

**FACIAL RECOGNITION SYSTEM FOR CROWD SECURITY**

**BY**

**TIA-KAZTENIE GIAM HUI ZHI**

**A REPORT**

**SUBMITTED TO**

**Universiti Tunku Abdul Rahman**

**in partial fulfillment of the requirements**

**for the degree of**

**BACHELOR OF INFORMATION SYSTEMS (HONOURS) BUSINESS INFORMATION  
SYSTEMS**

**Faculty of Information and Communication Technology**

**(Kampar Campus)**

**JUNE 2023**

## REPORT STATUS DECLARATION FORM

Title: FACIAL RECOGNITION SYSTEM FOR CROWD SECURITY

\_\_\_\_\_

\_\_\_\_\_

Academic Session: JUNE 2023

I TIA-KAZTENIE GIAM HUI ZHI

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in  
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



\_\_\_\_\_  
(Author's signature)



\_\_\_\_\_  
(Supervisor's signature)

**Address:**

381, LORONG 1 EASTERN  
OFF PISANG ROAD WEST  
93150 KUCHING SARAWAK.

SU LEE SENG

Supervisor's name

Date: 13 September 2023

Date: 13 September 2023

<b>Universiti Tunku Abdul Rahman</b>			
Form Title : <b>Sample of Submission Sheet for FYP/Dissertation/Thesis</b>			
Form Number: <b>FM-IAD-004</b>	Rev No.: <b>0</b>	Effective Date: <b>21 JUNE 2011</b>	Page No.: <b>1 of 1</b>

**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY  
(KAMPAR CAMPUS)**

**UNIVERSITI TUNKU ABDUL RAHMAN**

Date: 13 September 2023

**SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS**

It is hereby certified that Tia-Kaztenie Giam Hui Zhi (ID No: 20ACB05262) has completed this final year project entitled “Facial Recognition System for Crowd Security” under the supervision of Su Lee Seng (Supervisor) from the Department of Digital Economy Technology, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,



\_\_\_\_\_  
TIA-KAZTENIE GIAM HUI ZHI

\*Delete whichever not applicable

## DECLARATION OF ORIGINALITY

I declare that this report entitled “**METHODOLOGY, CONCEPT AND DESIGN OF A 2-MICRON CMOS DIGITAL BASED TEACHING CHIP USING FULL-CUSTOM DESIGN STYLE**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature :



Name : TIA-KAZTENIE GIAM HUI ZHI

Date : 13 September 2023



## **ACKNOWLEDGEMENTS**

I would like to express my sincere thanks and appreciation to my supervisor, Mr Su Lee Seng, who has been steadfast in his belief in me and pushed me to achieve what I thought I could not. I will not forget your guidance and wisdom. A million ‘thank-you’s is not enough to convey my gratitude to have you as my mentor, sir.

I also want to extend my heartfelt appreciation to Cheong Kok Siong, Ng Suet Eng, and Tan Su Hua, who have taught me much and supported me unwaveringly throughout this project. I hope we can achieve our dreams together. And finally, to my family and friends who encouraged me not to give up and gave their unconditional love throughout this project. I wouldn’t be here without all of you. Thank you all from the bottom of my heart.

## **ABSTRACT**

Public safety is a top priority, but crowded areas witness numerous crimes annually, posing a threat to global peace and security. Identifying criminals and potential threats before they commit heinous acts like bombings, mass shootings, child abduction, and sexual assaults in public spaces is vital. While CCTV cameras offer post-incident monitoring, integrating facial recognition technology with live video feeds can proactively prevent such tragedies. A facial recognition system is developed to identify known criminals and missing persons from a face database, enabling public surveillance cameras to track their whereabouts, monitor their activities, and notify authorities promptly when needed. This Python project uses Convolutional Neural Network (CNN) face recognition with Dlib and Haar Cascade Classifier to effectively detect and monitor known and potentially dangerous individuals in public areas, facilitating swift emergency responses when necessary, while keeping watch for missing persons. The system developed uses Firebase's Realtime Database and Storage Bucket to store and retrieve data in real-time to expedite system functionalities like reports generation and database management.

# TABLE OF CONTENTS

<b>TITLE PAGE</b>	<b>i</b>
<b>REPORT STATUS DECLARATION FORM</b>	<b>ii</b>
<b>FYP THESIS SUBMISSION FORM</b>	<b>iii</b>
<b>DECLARATION OF ORIGINALITY</b>	<b>iv</b>
<b>ACKNOWLEDGEMENTS</b>	<b>v</b>
<b>ABSTRACT</b>	<b>vi</b>
<b>TABLE OF CONTENTS</b>	<b>vii</b>
<b>LIST OF FIGURES</b>	<b>x</b>
<b>LIST OF TABLES</b>	<b>xi</b>
<b>LIST OF SYMBOLS</b>	<b>xii</b>
<b>LIST OF ABBREVIATIONS</b>	<b>xiii</b>
<b>CHAPTER 1 INTRODUCTION</b>	<b>1</b>
1.1 Problem Statement and Motivation	3
1.2 Objectives	6
1.3 Project Scope and Direction	8
1.4 Contributions	10
1.5 Report Organization	10
<b>CHAPTER 2 LITERATURE REVIEW</b>	<b>11</b>
2.1 Facial Recognition System for Suspect Identification Using a Surveillance Camera []	11
2.2 Tracking Missing Person n Large Crowd Gathering Using Intelligent Video Surveillance []	13
2.3 A Cost-Efficient Real-Time Security Surveillance System Based on Facial Recognition Using Raspberry Pi and OpenCV []	15
2.4 Analysis of Face Recognition: DLIB and OpenCV []	16
2.5 FaceMe® Security System by CyberLink Corporation []	18
2.6 Public Security by Avigilon Corporation []	20

<b>CHAPTER 3 SYSTEM METHODOLOGY/APPROACH (FOR DEVELOPMENT-BASED PROJECT)</b>	<b>22</b>
3.1 Methodology	22
3.2 Project Timeline	24
3.3 System Design Diagram	28
3.3.1 Login Module	29
3.3.2 View People Module	31
3.3.3 View Video Feed Module	33
3.3.4 View Reports Module	36
<b>CHAPTER 4 SYSTEM DESIGN</b>	<b>39</b>
4.1 System Block Diagram	39
4.2 System Flow Description	41
4.2.1 Main.py	41
4.2.2 Dashboardui.py	42
4.2.3 Peopledbui.py	47
4.2.4 PersonInfo.py	54
4.2.5 Allcams.py	60
4.2.6 Reports.py	66
<b>CHAPTER 5 SYSTEM IMPLEMENTATION (FOR DEVELOPMENT-BASED PROJECT)</b>	<b>73</b>
5.1 Hardware Setup	73
5.2 Software Setup	74
5.3 Settings and Configuration	76
5.4 System Operation	77
<b>CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION</b>	<b>94</b>
6.1 System Evaluation Survey Results	94
6.2 Testing Setup and Results	104
6.3 Implementation Issues and Project Challenges	108
6.4 Objectives Evaluation	109

<b>CHAPTER 7 CONCLUSION AND RECOMMENDATION</b>	<b>110</b>
7.1 Conclusion	110
7.2 Recommendations	111
<b>REFERENCES</b>	<b>112</b>
<b>APPENDIX</b>	<b>114</b>
<b>WEEKLY LOG</b>	<b>119</b>
<b>POSTER</b>	<b>125</b>
<b>PLAGIARISM CHECK RESULT</b>	<b>126</b>
<b>FYP2 CHECKLIST</b>	<b>129</b>

## LIST OF FIGURES

<b>Figure Number</b>	<b>Title</b>	<b>Page</b>
Figure 2.2	Total Number of Faces Detected over Video Sequence in []	13
Figure 2.3.1	Flow of LBPH Method []	15
Figure 2.3.2	Basic Flowchart of How the System in [] Works	15
Figure 2.4.1	Results from the Speed Tests and Accuracy Tests of CNN, HOG, DNN, HAAR algorithms conducted by []	16
Figure 2.4.2	Summary of Processing Speed (per second) and Accuracy (FRR and FAR percentages) in []	17
Figure 2.5	FaceMe® Security Solution’s System Structure []	18
Figure 2.6	Avigilon’s Focus of Attention (FoA) User Navigation Screen []	20
Figure 3.1	Agile Methodology	22
Figure 3.2.1	Gantt Chart of Project (1/4)	24
Figure 3.2.2	Gantt Chart of Project (2/4)	25
Figure 3.2.3	Gantt Chart of Project (3/4)	26
Figure 3.2.4	Gantt Chart of Project (4/4)	27
Figure 3.3	System Use Case Diagram	28
Figure 3.3.1	Login Module – Activity Diagram	30
Figure 3.3.2	View People Module – Activity Diagram	32
Figure 3.3.3	View Video Feed Module – Activity Diagram	34
Figure 3.3.4	View Reports – Activity Diagram	37
Figure 4.1	System Block Diagram	39
Figure 4.2.1	Main.py	41
Figure 4.2.2.1	Dashboardui.py (1/4)	42
Figure 4.2.2.2	Dashboardui.py (2/4)	43
Figure 4.2.2.3	Dashboardui.py (3/4)	44
Figure 4.2.2.4	Dashboardui.py (4/4)	45
Figure 4.2.3.1	Peopledbui.py (1/6)	47
Figure 4.2.3.2	Peopledbui.py (2/6)	48

Figure 4.2.3.3	Peopledbui.py (3/6)	49
Figure 4.2.3.4	Peopledbui.py (4/6)	50
Figure 4.2.3.5	Peopledbui.py (5/6)	51
Figure 4.2.3.6	Peopledbui.py (6/6)	52
Figure 4.2.4.1	PersonInfo.py (1/6)	54
Figure 4.2.4.2	PersonInfo.py (2/6)	55
Figure 4.2.4.3	PersonInfo.py (3/6)	56
Figure 4.2.4.4	PersonInfo.py (4/6)	57
Figure 4.2.4.5	PersonInfo.py (5/6)	58
Figure 4.2.4.6	PersonInfo.py (6/6)	59
Figure 4.2.5.1	Allcams.py (1/6)	60
Figure 4.2.5.2	Allcams.py (2/6)	61
Figure 4.2.5.3	Allcams.py (3/6)	62
Figure 4.2.5.4	Allcams.py (4/6)	63
Figure 4.2.5.5	Allcams.py (5/6)	64
Figure 4.2.5.6	Allcams.py (6/6)	65
Figure 4.2.6.1	Reports.py (1/6)	66
Figure 4.2.6.2	Reports.py (2/6)	67
Figure 4.2.6.3	Reports.py (3/6)	68
Figure 4.2.6.4	Reports.py (4/6)	69
Figure 4.2.6.5	Reports.py (5/6)	70
Figure 4.2.6.6	Reports.py (6/6)	71
Figure 5.3	Rules configuration of Firebase’s Realtime Database expiring on 31 <sup>st</sup> December 2023	76
Figure 5.4.1	Login Screen	77
Figure 5.4.2	Login Unsuccessful	78
Figure 5.4.3	Login Successful	78
Figure 5.4.4	Main Dashboard Screen	79
Figure 5.4.5	All Cameras Surveillance with “Caution” Status Detected	80
Figure 5.4.6	All Cameras Surveillance with Faces Detected in Different Locations	80
Figure 5.4.7	View More Information on a Detected Person	81
Figure 5.4.8	Emergency Contacts	82

Figure 5.4.9	Database of People Screen	83
Figure 5.4.10	Search by Name	84
Figure 5.4.11	Search by Status	84
Figure 5.4.12	Sort Database by Date Detected	86
Figure 5.4.13	Sort Database by Name	86
Figure 5.4.14	Detailed Information on a Specific Person	87
Figure 5.4.15	A person's information before making changes	88
Figure 5.4.16	A person's information after making changes	88
Figure 5.4.17	Changes made are saved	89
Figure 5.4.18	Reports Generation Screen	90
Figure 5.4.19	Example of a report (1/2)	91
Figure 5.4.20	Example of a report (2/2)	91
Figure 5.4.21	Saving a report to CSV File	92
Figure 5.4.22	Example of saved CSV File	93
Figure 6.1.1	Respondents' familiarity with Security Surveillance Systems	94
Figure 6.1.2	Reasons for Using Security Surveillance Systems	95
Figure 6.1.3	Comfort of respondents using face recognition technology	95
Figure 6.1.4	How important respondents think face recognition technology is in security surveillance systems	96
Figure 6.1.5	Ease and Intuitiveness of Navigation	96
Figure 6.1.6	Rating of User Interface	97
Figure 6.1.7	Overall Satisfaction while Using System	97
Figure 6.1.8	Valuable or Useful System Features	98
Figure 6.1.9	Value of categorising people based on status	99
Figure 6.1.10	Suggestions to Improve Categorisation and Display of People	99
Figure 6.1.11	Accuracy of Face Recognition	100
Figure 6.1.12	Speed and Responsiveness of System	101
Figure 6.1.13	Value of Generated Reports and Analysis	101
Figure 6.1.14	Suggestions for Other Reports	102
Figure 6.1.15	Suggestions for Other Features or Improvements	102
Figure 6.1.16	Likely Recommendation to Others	103



## LIST OF TABLES

<b>Table Number</b>	<b>Title</b>	<b>Page</b>
Table 3.3.1	Login Module – Use Case Description	29
Table 3.3.2	View People Module – Use Case Description	31
Table 3.3.3	View Video Feed Module – Use Case Description	33
Table 3.3.4	View Reports Module – Use Case Description	36
Table 5.1	Specifications of laptop	73
Table 5.2	Specifications of android mobile phone	73
Table 6.2.1	Unit Test 1 – Login	104
Table 6.2.2	Unit Test 2 – Dashboard	104
Table 6.2.3	Unit Test 3 – Viewing All Camera Feeds	105
Table 6.2.4	Unit Test 4 – Viewing Database of People	106
Table 6.2.5	Unit Test 5 – Generating Reports	107

## LIST OF ABBREVIATIONS

<i>RAM</i>	Random Access Memory
<i>AI</i>	Artificial Intelligence
<i>ML</i>	Machine Learning
<i>DL</i>	Deep Learning
<i>CCTV</i>	Closed Circuit Television
<i>PIN</i>	Personal Identification Number
<i>OpenCV</i>	Open Computer Vision
<i>LBPH</i>	Local Binary Patterns Histogram
<i>PCA</i>	Principal Component Analysis
<i>LDA</i>	Linear Discriminate Analysis
<i>FRR</i>	False Rejection Rate
<i>FAR</i>	False Acceptance Rate
<i>RGB</i>	Red, Green, Blue
<i>HSI</i>	Hue, Saturation, Intensity
<i>CNN</i>	Convolutional Neural Network
<i>CNN</i>	Deep Neural Network
<i>HOG</i>	Histogram of Oriented Gradients
<i>SDK</i>	Software Development Kit
<i>API</i>	Application Programming Interface
<i>VMS</i>	Video Management System
<i>SQL</i>	Structured Query Language
<i>eKYC</i>	Electronic Know Your Customer
<i>ACC</i>	Avigilon Control Centre
<i>VIP</i>	Very Important Person
<i>MMOD</i>	Max Margin Object Detection
<i>UI</i>	User Interface
<i>IC</i>	Identification Card
<i>IDE</i>	Integrated Development Environment
<i>IP</i>	Internet Protocol
<i>USB</i>	Universal Serial Bus
<i>URL</i>	Uniform Resource Locator

# Chapter 1

## Introduction

As individuals gain a greater understanding of societal concerns, contemporary happenings, and past events such as the World Trade Centre's collapse and illicit networks engaged in human trafficking, the spotlight turns towards the need to prevent and ensure safety against these hazards. The technological progress aimed at countering terrorism and unlawful pursuits has swiftly evolved to protect the peace and security of civilians. Nonetheless, the occurrence of terrorism and criminal endeavours that jeopardize public order remains persistent in the present day. Daily, there are instances of heinous crimes that imperil public security. According to research conducted in 2017, approximately four children disappear daily in Malaysia [1]. This equates to nearly 1,500 children vanishing annually. It is truly disheartening that even in the contemporary 21st century, individuals can vanish without a trace and fall prey to criminal networks involved in activities like trafficking, homicide, sexual assault, and other grave offenses. Neglecting the application of technology to uphold the security and well-being of innocent individuals represents a significant failure in human progress.

Facial recognition stands out as a leading technological advancement. With its myriad applications, particularly in monitoring crowds, facial recognition plays a crucial role in identifying individuals with criminal intent who pose a threat to public safety or in locating individuals who might be lost within a crowd. While instances of crowd-related terrorism thankfully remain infrequent in Malaysia, surveillance technologies such as facial recognition retain their importance as preventive measures against other forms of criminal activity. This technology holds potential beyond the identification of terrorism suspects; it can also aid in identifying other potential wrongdoers. The system would activate when a wanted individual appears in any location covered by cameras connected to a central server. This server would house a database of individuals to be located, encompassing both criminal offenders and missing persons, complete with their biometric data. The implications of such a system are far-reaching, benefiting searches for missing individuals, apprehending criminals who attempt to blend into crowds, and even supervising minors in high-risk areas.

The process of identifying and recognizing faces presents a significant challenge due to a variety of factors, including the resolution of surveillance cameras, the movement of cameras, their distance from crowds, lighting conditions, facial expressions, the presence of accessories like hats and glasses, the dynamic nature of the crowd, and the presence of multiple subjects within each camera frame. Nonetheless, facial recognition technology can be advanced through the application of artificial intelligence (AI), specifically utilizing machine learning (ML) and deep learning (DL) techniques. AI mimics human facial features, while ML (a subset of AI) establishes a framework employing algorithms and probabilities to interpret patterns, enabling machines to comprehend visual data. Within the subset of Machine Learning, Deep Learning (DL) delves further by constructing architectures that replicate and learn from the human brain. DL encompasses three primary techniques: artificial neural networks, convolutional neural networks, and recurrent neural networks [2], each tailored for various types of data. In the context of creating a facial recognition system, the convolutional neural network is prominently employed, and some instances of recurrent neural networks are utilized for analysing video feeds. The identification of objects, specifically facial features, involves the application of the Viola-Jones method for detecting Haar-like features [2]. This method is implemented after converting the captured image to grayscale, enhancing system efficiency. In the case of video feeds, frames from cameras transformed into grayscale are processed through the facial recognition system to detect and identify faces. The utilization of grayscale mitigates the impact of low-light conditions on the accuracy of recognition.

China stands as one of the leading global pioneers in leveraging facial recognition technology to enhance crowd security and reduce overall crime rates. Back in 2016, China initiated the Sharp Eyes project across both urban and rural areas, leading to a noticeable decline in crimes such as theft, violence, and even arson, with rates dropping close to zero [3]. The deployment of facial recognition surveillance systems like this has effectively minimized risks to public safety, fostering a peaceful environment characterized by reduced crime rates. Initiatives like Sharp Eyes not only bolster security measures but also alleviate the strain on potentially understaffed law enforcement agencies. As a case in point, consider Pingyi county, where a population of one million was served by only about 300 law enforcement personnel; the Sharp

Eyes project significantly eased the burden on the county's police department [3]. Drawing insights from this, it becomes evident how facial recognition is progressively gaining significance and proving advantageous for crowd management and security surveillance.

## **1.1 Problem Statement and Motivation**

Traditional surveillance approaches are riddled with various constraints that impede public safety. Instances of suspicious activities within crowds frequently escape notice, and it's only after an event has transpired and concluded that CCTV surveillance footage is examined retrospectively. As the saying goes, it's wiser to prevent than to remedy. The incorporation of facial recognition technology into crowd surveillance setups offers a viable solution to mitigate numerous challenges. This integration has the potential to proactively address these issues, curbing the occurrence of incidents before they even unfold.

### **(i) Access to secure facilities and venues are susceptible to weaknesses and fraudulent incidents.**

Critical facilities such as airports, immigration borders, and concert venues commonly implement stringent security protocols to restrict entry exclusively to authorized individuals. The prevailing safeguarding methods entail identity badges or access cards; however, these can be prone to loss, theft, or counterfeiting. While augmenting these measures with passwords or personal identification numbers (PINs) may seem promising, human fallibility often leads to forgotten or inadequately protected codes, thereby jeopardizing confidentiality. Employing conventional means like these for crossing national borders or gaining access to crowded event venues leaves them vulnerable to fraudulent activities.

Implementing biometric-based security measures like facial recognition represents a significant advancement in securing sensitive facilities. Such measures would rely on an individual's unique facial features, making manipulation considerably more difficult without resorting to elaborate methods. Facial recognition systems could complement existing security methods such as access cards, badges, passwords, and PINs. A pertinent example is Malaysia Airports' introduction of a passenger reconciliation system in 2021.

This system scans and cross-references passengers' travel documents with the airport's database in real time to identify fraudulent travel documents [4]. While this step enhances airport security, there remains a potential risk posed by criminals or terrorists possessing stolen or forged documents. Immigration staff, burdened by fatigue or the sheer volume of faces they encounter, might struggle to distinguish such cases, potentially rendering security clearance susceptible to lapses.

**(ii) Human eyes are unreliable in pinpointing specific individuals in large crowds.**

When known criminals or suspected terrorists infiltrate crowds, it becomes highly challenging for law enforcement to effectively monitor their movements due to the inherent limitations of human vision. The ability to pinpoint specific individuals within busy and crowded settings, comprising hundreds or even thousands of people, is almost unattainable. In the unfortunate event of an undesirable incident, the chaotic nature of the crowd adds to the confusion and panic, potentially causing law enforcement personnel to become engulfed in the disorder. This environment provides suspects with opportunities to evade detection and engage in further criminal activities, causing more harm. Moreover, perpetrators can rapidly alter their external appearances, including clothing and hairstyles, which further complicates pursuit and confuses law enforcement efforts. This difficulty also extends to the search for missing people, not just criminals.

Expecting security personnel to maintain unwavering vigilance over a conventional CCTV surveillance feed from multiple cameras is impractical and potentially ineffective. Such personnel could easily become distracted or experience visual fatigue from prolonged screen exposure, thereby leading to the oversight of crucial incidents captured on camera or perilous activities unfolding within crowds. A more efficient allocation of resources involves leveraging an intelligent video surveillance feed to scrutinize and identify significant events or individuals of interest during extended periods of inactivity. This approach allows security personnel to fulfil their other responsibilities while the intelligent system augments their monitoring efforts.

(iii) **Comprehensive and current information of dangerous criminals and missing persons are not usually disseminated to the general public.**

It's infrequent for the faces of criminals or terrorists to be widely publicized or shared across all law enforcement agencies. Moreover, there can be instances where law enforcement authorities exhibit carelessness or lapse in maintaining a comprehensive database of both active criminals and those with prior criminal records, who are at large in the public domain. An illustrative example can be found in Houston, Texas, where a registered sex offender managed to evade arrest until November 2022, despite committing numerous public sexual offenses since 2012 [5]. This case underscores that the public and law enforcement might remain unaware of an individual's criminal history unless their identity is manually cross-referenced with police databases, a time-consuming process. Furthermore, this method is susceptible to human errors, allowing individuals to evade scrutiny. Deploying facial recognition-enabled CCTV cameras in the area could have promptly exposed the offender's presence and triggered alerts to area management, enabling proactive monitoring before he could perpetrate further sexual offenses. The current approach of conducting security checks on every individual entering a public venue, such as a children's playground, is not only inconvenient but also highly impractical in preventing incidents like kidnappings. This is even before considering the available resources to conduct such security checks.

Often, individuals are not fully informed about every missing person case reported in the news, particularly when it concerns kidnapped children. Furthermore, people might struggle to recall the appearance of these missing individuals. Human traffickers, who might not have prior criminal records, can exploit this lack of awareness, maneuvering through crowds while guiding their victims, as bystanders remain oblivious to the crime taking place before their eyes. Integrating facial recognition technology into CCTV surveillance can offer a potential solution. By detecting the identity of victims held against their will, the system could trigger an immediate response from law enforcement, potentially preventing a dire outcome.

Adopting this approach would streamline and expedite the process, ensuring accuracy in identity checks. Continuous, real-time identity verification through CCTV surveillance would facilitate timely alerts to security personnel only when persons of interest are

identified. This could enhance both the efficiency and effectiveness of security measures, potentially saving lives and thwarting criminal activities.

## 1.2 Objectives

### **i. Enhance security measures for accessing sensitive facilities and venues through the implementation of facial biometric recognition technology.**

Implementing facial recognition technology in sensitive facilities such as airports and immigration borders offers a robust security enhancement, replacing conventional methods like keys, access cards, or passwords. This approach not only reduces the chances of identity fraud but also ensures that these locations remain safeguarded from potentially dangerous individuals. By utilizing facial recognition to analyse specific biometric markers of individuals at entry points, only authorized personnel with designated access privileges can enter restricted areas.

The system can be tailored to incorporate a whitelist containing verified individuals recognized solely by the system. Anyone not on the whitelist would be denied access. Additionally, a blacklist could be configured to trigger alerts when a person from the list is detected, allowing security personnel to swiftly respond by closely monitoring the individual, escorting them off the premises, or involving the relevant authorities.

Integrating this system with CCTV surveillance cameras across the sensitive premises streamlines the security access validation process, ensuring that all individuals within the facility possess proper authorization. This comprehensive approach helps prevent unauthorized access and potential threats, ultimately contributing to the safety of everyone present. Should someone without access rights be detected, security personnel can take immediate measures to avert any undesirable outcomes.



**ii. Leverage a facial recognition system to identify suspects efficiently and accurately within densely populated areas.**

The developed system would possess the capability to swiftly detect and accurately identify individuals of interest, aiding law enforcement agencies in their pursuit. Security administrators can utilize the system's tracking functionality to designate a person of interest, which prompts the system to continuously monitor their movements within the premises. This proactive approach thwarts suspects from escaping into crowds and putting innocent individuals at risk. By calculating and providing real-time updates on the suspect's precise location, the system empowers law enforcement to swiftly apprehend them.

This approach proves especially beneficial for large-scale video surveillance setups covering extensive areas, necessitating multiple CCTV cameras for comprehensive monitoring. As each camera is assigned to oversee specific locations, flagging and continuously tracking a person's movements through the system considerably simplifies the process of locating them. This enables security personnel to make swift decisions based on the system's information, minimizing valuable time wasted and enabling them to reach the suspects promptly. Security personnel can efficiently assess whether there's reason for concern and swiftly alert emergency services, offering detailed insights into the situation and location through continuous monitoring of the flagged individual on the system's display. Of course, this feature is not only limited to locating and tracking criminal suspects or missing persons. It would be helpful in countless situations, like a parent misplacing their child in a crowd, allowing medics to know the precise location of a person in need of their assistance immediately, and many more.

**iii. Use facial recognition technology to identify potentially dangerous persons and missing individuals who may not be publicly recognised, thereby alerting authorities to potential threats.**

The developed system would systematically process all faces identified within crowds against a blacklist database. Upon detecting any individuals matching the entries on the blacklist, an alert would be promptly transmitted to local authorities, prompting heightened awareness of potential suspicious activities or immediate apprehension if the person is already wanted. This

approach effectively curbs public disturbances and fosters a proactive stance by law enforcement or venue management, as demonstrated by the incident referenced [5].

By keeping security personnel alert from the moment the system signals the presence of an individual with a criminal history, law enforcement can intervene and prevent crimes from escalating. Additionally, granting venue management, such as help-desk administrators, access to live video surveillance empowers them to participate in the keeping of public security. A practical illustration involves a store that had experienced theft in the past; logging the facial data of the thieves into the database enables swift identification upon their return, facilitating their immediate apprehension and deterrence.

This system serves to unite individuals in their vigilance against criminals and potentially hazardous individuals, past or present, by fostering collective watchfulness.

### **1.3 Project Scope and Direction**

The project entails the development of a computer-based web application designed to enhance public crowd security. This application functions by scanning individuals within the crowd to identify known criminals or terrorists, subsequently triggering alerts to initiate monitoring or pursuit actions by the system.

The following are the scope and features of the project application:

#### **i. Facial recognition module**

- The system can detect and identify faces present in static images, video recordings, and real-time live video feeds.
- The system has the capability to identify faces in low-light conditions, from various perspectives, while moving, and maintain tracking of the person in motion.
- The system can successfully identify faces even with accessories like hats or glasses.
- The system compares detected faces against a pre-existing database, and upon finding a match, it displays pertinent information about the identified individual.

**ii. Login and administration management**

- The system can log the identity of the security administrator using it, serving as a time-keeping feature.
- The system generates summaries and analytical reports detailing the identified individuals.

**iii. Database of people management**

- Administrators can add, remove, or update individuals' information in the database within the system, catering to the diverse requirements of different establishments.
- The system maintains a historical log of all detected individuals, ensuring a reference point for record retrieval.

**iv. Camera and video management**

- The system links with numerous camera feeds to conduct real-time recognition.
- Administrators have the capability to incorporate new cameras, eliminate existing ones, or modify camera details as necessary, facilitating scalability to accommodate distinct camera needs across various establishments.
- The system will also include the location of the person detected in their historical log based on the camera they were detected on.

**v. Security alerts and actions**

- Alerts will be triggered when the system identifies an individual flagged as dangerous, displaying their location within the premises.
- Alerts will be generated upon detecting a person of interest, such as a missing individual, revealing their location on the premises.
- The system has the capability to establish connections with local emergency services—medical, fire, and law enforcement—enabling swift communication by the security administrator if needed.
- Administrators can mark a person of interest to enable ongoing tracking of their movements within the premises.

## **1.4 Contributions**

This project demonstrates the practicality of employing methods for identifying and tracking criminals in real-time through live camera feeds. The project extensively utilizes the capabilities of Dlib and OpenCV libraries for facial recognition, specifically for monitoring crowds through video surveillance.

The primary objective of this project is to establish scalability, ensuring that even modestly budgeted enterprises can implement the developed system to enhance public safety. By showcasing the value of facial recognition systems in video surveillance and fortifying security for sensitive facilities, the project contributes to highlighting the necessity for such technology.

Furthermore, the project advances facial recognition technology by pushing the boundaries of Dlib and OpenCV, especially in handling multiple faces within a single frame and continuously processing them in live video feeds. This exploration of limits can lead to further breakthroughs in the field.

An additional significant contribution lies in emphasizing the importance of monitoring individuals with criminal backgrounds, particularly those with a history of kidnapping or other dangerous offenses. This project's widespread adoption could incentivize individuals with prior criminal records to adhere to lawful behaviour, fostering a safer society. Additionally, the project has the potential to expedite the location of missing persons or kidnapping victims, adding a humanitarian aspect to its impact.

## **1.5 Report Organization**

This report is arranged into 7 chapters. The first chapter is introduction to the project including problem statements and objectives, the second is literature review, the third is the system methodology used in this project and system designs. The fourth is the system design including the code for this project, the fifth is the system implementation, and the sixth is the evaluation and discussion of the system's strengths and weaknesses. Finally, the last chapter is the conclusion and recommendations.

# Chapter 2

## Literature Review

### 2.1 Facial Recognition System for Suspect Identification Using a Surveillance Camera

[6]

The process of facial recognition, as outlined in [6], comprises four main stages: object detection, feature extraction, model training, and real-time model execution. Object detection presents challenges due to factors like pixel levels and shadow intensity, impacting detection accuracy. The Viola-Jones framework and Haar cascades, utilized by Kumar et al., incorporate integral images for rapid feature computation. They employed Adaptive Boost (Adaboost) for feature selection and cascaded classifiers, achieving 15% faster computation.

The study's accuracy, as reported in [6], stands at 80% with respect to a face value threshold, influenced by the key factor used for model training to recognize images with similar attributes. Local Binary Patterns Histograms (LBPH) algorithms are introduced to reduce computational costs. Principal Component Analysis (PCA) and Linear Discriminate Analysis (LDA) yield higher success rates for probe-type recognition.

To ensure robust recognition under varied conditions, [6] converts video feeds to greyscale, minimizing visual noise and allowing the system to identify moving individuals in diverse lighting and orientation. Contextual information, including clothing recognition, bolsters identification, enhancing accuracy.

The developed system in [6] executes effective facial recognition through the following steps:

- i. Acquiring face images
- ii. Converting video to frames (pixels)
- iii. Detecting faces using Haar-like features on greyscale images
- iv. Extracting unique features

v. Applying the LBPH algorithm for face recognition with database images

vi. Reporting matches when found

The system successfully operates with live video feeds. Rigorous experimentation and performance evaluations were conducted, employing metrics such as false rejection rate (FRR) and false acceptance rate (FAR). Automated alerts streamline system supervision, triggered upon query image detection.

## 2.2 Tracking Missing Person in Large Crowd Gathering Using Intelligent Video Surveillance [7]

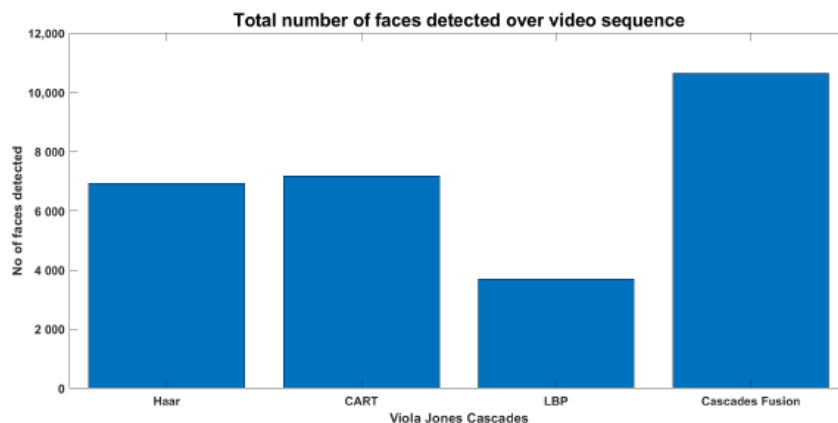
The study conducted by [7] focuses on tracking missing individuals within the vast crowds at Al-Nabawi Mosque in Madinah. The process for searching for missing persons closely parallels that of tracking criminals or terrorists within a database, with the primary distinction being the establishment of whitelists and blacklists to categorize persons of interest and differentiate security alerts accordingly. The system proposed by [7] primarily targets tracking a single individual in low-resolution images within a massive, uncontrolled crowd. The system's key steps are as follows:

i. Geofence Set Estimation

The entire premises are divided into a 5x5 grid of geofences, each assigned to a dedicated surveillance camera. The geo-location of the missing person and estimated time lapse are processed through the first algorithm, geofence set estimation. This step aims to approximate the missing person's location and reduce the search area.

ii. Faces Detection in Video Frames

A second algorithm, known as the tracking workflow, samples every 10th frame in the video feed and detects all faces within the assigned geofence using the Viola-Jones detector. Three concurrent face detection methods, including Cascaded CART, Cascaded Haar, and Cascaded LBP, are employed. The faces detected undergo overlap analysis using a Proposed Face Fusion algorithm to minimize false negatives, especially as the total number of detected faces increases through cascading.



**Figure 2.2 Total Number of Faces Detected over Video Sequence in [7]**

iii. Face Recognition

All detected faces are processed through five recognition algorithms to assign scores and identification numbers. Faces that receive consistent identifications from multiple algorithms are matched to the database, assuming that the detected face is correct.

iv. Missing Person Tracking

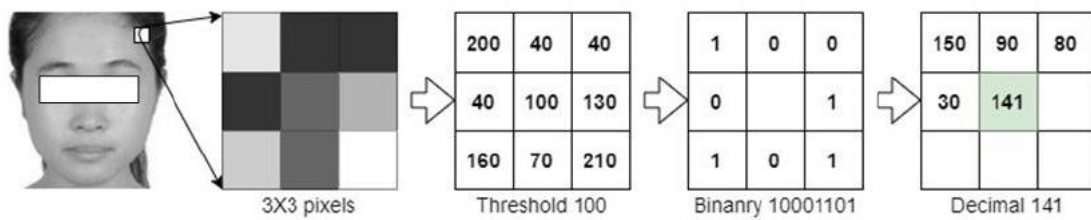
The location of each identified face is recorded and tracked across consecutive frames.

The study achieved notable success in recognizing and identifying missing individuals within large crowds. However, it has limitations when the missing person's face is obscured by certain head accessories.



