LAMPIRAN

Coding AlaramReceiver.java

```
package com.blanyal.remindme;


import android.app.AlarmManager;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import
android.content.pm.PackageManager;
import android.graphics.BitmapFactory;
import android.media.RingtoneManager;
import android.os.SystemClock;
import
android.support.v4.app.NotificationCompat;
import
android.support.v4.content.WakefulBroadca
stReceiver;

import java.util.Calendar;

public class AlarmReceiver extends
WakefulBroadcastReceiver {
    AlarmManager mAlarmManager;
    PendingIntent mPendingIntent;

    @Override
    public void onReceive(Context context,
Intent intent) {
        int mReceivedID =
Integer.parseInt(intent.getStringExtra(Remin
derEditActivity.EXTRA_REMINDER_ID))
;

        // Get notification title from Reminder
Database
        ReminderDatabase rb = new
ReminderDatabase(context);
        Reminder reminder =
rb.getReminder(mReceivedID);
        String mTitle = reminder.getTitle();

        // Create intent to open
ReminderEditActivity on notification click
        Intent editIntent = new Intent(context,
ReminderEditActivity.class);

editIntent.putExtra(ReminderEditActivity.E
XTRA_REMINDER_ID,
Integer.toString(mReceivedID));
        PendingIntent mClick =
PendingIntent.getActivity(context,
mReceivedID, editIntent,
PendingIntent.FLAG_UPDATE_CURREN
T);

        // Create Notification
        NotificationCompat.Builder mBuilder
= new NotificationCompat.Builder(context)

.setLargeIcon(BitmapFactory.decodeResour
ce(context.getResources(),
R.mipmap.ic_launcher))

.setSmallIcon(R.drawable.ic_alarm_on_whit
e_24dp)

.setContentTitle(context.getResources().getS
tring(R.string.app_name))
                .setTicker(mTitle)
                .setContentText(mTitle)

.setSound(RingtoneManager.getDefaultUri(
RingtoneManager.TYPE_NOTIFICATION)
)
                .setContentIntent(mClick)
                .setAutoCancel(true)
                .setOnlyAlertOnce(true);


        NotificationManager nManager =
(NotificationManager)
context.getSystemService(Context.NOTIFI
CATION_SERVICE);
        nManager.notify(mReceivedID,
mBuilder.build());
    }


    public void setAlarm(Context context,
Calendar calendar, int ID) {
        mAlarmManager = (AlarmManager)
context.getSystemService(Context.ALARM
_SERVICE);


        // Put Reminder ID in Intent Extra
        Intent intent = new Intent(context,
AlarmReceiver.class);

intent.putExtra(ReminderEditActivity.EXT
RA_REMINDER_ID, Integer.toString(ID));
        mPendingIntent =
PendingIntent.getBroadcast(context, ID,
intent,
PendingIntent.FLAG_CANCEL_CURREN
T);


        // Calculate notification time
        Calendar c = Calendar.getInstance();
        long currentTime =
c.getTimeInMillis();
        long diffTime =
calendar.getTimeInMillis() - currentTime;


        // Start alarm using notification time

mAlarmManager.set(AlarmManager.ELAPS
ED_REALTIME,
                SystemClock.elapsedRealtime() +
diffTime,
                mPendingIntent);


        // Restart alarm if device is rebooted
        ComponentName receiver = new
ComponentName(context,
BootReceiver.class);
        PackageManager pm =
context.getPackageManager();

pm.setComponentEnabledSetting(receiver,
PackageManager.COMPONENT_ENABLED_STATE_ENABLED,

PackageManager.DONT_KILL_APP);
    }

    public void setRepeatAlarm(Context context, Calendar calendar, int ID, long RepeatTime) {
        mAlarmManager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);

        // Put Reminder ID in Intent Extra
        Intent intent = new Intent(context, AlarmReceiver.class);

intent.putExtra(ReminderEditActivity.EXTRA_REMINDER_ID, Integer.toString(ID));
        mPendingIntent = PendingIntent.getBroadcast(context, ID, intent, PendingIntent.FLAG_CANCEL_CURRENT);

        // Calculate notification timein
        Calendar c = Calendar.getInstance();
        long currentTime = c.getTimeInMillis();
        long diffTime = calendar.getTimeInMillis() - currentTime;

