Teacher's SideKick: A Mobile Classroom Management Application using Android and SMS Technology

By

Braga, Arsenio III

Tagpuno, Riki Jonas

SCHOOL OF ARTS AND SCIENCES

ATENEO DE DAVAO UNIVERSITY

MARCH 2016

Teacher's SideKick: A Mobile Classroom Management Application using Android and SMS Technology

An Independent Study

Presented to

The Faculty of the Computer Studies Cluster

Ateneo de Davao University

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Information Technology

By

Braga, Arsenio III

Tagpuno, Riki Jonas

SCHOOL OF ARTS AND SCIENCES

ATENEO DE DAVAO UNIVERSITY

MARCH 2016

ACKNOWLEDGMENTS

This project would not have been possible without the help of our advisor Ma'am Ma. Teresa Te. Quindoy and our panelists Mr. Rey Aliño and Mr. Paolo Villanueva. The researchers would like to thank every teacher who spared a little bit of their time to test the application. We would also like to thank Ashrina Domingo, Raymond Trespeces, Kaytelle Lasaca, Dan Bongas, Dennis Pedroso, Rhon De Guzman, and Kenneth James Uy for helping us accomplish the application by means of suggesting how to solve a problem and suggesting teachers who can test the application. We would also give thanks to our classmates who pushed us hard to achieve our goal. Lastly, we would like to acknowledge our individual families for supporting us all throughout this semester, this wouldn't be possible without them.

Table of Contents

Teacher's SideKick: A Mobile Classroom Management Application using Android and SMS Technology

ARSENIO BRAGA III, Ateneo de Davao University

RIKI JONAS TAGPUNO, Ateneo de Davao University

Abstract

The project plans to create a classroom management application using Android in which it will decrease the workload of the teacher in doing paper works, managing classes is a difficult task to do and it is time-consuming. The proposed application will allow the teacher to save time, papers, and will be handy in managing all of his or her classes. The project is a Classroom Management application using Android OS. It contains features like assigning seat plans, checking of attendance, adding grades and uses SMS technology. This is beneficial for the teachers because automating the work involved in classroom management such as recording attendance will lessen the time and will save the user from additional paper works too.

General Terms: Classroom management, Attendance monitoring, android, Integrated Development Environment (IDE), database

Additional Key Words and Phrases: recording attendance, grades, seat plan

1. INTRODUCTION

1.1 Background of the Study

Over the years, manual attendance monitoring is being used in almost all academic institutions. Attendance monitoring is a significant part of the school's system. Most of the institutions today still use the traditional way of attendance monitoring and not only that this is time consuming but is also prone to errors. Attendance should be properly monitored because it can also affect the performance or rating of the student. Attendance monitoring is not the only concern of the teachers but also grade management. Now, this is okay if the teacher manages only one class but what if the teacher manages more than one class, this would add to the already busy and heavy workload of the teacher.

The researchers would want to implement a Classroom Management Application that would help ease the workload of the teachers by automating attendance monitoring, grade management, and informing the students of their status with the help of SMS technology. The proposed application will let the teacher input a class to which he/she can then import or add students for that class on the seat plan. The Classroom Management Application will also be used to check the attendance of the students and will also generate reports via SMS in order for the student to know about his/her grades and/or number of absences.

As of today, there are no current Classroom Management applications that are stable and/ or organized in Android. Current applications for example, Teacher Kit in Android, is not complete. There is no gradebook and the application doesn't inform students who are nearing the allowable number of absences.

1.2 Problem Statement

The main problem of the study is to automate the classroom management work such as attendance monitoring, grade monitoring, and informing students of their status.

The specific problems of the study are as follows:

(1) What are the needed information regarding classroom management?

(2) How to incorporate SMS Technology with Android?

(3) How to evaluate the proposed application?

1.3 Objectives

The main objective of the study is to implement the automation of classroom management work with a Classroom Management application.

The specific objectives of the study are as follows:

1 Identify the information needed for attendance monitoring.

2 Implement the SmsManager class in Android.

3 Set the needed evaluation process to the proposed application.

1.4 Significance of the Study

The proposed research will help the teachers through the use of the application. The application will improve classroom management by automating the work and would be more efficient than the traditional or manual classroom management. The proposed research will be implemented in an Android mobile-based application because most of the teachers are now using tablets running in Android and will be easily accessible for them. The proposed research will also save paper as it would automate the attendance and grade monitoring.

1.5 Scope and Limitations

The application is limited to classroom management work such as attendance monitoring, recording of grades, auto calculation of the student's record, and informing the student of his/her status if necessary. The application is limited to 40 students for the Lecture seat plan and 42 students for the Laboratory seat plan, once the seat plan has been saved it cannot be updated the user needs to reassign the seat plan. The application only informs the student on the number of absences the student has incurred for a specific class. It is only limited to importing students for a class and be able to export the grades and attendance of the students.

2. REVIEW OF RELATED LITERATURE

2.1 Problem

In the "Automated Attendance Monitoring System using Android Platform", the problem is that the process of marking attendance and maintaining data is not fully automated. This results to errors in computation and it wastes a lot of time.

The problem on the literature entitled "A Novel Approach of Mobile Based Student Attendance Tracking System Using Android Application", is that in India, an education system has been changing widely in the last 10 years due to the continuous development of technology. Now with that, it is observed that the process of manual attendance is time

consuming and also inefficient at times. Now this process is being left out with the continuous growth of technology.

The third literature, entitled “Android Application for Student Activity Register”, states that the maintenance and management of student information is a tiring process in educational institutions. The workload of the teachers is heavy when handling more than one class. The traditional way of recording attendance manually in a log book and then entered again on a desktop application is time consuming. It is also the same when recording marks of tests. Thus reports generated from these two processes would sometimes contain errors.

The problem in the fourth literature entitled, “Mobile Based Attendance Marking System Using Android and Biometrics”, is that managing, updating, and seeing the attendance of records of students with the non-automated way is inefficient and can be prone to errors.

2.2 Approach

The “Automated Attendance Monitoring System using Android Platform” used a client-server approach and will be programmed using Android programming language. This system follows a specific hardware and software architecture. This will provide the necessary solution of automating attendance monitoring. The system will consist of two apk files, one for the teacher and the other for the student. The AMS will be used to check attendance and generate reports. It is also stated in the literature that the student will mark the attendance by a single click on his/her device. The teacher will have the ability to generate reports of one or more than one student.

The researchers approach on the paper “A Novel Approach of Mobile Based Student Attendance Tracking System Using Android Application” is to create a mobile application for attendance monitoring with the concept of web services implemented on an Android application that communicates with the database residing on the remote server. The application will be installed on the teacher’s smart phone or Android device. It intends to be

user friendly and will require minimal input for marking attendance of a student. Also, the application will have a strong user authentication and the transmission of data via web service. The literature also states that the application would not store any data on the user device and instead would store it on the server database.

The third literature “Android Application for Student Activity Register”, used a smart phone based application using Android to automate the manual work and make it easier, secured and less error prone. The researchers integrated RFID technology which consists of the RFID reader and the RFID tags to reduce fake attendance.

The fourth literature “Mobile Based Attendance Marking System Using Android and Biometrics” implemented an Android mobile application with biometric scanner that communicates with the database and verification can be achieved.

2.3 Methodology

In the “Automated Attendance Monitoring System using Android Platform”, the researchers first compared their method to past methods that have been applied to monitor the attendance of the students. They then created the application with a different approach as mentioned in the 2.2 Approach subsection. They divided their system into two apf files. One for the student and one for the teacher, each has their own functionalities. They also learned the client-server approach for their system. They also implemented MySQL as their database for their system. The researchers also developed the application with Android programming language using Eclipse framework. They deployed a server into a personal computer using apache-Tomcat7. The main requirement for their AMS is an Android device for the client side that will run the application and a personal computer for the server side that stores the database.

In the “A Novel Approach of Mobile Based Student Attendance Tracking System Using Android Application”, the researchers first compared the existing system with the proposed system. They filtered out or improved the existing system’s disadvantages in order to provide a good and enhanced system. Then they identified what their system’s software

and hardware requirements are such as JSON (JavaScript Object Notation), Eclipse for their IDE and SQL Server 2008 or later. After finalizing what they have gathered, they proceeded to the implementation of their system. First with the user authentication, next the calling of web service, next with the attendance process by putting checkboxes beside the name of the student, lastly displaying the information of the student by getting the information from the server. The researchers then tested their finished system on the client and lecturer side respectively.

In the "Android Application for Student Activity Register", the researchers first compared the past works and identified which features would they like to incorporate to their system. Then they identified what system architecture they need for the application. After comparing and identifying, they then proceeded to naming and conceptualizing the modules of their system.

In the "Mobile Based Attendance Marking System Using Android and Biometrics", the researchers used GPRS or Wi-Fi for their connection to the server and PHP files are executed on the server with SQL queries. The researchers also learned the nature of biometrics and incorporated it with their application. First they scanned fingerprints using the biometric sensor and extracted the features of the fingerprint before storing it into the database. The researchers then retrieved the extraction and then proceeded to implementing the application with the use of the scanned fingerprint as a means of logging in to the application. They used a GT-511C1 fingerprint scanner because it is portable and it suits the needs of the system.

2.4 Comparison Table

Name	**Features**	**Lacking features**
Automated Attendance Monitoring System using Android Platform	• Record attendance • SMS reports	• Input test scores • Seat plan

A Novel Approach of Mobile Based Student Attendance Tracking System Using Android Application	• Record attendance • Attendance report	• Input test scores • SMS reports • Seat plan
Android Application for Student Activity Register	• Record attendance • Input test scores • Auto calculation of attendance and test scores	• SMS reports • Seat plan
Mobile Based Attendance Marking System Using Android and Biometrics	• Record attendance	• SMS reports • Input test scores • Seat plan
Proposed Mobile Classroom Management Application	• Record attendance • Seat plan • Input grades • SMS reports • Auto calculation of attendance and grades	

Table 1. The table compares what are the features of the related works compared to the proposed Mobile Classroom Management Application.

3. METHODOLOGY

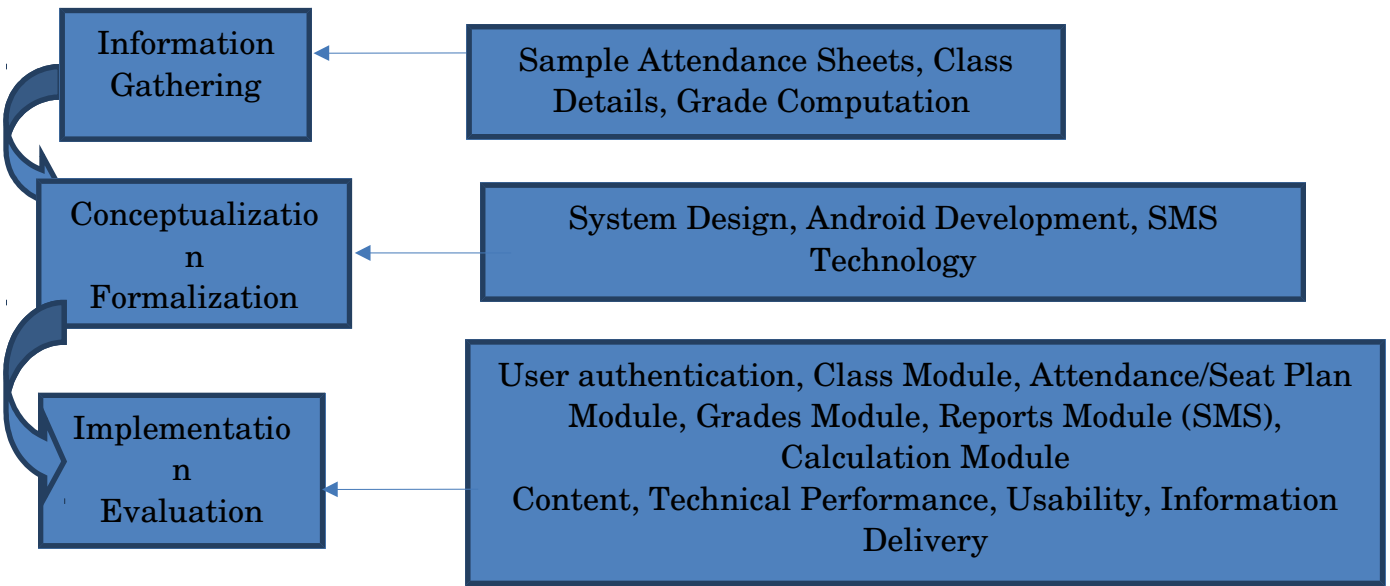


Figure 1. Shows the framework on how the researchers achieved the proposed system.

