MOUA – MOBILE OFFICE UTILITY APPLICATION

An office application that aims to combine notepad, calculator, calendar, chat and task managing features into one mobile app.

Built With – Android Studio & Android SDK

- Frontend & UI
  - XML
- Backend
  - Java
- Database Layer (Under-Construction/TBA)
  - SQL Lite
  - SQL Helper

How to Use – Download the OfficeApp-Installer.apk package to your Android Phone and install

GitHub Repository: https://github.com/muhammad-abdulqayyum/Mobile-Office-Application

Direct Download Link: https://github.com/muhammad-abdulqayyum/Mobile-Office-Application/raw/master/android-installer-file/OfficeApp-Installer.apk

By Muhammad Abdul-Qayyum, Kevin Harmon, Kwaku Saka

Table of Contents

INTRODUCTION

This section gives a scope and description for the project. It is an overview of all things covered in our groups Software Requirements Specification document. This document will feature the purpose, features, the interfaces and what the system will do, constraints of the application, and what application will do when given external stimuli. Some other components of this are the abbreviations and definitions used in our report and the purpose of our project.

PURPOSE

The goal of this report is to give a detailed look on what is required for the "Mobile Office Utility Application" (MOUA) software. This report will give detail to the applications features and components, as well as shed light on future components that may be added to the project.

SCOPE

This software is meant for businesses who want disseminate information and ideas throughout an organization. It does this by way of a speedy and user friendly user interface. This application is ideal because of the accessibility of the layout and it's multi-purpose functionality. The application will allow users to share notes and sketches within an organization. The application will also have other features such as a calendar which will let users schedule meetings and days off. The application will also have a calendar and chat/task management features

DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

Terms	Definition
User	Someone who interacts with mobile phone application
MOUA	Mobile Office Utility Application
User ID	A number which corresponds with the user in database that allows user to save and share information.
Position	The users position in the company.
Database	Collection of the information monitored by the application

Email	Company Email Address put into database for the use of sharing information.
User Name	Given user name set by company policy which will allow the user a name to log in with.
Password	Given password that goes with user name and user ID which is stored in database.

REFERENCE

OVERVIEW

The rest of the following document contains all of the information such as the code for the application, which hardware and software environments support the application, and all the functions of the application. The SRS is compiled by first explaining the functions and usability of the application. Then we later delve deeper into the technical side with an explanation of functions and the code.

OVERALL DESCRIPTION

The application has many uses and functions that will better help you organize and make your day manageable. There were many learning challenges to building the application that will be further expressed throughout the document. The group had to preparer by learning coding languages and structures needed for the project.

PRODUCT PERSPECTIVE

This application was created to be mobile center for communication within a company. There are other application with similar options like Evernote, Microsoft OneNote, and SimpleNote. With this application we will allow users to improve communication between all departments in their company. We will allow users to have a great system to manage their time and sync with their and other departments. This allows mobility to users to access notes and communicate with coworkers from anywhere.

PRODUCT FUNCTIONS

CALCULATOR

- The Calculator is a basic calculator with add, subtract, divide and multiply functions. More advanced math functions may be added in future releases of the application.

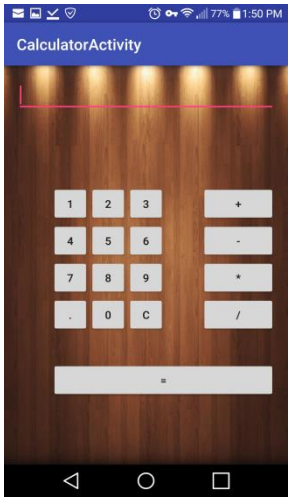


Figure 2.2.1

CALENDAR

- MOUA will allow you set dates, reminders, and schedule days with our calendar section of the app.
- MOUA users will be able to schedule and send days off to Human Resources department all from the app.

Figure 2.2.2

NOTEPAD

- MOUA will have a notepad in which you can save notes of any length and share with multiple members of your organization. The application will have an easy to use interface with multiple font and size options for writing notes.

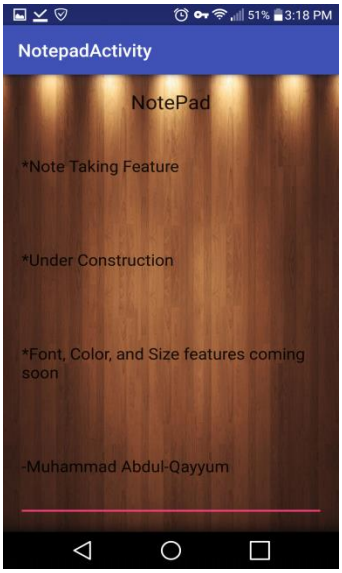


Figure 2.2.3

CHAT & TASK

- MOUA will have a chat feature that will allow employees in an organization to send secure messages to each other by registering accounts and signing in. We also aim to include a task management feature that will allow users and employees to assign tasks to each other, and set due dates for those tasks to be completed.

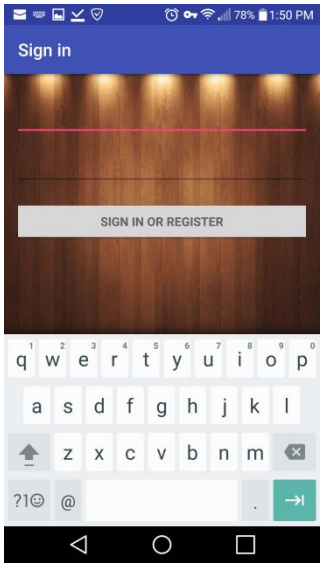


Figure 2.2.4

USER CHARACTERISTICS

Our application is simple and easy to use. Anyone who can use a phone can use this application. The application targets workers and employees, however any general user can find use from the applications calculator, calendar and notepad features. Potentially even home users can find use from the task

management features of the application by assigning chores to family members, and by using the app to keep track of due dates.

SYSTEM REQUIREMENTS

This application requires an Android Device to run and work properly. The minimum android version required for the application to install is Android 4.0.3 – IceCreamSandwich. This version has a market penetration of 97.8% so the vast majority of Android smartphone users will be able to use this application.

