IITM-CS3410 : SOFTWARE ENGINEERING LAB

Proxy Report

Aditya Reddy CS14B002

Sai Krishna CS14B018

Revanth Reddy CS14B025

Uday Teja CS14B044

Vedant Somani CS14B053

1 ABSTRACT

The Proxy Application presented in this paper is an attendance tracking application intended for use by universities to track attendance of the students. It allows the administrators to input the photographs of the class and record the attendance of the students based on the face recognition algorithm. This eliminates the issue of proxy in the universities, thereby saving time for the professor with no need to manually check for any issues in the attendance .

2 Introduction

The Proxy Application is a software product intended to be used by the universities to record the students' attendance based on the photographs uploaded of the class. It uses face recognition algorithms to achieve the task. It allows students to input their photographs and the professors to upload the class photographs. The photographs uploaded by the students are used as initial training data for the application. The photographs uploaded by the professor, after resolving any issues that arise, are again used to train the application. This helps in achieving better recognition rate as the time progresses. Any issues that are raised by the students can be resolved by the corresponding course professor via an interface in the application itself.

3 Project General Description

The main purpose of PROXY application is to track attendance from a few photographs of the class. This application has 4 major modules:

- First module is the Professor/TA interface. This interface lets professors upload images of the class, view attendance of any student of that class, answer queries raised by students about their attendance, and manually change the attendance in case the image recognition algorithm makes any error.
- Second module is the Student interface. This interface lets students upload their photos in different angles which will only be used for recognising their face. Students should also enter their details like Name, Roll no, Phone no through this interface. This information will be used for administrative purposes.
- Third module deals with the algorithm part. This is a very important module of this application. A good face recognition algorithm will be used here, so that very less number of errors are made.
- This last module deals with logging every activity done by professors or TA's or students on this application. This log helps in many purposes like recovery from a database crash, improvising this application, to find any malfunctioning part etc.

4 Functional Requirements

The main functional requirements of the Attendance from a Photograph application are the following:

1. The application shall provide a means for entering and storing
    - Courses enrollment information
    - Course teacher information
    - Students' attendance
    - Mappings generated between roll numbers and faces from the photograph
2. The application shall update the attendance of the students by recognizing faces from the photograph of the class uploaded by the professor.
3. The application should have a means for students to refute any incorrect mapping or attendance.
4. The application shall have utility for students to upload multiple photographs from which can be used in the training algorithm to generate the model.
5. The application shall have a provision for the professor to mark attendance for unrecognized faces or correct attendance for incorrectly recognized faces.
6. The application should be able to provide a history of the student's attendance.
7. The application shall log all the actions performed by various users including various queries raised by the students regarding erroneous attendance.
8. The application should be able to give privileged and unprivileged access based on the type of the user.
9. The application should notify the professor of the queries raised by the students.
10. The application shall use a machine learning model for recognizing faces from the photograph.
11. The application shall provide an interface for the professor to manually validate the mappings in case of a mismatch.
12. The application shall provide the exclusive right to modify an attendance only to the professor.

5 Non-Functional Requirements

The most important non-functional requirements of the Attendance from a Photograph application are the following:

1. The application shall be web based and written in Django
2. The face recognition shall be done by using any computer vision training algorithm on the training photos or by using readily available libraries.
3. Remote databases shall be used for storing photos and logs

6 Use Case Description

- **Enter Professor Information**
  Professor Details are stored and each professor is allotted an ID.

- **Enter Course Information**
  Each professor adds the details of the courses he/she are instructing, the roll numbers of students enrolled and the details of the TAs.

- **Upload Class Photographs**
  Multiple photographs of the day can be uploaded which will be sent to the recognition algorithm.

- **Answer Student Queries**
  There might arise the following issues.

  - Attended but marked absent
  - Attended and marked present but someone else's photo
  - Didn't attend but marked present.

  Appropriate actions will be taken to resolve them.

- **Manually Update Attendance**
  In the situation where the photograph is not taken, there is an option to manually update the attendance of the class.

- **Upload Student Photographs**
  The students will upload multiple photographs of themselves in different angles. These photographs once uploaded will go through one time approval by an admin, to avoid the case where a student uploads someone else's photograph

- **Raise Query for Attendance**
  The issues mentioned above can occur and in such case, they can raise an issue about it to the professor. Once raised, the professor and TAs are notified accordingly

- **Check Previous Attendance**
  The student has an option to check the previous attendances of the semester in the history tab.

7 Z Language Specifications

Home = Login

Login :
//Inputs
email?: Valid email address
password? : String

//Outputs
auth! = (Yes,No)
role! = (Professor,Student)

//Actions.
// IsPresent(x) is true if there exists an email x in the database.
// FetchPassword(x) gets the password corresponding with the x email address.
// FetchRole(x) gets the role corresponding with the x email address
auth! = No
type! = Student
IsPresent(email?)^FetchPassword(email?)==password? => success!=Yes
success!==Yes^FetchRole(email?)==Professor => role!=Professor
role!==Professor => **ProfessorHome**
role!==Student => **StudentHome**

ProfessorHome :
//Input
tab? : Selects the tab for further action

//Actions
tab? == ViewCourses => **ViewCourses**
tab? == AddCourse => **AddCourse**

ViewCourses :
// List of all the courses associated with the professor will be shown.
course? : The course to deal with

// Action - Control goes to the CoursePage with appropriate course details

CoursePage :
// Input
tab? : Selects the tab for further action

//Action
tab? == UploadPhotoAttendance => **UploadPhotoAttendance**
tab? == UploadManualAttendance => **UploadManualAttendance**
tab? == History => **History**
tab? == Queries => **Queries**
tab? == Log => **Log**
tab? == Back => **ProfessorHome**

UploadPhotoAttendance :
//Input
date? : The corresponding date
images? : Option to upload multiple images

//Output
mappings! : An array of detected photos with the corresponding names

//Action - Store the photo in the database

UploadManualAttendance :
// Input
date? : The corresponding date
document? : An excel document

//Action - Appropriate entries are made in the database.

History :
// A calendar will be shown and a date can be chosen
// All the mappings of that particular day are displayed
edit? : The professor can wish to edit an entry

//Action - Appropriate entries are made into the database

Queries :
// List of the all the queries sorted datewise are displayed.
// The professor can choose to resolve them.

//Action : A database entry is made when the professor resolves any of them

Log :
// Shows a list of all the manual changes in the attendance made for that particular course.

AddCourse :
// Input
coursename? :
coursenumber ? :
instructorname ? :
instructorId ? :
semester? :

// Action
The course is added and the database is updated accordingly.

StudentHome :
//Input
tab? : Select a tab for further action

//Action
tab? == AddPhotos => **AddPhotos**
tab? == ViewAttendance => **ViewAttendance**

AddPhotos :
// Input
images? : Multiple images can be uploaded by the student

//Output
The images are sent to the admin for authentication and once authenticated, these are appropriately stored in the database.

ViewAttendance :
// The list of courses the student is enrolled in is shown.
// Input
course ? : Student selects the course for which attendance is to be viewed.