        // Start alarm using initial notification time and repeat interval time

mAlarmManager.setRepeating(AlarmManager.ELAPSED_REALTIME,
                SystemClock.elapsedRealtime() + diffTime,
                RepeatTime , mPendingIntent);

        // Restart alarm if device is rebooted
        ComponentName receiver = new ComponentName(context, BootReceiver.class);
        PackageManager pm = context.getPackageManager();

pm.setComponentEnabledSetting(receiver,

PackageManager.COMPONENT_ENABLED_STATE_ENABLED,

PackageManager.DONT_KILL_APP);
    }

    public void cancelAlarm(Context context, int ID) {
        mAlarmManager = (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);

        // Cancel Alarm using Reminder ID
        mPendingIntent = PendingIntent.getBroadcast(context, ID,
new Intent(context, AlarmReceiver.class),
0);

mAlarmManager.cancel(mPendingIntent);


        // Disable alarm
        ComponentName receiver = new
ComponentName(context,
BootReceiver.class);
        PackageManager pm =
context.getPackageManager();

pm.setComponentEnabledSetting(receiver,

PackageManager.COMPONENT_ENABLE
D_STATE_DISABLED,

PackageManager.DONT_KILL_APP);
    }
}
```

DateRimeSorter.java

```
package com.blanyal.remindme;


// Class to create DateTime objects for easy
sorting
public class DateTimeSorter {
    public int mIndex;
    public String mDateTime;

    public DateTimeSorter(int index, String
DateTime){
        mIndex = index;
        mDateTime = DateTime;
    }

    public DateTimeSorter(){}



    public int getIndex() {
        return mIndex;
    }

    public void setIndex(int index) {
        mIndex = index;
    }

    public String getDateTime() {
        return mDateTime;
    }

    public void setDateTime(String
dateTime) {
        mDateTime = dateTime;
    }
}
```

RemenderDatabase.java

```
package com.blanyal.remindme;


import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import
android.database.sqlite.SQLiteDatabase;

import
android.database.sqlite.SQLiteOpenHelper;


import java.util.ArrayList;

import java.util.List;



public class ReminderDatabase extends
SQLiteOpenHelper {

   // Database Version

   private static final int
DATABASE_VERSION = 1;


   // Database Name

   private static final String
DATABASE_NAME =
"ReminderDatabase";


   // Table name

   private static final String
TABLE_REMINDERS = "ReminderTable";


   // Table Columns names

   private static final String KEY_ID = "id";

   private static final String KEY_TITLE =
"title";

   private static final String KEY_DATE =
"date";

   private static final String KEY_TIME =
"time";

   private static final String KEY_REPEAT
= "repeat";

   private static final String
KEY_REPEAT_NO = "repeat_no";

   private static final String
KEY_REPEAT_TYPE = "repeat_type";

   private static final String KEY_ACTIVE
= "active";


   public ReminderDatabase(Context
context) {

       super(context, DATABASE_NAME,
null, DATABASE_VERSION);

   }


   // Creating Tables

   @Override

   public void onCreate(SQLiteDatabase db)
{

       String
CREATE_REMINDERS_TABLE =
"CREATE TABLE " +
TABLE_REMINDERS +

               "("

               + KEY_ID + " INTEGER
PRIMARY KEY,"

               + KEY_TITLE + " TEXT,"
```

```
            + KEY_DATE + " TEXT,"
            + KEY_TIME + " INTEGER,"
            + KEY_REPEAT + "
BOOLEAN,"
            + KEY_REPEAT_NO + "
INTEGER,"
            + KEY_REPEAT_TYPE + "
TEXT,"
            + KEY_ACTIVE + " BOOLEAN"
+ ")";

db.execSQL(CREATE_REMINDERS_TAB
LE);
   }


   // Upgrading database
   @Override
   public void onUpgrade(SQLiteDatabase
db, int oldVersion, int newVersion) {
      // Drop older table if existed
      if (oldVersion >= newVersion)
         return;
      db.execSQL("DROP TABLE IF
EXISTS " + TABLE_REMINDERS);


      // Create tables again
      onCreate(db);
   }


   // Adding new Reminder
   public int addReminder(Reminder
reminder){
      SQLiteDatabase db =
this.getWritableDatabase();
      ContentValues values = new
