

ATTENDANCE TRACKING SYSTEM – ANDROID APPLICATION

A Project

Presented to the faculty of the Department of Computer Science

California State University, Sacramento

Submitted in partial satisfaction of  
the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Ratna Mehta

FALL  
2014

# ATTENDANCE TRACKING SYSTEM – ANDROID APPLICATION

A Project

by

Ratna Mehta

Approved by:

\_\_\_\_\_, Committee Chair  
Dr. Jinsong Ouyang

\_\_\_\_\_, Second Reader  
Dr. Ying Jin

\_\_\_\_\_  
Date

Student: Ratna Mehta

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

\_\_\_\_\_, Graduate Coordinator  
Dr. Nikrouz Faroughi

\_\_\_\_\_  
Date

Department of Computer Science

Abstract  
of  
ATTENDANCE TRACKING SYSTEM – ANDROID APPLICATION  
by  
Ratna Mehta

Taking attendance in a class can often be a time consuming and a manual process, which is prone to human error(s) and hence recording incorrect data. Also, querying the data per student can be the tedious process since it involves some sort of manual tracking/counting of days attended. With advent of smartphones and tablets which are very handy to use, this process can be made completely automated and error free by using the right technologies.

For this project, I am creating an android based application that can be used for Attendance tracking which is easy to use and free from any manual tasks. Users (Professor/Course Instructor) just have to add courses he/she plans to take for Fall/Spring/Summer semester in the application. For each class that he conducts, students will just have to scan their student ID badge and the application will capture their unique student id from the badge using OCR (Optical Character Recognition) technique via camera of the device. Once the student Id is recorded, application will update the appropriate attendance database for that particular course.

At any time, User can query the database to look at attendance of students for any particular course or look at attendance record for any student.

An application like this is not only guaranteed to be accurate but also extremely easy to use now that smartphones/tablets are used in common.

\_\_\_\_\_, Committee Chair  
Dr. Jinsong Ouyang

\_\_\_\_\_  
Date

## Acknowledgements

I would like to thank Dr. Jinsong Ouyang, my advisor for providing me an opportunity to work on this project, which significantly broadened my knowledge on Android Development. I thank him for continuously providing the feedback, help and support to complete the project successfully as I was completely new to android development when I started working on this idea

In addition, I would like to thank Dr. Ying Jin for her willingness to serve on the committee.

Lastly, I would like to thank the entire faculty and staff of the Department of Computer Science Engineering at California State University, Sacramento.

## TABLE OF CONTENTS

	Page
Acknowledgements.....	vi
List of Figures.....	x
Chapter	
1 INTRODUCTION .....	1
2 PROJECT REQUIREMENTS.....	2
2.1 Add/Edit Course.....	2
2.2 Populate Registered Students.....	3
2.3 Take Attendance .....	4
2.3.1 Using OCR/Student ID .....	4
2.3.2 Drop Down List .....	5
2.3.3 Manual Entry .....	6
2.4.1 View Entire Class Attendance .....	7
2.4.2 View Students Below Minimum Attendance Criteria .....	7
2.4.3 View Pie Chart.....	8
3 ANDROID DEVELOPMENT BASICS.....	9
3.1 Types of Application Components .....	10
3.1.1 Activities.....	10

3.1.2 Services.....	13
3.1.3 Content Providers .....	15
3.1.4 Broadcast Receivers.....	15
3.2 Android Application Files.....	16
3.2.1 Java Files .....	16
3.2.2 Layout Files .....	16
3.2.3 Manifest File.....	18
3.3 Storage Options in Android Operating System.....	20
3.3.1 Shared Preferences.....	20
3.3.2 SQLite Database .....	20
3.3.3 Internal Storage.....	21
3.3.4 External Storage.....	21
3.4 Tools Required.....	21
3.5 Creating Android Applications .....	22
3.5.1 Creating Android Applications.....	22
3.6 Executing Android Applications.....	25
3.6.1 Emulator .....	25
3.6.2 Physical Device.....	26
4 FEATURES IMPLEMENTATION.....	27



4.1 Application Home Screen .....	27
4.2 Manage Courses.....	28
4.2.1 Add Course Activity .....	28
4.2.2 Download Excel to Populate Student Directory for Every Course.....	31
4.3 Tracking Attendance.....	34
4.3.1 Optical Character Recognition (OCR Technology).....	34
4.3.2 Spinner Drop Down.....	38
4.3.3 Manually Adding the Student ID.....	40
4.4 View Attendance.....	42
4.4.1 View all Students Attendance.....	42
4.4.2 View Student Attendance by Absent Criteria.....	45
4.4.3 View Pie Chart.....	48
4.5 Enhanced OCR Functionality .....	51
4.5.1 Camera Class .....	52
4.5.2 Capture Activity.....	56
4.6 Future Work.....	60
5 CONCLUSION.....	62
References.....	62

## List of Figures

Figures	Page
Figure 2.1 Professor Use Case Diagram .....	2
Figure 2.2 Add/Edit Course Flow Diagram .....	3
Figure 2.3 Add Register Students Per Class .....	4
Figure 2.4 Mark Attendances via OCR.....	5
Figure 2.5 Mark Attendance via Drop-Down List .....	6
Figure 2.6 Mark Attendance Manually .....	6
Figure 2.7 View Course Attendances .....	7
Figure 2.8 View Students Below Minimum Attendance Criteria .....	8
Figure 2.9 Display % Attendance in Pie Chart .....	8
Figure 3.1 Android Lifecycle for Activity .....	12
Figure 3.2 Android Lifecycle for Services.....	14
Figure 3.3 Creating Android Application .....	23
Figure 4.1 Application Home Screen.....	27
Figure 4.2 Add Course Details.....	29
Figure 4.3 Download Excel Spreadsheet .....	32
Figure 4.4 Student ID Captured via OCR .....	38
Figure 4.5 Taking Students Attendance using Spinner .....	39
Figure 4.6 View All Student Attendance Details.....	43
Figure 4.7 View Student Attendance Details by Selecting Percentage Absent Cutoff.....	46
Figure 4.8 View Pie Chart .....	49

## Chapter 1

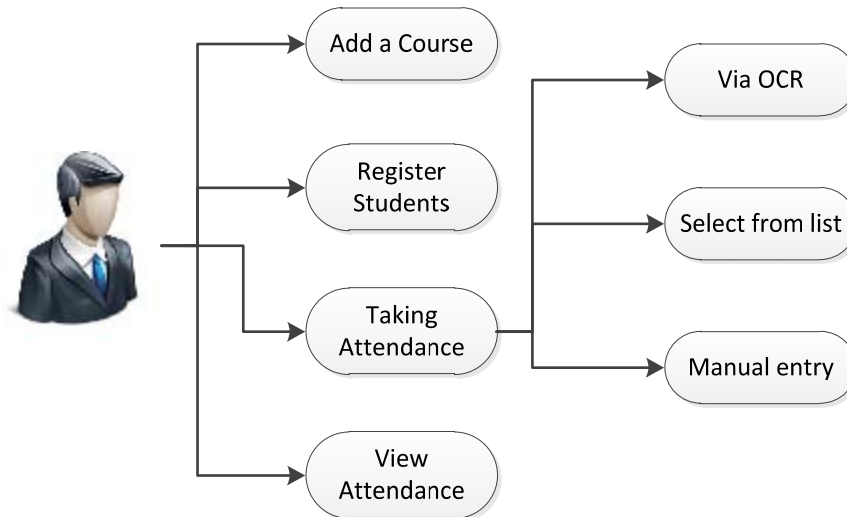
### INTRODUCTION

Taking attendance in a class can be time consuming and manual process for professor, which is prone to manual error while recording student's presence. This application will allow professor to scan student's id card and application will extract student id number from the scanned picture via OCR mechanism and record students' presence. If a student forgot his id card, professor can easily register him by selecting his name from drop list. At any time, professor can look at his class's attendance which is just few button clicks away and is guaranteed to be accurate. Professor can also get a list of students who did not meet the minimum percentage criteria (e.g. more than 5% OR 10% absent) for attendance.

## Chapter 2

## PROJECT REQUIREMENTS

Following figure shows different options available to user (professor).



**Figure 2.1 Professor Use Case Diagram**

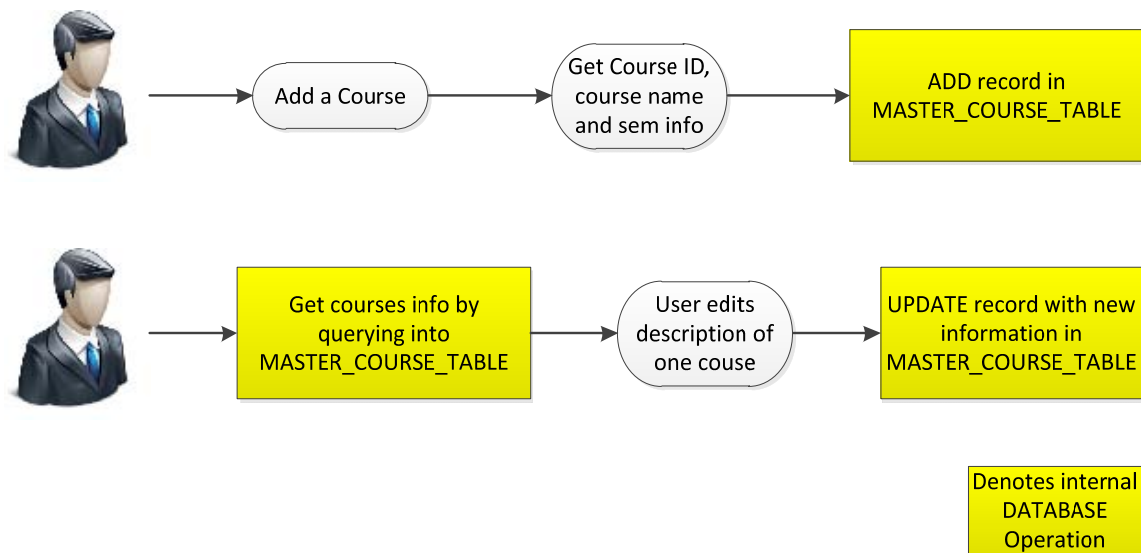
### 2.1 Add/Edit Course

As a user professor can add course that he plans to teach for a particular semester. To do that, user will click on “Manage Courses” → “Add a course” which would open up “Add New Course” form. User enters details like course id, course name and semester for which the course is enrolled. On adding a course, all the details are added to MASTER\_COURSE\_TABLE.

User can also edit the details of already added course. To do that, clicking on “Manage Courses” will list all the courses that have been added. Click on pencil icon of the course

that user wants to edit. It will open up edit form where user can update the name/description and save it, which will do an update to MASTER\_COURSE\_TABLE.

Following figure shows internal operation on add/manage course flow.

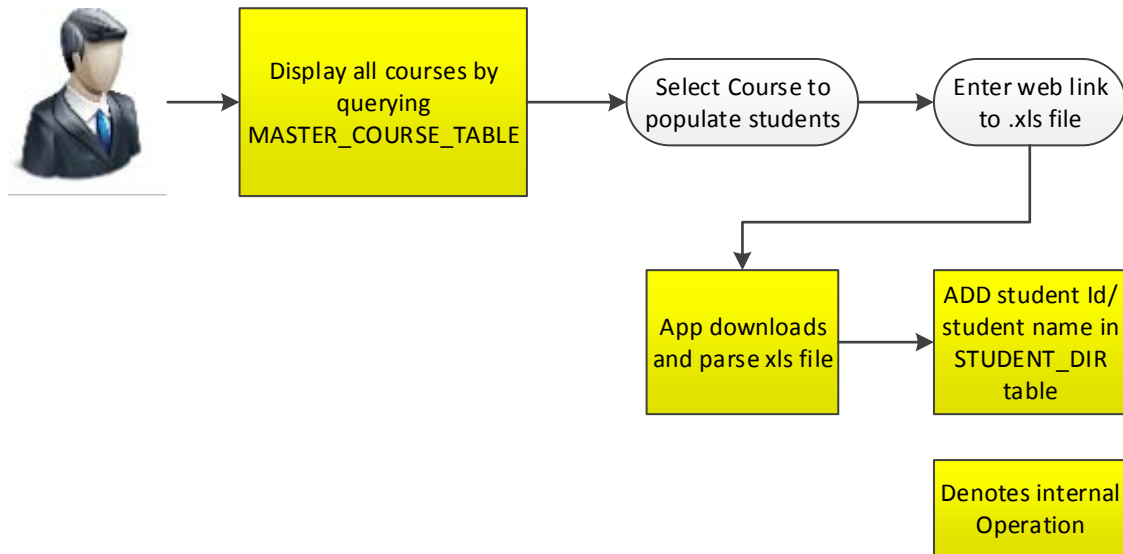


**Figure 2.2 Add/Edit Course Flow Diagram**

## 2.2 Populate Registered Students

Since attendance will be taken using student id number, from usability perspective it is much more useful if there is a mapping of student id to student name. Once the courses are added, user (prof) has an option to add details of students who are registered for a particular course. To do that, user has to provide a web link to excel file which contains list of students and their student id. Application will download the file from the web link, parse the excel file, and add student id and student name to MASTER\_STUDENT\_DIR table. Every time when a student's attendance is taken, look up of student name is done

based on student id that is captured via OCR. This mechanism can be used to detect scenario when un-registered student id has scanned his ID card.



