Progress Report on App Development (Week 05)

-by Ritesh Kumar

An Intern in UpSkill Campus, 20/07/2023

Ritesh Kumar

I am pleased to present you with a comprehensive report on app development (Week 05), which provides an overview of “Simple Alarm Setter in Android App” with the process, challenges, and best practices for successful app development. This report aims to make understanding of the key aspects of app development and making informed decisions in this domain.

**Progress Report on App Development (Week 05)**

# An Overview on App Development

In the 5th week of “6 Weeks of App Development – Project-Based Learning”. We will be building a Simple Alarm Setter Project using **Java** and **XML** in Android. Alarm plays a vital role in our day-to-day life. Nowadays alarm has become our wake-up assistant. Every mobile phone is associated with an alarm app. We will create this app using android studio. Android Studio provides a great unified environment to build apps for Android phones, tablets, Android Wear, Android TV, and Android Auto because it provides a very large number of app building features and it is also very easy to use.

Let’s move toward the project

* A Simple Alarm Setter in Android App

## Content:

* Create a New Project
* Working with the activity\_main.xml file
* Working with the MainActivity.java file
* Source Code
* Working with BroadCastReceiver (AlarmReceiver) class
* Playing with colors
* Changing theme of the app
* Adding permission in “AndroidManifest.xml”
* Output

### Challenges and Hurdles:

* **Working with BroadCastReceiver (AlarmReceiver) class**
* **Playing with colors**
* **Changing theme of the app**
* **Adding permission in “AndroidManifest.xml”**

#### Lesson Learned:

* **A Simple Alarm Setter in Android App**

We are going to implement this project using the **Java**language.

**Step By Step Implementation of Project :**

**Step 1: Create a New Project**

To create a new project in Android Studio please refer to How to Create/Start a New Project in Android Studio. Note that select Java as the programming language.

**Step 2: Working with the activity\_main.xml file**

Navigate to the app > res > layout > activity\_main.xml and add the below code to that file. In this file, we have added two items ‘TimePicker’ and ‘ToggleButton’. TimePicker is used to capture the alarm time and ToggleButton is added to set the alarm on or off. Initially, ToggleButton is set to off. It is set on when an alarm is set. Below is the code for the activity\_main.xml file.

**Xml**

<?**xml** version="1.0" encoding="utf-8"?>

<**LinearLayout**

    xmlns:android="<http://schemas.android.com/apk/res/android>"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    android:orientation="vertical">

    <!--Added Time picker just to pick the alarm time-->

    <!--gravity is aligned to center-->

    <**TimePicker**

        android:id="@+id/timePicker"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:layout\_gravity="center" />

    <!--Added Toggle Button to set the alarm on or off-->

    <!--ByDefault toggleButton is set to false-->

    <**ToggleButton**

        android:id="@+id/toggleButton"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:layout\_gravity="center"

        android:layout\_margin="20dp"

        android:checked="false"

        android:onClick="OnToggleClicked" />

    <!--"OnToggleClicked" method will be implemented in MainActivity.java -->

</**LinearLayout**>

**Step 3: Working with the MainActivity.java file**

Go to **MainActivity.java**Class. In MainActivity.java class onToggleClicked( ) method is implemented in which the current hour and the minute is set using the calendar. Alarm services are implemented using AlarmManager class. The alarm is set in such a way that it rings and vibrates repeatedly until the toggle button is turned off.

Below is the code for the **MainActivity.java** file. Comments are added inside the code to understand the code in more detail.

**Java**

**import** android.app.AlarmManager;

**import** android.app.PendingIntent;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.TimePicker;

**import** android.widget.Toast;

**import** android.widget.ToggleButton;

**import** androidx.appcompat.app.AppCompatActivity;

**import** java.util.Calendar;

**public** **class** MainActivity **extends** AppCompatActivity {

    TimePicker alarmTimePicker;

    PendingIntent pendingIntent;

    AlarmManager alarmManager;

    @Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

        setContentView(R.layout.activity\_main);

        alarmTimePicker = (TimePicker) findViewById(R.id.timePicker);

        alarmManager = (AlarmManager) getSystemService(ALARM\_SERVICE);

    }

    // OnToggleClicked() method is implemented the time functionality

**public** **void** OnToggleClicked(View view) {

**long** time;

**if** (((ToggleButton) view).isChecked()) {

            Toast.makeText(MainActivity.**this**, "ALARM ON", Toast.LENGTH\_SHORT).show();

            Calendar calendar = Calendar.getInstance();

            // calendar is called to get current time in hour and minute

            calendar.set(Calendar.HOUR\_OF\_DAY, alarmTimePicker.getCurrentHour());

            calendar.set(Calendar.MINUTE, alarmTimePicker.getCurrentMinute());

            // using intent i have class AlarmReceiver class which inherits

            // BroadcastReceiver

            Intent intent = **new** Intent(**this**, AlarmReceiver.**class**);

            // we call broadcast using pendingIntent

            pendingIntent = PendingIntent.getBroadcast(**this**, 0, intent, 0);

            time = (calendar.getTimeInMillis() - (calendar.getTimeInMillis() % 60000));

**if** (System.currentTimeMillis() > time) {

                // setting time as AM and PM

**if** (Calendar.AM\_PM == 0)

                    time = time + (1000 \* 60 \* 60 \* 12);

**else**

                    time = time + (1000 \* 60 \* 60 \* 24);

            }

            // Alarm rings continuously until toggle button is turned off

            alarmManager.setRepeating(AlarmManager.RTC\_WAKEUP, time, 10000, pendingIntent);

            // alarmManager.set(AlarmManager.RTC\_WAKEUP, System.currentTimeMillis() + (time \* 1000), pendingIntent);

        } **else** {

            alarmManager.cancel(pendingIntent);

            Toast.makeText(MainActivity.**this**, "ALARM OFF", Toast.LENGTH\_SHORT).show();

        }

    }

}

**Step 3: Working with BroadCastReceiver (AlarmReceiver) class**

Create a new java class named “AlarmReceiver.java” at the same place where MainActivity.java class resides. In this class onReceive() method is implemented. Here we have added vibration functionality and a default ringtone that starts to vibrate and ring when the alarm time is scheduled. Below is the code for the AlarmReceiver.java file. Comments are added inside the code to understand the code in more detail.

