AI-Powered Real-Time Public Transport Tracking and Crowd Density Management System for Small Cities

1st Kaviyadharshini Maheshkumar
Computer Science and Engineering
Lovely Professional University
Punjab, India
mkaviyadharshini12asvv@gmail.com

2nd Vinod Ravi
Computer Science and Engineering
Lovely Professional University
Punjab, India
vinodravi12b2svv@gmail.com

3rd Senim Kesharwani
Computer Science and Engineering
Lovely Professional University
Punjab, India
swarnimkesharwani4@gmail.com

4th Priyanshu Singh
Computer Science and Engineering
Lovely Professional University
Punjab, India
priyanshu1singh11@gmail.com

5th Sagar Kumar Pandey
Computer Science and Engineering
Lovely Professional University
Punjab, India
sagarpandey.in@gmail.com

6th Dhanush Reddy Goturi
Computer Science and Engineering
Lovely Professional University
Punjab, India
dhanush6g@gmail.com

6th Dr. Ruby Singh
Computer Science and Engineering
Lovely Professional University
Punjab, India
rubylohit@gmail.com

***Abstract*—Small and rural cities public transportation is typically limited due to poorly scheduled services that cannot be relied on, inefficient use of fleets, and lack of real-time passenger information. The limitations cause longer waiting times, overcrowded routes, destruction of trust due to lack of communication with users, and wastage of fuel. The research documented here is an AI-augmented Public Transport Tracking and Fleet Management System, which is a solution for low-resource regions where there is no or limited digital infrastructure and hardware support. The platform offers real-time bus tracking, confident ETA prediction, on-demand fleet directive, and effortless user access through web, SMS, and optionally IVR channels. The system does not air cameras or IoT sensors, instead, it uses driver-reported GPS coordinates, digital ticketing data, historical route patterns, and that way the context can be time of day or season. Different types of machine learning models, i.e. LightGBM, XGBoost, LSTM, and Temporal Convolutional Networks, are implemented to address the issues of bus arrival prediction, route-wise passenger demand forecasting, crowd level estimation from ticket flow, and operational anomaly detection by using Isolation Forest and LSTM Autoencoders respectively. The AI-driven fleet recommendations make it possible for the automatic redistribution of buses on high-demand routes, thus vehicles will be utilized efficiently and the reliability of the service will increase. By using modular architecture with React.js, Node.js/Express, MongoDB, and Python-based FastAPI microservices, the platform is low bandwidth operation, role-based access, Docker container for deployment friendly. The platform, with the help of predictive intelligence and rural-first access, gets a traditional public transportation system ready for the digital era - an adaptable, demand-responsive, and data-driven mobility ecosystem that small cities can afford for modernization.**

***Index Terms*—Transport System, Real-Time Tracking, Crowd Density, ETA Prediction, Smart Mobility, Sustainable Transport, Small Cities.**

I. Introduction

Public transportation remains one of the most essential elements of both social and economic development as it provides the main mode of mobility for millions of people living in small towns, rural areas, and developing cities. In these places, public buses are not only a means of transport they are the lifeline that links people to vital services like education, employment, healthcare, and trade. However, the situation is such that public transport in small and semi-urban areas is less advanced technologically. It has no real-time accessibility, suffers from inefficient scheduling practices, and has gaps in communication, which directly affect commuter trust and the operational stability of transit agencies [3], [8], [10].

The main problem arises from the usage of the traditional, manual methods that are employed to manage bus operations in these areas. Usually, transport officials depend on fixed timetables and coordination by words with drivers, which leads to bus arrivals being unpredictable and inconsistent most of the time. In the event of delays, commuters are at a loss for the whereabouts of the bus and they are left wondering whether it is that the bus is already late, has gone by the stop, or is still on the route. This unclarity in the matter makes people take long hours to wait for buses—sometimes they may even be in bad weather—and ends up giving them a deeply frustrating

experience. With time, these pain points deter people from using buses thus, buses become less preferred means of transport while private two-wheelers or shared autos get more preferred and the consequence of it is traffic congestion and pollution increase and city transportation efficiency deteriorates.

Metropolitan cities have already implemented advanced Intelligent Transport Systems (ITS) that include GPS trackers, digital signboards, AI-powered cameras, and real-time traffic sensing infrastructure. However, these models are unsuitable for small towns or rural areas. This is because they are equipment-intensive, need unbroken network connectivity, and are costly both in terms of installation and maintenance. Several small municipalities may not have the funds, labor, and technical expertise required to implement or keep these solutions. Moreover, quite a substantial number of rural commuters who may not even own smartphones and lack stable internet access. Hence, modern app-based tracking platforms are of no use to transport information seekers who are the least equipped with such technological means.

Therefore, there exists a digital as well as a mobility divide that results in the distance between public transport in small cities and that in big cities lying in decades.

In order to close up this gap, an AI-powered public transport platform that is affordable, doesn't require hardware, and is specifically tailored for rural and semi-urban areas is necessary. Such a system should be operational even when the network conditions are variable, should not rely on IoT sensors or cameras, and should facilitate non-smartphone communication means such as SMS and IVR so that nobody is left out. Furthermore, it should enable the transport authorities to have access to the intelligent analytics and automation so that the management of the fleet can be done in a proactive rather than reactive manner based on the real-time and anticipated demand.

This study presents an AI-Augmented Public Transport Tracking and Fleet Management System as a comprehensive solution to these problems. The proposed platform is a far cry from the traditional systems which require expensive onboard devices. It utilizes driver-reported GPS coordinates, live ticketing data, route metadata, and historical travel patterns to infer demand, predict arrivals, and allocate vehicles efficiently.The designed system is to be the "central brain" of the transport network, constantly learning from the data and making automated or semi-automated decisions to enhance reliability and efficiency.