The application currently does not have an iPhone version; however iPhone support may be added down the line.

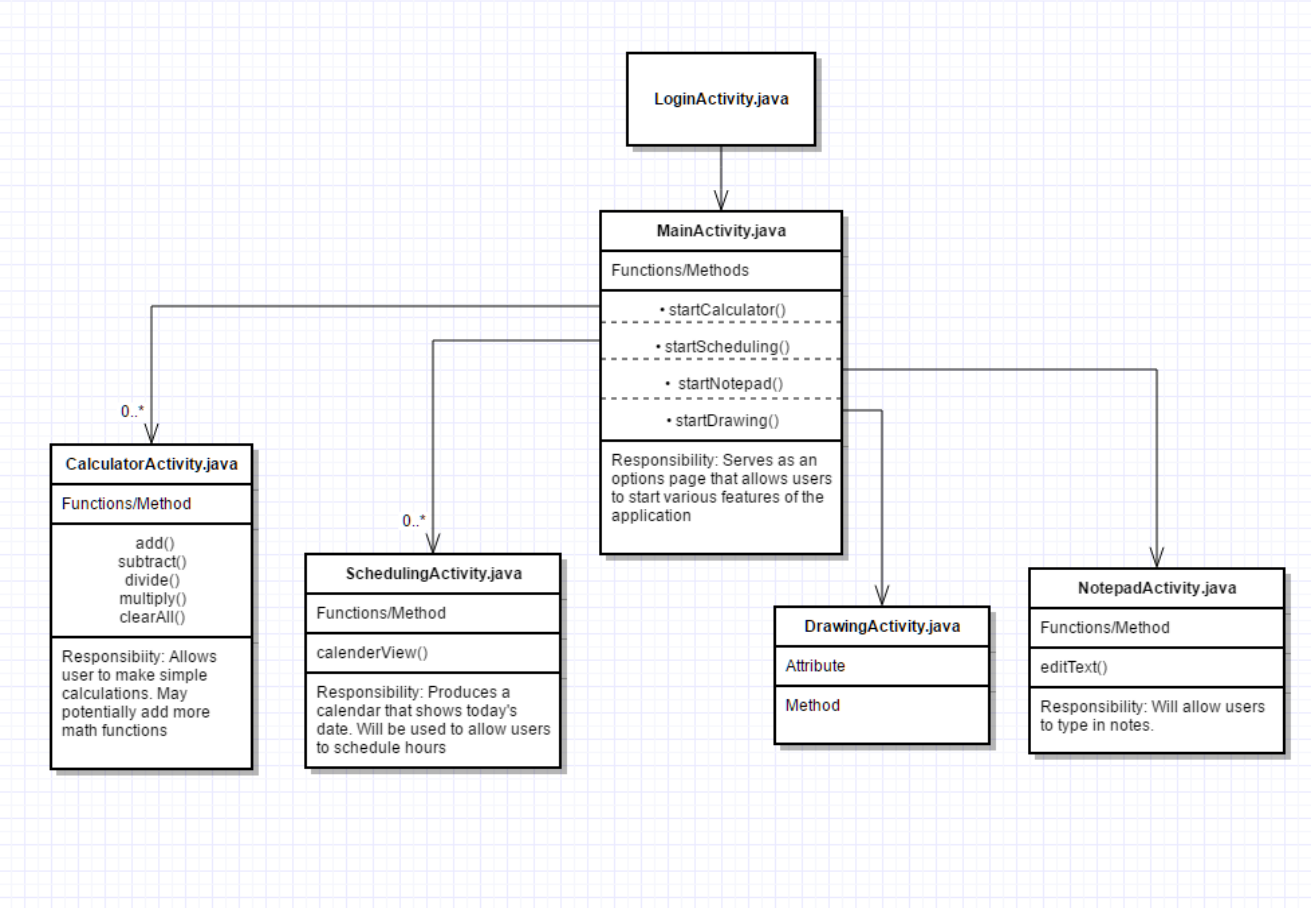
This application runs on both smartphones and tablets.

SPECIFICATION REQUIREMENTS

CLASS DIAGRAM

The diagram below is a Class Diagram. This diagram shows the major activities of the application. This figure is used as a basic framework of how we will create the project. It holds the classes required classes and the functions that make up those classes. The diagram also shows data flow and the paths the user will be able to take as they use the application.

Figure 3.1



USE CASE DIAGRAM

The Use Case Diagram below shows the use case of our system for multiple users. I will further describe the figure and use cases here in this portion of the report.