**Figure 2.3 Add Register Students Per Class**

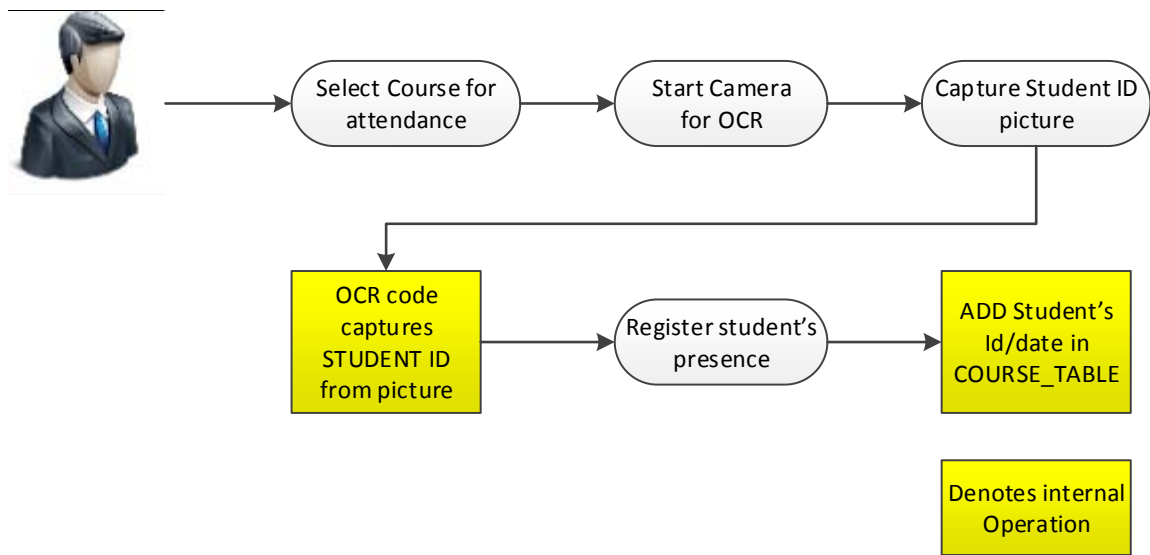
## 2.3 Take Attendance

This section talks about various ways of taking attendance of students during a regular class schedule. Use case is – when student enters the classroom, professor can register student’s presence in following 3 ways.

### 2.3.1 Using OCR/Student ID

OCR (Optical Character Recognition) is electronic conversion of images or printed text into machine-encoded text. This application uses OCR technique to scan the student ID card. Upon scanning, it captures the text i.e. student id number from the captured image. As a user, professor just needs to select course for which attendance is being taken and then scan student id. On successful scanning, Student ID form opens up that displays

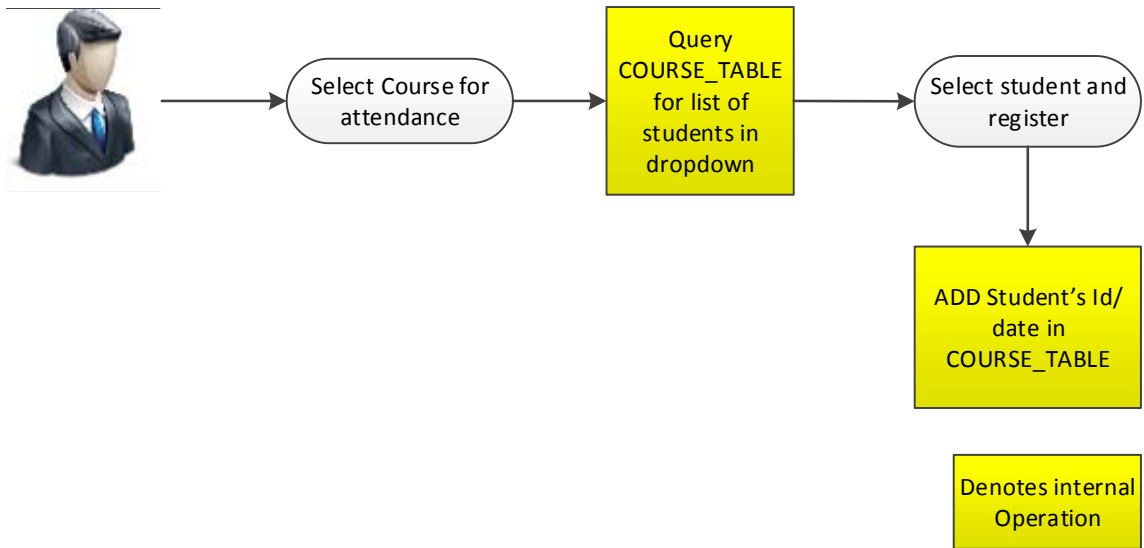
student id number, and clicking “Add student” will register student’s attendance in respective course’s database by adding a record in COURSE\_TABLE. Application will also store current date in order to track days on which student was present.



**Figure 2.4 Mark Attendances via OCR**

### 2.3.2 Drop Down List

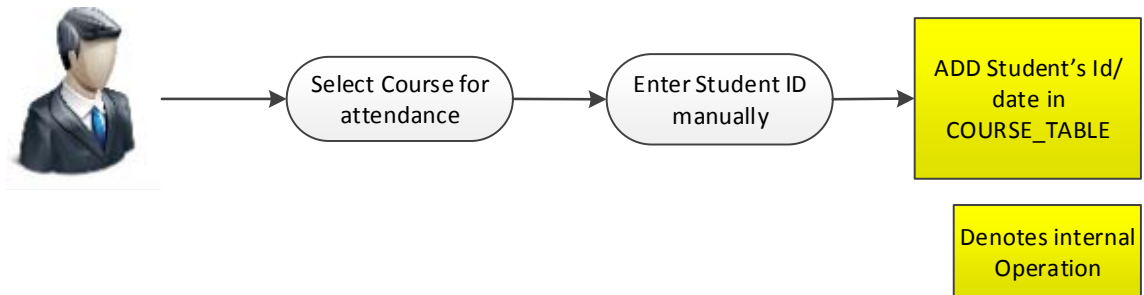
Another use case is when student forgets his student id card. In this case, instead of scanning id card professor can select student from drop down list. This list is populated depending on the course selected and hence displays only those students registered for this particular course. Upon selecting appropriate student name, student is marked present and record is added in COURSE\_TABLE.



**Figure 2.5 Mark Attendance via Drop-Down List**

### 2.3.3 Manual Entry

Last use case of taking attendance is manual entry of student id. In this case, professor will enter student's id manually and add him. This is an exceptional case when camera of device is not working and student is not yet registered for the course.



**Figure 2.6 Mark Attendance Manually**

### 2.4 View Attendance



Viewing attendance basically allows professor to look at attendance record of student – i.e. of total number of days classes were conducted, how many days was a student absent; and if absent, dates on which he was marked absent.

### 2.4.1 View Entire Class Attendance

This option allows professor to view the attendance of a selected course – displaying attendance of all the students including the absent days for students.

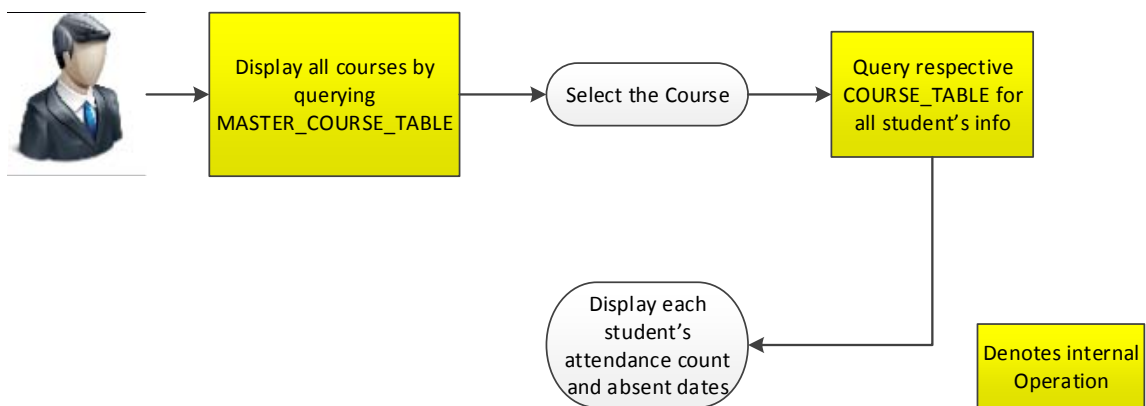
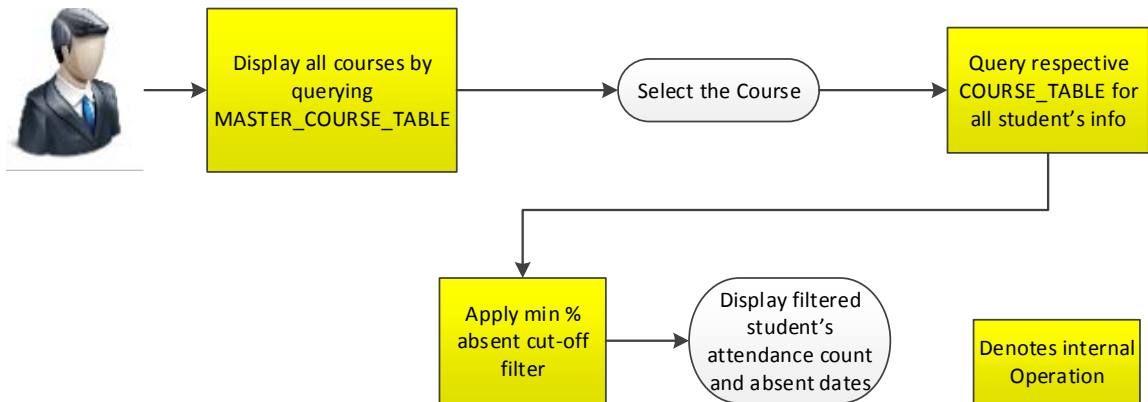


Figure 2.7 View Course Attendances

### 2.4.2 View Students Below Minimum Attendance Criteria

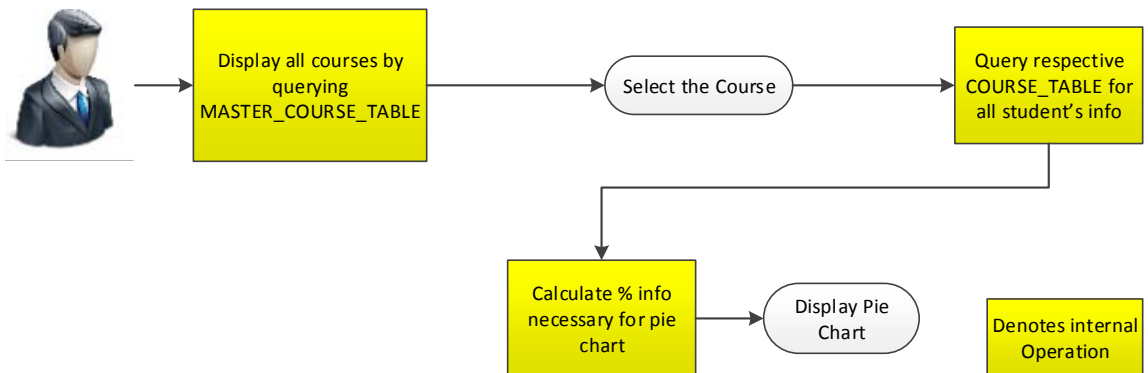
Using this feature, professor can look filter out students whose absent percentage was above the selected threshold value. For that, user has to select threshold cut off value like 5%, 10%, 15% etc from the drop down (spinner) list.