3.1 Information Gathering

The researchers would gather information such as sample attendance sheets, class details, and the computation of grades from teacher or beadles. These information must be from within the school itself. The researchers would also gather information online on how to develop Android programs, how to implement the SMS technology.

3.2 Conceptualization and Formalization

3.2.1 Android Development

The researchers would formalize the needed tools in creating the application such as the programming language and the Integrated Development Environment or IDE.

3.2.2 SMS Technology

The researchers would formalize the nature on how SMS technology will be implemented within the application such as what information will be sent to the students.

3.3 Implementation and Evaluation

3.3.1 Modules

3.3.1.1 User authentication module

In this module, when first running the application, the user will be prompted to sign up first before using the application. The user must provide a unique username and password. This is to provide security within the application.

3.3.1.2 Class module

In this module, the user must first add a class and input the subject code, subject name, time, and room of that class. After the user adds a class.

3.3.1.3 Attendance/Seat Plan Module

In this module, the user can record the student's attendance within the seat plan module. This is to prevent the hassle of manual roll call. The teacher can just look at who's not around and then immediately know who is absent with the use of the seat plan feature.

3.3.1.3 Grade module

The purpose of this module is to input the students' grades for that specific class.

3.3.1.4 Calculation module

This module calculates the total number of absences a student has incurred during class, also the letter grade equivalent of his/her grades.

3.3.1.5 Reports (SMS) module

The purpose of this module is to send SMS reports to the student informing his/her status in a specific class.

3.4 Conceptual Framework

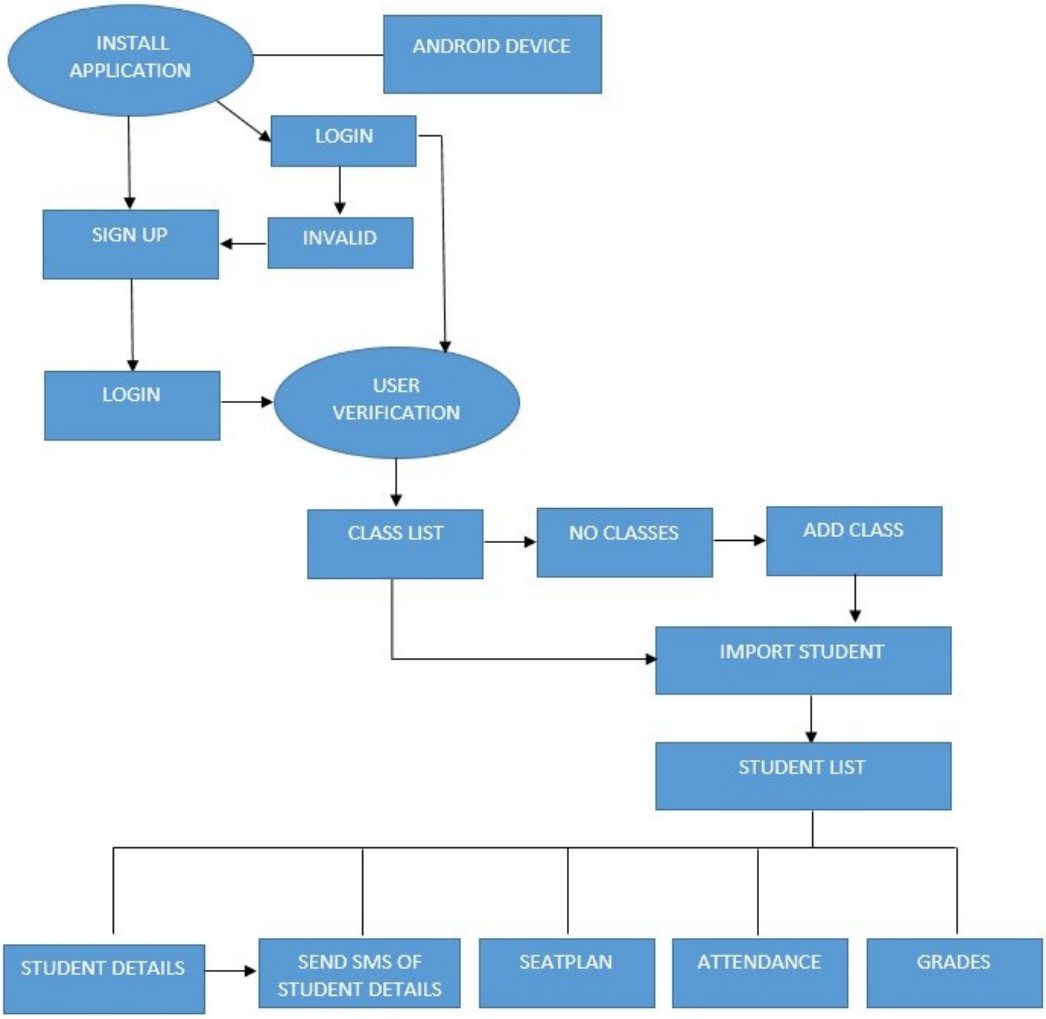


Figure 5. Conceptual Framework for the proposed Classroom Management Application

4. THEORETICAL BACKGROUND

4.1 Attendance Monitoring

4.1.1 What is Attendance Monitoring?

Attendance monitoring is the act of recording student attendance and monitoring absences and tardiness throughout the school year. This is important in most universities because attendance plays a big role in student performance.

4.1.1.1 Mobile Attendance Monitoring System

An application that prevents students to abuse the unrecorded number of absences, the application will keep track of every student attendance in class with high amount of certainty. The program will also help teachers to check and manage the attendance in a more efficient way and a more environmental friendly since it a paper free way of tracking the attendance.

4.2 Android Software Development

4.2.1 What is Android?

An operating system on mobile devices and currently being developed by Google. Android features different version like Lollipop 5.0, KitKat 4.4, Jelly Bean 4.3 and etc. Android's interface is based on touch screen technology which provides direct manipulation also Android's designs are primarily for touchscreen devices like tablets and smart phones. Many of Android's application where developed by Java programming language with the use of Android software development kit. Android is one of the world wide distributed operating system for mobile devices.

4.2.2 Programming Language

A formally constructed language designed to send instructions to a machine like a computer. A programming language can be used to control a certain behavior of a machine or in the researcher's case, an application.

4.2.2.1 Java

Java is a high-level computer programming language that is simple, object-oriented, distributed, and etc. Java programs run on every platform that supports java without the need of compiling it again. Java programming allows the user to write English based codes which make programming easy. Java ensures that once a program is compiled you can run it to any computer that has a java virtual machine. There are many applications and software that uses java programming, one of these applications is Android.

4.2.3 Android Studio

Android Studio is the official IDE for Android application. It is based on JetBrains' IntelliJ IDEA software and is designed specifically for developing Android applications.

4.2.4 SQLite

SQLite is an open source type of database. Now, this goes well with Android because Android comes with a built-in SQLite implementation for the database. This will be used in the application to store data and retrieve data.

4.2.5 SmsManager

Manages SMS operations such as sending data, text, and pdu SMS messages. Get this object by calling the static method SmsManager.getDefault().

4.3 Assessment/Evaluation

The proposed application is evaluated using the following question which has 3 major categories:

Usefulness:

Q1: In assigning seats, is it easier compared to the manual method?

Q2: In checking of attendance, is it easier than manual roll call?

Q3: Would you agree that the attendance report module is efficient and useful?

Q4: In recording of grades, is it convenient and easy to use?

Q5: Would you agree that the grade report module is efficient and useful?

Q6: Would the SMS feature of sending grades and number of absences to the student useful?

Q7: Would you agree that the importing of students handy and easy to use?

Q8: Would you agree that the exporting of grades and attendance convenient?

Q9: Would you agree that in using the application it improves my job performance?

Ease of Use:

Q1: Is learning to use the application easy?

Q2: Would it be easy for me to interact with the application?

Q3: Is the application quick to respond?

Q4: Is it easy to get used to in using the application?

Design:

Q1: Are the words easy to read?

Q2: Are the arrangement of screens proper?

Q3: Organization of information (Attendance, Grades, Seat Plan)

Q4: Are the menu buttons present useful?

5. RESULTS AND DISCUSSION

Testers:

1. Mr. Antonio Bulao – Computer Studies (Information Technology)
2. Mr. Jose Mari Freires – Computer Studies (Information Technology)
3. Mr. Patrick Paasa – Computer Studies (Information Technology)
4. Mr. Adrian Ablazo – Computer Studies (Information Systems)

5. Mr. Raul Vincent Lumapas – Computer Studies (Information Technology)
6. Mr. Bernie Jereza – Computer Studies (Information Technology)
7. Mr. Dante Calamba – Humanities and Letters (Language)
8. Mr. Eugene Bije – Finance
9. Dr. Antonio Emberda – University Research Council
10. Mr. Jayson James Bentayao – Philosophy
11. Mr. Kromyko Cruzado – Computer Studies
12. Mr. Romarico Robert Doctura - Mathematics
13. Ms. Leonore Loqueloque, CPA, DBM Chair – Accounting Technology
14. Ms. Duane Gravador – Philosophy
15. Ms. KC L Corro – Computer Studies
16. Ms. Novie Joy Pelobello – Computer Studies
17. Ms. Daryl Hagayuhay – Accountancy
18. Ms. Ruth Chew – Accounting Technology
19. Ms. Christine Pido – SCWA Faculty
20. Ms. Vangie Comicho – GenSan City High School Faculty

Summary:

The test was conducted with 20 different teachers, the proponents would let the participants used the proposed application and afterwards the researchers would let them answer a survey form.

The proposed application received good reviews from the participants, most of the participants agreed that the application was useful in doing classroom management like in checking attendance and assigning of grades. The proposed application SMS function had most of the best reviews. The application also had a good review in the ease of usage of the application, most of the participants find the application easy to use, due to the available tooltips that helps the participants on what to do next in the application. The participants find the design of the application good though there were some who recommended the improvement of the user interface of the application. The test results were good and the rating of the application was high due to the participants that agreed to the application.

