
# Computer Science and Engineering

©


## Project Design-Version 2.0

- Team Members


| Names | E-mails |
|---|---|
| Simeng Sun | ss7151@nyu.edu |
| Shantanu Ranjan | Sr3306@nyu.edu |
| Saudamini Srivastava | ss7679@nyu.edu |

## Revision History

| Date | Version | Description of Change |
|---|---|---|
| 3/23/2014 | 1.0 | Initial Design Document |
| 5/19/2014 | 2.0 | Final Design Document |
| | | |
| | | |

# Table of Contents

## 1. Problem Overview

The world is technology-savvy. Every person, however hardworking, secretly wishes that there is a technology that does the work for him. If not the entire work, at least some part of it. This does not mean that the mankind is edging towards lethargy. The reason behind the creation of any kind of technology is to lessen the workload of an individual.

This app aims to lessen the work of individuals, be it professor or a student by taking attendance that could otherwise be tedious and laborious if done manually.

### 1.1 Introduction

TickMark is an attendance collector android mobile application that is useful mainly for academic purposes. It allows collection of attendance in an efficient, quick and easy manner.

Traditionally, attendance is taken either by passing a sheet of paper around the class or calling out names and checking them off the list. All Students have to either sign their names or wait for their names to be called out. This can be very tedious, even time consuming. The time allotted for learning is spent in marking attendance, thereby decreasing efficiency.

Moreover, managing so many attendances for each day and each lecture is also difficult. It is almost cruel to ask a person to manage so many records and keep track of so many files in this age of technology.

TickMark, reduces effort, both, in terms of time and keeping track of attendances. It generates a one-time code for the Students to mark attendance. This way all that the Students have to do is start TickMark on their respective mobile phones, select the class name and mark attendance. The professor too, has to just generate the one-time code each time he wants to take attendance and the result is managed by the app.

## 2. Current Solutions and Related Work

As expected, the android market is filled with apps that are capable of collecting attendance. But that is all that they are capable of- 'collecting attendance'.

When it comes to managing the attendance collected, the efforts of the user are not reduced. For example, *Attendance Roster* lets you keep track of students' attendance but you are required to import the student list from your computer.

Other apps such as *Attendance Tracker* provide you with an interface wherein you have to manually put a check against names that are present and a cross against names that are absent.

But none of these apps provide the functionality of verifying if the Student is indeed present in the class. Also, all other apps can just do the attendance work. What if the Professor wants to do a quick poll? They would have to turn to another app or do it with paper work. TickMark provides a feature that you can take a small quick poll, and the procedure is similar to the taking attendance. The result is quick and clear. This will save a lot of time for both the Student and the Professor.

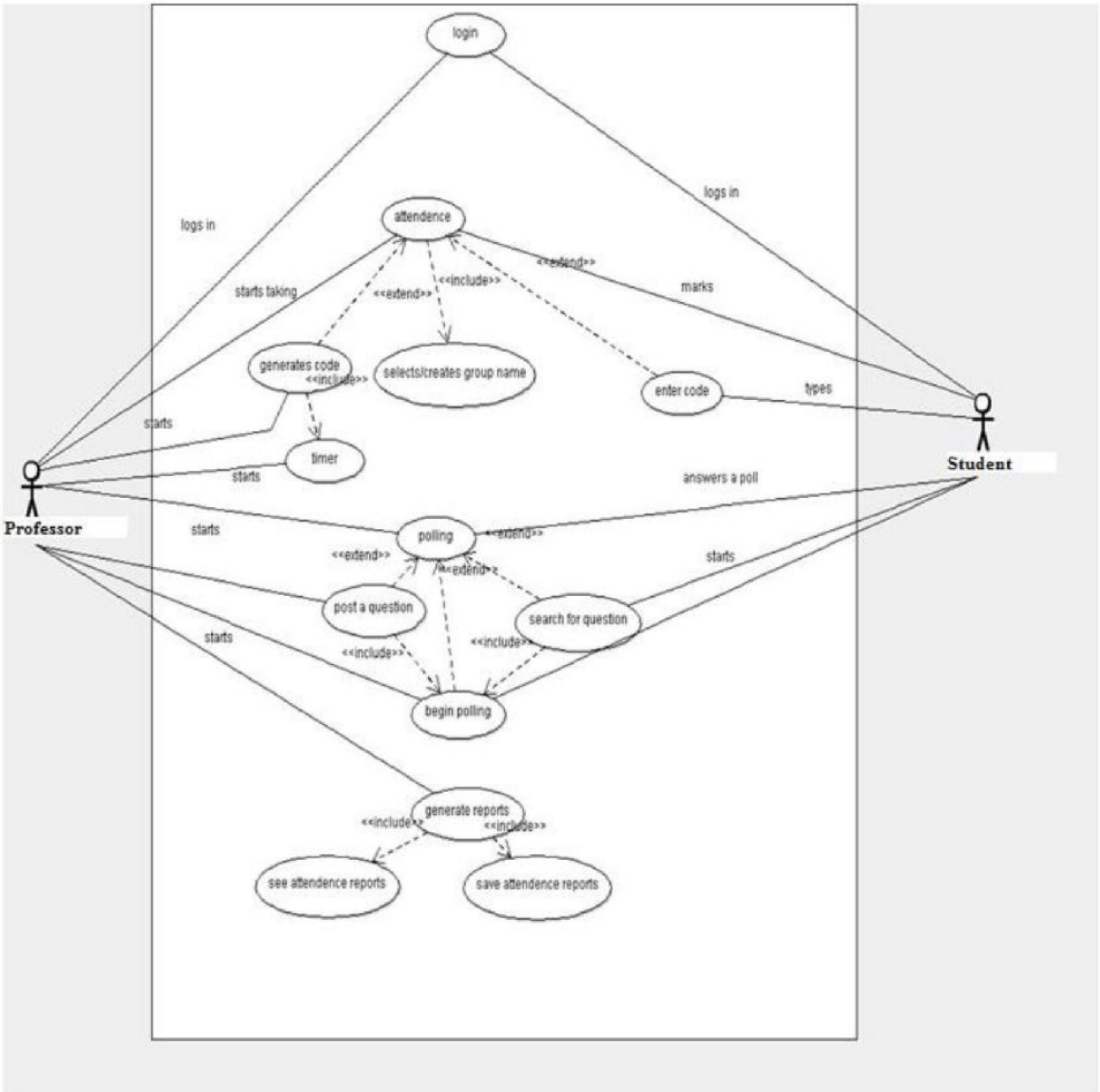
## 3. Solution

The want for an android application that can take attendance while being easy to manage and also efficient led to the idea of TickMark. This mobile application is different from its contemporaries in terms of successfully identifying if an individual should be allowed to mark attendance based on his location.

The following subsections elaborately describe the application's working with the help of various model diagrams.

## 3.1 Use Case Design

The following use case diagram describes the various activities the two main users of the application perform.



## 3.2 MVC Framework

This subsection describes the various components: models, views and controllers of the application.

### 3.2.1 Models:

Professor, Student, Group

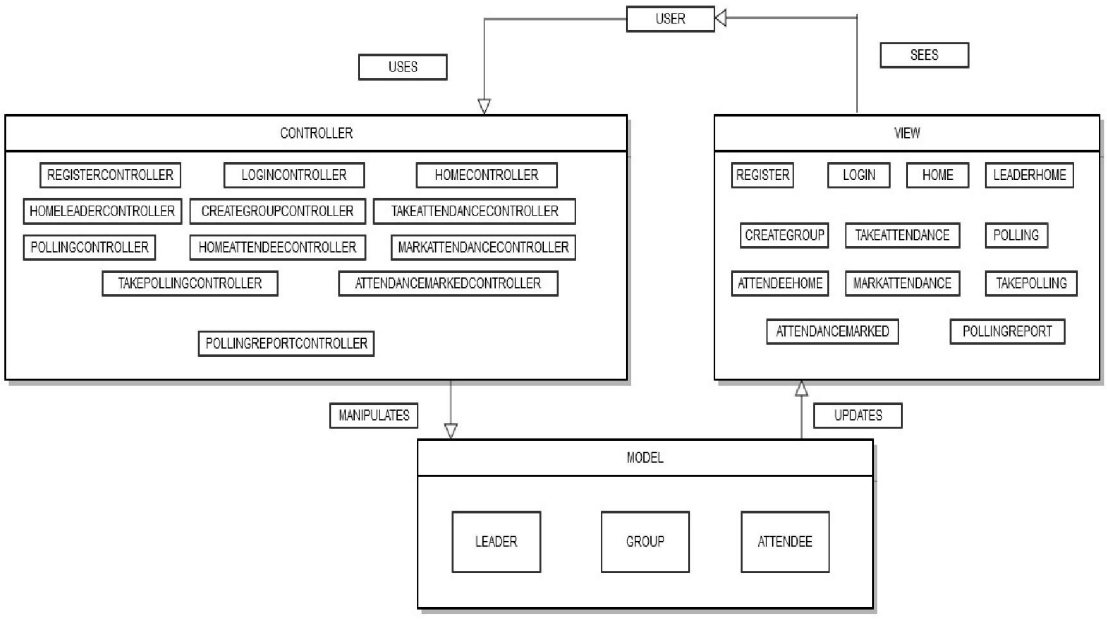
### 3.2.2 Views:

Register, Login, Home
Professor: Home, Create Group, Take Attendance, Polling
Student: Home, MarkAttendance, TakePolling
Report: AttendanceMarked, PollingReport

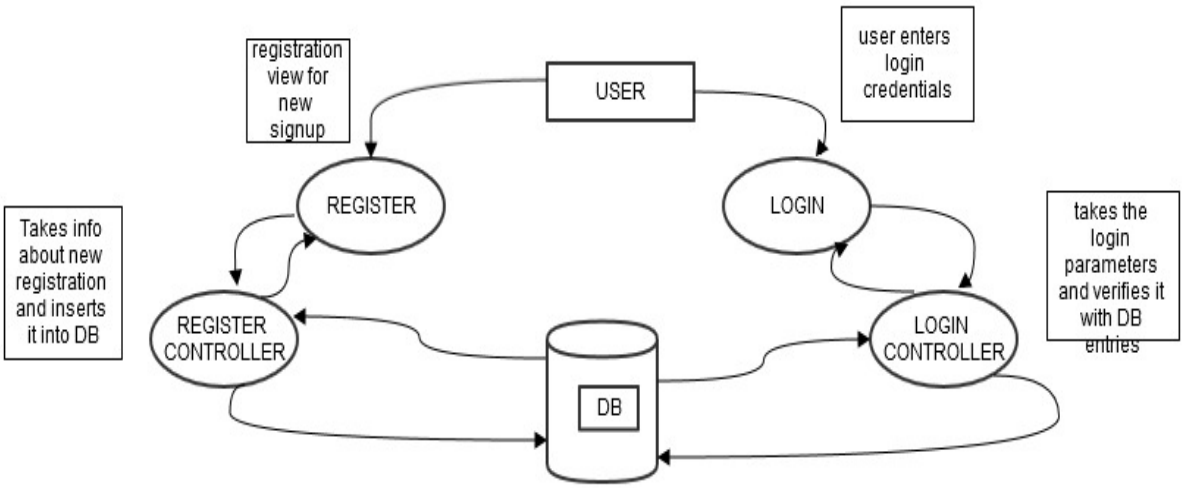
### 3.2.3 Controllers:

RegisterController,LoginController,HomeController
Professor:HomeProfessorController,CreateGroupController,TakeAttendanceController,PollingController.
Student: HomeStudentController,MarkAttendanceController,TakePollingController
Report:AttendanceMarkedController,PollingReportController.

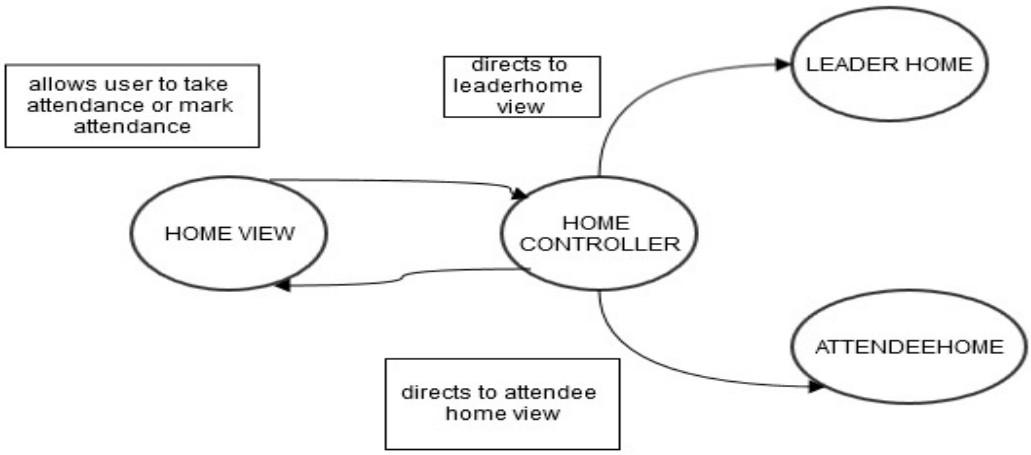
### 3.2.4 System component Architecture Design



