgetsFlutterBinding.ensureInitialized();

  await Firebase.initializeApp(); // □ Initialize Firebase

  runApp(const MyApp());

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override
```

```
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'College Bus Tracker',
      theme: ThemeData(primarySwatch: Colors.deepPurple),
      home: LoginScreen(),
    );
  }
}
```

Student Tracking Screen (StudentTrackingScreen.dart)

```
import 'dart:async';
import 'dart:math';
import 'dart:convert';
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:firebase_database/firebase_database.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';
import 'package:http/http.dart' as http;
import 'package:location/location.dart';
import 'package:flutter_polyline_points/flutter_polyline_points.dart';
import 'package:flutter_image_compress/flutter_image_compress.dart';
import 'package:flutter/services.dart';
```

```
import 'package:geolocator/geolocator.dart';
import 'TrackingModeSelectorScreen.dart';
import 'bus_selection_screen.dart';
class StudentTrackingScreen extends StatefulWidget {
  final String busId;
  final String studentId;
  final bool initialTrackBus;
   const StudentTrackingScreen({
    Key? key,
    required this.busId,
    required this.studentId,
    this.initialTrackBus = false,
  }) : super(key: key);

  @override
  _StudentTrackingScreenState createState() => _StudentTrackingScreenState();
}
class _StudentTrackingScreenState extends State<StudentTrackingScreen> {
  late GoogleMapController _mapController;
  final Completer<GoogleMapController> _controller = Completer();
  late DatabaseReference _busRef;
  late DatabaseReference _studentRef;
  late Timer _updateTimer;
```

```
  Location _location = Location();
  LatLng? _lastBusLocation; // For bearing calculation
 LatLng? _busLocation;
  LatLng? _studentLocation;
  LatLng? _destination;
  List<Map<String, dynamic>> _stations = [];
  String _etaMessage = "";
  Set<Marker> _markers = {};
  Set<Polyline> _polylines = {};
  StreamSubscription<DatabaseEvent>? _busSubscription;
  bool _trackBus = false;
  BitmapDescriptor? _arrowIcon;
 static const String _apiKey = "AIzaSyALIyJ73NuMuXfW29teqEN6YuI-_4y4EPA";
  @override
  void initState() {
    super.initState();
    _trackBus = widget.initialTrackBus;
    _busRef = FirebaseDatabase.instance.ref().child('buses/${widget.busId}');
    _studentRef = FirebaseDatabase.instance.ref().child('students/${widget.studentId}');
_listenForDestinationUpdates();
    _startListeningForBusLocation();


    _loadArrowIcon();
```

```
_updateTimer = Timer.periodic(const Duration(seconds: 5), (timer) {
    if (!_trackBus) {
      _updateStudentLocation();
    }
  });
 _getCurrentLocation();
if (_trackBus) {
    _updateBusLocationOnce(); // □ New line: push current location to Firebase
  }
 }
Future<void> _loadArrowIcon() async {
  final Uint8List arrowBytes =
  await rootBundle.load('assets/arrow.jpeg').then((value) => value.buffer.asUint8List());
  final compressedBytes = await FlutterImageCompress.compressWithList(
    arrowBytes,
    minWidth: 65,
    minHeight: 80,
    quality: 50,
    format: CompressFormat.jpeg,
  );
  setState(() {
    _arrowIcon = BitmapDescriptor.fromBytes(compressedBytes);
  });
```

```
}

  Future<void> _getCurrentLocation() async {

  LocationData locationData = await _location.getLocation();

  setState(() {

    _studentLocation = LatLng(locationData.latitude!, locationData.longitude!);

  });

  _updateMap();

  _fetchETA();

  _getRoute();

}


Future<void> _updateStudentLocation() async {

  LocationData locationData = await _location.getLocation();

  LatLng newLocation = LatLng(locationData.latitude!, locationData.longitude!);


  setState(() {

    _studentLocation = newLocation;

  });


  await _studentRef.update({

    'latitude': newLocation.latitude,

    'longitude': newLocation.longitude,

  });
```

```
 _updateMap();
  _getRoute();
  _fetchETA();

}
Future<void> _updateBusLocationOnce() async {
  try {
    final locationData = await _location.getLocation();
    LatLng currentLoc = LatLng(locationData.latitude!, locationData.longitude!);

    await _busRef.update({
      'latitude': currentLoc.latitude,
      'longitude': currentLoc.longitude,
    });

    setState(() {
      _busLocation = currentLoc;
    });

    _updateMap();
    _getRoute();
    _fetchETA();
  } catch (e) {
```

```
    print('Failed to update bus location: $e');
  }
}
void _startListeningForBusLocation() {
  _busSubscription = _busRef.onValue.listen((event) {
    if (!event.snapshot.exists) return;

    Map<String, dynamic> data = Map<String, dynamic>.from(event.snapshot.value as Map);
    LatLng newBusLocation = LatLng(data["latitude"], data["longitude"]);
    double bearing = 0;
    if (_lastBusLocation != null) {
      bearing = Geolocator.bearingBetween(
        _lastBusLocation!.latitude, _lastBusLocation!.longitude,
        newBusLocation.latitude, newBusLocation.longitude,
      );
    }
    _lastBusLocation = newBusLocation;

    setState(() {
      _busLocation = newBusLocation;
    });