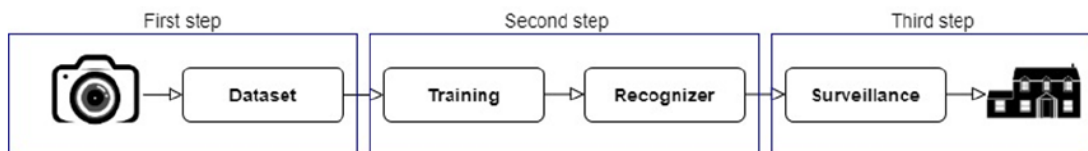
### 2.3 A Cost-Efficient Real-Time Security Surveillance System Based on Facial Recognition Using Raspberry Pi and OpenCV [8]

The system detailed in [8] shares similarities with the previously reviewed literature. Face detection employs the Haar Cascade algorithm for integral image construction, AdaBoost training, and Cascade Classifier creation, with the distinction that [8] employs a Raspberry Pi for computation. For face recognition, the system utilizes the Local Binary Pattern Histograms (LBPH) algorithm from the OpenCV library.



**Figure 2.3.1 Flow of LBPH Method [8]**

The process unfolds in three straightforward steps. First, a dataset is established by capturing images using a camera or importing stored images. The face detection algorithm categorizes faces as new or existing, assigns an ID, and integrates them into the dataset. After finalizing the dataset, the LBPH algorithm is trained to recognize the faces. The system is designed to remain passive if a recognized face is detected; however, if an unrecognized face is detected, it triggers an alert for security intervention.



**Figure 2.3.2 Basic Flowchart of How the System in [8] Works**

Primarily focusing on home security surveillance, the study also undertook a power consumption and price comparison analysis between Raspberry Pi and traditional computers. This aimed to guide homeowners in making informed decisions. Notably, the study's limitation lies in its testing approach, which assessed one person at a time. Consequently, its ability to handle larger crowds remained untested.

## 2.4 Analysis of Face Recognition: DLIB and OpenCV [9]

Two often-used open-source libraries for face recognition are Dlib and OpenCV. Dlib is a C++ library that handles management and manipulation of images. Dlib handles image processing with array2d objects containing any pixel types, including RGB, RGB Alpha, HSI, LAB, or Grayscale. OpenCV is one of the most popular machine learning and computer vision software libraries in the world, and it has more than 2500 optimized algorithms that includes image processing and facial recognition modules. [9] performed detailed analysis of these two software libraries for facial recognition, using metrics like speed and accuracy. The algorithms analyzed from Dlib were Convolved Neural Network (CNN) and Histogram of Oriented Gradients (HoG), while the algorithms from OpenCV were Deep Neural Network (DNN) and HAAR Cascades.

The study found that HoG was the fastest amongst the four algorithms, which took 0.011 seconds per image processed, but had a low accuracy rate of 0% False Acceptance Rate (FAR) and 27.27% False Rejection Rate (FRR). The study also found that DNN scored the highest in the accuracy test with 2.6% FAR and 11.69% FRR, but was the slowest amongst the four algorithms at 0.119 seconds per image processed.

Penguji an	Dlib		OpenCV	
	CNN	HoG	DNN	HAAR
1	1.828	0.469	5.922	3.281
2	1.766	0.578	6.266	2.828
3	1.781	0.859	6.000	2.656
4	1.781	0.484	5.656	2.750
5	1.797	0.453	5.766	2.359
6	1.938	0.438	6.281	2.609
7	1.781	0.500	6.031	2.656
8	2.047	0.516	5.625	2.500
9	1.828	0.516	6.000	2.609
10	1.766	0.453	5.969	2.422
<b>Rata-rata (detik)</b>	1.831	0.527	5.952	2.667
<b>Rata-rata per gambar (detik)</b>	0.037	0.011	0.119	0.053

	Dlib		OpenCV	
	CN N	HoG	DN N	HAA R
<b>Total detections</b>	119	112	136	101
<b>False rejection</b>	35	42	18	53
<b>False acceptance</b>	0	0	4	1
<b>FRR (%)</b>	22.7 3	27.2 7	11.69	34.42
<b>FAR (%)</b>	0.00	0.00	2.60	0.65

**Figure 2.4.1 Results from the Speed Tests and Accuracy Tests of CNN, HoG, DNN, HAAR algorithms conducted by [9]**

The study provided great insights into the different strengths and weaknesses of each algorithm for the usage of facial recognition. There was no conclusive evidence of which algorithm is the best, as each had their own features that were suited to different use-cases. Developers would have to determine their project objectives in order to pick the best algorithm for their development. HoG and HAAR works best with cameras placed at eye-level where faces will have minimal pose variance in front of the camera, while DNN, CNN, and HoG are suitable for cameras placed in various angles like in the case of CCTV cameras.

HoG may be the fastest algorithm, but it is weak in accuracy. DNN and CNN are more flexible in terms of detecting obscured faces with pose variants, but requires more high-quality images and processing time, which may not be too suitable for CCTV systems in low-cost establishments. Finally, HAAR Cascades is the third fastest algorithm and handles low quality images very well which is suitable of CCTV cameras but may be pose dependent.

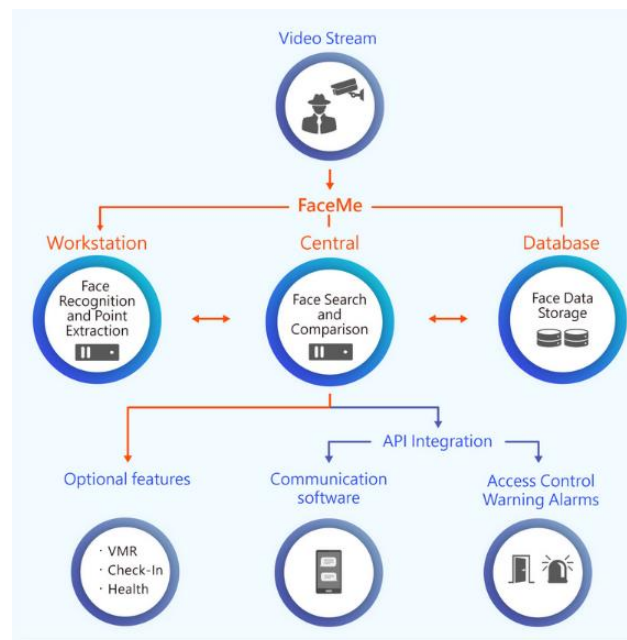
<b>Urutan</b>	<b>Kecepatan (sekon)</b>	<b>Akurasi (FRR%; FAR%)</b>
1	HoG (0.011)	DNN (11.69; 2.6)
2	CNN (0.037)	CNN (22.73; 0)
3	HAAR (0.053)	HoG (27.27, 0)
4	DNN (0.119)	HAAR (34.42, 0.65)

**Figure 2.4.2 Summary of Processing Speed (per second) and Accuracy (FRR and FAR percentages) in [9]**

## 2.5 FaceMe® Security System by CyberLink Corporation [10]

CyberLink Corp, a Taiwanese software company, has developed one of the world's premier AI and facial recognition engines named FaceMe®. This engine is available in multiple forms, including an SDK (software development kit), an API (application programming interface) platform, and integrated solutions tailored for specific requirements [10]. CyberLink positions this technology as a business solution, especially in the security domain, asserting its capability to address diverse security needs. These encompass secure access controls, AI-powered intelligent monitoring, functioning as a VMS (video management system) plug-in, issuing notifications and alerts, and accommodating facial coverings and gear wearing [11].

The standout feature of the FaceMe® engine lies in its adaptability, closely followed by its extensive array of functionalities. Its seamless integration with pre-existing video management systems (VMS) employed by businesses for security surveillance adds to its appeal and scalability. The engine utilizes AI to continuously monitor faces, conducting real-time matching against blacklist or whitelist entries and issuing smartphone notifications upon a match detection. The FaceMe® system boasts flexibility in both scalability and features, and its smooth integration with existing VMS and other communication systems should be a prime consideration in the development of the project.



**Figure 2.5 FaceMe® Security Solution's System Structure [11]**

The system's developer-centric central console incorporates load balancing and failover functionalities to distribute workload across workstations, ensuring efficiency and effectiveness. This prevents system overload when crowd numbers surge, avoiding tracking challenges caused by excessive foot traffic. Continuous connectivity with Microsoft's SQL database enables face matches for recognized individuals. Moreover, the system's versatility extends to incorporating APIs for additional features like datetime and attendance tracking. Developer tools within the FaceMe® platform encompass comprehensive management requirements, including record management, data analysis, and person database administration.

FaceMe® earned recognition as a premier facial recognition technology in the 2021 Face Recognition Vendor Test (FRVT) by the United States' National Institute of Standards and Technology (NIST) [12]. Notably, FaceMe® excels in both accuracy and speed, even under suboptimal lighting conditions and in the presence of facial accessories. The technology identifies faces by extracting n-dimensional vector sets and subsequently comparing them with connected databases or previously identified faces. The algorithm's precision extends beyond capturing basic human features such as age and race—it also demonstrates robust anti-spoofing capabilities. Anti-spoofing safeguards against security fraud where individuals attempt to deceive the system using face substitutes like masks or photos. FaceMe's ability to counteract this challenge is a significant achievement, considering the gravity of anti-spoofing vulnerabilities.

The excellence of FaceMe® has propelled the evolution of face recognition technology. Serving as a pioneering solution, the software has been adopted by numerous global organizations, including financial services company Good Finance, which employs it for eKYC (Electronic Know Your Customer) processes in online banking services [13]. Lessons from FaceMe's rendition of facial recognition technology can be extrapolated to enhance crowd security measures, particularly in the domains of identity validation and tracking.

## 2.6 Public Security by Avigilon Corporation [14]

Avigilon Corporation, a subsidiary of Motorola Solutions based in Canada, specializes in video surveillance software, equipment, and various access control devices. Avigilon offers a comprehensive ecosystem of video security solutions, encompassing a wide range of products in video infrastructure, software, and tailored systems to address diverse needs. One noteworthy aspect of Avigilon's approach is the integration of facial recognition and video analytics directly into their camera products, ensuring simplified installation and user-friendly operation. Their Artificial Intelligence and video analytics offerings encompass multiple features, including facial recognition, appearance search, unusual activity, or motion detection, among others [14].

At the heart of Avigilon's offering is the Avigilon Control Center (ACC), a cloud-connected software solution that seamlessly integrates their cameras, establishing a unified video surveillance system enhanced by AI-driven video analytics. Within ACC, security personnel can create a secure watch list or a database of individuals of interest. When these designated individuals are detected by the cameras, the system triggers alerts. ACC also leverages edge-based technology and intelligence for features like Unusual Activity Detection and Unusual Motion Detection. This approach enables rapid real-time analysis and performance, as the software processes data in close proximity to the ACC location. This localization conserves resources and enhances the efficiency of video feed analysis and notification dissemination.



**Figure 2.6 Avigilon's Focus of Attention (FoA) user navigation screen [14]**

Avigilon employs a self-learning algorithm that effectively distinguishes and categorizes various events, assigning them different-colored nodes for clear identification. This approach empowers users to swiftly assess and determine the significance of each flagged event without the need to sift through numerous camera screens. Instead, all alerts are conveniently displayed on a user-friendly dashboard, streamlining decision-making.

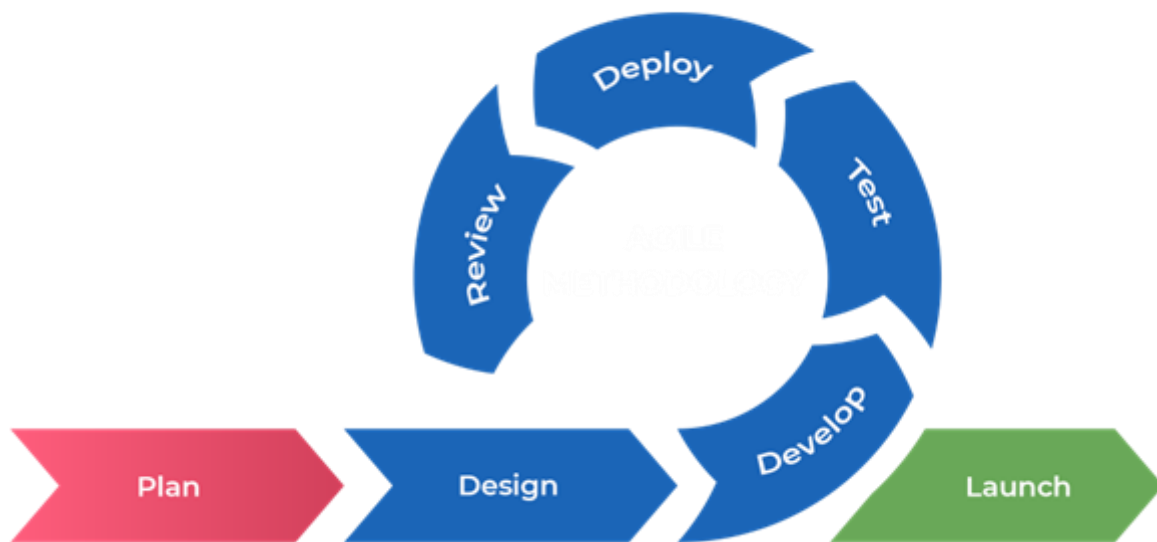
The system grants security personnel the capability to personalize watch lists based on their specific security requirements. This can be achieved by uploading well-defined reference images of individuals or utilizing the advanced Appearance Search feature. This cutting-edge technology rapidly scans through extensive hours of video feed, using textual descriptions, photos, or video content to locate specific persons. The management of watch lists is endowed with robust access controls to ensure data integrity. Notably, Avigilon's ACC retains the identities of individuals found in watch list matches, facilitating convenient record-keeping.

The process of managing secure watch lists through ACC is not only user-friendly and efficient, but it's also highly adaptable. Different faces within the database can be earmarked for distinct alarm triggers. For instance, individuals on a criminal blacklist can prompt one type of alert, while VIPs on a whitelist can trigger another. This level of flexibility simplifies and enhances the accuracy of security personnel's tasks, sparing them the effort of validating visual data on camera feeds.

# Chapter 3

## System Methodology and Approach

### 3.1 Methodology



**Figure 3.1 Agile Methodology**

The Agile methodology was adopted for project management due to its flexibility and speed in the development process. This approach allows for ongoing testing throughout the development cycle, ensuring the production of a high-quality product by promptly addressing any needed changes or potential issues. Agile is particularly suitable for small development teams, making it a fitting choice for this one-person project.

The project development process consists of the following steps, which are repeated continuously:

- i. Analysis and evaluation of system.
- ii. Planning for system modifications or process improvements.
- iii. Development and execution based on discussion and evaluation.
- iv. System testing.
- v. Meeting with the project supervisor.



Bi-weekly meetings were conducted with the project supervisor after each iteration. These meetings served as a platform to discuss challenges encountered and to generate new ideas. Subsequently, these ideas were explored and implemented over the following two weeks. This iterative and collaborative approach allows for the project's evolution and adaptation based on feedback and emerging insights.

### 3.2 Project Timeline

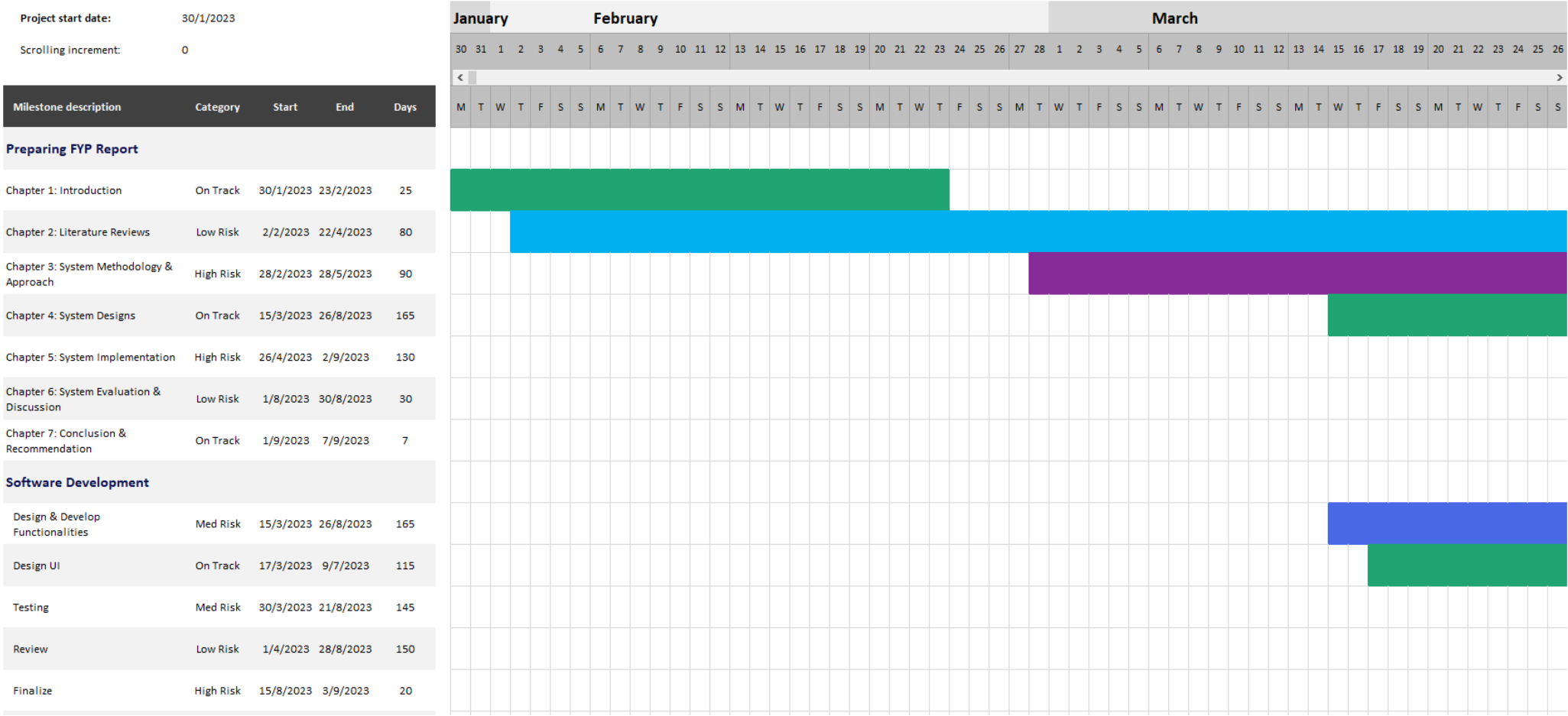


Figure 3.2.1 Gantt Chart of Project (1/4)

Project start date: 30/1/2023  
 Scrolling increment: 56

Milestone description	Category	Start	End	Days
<b>Preparing FYP Report</b>				
Chapter 1: Introduction	On Track	30/1/2023	23/2/2023	25
Chapter 2: Literature Reviews	Low Risk	2/2/2023	22/4/2023	80
Chapter 3: System Methodology & Approach	High Risk	28/2/2023	28/5/2023	90
Chapter 4: System Designs	On Track	15/3/2023	26/8/2023	165
Chapter 5: System Implementation	High Risk	26/4/2023	2/9/2023	130
Chapter 6: System Evaluation & Discussion	Low Risk	1/8/2023	30/8/2023	30
Chapter 7: Conclusion & Recommendation	On Track	1/9/2023	7/9/2023	7
<b>Software Development</b>				
Design & Develop Functionalities	Med Risk	15/3/2023	26/8/2023	165
Design UI	On Track	17/3/2023	9/7/2023	115
Testing	Med Risk	30/3/2023	21/8/2023	145
Review	Low Risk	1/4/2023	28/8/2023	150
Finalize	High Risk	15/8/2023	3/9/2023	20

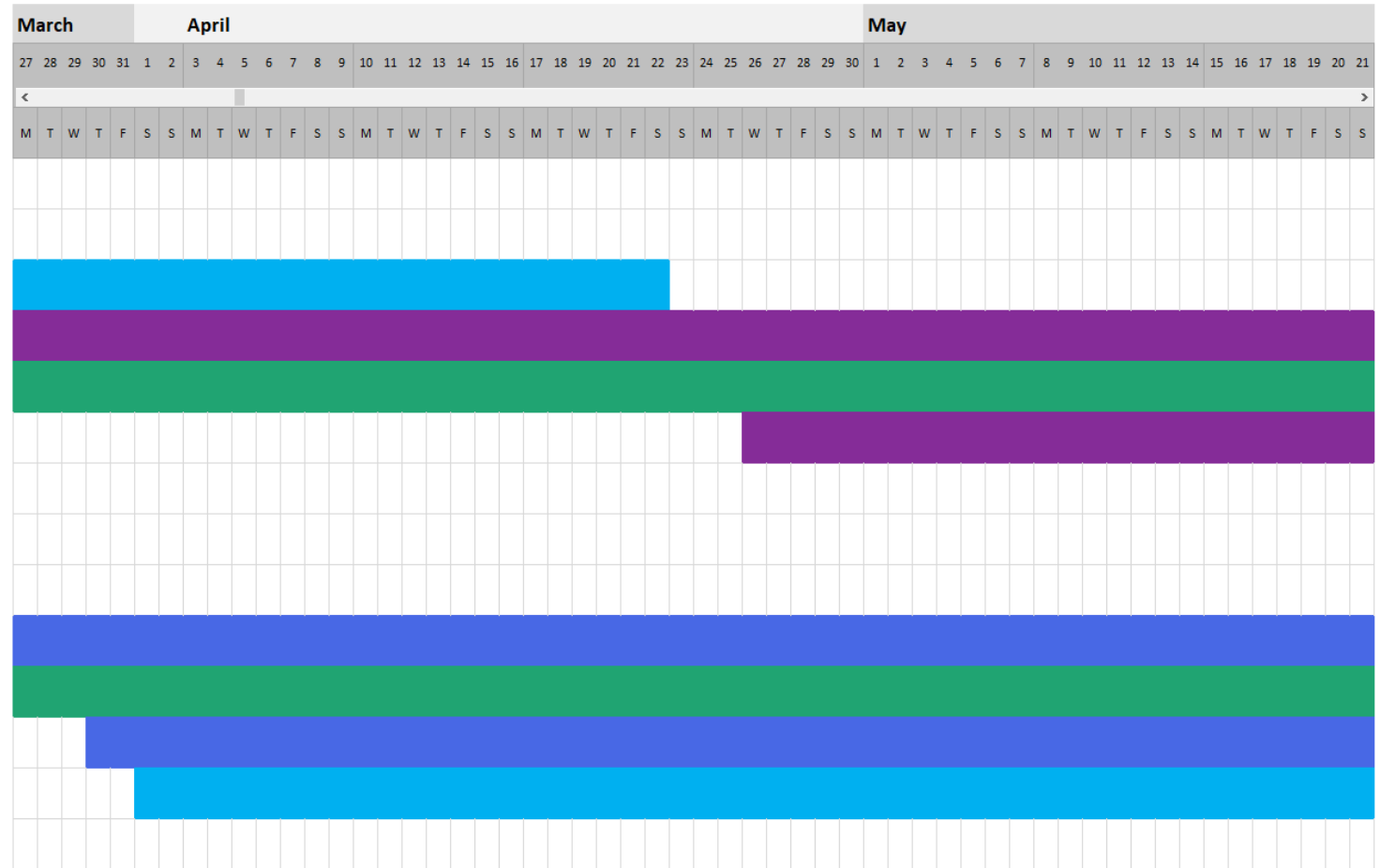


Figure 3.2.2 Gantt Chart of Project (2/4)

Project start date: 30/1/2023  
 Scrolling increment: 112

Milestone description	Category	Start	End	Days
<b>Preparing FYP Report</b>				
Chapter 1: Introduction	On Track	30/1/2023	7/2/2023	25
Chapter 2: Literature Reviews	Low Risk	2/2/2023	10/2/2023	80
Chapter 3: System Methodology & Approach	High Risk	28/2/2023	28/5/2023	90
Chapter 4: System Designs	On Track	15/3/2023	26/8/2023	165
Chapter 5: System Implementation	High Risk	26/4/2023	2/9/2023	130
Chapter 6: System Evaluation & Discussion	Low Risk	1/8/2023	30/8/2023	30
Chapter 7: Conclusion & Recommendation	On Track	1/9/2023	7/9/2023	7
<b>Software Development</b>				
Design & Develop Functionalities	Med Risk	15/3/2023	26/8/2023	165
Design UI	On Track	17/3/2023	9/7/2023	115
Testing	Med Risk	30/3/2023	21/8/2023	145
Review	Low Risk	1/4/2023	28/8/2023	150
Finalize	High Risk	15/8/2023	3/9/2023	20

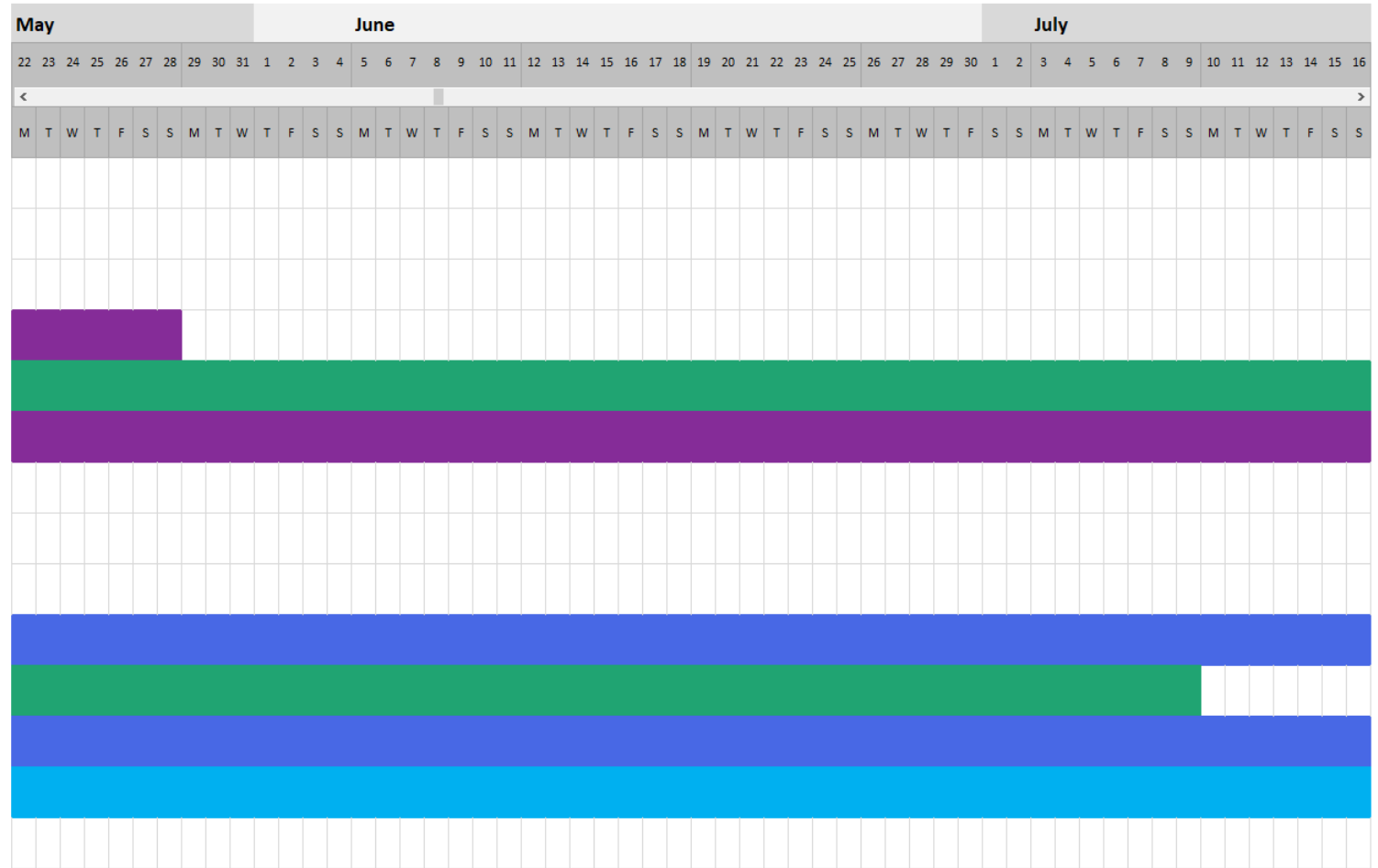


Figure 3.2.3 Gantt Chart of Project (3/4)

Project start date: 30/1/2023  
 Scrolling increment: 168

Milestone description	Category	Start	End	Days
<b>Preparing FYP Report</b>				
Chapter 1: Introduction	On Track	30/1/2023	7/2/2023	25
Chapter 2: Literature Reviews	Low Risk	2/2/2023	10/2/2023	80
Chapter 3: System Methodology & Approach	High Risk	28/2/2023	28/5/2023	90
Chapter 4: System Designs	On Track	15/3/2023	26/8/2023	165
Chapter 5: System Implementation	High Risk	26/4/2023	2/9/2023	130
Chapter 6: System Evaluation & Discussion	Low Risk	1/8/2023	30/8/2023	30
Chapter 7: Conclusion & Recommendation	On Track	1/9/2023	7/9/2023	7
<b>Software Development</b>				
Design & Develop Functionalities	Med Risk	15/3/2023	26/8/2023	165
Design UI	On Track	17/3/2023	9/7/2023	115
Testing	Med Risk	30/3/2023	21/8/2023	145
Review	Low Risk	1/4/2023	28/8/2023	150
Finalize	High Risk	15/8/2023	3/9/2023	20

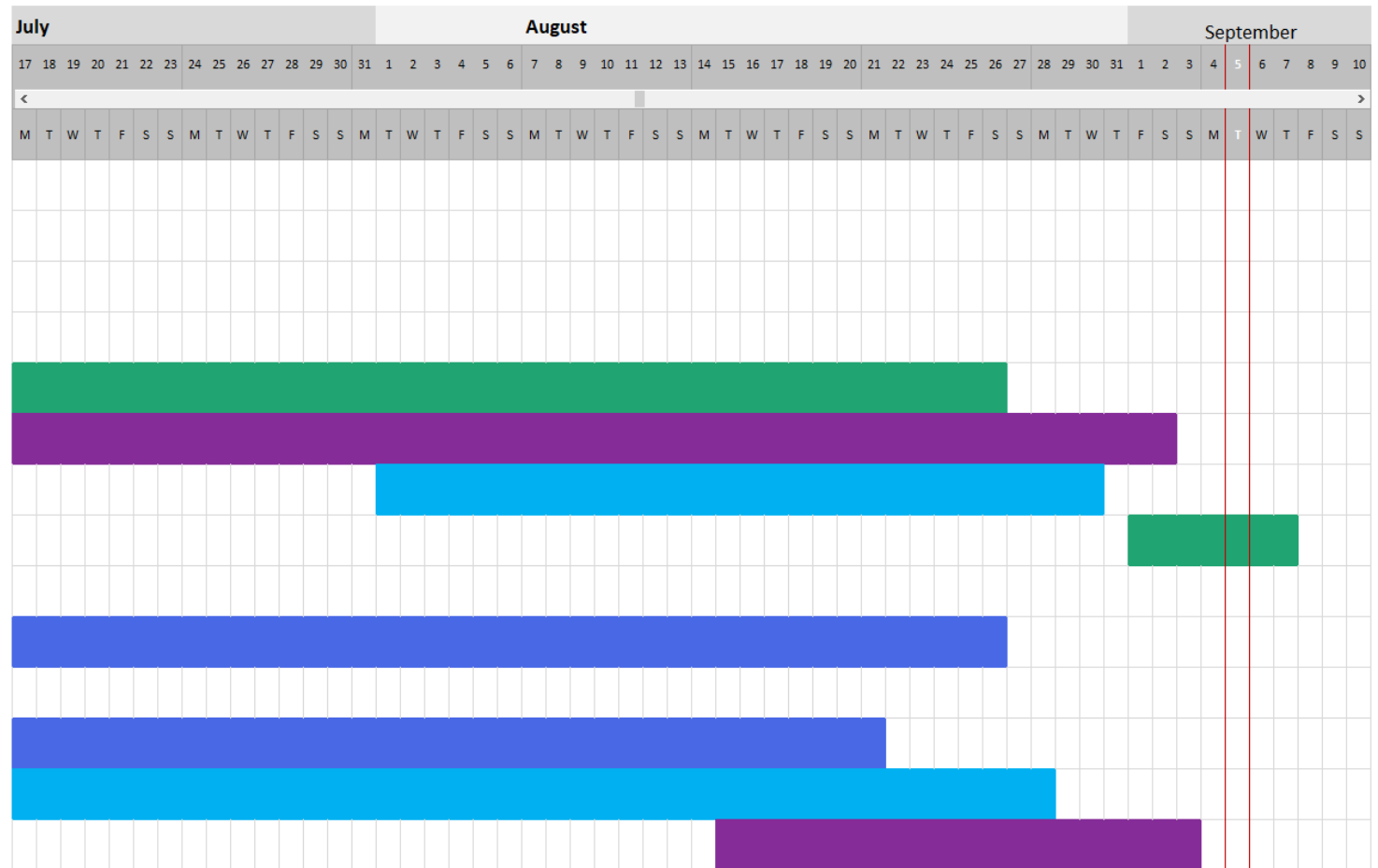
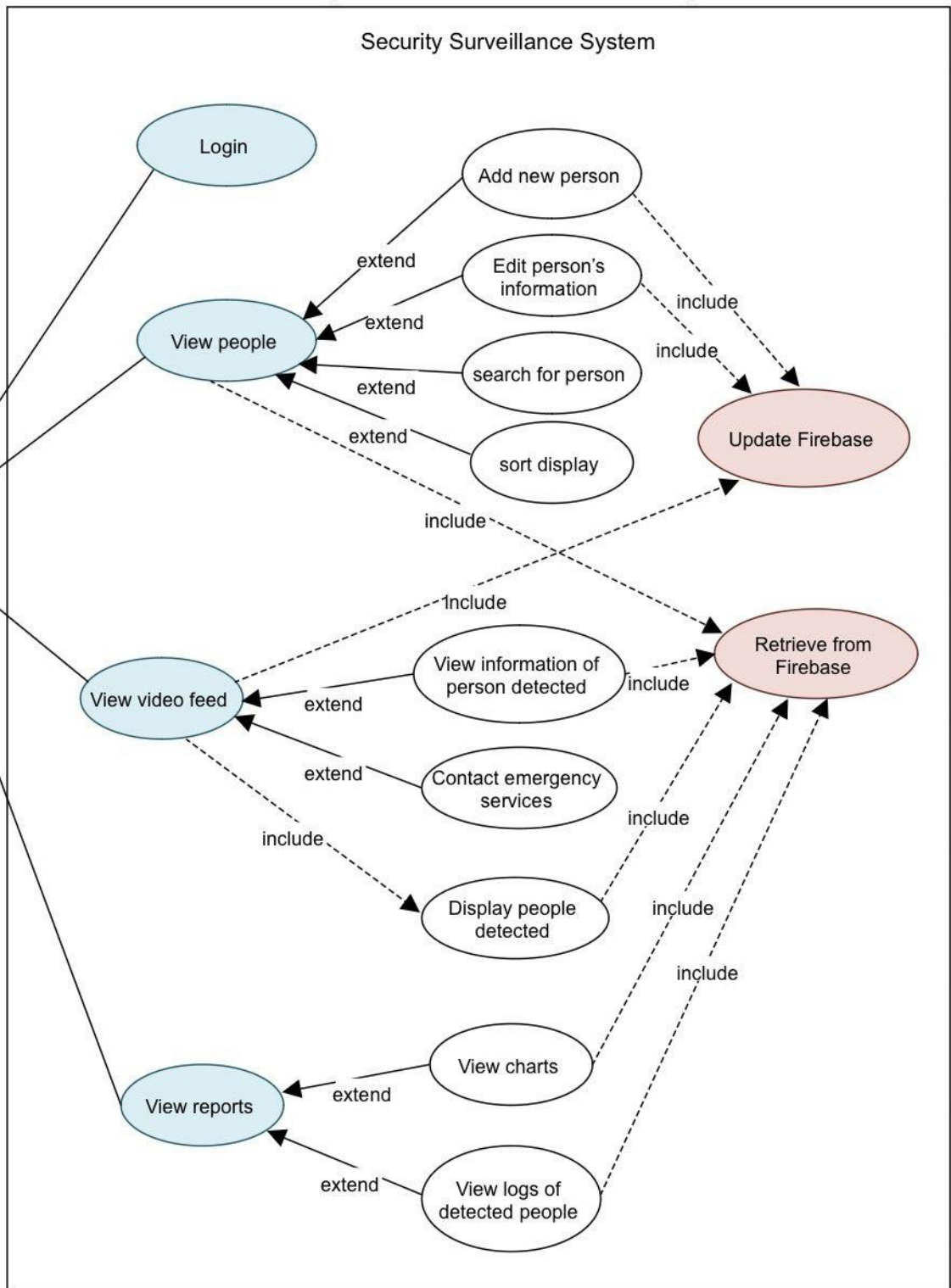


