SOFTWARE ENGINEERING PROJECT (18IS55)

Design Document for

“Attendance Management System using Facial Recognition”

Team Members:

1. Omkar Kabbur (1RV20IS028)
2. Pratham Agarwal(1RV20IS034)

1. Overview

Facial recognition is software that thoroughly maps a person's facial attributes and stores them in the database to be used as a reference whenever they need the recognition. To validate an individual's identification, the software compares a live capture or digital image to a piece of saved facial information using deep learning techniques.The method of recognizing or verifying a person's facial features is known as facial recognition

.This project aims to automate the traditional attendance system where the attendance is marked manually. It also enables an organization to maintain its records like in-time, out time, break time and attendance digitally. Digitalization of the system would also help in better visualization of the data using graphs to display the no. of students present each day

Classes: (Basic building blocks of Attendance Management System)

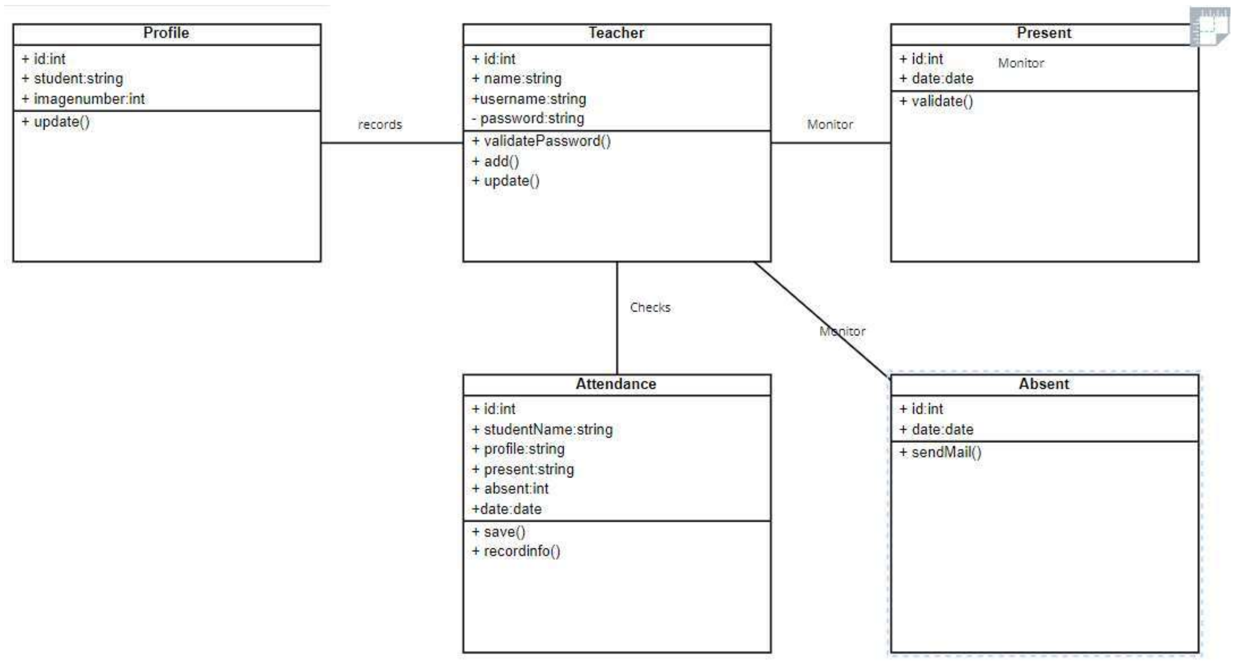
1	Student	Presents all the students of the department/college.
2	Teacher	Presents the teachers who teach a subject for a particular semester.
3	Attendance	stores all the records of student,date,profile etc
4	Profile	Contains all the students image,information
5	Present	marks the date,and ids of students who are present
6	Absent	has mail method,stores date and student details

2. System Structure

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. Class diagrams are a useful tool for modeling the static aspects of a system, including the relationships between objects and the attributes of those objects. They are typically used in the design phase of software development to help visualize and communicate the design of a system. In a class diagram, classes are represented as boxes with three compartments. The top compartment contains the name of the class, the middle compartment contains the attributes of the class, and the bottom compartment contains the operations that can be performed on the class. The relationships between classes are shown using lines connecting the classes, with different symbols used to indicate the type of relationship.

