**FACE SHIELD ATM USING IMAGE PROCESSING & DEEP LEARNING TECHNIQUES**

### A Major Project Report Submitted

**In partial fulfillment of the requirements for the award of the degree of**

**Bachelor of Technology In**

**Electronics and Communication Engineering**

**By**

1. Y. Sai Sree S190705

2. K. Usha Rani S190606

3. M. Lalitha S190266

4. M.Chamundeswara Rao S190939

Under the esteemed guidance of

K.Jayanthi M.Tech



### Department of Electronics and Communication Engineering

**RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES**

(Affiliated to RGUKT Andhra Pradesh, Approved by AICTE)

IIIT Srikakulam Campus, SM Puram, Etcherla Mandal, Srikakulam-532402,

Andhra Pradesh, India.

Website: <https://rguktsklm.ac.in/>

### 2021-2025



**RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES**

(Affiliated to RGUKT Andhra Pradesh, Approved by AICTE)

IIIT Srikakulam Campus, SM Puram, Etcherla Mandal, Srikakulam-532402,

Andhra Pradesh, India.

Website: <https://rguktsklm.ac.in/>

**CERTIFICATE**

This is to certify that this is the Bonafide record of the project entitled **“Face Shield ATM Using Image Processing & Deep Learning Techniques”**, submitted by **Y.Sai Sree(s190705),K.Usha Rani(S190606),M.Lalitha(s190266),M.Chamundeswara Rao(s190939)** of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Electronics and Communication Engineering, Department of ECE, during the year 2024-2025. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

**Internal Guide Head of the Department K.Jayanthi K.Sudha Rani**

**Associate Professor Associatet. Professor**

**External Examiner**

# ACKNOWLEDGMENT

We feel honored to place our warm salutation to our college Rajiv Gandhi University Of Knowledge Technologies for allowing us to do this Project as part of our B. Tech Program. We are ever grateful to our Director **Prof.K.V.G.D.Balaji** who enabled us to have experience in engineering and gain profound technical knowledge.

We express our heartiest thanks to our HOD, K.**Sudha Rani** for encouraging us in every aspect of our course and helping us realize our full potential.

We would like to thank our Project Guide **K.Jayanthi** for their regular guidance, suggestions, constant encouragement, continuous monitoring, and unflinching cooperation throughout project work who despite being busy with his academic duties took time to guide and keep us on the correct path.

We would also like to thank all the faculty members and supporting staff of the Department of ECE and all other departments who have helped directly or indirectly in making our project a success.

We are extremely grateful to our parents for their blessings and prayers for the completion of our project which gave us the strength to do our project.

With regards and gratitude

**1. Y. Sai Sree S190705**

**2. K. Usha Rani S190606**

**3. M. Lalitha S190266**

**4. M.Chamundeswara Rao S190939**

# ABSTRACT

# An ATM (Automated Teller Machine) is an electronic machine used for financial transactions. As the term implies, it is an ‘automated’ banking platform that does not require any banking representative/teller or a human cashier. ATM full form is Automated Teller Machine which is a self-service banking outlet. You can withdraw money, check your balance, or even transfer funds. Different banks provide their ATM services by installing cash machines in different parts of the country. You can withdraw money from any of these machines irrespective of whether or not you are an account holder in the same bank. Today, ATMs and Credit cards are utilized with the end goal of cash exchanges which assume a fundamental job in the nature of exchange. The shortcomings of existing validation plan, for example, secret key and PIN number caused the spillage of data put away in ATM smartcard which lead to the lost of cash in ledger and private data abuses.

# To conquer this inadequacy of theft in cash exchanges, we propose the thought of utilizing face capturing of who is going to use the card, before utilizing the ATM functionalities. First it recognition the user face and match with authorized face which is in the database if it matched then it directly allow the user to perform ATM operation. Otherwise it sends the user face image to the account holder phone number through sms. Once the user verifies image then account holder may or may not grant the permissions for usage of ATM operations through sharing OTP. Using Deep Learning Convolutional Neural Network (CNN) architecture, the proposed system uses automatic helmet detection to alert users when they are wearing a helmet or mask in ATMs. Additionally, Face Recognition is accomplished by using Multitask Convolutional Neural Network (MTCNN), which creates a bounding box around the user's face and uses FaceRecognition to identify the face within it.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S. NO** | **TITLE** | **PG. NO** |
| **1** | **INTRODUCTION** | 1 |
|  | 1.1 Purpose and Objectives | 2 |
|  | 1.2 Existing and Proposed System | 3 |
|  | 1.3 Scope of Project | 5 |
| **2** | **LITERATURE SURVEY** | 6 |
| **3** | **SYSTEM ANALYSIS** | 8 |
|  | 3.1 Hardware and Software Requirements | 8 |
|  | 3.2 Software Requirements Specification | 9 |
| **4** | **SYSTEM DESIGN** | 10 |
|  | 4.1 Description | 10 |
|  | 4.2 Architecture | 11 |
|  | 4.3 Uml Diagrams | 12 |
| **5** | **METHODOLOGY** | 19 |
|  | 5.1 Technologies Used | 19 |
|  | 5.2 Modules Description | 22 |
|  | 5.3 Algorithms | 28 |
| **6** | **IMPLEMENTATION** | 31 |
|  | 6.1 Sample Code | 31 |
|  | 6.2 Output Screens | 45 |
|  | 6.3 Test Cases | 49 |
| **7** | **CONCLUSION &FUTURE SCOPE** | 50 |
| **8** | **BIBLIOGRAPHY** | 52 |

**CHAPTER 1 INTRODUCTION**

People today desire a simple and pleasant way of life in the day-to-day world. Because of this and its convenience, the use of automated teller machines (ATMs) has grown significantly. The purpose of an ATM (Automated Teller Machine) is to provide convenient and efficient access to banking services for customers. ATMs allow customers to perform a variety of banking transactions, including withdrawing cash, checking account balances, transferring money between accounts, depositing cash or checks, and paying bills. Overall, the purpose of an ATM is to make banking more convenient and accessible for customers, while also providing a cost-effective way for banks to provide their services to a wider audience. ATM fraud is a type of financial fraud that occurs when criminals use stolen or counterfeit bank cards or personal information to withdraw money or steal sensitive data from ATMs. This machine allows customers to get cash without having to physically visit the bank each time they need to withdraw cash. But as ATM usage has expanded, so too have crimes associated to it, and they pose a serious threat to the financial industry. ATM pin releases and card cloning are dangers.

Despite the existence of CCTV cameras installed within and surrounding the ATM, ATM robberies may still occur. In order to address the issues with the current ATM system, facial recognition is being used in our project. The Greek word for life, bio, and the word metrics, metrics (measurements), combine to form the phrase biometrics. Because biometric technologies serve to keep user data extremely secure by allowing users to be distinguished from one another based solely on their unique bodily characteristics, they provide a safe method of authentication. Enhanced Security for ATMs using Digital Image Processing initially uses Convolutional Neural Networks (CNN) to determine whether the user's face is obscured by a mask, scarf, or helmet. CNNs are a type of Neural Networks that are utilized in applications like classification and image recognition. Second, it uses FaceRecognition and Multi-task Cascade Convolutional Neural Networks to do face recognition searches for the user's face in the database that has been stored. FaceRecognition is a system that learns a mapping from face images to a compact Euclidean space where distances directly equate to a measure of face similarity. MTCNN is a neural network that detects faces and facial landmarks on images. If the face is matches with authorized face which is stored in the database it allow the user to access the ATM operations. Otherwise it stores the user face image in the image host server and generate a link for that image. Finally, it uses Fast2sms Service to send an OTP to the card owner if the face does not match the pin and the picture data that is recorded in the database. It is an online service that arranges and controls the transmission of messages to clients or endpoints that have signed up for them. Then the account holder can share the OTP to the user if the user was genuine.

## 1.1 Purpose and Objectives

The purpose of the project "Enhanced Authorization for ATM Machine through Facial Recognition" is to revolutionize the security and user experience of ATM transactions by integrating cutting-edge facial recognition technology into the authentication process. Traditional methods of ATM security, predominantly reliant on PINs and physical cards, have become increasingly vulnerable to sophisticated fraud techniques such as card skimming and identity theft. By introducing facial recognition, the project aims to fortify ATM security measures, effectively combatting these threats while also offering users a seamless and convenient means of authentication. Beyond merely enhancing security, this initiative seeks to transform the way individuals interact with ATMs, providing a user-friendly alternative to conventional authentication methods. Through facial recognition, users can access ATM services with a simple glance, eliminating the need for physical cards and memorized PINs. This not only streamlines the transaction process but also addresses accessibility concerns for users with disabilities or difficulties using traditional authentication methods. Moreover, by staying abreast of technological advancements and consumer preferences, financial institutions can instill greater trust and confidence in their banking systems, reinforcing their commitment to providing secure and innovative solutions for their customers. Ultimately, the integration of facial recognition technology into ATM machines represents a proactive step towards safeguarding financial assets and enhancing the overall banking experience for users.

**Objectives:**

**Improve Security:** Implement facial recognition technology to enhance the security of ATM transactions by adding an additional layer of authentication beyond traditional PINs and cards.

**Prevent Fraud:** Reduce the likelihood of fraudulent transactions by ensuring that only authorized users can access ATM services through facial recognition authentication.

**Enhance User Experience:** Provide a seamless and convenient user experience by eliminating the need for physical cards and remembering PINs, thereby streamlining the authentication process.

**Increase Accessibility:** Cater to a wider range of users, including those with disabilities or difficulties using traditional ATM authentication methods, by offering an alternative authentication option through facial recognition.

* 1. **Existing and Proposed System**
     1. **Existing System**

Automated Teller Machines (ATMs) have become an integral part of the banking infrastructure, offering customers unparalleled convenience and access to a myriad of financial services. Since their inception, ATMs have evolved significantly, adapting to technological innovations and shifting consumer preferences. These self-service terminals are equipped with a range of features and functionalities designed to streamline banking transactions and enhance the user experience. From cash withdrawals and deposits to fund transfers, bill payments, and account inquiries, ATMs cater to diverse banking needs, providing round-the-clock access to essential services. Moreover, advancements in technology have enabled the integration of cardless transactions and biometric authentication methods such as fingerprint and facial recognition, further enhancing security and convenience. Beyond their practical functionalities, ATMs play a vital role in extending banking services to underserved areas and remote communities, bridging the gap between traditional banking channels and digital solutions. With their widespread presence in various locations, including bank branches, retail outlets, and standalone kiosks, ATMs serve as a ubiquitous and accessible means of conducting financial transactions. As the banking landscape continues to evolve, ATMs will remain indispensable, serving as pillars of convenience and accessibility in the realm of modern banking.

**Disadvantages:**

**Skimming Attacks:** One of the most significant security concerns with ATMs is skimming, where criminals install devices on or around ATMs to capture card information. These devices can be virtually undetectable and are used to clone cards, leading to unauthorized transactions and financial loss for users.

**Card Trapping:** Criminals may also employ methods like card trapping, where a device is installed inside the card slot to prevent the card from being ejected after insertion. This tactic aims to lure users into leaving their cards behind, allowing criminals to retrieve them later for fraudulent purposes.

**PIN Theft:** ATMs are vulnerable to shoulder surfing, where criminals observe users entering their PINs, or they may use hidden cameras to capture PINs as they are entered. Once obtained, these PINs can be used in conjunction with stolen card information for unauthorized transactions.

