Programming for Mobile Devices 2020 – 2021

Project Report

THEME, DEVELOPMENT & IMPLEMENTATION

App Name: Alarmster

Developed by:

Todescu Paul - Eugen **Computer Science Student**
West University of Timisoara
Year 2

Supervisor:

Liviu Octavian Mafteiu-Scai

16 May, 2021

1. Abstract

The motivation of this document is to provide the required information regarding the mobile application theme and development. This document explains the technical and functional requirements of the application and provides details regarding its implementation. The aim of this report is to offer a complete overview over the application infrastructure by explaining all the included features, services and components, as well as the graphical user interface.

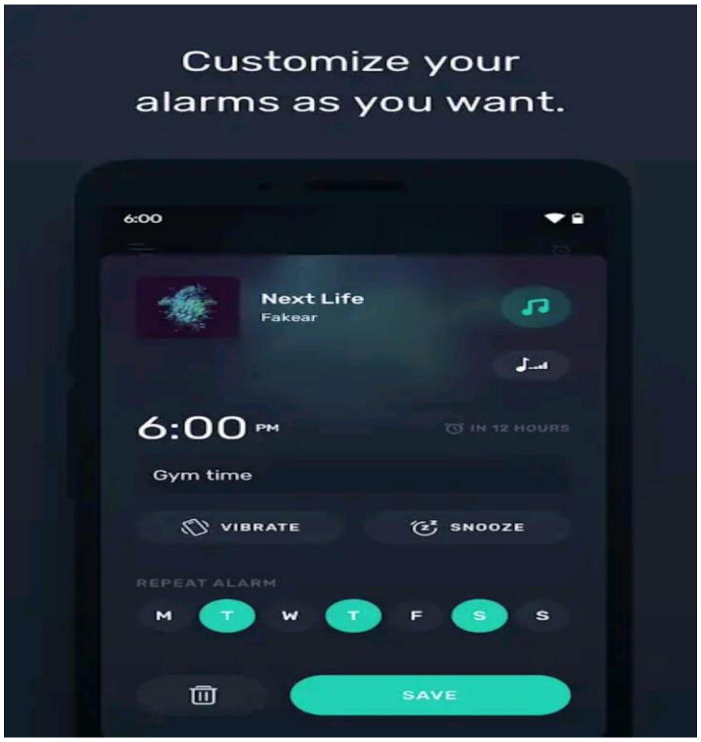
2. Purpose and Users

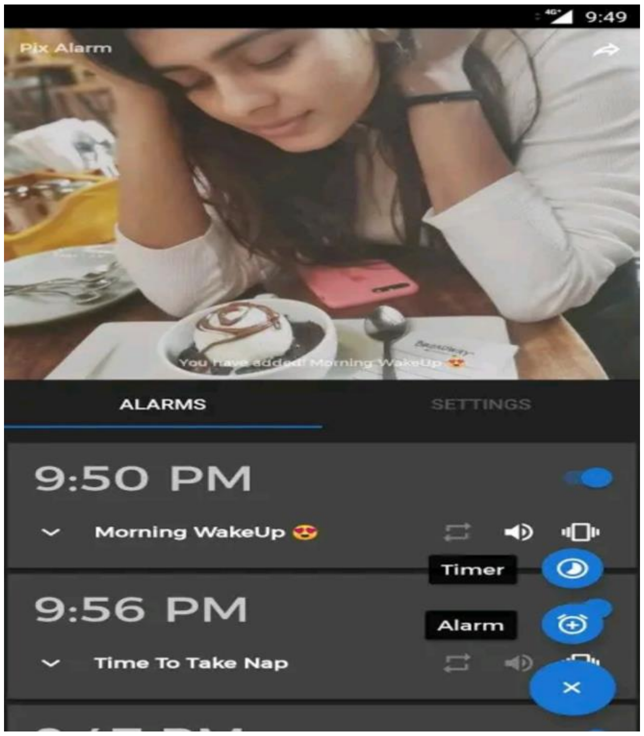
The project consists in an alarm clock application that enables users to have full control over multiple customization options. The app is oriented towards a large number of potential end users who have a fixed daily schedule and are in need for a mobile application which enables them to efficiently wake up at a specific time in a friendly manner. A lot of the times, the way a person starts the morning can set the tone for the rest of the day, consequently, the aim of the application is to facilitate this process and improve the mood and productivity of the users.

3. Original Contribution and Apps Analysis

After a thorough analysis and comparison of different apps similar to the one proposed in this report, it came to my conclusion that there aren't to many options available on the market that offer the same level of customization that this project aims to accomplish. The fact that the user will be able to set custom images, songs and text all in one place allows this app to be an original contribution to the market. Moreover, the app allows the user to select any color for chosen text message, option which I have not come across in other apps.

A quick search on the Google Play store would reveal a significant number of popular apps, such as *Alarmy* or *Pix Alarm,* which are based on similar principles as the ones described in this report. The images below illustrate the two mentioned applications.