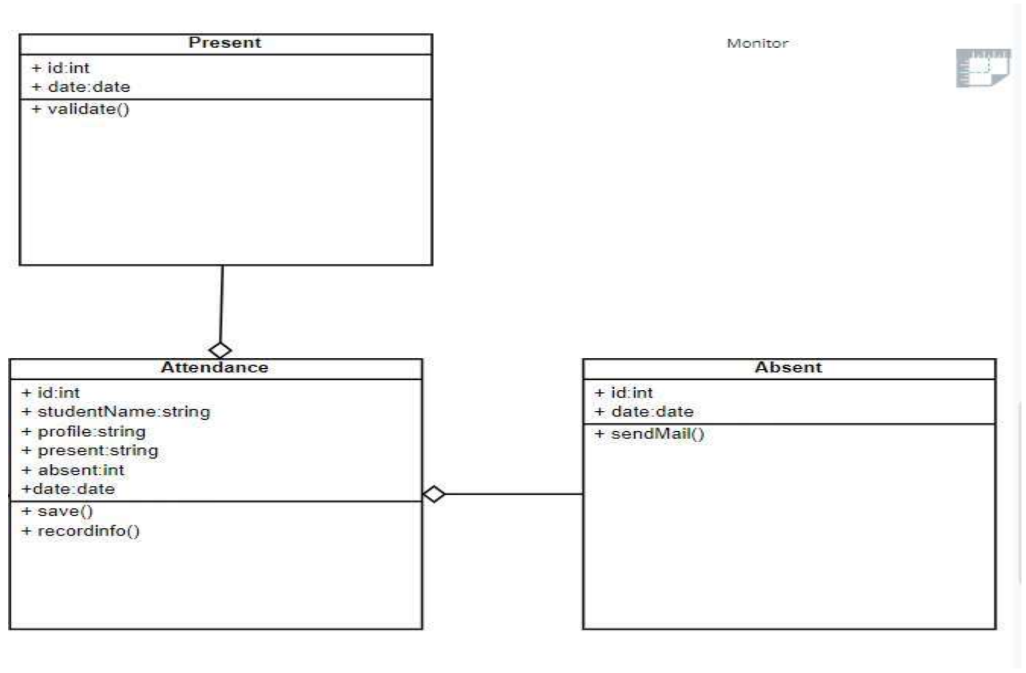
Association structures:

There is an association structure between the Profile and Teacher classes,between Teacher and Absent class and between Teacher and Present classes. This is because a user id is associated with every Teacher, Present ,Absent and Profile instance.



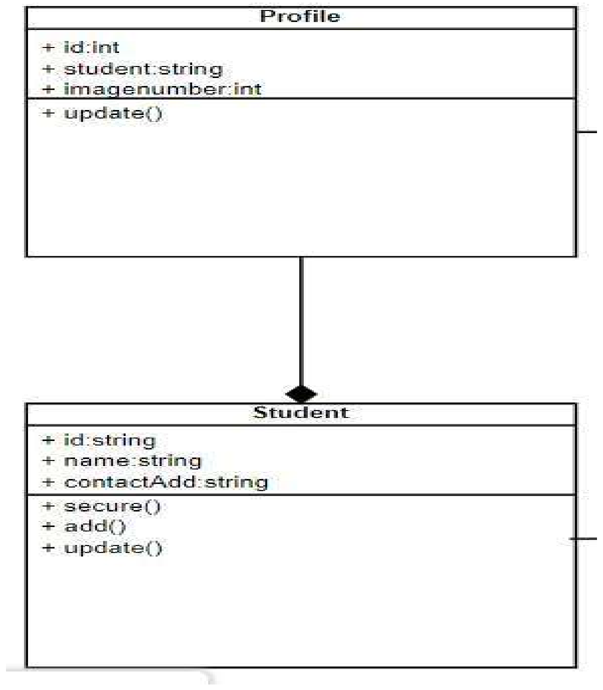
Aggregation structures:

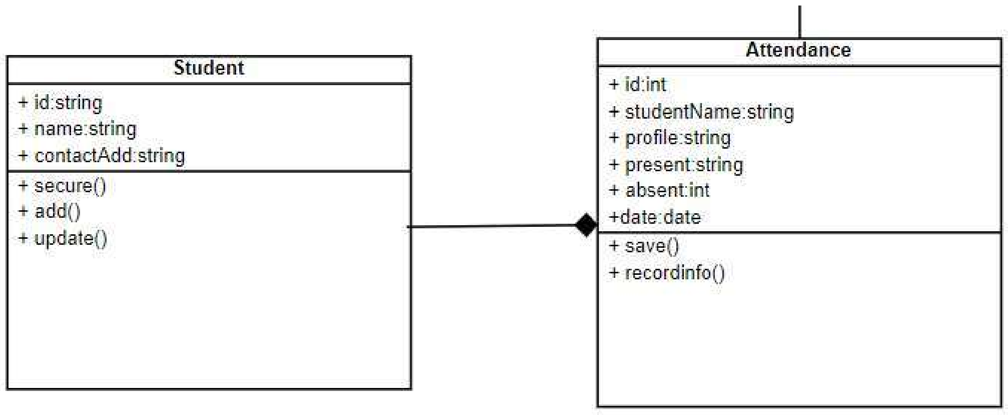
An aggregation structure exists between Attendance and Present and Attendance and Absent classes.
Attendance and Present have an aggregation structure as a Present instance is always associated with an Attendance instance.
Attendance and Absent have an aggregation structure as Absent instance is generated is always associated with a Attendance instance



