International Journal of Engineering Trends and Technology
ISSN: 2231-5381/ https://doi.org/10.14445/22315381/IJETT-V74I8P115

Volume 74 Issue 8, 222-238, August 2026

Original Article

Implementation of a Smart IoT-Based Trotro Passenger Pickup System in Ghana

Isaac Adjaye Aboagye[1], Margaret Richardson Ansah[2], Wiafe Owusu-Banahene[3], Prince Kotoko[4], Alexander Awuviri[5]

[1,2,3,4,5]*Department of Computer Engineering, University of Ghana, Legon, Accra, Ghana.*

[1]*Corresponding Author : iaaboagye@ug.edu.gh*

Received: 25 February 2026 Revised: 14 July 2026 Accepted: 22 July 2026 Published: 29 August 2026

Abstract *- As the number of people living in cities increases, there is a need to improve the transportation system. In Ghana, the public transport systems in the urban areas are faced with longer waiting times, overcrowding at bus stops, and a lack of a communication system between bus stops and the drivers of the vehicles. Recent developments in the Internet of Things (IoT), Artificial Intelligence (AI), location management microservices, computer vision, cloud computing, and embedded systems offer novel opportunities to overcome these challenges within the public transportation system. In this research, a low-cost passenger counting and tracking system is developed and integrated into a Trotro public transport system in Ghana. The system is designed using a Raspberry Pi with a camera module attached to it and a portable power bank. The system uses a computer vision model (YOLOv11) trained to count passengers at a bus stop by recognizing human heads and bodies in their entirety. This dual detection method gives the system robustness against overcrowding. When only heads can be seen, it counts by the number of heads. When full bodies can be seen, it counts by the number of bodies. When both can be seen, the model ensures that the head and body bounding boxes are paired one-to-one to prevent overcounting. After counting, the data is sent to a web server where it is stored and processed for use in a mobile application. The system has an in-built Bluetooth-enabled speaker that announces the driver's trip information to passengers at the bus stop. The research demonstrates how technologies can improve the transportation system in Ghana.*

Keywords *- Trotro, Computer vision, Cloud computing, Mobile application, Microservices, Public transport.*

1. Introduction

Trotros are the most common and accessible mode of public transportation used by most commuters in Ghana. They are low-cost and easy to access. Issues such as unpredictable arrival and departure times lower the credibility of the services. Commuters spend a lot of time waiting at the bus stops without clear notice of the arrival of the next bus due to improper trip planning and coordination. Trotro drivers can pick up and drop off passengers anywhere along their routes. This makes it difficult to provide dependable, scheduled transportation that can meet urban needs [1-5]. The absence of a passenger monitoring system that can be integrated with existing transportation systems is a major cause of the aforementioned challenges with Trotro transport systems.

Most computer vision-related research is not oriented towards the Trotro bus-stop scenario, but towards other scenarios such as crowd monitoring, animal monitoring, and vehicle detection. Other studies also focus on accessibility to the bus stop, bus information, or comfort rather than real-time passenger demand for driver decision support. Without an efficient passenger-tracking system, public transport drivers are likely to continue facing challenges such as inefficient route management, reduced revenue generation, and declining passenger approval. The introduction of ride-hailing services such as Uber, Bolt, and Yango in recent years has further changed the transport environment by providing passengers with technologically enabled, predictable, and comfortable alternatives to the traditional Trotro services [6-12].

However, the majority of the populace still prefer the Trotro since they are cheaper. There is therefore the need to infuse smart systems to infuse smart systems to overcome the problems associated with Trotro transport systems. The research presented in this paper introduces a smart IoT-based trotro passenger pickup system to improve local transportation in Ghana. The system consists of a Raspberry Pi 5, a camera module, and a computer vision model trained with YOLOv11, a cloud-based server, and a mobile driver application. The system will count the number of passengers in defined destination areas and send that number to a server. Drivers will be informed of the availability of passengers in a defined location with GPS support. A driver who logs on to the application will have priority to pick up passengers.

The system allows Trotro drivers to optimize their routes, reduce delays, and unnecessary stops by providing them with accurate and timely bus stop and passenger information. This research attempts to provide a solution aimed at affordability and scalability using low-cost hardware, including Raspberry Pi 5 and camera modules, and processing on the cloud [13-17]. Incorporating such smart systems in the Ghanaian public transportation system can make Trotros more efficient and competitive.

This will help bus drivers to access real-time passenger data via a mobile application to be able to make optimal route planning and minimize unnecessary stops [18, 19]. The system will enhance passenger experience and decrease waiting time by ensuring that bus arrivals and passenger demands (pick-up and drop-off) are better coordinated.

Further, the system will reduce overcrowding and improve reliability by cutting down the inefficiencies of the public transport system and facilitating communication between passengers and drivers [20-22]. The novelty of the study is the local adaptation of the technology for the passenger counting application in the local transportation environment in Ghana. The system differs from conventional IoT bus-stop models by linking the passenger count within the bus to the driver's route-selection decisions, using data obtained from camera outputs. The study presents an end-to-end demonstration of passenger detection, communicating with a server, alerting the driver, and communicating with the passengers using audio messages in a Ghanaian Trotro environment as opposed to more general computer vision and IoT research.

The research has been divided into five sections. The background study, problem statements, objectives, scope, and importance of the proposed research are discussed in Section I. In Section II, available literature will be reviewed, emphasizing their architecture, findings, and gaps.

In Section III, engineering design and system development are proposed. Section IV is design implementation and testing. In this section, the prototype deployment and testing are done. Also, the results obtained are discussed. Section V is the conclusion and recommendations. In this chapter, the accomplishments, limitations, and general contributions of the research are presented.

2. Literature Review

A literature review of related work is presented. The review took into consideration related work in integrating smart technologies into transport systems, focusing on the development of Smart IoT-Based systems. Furthermore, the section highlights the strengths and limitations of the available approaches. The following summary and critical examination of the available literature focuses on design procedures and the implementation of systems.

Raccagni et al. [23] designed a precision Automatic Passenger Counting (APC) system with Open Mass Transit (OMT). The suggested framework was divided into two major blocks. The first block was the selection of representative stations according to structural formations and passenger flows. The second block was the analysis of APC performance on an aggregated and disaggregated basis with error measures, such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Symmetric Mean Absolute Percentage Error (SMAPE), with confidence intervals to distinguish between systematic and random errors.

They were able to establish a detailed algorithm that incorporated station selection, data gathering, pre-processing, error determination, pattern estimation, and accuracy verification of APC systems in OMT. The limitation of the research was that data were only collected during certain times, and this may not capture the seasonal or temporal changes in passenger behavior, including fluctuations based on commuting patterns or special events. Lack of financial support and time was also a limitation. Finally, congestion and the availability of light were limitations.

Schutz et al. [24] proposed the utilization of the deep learning algorithm YOLOv4 to detect and track red foxes within an experimental laboratory and under experimental conditions. The research employs video surveillance records of a vaccine experiment to trace the behavior and distribution patterns of foxes in the absence of human intervention to cope with the shortcomings of manual observation. The system includes video data gathering of caged foxes, image extraction and labelling, YOLOv4 training with CSPDarknet53 backbone and PAN, movement analysis based on bounding box positions and vector norms, and multicamera joint analysis using a decision tree.

There was high precision in detecting foxes, with a Mean Average Precision (mAP) of 100. Also, patterns of movement were successfully generated. There were limitations with the detection of infrequently occurring fox positions that were not part of the training set, indicating the importance of an expanded training set. The comparison was made regarding a subset of data in a 450-day experiment, indicating that bigger datasets will be required to set the reference behavior patterns when in captivity conditions.

Chan et al. [25] presented a privacy-preserving framework for estimating the size of inhomogeneous crowds, which are made up of pedestrians moving in varied directions, without utilizing explicit object segmentation or tracking. The system contains crowd segmentation with dynamic texture, the EM algorithm, perspective normalization with weighted pixels, feature extraction, 29 holistic features (segment, edge, texture), and Gaussian Process regression with combined kernels to count per direction of motion. They created a privacy-saving vision system to estimate the size of the crowd

without object recognition and tracking, which could be used on hardware that produces low-level features. They had high accuracy, with 91 and 98 percent of crowd counts within 3 and within 2 people, respectively, of ground-truth accuracy of the "away" and "towards" crowd, respectively, on average of 1.62 and 0.87 people, respectively.

There was difficulty in detecting fast-moving objects such as bicycles, skateboarders, and golf carts. A bigger training set with a more comprehensive representation of outliers is needed to support fast-moving cars, which means that they may be improved in the future.

Gomaa et al. [26] focused on the development of an effective vehicle detection and counting plan in Intelligent Transportation Systems (ITS) based on image processing and computer vision methods. It overcomes the constraint of classical sensor-based approaches, which are expensive and less adaptable. The goal of the approach is to increase the detection and counting accuracy and minimize computational complexity to address the issues of changing light conditions, shadows, and partial occlusions. The system has three steps: YOLO-v2 detection with transfer learning on ResNet-50 every N-frames, feature refinement with a KLT tracker, and K-means clustering and counting with bounding box intersection with a 25-threshold value. A fast and dependable vehicle detection and counting approach was used, and it obtained faster results.

