FaceID: Face and ID-Based Access System

Harinarayan
2201085

Advisor: **Dr. Parashjyoti Borah**

Department of Computer Science and Engineering
Indian Institute of Information Technology

This report is submitted for the course of
CS300 : Project -I

IIIT Guwahati April 2025

Acknowledgement

I sincerely acknowledge the invaluable guidance and constant support of Dr. Parashjyoti Borah throughout this project. I extend my heartfelt gratitude to all the faculty members of the Department of Computer Science and Engineering, IIIT Guwahati, for their encouragement and knowledge. I am also thankful to my friends for being part of this memorable journey, and to my parents and family for their unwavering support and motivation at every step.

Index

Sr. No.	Content	Page. no
1	Introduction	4
2	System Architecture	6
3	Modules And Working	7
4	Experiment, Learning and Evaluation	
	4.1 Learning	9
	4.2 Experiment	15
	4.3 Evaluation	18
5	Technologies Used	21
6	Frontend	19
7	Summary	
	7.1 Future Scopes	26
	7.2 Conclusion	27
	Reference	28
	Appendix	28

Chapter 1 : Introduction

Abstract

This project leverages the power of modern machine learning techniques in the domain of facial recognition to streamline the process of secure entry and exit in restricted premises. Traditional entry systems often involve manual verification of both identity and documentation, typically requiring a person to present an ID and record their details using pen and paper. Such methods are not only time-consuming but also prone to human error and inefficiency.

To address this, we propose an automated system where an individual simply presents their face to a camera while holding their ID card close to their face. The system captures this image, detects and separates both the face and ID card from the frame, extracts identifying information (such as a roll number or ID number) via Optical Character Recognition (OCR), and verifies the person's identity using facial embeddings.

We developed this solution by integrating and working with several modern technologies, including Pytesseract for text recognition, and InsightFace's ArcFace model for face embedding and verification. The final product is a fully functional Android application that serves as the front-end interface to the system, enabling secure, monitored, and efficient access control for organisations.

Introduction

Background

In security-sensitive environments such as corporate offices, research facilities, or examination centres, it is common practice to verify both a person's face and their identification card (ID) before granting access. Traditionally, this process involves human intervention where security personnel manually inspect IDs and maintain physical logs. This method is not only time-consuming but also prone to human error, manipulation, and inefficiencies in record-keeping.

The growing demand for automated, secure, and contactless entry systems highlights the need for a reliable solution that combines facial recognition with ID verification—ensuring both who the person is and that their presented identity is valid.

Motivation

This project was inspired by the need to simplify and modernise conventional entry-exit systems without compromising on security. With the availability of powerful machine learning models and image processing tools, we saw an opportunity to build a seamless solution that could automatically detect a person's face and their ID from a single image and perform identity verification using robust, open-source technologies.

Real-world Use Cases

- Corporate buildings requiring authenticated staff entry without manual checks.
- Exam centres where both ID/admit cards and faces need to be verified quickly.
- Government facilities demanding high-security entry monitoring.

• Colleges and hostels for student in/out tracking using roll numbers and photo IDs.By combining face recognition (InsightFace with ArcFace) and text extraction (EasyOCR and PyTesseract) into a single pipeline, our solution enables a contactless, efficient, and automated system for identity verification and logging.

Chapter 2 : System Architecture

This section outlines the complete workflow of our system, showcasing how various components interact to deliver a seamless, automated face and ID verification experience.

At a high level, the process begins with capturing an image of a person holding their ID card near their face. This image serves as the input to the pipeline. The system then processes this image through a sequence of modules—each responsible for a distinct task such as face detection, ID region detection, optical character recognition (OCR), face verification and creating digital records

The following flowchart illustrates the end-to-end journey of data through our pipeline:

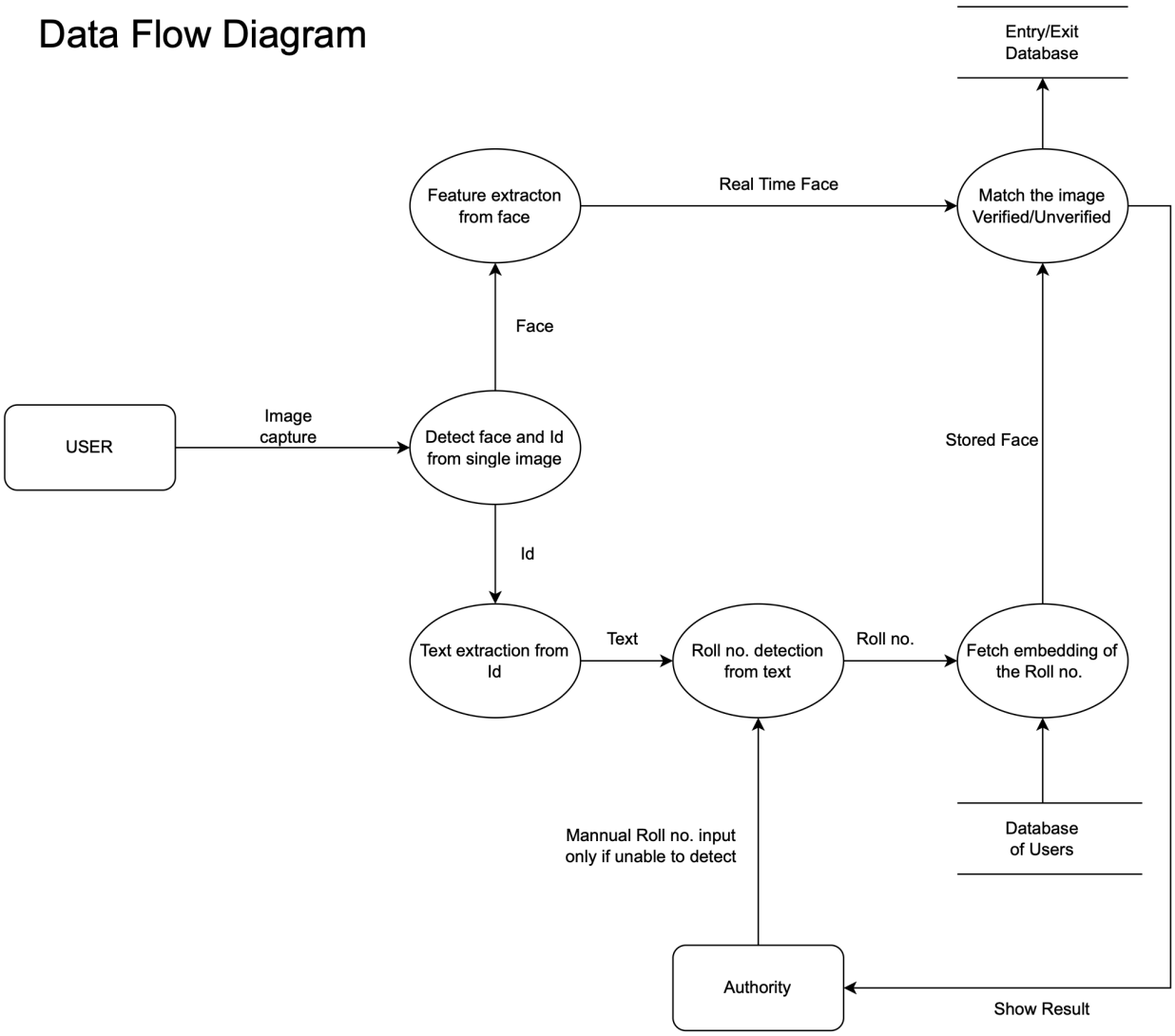


Fig.1 Data Flow Diagram

Chapter 3 : Modules and Working

Here it presents a concise overview of each core module in our system. Together, these modules work in harmony to automate the entry-exit verification process based on facial recognition and roll number extraction.

