Bus Tracking and Monitoring System

Prof. Dr. Girish Kotwal[1], Chaitanyaa Deokar[2], Mandar Deotale[3], Aryan Desai[4], Purva Daundkar[5], Aftab Desai[6], Samyak Dawale[7]
[1]*Industrial & Prod. Engg. Department Vishwakarma Institute of Technology, Pune, India*
[2,3,4,5,6,7] *Department of Engineering of Sciences and Humanities Vishwakarma Institute of Technology, Pune, India*

Abstract — ***The Bus Tracking and Monitoring App is an innovative solution designed to simplify the process of navigating public transportation by providing real-time bus location tracking and optimized route planning. This project addresses common commuter challenges by offering two main features: route optimization and real-time bus tracking. The route optimization feature assists users in finding the most efficient route between two points, taking into account direct and indirect routes, costs, estimated arrival times, and total journey duration. The real-time bus tracking feature utilizes a web application and Firestore database to continuously update and display bus locations on an interactive map. This enables commuters to plan their journeys effectively and avoid the frustration of waiting for buses that may not arrive on time. By entering their starting point and destination, users receive the best route options tailored to their preferences. They can also access real-time information on bus locations and estimated arrival times, empowering them to make informed travel decisions and avoid unnecessary delays. The app aims to enhance the commuting experience by reducing wait times and improving reliability. Through its user-friendly interface, the Bus Tracking and Monitoring App offers a convenient and efficient way for commuters to navigate public transportation systems. In conclusion, by integrating route optimization with real-time tracking, this app significantly improves the efficiency and convenience of public transit, ultimately leading to a more streamlined and satisfactory commuting experience.***

Key Words: Route Optimization, Bus tracking, Real-time bus tracking, Firebase.

I. INTRODUCTION

1. The Bus Tracking and Monitoring App is designed to simplify the process of navigating public transportation by providing commuters with tools to find the most efficient bus routes and track bus locations in real-time. This project aims to address common challenges faced by public transit users, offering a more convenient and efficient way to plan journeys.

2. The application offers two primary features:

- Route optimization
- Real-time bus tracking.

The route optimization feature helps users determine the best route between two points by considering factors such as direct and indirect routes, costs, estimated time of arrival, and overall journey time. The real-time bus tracking feature utilizes a web application and Firestore database to continuously update and display bus locations on an interactive map interface.

3. By providing real-time information on bus locations, the app enables commuters to plan their journeys effectively, avoiding the frustration of waiting for buses that may not arrive on schedule. Users simply enter their starting point and destination, and the app generates the best route options based on their preferences. Real-time data on bus locations and estimated arrival times further enhances the user experience, allowing for more informed travel decisions.

4. The Bus tracking and monitoring app significantly enhances the commuting experience by reducing wait times and improving the reliability of public transportation. By integrating route optimization with real-time tracking, the app offers a user-friendly solution that empowers commuters to navigate public transportation systems more efficiently, ultimately leading to a more streamlined and satisfactory travel experience.