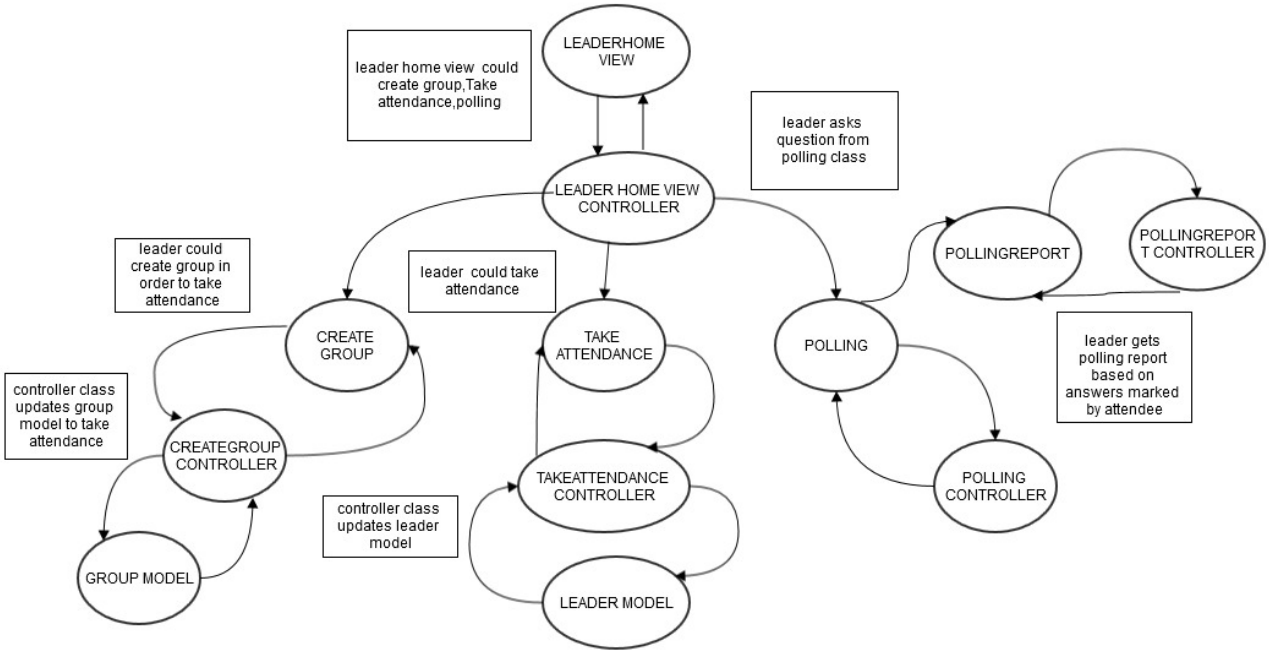
Login and Registration View



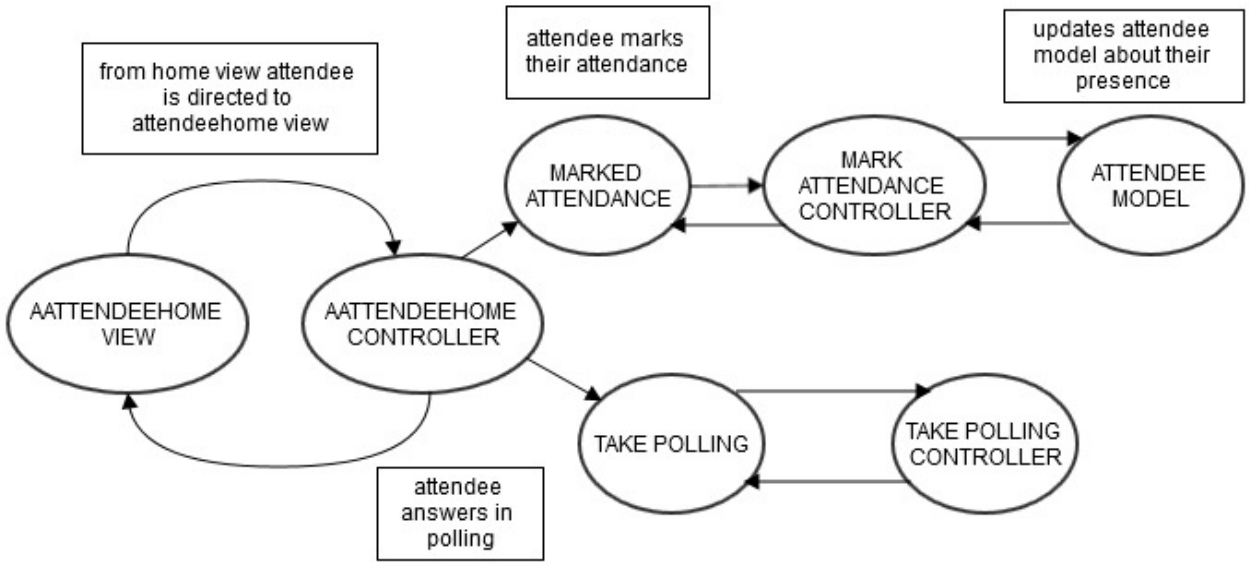

Home View



Professor View

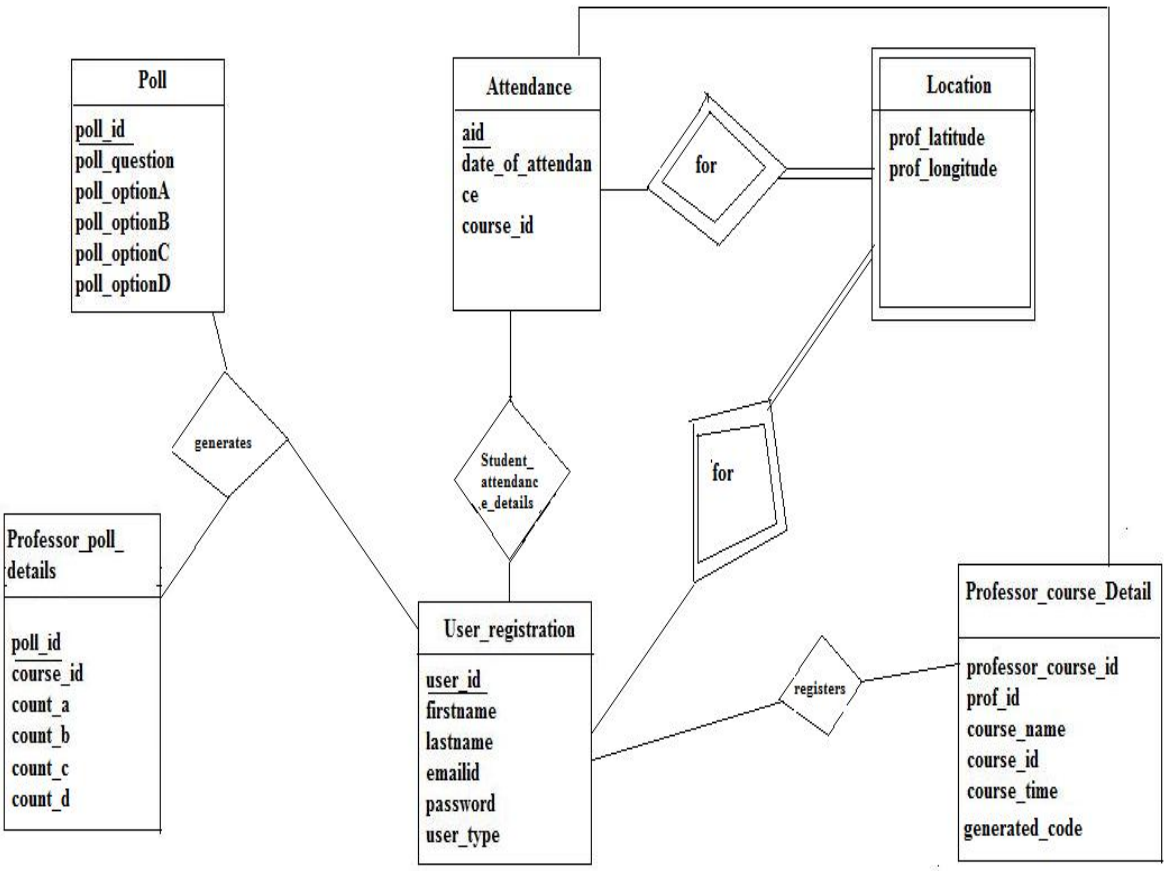



Student View



## 3.3 Database Design

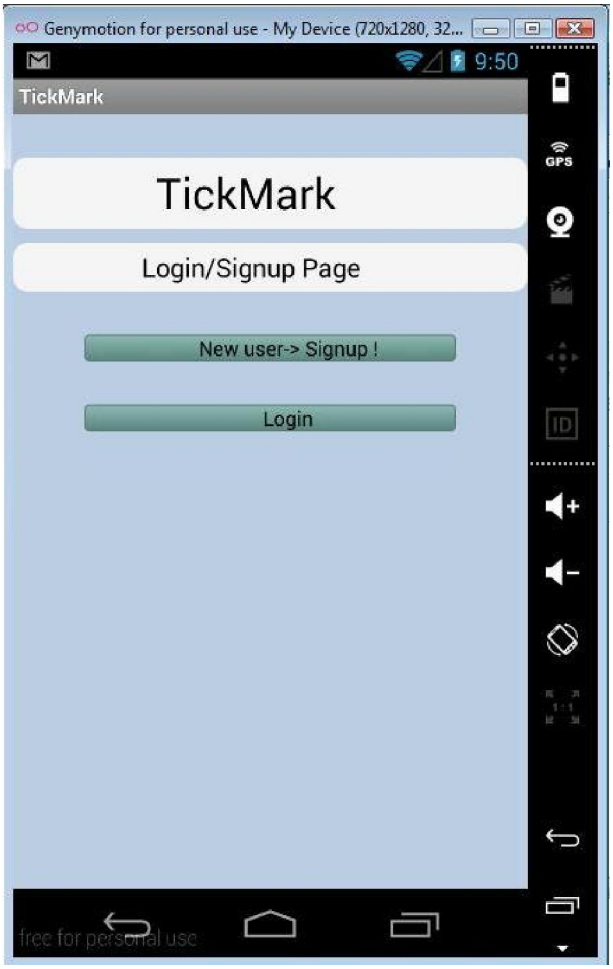
The data that is stored behind follows the following relational design.



## 3.4 UI Screens and Functionalities

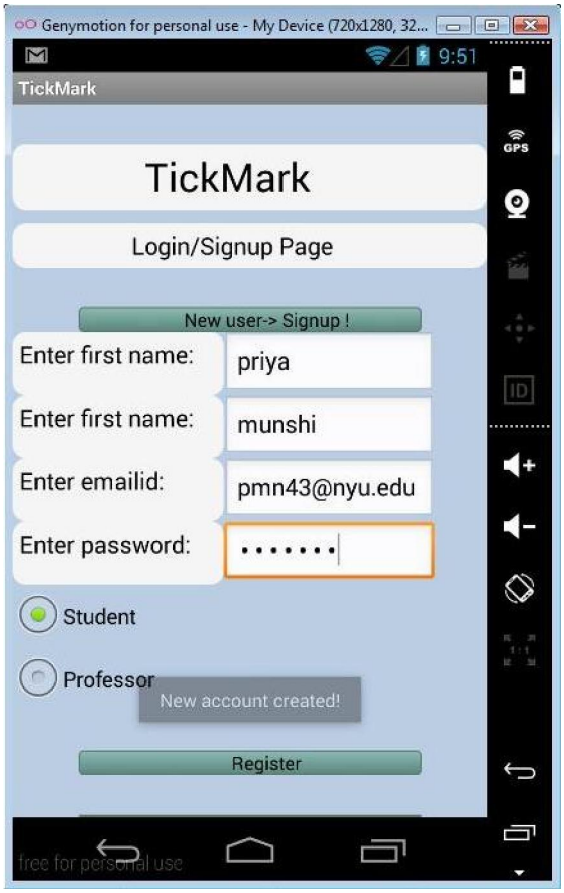
The entire app is divided into functionalities for two main users: Professor and Student and three major events: Take/Mark Attendance, Take/Mark poll and Create courses. Following pages describe each of these events in details:

Sign in and Log in Page:




A user can sign-up if he is new to the app. Or else, he can login with registered credentials.

