**NYIT**

**CSCI 665**

**Software Engineering**

**Course Project Final Report**

**Instructor: Dr. Wenjia Li**

**Student name ID**

**Chu Lin 1176181**

**Jingyun Wang 1201338**

**Pak Sam Wat 0789313**

**Xinchen Zhou 1166716**

**Introduction**

Every second counts; time is more precious than gold. Assuming 5 seconds are used for checking 1 student attendance in a class. 2.5 minutes has been spent for a 30 students class. As there are 15 classes per semester, total 37.5 minutes spent. The tuition fee for a 3 credits class is USD3,855 and total lecture time 2,700 minutes, which mean every minute of the lecture is USD1.43. By the simple math, every student spends USD 53.63 on checking attendance. For a 30 students class, USD1,608.75 wasted per class per semester

Our application is an application that target on checking the attendance of a class. The main objective is allowing the students to sign-in at their class simultaneously and minimize the time used on checking attendance. Our idea is student would sign in of a class through an android device. By checking the time when they sign in and their location from the GPS to determine if the record should be treated as valid and put into the database as a record. In the following, the whole process of developing this smart sign-in application will be shared and a manual handbook will be provided at the last of this report.

**Planning**

Our team have the first discussion right after we decided the android sign-in application is going to be created. Since we are a new formed team, we share our experience, technical skill and programming language with each other in order to have a better specialization in this project.

We all agree the agile programming method is much more suitable for us as we are not confident to have everything well planned in the first step. For instance, we have a severe discussion upon the database design. Therefore, the agile method is flexible for us to add what we need during the whole development process. Besides, according to our living style, our team is capable to have a 24/7 programming over the whole process. At last, several tasks are divided and assigned to all members with acknowledged schedule.

XinChen, Zhou (Kira) – In responsible of being coordinator in the team, main frame of application developer, debugger and taking red-eye shift

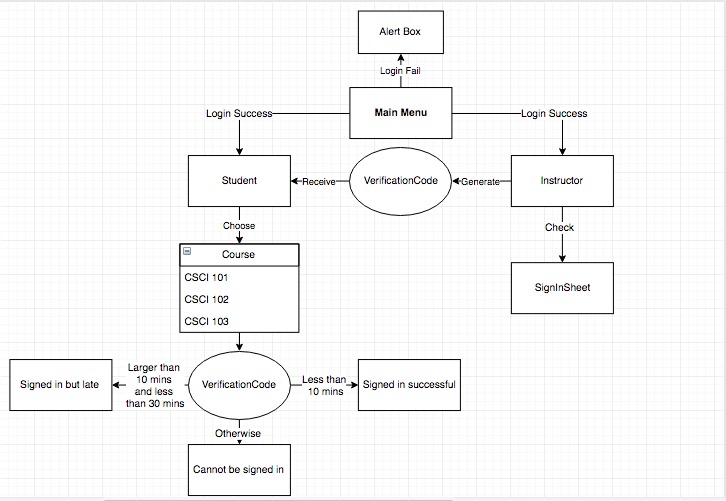
JinYun, Wang (Anderson) – In responsible of writing query for inserting data, developing time different calculation function and dealing with time datatype mismatch on different platform and taking night shift

Lin Chu (CC) – In responsible of role playing as client, developing function of obtaining GPS form the local device and calculating the distance between two GPSs location and taking night shift

Pak Sam, Wat (Sam) – In responsible of building database, managing data in database, developing the function of attendance checking at instructor side, taking morning shift.

**Design**

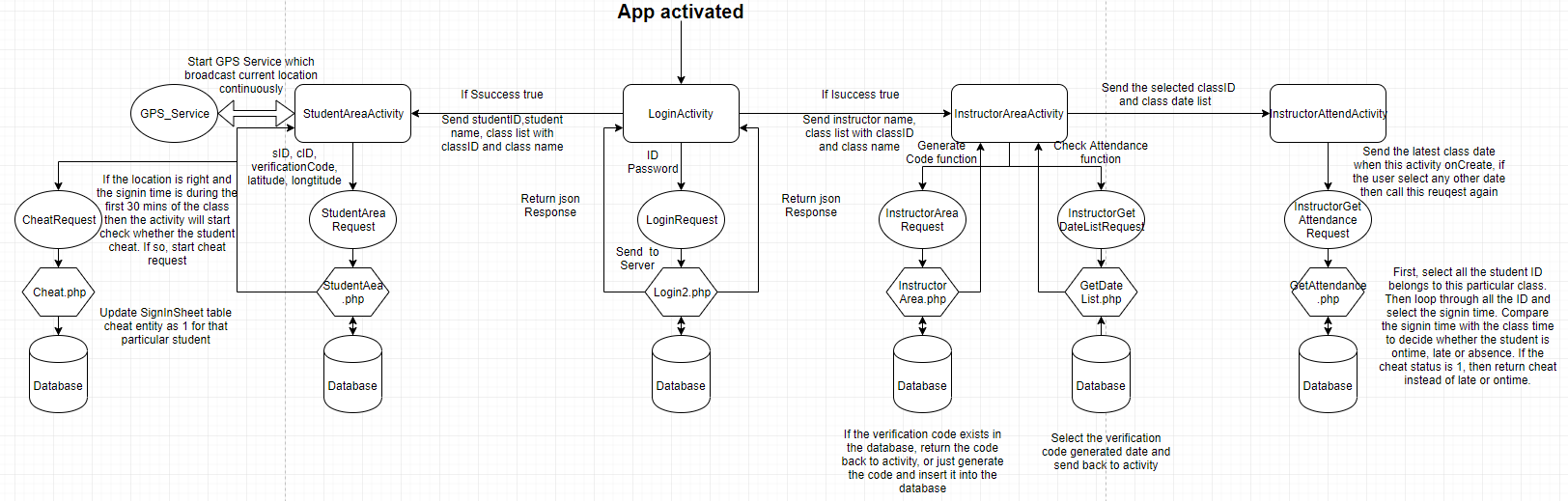
The goal of our application is to minimize the time spent on checking the attendance in a class. We, therefore, have set several objectives in order to achieve it. First of all, the use of the UI should be as simple as possible. Ideally, pressing buttons only with not entry. Besides, the speed of running code and accessing database should be guaranteed. In the other word, the hierarchy of the coding should be well considered. The following is the basic design of the sign in application.