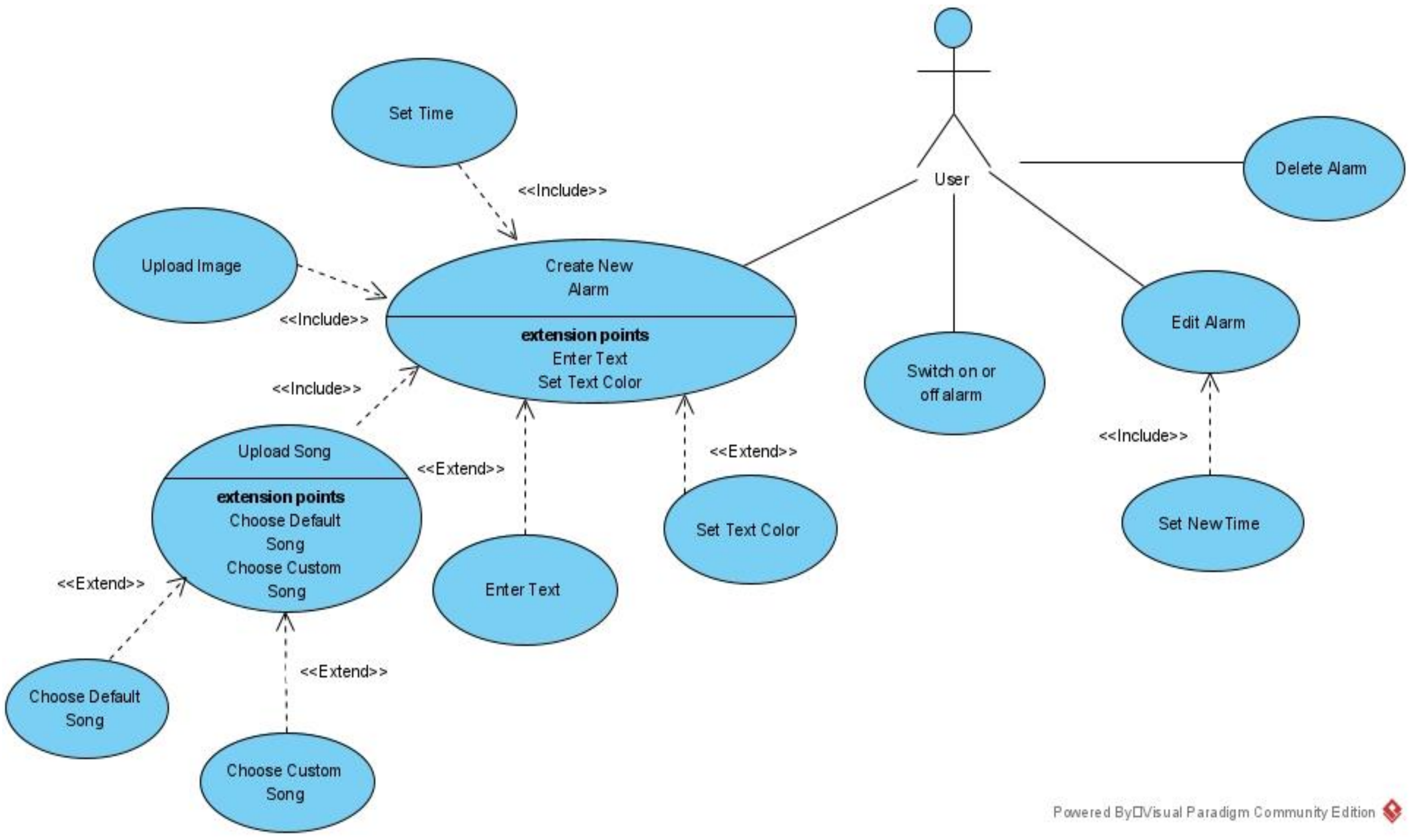




4. Functionality

The application was developed using the Android Studio IDE and the code was written with the support of the java programming language. Since the user and application settings can be private to the app, the system will store all the required data and information locally, using a MySQL database. The user interface was acomplished using various pre-built UI components that are provided by the Android's framework for XML.

The case use diagram below illustrates how the functionalities of the application are intended to be used by the end user.



5. User Manual

The application includes a rich set of features by providing multiple customization options. After opening the app, the user will be greeted with a main menu from where the option of creating a new alarm can be selected. The already created alarms, as well as the next scheduled alarm are also visible in the main menu, along with the options of toggling on and off or editing and deleting a specific alarm, according to the user's preferences. The application will allow for a maximum of seven alarms to be created, one for each day of the week.

The list provided below explains all the features that the user will be able to take advantage of while creating a new alarm.

1. The first an most important feature that is common to any alarm clock application is the possibility of setting the time when the alarm will be set off. Here, the user will have the option of choosing the week day, from Monday to Sunday, the hour and the minute at which the alarm will be triggered. The user can also choose between the 24 hour European standard or the AM/PM time format.
2. Next, an accosting signal is necessary and forms the basis of any alarm. Here, the user can either choose one of the default ringtones which are already supplied with the application, or choose to upload a song from the media library. After the selection is done, the user must press on the save button and continue with the rest of the customization options.
3. An optional feature for entering a custom text will also be presented to the user. The inputted text will be overlay-ed over the background picture in order to create an overall more customized presentation of the alarm. Here the user can also set the color for the imputed text, which by default is set to white.
4. The user will also be able to import a picture from the image gallery and set it as background for the alarm such that the chosen image will be the first thing the user sees after waking up. This feature will be optional however and a default image will be set as background if the user decides to not take advantage of this option.

After entering all the required fields, the user must press on the confirm button and the newly added alarm will be visible on the main screen.

The following options listed below are specific for each of the created alarms:

1. Users can choose to edit the time of an already existing alarm by pressing the edit button for the desired alarm in the main screen. When selecting this option, a new time picker will be displayed and the user can choose the newly desired time for that alarm.
2. A switch button is also present for each of the alarms and can be used to toggle on or off an alarm.
3. Lastly, each of the created alarm has a delete button, which can be used in order to delete an alarm. In this case, a prompt message will first be displayed which will ask for confirmation. After pressing the ok button, the chosen alarm will be deleted.

