A Project report

On

# Surveillance System with Human Intrusion Detection

Submitted in partial fulfilment of the requirement of

**University of Mumbai**

For the Degree of

**Bachelor of Engineering**

*in*

**COMPUTER ENGINEERING**

*Submitted by*

**Lawrence Rodriques**
**Swati Padmanabhan**
**Sumit Prasad**
**Neha Auti**

*Supervised by*

**Mrs. Nupur Gaikwad**


**Department of Computer Engineering**

**Fr. Conceicao Rodrigues Institute of Technology**
**Sector 9A, Vashi, Navi Mumbai - 400703**

**UNIVERSITY OF MUMBAI**

**2023-2024**

# APPROVAL SHEET

This is to certify that the project entitled

**"Surveillance System with Human Intrusion Detection "**

**Submitted by**


| | |
|---|---|
| **Lawrence Rodriques** | **1020238** |
| **Swati Padmanabhan** | **1020266** |
| **Sumit Prasad** | **1020267** |
| **Neha Auti** | **1020268** |


**Supervisors :** _____

**Project Coordinator :** _____

**Examiners : 1.** _____

**2.** _____

**Head of Department :** _____

**Date :**

**Place :**

# Declaration

We declare that this written submission for B.E. Project entitled "**Surveillance System with Human Intrusion Detection** " represent our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declared that we have adhered to all principles of academic honesty and integrity and have not misrepresented, fabricated, or falsified any ideas/data/facts/sources in our submission. We understand that any violation of the above will cause disciplinary action by the institute and also evoke penal action from the sources that have thus not been properly cited or from whom paper permission has not been taken when needed.

**Project Group Members:**

1. Lawrence Rodriques, 1020238

_____

2. Swati Padmanabhan, 1020266

_____

3. Sumit Prasad, 1020267

_____

4. Neha Auti, 1020268

_____

# Abstract

Security in restricted areas is essential for protecting valuable assets, sensitive information, ensuring the safety from intruders. Traditional security systems have many limitations, where they cannot authenticate whether the entered person is an intruder or not. To address this challenge by implementing a real-time face recognition-based surveillance system, is the goal of this project. Realtime Intrusion detection system provides surveillance for restricted and confidential areas with help of face recognition and detection, when an intruder or unauthorized person enters the area, this system will give an alert to the respective in charge through various channels, including email, messaging services, and direct phone calls. In this system, the OpenCV python library along with several algorithms are used to abstract the facial features and to take the input dataset. For face detection, the system utilizes SCRFD and YOLO, and it employs Arcface for accurate face recognition. This technique ensures the system can distinguish between an intruder and an authorized individual entering the secured area. This proactive approach enhances surveillance efficiency and reinforces the safety and integrity of restricted areas. For instance, when an individual enters the restricted area, the system captures and analyses their face. It then verifies whether the detected face matches any authorized faces in the registered user database. If there's no match, the system identifies the person as an intruder and promptly sends an alert to the designated authority. To enhance accessibility, a user-friendly graphical interface (GUI) has been developed using Python's Tkinter.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

The widespread integration of closed-circuit television (CCTV) cameras highlights the critical role of surveillance systems in ensuring the security of private spaces. However, manually identifying authorized and unauthorized individuals, along with detecting potential intruders within the extensive CCTV network, presents significant challenges. Traditional security systems, responsible for safeguarding valuable assets and ensuring the security of restricted areas, face limitations in authenticating individuals. This challenge becomes particularly evident when distinguishing between authorized personnel and potential intruders, highlighting the urgent need for a more sophisticated and efficient surveillance solution. Traditional monitoring techniques compound these challenges, proving inefficient and inaccurate, especially in responding promptly to intrusions. This critical gap underscores the pressing necessity for a novel and automated solution that leverages cutting-edge Computer Vision and Machine Learning techniques. The project is designed to speed up the identification and timely notification of intrusions, placing emphasis on enhancing surveillance analytics, strengthening security measures, and ensuring a swift response to potential threats. The goal is to significantly advance the fields of surveillance analytics, security, and intrusion response through the integration of facial recognition and real-time capabilities. By addressing the deficiencies in traditional monitoring methodologies, the project aims to redefine the effectiveness of security operations, ensuring a proactive and responsive approach to potential intrusions. The implementation of a real-time face recognition-based surveillance system is poised to represent a transformative leap forward, promising to significantly impact and advance the realms of surveillance, security, and emergency response operations.

## 1.2 Motivation

To addresses critical security concerns in restricted areas by introducing a real-time face recognition-based surveillance system. Traditional security systems often struggle to authenticate individuals, leaving valuable assets vulnerable. Unlike unmonitored CCTVs, this system employs advanced face detection and recognition algorithms to promptly identify intruders. The timely alert mechanism, utilizing various communication channels, ensures a swift response from designated authorities. By overcoming the limitations of manual video tracking, the system enhances surveillance efficiency. Additionally, a user-friendly graphical interface has been developed

to make the technology accessible, reinforcing the safety and integrity of restricted areas.

## 1.3 Aim and Objective

**Aim:** The project aims to develop a real-time intruder detection and alert system that enhances traditional CCTV systems by integrating advanced technologies such as machine learning and computer vision. The primary focus is on efficient face detection and recognition, allowing the system to autonomously identify unauthorized individuals within predefined zones. This approach aims to provide early alerts to administrators and security professionals, thereby improving the overall efficiency and reliability of security measures in private spaces.

**Objectives:**

- **User-Friendly Interface:** Design and develop an user-friendly interface for administrators to monitor and manage the system effectively, providing insights into intruder detection and overall security status.

- **Real-time Detection and Immediate Response:** Implement a system that delivers real-time detection of human intrusions and initiates immediate responses upon detection. The system will promptly alert security personnel and administrators through various channels, including email, messaging services, and direct phone calls, ensuring rapid reactions to potential security breaches.

- **Integrated Face Recognition Surveillance:** Perform comprehensive evaluations of the system's functionality in a variety of surveillance scenarios, covering both indoor and outdoor environments. Assess its performance under different lighting conditions and potential occlusions to ensure adaptability and robustness across diverse real-world situations.

- **Face Recognition API:** Develop a versatile Face Recognition API with the objective of providing a scalable and interoperable solution, seamlessly integrable into multiple systems. The API aims to offer reliable and efficient facial recognition capabilities, enhancing security and user identification across diverse applications and platforms.

## 1.4 Report Outline

The report provides an overview of our strategy for crafting an innovative surveillance-based system with human detection capabilities. Our key objective is to engineer a seamless solution empowering administrators to promptly and precisely recognize individuals or objects of interest, streamlining operations without reliance on external software tools. This document offers a detailed exploration of our methodology, covering essential project elements such as video processing, sophisticated computer vision algorithms, and intuitive interface design.

# Chapter 2

# Study Of the System

The project focuses on developing an intelligent surveillance system that combines surveillance cameras with advanced artificial intelligence. It showcases how the system actively monitors and swiftly detects intrusions using state-of-the-art machine learning techniques. By analyzing video footage in real-time, it automatically identifies potential security threats and unusual activities, enhancing overall safety. Moreover, the project highlights the versatility of this technology across various sectors, from safeguarding sensitive areas to ensuring the security of residential properties. With its adaptive and learning capabilities, the system stands as a reliable solution to address the evolving security challenges of today's world.

## 2.1 About the Technique

The Arcface algorithm, known for its efficiency and precision in face recognition tasks. By employing an additive angular margin loss function, Arcface optimizes the discriminative power of learned features, ensuring high-precision face recognition even in challenging conditions. For face detection, we employ a combination of SCRFD and YOLO. SCRFD, designed for handling faces of various scales and resolutions, excels in detecting small faces within large scenes often encountered in CCTV footage. Meanwhile, YOLO's speed and accuracy in object detection, treating it as a regression problem, enable predicting bounding boxes and class probabilities swiftly in a single pass, ideal for real-time applications. This integrated approach allows our system to accurately detect and recognize faces in real-time, enhancing security and surveillance measures effectively.

## 2.2 Various Available Techniques

1. **YOLO (You Only Look Once)**: YOLO (You Only Look Once) is a well-known real-time object detection algorithm recognized for its efficiency in swiftly detecting objects. It functions by employing a single convolutional neural network (CNN) across the entire image, dividing it into regions and predicting bounding boxes and probabilities for each region. YOLO is acclaimed for its capacity to achieve high accuracy in real-time, necessitating only one forward propagation pass through the neural network for predictions. Following non-max suppression to ensure each object is detected only once, YOLO provides outputs of recognized objects along with their respective bounding boxes. This method allows a single CNN to simultaneously predict

multiple bounding boxes and class probabilities, thereby optimizing detection performance directly on full images.

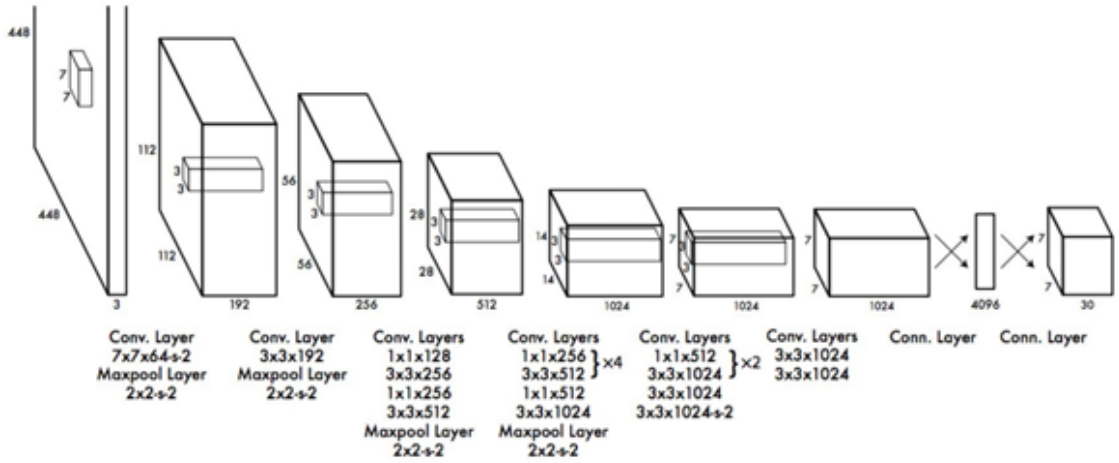



Figure 2.1: YOLO Architecture

2. **OpenCV (Open Source Computer Vision Library)**:

   OpenCV offers an extensive array of tools for analyzing images and videos, including face detection capabilities. Its arsenal includes pre-trained models such as Haar cascades and deep learning-based methods, making it highly adaptable for pinpointing faces within videos. Often integrated with other algorithms for face recognition and identification, OpenCV stands as a versatile solution in the realm of computer vision.

3. **MTCNN (Multi-task Cascaded Convolutional Networks)**:

   MTCNN stands as a remarkable deep learning-based face detection algorithm meticulously crafted for both efficiency and precision in identifying faces. Its prowess lies in its ability to adeptly manage faces of diverse scales and orientations, making it an invaluable tool for scrutinizing video content across a spectrum of conditions.

| ***Model*** | ***Accuracy*** |
|---|---|
| YOLOv5-Face | 99-97% |
| MTCNN | 97-16% |

Table 2.1: Algorithms Comparison

4. **DeepFace**:

   DeepFace is a renowned deep learning framework focused on face recognition and verification tasks. While its main purpose lies in recognizing and verifying faces rather than detecting them, it can be effectively combined with face detection algorithms to identify and authenticate faces within video streams. Notably praised for its exceptional accuracy in recognizing individuals, DeepFace stands out in the field of facial recognition technology.

5. **Viola-Jones (Haar Cascade)**:

   The Viola-Jones face detection method, commonly implemented using Haar cascades, stands as a classic yet efficient technique for identifying faces. Renowned for its lightweight nature, it proves ideal for real-time applications.