    _updateMap();
```

```
    _getRoute();
    _fetchETA();

  }, onError: (error) {
    print("□ Error fetching bus location: $error");
  });
}
/// Replaces your one-time fetch. Listens for real-time updates!
/// Listen to the full bus node to get both destination and stations
void _listenForDestinationUpdates() {
  FirebaseDatabase.instance
      .ref('buses/${widget.busId}')
      .onValue
      .listen((event) {
    final raw = event.snapshot.value;
    print("□ Raw bus snapshot: $raw");
    if (raw == null) {
      print("□ Bus node is null or missing!");
      return;
    }
final busData = Map<String, dynamic>.from(raw as Map);

    // 1) Parse destination
```

```
if (busData['destination'] == null) {
  print("□ 'destination' key is missing on bus node!");
  return;
}
final destMap = Map<String, dynamic>.from(busData['destination']);
final newDestination = LatLng(destMap['latitude'], destMap['longitude']);

// 2) Parse stations (siblings of destination)
if (busData['stations'] == null) {
  print("□ 'stations' key is missing on bus node!");
  return;
}
// □ CORRECT: stations is a List of Maps
final rawStations = busData['stations'] as List<dynamic>;
final stationsList = rawStations
    .map((s) => Map<String, dynamic>.from(s as Map))
    .toList();
print("□ Loaded stationsList: $stationsList");

// 3) Update state
setState(() {
  _destination = newDestination;
  _stations    = stationsList;
```

```
      });

      _updateMap();
    }, onError: (e) {
      print("□ Error listening to buses/${widget.busId}: $e");
    });
  }

  Future<void> _getRoute() async {
    if ((_trackBus && _busLocation == null) || (!_trackBus && _studentLocation == null) || _destination
== null) return;

    LatLng start = _trackBus ? _busLocation! : _studentLocation!;
    LatLng end = _destination!;

    // OSRM API URL
    final Uri url = Uri.parse(
        "https://router.project-osrm.org/route/v1/driving/"
            "${start.longitude},${start.latitude};"
            "${end.longitude},${end.latitude}"
            "?overview=full&geometries=geojson"
    );

    final response = await http.get(url);
```

```
final data = jsonDecode(response.body);

if (response.statusCode == 200 && data["routes"] != null && data["routes"].isNotEmpty) {
  final geometry = data["routes"][0]["geometry"];
  List coordinates = geometry["coordinates"];

  List<LatLng> polylinePoints = coordinates.map<LatLng>((coord) {
    return LatLng(coord[1], coord[0]); // [lng, lat] => LatLng
  }).toList();

  setState(() {
    _polylines = {
      Polyline(
        polylineId: const PolylineId("route"),
        color: Colors.blue,
        width: 5,
        points: polylinePoints,
      ),
    };
  });
} else {
  print("□ Error fetching route: ${response.body}");
}
```

```
  }

  Future<void> _fetchETA() async {

    if ((_trackBus && _busLocation == null) || (!_trackBus && _studentLocation == null) || _destination
== null) return;


    LatLng start = _trackBus ? _busLocation! : _studentLocation!;

    LatLng end = _destination!;

    // OSRM API URL

    final Uri url = Uri.parse(

        "https://router.project-osrm.org/route/v1/driving/"

            "${start.longitude},${start.latitude};"

            "${end.longitude},${end.latitude}?overview=false"

    );


    try {

      final response = await http.get(url);


      if (response.statusCode == 200) {

        final data = jsonDecode(response.body);


        if (data.containsKey("routes") && data["routes"].isNotEmpty) {

          int durationInSeconds = data["routes"][0]["duration"].toInt();

          int minutes = (durationInSeconds / 60).round();

          if (minutes == 0) minutes = 1; // Prevents showing 0 min
```

```
        setState(() => _etaMessage = "ETA: $minutes min");
      } else {
        print("□ Error: No routes found in response.");
      }
    } else {
      print("□ Error fetching ETA: ${response.statusCode}");
    }
  } on SocketException {
    print("□ No internet connection");
  } on HttpException {
    print("□ HTTP error");
  } on FormatException {
    print("□ Invalid JSON format");
  } catch (e) {
    print("□ Exception: $e");
  }
}
@override
void dispose() {
  _busSubscription?.cancel();
  _updateTimer?.cancel();
  super.dispose();
```

```
}
void _updateMap() {
  Set<Marker> updatedMarkers = {};

  // Add Bus Marker
  if (_trackBus && _busLocation != null) {
    updatedMarkers.add(
      Marker(
        markerId: const MarkerId("bus"),
        position: _busLocation!,
        icon: _arrowIcon ?? BitmapDescriptor.defaultMarkerWithHue(BitmapDescriptor.hueViolet),
        rotation: _lastBusLocation != null
            ? Geolocator.bearingBetween(
          _lastBusLocation!.latitude,
          _lastBusLocation!.longitude,
          _busLocation!.latitude,
          _busLocation!.longitude,
        )
            : 0,
        anchor: const Offset(0.5, 0.5),
      ),
    );
  } else if (!_trackBus && _studentLocation != null) {
```

