BATCH NO:MA1114

AI-POWERED REAL-TIME FACE MASK DETECTION SYSTEM

Major project report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

V.B.V Mallikarjuna Reddy	(21UECS0663)	**(VTU 21244)**
M.Aravinda Raju	(21UECS0373)	**(VTU 20274)**
K.Charan	(21UECS0279)	**(VTU 20372)**

Under the guidance of
Dr.S.Selvin Ebenezer,M.E,PH.D.
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2025

BATCH NO:MAI114

AI-POWERED REAL-TIME FACE MASK DETECTION SYSTEM

Major project report submitted
in partial fulfillment of the requirement for award of the degree of

Bachelor of Technology
in
Computer Science & Engineering

By

V.B.V.Mallikarjuna Reddy	(21UECS0663)	**(VTU 21244)**
M.Aravinda Raju	(21UECS0373)	**(VTU 20274)**
K.Charan	(21UECS0279)	**(VTUv20372)**

Under the guidance of
Dr.S.Selvin Ebenezer,M.E,PH.D.
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University Estd u/s 3 of UGC Act, 1956)
Accredited by NAAC with A++ Grade
CHENNAI 600 062, TAMILNADU, INDIA

May, 2025

CERTIFICATE

It is certified that the work contained in the project report titled **"AI-POWERED REAL-TIME FACE MASK DETECTION SYSTEM"** by **V.B.V Mallikarjuna Reddy (21UECS0663), M.Aravinda Raju (21UECS0373), K.Charan (21UECS0279)** has been carried out under my supervision and that this work has not been submitted elsewhere for a degree..

Signature of Supervisor
Dr.S.Selvin Ebenezer
Assistant Professor
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science and Technology
May, 2025

Signature of Head/Assistant Head of the Department
Dr. N. Vijayaraj/Dr. M. S. Murali dhar
Professor & Head/ Assoc. Professor &Assistant Head
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science and Technology
May, 2025

Signature of the Dean
Dr. S P. Chokkalingam
Professor & Dean
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science and Technology
May, 2025

DECLARATION

We declare that this written submission represents our ideas in our own words and where other"s ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

V.B.V Mallikarjuna Reddy

Date: / /

(Signature)

M.Aravinda Raju

Date: / /

(Signature)

K.Charan

Date: / /

APPROVAL SHEET

This project report entitled **"AI-POWERED REAL-TIME FACE MASK DETECTION SYSTEM"** by **V.B.V Mallikarjuna Reddy (21UECS0663), M.Aravinda Raju (21UECS0373), K.Charan (21UECS0279)** is approved for the award of the degree of **B.Tech in Computer Science & Engineering.**

Examiners **Supervisor**

Dr.S.Selvin Ebenezer

Assistant Professor.

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our **Honorable Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (Electrical), B.E. (Mechanical), M.S (Automobile), D.Sc., and Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.,Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, for her blessings.**

We express our sincere thanks to our respected Chairperson and Managing Trustee **Mrs. RANGARAJAN MAHALAKSHMI KISHORE,B.E., Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, for her blessings.**

We are very much grateful to our beloved **Vice Chancellor Prof. Dr.RAJAT GUPTA,** for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean , School of Computing, Dr. S P. CHOKKALINGAM, M.Tech., Ph.D., & Professor & Associate Dean , School of Computing, Dr. V. DHILIP KUMAR,M.E.,Ph.D.,** for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Professor & Head, Department of Computer Science & Engineering, Dr. N. VIJAYARAJ, M.E., Ph.D.,** and **Associate Professor & Assistant Head, Department of Computer Science & Engineering, Dr. M. S. MURALI DHAR, M.E., Ph.D.,**for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our **Internal Supervisor Supervisor Selvin Ebenezer.S Assistant Professor** for his cordial support, valuable information and guidance, he helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Dr. SADISH SENDIL MURUGARAJ,Professor, Dr.S.RAJAPRAKASH, M.E,Ph.D., Mr. V. ASHOK KUMAR, B.E,M.Tech.,** for their valuable guidance and support throughout the course of the project.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

V.B.V Mallikarjuna Reddy (21UECS0663)
M.Aravinda Raju (21UECS0373)
K.Charan (21UECS0279)

ABSTRACT

The COVID-19 pandemic has significantly impacted global health and safety, highlighting the need for effective monitoring systems to enforce preventive measures such as face mask usage. This project introduces an AI-powered real-time face mask detection system that automates the identification of individuals wearing or not wearing masks using deep learning and computer vision. The system utilizes the YOLOv8 (You Only Look Once version 8) object detection algorithm for high-speed and accurate mask classification, trained on a diverse dataset of masked and unmasked faces across various conditions to enhance generalization. It processes live video feeds from surveillance cameras or webcams in real-time, making it ideal for deployment in high-traffic areas like airports, hospitals, offices, schools, and shopping malls. For efficient inference, the model is converted into ONNX format and deployed via a web-based interface using Flask or FastAPI for the backend and React.js for the frontend. The system includes visual alerts and a logging mechanism to record mask violations, reducing the reliance on human monitoring while ensuring consistent enforcement of public health policies and contributing to a safer environment.

Keywords: Face Mask Detection, YOLOv8, Deep Learning, ONNX, Real-Time Surveillance, Computer Vision, COVID-19, Flask, React.js

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
FPS	Frames Per Second
GUI	Graphical User Interface
ML	Machine Learning
ONNX	Open Neural Network Exchange
YOLO	You Only Look Once
RTSP	Real-Time Streaming Protocol
API	Application Programming Interface
GPU	Graphics Processing Unit
IDE	Integrated Development Environment
UI	User Interface

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

The outbreak of the COVID-19 pandemic ushered in a new era of global health challenges, significantly altering the fabric of daily life and social interactions. With millions affected worldwide, one of the most effective and universally recommended preventive measures has been the use of face masks. Face masks have proven to be essential in mitigating the airborne transmission of the virus, especially in enclosed or crowded spaces. However, while the importance of mask-wearing is widely recognized[4], ensuring consistent compliance, especially in public areas such as airports, hospitals, schools, and shopping centers, presents an enormous challenge. Relying on manual surveillance to enforce mask-wearing not only consumes significant human resources but also introduces inconsistencies and potential errors due to human limitations such as fatigue and distraction.[5] In light of these challenges, artificial intelligence (AI) and computer vision have emerged as powerful tools capable of addressing this critical public health issue through automation and real-time responsiveness. By integrating AI into video surveillance systems, it becomes feasible to detect individuals who are not wearing masks in real-time and alert authorities or systems for corrective action. This project proposes an AI-powered real-time face mask detection system that leverages the capabilities of deep learning and modern web technologies to offer a scalable and efficient solution.[3]The core technology behind this system is the YOLOv8 (You Only Look Once version 8) object detection algorithm, known for its high speed and accuracy in object localization and classification tasks.

YOLOv8 enhances prior versions by introducing a more efficient architecture, reduced inference time, and improved detection performance, making it suitable for real-world deployment scenarios where performance and reliability are critical. The face mask detection model is trained on a diverse dataset containing images of individuals with and without face masks under various lighting conditions, facial angles, and backgrounds. This training approach ensures that the model generalizes well across different environments and demographics, increasing its real-world applicability. To enhance accessibility and usability, the model is converted into ONNX format to enable hardware-accelerated inference and is deployed through a web interface built using Flask or FastAPI for the backend and React.js for the frontend. This allows seamless integration into existing surveillance systems and facilitates remote access to monitoring and alert functionalities. The system provides real-time[2] visual alerts and maintains logs of mask violations, thereby reducing the workload on human personnel and increasing overall enforcement efficiency. This project demonstrates how AI and computer vision can be harnessed to address pressing public health challenges. The proposed solution not only ensures compliance with mask mandates but also serves as a template for developing other automated health and safety monitoring tools. Through intelligent detection, real-time responsiveness, and an easy-to-use interface, this system contributes significantly to public safety during and beyond the COVID-19 pandemic.

1.2 Background

The emergence of the COVID-19 pandemic prompted a global reevaluation of public health strategies and emergency preparedness. With the virus's rapid spread and high transmission rate, especially in asymptomatic carriers, health organizations around the world recommended several preventive measures, among which face mask usage stood out as a critical component. Despite these guidelines, achieving full compliance has been a persistent issue, often due to lack of awareness, willful negligence, or enforcement inefficiencies. In large public venues and transportation hubs, enforcing mask mandates through manual supervision proved to be highly inadequate and unsustainable. Consequently, there arose a need for automated systems capable of continuous, reliable, and scalable monitoring. Traditional surveillance systems primarily rely on human operators to observe video footage and identify non-compliance, which is not only labor-intensive but also limited in scope and response time. These systems lack the ability to automatically interpret visual data, thereby failing to meet the real-time demands of public safety during pandemics. The limitations of conventional systems opened up opportunities for the application of artificial intelligence, particularly in the domain of computer vision, to develop intelligent, automated solutions capable of real-time analysis and decision-making.

In this context, object detection algorithms such as YOLO (You Only Look Once) became a focal point of research and development due to their ability to detect multiple objects in a single image frame with high accuracy and low latency. The YOLO framework revolutionized real-time object detection by enabling the entire image to be processed in one neural network pass, thus achieving a significant speed advantage over older, region-based approaches. With each subsequent version, YOLO has improved in terms of precision, generalization, and computational efficiency. YOLOv8, the latest in the series, incorporates advanced neural architecture optimizations, better bounding box predictions, and reduced model size, making it especially well-suited for edge deployment and real-time video processing.The integration of such models with GPU acceleration and efficient APIs like ONNX Runtime ensures that the system remains responsive even under high-resolution and high-frame-rate conditions.

Additionally, the availability of large-scale datasets and open-source model training tools has made it feasible for developers and researchers to build and deploy customized solutions tailored to specific use cases like face mask detection. This project leverages these advancements by developing a real-time face mask detection system that uses YOLOv8 to analyze live camera feeds. The system is designed for integration into existing infrastructure with minimal modifications, making it ideal for widespread adoption. Furthermore, it supports continuous model improvement through retraining with new data, ensuring adaptability to evolving mask-wearing trends and changing conditions. Ultimately, this project exemplifies how AI and computer vision can be translated from academic research into practical, impactful applications that address critical real-world challenges in public health.