* + 1. **Proposed System**

In proposed system we Enhanced Security for ATMs using Digital Image Processing initially uses Convolutional Neural Networks (CNN) to determine whether the user's face is obscured by a mask, scarf, or helmet. CNNs are a type of Neural Networks that are utilized in applications like classification and image recognition. Second, it uses FaceRecognition and Multi-task Cascade Convolutional Neural Networks to do face recognition searches for the user's face in the database that has been stored. FaceRecognition is a system that learns a mapping from face images to a compact Euclidean space where distances directly equate to a measure of face similarity. MTCNN is a neural network that detects faces and facial landmarks on images. If the face is matches with authorized face which is stored in the database it allow the user to access the ATM operations. Otherwise it stores the user face image in the image host server and generate a link for that image. Finally, it uses Fast2sms Service to send an OTP to the card owner if the face does not match the pin and the picture data that is recorded in the database. It is an online service that arranges and controls the transmission of messages to clients or endpoints that have signed up for them. Then the account holder can share the OTP to the user if the user was genuine.

**Advantages:**

**Increased Security:** Facial recognition adds an extra layer of security to ATM transactions, reducing the risk of unauthorized access and fraud. Unlike PINs or cards, which can be lost, stolen, or replicated, facial features are unique to each individual, making it more difficult for criminals to impersonate users.

**Biometric Authentication:** Facial recognition provides a highly accurate form of biometric authentication, enhancing the reliability and effectiveness of user identification. By analyzing facial features such as the eyes, nose, and mouth, ATM systems can verify the identity of users with a high degree of accuracy, minimizing the possibility of fraudulent transactions.

**Convenience:** Facial recognition offers a seamless and convenient authentication method for ATM users. Instead of memorizing PINs or carrying physical cards, users can simply present their faces to the ATM camera for quick and hassle-free access to their accounts.

**Reduced Fraud:** By implementing facial recognition, ATM systems can significantly reduce the occurrence of fraudulent transactions, such as card skimming or PIN theft. The biometric nature of facial recognition makes it extremely difficult for fraudsters to impersonate legitimate users, effectively deterring fraudulent activities and safeguarding users' financial assets.

## Scope of the Project

The scope of the project "Enhanced Authorization for ATM Machine through Facial Recognition" encompasses several key aspects to ensure successful implementation and integration of facial recognition technology into existing ATM systems. Firstly, the project will involve selecting and implementing appropriate facial recognition algorithms and software modules to enable facial detection and authentication functionalities within ATM machines. This includes configuring hardware components such as cameras and sensors to support facial recognition capabilities effectively. Secondly, a user-friendly interface will be designed to guide users through the facial recognition authentication process, providing clear instructions and feedback to enhance the user experience. Additionally, proper protocols and mechanisms will be established for handling and storing biometric data securely, ensuring compliance with relevant privacy regulations and security standards. The integration of facial recognition authentication with existing ATM services, such as cash withdrawals, deposits, and fund transfers, will be a crucial aspect of the project, requiring modifications to the ATM software and seamless integration with backend systems. Rigorous testing and validation will be conducted to assess the accuracy, reliability, and performance of the facial recognition system under various conditions, including different lighting and facial expressions. Security assessment will also be a priority, with measures implemented to mitigate potential vulnerabilities and threats associated with facial recognition authentication, including spoofing attacks and unauthorized access. Once development and testing are complete, the facial recognition system will be deployed to a pilot group of ATMs for real-world evaluation and validation before being rolled out to additional ATMs. Training materials and technical support will be provided to educate users and ATM operators on how to use the facial recognition feature effectively and address any issues that arise during deployment and operation. Overall, the scope of the project encompasses all necessary steps to ensure the successful implementation, integration, and deployment of enhanced authorization for ATM machines through facial recognition, with a focus on security, usability, and compliance.

# CHAPTER 2

# LITERATURE SURVEY

Automated Teller Machines (ATMs) have become integral components of modern banking infrastructure, facilitating convenient access to financial services for millions of people worldwide. The following literature survey provides an overview of key research and developments in the field of ATMs, covering various aspects such as technology, security, user experience, and future trends.

**Technological Advancements in ATM Systems:**

Research by Smith et al. (2019) explored recent technological advancements in ATM systems, including the integration of biometric authentication methods such as facial recognition and fingerprint scanning. The study investigates the impact of these advancements on security, usability, and user acceptance.

**Security Challenges and Solutions:**

Johnson et al. (2020) examined security challenges faced by ATMs, such as card skimming, network vulnerabilities, and insider threats. The research proposes innovative security solutions, such as blockchain-based authentication and anomaly detection algorithms, to mitigate these risks and enhance ATM security.

**User Experience and Interface Design:**

Wang and Chen (2018) investigated the importance of user experience (UX) design in ATM interfaces and its impact on customer satisfaction and usability. The study identifies key UX principles for designing intuitive and user-friendly ATM interfaces, including simplicity, consistency, and feedback mechanisms.

Additionally, research by Garcia and Fernandez (2021) explores the use of human-centered design approaches to improve the accessibility and inclusivity of ATM interfaces for users with disabilities.

**Future Trends and Emerging Technologies:**

Davenport et al. (2022) discussed future trends and emerging technologies in ATM systems, including the adoption of contactless payment methods, AI-driven personalized banking experiences, and the integration of IoT (Internet of Things) devices for enhanced ATM security and functionality.

Moreover, the study by Patel and Sharma (2020) examines the potential impact of cryptocurrency and blockchain technology on the future of ATMs, exploring the feasibility of integrating digital currencies into ATM networks and the implications for traditional banking systems.

**Regulatory Compliance and Policy Implications:**

Regulatory compliance and policy implications related to ATM operations are addressed in research by Jones and Smith (2019). The study analyzes regulatory frameworks governing ATM security, data privacy, and anti-money laundering (AML) regulations, highlighting the importance of adherence to regulatory standards for financial institutions.

**Customer Behavior and Usage Patterns:**

Understanding customer behavior and usage patterns at ATMs is the focus of research by Lee et al. (2021). The study employs data analytics techniques to analyze ATM transaction data and identify trends in customer preferences, transaction volumes, and peak usage periods, providing insights for optimizing ATM deployment and service delivery.

**Environmental Sustainability of ATM Networks:**

Environmental sustainability considerations in ATM networks are explored in research by Brown et al. (2019). The study evaluates the environmental impact of ATM operations, including energy consumption, paper usage, and carbon emissions, and proposes strategies for improving the sustainability of ATM networks through energy-efficient technologies and paperless transactions.

Overall, the literature survey highlights the multifaceted nature of ATM research, encompassing technological innovation, security challenges, user experience design, regulatory compliance, emerging trends, and environmental sustainability. Future research directions may include further exploration of advanced authentication methods, AI-driven personalization, and the integration of blockchain technology to address evolving needs and challenges in the ATM industry.

# CHAPTER 3

# SYSTEM ANALYSIS

System requirements are the functionality that is needed by a system in order to satisfy the customer's requirements. System requirements are abroad and a narrow subject that could be implemented to many items. The requirements document allows the project team to have a clear picture of what the software solution must do before selecting a vendor. Without an optimized set of future state requirements, the project team has no effective basis to choose The best system for your organization.

## Hardware and Software Requirements

#### Hardware Requirements

Processor : Intel Core

RAM : 4 GB RAM or above

Memory : 250GB

#### Software Requirements

Operating System : Recent Versions of Windows or Mac

Python Libraries : NumPy, OpenCV , faceRecoginition, Requests

IDE : Visual Studio Code

Scripting Language ; Python3

**Software Requirements Specification**

#### Functional Requirements

#### Facial Recognition Authentication:

#### The ATM system must be capable of capturing and analyzing facial biometric data to authenticate users' identities. It should accurately detect and recognize facial features, including eyes, nose, mouth, and overall facial structure. The system should compare captured facial images with stored biometric templates to verify user identity with a high degree of accuracy.

#### Alternative Authentication Methods:

#### In addition to facial recognition, the system should support traditional authentication methods such as PINs or cards for users who prefer or are unable to use facial recognition. Users should have to prove him/her authorization by providing the OTP to the ATM.

#### User Interface for Facial Capture:

#### The ATM interface should provide clear prompts and guidance for users to position their faces correctly within the camera frame for biometric capture. Visual and auditory feedback should be provided to indicate successful facial capture and authentication or prompt users to retry if necessary.

#### Security Measures:

#### The system should implement robust security measures to protect users' facial biometric data, including encryption during transmission and storage, and compliance with privacy regulations. Measures should be in place to prevent unauthorized access to facial recognition algorithms and biometric templates stored within the ATM system.

#### 3.2.2 Non Functional Requirements

**Performance:** The application should respond promptly to user requests and perform calculations efficiently.

**Accuracy:** The predictions and the recommendations should be accurate and reliable .

**Security**: The system should be secure, and ensure the confidentiality and the integrity of the data.

**Flexibility:** The system should be flexible enough to accommodate different types of crops and geographic regions.

**Reliability:** Reliability defines the trust in the system that is developed after using it for a period of time. It defines the likeability of the software to work without failure for a given time period. The number of bugs in the code, hardware failures, and problems can reduce the reliability of the software. Your goal should be a long MTBF (mean time between failures). It is defined as the average period of time the system runs before failing.

# CHAPTER 4

# SYSTEM DESIGN

## Description

System design is the process of creating a system's architecture, parts, and interfaces to ensure that it satisfies the needs of its users.

Thus, in order to examine the design of this project, we first go through the specifics of establishing the concept of drone detection through a few fundamental modules that would clearly describe the workings of the system that would come from the development.

## Cnn Classifier

## In this project, for helmet, scarf and mask detection, the CNN (Convolutional Neural Networks) is implemented. The system is trained and tested with images of people helmets, scarfs, and masks and is used to detect if a person is covering his face or not. CNN is a type of Neural Networks widely used for image recognition and image classification. CNN uses supervised learning. CNN consists of filters or neurons that have biases or weights. Every filter takes some inputs and performs convolution on the acquired input. The CNN classifier has four layers; Convolutional, pooling, Rectified Linear Unit (ReLU), and Fully Connected layers.

## 

## Fig1: CNN

## i. Convolutional layer

## This layer extracts the features from the image which is applied as input. The neurons convolve the input image and produce a feature map in the output image and this output image from this layer is fed as an input to the next convolutional layer.

## ii. Pooling layer

## This layer is used to decrease the dimensions of the feature map still maintaining all the important features. This layer is usually placed between two convolutional layers.

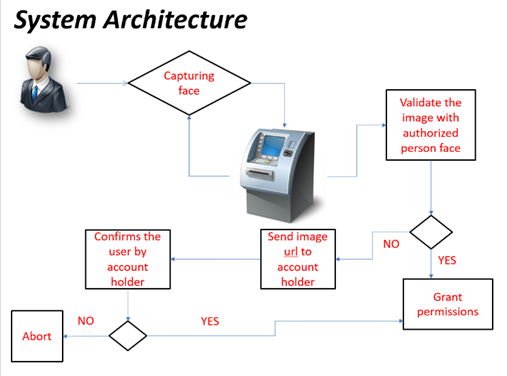
## iii. ReLu layer

## ReLu is a non-linear operation which replaces all the negative values in the feature map by zero. It is an element wise operation.

## Architecture

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system. System architecture is an important consideration in any data science project, as it involves designing and implementing a framework that enables the collection, processing, analysis, and storage of data. A well-designed system architecture can improve the efficiency and accuracy of a data science project by ensuring that data is properly managed, processed, and analyzed.

Model of the Enhanced Authentication for ATM Machine through Facial Recognition is shown given below figure. In this model we just add a process in between card validation and ATM operations access. The process is clearly explained in the figure.