II. METHODOLOGY

1. Initialization of StopToRoutes Map:
- Populate the stopToRoutes map to associate each bus stop with a list of bus routes passing through it.

```
public static void initializeRoutes() {
        String[][] busRoutes = {
                {"Bus 1", "Pune Station -> Shivajinagar -> FC Road -> JM Road -> Deccan", "30", "20"},
                {"Bus 2", "Swargate -> Katraj -> Sinhgad Road -> Warje", "45", "25"},
                {"Bus 3", "Kothrud -> Karve Nagar -> Chandani Chowk -> Bavdhan", "40", "30"},
                {"Bus 4", "Hadapsar -> Magarpatta -> Kharadi -> Vishrantwadi", "50", "25"},
                {"Bus 5", "Wakad -> Hinjewadi -> Balewadi -> Aundh -> University Circle", "60", "35"},
                {"Bus 6", "Katraj -> Market Yard -> Camp -> Koregaon Park -> Kalyani Nagar", "55", "30"},
                {"Bus 7", "Nigdi -> Akurdi -> Pimpri -> Chinchwad -> Bhosari", "40", "20"},
                {"Bus 8", "Bibvewadi -> Katraj -> Dhankawadi -> Warje", "35", "20"},
                {"Bus 9", "Vishrantwadi -> Dighi -> Moshi -> Chakan", "60", "30"},
                {"Bus 10", "Sangvi -> Pimple Gurav -> Pimple Saudagar -> Rahatani -> Wakad", "50", "25"}
        };
        for (String[] route : busRoutes) {
            String[] stops = route[ROUTE_DETAILS].split(" -> ");
            for (String stop : stops) {
                stopToRoutes.computeIfAbsent(stop, k -> new ArrayList<>()).add(route);
            }
        }
        Log.i("busRoutes", String.valueOf(stopToRoutes));
        Log.i("initalise","initalised");
    }
```

Fig.1 StopToRoutes

2. Identification of Possible Routes:

- For direct routes check if source and destination stops are present in the same bus route

-For indirect routes, it will find a common stop between source and destination and concatenate the two routes using that common stop.

```
public static List<RouteOption> findRoutes(String source, String destination) {
    List<RouteOption> routes = new ArrayList<>();
    LocalTime now = LocalTime.now();
    List<String[]> sourceRoutes = stopToRoutes.getOrDefault(source, new ArrayList<>());
    List<String[]> destinationRoutes = stopToRoutes.getOrDefault(destination, new ArrayList<>());

    for (String[] srcRoute : sourceRoutes) {
        if (Arrays.asList(srcRoute[ROUTE_DETAILS].split(" -> ")).contains(destination)) {
            routes.add(new RouteOption(srcRoute, srcRoute[ROUTE_DETAILS].split(" -> "),
            source, destination, now));
        }
        for (String[] destRoute : destinationRoutes) {
            String commonStop = findCommonStop(srcRoute[ROUTE_DETAILS].split(" -> "),
            destRoute[ROUTE_DETAILS].split(" -> "));
            if (commonStop != null && !commonStop.equals(source) && !commonStop.equals(destination)) {
                routes.add(new RouteOption(srcRoute, srcRoute[ROUTE_DETAILS].split(" -> "),
                source, commonStop, destRoute,
                destRoute[ROUTE_DETAILS].split(" -> "), destination, now));
            }
        }
    }
    return routes;
}
```

Fig.2 Identification of Possible Routes

3. Representation of Routes:

- Define the RouteOption class to encapsulate route details, including individual legs (bus routes), arrival times, total travel time, and cost.

```
private LocalTime calculateNextBusTime(LocalTime currentTime, String[] bus) {
    LocalTime firstBusTime = LocalTime.of(6, 0);  // assuming first bus starts at 6:00 AM
    while (firstBusTime.isBefore(currentTime)) {
        firstBusTime = firstBusTime.plusMinutes(60); // assuming buses run every hour
    }
    return firstBusTime;
}

private int calculateTravelTime(String[] bus, String[] stops, String start, String end) {
    int startIndex = Arrays.asList(stops).indexOf(start);
    int endIndex = Arrays.asList(stops).indexOf(end);
    return Integer.parseInt(bus[PuneBusRoutes.TIME]) * (endIndex - startIndex) / (stops.length - 1);
}
```