**Figure 2.8 View Students Below Minimum Attendance Criteria**

### 2.4.3 View Pie Chart

This feature allows user to view % of students categorized in 100%, 95 – 100%, 90 – 95% and less than 90% attendance bucket and displayed in form of a pie chart.



**Figure 2.9 Display % Attendance in Pie Chart**

## Chapter 3

### ANDROID DEVELOPMENT BASICS

Android is a Linux based Operating System that was developed by Android Inc., which was bought by Google later on. Android OS is designed mainly for smart phones and tablets that have touch screens/motion detectors. Despite being primarily designed for touchscreen input, it also has been used in game consoles, digital cameras, and other electronics. To give competition to Apple's iOS, Google made Android as open source and releases code under Apache License. In last few years, Android has become the most widely used operating system in the market and having open source has just increased its popularity many folds.

Android code is written primarily in Java programming language, and is compiled with the help of Android SDK tools. On compiling, it generates an Android package (commonly known as .apk) file which is used to install the application on android device.

App Components:

To aid in android development, there are mainly four types of components. Each component serves a distinct purpose and allows system to interact with your application in different ways. Broadly speaking, there are four types of components [1]

- Activity
- Services
- Content provider
- Broadcast receivers

### **3.1 Types of Application Components**

Android helps to create applications using these components. This will help to understand how one can build the components which will be the building blocks of the application.

#### **3.1.1 Activities**

An activity represents a single screen with a user interface. For example, in this Attendance Tracker application, activities include adding a course, scanning student to register presence or viewing student's attendance record. "Main" activity is the activity that is launched when any application starts. Different activities can start from one activity, keeping the previous activity in stack although the activity is stopped. Activities follow LIFO (Last In First Out) mechanism.

Following are the callback methods available in activity to perform necessary tasks [1].

`onCreate()`: This is called when the activity is first created. This is always followed by `onStart()`: This is called just before the activity becomes visible to the user. This is followed by `onResume()` if the activity comes to the foreground, or `onStop()` if it becomes hidden.

`OnResume()`: This is called just before the activity starts interacting with the user. This is always followed by `onPause()`.

`onPause()`: This is called when the system is about to start resuming another activity. And this is followed either by `onResume()` if the activity returns back to the front, or by `onStop()` if it becomes invisible to the user.

`onStop()`: This is called when the activity is no longer visible to the user. This is followed either by `onRestart()` if the activity is coming back to interact with the user, or by `onDestroy()` if this activity is going away

The Android Life cycle [1] for Activity looks like :

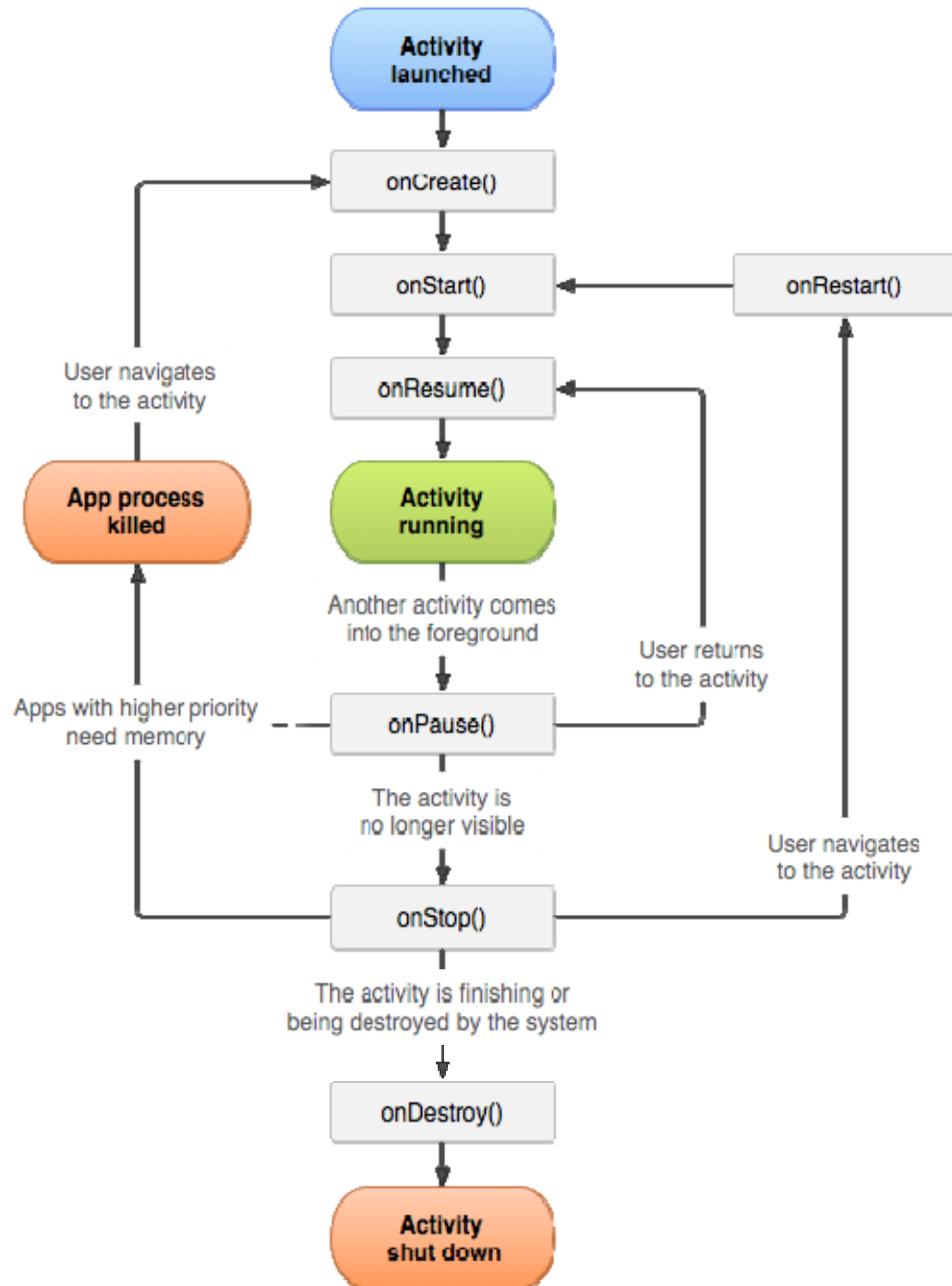


Figure 3.1 Android Lifecycle for Activity

`onRestart()`: This is called after the activity has been stopped, just prior to it being started again. This is always followed by `onStart()`.

`OnDestroy()`: This is called before the activity is destroyed. This is followed by nothing.

The life cycle of an activity can be described diagrammatically as Figure 3.1.

### **3.1.2 Services**

Services are the component in Android Operating System which runs in background. Hence, all the background tasks are done using the services component. This component is useful especially when a task is to be performed without impacting the user of its operation. This component does not provide any User Interface. For example, downloading a file from internet or loading image. All the services that are created in the application have to inherit the Service class provided in Android Operating System.

The diagrammatic representation of the life cycle of Services [1] can be found in Figure 3.2 as below

Service component has two forms [1]:

- (1) Started: When an activity starts `startService()`, then a service is "started". A service can run in the background indefinitely once started, even if the component that started it is destroyed. For example, it might download a image over the network. When the download is completed, the service should stop itself

(2) Bound: When an application component calls `bindService()`, a service is "bound".

A bound service provides a client-server interface that allows components to interact with the service, send requests and get results.

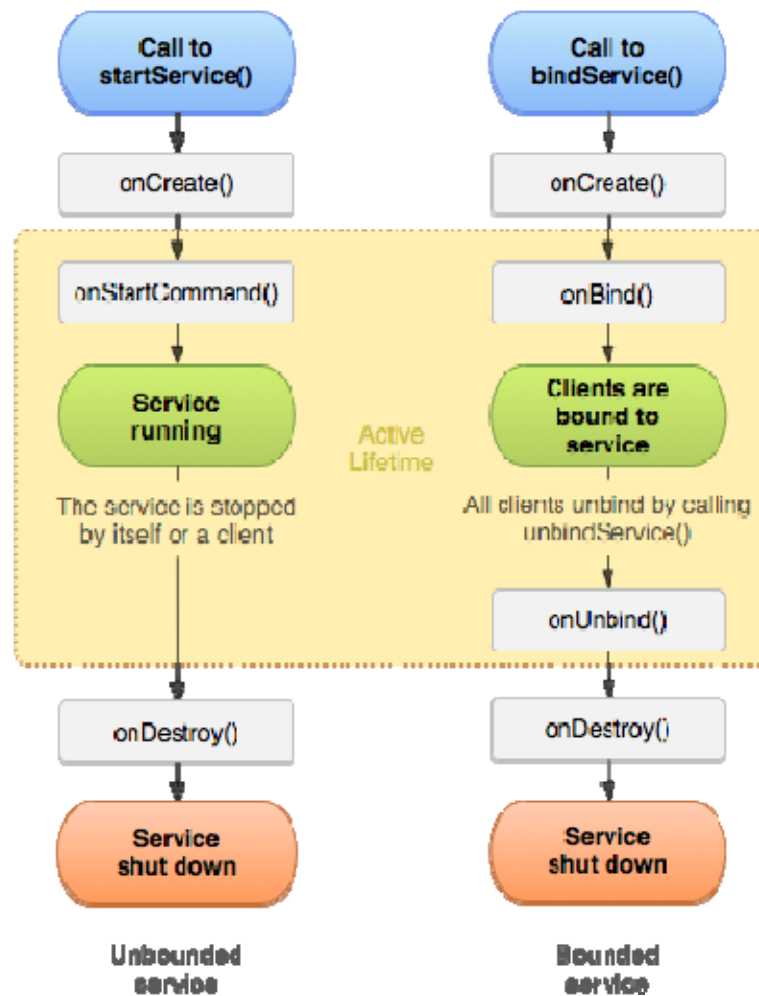


Figure 3.2 Android Lifecycle for Services



### **3.1.3 Content Providers**

This component manages the shared set of application data i.e. it provides a mechanism through which the data stored by one android application can be made accessible to other application. The application can store the data anywhere from SQLite Database to web to any persistent storage location. Through the content providers in the android operating system, the data stored can be modified, queried or even store the newly entered data in the application. This component is useful to write or read the data where the user cannot access (the protected storage on the data).

A content provider is implemented as a subclass of `ContentProvider` and an application accesses the data from a content provider with a `ContentResolver` client object. In order to insert data into provider, `ContentResolver.insert()` method is used. This method inserts a new row into the provider and returns content URI for that row. In order to update a row or delete a row `ContentResolver.update()` and `ContentResolver.delete()` methods can be used respectively [1].

### **3.1.4 Broadcast Receivers**

A broadcast receiver allows registering for system or application events. When any registered event occurs, receivers for an event will be notified by the Android runtime. System broadcast include screen turning off, the battery is low, or a picture was captured. Applications broadcast would include letting other applications know that some data has been downloaded to the device and is available for them to use.

The implementing class for a *receiver* extends the BroadcastReceiver class. If the event for which the broadcast receiver has registered happens the onReceive() method of the receiver is called by the Android system.

### **3.2 Android Application Files**

Files can be broadly divided into three categories [1].

- Java file
- Layout file
- Manifest file

#### **3.2.1 Java Files**

These files are the files where all the processing of the events happens, and allows user to interact with the system. These files are the heart of the android application. This is the place where onCreate(), onStart(), onPause(), onStop() etc. methods are defined. This part is responsible for getting the user inputs and processes the activities accordingly. Through these files, the layouts can be added dynamically, the user entered values in the text boxes or other input can be obtained and stored.

#### **3.2.2 Layout Files**

A layout defines the visual structure for a user interface, such as the UI for any activity. These files are responsible for defining the user Input. The Android framework gives you the flexibility to use either or both of these methods for declaring and managing your

application's UI. The user has the luxury of seeing how the designed layout will look like in the Graphical Mode and they have the option of selecting the device that they desire to view the layout. There are multiple interfaces that are possible in android. Few of them are enlisted below [1] –

- **Linear Layout** – Using this layout option, the developer can align all the components on the screen vertically or horizontally. This layout can be only in one direction. This layout is usually used when the position of the components are static and will not change during the course of the application,
- **Relative Layout** – Through this layout, the position of the components can be described in relation to other components. This is widely used as the position of the components may vary based on the screen resolution on different devices. This is recommended for the developers to ideally use for a large pool of devices due to varying screen size.
- **List View** – When there are multiple items to be displayed on the screen and they may not fit the size of the screen, then the List View is used. The items in the list are filled in the adapter from various sources like Database query, array etc.
- **Scroll View** – When the components are not fitting into physical display, then the scroll view layout is used to insert more components in the screen. A scroll view can be scrolled vertically only. If the developer wants to use the horizontal scroll, then there is another layout for android operating system – `HorizontalScrollView`. It is not recommended to use the `ListView` with the scroll view as the `ListView` takes care of scrolling on its own.

