SOFTWARE DESIGN DOCUMENT

PREPARED FOR

Dr. Balwinder Sodhi

Indian Institute of Technology Ropar

PREPARED BY

Amritpal Singh (2017csb1068)

Divyanshu (2017csb1074)

Jai Garg (2017csb1081)

Mehakjot Singh Sidhu (2017csb1090)

Bachelor of Technology, 3rd year students, Indian Institute of Technology Ropar

FEB 10, 2020

CONTENTS

1. Introduction

a. Purpose

The main purpose of this document is to describe the design and the implementation details of the Student Attendance Management System which was described in the Software Requirements Specification of the Student Attendance Management System. Its main purpose is to -

1. Provide the link between the Functional Specification and the detailed Technical Design documents.
2. Detail the functionality which will be provided by each component or group of components and show how the various components interact.
3. Provide a basis for the detailed design and development of the Student Attendance Management System.

b. Scope

The Application Design outlined in this document builds upon the scope defined in the Requirements phase.

2. Design Overview

a. Approach

This project is created and extended in multiple phases over the course of the project -

1. Requirements Phase - During the Requirements Phase, the initial version of this document is created, describing the candidate architecture to be validated in the System Design Phase.
2. System Design Phase - During the System Design phase, the Evolutionary Prototype is created and this document is finalized by establishing a sound architectural foundation for the Construction Phase.

3. Construction Phase – During the Construction Phase, this document is not expected to change radically; it is mainly updated to reflect changes in any interface definitions.

4. Transition / Training Phase – During the Transition/Training Phase, no further additions or modifications are made to this document.

b. Architectural Goals and Constraints

The overall architecture goals of the system is to provide a highly available and scalable online Student Attendance Management System for students, teachers and teaching assistants of any academic institute.

The Student Attendance Management System can be used in two ways -

1. To mark attendance of students by the instructor or the teaching assistant by clicking the photograph of the students.

2. To view the current attendance status of the class or individual student.

A key architectural goal is to leverage industry best practices for designing and developing a scalable application. To meet this goal, the design of the Student Attendance Management System will be based on the industry standard development guidelines.

Some architectural constraints limit the responsiveness and hardware requirements of the system. Let's get some ideas about the number of entries in our Master Attendance Table which contains the attendance information of every student for every course for every class, tutorial and lab. More formally, this table contains the entry number of the student, the course code, the instructor or the teaching assistant who marked the attendance, the timestamp of the attendance, and a boolean value denoting whether the student was present or absent.

Let’s say that the number of students in an academic institute is 10,000. Let’s also assume that each student, on average, attends 100 lectures per week (all courses combined), and the institute has 50 working weeks per semester (considering the worst case that the semester is an year-long one). This means that the data in our Master Attendance Table can be of the order of 10^7, which is not very huge in today’s scenario when we have excellent database management systems. Even if each entry requires 100 Bytes, this table would require 10^9 bytes, which is in the order of GigaBytes, which can easily be stored and queried. The problem is with storing the archived data which would also occupy the same amount of memory, but for the last 10 semesters. We need TeraBytes of storage to store the attendance information of the last few semesters. However, the client is not interested in attendance information for each and every day of the last semesters. He would only like to view the overall stats, for which the entries reduce significantly, since now we need to have only the overall attendance of each student for each course, and not for every lecture.

Also, querying this table every time for overall attendance details of a student (which can be a query made by the student) does not seem a good idea because the database might crash due to overwhelming number of such queries. For that, we have a Mini Master Attendance Table which contains the information about the last attendance of a particular student in a particular course. It also contains the current total attendance stats, such as the total number of classes held, and the total number of times a particular student was present. This means that this table will not have attendance details for each and every lecture/tutorial/lab, but the overall stats of a student for a particular course, which is queried very often.

c. Guiding Principles

1. Scalable - Scalability is the ability of the platform to scale both up and down to support varying numbers of users or transaction volumes. The application should be able to scale horizontally (by adding more servers) or vertically (by increasing hardware capacity or software efficiency).

2. Flexible - Flexibility is the ability of the application to adapt and evolve to accommodate new requirements without affecting the existing operations. This relies on a modular architecture, which isolates the complexity of integration, presentation, and business logic from each other in order to allow for the easy integration of new technologies and processes within the application. Our system is flexible because courses are not hard coded, neither is the metadata information of courses such as the attendance policy (passing percentage) and the ratio of attendance value depending upon the type of class (lecture, tutorial, or lab).

d. Technologies to be used

1. Django framework for back-end web development: Django is a framework for backend web applications based on Python provides simplicity (lesser code) and high security(automatically handles common security issues like clickjacking or SQL injection). Django is also suitable for applications to handle large traffic as is expected for our project.

2. React framework for front-end web development: React.js is a dominating frontend JavaScript technology. It's a tool to build UIs - a JavaScript library for building user interfaces. React helps to focus on display and interactivity and has high support of libraries including pre built components reducing large amounts of time. This makes react a sustainable choice, also considering large community support.