Figure 3.2.4 Gantt Chart of Project (4/4)

### 3.3 System Design Diagram

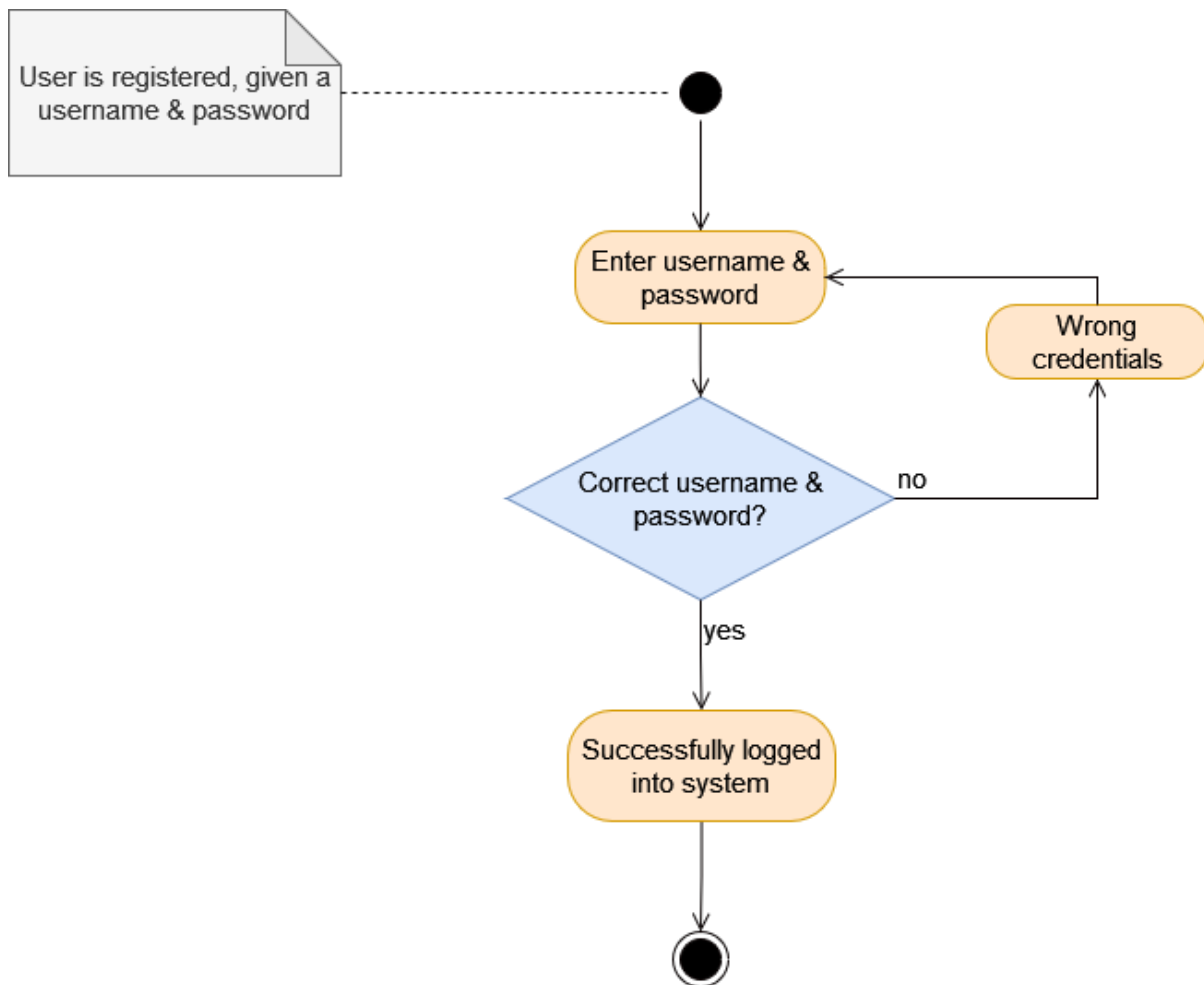


**Figure 3.3 System Use Case Diagram**

### 3.3.1 Login Module

<b>Use Case Name:</b> Login to System	<b>ID:</b> 1	<b>Importance Level:</b> High
<b>Primary Actor:</b> Security Administrator	<b>Use Case Type:</b> Detail, Essential	
<b>Stakeholders and Interests:</b> <b>Security Administrator:</b> log into security surveillance system		
<b>Brief Description:</b> This use case describes how the security administrator logs into their registered account in the surveillance system.		
<b>Trigger:</b> Administrator wants to perform their designated task – surveillance.		
<b>Type:</b> External		
<b>Relationship:</b> <b>Association:</b> Administrator <b>Include:</b> <b>Extend:</b> <b>Generalization:</b>		
<b>Normal Flow of Events:</b> <ol style="list-style-type: none"> <li>1. Administrator enters their assigned username and password to log in.</li> <li>2. If the username or password is not correct, the system will prompt administrator to try again.</li> <li>3. If the username and password entered is correct, the administrator will be able to enter the system.</li> </ol>		
<b>Sub Flows:</b> Not applicable		
<b>Alternate/ Exceptional Flows:</b> Not applicable		

**Table 3.3.1 Login Module – Use Case Description**



**Figure 3.3.1 Login Module Activity Diagram**

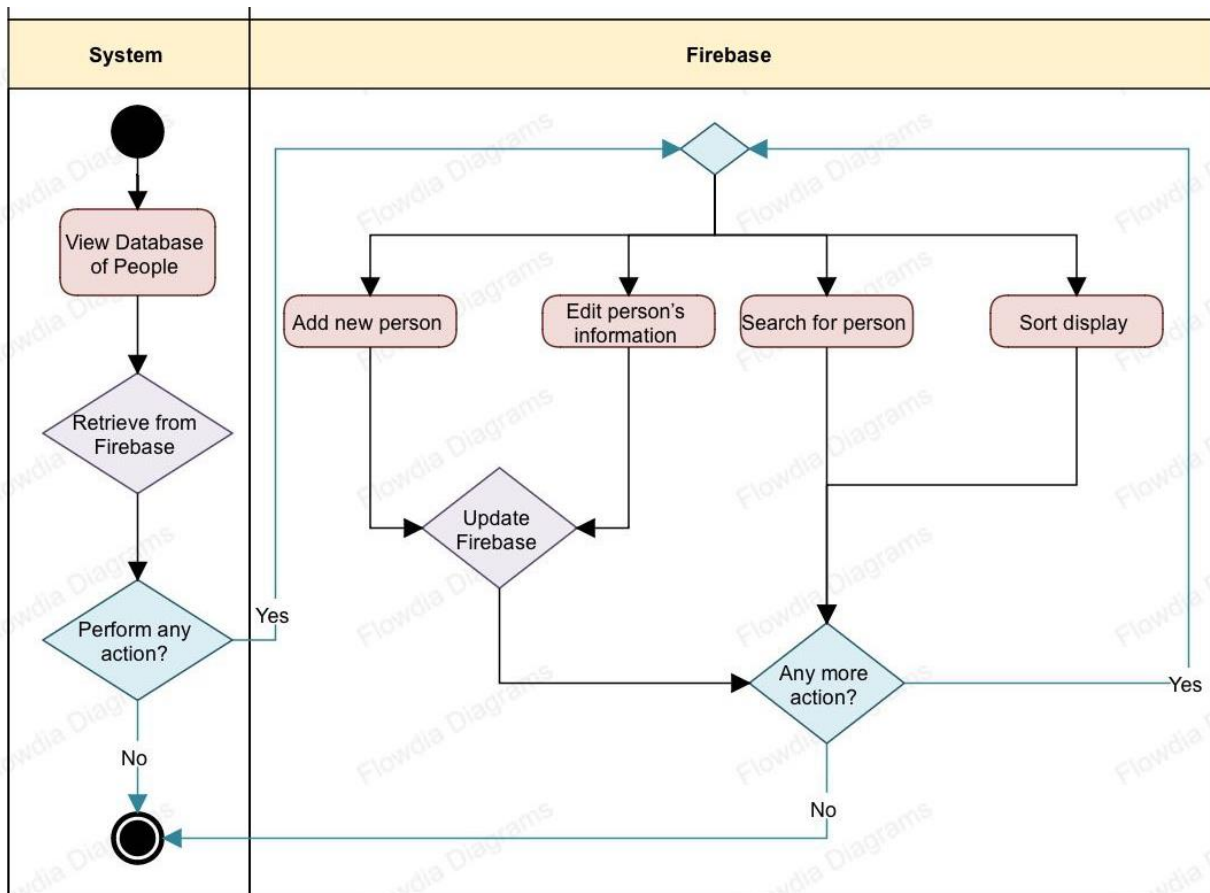
The user, which is a verified security personnel or administrator, will not be allowed to freely create accounts. An account will be created for them for security purposes, and they will be given their username and password. This is because this system is only for specific security personnel or administrators to enter. To login into the system, the administrator will enter their registered username and password. The system will verify that the credentials are correct. If either of them is wrong, the user will not be logged into the system and be prompted to try again. If the credentials are correct, they will be logged into the system and enter to the main dashboard.



### 3.3.2 View People Module

<b>Use Case Name:</b> View Database of People	<b>ID: 2</b>	<b>Importance Level: High</b>
<b>Primary Actor:</b> Security Administrator	<b>Use Case Type: Detail, Essential</b>	
<b>Stakeholders and Interests:</b> <b>Security Administrator:</b> to view all persons logged into the database and make changes like adding or editing the information of people.		
<b>Brief Description:</b> This use case describes how the security administrator enters the database of people to perform a few tasks like adding or editing information of people.		
<b>Trigger:</b> Administrator wants to view the database of people or make changes to the database.  <b>Type: Internal</b>		
<b>Relationships:</b> <b>Association:</b> Administrator <b>Include:</b> Update Firebase, retrieve from Firebase <b>Extend:</b> Add new person, edit person's information, search for person, sort display		
<b>Normal Flow of Events:</b> <ol style="list-style-type: none"> <li>1. Administrator opens the database to view all people.</li> <li>2. If administrator performs any changes to the database, the database will automatically be updated.</li> <li>3. If administrator has no actions to perform, they can exit the database and go back to surveillance.</li> </ol>		
<b>Sub Flows:</b> Not applicable		
<b>Alternate/ Exceptional Flows:</b> Not applicable		

**Table 3.3.2 View People Module – Use Case Description**



**Figure 3.3.2 View People Module – Activity Diagram**

When the admin opens the database of people, the system will retrieve records of all the people that are stored in Firebase and display it for the admin to view. If the admin does not perform any action other than to look at the information, they can just exit the database and the activity ends.

If the admin wants to perform any action, the admin can add a new person into the database or edit an existing person's information. This will automatically update Firebase and the new records will be displayed in the system. The admin can also search for any person based on information like name, nationality, status, and so on, and can also sort the display based on alphabetical order, status, or last detected. The admin can repeat any of these actions until no more actions are to be taken, and the activity ends when the admin exits the database.

### 3.3.3 View Video Feed Module

<b>Use Case Name:</b> View Video Feed	<b>ID: 3</b>	<b>Importance Level: High</b>
<b>Primary Actor:</b> Security Administrator	<b>Use Case Type: Detail, Essential</b>	
<b>Stakeholders and Interests:</b> <b>Security Administrator:</b> to perform surveillance on all persons in the premises through live video feed from connected security cameras.		
<b>Brief Description:</b> This use case describes how the security administrator performs surveillance by monitoring the live video feed from connected cameras.		
<b>Trigger:</b> Administrator wants to perform surveillance.  <b>Type: Internal</b>		
<b>Relationships:</b> <b>Association:</b> Administrator <b>Include:</b> Update Firebase, display people detected <b>Extend:</b> View information of person detected, contact emergency services		
<b>Normal Flow of Events:</b> <ol style="list-style-type: none"> <li>1. Administrator opens live video feed from connected cameras.</li> <li>2. If there are no suspicious persons detected, administrator does not have to perform any action and may continue to just monitor the video feed.</li> <li>3. If suspicious persons are detected, administrator may view more information on any possible criminal records of the person.</li> <li>4. If there is no cause for concern, administrator may continue to just monitor the video feed.</li> <li>5. If there is cause for concern, administrator may not decide to contact emergency services but continue observing the person.</li> </ol>		
<b>Sub Flows:</b> Not applicable		
<b>Alternate/ Exceptional Flows:</b> Not applicable		

**Table 3.3.3 View Video Feed Module – Use Case Description**

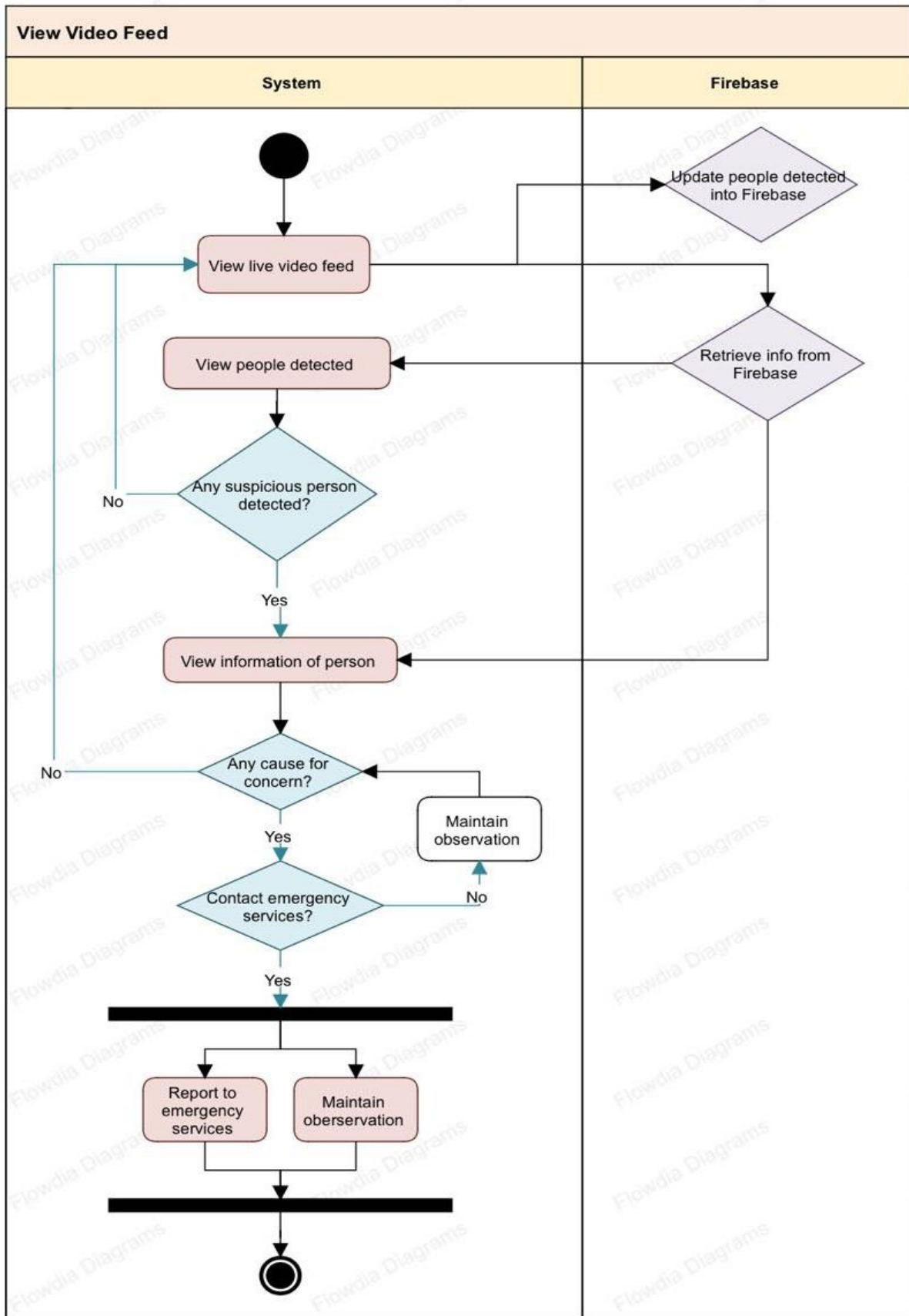


Figure 3.3.3 View Video Feed Module – Activity Diagram

The figure 3.3.3 illustrates the activity diagram of the admin viewing the live video feeds from multiple cameras. The video feeds will be continuously updating the Firebase with the people detected, and the system will display an updated list of people currently being detected on the cameras by retrieving the information from Firebase. Considering that the primary responsibility of the administrator or security personnel accessing the system is typically limited to monitoring safety, the ideal scenario is one where no actions are required while observing the video feed. In such a situation, it implies that every individual in the public area remains unharmed by any potentially suspicious or hazardous individuals.

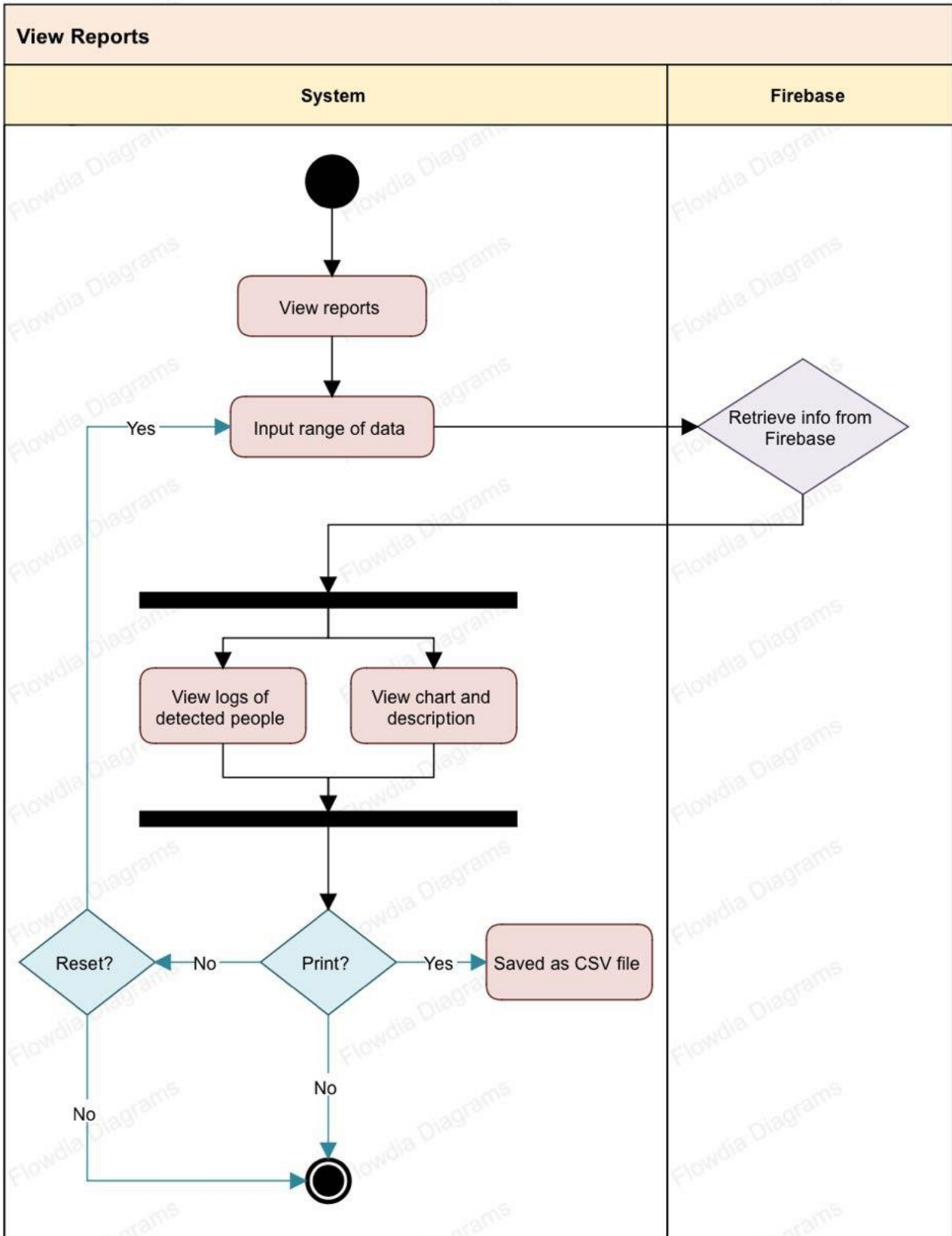
However, in the event of any suspicious person being detected, like a missing person, the admin will click to view detailed information of the person and determine if there is any cause for concern. If there is, the admin will have to determine if they should contact emergency services. If they do not, the admin should maintain observation. If the situation warrants contact with any emergency number like the head of security or law enforcement, the admin should maintain observation of the person while reporting to keep track of the person.

This activity ends when emergency services arrive on scene to handle the situation.

### 3.3.4 View Reports Module

<b>Use Case Name:</b> View Reports	<b>ID: 4</b>	<b>Importance Level:</b> <b>Medium</b>
<b>Primary Actor:</b> Security Administrator	<b>Use Case Type: Detail, Essential</b>	
<b>Stakeholders and Interests:</b> <b>Security Administrator:</b> to generate and view analytical reports of detected people for any decision making.		
<b>Brief Description:</b> This use case describes how the security administrator can generate personalized reports based on the database.		
<b>Trigger:</b> Administrator wants to generate reports.  <b>Type: Internal</b>		
<b>Relationships:</b> <b>Association:</b> Administrator <b>Include:</b> Retrieve from Firebase <b>Extend:</b> View charts, view logs of detected people		
<b>Normal Flow of Events:</b> <ol style="list-style-type: none"> <li>1. Administrator enters the reports generation section from dashboard.</li> <li>2. Admin will input the data range that they want to know more information on, like range of date and time, location, or status.</li> <li>3. The system will display the logs of detected people based on the input data and generate a chart and description based on the log.</li> <li>4. Admin can choose to print the log, which will generate a CSV file.</li> <li>5. Admin can also reset the input data fields and generate a new report.</li> </ol>		
<b>Sub Flows:</b> Not applicable		
<b>Alternate/ Exceptional Flows:</b> Not applicable		

**Table 3.3.4 View Reports Module – Use Case Description**



**Figure 3.3.4 View Reports - Activity Diagram**

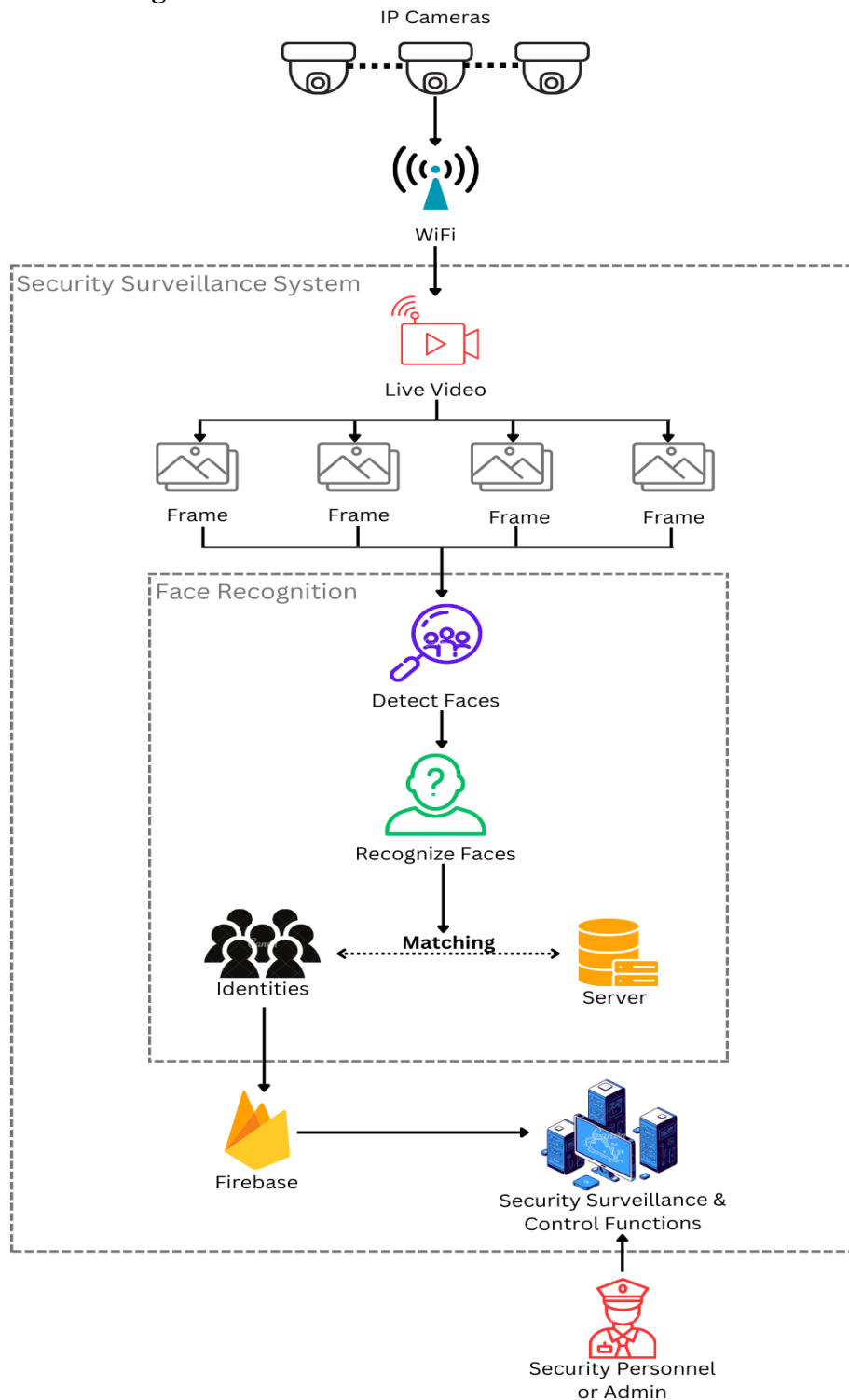
The figure 3.3.4 shows how the administrator can generate and look at personalized reports based on the information in the database. When the admin clicks on view reports, they will be prompted to input data fields for data ranges like date time, location, or status. The system will then retrieve the data from Firebase based on the data input. The system will display the logs of detected people and generate a chart and description based on that. The admin can choose whether they want to print this. If they do, the system will generate a CSV file of the log based on their input data. Otherwise, the admin can choose to reset their input data and generate new reports.



# Chapter 4

## System Design

### 4.1 System Block Diagram



**Figure 4.1 System Block Diagram**

Numerous IP cameras will be configured to video designated locations in the premises and will be connected through WIFI, although the application of this in real scenarios would be through private networks to avoid intrusion from unauthorized people. The video feed will be sent into the surveillance system and each frame will be extracted one by one. This is so that the face recognition system can detect faces on each frame.

This project uses Dlib's pre-trained CNN-based face detector model, which is the `mmod_human_face_detector.dat` model, and it also implements Max Margin Object Detection (MMOD) for better results. For the image-preprocessing steps, images of labelled faces are loaded into the system's local storage for the model to encode and perform matching. All images are also resized to have a fixed size of 216x216 pixels while maintaining the aspect ratio. The model also identifies faces using landmarks by two models that were also used in this project which are `shape_predictor_5_face_landmarks.dat` and `shape_predictor_68_face_landmarks.dat`. The model using 5 facial landmarks identifies the bottom of the nose and corners of the eyes and is used when the batch size is smaller. When the batch size is bigger, the model using 68 face landmarks will be used instead for higher accuracy. The video frames are also converted from BGR to greyscale to minimize noise and increase computation speed.

Once there are faces detected in the video frames, the system will find the closest match in the encoded labelled faces and recognize the face detected in the video frame. With the recognized face, the system will log the date, time, name of the person based on their label, and location into Firebase. The system is then able to perform various functions by accessing these logs.

## 4.2 System Flow Description

### 4.2.1 Main.py

```
1 from PySide2.QtWidgets import QApplication, QDialog, QMainWindow, QMessageBox
2 from GUI.loginui import Ui_LoginDialog, Ui_MainWindow
3 from GUI.dashboardui import DashboardDialog
4 import firebase_admin
5 from firebase_admin import credentials
6 from firebase_admin import db
7 from firebase_admin import storage
8
9 ...
14
15 class LoginDialog(QDialog):
16     def __init__(self):
17         super().__init__()
18         self.ui = Ui_LoginDialog()
19         self.ui.setupUi(self)
20         self.ui.loginButton.clicked.connect(self.login)
21
22     def login(self):
23         username = self.ui.usernameEdit.text()
24         password = self.ui.passwordEdit.text()
25
26         if username == "admin" and password == "admin":
27             QMessageBox.information(self, "Login Successful", "Login Successful!")
28             self.openMainWindow()
29         # else:
30         #     QMessageBox.warning(self, "Login Failed", "Invalid username or password!")
31
32     def openMainWindow(self):
33         self.dashboard = DashboardDialog()
34         self.dashboard.show()
35
36         self.accept()
37
38 def main():
39     app = QApplication([])
40     dialog = LoginDialog()
41     dialog.show()
42     app.exec_()
43
44 if __name__ == "__main__":
45     main()
```

Figure 4.2.1 Main.py

Upon starting the system, the user is prompted to log in with their assigned credentials. Users are not permitted to create accounts themselves as this is a security system with restricted access. If the user enters the correct credentials, in this case it is “admin” for both username and password, the user will be successfully logged in and will be directed to the dashboard. If the credentials are incorrect, the user will be prompted to try again with a warning dialog box.

## 4.2.2 Dashboardui.py

```
1 from datetime import datetime
2 from PySide2.QtCore import *
3 from PySide2.QtGui import *
4 from PySide2.QtWidgets import *
5 import cv2
6 import subprocess
7 from facerecogniser import FaceRecognition
8 from PySide2.QtWidgets import QDialog
9 from peopledbui import PeopleDBDialog
10 from allcams import CamsDialog
11 from reports import ReportsDialog
12 from detectLog import append_to_detect_log
13 import firebase_admin
14 from firebase_admin import credentials
15 from firebase_admin import db
16 from firebase_admin import storage
17
18 from GUI.init_firebase import firebase_admin
19 app = firebase_admin.get_app()
20
21 sfr = FaceRecognition()
22 sfr.load_encoding_images("images/")
23 detected = []
24
25 class DashboardDialog(QDialog):
26     def __init__(self):
27         super().__init__()
28         self.ui = Ui_Dialog()
29         self.ui.setupUi(self)
30         self.ui.people.clicked.connect(self.open_people_db) # to open peopledbui.py
31         self.ui.checkcam.clicked.connect(self.open_allcams)
32
33     def open_people_db(self):
34         self.people_db = PeopleDBDialog()
35         self.people_db.show()
36
37     def open_allcams(self):
38         # Provide the required argument to CamsDialog
39         detectlog = QListView()
40         self.all_cams = CamsDialog(detectlog)
41         self.all_cams.show()
42
43     def update_datetime_label(self):
44         current_datetime = QDateTime.currentDateTime().toString("hh:mm:ss \nMM/dd/yyyy")
45         self.ui.datetime_label.setText(current_datetime)
46         QTimer.singleShot(1000, self.update_datetime_label)
47
48 class Ui_Dialog(object):
49     def setupUi(self, Dialog):
50         if not Dialog.setObjectName():
51             Dialog.setObjectName("Dialog")
52         Dialog.resize(1123, 630)
53         self.cam = QFrame(Dialog)
54         self.cam.setObjectName("cam")
55         self.cam.setGeometry(QRect(540, 30, 700, 380))
56         self.cam setFrameShape(QFrame.StyledPanel)
57         self.cam setFrameShadow(QFrame.Raised)
58
```

Figure 4.2.2.1 Dashboardui.py (1/4)

```

59     self.video_player = VideoPlayer(self.cam)
60     self.video_player.setGeometry(self.cam.rect())
61     self.cam_layout = QVBoxLayout(self.cam)
62     self.cam_layout.addWidget(self.video_player)
63
64     self.checkcam = QPushButton(Dialog)
65     self.checkcam.setObjectName(u"checkcam")
66     self.checkcam.setGeometry(QRect(260, 80, 211, 51))
67
68     self.people = QPushButton(Dialog)
69     self.people.setObjectName(u"people")
70     self.people.setGeometry(QRect(260, 170, 211, 51))
71
72     self.detectedface = QFrame(Dialog)
73     self.detectedface.setObjectName(u"detected")
74     self.detectedface.setGeometry(QRect(30, 370, 461, 231))
75     self.detectedface setFrameShape(QFrame.StyledPanel)
76     self.detectedface.setFrameShadow(QFrame.Raised)
77     self.detected_layout = QVBoxLayout(self.detectedface)
78     self.detected_log = QTextEdit(self.detectedface)
79     self.detected_layout.addWidget(self.detected_log)
80
81     self.datetime = QWidget(Dialog)
82     self.datetime.setObjectName(u"datetime")
83     self.datetime.setGeometry(QRect(10, 50, 221, 141))
84     self.datetime.setStyleSheet(u"")
85     self.datetime_label = QLabel(self.datetime)
86     self.datetime_label.setObjectName(u"datetime_label")
87     self.datetime_label.setGeometry(QRect(0, 0, self.datetime.width(), self.datetime.height()))
88     self.datetime_label.setAlignment(Qt.AlignCenter)
89     self.datetime_label.setStyleSheet(u"font-size: 24px;")
90     self.update_datetime_label()
91
92     self.admin = QWidget(Dialog)
93     self.admin.setObjectName(u"admin")
94     self.admin.setGeometry(QRect(9, 10, 221, 41))
95     self.admin_label = QLabel(self.admin)
96     self.admin_label.setObjectName(u"admin_label")
97     self.admin_label.setGeometry(QRect(0, 0, self.admin.width(), self.admin.height()))
98     self.admin_label.setAlignment(Qt.AlignCenter)
99     self.admin_label.setStyleSheet(u"font-size: 16px; font-weight: bold;")
100    self.admin_label.setText("Administrator: Admin")
101
102    self.graph = QWidget(Dialog)
103    self.graph.setObjectName(u"graph")
104    self.graph.setGeometry(QRect(580, 370, 481, 231))
105    self.button_check_reports = QPushButton("Check Out Reports", self.graph)
106    self.button_check_reports.setGeometry(145, 90, 150, 30)
107    self.button_check_reports.clicked.connect(self.open_reports)
108
109    self.retranslateUi(Dialog)
110
111    QMetaObject.connectSlotsByName(Dialog)
112    # setupUi
113
114    def retranslateUi(self, Dialog):
115        Dialog.setWindowTitle(QCoreApplication.translate("Dialog", u"Dialog", None))
116        self.checkcam.setText(QCoreApplication.translate("Dialog", u"Check Cameras", None))

```

**Figure 4.2.2 Dashboardui.py (2/4)**

```

117     self.people.setText(QCoreApplication.translate("Dialog", u"Database of People", None))
118     # retranslateUi
119
120     def update_datetime_label(self):
121         current_datetime = QDateTime.currentDateTime().toString(u"hh:mm:ss \nMM/dd/yyyy")
122         self.datetime_label.setText(current_datetime)
123         QTimer.singleShot(1000, self.update_datetime_label)
124
125     def open_reports(self):
126         reports_dialog = ReportsDialog()
127         reports_dialog.exec_()
128
129     class VideoPlayer(QLabel):
130     def __init__(self, detected_log, parent=None):
131         super(VideoPlayer, self).__init__(parent)
132         self.video_capture = cv2.VideoCapture(1)
133         self.timer = QTimer()
134         self.timer.timeout.connect(self.next_frame)
135         self.timer.start(30)
136         self.detected_log = detected_log
137
138     def get_person_info(self, name):
139         ref = db.reference('People')
140         data = ref.get()
141
142         for index, item in enumerate(data):
143             det_name = item.get('name', '')
144             if det_name == name:
145                 return item
146
147         return None
148
149     def next_frame(self):
150         face_cascade = cv2.CascadeClassifier(cv2.data.harcascades + "haarcascade_frontalface_default.xml")
151         ret, frame = self.video_capture.read()
152         if ret:
153             # frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) # Convert the frame from BGR to RGB
154             # gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
155             gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
156
157             # Detect faces
158             faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=4)
159
160             face_locations, face_names = sfr.detect_known_faces(frame)
161
162             # Draw rectangle around the faces
163             for face_loc, name in zip(face_locations, face_names):
164                 # top=y1 left=x1 bottom=y2 right=x2
165                 y1, x1, y2, x2 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]
166
167                 cv2.putText(frame, name, (x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 0), 2)
168                 if "black" in name:
169                     color = (0, 0, 200)
170
171                 elif "vip" in name:
172                     color = (200, 0, 0)
173                 else:
174                     color = (0, 200, 0)

```

**Figure 4.2.2.3 Dashboardui.py (3/4)**

```

175
176     cv2.rectangle(frame, (x1, y1), (x2, y2), color, 4)
177
178     #frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR) # Convert the frame back to BGR
179     # Swap the red and blue channels
180     frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
181     frame[:, :, [0, 2]] = frame[:, :, [2, 0]]
182
183     image = QImage(frame.data, frame.shape[1], frame.shape[0], QImage.Format_RGB888)
184     pixmap = QPixmap.fromImage(image)
185     #self.image_label.setPixmap(pixmap)
186     self.setPixmap(pixmap.scaled(self.width(), self.height(), Qt.KeepAspectRatio))
187
188     # Update the log
189     now = datetime.now()
190     detection_date = now.strftime("%m/%d/%Y")
191     detection_time = now.strftime("%H:%M:%S")
192     location = "Cam2"
193
194     for name in face_names:
195         log_entry = f"{detection_date} | {detection_time} | {name}"
196
197         person_info = self.get_person_info(name)
198         if person_info:
199             status = person_info.get('status', 'Unknown')
200             log_entry += f" | {status}"
201             print("Person Detected Info:", person_info)
202
203         log_entry += f" | {location}"
204
205         append_to_detect_log(name, status, detection_date, detection_time, location)
206         self.parent().parent().ui.detected_log.append(log_entry)
207
208     # Display the frame in the video player
209     frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
210     image = QImage(frame, frame.shape[1], frame.shape[0], QImage.Format_RGB888)
211     pixmap = QPixmap.fromImage(image)
212     self.setPixmap(pixmap.scaled(self.width(), self.height(), Qt.KeepAspectRatio))
213
214
215 ▶ if __name__ == "__main__":
216     app = QApplication([])
217     dialog = DashboardDialog()
218     dialog.show()
219     app.exec_()

```

**Figure 4.2.2.4 Dashboardui.py (4/4)**

Once the user is successfully logged into the system, the user will see the main dashboard. The main dashboard has a few different functions that will be carried out. There are three buttons that can be clicked that will redirect the user to other functions. One of the buttons is to redirect to the Database of People dialog (peopledbui.py), one is to view all camera surveillance (allcams.py), and another is to view reports (reports.py). There is also a datetime label that keeps track of the current time and logs the identity of the user that is logged in.