//Action
course? != NULL => **SCoursePage** (with appropriate details)

SCoursePage :
// Input
date? : Select the date for which the attendance is supposed to be viewed.
tab? : For further action

// Output
photo? : The photo which is recognised for the day by the algorithm
status? : Status of attendance (Present, Absent)

// Action
tab? == RaiseQuery => **RaiseQuery**

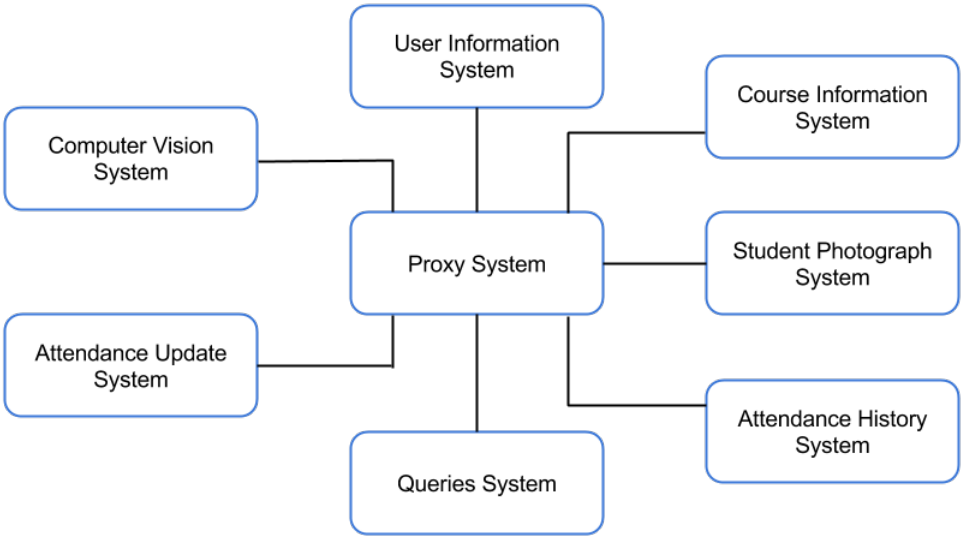
RaiseQuery :
// Input
querytype? : One of the three types of queries can be selected.

// Action
Send a notification to the professor.

8 Context Models

The proxy application is divided into two flows. One flow is for the students and the other is for the professors.

8.1 System Diagram



The various contexts can be illustrated as :

- **Professor information context**
  This context comes under the *User Information System*. In this context, the professor can see his/her details such as name, Email Id, phone, department and room number. The various actions that can be performed are :
  1. *Modify information*
     The action is used to update the professor information. On updating , it returns to the same context i.e the Professor information context.

- **Course Information Context**
  This context comes under the *Course Information system*. This context contains various details about the course. The actions that can be performed are :

  1. *Add Student*
     The action is used to add a student to the course. It returns to the same context.
  2. *View History*
     The action is used to get a summary/history of the course attendance. This action goes to the Attendance history context.
  3. *Give attendance*
     This action is used to manually update the attendance of students. It goes to the Update attendance context.

- **Update attendance context**
  This context comes under the *Attendance update system*. This context contains has the roll numbers of the various students whose attendance is to be updated. The actions that can be performed are :

  1. *Give attendance*
     This action updates the attendance for the students and returns to the Course information context.

- **Answer queries context**
  This context comes under the *Queries system*.It contains the various queries that have been raised by the students in each course. The actions that can be performed are:

  1. *Resolve query*
     This action is used to commit that a student's query has been resolved. The system remains in the same context after this action.

- **Class Photo context**
  This context comes under the *Computer Vision system*. It has the class photo for which attendance is to be obtained. The actions that can be performed are:

  1. *Upload photo*
     This action is used to send the class photograph to the computer vision model to identify the students in the photograph. After this , it goes to the Update attendance context.

- **Update photos context**
  This context comes under the *Student Photograpg system.* It contains a list of all the photos of the student which are being used by the computer vision model to identify his/her face in the class photograph. The actions that can be performed are:

  1. *Upload photo*
     This action adds the photo to the list of student photos. The system remains in the same context.
  2. *Delete photo*
     This action remove the photo from the list of student photos. The system remains in the same context.

- **Student Attendance history context**
  This context comes under the *Attendance history context.* It contains the summary/history of the student's attendance in different courses.The actions that can be performed are:

  1. *Choose date and course*
     This action shows the student attendance for that particular date and course. The system remains in the same context.

- **Student Information context**
  This context comes under the *User Information System.* It is shown by the student home page URL. In this context, the student can see his/her details such as name, roll number, email Id, phone number, photograph and all the courses that he/she is currently taking. There are no actions under this context as the student can only view the information but cannot modify it.

- **Raise Query context**
  This context comes under the *Queries System.* It is shown under student raise query URL. In this context, the student can raise a query. The various actions that can be performed are :

  1. *Submit Query*
     The action is used to submit the query to the corresponding course admins. The student has to choose the course and enter his query in a text field to submit his query. It then returns to View query context.

- **View Query context**
  This context comes under the *Queries System.* It is shown under the stu-

dent view query URL. In this context, the student can view all his queries, both resolved and pending. No actions can be performed under this context as the student is allowed to only view the queries and nothing else.

- **Professor Courses context**
  This context comes under the *Course Information System*. It is shown under professor home URL. Professor can view all his courses and can add a new course under this context. The various actions that can be performed are :

  1. *Choose Course*
     The professor can click on the course name to go to its home page. It then goes to Course Information context.
  2. *Add New Course*
     The professor can add a new course under this action. This takes him to professor addcourse URL. The system goes to Course Information context.

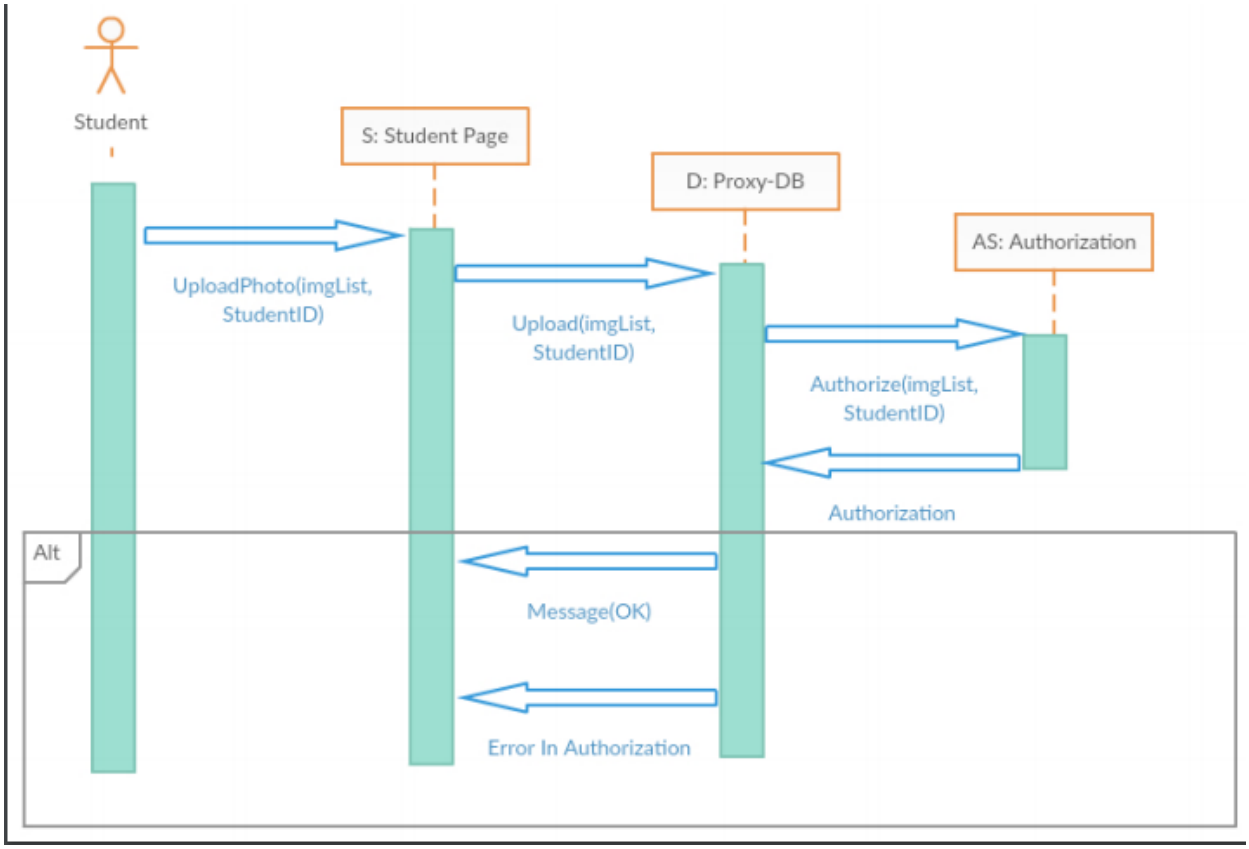
- **Attendance History context**
  It is shown under professor attendance history URL. Professor can view attendance information of a course on a given day which includes stats, photo mappings and unrecognized photos. The various actions that can be performed are :