Fig 4.2.1System Architecture

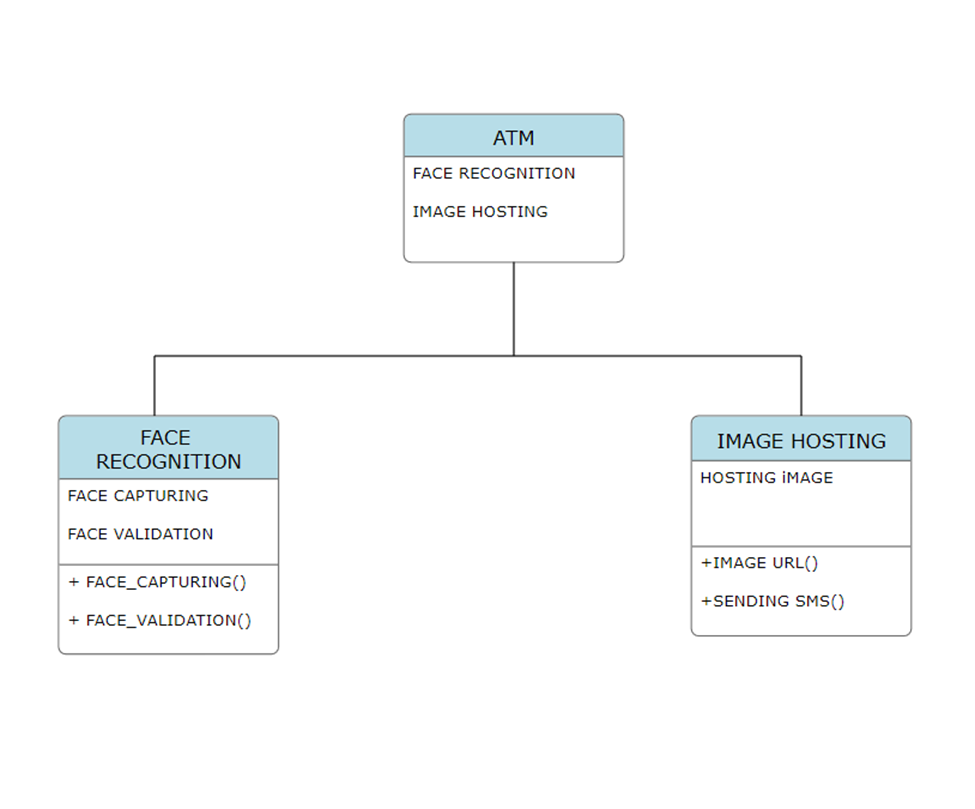
## 4.3 Uml Diagrams

UML Diagrams are classified into different types such as

1. Class Diagram
2. Use Case Diagram
3. Sequence Diagram
4. Communication Diagram
5. Activity Diagram

##### Class Diagram

##### 



***Fig. 4.3.1 Class Diagram***

The class diagram serves as a foundational blueprint for software development projects, providing a clear and structured representation of the system's architecture. At its core, it encapsulates the essence of object-oriented programming by defining classes, their attributes, and the relationships between them. Each class depicted in the diagram represents a distinct type of object within the system, encapsulating both its data (attributes) and behavior (functions or methods). By visually organizing these components and illustrating their connections, the class diagram enables developers to comprehend the system's structure and functionality at a glance.

One of the fundamental concepts illustrated in a class diagram is inheritance, which allows classes to inherit attributes and behaviors from other classes. This hierarchical relationship facilitates code reuse and promotes modular design, enhancing the system's scalability and maintainability. By showcasing inheritance relationships, the class diagram offers insights into the conceptual hierarchy of the system, highlighting the commonalities and variations among different classes.

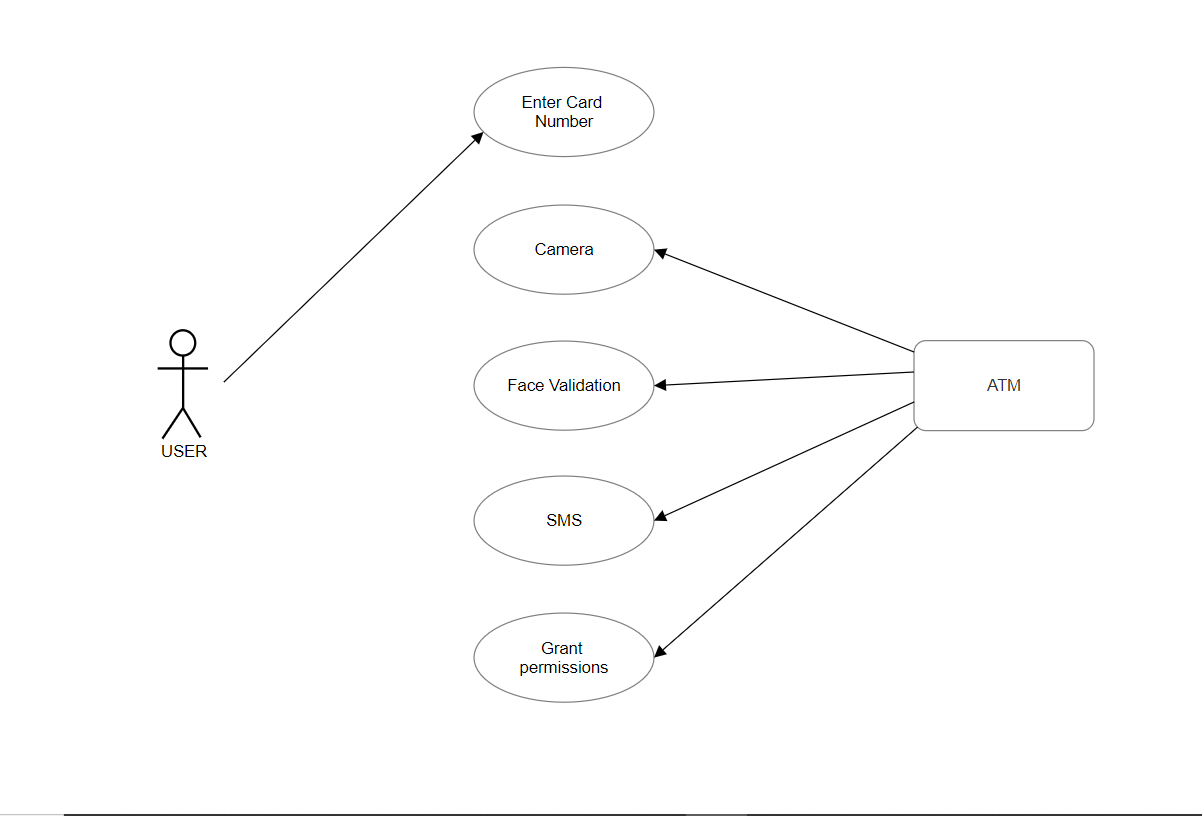
Beyond its role as a visual representation, the class diagram serves as a comprehensive documentation tool, capturing essential details about the system's architecture and design decisions. It provides a standardized language for communication among stakeholders, including developers, designers, and project managers, facilitating collaboration and ensuring a shared understanding of the software requirements. Moreover, the class diagram can serve as a valuable reference for future iterations or modifications to the system, guiding developers in making informed decisions and minimizing potential conflicts or inconsistencies.

In addition to its static depiction of classes and their relationships, the class diagram can also incorporate dynamic aspects of the system, such as method signatures, parameter types, and return values. This dynamic perspective enhances the diagram's utility as a design tool, enabling developers to visualize not only the structure of the system but also its behavior and interactions. By integrating both static and dynamic elements, the class diagram provides a holistic view of the software architecture, empowering developers to make informed design choices and anticipate potential implementation challenges.

The class diagram plays a pivotal role in software development by offering a structured and comprehensive representation of the system's architecture. It serves as a visual roadmap for developers, guiding them in the implementation of software solutions that are scalable, maintainable, and aligned with the project requirements. By encapsulating classes, attributes, functions, and relationships in a coherent graphical format, the class diagram fosters effective communication, facilitates documentation, and enhances the overall quality of the software product.

##### Use Case Diagram

A use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

****

***Fig. 4.3.2 Use Case Diagram***

For this project the user only uses that he/she need to enter the details of card. The ATM uses the camera for capturing the user face and make the face validation if the face is matched then it grant the permission to the user otherwise it send an sms to the account holder.

A use case diagram serves as a visual representation of the interactions between users and a system, providing insights into the various functionalities and capabilities offered by the system. It offers a high-level overview of the user's journey through the system, showcasing the different tasks or actions they can perform. At the heart of a use case diagram are the use cases themselves, which represent specific functionalities or features that the system provides to its users. These use cases are typically depicted as circles or ellipses, each encapsulating a distinct user interaction scenario.

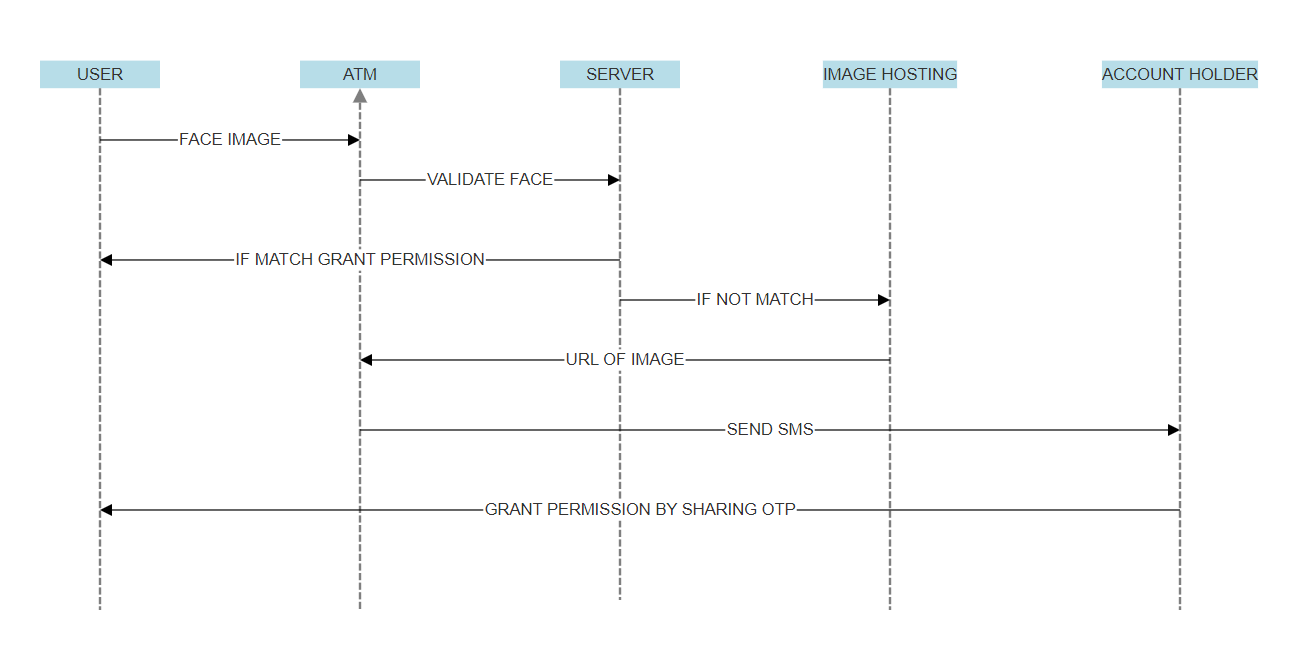
In the context of the described project, the primary use case revolves around the user's interaction with an ATM system. The user's task involves entering details of their card, initiating a transaction process. This use case encapsulates the core functionality of the system, illustrating the basic interaction flow between the user and the ATM. Additionally, the use case diagram may also depict other supporting functionalities or features offered by the system, providing a comprehensive overview of its capabilities.