One of the cameras providing live video feed will also be displayed on this dialog and the face recognition will be started. The system extracts each frame from the video feed to perform face recognition. An additional Cascade Classifier model from OpenCV is added to pick up on any faces that the CNN may not detect. Each frame is converted to greyscale so that the image is

simplified. The scale factor reduces the image size at each image scale to make detection more accurate. The minimum neighbours' parameter specifies how many neighbours each candidate rectangle should have to retain it, and it is set to four neighbours. Once there are faces detected and a rectangle is framed on each detected face, the image frames will be converted back to RGB colour. It will then swap the blue and red channels within the image data to ensure it is correctly formatted in RGB. A QImage object is created, suitable for graphical applications, with the image data, width, height, and RGB888 format. This series of operations prepares the image frame for display or further image-related tasks, especially in Qt-based applications or graphical user interfaces. The image frames are then displayed in the QPixmap.

With each recognised face, the system grabs the name from Firebase and displays it in the log entry together with the date and time detected, and location it is from. Location is based on the camera.



### 4.2.3 Peopledbui.py

```
1 import os
2 from PySide2.QtCore import *
3 from PySide2.QtGui import *
4 from PySide2.QtWidgets import *
5 import sys
6 #from personInfo import PersonInfoDialog
7 from personInfoTEST import PersonInfoDialog
8 #from detectLog import fetch_detectLog
9 #from personInfo import Ui_Dialog as Ui_PersonInfo
10 import pformui
11 from pformadd2_ui import Ui_Dialog
12 from pformadd2 import PFormAddDialog
13 from firebase_admin import db, credentials
14 from GUI.init_firebase import firebase_admin
15 app = firebase_admin.get_app()
16
17 class StatusDelegate(QStyledItemDelegate):
18     def paint(self, painter, option, index):
19         status = index.model().data(index)
20         if status == "Caution":
21             option.palette.setColor(QPalette.Text, QColor("red"))
22         elif status == "Missing":
23             option.palette.setColor(QPalette.Text, QColor("green"))
24         else:
25             option.palette.setColor(QPalette.Text, option.palette.color(QPalette.Text))
26
27         super().paint(painter, option, index)
28
29 class CustomProxyModel(QSortFilterProxyModel):
30     def __init__(self):
31         super().__init__()
32         self.status_filter = "All"
33
34     def set_status_filter(self, status):
35         self.status_filter = status
36         self.invalidateFilter()
37
38     def filterAcceptsRow(self, source_row, source_parent):
39         if self.status_filter == "All":
40             return True
41
42         index = self.sourceModel().index(source_row, 1, source_parent)
43         status = self.sourceModel().data(index)
44         return status == self.status_filter
45
46 class Ui_PeopleDBDialog(object):
47     def setupUi(self, PeopleDBDialog):
48         if not PeopleDBDialog.setObjectName():
49             PeopleDBDialog.setObjectName(u"PeopleDBDialog")
50             PeopleDBDialog.resize(1126, 692)
51             self.verticalLayout = QVBoxLayout(PeopleDBDialog)
52             self.verticalLayout.setObjectName(u"verticalLayout")
53             self.label = QLabel(PeopleDBDialog)
54             self.label.setObjectName(u"label")
55             self.label.setAlignment(Qt.AlignCenter)
56             self.verticalLayout.addWidget(self.label)
57
58             # Add a QLineEdit widget for searching
```

Figure 4.2.3.1 Peopledbui.py (1/6)

```

59     self.searchLineEdit = QLineEdit(PeopleDBDialog)
60     self.searchLineEdit.setObjectName(u"searchLineEdit")
61     self.searchLineEdit.setGeometry(QRect(350, 90, 311, 51))
62     self.searchLineEdit.textChanged.connect(self.search_data)
63
64     self.admin = QWidget(PeopleDBDialog)
65     self.admin.setObjectName(u"admin")
66     self.admin.setGeometry(QRect(10, 10, 221, 41))
67     self.admin_label = QLabel(self.admin)
68     self.admin_label.setObjectName(u"admin_label")
69     self.admin_label.setGeometry(QRect(0, 0, self.admin.width(), self.admin.height()))
70     self.admin_label.setAlignment(Qt.AlignCenter)
71     self.admin_label.setStyleSheet(u"font-size: 16px; font-weight: bold;")
72     self.admin_label.setText("Administrator: Admin")
73
74     self.datetime = QWidget(PeopleDBDialog)
75     self.datetime.setObjectName(u"datetime")
76     self.datetime.setGeometry(QRect(10, 50, 221, 141))
77     self.datetime.setStyleSheet(u"")
78     self.datetime.setStyleSheet(u"")
79     self.datetime_label = QLabel(self.datetime)
80     self.datetime_label.setObjectName(u"datetime_label")
81     self.datetime_label.setGeometry(QRect(0, 0, self.datetime.width(), self.datetime.height()))
82     self.datetime_label.setAlignment(Qt.AlignCenter)
83     self.datetime_label.setStyleSheet(u"font-size: 24px;")
84     self.update_datetime_label()
85
86     self.tableView = QTableView(PeopleDBDialog)
87     self.tableView.setObjectName(u"tableView")
88     self.tableView.setGeometry(QRect(10, 200, 1101, 481))
89     self.model = QStandardItemModel(0, 6)
90     self.model.setHorizontalHeaderLabels(["Name", "Status", "ID", "Nationality", "Criminal History",
91     "Last Detected", "Location"])
92     self.tableView.setModel(self.model)
93     self.tableView.setItemDelegateForColumn(1, StatusDelegate())
94     self.tableView.setEditTriggers(QAbstractItemView.NoEditTriggers)
95     self.tableView.setSelectionBehavior(QAbstractItemView.SelectRows)
96     horizontal_header = self.tableView.horizontalHeader()
97     horizontal_header.setSectionResizeMode(QHeaderView.Stretch)
98
99     # Create a QSortFilterProxyModel
100    self.proxyModel = QSortFilterProxyModel()
101    self.proxyModel.setSourceModel(self.model)
102    self.tableView.setModel(self.proxyModel) # Set the QSortFilterProxyModel as the model for QTableView
103    self.tableView.setItemDelegateForColumn(1, StatusDelegate())
104
105    self.sorter = QFrame(PeopleDBDialog)
106    self.sorter.setObjectName(u"sorter")
107    self.sorter.setGeometry(QRect(760, 80, 351, 101))
108    self.sorter setFrameShape(QFrame.StyledPanel)
109    self.sorter setFrameShadow(QFrame.Raised)
110
111    self.resetSortRadioButton = QRadioButton("No Sorting", self.sorter)
112    self.resetSortRadioButton.setObjectName(u"resetSortRadioButton")
113    self.resetSortRadioButton.setGeometry(QRect(10, 10, 101, 20))
114    self.resetSortRadioButton.clicked.connect(self.reset_sorting)
115    self.resetSortRadioButton.setChecked(True)
116    self.sortDateRadioButton = QRadioButton("Sort by Date Detected", self.sorter)

```

**Figure 4.2.3.2 Peopledbui.py (2/6)**

```

117     self.sortDateRadioButton.setObjectName(u"sortDateRadioButton")
118     self.sortDateRadioButton.setGeometry(QRect(10, 40, 191, 20))
119     self.sortDateRadioButton.clicked.connect(self.sort_data_by_date)
120     self.sortNameRadioButton = QRadioButton("Sort by Name", self.sorter)
121     self.sortNameRadioButton.setObjectName(u"sortNameRadioButton")
122     self.sortNameRadioButton.setGeometry(QRect(10, 70, 101, 20))
123     self.sortNameRadioButton.clicked.connect(self.sort_data_by_name)
124
125     self.header = QTextBrowser(PeopleDBDialog)
126     self.header.setObjectName(u"header")
127     self.header.setGeometry(QRect(345, 20, 391, 51))
128
129     self.backButton = QPushButton(PeopleDBDialog)
130     self.backButton.setObjectName(u"backButton")
131     self.backButton.clicked.connect(PeopleDBDialog.close)
132     self.verticalLayout.addWidget(self.backButton, alignment=Qt.AlignRight)
133
134     self.addnew = QPushButton(PeopleDBDialog)
135     self.addnew.setObjectName(u"addnew")
136     self.addnew.setGeometry(QRect(490, 160, 93, 28))
137
138     self.search = QPushButton(PeopleDBDialog)
139     self.search.setObjectName(u"search")
140     self.search.setGeometry(QRect(670, 100, 71, 28))
141
142     # self.proxyModel = CustomProxyModel()
143     # self.custom_filter = None
144
145     self.retranslateUi(PeopleDBDialog)
146
147     QMetaObject.connectSlotsByName(PeopleDBDialog)
148
149     # setupUi
150
151     def retranslateUi(self, PeopleDBDialog):
152         PeopleDBDialog.setWindowTitle(QCoreApplication.translate("PeopleDBDialog", u"People Database", None))
153         self.header.setHtml(QCoreApplication.translate("PeopleDBDialog", u"<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0"
154             u"//EN" "http://www.w3.org/TR/REC-html40/"
155             u"strict.dtd">\n"
156             "<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
157             "p, li { white-space: pre-wrap; }\n"
158             "</style></head><body style=\" font-family:'MS Shell Dlg 2'; font-size:7.8pt; font-weight:400; font-style:normal;\"\n"
159             "<p align=\"center\" style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0;"
160             " text-indent:0px;\"><span style=\" font-size:"
161             "18pt; font-weight:600;\">DATABASE OF PEOPLE"
162             "</span></p></body></html>", None))
163         self.backButton.setText(QCoreApplication.translate("PeopleDBDialog", u"Back to Dashboard", None))
164         self.addnew.setText(QCoreApplication.translate("PeopleDBDialog", u"Add New", None))
165         self.search.setText(QCoreApplication.translate("PeopleDBDialog", u"search", None))
166         self.label.setText(QCoreApplication.translate("PeopleDBDialog", u"People Database Screen", None))
167
168     # retranslateUi
169
170     def update_datetime_label(self):
171         current_datetime = QDateTime.currentDateTime().toString(u"hh:mm:ss \nMM/dd/yyyy")
172         self.datetime_label.setText(current_datetime)
173         QTimer.singleShot(1000, self.update_datetime_label)

```

**Figure 4.2.3.3 Peopledbui.py (3/6)**

```

175 def fetch_data(self):
176     ref = db.reference('People')
177     data = ref.get()
178     self.model.removeRows(0, self.model.rowCount())
179
180     if data:
181         for item in data:
182             if item is not None:
183                 name = item.get('name', '')
184                 status = item.get('status', '')
185                 ic = item.get('ic', '')
186                 nationality = item.get('nationality', '')
187                 criminal_history = item.get('criminal_history', '')
188                 last_detected_time = item.get('last_detected_time', '')
189                 location = item.get('location', '')
190
191                 # Create table items
192                 name_item = QStandardItem(name)
193                 status_item = QStandardItem(status)
194                 ic_item = QStandardItem(ic)
195                 nationality_item = QStandardItem(nationality)
196                 criminal_item = QStandardItem(criminal_history)
197                 lastdetected_item = QStandardItem(last_detected_time)
198                 location_item = QStandardItem(location)
199
200                 # Add the items to the model
201                 self.model.appendRow(
202                     [name_item, status_item, ic_item, nationality_item, criminal_item, lastdetected_item,
203                      location_item])
204             else:
205                 # If data is None, there is no data in the database, so just return without doing anything
206                 print("No data in the database.")
207
208 def search_data(self, text):
209     # Set the filter on the QSortFilterProxyModel
210     search_flags = Qt.CaseInsensitive | Qt.MatchContains
211     self.proxyModel.setFilterRegExp(text)
212     self.proxyModel.setFilterKeyColumn(-1) # Search all columns
213
214 def sort_data_by_date(self):
215     # Set the sorting order to ascending or descending based on the radio button selection
216     order = Qt.AscendingOrder if self.sortDateRadioButton.isChecked() else Qt.DescendingOrder
217     self.proxyModel.sort(5, order)
218
219 def sort_data_by_name(self):
220     # Set the sorting order to ascending or descending based on the radio button selection
221     order = Qt.AscendingOrder if self.sortNameRadioButton.isChecked() else Qt.DescendingOrder
222     self.proxyModel.sort(0, order)
223
224 def reset_sorting(self):
225     # Reset the sorting to its default state
226     self.proxyModel.sort(-1)
227
228 class PeopleDBDialog(QMainWindow):
229     def __init__(self):
230         super().__init__()
231         self.ui = Ui_PeopleDBDialog()
232         self.ui.setupUi(self)

```

**Figure 4.2.3.4 Peopledbui.py (4/6)**

```

234 self.proxyModel = CustomProxyModel()
235 self.proxyModel.setSourceModel(self.ui.model)
236
237 self.ui.backButton.clicked.connect(self.close)
238 self.ui.addnew.clicked.connect(self.add_new_clicked)
239
240 self.ui.tableView.setAlternatingRowColors(True)
241 self.ui.tableView.horizontalHeader().setStretchLastSection(True)
242
243 self.timer = QTimer(self)
244 self.timer.setInterval(1000)
245 self.timer.timeout.connect(self.fetch_data)
246 self.timer.start()
247
248 self.fetch_data()
249 self.ui.tableView.clicked.connect(self.row_item_clicked)
250
251 def add_new_clicked(self, data_name):
252     pformadd_dialog = PFormAddDialog()
253
254     if pformadd_dialog.exec_() == QDialog.Accepted:
255         name = pformadd_dialog.nameLineEdit.text()
256         status = pformadd_dialog.statusComboBox.currentText()
257         ic = pformadd_dialog.icLineEdit.text()
258         nationality = pformadd_dialog.nationalityLineEdit.text()
259         criminal_history = pformadd_dialog.criminalHistoryLineEdit.text()
260         #last_detected_time = pformadd_dialog.lastDetectedLineEdit.text()
261
262         date = pformadd_dialog.dateEdit.date().toString(Qt.ISODate)
263         time = pformadd_dialog.timeEdit.time().toString(Qt.ISODate)
264         location = pformadd_dialog.location.currentText()
265
266         last_detected_time = f"{date} {time}"
267
268         # Generate the data_name as count + 1
269         ref = db.reference('People')
270         people_data = ref.get()
271
272         # Handle the case where people_data is None
273         count = len(people_data) if people_data else 0
274         data_name = str(count)
275
276         data = {
277             "name": name,
278             "status": status,
279             "ic": ic,
280             "nationality": nationality,
281             "criminal_history": criminal_history,
282             "last_detected_time": last_detected_time,
283             "location": location,
284             'image': ""
285         }
286
287         # Push the new data to Firebase with the desired data_name
288         ref.child(data_name).set(data)
289
290 self.ui.fetch_data()
291

```

**Figure 4.2.3.5 Peopledbui.py (5/6)**

```

292 def row_item_clicked(self, index):
293     row = index.row()
294     pform = PersonInfoDialog()
295     pform.fetch_detectlog(index)
296     pform.table_item_clicked(index)
297
298     ...
338
339 def fetch_data(self):
340     people_ref = db.reference('People')
341     detect_log_ref = db.reference('DetectLog')
342
343     data = people_ref.get()
344
345     self.ui.model.clear()
346     self.ui.model.setHorizontalHeaderLabels(
347         ["Name", "Status", "ID", "Nationality", "Criminal History", "Last Detected", "Location"])
348
349     if data is not None:
350         detect_log_data = detect_log_ref.get()
351
352         for person_data in data:
353             if isinstance(person_data, dict):
354                 name = person_data.get('name', '')
355                 status = person_data.get('status', '')
356                 ic = person_data.get('ic', '')
357                 nationality = person_data.get('nationality', '')
358                 criminal_history = person_data.get('criminal_history', '')
359                 location = person_data.get('location', '')
360                 # Find the latest detection time for this person from DetectLog reference
361                 latest_detected_time = None
362                 for detection_event in detect_log_data:
363                     if detection_event is not None and detection_event['name'] == name:
364                         detection_time = f"{detection_event['detection_date']} {detection_event['detection_time']}"
365                         if latest_detected_time is None or detection_time > latest_detected_time:
366                             latest_detected_time = detection_time
367
368                 # Create table items
369                 name_item = QStandardItem(name)
370                 status_item = QStandardItem(status)
371                 ic_item = QStandardItem(ic)
372                 nationality_item = QStandardItem(nationality)
373                 criminal_item = QStandardItem(criminal_history)
374                 lastdetected_item = QStandardItem(latest_detected_time)
375                 location_item = QStandardItem(location)
376
377                 # Add the items to the model
378                 self.ui.model.appendRow(
379                     [name_item, status_item, ic_item, nationality_item, criminal_item, lastdetected_item,
380                      location_item])
381
382             else:
383                 print("No data in the database.")
384
385
386 if __name__ == "__main__":
387     app = QApplication([])
388     dialog = PeopleDBDialog()
389     dialog.show()
390     app.exec_()

```

**Figure 4.2.3.6 Peoplebui.py (6/6)**

The “StatusDelegate” class defines the colour schemes based on the status of the people. For example, the people with the status “Caution” will be red, “Missing” will be green” and others will be the default black.

The “CustomProxyModel” class handles the filtering and sorting of the database. The ‘set\_status\_filter’ method allow changing this filter to a specific status and it triggers a filter invalidation. The ‘filterAcceptsRow’ method defines the filtering logic. If the filter is set to “All”, it accepts all rows. Otherwise, it checks the status of the row in the source model and only accepted it if the status matches the ‘status\_filter’.

The UI\_PeopleDBDialog class sets up the UI for the dialog, and also includes function definitions for the search bar, sorter and filters. The “add\_new\_clicked” function creates a new dialog when clicked. This dialog is an empty form with the fields ‘name’, ‘status’, ‘ic’, ‘nationality’, ‘criminal\_history’, ‘last\_detected\_time’, and ‘location’. When the fields are filled in and saved, the new data is pushed into the Firebase.

The “row\_item\_clicked” function will grab the index of the row that the user clicked on and pass the index to personInfo.py where it will display the specific person’s detailed information by referencing the two tables in Firebase, namely ‘People’ and ‘DetectLog’, based on their index in the database.

## 4.2.4 PersonInfo.py

```
1 import sys
2 from PySide2.QtGui import QPixmap
3 from PySide2.QtGui import *
4 # from PyQt5.QtGui import QPixmap
5 from PySide2.QtWidgets import QMessageBox, QDialogButtonBox, QTableWidgetItem, QTableWidgetItem, QAbstractItemView, QTableView
6 from PySide2.QtWidgets import *
7 from PySide2.QtCore import Qt, QRect
8 #from PyQt5.QtGui import * #importing this will cause image display issue
9 from GUI.UI.personInfoUI import Ui_Dialog
10 #from detectLog import fetch_detectlog
11 from GUI.init_firebase import firebase_admin
12 from firebase_admin import credentials, storage
13 from firebase_admin import db
14 app = firebase_admin.get_app()
15
16
17 class PersonInfoDialog(QDialog, Ui_Dialog):
18     def __init__(self, key=None):
19         super(PersonInfoDialog, self).__init__()
20         self.ui = Ui_Dialog()
21         self.ui.setupUi(self)
22
23         self.ui.buttonBox.setStandardButtons(QDialogButtonBox.Cancel | QDialogButtonBox.Ok)
24         self.ui.buttonBox.accepted.connect(self.save_changes)
25         self.ui.buttonBox.rejected.connect(self.reject)
26         self.ui.detectedLog.setSelectionBehavior(QAbstractItemView.SelectRows)
27
28         #create detlog table here instead of UI file
29         self.detectedLog = QTableView()
30         self.detectedLog.setObjectName(u"detectedLog")
31         self.detectedLog.setGeometry(QRect(10, 240, 761, 251))
32         self.detLog_model = QStandardItemModel(0, 5) # Assuming 5 columns
33         header_labels = ["Name", "Status", "Detected Date", "Detected Time", "Location"]
34         self.detLog_model.setHorizontalHeaderLabels(header_labels)
35         self.ui.detectedLog.setModel(self.detLog_model)
36         horizontal_header = self.ui.detectedLog.horizontalHeader()
37         horizontal_header.setSectionResizeMode(QHeaderView.Stretch)
38
39         self.image = QLabel(self.ui.frame)
40         self.key = key
41
42         #self.fetch_detectlog()
43
44     def set_clicked_item_name(self, item_name, clicked_index):
45         print(clicked_index)
46         print(type(clicked_index))
47         new_index = int(clicked_index)
48         print("QWERTYYTRE", item_name)
49         f_x = item_name.split('\n')[1]
50         ppl_index = f_x[6:]
51         print(ppl_index)
52         #pform = PersonInfoDialog()
53         self.fetch_detectlog2(ppl_index)
54         self.table_item_clicked2(new_index)
55
56     def fetch_detectlog(self, index):
57         print("detectlog run")
58         detect_log_ref = db.reference('DetectLog')
```

Figure 4.2.4.1 PersonInfo.py (1/6)



```

59 detect_log_data = detect_log_ref.get()
60 print("DetectLog", type(detect_log_data))
61 print(detect_log_data)
62
63 ...
64
65
66
67 if detect_log_data is not None:
68     for item in reversed(detect_log_data): # Iterate in reverse to display latest records first
69         if isinstance(item,dict):
70             name = item.get('name','')
71             status = item.get('status','')
72             detection_date = item.get('detection_date','')
73             detection_time = item.get('detection_time','')
74             location = item.get('location', '')
75
76             name_item = QStandardItem(name)
77             status_item = QStandardItem(status)
78             detection_date_item = QStandardItem(detection_date)
79             detection_time_item = QStandardItem(detection_time)
80             location_item = QStandardItem(location)
81
82             xy = [name_item, status_item, detection_date_item, detection_time_item, location_item]
83
84             self.detlog_model.appendRow(xy)
85
86             # only display matched names
87             if name != index.data():
88                 for row in range(self.detlog_model.rowCount()):
89                     if self.detlog_model.item(row, 0).text() == name:
90                         self.ui.detectedLog.setRowHidden(row, True)
91
92             # make table uneditable
93             for column in range(self.detlog_model.columnCount()):
94                 self.detlog_model.item(self.detlog_model.rowCount() - 1, column).setFlags(
95                     Qt.ItemIsSelectable | Qt.ItemIsEnabled)
96
97 def table_item_clicked(self, index):
98     print("ok")
99     # Get the clicked row and column indices
100    row = index.row()
101    new_row = row
102    new_row = str(new_row)
103    print(new_row)
104
105    #name = self.ui.tableView.model().index(row, 0).data()
106    #print(name)
107    model = self.ui.detectedLog.model()
108    name_index = model.index(row, 0)
109    #selected_person_name = self.ui.tableView.model(new_row, 0).text()
110    selected_person_name = name_index.data()
111    #print(selected_person_name)
112    key = self.ui.detectedLog.model().index(row, 2).data()
113    #print(key)
114
115    ref = db.reference('People')
116    data = ref.child(new_row).get()
117

```

**Figure 4.2.4.2 PersonInfo.py (2/6)**

```

125     if data:
126         print(data)
127         self.set_key(new_row)
128
129         # Set the data in the form
130         self.ui.nameLineEdit.setText(data.get('name', ''))
131         self.ui.statusComboBox.setCurrentText(data.get('status', ''))
132         self.ui.icLineEdit.setText(data.get('ic', ''))
133         self.ui.nationalityLineEdit.setText(data.get('nationality', ''))
134         self.ui.criminalHistoryLineEdit.setText(data.get('criminal_history', ''))
135         self.ui.lastDetectedLineEdit.setText(data.get('last_detected_time', ''))
136         self.ui.location.setCurrentText(data.get('location', ''))
137
138         img = data.get('image', '')
139         print(img)
140
141         #pInfo.display_img(pInfo,firebase_image_url=img)
142
143     try:
144         # Get the bucket name and blob name from the gs:// URL
145         bucket_name, blob_name = img.replace('gs://', '').split('/', 1)
146
147         if not firebase_admin._apps:
148             cred = credentials.Certificate('path/to/serviceAccountKey.json')
149             firebase_admin.initialize_app(cred, {
150                 'storageBucket': "security-surveillance-b3ea5.appspot.com",
151                 'databaseURL':
152                     'https://security-surveillance-b3ea5-default-rtdb.asia-southeast1.firebaseio.com/'
153             })
154
155         scaled_pixmap = pixmap.scaled(QRect(0, 0, 200, 200).size(), Qt.KeepAspectRatio, Qt.SmoothTransformation)
156         self.image.setPixmap(scaled_pixmap)
157         self.image.setGeometry(0, 0, 200, 200)
158
159         result = self.exec_()
160
161         if result == QDialog.Accepted:
162             self.save_changes()
163             print("Data updated successfully.")
164         else:
165             print("data not saved")
166
167     except Exception as e:
168         print(f"Error: {e}")
169         print("not saved due to exception")
170
171     else:
172         print("Can't fetch data")
173
174 def fetch_detectlog2(self, name):
175     print("detectlog run")
176     detect_log_ref = db.reference('DetectLog')
177     detect_log_data = detect_log_ref.get()
178     print("DetectLog", type(detect_log_data))
179     print(detect_log_data)
180
181     self.detlog_model.clear()
182
183
184
185
186
187
188
189
190
191
192

```

**Figure 4.2.4.3 PersonInfo.py (3/6)**

```

193 header_labels = ["Name", "Status", "Detected Date", "Detected Time", "Location"]
194 self.detLog_model.setHorizontalHeaderLabels(header_labels)
195
196 if detect_log_data is not None:
197     for item in reversed(detect_log_data):
198         if isinstance(item, dict):
199             det_name = item.get('name', '')
200             print("print det name !!!", det_name)
201             status = item.get('status', '')
202             detection_date = item.get('detection_date', '')
203             detection_time = item.get('detection_time', '')
204             location = item.get('location', '')
205
206             name_item = QStandardItem(det_name)
207             status_item = QStandardItem(status)
208             detection_date_item = QStandardItem(detection_date)
209             detection_time_item = QStandardItem(detection_time)
210             location_item = QStandardItem(location)
211
212             xy = [name_item, status_item, detection_date_item, detection_time_item, location_item]
213
214             self.detLog_model.appendRow(xy)
215
216             # Only display matched names
217             if det_name != name:
218                 for row in range(self.detLog_model.rowCount()):
219                     if self.detLog_model.item(row, 0).text() == det_name:
220                         self.ui.detectedLog.setRowHidden(row, True)
221
222             # Make table uneditable
223             for column in range(self.detLog_model.columnCount()):
224                 self.detLog_model.item(self.detLog_model.rowCount() - 1, column).setFlags(
225                     Qt.ItemIsSelectable | Qt.ItemIsEnabled)
226
227 def table_item_clicked2(self, index):
228     print("ok")
229     # Get the clicked row and column indices
230     row = index
231     new_row = row
232     new_row = str(new_row)
233     print(new_row)
234
235     #name = self.ui.tableView.model().index(row, 0).data()
236     #print(name)
237     model = self.ui.detectedLog.model()
238     name_index = model.index(row, 0)
239     #selected_person_name = self.ui.tableView.model(new_row, 0).text()
240     selected_person_name = name_index.data()
241     #print(selected_person_name)
242     key = self.ui.detectedLog.model().index(row, 2).data()
243     #print(key)
244
245     ref = db.reference('People')
246     data = ref.child(new_row).get()
247
248     ...
254

```

Figure 4.2.4.4 PersonInfo.py (4/6)

```

255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311

```

```

if data:
    print(data)
    self.set_key(new_row)

    # Set the data in the form
    self.ui.nameLineEdit.setText(data.get('name', ''))
    self.ui.statusComboBox.setCurrentText(data.get('status', ''))
    self.ui.icLineEdit.setText(data.get('ic', ''))
    self.ui.nationalityLineEdit.setText(data.get('nationality', ''))
    self.ui.criminalHistoryLineEdit.setText(data.get('criminal_history', ''))
    self.ui.lastDetectedLineEdit.setText(data.get('last_detected_time', ''))
    self.ui.location.setCurrentText(data.get('location', ''))

    img = data.get('image', '')
    print(img)

    #pInfo.display_img(pInfo,firebase_image_url=img)

try:
    bucket_name, blob_name = img.replace('gs://', '').split('/', 1)

    if not firebase_admin._apps:
        cred = credentials.Certificate('path/to/serviceAccountKey.json')
        firebase_admin.initialize_app(cred, {
            'storageBucket': "security-surveillance-b3ea5.appspot.com",
            'databaseURL':
                'https://security-surveillance-b3ea5-default-rtdb.asia-southeast1.firebaseio.app/'
        })

    bucket = firebase_admin.storage.bucket()
    blob = bucket.blob(blob_name)
    print(blob)

    image_bytes = blob.download_as_bytes()

    pixmap = QPixmap()
    pixmap.loadFromData(image_bytes)

    scaled_pixmap = pixmap.scaled(QRect(0, 0, 200, 200).size(), Qt.KeepAspectRatio, Qt.SmoothTransformation)
    self.image.setPixmap(scaled_pixmap)
    self.image.setGeometry(0, 0, 200, 200)

    result = self.exec_()

    if result == QDialog.Accepted:
        self.save_changes()
        print("Data updated successfully.")
    else:
        print("data not saved")

except Exception as e:
    print(f"Error: {e}")
    print("not saved due to exception")

else:
    print("Can't fetch data")

```

**Figure 4.2.4.5 PersonInfo.py (5/6)**

```

312 def save_changes(self):
313     # Get the data from the form fields
314     name = self.ui.nameLineEdit.text()
315     status = self.ui.statusComboBox.currentText()
316     ic = self.ui.iCLineEdit.text()
317     nationality = self.ui.nationalityLineEdit.text()
318     criminal_history = self.ui.criminalHistoryLineEdit.text()
319     last_detected_time = self.ui.lastDetectedLineEdit.text()
320     location = self.ui.location.currentText()
321     img = self.ui.frame.frameRect()
322
323     # Prepare the data to be updated in Firebase
324     ref = db.reference('People')
325     data = {
326         'name': name,
327         'status': status,
328         'ic': ic,
329         'nationality': nationality,
330         'criminal_history': criminal_history,
331         'last_detected_time': last_detected_time,
332         'location': location
333     }
334     key = getattr(self, 'key', None) # Get the key from the attribute 'key'
335     if key:
336         ref.child(key).update(data)
337         QMessageBox.information(self.ui.nameLineEdit.window(), 'Success', 'Data updated successfully.')
338
339     # Close the dialog after saving changes
340     self.accept()
341
342     ...
343
344
345
346
347
348 def set_key(self, key):
349     self.key = key
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383

```

**Figure 4.2.4.6 PersonInfo.py (6/6)**

With the index passed from peopledbui.py, the relevant data is fetched from Firebase and displayed here. The person’s personal information like name, IC, status, and criminal history is shown in the format of a form, and their image blob is extracted from Firebase’s storage bucket to be displayed. The index will also match and retrieve all instances of the person being detected in the “DetectLog” table in Firebase and display them in the ‘detect\_log\_data’ table. The person’s last detected time and location will be displayed first from latest to oldest records. If any changes are made to the form, the key attribute will be used to update the data in Firebase.