  1. *Choose Data*
     The professor chooses a date from the calendar UI and the attendance information of the course on that day is automatically displayed. The system still remains in the same context.

- **Authenticate Photos context**
  It is shown under professor authenticate photos URL. The photographs to be authenticated are sorted according to roll numbers and grouped under respective courses. The various actions that can be performed are :

  1. *Authenticate*
     This action leads to authentication of a photo, one at a time. The system remains under the same context.
  2. *Authenticate All*
     The professor can authenticate all the photos of a student with this action. The system remains under the same context.

Use Case : Input Student Photographs
ID : UC1
Actor : Student
Precondition : • Student photographs taken from different angles are supposed to be added / deleted.
Flow of Events : 1. On logging in, there will be a tab 'Upload / Delete photo'. 2. On clicking the tab, there will be a list of all his/her photographs shown to him/her. 3. To delete a photo, the user can click 'delete'. 4. To upload a photo, the user can click 'upload new'. Then uploads a photo and clicks 'Submit'. 5. A request is sent to the admin to validate the photo.
Secondary Scenarios : • No photo uploaded • Admin rejected • Admin Accepted
Postcondition : • Uploaded photographs are stored in a database (based on admins response) and act as our training set.



Sequence diagram for input student photograph

Use Case : Input Student Details

ID : UC2

Actor : Student, Admin

Precondition :

- Student details to be added / modified.

Flow of Events :

1. On logging in, there will be a tab 'Upload / Delete photo'.
2. If the student details are not present, a page asking for the same will show up. Student is asked to input details like.
    - Name
    - Roll Number
    - Phone Number

    The student can then click 'submit'.
3. If the details are already present, the user can modify his/her phone number.
4. Appropriate changes are made in the database.

Secondary Scenarios :

- Insufficient details

Postcondition :

- In case of a phone number change, an email is to be sent to the user stating that there was a change of phone number from X to Y
- The details are stored / updated in the appropriate databases.

Use Case : Input Class Information

ID : UC3

Actor : Professor, TA

Precondition :

- Class Information needs to be entered / modified.

Flow of Events :

1. The actor can add a new class by clicking 'Add class'
2. On clicking so, the prompt asks for the following details
    - Course Number
    - Course Name (automatically updated, if data available)
    - Course Instructor ID
    - Course Instructor Name (automatically updated if data available)
    - Semester

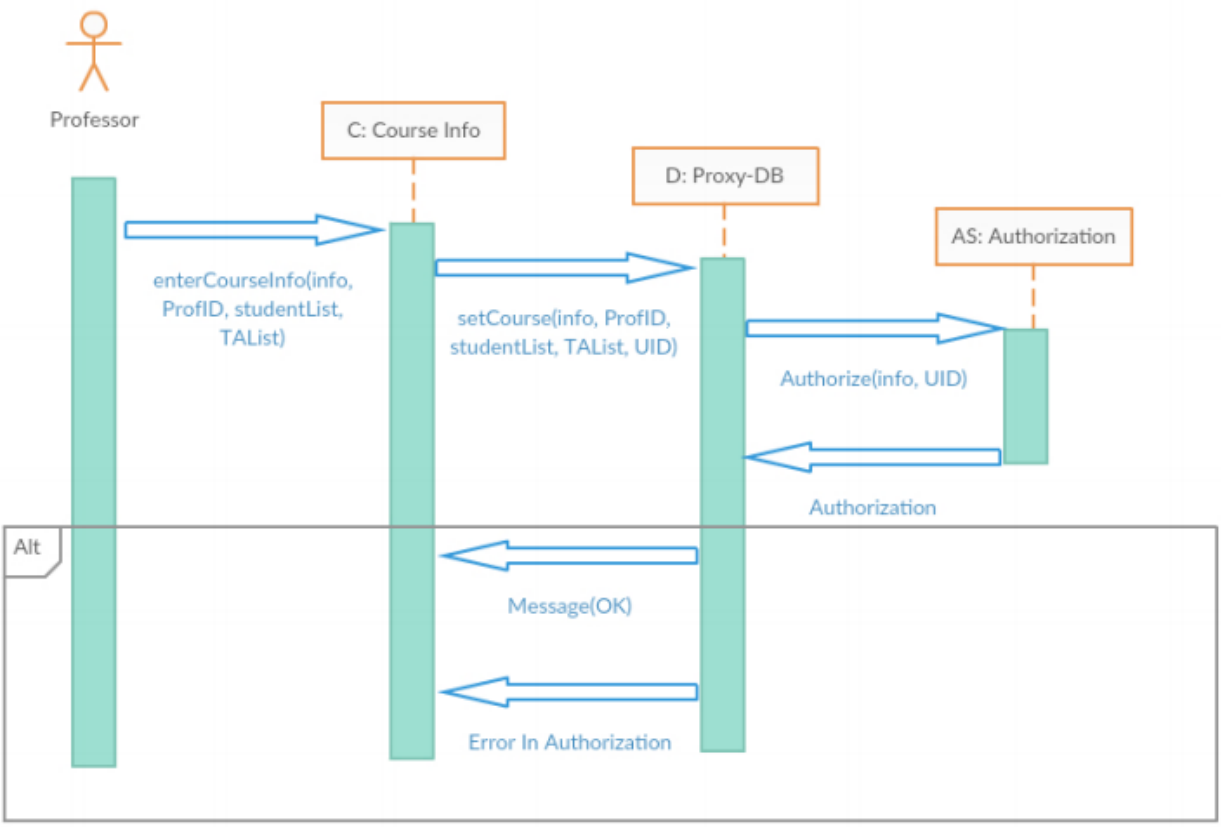
    On clicking 'Add' it takes to another page which prompts to enter the details of the students attending the course
3. The user is asked to enter roll no's of students enrolled.
4. The system validates the entry
5. Appropriate database entries are made

Secondary Scenarios :

- Class already exists
- Instructor does not exist
- Insufficient Details
- Student does not exist

Postcondition :

- The class is saved.



Sequence diagram for input class information

Use Case : Upload Attendance Photograph(s)

ID : UC4

Actor : Professor, TA

Precondition :

- Attendance is to be marked.

