**CHAPTER-1**

**INTRODUCTION**

**1.1 ANDROID**

**1.1.1 What is Android?**

Android is basically an operating system for smartphones. But we find now integrated into PDAs, touch pads or televisions, even cars (trip computer) or netbooks.

Android is an open source and Linux-based Operating System for mobile devices such as smartphones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies. The OS was created by the start-up of the same name, which is owned by Google since 2005 .

**1.1.2 Specifications**

This operating system is based on version 2.6 of Linux, so it has a monolithic system kernel, what means that all system functions and drivers are grouped into one block of code.

**Architecture** : Android consists of five layers: -The Linux kernel 2.6-which includes useful drivers that allow for example WiFi or Bluetooth.

The library written in C and C + + that provide higher level functionality such as an HTML engine, or a database (SQLite).

A **runtime environment** for applications based on a virtual machine, made for inefficient machines such as telephones. The aim is to translate JAVA in machine language understood by Android. A **JAVA** framework that allows applications running on the virtual machine to organize and cooperate.

**1.1.3 Characterisctics of Market**

**Competitors**

The principal competitor is **iPhone OS**. It is mainly for competing with Apple thatAndroid has been created.

**Palm OS** devices on PDA.

**Blackberry:** which team the same name smartphones

**Windows Mobile**: which team smartphones and PDAs.

**Symbian**: Cu$rrent Market Leader

**• Key partners:** To help launch Android, Google has created an alliance of thirty companies in order to develop standards for mobile devices. There is, among others: -Operators such as NTT Dokomo, T-Mobile or Bouygues Telecom -Of-equipment manufacturers like Sony Ericsson or Samsug -Manufacturers of semiconductors, including Intel and Nvidia -Corporate businesses. • market share The android market share continues to increase since its inception, and is likely to continue climbing because it is favored by big players like HTC , Sony Ericsson, Samsung, LG, Motorola, Dell, Acer . Moreover, according to IDC, android will be the 2nd mobile operating system used of the market in 2013. Here is the state of the market from 2006 to 2009. You have to know that the first mobile phone appeared in android date October 2008.

**1.2 Why Android?**

**Applications**

Google applications Android includes most of the time many Google applications like Gmail, YouTube or Maps. These applications are delivered with the machine most of the time, except in certain cases, such as some phones running android on which the provider has replaced Google applications by its own applications. Android applications are usually developed in the Java language using the Android Software Development Kit. Once developed, Android applications can be packaged easily and sold out either through a store such as Google Play or the Amazon Appstore.

**Widgets**

With android, it is possible to use widgets which are small tools that can most often get information. These widgets are directly visible on the main window.

**Android Market**

This is an online software store to buy applications. Developers who created applications can add them into the store, and these applications can be downloaded by users, they can be both free and paid.

**Multitasking**

Android allows multitasking in the sense that multiple applications can run simultaneously. With Task Manager it is possible view all running tasks and to switch from one to another easily.

**SDK**

A development kit has been put at disposal of everybody. Accordingly, any developer can create their own applications, or change the android platform. This kit contains a set of libraries, powerful tools for debugging and development, a phone emulator, thorough documentation, FAQs and tutorials.

**Modifiability**

This allows everyone to use, improve or transform the functions of Android for example transform the interface in function of uses , to transform the platform in a real system embedded Linux.

**1.3 Hardware Requirements**

1. Intel P4 processor with minimum 2.0Ghz Speed
2. RAM: Minimum 4GB(Recommended 8GB)
3. Hard Disk: Minimum 200 GB

**1.4 Features of Android:**

Android is a powerful operating system competing with Apple 4GS and support great features. Few of them are listed below:

**Beautiful UI:** Android OS basic screen provides a beautiful and intuitive user interface.

**Connectivity**”GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.

**Storage**: SQLite, a lightweight relational database, is used for data storage purposes.

**Media support** : H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP

**Messaging**: SMS and MMS

**Web browser**: Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3. Multi-touch Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.

**Multi-tasking:** User can jump from one task to another and same time various application can run simultaneously.

**Resizable widgets**: Widgets are resizable, so users can expand them to show more content or shrink them to save space.

**Multi-Language**: Support single direction and bi-directional text.

**GCM**: Google Cloud Messaging (GCM) is a service that let developers send short message data to their users on Android devices, without needing a proprietary sync solution.

**Wi-Fi Direct**: A technology that let apps discover and pair directly, over a high-bandwidth peer-to-peer connection.

**Android Beam:** A popular NFC-based technology that let users instantly share, just by touching two NFC-enabled phones together.

**CHAPTER-2   
 TRAINING WORK**

**2.1 Tools Used :-**

**Software Requirements:-**

1. JDK 1.8 - Eclipse

2. Database - MySQL Database Server 5.5

4. Jdbc Driver for MySQL Database Server - navicatlite

5. Operating System - Windows 7 Elementery Edition/ Win 8 / Linux Fedora

6. Supporting Tools: Adobe Photoshop,Adobe Illustrator

**2.2 Technologies Used**

Java and Android

**2.2.1 Java:**

A part from being a system independent language, there are other reasons too for the immense popularity of this language. Let us have a look at some of its features.

* Simple
* Object- oriented
* Distributed
* Robust and secure
* System Independence
* Scalability
* High performance
* Compiled and Interpreted
* Platform Independent and portable
* Simple and small
* Multithreaded and Interactive

It consists of two parts:-

1. JVM stands for Java Virtual Machine, which is run time environment to execute the java programs.
2. Java API (Application Programming Interface) that consists of inbuilt classes used in java programs.

* **JDBC** (Java Database Connectivity) is an API, which is used for the communication of java programs with different databases.

**2.2.2ANDROID**

### Eclipse ADT (Android Development Tools)

### Android Development Tools (ADT) is a plug in for the Eclipse IDE that is designed to provide an integrated environment in which to build Android applications. ADT extends the capabilities of Eclipse to let developers set up new Android projects, create an application UI, add packages based on the Android Framework API, debug their applications using the Android SDK tools, and export signed (or unsigned) .apk files in order to distribute their applications. It is a freeware available to download. It was the official IDE for Android but was replaced by Android Studio (based on Intelligent IDEA Community Edition).I’ve worked on ECLIPSE for JAVA and ANDROID STUDIO for ANDROID.

**2.3 CREATING ANDROID VIRTUAL DEVICE**

To test your Android applications you will need a virtual Android device. So before we start writing our code, let us create an Android virtual device. Launch Android AVD Manager using ANDROID STUDIO menu options Window > AVD Manager> which will launch Android AVD Manager. Use New button to create a new Android Virtual Device and enter the following information, before clicking Create AVD button.If your AVD is created successfully it means your environment is ready for Android application development. If you like, you can close this window using top-right cross button. Better you re-start your machine and once you are done with this last step, you are ready to proceed for your first Android example.

**2.4 APPLICATION COMPONENT**

Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file AndroidManifest.xml that describes each component of the application and how they interact.

**2.4.1 Basics of creating Activities**

To begin to program for Android I needed some basics, because some elements are very different, even if programming an application in Android uses the Java language, therefore, an object oriented language. Firstly, in an Android application, there is no main method:

**Public static void main(String[] args){…..}**

This method that allows to launch a program in java is not present in an application android.

**2.4.2 Activities**

An activity is a user interface that allows the user to interact with the screen, to perform actions. For example, a text messaging application could have an activity that displays a list of contacts to send messages. Once the contact is selected , activity could send information to a second activity that could serve to send the message to the contact. When an application is launched, what it displays is the result of an activity. At the code level, for create an activity , you must create a class that extends the Activity class. An activity has a required onCreate () method. It is the main method. To interact with the program, through the activity, there must be something displayed, that is why the activity, contains what is called views

An activity is implemented as a subclass of Activity class as

**public class MainActivity extends Activity {}**

**2.4.3 Views**

A View is the basic building block for user interface components. A View occupies a rectangular area on the screen.View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.).There is differents kinds of views, for example a ListView is able to display only an interactive list of what you want to display, while a WebView allows you to to display a web page. As said before, A view occupies a rectangular area on the screen. To organise these rectangles on the screen, there is a text file written in XML for every different screen.

**2.4.4 XML**

Xml means Extensible Markup Language. Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses. The goal of using Android's XML vocabulary, is to quickly design UI layouts and the screen elements they contain, in the same way that creating web pages in HTML : with a series of nested elements.

Here is an example

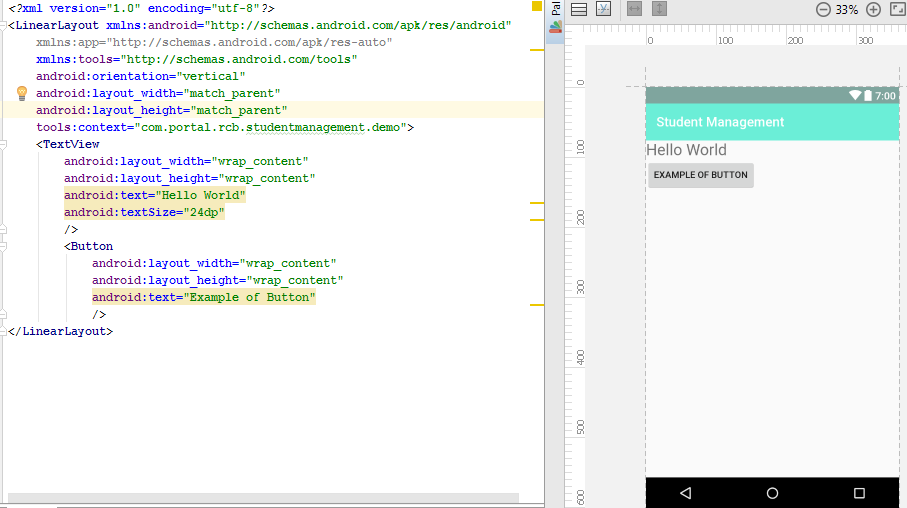


Fig:2.1- Xml Layout

here's an XML layout that uses a vertical LinearLayout to hold a TextView and a Button.It will be possible to modify attributes the elements in the activity class that use this XML file. For example to change the text.

