student attendance management system

FRAMS, a student attendance management system that uses facial recognition to track student attendance

By: Ailly Shilunga gr.10-11

# **TABLE OF CONTENT**

[1](#_Toc206877555)[**Introduction** 2](#_Toc206877556)

[Description of business/organization 2](#_Toc206877557)

[Description of problem 2](#_Toc206877558)

[Specific objects 4](#_Toc206877559)

[Overall aim 4](#_Toc206877560)

[**Saving Time and Making Attendance Faster** 4](#_Toc206877561)

[**Making Attendance More Accurate** 5](#_Toc206877562)

[**Preventing Cheating and Strengthening Security** 5](#_Toc206877563)

[**Helping Teachers and Schools Make Smart Decisions** 5](#_Toc206877564)

[**‍Making the System Easy to Use** 5](#_Toc206877565)

[**Respecting Privacy and Keeping Data Safe** 5](#_Toc206877566)

[**Improving the Whole School System** 5](#_Toc206877567)

[Description of other possible solutions 5](#_Toc206877568)

[**Evaluation of existing solution** 9](#_Toc206877569)

[**Design** 11](#_Toc206877570)

[Action Plan 11](#_Toc206877571)

[System Flowchart 12](#_Toc206877572)

[**Method of solution** 13](#_Toc206877573)

[Hardware requirements 16](#_Toc206877574)

[**Software requirements** 17](#_Toc206877575)

[Programing code 19](#_Toc206877576)

[**Testing** 23](#_Toc206877577)

[Test strategy 23](#_Toc206877578)

[Test Results 24](#_Toc206877579)

[**User Guide** 25](#_Toc206877580)

[**Evaluation and Development** 26](#_Toc206877581)

[Evaluation 26](#_Toc206877582)

[**Opportunities for system development** 27](#_Toc206877583)

# **Introduction**

## Description of business/organization

Jan Mohr Secondary School, a prestigious learning institution which aims to facilitate the provision of high-quality education to ensure that students achieve their full academic performance hence, their moto ***“Altyd my beste”***, (Afrikaans for “***Always my best”***),is institute located in the capital city of Namibia, Windhoek (Windhoek West to be precise).Jan Mohr has a very unique number when it comes to staff, consisting of 3 Secretaries, 4 HOD’s, about 4-5 Sport teachers, 1 librarian and 8 cleaners. In total there are about 58 teachers. The school has 4 laboratories, 1 for Computer Science, 1 for Office practice, 1 Science laboratory and 1 for Biology. The school also has a few sport fields. This institute consists favor’s varies students from grades ranging from gr.8-12. Yearly, the school enrolls about

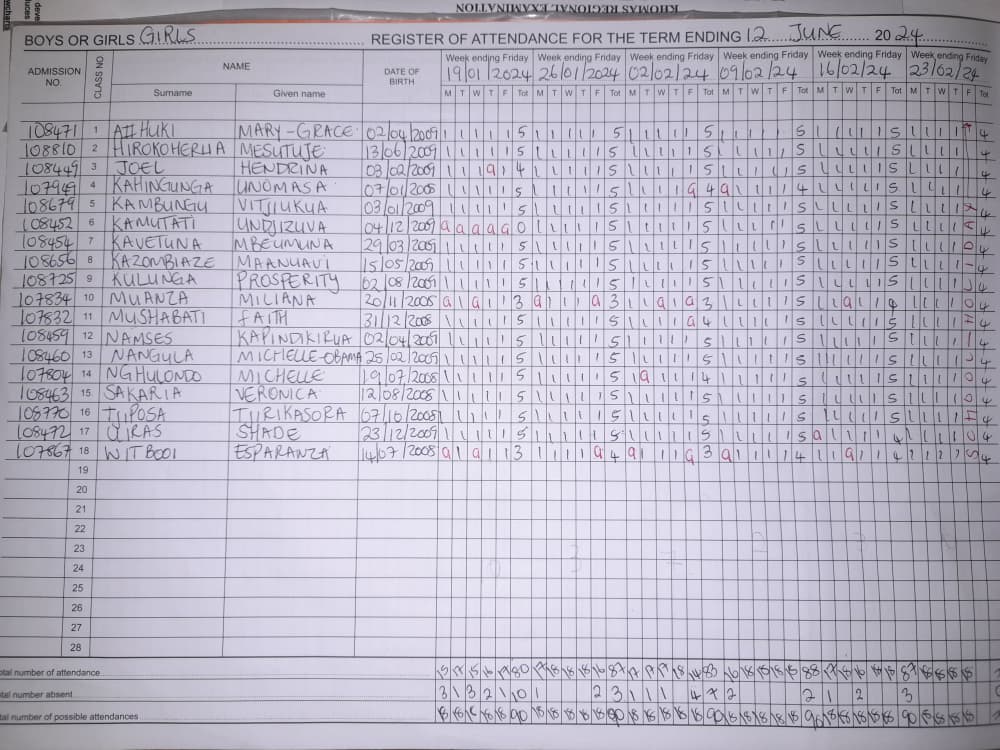
## Description of problem

Jan Mohr uses a paper based system to track student attendance using manual roll-call as one of their input methods. The current system put to play consist of three major parties: The register teacher, the class monitor of that class and the HOD (Head of department).

Each teacher is assigned a register class and selects their own class monitor depending on the class teacher’s choice after that is therefore handed a register file where they are required to take attendance of the whole week and is required to hand it in at their respected HOD. During this week, the class teacher works hand- in- hand with the class monitor to help track attendance, misbehavior and truancy. The class captain is handed a few class lists in order to take attendance. This class lists are normal class lists that have the names of students, id numbers and the date the class list was printed. These class lists are used for varies purpose such as recording marks, ticking of those who paid for a certain event, and to assign what team the learner will be on for sports day and many others

The class monitor is required to have a class list for their class lists every week as they will as t to take attendance. At the beginning of every first period, the class captain takes attendance of that day using manual roll-call and does so for the rest of the week. At the end of the week the teacher asks for the attendance that the class captain took for that week so that the teacher can update their register. An example of how the register file look like is given below





As illustrated above, that is the teacher’s attendance register for 9A that shows the student admission number (ID number), surname, name and date of birth. Each student has their own “row” of their own whereby their attendance is marked for a week (the 1’s or sticks indicate that the student was present that day, the number at the end of each week indicate that the learner was present for i.e. 5 days or 4 days, or 0).

The registers are then handed in on every Friday to the teacher’s respected HOD for further analysis, where the HOD checks if the teacher took attendance of that week and, than the HOD checks if there are any suspicious trends in the attendance of the week.

All in all, in many schools, taking attendance is still a manual process. Teachers often rely on paper registers, verbal roll-calls, or spreadsheets to record which students are present, absent, or late. This method is not only time-consuming but also prone to human error, manipulation, and inconsistency. In large classrooms, it can take several minutes just to complete attendance—time that could be better spent on teaching and learning.

Manual attendance also opens the door to dishonest practices. Students may ask friends to sign in for them, or teachers may accidentally mark the wrong student. Over time, these inaccuracies can affect school records, student performance tracking, and even disciplinary decisions.

Another major challenge is the lack of centralized student data. Without a digital system, schools struggle to maintain up-to-date records of who is enrolled, which class they belong to, and how often they attend. This makes it harder to identify students who need support or to generate reliable reports for parents and administrators.

FRAMS (Facial Recognition Attendance Management System) was created to solve these problems. By using facial recognition technology, FRAMS automates the attendance process, ensures accuracy, and encourages proper student registration. It saves time, reduces errors, and helps schools build a more organized and accountable system.

## Specific objects

Overall aim: To fully computerize the current student attendance management system

### **Saving Time and Making Attendance Faster**

Taking attendance manually—calling names, ticking boxes—can take several minutes every class. FRAMS speeds this up dramatically. With facial recognition, students are marked present automatically as soon as they walk in front of the camera. Teachers don’t need to do anything extra. This means more time for teaching and less time wasted on routine tasks.

Key Speed-Boosting Features

* Preloaded Class Roster:  
  Student profiles and face descriptors are loaded on dashboard initialization, so recognition starts immediately without fetching data mid-session.
* Real-Time Face Detection  
  The system scans the webcam feed every second, instantly identifying known faces and marking them present.
* Automatic Status Update  
  once a match is found, attendance is recorded and saved to localStorage and can to be downloaded with no additional clicks or confirmations.
* Parallel Processing:  
  multiple faces can be detected and matched simultaneously, enabling large classes to be processed in bulk.
* Smart Fall-Back:
* Unrecognized faces are ignored without interrupting the flow, ensuring no delays even when new or unexpected faces appear.

### **Making Attendance More Accurate**

Mistakes happen when attendance is done by hand—names get missed, marked wrong, or forgotten. FRAMS uses each student’s face to match them with their profile, so there’s no confusion. Once a student is recognized, the system updates their status instantly and correctly. This helps keep records clean and trustworthy.

### **Preventing Cheating and Strengthening Security**

Sometimes students ask friends to sign in for them when they’re not really in class. FRAMS stops this kind of cheating. Because it uses live video and facial recognition, only the actual student can be marked present. This helps schools keep honest records and makes sure everyone is treated fairly.

### **Helping Teachers and Schools Make Smart Decisions**

FRAMS doesn’t just take attendance—it also keeps track of patterns. If a student is often late or absent, the system can help teachers spot that early. Over time, schools can use this data to understand which classes have the most absences, which students might need support, and how attendance affects learning.

### **‍Making the System Easy to Use**

FRAMS is built to be simple and friendly. Teachers see clear buttons and color-coded labels for each student. If someone isn’t recognized by the camera, the teacher can still mark them manually with one click. The system works on laptops, desktops, and even tablets, so it fits into any classroom setup.

### **Respecting Privacy and Keeping Data Safe**

FRAMS takes privacy seriously. It doesn’t send student photos to the internet or store them on outside servers. Everything is handled safely inside the school’s system. Students and parents can feel confident that their information is protected and used only for attendance.

### **Improving the Whole School System**

To use FRAMS, every student must be properly registered in the school’s database. This encourages schools to keep accurate records of who is enrolled, which class they’re in, and how they’re doing. Over time, this helps build a stronger, more organized school system where everything is connected—from attendance to performance to support.

## Description of other possible solutions

When exploring ways to improve attendance tracking at the school, three potential approaches were evaluated:

1. **Spreadsheet-Based Attendance System**
2. **Off‑the‑Shelf School Management Software**
3. **Facial Recognition Attendance Management System (FRAMS)**

Each solution was assessed for suitability, efficiency, cost, and potential impact on the school.

**1. Spreadsheet-Based Attendance System**

**Description:**  
Teachers record attendance either manually on paper and later transfer it to a spreadsheet, or directly type it into applications such as Microsoft Excel or Google Sheets. Formulas can be used to calculate totals for present, absent, or late students.

**Advantages:**

* Low cost with no extra software purchase needed
* Easy to set up and familiar to most teachers
* Basic automation possible using built-in formulas
* Can sort and filter data by class, date, or student name

**Disadvantages:**

* Still a time-consuming and partially manual process
* Becomes slow and difficult to manage with large datasets
* No built-in protection against errors or data tampering
* Limited security features and no attendance fraud prevention

**2. Off‑the‑Shelf School Management Software**

**Description:**  
Commercial platforms (e.g., Edupac, Pastel Education) provide ready‑made modules for attendance, grading, timetables, and reporting.

**Advantages:**

* Specifically designed for schools
* Fast and accurate data entry and reporting
* Often comes with technical support and cloud backup
* Can integrate with other administrative tasks such as grading or fees

**Disadvantages:**

* High initial purchase or subscription costs
* Requires user training before effective use
* Limited customization for the school’s exact needs
* Dependent on vendor support and stable internet connection

**3. Proposed Solution: FRAMS (Facial Recognition Attendance Management System)**

**Description:**  
FRAMS uses AI-powered facial recognition to automate the entire attendance process. Once students are registered with their facial profiles, the system instantly identifies and marks them present as they arrive. Attendance data is stored securely and can be accessed in real time.

