Abstract

The mobile device market is rapidly developing. Most of these devices run Android Operating System. The Android application development is simplified by its SDK that provides tools and APIs needed to develop applications, favouring an easy integration with many resources available on the device. However, due to the limited resources available on mobile devices and the limited battery lifetime,the project of mobile apps have hard constraints specially on performance and energy consumption.

Google presents best practices for android development focusing on performance improvement. Android best practices for performance are small code changes proposed by Google to reduce execution time. This work evaluates and analyzes the impact of two of these best practices on performance and energy consumption. The practices are initially applied to experimental code and then on real world Android application and their impact on performance and power consumption are analyzed. The results obtained indicate that these practices have a positive impact on performance and energy efficiency.

By- Rhea Dsouza

Table of Contents

List of Figures

Chapter 1

Introduction

This chapter gives an overview of the dissertation work entitled “MedicoApp”. The problem statement, motivation for the project and the scope of the project are also explained in this chapter.

The remarkable problem is that patients forget to take the proper medicines in the proper proportion and at the proper time. Medication adherence, which refers to the degree or extent to which a patient takes the right medication at the right time according to a doctor’s prescription, has recently emerged as a serious issue because many studies have reported that non-adherence may critically affect the patient, thereby raising medical costs. The category of patients involve teachers, students, businessmen, housewives, children and also all of us who have busy schedules.

With technological advances, the development of applications for mobile devices has grown exponentially. However, mobile applications are significantly different from traditional applications, mainly due to limited resources available on mobile devices (e.g battery, memory, etc.). Thus, the software should be developed considering these restrictions and optimization should be applied in order to obtain an efficient code.

1.2 Problem Statement

The main task in the proposed work is to provide a platform where people can schedule alarms to take medicine, keep track of the covid cases and also purchase medicines.

1.2 Motivation

Most of the time patients may forget to take the medicines at proper time as per the prescribed duration which may cause late recovery from the disease or illness. Hence, we came up with this app to have a systematic medicine schedule.

1.3 Scope of the Project

This work focuses on providing an app for people to schedule their medicine time and also order medicines as and when they require.

1.4 Report Structure

This report is structured as follows:

- Chapter 1 presents an overview of the project in brief. Section 1.1 provides a brief problem statement.
- Chapter 2 explains the methodology that was used in this work to arrive at the final results.
- Chapter 3 presents the experimental results that were achieved.
- Chapter 4 presents the flow of the work carried out with the help of screen shots.
- Chapter 5 summarizes the initial goals of this work and reviews the results obtained. It also provides suggestions on how this project can be extended through future work.

Chapter 2

Software Requirements Specification

In this chapter the hardware and software requirements for developing the application are discussed. This chapter also provides an overview of the functional and non-functional requirements of the application.

2.1 Hardware Requirements

Any mobile SOC with 512 Mb ram with at least 120Mb free space.

2.2 Software Requirements

Android - Android with min android SDK version 16.

2.3 Functional Requirements

- Set the alarm
- Modify the alarm date and time
- Delete the alarm
- Search functionality
- Fetch and parse data from API
- Search functionality
- Order medicines

2.4 Non-Functional Requirements

Easy to use and single platform for many tasks.

Chapter 3

Design Implementation

This chapter illustrates the application design with the help of relevant diagram. The activity and layout files are shown here, and their working is explained.

3.1 Design of Application

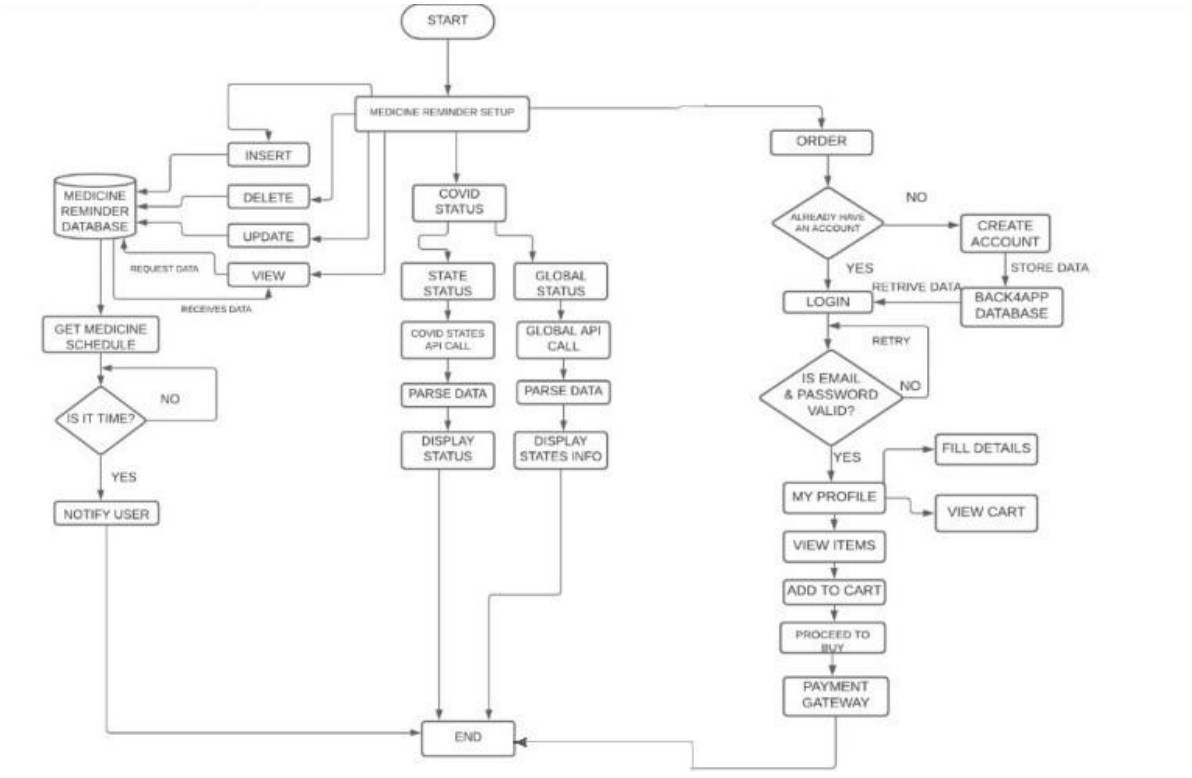


Fig 3.1: Architectural Diagram

3.2 Source

3.2.1 MainActivity.java

```
package com.rhea.medicoapp;
import android.app.AlarmManager;
```

```java
import android.app.DatePickerDialog;
import android.app.PendingIntent;
import android.app.TimePickerDialog;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.DatePicker;
import android.widget.TimePicker;
import android.widget.Toast;
import java.util.Calendar;
import com.rhea.medicoapp.DBHelper;
import com.rhea.medicoapp.AlarmReceiver;
import com.rhea.medicoapp.R;
public class MainActivity extends AppCompatActivity {
    EditText mname, mdate, mtime;
    Button insert, update, delete, view, btncheck;
    DatePickerDialog datePickerDialog;
    TimePickerDialog timePickerDialog;
    AlarmManager alarmManager;
    int uday, umonth, uyear, uhour, umin;
    boolean status;
    DBHelper DB;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getSupportActionBar().setDisplayShowHomeEnabled(true);

getSupportActionBar().setIcon(R.drawable.ic_baseline_medical_services_24);
        setContentView(R.layout.activity_main);
        btncheck = findViewById(R.id.btncheck);
        mname = findViewById(R.id.mname);
        mdate = findViewById(R.id.mdate);
        mtime = findViewById(R.id.mtime);
        insert = findViewById(R.id.btnInsert);
        update = findViewById(R.id.btnUpdate);
        delete = findViewById(R.id.btnDelete);
        view = findViewById(R.id.btnView);
        alarmManager = (AlarmManager) getSystemService(ALARM_SERVICE);
        DB = new DBHelper(this);
        addDate();
        addTime();
```

```
        addData();
        //addAlarm();
        updateData();
        deleteData();
        viewData();
        checkStatus();
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        int id=item.getItemId();

        if(id==R.id.title1){
            Toast.makeText(this,"order",Toast.LENGTH_SHORT).show();
            Intent intent = new Intent(MainActivity.this,
LoginActivity.class);
            startActivity(intent);
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    public void checkStatus()
    {
        btncheck.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new
Intent(MainActivity.this,GlobalStats.class);
                startActivity(intent);
            }
        });
    }

    public void addDate() {
        mdate.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final Calendar c1 = Calendar.getInstance();
                int cyear = c1.get(Calendar.YEAR);
                int cmonth = c1.get(Calendar.MONTH);
                int cday = c1.get(Calendar.DAY_OF_MONTH);
                datePickerDialog = new DatePickerDialog(MainActivity.this,
new DatePickerDialog.OnDateSetListener() {
```

```
                    @Override
                    public void onDateSet(DatePicker view, int year, int
month, int dayOfMonth) {
                        uyear = year;
                        umonth = month;
                        uday = dayOfMonth;
                        mdate.setText(dayOfMonth + "/" + (month + 1) +
"/" + year);

                    }
                }, cyear, cmonth, cday);
                datePickerDialog.show();

            }
        });
    }

    public void addTime() {
        mtime.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final Calendar c2 = Calendar.getInstance();
                int chour = c2.get(Calendar.HOUR_OF_DAY);
                int cmin = c2.get(Calendar.MINUTE);
                timePickerDialog = new TimePickerDialog(MainActivity.this,
new TimePickerDialog.OnTimeSetListener() {
                    @Override
                    public void onTimeSet(TimePicker view, int hourOfDay,
int minute) {
                        mtime.setText(hourOfDay + ":" + minute);
                        uhour = hourOfDay;
                        umin = minute;
                    }
                }, chour, cmin, false);
                timePickerDialog.show();

            }
        });
    }


    public void addData() {
        insert.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String name, date, time;
                name = mname.getText().toString();
                date = mdate.getText().toString();
                time = mtime.getText().toString();
                status = DB.insertuserdata(name, date, time);
                if (status) {
```

```
                    Toast.makeText(MainActivity.this, "Record Added",
Toast.LENGTH_SHORT).show();
                    mname.setText("");
                    mdate.setText("");
                    mtime.setText("");
                    addAlarm();
                } else
                    Toast.makeText(MainActivity.this, "Error!! Record not
added", Toast.LENGTH_SHORT).show();
            }
        });
    }
    public void updateData(){
        update.setOnClickListener(new View.OnClickListener()

        {
            @Override
            public void onClick (View view){
                String nameTXT = mname.getText().toString();
                String dateTXT = mdate.getText().toString();
                String timeTXT = mtime.getText().toString();

                Boolean checkupdatedata = DB.updateuserdata(nameTXT,
dateTXT, timeTXT);
                if (checkupdatedata == true)
                    Toast.makeText(MainActivity.this, "Entry Updated",
Toast.LENGTH_SHORT).show();
                else
                    Toast.makeText(MainActivity.this, "New Entry Not
Updated", Toast.LENGTH_SHORT).show();
            }
        });}

    public void deleteData(){
        delete.setOnClickListener(new View.OnClickListener()

        {
            @Override
            public void onClick (View view){
                String nameTXT = mname.getText().toString();
                Boolean checkudeletedata = DB.deletedata(nameTXT);
                if (checkudeletedata == true)
                    Toast.makeText(MainActivity.this, "Entry Deleted",
Toast.LENGTH_SHORT).show();
                else
                    Toast.makeText(MainActivity.this, "Entry Not Deleted",
Toast.LENGTH_SHORT).show();
            }
        });}
    public void viewData() {
```

```
        view.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Cursor res = DB.getdata();
                if (res.getCount() == 0) {
                    Toast.makeText(MainActivity.this, "No Entry Exists",
Toast.LENGTH_SHORT).show();
                    return;
                }
                StringBuffer buffer = new StringBuffer();
                while (res.moveToNext()) {
                    buffer.append("Name :" + res.getString(0) + "\n");
                    buffer.append("Date :" + res.getString(1) + "\n");
                    buffer.append("Time :" + res.getString(2) + "\n\n");
                }

                AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this);
                builder.setCancelable(true);
                builder.setTitle("User Entries");
                builder.setMessage(buffer.toString());
                builder.show();
            }
        });
    }

    public void addAlarm()
    {
        Toast.makeText(MainActivity.this, "Alarm is set!!",
Toast.LENGTH_SHORT).show();
        Calendar cal=Calendar.getInstance();
        cal.set(uyear,umonth,uday,uhour,umin);
        Intent intent = new Intent(MainActivity.this,AlarmReceiver.class);
        PendingIntent pendingIntent = PendingIntent.getBroadcast(this,
0,intent,0);

alarmManager.set(AlarmManager.RTC_WAKEUP,cal.getTimeInMillis(),pendingInt
ent);

    }


}
```

