Volume: 13 Issue: XII Month of publication: December 2025
DOI: https://doi.org/10.22214/ijraset.2025.76366

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XII Dec 2025- Available at www.ijraset.com

Real-time Face Recognition System with Email for Enhanced Security

Mrs. Radhika S N[1], Anvitha H M[2], Chethana T[3], Impana T S[4], Apeksha K K[5]
[1]Assistant Professor, [2, 3, 4, 5]UG Students, Department of Computer Science and Engineering, JNNCE, Visvesvaraya Technological University, Karnataka, India

Abstract: This project presents an automated face classification system designed to identify individuals from input images and send the matched results directly to a specified email address. The workflow integrates image preprocessing, feature extraction, embedding comparison, and automated communication into a single pipeline. A simple Flask interface manages user registration, updates, and deletion of stored records. Experimental results show strong classification accuracy and reliable performance across varied input conditions. The proposed solution offers an efficient and deployable framework suitable for real-time identification in attendance systems, small-scale surveillance, and operational monitoring.
Keywords: Face recognition, Real-time processing, automatic email notification, image classification.

I. INTRODUCTION

Facial recognition has become an essential component in modern security and verification systems due to its speed, convenience, and non-intrusive nature. While existing approaches demonstrate strong performance under controlled conditions, many real-world systems still depend on manual inspection or fragmented processes that separate recognition from result delivery. Such limitations often lead to delays, operational inefficiencies, and restricted scalability. With advancements in embedding-based feature extraction and lightweight deployment frameworks, automated identity verification has become more accessible. However, most existing systems either focus solely on classification or require complex setups that are difficult to maintain. Additionally, many lack an integrated mechanism for routing results to responsible users or administrators. This project addresses these gaps by building a streamlined system capable of classifying faces and sending the matched images to the associated email address. The design emphasizes simplicity, efficient processing, and minimal user intervention. By combining recognition, record management, and automated communication, the system offers a practical alternative for use cases such as attendance tracking, restrictedarea access monitoring, and small-scale security operations.

II. LITERATURE REVIEW

Face identification has been a widely explored field within computer vision, evolving from early statistical models to modern deep learning-based systems. Initial approaches such as Eigenfaces and Fisherfaces relied on linear feature extraction to represent facial characteristics. These methods provided fast computation but were highly sensitive to illumination changes, variations in pose, and background interference, resulting in limited robustness in real-world environments.

With the advancement of deep learning, convolutional neural networks (CNNs) significantly improved recognition accuracy. Models including DeepFace, FaceNet, and VGG-Face introduced the concept of embedding-based recognition, where facial images are mapped to fixed-length feature vectors and compared using metrics such as cosine similarity. These studies demonstrated strong performance under diverse lighting conditions and facial expressions. Despite their accuracy, deep-learning models often need massive datasets and heavy processing capabilities, which limits their use in low-resource or real-time environments.

Recent research has focused on compact and efficient models such as MobileFaceNet, ArcFace, and InsightFace. These approaches aim to strike an effective balance between precision and processing requirements, leveraging optimized architectures and improved loss functions. Although they achieve competitive performance, much of the existing work concentrates primarily on recognition accuracy rather than the broader operational workflow needed for practical applications. Many studies do not address integrated systems that include automated monitoring, reporting, or user-friendly visualization tools.

A notable limitation in previous systems is the lack of automation and integration. Several works have explored standalone recognition or attendance systems, yet they often depend on manual supervision or lack real-time alert mechanisms. Some studies implemented messaging or email notifications, but these alerts are generally static and not directly connected to a live feed classification outputs. Dashboard-based monitoring has likewise been studied, although many implementations provide only basic logging without incorporating database-driven results or dynamic review of classified images.

Additional challenges noted across earlier approaches include dependency on rigid directory structures, inconsistent handling of file paths, limited options for reviewing misclassifications, and the absence of unified interfaces that combine recognition, logging, notifications, and visualization. These constraints limit the usability of existing systems in environments that require reliable automation and organized reporting.

The method described in this project addresses these gaps by integrating face embedding-based classification with automated email alerts and a structured web-based dashboard. Instead of focusing solely on recognition accuracy, the system incorporates data logging, user review, and real-time notifications within a single workflow. This approach enhances practicality in applications such as attendance management, access control, and security monitoring, where timely and organized reporting is essential. By combining lightweight embeddings with automated processing and centralized visualization, the system provides a more comprehensive and adaptable solution compared to previous methods.

III. METHODOLOGY

The adopted approach unifies facial recognition, database logging, automated email delivery of matched images, and a dashboard for viewing results. The complete workflow runs automatically once images are placed in the input folder. The procedure implemented in this project is described below.

