



École Normale Supérieure de l'Enseignement Technique

Master1: Systèmes Distribués et Intelligence Artificielle

Facial Recognition Access Control System

Réalisé par:

Noussair Fannan
Yahya Ghallali

Encadré par:

Pr. Loubna AMINOU
Pr. Abdelmajid BOUSSELHAM

DATE: DÉCEMBRE 2024

Contents

1	Project Context	3
1.1	Context	3
1.1.1	Problem Statement	3
1.1.2	Solution	4
1.2	Project Architecture	4
1.2.1	Technical Design	4
1.2.2	Development Approach	4
2	Analysis and Design	5
2.1	Requirement Analysis	5
2.2	UML Design	6
2.2.1	Use Case Diagram	6
2.2.2	Sequence Diagram	7
2.2.3	Class Diagram	10
3	Demonstration	11
3.1	Front-End (JavaFX)	11
3.2	Back-End (Flask Server)	11
3.2.1	Why Choose Python in this Project?	11
3.3	Database (SQLite)	12
3.4	How It Works	12
3.5	Project Screenshots	13
3.5.1	Login Page	13
3.5.2	Access Login Page	13
3.5.3	Admin History Page	14
3.5.4	Admin Users Page	14
3.5.5	Admin Add User Page	15
3.5.6	Admin Delete User	15
4	Assessment	16
4.1	Project Assessment	16
4.2	Team Assessment	16
4.2.1	Technical Expertise	16
4.2.2	Strengths and Areas for Growth	17
4.3	Future Enhancements	17

List of Figures

2.1	Use Case Diagram	6
2.2	User Authentication Sequence Diagram	7
2.3	Admin Authentication Sequence Diagram	8
2.4	Display Users Sequence Diagram	8
2.5	Display History Sequence Diagram	9
2.6	Add User Sequence Diagram	9
2.7	Delete User Sequence Diagram	10
2.8	Class Diagram	10
3.1	Screenshot of the Login Page	13
3.2	Screenshot of the AccesScreenshot of the Login Page	13
3.3	Screenshot of the Admin History Page	14
3.4	Screenshot of the Admin Users Page	14
3.5	Screenshot of the Admin Add User Page	15
3.6	Screenshot of the Admin Delete User	15

Chapter 1

Project Context

1.1 Context

In today's fast-changing world, the need for secure and efficient access control systems has become more important than ever. Traditional systems, such as key cards, passwords, or manual checks, are no longer enough to meet the demands of modern organizations. These older methods often fail to provide the level of security and convenience that both businesses and users require.

Our project addresses this challenge by creating a cutting-edge facial recognition access control system. Unlike traditional systems, this solution uses advanced technology to verify identities quickly, securely, and without physical contact.

This report explains how the system works, the challenges it solves, and the possibilities it offers for the future.

1.1.1 Problem Statement

Many organizations still use traditional access control methods. While these systems have worked for years, they come with serious limitations:

1. **Lost or Stolen Credentials:** Physical access cards can be lost, stolen, or duplicated. This creates a major security risk.
2. **Delays and Inefficiency:** During busy times, systems that rely on manual checks or swiping cards can cause delays. Long lines and waiting times are frustrating for users and inefficient for organizations.
3. **Manual Oversight:** Traditional systems often require staff to monitor access points, which increases labor costs and introduces human error.
4. **Health Concerns:** After the COVID-19 pandemic, people are more cautious about touching shared devices, like fingerprint scanners or keypads. Touchless solutions are now preferred.

Recognizing these problems, we developed a facial recognition system to address these shortcomings. Our solution is touchless, fast, and secure, making it ideal for modern access control needs.

1.1.2 Solution

Our solution combines artificial intelligence (AI) and facial recognition technology in a way that is both user-friendly and reliable. The system uses a JavaFX application for the user interface and a Flask-based backend to handle facial recognition tasks. When a user approaches the system, their face is scanned and compared to the database in real time. If the face matches a stored profile, access is granted instantly. The system also keeps a record of every access attempt, which is useful for audits and security checks. This approach eliminates the need for physical keys, cards, or passwords. Users only need their face to gain access, making the system both convenient and secure.

1.2 Project Architecture

1.2.1 Technical Design

Our system is built using a client-server model, which means the work is divided between two main parts:

1. **Frontend (User Interface):** The frontend is the part of the system that users see and interact with.
 - (a) Built with **JavaFX**, it provides a simple and intuitive graphical interface.
 - (b) The application uses **OpenCV** to process video in real time, detecting faces as users approach the system.
 - (c) We used the **MVC (Model-View-Controller)** pattern, which keeps the code organized and easy to update in the future.
2. **Backend (Server):** The backend is the part of the system that handles complex tasks like analyzing and matching faces.
 - (a) Built with **Flask**, a lightweight web framework.
 - (b) Uses the **face_recognition** library, a powerful tool for facial detection and matching.
 - (c) Stores user profiles and access logs in a **SQLite database**.

The frontend and backend communicate through **Flask REST APIs**. Data is exchanged in JSON format, which is lightweight and easy to process, ensuring fast communication between the two parts.