**2.4.5 INTENT**

An activity can of course start another one, even if it but to do this, it will need a special object called Intent. An intent is basia description of an operation to be performed. It can launch an Activity, send a broadcastIntent to any interested BroadcastReceiver components, and communicate with a background Service. An Intent performs binding between the code in different applications. it can be thought of as the link between activities. It is possible to add some informations to an Intent, thanks to an object called bundle, that you add to the intent thanks to the method :

Intent.putExtras(Bundle objectBundle);

**2.4.6 ANDROID MANIFEST**

AndroidManifest.xml file is necessary for all android applications and must have this name in its root directory. In the manifest you can find essential informations about the application for the Android system, informations that the system must have before it can run any of the application's code. Here is what you can find in the Android manifest:

The name of the Java package for the application. The package name serves as a unique identifier for the application.

The description of the components of the application : the activities, services, broadcast receivers, and content providers that the application is composed of and under what conditions they can be launched .

The processes that will host application components.

The permissions the application must have in order to access protected parts of the API and interact with other applications.

The permissions that others are required to have in order to interact with the application's components.

The list of the Instrumentation classes that provide profiling and other information as the application is running. These declarations are present in the manifest only while the application is being developed and tested; they're removed before the application is published.

The minimum level of the Android API that the application requires. -The list of the libraries that the application must be linked against. Whit all theses elements, an application can be created

**2.5 My first Application**

For my first application, I was not really comfortable with android, so in this application there is only two differents screen, and so two differents activities, but it was a good training to try how to communicate between the activities, how to display elements (TextView, Buttons ...) on the screen and how to interact with them.

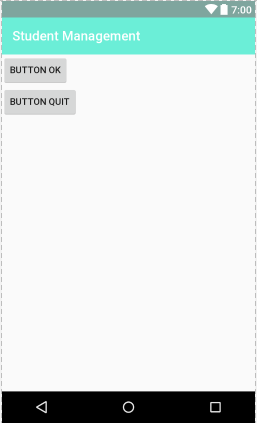
****

Fig:2.2-Buttons in Linear Layout

The first screen is drawn in a linear layout which is the simplest layout mechanism available on Android. you can add components horizontally or vertically’. This is a pretty simple way of getting components assembled on a screen. You can find the XML code for this screen in the Appendixes. As you will see, there is one TextView, two buttons and again one TextView.

The two buttons have been declared in the XML file, but it is not enough for them for being active. I must declare them in the code:

**Private void okbutton;**

**Private void quitbutton;**

Then you instantiate the buttons: they are linked with those of the xml file as follows:

**Okbutton=(Button)findviewbyid(R.id.okbutton);**

**Quitbutton=(Button)findviewbyid(R.id.quitbutton);**

In this way it is now possible to interact with the buttons, thanks to the next method:

**Button.setOnClickListener(OnClickListener listener)**

The aim of the 'OK' button is to access the process list, so to a new screen and thus a new activity: we will have a new Intent as explained previously. The action on the button code is as follow :

**okButton.setOnClickListener(new OnClickListener() {**

**public void onClick(View v) {**

**try{**

**Intent i = new Intent(TaskManager.this, ProcessList.class);**

**startActivity(i);**

**finish();**

**} catch(Exception e){**

**TextView ex = new TextView(TaskManager.this);**

**ex.setText(e.toString());**

**setContentView(ex);**

**Log.e(this.getClass().getSimpleName(), "Error :", e);**

**}**

**}**

**});**

To start an activity it is necessary to declare an intent and then implement it with what is called the context (current activity) and the name of the class that defines the next event (this one extends the Activity class). The next screen is a list of processes currently running on the device.

You must know that when the first of an application's components needs to be run, Android starts a Linux process for it with a single thread of execution. By default, all components of the application run in that process.

**2.6** **IMPROVEMENTS**

The improvements I could do would be first to add options to see this application such as the percentage of resources that each application takes,to see the relationship between processes. Finally a graphical look more presentable would have been preferable, but I had not the time to do it so I could not do better,somehow here are some additional components we need to use in our app to provide more functionality.

**2.7ADDITIONAL COMPONENTS**

**2.7.1 Services:**

A service is a component that runs in the background to perform long-running operations. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity. A service is implemented as a subclass of Service class as

**public class MyService extends Service { }**

Service may take essentially two states

**Started** A service is started when an application component, such as an activity, starts it by calling startService(). Once started, a service can run in the background indefinitely, even if the component that started it is destroyed.

**Bound** A service is bound when an application component binds to it by calling bindService(). A bound service offers a client-server interface that allows components to interact with the service, send requests, get results, and even do so across processes with inter-process communication (IPC).

A service has lifecycle callback methods that you can implement to monitor changes in the service's state and you can perform work at the appropriate stage.

**onStartCommand()** The system calls this method when another component, such as an activity, requests that the service be started, by calling startService(). If you implement this method, it is your responsibility to stop the service when its work is done, by calling stopSelf() or stopService() methods.

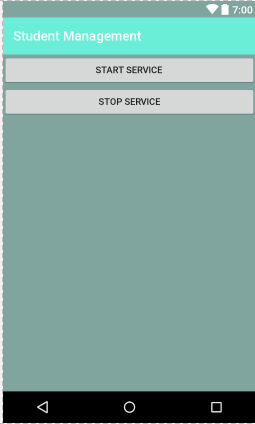


Fig:2.3-Services

**onBind()** The system calls this method when another component wants to bind with the service by calling bindService(). If you implement this method, you must provide an interface that clients use to communicate with the service, by returning an IBinder object.you must always implement this method, but if you don't want to allow binding, then you should return null.

**onUnbind()** The system calls this method when all clients have disconnected from a particular interface published by the service.

**onRebind()** The system calls this method when new clients have connected to the service, after it had previously been notified that all had disconnected in its onUnbind(Intent).

**onCreate()** The system calls this method when the service is first created using onStartCommand() or onBind(). This call is required to perform one-time setup.

**onDestroy()** The system calls this method when the service is no longer used and is being destroyed. Your service should implement this to clean up any resources such as threads, registered listeners, receivers, etc

**2.7.2 Broadcast Receivers**

Broadcast Receivers simply respond to broadcast messages from other applications or from the system. For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action. A broadcast receiver is implemented as a subclass of BroadcastReceiver class and each message is broadcasted as an Intent object

**public class MyReceiver extends BroadcastReceiver { }**

There are following two important steps to make BroadcastReceiver work for the system broadcasted intents:

1. Creating the Broadcast Receiver.
2. Registering Broadcast Receiver

There is one additional step in case you are going to implement your custom intents; then you will have to create and broadcast those intents.

**Creating the Broadcast Receiver**

A broadcast receiver is implemented as a subclass of BroadcastReceiver class and overriding the onReceive() method where each message is received as an Intent object parameter.

**public class MyReceiver extends BroadcastReceiver {**

**@Override public void onReceive(Context context, Intent intent) {**

**Toast.makeText(context, "Intent Detected.", Toast.LENGTH\_LONG).show();**

**} }**

**Registering Broadcast Receivers**

An application listens for specific broadcast intents by registering a broadcast receiver in AndroidManifest.xml file. Consider we are going to register MyReceiver for system generated event ACTION\_BOOT\_COMPLETED which is fired by the system once the Android system has completed the boot process.

Now whenever your Android device gets booted, it will be intercepted by BroadcastReceiver MyReceiver and implemented logic inside onReceive() will be executed.

There are several system generated events defined as final static fields in the Intent class

**android.intent.action.BATTERY\_CHANGED:** Sticky broadcast containing the charging state, level, and other information about the battery.

**android.intent.action.BATTERY\_LOW**:Indicates low battery condition on the device. **android.intent.action.BATTERY\_OKAY**:Indicates the battery is now okay after being low. **android.intent.action.BOOT\_COMPLETED**:This is broadcast once, after the system has finished booting. **android.intent.action.BUG\_REPORT**: Show activity for reporting a bug. android.intent.action.CALL Perform a call to someone specified by the data.

**android.intent.action.CALL\_BUTTON:** The user pressed the "call" button to go to the dialer or other appropriate UI for placing a call.

**android.intent.action.DATE\_CHANGED:** The date has changed.

**android.intent.action.REBOOT:** Have the device reboot.

**2.7.2 Content Providers**

A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class. The data may be stored in the file system, the database or somewhere else entirely. A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions.

**public class MyContentProvider extends ContentProvider{}**

A content provider behaves very much like a database where you can query it, edit its content, as well as add or delete content using insert(), update(), delete(), and query() methods. In most cases this data is stored in an SQlite database.

**Creating Content Providers**

Create Content Provider This involves number of simple steps to create your own content provider.

