Overcrowded and Overloaded Vehicle Detection Through CCTV

Submitted by:

Hassaan Waheed (20P-0474)
Ijaz Tufail (20P-0641)

Session: 2020-2025

National University of Computer and Emerging Sciences

Peshawar, Pakistan

June, 2025

Student's Declaration

We declare that this project titled "Overcrowded and Overloaded Vehicle Detection Through CCTV," submitted as a requirement for the award of the degree of Bachelor of Science in Computer Science, does not contain any material previously submitted for a degree in any university; and that, to the best of our knowledge, it does not contain any materials previously published or written by another person except where due reference is made in the text.

We understand that the Department of Computer Science, National University of Computer and Emerging Sciences, has a zero-tolerance policy towards plagiarism. Therefore, we, as authors of the above-mentioned thesis, solemnly declare that no portion of our thesis has been plagiarized, and any material used from other sources is properly referenced.

Student Name 1	Signature:
Student Name 2	Signature:

Verified by Plagiarism Cell Officer
Dated:

Certificate of Approval

The Department of Computer Science, National University of Computer and Emerging Sciences, accepts this thesis titled *Overloaded and Overcrowded Vehicle Detection through CCTV*, submitted by Hassaan Waheed (20P-0474), Ijaz Tufail (20P-0641), in its current form, and it is satisfying the dissertation requirements for the award of Bachelors Degree in Computer Science.

Supervisor **Signature:** ______________________
Mr. Samin Ahmed

____

Mr. Haroon Zafar
FYP Coordinator
National University of Computer and Emerging Sciences, Peshawar

____

Mr. Fazl e Basit
HoD of Department of Computer Science
National University of Computer and Emerging Sciences

Acknowledgements

We would like to express our gratitude to our supervisors, peers, and the Department of Computer Science for their invaluable guidance and support throughout this project. Special thanks to the organizations that provided data and resources essential for the system's development.

Abstract

This project aims to develop an automated system for detecting overcrowded and overloaded vehicles through CCTV. Leveraging deep learning and computer vision, the system analyzes live video feeds to identify violations in real-time, providing alerts to law enforcement agencies. By integrating cloud infrastructure for data processing and storage, the project seeks to enhance road safety and optimize traffic monitoring systems.

Contents

1 Introduction

Road traffic accidents are a significant global issue, with overcrowded and overloaded vehicles contributing heavily to fatalities. The inefficiency of manual monitoring methods, particularly in developing countries, necessitates an automated solution. This project introduces a scalable AI-based system using existing CCTV infrastructure to enhance road safety and enforce regulations effectively.

1.1 Background and Motivation

Urban traffic congestion is a growing challenge in modern cities, leading to increased travel time, environmental pollution, and road safety risks. Overloaded and overcrowded vehicles contribute significantly to this issue, posing dangers such as compromised structural integrity, higher accident rates, and inefficient traffic flow.

Traditional manual enforcement of overloaded vehicles is ineffective due to resource constraints and human error. The lack of an automated surveillance system allows violators to evade penalties, further worsening the problem.

With the advent of computer vision and machine learning, automated vehicle monitoring through CCTV has become a viable solution. By integrating AI-based analytics with real-time video feeds, authorities can detect overloaded and overcrowded vehicles efficiently, ensuring compliance with safety regulations and reducing traffic hazards.

This project aims to develop a smart surveillance system that leverages deep learning techniques to identify, track, and flag such violations in real-time.

2 Review of Literature

This section reviews existing research on vehicle detection systems, focusing on machine learning methods like YOLO and image processing techniques. Current solutions are analyzed for their limitations, such as low scalability and environmental dependencies, providing a foundation for the proposed system's objectives.

2.1 Performance Analysis of Deep Convolutional Network Architectures for Classification of Over-Volume Vehicles

#	**Summary of Findings**	**Implications**	**Limitations**	**Further Research Directions**
1	The research introduces DCNNs such as MobileNetV2, VGG19, ResNet50, and EfficientNet to classify over-volume vehicles. ResNet and EfficientNet performed best, achieving over 95% accuracy.	Aids traffic monitoring and helps enforce load regulations, improving road safety.	High computational cost, especially for ResNet models, due to complex architectures and processing requirements.	Developing a more balanced dataset or integrating additional vehicle classes for improved model generalization.
2	Utilized a dataset of 3054 images with three main categories. Augmentation techniques included flipping, noise addition, and warping.	Improves smoothness and accuracy, essential for handling variations in real-world vehicle types.	Limited dataset size restricts model performance and might not capture all vehicle scenarios.	Expand dataset to include a wider variety of vehicle types and environmental conditions to improve robustness.

2.2 Convolutional Neural Network for Overcrowded Public Transportation Pickup Truck Detection

#	**Summary of Findings**	**Implications**	**Limitations**	**Further Research Directions**
1	This study focuses on detecting overcrowded public transportation pickup trucks (PTPT) in Thailand using YOLOv5 and a custom-labeled dataset. Achieved 95.1% mAP at 33 FPS on GPU.	Demonstrates potential for reducing road fatalities by enforcing overloading laws through automated detection.	Performance limitations on CPU (2 FPS) hinder real-time application in low-resource environments.	Future studies could explore optimization techniques or hardware to improve inference speed.

2	Utilized various YOLOv5 models with transfer learning. YOLOv5L balanced speed and accuracy best.	Offers a reliable model for real-time object detection tasks.	Dataset limitations may restrict model generalization to other vehicle types and environmental conditions.	Expanding the dataset to cover a broader range of public transportation types and weather conditions.

2.3 Automatic License Plate Recognition (ALPR)

#	Summary of Findings	Implications	Limitations	Further Research Directions
1	ALPR systems extract license plate information from images, used widely in toll payments and traffic surveillance. Techniques vary, handling different environments and plate types.	Key for traffic management and payment systems, improving efficiency.	ALPR accuracy depends on image quality and environmental factors, such as lighting, occlusions, and varied plate designs.	Improving ALPR's robustness to diverse environments, languages, fonts, and plate conditions.

2.4 Automatic License Plate Recognition (LPR)

#	Summary of Findings	Implications	Limitations	Further Research Directions
1	The study proposes an LPR technique with two modules: a license plate locating module using fuzzy logic, and a license number identification module based on neural networks.	Useful for applications in various lighting and movement conditions.	Success rates decline under poor lighting, high speed, and diverse backgrounds.	Future work could address optimization in low-light and high-speed scenarios.

3 Project Vision

3.1 Problem Statement

The absence of automated systems for detecting overloaded and overcrowded vehicles has led to increased road fatalities and traffic inefficiencies, especially in resource-constrained environments.

3.2 Business Opportunity

By enabling real-time monitoring and enforcement, the system offers significant value to transportation departments and logistics companies, improving regulatory compliance and reducing operational risks.

3.3 Objectives

- To develop a robust vehicle monitoring system capable of real-time detection.
- To generate alerts for overcrowded and overloaded vehicles.
- To provide actionable analytics for traffic management.

3.4 Project Scope

The project focuses on urban highways, integrating with existing traffic surveillance systems to deliver scalable and efficient solutions.

3.5 Constraints

- Variations in video quality due to lighting and weather.
- Real-time computational limitations in high-traffic areas.
- Privacy concerns regarding surveillance data.

3.6 Stake Holder Description

The project focuses on urban highways, integrating with existing traffic surveillance systems to deliver scalable and efficient solutions.

3.6.1 Stakeholders Summary

Stakeholders in this project include individuals, organizations, or entities that are directly or indirectly impacted by the system. These stakeholders have varying levels of influence and interest in the project's development and outcome. The key stakeholders include:

- **End Users:** Individuals or businesses that will use the system for vehicle classification, license plate recognition, or public transport monitoring.
- **Traffic Authorities:** Government agencies responsible for road safety, traffic management, and law enforcement.

- **Researchers and Developers:** AI and machine learning professionals working on improving detection models and algorithms.
- **Regulatory Bodies:** Organizations that define policies for vehicle weight restrictions, overloading limits, and public transport regulations.
- **Hardware and Software Vendors:** Companies providing hardware components such as cameras, GPUs, and software tools used for model implementation.
- **Policy Makers:** Government officials responsible for enforcing traffic laws and policies.

3.7 Key High-Level Goals and Problems of Stakeholders

Each stakeholder has distinct goals and faces specific challenges related to the deployment and effectiveness of the system.

3.7.1 End Users

Goals:

- Ensure accurate and efficient vehicle classification and license plate recognition.
- Improve traffic flow and reduce congestion.
- Enhance user experience through real-time detections.

Problems:

- Potential errors in detection due to environmental factors (e.g., lighting, occlusions).
- The need for a user-friendly interface with minimal false positives.
- Hardware and software constraints affecting deployment in low-resource settings.

