eKYC using AI

Kishore K G 22BRS1095, Naga Rithesh 22BRS1006

***Abstract*—KYC is a mandatory process of identifying and verifying the client's identity while opening an account. KYC requires customers to provide physical documents, which are verified by businesses. eKYC, on the other hand, uses electronic means to verify customer identity which doesn’t require physical documentation. It is done by electronic means to verify customer identity .Generally eKCY is processed during a virtual call with a bank representative whose process may sometimes take hours to be completed .This project aims to automate the whole process of online eKYC by using AI . Traditional eKYC implementations have its own challenges, often involving document submissions, customer photographs, and one-time passwords (OTPs) that might be limited in ensuring the highest level of security and accuracy. By extracting customer details directly from scanned Aadhar card photographs, our project eliminates the need for manual data entry, reducing the risk of errors and enhancing user-friendliness. Enhanced database comparison automates the verification of customer details against the bank's records, ensuring accuracy and consistency while reducing the necessity for redundant data submissions. Our innovative in-person verification (IPV) process incorporates face recognition technology, strengthened security which results in accelerating the verification process. Customers no longer need to visit bank branches for physical verification, saving valuable time. With an unwavering dedication to user ease, security, and effectiveness, we offer a compelling option that guarantees a fast, secure, and smooth customer verification process.**

1. **INTRODUCTION**

KYC - Know Your Customer guidelines were issued in by RBI in to verify both the identity and address of bank customers when an account is opened. This is done to prevent fraud and financial crimes. Online KYC are generally done through a video call with a bank officer where customer’s KYC documents are verified and their signatures are recorded. In this project we use AI to authorize this KYC checking. It's a legal obligation that requires that businesses, particularly financial institutions, know and understand their customers' identity, financial activities, and associated risks. KYC's fundamental goal is to combat money laundering and other illicit financial activities by verifying the identity of individuals and entities before they can access financial services. [17] Traditionally, it is the responsibility of the banks to ensure that the customer is indeed who they claim to be. Under eKYC, a customer can register with a bank without any documents but offer their fingerprints. These will be matched with the Aadhaar database which, in turn, would be used for ascertaining identity and address.

RBI has approved UIDAI's eKYC service as a valid process for KYC verification under its Prevention of Money-laundering rule, 2005. According to it, banks have to obtain an express consent from the customer before accessing their database and obtaining their Aadhaar number. According to bankers, eKYC would make the process of acquiring new customers extremely

easy. Also, banks will be spared costs of scanning and retaining KYC documents. The KYC process involved in-person verifications, physical document submissions, and extensive paperwork which can sometimes take weeks of time to get processed. The digital age helped bring forth the era of Electronic Know Your Customer (eKYC), revolutionizing the way identity verification and customer onboarding are carried out. Electronic Know Your Customer or eKYC, marked a defining moment in the evolution of customer verification processes. This digital transformation simplified the KYC procedure, making it more efficient while also reducing operational costs. eKYC leverages the power of technology, particularly digital documentation and identity verification tools, to streamline the process. It enables customers to remotely provide their identity documents and personal information, eliminating the need for physical visit or documents, saving time and effort for both the customer and service providers.

eKYC introduced several transformative changes, including a streamlined process that simplified identity verification by allowing customers to upload digital copies of their identity documents, reducing errors that often occurred during manual data entry. The process became more accessible, enabling customers to complete it remotely from their homes or offices, enhancing the overall customer experience. Moreover, the transition from manual paperwork to digital documentation resulted in cost savings for businesses. Additionally, enhanced authentication and data encryption increased the security of customer data.

However, despite these advancements, eKYC faced its own set of challenges. Some eKYC processes still required customers to manually input personal information, which could be error-prone. Uploading identity documents proved to be complicated for some users, often resulting in image quality issues and data inconsistencies. This model simplifies this aspect by directly extracting data from scanned images. This streamlined process not only makes document submission more accessible but also reduces the likelihood of document quality issues. Traditional eKYC systems typically involve the storage of substantial customer data, which can be vulnerable to data breaches. Your project addresses this concern by reducing the storage of sensitive customer data. This model primarily extracts information from images and performs real-time verifications without retaining unnecessary customer data. This approach significantly enhances data security by limiting the exposure of customer information, thus mitigating the risks associated with data breaches. In some cases, eKYC processes still demanded in-person verification or physical document submission, negating the convenience and efficiency that digital verification promised. Traditional eKYC primarily relied on document checks and basic personal information, which might not be sufficient to prevent fraud and identity theft.