First of all you need to create a Content Provider class that extends the• ContentProviderbase class.

Secondly, you need to define your content provider URI address which will be used to access the content.

Next you will need to create your own database to keep the content.

Usually, Android uses SQLite database and framework needs to override onCreate() method which will use SQLite Open Helper method to create or open the provider's database. When your application is launched, the onCreate() handler of each of its Content Providers is called on the main application thread.

Next you will have to implement Content Provider queries to perform different database specific operations. Finally register your Content Provider in your activity file using tag. Here is the list of methods which you need to override in Content Provider class to have your Content Provider working:

**onCreate()** :This method is called when the provider is started.

**query()**: This method receives a request from a client. The result is returned as a Cursor object.

**insert()**:This method inserts a new record into the content provider.

**delete()** :This method deletes an existing record from the content provider.

**update()**: This method updates an existing record from the content provider.

**getType()**: This method returns the MIME type of the data at the given URI.

**2.7.4 Layouts**

Layouts View hierarchies that control screen format and appearance of the views.

We can use coordinator layout(introduced in Android studio 2.3.1)

Or Grid layout,linear layout,relative layout etc.

According to need of situation.

**2.7.5 Fragments**

Fragments Represent a behavior or a portion of user interface in an Activity.

**Phase I:** When a fragment gets created, it goes through the following states:

onAttach() onCreate() onCreateView() onActivityCreated()

**Phase II**: When the fragment becomes visible, it goes through these states:

onStart() onResume()

**Phase III**: When the fragment goes into the background mode, it goes through these states:

onPaused() onStop()

**Phase IV**: When the fragment is destroyed, it goes through the following states: onPaused() onStop() onDestroyView() onDestroy() onDetach()

Here is the list of important methods which you can override in your fragment class:  **onCreate()** The system calls this when creating the fragment. You should initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed.

**onCreateView()** The system calls this callback when it's time for the fragment to draw its user interface for the first time. To draw a UI for your fragment, you must return a View component from this method that is the root of your fragment's layout. You can return null if the fragment does not provide a UI.

**onPause()** The system calls this method as the first indication that the user is leaving the fragment. This is usually where you should commit any changes that should be persisted beyond the current user session.

**2.7.6 Resources**

Resources External elements, such as strings, constants and drawable pictures

**2.7.7 Intent and Filters**

An Android Intent is an object carrying an intent i.e. message from one component to another component within the application or outside the application. The intents can communicate messages among any of the three core components of an application - activities, services, and broadcast receivers. The intent itself, an Intent object, is a passive data structure holding an abstract description of an operation to be performed.

**Intent objects**

An Intent object is a bundle of information which is used by the component that receives the intent plus information used by the Android system. An Intent object can contain the following components based on what it is communicating or going to perform:

**Action:** This is mandatory part of the Intent object and is a string naming the action to be performed or, in the case of broadcast intents, the action that took place and is being reported. The action largely determines how the rest of the intent object is structured. The Intent class defines a number of action constants corresponding to different intents. Here is a list of Android Intent Standard Actions The action in an Intent object can be set by the setAction() method and read by getAction().

**Data:** The URI of the data to be acted on and the MIME type of that data. For example, if the action field is ACTION\_EDIT, the data field would contain the URI of the document to be displayed for editing. The setData() method specifies data only as a URI, setType() specifies it only as a MIME type, and setDataAndType() specifies it as both a URI and a MIME type. The URI is read by getData() and the type by getType().

**Category:** The category is an optional part of Intent object and it's a string containing additional information about the kind of component that should handle the intent. The addCategory() method places a category in an Intent object, removeCategory() deletes a category previously added, and getCategories() gets the set of all categories currently in the object.

**2.7.8 Layouts**

There are number of Layouts provided by Android which you will use in almost all the Android applications to provide different view, look and feel

**1 Linear Layout** LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.

**2 Relative Layout** RelativeLayout is a view group that displays child views in relative positions.

**3 Table Layout** TableLayout is a view that groups views into rows and columns.

**4 Absolute Layout** AbsoluteLayout enables you to specify the exact location of its children.

**5 Frame Layout** The FrameLayout is a placeholder on screen that you can use to display a single view.

**6 List View** ListView is a view group that displays a list of scrollable items.

**7 Grid View** GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

**Layout Attributes**

Each layout has a set of attributes which define the visual properties of that layout. There are few common attributes among all the layouts and there are other attributes which are specific to that layout. Following are common attributes and will be applied to all the layouts:

**android:id** This is the ID which uniquely identifies the view.

**android:layout\_width** This is the width of the layout.

**android:layout\_height** This is the height of the layout android:layout\_marginTop This is the extra space on the top side of the layout.

**android:layout\_marginBottom** This is the extra space on the bottom side of the layout. **android:layout\_marginLeft** This is the extra space on the left side of the layout.

**android:layout\_marginRight** This is the extra space on the right side of the layout

**android:layout\_gravity** This specifies how child Views are positioned.

**android:layout\_weight** This specifies how much of the extra space in the layout should be allocated to the View. **android:layout\_x** This specifies the x-coordinate of the layout.

**android:layout\_y** This specifies the y-coordinate of the layout.

**android:layout\_width** This is the width of the layout.

**android:layout\_width** This is the width of the layout.

**android:paddingLeft** This is the left padding filled for the layout.

**android:paddingRight** This is the right padding filled for the layout.

**android:paddingTop** This is the top padding filled for the layout.

**android:paddingBottom** This is the bottom padding filled for the layout

**2.7.9 UI Controls**

**TextView** This control is used to display text to the user.

**EditText** EditText is a pre-defined subclass of TextView that includes rich editing capabilities. **AutoCompleteTextView** The AutoCompleteTextView is a view that is similar to EditText, except that it shows a list of completion suggestions automatically while the user is typing.

**Button** A push-button that can be pressed, or clicked, by the user to perform an action.

**ImageButton** AbsoluteLayout enables you to specify the exact location of its children.

**CheckBox** An on/off switch that can be toggled by the user. You should use checkboxes when presenting users with a group of selectable options that are not mutually exclusive.

**ToggleButton** An on/off button with a light indicator.

**RadioButton** The RadioButton has two states: either checked or unchecked.

**RadioGroup** A RadioGroup is used to group together one or more RadioButtons.

**ProgressBar** The ProgressBar view provides visual feedback about some ongoing tasks, such as when you are performing a task in the background.

**Spinner** A drop-down list that allows users to select one value from a set.

**TimePicker** The TimePicker view enable users to select a time of the day, in either 24-hour mode or AM/PM mode.

**DatePicker** The DatePicker view enable users to select a date of the day.

**2.8 EVENT HANDLING**

Events are a useful way to collect data about a user's interaction with interactive components of your app, like button presses or screen touch etc. The Android framework maintains an event queue into which events are placed as they occur and then each event is removed from the queue on a first-in, first-out (FIFO) basis.

There are following three concepts related to Android Event Management:

**Event Listeners:** The View class is mainly involved in building up an Android GUI, same View class provides a number of Event Listeners. The Event Listener is the object that receives notification when an event occurs.

**Event Listeners Registration:** Event Registration is the process by• which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.

**Event Handlers:** When an event happens and we have registered and• event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

**OnClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick() event handler to handle such event.

**OnLongClickListener()** This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use onLongClick() event handler to handle such event.

**OnFocusChangeListener()** This is called when the widget loses its focus i.e. user goes away from the view item. You will use onFocusChange() event handler to handle such event.

**OnFocusChangeListener()** This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such event.

**OnTouchListener()** This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such event.

**OnMenuItemClickListener()** This is called when the user selects a menu item. You will use onMenuItemClick() event handler to handle such event.

**2.9 ANATOMY OF ANDROID APPLICATION**

**src** This contains the .java source files for your project. By default, it includes an MainActivity.java source file having an activity class that runs when your app is launched using the app icon.

**gen** This contains the .R file, a compiler-generated file that references all the resources found in your project. You should not modify this file

**bin** This folder contains the Android package files .apk built by the ADT during the build process and everything else needed to run an Android application.

**res/drawable-hdpi** This is a directory for drawable objects that are designed for highdensity screens.

**res/layout** This is a directory for files that define your app's user interface.

**res/value**s This is a directory for other various XML files that contain a collection of resources, such as strings and colors definitions.

**AndroidManifest.xml** This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

**2.10 SECOND APPLICATION**

**2.10.1 Databases:**

For the second project, I needed to learn a new thing: how to use databases in Android. I can use databases for JEE, but in android is different. In fact there are ready-made functions to manipulate sqlite which is the database built into android. In this part of the report, I will not re-explain the mechanisms already explained that were resolved such as how to display a textview or a list.

To use a database, we will create a class called Helper. This class will allow us to manipulate the database from any other class that has instantiated the object Helper. This class has elements and methods very specific .

First specific objects: a SQLitedatabase, and a class called openHelper that we will also create.

**private SQLiteDatabase db;**

**OpenHelper openHelper = new OpenHelper(this.context);**

**this.db = openHelper.getWritableDatabase();**

The class OpenHelper extends SQLiteOpenHelper. This class is used to actually create one table or several tables in a database, and fill the table in the moment of its creation, al this in the method onCreate (). this class allows to update the version of the table with the method onUpgrade (). The method OnCreate will be called only once, after that the table is created this method will no longer serve. For the class Helper, you can add all the methods used to select, add, edit or delete entries in the table.