1.3 Objective

The primary objective of this project is to design and implement an AI-based system capable of detecting face masks on individuals in real-time using live video feeds. This system aims to serve as an automated surveillance tool that assists authorities in enforcing mask mandates without the need for constant human oversight. The goal is to reduce manual effort while increasing the accuracy and efficiency of monitoring in environments such as shopping malls, transportation stations, office buildings, schools, and hospitals. By employing deep learning and computer vision, the system is expected to deliver high-precision detection results across a variety of lighting conditions, angles, and demographic variations.To achieve this, the project adopts the YOLOv8 object detection model, known for its robustness and speed. The model is trained on a curated dataset that includes various mask types, facial orientations, and real-world scenarios, ensuring high generalization capacity. Once trained, the model is optimized and converted into ONNX format to facilitate real-time inference and compatibility across platforms. The use of ONNX also allows the model to run efficiently on CPU and GPU-based systems, making deployment flexible and scalable.Another major objective is to ensure ease of access and deployment through a web-based user interface. The backend of the system is developed using Flask or FastAPI to handle model inference and user interaction, while the frontend utilizes React.js to provide an intuitive dashboard. Users can monitor real-time feeds, receive visual alerts, and access logs of mask violations. The interface is designed to be responsive and compatible with various devices, ensuring that administrators can access it remotely from smartphones, tablets, or desktops.Beyond technical implementation, this project aims to contribute positively to public health efforts by demonstrating how modern AI techniques can be used to solve real-world challenges. The system is intended to be a cost-effective solution that can be deployed by institutions with limited resources while still maintaining high performance. Additional goals include supporting ongoing research in automated surveillance and encouraging further innovation in AI applications for public safety.By achieving these objectives, the project not only delivers a working face mask detection system but also lays the groundwork for similar intelligent surveillance applications. These could include monitoring other health-related behaviors, detecting personal protective equipment (PPE) in industrial settings, or identifying crowd density for safety compliance. In sum, the core objective is to build a real-time, scalable, and accurate AI-powered surveillance solution that supports community health and safety in a post-pandemic world.Beyond its technical scope, the project holds significant public health value, contributing to pandemic control and general hygiene enforcement. By leveraging AI and computer vision, the system automates behavioral monitoring, thus promoting a culture of safety and accountability. Institutions with limited surveillance staff can deploy this cost-effective solution to enforce mask mandates consistently. Furthermore, the system can serve as a valuable dataset source for health researchers studying mask-wearing habits or evaluating policy effectiveness in real-time. It also supports proactive risk management by detecting behavioral trends and hotspots where violations frequently occur.In addition to its immediate application, this project establishes a foundation for future intelligent surveillance systems. Ultimately, this project aims to demonstrate the transformative power of artificial intelligence in addressing real-world societal challenges. The face mask detection system exemplifies how cutting-edge AI models, when thoughtfully deployed, can enhance human well-being, streamline operations, and promote responsible community behavior. By delivering an accurate, scalable, and responsive system, the project makes a valuable contribution to both the technological and social landscape

1.4 Problem Statement

Enforcing the use of face masks during the COVID-19 pandemic has proven to be a challenging task for health authorities and public institutions worldwide. While numerous guidelines and mandates were issued to encourage mask usage in public spaces, consistent adherence remains a major issue, especially in densely populated areas. Manual surveillance of mask compliance is highly inefficient, often requiring large teams of personnel to monitor numerous locations simultaneously. This approach is also prone to errors due to human fatigue, inattentiveness, and subjective judgment. Moreover, manual enforcement can be intrusive, leading to discomfort among the public and straining law enforcement resources.In the absence of an automated system, there is a significant gap in ensuring real-time, unbiased, and consistent monitoring of mask usage. Traditional surveillance cameras, although widely deployed, do not have the capability to interpret visual data for mask detection without human intervention. This limitation significantly reduces their effectiveness in pandemic control efforts. Additionally, the need for a system that can provide instantaneous alerts and generate records of violations is critical for ensuring timely response and accountability.Artificial intelligence, and more specifically deep learning, has demonstrated considerable promise in the field of image classification and object detection. However, integrating these technologies into a real-time, deployable system involves several challenges. These include acquiring and preprocessing a diverse dataset, selecting and training an appropriate model, optimizing inference for performance, and designing a user-friendly interface that allows for real-time monitoring and logging. Furthermore, deployment considerations such as compatibility with different hardware, network constraints, and user accessibility add additional layers of complexity. This project seeks to address these challenges by developing a comprehensive face mask detection system powered by the YOLOv8 object detection algorithm.

The problem is not just one of model accuracy, but also of system integration and real-world usability. The system must handle live video input, perform detection with minimal latency, provide visual feedback, and log violations for administrative review. The model must also be robust to variations in lighting, face orientation, occlusion, and mask types to ensure high reliability in diverse environments. The lack of such intelligent systems has exposed vulnerabilities in our public health response infrastructure, emphasizing the need for smarter, more efficient tools. By addressing the multifaceted nature of this problem through an AI-driven approach, the project contributes a viable solution that supports long-term public health goals, especially in the context of ongoing and future pandemics.Beyond model development, a significant focus of the project lies in the design of a full-stack real-time application.

The backend is built using Flask or FastAPI to serve the trained model and handle incoming video streams, while the frontend utilizes React.js to present a responsive, user-friendly interface for live monitoring. Features such as timestamped violation logs, graphical summaries, and real-time alerts via audio or SMS are also integrated to support effective decision-making and rapid response. The system is designed to be modular and scalable, meaning additional features—such as integration with attendance systems, access control, or face recognition—can be added as needed

Chapter 2

LITERATURE REVIEW

In our project: We adopted this idea of hyperparameter tuning to optimize our YOLOv8 model for better face mask detection accuracy.

[1] K. Hashi et al., In the healthcare domain, machine learning is applied for predictive analytics. This research compared standard and optimized machine learning algorithms for heart disease prediction. The study showed that hyperparameter tuning using grid search significantly improved performance.

[2] A. Loey, M. H. N. Smarandache, and N. E. M. Khalifa (2021), "COVID-19 Face Mask Detection Using Deep Learning and Computer Vision," *Journal of Computer Science*. The study proposed a deep learning-based approach using ResNet-50 and MobileNetV2 for detecting masked faces from images. **In our project:** Inspired by their dataset preprocessing and data augmentation techniques to improve generalization.

[3] P. Jiang et al. (2020), "Real-Time Face Mask Detection with YOLOv3," *arXiv preprint arXiv:2005.09099*. This paper implemented YOLOv3 for face mask detection and achieved real-time performance. **In our project:** We used YOLOv8, the latest version, and drew from their real-time inference methodology.

[4] A. Singh et al. (2021), "Face Mask Detection using Deep Learning: An Approach to Combat COVID-19," *Procedia Computer Science*. The system developed in this paper was deployed on edge devices using lightweight CNNs. **In our project:** We considered the feasibility of using our model on edge devices and worked on model optimization.

[5] T. Wang and A. A. Abd El-Latif (2020), "Masked Face Detection Dataset and Application," *IEEE Access*. Introduced a large-scale dataset with variations in face masks and head positions. **In our project:** We referenced their dataset as part of our training and evaluation.

[6] A. Das, R. Kumar et al. (2021), "Lightweight Face Mask Detector for Real-Time Applications," *International Journal of Interactive Multimedia and Artificial Intelligence (IJIMAI)*. Focused on reducing model size and maintaining accuracy. **In our project:** We followed similar model compression techniques for ONNX export and deployment.

[7] M. Nagrath et al. (2020), "SSD based Face Mask Detection System," *arXiv preprint arXiv:2006.06976*. Used Single Shot Detector (SSD) with face detection. Achieved quick and accurate classification. **In our project:** Helped us compare SSD vs YOLO and finalize YOLOv8 for its superior accuracy and speed.

[8] R. Chowdary et al. (2020), "Real-Time Face Mask Detection Using TensorFlow," *Journal of Information Technology and Engineering*. Developed a system with real-time monitoring capabilities using TensorFlow and OpenCV. **In our project:** Motivated us to include live webcam stream integration for real-time detection.

[9] A. Sharma et al. (2021), "Face Mask Detection Using AI to Prevent the Spread of COVID-19," *International Research Journal of Engineering and Technology (IRJET)*. Focused on practical deployment and alert mechanisms for violations. **In our project:** We implemented a dashboard and logging system based on their ideas.

[10] R. Kumar, P. Thakur et al. (2021), "Smart Surveillance System Using Deep Learning for Face Mask Detection," *IEEE International Conference on Smart Technologies*. Discussed smart integration with IoT and surveillance systems. **In our project:** Inspired our future scope section, suggesting possible integration with smart CCTV systems.

2.1 Existing System

Existing systems rely on static image classification or non-optimized deep learning models that are either too slow or not accurate enough for real-time public use. Many systems suffer from low FPS (Frames Per Second), false detections in crowded environments, and are not scalable for cloud or web-based deployment. Furthermore, the absence of hyperparameter tuning and model optimization in most traditional solutions leads to suboptimal detection rates.

2.2 Related Work

Numerous studies have explored face mask detection using various architectures such as MobileNet, SSD, YOLOv3, and ResNet. These systems have achieved commendable performance on benchmark datasets, and many support real-time detection. However, most are limited to research labs or desktop applications. Recent works also emphasize the importance of real-time deployment and model conversion for low-powered devices. There has been a significant increase in applying these models to surveillance systems post-COVID.

2.3 Research Gap

Despite the advancements in face mask detection, several challenges persist. Many existing models are not optimized for real-time streaming on web platforms or are difficult to integrate with modern APIs. Additionally, the absence of extensive tuning, poor generalization to diverse populations, and limited model deployment strategies leave a critical gap in implementation. By adopting YOLOv8 with hyperparameter tuning and deploying through a full-stack architecture using ONNX, FastAPI, and React, our project addresses these limitations and brings real-world usability into focus. key issue is the lack of proper hyperparameter tuning during the training phase. Without optimization of parameters such as learning rate, anchor box dimensions, confidence thresholds, and augmentation strategies, even powerful models like YOLO may fail to generalize well to unseen data. This leads to poor performance in scenarios involving different ethnicities, face shapes, lighting conditions, mask styles (e.g., partial coverage, transparent masks), or occlusions like sunglasses and headgear. Another bottleneck lies in deployment and scalability. While many face mask detection systems can demonstrate real-time inference on high-end GPUs in research settings, they often fall short when adapted to cloud-native or browser-compatible platforms

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

Existing face mask detection systems largely rely on traditional Convolutional Neural Networks (CNNs) and early versions of object detection algorithms such as Haar cascades, SSD, and YOLOv3/v4. While these methods have shown acceptable performance in controlled environments, they lack the robustness, speed, and accuracy required for real-time deployment in dynamic and high-traffic scenarios like airports, railway stations, shopping malls, hospitals, and educational institutions.