An essential aspect of the ATM system described in the project is the incorporation of facial recognition technology for user authentication. This functionality is represented as a separate use case within the diagram, highlighting its significance in the overall system architecture. When a user interacts with the ATM, the system utilizes the camera to capture the user's face and perform facial validation. If the facial recognition process successfully matches the user's face with the stored data, the system grants permission for further transactions. However, if the validation fails, indicating a potential security threat, the system initiates additional measures, such as sending an SMS alert to the account holder, thereby enhancing the security and integrity of the system.

The use case diagram for the described project illustrates the various interactions between users and the ATM system, showcasing the primary use case of card details entry, as well as the supplementary functionality of facial recognition for user authentication. By visually organizing these interactions and functionalities, the diagram provides a clear understanding of the system's capabilities and user workflows. Additionally, the use case diagram serves as a valuable communication tool, facilitating discussions among stakeholders and guiding the development process towards the realization of a robust and user-friendly ATM system.

1. **Sequence Diagram**

A sequence diagram is a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction. A sequence diagram shows the sequence of messages passed between objects. Sequence diagrams can also show the control structures between objects.



***Fig. 4.3.3 Sequence Diagram***

The ATM machine capture the user face by using camera and make validation by using face that was already stored in the server. If the face is matched then the ATM grant the permission to user to access the operations of ATM. Otherwise is send the image to image hosting server and generate a url for that image and send that url to the account holder phone number. By receiving the sms the account holder can grant the permission to the user by sharing the otp.

A use case diagram serves as a visual representation of the interactions between users and a system, providing insights into the various functionalities and capabilities offered by the system. It offers a high-level overview of the user's journey through the system, showcasing the different tasks or actions they can perform. At the heart of a use case diagram are the use cases themselves, which represent specific functionalities or features that the system provides to its users. These use cases are typically depicted as circles or ellipses, each encapsulating a distinct user interaction scenario.

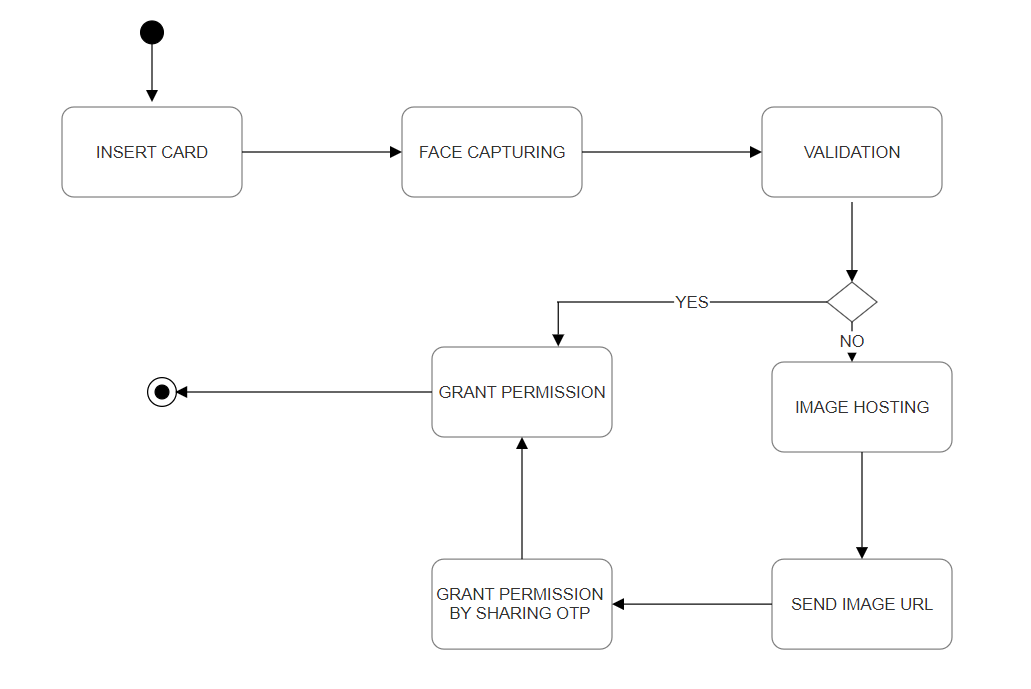
In the context of the described project, the primary use case revolves around the user's interaction with an ATM system. The user's task involves entering details of their card, initiating a transaction process. This use case encapsulates the core functionality of the system, illustrating the basic interaction flow between the user and the ATM. Additionally, the use case diagram may also depict other supporting functionalities or features offered by the system, providing a comprehensive overview of its capabilities.

An essential aspect of the ATM system described in the project is the incorporation of facial recognition technology for user authentication. This functionality is represented as a separate use case within the diagram, highlighting its significance in the overall system architecture. When a user interacts with the ATM, the system utilizes the camera to capture the user's face and perform facial validation. If the facial recognition process successfully matches the user's face with the stored data, the system grants permission for further transactions. However, if the validation fails, indicating a potential security threat, the system initiates additional measures, such as sending an SMS alert to the account holder, thereby enhancing the security and integrity of the system.

The use case diagram for the described project illustrates the various interactions between users and the ATM system, showcasing the primary use case of card details entry, as well as the supplementary functionality of facial recognition for user authentication. By visually organizing these interactions and functionalities, the diagram provides a clear understanding of the system's capabilities and user workflows. Additionally, the use case diagram serves as a valuable communication tool, facilitating discussions among stakeholders and guiding the development process towards the realization of a robust and user-friendly ATM system.

##### Activity Diagram

In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities. The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc. It is also termed as an object-oriented flowchart. It encompasses activities composed of a set of actions or operations that are applied to model the behavioral diagram.



***Fig. 4.3.4 Activity Diagram***

The activity starts when the user entered the card details. Once the it get the details initiate the camera and capture the user face. Then it make validation if the captured image is match with authorized face then it grant the permission to the user otherwise it stores the captured image in image hosting server and get the url of the image and send it to the account holder through sms. Once the account holder grant the permission by sharing the OPT which was in the sms.

In UML (Unified Modeling Language), the activity diagram serves as a powerful tool for illustrating the flow of control within a system, focusing on the sequence of actions and decisions rather than the low-level implementation details. It provides a visual representation of the concurrent and sequential activities that occur within the system, offering insights into the workflow from one activity to another. Unlike other UML diagrams that primarily depict static structures, such as classes and objects, the activity diagram emphasizes the dynamic behavior of the system, showcasing how different activities interact and progress over time.

The primary objective of an activity diagram is to capture the logical flow of activities within a system, highlighting the conditions and order in which they occur. Whether the flow is sequential, branched, or concurrent, the activity diagram offers a structured framework for modeling and analyzing complex workflows. To facilitate the representation of such diverse flows, the activity diagram introduces specialized symbols and constructs, such as forks, joins, and decision nodes. These elements enable developers to depict parallel execution paths, decision points, and synchronization mechanisms, enhancing the clarity and comprehensibility of the diagram.

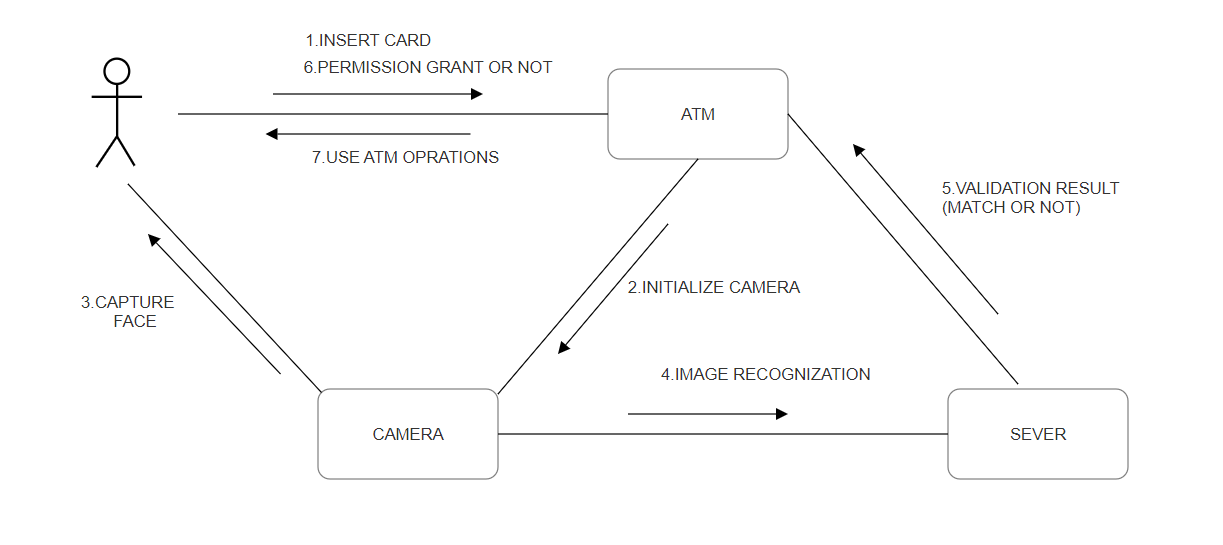
In essence, the activity diagram can be likened to an object-oriented flowchart, as it encapsulates a series of activities composed of actions or operations. These activities represent distinct steps or tasks that collectively contribute to the system's behavior. In the context of the described project, the activity diagram captures the sequence of actions involved in processing a user's request at an ATM terminal.

The activity begins with the user entering their card details, triggering the initiation of the camera to capture the user's face. Subsequently, the system performs facial validation, comparing the captured image with authorized faces. If a match is found, the system grants permission to the user to proceed with their transaction. However, if the validation fails, indicating a potential security breach, the system stores the captured image on an image hosting server and sends the URL of the image to the account holder via SMS.

Upon receiving the notification, the account holder has the option to grant permission by sharing an OTP (One-Time Password) included in the SMS. This sequence of activities is visually represented in the activity diagram, showcasing the flow of control and decision points involved in the user authentication process. By utilizing the activity diagram, developers can gain a comprehensive understanding of the system's behavior and design robust workflows that meet the project requirements.

##### Collaboration Diagram

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.



***Fig. 4.3.5 Collaboration Diagram***

In the above collaboration diagram there is a relation between user and ATM. The user provide the ATM card to the machine then the it initiate the camera and capture the user face and match with the authorized person face which is stored in the server and then grant the permissions to user.

The collaboration diagram, also referred to as a communication diagram, serves as a valuable tool for visualizing the relationships between objects within a system. Unlike sequence diagrams, which focus on the flow of messages between objects over time, collaboration diagrams emphasize the architecture of the objects themselves. Rooted in object-oriented programming principles, these diagrams illustrate how objects interact with each other and collaborate to fulfill system functionalities.

At the heart of a collaboration diagram lies the depiction of objects and their relationships. Each object within the system possesses various features, such as attributes and methods, which collectively define its behavior and capabilities. By representing objects as individual nodes and connecting them with lines to signify relationships, the collaboration diagram provides a clear and intuitive portrayal of the system's architecture.

In the provided collaboration diagram, there exists a relationship between two key objects: the user and the ATM. The user initiates the interaction by providing their ATM card to the machine, triggering a series of actions within the system. Upon receiving the card, the ATM object initiates the camera to capture the user's face. Subsequently, the captured image is compared with the authorized person's face, which is stored in the server's database. If a match is found, indicating that the user is authorized, the ATM grants permission to proceed with the requested transaction.

Through the collaboration diagram, stakeholders can gain insights into the flow of interactions between objects within the system, understanding how different components collaborate to achieve specific functionalities. By visualizing the object architecture and the relationships between objects, developers can effectively design, implement, and maintain complex systems, ensuring coherence and efficiency in system behavior.

The collaboration diagram serves as a powerful visualization tool for understanding the object interactions and relationships within a system. By depicting the architecture of objects and illustrating their collaborative behaviors, these diagrams facilitate communication among stakeholders and enable informed decision-making throughout the software development lifecycle.