In response to these challenges, this project emerges as an innovative solution, striving to further enhance the eKYC landscape. By extracting customer details directly from scanned Aadhar card photographs, we aim to eliminate the need for manual data entry, reducing the potential for errors and enhancing user-friendliness. We also automate the database comparison, ensuring precise verification and reducing the need for repeated data submissions. The inclusion of one-time passwords (OTPs) adds an extra layer of security by confirming the legitimacy of the customer's mobile number. Additionally, we introduce an innovative in-person verification process that leverages face recognition technology, fortifying security and expediting the verification process. This approach eliminates the need for time-consuming physical verifications, setting a new standard for the eKYC landscape. This project seeks to address the challenges of traditional eKYC, making customer verification both secure and hassle-free, representing a step toward transforming the way identity verification is conducted.

**II. RELATED WORKS**

Over the years few models have been suggested that proposed alternatives to Aadhar cards for to verify customer’s identity for eKYC .

eSign version 2.0 [1]

This model presents an alternative approach to conventional eKYC and eSign processes, thereby reducing dependence on Aadhaar cards. The process begins with a resident registering with a specific Application Service Provider (ASP). During this registration process, residents can choose between two separate authentication methods: OTP-based or biometric-based. In OTP-based authentication, the resident, by selecting this method, triggers a request to generate an OTP, which is transmitted via ASP and ESP to the Unique Identification Authority of India (UIDAI). In response, UIDAI will generate an OTP and send it to the resident's registered mobile number. For residents who want biometric authentication, this process includes scanning their fingerprint or iris using an approved device.

ASP plays an important role in calculating the cryptographic hash of the signed document. This cryptographic hash, combined with the resident's consent and Aadhaar number, is transmitted to the electronic signature service provider (ESP). ESP manages key elements of the process by collecting proof of resident authentication. Additionally, ESP generates a random symmetric key (SK\_ESP\_UIDAI) and a personal identification data object (PID), contributing to the overall security and efficiency of the digital signature process.

The first limitation concerns access control to eKYC data. In the current model, access control is limited to automatic, full, and unlimited access. The author suggests that citizens may need a more flexible access control mechanism. Residents must authenticate themselves for each request, and corresponding proof of authentication must be included with each eSign request. This can result in significant authentication times, especially if people want to electronically sign many documents. The author proposes a solution in which, after obtaining the resident's consent, their authentication proof is stored with the ESP during the first request and is subsequently reused in all subsequent requests.

eKYC using blockchain[16]

It is a novel blockchain-based e-KYC scheme called e-KYC TrustBlock based on the ciphertext policy attribute- based encryption (CP-ABE) method binding with the client consent enforcement to deliver trust, security and privacy compliance.

**Client Registration:**

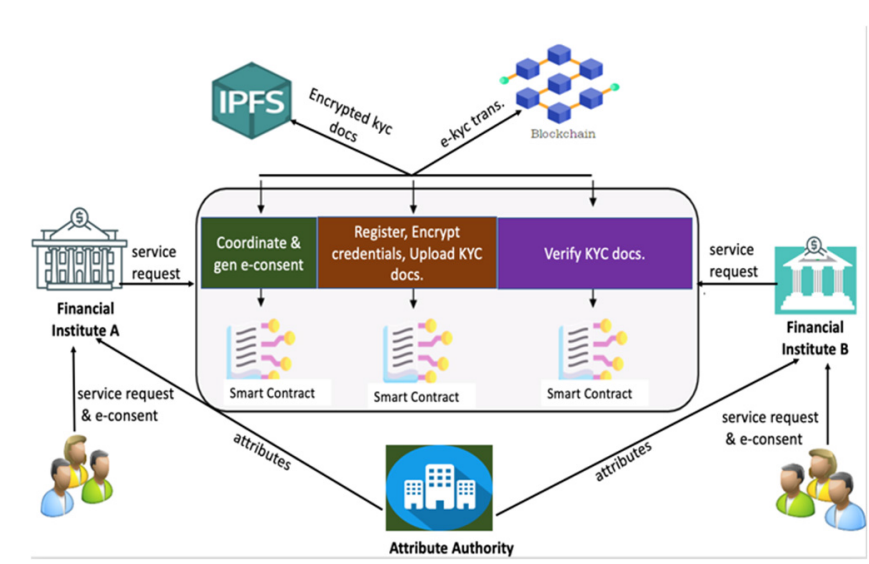
Clients initiate the registration process by providing their identity information and public key. The blockchain wallet generates a key pair (PubKClient\_id, PrivKClient\_id) for each client. Financial institutions (FIs) play a significant role in the process by calling the Master contract to generate e-consent. Clients digitally sign the e-consent using their private key (PrivKeyClient\_ID), ensuring their willingness to participate in the e-KYC process. FIs enroll clients into the system by invoking the Register contract. This step formally integrates clients into the e-KYC TrustBlock.