Besides, the class in Android Studio have to separate precisely. It is because every member incurs in the coding and we have to have a better idea what function is carry out by particular class. The same thing is required in the php coding.

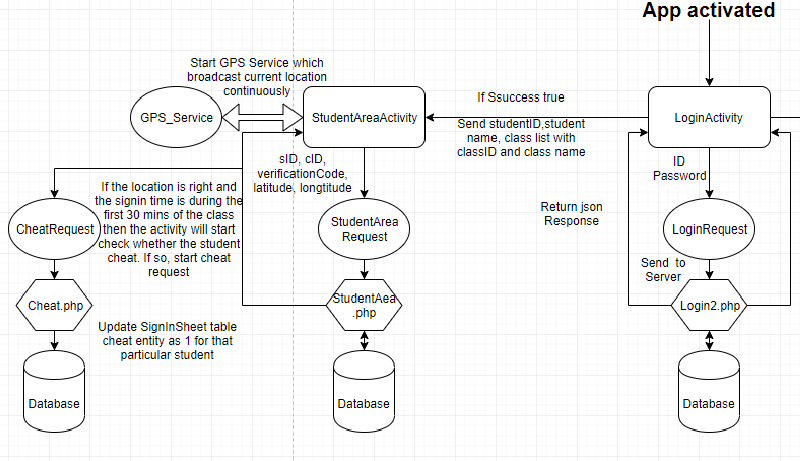
The design of the database is also one of our concern. Since almost every use at the frontend would result a change at the database, the database has to been precisely normalized in order to avoid the update anomaly.

**Application Structure**

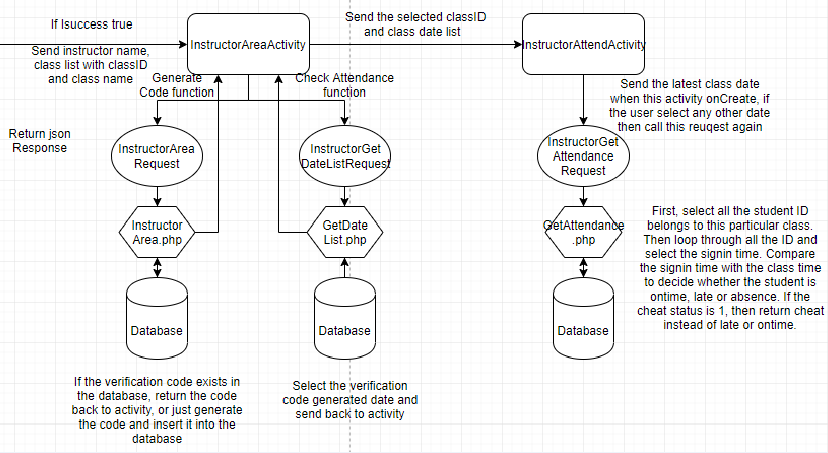


There are four main activities and a service in our sign in system. when App is activated, the LoginActivity will let user to input the password and user name, then it will check if it is student or teacher or invalid account, the data from loginActivity to loginRequest, then send to server

and database.



If student login suceessfully, LoginActivity will send studentID, student name, class list with classID and class name to studentAreaActivity. StudentAreaActivity will do two thing: starting GPS service which broadcast current location continuously, sending sID, cID, verificationCode, latitude, longtitude to studentAreaRequest, then through studentArea.php to database. During the sign in process, if the location is right and the sign in time is during the first 30 mins of the class then the activity will start check whether the student cheat. If so, start cheat request. After it, update sign in sheet table cheat entity as 1 for that particular student.

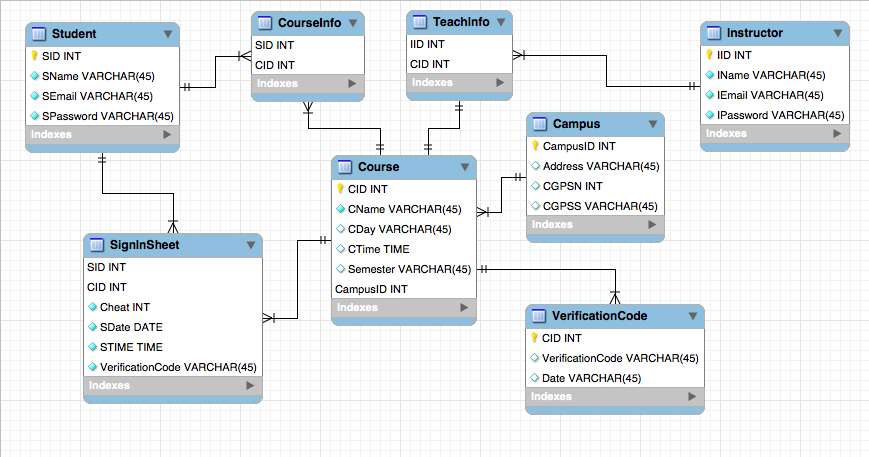


If instructor login successfully, LoginActivity will send the instructor name, class list with classID and class name to InstructorAreaActivity, which has two functions: one is gernerating code function, if the verification code exists in the database, return the code back to activity, or