**Advantages:**

* Fully automated — marks attendance within seconds
* Highly accurate, reducing human errors
* Prevents proxy or false attendance
* Improves the overall school record system by requiring up‑to‑date student registration
* Saves time, allowing teachers to focus on lessons
* Data can be used for analytics and performance tracking

**Disadvantages:**

* Requires initial investment in cameras and supporting hardware
* Needs ongoing system maintenance and software updates
* Privacy concerns require appropriate safeguards and consent management

Summary Table

|  |  |  |
| --- | --- | --- |
| **Solution** | **Advantages** | **Disadvantages** |
| **Spreadsheet-Based Attendance** | - Low cost - Familiar and easy to use - Basic automation with formulas | - Still partly manual - Slows down with large data - No built-in fraud prevention |
| **Off‑the‑Shelf School Management Software** | - Designed for schools - Fast, accurate reports - Includes technical support | - High cost - Requires training - Limited customization - Internet dependent |
| **FRAMS (Proposed Solution)** | -Fully automated - Accurate and secure - Prevents attendance fraud - Strengthens entire school record system | - Initial hardware cost - Maintenance required - Must manage privacy safeguards |

## **Evaluation of existing solution**

In many schools, attendance is still recorded using traditional methods such as paper registers, verball roll-calls, or spreadsheet-based systems. These approaches have been used for decades and are familiar to most teachers and administrators. While they may seem simple and cost-effective, they come with several limitations that affect accuracy, efficiency, and the overall management of student data.

**✅ Advantages of the Current System**

1. **Simplicity and Familiarity**  
   Teachers are already trained to use manual registers or spreadsheets, so no additional technical skills are required. This makes the system easy to adopt and maintain without specialized equipment.
2. **Low Initial Cost**  
   Paper registers and basic spreadsheet tools require minimal investment. Schools don’t need to purchase hardware or software licenses to implement them.
3. **Offline Accessibility**  
   Manual systems do not rely on internet connectivity or power supply. Attendance can be taken in any classroom, regardless of technical infrastructure.
4. **Flexibility in Recording Notes**  
   Teachers can jot down comments, reasons for absence, or other observations directly on paper, which may feel more personal and adaptable than digital systems.

**❌ Disadvantages of the Current System**

1. **Time-Consuming Process**  
   Calling out names and manually marking attendance can take several minutes per class, especially in large groups. This reduces valuable teaching time and can cause delays in lesson delivery.
2. **Prone to Human Error**  
   Mistakes such as marking the wrong student, skipping names, or forgetting to update records are common. These errors can lead to inaccurate reports and confusion during audits.
3. **Lack of Real-Time Data**  
   Manual records are not instantly accessible to administrators or parents. Teachers often need to submit attendance at the end of the day or week, which delays interventions for absent students.
4. **Risk of Manipulation or Proxy Attendance**  
   Students may ask friends to sign in for them, especially when using paper registers. This undermines the integrity of the attendance system and makes it difficult to track genuine participation.
5. **Storage and Retrieval Issues**  
   Paper records can be lost, damaged, or difficult to organize over time. Retrieving historical attendance data for a specific student or class can be tedious and unreliable.
6. **Limited Integration with School Systems**  
   Manual attendance is often disconnected from other school databases such as grading systems, student profiles, or disciplinary records. This makes it harder to analyze trends or generate comprehensive reports.

**Justification of the Proposed Solution: FRAMS**

The **Facial Recognition Attendance Management System (FRAMS)** stands out as the most effective option because it directly addresses the limitations of the other methods and aligns with the school’s long‑term goals.

**1. Efficiency & Time-Saving**

Unlike manual or semi‑automated systems, FRAMS marks attendance within seconds. This means teachers reclaim valuable classroom time to focus on teaching rather than record‑keeping.

**2. Accuracy & Fraud Prevention**

Facial recognition eliminates errors from manual entry and stops cases of proxy attendance (students answering for absent classmates). This strengthens the credibility of attendance records.

**3. Data Security & Accessibility**

Records are stored in a secure database, with role‑based access controls. Authorized staff can view real‑time attendance reports from anywhere with permission.

**4. Long‑Term Cost Effectiveness**

Although there is an initial hardware and setup cost, FRAMS reduces the ongoing labor cost associated with paper‑based or manual systems. Maintenance costs are predictable and can be budgeted annually.

**5. Supports Wider School Modernization**

FRAMS can integrate with other digital systems — such as grading, reporting, and ID card management — contributing to the school’s overall digital transformation strategy

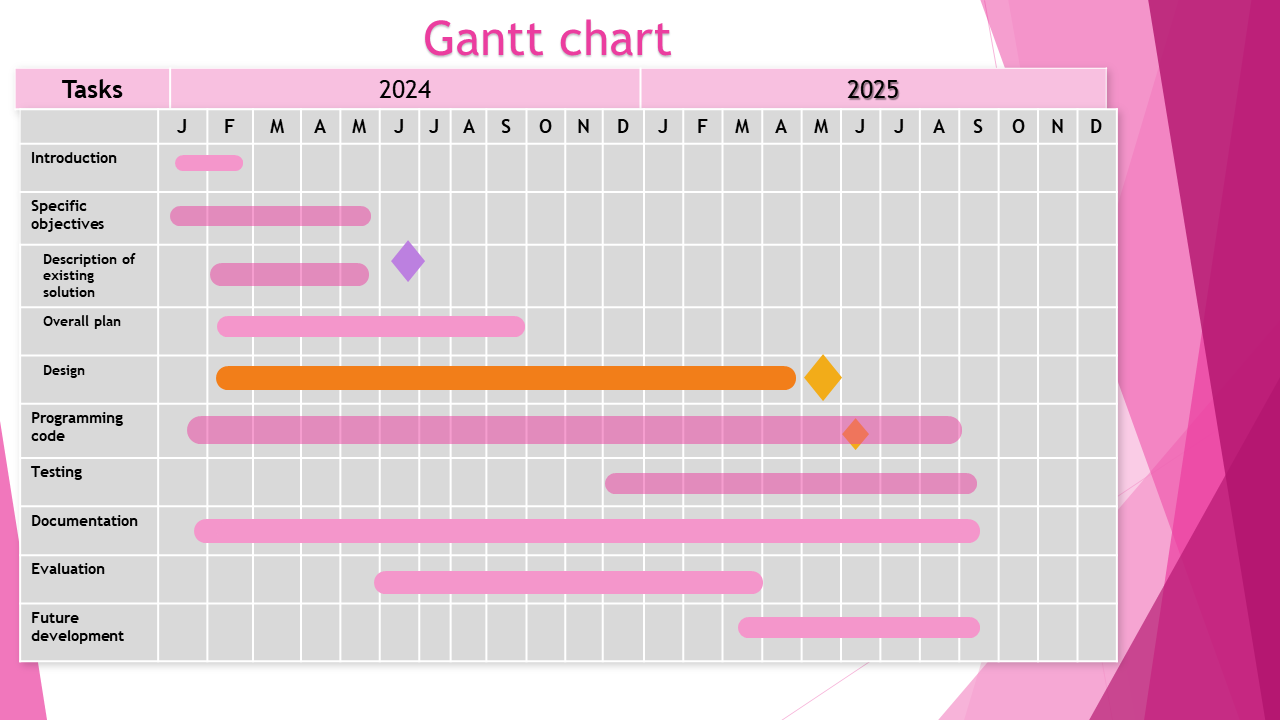
📌 Summary

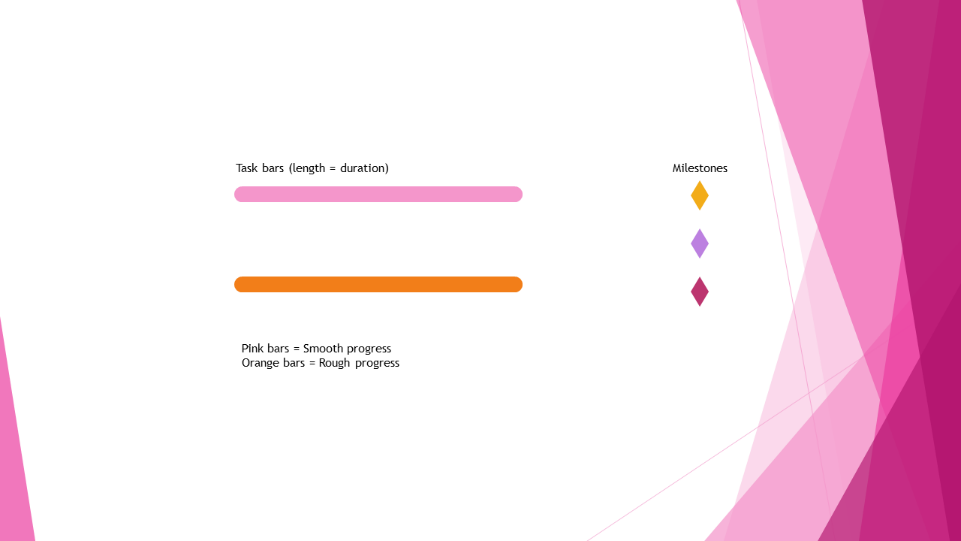
While the current attendance system is simple and widely used, it falls short in areas that matter most—speed, accuracy, security, and data management. As schools grow and technology becomes more accessible, there is a clear need for a smarter, more reliable solution that can automate attendance, reduce errors, and integrate seamlessly with the broader school system.

# **Design**

## Action Plan

The implementation of the Facial Recognition Attendance Management System (FRAMS) will be guided by both computer‑related and business‑related objectives. On the technical side, the system aims to automate attendance marking, ensure secure storage of records, and deliver real‑time reporting. From a business perspective, it is designed to enhance operational efficiency, reduce administrative workload, and provide accurate and timely data to support decision‑making. The action plan that follows aligns closely with the prepared Gantt chart, ensuring each phase and task is strategically scheduled and fully traceable.

****

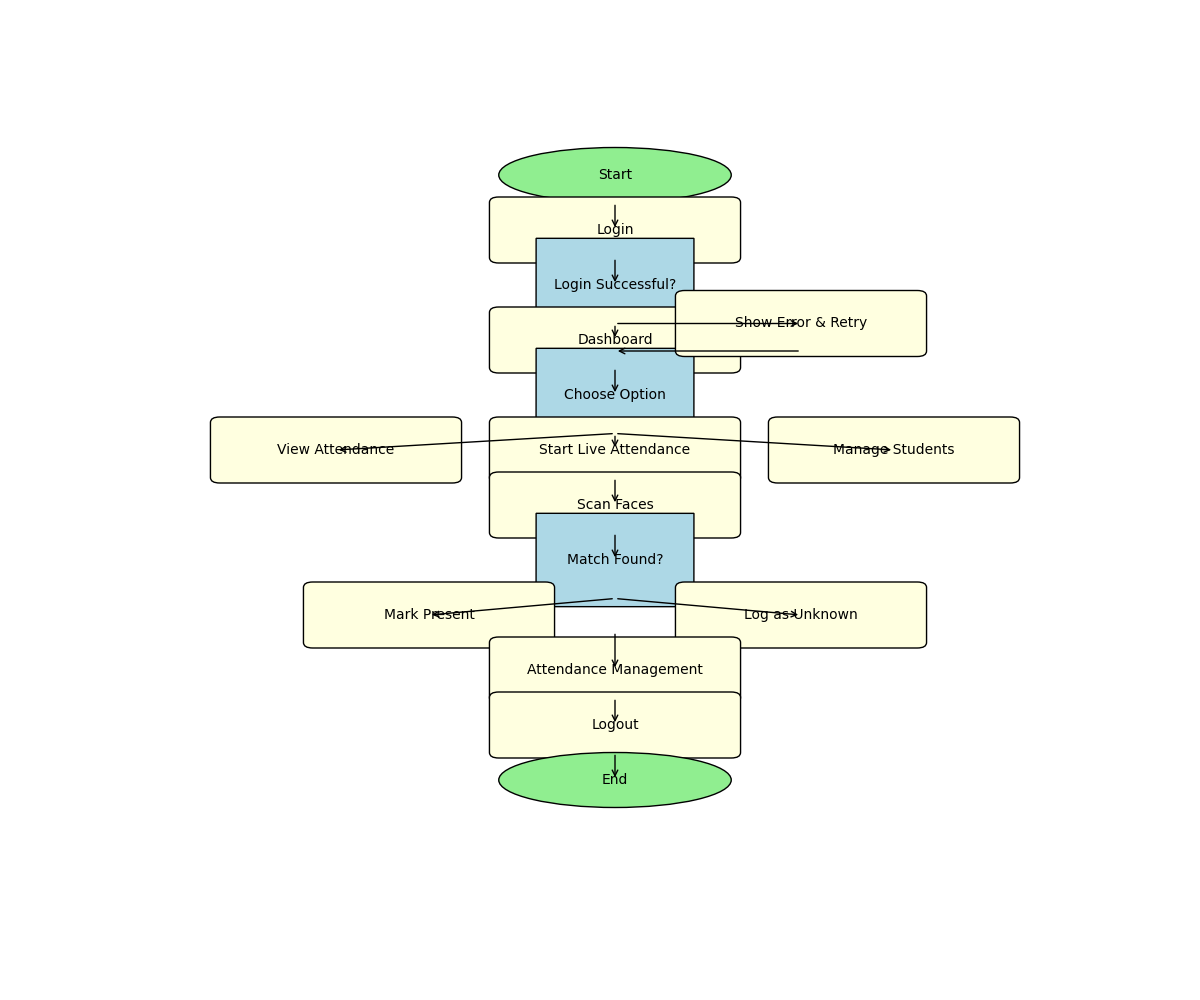
****

## System Flowchart

Create Account

SSign up

The system begins with a login screen. If credentials are valid, the user accesses the dashboard. From there, they can initiate live attendance, where facial recognition scans students. If a face matches, attendance is marked; if not, it's logged as unknown. Teachers can also manually manage attendance. The session ends with logout, returning to the login

