Blue text on a white background

Description automatically generated

**FaceSphere**

**Real-Time Face Recognition**

**Arzu Selin Yaşar**  
Computer Engineering  
Istinye University  
Istanbul   
Turkiye  
200722079@stu.istinye.edu.tr

**Rawan Z. M. Shat**  
Biomedical Engineering  
Istinye University  
Istanbul  
Turkiye  
210408214@stu.istinye.edu.tr

**Emir YILMAZ**  
Computer Engineering  
Istinye University  
Istanbul  
Turkiye  
200722094@stu.istinye.edu.tr

**İbrahim İsmail ONAY**  
Electric and Electronic Engineering  
Istinye University  
Istanbul  
Turkiye  
210408214@stu.istinye.edu.tr

**ABSTRACT**

FaceSphere is a service that uses Python and Flask to do instant face identification for secure access. Users must access it via a web interface that checks for face feed all through. For any detected face, comparison is done between the webcam photo and the database images of authenticated individuals. Therefore, anyone identified receives instructions on rotating his head to the left then right as an authenticity measure. By following the above steps, a successful verification results login access and storing of the entry time in the database; among other features like addition and removal of users, as well as logging and taking pictures about unauthorized access attempts.

**Keywords:** Real-time face recognition, anti-spoofing, access control, Python, Flask

**1. Introduction**

Facial recognition technology has been rampantly used in many different security applications although there are chances that it can be spoofed hence the necessity of integrating anti-spoofing measures. To handle the above challenges, FaceSphere has recreated a secure, real-time facial recognition system backed by internal anti-spoofing techniques, which guarantees strong and dependable access control.

**2. System Design**

The FaceSphere application utilizes a combination of Python, Flask, SQL Alchemy, Sqlite, OpenCV, and the DeepFace library to create an efficient face recognition system. The front-end interface is developed using HTML, CSS, and JavaScript, providing a user-friendly experience.

**2.1 Architecture**

The system three-tier system consists of a user interface which causes continuously captures video from a webcam; when a face is detected, it is compared against a pre-trained database of known faces. Once this has succeeded, the quickly requests user’s head to move leftwards then rightwards; this will confirm whether the detected face is consistent with other data previously captured.

**2.2 Database Management**

We employ an SQLite with SQLAlchemy database to store images of the user and log in times. This light database solution ensures the fastest possible data clearing and writing operations – it is vital in real-time applications.

**3. Methodology**

**3.1 Data Collection and Training**

Face recognition requires that user images are collected and stored in the database. The model trains so that it learns to recognize and tell apart different faces accurately across various lighting and viewing angles.

**3.2 Anti-Spoofing Measures**

The anti-spoofing measures work by moving your head. To prevent spoofing attempts using photographs or videos, turn your head to the left and then to the right while the system monitors the captured images for any inconsistencies.

**4. Implementation**

**4.1 Real-Time Face Recognition**

The component functioning with face recognition by time use OpenCV and the library known as DeepFace to capture face names from a camera feed automatically. It continuously checks the video stream; this guarantees quick identification and reaction.

**4.2 User Management**

Users can be included or deleted into the system via specific button from the interface. This allows the storage of the facial data of each new user so that recognition can take place later.

**5. Results**

When we tried the FaceSphere service extensively, we found out that it is highly precise in recognizing faces and is very good at stopping unauthorized people from entering. The software identified users who were registered with it and took their admission times, at the same time ensuring that no one could falsify data.

**6. Conclusion**

With integrated anti-spoofing measures, you can be assured that FaceSphere offers a very safe and accurate face recognition system. This software could be so useful for security improvement in different places based on the way it works when it comes to real-world scenarios. One way of expanding scalability is integrating extra security measures for any chance of improved performance.

**References**

1. **OpenCV for Face Detection and Recognition**

* **Title:** "Real-time Face Detection and Tracking Using OpenCV"
* **Authors:** C. R. Gour, S. Kumar
* **Publication:** International Journal of Computer Applications
* **Year:** 2010

1. **dlib for Facial Landmark Detection**

* **Title:** "One Millisecond Face Alignment with an Ensemble of Regression Trees"
* **Authors:** Vahid Kazemi, Josephine Sullivan
* **Publication:** IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
* **Year:** 2014

1. **Deep Learning for Face Recognition**

* **Title:** "DeepFace: Closing the Gap to Human-Level Performance in Face Verification"
* **Authors:** Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, Lior Wolf
* **Publication:** IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
* **Year:** 2014

1. **Flask for Web Development**

* **Title:** "Building Web Applications with Flask"
* **Authors:** Miguel Grinberg
* **Publication:** O'Reilly Media
* **Year:** 2018

1. **SQLAlchemy for Database Management**

* **Title:** "Mastering Flask Web Development"
* **Authors:** Daniel Gaspar
* **Publication:** Packt Publishing
* **Year:** 2018

1. **Face Verification with Deep Learning Models**

* **Title:** "FaceNet: A Unified Embedding for Face Recognition and Clustering"
* **Authors:** Florian Schroff, Dmitry Kalenichenko, James Philbin
* **Publication:** IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
* **Year:** 2015