## 2.3 Related Works

| Title | Author | Methodology |
|---|---|---|
| Realtime Intrusion Detection System Using OpenCV | Akula Surya Teja, Ginni Chandra Mohini, Dannana Dhanunjay, Dr. P M Manohar | This paper presents a real-time intrusion detection system that uses OpenCV for surveillance in restricted and confidential areas. Face recognition and detection are employed to alert when an intruder or unauthorized person enters the area. The system utilizes the OpenCV python library along with various algorithms for abstracting facial features and processing the input dataset. [1] |
| Implementation of Low Cost IoT Based Intruder Detection System by Face Recognition using Machine Learning | G. Mallikharjuna Rao, Haseena Palle, Pragna Dasari, Shivani Jannaikode, Dr. P M Manohar | This paper presents an IoT-based intrusion detection system that combines machine learning for facial recognition. It uses a PIR sensor to trigger a USB camera, capturing the individual's image, which is then processed for facial detection and recognition using machine learning algorithms and OpenCV. [2] |
| Analysis of Recent Trends in Face Recognition Systems | K. S. Krishnendu | This study explores various face recognition systems, analyzing RPRV, LWKPCA, SVM Model, LTrP based SPM, and a deep learning framework for CCTV image recognition. Through meticulous analysis and performance evaluations, we aim to identify the most effective approaches to guide future development.[3] |
| A Real-Time Framework for Human Face Detection and Recognition in CCTV Images | Rehmat Ullah, Hassan Hayat, Afsah Abid Siddiqui, Uzma Abid Siddiqui, Jebran Khan, Farman Ullah, Shoaib Hassan, Laiq Hasan | This paper presents a real-time framework for detecting and recognizing human faces in CCTV images. The system employs a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches. [4] |
| Face Recognition based Surveillance System Using FaceNet and MTCNN on Jetson TX2 | Edwin Jose, Greeshma M, Mithun Haridas T. P | This paper presents a surveillance system that uses FaceNet and MTCNN for face recognition. The system tracks the subject with the camera ID/location together with the timestamp and logs his presence in the database, using multiple camera installation. [5] |
| Transfer Learning with Deep CNNs for Gender Recognition and Age Estimation | Philip Smith, Cuixian Chen | This study utilizes transfer learning with deep CNNs for gender recognition and age estimation from images. The authors employ pretrained models like VGG19 and VGGFace, experimenting with design changes and training parameters to enhance prediction accuracy. They compare techniques such as input standardization, data augmentation, and label distribution age encoding. [6] |

| Title | Author | Methodology |
|---|---|---|
| Visual Sentinel: Data Analytics for Missing Subject Identification | Loorthu Infenda, Franz Cardoz, Daffril Cleetus, Ajinkya Deshpande | The system utilizes data analytics to detect missing subjects. However, a key challenge lies in addressing missingness within visual analytics, which can affect human decision-making. Acknowledging and addressing missing data is crucial for effective handling in this context. [7] |
| Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks | Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Yu Qiao | This paper presents a system that uses multitask cascaded convolutional networks for joint face detection and alignment. The system uses a cascaded structure with three stages of carefully designed deep convolutional networks that predict face and landmark location in a coarse-to-fine manner. [8] |
| FaceNet: A unified embedding for face recognition and clustering | F. Schroff, D. Kalenichenko, J. Philbin | This paper presents a system called FaceNet that directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. Once this space has been produced, tasks such as face recognition, verification and clustering can be easily implemented using standard techniques with FaceNet embeddings as feature vectors. [9] |
| Efficient System for Criminal Face Detection Technique on Innovative Facial Features To Improve Accuracy Using LBPH In Comparison With CNN | T. Sanjay, W. Deva Priya | The system addresses real-time processing challenges and offers insights into its implementation in surveillance scenarios. However, potential concerns include the computational demands of real-time systems and the accuracy of the LBPH algorithm under varying lighting conditions and angles. [10] |
| Implementation of Facial Recognition for Home Security Systems | R. Menaka, N. Archana, R. Dhanagopal, R. Ramesh | This paper explores advanced facial recognition algorithms for improved home security. It discusses integrating these techniques into existing systems to enhance reliability. Key challenges addressed include achieving high accuracy and dealing with various lighting conditions. [11] |
| Intrusion Detection System using Raspberry Pi and Telegram Integration | R. Prakash, P. Chithaluru | The project involves using a Raspberry Pi as a controller in a home setting. It integrates a Passive Infrared sensor, a webcam, and an LED with the Raspberry Pi. The system processes data and utilizes a Telegram bot API to send instant notifications and images of detected intruders. However, the paper notes a challenge in achieving high accuracy in real-time detection and instant alerts, reporting an overall accuracy rate of 84%. There's room for improvement in correctly identifying unknown intruders. [12] |

| Title | Author | Methodology |
|---|---|---|
| CCTV Intelligent Surveillance on Intruder Detection | Kajenthani Kanthaseelan, Paskaran Pirashaanthan, Jasmin Jelaxshana A.A.P, Akshaya Sivaramakrishnan, Kavinga Yapa Abeywardena, Tharika Munasinghe | The methodology involves building a dataset from daytime CCTV footage, transforming face images into vectors, and splitting the data into training and testing sets. Researchers implement a face detection system using the Haar Cascade classifier model capable of detecting multiple faces. The paper addresses challenges in creating a comprehensive dataset for intruder detection and emphasizes the need for an automated detection system to enhance physical security. Additionally, it highlights concerns about ensuring children's safety and providing alerts for busy parents. [15] |
| Enhanced Missing Object Detection System using YOLO | R. Menaka, N. Archana, R. Dhanagopal, R. Ramesh | The paper enhances the YOLO algorithm for missing object detection by integrating a Bidirectional Feature Pyramid Network (BiFPN) into YOLOv5. This addresses challenges of multi-scale and multi-level feature fusion, improving detection for objects of different sizes. It also discusses handling low-resolution content and overlapping objects, which impact detection accuracy. [16] |
| ArcFace: Additive Angular Margin Loss for Deep Face Recognition | Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, Stefanos Zafeiriou | The paper introduces ArcFace to enhance face recognition model discriminative power by adding an angular margin to the softmax loss function. This improves class separability by maximizing the decision boundary in angular space. Addressing susceptibility to label noise, the paper proposes sub-center ArcFace, which handles clean and noisy data separately for improved real-world performance. Extensive experimental validation across face recognition benchmarks is necessary to confirm ArcFace's effectiveness. [17] |

Table 2.2: Literature Review

# Chapter 3

# Proposed System

## 3.1 Problem Statement

Traditional CCTV surveillance systems in private spaces, notably warehouses, encounter significant obstacles in effective monitoring and security due to their dependence on human operators for footage analysis. The manual nature of this approach leads to time-consuming and less reliable processes, exacerbating the difficulty of distinguishing between authorized and unauthorized individuals and detecting potential intruders. With an increasing demand for more advanced security solutions, there is an urgent need to address these limitations. Our project focuses on enhancing CCTV systems by introducing real-time face recognition-based surveillance, utilizing machine learning and advanced technologies. The specific aim is to automate recognition processes, providing a timely and accurate identification of individuals, and promptly notifying security personnel of any unauthorized access.

## 3.2 Scope

The project's primary objective is to significantly enhance security measures through the development and implementation of an advanced surveillance system. This system will employ cutting-edge technologies to achieve real-time intrusion detection, ensuring the safety and integrity of secured areas. Additionally, emphasis will be placed on user-friendliness and performance monitoring, aiming to provide security personnel and administrators with intuitive tools for efficient operation and continuous evaluation of system effectiveness. Integration of cloud computing and artificial intelligence will empower the system to continuously evolve and adapt, maintaining a proactive stance against emerging security threats.

## 3.3 Proposed System

The objective of this project is to develop an advanced system capable of autonomously detecting unauthorized human presence within predefined zones. The primary goal is to equip administrators and security professionals with early alerts, enabling swift responses to security breaches. This proactive approach ensures the safety and integrity of restricted areas, fostering a secure environment for both property and individuals. By harnessing cutting-edge technology, the project seeks to enhance security and protection measures within these defined regions, ultimately improving the overall safety and security of these areas.

Project proposes enhancing CCTV systems with real-time face recognition-based surveillance. Leveraging the robust capabilities of the Arcface algorithm for accurate face recognition and SCRFD alongside YOLO for efficient face detection, the system aims to automate the recognition process. Upon detecting an intruder, the system will promptly notify security personnel through email, messages, and calls, significantly enhancing the efficiency and reliability of security measures in private spaces.

### 3.3.1 Approach

#### 3.3.1.1 Face Detection

Face detection is the process of identifying and locating human faces within images or video frames. We implemented the following face detection algorithms:

1. **YOLOv5-Face:** YOLOv5-Face stands out as a specialized iteration of the YOLOv5 object detection framework tailored exclusively for identifying human faces within images or video frames. While YOLOv5 possesses broad applicability in detecting diverse objects, YOLOv5-Face concentrates its efforts on refining face detection tasks. Much like its predecessor, YOLOv5-Face operates by segmenting input images into a grid, predicting bounding boxes, and class probabilities for objects within each grid cell. Nonetheless, the architecture and training regimen of YOLOv5-Face undergo fine-tuning specifically to enhance face detection accuracy and efficiency. Leveraging cutting-edge deep learning techniques including anchor boxes, focal loss, and feature pyramid networks, YOLOv5-Face demonstrates exceptional performance in face detection across varied datasets and scenarios. Its rapid and precise face detection capabilities render it indispensable for a spectrum of applications ranging from surveillance systems to facial recognition technology and augmented reality endeavors, where dependable face detection serves as a cornerstone.

2. **SCRFD:** SCRFD (Single-Shot Scale-Aware Face Detector) stands out as a recent breakthrough in face detection algorithms, engineered to efficiently identify faces across a broad spectrum of scales. Unlike conventional approaches dependent on multi-scale image pyramids or sliding windows, SCRFD adopts a single-shot methodology, significantly enhancing image processing speed. Through the integration of scale-aware anchor assignment and scale compensation modules, SCRFD adeptly manages faces of diverse sizes while maintaining both

precision and rapidity. This versatility positions SCRFD as an ideal solution for real-time applications demanding swift and accurate face detection under varying conditions, such as video surveillance, facial recognition systems, and photography software. Its seamless blend of speed and precision marks SCRFD as a substantial advancement in the domain of face detection algorithms.

3. **Cascade Classifier:** Cascade Classifier, pioneered by Viola and Jones in 2001, stands as a cornerstone in object detection methodologies. This technique leverages machine learning principles, employing a cascade function trained on extensive sets of positive (object) and negative (non-object) images. Particularly acclaimed for its efficacy in face detection, the classifier mandates a robust corpus of face and non-face images for its training phase. Through the extraction of features like Haar-like features, discernment between objects and non-objects is facilitated by scrutinizing pixel intensity variations. This approach optimally sieves through image data, efficiently identifying potential object regions.


Figure 3.1: Haar Cascade Frontal Face

#### 3.3.1.2 Face Identification

Face identification is the process of recognizing human faces within images or video frames. We implemented the following face detection algorithms:

1. **Arcface:** Arcface, short for ArcFace Loss, is a highly effective face recognition algorithm designed to generate discriminative feature embeddings for face identification tasks. At its core, Arcface utilizes

a deep convolutional neural network (CNN) architecture, typically based on ResNet or MobileNet, to extract facial features robustly from input images. What sets Arcface apart is its novel loss function, the ArcFace Loss, which introduces angular margin constraints to the embedding space during training. This margin-based approach encourages enhanced discrimination between different classes (faces) while simultaneously enforcing intra-class compactness, resulting in more discriminative and separable feature representations. By explicitly optimizing the angular relationship between feature vectors corresponding to different individuals, Arcface achieves state-of-the-art performance in face identification tasks, even in scenarios with large pose variations, occlusions, and illumination changes. Its superior accuracy, coupled with relatively efficient computation, has made Arcface a prominent choice in various applications, including security systems, access control, and identity verification technologies.

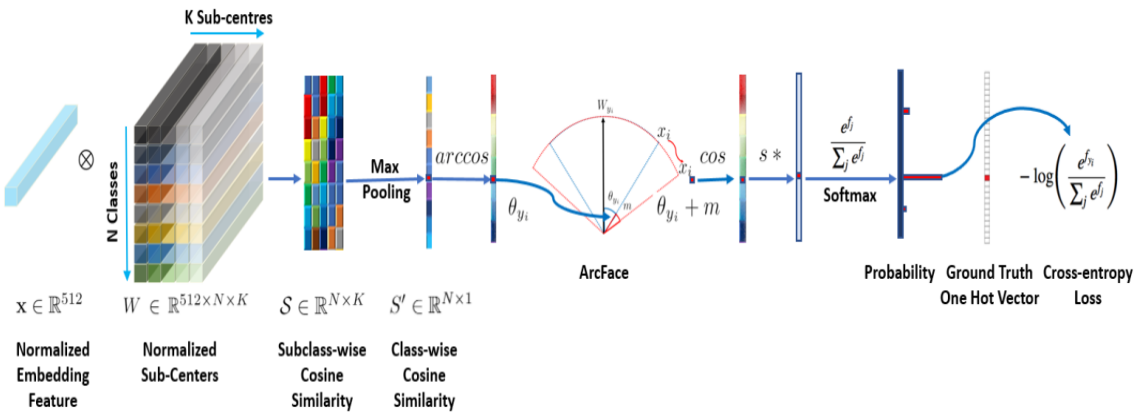



Figure 3.2: Arcface Architecture

#### 3.3.1.3 Human Intrusion Detection

1. **YOLOv8:** YOLOv8 emerges as a robust solution for intrusion detection, particularly within CCTV surveillance systems. Its innovative design integrates features that elevate its efficacy as an object detector. One notable advancement is the adoption of an anchor-free detection head, a departure from the anchor box method utilized in earlier YOLO iterations. This shift streamlines the detection process while bolstering accuracy, enhancing its capability to identify intrusions effectively. In the domain of human intrusion detection, YOLOv8 proves its mettle by accurately identifying and tracking various objects within the surveillance camera's field of view, including

potential intruders. This comprehensive approach amalgamates object detection, face detection, and intrusion detection tasks, facilitating thorough monitoring and analysis. Post-training, the software is equipped with fundamental functionalities for detecting and tracking unauthorized human intrusions, contributing significantly to the security and surveillance landscape.