## 4.2.5 Allcams.py

```
1  # -*- coding: utf-8 -*-
2  from datetime import datetime
3
4  #####
5  ## Form generated from reading UI file 'allcams.ui'
6  ##
7  ## Created by: Qt User Interface Compiler version 5.15.2
8  ##
9  ## WARNING! All changes made in this file will be lost when recompiling UI file!
10 #####
11
12 from PySide2.QtCore import *
13 from PySide2.QtGui import *
14 from PySide2.QtWidgets import *
15 import cv2
16 import numpy as np
17 import re
18 from facerecogniser import FaceRecognition
19 from PySide2.QtWidgets import QDialog
20 from detectLog import append_to_detect_log
21 #from personInfoTEST import PersonInfoDialog
22 from personInfo import PersonInfoDialog
23 from emergency import EmergencyDialog
24 from GUI.init_firebase import firebase_admin, db
25 app = firebase_admin.get_app()
26 sfr = FaceRecognition()
27 sfr.load_encoding_images("images/")
28 detected = []
29
30 class VideoWorker(QObject):
31     frame_ready = Signal(np.ndarray)
32
33     def __init__(self):
34         super().__init__()
35         self.video_source = None
36
37     def set_video_source(self, video_source):
38         self.video_source = video_source
39
40     @Slot()
41     def work(self):
42         if self.video_source is None:
43             return
44
45         cap = cv2.VideoCapture(self.video_source, apiPreference=cv2.CAP_MSMF)
46         if not cap.isOpened():
47             print("Failed to open video source:", self.video_source)
48             return
49
50         while True:
51             ret, frame = cap.read()
52             if not ret:
53                 break
54             self.frame_ready.emit(frame)
55
56         cap.release()
57
```

Figure 4.2.5.1 Allcams.py (1/6)

```

58 class Ui_CamsDialog(object):
59     def setupUi(self, Dialog):
60         if not Dialog.setObjectName():
61             Dialog.setObjectName(u"Dialog")
62         Dialog.resize(1124, 654)
63
64         self.cam1 = QFrame(Dialog)
65         self.cam1.setObjectName(u"cam1")
66         self.cam1.setGeometry(QRect(10, 10, 681, 311))
67         self.cam1 setFrameShape(QFrame.StyledPanel)
68         self.cam1 setFrameShadow(QFrame.Raised)
69         self.label_cam1 = QLabel(self.cam1)
70         self.label_cam1.setGeometry(QRect(0, 0, 691, 321))
71
72         self.cam2 = QFrame(Dialog)
73         self.cam2.setObjectName(u"cam2")
74         self.cam2.setGeometry(QRect(10, 320, 681, 321))
75         self.cam2 setFrameShape(QFrame.StyledPanel)
76         self.cam2 setFrameShadow(QFrame.Raised)
77         self.label_cam2 = QLabel(self.cam2)
78         self.label_cam2.setGeometry(QRect(0, 0, 691, 321))
79
80         self.backToDash = QPushButton(Dialog)
81         self.backToDash.setObjectName(u"backToDash")
82         self.backToDash.setGeometry(QRect(940, 20, 151, 51))
83         self.backToDash.clicked.connect(CamsDialog.close)
84
85         self.emergency = QPushButton(Dialog)
86         self.emergency.setObjectName(u"emergency")
87         self.emergency.setGeometry(QRect(750, 20, 151, 51))
88
89         self.verticalLayoutWidget = QWidget(Dialog)
90         self.verticalLayoutWidget.setObjectName(u"verticalLayoutWidget")
91         self.verticalLayoutWidget.setGeometry(QRect(710, 90, 391, 561))
92
93         self.detectLog = QListView(self.verticalLayoutWidget)
94         self.detectLog.setObjectName(u"detectLog")
95         self.detectLog.setContentsMargins(0, 0, 0, 0)
96         # self.detectLog.setGeometry(720, 90, 391, 561)
97
98         self.retranslateUi(Dialog)
99         QMetaObject.connectSlotsByName(Dialog)
100     # setupUi
101
102     def retranslateUi(self, Dialog):
103         Dialog.setWindowTitle(QCoreApplication.translate("Dialog", u"Dialog", None))
104         #if QT_CONFIG(tooltip)
105         self.backToDash.setToolTip(QCoreApplication.translate("Dialog", u"<html><head></body><p><span style=\" "
106         u"font-size:14pt;\>Back to Dashboard"
107         u"</span></p></body></html>", None))
108         #endif // QT_CONFIG(tooltip)
109         self.backToDash.setText(QCoreApplication.translate("Dialog", u"Back to Dashboard", None))
110         #if QT_CONFIG(tooltip)
111         self.emergency.setToolTip(QCoreApplication.translate("Dialog", u"<html><head></body><p><span style=\" "
112         u"font-size:14pt;\>Emergency Services</span>"
113         u"</p></body></html>", None))
114         #endif // QT_CONFIG(tooltip)

```

**Figure 4.2.5.2 Allcams.py (2/6)**

```

115         self.emergency.clicked.connect(self.open_emergency_dialog)
116         self.emergency.setText(QCoreApplication.translate("Dialog", u"Emergency Services", None))
117     # retranslateUi
118
119     def open_emergency_dialog(self):
120         self.emergency_dialog = EmergencyDialog()
121         self.emergency_dialog.exec_()
122
123
124     class CamsDialog(QDialog):
125     def __init__(self, detectLog, parent=None):
126         super().__init__()
127         self.ui = Ui_CamsDialog()
128         self.ui.setupUi(self)
129
130         self.ui.backToDash.clicked.connect(self.close) #FIGURE THIS OUT AGAIN
131
132         self.label_cam1 = QLabel(self.ui.cam1)
133         self.label_cam1.setGeometry(QRect(10, 10, 700, 300))
134         cam_name_label1 = QLabel("Cam1", self.label_cam1)
135         cam_name_label1.setGeometry(QRect(10, self.label_cam1.height() - 30, 100, 20))
136         cam_name_label1.setFont(QFont("Arial", 12))
137         cam_name_label1.setStyleSheet("background-color: white; color: black")
138
139         self.label_cam2 = QLabel(self.ui.cam2)
140         self.label_cam2.setGeometry(QRect(10, 10, 700, 300))
141         cam_name_label2 = QLabel("Cam2", self.label_cam2)
142         cam_name_label2.setGeometry(QRect(10, self.label_cam2.height() - 30, 100, 20))
143         cam_name_label2.setFont(QFont("Arial", 12))
144         cam_name_label2.setStyleSheet("background-color: white; color: black")
145
146         self.resize(1123, 650)
147
148         self.label_cam1.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
149         self.label_cam2.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
150         self.label_cam1.setScaledContents(True)
151         self.label_cam2.setScaledContents(True)
152
153         self.cam_locations = {
154             0: "Cam1",
155             1: "Cam2"
156         }
157         self.video_capture = [cv2.VideoCapture(0), cv2.VideoCapture(1)]
158         self.timer = QTimer()
159         self.timer.timeout.connect(self.next_frame)
160         self.timer.start(30)
161
162         self.detectlog = QListWidget(self.ui.detectLog)
163         self.detectlog.setSizePolicy(QSizePolicy.Fixed, QSizePolicy.Fixed)
164         self.detectlog.setFixedSize(391, 561)
165         self.ui.detectLog.setSizePolicy(QSizePolicy.Fixed, QSizePolicy.Fixed)
166         self.ui.detectLog.setFixedSize(391, 561)
167         self.detectlog.setAutoScroll(True)
168         self.detectlog.itemClicked.connect(self.itemClickedInList)
169

```

**Figure 4.2.5.3 Allcams.py (3/6)**



```

170 def next_frame(self):
171     face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
172
173     for camindex, capture in enumerate(self.video_capture):
174         ret, frame = capture.read()
175         if ret:
176             gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
177
178             # Detect faces
179             faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=4)
180
181             face_locations, face_names = sfr.detect_known_faces(frame)
182
183             for face_loc, name in zip(face_locations, face_names):
184                 # top=y1 left=x1 bottom=y2 right=x2
185                 y1, x1, y2, x2 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]
186
187                 cv2.putText(frame, name, (x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 0), 2)
188                 if "black" in name:
189                     color = (0, 0, 200)
190                 elif "vip" in name:
191                     color = (200, 0, 0)
192                 else:
193                     color = (0, 200, 0)
194
195                 cv2.rectangle(frame, (x1, y1), (x2, y2), color, 4)
196                 self.fetch_data(name, camindex)
197
198             # Swap the red and blue channels
199             frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
200             frame[:, :, [0, 2]] = frame[:, :, [2, 0]]
201
202             frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
203             image = QImage(frame, frame.shape[1], frame.shape[0], QImage.Format_RGB888)
204             pixmap = QPixmap.fromImage(image)
205
206             if camindex == 0:
207                 self.label_cam1.setPixmap(pixmap.scaled(self.label_cam1.width(), self.label_cam1.height(),
208                                                         Qt.KeepAspectRatio))
209             elif camindex == 1:
210                 self.label_cam2.setPixmap(pixmap.scaled(self.label_cam2.width(), self.label_cam2.height(),
211                                                         Qt.KeepAspectRatio))
212
213         else:
214             print(f"Error reading frame from camera {camindex}")
215
216
217 def capture_frame(self, video_source):
218     cap = cv2.VideoCapture(video_source, apiPreference=cv2.CAP_MSMF)
219     if not cap.isOpened():
220         print("Failed to open video source:", video_source)
221         return False, None
222
223     ret, frame = cap.read()
224     cap.release()
225     return ret, frame
226

```

**Figure 4.2.5.4 Allscams.py (4/6)**

```

227 def fetch_data(self, name, camindex):
228     #print(name)
229     ref = db.reference('People')
230     data = ref.get()
231     print("Fetched data type:", type(data))
232     print("Fetched data:", data)
233
234     # Check if data is not None
235     if data is not None:
236         #print(type(data))
237         for index, item in enumerate(data):
238             det_name = item.get('name', '')
239             #print("asdfghgfd", data.index)
240
241             # print(det_name)
242             if det_name == name:
243                 if isinstance(item, dict):
244
245                     numindex = index
246
247                     name2 = item.get('name', '')
248                     status = item.get('status', '')
249                     ic = item.get('ic', '')
250                     nationality = item.get('nationality', '')
251                     criminal_history = item.get('criminal_history', '')
252                     last_detected_time = item.get('last_detected_time', '')
253
254                     current_datetime = datetime.now()
255                     detection_date = current_datetime.strftime("%m/%d/%Y")
256                     detection_time = current_datetime.strftime("%H:%M:%S")
257                     location = self.cam_locations.get(camindex, 'Unknown Camera')
258
259                     if status == "Caution":
260                         text_color = QColor(255, 0, 0) # Red
261                     elif status == "Missing":
262                         text_color = QColor(0, 150, 50) # Green
263                     else:
264                         text_color = QColor(0, 0, 0)
265
266                     item_text = f"Index: {numindex}\nName: {name2}\nStatus: {status}\nDetected: {detection_date} \
267                                 f"{detection_time}\nLocation: {location}\n"
268                     item = QListWidgetItem(item_text)
269                     item.setForeground(text_color)
270                     self.detectlog.insertItem(0, item)
271                     append_to_detect_log(name2, status, detection_date, detection_time, location)
272
273                 else:
274                     # If data is None, there is no data in the database, so just return without doing anything
275                     print("No data in the database.")
276
277 def itemClickedinList(self, item):
278     clickedItem = item.text()
279     print(clickedItem)
280     f_x = clickedItem.split('\n')[0]
281
282     ppl_index = f_x[6:]
283     print(f_x)
284     print("capture index", ppl_index)

```

**Figure 4.2.5.5 Allcams.py (5/6)**

```

285     pform = PersonInfoDialog()
286
287     match = re.search(r"\b\d+\b", clickedItem)
288
289     if match:
290         clicked_index = match.group()
291         print("click value", clicked_index)
292
293     # If the detail window doesn't exist or has been closed, create a new instance
294     if not pform or not pform.isVisible():
295         self.pform = pform
296
297     # Set the clicked item name in the detail window
298     self.pform.set_clicked_item_name(clickedItem, clicked_index)
299
300
301 if __name__ == "__main__":
302     app = QApplication([])
303     detect_log_widget = QListWidget()
304     dialog = CamsDialog(detect_log_widget)
305     dialog.show()
306     app.exec_()

```

**Figure 4.2.5.6 Allcams.py (6/6)**

The “VideoWorker” class handles video processing tasks asynchronously. The class is initialised with a video source, set with ‘set\_video\_source’ so the system can be flexible with the number of video sources it takes. The ‘work’ method opens the video source using OpenCV, captures frames, and emits the ‘frame\_ready’ signal for each frame. When a video frame is ready for further processing, the ‘frame\_ready’ signal is emitted. If the video source fails to open, it prints an error message. Video processing is facilitated in a multi-threaded or event-driven application so that frames are processed as they become available.

The same face recognition processing steps occur here as in Dashboardui.py but is replicated as many times as there are video feeds. As the faces detected in the video feeds are recognised and matched against the database, the person’s basic details like name and status, along with the current datetime and location is appended into a list and colour-coded according to their status so that the user can be alert to the statuses.

Each item appended in the list is clickable. When an item is clicked, it matches the index in Firebase and opens the personInfo.py dialog with the information of the indexed person in detail. The information is updated in real-time.

## 4.2.6 Reports.py

```
1 import csv
2 from datetime import datetime
3 import matplotlib.pyplot as plt
4 import numpy as np
5 from collections import defaultdict
6 from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as FigureCanvas
7 from PySide2.QtCore import QDate, QTime
8 from PySide2.QtGui import QPalette, QColor
9 from PySide2.QtWidgets import QApplication, QDialog, QTableWidgetItem, QStyledItemDelegate, QWidget, \
10     QGraphicsScene, QFileDialog, QMessageBox
11 from matplotlib.lines import Line2D
12 from GUI.reportsui import Ui_Dialog
13 from firebase_admin import db
14 from GUI.init_firebase import firebase_admin
15 app = firebase_admin.get_app()
16
17
18 class StatusDelegate(QStyledItemDelegate):
19     def initStyleOption(self, option, index):
20         super().initStyleOption(option, index)
21
22         status = index.model().data(index)
23         if status == "Caution":
24             option.palette.setColor(QPalette.Text, QColor("red"))
25         elif status == "Missing":
26             option.palette.setColor(QPalette.Text, QColor("green"))
27         else:
28             option.palette.setColor(QPalette.Text, option.palette.color(QPalette.Text))
29
30
31 class ReportHandler:
32     def __init__(self, from_date, to_date, from_time, to_time, location, status, list_names_checked):
33         self.from_date = from_date
34         self.to_date = to_date
35         self.from_time = from_time
36         self.to_time = to_time
37         self.location = location
38         self.status = status
39         self.list_names_checked = list_names_checked
40
41     def fetch_data_from_firebase(self):
42         detect_log_ref = db.reference('DetectLog')
43         dl_data = detect_log_ref.get()
44
45         filtered_data = []
46
47         if dl_data is not None:
48             for record in dl_data:
49                 if isinstance(record, dict):
50                     required_keys = ['detection_date', 'detection_time', 'name', 'status', 'location']
51                     if all(key in record for key in required_keys):
52                         detection_datetime = datetime.strptime(record['detection_date'] + ' ' + record['detection_time'],
53                             "%m/%d/%Y %H:%M:%S")
54                         if (self.from_date <= detection_datetime.date() <= self.to_date) and \
55                             (self.from_time <= detection_datetime.time() <= self.to_time):
56
57                             # Check if status matches, status selected, or "All" is selected
58                             if self.status == "All" or record['status'] == self.status:
```

Figure 4.2.6.1 Reports.py (1/6)

```

59
60
61         # Check if location matches, "All" is selected, or location matches
62         if self.location == "All" or record['location'] == self.location:
63             filtered_data.append(record)
64
65     print(filtered_data)
66     return filtered_data
67
68
69 def process_data(self, filtered_data):
70     processed_data = {} # To store processed data with unique detections per day and their statuses
71
72     for record in filtered_data:
73         detection_datetime = datetime.strptime(record['detection_date'] + ' ' + record['detection_time'],
74                                               "%m/%d/%Y %H:%M:%S")
75
76         day = detection_datetime.date()
77         person_name = record['name']
78         status = record['status']
79
80         if day not in processed_data:
81             processed_data[day] = []
82
83         # Check if the person with the same status has already been added for that day
84         person_found = False
85         for entry in processed_data[day]:
86             if entry['name'] == person_name and entry['status'] == status:
87                 person_found = True
88                 break
89
90         if not person_found:
91             processed_data[day].append({'name': person_name, 'status': status})
92
93     processed_counts = {} # To store the final counts of unique detections per day for each status
94     for day, person_statuses in processed_data.items():
95         status_counts = {
96             "Caution": 0,
97             "Missing": 0,
98             "Clean": 0
99         }
100         for entry in person_statuses:
101             status = entry['status']
102             status_counts[status] += 1
103
104         processed_counts[day] = status_counts
105
106     return processed_counts
107
108 def generate_graph_explanation(self, processed_counts):
109     explanation = "<span style='font-weight: bold; font-size: 16pt;'>Detection Report:</span><br><br>"
110
111     # Add status counts to the explanation
112     for day, status_counts in processed_counts.items():
113         explanation += f"<span style='font-size: 14pt;'>Date: {day.strftime('%Y-%m-%d')}</span><br>"
114         for status, count in status_counts.items():
115             if count > 0:
116                 status_color = 'red' if status == 'Caution' else 'green' if status == 'Missing' else 'blue'
117                 explanation += f"<span style='color: {status_color}; font-size: 12pt;'>{status}: {count}</span><br>"
118         explanation += "<br>"

```

**Figure 4.2.6.2 Reports.py (2/6)**

```

116 explanation += "<br><span style='font-weight: bold; font-size: 14pt;'>Location Key:</span><br>"
117 explanation += "<span style='font-size: 12pt;'>1:</span> Camera 1 location<br>"
118 explanation += "<span style='font-size: 12pt;'>2:</span> Camera 2 location"
119 return explanation
120
121 def process_data2(self, filtered_data):
122     processed_data = {} # To store processed data with unique detections per day, status, and location
123
124     for record in filtered_data:
125         detection_datetime = datetime.strptime(record['detection_date'] + ' ' + record['detection_time'],
126                                               "%m/%d/%Y %H:%M:%S")
127         day = detection_datetime.date()
128         person_name = record['name']
129         status = record['status']
130         location = record['location']
131
132         if day not in processed_data:
133             processed_data[day] = []
134
135         # Check if the person with the same status and location has already been added for that day
136         person_found = False
137         for entry in processed_data[day]:
138             if entry['name'] == person_name and entry['status'] == status and entry['location'] == location:
139                 person_found = True
140                 break
141
142         if not person_found:
143             processed_data[day].append({'name': person_name, 'status': status, 'location': location})
144
145     data = [] # To store processed data with required fields for plot_graph
146     for day, person_info_list in processed_data.items():
147         for person_info in person_info_list:
148             data.append({'day': day, 'status': person_info['status'], 'location': person_info['location']})
149
150     return data
151
152 def plot_graph(self, data):
153     days = []
154     status_counts = {
155         "Caution": {"Cam1": [], "Cam2": []},
156         "Missing": {"Cam1": [], "Cam2": []},
157         "Clear": {"Cam1": [], "Cam2": []}
158     }
159
160     for record in data:
161         day = record['day']
162         status = record['status']
163         location = record['location']
164
165         if day not in days:
166             days.append(day)
167
168         status_counts[status][location].append(day)
169
170     # Count the occurrences of each status for each day and location
171     caution_cam1_counts = [status_counts['Caution']['Cam1'].count(day) for day in days]
172     caution_cam2_counts = [status_counts['Caution']['Cam2'].count(day) for day in days]
173     missing_cam1_counts = [status_counts['Missing']['Cam1'].count(day) for day in days]

```

**Figure 4.2.6.3 Reports.py (3/6)**

```

174 missing_cam2_counts = [status_counts['Missing']['Cam2'].count(day) for day in days]
175 clear_cam1_counts = [status_counts['Clear']['Cam1'].count(day) for day in days]
176 clear_cam2_counts = [status_counts['Clear']['Cam2'].count(day) for day in days]
177
178 fig, ax = plt.subplots()
179
180 caution_color = 'red'
181 missing_color = 'green'
182 clear_color = 'blue'
183
184 bar_width = 0.2
185 x = np.arange(len(days))
186
187 ax.bar(x - bar_width, caution_cam1_counts, width=bar_width, label='Caution', color=caution_color)
188 ax.bar(x - bar_width, missing_cam1_counts, width=bar_width, label='Missing', bottom=caution_cam1_counts,
189       color=missing_color)
190 ax.bar(x - bar_width, clear_cam1_counts, width=bar_width, label='Clear',
191       bottom=[sum(x) for x in zip(caution_cam1_counts, missing_cam1_counts)], color=clear_color)
192
193 ax.bar(x, caution_cam2_counts, width=bar_width, label='Caution', color=caution_color)
194 ax.bar(x, missing_cam2_counts, width=bar_width, label='Missing', bottom=caution_cam2_counts,
195       color=missing_color)
196 ax.bar(x, clear_cam2_counts, width=bar_width, label='Clear',
197       bottom=[sum(x) for x in zip(caution_cam2_counts, missing_cam2_counts)], color=clear_color)
198
199 ax.set_ylabel('Count')
200 ax.set_title('Detection Count by Status, Location, and Day')
201 ax.legend()
202
203 ax.set_xticks(x)
204 ax.set_xticklabels(days)
205 ax.legend()
206
207 # HARD TO SEE CAM LABEL HERE
208 # # Add labels below each column to indicate Cam1 and Cam2
209 # ax.set_xticks(x - bar_width / 2, minor=True)
210 # ax.set_xticklabels(['Cam1'] * len(days), minor=True, ha='center')
211 # ax.set_xticks(x + bar_width / 2, minor=True)
212 # ax.set_xticklabels(['Cam2'] * len(days), minor=True, ha='center')
213
214 # Add labels for Cam1 and Cam2 above each pair of bars
215 for i, day in enumerate(days):
216     ax.text(x[i] - bar_width / 2, max(clear_cam1_counts[i], clear_cam2_counts[i]) + 5, '1', ha='center',
217           va='bottom', fontsize=11, fontweight='bold')
218     ax.text(x[i] + bar_width / 2, max(clear_cam1_counts[i], clear_cam2_counts[i]) + 5, '2', ha='center',
219           va='bottom', fontsize=11, fontweight='bold')
220
221 # Convert the Matplotlib figure to PySide2-compatible canvas
222 canvas = FigureCanvas(fig)
223
224 return canvas
225
226
227 class ReportsDialog(QDialog):
228     def __init__(self):
229         super().__init__()
230         self.ui = Ui_Dialog()
231         self.ui.setupUi(self)

```

**Figure 4.2.6.4 Reports.py (4/6)**

```

232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289

today = QDate.currentDate()
empty_time = QTime(0, 0) # 00:00:00
end_of_day_time = QTime(23, 59, 59) # 23:59:59

self.ui.fromDateEdit.setDate(today)
self.ui.fromDateEdit.setDisplayFormat("MM/dd/yyyy")
self.ui.toDateEdit.setDate(today)
self.ui.toDateEdit.setDisplayFormat("MM/dd/yyyy")
self.ui.fromTimeTimeEdit.setTime(empty_time)
self.ui.fromTimeTimeEdit.setDisplayFormat("hh:mm:ss")
self.ui.toTimeTimeEdit.setTime(end_of_day_time)
self.ui.toTimeTimeEdit.setDisplayFormat("hh:mm:ss")

self.ui.locationComboBox.addItem("All", "Cam1", "Cam2")
self.ui.statusComboBox.addItem("All", "Caution", "Missing", "Clear")

self.ui.backBtn.clicked.connect(self.back_to_dashboard)
self.ui.okBtn.clicked.connect(self.ok_button_clicked)
self.ui.resetBtn.clicked.connect(self.reset_button_clicked)
self.ui.printBtn.clicked.connect(self.print_button_clicked)

header_labels = ['Name', 'Status', 'Detected Date', 'Detected Time', 'Location']
self.ui.tableLog.setHorizontalHeaderLabels(header_labels)
self.status_delegate = StatusDelegate()
self.ui.tableLog.setItemDelegateForColumn(1, self.status_delegate)
self.ui.tableLog.setEditTriggers(QTableWidget.NoEditTriggers)

# self.scroll_area.setWidget(self.comment)
# self.scroll_area.setWidgetResizable(True)

def back_to_dashboard(self):
    print("Back to dashboard clicked")
    self.close()

def reset_button_clicked(self):
    today = QDate.currentDate()
    empty_time = QTime(0, 0) # 00:00:00
    end_of_day_time = QTime(23, 59, 59) # 23:59:59

    self.ui.fromDateEdit.setDate(today)
    self.ui.toDateEdit.setDate(today)
    self.ui.fromTimeTimeEdit.setTime(empty_time)
    self.ui.toTimeTimeEdit.setTime(end_of_day_time)
    self.ui.locationComboBox.setCurrentIndex(0)
    self.ui.statusComboBox.setCurrentIndex(0)

    self.ui.graph.scene().clear()
    self.ui.comment.clear()
    self.ui.tableLog.setRowCount(0)

def ok_button_clicked(self):
    from_date = self.ui.fromDateEdit.date()
    to_date = self.ui.toDateEdit.date()
    from_time = self.ui.fromTimeTimeEdit.time()
    to_time = self.ui.toTimeTimeEdit.time()
    location = self.ui.locationComboBox.currentText()
    status = self.ui.statusComboBox.currentText()

```

**Figure 4.2.6.5 Reports.py (5/6)**



```

290 list_names_checked = self.ui.listNamesCheckBox.isChecked()
291
292 # Create an instance of ReportHandler and pass the form inputs
293 self.report_handler = ReportHandler(from_date, to_date, from_time, to_time, location, status, list_names_checked)
294 filtered_data = self.report_handler.fetch_data_from_firebase()
295 processed_counts = self.report_handler.process_data(filtered_data)
296 #filtered_data.plot_graph()
297
298 self.ui.tableLog.clearContents()
299 header_labels = ['Name', 'Status', 'Detected Date', 'Detected Time', 'Location']
300 self.ui.tableLog.setHorizontalHeaderLabels(header_labels)
301 self.ui.tableLog.setRowCount(len(filtered_data))
302 self.ui.tableLog.setColumnCount(len(header_labels))
303
304 for row, record in enumerate(filtered_data):
305     self.ui.tableLog.setItem(row, 0, QTableWidgetItem(record.get('name', '')))
306     self.ui.tableLog.setItem(row, 1, QTableWidgetItem(record.get('status', '')))
307     self.ui.tableLog.setItem(row, 2, QTableWidgetItem(record.get('detection_date', '')))
308     self.ui.tableLog.setItem(row, 3, QTableWidgetItem(record.get('detection_time', '')))
309     self.ui.tableLog.setItem(row, 4, QTableWidgetItem(record.get('location', '')))
310
311 # Create an instance of ReportHandler and pass the form inputs
312 self.report_handler = ReportHandler(from_date, to_date, from_time, to_time, location, status, list_names_checked)
313 filtered_data = self.report_handler.fetch_data_from_firebase()
314
315 # Plot the graph and get the canvas
316 #graph_canvas = self.report_handler.plot_graph(processed_counts)
317 data_for_plot = self.report_handler.process_data2(filtered_data)
318 graph_canvas = self.report_handler.plot_graph(data_for_plot)
319
320 scene = QGraphicsScene()
321 scene.addWidget(graph_canvas)
322 self.ui.graph.setScene(scene)
323
324 #graph_explanation = self.report_handler.generate_graph_explanation(processed_counts)
325 #self.ui.comment_label.setText("Testing for setText")
326 #self.ui.comment_label.setText(graph_explanation)
327 #self.ui.comment_scroll_area.verticalScrollBar().setValue(0)
328
329 processed_counts = self.report_handler.process_data(filtered_data)
330 explanation = self.report_handler.generate_graph_explanation(processed_counts)
331 #graph_explanation = self.report_handler.generate_graph_explanation(processed_counts)
332 self.ui.comment.setHtml(explanation)
333
334 def print_button_clicked(self):
335     file_path, _ = QFileDialog.getSaveFileName(self, "Save CSV File", "", "CSV Files (*.csv)")
336
337     if file_path:
338         try:
339             with open(file_path, mode='w', newline='', encoding='utf-8') as file:
340                 writer = csv.writer(file)
341                 column_headers = [self.ui.tableLog.horizontalHeaderItem(col).text() for col in
342                                 range(self.ui.tableLog.columnCount())]
343                 writer.writerow(column_headers)
344
345                 for row in range(self.ui.tableLog.rowCount()):
346                     row_data = [self.ui.tableLog.item(row, col).text() for col in
347                                range(self.ui.tableLog.columnCount())]
348                     writer.writerow(row_data)
349
350                 QMessageBox.information(self, "Export Successful", "Table exported to CSV file successfully.")
351         except Exception as e:
352             QMessageBox.critical(self, "Error", f"An error occurred: {str(e)}")
353
354 if __name__ == "__main__":
355     app = QApplication([])
356     reports_dialog = ReportsDialog()
357     reports_dialog.show()
358     app.exec_()
359

```

**Figure 4.2.6.6 Reports.py (6/6)**

Reports.py mainly references from the “DetectLog” table in Firebase. The ‘ReportHandler’ class handles the background processes of retrieving and processing the data from Firebase while the ‘ReportsDialog’ class defines the actions that can be taken by the user. A form with fields like “from date”, “to date”, “from time”, “to time”, and so on has its inputs passed to the handler class for processing when the “okay” button is clicked.

All instances of the relevant data are first retrieved from the database and displayed in the table. The data is then filtered to only included unique detections so that a person is only counted once if they were already detected on that day. They are counted separately based on their status and location for ease of analysis. The unique counts are then used to plot a stacked clustered bar graph with the relevant label. The unique counts are also used to generate a basic summary of reports.

When the “print” button is clicked, the table is generated into a CSV file and saved to local storage for further analysis. The “reset” button clears all form inputs to the default and clears the table and graph.

# Chapter 5

## System Implementation

### 5.1 Hardware Setup

The hardware used for the development of this project is an Asus laptop and an android mobile phone. The computer was used in system design, development, data analysis, implementation, and testing for this project. An android mobile phone acted as an alternative camera device to substitute for a CCTV camera when testing the module for multiple cameras.

Description	Specifications
Model	Asus A510U series
Processor	Intel Core i5-8250U
Operating System	Windows 10
Graphic	NVIDIA GeForce MX130
Memory	7.88GB RAM
Storage	446GB SSD

**Table 3.1 Specifications of laptop**

Description	Specifications
Model	Huawei Mate 10 Pro (BLA-L29)
Processor	HiSilicon Kirin 970
Operating System	Android 10.0.0
Memory	6.0GB RAM
Storage	128GB

**Table 3.2 Specifications of android mobile phone**

## 5.2 Software Setup

### 5.2.1 Software, Modules and Libraries

The Integrated Development Environment (IDE) utilized during the creation of this project is PyCharm Community Edition, version 2022.3.2. The project's programming language of choice for developing the facial recognition engine and surveillance system is Python, specifically version 3.9.1. QTDesigner version 5.9.7 was also used in the development of the security system with designing the user interface. It is a graphical user interface designer for Qt applications, which is suitable for Python programming.

The interpreters and libraries below were installed to develop and run the facial recognition engine and the surveillance system:

i. **DLib** – version 19.24.2

Dlib, developed by Davis King, is a versatile C++ software library with cross-platform capabilities. It finds application in various domains, including graphical user interfaces, machine learning, image processing, numerical optimization, and more. To integrate the dlib library into a PyCharm project, the following command was employed for installation: `pip install dlib`. This command allows for the straightforward inclusion of the dlib library within the PyCharm development environment, enabling developers to harness its capabilities for their projects.

ii. **Face-Recognition** – version 1.3.0

Author Adam Geitgey utilized dlib's deep learning algorithm to develop the face-recognition library, which boasts an impressive accuracy rate of 99.38% for detecting and identifying faces. To incorporate the face-recognition library into a project, the following command was employed for installation: `pip install face-recognition`. This straightforward command enables developers to readily integrate the face-recognition library into their projects, harnessing its high-accuracy facial detection and identification capabilities.

iii. **NumPy** – version 1.25.2

NumPy, developed by Travis E. Oliphant et al., is a powerful library that provides support for multidimensional arrays and matrices, along with a wide

range of complex mathematical functions. In this project, NumPy is imported to facilitate tasks related to machine learning, mathematical computation, and data analysis. Its capabilities enable developers to efficiently work with numerical data, perform various mathematical operations, and conduct data analysis tasks within the project.

iv. **OpenCV-python** – version 4.8.0.76

Originally created by Intel, opencv-python is a library primarily employed for real-time computer vision applications. In this project, its primary role is to execute tasks related to image and video processing, with a particular emphasis on tasks such as face recognition and tracking. To integrate this library into the project, the following command was utilized for installation: `pip install opencv-python`. This command ensures the incorporation of opencv-python into the project, enabling it to handle image and video processing tasks, particularly those related to face recognition and tracking.

v. **PySide2** – version 5.15.2.1

PySide2 facilitates Python bindings for the QT cross-platform application and user interface framework. In other words, PySide2 was used to convert UI files created in QTDesigner into Python files that can be programmed and include defined functions. This was done by running `PySide2-uic main.ui -o main.py` in the terminal.

vi. **Matplotlib** – version 3.7.2

Created by John D. Hunter and Michael Droettboom, matplotlib is a Python plotting package that can aid in creating static, animated, or interactive data visualizations in Python. This library was used in the creation of analytical reports in the system.

vii. **Firebase-Admin** – version 6.2.0

As this project utilized the cloud storage created by Google's Firebase, this packaged was installed so that the program can retrieve, read, and write data into the cloud storage.

### 5.3 Settings and Configuration

To launch the system on the computer, the “Run” button on PyCharm IDE is clicked on the main.py file. As a substitute for IP CCTV cameras, a mobile app called iVCam Webcam version 7.0.8, developed by e2eSoft, was installed onto the android mobile phone from Google Play Store. For the external camera on the android phone, the device must allow the system to transfer files and enable USB debugging if it is connected through a USB cable. Otherwise, the system can also detect iVCam Webcam devices on the same Wi-Fi network and connect with IP addresses.

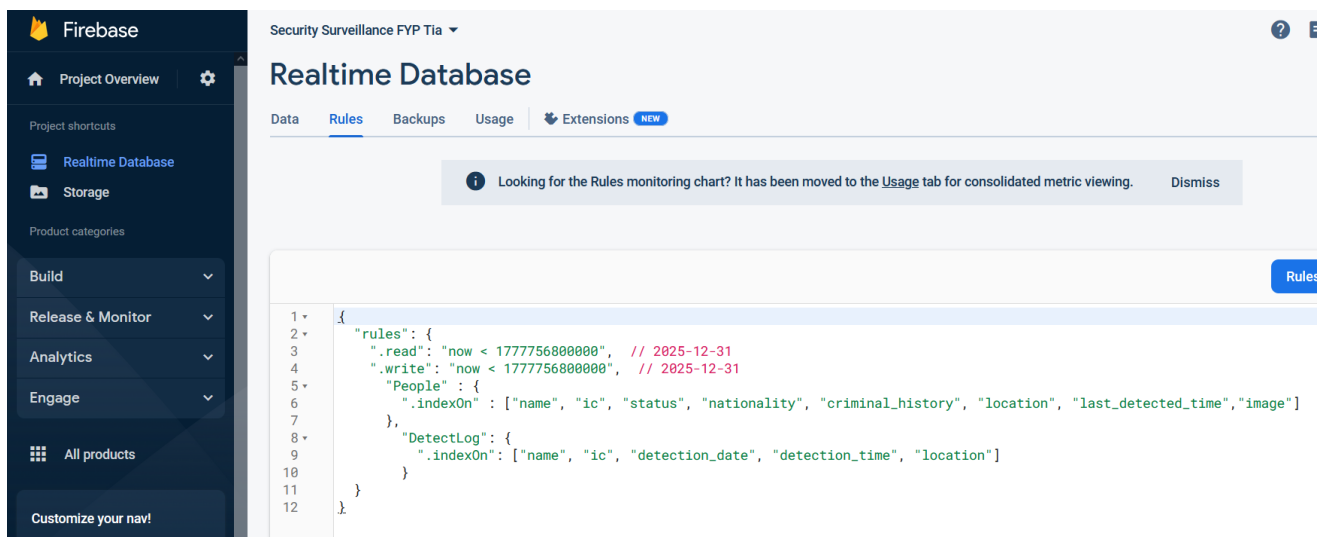
The system also uses Firebase’s Realtime Database and Firebase’s Storage Bucket to store people’s information, images, and the detected faces logs. Since this development only uses the free version of Firebase, the real time databases will be available until 31<sup>st</sup> December 2025 as configured. The reference URL to the Firebase is shown below.