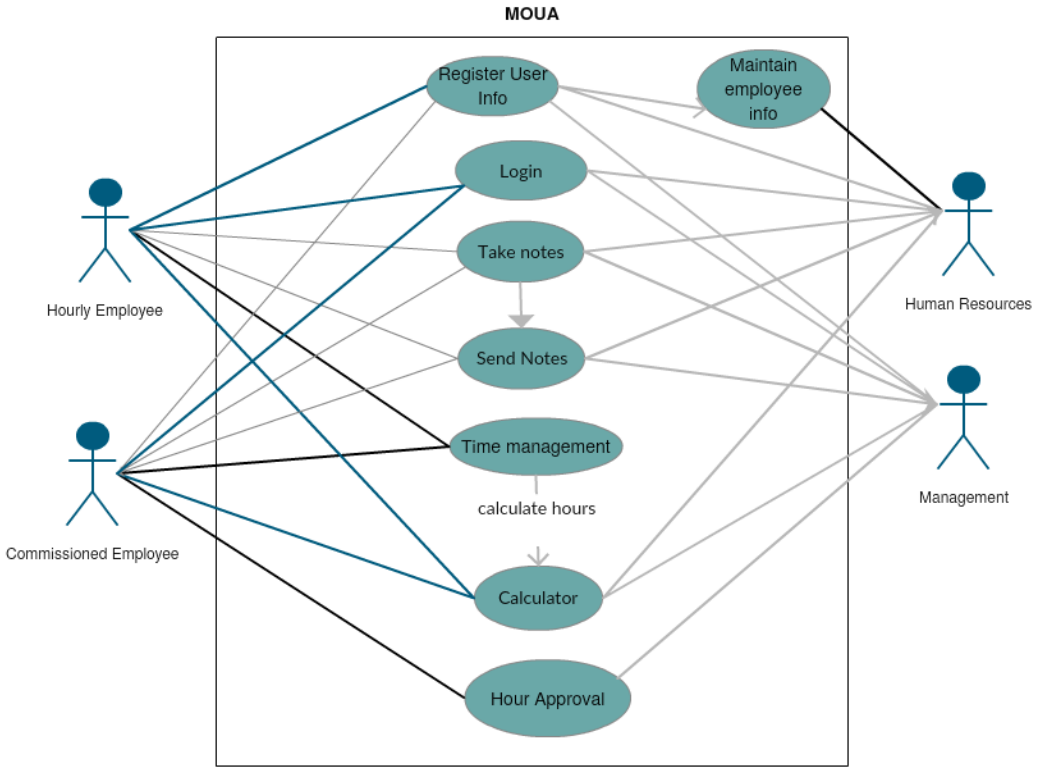


Figure 3.2

CRC DIAGRAM

In this section we have the CRC Diagram (Class Responsibility Collaborator) which is a diagram that shows the classes and their responsibilities and which other classes they collaborate with. As you can see in the diagram below all of classes interact with the Main Class.

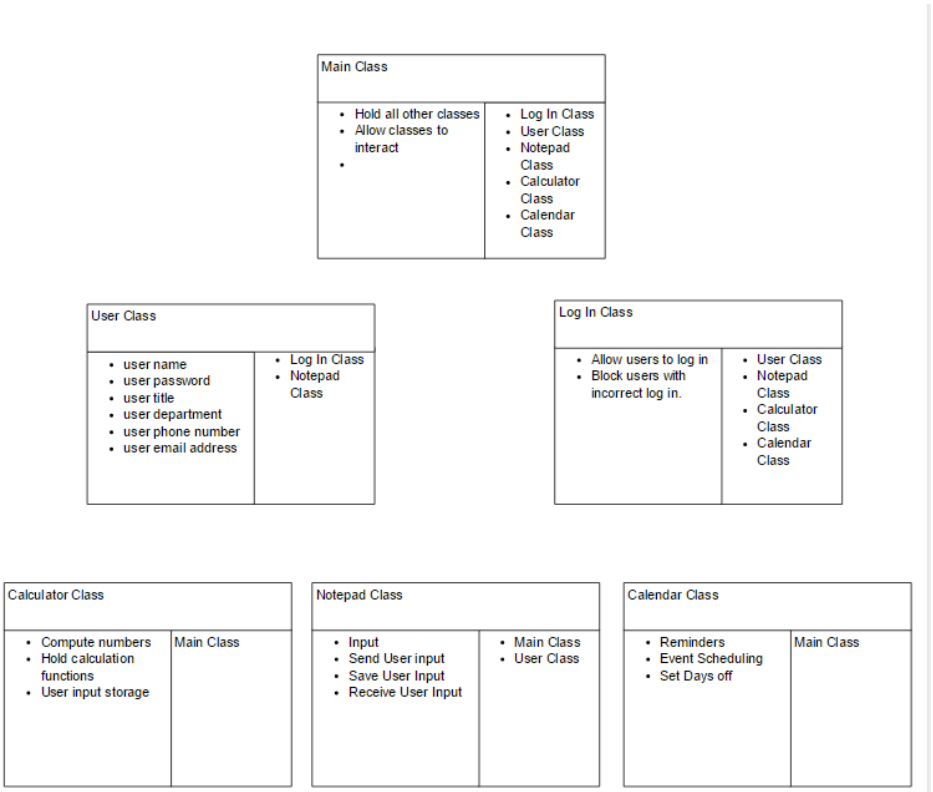


Figure 3.3

ACTIVITY DIAGRAM

This section of the report holds the Activity Diagram of the MOUA application. The diagram displays the paths a user can while utilizing the app. As you can see in the user has many choices in options once going through the initial log in activity.

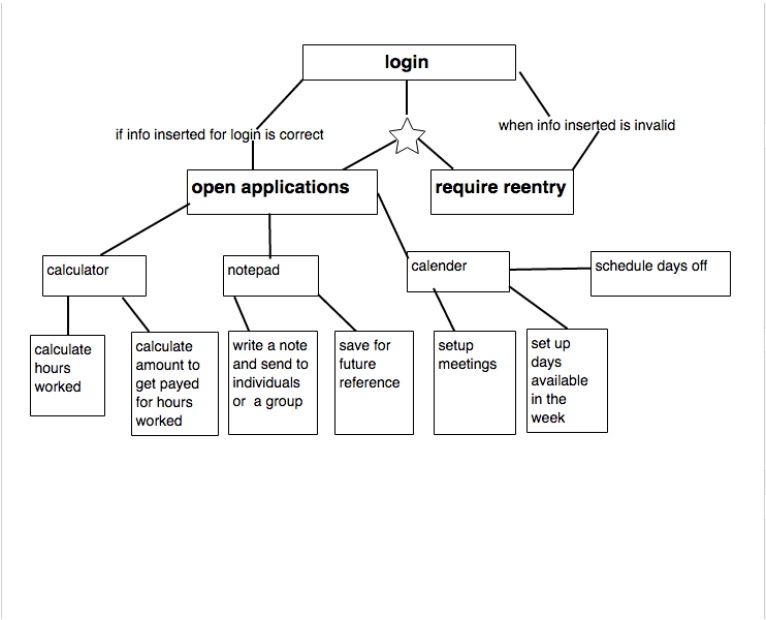
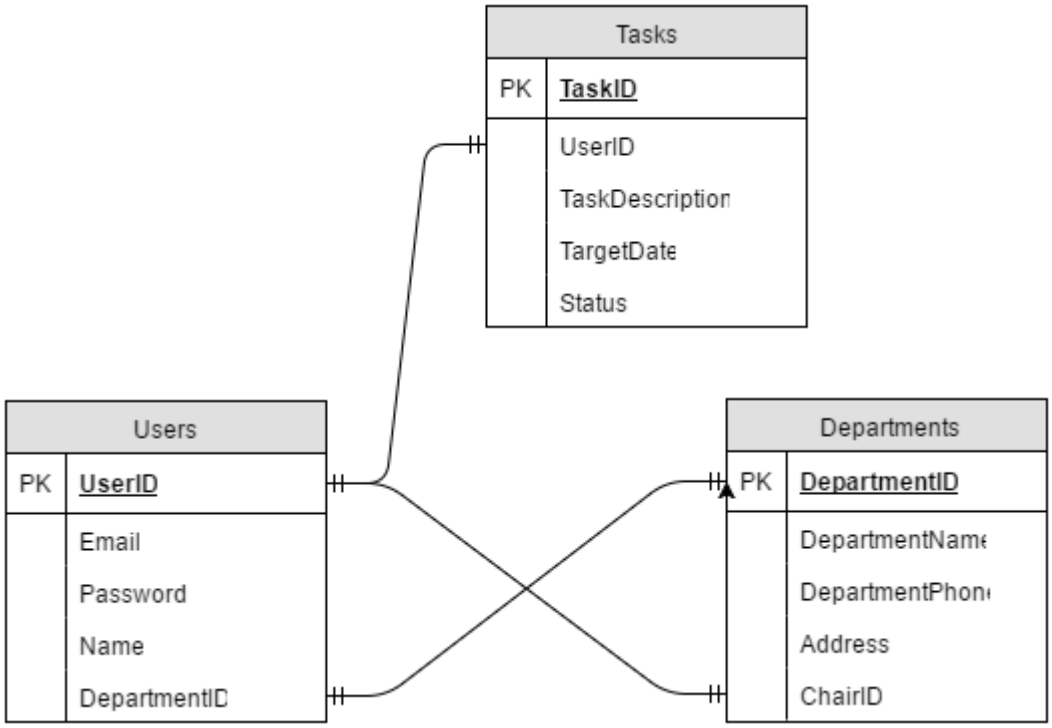