Composition structures:

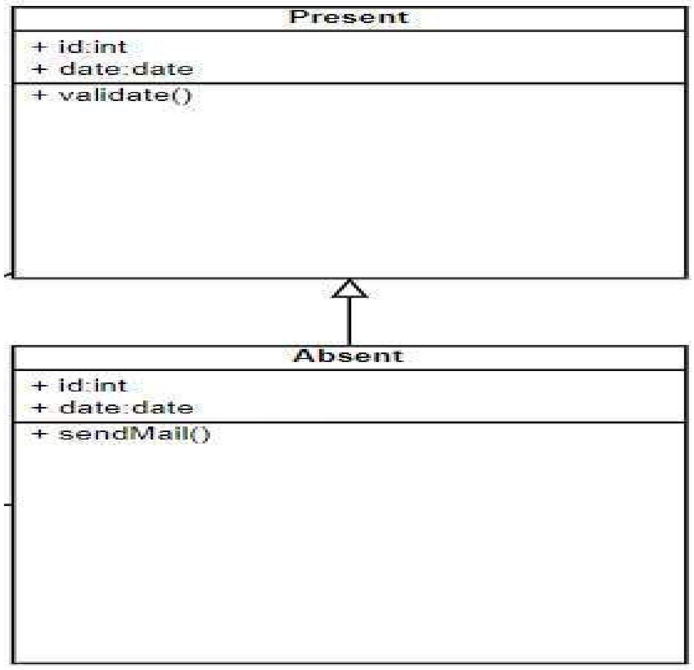
A composition structure exists between Attendance and Student and Profile and Student classes.
Attendance and Student have composition as Attendance class needs students.
Profile and Student have composition as a student class is has profile classes as its component and profile won't exist if students don't.