1.2.2 Development Approach

To ensure the system was reliable and easy to expand, we followed a modular development approach. Each part of the system was built as an independent component, which means we can update or improve one part without affecting the others.

We also implemented **role-based access control**:

1. **Administrators:** Manage user profiles, configure the system, and view access logs.
2. **Regular Users:** Simply scan their face to gain access.

Chapter 2

Analysis and Design

2.1 Requirement Analysis

The system's requirements were carefully analyzed to ensure comprehensive coverage of both functional and non-functional aspects.

Functional Requirements:

1. User Authentication
 - (a) Facial recognition and Identification
 - (b) Admin login with username/password
 - (c) User verification through facial matching
2. User Management
 - (a) Add new users with facial data
 - (b) View and manage user records
 - (c) Update user information
3. Access Control
 - (a) Real-time access verification
 - (b) Access logging
 - (c) History tracking
4. Administrative Functions
 - (a) User management dashboard
 - (b) Access history viewing
 - (c) System configuration

Non-Functional Requirements:

1. Performance
 - (a) Real-time face detection and recognition
 - (b) Quick response time for access decisions
 - (c) Efficient database operations
2. Security
 - (a) Secure storage of facial data
 - (b) Protected admin access
 - (c) Encrypted communication (encoding the user faces)
3. Usability
 - (a) Intuitive user interface
 - (b) Clear feedback on recognition status
 - (c) Easy navigation for administrators

2.2 UML Design

The system's architecture is documented through detailed UML diagrams that illustrate the relationships between components and the flow of operations.