```
      updatedMarkers.add(
        Marker(
          markerId: const MarkerId("student"),
          position: _studentLocation!,
          icon: BitmapDescriptor.defaultMarkerWithHue(BitmapDescriptor.hueBlue),
        ),
      );
    }
  // Add Destination Marker
    if (_destination != null) {
      updatedMarkers.add(
        Marker(
          markerId: const MarkerId("destination"),
          position: _destination!,
          icon: BitmapDescriptor.defaultMarkerWithHue(BitmapDescriptor.hueGreen),
        ),
      );
    }
  setState(() {
      _markers = updatedMarkers;
    });
   // Move camera to latest location
    if (_controller.isCompleted) {
```

```
    _controller.future.then((controller) {
      LatLng target = _trackBus ? _busLocation ?? _studentLocation! : _studentLocation!;
      controller.animateCamera(CameraUpdate.newLatLng(target));
    });
  }
}
/// Builds the horizontal station timeline, highlighting the next stop
Widget _buildStationTimeline() {
  if (_stations.isEmpty) {
    return const Padding(
      padding: EdgeInsets.all(16.0),
      child: Text('No stations available'),
    );
  }
  int nextStationIndex = -1;
  if (_busLocation != null) {
    double minDistance = double.infinity;
    for (int i = 0; i < _stations.length; i++) {
      final s = _stations[i];
      double d = Geolocator.distanceBetween(
        _busLocation!.latitude,
        _busLocation!.longitude,
        s['latitude'],
```

```
        s['longitude'],
      );
      if (d < minDistance) {
        minDistance = d;
        nextStationIndex = i;
      }
    }
  }
  return SizedBox(
    height: 100,
    child: ListView.builder(
      scrollDirection: Axis.horizontal,
      itemCount: _stations.length,
      itemBuilder: (context, index) {
        final station = _stations[index];
        final bool isNext = index == nextStationIndex;
        final bool isReached = _busLocation != null && index < nextStationIndex;
        return Row(
          children: [
            Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Container(
```

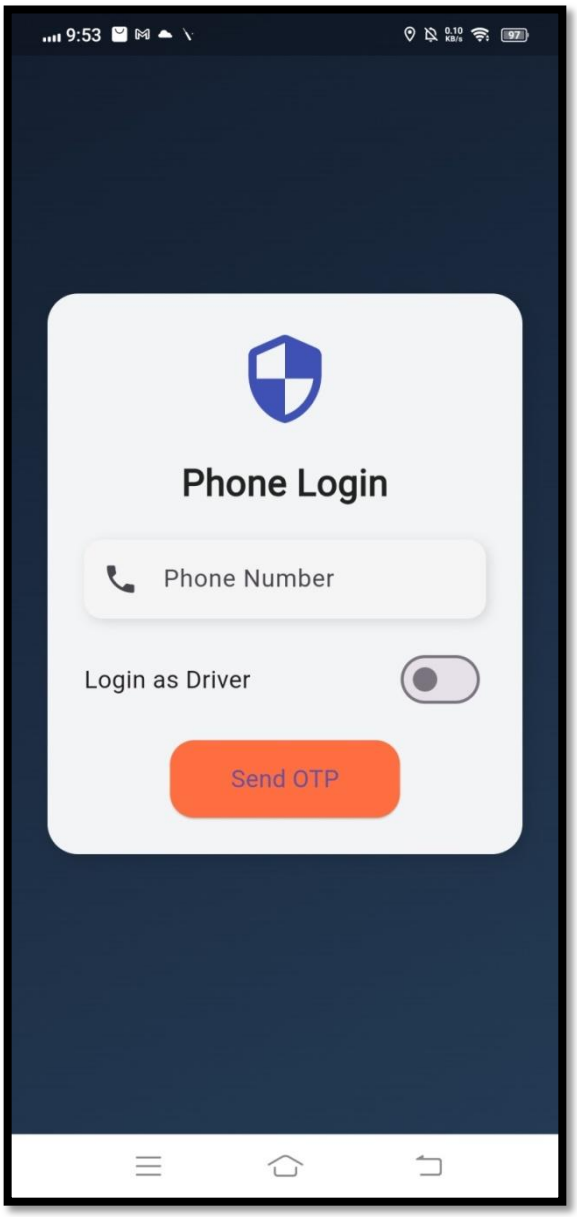
```
        width: 16,
        height: 16,
        decoration: BoxDecoration(
          color: isNext
              ? Colors.orange
              : (isReached ? Colors.blue : Colors.grey),
          shape: BoxShape.circle,
        ),
      ),
      const SizedBox(height: 4),
      Text(
        station['name'],
        style: TextStyle(
          fontSize: 12,
          fontWeight: isNext ? FontWeight.bold : FontWeight.normal,
        ),
      ),
    ],
  ),
  if (index != _stations.length - 1)
    Container(
      width: 40,
      height: 2,
```

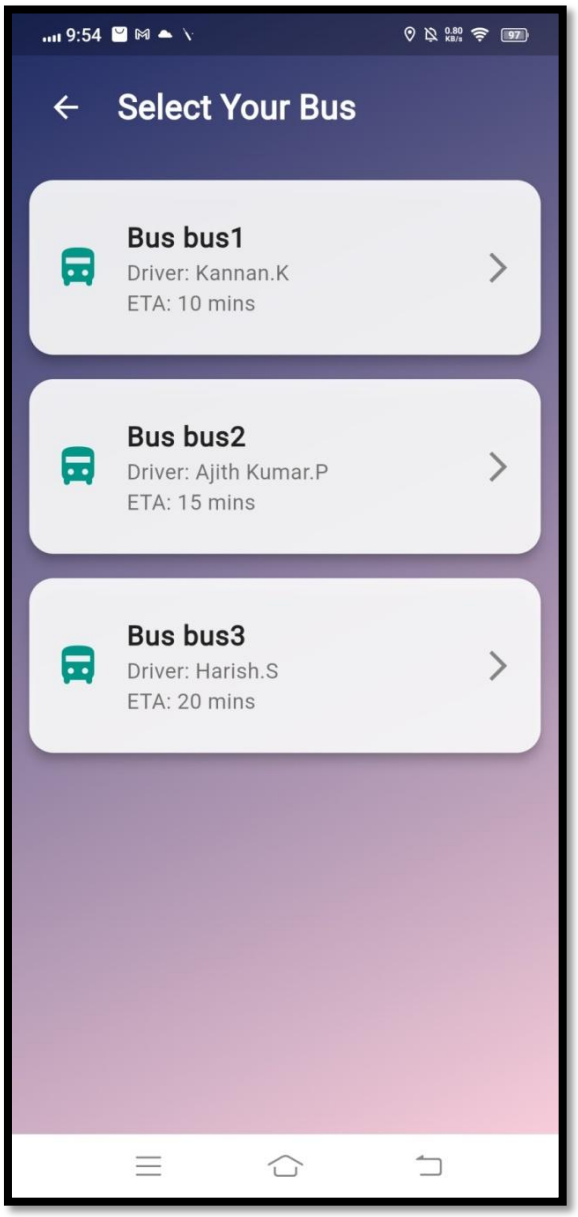
```
                color: index < nextStationIndex ? Colors.blue : Colors.grey,
              ),
            ],
          );
        },
      ),
    );
  }
  /// Wraps the timeline in a Card
  Widget _buildStationTimelineCard() {
    return Card(
      margin: const EdgeInsets.all(8),
      elevation: 4,
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(12)),
      child: Padding(
        padding: const EdgeInsets.symmetric(vertical: 10, horizontal: 6),
        child: _buildStationTimeline(),
      ),
    );
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