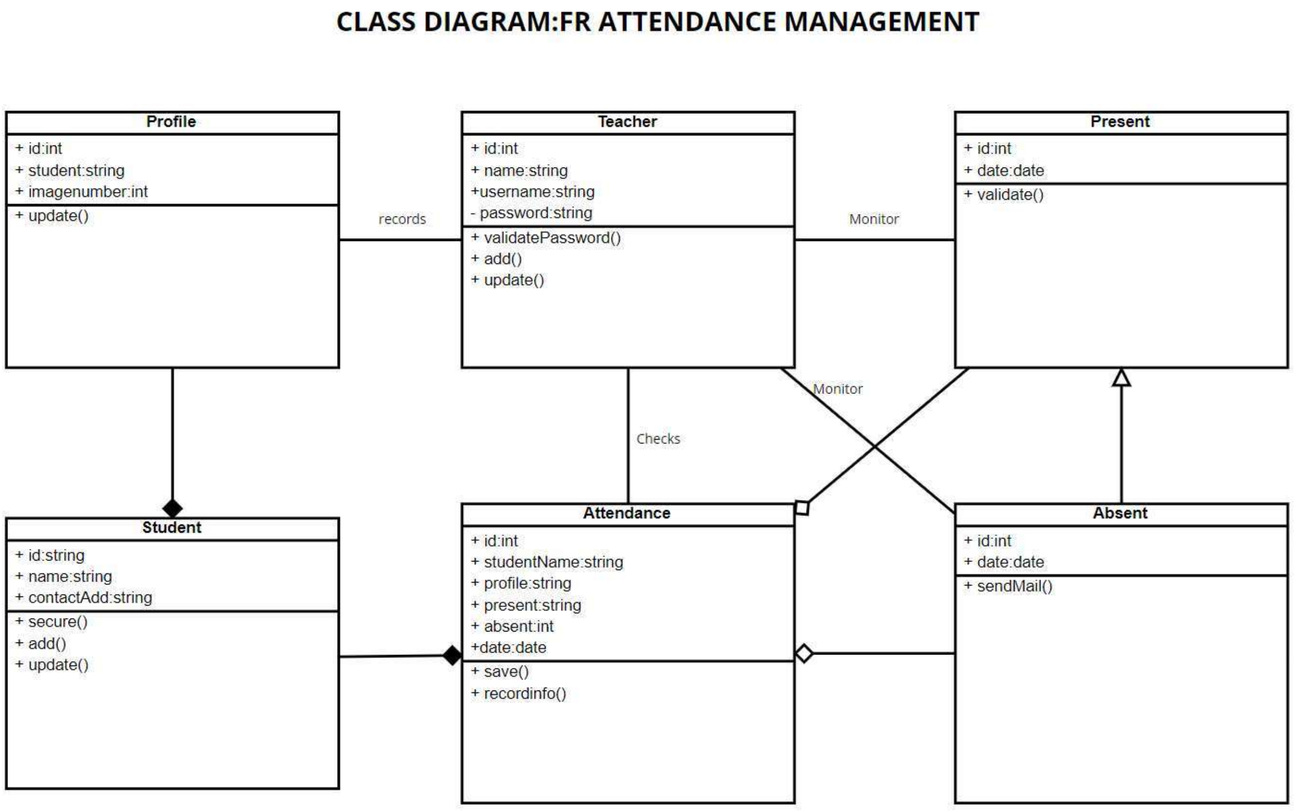


Inheritance structures:

Absent and Present have an inheritance structure as absent class instance are derived based on preset class.



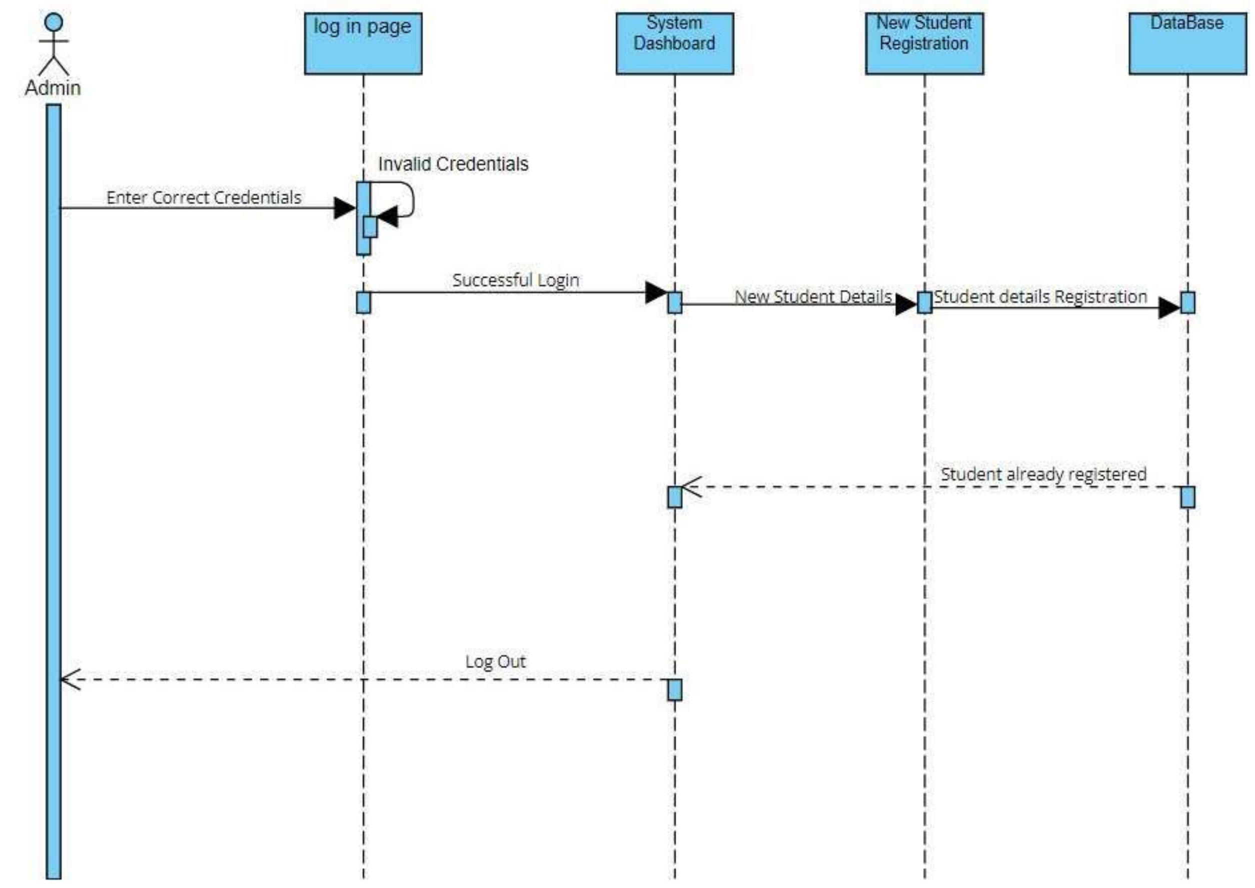
Complete class diagram:



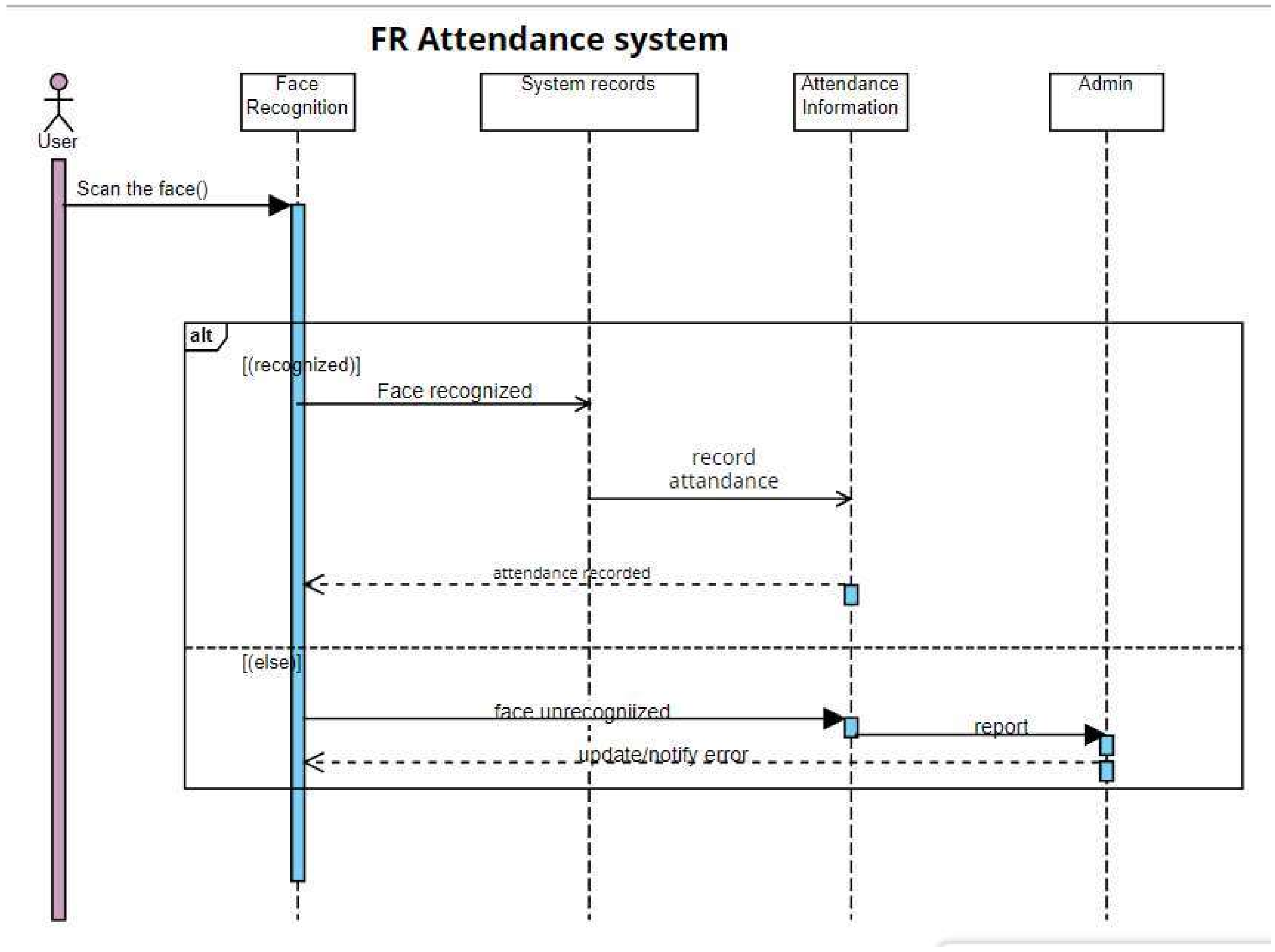
3. System Modeling

A sequence diagram is a type of diagram that shows how objects communicate with each other in a particular order. It is used to represent the interactions between objects in the context of a particular scenario. In a sequence diagram, the objects are represented as vertical lines on the left side of the diagram, and the interactions between them are represented as horizontal arrows that connect the objects. Each arrow represents a message being sent from one object to another, and the order of the arrows represents the sequence in which the messages are sent. Sequence diagrams are often used in software engineering to design and document the interactions between objects in a system. They can also be used to model the interactions between objects in other contexts, such as business processes or social interactions.