3. React Native framework for mobile app development: React native helps to build native applications for android and IOS without having to understand things like Objective-C, Kotlin etc. Also due to similar syntax to ReactJs, this choice will help to reduce learning time and cost.

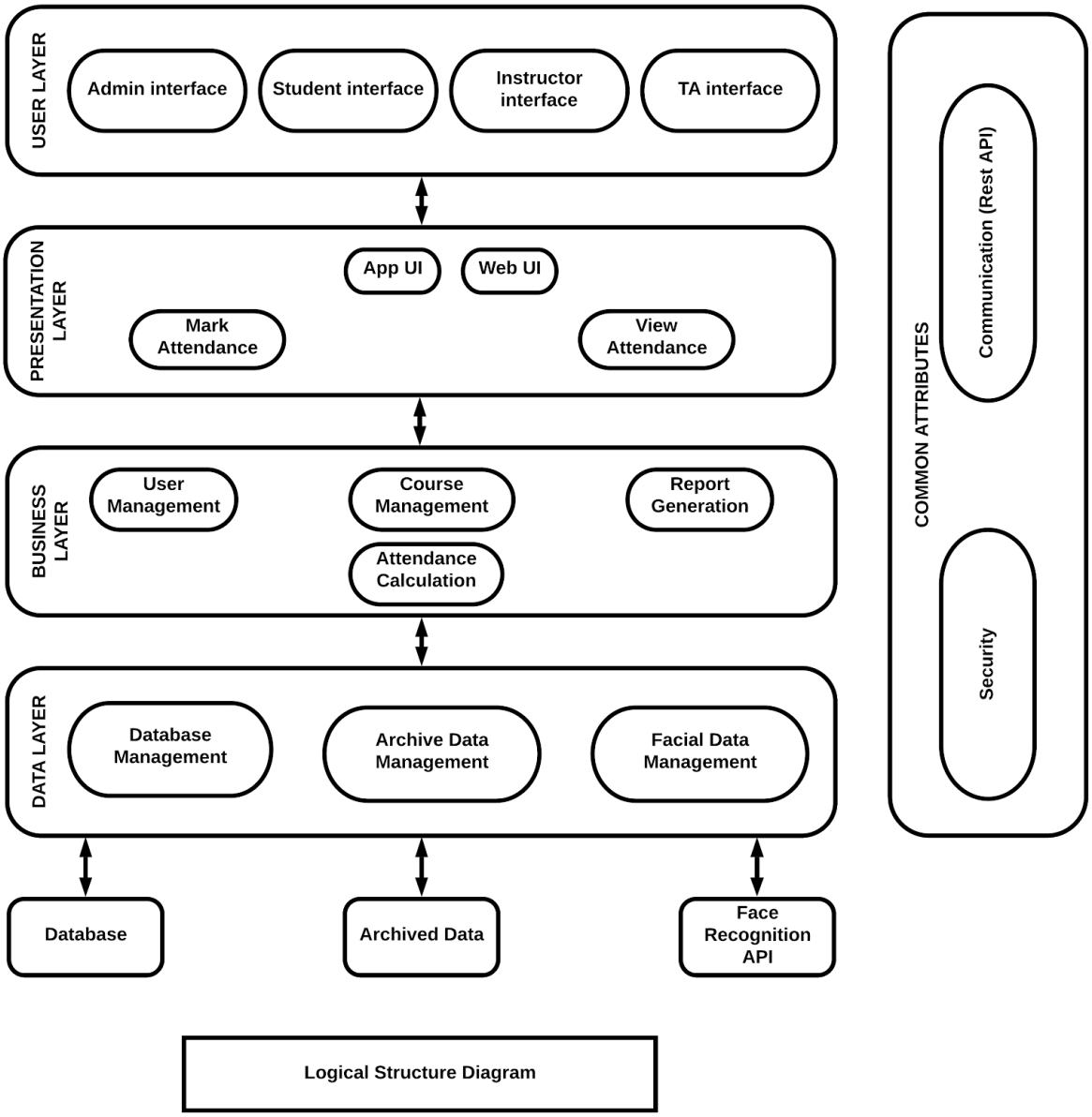
4. Database: Django (with help of ORM) provides freedom of choice for choosing database. Still for implementation we will be using MY SQL, due to its high compatibility with large databases as required for the project.

5. Nginx web server

6. REST : (Representational State Transfer) architecture for communication between systems(server side and client side). REST protocol totally separates the user interface from the server and the data storage and helps in **horizontal scaling.** Rest is independent of the platforms being used, and data transfer takes in XML or JASON format.

e. System Architecture

i. Microservices – Scalability is a major task for the project.To accomplish this the architecture style for the project will be microservices. These are small and lightweight services that execute a single functionality. This also provides ease in adding and removing functionalities or services. The independent services that will be implemented will be Facial Recognition, Generation of Reports, Course Management, and USer Management.



Logical Structure Diagram

f. Design Principles

Best practices and design principles will be applied in two main areas –

i. Presentation Services to individual desktops should be uncoupled –

1. Presentation services are delivered to a web browser rather than to custom client software. A range of modern browsers that support HTML is required.