Fig.3 Representation of Routes

4. Selection of the Best Route:

- Implement the selectBestRoute() method to iterate through RouteOption objects and select the best route based on predefined criteria such as shortest travel time and lowest cost.
- Populate the stopToRoutes map to associate each bus stop with a list of bus routes passing through it.
- Identify all possible routes between given source and destination stops.
- Consider both direct and indirect routes, checking for stops within the same bus route for direct routes and concatenating routes via a common stop for indirect routes.
- Define the RouteOption class to encapsulate route details, including individual legs (bus routes), arrival times, total travel time, and cost.
- Implement the selectBestRoute() method to iterate through RouteOption objects and select the best route based on predefined criteria such as shortest travel time and lowest cost.
- Implement the addLeg() method in the RouteOption class to add a leg (bus route) to the RouteOption object, calculating arrival times and partial route details.
- Develop the calculateNextBusTime() method to calculate the next available bus time based on the current time and an assumed hourly bus schedule.
- Create the calculateTravelTime() method to calculate the travel time between two stops on a given bus route using the LocalTime class from Java 8 for time calculations.

This sequence outlines the systematic steps involved in implementing route optimization features and defining essential methods within the RouteOption class for efficient route representation and selection.

```
public static RouteOption selectBestRoute(List<RouteOption> routes) {
        RouteOption bestOption = null;
        int bestTime = Integer.MAX_VALUE;
        int bestCost = Integer.MAX_VALUE;

        Log.i("Best Route","FInding best route....");
        for (RouteOption option : routes) {
            if (option.getTotalTime() < bestTime || (option.getTotalTime() == bestTime &&
            option.getTotalCost() < bestCost)) {
                bestTime = option.getTotalTime();
                bestCost = option.getTotalCost();
                bestOption = option;
            }
        }
        Log.i("Best route", String.valueOf(bestOption));
        return bestOption;
    }
```

Fig.4 Selection of Best Route

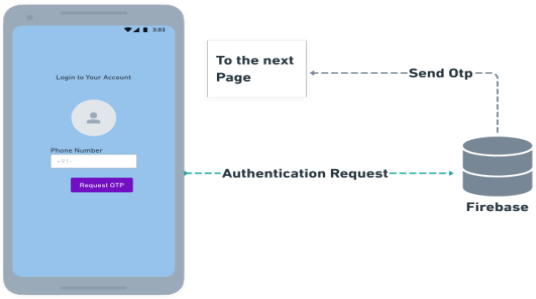

Fig.5 Login Page

I. Real-time Bus Tracking

The real-time bus tracking feature is implemented using a combination of a mobile application for bus drivers, Firebase Firestore for data storage, and a user-facing mobile application for displaying bus locations.

Driver Application

The driver application is responsible for continuously updating the bus location data in the Firestore database. It utilizes the following components:

1. Geolocator Package: The geolocator package is used to obtain the current location of the bus driver's device using GPS or network-based location services.
2. Location Updater: The location Updater() function is implemented as a custom action in the driver application. It performs the following tasks:

- Creates a reference to a specific document in the Firestore database, where the bus location data will be stored.
- Sets up a periodic timer that triggers every 5 seconds.
- Within the timer callback, it retrieves the current location using the
- Geolocator.get Current Position() method
- Converts the obtained location coordinates into GeoPoint object,
- which represents a geographic point in the Firestore database.
- Updates the specific document in Firestore with the new GeoPoint
- location data using the docRef.set() method.

User application:

1. Google Maps Integration: The user application seamlessly integrates the Google Maps SDK to render maps and display markers, ensuring a user-friendly interface.
2. Firestore Data Retrieval: The application efficiently retrieves bus location data by listening for changes to a specific document in Firestore. Upon detection of any change, it promptly fetches the updated GeoPoint data.
3. Map Rendering: Utilizing the retrieved GeoPoint data, the application generates a LatLng object, representing a precise geographical point on the map. Subsequently, it dynamically updates the map by clearing previous markers, placing a new marker at the updated location, and adjusting the camera position to focus on the latest bus location.

