International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 14, Issue 9 [September 2025] PP: 124-127

Face Finder

MR.SUBRAMANIAM E[*1], SUBHASRI KT[*2], SUJITH KUMAR S[*3],VIGNESH E[*4], SUGANTHAN S [*5], VIJAYA SRI KESAVAN [*6]

[*1]*Assisstant Professor, Sri Shakthi institute of engineering and technology, Coimbatore, Tamil Nadu, India*
[*2]*Student , Sri Shakthi institute of engineering and technology, Coimbatore, Tamil Nadu, India*
[*3]*Student , Sri Shakthi institute of engineering and technology, Coimbatore, Tamil Nadu, India*
[*4]*Student , Sri Shakthi institute of engineering and technology, Coimbatore, Tamil Nadu, India*
[*5]*Student , Sri Shakthi institute of engineering and technology, Coimbatore, Tamil Nadu, India*
[*6]*Student , Sri Shakthi institute of engineering and technology, Coimbatore, Tamil Nadu, India*

Abstract
A modern innovation, facial recognition technology enables fast and secure identification in diverse applications. When integrated into student information systems, it simplifies the retrieval of academic and personal data. This research outlines the development of a mobile-based facial recognition system using deep learning and image processing. Educational access can be improved by integrating intelligent verification tools. The system increases accuracy and reduces manual lookup efforts while supporting smarter academic management. As facial recognition in education is still evolving, its use for personalized student data access remains underexplored.
Keywords: *Facial recognition, student data, automation, mobile integration*

Date of Submission: 15-09-2025 Date of acceptance: 30-09-2025

I. INTRODUCTION

Face Finder is a mobile-based facial recognition system designed to identify students in real-time using advanced deep learning techniques. It transforms captured facial data into verifiable digital patterns using Siamese neural networks. By leveraging computer vision, machine learning, and mobile technology, the system accurately matches a scanned face with the existing student database. Through an intuitive Flutter interface and a Python-based backend, Face Finder enables seamless access to academic and personal records. This innovative approach enhances data retrieval, security, and management in educational settings, offering a dynamic tool for modern academic environments.

II. METHODOLOGY

1.Core Functionalities

1. **Facial Recognition**: Capture and identify student faces using the device camera and a pre-trained deep learning model.
2. **Student Data Retrieval**: Upon successful face match, fetch and display corresponding academic and personal information from the database.

2.System Architecture

1.Input Layer: Mobile device camera captures a live face image for processing.
2.Processing Layer:

- **Face Detection Module**: Detects facial features in real-time using computer vision libraries.
- **Feature Extraction Module**: Converts the detected face into an embedding using a Siamese Neural Network.
- **Matching Module**: Compares the embedding with the database to find the closest match.

3.Output Layer: Displays the matched student profile including name, ID, academic details, and photo on the app screen.

3.Key Components

1.Facial Recognition: Capture and identify student faces using the device camera and a pre-trained deep learning model.

2.Student Data Retrieval: Upon successful face match, fetch and display corresponding academic and personal information from the database.

III. MODELING AND ANALYSIS

1. Facial Data Collection and Model training

- **Dataset preparation**: Collect and preprocess a diverse dataset of student face images under varying lighting and angles to improve model generalization. Each image is aligned and resized to maintain consistency.
- **Feature encoding**: Utilize a Siamese Neural Network to learn feature embeddings that distinguish between different identities using contrastive or triplet loss.
- **Model training**: Train the network using TensorFlow, monitoring accuracy and loss over epochs. Employ data augmentation techniques to increase robustness and avoid overfitting.

2. Face capture and preprocessing

- **Live image capture**: Use the device's front-facing camera to capture real-time images of a user's face for recognition.
- **Image Normalization**: Apply grayscale conversion, histogram equalization, and resizing to match the input dimensions expected by the trained model.
- **Embedding Extraction**: Pass the preprocessed image through the trained network to generate a fixed-size feature vector representing the face.

3. Backend Integration with Flask

- **API Development**

1. Create RESTful endpoints using Flask to accept face image uploads from the Flutter frontend.
2. Load the trained model and student face embeddings into memory for real-time processing.