#### Firestore Realtime Database:

<https://security-surveillance-b3ea5-default-rtdb.asia-southeast1.firebaseio.com/>

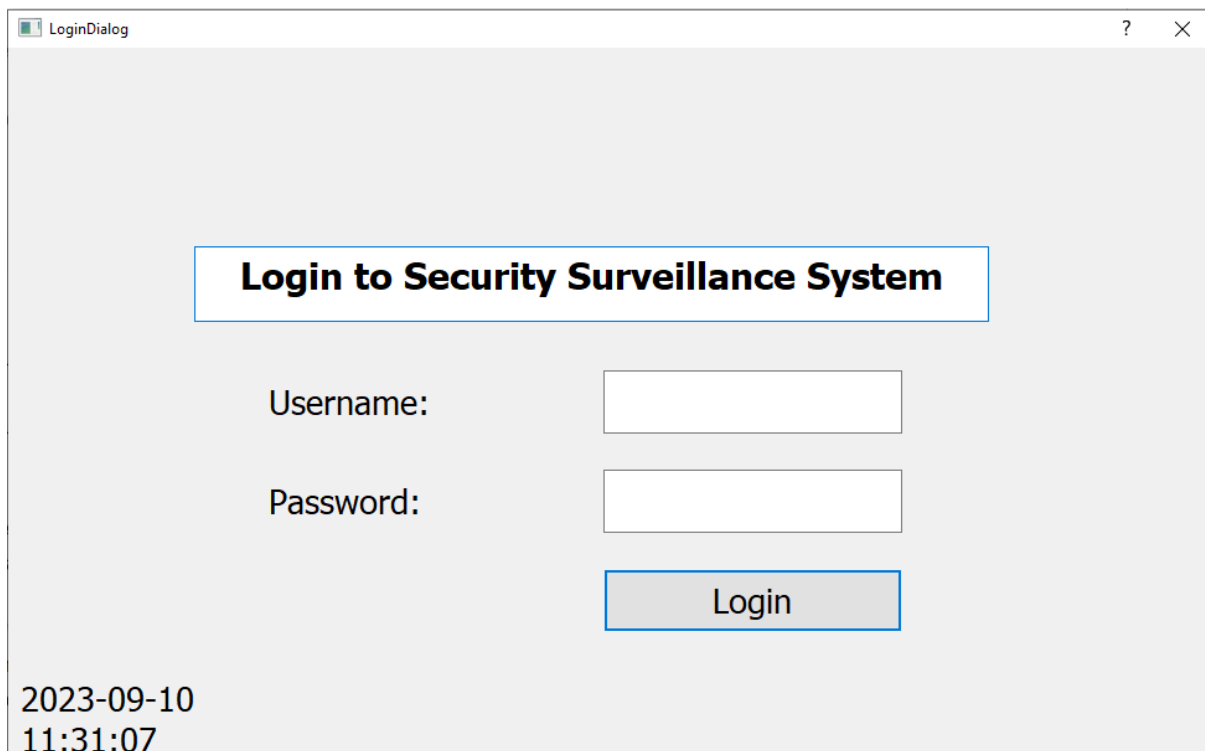
#### Firestore Storage Bucket folder path:

<gs://security-surveillance-b3ea5.appspot.com>



**Figure 5.3 Rules configuration of Firebase’s Realtime Database expiring on 31<sup>st</sup> December 2025**

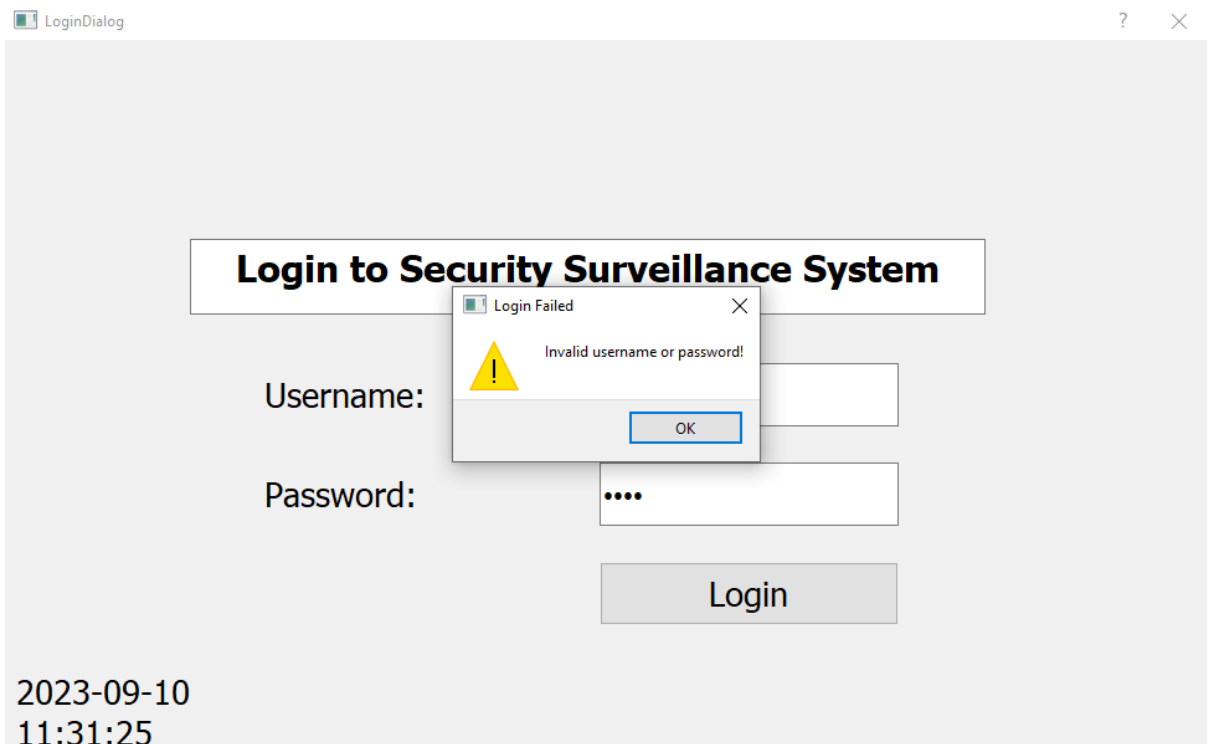
## 5.4 System Operation



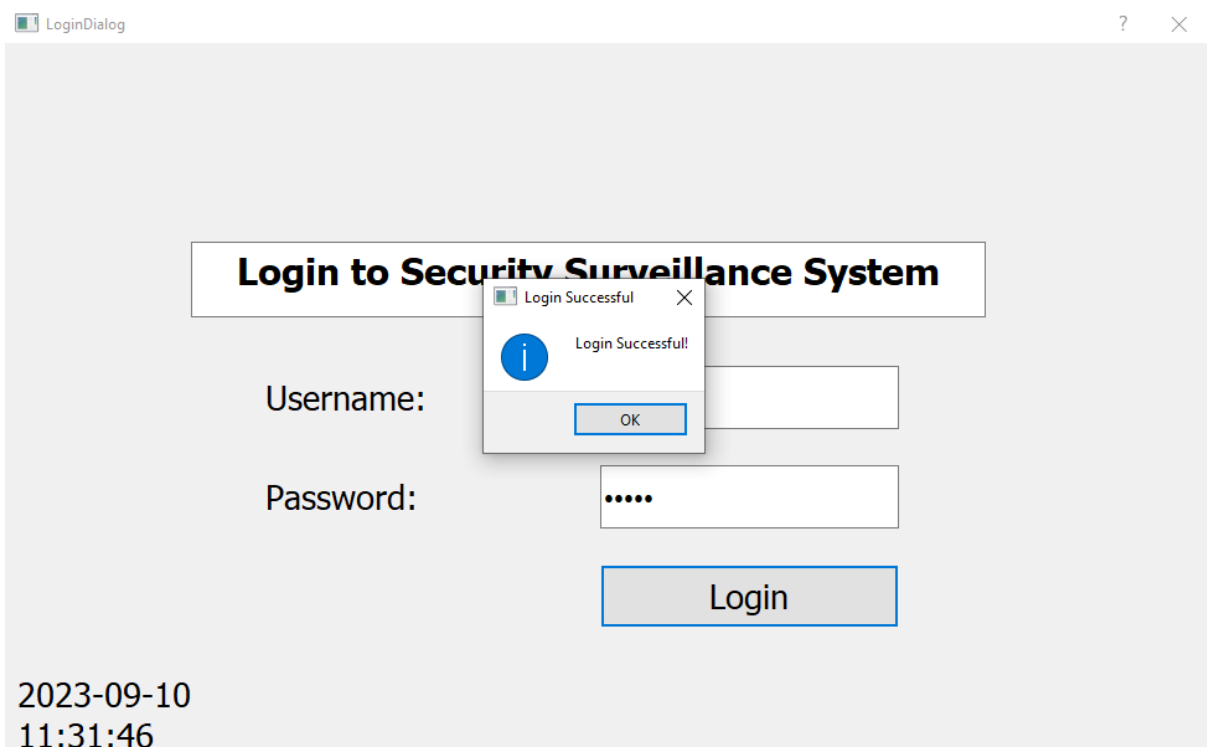
The screenshot shows a window titled "LoginDialog" with a standard Windows-style title bar (minimize, maximize, close buttons). The main content area has a light gray background. At the top center, there is a white rectangular box with a blue border containing the text "Login to Security Surveillance System" in bold black font. Below this, there are two labels: "Username:" and "Password:", each followed by a white rectangular input field with a thin gray border. Below the input fields is a gray rectangular button with a blue border and the text "Login" in black. In the bottom-left corner of the window, the date "2023-09-10" and time "11:31:07" are displayed in black text.

**Figure 5.4.1 Login Screen**

When the system is first launched, the user is prompted to log in with username and password. The current date and time are displayed on the screen, which is to keep logs of who uses the system at what time. There is no option of registering for an account because this is a system with restricted access and the main administrator will create the login credentials for the users.



**Figure 5.4.2 Login Unsuccessful**

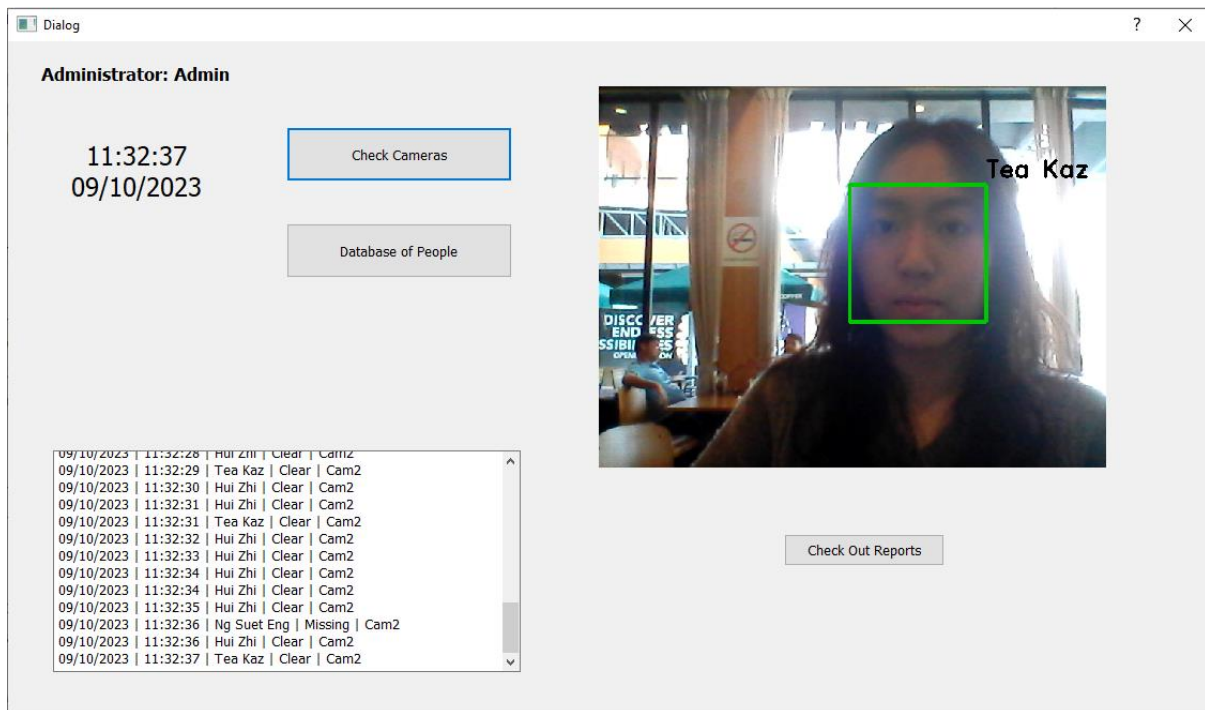


**Figure 5.4.3 Login Successful**

The figures above show the different messages that will be displayed if the login credentials are incorrect or correct. If the credentials are incorrect, a warning message will be displayed



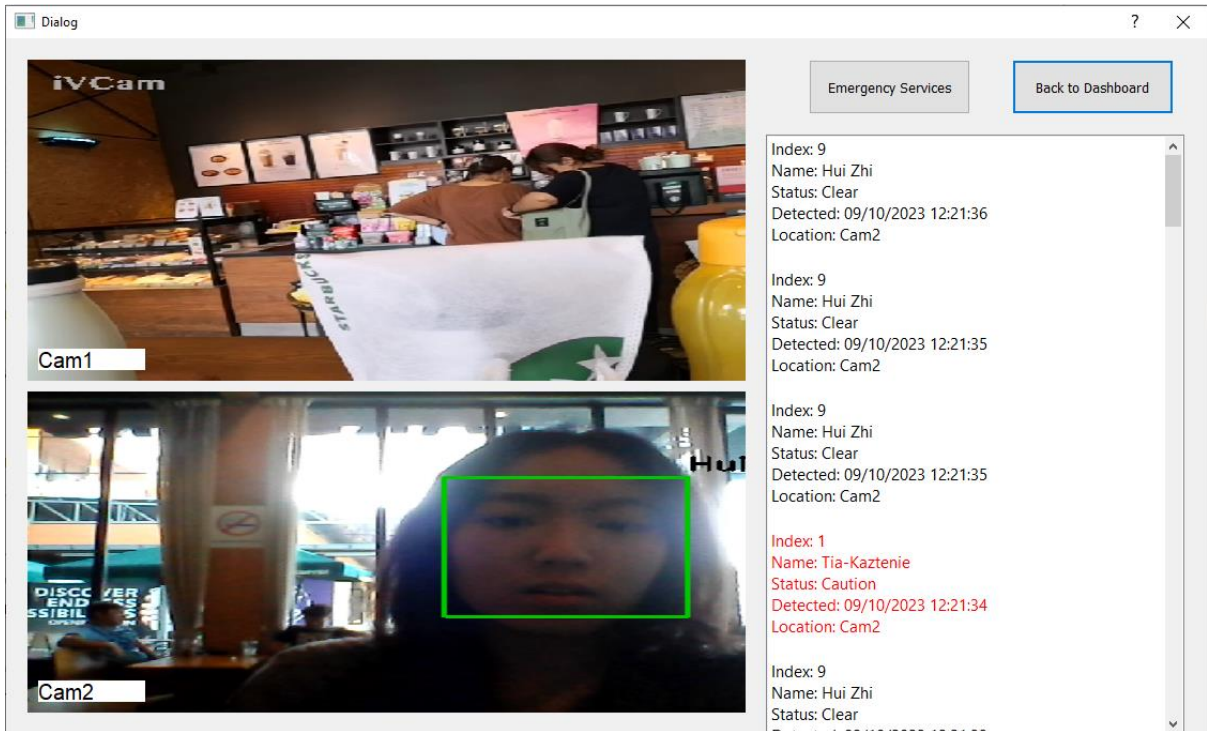
and the user will be prompted to try again. If the user is successful, the user will be logged into the main dashboard of the system.



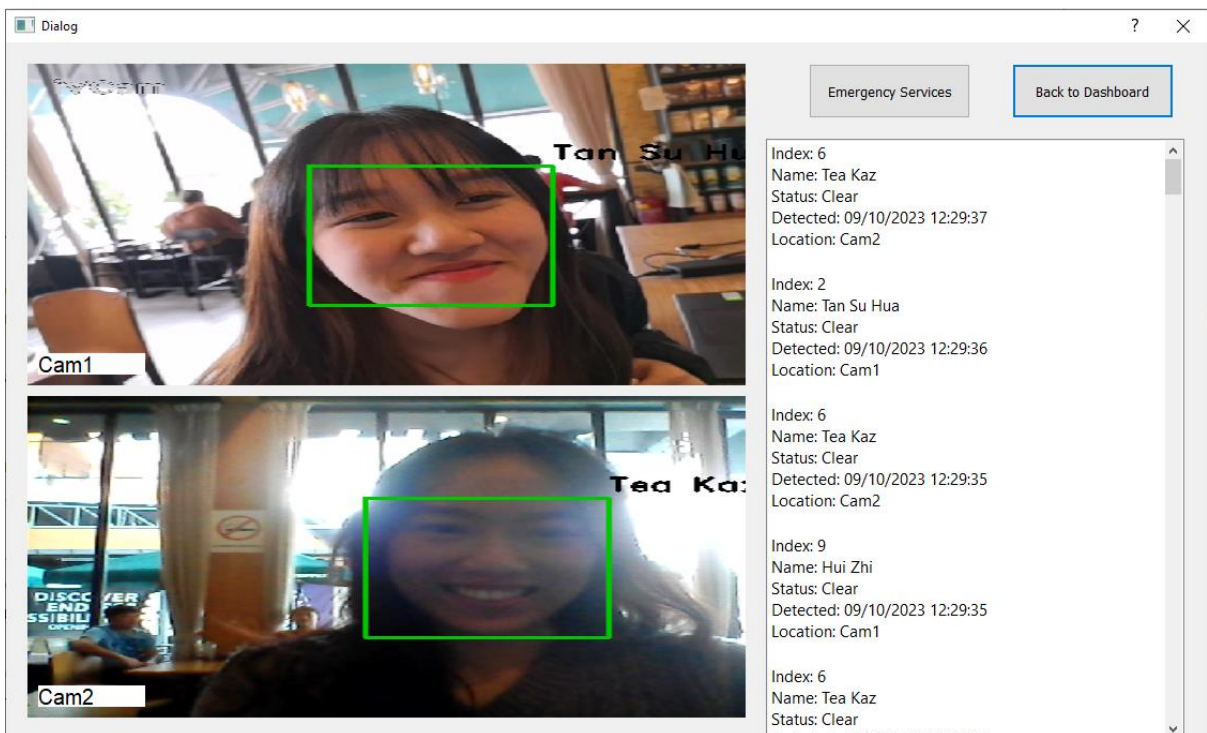
**Figure 5.4.4 Main Dashboard Screen**

The main dashboard can be divided into quarters of the screen. The top left quarter of the screen has the identity of the user logged in and the current date time. There are also two buttons, which are “Check Cameras” and “Database of People”. Another button on the bottom right corner of the screen, “Check Out Reports” leads to the report generation function.

The first camera feed is displayed on the dashboard and face recognition starts here. A green rectangle frames any face that is detected and includes the name of the person. The bottom left of the screen is a list that is constantly being updated with every new face being detected in the camera feed. The list includes the date and time that the face is detected, the location (or camera source) that it is from, the name of the person, and their status. This is for the user’s quick view so they can see if there are any persons of interest that were detected.



**Figure 5.4.5 All Cameras Surveillance with “Caution” Status Detected**



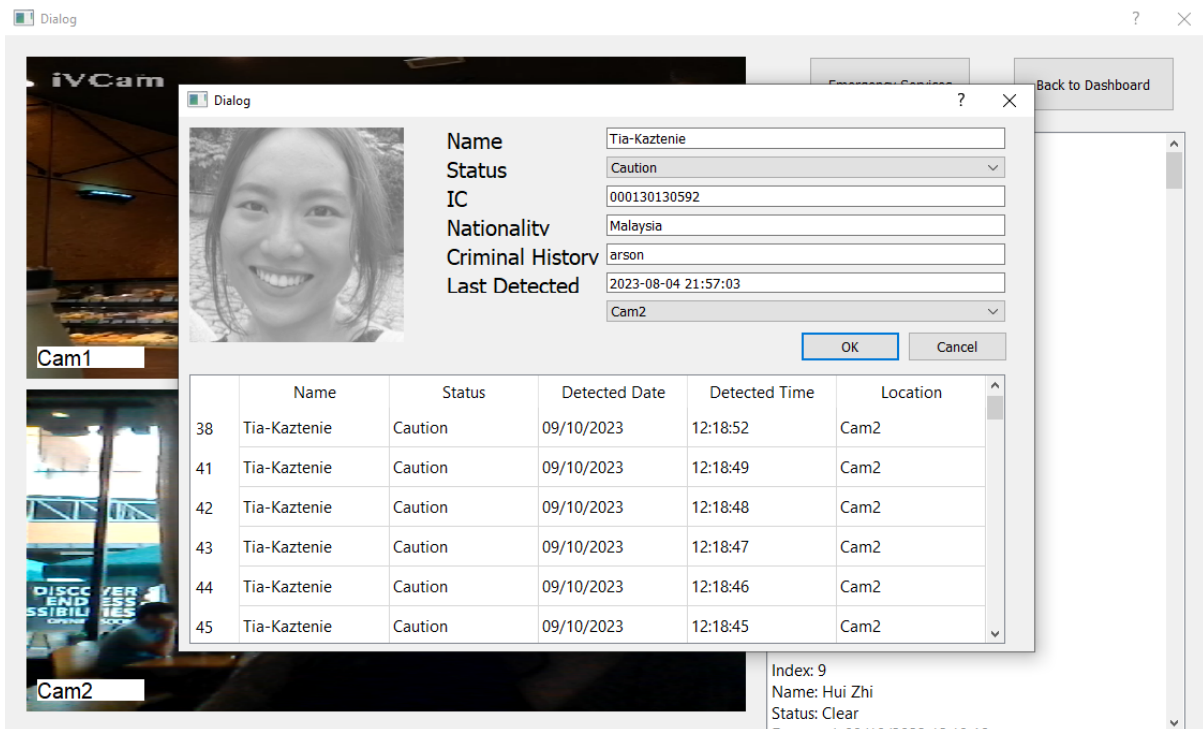
**Figure 5.4.6 All Cameras Surveillance with Faces Detected in Different Locations**

The figures above show the screens with all the surveillance cameras connected to the system. Each video feed is labelled with the camera source it is from, i.e., “Cam 1” or “Cam 2”. Every time a face is detected and recognised, it is appended to the list of people detected on the right

side of the screen with the date, time and location. If a face is detected from Cam 1, the location will be Cam 1, and vice versa for Cam 2. This would make tracing easier. For example, following the figures shown, “Tan Su Hua” is detected in Cam 1, thus her location is labelled as “Cam 1” in the appended list, and the same goes for “Tea Kaz” with Cam 2.

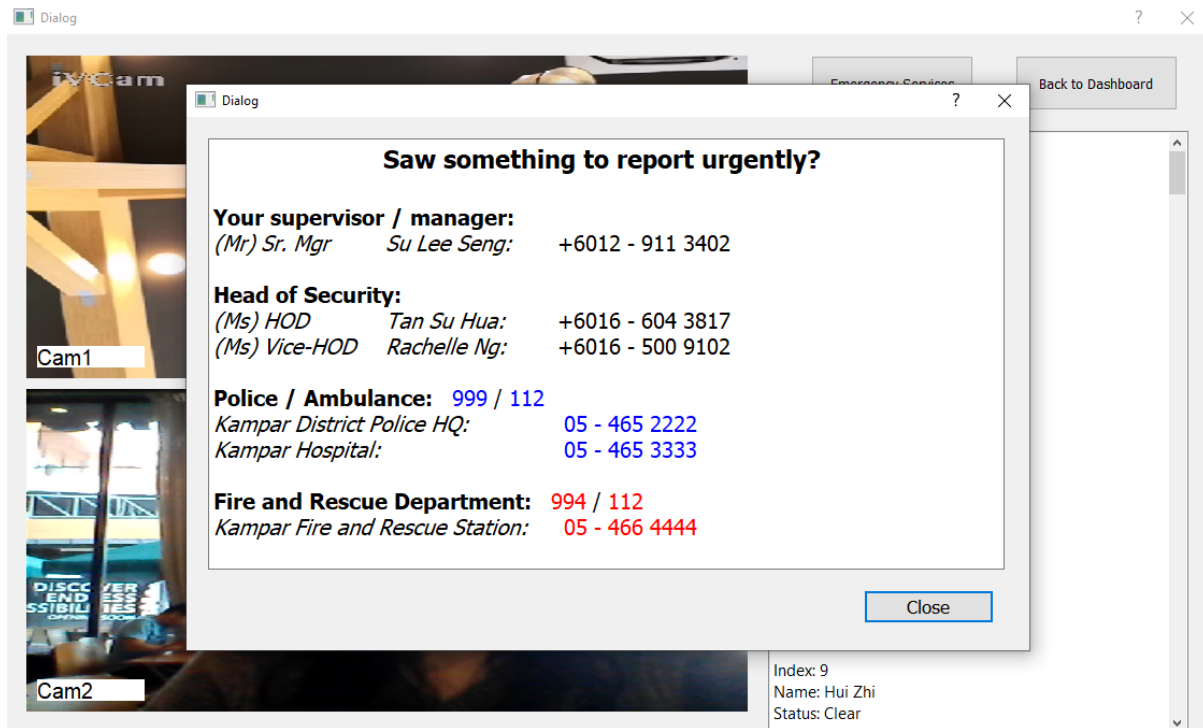
When a person with any special status like “Missing” or “Caution” is detected, their entry is labelled according to a specific colour. Red is for “Caution” for people with suspicious or dangerous criminal histories like robbery, homicide, terrorism and so on. Green is for people who are labelled as missing to help the administrator notice if any missing persons are detected.

If the user wishes to investigate more on a detected person, they can click on their entry in the appended list on the right and a dialog with the full details of the person will be displayed.



**Figure 5.4.7 View More Information on a Detected Person**

The figure above shows an example of a person’s information being displayed when clicked on from the list on the right. The user is able to look at their information like their criminal history, and the historical logs of when they were detected. This helps the user to determine what actions should be taken further, if any.



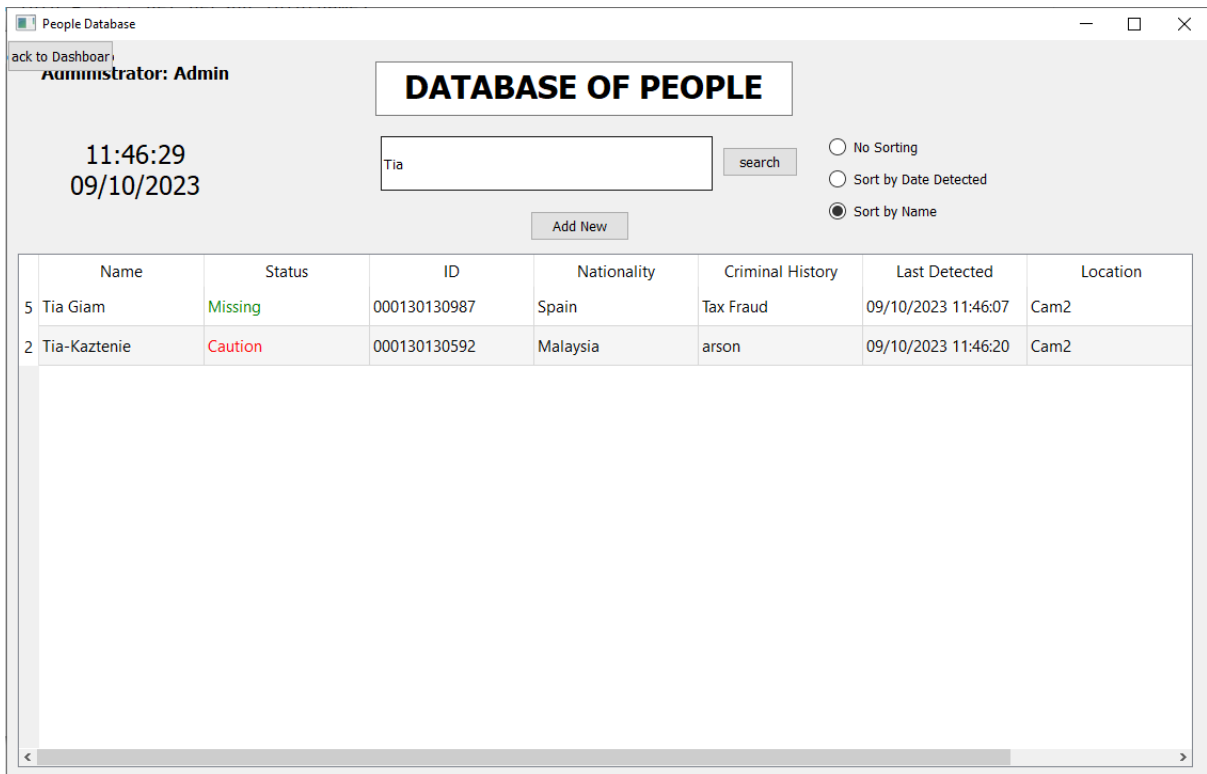
**Figure 5.4.8 Emergency Contacts**

If the user determines that any action should be taken, perhaps a medical emergency is needed or a missing person was found on camera, the user can click on the button on the top right of the screen labelled “Emergency Services”. A dialog showing all the necessary contact information will be displayed as shown in the figure above. The administrator can choose to contact their immediate supervisors if they are unsure of what actions to take, or they can immediately contact local emergency services like police, medical, and fire rescue services. These numbers will follow the nearest emergency services according to the organisation using this system.

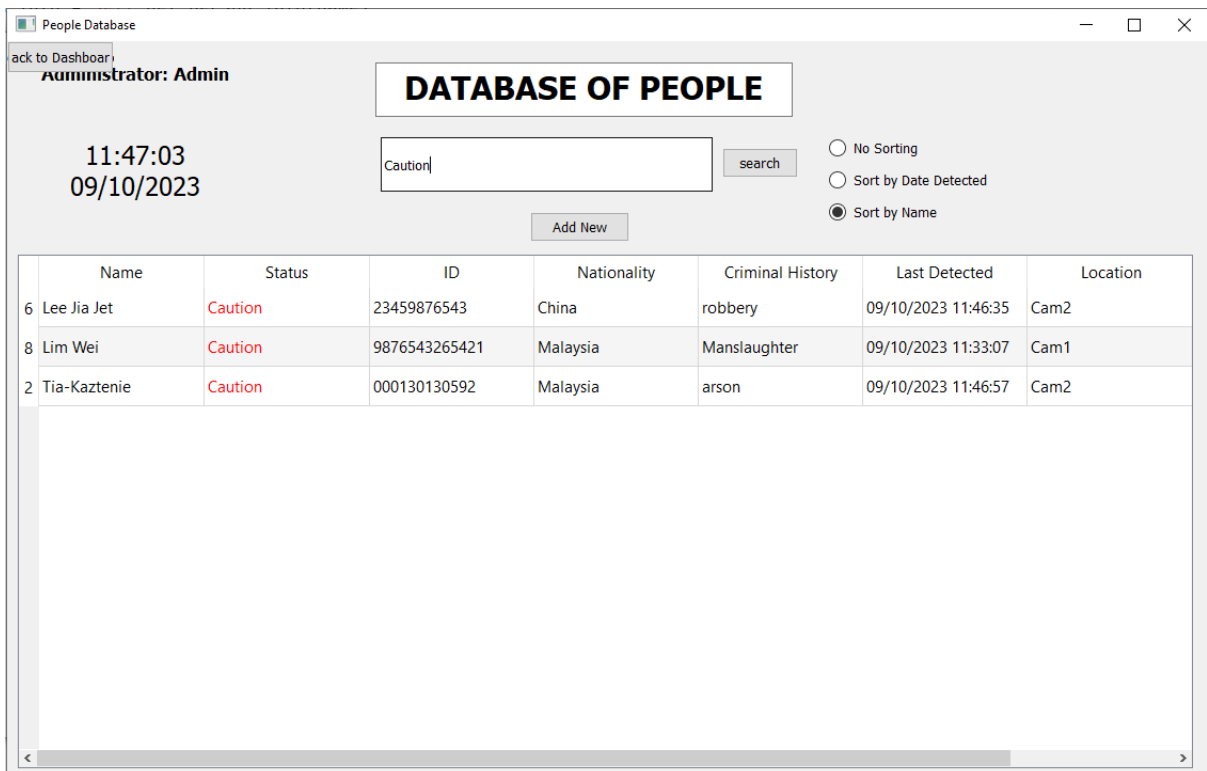
	Name	Status	ID	Nationality	Criminal History	Last Detected	Location
1	Cheong Kok Siong	Clear	990987654312	Russia	parking ticket	08/28/2023 12:31:22	Cam2
2	Tia-Kaztenie	Caution	000130130592	Malaysia	arson	09/10/2023 11:44:45	Cam2
3	Tan Su Hua	Clear	2367895421231	Singapore	speeding	08/27/2023 16:30:49	Cam1
4	Ng Suet Eng	Missing	17484987654321	Malaysia	NA	09/10/2023 11:33:58	Cam1
5	Tia Giam	Missing	000130130987	Spain	Tax Fraud	09/05/2023 17:41:39	Cam2
6	Lee Jia Jet	Caution	23459876543	China	robbery	09/10/2023 11:32:41	Cam2
7	Tea Kaz	Clear	000214135678	Spain	NA	09/10/2023 11:44:51	Cam1
8	Lim Wei	Caution	9876543265421	Malaysia	Manslaughter	09/10/2023 11:33:07	Cam1
9	Moh Ridwan	Clear	021234569876543	Indonesia	NA		
10	Hui Zhi	Clear	99098751234567	Taiwan	Jaywalking	09/10/2023 11:44:53	Cam1

**Figure 5.4.9 Database of People Screen**

From the main dashboard, the user will be directed to the screen shown in the figure above if they click on the button “Database of People”. The user can view all of the people that are recorded in Firebase database here. People who are recorded but do not have any history being detected will also be shown as such in the figure above, as seen by the person “Moh Ridwan” who has not been detected by the system.



**Figure 5.4.10 Search by Name**



**Figure 5.4.11 Search by Status**

The figures above show how the user can use the search bar to search for any specific person or category just by typing. If the user wishes to only see people with the status “Caution”,

typing it into the search bar immediately filters through the database and only shows people with the wanted status.

This can be done for any of the data fields as well. If the user wants to find people only from Malaysia, for example, the user can also type “Malaysia” in the search bar, and it will immediately filter out non-Malaysians and only display Malaysians.



