Advanced Biometric Security System for Automated Teller Machine Transactions

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

Bachelor of Technology
in
Computer Science & Engineering

By

MANDHADI VISHAL GOUD	(19UECS0579)	**(13296)**
MATMARI SHASHANK	(19UECS0544)	**(15273)**
A. NITHIN YADAV	(19UECS0005)	**(15274)**

Under the guidance of
Mr. NAJEEM DHEEN ABDUL MAJEETH, ME.,
ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

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

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

April, 2023

ADVANCED BIOMETRIC SECURITY SYSTEM FOR AUTOMATED TELLER MACHINE TRANSACTIONS

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

Bachelor of Technology
in
Computer Science & Engineering

By

MANDHADI VISHAL GOUD	(19UECS0579)	**(13296)**
MATMARI SHASHANK	(19UECS0544)	**(15273)**
A. NITHIN YADAV	(19UECS0005)	**(15274)**

Under the guidance of
Mr. NAJEEM DHEEN ABDUL MAJEETH, ME.,
ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
SCHOOL OF COMPUTING

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

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

April, 2023

CERTIFICATE

It is certified that the work contained in the project report titled ”Advanced Biometric Security System for Automated Teller Machine” by ”MANDHADI VISHAL GOUD (19UECS0579), MATMARI SHASHANK (19UECS0544), A. NITHIN YADAV (19UECS0005)” has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Signature of Supervisor
Mr. Najeem Dheen Abdul Majeeth
Assistant Professor
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr.Sagunthala R&D
Institute of Science & Technology
April, 2023

Signature of Head of the Department
Dr. M. S. Murli Dhar, M. E. Ph. D
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
April, 2023

Signature of the Dean
Dr. V. Srinivasa Rao
Professor & Dean
Computer Science & Engineering
School of Computing
Vel Tech Rangarajan Dr. Sagunthala R&D
Institute of Science & Technology
April, 2023

DECLARATION

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

(MANDHADI VISHAL GOUD)

Date: / /

(MATMARI SHASHANK)

Date: / /

(A.NITHIN YADAV)

Date: / /

APPROVAL SHEET

This project report entitled ”Advanced Biometric Security System for Automated Teller Machine Transactions” by MANDHADI VISHAL GOUD (19UECS0579), A.NITHIN.YADAV (19UECS0005), MATMARI SHASHANK(19UECS0544) is approved for the degree of B.Tech in Computer Science & Engineering.

Examiners **Supervisor**

Mr. NAJEEM DHEEN ABDUL MAJEETH, M. E.

Date: / /

Place:

ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B. E. (EEE), B. E. (MECH), M. S (AUTO),D. Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M. B. B. S.** Chairperson Managing Trustee and Vice President.

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

We record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. V. SRINIVASA RAO, M. Tech., Ph. D.,** for immense care and encouragement towards us throughout the course of this project.

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

We also take this opportunity to express a deep sense of gratitude to our Internal Supervisor **Mr A NAJEEM DHEEN, ME** for his cordial support, valuable information and guidance, he helped us in completing this project through various stages.

A special thanks to our **Project Coordinators Mr. V. ASHOK KUMAR, M. Tech., Ms. C. SHYAMALA KUMARI, M. E.,** for their valuable guidance and support throughout the course of the project.

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

MANDHADI VISHAL GOUD (19UECS0579)
MATMARI SHASHANK (19UECS0544)
A. NITHIN YADAV (19UECS0005)

ABSTRACT

The Advanced Facial Security System for Automated Teller Machine Transactions is a significant development in the field of security and efficiency for ATM transactions. With the help of machine learning algorithms such as Deep Neural Network and Support Vector Machine, the system provides highly accurate and efficient facial recognition to prevent fraudulent transactions. The system's architecture is designed to work with a standard ATM interface, making it easy to use for customers. The implementation of the project is based on the OpenCV and Torch libraries, which provide an extensive range of tools for image processing, computer vision, and deep learning. This combination of libraries enables the system to provide highly accurate results for facial recognition in real-time, ensuring that transactions are completed efficiently. To ensure the system's performance, accuracy analysis was performed, which revealed an accuracy of 80% - 84%. This result indicates the system's high accuracy in identifying the user, making it highly secure and reliable for ATM transactions. In conclusion, the proposed system's benefits include enhanced security, reduced transaction times, and a user-friendly interface. The use of machine learning algorithms such as DNN and SVM, along with OpenCV and Torch libraries, has made the system highly accurate, efficient, and reliable. The system's development aims to create a highly secure and efficient ATM system that utilizes the latest in facial recognition and machine learning technologies to improve the overall user experience.

Keywords: ATM, Tkinter, SVM, OpenCV, Tensor Flow, Scikit-learn, Python, DNN, Biometrics, AI.

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

- ATM- Automated Teller Machines
- SVM - Support Vector Machine
- DNN - Deep Neural Network
- PIN - Personal Identification Number
- AES - Advanced Encryption Standard
- FPGA - Field Programmable Gate Array
- LCD - Liquid Crystal Display
- VHSIC - Very High Speed Integrated Circuit Hardware
- RAM - Random Access Memory
- ROM - Read only Memory
- OTP - One Time Password
- RFID - Radio-Frequency IDentification
- LBP - Local Binary Patterns
- SIFT - Scale-Invariant Feature Transform
- AI - Artificial Intelligence

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 Introduction

All kinds of banking transactions are considered essential in people's lives, making banks deploy ATMs in multiple places to grant users easier access to their services. However, while using an ATM card, they may encounter many problems and difficulties. Some clients forget the PIN or card number while others forget or lose the card itself. Another problem may be frequent thefts and acts of forgery by criminals. All these problems are due to the banks' reliance on the traditional card-based system based on the PIN. Therefore, a solution had to be made to switch to a better method for identification and authorization of ATM card transactions. However, technologies are getting more sophisticated. Also, fraud methods are increasing rapidly.

To address this issue, biometric authentication methods, particularly facial recognition, have emerged as a promising solution for enhancing the security of ATM transactions. In this project, we propose an advanced biometric security system for ATM transactions that incorporates the latest developments in facial recognition technology and additional security measures. The system employs state-of-the-art machine learning algorithms to accurately match the user's facial landmarks with the stored biometric template and integrates multifactor authentication and secure communication protocols to ensure the highest level of security for ATM transactions. The goal of this project is to provide users with a convenient and secure authentication method for conducting financial transactions at ATMs, while also providing financial institutions with a powerful tool to protect against fraudulent activities..

1.2 Aim of the Project

The aim of the project is to enhance security and provide a convenient and secure method for conducting transactions using ATMs by using facial recognition technology and additional security measures. The traditional ATM card and PIN-based system is a simple and uncomplicated system. But it also makes it easier for fraud-

ulent activities to take place by either obtaining the PIN through a regular shoulder surfing attack, stealing the card, putting on a fake PIN Pad, and many other techniques. Thus, to overcome this issue, one of the most potent ways to support ATM security is the use of biometrics to identify the customer's identity. One of the most important goals of our project is "To verify and ensure that the person who is accessing the account is the authorized one through biometrics fusion." Therefore, we use a biometric verification and authentication method to identify the person (User) with solid security and determine his identity to successful and safe attempt to access the desired banking service.

1.3 Project Domain

The domain of machine learning is critical for the development of the Advanced facial Security System for ATM Transactions project. Machine learning is a branch of artificial intelligence that allows computers to learn and improve from experience without being explicitly programmed. In this project, machine learning techniques are used to develop an advanced facial recognition system that can improve security in ATMs by identifying users based on their facial features.

The machine learning domain knowledge required for this project includes expertise in various areas, including data preprocessing, feature selection, model selection, and model evaluation. The project team must have a deep understanding of different machine learning algorithms, such as support vector machines, decision trees, and neural networks, and their applications in computer vision and facial recognition.

To design an efficient and scalable machine learning pipeline that can process vast amounts of facial data, the team needs to have a strong understanding of data handling techniques, such as normalization and data augmentation. The team must also ensure that the system is accurate, reliable, and secure to prevent fraudulent activities, such as identity theft and card skimming.

In summary, the machine learning domain is fundamental for the Advanced facial Security System for ATM Transactions project, and the project team needs to have expertise in this domain to develop an effective and efficient facial recognition system for ATMs.

1.4 Scope of the Project

The Advanced Biometric Security System for ATM Transactions using machine learning project has significant scope in the banking and financial industry. The project aims to enhance ATM security by implementing an advanced facial recognition system that can identify users based on their facial features.

The project's scope includes developing a robust and scalable facial recognition system that can detect and prevent fraudulent activities, such as identity theft and card skimming. The system would be able to authenticate users based on facial features, thereby reducing the risk of ATM fraud.

The scope of the project also includes collecting and preprocessing facial data for training machine learning algorithms. This data may be obtained through various sources, including facial recognition databases, camera feeds, and customer identification documents.

Once the system is developed, there is scope for its integration into existing ATM systems, providing enhanced security and convenience to ATM users. Additionally, the system's underlying machine learning algorithms can be further refined and optimized to improve accuracy and reduce false positives.

The scope of the project extends beyond just ATM security and facial recognition. The knowledge and expertise gained in this project can be applied to other domains, such as surveillance, access control, and biometric authentication systems.

In conclusion, the Advanced Biometric Security System for Automated Teller Machine Transactions using machine learning project has significant scope and potential for improving ATM security and contributing to the broader field of biometric authentication systems.

Chapter 2

LITERATURE REVIEW

[1] Priyanka Mahajan., presented a model which suggests that there is an urgent need for improving security in banking region. In this paper discussion is made about the face recognition technology, an important field of biometrics which will be employed for the purpose of checks on frauds using ATMs. The recent progress in biometric identification techniques, has made a great efforts to rescue the unsafe situations at the ATM. Several facial recognition techniques are studied which include two approaches, appearance based and geometric based. A new facial recognition technique: 3-D technique is also reviewed in the paper. These techniques are widely used in e-passports and on the airports for entry of travellers and others. ATM systems today use no more than an access card and PIN for identity verification. If the proposed technology of facial recognition becomes widely used, faces would be protected as well as PINs.

[2] Nabihah Ahmad et al., presented a system that gives freedom to the user by changing the card to biometric security system to access the bank account using AES algorithm. The project is implemented using Field Programmable Gate Array DE2-115 board with Cyclone IV device, fingerprint scanner, and Multi-Touch LCD Second Edition (MTL2) using VHSIC Description Language VHDL. This project used 128-bits AES for recommend the device with the throughput around 19.016Gbps and utilized around 520 slices. This design offers a secure banking transaction with a low rea and high performance and very suited for restricted space environments for small amounts of RAM or ROM where either encryption or decryption is performed.

[3] Manish C M et al., described a study that replaces the ATM cards and PINS by the physiological biometric fingerprint authentication and face recognition. Moreover, the feature of OTP imparts privacy to the users and emancipates him/her from recalling PINS. In this system during enrollment the genuine user's fingerprint and face is retained in the database. The process of transaction begins by capturing and matching fingerprints and face patterns. The system will automatically distinguish

between real legitimate trait and fake samples. A GSM module connected to the Microcontroller will send a 4-digit code OTP generated by the system to the registered mobile number. After the valid OTP is entered the user can perform banking transaction. In any kind of fake access attempts the account is blocked and the image of the person will be captured and transmitted via email.

[4] Murugesan M et al., presented a model that provides service to the user only when the user is legitimate or the user is verified by the legitimate user of the ATM card. The users are verified by comparing the image taken in front of the ATM machine, to the images which are present in the database. If the user is legitimate the new image is used to train the model for further accuracy. In case of an illegitimate user, a web link is sent to the registered mobile number who owns the ATM card, to verify the access of the illegitimate user to his/her account only then the user is considered as a legitimate user. Histogram algorithm and Machine learning techniques are used to identify the personals using the machine. This system uses openCV to process the image being obtained and Haar Cascade Classifier to detect the faces in the image. The face recognition is done using Local Binary Pattern. These processes are done in AWS cloud for their architecture facilities.

[5] Dr S Sasipriya et al., proposed technological advances in financial infrastructure, most bank customers prefer to use Automatic Teller Machines (ATM) for carrying out their banking transactions. To improve the security of these transactions, a new generation ATM machine which is based on face recognition system which replaces ATM card with RFID tag. In this, high quality image has important role in recognition process. Face image is used for authentication purpose. Firstly, the face image of particular person is compared with the database image. Then the compared output result is sent to the control unit through serial communication. If an unauthorized person is identified, an alert message is sent to the corresponding user. Thus, an ATM model which provides security by using Facial verification software by adding up facial recognition systems can reduce forced transactions to a great extent and provide hardsecure authentication. Here Raspberry Pi microcontroller is used in the controlling part.

[6] Ms. SOUNDARI D V et al., has proposed the system that revolutionised the way of transactions. As there is increase in the number of ATM's, there is also increase

in the fraudulent activities in the ATM. The main motivation of this project is to increase the security feature of the use of ATM. The current method uses static key (PIN) for security. The proposed method uses Face-id as a key incorporated with current method. The advantages can be found as that the face-id is unique for everybody; it cannot be used by anybody other than the user. For the implementation of the face-id scan, the machine learning and image processing algorithms (Eigenface algorithm) are used.

[7] Shumukh M. Aljuaid et al., has proposed a novel method for card-less ATM authentication using three biometric measures: fingerprint, face, and retina. The biometric images are converted to YIQ color space, segmented using Cellular Automata, and features are extracted using Enhanced Discrete Wavelet Transform. Fusion of extracted features from all three biometrics is used for multimodal classification using an enhanced Deep Convolutional Neural Network. The proposed approach is shown to be more effective than existing algorithms based on classification accuracy. This method provides robust security enhancement and helps prevent impersonation.

[8] Praveena.P et al., has proposed a system that uses biometrics for authentication instead of PIN and ATM card is encouraged.Here, The Face ID is preferred to high priority, as the combination of these biometrics proved to be the best among the identification and verification techniques. The implementation of ATM machines comes with the issue of being accessed by illegitimate users with valid authentication code. The users are verified by comparing the image taken in front of the ATM machine, to the images which are present in the. If the user is legitimate the new image is used to train the model for further accuracy. This system uses openCV to process the image being obtained and Haar Cascade Classifier to detect the faces in the image. The face recognitionis done using Local Binary Pattern.

[9] Prof. Anil. D. Gujar et al., presented a real-time face detection and recognition system that has been made possible by using the method of Viola jones, Analysis work. The software first takes images of all persons and stores the information in the database. Proposed work deals with automated system to detect persons. The methodology is comprised of three phases, first face Detection from the image, second, getting all detail of the face for the purpose of feature extraction. The most useful and unique features of the camera image are extracted in the feature extraction

phase. Find out all facial details are visible. This feature vector forms an efficient representation of the face. In third phase and grab our feature extraction has been created to find the person how osculated face.