YOLO-v2 might have false positives and, therefore, needs further optimization with K-means clustering and optical flow, which still can fail to cover all detections, provided that the training images do not resemble the test scenes. The strategy concentrates on cars and does not categorize other types of objects (e.g., bicycles, pedestrians), and therefore, it can be applied only in situations with mixed traffic. Rajeswari et al. [27] presented an IoT-enabled greener design of the existing bus stop services, aimed at enhancing the accessibility and comfort of aging, physically challenged, and other vulnerable groups of people in urban areas. It included smart boards, sensors, and a mobile app to deliver real-time bus data, increase inclusivity, and facilitate social interaction using interconnected public spaces (SPA-Spaces).

The system design comprised hardware and software architecture. They created an IoT-powered smart bus stop that improves access for the elderly, people with disabilities, and other vulnerable groups. There were difficulties accessing smartphones or other technology to monitor bus timetables among low-income or vulnerable groups with limited access to technology. Also, there was no clear solution regarding privacy.

3. System Design and Development

The architecture of the passenger pick-up system, as shown in Figure 1, consists of six major components: a passenger counting system, a data management cloud server, a route and bus stop location management microservice, an authentication microservice to manage driver credentials, documents, bus, and location tracking, a driver mobile app, and an object-detection training pipeline. A Raspberry Pi 5 equipped with a camera module captures and processes images using a model to detect passengers at designated Regions Of Interest (ROI) at a bus stop. The localized data is transferred through Wi-Fi to a server hosted at Vercel, where it will be stored and displayed on a map on the Web. The Flutter mobile app, powered by Mapbox maps, will allow drivers to access the real-time location of passenger bus stops.

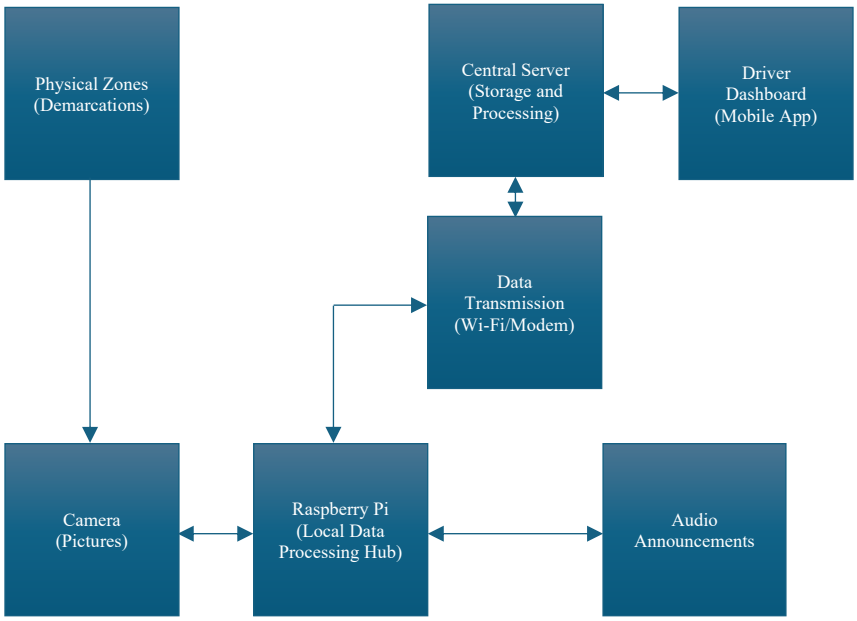


Fig. 1 The architecture of the passenger pickup system

The flowchart in Figure 2 represents the data flow and user interaction within the passenger pickup management system. This begins when a passenger stands in a demarcated area, which causes a camera to take a photo. The imaging system divides the image and identifies and counts the people in each zone. This information is transmitted to a central server that transmits zone-based counts to subscribed drivers on a mobile application.

Unless a driver has an interest in picking up passengers, the server informs waiting passengers through the bus stop speaker. In case a driver is interested, he or she picks a destination through the app. The system then checks whether the number of zones is a minimum number. If not, the process repeats. If yes, the driver will drive to the location, and the passengers can board the vehicle.

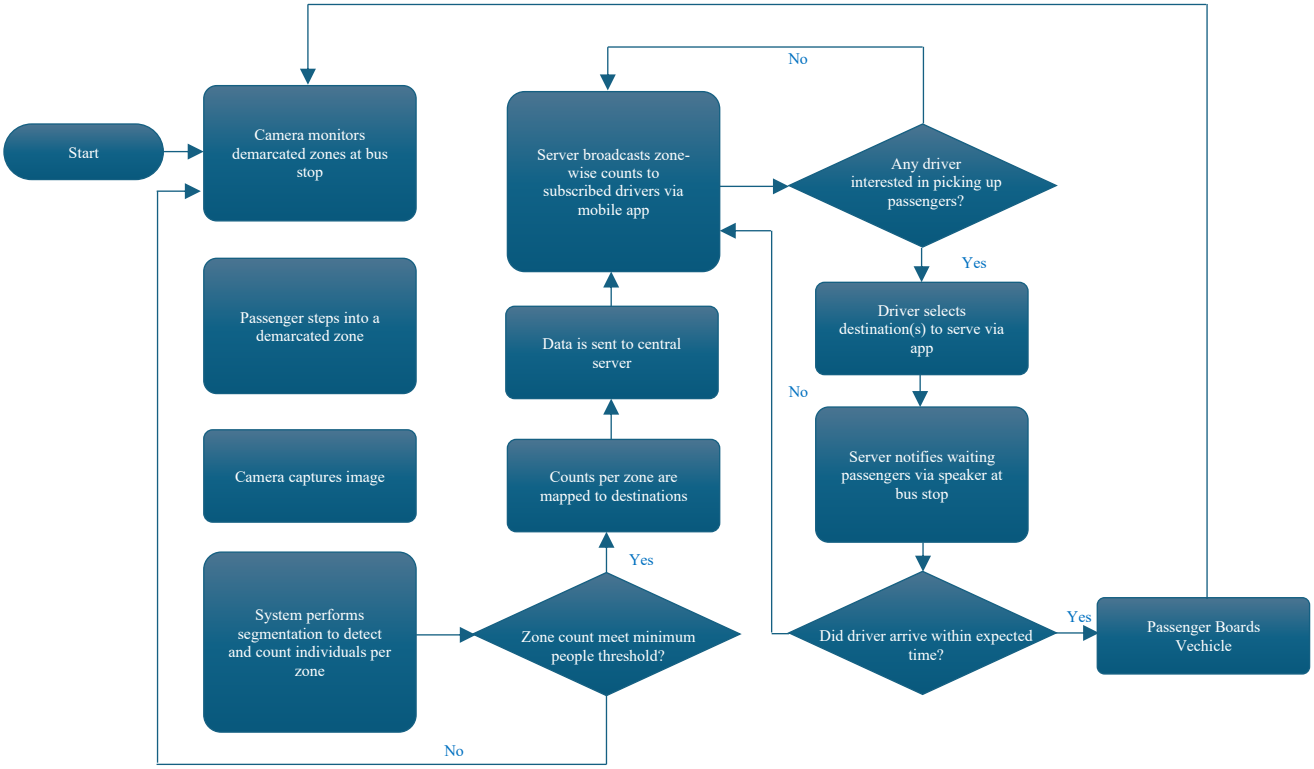


Fig. 2 Data flow and user interaction of the passenger pickup management system

3.1. Requirement Analysis and Specifications

The stakeholder engagement was conducted as part of the initial phase of the smart IoT-Based Trotro passenger pickup system to learn the needs, expectations, and reservations of drivers and passengers. This was done by conducting field interviews with drivers at the Madina bus station and with passengers through Google Forms.

Fig. 3 Stakeholder meeting with drivers

This was aimed at gathering information that would inform the basis for requirement analysis, system design decisions, and adoption strategies. The requirements were classified into functional and non-functional to make sure that the system design is precise, practical, and easy to use. The drivers and passengers were interviewed one after another and completed the forms based on their responses. Commercial drivers are the main end-users of the system because they will directly depend on passenger notifications and route optimization. Quantitative information of twelve 12 willing drivers includes the following:

- 25% of respondents reported that ride-hailing influences the movement of passengers and revenue.
- Full consensus on the knowledge of the number of passengers at the stops, eliciting real-time updates in planning trips, and even on how the system can reduce waiting hours.
- 67 % think it will save fuel.
- 25 % of them are worried about being too reliant on a mobile app.

There were twenty-six (26) respondents from the passenger side. The responses also offer valuable information on the current issues within the area of public transport and how the suggested solution can be expected to influence the issue.

Most respondents said that they used Trotros occasionally, and only a small proportion of them used them daily.

This implies that Trotros are still a popular form of transport, but their inability to be reliable prevents daily use. Below are the results shown in tabular form.