### 3.2.3 Manifest File

It is the main part of the android application. This file contains all the information about the application – what android operating system components are present in the application, what permissions are required by the application etc. For using any component, it has to be present in the manifest file otherwise that component or functionality will not be executed when the application is installed in the device.

The manifest file defines what version of the Android Operating System, the application is compatible with. It can be defined as –

```
<uses-sdk
    android:minSdkVersion="8"
    android:targetSdkVersion="18" />
```

The manifest file also defines what devices that the application can support. It can be defined as:

```
<supports-screens
    android:anyDensity="false"
    android:largeScreens="true"
    android:normalScreens="true"
    android:resizeable="true"
    android:smallScreens="true"
    android:xlargeScreens="false" />
```

The developer can define all the activities, their theme etc. in the manifest files as –

```

<activity
  android:name=".CaptureActivity"
  android:configChanges="orientation|keyboardHidden|screenSize"
  android:screenOrientation="Landscape"
  android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
  android:windowSoftInputMode="stateAlwaysHidden" >
</activity>

```

The permission that the application requires for the proper execution of the tasks/functionalities that are used in the application is declared in the manifest file. The sample of that can be found as:

```

<uses-permission android:name="android.permission.CAMERA" />
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

```

The broadcast receivers also have to be declared in the manifest file for the operating system recognition. They are declared in the manifest file as:

```

<receiver
  android:name="android.com.example.attendancetracker"
  android:label="@string/app_name" >
  <intent-filter>
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE"
  />
  </intent-filter>

  <meta-data
    android:name="android.appwidget.provider"
    android:resource="@xml/attendance_widget" />
</receiver>

```

### **3.3 Storage Options in Android Operating System**

While working with Android Operating System, there are two options where the user data can be stored. The options are [1] –

- Shared Preferences
- SQLite Database
- Internal Storage
- External Storage

The user has the option of using any of the options depending on the need of the application as well as the criticality of the data.

#### **3.3.1 Shared Preferences**

This mechanism stores and retrieves the data as key value pairs of the primitive data types like String, Integer (int) or Booleans. The data stored by this mechanism will be persisted across multiple sessions of the application. The data will remain as it is even if the application is killed several times.

#### **3.3.2 SQLite Database**

The operating system allows the developer to use the local databases. The databases created using the SQLite databases will be accessible to any class present in the application. These databases have to be accessed by their names. The data will be available across the application but not outside the application. For using this facility, the

developer has to create a subclass of SQLiteOpenHelper and override the onCreate method.

### **3.3.3 Internal Storage**

The operating system allows the data to be directly stored on the internal storage of the device. The data that is stored in the file is private to the application and the other applications that are present in the device cannot access the data stored. The user (owner) of the device cannot also access the data of the application. When the application is uninstalled, the files will be removed automatically from the internal storage of the device.

### **3.3.4 External Storage**

External storage can be any removable media like USB, SD Card etc. The data stored in this storage are public and can be accessed by anyone in the application. The user/application can read the data and have the option of modifying the data that is present in the file.

## **3.4 Tools Required**

Android applications can be developed across multiple IDE's and there are several plugins available for making the current IDE capable to write the android programs. Out of several IDE's, the popular IDE's are enlisted under –

- **Eclipse** – Any version of Eclipse can be made capable of writing the Android Applications after installing the Android Developer Tools plugin from Eclipse

Market. Once, this is installed, the user can go the File Menu and select Android Application as the new project.

- **Microsoft Visual Studio** (For mono-droid programs) – This is currently not very popular for cross platform IDE. This IDE is currently capable of writing android programs that are based on Mono-Droid.
- **Android Development Tool** – This IDE is provided by Google Inc. that has been configured by the developers at Google to write the Android applications in Eclipse. The primary IDE here is Eclipse.
- **Android Studio** – This is the IDE that is developed by Google Inc. Currently, it is still in the development mode and it was announced in April 2013. It is currently available for free. It is based on IntelliJ software. It is available for download across various operating systems like Windows, Mac, and Linux.

### **3.5 Creating Android Applications**

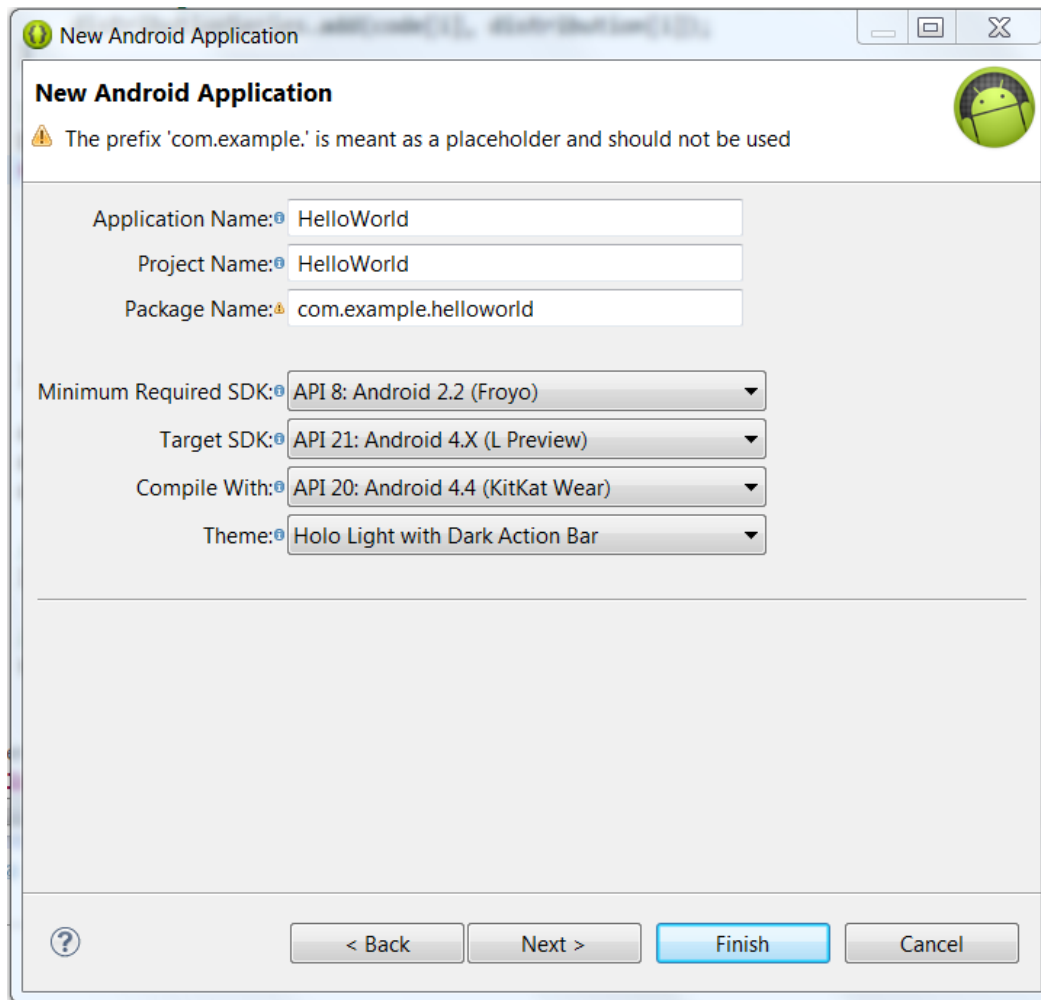
This section will help one understand how we can build the sample android application, how the code looks and the usage of the directory structure like layout, source etc. when new application is created.

#### **3.5.1 Creating Android Applications**

For creating the application, do the following in Eclipse.

File -> New -> Project. From the dialog that comes up, select Android -> Android Application Project. Enter the Application Name. Enter the minimum and Target SDK version (figure below). Clicking finish creates a new application.





**Figure 3.3 Creating Android Application**

Once the new application is created, the app by default stores hello world code and we need to just run it. The code snippet looks as:

```
package com.example.helloworld;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;

public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

The src directory contains all source files that are .java required for the application. The bin contains the files generated after the build is completed. It has the .apk file, which is the application binary that is installed on the device for the application to run. The res directory contains sub directories like drawable, layout, menu, values etc.

AndroidManifest.xml file is the manifest file for the application. It contains all the activities in the application, any permission needed and service defined [1].

### **3.6 Executing Android Applications**

Once the application has been developed sufficiently that it can be tested, then there are two options for the user to test the developed application. The user can tap on the run button in the IDE and the application will be loaded according to the options present at the run time. Those options are enlisted under –

- Emulator
- Physical Device

#### **3.6.1 Emulator**

The Android Development environment allows the developer to run the application through the emulator i.e. Android Virtual Device. Android Virtual Device can be created in Eclipse as follows:

Window-> Android Virtual Device Manager. In the dialog that opens, click ‘New’. This shows the AVD dialog where the developer needs to specify the name of the device, select Target, select Space and other configurations. After entering all the data, clicking on ‘Create’, creates a new AVD. After the AVD is created, clicking on ‘Start’ starts the emulator. The developer can pre-configure the AVD that is intended for the application to work with and then once it is ready then the user can just run the application on the emulator.

The IDE automatically installs the application on the device and will simulate the behavior of the physical device. This will help the developer in case they do not have a device that is running Android operating system.

### **3.6.2 Physical Device**

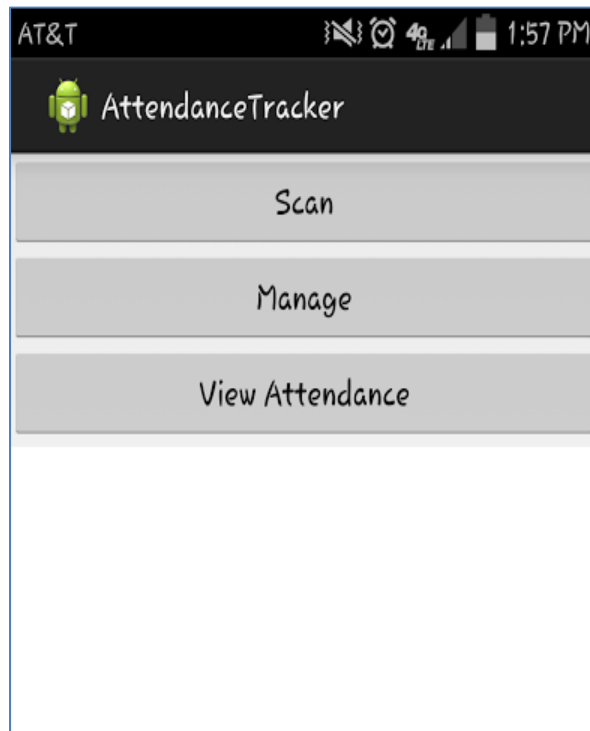
If the developer has the Android operating system device, then he/she can connect the device to the computer via USB cable. Once connected, the developer needs to enable USB debugging on the device. USB Debugging setting option can be found under Settings -> Developer options. On connecting the device, option for downloading USB driver for the device would appear if in case the driver is not already installed. After the device is connected to the computer, the user can run the application from the IDE and it will be installed on the device and application will open automatically. This option is faster as compared to the emulator in case the computer does not have enough resources to support emulator at high speed.

## Chapter 4

## FEATURES IMPLEMENTATION

**4.1 Application Home Screen**

This screen appears when the user opens the application. The screen looks as:



**Figure 4.1 Application Home Screen**

The professor can choose to do any activity from the available options. The professor can choose to add course for the semester and edit / delete the course details if required by tapping on “Manage” button. The professor can start the scan activity under which he / she can select the course for which attendance needs to be taken by tapping on the “Scan” button. After the course is selected, the student can scan their student Ids on which OCR Technology will be applied and the student Id will be retrieved which will be used for

taking the attendance. Later the professor can also start the View Attendance activity by tapping on the “View Attendance” where the professor will be able to take the look at the student attendance details for every course. The code snippet for Button OnClickListener looks as:

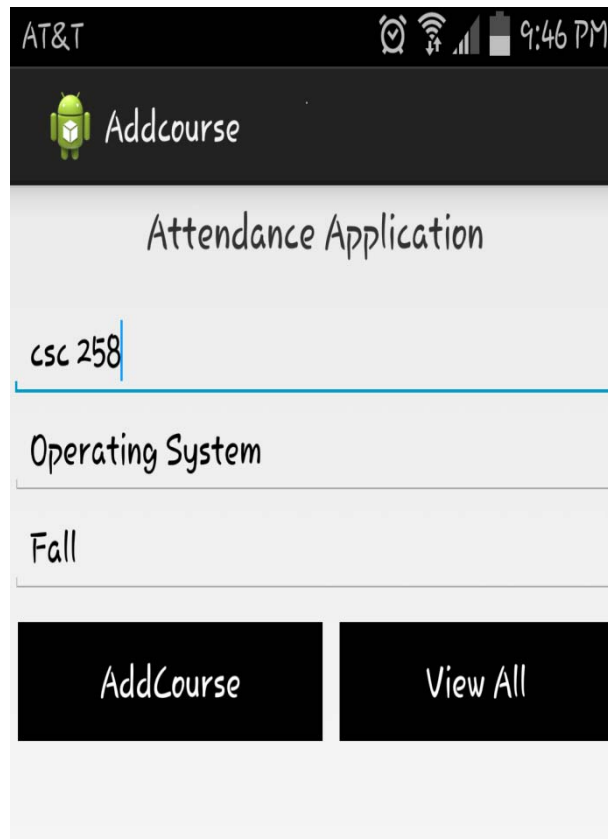
```
manage_button.setOnClickListener(new View.OnClickListener() {  
  