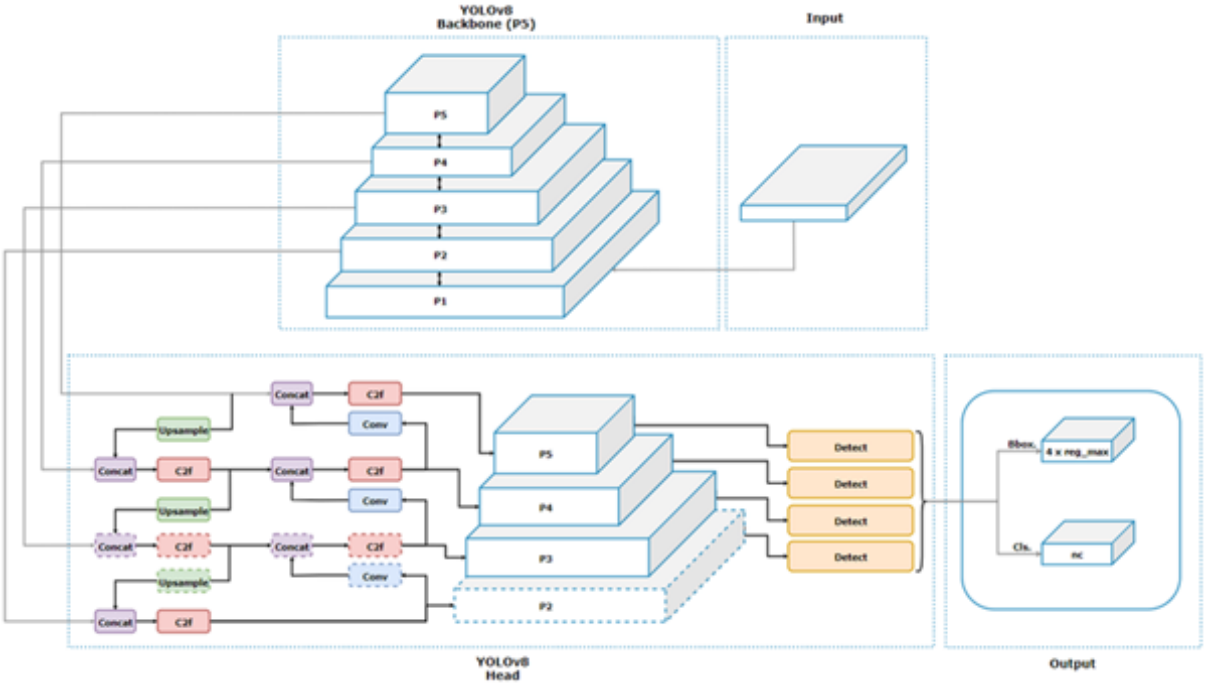



Figure 3.3: YOLOv8 Architecture

# Chapter 4

# Design of the System

## 4.1 Requirement Engineering

### 4.1.1 Requirement Elicitation

| ID | Requirement Description | Priority | Acceptance Criteria |
|---|---|---|---|
| REQ-001 | User Authentication | High | The system should allow authorized users to log in and access the surveillance system, distinguishing between various user roles, such as administrators, security personnel, and system managers. |
| REQ-002 | Video Surveillance | High | Users should be able to access and monitor live video feeds and recorded data from surveillance devices deployed at different locations. |
| REQ-003 | Surveillance Analysis | High | Recorded surveillance data should be analyzed and used by the system's AI algorithms to differentiate between authorized and unauthorized persons. |
| REQ-004 | Real-time Alerts | High | Users should have the capability to receive real-time alerts and notifications when the system detects unusual activities or potential human intrusions. |
| REQ-005 | Intrusion Detection | Medium | The recommendation system should provide continuous feedback and optimization of intrusion detection algorithms based on historical security events and system performance. |
| REQ-006 | Security Response | High | The system should facilitate communication with external security response teams or authorities, enabling users to escalate and respond to security incidents effectively. |
| REQ-007 | User Interface | High | The system should offer an intuitive user interface for managing and configuring surveillance devices, access control, and user permissions. It should also enable users to review historical security events and generate reports. |

Table 4.1: Requirement Elicitation Techniques

### 4.1.2 Software Lifecycle Model

The Agile model is our chosen Software Lifecycle Model. It allows for iterative development, where modules are developed in phases and integrated at the end of the lifecycle. The decision to use the Agile model is based on several factors:

1. **Adaptability to Changing Requirements:** Agile is known for its responsiveness to evolving project requirements. Its flexibility enables teams to accommodate shifting customer needs and market dynamics, particularly beneficial in fast-paced industries.
2. **Early Issue Detection and Resolution:** Agile methodologies emphasize continuous testing, enabling teams to identify and rectify issues early in the development process. This proactive approach minimizes the occurrence of significant problems later on, thus saving time and resources.
3. **Iterative and Incremental Development:** Agile breaks down projects into manageable iterations or sprints. Each iteration concludes with a potentially shippable product increment. This iterative approach facilitates ongoing improvement and allows for adjustments as needed.
4. **Improved Team Collaboration:** Agile encourages the formation of cross-functional teams comprising members from diverse backgrounds. This promotes collaboration and knowledge exchange, enhancing problem-solving capabilities and decision-making processes.
5. **Continuous Learning and Improvement:** Agile fosters a culture of reflection and regular process enhancements within teams. This commitment to learning and adaptation contributes to the long-term success of projects.
6. **Customization for Specific Projects:** Agile methodologies can be tailored to meet the unique requirements and constraints of individual projects. This adaptability makes it suitable for a wide array of software development endeavors.

### 4.1.3 Requirement Analysis

#### 4.1.3.1 Use Case Diagram

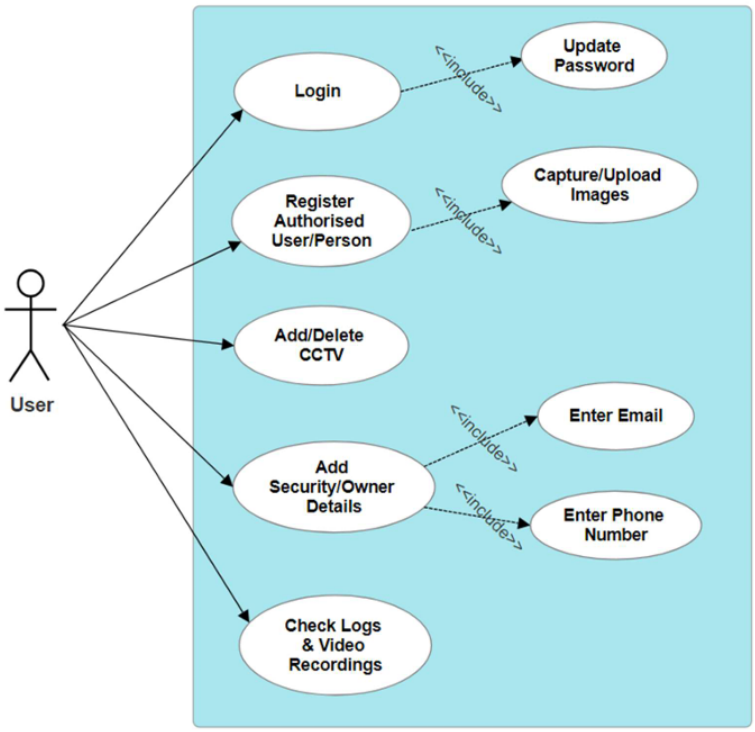



Figure 4.1: Use Case Diagram for Surveillance System

The use case diagram represents the interactions between a user and a security system.

- Login: The user can log into the system.
- Update Password: The user has the ability to change their password.
- Register Authorized User/Person: This allows the user to add authorized individuals to the system.
- Capture/Upload Images: Users can capture or upload images, likely for identification or verification purposes.
- Add/Delete CCTV: The user can manage CCTV devices by adding or removing them.
- Enter Email and Phone Number: Users are required to enter their contact details.
- Check Logs & Video Recordings: The user can review system logs and video recordings for monitoring purposes.

#### 4.1.3.2 Architecture Diagram

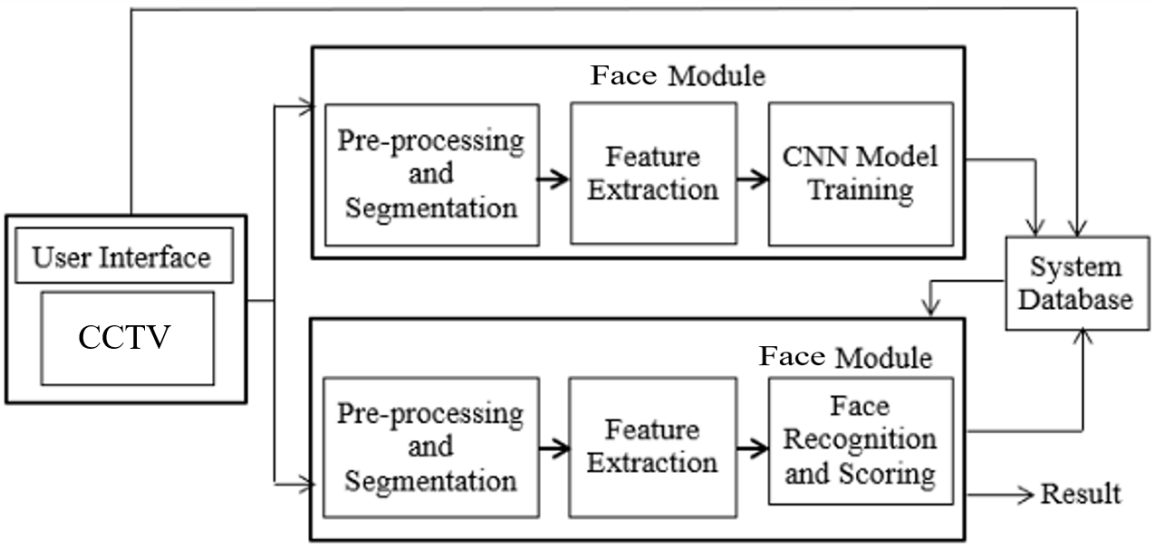



Figure 4.2: System Architecture Diagram

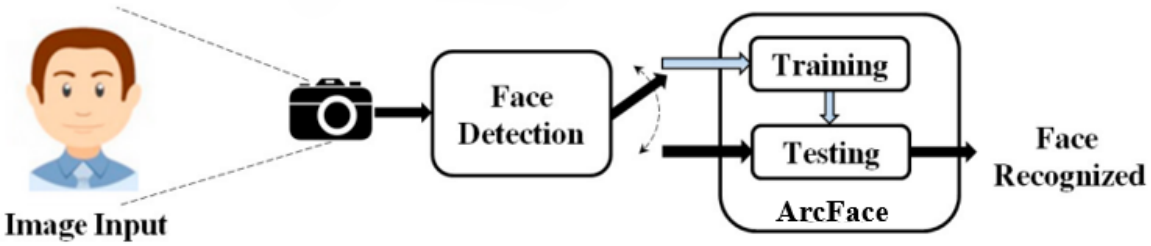



Figure 4.3: Image Capturing Architecture Diagram

#### 4.1.3.3 Cost Analysis

A cost analysis for a surveillance system incorporating human intrusion detection involves evaluating various expenses associated with its development, implementation, and maintenance. Several key cost components should be taken into account:

- **CCTV Cameras:** The system relies on CCTV footage for object detection, necessitating an investment in high-resolution CCTV cameras.
- **Training & Testing:** Training the neural network requires high-performance computers equipped with Graphic Processing Units (GPUs). Similarly, for testing or running the neural network on video frames, GPUs are essential, especially for achieving higher frame rates.

#### 4.1.3.4 Software Requirement

- Operating System : Linux
- Python
- Development Frameworks and Libraries:
  - OpenCV
  - Flask
  - PyTorch
- Database Management: MySQL
- IDE(Integrated Development Environment): Visual Studio Code, PyCharm
- Jupyter Notebook for coding and experimentation.

#### 4.1.3.5 Hardware Requirement

- Multi-core CPU or GPU
- Memory (RAM): 8 GB or more
- Webcam or Camera
- Internet connection