3.7.2 Traffic Authorities

Goals:

- Implement automated monitoring for overloaded vehicles and public transportation.
- Reduce road accidents caused by vehicle overloading.
- Streamline law enforcement using AI-powered detection.

Problems:

- High computational costs for real-time monitoring at scale.
- Variations in vehicle types affecting model accuracy.
- Need for infrastructure improvements (cameras, processing units) in low-resource environments.

3.7.3 Researchers and Developers

Goals:

- Improve deep learning models for vehicle classification.
- Optimize AI performance for real-time applications.
- Address dataset limitations and biases.

Problems:

- Lack of diverse datasets impacting model generalization.
- Overfitting issues when training with limited samples.
- High hardware requirements for advanced deep learning models.

3.7.4 Regulatory Bodies

Goals:

- Define policies and regulations for automated vehicle classification.
- Ensure ethical and unbiased AI deployment in traffic monitoring.
- Support innovation in AI-driven traffic management.

Problems:

- Lack of standardized regulations for AI-powered traffic systems.
- Privacy concerns related to automated vehicle tracking.
- Slow adoption of AI due to bureaucratic challenges.

3.7.5 Hardware and Software Vendors

Goals:

- Develop cost-effective hardware solutions for real-time traffic monitoring.
- Ensure compatibility between AI models and edge computing devices.
- Improve performance of AI software for large-scale deployment.

Problems:

- High costs of GPUs and processing hardware.
- Difficulty in integrating AI models with existing traffic monitoring infrastructure.
- Latency issues in real-time video processing.

3.7.6 Policy Makers

Goals:

- Establish regulations that balance AI innovation with ethical considerations.
- Promote AI-driven solutions to improve traffic safety.
- Foster collaboration between government agencies and AI researchers.

Problems:

- Balancing innovation with legal and ethical concerns.
- Resistance to AI adoption due to lack of awareness.
- Ensuring policies are flexible enough to accommodate future advancements.

4 Software Requirements Specifications

4.1 List of Features

- Detection of overloaded and overcrowded vehicles.
- Real-time alerts to enforcement authorities.
- Data analytics and reporting dashboard.

4.2 Functional Requirements

- Process live video feeds and detect violations.
- Generate and store alerts in real-time.
- Provide analytical insights for policy recommendations.

4.3 Non-Functional Requirements

- **Performance:** Ensure video feeds are processed with minimal latency.
- **Scalability:** Support multiple video streams across locations.
- **Security:** Encrypt data during transmission and storage to maintain privacy.
- **Reliability:** Maintain consistent performance under varying conditions.

4.4 Use Cases/ Use Case Diagram

The Use Case Diagram provides a high-level overview of the system's interactions with different users, highlighting the various functionalities the system offers. It serves as a visual representation of user roles and their respective actions, helping developers and stakeholders understand system requirements, user expectations, and functional boundaries. This diagram ensures clarity in system design by defining key use cases and their relationships with actors.

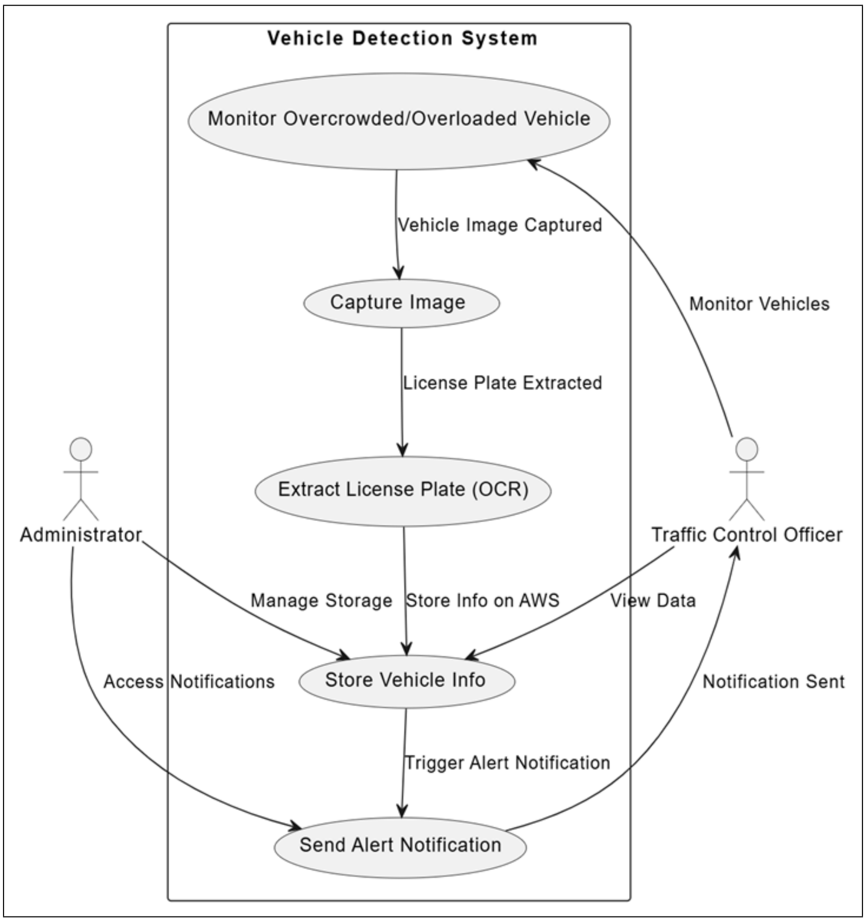

Figure 1: Use Case Diagram

4.5 Sequence Diagrams/System Sequence Diagram

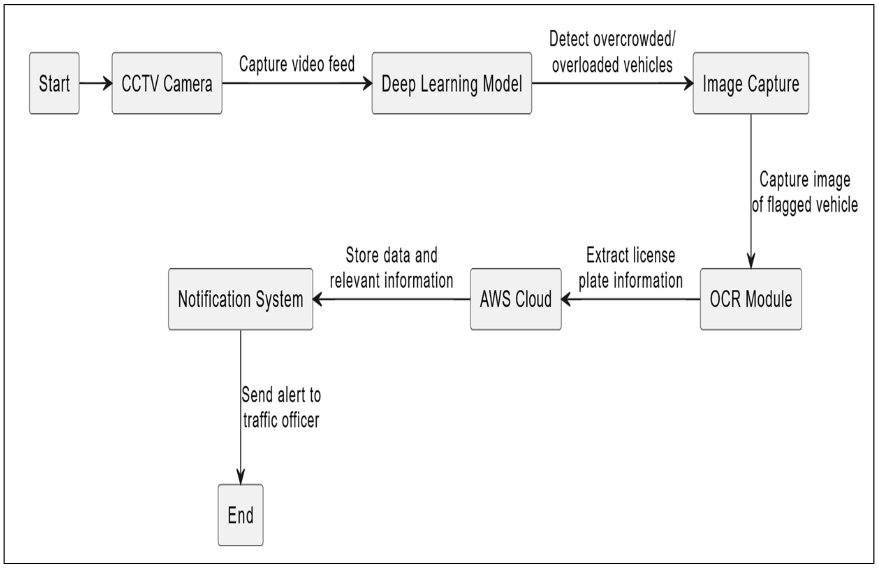

Figure 2: System Diagram

4.6 Test Plan (Test Level, Testing Techniques)

4.6.1 Test Levels

The testing process is structured in multiple levels to ensure system correctness and reliability.

- **Unit Testing:** Individual components such as the YOLO model, OCR module, and AWS storage are tested separately.
- **Integration Testing:** Validates how different modules interact (e.g., vehicle detection transferring data to OCR for license plate extraction).
- **System Testing:** Examines the entire system under various conditions to ensure stability, performance, and security.
- **User Acceptance Testing (UAT):** Conducted with traffic enforcement agencies (dummy) to validate the system in real-world applications.

4.6.2 Testing Techniques

- **Black-Box Testing:** Evaluates input-output correctness without considering internal logic.
- **White-Box Testing:** Examines internal workings, particularly for AI model accuracy and API responses.
- **Regression Testing:** Ensures that updates or changes do not break previously working features.
- **Performance Testing:** Measures response time, computational efficiency, and model latency.