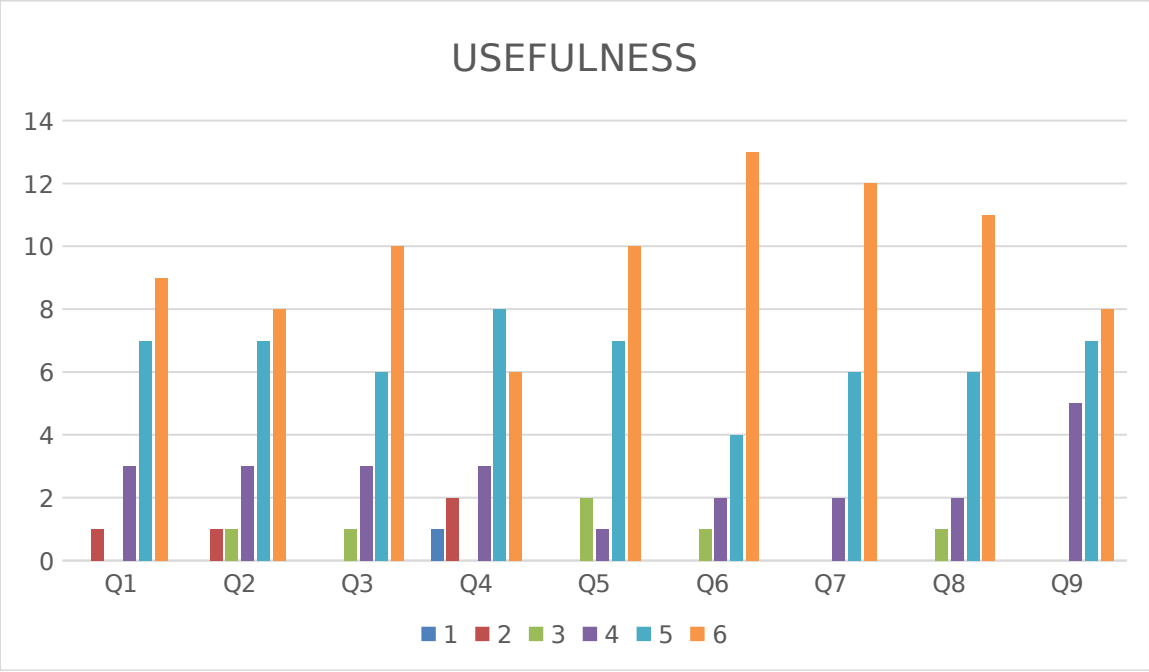
Formula: **Average Rating = Number of Response in each Rating / Total number of respondents**

Usefulness *Strongly Disagree 1..2..3..4..5..6 Strongly Agree*

	1	2	3	4	5	6	Total	Average Rating	Top Two Box
Q1: In assigning seats, is it easier compared to the manual method?		1 5%		3 15%	7 35%	9 45%	20	5.15	80 %
Q2: In checking of attendance, is it easier than manual roll call?		1 5%	1 5%	3 15%	7 35%	8 40%	20	5	75%
Q3: Would you agree that the attendance report module is efficient and useful?			1 5%	3 15%	6 30%	10 50%	20	5.25	80%
Q4: In recording of grades, is it convenient	1 5%	2 10%		3 15%	8 40%	6 30%	20	4.65	70%

and easy to use?									
Q5: Would you agree that the grade report module is efficient and useful?			2 10%	1 5%	7 35%	10 50%	20	5.25	85%
Q6: Would the SMS feature of sending grades and number of absences to the student useful?			1 5%	2 10%	4 20%	13 65%	20	5.45	85%
Q7: Would you agree that the importing of students handy and easy to use?				2 10%	6 30%	12 60%	20	5.5	90%
Q8: Would you agree that the exporting of grades and attendance			1 5%	2 10%	6 30%	11 55%	20	5.35	85%

convenient?									
Q9: Would you agree that in using the application can improve my job performance ?				5 25%	7 35%	8 40%	20	5.15	75%



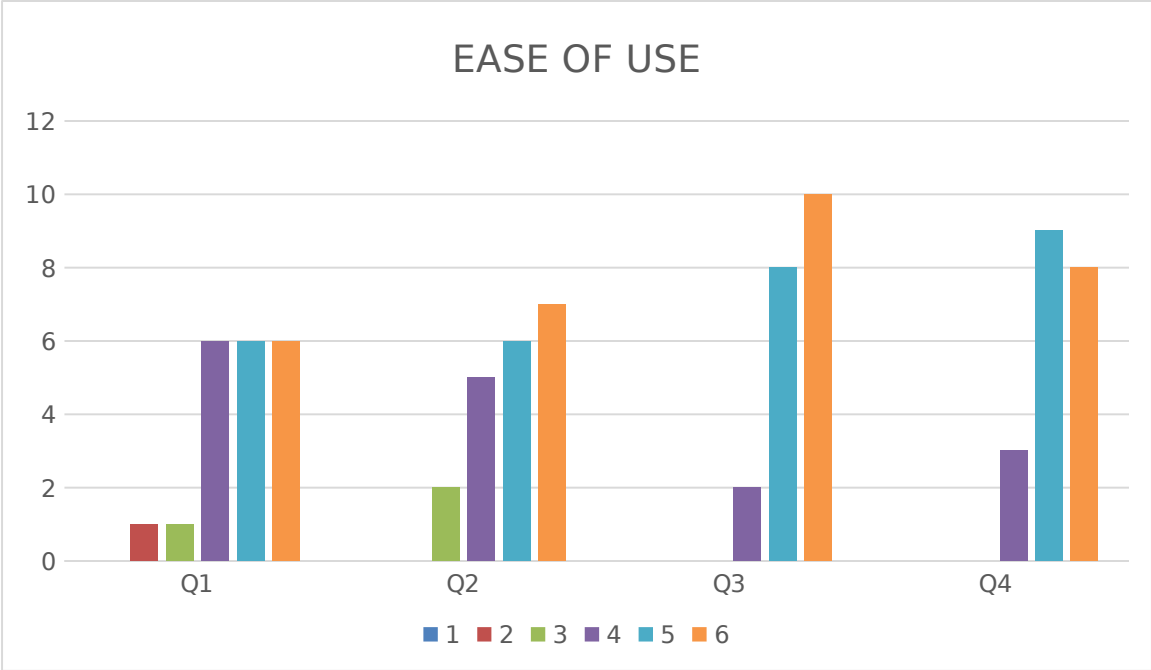
The results show that the respondents found the assignment of seating in the application easier compared to the manual way as shown in the average rating which is equal to 5.15. They found the checking of attendance easier with the average rating of 5. The participants also find the attendance module efficient to use and it has an average rating of 5.25. The recording of grades was

also helpful for the participants, and it had an average rating of 4.65 and the grade report module had an average rating of 5.25. The SMS also had a good impression to the respondents for it received an average rating of 5.45. The import function of the students was handy for the respondents and it had an average rating of 5.5. The export module for the grades and attendance of the student is useful for the respondents and had an average rating of 5.35. The respondents agreed that the application can improve job performance and had an average rating of 5.15

Ease of Use *Strongly Disagree 1..2..3..4..5..6 Strongly Agree*

	1	2	3	4	5	6	Total	Average Rating	Top Two Box
Q1: Is learning to use the application easy?		1 5%	1 5%	6 30%	6 30%	6 30%	20	4.75	60%
Q2: Would it be easy for me to interact with the application ?			2 10%	5 25%	6 30%	7 35%	20	4.9	65%
Q3: Is the application quick to respond?				2 10%	8 40%	10 50%	20	5.4	90%

Q4: Is it easy to get used to in using the application ?				3 15%	9 45%	8 40%	20	5.25	85%

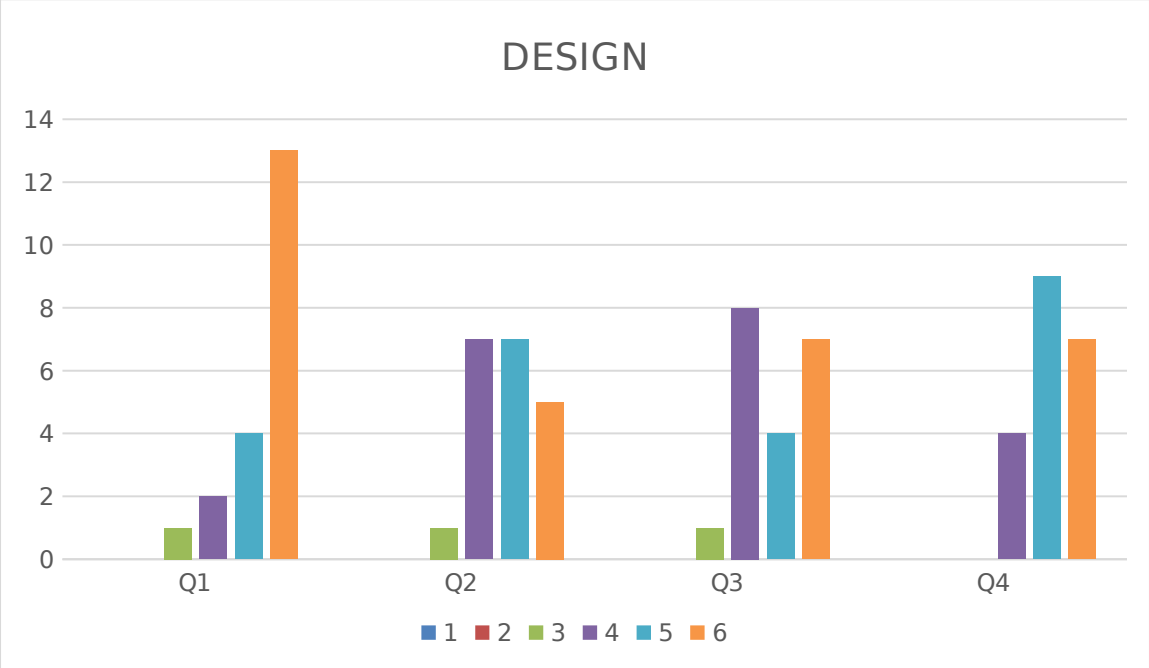


The results of the ease of use category of the survey show that the application was easy to learn and had an average rating of 4.75. The respondents also find the application easy to interact with and had an average rating of 4.9. The average rating in the application in which the respondents find the application quick to respond is 5.4. The respondents also find it easy to get used to the application due to the help of tooltips present and it gain an average rating of 5.25.

Design *Hard / Confusing 1..2..3..4..5..6 Easy / Very Clear*

	1	2	3	4	5	6	Total	Average Rating	Top Two Box

Q1: Are the words easy to read?			1 5%	2 10%	4 20%	13 65%	20	5.45	85%
Q2: Are the arrangement of screens proper?			1 5%	7 35%	7 35%	5 25%	20	4.8	60%
%Q3: Organization of information (Attendance, Grades, Seat Plan)			1 5%	8 40%	4 20%	7 35%	20	4.85	55%
Q4: Are the menu buttons present useful?				4 20%	9 45%	7 35%	20	5.15	80%



The results show that the respondents find the words in the application easy to read and gave it an average rating of 5.45. The sequencing of the screen also impressed most of the respondents and given it an average rating of 4.8. The organization of information in the application had a good feedback to the proponents and it gained an average rating of 4.85. Lastly, the menu buttons present in the application was useful for the respondents and they gave a positive feedback on it which had an average rating of 5.15

The acceptable rating for the application is from 5-6 namely, agree and strongly agree respectively. In percentage, 87.3% rated the application from 5-6 in most questions. The total Agreed Average is 5.14 which means that most of the respondents agreed that the application is useful and can help aid their work. 95%, those who answered from 4-6, of the respondents slightly agrees or strongly agrees that the application is useful. While 5% of the respondents were quite dissatisfied.