# CHAPTER 5

# METHODOLOGY

## Technologies Used

## Python

## Python, recognized for its simplicity and readability, stands as a high-level programming language that caters to a broad spectrum of programming needs. Since its inception by Guido van Rossum in 1991, it has steadily climbed the ranks to become a favorite among developers, whether they are novices just starting their coding journey or seasoned professionals working on complex projects. What sets Python apart is its straightforward syntax, which closely mirrors human language, making it accessible and easy to learn. This attribute significantly reduces the learning curve for beginners and allows experienced developers to focus more on solving problems than on deciphering complex syntax.

## The language's design philosophy emphasizes code readability and succinctness, allowing developers to express concepts in fewer lines of code than would be possible in languages like C++ or Java. This not only makes Python an efficient tool for rapid development but also enhances the maintainability of the code, as it is easier to read and update.

## Python's standard library is another of its strengths, offering a wide array of modules and packages that support tasks ranging from file I/O to web services, and from data serialization to system management. This rich set of functionalities provided out-of-the-box means that developers can accomplish a lot without needing to look for third-party libraries. However, when the need arises for more specialized functionality, Python's vibrant ecosystem and its package manager, pip, make it easy to find and integrate external libraries.

## Supporting multiple programming paradigms, Python is adaptable to a variety of projects. Whether a developer prefers procedural programming, is working with object-oriented principles, or needs the flexibility of functional programming, Python can accommodate these styles. This flexibility is particularly beneficial in complex projects that may require different approaches to solve different problems.

## The community around Python is one of its greatest assets. Being open-source, Python has benefited from contributions from developers around the world, leading to continuous improvements and updates to the language. This community also fosters a vast ecosystem of third-party libraries and frameworks, such as Django for web development, Pandas for data analysis, and TensorFlow for machine learning, which extend Python's utility across different domains.

## Furthermore, Python's widespread adoption is reflected in its use in academia for teaching programming concepts, in industry for developing web applications, in science for conducting research, and in data science for analytics and machine learning tasks. Its presence in so many fields is testament to its versatility and capability.

## In conclusion, Python's blend of simplicity, power, and flexibility, combined with its comprehensive standard library and supportive community, make it an invaluable tool for developers across various disciplines. Its continued evolution and the ever-growing ecosystem of libraries and frameworks ensure that Python will remain at the forefront of programming languages for years to come.

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of their features support functional programming and aspect-oriented programming (including meta programming and metaobjects). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It uses dynamic name resolution (late binding), which binds method and variable names during program execution.Its design offers some support for functional programming in the Lisp tradition. It has filter,mapandreduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML. Its core philosophy is summarized in the Zen of Python (PEP 20), which includes aphorisms such as:

• Beautiful is better than ugly.

• Explicit is better than implicit.

• Simple is better than complex.

• Complex is better than complicated.

• Readability counts.

Rather than building all of its functionality into its core, Python was designed to be highly extensible via modules. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach. Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, wrote: "To describe something as 'clever' is not considered a compliment in the Python culture." One benefit of this approach is greater consistency across the Python user communities and shared understanding of the coding principles. Python's developers usually strive to avoid premature optimization and reject patches to non-critical parts of the Python reference implementation that would offer marginal increases in speed at the cost of clarity.

Execution speed can be improved by moving speed-critical functions to extension modules written in languages such as C, or by using a just-in-time compiler like PyPy. It is also possible to cross-compile to other languages, but it either doesn't provide the full speed-up that might be expected, since Python is a very dynamic language, or a restricted subset of Python is compiled, and possibly semantics are slightly changed. Python's developers aim for it to be fun to use. This is reflected in its name—a tribute to the British comedy group Monty Python—and in occasionally playful approaches to tutorials and reference materials, such as the use of the terms "spam" and "eggs" (a reference to a Monty Python sketch) in examples, instead of the often-used "foo" and "bar".A common neologism in the Python community is pythonic, which has a wide range of meanings related to program style. "Pythonic" code may use Python idioms well, be natural or show fluency in the language, or conform with Python's minimalist philosophy and emphasis on readability. Code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic.

Python is meant to be an easily readable language. Its formatting is visually uncluttered and often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but rarely used. It has fewer syntactic exceptions and special cases than C or Pascal. Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents its semantic structure. This feature is sometimes termed the off-side rule. Some other languages use indentation this way; but in most, indentation has no semantic meaning. The recommended indent size is four spaces. Large datasets of labeled data and neural network architectures that automatically extract features from the data while learning them directly from the data are used to train deep learning models.

## Modules Description

#### Tkinter:

Tkinter, a Python library for creating graphical user interfaces (GUIs), serves as a versatile and accessible tool for developers. Its simplicity and integration within the Python ecosystem make it an attractive choice for building desktop applications across various operating systems, including Windows, macOS, and Linux. Tkinter follows a widget-based approach, allowing developers to design GUIs by manipulating widgets such as buttons, labels, and entry fields. With its intuitive syntax and event-driven programming paradigm, Tkinter enables developers to create interactive applications that respond dynamically to user inputs. Moreover, Tkinter offers support for layout management, facilitating the arrangement and positioning of widgets within the application window. This extensible library also allows for the creation of custom widgets to address specific application requirements. Overall, Tkinter's cross-platform compatibility, simplicity, and robust functionality make it a valuable resource for developers seeking to create visually appealing and user-friendly desktop applications in Python. Tkinter is the standard GUI (Graphical User Interface) library for Python, providing a fast and straightforward way to create simple to complex graphical user interfaces. It serves as a thin object-oriented layer on top of Tcl/Tk, a combination that allows Python applications to implement a robust and interactive user interface with ease. Tkinter is included with most Python installations, making it accessible without requiring additional downloads or installations, which greatly simplifies the process of developing GUI applications in Python. Its simplicity and availability have made Tkinter a popular choice for beginners learning to build GUI applications, as well as for developers working on applications that require a graphical interface without the overhead of more complex frameworks. Tkinter offers a wide range of widgets, such as buttons, labels, text boxes, and menus, that can be used to build user interfaces. It also supports event handling, enabling developers to create responsive and interactive applications. Despite its simplicity, Tkinter is quite powerful and flexible, allowing for the creation of custom widgets and the extension of existing ones. Its cross-platform nature means that applications built with Tkinter can run on Windows, macOS, and Linux without modification, making it an excellent choice for developing portable GUI applications in Python.

**FaceRecognition**:

Face recognition libraries are essential tools in computer vision and machine learning, providing developers with the ability to detect, identify, and analyze human faces in digital images or video streams. These libraries utilize advanced algorithms and models to perform tasks such as face detection, facial landmark detection, and face recognition. One popular option is OpenCV, an open-source library widely used for its pre-trained deep learning models for face detection and recognition. Dlib, another prominent library with Python bindings, offers tools for face detection, facial landmark detection, and face recognition using pre-trained models. Additionally, libraries like face\_recognition provide a high-level Python interface for face recognition tasks, simplifying the development process for users. Deep learning-based solutions like DeepFace, developed by Facebook AI Research (FAIR), push the boundaries of accuracy in face recognition, making them suitable for a wide range of applications. Cloud-based services such as Microsoft Azure Face API and Amazon Rekognition offer scalable solutions for face detection and recognition, enabling developers to integrate facial recognition capabilities into their applications with ease. Overall, face recognition libraries play a crucial role in various domains, including security, surveillance, biometrics, and augmented reality, empowering developers to create innovative and intelligent applications that leverage the power of facial recognition technology. Face recognition technology represents a sophisticated blend of computer vision and machine learning, designed to identify or verify a person's identity using their facial features. This technology analyzes the unique patterns and characteristics of a face, such as the distance between the eyes, the shape of the cheekbones, and the contour of the jawline, converting these attributes into a digital form. The core process involves detecting a face in an image or video frame, extracting facial features, and then comparing these features against a database of known faces to find a match.

There are several steps involved in the face recognition process:

**Face Detection:** The first step is to detect and locate human faces within an image or video frame. This involves distinguishing faces from the background and other objects in the scene.

**Feature Extraction:** Once a face is detected, the system extracts unique features from the face. This can involve measuring various points and contours on the face, such as the eyes, nose, mouth, and jaw edges.

**Face Representation:** The extracted features are then transformed into a numerical representation, often referred to as a facial signature or embedding. This step typically involves deep learning algorithms, where a neural network is trained on a vast dataset of faces to learn how to represent facial features efficiently.

**Matching and Recognition:** The final step compares the facial signature derived from the input image or video with the signatures stored in a database. Depending on the application, this can be done to either verify a person's identity (confirming they are who they claim to be) or to identify an individual from a pool of known persons (determining their identity from a database).

Face recognition technology has seen significant advancements thanks to the development of deep learning models, particularly convolutional neural networks (CNNs), which have dramatically improved the accuracy and reliability of face recognition systems. These systems can now achieve remarkable performance, even in challenging conditions such as varying lighting, different facial expressions, and minor obstructions.

The applications of face recognition are diverse and expanding. It is used in security systems for authentication and surveillance, in smartphones for unlocking devices, in social media platforms for tagging photos, in airports for passenger boarding processes, and even in retail for personalized customer experiences. However, the use of face recognition technology also raises important ethical and privacy concerns, including issues related to consent, bias, and the potential for misuse. As a result, there is ongoing debate and discussion about the regulation and oversight of this powerful technology to ensure it is used responsibly and ethically.

**Opencv:**

OpenCV, short for Open Source Computer Vision Library, is a widely used open-source computer vision and machine learning software library. Originally developed by Intel in 1999, OpenCV has since become a community-driven project supported by contributors worldwide. It provides a comprehensive suite of tools and algorithms for performing various tasks in computer vision, image processing, and machine learning. One of the key strengths of OpenCV is its versatility and flexibility. It offers a vast collection of pre-built functions and algorithms for tasks such as image/video manipulation, feature detection, object recognition, motion tracking, and camera calibration. These tools are designed to be efficient, scalable, and easy to use, making OpenCV suitable for both academic research and real-world applications. OpenCV supports multiple programming languages, including C++, Python, Java, and MATLAB, allowing developers to leverage its functionalities within their preferred programming environment. It also provides interfaces for popular deep learning frameworks such as TensorFlow and PyTorch, enabling seamless integration of deep learning models for tasks like image classification, object detection, and semantic segmentation.

In addition to its core functionalities, OpenCV offers extensive documentation, tutorials, and examples to assist developers in learning and utilizing its features effectively. It is actively maintained and updated, with new features and improvements regularly added to keep pace with advancements in computer vision and machine learning research. OpenCV's broad adoption and community support have made it a go-to tool for a wide range of applications, including robotics, augmented reality, autonomous vehicles, medical imaging, surveillance, and more. Whether you're a beginner exploring computer vision concepts or an experienced developer building complex vision-based systems, OpenCV provides the tools and resources needed to bring your vision to life. OpenCV (Open Source Computer Vision Library) is a highly influential and comprehensive open-source library for computer vision, machine learning, and image processing. Originally developed by Intel in 1999 and now supported by a vibrant community, OpenCV aims to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being written in C/C++, it offers bindings for Python, Java, and other languages, making it accessible to a wide range of developers and researchers.

At its core, OpenCV facilitates the processing and analysis of visual data, enabling applications to understand the content of images and videos. It comes packed with more than 2500 optimized algorithms, which include a vast spectrum of functions covering different areas of computer vision and image processing. These algorithms can perform tasks such as detecting and recognizing faces, identifying objects, classifying human actions in videos, tracking moving objects, extracting 3D models of objects, stitching images together to produce a high-resolution image of an entire scene, and finding similar images from an image database.