By- Rhea Dsouza

3.2.2 activity_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:padding="10dp"
    tools:context=".MainActivity"
    android:background="@drawable/gradient_background">
    <TextView
        android:id="@+id/texttitle"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Please enter the details below"
        android:textColor="#2F0835"
        android:textSize="24dp"
        android:layout_marginTop="20dp"
        />
    <EditText
        android:id="@+id/mname"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Medicine Name"
        android:background="#DFD2E6"
        android:textSize="24dp"
        android:fontFamily="monospace"
        android:layout_below="@+id/texttitle"
        android:inputType="textPersonName"/>

    <EditText
        android:id="@+id/mdate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="12px"
        android:hint="Consumption Date"
        android:background="#DFD2E6"
        android:fontFamily="monospace"
        android:textSize="24dp"
        android:layout_below="@+id/mname"
        android:inputType="number"/>
    <EditText
        android:id="@+id/mtime"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Consumption Time"
```

```
        android:layout_marginTop="16px"
        android:background="#DFD2E6"
        android:fontFamily="monospace"
        android:textSize="24dp"
        android:layout_below="@+id/mdate"
        android:inputType="number"/>

    <Button
        android:id="@+id/btnInsert"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="50px"
        android:layout_marginTop="255dp"
        android:background="@drawable/button_sel"
        android:text="Insert"
        android:textSize="24dp" />

    <Button
        android:id="@+id/btnUpdate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="80px"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_marginTop="255dp"
        android:layout_marginRight="10dp"
        android:background="@drawable/button_sel"
        android:text="UPDATE"
        android:textSize="24dp" />

    <Button
        android:id="@+id/btnDelete"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="390dp"
        android:background="@drawable/button_sel"
        android:layout_marginLeft="50px"
        android:text="DELETE"
        android:textSize="24dp" />

    <Button
        android:id="@+id/btnView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="390dp"
        android:layout_marginLeft="80px"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_marginRight="10dp"
        android:background="@drawable/button_sel"
        android:text="VIEW"
```

```
        android:textSize="24dp" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="450dp"
        android:layout_marginLeft="370px"
        android:background="@drawable/button_sel"
        android:text="COVID STATUS"
        android:textSize="20dp"
        android:id="@+id/btncheck"/>