People Database  
 Administrator: Admin  
 11:45:41  
 09/10/2023

**DATABASE OF PEOPLE**

search

Add New

No Sorting  
 Sort by Date Detected  
 Sort by Name

	Name	Status	ID	Nationality	Criminal History	Last Detected	Location
9	Moh Ridwan	Clear	021234569876543	Indonesia	NA		
3	Tan Su Hua	Clear	2367895421231	Singapore	speeding	08/27/2023 16:30:49	Cam1
1	Cheong Kok Siong	Clear	990987654312	Russia	parking ticket	08/28/2023 12:31:22	Cam2
5	Tia Giam	Missing	000130130987	Spain	Tax Fraud	09/05/2023 17:41:39	Cam2
8	Lim Wei	Caution	9876543265421	Malaysia	Manslaughter	09/10/2023 11:33:07	Cam1
4	Ng Suet Eng	Missing	17484987654321	Malaysia	NA	09/10/2023 11:33:58	Cam1
6	Lee Jia Jet	Caution	23459876543	China	robbery	09/10/2023 11:45:02	Cam2
10	Hui Zhi	Clear	99098751234567	Taiwan	Jaywalking	09/10/2023 11:45:33	Cam1
2	Tia-Kaztenie	Caution	000130130592	Malaysia	arson	09/10/2023 11:45:38	Cam2
7	Tea Kaz	Clear	000214135678	Spain	NA	09/10/2023 11:45:40	Cam1

**Figure 5.4.12 Sort Database by Date Detected**

People Database  
 Administrator: Admin  
 11:46:00  
 09/10/2023

**DATABASE OF PEOPLE**

search

Add New

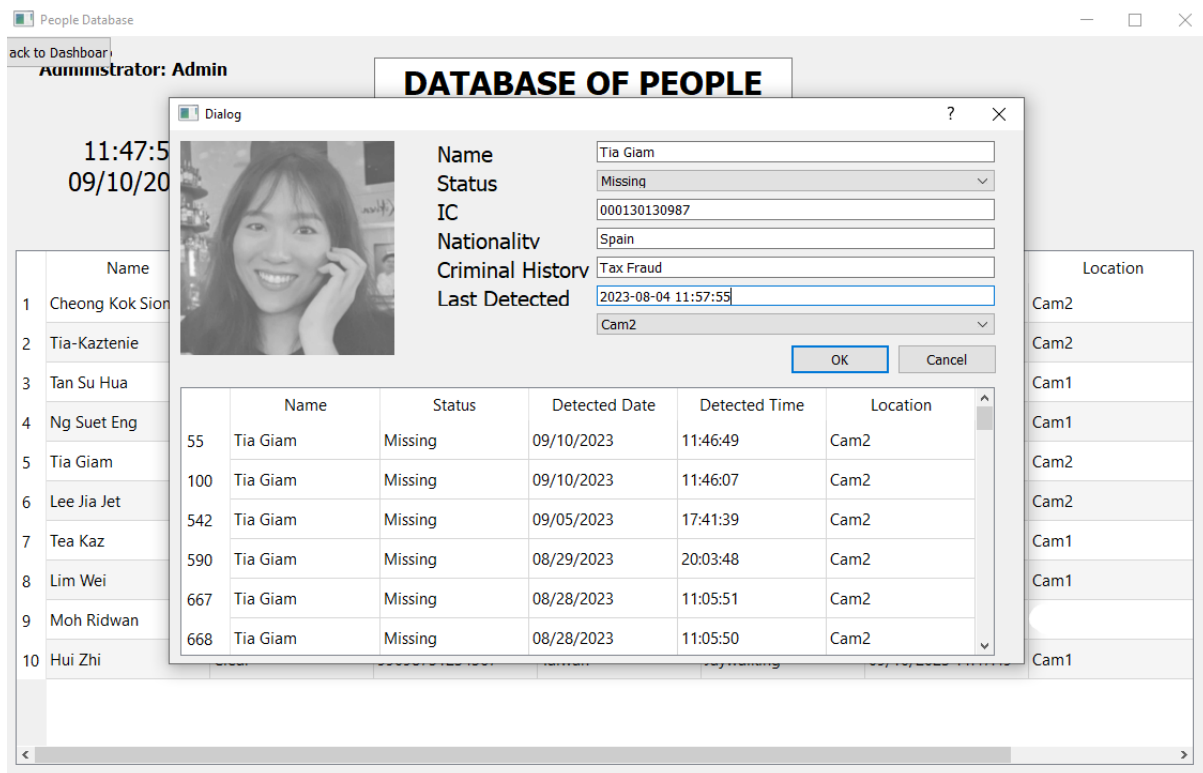
No Sorting  
 Sort by Date Detected  
 Sort by Name

	Name	Status	ID	Nationality	Criminal History	Last Detected	Location
1	Cheong Kok Siong	Clear	990987654312	Russia	parking ticket	08/28/2023 12:31:22	Cam2
10	Hui Zhi	Clear	99098751234567	Taiwan	Jaywalking	09/10/2023 11:46:00	Cam1
6	Lee Jia Jet	Caution	23459876543	China	robbery	09/10/2023 11:45:02	Cam2
8	Lim Wei	Caution	9876543265421	Malaysia	Manslaughter	09/10/2023 11:33:07	Cam1
9	Moh Ridwan	Clear	021234569876543	Indonesia	NA		
4	Ng Suet Eng	Missing	17484987654321	Malaysia	NA	09/10/2023 11:33:58	Cam1
3	Tan Su Hua	Clear	2367895421231	Singapore	speeding	08/27/2023 16:30:49	Cam1
7	Tea Kaz	Clear	000214135678	Spain	NA	09/10/2023 11:45:55	Cam1
5	Tia Giam	Missing	000130130987	Spain	Tax Fraud	09/05/2023 17:41:39	Cam2
2	Tia-Kaztenie	Caution	000130130592	Malaysia	arson	09/10/2023 11:45:56	Cam2

**Figure 5.4.13 Sort Database by Name**

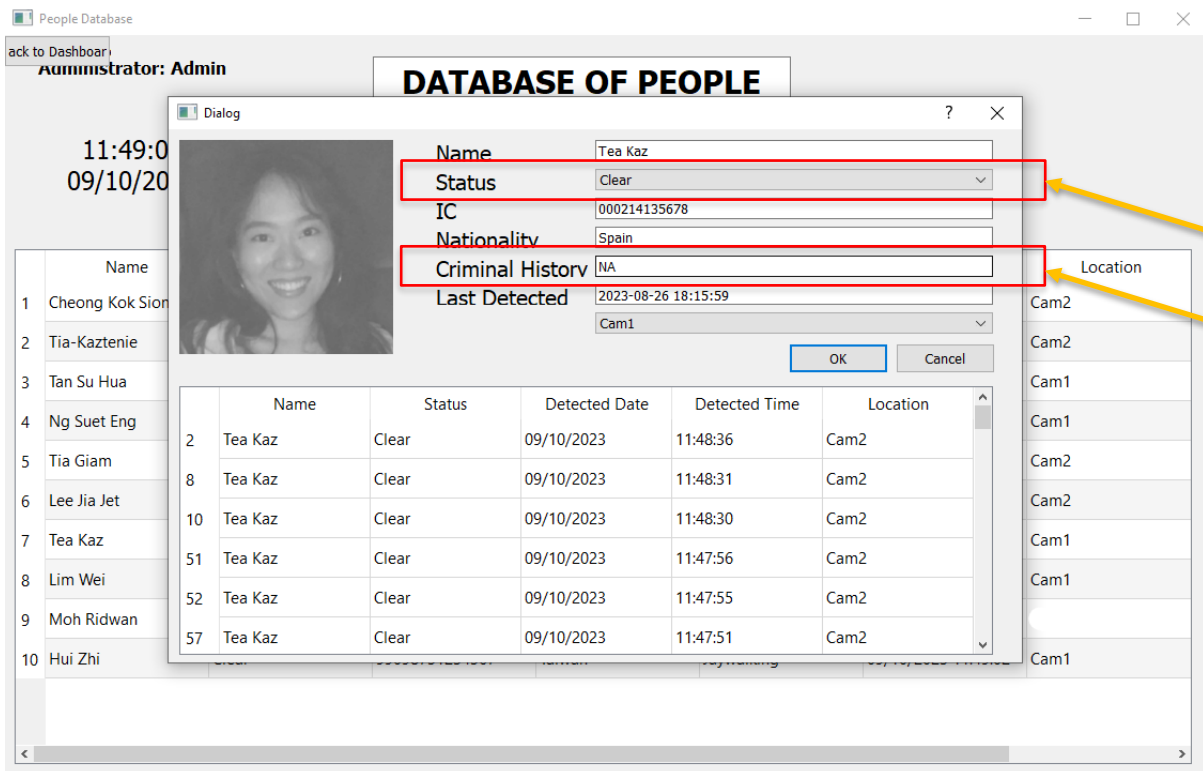
The figures above show how the user can also sort the display of the database by date detected from the latest one detected and sort names by alphabetical order.



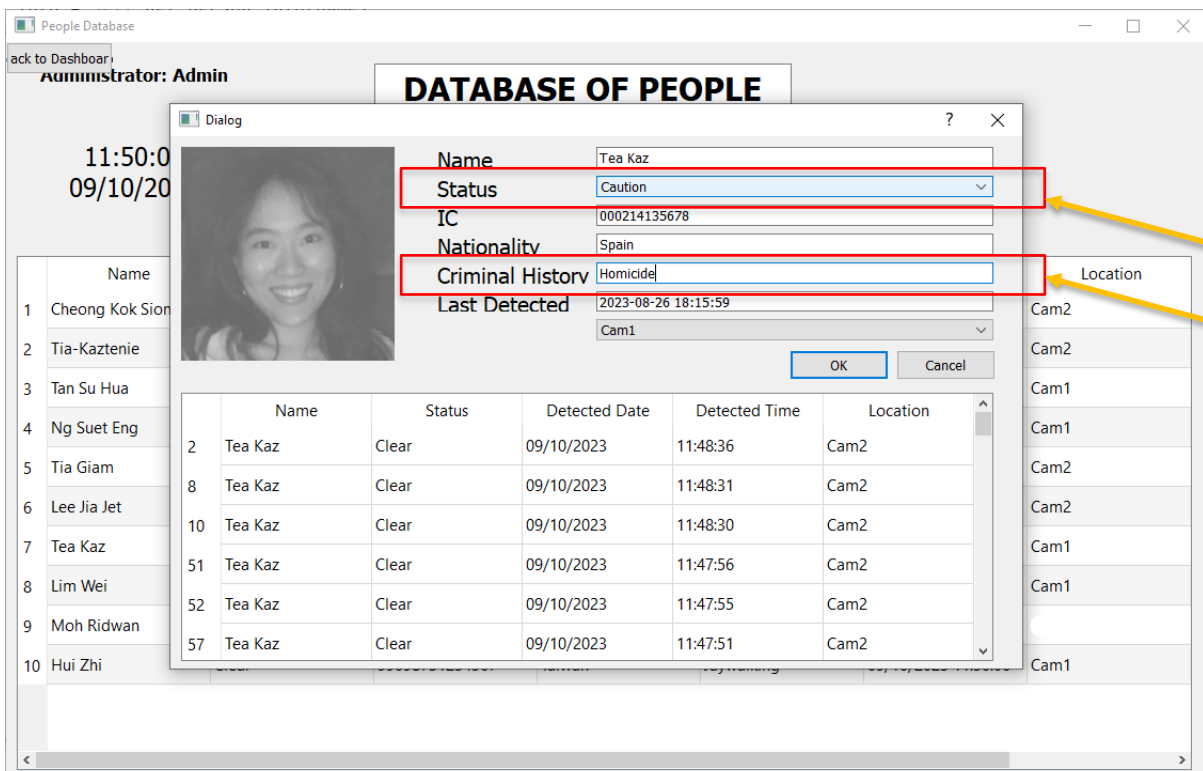


**Figure 5.4.14 Detailed Information on a Specific Person**

The figure above shows an example of when a user clicks onto any person’s row of information. A dialog with detailed information will be displayed, including all historical logs of when they were detected. When the user clicks “Cancel”, the user will exit this detailed view without making any changes to the person’s information.



**Figure 5.4.15 A person's information before making changes**



**Figure 5.4.16 The person's information after making changes**

People Database  
 Administrator: Admin  
 11:50:51  
 09/10/2023  
 Add New  
 search  
 No Sorting  
 Sort by Date Detected  
 Sort by Name

	Name	Status	ID	Nationality	Criminal History	Last Detected	Location
1	Cheong Kok Siong	Clear	990987654312	Russia	parking ticket	08/28/2023 12:31:22	Cam2
2	Tia-Kaztenie	Caution	000130130592	Malaysia	arson	09/10/2023 11:50:05	Cam2
3	Tan Su Hua	Clear	2367895421231	Singapore	speeding	08/27/2023 16:30:49	Cam1
4	Ng Suet Eng	Missing	17484987654321	Malaysia	NA	09/10/2023 11:46:01	Cam1
5	Tia Giam	Missing	000130130987	Spain	Tax Fraud	09/10/2023 11:48:15	Cam2
6	Lee Jia Jet	Caution	23459876543	China	robbery	09/10/2023 11:46:35	Cam2
7	Tea Kaz	Caution	000214135678	Spain	Homicide	09/10/2023 11:50:51	Cam1
8	Lim Wei	Caution	9876543265421	Malaysia	Manslaughter	09/10/2023 11:33:07	Cam1
9	Moh Ridwan	Clear	021234569876543	Indonesia	NA		
10	Hui Zhi	Clear	99098751234567	Taiwan	Jaywalking	09/10/2023 11:50:44	Cam1

**Figure 5.4.17 Changes made are saved**

The figures above show an example of if the user makes changes to update any person’s information. In this case, “Tea Kaz” has a new criminal history of homicide, and the status is now changed from “Clear” to “Caution”. Once the new information is updated and the “OK” button is clicked, the information is successfully updated and will be displayed in real time in the database.

The screenshot shows a web application window titled "Dialog". On the left side, there is a form with the following fields:

- From:** 09/10/2023
- From Time:** 00:00:00
- To:** 09/10/2023
- To Time:** 23:59:59
- Location:** All
- Status:** All
- List Names:**

Below the form are two buttons: "Ok" and "Reset". To the right of the form is a large empty rectangular area. At the bottom right of the window, there are two buttons: "Print" and "Back To Dashboard".

**Figure 5.4.18 Reports Generation Screen**

From the main dashboard, the user will be redirected to the screen shown in the figure above if they clicked on “Check Out Reports” button. The top left is a form for the user to input what the data ranges that they wish to generate reports from. The default view input is the current date time with all locations and statuses. The reset button will clear all inputs and change all data to the default.

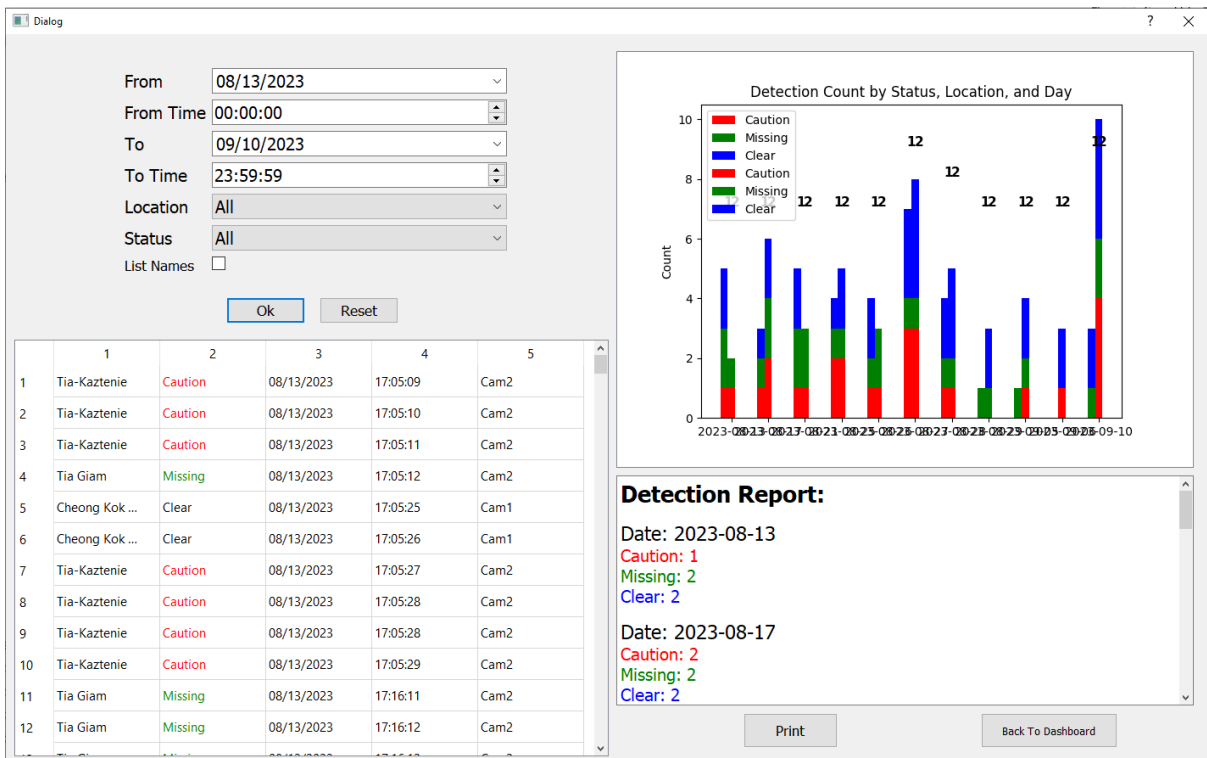


Figure 5.4.19 Example of a report (1/2)

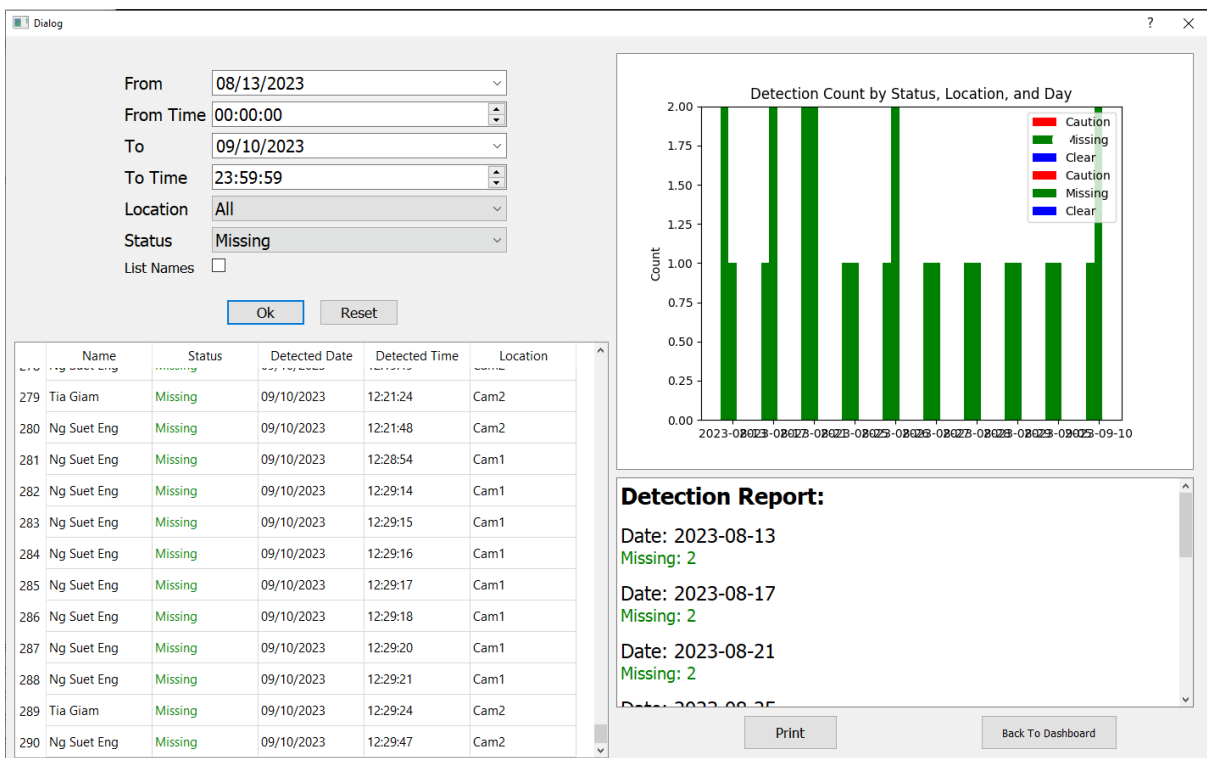


Figure 5.4.20 Example of a report (2/2)

The figures above show two examples of reports that can be generated based on the input. Based on the data input by the user, the table on the left will display all instances of detected

people from the database. A stacked clustered bar graph will be displayed on the top right based on the data in the table. Each cluster is for a date and colours are separated by status. A summary of the report is also shown on the bottom right under “Detection Report” to show the numbers of people detected on different dates.

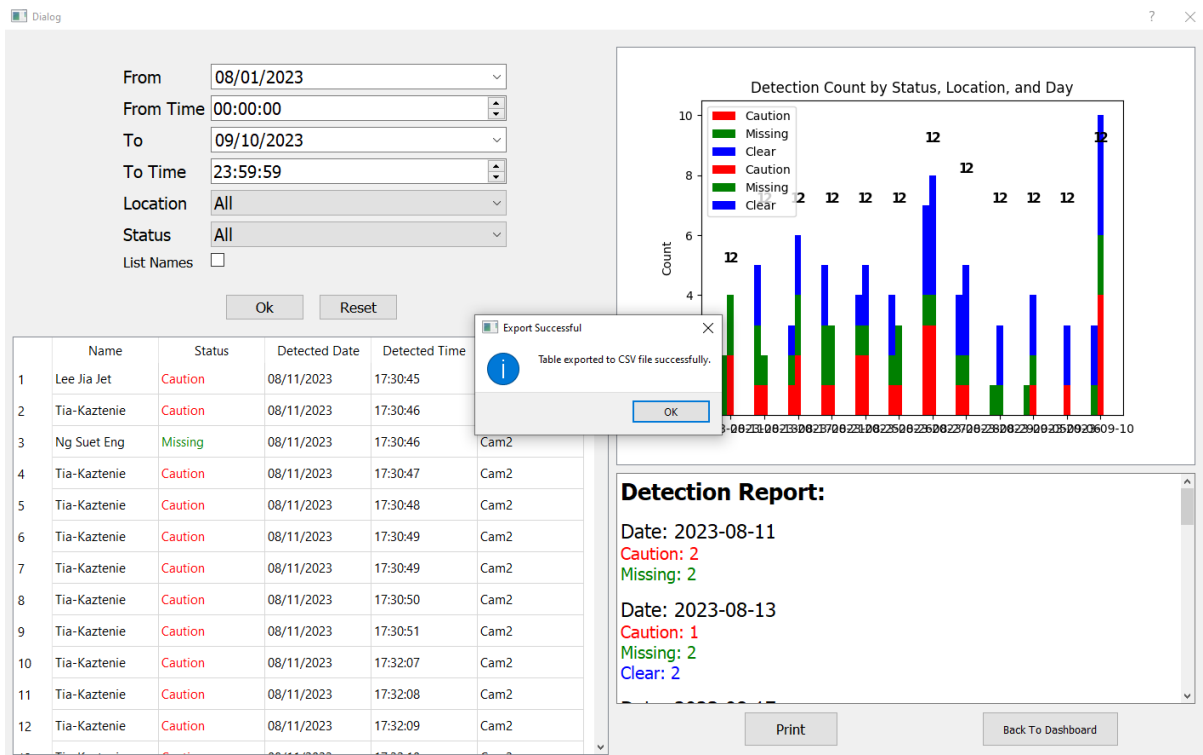


Figure 5.4.21 Saving a report to CSV File

	A	B	C	D	E	F	G	H
1	Name	Status	Detected Date	Detected Time	Location			
2	Lee Jia Jet	Caution	8/11/2023	17:30:45	Cam2			
3	Tia-Kaztenie	Caution	8/11/2023	17:30:46	Cam2			
4	Ng Suet Eng	Missing	8/11/2023	17:30:46	Cam2			
5	Tia-Kaztenie	Caution	8/11/2023	17:30:47	Cam2			
6	Tia-Kaztenie	Caution	8/11/2023	17:30:48	Cam2			
7	Tia-Kaztenie	Caution	8/11/2023	17:30:49	Cam2			
8	Tia-Kaztenie	Caution	8/11/2023	17:30:49	Cam2			
9	Tia-Kaztenie	Caution	8/11/2023	17:30:50	Cam2			
10	Tia-Kaztenie	Caution	8/11/2023	17:30:51	Cam2			
11	Tia-Kaztenie	Caution	8/11/2023	17:32:07	Cam2			
12	Tia-Kaztenie	Caution	8/11/2023	17:32:08	Cam2			
13	Tia-Kaztenie	Caution	8/11/2023	17:32:09	Cam2			
14	Tia-Kaztenie	Caution	8/11/2023	17:32:10	Cam2			

**Figure 5.4.22 Example of saved CSV File**

If the user clicks on the “Print” button on the bottom right of the screen, the report will be saved as a CSV file of all instances of detected people according to the input data. This CSV file will be saved to local storage so that it can be used for further analysis.

# Chapter 6

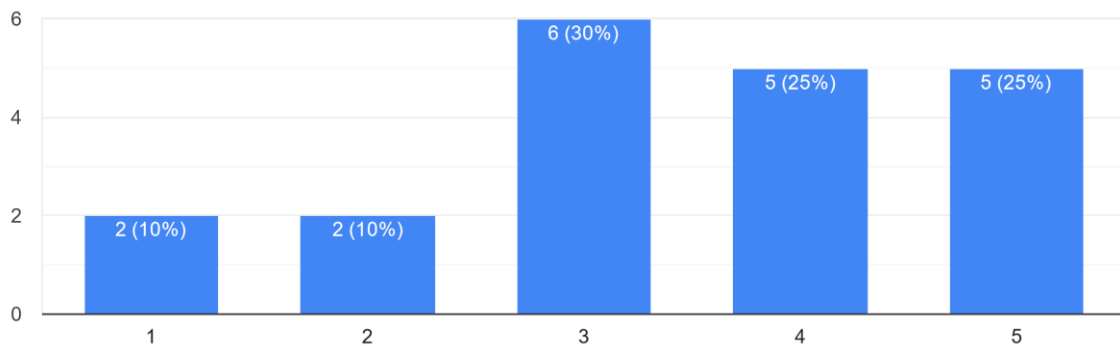
## System Evaluation and Discussion

### 6.1 System Evaluation Survey Results

After inviting 20 random students in UTAR to test out the system, they filled out a feedback survey form. The responses are as detailed.

1. How frequently do you interact with security surveillance systems or applications?

20 responses



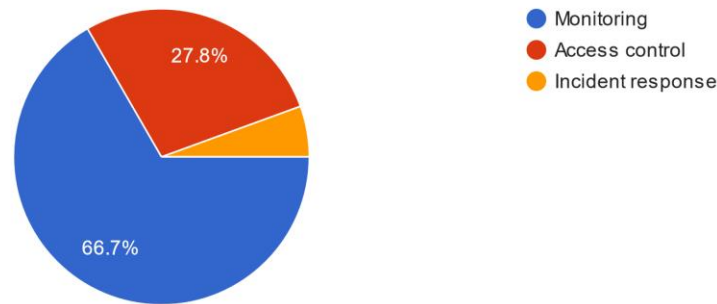
**Figure 6.1.1 Respondents' familiarity with Security Surveillance Systems**

The first question is to understand the respondents' familiarity with any form of security surveillance systems. A majority of the respondents are familiar with security surveillance systems as seen above, with only two respondents who have never interacted with any security surveillance application.



2. In what capacity do you primarily use the system(s)?

18 responses

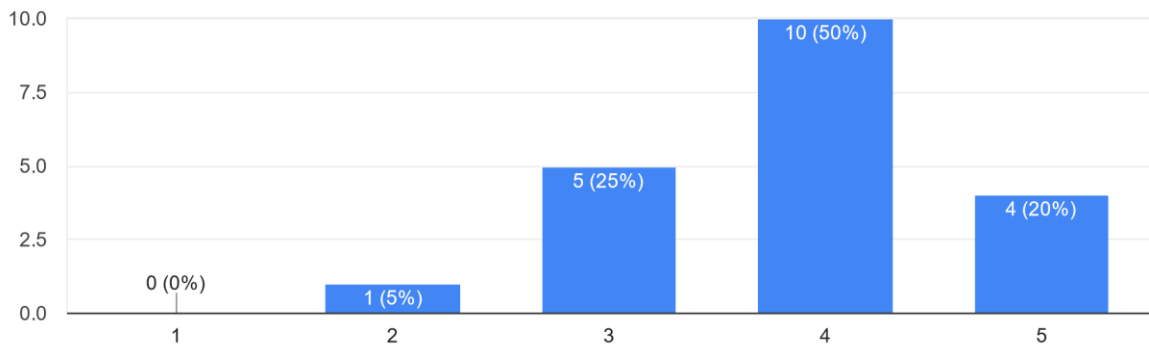


**Figure 6.1.2 Reasons for Using Security Surveillance Systems**

Most respondents used such systems for monitoring, followed by access control. Only one respondent used such systems for incident response. This may be because most respondents have basic home security surveillance systems and use them for monitoring the safety of their homes and families, and access control to entrances into their homes, rather than for any official incident responses.

3. How comfortable are you with using facial recognition technology?

20 responses

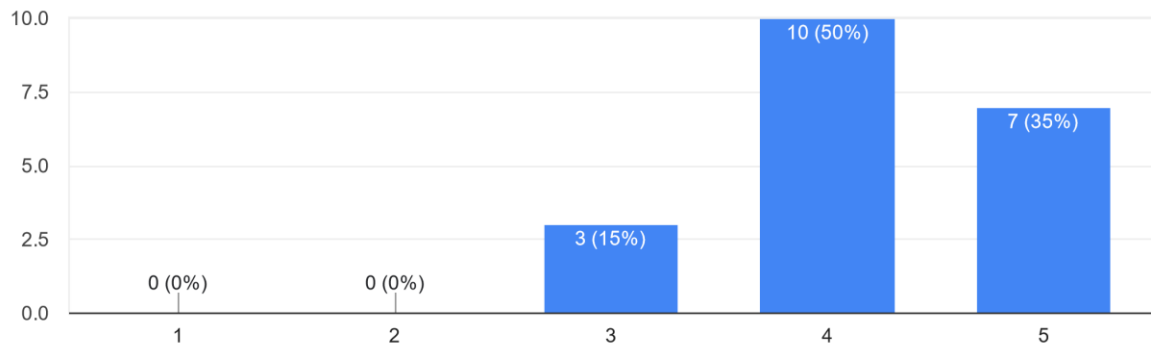


**Figure 6.1.3 Comfort of respondents using face recognition technology**

Most of the respondents are somewhat comfortable with using face recognition technology. This may be due to the prevalence of such technology in everyday usage, even in just unlocking mobile phones. Only one respondent was somewhat uncomfortable with the technology.

4. In your opinion, how important and necessary is a security surveillance system with facial recognition?

20 responses

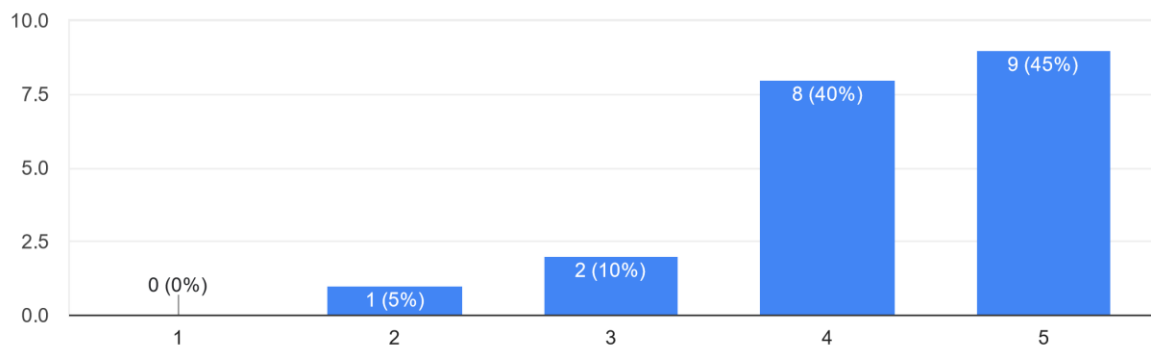


**Figure 6.1.4 How important respondents think face recognition technology is in security surveillance systems**

Most respondents believe that face recognition technology is very important for security surveillance systems. This shows that there is a demand for such an innovation.

5. How easy was it to understand and navigate the system?

20 responses

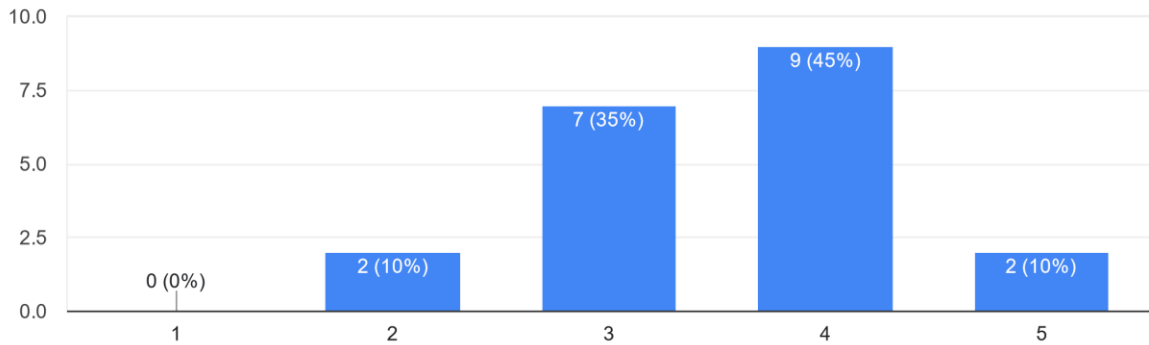


**Figure 6.1.5 Ease and Intuitiveness of Navigation**

Most respondents found that the developed system is very easy to understand and could easily navigate the system upon first usage. This means that the system does not have any overcomplicated design and is suitable for any user to use.

6. How would you rate your satisfaction with the user interface (UI) of the system?

20 responses

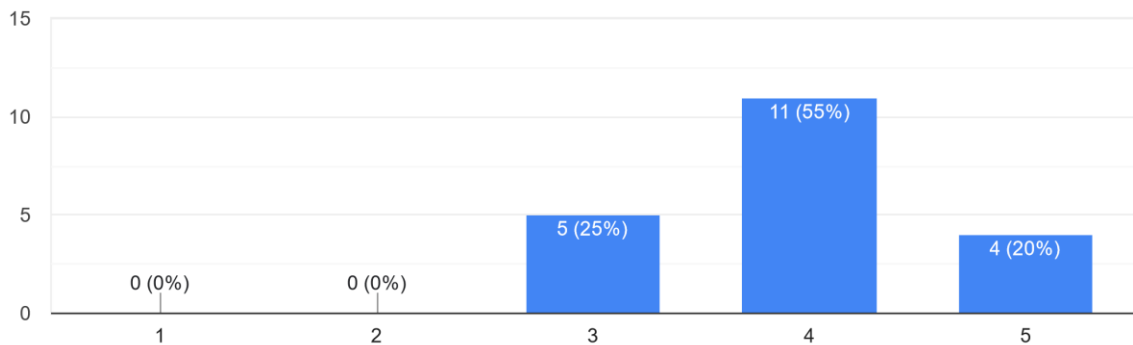


**Figure 6.1.6 Rating of User Interface**

Most of the users were satisfied with the UI design of the system, although many are neutral and there are also two respondents who are not satisfied with the UI. This means that there is room for improvement for the UI of the system.

7. Overall, how satisfied were you while using the system?

20 responses

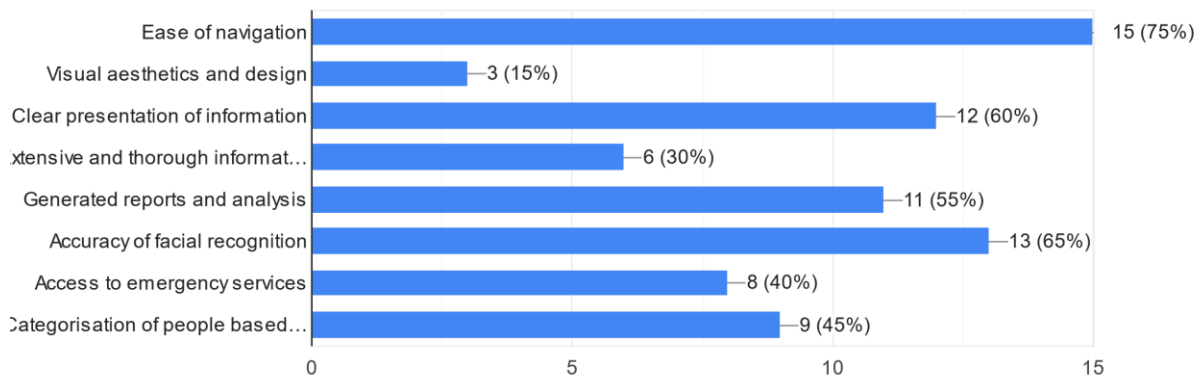


**Figure 6.1.7 Overall Satisfaction While Using System**

Most of the users were satisfied while using the system, which shows a positive review of the developed system, although there are still much room for improvement as can be seen by the five respondents who felt neutral while using the system.

