Secure ATM Banking System Using Facial Recognition Techniques Based on Deep Learning

ABSTRACT

Automated Teller Machine (ATM) is now used for transactions by different users in day to day life due to its high convenience. With the help of ATM's, banking is now easier. But there have been a lot of frauds that are occurring in ATM's. Thus there is a need to provide more security to these ATM's. The proposed system provides the real-time detection and real-world encounter through Haar Cascade & CNN, an analytical service. The software starts by taking pictures of everyone and keeps the details in a database. The proposed activity works with the default detection system. The method consists of three stages, the first is facial detection from the image, and the second is obtaining all the facial details for the purpose of expression recognition to detect liveness. The most useful features that differ from the camera image are extracted from the feature extraction section finding out if all the face details are visible. This feature vector creates an active face representation. In the third step our feature background is designed to find out what an osculated face looks like.

Keywords: CNN; Expression recognition; Face detection; Haar cascade; liveness detection.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF OUTPUTS

CHAPTER 1

1.INTRODUCTION

The advent of technology brings a wide range of tools that desire the greatest happiness of customers. An ATM is a machine that makes money transactions less efficient for customers. But it has both advantages and disadvantages. Current ATMs use no more than an access card and PIN for different authentication. This is at an ATM Using the Face Recognition System which shows the way to many fraudulent attempts and misconduct for card theft, PIN theft, theft and hacking of customer account information and other security features. Exploring the camera module based on visual performance comparisons depends on the similarities between the features extracted from the image regions and those from the image in question. Face recognition system is a system that automatically identifies a person from a digital photo source. One of the best ways to do this is by comparing selected facial features from a face and photo database. Providing a secure ATM System for face recognition and bringing brightness, computer vision lessons in human face recognition [1]. Face recognition program is a computer program that automatically detects or verifies a person from a digital photo. As there is a lot of deception and misuse of ATM Card so build a secure face-biometric authentication system, which will be available via live server and will be protected. The proposed system will assist in many ways in the banking sector. We can do tracking tasks: •Finding a valid Account holder using the CNN face algorithm. •Assessing user performance with the help of expression recognition. •Creating an automated, easy-to-use and complete automated system with high accuracy. •Reduce the amount of fraud in the ATM system by providing more security features such as real-time face recognition. •Identify all instances of objects from a known category, such as people or faces in a photo. The face detector can calculate the areas of the eyes, nose and mouth, in addition to the binding box of the face.

1.1 Purpose and Objectives of the Project

Purpose for Card less ATM Using Deep Learning Project:

The primary purpose of the Cardless ATM Using Deep Learning project is to enhance the security and convenience of ATM transactions by eliminating the need for physical cards and replacing them with facial recognition-based authentication. This approach aims to:

1. Increase Security: Prevent common ATM-related frauds such as card skimming, PIN theft, and unauthorized access by leveraging deep learning technologies for user authentication. The system ensures that only authorized users with a valid facial profile can perform transactions.

2. Enhance User Convenience: Provide a cardless and contactless ATM experience, allowing users to perform banking transactions without the need for physical ATM cards. This simplifies the process for users and reduces the risk of losing or misplacing cards.

3. Prevent Spoofing Attacks: Ensure the legitimacy of the user through liveness detection, which verifies that the face presented is from a live person rather than a photo or video, further enhancing the security of the system.

4. Improve Transaction Efficiency: Streamline the transaction process by integrating facial recognition with the traditional PIN or OTP verification methods. This ensures a multi-factor authentication system that is both fast and secure.

5. Reduce Operational Costs: Eliminate the need for card issuance and management by financial institutions, reducing both operational costs and environmental waste associated with plastic card production.

In summary, the purpose of this project is to create a more secure, efficient, and user-friendly ATM experience through the application of deep learning technologies, ultimately reducing fraud and enhancing the security of financial transactions.

Objectives of the Cardless ATM Using Deep Learning project:

1. Develop a cardless authentication system: Implement a secure and reliable system that replaces traditional ATM cards with facial recognition as the primary method for user authentication.

2. Utilize deep learning for face detection and recognition: Leverage Convolutional Neural Networks (CNNs) and Haar Cascade algorithms to accurately detect and recognize users' faces in real-time, ensuring a robust and scalable authentication process.

3. Integrate liveness detection: Implement a liveness detection mechanism to ensure that the presented face is from a live person, preventing fraud attempts using photos, videos, or 3D masks.

4. Enhance transaction security: Strengthen the overall security of ATM transactions by integrating biometric facial recognition with multi-factor authentication methods such as PINs or OTPs to prevent unauthorized access.

5. Ensure user privacy and data protection: Safeguard user biometric data by employing encryption and secure storage methods, ensuring that personal information is protected in compliance with data protection regulations.

6. Improve user experience: Provide a seamless, contactless ATM experience for users by enabling quick and easy access to banking services without the need for physical cards, thereby reducing wait times and enhancing convenience.

7. Reduce fraud and skimming incidents: Minimize the occurrence of fraud such as card skimming, PIN theft, and card cloning, which are common issues in traditional ATM systems.

8. Support scalability and adaptability: Design the system to be easily scalable and adaptable for future upgrades, such as integrating additional biometric authentication methods (e.g., fingerprint or iris recognition).

By achieving these objectives, the project aims to revolutionize the way users interact with ATMs, making transactions safer, more efficient, and user-friendly while minimizing fraud risks.

1.2 EXISTING & PROPOSED SYSTEM

Existing System

In the traditional ATM system, users are required to insert their debit or credit card into the ATM's card reader and then enter their personal identification number (PIN) using the keypad. The ATM processes these inputs and sends the card information along with the entered PIN to the bank's server for verification. Once the system confirms that the user's credentials are correct, it grants

access to a range of banking services such as cash withdrawals, balance inquiries, fund transfers, and other banking operations.

The core components of the traditional ATM include a card reader, keypad, display screen, cash dispenser, receipt printer, and network connection to the bank's servers. Each of these components plays a crucial role in the transaction process. The card reader captures the card's magnetic stripe or chip data, while the keypad allows users to input their PIN and transaction choices. Once the authentication is successful, the cash dispenser dispenses the requested amount of money, and the receipt printer provides a transaction summary.

One of the key limitations of the existing system is its reliance on physical cards and PINs for user identification and authentication. This system makes ATMs vulnerable to a variety of frauds, including card skimming (the theft of card details using external devices), PIN theft (via hidden cameras or shoulder surfing), and card cloning. Attackers can duplicate a user's card and misuse it to withdraw money. Furthermore, if a user's card is lost or stolen, it can result in unauthorized transactions if the PIN is compromised.

Moreover, the reliance on physical interaction (inserting a card, typing a PIN) is less convenient and can be prone to wear and tear over time, leading to malfunctioning ATMs. In recent years, there has been a growing need for a more secure and user-friendly alternative that reduces fraud risks, improves convenience, and enhances user experience. This is where modern, cardless solutions, such as those using facial recognition powered by deep learning, come into play.

Disadvantage:

Fraud Vulnerability: Physical cards and PINs are prone to card skimming, PIN theft, and card cloning.

Inconvenience: Reliance on physical cards and PINs can lead to wear and tear over time, causing malfunctioning ATMs or card loss.

Proposed System

The proposed cardless ATM system replaces traditional physical cards with facial recognition for biometric authentication. When a user approaches the ATM, the system captures their facial image using a camera and employs deep learning algorithms to detect and analyze the user's unique facial features. Additionally, liveness detection ensures that the face belongs to a live person and not a static image or video. The extracted features are then compared with the stored biometric data in the bank's database. Once the user's identity is confirmed, they can proceed with transactions like cash withdrawals or balance inquiries, with the system communicating with the bank for authorization.

This approach significantly enhances both security and convenience. By eliminating physical cards and PINs, the system mitigates risks associated with card skimming, PIN theft, and card cloning. Users no longer need to worry about losing their cards or exposing sensitive information, and since the system is contactless, it also addresses hygiene concerns. Moreover, this cardless solution reduces operational costs for banks by eliminating the need for card management, while providing a faster, more secure, and user-friendly banking experience.

Advantages:

Enhanced Security: Eliminates risks of card skimming, PIN theft, and card cloning.

Convenience: No physical cards or PINs required; system is fully contactless.

Cost Efficiency: Reduces banks' operational costs by eliminating the need for physical card management.

Hygiene: Contactless system addresses concerns of physical touchpoints, making it more sanitary.

1.3 SCOPE OF THE PROJECT

Biometric Authentication: Implement facial recognition technology to authenticate users, replacing traditional card-based methods. This includes capturing, processing, and verifying facial images using deep learning algorithms.

Liveness Detection: Integrate optional liveness detection to ensure that the facial recognition system distinguishes between live individuals and static images or videos, enhancing security.

User Interface: Develop a user-friendly interface on the ATM that allows users to initiate transactions, view results, and receive feedback without needing a physical card.

Transaction Management: Enable a range of standard ATM transactions, such as cash withdrawals, balance inquiries, and account transfers, once the user is authenticated.

Data Security: Ensure the secure storage and handling of biometric data and transaction information. Implement robust encryption and protection mechanisms to safeguard sensitive information.

Integration with Banking Systems: Interface with existing bank systems for user authentication and transaction processing, ensuring seamless communication and data exchange.

System Testing and Validation: Conduct thorough testing of the facial recognition system and overall ATM functionality to ensure accuracy, reliability, and user satisfaction.

Regulatory Compliance: Adhere to relevant regulations and standards for biometric data usage, privacy, and security to ensure legal and ethical compliance.

CHAPTER 2

2.LITERATURE SURVEY

"Facial Recognition for Banking: Security and Innovation"

Background

As banking services become increasingly digitized, there is a growing interest in adopting advanced security technologies to prevent fraud. One such technology is facial recognition, which is rapidly gaining traction in financial systems. Facial recognition systems leverage deep learning algorithms to analyze facial features and create unique biometric identifiers for each individual. In the context of cardless ATM transactions, this technology enables a secure, seamless user experience by replacing traditional authentication methods such as ATM cards or PINs.

Research Highlights

The rise of biometric authentication in banking, particularly facial recognition, offers potential for both security enhancement and user convenience. Studies such as Parkhi et al. (2015) highlight how facial recognition systems are now capable of achieving near-human accuracy.

A study by Taigman et al. (2014) demonstrated the scalability of deep learning-based facial recognition systems and their ability to handle large volumes of data. This is crucial for ATM systems, which must authenticate users swiftly in real-time.

Recent advancements in AI and machine learning, particularly convolutional neural networks (CNNs), have dramatically improved the accuracy and reliability of facial recognition systems. Schroff et al. (2015) discussed the introduction of FaceNet, which increased the ability to cluster and verify faces with high precision, making it suitable for high-security environments like ATMs.

"Cardless Transactions and Biometric Security: A New Frontier for Banking"

Background

Cardless ATM systems are a relatively new innovation aimed at reducing physical dependency on debit or credit cards. These systems can leverage biometric data, such as facial recognition, to authenticate users and permit cash withdrawals. With global trends toward digital banking, financial institutions are focusing on cardless ATM technology as an alternative to traditional methods that are vulnerable to card skimming, theft, or duplication.

Research Highlights

Jain et al. (2006) provided insights into the use of biometric data for secure transactions, emphasizing that facial recognition offers a superior layer of protection against fraud.

A whitepaper by Zhang and Ahmad (2019) discussed how cardless ATM technology could revolutionize banking by minimizing physical touchpoints and leveraging AI-driven security protocols like facial recognition.

U.S. patents, such as Patent No. 9,602,487 (2017), have set the foundation for combining facial recognition with ATMs, showcasing methods of authenticating users based on their unique biometric identifiers without requiring traditional forms of identification.

Key Research Works and Findings

✅ **P. Singh et al. (2021)**

Title: Secure ATM System Using Face Recognition

- Proposed CNN-based model using OpenCV and Dlib.
- Achieved 92% accuracy in controlled lighting.
- Emphasized the importance of anti-spoofing techniques.

✅ **S. Kumar & A. Reddy (2020)**

Title: Cardless Transactions in ATMs using Facial Authentication

- Introduced OTP verification in addition to face recognition.
- Achieved improved security and reduced false acceptance.

✅ **A. Sharma et al. (2019)**

Title: Facial Recognition in Banking Transactions

- Compared PCA vs Deep Learning methods.
- Deep learning models outperformed PCA in accuracy and reliability.

CHAPTER 3

3.SYSTEM ANALYSIS

3.1 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements

CPU: Intel i7 or i9, AMD Ryzen 7 or 9

RAM: 16GB Minimum - 32GB Maximum

Storage: SSD with at least 256GB or higher

Camera: Resolution at least 720p HD (1080p HD recommended), 30 fps minimum, HDR preferred, USB or high-resolution IP camera

Software Requirements

Operating Systems: Windows 10/11, macOS, Linux

Programming Language: Python 3.7 or higher

Libraries/Frameworks: TensorFlow/Keras, OpenCV, Numpy, Pandas, Matplotlib/Seaborn, scikit-learn, Haarcascades

IDE: Jupyter Notebook or JupyterLab, PyCharm, VS Code

Back-end : Mysql

3.2 SOFTWARE REQUIREMENT SPECIFICATION

1. Introduction

1.1 Purpose The purpose of this Software Requirement Specification (SRS) document is to outline the software requirements for the development of a cardless ATM system using deep learning technology. This system aims to replace traditional card-based authentication with biometric facial recognition to enhance security and user convenience.

1.2 Scope The cardless ATM system will include functionalities for facial image capture, face detection, feature extraction, user authentication, and transaction processing. The system will integrate with existing banking infrastructure to handle transaction requests securely and efficiently.

1.3 Definations:

- Cardless ATM: An ATM system that uses facial recognition for user authentication instead of physical cards.
- Facial Recognition: A biometric technology that identifies or verifies a person's identity using their facial features.
- Deep Learning: A subset of machine learning that uses neural networks with multiple layers to process and analyze data.

2. Functional Requirements

2.1 User Authentication

- Image Capture: The system must capture a high-resolution image of the user's face.
- Face Detection: The system must detect and isolate the face from the captured image.
- Feature Extraction: The system must extract unique facial features using a Convolutional Neural Network (CNN).
- Liveness Detection: The system should optionally verify that the face is from a live person to prevent spoofing.
- Authentication: The system must compare extracted features with stored data to authenticate the user.

