Ain Shams University
Faculty of Computer & Information Sciences
New Programs for Higher Education
Cybersecurity Program

جامعة عين شمس
كلية الحاسبات والمعلومات
برامج جديدة للتعليم العالي
برنامج الأمن السيبراني

Multimodal Biometric Border Control System

By:

Abdelrahman Khaled Abdullah Hamed

Abdelrahman Mohamed Yehia Sohsah

Moamen Mahmoud Ibrahim Mohamed

Nour Amr Mohamed Bahgat

Salma Abdelmonem Abdelrahman Abbas

Zeiad Mahmoud Gomaa Mohamed

Under Supervision of:

Dr. Hanan Hindy [BSc, MSc, PhD],
Computer Science Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

TA. Beshoy Victor [BSc],
Software Engineering Department,
Faculty of Computer and Information Sciences,
Ain Shams University.

June 2025

Acknowledgement

Alhamdulillah, all praise is due to **Allah (SWT)**, who gave us the ability, health, and time to complete this project. His mercy has surrounded us throughout this journey, especially during difficult and challenging moments. Without His guidance and blessings, none of this would have been possible.

We would like to express our deep and sincere gratitude to our supervisors, Dr. Hanan Hindy and TA. Beshoy Victor, for their continuous support, insightful advice and encouragement throughout this journey. This experience has been extremely enriching, and the lessons we have learned will always stay with us. Thank you for being such an impactful part of this chapter and making it truly unforgettable, we are forever thankful.

Last but certainly not least, we are truly thankful to our families for their prayers and sacrifices made to see us reach this milestone. Their love, patience and belief in us have been a constant source of strength.

To all those we have acknowledged: your presence has shaped this journey in countless ways. We dedicate this work to you and sincerely hope we have made you proud.

Abstract

The growing popularity of travel in the modern world coupled with inadequate authentication methods at border locations has contributed to the rise of border security incidents. The dominant solution of this problem is Biometric Border Control Systems. Previous research has primarily focused on unimodal biometric systems, which may reduce security events, but are still insufficient for high-security environments such as airports, especially with the continuous increase in travelers.

These types of systems are still vulnerable to spoofing attacks, intra-class variation and noisy data. This study aims to contribute to the reduction of border security incidents and address the scarcity of advanced systems in the field of multimodal biometric authentication with liveness detection.

The proposed methodology involves the identification of individuals and spoofing detection through the integration of two biometric traits: fingerprint and face. The pre-trained model DenseNet121 is used with custom classification tasks for identity verification and liveness detection. The two public datasets used for fingerprints and faces are FVC2002 and NUAA PI respectively. An additional created dataset was used to enhance fingerprint spoofing detection.

The presented results confirm the accurate identification of individuals along with reliable spoofing detection abilities. It was able to detect photo face spoofing attacks, software-generated fingerprint spoofing attacks and attacks done with the use of the spoofing materials tape, glue or clay.

Table of Contents

Acknowledgement ii
Abstract iii
Table of Contents iv
List of Figures viii
List of Tables xiii
List of Abbreviations xvi
Chapter 1 Introduction 1
1.1 Motivation 1
1.2 Problem Definition 3
1.3 Objectives 6
1.4 Time Plan 6
1.5 Document Organization 7
Chapter 2 Background 9
2.1 Description of the Project Field 9
2.2 Face and Fingerprint Biometrics 10
2.3 Literature Review 11
2.3.1 Multimodal Biometric Systems 11
2.3.2 Presentation Attack Detection 12
2.3.3 Biometric Security 15
2.4 Existing Similar Systems 17
2.4.1 SmartGate (Australia) 17
2.4.2 Global Entry (United States) 17
2.4.3 PARAFE (France) 18
2.4.4 e-Gate / Smart Gates (United Arab Emirates) 18
2.4.5 Comparison and Relevance to the Proposed System 19
Chapter 3 Analysis and Design 20
3.1 System Overview 20
3.1.1 System Architecture 20

3.1.2 System Users 22
3.2 System Analysis & Design 23
3.2.1 Use Case Diagram 23
3.2.2 Class Diagram 26
3.2.3 Sequence Diagram 31
3.2.4 Database Diagram 34
Chapter 4 Implementation and Testing 36
4.1 Datasets 36
4.1.1 NUAA PI 36
4.1.2 FVC2002 37
4.1.3 Created Dataset 38
4.2 Technologies and Methodologies 39
4.2.1 Software Tools and Libraries 39
4.2.2 Hardware Tools 42
4.2.3 Fingerprint Image Enhancement Techniques 46
4.2.4 AI Models and Configuration Settings 50
4.2.5 Face Detection and Capturing 53
4.2.6 Feature Extraction and Comparison 55
4.3 Data Protection and GDPR Compliance 55
4.3.1 Complying with Article 32 (1) (a) 56
4.3.2 Complying with Article 32 (2) 56
4.3.3 Password Policies and Multi-Factor Authentication 57
4.4 Experiments and Results 58
4.4.1 Fingerprint Acquisition 58
4.4.2 Fingerprint Image Enhancement 63
4.4.3 Fingerprint Identification and Spoofing AI Models 65
4.4.4 Face Detection and Capturing 75
4.4.5 Face Recognition and Spoofing Detection AI Model 78
4.4.6 Feature Extraction 79
4.4.7 Enrolment 79

4.4.8 Real-Time Testing and Threshold Determination 80
Chapter 5 User Manual 92
5.1 Traveler Flow: Normal Verification Flow 92
5.1.1 Home Screen 92
5.1.2 Demo Playback Page 92
5.1.3 Upload Your Passport 93
5.1.4 Fingerprint Scanning 95
5.1.5 Face Scanning 96
5.1.6 Identity Verification 97
5.2 Traveler Flow: Spoofing Attempts and Failed Verifications 97
5.2.1 Scenario 1: Passport Does Not Exist 97
5.2.2 Scenario 2: Blacklisted Passport 98
5.2.3 Scenario 3: Spoofed Fingerprint 98
5.2.4 Scenario 4: Fingerprint Identity Mismatch 100
5.2.5 Scenario 5: Misaligned Face 100
5.2.6 Scenario 6: Spoofed Face 101
5.2.7 Scenario 7: Face Identity Mismatch 103
5.3 Admin/ Officer Flow 104
5.3.1 Officer Login 104
5.3.2 Dashboard 107
5.3.3 Incident Management 110
5.3.4 User Enrolment 111
5.3.5 Officer Enrolment 116
5.3.6 Log Viewing 119
5.3.7 Officer Permissions 120
Chapter 6 Conclusion and Future Work 121
6.1 Conclusion 121
6.2 Future Work 122
References 123
Appendix A. Additional Tests and Experiments 131

Appendix B. Code Explanations and Outputs 133

List of Figures

Figure 1.1: Overall Project Timeline 7
Figure 2.1: Most Commonly used Biometrics in ABC Systems [5] 11
Figure 3.1: System Architecture 20
Figure 3.2: System Use Case Diagram 24
Figure 3.3: System Class Diagram 27
Figure 3.4: System Sequence Diagram 32
Figure 3.5: System Database Diagram 34
Figure 4.1: NUAA PI Sample Face Images 37
Figure 4.2: FVC2002 Sample Fingerprint Images 38
Figure 4.3: Created Dataset Sample Fingerprint Images 39
Figure 4.4: R307 Exterior Interface 42
Figure 4.5: R307 Fingerprint Sensor Digital Circuit Diagram 43
Figure 4.6: Wiring Connections of R307 to PC 43
Figure 4.7: R307 Data Package Format [47] 44
Figure 4.8: R307 GenImg Command Package Format [47] 45
Figure 4.9: R307 GenImg Response Package Format [47] 45
Figure 4.10: R307 UpImage Command Package Format [47] 46
Figure 4.11: R307 UpImage Response Package Format [47] 46
Figure 4.12: Ridges and Valleys in a Fingerprint 47
Figure 4.13: Application of Gaussian Blur in Different Intensities [49] 47
Figure 4.14: Fingerprint Image Before and After Applying a Threshold [50] 48
Figure 4.15: Erosion Applied on a Noisy Fingerprint Image [51] 48
Figure 4.16: Dilation Applied on a Noisy Fingerprint Image [51] 49
Figure 4.17: Opening Applied on a Noisy Fingerprint Image [51] 49
Figure 4.18: Closing Applied on a Noisy Fingerprint Image [51] 50
Figure 4.19: DenseNet Layer Connections 50

Figure 4.20: DenseNet-121 Architecture [23] 51
Figure 4.21: A General Representation of Training a Haar Classifier [59] 53
Figure 4.22: Dlib Indexes of the 68 Landmark Coordinates 54
Figure 4.23: Fingerprint Acquisition - Experiment 1 59
Figure 4.24: Fingerprint Acquisition - Experiment 2 59
Figure 4.25: Fingerprint Acquisition - Experiment 3 60
Figure 4.26: Fingerprint Acquisition - Experiment 4 61
Figure 4.27: Fingerprint Acquisition - Experiment 5 62
Figure 4.28: Fingerprint Acquisition - Experiment 5 (After Expansion) 62
Figure 4.29: Fingerprint Image Before and After Enhancement - Experiment 1 63
Figure 4.30: Fingerprint Image after Applying fingerprint_enhancer [45] - Experiment 2 64
Figure 4.31: Comparison of Enhancement Results 64
Figure 4.32: Face Acquisition - Experiment 1 75
Figure 4.33: Face Acquisition - Experiment 2 76
Figure 4.34: Face Acquisition - Experiment 3 77
Figure 4.35: Spoofed Fingerprint Image using Tape 81
Figure 4.36: Spoofed Fingerprint Image using Clay 81
Figure 4.37: Spoofed Fingerprint Image using Glue 82
Figure 4.38: Software-generated Fingerprint Image using SFinGe [46] 82
Figure 4.39: Real Fingerprint Image Captured using R307 83
Figure 4.40: Person E's Fingerprint (Left) and Person B's Fingerprint (Right) . 84
Figure 4.41: Person D's Fingerprint (Left) and Person C's Fingerprint (Right) . 84
Figure 4.42: Two Captured Fingerprint Images of Person A 85
Figure 4.43: Two Captured Fingerprint Images of Person B 86
Figure 4.44: ID Spoofed Face Image Detection (Left) and Captured Face Image (Right) 87

Figure 4.45: Uploaded Headshot (Spoofing) 87
Figure 4.46: 4*3 Headshot (Spoofed) 88
Figure 4.47: Genuine Face Image 88
Figure 4.48: Person F (Left) and Person E (Right) 89
Figure 4.49: Person D (Left) and Person F (Right) 89
Figure 4.50: Two Captured Face Images of Person D 90
Figure 4.51: Two Captured Face Images of Person G 90
Figure 5.1: Home Screen 92
Figure 5.2: Demo Video Player 93
Figure 5.3: Upload Passport 94
Figure 5.4: Passport Image Selection 94
Figure 5.5: Traveler Name Displayed after Passport Upload 94
Figure 5.6: Fingerprint Scanning 95
Figure 5.7: Fingerprint Scanning in Progress 95
Figure 5.8: Captured Fingerprint During Verification Process 96
Figure 5.9: Face Scanning (Aligned Properly) 96
Figure 5.10: Successful Traveler Verification 97
Figure 5.11: Uploading Unidentified Passport to the System Database 98
Figure 5.12: Blacklisted Passport Detected 98
Figure 5.13: Fingerprint Spoofing Attempt using a Clay Impression of a Fingerprint 99
Figure 5.14: Fingerprint Spoofing Attempt Detected 99
Figure 5.15: Fingerprint Identity Mismatch Detected 100
Figure 5.16: Misaligned Face Scanning 101
Figure 5.17: Face Spoofing Attempt using an ID 101
Figure 5.18: Face Spoofing Attempt Detected 102
Figure 5.19: Face Identity Mismatch 103
Figure 5.20: Face Identity Mismatch Detected 103

Figure 5.21: Officer Login Screen - Credentials Filled 104
Figure 5.22: Officer Login Screen - Fingerprint Scan 105
Figure 5.23: Officer Login Screen - Fingerprint Scanning in Progress 105
Figure 5.24: Officer Login Screen - Fingerprint Verified 105
Figure 5.25: Officer Login Screen - Face ID Scan 106
Figure 5.26: Officer Login Screen - Face ID Scanning in Progress 106
Figure 5.27: Officer Login Screen - Face ID Verified 107
Figure 5.28: Officer Dashboard 108
Figure 5.29: Officer Dashboard - Alerts 108
Figure 5.30: Officer Dashboard - Recent Visits 109
Figure 5.31: Officer Dashboard - Search Visitors 109
Figure 5.32: Officer Incident Management 110
Figure 5.33: Officer Incident Management - Removal of a User from Blacklist 110
Figure 5.34: Officer Incident Management - Removal of a User from Blacklist Successful 111
Figure 5.35: Officer Incident Management - Incident Status Updated 111
Figure 5.36: User Enrolment - Passport Upload 112
Figure 5.37: User Enrolment - Passport Image Selection 112
Figure 5.38: User Enrolment - Passport Upload Successful 113
Figure 5.39: User Enrolment - Fingerprint Scan 113
Figure 5.40: User Enrolment - Fingerprint Scanning in Progress 114
Figure 5.41: User Enrolment - Fingerprint Scan Complete 114
Figure 5.42: User Enrolment - Face Scan 115
Figure 5.43: User Enrolment - Face Scanning in Progress 115
Figure 5.44: User Enrolment - Face Scan Complete 115
Figure 5.45: Officer Enrolment Screen 116
Figure 5.46: Officer Enrolment Screen - Credentials Filled 117

Figure 5.47: Officer Enrolment - Fingerprint Scanning in Progress 117
Figure 5.48: Officer Enrolment - Fingerprint Scan Complete..... 118
Figure 5.49: Officer Enrolment - Face Scanning in Progress..... 118
Figure 5.50: Officer Enrolment - Face Scan Complete 119
Figure 5.51: Log Viewing..... 119
Figure A.1: Spoofed Fingerprint of Person A using Clay - Additional Experiments: 1 131
Figure A.2: Software-generated Fingerprint - Additional Experiments: 2..... 132
Figure B.1: Captured Image (Left) and Enhanced Image (Right)..... 138

List of Tables

Table 1.1: Biometric Traits used in Biometric Systems based on Existing Literature Review [10] 4
Table 2.1: Summary of Key Papers on Multimodal Biometric Systems 12
Table 2.2: Summary of Key Papers on Presentation Attack Detection 15
Table 2.3: Summary of Key Papers on Biometric Security 16
Table 4.1: NUAA PI Databases 36
Table 4.2: FVC2002 Database [38] 37
Table 4.3: Created Dataset Summary 38
Table 4.4: R307 Definition of Data Package [47] 44
Table 4.5: Fingerprint Identification and Spoofing Detection AI Model - Experiment 1 Configurations 65
Table 4.6: Fingerprint Identification and Spoofing Detection AI Model - Experiment 1 Results 66
Table 4.7: Fingerprint Identification and Spoofing Detection AI Model - Experiment 2 Configurations 67
Table 4.8: Fingerprint Identification and Spoofing Detection AI Model - Experiment 2 Results 68
Table 4.9: Fingerprint Identification and Spoofing Detection AI Model - Experiment 3 Configurations 70
Table 4.10: Fingerprint Identification and Spoofing Detection AI Model - Experiment 3 Results 70
Table 4.11: Fingerprint Identification AI Model - Experiment 4 Configurations 71
Table 4.12: Fingerprint Identification AI Model - Experiment 4 Results 72
Table 4.13: Fingerprint Spoofing Detection AI Model - Experiment 5 Configurations 72

Table 4.14: Fingerprint Spoofing Detection AI Model - Experiment 5 Results 73
Table 4.15: Fingerprint Spoofing Detection AI Model - Experiment 6 Configurations 74
Table 4.16: Fingerprint Spoofing Detection AI Model - Experiment 6 Results 74
Table 4.17: Face Identification and Spoofing Detection AI Model - Experiment 1 Configurations 78
Table 4.18: Face Identification and Spoofing Detection AI Model - Experiment 1 Results 78
Table 4.19: SoftMax Output and Prediction Result - Test 1, Experiment 1 (Real-time Testing) 81
Table 4.20: SoftMax Output and Prediction Result - Test 2, Experiment 1 (Real-time Testing) 82
Table 4.21: SoftMax Output and Prediction Result - Test 3, Experiment 1 (Real-time Testing) 82
Table 4.22: SoftMax Output and Prediction Result - Test 4, Experiment 1 (Real-time Testing) 83
Table 4.23: SoftMax Output and Prediction Result - Test 5, Experiment 1 (Real-time Testing) 83
Table 4.26: Matching Scores and System Decision - Test 1, Experiment 1 (Real-time Testing) 84
Table 4.25: Matching Scores and System Decision - Test 1, Experiment 2 (Real-time Testing) 85
Table 4.27: Matching Scores and System Decision - Test 2, Experiment 2 (Real-time Testing) 85
Table 4.28: Matching Scores and System Decision - Test 3, Experiment 2 (Real-time Testing) 86
Table 4.29: SoftMax Output and Prediction Result - Test 1, Experiment 1 (Real-time Testing) 87

Table 4.30: SoftMax Output and Prediction Result - Test 2, Experiment 1 (Real-time Testing) 87
Table 4.31: SoftMax Output and Prediction Result - Test 3, Experiment 1 (Real-time Testing) 88
Table 4.32: SoftMax Output and Prediction Result - Test 4, Experiment 1 (Real-time Testing) 88
Table 4.33: Matching Scores and System Decision - Test 1, Experiment 1 (Real-time Testing) 89
Table 4.34: Matching Scores and System Decision - Test 1, Experiment 2 (Real-time Testing) 90
Table 4.35: Matching Scores and System Decision - Test 2, Experiment 2 (Real-time Testing) 90
Table 4.36: Matching Scores and System Decision - Test 3, Experiment 2 (Real-time Testing) 91
Table A.1: SoftMax Output and Prediction Result - Additional Experiments: 1 131
Table A.2: Matching Scores and System Decision - Additional Experiments: 1 131
Table A.3: SoftMax Output and Prediction Result - Additional Experiments: 2 132
Table A.4: SoftMax Output and Prediction Result - Additional Experiments: 2 132

List of Abbreviations

ABC	Automated Border Control
ACE	Average Classification Error
ACER	Average Classification Error Rate
AES	Advanced Encryption Standard
AI	Artificial Intelligence
APC	Automated Passport Control
APCER	Attack Presentation Classification Error Rate
BPCER	Bona Fide Presentation Classification Error Rate
CN	Crossing Number
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DPbD	Data Protection by Design and by Default
DPIA	Data Protection Impact Assessments
EER	Equal Error Rate
EU	European Union
FAR	False Acceptance Rate
FPS	Frames per Second
FRR	False Rejection Rate
GDPR	General Data Protection Regulation
GPU	Graphics Processing Unit
HoG	Histogram of Oriented Gradients
IATA	International Air Transport Association
ICAO	International Civil Aviation Organization
ID	Identification
MFA	Multi-Factor Authentication
MTL	Multi-Task Learning

NN	Neural Network
OCR	Optical Character Recognition
PAD	Presentation Attack Detection
PCA	Principal Component Analysis
PIN	Personal Identification Number
ReLU	Rectified Linear Unit
ROC	Receiver-Operating Characteristic
ROI	Region of Interest
SGD	Stochastic Gradient Descent
SSD	Single-Shot Detector
SVM	Support Vector Machine
TTL	Transistor-Transistor Logic
UART	Universally Asynchronous Receiver/ Transmitter

Chapter 1 Introduction

1.1 Motivation

A “simple” border event, such as an illegal crossing, can have a severe impact on national security. Notable incidents include the Paris terrorist attacks (2015) where one of the perpetrators had illegally entered France via Greece. Also, the European Union (EU) migration crisis (2015) saw approximately one million irregular migrants arriving in the EU via the Eastern Mediterranean route [1].

Currently, the literature and research on cyber-threat landscape evolves around relevant infrastructures to the border control, such as ports, airports or defense systems [1]. While these areas are critical, the methods used for passenger control at borders often remain outdated. Typically, passenger controls are done manually by trained immigration officers who compare the passport and the physical appearance of the traveler, a process that is time-consuming, unreliable and prone to human error [2]. In other countries, this process is done by Automated Border Control (ABC) systems, but these systems often rely on unimodal biometric data, which can be insufficient for robust security.

This project aims to address these challenges by enhancing the performance of border control systems through the integration of multimodal biometrics. By integrating multiple biological features, higher security can be provided because these biological features have high uniqueness and unforgeability. Compared with traditional identity verification methods, multimodal biometric recognition technology does not require memorizing complex passwords or carrying identification documents, thus providing a more convenient user experience. The use of multimodal biometrics has shown a strong growth trend, increasing in usage rate from 6.27% in 2015 to 67.92% in 2021 [3]. This project will integrate two of the most accepted biometrics, namely face and fingerprint. The face is the primary biometric trait for e-Passports [4] because it offers many advantages. For

instance, it is accepted from a social perspective because it is used normally by people to identify each other. Furthermore, its capture is not intrusive, and human border guards are familiar with it. For this reason, many ABC systems rely on face recognition. In particular, 45% of the systems explored use face recognition, either alone or combined with other modalities. Some examples of systems that use face recognition are RAPID in Portugal, SmartGate in Australia, the Spanish ABC System, and Automated Passport Control (APC) in Canada and the United States [5].

Fingerprint recognition is currently the most mature biometric technology, with a great tradeoff between performance and social acceptance. Although fingerprints are optional in e-Passports [4], many ABC systems have employed them for traveler recognition. In fact, 56% of the ABC systems explored include fingerprint recognition systems, either alone or combined with other modalities. Some examples of systems that use fingerprint recognition are Global Entry in the United States, Passage Automated Rapid Flow at External Borders (PARAFE) in France, e-Channel in Hong Kong, and the Spanish ABC system [5]. The combination of fingerprint and face traits deliver reliable results in the time-critical environment due to many reasons such as processing time, sensor cost, capturing time and privacy [2]. For this reason, this project integrates fingerprint and face biometric traits.

Additionally, liveness detection methods are implemented to ensure that the biometric trait is not spoofed and comes from a living subject. By improving the reliability and efficiency of border control systems, this project seeks to ensure that borders are not only secure but also capable of handling the increasing flow of travelers.

1.2 Problem Definition

In an era of globalization and increasing displacement, border criminals and unauthorized immigrants have dramatically increased within the last few years due to the absence of proper authentication methods at border locations [2]. Illegal crossings or unauthorized events at border locations can lead to consequences such as national security threats when terrorists or smugglers are allowed through, potentially endangering public safety. Another significant consequence is the global spread of diseases, which can occur when unauthorized travelers are granted entry, potentially leading to a pandemic. Therefore, protection against border attacks is of utmost priority and a key challenge for all nations [1].

Photo Identifications (IDs) are widely used in the security domain. Passport officers are highly trained people who can compare photo ID and physical appearance at the borders. D. White *et al.* [6] discussed passport officers' errors in face matching. The researchers concluded that passport officers have poor performance at each level. In some cases, the authors identified that accurate results are not dictated by the number of experience years or extensive training. Finally, they brought forward, "Passport staff misses one in seven fake IDs" [6]. in the Australian context. This implies that there is a huge demand in the face recognition domain.

Last year, 2024, airlines connect nearly five billion people over 22,000 routes on 39 million flights [7]. With 18,614,160 flights in total to the end of this week (29th of June 2025), the average number of commercial flights per day is 102,275 [8]. Over the next 20 years, world passengers are expected to increase by 3.8% per year on average, resulting in over 4 billion additional passenger journeys in 2043 compared to 2023 [7]. This inevitably creates a more considerable risk of threat and need for improved security. One effective approach to enhance border

security is the use of biometric data for passenger controls. Biometrics is the study of measuring and analyzing specific physical or behavioral traits that are employed to identify a person. According to K. Jain *et al.* [9], any human physiological and/ or behavioral characteristic can be used as a biometric characteristic as long as it satisfies the requirements of distinctiveness, universality, permanence and collectability. Biometric traits include speech, facial features, iris, fingerprint, retina and signature for individual identification [10]. Most used biometric traits are the face, fingerprint and iris [5]. The comparison of biometric traits with respect to acceptability, performance, distinctiveness, universality, permanence, circumvention and measurability are shown in Table 1.1.

Table 1.1: Biometric Traits used in Biometric Systems based on Existing Literature Review [10]

Biometric Identifier	Universality	Uniqueness	Permanence	Measurability	Performance	Acceptability	Circumvention
Face	3	1	2	3	1	3	3
Fingerprint	2	3	3	2	3	2	2
Ear	3	2	3	2	2	3	2
Gait	2	1	1	3	1	3	2
Hand Geometry	2	2	2	3	2	2	2
Hand Vein	2	2	2	2	2	2	1
Iris	3	3	3	2	2	1	1
Retina	3	3	2	1	3	1	1
Voice	2	1	1	2	1	3	3
DNA	3	3	3	1	1	1	1
Signature	1	1	1	3	1	3	3
Keystroke	1	1	1	2	1	2	2

3: Highest rating, 2: Intermediate rating, 1: Lowest rating

The use of biometric technologies in automated checks is of paramount importance, as it improves the efficiency and effectiveness of the screening process and the security of the travelers. Based on the Global Passenger Survey conducted in 2024 by the International Air Transport Association (IATA) [11], 73% of the 10,000 travelers surveyed across 200 countries expressed a desire to use their biometric data. Biometric-based authentication systems have been successfully employed for three decades at the University of Georgia and for over a decade at the airports in San Francisco and Walt Disney World, with tens of thousands of daily users. The use of biometric technologies in many security applications has grown globally because of its major benefits with regard to authentication rate, universality, and security [10]. Unimodal fingerprint

biometric systems safeguard authentication information through the analysis of characteristic sequences and, face challenges such as vulnerability to spoof attacks, inter-class similarity, intra-class variation, non-universality, and noisy data. These challenges are addressed by multimodal biometric systems, in which various biometric sources compensate for each other's limitations [10]. To decide among the different biometric traits available for person recognition, including the most common face, fingerprint, and iris, but also signature, hand, or voice, the International Civil Aviation Organization (ICAO) [4] considered several features: global interoperability, uniformity, technical reliability, practicality, and durability. For global interoperability of the recognition systems, ICAO specifications mandate the use of facial data, and optionally, of fingerprint and iris data [5].

A common threat faced by biometric systems is spoofing, where attackers use fake fingerprints or facial masks to bypass security. To address this, this project incorporates liveness detection techniques, ensuring that only genuine biometric traits are accepted, preventing spoofing attempts [10]. In addition, the project uses encrypted biometric databases to safeguard biometric data from database breaches, which are increasingly common at border checkpoints. Encryption ensures that stored data is protected against unauthorized access or alteration, mitigating risks of identity theft or tampering [12].

Lastly, the proposed system includes measures to comply with General Data Protection Regulation (GDPR) standards [13], ensuring that users' biometric data is handled securely, further enhancing the trustworthiness of the system.

1.3 Objectives

The main objectives of this project are:

1. Accuracy and Reliability
    - Improve accuracy by minimizing false positives and false negatives through introducing multimodality.
2. Liveness Detection
    - Implement liveness detection techniques to prevent spoofing and presentation attacks.
3. Security Measures
    - Implement encryption algorithms, such as Advanced Encryption Standard (AES), to prevent stored biometric data from unauthorized access.
4. Real Time Processing
    - Optimize the system for real-time processing to handle a high volume of travelers without delays.
5. Privacy and Compliance
    - Develop the system to comply with the GDPR standard to safeguard individuals' privacy and ensure data protection.

1.4 Time Plan

The time plan manifests the main stages of the project and their corresponding time span. It starts with Literature Review and Survey, followed by Project Design and Architecture. Project Implementation and Testing comes next. Finally, documentation is an ongoing task from after the Survey phase onward. The time plan is outlined in Figure 1.1.

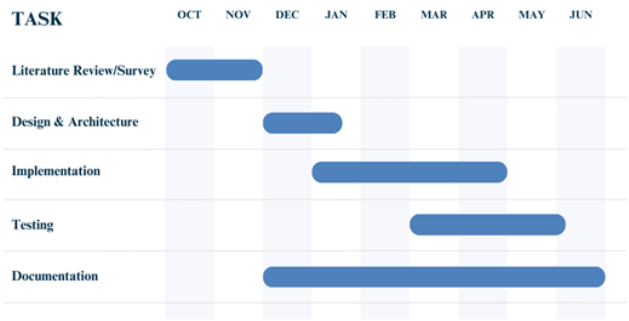

Figure 1.1: Overall Project Timeline

1.5 Document Organization

This documentation is structured as follows:

- Chapter 1: The current chapter introduced the project's motivation and problems it is targeting to solve.
- Chapter 2: Includes a description of the fields of the project, along with a comprehensive review of related works in each of relevant fields to this project. It also mentions several similar systems.
- Chapter 3: This chapter everything related to the system's analysis and design. Several diagrams are included in this chapter to highlight the system's architecture, functionalities, classes and database tables. Additionally, it states the users this system is intended for and their required knowledge to be able to take full advantage of it.
- Chapter 4: The core of this document; it starts by explaining the technologies and methods adopted during this project, then shows the experiments conducted in each phase of the project. This serves the purpose of showing the progress made between each experiment and the next until the final solution is reached. It also shows the evaluation of the system on real-life test cases.
- Chapter 5: A complete walkthrough of the system for all intended users. Screenshots of each step are shown for clarity and ease of understanding.

- Chapter 6: This concludes the document and recommends future work to better the project and achieve superior results.
- Appendix A: Includes additional tests and experiments.
- Appendix B: Contains the source code and functions of the system's main modules with a detailed explanation for each. It also includes a reference to the project's GitHub Repository.

Chapter 2 Background

2.1 Description of the Project Field

The field of this project centers on multimodal biometric systems applied to border control. Biometric systems leverage unique physiological or behavioral characteristics, such as facial features, fingerprints, iris patterns, voice, or signatures, to authenticate individuals [10]. Unlike traditional authentication methods (e.g., passwords or physical IDs), biometrics offer a higher degree of security due to their inherent uniqueness and resistance to forgery. The project specifically focuses on integrating multiple biometric modalities, namely face and fingerprint recognition, to enhance the security, efficiency, and reliability of ABC systems.

Border control systems are essential for ensuring national security, managing immigration, and facilitating the safe movement of travelers across international boundaries. With globalization and increasing travel volumes, projected to reach over 4 billion additional passenger journeys by 2043 compared to 2023 [7], the demand for robust, efficient, and secure border control solutions has intensified. Traditional manual checks, reliant on trained passport officers comparing photo IDs with physical appearances, are prone to human error and inefficiencies. Studies, such as those by D. White *et al.* [6], have highlighted significant error rates in manual face-matching by passport officers, with up to one in seven fake IDs missed in certain contexts. These limitations underscore the need for automated, technology-driven solutions.

Multimodal biometric systems address the shortcomings of unimodal systems, which rely on a single biometric trait (e.g., face or fingerprint alone) and are vulnerable to challenges such as spoofing attacks, intra-class variations, non-universality, and noisy data. By combining multiple biometric traits, multimodal systems compensate for individual modality limitations, achieving higher

accuracy and robustness [10]. For instance, face recognition is socially accepted and non-intrusive, making it the primary biometric for e-Passports, as mentioned by ICAO [4]. Fingerprint recognition, on the other hand, is mature, cost-effective, and widely deployed, offering a strong balance between performance and acceptability. The integration of these traits aligns with ICAO standards for global interoperability and enhances security by leveraging their complementary strengths [4].

The project also encompasses liveness detection where attackers use fake biometric samples (e.g., photographs or artificial fingerprints) to bypass security. Liveness detection ensures that biometric inputs come from a living subject, enhancing system trustworthiness. Additionally, the field involves data security and privacy, particularly in the context of biometric databases, which are susceptible to breaches. Encryption techniques, such as AES, and compliance with regulations like the GDPR [13] are integral to safeguarding sensitive biometric data against unauthorized access or tampering [12].

The system operates within this dynamic field, developing a multimodal biometric system that integrates face and fingerprint recognition with liveness detection and robust data security measures. By addressing the limitations of manual and unimodal systems, the project seeks to contribute to the advancement of secure, efficient, and automated border control processes, aligning with global trends toward increased adoption of multimodal biometrics.

2.2 Face and Fingerprint Biometrics

The selection of face and fingerprint biometrics is based on the research by Labati *et al.* [5] who show that fingerprint (56%) and face (40%) dominate ABC systems, justifying their use over iris (3.9%) due to prevalence and performance.

Most used Biometric Traits in ABC Systems according to [5] are shown in Figure 2.1.

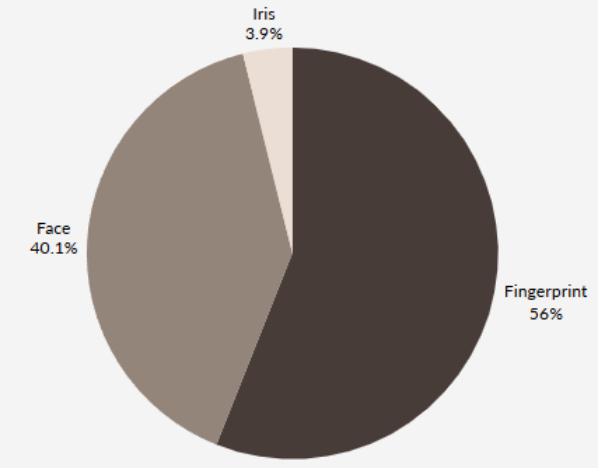

Figure 2.1: Most Commonly used Biometrics in ABC Systems [5]

2.3 Literature Review

2.3.1 Multimodal Biometric Systems

Thenuwara *et al.* [2] use deep learning and machine learning (Convolutional Neural Networks (CNN), Neural Networks (NN), SoftMax, Support Vector Machine (SVM)) on BANCA [14] and SDUMLA-HMT [15] datasets. Their work improves traveler identification efficiency with face and ear modalities, achieving real-time processing and high accuracy via a multi-agent system. Challenges include agent interoperability, real-time processing demands, and data privacy, suggesting future work on efficient algorithms and security enhancements.

Tomar and Singh [16] propose a hybrid cascaded-fusion framework using fingerprint Crossing Number (C.N) and face Principal Component Analysis (PCA) on a self-generated dataset of 1000 images. The work achieved 99.50% accuracy with max fusion at a 0.60 threshold, reducing verification time. Limitations include reliance on controlled datasets and potential image quality issues, with future plans to incorporate user-specific traits.

Sumalatha *et al.* [10] review fingerprint-based systems, noting unimodal limitations (e.g., spoofing) and multimodal advantages (e.g., 87.6% accuracy with iris fusion). This review emphasizes fusion techniques and liveness detection for border control, highlighting security threats and the need for template protection.

Table 2.1 outlines the methods and results of the main papers that use multimodal biometric systems.

