***Report of the Project Done in Partial Fulfilment of the Requirements for the Award of Degree of Bachelor of Technology in Electronics and Communication Engineering***

**SMARTFLOW: DENSITY BASED TRAFFIC MANAGEMENT SYSTEM**

**PROJECT PHASE II REPORT**

*Done by*

**Amaljith T P, Reg No: AJC21EC015**

**Binu Ajmal Shah, Reg No: AJC21EC034**

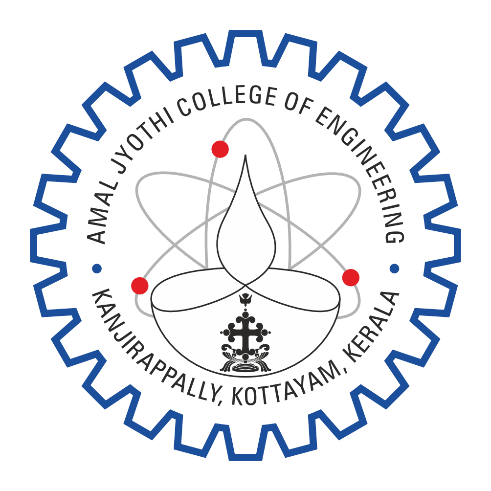
**Denil C Varghese, Reg No: AJC21EC038**

**Joel Jackson John, Reg No: AJC21EC048**

*Under the guidance of*

**Mr. Jaison C S**

Asst. Professor, Dept of ECE

****

**Bachelor of Technology**

**in**

**Electronics & Communication Engineering**

**Department of Electronics & Communication Engineering**

**Amal Jyothi College of Engineering (Autonomous)**

**(Affiliated to APJ Abdul Kalam Technological University, Thiruvananthapuram)**

**Kanjirappally – 686518**

**MARCH – 2025**

**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING**

**AMAL JYOTHI COLLEGE OF ENGINEERING (AUTONOMOUS)**

**(Affiliated to APJ Abdul Kalam Technological University)**

**Kanjirappally – 686518**

A logo with a white hat with a black background

Description automatically generated with medium confidence

**BONAFIDE CERTIFICATE**

This is to certify that the project phase II report entitled **“SmartFlow: Density Based Traffic Management System”** is a Bonafide report of the Eighth Semester Project Phase II [ECD416] done by **Amaljith T P (Reg No: AJC21EC015), Binu Ajmal Shah (Reg No: AJC21EC034), Denil C Varghese (Reg No: AJC21EC038), Joel Jackson John (Reg No: AJC21EC048)** in partial fulfilment of requirements for the award of degree of Bachelor of Technology in Electronics and Communication Engineering from APJ Abdul Kalam Technological University, on March 2025. They have done the phase II of the project with prior approval from the department.

**Mr. Jaison C S Mr. Shinto Sebastian**

Project Guide Project Coordinator

Asst. Professor Asst. Professor

**Dr. Geevarghese Titus**

Head of the Department

# **ACKNOWLEDGEMENT**

First, we sincerely thank the **Almighty** who is most beneficent and merciful for giving me the knowledge and courage to complete the preliminary work of the project successfully.

We derive immense pleasure in expressing our sincere thanks to the Principal **Dr. Lillykutty Jacob,** for her permission and for providing infrastructural facilities for the successful completion of our preliminary work of the project.

We express our gratitude to **Dr. Geevarghese Titus**, Head of the Department, ECE, for his encouragement and motivation during our preliminary work of the project. We express our heartfelt gratefulness to **Mr. Jaison C S,** Assistant Professor, Department of ECE, project guide, for his valuable guidance and suggestions during the project.

We also extend our sincere thanks to **Dr. Geevarghese Titus,** Professor **and Mr. Shinto Sebastian,** Assistant Professor, Department of ECE for their kind support and coordination.

Finally, we appreciate the patience and solid support of our parents and enthusiastic friends for their encouragement and moral support for this effort.

Amaljith T P

Binu Ajmal Shah

Denil C Varghese

Joel Jackson John

# **ABSTRACT**

Traffic congestion is increasing in cities and is responsible for increased travel times, fuel wastage, and pollution. The traditional traffic control systems are based on pre-determined fixed timings, which are not dynamic to actual traffic levels and therefore introduce inefficiencies. Emergency responders do not easily travel through congested traffic, creating delay in the critical time needed to attend emergency calls. Current automated traffic control systems employ sensors such as infrared (IR) and ultrasonic sensors to track vehicle concentration, but these methods require extra hardware expenses and intricate installation. An AI-based system that can make use of available surveillance infrastructure to manage traffic flow and enhance road efficiency in a smart manner is needed.

This work introduces a Density based Traffic Management System that utilizes Raspberry Pi and Pi Camera to record live traffic images and deep learning algorithms for vehicle detection and density calculations. The system dynamically optimizes traffic signal time based on the level of congestion to ensure smooth traffic flow. A backend based on Flask allows interaction among the processing unit, a MySQL database, a user and emergency vehicle mobile app, and an admin web portal. The Flutter-based and Visual Studio Code-implemented mobile app offers users real-time traffic updates, rerouting based on congestion, and emergency vehicle clearance functionalities. The web portal allows traffic authorities to view live data, manage system alerts, and override signals as needed. Through the combination of AI and already existing surveillance cameras, this system presents a cost-saving, scalable, and effective urban traffic management solution, enhancing response times during emergencies and alleviating congestion-induced pollution.

**Table of Contents**

[**i.** **ACKNOWLEDGEMENT** i](#_Toc193796817)

[**ii.** **ABSTRACT** ii](#_Toc193796818)

[**1.** **INTRODUCTION** 1](#_Toc193796819)

[**2.** **MOTIVATION AND PROBLEM STATEMENT** 2](#_Toc193796820)

[**2.1.** **Motive of the Project** 2](#_Toc193796823)

[**2.2.** **Objectives of the project** 3](#_Toc193796824)

[**3.** **LITRATURE SURVEY** 5](#_Toc193796825)

[**4.** **IMPLEMENTATION DETAILS** 17](#_Toc193796826)

[**4.1.** **DESCRIPTION OF SYSTEM COMPONENTS** 17](#_Toc193796828)

[**4.1.1.** **Raspberry PI & Pi Camera** 17](#_Toc193796829)

[**4.2.** **DESCRIPTION OF SOFTWARE USED** 19](#_Toc193796830)

[**4.2.1.** **Visual Studio Code** 19](#_Toc193796831)

[**4.2.2.** **PyCharm Community Edition** 19](#_Toc193796832)

[**4.3.** **DESCRIPTION OF DATABASE** 20](#_Toc193796833)

[**4.3.1.** **SQLyog** 20](#_Toc193796834)

[**5.** **SYSTEM OVERVIEW** 21](#_Toc193796835)

[**5.1.** **BLOCK DIAGRAM** 22](#_Toc193796838)

[**5.2.** **TRAFFIC CONTROL SYSTEM** 23](#_Toc193796842)

[**5.3.** **FLASK BACKEND AND DATABASE** 24](#_Toc193796847)

[**5.4.** **USER MOBILE APPLICATION** 26](#_Toc193796853)

[**5.5.** **ADMIN WEB INTERFACE** 29](#_Toc193796860)

[**5.6.** **SYSTEM WORKFLOW** 31](#_Toc193796868)

[**6.** **RESULTS AND DISCUSION** 33](#_Toc193796869)

[**7.** **FUTURE SCOPE** 36](#_Toc193796870)

[**8.** **CONCLUSION** 37](#_Toc193796871)

[**REFERENCES** 38](#_Toc193796872)

[**APPENDIX** 41](#_Toc193796873)

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| Figure 1.1 | Problems related with conventional Ttraffic Systems (A) Pollution from vehicle exhaust, (B) Ambulance blocked in critical traffic junction | 3 |
| Figure 4.1 | Raspberry Pi 4 | 18 |
| Figure 4.2 | Pi Camera | 19 |
| Figure 4.3 | Pycharm Community Edition 2023.1 | 21 |
| Figure 5.1 | Block Diagram of the system | 23 |
| Figure 5.2 | Loss Curve Analysis for vehicle detection in YOLO Model training | 24 |
| Figure 5.3 | SQLyog Dataset, (A) Shows congestion and emergency vehicle details, (B) Shows real-time traffic details in each lane, (C) Stores various feedback received from the users, (D)Stores different traffic control details | 26 |
| Figure 5.4 | Mobile application User and Emergency vehicle interface | 28 |
| Figure 5.5 | Admin Website Dashboard | 30 |
| Figure 5.6 | (A) Website Interface showing various user details, (B) Real-time traffic congestion data with manual overriding and re-routing features | 31 |
| Figure 5.7 | Emergency vehicle details | 32 |
| Figure 6.1 | Vehicle count detection using system camera | 34 |
| Figure 6.2 | Vehicle detection and processing through Raspberry Pi & Pi Camera | 35 |
| Figure 6.3 | Traffic lights adjusted based on count of vehicles in each junction | 36 |
| Figure 6.4 | Shows the App interface of the emergency vehicle | 36 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| Table 4.1 | Basic properties of Raspberry Pi 4 | 18-19 |

**CHAPTER 1**

# **INTRODUCTION**

Traffic congestion is still a major problem in urban transport, causing unnecessary delays, more fuel consumption, and high emissions of air pollutants. Conventional traffic signal systems are based on pre-programmed time cycles and do not adjust to actual traffic conditions. This inefficiency causes prolonged waiting at signals, wasted vehicle idling, and drastic disruptions for emergency vehicles that must navigate through clogged roads. Furthermore, most of the contemporary automated traffic control systems utilize costly sensor-based technologies like infrared (IR) and ultrasonic sensors to sense vehicle density, which are challenging to install and maintain [9]. The necessity for an affordable, intelligent, and adaptive traffic management solution has never been more critical than it is today.

This research proposes an AI-driven traffic control system that utilizes available surveillance cameras to track vehicle population and adaptively modify traffic signals. In contrast to conventional sensor-based systems, this system processes real-time traffic images through deep learning algorithms, which are more efficient and scalable. The system has a Flask-based backend for easy communication between components and a MySQL database for storing and retrieving data.[1]-[3] In addition, a mobile app built with Flutter allows drivers to view real-time traffic information, get rerouting recommendations, and alert congestion conditions, whereas emergency vehicles can use a special interface to call for priority passage through signals.

Additionally, an admin web interface provides traffic authorities with the ability to track congestion levels, oversee emergency vehicle requests, and override traffic signals manually if required. The combination of the elements makes traffic management more responsive, eliminating congestion and enhancing emergency response time. This is an AI-based solution that offers a sustainable and flexible solution for contemporary traffic issues, providing a pragmatic mechanism for improving road efficiency, lessening pollution, and enhancing city mobility.

**CHAPTER 2**

# **MOTIVATION AND PROBLEM STATEMENT**



## **Motive of the Project**

Congestion in traffic is one of the most urgent urban issues, responsible for great delay, extra fuel usage, and enormous levels of pollution. Conventional traffic control systems are based on fixed signal timings that do not take into consideration real-time traffic changes, resulting in inefficient traffic flow and longer wait times at intersections. The randomness of traffic flows further intensifies congestion, particularly during peak hours, and therefore it is a need for intelligent and adaptive traffic control.

One of the biggest issues with traffic congestion is its effect on emergency vehicles like ambulances, fire engines, and police vehicles. In densely congested areas, response times for emergency services are severely increased, sometimes leading to lethal outcomes. Current solutions like manually operated traffic signals or priority lanes are inefficient or not scalable to dynamic urban traffic patterns[16]-[17]. There is a compelling requirement for an AI-based, real-time solution that can ensure an uninterrupted path for ambulances as well as optimize regular traffic movement.

Another problem with today's intelligent traffic systems is that they rely on costly and complicated hardware, e.g., infrared (IR) and ultrasonic sensors, to measure vehicles passing through intersections. They are hard to maintain and may not be affordable for widespread use in large cities.[21] In contrast, making use of already installed surveillance cameras and utilizing AI-based image processing methods can provide a cost-saving and scalable approach to dynamic traffic control.

The inspiration here is to build a smart traffic management system powered by AI which effectively controls the traffic signals in an efficient way by monitoring current congestion levels by applying computer vision methods. A Raspberry Pi along with a Pi Camera is being used to catch live traffic flow and then processes it through deep learning methods, like YOLOv3,[12] to determine the density of vehicles at an intersection. This data is then utilized to dynamically optimize signal timings, thereby making traffic movement smoother and lessening unnecessary delay.

In addition, the project endeavors to create an easy-to-use mobile app that offers real-time traffic information, rerouting suggestions, and a facility for users to report traffic-congestion issues. For emergency vehicles, a priority-based setup is implemented with traffic light control along their path to smoothly pass through areas where traffic is congested. An admin web interface also complements system functionality in terms of monitoring congestion levels, registering emergency vehicles, and overriding traffic lights if required.

Cars on the road with smoke coming out of the window

AI-generated content may be incorrect. A traffic jam in a busy city

AI-generated content may be incorrect.

Figure 1.1; Problems related with conventional traffic systems (A)Pollution from vehicle exhaust, (B)Ambulance blocked in critical traffic junction

Through the combination of AI-powered traffic control, real-time surveillance, and a database-based decision-making system, this project presents an affordable and scalable solution to contemporary urban traffic management.[1]-[8] Deployment of such an intelligent system has the potential to greatly alleviate congestion, enhance road safety, and optimize emergency response times, and ultimately result in more efficient and sustainable urban mobility.