Flow of Events :

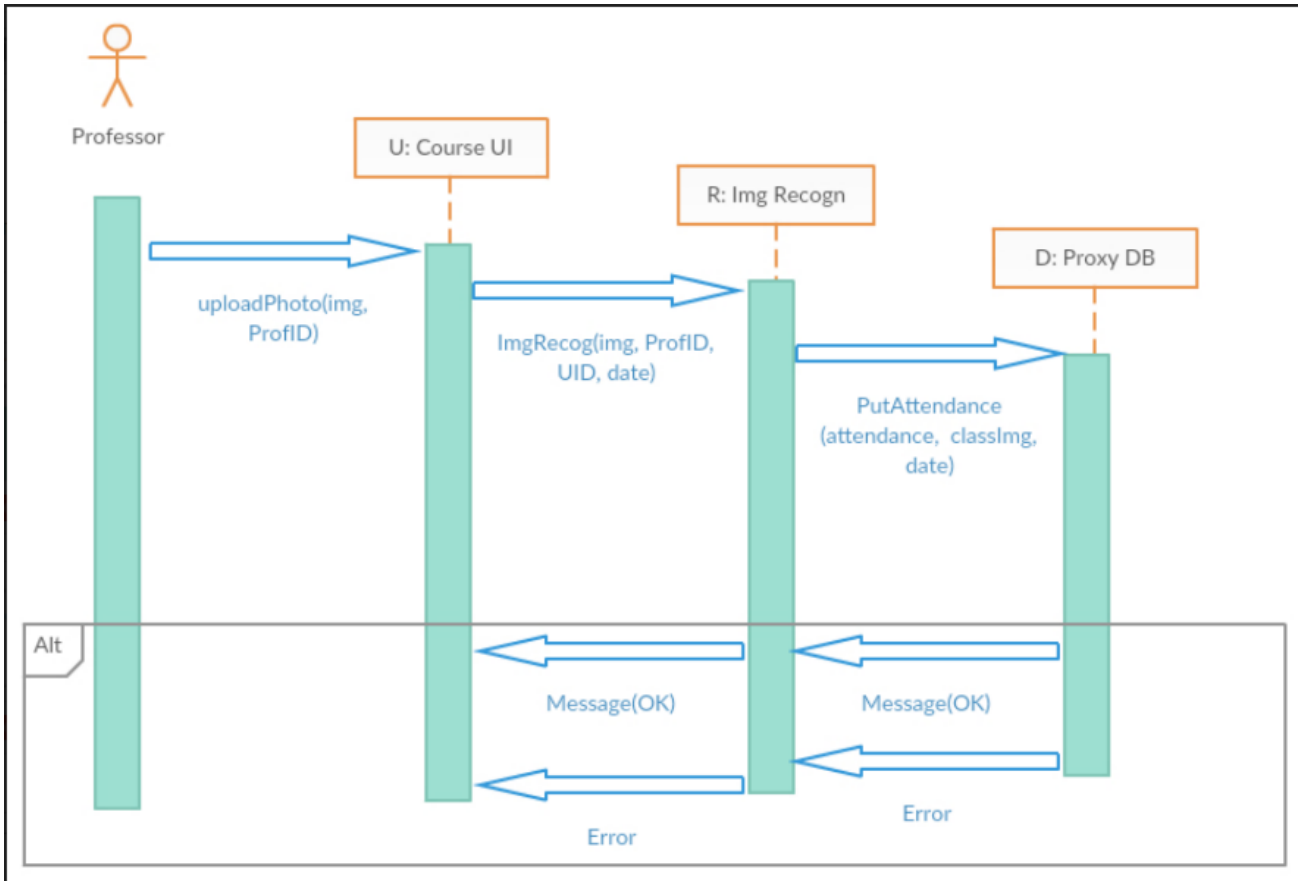
1. All the courses associated with the actor are listed and he/she selects the concerned course out of those.
2. Then a prompt asks for the date.
3. After entering the date, the actor can upload multiple photographs of the class
4. The photographs are stored in the database appropriately
5. On clicking submit, the algorithm is run and the extracted information is stored

Secondary Scenarios :

- Date not entered
- No photo uploaded

Postcondition :

- The photos are stored
- Face recognition algorithm is run
- The extracted information is stored (also used as training data for subsequent queries)



Sequence diagram for upload class photo

Use Case : View Attendance (Student)

ID : UC5

Actor : Student

Precondition :

- To view the attendance.

Flow of Events :

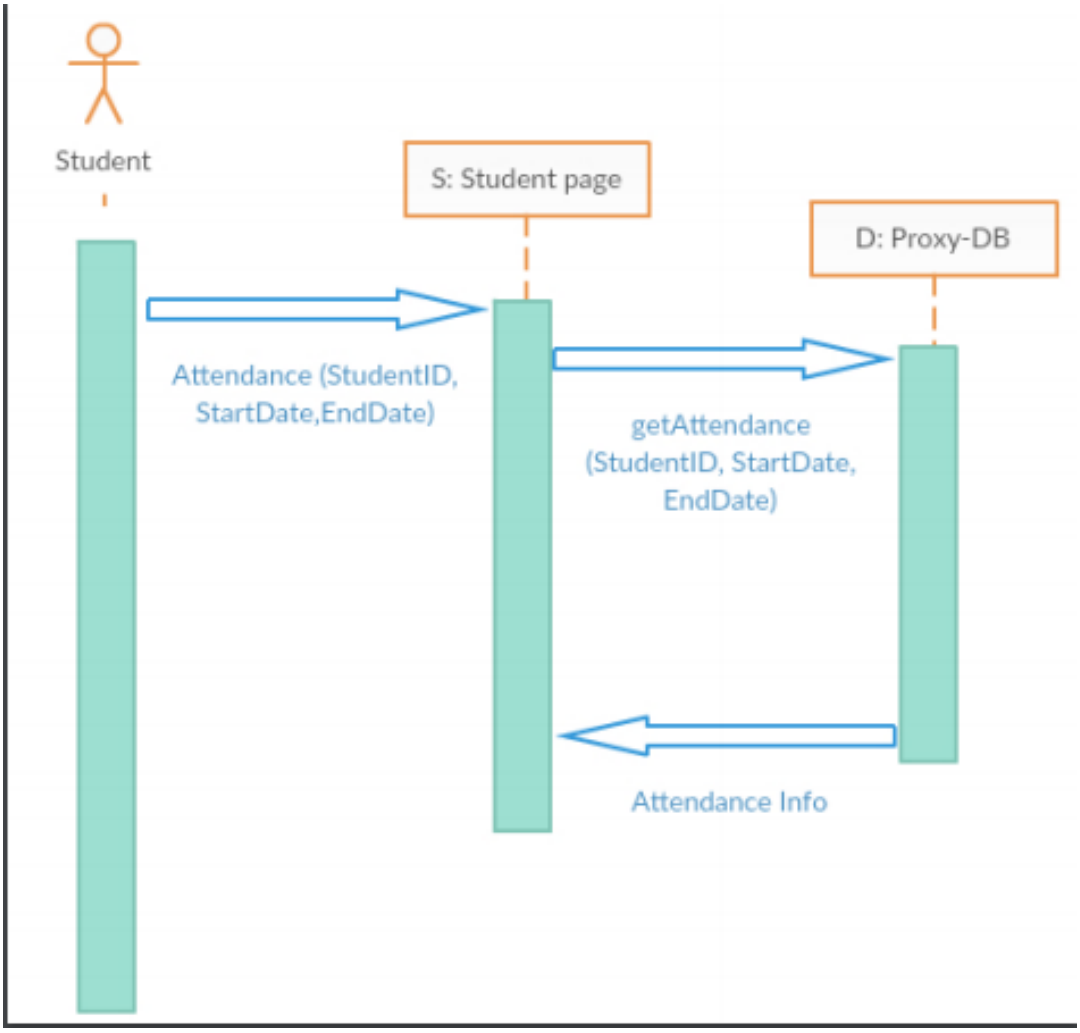
1. A list of all courses the student is currently enrolled into is shown. The student selects the course.
2. There will be two tabs
   - Current Week : Classes happened in the last 7 days
   - History : Earlier Classes
3. On clicking a date, the photo that is mapped on the corresponding date and the status (Present/Absent) is shown.
4. The student can raise a concern. If it is a concern from the current week, the TAs can resolve it, but the history can only be resolved by the professor
5. On raising a concern a log entry is made (which the professor can view)