**Document Submission:**

Clients submit their credential documents (Crenden Files) to FIs, which store these files in their local databases.The Register contract generates an AES session key and asks clients to encrypt the session key using their public key (PubKeyClient\_ID).The encrypted credential file (EncCredenFile) and the encrypted session key (ESK) are uploaded to the InterPlanetary File System (IPFS) and the blockchain, respectively.IPFS acts as a cloud database, storing encrypted KYC documents bound to user accounts. It generates a hash value of file IDs and client's citizen IDs with SHA-256.

**Blockchain for Transaction Storage :**

The blockchain stores all KYC-related activities. It ensures the security of client transactions by encrypting sensitive data and using cryptographic mechanisms and hash values to maintain data integrity.Register contract authenticates users, enrolls new users, and uploads encrypted credentials to IPFS.Master contract manages client profiles, stores hash values of citizen IDs, generates e-consent, and performs essential functions.Verify contract is responsible for KYC verification.



**Drawbacks:**

Complexity: The system's multi-layered structure with various entities, smart contracts and technologies can introduce complexity, potentially making implementation and management challenging.

Dependency on Digital Infrastructure: The system relies on digital infrastructure, including mobile networks, blockchain technology, and encryption methods. This reliance may exclude individuals or regions with limited access to these resources. Storing sensitive KYC data in a cloud system and blockchain requires robust security measures to protect against data breaches. Any breach could have severe consequences.

Exclusivity: The accessibility of the system may be limited to individuals or entities with specific technological resources, potentially excluding those who lack access to certain technologies.

**III. ARCHITECTURE OVERVIEW**

We simulated the whole eKCY process with a website, PostgreSQL online database , c and html programs . Figure 1 gives the overview of how the website , PostgreSQL database , c and html programs simultaneously works .

1. *Components of the System*

**HTML Website:** The user interface of the e-KYC system is built on HTML websites, which provide a user-friendly platform for customers to engage in the identity verification process. These websites are frontend of this project and point of contact for users and guide them through the various steps of the verification process. Users are prompted to upload photo of their Aadhar card and use the website to take a photo of their face while also providing user instructions.

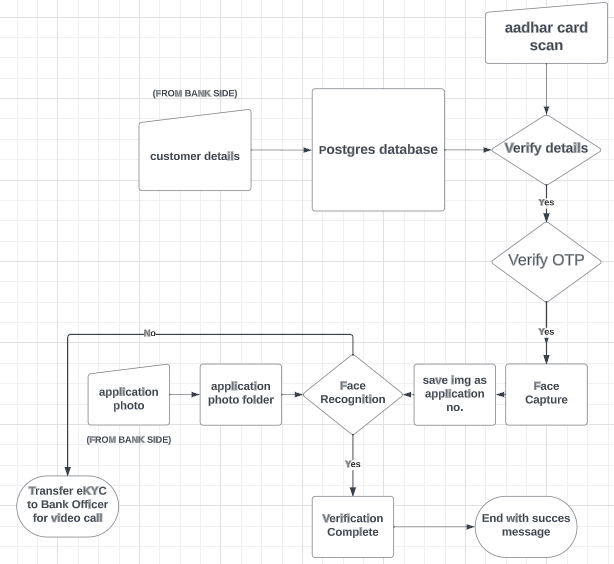
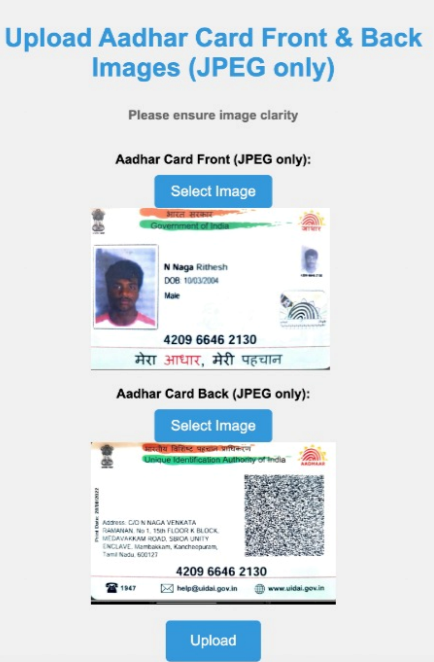
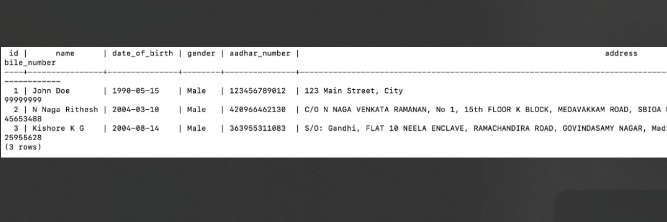


Figure 1

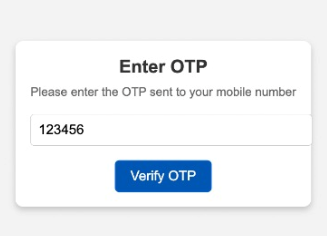
**Aadhar Card Scanning**: The e-KYC process commences with users uploading photographs of the front and back of their Aadhar cards to the website. The information is extracted using Json , open CV ,regular expression and pytesseract libraries. Then it is verified with customer information stored in PostgreSQL . The extract key information includes customer’s name, Aadhar number, gender and date of birth . This step automates the extraction process, ensuring accuracy, consistency and greatly reduces time .