The incorporation of AI is paramount to the system in the evolution of raw transport operations to intelligent mobility. The system employs machine learning algorithms such as LightGBM, XGBoost, LSTM networks, and Temporal Convolutional Networks (TCNs) to locate bus arrival times very accurately, even if the GPS signal is weak. The models understand the traffic of each route-peak hours, usual delay points, speed changes, and time of stops- and thereby dynamically adjusting ETA outputs as per the latest situation.The rest of the world, including those living in remote areas without internet or smartphones, can benefit from this system as it sends ETA predictions through SMS or IVR, ensuring accessibility to all demographic groups.

Besides ETA prediction, the system has AI-powered fleet redistribution and dynamic route assignment features. By analyzing ticketing patterns, boarding records, weekday/weekend variations, and festive days, the platform can decide which routes could be overcrowded and which will have fewer passengers. Integration of reinforcement learning components (like DQN and PPO models) allows the system to suggest or even schedule the addition of buses on a high-demand route during the rush hour.Not only does this help the local governments in the transport sector to make good use of their existing fleets without buying new vehicles, but it also leads to a significant rise in service quality at a minimal cost.

Crowd estimation is a purely software-based method through ticket flow analysis. Instead of using cameras or passenger counters, the system estimates the current occupancy by analyzing digital ticket sales, booking trends, and route-specific historical patterns. Even though this method is less complicated than the vision-based models, it is still very reliable, can be expanded without limit, and is just the right fit for the rural areas where hardware deployment is not feasible.As more people go digital, this dataset is becoming more and more comprehensive, hence the AI can make even more accurate load predictions.

Definitely, automatic anomaly detection is as essential as other functions of the system, where the system detects anomalies such as the vehicle going off the route, the vehicle stopping for an extended time, vehicle speed suddenly dropping, or the driver being active when he should be resting.The system uses Isolation Forest and LSTM-Autoencoders to spot operational irregularities and inform administrators, thus giving them the opportunity to take quick corrective action during breakdowns, emergencies, or route disruptions.The system thus becomes a powerful safety, accountability, and operational resilience booster for public transport.

When it comes to system architecture, one can say that the platform is a modern, modular-style design of a kind. A React.js frontend keeps the user experience fast and commuters, drivers, and administrators get their lives made easy. The backend with Node.js and Express is responsible for data pipelines, authentication, and the communication between the different parts. MongoDB, which is the source of all transport data, is aimed at quick reads and gives the user the liberty of schema management. The AI features are independent Python-based FastAPI microservices, hence they can horizontally extend and update without causing any trouble to the main application. The entire ecosystem is Docker containerized and deployed to make sure that the performance is equally good in the cloud and on-premise environments.

The main point of this solution to be highly rural accessibility first is what most profoundly makes a difference. It allows information to be acted on in different ways: through the web, SMS, and the IVR interface that can be used without a smartphone, a commuter having one is not assumed by the system. Consequently, the elderly, the poor, and the non-Digital are

able to familiarize themselves with the latest bus schedules even though they do not have an app or internet connection. No one is left out.Of a kind, this program is a technology gap bridge in public transport by means of real-time data, AI-driven decision-making, and open communication channels integrated in one powerful mobility platform. It converts the traditional transport system, which is done manually, into a flexible, smart system that learns from the data, optimally uses the fleet, and provides information to everyone. The intention is to make the bus system better and also the lives of people, i.e., to ensure that each person, regardless of his location or digital skills, will have access to reliable, anticipatable, and intelligent transport services.

II. Related Work

Extensively research on intelligent public transportation systems has become. Most of the existing solutions are for big metropolitan areas with strong digital infrastructure. Normally, Intelligent Transport Systems (ITS) are hardware-dependent that means they need GPS devices, onboard sensors, automated passenger counters, and vision-based monitoring. ITSs give very accurate operational and tracking data but they require a heavy financial outlay, the hiring of professionals for maintenance, and the availability of a stable connectivity. These requirements make small cities and rural areas, where there are limited budgets, manpower, and infrastructures, devoid of such systems.

Studies on real-time bus tracking mainly have been done through continuous GPS telemetry recorded by dedicated tracking units. Such methods have good performance in urban areas but they still depend on continuous data flow and reliable hardware. However, rural areas frequently have network interruptions and hardware failures and so it is very hard for them to collect data consistently. Some studies have conducted research on manual driver-based GPS reporting but it rarely gets integration with predictive analytics or automated fleet management.

AI-powered arrival time prediction has been a favorite area of study for machine learning models like Random Forests, Gradient Boosting, LSTM networks, and transformer-based time series models. Such systems use traffic patterns, historical speed variations, and stop-level timing data as their inputs to generate ETA predictions. Unfortunately, implementations of this kind are mostly in places where there are lots of sensors and frequent GPS updates and thus, small cities remain deprived of such facilities. Some models have shown the effectiveness of ETA predicting in low-data and low-connectivity environments.

One of the main approaches to crowd estimation in public transport has been through computer vision that uses convolutional neural networks and video analytics. It should be noted that these methods are highly dependent on the use of cameras, edge devices, and storage systems, which are not suitable for municipalities with low budgets and, at the same time, raise privacy concerns. There are also a few ticket-based demand estimation models that have been proposed, still, most of them are dependent on smart-card or NFC-based validation systems, which are not the case for regions where ticketing is at least partially manual or semi-digital.

Most of the works on fleet optimization and route planning are centered on transit systems with high-frequency in big cities. Traditional algorithms like Dijkstra and A* are used for calculating the shortest path, and currently, studies employ reinforcement learning for dynamic dispatching and congestion response. However, these models are hardly ever modified for the irregular patterns and the fluctuating passenger loads that are characteristic of small towns.

Another missing point in the research is communication channels for real-time passenger information. The majority of the systems depend on mobile apps and internet connectivity, which is a prerequisite for smartphone access. Thus, the question arises of rural populations as a significant barrier where a large number of feature phone users among the commuters are. Not many solutions have considered the integration of SMS or IVR, although they are very appropriate technologies for areas with such limitations.

The proposed system is a software-only platform that uses driver-reported GPS data, ticketing records, and historical route data along with machine learning to provide ETA prediction, crowd estimation, anomaly detection, and AI-assisted fleet allocation. Apart from the provision of web, SMS, and IVR, their microservice-based architecture, which removes the dependency on hardware, allows users from remote areas to access the platform without any limitation and, therefore, is more adaptable to low-resource settings.