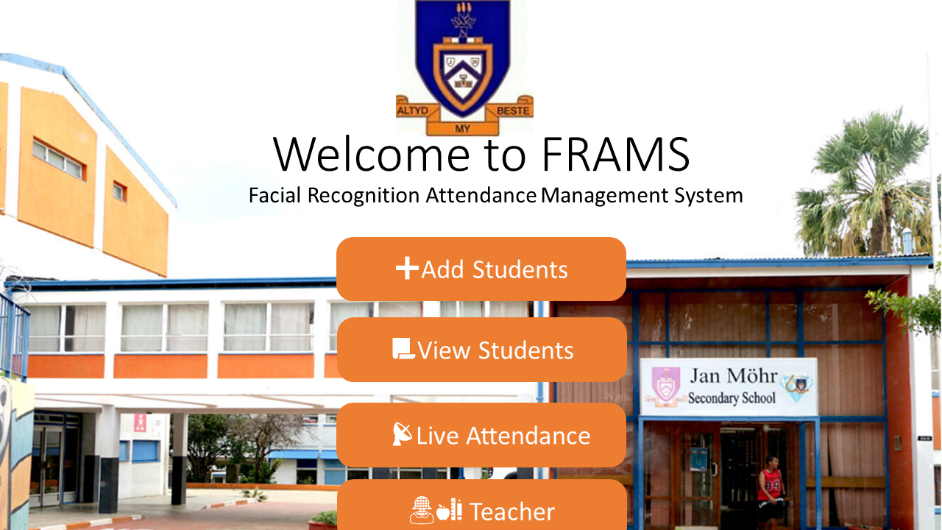


## **Method of solution**

The **Facial Recognition Attendance Management System (FRAMS)** is a custom‑built, computerized system designed to automate, secure, and streamline the process of recording student attendance in schools. The solution directly addresses the limitations of manual and semi‑manual attendance methods by leveraging **artificial intelligence** and **facial recognition technology** to accurately verify a student’s presence in real time.

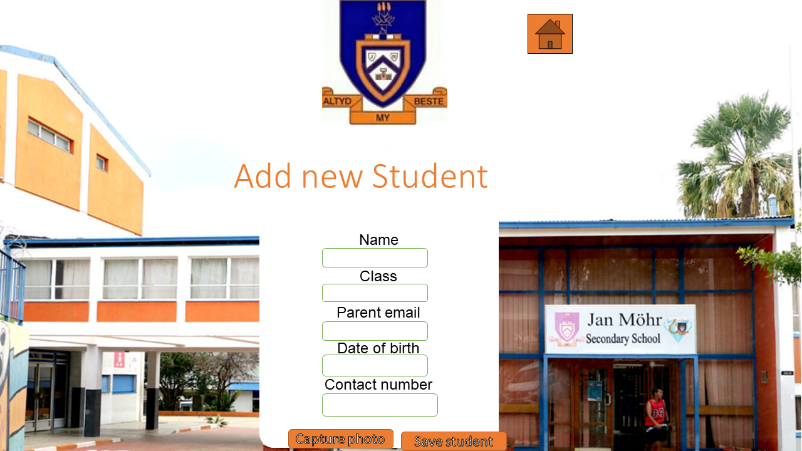
At its core, FRAMS uses a **web‑based interface** accessible on the school’s existing computers or laptops. It integrates with a camera or webcam to capture live facial images of students as they enter a classroom. The system compares each captured image to pre‑registered facial profiles stored in its secure database, and upon a successful match, automatically updates the attendance record for that student. This process takes less than a second per student, allowing the entire class attendance to be completed within moments, even in large groups.

**Registration and Setup**



Before FRAMS can be used for daily attendance, all students must be formally registered in the school’s database. This includes capturing each student’s:

* Full name and unique student ID
* Class allocation (grade, subject groups, or sections)
* Facial profile images taken from multiple angles
* Additional demographic details (optional, e.g., contact information for reports)



Once this page is launched, it should ask the user permission to access their webcam, once permission is granted, the students face is captured after all the necessary information is filled out

This registration process ensures the school maintains an **accurate and up‑to‑date centralised record** of every enrolled student. It also reinforces the integrity of the wider school system by requiring proper enrolment before attendance can be marked.

**Login and Access Control**



Teachers and authorised staff log in to FRAMS using unique usernames and secure passwords. Access levels can be configured so that:

* Teachers can only view and update attendance for their own classes
* Administrators can access and analyse attendance data across the school
* System operators can manage hardware, update student records, and configure facial recognition settings

This layered access structure protects sensitive data and ensures compliance with privacy requirements.

**Attendance Capture Methods**

****

This is where the student’s face will appear after the webcam is activated

This part will display a list of students that are present in that particular class

FRAMS offers two attendance modes:

1. **Live Facial Recognition Mode** – The system uses the connected camera to continuously scan and detect faces in real time. Recognised students are automatically marked as “Present,” with a timestamp recorded. Multiple students can be detected in a single frame, enabling extremely fast attendance marking.
2. **Manual Override Mode** – In cases where a student’s face is not recognised (e.g., wearing a mask, lighting issues), the teacher can manually mark the student as Present, Absent, or Late with a single click. This ensures flexibility and prevents disruption to lessons.

**Data Processing and Storage**

Captured attendance data is processed instantly and stored in the system’s secure database or local storage, with the following details:

* Student ID
* Date and time of attendance marking
* Attendance status (Present, Absent, Late)
* Teacher or staff member who recorded it
* Method of marking (Automatic via recognition / Manual)

These records are indexed for fast searching and retrieval, supporting quick queries such as:

* All absences for a given date
* Attendance history for a specific student
* Monthly attendance summaries for a class

**Reports and Outputs**

FRAMS can generate detailed attendance reports, which may include:

* Daily, weekly, and monthly summaries
* Class and subject attendance patterns
* Individual student attendance history
* Statistical insights such as absence percentages and punctuality trends

Reports can be viewed on‑screen, printed, or exported for further analysis. Colour‑coded dashboards provide teachers with an at‑a‑glance view of current class attendance.

**Security and Data Integrity**

Security is integral to FRAMS. All facial data is stored securely, with encryption applied where possible to protect biometric information. Role‑based access prevents unauthorised data viewing or editing. System activity logs ensure any changes are traceable for accountability.

**Integration and Scalability**

FRAMS can be scaled to suit a single class, a whole school, or even multiple campuses. The modular design allows future integration with:

* School management systems
* Student performance tracking tools
* Automated parent notification systems for absences

**Overall Benefits**

By replacing slow, error‑prone manual methods with automated recognition, FRAMS:

* Saves significant classroom time
* Provides highly accurate, verifiable records
* Eliminates proxy attendance fraud
* Encourages proper registration of all students in the school system
* Generates real‑time insights for better decision‑making

In essence, FRAMS is not just an attendance tool — it is a complete **digital attendance ecosystem**, ensuring efficiency, accuracy, and integrity in school record‑keeping while setting a strong foundation for broader digital transformation.

## Hardware requirements

The **Facial Recognition Attendance Management System (FRAMS)** requires a reliable combination of hardware components to ensure smooth operation, accurate facial recognition, and secure data storage. All devices listed below are directly necessary for the running of the system — there are no redundant or unnecessary items included. Each device serves a specific function in capturing, processing, storing, or displaying attendance information.

**Desktop or Laptop Computer**

The main platform for running FRAMS will be a desktop or laptop computer located in each classroom or at a designated attendance station.  
**Reason 1:** Provides sufficient processing power and memory to run the facial recognition software in real time without delays.  
**Reason 2:** Acts as the control hub, allowing the teacher to log in, view live recognition results, and generate attendance reports instantly.

**Webcam or HD Camera**

A webcam or high-definition camera is essential for capturing live images of students as they enter the classroom.  
**Reason 1:** Provides the video feed required for the AI-powered recognition algorithm to detect and match student faces accurately.  
**Reason 2:** Ensures image clarity and resolution suitable for accurate recognition under varying classroom lighting conditions.

**Local Server or Secure Storage Device**

Depending on the size of the school, attendance data and biometric templates will be stored securely either on a local server or a dedicated storage drive.  
**Reason 1:** Provides a secure, centralised location for all attendance records, ensuring data integrity and availability.  
**Reason 2:** Allows for regular backups to prevent data loss in case of device failure.

**Reliable Network Connection (LAN/Wi‑Fi)**

A stable wired or wireless network connection supports FRAMS functionality.  
**Reason 1:** Allows attendance data to sync in real time with the school’s administrative system for instant updates.

**Reason 2:** Enables secure communication between the classroom device and the central database for reporting and analysis.

**UPS (Uninterruptible Power Supply)**

A UPS is required to keep the computer and camera running during short power outages.  
**Reason 1:** Prevents disruption of the attendance process caused by unexpected power cuts.  
**Reason 2:** Protects sensitive hardware from damage due to sudden shutdowns or power surges.

|  |  |  |
| --- | --- | --- |
| **Hardware Device** | Reason 1 | Reason 2 |
| **Desktop/Laptop Computer** | Runs FRAMS software and recognition engine | Allows teachers to log in, monitor recognition, and generate reports |
| **Webcam/HD Camera** | Captures video feed for recognition | Ensures clear images for accurate detection |
| **Local Server / Secure Storage** | Centralised, secure attendance data storage | Enables regular backups and data protection |
| **Network Connection** | Syncs attendance data in real time | Connects devices to central database |
| **UPS (Power Backup)** | Maintains operation during short outages | Protects devices from sudden power loss |

**Hardware Summary Table**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## **Software requirements**

The successful implementation and operation of the Facial Recognition Attendance Management System (FRAMS) requires specific software components. Each software item listed below has been selected to ensure smooth performance, strong security, and full compatibility with the system’s objectives. There are no unnecessary items — all software directly supports the core functionality of FRAMS.

**1. Operating System**

The recommended operating system for FRAMS is **Windows 10 or Windows 11**.  
**Reason 1:** It provides a stable and reliable environment with built‑in support for modern web browsers, camera drivers, and AI processing libraries needed for facial recognition.  
**Reason 2:** Windows is widely used in educational institutions, which ensures compatibility with existing school infrastructure and makes it easier for staff to adapt to the system.

**2. Application Software**

The core FRAMS interface will run within a **modern web browser** such as **Microsoft Edge** or **Google Chrome**. F**ace‑api.js** (Version 0.22.2)  
Edge and Chrome provide the WebRTC and WebGL capabilities needed for in‑browser real‑time face recognition. **face‑api.js v stable release for0.22.2** is the most accurate facial detection and matching without server‑side processing.

**Reason 1:** These browsers support the latest web technologies (HTML5, JavaScript, WebRTC) which are essential for real‑time camera access and face recognition processing in the browser.  
**Reason 2:** They allow FRAMS to be platform‑independent on the front‑end, enabling easy updates and eliminating the need for complex local installations on every machine.

Additionally, **JavaScript runtime environments** (such as Node.js) may be used on the backend if the system is extended for centralised hosting, enabling smooth handling of data processing requests.

**Anti‑Virus Software**

A reputable anti‑virus program (e.g., **Microsoft Defender** or **Avast**) is essential.  
**Reason 1:** It safeguards the system from malware or viruses that could compromise attendance records or disrupt the recognition software.  
**Reason 2:** It protects both student data and biometric information from cyber threats, ensuring compliance with privacy and security policies.