2.2 Transaction Processing

- Transaction Initiation: Users should be able to initiate transactions (e.g., cash withdrawal, balance inquiry) after successful authentication.
- Transaction Authorization: The system must communicate with the bank to authorize and process transactions.
- Receipt Printing: The system should provide an option to print a transaction receipt if requested.

2.3 Data Security

- Biometric Data Storage: The system must securely store biometric data using encryption.
- Transaction Data Protection: The system must protect transaction data and ensure secure communication with banking servers.

3. Non-Functional Requirements

3.1 Performance

- Response Time: The system should capture and process facial images within 5 seconds.
- Accuracy: Facial recognition should have a minimum accuracy rate of 95% for authentication.

- User Interface: The interface must be intuitive and easy to navigate, with clear instructions for users.
- Accessibility: The system should be accessible to users with disabilities, including visual and hearing impairments.

3.2 Reliability

- System Availability: The system must be operational 24/7 with minimal downtime.
- Error Handling: The system should provide informative error messages and recovery options for failed authentication or transaction issues.

3.3 Security

- Data Encryption: All sensitive data, including biometric and transaction data, must be encrypted both in transit and at rest.
- Compliance: The system must comply with relevant regulations and standards for biometric data usage and privacy.

4. System Architecture

4.1 Components

- User Interface: Touchscreen or display for user interaction.
- Image Capture Module: Camera for capturing facial images.
- Deep Learning Module: For face detection, feature extraction, and liveness detection.
- Database: For storing user biometric and transaction data.
- Communication Module: For interfacing with the bank's authentication and transaction services.

4.2 Integration

- Bank System: Integration with existing bank systems for authentication and transaction processing.

CHAPTER 4

4.METHODOLOGY

The methodology of a cardless ATM banking system using facial recognition begins with the registration and enrollment phase, where the user's biometric data is first captured and stored securely. During this stage, customers visit their bank branch or use a secure mobile application to register their facial data. Advanced facial recognition algorithms, such as convolutional neural networks (CNNs), are used to extract unique facial features—like the distance between the eyes, nose shape, and jawline contour. This biometric data is then encrypted and stored in the bank's secure database, often accompanied by secondary authentication data such as a phone number, email ID, or a personal identification number (PIN) for multi-factor authentication.

In the authentication phase, when a customer visits an ATM, they can select the cardless transaction option. A high-resolution camera integrated with the ATM captures the live image of the user. This image is pre-processed to normalize lighting and angles, and then fed into a facial recognition system that compares it against the enrolled facial template stored in the bank's database. If the system finds a match with a high confidence score, the user is authenticated. The facial recognition process uses AI and machine learning models to ensure accuracy, even if the user's appearance has changed slightly due to age, hairstyle, or accessories like glasses.

To enhance security and reduce fraud, the system may include liveness detection and anti-spoofing mechanisms. These technologies detect whether the face being scanned is of a real, live person rather than a photo or video replay. Techniques such as blinking detection, head movement tracking, and infrared scanning are implemented to confirm the liveness of the subject.

Additionally, the system may cross-check the live location of the user or send an OTP (One-Time Password) to their registered mobile number for extra verification. This layered approach ensures that the ATM transaction is being initiated by the rightful account holder, minimizing the risk of identity theft or unauthorized access.

Finally, upon successful authentication, the user can proceed with regular ATM functions such as withdrawing cash, checking balances, or transferring money—all without using a physical card. The entire transaction is recorded and logged for audit purposes. The methodology not only enhances user convenience by eliminating the need to carry a debit or credit card, but also significantly improves security by relying on biometric authentication, which is far more difficult to duplicate than traditional card-and-PIN systems. Overall, this modern approach integrates facial recognition with banking infrastructure to offer a seamless and secure cardless ATM experience.

1. Facial Recognition Technology

Facial recognition is the core biometric technology used to identify and authenticate users based on their facial features. It involves capturing an image of the user's face and using algorithms (like CNNs or Eigenfaces) to extract key facial landmarks—such as the eyes, nose, and jawline. These features are converted into a digital template and compared against stored templates in the database. Modern facial recognition systems are powered by artificial intelligence and deep learning, allowing them to handle variations in lighting, angles, facial expressions, and even aging.

2. Machine Learning and Deep Learning Algorithms

Machine learning (ML) and deep learning (DL) algorithms play a major role in training facial recognition models. Convolutional Neural Networks (CNNs) are commonly used to learn and recognize complex patterns in facial data. These algorithms improve over time as they are exposed to more data, resulting in higher accuracy and faster processing. Deep learning models are also used

for liveness detection and spoofing prevention, helping the system differentiate between real users and fake representations (like photos or videos).

3.Liveness Detection and Anti-Spoofing Technology

To ensure security, the system integrates liveness detection to verify that the face presented is real and not a printed photo or digital screen. This is achieved using techniques like blink detection, head movement tracking, 3D facial mapping, and texture analysis. Some advanced systems also use infrared (IR) cameras or depth sensors to capture the depth of the user's face. These features help prevent fraudulent attempts to fool the system and ensure only live individuals can perform transactions.

4.Secure Cloud and Database Systems

All facial templates and user data are stored in secure, encrypted databases—often hosted on cloud platforms. These systems ensure scalability, real-time access, and protection against data breaches. Encryption techniques like AES (Advanced Encryption Standard) and secure transmission protocols like HTTPS are used to protect user data during storage and communication. Some systems may use blockchain for tamper-proof logging and identity verification as an added layer of security and transparency.

Python:

Python is a high-level, interpreted programming language known for its readability, simplicity, and vast ecosystem of libraries. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Python is widely used in fields like:

- Web Development
- Data Science
- Artificial Intelligence (AI)
- Machine Learning (ML)
- Computer Vision
- Cybersecurity

- Automation

Its versatility makes it a perfect fit for developing real-time, intelligent systems like cardless ATM banking solutions using facial recognition.

Why Use Python in a Cardless ATM System?

Python is a powerful tool for building cardless ATM systems due to:

- Robust libraries for facial recognition and machine learning
- Rapid development capabilities
- Strong support for real-time image processing
- Easy integration with databases and external APIs

What is HTML?

HTML (HyperText Markup Language) is the standard language used to create and structure content on the web. It defines the layout and structure of web pages using elements like headings, forms, images, videos, buttons, and more.

What is CSS?

CSS (Cascading Style Sheets) is used alongside HTML to style and design web pages. It controls the appearance of elements—such as colors, fonts, layouts, responsiveness, spacing, and animations.
Together, HTML and CSS allow developers to build user interfaces (UI) for web-based applications.

In a cardless ATM system with facial recognition, HTML and CSS play a key role in designing the user-facing interface, especially if the system includes:

- Web-based login terminals
- Mobile or desktop dashboards
- Kiosk-style ATM touchscreens running a browser-based UI

1. User Interface for ATM Interaction

- HTML is used to create UI components like:
  - Face capture buttons
  - Status messages (e.g., “Face recognized”, “Authentication failed”)
  - OTP input fields (if two-factor authentication is used)
  - Transaction selection (Withdraw, Balance Inquiry, etc.)
- CSS styles the interface to:
  - Make it responsive and touchscreen-friendly
  - Provide visual feedback (e.g., green/red borders for success/failure)
  - Ensure a professional and intuitive design

2. Integration with Python Backend

HTML & CSS are typically used with a Python backend (like Flask or Django). For example:

- HTML captures the user’s face using a browser camera.
- A JavaScript module captures the image and sends it to the Python backend for face recognition.
- After processing, the backend sends a result (e.g., authenticated or denied) that updates the UI dynamically.

3. Responsive Design for Mobile/Web ATM UI

CSS media queries and flexible layouts allow the system to adapt to:

- ATM touchscreen resolutions
- Mobile-based cardless ATM apps
- Desktop-based admin interfaces

4.1 MODULES DESCRIPTION

Modules Description:

1. User Interface (UI):

The ATM features an intuitive and user-friendly touchscreen display that allows users to interact with the system. Through the UI, users can select the type of transaction (cash withdrawal, balance inquiry, etc.) and receive real-time feedback about the status of their authentication and transaction. The interface is designed to ensure ease of use and accessibility, providing clear prompts and error

messages as needed.

2. Image Capture Module:

This module consists of a high-resolution camera mounted on the ATM, responsible for capturing the user's facial image. The camera ensures sufficient quality to facilitate accurate facial recognition even in varied lighting conditions. The captured image is sent to the deep learning modules for further processing.

3. Deep Learning Modules:

Face Detection: This module uses a deep learning-based algorithm to detect and isolate the user's face from the captured image, filtering out unnecessary background details. The system adjusts for head orientation, facial expressions, and other variations to ensure precise detection.

Feature Extraction: Once the face is detected, this module applies a convolutional neural network (CNN) to extract unique facial features. The features, such as eye distance, nose structure, and other biometric markers, are converted into a high-dimensional feature vector representing the user's face.

Liveness Detection (Optional): To prevent fraud, this module checks whether the detected face belongs to a live person and is not a static image, video, or mask. Techniques such as eye blinking detection or motion-based analysis are used to ensure the user is physically present at the ATM.

Authentication: The extracted features are compared to the pre-stored facial data of registered users stored in the bank's database. Euclidean distance or other similarity metrics are used to verify the match. If a match is found, the user is authenticated and granted access to the ATM's transaction system.

4. Transaction Management Module:

After successful authentication, the system activates the transaction management module, which processes user requests such as cash withdrawal, balance inquiry, or fund transfer. This module interfaces with the banking system's backend to execute the transaction and updates the user's account information accordingly. A confirmation is displayed on the UI once the transaction is complete.

5. Data Layer:

The data layer securely stores user profiles, including biometric data and past transaction records. It uses encryption protocols to protect sensitive information (e.g., facial feature vectors, transaction logs) and complies with data privacy regulations. The system ensures that only authorized personnel can access or modify the stored data. A database management system (such as PostgreSQL or MongoDB) is employed to manage the data.

6. Communication Module:

This module handles communication between the ATM and the bank's central server. It securely transmits transaction requests, facial data for comparison, and any other necessary information using HTTPS or other secure communication protocols. The module ensures real-time transaction processing and quick response times, facilitating smooth interaction between the ATM and banking systems for transaction authorization and account updates.

CHAPTER 5

5.SYSTEM DESIGN

The cardless ATM system design integrates facial recognition technology to replace traditional card-based methods. It consists of several key components:

1. User Interface: The ATM features a touchscreen or display for users to interact with, select transactions, and receive feedback.

2. Image Capture Module: A camera captures high-resolution images of the user's face.

3. Deep Learning Modules:

Face Detection identifies and isolates the user's face from the image.

Feature Extraction uses deep learning algorithms to analyze and extract unique facial features.

Liveness Detection (optional) checks if the face is from a live person to prevent spoofing.

Authentication compares the extracted features with stored data to verify the user's identity.

4. Transaction Management: Once authenticated, the system handles transaction requests, such as cash withdrawals or balance inquiries, and interfaces with the bank for authorization.

5. Data Layer: Securely stores user profiles, biometric data, and transaction records.

6. Communication Module: Connects with banking systems to process and authorize transactions.

The system is designed to enhance security and user experience by providing a seamless, cardless authentication process while ensuring data protection and scalability.

5.1 ARCHITECTURE:

User Interface (UI): Where the user interacts with the ATM.
Processing Layer: Handles image capture, face recognition, and transaction processing.
Data Layer: Manages user and transaction data.
External Interfaces: Includes bank services for authentication and transaction processing.

The cardless ATM system is designed with four interconnected layers, each playing a crucial role in ensuring a secure and efficient user experience.

The User Interface (ATM Interface) is where the interaction between the user and the ATM begins. Users approach the ATM and use the touchscreen or display to initiate transactions and biometric

authentication. This layer provides the visual and interactive elements that guide users through the process of authentication and transaction requests, making the system user-friendly and accessible.

User Interface Layer

This is the front-end part of the system where the user interacts with the ATM machine. It includes:

- **Touchscreen Display**: Shows cardless login options and transaction menus.
- **High-Resolution Camera**: Captures real-time facial images for authentication.
- **Optional Biometric Input**: Can include a fingerprint scanner or microphone for multi-factor authentication (optional).
- **Notification System**: Displays OTP confirmations or error messages.

Application Layer (ATM Software)

This is the logic layer of the ATM which handles:

- **User Request Handling**: Detects when a user selects "Cardless Access".
- **Face Capture Module**: Captures and preprocesses the user's facial image (cropping, alignment, normalization).
- **Liveness Detection Module**: Detects blinking, motion, or 3D depth to verify the user is real.
- **Communication Handler**: Sends the facial data securely to the server for verification and receives the response.

Backend Server and Database Layer

This layer is hosted in the bank's data center or cloud infrastructure. It contains:

- **Facial Recognition Engine**: Uses AI models (like CNN-based models) to compare the live image with stored templates.
- **User Profile Database**: Stores encrypted facial templates, linked account information, and transaction history.
- **Authentication Manager**: Performs multi-factor checks like OTP validation or mobile push notification.
- **Transaction Processor**: Executes requested actions like balance inquiry, cash withdrawal, or fund transfers once the user is authenticated.

The Processing Layer handles the core operations required for the cardless ATM system. It begins with Image Capture, where a camera captures a high-resolution image of the user's face. Face Recognition then uses deep learning models to process and analyze the image, detecting and recognizing the user's face. Following this, Authentication verifies the user's identity by comparing the extracted facial features with stored biometric data. Once authenticated, the system proceeds with Transaction Processing, managing requests such as cash withdrawals or balance inquiries.

The Data Layer manages all essential databases. The User Database stores biometric information, including facial features, which is critical for authentication. The Transaction Database records all transaction details to ensure accurate and secure financial operations, providing a reliable record of user activities.

Finally, the Bank System Layer interfaces with the processing layer to handle interactions with the banking infrastructure. The Authentication Service communicates with the bank's system to validate the user's credentials based on the biometric data provided. The Transaction Service manages and authorizes financial transactions, ensuring that all requests are processed accurately and securely.

Overall, this design provides a robust framework for the cardless ATM system, integrating advanced biometric authentication and deep learning techniques to offer a secure and efficient banking experience without the need for physical cards.