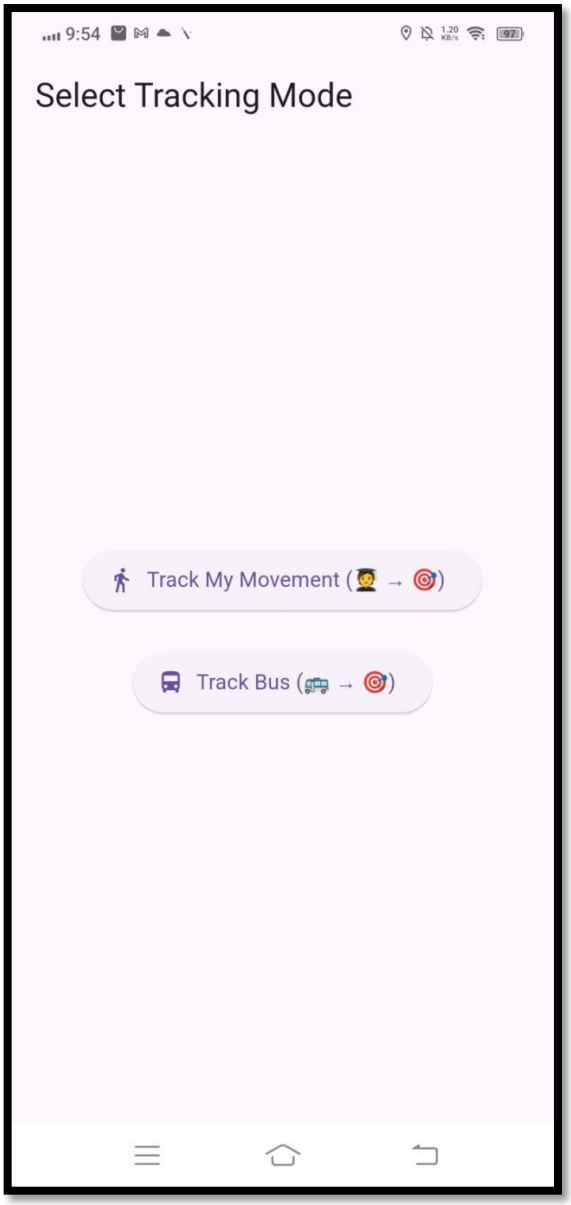
```
      appBar: AppBar(title: const Text("Bus Tracking"), backgroundColor: Colors.blue),
      body: Column(
        children: [
          _buildStationTimelineCard(),  // □ Add this above everything for station timeline
          Text(_trackBus ? "Tracking Bus (□ → □)" : "Tracking Student (□□ → □)", style:
TextStyle(fontSize: 16, fontWeight: FontWeight.bold)),
          Text(_etaMessage, style: TextStyle(fontSize: 16)),
          Expanded(
            child: GoogleMap(
              initialCameraPosition: const CameraPosition(target: LatLng(0, 0), zoom: 15),
              markers: _markers,
              polylines: _polylines,
              onMapCreated: (controller) {
                _controller.complete(controller);
                _mapController = controller;
              },
            ),
          ),
        ],
      ),
    );
  }
}
```

