DDS DESIGN UPDATE: FRAS

Project Summary

Our Facial Recognition Attendance System (FRAS) is a web-based application that aims to automate the existing, inefficient attendance taking process. It will offer its users each with their own intuitive dashboards based on the assigned user role.

Overall System Description

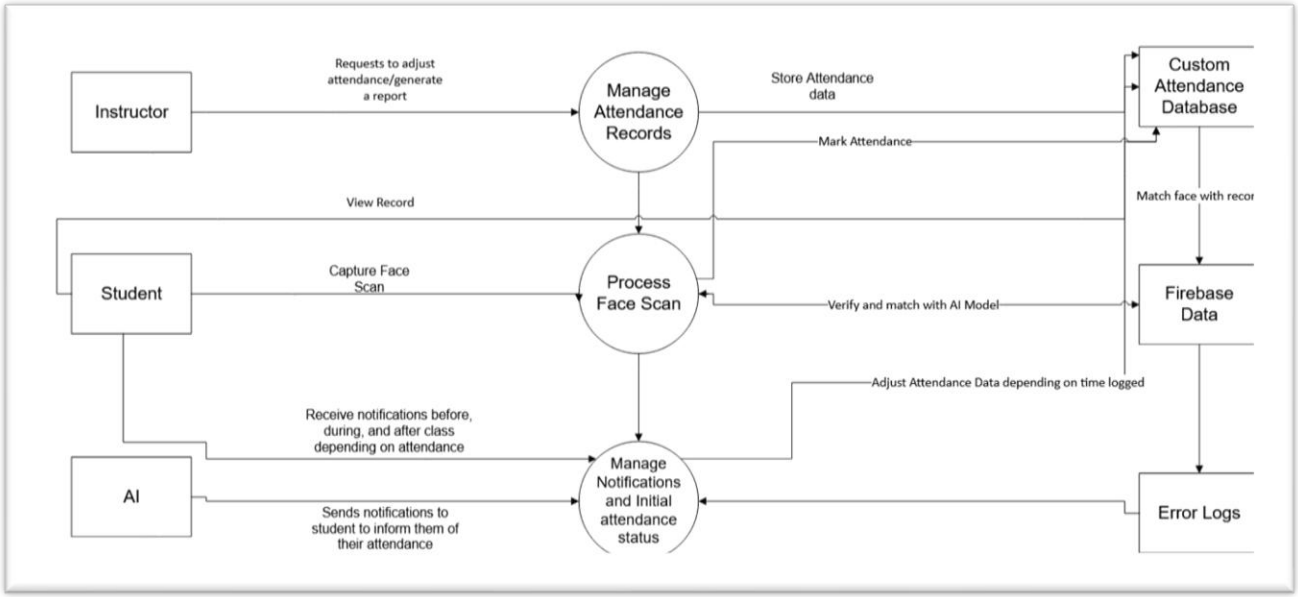
Our system is built using a React frontend, a Flask backend, and Firebase Firestore for data storage. Students can sign into our system to record their attendance for their classes and view their attendance history for each of their enrolled classes. Teachers will also be able to sign into our system which will direct them to their own dashboard. In their dashboard, they'll be able to view and manage individual and class attendance and export attendance analytics. Our backend handles all of the facial processing and recognition, IP geofencing checks, and our AI notification, while Firestore manages our role-based access, user data, and attendance logs.

Group 3:
Brendon Stepanek
Joshua Odegai
Zain Jamal
Neel Patel
Maximiliano Hernandez