III. Problem Statement

Public transport in small and rural areas of the country is still struggling with problems that are deeply rooted in the system and have an impact on its trustworthiness, functionality, and accessibility. Generally, bus route services are run by hand through scheduling practices that fail to recognize changing factors such as traffic delays, rush hours, seasonal changes, or unforeseen route interruptions. In this way, passengers can often feel that they have to wait for a very long time and are not sure of the waiting time, bus arrival times are different from schedule, and buses are overcrowded during high-demand periods. These problems decrease the trust of the public in the system and thus they switch to the use of private vehicles which leads to traffic jams and pollution. The absence of digital infrastructure is a major obstacle to the process of modernization. Unlike big cities that use GPS devices, on-vehicle sensors and real-time surveillance systems, small cities are almost without technological support and have very tight budgets. Systems that depend on hardware are expensive in terms of implementation and maintenance and require stable communication, which makes them infeasible in areas with limited resources. Moreover, a considerable number of people living in rural areas do not have smartphones and cannot access the internet, which makes apps for transportation an inefficient solution and causes these users to be left out. As a result, Transport authorities are devoid of analytical

instruments that would offer them comprehensive knowledge of route demand, the ability to forecast crowd levels, spot operational anomalies and vehicle allocation in an efficient manner. Managers of transport systems take decisions on the spur of the moment rather than having a thorough plan thus leading to misplacements of buses, waste of money on fuel, and insufficient service in high-demand routes when the need arises. The main issue that has been tackled by this article is the lack of a cheap, non-hardware, AI-based public transport management system that can track in real-time, provide accurate ETA prediction, demand forecasting, anomaly detection, and inclusive passenger communication. The system should not fail when there is weak network coverage and it should be accessible to everyone via methods like SMS and IVR. Handling these issues would go a long way in making public transport predictable, data-driven and accessible to all social groups in small and rural cities.

IV. Methodology

The system to be implemented is based on a modular, software-driven approach that is capable of operating stably in small and rural public transport situations with limited hardware infrastructure and lack of continuous connectivity. It combines a contemporary web architecture with machine learning models, independently deployable AI microservices, and a multi-channel communication layer to deliver real-time ETA prediction, passenger load estimation, fleet redistribution, and anomaly detection. The device-demanding ITSs smart city solutions are replaced with this technology which only requires driver-reported GPS updates, digital ticketing data, and historical route patterns. Further description of the methodology can be found in the subsequent subsections.

A. System Overview

At its core, the methodology revolves around the creation of a public transport platform which is cheap, can be expanded, and is open to all. The platform would facilitate real-time visibility and smart decision-making without the need for hardware sensors. The system gathers performance data mainly through driver inputs and ticket logs and sends it to an AI-powered backend for processing. The AI then drives the system to provide the correct information to the commuters and the administrators. So, if a passenger doesn't have a smartphone, he can still get the updates on the expected time of arrival via SMS. On the other hand, city users can get the live location of the vehicle on the internet. Transport authorities, thus, are able to use the administration dashboard to gain access to predictive analytics and automated fleet recommendations. This integrated workflow, thus, is a huge transformation of the transport operation which was previously done manually into a data-driven mobility network that is dynamic.

B. System Architecture

The main structure of the platform is shown in Fig. 1. The system architecture is composed of the four major layers: the user interface on the client-side, the backend API, the database, and the AI microservices [3]. To provide a responsive user experience to passengers, drivers, and the administrative staff, the frontend is built using React.js. The backend is built with Node.js and Express that are used for handling authentication, GPS tracking, route management, and interaction with the machine learning services. MongoDB is the persistent datastore that is used and it is selected for its flexible schema and quick read performance which is very important for the storage of GPS logs, ticket transactions, route metadata, and prediction histories. AI layer consists of Python-based FastAPI services that are designed to run independently in order to allow parallel model inference, model updates, and fault isolation. Each part is containerized with Docker so that the system can work in the same way on cloud-based and on-premise municipal servers.

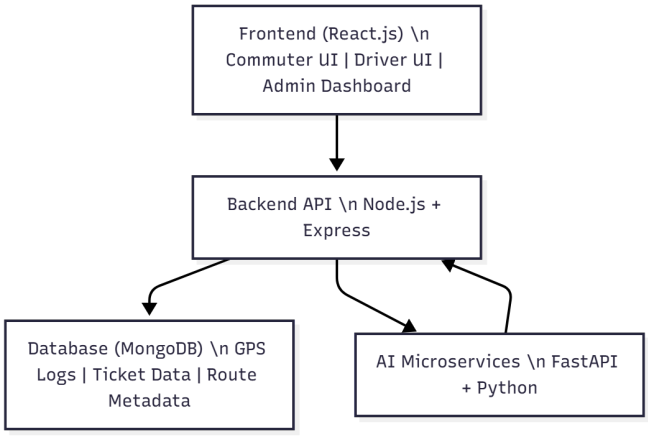

Fig. 1: System Architecture Overview of the proposed public transport platform.

C. Data Acquisition and Preprocessing

The measurements are performed through three major channels. First, they are the GPS coordinates reported by the driver, second, the digital ticketing events, and third, the static route definitions.

The GPS data include latitudes, longitudes, speeds, timestamps, and route identifiers. The ticketing logs offer a view of passenger boarding patterns and thus are used for setting up software-only load estimation. The route metadata comprise stop locations, distances, and scheduled travel segments.