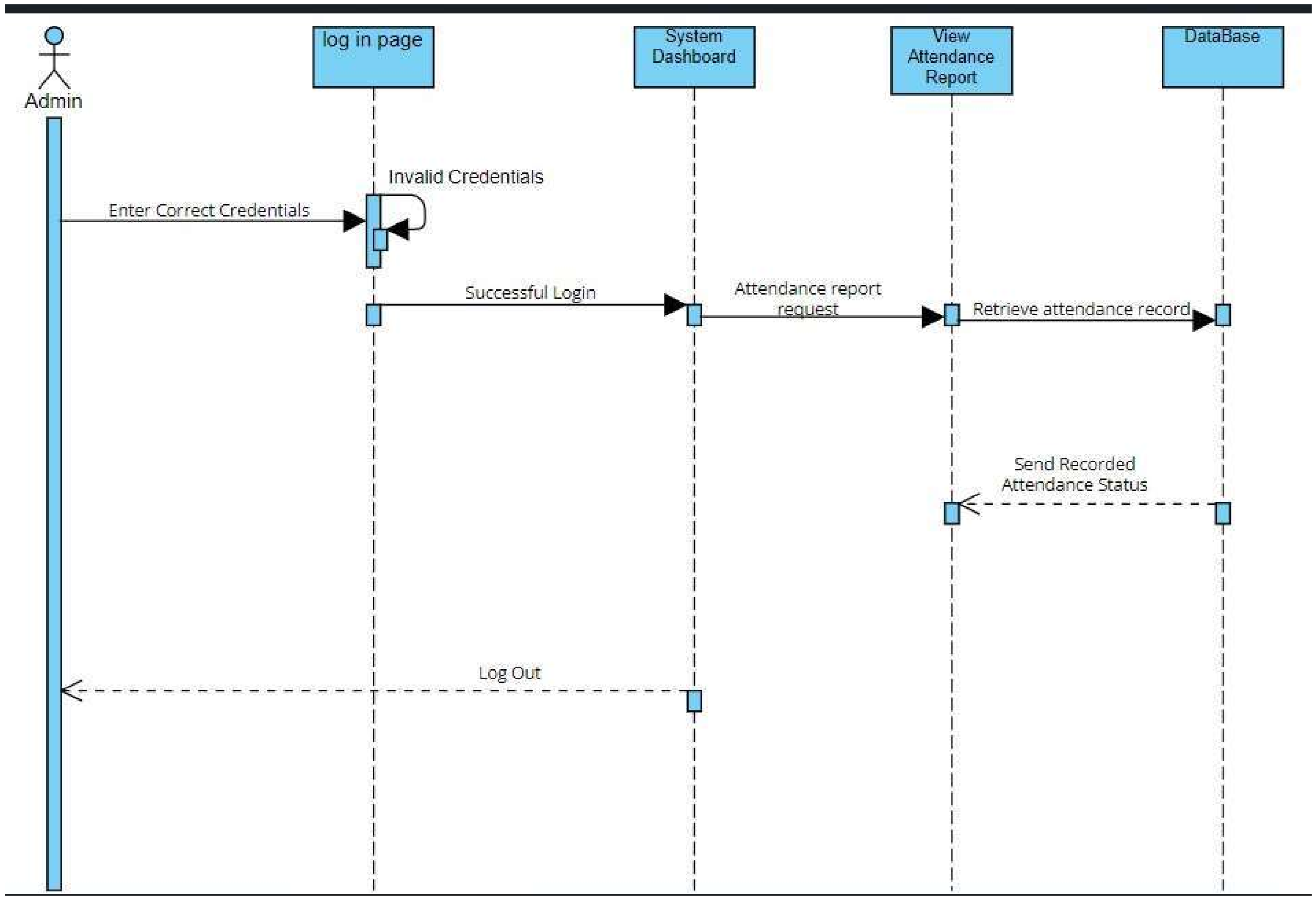
Principal Action: Registration of students