2.2.1 Use Case Diagram

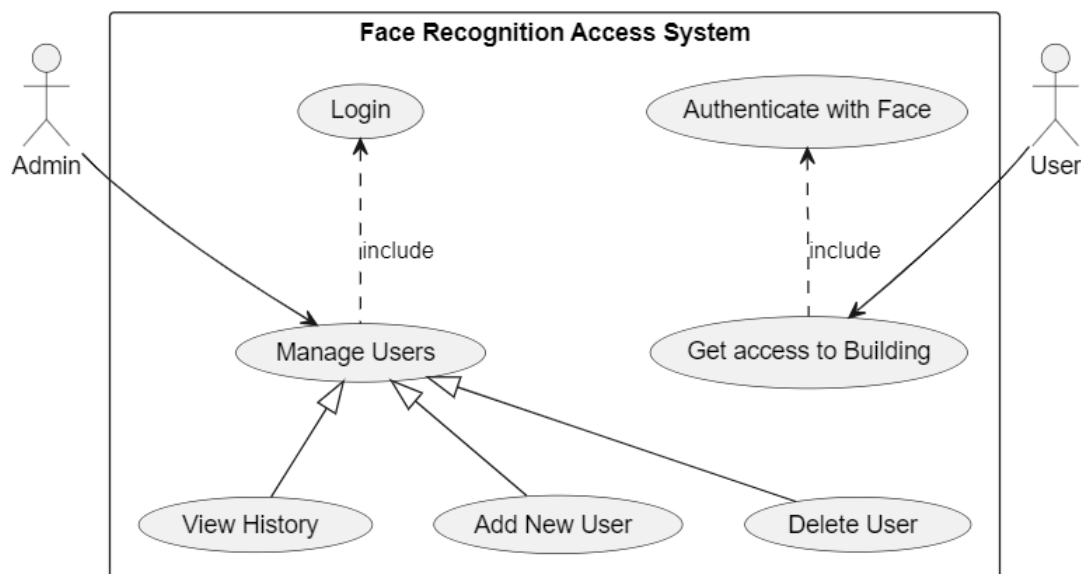


Figure 2.1: Use Case Diagram

2.2.2 Sequence Diagram

User authentication

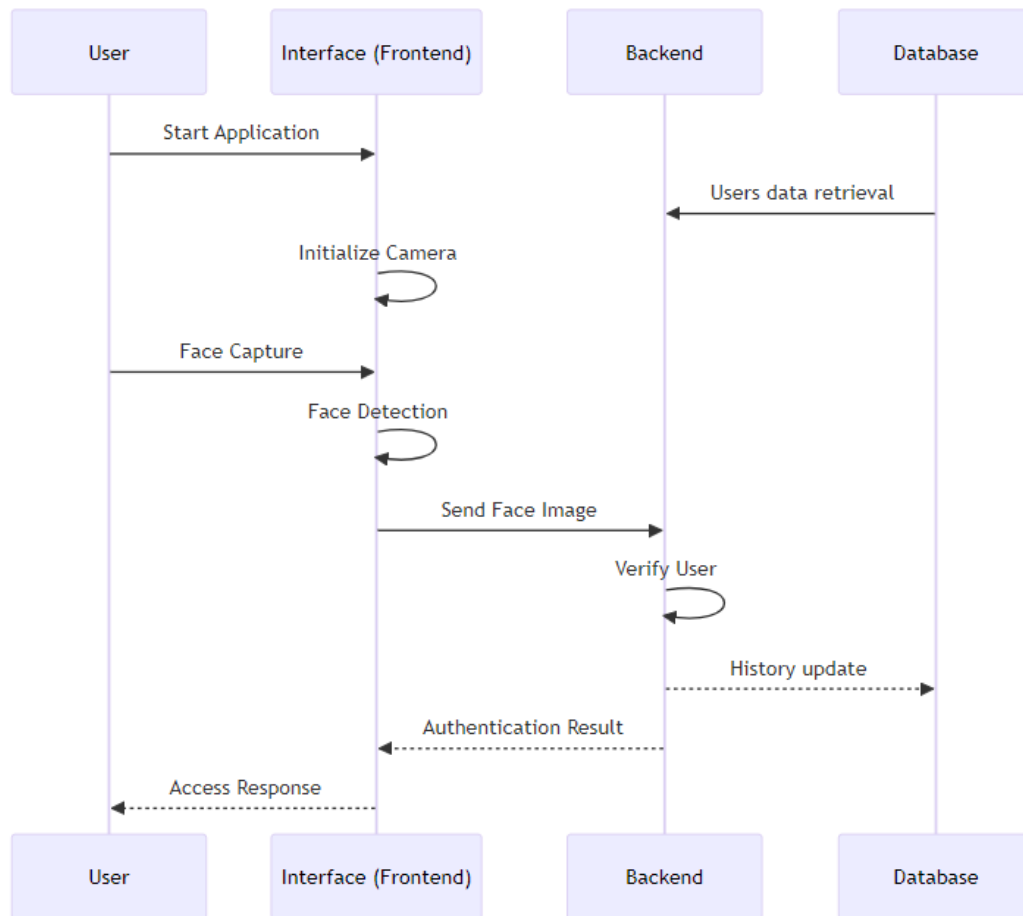


Figure 2.2: User Authentication Sequence Diagram

Admin authentication

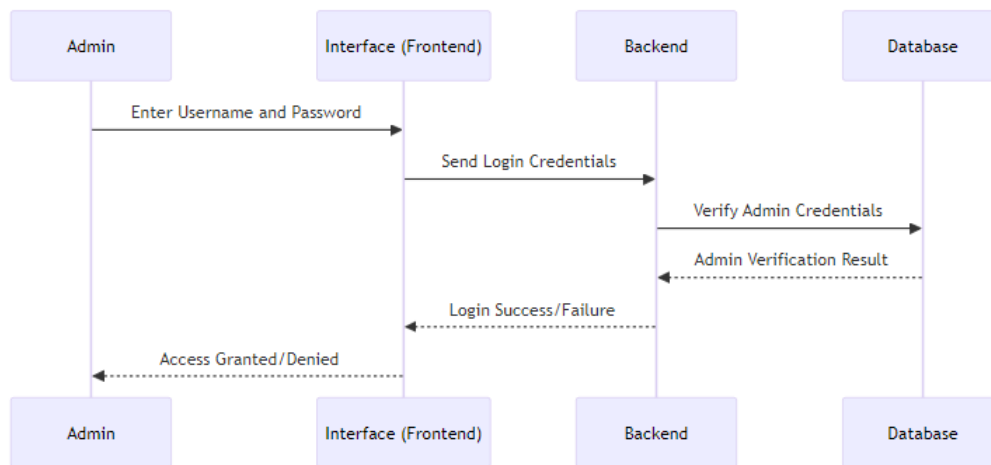


Figure 2.3: Admin Authentication Sequence Diagram

Admin display user

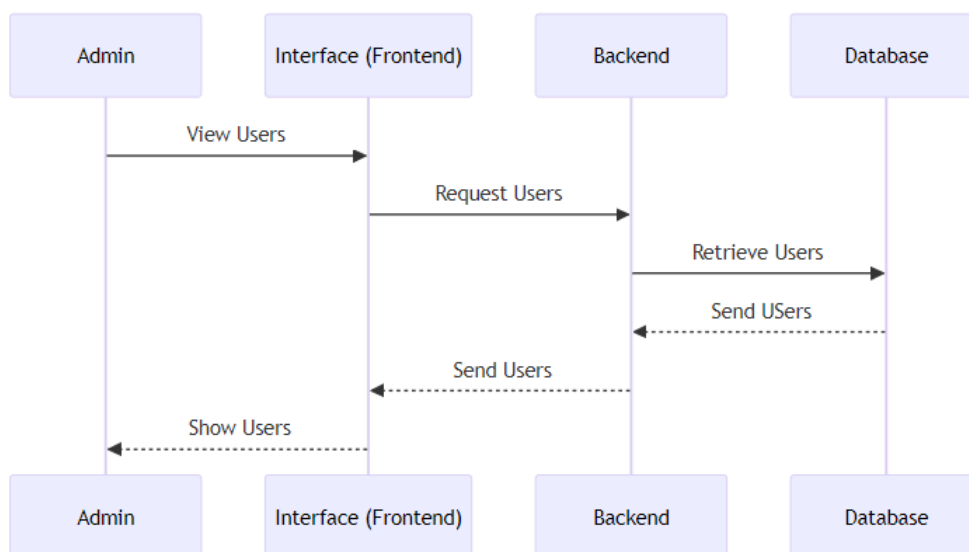


Figure 2.4: Display Users Sequence Diagram

Admin display history

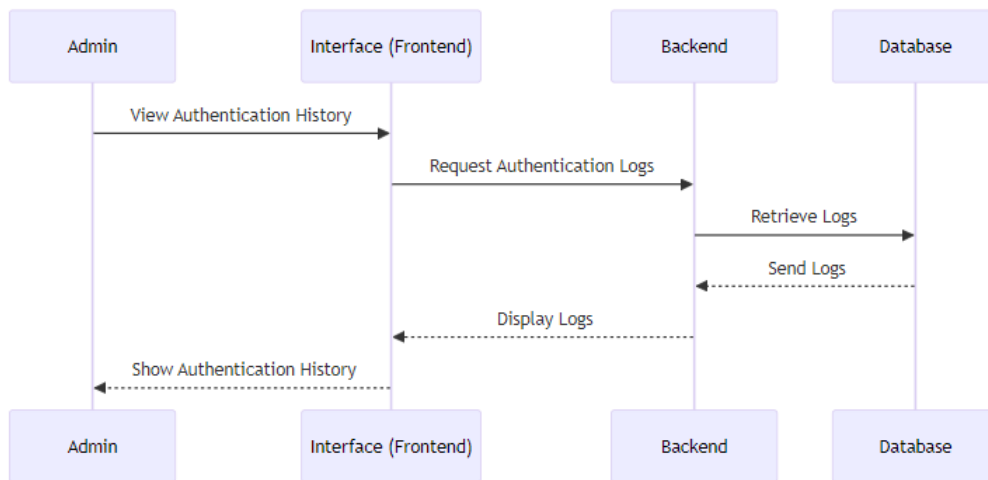


Figure 2.5: Display History Sequence Diagram

Admin add user

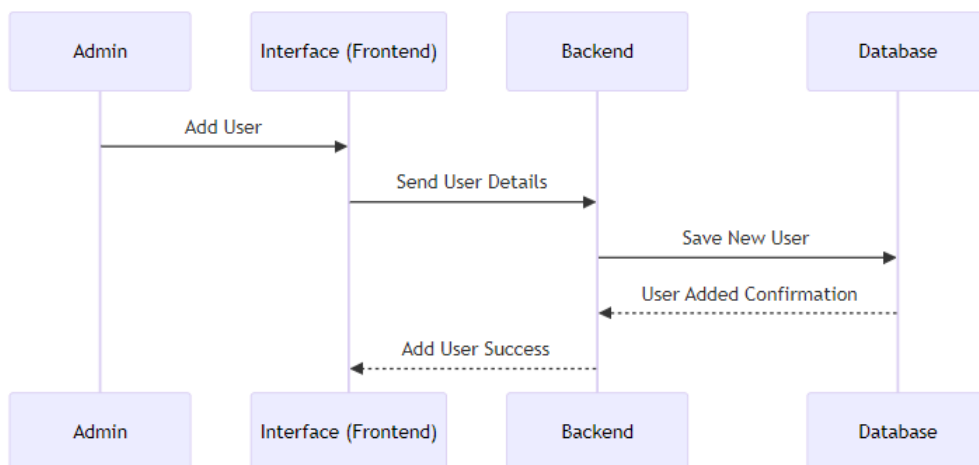


Figure 2.6: Add User Sequence Diagram

Admin delete user

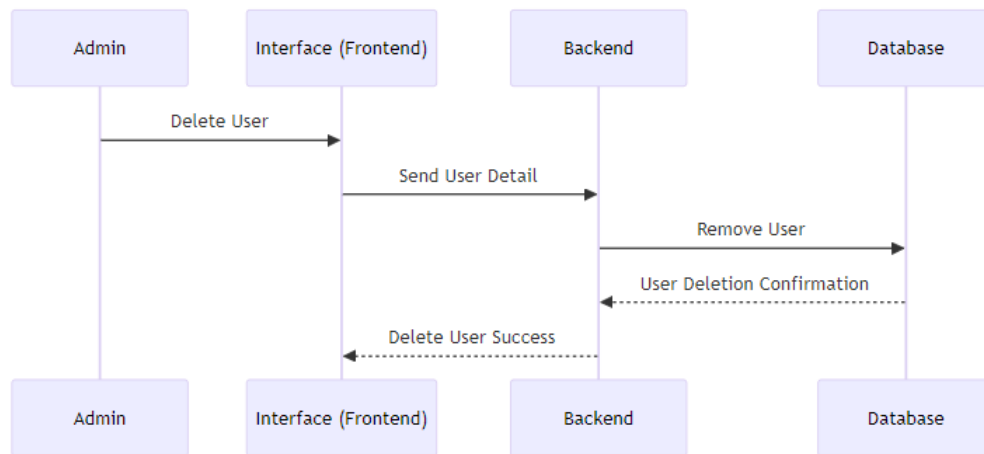


Figure 2.7: Delete User Sequence Diagram

2.2.3 Class Diagram

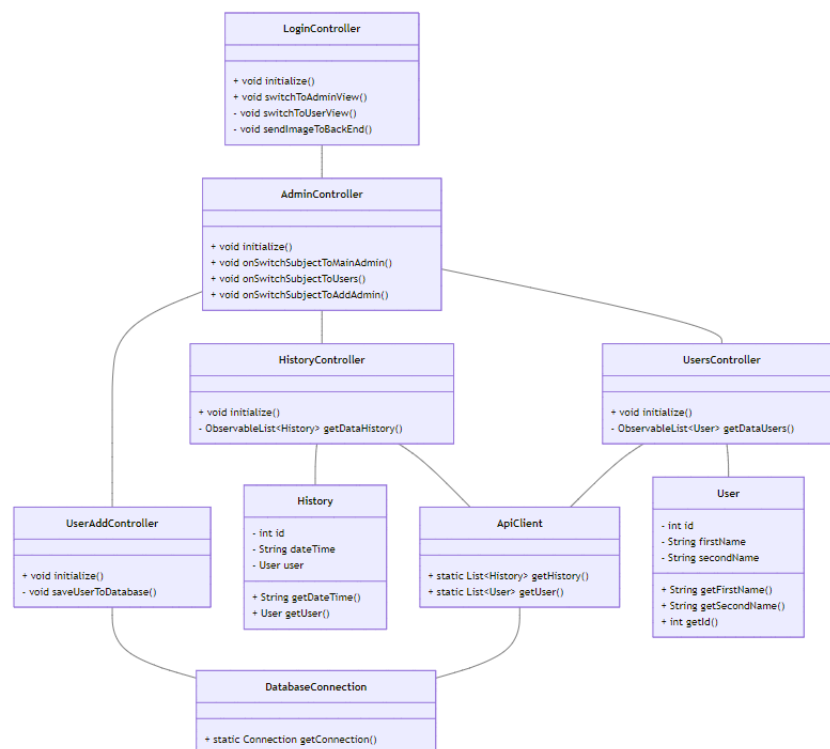


Figure 2.8: Class Diagram

Chapter 3

Demonstration

3.1 Front-End (JavaFX)

The front-end is built using JavaFX and is organized into multiple controllers and views:

- **LoginController:** Handles user authentication. It captures images from the webcam using OpenCV, sends them to the Flask server for facial recognition, and switches to the appropriate dashboard (Admin or User) based on the result.
- **AdminController:** Manages the admin dashboard, allowing admins to view user history, manage users, and add new users. It dynamically loads different FXML views (e.g., `Admin-TableView.fxml`, `Admin-TableUsers.fxml`) based on user actions.
- **HistoryController** and **UsersController:** Handle the display of user history and user lists, respectively, by fetching data from the Flask API.