2. The Student Attendance Management System's user interface will be designed in such a way that common user interface functionality will be implemented in a similar manner across the board. Examples of this include –

a. A uniform way of displaying informational and error messages to the users.

b. A uniform way of displaying required and optional fields in the screens.

ii. Business Rules should be encoded within the application development framework-

1. Business rules will need to be separated from the presentation and database frameworks.

2. Server applications are based on event-based systems. Complex server side event cascades will need to be supported.

3. Adoption of a component based framework needs to be considered to promote reuse of information objects.

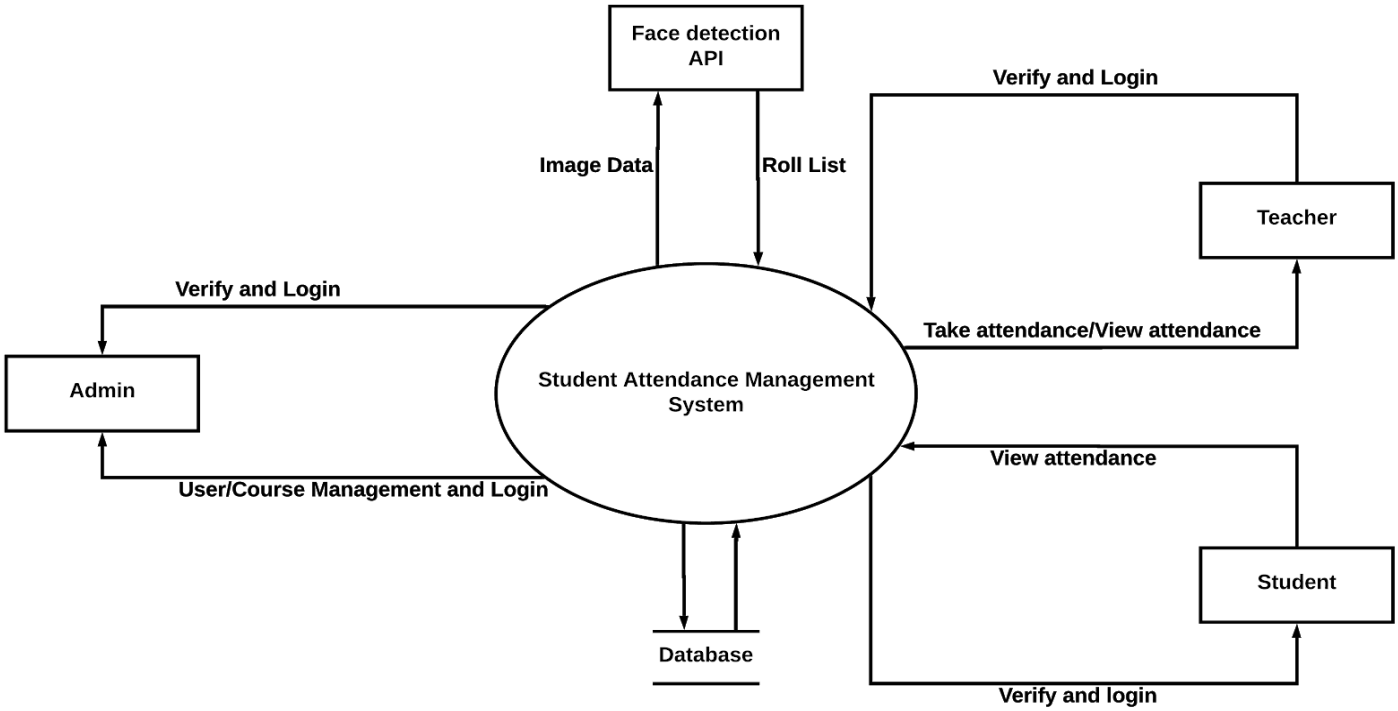
g. User Interface

The focus of user interface will be reachability of the components to the user. For example :

i. Taking Attendance: Since the instructor would have already logged in into the app. To take attendance of a class the user will be 2 clicks away from the desired action. That is, first the teacher will choose the course from the home page and next click on take attendance button.