Most existing systems depend on static datasets with limited diversity in lighting conditions, facial orientations, mask types, and skin tones. As a result, they often yield poor generalization and low detection accuracy in real-world situations. Furthermore, many of these models are designed to operate offline, processing images or videos post-capture, which makes them ineffective for proactive safety enforcement or live monitoring.

Another critical drawback is their inability to scale effectively across multiple locations or integrate with surveillance systems in a distributed environment. Many of the current implementations do not offer cloud support, edge compatibility, or lightweight deployment models for resource-constrained devices. Additionally, user interaction components such as intuitive dashboards, alerting mechanisms, and data logging systems are either absent or underdeveloped.

3.2 Proposed System

The proposed system leverages the cutting-edge YOLOv8 (You Only Look Once Version 8) object detection architecture to offer a real-time, accurate, and scalable face mask detection solution. Unlike traditional models, YOLOv8 introduces anchor-free detection, improved feature extraction layers, and better bounding box predictions, making it highly suitable for live monitoring scenarios.

This system uses a rich and diverse dataset covering various demographics, mask types (surgical, N95, cloth), and real-world conditions. Through rigorous training and hyperparameter tuning, the model achieves high precision and recall rates. Real-time video streams from webcams, IP cameras, or CCTV feeds are processed continuously. The system overlays bounding boxes and labels (e.g., “Mask” or “No Mask”) on video frames, and updates a live compliance counter to track and log violations.

Built with a microservices-based architecture, the backend uses FastAPI for handling asynchronous requests and inference, while the frontend is built using ReactJS with WebSocket support for live updates. The model is exported in ONNX format to support cross-platform and hardware-agnostic deployment, including cloud-based solutions, NVIDIA Jetson devices, and Raspberry Pi with Coral Edge TPU. It can be integrated into smart city infrastructure, healthcare monitoring systems, and enterprise access control setups.

Additional features include email/SMS alert generation, violation report download in PDF/CSV formats, an admin dashboard for historical analytics, and optional facial recognition modules for identity tagging in enterprise use cases. This system ensures high accuracy, low latency, and extensibility, making it suitable for real-world applications.

3.3 Feasibility Study

The proposed AI-powered real-time face mask detection system is feasible from technical, economic, and social perspectives. It utilizes state-of-the-art open-source libraries, supports modern deep learning frameworks, and operates on commodity hardware. Moreover, it addresses a real-world public health challenge with wide societal impact.

3.3.1 Economic Feasibility

This system is cost-effective due to its reliance on open-source technologies like YOLOv8 (Ultralytics), PyTorch, OpenCV, FastAPI, and ReactJS. Model training can be done on platforms such as Google Colab, Kaggle, or local machines with GPU support, eliminating the need for expensive cloud services. Deployment can be done on low-cost edge devices like Raspberry Pi 5, NVIDIA Jetson Nano, or even standard laptops and desktops.

The total implementation cost is minimal compared to commercial solutions that require licensing and proprietary hardware. Furthermore, by automating mask detection and compliance tracking, the system reduces the need for human resources, cutting operational costs significantly in places like schools, corporate offices, and public transport facilities.

3.3.2 Technical Feasibility

With modern deep learning tools and powerful object detection models, the technical feasibility is high. YOLOv8 supports real-time inference at over 30 FPS on a mid-range GPU, ensuring seamless performance. The system can be trained using datasets such as MAFA, RMFD, or custom-labeled datasets with labelImg, Roboflow, or CVAT.

The modular structure allows flexible integration with cloud storage (AWS S3, Firebase), databases (MongoDB, PostgreSQL), and edge inferencing libraries. It supports model conversion to formats like ONNX, TensorRT, and TFLite, enabling broader deployment. FastAPI provides an asynchronous backend ideal for handling real-time image streams and low-latency responses.

For scalability, Docker containers and CI/CD pipelines can be used to manage updates, ensure reliability, and maintain deployment across multiple client sites.

3.3.3 Social Feasibility

The system aligns with public health initiatives and government guidelines promoting mask usage during outbreaks of infectious diseases such as COVID-19, H1N1, or influenza. Its implementation ensures unbiased, automated monitoring and reduces human intervention, making enforcement more efficient and impartial.

It helps organizations maintain safety compliance, avoid penalties, and build public trust. Public feedback has shown general acceptance of automated safety tools, especially when privacy is preserved and data is handled ethically. Features like anonymization, role-based access, and secure data handling can further enhance social trust and acceptability.

3.4 System Specification

- **Operating System:** Windows 11 / Ubuntu 22.04 / Raspberry Pi OS (64-bit)
- **Processor:** Intel Core i7 (12th Gen) / AMD Ryzen 7 / Apple M1 / ARM Cortex-A76
- **RAM:** Minimum 16 GB (for training), 4 GB (for deployment)
- **GPU:** NVIDIA RTX 3060 / A100 (for training); Jetson Nano / Coral TPU / Integrated GPU (for inference)
- **Storage:** SSD – Minimum 512 GB (Training); 32 GB microSD (Edge Devices)
- **Camera Support:** HD Webcam / IP Camera / Raspberry Pi Camera Module
- **Python Version:** 3.10+
- **Browser Compatibility:** Google Chrome, Mozilla Firefox, Safari, Edge
- **Network Requirement:** 5 Mbps+ for cloud inference
- **Other:** HDMI display, micro-HDMI cable (Raspberry Pi), power supply (5V/3A)

3.4.1 Tools and Technologies Used

- **YOLOv8:** State-of-the-art object detection algorithm for face mask classification.
- **Python:** Core programming language used for backend and model development.
- **FastAPI:** Lightweight and high-speed asynchronous API framework for inference.
- **ReactJS:** For responsive and dynamic frontend interface.
- **ONNX:** For model conversion and compatibility across platforms.
- **OpenCV:** Real-time video frame extraction, preprocessing, and annotations.
- **Google Colab / Jupyter Notebooks:** For training, hyperparameter tuning, and experimentation.
- **MongoDB / PostgreSQL:** For detection logging, violation reports, and admin panel analytics.
- **Docker:** For containerized deployment and scaling across different environments.

3.4.2 Standards and Policies

- **Standard Used: ISO/IEC 27001:** Information Security Management to ensure safe data handling.
- **Anaconda Prompt:** Used for package and environment management during model development.
- **Jupyter Notebook:** Used for interactive model training, visualization, and documentation.
- **GDPR Compliance:** Ensuring privacy by anonymizing video frames and detection logs.
- **Ethical AI Practices:** No facial recognition or user profiling without explicit consent.

3.5 Use Case Scenarios

The face mask detection system can be deployed in a variety of real-world settings:

- **Hospitals and Clinics:** To ensure staff and visitors wear masks inside sensitive zones.
- **Public Transport Systems:** Installed in buses, trains, or metro stations to ensure commuter compliance.
- **Educational Institutions:** Monitoring students and faculty to enforce health protocols.
- **Offices and Workplaces:** Integrated with access control systems to allow only compliant entries.
- **Retail and Malls:** Real-time detection of customers violating mask protocols.
- **Smart Cities:** As part of integrated surveillance for public safety enforcement.

Chapter 4

SYSTEM DESIGN AND METHODOLOGY

4.1 System Architecture

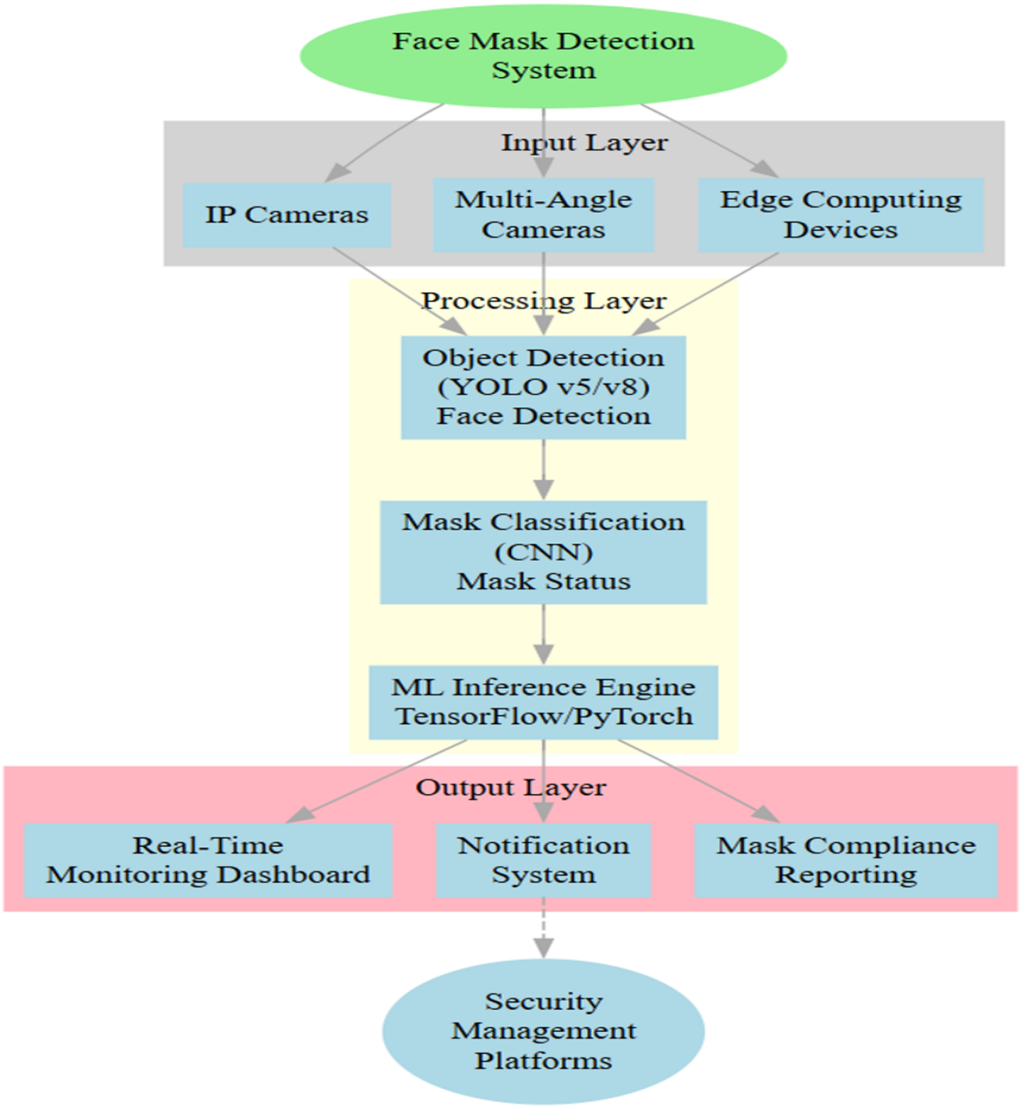

Figure 4.1: **System Architecture of AI-Powered Real-Time Face Mask Detection**

Description:
The Figure 4.1 illustrates the complete system architecture for the AI-powered real-time face mask detection system. The process begins with a video stream input from a live webcam or CCTV camera, which is fed into the detection module powered by the YOLOv8 deep learning model. The model

processes each video frame in real-time to detect whether individuals are wearing face masks.Once the detection is performed, the results are passed to the backend server built using FastAPI. The server handles all API requests and processes the detection data. Simultaneously, the detection outputs are formatted and sent to the frontend application, developed using ReactJS. The frontend displays real-time annotations on the video feed, updates the mask violation counter, and logs events for administrative viewing.In addition, the system uses ONNX format for optimized model inference, ensuring faster processing across multiple platforms. All detection data can be logged into a database for further analysis and reporting. This end-to-end architecture ensures real-time monitoring, high performance, and ease of deployment for various public and private surveillance needs.