just generate the code an insert it into the database. The other one is checking attendance function, select the verification code generated data and send back to activity. InstructorAttendActivity sends the latest class date when this activity onCreate, if the user select any other date then call this request again.

**Implementation and challenges**

The database is the first thing we focused on since it acts as the heart of this whole project. None of the function would be carried out without the support of the database. We spent a whole week to discuss, implementing query and improve the database structure. The challenge we faced is the modification of the database structure. As we have our client add and change the requirement during the development of the application, the initial database structure would not be capable to fulfill all the requirement. It has to be modified to fully implement the function. Based on the knowledge on database management, we have applied major modification and normalization in total 3 times. Below is the final table schema which allows the application to be fully functional.



In our project, before implement any other functions, we need an online database to store the data and a login interface to communicate with the user. Fortunately, I found Tonikami TV’s tutorial videos on Youtube [2]. I completed login interface as our starting point by following the videos. And I chose 000webhost as our project database as the videos introduced. The first thing needs to be changed is that we don’t need the register part since we are assuming that we can dump the school database into our database. Next difference is we have two different types of accounts: instructor account and student account. We have two tables contains the account and password, so we need to go through both tables in order to find the information that user input belongs to which type of account.

The next challenge was the class list. After logging in, whether he/she is a student or instructor, a class list should be presented. I decided to use a spinner to do this because the user also needs to select the class to work with. However, all the data that spinner needs should be collected before the activity is created, or another button to refresh the spinner is needed which is not we expected. Therefore, the response from the server during the login phase, besides the user authentication, should also contain the class list that related to that user. In this way, the login activity can inherit the data to student activity or instructor activity.

# In student account interface, we need GPS function to check the student’s location. Filp Vujovic’s Android - Get GPS location via service [3] is the tutorial I followed to get the android GPS function works. Another thing needs to be mentioned in this activity is how to check whether the student signs in this class the same day twice using the same device. During our discussion, we realized that our application would make cheat which means sign in for others more convenient. Unlike the signature has that person’s identity, it is almost impossible to discover whether the student signed in for others afterward if we don’t have anti-cheat algorithm. The algorithm we came up with is creating a local text file to record the information when the user successfully signed in.

# 

# If we got the response from the server said that the user signed in successfully, we will start the check. We are not doing that before is because the user may enter the wrong verification code or any other reason leads to the sign in fail. In this case, if we check the cheat before, the user will be wronged.

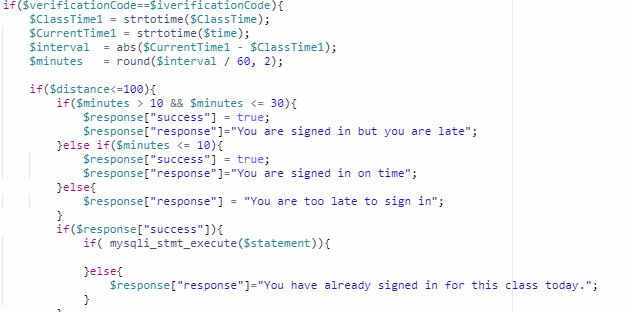
# First, let us focus on the else part which means there is no cheat found here, we will just record the signed in class ID, current date, and the signed in student ID to a certain local text file.

# Now, let’s go back to the if statement. It means after looking into the certain local text file and found this class is signed today before. Then we would consider it as cheat.

# It is impossible to sign the same student in twice for the same class in the same day, because in this case, the server side would return the message said you have already signed in. The only possibility left is this device is used to log in a different student account and perform the signing in.

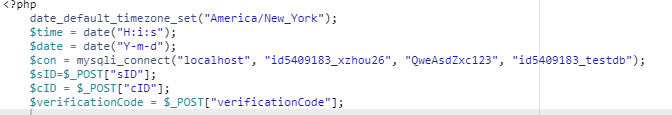
# We take the student ID which is recorded in the text file before out as cheatID, and make request to send the student ID current logged in and the cheatID to the server. The server side php would update the database for both students as cheat.