</RelativeLayout>
```

Here, activity_main.xml is the layout file for MainActivity.java. It consists of Edit text boxes and buttons for the user to set the reminder. In addition to this, the user can navigate to the page where the covid status is shown. If the user wants to order medicines, he has to create an account in order to do so.

Chapter 4

Screen Shots

The screen shots in this chapter depict the working of the application

4.1 Running Application

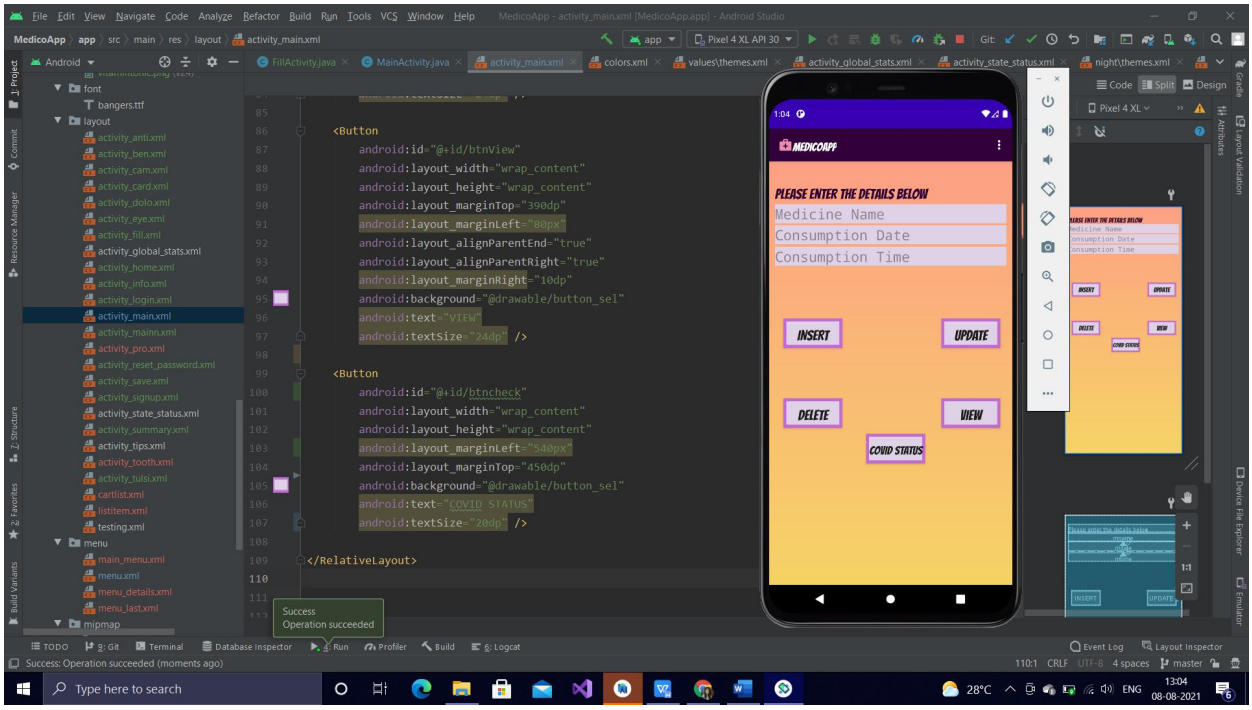


Fig 4.1.1 : Running Application

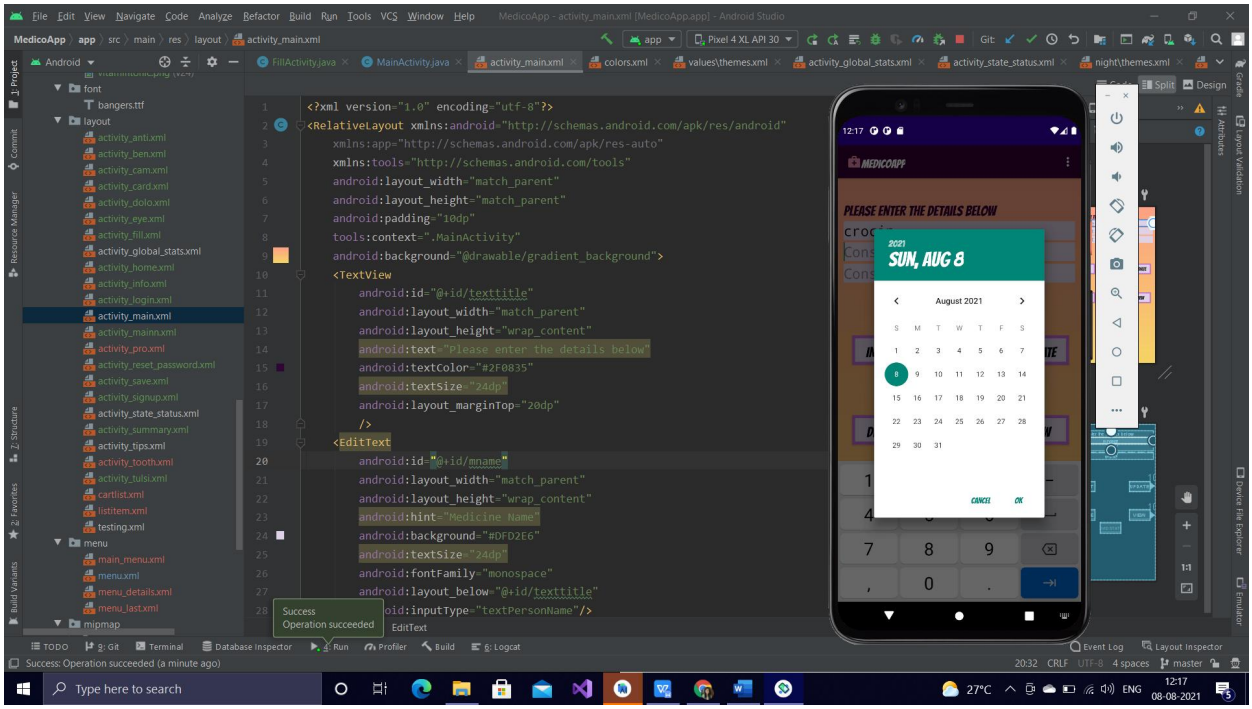


Fig 4.1.2: Set date

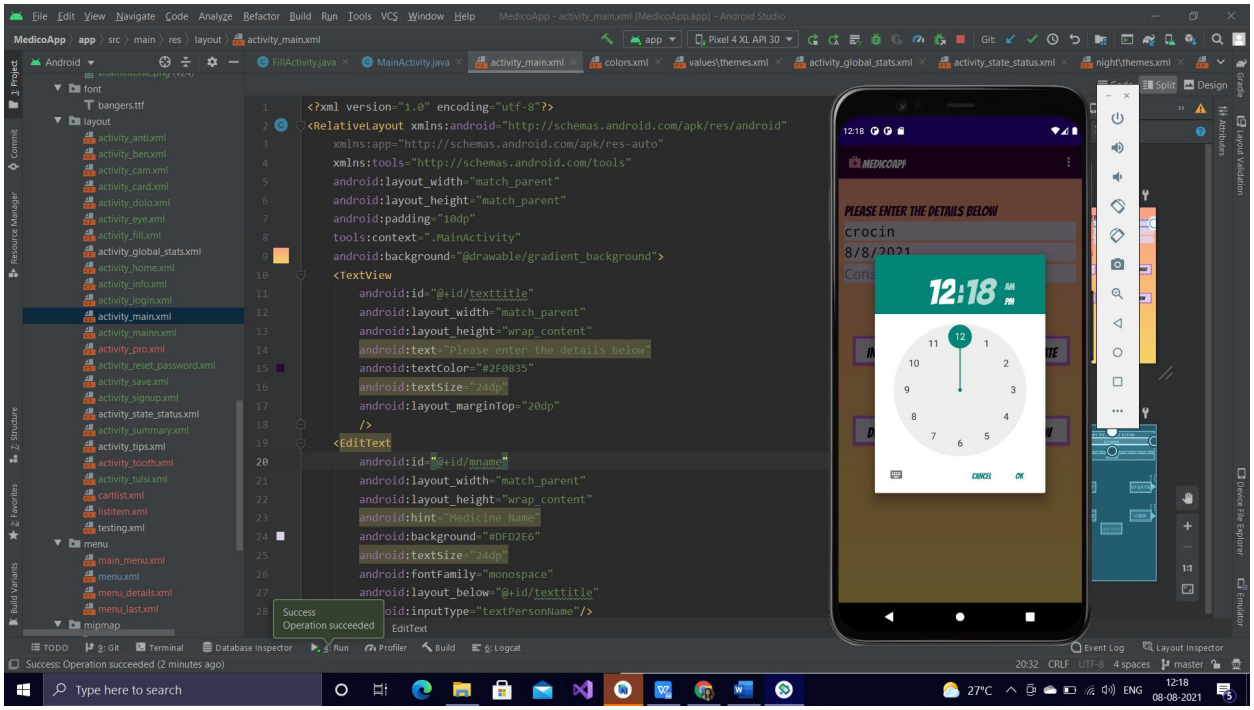


Fig 4.1.3: Set time

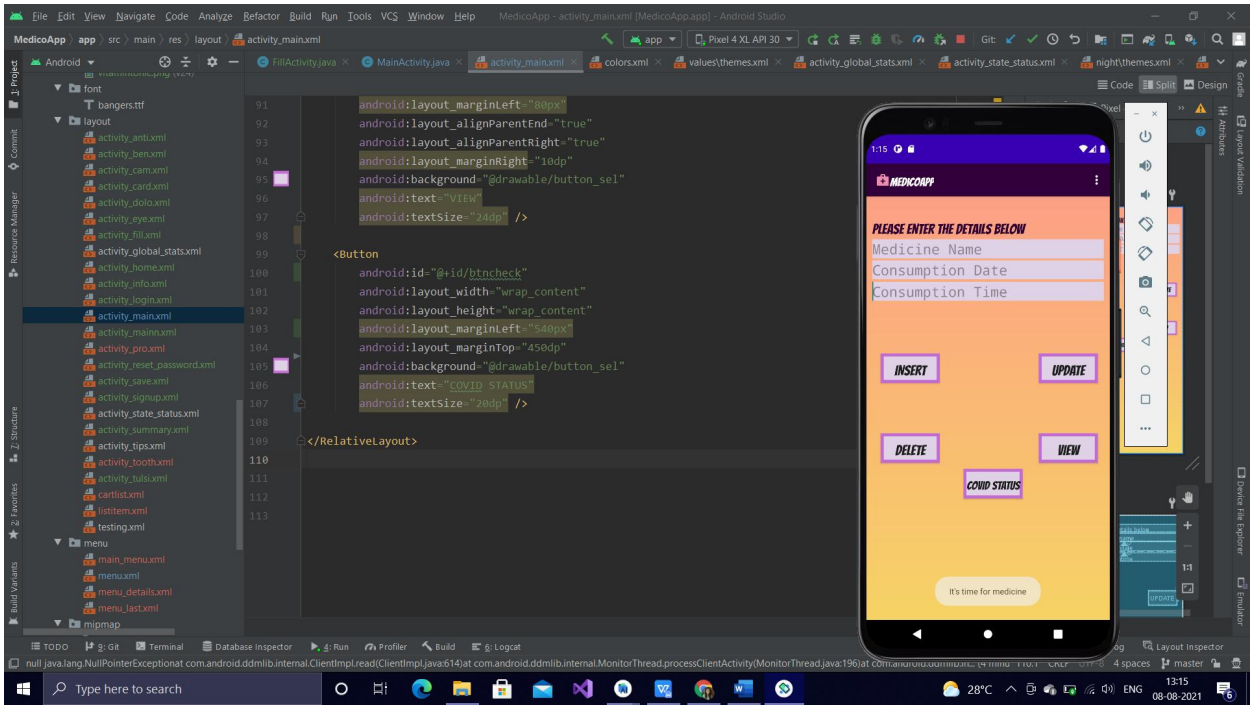


Fig 4.1.4: Alarm Rings

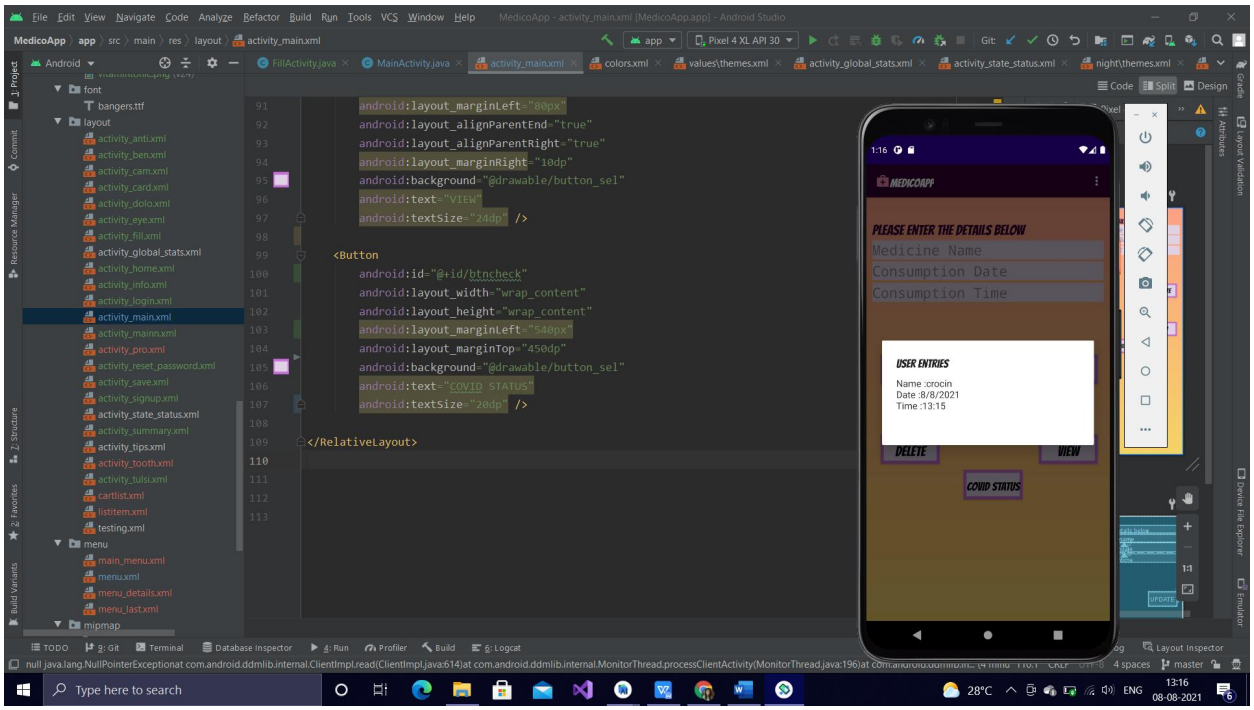


Fig 4.1.5: View Schedule

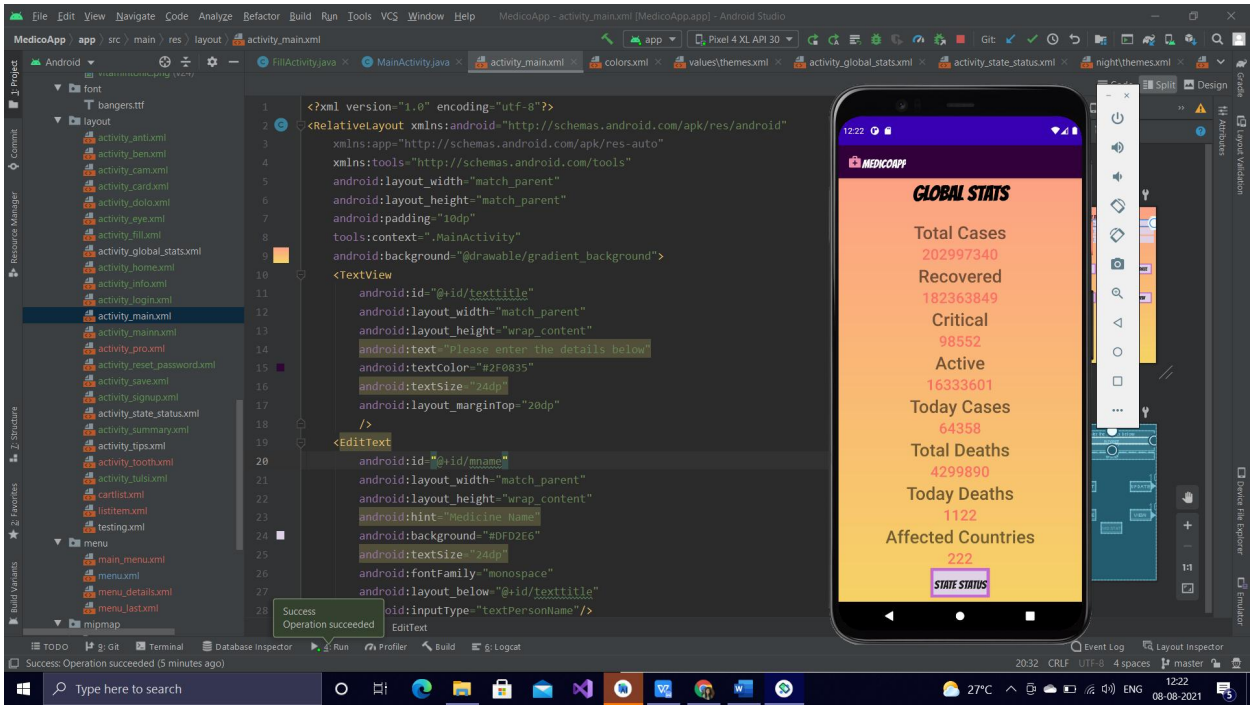


Fig 4.1.6: View global stats

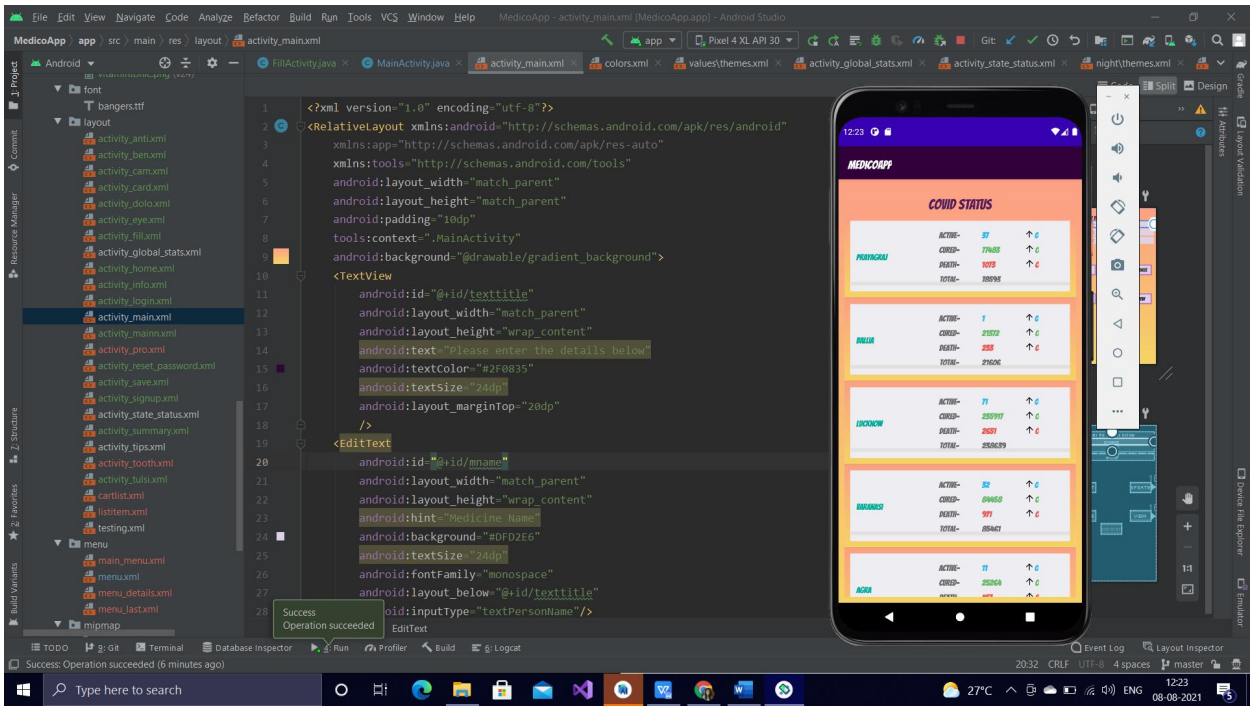


Fig 4.1.7: View State Stats

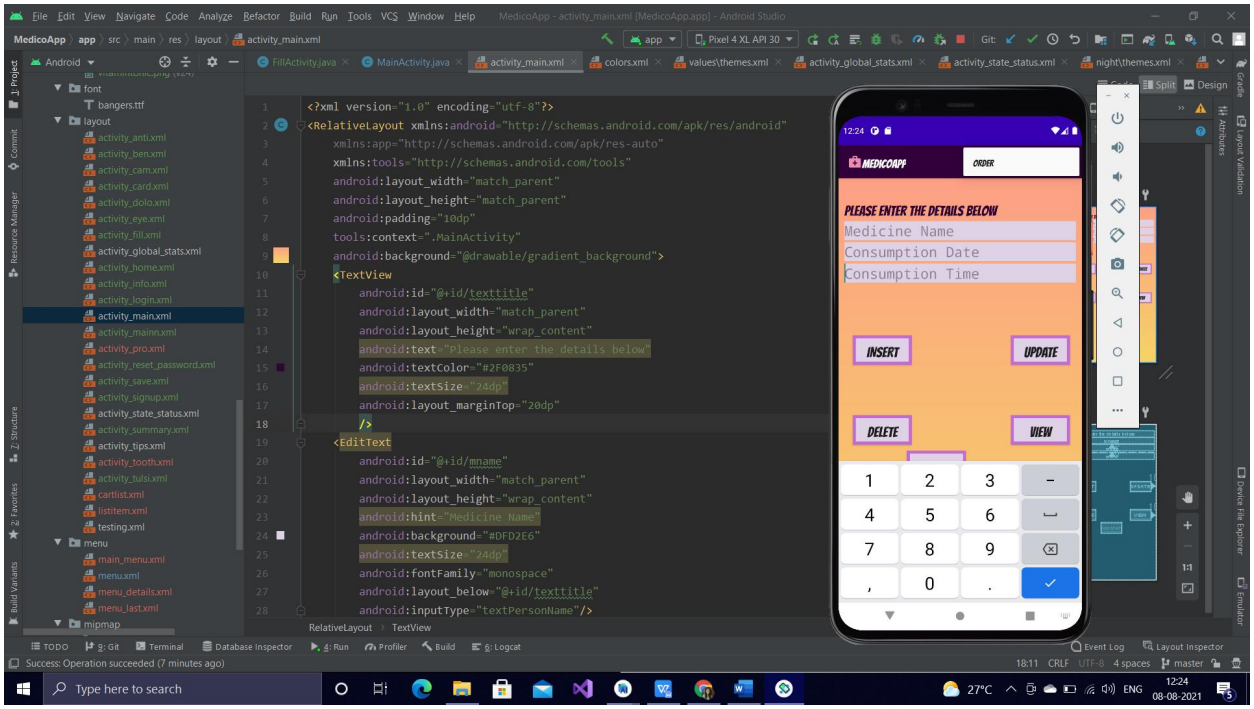


Fig 4.1.8: Order

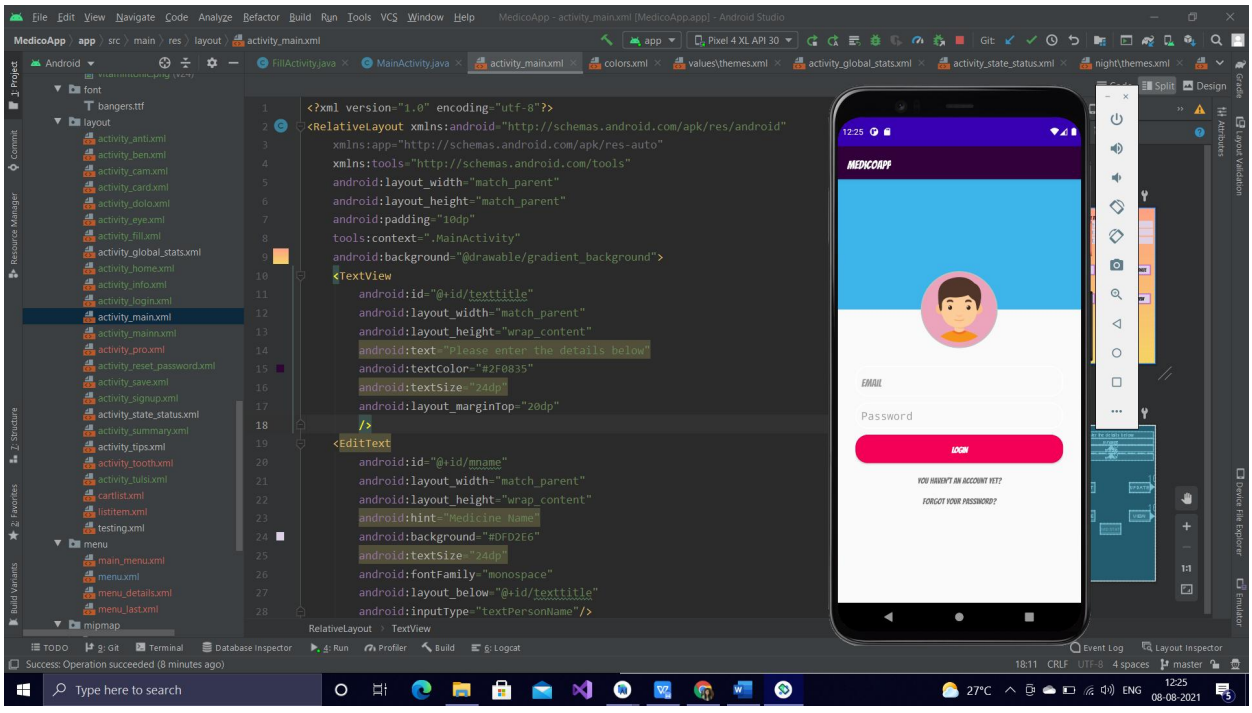


Fig 4.1.9: Login

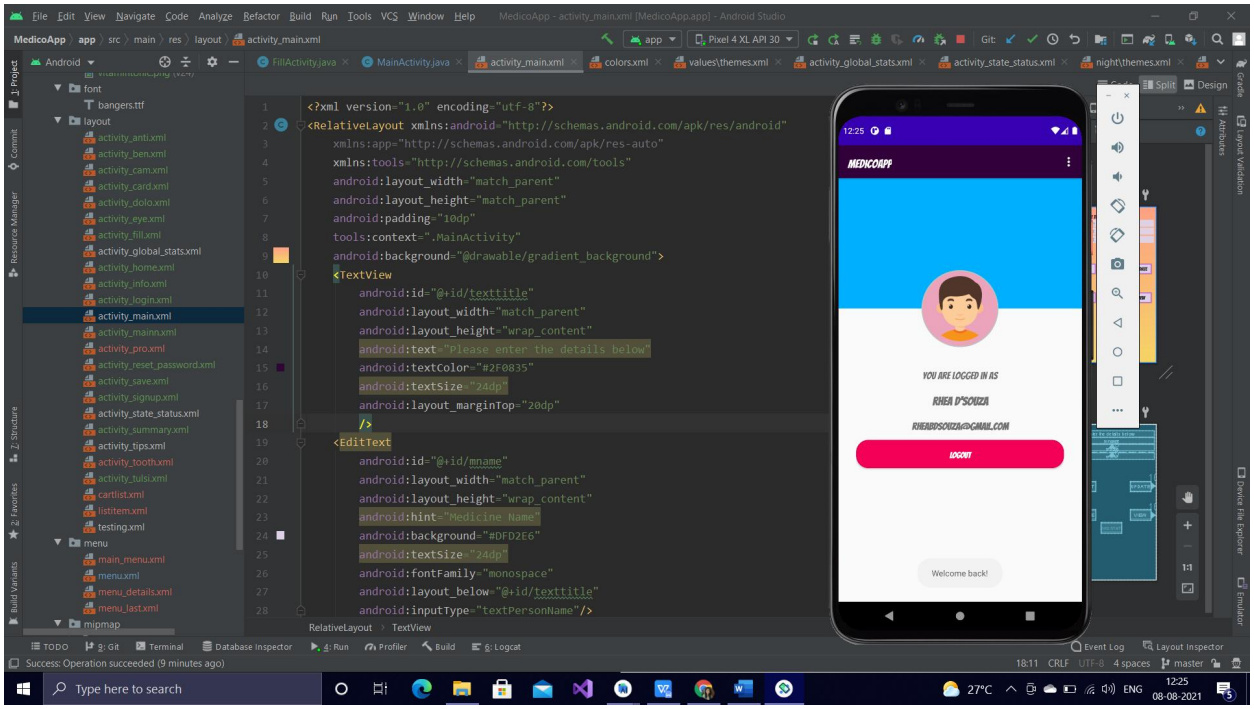


Fig 4.1.10: Login Successful

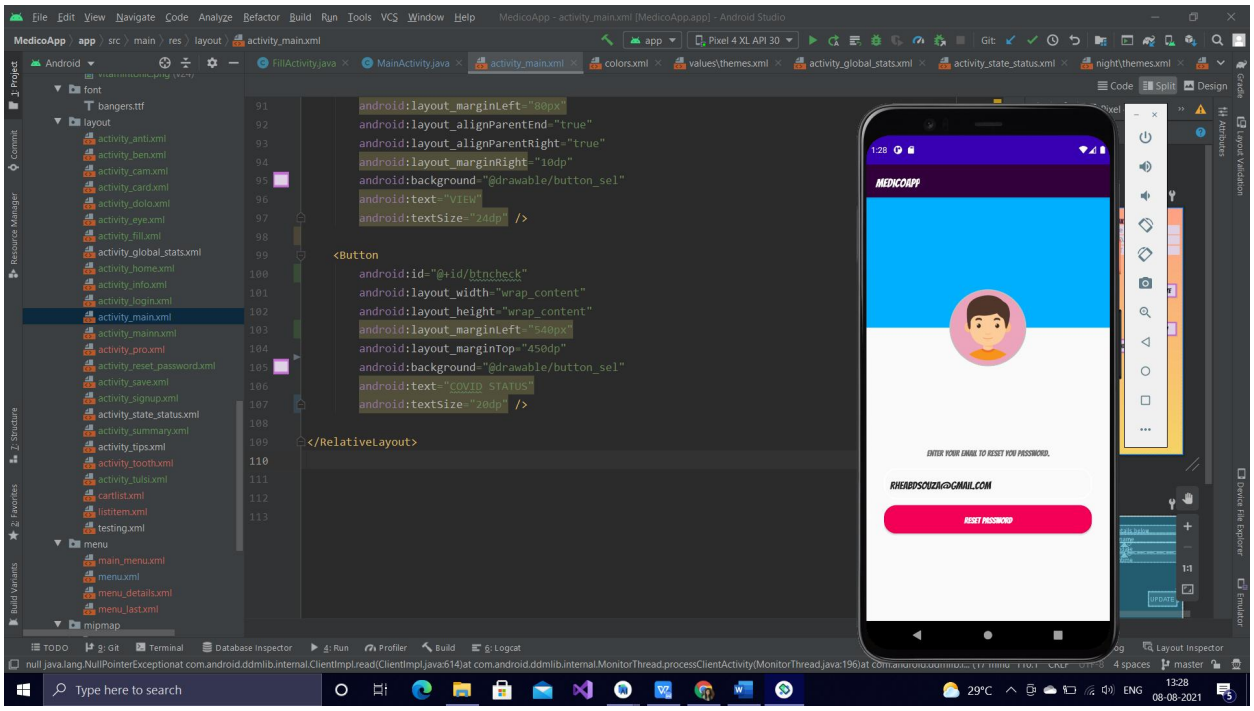


Fig 4.1.11: Forgot Password

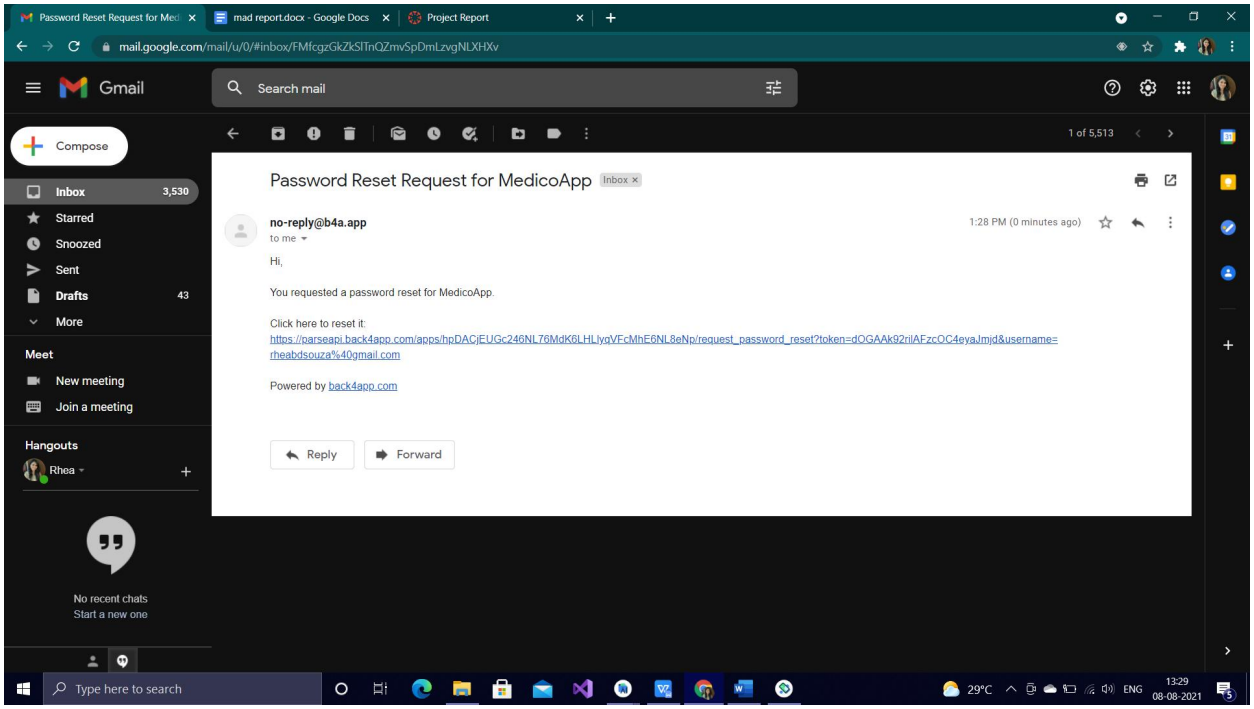


Fig 4.1.12: Link sent to mail

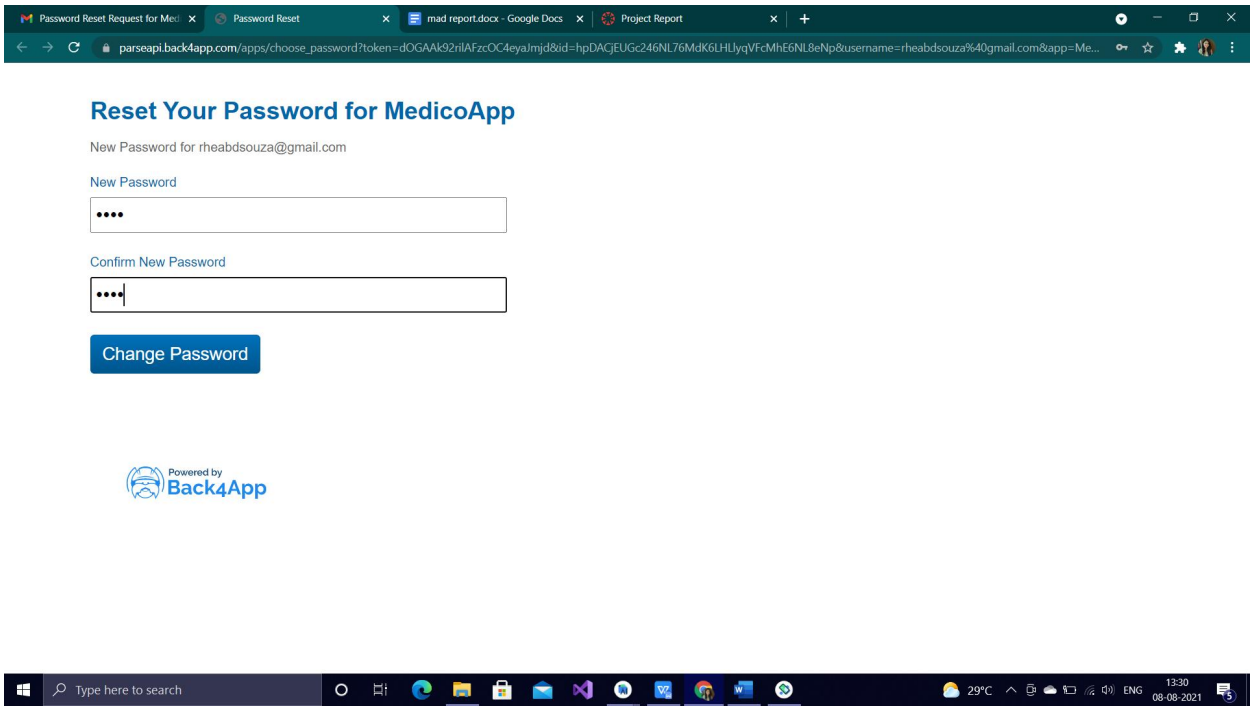


Fig 4.1.13: Reset Password

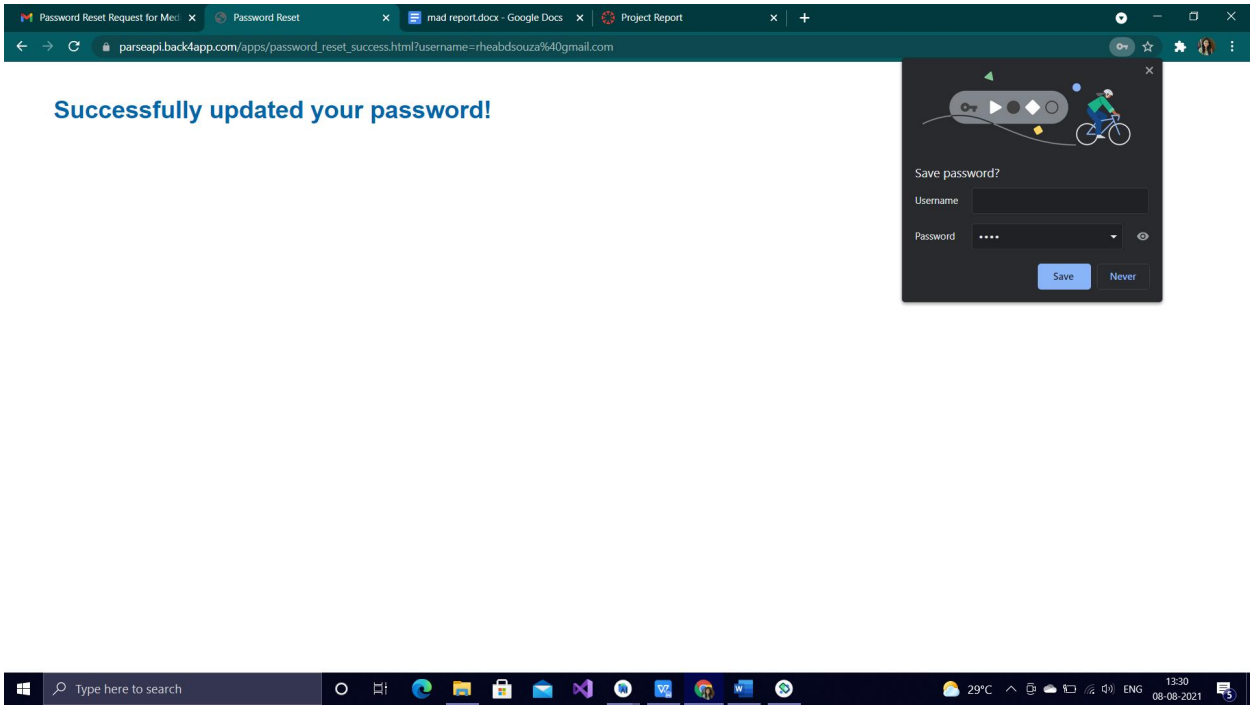


Fig 4.1.14: Successful Updation

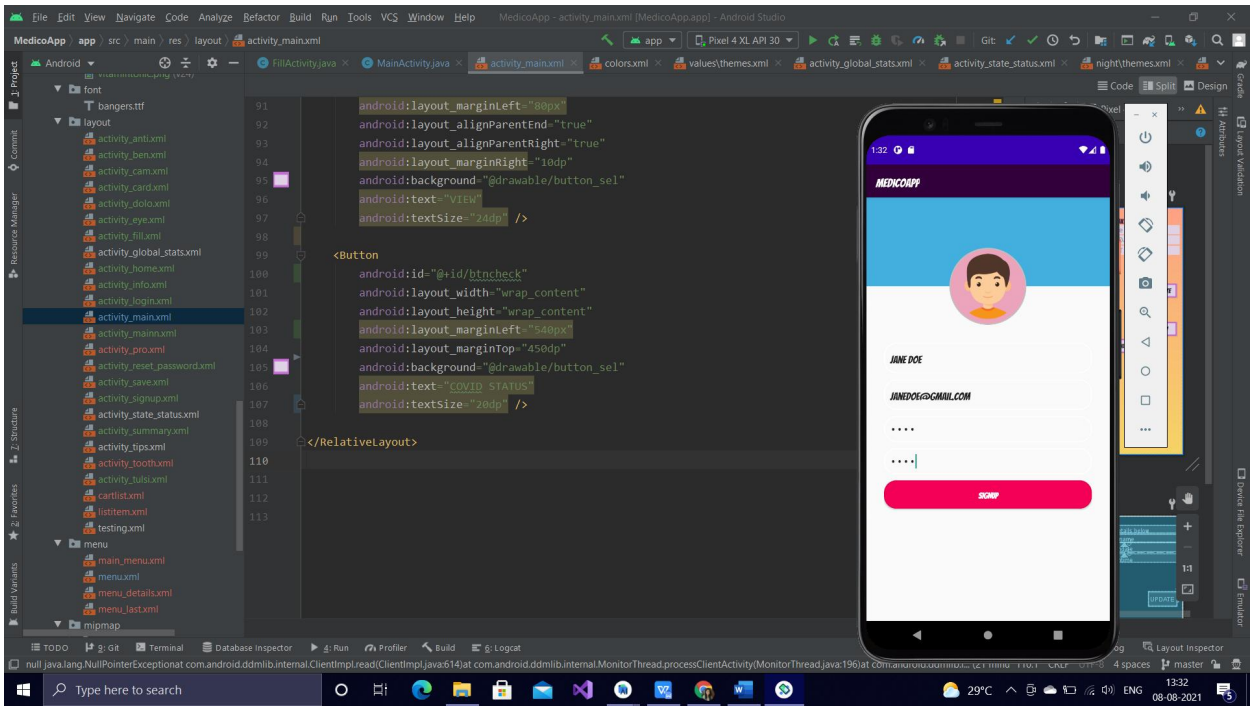


Fig 4.1.15: Create New User

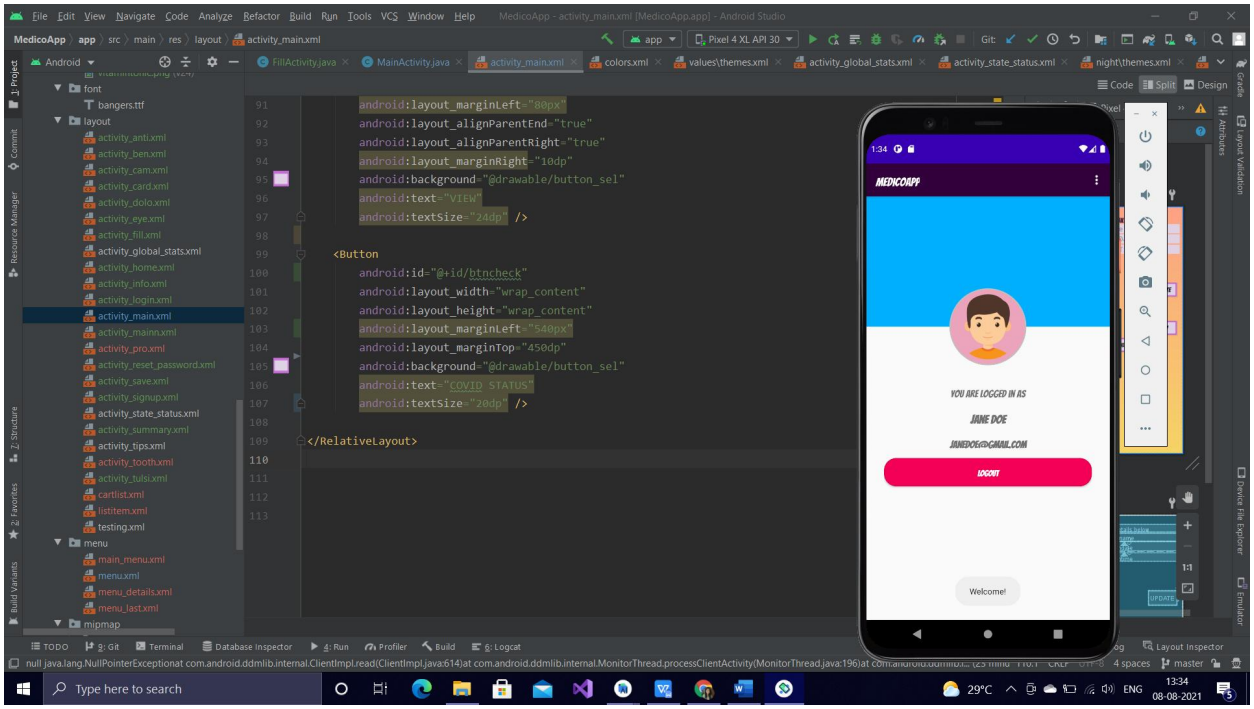


Fig 4.1.16: Successful creation

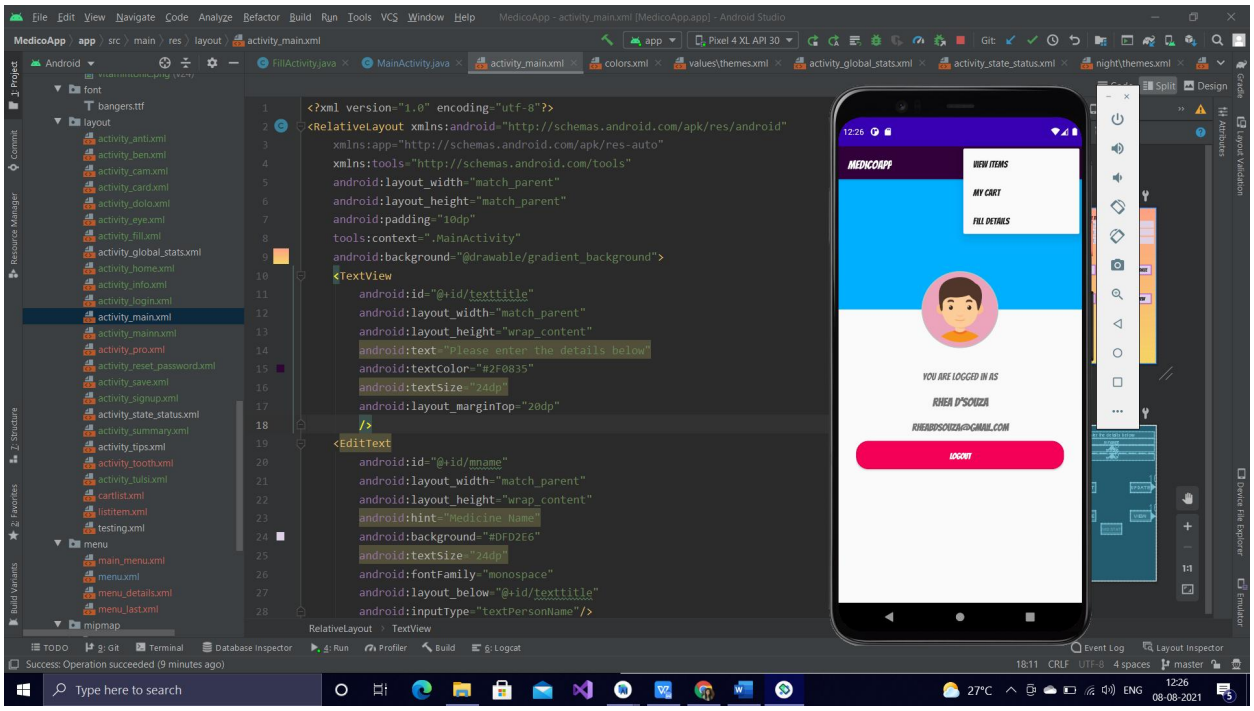


Fig 4.1.17: Navigate to view items

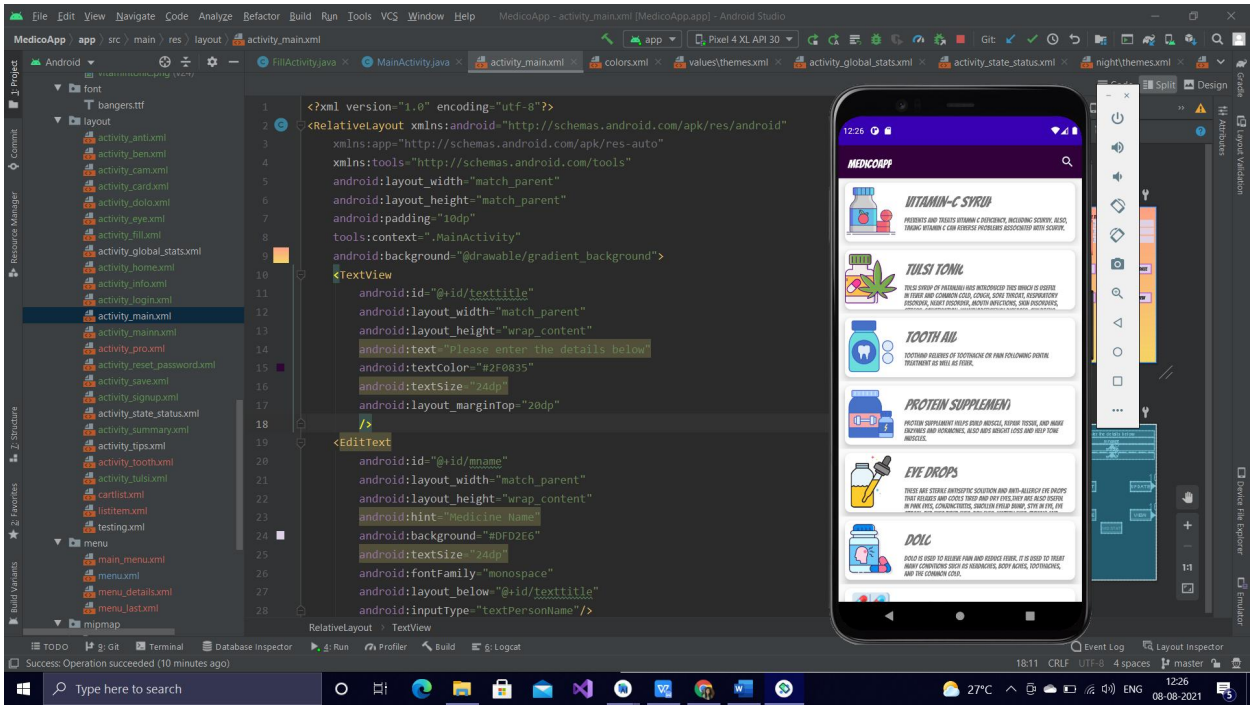


Fig 4.1.18: View items

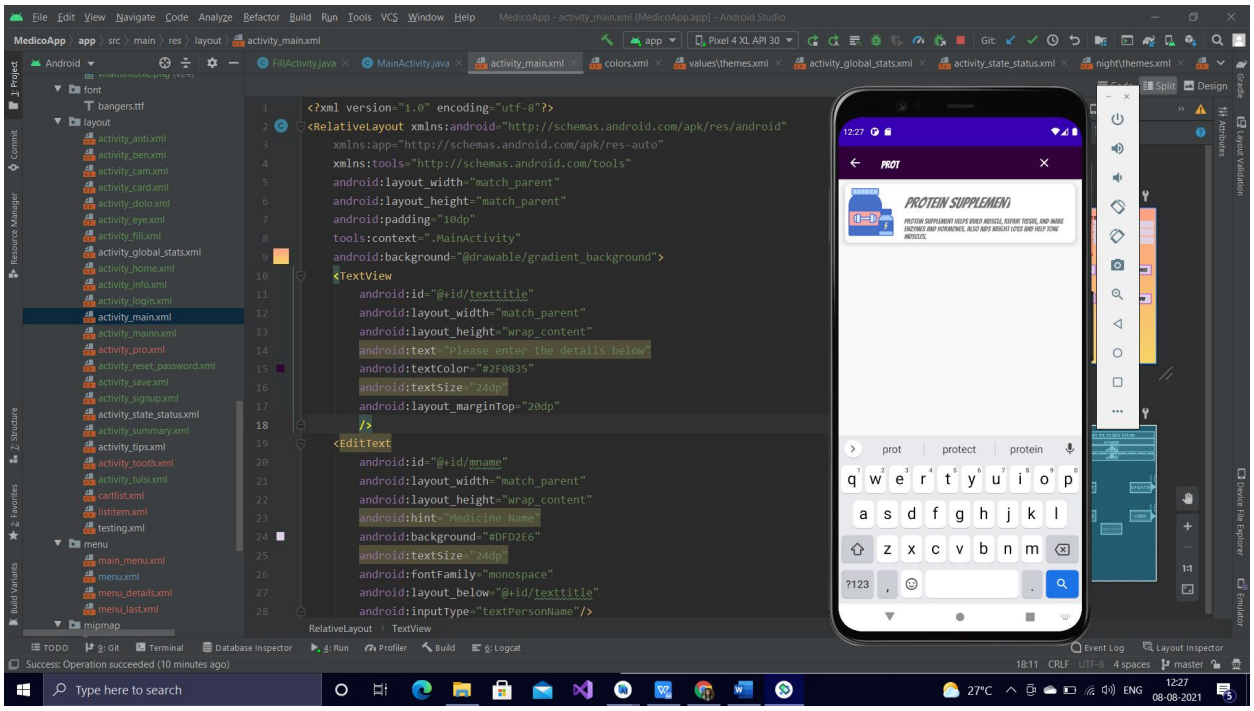


Fig 4.1.19: Search Functionality

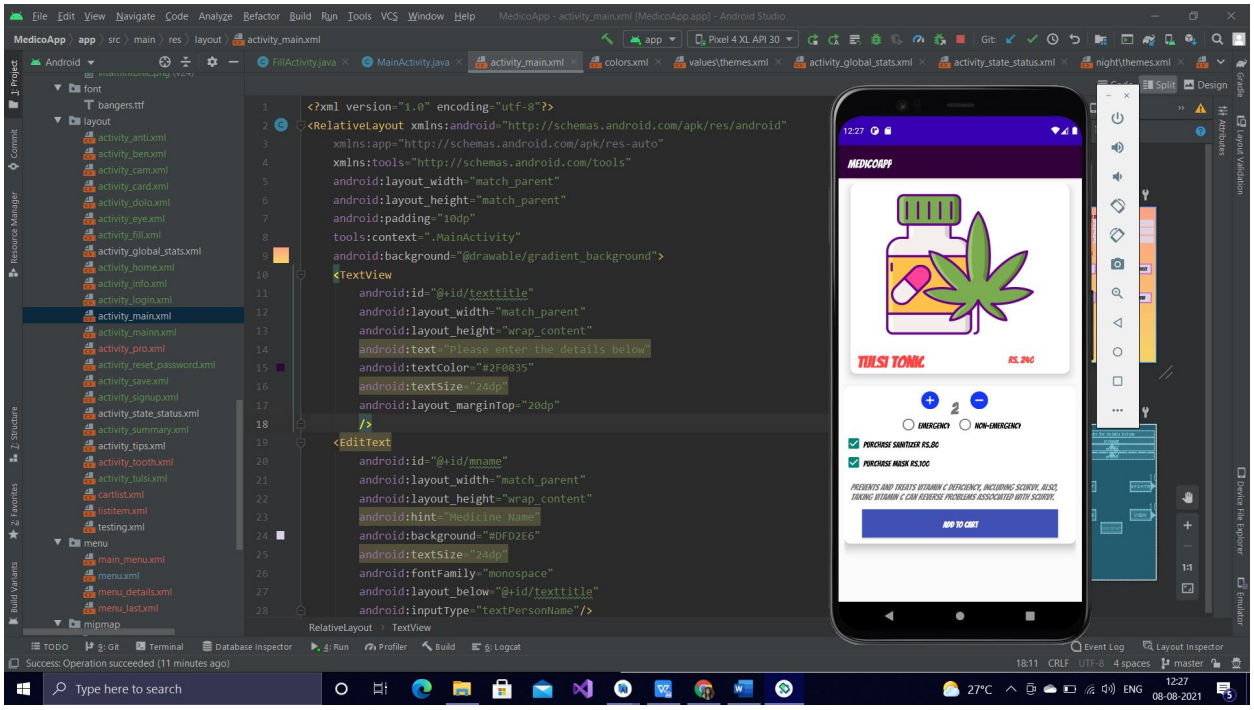


Fig 4.1.20: Select item