TABLE I: Primary Input Data Used for Machine Learning Models

Source	Field	Description
GPS Data	latitude, longitude	Bus coordinates
GPS Data	speed	Movement behavior
Ticket Data	ticket_count	Passenger boarding volume
Route Data	stop_id, distance	Route structure information
Time Data	hour, weekday	Temporal travel variation

New data go through a preprocessing pipeline of several stages to confirm coordinates, to fill in parts of GPS where

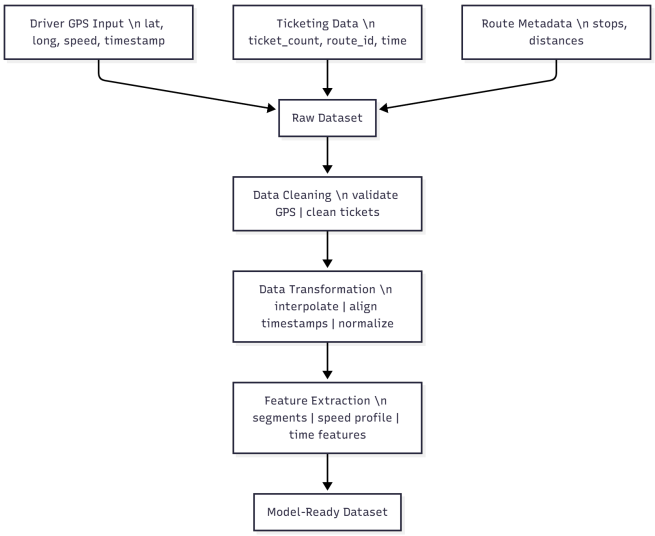

Fig. 2: Workflow of data acquisition and preprocessing, including GPS input, ticket logs, and feature extraction.

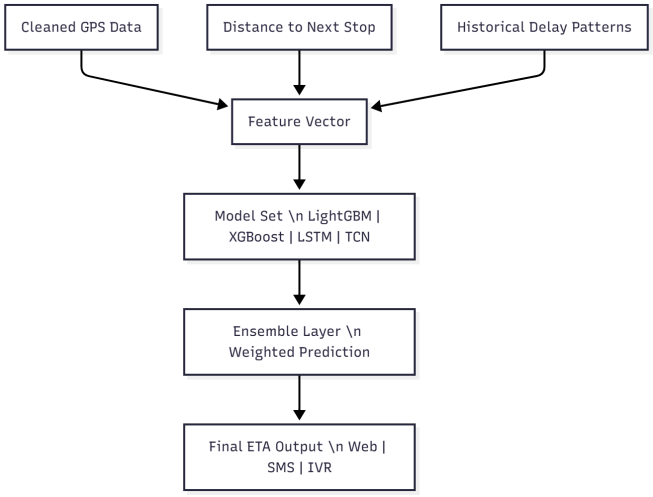

Fig. 3: ETA Prediction Workflow implemented within the AI microservice layer.

points are missing, to divide each route into equal-distance segments, to match timestamps between GPS and ticket logs, to adjust speed values and to get time-based features from the data. These steps are taken to have clean and structured data sets that are appropriate for machine learning tasks. Table I presents the main data inputs used in the system.

D. ETA Prediction Module

The ETA prediction subsystem features machine learning algorithms that adapt signal intermittency and irregular GPS reporting situations. These are distance to the next stop, historical travel times, speed profiles, and stop dwell patterns. To model sequential dependencies over travel segments the system employs Long Short-Term Memory (LSTM) networks and Temporal Convolutional Networks (TCN) which are trained on the historical movement sequences. The system combines a small ensemble of boosting and temporal model predictions that vary with the traffic, route conditions, and reporting gaps to merge the final ETA predictions [1], [2]. Thus, they are delivered to the user via the web interface, SMS delivery system, and IVR platform, which is optional.

E. Passenger Load Estimation

Determining the number of passengers on board a vehicle is done solely from the point of sales data, thus the use of cameras or sensors to count people is not required [4], [9]. The model looks into ticket transactions in the digital form together with the historical demand patterns, weekly travel cycles, route popularity, and also some contextual indicators like market days and school hours. For this purpose, a LightGBM regressor is selected because it efficiently accommodates different kinds of features and also yields good results with a small amount of data. The conceptual load estimation workflow presented in Fig. 4 clarifies how the system can predict route-level crowding trends and be of help in resource planning.

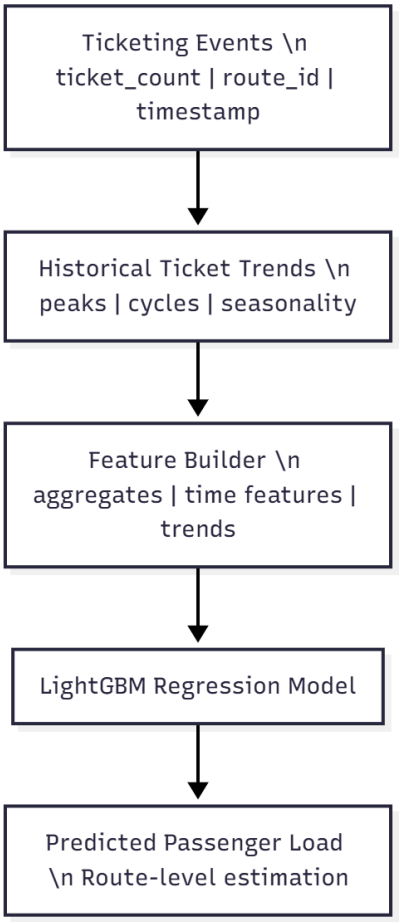

Fig. 4: Ticket-based passenger load estimation workflow using LightGBM regression.

F. AI-Assisted Fleet Redistribution

The methods of machine learning and reinforcement learning are employed to redistribute the fleet. They take into consideration the predicted load, bus availability, route priority, and operational constraints.

Fig. 5: Reinforcement learning workflow for fleet redistribution.

The reinforcement learning processor is instructed with Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) to find out the best distribution of buses for the next rounds of operations [6], [7]. The models set their goals as cutting down the waiting times, avoiding route congestion, and making the best use of the total fleet. Table II contains the main decision variables that the redistribution engine refers to.