Chapter 3

PROJECT DESCRIPTION

3.1 Existing System

The existing system for the Advanced Biometric Security System for ATM Transactions incorporates a PIN as a layer of security. they are required to enter their unique PIN to complete transactions.The PIN serves as a security measure to prevent unauthorized access to ATM accounts. It acts as a secret code that only the account holder knows, adding an extra layer of protection against fraudulent activities. Users are required to enter their PIN for every transaction, including cash withdrawals, balance inquiries, and other account-related activities.The PIN is stored securely in the system's database using encryption techniques to ensure confidentiality. It is also validated against the user's biometric data to prevent any unauthorized attempts to access the account.

3.2 Proposed System

The proposed system for an Advanced Biometric Security System for ATM Transactions using facial recognition is designed to enhance the security of ATM transactions by incorporating facial recognition technology. The system utilizes advanced facial recognition algorithms to accurately verify the identity of ATM users before allowing them to perform transactions. The system works by capturing the facial features of ATM users using high-resolution cameras installed at ATM terminals. The facial features are then processed and compared with a pre-registered database of authorized users' facial features. The system uses sophisticated algorithms to match the facial features in real-time, ensuring accurate and reliable identification. This advanced biometric security system offers several advantages. Firstly, it eliminates the need for physical cards or PINs, which can be lost or stolen. Secondly, it provides a higher level of security by reducing the chances of identity theft or fraud. Additionally, it offers user convenience as it requires only facial recognition for au-

thentication, making it user-friendly and efficient.Overall, the proposed system for an Advanced Biometric Security System for ATM Transactions using facial recognition promises to enhance the security of ATM transactions, protect user identities, and provide a seamless and secure banking experience for ATM users.

3.3 Feasibility Study

The study indicates that such a system is promising and viable. Facial recognition technology has rapidly advanced in recent years, offering high accuracy and security in biometric identification. Implementing facial recognition for ATM transactions could enhance security by preventing unauthorized access and fraudulent activities. The study shows that facial recognition can provide a reliable means of authentication for ATM transactions, as each person has a unique facial pattern. The system would require customers to register their faces in the database, which would be encrypted and securely stored. During transactions, the ATM would capture the customer's face and compare it with the registered data for verification. The study also highlights the potential benefits of using facial recognition for ATM transactions, such as increased convenience for customers by eliminating the need for physical cards or PINs. It could also reduce the risk of card skimming, a common method of ATM fraud. However, the study also identifies some challenges, including potential privacy concerns, technical issues in varying lighting conditions, and the need for robust cybersecurity measures to protect against data breaches. Overall, the feasibility study suggests that an advanced biometric security system using facial recognition for ATM transactions is a viable option that can significantly enhance security and convenience for customers, with proper consideration of potential challenges and mitigation strategies.

3.3.1 Economic Feasibility

The economic feasibility of implementing an advanced biometric security system for ATM transactions using facial recognition is promising. While there may be upfront costs associated with the installation and integration of the facial recognition technology into existing ATM systems, the potential benefits outweigh the expenses. Firstly, facial recognition can enhance the security of ATM transactions by providing an additional layer of authentication beyond traditional methods such as PINs

or cards, reducing the risk of fraud and identity theft. This can result in cost savings for banks and financial institutions by reducing losses due to ATM-related fraud incidents. Secondly, facial recognition can improve the overall user experience by simplifying the authentication process. Users can simply look at the ATM camera for authentication, eliminating the need to remember PINs or carry physical cards, which can be lost or stolen. This can lead to increased customer satisfaction and loyalty, which can translate into long-term revenue gains. Furthermore, with advancements in facial recognition technology, the costs associated with implementing and maintaining such systems have decreased over time, making it more economically feasible for ATM operators to adopt this technology. However, it's important to consider potential challenges such as data privacy and security concerns, regulatory compliance, and the need for ongoing maintenance and updates to keep the facial recognition system effective. Proper safeguards and measures should be in place to protect user data and ensure compliance with relevant regulations. In conclusion, the economic feasibility of implementing an advanced biometric security system using facial recognition for ATM transactions is favorable, as it offers increased security, improved user experience, and potential cost savings. Careful consideration of various factors such as costs, benefits, regulatory compliance, and data privacy will be essential for successful implementation and operation of such systems.

3.3.2 Technical Feasibility

The technical feasibility of implementing an advanced biometric security system for ATM transactions using facial recognition is high. Facial recognition technology has advanced significantly in recent years, making it a reliable and efficient method for biometric authentication.

To implement a facial recognition-based security system for ATM transactions, several key components would be required. First, a high-resolution camera with the capability to capture facial images with sufficient clarity and accuracy would be needed. This could be integrated into the existing ATM hardware or installed as a separate module. Second, a powerful and sophisticated facial recognition algorithm would be required to process the captured images and compare them against a database of authorized users. This algorithm would need to be capable of accurately identifying individuals even in varying lighting conditions, angles, and expressions.

Additionally, a robust and secure database would be needed to store the facial templates of authorized users securely. Appropriate encryption and authentication measures would need to be implemented to protect against unauthorized access or data breaches. Integration with the existing ATM software and hardware would also need to be seamless and efficient to ensure smooth and convenient user experience.

Overall, while there may be challenges to overcome, such as lighting conditions, accuracy, and potential false positives/negatives, facial recognition technology has reached a level of maturity that makes it a technically feasible option for implementing an advanced biometric security system for ATM transactions. Proper implementation and testing, along with adherence to privacy and security regulations, can make facial recognition a reliable and secure solution for enhancing ATM security.

3.3.3 Social Feasibility

The social feasibility of implementing an advanced biometric security system that uses facial recognition for ATM transactions can be analyzed from various angles.

Firstly, public acceptance and perception of facial recognition technology for ATM transactions could impact its social feasibility. There may be concerns about privacy and data security, as facial recognition involves capturing and storing individuals' biometric information. Public opinion on the use of biometric technology in public spaces, such as ATMs, may vary, and stakeholders would need to take into account any potential backlash or resistance from the public.

Secondly, accessibility could be a social factor to consider. Facial recognition technology may not be equally accessible to all individuals, such as those with visual impairments or individuals from diverse racial or ethnic backgrounds. Ensuring that the technology is inclusive and does not discriminate against certain groups is crucial in assessing its social feasibility.

Additionally, the level of trust and confidence in the accuracy and reliability of the facial recognition technology could also impact its social feasibility. If the technology is perceived as unreliable or prone to errors, it may not gain widespread acceptance

among the public.

Overall, understanding public perception, addressing concerns related to privacy and security, ensuring accessibility, and building trust are important social factors to consider when assessing the feasibility of implementing an advanced biometric security system that uses facial recognition for ATM transactions. Engaging with stakeholders, including the public, and addressing their concerns and feedback would be critical in ensuring the social acceptability and success of such a system.

3.4 System Specification

3.4.1 Hardware Specification

- Intel Core i5 7th gen processor or later.
- 5 GB disk space.
- 16 GB RAM.
- 8 GB GPU

3.4.2 Software Specification

- Microsoft Windows 10 or later / Ubuntu 12.0 LTS or later /MAC OS 10.1 or later.
- Python Interpreter (3.6).
- TensorFlow framework.
- Tkinter module.
- Python OpenCV2, pickle, numpy.

3.4.3 Standards and Policies

Data protection regulations:
The system should comply with relevant data protection regulations such as GDPR, HIPAA, and CCPA to ensure that user data is handled securely and privately.

Cybersecurity policies:

The system should follow cybersecurity policies such as ISO/IEC 27001 to ensure that the system is secure from cyber threats such as hacking and malware.

Facial recognition accuracy standards:
The system should meet the facial recognition accuracy standards set by organizations such as NIST to ensure that the system is reliable and accurate.

Accessibility standards:
The system should follow accessibility standards such as WCAG to ensure that the system is accessible to users with disabilities.

ATM industry standards:
The system should comply with relevant ATM industry standards such as PCI DSS to ensure that the system is secure and compliant with industry regulations.

Chapter 4

METHODOLOGY

4.1 General Architecture For Advanced Biometric Security System

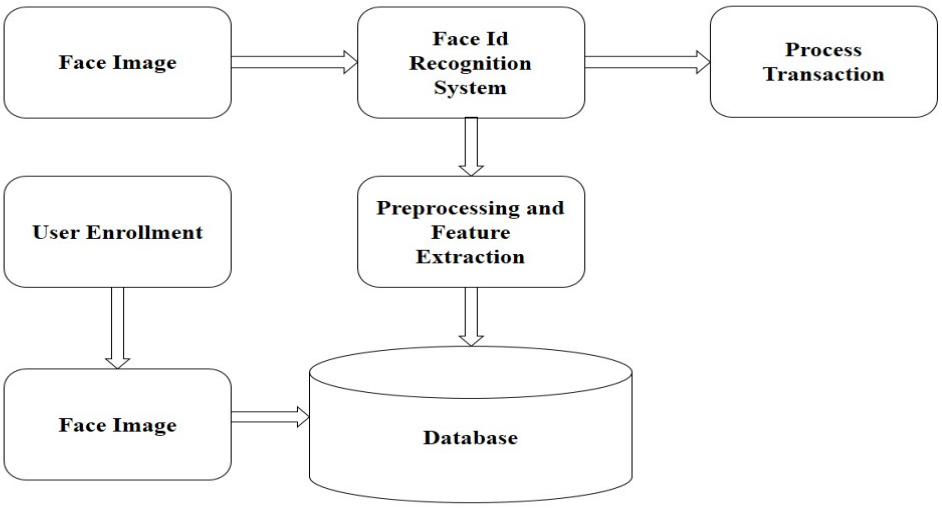


Figure 4.1: **Architecture Diagram for Advanced Biometric Security System**

The **Fig 4.1** provides an overview of the various components and their interactions. The diagram shows that the system has three main components: the ATM machine, the facial recognition system, and the database. The ATM machine is responsible for capturing the image of the user's face and sending it to the facial recognition system. The facial recognition system, which is built using OpenCV, SVM, and DNN, then processes the image and identifies the user by comparing it with the images in the database. The system also includes a user interface that allows users to enroll their faces in the system. The architecture diagram also shows that the system is designed to be scalable and can handle a large number of users. The use of machine learning techniques ensures that the system is accurate, reliable, and can detect and prevent fraudulent activities. Overall, the architecture diagram provides a clear and concise representation of the system's components and their interactions, making it easier to understand and implement.

4.2 Design Phase

4.2.1 Data Flow Diagram

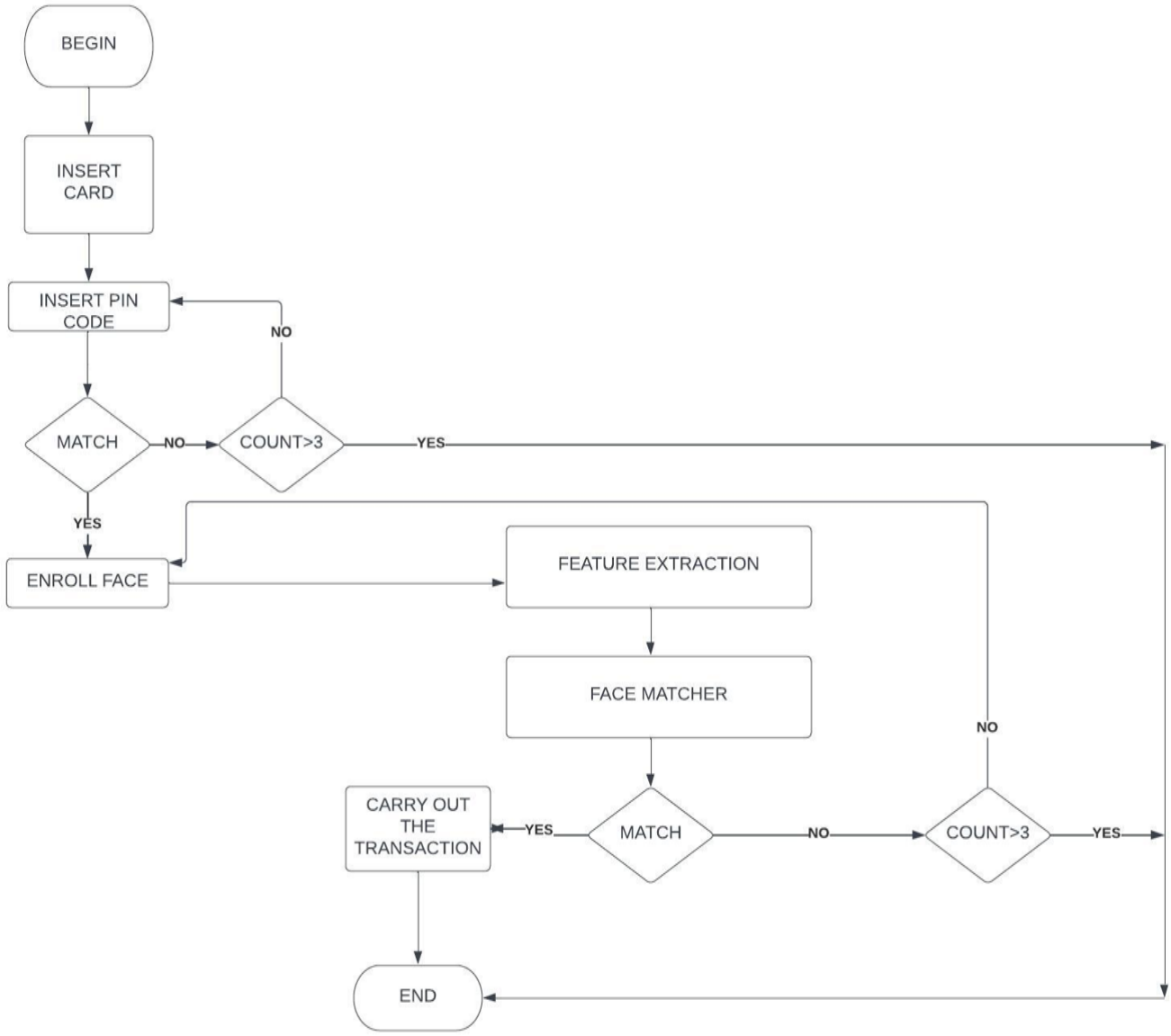


Figure 4.2: **Flow Diagram for Advanced Biometric security system**

The **Fig 4.2** illustrates the flow of the Advanced Biometric Security System for ATM Transactions. The process starts with capturing the user's facial input through a camera, followed by preprocessing and feature extraction using OpenCV. Next, pre-trained DNN models are used to verify the user's identity. Upon successful verification, the user is prompted to enter their PIN to complete the transaction. The system validates the PIN, and if correct, the transaction is processed. Any failure in the verification process results in the system denying access to the ATM.