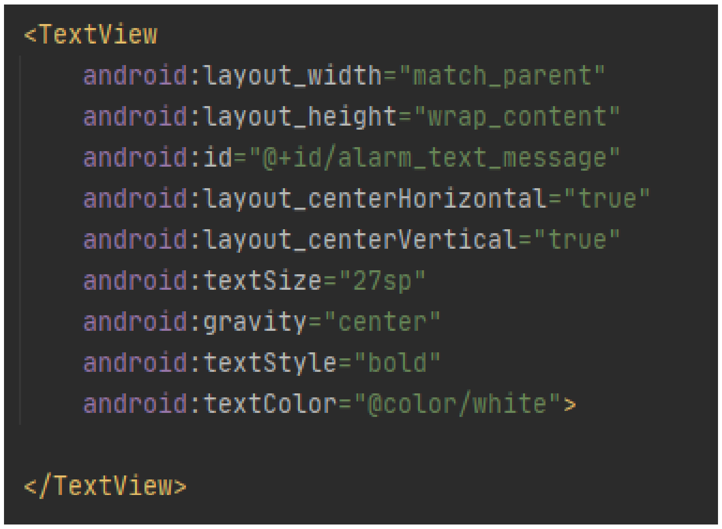
When the alarm goes off, the user can dismiss it by pressing the X button. If not dismissed, the alarm will sound by default for one minute.

6. Technical Manual

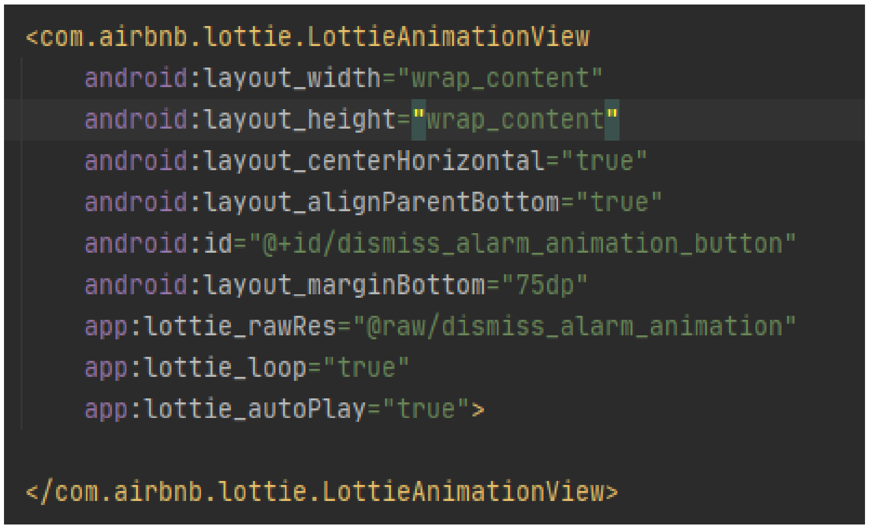
This section is intended to provide technical details regarding the implementation for the class *AlarmActivity.java*, which is used to display and sound the alarm.

The above mentioned activity is opened by a BroadcastReceiver class named *AlarmReceiver.java*, which starts the activity when the time of the alarm reaches the time of the system.

The layout for this activity is provided by the *activity_alarm.xml* resource file. It contains a relative layout with two components: a TextView for showing the message of the alarm and a LottieAnimationView for dismissing the alarm. Lottie is an external library for android which is used to render Adobe After Effects animations and is linked to the project using the following line: *implementation "com.airbnb.android:lottie:3.7.0"* in the *build.gradle* file.



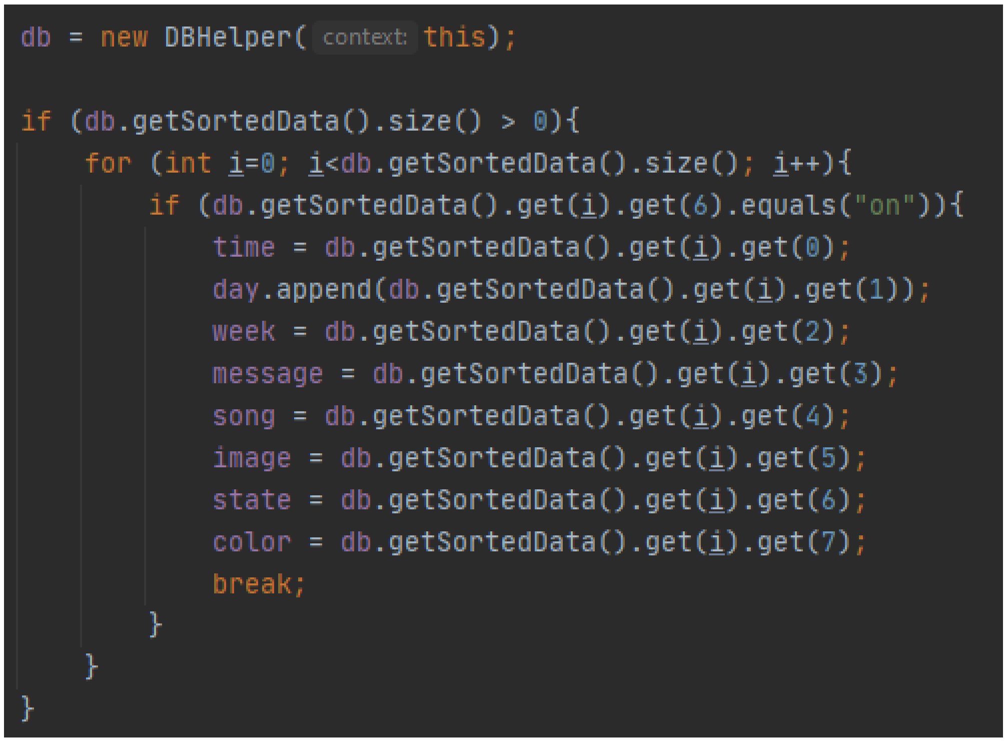
```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/alarm_text_message"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:textSize="27sp"
    android:gravity="center"
    android:textStyle="bold"
    android:textColor="@color/white">

</TextView>
```



```
<com.airbnb.lottie.LottieAnimationView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_alignParentBottom="true"
    android:id="@+id/dismiss_alarm_animation_button"
    android:layout_marginBottom="75dp"
    app:lottie_rawRes="@raw/dismiss_alarm_animation"
    app:lottie_loop="true"
    app:lottie_autoPlay="true">

</com.airbnb.lottie.LottieAnimationView>
```