Table 2.1: Summary of Key Papers on Multimodal Biometric Systems

Paper	Year	Methods used	Datasets	Results
[2]	2022	Deep learning and machine learning based methods for Fingerprint and face recognition at border control	1. (NIST BSSR1) 2. SDUMLA-HMT 3. BANCA 4. PRIVATE	(CNN and SoftMax) is Better than (CNN and SVM)
[16]	2021	1. C.N 2. PCA 3. Score level fusion (max level).	Self-generated	• 0.6 FAR • 0.4 FRR • 99.5% Accuracy

2.3.2 Presentation Attack Detection

Li and Ramachandra [17] review deep learning methods for detecting fingerprint presentation attacks, highlighting their advantage over traditional techniques by learning complex patterns. Their review categorizes approaches into contact-based, contactless, and smartphone-based, and emphasizes the use of diverse spoof materials like silicone and gelatin. Public datasets like LivDet [18] are discussed for training and benchmarking.

Yuan *et al.* [19] propose a multimodal CNN with weighted feature fusion for fingerprint and face liveness detection, improving accuracy by combining features from multiple models. Tested on datasets like LivDet [18] and NUAA [20], it outperforms traditional methods in various scenarios, with lower error

rates (Average Classification Error (ACE), False Acceptance Rate (FAR), False Rejection Rate (FRR)) and strong generalization across materials and sensors.

In [21], Jawade *et al.* introduce a motion-based method for fingerprint presentation attack detection by capturing gestures like sliding and twisting to reveal differences between real and fake fingerprints. Using a dataset of 3680 videos recorded at 30 Frames Per Second (FPS), a spatiotemporal multi-modal network combining ridge and minutiae features achieved an F1 score of 83.84%, outperforming static methods. The approach enhances security and generalization without extra hardware but requires higher computational resources.

Purnapatra *et al.* [22] address spoofing in noncontact fingerprint systems using smartphone cameras by introducing a dataset of 23,000+ images with various spoof materials. DenseNet-121 [23] and NasNetMobile were tested, with DenseNet-121 achieving low error rates (Attack Presentation Classification Error Rate (APCER) of 0.14% and Bona Fide Presentation Classification Error Rate (BPCER) of 0.18%) but struggling with printed spoofs. The results highlight the effectiveness of CNNs and the need for more robust models against unseen attacks.

A CNN-based approach for detecting spoof fingerprints in contact-based systems using the ATVS-FFp DB [24] dataset was presented by Alsubhi and Alsufyani [25]. The model achieved high accuracy—99.89% in training and 97.04% in testing—by focusing on morphological and textural fingerprint features. It proves to be efficient, cost-effective, and easily integrable into existing systems. However, its reliance on a single dataset and potential vulnerability to unseen attacks highlight the need for broader testing and improved generalization.

Amtefe *et al.* [26] review fingerprint liveness detection methods, highlighting a shift from handcrafted to deep learning-based features. The review emphasizes the challenge of generalizing to unseen spoof materials and suggests open-set training as a solution. While progress has been made, the paper calls for further innovation to improve spoof detection in real-world scenarios.

Sharma and Selwal [27] cover hardware, software, and challenge-response presentation attack detection (PAD) on custom datasets, addressing photo, video, and mask attacks. They highlight limited generalization and call for diverse datasets.

Ortega *et al.* [28] introduce FlyPAD, a dynamic PAD system for border control that detects attacks as travelers approach the e-gate. It handles five attack types using face tracking, verification, and SVM-based PAD adapted to distance ranges. Evaluated on two datasets, it achieved up to 76.2% accuracy, with the lowest Average Classification Error Rate (ACER) of 0.016 at 1–2 meters. The system is effective, with future plans to support more attack types and spatio-temporal analysis.

Shaheed *et al.* [29] review PAD across fingerprint, iris, and face on LivDet [18] and others, noting gaps in generalization and data privacy. They suggest synthetic data and transfer learning.

Table 2.2 summarizes the key papers on Presentation Attack Detection.

Table 2.2: Summary of Key Papers on Presentation Attack Detection

Paper	Year	Methods used	Datasets	Results
[19]	2022	• Multimodal convolutional neural networks (MCNN) • Weighted feature fusion • Region of Interest (ROI) extraction • Transfer learning • Fine-tuning	1. LivDet 2011 2. LivDet 2013 3. LivDet 2015 4. NUAA face dataset	Superior performance in both FLD and FaLD, with low ACE values (below 1% in some cases) and strong generalization
[28]	2020	• Machine Learning • Hardware-based PAD algorithms	1. FRAV-ABC-OnTheFly 2. FRAV-ABC-RB-OnTheFly	Real border Scenario: • 0.016 ACER • 76.2% Accuracy
[17]	2023	• CNNs • Focuses on deep learning-based methods for Fingerprint PAD	1. LivDet	Deep learning-based methods have shown improvement in detecting presentation attacks

2.3.3 Biometric Security

Dhakal [12] highlights the benefits of multi-biometric systems over unimodal ones, addressing security threats like spoofing and replay attacks. His work reviews protection methods such as biometric cryptosystems and multi-factor authentication, and outlines applications in law enforcement, military, and commercial sectors.

Tran *et al.* [30] review privacy-preserving biometric authentication methods, proposing a taxonomy of techniques like non-invertible transformations, biometric key generation, information hiding, and cryptographic protocols. This

study emphasizes the need for secure, irreversible, and unlinkable biometric protection as a standard.

Abohamra and Yayilgan [31] focus on privacy and data protection best practices under the EU's GDPR [13], especially in processing biometric data. This work guides data controllers, drawing on lessons from the SMILE project, and explains that biometric data is generally prohibited but allowed under specific exemptions like consent, vital interests, or public interest. It emphasizes user rights, Data Protection by Design and by Default (DPbD), and the need for Data Protection Impact Assessments (DPIAs) for large-scale processing.

The most relevant papers on biometric security are outlined in Table 2.3.

Table 2.3: Summary of Key Papers on Biometric Security

Paper	Year	Methods used	Datasets	Results
[12]	2021	• Multi-biometric systems including fingerprint, facial recognition, iris scanning, and voice recognition. • Machine learning	Not mentioned	Multiple biometric traits results have higher security and lower error rates compared to single biometric systems
[30]	2021	• Non-invertible Transformation • Direct Biometrics Key Generation • Information Hiding Techniques Protocol-based Protection	1. FVC2002 DB1-4 2. FVC2004 DB1-3 3. FERET 4. CMU-PIE 5. FRGC 6. FEI 7. Extended Yale	• Hashing technique achieved 0% as its best EER. • The fuzzy extractor algorithm yielded EER of 4.5%.
[31]	2021	• GDPR compliance • DPbD • DPIA	-	Guides data controllers under GDPR

2.4 Existing Similar Systems

2.4.1 SmartGate (Australia)

SmartGate [32] is an automated self-service border control system operated by the Australian Border Force (ABF) at ten Australian international airports, including Sydney, Melbourne, and Brisbane. Launched publicly in 2007, SmartGate uses face recognition to verify a traveler's identity by comparing a live facial image captured at the gate with the biometric data stored in an e-Passport chip, according to ICAO 2021 standards [4]. The system incorporates liveness detection, such as motion analysis, to prevent spoofing attacks [33].

Similarities to the Proposed System

Like the proposed system, SmartGate prioritizes face recognition for its social acceptance and non-intrusive nature. It also incorporates liveness detection to prevent spoofing, aligning with the project's objective. However, SmartGate is unimodal, relying solely on face recognition, whereas the proposed system integrates fingerprint recognition for enhanced accuracy and reliability.

2.4.2 Global Entry (United States)

Global Entry [34], administered by U.S. Customs and Border Protection (CBP), is a trusted traveler program that automates border control for pre-approved travelers at major U.S. airports and select international locations. Introduced in 2008, it uses fingerprint recognition during enrollment and face recognition at entry kiosks. This shift toward multimodality is ongoing, though not fully implemented across all locations.

Similarities to the Proposed System

Global Entry's use of face and fingerprint biometrics mirrors the proposed system's multimodal approach, aiming to minimize false positives and false negatives. Its focus on real-time processing and data security aligns with the

project's objectives, though the proposed system emphasizes GDPR compliance, which Global Entry does not explicitly address.

2.4.3 PARAFE (France)

PARAFE (Passage Automatisé Rapide aux Frontières Extérieures) [35] is an ABC system deployed at French airports, such as Paris-Charles de Gaulle, to expedite border control for European Union citizens and select third-country nationals with e-Passports. Operational since 2009, PARAFE primarily uses fingerprint recognition, with recent upgrades incorporating face recognition to comply with ICAO standards [4].

Similarities to the Proposed System

PARAFE's integration of face and fingerprint recognition and liveness detection closely resembles the proposed system's design. Its emphasis on real-time processing and encrypted data supports the project's goals, though the proposed system adds GDPR compliance for enhanced privacy.

2.4.4 e-Gate / Smart Gates (United Arab Emirates)

e-Gate [36], operated by the UAE Ministry of Interior at airports like Dubai International, automates border control for UAE residents, Gulf Cooperation Council nationals, and eligible visitors. Introduced in 2002 and enhanced with biometric upgrades, e-Gate uses face recognition and fingerprint recognition, making it a multimodal system that aligns closely with the proposed project.

Similarities to the Proposed System

e-Gate's use of face and fingerprint biometrics, liveness detection, and data security measures directly parallels the proposed system. Its focus on real-time processing in a high-traffic environment supports the project's objective of

handling high traveler volumes. The proposed system extends this by incorporating GDPR compliance for privacy and compliance.

2.4.5 Comparison and Relevance to the Proposed System

All the mentioned existing systems share core features with the proposed Multimodal Biometric Border Control System, including the use of face and fingerprint biometrics, liveness detection to counter spoofing, and encrypted data to ensure security. They demonstrate reduced processing times, improved accuracy, and enhanced user experience. However, the proposed system advances beyond these by integrating multimodal biometrics consistently (unlike SmartGate's unimodal focus), emphasizing GDPR compliance for privacy, and optimizing for real-time processing to handle high volume of travelers without delays. By building on the strengths of these systems, the project aims to address national security threats, such as illegal crossings and terrorism, while improving efficiency and reliability in border control.

Chapter 3 Analysis and Design

3.1 System Overview

3.1.1 System Architecture

Figure 3.1 show the overall system architecture. It is divided into three tiers: Data Capture, Processing and Decision, and Data layer. Each of these tiers is discussed in this chapter.

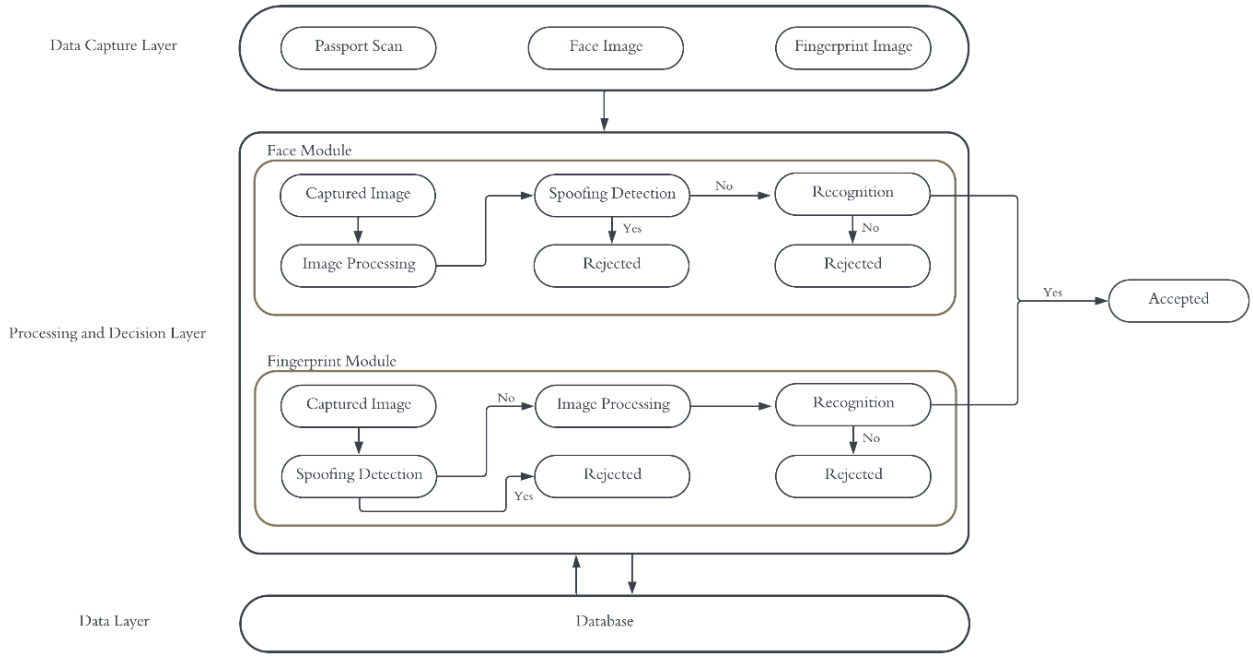

Figure 3.1: System Architecture

Data Capture Layer:

Acquisition:

a. Passport scan using Optical Character Recognition (OCR).
b. The capturing of the fingerprint image using a fingerprint sensor.
c. The capturing of the face image using a webcam.

Processing and Decision Layer:

1- Face Module:

a. **Captured Image**: The face image as it is, captured by a webcam.
b. **Image Processing:** Cropping the captured face image.

c. **Spoofing Detection:** Processed image is passed to a spoofing detection model whose function is to extract the features and classify the image as either real or fake.
    1. If the image is classified as a Fake, the user is rejected.
    2. Else, the Recognition program is executed.
d. **Recognition**: Processed image is passed to a recognition model whose function is to extract the features and compare them to those of a specific user ID (Passport Number) stored in the database.
    1. If the model matches the features extracted to those associated with this User ID, the system proceeds to fingerprint recognition for further confirmation. The traveler is allowed through only if both modalities match the same user.
    2. Else, the traveler is rejected.
e. **Enrolment**: Personal information from the user's passport is acquired using an OCR scan and biometric features are extracted from the processed image and saved in the database as a new user.

2- Fingerprint Module:

a. **Captured Image**: The fingerprint image as it is, captured by the fingerprint sensor.
b. **Spoofing Detection**: Captured image is passed to a spoofing detection model whose function is to extract the features and classify the image as either real or fake.
    1. If the image is classified as a Fake, the user is rejected.
    2. Else, image processing is applied to the image before passing it to the Recognition program.
c. **Image Processing**: Enhancing the captured image to highlight fingerprint features.

d. **Recognition**: Processed image is passed to a recognition model whose function is to extract the features and compare them to those of a specific user ID (Passport Number) stored in the database.
    1. If the model matches the features extracted to those associated with this User ID, and the fingerprint also matched, the traveler is allowed through.
    2. Else, the traveler is rejected.
e. **Enrolment**: Personal information from the user's passport is acquired using an OCR scan and biometric features are extracted from the processed image and saved in the database as a new user.

Data Layer:

Database: These are the created tables used in the system: DB_Keys, Blacklist, Travelers, Officers, Roles, Permissions, Role_Permissions, Logs, Incidents, Visits, each of which are described in Section 3.2.4.

3.1.2 System Users

A. Intended Users:

The users this system was designed to serve:

1. **Border Control Officers at Air, Land, and Sea Entry Points**: Officials responsible for verifying travelers' identities at airports, land crossings, and seaports will use the system for secure and efficient processing.
2. **Travelers Crossing Borders**: The system is designed to meet the needs of modern travelers who, according to the **IATA Global Passenger Survey 2024** [11], prioritize speed and convenience during identity checks and border processing. Additionally, 73% of respondents indicated a willingness to use **biometric identification technologies** instead of passports and boarding passes to streamline their journey.

B. User Characteristics

1. **Border Officers** will need:
    i. Basic understanding of principles followed in the GDPR [13].
    ii. Knowledge of border security protocols related to the handling of threats detected in the system, such as spoofing attempts or the entry of blacklisted individuals.
2. **Travelers** do not require experience as the system has a short instructional video showing how to use the system.

3.2 System Analysis & Design

3.2.1 Use Case Diagram

A use case diagram illustrates the interactions between external actors and the system's core functionalities. An actor represents a role played by an external entity that interacts with the system. Actors can be people, organizations or other systems. In this project, the actors are the Traveler and the Officer as mentioned previously in Section 3.1.2. A use case represents a feature or function of the system. This project's use case diagram can be seen in Figure 3.2.

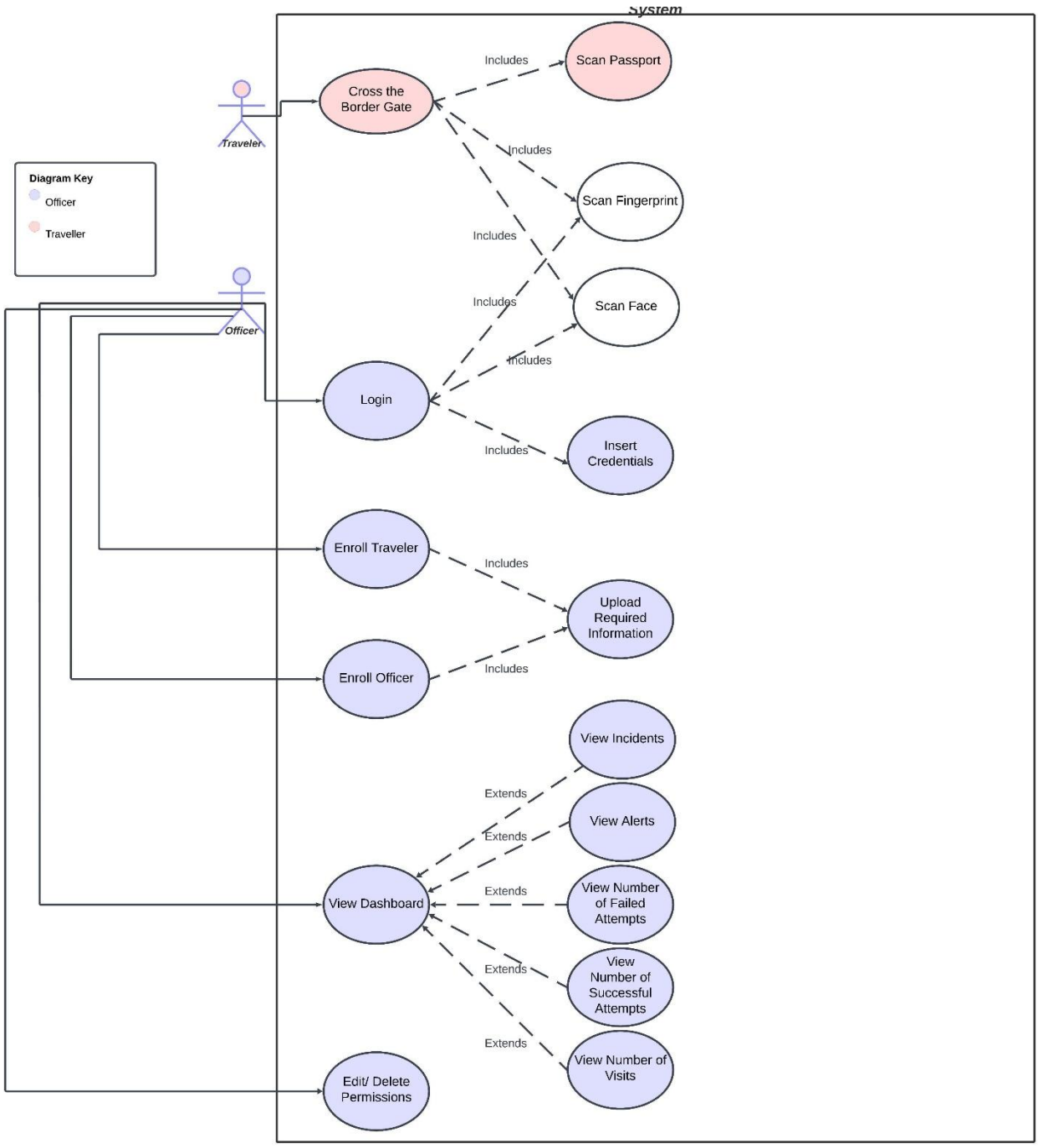

Figure 3.2: System Use Case Diagram

Traveler

The traveler interacts with the system for biometric recognition and authentication during border crossing.

1. Use Case 1: Cross the border gate

The “Cross Border Gate” use case **includes** several mandatory use cases, meaning that for the user to successfully cross the border gate, the included sub-processes must be executed as part of the overall process. The sub-processes are:

- Scan Passport: Travelers must scan their passport.
- Scan Fingerprint: Travelers must scan their fingerprint for verification.
- Scan Face: Travelers must scan their face for verification.

Officer

The officer interacts with the system to perform tasks essential for maintaining border control security.

1. Use Case 1: Login

“Login” feature **includes** similar sub-processes as “Cross the border gate” for travelers. The difference is that officers do not need to scan their passports, instead they login using their biometrics and credentials. The sub-processes are:

- Insert Credentials: Officers must login using a username and password that follow common password guidelines, see Section 4.3.3.
- Scan Fingerprint: Officers must scan their fingerprint for verification.
- Scan Face: Officers must scan their face for verification.

2. Use Case 2 and 3: Enroll Officers, Enroll Travelers

Officers with appropriate permissions can enroll other officers and travelers into the system’s database. This process **includes** another use case. Enrolment includes uploading the required information.

3. Use Case 4: View Dashboard

The dashboard is an admin functionality that is **extended by** several sub-processes. The extend relationship indicates that the behavior of the extending

use case may be inserted into the base use case at runtime, but only when a specific condition is met. The extending use cases are:

- View Incidents: Incidents are a list of flagged attempts.
- View Alerts: Alerts include spoofing attempts, identity mismatches and blacklisted individuals attempting to cross the border.
- View Number of Failed Attempts: Total number of failed attempts by travelers.
- View Number of Successful Attempts: Total number of successful attempts by travelers.
- View Number of Visits: Total number of border crosses.

4. Use Case 5: Edit/ Delete Permissions:

   Officers can edit or delete permissions of less privileged officers.

3.2.2 Class Diagram

The class diagram is a static structural diagram that represents the classes, their attributes, methods, and the relationships between them. A plus (+) sign indicates that this attribute or method is public, or visible to other classes, a minus (-) sign indicates that it is private, or not visible to other classes, and a hashtag (#) indicates that the class and any of its descendants have access to the attribute. The main classes of this project, along with their attributes and methods are shown in Figure 3.3 and mentioned below:

1. **Traveler**

A traveler is an individual attempting to cross the border.

Attributes:

+ Passport#	+ Name	+ Date of Birth
+ Nationality	+ Gender	+ Date of Expiry
+ Fingerprint Vector	+ Face Vector	

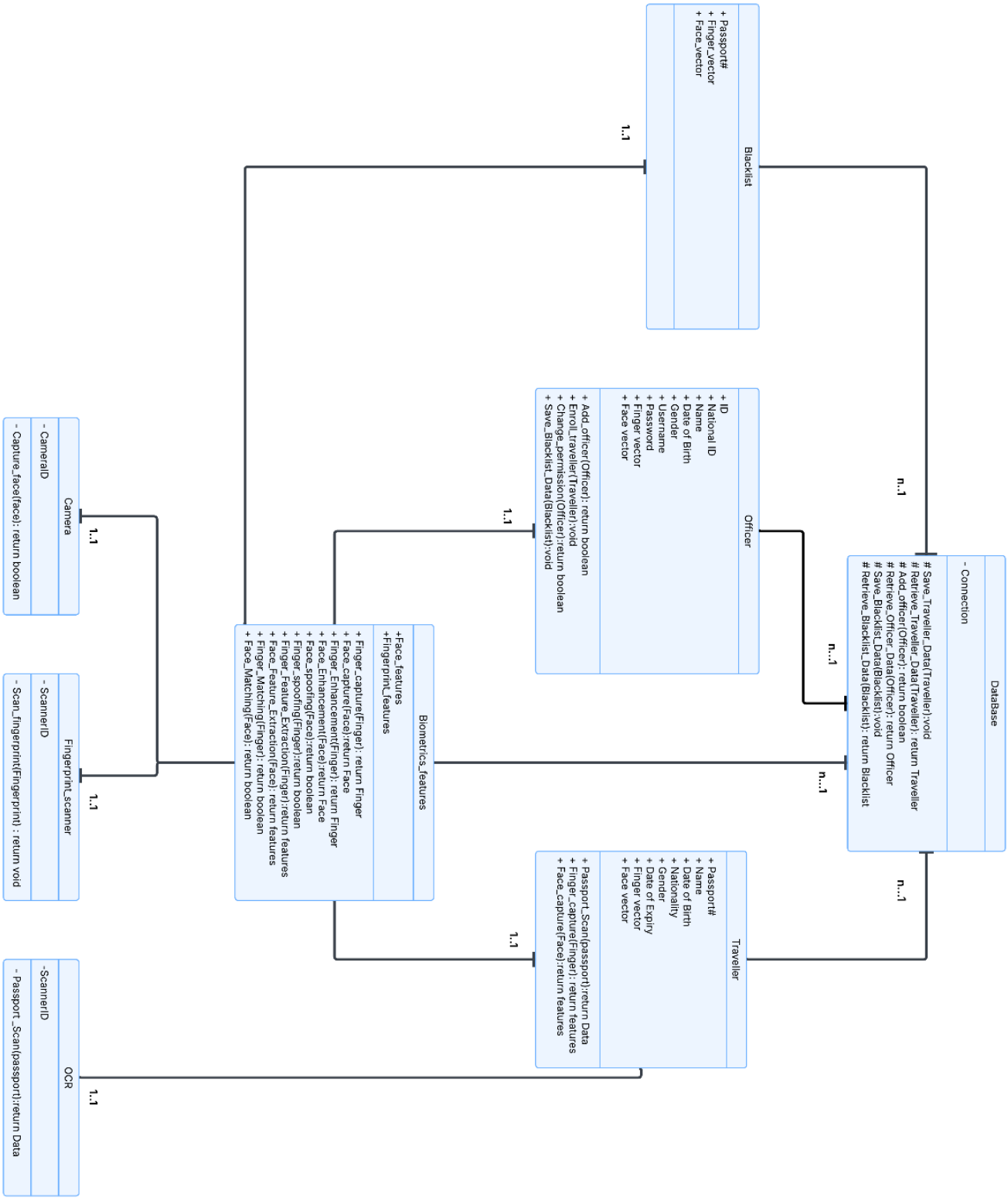

Figure 3.3: System Class Diagram

Methods:

+ Passport_Scan(passport): return Data
+ Finger_Capture(finger): return Features
+ Face_Capture(face): return Features

2. **Officer**

Officers are authorized personnel who manage system operations and traveler/ officer enrolment.

Attributes:

+ ID	+ National ID	+ Name
+ Date of Birth	+ Gender	+ Username
+ Password	+ Fingerprint Vector	+ Face Vector

Methods:

+ Add_Officer(Officer): return Boolean
+ Enrol_Traveler(Traveler): void
+ Change_permission(Officer): return Boolean
+ Save_Blacklist_Data(Blacklist): void

3. **Blacklist**

This represents individuals that are blacklisted and not allowed through the border.

Attributes:

+ Passport#	+ Fingerprint Vector	+ Face Vector

4. **Biometric Features**

This class provides core biometric processing functionalities for both fingerprint and face data.

Attributes:

+ Fingerprint_Features	+ Face_Features

Methods:

+ Finger_Capture(finger): return Finger
+ Face_Capture(face): return Face
+ Finger_Enhancement(finger): return Finger
+ Face_Enhancement(face): return Face
+ Finger_Spoofing(finger): return Boolean
+ Face_Spoofing(face): return Boolean
+ Finger_Feature_Extraction(finger): return Features
+ Face_Feature_Extraction(face): return Features
+ Fingerprint_Matching(finger): return Boolean
+ Face_Matching(face): return Boolean

5. **OCR**

The extraction of textual data from scanned passport documents using OCR.

Attributes:

+ ScannerID

Methods:

+ Passport_Scan(passport): return Data

6. **Fingerprint Scanner**

The acquisition of fingerprint images from a scanner are handled in this class.

Attributes:

+ ScannerID

Methods:

+ Scan_Fingerprint(fingerprint)

7. **Camera Scanner**

The acquisition of facial images using a webcam.

Attributes:

+ CameraID

Methods:

+ Capture_Face(face)

8. Database

Data storage of travelers, officers and blacklist information.

Attributes:

+ Connection

Methods:

Save_Traveler_Data(traveler): void
Retrieve_Traveler_Data(traveler): return Traveler
Add_Officer(officer): return Boolean
Retrieve_Officer_Data(officer): return Officer
Save_Blacklist_Data(blacklist): void
Retrieve_Blacklist_Data(blacklist): return Blacklist

3.2.3 Sequence Diagram

The sequence diagram emphasizes the order of interactions and the flow of control between system components or actors. Figure 3.4 is a sequence diagram of the process that occurs when a traveler attempts to cross the border. The sequence of steps is as follows:

1. The traveler enters the gate.
2. Traveler scans his/her passport with OCR.
3. Passport data is forwarded to the controller.
4. The controller sends the passport data to the database where whether he/she exists in the system is determined.
5. Database then determines whether the traveler is blacklisted.
6. Results from the database are returned to the controller.
7. The controller then forwards the results to the gate.
8. The fingerprint of the user is then captured and forwarded to the Fingerprint Controller along with the user's data.
9. The Fingerprint Controller checks if the fingerprint is spoofed, then it enhances the image and extracts the fingerprint's features.
10. The Controller then requests the stored fingerprint feature vector of the user attempting to cross the border using his/her passport number.
11. The database returns the feature vector to the Fingerprint Controller.
12. The retrieved features and extracted features are compared and the result is forwarded to the gate.

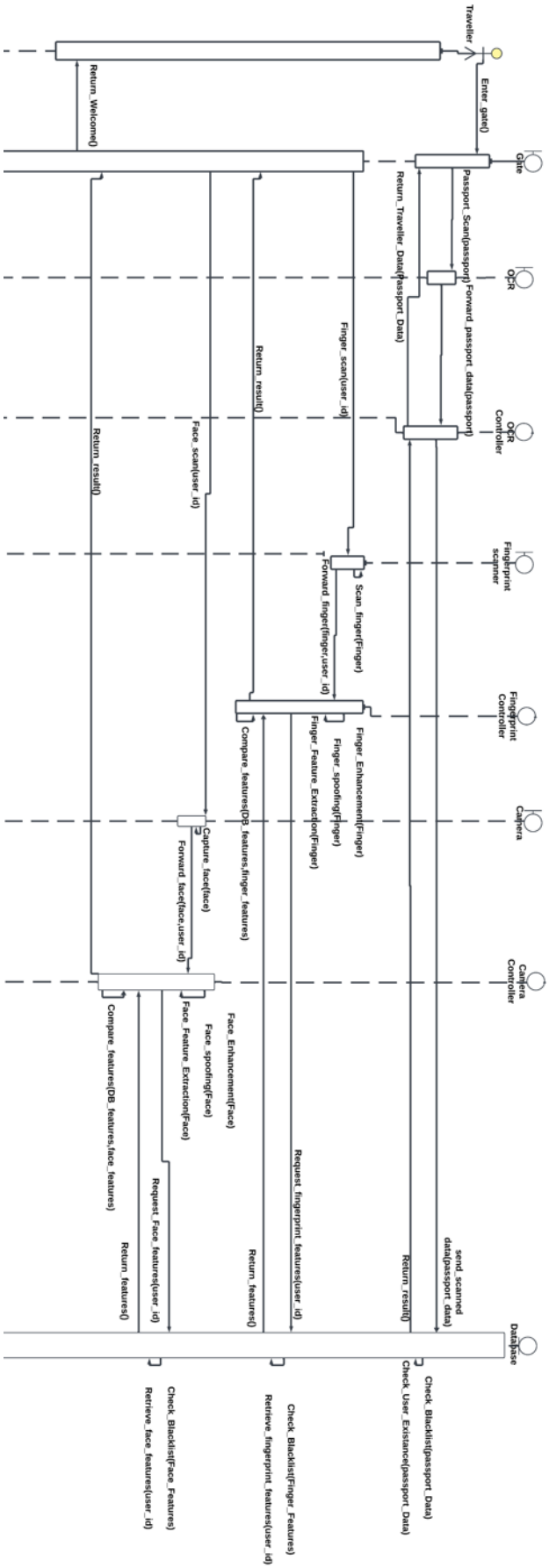

Figure 3.4: System Sequence Diagram

13. The face of the user is then captured using the Camera and forwarded to the Camera Controller along with the user's data.
14. The Camera Controller checks if the face image is spoofed and extracts the feature vector of the user's face.
15. The Controller then requests the stored face feature vector of the user attempting to cross the border using his/her passport number.
16. The database returns the feature vector to the Camera Controller.
17. The retrieved features and extracted features are compared and the result is forwarded to the gate.
18. If all is successful, the gate returns a welcome message to the traveler.
19. The Controller then requests the stored fingerprint feature vector of the user attempting to cross the border using his/her passport number.
20. The database returns the feature vector to the Fingerprint Controller.
21. The retrieved features and extracted features are compared and the result is forwarded to the gate.
22. The face of the user is then captured using the Camera and forwarded to the Camera Controller along with the user's data.
23. The Camera Controller checks if the face image is spoofed and extracts the feature vector of the user's face.
24. The Controller then requests the stored face feature vector of the user attempting to cross the border using his/her passport number.
25. The database returns the feature vector to the Camera Controller.
26. The retrieved features and extracted features are compared and the result is forwarded to the gate.
27. If all is successful, the gate returns a welcome message to the traveler.