Sponsor(s):
Professor Diana Rabah

Data Flow Diagrams

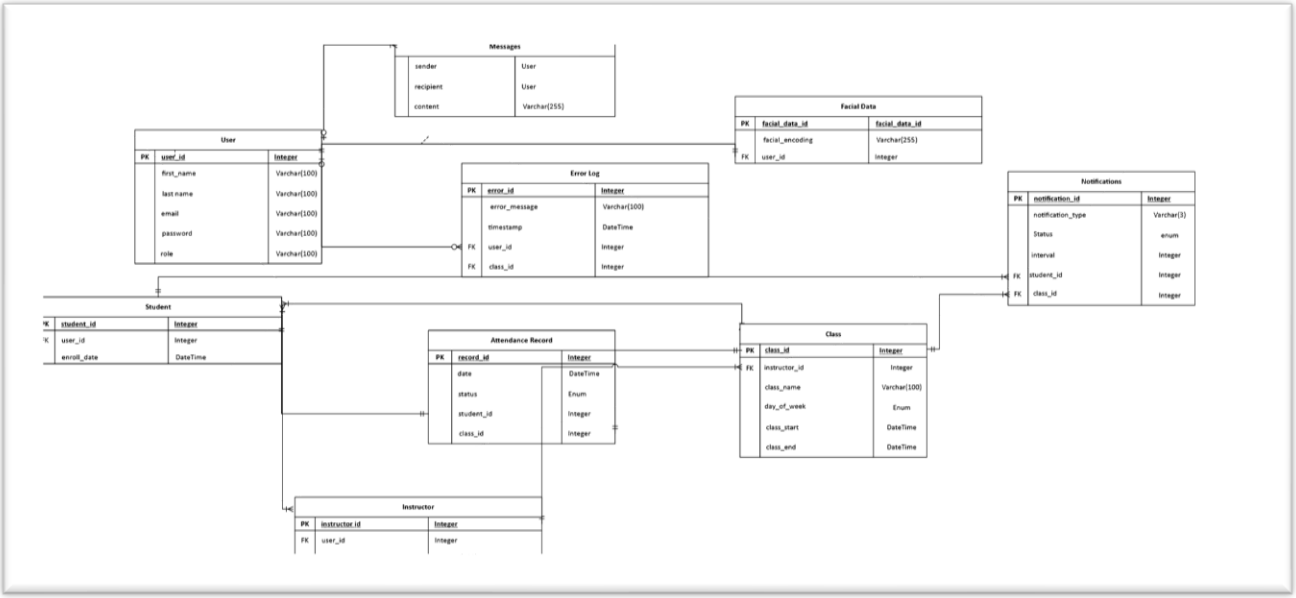
- User records information about everyone in the system (students or instructors).
- User is associated with Student and contains enrolment information.
- User is associated with Instructor and includes information about the department.
- The class is developed by instructors and entails schedule and timing.
- Attendance Record traces the absence, late and present state of the student.
- Notifications remind and update on attendance and class.
- Communication occurs between students and instructors, using messages.
- Facial Data stores the recognition information that is used to verify users.
- Error Log documents the error generated by users or classes.



Entity Relationship Diagrams

- Students scan their face to take attendance. The system takes the scan and compares it to existing records and records the outcome.
- Teachers can access attendance reports, create reports or demand corrections.
- The AI module is important as it confirms attendance data and sends notifications by default to the students. Students are alerted during and before and after the Class to ensure that they are informed of their attendance status.
- The attendance data is stored in a custom attendance database and face data are matched with records and maintained with firebase.
- Any problem that occurs, like misaligned data, or processing errors is recorded in the error logs to be checked and fixed.