Secondary Scenarios :

- Resolve an issue
  - Attended the class but marked as absent.
  - Didn't attend but marked as present.
  - Attended and marked as present, but not my photo.

Postcondition :

- A request is sent to professor / TA in case a concern a raised.



Sequence diagram for view attendance

Use Case : View Attendance (Professor / TA)

ID : UC6

Actor : Professor, TA

Precondition :

- To view the attendance and modify the extractions

Flow of Events :

1. A list of all courses the actor is associated with is shown. The actor selects the course.
2. There will be two tabs
   - Current Week : Classes happened in the last 7 days
   - History : Earlier Classes
3. On clicking a date, the uploaded photos for the date are shown.
4. There are also tabs like.
   - Recognised mappings : On clicking which all the recognised photos are shown against the student name.
   - Detected faces but not recognised : On clicking which there will be a list of photos shown which have been detected but the algorithm couldn't associate them with any of the students. The actor can map it to a particular student.
   - Show Consolidated Attendance : Attendance statistics will be shown.
   - Student Concerns : List of all the issues raised by students, which can be resolved according to the permissions set. On resolving these, a log entry is made.

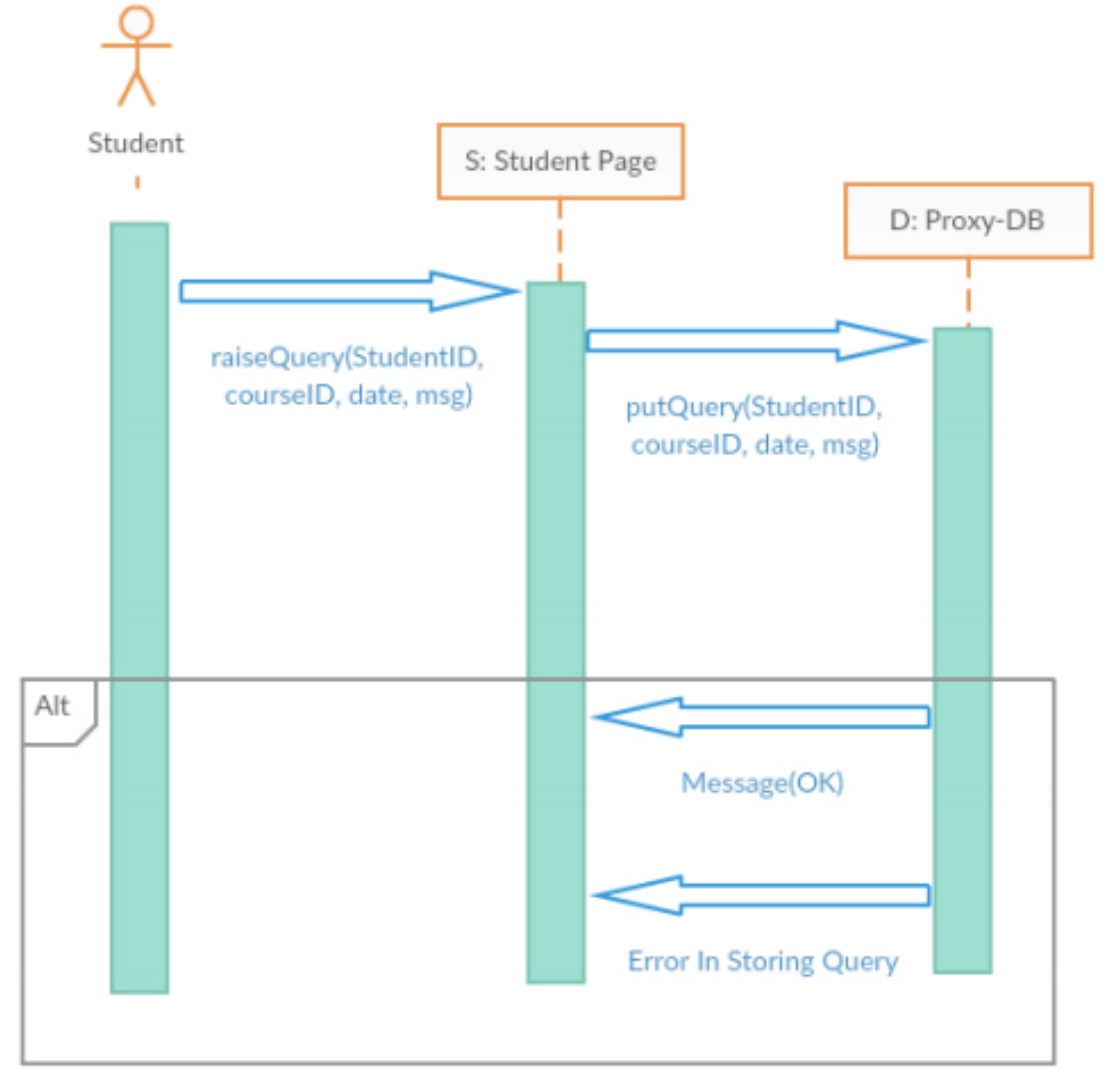
Secondary Scenarios :

- Resolve an issue
  - Attended the class but marked as absent.
  - Didn't attend but marked as present.
  - Attended and marked as present, but not my photo.

Postcondition :