3.2.4 Database Diagram

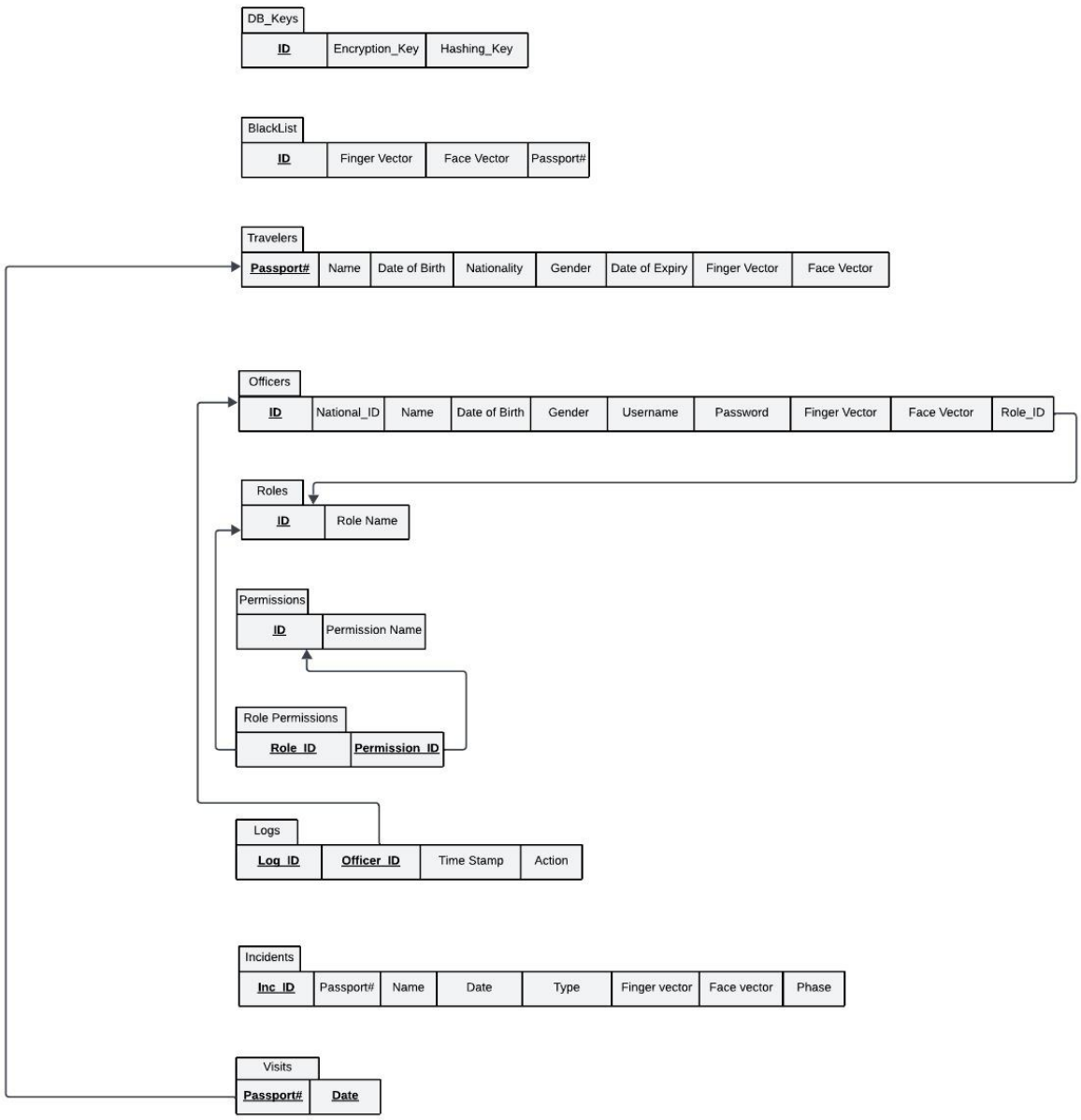

Figure 3.5: System Database Diagram

The ten tables in the database (shown in Figure 3.5) are:

1. **DB_Keys**: This table stores the cryptographic keys used for data encryption and hashing operations.
2. **Blacklist**: It stores the passport number, fingerprint and face features of blacklisted individuals.
3. **Travelers**: It stores passport details acquired from the OCR scan of the traveler's passport. It also has their fingerprint and face feature vectors for identity verification.
4. **Officers**: Along with an officer ID and a role ID, it stores passport details acquired from the OCR scan of the officer's passport as well as their credentials (username and password). It also stores each officer's biometric features. The stored data is used for authentication when logging into the system.
5. **Roles**: It includes a Role ID and its corresponding Role Name (Junior, Admin, Manager).
6. **Permissions**: This table has the IDs of Permissions and their corresponding names.
7. **Role Permissions**: This table links the permission id and role id to identify what actions each role has authorization or permission to perform.
8. **Logs**: It records each system log performed by airport officers. It includes a log ID, the officer's ID, a timestamp indicating when the action occurred, and a description of the specific action taken.
9. **Incidents**: This is a list of the attempts flagged by the system with details of the traveler associated with each. Incidents could be a spoofing attempt, identity mismatch or a blacklisted individual trying to cross the border. Incident information such as ID, type and phase are also stored.
10. **Visits**: It stores the passport number of a traveler and date for each visit to the country.

Chapter 4 Implementation and Testing

4.1 Datasets

4.1.1 NUAA PI

NUAA Photo Imposter dataset [20] is property of the Nanjing University of Aeronautics and Astronautics. It contains images of real access attempts and print-attacks of **15** users with about 500 images each. There is a total of **5105** real images and **7509** fake images. The images contain frontal faces with a neutral expression captured using a generic webcam. Users were also told to avoid eye-blinks and head movements. The attacks are performed using printed photographs on photographic paper and A4 paper. The dataset comprises three folders (DetectedFace, Raw, NormalizedFace) each containing two databases, Client and Imposter for genuine and spoofed images. Table 4.1 shows a description of each database offered by NUAA.

Table 4.1: NUAA PI Databases

Database	**Contains**	**Sensor Type**	**Subfolders**
Raw	Images in raw format	Generic Webcam	1. ClientFace 2. ImposterFace
DetectedFace	Images output by a face detector		
NormalizedFace	Geometrically normalized face based on eye coordinates		

Samples from each of the subsets in NUAA PI are shown in Figure 4.1.

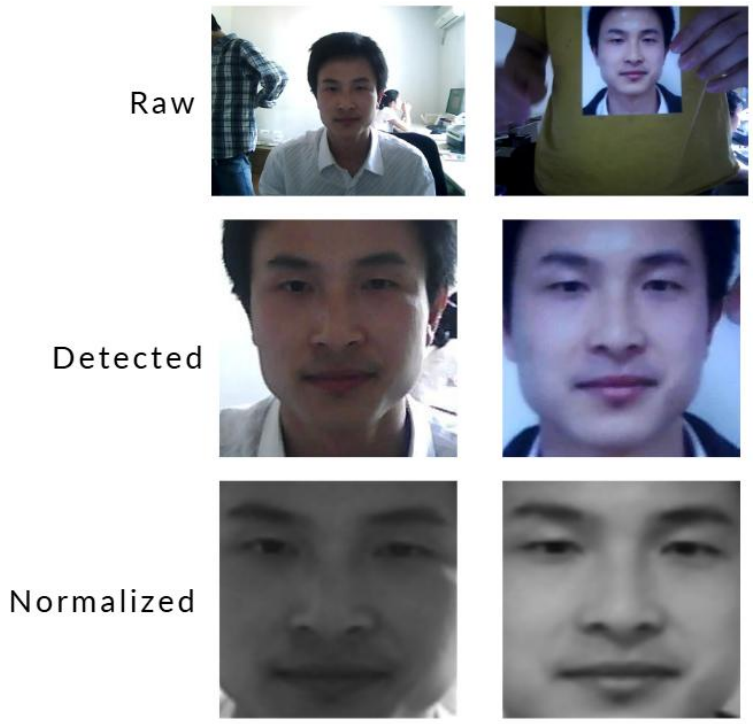

Figure 4.1: NUAA PI Sample Face Images
Each row has an image from ClientFace (Left), and ImposterFace (Right)

4.1.2 FVC2002

FVC2002 is the Second International Competition for Fingerprint Verification Algorithms [37] [38]. Four different databases (DB1, DB2, DB3 and DB4) were collected using different sensors/technologies. With a total of 110 identities, each database has 8 impressions of each identity's fingerprint, resulting in 880 fingerprints in each database and 3520 fingerprint images in all. Table 4.2 shows a description of each database. Samples from each database are shown in Figure 4.2 below.

Table 4.2: FVC2002 Database [38]

Database	**Sensor Type**	**Image Size**	**Resolution**
DB1	Optical Sensor	388x374 (142 Kpixels)	500 dpi
DB2	Optical Sensor	296x560 (162 Kpixels)	569 dpi
DB3	Capacitive Sensor	300x300 (88 Kpixels)	500 dpi
DB4	SFinGe v2.51	288x384 (108 Kpixels)	About 500 dpi

Figure 4.2: FVC2002 Sample Fingerprint Images
From left to right: DB1 (real), DB2 (real), DB3 (real) and DB4 (spoofed)

4.1.3 Created Dataset

Due to limited resources, during the implementation of this project a small-scale fingerprint dataset was created to support the development of the spoof detection model. The dataset consists of fingerprint images from a single individual, captured using an R307 optical fingerprint sensor.

A total of **202** images were collected and divided between four databases, three of which contain spoofed images using different materials to simulate presentation attacks. Table 4.3 shows a description of each database.

Table 4.3: Created Dataset Summary

Database/ Material Used	**Sensor Type**	**Number of Images**	**Image Size**	**Resolution**
Real	R307 Optical Sensor	101	256x288 (73.7 Kpixels)	500 dpi
Clay		49		
Tape		44		
Glue		8		

Samples from all databases are shown in Figure 4.3 below.

Figure 4.3: Created Dataset Sample Fingerprint Images
From left to right: Real, Clay, Tape and Glue

4.2 Technologies and Methodologies

The core technologies, tools and methodologies used throughout the project are presented in this subsection. It serves as the technical backbone of the project, outlining the frameworks, models, algorithms and libraries employed in both the face and fingerprint modules.

4.2.1 Software Tools and Libraries

4.2.1.1 Torch:

PyTorch [39] is a Python package that provides tensor computation (like NumPy) with strong Graphics Processing Unit (GPU) acceleration and deep neural networks built on a tape-based autograd system, meaning it automatically computes gradients.

In this project, PyTorch's submodule *nn* was used for its utilities that help define and train deep learning models.

4.2.1.2 Torchvision:

The Torchvision [40] package consists of popular datasets, model architectures, and common image transformations for computer vision.

Torchvision's submodules *transforms* and *models* were used to import DenseNet-121 [23] and preprocess the images before passing them to the Artificial Intelligence (AI) models.

4.2.1.3 NumPy (Numerical Python):

NumPy [41] is a Python library that adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Since it is designed to work with matrices, a NumPy array was used in this project to structure the raw fingerprint data into a 2D grid that matches the image's dimensions. This grid is then used by PIL [42] to create the image. NumPy was also used to convert extracted features for both the fingerprint and the face to NumPy arrays.

4.2.1.4 PIL (Python Imaging Library, Pillow):

PIL [42] supports the opening, manipulation and saving of different image file formats (BMP, JPEG, etc.).

PIL was used to convert the grid containing the raw fingerprint data into an image. Each number in the grid represents pixel brightness. For a greyscale fingerprint image, each pixel has a number from 0 (black) to 255 (white) to show its brightness. It was also used to convert images to RGB or greyscale as needed. Additionally, it was also used to load images during the designing of the models utilized during the course of this project.

4.2.1.5 OpenCV

OpenCV [43] is an open-source library that includes several hundreds of computer vision algorithms. It includes modules like *imgproc* (Image Processing), *imgcodecs* (Image file reading and writing), *objdetect* (Object Detection), *videoio* (Video I/O), *dnn* (Deep Neural Network) and more.

In this project, OpenCV handles video capture, face detection, image processing and image file manipulation.

4.2.1.6 OS

This module provides a portable way of using operating system dependent functionality. It is used to import/download images from/to specified folders or paths.

4.2.1.7 Cryptography

cryptography [44] is a package which provides cryptographic recipes and primitives to Python developers. It includes both high-level recipes and low-level interfaces to common cryptographic algorithms such as symmetric ciphers, message digests, and key derivation functions.

4.2.1.8 Fingerprint-enhancer

This is a program based on the method proposed by Hong *et al.* [45]; the usage of an oriented Gabor filter bank to enhance the fingerprint image. The orientation of the Gabor filters is decided by the orientation of ridges in the input image. The enhancement function is described in Appendix B: B.2.

4.2.1.9 SFinGe

Synthetic Fingerprint Generator (SFinGe) is an application developed by Cappelli *et al.* [46] . It is used to create synthetic fingerprints by utilizing Gabor-like space-variant filters for iteratively expanding an initially empty image containing just one or a few seeds. A directional image model, whose inputs are the number and location of the fingerprint cores and deltas, is used for tuning the filters. Very-realistic fingerprint images are obtained after the final noising-and-rendering stage [37].

4.2.1.10 PostgreSQL

An open-source object-relational database that uses and extends the SQL language.

The availability of a GUI tool, pgAdmin, makes the utilization of PostgreSQL simple and was therefore used in this project to create the tables mentioned in Section 3.2.4.

4.2.2 Hardware Tools

4.2.2.1 Logitech BRIO 100 Webcam

Full HD 1080p webcam with USB connectivity to the PC.

4.2.2.2 R307 Fingerprint Sensor

The R307 [47] is an optical fingerprint sensor with Transistor-Transistor Logic (TTL) interface.

4.2.2.2.1 Hardware Connectivity

The R307 sensor can be connected to an Arduino or to a computer through a USB-TTL. A USB-TTL interface is a USB module which provides 16 TTL I/O lines. This type of serial communication is frequently used with the Universally Asynchronous Receiver/ Transmitter transmission technique (UART). Figure 4.4 shows the exterior interface of the sensor.

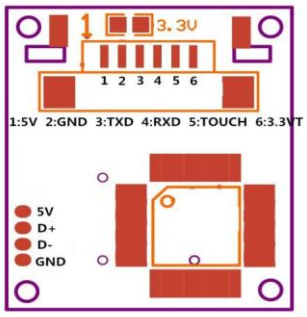

Figure 4.4: R307 Exterior Interface

Figure 4.5 shows a digital circuit diagram created using Cirkit Designer to show wiring connections between the R307 fingerprint sensor and the USB-to-TTL converter. The sensor's TX, RX, 5V, and GND pins are connected to the

corresponding RX, TX, VCC, and GND pins on the USB-to-TTL module, enabling direct serial communication with a PC.

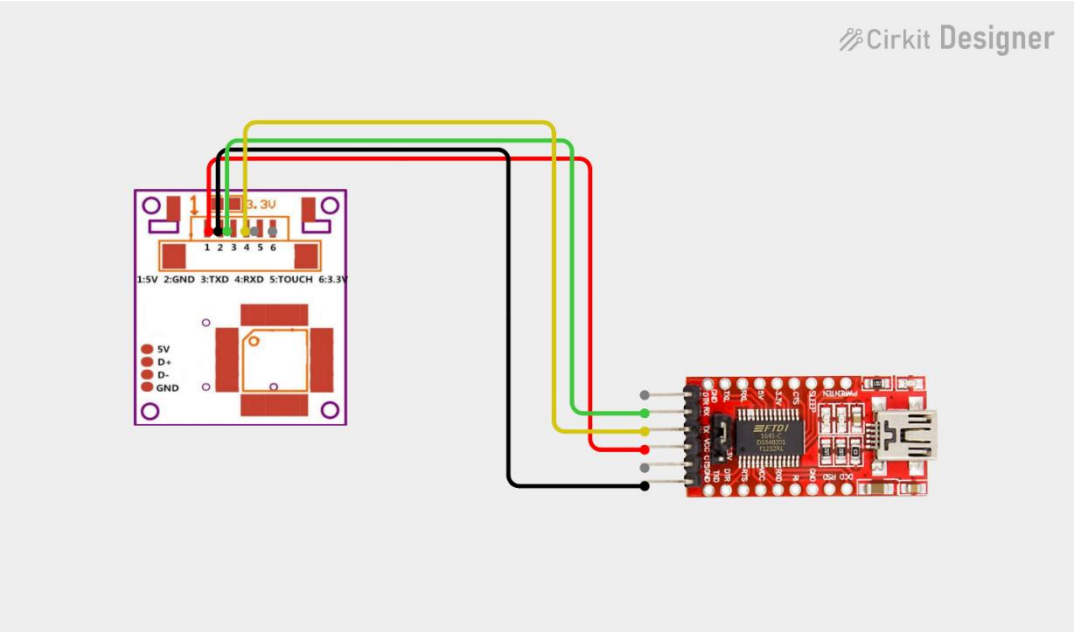

Figure 4.5: R307 Fingerprint Sensor Digital Circuit Diagram

The diagram shows color-coded connections between the R307 fingerprint sensor and the USB-to-TTL module. The visual highlights VCC (Red), GND (Black), TX to RX (Green), and RX to TX (Yellow) lines for correct serial communication setup

The USB-to-TTL module is then connected to the PC using a USB-A to Micro-USB cable, shown in Figure 4.6.

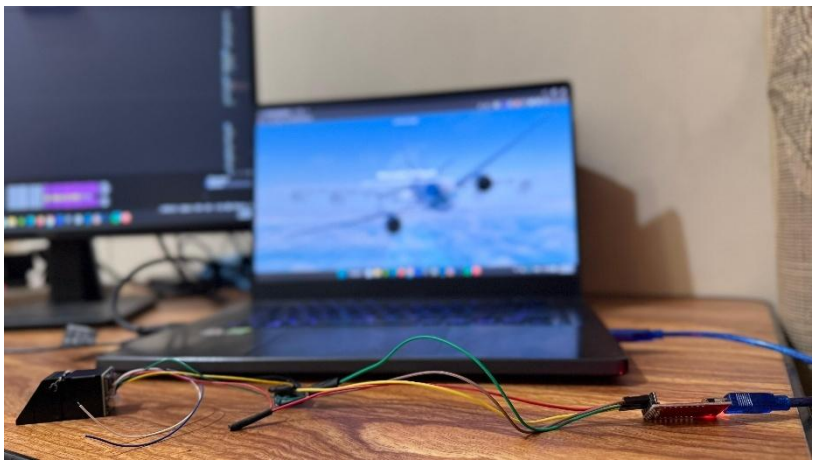

Figure 4.6: Wiring Connections of R307 to PC

4.2.2.2.2 Communication Protocol

When communicating, the transferring and receiving of commands/data/results are all wrapped in data package format (Figure 4.7).

Header	Adder	Package identifier	Package length	Package content (instuction/data/Parameter)	Checksum

Figure 4.7: R307 Data Package Format [47]

The definition of the data package is shown in Table 4.4 below.

Table 4.4: R307 Definition of Data Package [47]

Name	Symbol	Length	Description	
Header	Start	2 bytes	Fixed value of 0xEF01; High byte transferred first.	
Adder	ADDER	4 bytes	Default value is 0xFFFFFFFF, which can be modified by command. High byte transferred first and at wrong adder value, module will reject transfer.	
Package identifier	PID	1 byte	01H	Command packet.
			02H	Data packet: Data packet shall not appear alone in executing processes, must follow command packet or acknowledge packet.
			07H	Acknowledge packet.
			08H	End of Data packet.
Package length	LENGTH	2 bytes	Refers to the length of package content (command packets and data packets) plus the length of Checksum (2 bytes). Unit is byte. Max length is 256 bytes. And high byte is transferred first.	
Package contents	DATA	—	It can be commands, data, command's parameters, acknowledge result, etc. (fingerprint character value, template are all deemed as data);	
Checksum	SUM	2 bytes	The arithmetic sum of package identifier, package length and all package contents. Overflowing bits are omitted. High byte is transferred first.	

4.2.2.2.3 System Resources

The module system provides many resources for users' use, however only the one needed for this project is described below.

Image Buffer

The Image Buffer stores fingerprint images in a resolution of 256×288 pixels, using BMP format.

To reduce transmission time over UART, the sensor sends only the upper 4 bits (4-bit grayscale) of each pixel, resulting in 16 gray levels. Each byte transmitted encodes two adjacent pixels (one in the high nibble, one in the low nibble), effectively halving the data width. As a result, the sensor transmits an image of size 128×288 bytes, which the PC then expands back into a standard 256×288, 8-bit BMP format by multiplying each 4-bit pixel value by 16.

4.2.2.2.4 Module Instruction System

R307 provides 23 instructions that enable operations such as capturing fingerprint images, uploading and downloading them, performing matching and more. However, since this project aims to only capture and save the image using the sensor and then perform identification using AI models, only two of these instructions were necessary for this project, as follows:

- GenImg:

Function: capture fingerprint image and store it in the Image Buffer.

Command Package Format (Figure 4.8):

2 bytes	4 bytes	1 byte	2 bytes	1 byte	2 bytes
Header	Module address	Package identifier	Package length	Instruction code	Checksum
0xEF01	XXXX	01H	03H	01H	05H

Figure 4.8: R307 GenImg Command Package Format [47]

Response Package Format (Figure 4.9):

2 bytes	4 bytes	1 byte	2 bytes	1 byte	2 bytes
Header	Module address	Package identifier	Package length	Confirmation code	Checksum
0xEF01	XXXX	07H	03H	xxH	sum

Figure 4.9: R307 GenImg Response Package Format [47]

The confirmation code in the response packet could be:

1. **00H**: Finger collection success.
2. **01H**: Error when receiving package.
3. **02H**: Cannot detect finger.
4. **03H**: Fail to collect finger.

- UpImage:

Function: upload the image in Image Buffer to PC.

Command Package Format (Figure 4.10):

2 bytes	4 bytes	1 byte	2 bytes	1 byte	2 bytes
Header	Module address	Package identifier	Package length	Instruction code	Checksum
0xEF01	XXXX	01H	03H	0aH	000eH

Figure 4.10: R307 UpImage Command Package Format [47]

Response Package Format (Figure 4.11):

2 bytes	4 bytes	1 byte	2 bytes	1 byte	2 bytes
Header	Module address	Package identifier	Package length	Confirmation code	Checksum
0xEF01	XXXX	07H	03H	xxH	sum

Figure 4.11: R307 UpImage Response Package Format [47]

The confirmation code in the response packet could be:

1. **00H**: Ready to transfer the following data packet.
2. **01H**: Error when receiving package.
3. **0FH**: Failed to transfer data packet.

Fingerprint raw data will be transferred after the confirmation code.

4.2.3 Fingerprint Image Enhancement Techniques

4.2.3.1 Fingerprint Features

A fingerprint, as shown in Figure 4.12, is the reproduction of a fingertip epidermis, produced when a finger is pressed against a smooth surface. The most evident structural characteristic of a fingerprint is a pattern of interleaved *ridges*

and *valleys*; in a fingerprint image, ridges (also called ridge lines) are dark, whereas valleys are bright [37] [48].

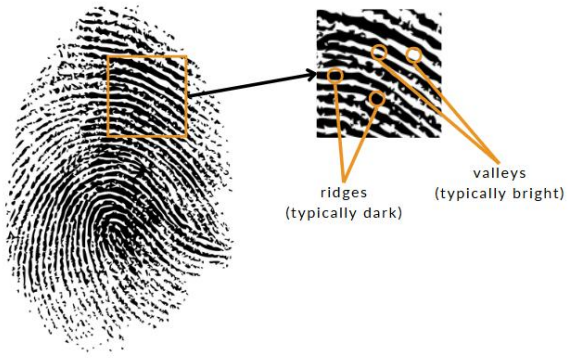

Figure 4.12: Ridges and Valleys in a Fingerprint

4.2.3.2 Gaussian Blur

In image processing, a Gaussian blur is the result of blurring an image by a Gaussian function. It is typically used to reduce image noise and detail. Results of Gaussian Blurring at increasing intensities can be seen in Figure 4.13.

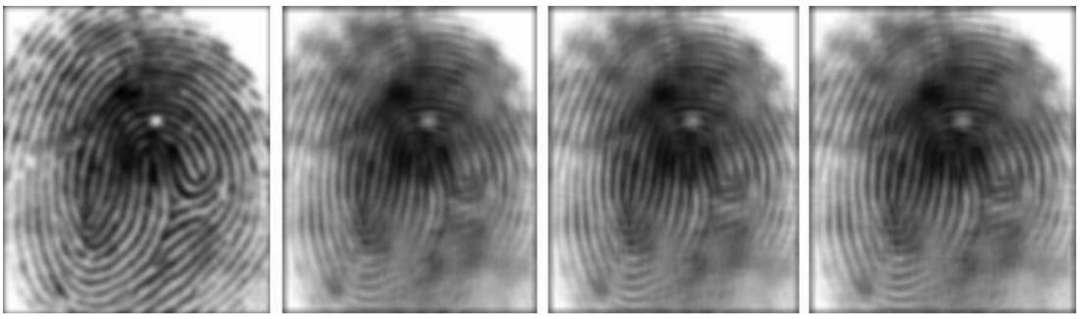

Figure 4.13: Application of Gaussian Blur in Different Intensities [49]

4.2.3.3 Thresholding

In digital image processing, thresholding is a method of segmenting images. Thresholding methods replace each pixel in an image with a black pixel if the image intensity $I_{i,j}$ is less than a fixed value called the threshold T, or a white pixel if the pixel intensity is greater than that threshold. A fingerprint image before and after applying a threshold is shown in Figure 4.14.

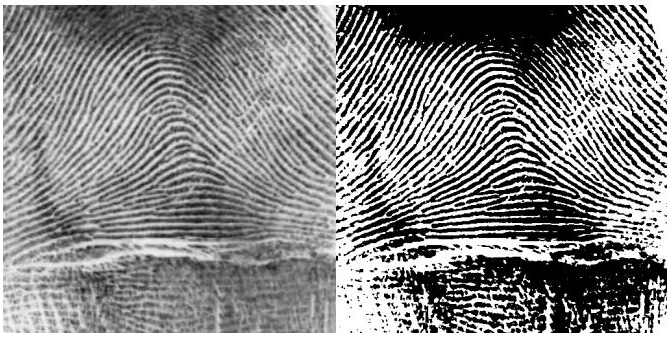

Figure 4.14: Fingerprint Image Before and After Applying a Threshold [50]

4.2.3.4 Erosion

Erosion is one of the fundamental operations in morphological image processing from which all other morphological operations are based. The erosion operation usually uses a structuring element for probing and reducing the shapes contained in the input image. Erosion can be useful when reducing noise in an image, as seen in Figure 4.15.

Figure 4.15: Erosion Applied on a Noisy Fingerprint Image [51]

4.2.3.5 Dilation

Dilation is the second fundamental operation in morphological image processing, along with erosion. The dilation operation, like erosion, uses a structuring element. It expands the shapes contained in the input image, as seen in Figure 4.16. It can be useful when closing the gaps in fingerprint images.

Figure 4.16: Dilation Applied on a Noisy Fingerprint Image [51]

4.2.3.6 Opening

Opening serves computer vision and image processing as a basic workhorse of morphological noise removal. It is erosion followed by dilation, resulting in the removal of small objects, or noise, from an image. The result of applying opening on a noisy fingerprint can be seen in Figure 4.17.

Figure 4.17: Opening Applied on a Noisy Fingerprint Image [51]

4.2.3.7 Closing

In image processing, closing is, together with opening, the basic workhorse of morphological noise removal. During closing, the dilation operation is implemented, followed by erosion. This removes small holes in an image, like those found in the fingerprint along a ridge. Figure 4.18 shows the effect of closing on a noisy fingerprint image.

Figure 4.18: Closing Applied on a Noisy Fingerprint Image [51]

4.2.4 AI Models and Configuration Settings

4.2.4.1 DenseNet121

DenseNet [23] is a Dense Convolutional Network which connects each layer to every other layer in a feed-forward fashion. Where traditional convolutional networks have L layers with L connections – one between each layer and its subsequent layer. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNet connections between layers are illustrated in Figure 4.19.

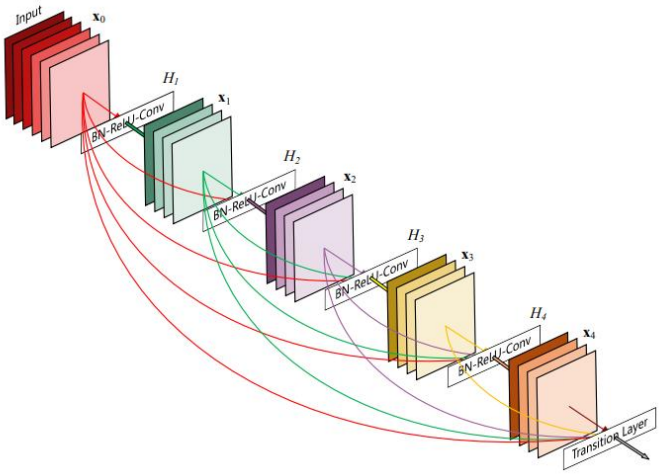

Figure 4.19: DenseNet Layer Connections

In contrast to other CNNs, such as ResNets, DenseNets do not combine features through summation, but instead through concatenation. A big advantage of DenseNets is the improved flow of information and gradients throughout the network, making them easy to train. Each layer has direct access to the gradients from the loss function and the original input signal, leading to an implicit deep

supervision [52]. Deep supervision enforces intermediate layers to learn discriminative features.

DenseNet-121 is a variant of Dense Convolutional Networks, characterized by a compact yet deep architecture. As illustrated in Figure 4.20 below, the network consists of:

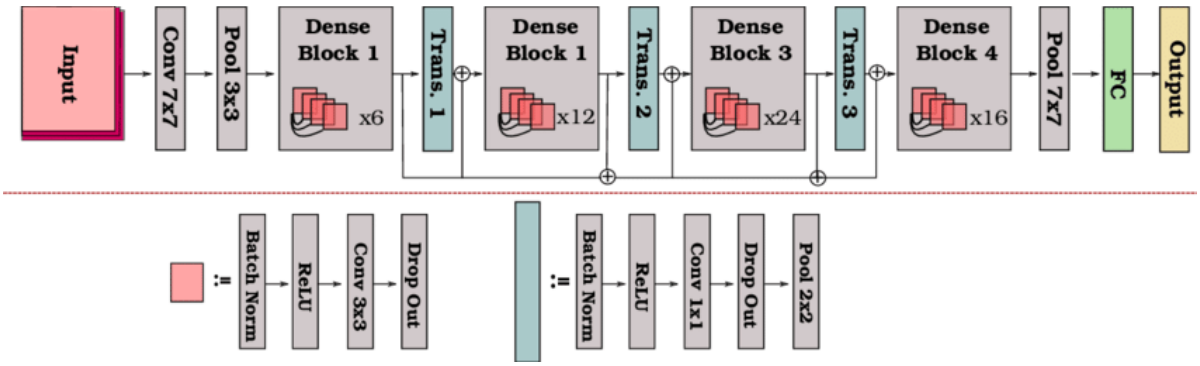

Figure 4.20: DenseNet-121 Architecture [23]

- A convolutional layer.
- Four dense blocks, each containing multiple densely connected convolutional layers.
- Three transition layers between the dense blocks, which include batch normalization layer, a 1x1 convolutional layer, and 2x2 average pooling.
- A final classification layer with 7x7 average pooling and a 1000D fully connected layer followed by SoftMax.

Each dense block in DenseNet-121 has a different number of layers:

- Dense Block 1: 6 layers.
- Dense Block 2: 12 layers.
- Dense Block 3: 24 layers.
- Dense Block 4: 16 layers.

This leads to a total of 121 layers, hence the name DenseNet-121.

DenseNet was trained and evaluated on three major datasets, **CIFAR** [53] [54], **SVHN** [55] and **ImageNet** [56]. In this work, a pre-trained DenseNet-121 model trained on the **ImageNet-1K** subset is leveraged and fine-tined to match the specific tasks required throughout this project.

ImageNet is a large database consisting of images scraped from online image searches using synonyms in multiple languages. The subset used, **ImageNet-1K**, involves 1000 classes, 1,281,167 training images, 50,000 validation images and 100,000 test images.

Pre-trained models are beneficial for reasons like saving time and utilizing learned features that can be transferred to your dataset.

4.2.4.2 Cross Entropy Loss

Entropy calculates the degree of randomness or disorder within a system. It is used to measure the uncertainty of an event. The greater the value of entropy, the greater the uncertainty for the probability distribution, and the smaller the value, the less uncertainty.

In the context of classification models, cross-entropy measures the difference between the discovered probability distribution by the model and the predicted values. According to the loss calculated, this function adjusts the weights of the machine learning model to try to reach an optimal solution. Its objective is to minimize errors by lowering the loss value.

4.2.4.3 Adam Optimizer

Short for “Adaptive Moment Estimation”, is an iterative optimization algorithm used to minimize the loss function during the training of neural networks. It uses the squared gradients to scale the learning rate and uses the moving average of the gradient instead of the gradient itself to reach the global minima [57].

4.2.5 Face Detection and Capturing

4.2.5.1 Haar Cascade Classifier

Object Detection using Haar-based cascade classifiers is an effective method proposed by Viola and Jones [58] It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. The *Cascading* aspect comes from the combination of weak classifiers to create a strong classifier using a cascade classifier. It is then used to detect objects in other images. A general representation of training a Haar classifier is shown in Figure 4.21.

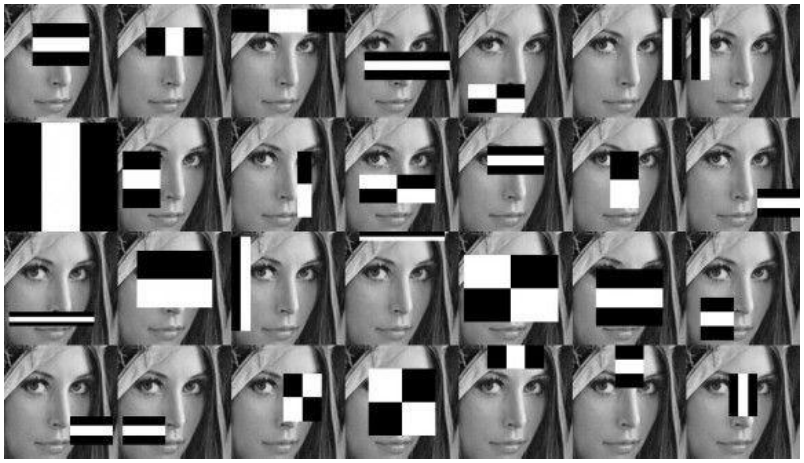

Figure 4.21: A General Representation of Training a Haar Classifier [59]

Adaboost training [60] is utilized to choose the best features and train the classifiers to use them; this is because most Haar features calculated will be irrelevant. It uses a learned threshold to separate non-objects from objects.

4.2.5.2 Dlib Frontal Face Detector

Dlib's frontal face detector is used to identify human faces in images and video streams. Dlib's Histogram of Oriented Gradients (HoG) + Linear Support Vector Machine (SVM) algorithm is used to detect front-on faces. It is based on the HoG feature descriptor with a linear SVM machine learning algorithm to perform face detection [61].

4.2.5.3 Caffe

The OpenCV's deep learning face detector runs on the Single-Shot Detector (SSD) framework with a ResNet base network. An SSD is an object detection algorithm that localizes and classifies objects within an image in a single pass [62], [63]. This pre-trained face detection model is taken advantage of in this project to detect faces during the face detection and capturing phase.

4.2.5.4 Dlib's Shape Predictor

This is a pre-trained facial landmark detector inside the Dlib library used to estimate the location of 68 (x,y)-coordinates that map to facial structures on the face [64]. The indexes of the 68 coordinates can be visualized in Figure 4.22 below.

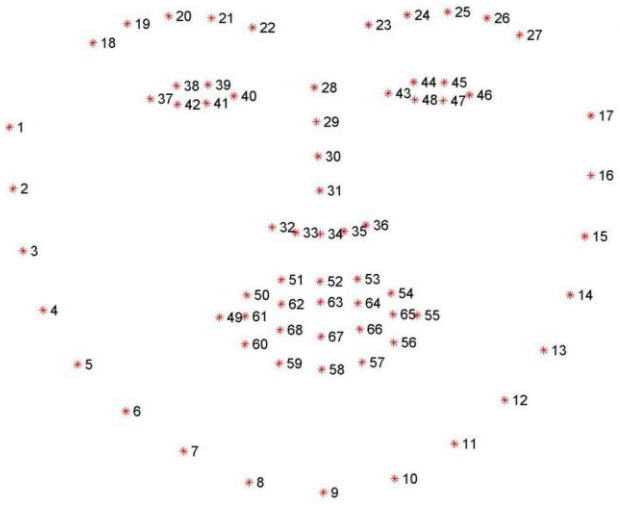

Figure 4.22: Dlib Indexes of the 68 Landmark Coordinates

These annotations are part of the 68-point **iBUG 300-W** [65], [66] dataset which the Dlib facial landmark predictor was trained on.

In order to detect facial landmarks, the face must first be localized within the image. This can be done using a number of ways, two of which are tested during this project: Caffe Model for Face Detection and Dlib's Face Detector using HoG. These approaches are described briefly in Sections 4.2.5.3 and 4.2.5.4.