## **Objectives of the Project**

1. Density based Adaptive Traffic Control: Implement an intelligent system that dynamically adjusts traffic signal durations based on real-time vehicle density, improving traffic flow efficiency.
2. Emergency Vehicle Priority System: Develop a mechanism for emergency vehicles to request priority passage, automatically adjusting signals to clear lanes and reduce delays.
3. User-Friendly Mobile Application: Provide a mobile app for users to access real-time traffic data, receive rerouting suggestions, and submit feedback regarding congestion and signal performance.
4. Comprehensive Admin Web Interface: Enable traffic authorities to monitor live congestion levels, approve emergency vehicle registrations, override signal controls when needed, and manage user feedback.
5. Database-Driven Traffic Management: Utilize a MySQL database to store essential traffic data, including congestion levels, user interactions, emergency requests, and system alerts, ensuring efficient data retrieval and decision-making.
6. Scalability and Cost-Effectiveness: Design a system that can be easily deployed in multiple urban areas using existing CCTV infrastructure, reducing costs and simplifying maintenance.
7. Enhanced Public Awareness and Participation: Encourage public involvement by allowing users to provide real-time feedback on traffic congestion and signal issues, contributing to continuous system improvements.
8. Energy Efficiency and Sustainability: Reduce unnecessary fuel consumption and emissions by optimizing traffic signal timings, leading to lower vehicle idling times and promoting environmental sustainability.

By addressing these objectives, this project provides a robust solution to modern traffic management challenges, enhancing road safety [1]-[13], minimizing congestion, and improving emergency response times while making efficient use of available technology.

**CHAPTER 3**

# **LITRATURE SURVEY**

**[1]. M.Pavan Kumar Reddy, V.Teja Vigneshwar, M.Shiva, B.Revanth, “AI driven Traffic Management System using Computer Vision and ML”, IJERST, Vol. 21 No. 1 (2025): Volume 21 No. 1 2025**

AI-driven traffic management systems utilizing computer vision and machine learning (ML) have revolutionized urban mobility by enabling real-time traffic monitoring, congestion prediction, and adaptive signal control. Computer vision models, such as YOLO and Faster R-CNN, accurately detect and classify vehicles, while ML algorithms analyze historical and real-time data to optimize traffic flow. Reinforcement learning techniques, such as Deep Q-Networks (DQNs), enhance adaptive signal timing by learning optimal traffic light configurations. IoT integration further enables real-time data collection from smart sensors and cameras, improving decision-making accuracy. Studies highlight significant reductions in congestion and travel time in smart city implementations, with AI-powered systems dynamically adjusting traffic signals based on vehicle density. However, challenges such as data privacy, computational demands, and infrastructure costs remain barriers to widespread adoption. Future advancements in edge computing, federated learning, and 5G connectivity will further enhance AI-driven traffic management, making urban transportation more efficient and sustainable

**[2]. Katsiaryna Bahamazava, “AI-driven scenarios for urban mobility: Quantifying the role of ODE models and scenario planning in reducing traffic congestion”, Department of Mathematical Sciences G.L. Lagrange, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino, 10129, Italy,2025**

AI-driven traffic management leverages machine learning, reinforcement learning, and computer vision to optimize urban mobility, integrating techniques like Ordinary Differential Equation (ODE) models and scenario planning to predict and mitigate congestion. ODE models provide a mathematical framework for modeling traffic dynamics, enabling real-time adjustments in signal control and vehicle flow. Scenario planning enhances decision-making by simulating various traffic conditions and optimizing responses through AI-driven predictive analytics. Studies show that reinforcement learning techniques, such as Deep Q-Networks, improve traffic signal efficiency, while computer vision models like YOLO facilitate real-time congestion monitoring. IoT and edge computing further enhance AI applications by enabling real-time data collection and processing. Implementations in cities like Pittsburgh and Hangzhou demonstrate significant reductions in congestion, yet challenges such as data privacy, infrastructure costs, and interoperability persist. Future advancements in Digital Twin technology and AI-enhanced scenario planning will drive the next generation of intelligent urban mobility solutions.

**[3]. Yahia Said, Yahya Alassaf, Refka Ghodhbani, Yazan Ahmad Alsariera, Taoufik, “AI-Based Helmet Violation Detection for TrafficManagement System”, Saidani, Olfa Ben Rhaiem, Mohamad Khaled Makhdoum, Manel HleiliCMES - Computer Modeling in Engineering and Sciences Volume 141, Issue 1, 20 August 2024, Pages 733-749**

AI-based helmet violation detection systems leverage computer vision and deep learning to enhance traffic management by identifying motorcyclists who fail to wear helmets. Advanced models like Convolutional Neural Networks (CNNs), YOLO, and Faster R-CNN enable real-time detection and classification of helmet usage from surveillance footage. Integrating AI with IoT and edge computing allows for efficient data processing and automated violation reporting to law enforcement agencies. Studies indicate that AI-driven detection systems improve compliance rates and road safety by providing automated enforcement mechanisms. Implementations in cities such as New Delhi and Jakarta have demonstrated significant reductions in helmet violations. However, challenges such as occlusions, varying lighting conditions, and privacy concerns need to be addressed. Future advancements in AI models, improved dataset diversity, and integration with intelligent traffic systems will further enhance helmet violation detection and overall traffic safety.

**[4]. Deng Pan a, Jiahao Lu a, Yao Li a b, Yuecheng Gao a c, “Unified embedded operating system for vehicle control and traffic management”, Journal of Industrial Information Integration Volume 44, March 2025, 100794**

A unified embedded operating system (UEOS) for vehicle control and traffic management integrates real-time data processing, AI-driven decision-making, and IoT connectivity to optimize urban mobility. Embedded systems in vehicles and traffic infrastructure facilitate seamless communication, enabling intelligent traffic control, autonomous vehicle coordination, and congestion mitigation. AI-powered algorithms, including reinforcement learning and predictive analytics, enhance decision-making for adaptive traffic signals and vehicle routing. Studies highlight the effectiveness of real-time operating systems (RTOS) in managing computational efficiency and ensuring system reliability. Implementations in smart cities demonstrate improvements in traffic flow, safety, and energy efficiency. However, challenges such as interoperability between different vehicle manufacturers, cybersecurity risks, and high deployment costs persist. Future advancements in edge computing, 5G connectivity, and blockchain security will further enhance UEOS, driving the evolution of intelligent transportation systems.

**[5]. Chiara Manfletti, Marta Guimarães, Claudia Soares, “AI for space traffic management”, Journal of Space Safety Engineering Volume 10, Issue 4, December 2023**

AI for space traffic management plays a crucial role in monitoring, predicting, and mitigating space debris risks while optimizing satellite and spacecraft operations. AI-driven tracking systems utilize sensor fusion from ground-based and satellite-based observations to enhance situational awareness. Reinforcement learning models assist in optimizing satellite in its peak form trajectories and resource allocation for efficient space traffic flow. Studies highlight the effectiveness of AI in reducing response time to potential threats and improving overall space safety. However, challenges such as data reliability, cybersecurity threats, and international policy coordination remain significant hurdles. Future advancements in AI-powered autonomous satellite operations, quantum computing for real-time decision-making, and collaborative AI frameworks will further strengthen space traffic management and sustainability in Earth's orbit. It is the method used for AI Management.

**[6]. Almatar, Khalid Mohammed, “Implementing AI-Driven Traffic Signal Systems for Enhanced Traffic Management”, International Journal of Sustainable Development & Planning, 2024, Vol 19, Issue 2, p781**

Traffic congestion in urban areas necessitates intelligent traffic signal control systems beyond traditional fixed-time and sensor-based approaches. AI-driven traffic management integrates machine learning, reinforcement learning, and computer vision to optimize traffic flow in real-time. Supervised learning models analyze historical data, while reinforcement learning systems, such as Deep Q-Networks, dynamically adjust signals based on real-time conditions. Computer vision techniques like YOLO and Faster R-CNN enable vehicle detection for congestion analysis. IoT and edge computing further enhance data collection and decision-making efficiency. Cities like Pittsburgh and Hangzhou have successfully reduced travel time and congestion using AI-based systems. However, challenges such as data privacy, infrastructure costs, and interoperability persist. Future advancements in 5G, Digital Twin technology, and cost-effective AI models will drive the evolution of intelligent traffic management, improving urban mobility and road safety.

**[7]. Baoming Wang, Haotian Zheng, Kun Qian, Xiaoan Zhan, Junliang Wang, “Edge computing and AI-driven intelligent traffic monitoring and optimization”, Electrical and Computer Engineering, University of Illinois Urbana Champaign, Urbana, IL, USA,2024**

Edge computing and AI-driven intelligent traffic monitoring and optimization enhance urban mobility by enabling real-time data processing, reducing latency, and improving traffic flow efficiency. AI-powered computer vision models, such as YOLO and Faster R-CNN, analyze traffic conditions from edge devices like smart cameras and IoT sensors, facilitating immediate congestion detection and adaptive signal control. Machine learning algorithms process historical and real-time data at the edge, optimizing traffic patterns and reducing dependency on centralized cloud systems. Reinforcement learning techniques, such as Deep Q-Networks (DQNs), enable adaptive traffic signal control based on dynamic traffic conditions. Studies demonstrate that edge computing reduces response time, enhances scalability, and improves data security by processing information closer to the source. However, challenges such as high infrastructure costs, interoperability between edge devices, and energy efficiency remain. Future advancements in 5G, federated learning, and energy-efficient edge AI models will further refine intelligent traffic management, ensuring more sustainable and responsive urban transportation systems.

**[8]. Seyed Reza Samaei, “A Comprehensive Algorithm for AI-Driven Transportation Improvements in Urban Areas”, Advanced Research in Science, Engineering and Technology, Belgium,2024**

A comprehensive algorithm for AI-driven transportation improvements in urban areas integrates machine learning, computer vision, and edge computing to enhance traffic management, reduce congestion, and optimize mobility. AI-powered models analyze real-time traffic data from sensors, cameras, and IoT devices to predict congestion patterns and dynamically adjust traffic signals. Reinforcement learning techniques, such as Deep Q-Networks (DQNs), optimize signal timing, while graph-based neural networks improve route planning and traffic flow efficiency. Computer vision models like YOLO and Faster R-CNN enable vehicle detection and classification, supporting automated enforcement and congestion monitoring. Edge computing reduces latency by processing traffic data closer to the source, enhancing real-time decision-making. Studies highlight the effectiveness of AI-driven algorithms in reducing travel time, improving fuel efficiency, and minimizing emissions in smart city implementations. However, challenges such as data privacy, interoperability, and infrastructure costs must be addressed. Future research should focus on federated learning, decentralized AI models, and 5G-enabled smart transportation networks to further improve urban traffic systems.

**[9]. Bibhu Dash1, Pawankumar Sharma, Meraj Ansari “A Data-Driven AI Framework to Improve Urban Mobility and Traffic Congestion in Smart Cities”, School of Computer and Information Sciences, University of the Cumberlands,2024**

A data-driven AI framework enhances urban mobility and reduces congestion in smart cities by leveraging machine learning, computer vision, and IoT-based real-time analytics. AI models analyze traffic data from cameras, sensors, and GPS to predict congestion and optimize signal timing. Reinforcement learning algorithms, such as Deep Q-Networks (DQNs), enable adaptive traffic control, while graph-based neural networks improve route optimization. Computer vision models like YOLO and Faster R-CNN facilitate vehicle detection and automated enforcement. Edge computing ensures low-latency data processing for real-time decision-making. Studies show AI-driven frameworks reduce travel time, fuel consumption, and emissions while improving road safety. Despite challenges like data privacy, infrastructure costs, and integration issues, advancements in federated learning, 5G, and decentralized AI will further refine smart city traffic management.

**[10]. Benson K. H. Hung, “Data-Driven Understanding Of AI-Based Traffic Signal Control And Its Implications On Low-Carbon Urban Transport Systems”, International Journal of Big Data Mining for Global WarmingVol. 04, No. 01, 2241001 (2022)**

A data-driven approach to AI-based traffic signal control enhances urban mobility while promoting low-carbon transport systems by optimizing signal timing, reducing congestion, and minimizing fuel consumption. AI models analyze traffic data from sensors, cameras, and GPS to predict flow patterns and adjust signals dynamically. Reinforcement learning techniques, such as Deep Q-Networks (DQNs), enable adaptive traffic management, while graph-based neural networks optimize route planning. Computer vision models like YOLO and Faster R-CNN support vehicle detection and automated enforcement. Edge computing ensures real-time data processing, lowering latency and improving decision-making. Studies highlight AI-driven traffic control’s role in reducing emissions, improving energy efficiency, and supporting sustainable urban transport. Despite challenges in data privacy, infrastructure costs, and system integration, advancements in federated learning, 5G, and decentralized AI will further strengthen low-carbon, intelligent traffic management systems.