ContentValues();


      values.put(KEY_TITLE ,
reminder.getTitle());
      values.put(KEY_DATE ,
reminder.getDate());
      values.put(KEY_TIME ,
reminder.getTime());
      values.put(KEY_REPEAT ,
reminder.getRepeat());
      values.put(KEY_REPEAT_NO ,
reminder.getRepeatNo());
      values.put(KEY_REPEAT_TYPE,
reminder.getRepeatType());
      values.put(KEY_ACTIVE,
reminder.getActive());


      // Inserting Row
      long ID =
db.insert(TABLE_REMINDERS, null,
values);
      db.close();
      return (int) ID;
   }


   // Getting single Reminder
   public Reminder getReminder(int id){
        SQLiteDatabase db =
this.getReadableDatabase();


        Cursor cursor =
db.query(TABLE_REMINDERS, new
String[]
                        {
                                KEY_ID,
                                KEY_TITLE,
                                KEY_DATE,
                                KEY_TIME,
                                KEY_REPEAT,
                                KEY_REPEAT_NO,
                                KEY_REPEAT_TYPE,
                                KEY_ACTIVE
                        }, KEY_ID + "=?",


                new String[] {String.valueOf(id)},
null, null, null, null);


        if (cursor != null)
            cursor.moveToFirst();


        Reminder reminder = new
Reminder(Integer.parseInt(cursor.getString(
0)), cursor.getString(1),
                cursor.getString(2),
cursor.getString(3), cursor.getString(4),
                cursor.getString(5),
cursor.getString(6), cursor.getString(7));
        return reminder;
    }


    // Getting all Reminders
    public List<Reminder>
getAllReminders(){
        List<Reminder> reminderList = new
ArrayList<>();


        // Select all Query
        String selectQuery = "SELECT *
FROM " + TABLE_REMINDERS;


        SQLiteDatabase db =
this.getWritableDatabase();
        Cursor cursor =
db.rawQuery(selectQuery, null);


        // Looping through all rows and adding
to list
        if(cursor.moveToFirst()){
            do{
                Reminder reminder = new
Reminder();

reminder.setID(Integer.parseInt(cursor.getSt
ring(0)));

reminder.setTitle(cursor.getString(1));

reminder.setDate(cursor.getString(2));
reminder.setTime(cursor.getString(3));

reminder.setRepeat(cursor.getString(4));

reminder.setRepeatNo(cursor.getString(5));

reminder.setRepeatType(cursor.getString(6)
);

reminder.setActive(cursor.getString(7));


            // Adding Reminders to list
            reminderList.add(reminder);
        } while (cursor.moveToNext());
    }
    return reminderList;
  }


  // Getting Reminders Count
  public int getRemindersCount(){
    String countQuery = "SELECT *
FROM " + TABLE_REMINDERS;
    SQLiteDatabase db =
this.getReadableDatabase();
    Cursor cursor =
db.rawQuery(countQuery,null);
    cursor.close();


    return cursor.getCount();
  }

  // Updating single Reminder
  public int updateReminder(Reminder
reminder){
    SQLiteDatabase db =
this.getWritableDatabase();
    ContentValues values = new
ContentValues();
    values.put(KEY_TITLE ,
reminder.getTitle());
    values.put(KEY_DATE ,
reminder.getDate());
    values.put(KEY_TIME ,
reminder.getTime());
    values.put(KEY_REPEAT ,
reminder.getRepeat());
    values.put(KEY_REPEAT_NO ,
reminder.getRepeatNo());
    values.put(KEY_REPEAT_TYPE,
reminder.getRepeatType());
    values.put(KEY_ACTIVE,
reminder.getActive());


    // Updating row
    return
db.update(TABLE_REMINDERS, values,
KEY_ID + "=?",
        new
String[]{String.valueOf(reminder.getID())})
;
  }


  // Deleting single Reminder
```

```
    public void deleteReminder(Reminder
reminder){
        SQLiteDatabase db =
this.getWritableDatabase();
        db.delete(TABLE_REMINDERS,
KEY_ID + "=?",
                new
String[]{String.valueOf(reminder.getID())})
;
        db.close();
    }
}
```

Roundrobbin.java

```
package com.blanyal.remindme;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

/**
 * Created by tegar fauzi on 2/22/2018.