4.2.6 Feature Extraction and Comparison

4.2.6.1 Cosine Similarity

A measure of similarity between two non-zero vectors. Cosine similarity is the cosine of the angle between the vectors. The cosine similarity always belongs to the interval [-1, +1]. In the scope of comparing fingerprint and face feature vectors, the similarity score is ranged between 0 and 1, with 0 meaning the vectors are orthogonal (completely different) and 1 meaning they are identical.

4.2.6.2 SoftMax

The SoftMax function takes as input a tuple z of K real numbers and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. That is, prior to applying SoftMax, some tuple components could be negative, or greater than one; and might not sum to 1; but after applying SoftMax, each component will be in the interval (0,1), and the components will add up to 1, so that they can be interpreted as probabilities.

4.3 Data Protection and GDPR Compliance

This project is developed in compliance with the EU's GDPR [13]. Most importantly, it adheres to the following articles:

- **Article 32 (1) (a)**: "*the pseudonymisation and encryption of personal data.*"
- **Article 32 (2)**: "*In assessing the appropriate level of security account shall be taken in particular of the risks that are presented by processing, in particular from accidental or unlawful destruction, loss, alteration, unauthorised disclosure of, or access to personal data transmitted, stored or otherwise processed.*"

4.3.1 Complying with Article 32 (1) (a)

Article 32 (1) (a) is achieved through the hashing and encryption of data. The algorithms used for both are mentioned below:

Hashing:

- **SHA-256**: Secure Hash Algorithm – 256 is used to hash officer passwords stored in the database.

Encryption:

- **AES**: Advanced Encryption Standard is a symmetric encryption algorithm used to encrypt the sensitive data in the database. The data encrypted is:
    - Face feature vectors
    - Fingerprint feature vectors
    - Passport numbers
    - Names
    - National IDs
    - Usernames
    - Incident Type
    - Incident Phase
    - Role and Permission IDs

PyCA's cryptography library [44] is utilized to encrypt and hash the required fields.

4.3.2 Complying with Article 32 (2)

Article 32 (2) is adhered to by bringing on the ideas of roles, permissions and access levels. In this project, three roles are defined: Junior, Admin and Manager, each with increasing levels of privilege. The permissions given to these roles are as follows:

- **Junior**: Log viewing and incident management.
- **Admin**: Log viewing, incident management and user enrolment.

- **Manager**: Log viewing, incident management, user enrolment, officer enrolment and officer permission adjustment.

Any action taken by an authorized member of the system is logged to maintain accountability.

4.3.3 Password Policies and Multi-Factor Authentication

The GDPR does not directly mention any specific requirements concerning password security. However, in a mission to achieve as much security as possible, some common password and authentication guidelines are implemented, such as:

- The minimum password length should be at least eight characters.
- Passwords should contain at least one character from each of the four-character categories: uppercase, lowercase, numeric and special characters.
- Passwords should never be stored in plaintext but should be encrypted using strong encryption algorithms.
- Users must authenticate with Multi-Factor Authentication (MFA) techniques.

Multi-Factor Authentication [67] is a security measure that protects individuals and organizations by requiring users to provide two or more authentication factors to access an account.

A user is first prompted for their username and password, but then they are required to verify their identity by some other means. The three common authentication methods to verify a user's identity are:

- Something you know: Information you know, such as a password or a Personal Identification Number (PIN).
- Something you have: Something you possess. For example, a code to a mobile phone.

- Something you are: A unique personal attribute, such as biometric authentication.

Since this project is based on the authentication of users using a multi modal biometric system, MFA is achieved here using credentials and the user's fingerprint and face.

4.4 Experiments and Results

This section shows the experimental procedures conducted to complete this project. Its purpose is to show the path followed to reach the final solution, highlighting the differences between each experiment and how one change can significantly change the results, capturing the step-by-step progress of each module. The experiments are divided according to the main modules of the system.

4.4.1 Fingerprint Acquisition

4.4.1.1 Experiment 1:

Initially, the sensor was connected to an Arduino Uno that was then connected to the PC. An Arduino was used to take advantage of the Adafruit Fingerprint library [68] since it implements all the sensor's instructions as ready to use functions.
After capturing a fingerprint image using the Arduino, the received bytes were forwarded to a Python script that converts the raw image data to a BMP format. However, only 24,000 bytes were received of entire data packets rather than just the fingerprint bytes (instead of the required 36,864 bytes of image data only). Additionally, the forwarded data included debugging text along with the image data, which further corrupted the output. The resulting image is shown in Figure 4.23 below.

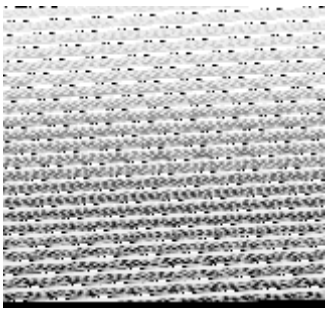

Figure 4.23: Fingerprint Acquisition - Experiment 1

Key issues identified during this experiment:

1. Only 24,000 bytes were received instead of 36,864 bytes of raw fingerprint data.
2. The received data included entire data packets, not just the raw fingerprint bytes.
3. Debugging text was forwarded along with the image data, corrupting the output.

Issue number 3 is addressed in the next experiment.

4.4.1.2 Experiment 2:

This trial addressed the third issue in the previous experiment: debugging text being forwarded with the image data. The only change in this experiment compared to Experiment 1 was that no debugging text was included in the raw image data. However, the number of bytes received was still 24,000, consisting of entire data packets. The data was then passed to the same Python script for conversion to BMP format. The resulting image is shown in Figure 4.24 below.

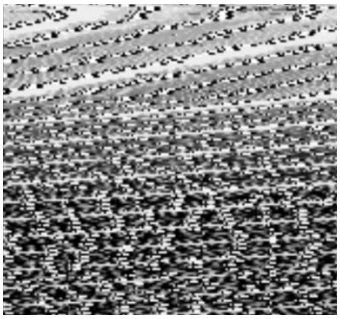

Figure 4.24: Fingerprint Acquisition - Experiment 2

Key issues in this experiment:

1. Only 24,000 bytes were received instead of 36,864 bytes of raw fingerprint data.
2. The received data included entire data packets, not just the raw fingerprint bytes.

4.4.1.3 Experiment 3:

After further analysis of this project's objectives, it was concluded that using an Arduino was not necessary since its primary advantages and usages were not required for the project. Instead connecting the sensor to the PC using a USB-TTL was more appropriate as it allowed full control with no bottleneck from Arduino.

In this experiment, the sensor was connected to the PC using USB-TTL. Functions that implement the sensor's instructions (see Section 4.2.2.2.4) were created and used to capture and upload the fingerprint image to the computer. The correct number of 36,864 bytes was received and expanded to 73,728 bytes to create a 256*288 image. The errors in this trial were that the converted bytes still consisted of whole data packets, and the width and height of the image were swapped during conversion to BMP. The resultant image is shown in Figure 4.25 below.

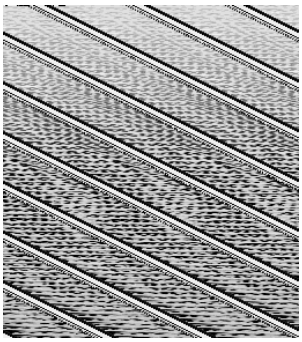

Figure 4.25: Fingerprint Acquisition - Experiment 3

The main errors in this experiment were:

1. The received bytes were of the whole data packet and not just the raw image data, causing the white lines.

2. The width and height of the image were swapped during conversion from raw image data to a BMP format.

4.4.1.4 Experiment 4:

The second error in Experiment 3 was addressed here: the width and height of the image being swapped during conversion.

The number of bytes received here was 49,536; much more than the required 36,864 but still including full data packets. In this trial, the fingerprint image could be seen, but the white lines from the rest of the data packet were still there. The resulting image is shown in Figure 4.26 below.

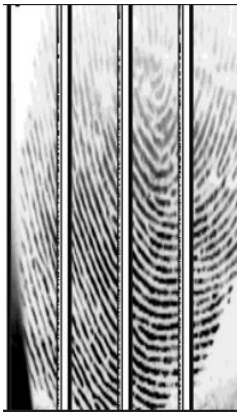

Figure 4.26: Fingerprint Acquisition - Experiment 4

The errors in this experiment were:

1. The number of received bytes exceeded the required number of 36,864.
2. The converted data still included full data packets.

4.4.1.5 Experiment 5:

After re-reading the sensor's documentation [47] and gaining a deeper understanding of how the data packets are structured and transmitted. It was at this point that the exact location of the fingerprint data within the packets, as well as the default data package length became clear, allowing for proper extraction.

The image data is located from the 10th to the 42nd byte, this can be seen in Table 4.4. Hence, the data was read in chunks of 43 and indices 9 to 41 were extracted as the image data. The resulting image after receiving 36,864 bytes of extracted fingerprint image data is shown in Figure 4.27 below.

Figure 4.27: Fingerprint Acquisition - Experiment 5

The final step was to expand the 128*288 image to a 256*288 BMP format image by multiplying each 4-bit pixel value by 16. The resulting image is shown in Figure 4.28 below.

Figure 4.28: Fingerprint Acquisition - Experiment 5 (After Expansion)

Experiment 5 is the last trial, and the final solution in the Fingerprint Acquisition phase.

4.4.2 Fingerprint Image Enhancement

4.4.2.1 Experiment 1:

The first approach involved using OpenCV's Image Processing module to process the fingerprint image and enhance features like ridges and valleys. The order of functions used to produce the enhanced image was:

1. **Gaussian blur**: Blurs the image and reduces noise.
2. **Thresholding**: Converts the blurred image to a binary one.
3. **Erosion**: Shrinks white regions to remove noise.
4. **Dilation**: Expands white regions to fill gaps.
5. **Opening**: Applies erosion followed by dilation to remove small noise.
6. **Closing**: Applies dilation followed by erosion to fill small gaps within ridges.

The resulting image after following this series of image processing functions on an image from the FVC2002 dataset is shown in Figure 4.29 below.

Figure 4.29: Fingerprint Image Before and After Enhancement - Experiment 1

4.4.2.2 Experiment 2:

After further research, a fingerprint enhancing library was found and tested on the same FVC2002 image as Experiment 1. This enhancement function is described thoroughly in Appendix B: B.2. The enhanced image is shown in Figure 4.30 below.

Figure 4.30: Fingerprint Image after Applying fingerprint_enhancer [45] - Experiment 2

A side-by-side comparison of the results is shown in Figure 4.31, where the left image represents the output from Experiment 1, and the right image represents the output from Experiment 2.

Figure 4.31: Comparison of Enhancement Results
Left: OpenCV-based Approach (Experiment 1), Right: fingerprint-enhancer (Experiment 2)

After evaluating the visual clarity and continuity of ridge structures, Experiment 2's method was chosen for further use in the system. It produced more accurate and consistent ridge patterns, which are essential for reliable feature extraction and fingerprint matching.

4.4.3 Fingerprint Identification and Spoofing AI Models

The DenseNet-121 [23] model, pre-trained on ImageNet, was selected as the backbone due to its dense connectivity and efficiency in feature extraction for image classification tasks. The optimizer used was Adam, and the loss function was categorical cross-entropy.

In the following subsections, experiments conducted before reaching the final models are presented. Each experiment is shown in a consistent format for ease of interpretation, beginning with a description of the experiment, followed by a table listing the primary hyperparameters and a table showing the results. Finally, each experiment is concluded with the limitations discovered during the trial.

4.4.3.1 Experiment 1:

In the initial trial of this stage, a multi-task model was trained using commonly used hyperparameter values.

Table 4.5 lists the primary hyperparameters and training setup used during the initial development of both the spoofing and identification models.

Table 4.5: Fingerprint Identification and Spoofing Detection AI Model - Experiment 1 Configurations

Training Parameters and Settings	**Value**	
Learning Rate	0.001	
Batch Size	8	
Dropout Rate	0.5	
Number of Epochs	100	
Databases Used	Identification: DB1, DB2	Spoof: DB4
Images Enhanced (Yes/No)	No	
Data Split (Train/Validate/Test)	Random 80/20/20	
Data Preprocessing	Normalization: using ImageNet statistics [56].	

Only DB1 and DB2 from the FVC2002 dataset were used to represent real fingerprint images. DB3 was excluded due to notable differences in image

characteristics compared to those captured with the R307 sensor, which is the hardware used in this system. Those differences are evident when comparing the images in Section 4.1.2 with the ones in Section 4.1.3. DB4, containing fake images, was the only one used for spoofing detection.

Under these settings, the spoofing detection model achieved an accuracy of 100.00%, while the identification model achieved an accuracy of 9.09%.

Table 4.6: Fingerprint Identification and Spoofing Detection AI Model - Experiment 1 Results

Identification Accuracy	Spoofing Detection Accuracy
9.09%	100.00%

The results found in this experiment were unsatisfactory, prompting a closer analysis of the strategies used. The following errors were concluded to have been made:

1. The identification model was unnecessarily trained on fake fingerprint data.
2. The random split failed to guarantee balanced representation across identities, meaning that some identities might not have been present in the training set at all.

4.4.3.2 Experiment 2:

This experiment aimed to address the data splitting errors made in the previous experiment. To ensure a balanced representation of all identities in the training set, a splitting by impressions method was proposed for the real images and a splitting by identities method for the fake images.

Real images:

Since there are 2 DBs used with 8 impressions each, there are a total of 16 impressions per identity. When split as 80/20/20 for training, validation and

testing respectively, 12 impressions are used for training and 4 impressions are split equally between validation and testing (2 for each).

Fake images:

Since spoofing detection achieved 100% accuracy in Experiment 1, the goal here was to prevent overfitting and generalize by holding out identities and testing on them. Applying an 80/20/20 split for training, validation and testing respectively resulted in 88 identities in training and 11 identities in each of validation and testing. Only DB4 contains generated fingerprint images, so there are 8 fake impressions per identity.

Table 4.7 lists the primary hyperparameters and training setup used during the second experiment in training and testing the fingerprint AI model.

Table 4.7: Fingerprint Identification and Spoofing Detection AI Model - Experiment 2 Configurations

Training Parameters and Settings	**Value**	
Learning Rate	0.001	
Batch Size	32	
Dropout Rate	0.5	
Number of Epochs	100	
Databases Used	Identification: DB1, DB2	Spoof: DB4
Images Enhanced (Yes/No)	No	
Data Split (Train/Validate/Test)	Real Images:	By impressions (12/2/2)
	Fake Images:	By identities (88/11/11)
Data Preprocessing	Normalization: using ImageNet statistics [56].	

Under these settings, the spoofing detection model achieved an accuracy of 98.05%, while the identification model achieved an accuracy of 64.61%.

Table 4.8: Fingerprint Identification and Spoofing Detection AI Model - Experiment 2 Results

Identification Accuracy	Spoofing Detection Accuracy
64.61%	98.05%

Experiment 2's results show a significant improvement over Experiment 1, but the achieved identification accuracy of 64.61% remains insufficient for deployment in high-security environments such as airport border control. Further analysis of the model's configuration and data handling strategies revealed the following limitations that may have impacted performance.

1. The absence of data augmentation means no diversity was introduced to the model.
2. Only including fake images when training spoofing detection; the model should train on both real and fake images to be able to differentiate.
3. The usage of a multi-task learning model for identification and spoofing with limited data.

Although Multi-Task Learning (MTL) can be efficient by allowing a single model to handle both spoofing detection and identification, this approach proved problematic in the context of limited data. In some cases, the simultaneous training of seemingly related tasks may hinder performance compared to single-task models. Commonly, MTL models employ task-specific modules on top of a joint feature representation obtained using a shared module. Since this joint representation must capture useful features across all tasks, MTL may hinder individual task performance if the different tasks seek conflicting representation, i.e., spoofing might rely on surface texture and noise, while identification depends on deep, fine-grained ridge patterns that differentiate identities.

4.4.3.3 Experiment 3:

Experiment 2 led to researching ways to overcome the limitations that come with using a multi-task model, one of the discovered ways was to create separate task heads and freeze layers. Task-specific heads allow the model to learn specialized features relevant to each task. Additionally, freezing one task-specific head enables focused training on the other task, isolating changes and allowing for task specialization without affecting the frozen task head. This is especially useful when one task performs well and does not need further training and continuing to do so hinders the other one's performance. Therefore, in this trial, two separate task heads were used, one for identification and for spoofing, the latter being frozen after 50 epochs to prioritize identification task.

Another issue that was addressed in this experiment was the absence of real images when training for spoofing detection. Hence, a stratified splitting approach was used to balance the amount of real and fake images given to the spoof head. Finally, data augmentation was used to enhance diversity during training.

Table 4.9 lists the primary hyperparameters and training setup used in this experiment with two separate task heads.

Table 4.9: Fingerprint Identification and Spoofing Detection AI Model - Experiment 3 Configurations

Training Parameters and Settings	**Value**	
Learning Rate	0.001	
Batch Size	32	
Dropout Rate	Identification Head: 0.6	Spoof Head: 0.5
Number of Epochs	100	
Databases Used	Identification: DB1, DB2	Spoof: DB1, DB2, DB4
Images Enhanced (Yes/No)	No	
Data Split (Train/Validate/Test)	Real Images:	By impressions (12/2/2)
	Real and Fake Images:	Stratified Split between Real and Fake
Data Preprocessing/ Augmentation	Normalization: using ImageNet statistics [56].	
	RandomHorizontalFlip: 0.5% probability	
	RandomVerticalFlip: 0.5% probability	
	RandomRotation: ±15 degrees	
	ColorJitter: ±0.2 brightness, ±0.2 contrast	

Under these settings, the spoofing detection model achieved an accuracy of 100.00%, while the identification model achieved an accuracy of 73.18%.

Table 4.10: Fingerprint Identification and Spoofing Detection AI Model - Experiment 3 Results

Identification Accuracy	**Spoofing Detection Accuracy**
73.18%	100.00%

The creation of two task-specific heads proved to be beneficial for the improvement of identification accuracy. However, the overall performance remained insufficient for high-security applications. The following limitation may have impacted performance.

1. Freezing the spoof head was not enough to raise identification accuracy to acceptable ranges in security environments.

4.4.3.4 Experiment 4:

Since freezing the spoof head after 50 epochs to prioritize identification still yielded unsatisfactory results, a two-model approach was tested here.

Identification Model

In addition to creating two models, enhancement, as shown in Section 4.4.2.2, was performed on the images passed to the identification model in order to eliminate the differences between DBs 1,2 and DB3. By doing so, 8 more impressions were available for each identity, resulting in a total of 24 impressions per identity. Since the enhanced fingerprint images are greyscale, the first layer of the model was modified to accept 1-channel inputs instead of 3-channel inputs (RGB). Lower learning and higher dropout rates were used here to prevent overfitting. Table 4.11 lists the primary hyperparameters and training setup used in the Identification model.

Table 4.11: Fingerprint Identification AI Model - Experiment 4 Configurations

Training Parameters and Settings	**Value**
Learning Rate	0.0001
Batch Size	32
Dropout Rate	0.6
Number of Epochs	55
Databases Used	DB1, DB2, DB3
Images Enhanced (Yes/No)	Yes
Data Split (Train/Validate/Test)	Stratified 70/15/15 split between DBs 1,2 and 3
Data Preprocessing/ Augmentation	Normalization: 1-channel images
	RandomHorizontalFlip: 0.5% probability
	RandomVerticalFlip: 0.5% probability
	RandomRotation: ±15 degrees

Under these settings, the identification model achieved an accuracy of 90.40%.

Table 4.12: Fingerprint Identification AI Model - Experiment 4 Results

Identification Accuracy
90.40%

Spoofing Model

As for the spoofing model, the images input were unenhanced because liveness detection usually relies on surface texture and noise, which are removed after enhancement. Finalizing the spoofing model involved two experiments:

4.4.3.5 Experiment 5:

Both real and fake images from the FVC2002 dataset were used to train the spoofing model.

Table 4.13 lists the primary hyperparameters and training setup used in this experiment with the spoofing model.

Table 4.13: Fingerprint Spoofing Detection AI Model - Experiment 5 Configurations

Training Parameters and Settings	Value
Learning Rate	0. 0001
Batch Size	32
Dropout Rate	0.5
Number of Epochs	10
Databases Used	DB1, DB2, DB3, DB4
Images Enhanced (Yes/No)	No
Data Split (Train/Validate/Test)	Stratified 70/15/15 split between Real and Fake images.
Data Preprocessing/ Augmentation	Normalization: using ImageNet statistics [56].
	RandomHorizontalFlip: 0.5% probability
	RandomVerticalFlip: 0.5% probability
	RandomRotation: ±15 degrees
	ColorJitter: ±0.2 brightness, ±0.2 contrast

Under these settings, the spoofing model achieved an accuracy of 100.00%.

Table 4.14: Fingerprint Spoofing Detection AI Model - Experiment 5 Results

Spoofing Accuracy
100.00%

While these results appear to be adequate for high-security environments such as airport borders, the model failed to detect actual spoofed fingerprints during real-world testing. This experiment is shown in Appendix A: Additional Tests and Experiments.

4.4.3.6 Experiment 6:

Real and fake images from the created dataset, samples shown in Section 4.1.3, were used in addition to FVC2002 dataset to ensure the model's ability to detect actual spoofing attempts and not just software-generated fingerprints.

Table 4.15 lists the primary hyperparameters and training setup used in this experiment with the spoofing model.

Table 4.15: Fingerprint Spoofing Detection AI Model - Experiment 6 Configurations

Training Parameters and Settings	**Value**
Learning Rate	0. 0001
Batch Size	32
Dropout Rate	0.5
Number of Epochs	10
Databases Used	DB1, DB2, DB3, DB4, Real (Created Dataset), Fake (Created Dataset)
Images Enhanced (Yes/No)	No
Data Split (Train/Validate/Test)	Stratified 70/15/15 split between Real and Fake images.
Data Preprocessing/ Augmentation	Normalization: using ImageNet statistics [56].
	RandomHorizontalFlip: 0.5% probability
	RandomVerticalFlip: 0.5% probability
	RandomRotation: ±15 degrees
	ColorJitter: ±0.2 brightness, ±0.2 contrast

Under these settings, the spoofing model achieved an accuracy of 99.82%.

Table 4.16: Fingerprint Spoofing Detection AI Model - Experiment 6 Results

Spoofing Accuracy
99.82%

The combined results of 90.40% identification accuracy and 99.82% spoofing detection accuracy were found to be satisfactory in regard to similar results reported in the literature, and the two models were tested in real-time. This is shown in Section 4.4.8.

4.4.4 Face Detection and Capturing

The face module begins with face detection and capturing with the webcam.

4.4.4.1 Experiment 1:

For this experiment, the Haar Cascade Classifier, as implemented in the OpenCV library and described in Section 4.2.5.1, was selected due to its widespread use and documented effectiveness for face detection. The classifier leverages Haar-like features and a cascaded decision-making process to identify faces in real-time video streams.

The experiment revealed several limitations in the Haar Cascade Classifier's performance under varying conditions. First, the classifier detected faces only when the subject's face was directly aligned with the camera. Additionally, minor changes in the subject's head position, such as tilting or shifting off-center, degraded detection accuracy and failed completely to detect face from side angles. The resulting bounding box for face detection is shown in Figure 4.32 below.

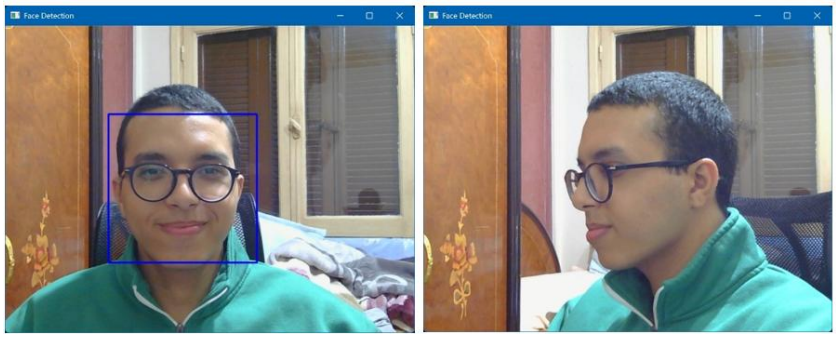

Figure 4.32: Face Acquisition - Experiment 1

Head tilt (Right) caused the face to not be detected

Key issues identified during this experiment:

1. Sensitivity to Positional Changes.
2. Failure to Detect Profile Views.

4.4.4.2 Experiment 2:

Following the limitations identified in the first experiment with the Haar Cascade Classifier, a pre-trained deep learning model based on the Caffe framework was employed, specifically designed for face detection. The Caffe model, briefly explained in Section 4.2.5.3, leveraging CNNs, was selected for its ability to handle diverse face orientations and environmental conditions, as demonstrated in prior studies. This approach aimed to address the issues of restricted detection angles, sensitivity to head position changes, and profile view failures observed in the previous experiment.

However, lack of alignment control hinders full automation, as it detects faces at any angle without restrictions, relying on users to position their faces correctly. Inconsistent orientations may impair subsequent modules like face enhancement and recognition, which require aligned features, conflicting with the system's goal of seamless, intervention-free face capture. The results are shown in Figure 4.33 below.

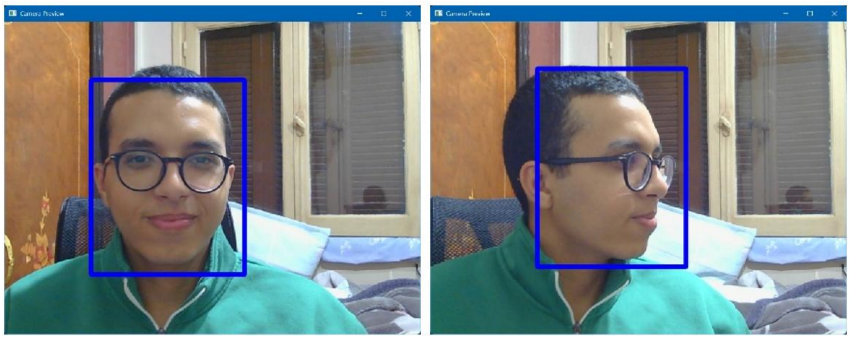

Figure 4.33: Face Acquisition - Experiment 2

Face is detected at any angle

The main errors discovered during this experiment were:

1. The dependency on user actions.
2. Inconsistency in facial orientation.

4.4.4.3 Experiment 3:

To address the issue of inconsistent face orientations in the Caffe model, this experiment integrates a facial landmark predictor using Dlib's 68-point shape predictor, described in Section 4.2.5.4, alongside the Caffe-based deep learning model. This approach enforces strict alignment criteria during the face capture phase, ensuring that captured faces were consistently near-frontal. This aligns with the system's requirement for standardized inputs to the face enhancement and recognition modules. The automated capture mechanism, triggered after a 5-second alignment hold of an aligned face and rejection of a misaligned face, minimized user intervention while maintaining user-friendly feedback. The results are displayed in Figure 4.34 below.

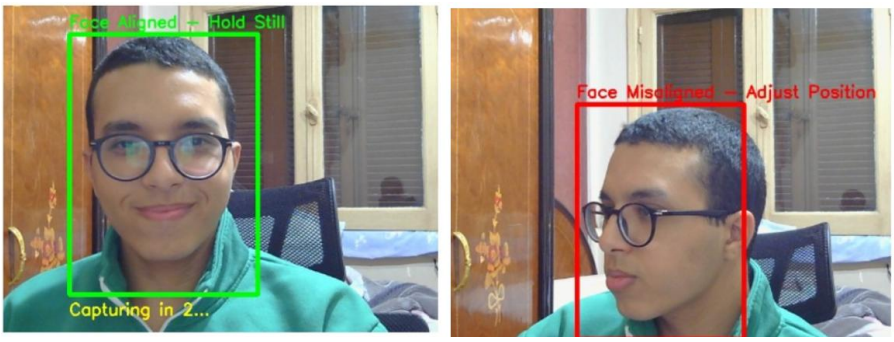

Figure 4.34: Face Acquisition - Experiment 3
Strict alignment criteria are enforced to ensure user maintains a frontal angle (Left)

This method was chosen to be the final technique used in the project as it not only addresses all the previously mentioned limitations in Experiments 1 and 2, but also aligns with the project's goals and requirements.

4.4.5 Face Recognition and Spoofing Detection AI Model

The model building phase focuses on developing a robust system for face recognition and spoofing detection, utilizing the DenseNet-121 architecture and the NUAA PI dataset. The face model is designed to simultaneously identify individuals (face recognition) and distinguish between real and fake faces (spoofing detection).

4.4.5.1 Experiment 1:

As a start to this phase, the original classifier was replaced with a custom head tailored for dual tasks.

Table 4.17 lists the primary hyperparameters and training setup used in this experiment with a dual-task head.

Table 4.17: Face Identification and Spoofing Detection AI Model - Experiment 1 Configurations

Training Parameters and Settings	**Value**
Learning Rate	0. 001
Batch Size	32
Dropout Rate	0.5
Number of Epochs	10
Features Extracted	512
Database(s) Used	DetectedFace
Images Enhanced (Yes/No)	No
Data Split (Train/Validate/Test)	Random 80/10/10
Data Preprocessing	Normalization: using ImageNet statistics [56].

Using these model configurations, the identification task achieved an accuracy of 99.60% and the spoofing task achieved an accuracy of 100.00% (as presented in Table 4.18).

Table 4.18: Face Identification and Spoofing Detection AI Model - Experiment 1 Results

Identification Accuracy	**Spoofing Detection Accuracy**
99.60%	100.00%

Since the model achieved high accuracies from the first trial, no more experiments were required in this phase. Experiment 1's model was selected as the final one for the face recognition and spoofing detection phase.

4.4.6 Feature Extraction

The developed models are then used to extract feature vectors representing the unique biometric characteristics of each user, one describing the facial traits and another describing fingerprint patterns. This is necessary before accurate identification of individuals can be performed.

The goal of feature extraction is to transform raw fingerprint images into high-dimensional numerical representations that can uniquely characterize a user. These vectors are used later for identity matching via similarity comparison.

To extract features, a forward hook is registered on the intermediate layer DenseBlock3 of the DenseNet-121 architecture used in both models, the identification model in the fingerprint module and the multi-task model in the face module. Forward hooks are attached to capture a layer's output. DenseBlock3 is a deep feature extraction layer that captures complex, high-level patterns in the input image. This layer can be seen in the model's architecture shown in Figure 4.20. In the fingerprint model, it learns to detect patterns such as ridge flow, bifurcations, and minutiae, while in the face model, it focuses on key facial regions, symmetry, and expression-invariant features.

4.4.7 Enrolment

The enrolment phase is a critical component of the biometric authentication system, as it establishes the initial identity profiles against which future authentication attempts will be compared. After the images are captured from the users, preprocessed, and its characteristics and patterns are represented as a

feature vector, that vector is stored in the system's database and linked to users by their IDs (Passport Number).

4.4.8 Real-Time Testing and Threshold Determination

Following the development and evaluation of the AI models, a real-time testing phase was conducted to simulate the system's actual use. This phase aimed to determine appropriate decision thresholds for both spoofing detection and identity recognition.

For identity recognition, the system computes cosine similarity between the feature vector of the captured fingerprint and those in the database. Features are extracted from DenseNet121's DenseBlock3.

For spoofing detection, the model outputs a probability distribution using SoftMax, a function that converts raw output scores into probabilities. The highest probability determines the predicted class (real or fake), and a threshold on that probability is used to accept or reject the input as genuine.

Fingerprint Module Decision Threshold Determination:

Different decision threshold values were tested to determine the one that best balances spoof detection and identity verification. Each threshold was evaluated using real-time test cases.

The description and results of the tests conducted in the determination of an appropriate threshold are shown in the following format: the threshold value in each experiment is stated, followed by a statement summarizing each test and the image the system was tested with. Lastly, a table displays the results depending on the model, SoftMax Probability if spoofing and Cosine Similarity if identification, along with the system's decision.

Spoofing:

4.4.8.2 Experiment 1: Threshold 0.80:

A decision threshold of 0.80 was tested using both real and fake images.

1. Test 1: Spoofed fingerprint using tape

Figure 4.35: Spoofed Fingerprint Image using Tape

Table 4.19: SoftMax Output and Prediction Result - Test 1, Experiment 1 (Real-time Testing)

SoftMax Probability	Prediction Result using Threshold 0.80
0.9901	Spoofing Detected

2. Test 2: Spoofed fingerprint using clay

Person A's fingerprint was imprinted on a clay model and Person B used it to attempt to spoof the system.

Figure 4.36: Spoofed Fingerprint Image using Clay

Table 4.20: SoftMax Output and Prediction Result - Test 2, Experiment 1 (Real-time Testing)

SoftMax Probability	Prediction Result using Threshold 0.80
0.9957	Spoofing Detected

3. Test 3: Spoofed fingerprint using glue

A glue-based fingerprint replica of Person A was created and Person B used it to attempt to spoof the system.

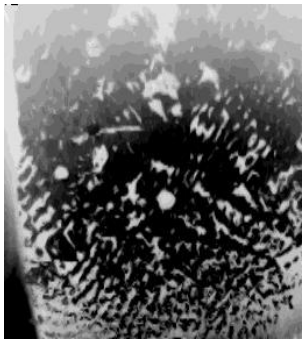

Figure 4.37: Spoofed Fingerprint Image using Glue

Table 4.21: SoftMax Output and Prediction Result - Test 3, Experiment 1 (Real-time Testing)

SoftMax Probability	Prediction Result using Threshold 0.80
0.9988	Spoofing Detected

4. Test 4: Software-generated fingerprint

A software-generated fingerprint using SFinGe [46]; a software used to create synthetic fingerprints.

Figure 4.38: Software-generated Fingerprint Image using SFinGe [46]

Table 4.22: SoftMax Output and Prediction Result - Test 4, Experiment 1 (Real-time Testing)

SoftMax Probability	Prediction Result using Threshold 0.80
0.9960	Spoofing Detected

5. Test 5: A real fingerprint image captured using R307

Figure 4.39: Real Fingerprint Image Captured using R307

Table 4.23: SoftMax Output and Prediction Result - Test 5, Experiment 1 (Real-time Testing)

SoftMax Probability	Prediction Result using Threshold 0.80
0.12	No Spoofing Detected

Through real-time testing, a threshold value of 0.8 consistently yielded correct classifications for both genuine fingerprints and all types of spoofing attempts. Although the spoofing attempts shown here are detected with a 0.99 SoftMax probability, the threshold should remain lower to ensure detection of stronger spoofing attacks or unseen spoofing materials.

Identification

The identification decision threshold needs to be high enough to detect impersonation attempts between similar looking individuals, but low enough to ensure that a genuine access attempt is not interrupted. Although this balance is important to prevent inconveniencing genuine travelers, in the context of a border crossing, security is of higher importance than convenience.

4.4.8.3 Experiment 1: Threshold 0.94:

1. Test 1: Person E attempting to falsely identify as Person B.

Figure 4.40: Person E's Fingerprint (Left) and Person B's Fingerprint (Right)

Table 4.26: Matching Scores and System Decision - Test 1, Experiment 1 (Real-time Testing)

Similarity Score	Result using Threshold 0.94
0.9417	Identity Verified

A threshold of 0.94 also resulted in a verification error, allowing Person E to impersonate Person B. Since the similarity between Persons B and E is 0.9417, the decision threshold needs to be raised.