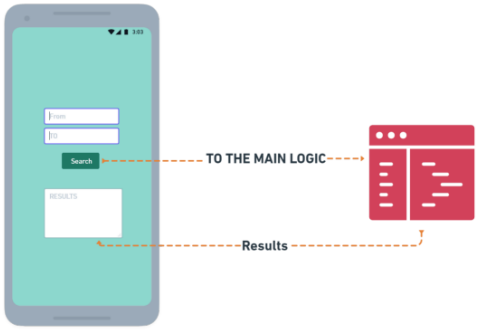

Fig.6 User Application

User Authentication:

Utilizing Firebase Authentication, the app offers a secure login process:
1. User Input: Users enter their phone number on the login screen.
2. Phone Number Validation: The app verifies the phone number for completeness and format accuracy.
3. Firebase Authentication:Valid phone numbers trigger Firebase Authentication, which sends an OTP via SMS.
4. OTP Verification:Users enter the OTP received on their phone, which Firebase verifies.
5. User Authentication: Upon successful OTP verification, users gain access to the app's features, ensuring system security and privacy.
This process ensures that only authorized users access the app, enhancing the overall system security.

Real-Time Bus Tracking: The app combines driver location updates with map rendering and Firestore integration for seamless bus tracking. Passengers view real-time bus locations on their mobile devices, ensuring a smooth experience for both drivers and passengers.

In summary, the application actively listens for Firestore document changes through the addSnapshotListener method, enabling real-time updates of bus locations on the map interface.

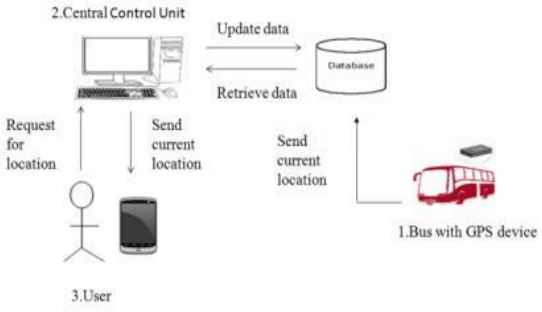

Fig.7 Flowchart

The frontend of the app is developed using modern web and mobile technologies, including Android for the bus driver's web application and Flutter for the user's mobile application.

IV.IMPLEMENTATION

Map activity: This activity sets up the Google Maps integration and handles the rendering of the bus location on the map.

FirebaseFirestore: The Firestore instance is used to retrieve the bus location data from the Firestore database.

OnMapReadyCallback: This interface is implemented by the map activity to handle the map's ready state and perform necessary operations.

The bus driver web application is developed using Flutter, a cross-platform framework for building mobile applications. The backend infrastructure is developed using Firebase, a comprehensive app development platform provided by Google. Firestore is a cloud-hosted, NoSQL document database provided by Firebase. It is used to store and retrieve the bus location data in real-time. The application creates a specific document in Firestore to store the bus location as a GeoPoint object.

V. RESULTS

- Flutter integration with Firebase and Firestore
- Custom actions for location updating and Firestore data handling
- Integration with the Google Maps SDK for map rendering and marker placement.