**2.11 WHAT DID I LEARN?**

**2.11.1 TECHNICALY**

First of all I gained additional skills in the Java programming language that was difficult for me before. I learned also how to use a lot of the components in eclipse such as the debugger. I also learned a new language that is XML, a language I didn't know at all earlier. Finally, this project allowed me to use my sql skills acquired during my studies

**APP Behaviours:The Callback methods**

**onCreate()** This is the first callback and called when the activity is first created.

**onStart()** This callback is called when the activity becomes visible to the user.

**onResume()** This is called when the user starts interacting with the application.

**onPause()** The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed. **onStop()** This callback is called when the activity is no longer visible.

**onDestroy()** This callback is called before the activity is destroyed by the system.

**onRestart()** This callback is called when the activity restarts after stopping it.

**2.11.2 HUMANLY**

This Training brought me a lot of things on a human level. First in relation to it myself. I had to acquire a large autonomy. Indeed I knew nothing about android beforre, and I had to learn how to use, and create things on android by myself. This brought me a lot. This autonomy is also linked to a research spirit. I had to try to find examples by myself in the books and UDACITY, try to test new things, for having nowhere to find answers. Finally I was able to acquire through this course and also thanks to the project Portal.

**CHAPTER-3   
 PROJECT WORK**

**INTRODUCTION**

College Portal application is designed to connect our college students with teachers and higher Authorities in one app. This App includes Registration of students and teachers. Students and Teachers has got different type of view of app.Teachers can control the students Registrations and logins,meanwhile a student can only be registered if it is approved by a teacher. Thus I’ve tried to implement a student teacher relationship by providing a must needed teachers control.

This app will also provide teachers the info about the number of students who has applied to get approved, how many of them has got the approved and also how many are still needed to be approved,this is shown arithematically as well as graphically.

One of the coolest feature of this app is it is a location aware app.I’ve made this app location aware in the contrast that whenever student come to a particular location say our college library, it will make their Mobiles silent,if installed.

Also,There is somethimg more for teachers that they can upload files as lecture notes,which will help students for their concepts in future while preparing for exams and also for interviews.

Students can be notified for college events, functions, or any type of emergencies with one click.It will help to make it easy.

Also, there is an option to make this feature optional for students whether they want to use this silent feature or not,making this a user friendly app.

Last but not the least,Teachers and students can communicate with each other just by clicking a floating button icon,separately.

**3.1 Existing System**

I’ve seen many apps like this but no app has these features in one app. Also there is no any app that could provide us that silent feature which is location aware.Students often forgets to put their phones on silent mode which results in disturbance to other people.well,that is what technology made for, To make our lives easier,simply.The features can be found in 3 or 4 separate apps but we’ve integrated them into one app.

**The Problems with existing System are as follows:**

1. Wastage of data
2. Higher the count of application,Higher they will consume RAM
3. Higher the count of apps, Higher the memory requied to keep them and install them
4. We can’t keep large number of apps running in our phone as it will cause disturbing effects,Draining battery being one of them.

**The Main Objective of App**

1. To establish a connectivity between teachers and students
2. To make students aware of college activities
3. To provide college a product which can be refined later.
4. Decrease in hesitation to talk together
5. More consistent Data Handling
6. A user friendly system, which doesn’t require any expert training.

**System Design**

System design contains Logical Design & Physical Design. Logical designing describes the structure & characteristics or features, like output, input, files, database & procedures. The physical design follows the logical design, actual software & a working system. There will be constraints like Hardware, Software, Cost, time & Interfaces. The most interesting part of the development of the project is the system design. System design is to deliver the requirements as specified in the feasibility report. The main objectives of the design are:

1. Practicality
2. Efficiency
3. Cost
4. Security

Structured design is a data flow methodology. The graphical representation of data flow, communication & defining the modules & their relationship with each is known as Structure Chart.

**Planning Phases**

Planning was needed to get to know that what to do and how to do?it sounds very time consuming and hectic but thanks to fluidui.com,It saved our a lot of time to plan our app efficientely in a feasible way.

**Problem Recognization**

A problem is well defined very rarely. It corps out with a vague feeling of some statements that lead to vague conclusions. So the first task is to get more crucial information by interviewing and meeting concerned people. It clarifies how the problem is felt, how often it occurs, how it affects the business and which departments are suffering with this. This phase consists of the following tasks.

**Problem Defination and Investigation**

This was a preliminary investigation done with a view to have a “feel” of the working of the proposed system. This phase has been identified the end-user directly involved in the system who were the managers, assistant officer and database administrator, and the development department. By understanding the working of database, its flow and also after conducting meetings and interviews with the concerned persons, a clear idea about the working was obtained. An explanatory note that serves to gain cooperation and avoid misunderstanding by setting out the purpose of the exercise clearly accomplishes each questionnaire. Observation technique is also used for fact finding. The work described at the time of interview is observed personally ads it reduces the chance of misunderstanding and omissions.

By the end of this phase, idea as to how the information enters the system, how it is stored, how it is processed, how information changes affects the working of the system

**Detailed Study of existing system**

This phase provides the overall requirement for the system what is to be done. Input for this phase is the information collected through several data collecting schemes such as survey, cross-questioning-answering etc and the raw data obtained which is not properly ordered and not in the precise manner.After collecting all the information and requirements, they were verified from the concerned persons by presenting a diagrammatic version of the proposed system. The points missing were added to the system specifications for the desired system. So this final document provides the system requirement specifications for the desired system. It helps in reducing the total development cost and also establishes the various points for validation and verification.

**3.2 Rectification in the new system**

Proposed system is expected to remove the problems of the existing system and provide value-added solution to the problem. Due to limitation in the existing system it is felt to switch over to a new system that covers all the possible aspects:

* **Availability:**

System will be available around the clock.Users can get information online any time.

* **Portability:**

The application is developed in JAVA. It would be portable to other operating system. Framework is available for the OS. As the database is made in Firebase, All the data is coming from Internet server, so it is easy to get info at anytime

* **Usability:**

The links to chat/discussions has been provided and are just one click away.Also it is user friendly and convenient app just of few mbs.

* **Security:**

Application will allow only valid users to access the system. Access to any application resource will depend upon Teacher designation. Security is based upon the Teacher user ID and Password for database at backend and Android Studio at front end.

**3.3 Data flow Diagrams**

Often they are a preliminary steps used to create an overview of the system which can later be elaborated.It shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored

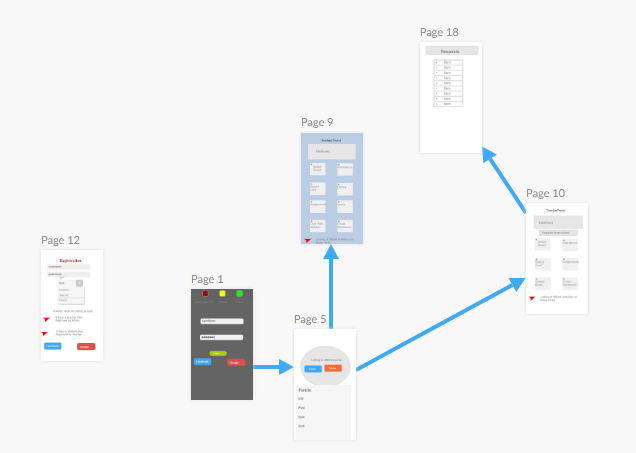
****

Fig:3.1-Data Flow Diagram

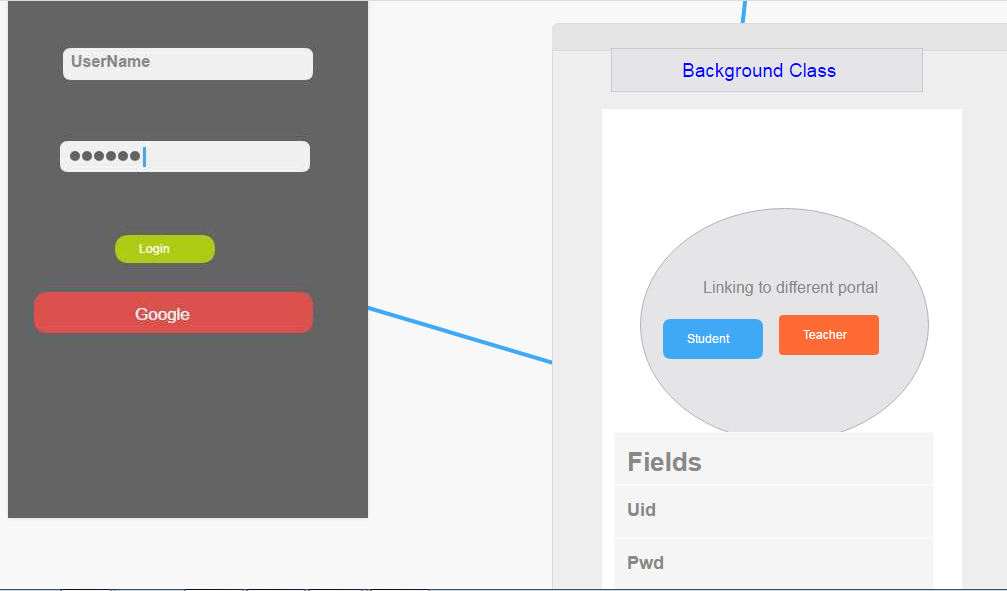


Fig:3.2-DFD of Login

**3.4 A look of our App UI**

**Login page**

Loginpage is here ,there are two fields named email and password.These two fields are Input Layout text views, there are proper validations on them eg: password less than 6 digits is not applicable.you can also login with google.

