Tunis Business School

End-of-Semester Project

Field of Study: **Information Technology**
Specialization: **Web Service Using Flask**

Topic

Web-Based Facial Authentication System: Theoretical Foundations

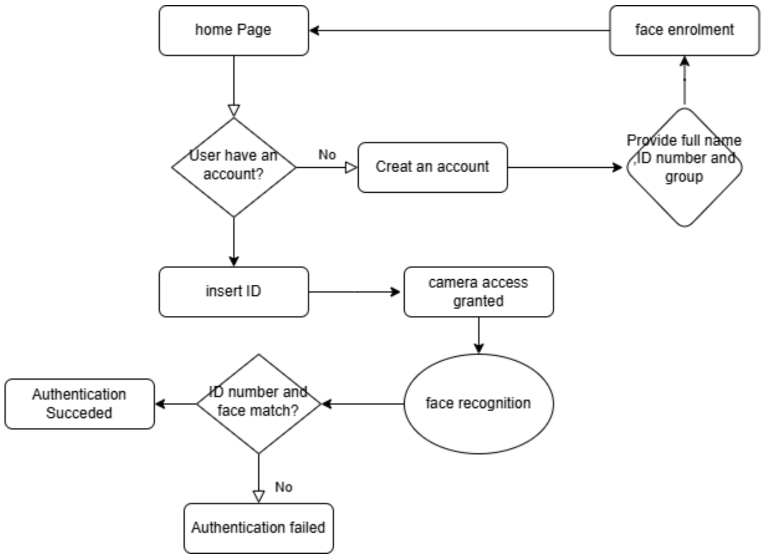
---

Presented by
Sinen Frej
Defended on: **May, 2025**

Academic Year: 2024/2025

1 System Architecture

The proposed system operates on a web-based architecture consisting of three core components: the client browser, a backend server, and a FAISS-based vector database. The client captures real-time video using the camera, sending frames to the server for processing. The server detects faces, generates embeddings, and queries the FAISS database to identify the student. The final authentication decision is then sent back to the client.



2 System Components

On the client side, a camera interface captures video at 15 frames per second. Frames are sent to the backend server through a secure WebSocket connection. This high frame rate ensures that detection and liveness verification (such as blinking) remain accurate.

The backend is implemented using Flask and integrates multiple services. YOLO is used for face detection, and for efficiency, input frames are resized to 50% of their original size before processing using OpenCV. The system uses Python's threading module to process multiple frames concurrently and reduce latency. After detecting the face, InsightFace is used to extract a 512-dimensional facial embedding.

3 Authentication Workflow

The system includes two primary phases: registration and verification. During registration, the student provides five reference images. These are embedded and stored in the FAISS vector database under a unique student identifier.

In the verification phase, multiple frames are captured and processed in real time. The student's ID card number is submitted, and the system checks whether the provided facial embedding matches the stored one associated with the same ID. Matching is done by calculating the distance between embeddings; if the closest match is within a threshold of 1, the identity is confirmed. A majority voting mechanism ensures robustness, and liveness detection helps prevent spoofing.

4 Development Tools

Phase	Tools
1. Prototyping	Python, OpenCV
2. Web Integration	Flask, JavaScript,html ,css
3. Database Setup	FAISS,Pickle dictionary
5. Performance Tuning	Threading, Frame Scaling

5 Data Exchange Flow

Frames are transmitted from the client to the server in JPEG format at 640×480 resolution. The server reduces the resolution before running detection, producing a 512-dimensional float32 embedding. This vector is queried against the FAISS database, which returns a list of candidate matches sorted by distance. A match is accepted if the smallest distance is less than the threshold of 1. Finally, the server returns a JSON object with the result and confidence score to the client interface.

The ID entered by the student is cross-validated using a dictionary stored in an `id_to_info.pkl` file, ensuring both biometric and credential match.

6 Front end

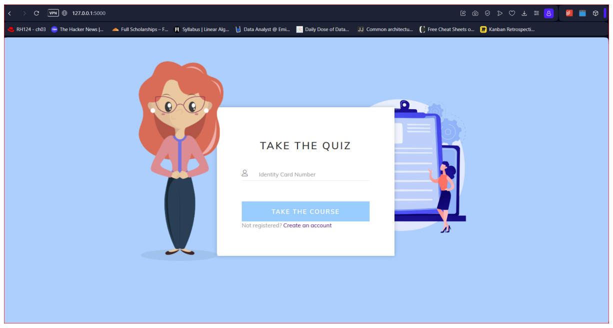


Figure 1: Home page

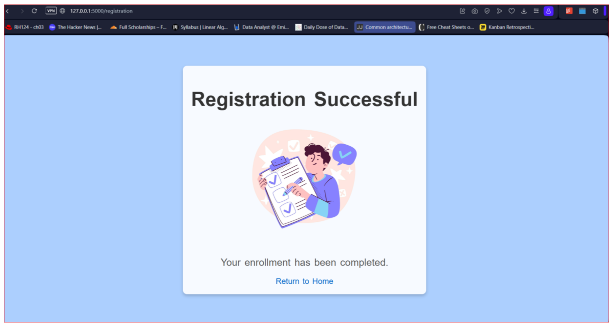

127.0.0.1:5000/registration
Registration Successful
Your enrollment has been completed.
Return to Home

Figure 2: Student has been registered successfully

7 Bibliography

You can find the project source code on GitHub at this repository.

References

[1] Redmon, J. *et al.* (2016). "You Only Look Once: Unified, Real-Time Object Detection". CVPR.

[2] Deng, J. *et al.* (2019). "ArcFace: Additive Angular Margin Loss for Deep Face Recognition". CVPR.

[3] Johnson, J. *et al.* (2017). "Billion-scale similarity search with GPUs". arXiv:1702.08734.