**PostgreSQL Database**: PostgreSQL is an object-relational database management system. PostgreSQL is an open-source descendant of this original Berkeley code. PostgreSQL comes with many features aimed to help developers build applications, administrators to protect data integrity and build fault-tolerant environments. The heart of the system's backend operations is the PostgreSQL database, responsible for the secure storage and retrieval of customer data i.e., customer’s name, Aadhar number, gender and date of birth. This information is done from bank’s side when they receive application form of customer. Each customer's record is associated with a unique identifier, ensuring data integrity and retrieval efficiency.



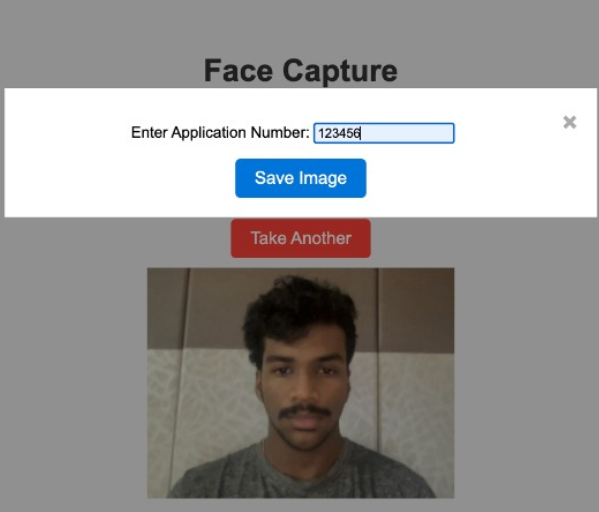
**OTP Verification:** Following successful data verification, the system requests that the user input an OTP (One-Time Password) sent to the mobile number associated with their Aadhar card. This step adds an extra layer of security and verifies that the user has control over the provided mobile number . Only when the OTP is verified , the website moves to face recognition step .



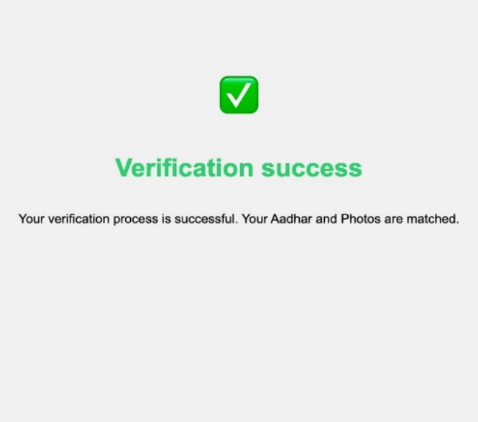
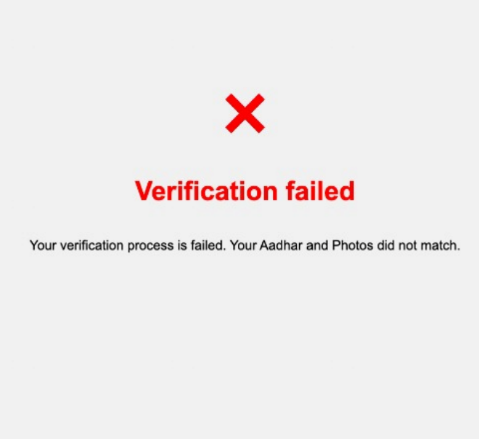
**Data Retrieval:** The data retrieval process in the e-KYC system begins by establishing a secure connection to the PostgreSQL database using the database name, username, password, host name, and port number. These parameters play a crucial role in ensuring a protected and authenticated connection to the database.

**Verification Against the Database**: Subsequently, the retrieved customer data is compared to the information provided during the e-KYC process to verify the authenticity of the customer's identity. Data retrieval is a pivotal operation, ensuring the accuracy and consistency of customer data in the e-KYC system. Specifically, the system matches the information uploaded during the customer's initial application. A successful match indicates that the customer's details are accurate and consistent.

**Photograph Capture**: The system captures a live photograph of the user and prompts user to send their application number. which is then compared to the photograph submitted with their original application. Facial recognition algorithms are used to verify the likeness between the two images.

****

**Facial Recognition**: In addition to image processing, the C code also incorporates facial recognition technology. This allows the system to compare a live photograph of the user with the photograph submitted with their original application. The facial recognition algorithms verify the likeness between the two images, ensuring that the user's identity matches the information previously provided.