4.4.8.4 Experiment 2: Threshold 0.95:

1. Test 1 (Impersonation): Person D attempting to falsely identify as Person C.

Figure 4.41: Person D's Fingerprint (Left) and Person C's Fingerprint (Right)

Table 4.25: Matching Scores and System Decision - Test 1, Experiment 2 (Real-time Testing)

Similarity Score	Result using Threshold 0.95
0.9454	Identity Mismatch

Threshold 0.95 was able to detect the impersonation attempt. It needs to also be tested with a genuine user to ensure that it does not mistakenly reject legitimate access.

2. Test 2 (Genuine): Person A identifying as themselves.

Figure 4.42: Two Captured Fingerprint Images of Person A

The fingerprint image on the left is the newly captured one of Person A. However, the one on the right is whose feature vector is stored in the database. The cosine similarity between the newly captured image and the stored image is found and shown in Table 4.27.

Table 4.27: Matching Scores and System Decision - Test 2, Experiment 2 (Real-time Testing)

Similarity Score	Result using Threshold 0.95
0.9634	Identity Verified

3. Test 3 (Genuine): Person B identifying as themselves.

Figure 4.43: Two Captured Fingerprint Images of Person B

The results shown in Table 4.28 are of an identical test to Test 2 above but with a different user to check the threshold's correctness.

Table 4.28: Matching Scores and System Decision - Test 3, Experiment 2 (Real-time Testing)

Similarity Score	**Result using Threshold 0.95**
0.9952	Identity Verified

At a decision threshold of 0.95, the system correctly flagged impersonation attempts and allowed legitimate users to be identified as themselves.

Face Module Decision Threshold Determination:

Spoofing

4.4.8.5 Experiment 1: Threshold 0.80:

A decision threshold of 0.80 was tested on real and fake images.

1. Test 1: Spoofed face image from a passport.

Figure 4.44: ID Spoofed Face Image Detection (Left) and Captured Face Image (Right)

Table 4.29: SoftMax Output and Prediction Result - Test 1, Experiment 1 (Real-time Testing)

SoftMax Probability	Prediction Result using Threshold 0.80
1.00	Spoofing Detected

2. Test 2: Uploaded face image

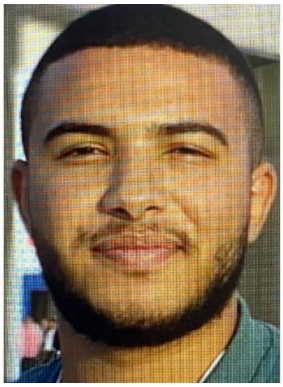

Figure 4.45: Uploaded Headshot (Spoofing)

Table 4.30: SoftMax Output and Prediction Result - Test 2, Experiment 1 (Real-time Testing)

SoftMax Probability	Prediction Result using Threshold 0.80
0.9994	Spoofing Detected

3. Test 3: Spoofed face image using a 4*3 headshot

*Figure 4.46: 4*3 Headshot (Spoofed)*

Table 4.31: SoftMax Output and Prediction Result - Test 3, Experiment 1 (Real-time Testing)

SoftMax Probability	Prediction Result using Threshold 0.80
0.9146	Spoofing Detected

4. Test 4: Genuine face image captured using the webcam

Figure 4.47: Genuine Face Image

Table 4.32: SoftMax Output and Prediction Result - Test 4, Experiment 1 (Real-time Testing)

SoftMax Probability	Prediction Result using Threshold 0.80
0.03	No Spoofing Detected

A threshold value of 0.80 consistently yielded correct classifications for both genuine face images and three different attempts at spoofing. As previously mentioned when testing the spoofing threshold for fingerprints, although the SoftMax probabilities here were all above 0.90, the threshold must be maintained at a lower number to detect more sophisticated spoofing attempts.

Identification

4.4.8.6 Experiment 1: Threshold 0.95:

1. Test 1 (Impersonation): Person F attempting to falsely identify as Person E.

Figure 4.48: Person F (Left) and Person E (Right)

Table 4.33: Matching Scores and System Decision - Test 1, Experiment 1 (Real-time Testing)

Similarity Score	**Result using Threshold 0.95**
0.9728	Identity Verified

Testing threshold 0.95 permitted an impersonation attempt. The similarity score between Persons F and E is 0.9728, indicating the need for a higher decision threshold.

4.4.8.7 Experiment 2: Threshold 0.98:

1. Test 1 (Impersonation): Person D falsely identifying as Person F.

Figure 4.49: Person D (Left) and Person F (Right)

Table 4.34: Matching Scores and System Decision - Test 1, Experiment 2 (Real-time Testing)

Similarity Score	Result using Threshold 0.98
0.9771	Identity Mismatch

2. Test 2 (Genuine): Person D identifying as themselves

Figure 4.50: Two Captured Face Images of Person D

This is the same idea as Test 2 above in the fingerprint identification threshold experiment i.e. a person's newly captured image is compared with his/her saved image.

Table 4.35: Matching Scores and System Decision - Test 2, Experiment 2 (Real-time Testing)

Similarity Score	Result using Threshold 0.98
0.9925	Identity Verified

3. Test 3 (Genuine): Person G identifying as themselves

Figure 4.51: Two Captured Face Images of Person G

This is a similar test to Test 2 above but with another user to validate the threshold selected.

Table 4.36: Matching Scores and System Decision - Test 3, Experiment 2 (Real-time Testing)

Similarity Score	Result using Threshold 0.98
0.9940	Identity Verified

A threshold of 0.98 correctly flagged impersonation attempts and allowed legitimate users to be identified as themselves.

To conclude, chapter 4 presented the complete implementation and testing process of the proposed system. It began by detailing the datasets and technologies utilized throughout the project. The chapter then outlined the step-by-step development of each component until a final solution was reached. These components were fingerprint acquisition, enhancement, identification and spoofing models of the fingerprint module. As well as face detection, capturing, recognition and spoofing model of the face module. Feature extraction and enrolment were shown once for both modules. Finally, the chapter concluded with real-time testing scenarios that validated the system's functionality and effectiveness under realistic conditions.

Chapter 5 User Manual

This chapter is a walkthrough of the whole application. By following this guide, users and admins will be able to navigate and utilize all features of the system effectively.

5.1 Traveler Flow: Normal Verification Flow

5.1.1 Home Screen

This is the first screen displayed upon launching the system (Figure 5.1). It provides two main options for the user:

- **Watch**: Plays a recorded demonstration of the system's functionality, guiding users through the identity verification steps.
- **Upload Your Passport**: Initiates the full identity verification process.

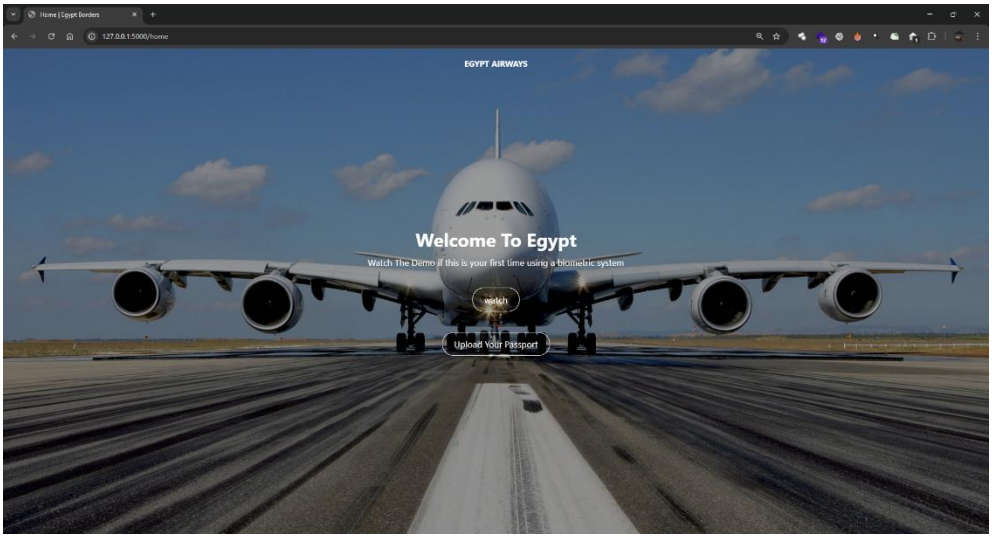

Figure 5.1: Home Screen

5.1.2 Demo Playback Page

After clicking "**Watch**" on the home screen, the system shows a pop-up window of the recorded video with a "**Close**" button below, as seen in Figure 5.2.

The video player allows the user to:

- Navigate through the video timeline.
- Adjust playback speed.
- Play the video in full screen mode.
- Control the volume.

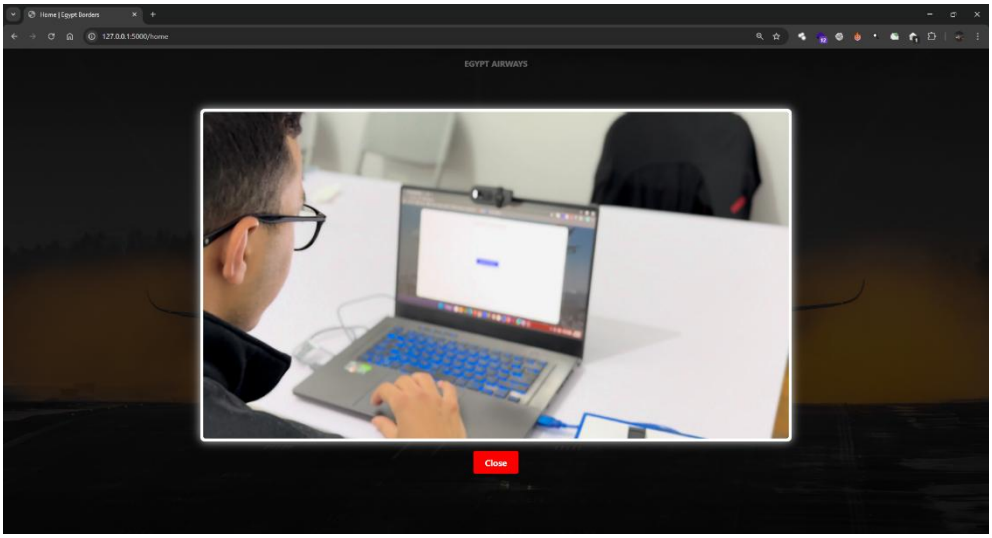

Figure 5.2: Demo Video Player

5.1.3 Upload Your Passport

The first step in the identity verification process. A screen is showed with a "**Upload Passport**" button, that when clicked, allows the user to select their passport image from a folder and upload it. After the successful upload of the passport, the OCR module extracts the necessary information from the image and displays the traveler's name. The process is illustrated in Figures 5.3, 5.4, and 5.5.

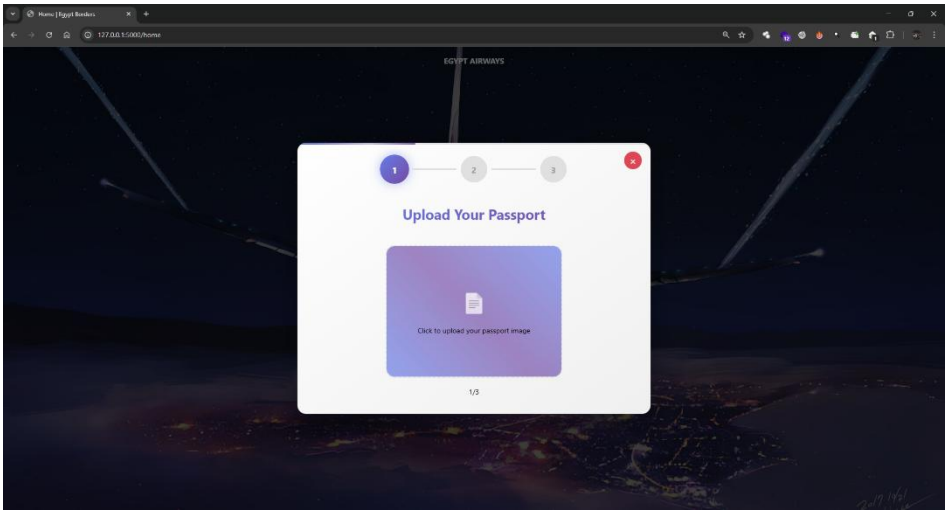

Figure 5.3: Upload Passport

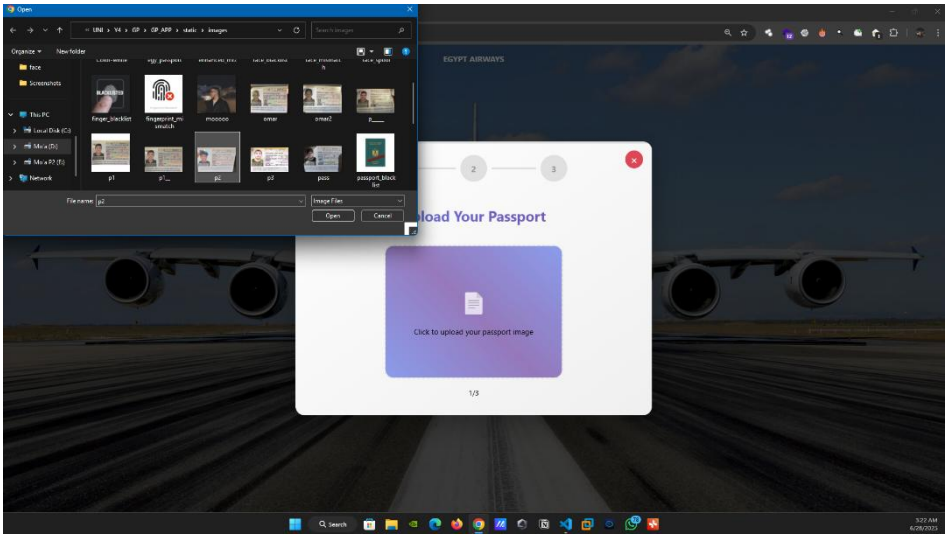

Figure 5.4: Passport Image Selection

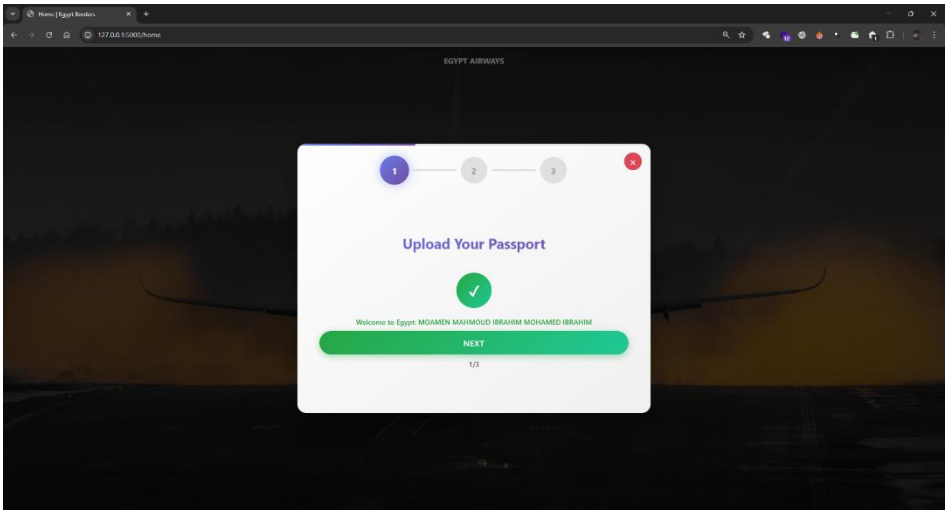

Figure 5.5: Traveler Name Displayed after Passport Upload

5.1.4 Fingerprint Scanning

Once the traveler clicks on "**Next**" in the previous screen, he/she is then prompted to scan their fingerprint. Once the traveler clicks on "**Start Fingerprint Scan**", the sensor is activated and capturing begins. If the fingerprint matches the traveler's identity, the captured image is shown on the screen. The flow of a successful traveler's fingerprint scan is shown in Figures 5.6 to 5.8.

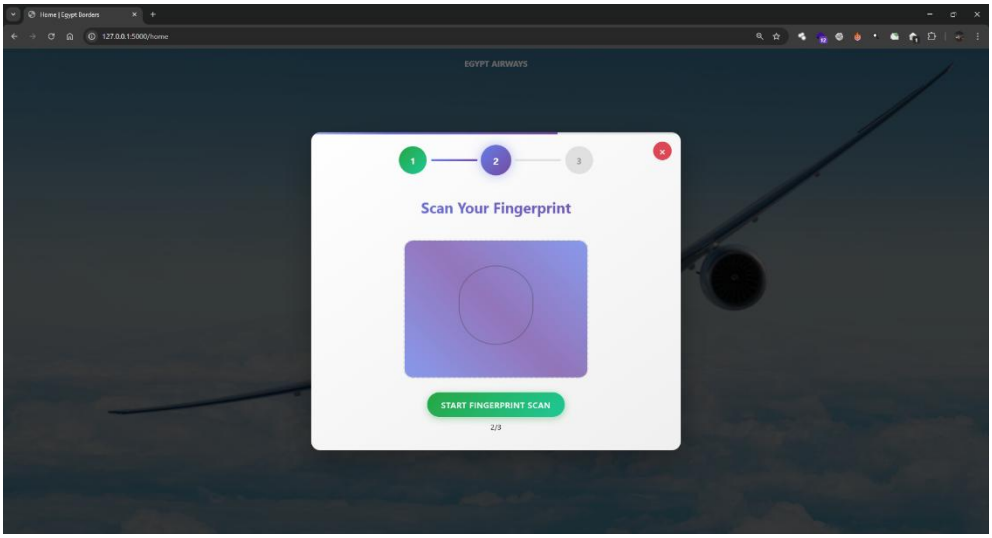

Figure 5.6: Fingerprint Scanning

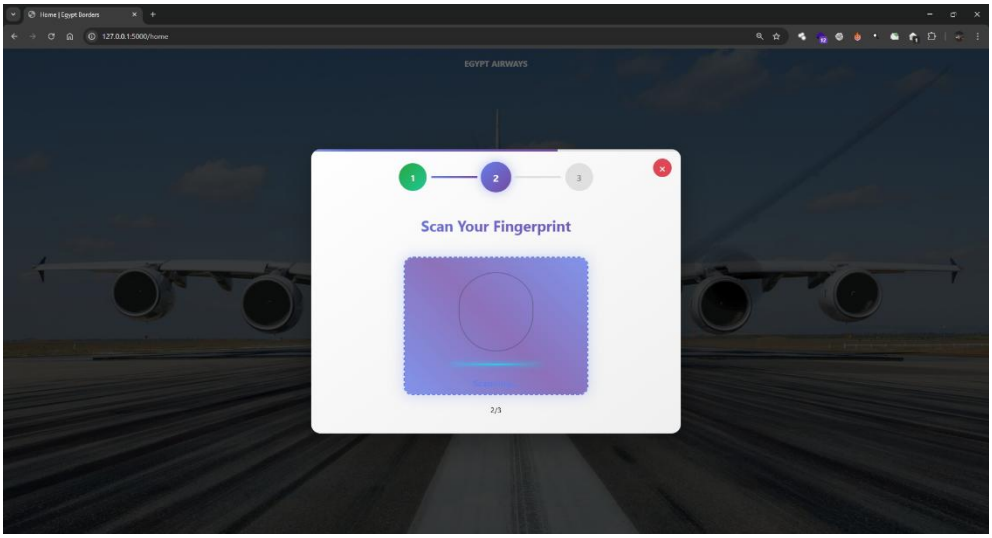

Figure 5.7: Fingerprint Scanning in Progress

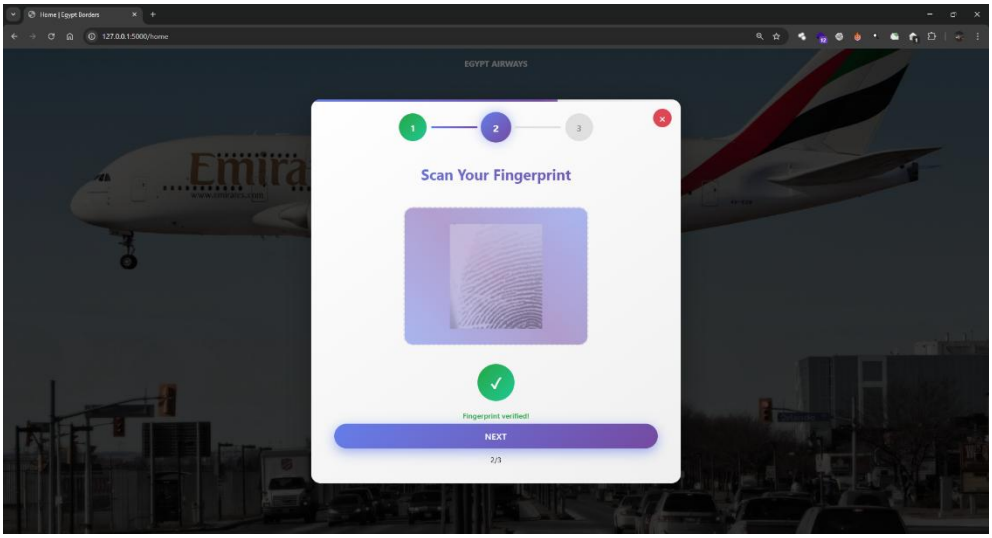

Figure 5.8: Captured Fingerprint During Verification Process

5.1.5 Face Scanning

After clicking "**Next**", the face scanning phase begins. The traveler is meant to center their face in the box appearing around their face. If the traveler's face is aligned, the box appears in green and displays a 5 second countdown before automatically capturing an image (Figure 5.9).

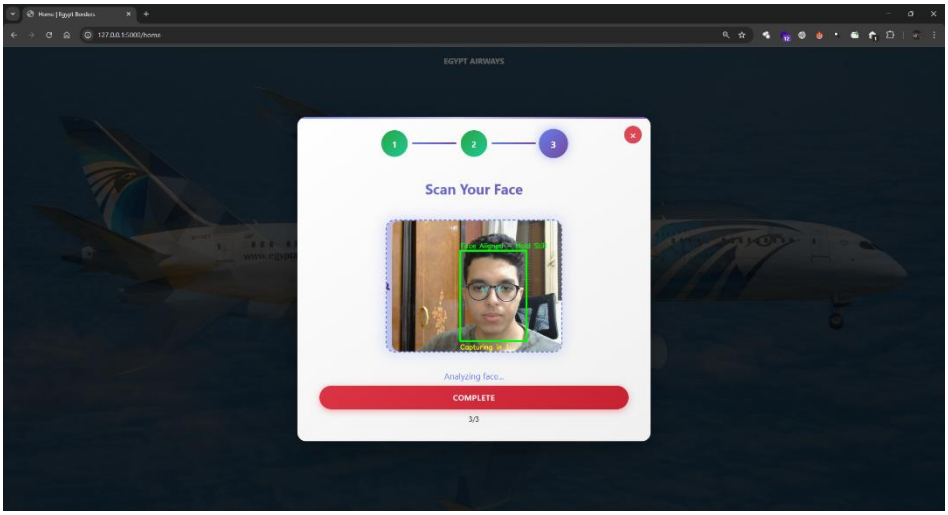

Figure 5.9: Face Scanning (Aligned Properly)

5.1.6 Identity Verification

If the process is followed correctly and the traveler is verified, a screen showing *"Welcome, <Traveler's Name>"* is displayed. This is seen in Figure 5.10.

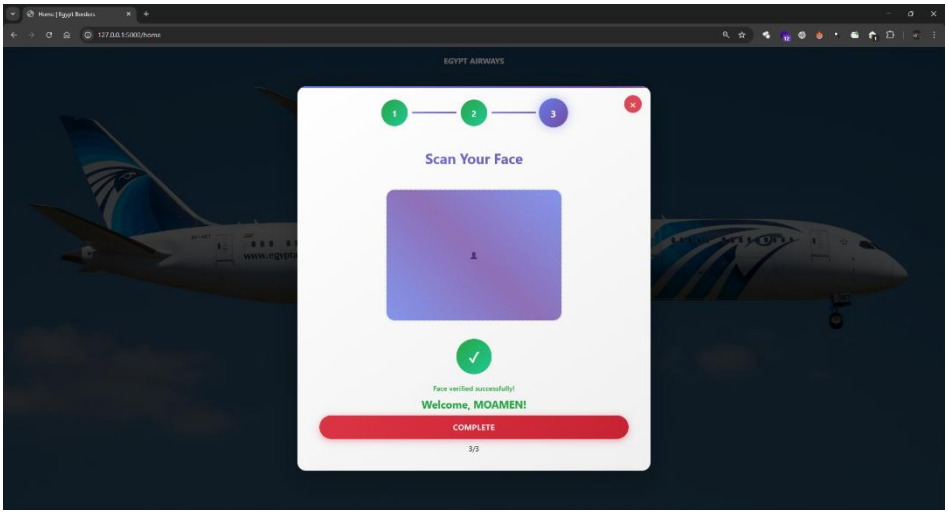

Figure 5.10: Successful Traveler Verification

5.2 Traveler Flow: Spoofing Attempts and Failed Verifications

5.2.1 Scenario 1: Passport Does Not Exist

When uploading the passport, a traveler might upload one that does not exist in the airport's database, meaning he/she needs to enroll with an officer. In this case, a message guiding the traveler to go to the officer is displayed (as seen in Figure 5.11).

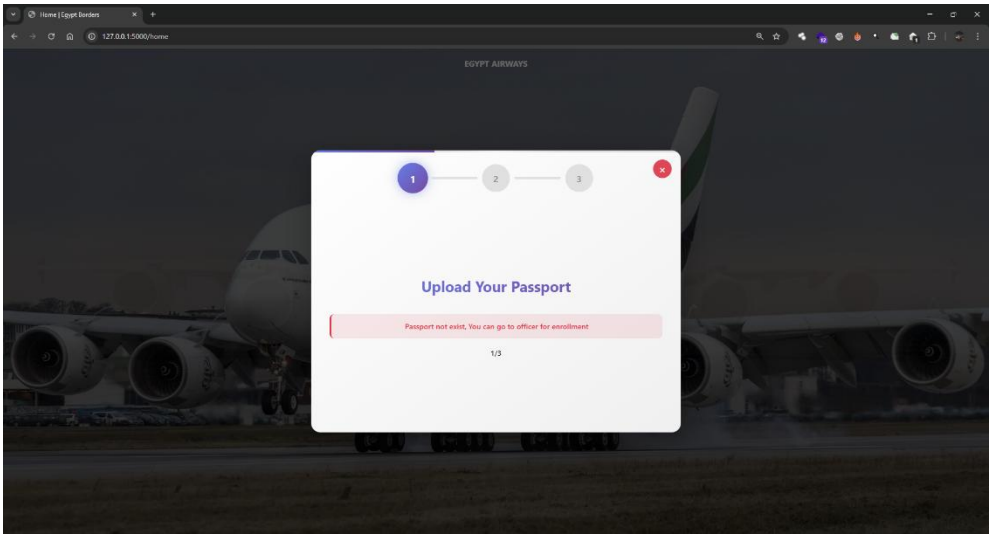

Figure 5.11: Uploading Unidentified Passport to the System Database

5.2.2 Scenario 2: Blacklisted Passport

A user might also upload a passport that belongs to a blacklisted identity. The system detects that and disrupts the flow (Figure 5.12).

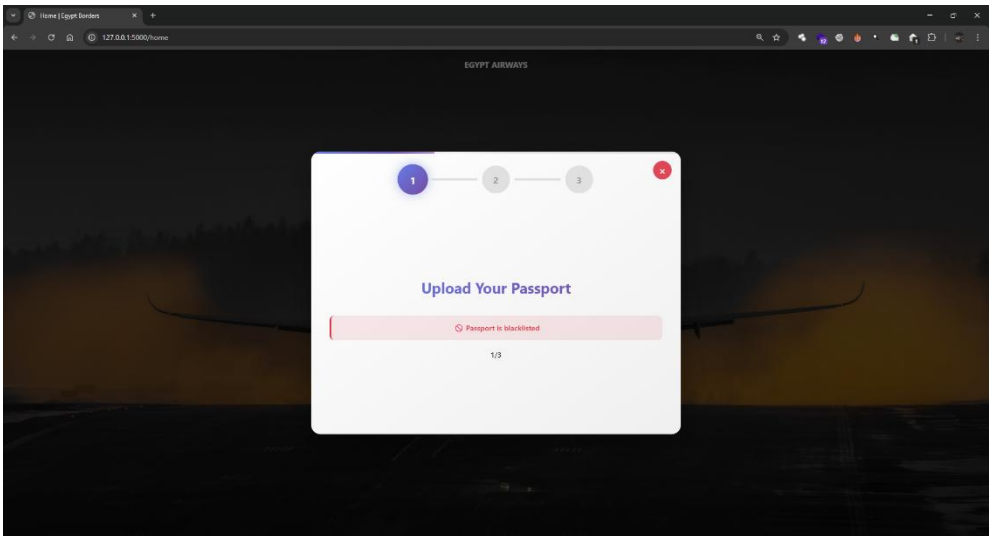

Figure 5.12: Blacklisted Passport Detected

Fingerprint Scanning

5.2.3 Scenario 3: Spoofed Fingerprint

The user might attempt to spoof the system using clay or tape around the finger. A clay-based fingerprint replica was created by pressing a real finger into soft

clay to leave a detailed impression of the fingerprint ridges. Once hardened, the imprint serves as a physical replica of the fingerprint pattern, which was then used in an attempt deceive the system during the liveness detection phase. This is shown in Figure 5.13.

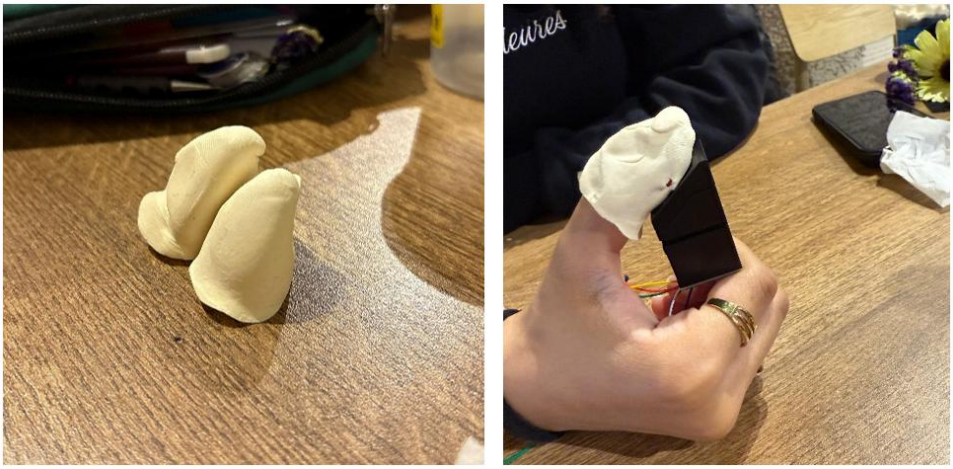

Figure 5.13: Fingerprint Spoofing Attempt using a Clay Impression of a Fingerprint

When a fingerprint spoofing attempt is detected, the system flow is disrupted and the text “Spoof Attempt Detected” is displayed, as seen in Figure 5.14.

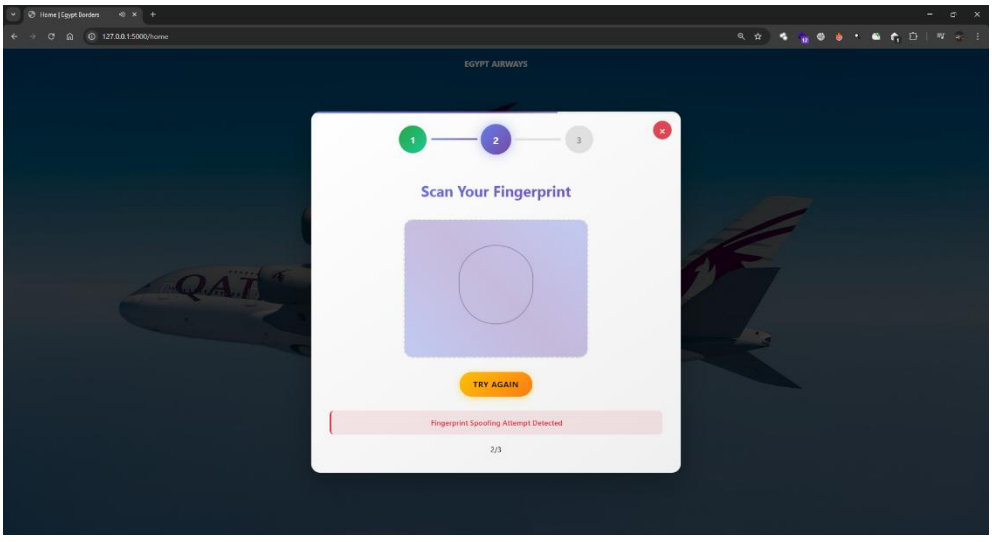

Figure 5.14: Fingerprint Spoofing Attempt Detected

5.2.4 Scenario 4: Fingerprint Identity Mismatch

The traveler could have uploaded the passport of a different identity and then attempt to gain access using his/her own biometric features. Since the system compares the captured biometric features with those of the user ID read from the passport, it detects an Identity Mismatch (Figure 5.15).

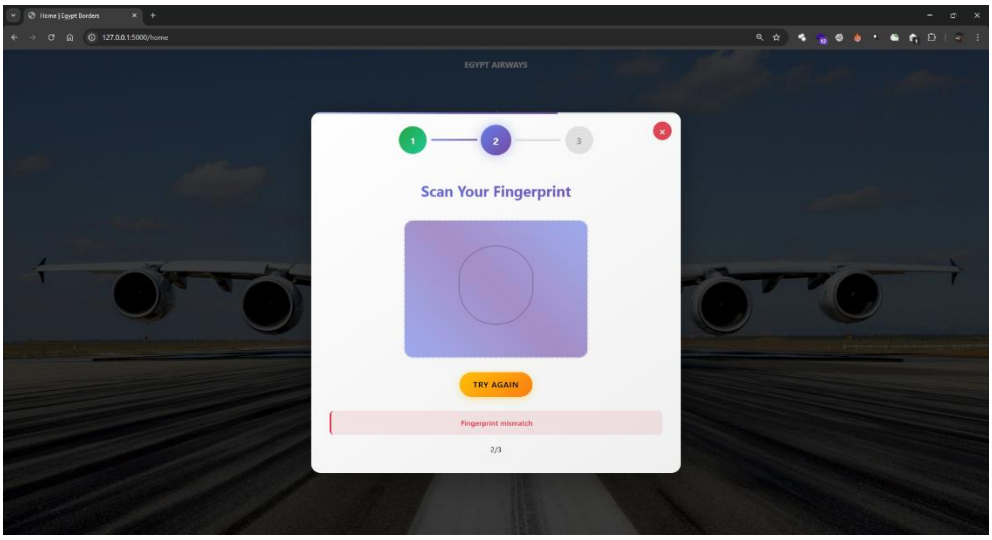

Figure 5.15: Fingerprint Identity Mismatch Detected

Face Scanning

5.2.5 Scenario 5: Misaligned Face

If the traveler is not directly facing the camera during face detection, a red box appears around their face with the text “Face Misaligned – Adjust Position” (shown in Figure 5.16). This box remains until the traveler aligns their face and the screen is as seen in Figure 5.9.