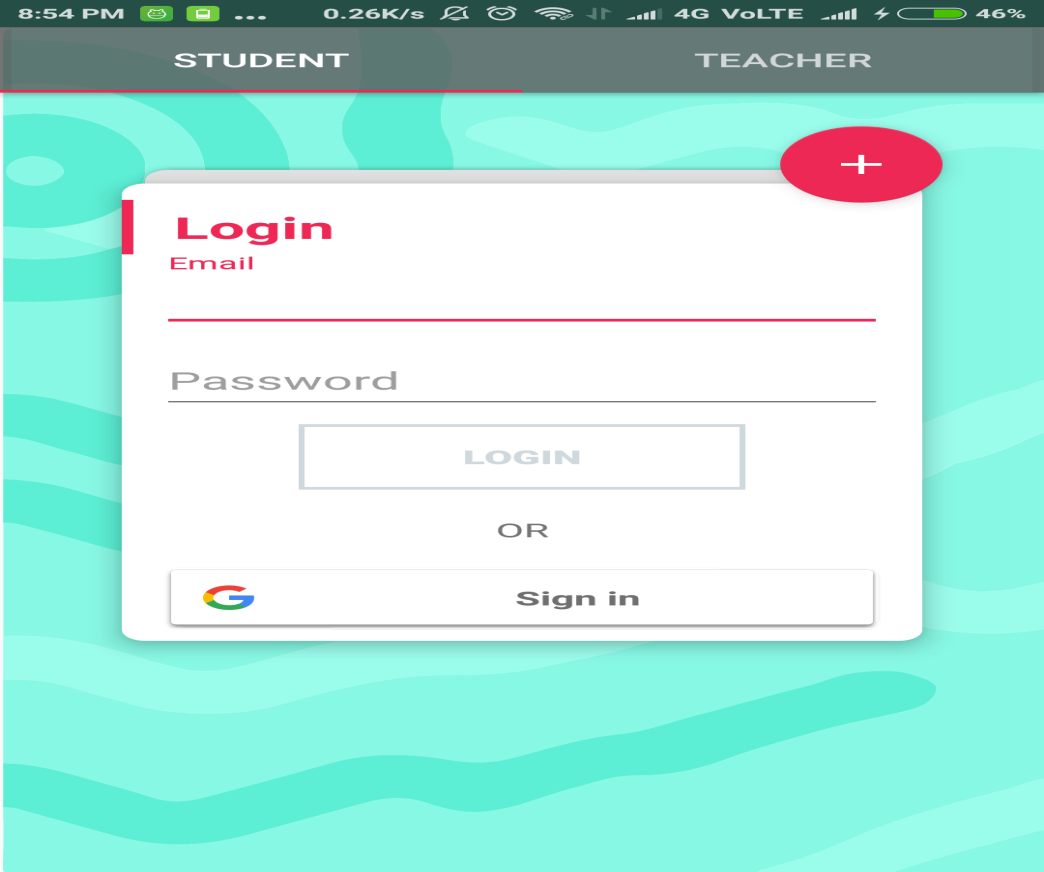


Fig :3.3-Login Page in A Tab Layout

Or you can register with a user name and password via the **floating action button** which will animate on that screen, and you can then again login with that user name and password on the login screen.

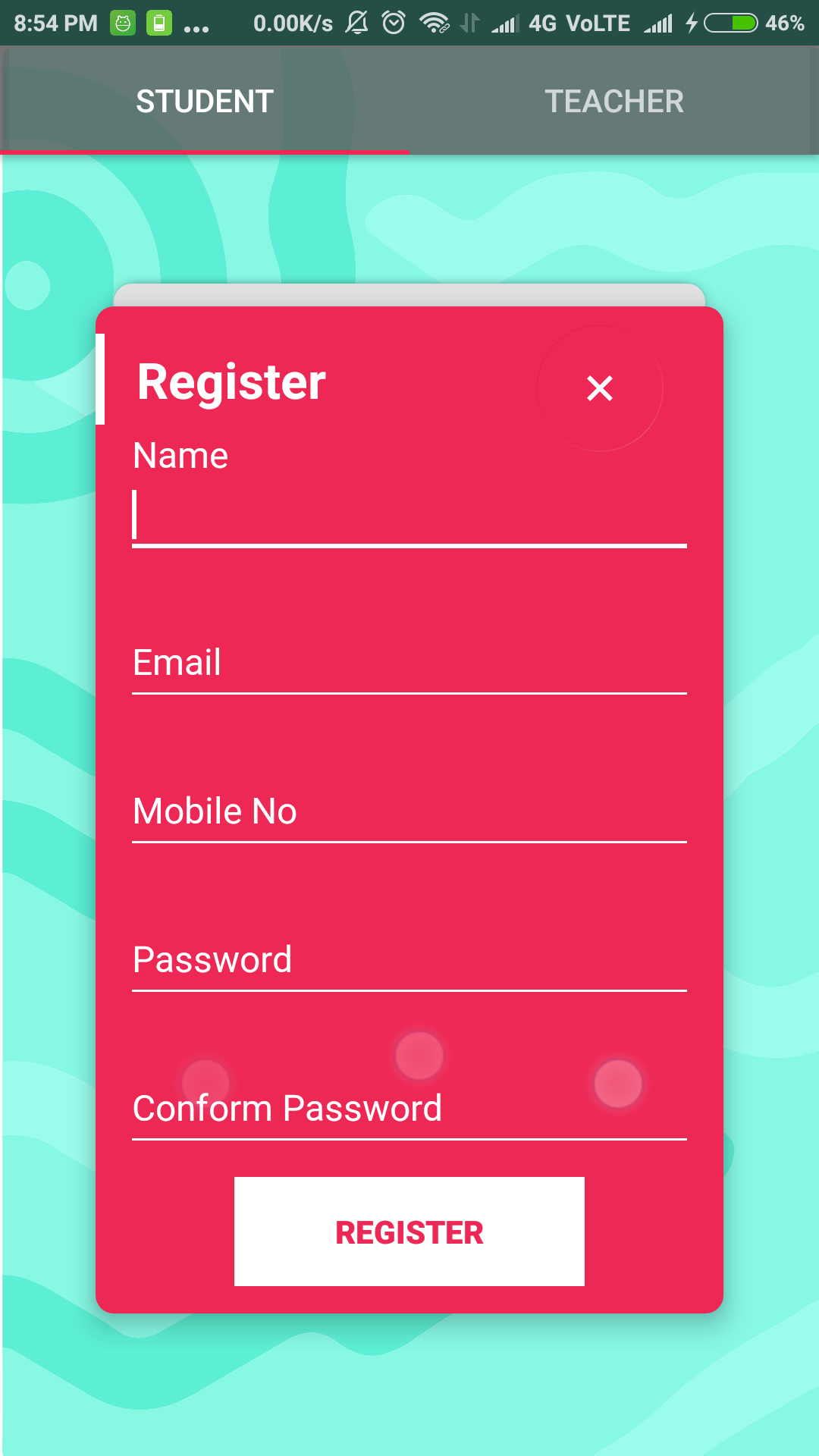


Fig:3.4- Registration Page

**Main activity screen**

After login students and teachers will be directed to their homescreens respectively.students will get the info about whether he is approved or yet to be approved. Teachers will be directed to a activity that will show them about the number of students have applied to get approved.Teachers are need to Approve the students for the successful login for students profile to get approved and take advantage of facilities

Also there is a dashboard on that various activities will appear accordingly.