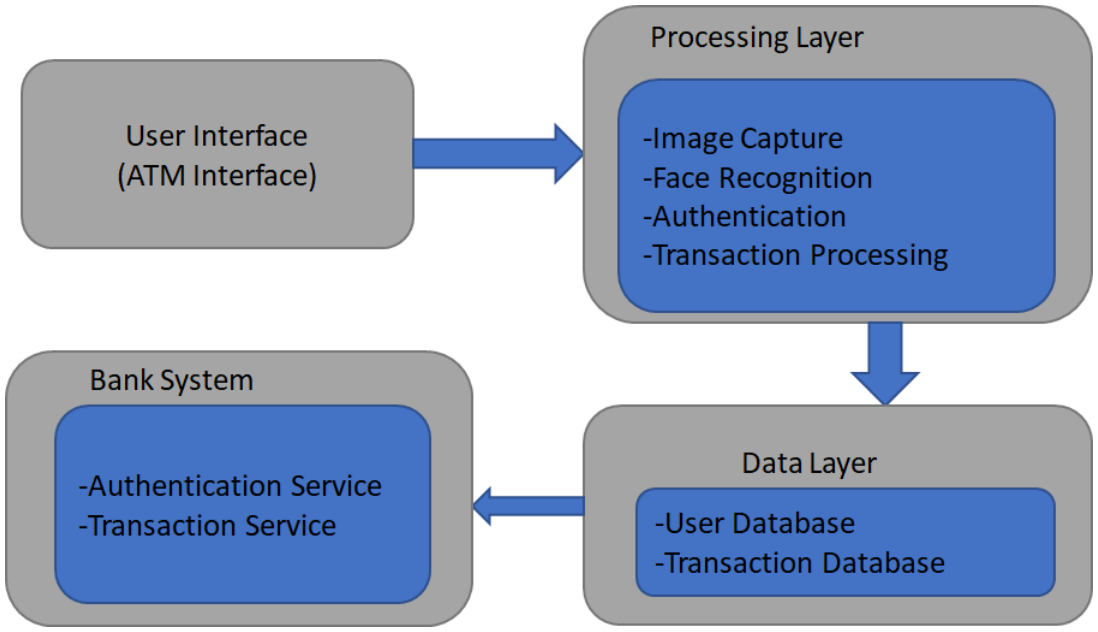
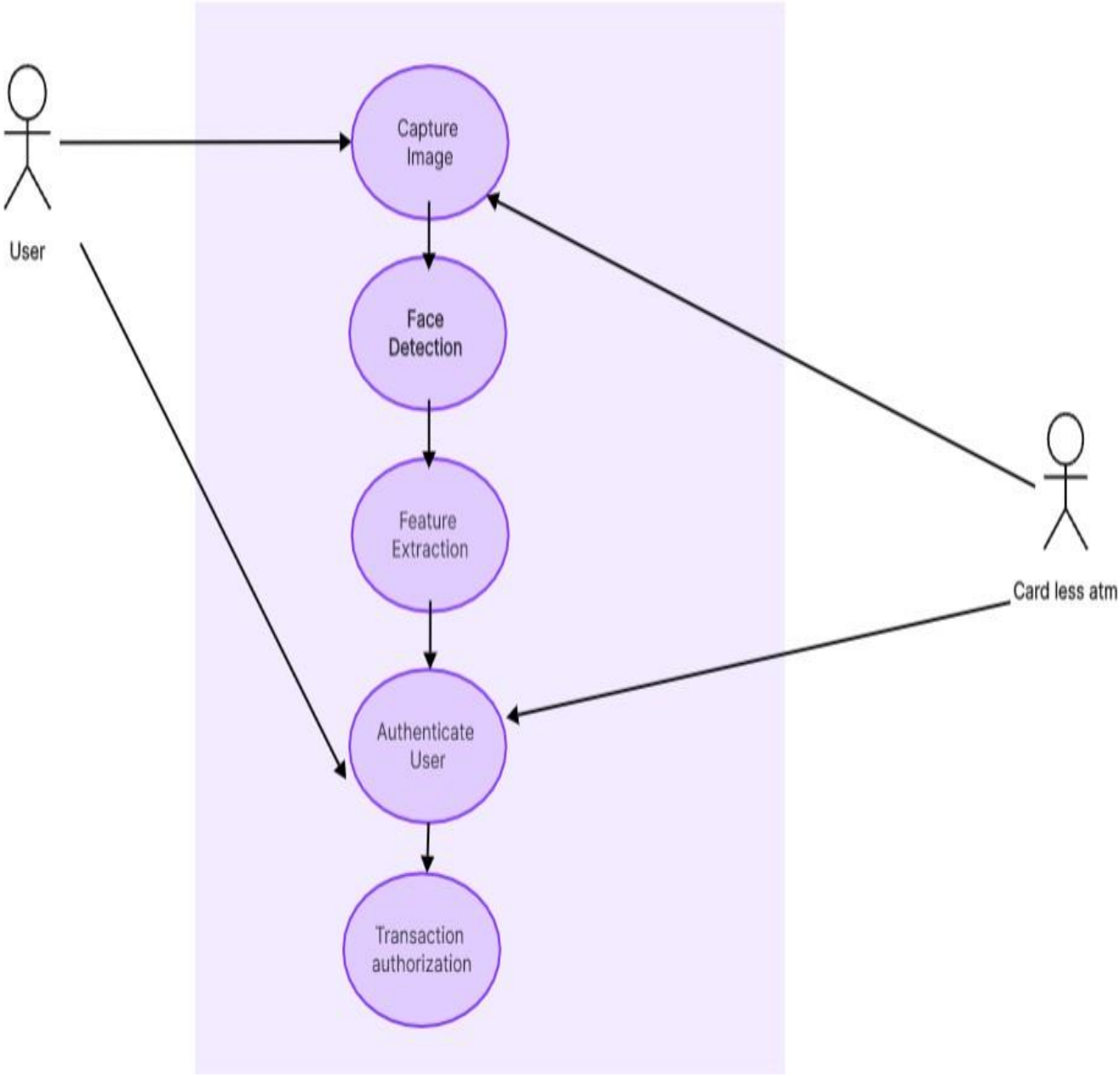


Fig. **Architecture Diagram**

5.2 UNIFIED MODELING LANGUAGE DIAGRAMS

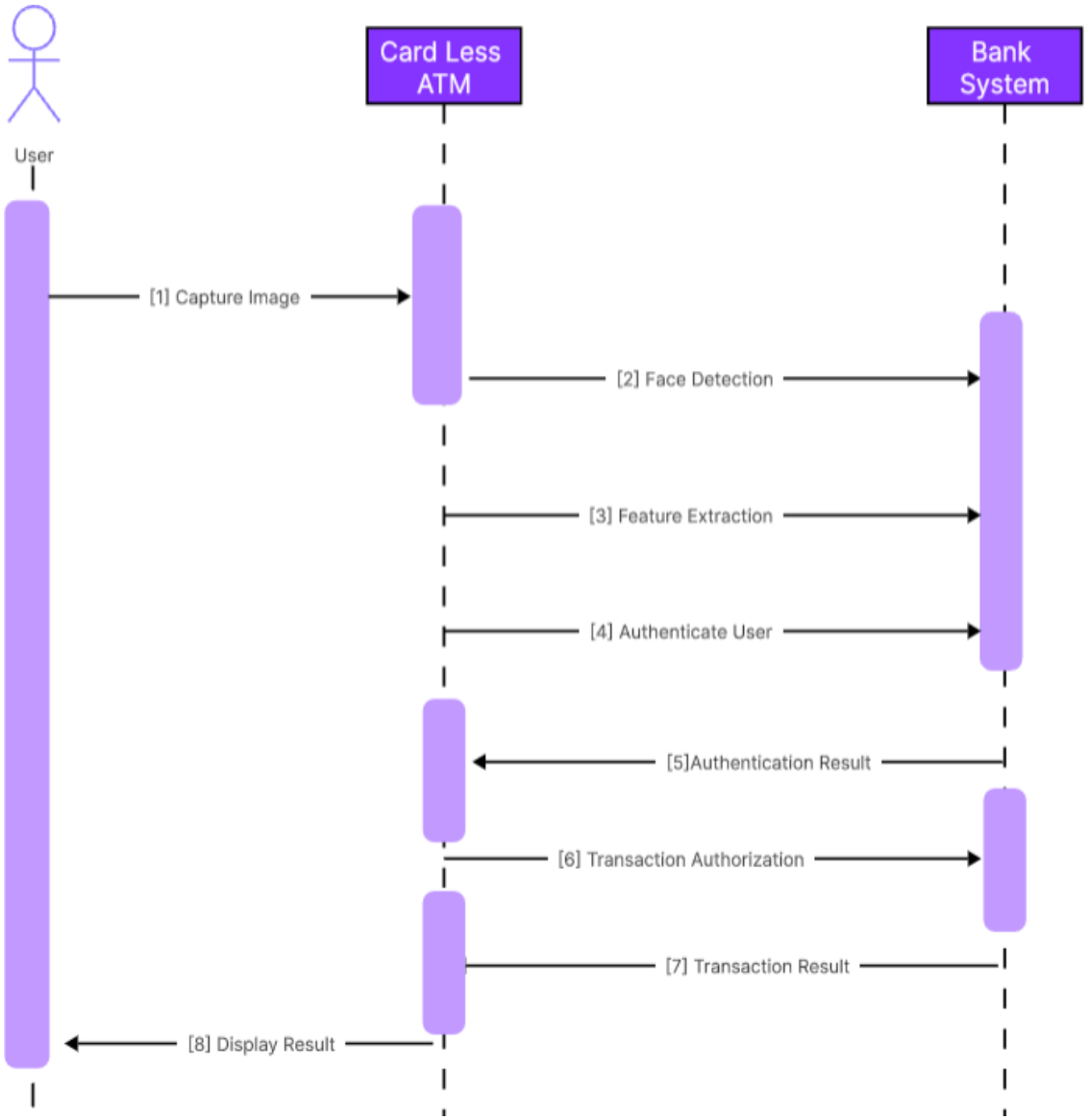
5.2.1 USE CASE Diagram

The use case diagram for the cardless ATM system illustrates interactions between the user and the ATM. It includes use cases for image capture, face detection, feature extraction, liveness detection, user authentication, and transaction processing, with the ATM acting as the primary actor facilitating secure, cardless transactions.



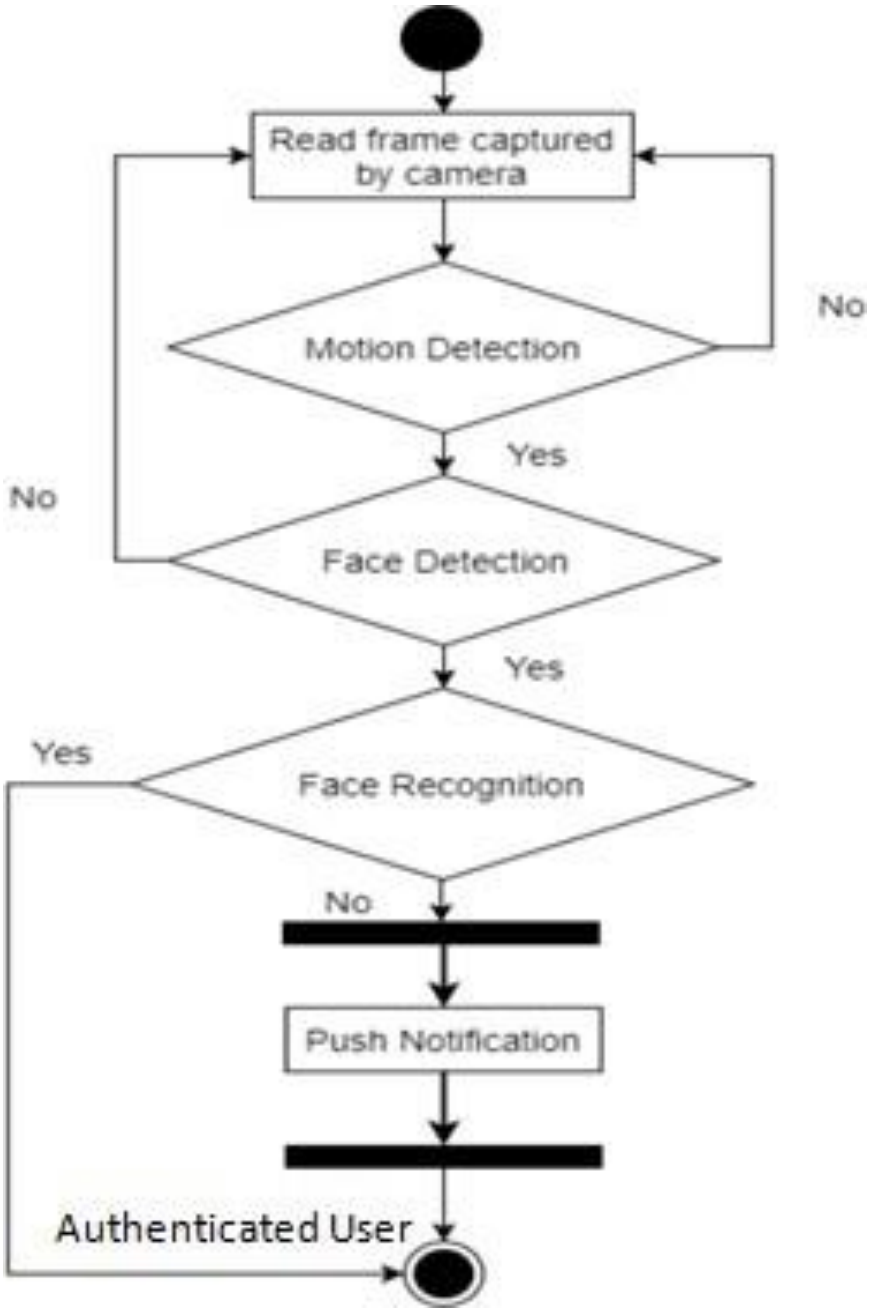
5.2.2 Sequence Diagram:

The sequence diagram for the cardless ATM system outlines the flow of interactions: the user initiates authentication via the interface, the system captures and processes the facial image, performs face recognition and liveness detection, verifies identity, and processes transactions, all while communicating with the bank system for authorization.