Discussion

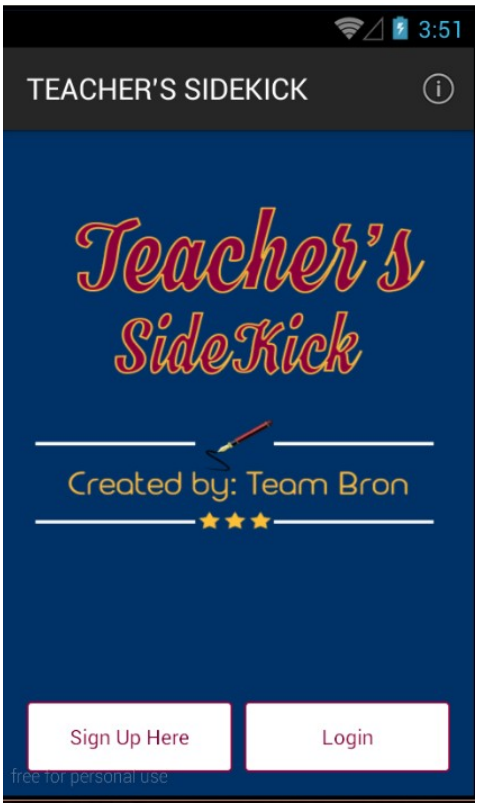


Figure 2. Homepage

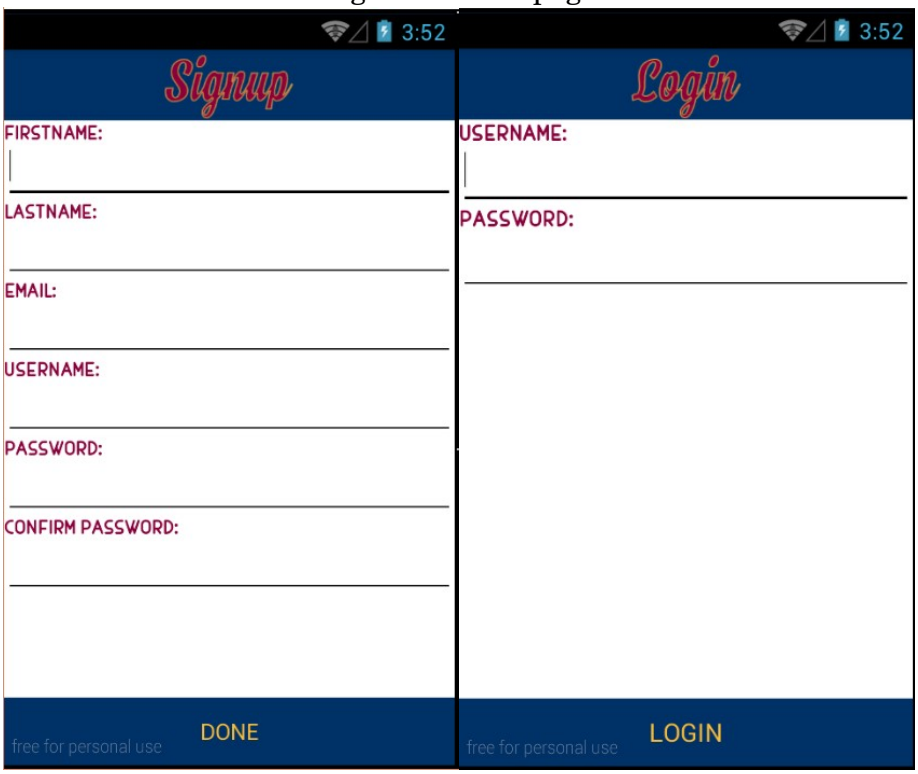


Figure 3. Sign up and Login

```
if(username.equals("")){
    Toast unameblank = Toast.makeText(Login.this, "PLEASE ENTER USERNAME.", Toast.LENGTH_SHORT);
    unameblank.show();
}
else if(password.equals("")){
    Toast passblank = Toast.makeText(Login.this, "PLEASE ENTER PASSWORD.", Toast.LENGTH_SHORT);
    passblank.show();
}
else if(password.equals(checkpass)){
    Bundle bundle = new Bundle();
    bundle.putString("USERNAME", String.valueOf(username));
    Intent i = new Intent(getBaseContext(), Home.class);
    i.putExtras(bundle);
    startActivity(i);
    Toast login = Toast.makeText(Login.this, "Welcome, " + username + "!", Toast.LENGTH_SHORT);
    login.show();
}
else{
    Toast temp = Toast.makeText(Login.this, "USERNAME AND PASSWORD DOESN'T MATCH.", Toast.LENGTH_SHORT);
    temp.show();
}
```

Figure 3.1 Code for Login

First, the program checks the field for the username if it is blank and displays a prompt if so. Same goes on for the password. The method password.equals(checkpass) checks the password and username if it matches in the database. If it's successful, the value for the username is stored in a Bundle which is a Class in Android that passes String values from one activity to another. Afterwards, it starts the Home activity which holds the Homepage of the application.

```
if(!pass1.equals(pass2)){
    Toast pass = Toast.makeText(SignUp.this, "PASSWORDS DOESN'T MATCH.", Toast.LENGTH_SHORT);
    pass.show();
}
else if(fname.equals("") || lname.equals("") || em.equals("") || uname.equals("") || pass1.equals("") || pass2.equals("")){
    Toast blank = Toast.makeText(SignUp.this, "PLEASE COMPLETE THE FORM.", Toast.LENGTH_SHORT);
    blank.show();
}
else if(helper.exist(uname)){
    Toast exist = Toast.makeText(SignUp.this, "USERNAME ALREADY EXISTS.", Toast.LENGTH_SHORT);
    exist.show();
}
else{
    User u = new User();
    u.setFname(fname);
    u.setLname(lname);
    u.setEmail(em);
    u.setUsername(uname);
    u.setPass(pass1);

    helper.insertUser(u);
    Toast insert = Toast.makeText(SignUp.this, "SUCCESSFUL.", Toast.LENGTH_SHORT);
    insert.show();

    Bundle bundle = new Bundle();
    bundle.putString("USERNAME", uname);
    Intent i = new Intent(getBaseContext(), Home.class);
    i.putExtras(bundle);
    startActivity(i);
    Toast login = Toast.makeText(SignUp.this, "Welcome, " + uname + "!", Toast.LENGTH_SHORT);
    login.show();
```

Figure 3.2 Code for Signup

The program checks the two fields of passwords if it matches and also checks the other fields if it's blank, if the statement is true, it then shows a specific prompt to inform the user. But if the statement isn't true, it goes to save the values for the User class. Each field is then set to a specific column in the database. Afterwards, if it's

successful, the user is redirected to the Homepage. The purpose of the Bundle class is to hold the string value for the username to pass from the SignUp page to the Homepage of the application.

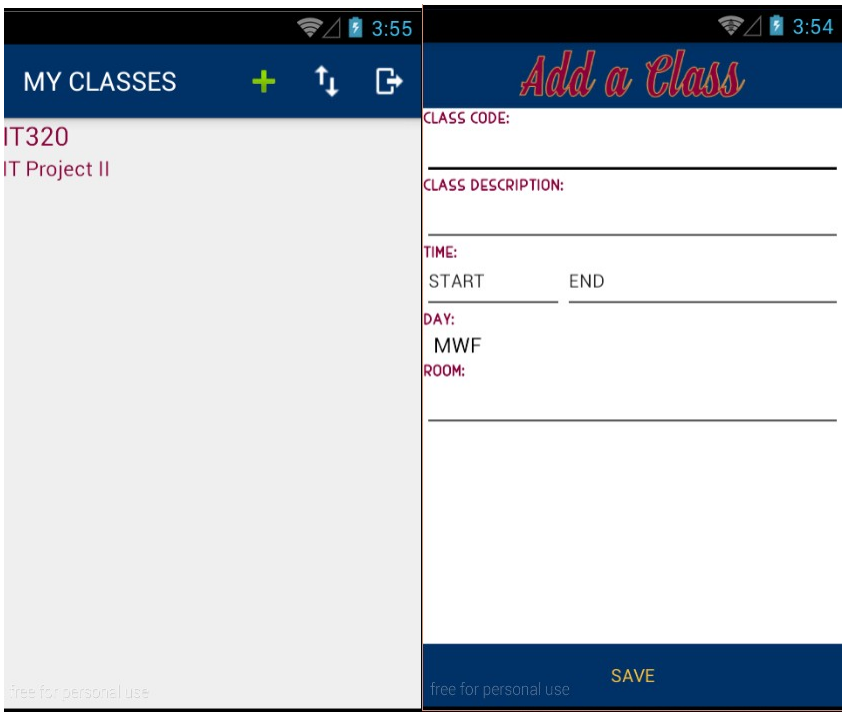


Figure 4. Class List and Adding a class form

```
if(classcodestr.equals("") || subjdescstr.equals("") || starttimestr.equals("") || endtimestr.equals("") || roomstr.equals("")){
    Toast blank = Toast.makeText(ClassForm.this, "PLEASE COMPLETE THE FORM.", Toast.LENGTH_SHORT);
    blank.show();
}
else{
    Class c = new Class();
    c.setID(Integer.parseInt(newID));
    c.setClasscode(classcodestr);
    c.setSubjdesc(subjdescstr);
    c.setStarttime(starttimestr);
    c.setEndtime(endtimestr);
    c.setDay(dayvalue);
    c.setRoom(roomstr);
    c.setUsername(username);
    classhelper.insertClass(c);

    Toast insert = Toast.makeText(ClassForm.this, "SUCCESSFUL.", Toast.LENGTH_SHORT);
    insert.show();
    classcode.setText("");
    subjdesc.setText("");
    starttime.setText("");
    endtime.setText("");
    room.setText("");

    Alert();
}
```

```
public void insertClass(Class c){
    db = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    String query2 = "select * from " + TABLE_CLASS;

    values.put(COLUMN_CLASSID, c.getID());
    values.put(COLUMN_CODE, c.getClasscode());
    values.put(COLUMN_DESC, c.getSubjdesc());
    values.put(COLUMN_START, c.getStarttime());
    values.put(COLUMN_END, c.getEndtime());
    values.put(COLUMN_DAY, String.valueOf(c.getDay()));
    values.put(COLUMN_ROOM, c.getRoom());
    values.put(COLUMN_USER, c.getUsername());

    db.insert(TABLE_CLASS, null, values);
    db.close();
}
```

Figure 4.1 Code for class form

The program checks the fields whether it is blank or not and if isn't blank, it then proceeds to inserting each value from the field to the specific column in the database. It uses the insertClass method which gets the values inputted and then stored using the ContentValues class in Android. The values are successfully stored in the insert method from the getWritableDatabase() in Android.