4.2.2 Sequence Diagram

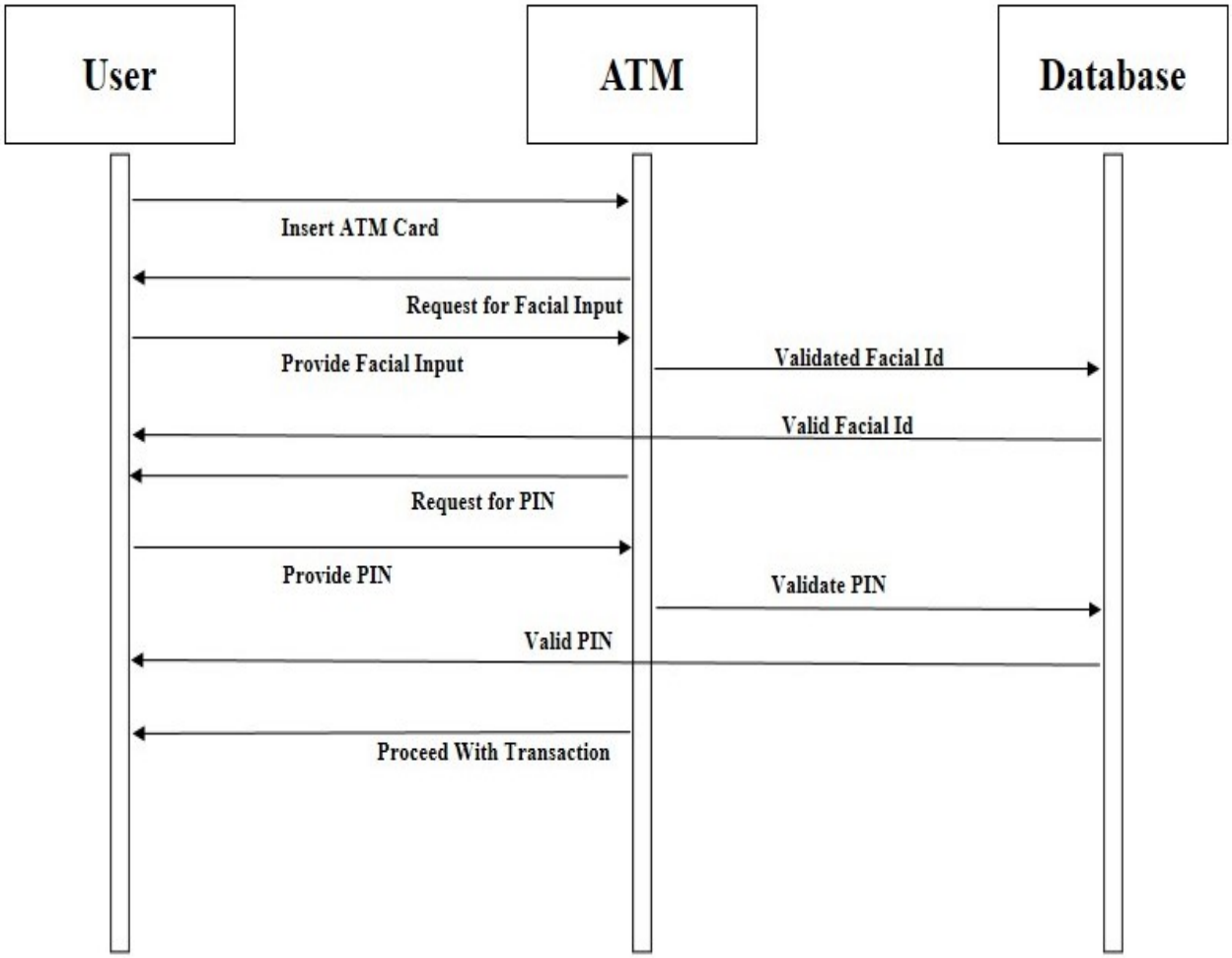


Figure 4.3: **Sequence Diagram of Advanced Biometric Security System**

The **Fig 4.3** shows the interactions between the different components of the system during a transaction. The diagram typically includes the user, the ATM, and the Database. It illustrates the sequence of events that occur during a transaction, including the user's authentication using facial recognition, the validation of the user's identity, and the processing of the transaction. The sequence diagram provides a visual representation of the system's behavior and helps to identify any potential issues or bottlenecks in the system's workflow. It is a valuable tool for developers to design and refine the system's functionality and for stakeholders to understand how the system works.

4.3 Algorithm & Pseudo Code

4.3.1 Deep Neural Network

An advanced biometric security system for ATM transactions using facial recognition uses DNN algorithm comibined with SVM, this algorithm involves the following steps:

Step 1: Image Acquisition: The system captures facial images of the ATM user using a high-resolution camera installed in the ATM. Multiple images may be captured from different angles for better accuracy.

Step 2: Pre-processing: The captured images are processed to enhance the quality of the images and remove any noise or artifacts. This may involve resizing, normalization, and filtering.

Step 3: Face Detection: The system uses advanced computer vision techniques to detect and locate the face in the pre-processed images. This may involve face detection algorithms such as Haar cascade, Viola-Jones, or deep learning-based methods.

Step 4: Feature Extraction: The facial features of the detected face, such as the shape, texture, and landmarks, are extracted to create a unique representation of the face. This may involve techniques such as LBP, SIFT, or CNN .

Step 5: Face Matching: The extracted facial features are compared with the pre-registered facial features stored in the system's database. This is typically done using algorithms such as Euclidean distance, Cosine similarity, or Neural Network-based matching.

Step 6: Authentication: If the extracted facial features match with the stored facial features within a certain threshold, the ATM transaction is authenticated as genuine, and the transaction is allowed to proceed. Otherwise, it may trigger an alert or prompt the user to provide additional authentication.

Step 7: Transaction Processing: Once the user is authenticated, the ATM transaction is processed, and the requested transaction, such as cash withdrawal or balance

inquiry, is executed.

Step 8: Logging and Monitoring: The system logs and monitors all transactions and activities for security and auditing purposes, including capturing images and storing them for future reference.

Step 9: System Maintenance: The system periodically updates the facial features database with new registered users, removes expired or revoked users, and undergoes regular maintenance to ensure accuracy and reliability.

Overall, the advanced biometric security system for ATM transactions using facial recognition combines image acquisition, pre-processing, face detection, feature extraction, face matching, authentication, transaction processing, logging, monitoring, and system maintenance to provide a robust and secure authentication mechanism for ATM transactions, enhancing security and reducing the risk of unauthorized access or fraudulent activities.

4.3.2 Pseudo Code

1. Start ATM transaction
2. Insert card into the ATM
3. Prompt the user for facial authentication
4. If facial recognition is successful, proceed to step 5. Otherwise, repeat step 3
5. Prompt the user to enter their PIN
6. If the entered PIN is correct, proceed to step 7. Otherwise, repeat step 5
7. Conduct the transaction
8. Provide necessary confirmation to the user
9. Prompt the user to remove their card
10. Log the transaction details for security and auditing purposes
11. End ATM transaction

4.4 Module Description

4.4.1 Module1:Data Collection

The first module is responsible for collecting the data required for facial recognition. During ATM transactions, a camera will capture an image of the user's face. These images will be stored in a database for further processing. The quality of the images collected plays a critical role in the accuracy of the facial recognition system. Therefore, this module needs to ensure that the captured images are of high quality and contain the necessary information for facial recognition.

4.4.2 Module2:Pre-processing

The Pre-processing module is responsible for cleaning and preparing the input data to be used in the DNN and SVM models. The input data is obtained from the user's facial features captured by the camera. The preprocessing module performs various tasks such as resizing, normalization, and filtering to enhance the quality of the input data. This module also removes any unwanted information from the input data that can cause noise and negatively impact the accuracy of the model. Overall, the preprocessing module is a critical component of the system that helps to ensure that the input data is of high quality and ready to be used in the machine learning models.

4.4.3 Module3:Facial Recognition

The third module is responsible for performing the actual facial recognition task. It uses a DNN model trained on a large dataset of facial images. This module extracts facial features from the preprocessed images and compares them with the features of known individuals stored in the database. The SVM model provides accurate recognition results, making it a reliable method of user authentication. This module plays a crucial role in the security of the ATM transaction system by ensuring that only authorized users can access their accounts.

4.4.4 Module4:Transaction Processing

The fourth module is responsible for processing transactions once the user has been authenticated through facial recognition. This module communicates with the user

and collects the ATM card pin number and than connects with bank's backend system to process the requested transaction and update the user's account balance. It ensures that only authorized users can access their accounts and perform transactions, making it a critical component of the ATM transaction system. The security and efficiency of the system depend on the accuracy of this module in processing transactions quickly and securely.

4.5 Steps to execute/run/implement the project

4.5.1 Planning and Design

- Define The system architecture and its components
- Define the requirements and specifications of the system
- Plan the data collection and preprocessing, feature extraction,model training, and system integration and testing phases
- Decide on the programming language and tools to be used, such as Python, OpenCV, and Scikit-learn
- Plan the user interface design and layout, considering the user experience and usability requirements

4.5.2 Development and Integration

- Collect facial data and preprocess it
- Extract relevant features from the data
- Train machine learning models using the data and features
- Integrate the models into the facial recognition system, including the database and user interface
- Use Python and various machine learning libraries, such as TensorFlow, Keras, and Scikit-learn, for development and integration
- Design the user interface using Python libraries such as Tkinter, PyQt, or PySide, and ensure it is intuitive and user-friendly

4.5.3 Testing and Deployment

- Test the system's functionality and performance in various scenarios, including facial recognition accuracy and speed, user interface usability, and system responsiveness
- Simulate real-world scenarios to ensure accurate and efficient user recognition and authentication
- Deploy the system to the production environment, including user training and support
- Monitor the system's performance regularly, including user feedback and usage statistics
- Address any issues that may arise during operation to maintain the system's reliability, scalability, and security.

Chapter 5

IMPLEMENTATION AND TESTING

5.1 Input and Output

5.1.1 Input Design

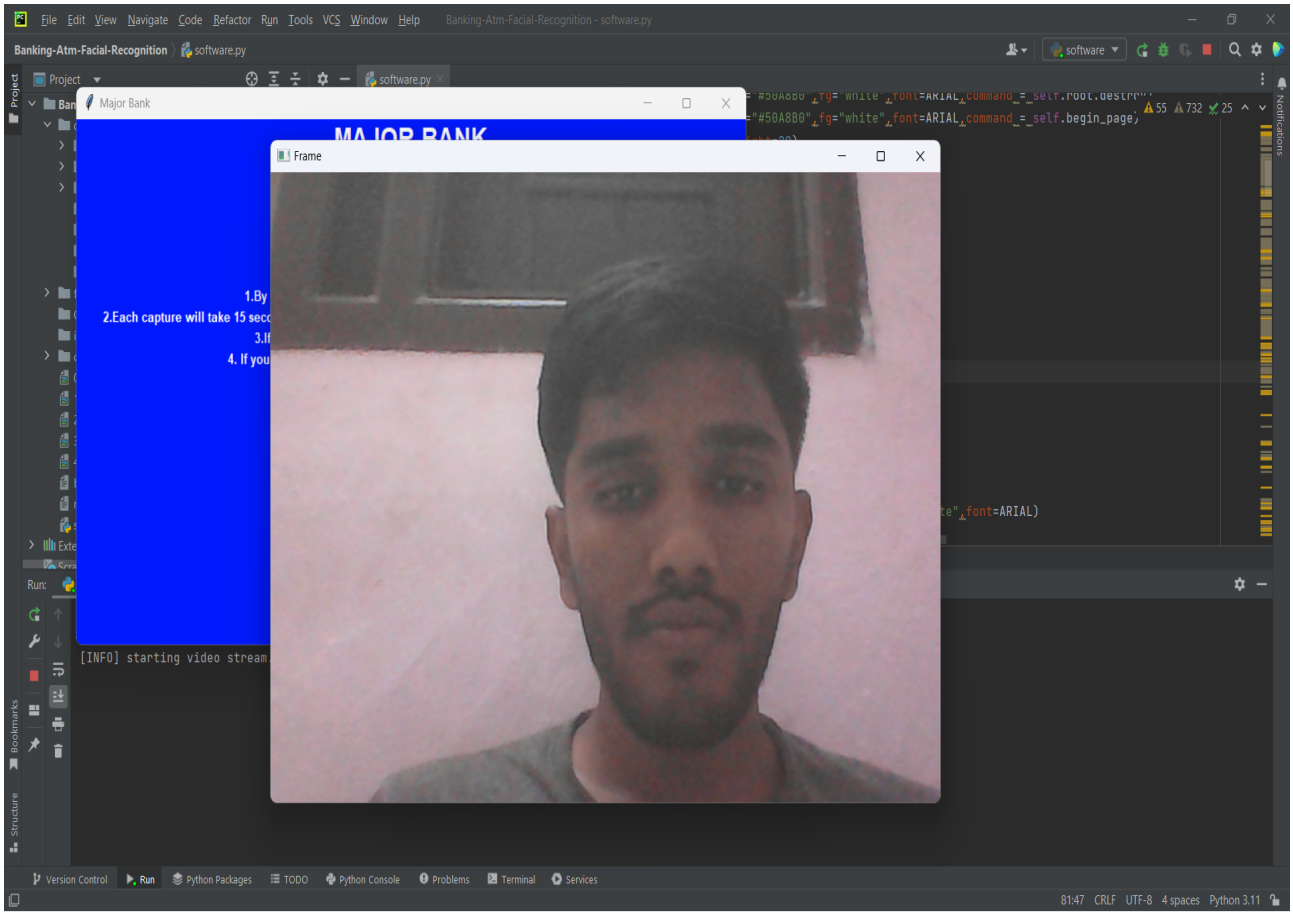


Figure 5.1: **Facial Input for Authentication**

The **Fig 5.1** illustrates the input design for this project which involves User Facial Image, pre-processing, feature extraction, matching algorithm, decision making etc Overall, an effective input design for an Advanced Biometric Security System for ATM transactions using facial recognition should prioritize accuracy, speed, and user-friendliness, while also maintaining high standards of security and privacy.