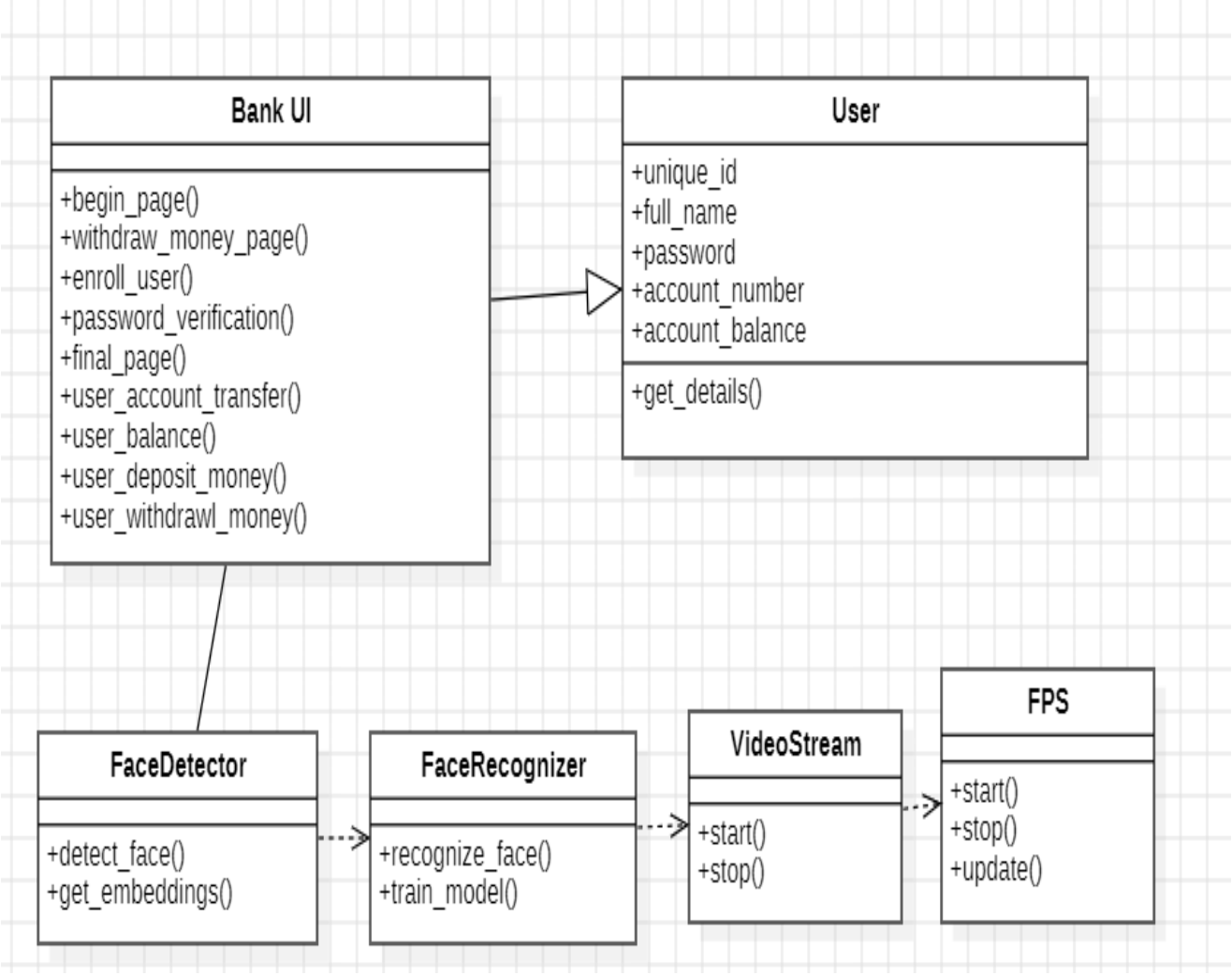
5.2.3 Activity Diagram:

The activity diagram for the cardless ATM system illustrates the sequence of steps from user interaction to transaction completion. It shows the process of capturing the user's facial image, performing face detection and recognition, verifying identity, and executing transactions, ensuring a smooth and secure cardless banking experience.



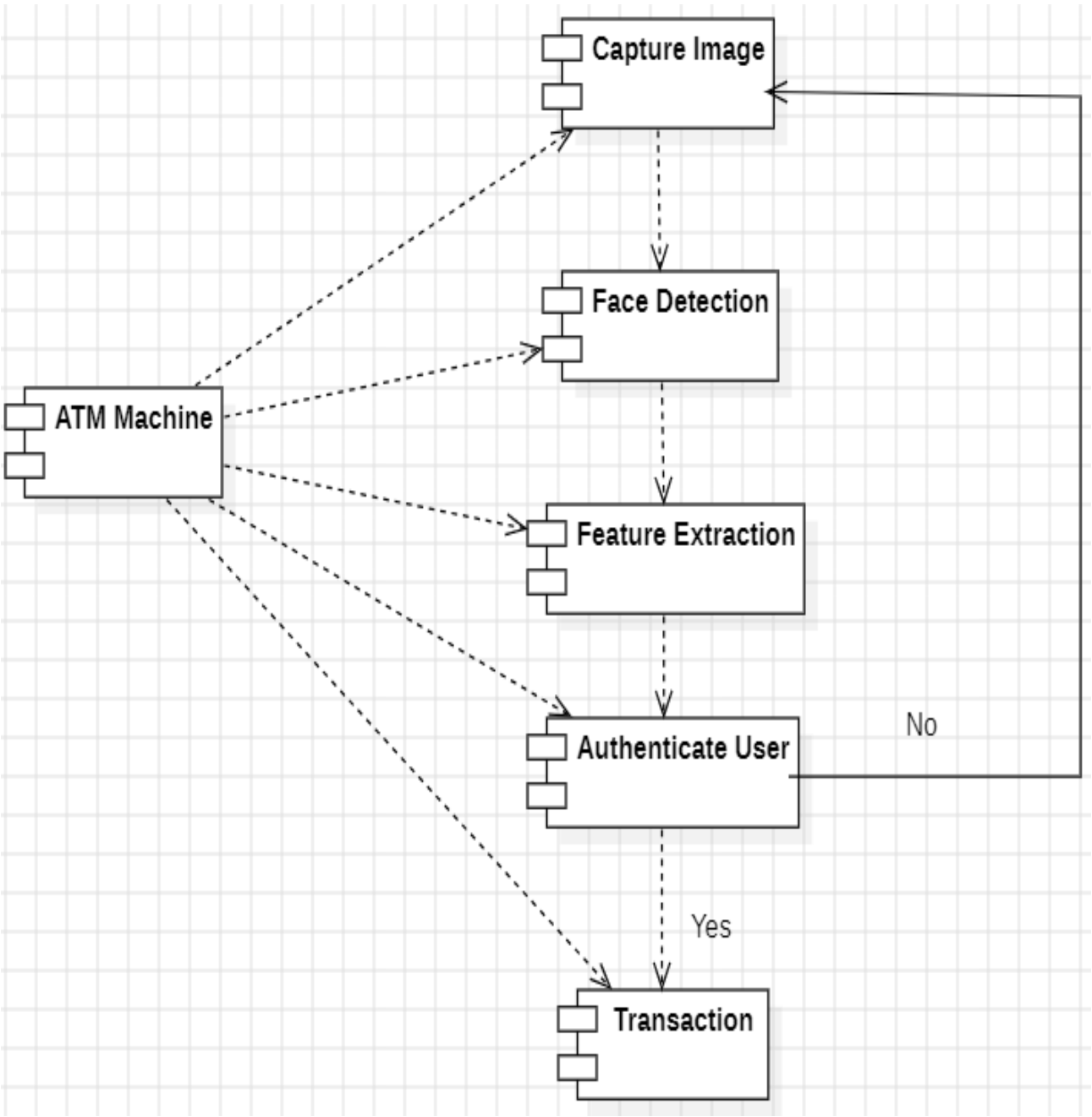
5.2.4 Class Diagram

A class diagram represents the static structure of a system, showing its classes, attributes, operations, and relationships among objects. This helps in understanding the system's architecture and provides a clear picture of the system's components and how they interact. By visualizing the classes and their relationships, developers can identify potential issues and improve the overall design of the system.



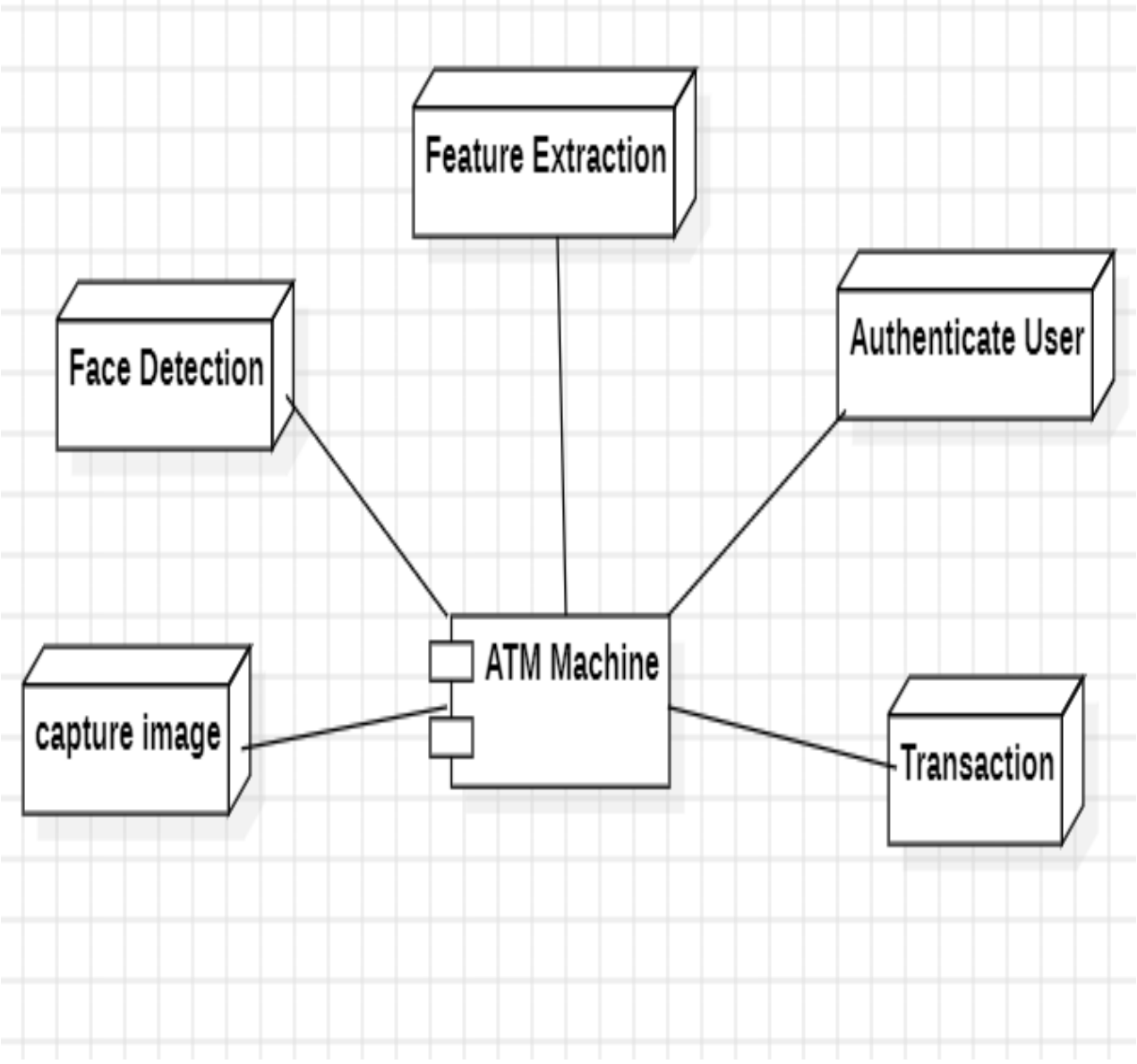
5.2.5 Component Diagram

A component diagram illustrates the organization and dependencies among a set of components in a system. It is useful for visualizing the high-level architecture of a software system and helps developers understand how the different components work together. The diagram consists of components, interfaces, and connectors that show the interactions between components, making it easier to identify dependencies and potential bottlenecks.