**Fallback Mechanism:** In cases where the facial recognition process does not yield a satisfactory match or any other part of the e-KYC process fails, a well-structured fallback mechanism is initiated. The user is informed that the electronic verification process cannot be completed, then the user can make the choice to either retry the eKYC process or continue with traditional KYC procedure where the customer has to engage in a virtual call with a bank agents or representatives, manually verifying the user's identity to ensure compliance with KYC regulations.

**IV. WORKING**

**Algorithm 1** Aadhar card information

**function** extract\_aadhar\_info(image\_path):

img = cv2.imread(image\_path)

gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

text = pytesseract.image\_to\_string(gray\_img, lang='eng')

text\_lines = text.split('\n')

text\_lines = [line for line in text\_lines if line.strip()]

cleaned\_text = '\n'.join(text\_lines)

aadhar\_number\_pattern = *r'Pattern of aardhar card* '

aadhar\_number\_match = re.search(aadhar\_number\_pattern, cleaned\_text)

aadhar\_number = aadhar\_number\_match.group().replace(" ", "") if aadhar\_number\_match else None

name\_pattern = *r'Pattern for name'*

name\_match = re.search(name\_pattern, cleaned\_text, re.MULTILINE)

dob\_pattern = *r'Pattern for DOB’*

dob\_match = re.search(dob\_pattern, cleaned\_text)

dob = dob\_match.group(1) if dob\_match else None

gender\_pattern = r'(Male|Female|Transgender)'

gender\_match = re.search(gender\_pattern, cleaned\_text)

gender = gender\_match.group() if gender\_match

extracted\_info = {

"Name": name,

"Aadhar Number": aadhar\_number,

"Date of Birth": dob,

"Gender": gender,

img = cv2.imread(image\_path\_address)

gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

text = pytesseract.image\_to\_string(gray\_img, lang='eng')

address\_pattern = r'Address:[\s\n]+((?:[^\n]+?\n){1,10})'

address\_match = re.search(address\_pattern, text)

address = address\_match.group(1) if address\_match

}

**function** check(extracted\_info):

db\_params = {

"database": "*name*",

"user": "*user name*",

"password": "*password*",

"host": "*localhost*",

"port": "*port number*"

}

try:

conn = psycopg2.connect(\*\*db\_params)

if conn.closed == 0:

print("Database connection is open.")

else:

print("Database connection is closed.")

cursor = conn.cursor()

query = "SELECT name,date\_of\_birth,address,gender,mobile\_number FROM aadhar\_details WHERE aadhar\_number = %s"

cursor.execute(query, (extracted\_info["Aadhar Number"],))

matching\_record = cursor.fetchone()

parts = matching\_record[1].strftime("%Y-%m-%d").split("-")

formatted\_date = f"{parts[2]}/{parts[1]}/{parts[0]}"

print(matching\_record[0])

print(formatted\_date)

print(matching\_record[2])

print(matching\_record[3])

if matching\_record:

if (matching\_record[0] == extracted\_info["Name"].strip() and

formatted\_date == extracted\_info["Date of Birth"].strip() and

matching\_record[2] == extracted\_info["Address"].strip() and

matching\_record[3] == extracted\_info["Gender"].strip()):

return matching\_record[4]

return 0

This code illustrates the utilization of various libraries, including OpenCV for image preprocessing, Tesseract for optical character recognition, and regular expressions for pattern matching. It effectively extracts essential Aadhar card information and organizes this data in a structured format for further processing or analysis. The combination of these libraries and techniques streamlines the automation of information extraction from images, facilitating processes that rely on this data. Check function serves to connect to a PostgreSQL database, execute a SQL query to retrieve customer information, and verify it with Aadhar card details .

**OpenCV (**cv2): OpenCV is a powerful computer vision library used to work with images and videos. In this code, OpenCV is employed to load an image from a specified file path and convert it to grayscale. The grayscale conversion simplifies the subsequent text extraction process by reducing the complexity of the image.

**Tesseract** (pytesseract): Tesseract is an OCR (Optical Character Recognition) engine developed by Google. The `pytesseract` library integrates Tesseract with Python, allowing for text extraction from images. In this code, Tesseract is used to extract text from the preprocessed grayscale image. The extracted text is stored as a string in the variable `text`.

**Regular Expressions** (re): The Python `re` library is used for pattern matching and searching within text. Within this code, regular expressions are leveraged to search for specific patterns or information within the extracted text. Regular expressions provide a versatile and powerful tool for locating structured data.