## 4.2 System Architecture

### 4.2.1 UI/UX diagram

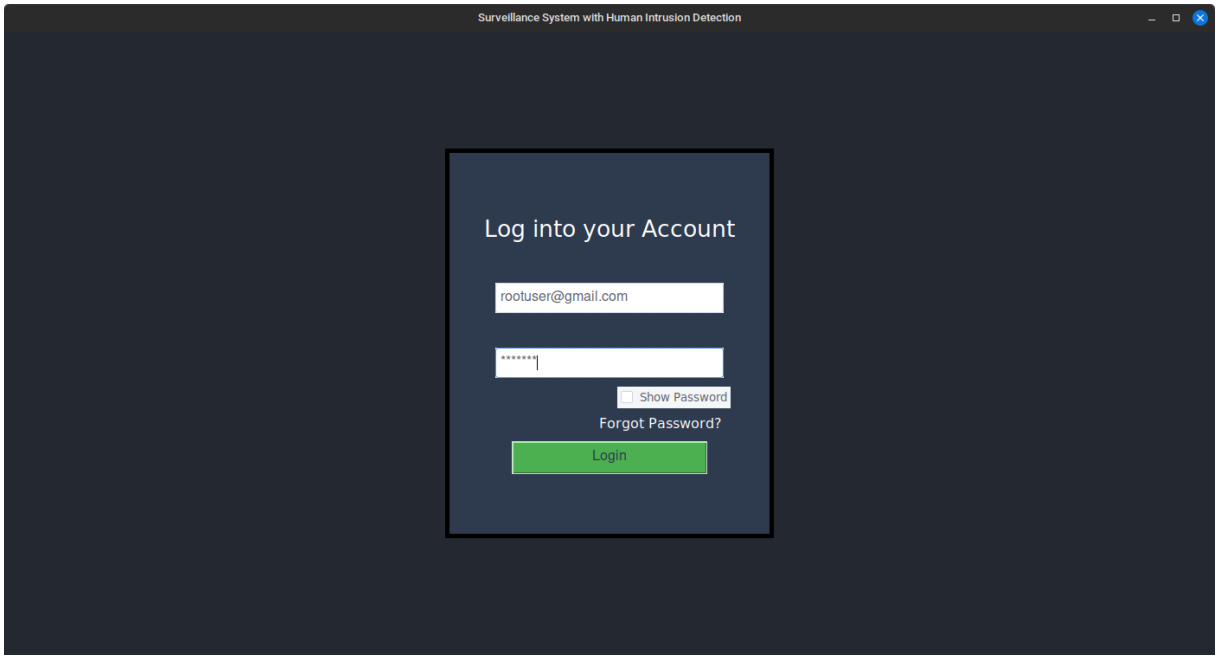



Figure 4.4: Login Page of System

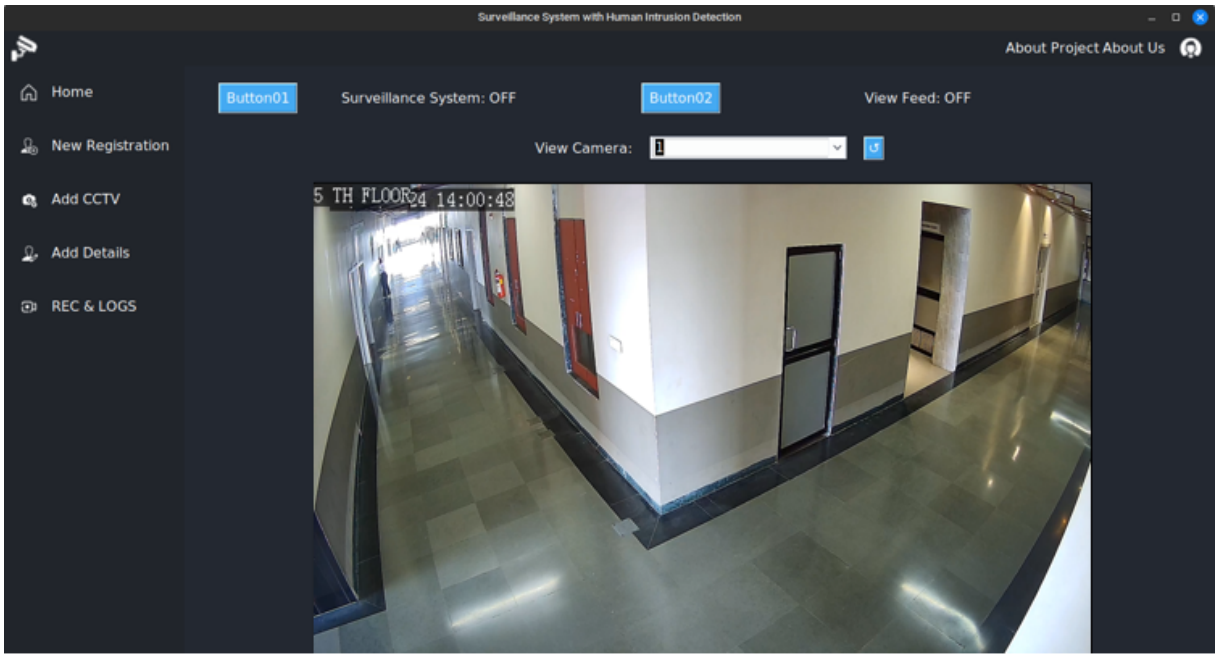



Figure 4.5: Home Page of System

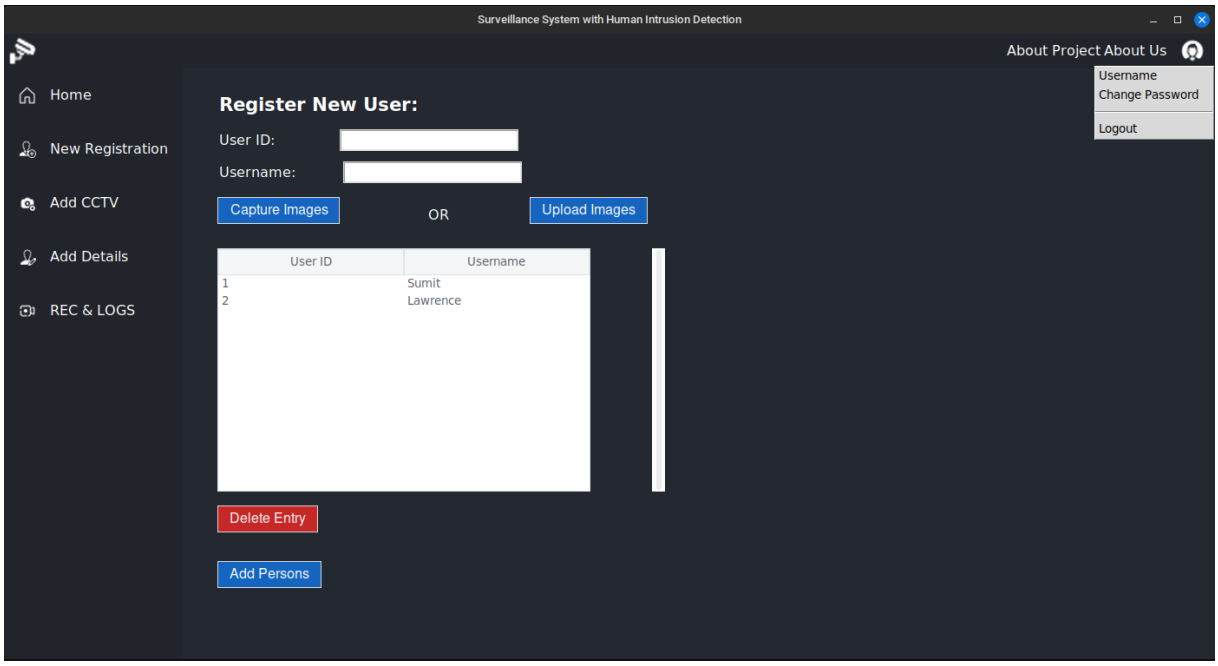



Figure 4.6: Authorised User Registration

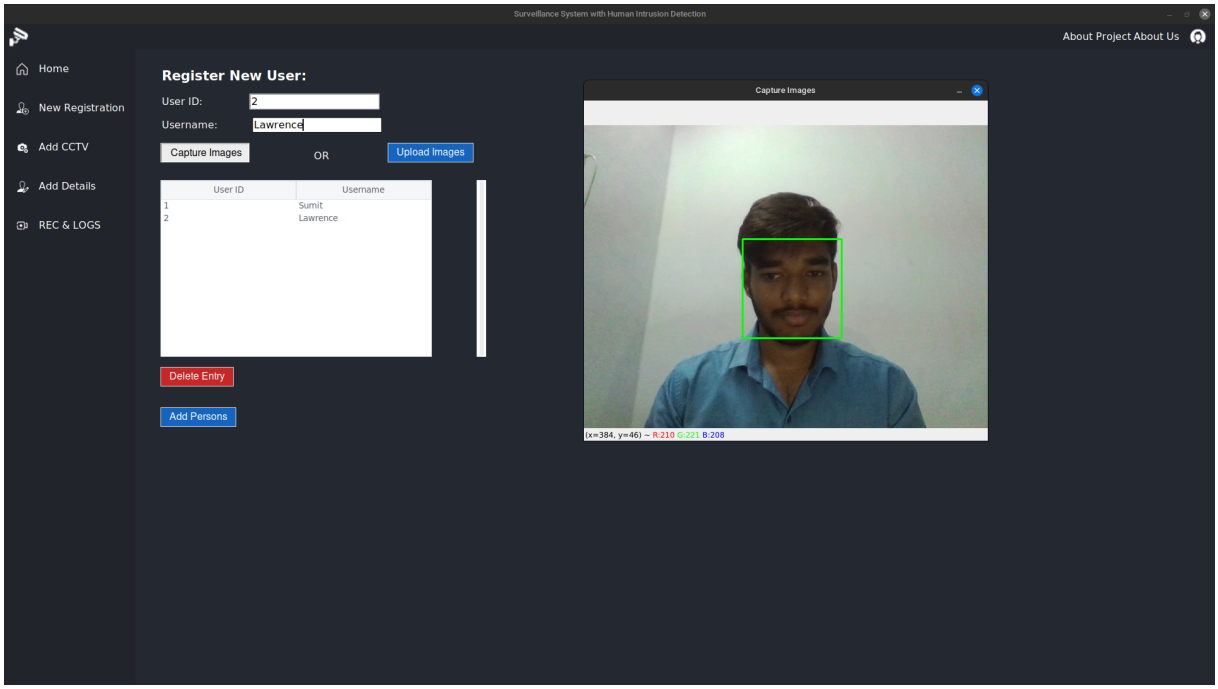



Figure 4.7: Authorised User Face Capturing


Figure 4.8: Add CCTV Ip address

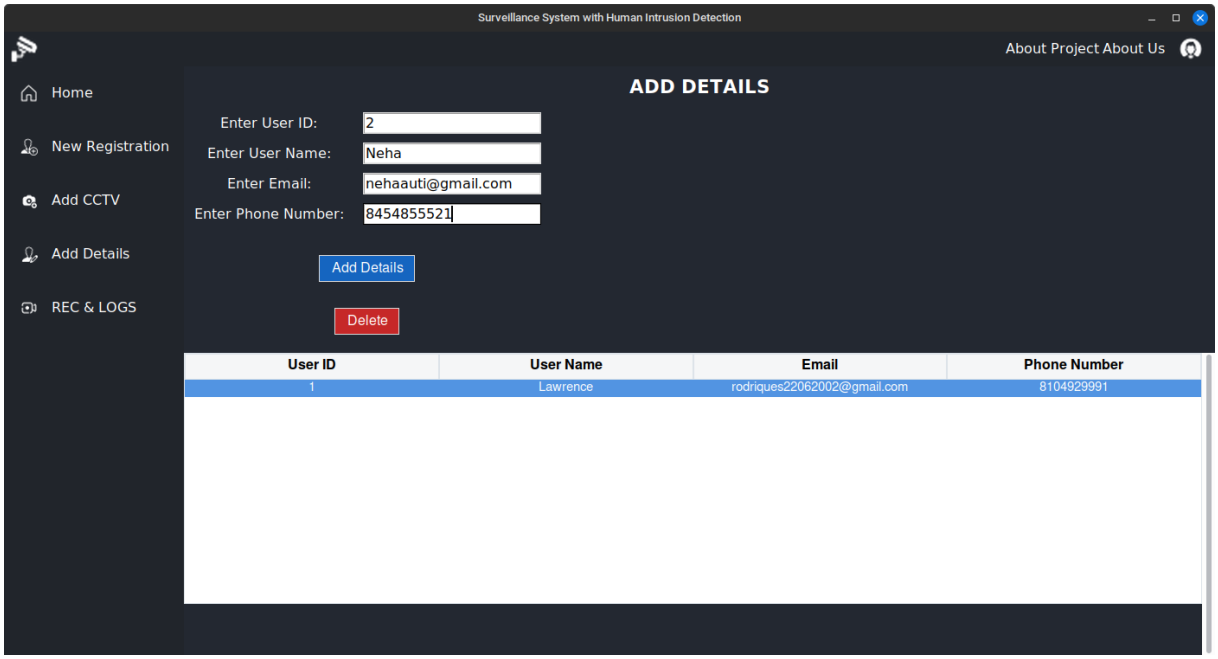



Figure 4.9: Add Security/Owner Details

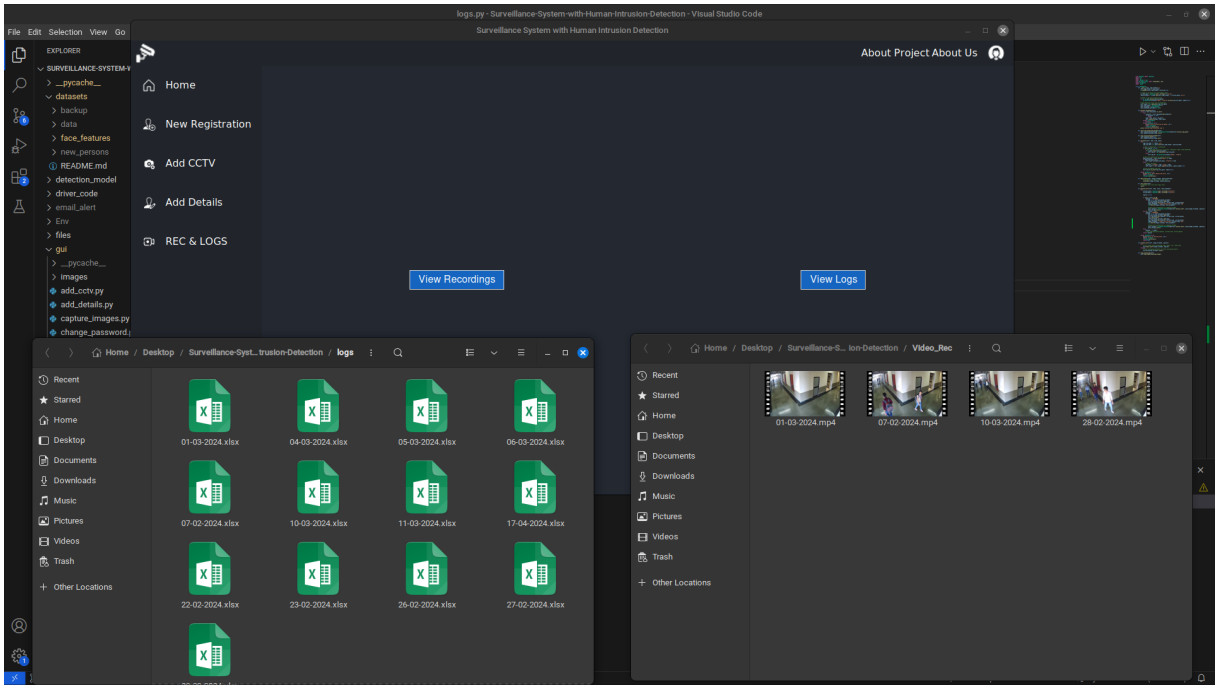



Figure 4.10: Recordings & Logs

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | **Name** | **Time** | | | | |
| 2 | 2_Lawrence | 23:57:35 | | | | |
| 3 | INTRUDER | 23:57:26 | | | | |
| 4 | 1_Sumit | 23:57:21 | | | | |
| 5 | | | | | | |
| 6 | | | | | | |
| 7 | | | | | | |
| 8 | | | | | | |
| 9 | | | | | | |
| 10 | | | | | | |
| 11 | | | | | | |
| 12 | | | | | | |
| 13 | | | | | | |
| 14 | | | | | | |
| 15 | | | | | | |