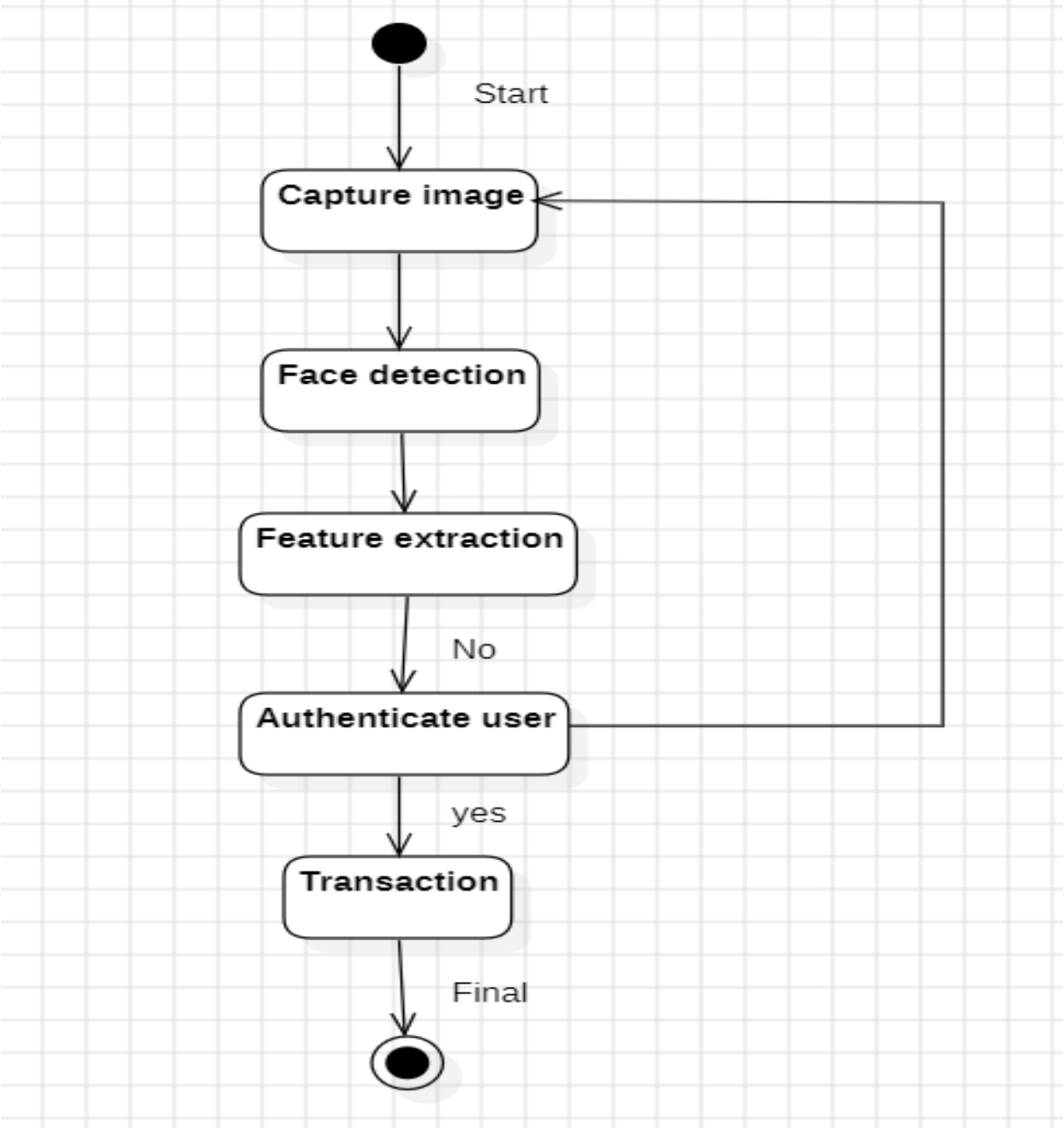
5.2.6 Deployment Diagram

A deployment diagram depicts the physical deployment of artifacts on nodes, visualizing the hardware topology of a system and how software components are distributed across it. This diagram is essential for understanding how the system will be deployed and how the different components will interact with each other in a physical environment. By considering the deployment diagram, developers can ensure that the system is scalable, efficient, and meets the required performance standards.



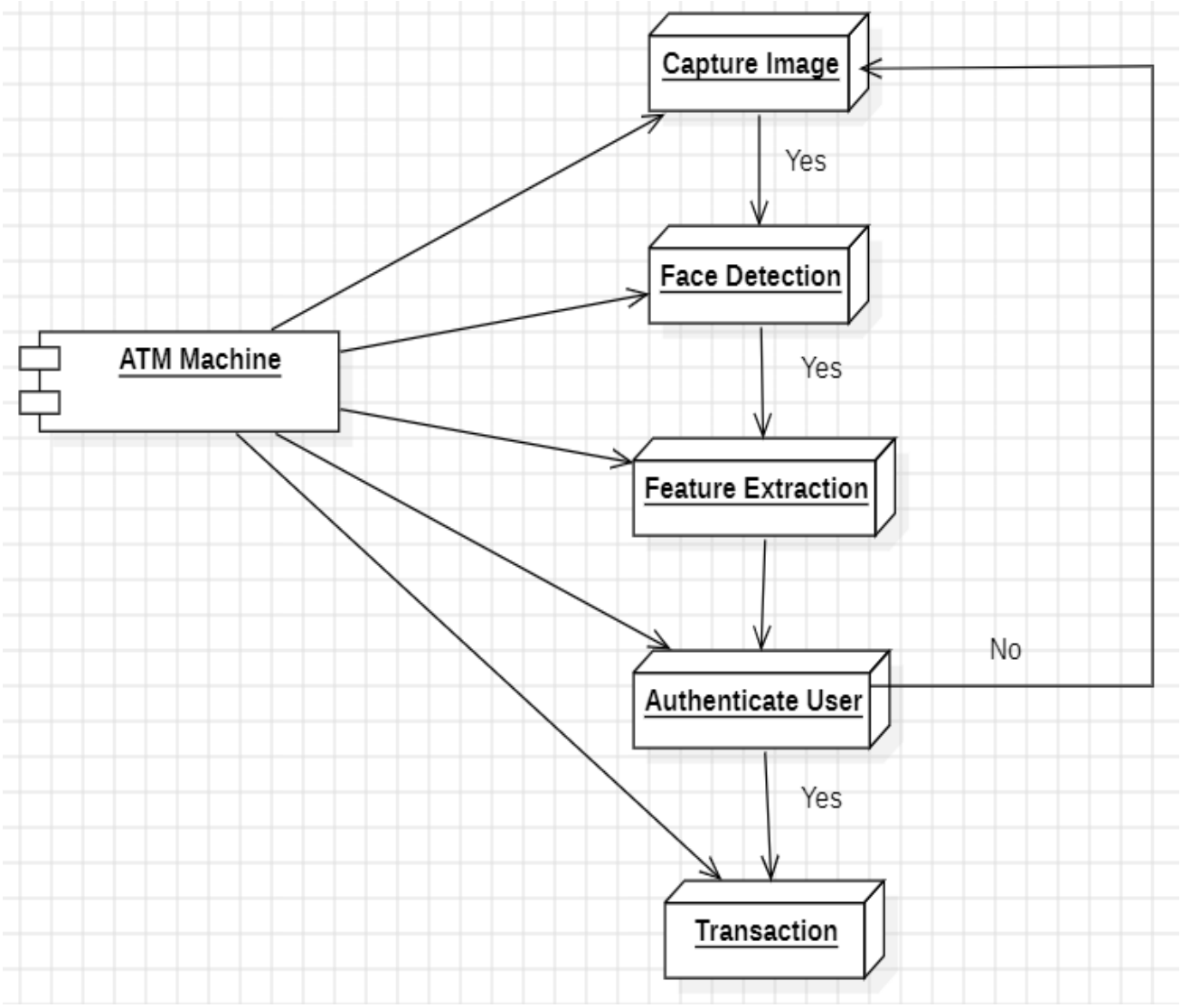
5.2.7 State Chart Diagram

A state chart diagram illustrates the dynamic behavior of a system, showing the states of an object and the transitions between those states in response to events. This diagram is useful for modeling complex behaviors and understanding how the system responds to different inputs and events. By visualizing the states and transitions, developers can identify potential issues and improve the overall behavior of the system.



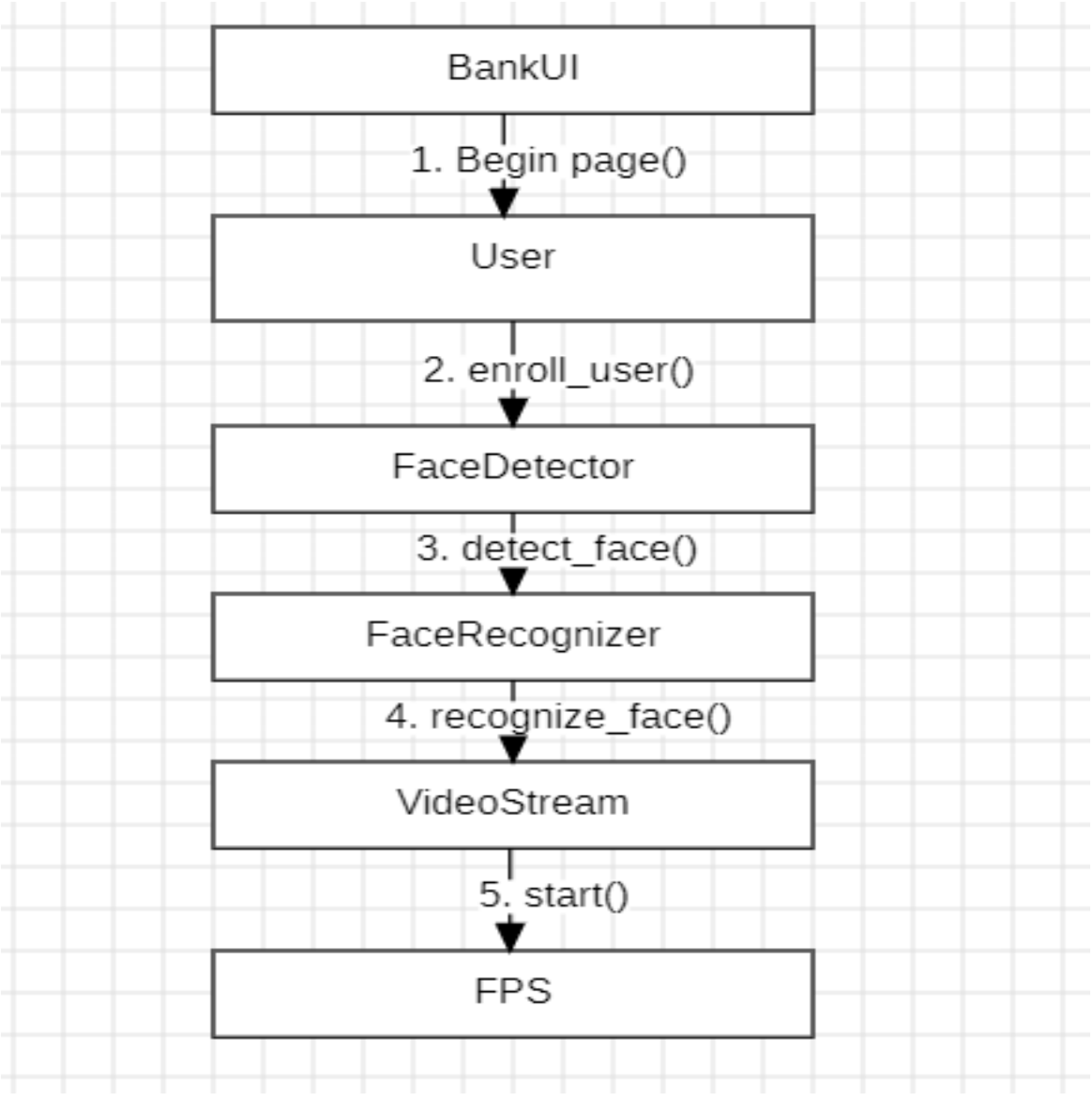
5.2.8 Object Diagram

An object diagram represents a snapshot of the instances of classes and their relationships at a specific point in time. It visualizes the state of a system during runtime, showing the objects, their attributes, and the links between them. This diagram is useful for understanding how the system behaves in different scenarios and how the objects interact with each other.



5.2.9 Collaboration Diagram

A collaboration diagram shows the interactions between objects in a system, emphasizing the links between them and the messages exchanged. This diagram is useful for understanding how the objects work together to achieve a common goal and how the messages are exchanged between them. By visualizing the interactions and messages, developers can identify potential issues and improve the overall communication between objects.



