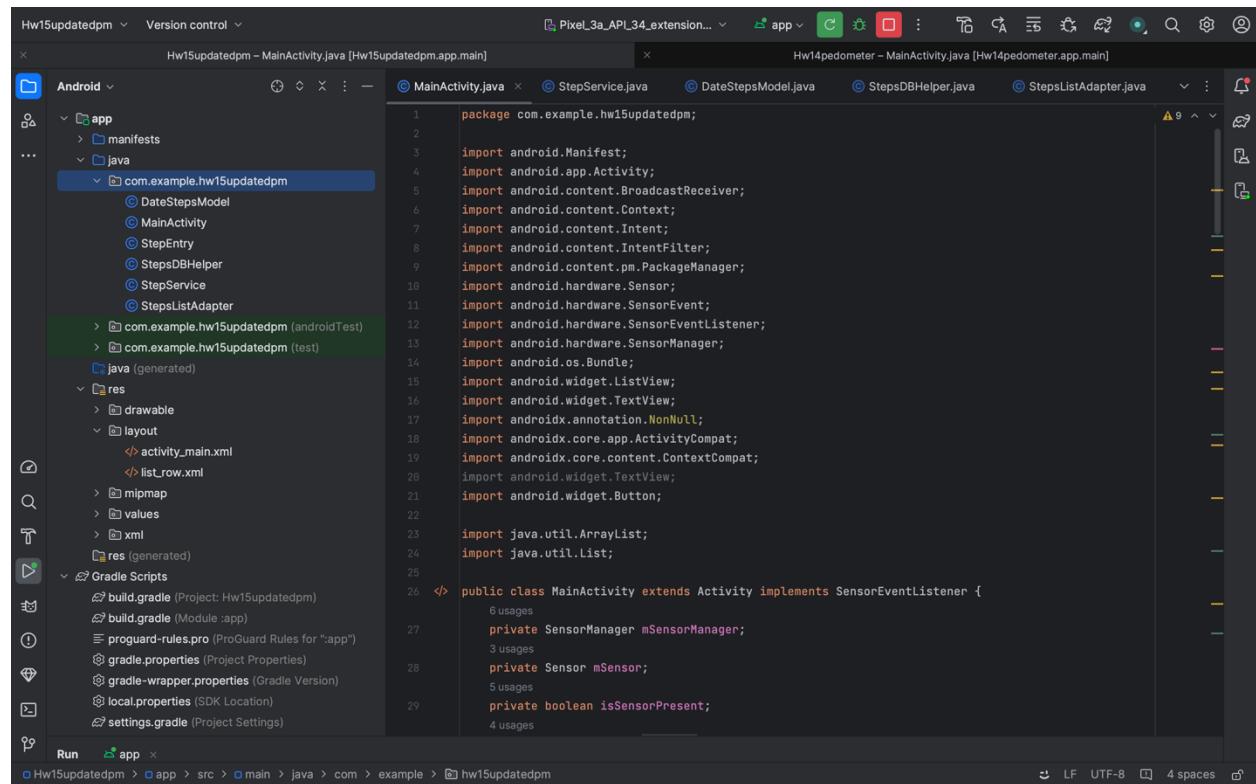


The improved pedometer app with the custom activity detection algorithm has a design issue: the step data are pulled from the SQLite database only at the start of the app. So, if you have run the app for a few minutes with multiple steps, the display the main activity won't change at all. To see the data collected, you would have to close the app and restart it.

To address this issue, modify the app so that whenever a new batch of steps are recorded, the display on the main activity would be updated. Alternatively, you may add a refresh button on the main activity layout to retrieve the data and display it.

The app actually has another issue: the step count is not as accurate as the step counter. Do some experiments (you will need to carry your phone and walk for say, 100 steps, and compare what is recorded in your app with respect to the actual steps you made) and tune the 3 thresholds used for step counting so that they would work for you – meaning the step count is almost as accurately as the actual number of steps.

Mainactivity. Java : The 'MainActivity.java' code provided is for an Android step counting app. It loads UI components, such as TextViews and a reset button, and checks for the step counter sensor's availability. The app seeks the necessary permission for activity recognition and launches a background service 'StepService' to track steps. It registers a broadcast receiver to receive step count information from the service and update the UI. The 'onSensorChanged' method updates the step count in real time and saves new step entries in a database. Overall, this code generates a working step counter app that includes sensor integration, database administration, and user interface updates.