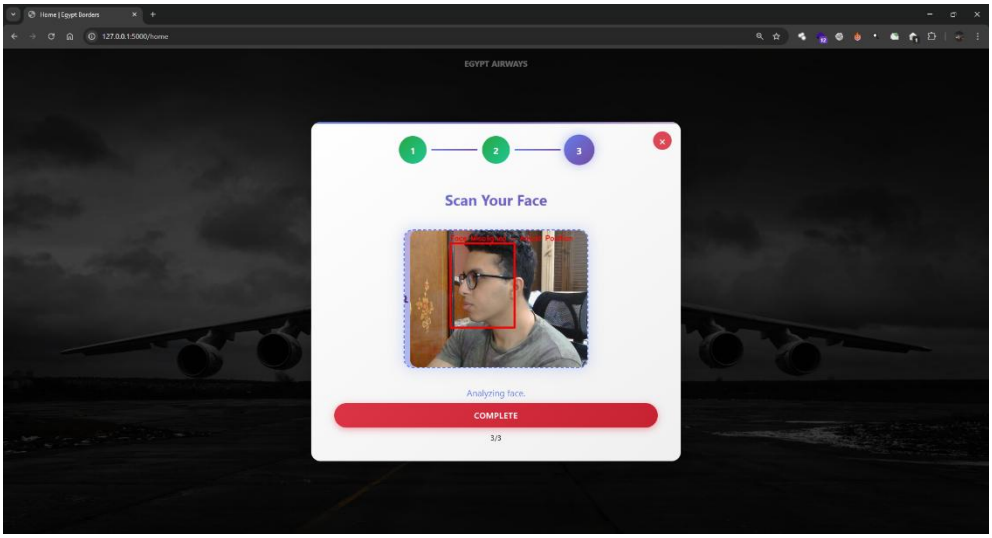

Figure 5.16: Misaligned Face Scanning

5.2.6 Scenario 6: Spoofed Face

The user might attempt to spoof the system by scanning the face picture on an ID card or a portrait photo. When a face spoofing attempt is detected, the system flow is disrupted. This scenario is shown in Figures 5.17 and 5.18.

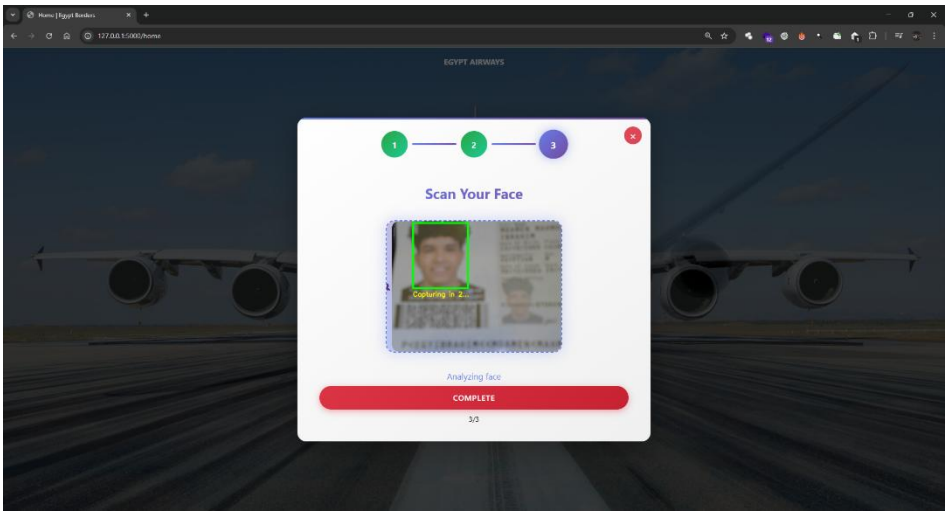

Figure 5.17: Face Spoofing Attempt using an ID

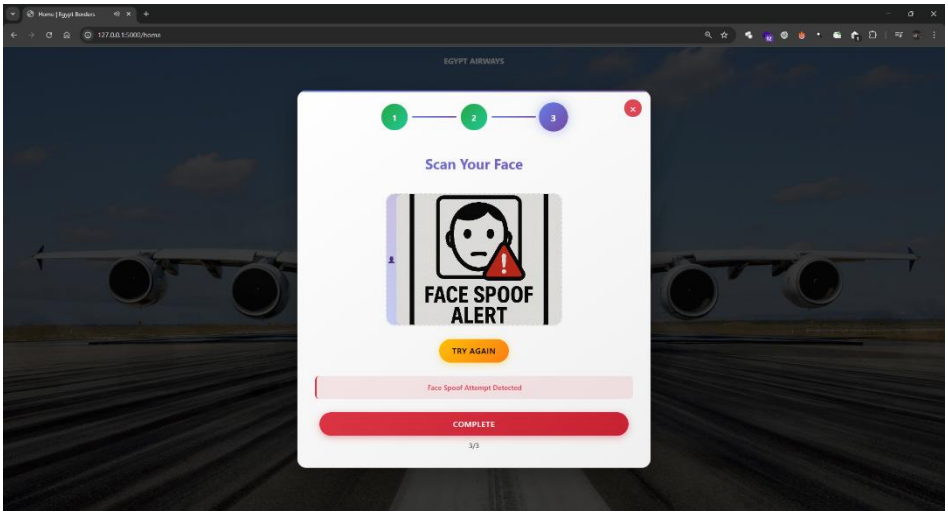

Figure 5.18: Face Spoofing Attempt Detected

5.2.7 Scenario 7: Face Identity Mismatch

The traveler could have uploaded the passport of a different identity and then attempt to proceed using his/her own biometric features. However, since the system compares the captured biometric features with those of the user ID read from the passport, it detects an Identity Mismatch. The scanning of the traveler's face followed by an Identity Mismatch detection is shown in Figures 5.19 and 5.20.

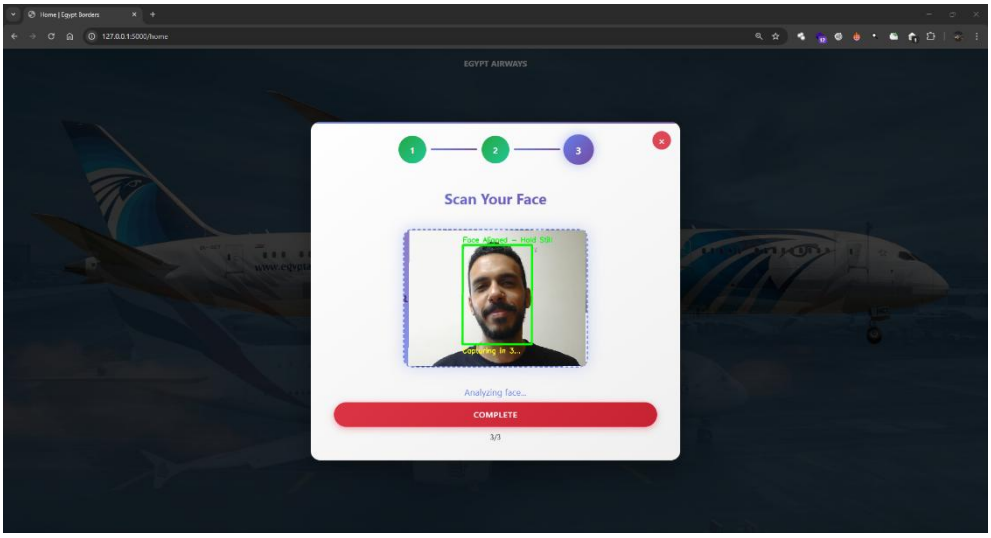

Figure 5.19: Face Identity Mismatch

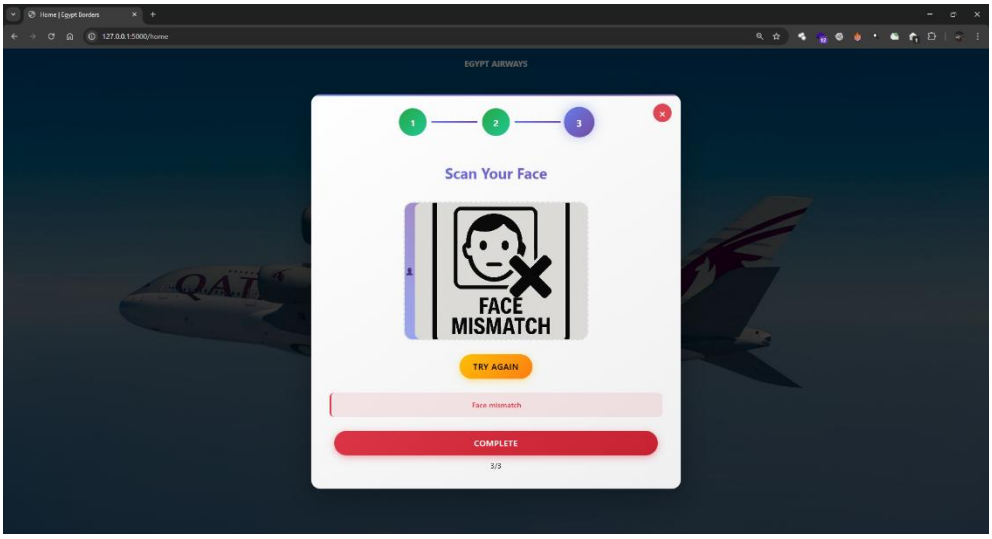

Figure 5.20: Face Identity Mismatch Detected

5.3 Admin/ Officer Flow

5.3.1 Officer Login

5.3.1.1 Credentials

The officer beings by logging in using his/her credentials; username and password. Figure 5.21 shows the admin login screen with credentials filled.

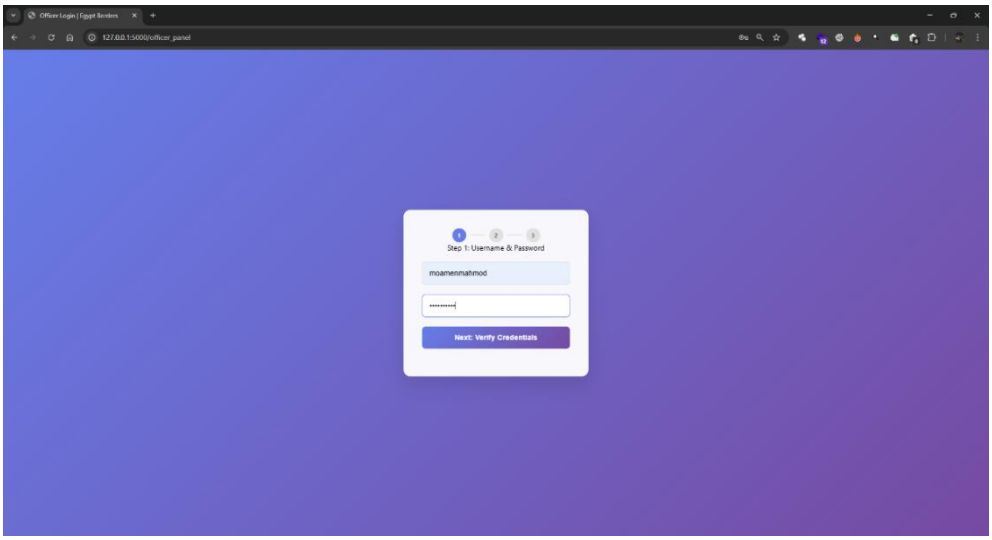

Figure 5.21: Officer Login Screen - Credentials Filled

5.3.1.2 Officer Biometric Sign-in: Fingerprint

Given that this project implements MFA, fingerprint biometric authentication is the second step of an officer's log-in process. The flow of scanning an officer's fingerprint is illustrated in Figures 5.22, 5.23 and 5.24.

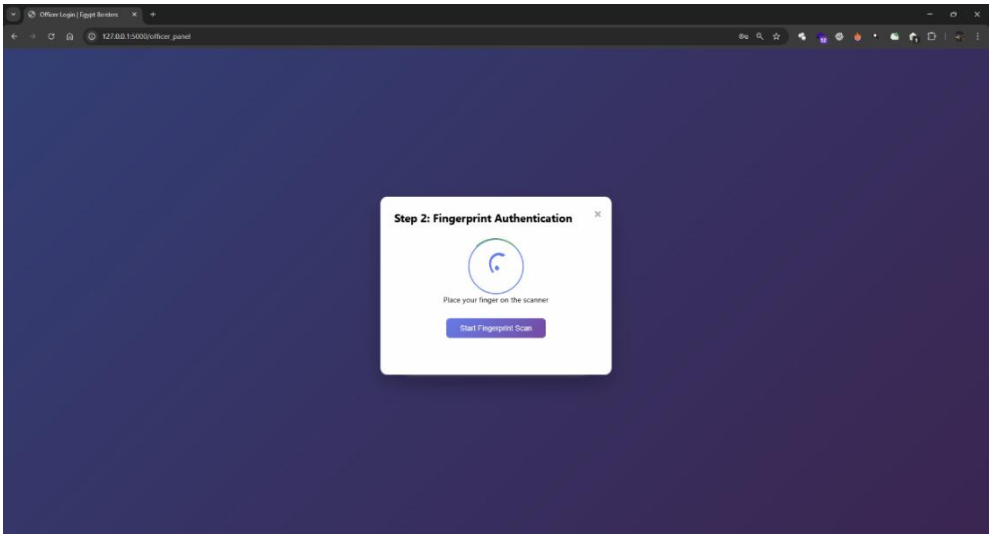

Figure 5.22: Officer Login Screen - Fingerprint Scan

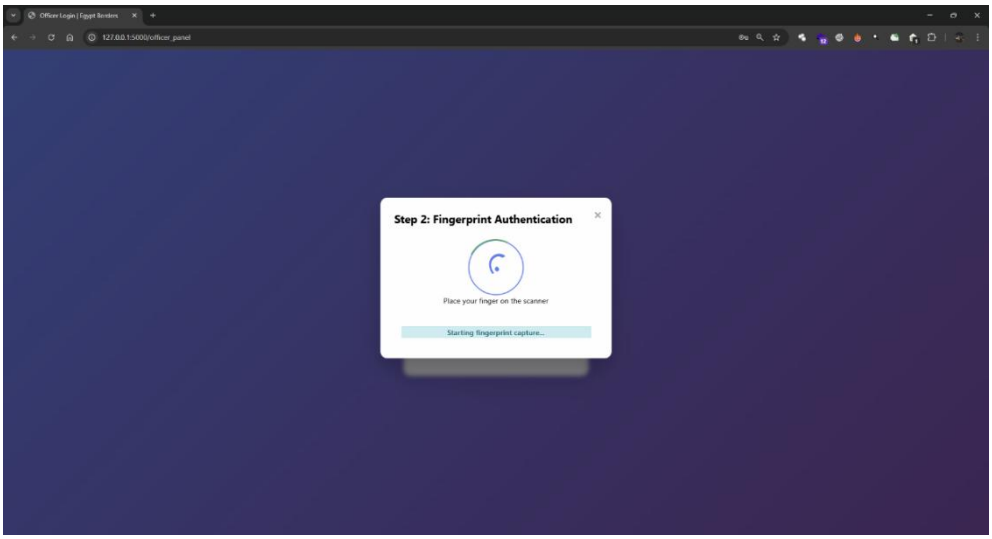

Figure 5.23: Officer Login Screen - Fingerprint Scanning in Progress

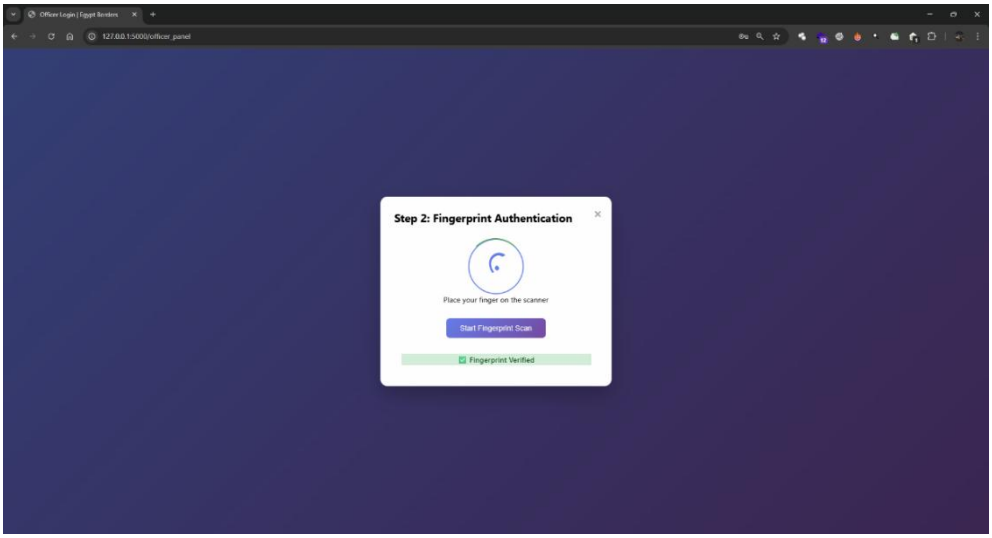

Figure 5.24: Officer Login Screen - Fingerprint Verified

The process of verifying an officer's fingerprint is the same as the one for verifying a traveler crossing the border; a newly captured feature vector compared with the stored one.

5.3.1.3 Officer Biometric Log-in: Face

Next is the authentication of the officer by the verification of his/her face ID. The scanning and verification of an officer's face is shown in Figures 5.25, 5.26 and 5.27.

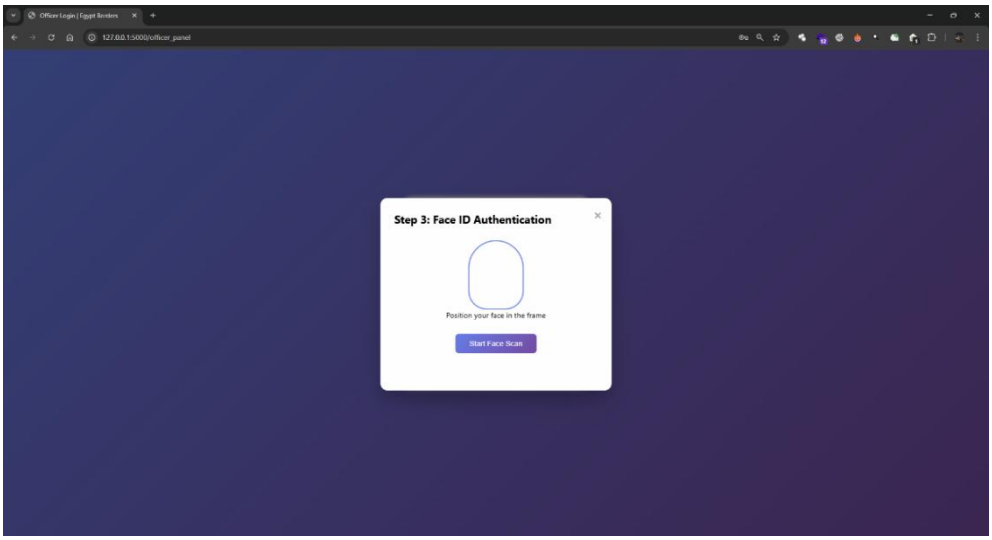

Figure 5.25: Officer Login Screen - Face ID Scan

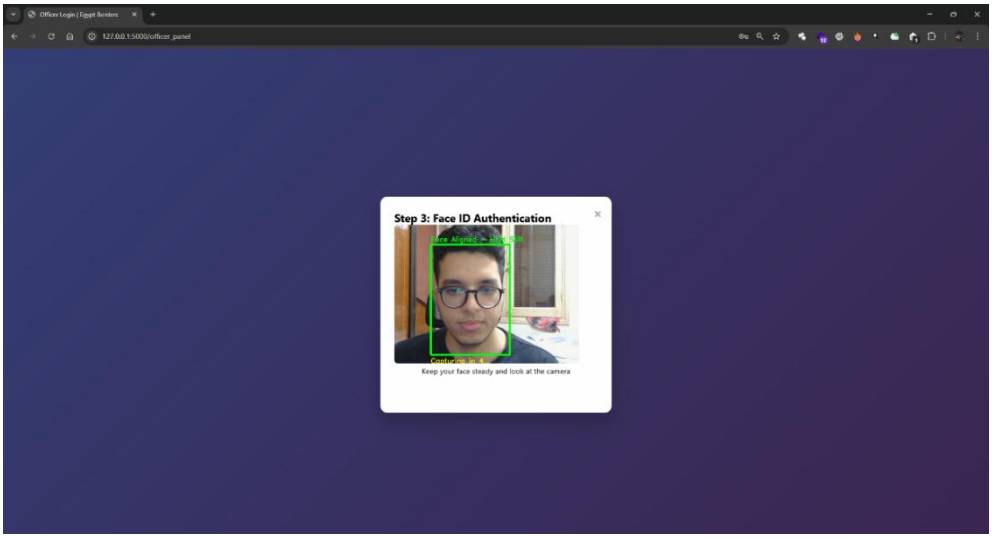

Figure 5.26: Officer Login Screen - Face ID Scanning in Progress

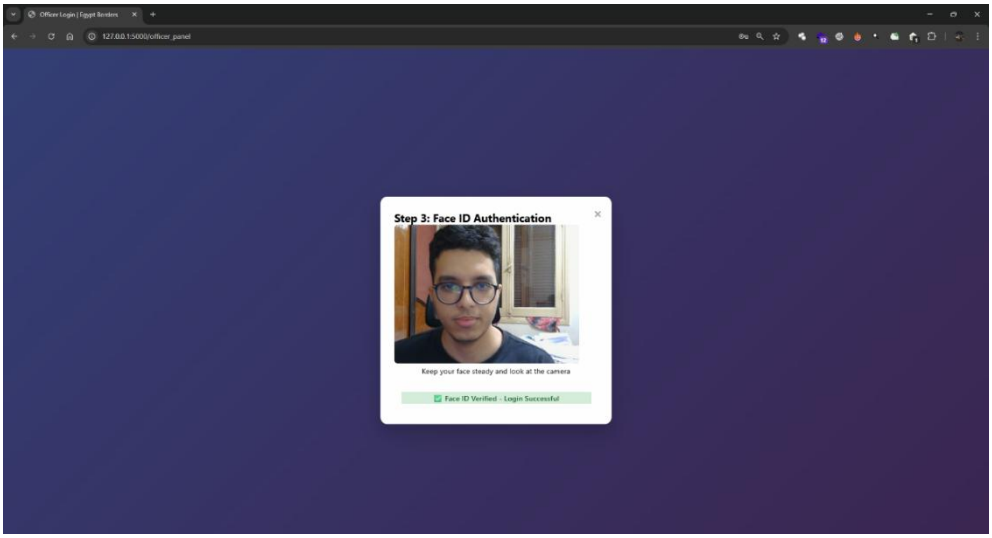

Figure 5.27: Officer Login Screen - Face ID Verified

5.3.2 Dashboard

This section will show the interface shown to the officer of the highest privilege: Manager. The differences in permissions are shown in Section 4.3.2 where certain actions are locked for officers of less access levels.

After the officer has logged in, he/she is greeted with a dashboard displaying the following:

- Total number of visits.
- Incidents today.
- Today's visits count.
- Number of blacklisted individuals attempting to cross.
- Recent alerts.
- Recent travelers.

The dashboard can be viewed in Figure 5.28.

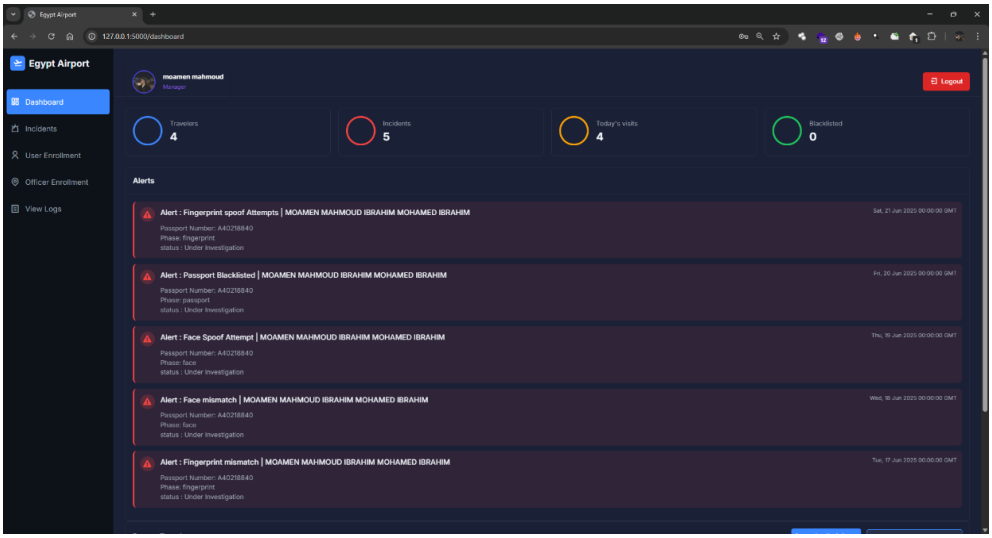

Figure 5.28: Officer Dashboard

5.3.2.1 Alert Viewing

The recent alerts and their details can be seen in this screenshot of the dashboard (Figure 5.29). Alert details include:

- Type.
- Traveler name.
- Traveler passport number.
- Phase.
- Status.

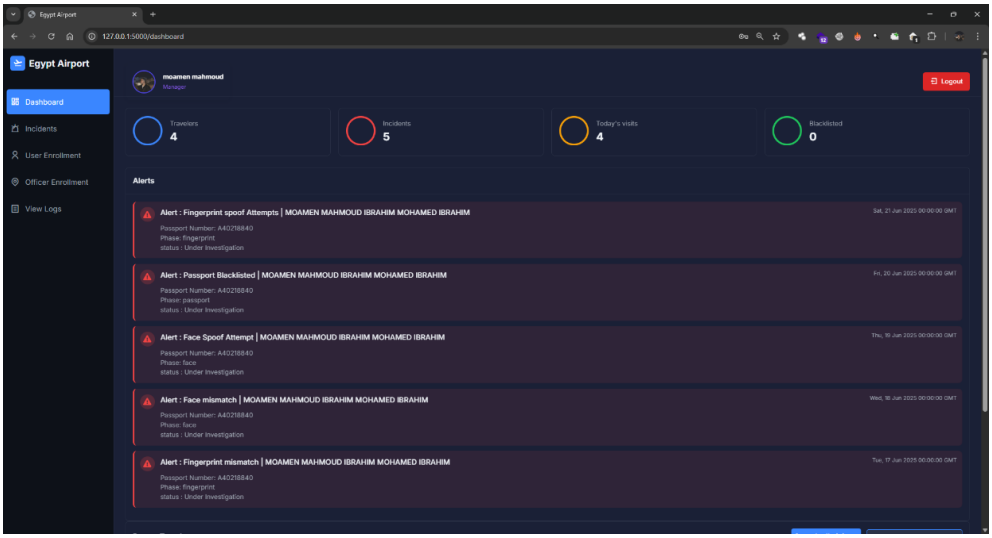

Figure 5.29: Officer Dashboard - Alerts

5.3.2.2 Recent Visits

Recent visits are listed, with the traveler's names, passport numbers and visit date and time displayed (Figure 5.30).

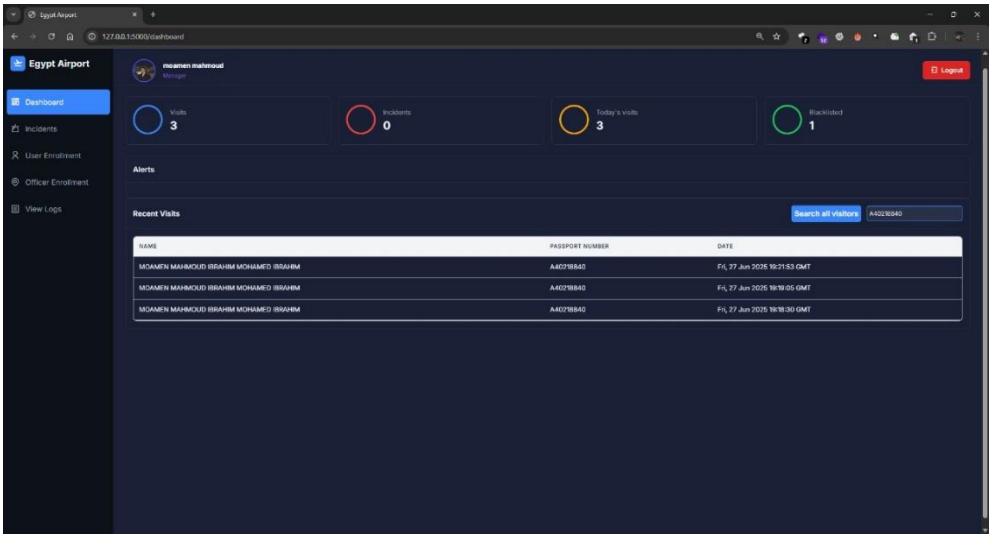

Figure 5.30: Officer Dashboard - Recent Visits

5.3.2.3 Search Visitors

Officers with the appropriate permission level can search for travelers using their passport numbers. Visitor search results for "A40218840" can be seen in Figure 5.31.

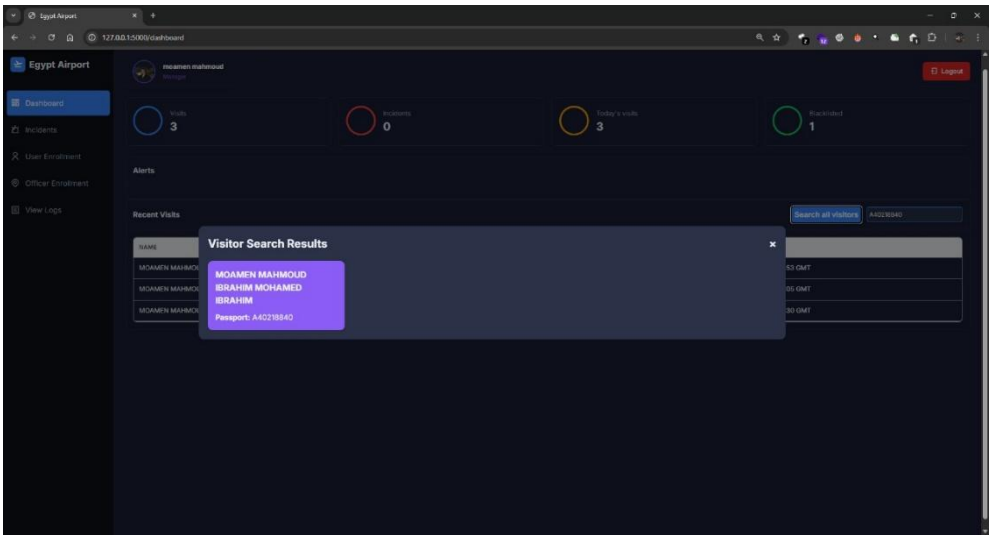

Figure 5.31: Officer Dashboard - Search Visitors

5.3.3 Incident Management

Authorized members of the staff can manage incidents (spoofing attempts, identity mismatches and blacklisted attempts). Incident management includes changing the status from “Under Investigation” to “Resolved” or vice versa. An authorized officer can also add/ remove users from the Blacklist. The complete set of options and actions is presented in Figures 5.32 - 5.35.