 */

class process
{
    int no;
    int bt;
    int wt;
    int tt;
    process()
    {
        bt=0;wt=0;tt=0;
    }
}
class RR
{
    public static void main(String
args[])throws IOException
    {
        InputStreamReader isr = new
InputStreamReader(System.in);
        BufferedReader stdin=new
BufferedReader(isr);
        int i,n,t,w=0,j,at,count;
        System.out.print("Masukan Jumlah
Proses ");
        n=Integer.parseInt(stdin.readLine());
        process p[]=new process[n+1];
        p[0]=new process();
        for(i=1;i<n+1;i++)
        {
            p[i]=new process();
            p[i].no=i;
            System.out.println("process
"+p[i].no);
            System.out.print("masukan burst
time : ");

p[i].bt=Integer.parseInt(stdin.readLine());
```

```
        }
        System.out.print("masukan time
quantum : ");
        t=Integer.parseInt(stdin.readLine());
        i=1;
        while(true)
        {
            count=0;
            p[i].wt=w;
            if(p[i].bt>=t)
            {
                at=t;
                p[i].bt=p[i].bt-t;
            }
            else
            {
                at=p[i].bt;
                p[i].bt=0;
            }
            w=w+at;
            for(j=1;j<n+1;j++)
            {
                if(p[i].bt==0)
                {
                    count=count + 1;
                }
            }
            if(count==n)
            {
                break;
            }
            i++;
            if(i==(n+1))
                i=1;
        }
        System.out.println("Waktu Tunggu");
        for(i=1;i<n+1;i++)
        {
            System.out.println("process
"+p[i].no+" : "+p[i].wt);
        }
    }
}
```

Main2Activty.java

```
package com.blanyal.remindme;

import android.content.Intent;
import
android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
```

```
public class Main2Activity extends
AppCompatActivity {

   Button button1, reset;

   private EditText kegiatan1, kegiatan2,
kegiatan3,bt1,bt2,bt3;

   TextView output;

   String k1,k2,k3;


   @Override

   protected void onCreate(Bundle
savedInstanceState) {

      super.onCreate(savedInstanceState);


setContentView(R.layout.activity_main2);


      kegiatan1 = (EditText)
findViewById(R.id.kegiatan1);

      kegiatan2 = (EditText)
findViewById(R.id.kegiatan2);

      kegiatan3 = (EditText)
findViewById(R.id.kegiatan3);

      bt1=(EditText)
findViewById(R.id.brustime1);

      bt2=(EditText)
findViewById(R.id.brustime2);

      bt3=(EditText)
findViewById(R.id.brustime3);

      reset = (Button)
findViewById(R.id.hapus);

      button1 = (Button)
findViewById(R.id.hitung);

      output=(TextView)
findViewById(R.id.hasil);

      reset.setOnClickListener(new
View.OnClickListener() {

         @Override

         public void onClick(View v) {

            kegiatan1.setText("");

            kegiatan2.setText("");

            kegiatan3.setText("");




         }

      });

      button1.setOnClickListener(new
View.OnClickListener()


      {

         @Override

         public void onClick(View v) {


            k1 =
kegiatan1.getText().toString();

            k2 =
kegiatan1.getText().toString();

            k3 =
kegiatan1.getText().toString();

            Intent i = null;

            output.setText(k1);

            output.setText(k2);

            output.setText(k3);

         }
```

```
        });
    }
}
```

AddActivity.java

```
package com.blanyal.remindme;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import
android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.text.Editable;
import android.text.InputType;
import android.text.TextWatcher;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import
com.getbase.floatingactionbutton.FloatingA
ctionButton;
import
com.wdullaer.materialdatetimepicker.date.D
atePickerDialog;
import
com.wdullaer.materialdatetimepicker.time.R
adialPickerLayout;
import
com.wdullaer.materialdatetimepicker.time.T
imePickerDialog;

import java.util.Calendar;


public class ReminderAddActivity extends
AppCompatActivity implements
        TimePickerDialog.OnTimeSetListener,
        DatePickerDialog.OnDateSetListener{

    private Toolbar mToolbar;
    private EditText mTitleText;
    private TextView mDateText,
mTimeText, mRepeatText, mRepeatNoText,
mRepeatTypeText;
    private FloatingActionButton mFAB1;
    private FloatingActionButton mFAB2;