**[11]. Heng Zeng , Manal Yunis, Ayman Khalil , Nawazish Mirza, “Towards a conceptual framework for AI- driven anomaly detection in smart city IoT networks for enhanced cybersecurity”, Journal of Innovation & Knowledge Volume 9, Issue 4, October–December 2024, 100601**

A conceptual framework for AI-driven anomaly detection in smart city IoT networks enhances cybersecurity by leveraging machine learning, real-time analytics, and edge computing. AI models analyze vast IoT data streams to detect deviations from normal network behavior, identifying potential cyber threats and vulnerabilities. Anomaly detection techniques, including supervised and unsupervised learning, improve threat detection accuracy, while deep learning models such as autoencoders and Generative Adversarial Networks (GANs) enhance intrusion detection. Edge computing reduces latency by processing security threats in real time, minimizing the risk of large-scale cyberattacks. Studies highlight AI-driven cybersecurity’s role in strengthening smart city resilience against cyber threats, ensuring data integrity, and protecting critical urban infrastructure. Despite challenges in data privacy, model interpretability, and integration complexity, advancements in federated learning, blockchain security, and zero-trust architectures will further enhance AI-powered anomaly detection in smart city IoT networks.

**[12]. Murat Bakirci, “Utilizing YOLOv8 for enhanced traffic monitoring in intelligent transportation systems (ITS) applications”, Digital Signal Processing Volume 152, September 2024**

Utilizing YOLOv8 for enhanced traffic monitoring in intelligent transportation systems (ITS) applications leverages advanced computer vision and deep learning techniques to improve real-time traffic analysis. YOLOv8, known for its high-speed and accurate object detection capabilities, enables efficient vehicle recognition, congestion monitoring, and anomaly detection in urban environments. Integrated with edge computing, it reduces latency by processing traffic data on-site, ensuring faster decision-making and improved responsiveness. Studies highlight YOLOv8’s effectiveness in optimizing traffic signal control, reducing congestion, and enhancing road safety by detecting violations such as illegal parking and helmet non-compliance. However, challenges such as computational demands, varying environmental conditions, and data privacy concerns persist. Future advancements in federated learning, model compression techniques, and 5G connectivity will further refine YOLOv8’s role in intelligent traffic monitoring, supporting sustainable and efficient urban mobility solutions.

**[13]. Fredy Nocua M, Wilson-Javier P´erez-Holguín, Camilo Pardo-Beainy, “Urban traffic monitoring based on deep learning on an embedded GPU”, Universidad Santo Tomas ´ – Grupo GIDINT Tunja Colombia,2025.**

Urban traffic monitoring based on deep learning on an embedded GPU enhances real-time traffic analysis by leveraging efficient neural networks and hardware acceleration. Deep learning models, such as YOLOv8 and Faster R-CNN, enable accurate vehicle detection, congestion assessment, and anomaly identification while running on embedded GPUs, which offer high computational efficiency with lower power consumption. This approach reduces latency, allowing for rapid decision-making in intelligent transportation systems (ITS). Studies highlight its effectiveness in adaptive traffic signal control, incident detection, and automated enforcement. However, challenges such as hardware limitations, thermal management, and integration complexity remain. Future research in model optimization, energy-efficient architectures, and edge AI will further improve the deployment of deep learning-based traffic monitoring on embedded GPUs, enhancing urban mobility and smart city applications.

**[14]. Allah Ditta, Muhammad Maroof Ahmed, Tehseen Mazhar, Tariq Shahzad, Yazan Alahmed, Habib Hamam, “Number plate recognition smart parking management system using IoT”, Measurement: Sensors Volume 37, February 2025, 101409**

Number plate recognition in smart parking management systems using IoT enhances automated vehicle access, security, and parking efficiency through AI-driven image processing and real-time data analytics. Deep learning models, such as YOLO and OCR-based techniques, enable accurate license plate detection and recognition, facilitating seamless entry, exit, and payment automation. IoT integration allows smart cameras and sensors to transmit real-time data to cloud-based platforms for monitoring and management. Studies highlight its effectiveness in reducing parking congestion, minimizing human intervention, and improving security. However, challenges such as varying lighting conditions, occlusions, and data privacy concerns persist. Future advancements in edge computing, federated learning, and blockchain-based secure data storage will further optimize smart parking management systems, enhancing urban mobility and smart city infrastructure.

**[15]. S. Abirami, M. Pethuraj, M. Uthayakumar, P. Chitra, “A systematic survey on big data and artificial intelligence algorithms for intelligent transportation system”, Case Studies on Transport Policy Volume 17, September 2024**

A systematic survey on big data and artificial intelligence algorithms for intelligent transportation systems (ITS) highlights the transformative role of AI and data-driven analytics in optimizing urban mobility. AI techniques, including machine learning, deep learning, and reinforcement learning, analyze vast traffic datasets to enhance congestion prediction, adaptive signal control, and route optimization. Big data frameworks process and store real-time traffic information collected from IoT sensors, GPS devices, and surveillance cameras, enabling accurate decision-making. Studies demonstrate that integrating AI with big data improves traffic efficiency, reduces travel time, and enhances road safety. However, challenges such as data privacy, high computational demands, and system interoperability remain. Future advancements in edge computing, federated learning, and quantum computing will further refine AI-driven ITS, ensuring scalable and sustainable transportation solutions for smart cities.

**[16]. Huszák, Árpád and Simon, Vilmos and Bokor, László and Tizedes, László and Pekár, Adrián, “An AI-Driven Intelligent Transportation System: Functional Architecture and Implementation”, An AI-Driven Intelligent Transportation System: Functional Architecture and Implementation. INFOCOMMUNICATIONS JOURNAL,2024**

An AI-driven intelligent transportation system (ITS) integrates machine learning, deep learning, and IoT technologies to enhance traffic management, reduce congestion, and improve mobility. The functional architecture of AI-driven ITS consists of data collection from IoT sensors, real-time processing through edge computing, and decision-making powered by AI algorithms such as reinforcement learning for adaptive traffic control and predictive analytics for congestion management. Studies highlight its effectiveness in optimizing traffic flow, reducing travel time, and enhancing road safety. Implementations in smart cities demonstrate improved transportation efficiency and sustainability. However, challenges such as data privacy, high computational costs, and system interoperability persist. Future advancements in federated learning, blockchain security, and 5G-enabled smart infrastructure will further refine AI-driven ITS, making urban mobility more intelligent and adaptive.

**[17]. Srinjoy Mitra , “AI-driven predictive models for traffic flow in IoT-driven smart cities”, Vol. 1 No. 2 (2024): Uncertainty Discourse and Applications**

AI-driven predictive models for traffic flow in IoT-driven smart cities leverage machine learning, deep learning, and big data analytics to optimize urban mobility. These models analyze real-time traffic data from IoT sensors, cameras, and GPS devices to predict congestion patterns and enhance traffic signal control. Techniques such as reinforcement learning and graph-based neural networks improve route optimization, reducing travel time and fuel consumption. Studies highlight their effectiveness in mitigating congestion, improving road safety, and supporting sustainable transportation. However, challenges such as data privacy, computational complexity, and system scalability persist. Future advancements in federated learning, edge computing, and 5G-enabled smart infrastructure will further refine AI-driven traffic prediction, enabling more efficient and adaptive urban transportation systems.

**[18]. Jun Tang, Caixian Ye, Xianlai Zhou, Lijun Xu, “YOLO-Fusion and Internet of Things: Advancing object detection in smart transportation”, Alexandria Engineering Journal Volume 107, November 2024**

YOLO-Fusion and the Internet of Things (IoT) enhance object detection in smart transportation by integrating advanced deep learning models with real-time sensor data. YOLO-Fusion combines multiple data sources, such as traffic cameras, LiDAR, and IoT-enabled sensors, to improve detection accuracy and robustness under varying environmental conditions. This fusion-based approach enhances vehicle recognition, pedestrian detection, and traffic monitoring, leading to more efficient traffic management. Studies highlight its effectiveness in optimizing traffic flow, reducing accidents, and improving autonomous vehicle navigation. However, challenges such as high computational demands, data synchronization, and privacy concerns persist. Future advancements in edge computing, federated learning, and 5G connectivity will further refine YOLO-Fusion's role in smart transportation, making urban mobility safer and more efficient.

**[19]. Shih-Yu Sun, Ting-Hao Hsu, Chia-Yen Huang, Cheng-Han Hsieh, Chun-Wei Tsai, “A Data Augmentation System for Traffic Violation Video Generation Based on Diffusion Model”, Procedia Computer Science Volume 251, 2024**

A data augmentation system for traffic violation video generation based on diffusion models enhances the training of AI-driven traffic monitoring systems by generating realistic synthetic data. Diffusion models create high-quality video sequences that simulate various traffic violations, such as signal jumping, overspeeding, and illegal lane changes, improving the robustness of deep learning models. These augmented datasets help overcome data scarcity issues and enhance model generalization for real-world scenarios. Studies highlight the effectiveness of diffusion-based augmentation in improving detection accuracy, reducing bias, and supporting automated law enforcement. However, challenges such as computational complexity, ethical concerns, and domain adaptation persist. Future advancements in generative AI, edge computing, and adversarial training will further refine diffusion-based traffic violation video generation, strengthening intelligent transportation systems.

**[20]. Quang Nguyen Huy Minh, Nen Nguyen Dinh, Long Viet Ho, Cuong Phan Huu, “Real-time traffic accident detection using yolov8”, Transportation Research Procedia Volume 85, 2025**

Real-time traffic accident detection using YOLOv8 leverages advanced deep learning techniques to enhance road safety and emergency response. YOLOv8's high-speed object detection enables rapid identification of accidents by analyzing live traffic camera feeds and IoT sensor data. Its real-time processing capability allows for quick alerts to traffic management centers and emergency responders, minimizing response times and reducing secondary collisions. Studies demonstrate YOLOv8's effectiveness in improving detection accuracy, reducing false alarms, and enhancing situational awareness. However, challenges such as occlusions, varying lighting conditions, and computational efficiency persist. Future research in edge computing, federated learning, and multi-modal data fusion will further refine real-time accident detection, making urban roads safer and more adaptive to emergencies.

**[21]. Laura Deveer, Laura Minet, “Real-time air quality prediction using traffic videos and machine learning”, Transportation Research Part D: Transport and Environment Volume 142, May 2025**

Real-time air quality prediction using traffic videos and machine learning enhances environmental monitoring by leveraging computer vision and predictive analytics. Machine learning models analyze traffic patterns, vehicle density, and meteorological factors from video feeds to estimate pollutant levels such as CO2, NOx, and PM2.5. Deep learning architecture, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), improve predictive accuracy by capturing spatiotemporal relationships in traffic data. Studies demonstrate the effectiveness of integrating video-based air quality prediction with IoT sensor networks for more precise and localized pollution monitoring. However, challenges such as data variability, sensor calibration, and real-time processing constraints persist. Future advancements in edge AI, federated learning, and hybrid AI models will further enhance the accuracy and scalability of real-time air quality prediction in smart cities.

**CHAPTER 4**

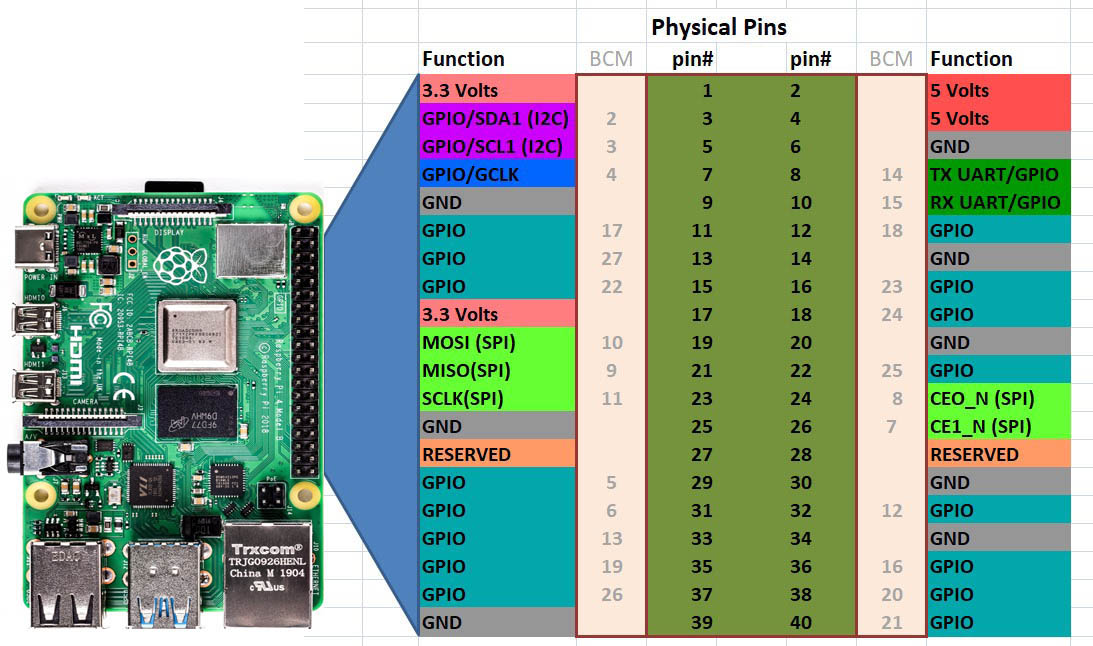
# **IMPLEMENTATION DETAILS**



## **DESCRIPTION OF SYSTEM COMPONENTS**

### **Raspberry Pi & Pi Camera**

The Raspberry Pi 4 is a small, low-cost, and versatile single-board computer that can be used for a wide array of projects, from home automation to media centers, and is powered by a quad-core processor and comes in various RAM options. A quad-core 64-bit Broadcom BCM2711 processor, offering significantly faster performance than previous models.