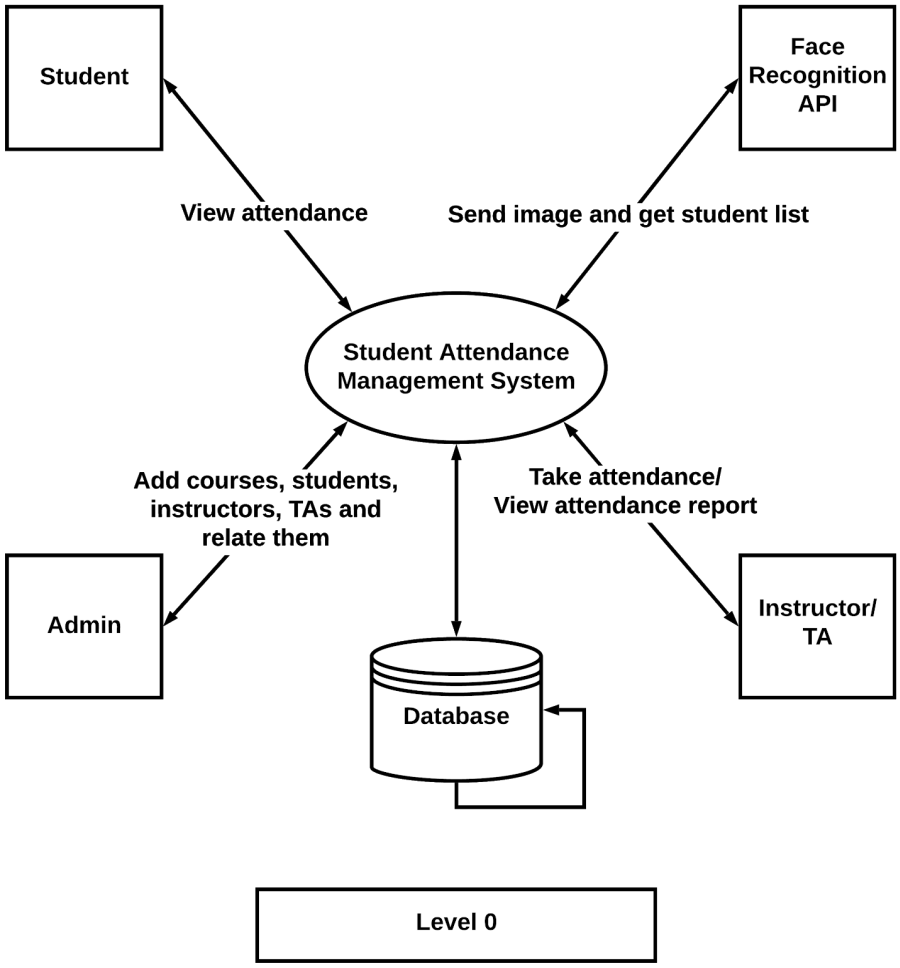
ii. Viewing Reports by students: Similar to the instructor's use case, the student will also be only one click aways from viewing the summary of attendance of the particular course by choosing the course from the home page itself.