Figure 4.11: Logs Entry

### 4.2.2 Block Diagram

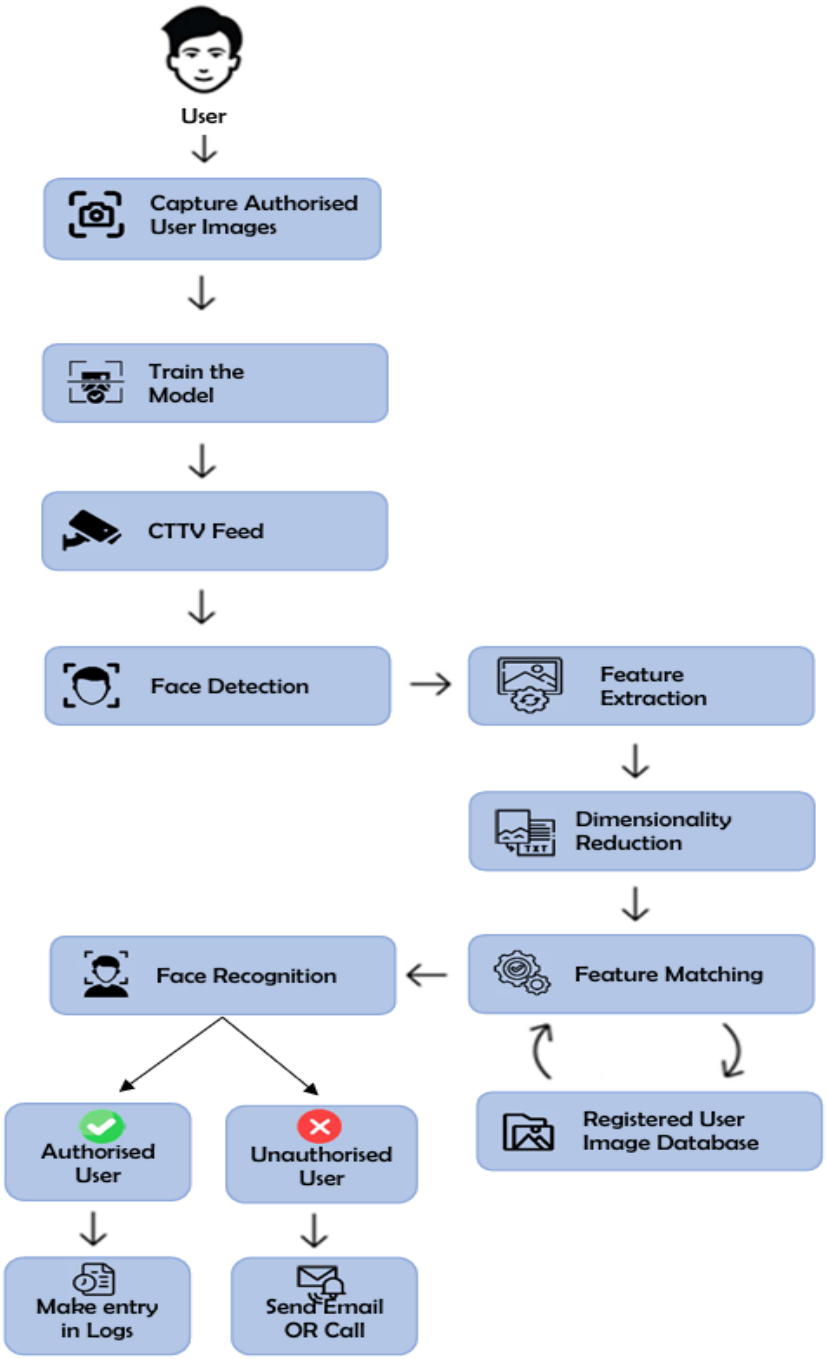



Figure 4.12: Block Diagram of the Surveillance System

The block diagram outlines a facial recognition system designed for security applications. It begins by capturing images of authorized users and training the model to identify their features. When the CCTV feed detects a person, the system extracts facial characteristics and matches them against the registered user database. If the individual is recognized as authorized, an entry is recorded; however, if unauthorized, an alert is promptly dispatched via email or phone call.

# Chapter 5

# Result and Discussion

### 5.0.1 Screenshots of the System


Figure 5.1: Face Detection through CCTV

Face detection using SCRFD and YOLOv5-face on the CCTV footage has proven highly effective, reliably identifying faces within a specific distance range.

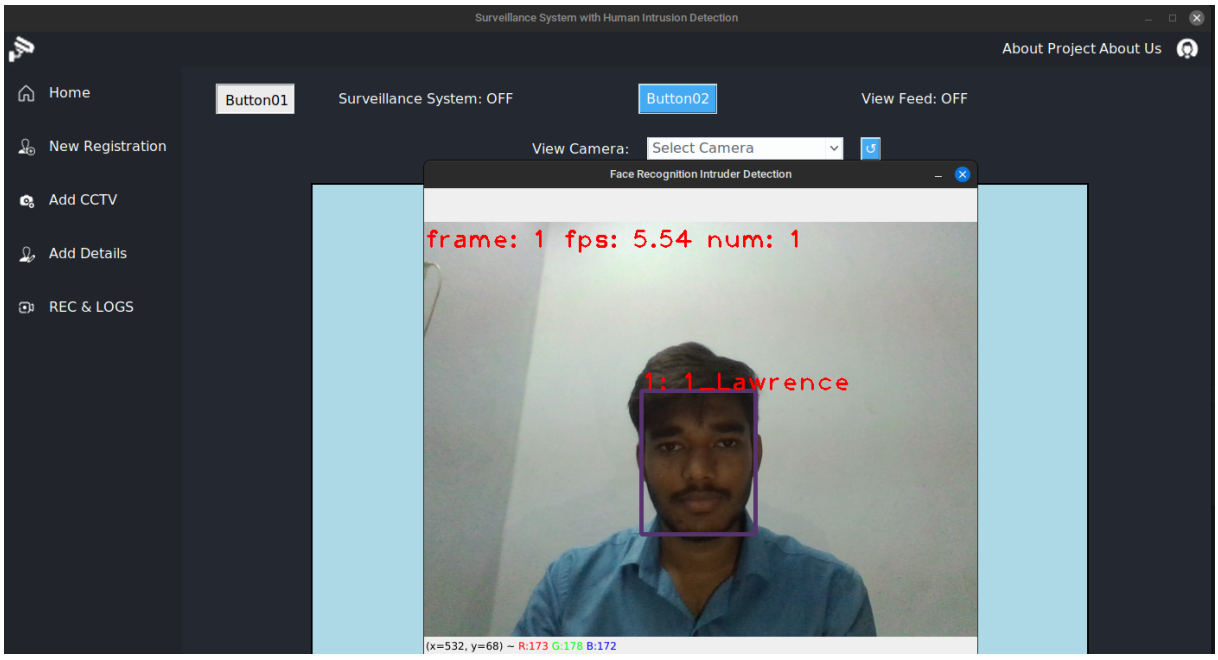



Figure 5.2: Face Recognition through Webcam

Arcface's face recognition system effectively detects intruders through a webcam, promptly sending email or call alerts upon detection.


Figure 5.3: Face Recognition through CCTV

Utilizing CCTV footage, the face recognition and intruder detection system seamlessly identifies unauthorized individuals, triggering immediate email and call notifications through integration with the Twilio API. This robust solution ensures swift response to security breaches, enhancing overall surveillance effectiveness.

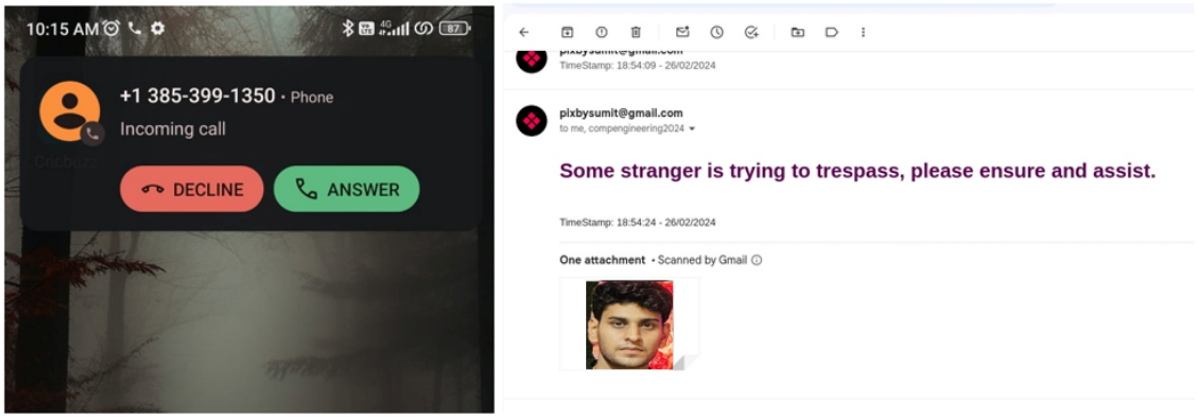



Figure 5.4: Email & Call Alerts

The system efficiently delivers email & call alerts to the registered owner or user, ensuring prompt notification of any detected intrusions or security breaches. This personalized approach enhances security responsiveness and facilitates timely action to mitigate potential threats.

## 5.1 Sample Code

```python
class NewRegistrationSection(tk.Frame):
    def __init__(self, master):
        super().__init__(master, bg="#232831", padx=40, pady=20)
        self.user_id_var = tk.StringVar()
        self.username_var = tk.StringVar()
        self.data_entries = []
        self.initialize_connection()
        self.create_registration_widgets()
        self.create_table()
        self.create_delete_button()
        self.update_table()

    def initialize_connection(self):
        self.database = DatabaseHandler()
        self.conn, self.cursor = self.database.initialize_connection()

    def create_registration_widgets(self):
        title_label = tk.Label(self, text="Register New User:", font=('Century Gothic', 16, 'bold'), bg="#232831", fg="white", pady=8)
        title_label.grid(row=0, column=0, columnspan=3, sticky="w")

        user_id_label = tk.Label(self, text="User ID:", font=('Century Gothic', 12), bg="#232831", fg="white")
        user_id_label.grid(row=1, column=0, pady=6, sticky="w")
        user_id_entry = tk.Entry(self, textvariable=self.user_id_var, font=('Century Gothic', 12))
        user_id_entry.grid(row=1, column=1, pady=6, sticky="w")

        username_label = tk.Label(self, text="Username:", font=('Century Gothic', 12), bg="#232831", fg="white")
        username_label.grid(row=2, column=0, pady=6, sticky="w")
        username_entry = tk.Entry(self, textvariable=self.username_var, font=('Century Gothic', 12))
        username_entry.grid(row=2, column=1, pady=6, padx=4, sticky="w")

        capture_button = tk.Button(
            self,
            text="Capture Images",
            command=self.capture_images,
            bg="#1565C0",
            fg="white",
            font=("Helvetica", 12),
            relief=tk.FLAT,
        )
        capture_button.grid(row=3, column=0, pady=(10, 20), sticky="w")
```

Figure 5.5: Code for Capturing User Images

```python
def add_persons(backup_dir, add_persons_dir, faces_save_dir, features_path):
    """
    Add a new person to the face recognition database.

    Args:
        backup_dir (str): Directory to save backup data.
        add_persons_dir (str): Directory containing images of the new person.
        faces_save_dir (str): Directory to save the extracted faces.
        features_path (str): Path to save face features.
    """
    # Initialize lists to store names and features of added images
    images_name = []
    images_emb = []

    # Read the folder with images of the new person, extract faces, and save them
    for name_person in os.listdir(add_persons_dir):
        person_image_path = os.path.join(add_persons_dir, name_person)

        # Create a directory to save the faces of the person
        person_face_path = os.path.join(faces_save_dir, name_person)
        os.makedirs(person_face_path, exist_ok=True)

        for image_name in os.listdir(person_image_path):
            if image_name.endswith(("png", "jpg", "jpeg")):
                input_image = cv2.imread(os.path.join(person_image_path, image_name))

                # Detect faces and landmarks using the face detector
                bboxes, landmarks = detector.detect(image=input_image)

                # Extract faces
                for i in range(len(bboxes)):
                    # Get the number of files in the person's path
                    number_files = len(os.listdir(person_face_path))

                    # Get the location of the face
                    x1, y1, x2, y2, score = bboxes[i]

                    # Extract the face from the image
                    face_image = input_image[y1:y2, x1:x2]

                    # Path to save the face
                    path_save_face = os.path.join(person_face_path, f"{number_files}.jpg")

                    # Save the face to the database
                    cv2.imwrite(path_save_face, face_image)

                    # Extract features from the face
                    images_emb.append(get_feature(face_image=face_image))
                    images_name.append(name_person)
```

Figure 5.6: Code for Adding Authorised User

```
import mysql.connector

class DatabaseHandler:
    def __init__(self):
        self.conn, self.cursor = self.initialize_connection()

    def initialize_connection(self):
        conn = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="users"
        )

        cursor = conn.cursor()
        self.create_table(cursor)

        return conn, cursor

    def create_table(self, cursor):
        cursor.execute("SHOW TABLES")
        temp = cursor.fetchall()
        tables = [item[0] for item in temp]

        if "users" not in tables:
            cursor.execute("""CREATE TABLE IF NOT EXISTS users (
                id INT AUTO_INCREMENT PRIMARY KEY,
                password VARCHAR(30),
                email VARCHAR(100) UNIQUE
            )""")

    def create_add_details_table(self, cursor):
        cursor.execute("SHOW TABLES")
        temp = cursor.fetchall()
        tables = [item[0] for item in temp]

        if "add_details" not in tables:
            cursor.execute("""CREATE TABLE IF NOT EXISTS add_details (
                user_id INT PRIMARY KEY,
                user_name VARCHAR(255),
                email VARCHAR(100),
                phone_number VARCHAR(20)
            )""")

    def create_table_new(self):
        self.cursor.execute("""
            CREATE TABLE IF NOT EXISTS newreg (
                user_id INT PRIMARY KEY,
                username VARCHAR(255) NOT NULL
            )
        """)
```