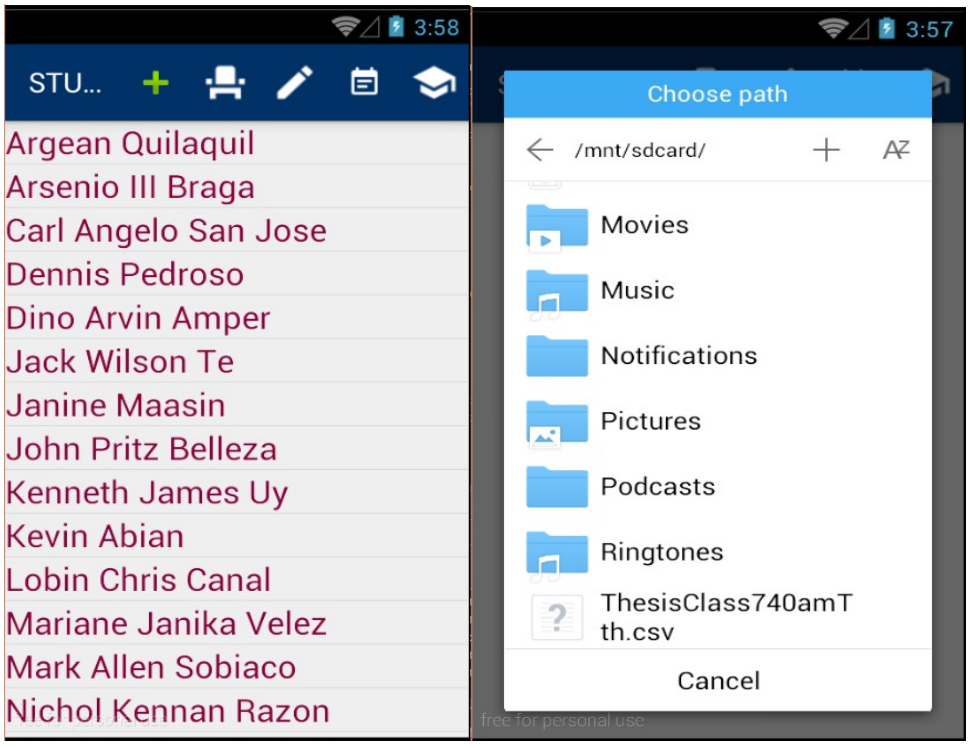


Figure 5. Student List and Importing students form

```
Intent fileintent = new Intent(Intent.ACTION_GET_CONTENT);
fileintent.setType("gagt/sdf");
try {
    startActivityForResult(fileintent, requestcode);
} catch (ActivityNotFoundException e) {
    Toast.makeText(Menu.this, "No activity can handle picking a file. Showing alternatives.", Toast.LENGTH_SHORT).show();
}
if (data == null)
    return;
switch (requestCode) {
    case requestcode:
        String filepath = data.getData().getPath();
        studhelper = new DBHelper(getApplicationContext());
        SQLiteDatabase db = studhelper.getWritableDatabase();
        String tableName = "student";
        String gradeTable = "grades";
        db.execSQL("delete from " + gradeTable + " where g_class = '" + tfid.getText().toString() + "'");
        db.execSQL("delete from " + tableName + " where studid !=0 and stud_classid = '" + tfid.getText().toString() + "'");
        try {
            if (resultCode == RESULT_OK) {
                try {
                    FileReader file = new FileReader(filepath);
                    BufferedReader buffer = new BufferedReader(file);
                    ContentValues contentValues = new ContentValues();
                    ContentValues contentValues2 = new ContentValues();
                    String line = "";
                    db.beginTransaction();
                while ((line = buffer.readLine()) != null) {
                    String[] str = line.split(",", 3);  // defining 3 columns with null or blank field //values acceptance
                    //Name, Contact, Other Info, Class ID
                    String name = str[0].toString();
                    String contact = str[1].toString();
                    String other = str[2].toString();
                    String classid = tfid.getText().toString();
                    contentValues.put("stud_name", name);
                    contentValues.put("stud_contact", contact);
                    contentValues.put("stud_otherinfo", other);
                    contentValues.put("stud_classid", classid);
                    contentValues2.put("g_student", name);
                    contentValues2.put("g_class", classid);
                    db.insert(tableName, null, contentValues);
                    db.insert(gradeTable, null, contentValues2);
                    populateListView();
                    populateLec();
                    populateLab();
                }
                db.setTransactionSuccessful();
                db.endTransaction();
            } catch (IOException e) {
                if (db.inTransaction())
                    db.endTransaction();
                Dialog d = new Dialog(this);
                d.setTitle(e.getMessage().toString() + "first");
                d.show();
                // db.endTransaction();
            }
```

Figure 5.1 Code for importing students

The first part starts a file picker using the Intent class. If there isn't a suitable file picker for the application present in your phone, it cancels the process. But if there is a suitable file picker, it proceeds to the requestcode case which accesses the database and deletes the files present in the database if the data imported is already present to avoid duplication. It then tries to read the file by line and stored in the ContentValues class in Android. The program continues to read the file until it is equal to null. Each line in the file are separated by a comma and then stored to the database from the ContentValues class and inserted using the getWritableDatabase() method in Android.

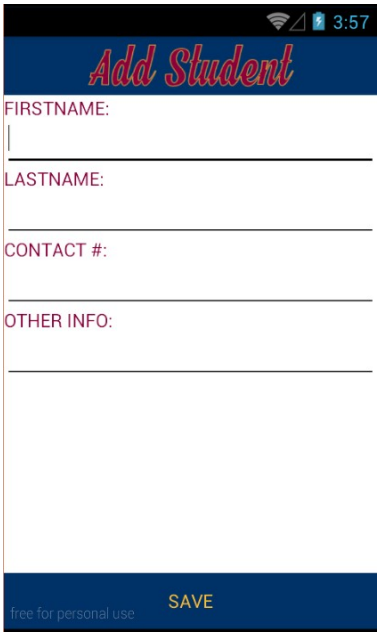


Figure 6. Adding a Student form

```
if(v.getId() == R.id.bSaveStud){
    if(studfirst.equals("") || studlast.equals("") || studcont.equals("")){
        Toast.makeText(StudentForm.this,"Please complete the form.",Toast.LENGTH_SHORT).show();
    }
    else {
        Student s = new Student();
        s.setFirstname(studfirstname);
        s.setLastname(studlastname);
        s.setContact(studcontact);
        s.setOtherinfo(studotherinfo);
        s.setClassid(studclassid);
        studhelper.insertStudent(s);

        ContentValues contentValues = new ContentValues();
        SQLiteDatabase db = studhelper.getWritableDatabase();
        contentValues.put("g_student", studfirstname + " " + studlastname);
        contentValues.put("g_class", studclassid);
        db.insert("grades", null, contentValues);

        Toast insert = Toast.makeText(StudentForm.this, "SUCCESSFUL.", Toast.LENGTH_SHORT);
        insert.show();
    }
```

```
public void insertStudent(Student s){
    db = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    String query2 = "select * from " + TABLE_STUDENT;

    values.put(COLUMN_SNAME, s.getFirstname() + " " + s.getLastname());
    values.put(COLUMN_SCONTACT, s.getContact());
    values.put(COLUMN_SOTHERINFO, s.getOtherinfo());
    values.put(COLUMN_SCLASSID, s.getClassid());

    db.insert(TABLE_STUDENT, null, values);
    db.close();
}
```

Figure 6.1 Code for Adding a student manually

The program checks the fields whether it is blank or not and if isn't blank, it then proceeds to inserting each value from the field to the specific column in the database. It uses the insertUser method which gets the values inputted and then stored using the ContentValues class in Android. The values are successfully stored in the insert method from the getWritableDatabase() in Android.

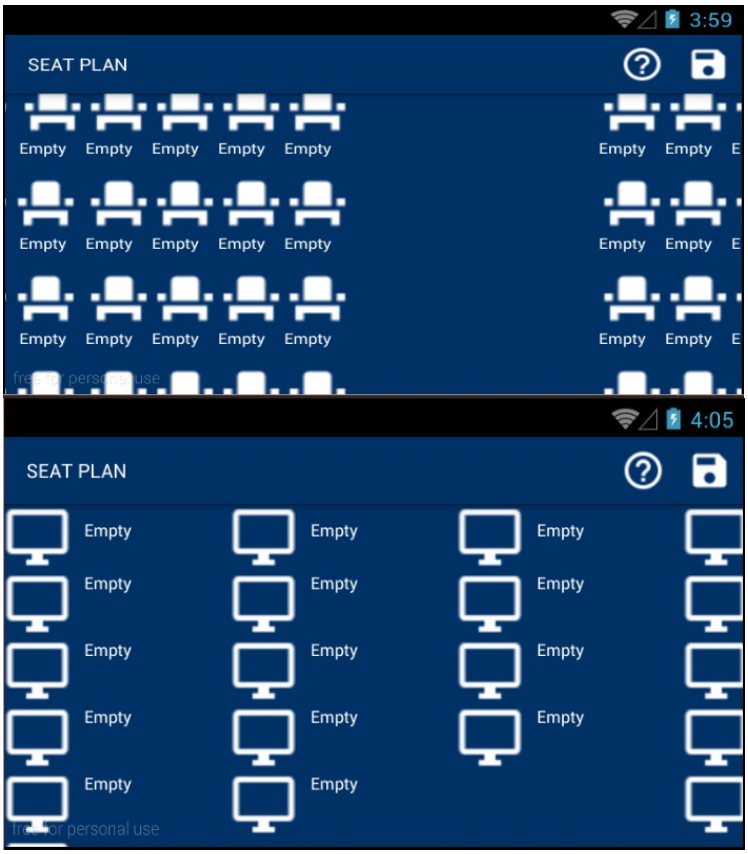


Figure 7. Seat plan for lecture and laboratory

```
s1 = (TextView)findViewById(R.id.tvseat1);
s1.setOnClickListener((v) -> {
        cleared = false;
        studentSpinner.performClick();
        studentSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                String str = parent.getItemAtPosition(position).toString();
                if(!cleared) {
                    if(str.equals("Select student:")){
                        Toast.makeText(SeatPlan.this, "Not permitted.", Toast.LENGTH_SHORT).show();
                    }
                    else {
                        if(s1.getText().toString().equals("Empty")){
                            s1.setText(str);
                            dataAdapter.remove(str);
                            dataAdapter.notifyDataSetChanged();
                            cleared = true;
                            studentSpinner.setSelection(0,true);
                        }
                        else {
                            dataAdapter.add(s1.getText().toString());
                            s1.setText(str);
                            dataAdapter.remove(str);
                            dataAdapter.notifyDataSetChanged();
                            cleared = true;
                            studentSpinner.setSelection(0, true);
                        }
                    }
```

Figure 7.1 Code for assigning a student to a seat

The program checks if the TextView is tapped or pressed, if it's tapped, the program proceeds to showing a list of students in a Dialog Box. The user then selects a student from the Dialog Box and the program handles the user's actions. The user isn't able to select the "Select student:" string in the Dialog Box. But if the user selects a student from the Dialog Box, it then gets the string value of the selected student and then sets it into the TextView. The selected string is then removed from the list and the list is notified after the change so that it updates the list of students. The Spinner, which holds the list of students, is then set so that the value in index 0 can be selected again so that the user can continuously select a student. But if the user wishes to change the student, the program then gets the string value of the student in the TextView and added to the list so that the student can be selected once again after the change.