4.6.3 Test Cases

Test Case ID	Description	Expected Output	Actual Output	Status
TC-001	Detect an overloaded vehicle in daylight	System detects and flags the vehicle	TBD	Pending
TC-002	Detect an over-crowded bus in low-light conditions	System correctly identifies overcapacity	TBD	Pending
TC-003	Extract a license plate from a blurry image	System accurately extracts numbers	TBD	Pending
TC-004	Send an alert when an overloaded vehicle is detected	Notification is received by the officer	TBD	Pending
TC-005	Process video feed with multiple vehicles simultaneously	System correctly detects all target vehicles	TBD	Pending
TC-006	Detect an overloaded vehicle in extreme weather conditions (rain, fog)	System correctly identifies overloading despite environmental disturbances	TBD	Pending
TC-007	Process real-time video at different FPS (Frames per second) rates	System maintains consistent performance	TBD	Pending
TC-008	Test the system's response when a partially visible vehicle is present	The system correctly detects and classifies the vehicle	TBD	Pending
TC-009	Detect motorcycles and small vehicles	The system correctly ignores non-relevant vehicle types	TBD	Pending
TC-010	Test storage and retrieval of detection records from AWS	The system successfully logs and retrieves detection records without data loss	TBD	Pending

Table 5: Test Cases for Overloaded and Overcrowded Vehicle Detection System

4.7 Software Development Plan

4.7.1 Development Methodology

The project follows an **Agile Scrum** methodology, ensuring iterative development and flexibility.

- **Sprints:** Each sprint lasts **two weeks** with well-defined deliverables.
- **Daily Standups:** Regular discussions to track progress and resolve issues.
- **Sprint Reviews:** Evaluation of detection accuracy, system performance, and user feedback.

4.7.2 Development Phases

Phase	Task	Duration	Responsible Team
Phase 1	Research and Requirement Gathering	2 Weeks	Hassaan
Phase 2	Model Training and Dataset Preparation	4 Weeks	Ijaz
Phase 3	YOLO Model Integration	3 Weeks	Ijaz
Phase 4	OCR Module Development	2 Weeks	Ijaz
Phase 5	Cloud Storage & AWS Deployment	3 Weeks	Hassaan
Phase 6	Alert Notification System	2 Weeks	Ijaz
Phase 7	Testing and Debugging	4 Weeks	Hassaan
Phase 8	Deployment and Final Review	2 Weeks	Hassaan

Table 6: Development Phases for the Project

4.7.3 Technology Stack

Component	Technology Used
Object Detection	YOLOv11L
OCR (License Plate Recognition)	Tesseract, EasyOCR
Backend	Python (Flask/Django)
Cloud Storage	AWS S3, DynamoDB
Frontend	React.js / Vue.js
Notifications	Firebase, Twilio API

Table 7: Technology Stack Used in the Project

4.7.4 Risk Assessment and Mitigation

Risk	**Impact**	**Mitigation Strategy**
Model misclassifies vehicles	High	Retrain the model with a diverse dataset and improve feature extraction techniques.
OCR fails in low-light images	Medium	Use preprocessing techniques such as adaptive histogram equalization and noise reduction before OCR extraction.
High latency in cloud processing	High	Implement edge computing or optimize cloud infrastructure for reduced processing time.
Integration issues between modules	Medium	Conduct regular integration testing and use API version control to ensure module compatibility.
Hardware limitations for real-time processing	High	Optimize model inference using TensorRT, ONNX, or other AI model compression techniques.
Privacy concerns regarding vehicle tracking	High	Implement encryption techniques and limit data storage duration to comply with data privacy regulations.
Camera quality affecting detection accuracy	Medium	Use high-resolution cameras and apply AI-based super-resolution techniques to enhance input quality.
Incorrect alerts leading to enforcement errors	Medium	Use a confidence threshold for alerts and provide a manual verification step for authorities.
Scalability issues when expanding to multiple locations	High	Design a modular and distributed architecture to handle multiple video streams efficiently.