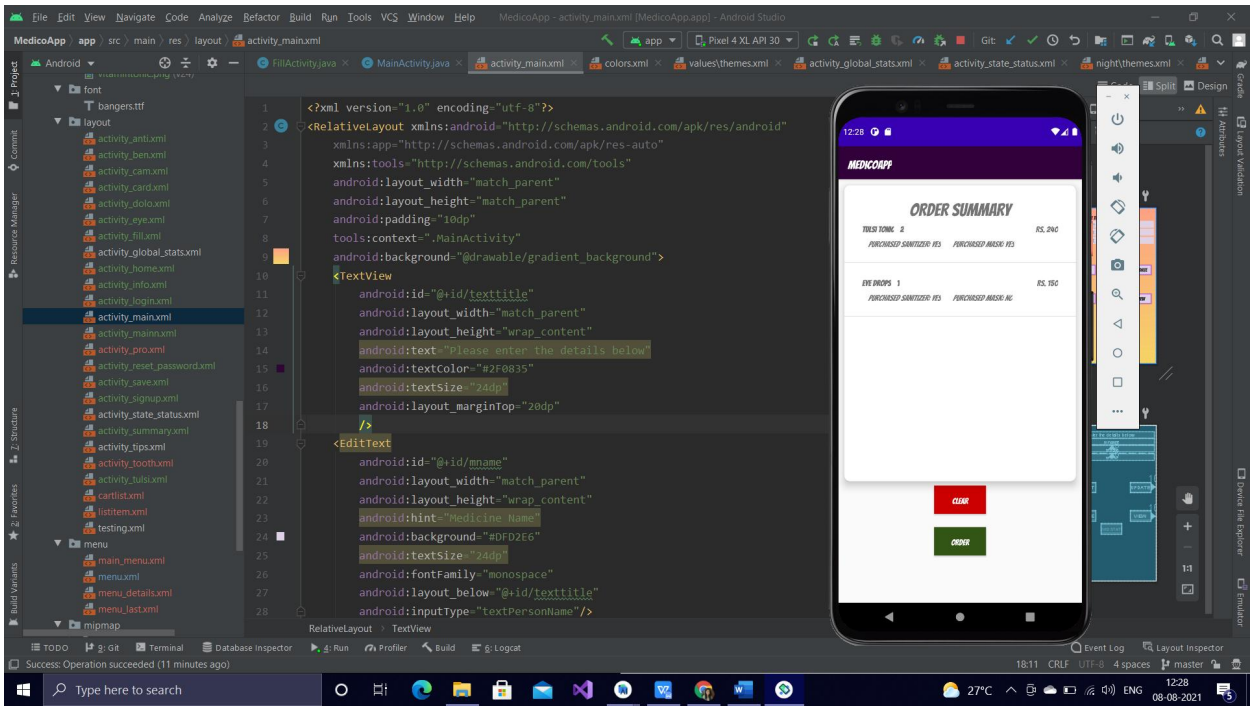


Fig 4.1.21: Order summary

By- Rhea Dsouza

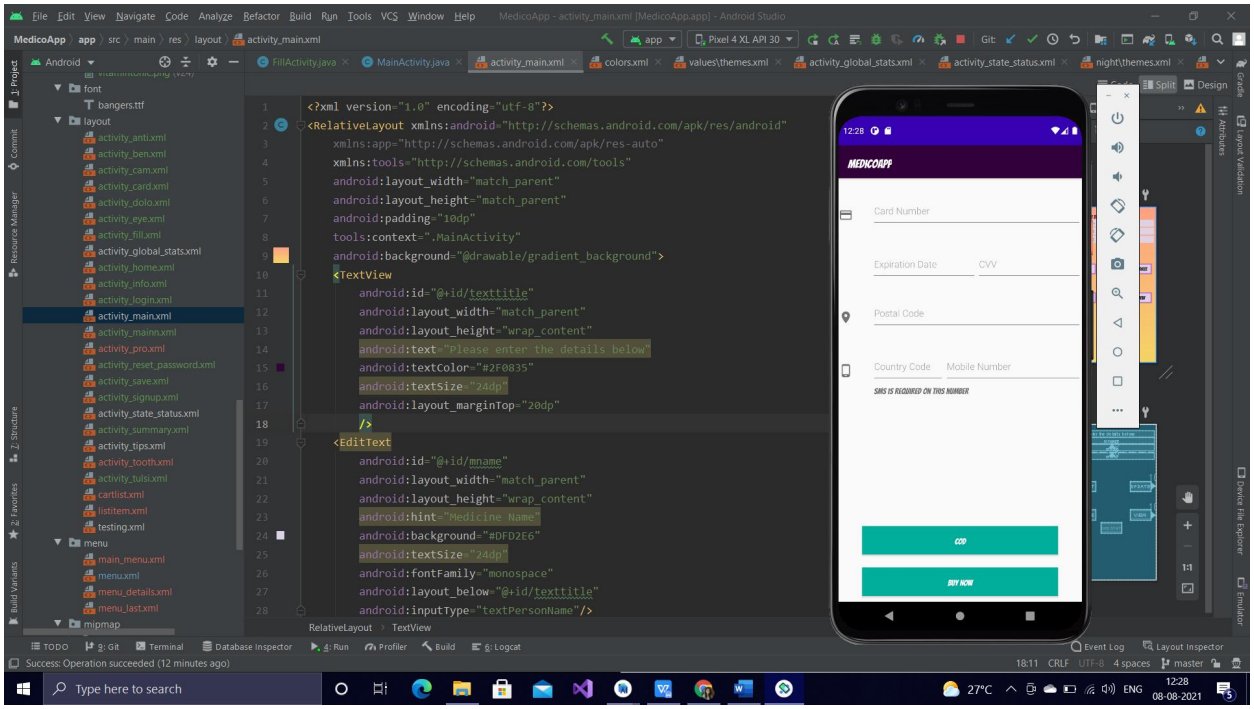


Fig 4.1.22: Payment gateway

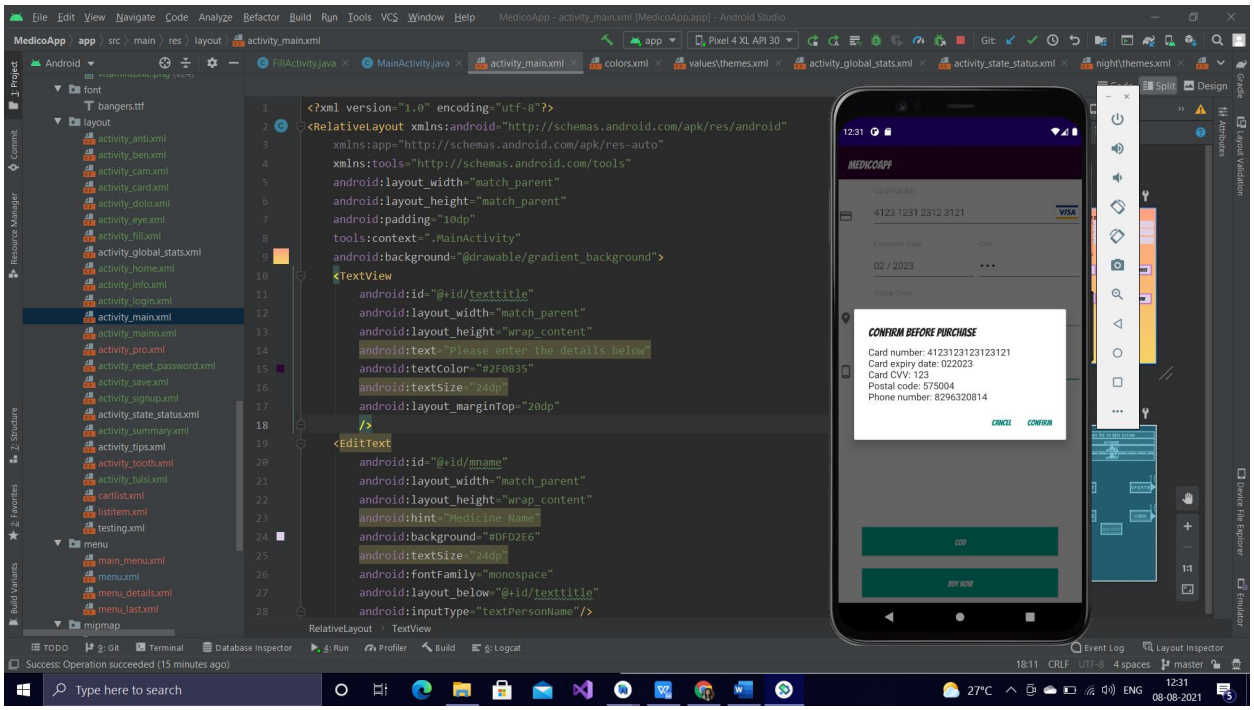


Fig 4.1.23: Confirm payment

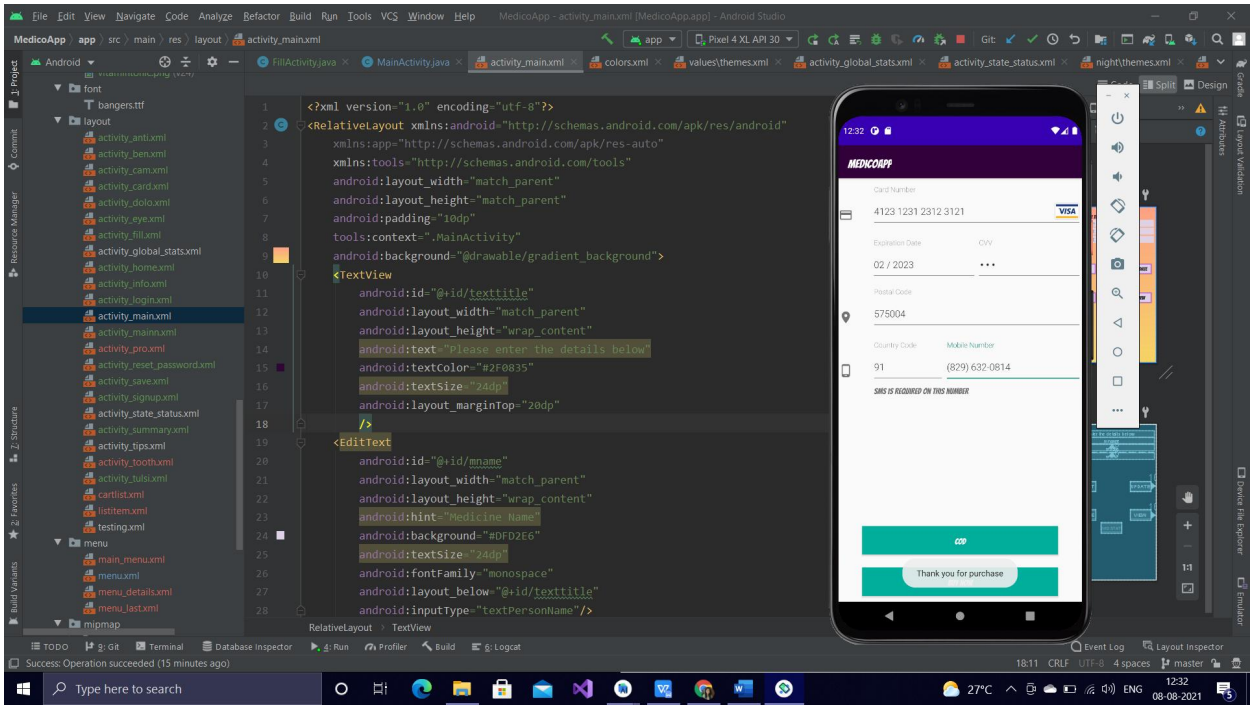


Fig 4.1.24: Successful Payment

4.2 Method Profiling

The DDMS perspective is open for method profiling. The application to be traced is selected and profiling tab is clicked to start method tracing. The test activity is then executed with the method tracing occurring in the background.

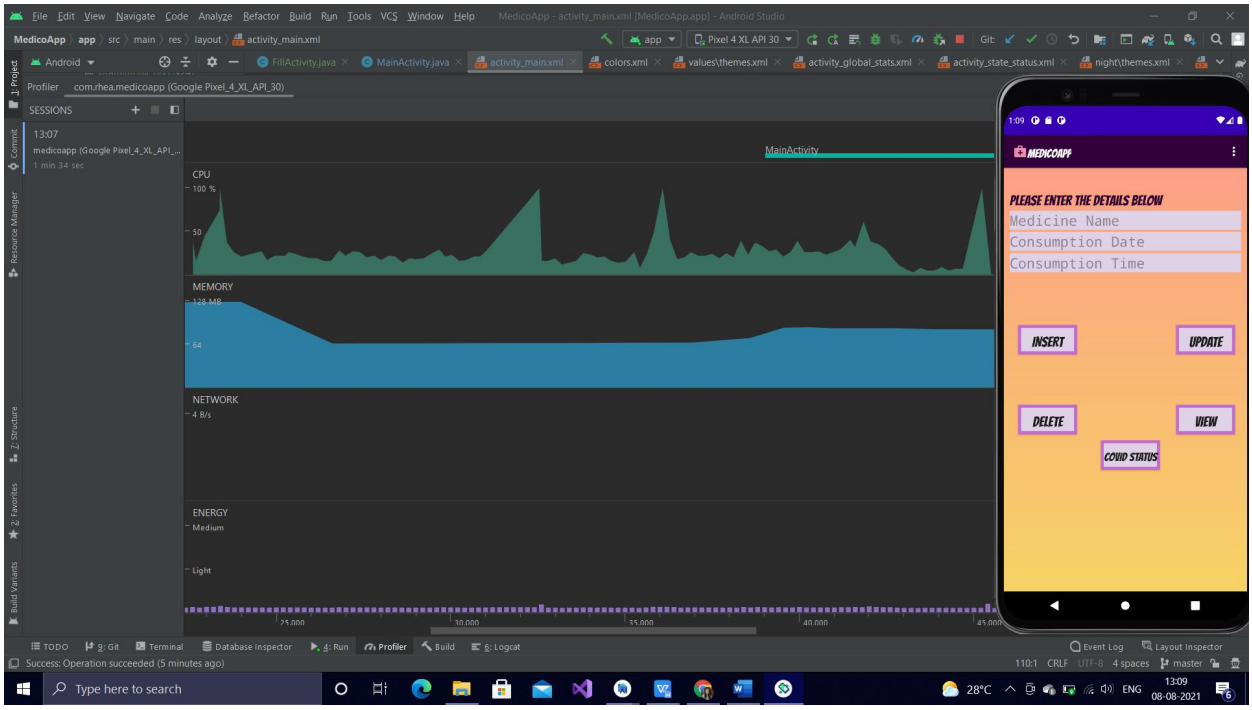


Fig: 4.2.1

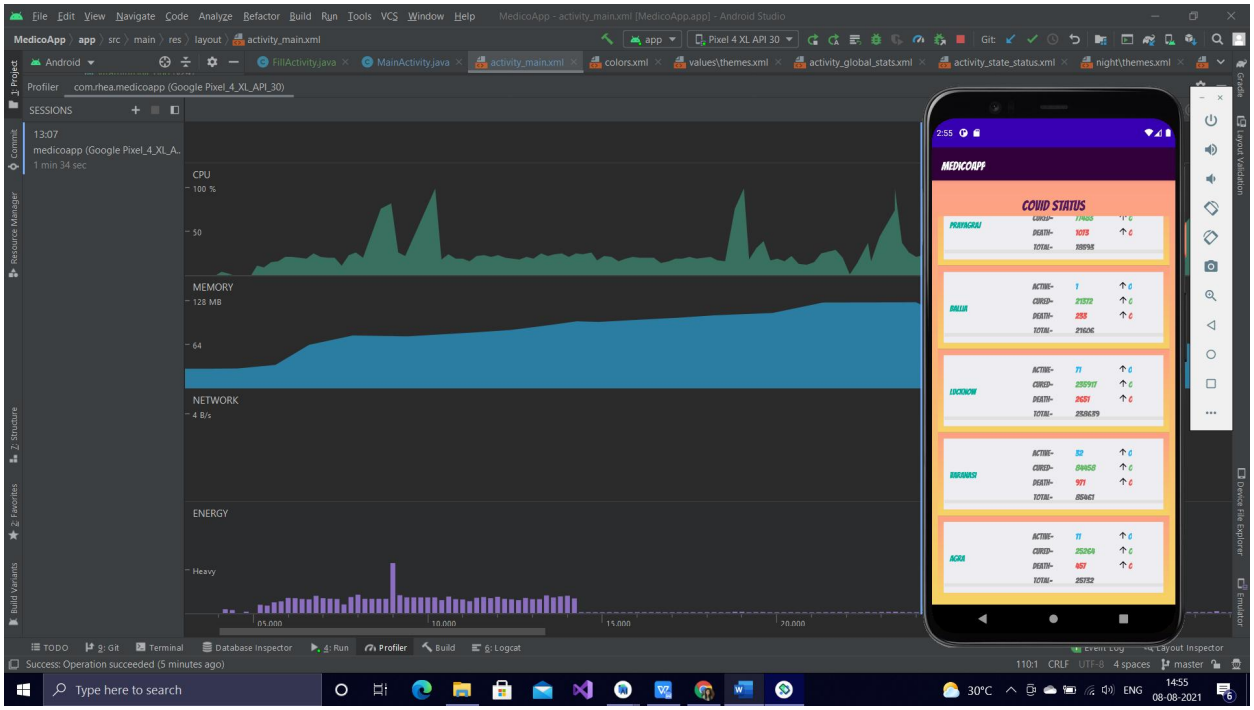


Fig 4.2.2

By- Rhea Dsouza

References

1) https://www.geeksforgeeks.org/android-tutorial/

2) https://www.tutorialspoint.com/android/android_studio.htm

3) Erik Hellman, “Android Programming –Pushing the Limits”,1stEdition, Wiley India Pvt Ltd, 2014.

4) Dawn Griffiths and David Griffiths, “Head First Android Development”,1stEdition, O‟Reilly SPD Publishers, 2015.

5) https://developer.android.com/training/volley

Appendix - A

A.1 Android

Android is an open source software stack for mobile device that includes an operating system, middleware and applications. Android software was initially developed by Android Inc., which was purchased by Google Inc., in 2005. Google published the entire source code in 2008 under the Apache License.

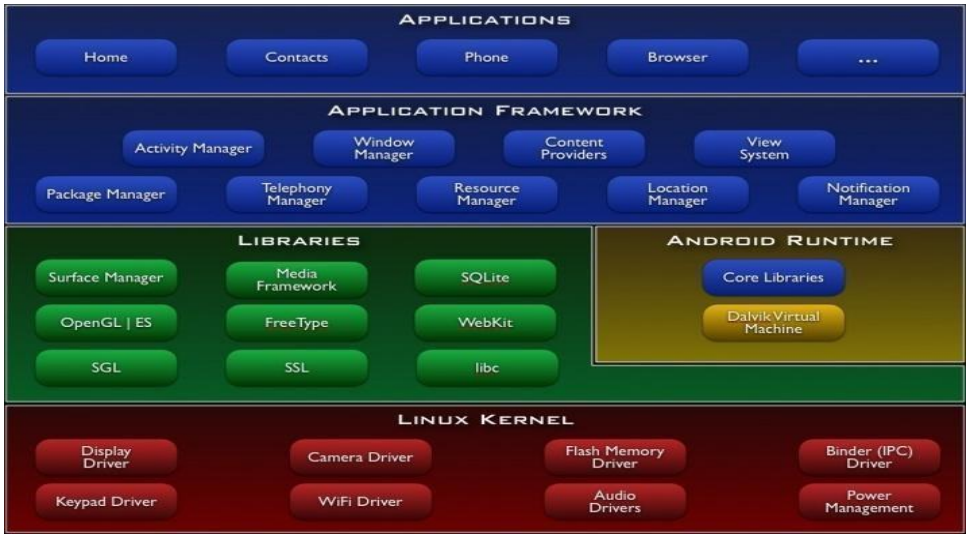
The Android Open Source Project is currently the official site that maintains the Android Source code and development cycle.The Android mobile operating system is developed based on a modified Linux 2.6 kernel. The difference between the modified and standard Linux kernels is some of the libraries and drivers are either modified or completely rewritten to maximize the efficiency of running on mobile systems with limited resources. The Android community has implemented its own C library(Bionic C) and Java virtual machine(Dalvik Virtual Machine, or DVM).