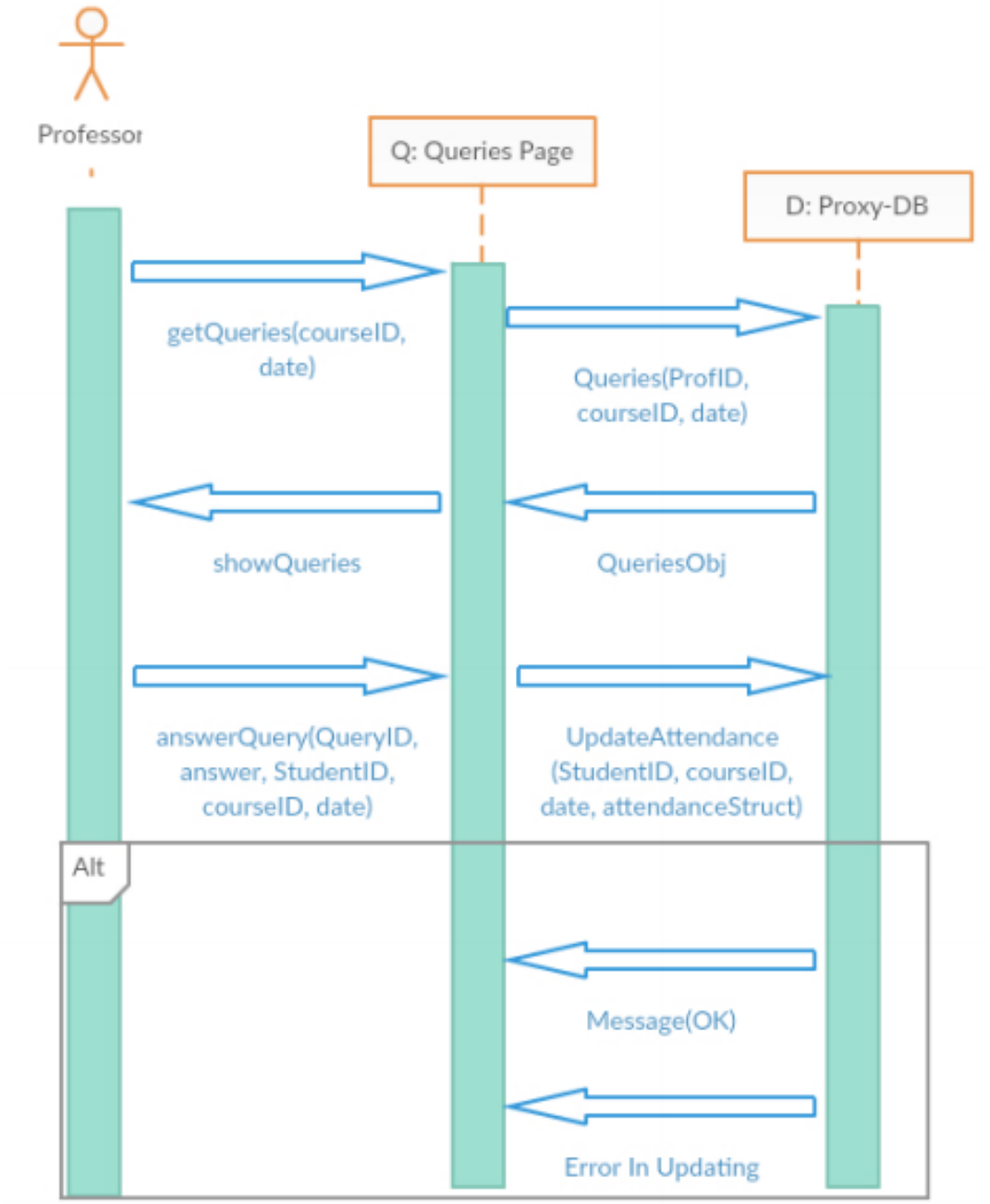
- A request is sent to professor / TA in case a concern a raised.

Use Case : Raise Query (Student)
ID : UC7
Actor : Student
Precondition : • Students can raise queries to resolve issues with their attendance • Issues include : – Attended the class but marked as absent. – Didn't attend but marked as present. – Attended and marked as present, but not my photo.
Flow of Events : 1. On the page the student checks his/her attendance, to resolve issues if any the student clicks 'Raise Query' tab. 2. Student is directed to a mail interface with the From(Student), To (Prof/TA), and the class details prefilled. 3. Student describes his issue and submits the query.
Secondary Scenarios : None
Postcondition : • A request is sent to professor / TA in case a concern a raised.



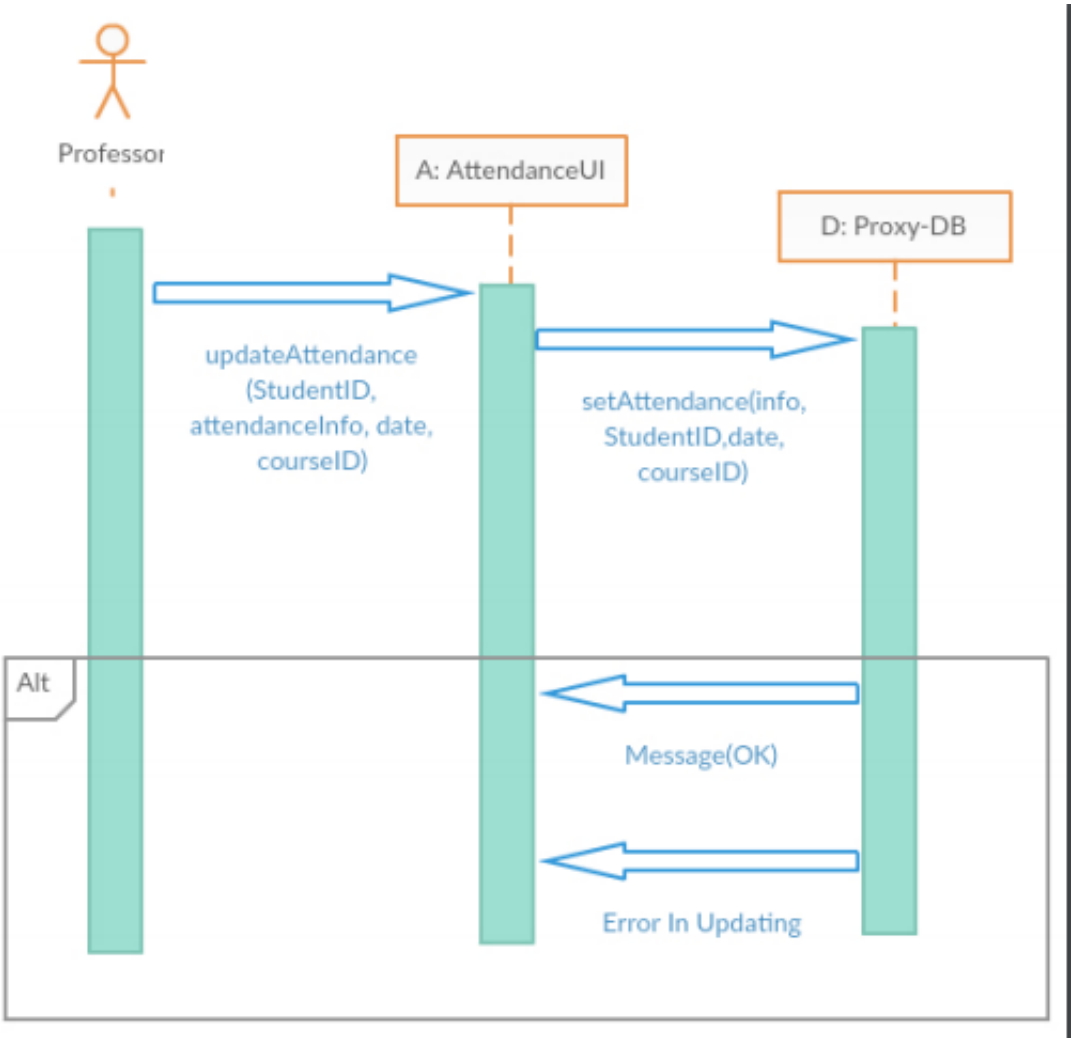
Sequence diagram for raise query

Use Case : Answer Student Query (Professor/TA)
ID : UC8
Actor : Professor/TA
Precondition : • Professor/TA can view the queries raised by students and respond to them.
Flow of Events : 1. Professor/TA views the queries raised by students. 2. Then the professor/TA and student discuss about the query. 3. Professor/TA identifies whether the query is genuine or not. 4. If genuine, professor/TA manually updates the attendance such that the query is resolved. 5. A log entry is made in the database.
Secondary Scenarios : • Query raised by student is not genuine.
Postcondition : • Professor/TA resolves the query.