**4. Drivers and Supporting Software**

**Camera drivers** and related utilities must be installed for the webcam or camera hardware to function correctly.  
**Reason 1:** Drivers enable the operating system and browser to communicate directly with the camera, ensuring smooth video feed capture for the recognition process.  
**Reason 2:** Updated drivers improve compatibility, performance, and image quality, which directly impacts the accuracy of the facial recognition algorithm.

Logitech Capture.11) — for Logitech webcams like the C920 series

**NVIDIA GeForce** (Version 552. Game Ready Driver44) — if using NVIDIA GPUs for accelerated image processing

**Intel Graphics Driver** (Version for Intel integrated 31.0.101.5333) — graphics  
These ensure webcams deliver stable, high‑definition video to the browser, and that GPU acceleration works correctly for AI tasks.

Where needed, **graphics drivers** should also be up‑to‑date to support accelerated image processing.

**Software Summary Table**

|  |  |  |
| --- | --- | --- |
| **Software Type** | **Technical Specification** | **Reasons for use** |
| **Operating System** | Microsoft Windows 11 Pro (64-bit) | -Stable, secure environment with camera & AI support  - Widely used in schools for compatibility |
| **Application Software** | Microsoft stack; strong security Edge v126+ / Google Chrome v126+ + face‑api.js v0.22.2 | -Supports modern web tech for recognition  -Easy updates, no complex installation |
| **Anti‑Virus 6** | Defender AV (Platform Avast Premium Security 4.18.24050.7) / v24. | -Protects from malware & cyber threats  -Secures sensitive biometric and attendance data |
| **Drivers** | Logitech + NVIDIA GeForce GRD v552.44 / Intel Graphics Driver v31.0.101.5333 | - Ensures HD webcam output and optimised GPU acceleration for facial recognition performance. |

# Programing code

Below are codes for all my pages;

**This is the launching script:**

{

    // Use IntelliSense to learn about possible attributes.

    // Hover to view descriptions of existing attributes.

    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387

    "version": "0.2.0",

    "configurations": [

        {

            "type": "pwa-chrome",

            "request": "launch",

            "name": "Open index.html",

            "file": "c:\\Users\\S.S pc1\\Desktop\\FRAMS\\public\\index.html"

        }

    ]

}

**The Landing Page:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>FRAMS Dashboard</title>

  <link rel="stylesheet" href="style.css">

</head>

<body>

  <header class="center-header">

    <img src="assets/logo.jpeg.jpg" alt="FRAMS Logo" class="logo">

    <h1>Welcome to FRAMS</h1>

    <p>Facial Recognition Attendance Management System</p>

  </header>

  <main class="center-buttons">

    <button onclick="location.href='add-student.html'">➕ Add Student</button>

    <button onclick="location.href='student-list.html'">📜 View Students</button>

    <button onclick="location.href='live-attendance.html'">📡 Live Attendance</button>

    <button onclick="location.href='Login-signup.html'"> 🧑🏾‍🏫Teacher </button>

  </main>

</body>

</html>

<!-- Your Script -->

<script defer src="models/face-api.min.js"></script>

<script defer src="app.js"></script>

**The landing page’s script:**

console.log("Hello from FRAMS!");

git addconst express = require('express');

const app = express();

app.get('/', (req, res) => {

  res.send('Welcome to FRAMS!');

});

app.listen(3000, () => {

  console.log('Server is running on port 3000');

});vvss

**Login-Signup.**

<!-- index.html, styles css and java script -->

<!-- This page handles login for existing teachers and signup for new teachers -->

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Teacher Dashboard</title>

  <link rel="stylesheet" href="style.css">

  <style>

    .center-header {

      text-align: center;

      padding: 2rem 1rem;

      color: black;

    }

    .center-h1 {

      text-align: center;

      font-size: 2rem;

      margin-top: 1rem;

      color: black;

    }

    .center-buttons {

      display: flex;

      flex-direction: column;

      align-items: center;

      gap: 1rem;

      margin-top: 2rem;

    }

    .center-buttons button {

      background: orange;

      color: #fff;

      border: none;

      padding: 0.8rem 1.5rem;

      font-size: 1rem;

      border-radius: 8px;

      cursor: pointer;

      width: 80%;

      max-width: 300px;

      transition: background 0.3s;

    }

    .center-buttons button:hover {

      background: dodgerblue;

    }

    .back-button {

      background: orange;

      margin-top: 2rem;

    }

    .back-button:hover {

      background: dodgerblue;

    }

  </style>

</head>

<body>

    <a href="index.html" class="home-button">🏠Home</a>

  <header class="center-header">

    <img src="assets/logo.jpeg.jpg" alt="FRAMS Logo" class="logo">

    <h1>Welcome to FRAMS</h1>

    <p>Facial Recognition Attendance Management System</p>

  </header>

  <h1 class="center-h1">Teacher </h1>

  <div class="center-buttons">

    <button onclick="location.href='teacher-signup.html'">📝 Sign Up</button>

    <button onclick="location.href='teacher-login.html'">🔐 Log In</button>

  </div>

</body>

</html>

<!-- index.html -->

<!-- This page handles login for existing teachers and signup for new teachers -->

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>FRAMS Dashboard</title>

  <link rel="stylesheet" href="style.css">

  <style>

    /\* Top-right nav buttons \*/

    .nav-buttons {

      position: absolute;

      top: 20px;

      right: 20px;

      display: flex;

      gap: 10px;

    }

    .nav-button {

      background-color: orange;

      color: #333;

      padding: 10px 20px;

      border-radius: 25px;

      text-decoration: none;

      font-weight: bold;

      box-shadow: 0 4px 6px rgba(0,0,0,0.1);

      transition: background-color 0.3s ease, transform 0.2s ease;

    }

    .nav-button:hover {

      background-color: orange;

      transform: scale(1.05);

    }

    /\* Centered form container \*/

    .form-wrapper {

      display: flex;

      justify-content: center;

      align-items: center;

      height: 100vh;

      flex-direction: column;

    }

    .form-container {

      width: 360px;

      padding: 2rem;

      background: #fff;

      border-radius: 8px;

      box-shadow: 0 6px 12px rgba(0,0,0,0.1);

      text-align: center;

    }

    .form-container h2 {

      margin-bottom: 1rem;

    }

    .form-container input {

      width: 100%;

      padding: 10px;

      margin: 8px 0;

      border-radius: 5px;

      border: 1px solid #ccc;

      box-sizing: border-box;

    }

    .form-container button {

      width: 100%;

      padding: 12px;

      margin-top: 12px;

      background-color: orange;

      color: #fff;

      border: none;

      border-radius: 5px;

      cursor: pointer;

      font-size: 1rem;

    }

    .form-container button:hover {

      background-color: #005fa3;

    }

    .toggle-container {

      text-align: left;

      font-size: 0.9rem;

      color: #555;

      margin: -4px 0 12px;

    }

    .forgot-link {

      display: block;

      margin-top: 12px;

      font-size: 0.9rem;

      color: dodgerblue;

      text-decoration: none;

    }

    .forgot-link:hover {

      text-decoration: underline;

    }

    .center-header {

      text-align: center;

      margin-bottom: 2rem;

    }

    .logo {

      display: block;

      margin: 0 auto 1rem auto;

      max-height: 120px;

    }

  </style>

</head>

<body>

  <!-- Top-right buttons -->

  <div class="nav-buttons">

    <a href="index.html" class="nav-button">🏠 Home</a>

    <a href="Login-signup.html" class="nav-button">🔙 Back</a>

  </div>

  <!-- Centered form and header -->

  <div class="form-wrapper">

    <header class="center-header">

      <img src="assets/logo.jpeg.jpg" alt="FRAMS Logo" class="logo">

      <h1>Welcome to FRAMS</h1>

      <p>Facial Recognition Attendance Management System</p>

    </header>

    <div class="form-container">

      <h2>🔐 Log In</h2>

      <form id="loginForm">

        <input type="text" id="username" placeholder="Username" required>

        <input type="password" id="password" placeholder="Password" required>

        <div class="toggle-container">

          <label>

            <input type="checkbox" id="togglePassword"> Show Password

          </label>

        </div>

        <button type="submit">Log In</button>

        <a href="forgot-password.html" class="forgot-link">Forgot Password?</a>

      </form>

    </div>

  </div>

  <script>

  // Show/hide password

  document.getElementById("togglePassword").addEventListener("change", function() {

    document.getElementById("password").type = this.checked ? "text" : "password";

  });

  // Login logic

  document.getElementById("loginForm").addEventListener("submit", function(e) {

    e.preventDefault();

    const username = document.getElementById("username").value.trim();

    const pwd = document.getElementById("password").value.trim();

    const teachers = JSON.parse(localStorage.getItem("teachers")) || [];

    const teacher = teachers.find(t => t.username === username && t.password === pwd);

    if (teacher) {

      localStorage.setItem("currentTeacher", JSON.stringify(teacher));

      alert("Welcome, " + teacher.Surname.Name + "!");

      location.href = "teacher.html"; // ✅ Redirect to teacher.html

    } else {

      alert("❌ Invalid credentials. Try again or click 'Forgot Password?'");

    }

  });

</script>

</body>

</html>

<script defer src="app.js"></script>

**The login page:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>FRAMS – Teacher Login</title>

  <link rel="stylesheet" href="style.css">

  <style>

  /\* Top-right nav buttons \*/

  .nav-buttons {

    position: absolute;top: 20px;right: 20px;display: flex;gap: 10px;

  }

  .nav-button{

   background-color: orange;color: #333;padding: 10px 20px;

   border-radius: 25px;

   text-decoration: none;

   font-weight: bold;

   box-shadow: 0 4px 6px rgba(0,0,0,0.1);

   transition: background-color 0.3s ease,transform 0.2s ease;

  }

   .nav-button:hover

    {background-color: orange;

    transform: scale(1.05);

  }

   /\* Centered form wrapper \*/

.form-wrapper {

  display: flex;

  justify-content: center;

  align-items: center;

  height: 100vh;

  flex-direction: column;

  padding: 1rem; /\* added breathing space so it doesn't stick to edges \*/

}

/\* Form container - more horizontal \*/

.form-container {

  width: 550px;                /\* wider but not full page \*/

  max-width: 90%;              /\* ensures it stays good on smaller screens \*/

  padding: 2rem;

  background: #fff;

  border-radius: 8px;

  box-shadow: 0 6px 12px rgba(0,0,0,0.1);

  text-align: center;

}

     .form-container h2{

      margin-bottom: 1rem;

     }

     .form-container input,

     .form-container select {

      width: 60%;

     padding: 10px;

     margin: 8px 0;

     border-radius: 5px;

     border: 1px solid #ccc;

     box-sizing: border-box;

    }

     .form-container button {

      width: 60%;

      padding: 12px;

      margin-top: 12px;

      background-color: orange;

      color: #fff;

      border: none;

      border-radius: 5px;

      cursor: pointer;

      font-size: 1rem;

      }

      .form-container button:hover {

        background-color: #005fa3;

        }

        .center-header

        {text-align: center;

        margin-bottom: 2rem;

      }

        .logo {

          display: block;

        margin: 0 auto 1rem auto;

        max-height: 120px;

      }

        </style>

        </head>

        <body>

<!-- Top-right buttons -->

<div class="nav-buttons">

  <a href="index.html" class="nav-button">🏠 Home</a><a href="Login-signup.html" class="nav-button">🔙 Back</a>

</div>

  </style>

</head>

<body>

  <!-- nav pills -->

  <div class="nav-buttons">

    <a href="index.html" class="nav-button">🏠 Home</a>

    <a href="Login-signup.html" class="nav-button">🔙 Back</a>

  </div>

  <div class="form-wrapper">

    <div class="form-container">

      <header class="center-header">

        <img src="assets/logo.jpeg.jpg" alt="FRAMS Logo" class="logo">

        <h1>Welcome to FRAMS</h1>

        <p>Facial Recognition Attendance Management System</p>

      </header>

      <h2>🔐 Log In</h2>

      <form id="loginForm">

        <input type="text"     id="username" placeholder="Username" required>

        <input type="password" id="password" placeholder="Password" required>

        <!-- Register Class as text input -->

        <input

          type="text"

          id="classId"

          placeholder="Register Class (e.g. 11A)"

          required

        >

        <div class="toggle-container">

          <label>

            <input type="checkbox" id="togglePassword">

            Show Password

          </label>

        </div>

        <button type="submit">Log In</button>

        <a href="forgot-password.html" class="forgot-link">Forgot Password?</a>

      </form>

    </div>

  </div>

  <script>

    // Password toggle

    document.getElementById("togglePassword")

      .addEventListener("change", function() {

        document.getElementById("password").type =

          this.checked ? "text" : "password";