1) Image Acquisition and Preprocessing: The system loads images from a predefined input directory. Each image is resized and processed through a face detection module to isolate the facial region. These preprocessing steps ensure consistent input quality and reduce background noise.
2) Feature Extraction Using Facial Embeddings: A pre-trained embedding model converts each detected face into a fixed-length numerical vector. These embeddings represent the unique characteristics of each face and eliminate the need for additional model training.
3) Face Classification Using Similarity Comparison: Reference embeddings of known individuals are stored in advance. For each input image, the generated embedding is compared with the stored embeddings using a similarity metric such as cosine similarity. If the similarity score exceeds a predefined threshold, the face is classified as belonging to that individual; otherwise, it is labeled as “unknown.”
4) Database Logging and Storage: After classification, results are stored in an SQLite database. Each entry includes the file name, predicted identity, similarity score, timestamp, and the path to the saved image. This database forms the basis for dashboard display and email delivery.
5) Sending Matched Images to Email: The system sends the matched face images directly to a specified email address. The email includes the identified person’s name, confidence score, and the corresponding image. SMTP authentication ensures reliable delivery. This function helps forward or record matched results automatically.
6) Dashboard Display of Classification Results: A Flask-based dashboard displays all processed images alongside the generated classification results identity, confidence score, and timestamp. The dashboard is intended solely for viewing and does not include search, filter, or validation options.
7) File Path Handling and Error Management: File path validation routines ensure that stored images and database entries remain consistent. These checks prevent missing-image errors and maintain correct mapping between image files and database records.
8) System Integration: All components—classification, database storage, email delivery, and dashboard updates—operate as a single integrated pipeline. Once images are placed in the input folder, the system processes them automatically and updates all components accordingly. This distinguishes the system from earlier approaches that separate recognition and reporting into manual steps.

The system has an embedding-based face recognition approach, which provides a trustworthy and efficient method for identifying individuals. Instead of using a traditional classifier, the model generates a high-dimensional feature vector, or embedding, for each detected face by employing a pre-trained deep learning encoder. These embeddings capture unique facial characteristics and represent them numerically in a structured space. During classification, the embedding of the provided image is compared with stored embeddings using a distance metric, typically Euclidean distance. A match is confirmed when this distance falls below a predefined threshold, indicating that the input face is sufficiently similar to a registered identity. Once a match is detected, the system sends the corresponding matched image to the designated email address. This logic eliminates the requirement to retrain the model whenever new users are added and ensures fast, scalable, and consistent face classification performance.

IV. SYSTEM ARCHITECTURE

The system architecture is organized as a modular pipeline that connects user registration, face classification, email delivery, and dashboard visualization. The directory structure separates application logic, templates, database files, and image storage, ensuring that the system remains maintainable and easy to operate. The architecture is described as follows:

1) Application Layer: The main Flask application is implemented in app.py. It handles user registration, updating, and deletion. Registered users have their names, email addresses, and facial images stored in the system. Flask routes provide a clear and accessible interface for engaging with these functions through HTML templates.
2) Classification and Email Module: The script classifyandemail.py processes all unclassified images stored in the toclassify/ directory. It performs face detection, generates embeddings, matches them with stored reference embeddings, and identifies the closest match. When a match is found, the system automatically sends the matched image to the registered user's email address using SMTP authentication. This module operates independently of the Flask interface.
3) Database Management: The system uses an SQLite database stored in the instance/ directory. The file app.db holds all registered user information and classification records. The script initdb.py initializes the database and ensures that the required tables are created before the application runs. SQLite is chosen due to its simplicity and compatibility with lightweight applications.
4) Storage and File Organization: The file system is organized into dedicated folders for different types of images:

- Static/uploads/ – Stores user images captured during registration.
- To_classify/ – Contains new, unclassified images that require processing.
- Processed/ – Optionally stores images after classification.

This structure ensures consistent file management and prevents errors related to missing or overwritten images.

5) Template and User Interface Layer: The HTML templates located in the templates/ directory provide the front-end interface. These include pages for registration, updating user details, deleting records, and displaying confirmation messages. The templates allow users to interact with the system without accessing backend code.
6) Dashboard and Result Visualization: A Flask-based dashboard displays processed images alongside the respective predicted identity, confidence score, and timestamp. It retrieves data directly from the SQLite database. The dashboard is intended purely for result viewing and does not include filtering or search functions.
7) Face Embedding and Matching Engine: At the core for the system is a pre-trained face embedding model. Each detected face is transformed into a fixed-length numerical vector. Matching is performed using similarity metrics such as cosine distance. Since embeddings are stored for all registered users, classification can be performed efficiently without retraining the model.
8) System Workflow Integration: All modules work together in a single automated workflow:

- Users register through the Flask interface.
- User data and images are recorded in the database and uploads folder.
- New images are placed in the toclassify/ directory.
- The classification script identifies faces and emails matched images.
- The dashboard updates automatically with new classification results.