**Data Extraction and Storage:** After text extraction and cleaning (removing empty lines), the code proceeds to search for specific patterns related to Aadhar card information. Here's how each piece of information is extracted:

**Extracting Aadhar info**: The code looks for patterns related to the Aadhar number. If a match is found, it is cleaned by removing whitespace, and the Aadhar number is stored in the `aadhar\_number` variable. In a similar manner, the code searches for patterns related to the name. The discovered name is stored in the `name` variable. The code searches for a pattern related to the date of birth (DOB). If a match is found, the DOB is stored in the `dob` variable. .A regular expression pattern is used to search for gender information, which can be "Male," "Female," or "Transgender." The extracted gender is stored in the `gender` variable.

**Data Storage and Return:** The extracted information is structured and stored in a Python dictionary called `extracted\_info`. This dictionary serves as a well-organized container for storing the different pieces of information extracted from the Aadhar card. The dictionary's keys represent the types of information (e.g., "Name," "Aadhar Number," "Date of Birth," and "Gender"), and the values hold the extracted data.

**Database Connection Parameters:** The code defines a dictionary named db\_params that contains parameters required to establish a connection to a PostgreSQL database. These parameters include the name of the database, the database user, the user's password, the host (server), and the port for the database.The code attempts to establish a connection to the PostgreSQL database using the psycopg2 library. It checks the connection status to determine whether it's open or closed.

**SQL Query:** A cursor is created to interact with the database. The cursor allows for executing SQL queries against the database. The code defines an SQL query that retrieves specific information from a table named aadhar\_details. The query selects the name, date of birth, address, gender, and mobile number based on a matching Aadhar number, which is provided as a parameter. The SQL query is executed with the extracted Aadhar Number. The result of the query is stored in the matching\_record variable.

**Data Formatting and Printing**: The code formats the date of birth retrieved from the database and prints various pieces of information, including name, date of birth, address, and gender.

**Information Validation:** The extracted information is validated against the retrieved record. If the information matches (name, date of birth, address, and gender), the mobile number is returned. If there is no match, it returns 0 to indicate a mismatch.

**Error Handling:** The code includes error handling to catch any exceptions that may occur during the database connection or query execution.

**Algorithm 2** Face recognition

**function:** read\_img(path):

img=cv2.imread(path)

(h,w)=img.shape[:2]

width=500

ratio=width/float(w)

height=int(h\*ratio)

return cv2.resize(img,(width,height))

**function:** main**()**

cus\_no=int(input()) #from website

appl\_photo\_encoding=[]

appl\_no=[]

addr="Directory of folder"

for file in os.listdir("Addres of appl. photos "):

if file.endswith(".png"):

appl\_addr=addr+"\\application photo\\"+file

img1=read\_img(appl\_addr)

cv2.imshow("hi",img1)

cv2.waitKey(1000)

img1\_enc=face\_recognition.face\_encodings(img1)[0]

appl\_photo\_encoding.append(img1\_enc)

appl\_no.append(file.split('.')[0])

print(appl\_addr)

captr\_addr=addr+"\\captures\\"+str(cus\_no)+".png" #Address of photo requested customer

img2=read\_img(captr\_addr)

img2\_enc=face\_recognition.face\_encodings(img2)[0]

print(captr\_addr)

print("The encoding of captures image:\n\n")

print(img2\_enc)

result=face\_recognition.compare\_faces(appl\_photo\_encoding,img2\_enc)

count=0

for i in range(len(result)):

if result[i]==True:

This code uses OpenCV for image reading and resizing, the os library to access image files, and the face\_recognition library for face encoding and comparison. These libraries are essential for face recognition and image processing tasks, making it possible to determine if a captured face matches any of the application photos.The os library is used to work with the operating system. In this code, it is used to iterate through files in specific directories, allowing the program to access image files in the application photo and captures folders.

1. *Image Processing*

The code uses the cv2.imread function to load images from file paths. Images are loaded as multi-dimensional arrays, which represent pixel values. These arrays contain color information, allowing for further analysis and manipulation. After loading an image, the code resizes it using cv2.resize. Image resizing is vital for ensuring uniformity and consistency when comparing different images. In this code, images are resized to have a specified width of 500 pixels while maintaining their original aspect ratio. Resizing is essential because it standardizes the dimensions of images, making it easier to compare and analyze them. To provide a visual representation of the images, the code uses cv2.imshow to display them briefly. This can be helpful for debugging and verification purposes.

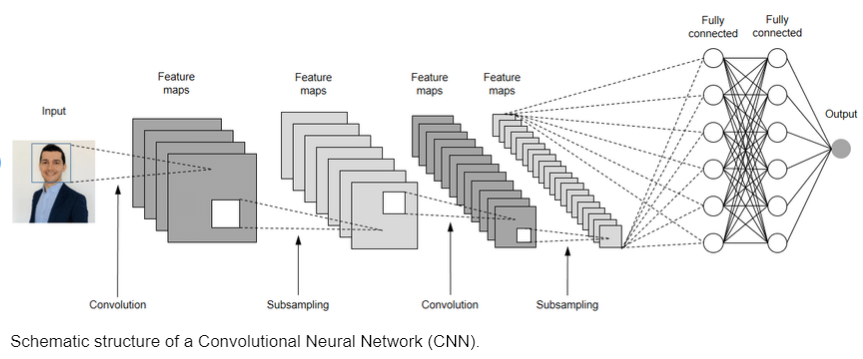
1. *Face Recognition*

The face\_recognition model was trained from scratch on a dataset of about 3 million faces. This dataset is derived from a number of datasets. The face scrub dataset[2], the VGG dataset[1] .