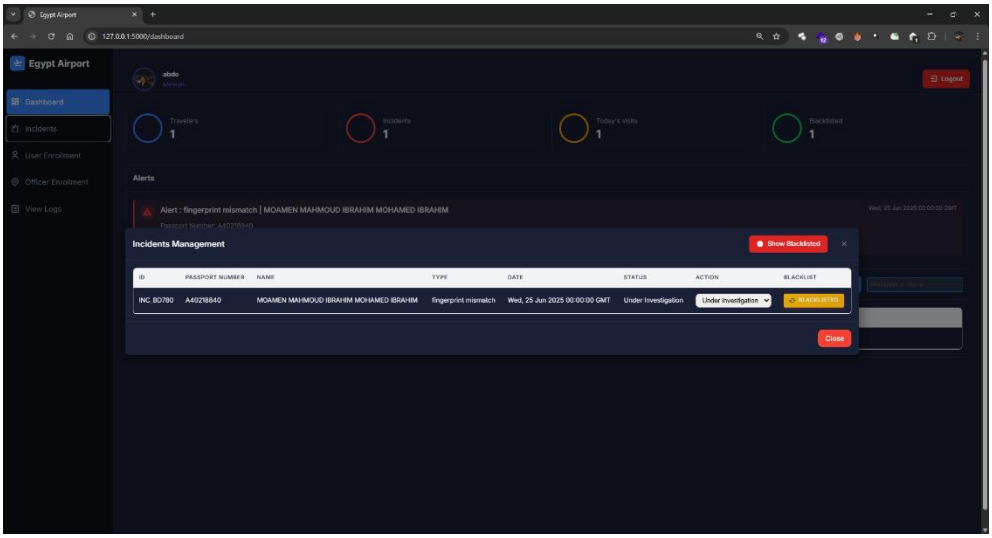

Figure 5.32: Officer Incident Management

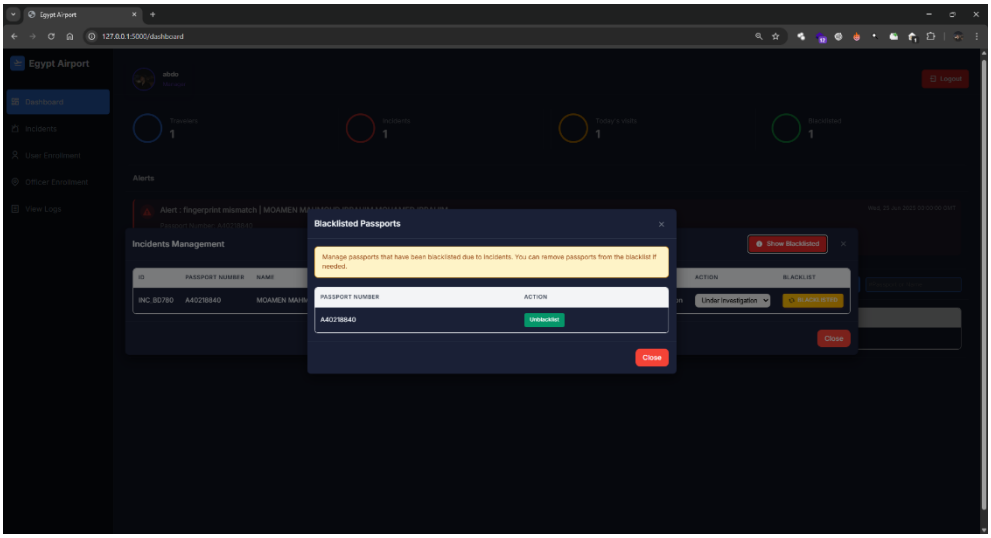

Figure 5.33: Officer Incident Management - Removal of a User from Blacklist

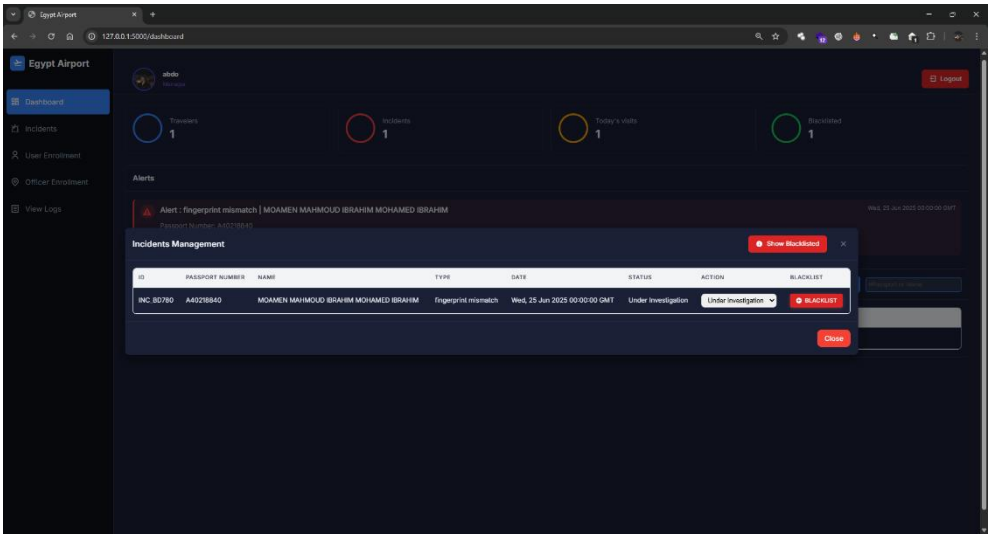

Figure 5.34: Officer Incident Management - Removal of a User from Blacklist Successful

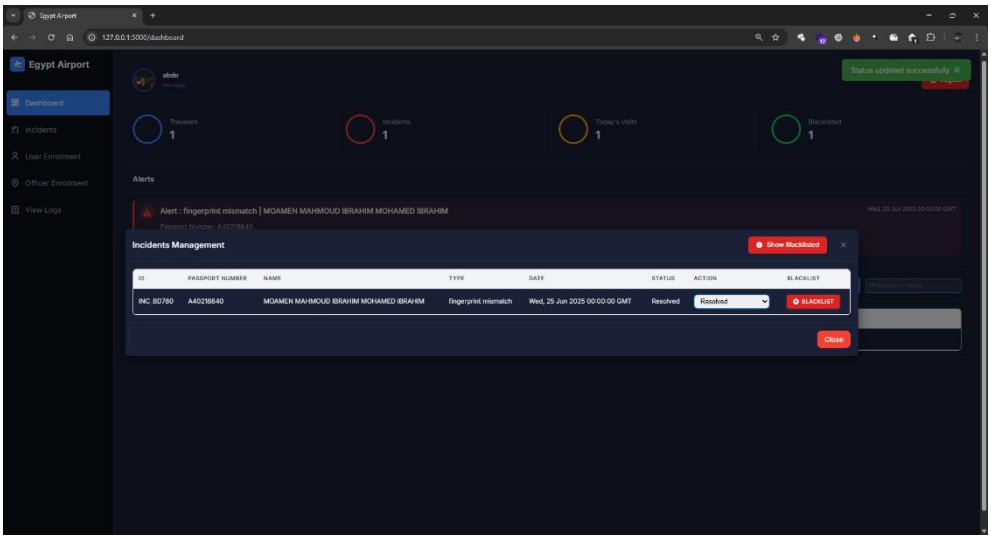

Figure 5.35: Officer Incident Management - Incident Status Updated

5.3.4 User Enrolment

5.3.4.1 Enrolling User's Personal Information

Enrolling a new (first time visitor) traveler to the system includes uploading his/her personal information, followed by his/her fingerprint and face. User's personal information is acquired through an OCR scan of a passport. The information extracted and uploaded is:

- Passport Number.
- Full name.
- Date of Birth.

- Nationality.
- Gender.
- Date of Expiry.

The enrolment of a user's passport is shown in Figures 5.36, 5.37, and 5.38.

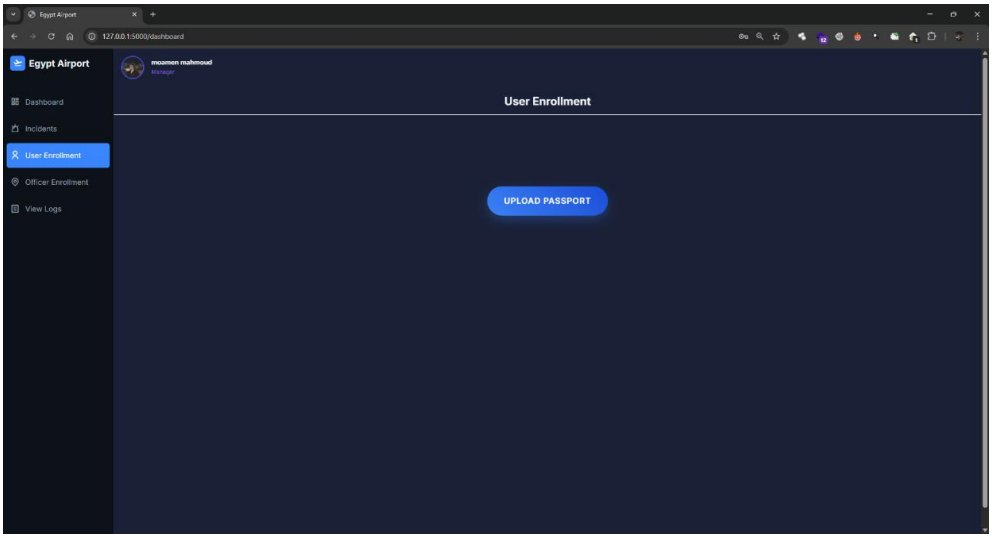

Figure 5.36: User Enrolment - Passport Upload

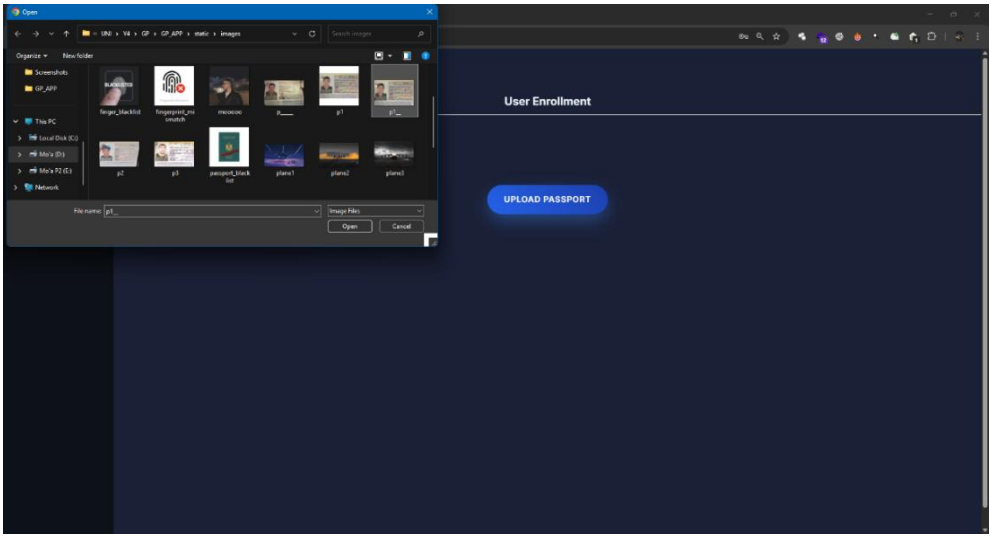

Figure 5.37: User Enrolment - Passport Image Selection

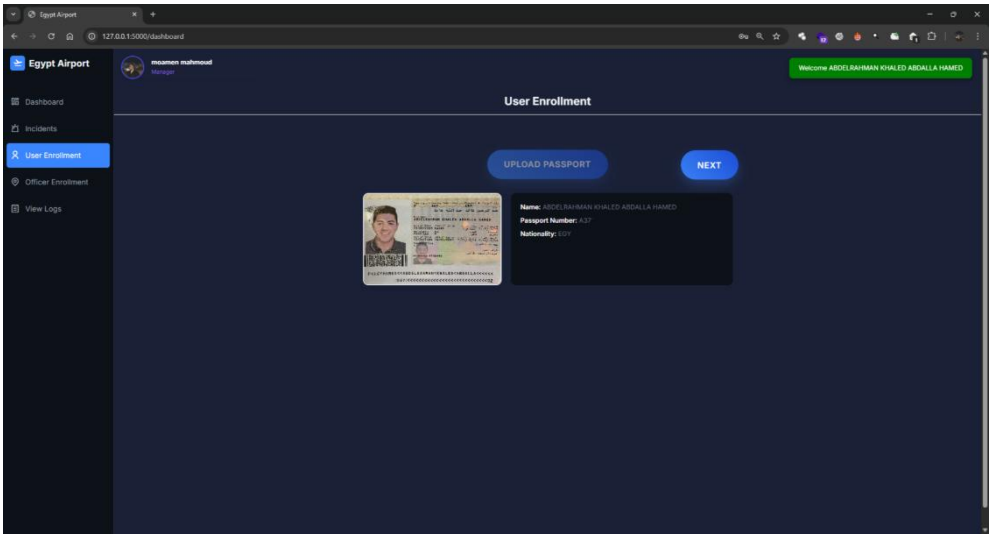

Figure 5.38: User Enrolment - Passport Upload Successful

5.3.4.2 Enrolling User's Fingerprint

Next is the enrolment of the traveler's biometric features, starting with the fingerprint. The complete enrolment of a traveler's fingerprint features is presented in Figures 5.39 - 5.41.

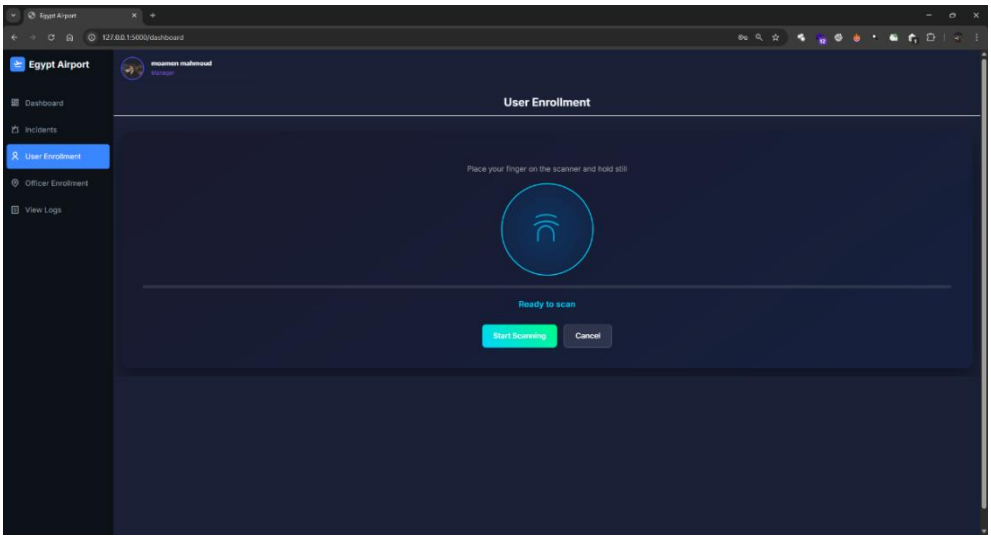

Figure 5.39: User Enrolment - Fingerprint Scan

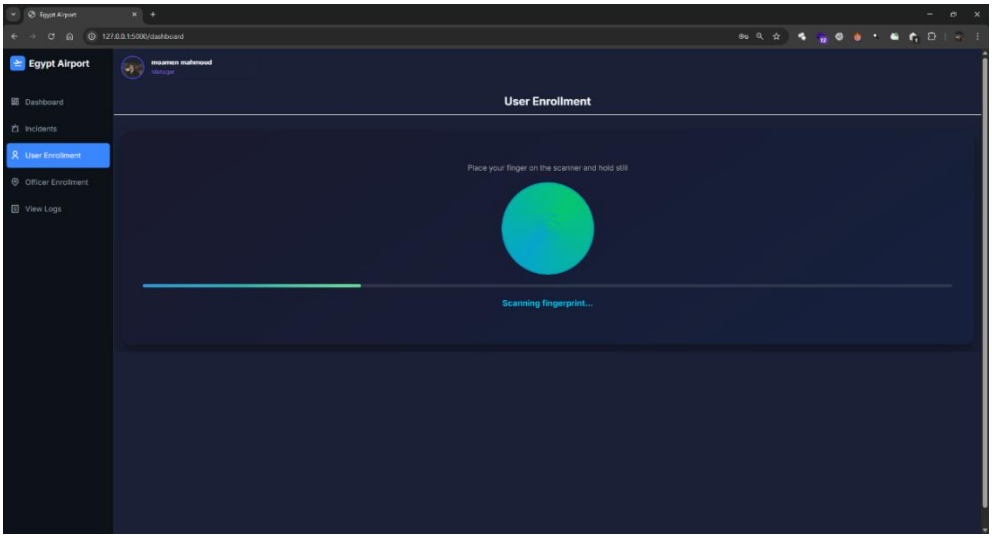

Figure 5.40: User Enrolment - Fingerprint Scanning in Progress

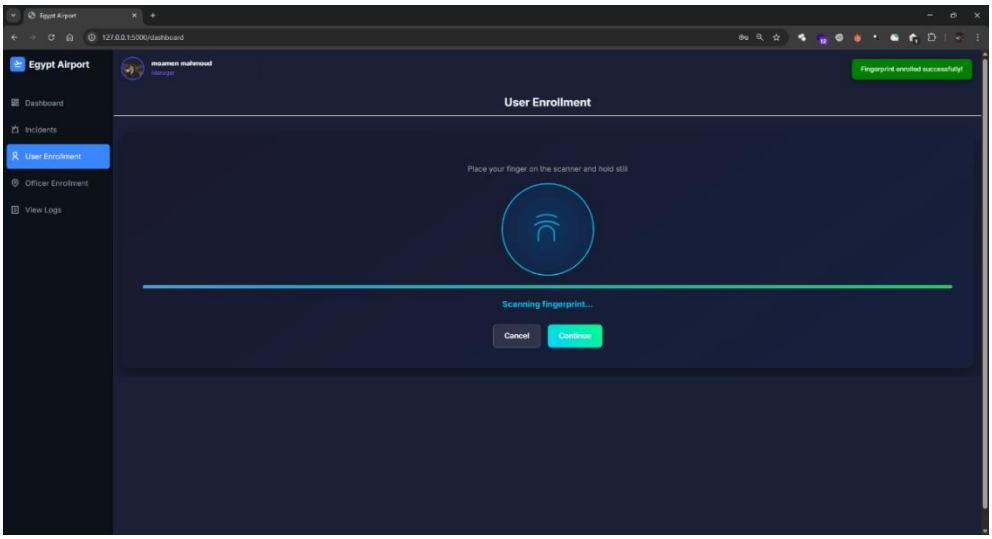

Figure 5.41: User Enrolment - Fingerprint Scan Complete

5.3.4.3 Enrolling User's Face

Finally, the user's face ID is captured and saved (Figures 5.42 - 5.44), this completes the User Enrolment process.

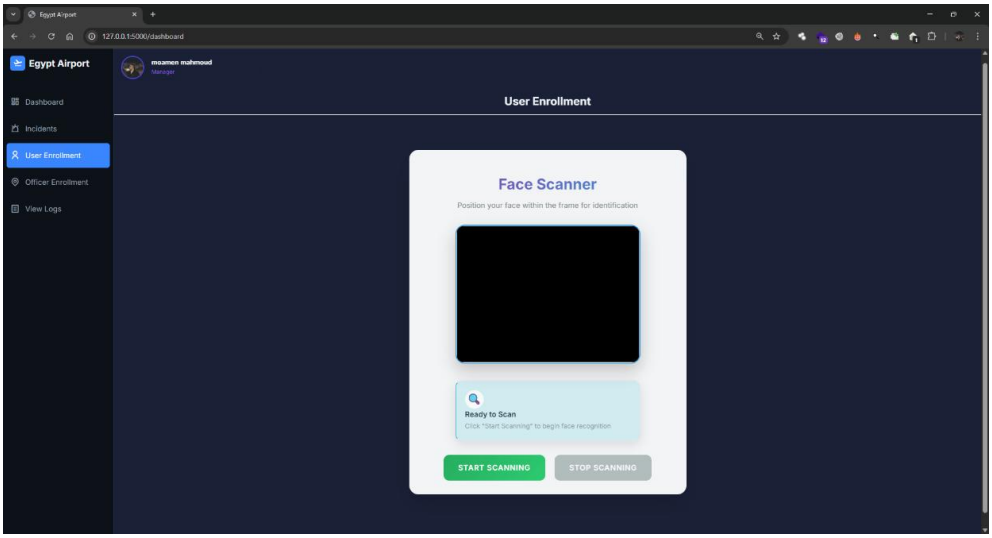

Figure 5.42: User Enrolment - Face Scan

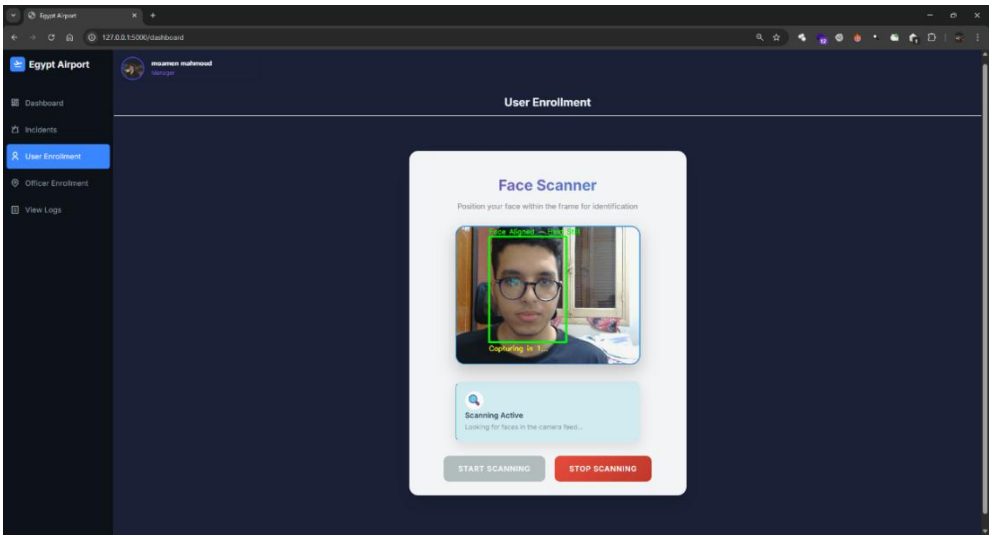

Figure 5.43: User Enrolment - Face Scanning in Progress

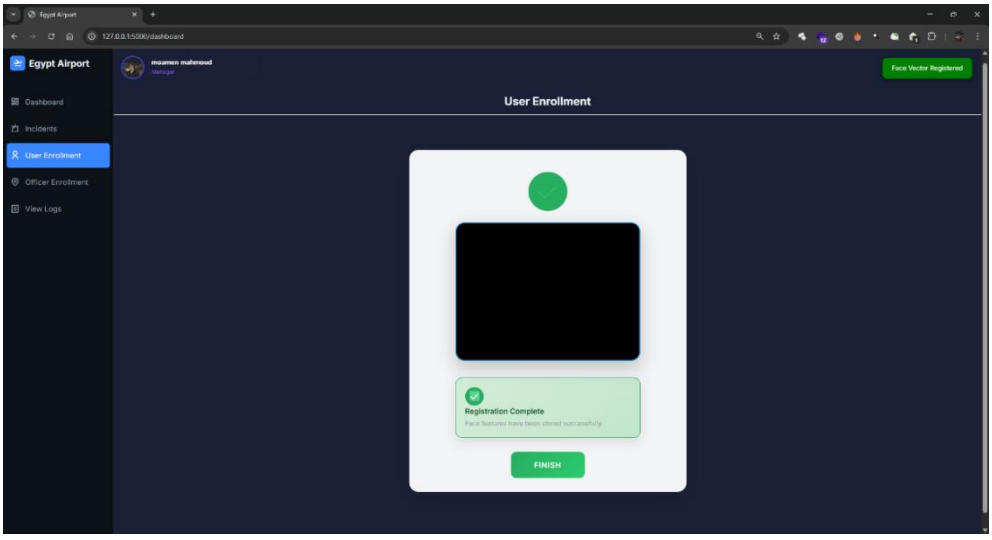

Figure 5.44: User Enrolment - Face Scan Complete

5.3.5 Officer Enrolment

5.3.5.1 Enrolling Officer's Personal Information

The process of enrolling an officer starts by entering his/her personal information. The information needed is:

- The officer's National ID number.
- Full name.
- Date of Birth.
- Gender.
- Username.
- Password.
- Role (as defined in Section 4.3.2).

The initial officer enrolment screen before and after filling in his/her personal details is shown in Figures 5.45 and 5.46.

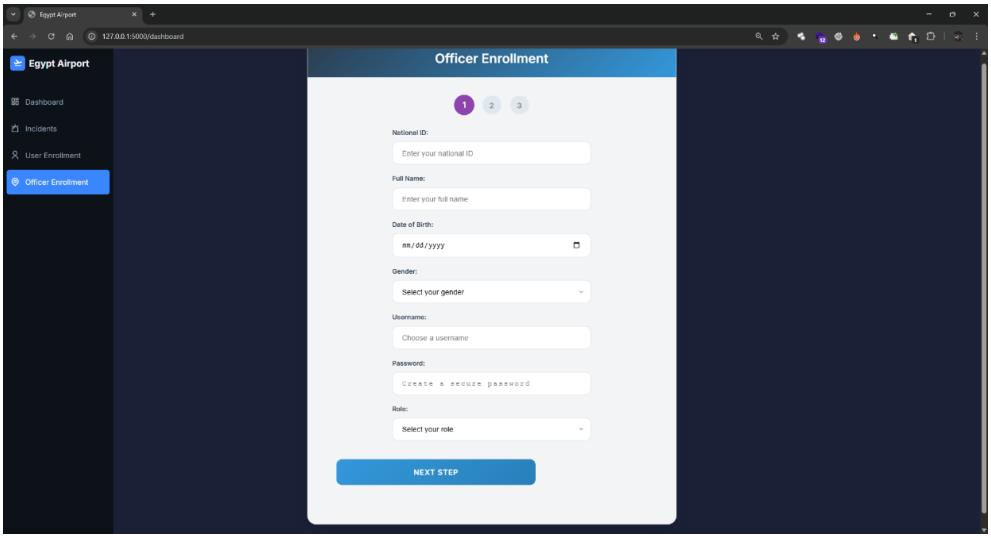

Figure 5.45: Officer Enrolment Screen

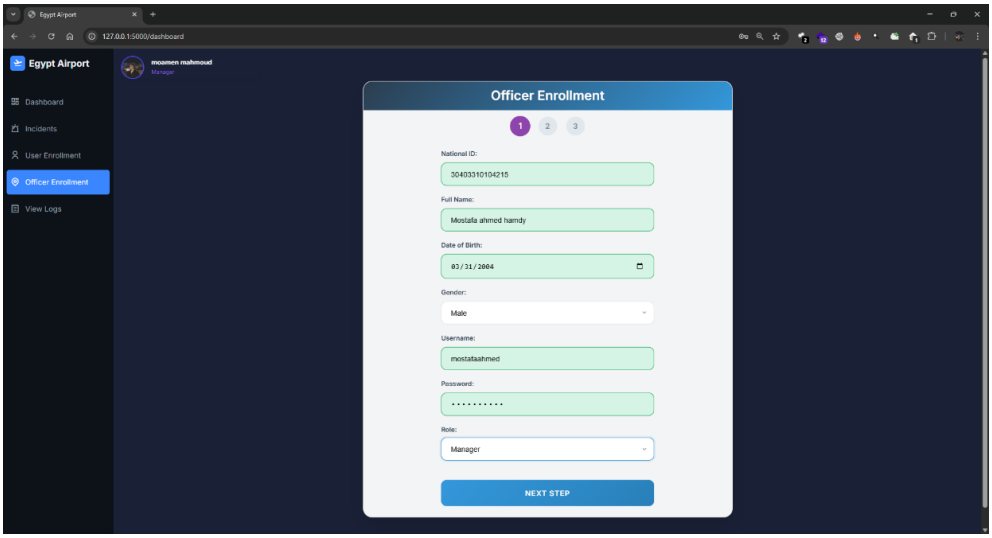

Figure 5.46: Officer Enrolment Screen - Credentials Filled

5.3.5.2 Enrolling Officer's Fingerprint

Following the uploading of the new officer's credentials and personal information, his/her biometric features are captured. This is in alignment with MFA. Enrolment of the new officer's fingerprint is shown in Figures 5.47 and 5.48.

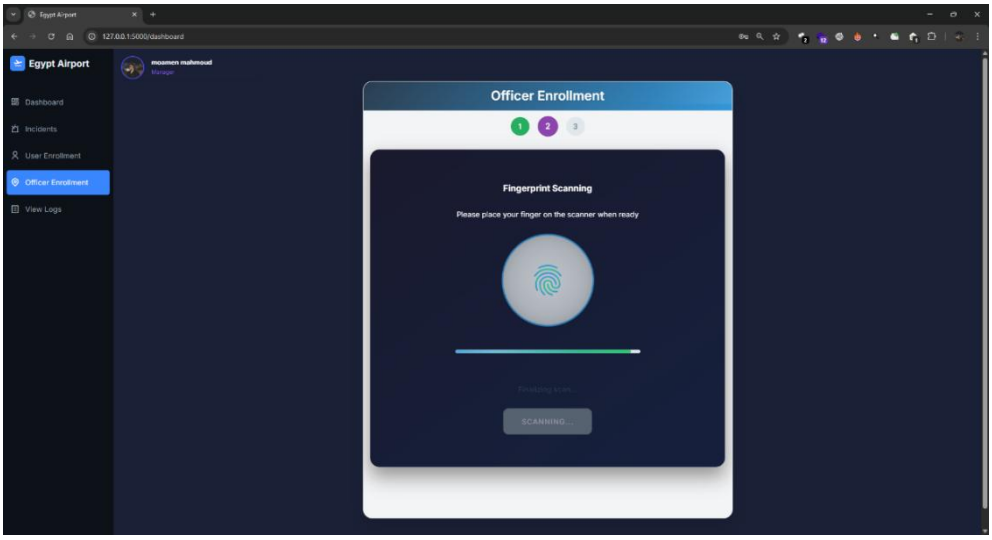

Figure 5.47: Officer Enrolment - Fingerprint Scanning in Progress

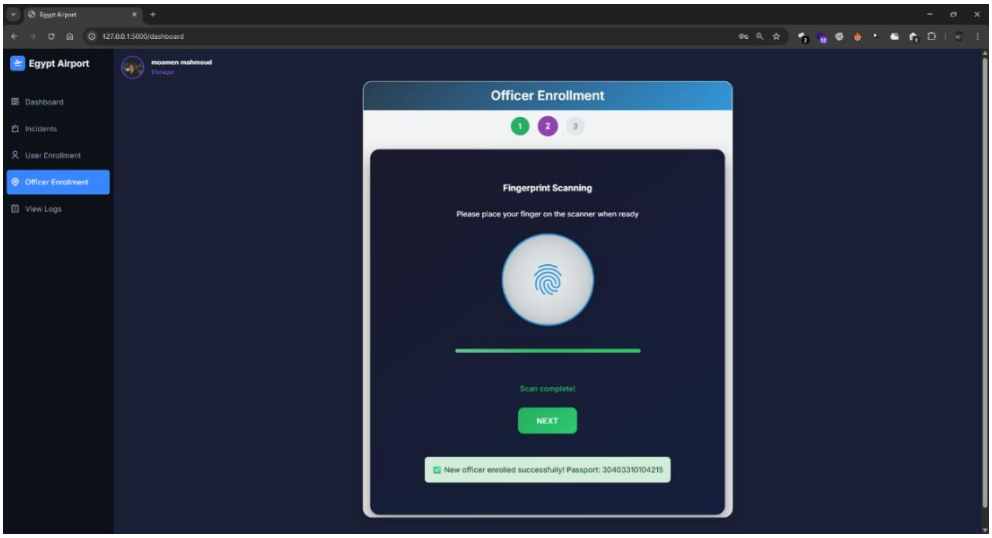

Figure 5.48: Officer Enrolment - Fingerprint Scan Complete

5.3.5.3 Enrolling Officer's Face

Following the enrolment of the fingerprint is the enrolment of the face, as seen in Figures 5.49 and 5.50.

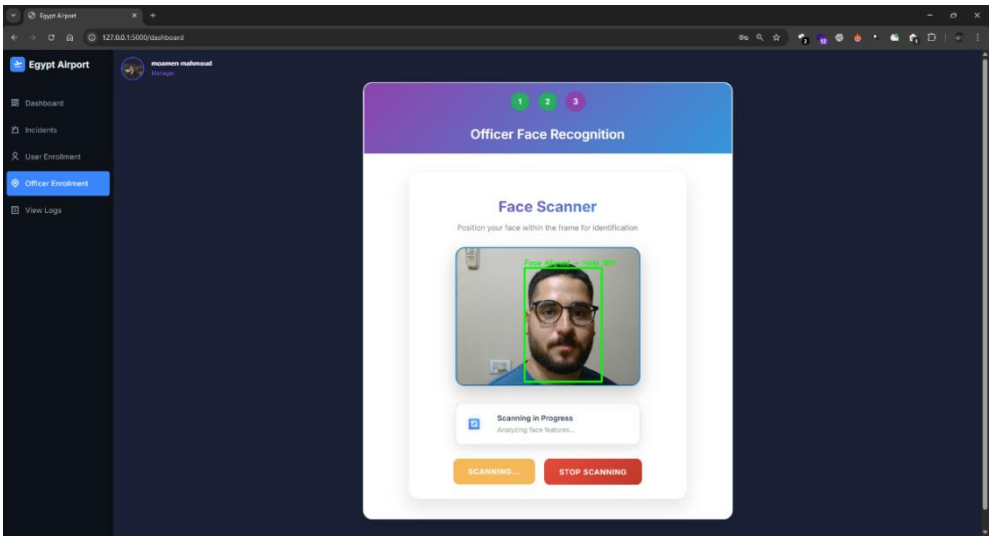

Figure 5.49: Officer Enrolment - Face Scanning in Progress

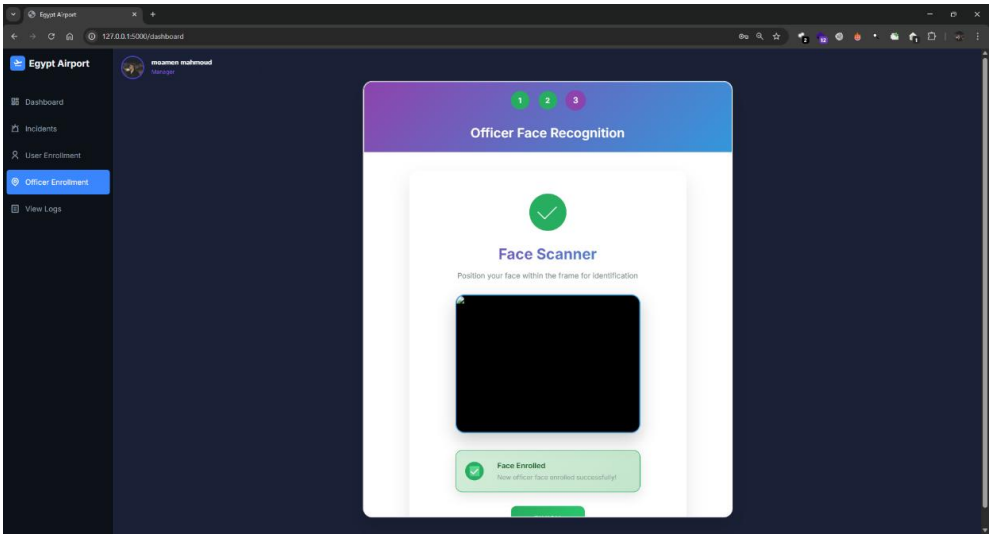

Figure 5.50: Officer Enrolment - Face Scan Complete

This step concludes the enrolment of a new officer into the system.

5.3.6 Log Viewing

Logs show the actions performed on the system while identifying the officer by name. A timestamp is also displayed. Logs like "*Officer logged in*", "*Changed status of an incident to [Under Investigation] [INC _001A1]*", and "*Enrolled new user [A40218840]*" with the responsible officer's name and the action's timestamp can be seen in Figure 5.51.

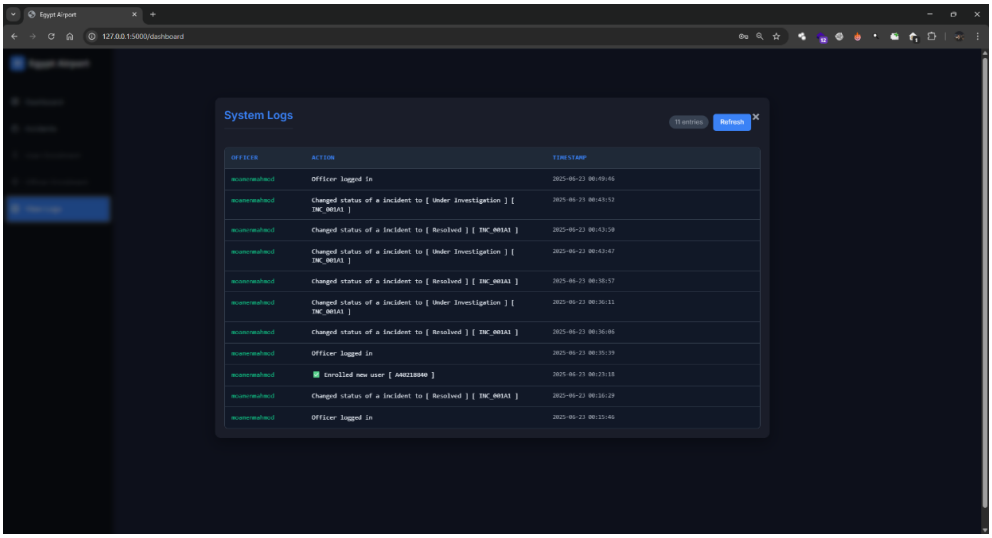

Figure 5.51: Log Viewing

5.3.7 Officer Permissions

This section is meant to show the difference in permissions between officers according to their roles. All actions shown in the Admin/ Officer Flow can be carried out by a Manager.

5.3.7.1 Admin

Admins have access to view logs, manage incidents and enroll users. Therefore, Sections 5.3.3, 5.3.4, 5.3.6 and their Figures show actions Admins can perform.

5.3.7.2 Junior

Junior officers are only allowed to view logs and manage incidents. These actions can be seen in the Figures shown in Sections 5.3.3 and 5.3.6.