The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar shows the project structure for "Hw15updatedpm". It includes the "app" module with "src" containing "main", "java", and "res" directories. The "java" directory contains packages for "com.example.hw15updatedpm" and "com.example.hw15updatedpm (test)".
- Code Editor:** The right pane displays the "MainActivity.java" code. The code imports various Android classes and interfaces, including `SensorManager`, `SensorEvent`, and `SensorEventListener`. It defines a class `MainActivity` that extends `Activity` and implements `SensorEventListener`. The class contains methods for handling sensor events and managing step data.
- Toolbars and Status Bar:** The top bar shows tabs for "Pixel_3a_API_34.extension...", "app", and "MainActivity.java". The bottom status bar shows file type indicators (LF), encoding (UTF-8), and code style settings (4 spaces).

```

Hw15updatedpm - MainActivity.java [Hw15updatedpm.app.main]
Hw14pedometer - MainActivity.java [Hw14pedometer.app.main]

Android
  app
    manifests
    java
      com.example.hw15updatedpm
        DateStepsModel
        MainActivity
        StepEntry
        StepsDBHelper
        StepService
        StepsListAdapter
      com.example.hw15updatedpm (androidTest)
      com.example.hw15updatedpm (test)
    java (generated)
  res
    drawable
    layout
      activity_main.xml
      list_row.xml
    mipmap
    values
    xml
    res (generated)
  Gradle Scripts
    build.gradle (Project: Hw15updatedpm)
    build.gradle (Module: app)
    proguard-rules.pro (ProGuard Rules for ":app")
    gradle.properties (Project Properties)
    gradle-wrapper.properties (Gradle Version)
    local.properties (SDK Location)
    settings.gradle (Project Settings)

Run app x

Hw15updatedpm > app > src > main > java > com > example > hw15updatedpm

```

```

1 package com.example.hw15updatedpm;
2
3 import android.Manifest;
4 import android.app.Activity;
5 import android.content.BroadcastReceiver;
6 import android.content.Context;
7 import android.content.Intent;
8 import android.content.IntentFilter;
9 import android.content.pm.PackageManager;
10 import android.hardware.Sensor;
11 import android.hardware.SensorEvent;
12 import android.hardware.SensorEventListener;
13 import android.hardware.SensorManager;
14 import android.os.Bundle;
15 import android.widget.ListView;
16 import android.widget.TextView;
17 import androidx.annotation.NonNull;
18 import androidx.core.app.ActivityCompat;
19 import androidx.core.content.ContextCompat;
20 import android.widget.TextView;
21 import android.widget.Button;
22
23 import java.util.ArrayList;
24 import java.util.List;
25
26 //>> public class MainActivity extends Activity implements SensorEventListener {
27   6 usages
28     private SensorManager mSensorManager;
29   3 usages
30     private Sensor mSensor;
31   5 usages
32     private boolean isSensorPresent;
33   4 usages

```

Hw15updatedpm - Version control

MainActivity.java [Hw15updatedpm.app.main]

```

public class MainActivity extends Activity implements SensorEventListener {
    ...
    private SensorManager mSensorManager;
    private Sensor mSensor;
    private boolean isSensorPresent;
    private StepsDBHelper dbHelper;
    private BroadcastReceiver stepUpdateReceiver;
    private ListView listViewSteps;
    private StepsListAdapter adapter;
    private TextView mStepsSinceReboot;
    private Button resetButton;
    private int initialStepCount = 1;
    private int lastStepCount = 0;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ...
    }
}

```

Run app

Hw15updatedpm - Version control

MainActivity.java [Hw15updatedpm.app.main]

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // Initializing components
    mStepsSinceReboot = findViewById(R.id.stepssincereboot);
    listViewSteps = findViewById(R.id.listViewSteps);
    resetButton = findViewById(R.id.resetButton);
    stepEntries = new ArrayList<>();
    adapter = new StepsListAdapter(this, R.layout.list_row, stepEntries);
    listViewSteps.setAdapter(adapter);
    ...
    // Initialize sensor and database helper
    dbHelper = new StepsDBHelper(context);
    mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    if (mSensorManager.getDefaultSensor(Sensor.TYPE_STEP_COUNTER) != null) {
        mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_STEP_COUNTER);
        isSensorPresent = true;
    } else {
        isSensorPresent = false;
        mStepsSinceReboot.setText("Step Counter Sensor not available!");
    }
    ...
    // Requesting permissions
    if (ContextCompat.checkSelfPermission(context, Manifest.permission.ACTIVITY_RECOGNITION) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(activity, new String[]{Manifest.permission.ACTIVITY_RECOGNITION}, REQUEST_PERMISSIONS_CODE);
    }
    // Start the StepService
    startService(new Intent(packageContext, StepService.class));
    // Registering broadcast receiver to get updates from StepService
    stepUpdateReceiver = (BroadcastReceiver) (context, intent) > {
        updateStepList();
    };
}

```

Run app

Hw15updatedpm - Version control

MainActivity.java [Hw15updatedpm.app.main]