The code uses face\_recognition.face\_encodings to encode faces within images. Face encoding is a process of extracting distinctive features from a face, which can then be compared with other face encodings. These features capture unique details such as the arrangement of facial landmarks, and they are represented as a numerical vector.

The face recognition component is driven by the face\_recognition library, which is built on top of the dlib library and simplifies the process of face recognition. The code leverages this library to encode faces and compare them for matching.

After encoding faces in application photos, the code calculates the encoding of the customer's requested image. It then employs face\_recognition.compare\_faces to compare the encoding of the customer's face with the encodings of the application photos. The result is a list of Boolean values, indicating whether there is a match between the customer's face and any of the application photos. If the Boolean value is True, it signifies a match; if False, it means there is no match.



Neural networks, being an integral part of deep learning algorithms, form a subset within the broader field of machine learning. Node layers make up their structure, which consists of an input layer, one or more hidden layers, and an output layer. Normally every individual node establishes a connection with another node, while simultaneously possessing specific weight and threshold attributes. In case the output of a particular node exceeds the given threshold value, the node becomes activated and transmits data to the subsequent layer of the network. In case of any other situation, the data is not transmitted to the subsequent layer of the network.

CNNs are commonly used for computer vision tasks and classification purposes. Before the invention of CNNs, the identification of objects in images required manual and laborious feature extraction methods. On the other hand, when

it comes to image classification and object recognition tasks, convolutional neural networks offer a more scalable solution by utilizing principles from linear algebra, particularly matrix multiplication, to detect patterns present in an image.

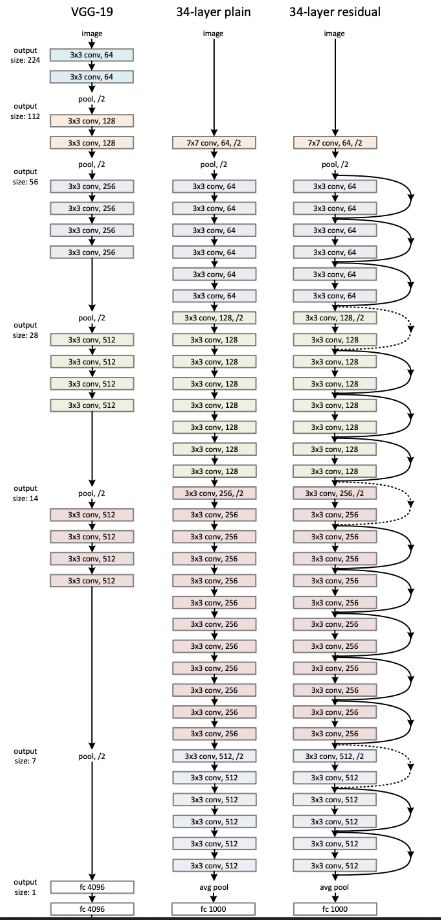


Figure 3

**[**9] ResNets are one of the most efficient Neural Network Architectures, as they help in maintaining a low error rate much deeper in the network.It performs really well where deep neural networks are required, such as feature extraction, semantic segmentations, various Generative Adversarial Network architectures. [10 ] Resnet34 is a state-of-the-art image

classification model, structured as a 34 layer convolutional neural network and defined in "Deep Residual Learning for Image Recognition". Restnet34 is pre-trained on the ImageNet dataset which contains 100,000+ images across 200 different classes .However, RestNet is different from traditional neural networks in the sense that it takes residuals from each layer and uses them in the subsequent connected layers as shown in figure3.

In a ResNet with 29 convolutional layers, the network consists of a series of building blocks, each containing multiple convolutional layers. These convolutional layers are typically followed by batch normalization and rectified linear unit (ReLU) activation functions. The key innovation of ResNet is the introduction of skip connections or shortcuts, which allow the network to skip one or more layers, thus facilitating the flow of gradients during training.