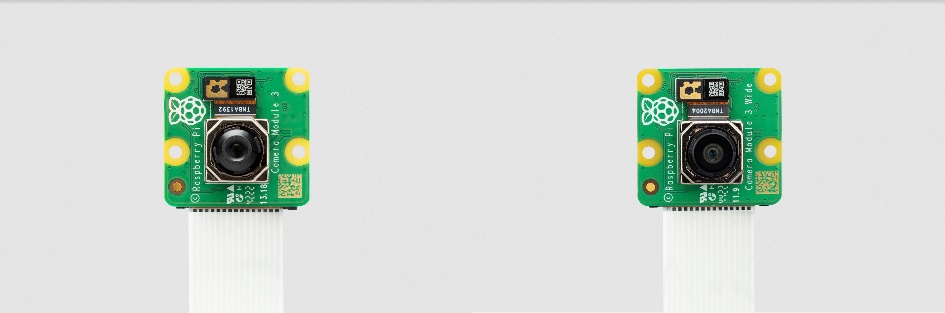


*Figure 2: Raspberry Pi 4*

*Table 4.1: Basic properties of Raspberry Pi 4*

|  |  |
| --- | --- |
| **Features** | **Specification** |
| Processor | Broadcom BCM2711, Quad-core Cortex-A72 (ARM v8) 64-bit SoC 1.5GHz |
| RAM Options | 2GB, 4GB, or 8GB LPDDR4-3200 SDRAM |
| GPU | VideoCore VI 500 MHz |
| Storage | MicroSD card slot (for OS and file storage) |
| USB Ports | 2 × USB 3.0, 2 × USB 2.0 |
| Ethernet | Gigabit Ethernet |
| Wireless | Dual-band 2.4/5.0 GHz Wi-Fi, Bluetooth 5.0, BLE |
| Video Output | 2 × Micro HDMI ports (supports up to 4K resolution) |
| Audio Output | 3.5mm audio jack and HDMI audio |
| GPIO Pins | 40-pin GPIO header |
| Power Supply | USB-C (5V/3A) |
| Operating System | Raspberry Pi OS (formerly Raspbian), Ubuntu, etc. |

The Raspberry Pi Camera Module is a small, compact camera designed to attach to Raspberry Pi computers, allowing users to capture still images and video, with various models offering different resolutions and features[2]-[7]. The camera module is designed to connect directly to the Raspberry Pi's Camera Serial Interface (CSI) connector. It can capture both still images and videos, with support for resolutions up to 1080p at 30 frames per second.



*Figure 4.2: Pi Camera*

## **DESCRIPTION OF SOFTWARE USED**

### **Visual Studio Code**

Visual Studio Code, or VS Code, is a light but robust source-code editor from Microsoft. It is extensively used for various programming languages such as Python, JavaScript, Dart, and others. VS Code is famous for its responsiveness, flexibility, and maximum support for extensions, which makes it a favorite among developers.

VS Code offers intelligent code completion, syntax coloring, debugging, Git support, and an inbuilt terminal to boost the coding experience. Its marketplace enables developers to install plugins for various frameworks such as Flutter, Flask, and MySQL, which were vital during our project development.

For the SmartFlow AI-Based Traffic Management System, VS Code was utilized to implement the Flutter-based mobile app. It assisted in crafting the driver-friendly and emergency vehicle-friendly interface, incorporating real-time traffic information, and supporting communication with the Flask backend[12]-[18]. The Flutter and Dart extensions within VS Code ensured easy development and debugging of the mobile app with efficiency and scalability.

### **PyCharm Community Edition**

PyCharm is a robust Integrated Development Environment (IDE) aimed at Python programming. Created by JetBrains, PyCharm is a strong and effective platform to write, debug, and develop Python code. PyCharm is used by developers to create applications across diverse fields, ranging from web development to data science and artificial intelligence.

It includes smart code completion, syntax highlighting, and a feature-rich debugger that streamlines the development process. PyCharm is capable of supporting several frameworks like Flask, Django, and FastAPI, and thus, it is the perfect option for backend development [8]-[21]. It also provides integrated version control support (Git, SVN), database support, and an intuitive interface.

A screenshot of a computer program

AI-generated content may be incorrect.

*Figure 4.3: Pycharm Community Edition 2023.1*

## **DESCRIPTION OF DATABASE**

### **SQLyog**

SQLyog is a robust and interactive MySQL management tool employed for database administration and optimization. SQLyog presents an easy-to-use graphical user interface (GUI) through which developers can easily manage MySQL databases without much command-line execution. SQLyog presents capabilities like database design, running queries, data synchronization, scheduled backups, and real-time monitoring, making it a valuable tool for database management.

SQLyog was utilized to handle the MySQL database in the SmartFlow Density based Traffic Management System. It stored key information like user login credentials, emergency vehicle registration, traffic congestion levels in real time, system notifications, and user ratings[12]. The software provided seamless interaction with the database with easy insertion, retrieval, and management of data. Its performance tuning and query builder features enabled efficient optimization of database operations to make the system react effectively to traffic situations and user queries.

**CHAPTER 5**

# **SYSTEM OVERVIEW**

The Smartflow Density based Traffic Management System is designed to optimize traffic flow using artificial intelligence, real-time surveillance, and automated decision-making. The system integrates various components, including image-sensing cameras, a Raspberry Pi processing unit, a MySQL database, a website for admin monitoring, and a mobile application for users and emergency vehicles. The architecture ensures seamless communication between these components to provide real-time traffic control and improve congestion management.

The system functions as follows:

• Image Processing & AI-Based Analysis: Cameras installed at junctions capture live traffic images, which are processed using AI algorithms running on the Raspberry Pi. The AI model determines vehicle density at each lane and dynamically adjusts the traffic signal duration accordingly.

• Data Storage & Management: The processed traffic data, user credentials, emergency vehicle registrations, and congestion alerts are stored in a MySQL database.

• Web-Based Admin Interface: The admin panel allows authorities to monitor real-time traffic data, view congestion alerts, and approve emergency vehicle registrations.

• User & Emergency Mobile Applications: The mobile app provides users with live congestion details, rerouting suggestions, and feedback options. Emergency vehicle users can manipulate signals on their selected lanes for quick passage.

• Flask-Based Backend Communication: A Flask server acts as the middleware, handling requests from the mobile application[1]-[21], web dashboard, and Raspberry Pi AI module. It manages data retrieval, user authentication, and system alerts.

This modular architecture ensures efficient traffic management by leveraging AI-driven automation, real-time monitoring, and interactive user interfaces.



## **BLOCK DIAGRAM**

A diagram of a computer hardware system

AI-generated content may be incorrect.

*Figure 5.1: Block Diagram of the system*

The block diagram presents the workflow of the AI-powered Traffic Management System, combining image processing, real-time traffic management, and user interaction. Live images of traffic congestion at intersections are captured through the Raspberry Pi Camera. Images are processed by YOLO, an AI-powered object detection model, to estimate vehicle density. Depending on the level of congestion, the system dynamically modulates the duration of the green signal for the best traffic flow. The processed data is routed to the Flask backend, which coordinates intercomponent communication and stores the data in a MySQL database. The mobile application based on Flutter enables users to see current traffic information, receive suggested rerouting, and provide feedback. Emergency vehicles can request lane clearance, which produces an automatic green signal[1]-[21]. Traffic officials use a Flask-driven web dashboard to monitor traffic congestion, approve emergency vehicle registration, and override signals manually if necessary. This system improves traffic flow, alleviates congestion, and facilitates quicker movement of emergency vehicles.



## **TRAFFIC CONTROL SYSTEM**

The traffic signal control system uses AI, employing Raspberry Pi as the processing unit and a Pi Camera to obtain real-time images of traffic volume at road intersections. The system dynamically controls traffic signals depending on the number of vehicles in every lane, hence enhancing traffic flow efficiency and reducing congestion.

The system uses the image processing aspect of the system YOLOv3 (You Only Look Once, version 3), a deep learning object detection algorithm, to correctly determine the number of vehicles in a specified lane. Real-time processing is done on captured images, with the YOLOv3 model identifying and classifying the vehicles and then deciding the density in each lane. The results from the processing are utilized in dynamically assigning the duration of the green light in relation to levels of congestion.

Every intersection in the system has four lanes, with each lane having an LED-based traffic governing system [1]-[17]. There are three LEDs (red, yellow, and green) in each lane to display various states of traffic signals.

A screenshot of a computer screen

AI-generated content may be incorrect.

*Figure 5.2: Loss Curve Analysis for vehicle detection in YOLO Model training*

The system works step-by-step:

* Traffic density analysis: The Pi Camera captures an image of the first lane and processes it using YOLOv3 to determine vehicle count.
* Dynamic signal adjustment: Based on the vehicle density, the green signal duration is assigned proportionally. If a lane is highly congested, the green light remains on longer to allow more vehicles to pass. If the lane has fewer vehicles, the green duration is reduced to prioritize other lanes.
* Lane switching: Once the assigned time for the green signal elapses, the system moves to the next lane by capturing a new image and repeating the process.
* LED-based traffic control: The green LED is activated for the current lane while the red LEDs remain active for the other lanes. A brief yellow light transition is used before switching signals to alert drivers.

This solution provides a complete adaptive traffic control system that gets rid of the constraints of the conventional fixed-timer systems. Through dynamic regulation of signal duration, the system minimizes waiting times, relieves congestion, and enables the passage of emergency vehicles more efficiently.



## **FLASK BACKEND AND DATABASE**

The Flask backend and database management are essential in the AI-based traffic management system as they enable smooth communication among the AI processing module, the mobile application, and the admin web interface. Flask, being an efficient and lightweight Python-based web framework, serves to handle system component requests, process data, and send corresponding responses. It facilitates seamless interaction among the Raspberry Pi-based traffic control system, the database, and the applications that face users. The system uses MySQL as its main database to save important information, such as user login credentials, emergency vehicle registration data, real-time traffic congestion levels, emergency signals, lane clearance requests, and user feedback. The database gets updated automatically in real time through inputs received from the Raspberry Pi module and AI model so that traffic decision-making and control remain accurate. The mobile application and admin console receive live traffic information through Flask APIs, maximizing system efficiency as well as dependability.

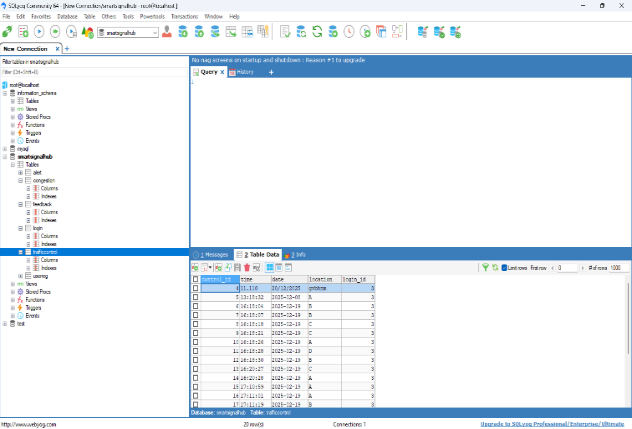
The Flask backend is instrumental in controlling the exchange of data between the AI-driven traffic management system, the admin site, and the user mobile app. With its lightweight and robust Python web framework, Flask facilitates smooth interaction among various components, processing HTTP requests, API endpoints, and real-time data exchange. The Flask server is also in charge of obtaining congestion information from the Raspberry Pi, handling user authentication requests, handling emergency vehicle clearance, and updating the database with real-time traffic data. Through the function of serving as a link between the AI processing module and user interfaces, Flask facilitates the flow of data across the whole system.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.3: SQLyog Dataset, (A) Shows congestion and emergency vehicle details, (B) Shows real-time traffic details in each lane, (C) Stores various feedback received from the users, (D)Stores different traffic control details*

For database management, the system employs MySQL, a relational database that efficiently stores and retrieves data[12]. The database schema is structured to handle multiple functionalities, ensuring real-time updates and maintaining a historical log of traffic data. The database is responsible for storing:

* User Authentication and Registration: The system securely stores user login credentials, email, phone number, and passwords, ensuring proper authentication.
* Emergency Vehicle Registration: Emergency vehicles have a separate registration system, where details like vehicle registration number, driver credentials, and assigned unique emergency IDs are stored. This allows the system to recognize emergency vehicles and prioritize lane clearance.
* Real-time Traffic Congestion Data: The Raspberry Pi module continuously updates the database with vehicle density levels at each junction. This data is then accessed by the admin dashboard and user app to display congestion updates and suggest alternate routes.
* Emergency Signals and Lane Clearance Requests: When an emergency vehicle requests lane clearance, the system updates the database to indicate which lane should receive priority green signal activation. The request is immediately processed by the Flask backend, which communicates with the AI module to execute lane clearance.
* User Feedback and System Alerts: Users can provide feedback regarding traffic conditions, system performance, or other concerns, which is stored in the database. Additionally, system-generated alerts (e.g., high congestion, malfunction reports, or emergency requests) are recorded and displayed in the admin interface.

Through the combination of Flask and MySQL, the system ensures efficient data handling, fast response times, and secure information storage, making the AI-powered traffic management system highly scalable and reliable.