    @Override  
    public void onClick(View v) {  
        // TODO Auto-generated method stub  
        Intent addcourse = new  
        Intent(WelcomeForm.this,MainAddCourseForm.class);  
        addcourse.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP  
        | Intent.FLAG_ACTIVITY_NEW_TASK);  
  
        startActivity(addcourse);  
  
    }  
});
```

## 4.2 Manage Courses

Professor can add the course which he/ she is going to take for the semester in order to take the attendance, edit / delete the course details if required and then downloading the Excel spreadsheet from the server which contains the details (E.g. Student id, Student Name, Semester) of the students under Manage Activity.

### 4.2.1 Add Course Activity

Professor can add the course details for the semester in order to take the attendance by clicking on the manage course button/ Add Course. The screen looks as:



**Figure 4.2 Add Course Details**

When professor clicks on “AddCourse” button after filling out the details, `add_save_btn` function is being called which executes the method for adding the course details to the database. Initially all the course details entered by the professor is stored in temporary variables. Later the check is being made whether the course already exists in the database using function `Get_CourseNo_Count`. If course count is greater than 0 then professor will be prompted “Course Already exists” else the new course entry will be made in the

database using Add\_course function. The code snippet for adding course details in the database looks as:

```

add_save_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        valid_cnumber =
add_cnumber.getText().toString().toUpperCase().trim().replaceAll(" ", "");
        valid_coursename = add_coursename.getText().toString();
        valid_coursesem = add_coursesem.getText().toString().toUpperCase().replaceAll(" ", "");
        int data = 0;
        String dates = new SimpleDateFormat("yyyy-MM-dd").format(new Date());
        String firstdate = "";
        int crscount = dbHelper.Get_CourseNo_Count(valid_cnumber);
        if(crscount > 0 ){
            Toast_msg = "Course Already Exists";
            Show_Toast(Toast_msg);
        }else{
            // TODO Auto-generated method stub
            // check the value state is null or not
            if(valid_cnumber!=null && valid_coursename !=null &&
                valid_coursesem != null && valid_cnumber.length()!=0 &&
                valid_coursename.length()!=0 && valid_coursesem.length()!=0){


                dbHelper.Add_Course(new
                Course(valid_cnumber,valid_coursename,valid_coursesem,data,firstdate))
                ;

                Toast_msg = "Data inserted successfully ";
                Show_Toast(Toast_msg);
                Reset_Text();
            }
            else{
                Toast_msg = "Data missing";
                Show_Toast(Toast_msg); } } } } );

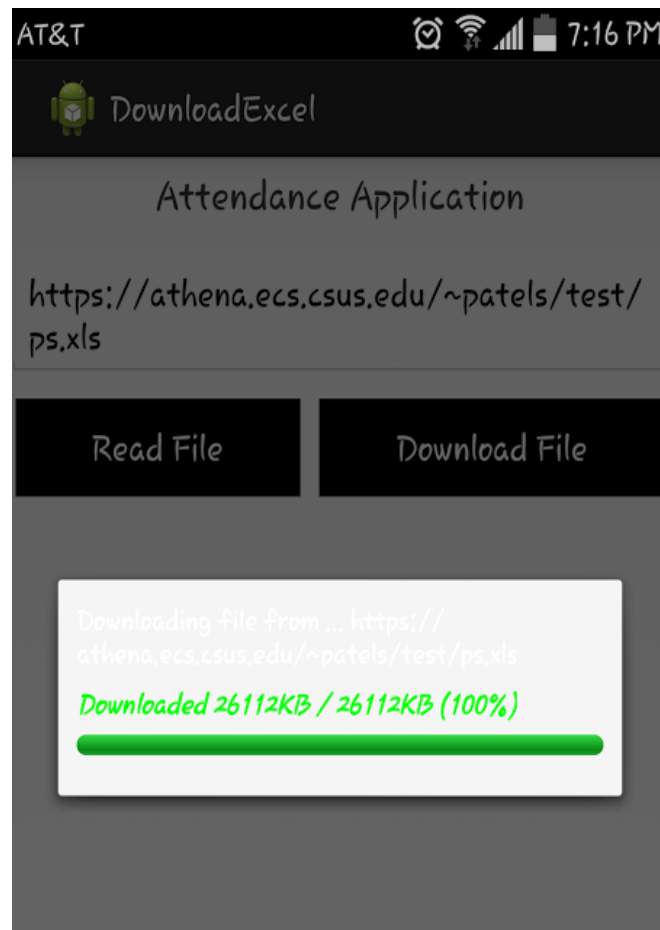
```



#### 4.2.2 Download Excel to Populate Student Directory for Every Course

Professor need to download Excel Spreadsheet by clicking on Excel Icon  corresponding to the course which contains student details (E.g. student ID and Student name). This data gets populated in the student directory table for every course.

When student attendance is taken, the Student ID captured using OCR Technology is compared with the student directory table to ensure student has registered for the class or not. Professor needs to enter the hyperlink in the Edit Text box provided, click on Download file button which will download the file from the server [6] [7]. After clicking on Read File button the application will read the contents of the excel file and store it in the respective student directory database. Apache POI jar file which is the Java API is being used for text extractions form different file formats. I have used this API to read the contents of the excel file and store the excel file contents to the respective student directory database [6] [7]. The screen looks as:



**Figure 4.3 Download Excel Spreadsheet**

On Clicking “Download file” button, downloadFile function is executed where the server path entered by the professor is passed as the argument to this function. Here, HttpURLConnection API is used for sending / receiving the data from the path provided which is stored in the variable download\_file\_path [7]. The Excel file is downloaded from the server and the contents of the file are stored on the SD Card under “ps.xls” file. When the file download is in progress, the progress bar will denote the status / progress

in percentage format for the file download [7]. The code snippet for downloading the file from the server looks as:

```

void downloadFile(String dwnload_file_path){

try {
URL url = new URL(dwnload_file_path);

URLConnection urlConnection = (URLConnection) url.openConnection();
urlConnection.setRequestMethod("GET");
urlConnection.setDoOutput(true);

//connect
urlConnection.connect();

//set the path where we want to save the file
File SDCardRoot = Environment.getExternalStorageDirectory();
//create a new file, to save the downloaded file
File file = new File(SDCardRoot,"ps.xls");

FileOutputStream fileOutput = new FileOutputStream(file);

//Stream used for reading the data from the internet
InputStream inputStream = urlConnection.getInputStream();

//this is the total size of the file which we are downloading
totalSize = urlConnection.getContentLength();

runOnUiThread(new Runnable() {
public void run() {
pb.setMax(totalSize);
}
});
//create a buffer...
byte[] buffer = new byte[1024];
int bufferLength = 0;

while ( (bufferLength = inputStream.read(buffer)) > 0 ) {
fileOutput.write(buffer, 0, bufferLength);
downloadedSize += bufferLength;
// update the progressbar //
runOnUiThread(new Runnable() {
public void run() {
pb.setProgress(downloadedSize);
float per = ((float)downloadedSize/totalSize) * 100;
cur_val.setText("Downloaded " + downloadedSize + "KB / " + totalSize + "KB ("
+ (int)per + "%) " );
}
});}fileOutput.close();}

```

### **4.3 Tracking Attendance**

Students Attendance for each course can be taken using 3 methods. When student attendance is taken, initially the student Id is matched with the Id present in the student directory database. If the student Id exists in the student directory database then the student name corresponding to that student Id will be captured and attendance will be taken. The student attendance will be tracked in the database for that particular course.

When student attendance is taken, the application will track whether the student is scanning for the first time or again. If student is scanning for the attendance on the same day for that course more than 1 time, then student will be indicated that “Student is scanning again”. If next day attendance is taken then the student will be indicated that “Genuine attendance taken”

#### **4.3.1 Optical Character Recognition (OCR Technology)**

The Application is using OCR technology to scan ID of students using Student ID card to take the attendance.

The OCR engine which is used in this application is Tesseract which is an optical character recognition engine for various operating systems [2]. This has been sponsored by Google and it is free software. Till date it is considered as the most accurate open source OCR engine available. When it is combined with the Leptonica Image Processing Library it can read a wide variety of image formats and convert them to text in over 60 languages. This has been improved extensively by Google. It is released under the Apache License 2.0.

The steps for building the OCR library is as follows [2] [6]:

- The first step would be to download the source or clone from this path <https://github.com/rmtheis/tess-two>. This project contains tools for compiling the Tesseract, and JPEG libraries for use on Android. This contains an Eclipse Android library project that provides a Java API for accessing natively-compiled Tesseract and Leptonica APIs. The eyes-two code is not required for this OCR library.

- To build this project using these commands :

```
cd <project-directory>/tess-two
```

```
ndk-build
```

```
android update project --path .
```

```
ant release
```

- Next step would be to import the project as a library in Eclipse. File -> Import -> Existing Projects into workspace -> tess-two directory.

Then right click the project, Android Tools -> Fix Project Properties.

Then right click -> Properties -> Android -> Check Is Library.

- Configure your project to use the tess-two project as a library project:

right click the project name -> Properties -> Android -> Library -> Add, and choose *tess-two*. Once this library is selected, OCR can be applied on any image using this library.

This library is used in our code.

- The rotation and image type should be modified as [3]:

```
ExifInterface exif = new ExifInterface(_path);
int exifOrientation = exif.getAttributeInt(
    ExifInterface.TAG_ORIENTATION,
    ExifInterface.ORIENTATION_NORMAL);

Log.v(TAG, "Orient: " + exifOrientation);

int rotate = 0;

switch (exifOrientation) {
    case ExifInterface.ORIENTATION_ROTATE_90:
        rotate = 90;
        break;
    case ExifInterface.ORIENTATION_ROTATE_180:
        rotate = 180;
        break;
    case ExifInterface.ORIENTATION_ROTATE_270:
        rotate = 270;
        break;
}

Log.v(TAG, "Rotation: " + rotate);

if (rotate != 0) {

    // Getting width & height of the given image.
    int w = bitmap.getWidth();
    int h = bitmap.getHeight();

    // Setting pre rotate
    Matrix mtx = new Matrix();
    mtx.preRotate(rotate);

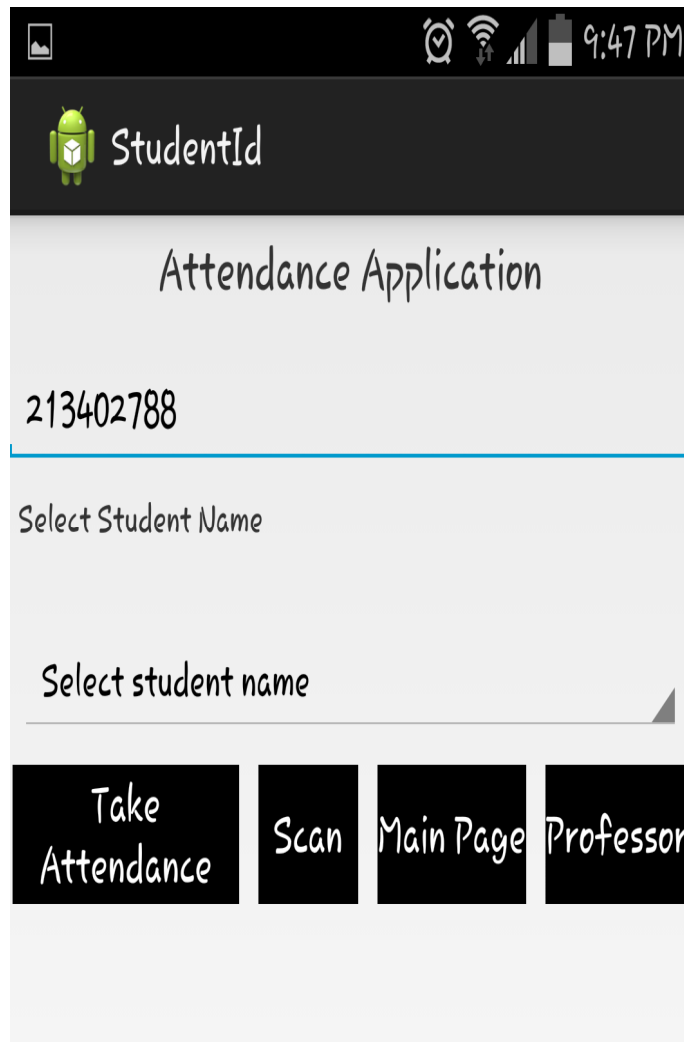
    // Rotating Bitmap
    bitmap = Bitmap.createBitmap(bitmap, 0, 0, w, h, mtx, false);
}

// Convert to ARGB_8888, required by tess
bitmap = bitmap.copy(Bitmap.Config.ARGB_8888, true);
```

- Now we have the image in the bitmap, and we can simply use the TessBaseAPI to run the OCR like follows [2]. The DATA\_PATH = Path to the storage and lang= for which the language data exists.