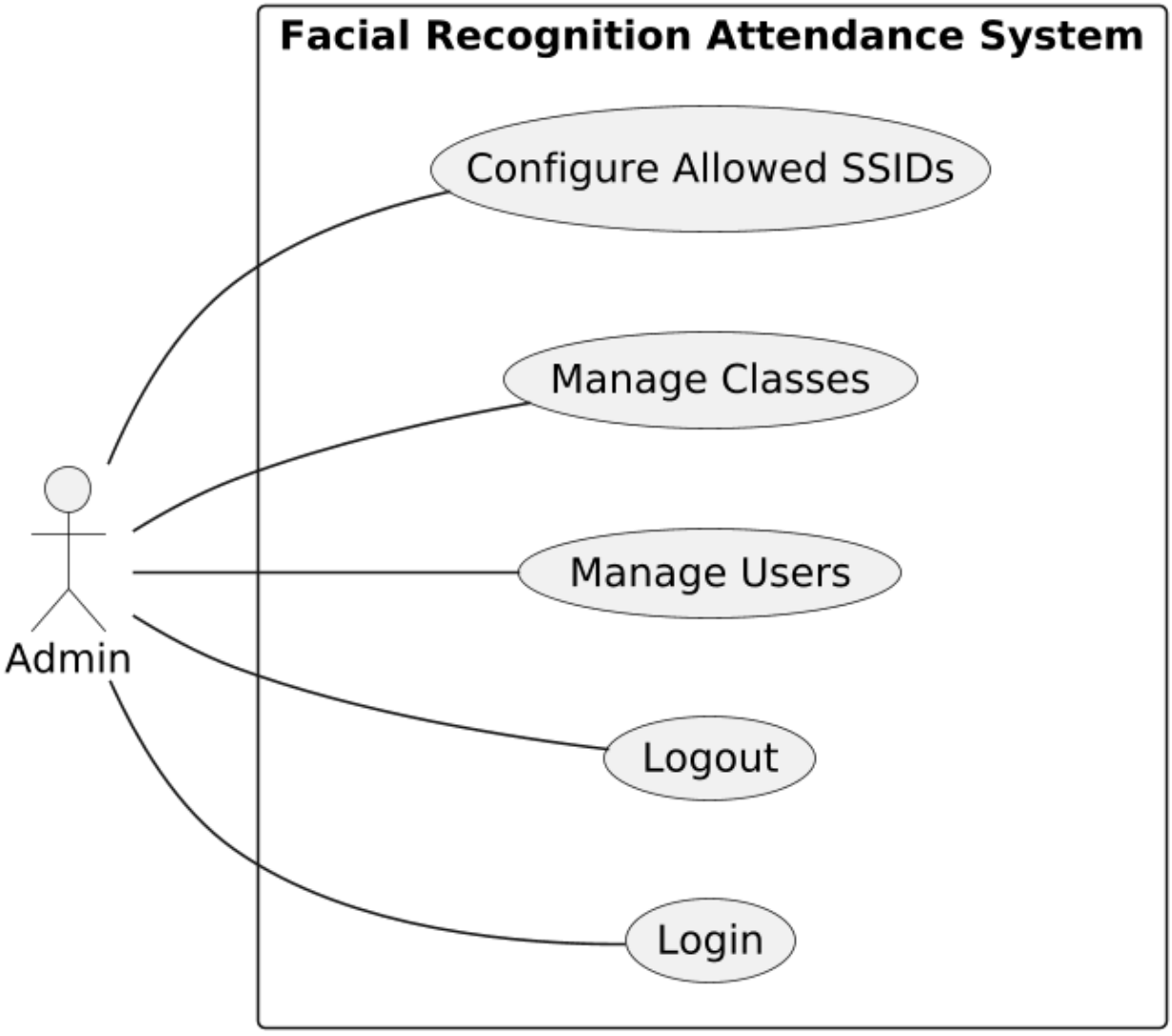
In short: the system captures attendance through facial recognition, verifies it with AI, stores the data securely, and keeps both students and instructors informed with real-time notifications.