Table 8: Risk Assessment and Mitigation Strategies

4.8 Wire-frames

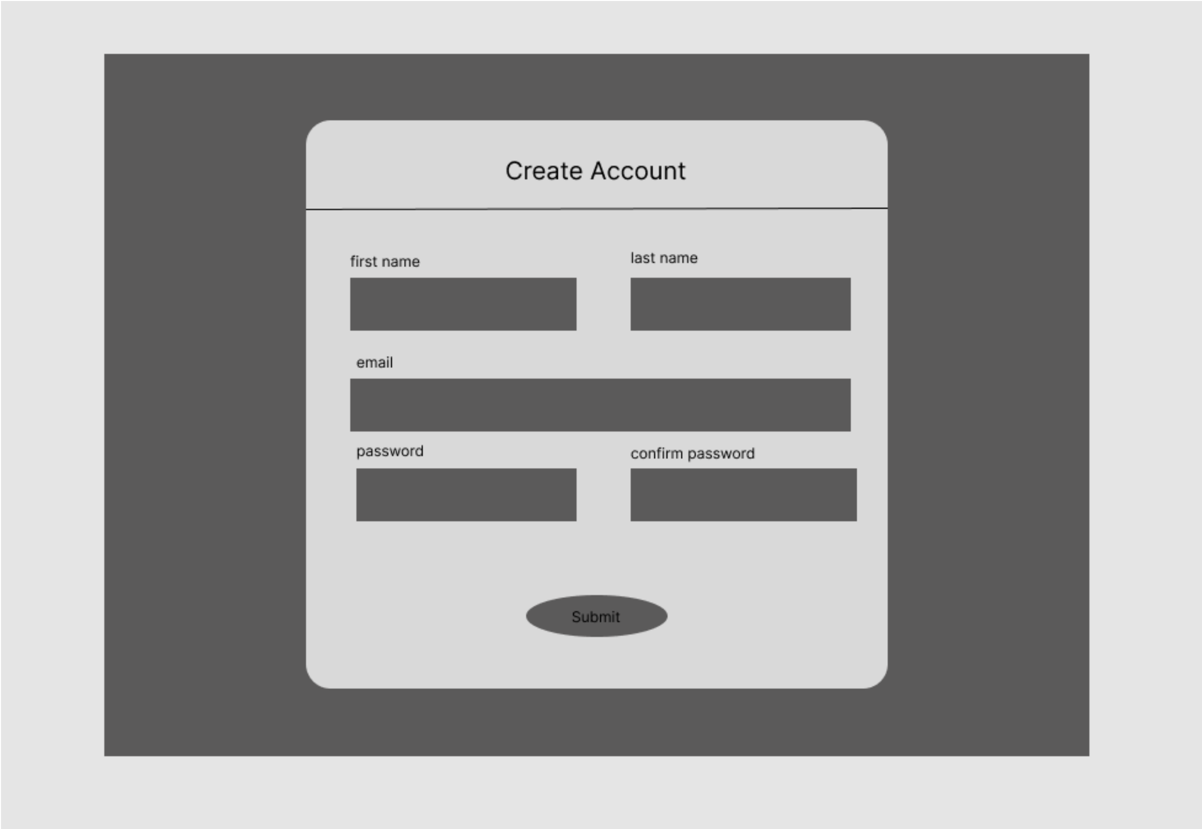

Figure 3: SignUp

Figure 4: Homepage

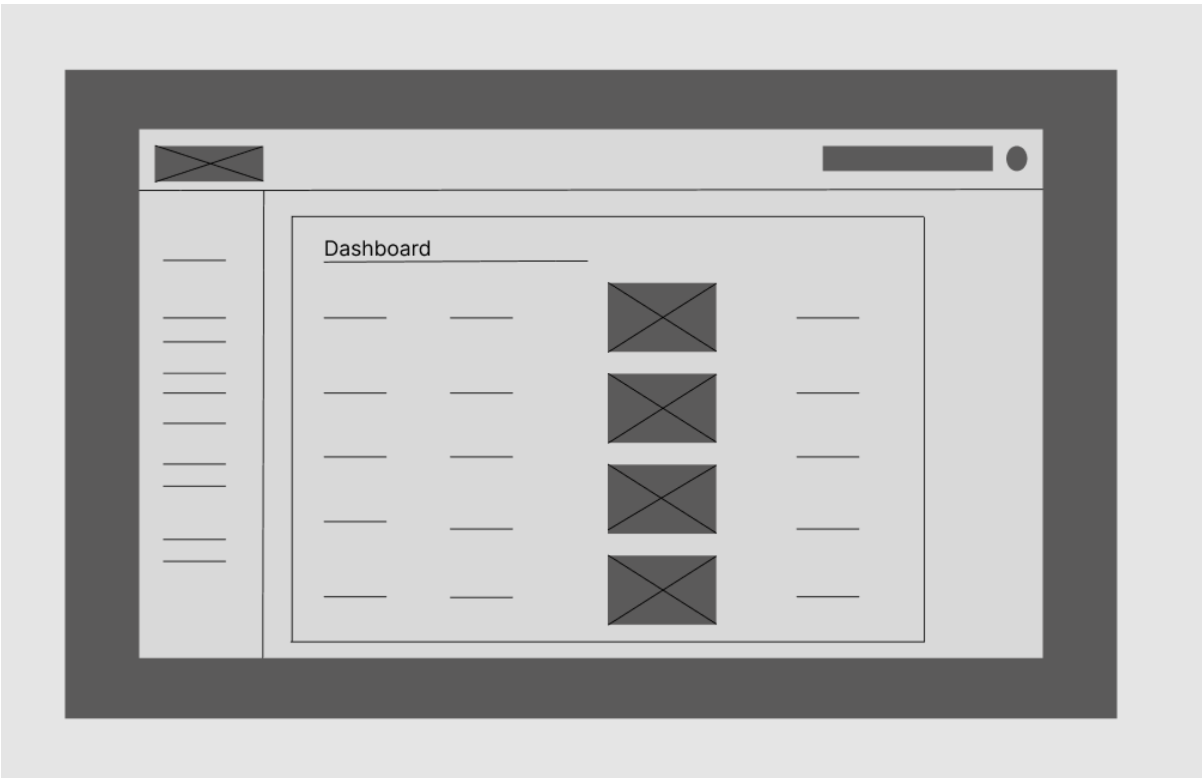

Figure 5: Dashboard

4.9 UI Screens

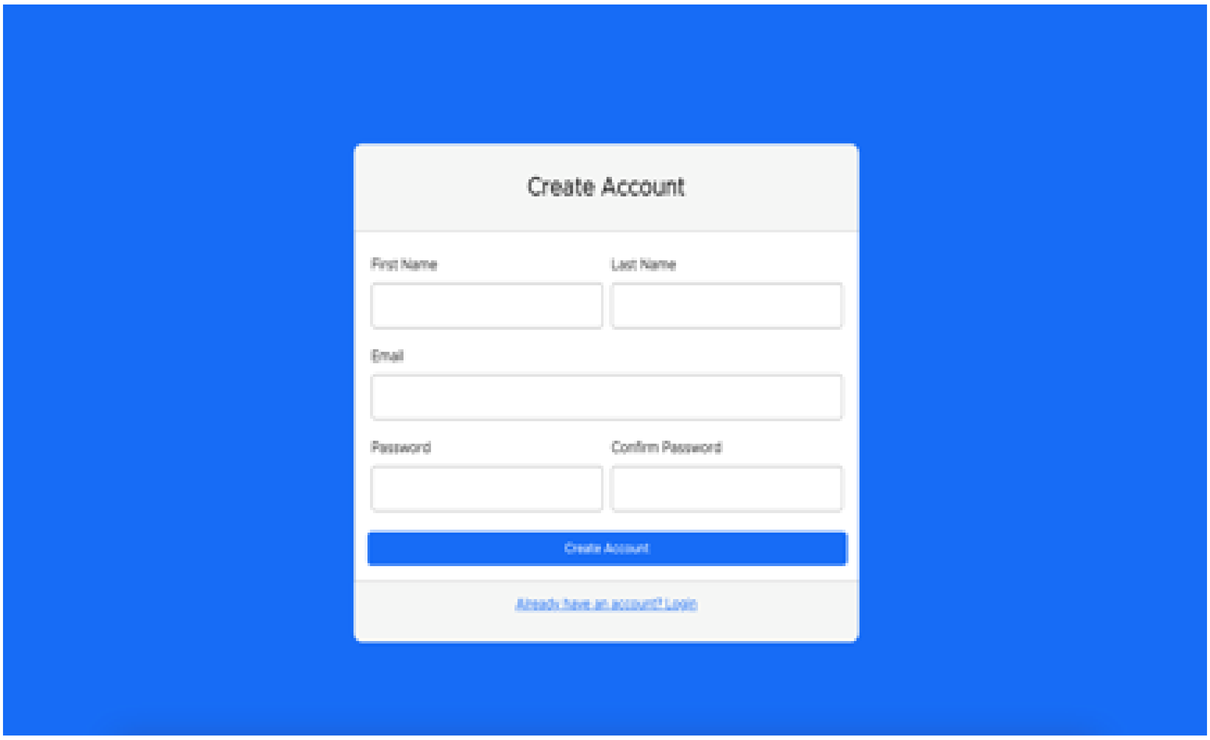

Figure 6: SignUp

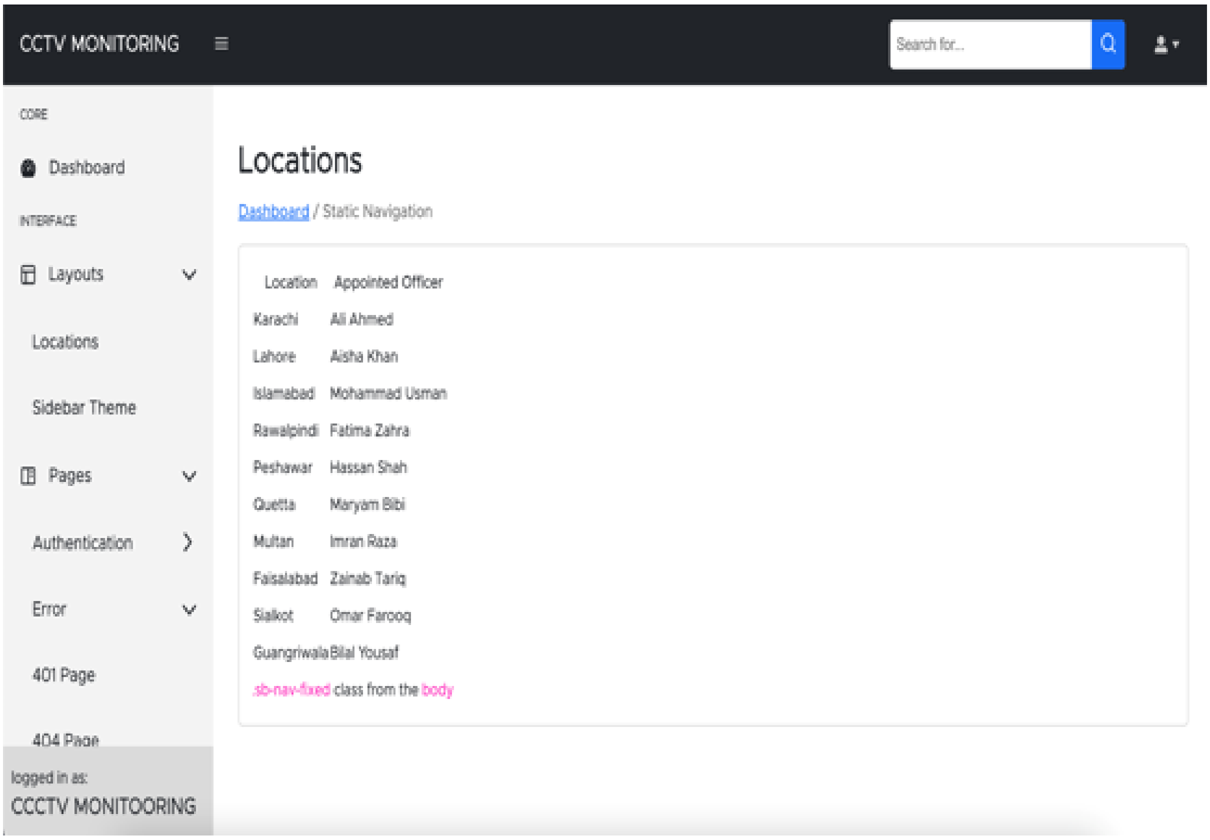

Figure 7: Home

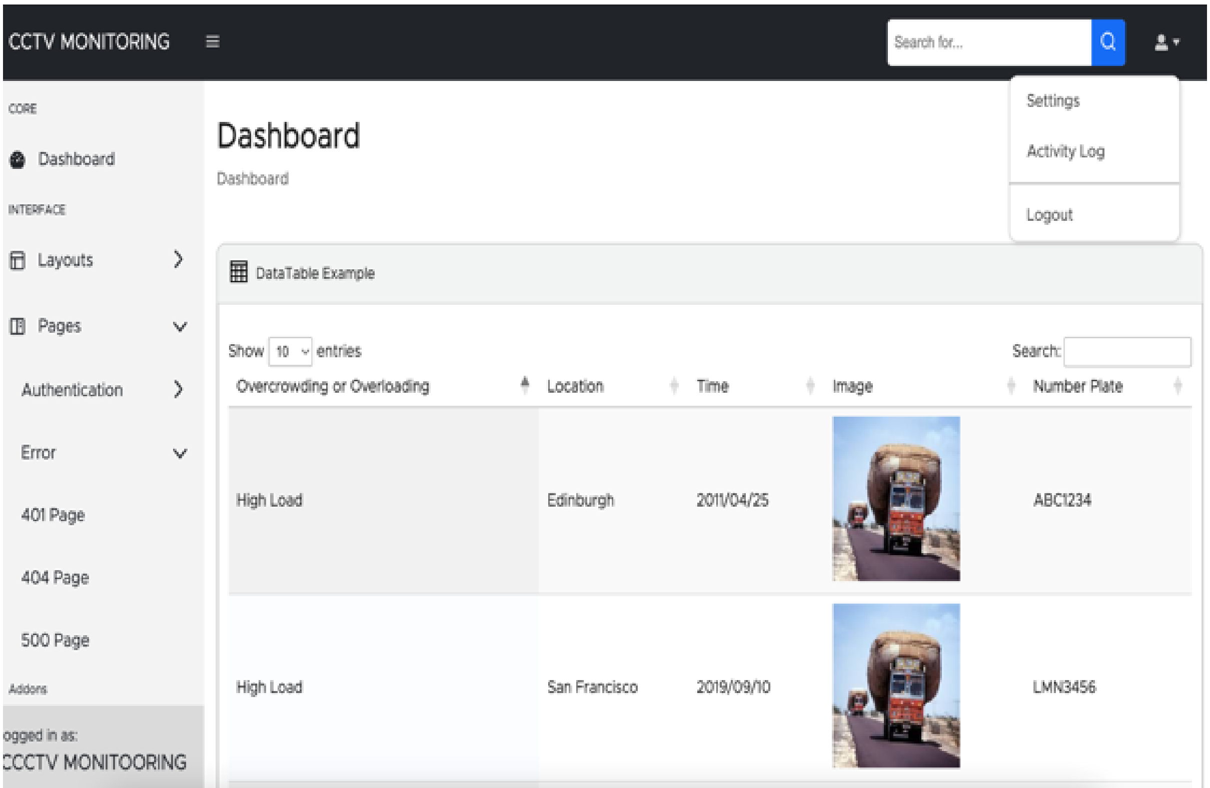

Figure 8: Dashboard

5 Project Timeline

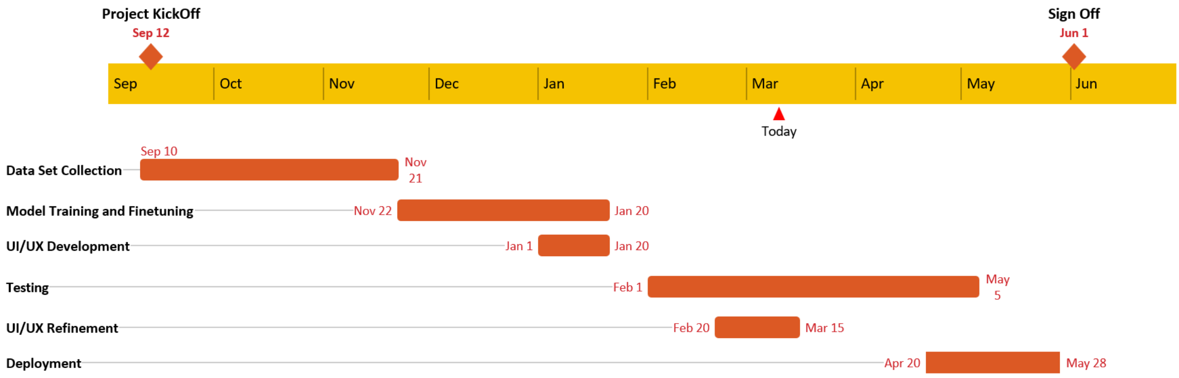

Figure 9: Gantt Chart Timeline

The project development is divided into two semesters FYP 1 and FYP 2

5.1 FYP 1

5.1.1 Task 0 - Initial Development Phase

Data Set Collection: Gathering and preprocessing relevant data for the project, ensuring its quality and completeness for further development.

5.1.2 Task 1 - Model Training and Finetuning

Training the Model: Implementing machine learning techniques to fine-tune the model, optimizing for accuracy and efficiency.

5.1.3 Task 2 - UI/UX Development

User Interface Design: Developing the user interface, focusing on a seamless user experience with an intuitive design.

5.2 FYP 2

5.2.1 Task 0 - Testing

System Testing: Conducting rigorous tests to identify and resolve bugs, ensuring the stability and reliability of the system.