In carrying out the function of checking if the student has signed late or not, we tried to compare the student attendance by comparing the current time of student signed in and the course time on that specific date. We implemented this function mainly in php on our server. Below is the basic function to calculate the time difference:

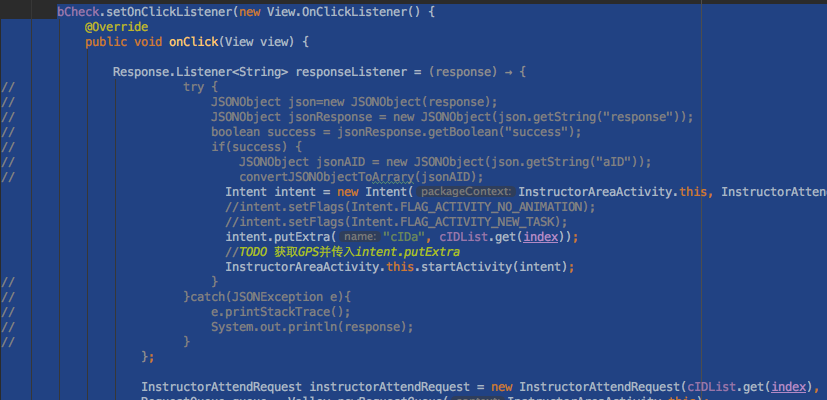


We first want to check if the time difference is bigger than 10 mins and less than 30 mins because if it is between this range, the student is late to the class, but he/she can still sign in. Another scenario is when the time is less or equal 10 mins which means students will be signed in on time. However, if the students sign in time is not in these range, students will not be allowed to sign in.

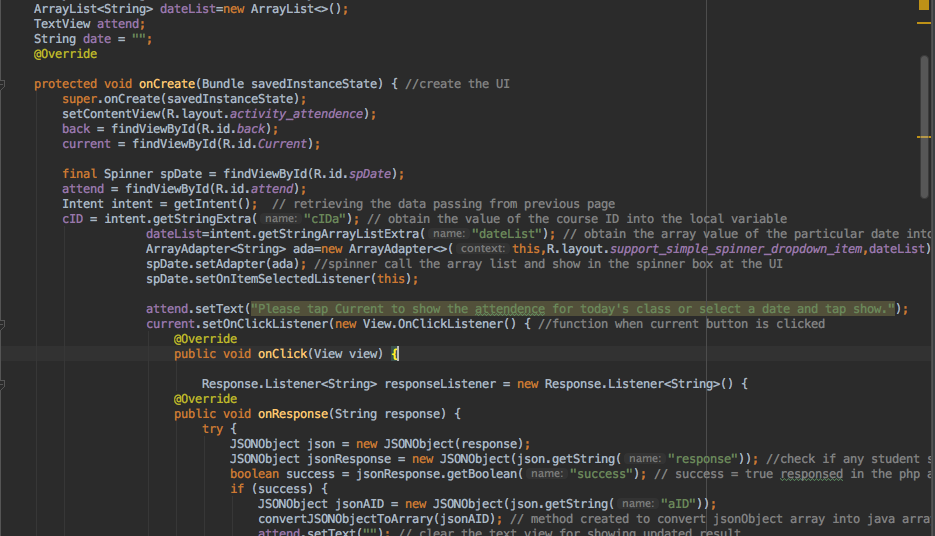
During the implementation, we find out the time zone had some problem because every time we insert a new time in the table, the time will be 4 hours later than the current Eastern standard Time. We are assuming the server might be in different region that is later than Eastern Standard Time. To overcome this problem, we added a new default in our StudentArea php.file. Below is where we set the new default time zone:



To the part of printing the attendance, we tried to implement this function by pressing a single button. Once pressing, the instructor would able to choose if the current date of attendance would be shown or a particular date of attendance would be shown. While implementation, we found that there are too many buttons on a single page. It makes user feel confused while using it and the coding in the class is not easy readable at all. Therefore, we decided to separate the check attendance function into another class and also another page.



The above has been simplified by the intent function which is used to inherit the desired data to other class. The exact way we make it work is to store the course ID which chosen after instructor login in. When the user wants to check the attendance, the application will lead the user to another page and the course ID chosen will be sent to other class for processing.



By this, the burden of the class after the instructor sign in has been eased. Also, the layout at the application has been improved to be more readable.

To getting the student real time position and calculate the distance between student position with the course location to make sure that student is at the class. The first step is creating a java class to get GPS position when sign-in button is clicked, second thing is delivering this GPS data to server, according to course information

it will give back a distance value, then deliver this data to sign in function module to check if this student sign in successfully.

Core function code:

<1> get gps position

locationManager=(LocationManager) getApplicationContext().getSystemService(Context.LOCATION\_SERVICE);

locationManager.requestLocationUpdates(LocationManager.GPS\_PROVIDER,3000,0,listener);

<2> deliver the data

public void onLocationChanged(Location location) {

Intent i=new Intent("location\_update");

i.putExtra("latitude",location.getLatitude());

i.putExtra("longtitude",location.getLongitude());

sendBroadcast(i);

}

<3> caculate distance

function getDistance($longitude1, $latitude1, $longitude2, $latitude2, $unit=1, $decimal=2){

$EARTH\_RADIUS = 6370.996;

$PI = 3.1415926;

$radLat1 = $latitude1 \* $PI / 180.0;

$radLat2 = $latitude2 \* $PI / 180.0;

$radLng1 = $longitude1 \* $PI / 180.0;

$radLng2 = $longitude2 \* $PI /180.0;

$a = $radLat1 - $radLat2;

$b = $radLng1 - $radLng2;

$distance = 2 \* asin(sqrt(pow(sin($a/2),2) + cos($radLat1) \* cos($radLat2) \* pow(sin($b/2),2)));

$distance = $distance \* $EARTH\_RADIUS \* 1000;

if($unit==2){

$distance = $distance / 1000;

}

return round($distance, $decimal);

