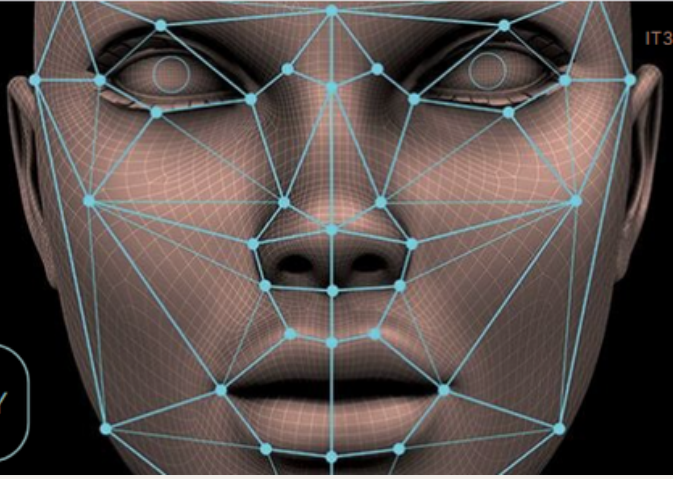
21/03/2024

IT360: INFORMATION INSURANCE AND SECURITY

Journey To The Future



FOR VERIFIED USERS ONLY

Introducing

3 E Y E S

Web-Based Facial Authentication System

Realized by:

Sarra Ibn El Haj - Ju.Fin/IT

Becher Zribi- Ju.Fin/IT

Amine Maaloul - Ju.BA/IT

Raouf Lakhoues - Ju.BA/IT

Tools

1- User Interface (UI) Development:

Templating Engine:

- **Flask Templates:** Flask offers its own templating engine for defining UI structure and integrating Python logic. It's a lightweight and flexible option that works well.

Front-End Frameworks:

- **Bootstrap:** This remains an excellent choice for building responsive and modern UIs with Flask. Bootstrap offers pre-built components, styles, and JavaScript functionalities that can significantly streamline our development process. It integrates seamlessly with Flask templates.

JavaScript Libraries:

- **WebSockets:** Since we require real-time communication between the UI and backend (e.g., photo capture), we will consider using the websockets library. It enables bi-directional communication for functionalities like sending capture commands to the backend and receiving captured images in real-time.

Tools

2-Facial Recognition:

This task will be performed using one, or a combination, of these libraries, depending on the solution we will judge to be the best.

- **cv2.CascadeClassifier:** This module provides functionalities for loading and using pre-trained Haar cascade classifiers for face detection. It allows us to load a pre-trained Haar cascade model from a .xml file, detect faces within an image using the loaded classifier, obtain a list of bounding boxes around the detected faces and explore different pre-trained models available in OpenCV to find the best balance between accuracy and speed for our specific use case.
- **cv2.face:** This module offers functionalities for various facial recognition tasks, although it doesn't provide built-in recognition algorithms. However, it includes tools for **Feature Extraction** like cv2.eigenfaces.EigenFaces or cv2.LBPHFaceRecognizer can be used to implement techniques like Eigenfaces or Local Binary Patterns Histograms (LBPH) for extracting facial features from detected faces and **Training and Saving Models**
- **cv2.imgproc:** This module provides functionalities for image pre-processing, which can improve the accuracy of face detection and feature extraction. While not essential, consider using techniques like **Grayscale Conversion** and **Noise Reduction**

Tools

- **Silent-Face-Anti-Spoofing:** silent face anti-spoofing detection technology is to judge whether the face in front of the machine is real or fake.

(GitHublink: https://github.com/computervisioneng/Silent-Face-Anti-Spoofing)

3-Database Management:

- **SQLite & Local Files:**

For our facial recognition project with a small user base, a lightweight approach using SQLite and local files is ideal. SQLite efficiently stores user credentials like usernames and passwords. Extracted facial features, obtained during training using techniques like Eigenfaces or LBPH, are saved as separate files on the server. These files are named or linked to the user's unique identifier for easy retrieval during recognition. This approach balances simplicity and efficiency while maintaining security through password hashing and separate feature storage.

Project Development Phases:

1-Project Initiation Phase:

Task 1: Project Kickoff

- Define project goals and objectives.

Task 2: Research and Analysis

- **Subtask 2.1: Research Facial Recognition Algorithms:** Analyze different facial recognition algorithms (e.g., Eigenfaces, Local Binary Patterns) and their suitability based on accuracy, speed, and computational resources.
- **Subtask 2.2: Security Analysis:** Research potential security vulnerabilities in facial authentication systems and mitigation strategies (e.g., data storage, user privacy).
- **Subtask 2.3: Document Project Requirements:** Document user requirements (e.g., login process, error handling), functional requirements (e.g., facial recognition accuracy), and non-functional requirements (e.g., performance, scalability).

2-Design Phase:

Task 3: System Architecture Design

- **Subtask 3.1: Identify System Components:** Identify core components like user management, facial recognition engine, database, and authentication module.

Project Development Phases:

- **Subtask 3.3: Design Data Flow and Storage:** Design how facial data is captured, stored securely (hashed), and accessed during authentication.

Task 4: User Interface Design:

- **Subtask 4.1: Create Wireframes/Mockups:** Design user interface layouts for login, registration, and facial capture functionalities.
- **Subtask 4.2: User Feedback and Refinement:** Conduct user testing to gather feedback on usability and refine UI designs based on user input.
- **Subtask 4.3: Finalize UI Designs:** Finalize user interface mockups with clear visuals and user flows for development.

3-Development Phase:

Task 5: Backend Development

- **Subtask 5.1: Implement Facial Recognition Module:** Integrate our chosen facial recognition library (OpenCV) and implement functionalities for face detection, feature extraction, and recognition.
- **Subtask 5.2: User Management and Authentication:** Develop functionalities for user registration, secure password storage (hashing), login process with facial recognition, and session management.
- **Subtask 5.3: Database Integration:** Integrate the chosen database for storing user credentials, potentially hashed facial feature data, and other relevant information.

Project Development Phases:

- **Subtask 5.4: Security Implementation:** Implement security measures like secure password hashing, user input validation, and session management to prevent attacks like SQL injection and unauthorized access.

Task 6: Frontend Development

- **Subtask 6.1: Develop Login and Registration Functionality:** Implement user login forms, registration forms, and error handling for invalid credentials or unsuccessful registrations.
- **Subtask 6.2: Implement Facial Capture Interface:** Develop user interface elements for capturing facial images through webcam access and handling user interaction during capture.
- **Subtask 6.3: Build Authentication Display:** Develop the user interface elements that display the authentication result (successful login or failed recognition).

4-Testing and Bug Fixing Phase:

Task 7: Testing and Bug Fixing

- **Subtask 7.1: Develop Test Cases:** Create comprehensive test cases for all functionalities (user registration, login, facial recognition accuracy) with different scenarios (e.g., good lighting, variations in pose).

Project Development Phases:

- **Subtask 7.2: Identify and Fix Bugs:** Execute test cases, identify bugs or issues and ensure fixes are implemented and retested.

5-Deployment Phase:

Task 8: Deployment Planning

- **Subtask 8.1: Define deployment environment and requirements:** Choose the deployment environment (e.g., local server, cloud platform).
- **Subtask 8.2: Prepare deployment checklist:** including configuration settings, database setup, and security considerations.

Task 9: System Deployment

Task 10: User Documentation

- **Subtask 10.1: Create user documentation and guides**

6-Project Closure Phase:

Task 11: Finalize Project Documentation