Use Case Diagrams

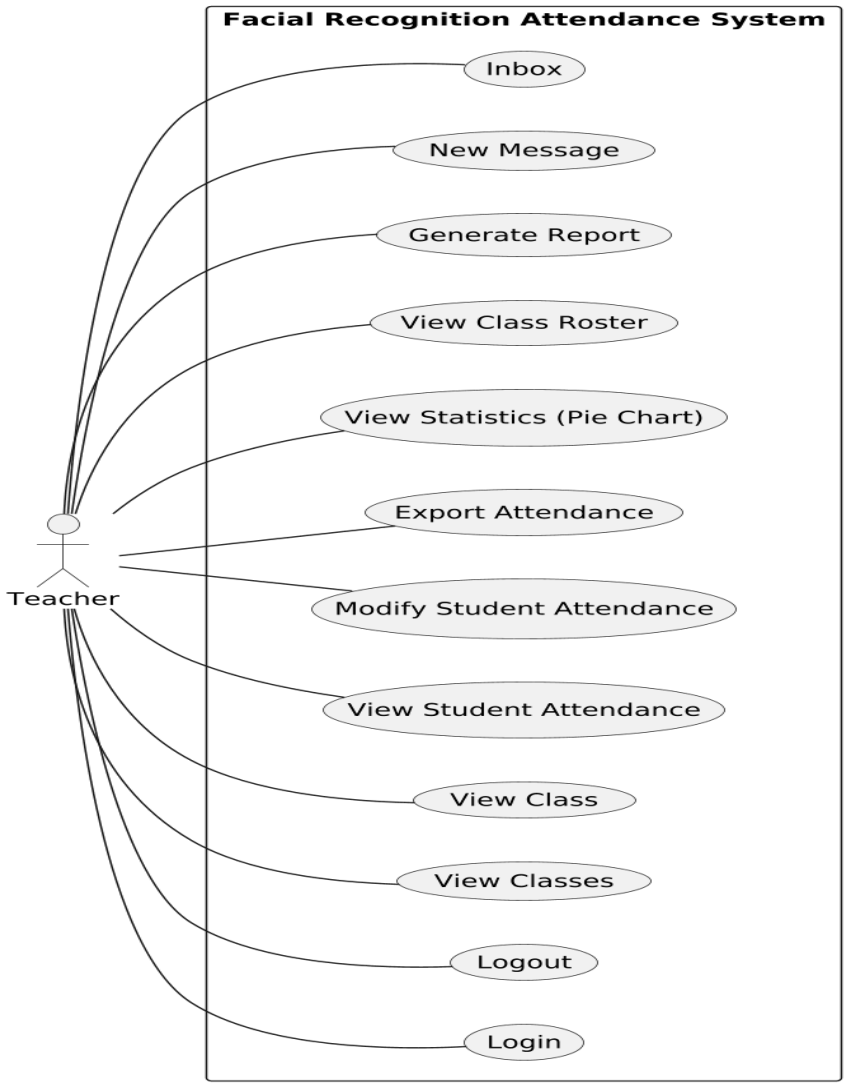
Admin

Admins have access to manage users and classes as well as configuring allowed SSIDs to ensure attendance is only recorded from an approved network.