Figure 3.4

RELATIONAL MODEL/DATABASE DIAGRAM



The above diagram is an Entity-Relationship Diagram that models out the attributes and tables needed for the task management feature of the application. In order to implement this feature, the ability to create database files must be imported via SQL Helper in Android Studio. The database will be coded in SQL lite and stored in a source database .db file and packaged internally into the application once it is compiled.

CODE DOCUMENTATION

MAIN SCREEN - MAINACTIVITY.JAVA/CONTENT_MAIN.XML/ACTIVITY_MAIN.XML

By Muhammad Abdul-Qayyum

BACKEND - MAINACTIVITY.JAVA

The backend of this activity is located in the public class MainActivity. The class has 4 button functions that allow for entry into the other features of the application; startNotepad, startDrawing, startCalculator and startScheduling.

```
void startNotepad (View view)
{
    Intent intent = new Intent(this, NotepadActivity.class);
    startActivity(intent);
}

void startDrawing (View view)
{
    Intent intent = new Intent(this, DrawingActivity.class);
    startActivity(intent);
}

void startCalculator (View view)
{
    Intent intent = new Intent(this, CalculatorActivity.class);
    startActivity(intent);
}

void startScheduling (View view)
{
    Intent intent = new Intent(this, SchedulingActivity.class);
    startActivity(intent);
}
```

The ability to move from screen to screen in Android is handled by "Intent" objects. Intent is a predefined class within the Android Studio SDK which allows can be imported via `import android.content.Intent;`. To use intents, you declare an Intent object via `Intent intent = new Intent(this, SchedulingActivity.class);` The function parameters accepted define the start and end acvitiees/classes of the intent. `this` Refers to the current class/activity/screen the application is on. The second parameter defines the activity/class/screen the Intent will take the application/user to. In the above line of code, the Intent refers to the "SchedulingActivity" which is the class that holds the Calender and Scheduling features of MOUA, the office utility app.

FRONTEND - CONTENT_MAIN.XML, ACTIVITY_MAIN.XML

The front end design of the MainActivity, and each activity is handled by corresponding content and activity xml layout files. So for the Main Activity that would be content_main.xml and activity_main.xml. These files are linked to the MainActivity.Java file via the command **`tools:context="com.example.hafiz.officeapp.MainActivity".`** This is important because it is how the backend is linked to the frontend. After defining the correct class, content_main.xml file now has access to the functions you define in MainActivity.Java.

Each UI Element is defined via tags. Therefore, Buttons are defined the <Button … /> tag and EditTexts are defined via the <EditText … /> tag. The elements for buttons and the Analog clock are defined in the content_main layer. The activity_main layer is mostly empty, but it is what would allow for use of the Floating Action Button and the drop down options menu in the tool bar above.

The Main Screen has 4 buttons all defined in the content main; Calculator, Calender, Chat Login, and Notepad. In the tags of these buttons, the functions they call in the above MainActivity.Java class are defined in the xml via the **android:onClick** method.

The background of the application is also handled in the xml via the following method.

```
android:background="@mipmap/wooden_background">
```

content_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   app:layout_behavior="@string/appbar_scrolling_view_behavior"
   tools:context="com.example.hafiz.officeapp.MainActivity"
   tools:showIn="@layout/activity_main"
   android:background="@mipmap/wooden_background">

   <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Calculator"
      android:id="@+id/button"
```

```
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:layout_marginBottom="129dp"
    android:onClick="startCalculator"/>

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Calender"
    android:id="@+id/button2"
    android:layout_alignTop="@+id/button"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:onClick="startScheduling"/>

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Notepad"
    android:id="@+id/button3"
    android:layout_centerVertical="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true"
    android:onClick="startNotepad"/>

  <Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Chat Login"
    android:id="@+id/button4"
    android:layout_alignTop="@+id/button3"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true"
    android:onClick="startLogin"/>

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Office Application"
    android:id="@+id/textView"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="25dp" />