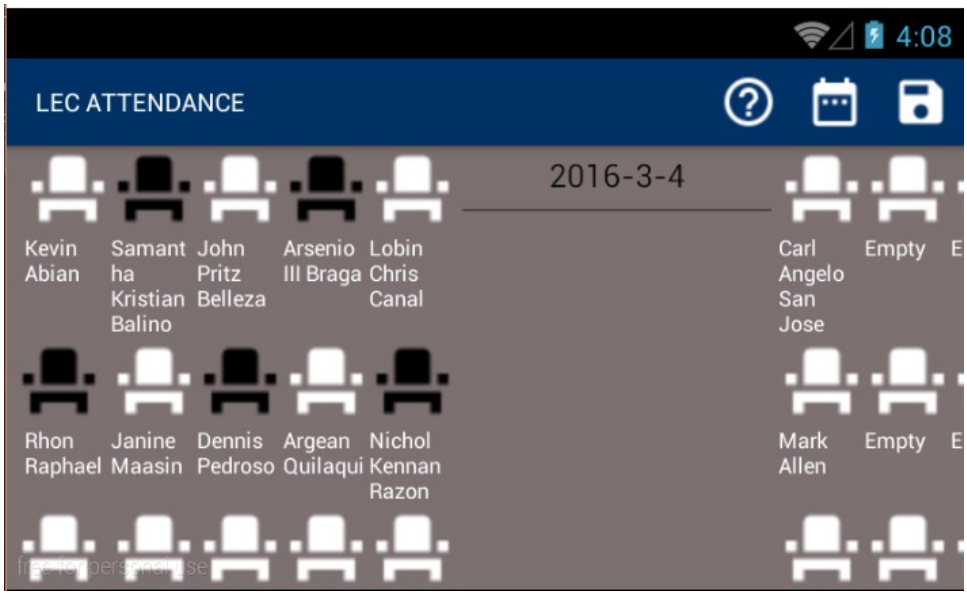


Figure 8. Attendance Module

```
ivi = (ImageView)findViewById(R.id.ivi);
ivi.setOnClickListener((v) → {
        if (count[0] == 0) {
            ivi.setImageResource(R.drawable.ic_event_seat_black_24dp);
            count[0] = count[0] + 1;
            stat1.setText("Absent");
        } else if (count[0] != 0) {
            ivi.setImageResource(R.drawable.ic_event_seat_white_24dp);
            count[0] = 0;
            stat1.setText("Present");
        } else {
            count[0] = 0;
        }
});
```

Figure 8.1 Code for Attendance Module

The program checks if the ImageView is tapped, the ImageView is the seat in the GUI. A count variable which is an Array handles the tap action from the user. By default, each seat is set to 0 which is Present. If the user taps the ImageView, it then adds 1 to the variable so that it is Absent. But if the user taps the seat which is Absent, it then resets the count variable back to 0 and is back to being Present.

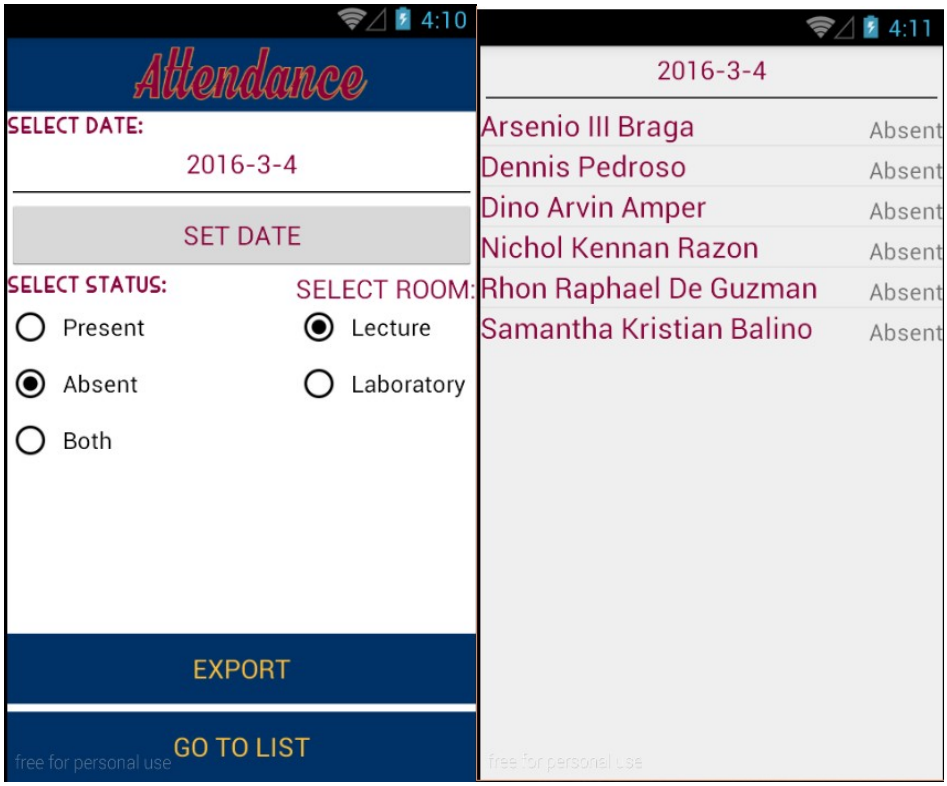
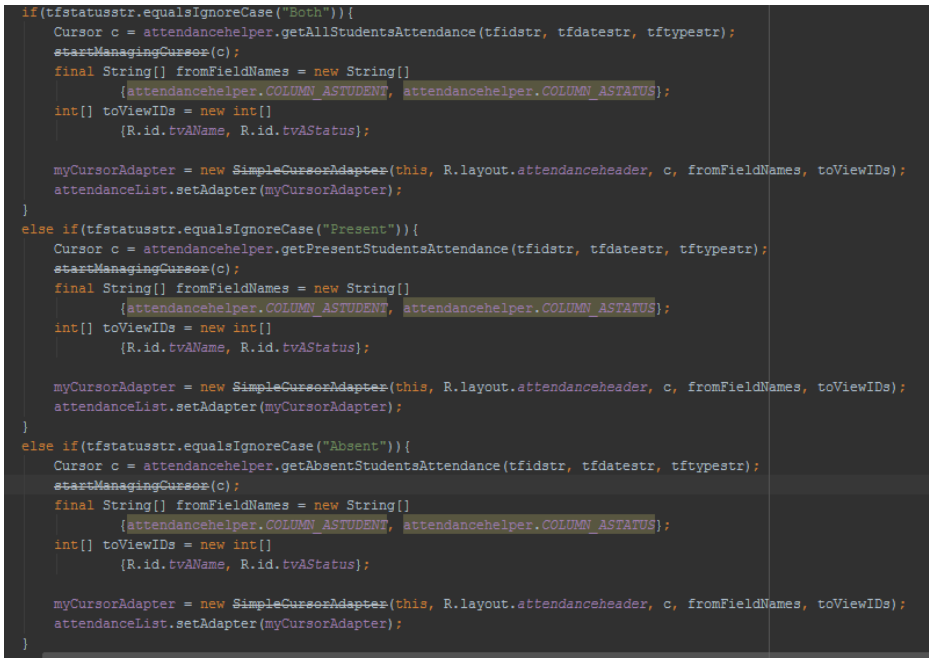


Figure 9. Attendance Master List

```
int selectedId = radioGroup.getCheckedRadioButtonId();
int selectedId2 = radioGroup2.getCheckedRadioButtonId();
selectedRb = (RadioButton)findViewById(selectedId);
selectedRb2 = (RadioButton)findViewById(selectedId2);
if(radioGroup.getCheckedRadioButtonId() == -1 && radioGroup2.getCheckedRadioButtonId() == -1 && tfDate.getText().toString().equals(""))
    Toast.makeText(AttendanceForm.this,"Please set date, status and room type.",Toast.LENGTH_SHORT).show();
}
else if(radioGroup.getCheckedRadioButtonId() == -1){
    Toast.makeText(AttendanceForm.this,"Please select status.",Toast.LENGTH_SHORT).show();
}
else if(radioGroup2.getCheckedRadioButtonId() == -1){
    Toast.makeText(AttendanceForm.this,"Please select room type.",Toast.LENGTH_SHORT).show();
}
else if(tfDate.getText().toString().equals("")){
    Toast.makeText(AttendanceForm.this,"Please set date.",Toast.LENGTH_SHORT).show();
}
else {
    String datetopass = tfDate.getText().toString();
    String idtopass = tfid.getText().toString();
    String statustopass = selectedRb.getText().toString();
    String typetopass = selectedRb2.getText().toString();
    Bundle bundle = new Bundle();
    bundle.putString("DATE", datetopass);
    bundle.putString("CLASSIDTOPASS", idtopass);
    bundle.putString("STATUS", statustopass);
    bundle.putString("TYPE", typetopass);
    Intent i = new Intent(getBaseContext(), AttendanceList.class);
    i.putExtras(bundle);
    startActivity(i);
}
```



```
if(tfstatusstr.equalsIgnoreCase("Both")){
    Cursor c = attendancehelper.getAllStudentsAttendance(tfidstr, tfdatestr, tftypestr);
    startManagingCursor(c);
    final String[] fromFieldNames = new String[]
            {attendancehelper.COLUMN_ASTUDENT, attendancehelper.COLUMN_ASTATUS};
    int[] toViewIDs = new int[]
            {R.id.tvAName, R.id.tvAStatus};

    myCursorAdapter = new SimpleCursorAdapter(this, R.layout.attendanceheader, c, fromFieldNames, toViewIDs);
    attendanceList.setAdapter(myCursorAdapter);
}
else if(tfstatusstr.equalsIgnoreCase("Present")){
    Cursor c = attendancehelper.getPresentStudentsAttendance(tfidstr, tfdatestr, tftypestr);
    startManagingCursor(c);
    final String[] fromFieldNames = new String[]
            {attendancehelper.COLUMN_ASTUDENT, attendancehelper.COLUMN_ASTATUS};
    int[] toViewIDs = new int[]
            {R.id.tvAName, R.id.tvAStatus};

    myCursorAdapter = new SimpleCursorAdapter(this, R.layout.attendanceheader, c, fromFieldNames, toViewIDs);
    attendanceList.setAdapter(myCursorAdapter);
}
else if(tfstatusstr.equalsIgnoreCase("Absent")){
    Cursor c = attendancehelper.getAbsentStudentsAttendance(tfidstr, tfdatestr, tftypestr);
    startManagingCursor(c);
    final String[] fromFieldNames = new String[]
            {attendancehelper.COLUMN_ASTUDENT, attendancehelper.COLUMN_ASTATUS};
    int[] toViewIDs = new int[]
            {R.id.tvAName, R.id.tvAStatus};

    myCursorAdapter = new SimpleCursorAdapter(this, R.layout.attendanceheader, c, fromFieldNames, toViewIDs);
    attendanceList.setAdapter(myCursorAdapter);
}
```

Figure 9.1 Code for Attendance Master List

The program checks the conditions selected by the user, it then stores each selection the Bundle class in Android that holds String values for passing from one Activity to another. The values gathered are then used for sufficing the three different methods of getting the student's attendance namely, getAllStudentsAttendance(), getPresentStudentsAttendance(), and getAbsentStudentsAttendance(). The parameters for each method are the class ID, Date, and the Room Type, those values are gathered from the previous activity where the user set the conditions. Now the rows retrieved from the database are now set to each TextView from the attendanceheader layout and is set to a List.