- **UserAddController:** Allows admins to add new users by entering their details and selecting an image, which is then saved to the database.

3.2 Back-End (Flask Server)

The Flask server (`server.py`) handles facial recognition and database interactions:

- **Facial Recognition:** When an image is uploaded from the JavaFX client, the server uses the `face_recognition` library to detect faces, encode them, and compare them against stored encodings in the database. If a match is found, the user's information is returned, and their login history is updated.
- **API Endpoints:** The server provides endpoints for fetching user history (`/FRapi/history`), retrieving user data (`/FRapi/users`), and uploading images for facial recognition and updating the history (`/upload`).

3.2.1 Why Choose Python in this Project?

In any project, the key objectives are **speed of development**, **long-term stability**, and **reliability**. Python excels in these areas, making it the optimal choice for deploying AI models. Here's why:

- **Dominance in AI Development**
 - Python is the most widely used language for AI and machine learning, with a vast ecosystem of libraries and frameworks like TensorFlow, PyTorch, and Scikit-learn. Its extensive adoption ensures better community support and rapid integration of cutting-edge research.
- **Versatility Through Abstraction**
 - Python's high level of abstraction simplifies complex tasks, making it incredibly versatile. This allows developers to focus on solving AI problems without being bogged down by low-level programming details.
- **Development Efficiency vs. Runtime Performance**
 - While Java is inherently faster than Python in terms of runtime (measured in milliseconds or, at most, a few seconds), the difference is negligible for most AI applications.
 - The real advantage lies in the ****time saved during development****. Python's concise and intuitive syntax significantly reduces the time needed to write, debug, and maintain code.
 - Additionally, the AI models used in both Python and Java are often implemented in C++ under the hood, which minimizes performance differences in computational tasks.

In summary, Python's widespread use, simplicity, and efficiency in development make it the preferred choice for AI projects, especially when balancing development speed with reliable performance.

3.3 Database (SQLite)

The SQLite database (`ProjectFR.db`) stores:

- **User Data:** Includes user IDs, names, image paths, and facial encodings.
- **History:** Tracks user login attempts with timestamps and user IDs.

3.4 How It Works

1. **Login:** The user logs in via the JavaFX interface. The webcam captures an image, which is sent to the Flask server for facial recognition. If a match is found, the user is allowed to enter.
2. **Admin Dashboard:** Admins can view user history, manage users, and add new users. The UI dynamically loads different views based on admin actions.
3. **Facial Recognition:** The Flask server processes uploaded images, compares them against stored encodings, and updates the database with login history.