If the user clicks on the signup button, he is asked to fill in a form for registration.




The email id is used as the identity of the user throughout the app. The user can select to register as a student or as a professor. Under either choice, his credentials would be stored and a check will be made to the database if he tries to access functions that are beyond his usage.

After successful registration, the user can login from the same page by clicking on the login button.


After entering his email id and the correct password, the user presses Enter to login button.

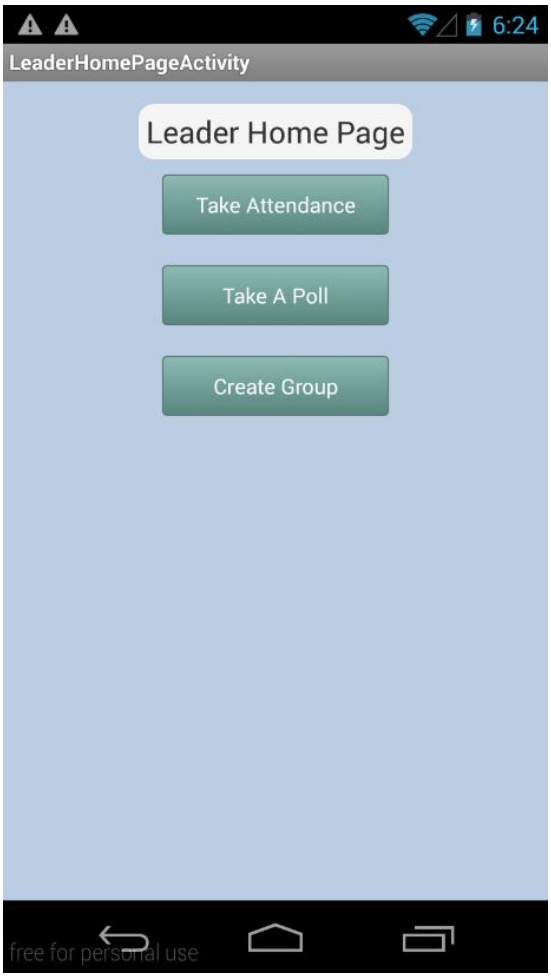
After successful login the user is directed to the homepage:


The app assumes that you are a student by default. On load of this page, two things happen: All the courses that the student is enrolled for are loaded dynamically. Also all the professors who take up the courses are loaded dynamically.
Depending whether user wants to mark attendance or answer a poll, the app navigates accordingly to two different pages.

If user is a professor though, he will click on the “Leader?” button which authenticates against his user id, if indeed he is a professor. If yes, he is navigated to the Professor home page.

If the user clicks on “Leader?” button then he is directed to professor page where he could create his group and could take attendance. The flow of Professor is described below:

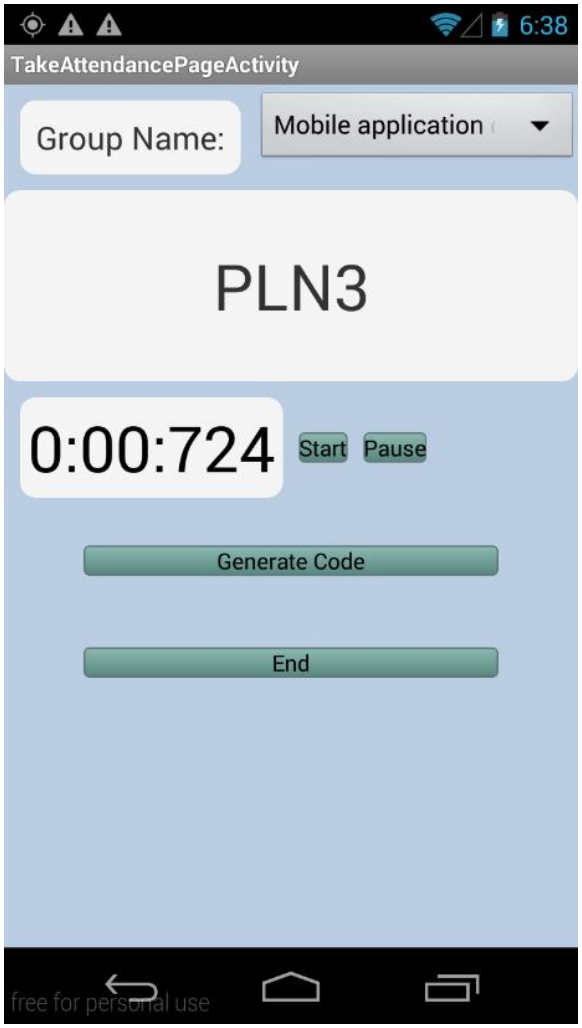
The Professor flow: Homepage




It allows professor to take an attendance, take a poll or create a new group/course.

If professor clicks on Take Attendance button then he is navigated to TakeAttendancePage where he can take attendance for a particular class. If professor clicks on Take Poll button then he is navigated to TakePollPage where he can generate as many polls per class as he wants. If professor clicks on Create Group button then he is navigated to CreateGroupPage where he can create new class groups to take attendance for or to take poll for.

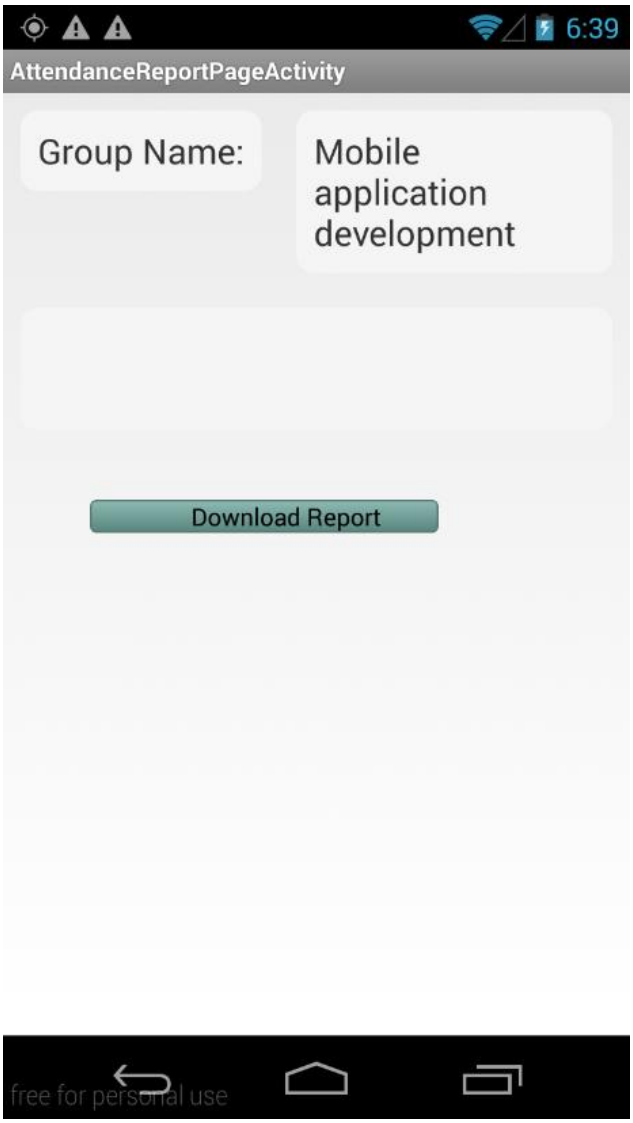
If Professor chooses Take Attendance button then he will be directed to this page.