  <AnalogClock
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/analogClock"
    android:layout_below="@+id/textView"
    android:layout_centerHorizontal="true" />
</RelativeLayout>
```

NOTEPAD - NOTEPADACTIVITY.JAVA/CONTENT_NOTEPAD.XML/ACTIVITY_NOTEPAD.XML

By Kevin Harmon

BACKEND – NOTEPADACTIVITY.JAVA

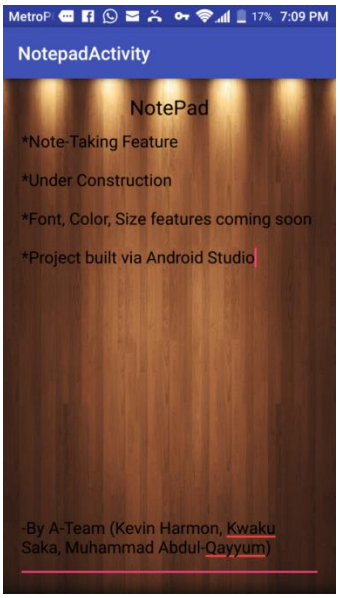
```
package com.example.hafiz.officeapp;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;

public class NotepadActivity extends AppCompatActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_notepad);
      Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
      setSupportActionBar(toolbar);

   }

}
```

Each Screen/Activity of the Application has an option menu. The backend operations of the screen are handled via

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

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}
```

This code is the same on all activities at the moment. The purpose of the menu is to control settings and preferences of the various features (activities) of the application. For example, In the future NotepadActivity will have options that control font size, bold, italics, color etc. The users will be able to control these settings via the options menu.

FRONTEND – CONTENT_NOTEPAD.XML, ACTIVITY_NOTEPAD.XML




```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   app:layout_behavior="@string/appbar_scrolling_view_behavior"
   tools:context="com.example.hafiz.officeapp.NotepadActivity"
   tools:showIn="@layout/activity_notepad"
   android:background="@mipmap/wooden_background">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textAppearance="?android:attr/textAppearanceLarge"
      android:text="NotePad"
      android:id="@+id/textView2"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true" />
```

```
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="textMultiLine"
        android:ems="10"
        android:id="@+id/editText"
        android:layout_below="@+id/textView2"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />
</RelativeLayout>
```

The Text Field of the Notepad is defined with an <EditText /> Tags in the content_notepad.xml file.

CALENDER – SCHEDULINGACTIVITY.JAVA/CONTENT_SCHEDULING.XML

By Muhammad Abdul-Qayyum

BACKEND – SCHEDULINGACTIVITY.JAVA

```
package com.example.hafiz.officeapp;

import android.os.Bundle;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;

public class SchedulingActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_scheduling);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
    }

}
```

FRONTEND – ACTIVITY_SCHEDULING.XML, CONTENT_SCEDULING.XML

activity_scheduling.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="com.example.hafiz.officeapp.SchedulingActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay" />

    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/content_scheduling" />


</android.support.design.widget.CoordinatorLayout>
```

content_scheduling.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.hafiz.officeapp.SchedulingActivity"
    tools:showIn="@layout/activity_scheduling"
    android:background="@mipmap/wooden_background">

    <CalendarView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/calendarView"
        android:layout_marginTop="92dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />

    <DatePicker
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/datePicker"
        android:layout_below="@+id/calendarView"
        android:layout_centerHorizontal="true"
```

```
    android:layout_marginTop="53dp" />
</RelativeLayout>
```

The Calender portion of this application is defined via CalenderView and DatePicker in the content_main.xml file.

CALCULATOR - CALCULATORACTIVITY.JAVA/CONTENT_CALCULATOR.XML/ACTIVITY_CALCULATOR.XML

By Kwaku Saka

BACKEND – CALCULATORACTIVITY.JAVA

CalculatorActivity Class

Functions:

Has an "equal" function that calculates numbers user input, and will do correct operation based on which button has been pressed using if-else statements

buttonAdd

```
if (mAddition == true){

   edt1.setText(mValueOne + mValueTwo +"");
   mAddition=false;
}
```

buttonSub

```
if (mSubtract == true){
   edt1.setText(mValueOne - mValueTwo+"");
   mSubtract=false;
}
```

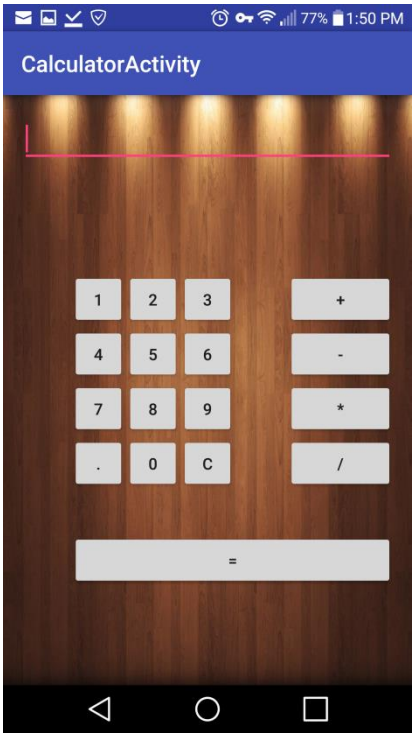
buttonMul

```
if (mMultiplication == true){
   edt1.setText(mValueOne * mValueTwo+"");
   mMultiplication=false;
}
```

buttonDiv

```
if (mDivision == true){
   edt1.setText(mValueOne / mValueTwo+"");
   mDivision=false;
}
```

FRONTEND – CONTENT_CALCULATOR.XML/ACTIVITY_CALCULATOR.XML



content_calculator.xml

The Button Elements of the Calculator are defined in the Layout xml file content_calculator as <Button />.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.example.hafiz.officeapp.CalculatorActivity"
    tools:showIn="@layout/activity_calculator"
    android:background="@mipmap/wooden_background">


    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/edt1" />


    <Button
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="1"
        android:id="@+id/button1"
        android:layout_marginTop="94dp"
        android:layout_below="@+id/edt1"
        android:layout_toStartOf="@+id/button4"
```