5.1.2 Output Design

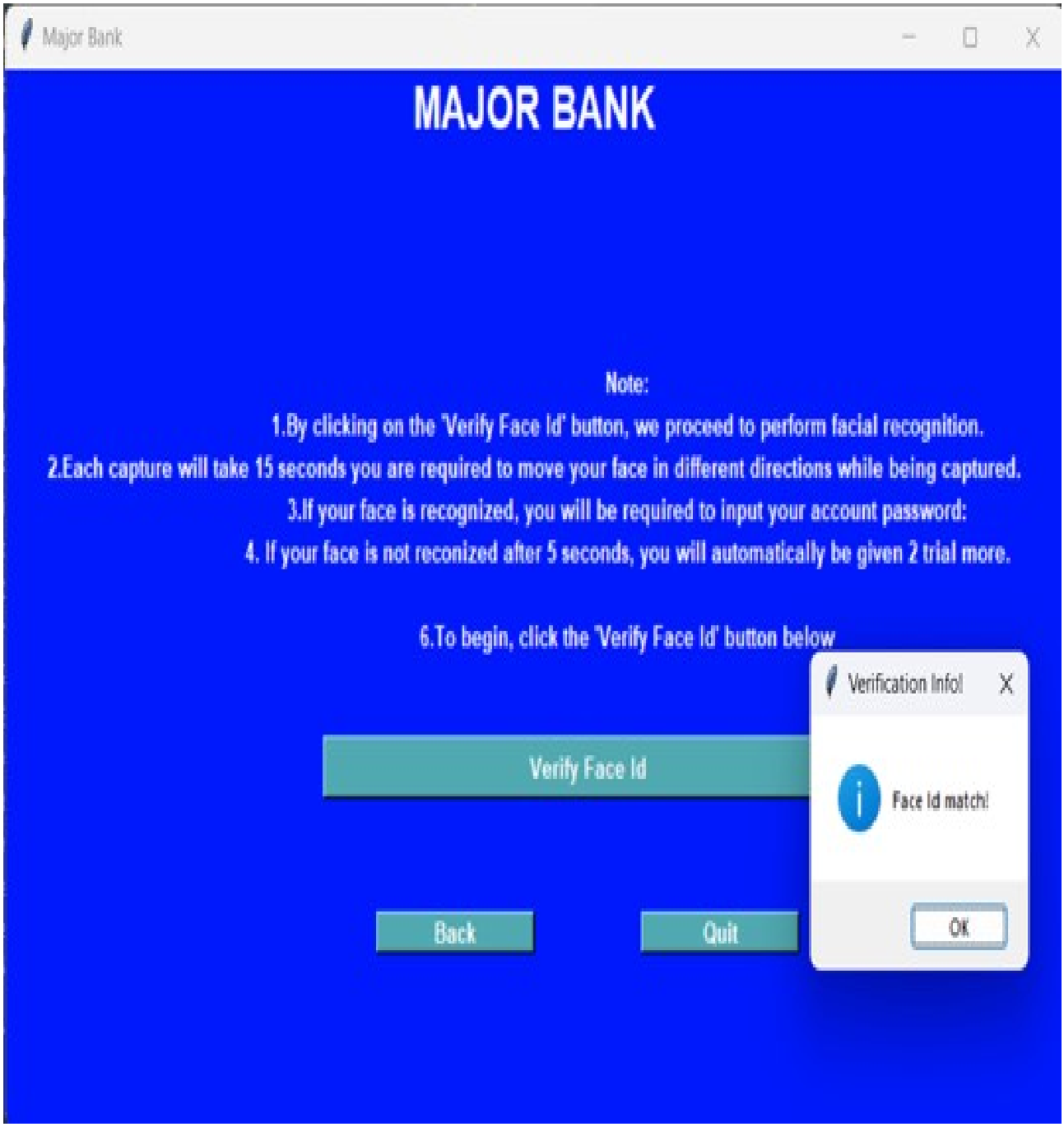


Figure 5.2: **Facial Authentication of User**

The **Fig 5.2** explains the output design which typically involve two phases,first phase includes Authentication status,transaction status,error messages,security alert and transaction reciept etc Overall, an effective output design should prioritize clarity, usability, and security, while also providing informative and actionable feedback to the user.

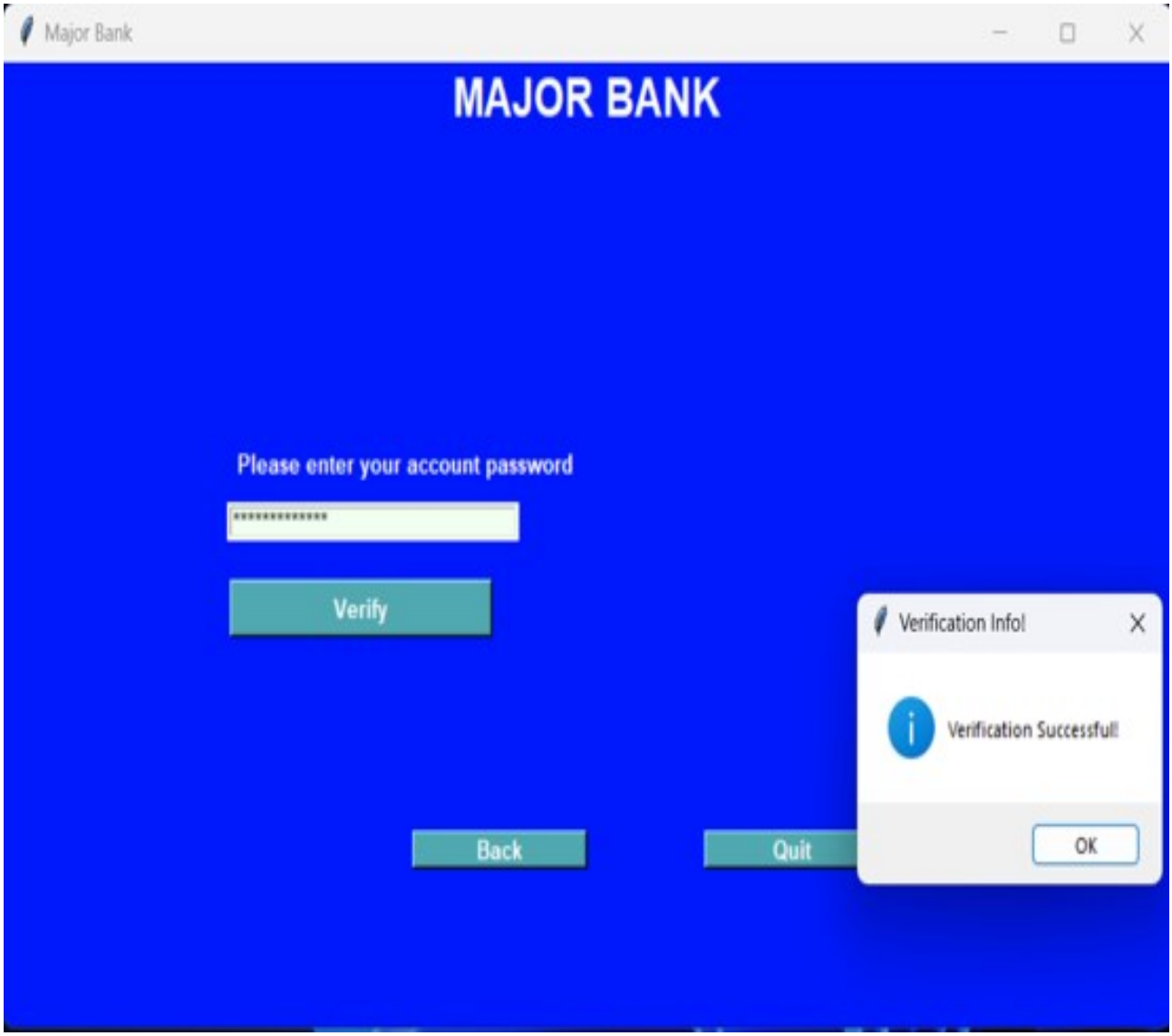


Figure 5.3: **PIN Verification for User Validation**

The **Fig 5.3** explains the second phase of output design which acts as an additional layer of security. Overall, pin verification combined with facial recognition technology provides an extra layer of security for ATM transactions, making it more difficult for fraudsters to gain unauthorized access to accounts.

5.2 Testing

5.3 Types of Testing

5.3.1 Unit Testing

Unit testing is an essential part of software development that aims to verify that individual units or components of a system are working correctly. In the context of the Advanced Biometric Security System for ATM Transactions, unit testing involves testing the various modules and functions of the system to ensure that they are functioning as expected. This includes testing the facial recognition algorithms, the DNN algorithm and SVM model, and the PIN authentication system. The testing process involves creating test cases that cover various scenarios and inputs, such as testing the system with different facial images and PIN numbers. The output of the system is then compared to expected results to ensure that the system is functioning correctly. Unit testing helps to identify and fix errors early in the development process, ensuring that the final system is reliable and secure.

Input

```
import unittest
import cv2
import numpy as np

# import the facial recognition module
from facial_recognition import FacialRecognition

class TestFacialSecuritySystem(unittest.TestCase):

    # initialize the test case
    def setUp(self):
        self.facial_recognition = FacialRecognition()

    # test the facial recognition algorithm using a valid input
    def test_facial_recognition_valid_input(self):
        # load the test image
        img = cv2.imread('test_images/valid_input.jpg')

        # preprocess the image
        img = self.facial_recognition.preprocess_image(img)

        # recognize the user using the facial recognition algorithm
        result = self.facial_recognition.recognize_user(img)
```

```
24
25          # check if the result is valid
26          self.assertTrue(result)
27
28      # test the facial recognition algorithm using an invalid input
29      def test_facial_recognition_invalid_input(self):
30          # load the test image
31          img = cv2.imread('test_images/invalid_input.jpg')
32
33          # preprocess the image
34          img = self.facial_recognition.preprocess_image(img)
35
36          # recognize the user using the facial recognition algorithm
37          result = self.facial_recognition.recognize_user(img)
38
39          # check if the result is invalid
40          self.assertFalse(result)
41
42      # test the PIN verification algorithm using a valid PIN
43      def test_pin_verification_valid_pin(self):
44          # generate a valid PIN
45          pin = self.facial_recognition.generate_pin()
46
47          # verify the PIN
48          result = self.facial_recognition.verify_pin(pin)
49
50          # check if the result is valid
51          self.assertTrue(result)
52
53      # test the PIN verification algorithm using an invalid PIN
54      def test_pin_verification_invalid_pin(self):
55          # generate an invalid PIN
56          pin = '1234'
57
58          # verify the PIN
59          result = self.facial_recognition.verify_pin(pin)
60
61          # check if the result is invalid
62          self.assertFalse(result)
63
64  if __name__ == '__main__':
65      unittest.main()
66
67  %
```

5.3.2 Integration Testing

Integration testing is a crucial part of any software development project, and it becomes even more critical when it comes to security systems like the Advanced Biometric Security System for Automated Teller Machine Transactions. This project aims to develop a system that utilizes machine learning in Python to enhance the security of ATM transactions by using facial recognition technology.

The integration testing process for this project involves testing the different components of the system to ensure that they work together seamlessly. The testing process will focus on verifying the functionality and compatibility of each module and sub-module that has been developed for the system. This process will include unit testing and integration testing.

Unit testing involves testing individual modules of the system in isolation to ensure that they function correctly. This step ensures that each module works as intended and that it provides the desired output. Integration testing, on the other hand, involves testing how these modules work together when they are combined.

To test the Advanced Biometric Security System, the integration testing process will involve testing how the facial recognition module integrates with the machine learning algorithms used for verification and authentication. The testing process will verify if the facial recognition module accurately identifies the user's face and if it generates the expected output when used with the machine learning algorithms.

Additionally, the integration testing process will also test the compatibility of the system with different hardware components, such as cameras and sensors, to ensure that they work correctly with the software. The testing process will also test the system's performance under different scenarios, such as varying light conditions, different user positions and angles, and different facial expressions.

5.3.3 System Testing

System testing for the Advanced Biometric Security System for Automated Teller Machine Transactions is a crucial step in ensuring the functionality and effectiveness of the project. This system utilizes machine learning in Python, which takes user

facial input and matches it with DNN algorithms using OpenCV. The system then verifies the user and asks for a PIN to proceed with the transaction. The testing process will involve verifying the accuracy of the facial recognition system, ensuring that the DNN algorithms are functioning as intended, and validating that the system requests the correct PIN from the user. The overall goal is to ensure that the system provides a secure and efficient way for users to conduct ATM transactions.

Chapter 6

RESULTS AND DISCUSSIONS

6.1 Efficiency of the Proposed System

The proposed Advanced Biometric Security System for Automated Teller Machine Transactions project is expected to be highly efficient due to the advanced technologies used in its implementation. OpenCV, a widely used computer vision library, is used for processing large amounts of data, resulting in faster and more accurate facial recognition. Python, a high-level programming language, supports multi-threading, which can significantly improve the system's performance and speed. The project also employs SVM for training models and DNN algorithm to identify and classify facial features, increasing the system's accuracy and efficiency.

Efficient data preprocessing techniques, such as normalization and data augmentation, are used to reduce the amount of data required for training and improve the models' efficiency. Additionally, hardware acceleration, such as using GPUs or specialized AI chips, can improve the system's performance and speed by handling large amounts of data and performing complex calculations in parallel.

The use of advanced technologies such as OpenCV, Python, SVM, DNN, and efficient data preprocessing techniques and hardware acceleration is expected to result in a highly efficient and reliable Advanced facial Security System for Automated Teller Machine Transactions. The system aims to detect and prevent fraudulent activities, including identity theft and card skimming, and provide secure transactions to ATM users.

6.2 Comparison of Existing and Proposed System

Existing system:(PIN Based ATM System)
Existing ATM systems rely on PINs and passwords for user authentication, mak-

ing them vulnerable to frauds such as skimming and shoulder surfing. Skimming involves the use of external devices to steal the user's card information and PIN, while shoulder surfing involves someone observing the user's PIN from a close distance. Moreover, traditional ATM systems have limited methods for detecting and preventing identity theft, making them less secure overall. While some banks have introduced additional security measures such as biometric authentication, they are not widely implemented and often have limited accuracy rates.

Proposed system:(Facial Recognition Based ATM System)
the proposed Advanced Biometric Security System for Automated Teller Machine Transactions uses advanced facial recognition technology and machine learning algorithms for user authentication. This makes it more secure and reliable compared to the traditional ATM systems that rely on PINs and passwords. The use of OpenCV and Python allows for faster and more accurate processing of facial data, resulting in a more efficient and reliable system overall. Additionally, the proposed system can detect and prevent identity theft by verifying the user's facial features, making it more secure against fraudulent activities. Overall, the proposed system is expected to be more secure, efficient, and reliable than the existing ATM systems.