First, professor selects the course name for which he wants to take attendance from the list of courses loaded. Then, Professor clicks on Generate Code button to display a unique 4 digit code. The code is stored to web server database along with professor's location coordinates to allow students to mark attendance after location matches with that of professor. Here a PHP file connects to web server and inserts/updates the records in the database stored on the Google app engine. Professor could see the attendance report by clicking on End button after all students have marked their attendance.
A special mention has to be made to the unique code generation algorithm that is being used here. This algorithm generates unique code each time it is made a call. It creates a 4 digit different code every time it is called. The class in java that does it is as follows:

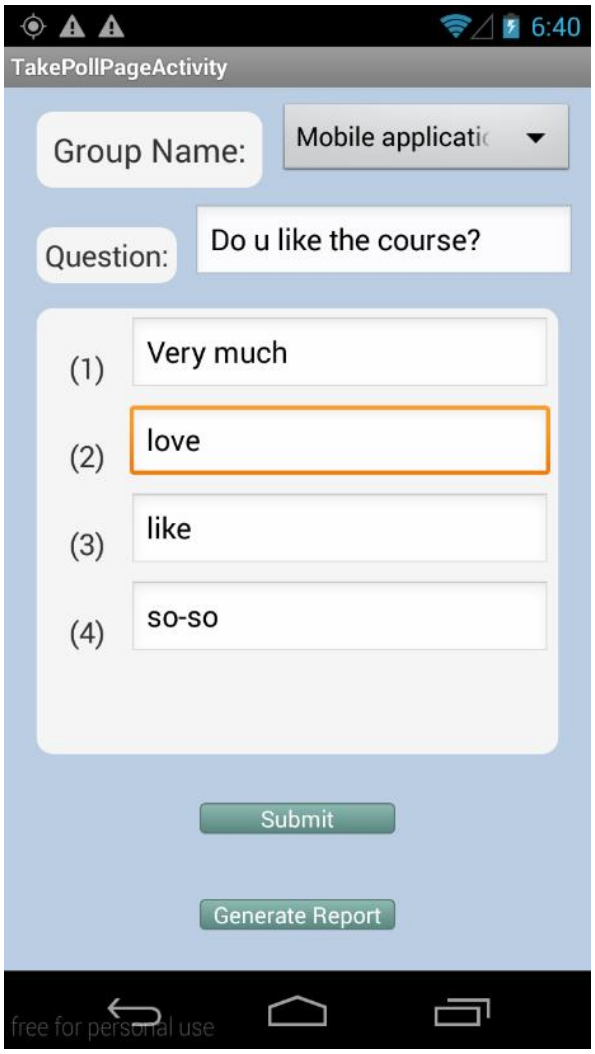
```
public class MyStringRandomGen {

	private static final String CHAR_LIST
="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ12345678
90";
	private static final int RANDOM_STRING_LENGTH = 4;

	/**
	 * This method generates random string
	 * @return
	 */
	public String generateRandomString(){

	    StringBuffer randStr = new StringBuffer();
	    for(int i=0; i<RANDOM_STRING_LENGTH; i++){
	        int number = getRandomNumber();
	        char ch = CHAR_LIST.charAt(number);
	        randStr.append(ch);
	    }
	    return randStr.toString();
	}

	/**
	 * This method generates random numbers
	 * @return int
	 */
	private int getRandomNumber() {
	    int randomInt = 0;
	    Random randomGenerator = new Random();
	    randomInt =
randomGenerator.nextInt(CHAR_LIST.length());
	    if (randomInt - 1 == -1) {
	        return randomInt;
	    } else {
	        return randomInt - 1;
	    }
	}
  }
```

After attendance has been marked, if the professor clicks on end button then he is directed to the report page.



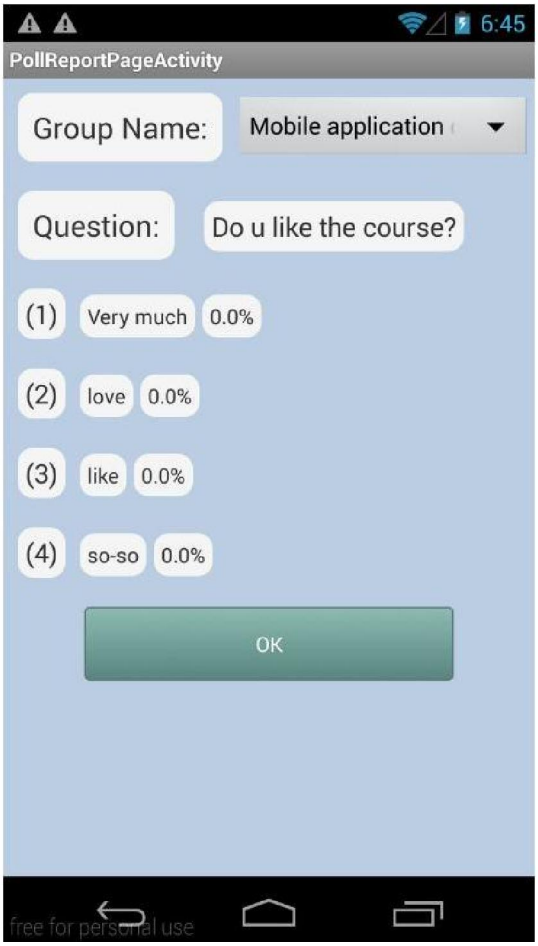

Once the attendance has been marked by the students/Student then a report could be generated in a file format and the Professor would be allowed to save it on his device or in database.

If the Professor chooses to take a poll then he could either take feedback or could ask questions based on the lecture/presentation. He would then be redirected to this page:



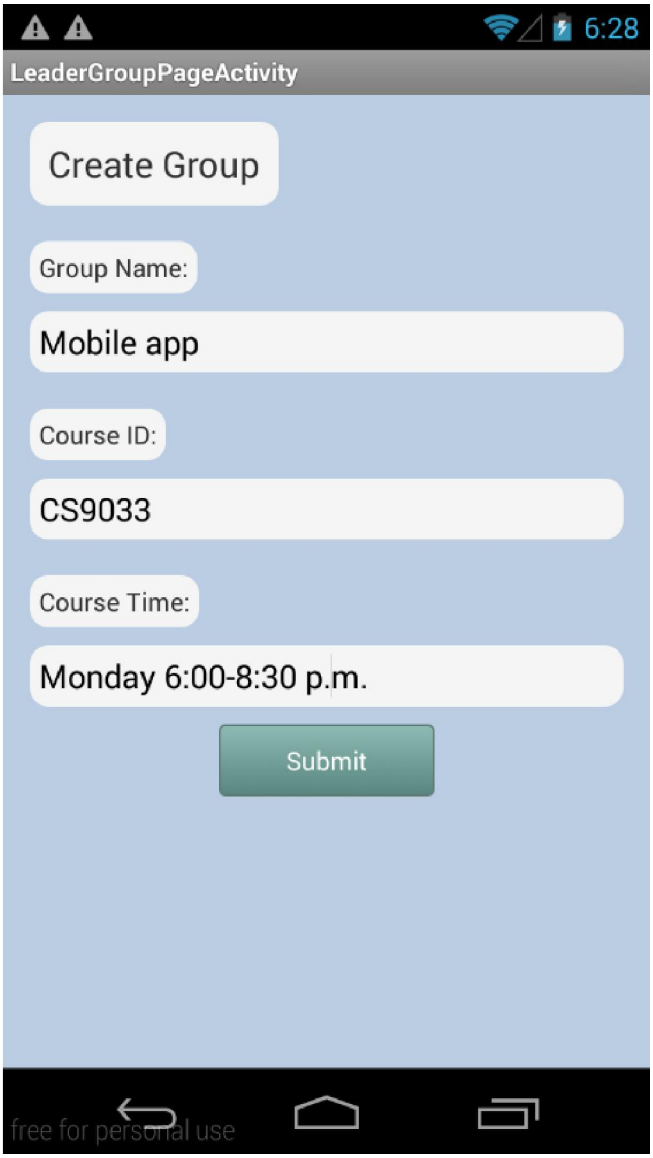

Here the professor selects the course name from drop down list for which he wants to create a poll. Then, he sets the poll question along with the options. After he has done that, he clicks on submit button which stores the generated poll on the web server. Once all the students have taken the poll, the professor could click on generate report button to see who all have taken poll and what are the results.