3.5 Project Screenshots

3.5.1 Login Page

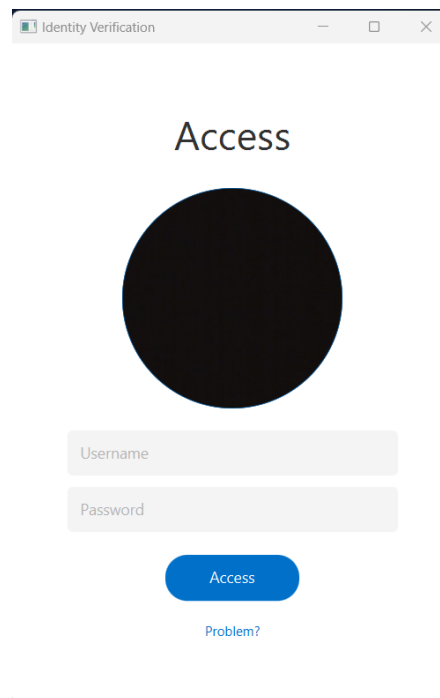


Figure 3.1: Screenshot of the Login Page

3.5.2 Access Login Page

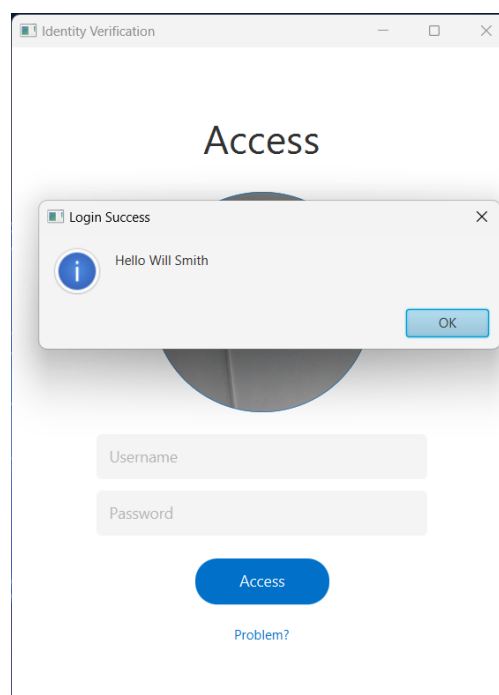
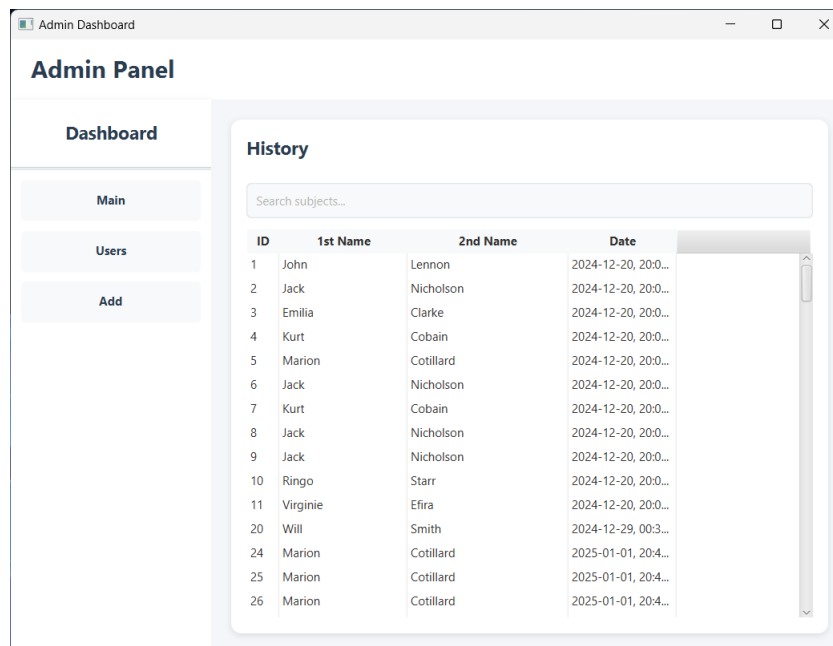