    private Calendar mCalendar;
    private int mYear, mMonth, mHour,
mMinute, mDay;
    private long mRepeatTime;
    private String mTitle;
    private String mTime;
    private String mDate;
    private String mRepeat;

    private String mRepeatNo;

    private String mRepeatType;

    private String mActive;


    // Values for orientation change

    private static final String KEY_TITLE =
"title_key";

    private static final String KEY_TIME =
"time_key";

    private static final String KEY_DATE =
"date_key";

    private static final String KEY_REPEAT
= "repeat_key";

    private static final String
KEY_REPEAT_NO = "repeat_no_key";

    private static final String
KEY_REPEAT_TYPE =
"repeat_type_key";

    private static final String KEY_ACTIVE
= "active_key";


    // Constant values in milliseconds

    private static final long milMinute =
60000L;

    private static final long milHour =
3600000L;

    private static final long milDay =
86400000L;

    private static final long milWeek =
604800000L;

    private static final long milMonth =
2592000000L;


    @Override

    protected void onCreate(Bundle
savedInstanceState) {

        super.onCreate(savedInstanceState);


setContentView(R.layout.activity_add_remi
nder);


        // Initialize Views

        mToolbar = (Toolbar)
findViewById(R.id.toolbar);

        mTitleText = (EditText)
findViewById(R.id.reminder_title);

        mDateText = (TextView)
findViewById(R.id.set_date);

        mTimeText = (TextView)
findViewById(R.id.set_time);

        mRepeatText = (TextView)
findViewById(R.id.set_repeat);

        mRepeatNoText = (TextView)
findViewById(R.id.set_repeat_no);

        mRepeatTypeText = (TextView)
findViewById(R.id.set_repeat_type);

        mFAB1 = (FloatingActionButton)
findViewById(R.id.starred1);

        mFAB2 = (FloatingActionButton)
findViewById(R.id.starred2);


        // Setup Toolbar

        setSupportActionBar(mToolbar);
```

```
getSupportActionBar().setTitle(R.string.title
_activity_add_reminder);

getSupportActionBar().setDisplayHomeAsU
pEnabled(true);

getSupportActionBar().setHomeButtonEnab
led(true);

        // Initialize default values
        mActive = "true";
        mRepeat = "true";
        mRepeatNo = Integer.toString(1);
        mRepeatType = "Hour";

        mCalendar = Calendar.getInstance();
        mHour =
mCalendar.get(Calendar.HOUR_OF_DAY);
        mMinute =
mCalendar.get(Calendar.MINUTE);
        mYear =
mCalendar.get(Calendar.YEAR);
        mMonth =
mCalendar.get(Calendar.MONTH) + 1;
        mDay =
mCalendar.get(Calendar.DATE);

        mDate = mDay + "/" + mMonth + "/" +
mYear;
        mTime = mHour + ":" + mMinute;

        // Setup Reminder Title EditText

mTitleText.addTextChangedListener(new
TextWatcher() {

            @Override
            public void
beforeTextChanged(CharSequence s, int
start, int count, int after) {}

            @Override
            public void
onTextChanged(CharSequence s, int start,
int before, int count) {
                mTitle = s.toString().trim();
                mTitleText.setError(null);
            }

            @Override
            public void
afterTextChanged(Editable s) {}
        });

        // Setup TextViews using reminder
values
        mDateText.setText(mDate);
        mTimeText.setText(mTime);
        mRepeatNoText.setText(mRepeatNo);

mRepeatTypeText.setText(mRepeatType);
        mRepeatText.setText("Every " +
mRepeatNo + " " + mRepeatType + "(s)");
```

```
        // To save state on device rotation
        if (savedInstanceState != null) {
            String savedTitle = 
savedInstanceState.getString(KEY_TITLE);
            mTitleText.setText(savedTitle);
            mTitle = savedTitle;

            String savedTime = 
savedInstanceState.getString(KEY_TIME);
            mTimeText.setText(savedTime);
            mTime = savedTime;

            String savedDate = 
savedInstanceState.getString(KEY_DATE);
            mDateText.setText(savedDate);
            mDate = savedDate;

            String saveRepeat = 
savedInstanceState.getString(KEY_REPEA
T);
            mRepeatText.setText(saveRepeat);
            mRepeat = saveRepeat;

            String savedRepeatNo = 
savedInstanceState.getString(KEY_REPEA
T_NO);

mRepeatNoText.setText(savedRepeatNo);
            mRepeatNo = savedRepeatNo;

            String savedRepeatType = 