1. Image Input Module

The system supports two modes of image acquisition:

- Image Upload: Users* can upload an image from the device gallery or file system.
- Live Capture: Users* can capture an image in real-time using the camera.

In both cases, the image must clearly show the user's face and the ID (with roll number) held close to the face.

2. Face Detection & Embedding Extraction

The captured image is processed using RetinaFace, a robust and accurate face detection model that identifies facial regions with high precision.

Once detected, the face region is passed to InsightFace's buffalo_l model, a pre-trained ArcFace model built on a ResNet-50 architecture. This generates a 512-dimensional face embedding, uniquely representing the user's facial features for verification.

3. Roll Number Detection via OCR

Instead of detecting the entire ID card, we directly extract the roll number from the image using:

- Pytesseract: An OCR engine used to recognise and extract textual data from the ID card region.
- Regex (re): Regular expressions are applied to filter and identify valid roll numbers from the raw OCR output.

4. Manual Roll Number Input

In scenarios where OCR fails or the roll number is not clearly visible in the image, the system allows manual input of the roll number.This fallback ensures continuity in verification without re-capturing or re-uploading the image.

5. Databases

Two SQLite databases are used:

- User Embeddings Database: Stores pre-computed face embeddings indexed by roll number. This acts as the verification ground truth.
- Entry/Exit Log Database: Records successful verifications with roll number, date, and time.

The system automatically differentiates between entry and exit based on previous records and sorts them accordingly.

6. Embedding Comparison & Result Evaluation

Once the face embedding from the input image is extracted, it is compared with the stored embedding (based on the detected or manually entered roll number).

In this threshold-based approach, a statistical method is employed to determine an optimal threshold over the cosine similarity between two embeddings, facilitating the decision on whether they correspond to the same individual.

Only on successful verification, the entry or exit is logged in the database, ensuring security and accuracy.

* A authority that is responsible for monitoring the real time access at access point of the premises like security guard

Chapter 4 : Experiments, Learning and Evaluation

4.1 Learning

A Brief History of Face Recognition

Face recognition (FR) has grown into one of the most trusted biometric technologies, widely adopted in sectors like security, finance, defence, and consumer tech.

Early Era – Handcrafted Features : In the early 2000s, FR relied on handcrafted features such as Gabor filters and Local Binary Patterns (LBP). These methods offered some robustness against lighting and expressions but struggled with generalisation. They addressed specific challenges in isolation and plateaued at around 95% accuracy on benchmarks like LFW.

2012 – Rise of Deep Learning : A turning point came with AlexNet in 2012, which introduced deep learning to the vision community. CNNs automatically learned hierarchical features—from edges to facial attributes—outperforming traditional approaches.

2014 – DeepFace & Beyond : Facebook's DeepFace model (2014) achieved 97.35% accuracy on LFW, nearly matching human performance. Trained on millions of images using deep neural networks, it ushered in the deep learning era for FR.

Modern Developments :Today, models like FaceNet, ArcFace, and InsightFace have pushed accuracy above 99.8%, offering robust face recognition under diverse real-world conditions. Deep learning now powers the full FR pipeline—from face detection to embedding comparison—with models like FaceNet being widely used in production systems.

Overview of face recognition

A modern Face Recognition (FR) system typically consists of three main components:

1. Face Detection & Alignment

   The system begins by locating the face in an image using a face detector. Then, facial landmarks are used to align the face into a standardised format for consistent processing.

2. Feature Extraction

   Deep neural networks (e.g., ResNet-based architectures like ArcFace) are used to extract identity-specific embeddings from aligned faces. These features are compact and highly discriminative, thanks to specialised training losses like Softmax variants and Additive angular margin-based losses.

3. Face Matching

   Finally, face verification (1:1) or identification (1:N) is performed using similarity metrics like cosine or L2 distance. For improved performance, additional techniques like metric learning or sparse classifiers can be used.

Additional pre-processing, such as face anti-spoofing and pose normalisation, helps increase robustness under real-world conditions like occlusions, expressions, and lighting variations.

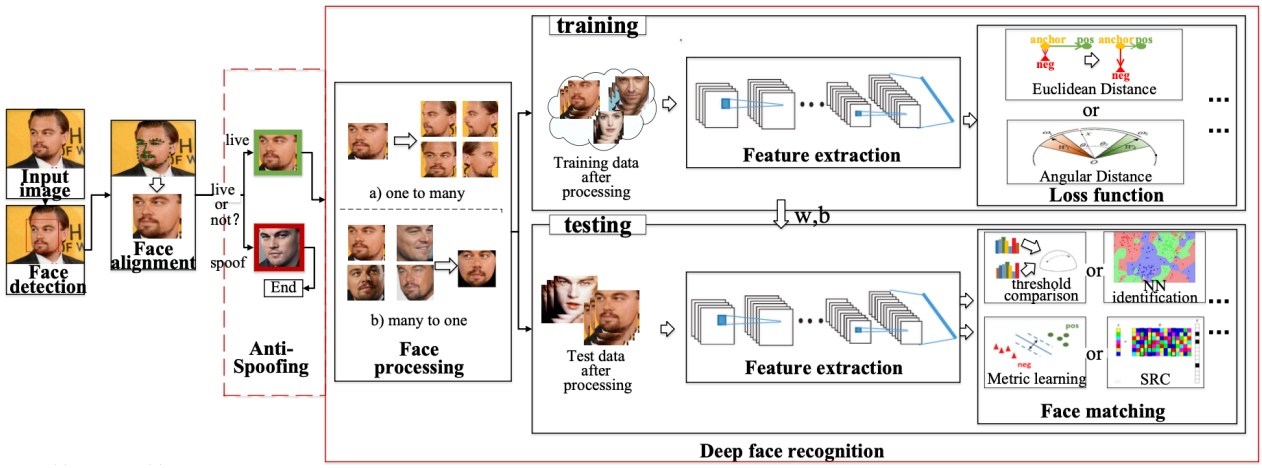


Fig.2 Face recognition method

Overview of Loss Functions in Face Recognition

Face recognition systems rely on deep learning models to create discriminative feature embeddings that can distinguish between different identities. The loss function plays a crucial role in training these models by guiding how well the network learns to separate different identities while keeping the same identity's features close together.

Traditional Softmax Loss and Its Limitations

The standard softmax loss for classification is defined as:

$$L_1 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}}$$

Where:

- x_i is the deep feature of the i-th sample
- W_j represents the j-th column of the weight matrix
- b_j is the bias term
- y_i is the class label of the i-th sample

However, softmax loss has two significant limitations:

1. It doesn't explicitly optimise feature embeddings to maximise intra-class similarity and inter-class differences
2. The learned features might be separable for closed-set classification but lack discriminative power for open-set face recognition

Evolution to ArcFace

From Softmax to Normalised Softmax

The first improvement is to normalise both the features and weights, transforming the logit (dot product) into a cosine similarity:

$$W_j^T x_i = \| W_j \| \cdot \| x_i \| \cdot \cos\theta_j$$

After normalisation and re-scaling by a factor s, the loss becomes:

$$L_2 = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}} + \sum_{j=1, j\neq y_i}^{n} e^{s\cos\theta_j}}$$

This places all feature vectors on a hypersphere of radius s, making the prediction dependent only on the angle between the feature and weight vectors.