}

<4> check and get the active Permissions

private boolean runtime\_permissions() {

System.out.println(Build.VERSION.SDK\_INT);

if (Build.VERSION.SDK\_INT >= 23 && ContextCompat.checkSelfPermission(this, Manifest.permission

.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED && ContextCompat.checkSelfPermission(this, Manifest.permission

.ACCESS\_COARSE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {

requestPermissions(new String[]{Manifest.permission.ACCESS\_FINE\_LOCATION, Manifest.permission.ACCESS\_COARSE\_LOCATION}, 100);

return true;

}

return false;

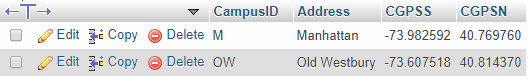
}

The locationManager did not get the GPS\_SERVICE, there were two reasons: one was that I failed to set listener to update the GPS change,

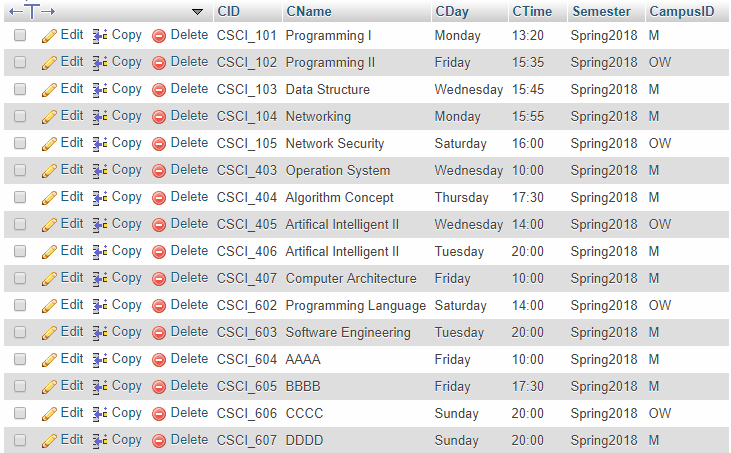
another one is that I forgot to register the permission in AndroidManifest.xml. After using android studio debug tool, I found out where the problems are and fixed them.

**Database Screen Shot:**

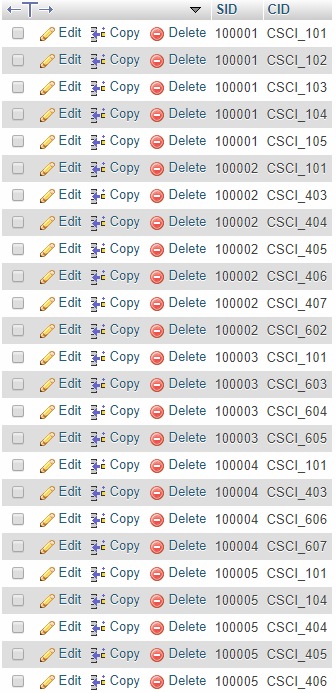
Campus;



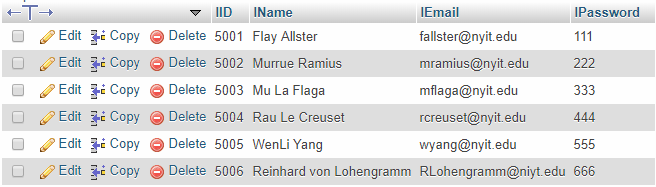
Course:



CourseInfo:



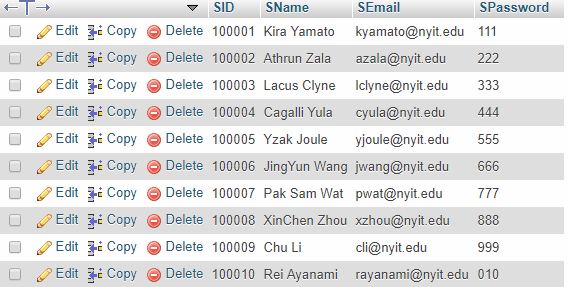
Instructor:



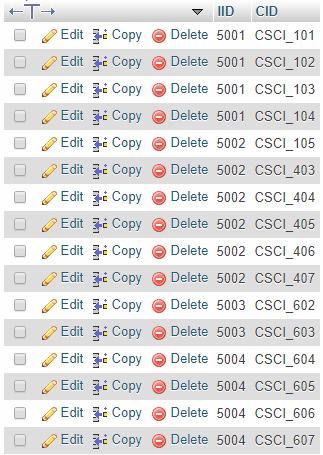
SignSheet:



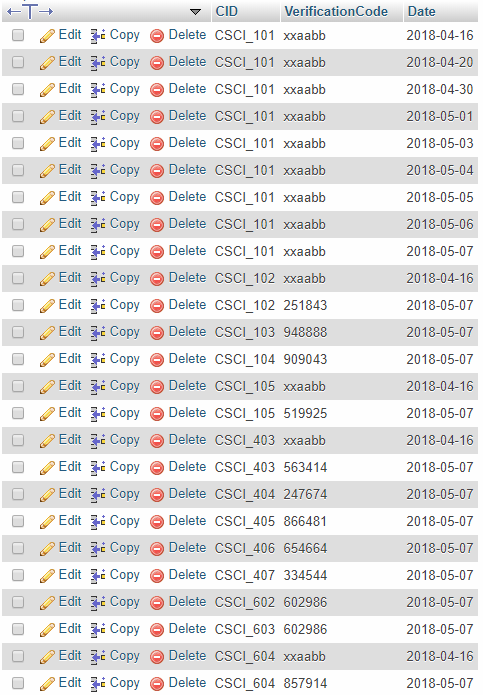
Student:



TeacherInfo

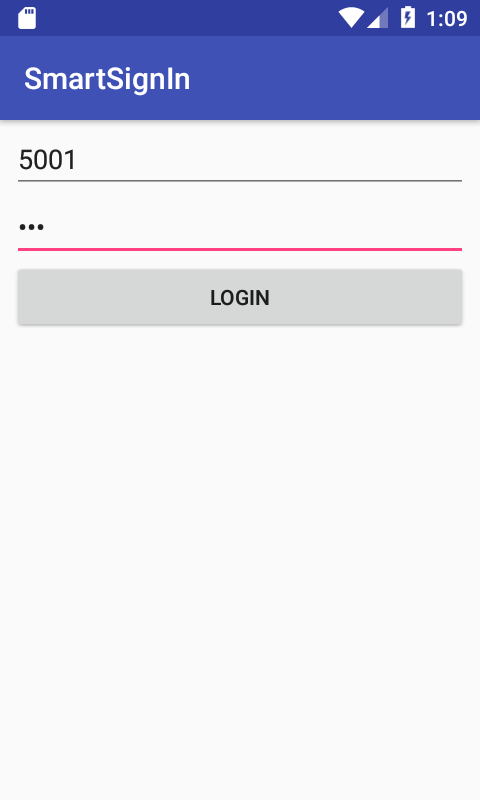
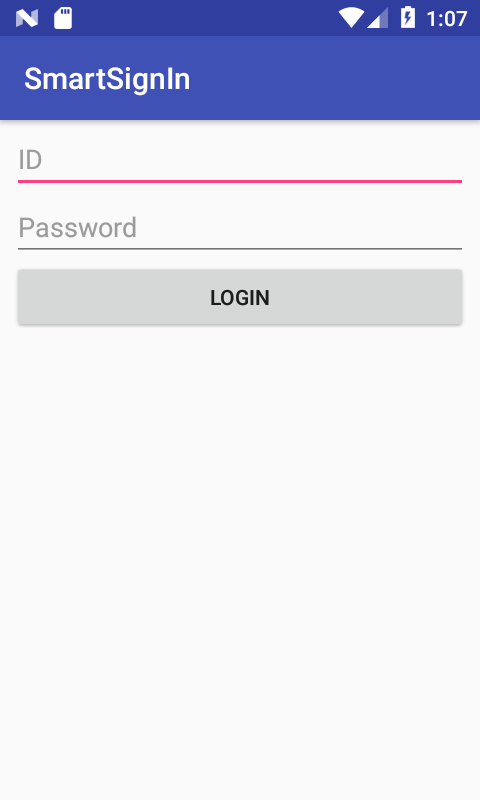


VerificationCode:

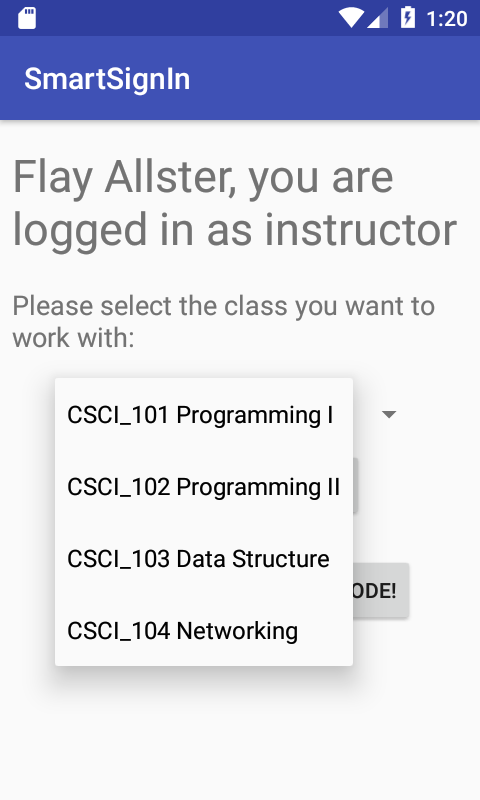
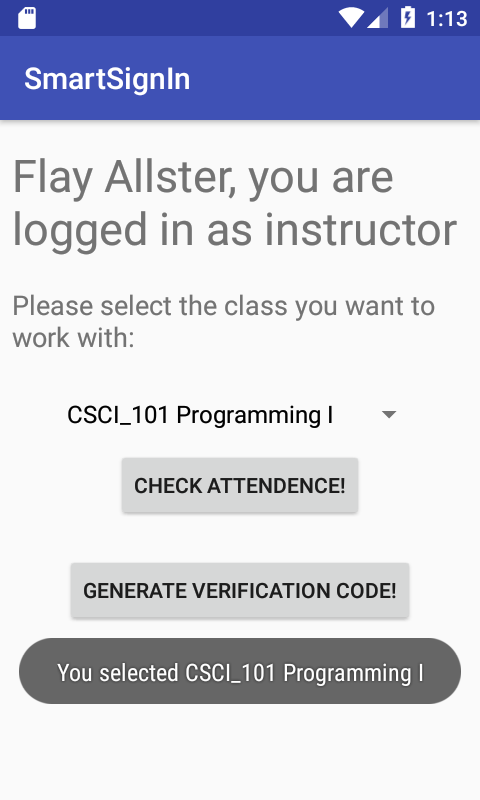