5.2.2 Task 1 - UI/UX Refinement

Enhancing User Experience: Making improvements in the UI based on testing feedback, optimizing design and usability.

5.2.3 Task 2 - Deployment

Final Deployment: Launching the system for practical use, ensuring all components work as intended and meet project objectives.

5.2.4 Task 3 - Documentation

Final Report: Finalizing project documentation and compiling detailed reports summarizing the development process and outcomes.

6 Iteration 1

The first iteration is expected to be completed by the midterm of FYP-1. Iteration 1 of this project consists of two tasks: Task 0 of FYP-1, i.e., Data Set Collection. Working on Task 1, i.e., Model Training and Finetuning, will be in progress during Iteration 1.

6.1 Domain Model/ Class Diagram

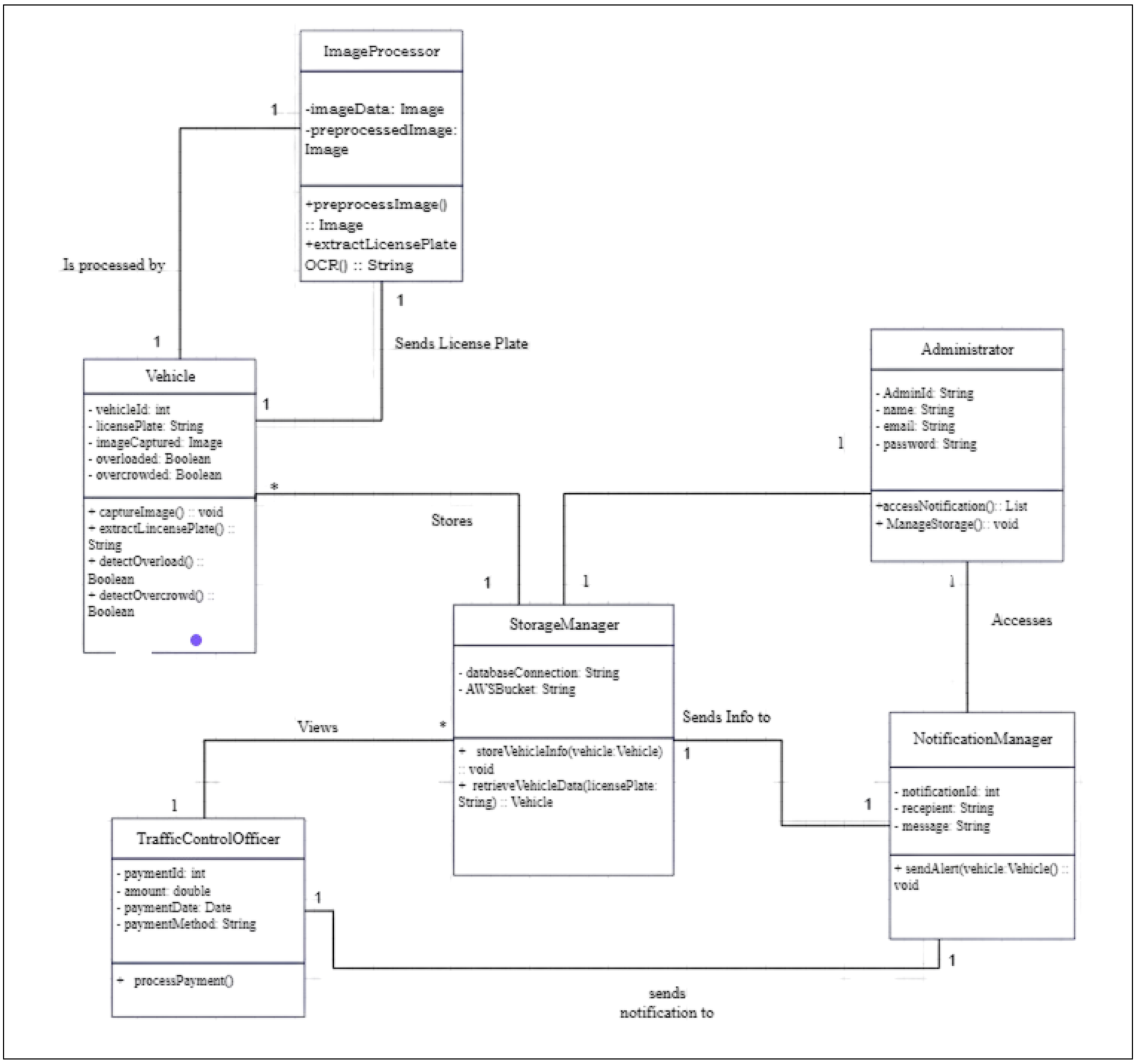

Figure 10: Class Diagram

6.2 Activity Diagram

This Activity diagram is designed to show the operational flow of the system clearly, focusing on how inputs are handled and leading to different results. It acts as a visual tool for developers and stakeholders to understand how the system works and how its parts depend on each other.