Chapter 6 Conclusion and Future Work

6.1 Conclusion

This document presented the design and implementation of a cascade-based multimodal biometric border control system that integrates both face and fingerprint recognition, with spoof detection capabilities using deep learning models. It utilized a DenseNet-121 backbone with custom classifier heads for identity verification and spoofing detection, along with image processing techniques for fingerprint enhancement. Three datasets were used for the training and evaluation of the models: FVC2002, NUAA PI, and a created dataset. The work implemented in this study yielded accuracies of 90.40% and 99.60% for fingerprint and face identification respectively, and 99.82% and 100.00% for fingerprint and face spoofing detection respectively.

In addition to that, data security and privacy of the travelers was ensured by complying with the GDPR and implementing encryption, hashing and password guidelines.

The main contribution of this work is the addition of liveness detection to multimodal biometric systems. Unlike some of the existing systems that focus solely on identity verification, this system ensures both the accurate identification and verification of users, as well as the prevention of spoofing attacks using different materials. This was achieved by creating a dataset using Tape, Clay and Glue as spoofing materials, so not only can the system detect software generated fingerprints but real attempts as well. These results build towards allowing this project to be deployable in high security environments.

6.2 Future Work

Future work can extend this study by expanding the dataset for greater diversity of identities and materials used during spoofing attacks. Added to this could also be the detection of the specific material used in an attack, enabling a deeper understanding of threat patterns. In the context of identity verification, improving classification accuracy and reducing error rates are always important goals. Furthermore, evaluating the system in an uncontrolled environment with varying lighting and backgrounds is an essential step to prepare the system for real-world deployment.

Another potential improvement involves integrating a mobile application that securely stores the user's e-passport data, enabling a seamless and contactless border crossing experience. This mobile application could serve as a digital identity wallet, allowing travelers to present their credentials during various phases of the border control process without the need for physical documents.

Finally, future enhancements should address the security of data transmission and system communication channels. Implementing strong network security measures would be critical to ensure no attacker can hijack the system and falsely authenticate themselves.

References

[1] C. P. and S. E., "Cyber-threat landscape of border control infrastructures," *International Journal of Critical Infrastructure Protection,* vol. 36, p. 100503, 2022.

[2] S. S. Thenuwara, C. Premachandra and H. Kawanaka, "A multi-agent based enhancement for multimodal biometric system at border control," *Array,* vol. 14, p. 100171, 2022.

[3] F. Xia, X. Wei, Q. Zhao, B. Jin, Z. Yu, Z. Zha, D. Xiao, F. Long, G. Wu and J. Hu, "Exploration on the Application of Multimodal Biometric Feature Recognition in System Access Control," in *2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT)*, 2024.

[4] International Civil Aviation Organization, Machine Readable Travel Documents (Doc. 9303), Montréal: International Civil Aviation Organization, 2021.

[5] R. D. Labati, A. Genovese, E. Muñoz, V. Piuri, F. Scotti and G. Sforza, "Biometric Recognition in Automated Border Control: A Survey," *ACM Computing Surveys,* vol. 49, no. 2, pp. 1-39, 2016.

[6] D. White, R. I. Kemp, R. Jenkins, M. Matheson and A. M. Burton, "Passport Officers' Errors in Face Matching," *PLOS ONE,* vol. 9, pp. 1-6, 2014.

[7] International Air Transport Association, "Annual Report of the Air Transport Industry," June 2024. [Online]. Available: https://www.iata.org/contentassets/c81222d96c9a4e0bb4ff6ced0126f0bb/iata-annual-review-2024.pdf. [Accessed October 2024].

[8] OAG, "Airline Frequency and Capacity Statistics," 2025. [Online]. Available: https://www.oag.com/airline-frequency-and-capacity-statistics. [Accessed 26 June 2025].

[9] A. K. Jain, A. Ross and S. Prabhakar, "An introduction to biometric recognition," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 14, no. 1, pp. 4-20, 2004.

[10] U. Sumalatha, K. K. Prakasha, S. Prabhu and V. C. Nayak, "A Comprehensive Review of Unimodal and Multimodal Fingerprint Biometric Authentication Systems: Fusion, Attacks, and Template Protection," *IEEE Access,* vol. 12, pp. 64300-64334, 2024.

[11] International Air Transport Association, "Global Passenger Survey," 2024.

[12] S. Dhakal, *Multi-Biometric Systems: Their Application and Security,* Turku University of Applied Sciences, 2021.

[13] European Union, "General Data Protection Regulation," 2018. [Online]. Available: https://gdpr-info.eu/. [Accessed Feburary 2025].

[14] E. Bailly-Bailliere, S. Bengio, F. Bimbot, M. Hamouz, J. Kittler, J. Mariethoz, J. Matas, K. Messer, V. Popovici, F. Poree, B. Ruiz and J.-P. Thiran, "The BANCA Database and Evaluation Protocol," in *Audio- and Video-Based Biometric Person Authentication*, Springer Berlin Heidelberg, 2003, pp. 625-638.

[15] Y. Yin, L. Liu and X. Sun, "SDUMLA-HMT: A Multimodal Biometric Database," in *Biometric Recognition*, Springer Berlin Heidelberg, 2011, pp. 260-268.

[16] P. Tomar and R. C. Singh, "Cascade-based Multimodal Biometric Recognition System with Fingerprint and Face," *Macromolecular Symposia,* vol. 397, no. 1, p. 2000271, 2021.

[17] L. Hailin and R. Ramachandra, "Deep Learning based Fingerprint Presentation Attack Detection: A Comprehensive Survey," *ACM Computing Survey Journal,* p. 29, 2023.

[18] Liveness Detection Competition Series, *LivDet,* 2021.

[19] C. Yuan, S. Jiao, X. Sun and Q. M. Jonathan Wu, "MFFFLD: A Multimodal-Feature-Fusion-Based Fingerprint Liveness Detection," *IEEE Transactions on Cognitive and Developmental Systems,* vol. 14, no. 2, pp. 648-661, 2022.

[20] X. Tan, Y. Li, J. Liu and L. Jiang, "Face Liveness Detection from a Single Image with Sparse Low Rank Bilinear Discriminative Model," in *European Conference on Computer Vision*, 2010.

[21] B. Jawade, S. Subramanya, A. Dabhade, S. Setlur and V. Govindaraju, "GestSpoof: Gesture Based Spatio-Temporal Representation Learning for Robust Fingerprint Presentation Attack Detection," in *IEEE 18th International Conference on Automatic Face and Gesture Recognition (FG)*, Istanbul, Turkiye, 2024.

[22] S. Purnapatra, C. Miller-Lynch, S. Miner, Y. Liu, K. Bahmani, S. Dey and S. Schuckers, "Presentation Attack Detection with Advanced CNN Models for Noncontact-based Fingerprint Systems," in *2023 11th International Workshop on Biometrics and Forensics (IWBF)*, 2023.

[23] G. Huang, Z. Liu, L. van der Maaten and K. Q. Weinberger, *Densely Connected Convolutional Networks,* 2018.

[24] J. Galbally, F. Alonso-Fernandez, J. Fierrez and J. Ortega-Garcia, "A high performance fingerprint liveness detection method based on quality related features," *Future Generation Computer Systems,* vol. 28, no. 1, pp. 311-321, 2012.

[25] A. Alsubhi and N. Alsufyani, "Fingerprint Presentation Attack Detection Algorithm," in *2022 Fifth National Conference of Saudi Computers Colleges (NCCC)*, 2022.

[26] D. S. Ametefe, S. S. Sarnin, D. M. Ali and M. Z. Zaheer, "Fingerprint Liveness Detection Schemes: A Review on Presentation Attack," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization,* vol. 10, pp. 1-24, 2022.

[27] D. Sharma and A. Selwal, "A survey on face presentation attack detection mechanisms: hitherto and future perspectives," *Multimedia Systems,* June 2023.

[28] D. Ortega-Delcampo, A. Fernández-Isabel, I. M. de Diego, C. Conde and E. Cabello, "Dynamic Facial Presentation Attack Detection for Automated Border Control Systems," *Computers & Security,* vol. 92, p. 101744, 2020.

[29] K. Shaheed, P. Szczuko, M. Kumar, I. Qureshi, Q. Abbas and I. Ullah, "Deep learning techniques for biometric security: A systematic review of presentation attack detection systems," *Engineering Applications of Artificial Intelligence,* vol. 129, p. 107569, 2024.

[30] Q. N. Tran, B. P. Turnbull and J. Hu, "Biometrics and Privacy-Preservation: How Do They Evolve?," *IEEE Open Journal of the Computer Society,* vol. 2, pp. 179-191, 2021.

[31] M. Abohamra and S. Y. Yayilgan, *An Example of Privacy and Data Protection Best Practices,* 2022.

[32] Australian Border Force (ABF), *SmartGate,* 2007.

[33] B. C. Lovell, A. Bigdeli and S. Mau, "Invited paper: Embedded face and biometric technologies for national and border security," *CVPR 2011 WORKSHOPS,* pp. 117-122, 2011.

[34] Customs and Border Protection, *Global Entry,* 2008.

[35] France Republic, *PARAFE (Passage Automatisé Rapide aux Frontières Extérieures),* 2009.

[36] UAE Ministry of Interior, *e-Gate / Smart Gates,* 2002.

[37] D. Maltoni, D. Maio, S. Prabhakar and F. J., Handbook of Fingerprint Recognition, 2022.

[38] Second International Competition for Fingerprint Verification Algorithms, "FVC2002," 2002. [Online]. Available: http://bias.csr.unibo.it/fvc2002/. [Accessed 20 June 2025].

[39] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic differentiation in PyTorch," 2017.

[40] TorchVision maintainers and contributors, "TorchVision: PyTorch's Computer Vision library," 2016. [Online]. Available: https://github.com/pytorch/vision.git. [Accessed 20 June 2025].

[41] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Rio, M. Wiebe, P. Peterson, P. Gerard-Merchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke and T. E. Oliphant, "Array programming with NumPy," *Nature,* vol. 585, pp. 357-362, 2020.

[42] P. Umesh, "Image Processing in Python," *CSI Communications,* vol. 23, 2012.

[43] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools,* 2000.

[44] PyCA, *cryptography,* 2014.

[45] L. Hong, Y. Wan and A. Jain, "Fingerprint image enhancement: algorithm and performance evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, no. 8, pp. 777-789, 1998.

[46] R. Cappelli, D. Maltoni, D. Maio and A. Erol, "Synthetic fingerprint-image generation," in *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, 2000.

[47] Hangzhou Grow Technology Co., Ltd, "R307 Fingerprint Module User Manual," Feb 2011. [Online]. Available: https://www.openhacks.com/uploadsproductos/r307_fingerprint_module_user_manual.pdf. [Accessed 15 June 2025].

[48] University of Bologna, "Biometric System Laboratory," [Online]. Available: https://biolab.csr.unibo.it/research.asp?organize=Activities&select=&selObj=111&pathSubj=111&Req=&. [Accessed June 2025].

[49] A. Rattani, C. Chen and A. Ross, "Evaluation of Texture Descriptors for Automated Gender Estimation from Fingerprints," in *European Conference on Computer Vision*, 2015.

[50] A. Javed, M. Fasihullah, M. A. Munir, I. Usman and M. F. Shafique, "A New Additive Watermarking Technique for Multimodal Biometric Identification," vol. 3, 2013.

[51] R. C. Gonzalez and R. E. Woods, "Erosion, Dilation, Opening, Closing, Boundary Extraction, Connection," in *Digital Image Processing (3rd Edition)*, 2006.

[52] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang and Z. Tu, *Deeply-Supervised Nets,* 2014.

[53] A. Krizhevsky, V. Nair and G. Hinton, *CIFAR Dataset,* 2009.

[54] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.

[55] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu and A. Y. Ng, "Reading Digits in Natural Images with Unsupervised Feature Learning," *NIPS*

Workshop on Deep Learning and Unsupervised Feature Learning 2011, 2011.

[56] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[57] N. Vishwakarma, "What is Adam Optimizer?," Analytics Vidhya, 23 April 2025. [Online]. Available: https://www.analyticsvidhya.com/blog/2023/09/what-is-adam-optimizer/#h-how-adam-optimizer-works. [Accessed 20 June 2025].

[58] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple," in *Accepted Conference on Computer Vision and Pattern Recognition 2001*, 2001.

[59] A. Mittal, "Haar Cascades, Explained," Medium, 21 December 2020. [Online]. Available: https://medium.com/analytics-vidhya/haar-cascades-explained-38210e57970d. [Accessed 25 June 2025].

[60] "Understanding the AdaBoost Algorithm," Medium, 7 July 2023. [Online]. Available: https://medium.com/@datasciencewizards/understanding-the-adaboost-algorithm-2e9344d83d9b. [Accessed 20 June 2025].

[61] S. Jack, "Face detection using dlib HOG," Medium, 17 July 2020. [Online]. Available: https://medium.com/mlcrunch/face-detection-using-dlib-hog-198414837945. [Accessed 20 June 2025].

[62] V. Olufemi, "Face Detection With OpenCV," Medium, 2020. [Online]. Available: https://medium.com/@victorolufemi/face-detection-c27228ccc6f.

[63] Y. Jia, E. Shelhamer, J. Donahue and S. Karayev, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv preprint arXiv:1408.5093,* 2014.

[64] A. Rosebrock, "Training a custom dlib shape predictor," pyimagesearch, 2019. [Online]. Available: https://pyimagesearch.com/2019/12/16/training-a-custom-dlib-shape-predictor/.

[65] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou and M. Pantic, "300 faces in-the-wild challenge: Database and results," *Image and Vision Computing,* vol. 47, pp. 3-18, 2016.

[66] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2013.

[67] "Multi-Factor Authentication: Benefits, Best Practices & More," FORTINET, [Online]. Available: https://www.fortinet.com/resources/cyberglossary/multi-factor-authentication?utm_source=blog&utm_medium=blog&utm_campaign. [Accessed 20 June 2025].

[68] adafruit, "Adafruit Fingerprint Sensor Library," GitHub Repository, 2024. [Online]. Available: https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library.git. [Accessed January 2025].

[69] A. Khaled, A. Sohsah, N. Bahgat, S. Abdelmonem, Z. Mahmoud and M. Mahmoud, "Multimodal Biometric Border Control System," Github Repository, 2025. [Online]. Available: https://github.com/Abdellrhman-Khaled/Graduation-Project.git. [Accessed 25 June 2025].

Appendix A. Additional Tests and Experiments

The spoof model before training on the created dataset containing different spoofing materials, could not detect actual spoofing attempts and was only able to detect generated fingerprints using SFinGe [37] [46].

Figure A.1 below shows a spoofed fingerprint of Person A using clay.

Figure A.1: Spoofed Fingerprint of Person A using Clay - Additional Experiments: 1

The model was tested using a decision threshold of 0.8, and the corresponding results are presented in Table A.1 below.

Table A.1: SoftMax Output and Prediction Result - Additional Experiments: 1

SoftMax Probability	Prediction Result using Threshold 0.8
0.0012	No Spoofing Detected

Not only did it allow this spoofing attempt, but it also verified the spoofed fingerprint as Person A. Table A.2 below shows the similarity score found between this spoofed fingerprint and Person A.

Table A.2: Matching Scores and System Decision - Additional Experiments: 1

Similarity Score	Result using Threshold 0.93
0.9343	Identity Verified

This real-time test proved that the improvement of the spoofing model was necessary for this system to be considered for deployment.

Following the training of the spoofing model on real spoofing attempts, it was tested again using the same image. The results are shown in Table A.3 below.

Table A.3: SoftMax Output and Prediction Result - Additional Experiments: 2

SoftMax Probability	Prediction Result using Threshold 0.8
0.9830	Spoofing Detected

It was also tested with a software-generated fingerprint to ensure it was still robust against them. Figure A.2 shows the fake fingerprint used to test the model.

Figure A.2: Software-generated Fingerprint - Additional Experiments: 2

This test produced the following results presented in Table A.4 below.

Table A.4: SoftMax Output and Prediction Result - Additional Experiments: 2

SoftMax Probability	Prediction Result using Threshold 0.8
0.9986	Spoofing Detected

The spoofing model can now identify and prevent attacks with spoofed fingerprints of individuals saved in the database. These spoof samples include those created using materials such as glue, tape, and clay, as well as synthetically generated fingerprints. This strengthens the system's defense against impersonation and real spoofing attempts.

Appendix B. Code Explanations and Outputs

This appendix provides detailed explanations of some core code components implemented throughout the project along with relevant outputs. The full source code can be accessed on GitHub at [69].

B.1 Fingerprint Acquisition:

1. *def send_raw_command(ser, command_hex):*
   command_bytes = bytearray(command_hex)
   ser.write(command_bytes)
   response = ser.read(12)
   return response

This function sends a hexadecimal command to the sensor through the serial port and reads the 12-byte response.

Input:

- **ser**: a serial connection initialized in the main function. ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=5)
- **command_hex**: The hexadecimal instruction that is going to be sent to the sensor.

Output:

- **response**: 12-byte response from the sensor.

Example 1 (Successful fingerprint capture):

Input:

- **ser** = serial.Serial("COM5", 115200, timeout=5)
- **command_hex** = [0xEF, 0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0x01, 0x00, 0x03, 0x01, 0x00, 0x05]

Output:

- **response** = [0xEF, 0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0x07, 0x00, 0x03, 0x00, 0x00, 0x0A]

Example 2 (Unsuccessful fingerprint capture):

Input:

- **ser** = serial.Serial("COM5", 115200, timeout=5)

- **command_hex** = [0xEF, 0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0x01, 0x00, 0x03, 0x01, 0x00, 0x05]

Output:

- **response** = [0xEF, 0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0x07, 0x00, 0x03, 0x02, 0x00, 0x0C]

2. *def parse_response(response):*

```
if len(response) < 12:
    return False, "Invalid response packet"
status_code = response[9]
if status_code == 0x00:
    return True, "Success"
error_codes = {
    0x01: "Error when receiving package",
    0x02: "No finger detected",
    0x03: "Fail to collect finger",
}
return False, error_codes.get(status_code, f"Unknown error (code {status_code:02X})")
```

This function parses the response received through the serial port and provides feedback.

Input:

- **response**: The 12-byte response from the sensor after sending a command.

Output:

- Boolean true or false.
- **status_code**: Error code description, if any.

Example 1:

Input:

- **response** = [0xEF, 0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0x07, 0x00, 0x03, 0x00, 0x00, 0x0A]

Output:

- True

Example 2:

Input:

- **response** = [0xEF, 0x01, 0xFF, 0xFF, 0xFF, 0xFF, 0x07, 0x00, 0x03, 0x02, 0x00, 0x0C]

Output:

- False, "No finger detected".

3. *def convert_and_save_image(raw_data, filename="first_try.bmp"):*

```
num_pixels = len(raw_data)
width = 128
height = 288
if width * height != num_pixels:
    raise ValueError(f"Image dimensions mismatch: {width}x{height} does not match {num_pixels} pixels.")
try:
    image_data = np.array(raw_data, dtype=np.uint8).reshape((height, width))
except ValueError as e:
    print(f"Error reshaping data: {e}")
    return
img = Image.fromarray(image_data)
print(f"Image saved as '{filename}' with dimensions {width}x{height}")
expanded_image = np.zeros((288, 256), dtype=np.uint8)
byte_index = 0
for y in range(288):
    for x in range(0, 256, 2):
        byte_value = raw_data[byte_index]
        high_pixel = (byte_value >> 4) & 0x0F
        low_pixel = byte_value & 0x0F
        expanded_image[y, x] = high_pixel * 16
        expanded_image[y, x + 1] = low_pixel * 16
        byte_index += 1
img = Image.fromarray(expanded_image)
print("Image saved with dimensions 256x288")
return img
```

The purpose of this function is to convert the captured raw fingerprint image data to an image. It first converts the 36,864 bytes of raw data into a 128*288 image, then expands it to a 256*288 image by multiplying each 4-bit pixel value by 16.

It has additional exception handling for when image dimensions do not match the number of raw bytes and if an unexpected error occurs when reshaping the raw data into a NumPy array representing the image.

Input:

- **raw_data**: raw image data bytes.

Output:

- **img**: the expanded images with the desired dimensions of 256*288.

4. *def convert_bmp_to_jpg(image):*

```
try:
    jpg_image = image.convert("RGB")  # Convert BMP to RGB (JPG format)
    return jpg_image  # Return the converted image
except Exception as e:
    print(f"Error: {e}")
    return None
```

This is used to convert an image's extension to JPG. It is used in *capture_fingerprint*.

5. *def capture_fingerprint():*

```
ser = None
try:
    ser = serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=5)
    print(f"Connected to {SERIAL_PORT} at {BAUD_RATE} baud.")
    capture_command = [
        0xEF, 0x01,
        0xFF, 0xFF, 0xFF, 0xFF,
        0x01,
        0x00, 0x03,
        0x01,
        0x00, 0x05
    ]
    print("Sending capture command...")
    response = send_raw_command(ser, capture_command)
    success, message = parse_response(response)
    if not success:
        print(f"Error capturing image. {message}")
        return None
    print("Image captured successfully.")
    upload_command = [
        0xEF, 0x01,
        0xFF, 0xFF, 0xFF, 0xFF,
        0x01,
```

```
            0x00, 0x03,
            0x0A,
            0x00, 0x0E
        ]
        print("Sending upload command...")
        response = send_raw_command(ser, upload_command)
        success, message = parse_response(response)
        if not success:
            print(f"Error uploading image. {message}")
            return None
        image_data = bytearray()
        expected_size = 36864
        while len(image_data) < expected_size:
            print(f"Expected size: {expected_size}, Received: {len(image_data)}")
            chunk = ser.read(43)
            if not chunk:
                break
            data = chunk[9:41]
            image_data.extend(data)
        if len(image_data) == 0:
            print("Error: No image data received.")
            return None
        if len(image_data) != expected_size:
            print(f"Warning: Expected {expected_size} bytes, but received {len(image_data)} bytes.")
        img = convert_and_save_image(image_data)
        img2 = convert_bmp_to_jpg(img)
        img2.save("static\\captured_finger\\cap.jpg")
        img3 = enhance_fingerprint(img2)
        img_pil = Image.fromarray(img3)
        final_path = "static\\captured_finger\\done.jpg"
        img_pil.save(final_path)
        notenhanced = "static\\captured_finger\\cap.jpg"
        enhanced = "static\\captured_finger\\done.jpg"
        return notenhanced, enhanced
    except Exception as e:
        print(f"Error: {e}")
```

```
        return None
    finally:
        if ser and ser.is_open:
            ser.close()
```

Utilizes the functions *send_raw_command*, *parse_response* and *convert_and_save_image*, *enhance_fingerprint* and *convert_bmp_to_jpg*. It captures and uploads the fingerprint image as a BMP 8-bit image to the computer by sending the hexadecimal command for capturing followed by the command for uploading and ensuring they were successful.

During uploading, data is read in chunks of 43 and raw fingerprint data is extracted from the 10th to the 42nd byte and appended to a byte array representing the raw image data. It then converts the image from BMP to JPG and enhances it using the fingerprint-enhancer.

Exception handling is carried out, and if all is successful, the serial port is closed after.

Output:

- **notenhanced**: The fingerprint image before enhancement.
- **enhanced**: The fingerprint image after enhancement.

Example:

Output:

- The captured image before and after enhancement is shown in Figure B.1 below.

Figure B.1: Captured Image (Left) and Enhanced Image (Right)

B.2 Fingerprint Enhancement:

1. *def enhance_fingerprint(img):*

```
    """
    Enhances a fingerprint image using fingerprint_enhancer.
    :param img: Input image (PIL Image object or NumPy array)
    :return: Enhanced fingerprint image (NumPy array)
    """
    # Convert PIL Image to NumPy array if necessary
    if isinstance(img, Image.Image):
        img = np.array(img.convert("L"))  # Convert to grayscale
    print("Enhancing fingerprint...")
    # Enhance the fingerprint image
    enhanced_img = fingerprint_enhancer.enhance_fingerprint(img)
    # Convert from bool to uint8 (needed for saving and displaying)
    enhanced_img = (enhanced_img * 255).astype(np.uint8)
    return enhanced_img
```

This is an implementation of the fingerprint-enhancer [45] library's function *enhance_fingerprint*.

Input:

- **img**: the captured image.

Output:

- **enhancedimg**: image after enhancing.

2. *def enhance_fingerprint(*

```
    img: np.ndarray,
    resize: bool = False,
    ridge_segment_blksze: int = 16,
    ridge_segment_thresh: float = 0.1,
    gradient_sigma: int = 1,
    block_sigma: int = 7,
    orient_smooth_sigma: int = 7,
    ridge_freq_blksze: int = 38,
    ridge_freq_windsze: int = 5,
    min_wave_length: int = 5,
    max_wave_length: int = 15,
```

```
    relative_scale_factor_x: float = 0.65,
    relative_scale_factor_y: float = 0.65,
    angle_inc: float = 3.0,
    ridge_filter_thresh: int = -3,
    invert_output: bool = False,
) -> np.ndarray:
    image_enhancer = FingerprintImageEnhancer(
        ridge_segment_blksze,
        ridge_segment_thresh,
        gradient_sigma,
        block_sigma,
        orient_smooth_sigma,
        ridge_freq_blksze,
        ridge_freq_windsze,
        min_wave_length,
        max_wave_length,
        relative_scale_factor_x,
        relative_scale_factor_y,
        angle_inc,
        ridge_filter_thresh,
    )  # Create object called image_enhancer
    enhanced_output = image_enhancer.enhance(img, resize, invert_output=invert_output)
    return enhanced_output
```

Fingerprint-enhancer's [45] function enhance_fingerprint enhances a fingerprint image using a configurable image enhancement pipeline. The pipeline is as follows:

1. **resize**: resizes the input to a fixed size. Default is False.
2. **ridge_segment_blksze**: sizes of blocks used to divide the image for ridge segmentation. Default is 16*16.
3. **ridge_segment_thresh**: threshold for determining ridge vs non-ridge regions. Defaults to 0.1.
4. **gradient_sigma**: sigma value for Gaussian smoothing when computing gradients. Defaults to 1.
5. **block_sigma**: sigma value for Gaussian smoothing during orientation estimation. Defaults to 7.
6. **orient_smooth_sigma**: sigma value for smoothing the orientation field. Default value is 7.
7. **ridge_freq_blksze**: block size used when calculating the frequency of ridge patterns. Defaults to 38.

8. **ridge_freq_windsize**: window size for ridge frequency calculations. Defaults to 5.
9. **min_wave_length**: minimum allowed wavelength between ridges. Defaults to 5.
10. **max_value_length**: maximum allowed wavelength between ridges. Defaults to 15.
11. **relative_scale_factor_x**: scaling factor for adjusting image resolution in the x direction. Defaults to 0.65.
12. **relative_scale_factor_y**: scaling factor for adjusting image resolution in the y direction. Defaults to 0.65.
13. **angle_inc**: angle increment for applying Gabor filtering. Defaults to 3.0.
14. **ridge_filter_thresh**: threshold to apply on the ridge filter response. Defaults to -3.
15. **invert_output**: flag to invert the enhanced output. Default is False.

All these parameters can be taken as input, or their default values can be used.

Output:

- **enhanced_output**: enhanced image.

B.3 Face Detection and Capturing

1. *# Model and Directory Parameters*
   model_prototxt = "models/deploy.prototxt"
   model_weights = "models/res10_300x300_ssd_iter_140000.caffemodel"
   predictor_path = "models/shape_predictor_68_face_landmarks.dat"
   in_width = 300
   in_height = 300
   mean = [104, 117, 123]
   conf_threshold = 0.7
   save_dir = "captured_faces"
   countdown_duration = 5 # Capture delay in seconds

This code segment handles the configuration of the model as well as path definitions. Model settings include:

- **in_width, in_height**: model input size set to expect images of size 300*300.
- **mean**: channel-wise mean values set to match the pre-processing used for the Caffe-based face detector.
- **conf_threshold**: the confidence threshold set for detecting faces; any detection with confidence lower than 0.7 is ignored.
- **countdown_duration**: the number of seconds to wait before capturing the image after the face alignment is detected.

2. *def initialize_camera():*
   """
   Initialize the webcam and create a display window for real-time preview.
   """
   cap = cv2.VideoCapture(0)
   win_name = 'Camera Preview'
   cv2.namedWindow(win_name, cv2.WINDOW_NORMAL)
   cv2.resizeWindow(win_name, 800, 600)
   return cap, win_name

initialize_camera initializes the webcam with ID 0 and creates a resizable preview window to display the camera feed at 800x600.

Output:

- **cap**: the video capture object.
- **win_name**: the name of the display window.

3. *def load_models():*

   """

   Load the Caffe face detection model and Dlib facial landmark predictor.

   """

   net = cv2.dnn.readNetFromCaffe(model_prototxt, model_weights)

   detector = dlib.get_frontal_face_detector()

   predictor = dlib.shape_predictor(predictor_path)

   return net, detector, predictor

This function combines the loading of all the required models for face detection and capturing. It loads:

- A Caffe-based deep neural network model for face detection.
- A Dlib frontal face detector.
- A Dlib 68-point facial landmark predictor.

Output:

- **net**: the Caffe-based face detection model.
- **detector**: Dlib's frontal face detector.
- **predictor**: Dlib's shape predictor.

4. *def detect_faces(frame, net, frame_width, frame_height):*

   """

   Detect faces in a frame using the Caffe model and return bounding box coordinates.

   """

   blob = cv2.dnn.blobFromImage(frame, 1.0, (in_width, in_height), mean, swapRB=False, crop=False)

   net.setInput(blob)

   detections = net.forward()

   faces = []

   for i in range(detections.shape[2]):

      confidence = detections[0, 0, i, 2]

      if confidence > conf_threshold:

         *x1 = int(detections[0, 0, i, 3] * frame_width)*

         *y1 = int(detections[0, 0, i, 4] * frame_height)*

         *x2 = int(detections[0, 0, i, 5] * frame_width)*

```
            y2 = int(detections[0, 0, i, 6] * frame_height)
            faces.append((x1, y1, x2, y2))
    return faces
```

This function detects faces in a video frame using the Caffe model, returning bounding box coordinates for faces with confidence above a threshold. The frame is preprocessed to match the model's input requirements; frame is converted into a blob.

Input:

- **frame**: the input frame in BGR format.
- **net**: the Caffe-based face detection model.
- **frame_width**: width of the input frame.
- **frame_height**: height of the input frame.

Output:

- **faces**: a list of all valid face bounding boxes in the current frame.

5.
```
def expand_bounding_box(x1, y1, x2, y2, frame_width, frame_height):
    """
    Expand the bounding box around a detected face to ensure full face capture.
    """
    width_expand = int((x2 - x1) * 0.2)
    height_expand = int((y2 - y1) * 0.2)
    x1 = max(0, x1 - width_expand)
    x2 = min(frame_width, x2 + width_expand)
    y1 = max(0, y1 - height_expand)
    y2 = min(frame_height, y2 + height_expand)
    return x1, y1, x2, y2
```

The original bounding box coordinates of a detected face are expanded slightly in all directions using this function. This is to ensure that the entire face is safely captured. Frame height and width are necessary here to prevent the new coordinates from going out of frame.

Input:

- **x1, y1**: top-left corner of the original box.
- **x2, y2**: bottom-right corner of the original box.
- **frame_width**: width of the frame.
- **frame_height**: height of the frame.

Output:

- **x1, y1, x2, y2**: expanded bounding box coordinates.

6. *def detect_landmarks(frame, detector, predictor):*

```
"""
Detect facial landmarks in a grayscale frame using Dlib.
"""
gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
dlib_rects = detector(gray_frame)
if len(dlib_rects) > 0:
    landmarks = predictor(gray_frame, dlib_rects[0]).parts()
    return landmarks
return None
```

The face detection and shape prediction models are utilized here to extract facial landmarks from a given video frame. The frame is converted to greyscale to match Dlib detector's requirements.

Input:

- **frame**: the frame in BGR format.
- **detector**: Dlib's frontal face detector.
- **predictor**: Dlib's shape predictor.

Output:

- **landmarks**: a list of 68 facial landmarks returned by Dlib's shape predictor.

7. *def is_face_straight(landmarks):*

```
"""
Check if the face is straight using eye and nose alignment.
"""
left_eye = (landmarks[36].x, landmarks[36].y)
right_eye = (landmarks[45].x, landmarks[45].y)
nose_tip = (landmarks[30].x, landmarks[30].y)
```

```
    eye_center_x = (left_eye[0] + right_eye[0]) / 2
    eye_nose_diff = abs(nose_tip[0] - eye_center_x)

    return eye_nose_diff < 20  # Increased tolerance for slight misalignment
```

This function determines face alignment based on eye and nose position symmetry. It retrieves the coordinates of the left eye's outer corner (landmark 36), the right eye's outer corner (landmark 45) and the center tip of the nose (landmark 30). The x-coordinate between the left and right eyes is calculated; this is where the nose should ideally align vertically for a straight face. Finally, the absolute horizontal distance between the nose tip's x-coordinate and the eye center is computed. If the value is small (within 20 pixels), the face is likely facing forward, if it is large, the face may be tilted left or right.

Input:

- **landmarks**: a list of 68 facial landmarks returned by Dlib's shape predictor.

Output:

- Boolean true or false.

8.
```
def overlay_feedback(frame, x1, y1, x2, y2, face_valid, capture_start_time):
    """
    Overlay visual feedback (bounding box, alignment message, countdown) on the frame.
    """
    overlay_color = (0, 255, 0) if face_valid else (0, 0, 255)
    message = "Face Aligned - Hold Still" if face_valid else "Face Misaligned - Adjust Position"
    cv2.rectangle(frame, (x1, y1), (x2, y2), overlay_color, 5)
    cv2.putText(frame, message, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
overlay_color, 2)
    if face_valid and capture_start_time:
        time_elapsed = time.time() - capture_start_time
        countdown = countdown_duration - int(time_elapsed)
        if countdown > 0:
            cv2.putText(frame, f"Capturing in {countdown}...", (x1, y2 + 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2)
        return countdown
    return None
```

This function is defined to overlay visual feedback: colored boxes and text on a video frame. If the face is aligned, a green box with the message "Face Aligned – Hold Still" is displayed along with the

countdown. If the face is misaligned, a red box with the message “Face Misaligned – Adjust Position” is displayed.

Input:

- **frame**: the current frame to draw on.
- **x1, y1, x2, y2**: coordinates of the face bounding box.
- **face_valid**: Boolean indicating whether the face is properly aligned.
- **capture_start_time**: timestamp when the valid face was first detected.

Output:

- **countdown**: if the face is valid and aligned, the current countdown is returned.

9.
```
def save_face_image(frame, x1, y1, x2, y2, save_dir):
    """
    Save the detected face region to a file.
    """
    face_roi = frame[y1:y2, x1:x2]
    if not os.path.exists(save_dir):
        os.makedirs(save_dir)
    image_path = os.path.join(save_dir, "captured_face.jpg")
    cv2.imwrite(image_path, face_roi)
    print(f"Face saved at: {image_path}")
    return image_path
```

Finally, the detected face region is saved to a file.

Input:

- **frame**: the frame containing the detected face.
- **x1, y1, x2, y2**: coordinates of the face bounding box.
- **save_dir**: directory path where the face image is going to be saved.

Output:

- **image_path**: the path to the saved image.