## **USER MOBILE APPLICATION**

The Smartflow Traffic Management System based on artificial intelligence includes a mobile application that is meant to improve user experience through real-time traffic information, congestion rerouting features, and emergency vehicle clearance features. The mobile application was created with Flutter using Visual Studio Code to make it compatible for Android. The reason Flutter was used is that it can offer high-performance, single-codebase, smooth user interface, and dynamic functionality. The application is linked to a Flask backend, which acts as the interface between the mobile app, MySQL database, and AI-based traffic management system. With this smooth integration, there can be effective communication and real-time updates, such that users have access to the most appropriate and current traffic data.

A screenshot of a smart signal hub

AI-generated content may be incorrect.A screenshot of a phone

AI-generated content may be incorrect.A screen shot of a phone

AI-generated content may be incorrect.A screen shot of a phone

AI-generated content may be incorrect.

*Figure 5.4: (A) Mobile application User and Emergency vehicle interface, (B) Login page for user login, (C) User app interface providing live traffic details, rerouting suggestions and congestion details, (D) Emergency vehicle interface allowing vehicles to manually notify the traffic signal to reduce traffic congestion.*

The mobile app has an authenticated system with secure registration allowing regular users as well as emergency vehicles to register separately. The registration is mandatory for providing vital information, including name, email, phone number, and the vehicle's registration number. For emergency vehicles, there is a supplementary verification process, making sure only authorized individuals, such as ambulance drivers, fire department personnel, and police officers, can avail special privileges[18]-[21]. All the registered information is stored securely in the MySQL database, and the credentials of users are checked by an admin through the website dashboard prior to login access. A password recovery function is also implemented, where users can reset their credentials through an automated email-based system if they happen to forget their login credentials.

The app also offers two different user interfaces, one for normal users and one for emergency vehicles, each being custom-built to suit their particular requirements. Normal users receive real-time information on traffic congestion at different junctions, which helps them make the right decision to travel. When a junction is congested, the app offers rerouting advice so that users do not have to wait for long periods of time and can take the shortest route. Also, users can provide feedback about traffic problems, like faulty signals, road conditions, or hotspots of congestion, to the admin directly. This feature makes the system constantly improve by enabling authorities to take appropriate action on the basis of user feedback.

For emergency vehicles, the app has all the features that are available to normal users but with extra functionalities that give priority to their movement in congested traffic. Emergency vehicle drivers can choose the exact lane they are going to cross, and the system will turn the traffic signal green for that lane automatically. This provides fast clearance and unobstructed passage for ambulances, fire engines, and police cars, minimizing delays that may cause life-threatening consequences. The Raspberry Pi and Pi camera-powered AI traffic control system, which is running all the time to monitor traffic density and adjust signals in real time, has normal traffic signal processes overridden by emergency vehicle clearance to clear the path for them.

The Flask-based backend of the app is used, which is a lightweight, efficient web framework built using Python to manage the interaction between the AI traffic management system, mobile app, and website. Flask handles user authentication, data fetching, and emergency signal processing. When a user logs in, their credentials are checked against the MySQL database to provide a secure login experience[8]-[17]. The Flask backend also retrieves real-time traffic updates from the Raspberry Pi-based AI module and shows congestion levels within the app in real time. Furthermore, when an emergency vehicle asks for a signal override, the Flask backend handles the request and adjusts the AI-managed traffic lights accordingly.

The MySQL database holds important system informations, such as user accounts, emergency vehicle registrations, current traffic conditions, and feedback history. It holds records of send and receive alert signals, emergency signal overrides, and session records of every registered user. The database guarantees the efficient and secure processing of every request from the mobile app, website, and AI system without unauthorized access, ensuring a constant data flow among system components.

Overall, the mobile application serves as a crucial interface that bridges the gap between road users, emergency responders, and traffic management authorities. By integrating AI-driven traffic signal control, Flask-based backend communication, and real-time data processing, the system optimizes urban traffic flow, reduces congestion, and enhances emergency response times. The application of Flutter, Flask, and MySQL makes the app scalable, efficient, and strong, so it can meet the increasing needs of contemporary smart cities while giving users a smooth and responsive experience.



## **ADMIN WEB INTERFACE**

Admin Web Interface is a key part of the AI-powered traffic management system, offering real-time traffic observation and control of traffic signals to traffic authorities. The web dashboard was coded with Flask as the backend and HTML as the frontend, where the development environment is PyCharm. It is the focal point of control, combining real-time traffic observations, emergency vehicle registration, and system notifications to maintain fluid traffic flow.

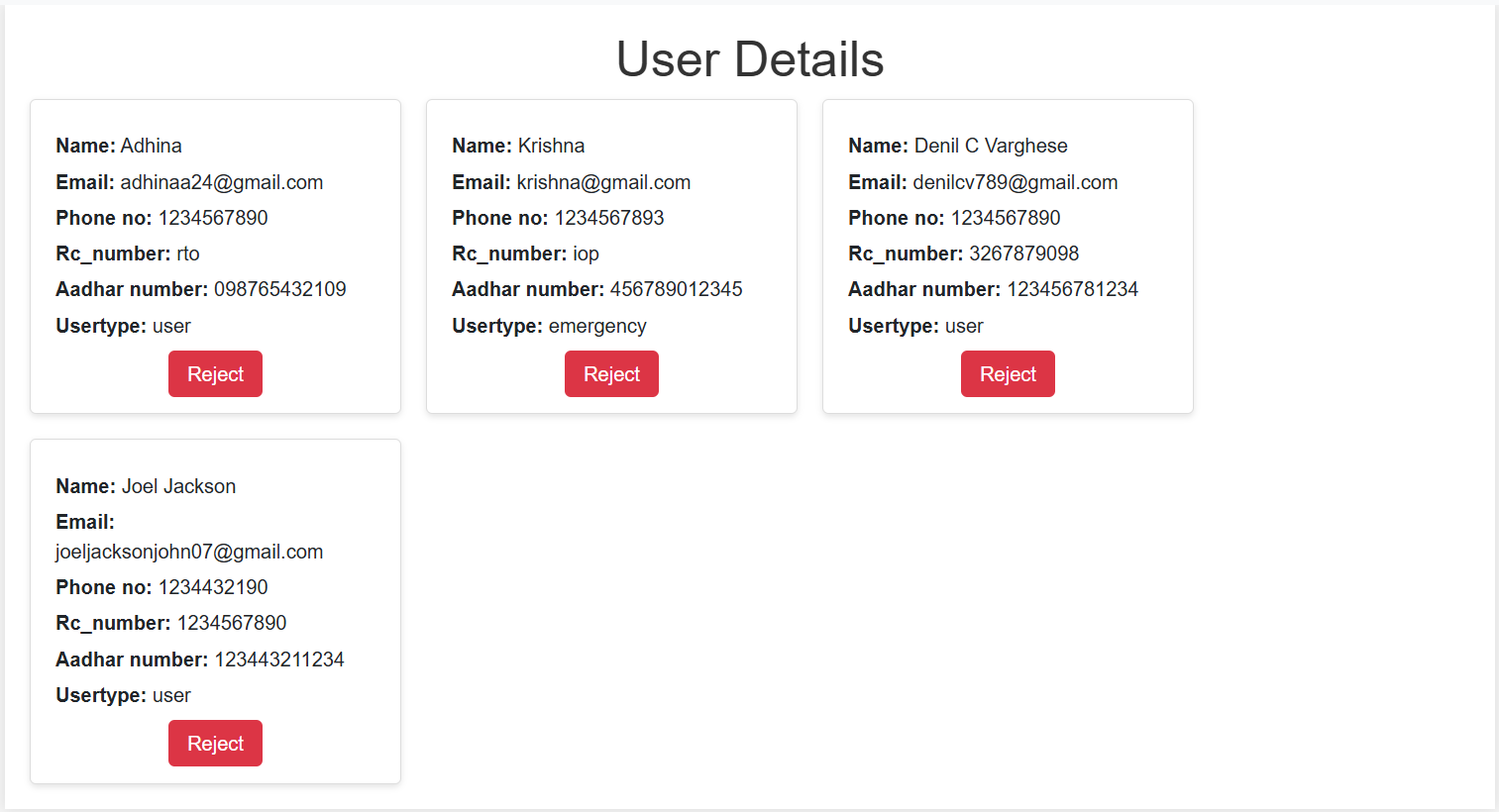
A screenshot of a computer

AI-generated content may be incorrect.

*Figure 5.5: Admin Website Dashboard*

The admin dashboard is created to offer a transparent and easy-to-use interface to monitor real-time traffic conditions. It gets constant updates from the Raspberry Pi cameras installed at traffic intersections. The AI model processes the images captured and calculates the levels of congestion, which are reflected on the dashboard. Authorities can then analyze traffic density at various junctions and take action if sudden congestion is detected.

One major feature of the interface is verification and approval of emergency vehicle registration. Ambulances and fire trucks register via the mobile application by entering information such as vehicle registration number, email address, phone number, and emergency vehicle ID[8]-[21]. These details are cross-checked by the admin for authentication prior to granting permission. Upon approval, the emergency vehicle is awarded special permissions, including overriding signals and clearing lanes when necessary. The system also manages user reports and warnings, enabling users to report complaints based on traffic jams or malfunctioning signals through the mobile app. The admin dashboard shows such reports, facilitating officials to address the complaints in a timely manner. Emergency lane clearance request alerts are also given priority in order to quickly respond.

A screenshot of a web page

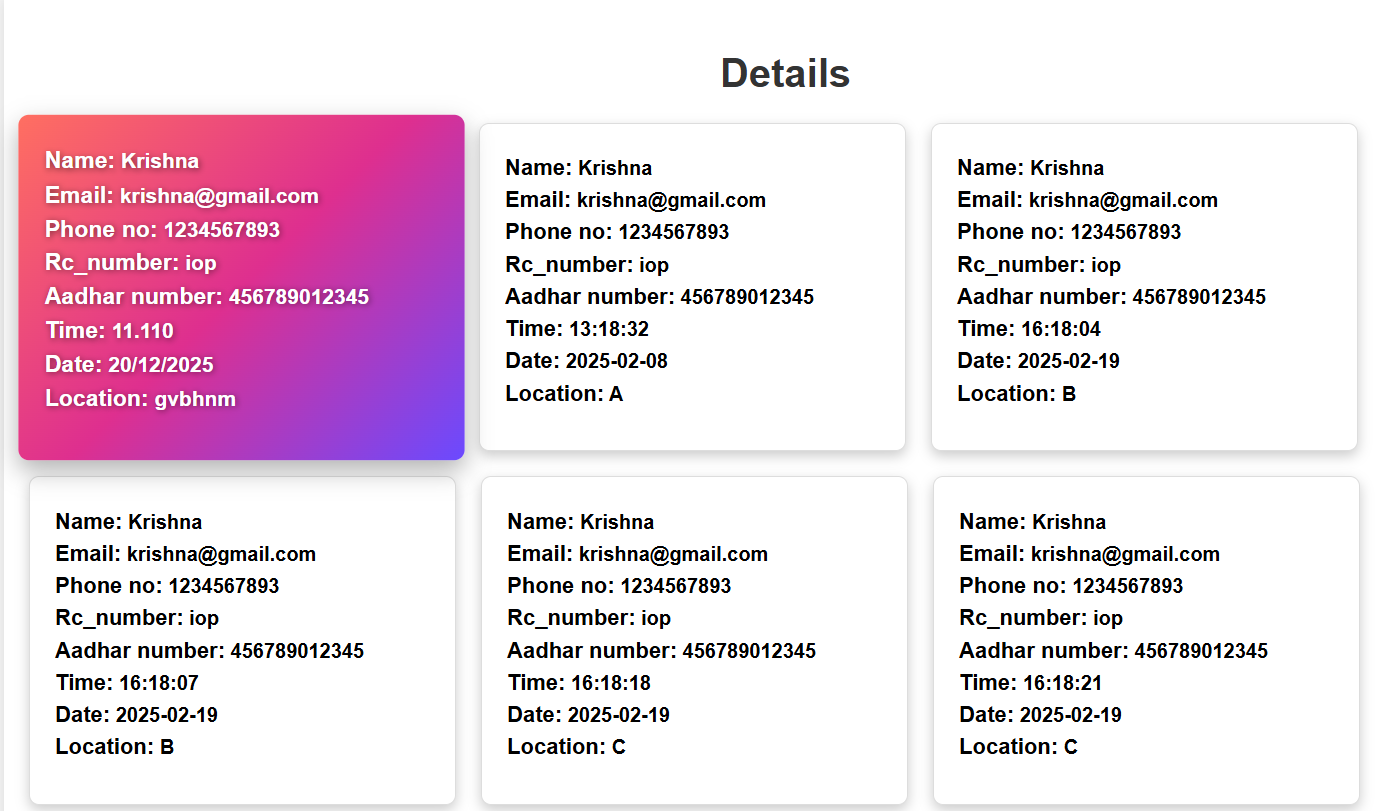
AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

*Figure 5.6: (A) Website Interface showing various user details, (B) Real-time traffic congestion data with manual overriding and re-routing features, (C) Manual route suggestion feature of the Admin*

Manual signal override is perhaps the most essential feature. Even though traffic lights are dynamically controlled according to AI-identified congestion, the admin still has override capabilities in case of unexpected traffic accidents, emergencies, or incidents. This makes sure that even in situations where AI-driven automation is not sufficient, human override can improve traffic flow.



*Figure 5.7: Emergency vehicle details, (location, time and junction it crossed is displayed for future reference*

Through the incorporation of these elements, Admin Web Interface makes an all-inclusive platform for observing and operating urban traffic in an efficient manner, easing congestion, lessening emergency response times, and boosting overall traffic management.