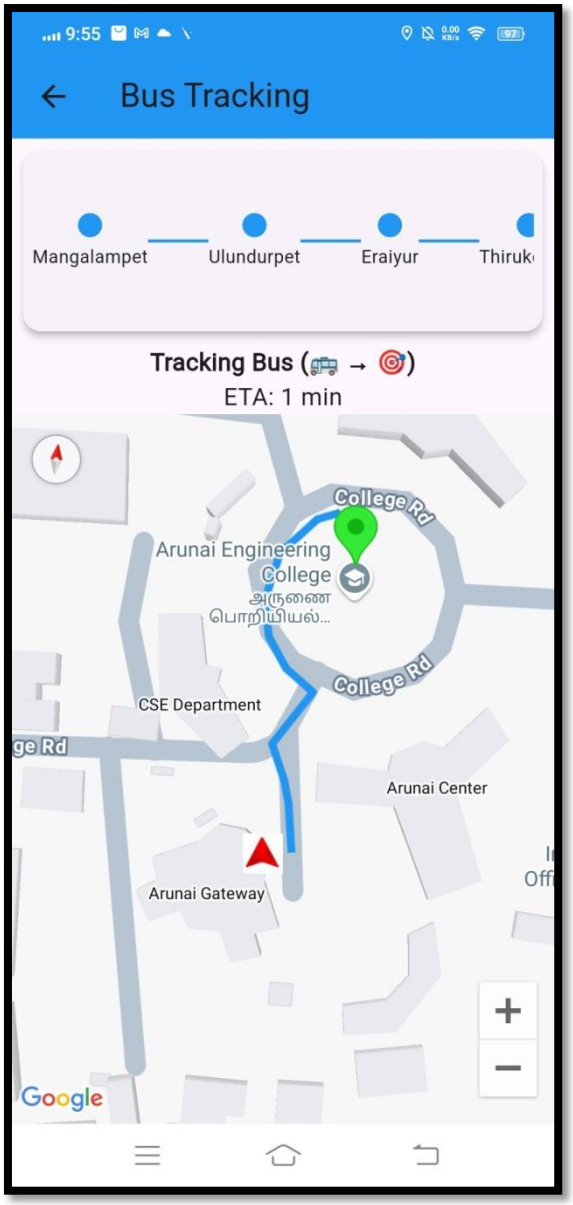
CHAPTER 12

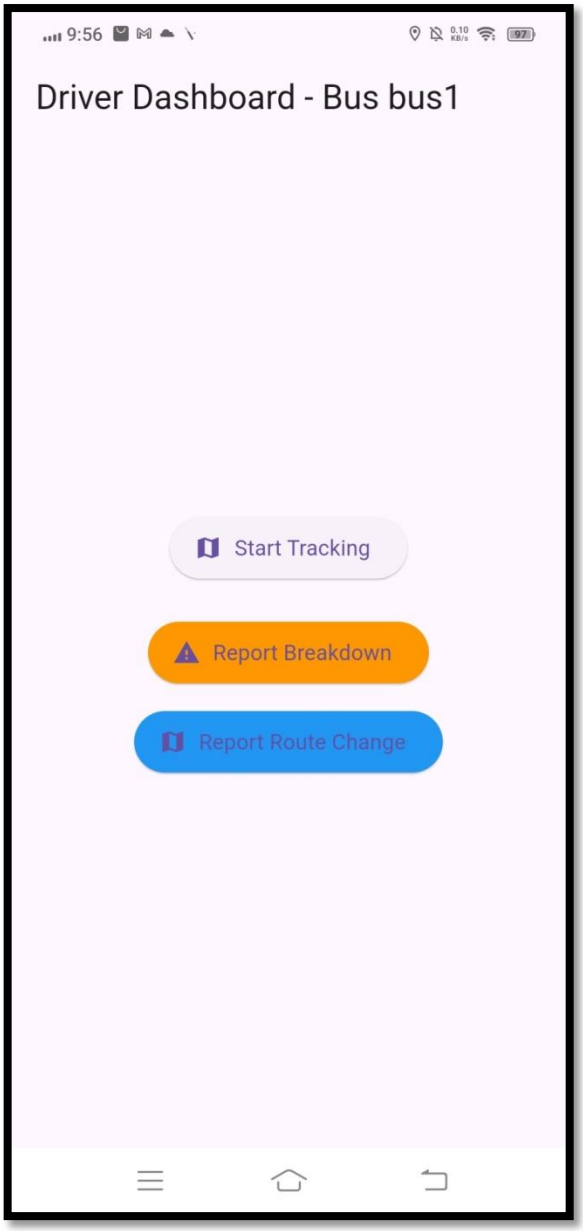
12. OUTPUT:

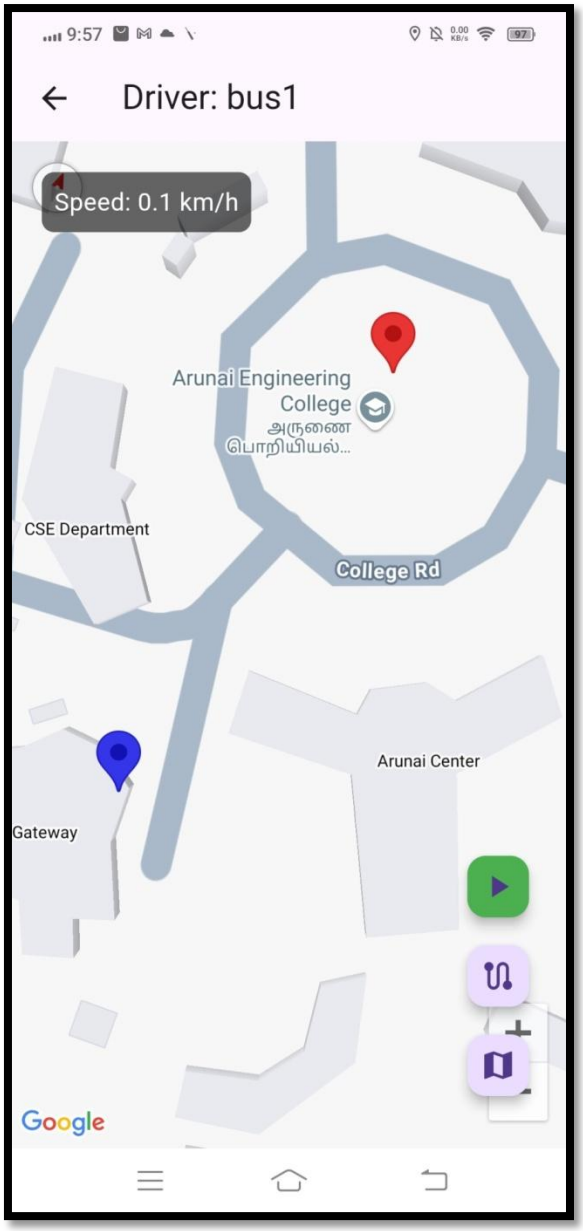


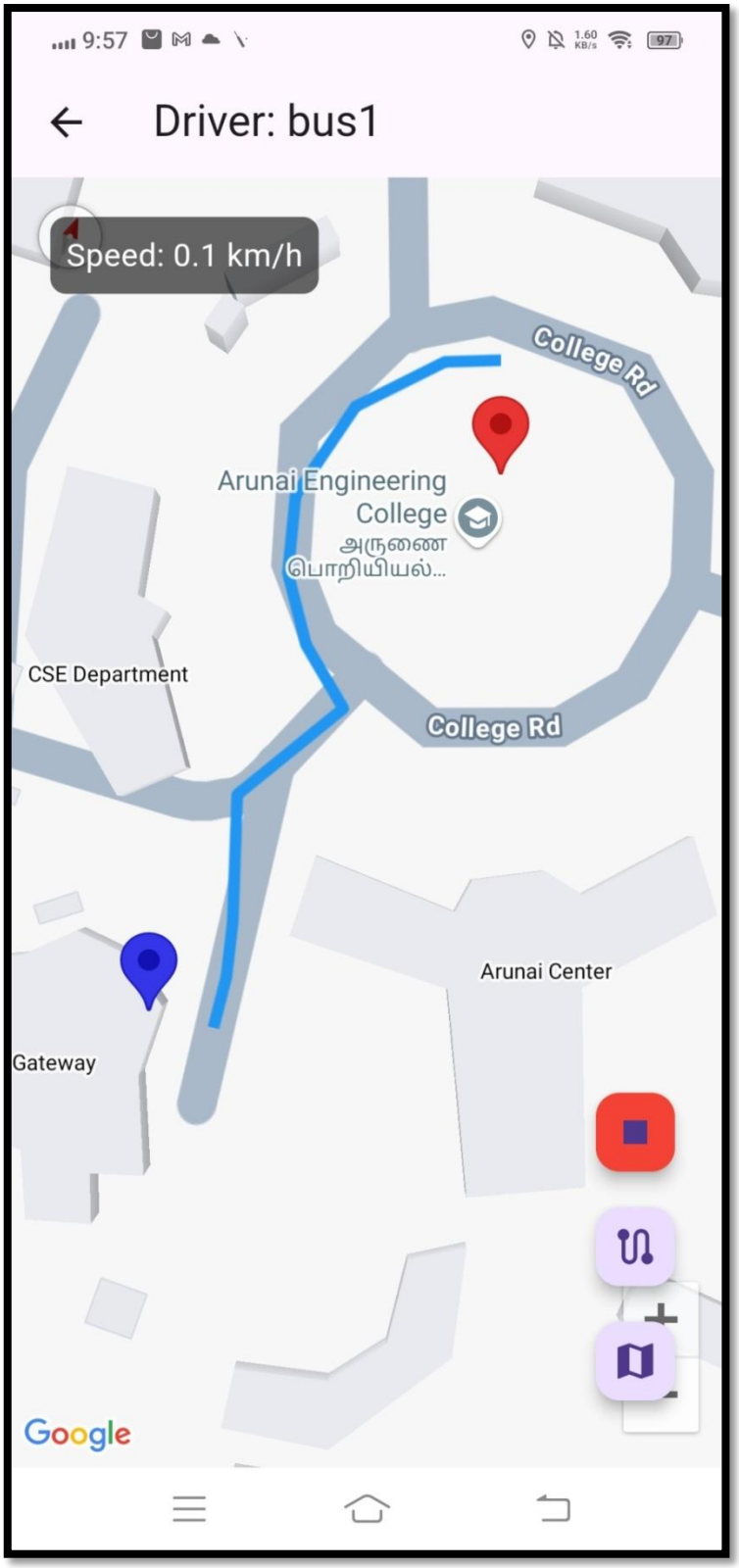



9:54
Select Tracking Mode
Track My Movement (🧑‍🦱 → 🎯)
Track Bus (🚌 → 🎯)


9:55
Bus Tracking
Mangalampet
Ulundurpet
Eraiyur
Thiruk
Tracking Bus (🚌 → 🎯)
ETA: 1 min
College Rd
Arunai Engineering College
அருணை பொறியியல்...
College Rd
CSE Department
ge Rd
Arunai Center
Arunai Gateway
Google


9:56
Driver Dashboard - Bus bus1
Start Tracking
Report Breakdown
Report Route Change


9:57
Driver: bus1
Speed: 0.1 km/h
Arunai Engineering College
அருணை பொறியியல்...
CSE Department
College Rd
Arunai Center
Gateway
Google


9:57
Driver: bus1
Speed: 0.1 km/h
College Rd
Arunai Engineering College
அருணை பொறியியல்...
CSE Department
College Rd
Arunai Center
Gateway
Google

CHAPTER 13

13. APPLICATIONS

13.1 Real-Time Use Cases

The Smart College Bus Tracker app offers real-time applications that enhance transportation convenience, safety, and communication within a college environment. Here are the primary use cases:

1. Live Bus Tracking

- Students can view the **live location** of their assigned college bus on a map using GPS data.
- Eliminates uncertainty and unnecessary waiting at stops, especially during delays or bad weather.

2. Estimated Time of Arrival (ETA)

- Students receive dynamic ETAs based on current traffic, speed, and route conditions.
- Allows students to plan their time better and avoid missed buses.

3. Driver Route Navigation

- Drivers can view their assigned routes and stops through the app interface.
- The app ensures that drivers stick to the correct paths and maintain proper timing.

4. Admin Monitoring

- Admins can monitor all running buses in real-time, track delays, and manage routes if needed.
- Useful for optimizing fleet performance and ensuring punctuality.

5. Alert & Notification System

- Real-time alerts can be sent to students in case of **route changes, breakdowns, or delays**.
- Can also be used during emergencies to notify all stakeholders quickly.

6. Attendance or Boarding Confirmation (Optional Feature)

- Future versions of the app may include automatic student check-ins when they board the bus via GPS geofencing or QR scan, aiding attendance tracking.

13.2 Benefits to Stakeholders

This project directly benefits multiple groups within the college ecosystem. Below is a breakdown:

A. Students

- **Reduced Waiting Time**: Know exactly when the bus will arrive, avoiding long wait times.