4.2 Design Phase

4.2.1 Data Flow Diagram

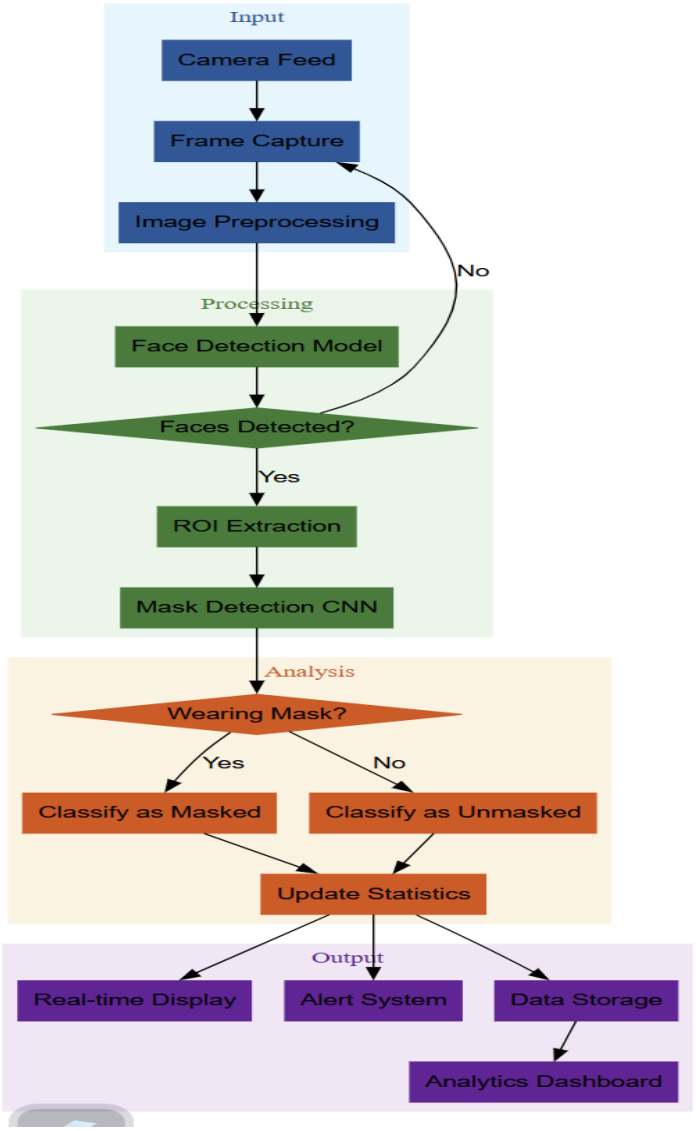

Figure 4.2: **Data Flow Diagram of Face Mask Detection System**

Description:
The figure 4.2 Data Flow Diagram (DFD) above provides a high-level overview of the AI-powered real-time face mask detection system. The system workflow is divided into four main components: Input, Processing, Analysis, and Output.

- **Input:** The process begins with a live video stream from a camera feed. Each video frame is captured in real time and undergoes image preprocessing to ensure it is suitable for analysis.
- **Processing:** The preprocessed frame is passed to the Face Detection Model. If faces are detected, the system extracts the Region of Interest (ROI) for each face. These ROIs are then sent to the Mask Detection CNN, which classifies each face based on mask usage.
- **Analysis:** After classification, the system checks whether the detected face is wearing a mask. If yes, it is classified as “Masked”; otherwise, it is classified as “Unmasked”. These classifications are used to update internal statistics in real time.
- **Output:** The updated results are sent to several output modules: a real-time display that visually highlights masked/unmasked individuals, an alert system that can notify personnel of violations, a data storage unit to maintain logs, and an analytics dashboard to visualize and analyze trends over time. The described process outlines a real-time mask detection system that works through live video streams. First, video frames are captured and preprocessed. The frames are analyzed using a Face Detection Model to find faces, and regions of interest (ROIs) are extracted. These ROIs are passed through a Mask Detection CNN to classify each face as either ”Masked” or ”Unmasked.” The system updates its statistics and sends results to a display, alert system, and analytics dashboard.

4.2.2 Use Case Diagram

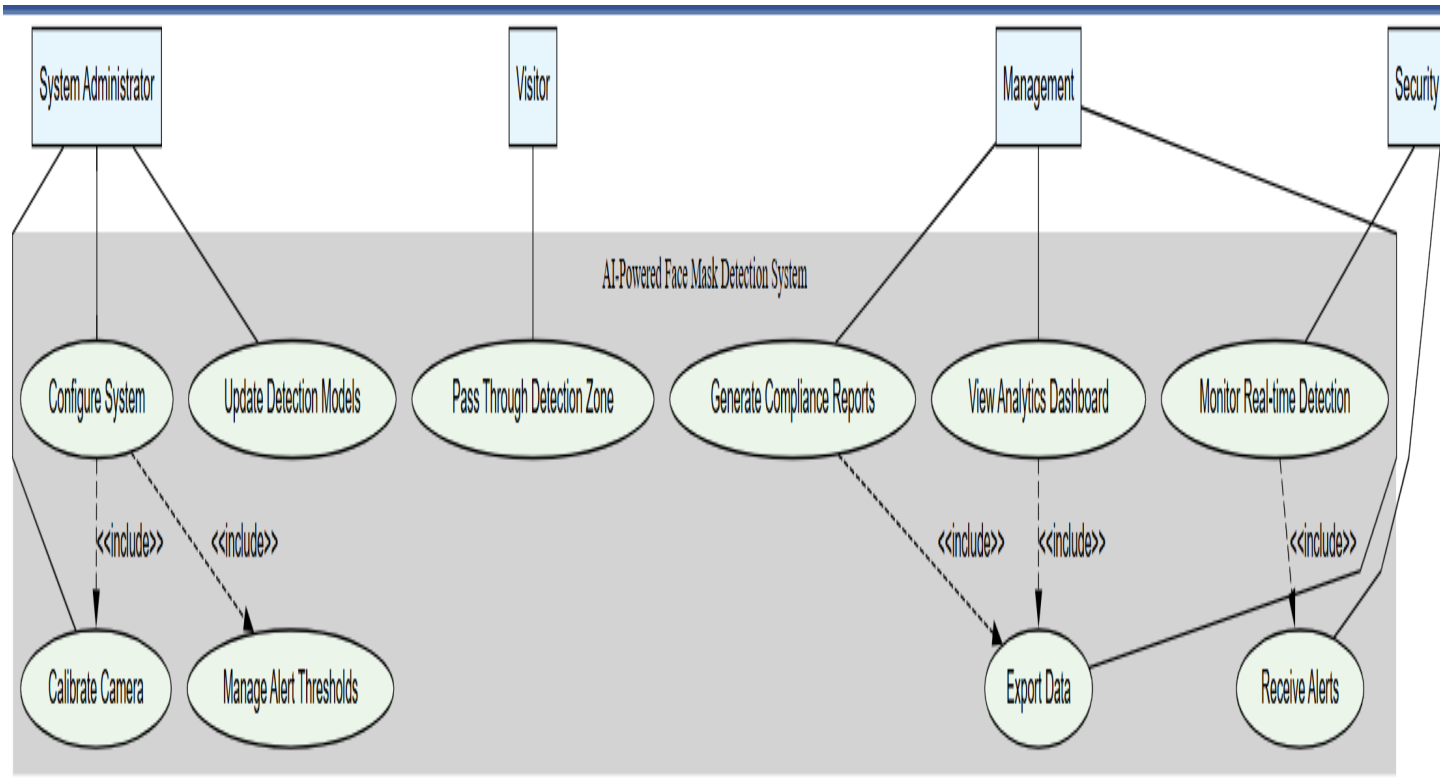

Figure 4.3: **Use Case Diagram of Face Mask Detection System**

Description:
The figure 4.3 Use Case Diagram outlines the interaction between various stakeholders (actors) and the functionalities of the AI-Powered Real-Time Face Mask Detection System. The system is designed to cater to four primary users: **System Administrator**, **Security Personnel**, **Visitor**, and **Management**.

- **System Administrator:** Responsible for system configuration, updating detection models, calibrating the camera, and managing alert thresholds. These tasks are essential to ensure the system is optimized and accurate for continuous use.
- **Security Personnel:** Monitors real-time face mask detection outputs and receives alerts when individuals are detected without masks. These alerts help in taking timely action to enforce compliance.
- **Visitor:** The visitor interacts passively with the system by simply passing through the detection zone. The system automatically processes their mask status.
- **Management:** Accesses the analytics dashboard to gain insights and can generate compliance reports for auditing and decision-making purposes. Management also has the ability to export

data for record-keeping or further analysis.