```

// Start the StepService
startService(new Intent(packageContext, StepService.class));
// Registering broadcast receiver to get updates from StepService
stepUpdateReceiver = (BroadcastReceiver) (context, intent) > {
    updateStepList();
};
// Reset button setup
resetButton.setOnClickListener(v > {
    // Reset the steps
    dbHelper.resetSteps();
    lastStepCount = 0;
    updateStepList();
    mStepsSinceReboot.setText("Steps walked: 0");
});
updateStepList();
}

@Override
protected void onResume() {
    super.onResume();
    if (isSensorPresent & ContextCompat.checkSelfPermission(context, Manifest.permission.ACTIVITY_RECOGNITION) != PackageManager.PERMISSION_GRANTED) {
        mSensorManager.registerListener(listener, mSensor, SensorManager.SENSOR_DELAY_GAME);
    }
    // Registering for broadcasts when the steps are updated
    registerReceiver(stepUpdateReceiver, new IntentFilter(StepService.ACTION_UPDATE_STEPS));
}

@Override
protected void onPause() {
    super.onPause();
    if (isSensorPresent) {
        mSensorManager.unregisterListener(this);
    }
    unregisterReceiver(stepUpdateReceiver);
}

```

Run app

```

private void updateStepList() {
    // To Fetch data from the database and update the list view
    stepEntries.clear();
    stepEntries.addAll(dbHelper.getAggregatedStepEntries());
    adapter.notifyDataSetChanged();
}

@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_STEP_COUNTER) {
        int totalStepsSinceReboot = (int) event.values[0];

        if (initialStepCount == -1) {
            initialStepCount = totalStepsSinceReboot;
        }

        int currentSessionSteps = totalStepsSinceReboot - initialStepCount;
        mStepsSinceReboot.setText("Steps walked: " + currentSessionSteps);
        if (currentSessionSteps > lastStepCount) {
            int newSteps = currentSessionSteps - lastStepCount;
            lastStepCount = currentSessionSteps;
            if (newSteps > 0) {
                dbHelper.createStepsEntry(newSteps);
                sendBroadcast(new Intent(StepService.ACTION_UPDATE_STEPS));
            }
        }
    }
}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == 1) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            if (isSensorPresent) {
                mSensorManager.registerListener(listener, mSensor, SensorManager.SENSOR_DELAY_UI);
            }
        }
    }
}

```