- **Peace of Mind**: Real-time updates assure students of bus location and route.
- **Ease of Use**: Simple interface and real-time data improve daily commuting experience.
- **Time Management**: Accurate ETAs help students manage their morning routines better.

B. Drivers

- **Navigation Assistance**: Route guidance and stop information help new or substitute drivers.
- **Communication Simplified**: Reduces the need to call students/admins for updates.
- **Performance Tracking**: Enables better monitoring of timeliness and routes taken.

C. College Administration

- **Operational Efficiency**: Real-time insights into the location and status of the entire bus fleet.
- **Data-Driven Decisions**: Can analyze route usage, punctuality, and optimize bus deployment.
- **Enhanced Safety**: Increases accountability and ensures safer student transportation.
- **Scalability**: Easily extendable to additional buses, routes, or campuses.

D. Parents (Optional Stakeholder)

If future versions include parental access, they can track their child's bus**, enhancing peace of mind regarding safety and punctuality.**

CHAPTER 14

14. FUTURE ENHANCEMENTS

The current version of the Smart College Bus Tracker app effectively supports real-time bus tracking, enhancing student convenience and safety. However, to future-proof the system and align with evolving user expectations, the following advanced features can be integrated:

1. RFID or QR-Based Boarding System

- **How It Works**: Students will scan a unique QR code or tap an RFID card when they board or exit the bus.
- **Benefits**:
    - **Automated Attendance**: Automatically logs each student's bus usage.
    - **Parental Notifications**: Instantly alerts parents when their child boards or leaves the bus.
    - **Enhanced Security**: Prevents unauthorized individuals from using the bus service.
    - **Data Analytics**: Provides insights into student travel patterns and bus occupancy rates.

2. Parental Access Portal

- **How It Works**: A dedicated login interface (web or mobile) for parents linked to their child's account.
- **Features**:
    - View real-time bus location and estimated time of arrival (ETA).
    - Receive instant alerts for boarding, delays, or emergencies.
    - Communicate with transport authorities in case of issues.
- **Benefits**:
    - Peace of mind for parents.
    - Transparency and trust in college transportation.