CHAPTER 6

6.CODE IMPLEMENTATION

Python

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

6.1Sample Code:

```
from imutils import paths
import numpy as np
import argparse
import imutils
import pickle
import cv2
from pathlib import Path
from collections import Counter
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

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

class BankUi:
    def __init__(self, root):
        self.root = root
          self.header = Label(self.root, text="Automated Teller Machine (ATM)", 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(x=340, y=340, width=200, height=40)
        self.button1.place(x=155, y=230, width=500, 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()

    def withdraw_money_page(self):
        self.frame.destroy()
        self.frame = Frame(self.root, bg="#0019fc", width=900, height=500)
        self.label1 = Label(self.frame, text="Note:", bg="#0019fc", fg="white", font=ARIAL)
        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)
        self.label3 = Label(self.frame, text="2.Each capture will take 15 seconds are you are required
to move your face in different directions while being captured.", bg="#0019fc", fg="white",
font=ARIAL)
        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)
        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)
         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)
         self.label7 = Label(self.frame, text="6.To begin, click the 'Verify Face Id' button below",
bg="#0019fc", fg="white", font=ARIAL)
            self.button = Button(self.frame, text="Verify Face Id", bg="#50A8B0", fg="white",
font=ARIAL, command=self.video_check)
             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.label1.place(x=100, y=100, width= 800, height=20)
        self.label2.place(x=100, y=120, width=800, height=20)
        self.label3.place(x=100, y=140, width=800, height=20)
        self.label4.place(x=100, y=160, width=800, height=20)
        self.label5.place(x=100, y=180, width=800, height=20)
        self.label6.place(x=100, y=200, width=800, height=20)
        self.label7.place(x=100, y=220, width=800, height=20)
        self.button.place(x=100, y=250, width=800, 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)
            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.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.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')
        if data[data.loc[:, 'unique_id'] == self.real_user].empty:
            messagebox._show("Verification Info!", "User  not found")
            self.begin_page()
        else:
                    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()

    def final_page(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()

    def user_account_transfer(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()
```

```
        self.label11 = Label(self.frame, text="Please enter the recipient's account number",
bg="#0019fc", fg="white", font=ARIAL)
        self.label21 = Label(self.frame, text="Please enter the amount to be transferred",
bg="#0019fc", fg="white", font=ARIAL)
    self.button1 = Button(self.frame, text="Transfer", bg="#50A8B0", fg="white", font=ARIAL,
command=self.user_account_transfer_transc)
     self.entry11 = Entry(self.frame, bg="honeydew", highlightcolor="#50A8B0",
                  highlightthickness=2,
                  highlightbackground="white")

  def user_account_transfer_transc(self):
    data = pd.read_csv('bank_details.csv')

    recipient_account = int(self.entry11.get())
    transfer_amount = int(self.entry21.get())

    # Validate recipient account number
    if recipient_account not in data['account_number'].values:
      messagebox.showerror("Transfer Info!", "Invalid account number")
      return

    # Validate if the user is trying to transfer to themselves
   if recipient_account == data[data['unique_id'] == self.real_user]['account_number'].values[0]:
      messagebox.showerror("Transfer Info!", "You cannot transfer money to yourself")
      return
    # Validate sufficient funds
    user_balance = data[data['unique_id'] == self.real_user]['account_balance'].values[0]
    if transfer_amount > user_balance:
      messagebox.showerror("Transfer Info!", "Insufficient balance")
      return
    # Perform the transfer
   data.loc[data['account_number'] == recipient_account, 'account_balance'] += transfer_amount
    data.loc[data['unique_id'] == self.real_user, 'account_balance'] -= transfer_amount
    data.to_csv('bank_details.csv', index=False)
    messagebox.showinfo("Transfer Info!", "Transfer successful!")
  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()
            data = pd.read_csv('bank_details.csv')
            text = data[data.loc[:,'unique_id'] ==
    self.real_user].loc[:,'account_balance'].values[0]
            self.label = Label(self.frame, text= 'Current Account
    Balance: ' + 'N' + str(text),font=ARIAL)
            self.label.place(x=200, y=100, width=300, height=100)



def train_model(self):
        #summary
        print("[INFO] loading face embeddings...")
        data = pickle.loads(open('output/embeddings.pickle', "rb").read())
        le = LabelEncoder()
        labels = le.fit_transform(data["names"])
        # train the model used to accept the 128-d embeddings of the face
and
        # then produce the actual face recognition
        print("[INFO] training model...")
        recognizer = SVC(C=1.0, kernel="linear", probability=True)
        recognizer.fit(data["embeddings"], labels)
        # write the actual face recognition model to disk
        f = open('output/recognizer.pickle', "wb")
        f.write(pickle.dumps(recognizer))
        f.close()

        # write the label encoder to disk
        f = open('output/le.pickle', "wb")
        f.write(pickle.dumps(le))
        f.close()



def video_check(self):
```

```
        detector =
cv2.dnn.readNetFromCaffe('face_detection_model/deploy.prototxt',
'face_detection_model/res10_300x300_ssd_iter_140000.caffemodel')
        #summary
        # load our serialized face embedding model from disk
        print("[INFO] loading face recognizer...")
        embedder = cv2.dnn.readNetFromTorch('nn4.small2.v1.t7')

        # load the actual face recognition model along with the label
encoder
        recognizer = pickle.loads(open('output/recognizer.pickle',
"rb").read())
        le = pickle.loads(open('output/le.pickle', "rb").read())

        # initialize the video stream, then allow the camera sensor to
warm up
        print("[INFO] starting video stream...")
        vs = VideoStream(src=0).start()
        time.sleep(2.0)

        #run check for only 15seconds and then stop
        timeout = time.time() + 5

        # start the FPS throughput estimator
        fps = FPS().start()

           # loop over frames from the video file stream
        real_user_list = []
        while True:

           #run check for only 15seconds and then stop
           if time.time() > timeout :
              cv2.destroyWindow("Frame")
              break;

           # grab the frame from the threaded video stream
           frame = vs.read()

           # resize the frame to have a width of 600 pixels (while
           # maintaining the aspect ratio), and then grab the image
           # dimensions
           frame = imutils.resize(frame, width=800, height=200)
           (h, w) = frame.shape[:2]

           # construct a blob from the image
           imageBlob = cv2.dnn.blobFromImage(
              cv2.resize(frame, (300, 300)), 1.0, (300, 300),
```

```
    (104.0, 177.0, 123.0), swapRB=False, crop=False)

# apply OpenCV's deep learning-based face detector to localize
# faces in the input image
detector.setInput(imageBlob)
detections = detector.forward()

#TODO: if 2 faces are detected alert the user of a warning
# loop over the detections
for i in range(0, detections.shape[2]):
    # extract the confidence (i.e., probability) associated with
    # the prediction
    confidence = detections[0, 0, i, 2]

    # filter out weak detections
    if confidence > 0.5:
        # compute the (x, y)-coordinates of the bounding box for
        # the face
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        # extract the face ROI
        face = frame[startY:endY, startX:endX]
        (fH, fW) = face.shape[:2]

        # ensure the face width and height are sufficiently large
        if fW < 20 or fH < 20:
            continue

        # construct a blob for the face ROI, then pass the blob
        # through our face embedding model to obtain the 128-d
        # quantification of the face
        faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255,
            (96, 96), (0, 0, 0), swapRB=True, crop=False)
        embedder.setInput(faceBlob)
        vec = embedder.forward()

        # perform classification to recognize the face
        preds = recognizer.predict_proba(vec)[0]
        j = np.argmax(preds)
        proba = preds[j]
        name = le.classes_[j]

        # # draw the bounding box of the face along with the
        # # associated probability
        # text = "{}: {:.2f}%".format(name, proba * 100)
        # y = startY - 10 if startY - 10 > 10 else startY + 10
```

```
                # cv2.rectangle(frame, (startX, startY), (endX, endY),
                #     (0, 0, 255), 2)
                # cv2.putText(frame, text, (startX, y),
                #     cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0, 0, 255),
2)
                #TODO: Handle if 2 faces are given.
                #Decision boundary
                if (name == 'unknown') or (proba *100) < 50:
                    print("Fraud detected")
                    real_user_list.append(name)
                else:
                    #cv2.destroyWindow("Frame")
                    real_user_list.append(name)
                    break;


            # update the FPS counter
            fps.update()

            # show the output frame
            cv2.imshow("Frame", frame)
            key = cv2.waitKey(1) & 0xFF

            # if the `q` key was pressed, break from the loop
            if key == ord("q"):
                break

        # stop the timer and display FPS information
        fps.stop()
        print("[INFO] elasped time: {:.2f}".format(fps.elapsed()))
        print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))


        # do a bit of cleanup
        cv2.destroyAllWindows()
        vs.stop()
        print(real_user_list)

        try:
            Counter(real_user_list).most_common(1)[0][0] == 'unknown'
        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("Its My Bank")
root.geometry("800x500")
root.configure(bg="blue")
# icon = PhotoImage(file="IMG-f-WA0011 copy.png")
# root.tk.call("wm",'iconphoto',root._w,icon)
obj = BankUi(root)
root.mainloop()
```