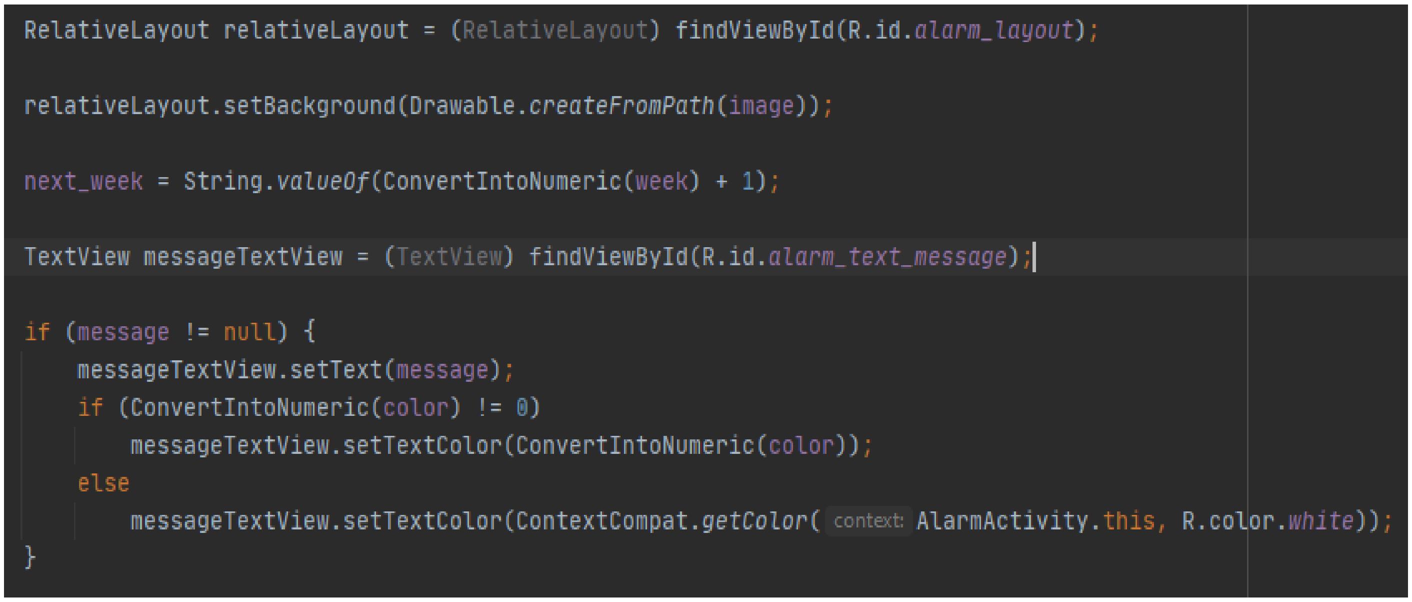
The activity is initialized in the onCreate(Bundle) method. Here, the following operations are performed:

1. First step was to add the following flags to the activity:
   - *FLAG_FULLSCREEN*, which sets the activity to be displayed in full screen.
   - *FLAG_DISMISS_KEYGUARD* and *FLAG_SHOW_WHEN_LOCKED*, in order to not show the lock-screen when the alarm launches
   - *FLAG_TURN_SCREEN_ON*, used for turning on the device when the activity is launched
   - FLAG_KEEP_SCREEN_ON, keeps the screen on until the user dismisses the alarm
2. Next, the function setContentView(int) sets the UI of the activity to the above mentioned layout resource file.
3. After that, the information regarding the triggered alarm needs to be fetched from the database. This information includes the time, day, week message, song, image, state, and color. In the code sequence below, the function getSortedData() retreives a list containing each row in the database table, sorted based on the week and day. Each row contains an alarm. In this way, since all the alarms are sorted, we only need the first alarm which has an on state, and we acomplish this using a for loop. String variables are used in order to hold and further use the information about the alarm.