```

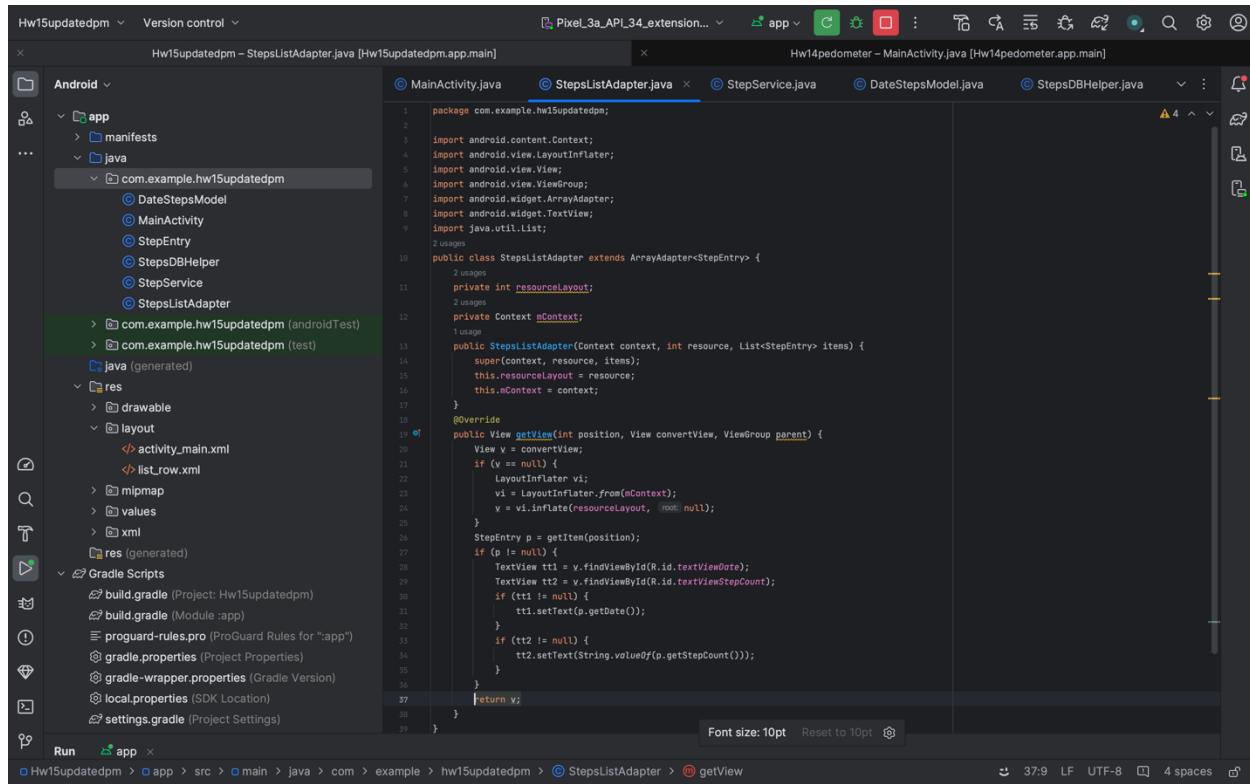
int currentSessionSteps = totalStepsSinceReboot - initialStepCount;
mStepsSinceReboot.setText("Steps walked: " + currentSessionSteps);
if (currentSessionSteps > lastStepCount) {
    int newSteps = currentSessionSteps - lastStepCount;
    lastStepCount = currentSessionSteps;
    if (newSteps > 0) {
        dbHelper.createStepsEntry(newSteps);
        sendBroadcast(new Intent(StepService.ACTION_UPDATE_STEPS));
    }
}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == 1) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            if (isSensorPresent) {
                mSensorManager.registerListener(listener, mSensor, SensorManager.SENSOR_DELAY_UI);
            }
        }
    }
}

```

StepsListAdapter.java : In Android, the 'StepsListAdapter' class is used to populate a 'ListView' with step count data. It extends ArrayAdapter<StepEntry>, a general adapter for dealing with lists of 'StepEntry' objects. It inflates the layout for each list item in its 'getView' method and sets the date and step count values from the 'StepEntry' objects. This adapter functions as a link between the data (step entries) and the UI (list view) by efficiently displaying step count information in the list view.



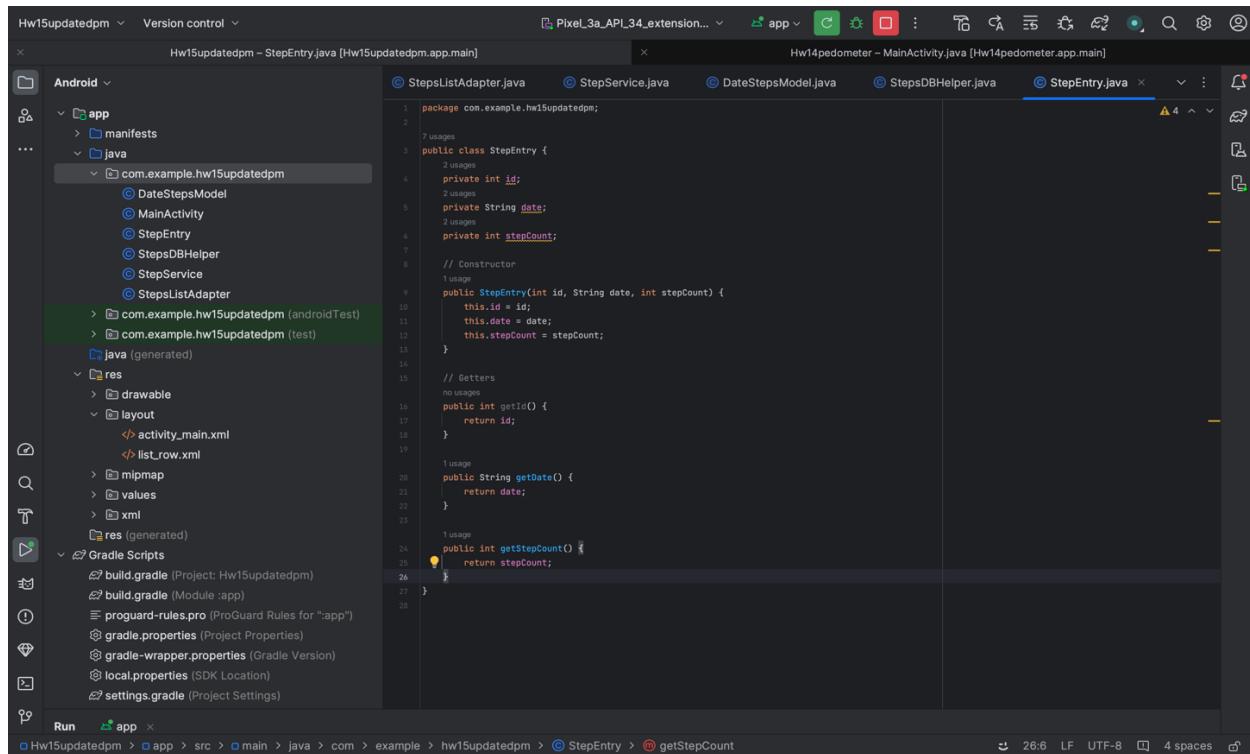
The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `StepsListAdapter.java` file from the `com.example.hw15updatedpm` package. The code implements a custom adapter for a list of step entries, extending `ArrayAdapter<StepEntry>`. It overrides the `getView` method to inflate a layout for each item in the list.

