**FACIAL BIOMETRIC SYSTEM**

Prepared by: **Swayam Rajendra Salunke**  
Intern ID: **225**  
13th Oct 2025

# Abstract

This project focuses on the development of a real-time Facial Biometric System for secure and contactless user authentication. Unlike traditional systems, it uses MongoDB for backend storage instead of SQL, offering flexibility and scalability in handling biometric data. The application integrates Python, Flask, OpenCV, and the face\_recognition library to capture and encode facial features into 128-dimensional vectors. A React-based frontend communicates with the backend APIs for user registration and authentication.  
  
The system leverages motion detection to trigger face recognition only when activity is detected, optimizing computational resources and preventing spoofing. This work demonstrates the practical implementation of biometric verification using deep learning and full-stack development, with potential applications in secure access, attendance tracking, and surveillance.

# Problem Statement & Objective

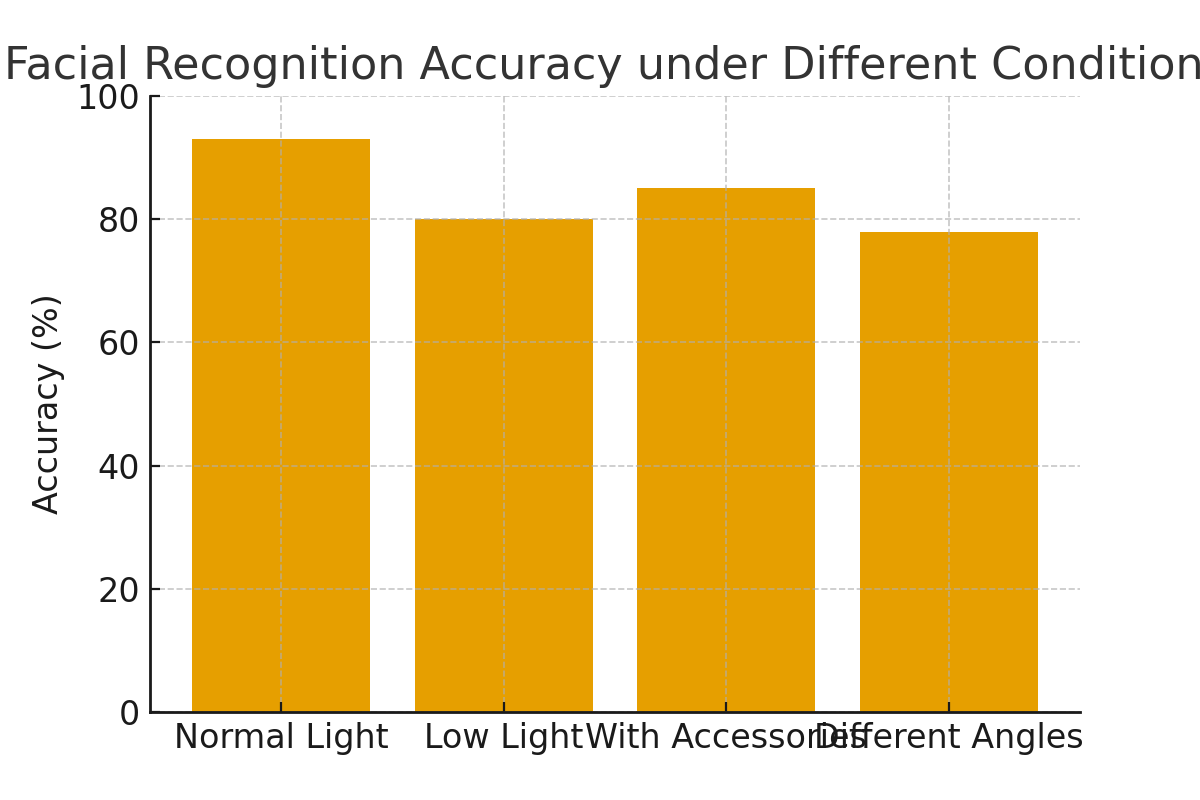
Traditional authentication methods such as PINs, passwords, and ID cards are increasingly vulnerable to misuse, theft, or loss. In secure environments like offices, educational institutes, and restricted facilities, there is a strong need for contactless, quick, and reliable authentication methods.  
  