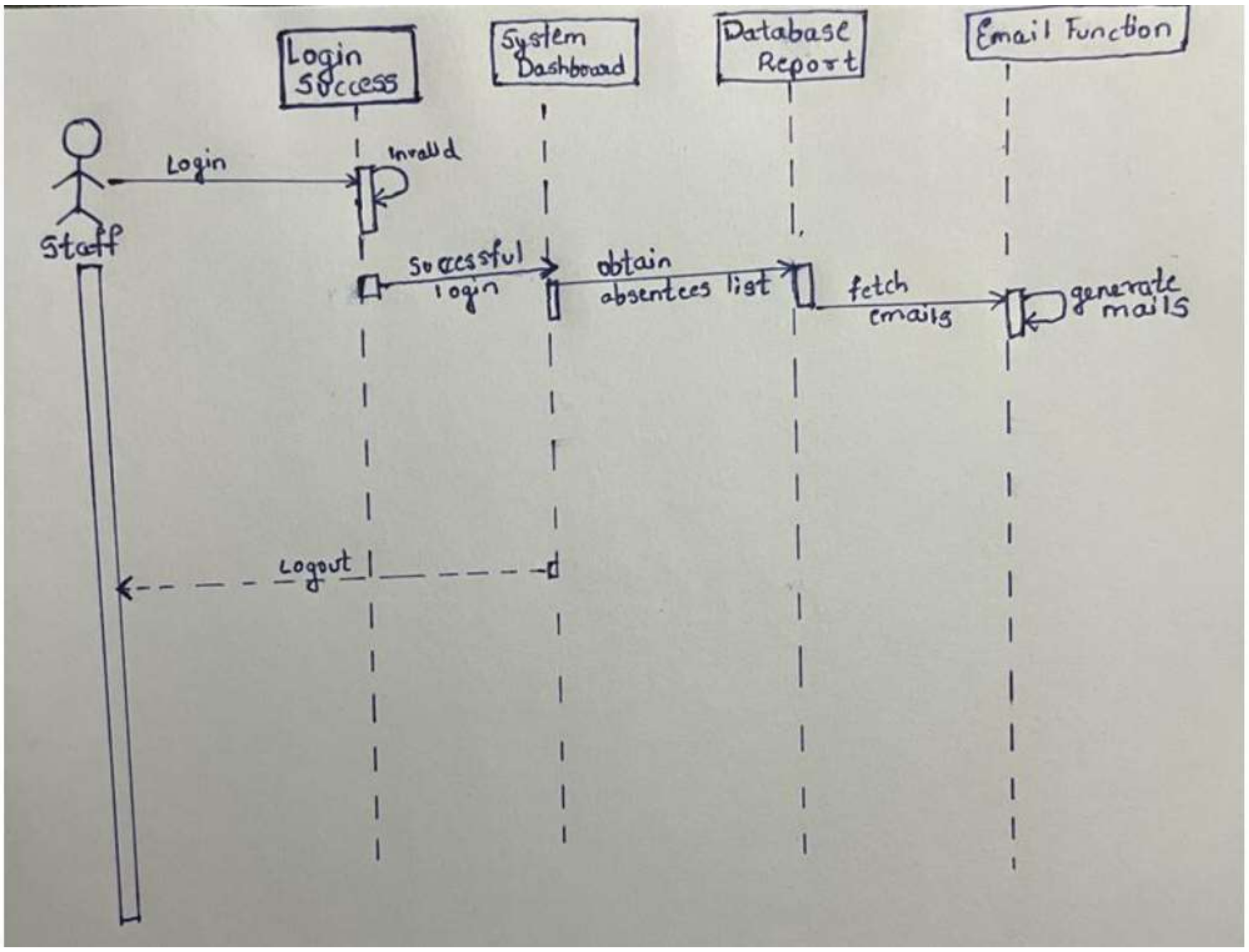
Principal Action: Scan Face



Principal Action: View Attendance


Admin
log in page
System Dashboard
View Attendance Report
DataBase
Invalid Credentials
Enter Correct Credentials
Successful Login
Attendance report request
Retrieve attendance record
Send Recorded Attendance Status
Log Out

Principal Action: Send Email



Detail Design Specification:

It consists of a list of main classes and their attributes and methods.

```
public class Profile
{
     public int id;
     public int imagenumber;
     public string student;


     public Profile()
     public void update()

public class Student
{
     public string id;
     public string name;
  public string contactAdd;

     public Student()
     public void update()
     public void add()
     public void secure()
}

public class Teacher
{
     public int id;
     public string name;
     public string username;
     public string password;

     public Teacher()
     public bool
     validatePassword()
     public void add()
     public void update()
}

public class Attendance
{
     public int id;
     public string
     studentName;
     public string profile;
     public string present;
     public int absent;
     public date date;

     public void Attendance()
     public void save()
     public void recordInfo()

}
```

```
public class Present
{
    public int id;
    public date date;



   public Present()
    public void validate()

}

public class Absent extends Present
{
    public int id;
    public date date;



    public Absent()

    public void sendMail()

}
```