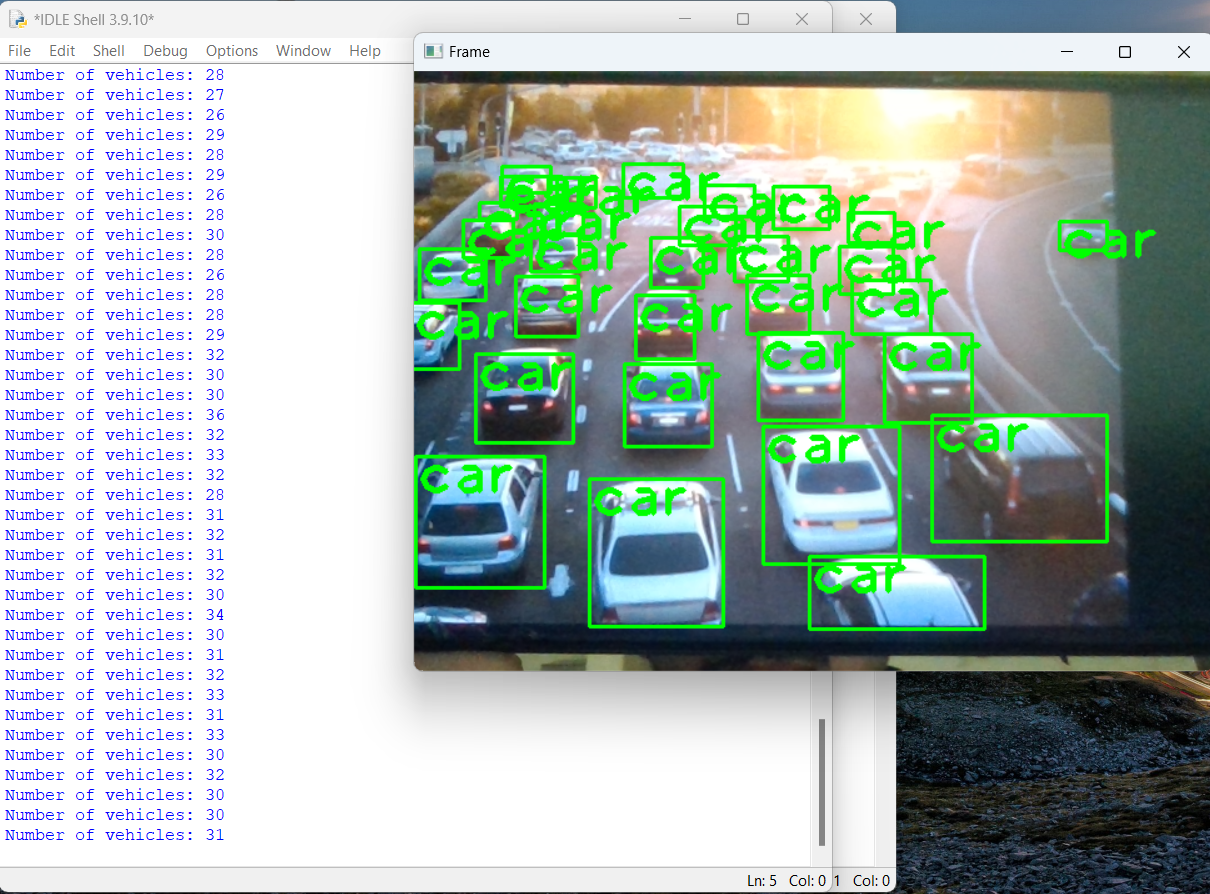
## **SYSTEM WORKFLOW**

1. The Raspberry Pi Camera captures real-time images of traffic at each junction.
2. The captured image is processed using YOLO, an AI-based object detection model, to determine vehicle density in each lane.
3. The AI module analyzes the number of vehicles and determines the appropriate green signal duration for the lane based on congestion levels.
4. The processed traffic data is sent to the Flask backend, which acts as the communication bridge between AI processing, the database, the web interface, and the mobile application.
5. The Flask backend stores real-time traffic congestion data, emergency signals, and user feedback in the MySQL database.
6. The database sends live traffic updates to both the admin web interface and the user mobile application.
7. The web-based admin dashboard allows traffic authorities to monitor congestion levels, review emergency vehicle registrations, and manage system alerts.
8. The mobile application provides users with real-time traffic conditions, alternative rerouting suggestions, and feedback submission options.
9. Emergency vehicles can log in separately through the app to request lane clearance, which is sent to the admin for approval.
10. Upon approval, the system prioritizes the requested lane by turning the traffic signal green, allowing emergency vehicles to pass smoothly.
11. The AI system dynamically updates the traffic signal sequence based on the latest congestion data and emergency requests.
12. The updated signal timing is communicated to the traffic light controller, which adjusts the LED signals accordingly.
13. The cycle continues with new image captures and AI-based adjustments to maintain optimal traffic flow.

**CHAPTER 6**

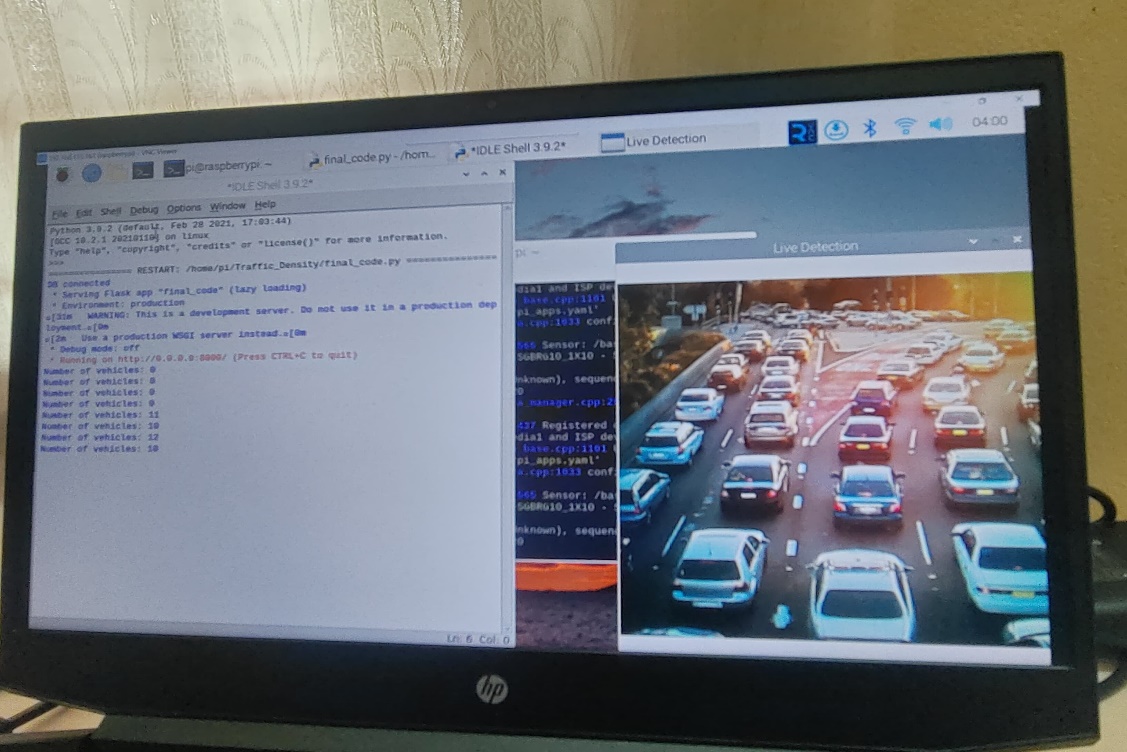
# **RESULTS AND DISCUSION**

The outcomes of this project prove the success of a density-based traffic control system through image processing methods. Through the use of Raspberry Pi with a Pi Camera, the system effectively captures real-time traffic scenarios, processes vehicle density through YOLOv3, and adapts traffic signal lengths dynamically. The information is saved in a MySQL database and accessed via a Flask-based backend with real-time monitoring through a web-based admin interface and mobile app. The system maintains smooth traffic flow by prioritizing the heavily congested lanes and granting emergency vehicles lane-clearing capabilities. The use of this intelligent traffic control system highlights an effective, cost-efficient solution that makes use of existing surveillance infrastructure with fewer sensor-based hardware requirements. The output achieved using the system proves its capability to reduce congestion, optimize emergency response times, and enhance general urban traffic management. Subsequent sections give a detailed explanation of the outcomes observed.



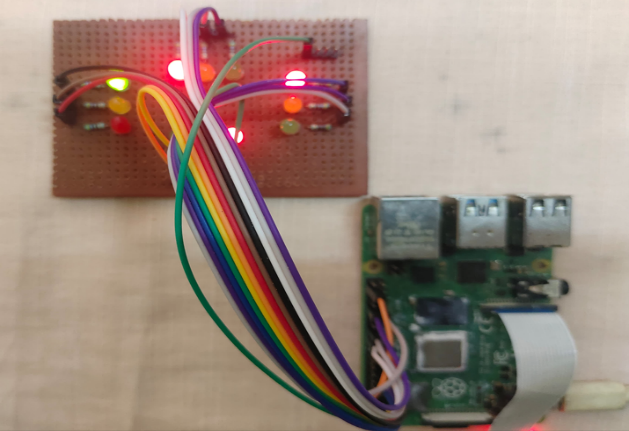
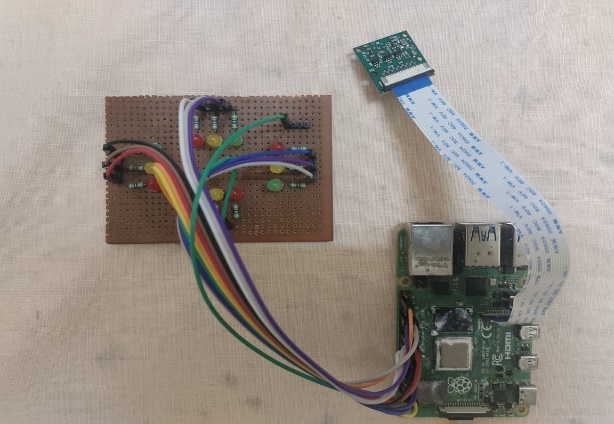
*Figure 6.1: Vehicle count detection using system camera*

The picture illustrates the computer webcam-based vehicle detection and counting system. The system is able to accurately detect and tag vehicles in real time through image processing methods, with each detected car highlighted inside a green rectangle. Displayed on the left is the number of detected vehicles, changing dynamically as new frames are processed. The applied model obtains about 90% accuracy for vehicle detection and a response time of around 0.5 seconds. This guarantees efficient and accurate estimation of traffic density, which is essential for real-time dynamic adjustment of signal timings and traffic flow management.



*Figure 6.2:: Vehicle detection and processing through Raspberry Pi & Pi Camera*

The photo illustrates the vehicle detection with a Raspberry Pi 4 and a Pi Camera. The system takes real-time traffic images, processes them for vehicle detection, and dynamically updates the number. The number of vehicles detected is printed to the terminal, and a live detection window offers a graphical view of the processed traffic image. Because the Pi Camera is utilized, the detection accuracy is between 65-80%, which is a bit lower compared to high-performance cameras because of hardware constraints. Nevertheless, the system still maintains a response time of less than 1 second, which makes it effective for real-time traffic density estimation and signal control.



*Figure 6.3:: Traffic lights adjusted based on count of vehicles in each junction*

The traffic light system is dynamically controlled based on real-time vehicle detection at each junction. The Raspberry Pi processes vehicle count data and adjusts the signal timing accordingly to optimize traffic flow.

A screenshot of a phone

AI-generated content may be incorrect.A circuit board with wires and lights

AI-generated content may be incorrect.

*Figure 6.4: (A) Shows the App interface of the emergency vehicle, (B) Traffci light adjusted based on request sent by emergency vehicle via Smart Signal App*

The emergency vehicle app interface allows real-time traffic signal adjustments based on priority requests from emergency vehicles. When an emergency vehicle approaches a junction, it sends a request via the Smart Signal App. The system then prioritizes clearing traffic from that lane by extending the green light duration or instantly switching signals to ensure smooth passage. This feature significantly reduces response time for emergency services, ensuring ambulances, fire trucks, and police vehicles can navigate through congested areas efficiently. The interface also displays traffic density at different junctions, helping drivers make informed route decisions.

**CHAPTER 7**

# **FUTURE SCOPE**

The SmartFlow Density based Traffic Management System has huge scope for future upgrades and larger-scale implementations. A major area for growth is integrating more sophisticated AI models for even more accurate vehicle detection and classification, with the use of more sophisticated deep learning methods. Such developments can enhance the accuracy of congestion evaluation, making traffic signal control even more responsive and adaptive. Also, integrating real-time weather and road condition monitoring can further optimize traffic management by optimizing signal timings for rain, fog, or accidents. New developments in the future can also be in the form of vehicle-to-infrastructure (V2I) communication, which enables smart cars to communicate directly with traffic lights for enhanced flow efficiency.

Another potential direction is taking the system to a city-wide smart traffic network with multiple junctions linked via a centralized cloud-based platform. Through the use of 5G and edge computing, real-time traffic analysis can be communicated among all junctions to facilitate efficient synchronized adjustment of signals as well as more intelligent rerouting mechanisms. The system also supports integration with public transport systems for bus or tram priority, lowering delays and enhancing urban mobility in general. Furthermore, blockchain technology can be utilized to improve data security and transparency, making traffic data tamper-proof while facilitating decentralized control among several city traffic departments.