```

1 package com.example.hw15updatedpm;
2
3 import android.content.Context;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7 import android.widget.ArrayAdapter;
8 import android.widget.TextView;
9 import java.util.List;
10
11 public class StepsListAdapter extends ArrayAdapter<StepEntry> {
12     private int resourceLayout;
13     private Context mContext;
14
15     public StepsListAdapter(Context context, int resource, List<StepEntry> items) {
16         super(context, resource, items);
17         this.resourceLayout = resource;
18         this.mContext = context;
19     }
20
21     @Override
22     public View getView(int position, View convertView, ViewGroup parent) {
23         View v = convertView;
24         if (v == null) {
25             LayoutInflater vi =
26                 LayoutInflater.from(mContext);
27             v = vi.inflate(resourceLayout, parent, false);
28         }
29         StepEntry p = getItem(position);
30         if (p != null) {
31             TextView tt1 = v.findViewById(R.id.textViewDate);
32             TextView tt2 = v.findViewById(R.id.textViewStepCount);
33             if (tt1 != null) {
34                 tt1.setText(p.getDate());
35             }
36             if (tt2 != null) {
37                 tt2.setText(String.valueOf(p.getStepCount()));
38             }
39         }
40         return v;
41     }
42 }

```

StepEntry : The 'StepEntry' class represents a step count entry with ID, date, and step count attributes. It is used to encapsulate step count data and provides getters for accessing these characteristics, aiding systematic step count storage and retrieval.



The screenshot shows the Android Studio interface with the project structure on the left and the code editor on the right. The code editor displays the `StepEntry.java` file from the `com.example.hw15updatedpm` package. The class defines attributes for `id`, `date`, and `stepCount`, along with constructor and getter methods.

```

1 package com.example.hw15updatedpm;
2
3 public class StepEntry {
4     private int id;
5     private String date;
6     private int stepCount;
7
8     // Constructor
9     public StepEntry(int id, String date, int stepCount) {
10         this.id = id;
11         this.date = date;
12         this.stepCount = stepCount;
13     }
14
15     // Getters
16     public int getId() {
17         return id;
18     }
19
20     public String getDate() {
21         return date;
22     }
23
24     public int getStepCount() {
25         return stepCount;
26     }
27 }

```

`Steps DBHelper.java` : The class is a required component of every Android app, acting as a SQLite database helper for managing step count data. It creates and maintains the 'StepsSummary' database table, which has columns for ID, step count, and creation date. This class contains critical methods for the app's functionality, such as 'createStepsEntry', which adds new step entries with the current step count and date, 'getTodayStepCount', which retrieves the total step count for the current day, 'getAggregatedStepEntries', which aggregates step count data by date, and 'resetSteps', which clears all step count entries, effectively resetting the step count. It allows for quick data storage and retrieval procedures for the Android app's step tracking.

The screenshot shows the Android Studio interface with the following details:

- Project Structure:** The left sidebar displays the project structure under "Hw15updatedpm". It includes the app module with Java files like MainActivity.java, StepService.java, DateStepsModel.java, and StepsDBHelper.java. The test module contains androidTest and test packages. Resource files (res) include drawable, layout (activity_main.xml, list_row.xml), mipmap, values, and xml. Gradle Scripts show build.gradle, build.gradle (Module: app), proguard-rules.pro, gradle.properties, gradle-wrapper.properties, local.properties, and settings.gradle.
- Code Editor:** The main window shows the code for StepsDBHelper.java. The code defines a SQLiteOpenHelper subclass named StepsDBHelper that extends SQLiteOpenHelper. It includes methods for onCreate and onUpgrade, and a static inner class named StepsDBModel that defines the table structure for the database.