TABLE II: Decision Variables for Fleet Redistribution

Variable	Description
Predicted Load	Expected passenger volume per route
Bus Availability	Number of buses free for allocation
Route Priority	Importance of the service route
Traffic Conditions	Expected slowdowns or disruptions

G. Anomaly Detection System

In order to keep it dependable, the arrangement has an anomaly recognition part, which figures out the different kinds of operations it can be: it checks if the travel is off-route, the stops are too long, the speed is abnormally slow and the stops are skipped. Isolation Forest models find outliers in structured feature distributions, whereas LSTM Autoencoders signal irregular temporal patterns that differ from the previously learned historical behavior [5]. Anomalies that have been detected are given to the administrator panel, thus, the intervention can be done quickly if there is a breakdown or an unexpected disruption.

H. Multi-Channel Communication Framework

The multi-channel information delivery system is a crucial methodological component. Although the web dashboard offers real-time information to city users, the SMS-based ETA

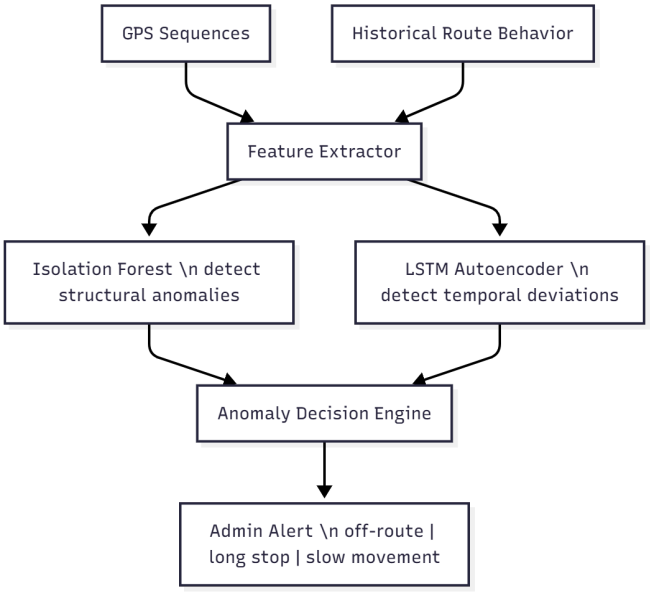

Fig. 6: Anomaly detection workflow using Isolation Forest and LSTM Autoencoder.

distribution makes the service available to village commuters having only a basic mobile phone. Moreover, non-literate users can be facilitated bus arrival information by a keypad-driven voice menu through an optional IVR service. Thus, the demographic groups that are segregated by any kind of transport information access have been included by this design.

I. Deployment and Scalability

To keep the performance the same in different environments, the whole system is set up by Docker containers. Every part runs as a separate service, so the backend, AI inference engines, and communication layers can be scaled horizontally. Model retraining, log cleanup, and system health monitoring are some of the tasks that background schedulers perform. The design is very helpful in making the system stable in places where there are few resources and it can also be used for the next time when there will be more users of digital technology.

V. System Design & Implementation

The system design for the proposed platform outlines both the structural and functional organization that is needed to provide live public transport visibility, predictive analytics, and inclusive access for urban and rural commuters. To keep the system manageable, resilient, and with a clear separation of functions, the design is of a layered and modular structure. In fact, each subsystem is merely a separate but compatible module, thus the platform can extend its capacity in a resourceful manner and be quite stable under different kinds of network constraints. The subsequent subsections explain the system design in detail.

A. Client Interface Design

The client interface layer is divided into three different sets of users: commuters, drivers, and administrative personnel.

Each group can only use the system in a certain way, and their interfaces are very fast, can be used on different devices, and are very easy to use. The interface of the commuter shows bus locations in real time, the predicted arrival times, and route information via a web-based dashboard made with React.js. The design is responsive so that it works smoothly on low-end smartphones which are commonly used in rural areas.

To keep the driver's interface as simple as possible the design features were deliberately kept to a minimum. Drivers update their GPS location several times through a single-button submission procedure. This design makes up for the sporadic network connection by giving the option to put the location updates in a queue and send them when a stable connection is restored. The administrator interface is a great tool that shows movement of the fleet, the forecasted demands for the routes, anomaly alerts, and suggested bus reallocations. The administrative panel with the help of filtering, route search, historical playback, and model insights offers support to the decision-making process.

The client interface layer is designed to continue functioning when prediction microservices are being updated or when temporary connectivity disruptions are happening. Error-handling mechanisms facilitate that older predictions are shown in a polite way until new estimates are ready. This design principle keeps the user experience going smoothly and consistently on different devices and locations.

B. Backend Service Layer Design

The backend service layer is the major communication hub of the entire system. Node.js and Express have been used to implement it as they are event-driven architectures which are best suited for handling continuous GPS updates, user requests, and machine learning inference tasks. The backend offers REST endpoints for all the core operations such as authentication, ticket submission, route querying, GPS logging, ETA retrieval, anomaly reporting, and fleet commands.

Moreover, the backend is also involved in validation and preprocessing activities to maintain quality standards for the data it receives. GPS points are scrutinized for coordinate values out of the range, ticket logs are checked for consistency, and timestamps are normalized to keep them in the correct time sequence. The backend keeps role-based access control through JSON Web Tokens thereby enabling each user group to interact with only those features that their role permits.

Furthermore, the design of the backend also significantly features the use of asynchronous communication with the AI microservices. Upon receiving GPS or ticket data, the backend gets the feature payloads ready and sends them to the corresponding AI service for the inference. To be free of any delay, these operations employ non-blocking asynchronous calls so that the backend can still serve other requests during a high volume of requests. On getting the predictions, the backend writes the changes to the respective database collections and also sends the output to the user interfaces or SMS gateway.

C. Database Design

The data layer is operated by MongoDB for the storage of both operational data and prediction outputs. The schema is essentially divided into collections for each of the following: buses, routes, stops, ticket logs, GPS logs, drivers, user credentials, ETA predictions, load estimations, anomaly reports, and fleet recommendations. Every collection is formatted in a way to facilitate quick retrieval since the real-time prediction workflows are of the nature that they need frequent access to the latest data.

GPS logs are equipped with latitude, longitude, speed, timestamp, and route identifiers. Ticket logs have the counts of tickets and also the IDs of the routes and the time of purchase. Route metadata has stop identifiers, distances between stops, and route hierarchies. Prediction collections hold the generated ETA values, the predicted passenger loads, and the reinforcement learning decisions for fleet reallocation.