```
    android:layout_alignRight="@+id/button4"
    android:layout_alignEnd="@+id/button4" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="2"
    android:id="@+id/button2"
    android:layout_alignTop="@+id/button1"
    android:layout_toLeftOf="@+id/button3"
    android:layout_toStartOf="@+id/button3" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="3"
    android:id="@+id/button3"
    android:layout_alignTop="@+id/button2"
    android:layout_centerHorizontal="true" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="4"
    android:id="@+id/button4"
    android:layout_below="@+id/button1"
    android:layout_toLeftOf="@+id/button2" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="5"
    android:id="@+id/button5"
    android:layout_alignBottom="@+id/button4"
    android:layout_alignLeft="@+id/button2"
    android:layout_alignStart="@+id/button2" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="6"
    android:id="@+id/button6"
    android:layout_below="@+id/button3"
    android:layout_alignLeft="@+id/button3"
    android:layout_alignStart="@+id/button3" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="7"
    android:id="@+id/button7"
    android:layout_below="@+id/button4"
    android:layout_toLeftOf="@+id/button2" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="8"
    android:id="@+id/button8"
    android:layout_below="@+id/button5"
    android:layout_alignLeft="@+id/button5"
    android:layout_alignStart="@+id/button5" />

<Button
```

```
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="9"
    android:id="@+id/button9"
    android:layout_below="@+id/button6"
    android:layout_alignLeft="@+id/button6"
    android:layout_alignStart="@+id/button6" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="+"
    android:id="@+id/buttonadd"
    android:layout_alignTop="@+id/button3"
    android:layout_toRightOf="@+id/button3"
    android:layout_marginLeft="46dp"
    android:layout_marginStart="46dp"
    android:layout_alignRight="@+id/edt1"
    android:layout_alignEnd="@+id/edt1" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="-"
    android:id="@+id/buttonsub"
    android:layout_below="@+id/buttonadd"
    android:layout_alignLeft="@+id/buttonadd"
    android:layout_alignStart="@+id/buttonadd"
    android:layout_alignRight="@+id/buttonadd"
    android:layout_alignEnd="@+id/buttonadd" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="*"
    android:id="@+id/buttonmul"
    android:layout_below="@+id/buttonsub"
    android:layout_alignLeft="@+id/buttonsub"
    android:layout_alignStart="@+id/buttonsub"
    android:layout_alignParentRight="true"
    android:layout_alignParentEnd="true" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="."
    android:id="@+id/button10"
    android:layout_below="@+id/button7"
    android:layout_toLeftOf="@+id/button2" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="0"
    android:id="@+id/button0"
    android:layout_below="@+id/button8"
    android:layout_alignLeft="@+id/button8"
    android:layout_alignStart="@+id/button8" />

<Button
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="C"
    android:id="@+id/buttonC"
```

```
        android:layout_below="@+id/button9"
        android:layout_alignLeft="@+id/button9"
        android:layout_alignStart="@+id/button9" />

    <Button
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="/"
        android:id="@+id/buttondiv"
        android:layout_below="@+id/buttonmul"
        android:layout_alignLeft="@+id/buttonmul"
        android:layout_alignStart="@+id/buttonmul"
        android:layout_alignRight="@+id/buttonmul"
        android:layout_alignEnd="@+id/buttonmul" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="="
        android:id="@+id/buttoneql"
        android:layout_below="@+id/button0"
        android:layout_marginTop="37dp"
        android:layout_alignRight="@+id/buttondiv"
        android:layout_alignEnd="@+id/buttondiv"
        android:layout_alignLeft="@+id/button10"
        android:layout_alignStart="@+id/button10" />
</RelativeLayout>
```

CHAT - LOGINACTIVITY.JAVA/ACTIVITY_LOGIN.XML

By Muhammad Abdul-Qayyum, Kevin Harmon, Kwaku Saka

Android Studio features a Default Code for Creating Accounts and signing. The code however is not fully implemented by default and developers must work to finish and implement it. In order to implement our chat feature, all 3 members worked to study the code and finish the login in portion of our Chat Feature. This portion of the project is still under construction.

BACKEND – LOGINACTIVITY.JAVA

```
package com.example.hafiz.officeapp;

import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.annotation.TargetApi;
import android.content.pm.PackageManager;
import android.support.annotation.NonNull;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.app.LoaderManager.LoaderCallbacks;

import android.content.CursorLoader;
import android.content.Loader;
import android.database.Cursor;
import android.net.Uri;
import android.os.AsyncTask;

import android.os.Build;
import android.os.Bundle;
```

```
import android.provider.ContactsContract;
import android.text.TextUtils;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.inputmethod.EditorInfo;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;

import static android.Manifest.permission.READ_CONTACTS;

/**
 * A login screen that offers login via email/password.
 */
public class LoginActivity extends AppCompatActivity implements LoaderCallbacks<Cursor> {

    /**
     * Id to identity READ_CONTACTS permission request.
     */
    private static final int REQUEST_READ_CONTACTS = 0;

    /**
     * A dummy authentication store containing known user names and passwords.
     * TODO: remove after connecting to a real authentication system.
     */
    private static final String[] DUMMY_CREDENTIALS = new String[]{
            "foo@example.com:hello", "bar@example.com:world"
    };
    /**
     * Keep track of the login task to ensure we can cancel it if requested.
     */
    private UserLoginTask mAuthTask = null;

    // UI references.
    private AutoCompleteTextView mEmailView;
    private EditText mPasswordView;
    private View mProgressView;
    private View mLoginFormView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        // Set up the login form.
        mEmailView = (AutoCompleteTextView) findViewById(R.id.email);
        populateAutoComplete();

        mPasswordView = (EditText) findViewById(R.id.password);
        mPasswordView.setOnEditorActionListener(new TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView textView, int id, KeyEvent keyEvent) {
                if (id == R.id.login || id == EditorInfo.IME_NULL) {
                    attemptLogin();
                    return true;
                }
                return false;
            }
        });

        Button mEmailSignInButton = (Button) findViewById(R.id.email_sign_in_button);
```