```
public class MainActivity extends AppCompatActivity {
    private StepsListRecyclerView recyclerView;
    private StepsListAdapter adapter;
    private StepService stepService;
    private DateStepsModel dateStepsModel;
    private StepsDBHelper stepsDBHelper;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recyclerView = findViewById(R.id.recyclerView);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        adapter = new StepsListAdapter();
        recyclerView.setAdapter(adapter);
        stepService = new StepService();
        dateStepsModel = new DateStepsModel();
        stepsDBHelper = new StepsDBHelper(this);
        stepsDBHelper.createDatabase();
        stepService.getStepCount(dateStepsModel);
        adapter.setSteps(dateStepsModel.getSteps());
    }

    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_STEPS_SUMMARY);
        onCreate(db);
    }

    public boolean createStepsEntry(int steps) {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd", Locale.getDefault());
        String todayDate = sdf.format(new Date());
        ContentValues values = new ContentValues();
        values.put(STEPS_COUNT, steps);
        values.put(CREATION_DATE, todayDate);

        long result = db.insert(TABLE_STEPS_SUMMARY, null, values);
        db.close();
        return result != -1;
    }

    public int getTodayStepCount() {
        int stepCount = 0;
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd", Locale.getDefault());
        String todayDate = sdf.format(new Date());

        SQLiteDatabase db = this.getReadableDatabase();
        Cursor cursor = db.rawQuery(TABLE_STEPS_SUMMARY, new String[]{String.format("%s %s", "SUM", STEPS_COUNT), "as total"}, new String[]{todayDate}, null, null, null, null);
        cursor.moveToFirst();
        if (cursor.moveToFirst()) {
            int columnIndex = cursor.getColumnIndex("total");
            if (columnIndex != -1) {
                stepCount = cursor.getInt(columnIndex);
            }
        }
        cursor.close();
        db.close();
        return stepCount;
    }
}
```