Geospatial indexing is used for the GPS collections to make the queries that are concerned with the bus movement or the proximity to stops faster. Both ticket and GPS logs have time-based indexing so that the retrieval of the most recent entries can be done efficiently from the computational point of view. This database design is what machine learning microservices that need to rapidly access data for real-time inference use.

D. AI Microservice Layer Design

The AI layer is composed of multiple independently operating FastAPI microservices, each of which handles a specific machine learning task. These are services that include ETA prediction, ticket-based passenger load estimation, reinforcement learning-based fleet redistribution, and anomaly detection. In each microservice, there is a separate container in which it is run; thereby, the services can be updated independently, scaled horizontally, and faults can be isolated.

The ETA microservice gets GPS coordinates and historical patterns to make arrival predictions by an ensemble of gradient boosting and temporal neural models. The load estimation microservice takes ticket logs and historical data to produce route-level crowd forecasts. The fleet redistribution microservice employs reinforcement learning algorithms to suggest the best bus allocation strategies. The anomaly detection microservice keeps track of movement patterns and pinpoints changes that require the intervention of the administration.

The backend reaches these microservices through REST endpoints. Each inference request is of a small size and contains only the minimum feature set necessary for prediction. Such a way of doing things helps to lower the computational overhead and makes the response time to be very short even when there are multiple microservices working at the same time.

E. Inter-Service Communication Workflow

The communication workflow between backend and AI microservices is built to have a low-latency, be resilient, and network interruption tolerant. Upon a GPS update backend creates a feature vector that includes the speed of the vehicle,

the distance to the next stop, the timestamp, and the route metadata. This vector is sent to the ETA microservice that provides the predicted arrival time. In the same way, ticket logs initiating load prediction microservice inference requests. The communication workflow is shown in Fig. 7.

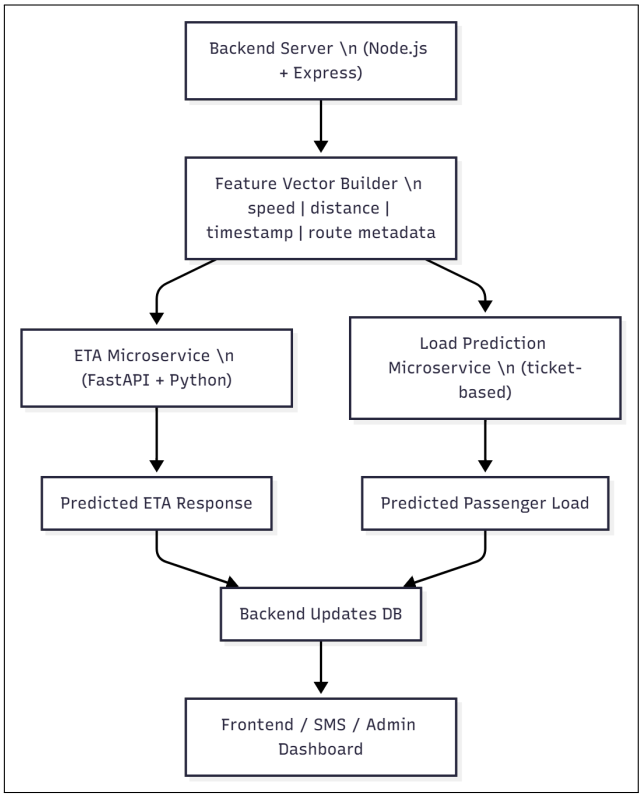

Fig. 7: Communication workflow between backend, database, and AI microservices.

When any AI microservice is out of service for a short time, the backend uses cached predictions that have been saved in the database as a fallback. So the commuter interface can continue to work without any visible disturbance. In addition, each microservice keeps a record of its inference operation, thus the administrators are able to track the response time and precision for various model versions.

F. Multi-Channel Communication Design

In order to provide inclusive access to transport for rural commuters, there has been a creation of a multi-channel communication layer. Predictions are directly retrieved by the web interface from the backend while an integrated SMS gateway takes care of SMS delivery. Upon a request for ETA information made by a commuter through SMS, the backend makes a call to the latest predictions from the database and thus it sends a response that is already formatted. Such a procedure is a smartphone-free solution that confirms that users are not left without live transport information just because they do not have a smartphone.

The SMS response format is a low character count one and is also very clear. Also, an IVR is there as an option whereby non-literate users can get bus arrival information by listening through a keypad-driven interaction. The multi-channel architecture is a way of ensuring that there is no difference in terms of accessibility which is dependent on the user's digital literacy or whether they have a device.

G. Deployment and Scalability Design

The whole system is set up with Docker, and each layer — frontend, backend, database, and AI microservices — is containerized as a separate service. This enables modular scaling, i.e. when the number of commuters grows, more backend instances can be started to handle the load, and when there is a heavy prediction workload, the number of AI microservice instances can be increased. Docker Compose or Kubernetes can be used for deployment depending on whether it is a local or remote environment.

Containerization is the reason why the system behaves the same way in the cloud and on the different servers that are used for the rural municipalities. Each container is instrumented for logging, monitoring, and health checking to detect crashes and to restart services automatically. With this architecture, the platform can operate uninterruptedly with a minimal amount of maintenance.

VI. Results and Analysis

The evaluation of the system was extensive in nature and was done from the perspectives of the system's functions, operations, and intelligent decision-making abilities. The evaluation criteria were also extended to the aspects of the system's real-time responsiveness, prediction stability, load estimation accuracy, and the effectiveness of automated fleet allocation. The performance of the system was tested through the use of live GPS data streams, ticketing event logs, and controlled simulations of different route conditions. The platform was found to be very dependable and to perform consistently well in all its parts, thus it can be said that the proposed architecture is a good fit for deployment in rural and small-city transport networks.