```
    mEmailSignInButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
            attemptLogin();
        }
    });

    mLoginFormView = findViewById(R.id.login_form);
    mProgressView = findViewById(R.id.login_progress);
}

private void populateAutoComplete() {
    if (!mayRequestContacts()) {
        return;
    }

    getLoaderManager().initLoader(0, null, this);
}

private boolean mayRequestContacts() {
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
        return true;
    }
    if (checkSelfPermission(READ_CONTACTS) == PackageManager.PERMISSION_GRANTED) {
        return true;
    }
    if (shouldShowRequestPermissionRationale(READ_CONTACTS)) {
        Snackbar.make(mEmailView, R.string.permission_rationale, Snackbar.LENGTH_INDEFINITE)
                .setAction(android.R.string.ok, new View.OnClickListener() {
                    @Override
                    @TargetApi(Build.VERSION_CODES.M)
                    public void onClick(View v) {
                        requestPermissions(new String[]{READ_CONTACTS}, REQUEST_READ_CONTACTS);
                    }
                });
    } else {
        requestPermissions(new String[]{READ_CONTACTS}, REQUEST_READ_CONTACTS);
    }
    return false;
}

/**
 * Callback received when a permissions request has been completed.
 */
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    if (requestCode == REQUEST_READ_CONTACTS) {
        if (grantResults.length == 1 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            populateAutoComplete();
        }
    }
}


/**
 * Attempts to sign in or register the account specified by the login form.
 * If there are form errors (invalid email, missing fields, etc.), the
 * errors are presented and no actual login attempt is made.
 */
private void attemptLogin() {
    if (mAuthTask != null) {
        return;
    }

    // Reset errors.
```

```
    mEmailView.setError(null);
    mPasswordView.setError(null);

    // Store values at the time of the login attempt.
    String email = mEmailView.getText().toString();
    String password = mPasswordView.getText().toString();

    boolean cancel = false;
    View focusView = null;

    // Check for a valid password, if the user entered one.
    if (!TextUtils.isEmpty(password) && !isPasswordValid(password)) {
        mPasswordView.setError(getString(R.string.error_invalid_password));
        focusView = mPasswordView;
        cancel = true;
    }

    // Check for a valid email address.
    if (TextUtils.isEmpty(email)) {
        mEmailView.setError(getString(R.string.error_field_required));
        focusView = mEmailView;
        cancel = true;
    } else if (!isEmailValid(email)) {
        mEmailView.setError(getString(R.string.error_invalid_email));
        focusView = mEmailView;
        cancel = true;
    }

    if (cancel) {
        // There was an error; don't attempt login and focus the first
        // form field with an error.
        focusView.requestFocus();
    } else {
        // Show a progress spinner, and kick off a background task to
        // perform the user login attempt.
        showProgress(true);
        mAuthTask = new UserLoginTask(email, password);
        mAuthTask.execute((Void) null);
    }
}

private boolean isEmailValid(String email) {
    //TODO: Replace this with your own logic
    return email.contains("@");
}

private boolean isPasswordValid(String password) {
    //TODO: Replace this with your own logic
    return password.length() > 4;
}

/**
 * Shows the progress UI and hides the login form.
 */
@TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
private void showProgress(final boolean show) {
    // On Honeycomb MR2 we have the ViewPropertyAnimator APIs, which allow
    // for very easy animations. If available, use these APIs to fade-in
    // the progress spinner.
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB_MR2) {
        int shortAnimTime = getResources().getInteger(android.R.integer.config_shortAnimTime);

        mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
        mLoginFormView.animate().setDuration(shortAnimTime).alpha(
                show ? 0 : 1).setListener(new AnimatorListenerAdapter() {
            @Override
```

```
            public void onAnimationEnd(Animator animation) {
                mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
            }
        });

        mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
        mProgressView.animate().setDuration(shortAnimTime).alpha(
                show ? 1 : 0).setListener(new AnimatorListenerAdapter() {
            @Override
            public void onAnimationEnd(Animator animation) {
                mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
            }
        });
    } else {
        // The ViewPropertyAnimator APIs are not available, so simply show
        // and hide the relevant UI components.
        mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);
        mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
    }
}

@Override
public Loader<Cursor> onCreateLoader(int i, Bundle bundle) {
    return new CursorLoader(this,
            // Retrieve data rows for the device user's 'profile' contact.
            Uri.withAppendedPath(ContactsContract.Profile.CONTENT_URI,
                    ContactsContract.Contacts.Data.CONTENT_DIRECTORY), ProfileQuery.PROJECTION,

            // Select only email addresses.
            ContactsContract.Contacts.Data.MIMETYPE +
                    " = ?", new String[]{ContactsContract.CommonDataKinds.Email
            .CONTENT_ITEM_TYPE},

            // Show primary email addresses first. Note that there won't be
            // a primary email address if the user hasn't specified one.
            ContactsContract.Contacts.Data.IS_PRIMARY + " DESC");
}

@Override
public void onLoadFinished(Loader<Cursor> cursorLoader, Cursor cursor) {
    List<String> emails = new ArrayList<>();
    cursor.moveToFirst();
    while (!cursor.isAfterLast()) {
        emails.add(cursor.getString(ProfileQuery.ADDRESS));
        cursor.moveToNext();
    }

    addEmailsToAutoComplete(emails);
}

@Override
public void onLoaderReset(Loader<Cursor> cursorLoader) {

}

private void addEmailsToAutoComplete(List<String> emailAddressCollection) {
    //Create adapter to tell the AutoCompleteTextView what to show in its dropdown list.
    ArrayAdapter<String> adapter =
            new ArrayAdapter<>(LoginActivity.this,
                    android.R.layout.simple_dropdown_item_1line, emailAddressCollection);

    mEmailView.setAdapter(adapter);
}


private interface ProfileQuery {
```

```
        String[] PROJECTION = {
                ContactsContract.CommonDataKinds.Email.ADDRESS,
                ContactsContract.CommonDataKinds.Email.IS_PRIMARY,
        };

        int ADDRESS = 0;
        int IS_PRIMARY = 1;
    }

    /**
     * Represents an asynchronous login/registration task used to authenticate
     * the user.
     */
    public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {

        private final String mEmail;
        private final String mPassword;

        UserLoginTask(String email, String password) {
            mEmail = email;
            mPassword = password;
        }

        @Override
        protected Boolean doInBackground(Void... params) {
            // TODO: attempt authentication against a network service.