ArcFace: Adding Angular Margin Penalty

ArcFace introduces an additive angular margin penalty m to the target angle:

$$L_3 = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s\left(\cos\left(\theta_{y_i} + m\right)\right)}}{e^{s\left(\cos\left(\theta_{y_i} + m\right)\right)} + \sum_{j=1, j\neq y_i}^{n} e^{s\cos\theta_j}}$$

The key innovation is applying the margin directly to the angle θ_{yi} before converting back to cosine space. This corresponds exactly to adding margin on the geodesic distance of the hypersphere.

Why ArcFace is Superior

1. **Geometric Interpretability**: ArcFace has a clear geometric meaning as it directly operates on the angles between feature vectors, which represent the geodesic distances on the hypersphere.
2. **Constant Linear Angular Margin**: Unlike other methods like SphereFace and CosFace that have nonlinear angular margins, ArcFace maintains a constant angular margin throughout the interval, which provides consistent and stable gradients during training.
3. **Enhanced Discriminative Power**: ArcFace simultaneously enhances intra-class compactness (features from the same identity cluster closer) and inter-class discrepancy (features from different identities stay farther apart).
4. **Implementation Simplicity**: ArcFace is straightforward to implement (only a few lines of code) and doesn't require joint supervision from the standard softmax loss to stabilise training, unlike SphereFace.
5. **State-of-the-Art Performance**: Experimental results show that ArcFace consistently outperforms other loss functions across multiple benchmark datasets. For example, on the

MegaFace Challenge, ArcFace achieves 81.03% for identification and 96.98% for verification, surpassing CosFace and other methods.

Comparison with Other Advanced Loss Functions

vs. SphereFace

SphereFace uses a multiplicative angular margin, defined as:

- Uses $\cos(m\theta_{yi})$ instead of $\cos(\theta_{yi} + m)$
- Requires approximations to compute, leading to unstable training
- Often needs to be combined with softmax loss to converge

vs. CosFace

CosFace (also known as Large Margin Cosine Loss) applies an additive cosine margin:

- Uses $\cos\theta_{yi} - m$ instead of $\cos(\theta_{yi} + m)$
- Works directly in cosine space, not angular space
- Has a nonlinear effect on the angular margin

Empirical Comparisons

In direct comparisons from the paper:

- On LFW: ArcFace (99.53%) outperforms SphereFace (99.42%) and CosFace (99.51%)
- On CFP-FP: ArcFace (95.56%) outperforms SphereFace (94.38%) and CosFace (95.44%)
- On AgeDB-30: ArcFace (95.15%) outperforms SphereFace (91.70%) and CosFace (94.56%)

Conclusion

ArcFace represents a significant advancement in loss functions for face recognition. By directly optimising the geodesic distance on the hypersphere through an additive angular margin, it creates more discriminative face embeddings with enhanced intra-class compactness and inter-class separation. Its clear geometric interpretation, implementation simplicity, and superior performance across multiple benchmarks make it an excellent choice for face recognition systems.

Face Verification vs. Face Identification

Face verification determines if two facial images belong to the same person (1:1 matching), answering "Are you who you claim to be?" Face identification, on the other hand, compares a facial image against a database to determine identity (1:N matching), answering "Who are you?" As shown in the data, verification consistently achieves higher accuracy across all methods compared

to identification.

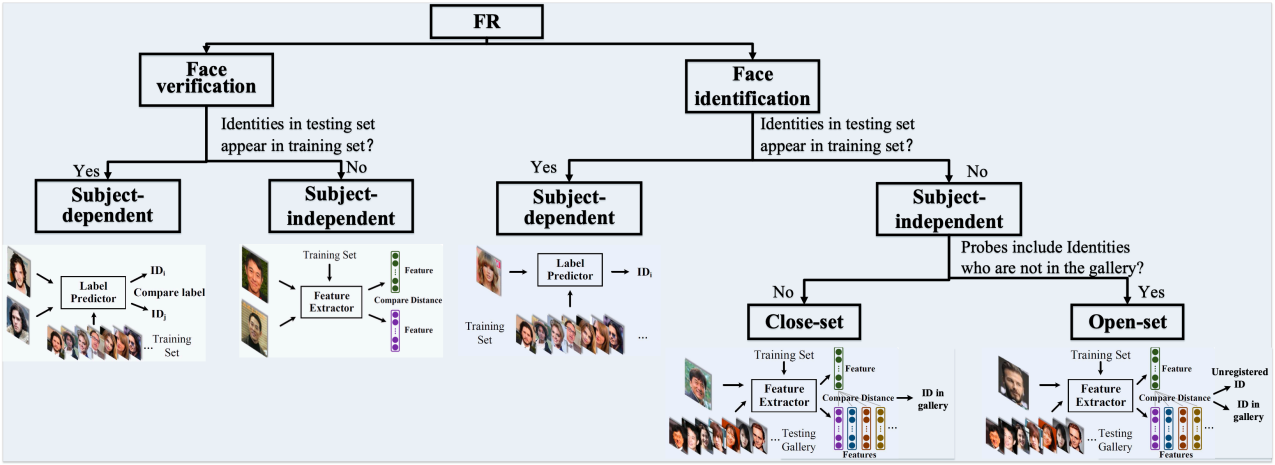


Fig.3 Face identification vs Face recognition

Benefits of Face Verification:

1. Higher accuracy rates across all methods (10-15% better on average), with top models achieving over 98% verification accuracy compared to lower identification rates. For example, CosFace shows 89.88% verification accuracy versus 77.11% for identification.

2. Computationally more efficient as it only requires comparing two images rather than searching through an entire database, making it ideal for real-time applications and devices with limited processing power. This efficiency translates to faster processing times and lower hardware requirements.

3. Better scalability as verification performance remains stable even as the number of identities increases, while identification becomes more challenging with larger databases. This makes verification particularly valuable for growing systems where new users are frequently added.

4. More suitable for security-critical applications like access control and financial transactions where the primary concern is confirming a claimed identity rather than identifying an unknown person from many possibilities.

Methods	Id (%)	Ver (%)
Softmax [15]	54.85	65.92
Contrastive Loss[15, 30]	65.21	78.86
Triplet [15, 27]	64.79	78.32
Center Loss[36]	65.49	80.14
SphereFace [15]	72.729	85.561
CosFace [35]	77.11	89.88
AM-Softmax [33]	72.47	84.44
SphereFace+ [14]	73.03	-
CASIA, R50, ArcFace	77.50	92.34
CASIA, R50, ArcFace, R	91.75	93.69
FaceNet [27]	70.49	86.47
CosFace [35]	82.72	96.65
MS1MV2, R100, ArcFace	81.03	96.98
MS1MV2, R100, CosFace	80.56	96.56
MS1MV2, R100, ArcFace, R	98.35	98.48
MS1MV2, R100, CosFace, R	97.91	97.91

Table.1 : Face identification and verification evaluation of different methods on MegaFace Challenge1as the probe set. “Id” refers to the rank-1 face identification accuracy with 1M distractors, and “Ver” refers to the face verification TAR at 10−6 FAR.

4.2 Experiment

To evaluate the effectiveness of our face verification system, we performed a series of experiments using a well-known benchmark dataset and a robust deep learning model. The entire pipeline—from embedding generation to threshold calculation—was built and tested with real data.