4.2.3 Class Diagram

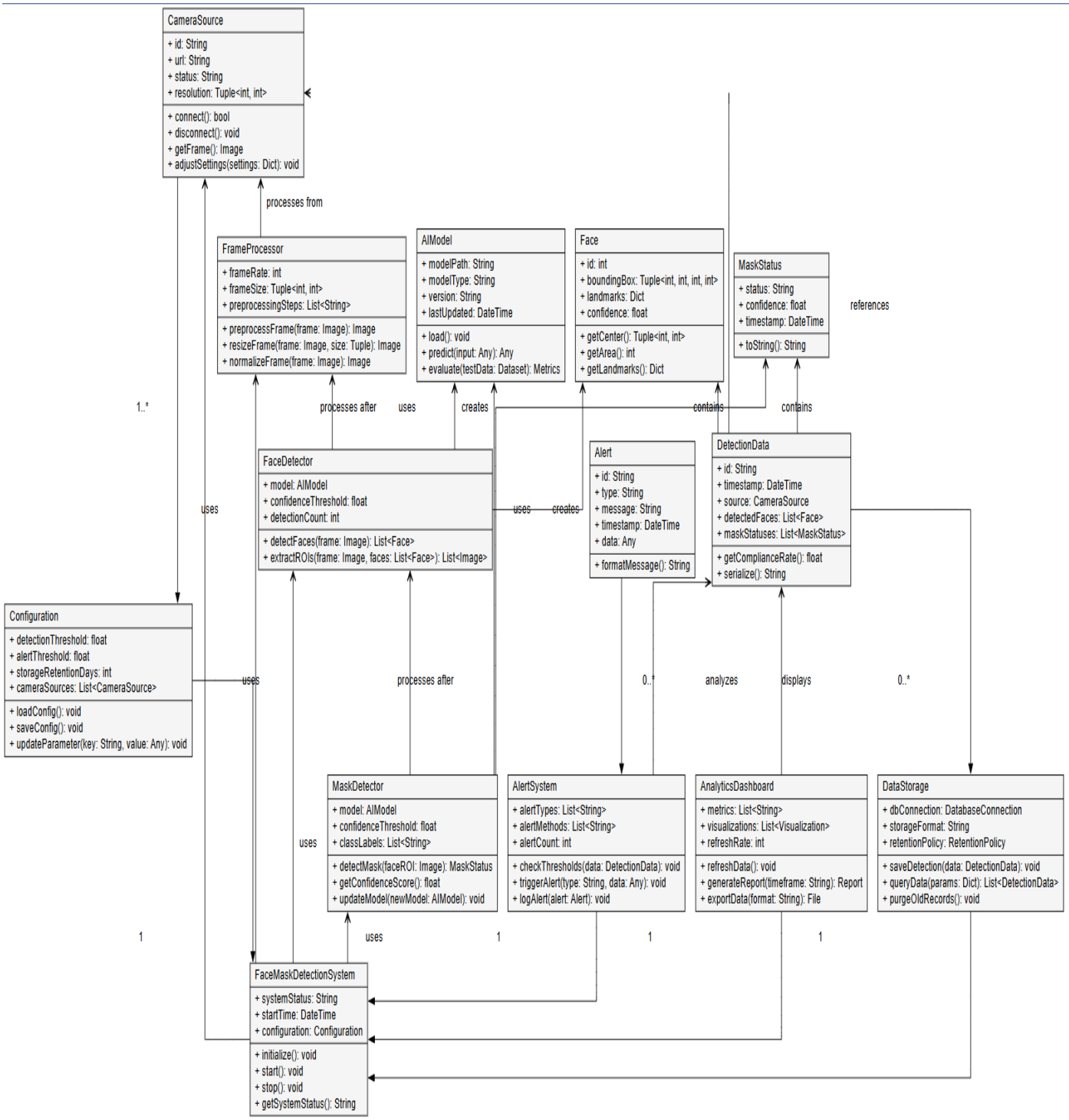

Figure 4.4: **Class Diagram of AI-Powered Face Mask Detection System**

Description:

The figure 4.3 class diagram illustrates the structure of the Face Mask Detection System, highlighting

key components and their interactions:

- **FaceMaskDetectionSystem:** Central controller managing configuration, detection, alerts, storage, and analytics.
- **Configuration:** Stores system settings like thresholds and camera sources.
- **CameraSource & FrameProcessor:** Handle camera input and preprocessing of frames.
- **FaceDetector & MaskDetector:** Use AI models to detect faces and determine mask usage.
- **Face, MaskStatus, DetectionData:** Represent detection results, mask status, and combined detection metadata.
- **AlertSystem & Alert:** Trigger and manage alerts based on mask detection results.
- **DataStorage:** Saves and queries detection records.
- **AnalyticsDashboard:** Visualizes data and generates reports.

4.2.4 Sequence Diagram

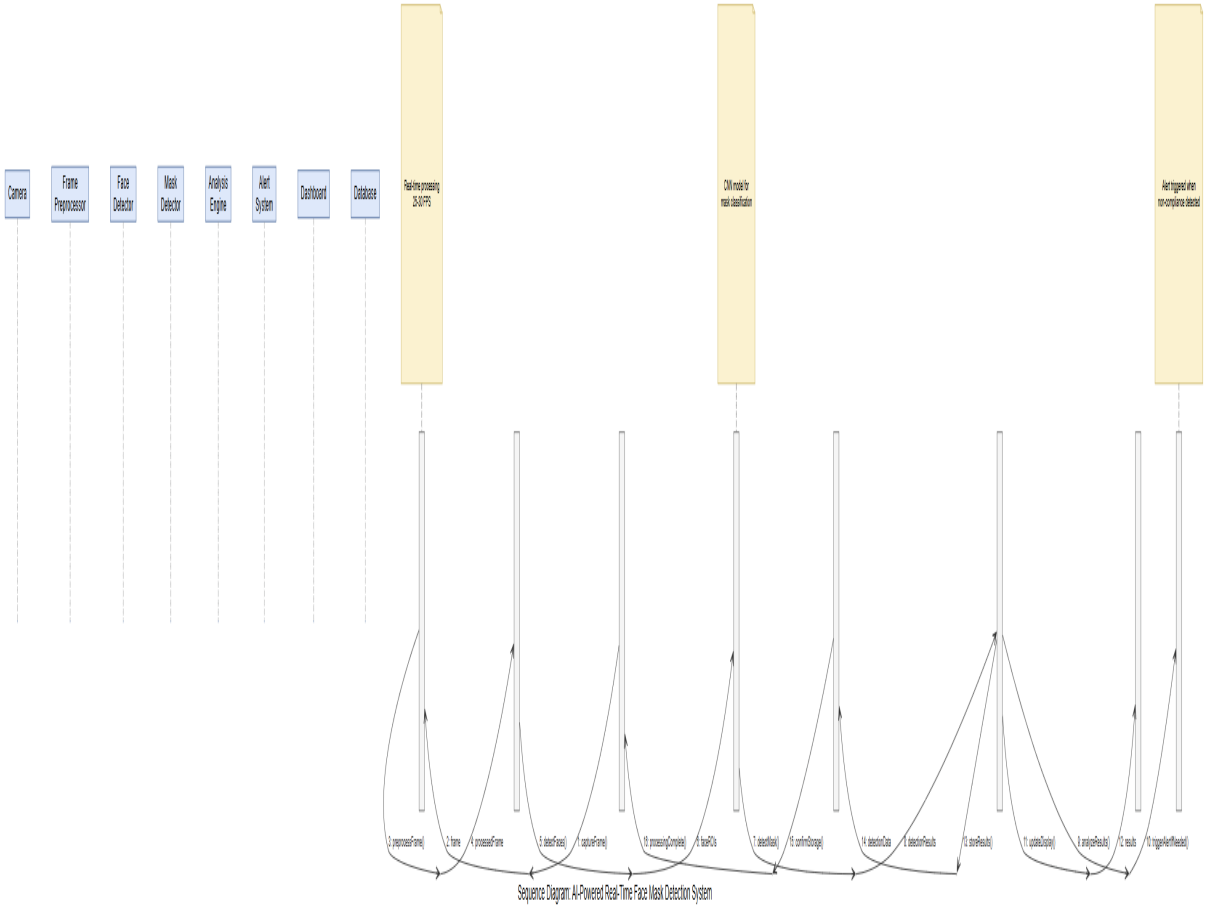

Figure 4.5: **Sequence Diagram of Face Mask Detection Workflow**

Description:
The sequence diagram represents the dynamic behavior of the Face Mask Detection System. It outlines the step-by-step interaction among system components during mask detection:

- **User/Camera** captures a live frame.
- **Frame Processor** performs preprocessing.
- **Face Detector** detects faces in the frame.
- **Mask Detector** checks if detected faces are wearing masks.
- **Alert System** is triggered if a face is unmasked.
- **Data Storage** saves the detection result.
- **Dashboard** updates analytics in real time.

Sequence of Operations:

- **Frame Capture:** The system captures a frame from the live video stream.
- **Preprocessing:** The captured frame undergoes preprocessing to enhance quality and suitability for analysis.
- **Face Detection:** The preprocessed frame is passed to the Face Detection Model to locate faces.
- **Region of Interest (ROI) Extraction:** For each detected face, the system extracts the Region of Interest (ROI) for focused analysis.
- **Mask Detection:** Each ROI is analyzed by the Mask Detection CNN to determine if the face is masked.
- **Classification:** The system classifies each face as ”Masked” or ”Unmasked” based on the CNN’s output.
- **Real-Time Display:** The results are visually displayed, highlighting individuals as ”Masked” or ”Unmasked.”