Figure 5.7: MySQL database Code

```python
class FaceRecognizer:
    def __init__(self):
        self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        self.detector = SCRFD(model_file="driver_code/face_detection/scrfd/weights/scrfd_2.5g_bnkps.onnx")
        self.recognizer = iresnet_inference(
            model_name="r100", path="driver_code/face_recognitions/arcface/weights/arcface_r100.pth", device=self.device
        )
        self.images_names, self.images_embs = read_features(feature_path="datasets/face_features/feature")
        self.id_face_mapping = {}
        self.data_mapping = {
            "raw_image": None,
            "tracking_ids": [],
            "detection_bboxes": [],
            "detection_landmarks": [],
            "tracking_bboxes": [],
        }
        self.exit_threads = False
        self.logs_folder = "logs"
        os.makedirs(self.logs_folder, exist_ok=True)
        self.logs_handler = LogsHandler(logs_folder=self.logs_folder)

    def load_config(self, file_name):
        with open(file_name, "r") as stream:
            try:
                return yaml.safe_load(stream)
            except yaml.YAMLError as exc:
                print(exc)

    def process_tracking(self, frame, detector, tracker, args, frame_id, fps):
        outputs, img_info, bboxes, landmarks = detector.detect_tracking(image=frame)

        if outputs is not None:
            tracking_tlwhs, tracking_ids, tracking_scores, tracking_bboxes = [], [], [], []

            for t in tracker.update(outputs, [img_info["height"], img_info["width"]], (128, 128)):
                tlwh, tid = t.tlwh, t.track_id
                vertical = tlwh[2] / tlwh[3] > args["aspect_ratio_thresh"]

                if tlwh[2] * tlwh[3] > args["min_box_area"] and not vertical:
                    x1, y1, w, h = tlwh
                    tracking_bboxes.append([x1, y1, x1 + w, y1 + h])
                    tracking_tlwhs.append(tlwh)
                    tracking_ids.append(tid)
                    tracking_scores.append(t.score)

            tracking_image = plot_tracking(img_info["raw_img"], tracking_tlwhs, tracking_ids, names=self.id_face_mapping, frame_id=frame_id + 1, fps=fps)
        else:
            tracking_image = img_info["raw_img"]

        self.data_mapping["raw_image"] = img_info["raw_img"]
        self.data_mapping["detection_bboxes"] = bboxes
        self.data_mapping["detection_landmarks"] = landmarks
        self.data_mapping["tracking_ids"] = tracking_ids
        self.data_mapping["tracking_bboxes"] = tracking_bboxes

        return tracking_image
```

Figure 5.8: Code for Processing Face Recognition

```python
class RecAndLogsSection(tk.Frame):
    def __init__(self, master):
        super().__init__(master, bg="#232831")
        self.recording_path = "/home/lawrence/Desktop/Surveillance-System-with-Human-Intrusion-Detection/Video_Rec"
        self.logs_path = "/home/lawrence/Desktop/Surveillance-System-with-Human-Intrusion-Detection/logs"
        self.create_buttons()

    def create_buttons(self):
        button1 = tk.Button(
            self,
            text="View Recordings",
            command=self.open_recordings_folder,
            bg="#1565C0",
            fg="white",
            font=("Helvetica", 12),
            relief=tk.FLAT,
        )
        button1.grid(row=0, column=0, padx=50)

        button2 = tk.Button(
            self,
            text="View Logs",
            command=self.open_logs_folder,
            bg="#1565C0",
            fg="white",
            font=("Helvetica", 12),
            relief=tk.FLAT,
        )
        button2.grid(row=0, column=1, padx=50)

        self.grid_rowconfigure(0, weight=1)
        self.grid_columnconfigure(0, weight=1)
        self.grid_columnconfigure(1, weight=1)

    def open_recordings_folder(self):
        self.open_folder(self.recording_path)

    def open_logs_folder(self):
        self.open_folder(self.logs_path)
```

Figure 5.9: Code for Recording & Logs

```python
class LogsHandler:
    def __init__(self, logs_folder=""):
        self.logs_folder = logs_folder
        os.makedirs(self.logs_folder, exist_ok=True)

        # Create a new Excel file with today's date
        current_date = datetime.now().strftime("%d-%m-%Y")
        log_file_path = os.path.join(self.logs_folder, f"{current_date}.xlsx")

        # Check if the file already exists
        if not os.path.exists(log_file_path):
            pd.DataFrame(columns=["Name", "Time"]).to_excel(log_file_path, index=False)

        # Read the Excel file into the DataFrame
        self.log_df = pd.read_excel(log_file_path)
        self.stop_event = threading.Event()
        self.log_queue = queue.Queue()
        self.processed_intruders = set()

    def process_log_queue(self):
        while not self.stop_event.is_set():
            try:
                log_entry = self.log_queue.get(timeout=1)
                if log_entry is None:
                    break
                name, time, date = log_entry
                self.log_entry(name, time, date)
            except queue.Empty:
                continue
            except Exception as e:
                print(f"Error processing log queue: {e}")
                import traceback
                traceback.print_exc()
        print("Exiting log processing loop.")

    def start_log_processing_thread(self):
        self.log_processing_thread = threading.Thread(target=self.process_log_queue)
        self.log_processing_thread.start()

    def stop_log_processing_thread(self):
        self.log_queue.put(None)
        self.log_processing_thread.join()
```

Figure 5.10: Code for Logs Entry

```python
class ObjectDetection:
    def __init__(self):
        # default parameters
        self.email_sent = False

        self.model = YOLO("detection_model/yolov8n.pt")

        # visual information
        self.annotator = None
        self.start_time = 0
        self.end_time = 0

        # device information
        self.device = 'cuda' if torch.cuda.is_available() else 'cpu'

        # Set the class of interest (person)
        self.person_class_name = 'person'

    def predict(self, im0):
        results = self.model(im0)
        return results

    def display_fps(self, im0):
        self.end_time = time()
        fps = 1 / np.round(self.end_time - self.start_time, 2)
        text = f'FPS: {int(fps)}'
        text_size = cv2.getTextSize(text, cv2.FONT_HERSHEY_SIMPLEX, 1.0, 2)[0]
        gap = 10
        cv2.rectangle(im0, (20 - gap, 70 - text_size[1] - gap), (20 + text_size[0] + gap, 70 + gap), (255, 255, 255), -1)
        cv2.putText(im0, text, (20, 70), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 0), 2)

    def plot_bboxes(self, results, im0):
        class_ids = []
        self.annotator = Annotator(im0, 3, results[0].names)
        boxes = results[0].boxes.xyxy.cpu()
        clss = results[0].boxes.cls.cpu().tolist()
        names = results[0].names
        for box, cls in zip(boxes, clss):
            if names[int(cls)] == self.person_class_name:  # Only consider person class
                class_ids.append(cls)
                label_text = "Intruder"
                self.annotator.box_label(box, label=label_text, color=colors(int(cls), True))
        return im0, class_ids
```

Figure 5.11: Code for Face Detection

```python
database = DatabaseHandler()
conn, cursor = database.initialize_connection()

current_time = datetime.now().strftime("%H:%M:%S")
def send_email(image_filename, object_detected):
    # Send an email with the image as an attachment
    sender = os.environ['EMAIL_SENDER']
    password = os.environ['EMAIL_PASSWORD']

    query = "SELECT * FROM add_details"
    cursor.execute(query)
    rows = cursor.fetchall()

    cc = [row[2] for row in rows]
    current_time = datetime.now().strftime("%H:%M:%S - %d/%m/%Y")
    receiver = "rodriques22062002@gmail.com"
    subject = "Intruder detected - Please check logs and take action."
    email_body = f"<h1>Some stranger is trying to trespass, please ensure and assist. </h1><br>TimeStamp: {current_time}"

    message = MIMEMultipart()
    message['From'] = sender
    message['To'] = receiver
    message['cc'] = ', '.join(cc)  # Convert the list to a comma-separated string
    message['Subject'] = subject
    message.attach(MIMEText(email_body, 'html'))
    if image_filename:
        # Attach the captured unknown face image
        filename = os.path.basename(image_filename)
        attachment = open(image_filename, "rb").read()

        image = MIMEImage(attachment, name=filename)
        message.attach(image)
    toaddrs = [receiver] + cc

    # SMTP connection
    with s.SMTP("smtp.gmail.com") as connection:
        connection.starttls()
        connection.login(user=sender, password=password)

        connection.sendmail(
            from_addr=sender,
            to_addrs=toaddrs,
            msg=message.as_string(),
        )
    print(f"Email sent: {current_time}")
def call():
    account_sid = os.environ['TWILIO_SID']
    auth_token = os.environ['TWILIO_TOKEN']
    number = os.environ['TWILIO_NUMBER']

    client = Client(account_sid, auth_token)

    query = "SELECT * FROM add_details"
    cursor.execute(query)
    rows = cursor.fetchall()
    recpt = '+91' + str(rows[0][3])

    call = client.calls.create(
                        to=recpt,
                        from_=number,
                        url='https://handler.twilio.com/twiml/EH4ec1a9bb230f23eb949ccd0642754612',
                    )
```

Figure 5.12: Code for Email & Call Alert

```python
class Yolov5Face(object):
    def __init__(self, model_file=None):
        """
        Initialize the Detector class.

        :param model_path: Path to the YOLOv5 model file (default is yolov5n-0.5.pt)
        """
        device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

        self.device = device
        self.model = attempt_load(model_file, map_location=device)

        # Parameters
        self.size_convert = 128  # Size for image conversion
        self.conf_thres = 0.4  # Confidence threshold
        self.iou_thres = 0.5  # Intersection over Union threshold

    def resize_image(self, img0, img_size):
        """
        Resize the input image.

        :param img0: The input image to be resized.
        :param img_size: The desired size for the image.

        :return: The resized and preprocessed image.
        """
        h0, w0 = img0.shape[:2]  # Original height and width
        r = img_size / max(h0, w0)  # Resize image to img_size

        if r != 1:
            interp = cv2.INTER_AREA if r < 1 else cv2.INTER_LINEAR
            img0 = cv2.resize(img0, (int(w0 * r), int(h0 * r)), interpolation=interp)

        imgsz = check_img_size(img_size, s=self.model.stride.max())  # Check img_size
        img = letterbox(img0, new_shape=imgsz)[0]

        img = img[:, :, ::-1].transpose(2, 0, 1).copy()  # BGR to RGB, to 3x416x416
        img = torch.from_numpy(img).to(self.device)
        img = img.float()  # uint8 to fp16/32
        img /= 255.0  # 0 - 255 to 0.0 - 1.0

        return img

    def scale_coords_landmarks(self, img1_shape, coords, img0_shape, ratio_pad=None):
        """
        Rescale coordinates from img1_shape to img0_shape.

        :param img1_shape: Shape of the source image.
        :param coords: Coordinates to be rescaled.
        :param img0_shape: Shape of the target image.
        :param ratio_pad: Padding ratio.

        :return: Rescaled coordinates.
        """
        if ratio_pad is None:  # Calculate from img0_shape
            gain = min(img1_shape[0] / img0_shape[0], img1_shape[1] / img0_shape[1])
            pad = (img1_shape[1] - img0_shape[1] * gain) / 2, (
                img1_shape[0] - img0_shape[0] * gain
            ) / 2
        else:
            gain = ratio_pad[0][0]
            pad = ratio_pad[1]

        coords[:, [0, 2, 4, 6, 8]] -= pad[0]  # x padding
        coords[:, [1, 3, 5, 7, 9]] -= pad[1]  # y padding
        coords[:, :10] /= gain
        coords[:, :10] = coords[:, :10].clamp(
            0, img0_shape[1]
        )  # Clamp x and y coordinates

        # Reshape the coordinates into the desired format
        coords = coords.reshape(-1, 5, 2)
        return coords
```

Figure 5.13: Yolov5 Face Detection Code

```python
class SCRFD:
    def __init__(self, model_file=None, session=None):
        self.model_file = model_file
        self.session = session
        self.taskname = "detection"
        self.batched = False
        if self.session is None:
            assert self.model_file is not None
            assert osp.exists(self.model_file)
            self.session = onnxruntime.InferenceSession(self.model_file, None)
        self.center_cache = {}
        self.nms_thresh = 0.4

        self._init_vars()

    def _init_vars(self):
        input_cfg = self.session.get_inputs()[0]
        input_shape = input_cfg.shape
        if isinstance(input_shape[2], str):
            self.input_size = None
        else:
            self.input_size = tuple(input_shape[2:4][::-1])
        input_name = input_cfg.name
        outputs = self.session.get_outputs()
        if len(outputs[0].shape) == 3:
            self.batched = True
        output_names = []
        for o in outputs:
            output_names.append(o.name)
        self.input_name = input_name
        self.output_names = output_names
        self.use_kps = False
        self._num_anchors = 1
        if len(outputs) == 6:
            self.fmc = 3
            self._feat_stride_fpn = [8, 16, 32]
            self._num_anchors = 2
        elif len(outputs) == 9:
            self.fmc = 3
            self._feat_stride_fpn = [8, 16, 32]
            self._num_anchors = 2
            self.use_kps = True
        elif len(outputs) == 10:
            self.fmc = 5
            self._feat_stride_fpn = [8, 16, 32, 64, 128]
            self._num_anchors = 1
        elif len(outputs) == 15:
            self.fmc = 5
            self._feat_stride_fpn = [8, 16, 32, 64, 128]
            self._num_anchors = 1
            self.use_kps = True

    def prepare(self, ctx_id, **kwargs):
        if ctx_id < 0:
            self.session.set_providers(["CPUExecutionProvider"])
        nms_thresh = kwargs.get("nms_thresh", None)
        if nms_thresh is not None:
            self.nms_thresh = nms_thresh
        input_size = kwargs.get("input_size", None)
        if input_size is not None:
            if self.input_size is not None:
                print("warning: det_size is already set in scrfd model, ignore")
            else:
                self.input_size = input_size

    def forward(self, img, thresh):
        scores_list = []
        bboxes_list = []
        kpss_list = []
        input_size = tuple(img.shape[0:2][::-1])
        blob = cv2.dnn.blobFromImage(
            img, 1.0 / 128, input_size, (127.5, 127.5, 127.5), swapRB=True
        )
```