Figure 3.2: Screenshot of the AccessScreenshot of the Login Page

3.5.3 Admin History Page

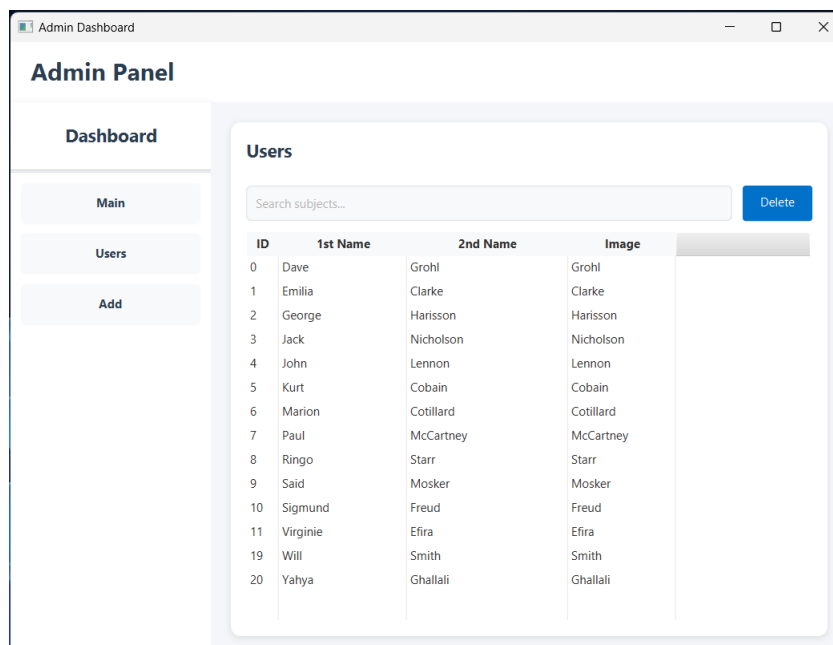


The screenshot shows the Admin History Page. On the left is a sidebar with 'Admin Panel' and a 'Dashboard' menu containing 'Main', 'Users', and 'Add'. The main content area is titled 'History' and features a search bar 'Search subjects...'. Below the search bar is a table with columns 'ID', '1st Name', '2nd Name', and 'Date'. The table contains 13 rows of data, with the last row partially cut off.

ID	1st Name	2nd Name	Date
1	John	Lennon	2024-12-20, 20:0...
2	Jack	Nicholson	2024-12-20, 20:0...
3	Emilia	Clarke	2024-12-20, 20:0...
4	Kurt	Cobain	2024-12-20, 20:0...
5	Marion	Cotillard	2024-12-20, 20:0...
6	Jack	Nicholson	2024-12-20, 20:0...
7	Kurt	Cobain	2024-12-20, 20:0...
8	Jack	Nicholson	2024-12-20, 20:0...
9	Jack	Nicholson	2024-12-20, 20:0...
10	Ringo	Starr	2024-12-20, 20:0...
11	Virginie	Efra	2024-12-20, 20:0...
20	Will	Smith	2024-12-29, 00:3...
24	Marion	Cotillard	2025-01-01, 20:4...
25	Marion	Cotillard	2025-01-01, 20:4...
26	Marion	Cotillard	2025-01-01, 20:4...

Figure 3.3: Screenshot of the Admin History Page

3.5.4 Admin Users Page

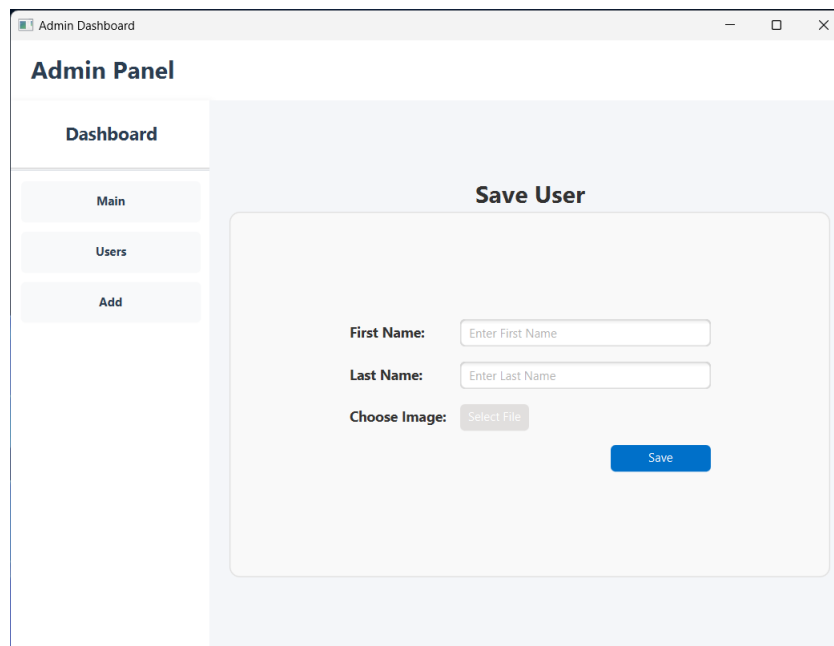


The screenshot shows the Admin Users Page. On the left is a sidebar with 'Admin Panel' and a 'Dashboard' menu containing 'Main', 'Users', and 'Add'. The main content area is titled 'Users' and features a search bar 'Search subjects...' and a 'Delete' button. Below the search bar is a table with columns 'ID', '1st Name', '2nd Name', and 'Image'. The table contains 13 rows of data, with the last row partially cut off.

ID	1st Name	2nd Name	Image
0	Dave	Grohl	Grohl
1	Emilia	Clarke	Clarke
2	George	Harrison	Harrison
3	Jack	Nicholson	Nicholson
4	John	Lennon	Lennon
5	Kurt	Cobain	Cobain
6	Marion	Cotillard	Cotillard
7	Paul	McCartney	McCartney
8	Ringo	Starr	Starr
9	Said	Mosker	Mosker
10	Sigmund	Freud	Freud
11	Virginie	Efra	Efra
19	Will	Smith	Smith
20	Yahya	Ghallali	Ghallali