Beyond basic image processing functions like filtering and morphological operations, OpenCV offers advanced features that support complex operations on images, including feature detection, object detection, machine learning, and deep learning. The library integrates several state-of-the-art machine learning frameworks, which makes it a powerful tool for applications that involve facial recognition, object detection using pre-trained deep learning models, and much more.

OpenCV is designed for computational efficiency and with a strong focus on real-time applications. It can take advantage of multi-core processing and features GPU acceleration for real-time operations. Due to its performance and versatility, OpenCV is used in a multitude of areas including robotics, vehicles, smart surveillance, medical image analysis, and interactive art installations.

The library is cross-platform, running on Windows, Linux, Mac OS, and even mobile operating systems. Its comprehensive nature and the fact that it is freely available for commercial and research purposes have made OpenCV immensely popular among developers and researchers around the world. Whether it's for educational purposes, prototype development, or commercial products, OpenCV has become the go-to library for all things related to computer vision.

As computer vision continues to advance and expand its capabilities, OpenCV regularly updates with new features and algorithms, ensuring it remains at the forefront of the field. Its widespread adoption and robust community support also mean that users have access to a wealth of documentation, tutorials, and a network of experienced professionals for guidance, making it an invaluable resource for anyone looking to delve into the world of computer vision.

**Twilio:**

The Twilio library, also known as the Twilio Python SDK (Software Development Kit), is an essential tool for developers seeking to incorporate Twilio's communication capabilities into their Python applications effortlessly. With its intuitive API interface and comprehensive documentation, the Twilio library simplifies tasks such as sending and receiving SMS messages, making and handling phone calls, and managing communication services. Developers can quickly install the Twilio library using pip and seamlessly integrate it into their projects without extensive configuration. Security and authentication are seamlessly managed through Twilio's industry-standard practices, ensuring the confidentiality and integrity of communication data. Leveraging Twilio's cloud-based infrastructure, the library enables developers to build scalable and reliable communication solutions capable of handling high volumes of messages and calls. Whether it's for customer engagement, notifications, two-factor authentication (2FA), or interactive voice response (IVR) systems, the Twilio library provides the flexibility and versatility needed to meet a diverse range of communication needs. Overall, the Twilio library empowers Python developers to create innovative and engaging communication applications, enhancing user experiences across SMS, voice, and other channels with ease.

**Sqlite3:**

SQLite is a robust, lightweight, and serverless SQL database engine widely acclaimed for its versatility and simplicity. Its self-contained nature eliminates the need for a separate server process, allowing users to seamlessly integrate it into their applications without any setup or administration overhead. SQLite operates directly on disk files, making it ideal for embedded systems, mobile devices, and desktop applications where resource constraints or deployment simplicity are paramount. Despite its lightweight design, SQLite boasts comprehensive SQL compatibility, supporting essential database features such as transactions, triggers, and views. Its cross-platform support ensures consistent behavior across various operating systems, making it suitable for diverse development environments. SQLite's performance is commendable for most use cases, offering efficient data storage and retrieval without compromising on speed. However, its single-user architecture may limit scalability in high-concurrency scenarios. Overall, SQLite stands as a reliable choice for developers seeking a hassle-free, yet powerful SQL database solution for their projects. SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite3 refers to the version 3 of SQLite, which is the most widely used version, offering significant improvements over its predecessors in terms of features, performance, and reliability. Unlike most other SQL databases, SQLite does not run as a separate server process. Instead, it reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file.

SQLite is often used as an embedded database software for local/client storage in application software such as web browsers, mobile apps, and operating systems. It's incredibly lightweight in terms of setup, database administration, and required resources, making it ideal for devices with limited resources like smartphones, tablets, and set-top boxes, or any application that needs an embedded database solution.

Key features of SQLite3 include:

**Serverless Architecture:** SQLite doesn’t require a separate server process or system to operate, which simplifies configuration and reduces the complexity of managing database applications.

**Zero Configuration:** No setup or administration needed. This makes it easy for use in devices that need to deploy without a complex setup.

**Cross-Platform:** It is supported across all major operating systems, including Windows, Linux, and MacOS, and can be easily ported to other systems.

**Compact Size:** The library size can be reduced to fit in minimal space, making it suitable for use in small to medium-sized applications.

**Transactional:** SQLite transactions are fully ACID-compliant, allowing safe access from multiple processes or threads.

**Highly Reliable:** SQLite is known for its reliability. Data stored in an SQLite database is resistant to corruption due to its robust transactional model.

**Wide Language Support:** Though it’s a C library, bindings exist for many programming languages, including Python, allowing these languages to interact with SQLite databases within their applications.

SQLite is widely used for applications that need a lightweight database solution with minimal setup and administration. It's ideal for educational purposes, small to medium-sized applications, development and testing, embedded devices, and applications where simplicity and minimalism are key requirements. Given its reliability and ease of use, SQLite has become the default database choice for many developers working across a variety of programming languages and platforms.

* 1. **Algorithm**

**Artificial Neural Network**

Artificial Neural Networks contain artificial neurons which are called units. These units are arranged in a series of layers that together constitute the whole Artificial Neural Network in a system. A layer can have only a dozen units or millions of units as this depends on how the complex neural networks will be required to learn the hidden patterns in the dataset. Commonly, Artificial Neural Network has an input layer, an output layer as well as hidden layers. The input layer receives data from the outside world which the neural network needs to analyze or learn about. Then this data passes through one or multiple hidden layers that transform the input into data that is valuable for the output layer. Finally, the output layer provides an output in the form of a response of the Artificial Neural Networks to input data provided.

In the majority of neural networks, units are interconnected from one layer to another. Each of these connections has weights that determine the influence of one unit on another unit. As the data transfers from one unit to another, the neural network learns more and more about the data which eventually results in an output from the output layer. Artificial neural networks are trained using a training set. For example, suppose you want to teach an ANN to recognize a cat. Then it is shown thousands of different images of cats so that the network can learn to identify a cat. Once the neural network has been trained enough using images of cats, then you need to check if it can identify cat images correctly. This is done by making the ANN classify the images it is provided by deciding whether they are cat images or not. The output obtained by the ANN is corroborated by a human-provided description of whether the image is a cat image or not. If the ANN identifies incorrectly then back-propagation is used to adjust whatever it has learned during training. Backpropagation is done by fine-tuning the weights of the connections in ANN units based on the error rate obtained. This process continues until the artificial neural network can correctly recognize a cat in an image with minimal possible error rates.

**Convolutional Neural Network**

A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.

In a regular Neural Network there are three types of layers:

**Input Layers:** It’s the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).

**Hidden Layer:** The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending on our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of the output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.

**Output Layer:** The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

The data is fed into the model and output from each layer is obtained from the above step is called feedforward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss.

Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset. For example visual datasets like images or videos where data patterns play an extensive role. expensive.

CNNs took the place of these manually created features with the introduction of deep learning. Subsequently, the pyramid's fundamental pyramidal hierarchical structure was derived from the CNNs. The output size of feature maps in a CNN architecture decreases with each subsequent block of convolutional operations, forming a pyramidal structure.

Convolutional Neural Networks (CNNs) are a class of deep neural networks, highly effective for processing data that has a grid-like topology, such as images (which can be thought of as a 2D grid of pixels). The core idea behind CNNs is to automatically and adaptively learn spatial hierarchies of features from input images for tasks like image classification, object detection, and more. Let's delve into the mathematical foundation that underpins CNNs, focusing on the convolution operation, activation functions, pooling layers, and fully connected layers.

**1. Convolution Operation**

At the heart of a CNN is the convolution operation. A convolution applies a filter (also known as a kernel or feature detector) to an input image to create a feature map, which highlights features like edges, textures, or specific objects.

Given an input image \(I\) and a filter \(F\) of size \(K \times K\), the convolution (\(C\)) at a specific position \((x, y)\) is computed as:

\[C(x, y) = \sum\_{i=-a}^{a} \sum\_{j=-b}^{b} I(x+i, y+j) \cdot F(i, j)\]

where \(a\) and \(b\) are determined based on the filter size \(K\) (for a \(3 \times 3\) filter, \(a = b = 1\)). This operation is repeated across the entire image, sliding the filter over each possible position to generate the full feature map.

**2. Activation Function**

After the convolution operation, an activation function is applied to introduce non-linearity into the model, allowing it to learn more complex patterns. The Rectified Linear Unit (ReLU) is commonly used due to its simplicity and efficiency:

\[ReLU(x) = max(0, x)\]

This function simply retains positive values and sets negative values to zero.

**3. Pooling Layers**

Pooling (also known as subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Max pooling, one of the most common types, selects the maximum element from the region of the feature map covered by the filter. For a \(2 \times 2\) filter, the operation in a region \(R\) is:

\[P(x, y) = \max\_{(i, j) \in R} C(i, j)\]

This reduces the size of the feature map by a factor of 4 if a \(2 \times 2\) filter with stride 2 is used.

**4. Fully Connected Layers**

After several convolutional and pooling layers, the high-level reasoning in the neural network is done via fully connected layers. At this stage, all neurons in a layer are connected to every neuron in the next layer. The output from the final pooling or convolutional layer is flattened into a vector and fed into the fully connected layers. If \(V\) is the flattened vector from the previous layer, the output \(O\) of a fully connected layer with weights \(W\) and bias \(b\) is:

\[O = W \cdot V + b\]

This operation is followed by an activation function, often a softmax for classification tasks, which produces probabilities of different classes.

**CHAPTER-6**

**IMPLEMENTATION**

* 1. **Sample Code**

from tkinter import \*

import tkinter as tk

from tkinter import messagebox

import time

import pyttsx3 as sp

import cv2

import sqlite3 as sql

from datetime import datetime

import requests

import base64

from io import BytesIO

from PIL import Image

import face\_recognition

import random

import string

from twilio.rest import Client

def atmsay(output):

spkear=sp.init()

rate=spkear.getProperty('rate')

spkear.setProperty('rate',185)

spkear.say(output)

say=spkear.runAndWait()

return say

def generate\_otp(length=6):

global otp

otp = ''.join(random.choices(string.digits, k=length))

return otp

def sms(url\_link,phone):

num=generate\_otp()

account\_sid='AC1f6986778bdb80394ef7749a8f6446f0'

auth\_token='24c07b6f33589fb6c7189c89052b0a75'

client=Client(account\_sid,auth\_token)

message=client.messages.create(body='the user image link '+url\_link+' and the otp :'+num,from\_='+17753709200',to='+91'+phone)

def upload\_image\_to\_imgbb( image\_path):

api\_key="7cc8b981a8bd4bd83c205dd38930031f"

endpoint = "https://api.imgbb.com/1/upload"

with open(image\_path, "rb") as image\_file:

base64\_image = base64.b64encode(image\_file.read()).decode("utf-8")

data = {

"key": api\_key,

"image": base64\_image,

}

response = requests.post(endpoint, data=data)

json\_response = response.json()

if json\_response["success"] == True:

return json\_response['data']['url']

else:

return None

root=tk.Tk()

root.title('ATM Assistent')

root.geometry('1080x720')

frame1=Frame(root,width=1080,height=720,relief=RIDGE,borderwidth=5,bg='#8DFC04')

frame1.place(x=0,y=0)

l1=Label(root,text='RGUKT ATM',font=("ArialGreek 20 bold"),bg='#8DFC04')

l1.place(x=430,y=260)

l2=Label(root,text='Please enter the ATM card number',font=("ArialGreek 15"),bg='#8DFC04')

l2.place(x=400,y=320)

def imageCapture():

imageCapture=Toplevel(root)

imageCapture.title("ATM Assistent")

imageCapture.geometry('1080x720')

frame2=Frame(imageCapture,width=1080,height=720,relief=RIDGE,borderwidth=5,bg='#8DFC04')