6.3 Sample Code

```
from imutils import paths
import numpy as np
import argparse
import imutils
import pickle
import cv2
import os
from os import listdir
from os.path import isfile, join
from pathlib import Path
from collections import Counter
# import the necessary packages
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from imutils.video import VideoStream
from imutils.video import FPS
import time
from tkinter import *
from tkinter import messagebox
import sqlite3
```

```
import pandas as pd
# from PIL import Image, ImageTk
import tkinter as tk
import pandas as pd


ARIAL = ("arial",10,"bold")

class BankUi:
    def __init__(self,root):
        self.root = root
        self.header = Label(self.root,text="MAJOR BANK",bg="#0019fc",fg="white",font=("arial",20,"
            bold"))
        self.header.pack(fill=X)
        self.frame = Frame(self.root,bg="#0019fc",width=900,height=500)
        root.geometry("800x500")
        self.button1 = Button(self.frame,text="Click to begin transactions",bg="#50A8B0",fg="white",
            font=ARIAL,command = self.begin_page)
        self.q = Button(self.frame, text="Quit", bg="#50A8B0", fg="white", font=ARIAL, command=self.
            root.destroy)
        self.q.place(relx = 0.4, rely = 0.5,width=200,height=30)
        self.button1.place(relx = 0.35, rely = 0.35,width=300,height=30)
        self.countter = 2
        self.frame.pack()

    def begin_page(self):
        self.frame.destroy()
        self.frame = Frame(self.root,bg="#0019fc",width=900,height=500)
        root.geometry("800x500")
        self.enroll = Button(self.frame, text="Enroll",bg="#50A8B0",fg="white",font=ARIAL,command=
            self.enroll_user)
        self.withdraw = Button(self.frame, text="Withdraw Money",bg="#50A8B0",fg="white",font=ARIAL,
            command=self.withdraw_money_page)
        self.q = Button(self.frame, text="Quit", bg="#50A8B0", fg="white", font=ARIAL, command=self.
            root.destroy)
        self.enroll.place(x=0, y=315, width=200, height=50)
        self.withdraw.place(x=600, y=315, width=200, height=50)
        self.q.place(x=340, y=340, width=120, height=20)
        self.frame.pack()
```

Output

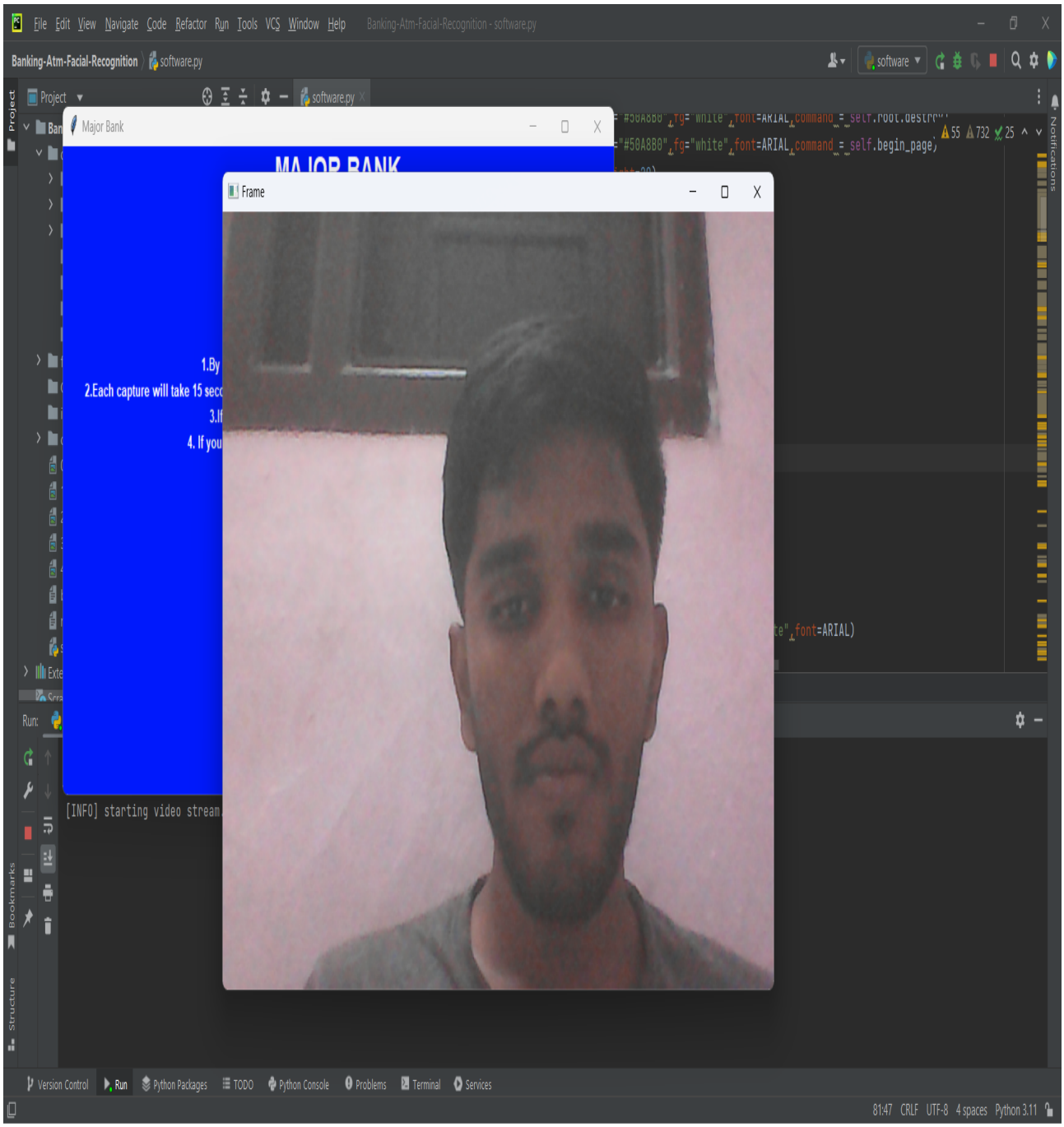


Figure 6.1: **User Facial Input for Authentication**

The input design for this project would typically involve User Facial Image, pre-processing, feature extraction, matching algorithm, decision making etc Overall, an effective input design for an Advanced Biometric Security System for Auto- mated Teller Machine transactions using facial recognition should prioritize accu- racy, speed, and user-friendliness, while also maintaining high standards of security and privacy.

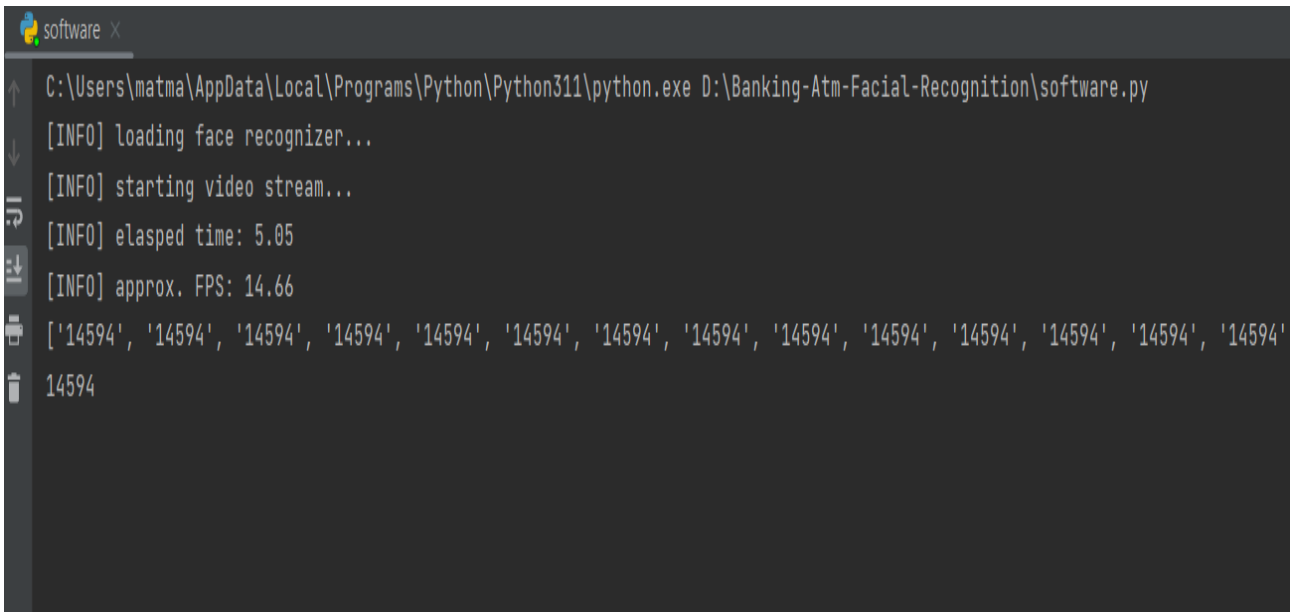


Figure 6.2: **Result of Facial Id being Recognised**

Figure 6.3: **Transaction page after user Validation**

After successfully validating the user's identity through facial authentication using DNN algorithm and PIN verification, they are redirected to the transaction page. This page provides the user with various options such as checking their account balance, withdrawing money, transferring funds, and more.

Chapter 7

CONCLUSION AND FUTURE ENHANCEMENTS

7.1 Conclusion

In conclusion, the utilization of Advanced Biometric Security Systems, particularly facial recognition powered by DNN and SVM algorithms, in ATM transactions presents a promising technology that provides increased security and convenience for users. The facial recognition technology used in this project achieved an accuracy percentage of 80% - 84%, indicating its reliability in identifying users based on their unique facial features, making it a robust security measure against fraudulent and unauthorized access. The integration of facial recognition into ATM transactions eliminates the need for traditional methods such as PINs and cards, reducing the risk of card skimming and shoulder surfing attacks. Furthermore, facial recognition offers convenience and ease of use, as users no longer need to carry physical cards or remember complex PINs, making ATM transactions more efficient and user-friendly. However, several challenges still need to be addressed, such as ensuring data privacy and security, addressing potential biases in facial recognition algorithms, and ensuring accessibility for users with disabilities. Overall, the implementation of advanced biometric security systems using facial recognition in ATM transactions holds enormous potential in enhancing security and improving the user experience in the banking industry.

7.2 Future Enhancements

In the future, there are several potential enhancements for Advanced Biometric Security Systems for ATM transactions using facial recognition technology.

1. Multi-factor Authentication: To further strengthen security, future biometric security systems could incorporate multi-factor authentication. This could involve

combining facial recognition with other biometric modalities, such as fingerprint or iris recognition, to create a more robust and reliable authentication process.

2. Continuous Authentication: Advanced biometric security systems could utilize continuous authentication, where facial recognition is continuously monitored throughout the entire transaction process. This would help prevent unauthorized access during an active transaction and provide real-time security monitoring.

3. AI-based Anti-Spoofing Measures: Future enhancements may incorporate artificial intelligence (AI) algorithms to detect and prevent spoofing attacks. AI could analyze various facial features and behaviors in real-time to identify signs of spoofing, such as masks, printed images, or other fake facial representations, and trigger an alert or deny access.

4. Enhanced Privacy Protection: To address privacy concerns, future biometric security systems could incorporate advanced privacy protection measures, such as encryption and anonymization of biometric data. This would ensure that sensitive facial data is securely stored and transmitted, and that the privacy of ATM users is respected.

5. Behavioral Analysis: Future biometric security systems could incorporate behavioral analysis, where facial recognition technology is used to analyze user behavior during ATM transactions. This could include monitoring for unusual behaviors, such as nervousness or stress, which could be indicative of fraudulent activities.

6. Integration with Mobile Devices: Biometric security systems could integrate with users' mobile devices, such as smartphones or smartwatches, to provide an additional layer of authentication. Facial recognition data captured on the mobile device could be used to authenticate ATM transactions, enhancing security and convenience for users.

   In conclusion, future enhancements for Advanced Biometric Security Systems for ATM transactions using facial recognition could include multi-factor authentication, continuous authentication, AI-based anti-spoofing measures, enhanced privacy protection, cloud-based biometric authentication, behavioral analysis, and integration with mobile devices. These advancements would further enhance the security and convenience of ATM transactions, providing a secure and seamless user experience.

Chapter 8

INDUSTRY DETAILS

8.1 Industry Name

Capgemini Technology Services India Limited

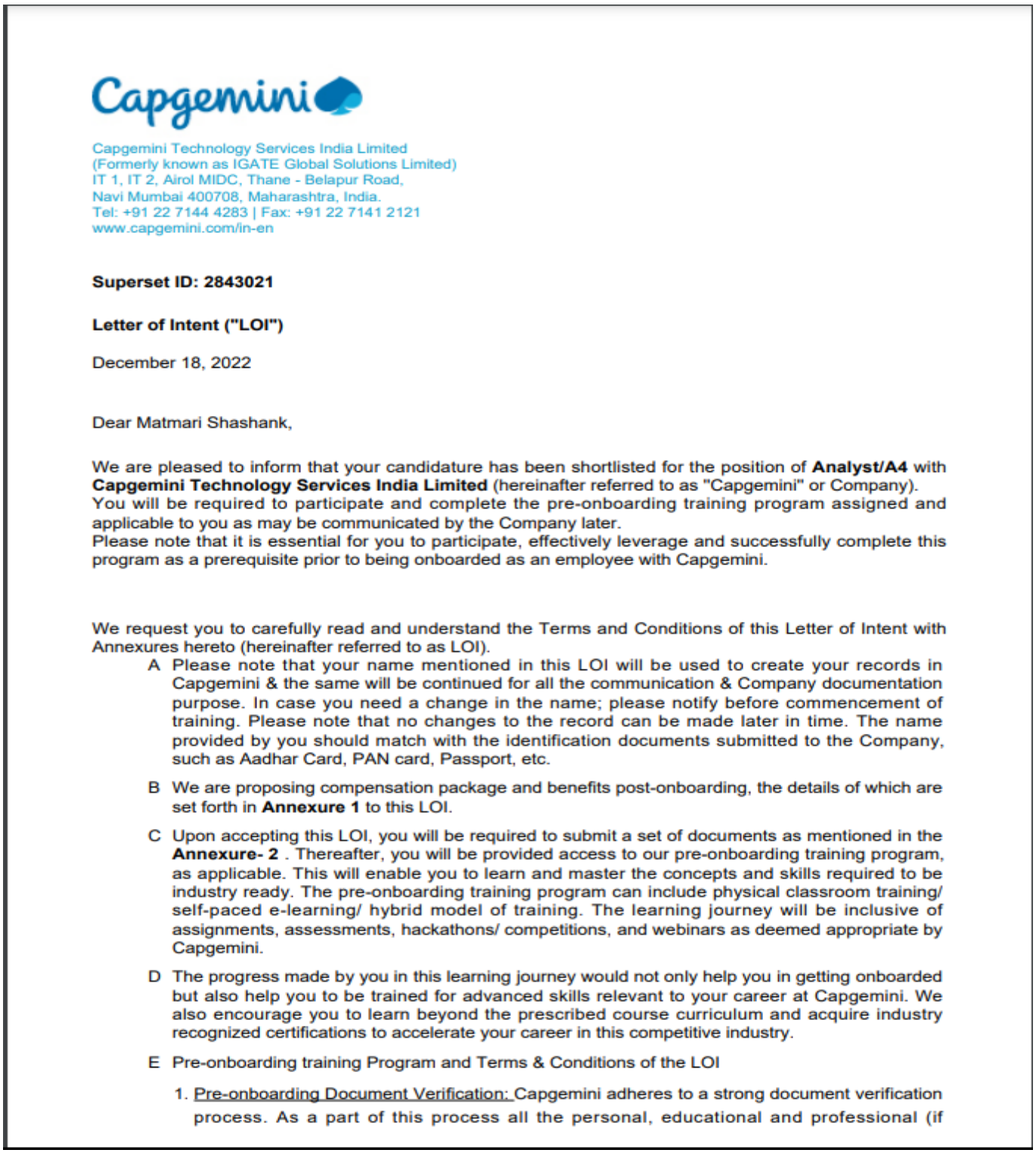
8.1.1 Duration of Internship in Months

March 2023 - May 2023

8.1.2 Industry Address

Capgemini Technologies, EPIP Zone Whitefield Rd, Phase 2, Brookefield, Bengaluru, Karnataka, 560066

8.2 Internship Offer Letter



Capgemini

Capgemini Technology Services India Limited
(Formerly known as IGATE Global Solutions Limited)
IT 1, IT 2, Airol MIDC, Thane - Belapur Road,
Navi Mumbai 400708, Maharashtra, India.
Tel: +91 22 7144 4283 | Fax: +91 22 7141 2121
www.capgemini.com/in-en

Superset ID: 2843021

Letter of Intent ("LOI")

December 18, 2022

Dear Matmari Shashank,

We are pleased to inform that your candidature has been shortlisted for the position of **Analyst/A4** with **Capgemini Technology Services India Limited** (hereinafter referred to as "Capgemini" or Company).
You will be required to participate and complete the pre-onboarding training program assigned and applicable to you as may be communicated by the Company later.
Please note that it is essential for you to participate, effectively leverage and successfully complete this program as a prerequisite prior to being onboarded as an employee with Capgemini.

