Système de Reconnaissance Faciale pour la Validation Sécurisée des Tickets

Projet Académique

1) Descriptif du Projet

Titre du projet

Système de Reconnaissance Faciale pour la Validation Sécurisée des Tickets Projet Académique

Résumé

Ce projet consiste à développer un prototype capable de lier un ticket numérique à l'identité faciale d'un utilisateur, afin daugmenter la sécurité et de fluidifier laccès à un événement (exemple : Coupe du Monde 2030). Tout le système fonctionne en local, avec les photos des membres de léquipe uniquement, sans stockage externe ni usage réel.

Objectifs du projet

— Capturer une image du visage via webcam ou upload.
— Générer un embedding facial (FaceNet, ArcFace).
— Associer embedding ticket.
— Reconnaître le visage à lentrée via webcam.
— Valider ou refuser automatiquement le ticket.

Objectifs pédagogiques

— Comprendre les techniques modernes de reconnaissance faciale.
— Tester différents modèles IA (ArcFace, FaceNet, Dlib).
— Concevoir une architecture simple (frontend + backend + base locale).
— Développer un prototype fonctionnel.
— Étudier les limites : précision, anti-spoofing, éclairement, distance...

2) Étude Comparative Solutions du Marché

Conclusion

Pour un projet académique :
— ArcFace ou FaceNet reconnaissance précise.
— OpenCV détection des visages.
— Exécution 100% locale pour éviter les risques légaux.

Solution	Avantages	Inconvénients	Adaptation Projet Académique
Amazon Rekognition	Très précis, API simple, anti-spoofing	Payant, stockage cloud obligatoire	Non adapté
Microsoft Face API	Haute précision, modèles avancés	Données envoyées hors local	Non adapté
Face++	Rapide, bon SDK	Cloud chinois, confidentialité limitée	Non adapté
OpenCV + LBPH	Simple, open-source, local	Peu précis, sensible à léclairage	Test basique
Dlib (HOG + SVM)	Très simple à implémenter	Moins performant	Démonstration académique
FaceNet / ArcFace	Haute précision, embeddings efficaces	Plus complexe, idéal GPU	Meilleur choix

4) Ressources Documentation

Reconnaissance Faciale

— ArcFace : https://github.com/deepinsight/insightface
— FaceNet : https://github.com/davidsandberg/facenet
— Dlib : http://dlib.net/face_recognition.py.html
— OpenCV Documentation : https://docs.opencv.org/

Tickets / QR Codes

— Python qrcode : https://pypi.org/project/qrcode/
— ReportLab (PDF) : https://www.reportlab.com/docs/

Bases de Données Locales

— SQLite Documentation : https://www.sqlite.org/docs.html

Frameworks

— Backend : FastAPI / Flask
— Frontend : React ou HTML + JS (getUserMedia)
— Napkin AI pour schématisation
— Overleaf pour le rapport

5) Cahier des Charges (Version Académique)

1. Objectif du projet

— Capturer la photo de lutilisateur.
— Générer lempreinte faciale (embedding).
— Associer embedding ticket.
— Vérifier le visage via caméra locale.

— Valider/refuser automatiquement.
— Aucune donnée envoyée en ligne.

2. Fonctionnalités attendues

1) Côté Utilisateur (Frontend)
— Interface simple.
— Capture du visage.
— Liveness check simple (mouvement / clignement).
— Génération du ticket (PDF ou numérique).
2) Côté Backend
— Détection visage (OpenCV).
— Extraction embedding (FaceNet / ArcFace).
— Stockage local SQLite.
— API REST :
  — /enroll
  — /verify
  — /generate_ticket
3) Système de Vérification (Entrée du Stade)
— Capture en temps réel.
— Comparaison des embeddings.
— Score de similarité.
— Décision : autoriser ou refuser.

3. Contraintes

— 100% local.
— Aucune donnée stockée après les tests.
— Pas de cloud.
— Compatible webcam standard.
— Latence < 1 seconde.

4. Technologies proposées

— Python / FastAPI
— OpenCV
— ArcFace / FaceNet
— SQLite
— React ou HTML+JS
— Napkin AI / Overleaf

5. Livrables

— Diagramme darchitecture
— Cahier des charges
— Rapport technique
— Prototype local fonctionnel
— Démonstration (vidéo ou live)