The Report Page shows the following details:



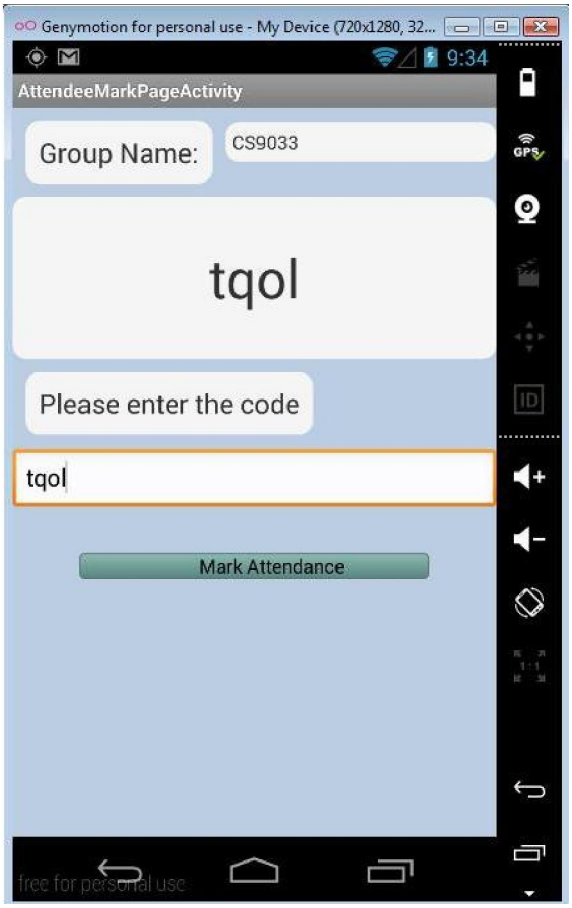

Once the students have taken poll, their percentage would be displayed. A point to be noted is that the polls are anonymous. It does not store who marked what.

If the Professor chooses to create a new group/class, he will click on the create group button. He would then be redirected to a page on the app where he will be allowed to create a new group by entering a few details.




Here professor enters group name,course Id,course time and clicks on submit button. The data gets stored to the Google app engine database via php scripts.

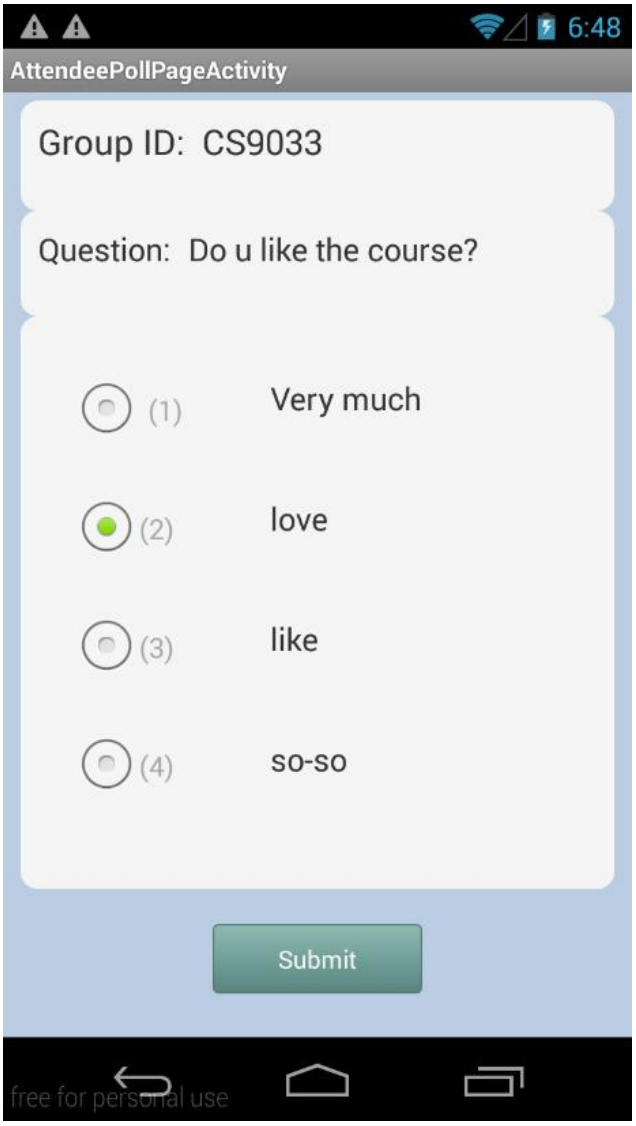
The functions for a Student are as shown next. If the user is indeed a student, he can either mark his attendance or take a poll.
If he decides to mark attendance, he is directed to the mark attendance page.




The code shown is the code that the professor generated for the course recently.
The student is asked to enter the code and successfully mark his attendance. Upon clicking the mark attendance button, an entry is made in the database at the Google app engine server for the corresponding student against this attendance id.

Note that this happens if the GPS coordinates of the student are within a certain range of that of the professor. All this happens in the background.

If the student has to answer a poll, he will be directed to a page that allows him to take polls.




The student answers the poll for the respective course and clicks on submit button. The answer gets stored on database.

## 3.5. Server code snippets

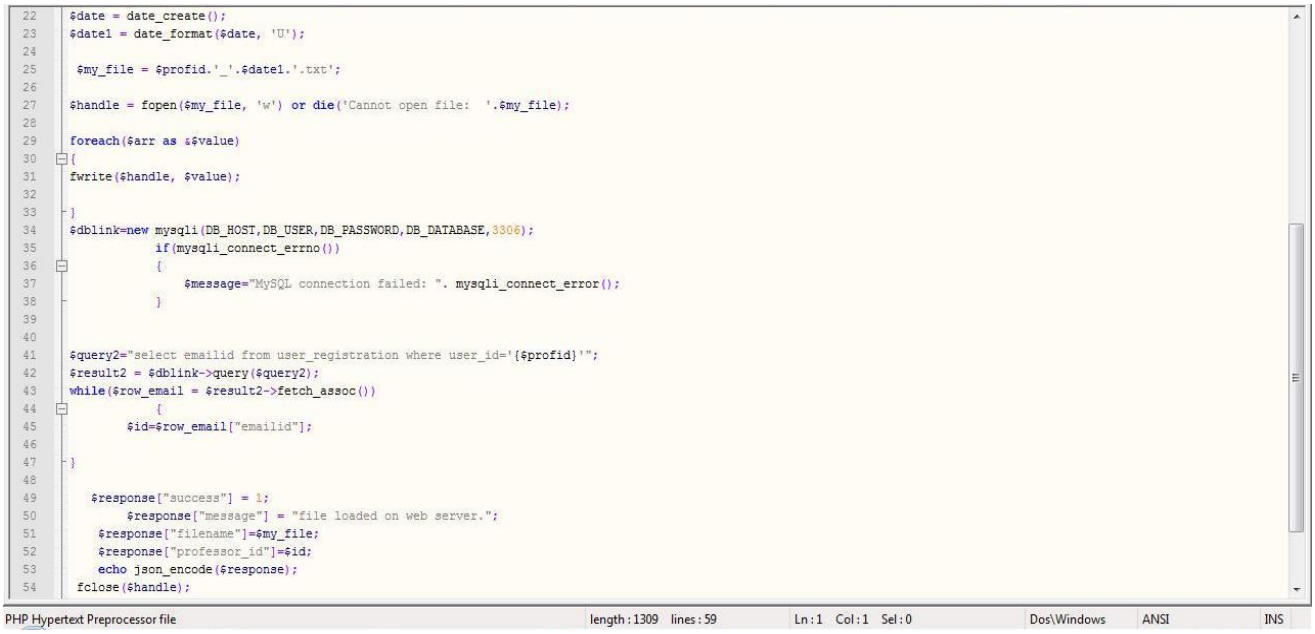
The following are some snippets from important php files that are residing at the web server and performing certain functionalities.