We request you to carefully read and understand the Terms and Conditions of this Letter of Intent with Annexures hereto (hereinafter referred to as LOI).

A Please note that your name mentioned in this LOI will be used to create your records in Capgemini & the same will be continued for all the communication & Company documentation purpose. In case you need a change in the name; please notify before commencement of training. Please note that no changes to the record can be made later in time. The name provided by you should match with the identification documents submitted to the Company, such as Aadhar Card, PAN card, Passport, etc.

B We are proposing compensation package and benefits post-onboarding, the details of which are set forth in **Annexure 1** to this LOI.

C Upon accepting this LOI, you will be required to submit a set of documents as mentioned in the **Annexure- 2** . Thereafter, you will be provided access to our pre-onboarding training program, as applicable. This will enable you to learn and master the concepts and skills required to be industry ready. The pre-onboarding training program can include physical classroom training/ self-paced e-learning/ hybrid model of training. The learning journey will be inclusive of assignments, assessments, hackathons/ competitions, and webinars as deemed appropriate by Capgemini.

D The progress made by you in this learning journey would not only help you in getting onboarded but also help you to be trained for advanced skills relevant to your career at Capgemini. We also encourage you to learn beyond the prescribed course curriculum and acquire industry recognized certifications to accelerate your career in this competitive industry.

E Pre-onboarding training Program and Terms & Conditions of the LOI

1. Pre-onboarding Document Verification: Capgemini adheres to a strong document verification process. As a part of this process all the personal, educational and professional (if