Based on interactions with the passengers, 62% expressedthe view that better organization of the trotro transport system would encourage more people to patronize the service

Table 1. Summary of how often passengers use Trotro

Frequency of Use	Count	Percentage
Once a week or less	11	42%
Several times a week	9	35%
Every day	2	8%
Twice a month	1	4%
Very rarely	1	4%
Never	1	4%
Blank	1	4%

Table 2. Summary of passenger perception of bus stop organization

Stop Organizations	Count	Percentage
Not organized	16	62%
Somewhat organized	9	35%
Very organized	1	4%

Table 3. Passenger opinion of readiness to use Trotro if it is made better

Response	Count	Percentage
Yes	15	58%
Maybe	10	38%
No	1	4%

3.2. System Design and Development Process

The design and development process is divided into hardware and software development. The model development and training aimed at building a robust passenger-counting system that will allow station drivers to track passenger counts to different destinations. The process incorporated SolidWorks modeling of the structural design, physical assembly, and integration testing of functionality. The software development is based on a modular microservices architecture that comprises an edge device, a server backend, a map microservice, an authentication microservice, and a mobile application. The hardware workflow is designed for sequential processing to minimize latency: it begins with camera initialization and image capture, followed by local processing on the Pi, YOLOv11-based object detection, count aggregation, and output to both the Bluetooth speaker and the server.

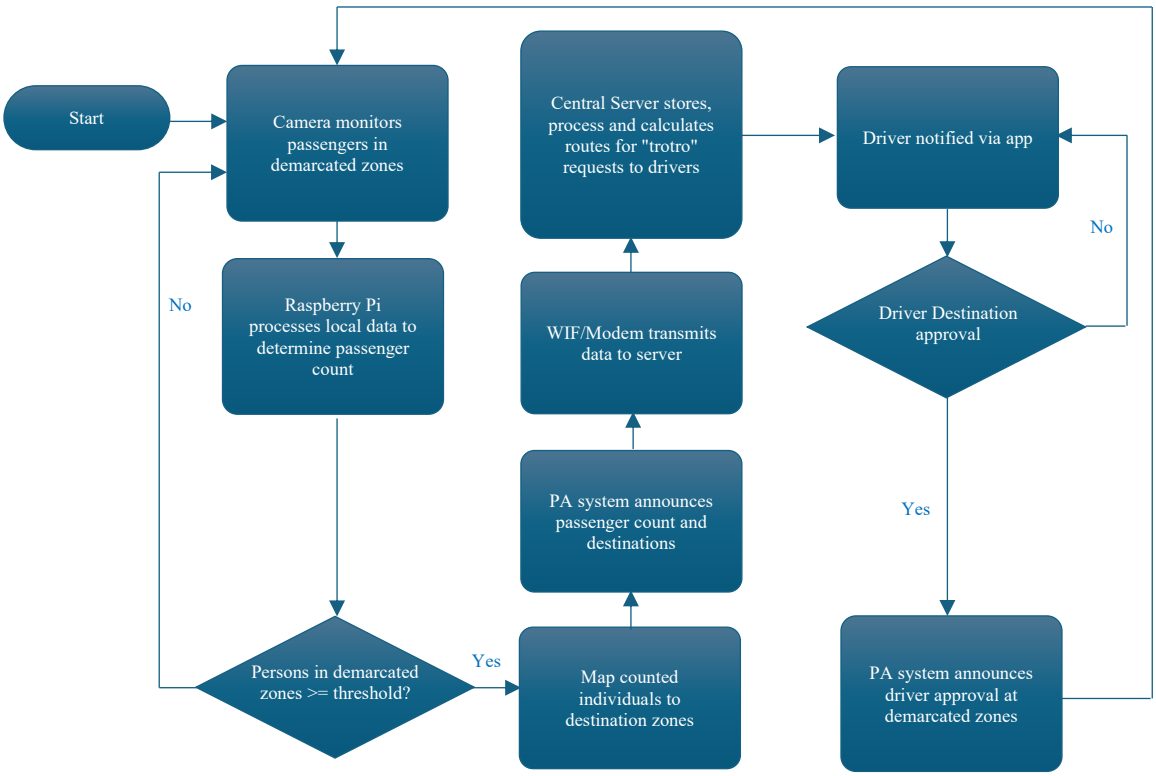


Fig. 4 Diagram of the proposed edge hardware system flow chart

The software development is based on a modular microservices architecture that comprises the following components: edge device, server backend, map microservice, authentication microservice, and mobile application. The edge system uses a FastAPI server serving local image processing and transmitting data. This is shown in Figure 5. The flow chart represents a system in which a camera will constantly survey designated areas. It takes snapshots of images and timestamps and applies an ML model to divide the picture and identify the number of people in each area. When the number of people in any zone reaches a certain level, the numbers, zone ID, and timestamps are sent to a central server through Wi-Fi/LTE. Otherwise, the system waits at a rate limit and restarts monitoring. The counts will also be mapped to the destination ID when the threshold is reached. The web server in Figure 6 handles data storage, visualization, as well as API endpoints using Django on PostgreSQL and Backblaze B2.