- **Matching Algorithm**:

1. Compute cosine similarity between the input embedding and stored embeddings.
2. If similarity exceeds a predefined threshold, return the corresponding student information from the database.

4. Application Build and Deployment

- **Frontend Integration**

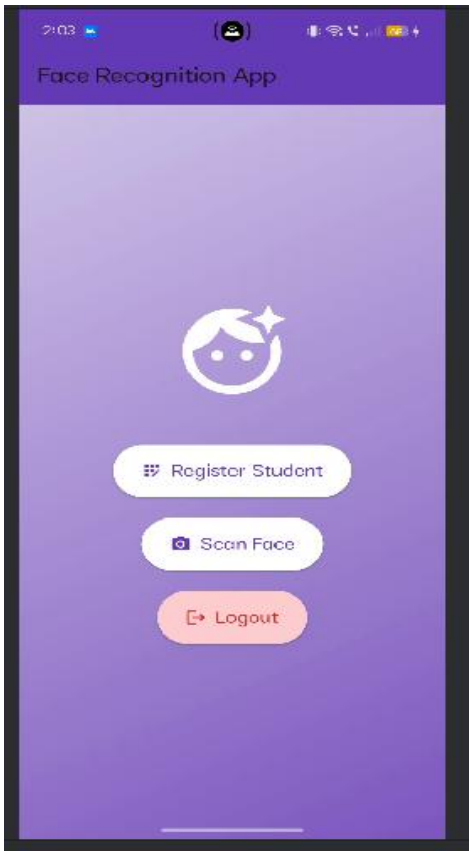
1. Implement a Flutter-based mobile app to access the front camera and capture face images.
2. Use http.MultipartRequest to send images to the Flask server securely.

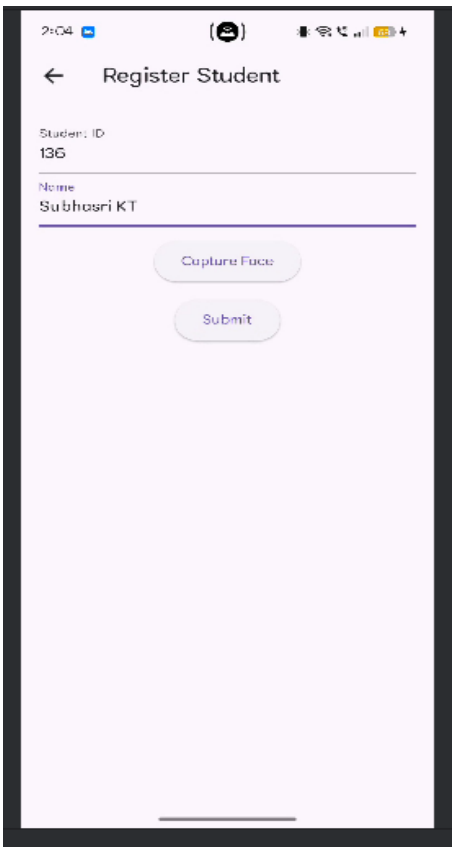
- **System Testing**:

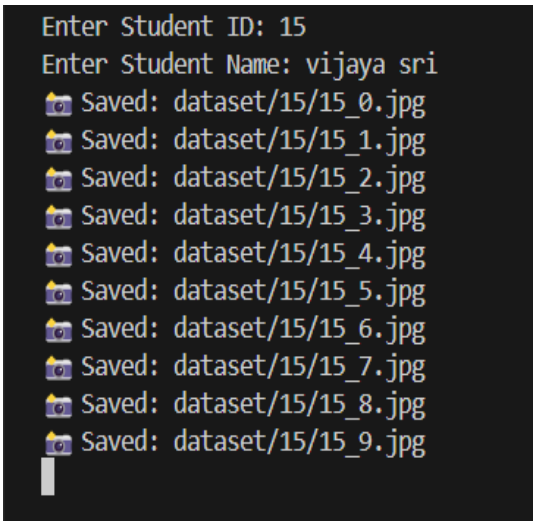
Test the system across devices in various lighting conditions to validate recognition accuracy. Verify that matched results are timely and student data is reliably displayed upon identification.