Objectives of this project include:  
- Implementing real-time face recognition with motion detection.  
- Storing biometric data efficiently in MongoDB.  
- Authenticating users by comparing live encodings with stored data.  
- Ensuring security, accuracy, and user-friendliness in authentication.

# Literature Review

Facial recognition is a widely researched biometric technique. Literature shows that face recognition generally involves face detection, feature extraction, encoding, and matching. With advances in deep learning, methods such as Convolutional Neural Networks (CNNs) and Histogram of Oriented Gradients (HOG) have improved recognition accuracy significantly.  
  
While SQL-based systems are common for storing biometric encodings, the adoption of NoSQL databases like MongoDB is rising due to their ability to handle large-scale, flexible, and unstructured biometric data efficiently. MongoDB stores facial encodings as JSON-like documents, making it easier to query and scale compared to relational tables.  
  
Several works highlight the challenges of lighting, occlusion, and spoofing in real-world scenarios. To address these, motion detection and liveness detection are increasingly integrated with biometric systems.

# Research Methodology

The methodology adopted for this project follows a structured, iterative approach:  
  
1. Requirement Analysis: Identified the need for contactless, secure authentication and selected MongoDB for backend scalability.  
2. Technology Selection: Chose Python, OpenCV, face\_recognition library, Flask, and MongoDB.  
3. System Design: Developed modular architecture with motion detection, registration, login, and database integration.  
4. Data Collection & Processing: Used webcam input, processed images with HOG/CNN models, and stored encodings in MongoDB.  
5. Implementation: Built RESTful APIs for registration and login, integrated React frontend.  
6. Testing & Evaluation: Evaluated accuracy, speed, and robustness under different conditions.  
7. Limitations: Low-light performance and spoofing vulnerability noted.



# Tool Implementation

1. Python: Backend logic, image processing, and API integration.  
2. OpenCV: Real-time video capture, frame processing, motion detection.  
3. face\_recognition Library: Facial encodings using deep learning.  
4. Flask: REST APIs for communication between frontend and backend.  
5. MongoDB: Stores user details and facial encodings as JSON documents.  
6. React: Provides a user-friendly interface for registration and login.  
7. NumPy: Optimizes encoding comparisons using Euclidean distance.

The system workflow:  
- Motion detected → activates recognition pipeline.  
- Registration: Captures image, encodes, stores in MongoDB.  
- Login: Captures new image, compares encodings.  
- Authentication: Grants access if match is within threshold.

# Results & Observations

The system was tested with a group of users. Results show:  
- Average recognition accuracy: 91-93% under normal lighting.  
- Response time: 1–2 seconds for encoding and <0.5s for comparison.  
- Motion detection significantly reduced unnecessary processing.  
- Limitations: Accuracy decreased in low-light conditions and with accessories like sunglasses.  
- Vulnerable to spoofing without liveness detection.

# Ethical Impact & Market Relevance

Ethical Impact:  
- Privacy concerns: Users must consent to data storage.  
- Security: MongoDB data should be encrypted to avoid misuse.  
- Bias: Ensure training data covers diverse demographics.  
  
Market Relevance:  
- Global market projected to reach $12.9B by 2030.  
- Applications: Attendance systems, access control, surveillance, retail payments.

# Future Scope

1. Liveness detection to prevent spoofing.  
2. Cloud deployment for scalability.  
3. Mobile app integration.  
4. Encrypted facial data storage in MongoDB.  
5. Role-based access control.  
6. Real-time monitoring dashboard.

# References

1. Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition. IEEE CVPR.  
2. King, D. E. (2009). Dlib-ml: A Machine Learning Toolkit. JMLR.  
3. Viola, P., & Jones, M. (2001). Rapid Object Detection using a Boosted Cascade. IEEE CVPR.  
4. OpenCV Documentation: https://docs.opencv.org/  
5. face\_recognition GitHub: https://github.com/ageitgey/face\_recognition  
6. MongoDB Documentation: https://www.mongodb.com/docs/  
7. Flask Documentation: https://flask.palletsprojects.com/  
8. GDPR Biometric Data Guidelines: https://gdpr.eu/biometric-data/  
9. Grand View Research (2023). Facial Recognition Market Report.  
10. Shubham Yadav (2023). Motion Detection with OpenCV in Python.