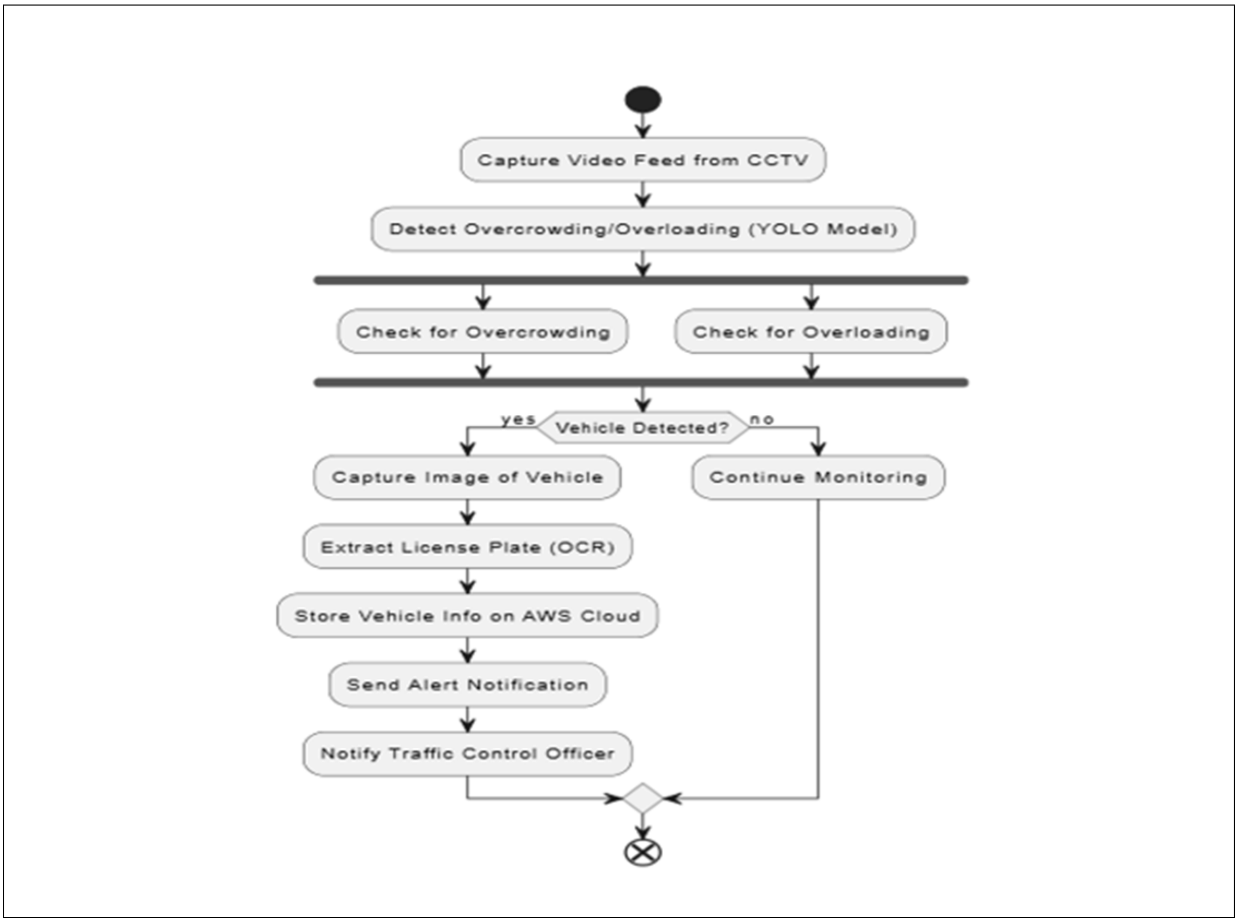

Figure 11: Activity Diagram

6.3 Component Diagram

This Component diagram is designed to show the components of the system clearly, focusing on how user interacts with these components and what is the relationship between them. It acts as a visual tool for developers and stakeholders to understand how the system works and how its parts depend on each other.

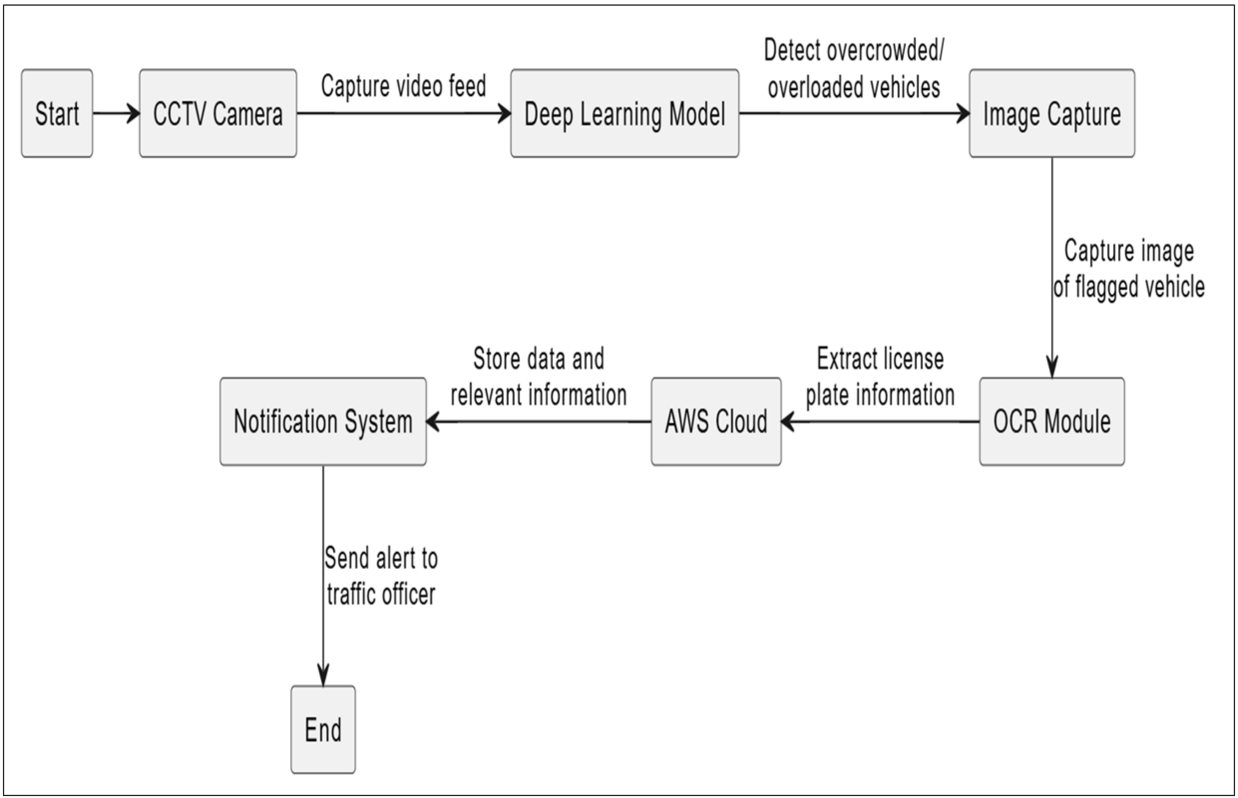

Figure 12: Component Diagram

6.4 Data Flow Diagram

This Data Flow Diagram (DFD) provides a structured representation of the system's data flow, illustrating how information moves between external entities, processes, and data stores. It serves as a crucial tool for understanding data processing within the system, ensuring clarity in system design and functionality. By mapping out data interactions, the DFD helps developers and stakeholders visualize how different components communicate and how data is transformed throughout the system.