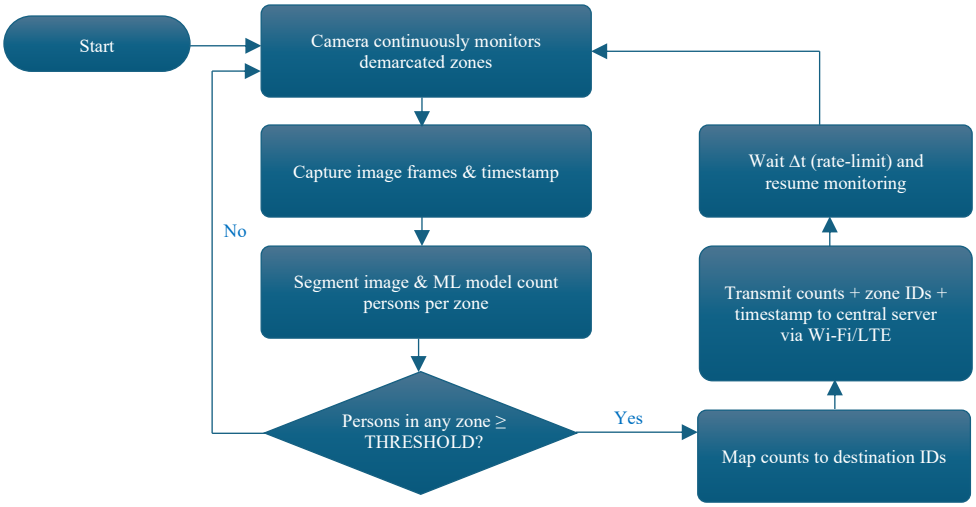


Fig. 5 Diagram of the proposed edge software system chart

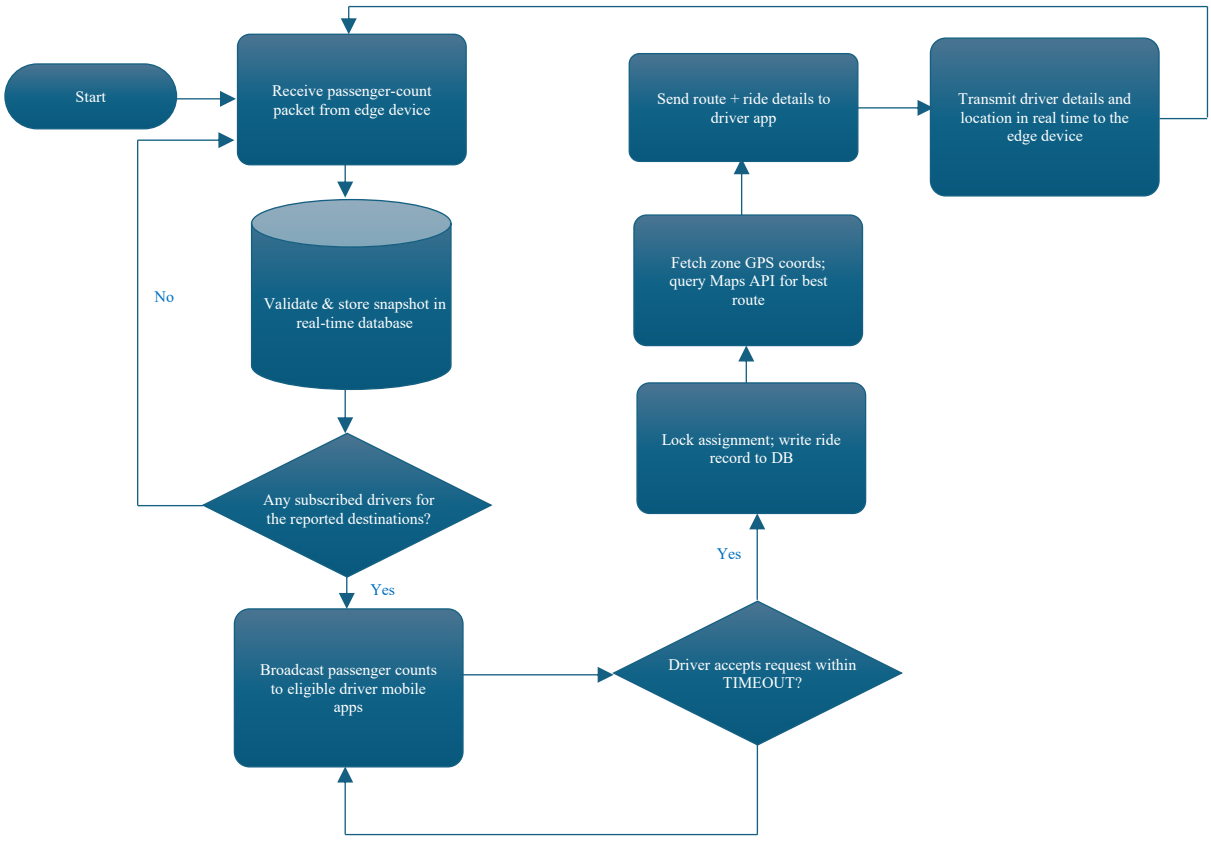


Fig. 6 Diagram of the proposed server software system flowchart

Driver registration, login, along with JWT tokens, which relate to vehicle information (capacity, type), are implemented in the form of a separate Django application to isolate them. The Map Microservice is a FastAPI service that offers geospatial capability to the passenger-counting system. It deals with geocoding, routing, tracking driver location, system demand retrieval, active trip tracking, and matching drivers with passengers, as well as integrating with the authentication microservice and the Django web server. The service has been designed to provide secure, real-time, and reliable GeoJSON format responses to serve the mobile application and the edge devices.

3.3. Theoretical Framework and Conclusive Assumptions

The theory that will be used for the design of the system will not store images but focus on the anonymous counting of passengers.

This model underlines the need to use privacy as a matter of default on technological systems on a case-by-case basis [28]. This will ensure that the privacy of the passengers is protected.

The system adopts a privacy-by-default approach, ensuring that no personal data is collected or stored. Within the Ghanaian public transport context, passengers are accustomed to being visible in public spaces but are generally not accustomed to continuous monitoring or surveillance [29-31].

The information flow is matched to the social norms and expectations of the population. Ghana Data Protection Act, 2012 (Act 843) requires legal, limited, and specific data gathering. The design will be consistent with the principles of data minimization.

3.4. Model Training and Development

This section aims to build a robust passenger-counting system, which will allow the station drivers to keep track of the number of passengers going to different destinations.

The state-of-the-art object detector models like YOLOv9, YOLOv11, RT-DETR, and Faster R-CNN will be used. The results are compared, and the strengths and weaknesses are determined. This will help us to choose the best model to work with. The key objectives are to:

- Process and transform the CrowdHuman dataset into the COCO and YOLO(Ultralytics) formats.
- Train several models, such as head-only and full-visible-body models, to enhance occlusion robustness.
- Compare and contrast models based on precision, recall, and Mean Average Precision (mAP) measures.

3.4.1. Dataset and Annotation Summary:

- Source: CrowdHuman benchmark dataset, developed to detect human faces in a crowded scene with extreme occlusion.
- Annotations: Person (bounding box of visible body) and Head (head-region bounding box).
- Split: Training set: 3,000 images, approximately 130,000 annotations. Validation set: 767 images, approximately 32,000 annotations.

Class mapping: 0 = head, 1 = person

The model and training configuration are shown in Table 4.

Table 4. Model training and configuration

Model	Framework	Backbone	Batch Size
YOLOv11	Ultralytics YOLO	CSP-Darknet	8
YOLOv9	Ultralytics YOLO	CSP-Pelee	8
RT-DETR	Ultralytics YOLO	Hybrid	8
Faster R-CNN	MMDetection	ResNet-50	4

3.4.2. Training Notes

All images were annotated for multilabel classification (head and/or full body). Data augmentation and pre-trained backbone weights were combined, which guaranteed the models would be able to learn features and deliver meaningful evaluations. Table 5 shows the final model comparison based on performance metrics.

Table 5. Final model comparison based on performance metrics

Metric	YOLO v11	YOLO v9	RT-DETR	Faster R-CNN
mAP@50 (all)	0.744	0.706	0.727	0.696
mAP@50–95 (all)	0.482	0.433	0.440	0.390
Precision (all)	0.840	0.764	0.830	-
Recall (all)	0.659	0.630	0.645	-
AP large (FRCNN)	-	-	-	0.514

From Table 5, the following was observed:

- Best by Overall mAP: YOLOv11 has the highest mAP Highest mAP@50 = 0.744 and mAP@50-95 = 0.482.
- Precision-Recall Tradeoff: YOLOv11 and RT-DETR have the highest balance; YOLOv9 is nearly 4% behind on mAP.

- Occlusion Handling: YOLOv11 experiences the largest increase in head recall compared to other models when the head annotation is present.
- Faster R-CNN: It performs well on AP at 50 (0.696) but has low multi-IoU mAP, which means it is not strong at localizing on very strict thresholds.

Four state-of-the-art object detectors were compared in this section on the CrowdHuman dataset. YOLOv11 was the best-balanced and most accurate model, especially when trained with full-body and head annotations with limited resources. Its edge-device optimization and NCNN format also make it easy to run on lightweight devices like the Raspberry Pi 5.

3.5. Detection Algorithm

A multi-stage detection algorithm is applied to refine the number of passengers from YOLOv11 predictions. The region-based counting, density-aware thresholding, head-body association, duplicate removal, and quality checks before uploading counts are included in the processing of the counts of passengers.

The camera image is divided into Regions Of Interest (ROIs) based on destination. Each ROI is a waiting area for a specific destination, such as Taifa, Haatso, Madina, or Unknown. This design is required because it is not just the number of passengers at the bus stop that is important, but rather how many passengers are waiting for a trotro bus to their destinations. The regions are manually set up based on a sample image taken by the edge device. Once calibrated, the Raspberry Pi processes each region individually and determines the number of counts to be sent to each region's destination. The system applies perspective correction to each region before inference, using four corner points if available. This reduces the angle of the camera and provides a more uniform camera angle of view of the waiting area by the detector. If no valid region is found, the system defaults to full-image counting.

The YOLOv11 model identifies heads and bodies in the image, then runs crowd estimation on each region. Studies have shown that density, scale variation, perspective, and occlusion affect crowd counting accuracy [32-34]. To accomplish this, the system that is proposed does not necessarily operate in all scenes in the same way. The system adapts to different scenes using four simple indices: detections, spatial pattern, head-to-body ratio, and average detection distance.

Three scene levels are detected: sparse scenes by body, moderate scenes by both head and body, and dense scenes by body detection. This is consistent with flexible crowd counting methods that use different counting methods in different crowd scenarios. The occlusion is estimated as a ratio of head to body: *occlusion ratio = min(1.0, max(0.0, (head-to-body ratio - 1.0) / 2.0))*. A high value of the occlusion ratio suggests that full bodies are being hidden while heads remain visible. In such cases, the system lowers the confidence threshold and enables head-body fusion.

3.5.1. Head-Body Fusion and Duplicate Removal

A passenger could be detected twice (full body and head). The algorithm distinguishes body and head detections, then performs non-maximum suppression to minimize errors. In the case of multiple bounding boxes that are grouped around the same object, non-maximum suppression preserves the most salient bounding box [35, 36]. At this point, the algorithm determines if each head is part of an existing body that has been identified. It matches heads with existing bodies based on vertical position, horizontal alignment, box size, and containment. If two heads match a body (head and body), the passenger is counted once. If nobody can be matched with a head, the head is considered an independent detection, and the passenger is partially visible. In crowded scenes, head information aids pedestrian detection when partially occluded. The final count is *final count = body detections + unmatched head detections*. The final count is the sum of body detections and unmatched head detections. Spatial clustering combines detections in dense scenes, minimizing double-counting of passengers in overlapping boxes.

3.5.2 Quality Check and Count Upload

A region count is subjected to a quality check before it gets to the server. The check considers various factors of the count, including its physical plausibility based on region size, and consistency with the previous count, as well as the confidence and consistency of the box size of the detections. For crowd counting systems that have different scales and scenes in different images, consistency checks are essential [37]. A reasonable count is returned to the back-end server with destination labels. The driver application receives the processed count of the destination.

The detection logic improves the prototype's reliability but doesn't eliminate counting errors. Passengers can still be misassigned to the wrong destination and counted twice when standing across zone boundaries. The head of a passenger and his/her body can be obscured as well in case of heavy occlusion. However, the multistage pipeline improves the accuracy of the detection process over traditional normal models.

4. Implementation and Testing

Three key steps, namely hardware design, setting up software/microservice architecture, and implementation and physical deployment, were used to develop the system. At the hardware level, the Raspberry Pi 5 (8 GB RAM) was used to perform the local image processing, execute services, and connect with the peripherals. The Raspberry Pi Camera Module 3, with high-resolution and a wide angle, was used to take pictures of waiting passengers to detect and count. A

built-in Bluetooth music box speaker was used to provide audio-based announcements and passenger warnings.