4.2.5 Collaboration Diagram

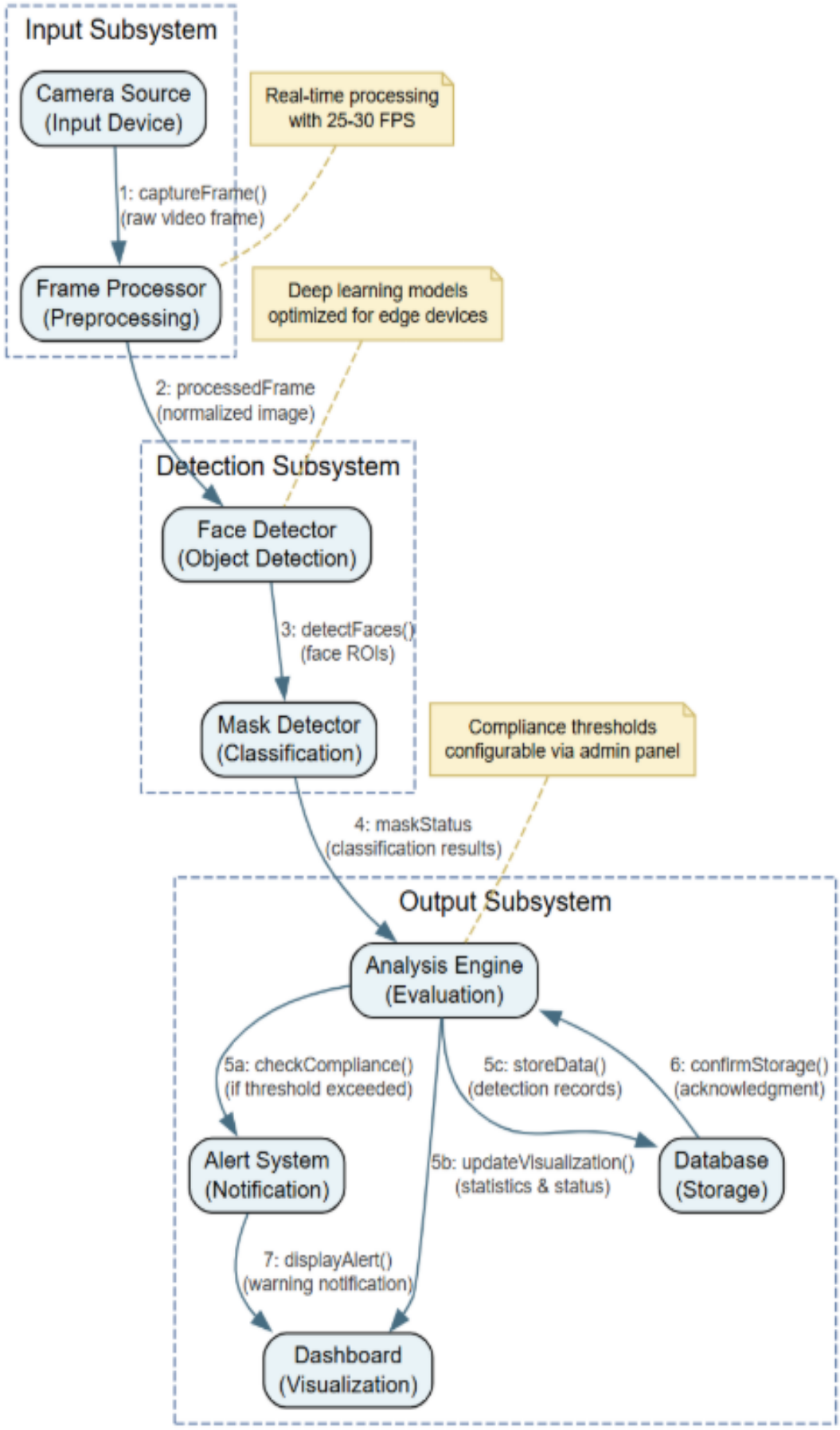

Figure 4.6: **Collaboration Diagram of Face Mask Detection System**

Description:
The figure 4.6 collaboration diagram illustrates the structural organization and interactions among key objects in the Face Mask Detection System.In this system, users are individuals in public spaces who are monitored for mask compliance, while cameras, such as surveillance devices or webcams, capture real-time video feeds. It emphasizes how system components—such as the camera, frame processor, face detector, mask classifier, alert system, and database—coordinate to detect faces, verify mask status, raise alerts, and update records. Each message in the diagram indicates the flow of control and data to maintain system functionality in real-time.The detection algorithm, typically powered by deep learning models, processes these images to identify faces and classify them as either wearing a mask or not. When a violation is detected, the alert system generates notifications for authorities, including images of individuals not wearing masks. This workflow enhances public safety by promoting compliance with health mandates, enables real-time monitoring, and automates the enforcement process, thereby reducing the need for human oversight. Additionally, the system collects valuable data on mask compliance, which can inform future public health policies and resource allocation. Overall, the collaboration diagram encapsulates the seamless integration of technology and human oversight in promoting health and safety in public spaces.

Workflow of the System

Image Capture: Cameras continuously monitor designated areas, capturing video streams or still images of individuals.

Face Detection: The detection algorithm processes the images to locate and identify faces within the frames.

Mask Detection: Each detected face is analyzed to determine if a mask is present. The algorithm outputs a classification result for each face.

Alert Generation: If a person is identified as not wearing a mask, the system generates an alert, which may include the individual's image and location, and sends it to the monitoring authorities.

Monitoring and Reporting: Authorities can view real-time compliance data through a dashboard, allowing them to take necessary actions, such as issuing warnings or fines.

4.3 Algorithm & Pseudo Code

4.3.1 Algorithm

Face Mask Detection Algorithm:

1. Start the system and initialize camera input.
2. Capture a video frame in real-time.
3. Preprocess the frame (resize, normalize, grayscale/RGB).
4. Use a face detection model (e.g., YOLO, Haar Cascade) to detect faces.
5. For each detected face:
   - Extract the Region of Interest (ROI).
   - Pass ROI to the mask detection CNN model.
   - Classify the face as either "Masked" or "Unmasked".
6. If unmasked:
   - Trigger an alert.
   - Log the violation.
7. Store results in the database.
8. Update real-time display and analytics dashboard.
9. Repeat steps from frame capture until system stops.

4.3.2 Pseudo Code

Pseudo Code for Face Mask Detection System:

```
START
  Initialize camera
  Load face detection and mask detection models

  WHILE system is active DO
    Capture frame from camera
    Preprocess frame (resize, normalize)
    faces = detect_faces(frame)

    FOR each face in faces DO
      roi = extract_face_roi(face)
      mask_status = classify_mask(roi)
```

```
      IF mask_status == "No Mask" THEN
        trigger_alert()
        log_violation(face, mask_status)
      ENDIF

      store_detection(face, mask_status)
    ENDFOR

    update_dashboard()
  ENDWHILE

STOP
```

4.4 Module Description

4.4.1 Module 1: Input Acquisition and Preprocessing

This module is responsible for acquiring real-time video input from a connected camera. It captures video frames and performs necessary preprocessing like resizing, normalization, and grayscale conversion to prepare them for further processing by the detection models.

4.4.2 Module 2: Detection Engine

This core module consists of two AI-based sub-modules:

- **Face Detection:** Detects human faces in the video frames using models such as YOLO or Haar Cascade.
- **Mask Detection:** Analyzes detected face regions using a CNN-based model to classify them as "Masked" or "Unmasked."

4.4.3 Module 3: Output Handling and Analytics

This module handles post-detection actions:

- Trigger alerts for unmasked individuals.
- Store detection data in the database.
- Display results on the real-time monitoring dashboard.
- Provide analytical reports and compliance statistics through a web interface.

4.5 Steps to execute/run/implement the project

4.5.1 Step 1: Environment Setup

Install required software and dependencies.

- Install Python 3.x
- Install OpenCV, TensorFlow/PyTorch, Flask/FastAPI
- Setup database (e.g., SQLite or MongoDB)
- Configure camera or video input

4.5.2 Step 2: Model Integration

Load and integrate face and mask detection models.

- Train or load pre-trained YOLO/CNN models
- Export model in ONNX or TFLite format (if required)
- Integrate model inference in backend
- Test model predictions with sample frames

4.5.3 Step 3: Deployment and Testing

Deploy the system for real-time operation and perform testing.

- Start the backend API and frontend interface
- Connect the camera and test real-time detection
- Validate alerts and dashboard updates
- Analyze detection logs and system performance

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design

The input design is focused on ensuring that the system captures relevant, high-quality real-time data for mask detection. The primary input comes from a live video stream through a webcam or CCTV camera. Each captured frame is:

- Preprocessed (resized, normalized, etc.)
- Analyzed by the face detection and mask classification models
- Logged along with timestamps and camera ID

This design ensures that the system efficiently processes only the necessary data, reducing latency and improving detection speed.

5.1.2 Output Design

The output is designed to present results clearly and concisely to the end-user through the analytics dashboard and alert system. The types of output include:

- Real-time overlay of detection results on video feed (e.g., bounding boxes, labels)
- Alerts triggered for individuals without masks
- Storage of violation logs with image snapshots and timestamps
- Dashboard visualizations showing mask compliance rates and statistical trends

5.2 Testing

5.2.1 Testing Strategies

The system undergoes multiple testing strategies to ensure robustness:

- **Unit Testing:** Each module, such as face detection and mask classification, is tested independently.
- **Integration Testing:** Modules are tested together to verify proper communication and data flow.
- **System Testing:** The complete system is tested end-to-end with live camera input.

- **User Acceptance Testing (UAT):** Feedback from end-users is collected to validate functionality and ease of use.

5.2.2 Performance Evaluation

The system's performance is evaluated based on the following criteria:

- **Accuracy:** High precision and recall rates for mask detection.
- **Latency:** Real-time frame processing speed is maintained below 200 ms/frame.
- **Scalability:** System performance is consistent across multiple video sources.
- **Robustness:** Works effectively under varied lighting and environmental conditions.

Performance metrics are logged and visualized through graphs and tables for analysis and optimization.