3. AI-Based ETA Prediction

- **How It Works**: Leverages Machine Learning to analyze historical data (traffic trends, weather conditions, driver patterns) for more accurate ETA.
- **Benefits**:
    - Increased reliability of arrival time predictions.
    - Adaptive learning that improves accuracy over time.

- Helps students plan better and reduces waiting time.

4. Offline Mode Support

- **How It Works**: Stores route and map data locally on the device to ensure functionality during poor or no internet connectivity.
- **Benefits**:
  - Tracking remains uninterrupted in areas with weak network coverage.
  - Critical safety features like SOS or attendance still function in offline mode.
  - Syncs data automatically when internet is restored.

5. In-App Chat or SOS Button

- **Features**:
  - **Live Chat**: Allows students to communicate with drivers or support staff in case of delays, confusion, or lost items.
  - **SOS Button**: Sends emergency alert with live location to college authorities/security in case of distress.
- **Benefits**:
  - Quick support during emergencies.
  - Adds a layer of student safety and assurance.
  - Reduces communication gaps.

6. Multi-Campus and Multi-Route Support

- **How It Works**: The system supports route mapping and tracking for multiple campuses and interlinked transport networks.
- **Features**:
  - Custom route configuration for each campus/zone.
  - Assign buses to different routes and shifts dynamically.
  - Route-specific analytics (occupancy, punctuality, incidents).
- **Benefits**:
  - Scalability for institutions with larger transport fleets.
  - Flexible route planning and optimization.
  - Centralized management of all campuses.

7. Admin Dashboard (Web-Based)

- **How It Works**: A browser-accessible dashboard for transport administrators.
- **Features**:
  - View real-time bus locations, driver profiles, and route status.
  - Analytics on punctuality, route efficiency, and student attendance.
  - Add or modify routes, assign drivers, and broadcast messages.
- **Benefits**:
  - Centralized transport operations.
  - Easier monitoring and data-driven decision-making.
  - Enhanced response to on-ground issues or emergencies.

CHAPTER 15

15. REFERENCES

[1] Google Developers, "Google Maps Platform Documentation," [Online]. Available: https://developers.google.com/maps/documentation. [Accessed: 01-May-2025].

[2] Firebase, "Firebase Documentation – Authentication, Firestore, Realtime Database," [Online]. Available: https://firebase.google.com/docs. [Accessed: 01-May-2025].

[3] Flutter Team, "Flutter – Beautiful Native Apps in Record Time," [Online]. Available: https://docs.flutter.dev. [Accessed: 01-May-2025].

[4] Dart Dev Team, "Dart Language Guide," [Online]. Available: https://dart.dev/guides. [Accessed: 01-May-2025].

[5] Baseflow, "Geolocator Plugin for Flutter," [Online]. Available: https://pub.dev/packages/geolocator. [Accessed: 01-May-2025].

[6] Lyokone, "Location Plugin for Flutter," [Online]. Available: https://pub.dev/packages/location. [Accessed: 01-May-2025].

[7] Firebase, "Firebase Cloud Messaging (FCM)," [Online]. Available: https://firebase.google.com/docs/cloud-messaging. [Accessed: 01-May-2025].

[8] M. Wazid, A. K. Das, and V. Odelu, "Real-Time Vehicle Tracking System using GPS and GSM," *International Journal of Computer Applications*, vol. 975, no. 8887, pp. 1–5, 2017.

[9] OWASP Foundation, "Mobile App Security Best Practices – OWASP Top 10," [Online]. Available: https://owasp.org/www-project-mobile-top-10/. [Accessed: 01-May-2025].

[10] Android Developers, "Android Developer Guide," [Online]. Available: http://developer.android.com/. [Accessed: 01-May-2025].

[11] H. Cohen, "Mobile Marketing: 56 Must Have Facts," [Online]. Available: http://heidicohen.com/mobile-marketing-must-have-facts/. [Accessed: 01-May-2025].

[12] Eclipse Foundation, "About the Eclipse Foundation," [Online]. Available: http://www.eclipse.org/org/. [Accessed: 01-May-2025].

[13] Fleishman Hillard, “2012 Digital Influence Study,” [Online]. Available: http://www.factbrowser.com/facts/4671/. [Accessed: 01-May-2025].

[14] N.-E. Frantzell, “An Introduction to SQLite, an Open Source Embeddable Database,” IBM DeveloperWorks, [Online]. Available: http://www.ibm.com/developerworks/opensource/library/os-sqlite/. [Accessed: 01-May-2025].

[15] GeoNames, “About GeoNames,” [Online]. Available: http://www.geonames.org/about.html. [Accessed: 01-May-2025].

[16] W.-C. Hu, T. Wiggen, and H.-J. Yang, “Mobile Commerce Systems and Payment Methods,” in *Proc. 2005 Information Resources Management Association (IRMA) Int. Conf.*, San Diego, pp. 769–771, May 2005.

[17] JiWire, “Mobile Audience Insights Report Q2 2011,” [Online]. Available: http://www.factbrowser.com/facts/3196/. [Accessed: 01-May-2025].

[18] K. Kolodziej and J. Hjelm, *Local Positioning Systems: LBS Applications and Services*, CRC Press, 2006.