The software architecture is microservice-based, modular, and scalable. On the boundary, inference is done with YOLOv11. Local detection requests are coordinated with a FastAPI service, and the results are synchronized with the web server and are cached.

The backend, written in Django against a PostgreSQL database server, contains the profiles of drivers, routes, trip logs, and passenger numbers. PostGIS was used to extend the locational data management capabilities of PostgreSQL. The authentication service, dealing with driver login and token-based security, and the Map Service (Mapbox), which connects detected passenger counts to bus stop coordinates and destinations to plan routes, are supporting microservices. There was also a mobile application that created an interactive interface, enabling drivers to view the number of passengers at each destination in real-time on a map and make decisions on whether to stop or pass by. The enclosed system was mounted on a pole. The enclosure will protect the system against dust, rain, theft, and any environmental disaster. The pole frame firmly holds the system in such a way that the camera is well placed to capture passengers at the bus stop.

4.1. System Implementation Process

The implementation of the system was done in four (4) phases, namely: API development and integration, software implementation, hardware assembly, and testing and deployment. The implementation started with the development and deployment of REST APIs to establish secure, reliable, and efficient communication among the edge device, server, and mobile platform. This was followed by the implementation of the edge detection models, server-side services, and microservices.

In the hardware implementation, the Raspberry Pi 5, camera, speaker, and power system, a protective enclosure, and a metal pole were assembled. The functionality, reliability, and performance were tested to ascertain their functionality. The system was designed as a collection of four microservices, with each portion responsible for specific functionality.

The Django Backend helped in passenger detection and trip management. The FastAPI Edge Server runs the YOLOv11-powered image processing model deployed on the edge device for real-time passenger detection. The Node.js Authentication Service handles driver authentication and management, while the FastAPI Map Microservice provides geospatial services, including location processing and map-based operations.

Mapbox was used to provide real-time routing. Administrators can directly communicate with the dashboard and manage trips, monitor passenger counts, and review driver data. The server also interacts with other microservices such as the Node.js Authentication Service, the FastAPI Map Microservice, and the FastAPI Edge Server, and enables smooth integration throughout the system. The Django Backend server will have its own frontend interface where the administrators will be able to view the activity in the system, configure the system, and view the number of passengers in real-time.

The dashboard is built in an easy-to-use format, which allows a user to access major metrics and active trips. The frontend interface (consisting of the login page, dashboard overview, and passenger count visualization) is shown in Figures 7 and 8.

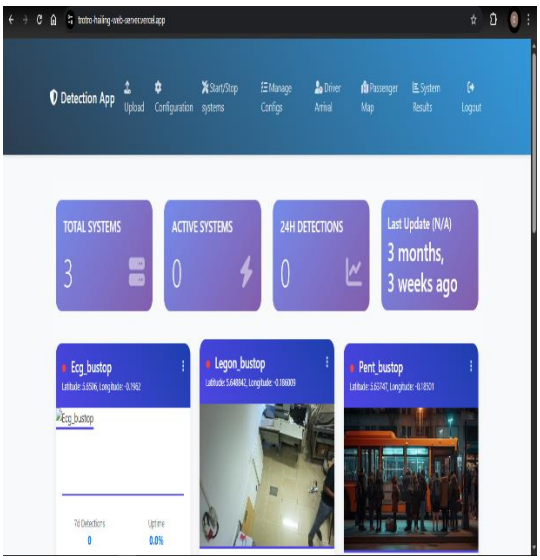


Fig. 7 An admin dashboard that gives a general overview of existing systems at bus stops

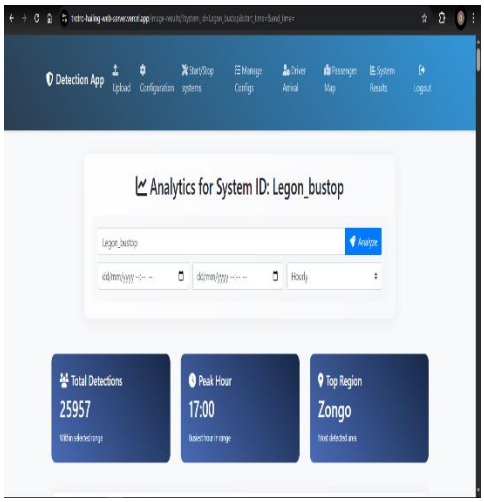


Fig. 8 An admin page for passenger count data analysis

The FastAPI Edge Server is powered on edge devices (e.g., Raspberry Pi 5) and processes AI-driven passenger detection with camera feeds. It scans images, counts the number of passengers, and transmits the data to the Django Backend. Audio announcements are also initiated by the server over the connected Bluetooth speaker. It communicates directly with the Django server to store the data, with the Map Microservice to make decisions, and with the authentication service to notify the drivers with specific information. The authentication service deals with the registration of drivers, the login process, the management of the vehicle, and the tracking of the vehicle. It makes sure that only authenticated drivers can access the system and gives secure tokens to access the rest of the services, such as the Map Microservice and the Django Backend. It also has driver-specific data, which aids in route matching and assigning passengers. The Map Microservice oversees geospatial operations, such as matching drivers and passengers, routing, and geocoding. It transforms the information on the Backend into GeoJSON to present active trips and passenger demand in real-time on the mobile application. The service communicates with the Django Backend to get trip information, the authentication service to get driver position, and the Edge Server to get real-time information. Microservices are connected by means of RESTful APIs by applying secure authentication. In the Edge–Django, POST requests are made to send passenger counts and pictures. Ngrok and offline caching ensure reliability. The Check-in Map Service provides the position of drivers to match on a route based on JWT/X-API-KEY. The Django-Map Service will pass the data on passenger demand and active trips to be visualized and routed. The Mapbox Integration will offer geocoding, route calculation, and dynamic driver-passenger matching. To validate the API functionality, integration, and security, Postman was used. The results for the API endpoints for login and driver registration are shown in Figures 9(a) and 9(b).

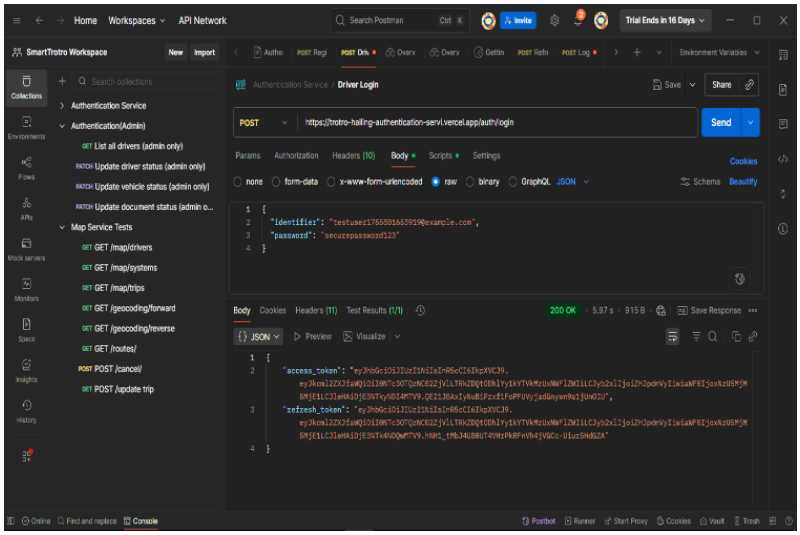


Fig. 9(a) Testing of the API endpoint for login

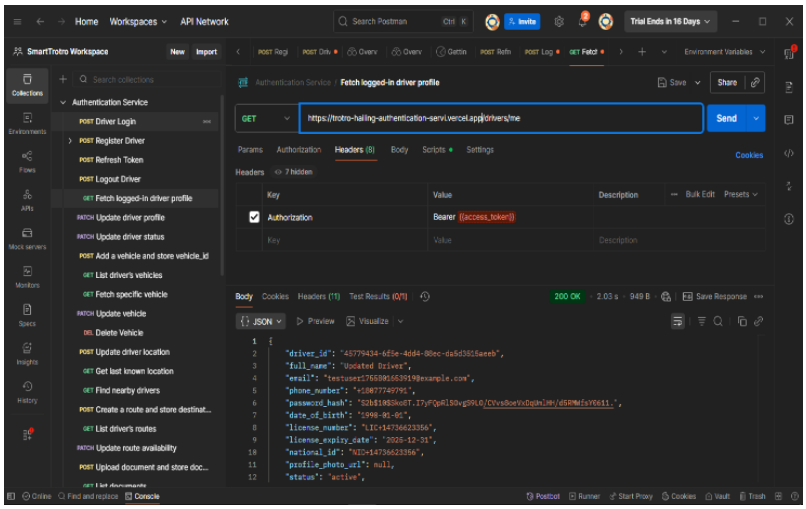


Fig. 9(b) Testing the API endpoint for driver registration.

The mobile application has Authentication Interfaces, Home and Dashboard, Trip Management Interfaces, Profile Interfaces, Vehicle Management Interfaces, Document Management Interfaces, Widget Organization, Responsive Layout Rules, File Structure, and Classes and Walkthroughs. Figures 10(a)-(c) show some of the implemented mobile App pages.