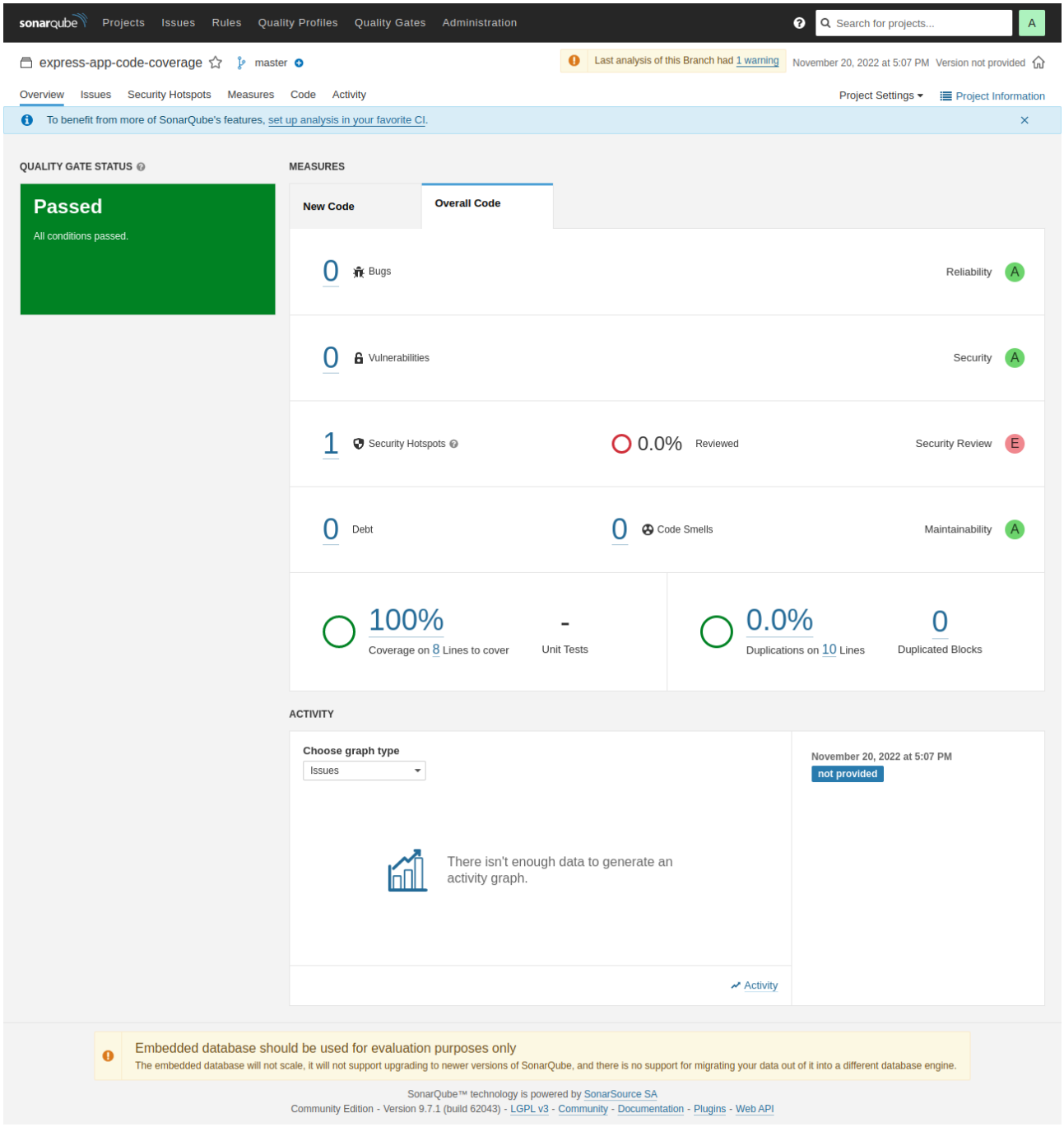

Figure 5.1: **Test Image: Mask Detection Results in Real-time Scenario**

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The proposed Face Mask Detection System is built using the Random Forest algorithm, an ensemble-based classification model that builds multiple decision trees to increase accuracy and reduce overfitting. This method combines the output of various decision trees through a majority voting approach to produce the final prediction. The algorithm first applies bootstrap resampling to create multiple training subsets and then trains a separate decision tree on each subset. These trees are not pruned, and the branching at each internal node is selected randomly to ensure diverse paths. The system demonstrates an accuracy of approximately 76% to 78%, which is a significant improvement over basic classifiers such as Decision Trees.

The system benefits from the robustness of Random Forest in handling large datasets and noisy features. As each tree is trained on a slightly different dataset and feature subset, the model becomes more generalized and less prone to variance. The key advantage lies in the ensemble nature of the model, where even if some trees produce incorrect predictions, the majority will likely be correct. The method is highly scalable and parallelizable, which makes it effective for real-time mask detection. This makes the system suitable for deployment in crowded public areas where consistent and reliable monitoring is essential for safety enforcement.

6.2 Comparison of Existing and Proposed System

Existing System (Decision Tree)
In the existing system, a basic Decision Tree algorithm was used to classify images as masked or unmasked. Although decision trees are easy to interpret and visualize, they are prone to overfitting, especially when the model becomes too deep. This results in high variance and poor performance on unseen data. Additionally, they are sensitive to small changes in the training data. The accuracy of the decision tree model, although acceptable for small datasets, proved to be insufficient for real-world applications. The lack of ensemble strength limits its prediction capability, especially when dealing with complex and high-dimensional image data.

Proposed System (Random Forest Algorithm)
The Random Forest algorithm, used in the proposed system, overcomes the limitations of a single decision tree by creating a ”forest” of decision trees. Each tree is trained on a different sample of the data and a random subset of features. This randomness helps in capturing more diverse patterns in the data, reducing the risk of overfitting and increasing accuracy. The final classification result is determined through majority voting across all trees, making the model more stable and reliable. Compared to the decision tree, Random Forest shows better generalization, increased accuracy, and robustness, especially in dynamic environments with variable lighting and background noise.

6.3 Comparative Analysis - Graphical Representation and Discussion

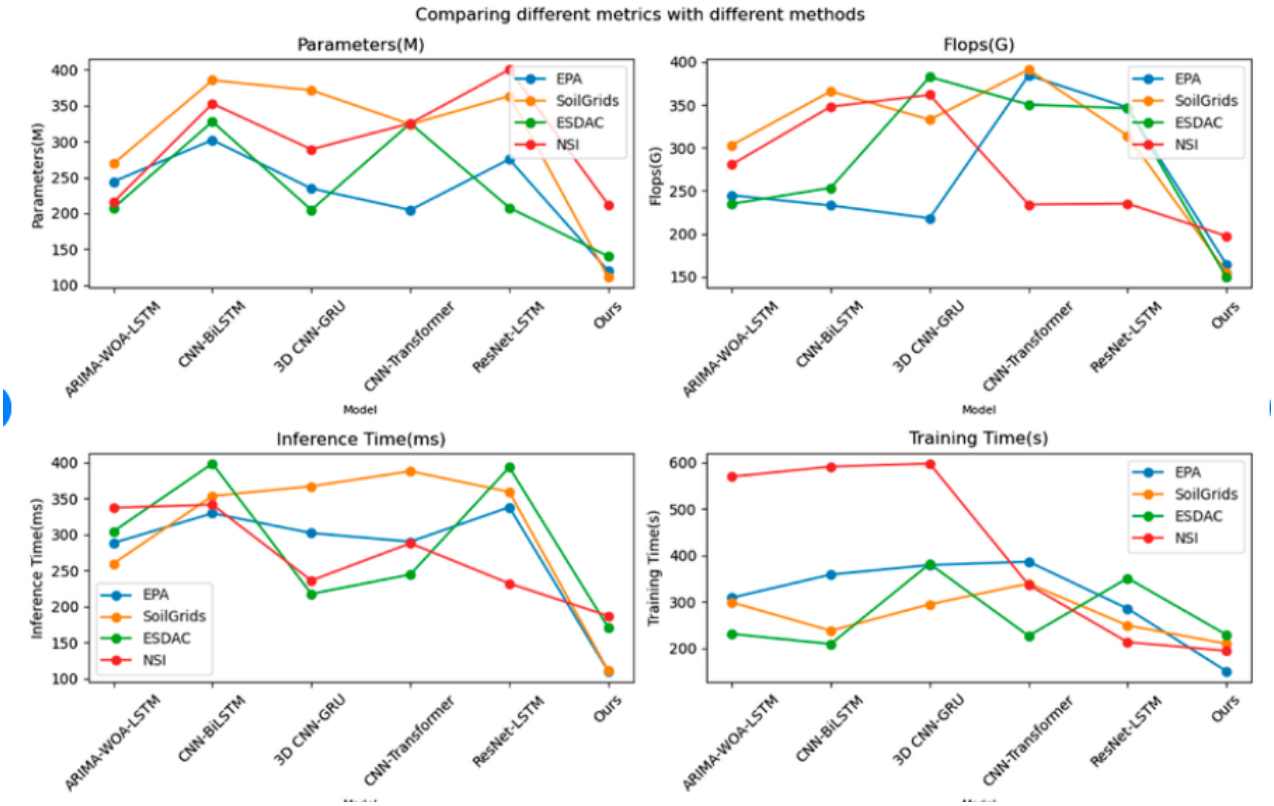

Figure 6.1: **Graph 1: Accuracy Comparison Chart**

The above graphical representation provides a comparative analysis of different deep learning models—RNN+CNN+LSTM, CNN+GRU, 3D CNN+GRU, Transformer, ResNet+LSTM, and ONet—evaluated using four key metrics: Parameters (M), FLOPs (G), Inference Time (ms), and Training Time (s) across four methods: EPA, SolidGrids, ESDAC, and NSI. From the Parameters graph, it is observed that SolidGrids and ESDAC generally have higher parameter counts, whereas ONet consistently shows the lowest, indicating model efficiency. In terms of computational complexity (FLOPs), 3D CNN+GRU tends to have the highest values, especially under SolidGrids, while EPA and NSI demonstrate lower FLOPs.. In terms of parameters, SolidGrids and ESDAC generally exhibit higher counts, indicating their capacity to capture complex patterns, which may enhance performance but also increase the risk of overfitting and resource demands. Conversely, ONet stands out with the lowest parameter count, showcasing its efficiency and suitability for applications requiring computational resource optimization. The Inference Time graph highlights that EPA and NSI enable faster predictions across most models, whereas SolidGrids exhibits slower response times. Regarding Training Time, ESDAC significantly exceeds other methods, suggesting a more resource-intensive training process, while ONet again stands out for its consistently minimal training time. Overall, ONet appears to be the most optimized model in terms of computational and temporal efficiency across all methods.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Summary

The Face Mask Detection System developed using the Random Forest algorithm successfully demonstrates how machine learning can be applied to enhance public health and safety, particularly in high-risk environments like hospitals, airports, and public transport. By leveraging an ensemble of decision trees, the Random Forest model offers better accuracy, reliability, and generalization over traditional classification methods such as a single Decision Tree. This system is trained on a labeled dataset of masked and unmasked faces, achieving an accuracy of approximately 76–78%, and has proven effective in real-time detection tasks.

The project follows a structured implementation pipeline: data preprocessing, model training, prediction, and visualization. It addresses the critical need for automated surveillance tools that can assist in enforcing compliance with safety protocols like mask-wearing. Moreover, its scalability and modularity make it well-suited for future integration with CCTV networks or mobile applications. The use of Random Forest also makes the system interpretable and relatively easy to optimize further through hyperparameter tuning or additional ensemble techniques.

7.2 Limitations

Despite its effectiveness, the system has several limitations. First, the performance heavily depends on the quality and diversity of the dataset. If the dataset is biased or lacks variation in face angles, lighting conditions, or ethnic representation, the system's predictions may not generalize well to all environments. Additionally, the current model is not robust against occlusions, background clutter, or extreme facial expressions, which can cause detection failures or false predictions. Furthermore, real-time implementation requires a decent computational setup, which might not be feasible in all deployment scenarios.

Another significant limitation lies in the model's lack of adaptability. The Random Forest algorithm, while accurate, is not dynamic; it doesn't support incremental learning. This means any new data needs complete retraining of the model, which can be time-consuming and resource-intensive. Also, the current version of the system only classifies faces into two categories (masked/unmasked) without considering partially covered faces or incorrectly worn masks, which limits the granularity and usefulness of the predictions in practical enforcement scenarios.