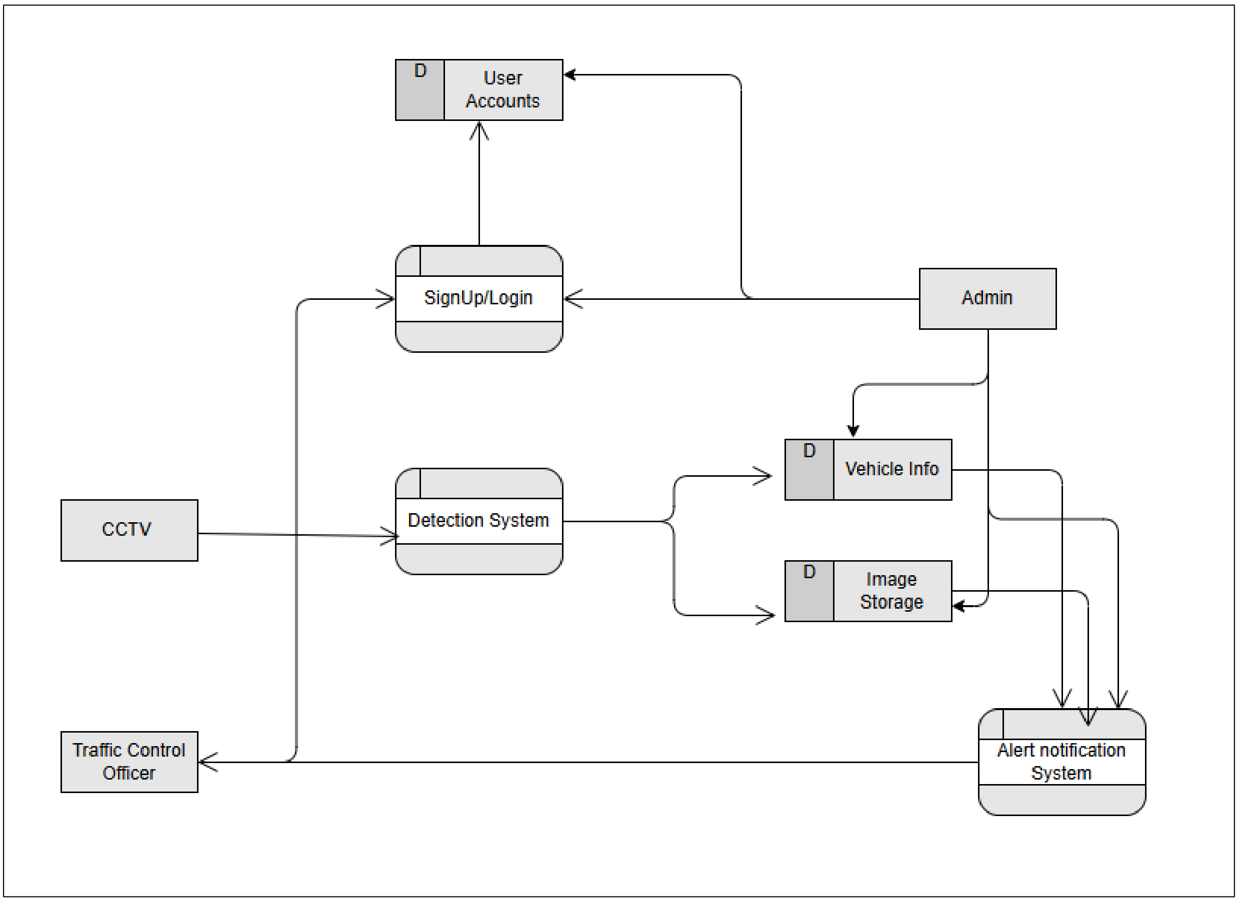

Figure 13: Data Flow Diagram

7 Iteration 2

The second iteration is expected to be completed by the end of FYP-1. Iteration 2 of this project consists of two tasks: Task 1 of FYP-1, i.e., Model Training and Finetuning. Working on Task 2, i.e., UI/UX Development, will be in progress during Iteration 2.

7.1 Model Training/Finetuning

In this phase of the project, the model will be fine-tuned to improve its accuracy in detecting and analyzing vehicle-related data from CCTV footage. The fine-tuning process involves adjusting hyperparameters, training on additional datasets, and optimizing the model for real-world scenarios. This will ensure that the system effectively identifies overloaded or overcrowded vehicles while minimizing false positives. The refined model will then be integrated into the detection system for testing and validation.