**Manual**

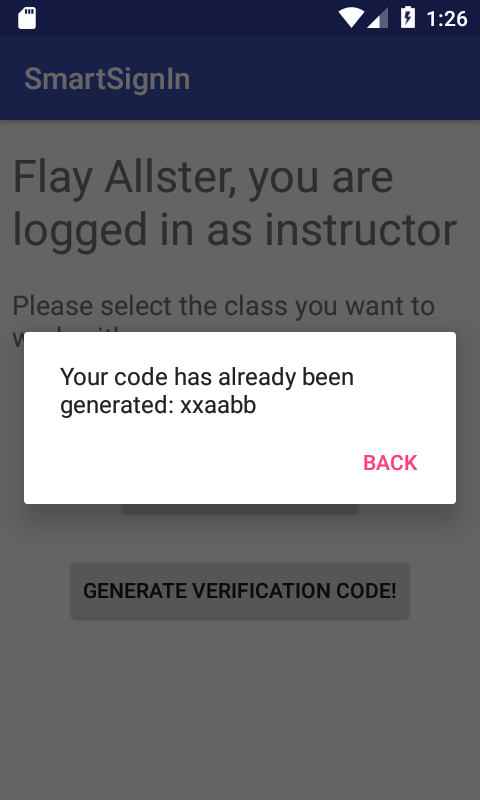
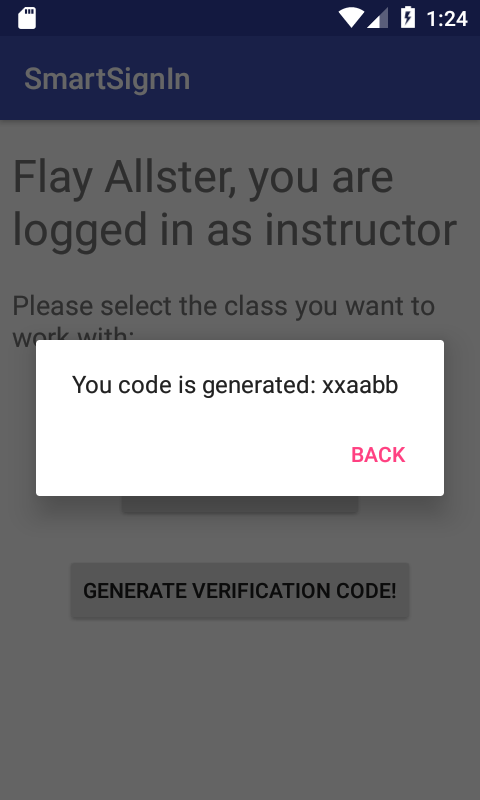
*Sign in as instructor:*



First of all, user would be able to enter his/her ID and password to get access to sign in application.



When user is successfully logged in, a welcome message will be show at the top of the screen. User may allow to select the course which performing function on. A message will pop up for your chosen course.



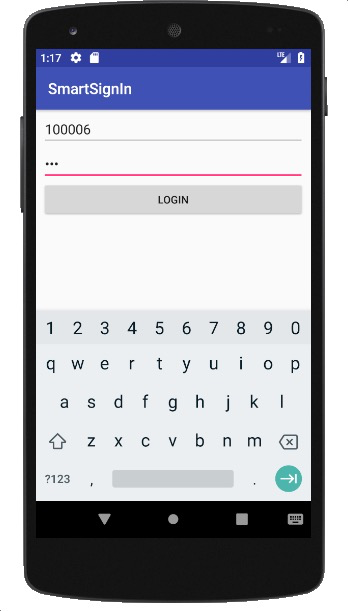
By clicking the “Generate Verification Code!”, a new verification code will be generated and store in database. A reminder message will be appeared. If the verification code has been generated today, application will remind user verification code has been already generated. A notice here is verification of every course would be only generated once per day.



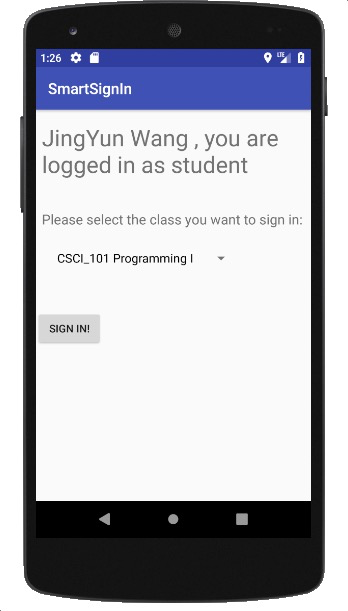
After clicking the “Check Attendance” button, user will be leaded to another page and today’s attendance list will be shown automatically. Student sign in status is shown as On Time, Cheat, Late or Absence by following the student name. User can select and check the attendance list of a particular date

*Sign in as student*

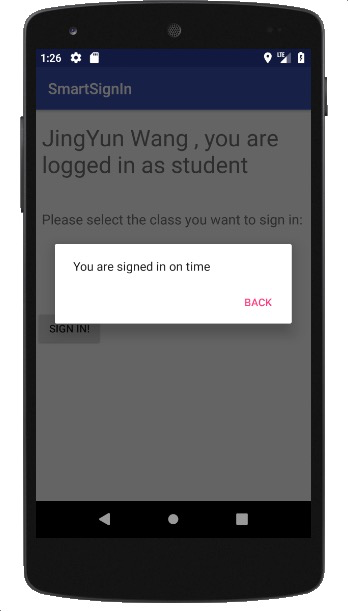
Student Login Page:



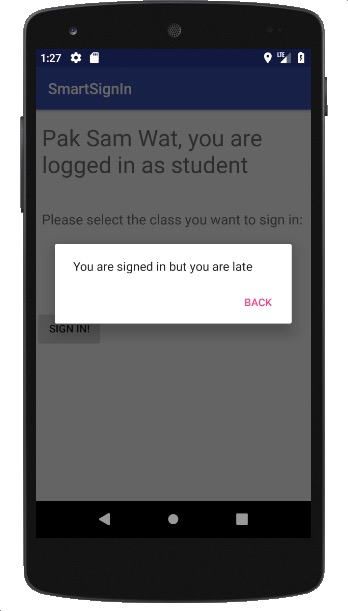
After login successful, user can choose a specific course to sign in.



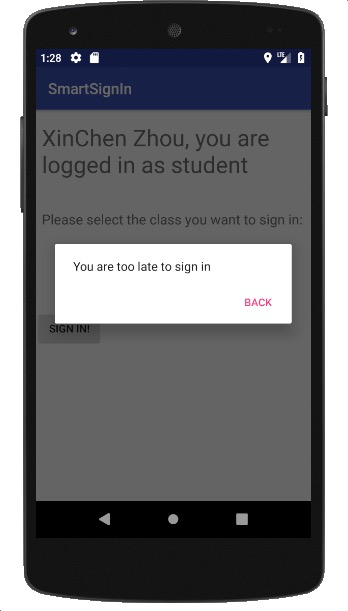
If Student login 10 mins after the class begin, he/she will be login successful.



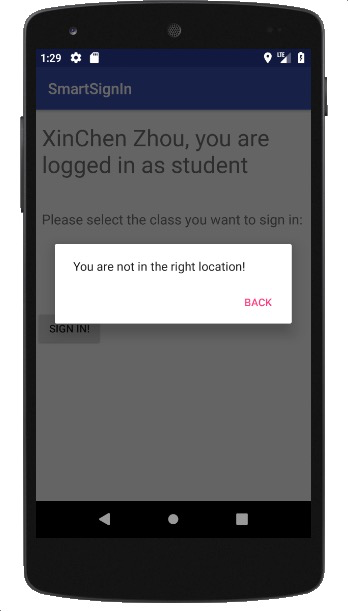
If the student login within 10 mins and 30 mins, she/he can still sign in but it will be late.



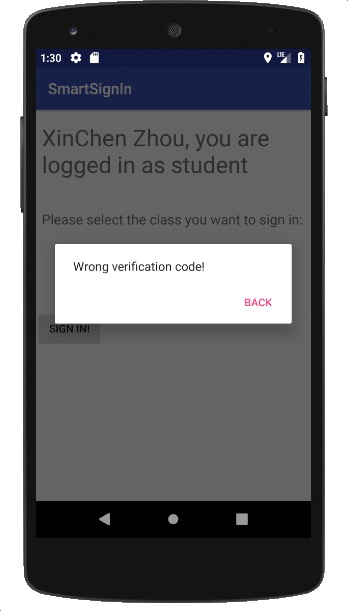
If the student is too late for the class, the student will not be allowed to sign in.



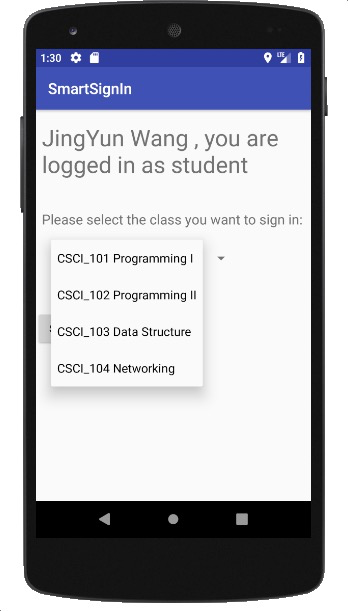
The if the student’s location is not matching the class location, it will show that student is not in the class



If student enter wrong verification code, it will show an alert box to tell user she/he entered wrong verification code.



One student can have multiple courses



**Conclusion and Future works**

Our application minimizes the time of students sign in for their class and provides an efficient way for instructors to check the attendance. Using GPS function and verification code two-way authentication compose with anti-cheat algorithm will ensure the student is in the classroom in most cases.

Of course, there are many things need to be improved. Due to the time limitation, we are assuming that each class only has one section while in real case one class may have two or more sections that given by the same or different instructor. Besides that, our database only contains the classes are given in the current semester. However, the database is growing as time goes on. We need a function in our app in order to display current semester’s classes only for the user. In addition, since it is a sign in application, it is unfair for the students that do not have an Android phone. Maybe we should also develop an iOS version in the future as well.

Although there is still a lot work need to be done, we believe that our application has considerable practical value in real world.

Our project code: https://github.com/xBlacKirAx/LoginRegister

**Reference**

1. <https://www.tutorialspoint.com/android/android_php_mysql.htm>
2. <https://www.youtube.com/watch?v=QxffHgiJ64M&list=PLe60o7ed8E-TztoF2K3y4VdDgT6APZ0ka>
3. <https://www.youtube.com/watch?v=lvcGh2ZgHeA>