Figure 5.14: SCRFD Face Detection Code

```python
class IResNet(nn.Module):
    fc_scale = 7 * 7

    def __init__(
        self,
        block,
        layers,
        dropout=0,
        num_features=512,
        zero_init_residual=False,
        groups=1,
        width_per_group=64,
        replace_stride_with_dilation=None,
        fp16=False,
    ):
        super(IResNet, self).__init__()
        self.fp16 = fp16
        self.inplanes = 64
        self.dilation = 1
        if replace_stride_with_dilation is None:
            replace_stride_with_dilation = [False, False, False]
        if len(replace_stride_with_dilation) != 3:
            raise ValueError(
                "replace_stride_with_dilation should be None "
                "or a 3-element tuple, got {}".format(replace_stride_with_dilation)
            )
        self.groups = groups
        self.base_width = width_per_group
        self.conv1 = nn.Conv2d(3, self.inplanes, kernel_size=3, stride=1, padding=1, bias=False)
        self.bn1 = nn.BatchNorm2d(self.inplanes, eps=1e-05)
        self.prelu = nn.PReLU(self.inplanes)
        self.layer1 = self._make_layer(block, 64, layers[0], stride=2)
        self.layer2 = self._make_layer(
            block, 128, layers[1], stride=2, dilate=replace_stride_with_dilation[0]
        )
        self.layer3 = self._make_layer(
            block, 256, layers[2], stride=2, dilate=replace_stride_with_dilation[1]
        )
        self.layer4 = self._make_layer(
            block, 512, layers[3], stride=2, dilate=replace_stride_with_dilation[2]
        )
        self.bn2 = nn.BatchNorm2d(
            512 * block.expansion,
            eps=1e-05,
        )
        self.dropout = nn.Dropout(p=dropout, inplace=True)
        self.fc = nn.Linear(512 * block.expansion * self.fc_scale, num_features)
        self.features = nn.BatchNorm1d(num_features, eps=1e-05)
        nn.init.constant_(self.features.weight, 1.0)
        self.features.weight.requires_grad = False

        for m in self.modules():
            if isinstance(m, nn.Conv2d):
                nn.init.normal_(m.weight, 0, 0.1)
            elif isinstance(m, (nn.BatchNorm2d, nn.GroupNorm)):
                nn.init.constant_(m.weight, 1)
                nn.init.constant_(m.bias, 0)

        if zero_init_residual:
            for m in self.modules():
                if isinstance(m, IBasicBlock):
                    nn.init.constant_(m.bn2.weight, 0)

    def _make_layer(self, block, planes, blocks, stride=1, dilate=False):
        downsample = None
        previous_dilation = self.dilation
        if dilate:
            self.dilation *= stride
            stride = 1
        if stride != 1 or self.inplanes != planes * block.expansion:
            downsample = nn.Sequential(
                conv1x1(self.inplanes, planes * block.expansion, stride),
                nn.BatchNorm2d(
```

Figure 5.15: Arcface Face Recognition Code

# Chapter 6

# Conclusion & Future Scope

The Surveillance System with Human Intrusion Detection project has effectively addressed the limitations of conventional security systems by introducing a real-time face recognition-based surveillance system. Through the utilization of advanced algorithms and the OpenCV Python library, the system ensures robust surveillance in restricted areas. Its distinguishing feature lies in its ability to differentiate between intruders and authorized individuals, thereby significantly enhancing the security and integrity of monitored zones. By integrating SCRFD and YOLO for face detection, complemented by ArcFace for recognition, the system has achieved an impressive accuracy rate surpassing 90%. However, it's imperative to recognize that the accuracy is subject to the quality of the dataset and surveillance camera employed. Furthermore, the development of a user-friendly graphical interface using Python's Tkinter enhances the system's accessibility, making it applicable across various domains. This project's success highlights the rapid evolution of face recognition technology, with continual efforts aimed at improving accuracy while reducing processing overheads. The API developed as part of this project holds potential for widespread adoption in systems requiring robust face recognition capabilities.

# References


[1] Akula Surya Teja, Ginni Chandra Mohini, Dannana Dhanunjay, Dr. P M Manohar, "Realtime Intrusion Detection System Using Open CV," *Journal of Survey in Fisheries Sciences*, Visakhapatnam, India, 2023, pp. 2734-2740.

[2] G. Mallikharjuna Rao, Haseena Palle, Pragna Dasari, Shivani Jannaikode, Dr. P M Manohar, "Implementation of Low Cost IoT Based Intruder Detection System by Face Recognition using Machine Learning," *Turkish Journal of Computer and Mathematics Education*, Vol. 12, No. 13, 2021, pp. 353-362.

[3] K. S. Krishnendu, "Analysis of Recent Trends in Face Recognition Systems," 2023, *arXiv:2304.11725*.

[4] Rehmat Ullah, Hassan Hayat, Afsah Abid Siddiqui, Uzma Abid Siddiqui, Jebran Khan, Farman Ullah, Shoaib Hassan, Laiq Hasan., "A Real-Time Framework for Human Face Detection and Recognition in CCTV Images," *Mathematical Problems in Engineering*, 2022, Hindawi.

[5] Edwin Jose, Greeshma M, Mithun Haridas T. P, "Face Recognition based Surveillance System Using FaceNet and MTCNN on Jetson TX2," in *Proceedings of the 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, 2019.

[6] Philip Smith, Cuixian Chen, "Transfer Learning with Deep CNNs for Gender Recognition and Age Estimation," 2018, *arXiv:1811.07344v1 [cs.CV]*, 2018.

[7] Loorthu Infenda, Franz Cardoz, Daffril Cleetus, Ajinkya Deshpande, "Visual Sentinel: Data Analytics for Missing Subject Identification," in *6th International Conference on 'Emerging Technologies in Digital Transformation and Aligned Education'*, Pune, India, 2023.

[8] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Yu Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," *IEEE SIGNAL PROCESSING LETTERS*, October 10, 2016.

[9] F. Schroff, D. Kalenichenko, J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, 2015, pp. 815-823.

[10] T. Sanjay, W. Deva Priya, "Efficient System for Criminal Face Detection Technique on Innovative Facial Features To Improve Accuracy Using LBPH In Comparison With CNN," *Journal of Pharmaceutical Negative Results*, 2022. DOI: 10.47750/pnr.2022.13.S03.085.

[11] Arnab Pushilal, Sulakshana Chakraborty, Raunak Singhania, P. Mahalakshmi, "Implementation of Facial Recognition for Home Security Systems," *International Journal of Engineering & Technology*, Vol. 7, No. 4.10, 2018, pp. 55-58.

[12] Mfundo Zuma, Mfundo Zuma, Pius A. Owolawi, Vusi Malele, Kehinde Odeyemi, Gbolahan Aiyetoro, Joseph S. Ojo, "Intrusion Detection System using Raspberry Pi and Telegram Integration," in *Proceedings of icARTi '21*, 2021.

[13] R. Prakash, P. Chithaluru, "Active Security by Implementing Intrusion Detection and Facial Recognition," in *Nanoelectronics, Circuits and Communication Systems*, Lecture Notes in Electrical Engineering, vol 692, Springer, Singapore, 2021.

[14] Bazama A, Mansur F, Alsharef N. "Security System by Face Recognition," *Alq J Med App Sci*, 2021;4(2):58-67.

[15] Kajenthani Kanthaseelan, Paskaran Pirashaanthan, Jasmin Jelaxshana A.A.P, Akshaya Sivaramakrishnan, Kavinga Yapa Abeywardena, Tharika Munasinghe, "CCTV Intelligent Surveillance on Intruder Detection," *International Journal of Computer Applications*, Volume 174, Issue 14, 2021.

[16] R. Menaka, N. Archana, R. Dhanagopal, R. Ramesh, "Enhanced Missing Object Detection System using YOLO," in *Proceedings of the 2020*

*6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, 2020.

[17] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, Stefanos Zafeiriou, "ArcFace: Additive Angular Margin Loss for Deep Face Recognition," in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA, 2019, pp. 4685-4694.

# Acknowledgement

Success of a project like this involving high technical expertise, patience and massive support of guides, is possible when team members work together. We take this opportunity to express our gratitude to those who have been instrumental in the successful completion of this project. We would like to show our appreciation to **Mrs. Nupur Gaikwad** for her tremendous support and help, without her this project would have reached nowhere. We would also like to thank our project coordinator **Dr. Smita Dange** for providing us with regular inputs about documentation and project timeline. A big thanks to our HOD **Dr. Kiruthika M** for all the encouragement given to our team. We would also like to thank our principal, **Dr. S. M. Khot**, and our college, **Fr. C. Rodrigues Institute of Technology, Vashi**, for giving us the opportunity and the environment to learn and grow.