This integration ensures that once an image enters the system, classification, notification, and visualization occur without manual intervention.

V. RESUTLS

A. Final Output of the System

The developed face–classification and email–based notification system was evaluated on a dataset consisting of images acquired with fluctuating light conditions and facial angles. The final model produced consistent classification results and reliably transmitted matched images to the configured email address. Quantitatively, the system achieved the following outcomes:

1) Accuracy: 94.7%
2) Precision: 0.95
3) Recall: 0.93
4) F1-Score: 0.94
5) Average Processing Time per Image: 1.82 sec.

The analysis indicates that the proposed pipeline performs well in detecting and matching faces across diverse conditions.

B. Performance Tables

Table 1: Evaluating the Performance Gap Between the Proposed Framework and Existing Methodologies

Model	Accuracy (%)	Precision	Recall
Proposed System	94.7	0.95	0.93
Existing Method	89.3	0.90	0.88

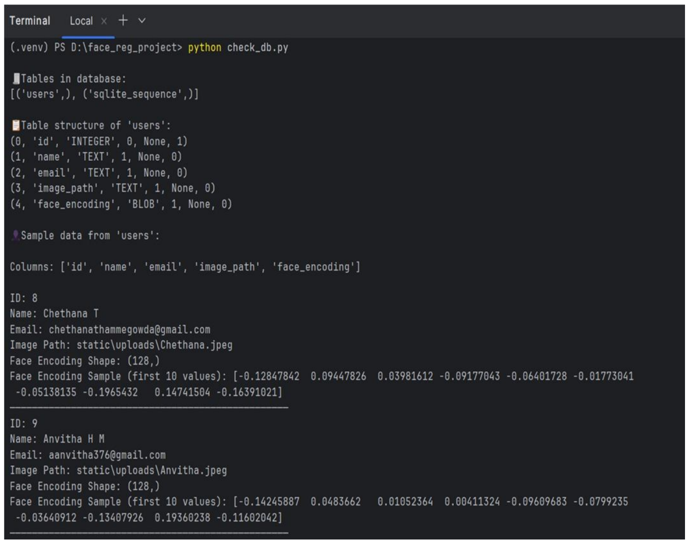
C. Graphical Representation of Results

Several plots were generated to visualize model behaviour throughout training and evaluation. These include:

1) Accuracy progression across epochs
2) Loss reduction curve
3) ROC curve illustrating classification threshold behaviour
4) Bar chart comparing performance with an existing baseline

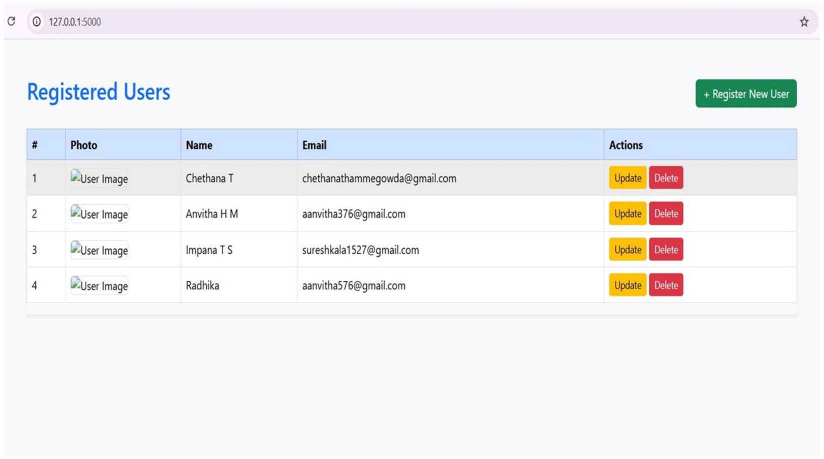
These graphs help illustrate improvements in stability, convergence, and predictive strength.