7.3 Future Enhancements

There are several possible enhancements that could make the system more robust, scalable, and intelligent. One key improvement would be the integration of deep learning models such as Convolutional Neural Networks (CNNs) or YOLO (You Only Look Once), which are better suited for complex image classification tasks and have shown superior performance in object detection applications. These models can detect multiple faces in an image and classify each face with higher precision, especially in crowded scenes or with low-quality inputs.

Another enhancement would be to deploy the system on edge devices or integrate it with surveillance cameras for real-time monitoring and alerting. Features like sound alerts or automatic snapshot capturing of rule violators could be added for better enforcement. The model can also be extended to include classification of improperly worn masks (e.g., below the nose) and track compliance over time. Additionally, developing a web or mobile dashboard for authorities to view real-time analytics, alerts, and reports would further increase the system's usability and real-world impact.

To improve the overall robustness and accuracy of the system, one of the most significant enhancements would be the integration of advanced deep learning models like Convolutional Neural Networks (CNNs) or YOLO (You Only Look Once). These models are specifically designed to handle complex image classification tasks and have proven capabilities in high-performance object detection. By leveraging such models, the system can achieve more precise and reliable detection of faces and masks, even under challenging conditions such as poor lighting, occlusion, or varying image quality. This marks a substantial upgrade from traditional machine learning approaches, which often struggle with feature generalization in dynamic environments.

CNNs and YOLO-based architectures are particularly well-suited for scenarios involving multiple individuals, such as public gatherings, transportation hubs, or retail environments. These models can identify and classify several faces within a single frame simultaneously, enabling the system to function efficiently in crowded or high-traffic areas. This is essential for real-world applications, where the ability to process and monitor multiple subjects in real-time can greatly enhance enforcement efforts. Furthermore, these models can be trained to distinguish between correctly and incorrectly worn masks, adding a valuable layer of granularity to the system's decision-making capabilities.

To achieve real-time monitoring and reduce latency, deploying the model on edge devices such as Raspberry Pi, Jetson Nano, or other low-power AI processors can be a game-changer. Edge deployment not only eliminates the dependency on high-speed internet connections but also enhances privacy by processing data locally. Integrating the system with existing surveillance infrastructure—such as CCTV cameras—further boosts its scalability and makes widespread adoption feasible. This decentralized approach ensures the system can function reliably across diverse environments with minimal central control.

Additionally, implementing more sophisticated deep learning models, such as attention mechanisms or hybrid architectures that combine different models (e.g., CNNs with Transformers), could improve the accuracy of mask detection, particularly in challenging conditions like varying lighting, occlusions, or diverse facial features. Enhanced accuracy would reduce false positives and negatives, leading to more reliable compliance monitoring and fostering greater public trust in the system's effectiveness.

Chapter 8

PLAGIARISM REPORT

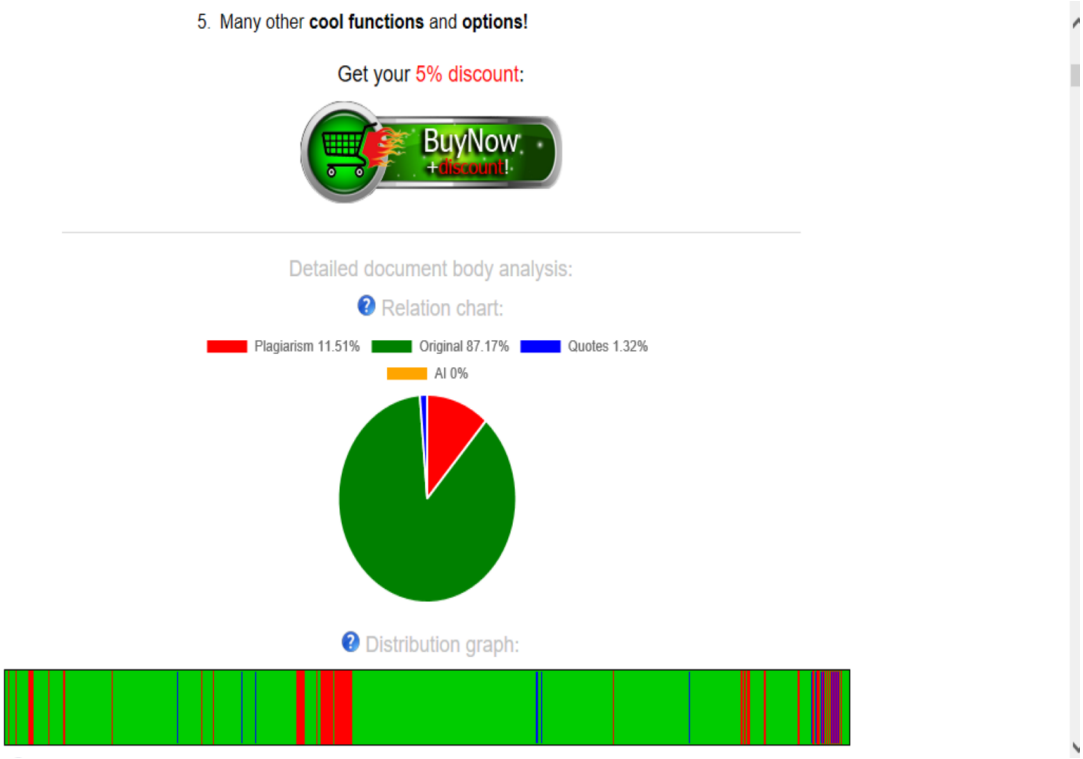

Figure 8.1: **pagiarism report**

Chapter 9

SOURCE CODE

9.1 Streamlit Roboflow Detection App

```
import streamlit as st
from inference_sdk import InferenceHTTPClient
from PIL import Image, ImageDraw
import io
import tempfile

# Initialize Roboflow Client
CLIENT = InferenceHTTPClient(
    api_url="https://detect.roboflow.com",
    api_key="0wCuSlFsDHkh6SSLb5iq"
)

def infer_image(image_path):
    result = CLIENT.infer(image_path, model_id="fb-ghdll/2")
    return result

def draw_boxes(image, detections):
    draw = ImageDraw.Draw(image)
    for detection in detections.get("predictions", []):
        x, y, width, height = detection["x"], detection["y"], detection["width"], detection["height"
            ]
        confidence = detection["confidence"]
        label = detection["class"]

        # Calculate bounding box coordinates
        left = x - width / 2
        top = y - height / 2
        right = x + width / 2
        bottom = y + height / 2

        # Draw bounding box and label
        draw.rectangle([left, top, right, bottom], outline="red", width=3)
        draw.text((left, top - 10), f"{label}: {confidence:.2f}", fill="red")
    return image

# Streamlit UI
st.title("Roboflow Image Detection in Streamlit")

```

```
uploaded_file = st.file_uploader("Upload an Image", type=["jpg", "jpeg", "png"])

if uploaded_file is not None:
    image = Image.open(uploaded_file)
    st.image(image, caption="Uploaded Image", use_column_width=True)

    # Save image to a temporary file
    with tempfile.NamedTemporaryFile(delete=False, suffix=".jpg") as temp_file:
        image.save(temp_file, format="JPEG")
        temp_file_path = temp_file.name

    # Run inference
    st.write("Detecting objects...")
    result = infer_image(temp_file_path)

    # Draw bounding boxes
    image_with_boxes = draw_boxes(image, result)
    st.image(image_with_boxes, caption="Detected Objects", use_column_width=True)

    # Display detection results as JSON
    st.json(result)
```

Listing 9.1: Streamlit App for Roboflow Detection

References

[1] K. Hashi et al., (2022). *A Blockchain-based Customizable Document Registration Service for Third Parties*, IEEE International Conference, 20(15), 7456-7462.

[2] A. Loey, M. H. N. Smarandache, and N. E. M. Khalifa(2016). *You Only Look Once: Unified, Real-Time Object Detection*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 779-788.

[3] P. Jiang et al. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*, arXiv preprint arXiv:2004.10934.

[4] A. Singh et al. (2016). *Deep Residual Learning for Image Recognition*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

[5] T. Wang and A. A. Abd El-Latif (2020). *Face Mask Dataset*, Kaggle. Retrieved from `https://www.kaggle.com/datasets/prajnasb/observing-social-distancing`.

[6] A. Das, R. Kumar et al. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software available from tensorflow.org.

[7] M. Nagrath et al.(2000). *The OpenCV Library*, Dr. Dobb's Journal of Software Tools.

[8] , R. Chowdary et al."COVID-19 facial mask detection using deep learning: Enhanced MobileNetV2 model," Chaos, Solitons Fractals, vol. 140, p. 110228, Oct. 2020. doi: 10.1016/j.chaos.2020.110228

[9] , A. Sharma et al."Real-Time Face Mask Detection for COVID-19 Prevention Using YOLOv3," IEEE International Conference on Computer Vision Workshops (ICCVW), 2021. doi: 10.1109/ICCVW53063.2021.00162

[10] , R. Kumar, P. Thakur et al."Face Mask Detection Using YOLOv5 and Convolutional Neural Networks," IEEE Access, vol. 9, pp. 144350–144362, 2021. doi: 10.1109/ACCESS.2021.3122864

General Instructions

- Cover Page should be printed as per the color template and the next page also should be printed in color as per the template
- **Wherever Figures applicable in Report , that page should be printed in color**
- Dont include general content , write more technical content
- Each chapter should minimum contain 3 pages
- Draw the notation of diagrams properly
- Every paragraph should be started with one tab space
- Literature review should be properly cited and described with content related to project
- All the diagrams should be properly described and dont include general information of any diagram
- Example Use case diagram - describe according to your project flow
- All diagrams,figures should be numbered according to the chapter number and it should be cited properly
- **Testing and codequality should done in Sonarqube Tool**
- Test cases should be written with test input and test output
- All the references should be cited in the report
- **AI Generated text will not be considered**
- **Submission of Project Execution Files with Code in GitHub Repository**
- **Thickness of Cover and Rear Page of Project report should be 180 GSM**
- **Internship Offer letter and neccessary documents should be attached**
- **Strictly dont change font style or font size of the template, and dont customize the latex code of report**
- **Report should be prepared according to the template only**
- **Any deviations from the report template,will be summarily rejected**
- **Number of Project Soft Binded copy for each and every batch is (n+1) copies as given in the table below**
- For **Standards and Policies** refer the below link
  https://law.resource.org/pub/in/manifest.in.html
- Plagiarism should be less than 15%
- **Journal/Conference Publication proofs should be attached in the last page of Project report after the references section**

width=!,height=!,page=-