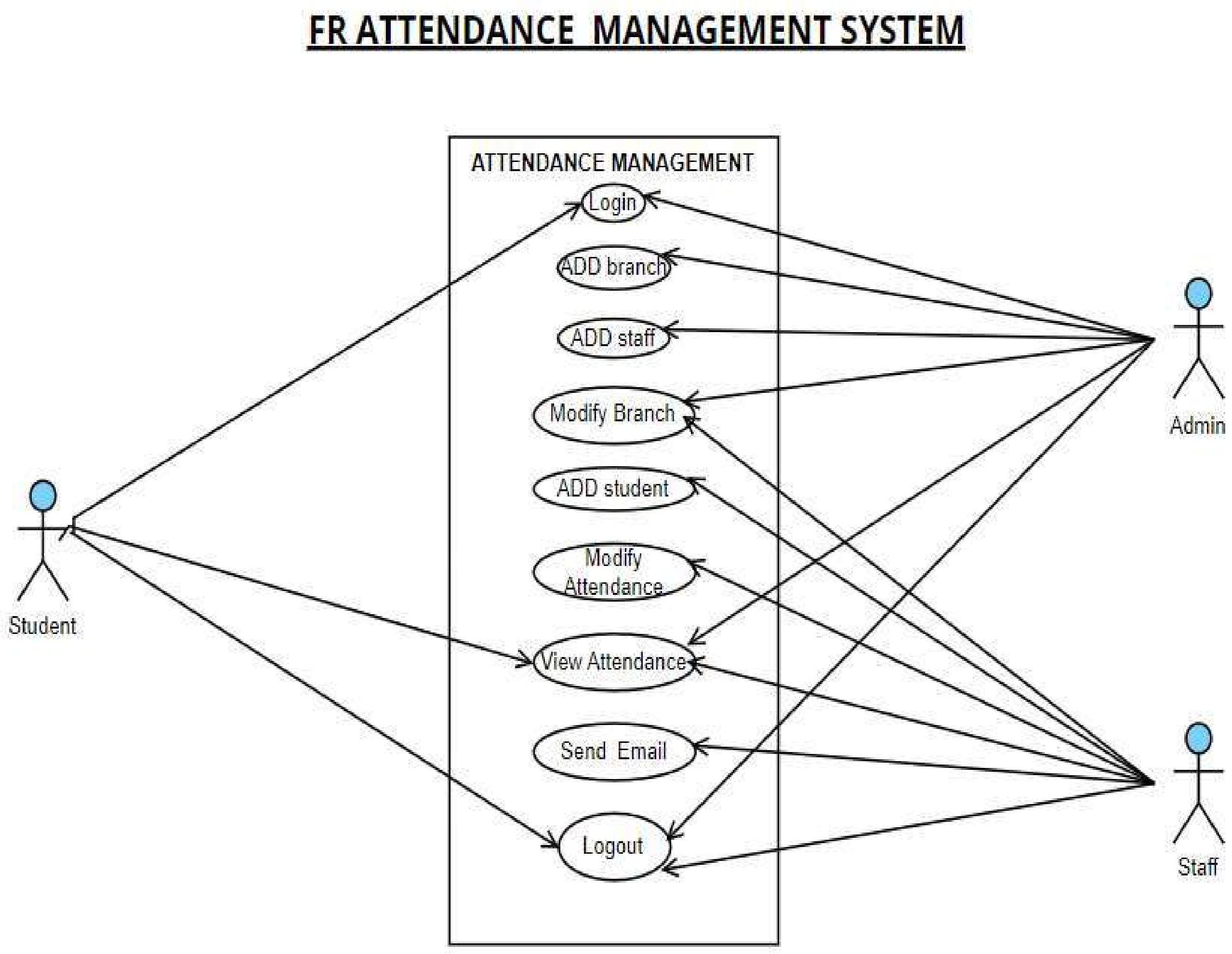
4. Use Case Diagram

A use case diagram is a type of visual diagram used in the field of software engineering to represent the actions that a system or organization can perform, as well as the actors that interact with the system. It is used to model the functionality of a system and identify the requirements of that system.
A use case diagram consists of a number of shapes and symbols, including

Actors: These are represented by stick figures and represent the people or systems that interact with the system being modeled.
Use cases: These are represented by ovals and represent the actions that the system can perform.
Lines: These connect the actors to the use cases and show the interactions between them.

Use case diagrams are useful for understanding the requirements of a system and for identifying the interactions between actors and the system. They can also be used to help identify potential problems or areas where the system may need improvement.

Here is an example of a use case diagram for our system:



The following tables summarize the actors involved and their respective use cases.

Actor: Student

Use Case #	Use Case
1	Login
2	View attendance
3	Logout

Actor: Admin

Use Case #	Use Case
1	Login
2	add branch
3	add staff
4	modify branch
5	view attendance
6	logout

Actor: Staff

Use Case #	Use Case
1	modify branch
2	add student
3	modify attendance
4	view attendance
5	send email
6	logout