**Java**

**import** android.content.BroadcastReceiver;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.media.Ringtone;

**import** android.media.RingtoneManager;

**import** android.net.Uri;

**import** android.os.Build;

**import** android.os.Vibrator;

**import** android.widget.Toast;

**import** androidx.annotation.RequiresApi;

**public** **class** AlarmReceiver **extends** BroadcastReceiver {

    @RequiresApi(api = Build.VERSION\_CODES.Q)

    @Override

    // implement onReceive() method

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

        // we will use vibrator first

        Vibrator vibrator = (Vibrator) context.getSystemService(Context.VIBRATOR\_SERVICE);

        vibrator.vibrate(4000);

        Toast.makeText(context, "Alarm! Wake up! Wake up!", Toast.LENGTH\_LONG).show();

        Uri alarmUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE\_ALARM);

**if** (alarmUri == **null**) {

            alarmUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE\_NOTIFICATION);

        }

        // setting default ringtone

        Ringtone ringtone = RingtoneManager.getRingtone(context, alarmUri);

        // play ringtone

        ringtone.play();

    }

}

**Step 5: Playing with the colors**

Go to the “values” folder first then choose the colors.xml file. In the colors.xml file, you can keep colors of your choice as many as you want to use in your app. You have to just give the name and put the color code of the respective colors. I have kept the AppBar color as “#0F9D58” which we have named as “colorPrimary”.

**Xml**

<?**xml** version="1.0" encoding="utf-8"?>

<**resources**>

    <**color** name="colorPrimary">#0F9D58</**color**>

    <**color** name="colorPrimaryDark">#0F4C2E</**color**>

    <**color** name="colorAccent">#9D0F9B</**color**>

</**resources**>

**Step 6: Changing the theme of the app**

Go to the “values” folder first then choose the themes.xml file. In the theme.xml file, we have used “Theme.AppCompat.Light.DarkActionBar” which is a light theme with a dark ActionBar. We can use a light theme with a light action bar using “Theme.AppCompat.Light.LightActionBar”, it all depends on our choice and need.

**Xml**

<**resources**>

    <!-- Base application theme. -->

    <**style** name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">

        <!-- Customize your theme here. -->

        <**item** name="colorPrimary">@color/colorPrimary</**item**>

        <**item** name="colorPrimaryDark">@color/colorPrimaryDark</**item**>

        <**item** name="colorAccent">@color/colorAccent</**item**>

    </**style**>

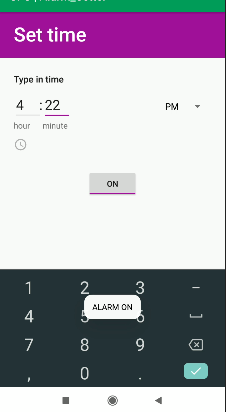
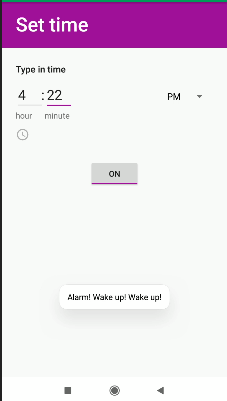
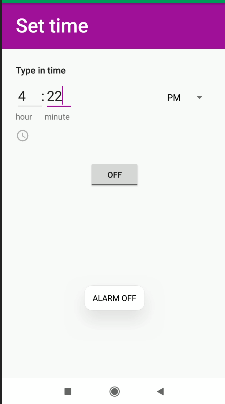
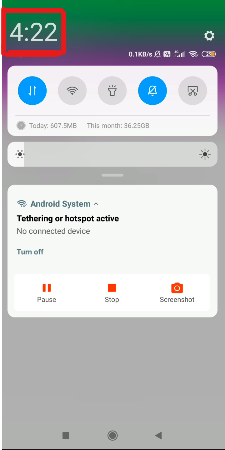
</**resources**>

**Step 7: Add permission in “AndroidManifest.xml”**

Go to the “AndroidManifest.xml” file. A BroadcastReceiver is registered in AndroidManifest.xml by adding a receiver section after the application section is over. Also, give permission to vibrate using:

**<uses-permission android:name=”android.permission.VIBRATE” />**

**Output:**

**   **

Here we have manually set the alarm time. You can also set it by adjusting the clock showing in front. You will have to wait for the alarm time. It will continuously show “Alarm! Wake up! Wake up!” and rings and vibrates until the toggle-button is turned off.

##### **References:**

* <https://developer.android.com/studio>
* https://developer.android.com/
* <https://www.geeksforgeeks.org/>

Please note that this report provides a general overview of app development, and specific recommendations or strategies should be tailored to your organization's unique requirements and goals. Should you require further assistance or have any questions, please do not hesitate to contact me.

Thank you for the opportunity to prepare this report, and I hope it proves valuable in your app development endeavors. I look forward to discussing this topic further and assisting you in your future projects.

Sincerely,

Ritesh Kumar (9330771068)

Email: kumarritesh6769@gmail.com