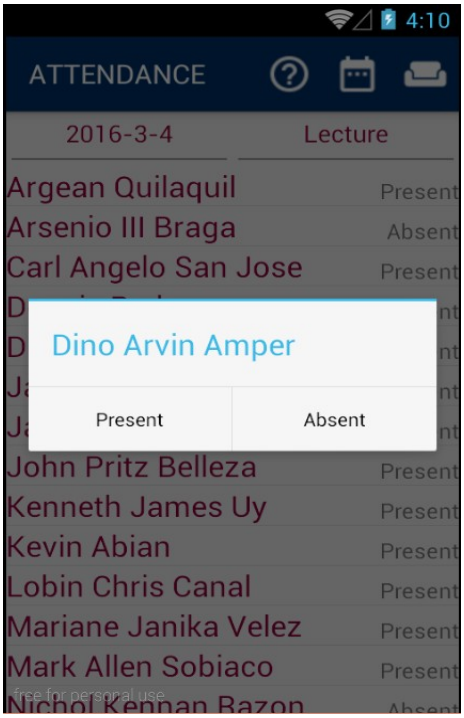


Figure 10. Update Attendance

```
classList.setOnItemClickListener((parent, view, position, id) -> {
        idcursor = (Cursor) parent.getItemAtPosition(position);
        count = position;
        studentname = idcursor.getString(idcursor.getColumnIndex("attendance_student"));

        final AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(UpdateAttendance.this);
        alertDialogBuilder.setTitle(studentname);

        try {
            // setup a dialog window
            alertDialogBuilder.setCancelable(false)
                    .setPositiveButton("Absent", (dialog, id) -> {
                            Att2 a = new Att2();

                            a.setAttendanceClass(classid);
                            a.setAttendanceStudent(studentname);
                            a.setAttendanceDate(tfDate.getText().toString());
                            a.setAttendanceStatus("Absent");

                            studhelper.updateAttendance(a);
                            populateList();
                    })
                    .setNegativeButton("Present",
                            (dialog, id) -> {
                                    Att2 a = new Att2();

                                    a.setAttendanceClass(classid);
                                    a.setAttendanceStudent(studentname);
                                    a.setAttendanceDate(tfDate.getText().toString());
                                    a.setAttendanceStatus("Present");

                                    studhelper.updateAttendance(a);
                                    populateList();
```

```
//ATTENDANCE
public void updateAttendance(Att2 a){
    db = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    String whereClause = "attendance_date = '" + a.getAttendanceDate() + "' and attendance_student = '" + a.getAttendanceStudent() + "'";
    values.put(COLUMN_ASTATUS, a.getAttendanceStatus());

    db.update(TABLE_ATTENDANCE, values, whereClause, null);
    db.close();
}
```

Figure 10.1 Code for updating a student

The program checks if the user tapped or pressed a student from the List. A Dialog Box then appears which holds the Present and Absent choices for the user. Once the user selects a status, it then updates it in the database using the updateAtendance() method, which selects the specific column where the attendance_date and attendance_student values are from the values retrieved upon tapping a student's name.

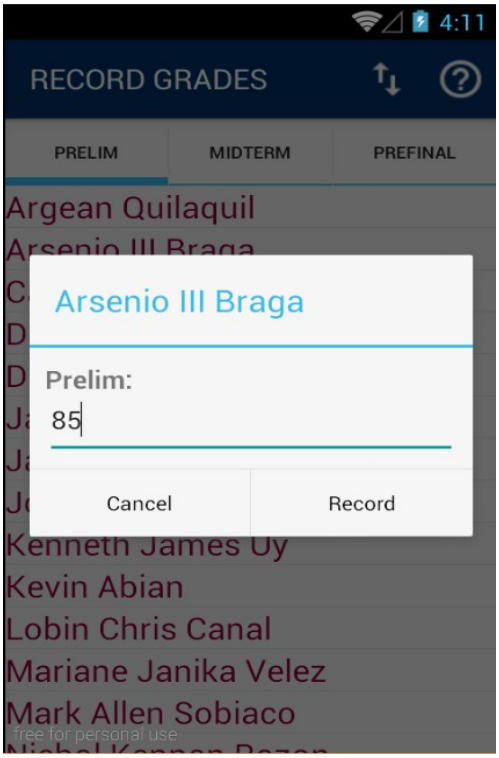
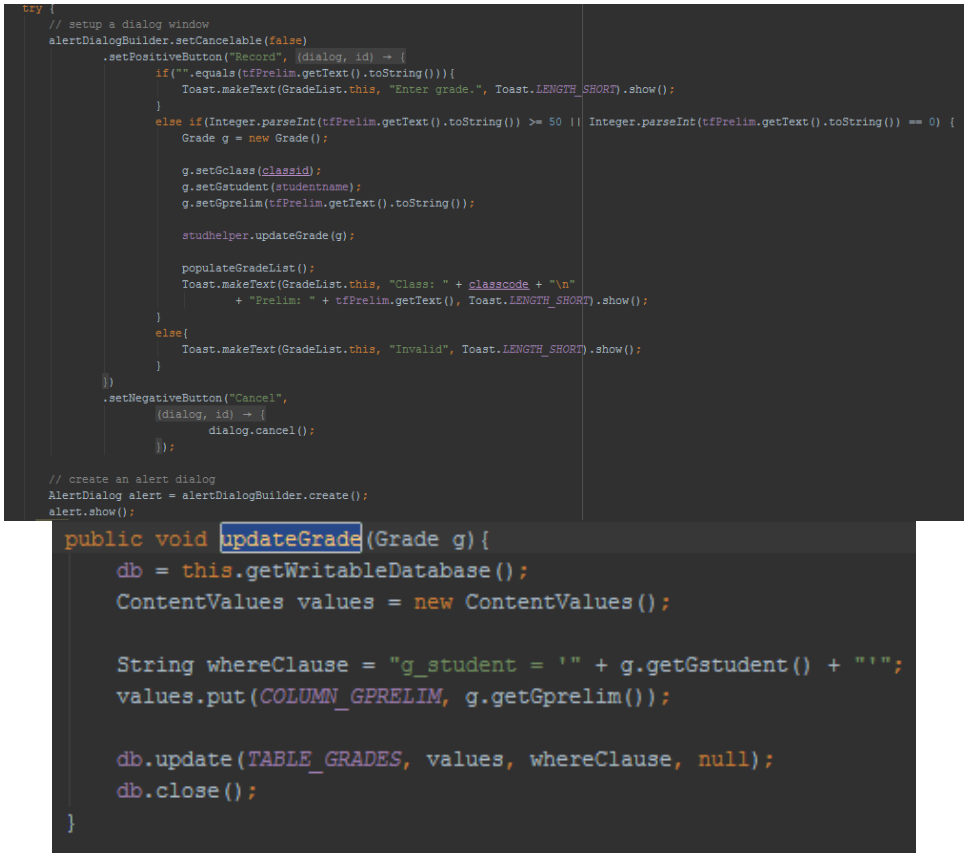


Figure 11. Record Grades form



```
try {
    // setup a dialog window
    alertDialogBuilder.setCancelable(false)
            .setPositiveButton("Record", (dialog, id) -> {
                if("".equals(tfPrelim.getText().toString())){
                    Toast.makeText(GradeList.this, "Enter grade.", Toast.LENGTH_SHORT).show();
                }
                else if(Integer.parseInt(tfPrelim.getText().toString()) >= 50 || Integer.parseInt(tfPrelim.getText().toString()) == 0) {
                    Grade g = new Grade();

                    g.setGclass(classid);
                    g.setGstudent(studentname);
                    g.setGprelim(tfPrelim.getText().toString());

                    studhelper.updateGrade(g);

                    populateGradeList();
                    Toast.makeText(GradeList.this, "Class: " + classcode + "\n"
                            + "Prelim: " + tfPrelim.getText(), Toast.LENGTH_SHORT).show();
                }
                else{
                    Toast.makeText(GradeList.this, "Invalid", Toast.LENGTH_SHORT).show();
                }
            })
            .setNegativeButton("Cancel",
                    (dialog, id) -> {
                        dialog.cancel();
                    });

    // create an alert dialog
    AlertDialog alert = alertDialogBuilder.create();
    alert.show();
```

```
public void updateGrade(Grade g){
    db = this.getWritableDatabase();
    ContentValues values = new ContentValues();

    String whereClause = "g_student = '" + g.getGstudent() + "'";
    values.put(COLUMN_GPRELIM, g.getGprelim());

    db.update(TABLE_GRADES, values, whereClause, null);
    db.close();
}
```

Figure 11.1 Code for recording grades

The program checks if the user tapped or pressed a student's name. An Input Box appears which asks for the grade input from the user. The program then checks if user inputted a grade or not. The program also checks if the user inputted a grade that is 0 or within the range of 50-100. It then upgrades the grade from the database using the updateGrade() method which selects the specific row in the database where the g_student is equal to the student selected from the list. If it isn't within the range, it rejects the input and prompts the user.

```
case requestcode:
    String filepath = data.getData().getPath();
    studhelper = new DBHelper(getApplicationContext());
    SQLiteDatabase db = studhelper.getWritableDatabase();
    String gradesTable = "grades";
    db.execSQL("delete from " + gradesTable + " where g_class = '" + tfid.getText().toString()  + "'");
    try {
        if (resultCode == RESULT_OK) {
            try {
                FileReader file = new FileReader(filepath);
                BufferedReader buffer = new BufferedReader(file);
                ContentValues contentValues = new ContentValues();
                String line = "";
                db.beginTransaction();
                while ((line = buffer.readLine()) != null) {
                    String[] str = line.split(",", 6);
                    String name = str[1].toString();
                    String classid = str[2].toString();
                    String prelim = str[3].toString();
                    String midterm = str[4].toString();
                    String prefinal = str[5].toString();
                    contentValues.put("g_student", name);
                    contentValues.put("g_class", classid);
                    contentValues.put("g_prelim", prelim);
                    contentValues.put("g_midterm", midterm);
                    contentValues.put("g_prefinal", prefinal);
                    db.insert(gradesTable, null, contentValues);
                    populateGradeList();
                    populateMidGradeList();
                    populatePrefGradeList();
                }
                db.setTransactionSuccessful();
```

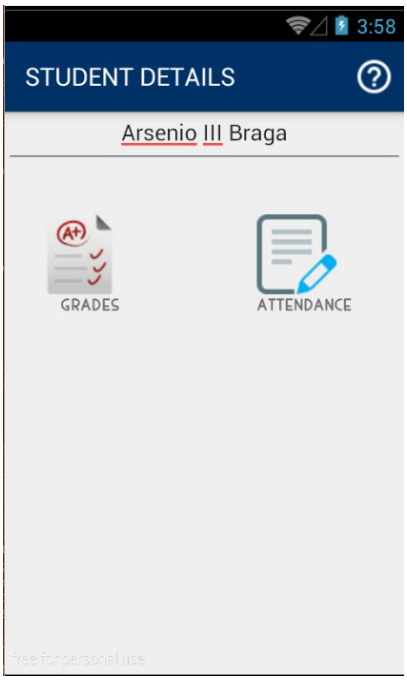
Figure 11.2 Code for importing grades

The first part starts a file picker using the Intent class. If there isn't a suitable file picker for the application present in your phone, it cancels the process. But if there is a suitable file picker, it proceeds to the requestcode case which accesses the database and deletes the files present in the database if the data imported is already present to avoid duplication. It then tries to read the file by line and stored in the ContentValues class in Android. The program continues to read the file until it is equal to null. Each line in the file are separated by a comma and then stored to the database from the ContentValues class and inserted using the getWritableDatabase() method in Android.

```
SQLiteDatabase sqldb = studhelper.getReadableDatabase(); //My Database class
Cursor c = null;
try {
    c = sqldb.rawQuery("select * from grades where g_student != 'Select student:' and g_class = '" + tfid.getText().toString() + "'", null);
    int rowcount = 0;
    int colcount = 0;
    File sdCardDir = Environment.getExternalStorageDirectory();
    String filename = tvCode.getText().toString()  + "Grades_Backup.csv";
    // the name of the file to export with
    File saveFile = new File(sdCardDir, filename);
    FileWriter fw = new FileWriter(saveFile);
    BufferedWriter bw = new BufferedWriter(fw);
    rowcount = c.getCount();
    colcount = c.getColumnCount();
    if (rowcount > 0) {
        c.moveToFirst();
        for (int x = 0; x < colcount; x++) {
            if (x != colcount - 1) {
                bw.write(c.getColumnName(x) + ",");
            } else {
                bw.write(c.getColumnName(x));
            }
        }
        bw.newLine();
        for (int x = 0; x < rowcount; x++) {
            c.moveToPosition(x);
            for (int j = 0; j < colcount; j++) {
                if (j != colcount - 1)
                    bw.write(c.getString(j) + ",");
                else
                    bw.write(c.getString(j));
            }
            bw.newLine();
         }
        bw.flush();
        Toast.makeText(GradeList.this, "Exported successfully.", Toast.LENGTH_SHORT).show();
    }
} catch (Exception ex) {
    if (sqldb.isOpen()) {
        sqldb.close();
        Toast.makeText(GradeList.this, "Complete all grades before exporting.", Toast.LENGTH_SHORT).show();
    }
} finally {
}
```