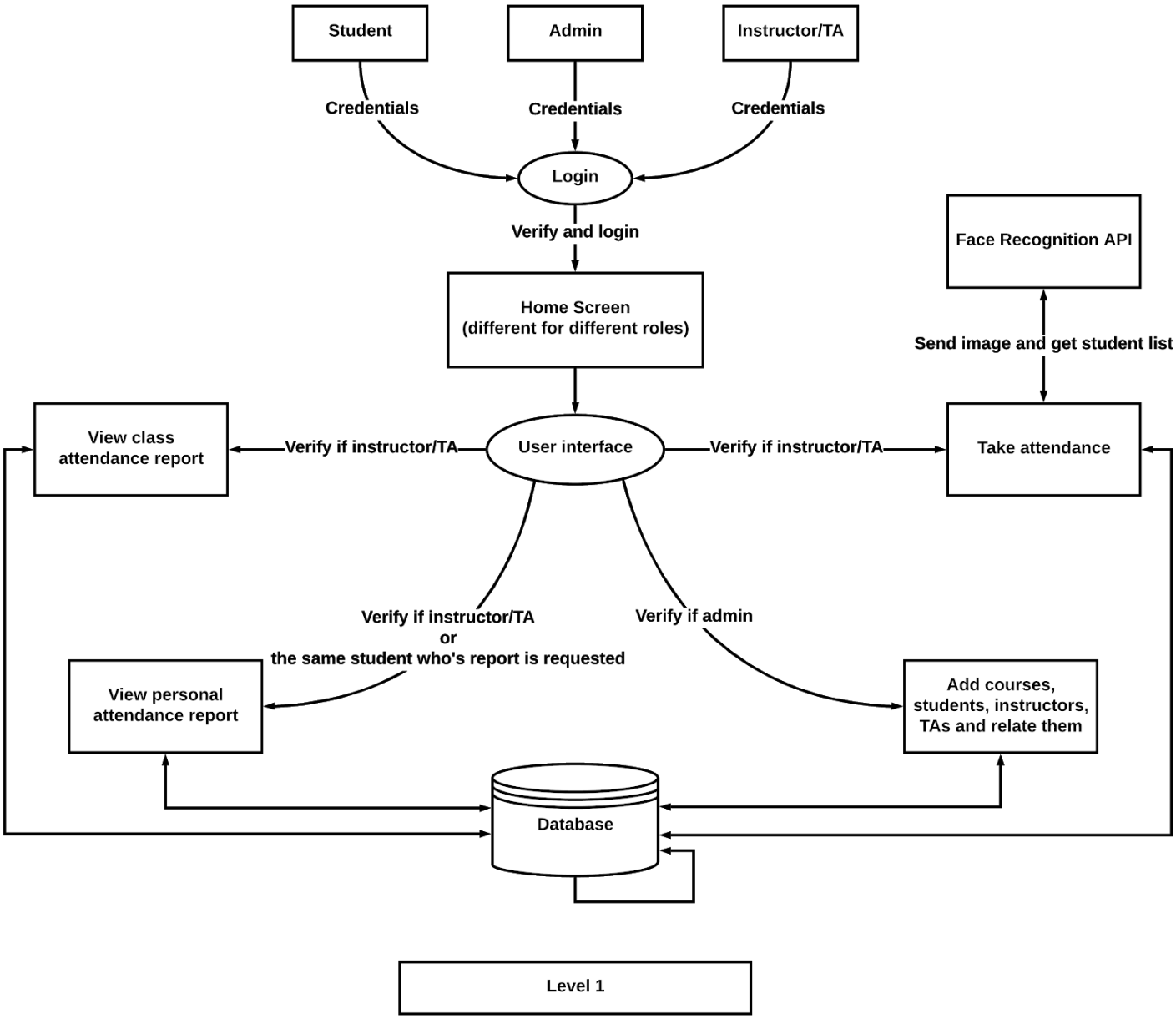
3. Context diagram for the whole system

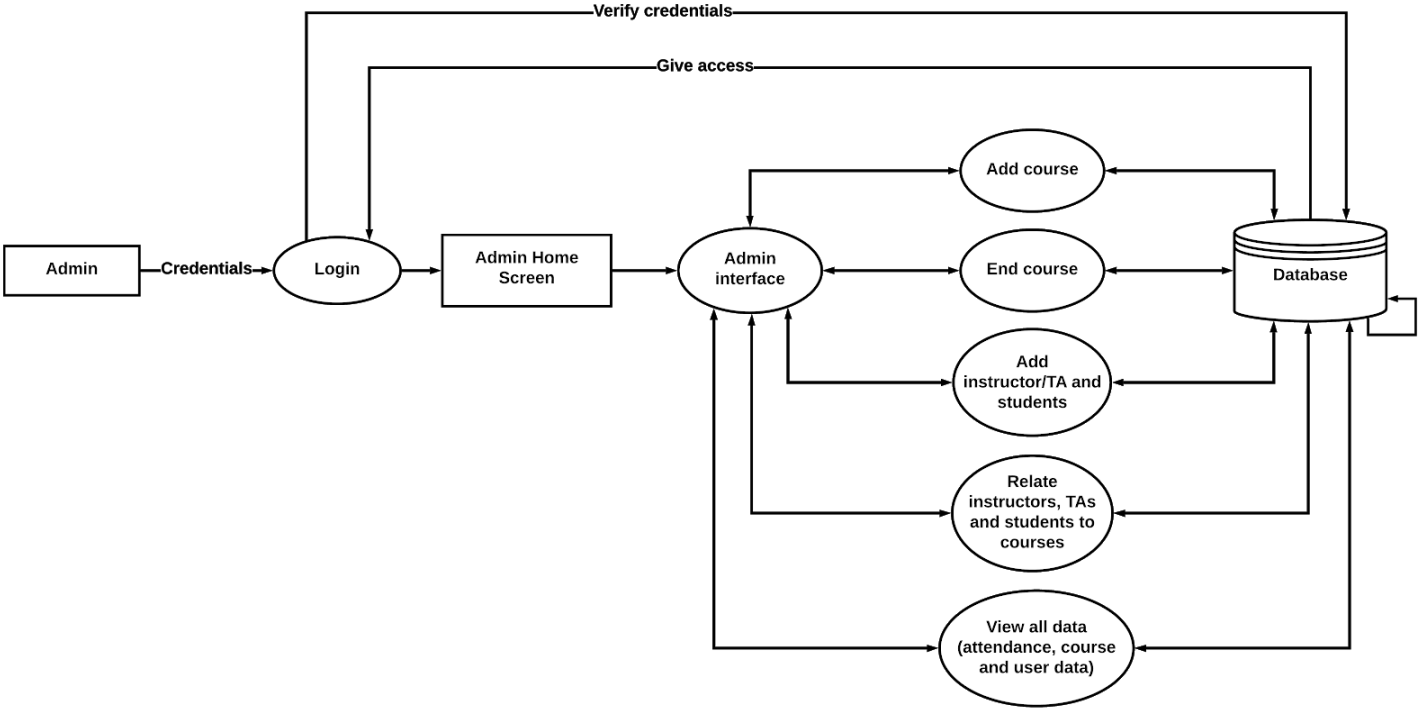


Context Diagram

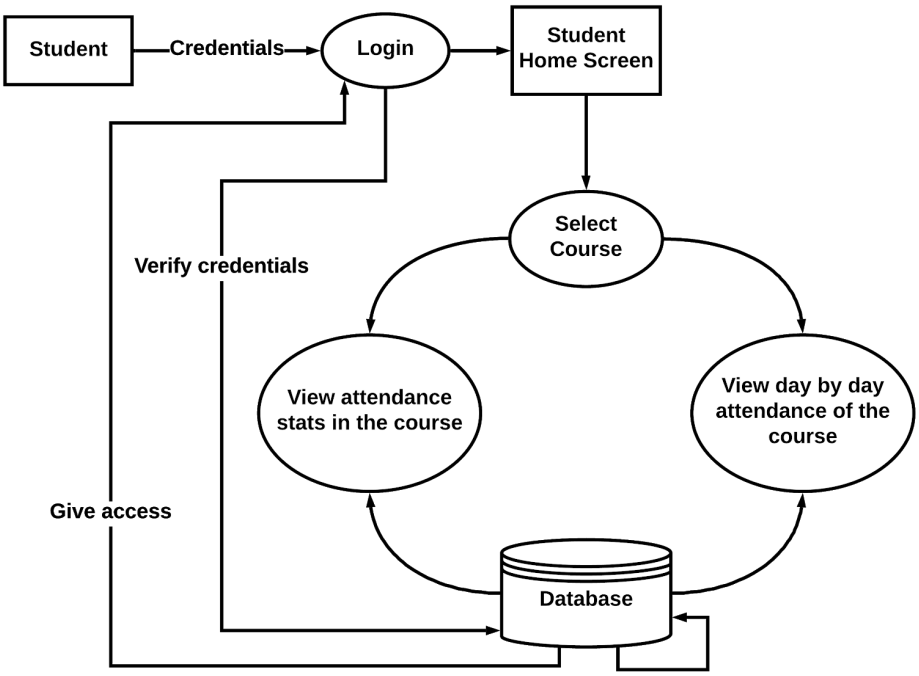
4. Data Flow Diagrams




Student
Admin
Instructor/TA
Credentials
Credentials
Credentials
Login
Verify and login
Face Recognition API
Home Screen
(different for different roles)
Send image and get student list