1. To check location of a student, the app fetches the location of the professor that is stored in the database. The following code does this.

```
$row = $result->fetch_assoc();
$aid = $row["aid"];

$course1["aid"] = $aid ;

$query2="SELECT `prof_latitude`, `prof_longitude` FROM `location` WHERE `aid` = '$aid' and `prof_id` ='$pid'";
$result1 = $dblink->query($query2);
if($result1)
{
if((mysqli_num_rows($result1)) > 0)
{
$row1 = $result1->fetch_assoc();
$lat = $row1["prof_latitude"];
$lon = $row1["prof_longitude"];

    $course1["lat"] = $lat;
    $course1["lon"] = $lon;
    }
}
}
$course1["success"] = 1;
echo json_encode($course1);
}
}
else
{

    $course1["success"] = 0;
    $course1["error"] = "Required field(s) is missing";


    echo json_encode($course1);
}
```

PHP Hypertext Preprocessor file | length : 1718 lines : 79 | Ln : 34 Col : 16 Sel : 0 | UNIX | ANSI | INS

2. To generate reports, the app pulls a file from the server.



```php
$date = date_create();
$date1 = date_format($date, 'U');

 $my_file = $profid.'_'.$date1.'.txt';

$handle = fopen($my_file, 'w') or die('Cannot open file:  '.$my_file);

foreach($arr as &$value)
{
fwrite($handle, $value);

}
$dblink=new mysqli(DB_HOST,DB_USER,DB_PASSWORD,DB_DATABASE,3306);
            if(mysqli_connect_errno())
            {
                $message="MySQL connection failed: ". mysqli_connect_error();
            }


$query2="select emailid from user_registration where user_id='{$profid}'";
$result2 = $dblink->query($query2);
while($row_email = $result2->fetch_assoc())
            {
        $id=$row_email["emailid"];

}

   $response["success"] = 1;
        $response["message"] = "file loaded on web server.";
    $response["filename"]=$my_file;
    $response["professor_id"]=$id;
    echo json_encode($response);
fclose($handle);
```

PHP Hypertext Preprocessor file | length : 1309 lines : 59 | Ln : 1 Col : 1 Sel : 0 | Dos\Windows | ANSI | INS

### 3.6 Android Components

The application will use the following android components:

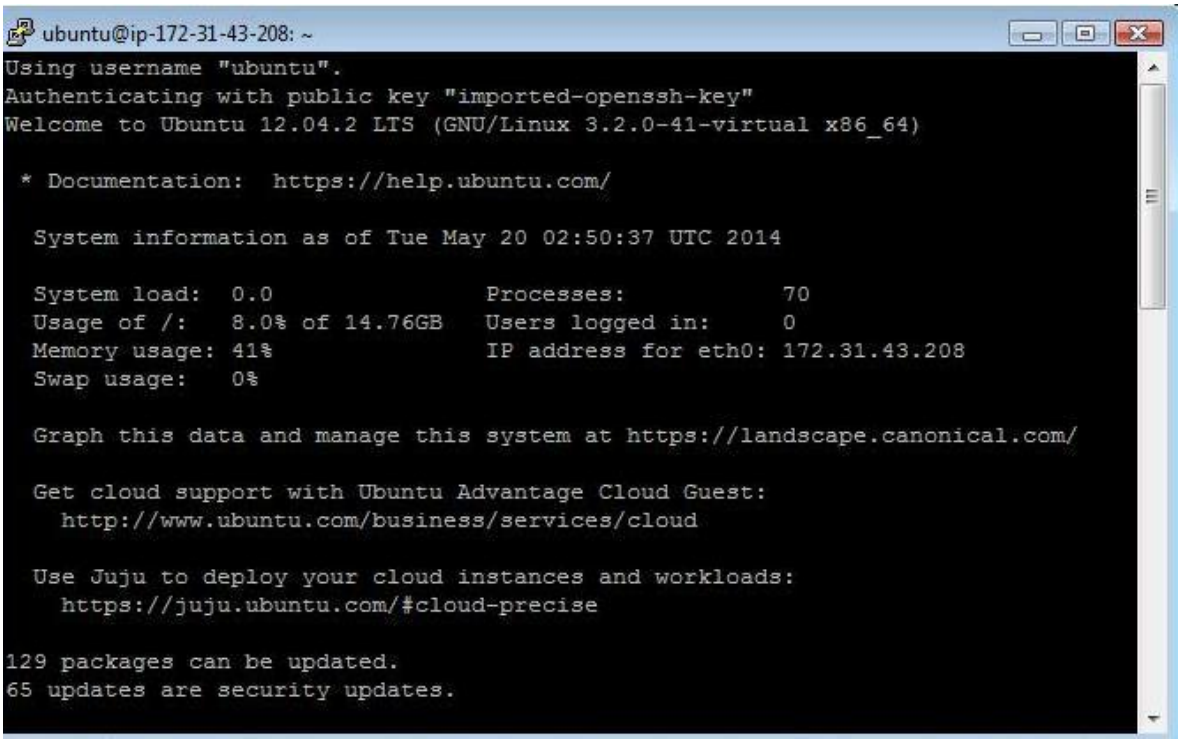
1. Wifi - Internet: In order to login and take/mark attendance, the app will require connection to Internet or wifi.

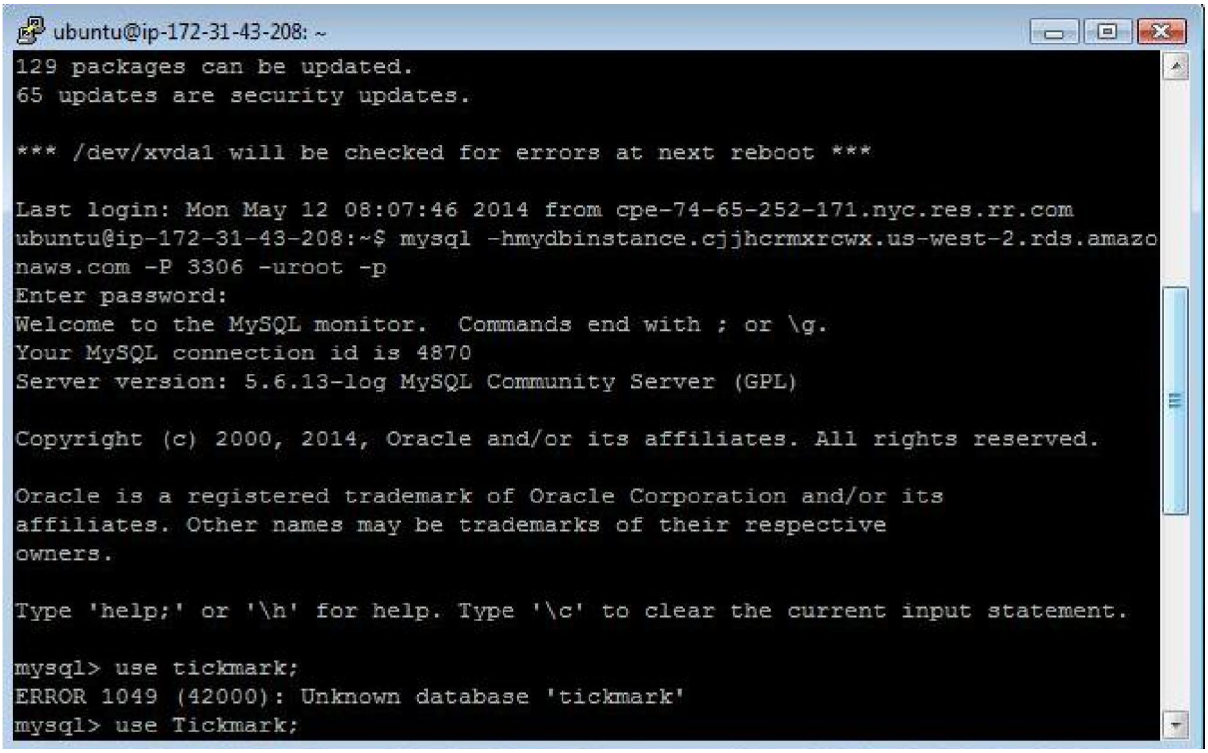
2. GPS: The app requires location based services of a smartphone.