The splash screen shows the app logo and branding and preloads resources. The login screen provides an email or phone number, a password, and input validation and error. The signup Screen gathers the information of a new user, like name, contact, and password. The password reset screen allows users to safely change their credentials. The home screen is the main hub with an embedded Mapbox map. It is also here that the user can request rides, see the vehicles near them, and access other modules in the app. The navigation drawer or bottom navigation navigates directly to profile, documents, vehicles, settings, and trip history. The design process illustrates why modular organization (MVVM with Riverpod) can be used to create clean, testable, and scalable Flutter apps. All features (auth, trips, vehicles, documents) are separated by their own repository and view model. Reusable common elements (buttons, input fields, snackbars) help to maintain the consistency of the UI throughout the application.

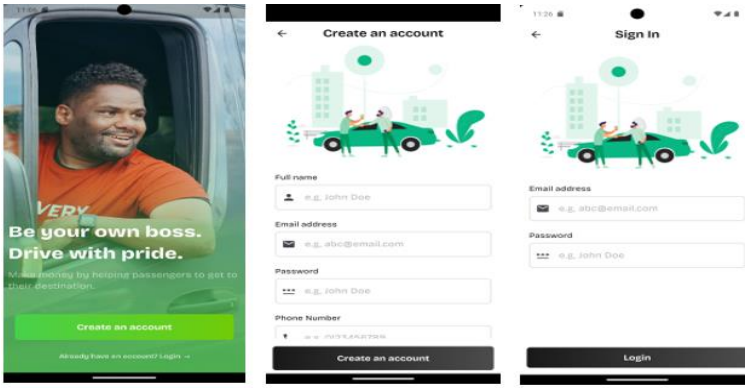


Fig. 10(a) Landing, signup, and Sign-in screen

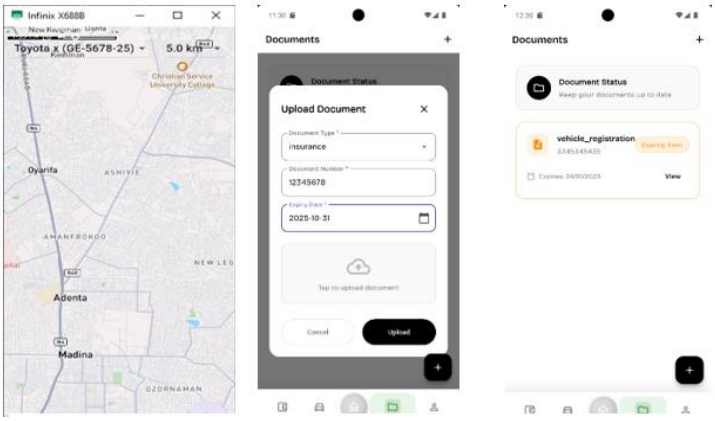


Fig. 10(b) Driver map home page and document uphold screen

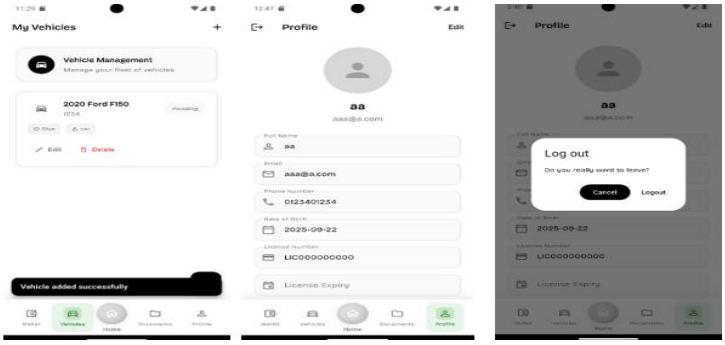


Fig. 10(c) Driver vehicle, profile, and logout screens.

4.2. Hardware Implementation

The system hardware implementation used a Raspberry PI 5, a Camera Module 3, FPC cables, power banks, and Bluetooth speakers mounted on a metal pole. The pole safely supports the system enclosure and provides the best field of view for the camera. A casing to physically protect the Raspberry Pi, power bank, camera, and speaker, and to manage cables, was constructed. A junction box and homemade brackets were used to mount the system on a pole. The enclosure was tilted at an angle, which led to the most effective field of view of the camera, which could effectively detect passengers.

Fig. 11(a) Mounting the system on the pole

Fig. 11(b) A mounted system.

Various heights were experimented with, finding the optimal height that captures the largest area with minimal occlusion. After checking the views of the various heights, a height of 2.97m gave us the best aerial view that allowed the camera to capture passengers appropriately and prevent issues such as occlusion.

4.3. System Testing

In order to make the test site appear like a real bus stop setup, some demarcations of the various passenger destinations were made clear. A set of zones of interest was developed within the field of view of the Raspberry Pi camera: 120°, and each zone of interest represented one of the main destinations of the passengers. In this test, four regions were determined: Taifa, Haatso, Madina, and Unknown. The Unknown area was allocated to people who could not be categorized within the major destinations, allowing the drivers to choose whether to pick them up. Physical markers were used to draw ground demarcations, but digital zones were set up on the web interface of the system. Below are pictures of the configuration and pictures of the physical demarcations.

Fig. 12 Creating various regions of interest that correspond to several destinations

The Raspberry Pi was remotely activated by the server, which took a picture of the bus stop zone. Segmentation values, including the width and height of each region, were defined using this image and sent back to the edge system. These coordinates were then applied by the Pi to subdivide captured images in real-time and place the individuals identified in the correct areas.

The fetch image button triggers the edge system remotely to take a sample picture of its area and send it to the server for configuration. The configured area for each region on the administrator dashboard will be sent and stored on the Raspberry Pi edge system. A preview of several ROIs for each destination was used for the image segmentation and processing of passenger count, as displayed on the admin dashboard. After the demarcation process and system setups were done, volunteers were stationed in the different regions to test whether the system was able to identify and count people correctly.

The numbers were then sent to the central server and shown on the mobile driver application. Viewers of the app might see the passengers at each destination in real-time, choose the desirable passengers, and track the route to the stop on the map.

Fig. 13 Five (5) volunteers evenly distributed

Scenario 1 – Normal Case A

- Test 1 setup with 5 volunteers in demarcated areas (Taifa, Haatso, Madina, Unknown).
- Volunteers were evenly distributed across the zones.
- No occlusion or boundary interference occurred.

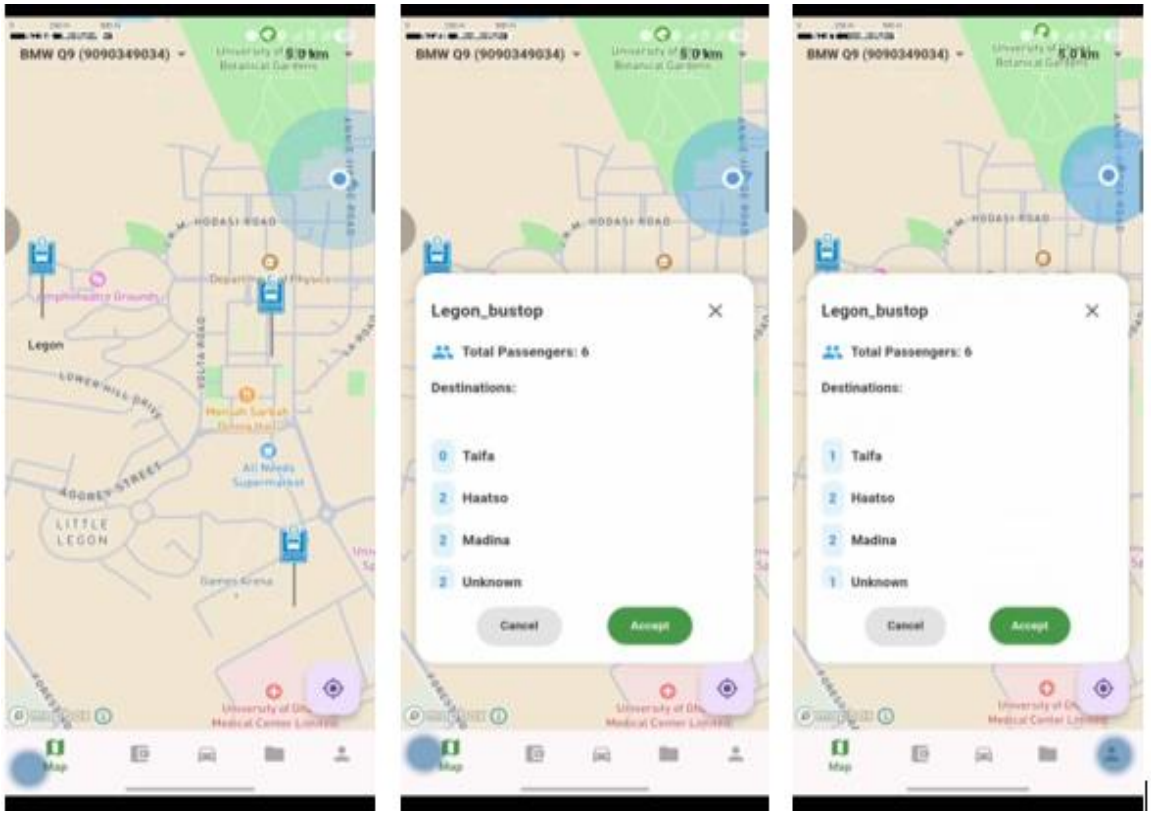
Scenario 2 – Normal Case B

- Test 2 setup with volunteers spaced across destinations.
- The Taifa region had no passengers, while other destinations had exactly 2 passengers each.
- This scenario tested the system's ability to avoid false positives in empty zones.