Sequence diagram for answer student query

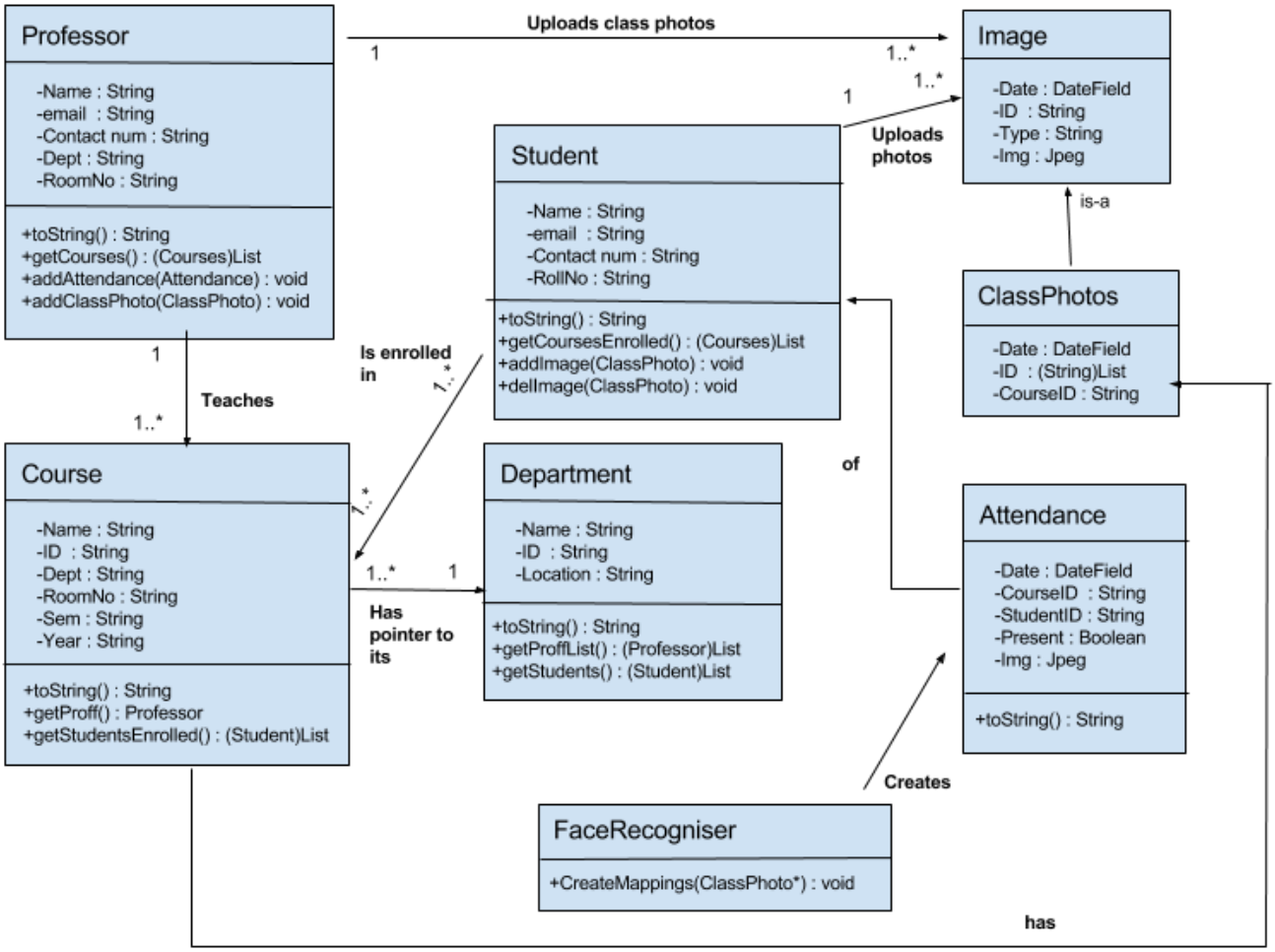
Use Case : Manually Update Attendance(Professor/TA)
ID : UC9
Actor : Professor/TA
Precondition : • There should be an option for the professor to give attendance manually or via photograph. • The photograph has not been taken but attendance is to be updated.
Flow of Events : 1. The professor/TA manually gives attendance to the students based on attendance which was taken in the class. 2. The updated attendance reflects in the student's view attendance portal.
Secondary Scenarios : • No secondary scenarios since the process is done manually.
Postcondition : • It should be mentioned in the student's view attendance portal, that the attendance was given manually. • The students should not have an option to raise queries.



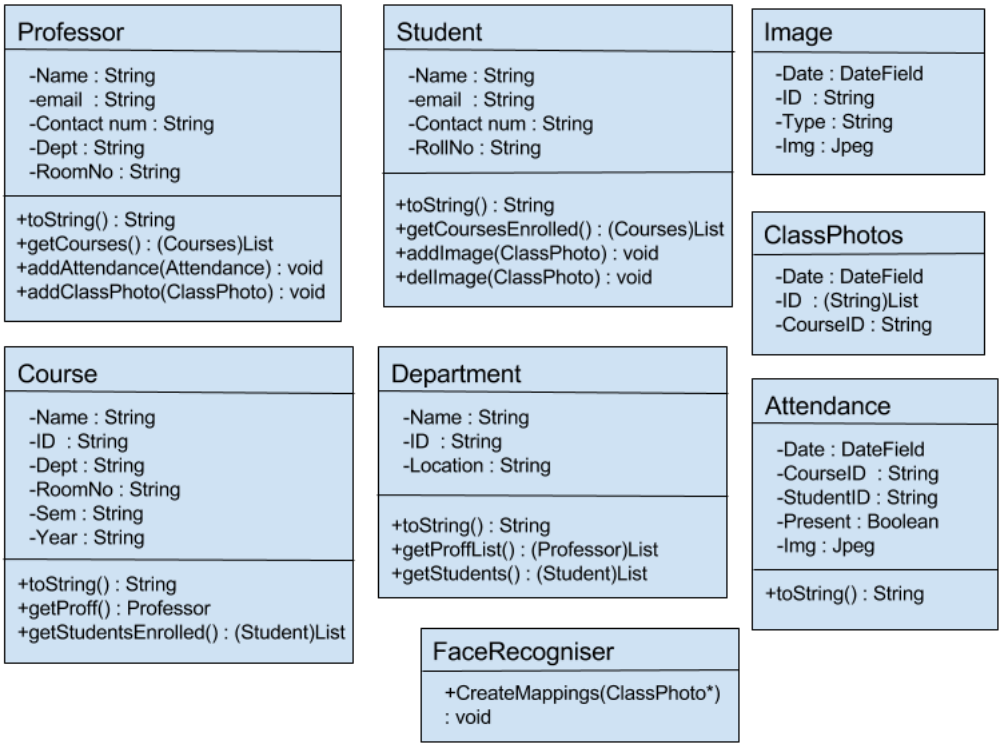
Sequence diagram for manually update attendance