**V. CONCLUSION AND FUTURE DEVELOPMENT**

This model's potential in its field goes beyond its current capabilities, it has a lot of space to grow. In the dynamic landscape of India's banking sector, where the digital revolution is reshaping traditional practices, this project emerges with immense potential to automate electronic Know Your Customer (eKYC) process. By automating the extraction of details from Aadhar card photos and conducting data comparisons. This project not only makes it easier to authenticate, but also addresses the specific challenges posed by the unique identity landscape in India. The potential of this project is enhanced by its adaptability to the regulatory environment. While currently relying on manual or digital data uploads from bank, the project anticipates the future integration with systems like DigiLocker, aligning with digitalization and data centralization ambitions of the Indian government. The envisioned integration of advanced AI models for face recognition and direct data transfer from DigiLocker to PostgreSQL resonate with the broader governmental push for a Digital India. These future developments highlight the project's potential not only for continuous improvement but also for becoming an integral part of India's evolving banking landscape. In an era where businesses and industries are progressively in search of more secure, efficient, and technologically advanced solutions, this project is well-positioned to satisfy these demands.

In the future, one of the key area of improvements could involve implementing advanced AI models for face recognition. Leveraging more advanced deep learning and neural networks can significantly improve the accuracy of facial recognition and making it work even with cheapest webcam .This will make this eKCY model more accessible, making the verification process even more robust.

Additionally, the project could explore integration with the Indian government's DigiLocker system. By obtaining government permission and connecting directly to DigiLocker, the project could offer a more efficient eKYC process. It would enable customers to grant access to their KYC-related documents stored in DigiLocker, simplifying the data retrieval and verification process.

Integration of blockchain technology will present a promising route for the project's future development. By recording all eKYC transactions on an immutable ledger, this innovation enhances the security of data uploads, face recognition results, and verification outcomes. Customers gain great control over their data, access action requires explicit consent, cryptographically secured on the blockchain. The decentralized nature of blockchain reduces single points of failure and enhances transparency and auditability .Smart contracts can further automate eKYC procedures, reducing the need for manual intervention and ensuring data privacy.The incorporation of blockchain technology not only bolsters security and transparency but also empowers customers to control their eKYC data. It ensures that data breaches are less likely, enhances data privacy, and offers a decentralized, tamper-proof approach to KYC.

This forward-thinking development positions the project at the forefront of eKYC technology, promising an efficient, secure, and customer-friendly verification experience that adapts to the ever-changing digital landscape.

**VII. REFERENCES**

[1] Using Privacy Enhancing and Fine-Grained Access Controlled eKYC to implement Privacy Aware eSign Puneet Bakshi, Sukumar Nandi ISSN: 2415-6698

[2]https://buildmedia.readthedocs.org/media/pdf/face- recognition/stable/face-recognition.pdf

[3] https://pypi.org/project/face-recognition/

[4] https://vis-www.cs.umass.edu/lfw/

[5]https://www.sciencedirect.com/science/article/abs/pii/S0378720621001270

[6]https://www.researchgate.net/profile/AhmadGhozi/publication/360680686the\_urgency\_of\_electronic\_know\_your\_customer\_ekyc\_how\_electronic\_customer\_identificationworks\_to\_prevent\_money\_laundering\_in\_the\_fintech\_industry/links/63f89b1bb1704f343f7c7564/the-urgency-of-electronic-know-your-customer-e-kyc-how-electronic-customer-identification-works-to-prevent-money-laundering-in-the-fintech-industry.pdf

[7]https://scihub.se/https://link.springer.com/article/10.1007/s11277-021-08977-0

[8]https://link.springer.com/article/10.1007/s40804-019-00135-1

[9]https://towardsdatascience.com/resnets-why-do-they-perform-better-than-classic-convnets-conceptual-analysis-6a9c82e06e53

[10]https://roboflow.com/model/resnet34#:~:text=What%20is%20Resnet34%3F,images%20across%20200%20different%20classes.

[11] O. M. Parkhi, A. Vedaldi, A. Zisserman Deep Face Recognition British Machine Vision Conference, 2015.

[12] H.-W. Ng, S. Winkler. A data-driven approach to cleaning large face datasets. Proc. IEEE International Conference on Image Processing (ICIP), Paris, France, Oct. 27-30, 2014

[13] UIDAI, “Unique Identification Authority of India”, [Online]. Available: https://uidai.gov.in (2017).

[14] Wikipedia, “Aadhaar”, [Online]. Available:

https://en.wikipedia.org/wiki/Aadhaar (2017).

[15] Reserve Bank of India, “Master Direction - Know Your

Customer (KYC) Direction, 2016”, [Online]. Available:

https://rbidocs.rbi.org.in/rdocs/notification/PDF

s/18MDKYCD8E68EB1 3629A4A82BE8E06E606C57E57.PDF,

2018.

[16] Enabling Trust and Privacy-Preserving e-KYC System Using Blockchain SOMCHART FUGKEAW , (Member, IEEE)

Sirindhorn International Institute of Technology, Thammasat University, Bangkok 12000, Thailand

[17]https://timesofindia.indiatimes.com/business/india-business/banks-say-ekyc-errors-not-their-responsibility/articleshow/22567520.cms