Scenario 3 – Edge Case (Boundary and Occlusion)

- Test 3 setup with 6 volunteers.
- One passenger stood at a boundary between Taifa and Haatso, while another was partially occluded by a volunteer in front.
- Designed to test robustness in real-world conditions.

4.3.1. Test Scenarios



(a) **(b)** **(c)**

Fig. 14(a) Bus Stop Map Output for Test 1, (b) Bus Stop Map Output for Test 2, and (c) Bus Stop Map Output for Test 3.

Result A - Normal Case A

- Figure 14(a): Bus stop map output for Test 1.
- The system was able to detect all five (5) passengers, and the exact counts were displayed per destination.
- The accuracy rate was 100% across all zones.

Result B - Normal Case B

- The output bus stop map for Test 2 is shown in Figure 14(b). The demarcated region labelled Taifa was correctly reported as 0 passengers.

- Two passengers each were identified in the zones demarcated Haatso and Madina, which are consistent with ground truth.
- Accuracy: 100%, confirming reliable zone-specific detection.

Result C – Edge Case

- The output bus stop map for Test 3 is shown in Figure 14(c).
- The passengers at the boundary were double-counted for both Taifa and Haatso.
- In the Unknown zone, 2 passengers were present, but only 1 was detected due to occlusion.
- Accuracy: Reduced due to overlapping and visibility issues.

4.4. Code for Capturing Images: capture_image.py

```
import os
import time
import subprocess
from config. settings import LOG_DIR
def capture_image(output_path):
    try:
        # Ensure image directory exists
        os.makedirs(os.path.dirname(output_path),
exist_ok=True)
        # Command to capture an image using libcamera-still
        cmd = [
            "libcamera-still",
            "-o", output_path,
            "--width", "1280",
            "--height", "720",
            "--nopreview",
            "-t", "1000"  # wait 1 second for camera to warm up
        ]
        # Run the command
        result = subprocess.run(cmd, capture_output=True,
text=True)
        if result.returncode != 0:
            raise Exception(f"libcamera-still failed:
{result.stderr.strip()}")
        return True
    except Exception as e:
    os.makedirs(LOG_DIR, exist_ok=True)
    with open(os.path.join(LOG_DIR, "errors.log"), "a") as f:
            f.write(f"[ERROR] {time.ctime()}: capture_image:
{str(e)}\n")

return False
```

The system also incorporated driver announcements in addition to the number of passengers. Whenever a driver picked a trip, the system sent a notification to confirm that he or she had plans to pick up passengers. Equally, a cancellation caused the notice on the trip to be cancelled. These functions were experimented with and confirmed to provide reliable communication among the passengers, servers, and drivers.

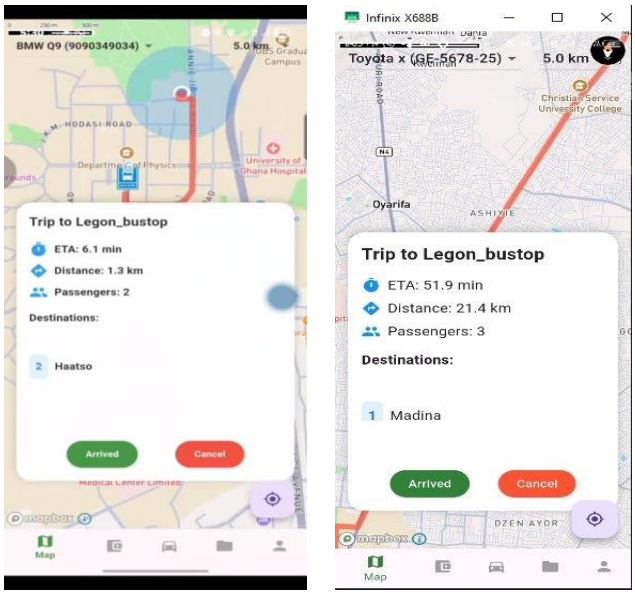


Fig. 15 Confirmed trip indicating ETA distance, and route to bus stop on the map

Map route connecting the driver to the bus stop after the trip has been confirmed, indicating estimated distance and time based on different driver locations. The experiment conducted in normal and edge case conditions shows that the passenger detection system is capable of high accuracy in ideal test conditions. With the system, both in Normal Case A and Normal Case B, the system was correct. Passengers in each destination zone were accurately recognized, and no false positives in empty zones, like Taifa, were obtained. This establishes the accuracy of the system in cases where passengers are well spread and not hidden or encroached by regional demarcation borders.

Occupancies of passengers within the boundaries of the zones were at risk of being counted twice. In a controlled environment, the system worked well. This will not be the same in a crowded area with fast movement patterns. Drivers could view the actual counts at each destination, the estimated trip time, and the routing instructions. The system was reliable, under usual test conditions, with the ability to match the expected passenger numbers in every zone. Although the performance was good in controlled conditions, the errors of counting boundaries and occlusion decreased the accuracy in the edge test.

Despite the introduction of offline caching, the performance of the system remains reliant on the availability of stable internet to pass counts to the backend and update driver applications. The prototype was tested in a controlled location. In a real bus stop scenario, accuracy may be affected due to random movement and non-compliant passengers.

4.5. Statistical Analysis of Prototype Evaluation

The evaluation of the prototype was done using some selected volunteers to test the functionality and runtime logs from the edge device. The accuracy of the passenger count was tested in destination-based zones. The logs assessed upload reliability, response time, heartbeat status, and resource use. Table 6 shows the results obtained from the controlled passenger count.

Table 6. Results obtained from controlled passenger count

Case	Setup	Outcome
Normal Zone A	5 volunteers across the zones	All passengers were detected correctly
Normal Zone B	Taifa is empty; other zones are occupied	Empty zone reported 0; occupied zones matched the expected counts
Edge Zone	6 volunteers with boundary overlap and occlusion	Boundary double-count and one missed the Unknown-zone passenger

When the volunteers were clearly visible and positioned within their assigned zones, the two normal cases showed correct counting. This shows that boundary overlap can cause double-counting, while occlusion can cause missed detections. This confirms that the system is reliable under controlled visibility. It requires better boundary handling and field calibration for crowded bus-stop conditions.

Table 7. Runtime performance summary

Metric	Result
Upload success rate	100%
Upload delay	4.39 +/- 0.70 s
Upload-delay range	3.17-7.72 s
Connected heartbeats	118/118
Mean CPU / memory use	60.23% / 68.97%

The counts from the edge device were submitted successfully. The mean count-to-upload delay in the case of the prototype was 4.39 s, indicating that the prototype can transmit count information in a short response time in controlled operation. Despite the promising results obtained from the system, some constraints and limitations affected the performance of the system. The limitations include:

- Ambiguity at the boundary: The passengers who were in the boundary areas were counted twice.
- Occlusion Sensitivity: The detection model fails to correctly register all the people when standing in front of each other. But this is one of the typical weaknesses of computer vision models.
- Environmental Conditions: Conditions such as visibility, shadows, weather changes, etc., would have a significant impact on the real-world deployment of the system.
- Hardware Limits: High-performing devices capable of handling high-level computations are required. Larger models require more extensive optimization to achieve efficient performance.
- Deployment Constraints: The prototype was tested with volunteer subjects in a controlled environment.
- The performance and accuracy will be affected during field deployment due to the random movement of the passengers.

In all, the system showed that a computer vision model, IoT hardware, and mobile applications can be integrated to provide solutions for our local transport systems.

5. Conclusion and Recommendation

In this research, A smart passenger monitoring system was designed and developed to improve local transportation in Ghana. The system offers an affordable, practical, and smart solution to solving the inefficiencies that exist in our local transport. Edge AI was integrated with cloud microservices and user-facing applications.

The research has proven that such a system is not only technically feasible but also socially effective in terms of urban mobility. The research shows how edge AI (YOLOv11 on Raspberry Pi) can be utilized to detect people in real time and reduce waiting time and congestion at the bus stop. The research presents a replicable and scalable model to be applied in other cities, which will make transport systems smarter in Ghana and elsewhere.