8.3 STRUCTURAL MODELS

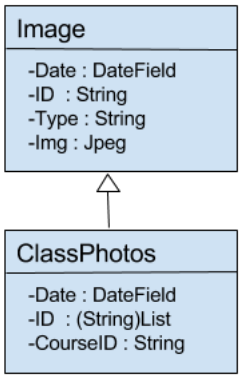
Associations

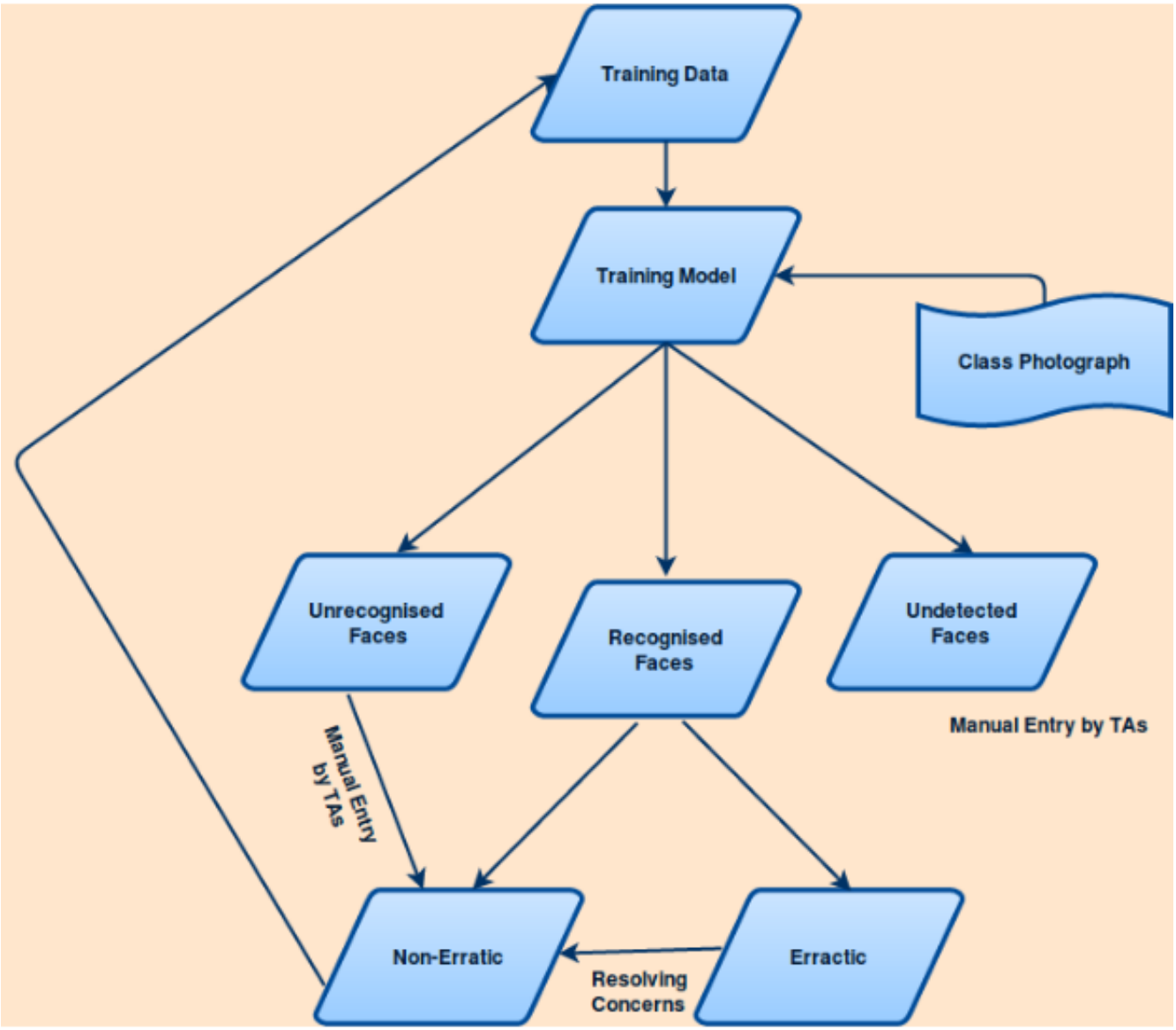


Entities



Hierarchy





Data Driven Model

The photograph on input goes through the training algorithms which is trained by the student's photographs.

There are four types of output types.

1. **Wrongly recognised face** which means that the training algorithm detected that face but recognised it wrongly and mapped it to some other person. This this fall under erratic which is then corrected by the TA / professor after which they turn to become non-erratic and can be used as training data from now on.

2. **Correctly recognised face** which means that the training algorithm not

only detected that face but also recognised it correctly. This falls under non-erratic. This can be used as training data for further uses.

3. **Unrecognised faces**. These are faces which are detected by the algorithm but not recognised to get mapped with a particular student. The TA/ professor Marks the attendance of these people manually.

4. **Undetected faces**. These are the faces which are not detected at all. For them too TA / professor makes a manual entry on request .

9 Computer Vision

For computer vision, we use Microsoft Face Api.4
Here, we illustrate the various tasks that are performed for various user actions:

1. Professor creates a new course.
   Action : For each course, we create something that's called a **Person Group**
   Description : Each course has a person group. Each person group contains various persons who are students. Initially, the person group is empty.

2. Admin authenticates student photograph
   Action : Upload photo to **all the Person Groups** the student is part of and each of these groups is put for **training**.
   Description: Intialliy, once a student uploads his photograph, it is sent to admin for authentication. If authentication is done,t his photograph is used for identifying the student in class photographs. Thus, each of the courses that the student is a part of needs to have this photograph.

3. Professor adds student to a course.
   Action : **Create a person** in the person group and upload current student′s photographs into it. Also **train the person group**.
   Description : If a professor adds a student to a course, we create a person representing that student in the person group. We store the personId and corresponding studentId in the database. Now,all photographs the student has uploaded to date are uploaded to the person group. The person group is put for training.

4. Professor uploads class photograph.
   Action: **Detect Faces** in the photograph. **Identify** the students from these faces.
   Description: We detect all the faces in the class photograph. We send these faces in batches of 10. For each of these faces, we get the corresponding person.

5. Student deletes photograph.
   Action: Remove student photograph from **all person groups** he/she is a part of and put all these groups for **training**
   Description : We remove student photograph from all the courses of the student. These person groups are retrained.

10 Role Isolation

Typically, we would not want professor to access student pages and student to access professor pages. Thus we require isolation between professor pages and student pages. This is achieved by using session variables and method decorators. Session variables are set during login and logout to identify the current user. Method decorators and wrappers for the views check the session variables before executing the views and thus ensure appropriate access to the users.

11 Image Size Standardization

The size of photos uploaded can have a wide range. It can change from as low as 100 Kb for student photographs to as high as 3Mb for class photographs. Thus, we standardize image photographs before uploading to the cloud database. Images are compressed to approximately 40-45Kb size. We use Python Imaging Library for this. One advantage of this is improved speed as uploading larger images to the cloud consumes a lot of time.

12 Logging

For logging, we implemented our own logger on top of Django default logging. The log files are stored in folder logs . The logging info is appended to the end of the current log file in the logs directly. After this file reaches a pre-defined limit, a new log file is generated with a different name.

The way the log info is printed is as follows
[10/Apr/2017 21:24:59] [student,views.py,get:46] [INFO] [g.reva@gmail.com] Retrieved Student Info Successfully

It can be seen as :

1. Date and Time of log
2. Django App name, File name in App, Method name in file, Line number in file
3. Log type : We can have 3 log types , INFO for general messages, ERROR for error messages and DEBUG for debug messages
4. The user
5. Message

Logging in this way makes debugging in multi user applications much easier as we have the exact location in code where something has gone wrong, the time at which it went wrong and the user for which it went wrong.