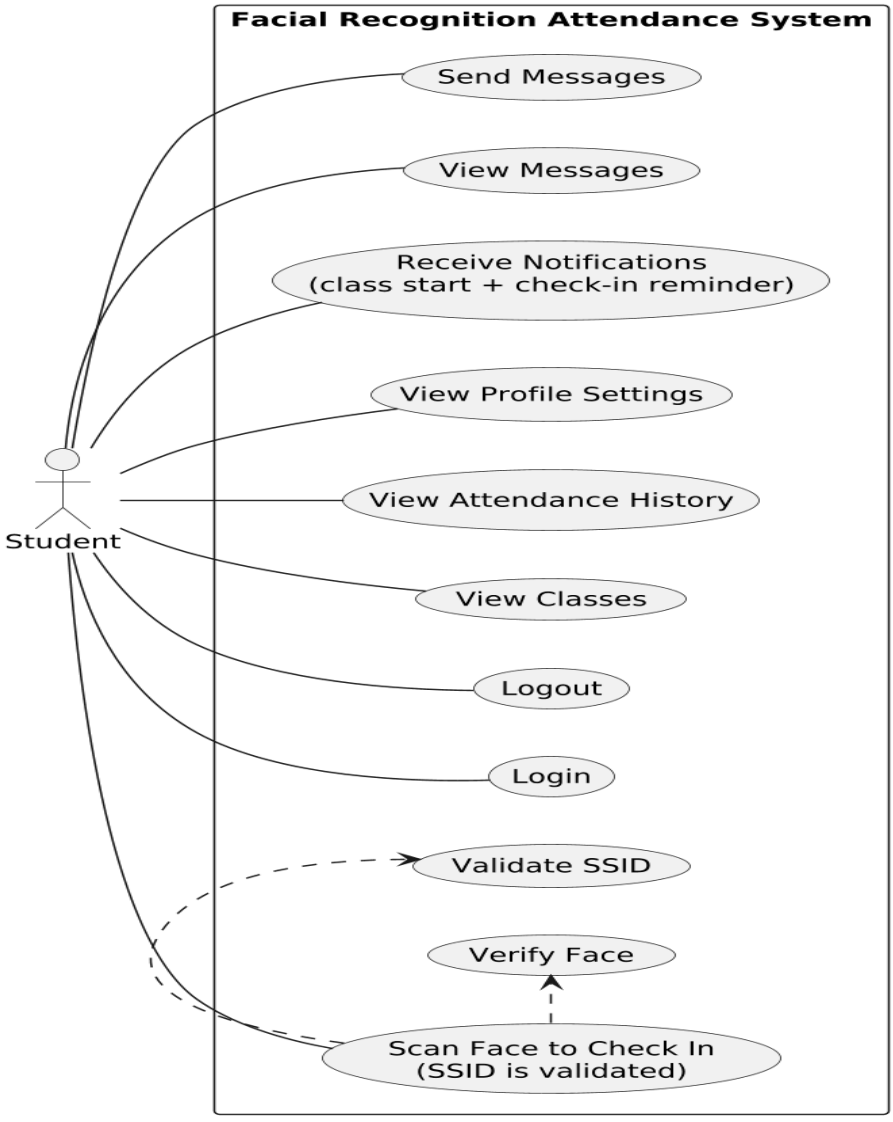
Teacher

Teachers can log in and out, view their classes, manage student attendance, and generate attendance reports. They'll be able to modify attendance records, export data, view statistics, access class rosters, and use the built-in messaging system.



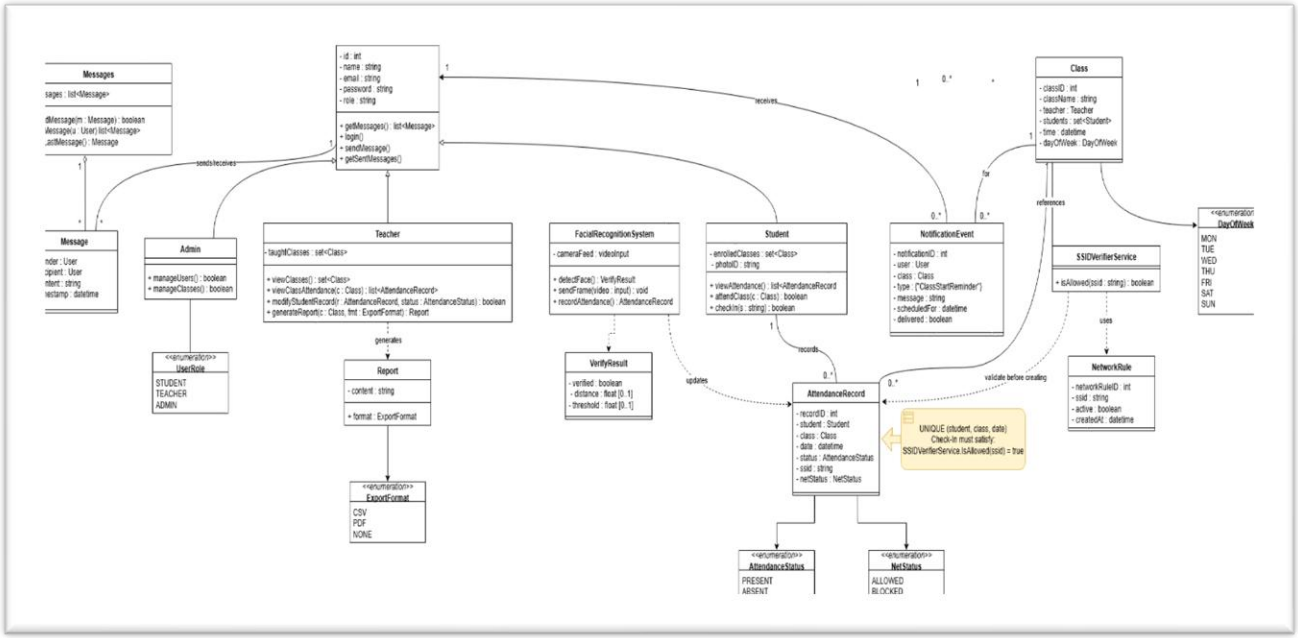
Student

The student will be able to log in and out, view their classes, and view their attendance history. They can record their attendance after passing a SSID validation check, receive class reminders, and use the built-in messaging system.