Improvements can also include improved emergency vehicle tracking through the use of GPS and AI-based predictive modeling to predict traffic congestion and clear lanes beforehand. Furthermore, the user mobile app can be extended to incorporate voice-assisted navigation, congestion heatmaps, and personalized route suggestions. In response to mounting sustainability concerns, the system can also be modified to facilitate eco-friendly programs like making the use of electric vehicles more popular by giving preference to green lights for them in particular areas [1]-[21]. Finally, the incessant innovation in AI, IoT, and big data analytics will shape the future of intelligent traffic systems, rendering urban commuting safer, faster, and more environmentally friendly for future generations.

**CHAPTER 8**

# **CONCLUSION**

The SmartFlow Density based Traffic Management System offers a strong and effective solution to current traffic congestion problems through the convergence of artificial intelligence, IoT, and real-time monitoring. The system applies Raspberry Pi, Pi Camera, and AI-powered image processing (YOLOv3) to estimate vehicle density at intersections and adaptively modify signal lengths. This is achieved through ensuring an adaptive and data-driven traffic control system that greatly minimizes unnecessary waiting at signals, creating a smoother traffic flow and greater road efficiency. Through the deployment of a density-based signal control system, traffic distribution is maximized and congestion minimized overall, especally in denser urban populations.

To enable smooth data communication, the system utilizes a Flask-based backend and MySQL database to process traffic data in real-time, implement user authentication, emergency vehicle management, and congestion monitoring. The friendly mobile application, developed with Flutter on Visual Studio Code, offers real-time traffic information, rerouting proposals, and emergency vehicle clearance, which enhance road user convenience and safety. Also, the admin web interface, which is coded in Flask and PyCharm, is a central control panel that traffic authorities can use to view real-time traffic information, authorize emergency applications, and override signal controls manually when needed.

In addition to enhancing traffic efficiency and emergency response times, the system has wider implications for energy conservation and environmental gains. By minimizing vehicle idling time at intersections, the system leads to lower fuel consumption and carbon emissions, supporting sustainable urban mobility. Moreover, the capability to detect and respond to real-time congestion assists in reducing road stress and enhancing overall public transportation efficiency. The inclusion of intelligent traffic management by AI-driven automation represents a major leap in tackling the increasing problems of urbanization and making cities more efficient, secure, and eco-friendly.

# **REFERENCES**

1. Almatar, Khalid Mohammed, *“Implementing AI-Driven Traffic Signal Systems for Enhanced Traffic Management”*, International Journal of Sustainable Development & Planning, 2024, Vol 19, Issue 2, p781
2. Katsiaryna Bahamazava, *“AI-driven scenarios for urban mobility: Quantifying the role of ODE models and scenario planning in reducing traffic congestion”*, Department of Mathematical Sciences G.L. Lagrange, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Torino, 10129, Italy,2025
3. Yahia Said, Yahya Alassaf, Refka Ghodhbani, Yazan Ahmad Alsariera, Taoufik, *“AI-Based Helmet Violation Detection for TrafficManagement System”*, Saidani, Olfa Ben Rhaiem, Mohamad Khaled Makhdoum, Manel HleiliCMES - Computer Modeling in Engineering and Sciences Volume 141, Issue 1, 20 August 2024, Pages 733-749
4. Deng Pan a, Jiahao Lu a, Yao Li a b, Yuecheng Gao a c, *“Unified embedded operating system for vehicle control and traffic management”*, Journal of Industrial Information Integration Volume 44, March 2025, 100794
5. Chiara Manfletti, Marta Guimarães, Claudia Soares, *“AI for space traffic management”*, Journal of Space Safety Engineering Volume 10, Issue 4, December 2023
6. M.Pavan Kumar Reddy, V.Teja Vigneshwar, M.Shiva, B.Revanth, *“AI driven Traffic Management System using Computer Vision and ML”*, IJERST, Vol. 21 No. 1 (2025): Volume 21 No. 1 2025
7. Baoming Wang, Haotian Zheng, Kun Qian, Xiaoan Zhan, Junliang Wang, *“Edge computing and AI-driven intelligent traffic monitoring and optimization”*, Electrical and Computer Engineering, University of Illinois Urbana Champaign, Urbana, IL, USA,2024
8. Seyed Reza Samaei, *“A Comprehensive Algorithm for AI-Driven Transportation Improvements in Urban Areas”*, Advanced Research in Science, Engineering and Technology, Belgium,2024
9. Bibhu Dash1, Pawankumar Sharma, Meraj Ansari *“A Data-Driven AI Framework to Improve Urban Mobility and Traffic Congestion in Smart Cities”*, School of Computer and Information Sciences, University of the Cumberlands,2024
10. Benson K. H. Hung, *“Data-Driven Understanding Of AI-Based Traffic Signal Control And Its Implications On Low-Carbon Urban Transport Systems”*, International Journal of Big Data Mining for Global WarmingVol. 04, No. 01, 2241001 (2022)
11. Heng Zeng , Manal Yunis, Ayman Khalil , Nawazish Mirza, *“Towards a conceptual framework for AI- driven anomaly detection in smart city IoT networks for enhanced cybersecurity”*, Journal of Innovation & Knowledge Volume 9, Issue 4, October–December 2024, 100601
12. Murat Bakirci, *“Utilizing YOLOv8 for enhanced traffic monitoring in intelligent transportation systems (ITS) applications”*, Digital Signal Processing Volume 152, September 2024
13. Fredy Nocua M, Wilson-Javier P´erez-Holguín, Camilo Pardo-Beainy, *“Urban traffic monitoring based on deep learning on an embedded GPU”*, Universidad Santo Tomas ´ – Grupo GIDINT Tunja Colombia,2025.
14. Allah Ditta, Muhammad Maroof Ahmed, Tehseen Mazhar, Tariq Shahzad, Yazan Alahmed, Habib Hamam, *“Number plate recognition smart parking management system using IoT”*, Measurement: Sensors Volume 37, February 2025, 101409
15. S. Abirami, M. Pethuraj, M. Uthayakumar, P. Chitra, *“A systematic survey on big data and artificial intelligence algorithms for intelligent transportation system”*, Case Studies on Transport Policy Volume 17, September 2024
16. Huszák, Árpád and Simon, Vilmos and Bokor, László and Tizedes, László and Pekár, Adrián, *“An AI-Driven Intelligent Transportation System: Functional Architecture and Implementation”*, An AI-Driven Intelligent Transportation System: Functional Architecture and Implementation. INFOCOMMUNICATIONS JOURNAL,2024
17. Srinjoy Mitra , *“AI-driven predictive models for traffic flow in IoT-driven smart cities”*, Vol. 1 No. 2 (2024): Uncertainty Discourse and Applications
18. Jun Tang, Caixian Ye, Xianlai Zhou, Lijun Xu, *“YOLO-Fusion and Internet of Things: Advancing object detection in smart transportation”*, Alexandria Engineering Journal Volume 107, November 2024
19. Shih-Yu Sun, Ting-Hao Hsu, Chia-Yen Huang, Cheng-Han Hsieh, Chun-Wei Tsai, *“A Data Augmentation System for Traffic Violation Video Generation Based on Diffusion Model”*, Procedia Computer Science Volume 251, 2024
20. Quang Nguyen Huy Minh, Nen Nguyen Dinh, Long Viet Ho, Cuong Phan Huu, *“Real-time traffic accident detection using yolov8”*, Transportation Research Procedia Volume 85, 2025
21. Laura Deveer, Laura Minet , *“Real-time air quality prediction using traffic videos and machine learning”*, Transportation Research Part D: Transport and Environment Volume 142, May 2025

# **APPENDIX**

**Flask Backend**

**Application Website Interface**

import smtplib  
from email.mime.text import MIMEText  
  
from flask import \*  
import pymysql  
flow = Flask(\_\_name\_\_)  
con = pymysql.connect(host='localhost',user='root',password='root',db='smartsignalhub',charset='utf8')  
cmd = con.cursor()  
  
@flow.route("/userregister", methods=['GET', 'POST'])  
def userregister():  
 data = request.json  
 print(data)  
 name = data.get("name")  
 phone = data.get("phone")  
 email = data.get("email")  
 rc = data.get("rcNumber")  
 adhr = data.get("adharNumber")  
 password = data.get("password")  
 usertype = data.get("usertype")  
 emer\_id = data.get("id")  
  
 print(name, phone, email, rc, adhr, password, usertype, emer\_id)  
  
 # Check for existing data  
 cmd.execute("""  
 SELECT rc\_number, emergency\_id   
 FROM userreg   
 WHERE rc\_number=%s OR emergency\_id=%s  
 """, (rc,emer\_id))  
 res = cmd.fetchone()  
 print("res:", res)  
  
 if res is None:  
 # Insert into login table  
 cmd.execute("INSERT INTO login (email, password, type, otp) VALUES (%s, %s, 'pending', NULL)",  
 (email, password))  
  
 loginid = cmd.lastrowid  
  
 # Insert into userreg table  
 cmd.execute("""  
 INSERT INTO userreg (email, name, phone, rc\_number, aadhar\_number, login\_id, usertype, emergency\_id)  
 VALUES (%s, %s, %s, %s, %s, %s, %s, %s)  
 """, (email, name, phone, rc, adhr, loginid, usertype, emer\_id if emer\_id else None))  
  
 con.commit()  
 return jsonify({'task': "Successfully inserted"})  
 else:  
 # Identify which data already exists  
 existing\_data = []  
 if res[0] == rc:  
 existing\_data.append("RC Number already exists.")  
  
 if res[1] == emer\_id:  
 existing\_data.append("Emergency ID already exists.")  
  
 return jsonify({'task': "Failed", 'existing\_data': existing\_data})  
  
  
@flow.route("/logincheck",methods=['get','post'])  
def logincheck():  
 print(request.args)  
 email = request.args.get("username")  
 passwrd = request.args.get("password")  
 cmd.execute("select \* from login where email='"+str(email)+"' and password='"+str(passwrd)+"'")  
 result = cmd.fetchone()  
 print(result)  
 if result is None:  
 return jsonify({'task': "invalid"})  
 elif result[3] == 'user':  
 return jsonify({'task': "success",'loginid': result[0],'type':result[3]})  
 elif result[3] == 'admin':  
 return jsonify({'task':"success",'loginid':result[0],'type':result[3]})  
 elif result[3] == 'emergency':  
 return jsonify({'task': "success", 'loginid': result[0], 'type': result[3]})  
 else:  
 return jsonify({'task': "failed"})  
  
@flow.route("/getprofile",methods=['get','post'])  
def getprofile():  
 id=request.args.get("lid")  
 print(id)  
 cmd.execute("SELECT \* FROM `userreg` WHERE `login\_id`='"+id+"'")  
 s=cmd.fetchone()  
 print(s)  
 header=[x[0] for x in cmd.description]  
 json\_data=[]  
 if s:  
 json\_data.append(dict(zip(header, s)))  
 print(json\_data)  
 return jsonify(json\_data)  
  
@flow.route("/updateprofiledetails",methods=['get','post'])  
def updateprofiledetails():  
 data = request.json  
 print(data)  
 name = data.get("name")  
 phone = data.get("phone")  
 lid = data.get("lid")  
 cmd.execute("update userreg details set name='"+name+"',phone='"+phone+"' where login\_id='"+lid+"'")  
 con.commit()  
 return jsonify({'task':"success"})  
  
@flow.route("/deleteuser",methods=['get','post'])  
def deleteuser():  
 logid = request.args.get("login\_id")  
 cmd.execute("delete from user\_registration where login\_id='"+str(logid)+"'")  
 con.commit()  
 cmd.execute("DELETE FROM `login` WHERE `login\_id`='"+str(logid)+"'")  
 con.commit()  
 return jsonify({'task':"success"})  
  
  
@flow.route("/congestion\_alerts",methods=['get','post'])  
def congestion\_alerts():  
 cmd.execute("SELECT \* FROM `alerts` where alert\_type='congestion'")  
 # return jsonify({'task': "success"})  
 s = cmd.fetchall()  
 header = [x[0] for x in cmd.description]  
 json\_data = []  
 for result in s:  
 json\_data.append(dict(zip(header, result)))  
 print(json\_data)  
 return jsonify(json\_data)  
  
@flow.route("/changepassword",methods=['get','post'])  
def changepassword():  
 oldpassword = request.args.get("oldpassword")  
 newpassword = request.args.get("newpassword")  
 cmd.execute("select \* from login where password='"+oldpassword+"'")  
 result = cmd.fetchone()  
 if result is None:  
 return jsonify({'task': "Invalid"})  
 else:  
 cmd.execute("UPDATE login SET PASSWORD='"+newpassword+"' WHERE PASSWORD='"+oldpassword+"'")  
 con.commit()  
 return jsonify({'task': "Updated Succeccfully"})  
 return jsonify({'task': "success"})  
  
  
  
@flow.route("/traffic\_control",methods=['get','post'])  
def traffic\_control():  
 print(request.form)  
 Time = request.form.get("time")  
 Date = request.form.get("date")  
 Location = request.form.get("location")  
 route = request.form.get("route")  
 loginid = request.form.get("loginid")  
 cmd.execute("INSERT INTO traffic\_control VALUES(NULL,'"+Time+"','"+Date+"','"+Location+"','"+route+"','"+str(loginid)+"')")  
 con.commit()  
 return jsonify({'task': "success"})  
  