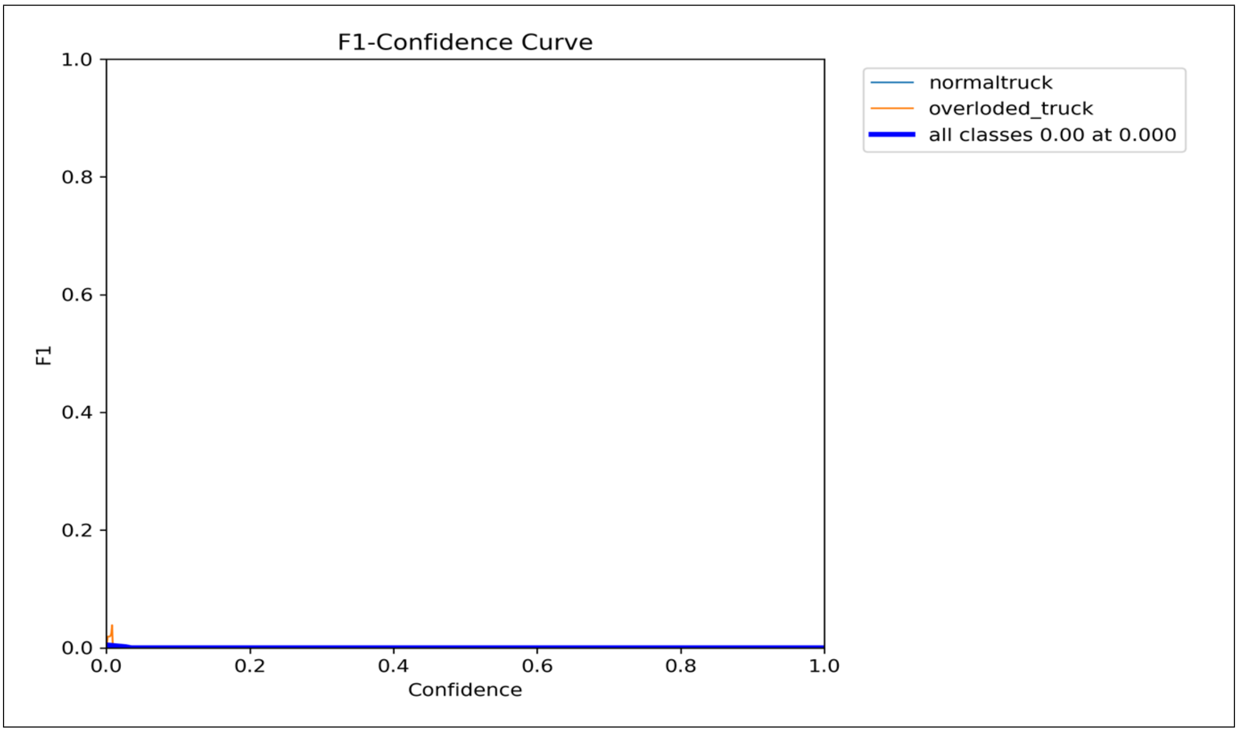

Figure 14: F1 Score

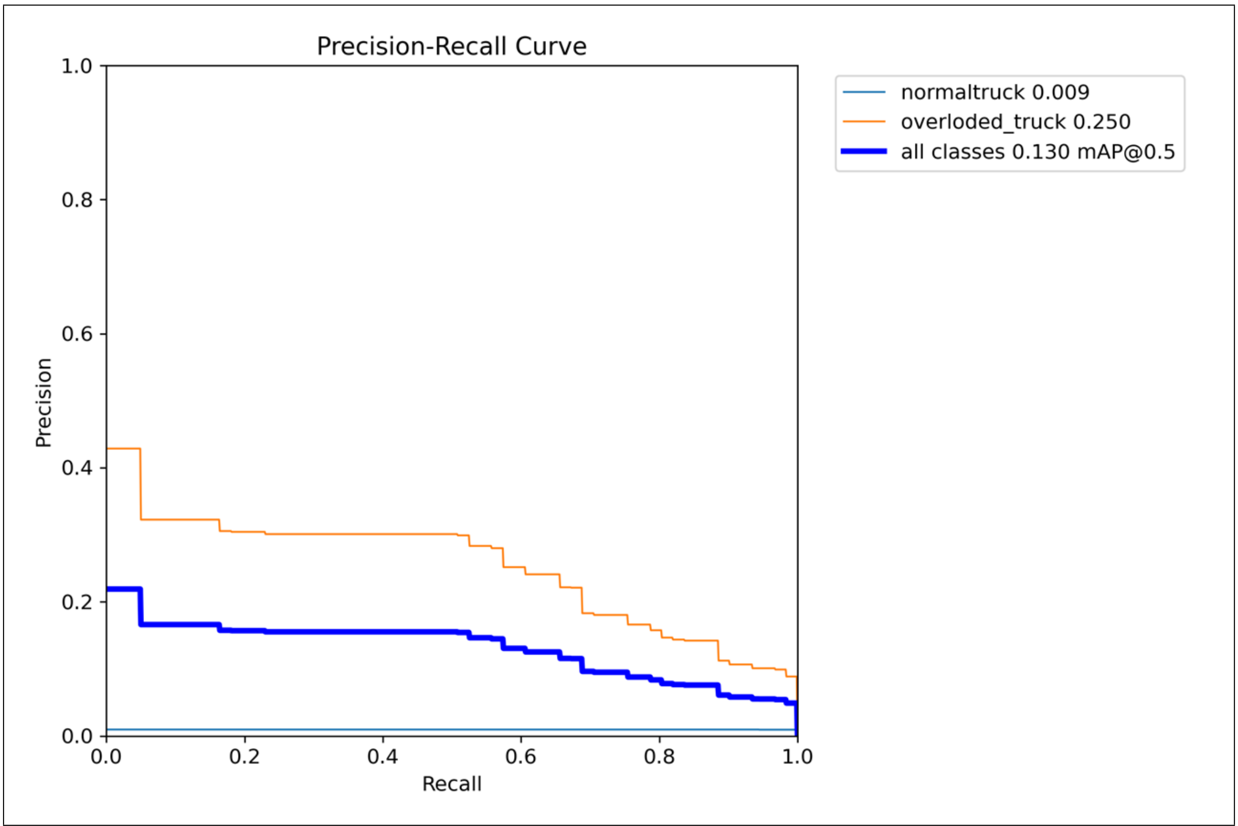

Figure 15: Precision-Recall Curve

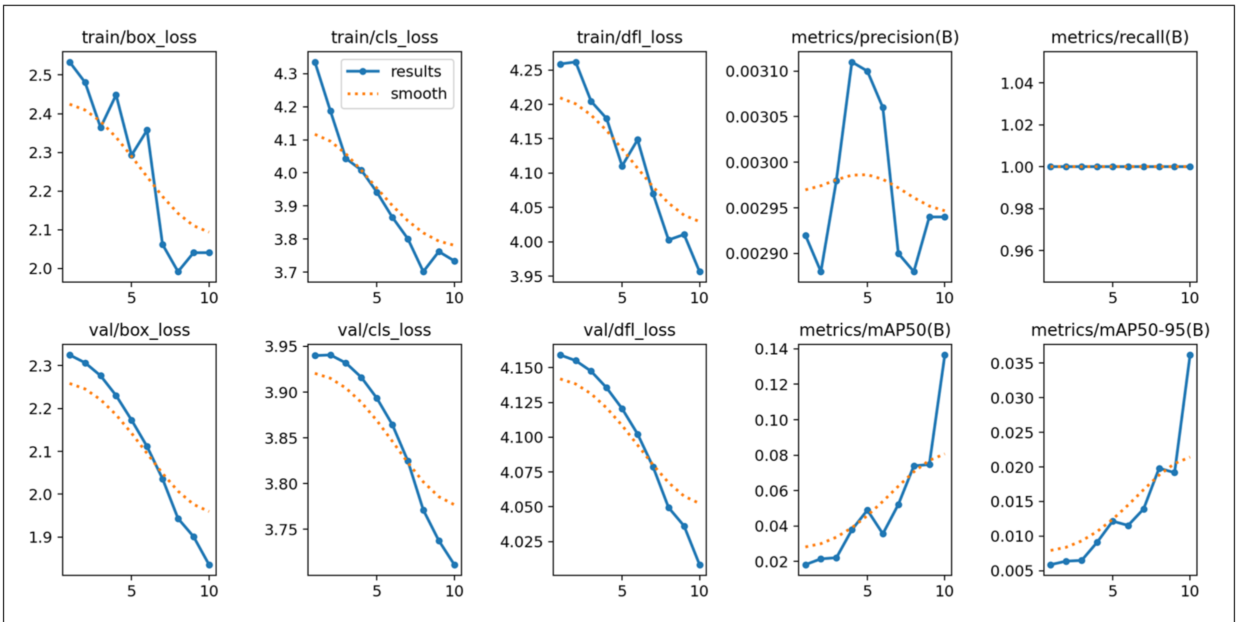

Figure 16: Finetuning Results

7.2 UI/UX Development

7.3 Code Comments

Comments are used to explain different complex secitons of code for ensuring readability and for future maintenance. So, comprehensive and consistent approach was used for commenting throughout the system's code.

Guideline	**Description**
Use Clear and Concise Comments	Comments should be brief and focused, explaining the intent or purpose of the code block without stating the obvious.
Comment the Why, Not the What	Comments should explain why something is done a certain way or why a particular approach was chosen, not just restate what the code does.
Follow PEP 257 Docstring Conventions	Use docstrings to describe modules, functions, classes, and methods. Docstrings should be enclosed in triple quotes and follow the PEP 257 conventions.
Comment Formatting	Use proper formatting and indentation for comments to improve readability. Avoid long lines of comments that can be difficult to read.
Update and Maintain Comments	Keep comments up to date. When you modify code, ensure that associated comments are modified to reflect the changes.
Avoid Redundant Comments	Don't write comments that duplicate the code. Comments should provide insights that aren't immediately obvious from the code itself.
Use Inline Comments Sparingly	Inline comments should be used sparingly and explain complex or non-intuitive code segments. Overusing inline comments can clutter the code.
Avoid Excessive Comments	Well-written code should be self-explanatory, reducing the need for excessive comments.

Table 9: Guidelines for Effective Code Comments

7.4 Naming Conventions

Element	**Convention**
Variables, Functions, Methods	`snake_case`
Constants	`ALL_CAPS`
Classes	`CapWords`
Modules and Packages	`lower_case`
Private Variables/Methods	`_single_underscore`
Protected Members	`__double_underscore`

Table 10: Python Naming Conventions

8 Iteration 3

The third iteration is expected to be completed by the mid of FYP-2. Iteration 3 of this project consists of two tasks: Task 0 of FYP-2, i.e., Testing. Working on Task 1, i.e., UI/UX Refinement, will be in progress during Iteration 3.

8.1 Testing

Testing ensures that the system functions as expected by validating its components and overall behavior. It involves systematically evaluating the system for correctness, reliability, and performance. By conducting thorough testing, potential errors and inconsistencies can be identified and resolved before deployment.

8.1.1 Unit Testing

Test Case ID	**Component**	**Test Scenario**	**Input**	**Expected Output**	**Status**
UT-01	User Authentication	Verify user login with valid credentials	Valid username & password	Successful login	Pass/Fail
UT-02	User Authentication	Verify login with incorrect credentials	Invalid username/password	Login failure message	Pass/Fail
UT-03	Detection System	Detects vehicle from CCTV input	Image of vehicle	Correct detection of vehicle	Pass/Fail
UT-04	Detection System	Detects unauthorized vehicle	Image of unauthorized vehicle	Unauthorized vehicle alert	Pass/Fail
UT-05	Image Storage	Save captured images correctly	Image file	Image stored successfully	Pass/Fail
UT-06	Alert Notification	Sends alert to Traffic Officer	Unauthorized vehicle detected	Alert sent to Traffic Officer	Pass/Fail
UT-07	Database	Store vehicle information correctly	Vehicle info	Vehicle info stored successfully	Pass/Fail

Table 11: Unit Testing Test Cases

8.1.2 Integration Testing

Test Case ID	Modules Integrated	Test Scenario	Input	Expected Output	Status
IT-01	User Authentication & Database	Verify user login integration	Valid username & password	User is authenticated and logged in	Pass/Fail
IT-02	CCTV & Detection System	Check if CCTV feeds are processed	Video feed	Vehicle images extracted for detection	Pass/Fail
IT-03	Detection System & Image Storage	Ensure detected vehicle images are stored	Vehicle image	Image is saved in the database	Pass/Fail
IT-04	Detection System & Alert System	Verify alert generation for unauthorized vehicles	Unauthorized vehicle detected	Alert sent to notification system	Pass/Fail
IT-05	Notification System & Traffic Officer	Check if traffic officer receives alerts	Unauthorized vehicle detected	Traffic officer receives an alert	Pass/Fail
IT-06	Vehicle Info Database & Detection System	Ensure vehicle database is used for detection	Vehicle license plate	Correct identification of vehicle status	Pass/Fail

Table 12: Integration Testing Test Cases

8.2 UI/UX Refinement

9 Conclusions and Future Work

The system demonstrates the potential to automate the detection of overcrowded and overloaded vehicles, significantly improving road safety. Future work includes expanding the system's capabilities to detect other traffic violations and enhancing its adaptability to diverse environments.