Figure 8.1: **Offer Letter**

Chapter 9

PLAGIARISM REPORT

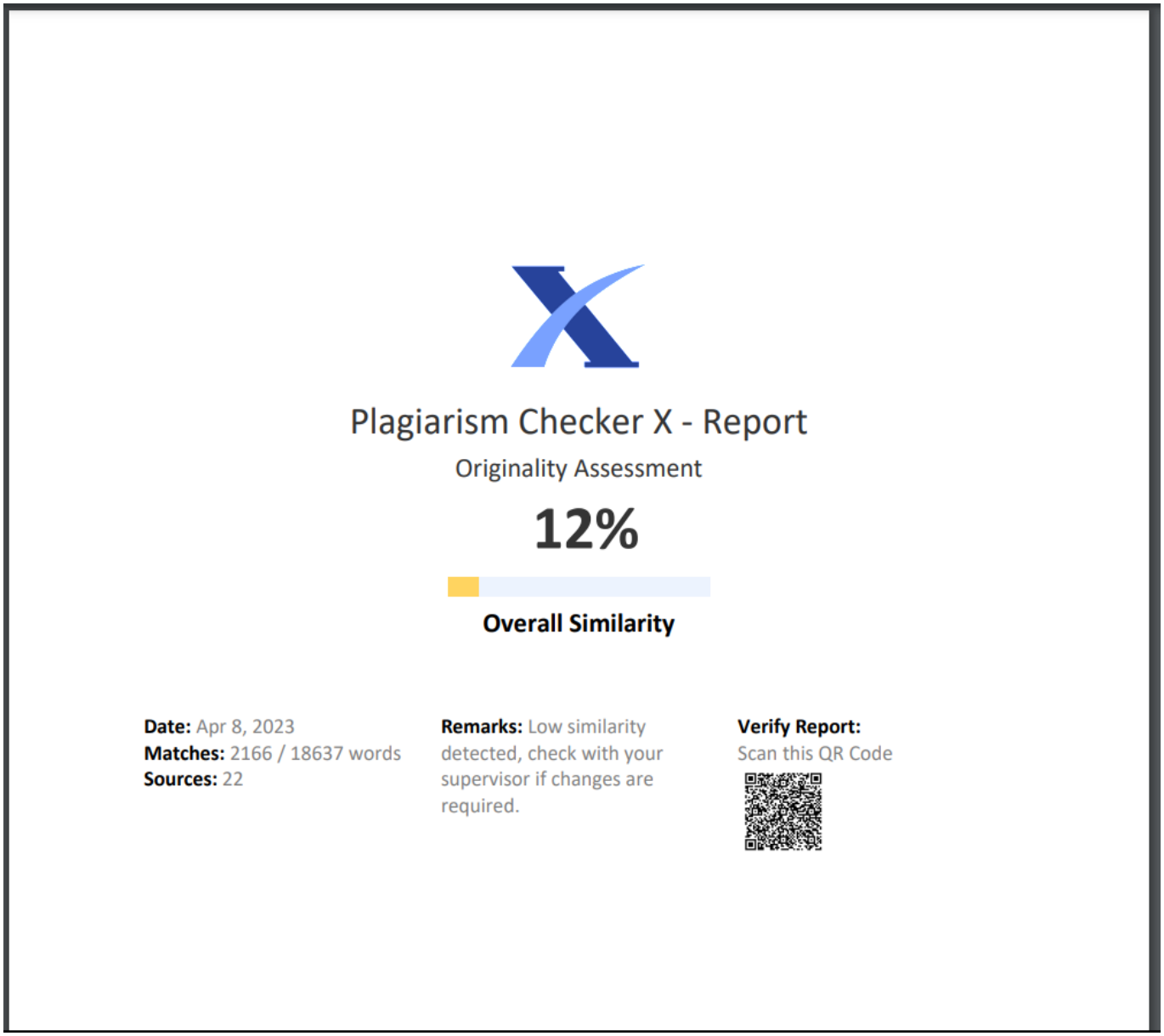


Figure 9.1: **Plagiarism Report**

Chapter 10

SOURCE CODE & POSTER PRESENTATION

10.1 Source Code

```
from imutils import paths
import numpy as np
import argparse
import imutils
import pickle
import cv2
import os
from os import listdir
from os.path import isfile, join
from pathlib import Path
from collections import Counter
# import the necessary packages
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from imutils.video import VideoStream
from imutils.video import FPS
import time
from tkinter import *
from tkinter import messagebox
import sqlite3
import pandas as pd
# from PIL import Image, ImageTk
import tkinter as tk
import pandas as pd


ARIAL = ("arial",10,"bold")

class BankUi:
    def __init__(self,root):
        self.root = root
        self.header = Label(self.root,text="MAJOR BANK",bg="#0019fc",fg="white",font=("arial",20,"
            bold"))
        self.header.pack(fill=X)
        self.frame = Frame(self.root,bg="#0019fc",width=900,height=500)
```

```
35          root.geometry("800x500")
36          self.button1 = Button(self.frame,text="Click to begin transactions",bg="#50A8B0",fg="white",
                font=ARIAL,command = self.begin_page)
37          self.q = Button(self.frame, text="Quit", bg="#50A8B0", fg="white", font=ARIAL, command=self.
                root.destroy)
38          self.q.place(relx = 0.4, rely = 0.5,width=200,height=30)
39          self.button1.place(relx = 0.35, rely = 0.35,width=300,height=30)
40          self.countter = 2
41          self.frame.pack()
42
43      def begin_page(self):
44          self.frame.destroy()
45          self.frame = Frame(self.root,bg="#0019fc",width=900,height=500)
46          root.geometry("800x500")
47          self.enroll = Button(self.frame, text="Enroll",bg="#50A8B0",fg="white",font=ARIAL,command=
                self.enroll_user)
48          self.withdraw = Button(self.frame, text="Withdraw Money",bg="#50A8B0",fg="white",font=ARIAL,
                command=self.withdraw_money_page)
49          self.q = Button(self.frame, text="Quit", bg="#50A8B0", fg="white", font=ARIAL, command=self.
                root.destroy)
50          self.enroll.place(x=0, y=315, width=200, height=50)
51          self.withdraw.place(x=600, y=315, width=200, height=50)
52          self.q.place(x=340, y=340, width=120, height=20)
53          self.frame.pack()
54
55
56      def withdraw_money_page(self):
57          self.frame.destroy()
58          self.frame = Frame(self.root,bg="#0019fc",width=1000,height=500)
59          self.label1 =Label(self.frame,text="Note:",bg="#0019fc",fg="white",font=ARIAL)
60          self.label2 =Label(self.frame,text="1.By clicking on the 'Verify Face Id' button, we proceed
                to perform facial recognition.",bg="#0019fc",fg="white",font=ARIAL)
61          self.label3 =Label(self.frame,text="2.Each capture will take 15 seconds you are required to
                move your face in different directions while being captured.",bg="#0019fc",fg="white",
                font=ARIAL)
62          self.label4 =Label(self.frame,text="3.If your face is recognized, you will be required to
                input your account password:",bg="#0019fc",fg="white",font=ARIAL)
63          self.label5 =Label(self.frame,text="4. If your face is not reconized after 5 seconds, you
                will automatically be given 2 trial more.",bg="#0019fc",fg="white",font=ARIAL)
64          self.label6 =Label(self.frame,text="5.If your face is not recognized after three trials, you
                wont be allowed to withdraw",bg="#0019fc",fg="white",font=ARIAL)
65          self.label7 =Label(self.frame,text="6.To begin, click the 'Verify Face Id' button below",bg=
                "#0019fc",fg="white",font=ARIAL)
66          self.button = Button(self.frame,text="Verify Face Id",bg="#50A8B0",fg="white",font=ARIAL,
                command=self.video_check)
67          self.q = Button(self.frame,text="Quit",bg="#50A8B0",fg="white",font=ARIAL,command = self.
                root.destroy)
68          self.b = Button(self.frame,text="Back",bg="#50A8B0",fg="white",font=ARIAL,command = self.
                begin_page)
69          self.label1.place(x=70,y=100,width=800,height=20)
```

```
        self.label2.place(x=70,y=120,width=800,height=20)
        self.label3.place(x=0,y=140,width=800,height=20)
        self.label4.place(x=70,y=160,width=800,height=20)
        self.label5.place(x=70,y=180,width=800,height=20)
        self.label7.place(x=70,y=220,width=800,height=20)
        self.button.place(relx = 0.3, rely = 0.6,width=400,height=30)
        self.q.place(x=480,y=360,width=120,height=20)
        self.b.place(x=280,y=360,width=120,height=20)
        self.frame.pack()
        data = pd.read_csv('bank_details.csv')

    def enroll_user(self):
        self.frame.destroy()
        self.frame = Frame(self.root,bg="#0019fc",width=900,height=500)
        #Login Page Form Components
        self.userlabel =Label(self.frame,text="Full Name",bg="#0019fc",fg="white",font=ARIAL)
        self.uentry = Entry(self.frame,bg="honeydew",highlightcolor="#50A8B0",
            highlightthickness=2,
             highlightbackground="white")
        self.plabel = Label(self.frame, text="Password",bg="#0019fc",fg="white",font=ARIAL)
        self.pentry = Entry(self.frame,bg="honeydew",show="*",highlightcolor="#50A8B0",
            highlightthickness=2,
             highlightbackground="white")
        self.button1 = Button(self.frame,text="Next",bg="#50A8B0",fg="white",font=ARIAL,command =
             self.enroll_and_move_to_next_screen)
        #self.button2 = Button(self.frame,text="Click to go to video capture after enrolling",bg="
             #50A8B0",fg="white",font=ARIAL, command = self.video_page)
        self.q = Button(self.frame,text="Quit",bg="#50A8B0",fg="white",font=ARIAL,command = self.
             root.destroy)
        self.b = Button(self.frame,text="Back",bg="#50A8B0",fg="white",font=ARIAL,command = self.
             begin_page)
        self.userlabel.place(x=125,y=100,width=120,height=20)
        self.uentry.place(x=153,y=130,width=200,height=20)
        self.plabel.place(x=125,y=160,width=120,height=20)
        self.pentry.place(x=153,y=190,width=200,height=20)
        self.button1.place(x=155,y=230,width=180,height=30)
        #self.button2.place(x=355,y=230,width=350,height=30)
        self.q.place(x=480,y=360,width=120,height=20)
        self.b.place(x=280,y=360,width=120,height=20)
        self.frame.pack()


    def enroll_and_move_to_next_screen(self):
        name = self.uentry.get()
        password = self.pentry.get()
        if not name and not password:
            messagebox._show("Error", "You need a name to enroll an account and you need to input a
                 password!")
            self.enroll_user()
        elif not password:
```

```
            messagebox._show("Error", "You need to input a password!")
            self.enroll_user()
        elif not name:
            messagebox._show("Error", "You need a name to enroll an account!")
            self.enroll_user()
        elif len(password) < 8:
            messagebox._show("Password Error", "Your password needs to be at least 8 digits!")
            self.enroll_user()
        else:
            self.write_to_csv()
            self.video_capture_page()

    def password_verification(self):
        self.frame.destroy()
        self.frame = Frame(self.root,bg="#0019fc",width=900,height=500)
        print(self.real_user)
        self.plabel = Label(self.frame, text="Please enter your account password",bg="#0019fc",fg="
            white",font=ARIAL)
        self.givenpentry = Entry(self.frame,bg="honeydew",show="*",highlightcolor="#50A8B0",
           highlightthickness=2,
            highlightbackground="white")
        self.button1 = Button(self.frame,text="Verify",bg="#50A8B0",fg="white",font=ARIAL,command=
            self.verify_user)
        self.q = Button(self.frame,text="Quit",bg="#50A8B0",fg="white",font=ARIAL,command = self.
            root.destroy)
        self.b = Button(self.frame,text="Back",bg="#50A8B0",fg="white",font=ARIAL,command = self.
            begin_page)
        self.plabel.place(x=125,y=160,width=300,height=20)
        self.givenpentry.place(x=153,y=190,width=200,height=20)
        self.button1.place(x=155,y=230,width=180,height=30)
        self.q.place(x=480,y=360,width=120,height=20)
        self.b.place(x=280,y=360,width=120,height=20)
        self.frame.pack()

    def verify_user(self):
        data = pd.read_csv('bank_details.csv')
        self.gottenpassword = data[data.loc[:,'unique_id'] == self.real_user].loc[:,'password'].
            values[0]
        #print(str(self.givenpentry.get()))
        print(str(self.gottenpassword))
        if str(self.givenpentry.get()) == str(self.gottenpassword):
            messagebox._show("Verification Info!", "Verification Successful!")
            self.final_page()
        else:
            messagebox._show("Verification Info!", "Verification Failed")
            self.begin_page()
```

```
160     def final_page(self):
161         self.frame.destroy()
162         self.frame = Frame(self.root,bg="#0019fc",width=900,height=500)
163         self.detail = Button(self.frame,text="Transfer",bg="#50A8B0",fg="white",font=ARIAL,command =
                self.user_account_transfer)
164         self.enquiry = Button(self.frame, text="Balance Enquiry",bg="#50A8B0",fg="white",font=ARIAL,
                command = self.user_balance)
165         self.deposit = Button(self.frame, text="Deposit Money",bg="#50A8B0",fg="white",font=ARIAL,
                command = self.user_deposit_money)
166         self.withdrawl = Button(self.frame, text="Withdrawl Money",bg="#50A8B0",fg="white",font=
                ARIAL,command = self.user_withdrawl_money)
167         self.q = Button(self.frame, text="Log out", bg="#50A8B0", fg="white", font=ARIAL, command=
                self.begin_page)
168         self.detail.place(x=0,y=0,width=200,height=50)
169         self.enquiry.place(x=0, y=315, width=200, height=50)
170         self.deposit.place(x=600, y=0, width=200, height=50)
171         self.withdrawl.place(x=600, y=315, width=200, height=50)
172         self.q.place(x=340, y=340, width=120, height=20)
173         self.frame.pack()
174
175
176     def user_account_transfer(self):
177         self.frame.destroy()
178         self.frame = Frame(self.root,bg="#0019fc",width=900,height=500)
179         self.detail = Button(self.frame,text="Transfer",bg="#50A8B0",fg="white",font=ARIAL,command =
                self.user_account_transfer)
180         self.enquiry = Button(self.frame, text="Balance Enquiry",bg="#50A8B0",fg="white",font=ARIAL,
                command = self.user_balance)
181         self.deposit = Button(self.frame, text="Deposit Money",bg="#50A8B0",fg="white",font=ARIAL,
                command = self.user_deposit_money)
182         self.withdrawl = Button(self.frame, text="Withdrawl Money",bg="#50A8B0",fg="white",font=
                ARIAL,command = self.user_withdrawl_money)
183         self.q = Button(self.frame, text="Log out", bg="#50A8B0", fg="white", font=ARIAL, command=
                self.begin_page)
184         self.detail.place(x=0,y=0,width=200,height=50)
185         self.enquiry.place(x=0, y=315, width=200, height=50)
186         self.deposit.place(x=600, y=0, width=200, height=50)
187         self.withdrawl.place(x=600, y=315, width=200, height=50)
188         self.q.place(x=340, y=340, width=120, height=20)
189         self.frame.pack()
190         self.label11 = Label(self.frame, text="Please enter the reciepient's account number",bg="
                #0019fc",fg="white",font=ARIAL)
191         self.label21 = Label(self.frame, text="Please enter the amount to be transferred",bg="#0019
                fc",fg="white",font=ARIAL)
192         self.button1 = Button(self.frame,text="Transfer",bg="#50A8B0",fg="white",font=ARIAL,command=
                self.user_account_transfer_transc)
193         self.entry11 = Entry(self.frame,bg="honeydew",highlightcolor="#50A8B0",
194            highlightthickness=2,
195             highlightbackground="white")
196         self.entry21 = Entry(self.frame,bg="honeydew",highlightcolor="#50A8B0",
```

```
            highlightthickness=2,
            highlightbackground="white")
        self.label11.place(x=200,y=130,width=300,height=20)
        self.entry11.place(x=200,y=160,width=300,height=20)
        self.label21.place(x=185,y=190,width=300,height=20)
        self.entry21.place(x=200,y=210,width=300,height=20)
        self.button1.place(x=200,y=250,width=180,height=30)


    def user_account_transfer_transc(self):
        data = pd.read_csv('bank_details.csv')
        if int(self.entry11.get()) not in data['account_number'].values:
            messagebox._show("Transfer Info!", "Invalid account number")
        elif int(self.entry11.get()) == self.real_user:
            messagebox._show("Transfer Info!", "Sorry, you cannot make a transfer to yourself")
        elif int(self.entry21.get()) >= data[data.loc[:,'unique_id'] == self.real_user].loc[:,'
            account_balance'].values[0]:
            messagebox._show("Transfer Info!", "Insufficient Funds")
        else:
            data = pd.read_csv('bank_details.csv')
            update_data = data.set_index('account_number')
            update_data.loc[int(self.entry11.get()),'account_balance'] += int(self.entry21.get())
            update_data.loc[data[data.loc[:,'unique_id'] == self.real_user].loc[:,'account_number'].
                values[0],'account_balance'] -= int(self.entry21.get())
            update_data['account_number'] = update_data.index
            update_data.reset_index(drop = True, inplace= True)
            update_data = update_data.reindex(labels = ['unique_id','account_number','name','bank',
                'password','account_balance'], axis = 1)
            update_data.to_csv('bank_details.csv',index = None)
            messagebox._show("Transfer Info!", "Successfully Transferred")

    def user_balance(self):
        self.frame.destroy()
        self.frame = Frame(self.root,bg="#0019fc",width=900,height=500)
        self.detail = Button(self.frame,text="Transfer",bg="#50A8B0",fg="white",font=ARIAL,command =
            self.user_account_transfer)
        self.enquiry = Button(self.frame, text="Balance Enquiry",bg="#50A8B0",fg="white",font=ARIAL,
            command = self.user_balance)
        self.deposit = Button(self.frame, text="Deposit Money",bg="#50A8B0",fg="white",font=ARIAL,
            command = self.user_deposit_money)
        self.withdrawl = Button(self.frame, text="Withdrawl Money",bg="#50A8B0",fg="white",font=
            ARIAL,command = self.user_withdrawl_money)
        self.q = Button(self.frame, text="Log out", bg="#50A8B0", fg="white", font=ARIAL, command=
            self.begin_page)
        self.detail.place(x=0,y=0,width=200,height=50)
        self.enquiry.place(x=0, y=315, width=200, height=50)
        self.deposit.place(x=600, y=0, width=200, height=50)
        self.withdrawl.place(x=600, y=315, width=200, height=50)
        self.q.place(x=340, y=340, width=120, height=20)
        self.frame.pack()
```

```
239         data = pd.read_csv('bank_details.csv')
240         text = data[data.loc[:,'unique_id'] == self.real_user].loc[:,'account_balance'].values[0]
241         self.label = Label(self.frame, text= 'Current Account Balance: ' + 'N' + str(text),font=
                ARIAL)
242         self.label.place(x=200, y=100, width=300, height=100)
243
244     def user_deposit_money(self):
245         self.frame.destroy()
246         self.frame = Frame(self.root,bg="#0019fc",width=900,height=500)
247         self.detail = Button(self.frame,text="Transfer",bg="#50A8B0",fg="white",font=ARIAL,command =
                self.user_account_transfer)
248         self.enquiry = Button(self.frame, text="Balance Enquiry",bg="#50A8B0",fg="white",font=ARIAL,
                command = self.user_balance)
249         self.deposit = Button(self.frame, text="Deposit Money",bg="#50A8B0",fg="white",font=ARIAL,
                command = self.user_deposit_money)
250         self.withdrawl = Button(self.frame, text="Withdrawl Money",bg="#50A8B0",fg="white",font=
                ARIAL,command = self.user_withdrawl_money)
251         self.q = Button(self.frame, text="Log out", bg="#50A8B0", fg="white", font=ARIAL, command=
                self.begin_page)
252         self.detail.place(x=0,y=0,width=200,height=50)
253         self.enquiry.place(x=0, y=315, width=200, height=50)
254         self.deposit.place(x=600, y=0, width=200, height=50)
255         self.withdrawl.place(x=600, y=315, width=200, height=50)
256         self.q.place(x=340, y=340, width=120, height=20)
257         self.frame.pack()
258         self.label = Label(self.frame, text="Enter amount", font=ARIAL)
259         self.label.place(x=200, y=100, width=300, height=100)
260         self.money_box = Entry(self.frame,bg="honeydew",highlightcolor="#50A8B0",
261            highlightthickness=2,
262             highlightbackground="white")
263         self.submitButton = Button(self.frame,text="Deposit",bg="#50A8B0",fg="white",font=ARIAL)
264
265         self.money_box.place(x=200,y=100,width=200,height=20)
266         self.submitButton.place(x=445,y=100,width=55,height=20)
267         self.submitButton.bind("<Button-1>",self.user_deposit_trans)
268
269     def user_deposit_trans(self,flag):
270         data = pd.read_csv('bank_details.csv')
271         data = pd.read_csv('bank_details.csv')
272         update_data = data.set_index('unique_id')
273         update_data.loc[self.real_user,'account_balance'] += int(self.money_box.get())
274         update_data.reset_index(inplace=True)
275         update_data.columns = ['unique_id','account_number','name','bank', 'password','
                account_balance']
276         update_data.to_csv('bank_details.csv',index = None)
277         messagebox._show("Deposit Info!", "Successfully Deposited!")
278
279     def user_withdrawl_money(self):
280         self.label = Label(self.frame, text="Enter amount", font=ARIAL)
281         self.label.place(x=200, y=100, width=300, height=100)
```

```
282         self.money_box = Entry(self.frame,bg="honeydew",highlightcolor="#50A8B0",
283             highlightthickness=2,
284              highlightbackground="white")
285         self.submitButton = Button(self.frame,text="Withdraw",bg="#50A8B0",fg="white",font=ARIAL)
286 
287         self.money_box.place(x=200,y=100,width=200,height=20)
288         self.submitButton.place(x=435,y=100,width=70,height=20)
289         self.submitButton.bind("<Button-1>",self.user_withdrawl_trans)
290 
291     def user_withdrawl_trans(self,flag):
292         data = pd.read_csv('bank_details.csv')
293         update_data = data.set_index('unique_id')
294         if int(self.money_box.get()) <= update_data.loc[self.real_user,'account_balance']:
295             update_data.loc[self.real_user,'account_balance'] -= int(self.money_box.get())
296             update_data.reset_index(inplace=True)
297             update_data.columns = ['unique_id','account_number','name','bank', 'password','
                    account_balance']
298             update_data.to_csv('bank_details.csv',index = None)
299             messagebox._show("Withdrwawal Info!", "Successfully Withdrwan, please take your cash")
300         else:
301             messagebox._show("Withdrwal Info!", "Insufficient Funds")
302 
303 
304 
305 
306 
307 
308     def write_to_csv(self):
309         import csv
310         from random import randint
311         n = 10;range_start = 10**(n-1);range_end = (10**n)-1
312         account_number = randint(range_start, range_end)
313         n = 5;range_start = 10**(n-1);range_end = (10**n)-1
314         unique_id = randint(range_start, range_end)
315         bank = "Major Bank"
316         account_balance = "10000"
317         name = self.uentry.get()
318         password = self.pentry.get()
319         with open(r'bank_details.csv','a', newline = '\n') as f:
320             writer = csv.writer(f)
321             writer.writerow([unique_id,account_number,name,bank, password, account_balance])
322         messagebox._show("Enrollment Info!", "Successfully Enrolled!")
323 
324     def video_capture_page(self):
325         self.frame.destroy()
326         self.frame = Frame(self.root,bg="#0019fc",width=900,height=500)
327         #Login Page Form Components
328         self.label1 =Label(self.frame,text="Note:",bg="#0019fc",fg="white",font=ARIAL)
329         self.label2 =Label(self.frame,text="1.By clicking on the 'Capture' button below, your image
                gets captured ",bg="#0019fc",fg="white",font=ARIAL)
```

```
330         self.label3 =Label(self.frame,text="2.You will be required to capture 5 images for full
                registration",bg="#0019fc",fg="white",font=ARIAL)
331         self.label4 =Label(self.frame,text="3.To capture each image click the space bar on your
                keyboard when the camera turn on:",bg="#0019fc",fg="white",font=ARIAL)
332         self.label5 =Label(self.frame,text="4. Please wait till you are notified that your capture
                was successful before leaving the page",bg="#0019fc",fg="white",font=ARIAL)
333         data = pd.read_csv('bank_details.csv')
334         self.label6 =Label(self.frame,text="5.To begin, click the 'Capture' button below and click
                the space bar to capture a new image",bg="#0019fc",fg="white",font=ARIAL)
335         self.button = Button(self.frame,text="Capture",bg="#50A8B0",fg="white",font=ARIAL,command=
                self.captureuser)
336         #self.q = Button(self.frame,text="Quit",bg="#50A8B0",fg="white",font=ARIAL,command = self.
                root.destroy)
337         #self.b = Button(self.frame,text="Back",bg="#50A8B0",fg="white",font=ARIAL,command = self.
                enroll_user)
338         self.label1.place(x=100,y=100,width=600,height=20)
339         self.label2.place(x=100,y=120,width=600,height=20)
340         self.label3.place(x=100,y=140,width=600,height=20)
341         self.label4.place(x=100,y=160,width=600,height=20)
342         self.label5.place(x=100,y=180,width=600,height=20)
343         self.label6.place(x=100,y=200,width=600,height=20)
344         self.button.place(x=100,y=230,width=600,height=30)
345         #self.q.place(x=480,y=360,width=120,height=20)
346         #self.b.place(x=280,y=360,width=120,height=20)
347         self.frame.pack()
348
349     #hit space bar to capture
350     def captureuser(self):
351         data = pd.read_csv('bank_details.csv')
352         name = data.loc[:,'unique_id'].values[-1]
353         cam = cv2.VideoCapture(0)
354
355         cv2.namedWindow("capture")
356
357         img_counter = 0
358
359         dirname = f'dataset/{name}'
360         os.mkdir(dirname)
361
362         while True:
363             ret, frame = cam.read()
364             cv2.imshow("capture", frame)
365
366             if img_counter == 5:
367                 cv2.destroyWindow("capture")
368                 break
369             if not ret:
370                 break
371             k = cv2.waitKey(1)
372
```

```
            if k%256 == 27:
                # ESC pressed
                print("Escape hit, closing...")
                break
            elif k%256 == 32:
                path = f'dataset/{name}'
                img_name = "{}.jpg".format(img_counter)
                cv2.imwrite(os.path.join(path , img_name), frame)
                cv2.imwrite(img_name, frame)
                print("{} written!".format(img_name))
                img_counter += 1

        cam.release()

        cv2.destroyAllWindows()

        self.get_embeddings()
        #self.get_embeddings()
        self.train_model()
        messagebox._show('Registration Info!", "Face Id Successfully Registered!')
        self.begin_page()



    def get_embeddings(self):
        ap = argparse.ArgumentParser()
        ap.add_argument("-i", "--dataset", required=True,
            help="path to input directory of faces + images")
        ap.add_argument("-e", "--embeddings", required=True,
            help="path to output serialized db of facial embeddings")
        ap.add_argument("-d", "--detector", required=True,
            help="path to OpenCV's deep learning face detector")
        ap.add_argument("-m", "--embedding-model", required=True,
            help="path to OpenCV's deep learning face embedding model")
        ap.add_argument("-c", "--confidence", type=float, default=0.5,
            help="minimum probability to filter weak detections")
        print("[INFO] loading face detector...")

        detector = cv2.dnn.readNetFromCaffe('face_detection_model/deploy.prototxt', '
            face_detection_model/res10_300x300_ssd_iter_140000.caffemodel')
        embedder = cv2.dnn.readNetFromTorch('nn4.small2.v1.t7')

        print("[INFO] quantifying faces...")
        imagePaths = list(paths.list_images('dataset'))
        knownEmbeddings = []
        knownNames = []
        total = 0
        for (i, imagePath) in enumerate(imagePaths):
            print("[INFO] processing image {}/{}".format(i + 1,
                len(imagePaths)))
```

```
            name = imagePath.split(os.path.sep)[-2]

            image = cv2.imread(imagePath)
            image = imutils.resize(image, width=600)
            (h, w) = image.shape[:2]
            imageBlob = cv2.dnn.blobFromImage(
                cv2.resize(image, (300, 300)), 1.0, (300, 300),
                (104.0, 177.0, 123.0), swapRB=False, crop=False)

            detector.setInput(imageBlob)
            detections = detector.forward()

            if len(detections) > 0:
                i = np.argmax(detections[0, 0, :, 2])
                confidence = detections[0, 0, i, 2]

                if confidence > 0.5:
                    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
                    (startX, startY, endX, endY) = box.astype("int")

                    face = image[startY:endY, startX:endX]
                    (fH, fW) = face.shape[:2]

                    if fW < 20 or fH < 20:
                        continue

                    faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255,
                        (96, 96), (0, 0, 0), swapRB=True, crop=False)
                    embedder.setInput(faceBlob)
                    vec = embedder.forward()

                    knownNames.append(name)
                    knownEmbeddings.append(vec.flatten())
                    total += 1
        print("[INFO] serializing {} encodings...".format(total))
        data = {"embeddings": knownEmbeddings, "names": knownNames}
        f = open('output/embeddings.pickle', "wb")
        f.write(pickle.dumps(data))
        f.close()



    def train_model(self):
        print("[INFO] loading face embeddings...")
        data = pickle.loads(open('output/embeddings.pickle', "rb").read())
        le = LabelEncoder()
        labels = le.fit_transform(data["names"])
        print("[INFO] training model...")
        recognizer = SVC(C=1.0, kernel="linear", probability=True)
        recognizer.fit(data["embeddings"], labels)
```

```
472         f = open('output/recognizer.pickle', "wb")
473         f.write(pickle.dumps(recognizer))
474         f.close()
475 
476         f = open('output/le.pickle', "wb")
477         f.write(pickle.dumps(le))
478         f.close()
479 
480 
481 
482 
483     def video_check(self):
484 
485         detector = cv2.dnn.readNetFromCaffe('face_detection_model/deploy.prototxt', '
                face_detection_model/res10_300x300_ssd_iter_140000.caffemodel')
486         print("[INFO] loading face recognizer...")
487         embedder = cv2.dnn.readNetFromTorch('nn4.small2.v1.t7')
488 
489         recognizer = pickle.loads(open('output/recognizer.pickle', "rb").read())
490         le = pickle.loads(open('output/le.pickle', "rb").read())
491 
492         print("[INFO] starting video stream...")
493         vs = VideoStream(src=0).start()
494         time.sleep(2.0)
495 
496         timeout = time.time() + 5
497 
498         fps = FPS().start()
499 
500         real_user_list = []
501         while True:
502 
503             if time.time() > timeout :
504                 cv2.destroyWindow("Frame")
505                 break;
506 
507             frame = vs.read()
508 
509             frame = imutils.resize(frame, width=800, height=200)
510             (h, w) = frame.shape[:2]
511 
512             imageBlob = cv2.dnn.blobFromImage(
513                 cv2.resize(frame, (300, 300)), 1.0, (300, 300),
514                 (104.0, 177.0, 123.0), swapRB=False, crop=False)
515 
516             detector.setInput(imageBlob)
517             detections = detector.forward()
518 
519             for i in range(0, detections.shape[2]):
520                 confidence = detections[0, 0, i, 2]
```

```
521
522                 if confidence > 0.5:
523                     box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
524                     (startX, startY, endX, endY) = box.astype("int")
525
526                     face = frame[startY:endY, startX:endX]
527                     (fH, fW) = face.shape[:2]
528
529                     if fW < 20 or fH < 20:
530                         continue
531
532                     faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255,
533                         (96, 96), (0, 0, 0), swapRB=True, crop=False)
534                     embedder.setInput(faceBlob)
535                     vec = embedder.forward()
536
537                     preds = recognizer.predict_proba(vec)[0]
538                     j = np.argmax(preds)
539                     proba = preds[j]
540                     name = le.classes_[j]
541
542                     if (name == 'unknown') or (proba *100) < 50:
543                         print("Fraud detected")
544                         real_user_list.append(name)
545                     else:
546                         #cv2.destroyWindow("Frame")
547                         real_user_list.append(name)
548                         break;
549
550
551             fps.update()
552
553             cv2.imshow("Frame", frame)
554             key = cv2.waitKey(1) & 0xFF
555
556             if key == ord("q"):
557                 break
558
559         fps.stop()
560         print("[INFO] elasped time: {:.2f}".format(fps.elapsed()))
561         print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
562
563
564         cv2.destroyAllWindows()
565         vs.stop()
566         print(real_user_list)
567
568         try:
569             Counter(real_user_list).most_common(1)[0][0] == 'unknown'
570         except IndexError:
```

```
            if self.countter != 0:
                messagebox._show("Verification Info!", "Face Id match failed! You have {} trials left".format(self.countter))
                self.countter = self.countter - 1
                self.video_check()
            else:
                messagebox._show("Verification Info!", "Face Id match failed! You cannot withdraw at this time, try again later")
                self.begin_page()
                self.countter = 2


        else:
            if Counter(real_user_list).most_common(1)[0][0] == 'unknown':
                if self.countter != 0:
                    messagebox._show("Verification Info!", "Face Id match failed! You have {} trials left".format(self.countter))
                    self.countter = self.countter - 1
                    self.video_check()
                else:
                    messagebox._show("Verification Info!", "Face Id match failed! You cannot withdraw at this time, try again later")
                    self.begin_page()
                    self.countter = 2

            else:
                self.real_user = int(Counter(real_user_list).most_common(1)[0][0])
                messagebox._show("Verification Info!", "Face Id match!")
                self.password_verification()


root = Tk()
root.title("Major Bank")
root.geometry("800x500")
root.configure(bg="blue")
obj = BankUi(root)
root.mainloop()
```