  
@flow.route("/livestats", methods=['get','post'])  
def livestats():  
 cmd.execute("SELECT \* FROM congestion")  
 s = cmd.fetchall()  
 print(s)  
 header = [x[0] for x in cmd.description]  
 json\_data = []  
 for result in s:  
 # Convert timedelta object to string representation  
 result = list(result)  
 result[1] = str(result[1])  
 json\_data.append(dict(zip(header, result)))  
 print(json\_data)  
 return jsonify(json\_data)  
@flow.route('/sendfeedback',methods=['post','get'])  
def sendfeedback():  
 id=request.args.get('lid')  
 feedback=request.args.get('feedback')  
 cmd.execute("insert into feedback values(null,'"+feedback+"','"+str(id)+"')")  
 con.commit()  
 return jsonify({'task': "success"})  
  
  
from datetime import datetime  
  
  
@flow.route('/sendemergencyalert', methods=['POST', 'GET'])  
def sendemergencyalert():  
 lid = request.args.get('lid')  
 data = request.get\_json()  
 print(data, '\*\*\*\*\*\*\*\*\*\*\*')  
  
 location = data.get('location')  
 alert = data.get('alert')  
  
 # Get current date and time  
 now = datetime.now()  
 current\_date = now.strftime("%Y-%m-%d") # Format: YYYY-MM-DD  
 current\_time = now.strftime("%I:%M%p") # Format: HH:MMam/pm  
  
 cmd.execute("insert into alert values (null,'"+current\_date+"','"+current\_time+"','"+location+"','"+alert+"','"+lid+"')")  
  
 con.commit()  
  
 return {"message": "Alert sent successfully!"}  
@flow.route("/checkotp",methods=['post','get'])  
def checkotp():  
 email = request.args.get("email")  
 print(email)  
 otp=request.args.get("otp")  
 cmd.execute("SELECT otp,login\_id FROM login WHERE email='"+email+"'")  
 s=cmd.fetchone()  
 otp\_value = s[0]  
 lid=s[1]  
  
 if otp==otp\_value:  
 password=request.args.get("password")  
 cmd.execute("UPDATE login SET password='"+password+"' WHERE login\_id='"+str(lid)+"'")  
 con.commit()  
 return jsonify({'status': 'sucess', 'message': 'Success'})  
 else:  
 return jsonify({'status': 'error', 'message': 'Incorrect OTP'})  
  
  
@flow.route("/forgot\_password", methods=['POST'])  
def forgot\_password():  
 # email = request.json.get('email')  
 import random  
 list\_of\_chars1 = "2345698"  
  
 # inside your function  
 otp = "11"  
 length1 = 3  
 for \_ in range(length1):  
 otp += random.choice(list\_of\_chars1)  
  
 mymail = request.args.get('email')  
 print(mymail)# Fetch the email address  
 cmd.execute("SELECT \* FROM login WHERE email='"+mymail+"'")  
 result=cmd.fetchone()  
 print(result)  
 if result:  
 # mymail='anjuraj118@gmail.com'  
  
  
 try:  
 gmail = smtplib.SMTP('smtp.gmail.com', 587)  
 gmail.ehlo()  
 gmail.starttls()  
 gmail.login('kaswathi036@gmail.com', 'zboc nllq pnja zmwp')  
  
 msg = MIMEText("Your otp number is: " + otp)  
 msg['Subject'] = 'otp'  
 msg['To'] = mymail  
 msg['From'] = 'www.kaswathi036@gmail.com'  
  
 gmail.send\_message(msg)  
 gmail.quit()  
 cmd.execute("UPDATE login SET otp='"+otp+"' WHERE email='"+mymail+"'")  
 con.commit()  
 return jsonify({'status': 'sucess', 'message': 'Success.'})  
 except Exception as e:  
 print("Couldn't sendemail:",e)  
  
 return jsonify({'status': 'error', 'message': 'User notfound.'})  
 else:  
 return jsonify({'status': 'error', 'message': 'Incorrect Email.'})  
@flow.route('/controljunctions',methods=['post','get'])  
def controljunctions():  
 data=request.json  
 junction=data.get('junction')  
 lid=data.get('lid')  
  
 return jsonify({'status': 'sucess', 'message': 'Success.'})  
  
  
  
flow.run(host='0.0.0.0',port=8000)

**Flask-Based AI Traffic Management System with User Authentication & Real-Time Traffic Control**

import smtplib  
from email.mime.text import MIMEText  
  
from flask import \*  
import pymysql as pymysql  
  
flow= Flask(\_\_name\_\_)  
flow.secret\_key = "abc"  
con = pymysql.connect(host="localhost", user="root", password="root", port=3306, db="smartsignalhub", charset="utf8")  
cmd = con.cursor()  
import functools  
def login\_required(func):  
 @functools.wraps(func)  
 def secure\_function():  
 if "lid" not in session:  
 return redirect("/")  
 return func()  
 return secure\_function  
  
  
@flow.route('/logout')  
def logout():  
 session.clear()  
 return redirect('/')  
  
  
  
@flow.route('/')  
def login():  
 return render\_template("login.html")  
@flow.route('/logincheck', methods=['post'])  
def logincheck():  
 user = request.form['username']  
 psd = request.form['password']  
 cmd.execute("select \* from `login` where email='" + user + "' and password='" + psd + "'")  
 result = cmd.fetchone()  
 if result is None:  
 return '''<script>alert("INVALID USERNAME AND PASSWORD");window.location='/'</script>'''  
 elif result[3] == "admin":  
 session['lid'] = result[0]  
 cmd.execute("SELECT `userreg`.\*,`feedback`.\* FROM `feedback` JOIN `userreg` WHERE `userreg`.`login\_id`=`feedback`.`userid`")  
 result=cmd.fetchall()  
 cmd.execute("SELECT COUNT(\*) AS requestcount FROM userreg JOIN login ON login.login\_id = userreg.login\_id WHERE login.type = 'user' OR login.type = 'emergency'")  
 usercount = cmd.fetchone()[0]  
 cmd.execute("SELECT COUNT(\*) AS pending\_count FROM userreg JOIN login ON login.login\_id = userreg.login\_id WHERE login.type = 'pending'")  
 requestcount=cmd.fetchone()[0]  
 cmd.execute("SELECT COUNT(\*) AS rejected\_count FROM userreg JOIN login ON login.login\_id = userreg.login\_id WHERE login.type = 'rejected'")  
 rejected=cmd.fetchone()[0]  
 cmd.execute("SELECT COUNT(\*) AS emergency\_count FROM userreg WHERE usertype = 'emergency'")  
 emergencycount=cmd.fetchone()[0]  
 cmd.execute("select feedback from feedback")  
 feedback=cmd.fetchone()  
 return render\_template('index.html',value=result,usercount=usercount,requestcount=requestcount,rejected=rejected,emergencycount=emergencycount,feedback=feedback)  
@flow.route('/index')  
def index():  
 cmd.execute(  
 "SELECT `userreg`.\*,`feedback`.\* FROM `feedback` JOIN `userreg` WHERE `userreg`.`login\_id`=`feedback`.`userid`")  
 result = cmd.fetchall()  
 cmd.execute("SELECT COUNT(\*) AS requestcount FROM userreg JOIN login ON login.login\_id = userreg.login\_id WHERE login.type = 'user' OR login.type = 'emergency'")  
 usercount=cmd.fetchone()[0]  
 cmd.execute("SELECT COUNT(\*) AS pending\_count FROM userreg JOIN login ON login.login\_id = userreg.login\_id WHERE login.type = 'pending'")  
 requestcount = cmd.fetchone()[0]  
 cmd.execute("SELECT COUNT(\*) AS rejected\_count FROM userreg JOIN login ON login.login\_id = userreg.login\_id WHERE login.type = 'rejected'")  
 rejected = cmd.fetchone()[0]  
 cmd.execute("SELECT COUNT(\*) FROM `login` JOIN `userreg` ON `login`.`login\_id` = `userreg`.`login\_id` WHERE `login`.`type` = 'emergency'")  
 emergencycount = cmd.fetchone()[0]  
 cmd.execute("select feedback from feedback")  
 feedback = cmd.fetchone()[0]  
 return render\_template('index.html',value=result,usercount=usercount,requestcount=requestcount,rejected=rejected,emergencycount=emergencycount,feedback=feedback)  
  
@flow.route('/users')  
def users():  
 return render\_template("users.html")  
  
  
  
@flow.route('/viewuser')  
@login\_required  
def viewuser():  
 cmd.execute("select userreg.\*,login.type from `userreg` join login on login.`login\_id`=`userreg`.`login\_id` where login.type='user' or login.type='emergency'")  
 result=cmd.fetchall()  
 return render\_template("view\_user.html",value=result)  
@flow.route('/viewnewrequest')  
@login\_required  
def viewnewrequest():  
 cmd.execute("SELECT userreg.\*, login.type FROM `userreg` JOIN `login` ON `login`.`login\_id` = `userreg`.`login\_id` WHERE login.type = 'pending'")  
 result = cmd.fetchall()  
 return render\_template("newrequest.html", value=result)  
@flow.route('/emergencyvehicles')  
@login\_required  
def emergencyvehicles():  
 cmd.execute("SELECT `login`.\*,`userreg`.\* FROM `login` JOIN `userreg` ON `login`.`login\_id`=`userreg`.`login\_id` WHERE login.`type`='emergency'")  
 result=cmd.fetchall()  
 return render\_template('view\_emergency.html',value=result)  
  
@flow.route('/viewrejecteduser')  
@login\_required  
def viewrejecteduser():  
 cmd.execute("select userreg.\*,login.type from `userreg` join login on login.`login\_id`=`userreg`.`login\_id` where login.type='rejected' ")  
 result=cmd.fetchall()  
 return render\_template("rejecteduserview.html",value=result)  
@flow.route('/newrequestconfirm')  
@login\_required  
def newrequestconfirm():  
 usertype = request.args.get("type")  
 loginid = request.args.get("id")  
 print(usertype)  
 print(loginid)  
 session['type']=usertype  
 session['id']=loginid  
 return render\_template("newrequestconfirm.html")  
@flow.route('/rejnewrequestconfirm')  
@login\_required  
def rejnewrequestconfirm():  
 usertype = request.args.get("type")  
 loginid = request.args.get("id")  
 print(usertype)  
 print(loginid)  
 session['type']=usertype  
 session['id']=loginid  
 return render\_template("rejectnewrequestconfirm.html")  
@flow.route('/acceptuser',methods=['post'])  
@login\_required  
def acceptuser():  
 usertype=session["type"]  
 loginid=session["id"]  
 cmd.execute("UPDATE login JOIN userreg ON login.`login\_id`=`userreg`.`login\_id` SET login.`type`='"+usertype+"' WHERE userreg.`usertype`='"+usertype+"' AND `login`.`login\_id`='"+loginid+"' ")  
 con.commit()  
 return '''<script>alert("Accepted Succesfully");window.location='/index'</script>'''  
@flow.route('/rejectuser',methods=['post'])  
@login\_required  
def rejectuser():  
 type=session['type']  
 print(type,'\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  
  
 loginid=session["id"]  
 print(loginid,'\*\*\*\*\*\*\*\*\*\*\*')  
 cmd.execute("UPDATE login JOIN userreg ON login.`login\_id`=`userreg`.`login\_id` SET login.`type`='rejected' WHERE `login`.`login\_id`='"+loginid+"'AND userreg.`usertype`='"+type+"'")  
 con.commit()  
 return '''<script>alert("rejected Sucessfully");window.location='/index'</script>'''  
@flow.route('/viewtraffic')  
@login\_required  
def viewtraffic():  
 cmd.execute("""  
 SELECT `trafficcontrol`.\*, `userreg`.\*   
 FROM `trafficcontrol`   
 JOIN `userreg`   
 ON `trafficcontrol`.`login\_id` = `userreg`.`login\_id`  
 """)  
 res = cmd.fetchall()  
 print(res)  
 return render\_template("viewtrafficcontrol.html", value=res)  
  
@flow.route('/viewcongestion')  
@login\_required  
def viewcongestion():  
 cmd.execute("SELECT `congestion`.\* FROM `congestion`")  
 result=cmd.fetchall()  
 return render\_template("viewcongestion.html",value=result)  
@flow.route('/suggestroute')  
def suggestroute():  
 conid=request.args.get("conid")  
 session["congestionid"]=conid  
 return render\_template("suggestroute.html")  
@flow.route('/updateroute',methods=['post'])  
@login\_required  
def updateroute():  
  
 conid=session["congestionid"]  
 route=request.form["sroute"]  
 cmd.execute("update congestion set suggested\_route='"+route+"' where congestion\_id='"+conid+"'")  
 con.commit()  
 return '''<script>alert(" SUCCESS");window.location='/viewcongestion'</script>'''  
@flow.route('/alert')  
@login\_required  
def alert():  
 cmd.execute("SELECT `alert`.\*, `userreg`.\* FROM `alert` JOIN `userreg` ON `alert`.`loginid` = `userreg`.`login\_id` ORDER BY `alert`.`date` DESC LIMIT 1")  
 res=cmd.fetchone()  
  
 return render\_template("alert.html",value=res)  
  
  
  
flow.run(debug=True)