Dataset

In this project, we utilised the Cross-Age Labeled Faces in the Wild (CALFW) dataset — a refined extension of the original LFW dataset designed to introduce more realistic challenges for face verification.

While the original LFW dataset has long served as a benchmark for unconstrained face verification, its accuracy saturation (~99%) no longer reflects real-world performance. CALFW addresses this by introducing age variation between positive pairs (same identity) and ensuring controlled negative pairs (different identity, but same gender and race). This increases intra-class variance while keeping inter-class variance minimal, making verification significantly harder and more practical.

CALFW contains:

- 3,000 age-separated positive pairs
- Carefully selected negative pairs with minimal attribute bias
- Same image count and identities as LFW to maintain compatibility with traditional benchmarks

This dataset allowed us to simulate more realistic face recognition scenarios, particularly in applications requiring long-term identity verification.

Dataset link: [CALFW Dataset URL]

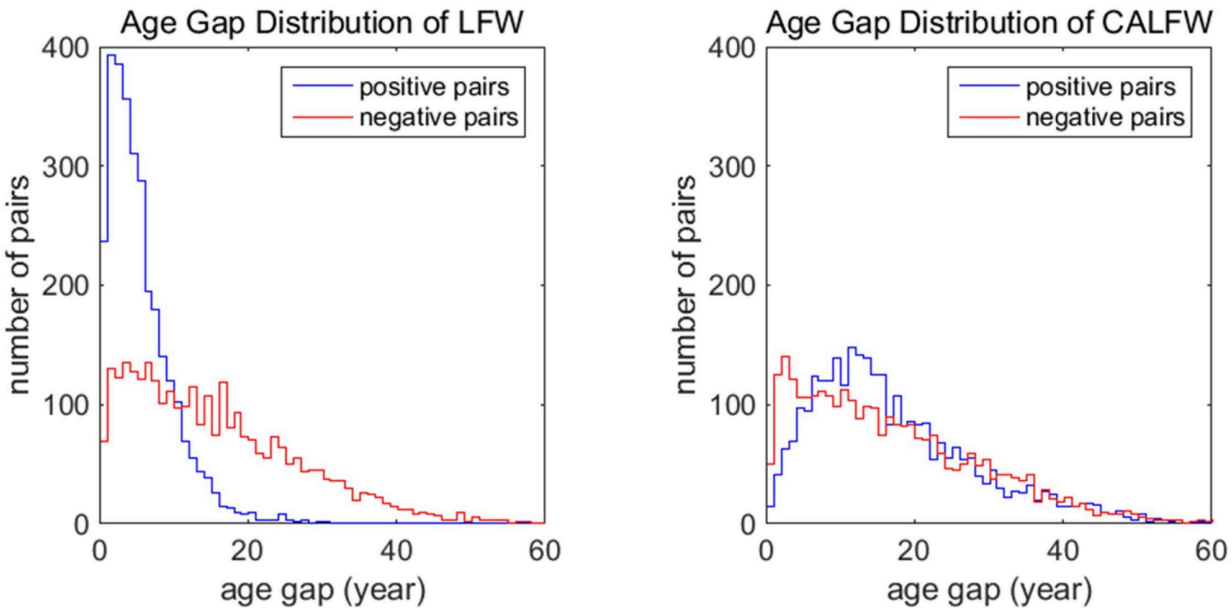


Fig.4 Age variation between LFW and CALFW

There was a lot impact on accuracy while using this dataset :

Method	LFW	CALFW
Centerface[1]	1.17	14.52 (↑ 1241%)
SphereFace[2]	0.65	9.70 (↑ 1492%)
VGGFace2[3]	0.49	9.43 (↑ 1924%)
ArcFace[4]	0.10	4.55 (↑ 4550%)

Table 2: COMPARISON OF 10-FOLD VALIDATION ERROR (%) OF FOUR SOTA DEEP FACE RECOGNITION MODELS. THE INCREASE OF ERROR IS ALSO ENUMERATED WHEN TRANSFERRING FROM LFW TO CALFW.

Face Embedding Model

For feature extraction, we used the buffalo_l model from InsightFace, which is built on a ResNet-50 backbone with ArcFace as the loss function. ArcFace is particularly known for producing highly discriminative face embeddings by enforcing angular margin penalties between classes. This model outputs 512-dimensional embeddings, which are robust to variations like pose, lighting, and expression.

Embedding Generation and Storage :We processed the dataset, which consisted of approximately 12,172 face images, to generate embeddings using the pre-trained InsightFace model. Each image's embedding was stored in a local SQLite database, indexed by identity.

Similarity Calculation

To evaluate our face verification capability, we performed two types of comparisons using cosine similarity between embedding pairs:

1. Genuine Pairs (Same Person)

For each individual in the dataset with n images, we calculated the similarity between every possible pair of their images. For instance, if person A had 3 images and person B had 4 images:

- A would yield 3 similarity scores (C(3,2) = 3)
- B would yield 6 similarity scores (C(4,2) = 6)

This ensured we had comprehensive coverage of within-class similarity. This way we got 13,951 entries.

2. Imposter Pairs (Different Persons)

To ensure balance with the genuine class, for each embedding in the database, we randomly selected exactly one embedding from a different identity and computed the cosine similarity. This generated a matching number of imposter comparisons to the genuine ones. This way we got 12,172 entries.

Bayesian Thresholding for Verification

Once we had our similarity scores for both genuine and imposter classes, we employed Bayesian Inference to determine an optimal decision threshold. Here's how the process worked:

- We plotted the distribution of similarity scores for both classes.
- We derived the probability density functions (PDFs) for both classes.
- The Bayesian threshold was determined as the intersection point of these two distributions i.e., the score at which the posterior probabilities of a match and non-match are equal.

This threshold was then used to make the idea of final verification decision in real-time scenarios.we will further this threshold to determine the actual final threshold which will be in close proximity to it and depends on our requirements.