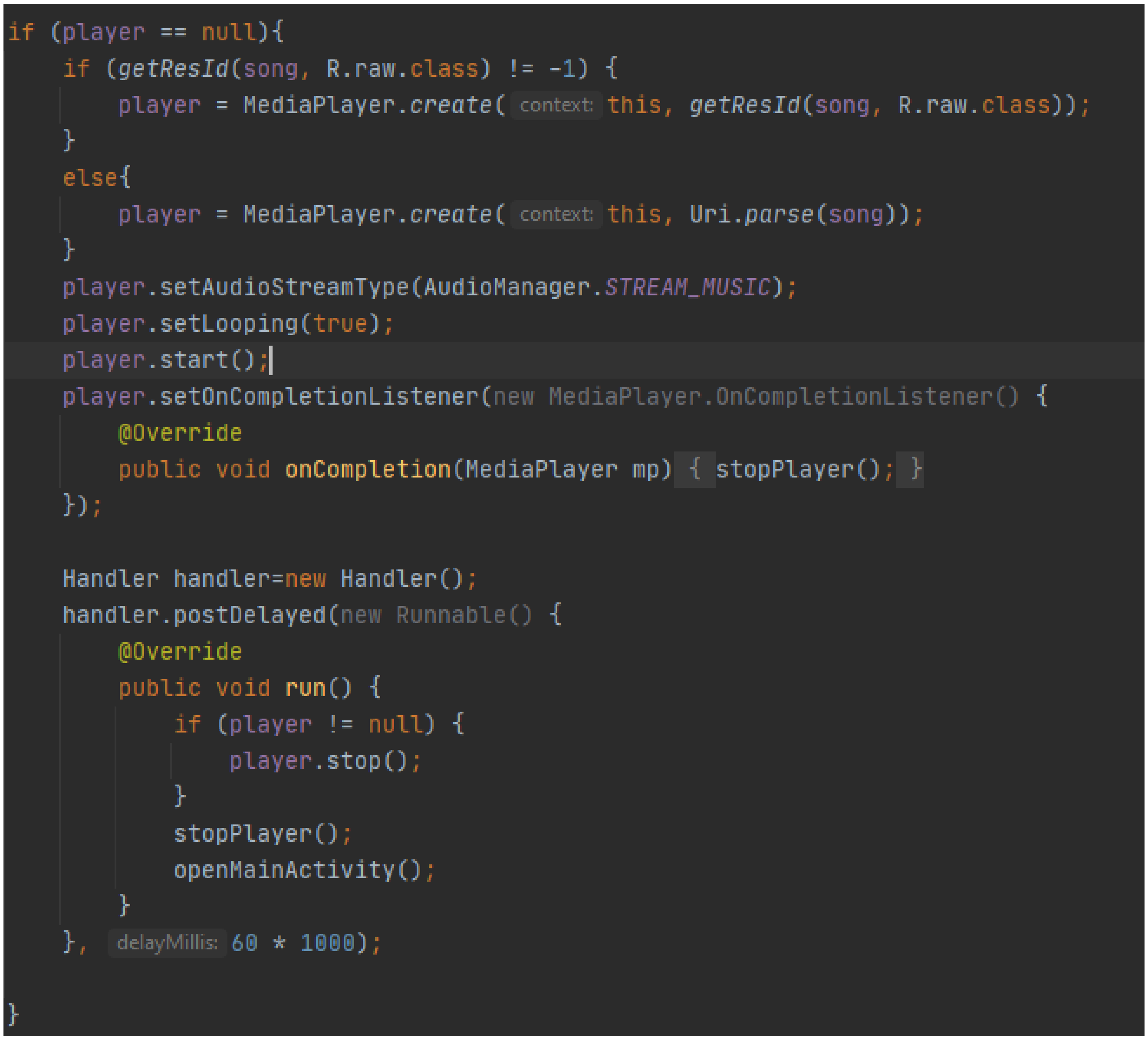
```
db = new DBHelper( context: this);

if (db.getSortedData().size() > 0){
    for (int i=0; i<db.getSortedData().size(); i++){
        if (db.getSortedData().get(i).get(6).equals("on")){
            time = db.getSortedData().get(i).get(0);
            day.append(db.getSortedData().get(i).get(1));
            week = db.getSortedData().get(i).get(2);
            message = db.getSortedData().get(i).get(3);
            song = db.getSortedData().get(i).get(4);
            image = db.getSortedData().get(i).get(5);
            state = db.getSortedData().get(i).get(6);
            color = db.getSortedData().get(i).get(7);
            break;
        }
    }
}
```

4. The function findViewById() is used in several places in order to access and update a view that was defined in the layout file. We use this function to get the layout of the alarm, for which we then change the background to the image obtained from the database. We use the same principle to get the TextView for the message and set the text and the color recorded from the database. Since the message and color are optional, we need to check if they were supplied.



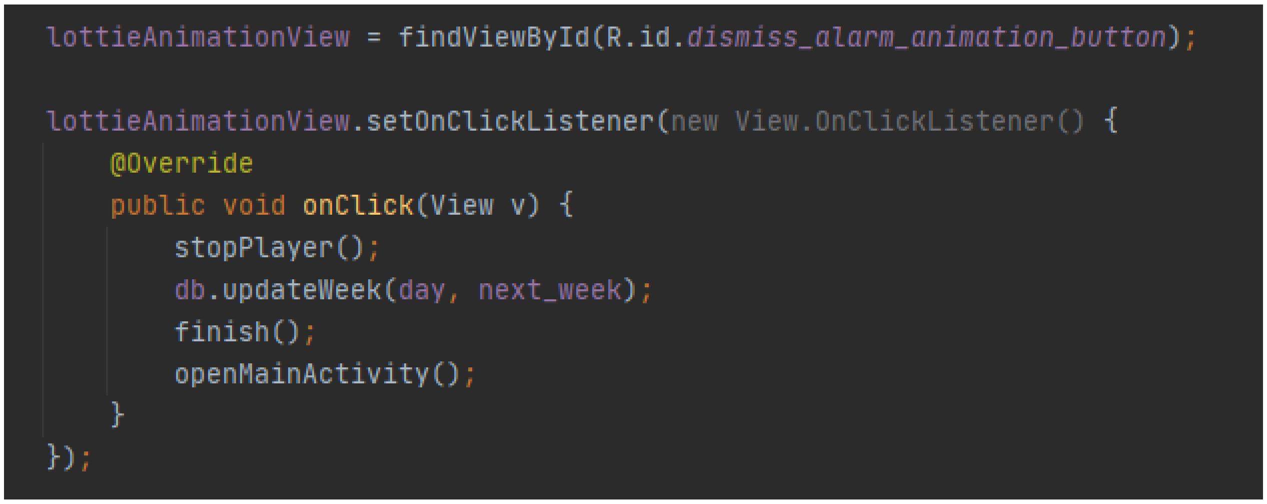
```
RelativeLayout relativeLayout = (RelativeLayout) findViewById(R.id.alarm_layout);

relativeLayout.setBackground(Drawable.createFromPath(image));

next_week = String.valueOf(ConvertIntoNumeric(week) + 1);

TextView messageTextView = (TextView) findViewById(R.id.alarm_text_message);

if (message != null) {
    messageTextView.setText(message);
    if (ConvertIntoNumeric(color) != 0)
        messageTextView.setTextColor(ConvertIntoNumeric(color));
    else
        messageTextView.setTextColor(ContextCompat.getColor( context: AlarmActivity.this, R.color.white));
}
```

5. Next, we need to play the song that was provided.
    - The custom define function getResId() is used in order to check if the song is imported or default. The default songs are stored in the drawable/raw directory.
    - In order to play the song, we use a MediaPlayer called **player**, which we initialize by calling the create() method from the MediaPlayer class, which takes 2 arguments: a context and the resource id or the path for the audio file.
    - After the player was created, we put in on loop, we start it and we call the setOnCompletionListener() function, which stops the player when the song was completed or it stops it after 60 seconds using the Handler class.