savedInstanceState.getString(KEY_REPEA
T_TYPE);

mRepeatTypeText.setText(savedRepeatTyp
e);
            mRepeatType = savedRepeatType;

            mActive = 
savedInstanceState.getString(KEY_ACTIV
E);
        }

        // Setup up active buttons
        if (mActive.equals("false")) {

mFAB1.setVisibility(View.VISIBLE);
            mFAB2.setVisibility(View.GONE);

        } else if (mActive.equals("true")) {
            mFAB1.setVisibility(View.GONE);

mFAB2.setVisibility(View.VISIBLE);
        }
    }

    // To save state on device rotation
    @Override
    protected void onSaveInstanceState
(Bundle outState) {
        super.onSaveInstanceState(outState);
```

```
outState.putCharSequence(KEY_TITLE,
mTitleText.getText());

outState.putCharSequence(KEY_TIME,
mTimeText.getText());

outState.putCharSequence(KEY_DATE,
mDateText.getText());

outState.putCharSequence(KEY_REPEAT,
mRepeatText.getText());

outState.putCharSequence(KEY_REPEAT_
NO, mRepeatNoText.getText());

outState.putCharSequence(KEY_REPEAT_
TYPE, mRepeatTypeText.getText());

outState.putCharSequence(KEY_ACTIVE,
mActive);
    }

    // On clicking Time picker
    public void setTime(View v){
        Calendar now =
Calendar.getInstance();
        TimePickerDialog tpd =
TimePickerDialog.newInstance(
                this,

now.get(Calendar.HOUR_OF_DAY),
                now.get(Calendar.MINUTE),
                false
        );
        tpd.setThemeDark(false);
        tpd.show(getFragmentManager(),
"Timepickerdialog");
    }

    // On clicking Date picker
    public void setDate(View v){
        Calendar now =
Calendar.getInstance();
        DatePickerDialog dpd =
DatePickerDialog.newInstance(
                this,
                now.get(Calendar.YEAR),
                now.get(Calendar.MONTH),

now.get(Calendar.DAY_OF_MONTH)
        );
        dpd.show(getFragmentManager(),
"Datepickerdialog");
    }

    // Obtain time from time picker
    @Override
    public void
onTimeSet(RadialPickerLayout view, int
hourOfDay, int minute) {
        mHour = hourOfDay;
        mMinute = minute;
        if (minute < 10) {
            mTime = hourOfDay + ":" + "0" +
minute;
        } else {
            mTime = hourOfDay + ":" + minute;
        }
        mTimeText.setText(mTime);
    }

    // Obtain date from date picker
    @Override
    public void onDateSet(DatePickerDialog
view, int year, int monthOfYear, int
dayOfMonth) {
        monthOfYear ++;
        mDay = dayOfMonth;
        mMonth = monthOfYear;
        mYear = year;
        mDate = dayOfMonth + "/" +
monthOfYear + "/" + year;
        mDateText.setText(mDate);
    }

    // On clicking the active button
    public void selectFab1(View v) {
        mFAB1 = (FloatingActionButton)
findViewById(R.id.starred1);
        mFAB1.setVisibility(View.GONE);
        mFAB2 = (FloatingActionButton)
findViewById(R.id.starred2);
        mFAB2.setVisibility(View.VISIBLE);
        mActive = "true";
    }

    // On clicking the inactive button
    public void selectFab2(View v) {
        mFAB2 = (FloatingActionButton)
findViewById(R.id.starred2);
        mFAB2.setVisibility(View.GONE);
        mFAB1 = (FloatingActionButton)
findViewById(R.id.starred1);
        mFAB1.setVisibility(View.VISIBLE);
        mActive = "false";
    }

    // On clicking the repeat switch
    public void onSwitchRepeat(View view)
{
        boolean on = ((Switch)
view).isChecked();
        if (on) {
            mRepeat = "true";
            mRepeatText.setText("Every " +
mRepeatNo + " " + mRepeatType + "(s)");
        } else {
            mRepeat = "false";

mRepeatText.setText(R.string.repeat_off);
        }
    }
    // On clicking repeat type button
    public void selectRepeatType(View v){
        final String[] items = new String[5];

        items[0] = "Minute";
        items[1] = "Hour";
        items[2] = "Day";
        items[3] = "Week";
        items[4] = "Month";

        // Create List Dialog
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Select Type");
        builder.setItems(items, new DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int item) {

                mRepeatType = items[item];
                mRepeatTypeText.setText(mRepeatType);
                mRepeatText.setText("Every " + mRepeatNo + " " + mRepeatType + "(s)");
            }
        });
        AlertDialog alert = builder.create();
        alert.show();
    }