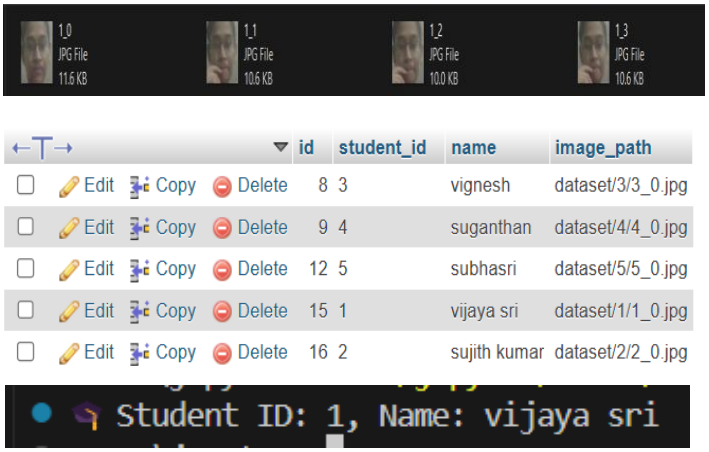
IV. RESULTS AND DISCUSSION

The implementation of the Face Finder project using Flutter, Python (Flask), and a Siamese Neural Network successfully met its objective of enabling real-time facial recognition and student data retrieval. The system exhibited high accuracy in identifying known faces from a pre-trained dataset, maintaining consistent performance under varying facial angles and moderate lighting conditions. The facial embeddings generated by the model provided reliable matching, and the Flask API processed and returned student data with minimal latency. The user interface, built in Flutter, allowed for efficient image capture and a responsive experience across supported Android devices. End-to-end testing showed smooth communication between the mobile app and backend server, with recognition results displayed in under two seconds on average. The system's performance was stable across modern devices, although slightly degraded on older phones due to limited computational power and memory. Some challenges were observed in distinguishing faces with subtle differences or under very low lighting, which impacted recognition confidence. These can be improved with expanded training data and better preprocessing techniques such as adaptive histogram equalization. The intuitive design and reliable functionality of the application make it valuable for secure student verification in educational environments. Future extensions may include adding face enrollment, attendance tracking, and integration with cloud databases to enhance scalability. Overall, the project delivered a functional, efficient, and extensible face recognition solution tailored for real-world academic applications.









V. CONCLUSION

In conclusion, the Face Finder project combining Flutter, Flask, and a Siamese Neural Network effectively demonstrated the practical use of facial recognition for secure student identification. The system reliably captured facial data through a mobile interface, processed it using deep learning, and retrieved associated student details with speed and precision. Its successful deployment highlights the potential of AI and mobile integration for real-time verification in academic and institutional settings. The application proved responsive and accurate under normal conditions, offering a seamless user experience. However, limitations such as dependency on high-quality training data, device performance, and varying lighting conditions indicate opportunities for enhancement. Future work may focus on expanding the training dataset to improve recognition accuracy, integrating liveness detection to prevent spoofing, and supporting multi-platform compatibility. Moreover, adding features like face-based attendance tracking and cloud storage could broaden its utility in educational administration. Introducing offline recognition capabilities and optimizing the model for edge devices would

further extend its accessibility. Overall, the Face Finder system lays a strong foundation for a robust, scalable, and user-friendly facial recognition solution, with promising applications in security, education, and beyond.

REFERENCES

[1] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
[2] Chollet, F. (2018). *Deep Learning with Python*. Manning Publications.
[3] Brownlee, J. (2019). *Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python*. Machine Learning Mastery.
[4] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.
[5] Rosebrock, A. (2021). *Face Recognition with Python: Build Facial Recognition Systems Using Deep Learning*. PyImageSearch.
[6] Patel, R. (2020). *Real-Time Face Recognition Using Python and OpenCV*. Independent Publishing.