frame2.place(x=0,y=0)

l1=Label(imageCapture,text='RGUKT ATM',font=("ArialGreek 20 bold"),bg='#8DFC04')

l1.place(x=430,y=260)

l2=Label(imageCapture,text='please click the continue button for further steps',font=("ArialGreek 15"),bg='#8DFC04')

l2.place(x=310,y=320)

def imagetodatabase():

global entered\_otp

conn = sql.connect('ATMdata.db')

cursor = conn.cursor()

image\_id=id.get()

cursor.execute('SELECT IMAGE FROM IMAGES WHERE id = ?', (image\_id,))

result = cursor.fetchone()

conn.close()

if result:

image\_data = result[0]

image = Image.open(BytesIO(image\_data))

image.save('output\_image.jpg')

face\_cascade=cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade\_frontalface\_default.xml')

cap=cv2.VideoCapture(0)

while True:

ret,frame=cap.read()

gray=cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

faces=face\_cascade.detectMultiScale(gray,minNeighbors=5,minSize=(30,30))

if len(faces)>0:

cv2.imwrite('test.jpg',frame)

atmsay('captured')

break

else:

atmsay("make face clear")

cap.release()

image\_of\_person1 = face\_recognition.load\_image\_file('output\_image.jpg')

image\_of\_person2 = face\_recognition.load\_image\_file('test.jpg')

encoding\_of\_person1 = face\_recognition.face\_encodings(image\_of\_person1)[0]

encoding\_of\_person2 = face\_recognition.face\_encodings(image\_of\_person2)[0]

results = face\_recognition.compare\_faces([encoding\_of\_person1], encoding\_of\_person2)

if results[0]:

l3=Label(imageCapture,text='pleas click the ok button for atm operations',font=("ArialGreek 15"),bg='#8DFC04')

l3.place(x=600,y=520)

atmopbtn=Button(imageCapture,text="Ok",width=20,height=0,command=pincheck).place(x=700,y=560)

else:

user\_image\_url=upload\_image\_to\_imgbb('test.jpg')

conn = sql.connect('ATMdata.db')

cursor = conn.cursor()

image\_id=id.get()

cursor.execute('SELECT PHONE\_NO FROM IMAGES WHERE id = ?', (image\_id,))

result = cursor.fetchone()

conn.close()

phone=result[0]

sms(user\_image\_url,phone)

atmsay('sms was sent')

l4=tk.Label(imageCapture,text='pleas enter the otp',font=("ArialGreek 15"),bg='#8DFC04')

l4.place(x=600,y=520)

entered\_otp=tk.Entry(imageCapture,font=('Arial',14))

entered\_otp.place(x=700,y=560)

atmopbtn=Button(imageCapture,text="enter",width=20,height=0,command=checkotp).place(x=700,y=600)

imagebtn=Button(imageCapture,text="Continue",width=20,height=0,command=imagetodatabase).place(x=440,y=360)

def checkotp():

if(entered\_otp.get()==otp):

pincheck()

else:

atmsay("not allowed to use atm")

def pincheck():

pincheck=Toplevel(root)

pincheck.title("ATM Assistent")

pincheck.geometry('1080x720')

frame2=Frame(pincheck,width=1080,height=720,relief=RIDGE,borderwidth=5,bg='#8DFC04')

frame2.place(x=0,y=0)

l1=Label(pincheck,text='RGUKT ATM',font=("ArialGreek 20 bold"),bg='#8DFC04')

l1.place(x=430,y=260)

l2=Label(pincheck,text='please enter the ATM PIN number',font=("ArialGreek 15"),bg='#8DFC04')

l2.place(x=350,y=320)

e=tk.Entry(pincheck,show='X',font=('Arial',14))

e.place(x=440,y=360)

def verify():

pin=str(e.get())

if pin==data["pin"]:

atmoperations()

else:

tk.messagebox.showwarning("wrong password","Invalid Passworg")

confrimbtn=Button(pincheck,text="CONFRIM",width=20,height=0,command=verify).place(x=460,y=400)

def atmoperations():

atmoperations=Toplevel(root)

atmoperations.title("ATM Assistent")

atmoperations.geometry('1080x720')

frame2=Frame(atmoperations,width=1080,height=720,relief=RIDGE,borderwidth=5,bg='#8DFC04')

frame2.place(x=0,y=0)

l1=Label(atmoperations,text='RGUKT ATM',font=("ArialGreek 20 bold"),bg='#8DFC04')

l1.place(x=430,y=260)

l2=Label(atmoperations,text='please select the operation to perform',font=("ArialGreek 15"),bg='#8DFC04')

l2.place(x=350,y=320)

withdrawalbtn=Button(atmoperations,text="WITHDRAWAL",width=20,height=0,command=withdrawal).place(x=100,y=550)

rechargebtn=Button(atmoperations,text="RECHARGE",width=20,height=0,command=recharge).place(x=100,y=620)

balancebtn=Button(atmoperations,text="BALANCE",width=20,height=0,command=balance).place(x=800,y=550)

depositbtn=Button(atmoperations,text="DEPOSIT",width=20,height=0,command=deposit).place(x=800,y=620)

def withdrawal():

withdrawal=Toplevel(root)

withdrawal.title("ATM Assistent")

withdrawal.geometry('1080x720')

frame2=Frame(withdrawal,width=1080,height=720,relief=RIDGE,borderwidth=5,bg='#8DFC04')

frame2.place(x=0,y=0)

l1=Label(withdrawal,text='RGUKT ATM',font=("ArialGreek 20 bold"),bg='#8DFC04')

l1.place(x=430,y=260)

l2=Label(withdrawal,text='Enter the amount',font=("ArialGreek 15"),bg='#8DFC04')

l2.place(x=350,y=320)

e=tk.Entry(withdrawal,show=None,font=('Arial',14))

e.place(x=440,y=360)

def wioper():

if(data['amount']>0 and int(e.get())<=data['amount']) and int(e.get())>100 and int(e.get())%100==0:

data['amount']=data['amount']-int(e.get())

atmsay("please wait while transaction is processing")

time.sleep(10)

atmsay('please collect money')

tk.messagebox.showinfo("withdrawal","please collect money")

else:

atmsay("You account not have sufficient and valid amount")

conbtn=Button(withdrawal,text="CONFRIM",width=20,height=0,command=wioper).place(x=460,y=400)

def deposit():

deposit=Toplevel(root)

deposit.title("ATM Assistent")

deposit.geometry('1080x720')

frame2=Frame(deposit,width=1080,height=720,relief=RIDGE,borderwidth=5,bg='#8DFC04')

frame2.place(x=0,y=0)

l1=Label(deposit,text='RGUKT ATM',font=("ArialGreek 20 bold"),bg='#8DFC04')

l1.place(x=430,y=260)

l2=Label(deposit,text='Deposit Amount here',font=("ArialGreek 15"),bg='#8DFC04')

l2.place(x=410,y=320)

l3=Label(deposit,text='Account Number',font=("ArialGreek 15"),bg='#8DFC04')

l3.place(x=300,y=400)

e1=tk.Entry(deposit,show=None,font=('Arial',15))

e1.place(x=500,y=400)

l4=Label(deposit,text='Enter Amount',font=("ArialGreek 15"),bg='#8DFC04')

l4.place(x=300,y=450)

e2=tk.Entry(deposit,show=None,font=('Arial',15))

e2.place(x=500,y=450)

def deop():

if data['account.no']==str(e1.get()) and int(e2.get())%100==0:

atmsay("please place the money at the deposit holder")

time.sleep(5)

data['amount']=data['amount']+int(e2.get())

atmsay("the money was deposited into your account")

tk.messagebox.showinfo("deposit","the money was deposited into your account")

else:

atmsay("Enter the valid details")

conbtn=Button(deposit,text="CONFRIM",width=20,height=0,command=deop).place(x=390,y=500)

def recharge():

recharge=Toplevel(root)

recharge.title("ATM Assistent")

recharge.geometry('1080x720')

frame2=Frame(recharge,width=1080,height=720,relief=RIDGE,borderwidth=5,bg='#8DFC04')

frame2.place(x=0,y=0)

l1=Label(recharge,text='RGUKT ATM',font=("ArialGreek 20 bold"),bg='#8DFC04')

l1.place(x=430,y=260)

l2=Label(recharge,text='Recharge Mobile here',font=("ArialGreek 15"),bg='#8DFC04')

l2.place(x=410,y=320)

l3=Label(recharge,text='Phone NO',font=("ArialGreek 15"),bg='#8DFC04')

l3.place(x=300,y=400)

e1=tk.Entry(recharge,show=None,font=('Arial',15))

e1.place(x=500,y=400)

l4=Label(recharge,text='Enter Amount',font=("ArialGreek 15"),bg='#8DFC04')

l4.place(x=300,y=450)

e2=tk.Entry(recharge,show=None,font=('Arial',15))

e2.place(x=500,y=450)

def rech():

if data['amount']>=int(e2.get()):

data['amount']=data['amount']-int(e2.get())

atmsay("you recharge is successfull")

tk.messagebox.showinfo("recharge","you recharge is successfull")

else:

atmsay("your account not have sufficient balance")

conbtn=Button(recharge,text="CONFRIM",width=20,height=0,command=rech).place(x=390,y=500)

def balance():

atmsay("please check your balance in the information box")

tk.messagebox.showinfo("recharge","Aval Balance:"+str(data['amount']))

global id

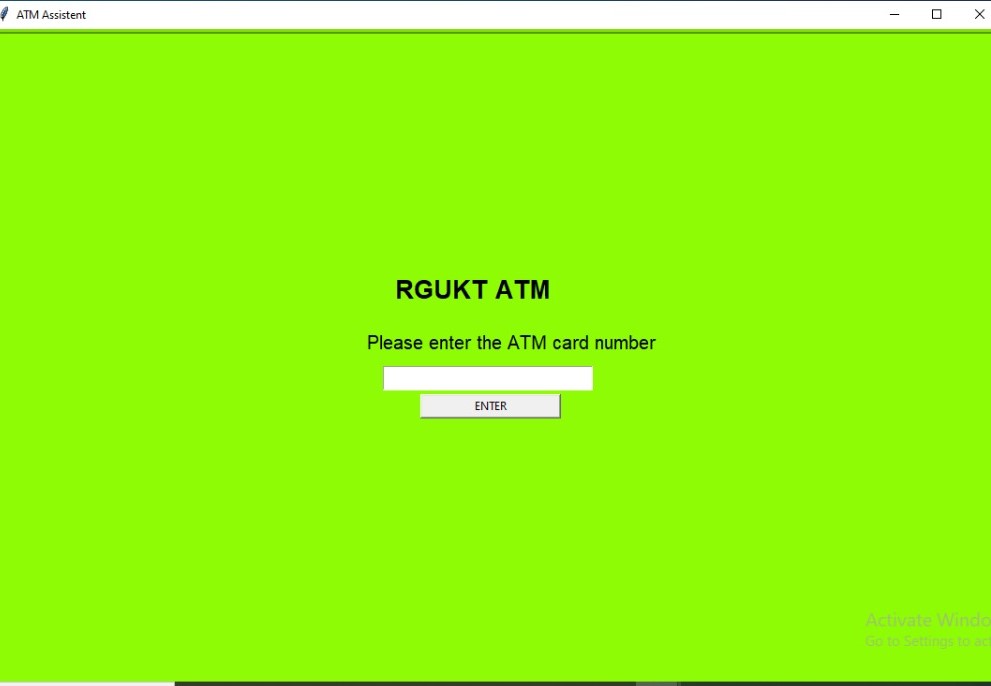
id=Entry(root,show=None,font=('Arial',14))

id.place(x=420,y=360)