**Project Group Members:**

1. Lawrence Rodriques, 1020238

2. Swati Padmanabhan, 1020266

3. Sumit Prasad, 1020267

4. Neha Auti, 1020268

# Appendix A : Timeline Chart

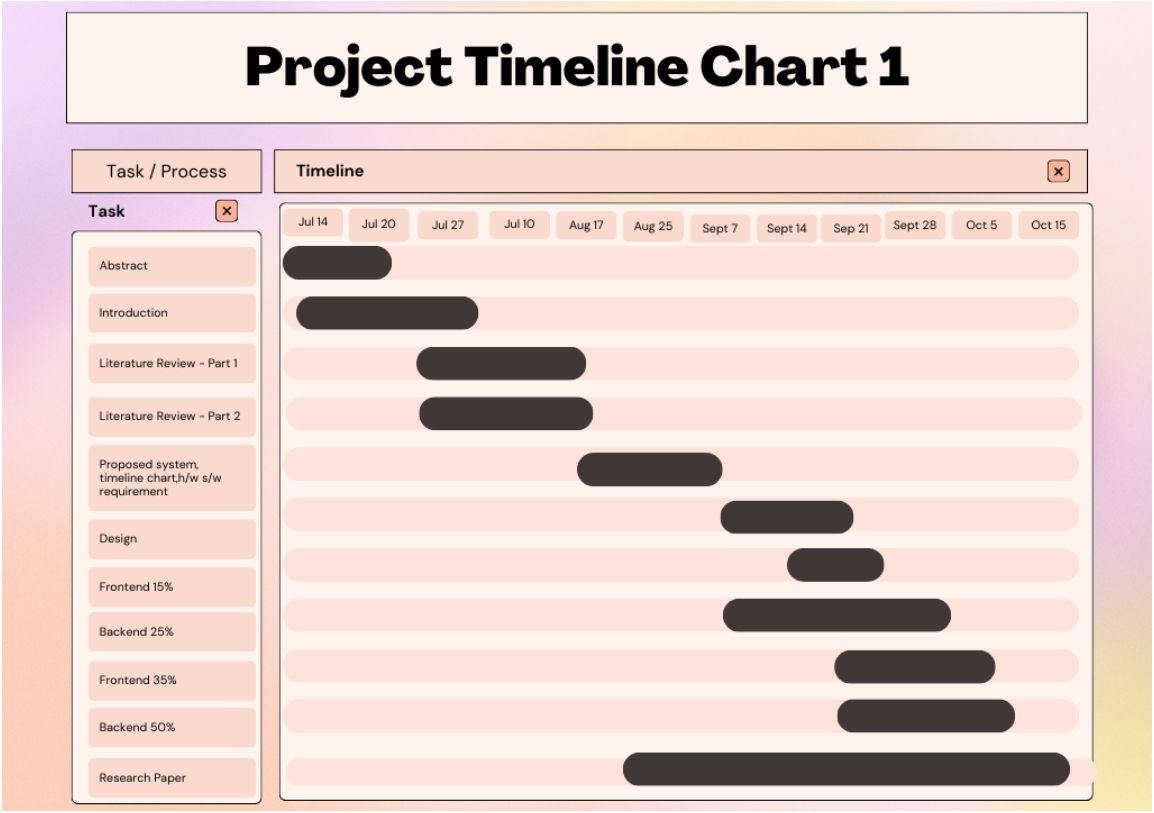



Figure 6.1: Timeline Chart 1

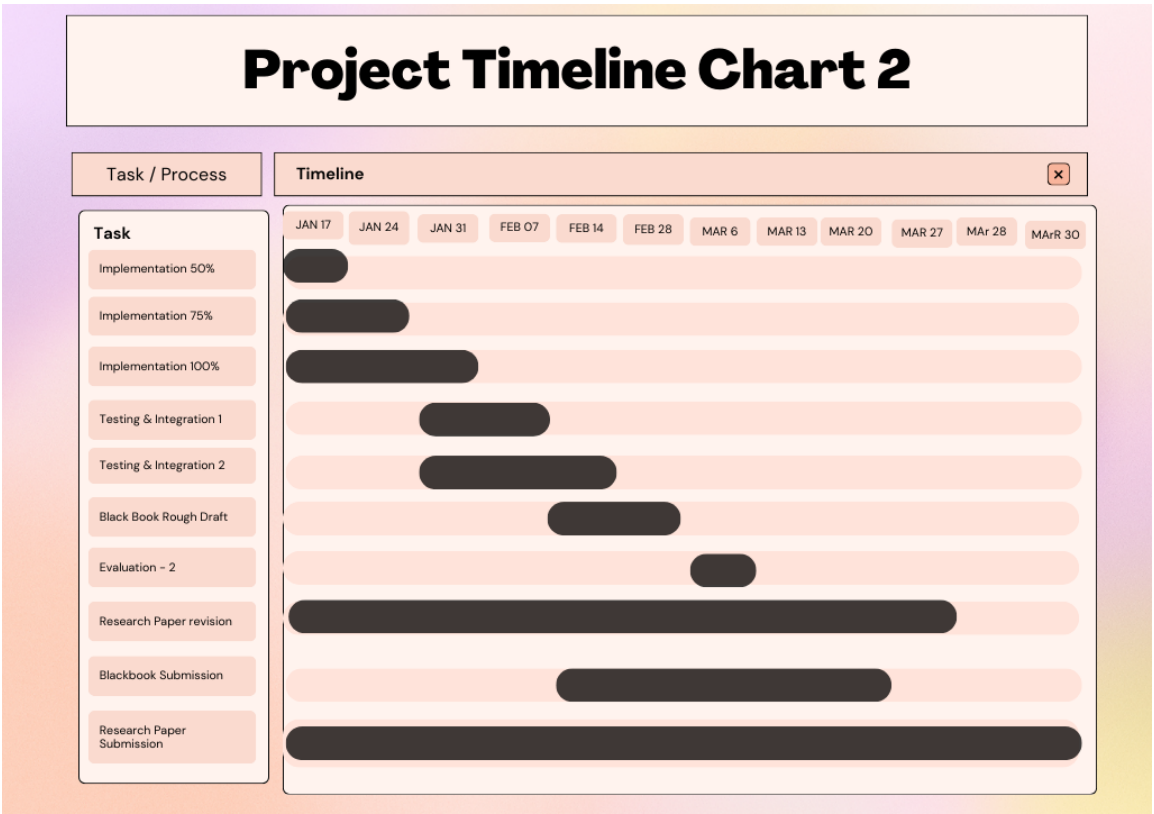



Figure 6.2: Timeline Chart 2

# Appendix B : Project Outcome

Upon successful completion of the project, our team participated in various project presentation competitions, where we showcased our innovative solution and received recognition for our efforts. Below are the certificates we obtained during these competitions:

- Intercollegiate Idea and Project Presentation (Feb 28, 2024)

RIZVI EDUCATION SOCIETY'S

RIZVI COLLEGE OF ENGINEERING

Institution's Innovation Council

National Science Day-2024

CERTIFICATE OF PARTICIPATION

IS PRESENTED TO

Mr./Ms. LAWRENCE RODRIQUES of

FR. C. RODRIGUES INSTITUTE OF TECHNOLOGY, VASHI

FOR PARTICIPATING

in Intercollegiate Idea and Project Presentation Organized by Institution's Innovation Council (IIC), Rizvi College of Engineering, Bandra, Mumbai.

28 February, 2024

Dr. Varsha Shah
Principal-RCOE

RIZVI EDUCATION SOCIETY'S

RIZVI COLLEGE OF ENGINEERING

Institution's Innovation Council

National Science Day-2024

CERTIFICATE OF PARTICIPATION

IS PRESENTED TO

Mr./Ms. SWATI PADMANABHAN of

FR. C. RODRIGUES INSTITUTE OF TECHNOLOGY, VASHI

FOR PARTICIPATING

in Intercollegiate Idea and Project Presentation Organized by Institution's Innovation Council (IIC), Rizvi College of Engineering, Bandra, Mumbai.

28 February, 2024

Dr. Varsha Shah
Principal-RCOE

RIZVI EDUCATION SOCIETY'S

RIZVI COLLEGE OF ENGINEERING

Institution's Innovation Council

National Science Day-2024

CERTIFICATE OF PARTICIPATION

IS PRESENTED TO

Mr./Ms. SUMIT PRASAD of

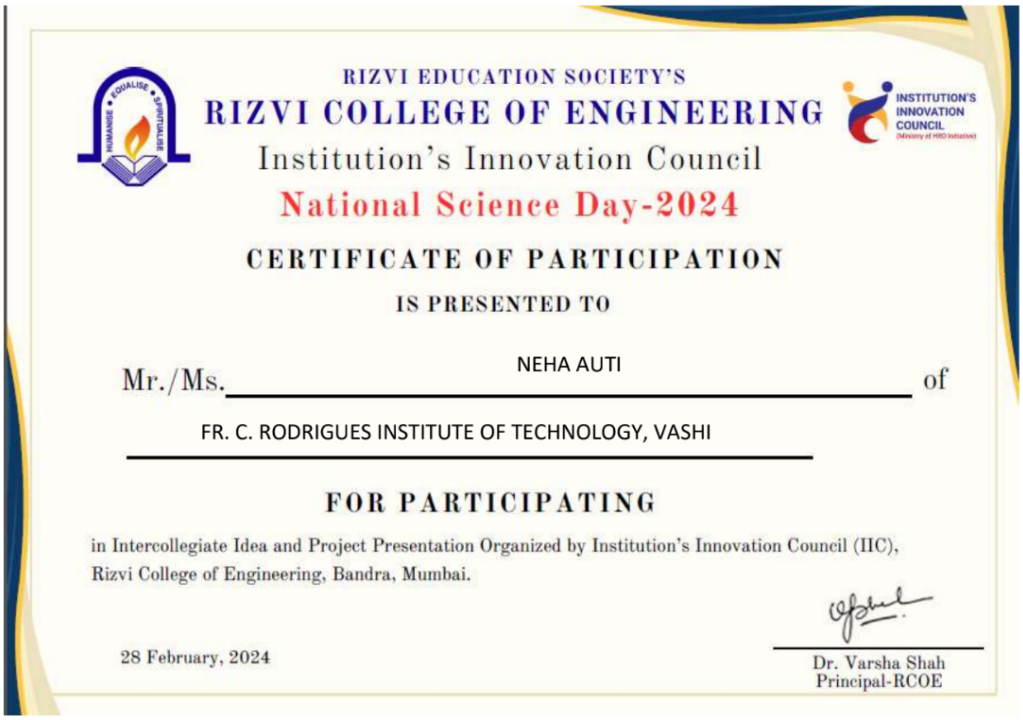
FR. C. RODRIGUES INSTITUTE OF TECHNOLOGY, VASHI

FOR PARTICIPATING

in Intercollegiate Idea and Project Presentation Organized by Institution's Innovation Council (IIC), Rizvi College of Engineering, Bandra, Mumbai.

28 February, 2024

Dr. Varsha Shah
Principal-RCOE



RIZVI EDUCATION SOCIETY'S

RIZVI COLLEGE OF ENGINEERING

Institution's Innovation Council

National Science Day-2024

CERTIFICATE OF PARTICIPATION

IS PRESENTED TO

Mr./Ms. NEHA AUTI of

FR. C. RODRIGUES INSTITUTE OF TECHNOLOGY, VASHI

FOR PARTICIPATING

in Intercollegiate Idea and Project Presentation Organized by Institution's Innovation Council (IIC), Rizvi College of Engineering, Bandra, Mumbai.

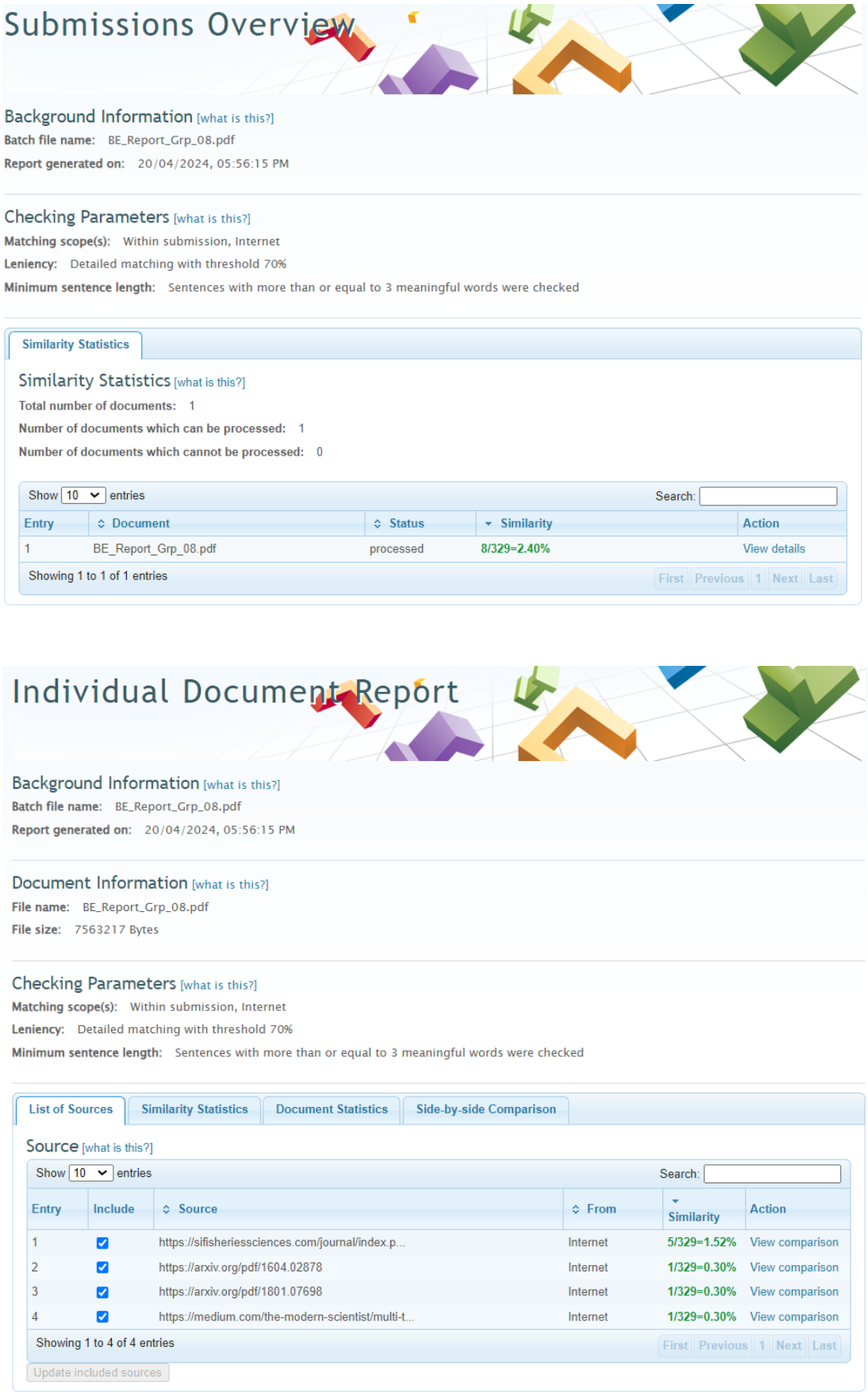
28 February, 2024

Dr. Varsha Shah
Principal-RCOE

Our paper titled "Surveillance System with Human Intrusion Detection" has been "Accepted" for publishing in "International Journal of Innovative Science and Research Technology".

# Appendix C : Plagiarism Report

## VeriGuide - Plagiarism Report



Submissions Overview

Background Information [what is this?]

Batch file name: BE_Report_Grp_08.pdf

Report generated on: 20/04/2024, 05:56:15 PM

Checking Parameters [what is this?]

Matching scope(s): Within submission, Internet

Leniency: Detailed matching with threshold 70%

Minimum sentence length: Sentences with more than or equal to 3 meaningful words were checked

Similarity Statistics

Similarity Statistics [what is this?]

Total number of documents: 1

Number of documents which can be processed: 1

Number of documents which cannot be processed: 0

Show 10 entries Search:

| Entry | Document | Status | Similarity | Action |
|---|---|---|---|---|
| 1 | BE_Report_Grp_08.pdf | processed | 8/329=2.40% | View details |

Showing 1 to 1 of 1 entries First Previous 1 Next Last

Individual Document Report

Background Information [what is this?]

Batch file name: BE_Report_Grp_08.pdf

Report generated on: 20/04/2024, 05:56:15 PM

Document Information [what is this?]

File name: BE_Report_Grp_08.pdf

File size: 7563217 Bytes

Checking Parameters [what is this?]

Matching scope(s): Within submission, Internet

Leniency: Detailed matching with threshold 70%

Minimum sentence length: Sentences with more than or equal to 3 meaningful words were checked

List of Sources | Similarity Statistics | Document Statistics | Side-by-side Comparison

Source [what is this?]

Show 10 entries Search:

| Entry | Include | Source | From | Similarity | Action |
|---|---|---|---|---|---|
| 1 | ☑ | https://sifisheriessciences.com/journal/index.p... | Internet | 5/329=1.52% | View comparison |
| 2 | ☑ | https://arxiv.org/pdf/1604.02878 | Internet | 1/329=0.30% | View comparison |
| 3 | ☑ | https://arxiv.org/pdf/1801.07698 | Internet | 1/329=0.30% | View comparison |
| 4 | ☑ | https://medium.com/the-modern-scientist/multi-t... | Internet | 1/329=0.30% | View comparison |

Showing 1 to 4 of 4 entries First Previous 1 Next Last

Update included sources