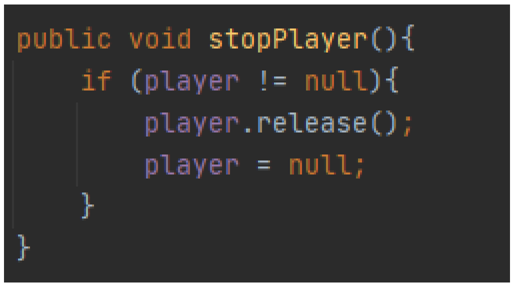
```java
if (player == null){
    if (getResId(song, R.raw.class) != -1) {
        player = MediaPlayer.create( context: this, getResId(song, R.raw.class));
    }
    else{
        player = MediaPlayer.create( context: this, Uri.parse(song));
    }
    player.setAudioStreamType(AudioManager.STREAM_MUSIC);
    player.setLooping(true);
    player.start();
    player.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
        @Override
        public void onCompletion(MediaPlayer mp) { stopPlayer(); }
    });

    Handler handler=new Handler();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            if (player != null) {
                player.stop();
            }
            stopPlayer();
            openMainActivity();
        }
    }, delayMillis: 60 * 1000);

}
```

6. The final step in the onCreate() method is to get the lottieAnimationView and call the setOnclickListener() on it, which will do the following:
    - We first stop the player
    - We increment the week by one and update it in the database, such that the alarm is rescheduled for the next week, on the same day and at the same time.
    - we call finish(), since the user shouldn't be allowed to return to this activity after it has been closed.
    - we call openMainActivity() method which takes the user to the main activity.

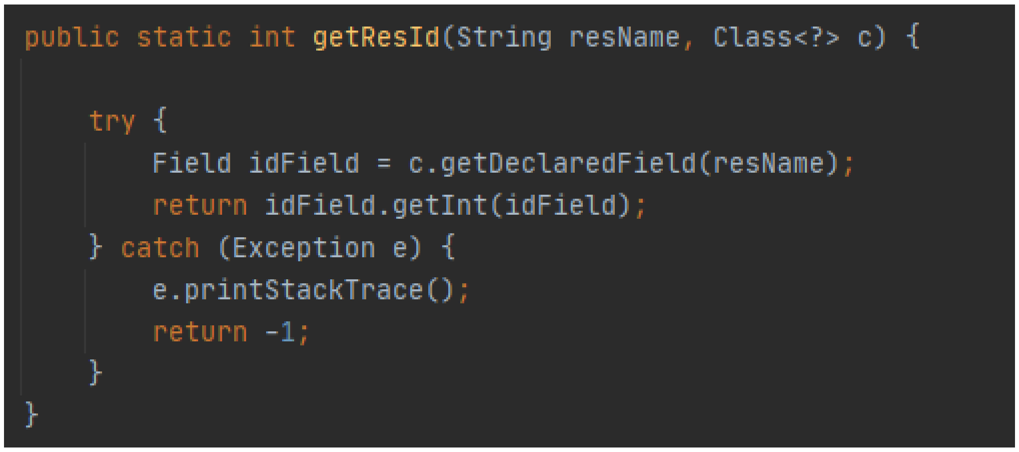

```java
lottieAnimationView = findViewById(R.id.dismiss_alarm_animation_button);

lottieAnimationView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        stopPlayer();
        db.updateWeek(day, next_week);
        finish();
        openMainActivity();
    }
});
```

Other functions included in this activity are:

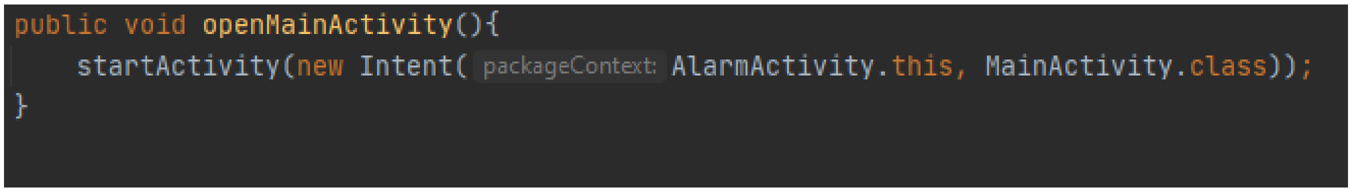
1. stopPlayer(), which releases the MediaPlayer resources and causes the player to stop.


```java
public void stopPlayer(){
    if (player != null){
        player.release();
        player = null;
    }
}
```

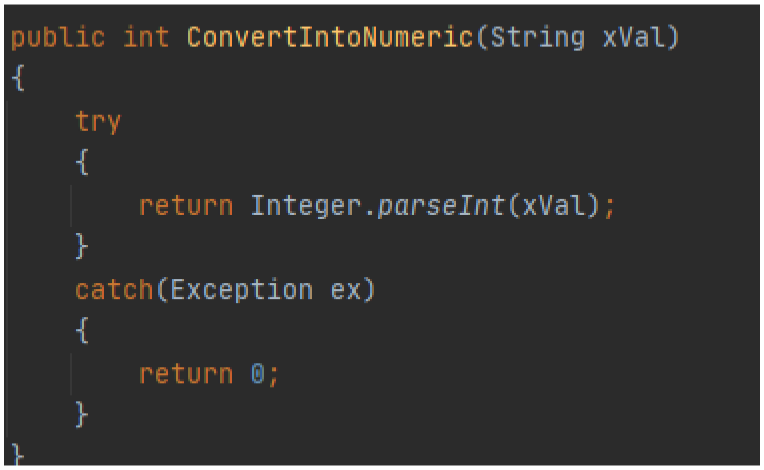
2. getResId(), which takes as arguments a string and a template class and returns the id of the resource associated to the input string if it exists, or -1 otherwise.


```java
public static int getResId(String resName, Class<?> c) {

    try {
        Field idField = c.getDeclaredField(resName);
        return idField.getInt(idField);
    } catch (Exception e) {
        e.printStackTrace();
        return -1;
    }
}
```

3. openMainActivity(), which starts the main activity by calling the startActivity() method on the respective intent.