```

1  package com.example.hw15updatedpm;
2
3  import android.database.sqlite.SQLiteDatabase;
4  import android.database.sqlite.SQLiteOpenHelper;
5  import android.util.Log;
6
7  public class StepsDBHelper extends SQLiteOpenHelper {
8
9      private static final String DATABASE_NAME = "StepCounterDatabase";
10     private static final int DATABASE_VERSION = 1;
11
12     public static final String TABLE_STEPS_SUMMARY = "steps_summary";
13
14     public static final String COLUMN_DATE = "creation_date";
15     public static final String COLUMN_STEPS = "steps";
16
17     private static final String SQL_CREATE_TABLE_STEPS_SUMMARY =
18         "CREATE TABLE IF NOT EXISTS " + TABLE_STEPS_SUMMARY + "(" +
19             COLUMN_DATE + " TEXT, " +
20             COLUMN_STEPS + " INTEGER, " +
21             "PRIMARY KEY (" + COLUMN_DATE + "));";
22
23     private static final String SQL_DELETE_TABLE_STEPS_SUMMARY =
24         "DROP TABLE IF EXISTS " + TABLE_STEPS_SUMMARY + ";";
25
26     public StepsDBHelper(Context context) {
27         super(context, DATABASE_NAME, null, DATABASE_VERSION);
28     }
29
30     @Override
31     public void onCreate(SQLiteDatabase db) {
32         db.execSQL(SQL_CREATE_TABLE_STEPS_SUMMARY);
33     }
34
35     @Override
36     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
37         db.execSQL(SQL_DELETE_TABLE_STEPS_SUMMARY);
38         onCreate(db);
39     }
40
41     public Cursor getAllEntries() {
42         SQLiteDatabase db = this.getReadableDatabase();
43
44         String query = "SELECT * FROM " + TABLE_STEPS_SUMMARY + " ORDER BY " + COLUMN_DATE + " DESC";
45
46         Cursor cursor = db.rawQuery(query, null);
47
48         if (cursor != null) {
49             cursor.moveToFirst();
50
51             while (!cursor.isLast()) {
52                 Log.d("StepCounterDatabase", "Entry: " + cursor.getString(0) + ", " + cursor.getInt(1));
53                 cursor.moveToNext();
54             }
55         }
56
57         return cursor;
58     }
59
60     public void addEntry(String date, int steps) {
61         SQLiteDatabase db = this.getWritableDatabase();
62
63         ContentValues contentValues = new ContentValues();
64         contentValues.put(COLUMN_DATE, date);
65         contentValues.put(COLUMN_STEPS, steps);
66
67         db.insert(TABLE_STEPS_SUMMARY, null, contentValues);
68     }
69
70     public void resetSteps() {
71         SQLiteDatabase db = this.getWritableDatabase();
72
73         db.delete(TABLE_STEPS_SUMMARY, null, null);
74         db.close();
75     }
76
77     public List<StepEntry> getAggregatedEntries() {
78         List<StepEntry> aggregatedEntries = new ArrayList<>();
79         SQLiteDatabase db = this.getReadableDatabase();
80
81         String query = "SELECT * FROM " + TABLE_STEPS_SUMMARY + " GROUP BY " + COLUMN_DATE + " ORDER BY " + COLUMN_DATE + " DESC";
82
83         Cursor cursor = db.rawQuery(query, null);
84
85         if (cursor != null) {
86             cursor.moveToFirst();
87
88             while (cursor.moveToNext()) {
89                 StepEntry entry = new StepEntry(cursor.getString(0), cursor.getInt(1));
90                 aggregatedEntries.add(entry);
91             }
92         }
93
94         cursor.close();
95
96         return aggregatedEntries;
97     }
98
99     public void close() {
100        db.close();
101    }
102 }

```

StepService.java : The 'StepService' class is an Android service that tracks steps using the step counter sensor. It adds the sensor listener to the 'onCreate' method in order to observe step events. When step events occur, the 'StepsDBHelper' records the step count in the database and transmits an update signal. This service operates in the background, ensuring that step tracking continues even when the app is not in use.

```

1  package com.example.hw15updatedpm;
2
3  import android.app.Service;
4  import android.content.Intent;
5  import android.hardware.Sensor;
6  import android.hardware.SensorEvent;
7  import android.hardware.SensorEventListener;
8  import android.hardware.SensorManager;
9  import android.os.IBinder;
10  import android.annotation.Nullable;
11
12  public class StepService extends Service implements SensorEventListener {
13
14     private SensorManager sensorManager;
15     private Sensor stepCounterSensor;
16
17     private StepsDBHelper dbHelper;
18
19     public static final String ACTION_UPDATE_STEPS = "com.example.hw15updatedpm.ACTION_UPDATE_STEPS";
20
21     @Override
22     public void onCreate() {
23         super.onCreate();
24         sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
25         stepCounterSensor = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_COUNTER);
26         dbHelper = new StepsDBHelper(getApplicationContext());
27
28         if (stepCounterSensor != null) {
29             sensorManager.registerListener(this, stepCounterSensor, SensorManager.SENSOR_DELAY_UI);
30         }
31     }
32
33     @Override
34     public int onStartCommand(Intent intent, int flags, int startId) {
35         return START_STICKY;
36     }
37
38     @Override
39     public void onDestroy() {
40         super.onDestroy();
41         if (sensorManager != null) {
42             sensorManager.unregisterListener(this);
43         }
44     }
45
46     @Override
47     public void onSensorChanged(SensorEvent event) {
48         if (event.sensor.getType() == Sensor.TYPE_STEP_COUNTER) {
49             int steps = (int) event.values[0];
50
51             dbHelper.addEntry(event.timestamp, steps);
52         }
53     }
54
55     @Override
56     public IBinder onBind(Intent intent) {
57         return null;
58     }
59
60 }

```