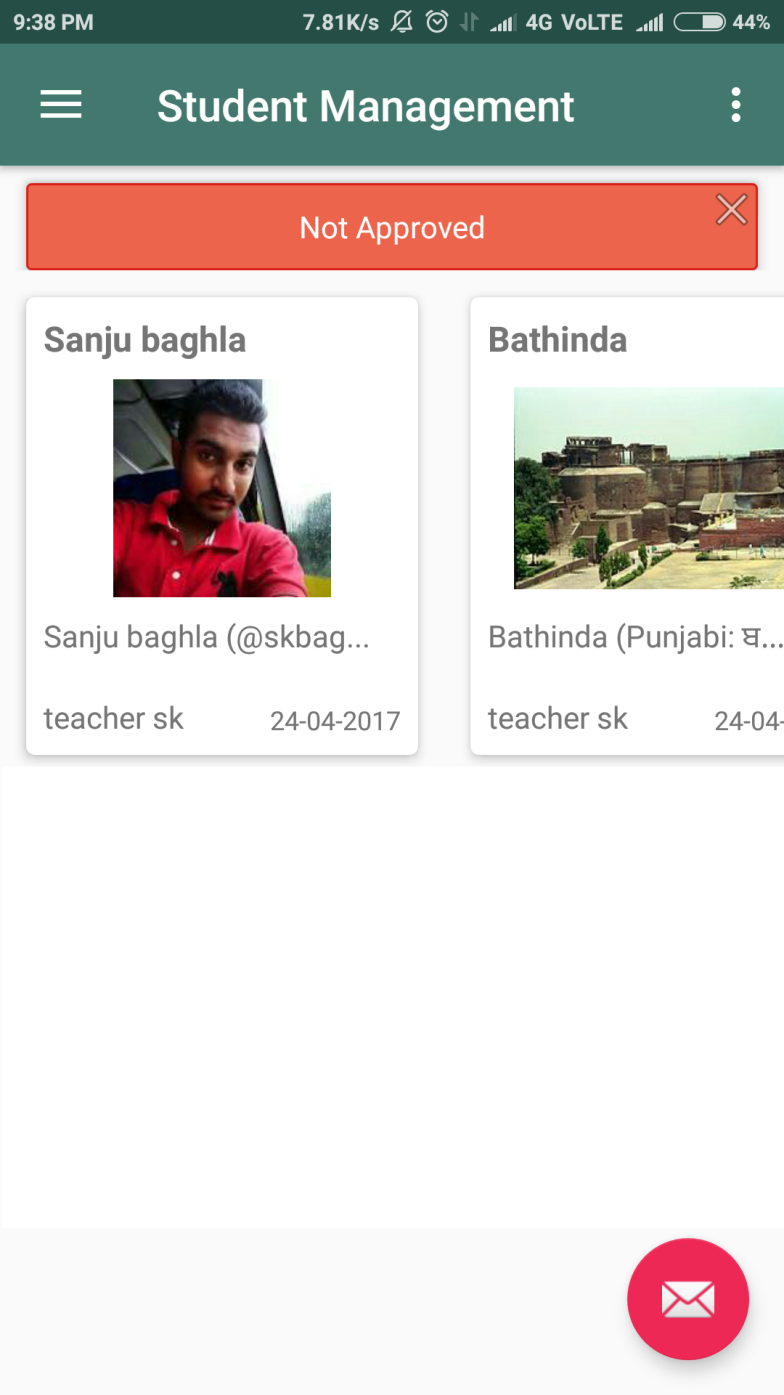


Fig:3.5-Home Page of Student

Also there are floating action buttons for both accounts to chat or discussion which will lead them to their chat window.

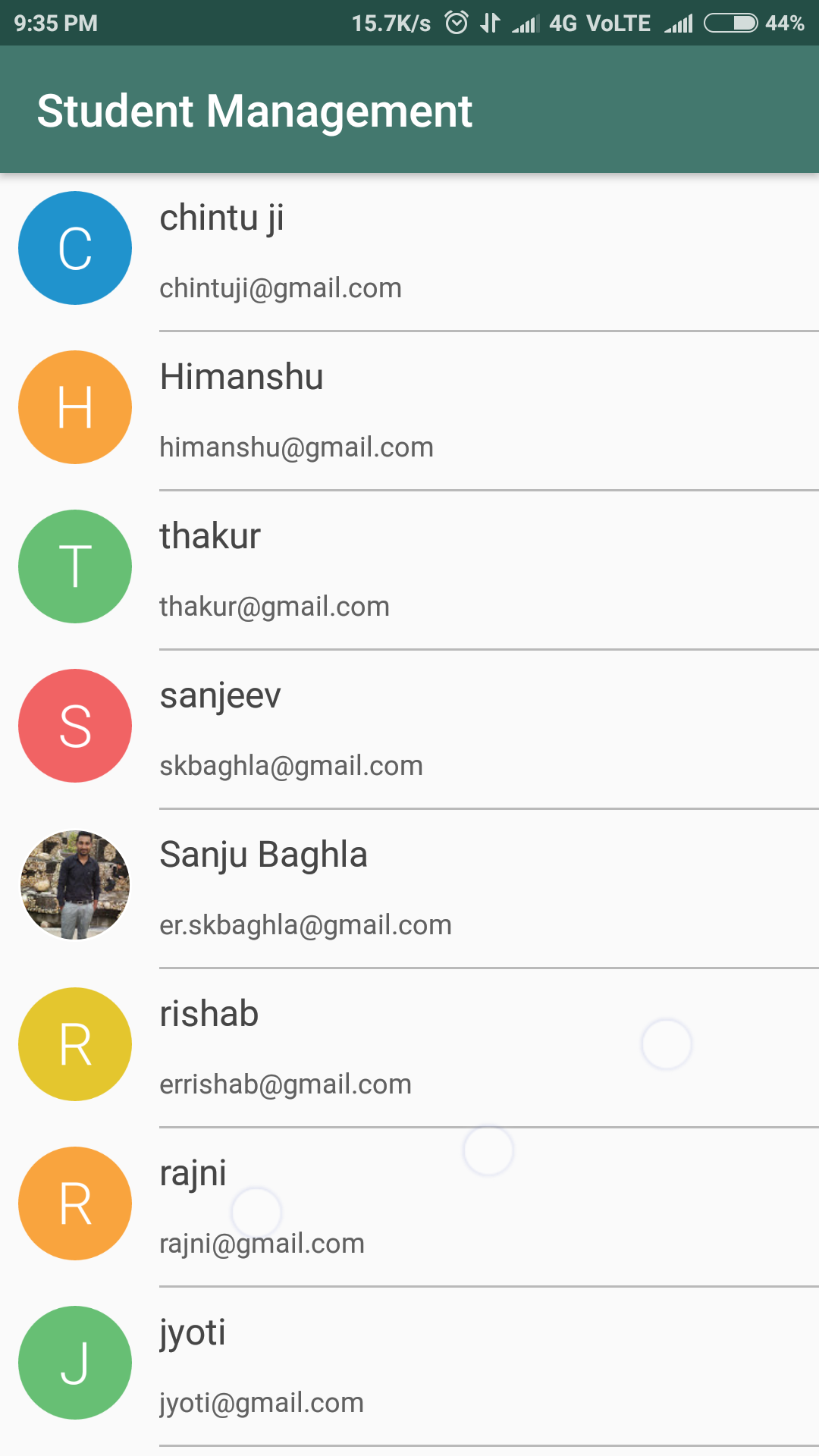


Fig:3.6-List of Teachers For Chat

There is navigation bar,which is slide left and consists of sections like notes,gallery and a floating calculator which opens when clicked and do their respective works as their name suggests.

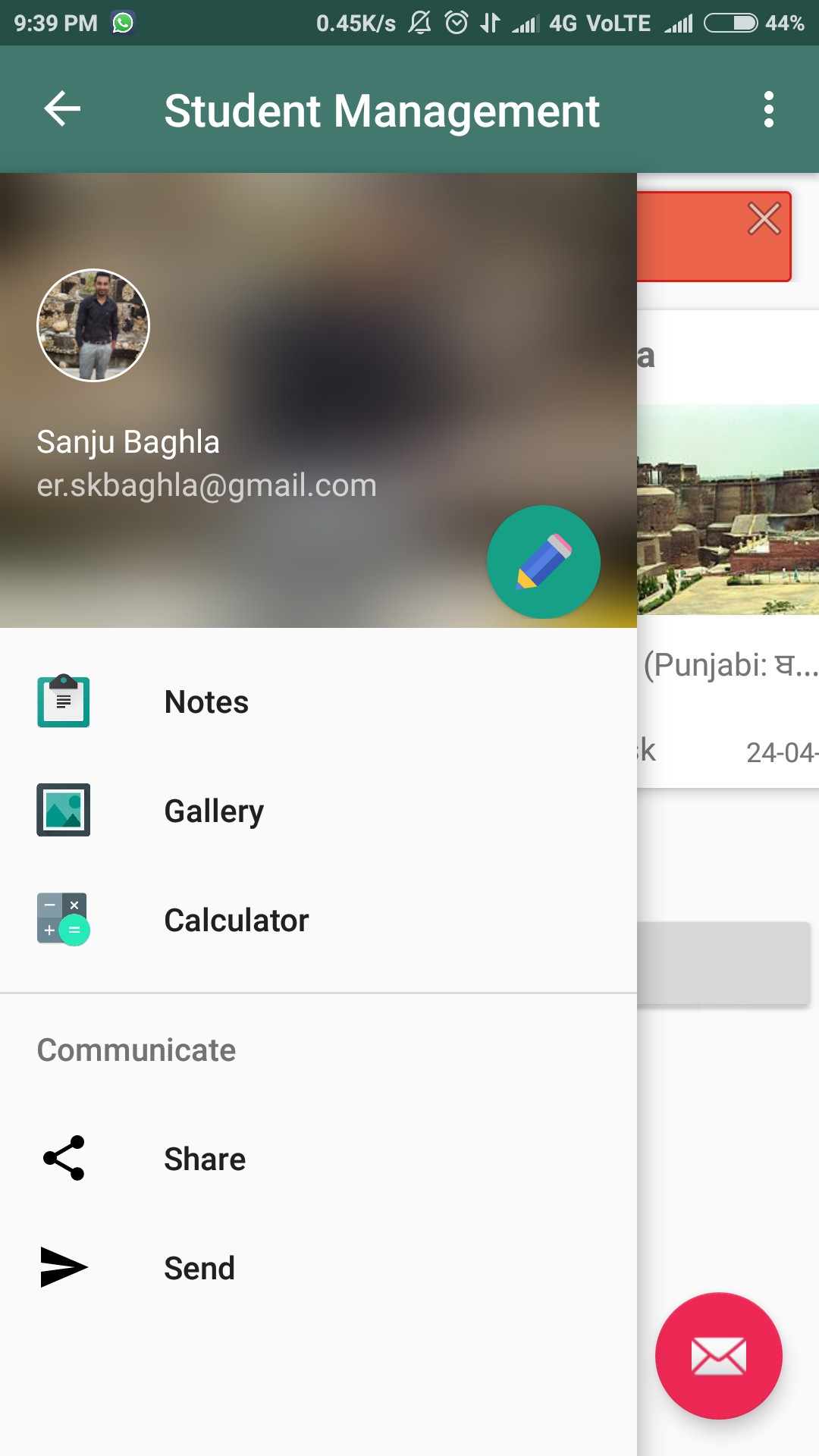


Fig:3.7-Student Left Drawer

Also an option to edit one’s profile that will take them to edit profile section of our app,they can edit their profile accordingly.