Figure 11.3 Code for exporting grades

The program first checks the query of the grades table that returns a specific grade list for a class. A variable is also declared to get the String value of the Class Code and concatenate it with “Grades_Backup.csv” for filename convention. A variable is also declared for where the file is saved. The program then checks if the rowcount > 0 and continues to check each row and column from the database to store it into the file. Each column is separed by a comma as it is being written in the file. When it is done, it ends the BufferedWriter class and shows a prompt that exporting is done. But if the program sees that not every student has a grade from Prelim to Prefinals, it won’t export and will show a prompt that the user must complete all grades before exporting.



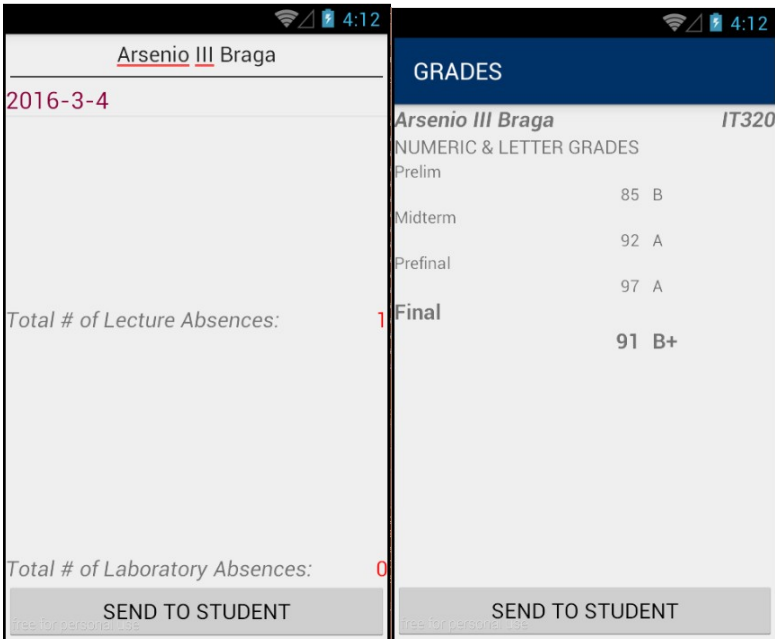


Figure 12. Students Report - Attendance Report and Grades Report

```
try {
    if ((Integer.parseInt(nPrelim.getText().toString()) >= 92)) {
        lPrelim.setText("A");
    } else if ((Integer.parseInt(nPrelim.getText().toString()) >= 88)) {
        lPrelim.setText("B+");
    } else if ((Integer.parseInt(nPrelim.getText().toString()) >= 84)) {
        lPrelim.setText("B");
    } else if ((Integer.parseInt(nPrelim.getText().toString()) >= 80)) {
        lPrelim.setText("C+");
    } else if ((Integer.parseInt(nPrelim.getText().toString()) >= 76)) {
        lPrelim.setText("C");
    } else if ((Integer.parseInt(nPrelim.getText().toString()) >= 72)) {
        lPrelim.setText("D");
    } else if ((Integer.parseInt(nPrelim.getText().toString()) >= 0)) {
        lPrelim.setText("F");
        lPrelim.setTextColor(Color.RED);
        nPrelim.setTextColor(Color.RED);
    } else if ((Integer.parseInt(nPrelim.getText().toString()) >= 101)) {
        Toast.makeText(GradeReport.this, "Invalid.", Toast.LENGTH_SHORT).show();
    }

prelimG = Integer.parseInt(nPrelim.getText().toString());
midtermG = Integer.parseInt(nMidterm.getText().toString());
prefinalG = Integer.parseInt(nPrefinal.getText().toString());

finalG = (prelimG + midtermG + prefinalG) / 3;
nFinal.setText(Integer.toString(finalG));

if (Integer.parseInt(nFinal.getText().toString()) >= 92) {
    lFinal.setText("A");
} else if ((Integer.parseInt(nFinal.getText().toString()) >= 88)) {
    lFinal.setText("B+");
} else if ((Integer.parseInt(nFinal.getText().toString()) >= 84)) {
    lFinal.setText("B");
} else if ((Integer.parseInt(nFinal.getText().toString()) >= 80)) {
    lFinal.setText("C+");
} else if ((Integer.parseInt(nFinal.getText().toString()) >= 76)) {
    lFinal.setText("C");
} else if ((Integer.parseInt(nFinal.getText().toString()) >= 72)) {
    lFinal.setText("D");
} else if ((Integer.parseInt(nFinal.getText().toString()) >= 0)) {
    lFinal.setText("F");
    lFinal.setTextColor(Color.RED);
    nFinal.setTextColor(Color.RED);
} else if ((Integer.parseInt(nFinal.getText().toString()) >= 101)) {
    Toast.makeText(GradeReport.this, "Invalid.", Toast.LENGTH_SHORT).show();
}
```

Figure 12.1 Code for converting numeric grade to letter grade

```
private void sendSms() {
    String SENT = "Message Sent.";
    String DELIVERED = "Message Delivered.";
    PendingIntent sentPI = PendingIntent.getBroadcast(this, 0, new Intent(SENT), 0);
    PendingIntent deliveredPI = PendingIntent.getBroadcast(this, 0, new Intent(DELIVERED), 0);

    registerReceiver((arg0, arg1) -> {
            switch (getResultCode()){
                case Activity.RESULT_OK:
                    Toast.makeText(GradeReport.this, "SMS sent", Toast.LENGTH_LONG).show();
                    break;
                case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                    Toast.makeText(GradeReport.this, "Generic Failure", Toast.LENGTH_LONG).show();//WRONG NUMBER
                    break;
                case SmsManager.RESULT_ERROR_NO_SERVICE:
                    Toast.makeText(GradeReport.this, "No service", Toast.LENGTH_LONG).show();
                    break;
            }
    }, new IntentFilter(SENT));
    registerReceiver((arg0, arg1) -> {
            switch (getResultCode()) {
                case Activity.RESULT_OK:
                    Toast.makeText(GradeReport.this, "SMS delivered", Toast.LENGTH_LONG).show();
                    break;
                case Activity.RESULT_CANCELED:
                    Toast.makeText(GradeReport.this, "SMS not delivered", Toast.LENGTH_LONG).show();
                    break;
            }
    }, new IntentFilter(DELIVERED));
    String no = tvNumber.getText().toString();
    SmsManager manager=SmsManager.getDefault();
    manager.sendTextMessage(no, null, tfMessage.getText().toString(), sentPI, deliveredPI);
}
```

Figure 12.2 Code for sending SMS – Same with Attendance

The program checks whether the SMS is successfully sent. First, it switches between the four cases namely, RESULT_OK, RESULT_ERROR_GENERIC_FAILURE, RESULT_ERROR_NO_SERVICE, and RESULT_CANCELED. If it proceeds without any problem, the SMS is sent to the student but if for example, the student has an invalid number, then it results to a generic failure. Also when there is no service present, a prompt will alert the user. The SmsManager class which holds the parameters for the number, message, and two Intents which holds for the message being sent and is successfully delivered.

6. CONCLUSION AND RECOMMENDATION

6.1 Conclusion

After deliberate research and testing, the researchers garnered the following conclusions:

First, the researchers found out that the application is useful to the teachers who tried using the proposed application.

Second, the application had a good review from those who participated in the survey, most of the participants agreed that the application makes classroom management easier

Third, the participants also had interacted with the application with ease, thus the application was easy to get used to using it. The tooltips that is added to the application help the participants in using the application

Last, using android and SMS technology in developing the application was quite useful in making classroom management work more efficient and faster.

6.2 Recommendation

After the survey the recommendations of the teachers are:

- An improved graphical user-interface
- An upgrade or edit option
- Include a seat plan for each laboratory present in the Computer Studies division
- Include a backup feature that connects to the Cloud
- Lecture seat plan shouldn't be limited to 40 students.

REFERENCES

Akhila, K., Prathyusha, B., PavanKumar, M., Amrutha, M. A Novel Approach of Mobile Based Student Attendance Tracking System Using Android Application. International Journal of Engineering Research & Technology, K.L. University. Retrieved January 28, 2015.
http://www.ijert.org/view-pdf/3230/a-novel-approach-of-mobile-based-student-attendance-tracking-system-using-android-application

Jessenth Ebenezer, P., Muralidharan, M.R., Srikanth, S., Ramesh, E., Prabhu, Mr. S. Android Application for Student Activity Register. International Journal of Research in Engineering & Advanced Technology. Department of Computer Science and Engineering, S.A. Engineering College. Retrieved February 10, 2015.
http://www.ijreat.org/Papers%202014/Issue8/IJREATV2I2090.pdf

Kumbhar, A., Wanjara, K., Trivedi, D., Kharatkar, A., Sharma, D. Automated Attendance Monitoring System using Android Platform. International Journal of Current Engineering and Technology, INPRESSCO. Department of Computer Engineering, K.J. Somaiya College of Engineering, Mumbai, India. Retrieved February 10, 2015.
http://inpressco.com/wp-content/uploads/2014/04/Paper1201096-1099.pdf

Shanbhag, G., Jivani, H., Shahi, S. Mobile Based Attendance Marking System Using Android and Biometrics. International Journal for Innovative Research in Science & Technology. Department of Computer Rajiv Gandhi Institute of Technology, Mumbai. Retrieved March 2, 2015.
http://www.ijirst.org/articles/IJIRSTV1I1027.pdf

Ph. D. Martinez, C. The Importance of Evaluation. Guide Star. Retrieved March 7, 2015.
http://www.guidestar.org/rxa/news/articles/2005/importance-of-evaluation.aspx

SQLite. Retrieved March 7, 2015.
http://www.sqlite.org/about.html

Android, the world's most popular mobile platform. Android Developers. Retrieved February 18, 2015.
http://developer.android.com/about/index.html

Java (programming language). Retrieved March 4, 2015.
http://en.wikipedia.org/wiki/Java_%28programming_language%29

TeacherKit. Retrieved March 7, 2015.
http://www.teacherkit.net/

Android Studio Overview. Retrieved September 22, 2015.
http://developer.android.com/tools/studio/index.html