      });

    // Login logic

    document.getElementById("loginForm")

      .addEventListener("submit", function(e) {

        e.preventDefault();

        const username = document.getElementById("username").value.trim();

        const pwd      = document.getElementById("password").value.trim();

        const classId  = document.getElementById("classId").value.trim();

        const teachers = JSON.parse(localStorage.getItem("teachers")) || [];

        const teacher = teachers.find(t =>

          t.username === username &&

          t.password === pwd

        );

        if (!teacher) {

          return alert("❌ Invalid credentials.");

        }

        teacher.classId = classId;

        localStorage.setItem("currentTeacher", JSON.stringify(teacher));

        alert(`Welcome, ${teacher.title}.${teacher.firstName}!\nClass: ${classId}`);

        window.location.href = "teacher.html";

      });

  const passwordInput = document.querySelector('input[name="password"]');

  const passwordHelp = document.getElementById('passwordHelp');

  passwordInput.addEventListener('input', () => {

    if (passwordInput.value.length < 8) {

      passwordHelp.textContent = "Too short — must be at least 8 characters.";

      passwordHelp.style.color = "red";

    } else if (passwordInput.value.length > 20) {

      passwordHelp.textContent = "Too long — maximum is 20 characters.";

      passwordHelp.style.color = "red";

    } else {

      passwordHelp.textContent = "Looks good ✅";

      passwordHelp.style.color = "green";

    }

  });

</script>

  </script>

</body>

</html>

**Sign up page:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>FRAMS Dashboard – Sign Up</title>

  <link rel="stylesheet" href="style.css">

  <style>

    /\* Top-right nav buttons \*/

  .nav-buttons {

    position: absolute;top: 20px;right: 20px;display: flex;gap: 10px;

  }

  .nav-button{

   background-color: orange;color: #333;padding: 10px 20px;

   border-radius: 25px;

   text-decoration: none;

   font-weight: bold;

   box-shadow: 0 4px 6px rgba(0,0,0,0.1);

   transition: background-color 0.3s ease,transform 0.2s ease;

  }

   .nav-button:hover

    {background-color: orange;

    transform: scale(1.05);

  }

    /\* Centered form container \*/

    .form-wrapper {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    flex-direction: column;

  }

    .form-container {

      width: 360px;

    padding: 2rem;

    background: #fff;

    border-radius: 8px;

    box-shadow: 0 6px 12px rgba(0,0,0,0.1);

     text-align: center;}

     .form-container h2{

      margin-bottom: 1rem;

     }

     .form-container input,

     .form-container select {

      width: 100%;

     padding: 10px;

     margin: 8px 0;

     border-radius: 5px;

     border: 1px solid #ccc;

     box-sizing: border-box;

    }

     .form-container button {

      width: 100%;

      padding: 12px;

      margin-top: 12px;

      background-color: orange;

      color: #fff;

      border: none;

      border-radius: 5px;

      cursor: pointer;

      font-size: 1rem;

      }

      .form-container button:hover {

        background-color: #005fa3;

        }

        .center-header

        {text-align: center;

        margin-bottom: 2rem;

      }

        .logo {

          display: block;

        margin: 0 auto 1rem auto;

        max-height: 120px;

      }

    /\* Top-right nav buttons \*/

    .nav-buttons {

      position: absolute;

      top: 20px;

      right: 20px;

      display: flex;

      gap: 10px;

    }

    .nav-button {

      background-color: orange;

      color: #333;

      padding: 6px 14px;

      border-radius: 20px;

      text-decoration: none;

      font-size: 0.85rem;

      font-weight: bold;

      box-shadow: 0 3px 5px rgba(0,0,0,0.1);

      transition: background-color 0.2s, transform 0.15s;

    }

    .nav-button:hover {

      background-color: darkorange;

      transform: scale(1.05);

    }

 /\* Centered form wrapper \*/

.form-wrapper {

  display: flex;

  justify-content: center;

  align-items: center;

  height: 100vh;

  flex-direction: column;

  padding: 1rem; /\* added breathing space so it doesn't stick to edges \*/

}

/\* Form container - more horizontal \*/

.form-container {

  width: 600px;                /\* wider but not full page \*/

  max-width: 90%;              /\* ensures it stays good on smaller screens \*/

  padding: 2rem;

  background: #fff;

  border-radius: 8px;

  box-shadow: 0 6px 12px rgba(0,0,0,0.1);

  text-align: center;

}

    .form-container h2 {

      margin-bottom: 1rem;

    }

    .form-container input,

    .form-container select {

      width: 100%;

      padding: 10px;

      margin: 8px 0;

      border: 1px solid #ccc;

      border-radius: 5px;

      box-sizing: border-box;

      font-size: 1rem;

    }

    .form-container button {

      width: 100%;

      padding: 12px;

      margin-top: 12px;

      background-color: orange;

      color: #fff;

      border: none;

      border-radius: 5px;

      cursor: pointer;

      font-size: 1rem;

    }

    .form-container button:hover {

      background-color: #005fa3;

    }

    .toggle-container {

      text-align: left;

      font-size: 0.9rem;

      margin: -4px 0 12px;

    }

    .username-display {

      margin-bottom: 10px;

      font-size: 0.9rem;

      height: 1.2em;

    }

    .center-header {

      margin-bottom: 2rem;

      text-align: center;

    }

    .logo {

      display: block;

      margin: 0 auto 1rem;

      max-height: 120px;

    }

  </style>

</head>

<body>

  <!-- nav pills -->

  <div class="nav-buttons">

    <a href="index.html" class="nav-button">🏠 Home</a>

    <a href="Login-signup.html" class="nav-button">🔙 Back</a>

  </div>

  <div class="form-wrapper">

    <div class="form-container">

      <header class="center-header">

        <img src="assets/logo.jpeg.jpg" alt="FRAMS Logo" class="logo">

        <h1>Welcome to FRAMS</h1>

        <p>Facial Recognition Attendance Management System</p>

      </header>

      <h2>📝 Sign Up</h2>

      <form id="signupForm">

        <label for="title">Title</label>

        <select id="title" required>

          <option value="">Select Title</option>

          <option value="Mr">Mr.</option>

          <option value="Mrs">Mrs.</option>

          <option value="Ms">Ms.</option>

          <option value="Dr">Dr.</option>

        </select>

        <input type="text" id="firstName" placeholder="First Name" required>

        <input type="text" id="surname"   placeholder="Surname"    required>

        <!-- Register Class as text input -->

        <input

          type="text"

          id="classId"

          placeholder="Register Class (e.g. 11A)"

          required

        >

        <input

  type="password"

  name="password"

  placeholder="Enter password"

  minlength="8"

  maxlength="20"

  required

>

<small id="passwordHelp" style="color: rgb(247, 6, 6); display: block; margin-top: 5px;">

  Password must be between 8–20 characters.

</small>

        <label>

            <input type="checkbox" id="toggleSignupPassword">

            Show Password

          </label>

        </div>

        <div class="username-display" id="usernameDisplay"></div>

        <button type="submit">Sign Up</button>

      </form>

    </div>

  </div>

  <script>

    // Password toggle

    document.getElementById("toggleSignupPassword")

      .addEventListener("change", function() {

        document.getElementById("signupPassword").type =

          this.checked ? "text" : "password";

      });

    // Live username preview

    function updateUsername() {

      const title     = document.getElementById("title").value;

      const firstName = document.getElementById("firstName").value.trim();

      const display   = document.getElementById("usernameDisplay");

      if (title && firstName) {

        display.textContent = "Your username will be: " +

          `${title}.${firstName}`.replace(/\s+/g, '');

      } else {

        display.textContent = "";

      }

    }

    document.getElementById("firstName").addEventListener("input", updateUsername);

    document.getElementById("title").addEventListener("change", updateUsername);

    // Handle signup

    document.getElementById("signupForm")

      .addEventListener("submit", function(e) {

        e.preventDefault();

        const title     = document.getElementById("title").value;

        const firstName = document.getElementById("firstName").value.trim();

        const surname   = document.getElementById("surname").value.trim();

        const classId   = document.getElementById("classId").value.trim();

        const password  = document.getElementById("signupPassword").value;

        const username  = `${title}.${firstName}`.replace(/\s+/g, '');

        const teacher = { title, firstName, surname, classId, password, username };

        let teachers = JSON.parse(localStorage.getItem("teachers")) || [];

        teachers.push(teacher);

        localStorage.setItem("teachers", JSON.stringify(teachers));

        localStorage.setItem("currentTeacher", JSON.stringify(teacher));

        alert(`✅ Profile created!\nUsername: ${username}\nClass: ${classId}`);

        window.location.href = "teacher.html";

      });

  </script>

</body>

</html>

**Adding a student:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>FRAMS Dashboard</title>

  <link rel="stylesheet" href="style.css">

  <script defer src="add-student.js"></script>

</head>

<body>

  <header class="center-header">

    <img src="assets/logo.jpeg.jpg" alt="FRAMS Logo" class="logo">

    <h1>Welcome to FRAMS</h1>

    <p>Facial Recognition Attendance Management System</p>

<body>

  <a href="index.html" class="home-button">🏠Home</a>

  <header>

    <img src="asstes/logo.jpeg.jpeg" alt="">

  </header>

  <main>

    <div class="form-card">

      <h2>Add New Student</h2>

      <form id="add-student-form">

        <label for="student-name">Name</label>

        <input type="text" id="student-name" required>

        <label for="student-class">Class</label>

        <input type="text" id="student-class" required>

        <label for="parent-email">Parent Email</label>

        <input type="email" id="parent-email" required>

        <!-- Date of Birth Field -->

<label for="dob">Date of Birth</label>

<input id="dob" type="date" pattern="\d{4}/\d{2}/\d{2}" required />

<small>Format: YYYY/MM/DD</small>

<!-- Parent Contact Number Field -->

<label for="parentContact">Parent Contact Number</label>

<input id="parentContact" type="tel" pattern="^\+264\d{8}$" placeholder="+264XXXXXXXX" required />

<small>Format: +264 followed by 8 digits</small>

        <div class="camera">

          <video id="enroll-webcam" autoplay muted></video>

          <canvas id="enroll-snapshot" class="hidden"></canvas>

        </div>

        <button type="button" id="capture-enroll">📸 Capture Photo</button>

        <button type="submit">💾 Save Student</button>

      </form>

    </div>

    <div class="form-card">

      <h2>Enrolled Students</h2>

      <ul id="student-list"></ul>

    </div>

  </main>

</body>

</html>

**Viewing students:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>All Students</title>

  <link rel="stylesheet" href="style.css">

  <script defer src="list.js"></script>

</head>

<body>

  <a href="index.html" class="home-button">🏠Home</a>

  <!-- Consistent Header -->

  <header class="center-header">

    <img src="assets/logo.jpeg.jpg" alt="FRAMS Logo" class="logo">

    <h1>FRAMS</h1>

    <p>Facial Recognition Attendance Management System</p>

  </header>

  <main>

    <div class="form-card">

      <h2>Enrolled Students</h2>

      <ul id="student-list"></ul>

    </div>

  </main>

</body>

</html>

**The style for some pages:**

/\* Reset + base \*/

\* { box-sizing: border-box; margin:0; padding:0; }

body {

  font-family: Arial, sans-serif;

  color: #333;

  background: url("assets/bg.png.png") no-repeat center center fixed;

  background-size: cover;

}

/\* Main container \*/

main {

  max-width: 600px;

  margin: 2rem auto;

  background: rgba(255,255,255,0.9);

  border-radius: 6px;

  padding: 1.5rem;

}

/\* Form \*/

form label { display: block; margin-bottom: 0.5rem; font-weight: bold; }

input {

  width: 100%; padding: 0.5rem; margin-bottom: 1rem;

  border:1px solid #fff(204, 204, 204); border-radius:4px;

}

/\* Buttons \*/

button {

  background: #FFA500; color:#fff; border:#333;

  padding:0.6rem 1.2rem; border-radius:4px; cursor:pointer;

  transition: background 0.2s;

}

button:hover { background:dodgerblue; }

/\* Camera & snapshot \*/

.camera { text-align:center; margin:1rem 0; }

video, canvas {

  width:320px; max-width:100%; border:1px solid #ccc; border-radius:4px;

}

.hidden { display:none; }

/\* Student list \*/

ul#student-list { list-style:none; padding:0; }

ul#student-list li {

  display:flex; align-items:center; margin-bottom:0.8rem;