Edit profile link is also provided in toolbar along with settings option,in which user can turn the location feature on or off.

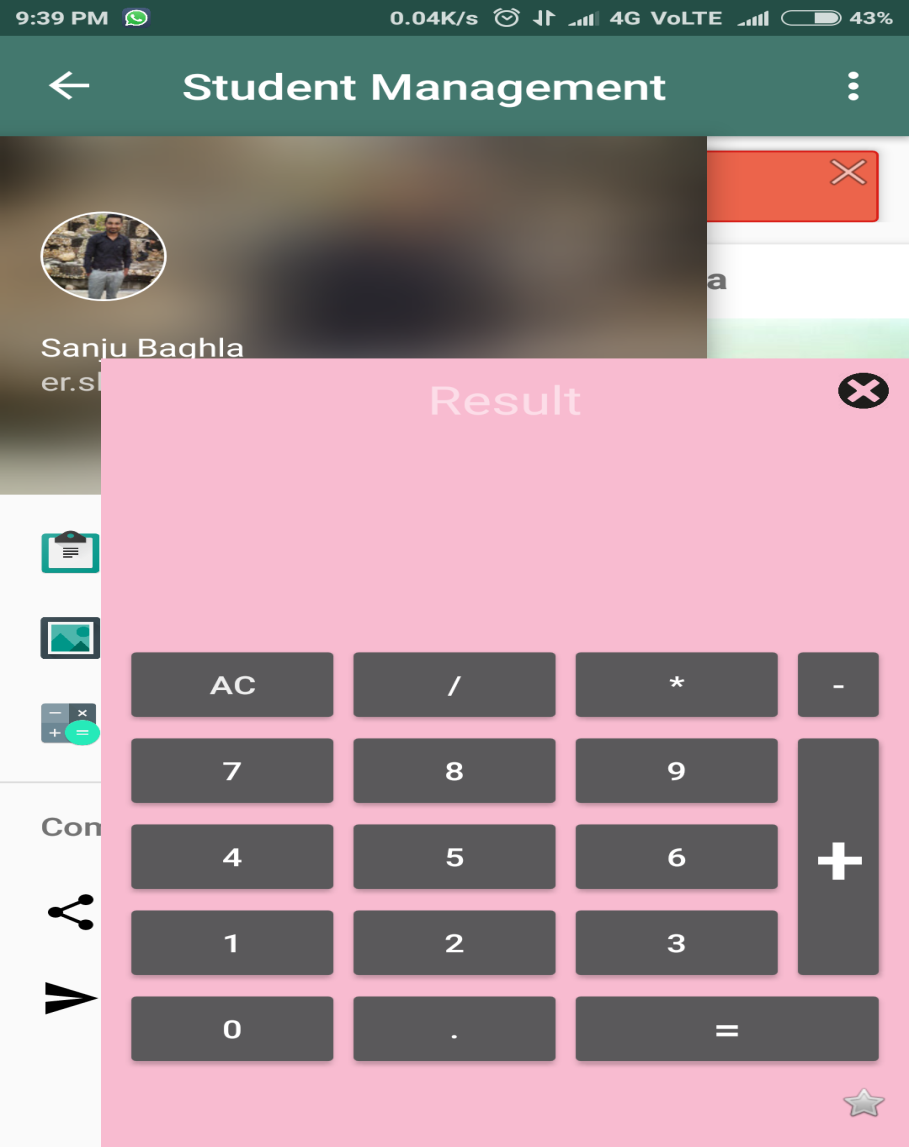


Fig:3.8-Floating Calculator

Last but not the least there is signout button,we all know about it that it sign out the user.

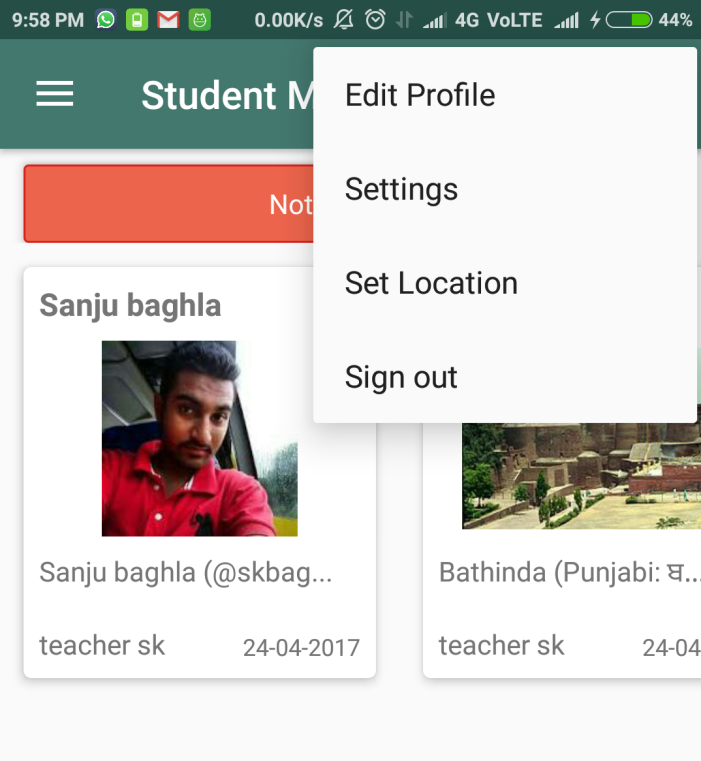


Fig:3.9-Toolbar Options

**Teacher Homepage**

Teachers will directed to a activity in which they can check how many students need to be approved. They can see the info arithematically as well as graphically.

Also in the navigation bar there are links to watch approved or non approved students.

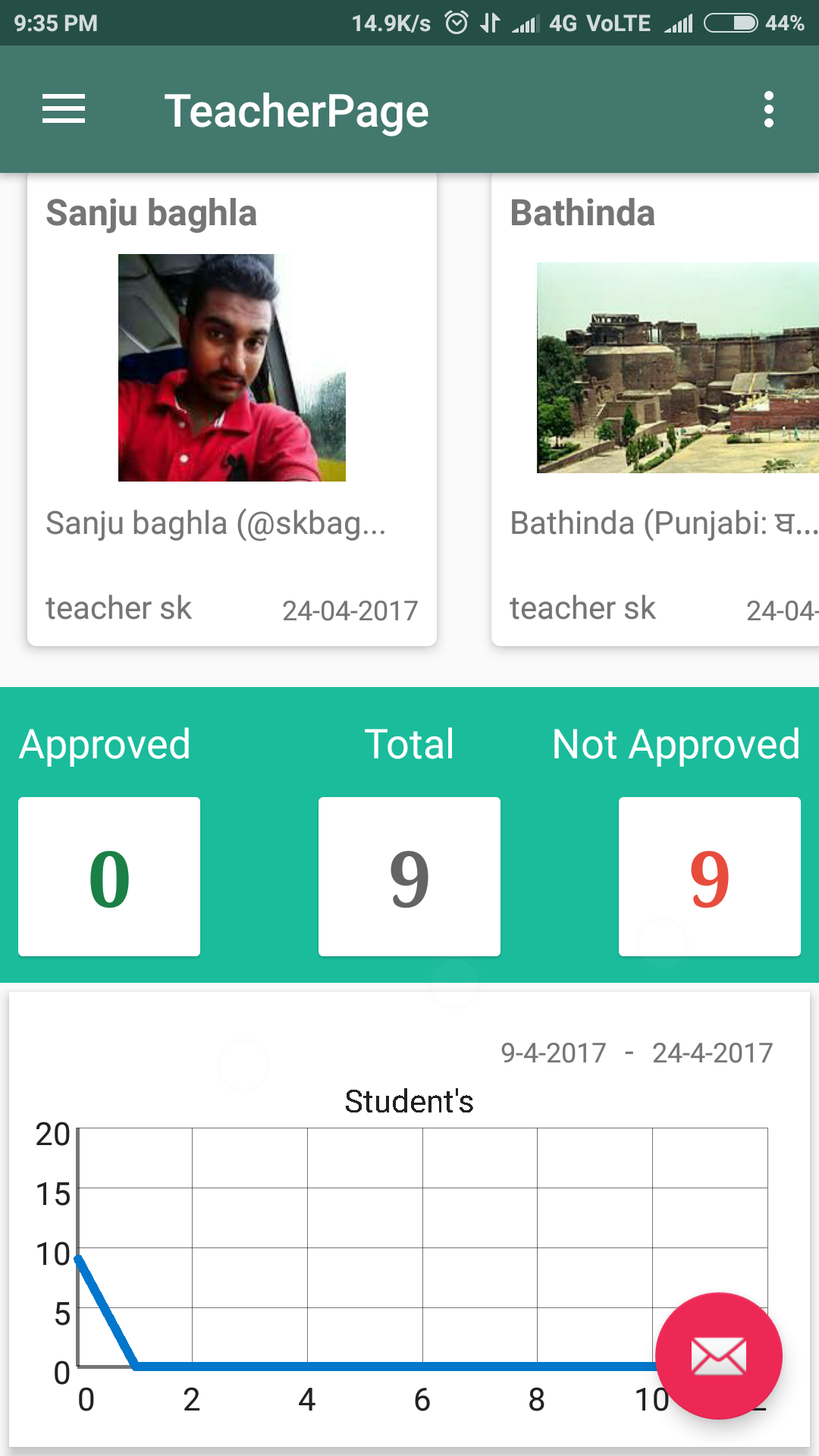


Fig:3.10-Home Page of Teacher

There is a link to upload notes which can be then seen by students.