Class Diagrams

The class diagram shows our system's users, classes, and attendance process. It shows how students, admins, and teachers will interact with classes, attendance records, messaging, reports, and the facial recognition system. The updates to this diagram include our recent addition of the AI notification for reminders as well as SSID validation and added fields in the AttendanceRecord class for network verification/tracking.



Class Diagram Pseudocode

Attendance Recording

```
FUNCTION scanAndCheckIn(userId, classId, faceImage, ssid)
  IF NOT isAllowedSSID(ssid) THEN
    RETURN "Invalid SSID"
  END IF
  SET match = detectFace(faceImage)
  DECLARE status
  IF match IS TRUE THEN
    SET status = "PRESENT"
  ELSE
    SET status = "ABSENT"
  END IF
  CALL saveAttendance(userId, classId, status, ssid, currentTime())
END FUNCTION
```

Class-Start & Reminder Notifications

```
FUNCTION sendClassStartReminders(currentTime)
  FOR EACH c IN classes
    IF c.startTime EQUALS currentTime MINUS 5 minutes THEN
      FOR EACH s IN c.enrolledStudents
        CALL createNotification(s, c, "ClassStartReminder")
      END FOR
    END IF
    IF c.startTime EQUALS currentTime THEN
      FOR EACH s IN c.enrolledStudents
        IF NOT hasAttendance(s, c, today()) THEN
          CALL createNotification(s, c, "RecordAttendanceReminder")
        END IF
      END FOR
    END IF
  END FOR
END FUNCTION
```

Modify Attendance Record

```
FUNCTION modifyStudentAttendance(teacherId, recordId, newStatus)
  SET record = loadRecord(recordId)

  IF record IS NULL THEN
    RETURN "Record not found"
  END IF
  IF record.class.teacherId NOT EQUALS teacherId THEN
    RETURN "Unauthorized"
  END IF
  SET record.status = newStatus
  SET record.updatedAt = currentTime()
  CALL saveRecord(record)
END FUNCTION
```

Export Attendance

```
FUNCTION exportAttendance(classId, startDate, endDate, format)
  SET records = getRecords(classId, startDate, endDate)
  IF records.length EQUALS 0 THEN
    RETURN "No records found"
  END IF
  SET table = formatRecords(records)
  SET file = generateFile(table, format)  // CSV or PDF
  RETURN file
END FUNCTION
```

View Attendance History

```
FUNCTION fetchAttendanceHistory()
  SET res TO AWAIT api.get("/attendance/<userID>")
  RETURN res.data
END FUNCTION
```

View Attendance Statistics

```
FUNCTION loadClassStats(classID)
  SET res TO AWAIT api.get("/classes/" + classID + "/stats")
  RETURN res.data
END FUNCTION
```

View Class Roster

```
FUNCTION getRoster(classID) ASYNC
    SET res = AWAIT api.get("/classes/" + classID + "/roster")
    RETURN res.data
END FUNCTION
```

Manage Users

```
def create_user(admin_id, email, role, name):
    if not is_admin(admin_id):
        return "Unauthorized"
    user = User(email=email, role=role, name=name, created_at=now())
    save_user(user)
    return user.id
```

Manage Classes

```
FUNCTION create_class(userID, className, schedule)
  IF NOT is_admin(userID) THEN
    RETURN "Unauthorized"
  END IF
  SET cls = CREATE Class WITH className=className, userID=userID,
schedule=schedule
  CALL save_class(cls)
  RETURN cls.id
END FUNCTION
```