    // On clicking repeat interval button
    public void setRepeatNo(View v){
        AlertDialog.Builder alert = new AlertDialog.Builder(this);
        alert.setTitle("Enter Number");

        // Create EditText box to input repeat number
        final EditText input = new EditText(this);
        input.setInputType(InputType.TYPE_CLASS_NUMBER);
        alert.setView(input);
        alert.setPositiveButton("Ok",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int whichButton) {

                        if (input.getText().toString().length() == 0) {
                            mRepeatNo = Integer.toString(1);
                            mRepeatNoText.setText(mRepeatNo);
                            mRepeatText.setText("Every " + mRepeatNo + " " + mRepeatType + "(s)");
                        }
                        else {
```

```
                    mRepeatNo =
input.getText().toString().trim();

mRepeatNoText.setText(mRepeatNo);

mRepeatText.setText("Every " +
mRepeatNo + " " + mRepeatType + "(s)");
                }
            }
        });
        alert.setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface
dialog, int whichButton) {
                // do nothing
            }
        });
        alert.show();
    }

    // On clicking the save button
    public void saveReminder(){
        ReminderDatabase rb = new
ReminderDatabase(this);

        // Creating Reminder
        int ID = rb.addReminder(new
Reminder(mTitle, mDate, mTime, mRepeat,
mRepeatNo, mRepeatType, mActive));

        // Set up calender for creating the
notification
        mCalendar.set(Calendar.MONTH, --
mMonth);
        mCalendar.set(Calendar.YEAR,
mYear);

mCalendar.set(Calendar.DAY_OF_MONT
H, mDay);

mCalendar.set(Calendar.HOUR_OF_DAY,
mHour);
        mCalendar.set(Calendar.MINUTE,
mMinute);
        mCalendar.set(Calendar.SECOND, 0);

        // Check repeat type
        if (mRepeatType.equals("Minute")) {
            mRepeatTime =
Integer.parseInt(mRepeatNo) * milMinute;
        } else if
(mRepeatType.equals("Hour")) {
            mRepeatTime =
Integer.parseInt(mRepeatNo) * milHour;
        } else if (mRepeatType.equals("Day"))
{
            mRepeatTime =
Integer.parseInt(mRepeatNo) * milDay;
        } else if
(mRepeatType.equals("Week")) {
            mRepeatTime =
Integer.parseInt(mRepeatNo) * milWeek;
        } else if
(mRepeatType.equals("Month")) {
            mRepeatTime =
Integer.parseInt(mRepeatNo) * milMonth;
```

```
        }


        // Create a new notification
        if (mActive.equals("true")) {
            if (mRepeat.equals("true")) {
                new
AlarmReceiver().setRepeatAlarm(getApplic
ationContext(), mCalendar, ID,
mRepeatTime);
            } else if (mRepeat.equals("false")) {
                new
AlarmReceiver().setAlarm(getApplicationC
ontext(), mCalendar, ID);
            }
        }


        // Create toast to confirm new reminder

Toast.makeText(getApplicationContext(),
"Saved",
                Toast.LENGTH_SHORT).show();


        onBackPressed();
    }


    // On pressing the back button
    @Override
    public void onBackPressed() {
        super.onBackPressed();
    }
```

```
    // Creating the menu
    @Override
    public boolean
onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.menu_add
_reminder, menu);
        return true;
    }


    // On clicking menu buttons
    @Override
    public boolean
onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {


            // On clicking the back arrow
            // Discard any changes
            case android.R.id.home:
                onBackPressed();
                return true;


            // On clicking save reminder button
            // Update reminder
            case R.id.save_reminder:
                mTitleText.setText(mTitle);


                if
(mTitleText.getText().toString().length() ==
0)
```

```
                    mTitleText.setError("Reminder
Title cannot be blank!");


                else {
                    saveReminder();
                }
                return true;


            // On clicking discard reminder
button
            // Discard any changes
            case R.id.discard_reminder:

Toast.makeText(getApplicationContext(),
"Discarded",

Toast.LENGTH_SHORT).show();


                onBackPressed();
                return true;


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