The version history of the Android mobile operating system began with the release of the Android beta in November 2007. The first commercial version, Android 1.0, was released in September 2008. Android is under ongoing development by Google and the Open Handset Alliance (OHA), and has seen a number of updates to its base operating system since its initial release.

Since April 2009, Android versions have been developed under a confectionery-themed code name and released in alphabetical order: Cupcake (1.5), Donut (1.6), Eclair (2.0-2.1), Froyo (2.2-2.2.3), Gingerbread (2.3-2.3.7), Honeycomb (3.0-3.2.6), Ice Cream Sandwich (4.0-4.0.4), Jelly Bean (4.1-4.3), and KitKat (4.4-4.4.2). On 3 September 2013, Google announced that 1 billion activated devices now use the Android OS worldwide. The most recent major Android update was KitKat 4.4, which was released to commercial devices on 22 November 2013.

A.2 Android Architecture

Android system architecture is composed of four layers. Mobile applications are on up-permost layer, underneath that the second layer contains the application framework. The application framework is a built-in toolkit, which provides set of services to the Android developers in order to build innovative and efficient Android applications. The third layer provides the C/C++ native libraries and Android Runtime (which further consists of two modules, Dalvik virtual machine and Android core libraries). The last layer is Linux kernel that manages low level resources; such as memory management, power management, hardware drivers, process management, etc.



Android Architecture

A.2.1 Application Layer