It is essential to have stable internet connectivity to synchronize data. Lighting conditions, weather conditions, and the location of cameras affected the accuracy of detection. To improve the system and make it ready for large-scale deployment, YOLOv11 can be trained on a more varied dataset that includes edge cases to reduce misclassification. The process of zone demarcation can be enhanced, such as through dynamic geo-fencing or adaptive clustering, to reduce the number of boundary errors. The driver app could be extended to enable additional features such as demand forecasting, historical analytics, and passenger safety alerts. In general, the project has proven that such a system is not only technically feasible but also socially effective in terms of urban mobility.

Funding Statement

This research was funded by the authors.

Acknowledgements

We are grateful to Dr. Nii Longdon Sowah for his valuable advice and support.

References

[1] Lukas Siegfried Hoose, "*Finding Ways to Move through Accra´S Traffic: A Study about Urban Transportation in One of Africa's Fastest-Growing Cities*," Master's Thesis, The University of Bergen, 2022. [Google Scholar] [Publisher Link]

[2] Emmanuel Dzisi, Daniel Atuah Obeng, and Yaw Adubofour Tuffour, "Modifying the SERVPERF to Assess Paratransit Minibus Taxis, *Trotro* in Ghana, and the Relevance of Mobility-as-a-Service Features to the Service," *Heliyon*, vol. 7, no. 5, pp. 1-9, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[3] William Agyemang, "Measurement of Service Quality of "Trotro" as Public Transportation in Ghana: A Case Study of the City of Kumasi," *Southern African Transport Conferences*, pp. 283-291, 2013. [Google Scholar] [Publisher Link]

[4] Emmanuel Dzisi et al., "Digitalization of the Paratransit (*Trotro*) using Mobility as a Service: What are the Adoption Intentions of Operators and Operator Unions in Ghana?," *Research in Transportation Business and Management*, vol. 47, pp. 1-15, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[5] Joseph Kofi Wireko, "On-Demand Internet-based Business Model in Informal Public Transportation System in Developing Countries," *SSRN*, pp. 1-14, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[6] Cynthia Barnhart, Douglas Fearing, and Vikrant Vaze, "Modeling Passenger Travel and Delays in the National Air Transportation System," *Operations Research*," vol. 62, no. 3, pp. 580-601, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[7] Marieke S. Van Der Tuin, and Adam J. Pel, "The Disruption Transport Model: Computing user Delays Resulting from Infrastructure Failures for Multi-Modal Passenger and Freight Traffic," *European Transport Research Review*, vol. 12, no. 1, pp. 1-10, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[8] Abderraouf Boussif, and Mohamed Ghazel, "Model-based Monitoring of a Train Passenger Access System," *IEEE Access*, vol. 6, pp. 41619-41632, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[9] Xing-Gang Luo et al., "A New Framework of Intelligent Public Transportation Systems based on the Internet of Things," *IEEE Access*, vol. 7, pp. 55290-55304, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[10] Fahad Younas, "*Factors Influencing Consumer Choice in Ride-Sharing Platforms: A Study on Uber and Bolt in Helsinki*," Centria University of Applied Sciences, 2025. [Google Scholar] [Publisher Link]

[11] Hampus Johansson, and Sebastian Leijen, "*Competing for the Curb: A Study of the Impact of Ride-Hailing Companies on Traditional Taxi Services*," Master Theses, University of Gothenburg, 2023. [Google Scholar] [Publisher Link]

[12] Dominic Edem Hotor, "Accessibility and use of Public Transport Services by Persons with Disabilities (PWDS) in Ghana," *SSRN*, pp. 1-284, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[13] Aleksander Radovan et al., "Review of Passenger Counting in Public Transport Concepts with Solution Proposal based on Image Processing and Machine Learning," *Eng*, vol. 5, no. 4, pp. 3284-3315, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] Aktumar Rakhymova et al., "Development of an Intelligent Passenger Counting System for Enhancing Public Transport Efficiency and Optimizing Route Networks," *Journal of Problems in Computer Science and Information Technologies*, vol. 2, no. 1, pp. 3-14, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[15] Adel Nadjaran Toosi, Jungmin Son, and Rajkumar Buyya, "Clouds-Pi: A Low-Cost Raspberry Pi-based Micro Data Center for Software-Defined Cloud Computing," *IEEE Cloud Computing*, vol. 5, no. 5, pp. 81-91, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[16] Fung Po Tso et al., "The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infrastructures," *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, Philadelphia, PA, USA, pp. 108-112, 2013. [CrossRef] [Google Scholar] [Publisher Link]

[17] Sheik Murad Hassan Anik et al., "A Cost-Effective, Scalable, and Portable IoT Data Infrastructure for Indoor Environment Sensing," *Journal of Building Engineering*, vol. 49, pp. 1-15, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] Kai Lu et al., "A Review of Big Data Applications in Urban Transit Systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 5, pp. 2535-2552, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[19] Carlos Romero et al., "Potential Demand for Bus Commuting Trips in Metropolitan Corridors through the use of Real-Time Information Tools," *International Journal of Sustainable Transportation*, vol. 16, no. 4, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[20] M. Rohith, Ajeet Sunil, and Mohana, "Comparative Analysis of Edge Computing and Edge Devices: Key Technology in IoT and Computer Vision Applications," *2021 International Conference on Recent Trends on Electronics, Information, Communication and Technology*, Bangalore, India, pp. 722-727, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Lavanya Sharma, *Rise of Computer Vision and Internet of Things*, 1st ed., Chapman and Hall/CRC, 2022. [Google Scholar] [Publisher Link]

[22] Nashwan Adnan Othman, and Ilhan Aydin, "A New IoT Combined Body Detection of People by using Computer Vision for a Security Application," *2017 9th International Conference on Computational Intelligence and Communication Networks*, Girne, Northern Cyprus, pp. 108-112, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[23] Benedetto Barabino, Lorenzo Raccagni, and Roberto Ventura, "Accuracy of Automatic Passenger Counting in Open Mass Transit Systems: Insights from Italy," *The Open Transportation Journal*, vol. 9, no. 1, pp. 1-15, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[24] Anne K. Schütz et al., "Application of Yolov4 for Detection and Motion Monitoring of Red Foxes," *Animals*, vol. 11, no. 6, pp. 1-18, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[25] Antoni B. Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos, "Privacy-Preserving Crowd Monitoring: Counting People without People Models or Tracking," *2008 IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, AK, USA, pp. 1-7, 2008. [CrossRef] [Google Scholar] [Publisher Link]

[26] Ahmed Gomaa et al., "Faster CNN-based Vehicle Detection and Counting Strategy for Fixed Camera Scenes," *Multimedia Tools and Applications*, vol. 81, no. 18, pp. 25443-25471, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[27] S. Rajeswari, and L. Priyadharshini, "Smart Model Bus Stop for Special Needy Aged Individuals," *International Journal of Smart Applications and Technologies*, vol. 16, no. 1, pp. 1-12, 2025. [CrossRef] [Publisher Link]

[28] Arianna Alessi et al., "*Privacy by Design and by Default in Software Development in Order to Prevent Unlawful Processing of Personal Data*," Privacy Certifications Impact on Software Development and Liabilities, pp. 1-56, 2021. [Google Scholar]

[29] E.F. Sam, and A.M. Abane, "Enhancing Passenger Safety and Security in Ghana: Appraising Public Transport Operators' Recent Interventions," *Journal of Science and Technology (Ghana)*, vol. 37, no. 1, pp. 101-112, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[30] Jennifer Hart, *Informality, Urban Transport Infrastructure, and the Lessons of History in Accra, Ghana*, Routledge Handbook of Urban Planning in Africa, pp. 320-340, 2019. [Google Scholar]

[31] Enoch F. Sam et al., "Public Bus Passenger Safety Evaluations in Ghana: A Phenomenological Constructivist Exploration," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 58, pp. 339-350, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[32] Amanda Amalia Lubis, Adrian Prasasta, and Dewi Anita Sari, "Towards Efficient Crowd Counting and Behavior Analysis using YOLOv11," *International Journal of Technology and Modeling*, vol. 4, no. 1, pp. 35-47, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[33] Emanuele Frontoni, Marina Paolanti, and Rocco Pietrini, *People Counting in Crowded Environment and Re-Identification*, RGB-D Image Analysis and Processing, Springer, pp. 397-425, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[34] Vikas Kamra et al., "A Novel Approach for Crowd Analysis and Density Estimation by using Machine Learning Techniques," *2024 International Conference on Intelligent Systems for Cybersecurity*, Gurugram, India, pp. 1-6, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[35] Meiling Gong et al., "A Review of Non-Maximum Suppression Algorithms for Deep Learning Target Detection," *Seventh Symposium on Novel Photoelectronic Detection Technology and Applications*, vol. 11763, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[36] Vinod Biradar, and Karuna Chandrashekhar Gull, "Multiple Object Detection and Tracking using Deep Learning Framework with Non-Maximum Suppression," *International Journal of Intelligent Engineering and Systems*, vol. 18, no. 1, pp. 106-117, 2025. [CrossRef] [Google Scholar]

[37] Zan Shen et al., "Crowd Counting Via Adversarial Cross-Scale Consistency Pursuit," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, pp. 5245-5254, 2018. [CrossRef] [Google Scholar] [Publisher Link]