6.2 Output Screens:

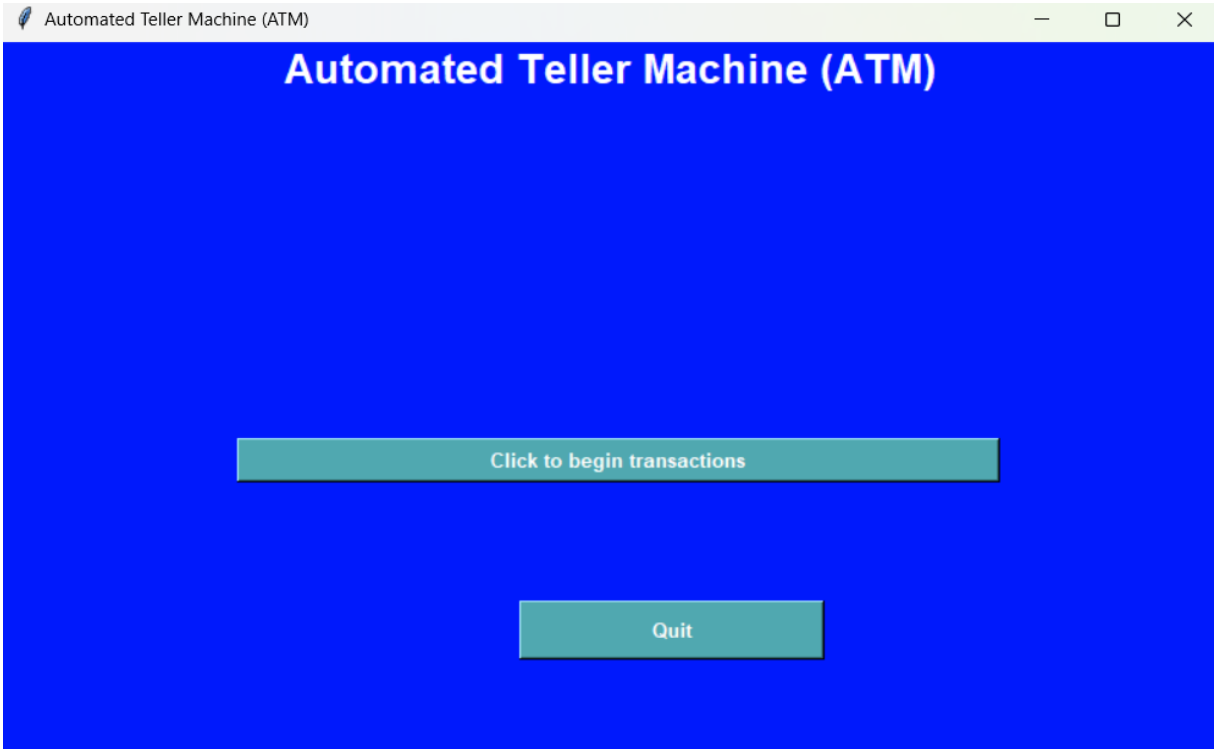


Fig 6.2.1 User Interface

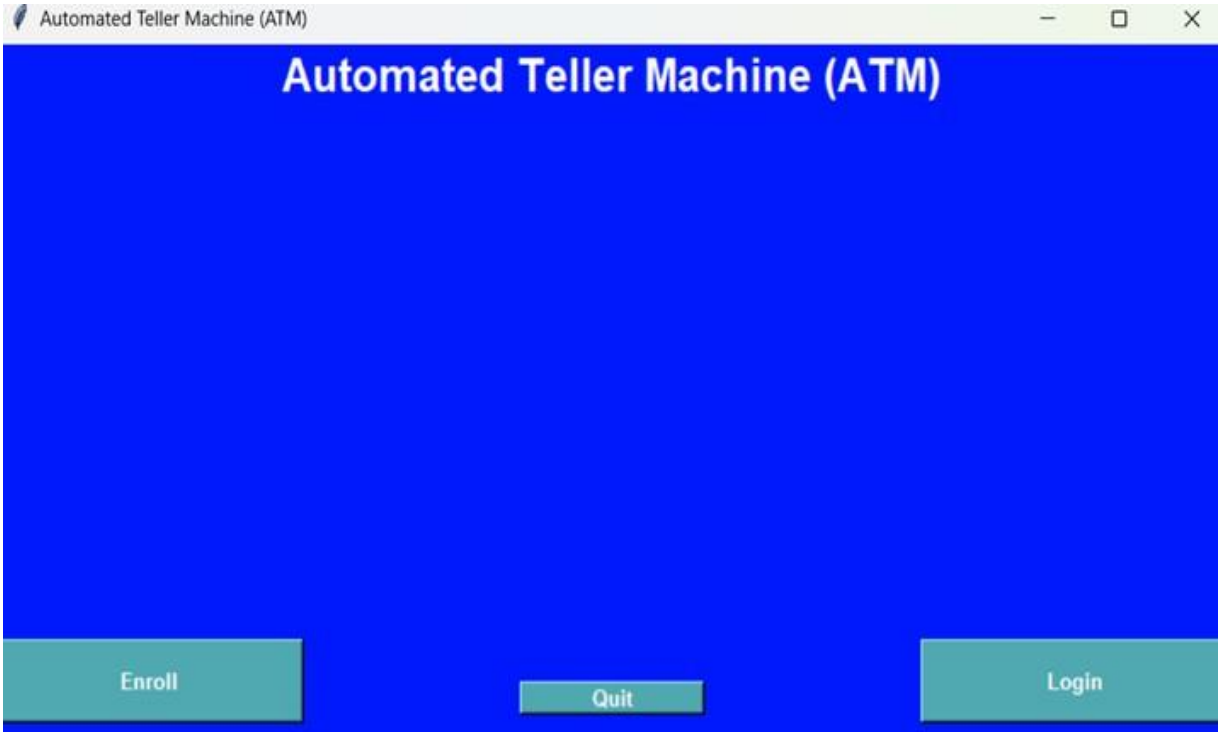


Fig 6.2.2 Enroll and login interface

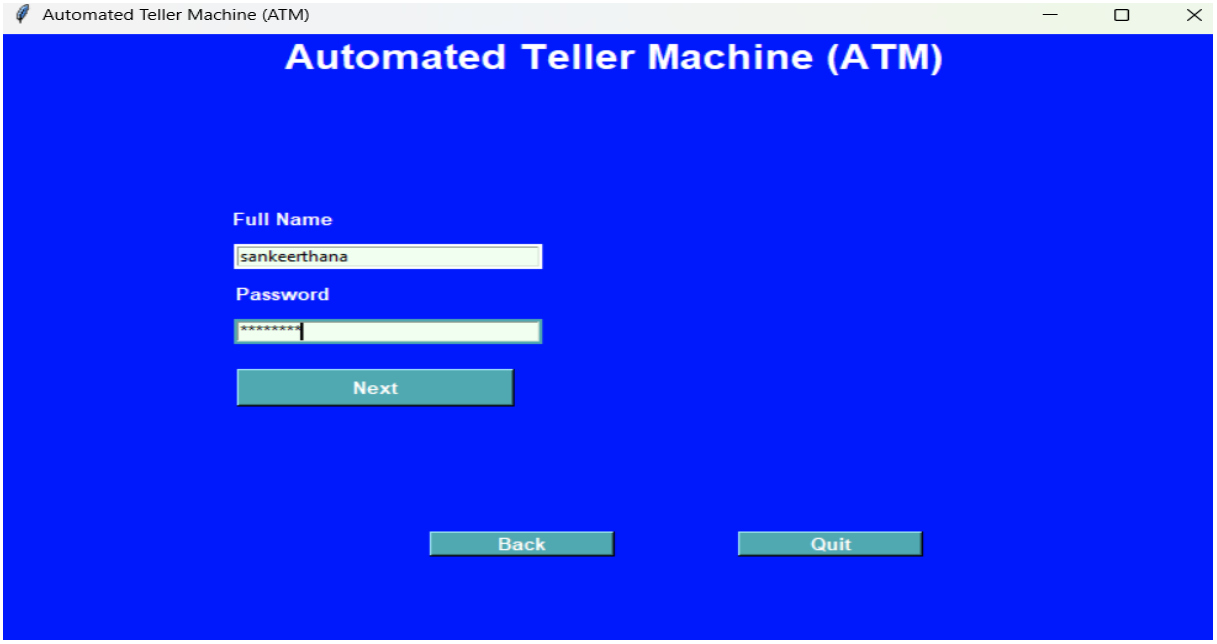


Fig 6.2.3 Saving user details

Fig 6.2.4 Capture face

Fig 6.2.6 Verification to login into user account

Fig 6.2.7 Successful face id match

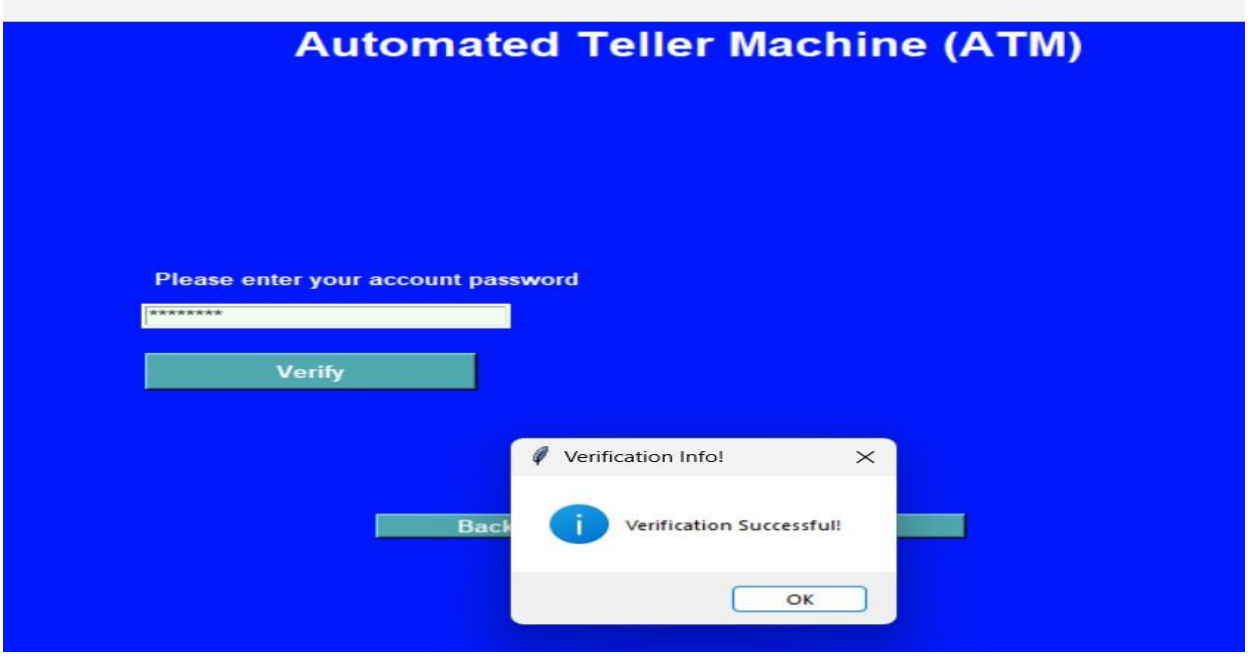


Fig 6.2.8 Verification through password

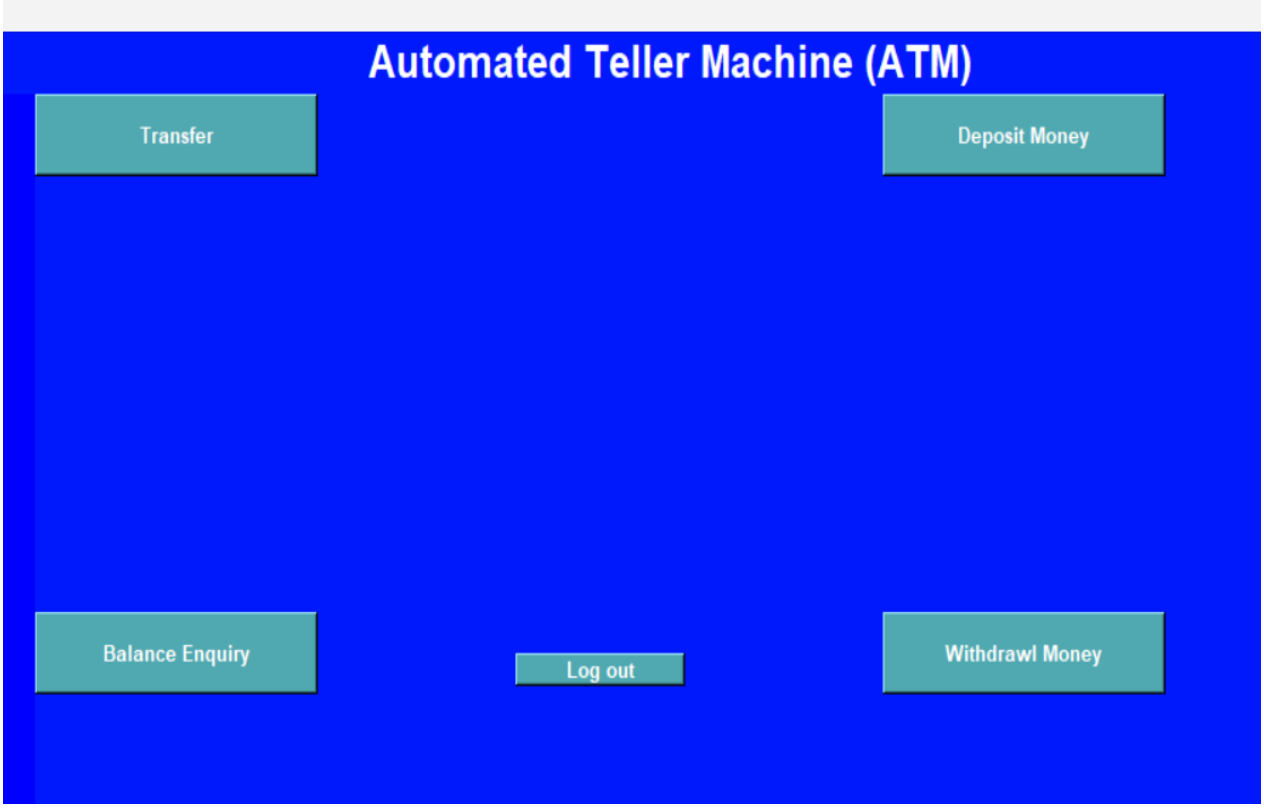


Fig 6.2.9 ATM interface

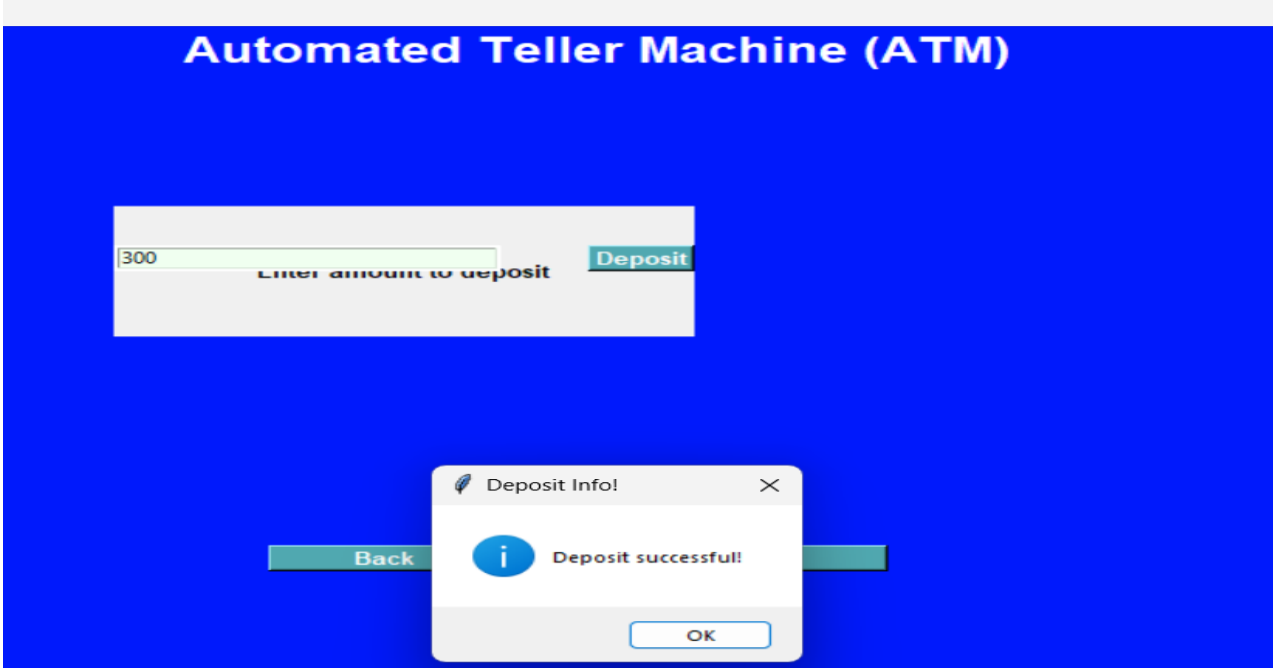
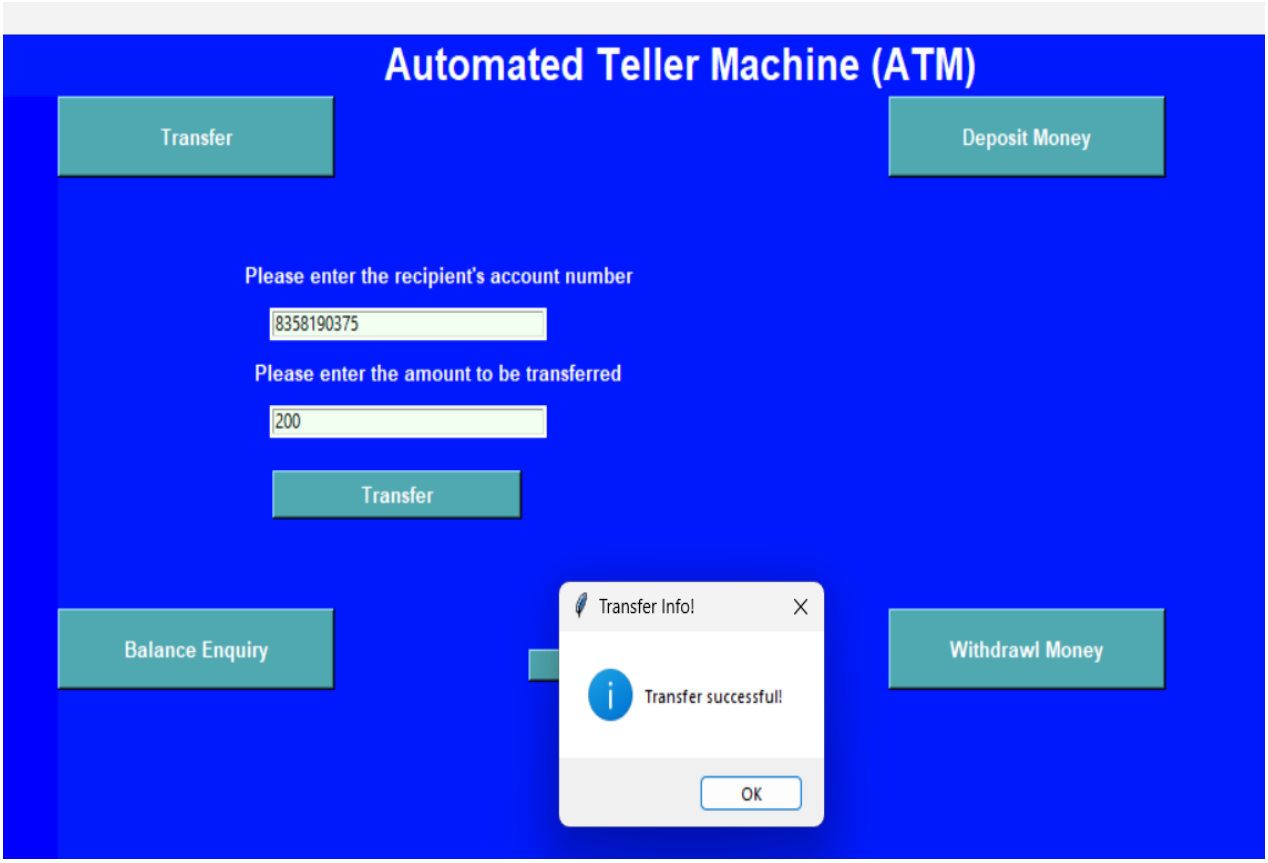