}

ul#student-list img {

  width:60px; height:60px; border-radius:50%; object-fit:cover;

  margin-right:1rem;

}

/\* Centered form card \*/

.form-card {

  background: rgba(255, 255, 255, 0.95);

  padding: 2rem;

  border-radius: 10px;

  box-shadow: 0 8px 24px rgba(0,0,0,0.1);

  margin: 2rem auto;

  max-width: 500px;

}

.form-card h2 {

  text-align: center;

  margin-bottom: 1rem;

  color: #FFA500;

}

/\* Logo & Header \*/

.center-header {

  text-align: center;

  padding: 2rem 1rem;

  color:#333;

}

.logo {

  display: block;

  margin: 0 auto 1rem auto;

  max-width: 160px;

  height: auto;

}

/\* Button layout \*/

.center-buttons {

  display: flex;

  flex-direction: column;

  align-items: center;

  gap: 1rem;

  margin-top: 2rem;

}

.center-buttons button {

  background: #FFA500;

  color: #fff;

  border: none;

  padding: 1rem 2rem;

  font-size: 1.1rem;

  border-color: #333;

  border-radius: 8px;

  cursor: pointer;

  width: 80%;

  max-width: 300px;

  transition: background 0.3s;

}

.center-buttons button:hover {

  background: dodgerblue;

}

.center-buttons {

  display: flex;

  flex-direction: column;

  align-items: center;

  gap: 1rem;

  margin-top: 2rem;

}

.center-buttons button {

  background: #FFA500;

  color: #fff;

  border: none;

  padding: 1rem 2rem;

  font-size: 1.1rem;

  border-radius: 8px;

  cursor: pointer;

  width: 80%;

  max-width: 300px;

  transition: background 0.3s;

}

.center-buttons button:hover {

  background: dodgerblue;

}

.home-button,

.back-button {

  position: absolute;

  right: 20px;

  background-color: orange;

  color: #333;

  padding: 10px 20px;

  border-radius: 25px;

  text-decoration: none;

  font-weight: bold;

  box-shadow: 0 4px 6px rgba(0,0,0,0.1);

  transition: background-color 0.3s ease, transform 0.2s ease;

}

.home-button {

  top: 20px;

}

.back-button {

  top: 20px; /\* Just next to the Home button \*/

}

.home-button:hover,

.back-button:hover {

  background-color: orange;

  transform: scale(1.05);

}

/\* centered from\*/

.sign-up form{

 background: rgba(255, 255, 255, 0.95);

padding: 2rem;

border-radius: 10px;

box-shadow: 0 8px 24px rgba(0,0,0,0.1);

margin: 2rem auto;

max-width: 500px;

}

**The script for adding a new student:**

// — DOM refs —

const videoEl    = document.getElementById('enroll-webcam');

const canvasEl   = document.getElementById('enroll-snapshot');

const ctx        = canvasEl.getContext('2d');

const capBtn     = document.getElementById('capture-enroll');

const form       = document.getElementById('add-student-form');

const listEl     = document.getElementById('student-list');

let students = JSON.parse(localStorage.getItem('students') || '[]');

let snapshot = '';

// 1) Start webcam

navigator.mediaDevices.getUserMedia({ video:true })

  .then(stream => videoEl.srcObject = stream)

  .catch(err => alert('Camera error: '+err));

// 2) Capture photo

capBtn.addEventListener('click', () => {

  canvasEl.width  = videoEl.videoWidth;

  canvasEl.height = videoEl.videoHeight;

  ctx.drawImage(videoEl, 0, 0);

  snapshot = canvasEl.toDataURL('image/png');

  canvasEl.classList.remove('hidden');

});

// 3) Render student list

function renderList() {

  listEl.innerHTML = '';

  if (!students.length) {

    return listEl.innerHTML = '<li>No students enrolled.</li>';

  }

  students.forEach(s => {

    const li  = document.createElement('li');

    const img = document.createElement('img');

    img.src   = s.face || 'images/placeholder.png';

    const div = document.createElement('div');

    div.innerHTML = `<strong>${s.name}</strong><br>${s.class}<br><em>${s.parentEmail}</em>`;

    li.append(img, div);

    listEl.append(li);

  });

}

// 4) Handle form submit

form.addEventListener('submit', e => {

  e.preventDefault();

  const name  = document.getElementById('student-name').value.trim();

  const cls   = document.getElementById('student-class').value.trim();

  const email = document.getElementById('parent-email').value.trim();

  if (!name||!cls||!email) return alert('Fill all fields.');

  if (!snapshot) return alert('Capture a photo first.');

  const newStu = { id:Date.now(), name, class:cls, parentEmail:email, face:snapshot };

  students.push(newStu);

  localStorage.setItem('students', JSON.stringify(students));

  alert(`Saved ${name}!`);

  form.reset();

  canvasEl.classList.add('hidden');

  snapshot = '';

  renderList();

});

// 5) On load

window.addEventListener('DOMContentLoaded', renderList);

document.getElementById('signupForm').addEventListener('submit', function(e) {

  const dob = document.getElementById('dob').value;

  const contact = document.getElementById('parentContact').value;

  const dobPattern = /^\d{4}\/\d{2}\/\d{2}$/;

  const contactPattern = /^\+264\d{8}$/;

  if (!dobPattern.test(dob)) {

    e.preventDefault();

    alert("Date of Birth must be in YYYY/MM/DD format.");

    return;

  }

  if (!contactPattern.test(contact)) {

    e.preventDefault();

    alert("Parent contact number must start with +264 and be followed by 8 digits.");

    return;

  }

});

**The forgot password page:**

<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Forgot Password – FRAMS</title>

  <link rel="stylesheet" href="style.css">

  <style>

    /\* Same styling as before \*/

    .form-wrapper {

      display: flex;

      justify-content: center;

      align-items: center;

      height: 100vh;

      flex-direction: column;

    }

    .form-container {

      width: 360px;

      padding: 2rem;

      background: #fff;

      border-radius: 8px;

      box-shadow: 0 6px 12px rgba(0,0,0,0.1);

      text-align: center;

    }

    input, button {

      width: 100%;

      padding: 10px;

      margin: 8px 0;

      border-radius: 5px;

      border: 1px solid #ccc;

      box-sizing: border-box;

    }

    button {

      background-color: #007acc;

      color: white;

      border: none;

      cursor: pointer;

    }

    button:hover {

      background-color: #005fa3;

    }

    .result-message {

      margin-top: 12px;

      font-size: 0.95rem;

      color: #333;

    }

  </style>

</head>

<body>

  <div class="form-wrapper">

    <div class="form-container">

      <h2>🔐 Forgot Password</h2>

      <form id="forgotForm">

        <input type="email" id="email" placeholder="Enter your registered email" required>

        <button type="submit">Find Account</button>

      </form>

      <div class="result-message" id="resultMessage"></div>

    </div>

  </div>

  <script>

    document.getElementById("forgotForm").addEventListener("submit", function(e) {

      e.preventDefault();

      const emailInput = document.getElementById("email").value.trim();

      const resultMessage = document.getElementById("resultMessage");

      const teachers = JSON.parse(localStorage.getItem("teachers")) || [];

      const teacher = teachers.find(t => t.email === emailInput);

      if (teacher) {

        resultMessage.innerHTML = `

          ✅ Account found for <strong>${teacher.name} ${teacher.surname}</strong><br>

          Enter a new password below:

          <input type="password" id="newPassword" placeholder="New Password">

          <button onclick="resetPassword('${teacher.email}')">Reset Password</button>

        `;

      } else {

        resultMessage.textContent = "❌ Email not found. Please check and try again.";

      }

    });

    function resetPassword(email) {

      const newPwd = document.getElementById("newPassword").value.trim();

      if (!newPwd) return alert("Please enter a new password.");

      let teachers = JSON.parse(localStorage.getItem("teachers")) || [];

      const index = teachers.findIndex(t => t.email === email);

      if (index !== -1) {

        teachers[index].password = newPwd;

        localStorage.setItem("teachers", JSON.stringify(teachers));

        alert("✅ Password reset successfully!");

        location.href = "teacher-login.html";

      }

    }

  </script>

</body>

</html>

**This script is basically the entire system:**

// ─── GLOBALS & DOM NODES ──────────────────────────────────────

const navEnroll        = document.getElementById('nav-enroll');

const navRegister      = document.getElementById('nav-register');

const enrollSection    = document.getElementById('enroll-section');

const registerSection  = document.getElementById('register-section');

const enrollWebcam     = document.getElementById('enroll-webcam');

const enrollCanvas     = document.getElementById('enroll-snapshot');

const enrollCtx        = enrollCanvas.getContext('2d');

const captureEnrollBtn = document.getElementById('capture-enroll');

const addStudentForm   = document.getElementById('add-student-form');

const studentListEl    = document.getElementById('student-list');

const studentSelect    = document.getElementById('student-select');

const startCameraBtn   = document.getElementById('start-camera');

const regWebcam        = document.getElementById('reg-webcam');

const regCanvas        = document.getElementById('reg-snapshot');

const regCtx           = regCanvas.getContext('2d');

const captureFaceBtn   = document.getElementById('capture-face');

const registerFaceBtn  = document.getElementById('register-face');

let students     = [];   // { id, name, class, parentEmail, face, descriptor }

let enrollImage  = '';

let registerImage = '';

// ─── STORAGE HELPERS ─────────────────────────────────────────

function loadStudents() {

  const saved = localStorage.getItem('students');

  students = saved ? JSON.parse(saved) : [];

}

function saveStudents() {

  localStorage.setItem('students', JSON.stringify(students));

}

// ─── RENDER LIST & DROPDOWN ─────────────────────────────────

function renderStudentList() {

  studentListEl.innerHTML = '';

  if (students.length === 0) {

    studentListEl.innerHTML = '<li>No students enrolled.</li>';

    return;

  }

  students.forEach(s => {

    const li  = document.createElement('li');

    const img = document.createElement('img');

    img.src   = s.face || 'assets/placeholder.png';

    const info = document.createElement('div');

    info.innerHTML = `<strong>${s.name}</strong><br>${s.class}`;

    li.append(img, info);

    studentListEl.append(li);

  });

}

function populateStudentDropdown() {

  studentSelect.innerHTML = '';

  if (!students.length) {

    const opt = document.createElement('option');

    opt.textContent = 'No students';

    opt.disabled = true;

    studentSelect.append(opt);

    return;

  }

  students.forEach(s => {

    const opt = document.createElement('option');

    opt.value = s.id;

    opt.textContent = `${s.name} — ${s.class}`;

    studentSelect.append(opt);

  });

}

function initNav() {

  if (!navEnroll || !navRegister) return; // ✅ Prevent errors on pages without nav

  enrollSection.classList.remove('hidden');

  registerSection.classList.add('hidden');

  navEnroll.addEventListener('click', () => {

    enrollSection.classList.remove('hidden');

    registerSection.classList.add('hidden');

  });

  navRegister.addEventListener('click', () => {

    registerSection.classList.remove('hidden');

    enrollSection.classList.add('hidden');

    populateStudentDropdown();

  });

}

// ─── ENROLLMENT LOGIC ───────────────────────────────────────

function initEnrollment() {

  navigator.mediaDevices

    .getUserMedia({ video: true })

    .then(stream => enrollWebcam.srcObject = stream)

    .catch(err => alert('Camera error: ' + err));

  captureEnrollBtn.addEventListener('click', () => {

    enrollCtx.drawImage(enrollWebcam, 0, 0, enrollCanvas.width, enrollCanvas.height);

    enrollImage = enrollCanvas.toDataURL('image/png');

    enrollCanvas.classList.remove('hidden');

  });

  addStudentForm.addEventListener('submit', e => {

    e.preventDefault();

    const name  = document.getElementById('student-name').value.trim();

    const cls   = document.getElementById('student-class').value.trim();

    const email = document.getElementById('parent-email').value.trim();

    if (!name || !cls || !email) {

      return alert('All fields are required.');

    }

    if (!enrollImage) {

      return alert('Please capture a photo.');

    }

    const newStu = {

      id:          Date.now(),

      name,

      class:       cls,

      parentEmail: email,

      face:        enrollImage,

      descriptor:  null

    };

    students.push(newStu);

    saveStudents();

    alert('✅ Added ' + name);

    addStudentForm.reset();

    enrollCanvas.classList.add('hidden');

    enrollImage = '';

    renderStudentList();

  });