Android provides a lot of applications which come with its release including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language. Moreover, the number of developers who interested in developing Android's application is increasing and provides a huge market of application to chose.

A.2.2 Application Framework Layer

Application framework enables reuse and replacement of components. Unlike Window mobile which restricts developer from system API, developers have full access to the same framework APIs used by the core's applications. The strength points of Android is that the application architecture is designed for reusing of components which means any application can publish its capabilities and any other application may the make use of those capabilities based on the definition of Intent. Android provide a set of services and system, including:

- Views: View in Android is very different from the definition of "view" in Symbian OS or Window mobile. It can be a list, a grid, text box, button or even an embedded web browser.

- Content provider: store and retrieve data and make it accessible to all applications. They are the only way to share data across applications; there's no common storage area that all Android packages can access.

- Resource manager: Resources are external files that are used by developer's code and compiled into their application at build time. Android supports a number of different kinds of resource files, including XML, PNG, and JPEG files. Resources are externalized from source code, and XML files are compiled into a binary, fast loading format for efficiency reasons. Strings likewise are compressed into a more efficient storage form. All resource has its own id and will be added to special interface file R.java automatically.

- Notification manager: enables all application to display custom alerts in the status bar notifying user of what happens in the back ground.

- Activity manager: manages the lifecycle of application and provide common navi- gation back stack. An activity focuses on what user can do by interacting with user. Activity will for example create a window and place UI component to be displayed to user.

A.2.3 Libraries

The libraries are all written in C/C++ internally, but will be called through Java interfaces. These capabilities are exposed to developers through the Android application framework. Some of core libraries are:

- System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices.
- Media Libraries - based on PacketVideo's OpenCORE; the libraries support play- back and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG,and PNG
- Surface Manager - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
- LibWebCore - a modern web browser engine which powers both the Android browser and an embeddable web view
- SGL - the underlying 2D graphics engine
- 3D libraries - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- FreeType - bitmap and vector font rendering
- SQLite - a powerful and lightweight relational database engine available to all applications

A.2.4 Android Runtime

A set of core libraries which provides most the functionality available in the core library of java programming language. Android runtime includes the Dalvik Virtual Machine. Dalvik runs dex files, which are coverted at compile time from standard class and jar files. Dex files are more compact and efficient than class files, an important consideration for the limited memory and battery powered devices that Android targets. The core Java libraries are also part of the Android runtime. They are written in Java, as is everything above this layer. Here,

Android provides a substantial set of the Java 5 Standard Edition packages, including Collections, I/O, and so forth.

A.2.5 Dalvik Virtual Machine