### 8. What system features do you find to be valuable or useful?

20 responses

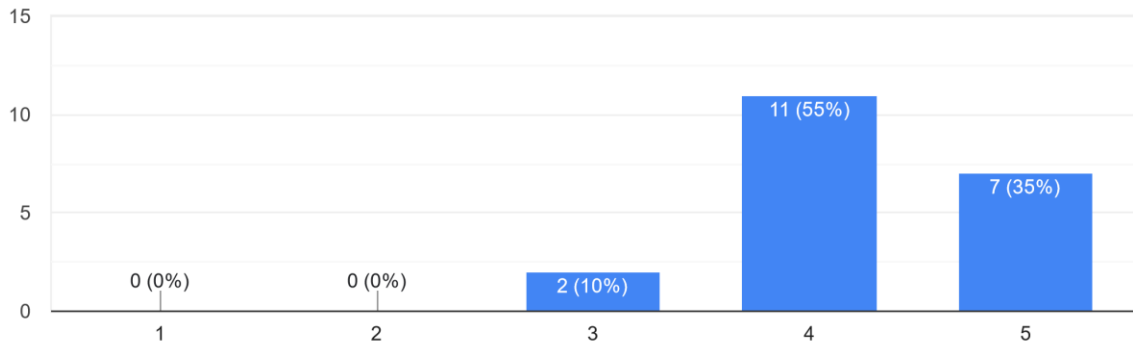


**Figure 6.1.8 Valuable or Useful System Features**

As shown in the figure 6.1.8, fifteen respondents chose ease of navigation, thirteen chose accuracy of facial recognition, twelve chose clear presentation of information, eleven chose generated reports and analysis, nine chose categorisation of people based on status, eight chose access to emergency services, six chose extensive and thorough information and functions, and only three chose visual aesthetics and design. This shows that the system's strengths lie in its ease of navigation, accuracy of facial recognition, and clear presentation of information, while it is weak in visual aesthetics and providing extensive functionality.

9. Do you find the display and categorisation of different people based on their status helpful in maintaining security?

20 responses



**Figure 6.1.9 Value of categorising people based on status**

Most respondents find that the categorisation and display of people based on their status is helpful in maintaining security. This may be because it is easier to keep track and monitor safety when the system has an automatic colour filter to show different degrees of categorisation of people.

10. Is there any aspect of this categorisation and display that you believe should be changed or could be improved? If so, please elaborate.

5 responses

How do you rate if somebody's criminal history is for 'Clear' or 'Caution'? Maybe they were wrongly accused of their crime / they repented, or they are somebody with no criminal history and suddenly wants to do something terrible

-

UI could be improved

improve the UI design

how to categorise if somebody is really dangerous or not? maybe will have racial or gender bias

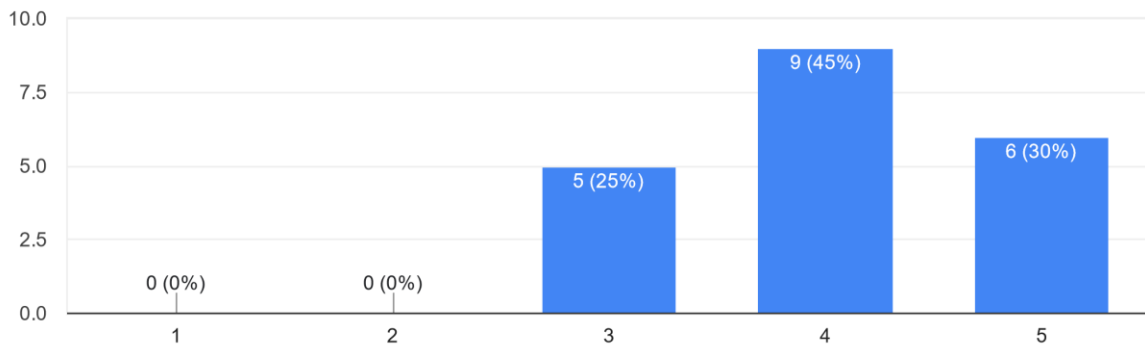
**Figure 6.1.10 Suggestions to Improve Categorisation and Display of People**

After omitting the blank response, 4 respondents provided replies to the question to whether there are any suggestions to improving or changing the method of categorising people based

on their status. Two of the suggestions were to improve UI design, which is irrelevant to the question, but will be kept for general suggestions for the system. Two of the remaining responses questioned how people would be categorised and cited the possibility of discrimination. These are ethical concerns that must be seriously considered. A list of relevant crimes and their danger levels would have to be drawn up and followed to avoid any gender or racial discrimination.

11. How accurate do you find the facial recognition feature in identifying individuals?

20 responses

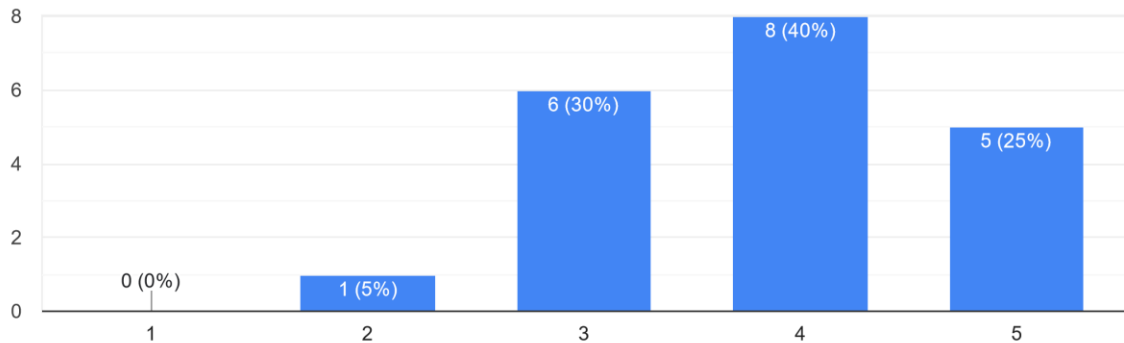


**Figure 6.1.11 Accuracy of Face Recognition**

The majority of respondents found that the face recognition was accurate, and this would be due to the usage of the CNN-model. As this project was delivered for testing using the default webcam of the laptop it was developed on and an Android mobile camera, it can be assumed that the accuracy of the face recognition will be higher when it uses actual high-definition CCTV cameras.

12. How satisfied are you with the speed and responsiveness of the system when processing facial recognition data?

20 responses

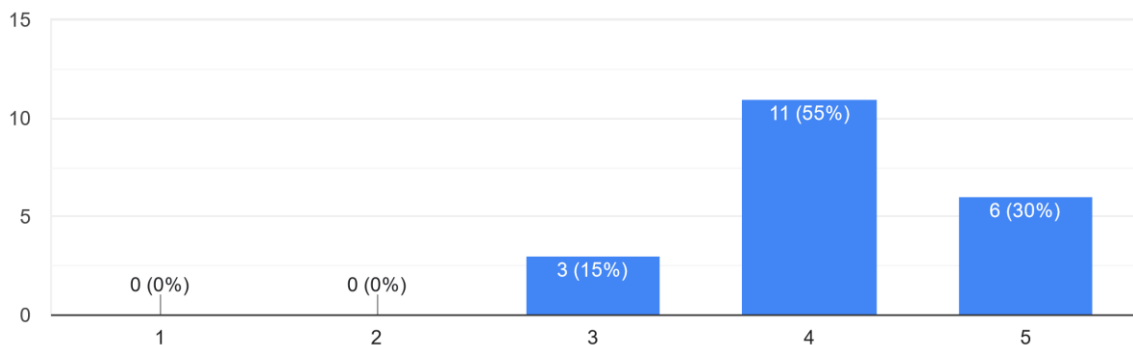


**Figure 6.1.12 Speed and Responsiveness of System**

Most of the respondents are satisfied with the speed and responsiveness of the system when processing face recognition data. However, one respondent is slightly unsatisfied. This may be due to the slow CPU power of the laptop that the system is deployed on when provided to respondents for testing.

13. Do you find the generated reports and analysis by the system to be helpful?

20 responses



**Figure 6.1.13 Value of Generated Reports and Analysis**

Most respondents are satisfied with the value of the generated reports and analysis. There may still be room for improvement and expansion of generated reports.

14. What other generated reports and analysis would you like the system to have?

4 responses

A screenshot of a survey interface showing four responses to question 14. The responses are displayed in a list format, each in a light gray box. The first response is a hyphen (-). The second response is 'phone number'. The third response is 'report of admins that enter the system'. The fourth response is 'number of people detected during different time of day'.

**Figure 6.1.14 Suggestions for Other Reports**

After omitting the blank response, there are three suggestions for other reports that the system can generate that users would be interested to have. A respondent suggested having a report of phone numbers, although the context is unclear as to what they mean. Another respondent suggested having a report of administrators that access the system, which can be useful for internal analysis. Finally, another respondent also suggested having a report of the number of people detected during different times of day, which can be useful for high traffic public locations like shopping malls and train stations.

15. What additional features or improvements would you like to see in the system?

You may also share any experience you had with the system.

4 responses

A screenshot of a survey interface showing three responses to question 15. The responses are displayed in a list format, each in a light gray box. The first response is a hyphen (-). The second response is 'Able to screenshot an unknown person in the video and save that person's identity'. The third response is 'function to capture image of unknown person and save their identity'.

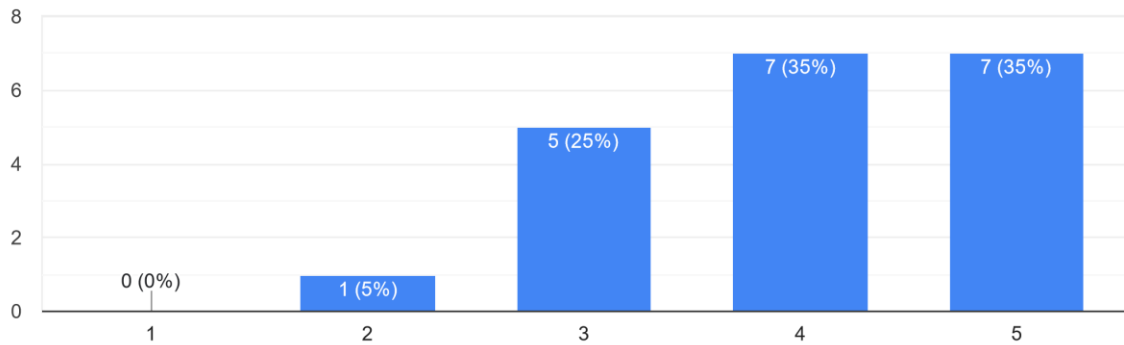
**Figure 6.1.15 Suggestions for Other Features or Improvements**

An open-ended question was also provided to respondents for any comments they have with the system or recommendations. After including the comments of improving the UI design from Question 10, there are four general suggestions of improvement for the system. Respondents suggested being able to capture the image of an unknown or unrecorded person from the video feed and saving their identity. This would be immensely useful and requires further exploration.



16. How likely are you to recommend this security surveillance system to others?

20 responses



**Figure 6.1.16 Likely Recommendation to Others**

Most of the respondents are highly likely to recommend this security surveillance system to others. However, one respondent was unlikely to. This means that the system still has room for improvement, and will take suggestions into consideration to improve, such as UI design and adding more features.

## 6.2 Testing Setup and Results

### 6.2.1 Unit Test 1 – Login

Objective: Ensure only authorised user logs into system.

<b>Input</b>	<b>Expected Output</b>	<b>Actual Output</b>
User login using authorised username and password.	User successfully logs into the system.	User successfully logs into the system.
User login using unauthorised username or password.	User is shown a warning message and is unable to log into the system.	User is shown a warning message and is unable to log into the system.
User login with no input.	User is shown a warning message and is unable to log into the system.	User is unable to log into the system.

**Table 6.2.1 Unit Test 1 – Login**

### 6.2.2 Unit Test 2 – Dashboard

Objective: Ensure that user can interact with functionalities.

<b>Input</b>	<b>Expected Output</b>	<b>Actual Output</b>
Click “Check Cameras” button	User is redirected to dialog with all camera feeds.	A dialog with all camera feeds is opened.  Video feed in Dashboard dialog is paused when camera source is used by new dialog showing all camera feeds.
Click “Database of People” button	User is redirected to dialog with people database.	A new dialog with people database is opened.
Click “Check Out Reports” button	User is redirected to dialog to generate reports.	A new dialog to generate reports is opened.

**Table 6.2.2 Unit Test 2 – Dashboard**

### 6.2.3 Unit Test 3 – Viewing All Camera Feeds

Objective: Ensure that user can interact with all functionalities while monitoring video feeds.

<b>Input</b>	<b>Expected Output</b>	<b>Actual Output</b>
Click on any detected person in the people detected list.	User views detailed information of the person they clicked on.	A dialog with detailed information of the person is opened.
Click “Emergency Services” button	User directly contacts nearest emergency services.	A dialog showing different emergency contact information is opened.
Click “Close” button in Emergency Services dialog	The emergency contacts dialog is closed.	The emergency contacts dialog is closed.
Click “Back to Dashboard” button	User is redirected back to dashboard.	All cameras dialog is closed.
Make any changes to a person’s information when clicked “Ok”.	Changes saved successfully.	Changes saved successfully.

**Table 6.2.3 Unit Test 3 – Viewing All Camera Feeds**

## 6.2.4 Unit Test 4 – Viewing Database of People

Objective: Ensure that user can interact with all functionalities while viewing database of people.

<b>Input</b>	<b>Expected Output</b>	<b>Actual Output</b>
Click into any row in the table.	The correct person's full information is displayed.	A dialog displaying the person's full information is opened and displayed.
Make changes to information and clicked "Ok" button.	The new data is saved successfully.	A message box saying the data is saved successfully appears.  The data is successfully updated in Firebase and displayed on the system.
Click "Cancel" after making any changes to information.	Changes are not saved.  Person's detailed information dialog is closed.	Changes are not saved.  Person's detailed information dialog is closed.
Type anything into the search bar.	Table automatically filters and displays rows that have the keyword typed in the search bar.	Table automatically filters and displays rows that have the keyword typed in the search bar.
Click "Sort by Date Detected" radio button.	Table automatically sorts itself by the date detected from oldest to latest.	Table automatically sorts itself by the date detected from oldest to latest.
Click "Sort by Name" radio button.	Table automatically sorts itself alphabetically by name of person.	Table automatically sorts itself alphabetically by name of person.
Click "No Sorting" radio button.	Table resets all sorting to default display following Firebase.	Table resets all sorting to default display following Firebase.

**Table 6.2.4 Unit Test 4 – Viewing Database of People**

### 6.2.5 Unit Test 5 – Generating Reports

Objective: Ensure that user can interact with all functionalities while generating reports.

<b>Input</b>	<b>Expected Output</b>	<b>Actual Output</b>
Click “Ok” after input data fields in form.	Graph is generated.	Graph is generated.  Summary of graph is displayed.  Table of all instances based on data range input is displayed.
Click “Reset” button.	All fields in form are emptied.  Table, graph, and summary of graph is cleared.	All fields in form are reset to default inputs.  Table, graph, and summary of graph is cleared.
Click “Print” button.	Save data output into CSV file.	Data output is saved into CSV file in local storage.
Click “Back to Dashboard” button.	User is redirected back to dashboard.	Reports dialog is closed.

**Table 6.2.5 Unit Test 5 – Generating Reports**

### **6.3 Implementation Issues and Project Challenges**

Numerous challenges were faced during the development of this project. With the various options available, it was a struggle to find which algorithms would work best in the system with the computing resources on hand. Certain algorithms were better in terms of accuracy and speed, but the laptop used in the development of this project could not support their heavy resource consumption, especially of RAM.

Testing of the system may also not be fully accurate as testing was not done with actual CCTV cameras, or with cameras placed in high location where these cameras would usually be placed. The testing was often done with faces closer to the camera, so the results may be skewed to have more positive results. Testing was also mostly done on the developers' own face without testing on hundreds of faces at any one time. The greatest number of faces on frame was only five, thus the system could not be tested on how much it could handle face recognition processing.

During the initial stages of the project, there were plans to make the system a web application by using Django or other similar web-service that support web applications. However, there were too many challenges in understanding how to utilise these services while juggling the development of the system. This was exchanged for creating a computer application instead. This may not be preferable since web applications are more flexible for actual deployment.

## 6.4 Objectives Evaluation

From the survey results that were collected, 75% of respondents feel that the face recognition system is accurate, with 30% of them feeling that it is extremely accurate. The CNN model used also has a high accuracy rate. With high accuracy of face recognition, it would be more efficient and effective at detecting people in a blacklist or whitelist in a location, especially among crowds.

Having a security system that is equipped with facial recognition is also without a doubt more beneficial than not at maintaining security access, especially in locations that necessitate access control like government buildings and concert halls.

To reiterate, the project objectives are:

- Enhance security measures for accessing sensitive facilities and venues through the implementation of facial biometric recognition technology.
- Leverage a facial recognition system to identify suspects efficiently and accurately within densely populated areas.
- Use facial recognition technology to identify potentially dangerous persons and missing individuals who may not be publicly recognised, thereby alerting authorities to potential threats.

These objectives were mostly met by the system developed. Although there were difficulties in pushing the test limits for the second objectives due to project challenges, the project can be further explored to include additional equipment which can expand the scope of the project.

# Chapter 7

## Conclusion and Recommendations

### 7.1 Conclusion

This project involves developing a computer application system designed for security surveillance. It incorporates facial recognition algorithms to swiftly and accurately identify dangerous individuals and missing persons in public areas, ensuring crowd safety and security. While CCTV surveillance is prevalent today, traditional systems primarily serve as historical records of incidents, where recordings are referred when an unwanted situation has already occurred. In contrast, this project aims to proactively prevent crimes and incidents by detecting and recognising potential threats within crowds. Unlike existing surveillance solutions which are typically complex and costly, like Avigilon and FaceMe, the system developed in this project enables straightforward monitoring of individuals entering premises, flexible camera management, and prompt contact with emergency services upon identifying dangerous individuals. The system offers accessibility and scalability to various organisation and establishments, ensuring safety and peace in as many locations as possible.

The main face recognition algorithm used in this project is Convolutional Neural Network (CNN) with Dlib and added Cascade Classifier after researching the many algorithms available. This Python programme is able to accurately recognise people that it has records of in its database. This system uses Firebase's Realtime Database and Storage Bucket for efficient real-time storage and data retrieval. The system is also able to generate reports with the data it collected for analytics purposes.

Development was challenging, especially due to the lack of resources in terms of devices. There would have to be more testing to push the limits of the system by attaching more camera sources and from multiple angles, while having the computing power to process the numerous video feeds and face recognition. In conclusion, although system testers responded that they were overall satisfied with the application, there is still much room for improvement and exploration of more functionalities.



## 7.2 Recommendations

Following the suggestions from survey respondents, the system's UI design has much room for improvement. Design choices like colour schemes can be considered, especially for security personnel who may be tasked with monitoring the digital screens so having darker colours like dark blue or dark grey would be suitable. Since UI design was not at the forefront of the planning due to the focus of developing the functionalities of the system, future working on this project can have this aspect as a priority.

More functionalities can also be added. As a respondent had suggested, an extremely useful function to have is to be able to save the image of an unknown person detected so that identification can be done. This would be an answer to questions raised of detecting unknown persons that may be suspicious. In addition to this, the project should also explore recording all video footage and storing it in a cloud server perhaps temporarily for six months. This would be useful for reference if any situations arise.

Other than that, the project can also explore having more reports that can be generated, as suggested by survey respondents. Reports like checking the logs of all administrators and traffic during different hours of the days would be useful for various executive decision-making.

## REFERENCES

- [1] Vignes, “4 children go missing every day in Malaysia, here's what you can do about it...,” AskLegal.MY, 12 March 2019. [Online]. Available: <https://asklegal.my/p/child-missing-Malaysia-nur-alerts-mms-warning-signup>. [Accessed 18 March 2023].
- [2] A. Ciocan, “AI for Facial Recognition Technology,” Academia.edu, 2021. [Online]. Available: [https://www.academia.edu/50976731/AI\\_for\\_Facial\\_Recognition\\_Technology](https://www.academia.edu/50976731/AI_for_Facial_Recognition_Technology). [Accessed 27 November 2022].
- [3] 王淑静, “临沂“雪亮工程”:治安防控 群众真正参与进来了,” 中国长安网, 2 October 2016. [Online]. Available: <https://archive.li/7gpbm>. [Accessed 29 November 2022].
- [4] Airport Technology, “News: Malaysia Airports Introduces Passenger Reconciliation System,” Airport-Technology.com, 8 February 2021. [Online]. Available: <https://www.airport-technology.com/news/malaysia-airports-malaysia-airlines-introduce/>. [Accessed 28 November 2022].
- [5] B. Taylor, “Registered sex offender charged with masturbating in public places more than a dozen times,” ABC13 News, 1 November 2022. [Online]. Available: <https://abc13.com/registered-sex-offender-clarence-dean-dorris-public-masturbation-charged-with-indecent-explore/12403483/>. [Accessed 28 November 2022].
- [6] V. D. A. Kumar, V. D. A. Kumar, S. Malathi, K. Vengatesan and M. Ramakrishnan, “Facial Recognition System for Suspect Identification Using a Surveillance Camera,” Pleiades Publishing, 2018.
- [7] A. Nadeem, M. Ashraf, N. Qadeer, K. Rizwan, A. Mehmood, A. AlZahrani, F. Noor and Q. H. Abbasi, “Tracking Missing Person in Large Crowd Gathering Using Intelligent Video Surveillance,” *Artificial Intelligence Methods for Smart Cities*, vol. 22, no. 14, p. 25, 2022.

- [8] J. Ijaradar and J. Xu, "A Cost-efficient Real-time Security Surveillance System Based on Facial Recognition Using Raspberry Pi and OpenCV," *Current Journal of Applied Science and Technology*, vol. 41, no. 5, p. 12, 2022.
- [9] K. K. Suwarno Liang, "Analysis of Face Recognition Algorithm: Dlib and OpenCV," *JITE (Journal of Informatics and Telecommunication Engineering)*, vol. 4, no. July, p. 12, 2020.
- [10] CyberLink Corp., "FaceMe," CyberLink Corp., [Online]. Available: <https://www.cyberlink.com/faceme>. [Accessed 28 March 2023].
- [11] CyberLink Corp., "FaceMe Solution - Security," CyberLink Corp., [Online]. Available: <https://www.cyberlink.com/faceme/solution/security/overview>. [Accessed 13 March 2023].
- [12] CyberLink Corporation, "Newsroom - CyberLink FaceMe® Tops Facial Recognition Vendor Test," CyberLink Corporation, 28 January 2021. [Online]. Available: [https://www.cyberlink.com/eng/press\\_room/view\\_4709.html](https://www.cyberlink.com/eng/press_room/view_4709.html). [Accessed 28 March 2023].
- [13] Business Wire, "Good Finance Chooses CyberLink's FaceMe® Facial Recognition Technology to Perform Identity Verification for Its Online Banking Services," Yahoo! Finance, 19 July 2022. [Online]. Available: <https://finance.yahoo.com/news/good-finance-chooses-cyberlink-faceme-130000499.html>. [Accessed 28 March 2023].
- [14] Avigilon Corp., "Artificial Intelligence and Video Analytics," Avigilon Corporation, [Online]. Available: <https://www.avigilon.com/products/ai-video-analytics>. [Accessed 2023 March 31].


# APPENDICES

## SURVEY FEEDBACK

### Feedback Survey of User Experience for Security Surveillance System with Facial Recognition - a UTAR Final Year Project

I am Tia-Kaztenie Giam, a final year undergraduate student in Universiti Tunku Abdul Rahman, studying Bachelor of Information Systems (Honours) Business Information Systems. This system was developed for my final year project.

This survey is to collect valuable feedback and insights regarding the security surveillance system that was developed and provided to you for testing. Your feedback is vital to construct a comprehensive, useful, and user-friendly security surveillance system. This survey will take 5 minutes to complete.

tia.kaztenieghz@1utar.my [Switch account](#) 

\* Indicates required question


**Email \***

Record tia.kaztenieghz@1utar.my as the email to be included with my response

**Acknowledgements \***

By proceeding to complete this survey, you acknowledge that your feedback regarding the user experience of the security surveillance system with facial recognition will be used for analysis and further improvement of the system. Your input is invaluable in helping enhance the system's features and usability. Your responses will be treated confidentially and used only for research purposes. Your time and insights are appreciated in contributing to the enhancement of the security solution.

Yes, I agree

[Next](#)  Page 1 of 3 [Clear form](#)

Never submit passwords through Google Forms.

This form was created inside of Universiti Tunku Abdul Rahman. [Report Abuse](#)

Google Forms

## System Usage

This section is to understand your usage of similar systems.

1. How frequently do you interact with security surveillance systems or applications?

*Do you have a home surveillance system / children monitor / pet monitor system for your home?*

*Do you work with security surveillance systems often in work / study?*

1      2      3      4      5

Never                                    Very Frequently

2. In what capacity do you primarily use the system(s)?

*Ignore this question if you answered "Never" for the previous question.*

- Monitoring
- Access control
- Incident response
- Other: \_\_\_\_\_

3. How comfortable are you with using facial recognition technology? \*

1      2      3      4      5

Not comfortable                                    Very comfortable

4. In your opinion, how important and necessary is a security surveillance system \* with facial recognition?

1      2      3      4      5

Not necessary or important                                    Very necessary and important

[Back](#)

[Next](#)

Page 2 of 3

[Clear form](#)

Never submit passwords through Google Forms.

This form was created inside of Universiti Tunku Abdul Rahman. [Report Abuse](#)

Google Forms

## User Experience

This section is to understand your thoughts and impressions after trying out the security surveillance system that was developed.

5. How easy was it to understand and navigate the system? \*

	1	2	3	4	5	
Very difficult	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very easy

6. How would you rate your satisfaction with the user interface (UI) of the system? \*

	1	2	3	4	5	
Very unsatisfied	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very satisfied

7. Overall, how satisfied were you while using the system? \*

	1	2	3	4	5	
Very unsatisfied	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very satisfied

8. What system features do you find to be valuable or useful? \*

- Ease of navigation
- Visual aesthetics and design
- Clear presentation of information
- Extensive and thorough information and functions
- Generated reports and analysis
- Accuracy of facial recognition
- Access to emergency services
- Categorisation of people based on status
- Other: \_\_\_\_\_

9. Do you find the display and categorisation of different people based on their status helpful in maintaining security? \*

*i.e., since people are categorised by status ("Caution", "Missing", "Clear") and displayed as such by colour (red, green, black), do you think this is a good method of security surveillance?*

	1	2	3	4	5	
Not helpful	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very helpful

10. Is there any aspect of this categorisation and display that you believe should be changed or could be improved? If so, please elaborate.

Your answer

---

11. How accurate do you find the facial recognition feature in identifying individuals? \*

	1	2	3	4	5	
Very inaccurate	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very accurate

12. How satisfied are you with the speed and responsiveness of the system when processing facial recognition data? \*

	1	2	3	4	5	
Very unsatisfied	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very satisfied

13. Do you find the generated reports and analysis by the system to be helpful? \*

	1	2	3	4	5	
Very unhelpful	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very helpful

14. What other generated reports and analysis would you like the system to have?

Your answer

---

15. What additional features or improvements would you like to see in the system?  
You may also share any experience you had with the system.

Your answer

16. How likely are you to recommend this security surveillance system to others? \*

	1	2	3	4	5	
Very unlikely	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very likely

[Back](#)

[Submit](#)

Page 3 of 3

[Clear form](#)

Never submit passwords through Google Forms.

This form was created inside of Universiti Tunku Abdul Rahman. [Report Abuse](#)

Google Forms



# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> 3, 3	<b>Study week no.:</b> 2
<b>Student Name &amp; ID:</b> Tia-Kaztenie Giam Hui Zhi 20ACB05262	
<b>Supervisor:</b> Mr Su Lee Seng	
<b>Project Title:</b> Facial Recognition System for Crowd Security	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Planning for project timeline
- Discussion system features

## 2. WORK TO BE DONE

- Explore system features with Python code.
- Revise Chapter 1 and 2 in report

## 3. PROBLEMS ENCOUNTERED

- No issues yet

## 4. SELF EVALUATION OF THE PROGRESS

- To put more effort with exploring system design



\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> 3, 3	<b>Study week no.:</b> 4
<b>Student Name &amp; ID:</b> Tia-Kaztenie Giam Hui Zhi 20ACB05262	
<b>Supervisor:</b> Mr Su Lee Seng	
<b>Project Title:</b> Facial Recognition System for Crowd Security	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Revised Chapter 1 and 2 in report
- Designed UI screens

## 2. WORK TO BE DONE

- Continue develop system features.
- Explore database options.

## 3. PROBLEMS ENCOUNTERED

- Connecting system features to designed UI screens.

## 4. SELF EVALUATION OF THE PROGRESS

- More effort is needed to understand development



\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> 3, 3	<b>Study week no.:</b> 6
<b>Student Name &amp; ID:</b> Tia-Kaztenie Giam Hui Zhi 20ACB05262	
<b>Supervisor:</b> Mr Su Lee Seng	
<b>Project Title:</b> Facial Recognition System for Crowd Security	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Connect to Firebase Realtime Database
- Continue development of system

## 2. WORK TO BE DONE

- Continue develop system features.
- Connect more than one camera.

## 3. PROBLEMS ENCOUNTERED

- UI files and functional files are too messy.

## 4. SELF EVALUATION OF THE PROGRESS

- Have better understanding of code and file organisation.



\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> 3, 3	<b>Study week no.:</b> 8
<b>Student Name &amp; ID:</b> Tia-Kaztenie Giam Hui Zhi 20ACB05262	
<b>Supervisor:</b> Mr Su Lee Seng	
<b>Project Title:</b> Facial Recognition System for Crowd Security	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Able to connect more than one camera to system.

## 2. WORK TO BE DONE

- Figure out lag and Firebase issues

## 3. PROBLEMS ENCOUNTERED

- Lag issues
- Firebase issues

## 4. SELF EVALUATION OF THE PROGRESS

- Need better understanding and practice for coding.



\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> 3, 3	<b>Study week no.:</b> 10
<b>Student Name &amp; ID:</b> Tia-Kaztenie Giam Hui Zhi 20ACB05262	
<b>Supervisor:</b> Mr Su Lee Seng	
<b>Project Title:</b> Facial Recognition System for Crowd Security	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Fixed lag and Firebase issues

## 2. WORK TO BE DONE

- Complete development of system
- System testing and evaluation

## 3. PROBLEMS ENCOUNTERED

- Many bugs in system

## 4. SELF EVALUATION OF THE PROGRESS

- More effort is needed to be on schedule.



\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

# FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

<b>Trimester, Year:</b> 3, 3	<b>Study week no.:</b> 12
<b>Student Name &amp; ID:</b> Tia-Kaztenie Giam Hui Zhi 20ACB05262	
<b>Supervisor:</b> Mr Su Lee Seng	
<b>Project Title:</b> Facial Recognition System for Crowd Security	

## 1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Complete system testing and evaluation

## 2. WORK TO BE DONE

- Complete report
- Finalise system development lifecycle

## 3. PROBLEMS ENCOUNTERED

- No issues

## 4. SELF EVALUATION OF THE PROGRESS

- Project is on schedule



\_\_\_\_\_  
Supervisor's signature



\_\_\_\_\_  
Student's signature

# POSTER



UNIVERSITI TUNKU ABDUL RAHMAN  
FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY  
FINAL YEAR PROJECT

# FACIAL RECOGNITION SYSTEM FOR CROWD SECURITY

## INTRODUCTION

Security surveillance is needed to keep the safety of the public. However, outdated surveillance VMS may not prevent crimes as well as having a facial recognition system integrated to detect suspicious persons.

## OBJECTIVES

Keep local authorities alert for suspicious or missing persons by using FR technology to quickly and accurately identify them, thus strengthening the security of sensitive facilities and maintaining crowd security.

## METHODS & APPROACHES

- Developed using Python programming with OpenCV and Face-Recognition libraries
- Using agile methodology - constant testing and reiteration for the best outcome
- Keeping it direct and scalable for organisations from tiny mom-and-pop stores to giant concert venues
- Provide summarised reports and analytics for overviews
- Administer the database of people to control and tighten security
- Add or remove cameras into the system to desired scale

## CONCLUSION

- A highly scalable and straightforward video surveillance system web application integrated with facial recognition technology and alert system that can replace outdated surveillance systems

## PROJECT DEVELOPER

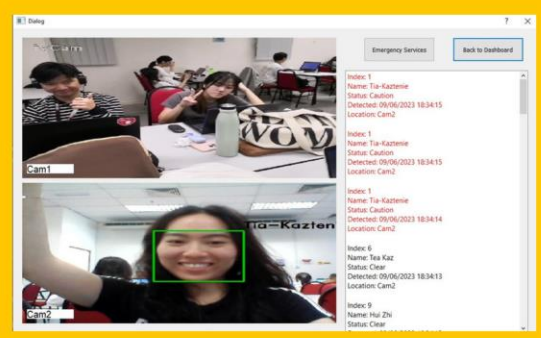
Tia-Kaztenie Giam Hui Zhi  
Business Information Systems

## PROJECT SUPERVISOR

Ts. Su Lee Seng







## PLAGIARISM CHECK RESULT

FYP2

### ORIGINALITY REPORT

<b>3%</b>	<b>2%</b>	<b>0%</b>	<b>%</b>
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	<b>eprints.utar.edu.my</b> Internet Source	<b>1%</b>
<b>2</b>	<b>pdfs.semanticscholar.org</b> Internet Source	<b>&lt;1%</b>
<b>3</b>	<b>eprints.utm.my</b> Internet Source	<b>&lt;1%</b>
<b>4</b>	<b>www.rivier.edu</b> Internet Source	<b>&lt;1%</b>
<b>5</b>	<b>archive.org</b> Internet Source	<b>&lt;1%</b>
<b>6</b>	<b>Muhammad Nugraha, Ricak Agus, Halimil Fathi, Rizki Baginda. "DEVELOPMENT A WEB-BASED STUDENT INTERNSHIP APPLICATION USING LARAVEL FRAMEWORK", Journal of Information Technology and Its Utilization, 2023</b> Publication	<b>&lt;1%</b>
<b>7</b>	<b>github.com</b> Internet Source	<b>&lt;1%</b>



8 Lingyu Wang, Graham Leedham. "Chapter 7 A Thermal Hand Vein Pattern Verification System", Springer Science and Business Media LLC, 2005 <1 %  
Publication

---

9 [papers.academic-conferences.org](http://papers.academic-conferences.org) <1 %  
Internet Source

---

10 [spie.org](http://spie.org) <1 %  
Internet Source

---

Exclude quotes On

Exclude matches < 10 words

Exclude bibliography On

<b>Universiti Tunku Abdul Rahman</b>			
<b>Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)</b>			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1



**FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY**

<b>Full Name(s) of Candidate(s)</b>	Tia-Kaztenie Giam Hui Zhi
<b>ID Number(s)</b>	20ACB05262
<b>Programme / Course</b>	Bachelor of Information Systems (Honours) Business Information Systems
<b>Title of Final Year Project</b>	Facial Recognition for Crowd Security

<b>Similarity</b>	<b>Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)</b>
<b>Overall similarity index: <u>  3  </u> %</b>  <b>Similarity by source</b> Internet Sources: <u>  2  </u> % Publications: <u>  0  </u> % Student Papers: <u>  0  </u> %	
<b>Number of individual sources listed of more than 3% similarity: <u>  0  </u></b>	
<b>Parameters of originality required and limits approved by UTAR are as Follows:</b> (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

*Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.*

*SuLee*

\_\_\_\_\_  
Signature of Supervisor

\_\_\_\_\_  
Signature of Co-Supervisor

Name: Su Lee Seng

Name: \_\_\_\_\_

Date: 13 September 2023

Date: \_\_\_\_\_



**UNIVERSITI TUNKU ABDUL RAHMAN**

**FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY  
(KAMPAR CAMPUS)**

**CHECKLIST FOR FYP2 THESIS SUBMISSION**

Student Id	20ACB05262
Student Name	Tia-Kaztenie Giam Hui Zhi
Supervisor Name	Su Lee Seng

<b>TICK (√)</b>	<b>DOCUMENT ITEMS</b>
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
	Front Plastic Cover (for hardcopy)
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

\*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)

Date: 13 September 2023