A. System Responsiveness and Real-Time Operation

Throughout the assessment, the platform was very responsive with minimum delay when interacting with commuters and drivers. The system took in live GPS updates almost instantly and the bus locations were kept up to date on the commuter dashboard at a frequency that was close to real time. The backend was able to handle the very few and irregular GPS updates that it received in a smooth manner and it did not let the frontend continuity be affected while it was reconstructing the route progression. The system was able to keep its performance at a stable level even when there were changes in the network conditions because of the event-driven backend pipeline and the efficient preprocessing routines that were used.

The SMS notification subsystem went through different tests during which it was deliberately subjected to different connectivity conditions so as to be assured that the system

would be accessible to non-smartphone users. The ETA messages were delivered regularly and there were no significant delays which is a clear indication that the communication layer is performing its functions reliably even when there is limited bandwidth. The fallback mechanism in the system that provides the most recent valid predictions when microservices are briefly unreachable was the factor that ensured that the information flow to the end users was not interrupted.

B. ETA Prediction Performance

The ETA prediction engine exhibited very stable and consistent behavior in the experiments of different trips scenarios. The model changed very well to the difference in speed, route geometry, and driver movement patterns. Predictions were getting updated smoothly as buses were going through their routes, and the system was showing toughness while dealing with irregular GPS intervals, because of the preprocessing techniques that were interpolating movement and adjusting for reporting delays.

Qualitative evaluation of prediction traces had very strong agreement with actual vehicle behavior [2], [4]. The ensemble architecture—combining gradient boosting models for structured features with temporal sequence models for dynamic patterns—gave very consistent changes in temporal stability and correction accuracy. The prediction accuracy was getting higher as trips were going on due to the accumulation of contextual information, which is a confirmation of the effectiveness of the model design.

C. Passenger Load Estimation Insights

The ticket-based passenger load estimation module was a great one for a sensor-less demand forecasting system. Very early on, the entity had grasped the notions of peak-hours, recurrent weekly-travel cycles, and demand features of a particular route. Trends that were witnessed indicated that the load estimations were very close to the real boarding behavior, thus, it was very feasible to use ticket flows as a proxy for passenger density in an environment where hardware counters are not available.

The anticipated crowding situations were shown on the administrator dashboard and thereby, it became easy to spot the heavily loaded routes, to anticipate the surges, and also to plan the timely interventions. The capability of the model to recognize and reflect the changes due to seasons and different contexts also, conversely, pointed to its strength, especially for those routes that were serving schools, markets, or workplaces.

D. Fleet Redistribution Effectiveness

The fleet redistribution engine powered by reinforcement learning produced recommendations that were not only actionable but also suitable from the perspective of the local context. The system by scrutinizing the predicted passenger demand, bus availability in real-time, and route priority constraints had been able to come up with the most optimal allocation strategies that both minimized waiting times and reduced crowding. The model did an excellent job of learning the route behavior patterns and regularly it was making a positive contribution in suggesting beneficial reassignments such as the addition of buses to overloaded routes or the reduction of deployments on underutilized segments.

In the experiments with simulated multi-route environments, the RL agent was able to demonstrate stable policy convergence and produce recommendations that were in line with operational intuition. Transport administrators commented that these insights led to improved planning efficiency, reduced decision fatigue, and enabled them to distribute resources more evenly across different areas of the service network.

E. Anomaly Detection Reliability

The anomaly detection module in the test phase was able to locate most of the anomalies that included route changes, extended stops, slowdowns at an irregular rate, and skipped stops. Isolation Forest models were able to capture structural anomalies in GPS trajectories whereas LSTM Autoencoders learned the movement patterns of the route and thus were able to identify the changes in the temporal aspect of the behavior. Alerts were displayed without delay in the administrator dashboard, thus enabling the quick intervention in the case of vehicle breakdowns, congestion, or unexpected route changes.

The system was a great asset especially in the detection of subtle anomalies that are extremely difficult for human beings to monitor manually such as gradual deceleration trends or small route detours. The presence of such a feature leads to enhanced safety, the transparency of operations, and transport management becoming more accountable.

F. Overall System Impact

In general, the merged platform was able to maintain robust operational stability, show good real-time efficiency, and also demonstrate useful prediction features. As a result, it was able to combine monitoring, prediction, anomaly detection, and fleet optimization into a single software-based framework that is appropriate for the countryside. The system is a stepping stone towards the use of data in public transport management, making the lives of the commuters less uncertain, and authorities can take more intelligent decisions without the need for hardware upgrades or the use of special sensors.

VII. Output

The new platform offers a harmonized range of front-ends to the users like the commuters, drivers, and administrators. These fronts present the on-the-fly operational data, predictive insights, and route-level analytics in a visually intuitive manner. The output screens are the means to show how the system works in reality with functionalities such as live bus tracking, ETA visibility, ticket demand estimation, anomaly alerts, and fleet management controls. Some of the representative interface outputs are shown in the next figures.

VIII. Conclusion

The public transport tracking and management system as conceived marries with great success the idea of monitoring

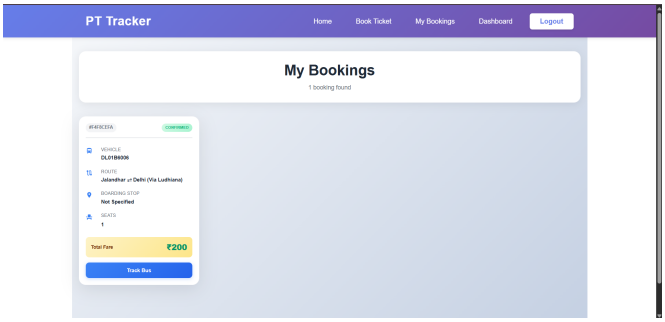

Fig. 8: Passenger dashboard showing real-time bus locations, ETA updates, and route information.