}

// ─── FACE REGISTRATION LOGIC ────────────────────────────────

function initRegistration() {

  // Load face-api.js models

  Promise.all([

    faceapi.nets.ssdMobilenetv1.loadFromUri('models'),

    faceapi.nets.faceLandmark68Net.loadFromUri('models'),

    faceapi.nets.faceRecognitionNet.loadFromUri('models')

  ]).catch(err => alert('Model load failed: ' + err));

  startCameraBtn.addEventListener('click', async () => {

    try {

      const stream = await navigator.mediaDevices.getUserMedia({ video: true });

      regWebcam.srcObject = stream;

      regWebcam.classList.remove('hidden');

      captureFaceBtn.classList.remove('hidden');

    } catch (err) {

      alert('Camera error: ' + err);

    }

  });

  captureFaceBtn.addEventListener('click', () => {

    regCtx.drawImage(regWebcam, 0, 0, regCanvas.width, regCanvas.height);

    registerImage = regCanvas.toDataURL('image/png');

    regCanvas.classList.remove('hidden');

    registerFaceBtn.classList.remove('hidden');

  });

  registerFaceBtn.addEventListener('click', async () => {

    const sid     = studentSelect.value;

    const student = students.find(s => s.id == sid);

    if (!student) {

      return alert('Select a student.');

    }

    const detection = await faceapi

      .detectSingleFace(regCanvas)

      .withFaceLandmarks()

      .withFaceDescriptor();

    if (!detection) {

      return alert('❌ No face detected. Try again.');

    }

    student.descriptor = Array.from(detection.descriptor);

    student.face       = registerImage;

    saveStudents();

    alert(`✅ Face registered for ${student.name}`);

    regCanvas.classList.add('hidden');

    registerFaceBtn.classList.add('hidden');

  });

}

// ─── INITIALIZE APP ────────────────────────────────────────

document.addEventListener('DOMContentLoaded', () => {

  loadStudents();

  initNav();

  renderStudentList();

  initEnrollment();

  initRegistration();

});

// live facial atttendance capture //

const video = document.getElementById('video');

    const log = document.getElementById('log');

    let labeledDescriptors = [];

    async function loadModels() {

      await faceapi.nets.tinyFaceDetector.loadFromUri('models');

      await faceapi.nets.faceRecognitionNet.loadFromUri('models');

      await faceapi.nets.faceLandmark68Net.loadFromUri('models');

    }

    function loadRegisteredFaces() {

      const data = JSON.parse(localStorage.getItem('students')) || [];

      return data

        .filter(s => s.descriptor)

        .map(s => new faceapi.LabeledFaceDescriptors(

          s.name,

          [new Float32Array(s.descriptor)]

        ));

    }

    async function startVideo() {

      const stream = await navigator.mediaDevices.getUserMedia({ video: {} });

      video.srcObject = stream;

    }

    function logAttendance(name) {

      const time = new Date().toLocaleTimeString();

      const entry = document.createElement('div');

      entry.className = 'entry';

      entry.textContent = `${name} checked in at ${time}`;

      log.appendChild(entry);

      const records = JSON.parse(localStorage.getItem('attendanceLog')) || [];

      records.push({ name, time });

      localStorage.setItem('attendanceLog', JSON.stringify(records));

    }

    async function runRecognition() {

      const faceMatcher = new faceapi.FaceMatcher(labeledDescriptors, 0.6);

      video.addEventListener('play', () => {

        const canvas = faceapi.createCanvasFromMedia(video);

        document.body.append(canvas);

        const displaySize = { width: video.width, height: video.height };

        faceapi.matchDimensions(canvas, displaySize);

        setInterval(async () => {

          const detections = await faceapi.detectAllFaces(video, new faceapi.TinyFaceDetectorOptions())

            .withFaceLandmarks().withFaceDescriptors();

          const resized = faceapi.resizeResults(detections, displaySize);

          canvas.getContext('2d').clearRect(0, 0, canvas.width, canvas.height);

          faceapi.draw.drawDetections(canvas, resized);

          resized.forEach(detection => {

            const match = faceMatcher.findBestMatch(detection.descriptor);

            if (match.label !== 'unknown') {

              logAttendance(match.label);

            }

          });

        }, 2000);

      });

    }

    (async () => {

      await loadModels();

      labeledDescriptors = loadRegisteredFaces();

      await startVideo();

      runRecognition();

    })();

    document.getElementById('signupForm').addEventListener('submit', function(e) {

  const dob = document.getElementById('dob').value;

  const contact = document.getElementById('parentContact').value;

  const dobPattern = /^\d{4}\/\d{2}\/\d{2}$/;

  const contactPattern = /^\+264\d{8}$/;

  if (!dobPattern.test(dob)) {

    e.preventDefault();

    alert("Date of Birth must be in YYYY/MM/DD format.");

    return;

  }

  if (!contactPattern.test(contact)) {

    e.preventDefault();

    alert("Parent contact number must start with +264 and be followed by 8 digits.");

    return;

  }

});

# **Technical documentation**

### Technology Stack

* **Frontend**: HTML5, CSS3, JavaScript (ES6+)
* **Face Recognition**: face-api.js library
* **Storage**: Browser localStorage
* **Camera**: WebRTC getUserMedia API
* **Models**: TinyFaceDetector, FaceLandmark68Net, FaceRecognitionNet

## System Components

### 1. Authentication System (Login-signup.html)

* Teacher registration and login
* Data validation and storage
* Session management via localStorage

### 2. Student Management (student-management.html)

* Student enrollment with photo capture
* Base64 image storage
* Class assignment

### 3. Teacher Dashboard (teacher.html)

* Attendance overview
* Manual attendance marking
* Export functionality

### 4. Live Attendance System (live-attendance.html)

* Real-time face recognition
* Camera feed processing
* Automatic attendance marking

## API Reference

### Face Recognition Functions

#### loadFaceAPIScript()

Loads the face-api.js library from multiple CDN sources.

javascript

async function loadFaceAPIScript()

*// Returns: Promise<void>*

*// Throws: Error if all CDN sources fail*

#### loadModels()

Loads the required ML models for face recognition.

javascript

async function loadModels()

*// Models loaded:*

*// - tinyFaceDetector*

*// - faceLandmark68Net*

*// - faceRecognitionNet*

#### processFaceDescriptors()

Processes student photos to create face descriptors.

javascript

async function processFaceDescriptors()

*// Creates face descriptors for enrolled students*

*// Stores descriptors in knownFaceDescriptors array*

### Data Management Functions

#### saveStudent(studentData)

Saves student information to localStorage.

javascript

function saveStudent(studentData)

*// Parameters:*

*// - studentData: {id, name, classId, photoBase64, faceDescriptor}*

#### getAttendanceData()

Retrieves attendance records from localStorage.

javascript

function getAttendanceData()

*// Returns: Object with student attendance records*

### Camera Functions

#### startCamera()

Initializes the webcam feed for live attendance.

javascript

async function startCamera()

*// Returns: MediaStream object*

*// Requires: Camera permissions*

## Database Schema (localStorage)

### Students Collection

javascript

{

"students": [

{

"id": "1756401116772",

"name": "John Doe",

"classId": "11B",

"photoBase64": "data:image/jpeg;base64,/9j/4AAQ...",

"faceDescriptor": [Float32Array],

"enrolledDate": "2024-01-15T10:30:00Z"

}

]

}

### Teachers Collection

javascript

{

"teachers": [

{

"id": "teacher\_001",

"title": "Mr",

"firstName": "John",

"surname": "Smith",

"classId": "11B",

"subject": "Mathematics",

"hashedPassword": "hash\_value"

}

]

}

### Attendance Records

javascript

{

"attendance": {

"1756401116772": {

"status": "Present",

"time": "2024-01-15T09:15:00Z",

"confidence": 0.89,

"method": "face\_recognition"

}

}

}

### Current Session

javascript

{

"currentTeacher": {

"id": "teacher\_001",

"firstName": "John",

"surname": "Smith",

"classId": "11B"

}

}

## Face Recognition Implementation

### Model Configuration

javascript

const MODEL\_URL = 'https://raw.githubusercontent.com/justadudewhohacks/face-api.js/master/weights';

*// Models used:*

- tiny\_face\_detector (lightweight, fast detection)

- face\_landmark\_68 (facial landmarks)

- face\_recognition (face descriptors)

### Detection Parameters

javascript

const detectionOptions = {

inputSize: 416,

scoreThreshold: 0.5,

minConfidence: 0.6

}

### Recognition Process

1. **Face Detection**: Locate faces in the camera feed
2. **Landmark Extraction**: Identify facial landmarks
3. **Descriptor Generation**: Create 128-dimensional face descriptor
4. **Comparison**: Compare with stored student descriptors
5. **Matching**: Find best match above confidence threshold

### Performance Optimization

* Uses TinyFaceDetector for speed
* Processes every 3rd frame to reduce load
* Implements confidence thresholds
* GPU acceleration when available

## Troubleshooting

### Common Issues

#### 1. Face-API Not Loading

Error: "faceapi is not defined"

**Solution**: Check CDN connectivity, try alternative CDN sources

#### 2. Model Loading Failures

Error: "Failed to fetch"

**Solution**:

* Verify internet connection
* Check CORS policy
* Use CDN fallback: GitHub raw content

#### 3. Camera Access Denied

Error: "Permission denied"

**Solution**:

* Enable camera permissions in browser
* Use HTTPS (required for camera access)
* Check browser compatibility

#### 4. Face Recognition Accuracy Issues

Low confidence scores or false matches

**Solution**:

* Ensure good lighting during enrollment
* Use high-quality photos (min 200x200px)
* Re-enroll with multiple angles
* Adjust confidence thresholds

#### 5. Performance Issues

High memory usage or slow processing

**Solution**:

* Reduce camera resolution
* Process fewer frames per second
* Clear browser cache
* Close other browser tabs

### Debug Mode

Enable debug logging:

javascript

const DEBUG\_MODE = true; *// Set to true for debugging*

## Known Issues

### Browser Compatibility

* **Chrome**: Full support ✅
* **Firefox**: Face recognition may be slower ⚠️
* **Safari**: Limited WebRTC support ⚠️
* **Edge**: Full support ✅

### File Protocol Limitations

* Cannot load local images via file:// protocol
* Base64 fallback implemented for offline usage
* Requires web server for full functionality

### Performance Limitations

* Face recognition is CPU-intensive
* GPU memory usage can be high (65MB+)
* Processing speed depends on device capabilities

### Storage Limitations

* localStorage has 5-10MB limit per domain
* Large number of students may exceed limits
* No data persistence across browsers/devices

## Security Considerations

### Data Storage

* Face descriptors stored locally (not on servers)
* No biometric data transmission
* Student photos stored as Base64 (encrypted)

### Privacy

* Camera access only during attendance sessions
* No video recording or storage
* Face data processed locally

### Access Control

* Teacher authentication required
* Class-based access restrictions
* Session-based permissions

# **Testing**

## Test strategy

## Test Results

# **User Guide**

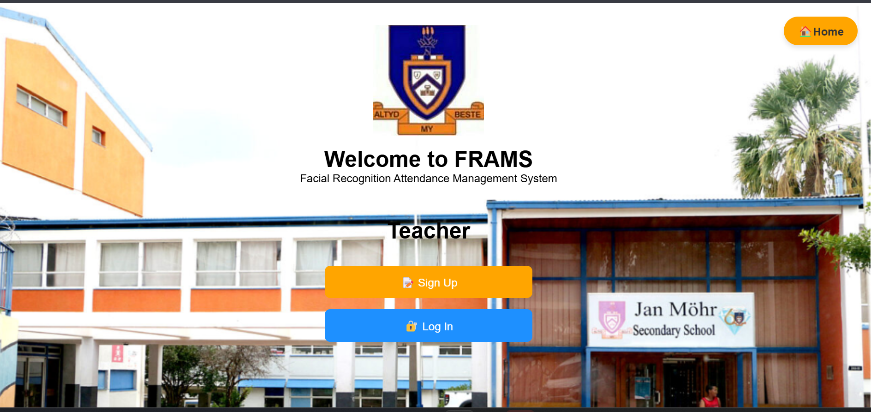
**1. Introduction**

The Facial Recognition Attendance Management System (FRAMS) is designed to streamline and automate the process of recording student attendance using facial recognition technology. This guide explains how to log in, navigate the interface, manage students, and record attendance.