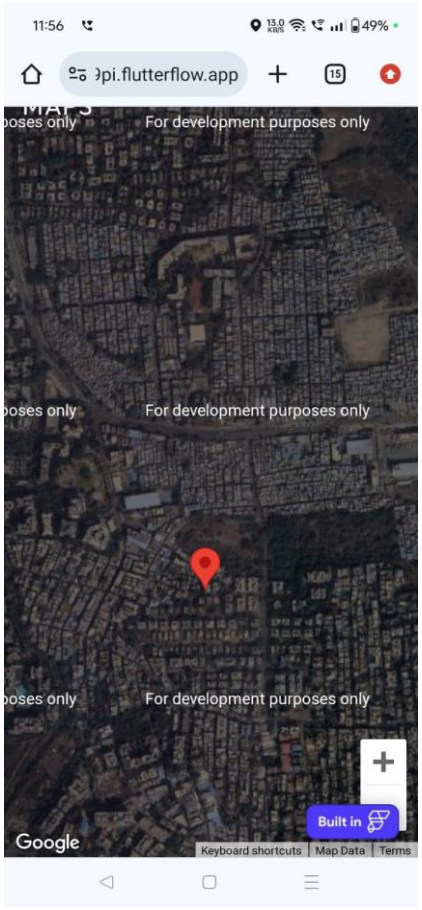

Fig.8 Displaying the Google Maps

VI.CONCLUSION

The primary objective of this work is to significantly enhance the Bus Tracking system by integrating essential features that will improve its functionality and user experience. These features include projecting accurate bus timings to ensure passengers have reliable information on when buses will arrive and depart, as well as displaying the correct bus numbers to avoid any confusion. Additionally, we aim to incorporate a GPS tracker into the system. This will provide real-time, precise location data of the buses, allowing users to track their journeys more effectively and make informed decisions about their travel plans. This system would utilize a combination of GPS and GSM modules, paired with a high-speed processor, to offer enhanced monitoring capabilities. Such a system could be deployed not only in buses but also in cars and trucks, showcasing its versatility and wide-ranging applicability. By implementing these improvements and future enhancements, the project aims to provide a robust, reliable, and user-friendly Bus Tracking system that meets the needs of modern commuters and contributes to more efficient transportation management.

VII. FUTURE SCOPE

The primary goal of this project is to enhance the current Bus Tracking system by introducing key features such as real-time bus schedule updates, precise bus number identification, and the integration of GPS tracking for improved location accuracy. Additionally, the system can be extended to private travel agencies to track their buses efficiently.

In the future, the system can evolve into a comprehensive vehicle monitoring solution by incorporating GPS and GSM modules alongside a high-speed processor. This would make it suitable for various vehicles, including buses, cars, and trucks, ensuring broad applicability. Furthermore, this Android application will increase the productivity and efficiency of bus transportation, benefiting both public and private transportation services.

REFERENCES

[1] Authors ManiniKumbhar, MeghanaSurvase and Pratibha MAvdhutSalunk, Real Time Web Based Bus Tracking System, International Journal of Engineering Research and Technology(IJERT), 2020

[2] M. A. Hannan, A. M.Mustapha,A.Hussain and H.Basri, Intelligent Bus Monitoring and Management System, World Congress on Engineering, 2012

[3] S. Eken and A. Sayar, "A smart bus tracking system based on location-aware services and QR codes," 2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings, Alberobello, Italy, 2014, pp. 299-303, doi:10.1109/INISTA.2014.6873634.

[4] R.Maruthi and C.Jayakumari ,SMS based Bus Tracking System using Open Source Technologies, International Journal of Computer Applications, 2014

[5] Kushal Gogri , Ankeet Bhanushali , Akshay Sonawane , Milind Khairnar, 2020, Real Time Bus Tracking System, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 09, Issue 06 (June 2020)

[6] Md. Marufi Rahman, Jannatul Robaiat Mou, Kusum Tara and Md. Ismail Sarkar, Real Time Google Map and Arduino Based VehicleTracking System, Institute of Electrical and Electronics Engineers, 2nd International Conference on Electrical, Computer & Telecommunication Engineering(ICECTE), 2016

[7] Salava V Satyanarayana; M. Manasa; B. Vishwanth; V. Akanksha and D.Anil Sai,IOT Based Bus Tracking and monitoring system,Institute of Electrical and Electronics Engineers,3rd International conference on Artificial Intelligence and Signal Processing (AISP), 2023.

[8] Mona Kumari, Ajitesh Kumar and Arbaz Khan, IoT Based Intelligent Real-Time System for Bus Tracking and Monitoring, 2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC),2020

[9] Dr. Nookala Venu and Dr.A.Arun Kumar, Design of Bus Tracking and Fuel Monitoring System using IoT, High Technology Letters,Volume 29, Issue 10, 2023

[10] Google Maps Platform Documentation. (n.d.). Retrieved from https://developers.google.com/maps/documentation

[11] Firebase Documentation. (n.d.). Retrieved from https://firebase.google.com/docs