View class
attendance report
Verify if instructor/TA
User interface
Verify if instructor/TA
Take attendance
Verify if instructor/TA
or
the same student who's report is requested
Verify if admin
View personal
attendance report
Add courses,
students, instructors,
TAs and relate them
Database
Level 1



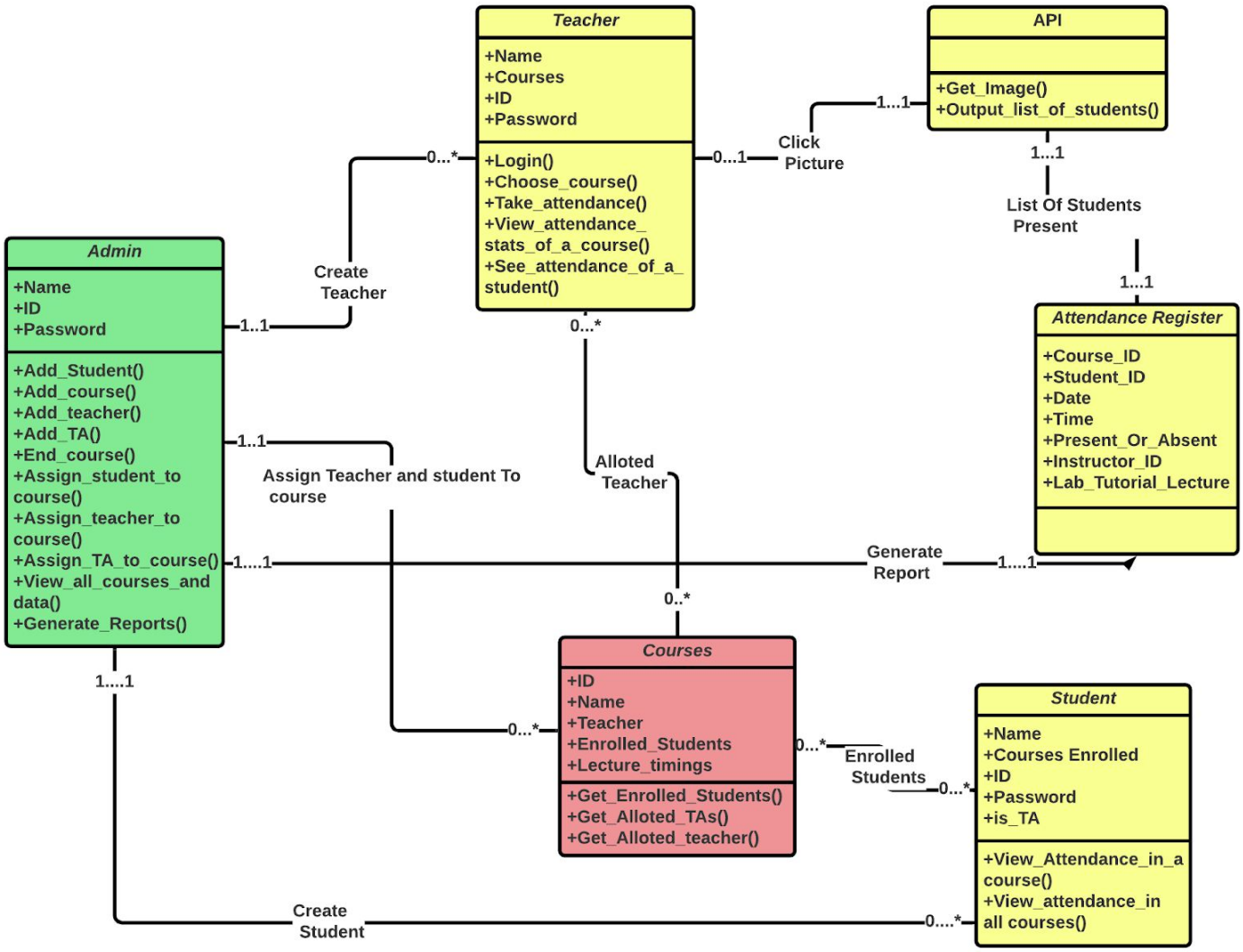
Level 2 (Admin)