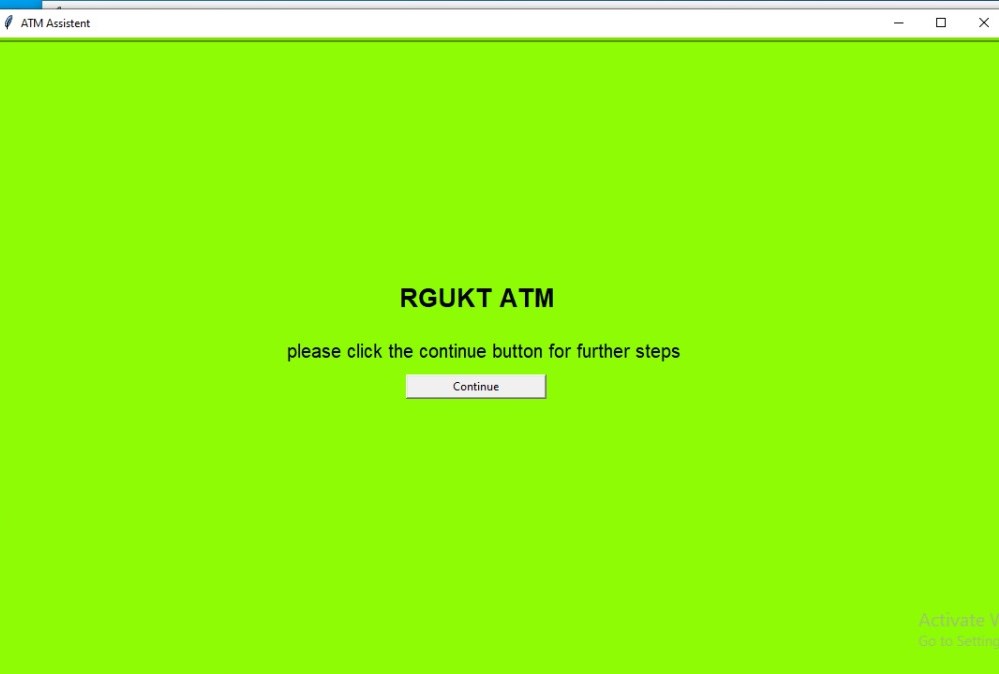
insertbtn=Button(root,text="ENTER",width=20,height=0,command=imageCapture).place(x=460,y=390)

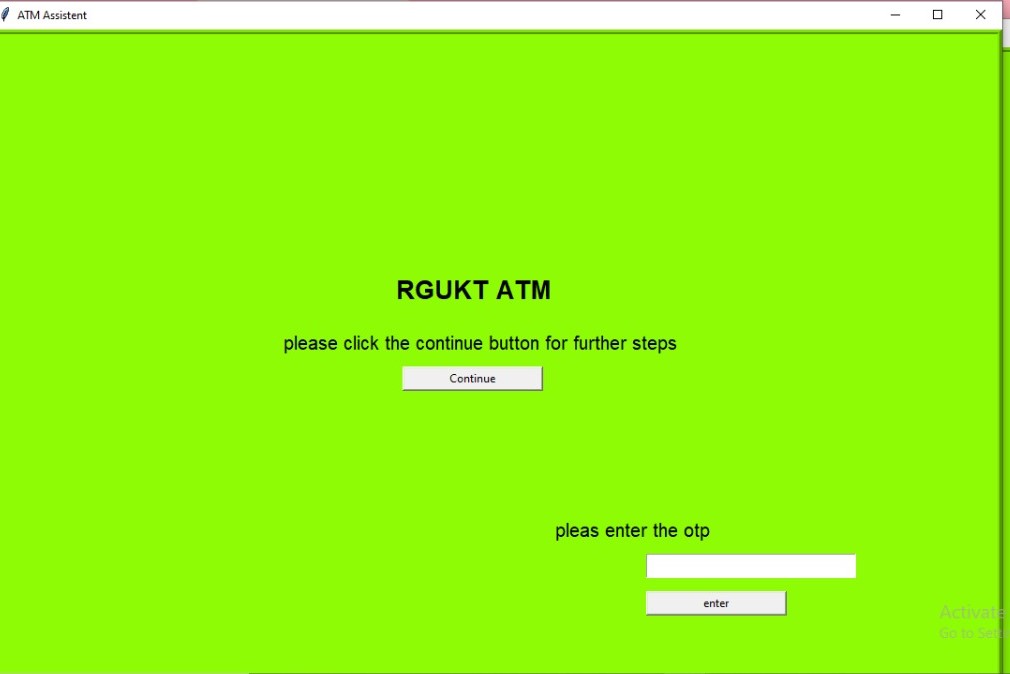
root.mainloop()

**6.2 Output Screens**

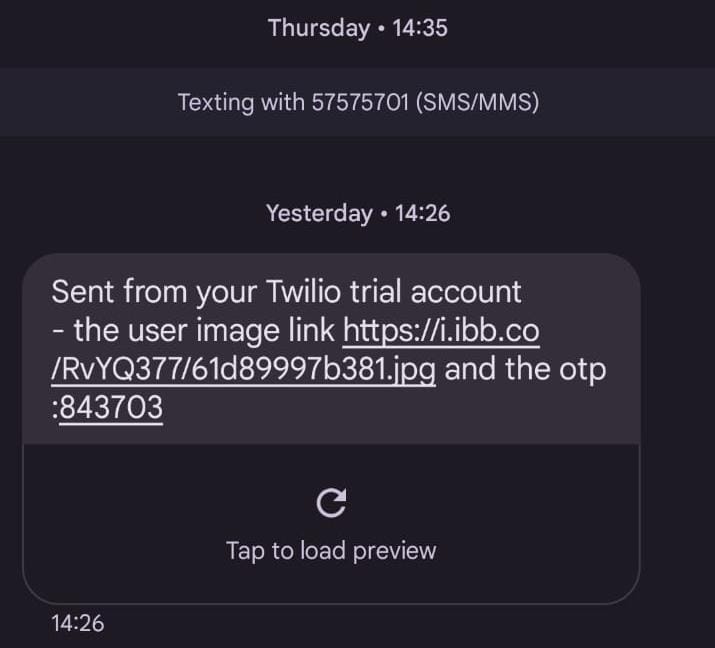
****

**Fig 6.2.1 Home Page**

****

**Fig 6.2.2 Face Capturing Screen**

**Fig 6.2.3 Permission Screen**

****

**Fig 6.2.4 Image Link and OTP**

* 1. **Test Case**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Expected output** | **Result** |
| **TC1** | Verification of user face in real time | Provide results face is matched or not | pass |
| **TC2** | Sending sms to account holder phone number. | Account holder gets the sms which contains OTP and image url. | pass |
| **TC3** | Verifying the OTP | Allows the user by verifying the OTP. | pass |

**CHAPTER-7**

**CONCLUSION & FUTURE SCOPE**

**Conclusion**

In the context of automated teller machines, our survey demonstrates the development of fusing convolutional neural networks (CNN) with facial recognition technology (ATMs). The programs covered in this article are significant steps toward resolving serious security problems, like card fraud, theft, and inaccessibility that have long dogged financial institutions. Our methods, which combine sophisticated fraud detection algorithms with intuitive features, boost ATM security while also giving users a better overall banking experience. This project exemplifies the financial industry's unwavering commitment to providing secure and easy banking services as we enter an era where digital transformation is essential. He discusses the development of the CNN model and highlights that there is constant work to be done to achieve user-centered modeling and immediate danger detection. Future developments in the industry are anticipated, and security protocols should keep getting better while safeguarding user privacy and ease. The combination of CNN and face recognition technology offers people a glimmer of hope in this ever-changing landscape by safeguarding their financial transactions and enabling them to use ATMs with confidence in an increasingly digital world.

**Future Scope**

To improve the project "Enhanced Authorization for ATM Machine through Facial Recognition," several strategic steps can be taken to enhance its effectiveness, security, and user experience. Firstly, investing in research and development to refine facial recognition algorithms and enhance their accuracy and robustness in various conditions would be crucial. This involves collaborating with experts in computer vision and machine learning to incorporate the latest advancements in facial recognition technology.

Secondly, integrating additional biometric modalities such as iris recognition or fingerprint scanning can strengthen the authentication process, providing a multi-factor authentication approach for enhanced security. This would require conducting feasibility studies and ensuring seamless integration with existing facial recognition infrastructure.

Thirdly, implementing real-time fraud detection and prevention mechanisms using machine learning algorithms can proactively identify and mitigate security threats, safeguarding ATM transactions from unauthorized access and fraudulent activities. This entails analyzing transaction patterns, user behavior, and biometric data to detect anomalies and trigger appropriate security measures.

Furthermore, exploring the adoption of blockchain technology for secure authentication and transaction verification can enhance the integrity and transparency of ATM transactions. By leveraging blockchain's decentralized and immutable nature, the project can ensure tamper-resistant record-keeping and protect sensitive transaction data from unauthorized modifications.

Moreover, prioritizing user experience enhancements such as accessibility features, multilingual support, and intuitive interfaces can make ATM transactions more inclusive and user-friendly for diverse user groups. This involves conducting user research, gathering feedback, and iteratively improving the user interface and interaction design based on user needs and preferences.

By implementing these strategic improvements, the project can enhance the security, usability, and regulatory compliance of ATM transactions through facial recognition technology, providing a safer and more convenient banking experience for users.

**CHAPTER-8**

**BIBLIOGRAPHY**

1. Arjun Kumar V “ATM Security Using Face Recognition”, International Journal of Current Engineering and Scientific Research (IJCESR) IN 2018.
2. Authors: Anjali S. More, Sayali A. Kamble, Pooja K. Bharambe, Kamini V. Satpute, Poonam N. Gajbhare “Enhancement in ATM Machine Facility Using Face Recognition Security and OTP with Shuffle Keyboard,” (IJCESR) IN 2023.
3. Smith, “Technological Advancements in ATM Systems,” in International Conference for Convergence in Technology (I2CT), 2019.
4. Johnso,” Security Challenges and Solutions,”,2020
5. J. J. Patoliya and M. M. Desai, “Face detection-based ATM security system using embedded Linux platform,” in 2017 2nd International Conference for Convergence in Technology (I2CT), 2017.
6. M. Karovaliya, S. Karedia, S. Oza, and D. R. Kalbande, “Enhanced security for ATM machine with OTP and facial recognition features,” Procedia Comput. Sci., vol. 45, pp. 390–396, 2015.
7. S. Sasipriya, P. M. Kumar, and S. Shenbagadevi, Face recognition based new generation ATM system.
8. M. Karovaliya and S. Karedia, Sharad Oza Enhanced Security for ATM Machine with Otp And Facial Recognition Features. 2015.
9. Jones and Smith shared Regulatory Compliance and Policy Implications for AIM Machine in 2019
10. T. Kwon and S. Na, “SteganoPIN: Two-Faced Human-Machine Interface for Practical Enforcement of PIN Entry Security,” IEEE TRANSACTIONS ON HUMANMACHINE SYSTEMS, vol. 46, no. 1, pp. 1–8, 2015.
11. V. Hiremath and A. Mayakar, Face recognition using Eigenface approach.“Enhanced Principal Component Analysis Recognition Performance.
12. Selvakumar, Logesh, M. Vishnu, Maniraj, and P. Kumar, “Face biometric authentication system for ATM using deep learning,” in 2022 3rd International Conference on Electronics and Sustainable Communication Systems (ICESC), 2022, pp. 647–655.
13. A. Kowshika, P. Sumathi, K. S. Sandra, A. Santhosh kumar, and R. Gokulkrishnan, “Facepin: Face biometric authentication system for ATM using deep learning,” NVEO, pp. 1859–1872, 2022.