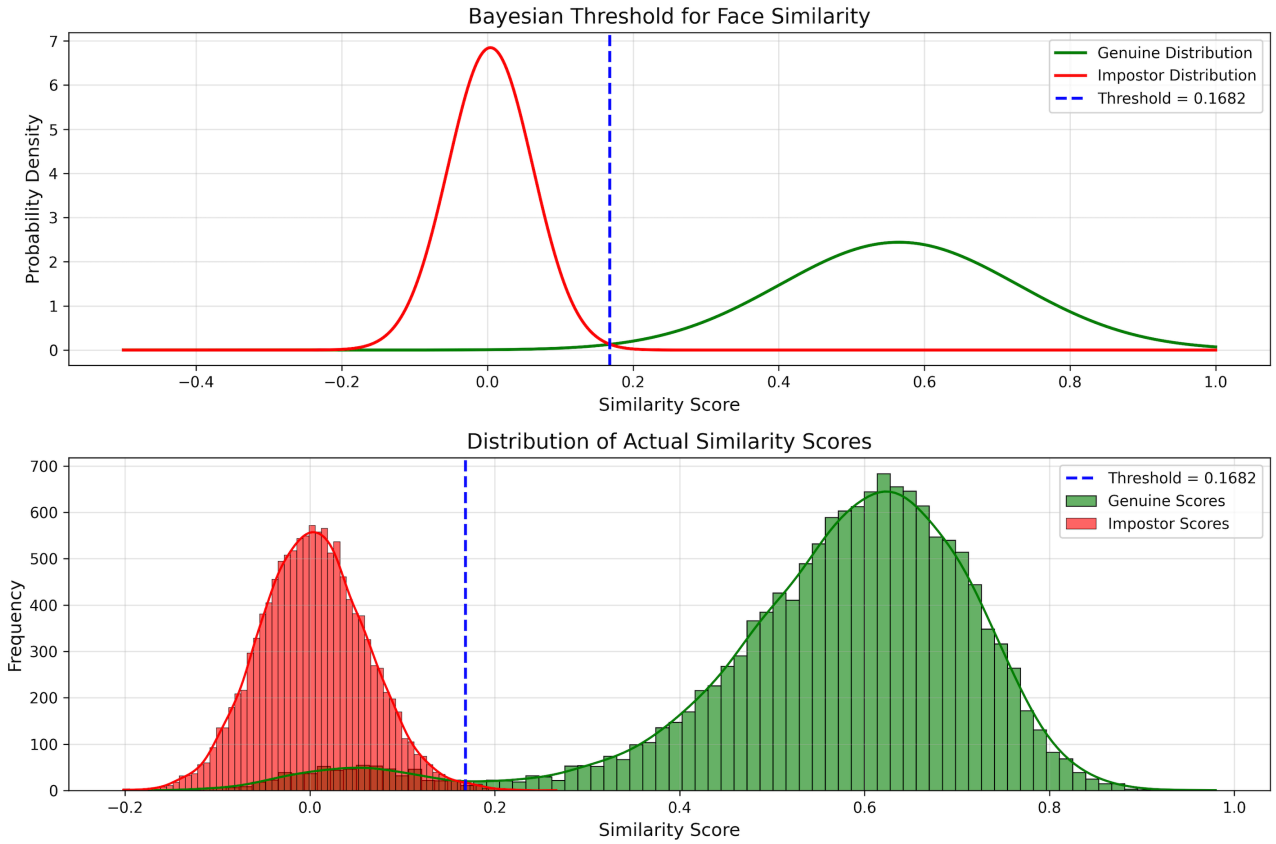


Fig.5 Probability distribution of genuine and imposter scores along with Bayesian Threshold

4.3 Evaluation

Evaluation Metrics

To comprehensively assess the performance of our face verification system, we employed a variety of standard evaluation metrics. Each of these metrics helps analyse a different aspect of the model's ability to distinguish between genuine and imposter pairs. Below is an overview of the key metrics used:

Accuracy measures the overall correctness of the verification system by calculating the ratio of correct predictions (both true positives and true negatives) to the total number of predictions. It provides a general indication of how well the model performs but may not be sufficient on its own, especially in imbalanced datasets where genuine or imposter pairs dominate. Accuracy is useful when false positives and false negatives are equally critical.

Precision is the ratio of true positives to the total predicted positives. In the context of face recognition, it tells us how many of the predicted "matches" (i.e., pairs classified as same person) are actually correct. High precision is essential in high-security scenarios where false acceptance (mistakenly identifying a person) must be minimised.

Recall (Sensitivity or True Acceptance Rate) measures the ability of the system to correctly identify all actual positives (i.e., genuine pairs). It is particularly crucial in applications where failing to recognise a legitimate person (false rejection) is costly or inconvenient. High recall ensures that most genuine matches are accepted.

The **F1 Score** is the harmonic mean of precision and recall. It offers a balance between the two, especially when there's an uneven distribution between positive and negative classes. In face recognition, F1 Score gives a single metric that considers both the need for few false acceptances and few false rejections.

True Acceptance Rate (TAR) is the proportion of genuine matches that are correctly identified by the system. It is similar to recall but is often used in biometric evaluations, especially when reporting performance against a specific threshold or False Acceptance Rate. A high TAR indicates that the system effectively authenticates valid users.

True Rejection Rate (TRR) reflects the proportion of imposter attempts correctly rejected by the system. It is crucial in applications where preventing unauthorised access is a top priority. High TRR signifies strong protection against identity spoofing or impersonation.

False Acceptance Rate (FAR) indicates the percentage of imposter attempts that were mistakenly accepted as genuine. In security-sensitive systems, a low FAR is critical as false acceptances can lead to unauthorised access. It is inversely related to TRR and must be minimised.

False Rejection Rate (FRR) measures the percentage of genuine users who are incorrectly rejected by the system. A high FRR can lead to user frustration and operational inefficiencies. FRR is particularly important in user-facing applications where convenience matters.

Here all data derived from threshold_metric.txt

Face Recognition Threshold Recommendations

1. For High Security Applications

Recommended Threshold: 0.2682

- **Key Metrics**: FAR = 0.0000, FRR = 0.0568
- **Justification**: This threshold guarantees zero false accepts, making it ideal for high-security environments where preventing unauthorised access is the absolute priority. The 5.68% false rejection rate is an acceptable trade-off for maximum security assurance.

2. For Balanced Security/Usability Requirements

Recommended Threshold: 0.1825

- **Key Metrics**: Accuracy = 0.9740, F1 = 0.9751, FAR = 0.0026
- **Justification**: This threshold provides the highest overall accuracy and F1 score, offering an excellent balance between security and convenience. With a very low false accept rate (0.26%) and reasonable false reject rate (4.64%), it's suitable for environments requiring both security and operational efficiency.

3. For Surveillance Applications

Recommended Threshold: 0.1111

- **Key Metrics**: FAR = 0.0353, FRR = 0.0378
- **Justification**: This threshold represents the Equal Error Rate point where false accepts and false rejects are nearly balanced. For surveillance systems that need to identify persons of interest without excessive false alarms or missed detections, this provides optimal performance balance for monitoring large populations.