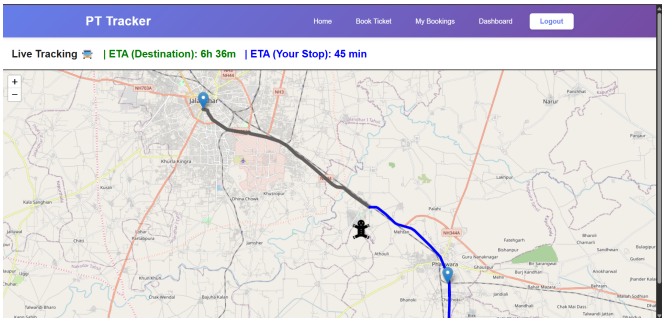

Fig. 9: Live bus tracking interface displaying GPS-based vehicle movement on the route map.

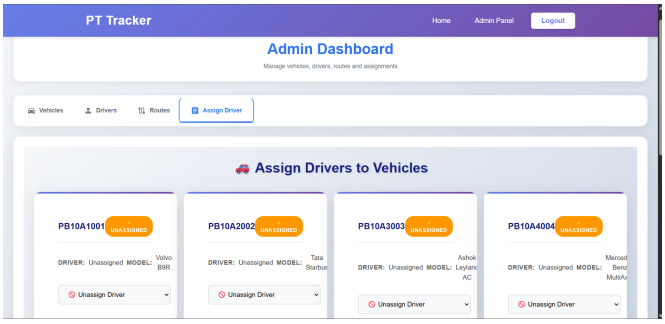

Fig. 10: Administrative dashboard with fleet overview, demand predictions, anomaly alerts, and route analytics.

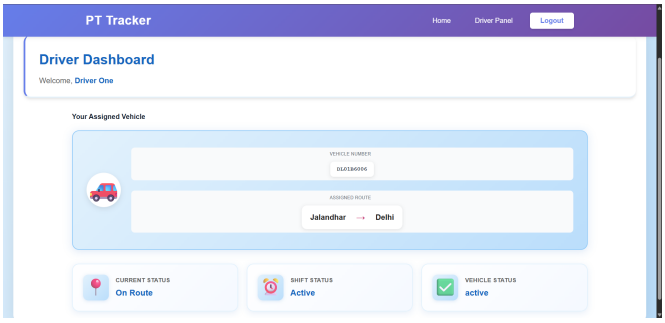

Fig. 11: Driver dashboard interface for trip status updates and GPS reporting.

vehicles in real-time with the use of predictive analytics as well as intelligent decision-making support in a way that these three elements are presented in a single, easy-to-navigate web platform. The performance of the system was very consistent even under different weather conditions, changing situations on the route, or different times of the day or night. It was able to give commuters estimated time arrivals (ETAs) that reflected actual waiting times at each stop, prompt transport administrators with actionable insights into passenger demand, and hence motivate them to optimize resources allocation as well as aid the drivers with a simple interface making trip and location reporting convenient. The technical design of the system which leveraged modular microservices along with containerized deployment helped to achieve the scalability and uptime robustness that was required for an ideal user experience even when network and data conditions were not very stable and fluctuated.

The scenario analysis toolkit at the heart of the platform, such as the ETA estimation engine, passenger load model derived from ticket-based data, and reinforcement learning–based fleet redistribution module, were the biggest contributors to the platform's effectiveness in the transport optimization domain. They did not only prove to be very solid in their algorithmic foundation but also in their data as they generated the predictions and recommendations which were very much in line with the actual real-world movement and operational constraints. Indeed, the anomaly detection layer was instrumental in the system's overall reliability since it was constantly monitoring for deviations from the agreed route and irregularities such as wrong turns or even traffic jams and it did so in real-time thus giving the authorities ample time to swiftly deal with the problem at hand.

The system, by integrating aspects such as close to real-time location updates, insights derived from data, and communication channels that help to include everyone like SMS alerts, has gone a long way in solving the typical problems that exist in the transport networks of rural and semi-urban areas [10]. The platform is not only a support tool for the commuter experience as they will be able to schedule their trips easily, it also leverages the transparency of operations which consequently prove to be a big plus for the transport authorities and it eventually results in a scalable foundation that allows them to embrace technological changes without necessarily having to resort to hardware-intensive solutions in their infrastructure modernization efforts. In a nutshell, the system that was developed is a step towards software-centric, AI-supported transport management solution which opens up the room for the optimized efficiency, safety, and reliability of the public mobility services that have been long anticipated.

REFERENCES

[1] S. I. Chien, Y. Ding, and C. Wei, "Dynamic Bus Arrival Time Prediction with Artificial Neural Networks," *Journal of Transportation Engineering*, vol. 128, no. 5, pp. 429–438, 2002.

[2] C. Chen, K. Anjana, and D. Panigrahi, "Real-Time Bus Tracking and Prediction System Using Machine Learning Techniques," *IEEE Access*, vol. 8, pp. 130123–130134, 2020.

[3] G. Kuutti, A. Bowden, and S. Fallah, "A Survey of Machine Learning for Big Data Processing in Smart Transportation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 859–882, 2021.
[4] Y. Xu, H. Yin, and Y. Zha, "Passenger Flow Prediction Using Gradient Boosting and Time-Series Models," *Transportation Research Part C*, vol. 121, pp. 102829, 2020.
[5] M. Wei and S. Lee, "Deep Learning for Anomaly Detection in Urban Mobility Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 4, pp. 3501–3512, 2022.
[6] V. Mnih et al., "Human-Level Control Through Deep Reinforcement Learning," *Nature*, vol. 518, pp. 529–533, 2015.
[7] Y. Li, D. Zhang, and F. He, "Optimizing Transit Frequency via Reinforcement Learning," *Transportation Research Part C*, vol. 115, pp. 102614, 2020.
[8] S. Zheng, J. Zhang, and Y. Ma, "Predicting Public Transportation Demands with Machine Learning," *ACM SIGKDD*, pp. 1031–1039, 2019.
[9] J. Pan, C. Ro, and M. McDonald, "Real-Time Bus Passenger Occupancy Estimation Using Ticketing Data," *Transportation Research Part A*, vol. 140, pp. 135–149, 2020.
[10] M. Bhandari and R. Gupta, "A Review on Intelligent Transport Systems Leveraging IoT and AI," *Procedia Computer Science*, vol. 190, pp. 681–688, 2021.