```
TessBaseAPI baseApi = new TessBaseAPI();  
baseApi.init(DATA_PATH,  
lang, TessBaseAPI.OEM_DEFAULT);  
baseApi.init(DATA_PATH, lang);  
baseApi.setImage(bitmap);  
String recognizedText = baseApi.getUTF8Text();  
baseApi.end()
```

First method is via the OCR technology where student will scan the Sac State Student Id card using the camera module. Once the student Id is captured, OCR is applied which will grab the Student Id from the image and populate in the Edit Text Box. The screen looks as:



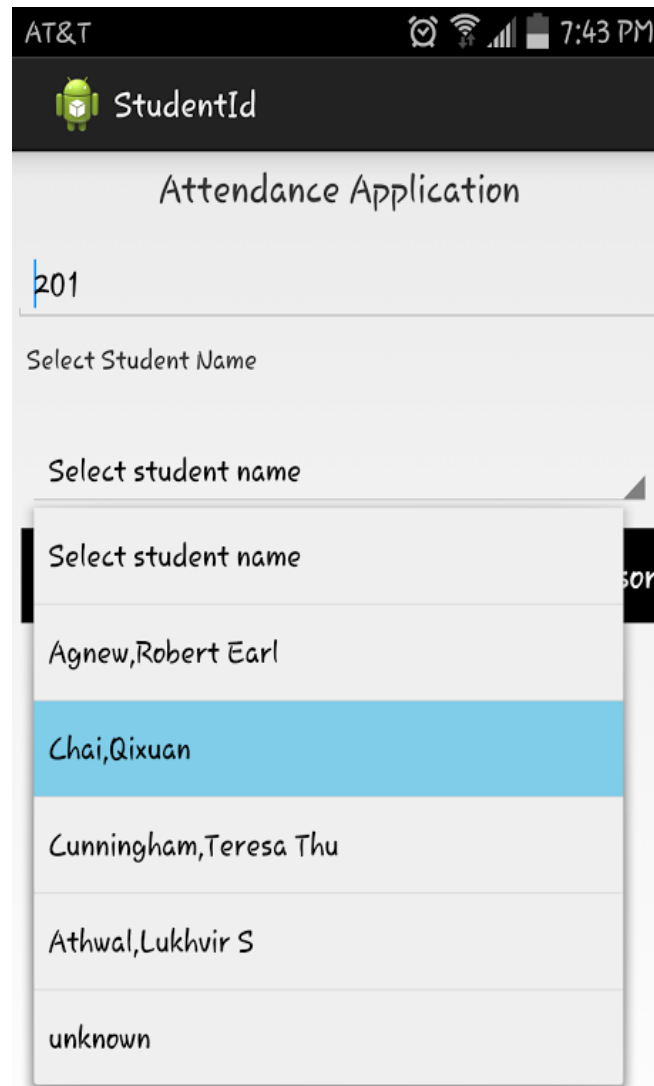
**Figure 4.4 Student ID Captured via OCR**

#### **4.3.2 Spinner Drop Down**

The second method to take the students attendance for the particular course is via the spinner drop down menu where professor can select the name of the student from the spinner drop down menu. Once the name is selected, the id of the student will be



populated in the Edit Text box and then professor can click on Take attendance button to take the attendance of the student. The screen looks as:



**Figure 4.5 Taking Students Attendance using Spinner**

### **4.3.3 Manually Adding the Student ID**

The third method to take the attendance for the students is by manual method. The professor/ student can add the Student ID manually in the Edit Text box and then click on Take Attendance Button to take the attendance. After the click event `add_save_btn` method is executed for taking the attendance of the students. Here, when the professor manually enters the student Id, the student Id is stored in the variable named `valid_studnumber`. Then all the details of the student are extracted from the database for the entered Student ID using function `Get_Studname`. Then, the code checks whether student attendance is already taken or not. To verify that, student count using the Student Id is calculated using function `IsStudentPresent`. Based on the count returned student attendance is taken. If `studcount` value equals 0, then this is the first time student attendance is taken for the course. If `studcount` values equals 1, then student attendance is being taken on the same day for the course and hence attendance will not be taken since his / her attendance is already taken for the class. If `studcount` value equals 2, then student attendance will be taken genuinely since the next class is being conducted for the course. The code snippet for taking the student attendance looks as:

```

add_save_btn.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {

        //valid_studnumber = OcrResult;
        valid_studnumber = add_studnumber.getText().toString();
        String studname;
        studname = dbHandler.Get_Studname(valid_studnumber);

        String lastdate = new SimpleDateFormat("yyyy-MM-
dd").format(new Date());
        int studcount = dbHandler.IsStudentPresent(atttable, valid_studnumber);
        System.out.println(studcount);
        Toast_msg = "Student Present?: " + studcount;
        Show_Toast(Toast_msg);
        // TODO Auto-generated method stub
        // check the value state is null or not
        if(studcount == 0)
        {
            if(valid_studnumber!=null &&
valid_studnumber.length()!=0){

                int pcount = 1;
                dbHandler.Add_Student(new
Student(valid_studnumber,studname,pcount,lastdate), atttable);

                Toast_msg = "Data inserted successfully " +
valid_studnumber + " " + studname + " " + pcount + " " + lastdate ;
                System.out.println(Toast_msg);
                Show_Toast(Toast_msg);
                globalText.setOcrText("");
                Reset_Text();
            }
            else{
                Toast_msg = "Data missing";
                Show_Toast(Toast_msg);
            }
        }else if(studcount == 1){
            Toast_msg="Attendance Already taken.Student is
scanning again";

            Show_Toast(Toast_msg);
            globalText.setOcrText("");
            Reset_Text();
        }else if (studcount ==2){
            Toast_msg = "Student attended the next class.
Geniune Attendance taken successfully.";
            Show_Toast(Toast_msg);
            globalText.setOcrText("");
            Reset_Text(); } } });}

```

#### **4.4 View Attendance**

Professor can view the student's attendance details for every course under View Attendance Category.

##### **4.4.1 View all Students Attendance**

When the professor clicks View Attendance button from the home screen of the application, by default all the student attendance details will be displayed. The details consists of Student name, student Id, present count / Total classes conducted for the course selected and the absent dates on which student was absent for that class. The date row will be empty if student has attended all the classes. The screen looks as:

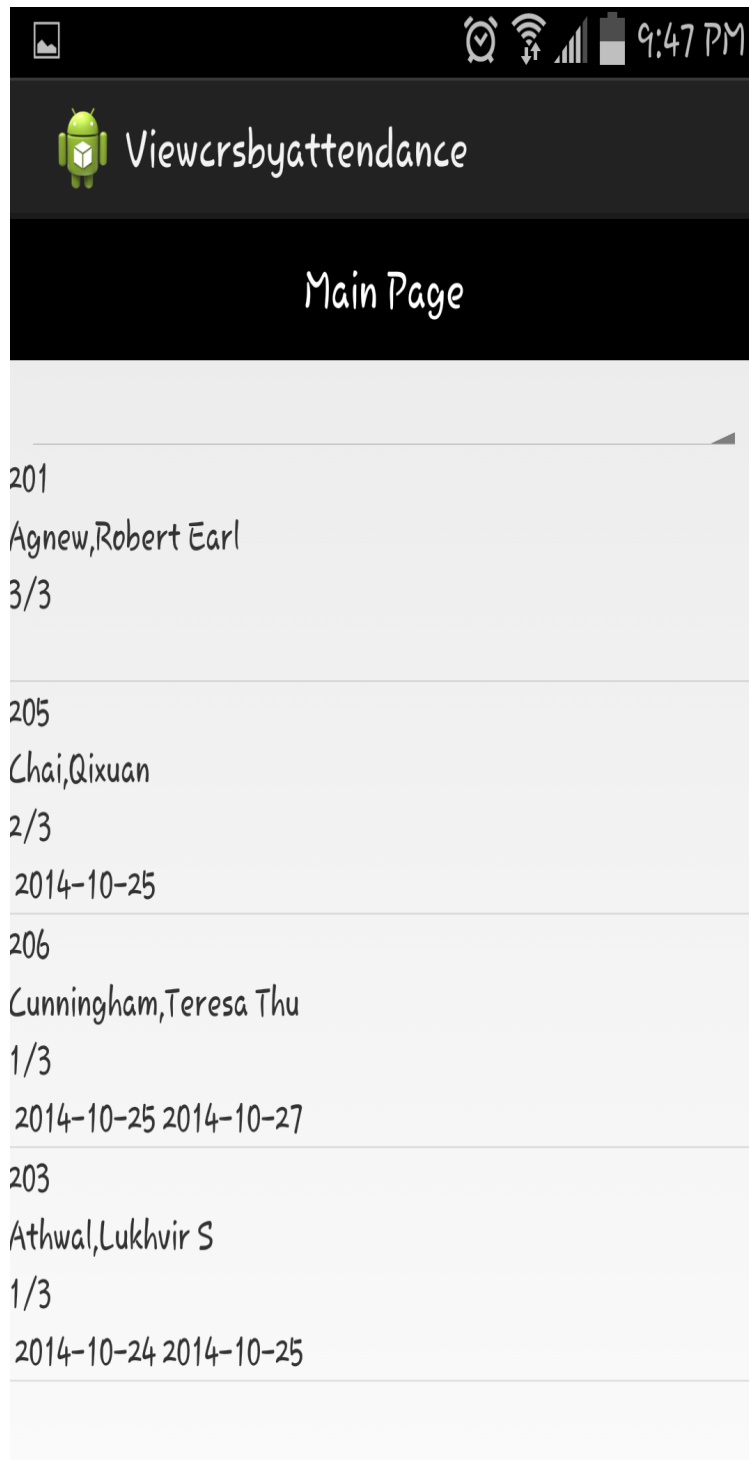


Figure 4.6 View All Student Attendance Details

When professor wishes to view the student attendance details, getView method is executed which will list all the student attendance details for the course selected. Here I have used holder which will store the student attendance details in the form of the array list. Also, the database where student details are stored is being queried to list all the dates when student was absent. The code snippet for viewing attendance looks as:

```

@Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View row = convertView;
        UserHolder holder = null;

        if (row == null) {

            LayoutInflater inflater = LayoutInflater.from(activity);

            row = inflater.inflate(layoutResourceId, parent, false);
            holder = new UserHolder();

            holder.studnumber = (TextView)
row.findViewById(R.id.user_studnumber_txt);
            holder.studname = (TextView)
row.findViewById(R.id.user_studname_txt);
            holder.studprecnt = (TextView)
row.findViewById(R.id.user_studprecount_txt);
            holder.studlastdate = (TextView)
row.findViewById(R.id.user_studlastdate_txt);
            //holder.viewattendance = (Button)
row.findViewById(R.id.viewcourseatt);

            row.setTag(holder);
        } else {

            holder = (UserHolder) row.getTag();
        }
        stud = data.get(position);
        holder.studnumber.setText(stud.getStudentId());
        holder.studname.setText(stud.getStudentName());

        holder.studprecnt.setText(String.valueOf(stud.getPresentCnt()) + "/"
+ String.valueOf(classcount));