Dalvik is a major piece of Google's Android, runs Java platform applications which have been converted into a compact Dalvik Excutable format suitable for systems that are con- strained in terms of memory and processor speed. Unlike most virtual machines and true java virtual machine which are stack machines, Dalvik VM is a register based architecture.

Stack based machines must use instructions to load data on the stack and manipulate that data and thus require more instructions than register machines to implement the same high level code.However, the instructions in a register machine must encode the source and des- tination registers and therefore tend to be larger. This difference is primarily of importance to VM interpreters for whom opcode dispatch tends to be expensive and other factors are relevant for JIT compilation. Being optimized for low memory requirements, Dalvik VM use less space, has no JIT compiler and uses its own byte code, not java byte code.

A.2.6 Linux Kernel:

Starting at the bottom is the Linux kernel. Android uses it for its device drivers, memory management, process management and networking. How ever we will never be program- ming to this layer directly. Android relies on Linux kernel version 2.6 for core system services. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

A.3 Parsing PowerTutor log file

PowerTutor generates a text based log file that contains per process power consumption.The applications running on device or emulator are identified by their process id. PowerTutor log file gives per second power consumed (in milliwatts) by the processes running on the device.

For retrieving the power value of specific application from this log file a JAR file called PowerTutorParser.jar was used which is freely available over internet. This JAR file takes the log file as input with the application process id and generates a CSV file that can be used to estimate the power consumption of the application of interest.

By- Rhea Dsouza