**2. System Requirements**

* **Device**: Desktop or laptop with a webcam (mobile/tablet support optional if enabled)
* **Browser**: Latest version of Chrome, Edge, or Firefox
* **Internet**: Stable connection for real‑time recognition and data synchronisation
* **Permissions**: Allow camera access when prompted

1. **Logging In**

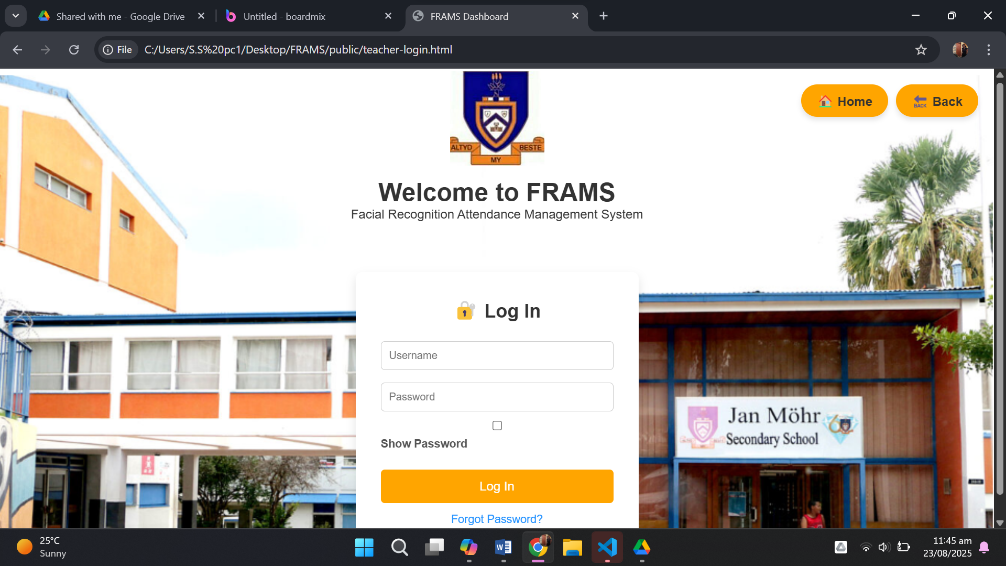


1. Open your browser and navigate to the FRAMS login page.
2. Enter your **username** and **password**.

* Enter your **Register Class** (e.g., 11A).

1. Click **Log In**.
2. If credentials are correct, you will be redirected to the **Teacher Dashboard**.

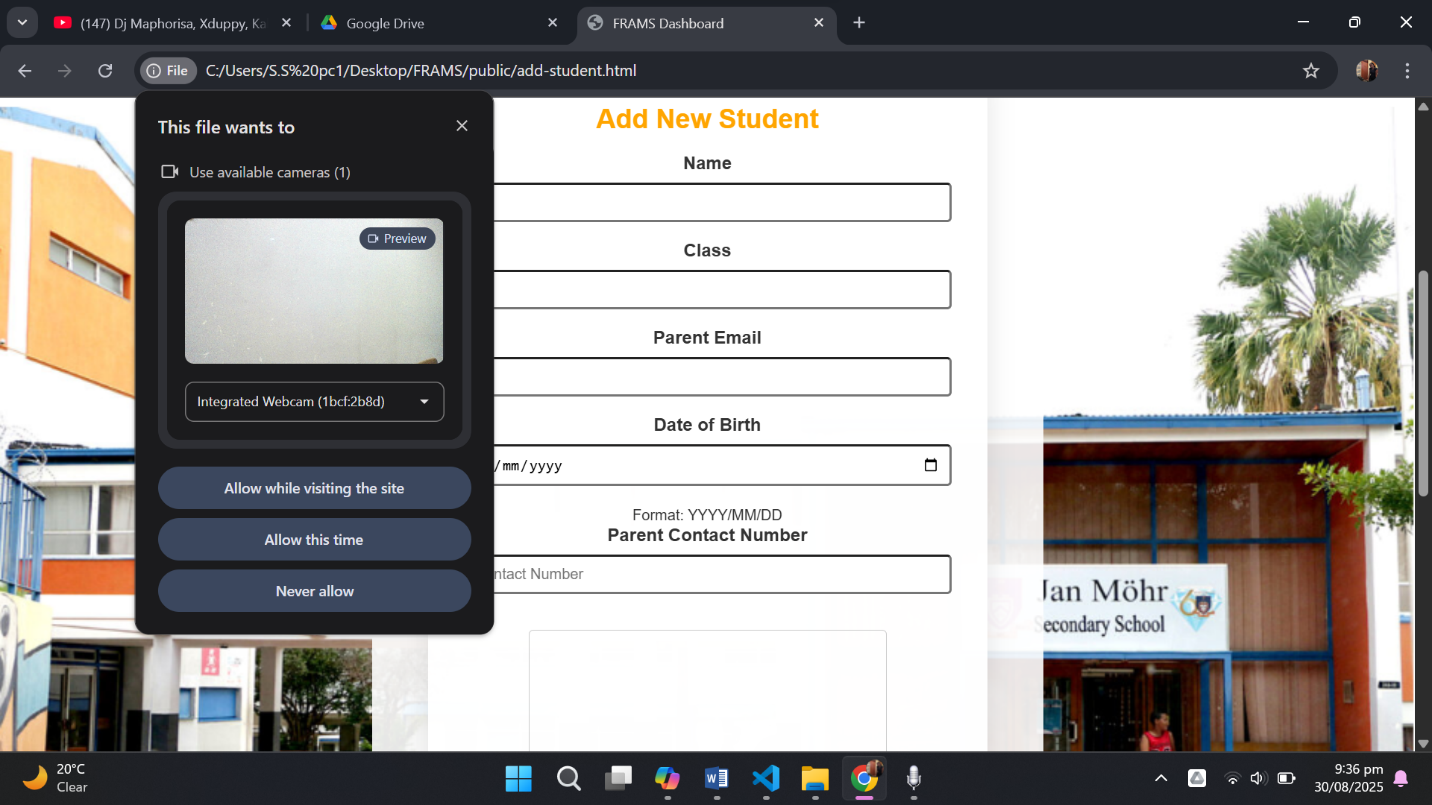
**Navigating the Teacher Dashboard**

****

The dashboard is divided into key sections:

* **Teacher Info Panel**: Displays your name, initials, class, and login time.
* **Filters**: Allows you to filter students by date, status, or search by name/ID.
* **Action Buttons**: Quick access to Refresh, Save, Live Attendance, and Logout.
* **Attendance Table**: Lists all students in your class with their ID, status, and last marked time.
* **Summary Cards**: Show counts for Present, Absent, Late, and Total students.

1. **Adding a Student**



1. Navigate to the **Add Student** page.
2. Fill in the required fields:
   * Full Name
   * Student ID
   * Class ID
   * Contact Number (format: digits, spaces, +, or -, 7–15 characters)

Upload a clear, front‑facing photo of the student (JPEG/PNG).

Click **Save** to store the student in the system.

**6. Marking Attendance Manually**

1. On the Teacher Dashboard, locate the student’s row.
2. Click **P** (Present), **A** (Absent), or **L** (Late).
3. The status will update instantly and the time will be recorded.
4. Click **Save** to store the updated attendance in local storage.

**7. Using Live Attendance**

1. Click the **Live Attendance** button.
2. Allow camera access when prompted.
3. The system will scan faces in real time.

When a match is found, the student is automatically marked **Present** and added to the Present list. Attendance is saved automatically with the date and time.

**8. Viewing Attendance Records**

1. Use the **Date Filter** to select the desired date.
2. The table will update to show attendance for that day.
3. Use the **Search** bar to find a specific student.

**9. Logging Out**

* Click the **Logout** button in the Teacher Info Panel.
* This will clear your session and return you to the login page.

**10. Tips for Best Results**

* Ensure the camera is positioned at eye level with good lighting.
* Use high‑quality, recent student photos for better recognition accuracy.
* Regularly update student images if appearances change significantly.
* Always click **Save** after making manual changes to attendance.

**11. Troubleshooting**

* **Camera not working**: Check browser permissions and ensure no other app is using the camera.
* **Face not recognised**: Verify the student’s photo is clear and correctly named in the assets/faces folder.
* **Login issues**: Ensure username, password, and class ID are correct; reset password if necessary.

**12. Support**

For technical assistance, contact the system administrator or IT support team with details of the issue, including screenshots if possible.

# **Evaluation and Development**

## Evaluation

The Facial Recognition Attendance Management System (FRAMS) has been successfully implemented to automate the process of recording student attendance through facial recognition technology. The system has demonstrated clear benefits in terms of efficiency, accuracy, and ease of use compared to traditional manual methods. Teachers can log in securely, manage their class lists, and mark attendance in real time, while the system stores and organises records for future reference.

Key strengths identified during evaluation include:

* **Automation of Attendance**: The use of facial recognition significantly reduces the time required to take attendance, freeing up valuable teaching time.
* **Improved Accuracy**: The system eliminates common human errors associated with manual entry, such as mis‑marking or overlooking students.
* **User‑Friendly Interface**: The dashboard and controls are intuitive, allowing teachers to navigate and perform tasks with minimal training.
* **Data Storage and Retrieval**: Attendance records are stored digitally, making it easy to retrieve historical data for reporting or analysis.
* **Security Measures**: Password‑protected logins and role‑based access help safeguard sensitive information.

However, the evaluation also revealed some limitations:

* **Dependence on Image Quality**: Recognition accuracy can be affected by poor lighting, camera quality, or changes in a student’s appearance.
* **Limited Device Support**: The system currently runs best on desktop browsers, with

# **Opportunities for system development**

The current Facial Recognition Attendance Management System (FRAMS) delivers on its core objective of automating attendance tracking through facial recognition. However, there is significant scope to expand its capabilities, improve its performance, and enhance the overall user experience. The following opportunities outline potential directions for future development:

1. **Improved Facial Recognition Accuracy**
   * Upgrade to more advanced AI models capable of handling varied lighting, camera angles, and partial occlusions.
   * Implement adaptive learning so the system refines recognition accuracy over time using new, consented images.
2. **Multi‑Camera and Multi‑Location Support**
   * Enable simultaneous monitoring from multiple classrooms or entry points.
   * Provide centralized dashboards for institutions with multiple campuses.
3. **Mobile Application Integration**
   * Develop mobile apps for teachers and administrators to view attendance in real time, approve exceptions, and receive alerts.
   * Allow students to securely view their own attendance records.
4. **Offline Mode and Data Synchronization**
   * Introduce offline capture for areas with poor connectivity.
   * Automatically sync attendance data to the central database when the connection is restored.
5. **Advanced Reporting and Analytics**
   * Offer detailed analytics on attendance trends, punctuality, and absence patterns.
   * Use predictive analytics to identify students at risk of chronic absenteeism.
6. **Integration with School Management Systems**
   * Build APIs to connect FRAMS with existing Student Information Systems (SIS) or Learning Management Systems (LMS).
   * Automate updates to student records and timetables.
7. **Enhanced Security and Privacy Controls**
   * Strengthen encryption for biometric data in storage and transit.
   * Introduce role‑based access control and detailed audit logs.
   * Provide transparent consent management to meet data protection regulations.
8. **User Experience Enhancements**
   * Refine the interface for faster navigation and more intuitive controls.
   * Add accessibility features such as high‑contrast mode, screen reader compatibility, and multi‑language support.
9. **Automated Notifications and Alerts**
   * Send SMS or email alerts to parents/guardians when a student is absent or late.
   * Notify administrators of unusual attendance patterns.
10. **Cloud Deployment and Scalability**
    * Migrate to a cloud‑based architecture for improved scalability and reliability.
    * Leverage cloud services for automated backups and disaster recovery.
11. **Multi‑Factor Attendance Verification**
    * Combine facial recognition with other methods such as RFID cards, QR codes, or fingerprint scanning for higher accuracy.
12. **AI‑Driven Insights**

* Correlate attendance data with academic performance

# Maintenance

## Support & Maintenance

### Regular Updates

* Update face-api.js library regularly
* Monitor CDN availability
* Test browser compatibility
* Review security measures

### Backup Procedures

javascript

*// Export data for backup*

function exportData() {

const data = {

students: localStorage.getItem('students'),

teachers: localStorage.getItem('teachers'),

attendance: localStorage.getItem('attendance')

};

return JSON.stringify(data);

}

### System Monitoring

* Monitor face recognition accuracy
* Track system performance
* Log error rates
* Monitor storage usage

**Conclusion**  
By pursuing these opportunities, FRAMS can evolve from a reliable attendance tracker into a comprehensive, intelligent, and secure student management platform. These enhancements would improve operational efficiency, strengthen compliance, and deliver richer insights for decision‑making, ultimately benefiting administrators, educators, students, and parents alike.