10.2 Poster Presentation

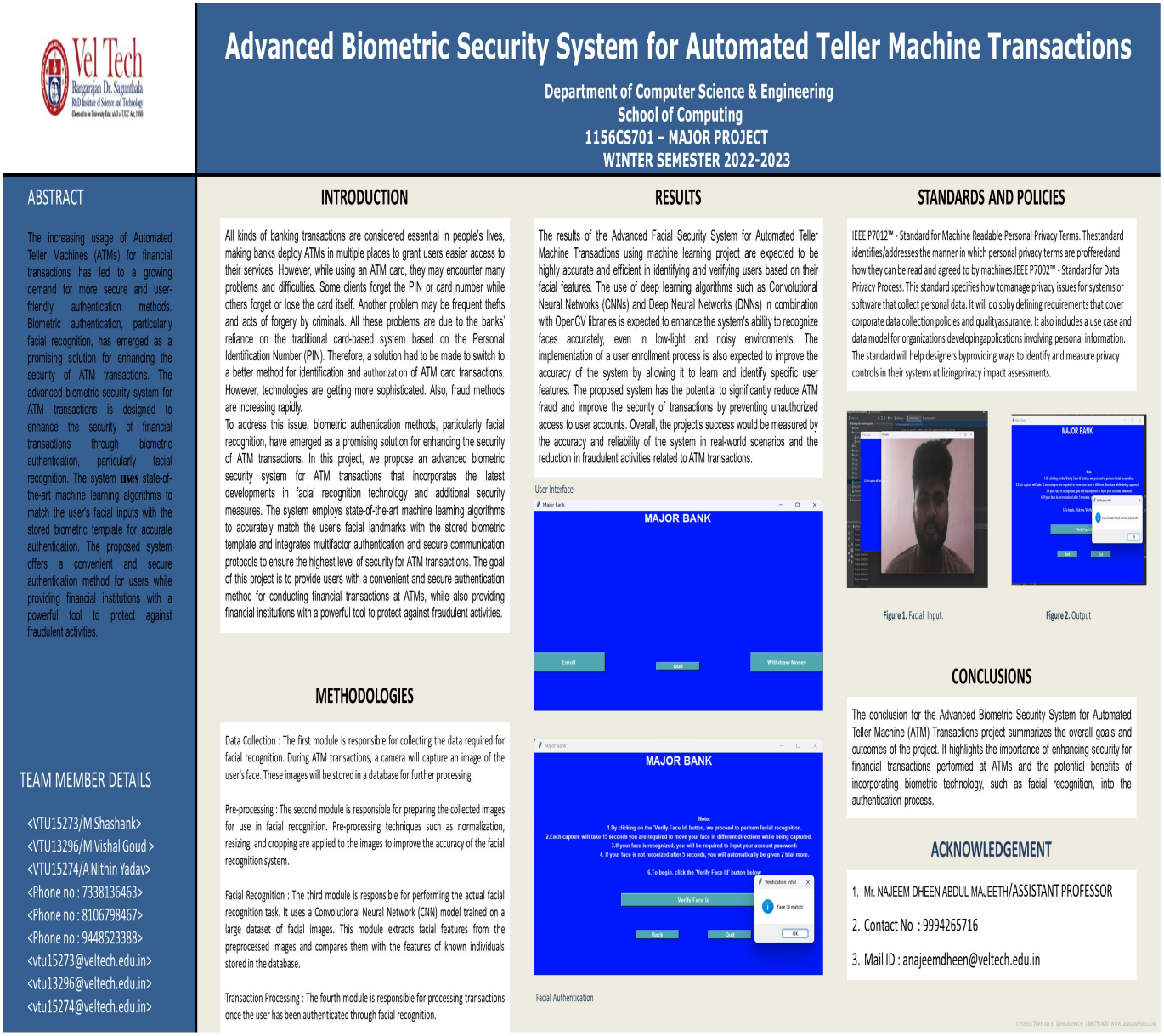


Figure 10.1: **Poster Presentation**

References

[1] Priyanka Mahajan, May 2016, “New Approach in Biometrics to Combat the ATM Frauds: Facial Recognition”, International Journal Of Engineering And Computer Science, ISSN: 2319-7242, Volume 5, Issue 5.

[2] Nabihah Ahmad, A. Aminurdin M. Rifen, Mohd Helmy Abd Wahab, 2016, “ATM Biometric Security System Design Using FPGA Implementation”, International Engineering Research and Innovation Symposium, Vol.160.

[3] Manish C M,N Chirag,Praveen H R,Darshan M J,D Khasim Vali, Jan 2020, ”Cardless ATM Transaction Using Biometric And Face Recognition”, International Journal of Scientific & Engineering Research, ISSN 2229-5518, Vol 11, Issue 1.

[4] Murugesan M, Santhosh M, Sasi Kumar T, Sasiwarman M, Valanarasu I, Mar - Apr 2020, “Securing ATM Transactions using Face Recognition”, International Journal of Advanced Trends in Computer Science and Engineering, ISSN 2278-3091, Volume 9, Issue 2.

[5] Dr S Sasipriya, Dr P. Mayil Vel Kumar, S. Shenbagadevi, 2020, “FACE RECOGNITION BASED NEW GENERATION ATM SYSTEM”, European Journal of Molecular Clinical Medicine, ISSN 2515-8260, Vol 7, Issue 4.

[6] Ms Soundari D V, Aravind R,Edwin Raj K, Abishek S, 2021, “Enhanced Security Feature of ATM’S Through Facial Recognition”, International Conference on Intelligent Computing and Control Systems, ISBN: 978-0-7381-1327-2 .

[7] Shumukh M Alijuaid, Arshiya S Ansari, 2021, “Automated Teller Machine Authentication Using Biometric”, Computer Systems Science Engineering, Vol 41, Issue 3.

[8] Praveena. P, Savithra. V, Saratha. R, Monisha. M, Ashwini. R, Aug 2021, “Face Detection Open CV Based ATM Security System ”, International Journal for Modern Trends in Science and Technology, ISSN: 2455-3778, Vol 7, Issue 8.

[9] Prof. Anil. D. Gujar, Nikita B Sawant, Tejas L Hake, Aadesh A Shete, Shreekar M Deshmukh, May 2022, “Face Recognition Open CV Based ATM Security System”, International Journal for Research in Applied Science Engineering Technology, ISSN: 2321-9653, Vol 10, Issue 5.