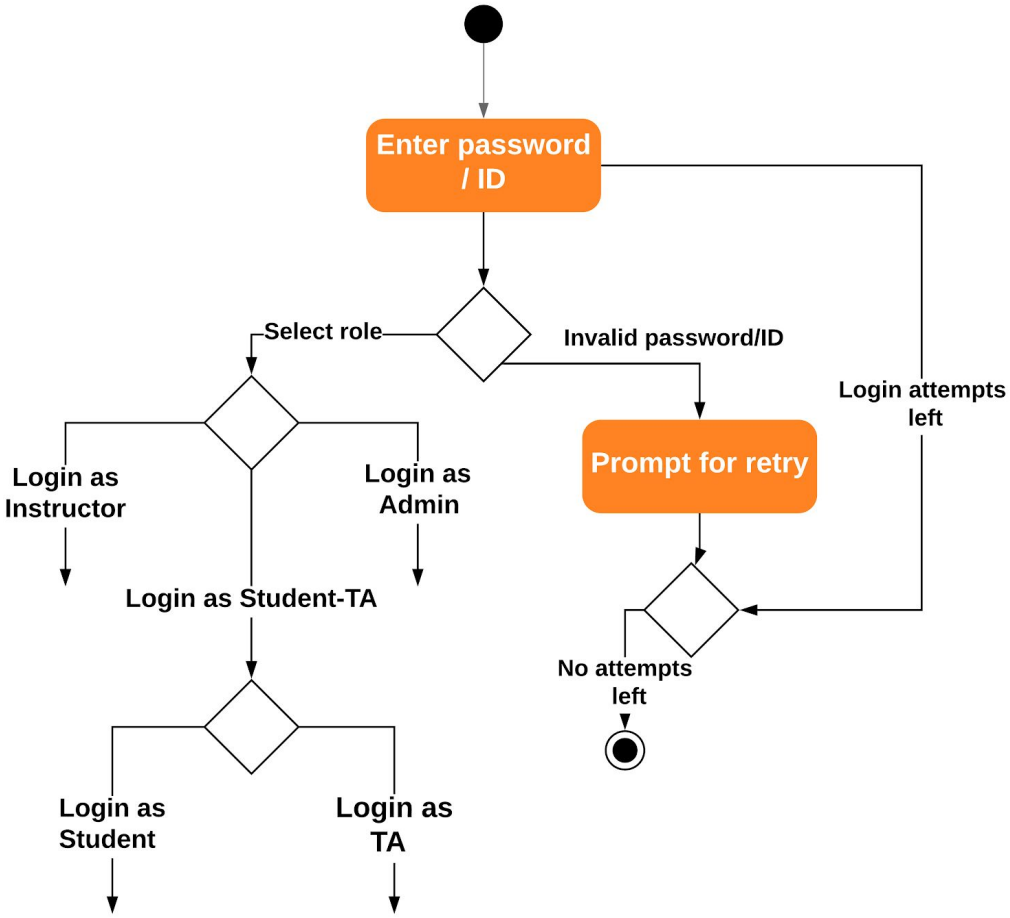
Level 2 (Student)