Figure 3.4: Screenshot of the Admin Users Page

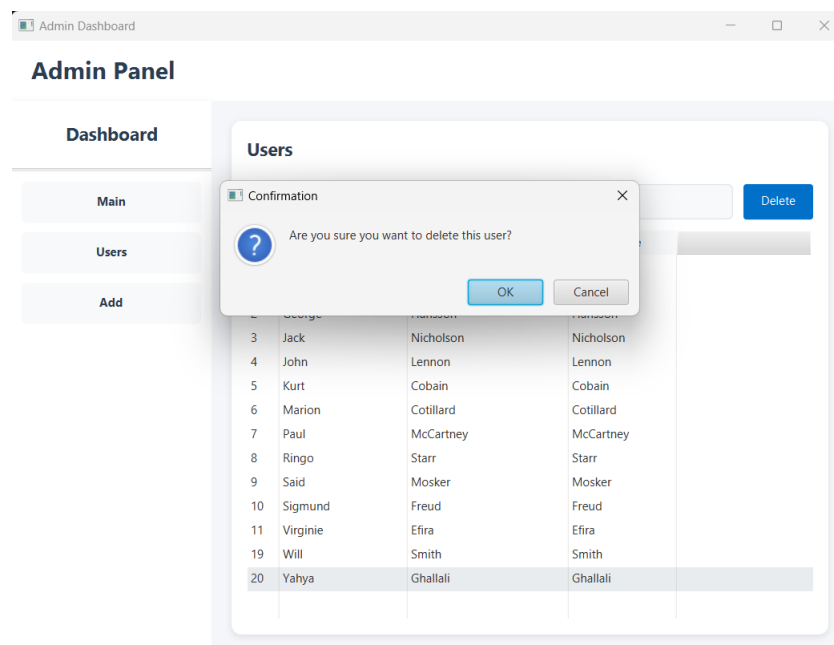
3.5.5 Admin Add User Page



The screenshot shows a web application window titled "Admin Dashboard". On the left is a sidebar labeled "Admin Panel" with a "Dashboard" header and three menu items: "Main", "Users", and "Add". The "Add" menu item is highlighted. The main content area is titled "Save User" and contains a form with three fields: "First Name:" with a text input labeled "Enter First Name", "Last Name:" with a text input labeled "Enter Last Name", and "Choose Image:" with a "Select File" button. A blue "Save" button is located at the bottom right of the form.

Figure 3.5: Screenshot of the Admin Add User Page

3.5.6 Admin Delete User



The screenshot shows the "Admin Dashboard" window with the "Users" menu item highlighted in the sidebar. The main content area displays a table of users. A "Confirmation" dialog box is overlaid on the table, asking "Are you sure you want to delete this user?" with "OK" and "Cancel" buttons. A "Delete" button is visible in the top right corner of the table area.

	First Name	Last Name	Full Name
1	George	Nicholson	Nicholson
3	Jack	Nicholson	Nicholson
4	John	Lennon	Lennon
5	Kurt	Cobain	Cobain
6	Marion	Cotillard	Cotillard
7	Paul	McCartney	McCartney
8	Ringo	Starr	Starr
9	Said	Mosker	Mosker
10	Sigmund	Freud	Freud
11	Virginie	Efra	Efra
19	Will	Smith	Smith
20	Yahya	Ghallali	Ghallali

Figure 3.6: Screenshot of the Admin Delete User

Chapter 4

Assessment

4.1 Project Assessment

Our project has successfully accomplished its primary objectives, demonstrating significant progress and effectiveness. The facial recognition system we developed is characterized by several key attributes:

- **Fast:** It is capable of processing faces in real time, ensuring that there is minimal delay in recognition. This rapid processing allows for seamless interactions, making it highly efficient for various applications.
- **Accurate:** The system exhibits a high level of accuracy, as it can effectively recognize users even when they are in different lighting conditions. This adaptability enhances its reliability, ensuring that it performs well in a variety of environments.
- **User-Friendly:** Feedback from both administrators and regular users indicates that they found the system to be easy to navigate and operate. Its intuitive design contributes to a positive user experience, making it accessible to individuals with varying levels of technical expertise.

4.2 Team Assessment

The project's success reflects the dedication and collaboration of the two-person team, where each member played a critical role:

4.2.1 Technical Expertise

Both team members demonstrated strong technical skills:

- One focused on the frontend, building a user-friendly JavaFX interface and integrating OpenCV for real-time video processing.
- The other concentrated on the backend, developing the Flask REST API, implementing the `face_recognition` library, and managing the SQLite database for secure data storage.

Together, we ensured smooth communication between components and high system performance. Importantly, we contributed to each other's areas when needed.

4.2.2 Strengths and Areas for Growth

- **Strengths:** Our ability to divide tasks efficiently and focus on their strengths resulted in a well-designed, functional system.
- **Improvements:** Allocating more time for testing and cross-training could further enhance their workflow and adaptability in future projects.

4.3 Future Enhancements

The current system provides a solid foundation, but there's room for growth. In the future, we plan to:

- Store data in the cloud for better scalability and easier management, for instance in large sites using distributed systems.
- Create a mobile app to let administrators manage the system remotely.
- Implement this system into real-life, using camera, Arduino or any small computing unit capable of running the recognition model and communicating directly with the admin using internet.

These features will make the system even more powerful, while keeping its current strengths intact.