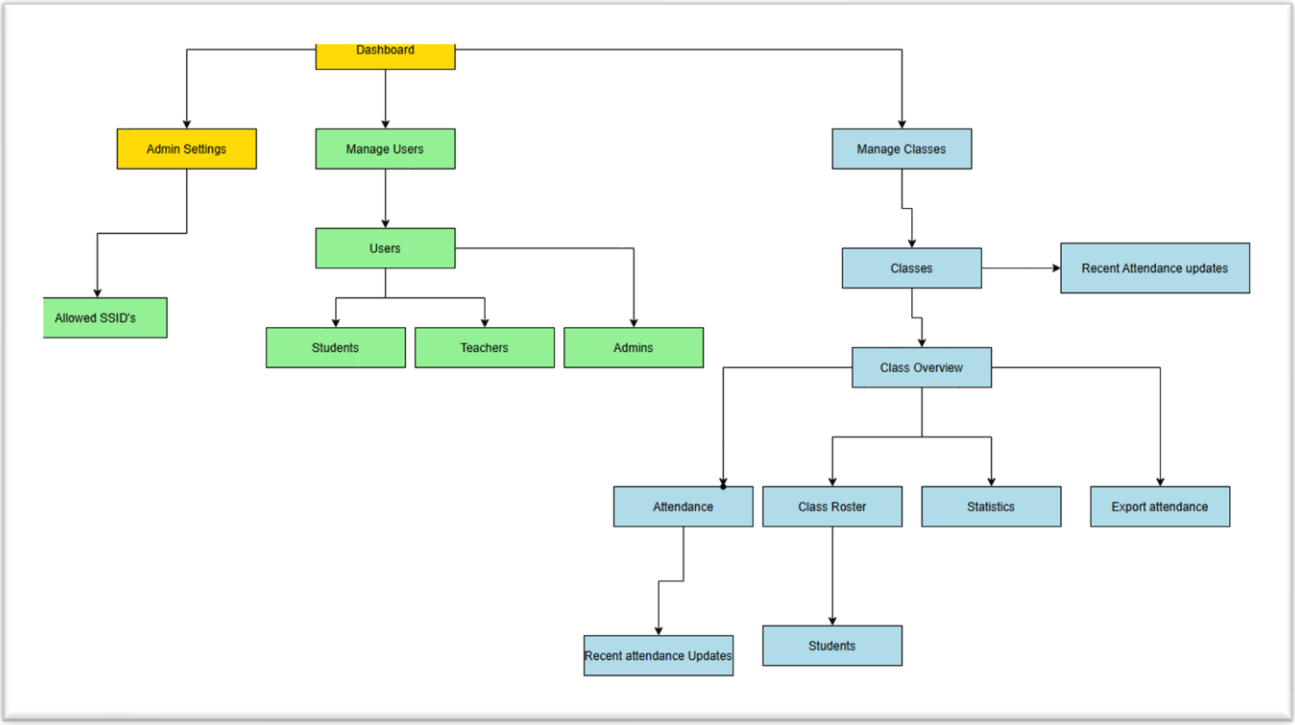
Information Architecture Diagram

Admin

Admin Flow Diagram The admin system begins with the main dashboard. From here, administrators can adjust system settings, manage users, and manage classes.

- Admin Settings include configuring allowed SSIDs to control system access.
- Manage Users organizes the system into Students, Teachers, and Admins, ensuring role-based access and proper management.
- Manage Classes allow admins to create and oversee classes. Each class contains an overview that branches into attendance tracking, class roster, statistics, and export options. Attendance also ties into recent updates to ensure accuracy.

In short: The admin panel centralizes control over settings, user roles, and class management, while linking attendance and performance data for complete oversight.