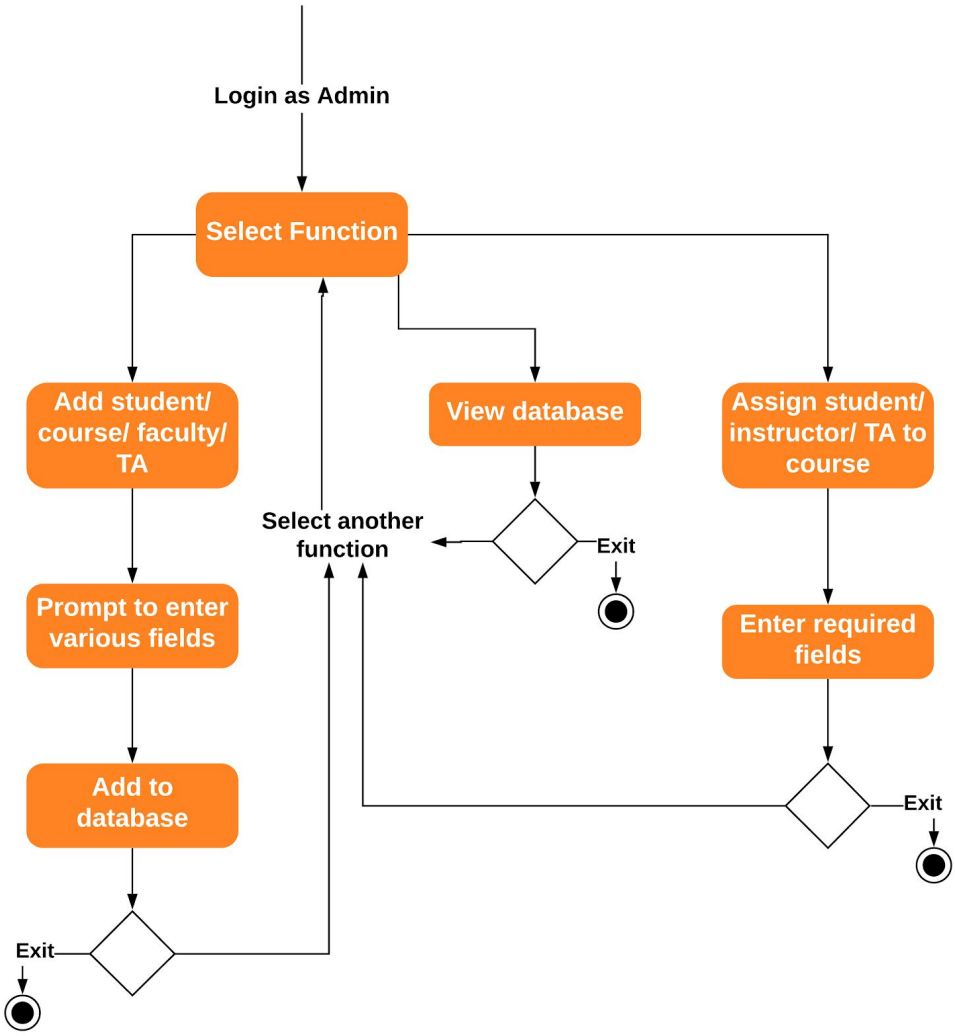
5. UML Diagrams

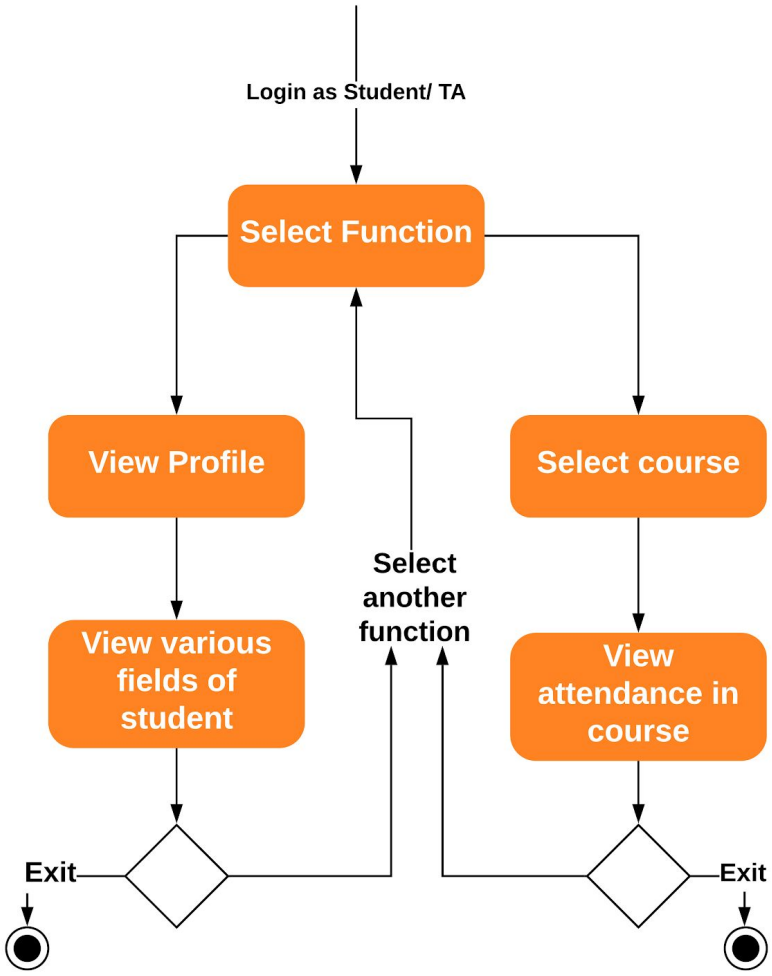
a. Class Diagram



b. Activity Diagram




Login as Admin
Select Function
Add student/ course/ faculty/ TA
Prompt to enter various fields
Add to database
Exit
Select another function
View database
Exit
Assign student/ instructor/ TA to course
Enter required fields
Exit


Login as Student/ TA
Select Function
View Profile
Select course
Select another function
View various fields of student
View attendance in course
Exit
Exit

6. Deployment diagram