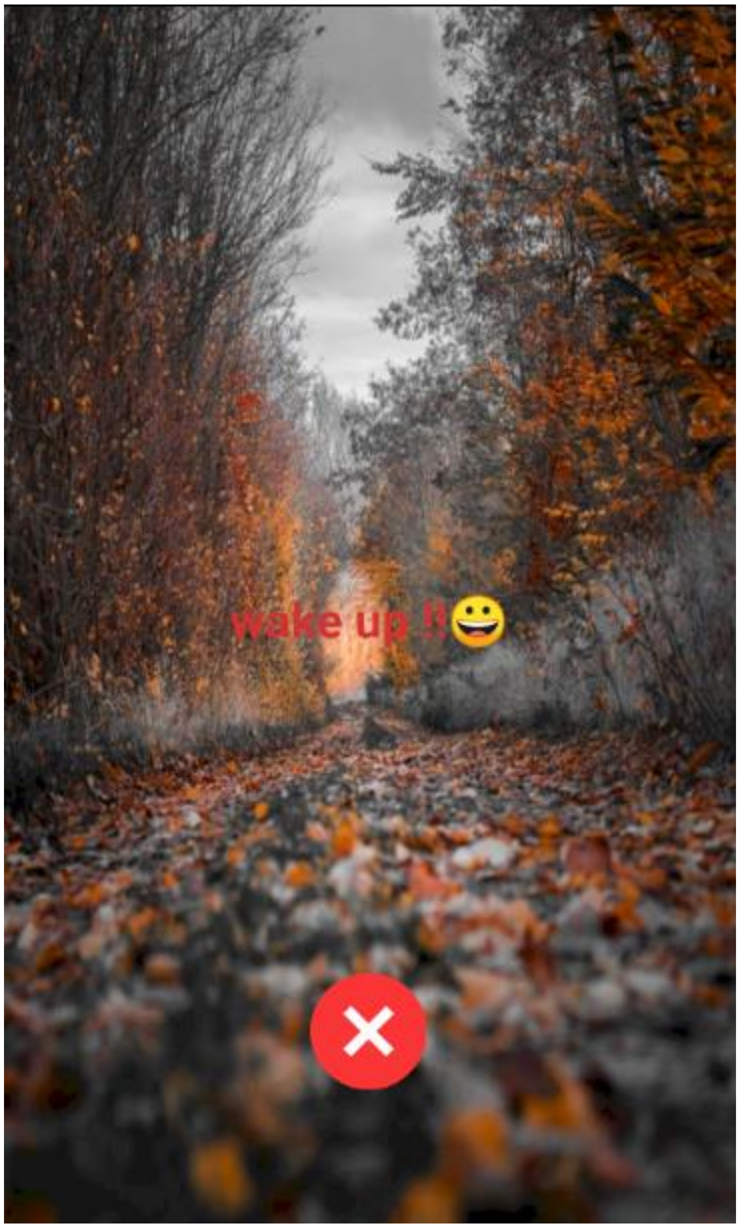

```java
public void openMainActivity(){
    startActivity(new Intent( packageContext: AlarmActivity.this, MainActivity.class));
}
```

