# WOLF ATTEND: ATTENDANCE MANAGEMENT SYSTEM
# SE-Team Q (Implementation 1)
# Multiple Face Recognition System

## Description

We are implementing the multiple face recognition system for managing attendance of a lecture. We are making an android application thereby making a part for the one-time registration of the students and then that application has an option to take the photo of the whole class. We are using a Rest API from a company named Kairos which has done great work in the field of face recognition, it helps in storing the images in galleries as well as help them retrieving them from the Json objects.

## Done

We have made the basic User Interface with all the functionality from taking a picture to registering is working using the tested application made till date. The User interface is self-explanatory and the users can understand the output as they go along using the application. The communication with the API is achieved from the external way in the command prompt but am presently figuring out how to get it in android application.

We have converted the bitmap image to base64 image using the conversion method that is been written.

## Doing

We are working on communicating with the API there by understanding how to store the picture i.e. image in the gallery when registered and how to recognize the picture when it is ready to be recognized means to be retrieved from the gallery and get the name and user credentials thereby giving him/her marked present when they are present in the lecture.

## Road Blocks

There are some minor roadblocks that we have not yet figured out like how to make the database to maintain the attendance of the students (i.e. the main reason this system is been made). How to get or filter the particular details from the Json object from the API to get the normal output thereby getting the name or user id whatever required for the marking of the attendance in the system.

We are having problems with detection of multiple faces in which there are low quality pictures of the registered picture have been quite different then the new ones.

We are not sure as to how accurate it is also how many people are able to be captured and recognized in a click. According to trial done we had tested for about 7 people but only 5 were recognized. There may have been multiple reasons for such an output but we are not sure as to what the situation can be while doing in the classroom where there are at least 20 students. We are not able to come up with the different unit test cases to be done. Like in this case we have been thinking to make a test case of for the photo conversion. But, we are not able to get about how to go about how to get more of these test cases done.

# SE Team - Q (Implementation 2)
# Attendance marking using Barcode Scanner

## Description

We are implementing a barcode scanner marking the lecture attendance. The idea behind implementing a barcode scanner is that whenever a student enters a classroom or a lecture hall, there will be a barcode put up which will be scanned by the student using our Android application. The student will have login using his credentials before opening the barcode scanner in the Android application. The barcode will have data about the class and once the student scans the barcode, the data from the barcode and his credentials will be entered in the database.

## Done

The barcode scanner has been implemented. It scans the barcode and displays whatever is scanned using the barcode on the screen. This makes use of the Android phone's camera and whatever string is retrieved is displayed on the screen.

## Doing

The UI of the login screen and the user sign-up page is being made. Also, the database connectivity of the Android application with SQLite is being done. The user sign-up page is a bit of a debatable option. However, we plan to use the same database for all the three implementations. So, we are currently designing the database in such a way that it should adapt to the data from all the three implementations. Also, we want to integrate the three Android applications into a single application. So, we are designing the ways in which integrating all the three applications will be better – UI wise and also, it should not hamper the performance of the individual application. Similarly, we had an idea in mind to have two barcodes instead of one before and after the class. So, we are discussing the pros and the cons of using this idea to improve the accuracy of the attendance management system.

## Road Blocks

There are some minor roadblocks that we have not yet figured out like how to make the database to maintain the attendance of the students (i.e. the main reason this system is been made). Is the data which is being stored into the database enough to maintain the attendance or we need to retrieve more information from the user in order to mark his/her attendance. We are having problems with scanning barcodes with longer string lengths so we are yet to decide how much data needs to be fit into the barcode.

Also, we are trying to figure out the need for the existence of the sign-up page. If we are somehow able to authenticate the user using their unity ids by implementing some authentication system like oAuth, we are debating on which one to use.

# SE-Team Q (Implementation 3)
# Attendance system using GPS location tracking

## Description

One of the unique features of mobile devices is location awareness. Based on our January report, more than 95% of the students bring their smartphones to the class, and almost every smartphone has a built in GPS sensor. Therefore, our group came up with the idea which uses the built in GPS sensor to gather the longitudes and latitudes of the students and compare the locations with the instructor's current location. If the distance between the student and the instructor is less than an acceptable distance (e.g. 10 meters), we could safely assume that the student is present and in the classroom, otherwise, the student is absent.

The formula to calculate the distance between two GPS locations is shown below, the result is in meters..

Distance = sin ( lat 1) * sin ( lat2) + cos ( lat1) * cos( lat2) * cos ( lon1 - lon2)
Distance = arccos ( Distance) * 60 * 1.1515 * 1.609344 / 1000

## Done

Since we are building software for smartphones, we've decided to develop an Android application using the Google Play services (GoogleApiClient) and FusedLocationApi. At the current stage of development, we have been able to implement a homepage, where users can choose to login as an instructor or a student. When login as a student, user can click the "Request Location" button to get the current location of the device as a longitude and latitude pair. On the other hand, the instructor can also request the location, in addition, they can also receive locations sent from all the student, and then calculate the distance between them.

## Doing

One of the functionalities which we haven't yet implemented, is the feature that allows student to send data to the instructor. We plan to implement this feature by using Socket, but we still need to learn more about the Android Socket API.

## RoadBlocks

Storing the data can be the most difficult part of this project. Since I don't have any experiences with database in Android applications, for simplicity, we can just store the data in a flat file. However, this approach certainly increases the difficulty of retrieving the data. Therefore, we might eventually use database but for now, we will be using flat file for development and testing. Also, the accuracy of the GPS sensors differ from each smartphone. It is important for us to determine a good acceptable range between student's and instructor's location. Therefore, more work need to be done on this problem.