Also, a link to put notice on notice board activity which would be then shown to students on their notice boards.

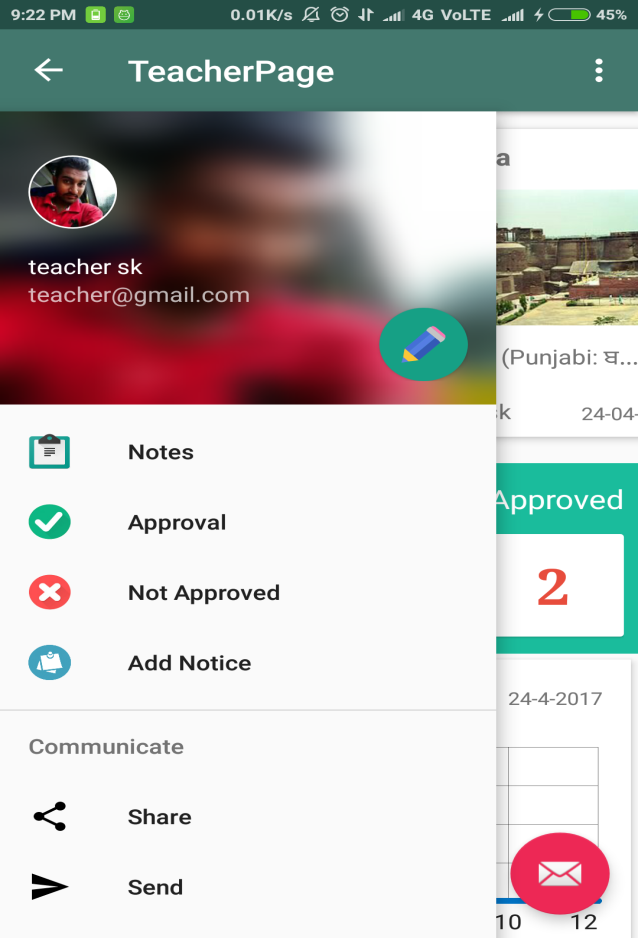


Fig:3.11-Teacher Left Drawer

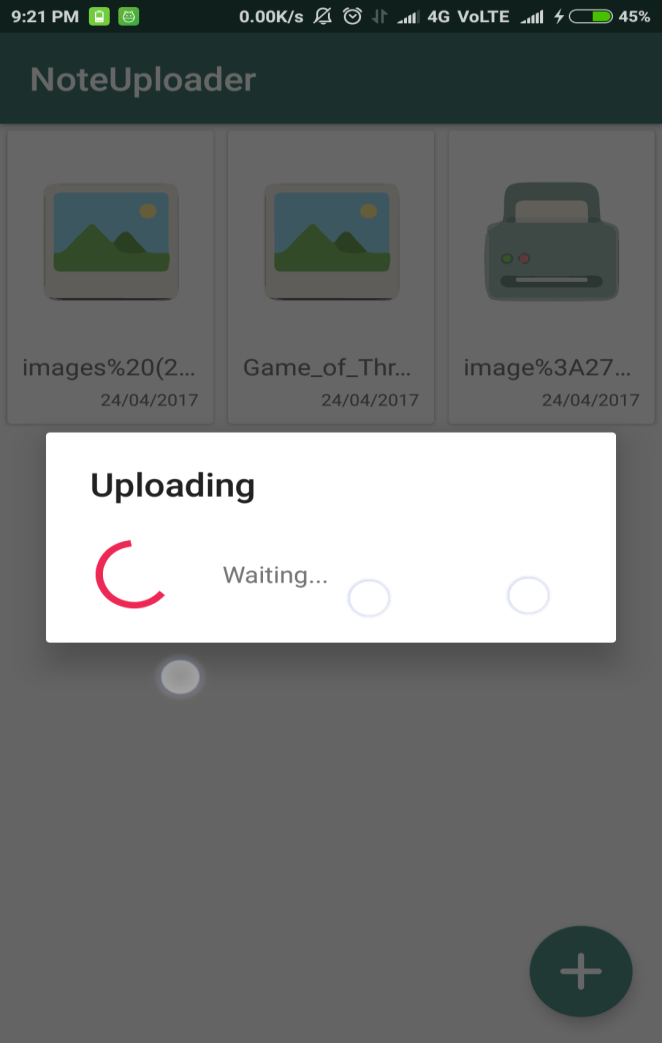


Fig:3.12-Notes uploader Activity

There is an Activity Add Notice in which we can add notice with Title,Desc and a Pic for the dashboard

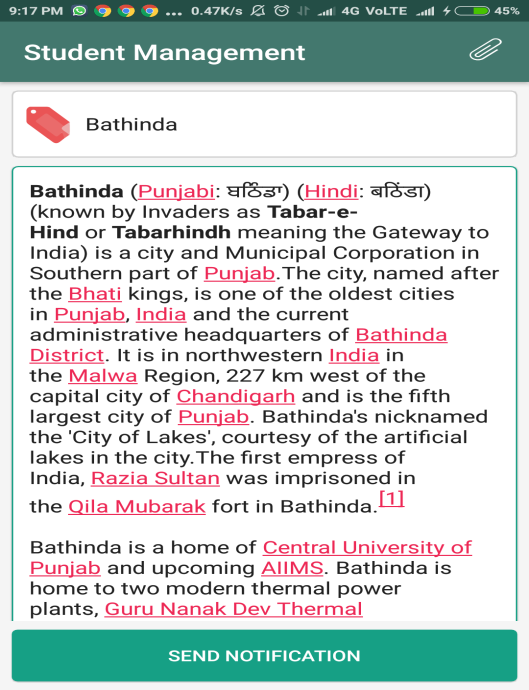


Fig:3.13-Notice Board Activity

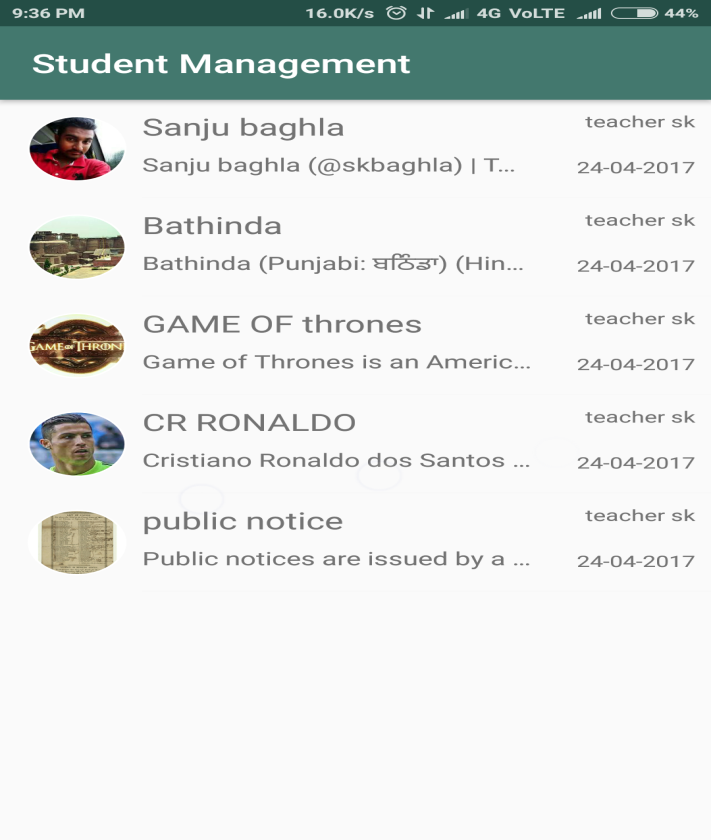


Fig:3.14-List of Notices in th Dashboard

**3.5 A HIGHLIGHT TO APP FEATURES:**

* Auto Silent mode when needed,if installed and not unchecked
* Separate accounts for teachers and students
* Teachers approving students registrations
* Teachers adding notes and notices on the dashboard
* In-Built Floating Calculator
* A chat app included to work as discussions between teachers and students.

**CHAPTER-4  
 CONCLUSION**

Thanks to this Training, which was also planned for this, I discovered a technology, a culture

and beautiful people also learned new skills, and used all the things that I learned for two years. I do not regret anything in this experience and I thank all those who have allowed such a thing feasible. The people I met here are not just classmates, roommates and professors, they are friends, and once again I thank you all.

**CHAPTER-5  
 REFERENCES**

**Websites:**

console.developer.google.com

developer.android.com

console.firebase.google.com

[www.stackoverflow.com](http://www.stackoverflow.com)

[www.AndroidAuthority.com](http://www.AndroidAuthority.com)

[www.Androidhive.com](http://www.Androidhive.com)

http:://rcbproject123.000webhostapp.com