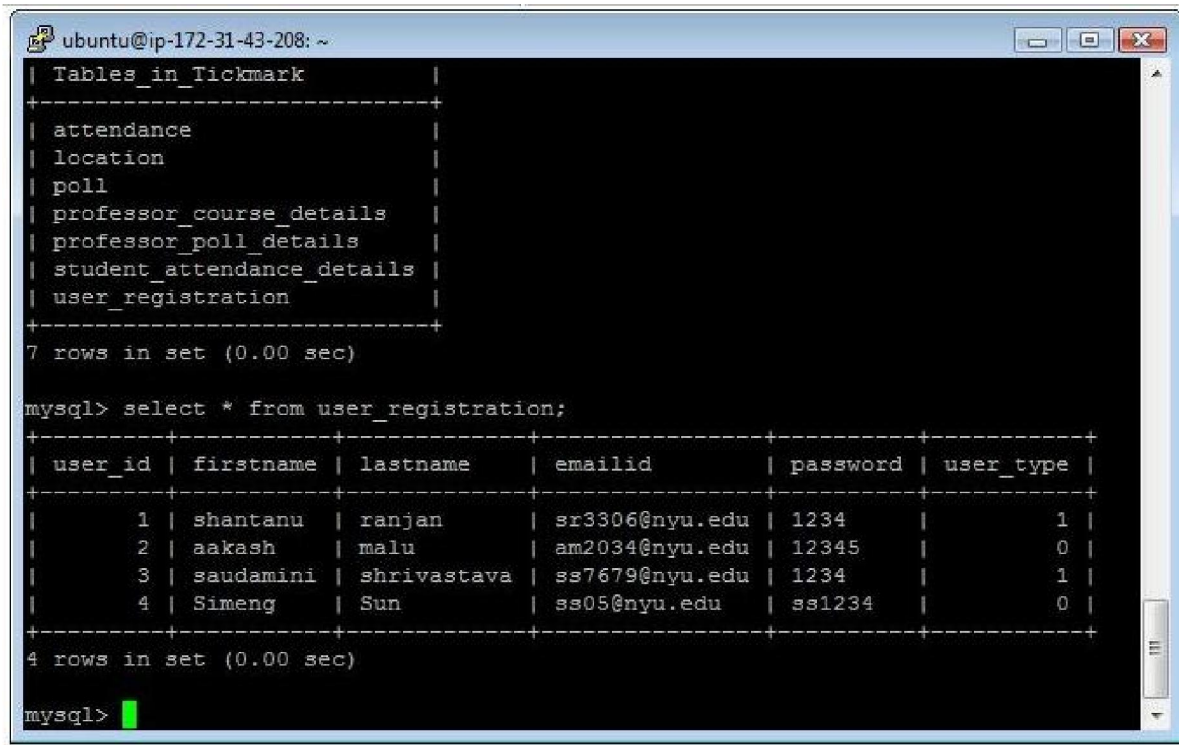
3. Amazon web service/Google App Engine: The app requires a backend web server and android web services provide a free web server.

   For working on the Google app engine:
   First we registered an app ID which should be unique with the others' apps so we could reach it with internet. Within that web app, we built the cloud SQL instance, and connected it to our MySQL client on the local machine. Then we built the whole database with it. We also had to set the yaml file to make sure it contains all php files. At the end of it, we just used the Google app engine client to deploy the app.

   For working on amazon web services:
   First we had to create an account on AWS. Then next we had to create an instance of EC2 web server(possibly a linux image)and create a relational database on that EC2 server. Once the connections were established, we could talk to it using php scripts deployed on the server.


ubuntu@ip-172-31-43-208: ~
Using username "ubuntu".
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 12.04.2 LTS (GNU/Linux 3.2.0-41-virtual x86_64)
* Documentation: https://help.ubuntu.com/
System information as of Tue May 20 02:50:37 UTC 2014
System load: 0.0 Processes: 70
Usage of /: 8.0% of 14.76GB Users logged in: 0
Memory usage: 41% IP address for eth0: 172.31.43.208
Swap usage: 0%
Graph this data and manage this system at https://landscape.canonical.com/
Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
Use Juju to deploy your cloud instances and workloads:
https://juju.ubuntu.com/#cloud-precise
129 packages can be updated.
65 updates are security updates.


ubuntu@ip-172-31-43-208: ~
129 packages can be updated.
65 updates are security updates.
*** /dev/xvda1 will be checked for errors at next reboot ***
Last login: Mon May 12 08:07:46 2014 from cpe-74-65-252-171.nyc.res.rr.com
ubuntu@ip-172-31-43-208:~$ mysql -hmydbinstance.cjjhcrmxrcwx.us-west-2.rds.amazo
naws.com -P 3306 -uroot -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 4870
Server version: 5.6.13-log MySQL Community Server (GPL)
Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> use tickmark;
ERROR 1049 (42000): Unknown database 'tickmark'
mysql> use Tickmark;




4. SQLite: Not all data would be sent to the server. Local data would be stored using android's local database.

## 4. Project Member Breakdown

There are three project members. Each of them will be contributing equally towards the development of the application. Each member has been assigned few tasks.

1. Saudamini Shrivastava

- Gathering of project requirements
- Server and database setup on AWS(amazon web services)
- Development of login and signup modules of the app(both UI as well as functionality)
- Development of homepage and student attendance modules of the app(both UI as well as functionality)
- Implemented location based feature in the app.
- Worked on creating php files for server related functionalities.

2. Simeng Sun

- Server and database setup on Google app engine.
- Development of creating new class module of the app(both UI as well as functionality)
- Development of student poll module of the app(both UI as well as functionality)
- Provided creative inputs to improve security of the app.
- Performed integration testing and debugging of all modules.
- Worked on creating php files for server related functionalities.

3. Shantanu Ranjan

- Development of Professor homepage, take attendance and take poll modules of the app(both UI as well as functionality)
- Development of report modules for attendance as poll of the app(both UI and fucntionality)
- Implemented unique code generation algorithm.
- Worked on creating php files for server related functionalities.
- Created Scripts for database tables.

## 5. Milestones

| Date | Task completed |
|---|---|
| **3/23/2014** | Initial Design Document Due |
| **4/4/2014** | • Completion of UI and all activities<br>• Completion of flow of application |

| | | |
|---|---|---|
| Week 1 | 3/28/2014 | Creating activities for all models |
| Week 2 | 4/4/2014 | Connecting all activities in a basic flow |

| | |
|---|---|
| **4/7/2014** | Initial prototype presentation |
| **4/28/2014** | Presentation at AT&T |

| | | |
|---|---|---|
| Week 1 | 4/12/2014 | Web server connection setup |
| Week 2 | 4/19/2014 | Database connection setup |
| Week 3 | 4/26/2014 | Completion of application |

| | |
|---|---|
| **5/12/2014** | Final Demo |

| | | |
|---|---|---|
| Week 1 | 5/12/2014 | Implementation of new ideas and Testing |

## 6. Future work

Any project is incomplete without identifying the kind of enhancements that can be made to it in order to better it. Without any kind of improvement there is no growth.
After careful consideration and based on the feedback received from the various people who have used the app, we have decided that the following features could be improved and enhanced or included in order to better the experience of using TickMark.

- Improving GUI
  We understand that the first impression can be the last impression. Thus, the interface to an app is, if not the most, but an equally important feature in comparison to its functionality. We have decided to improve the GUI so that it looks more appealing and is fun to use for a professor as well as a student. Using small fragments, creating a tab-style layout for different functionalities is the main goal looking forward.

- Mailing of attendance and poll reports directly to the professor's email account
  Currently the app allows the professor to see the report generated after an attendance and once he has done that, it gets stored on the web server hosting the app. However, there was no option of dismissing the report and looking at it later. Hence we have decided we would to include a feature wherein the reports generated are mailed directly to the professor's email with which he has signed in. This feature would not mandate for the professor to look at the report then and there. He can see it as and when he wants and as many times he wants. Moreover, there will be two copies of the report.

- Customization of the student locator functionality
  The app allows only those students to mark their attendance who are in a fixed range or radius of the professor. But what if the professor is conducting lectures in a big auditorium instead of a small classroom? Hence, the feature that we would be working on next would be to allow the professor to set the range within which he expects his students to be. Thus, there will be a default range which can be changed based upon the requirements of a particular instance.

- A notifications feature
  This will be like the notifications feature of blackboard. The professor can post important announcements and notifications such as "Homework 4 due tonight" or "Final demo grades are up!!" for a particular course and class and it will be broadcast to all students who have registered for that class.

## 7. Bibliography


1. https://play.google.com/store/apps/details?id=com.flapsapps.roster
2. https://play.google.com/store/apps/details?id=peterman.apps.attendance
3. AWS services documentation.
4. Google App engine documentation.