            try {
                // Simulate network access.
                Thread.sleep(2000);
            } catch (InterruptedException e) {
                return false;
            }

            for (String credential : DUMMY_CREDENTIALS) {
                String[] pieces = credential.split(":");
                if (pieces[0].equals(mEmail)) {
                    // Account exists, return true if the password matches.
                    return pieces[1].equals(mPassword);
                }
            }

            // TODO: register the new account here.
            return true;
        }

        @Override
        protected void onPostExecute(final Boolean success) {
            mAuthTask = null;
            showProgress(false);

            if (success) {
                finish();
            } else {
                mPasswordView.setError(getString(R.string.error_incorrect_password));
                mPasswordView.requestFocus();
            }
        }

        @Override
        protected void onCancelled() {
            mAuthTask = null;
            showProgress(false);
        }
    }
}
```

FRONTEND – ACTIVITY_LOGIN.XML

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center_horizontal"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.hafiz.officeapp.LoginActivity"
    android:background="@mipmap/wooden_background">

    <!-- Login progress -->
    <ProgressBar
        android:id="@+id/login_progress"
        style="?android:attr/progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:visibility="gone" />

    <ScrollView
        android:id="@+id/login_form"
        android:layout_width="716dp"
        android:layout_height="613dp">

        <LinearLayout
            android:id="@+id/email_login_form"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <AutoCompleteTextView
                    android:id="@+id/email"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="@string/prompt_email"
                    android:inputType="textEmailAddress"
                    android:maxLines="1"
                    android:singleLine="true" />

            </android.support.design.widget.TextInputLayout>

            <android.support.design.widget.TextInputLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content">

                <EditText
                    android:id="@+id/password"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:hint="@string/prompt_password"
                    android:imeActionId="@+id/login"
                    android:imeActionLabel="@string/action_sign_in_short"
                    android:imeOptions="actionUnspecified"
```

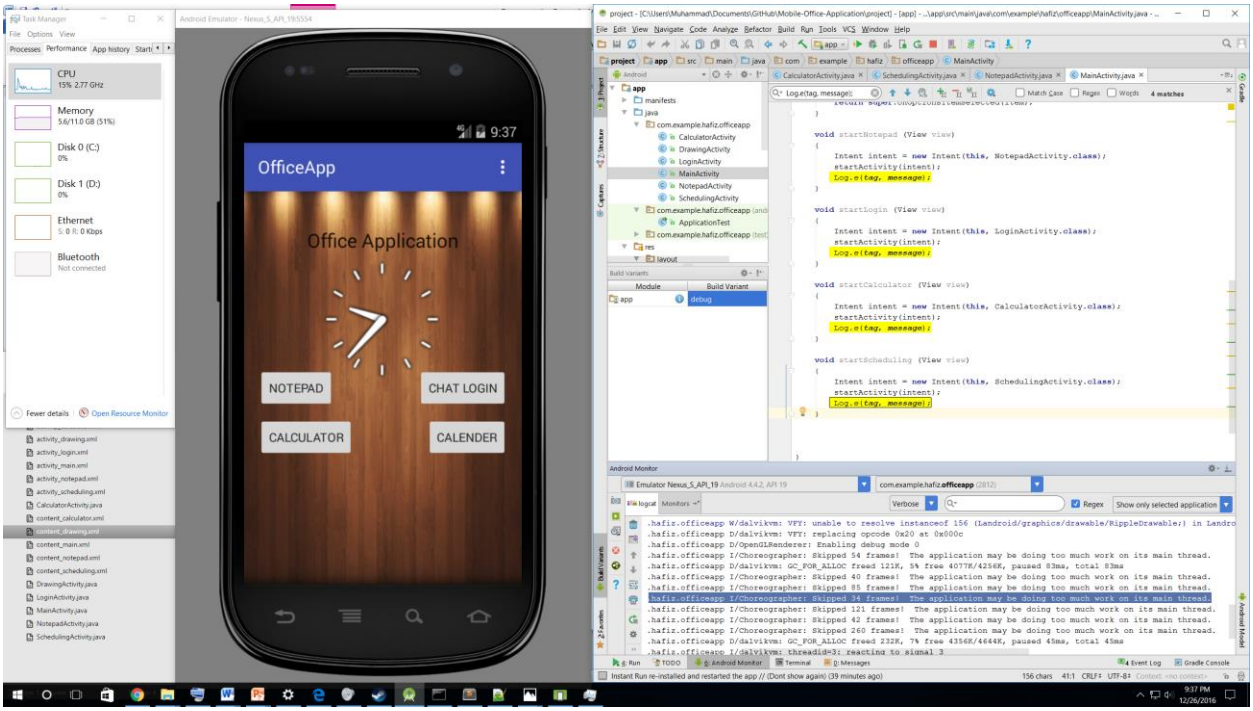
```
            android:inputType="textPassword"
            android:maxLines="1"
            android:singleLine="true" />

        </android.support.design.widget.TextInputLayout>

        <Button
          android:id="@+id/email_sign_in_button"
          style="?android:textAppearanceSmall"
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:layout_marginTop="16dp"
          android:text="@string/action_sign_in"
          android:textStyle="bold" />

      </LinearLayout>
    </ScrollView>
</LinearLayout>
```

PRODUCT TESTING

Testing for this project is done via Android Studio by creating an Android Virtual Device (AVD). This is an emulator that allows us to run Android on our machine alongside the Android Studio IDE. The Virtual Device allows us to test the application live while developing. To test the functionality of buttons to make sure they don't crash when used, we clicked each button on each screen multiple times to ensure consistency. If the application does crash, the console in the figure below reveals which function of the code is the cause of the crash. We also used the Log method below to log errors and info in the event of a crash, or other unintended results.

```
private static final String tag = "MyActivity";
private static final String message = "";
Log.e(tag, message);
```

The figure below is an example of how log is used during development.



The way log works is that you call the log function at a certain part of your code, and it returns a message stating the cause of your error. When used in conjunction with the "logcat" console; located in the bottom right of the image above; we were able to pinpoint and resolve any errors that came up in the project.

CONCLUSION

In conclusion our project is an application that has a variety of uses for both home and enterprise users. Our hope is to create an application that makes it easier for everyone to manage tasks and jobs.