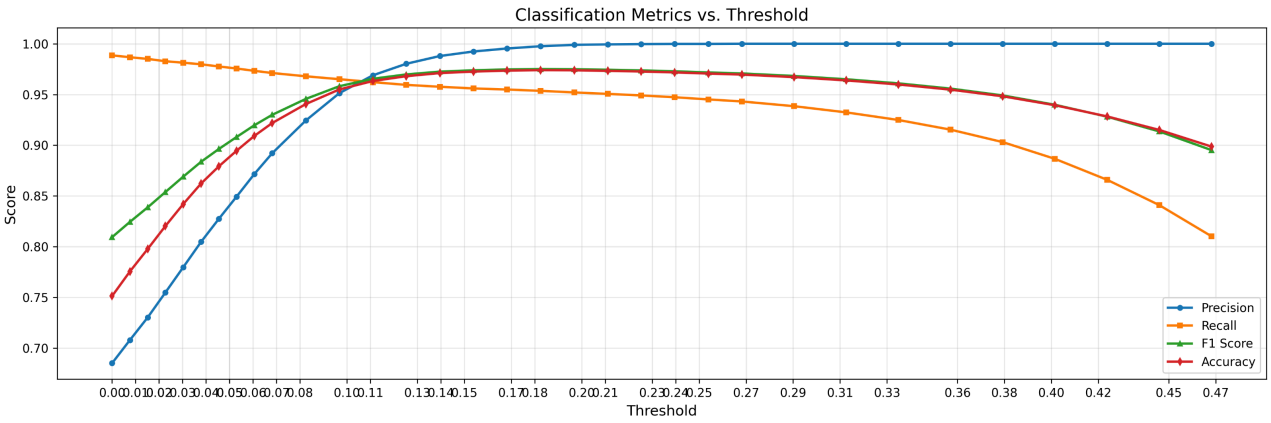


Fig.6 Precision, Recall, F1 Score and Accuracy at different threshold

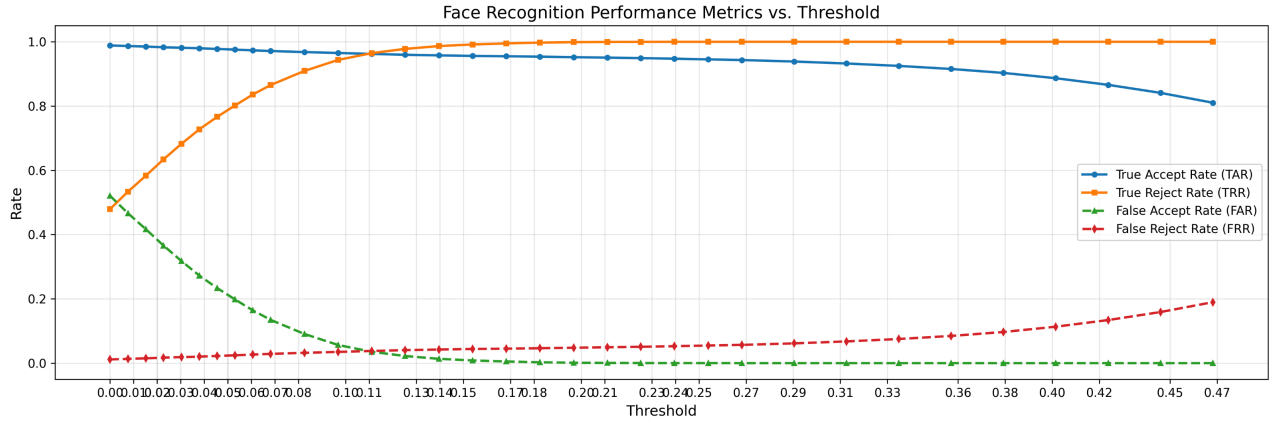


Fig.7 TAR, TRR, FAR and FRR at different Threshold

Chapter 5 : Technologies Used

Our project integrates a variety of tools and technologies across different domains to create a seamless and efficient facial recognition system:

Platform & Development Tools

- macOS – Primary development environment
- Android Studio – For mobile app development
- Visual Studio Code (VSCode) – For backend and script development
- Google Colab – Used for experimentation and training/testing in the cloud

Mobile App

- Jetpack Compose – UI framework for building modern Android UIs
- CameraX – Used for high-quality image capture
- Kotlin – Primary language for Android development
- Retrofit – For handling API calls between frontend and backend

Backend & Server

- FastAPI – Lightweight, high-performance Python web framework
- Python 3.13 – Core backend language
- Tesseract OCR – For extracting text (roll number) from ID images
- OpenCV – For image processing and manipulation
- InsightFace – For face detection and embedding generation
- ONNX – Format used for exporting and running AI models efficiently

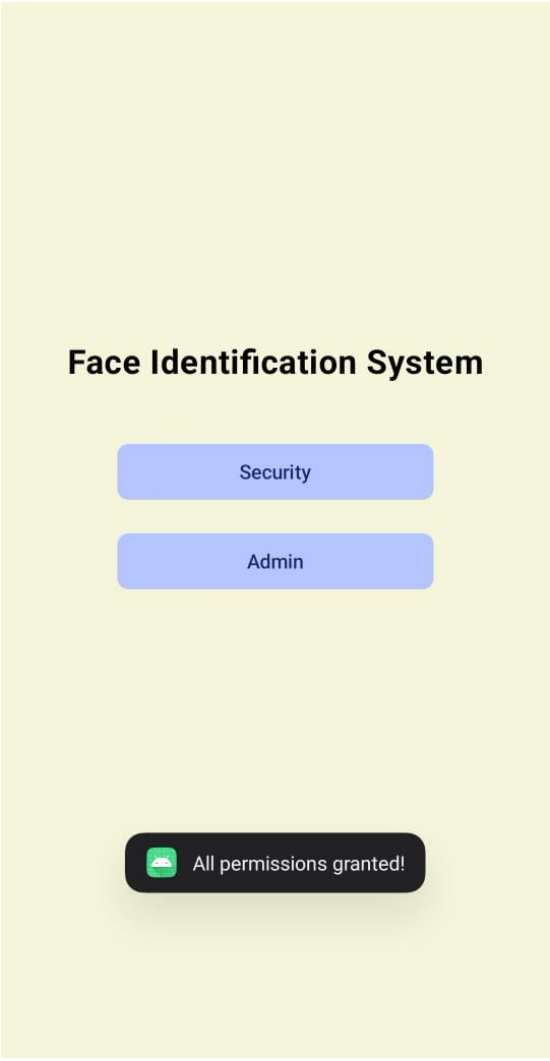
Chapter 6 : Frontend Application

The application opens with a role selection screen where the user can choose between Security and Admin. If Security is selected, the user logs in and proceeds to a screen offering two options: Capture Image using the camera or Upload Image from the device. Once an image is provided, the app attempts to detect the roll number from the ID using OCR. If successful, face verification is performed and the user is taken to a result screen showing the verification status, roll number, similarity score, and both stored and captured face images. If roll number detection fails, the user is redirected to enter it manually before verification continues.

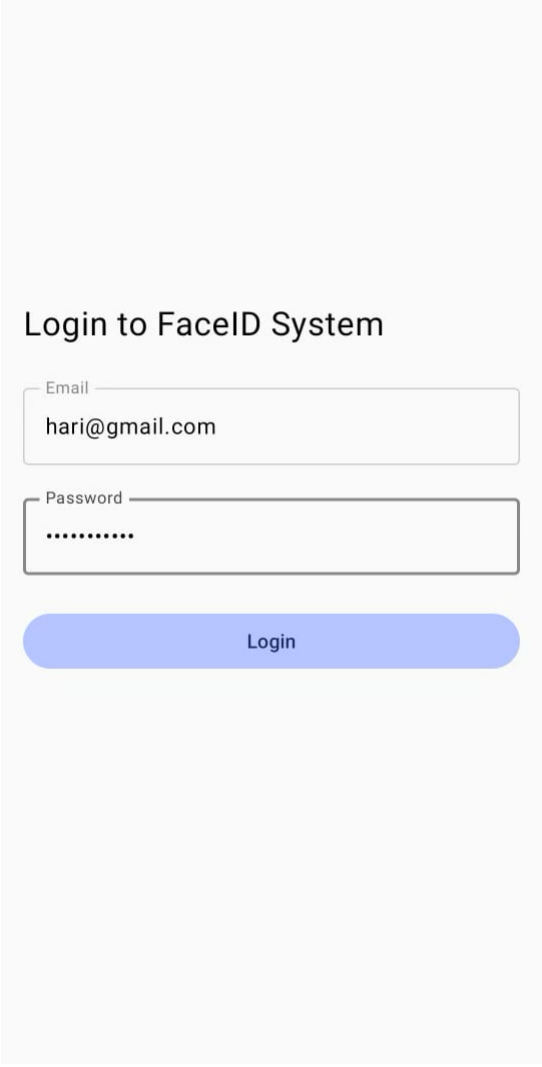
If Admin is selected, the user logs in to access a dashboard with two main options: Manage Security and View Reports. In the security section, the admin can add, edit, view, or delete security personnel. The report section displays logs of verified entries and exits, showing roll number, date, time, and entry/exit status, along with an option to delete all records. This flow ensures clear separation of duties and easy navigation for both roles.