        List<String> classdates = Arrays.asList(classdate.split(","));

        String studprdate = stud.getLastdate();
    
```

```

List<String> studdates = Arrays.asList(studprdate.split(","));
int clasdatelength = classdates.size();
int studdatelength = studdates.size();
String absentdates = "";
for (int i = 0; i<clasdatelength; i++){
    boolean present = false;
    for (int j = 0; j<studdatelength; j++){
        if(classdates.get(i).equals(studdates.get(j))){
            present = true;
            System.out.println("Present date : " +
classdates.get(i));
                break;
            }
        }
    if(present == false){
        System.out.println("Absent date : " +
classdates.get(i));
            absentdates = absentdates + " " +classdates.get(i);
        }
    }
    holder.studlastdate.setText(absentdates);
    // holder.studlastdate.setText(stud.getLastdate());
    System.out.println("Holder stud number selected " +
stud.getStudentId());
    System.out.println("Holder stud name selected " +
stud.getStudentName());

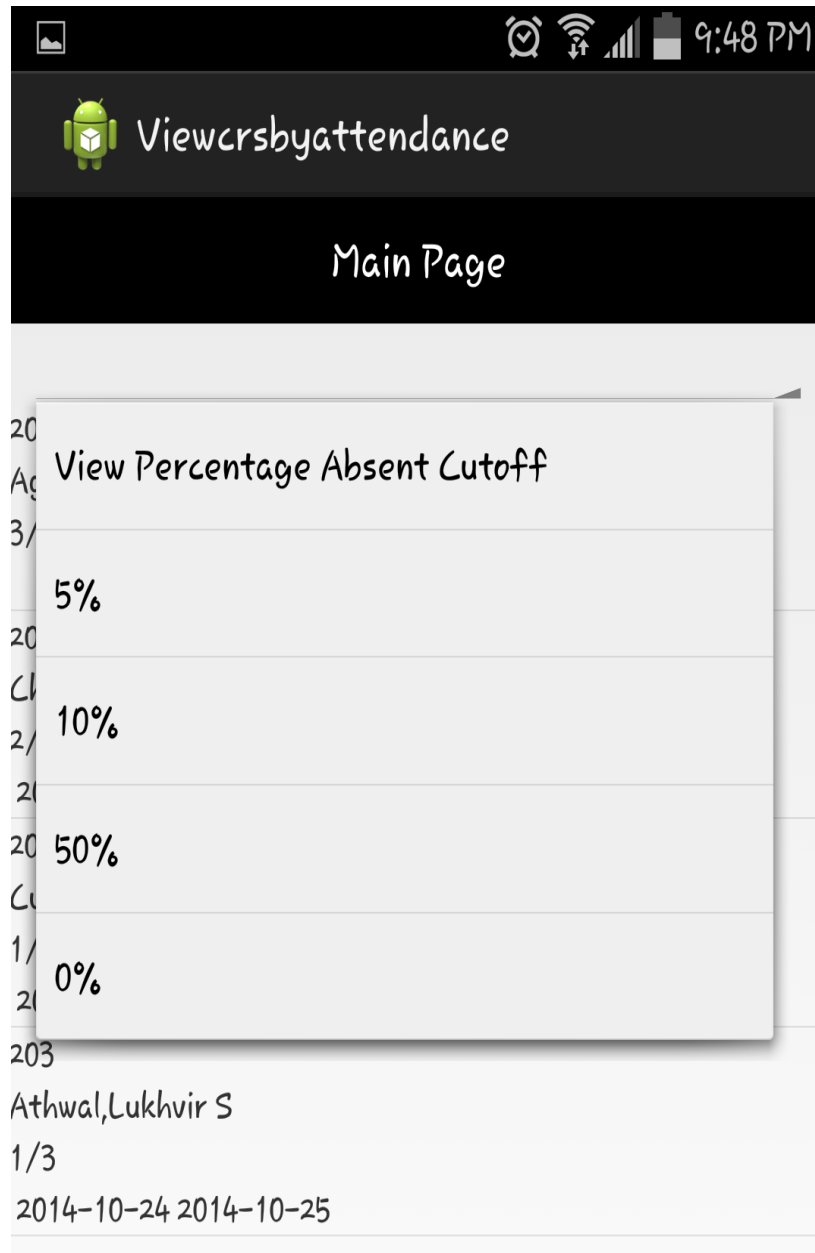
    return row;
}

```

#### 4.4.2 View Student Attendance by Absent Criteria

Professor can view the Student attendance details by selecting the absent criteria from the spinner. Spinner dropdown has different percentage criteria values. Once the professor selects the criteria, all the students absent details falling under the criteria will be listed.

The screen looks as:



**Figure 4.7 View Student Attendance Details by Selecting Percentage Absent Cutoff**

This is implemented using Spinners which provide a quick way to select one value from a set. The spinner shows its current value in default state. In order to select a new value,



need to touch on the spinner which will in turn display a dropdown menu with all available items.

The spinner is added to the layout with the Spinner object [6]. This is performed in the XML layout with a <Spinner> element like below :

```
<Spinner
    android:id="@+id/perspinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:prompt="@string/spinner_title"
    android:layout_marginTop="20dip"
    android:layout_marginLeft="8dip"
    android:layout_marginRight="8dip"
/>
```

An array adapter is used to provide the choices for the spinner. The choices can come from an SpinnerAdapter such as an ArrayAdapter if the choices are in an array.

```
ArrayAdapter<String> adapter_state = new ArrayAdapter<String>(this,
    android.R.layout.simple_spinner_item, viewatt);
    adapter_state

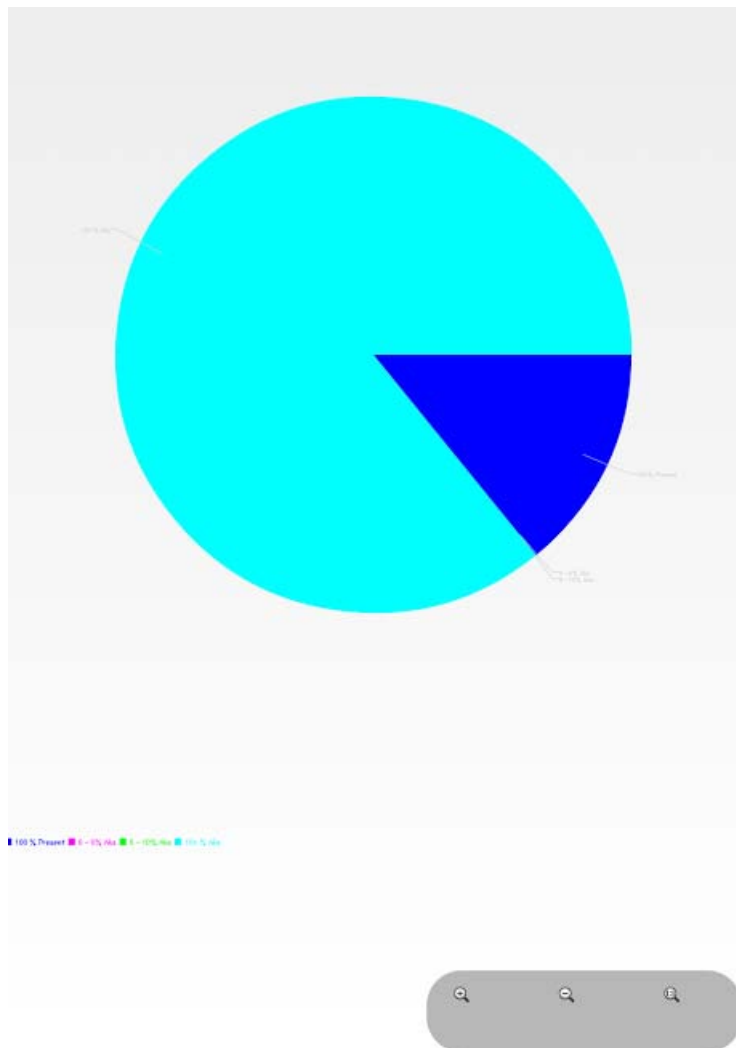
    .setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    spinner.setAdapter(adapter_state);
    spinner.setOnItemClickListener(this);
```

Next step is to call setAdapter() to apply the adapter to the Spinner.

```
public void onItemSelected(AdapterView<?> parent, View view, int position,
long id) {
// TODO Auto-generated method stub
if(firsttime == true){
firsttime = false;
return; }
firsttime = true;
spinner.setSelection(position);
String viewatt = (String) spinner.getSelectedItem();
Intent viewcrsatt = new
Intent(Viewcrsbyattendance.this,Viewcrsbyattendance.class);
String USER_ID = (getIntent().getStringExtra("USER_ID"));
viewcrsatt.putExtra("viewperatt", viewatt);
viewcrsatt.putExtra("USER_ID", USER_ID);
startActivity(viewcrsatt); }
```

#### 4.4.3 View Pie Chart

Professor can view the student attendance information in the form of the pie chart. Pie chart illustrates how much percentage of students attended the class for the total number of classes conducted. The screen looks as:



**Figure 4.8 View Pie Chart**

In order to draw the pie chart AChartEngine jar file is being used. AChartEngine is a charting library that is used in Android application to display statistical data in the form of bar chart, pie chart etc. [6] [11]. Here, pie chart is constructed using AChartEngine jar. The design components that needs to be defined for drawing the pie chart are [11]:

- Renderers – This is used to customize the charts appearance like colors, fonts etc.
- ChartFactory – gets an instance of a dataset and an instance of a renderer and returns the pie chart embedded into an Intent (for the case when the chart fills an Activity) or a View (when the chart is a part of an Activity, together with other widgets).
- Tools – interaction tools like zoom feature.

Here the categories are being defined in the array named code on the basis of which student attendance distribution details will be displayed. After all the renderers and the distribution series is defined, Intent ChartFactory.getPiechartIntent is used to draw the final pie chart where the defined DistributionSeries, Renderer information etc. is passed as argument. Since pie chart needs to be constructed, AChartEnginePiechartDemo view is passed to ChartFactory Intent [6] [10].

The code snippet for viewing the pie chart looks as:

```

private void viewpiechart(){

    final GlobalClass globalText = (GlobalClass)
getApplicationContext();
    String[] code = new String[] {
        "100 % Present", "0 - 5% Abs", "5 - 10% Abs", "10+ % Abs" };
    int[] distribution = {1,2,3,4} ;

// Populate Distribution Array with total percentage students who falls under
the categories defined in code string array.
    for(int i = 0; i < 4 ; i++){

        distribution[i] = globalText.getbucket(i);
        /*if(i == 0 && distribution[i] == 0){
            distribution[i] = 100;
        }*/
        String Toast_msg = "Distribution " +distribution[i];
        Show_Toast(Toast_msg);
    }
    int[] colors = { Color.BLUE, Color.MAGENTA, Color.GREEN, Color.CYAN};

// Instantiating CategorySeries to plot Pie Chart
    for(int i=0 ;i < distribution.length;i++){
        distributionSeries.add(code[i], distribution[i]);
    }

// Instantiating a renderer for the Pie Chart
    DefaultRenderer defaultRenderer = new DefaultRenderer();
    for(int i = 0 ;i<distribution.length;i++){
        SimpleSeriesRenderer seriesRenderer = new SimpleSeriesRenderer();
        seriesRenderer.setColor(colors[i]);
        seriesRenderer.setDisplayChartValues(true);
        defaultRenderer.addSeriesRenderer(seriesRenderer);
    }

    Intent intent = ChartFactory.getPieChartIntent(getBaseContext(),
distributionSeries , defaultRenderer, "AChartEnginePieChartDemo");
        startActivity(intent);
}

```

#### 4.5 Enhanced OCR Functionality

This application uses OCR technology which has been implemented by using Tess library in this application. The steps required to pull out the ID from the Student ID card are first

click the picture and click on Done. This basically requires two separate steps to pull out the ID from the student Id card. But it is also possible to pull the ID in just one step using OCR. And the way this can be done is to create two layers: one layer would be the camera preview and other is the viewfinder to get specific part of the image [2] [3] [6].

#### 4.5.1 Camera Class

The Camera class will explain briefly with sample code snippets regarding how the user will be notified on completion of camera auto focus, whether camera driver has been initialized or not, how preview frames will be displayed.

##### **Auto Focus manager class:**

This class is mainly used to notify on completion of camera auto focus onAutoFocus is called when the camera auto focus completes. If the camera does not support auto-focus and autoFocus is called, onAutoFocus will be called immediately with a fake value of success set to true. This routine does not lock auto-exposure and auto-white balance after it completes. Success returns true if focus was successful and false otherwise.

Camera is the camera service object. The code snippet for onAutoFocus [3] looks as:

```
public synchronized void onAutoFocus(boolean success, Camera theCamera) {
    if (active && !manual) {
        outstandingTask = new TimerTask() {
            @Override
            public void run() {
                checkAndStart();
            }
        };
        timer.schedule(outstandingTask, AUTO_FOCUS_INTERVAL_MS);}
    manual = false; }
```

**PreviewCallbackclass:**

This is called when preview frames are displayed. The parameters used are data which is the content of the preview frame in the format defined by ImageFormat, which can be queried with getPreviewFormat() [3]. The code snippet for onPreviewFrame looks as:

```
public void onPreviewFrame(byte[] data, Camera camera) {
    Point cameraResolution = configManager.getCameraResolution();
    Handler thePreviewHandler = previewHandler;
    if (cameraResolution != null && thePreviewHandler != null) {
        Message message = thePreviewHandler.obtainMessage(previewMessage,
cameraResolution.x,cameraResolution.y, data);
        message.sendToTarget();
        previewHandler = null;
    } else {
        Log.d(TAG, "Got preview callback, but no handler or resolution
available");
    }
}
```

**CameraManagerClass:**

This class will make sure a camera driver is opened and the hardware parameters are initialized. The holder is the surface object which the camera will draw preview frames into and finally an IO exception which indicates that the camera driver failed to open.

The code snippet for CameraManagerClass [3] looks as:

```

public synchronized void openDriver(SurfaceHolder holder) throws IOException {
    Camera theCamera = camera;
    if (theCamera == null) {
        theCamera = Camera.open();
        if (theCamera == null) {
            throw new IOException();
        }
        camera = theCamera;
    }
    camera.setPreviewDisplay(holder);
    if (!initialized) {
        initialized = true;
        configManager.initFromCameraParameters(theCamera);
        if (requestedFramingRectWidth > 0 && requestedFramingRectHeight > 0) {
            adjustFramingRect(requestedFramingRectWidth,
                requestedFramingRectHeight);
            requestedFramingRectWidth = 0;
            requestedFramingRectHeight = 0;
        }
    }
    configManager.setDesiredCameraParameters(theCamera);

    SharedPreferences prefs =
    PreferenceManager.getDefaultSharedPreferences(context);
    reverseImage = prefs.getBoolean(PreferencesActivity.KEY_REVERSE_IMAGE,
    false);
}

```

The next step would be to ask the camera hardware to begin drawing preview frames to the screen. The code snippet for startPreview [3] looks as:

```

public synchronized void startPreview() {
    Camera theCamera = camera;
    if (theCamera != null && !previewing) {
        theCamera.startPreview();
        previewing = true;
        autoFocusManager = new AutoFocusManager(context, camera);
    }
}

```



The code snippet to tell the camera to stop drawing the frames looks as:

```
public synchronized void stopPreview() {
    if (autoFocusManager != null) {
        autoFocusManager.stop();
        autoFocusManager = null;
    }
    if (camera != null && previewing) {
        camera.stopPreview();
        previewCallback.setHandler(null, 0);
        previewing = false;
    }
}
```

A single preview frame will be returned to the handler supplied. The data will arrive as as byte[] in the message.obj field, with width and height encoded as message.arg1 and message.arg2, respectively. The handler is the one to send the message to. And the message is the field of the message to be sent [3] [6]. The code snippet for requesting OCR Decode looks as:

```
public synchronized void requestOcrDecode(Handler handler, int message) {
    Camera theCamera = camera;
    if (theCamera != null && previewing) {
        previewCallback.setHandler(handler, message);
        theCamera.setOneShotPreviewCallback(previewCallback);
    }
}
```

### 4.5.2 Capture Activity

Capture Activity class will help understand in what way one can select a region, in this case it will be selecting student id region from the student Id card image and then saving it in the view which will be used to display the captured Student Id, how one can setup the camera preview surface and initialize the OCR engine to.

#### View Finder Class:

The second layer is creating a view finder for the camera capture activity. This must be done so that we can select a region from the capture and save image from bitmap view inside viewfinder.

In order to get a framing rectangle in the captured region, we use Rect. Rect holds four integer coordinates for a rectangle. The rectangle is represented by the coordinates of its 4 edges (left, top, right bottom). These fields can be accessed directly. The width() and height() is used to retrieve the rectangle's width and height [3].

```
Rect frame = cameraManager.getFramingRect();
    if (frame == null) {
        return;
    }
```

```
int width = canvas.getWidth();
int height = canvas.getHeight();
```

In order to differentiate the background from the foreground, need to draw the exterior.

```
paint.setColor(maskColor);
canvas.drawRect(0, 0, width, frame.top, paint);
canvas.drawRect(0, frame.top, frame.left, frame.bottom + 1, paint);
canvas.drawRect(frame.right + 1, frame.top, width, frame.bottom + 1, paint);
canvas.drawRect(0, frame.bottom + 1, width, height, paint);
```

To display the ID captured from the rect, a flag is used to represent the results from TessBaseApi. When this flag is true then the Id is retrieved from the result text and then draw a bounding box around that Id. The code snippet looks as:

```
static final boolean DRAW_WORD_BOXES = true;

if (DRAW_WORD_BOXES || DRAW_WORD_TEXT) {
wordBoundingBoxes = resultText.getWordBoundingBoxes();
}

if (DRAW_WORD_BOXES) {
paint.setAlpha(0xFF);
paint.setColor(0xFF00CCFF);
paint.setStyle(Style.STROKE);
paint.setStrokeWidth(1);
for (int i = 0; i < wordBoundingBoxes.size(); i++) {

rect = wordBoundingBoxes.get(i);
canvas.drawRect(
frame.left + rect.left * scaleX,
frame.top + rect.top * scaleY,
frame.left + rect.right * scaleX,
frame.top + rect.bottom * scaleY, paint);
}
}

if (DRAW_WORD_TEXT) {
words = resultText.getText().replace("\n", " ").split(" ");
int[] wordConfidences = resultText.getWordConfidences();
for (int i = 0; i < wordBoundingBoxes.size(); i++) {
boolean isWordBlank = true;
try {
if (!words[i].equals("")) {
isWordBlank = false;
}
} catch (ArrayIndexOutOfBoundsException e) {
e.printStackTrace();
}
}
```

### CaptureActivityClass:

This class will setup the camera preview surface and initialize the OCR engine. And in case OCR is already initialized, then just start the camera [3].

```

surfaceView = (SurfaceView) findViewById(R.id.preview_view);
surfaceHolder = surfaceView.getHolder();
if (!hasSurface) {
    surfaceHolder.addCallback(this);
    surfaceHolder.(SurfaceHolder);
}

boolean doNewInit = (baseApi == null) ||
!sourceLanguageCodeOcr.equals(previousSourceLanguageCodeOcr) ||
    ocrEngineMode != previousOcrEngineMode;
if (doNewInit) {
    File storageDirectory = getStorageDirectory();
    if (storageDirectory != null) {
        initOcrEngine(storageDirectory, sourceLanguageCodeOcr,
sourceLanguageReadable);
    }
} else {
    resumeOCR();
}

```

Next is to start or restart the recognition after the OCR engine has been initialized or after the app regains focus. This sets the state related settings and OCR engine parameters and requests camera initialization. The code snippet for resuming the OCR Engine looks as:

```

void resumeOCR() {
    Log.d(TAG, "resumeOCR()");

    // This method is called when Tesseract has already been successfully
    initialized, so set
    // isEngineReady = true here.
    isEngineReady = true;

    isPaused = false;

    if (handler != null) {
        handler.resetState();
    }
    if (baseApi != null) {
        baseApi.setPageSegMode(pageSegmentationMode);
        baseApi.setVariable(TessBaseAPI.VAR_CHAR_BLACKLIST, characterBlacklist);
        baseApi.setVariable(TessBaseAPI.VAR_CHAR_WHITELIST, characterWhitelist);
    }

    if (hasSurface) {
        initCamera(surfaceHolder);
    }
}

```

Once ocrResults represents successful results, the recognized text needs to be displayed in the Textbox on the screen. When the results are captured successfully we need to turn off the capture related UI elements like shutter button, camera button etc. We only want the visibility for the result View where successful results will be displayed. The results are binded with the TextBox named ocr\_result\_text\_view where the captured results will be displayed. The code snippet for displaying the successful results [3] looks as:

```

boolean handleOcrDecode(OcrResult ocrResult) {

```

```

        lastResult = ocrResult;

        // Test whether the result is null
        if (ocrResult.getText() == null || ocrResult.getText().equals("")) {
            Toast toast = Toast.makeText(this, "OCR failed. Please try
again.", Toast.LENGTH_SHORT);
            toast.setGravity(Gravity.TOP, 0, 0);
            toast.show();
            return false;
        }

        // Turn off capture-related UI elements
        shutterButton.setVisibility(View.GONE);
        statusViewBottom.setVisibility(View.GONE);
        statusViewTop.setVisibility(View.GONE);
        cameraButtonView.setVisibility(View.GONE);
        viewfinderView.setVisibility(View.GONE);
        resultView.setVisibility(View.VISIBLE);

        ImageView bitmapImageView = (ImageView)
findViewById(R.id.image_view);
        lastBitmap = ocrResult.getBitmap();
        if (lastBitmap == null) {
            bitmapImageView.setImageBitmap(BitmapFactory.decodeResource(getResources(),
                R.drawable.ic_launcher));
        } else {
            bitmapImageView.setImageBitmap(lastBitmap);
        }

        // Display the recognized text
        TextView sourceLanguageTextView = (TextView)
findViewById(R.id.source_language_text_view);
        sourceLanguageTextView.setText(sourceLanguageReadable);
        TextView ocrResultTextView = (TextView)
findViewById(R.id.ocr_result_text_view);
        ocrResultTextView.setText(ocrResult.getText());

        final GlobalClass globalText = (GlobalClass)
getApplicationContext();
        globalText.setOcrText(ocrResult.getText());
        return true;
    }

```

## 4.6 Future Work

This application utilizes OCR technique to capture student Id number from ID card. However, current implementation requires user to crop and select the area on screen before taking the picture. This method is very accurate but is little time consuming as user needs to crop the area before capturing. Other alternative is that user should scan the whole picture and application should extract student id from the picture. But this method is not effective 100% of time while capturing student id. Future prospective for this application would be to improvise on this alternative.

Instead of using OCR technology, barcode scanner can be implemented to take the students attendance using student ID card. This alternative will be accurate and time constraint issue will be resolved.

Another feature that can be added is to mark student's scanning time during the scheduled class and flag the attendance if student arrive late by certain amount of time after the class has resumed.

## CONCLUSION

Registering attendance during a class can be very time consuming process, which requires manual steps and is error prone. This application solves all the downfalls of registering attendance manually. It allows professor to scan student's Id card and registers student's presence, which not only saves time but is also error free. Additionally, professor has the luxury to look at attendance record of students whenever he wants. Just a few clicks away! Professor can also query and look up only those students whose % attendance did not meet minimum criteria.

## References

[1] Android Developer guide



<http://developer.android.com/>

[2] Research on Tesseract, most accurate OCR engine available.

<https://code.google.com/p/tesseract-ocr/>

[3] An overview of the Tesseract OCR engine by Ray Smith

[http://static.googleusercontent.com/external\\_content/untrusted\\_dlcp/research.google.com/en/us/pubs/archive/33418.pdf](http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en/us/pubs/archive/33418.pdf)

[4] “Android Cookbook” by Ian F. Darwin (April 2012), *O’Reilly Media, Inc.*

<http://androidcookbook.com/home.seam>

[5] Android Developers Blog

<http://android-developers.blogspot.com/>

[6] Stack Overflow

<http://stackoverflow.com/>

[7] Android Snippets To Download the file from the server

<http://www.androidsnippets.com/download-an-http-file-to-sdcard-with-progress-notification>

[8] Tutorials for learning Android Operating System -

<http://www.youtube.com/user/thenewboston?feature=watch>

[9] Learn Android SDK from Scratch

<http://code.tutsplus.com/series/learn-android-sdk-from-scratch--mobile-21677>

[10] Tutorial for drawing pie chart using AChartEngine

<http://wptrafficanalyzer.in/blog/android-drawing-pie-chart-using-achartengine>

[11] Brief Introduction for using AChartEngine.

<http://www.javacodegeeks.com/2012/12/achartengine-a-charting-library-for-android-applications.html>