Fig 6.2.10 Money deposit



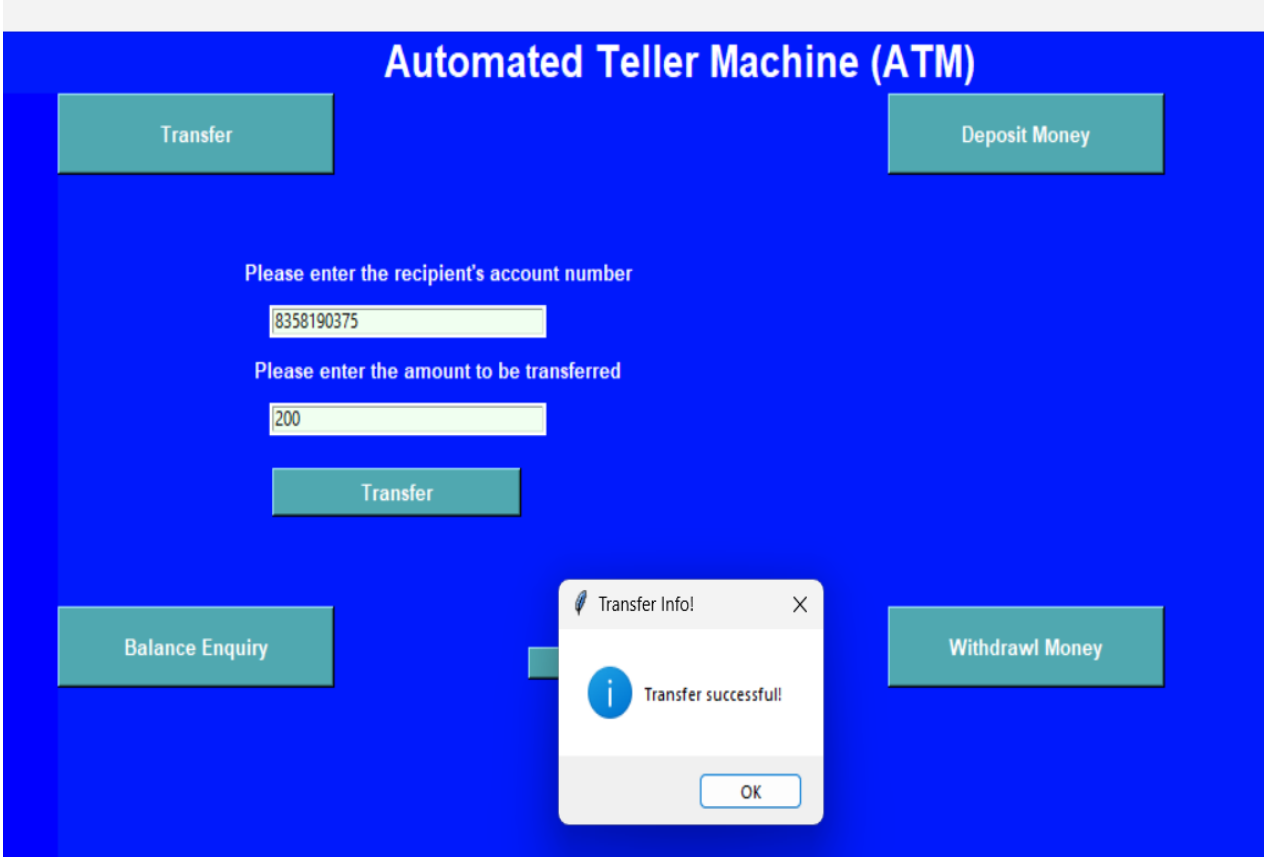


Fig 6.2.11 Money tranfer to another account

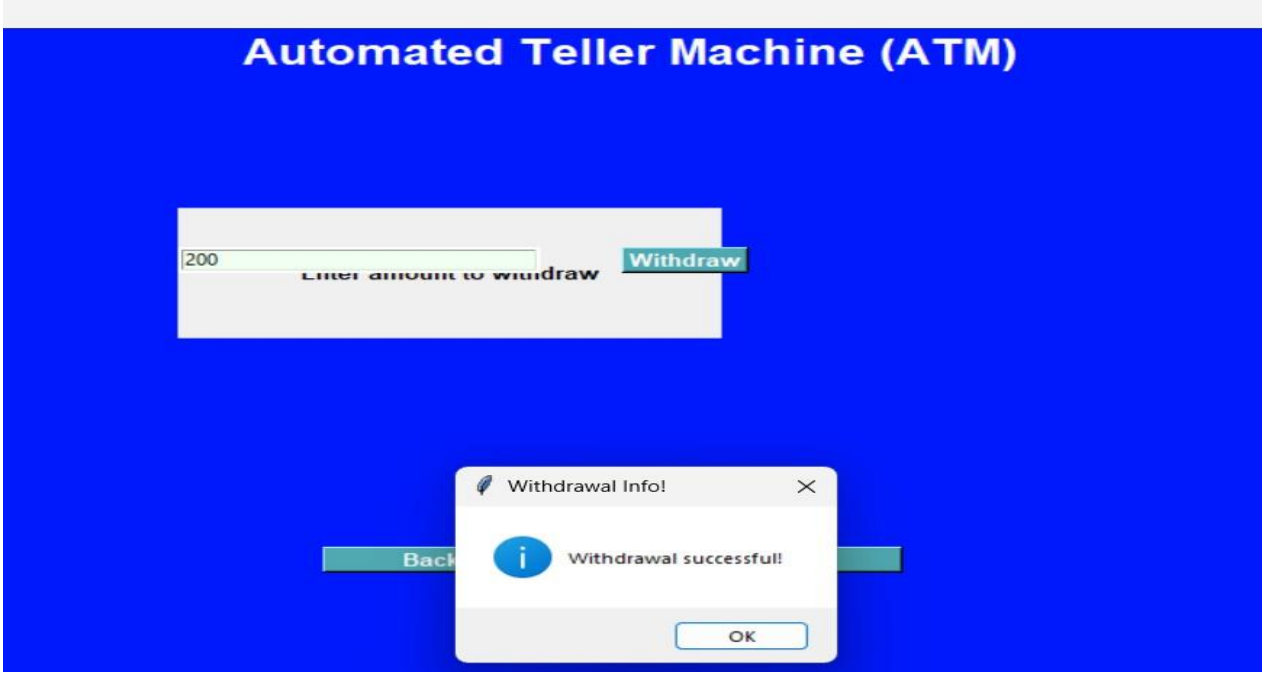
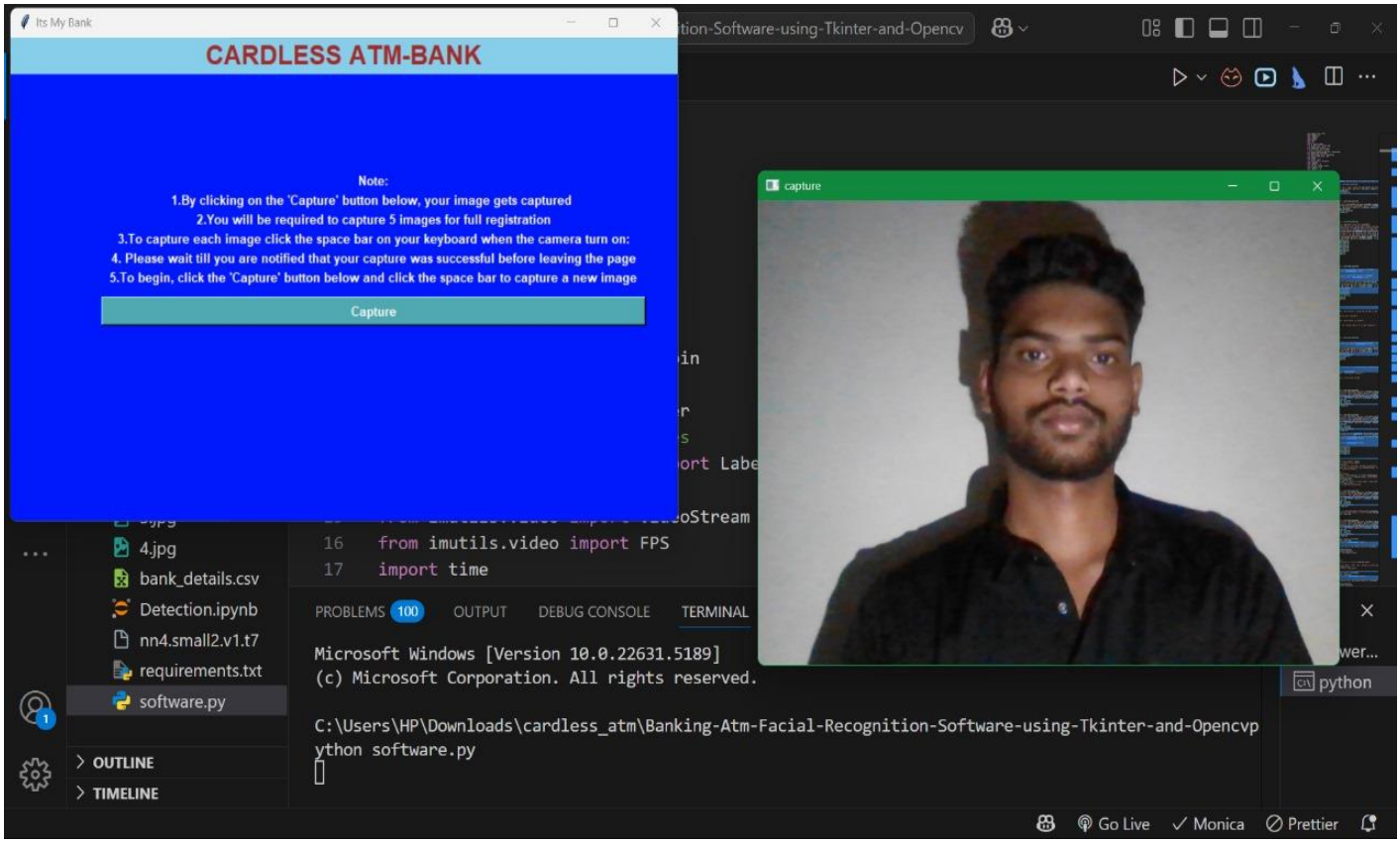
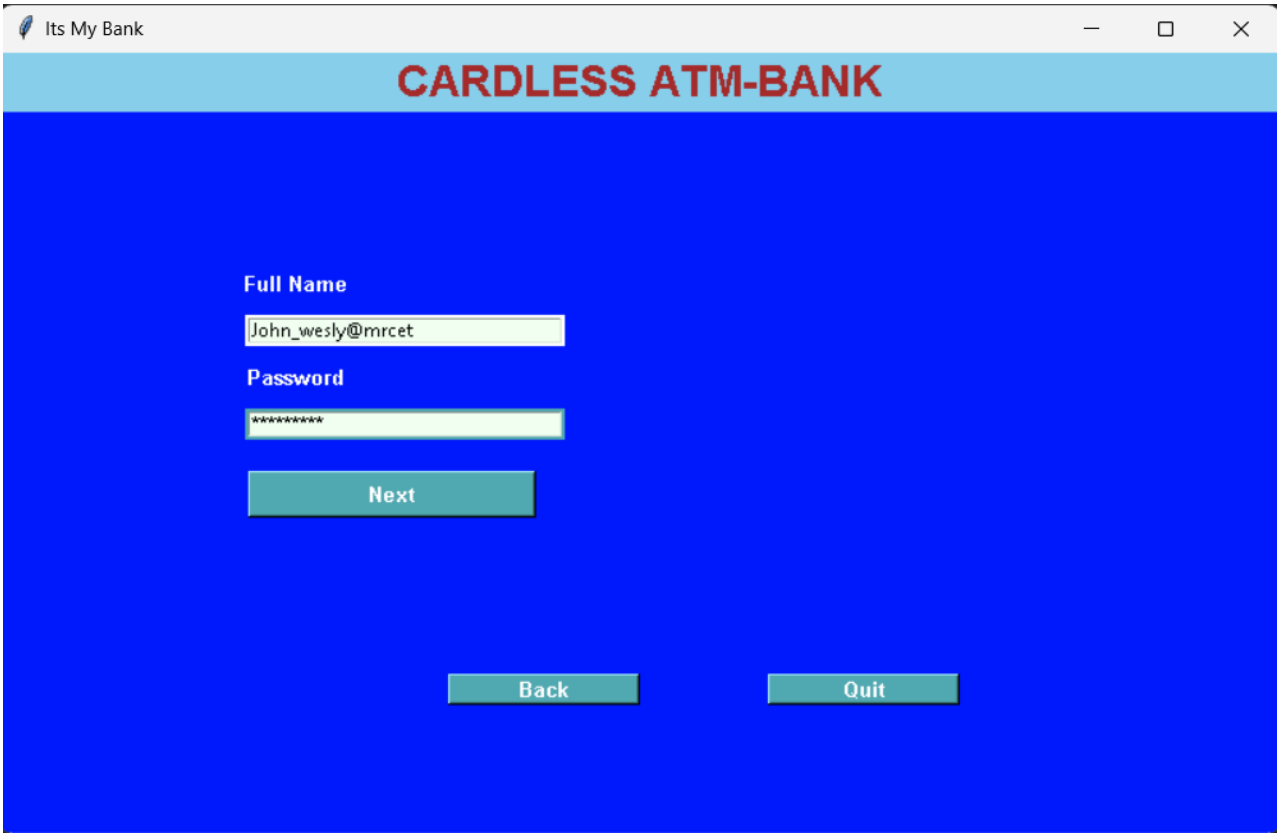


Fig 6.2.12 Successful money withdrawal


Its My Bank
CARDLESS ATM-BANK
Note:
1.By clicking on the 'Capture' button below, your image gets captured
2.You will be required to capture 5 images for full registration
3.To capture each image click the space bar on your keyboard when the camera turn on:
4. Please wait till you are notified that your capture was successful before leaving the page
5.To begin, click the 'Capture' button below and click the space bar to capture a new image
Capture
capture
4.jpg
bank_details.csv
Detection.ipynb
nn4.small2.v1.t7
requirements.txt
software.py
from imutils.video import FPS
import time
PROBLEMS 100
OUTPUT
DEBUG CONSOLE
TERMINAL
Microsoft Windows [Version 10.0.22631.5189]
(c) Microsoft Corporation. All rights reserved.
C:\Users\HP\Downloads\cardless_atm\Banking-Atm-Facial-Recognition-Software-using-Tkinter-and-Opencvpython software.py
OUTLINE
TIMELINE
python
Go Live
Monica
Prettier


Its My Bank
CARDLESS ATM-BANK
Full Name
John_wesly@mrcet
Password
****
Next
Back
Quit

7.CONCLUSION AND FUTURE SCOPE

Conclusion:

The implementation of a Cardless ATM Banking System using facial recognition techniques provides a secure, convenient, and user-friendly alternative to traditional ATM transactions. By leveraging biometric authentication, the system eliminates the need for physical ATM cards and PINs, thereby reducing the risk of card theft, skimming, and unauthorized access. The facial recognition system ensures that only the authorized user can access their account, enhancing overall banking security. The project demonstrates that facial recognition can be integrated effectively into ATM systems with acceptable accuracy and reliability, offering a modern approach to banking services.

Future Scope:

1. ;’’’’’’’’’’’’’;’;Future versions of the system can incorporate additional biometric features such as fingerprint or iris scanning for enhanced security.
2. **AI and Deep Learning Improvements:**
   Incorporating advanced deep learning models can improve facial recognition accuracy even in low-light conditions, varying angles, and for users with changes in appearance.
3. **Cloud-Based Authentication:**
   Moving the facial data processing to secure cloud infrastructure can reduce hardware dependencies and improve scalability.
4. **Liveness Detection:**
   To avoid spoofing using photos or videos, implementing liveness detection techniques can further improve the robustness of the system.
5. **Wider Banking Integration:**
   The system can be extended beyond ATMs to include secure access to online banking, mobile apps, and branch services