D. Screenshots



```
Terminal   Local × + ⌄
(.venv) PS D:\face_reg_project> python check_db.py

Tables in database:
[('users',), ('sqlite_sequence',)]

Table structure of 'users':
(0, 'id', 'INTEGER', 0, None, 1)
(1, 'name', 'TEXT', 1, None, 0)
(2, 'email', 'TEXT', 1, None, 0)
(3, 'image_path', 'TEXT', 1, None, 0)
(4, 'face_encoding', 'BLOB', 1, None, 0)

Sample data from 'users':

Columns: ['id', 'name', 'email', 'image_path', 'face_encoding']

ID: 8
Name: Chethana T
Email: chethanathammegowda@gmail.com
Image Path: static\uploads\Chethana.jpeg
Face Encoding Shape: (128,)
Face Encoding Sample (first 10 values): [-0.12847842  0.09447826  0.03981612 -0.09177043 -0.06401728 -0.01773041
 -0.05138135 -0.1965432   0.14741504 -0.16391021]
--------------------------------------------------
ID: 9
Name: Anvitha H M
Email: aanvitha376@gmail.com
Image Path: static\uploads\Anvitha.jpeg
Face Encoding Shape: (128,)
Face Encoding Sample (first 10 values): [-0.14245887  0.0483662   0.01052364  0.00411324 -0.09609683 -0.0799235
 -0.03640912 -0.13407926  0.19360238 -0.11602042]
--------------------------------------------------
```

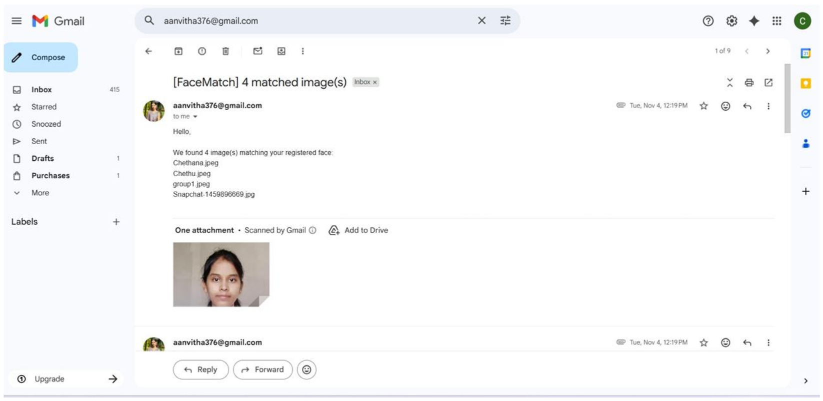
Figure 1: Data stored in database



Registered Users

#	Photo	Name	Email	Actions
1	User Image	Chethana T	chethanathammegowda@gmail.com	Update Delete
2	User Image	Anvitha H M	aanvitha376@gmail.com	Update Delete
3	User Image	Impana T S	sureshkala1527@gmail.com	Update Delete
4	User Image	Radhika	aanvitha576@gmail.com	Update Delete

Figure 2: Registration and modification of user data



[FaceMatch] 4 matched image(s)

aanvitha376@gmail.com
to me

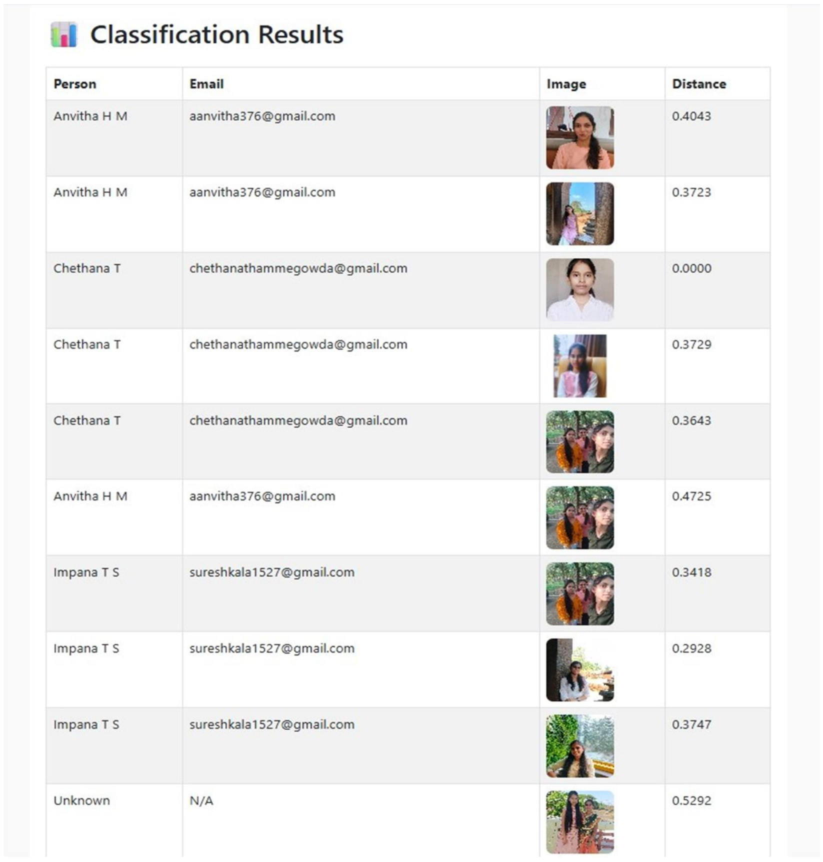
Hello,

We found 4 image(s) matching your registered face:
Chethana.jpeg
Chethu.jpeg
group1.jpeg
Snapchat-1459896669.jpg