Application Screenshot

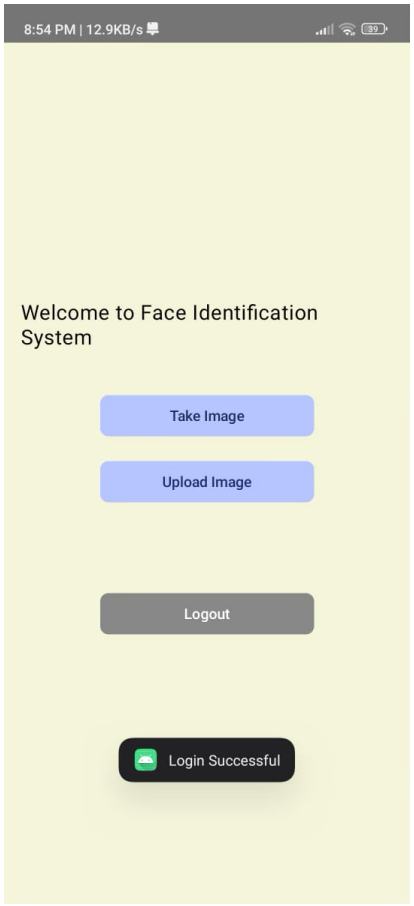
Main Screen



Security Login Page



Security Dashboard:



Camera Screen on Take Image

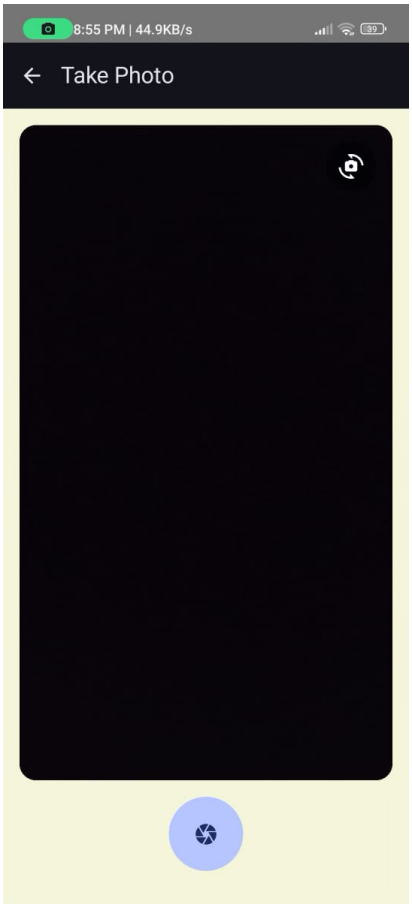
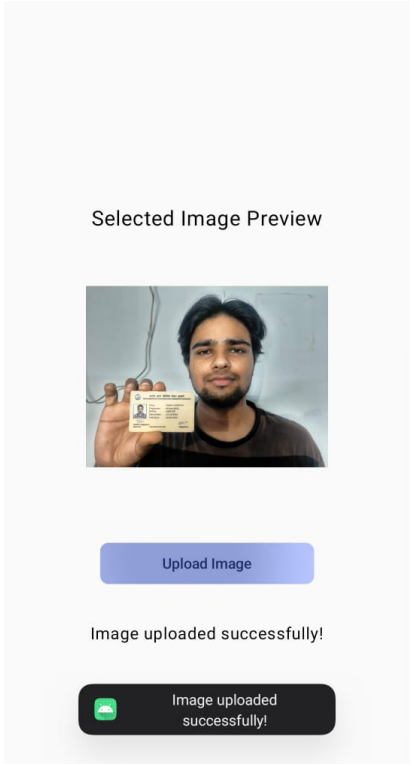
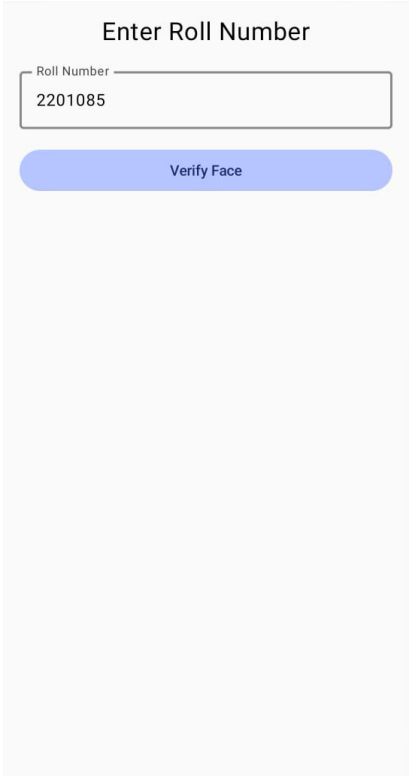


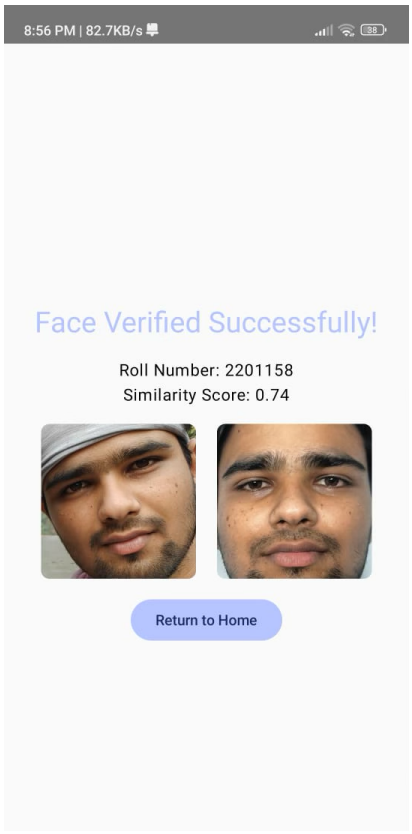
Image Upload Screen:



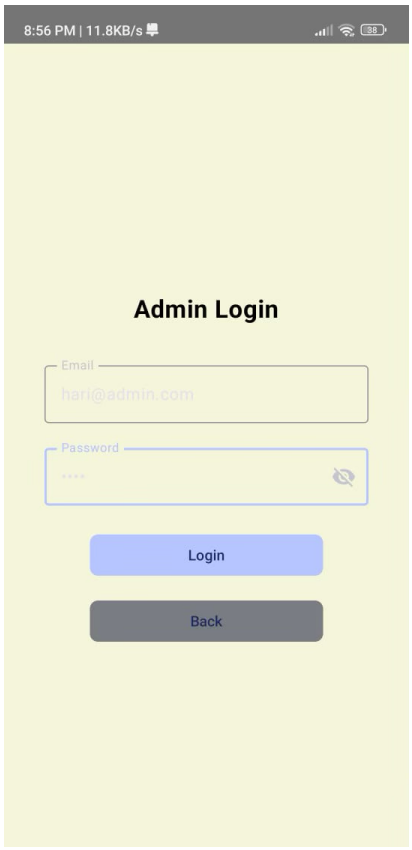
Manual Rollno. Entry:



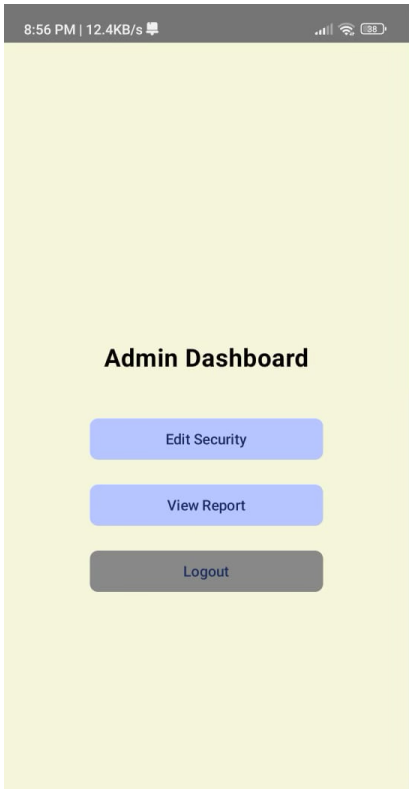
Final Result Screen



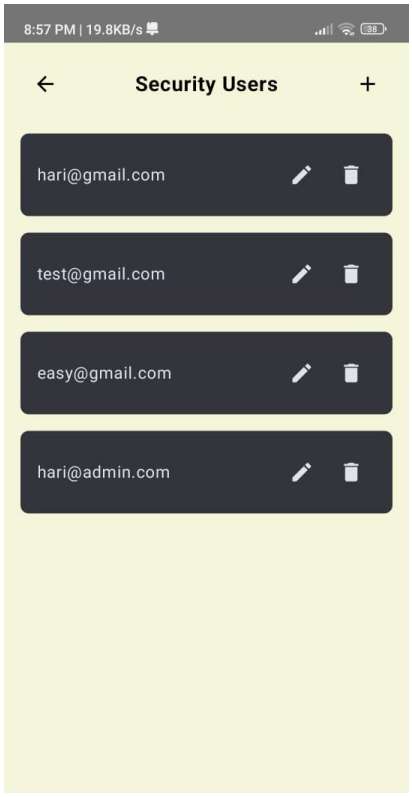
Admin Login



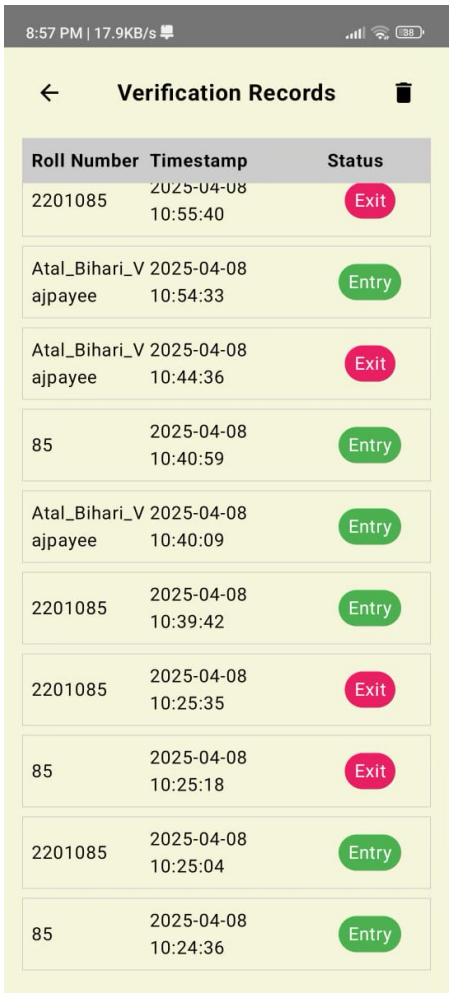
Admin Dashboard



Security Management



Verification Record



Chapter 7 : Future scopes and Conclusion

7.1 Future Scopes

1. Real-time Enhancements:

In the current implementation, verification is image-based. Moving forward, the system can incorporate real-time face recognition through live video streaming. This would allow for continuous monitoring and eliminate the need for manual image capture. Additionally, integrating liveness detection techniques can help prevent spoofing attempts using photos or videos, enhancing the system's overall reliability and security.

2. Scalability & Accessibility:

As the system expands across departments or organisations, transitioning from a local SQLite database to a cloud-based solution would allow for centralised access and better data management. This would also facilitate integration with larger enterprise systems. Moreover, enhancing the OCR module with multi-language support would make the system usable in multilingual environments, increasing its adaptability across different regions and institutions.

3. Smart Monitoring and Analytics:

Implementing automated alert systems for failed or suspicious verification attempts could help enhance administrative control. Additionally, integrating a visual dashboard with graphical reports (e.g., daily entries, peak hours, or frequent users) would provide valuable insights to admins, aiding in better decision-making and operational efficiency. This would also allow for easy trend analysis and auditing of security logs.

7.2 Conclusion

This project marks the culmination of an extensive and enriching journey into the world of machine learning, computer vision, and real-world software development. The primary goal was to create a robust and practical system that facilitates secure and automated entry/exit through facial and ID verification. What began as a simple idea evolved into a fully functional Android application, designed using Jetpack Compose and CameraX, and backed by powerful technologies like FastAPI, SQLite, OpenCV, InsightFace, and Tesseract OCR.

From the backend, we learned to work with model integration, face embedding generation using a pre-trained ArcFace-based buffalo_l model from InsightFace, and roll number extraction using OCR techniques. The frontend development introduced us to responsive UI design and effective navigation flows between camera capture, manual input, and result visualisation. We handled network communication through Retrofit and achieved seamless integration with the backend.

A major milestone of this project was the experimentation phase. We implemented a rigorous testing pipeline using the CALFW dataset, where we computed cosine similarities for genuine and imposter face pairs and used Bayesian thresholding to decide on face verification. This not only validated our approach with empirical evidence but also taught us about performance metrics such as precision, recall, F1-score, TAR, FAR, and more — grounding the solution in data-driven evaluation.

In the process, we explored a wide array of tools, libraries, and algorithms, debugging real-time errors, optimising image handling, and even performing ONNX-based model experimentation for ID detection. The project also emphasised good software practices like modularity, database schema design, and error handling.

More than just technical achievements, this project served as a holistic learning experience — from building an end-to-end system to overcoming practical challenges like image size handling, UI constraints, and performance tuning. It instilled not only a deeper understanding of face recognition systems but also the confidence to tackle full-stack applications that merge AI with real-world usability.

Ultimately, the result is a secure, smart, and automated face-ID verification system that not only works efficiently but is ready to scale and adapt to broader organisational needs.

References

1. Mei Wang a , Weihong Deng School of Artificial Intelligence, Beijing 100876, China “Deep face recognition: A survey” August 2020

2. J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “Arcface: Additive angular margin loss for deep face recognition,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4690–4699.

3. buffalo_l latest face recognition model by InsightFace project is mainly maintained By Jia Guo and Jiankang Deng. https://github.com/deepinsight/insightface/blob/v0.7/README.md

4. A. Dutta, R. Veldhuis and L. Spreeuwers, "A Bayesian model for predicting face recognition performance using image quality," *IEEE International Joint Conference on Biometrics*, Clearwater, FL, USA, 2014, pp. 1-8, doi: 10.1109/BTAS.2014.6996248.

5. T. Zheng, W. Deng, and J. Hu, “Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments,” arXiv preprint arXiv:1708.08197, 2017.

6. De Mel V L B Department of Computer Science and Engineering University of Moratuwa, Sri Lanka “Survey of Evaluation Metrics in Facial Recognition Systems” July 2023

Appendix

1. Testing code on Google Colab : [Code_File]
2. Output Confusion Metrics and Evaluation scores at different thresholds : [Output_File