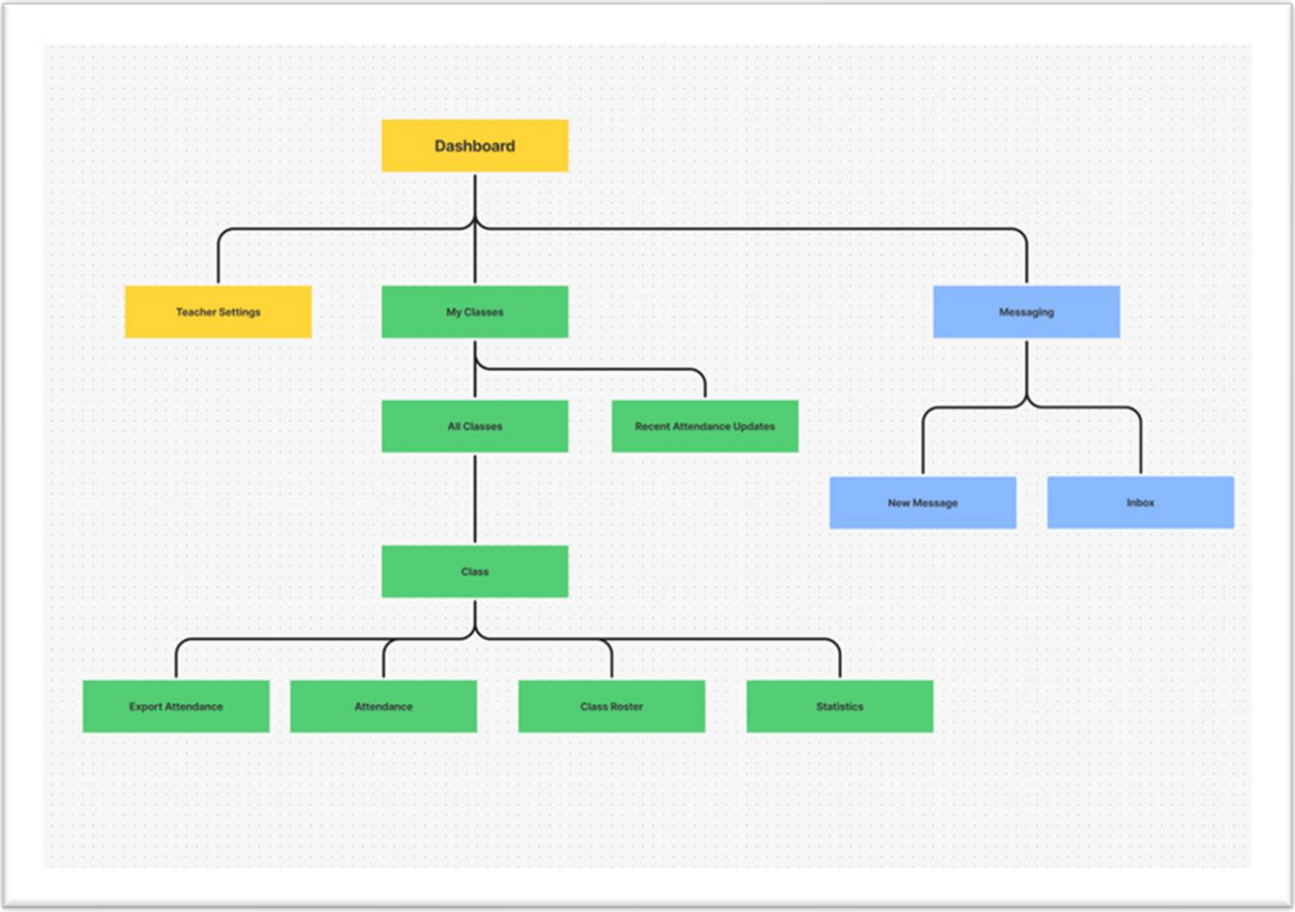


Teacher

The teacher system begins with the main dashboard. From here, teachers can adjust their personal settings, view their classes, and access messaging.

- **My Classes** provides access to all assigned classes and recent attendance updates.
- Within each **Class**, teachers can view and manage attendance, review the class roster, check statistics, and export attendance records.
- **Messaging** allows teachers to send new messages or check their inbox for communication.

In short: The teacher panel focuses on class management, attendance management, and communication tools, giving teachers control over student records and reporting.



Student

The student dashboard gives access to personal settings, classes, and communication tools.

- Student Settings provide customization of account preferences.
- My Classes leads to individual class pages, which include an overview and attendance records so students can track their progress.
- Messaging supports communication through new messages and an inbox, with notifications alerting students to updates.

In short: The student interface provides quick access to personal settings, class details, and messaging, ensuring students remain informed about attendance and communication.