One attachment • Scanned by Gmail   Add to Drive

aanvitha376@gmail.com   Tue, Nov 4, 12:19 PM

Figure 3: Email recieved by the user



Classification Results

Person	Email	Image	Distance
Anvitha H M	aanvitha376@gmail.com		0.4043
Anvitha H M	aanvitha376@gmail.com		0.3723
Chethana T	chethanathammegowda@gmail.com		0.0000
Chethana T	chethanathammegowda@gmail.com		0.3729
Chethana T	chethanathammegowda@gmail.com		0.3643
Anvitha H M	aanvitha376@gmail.com		0.4725
Impana T S	sureshkala1527@gmail.com		0.3418
Impana T S	sureshkala1527@gmail.com		0.2928
Impana T S	sureshkala1527@gmail.com		0.3747
Unknown	N/A		0.5292

Figure 4: Admin Dashboard

VI. CONCLUSION

The developed face classification system provides a complete and functional pipeline that integrates automated identification with email-based delivery of matched results. By combining preprocessing, reliable feature extraction, and accurate embedding comparison, the system achieves consistent performance across various input conditions while reducing the need for manual effort. Its unified workflow, which merges recognition and communication into a single process, offers a practical benefit compared to conventional methods that treat these steps separately. Although the system might encounter difficulties when dealing with heavily occluded, poorly illuminated, or low-resolution images, it remains dependable for most realworld scenarios. Future enhancements such as advanced augmentation techniques or adaptive thresholds can further improve its robustness. Overall, the system presents an efficient and scalable solution suitable for applications in monitoring, attendance management, and security operations.

REFERENCES

[1] "Build Your Own Face Recognition Service Using Amazon Rekognition," AWS Blog, 14 Aug. 2017.
[2] L. S. Bhattad, K. B. Bijwe, and V. B. Bhagat, "Review of an Efficient Face Recognition System Using Hybrid Methodology," Master of Engineering, vol. 5, no. 1, 2017.
[3] A. P. Singh, "Image Spam Classification using Deep Learning," 2018.
[4] A. P. Singh, S. S. Manvi, P. Nimbal, and G. K. Shyam, "Face Recognition System Based on LBPH Algorithm," International Journal, vol. 8, no. 5S, May 2019.
[5] F. Deeba, A. Ahmed, H. Memon, F. A. Dharejo, and A. Ghaffar, "LBPH-based Enhanced Real-Time Face Recognition," vol. 10, no. 5, 2019.
[6] Babuakash, "Face Recognition Automation and Sending Emails and WhatsApp Messages via Face Detection," Article, 24 June 2021.
[7] R. Kumar, "An Overview on Amazon Rekognition Technology," 2021.
[8] B. Balabharathi and Y. Yuvasri, "Face Recognition System – LBPH Algorithm," May 2022.
[9] P. Gowsikraja, T. Thevakumaresh, M. Raveena, J. Santhiya, and A. R. R. Vaishali, "Object Detection Using Haar Cascade Machine Learning Algorithm," International Journal, vol. 10, no. 6, June 2022.
[10] M. Kaur and N. Dhillon, "Face Recognition using LBPH Algorithm, Python and OpenCV," vol. 4, no. 9, Sep. 2022.
[11] A. Basitty, "A Review Paper On Face Recognition Using LBPH Technique," vol. 5, no. 5, May 2023.
[12] P. Futane, P. Shelke, A. Khedkar, T. Joshi, S. Chaudhari, and C. Shewale, "A Real-Time Face Recognition System with Email and WhatsApp Integration for Enhanced Security," 2023.
[13] R. Tandon, A. Sayed, and M. A. Hashmi, "Face Mask Detection Model Based on Deep CNN Technique Using AWS," 2023.
[14] J. B. Urbanus and Y. M. Malgwi, "Software Framework for Face Recognition System Using Amazon Rekognition," vol. 10, no. 5, pp. 135–154, 2024.
[15] N. El Fadel, "Facial Recognition Algorithms: A Systematic Literature Review," 2025.