4. ConvertIntoNomeric(), which takes a string as input and converts it to int type.

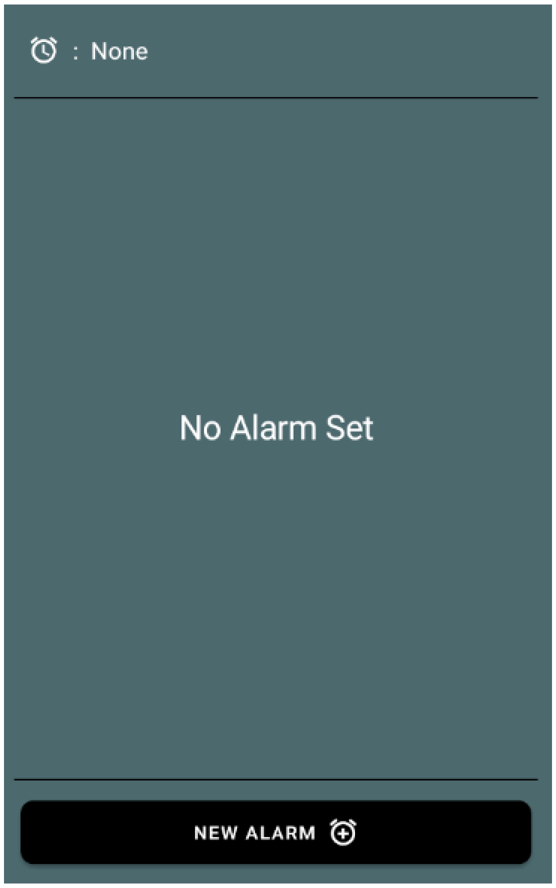


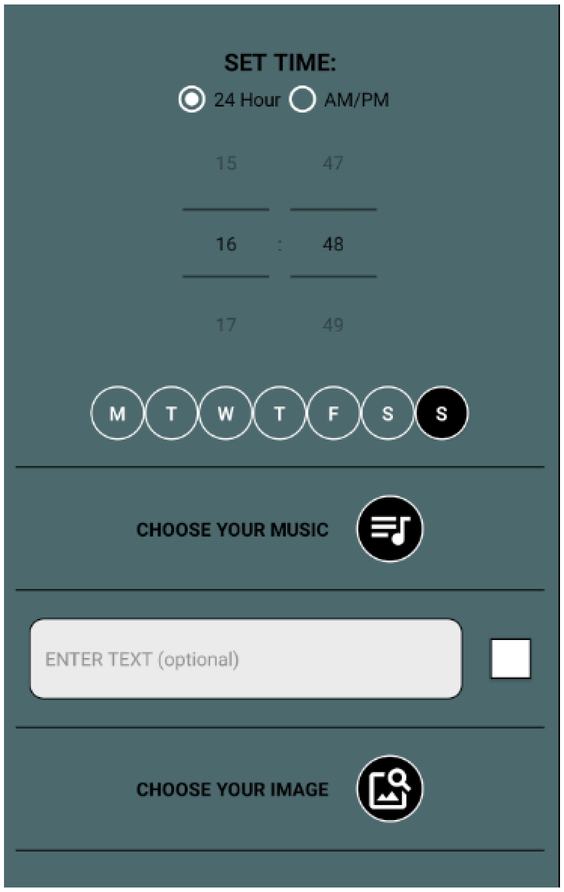
```
public int ConvertIntoNumeric(String xVal)
{
    try
    {
        return Integer.parseInt(xVal);
    }
    catch(Exception ex)
    {
        return 0;
    }
}
```

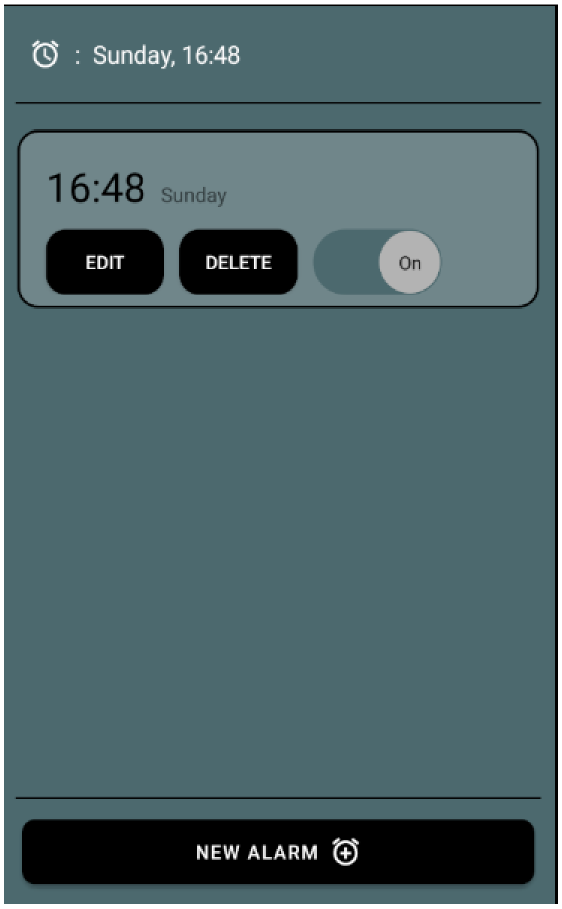
This is the final output of this activity:

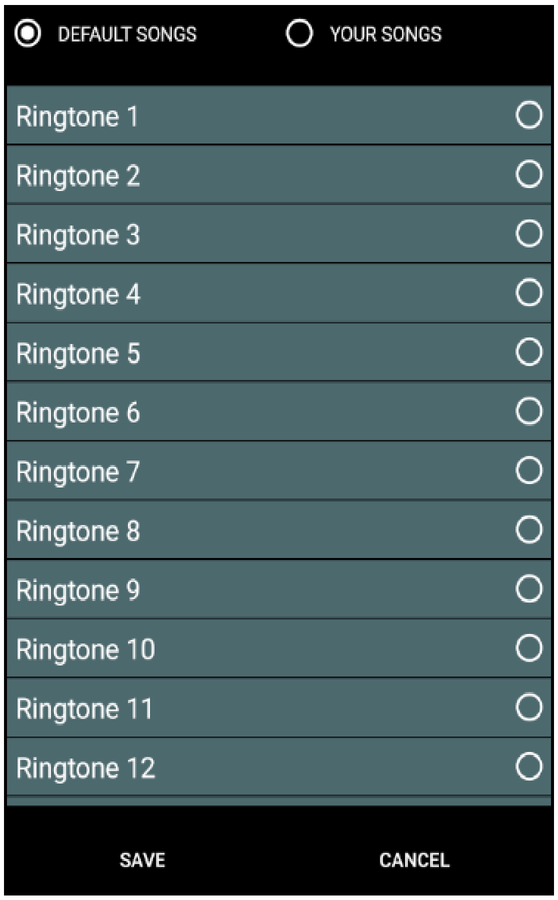


7. App Screenshots









8. Conclusions and Future Work

This project was a pleasant and fun experience which allowed me to increase my knowledge base in android development and I am satisfied with the outcome.

When it comes to future work, I will integrate an image editor to further expand the available customization options.

9. References

External libraries used:

- 'com.github.yukuku:ambilwarna:2.0.1'
- "com.airbnb.android:lottie:3.7.0"

Bibliography:

- https://lottiefiles.com/
- https://abhiandroid.com/ui/dynamic-relativelayout-params-programmatically.html
- https://priyankacool10.wordpress.com/2014/03/26/how-to-pass-data-from-one-activity-to-another-in-android/
- https://abhiandroid.com/ui/timepicker
- https://developer.android.com