```

public static final String ACTION_UPDATE_STEPS = "com.example.hw15updatedpm.ACTION_UPDATE_STEPS";
    ...
    @Override
    public void onCreate() {
        super.onCreate();
        sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        stepCounterSensor = sensorManager.getDefaultSensor(Sensor.TYPE_STEP_COUNTER);
        dbHelper = new StepsDBHelper(context: this);
        if (stepCounterSensor != null) {
            sensorManager.registerListener( listener: this, stepCounterSensor, SensorManager.SENSOR_DELAY_UI);
        }
    }
    ...
    @Override
    public void onStartCommand(Intent intent, int flags, int startId) { return START_STICKY; }
    ...
    @Override
    public void onDestroy() {
        super.onDestroy();
        if (sensorManager != null) {
            sensorManager.unregisterListener(this);
        }
    }
    ...
    @Override
    public void onSensorChanged(SensorEvent event) {
        if (event.sensor.getType() == Sensor.TYPE_STEP_COUNTER) {
            int steps = (int) event.values[0];
            if (dbHelper.createStepsEntry(steps)) {
                sendBroadcast(new Intent(ACTION_UPDATE_STEPS));
            }
        }
    }
    ...
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // Not used
    }
    ...
    @Override
    public IBinder onBind(Intent intent) { return null; }
}

```

DateStepsModel.java : In the Android application, the 'DateStepsModel' class is a simple data model. It contains date-step count pairs, as well as getters and setters for both the date and the step count. Within the app's functionality, this class is often used to represent and manage step count data as well as matching dates.

```

package com.example.hw15updatedpm;
...
no usages
public class DateStepsModel {
    private String mDate;
    private int mStepCount;
    ...
    public DateStepsModel(String date, int stepCount) {
        this.mDate = date;
        this.mStepCount = stepCount;
    }
    ...
    public String getDate() { return mDate; }
    ...
    public void setDate(String date) { this.mDate = date; }
    ...
    public int getStepCount() { return mStepCount; }
    ...
    public void setStepCount(int stepCount) { this.mStepCount = stepCount; }
}

```

Activity_main.xml : In the Android application, the 'DateStepsModel' class is a simple data model. It contains date-step count pairs, as well as getters and setters for both the date and the step count. Within the app's functionality, this class is often used to represent and manage step count data as well as matching dates.

```

<?xml version="1.0" encoding="utf-8?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"

    <TextView
        android:id="@+id/stepsincereboot"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Steps walked: 0"
        android:textSize="18sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="16dp" />

    <ListView
        android:id="@+id/listViewSteps"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:divider="@android:color/darker_gray"
        android:dividerHeight="0.5dp"
        app:layout_constraintTop_toBottomOf="@+id/stepsincereboot"
        app:layout_constraintBottom_toTopOf="@+id/buttonLayout"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <LinearLayout
        android:id="@+id/buttonLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="center"
        android:padding="16dp"
        app:layout_constraintBottom_toBottomOf="parent">

        <Button
            android:id="@+id/refreshButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Refresh" />

        <Button
            android:id="@+id/resetButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Reset"
            android:layout_marginStart="16dp" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

```

<?xml version="1.0" encoding="utf-8?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"

    <TextView
        android:id="@+id/stepsincereboot"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="Steps walked: 0"
        android:textSize="18sp"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:layout_marginTop="16dp" />

    <ListView
        android:id="@+id/listViewSteps"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:divider="@android:color/darker_gray"
        android:dividerHeight="0.5dp"
        app:layout_constraintTop_toBottomOf="@+id/stepsincereboot"
        app:layout_constraintBottom_toTopOf="@+id/buttonLayout"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <LinearLayout
        android:id="@+id/buttonLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:gravity="center"
        android:padding="16dp"
        app:layout_constraintBottom_toBottomOf="parent">

        <Button
            android:id="@+id/refreshButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Refresh" />

        <Button
            android:id="@+id/resetButton"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Reset"
            android:layout_marginStart="16dp" />
    </LinearLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

List_rows.xml : This XML layout defines a horizontal 'LinearLayout' with two 'TextView' elements inside it. The first 'TextView' (identified by '@+id/textViewDate') displays a date, and the second 'TextView' (identified by '@+id/textViewStepCount') displays a step count. The layout is designed for displaying date-step count pairs in a simple and horizontal format.

The screenshot shows the Android Studio interface. The left sidebar displays the project structure under 'Android'. The main area shows the XML code for 'list_row.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="wrap_content"
    android:orientation="horizontal"
    android:padding="10dp">

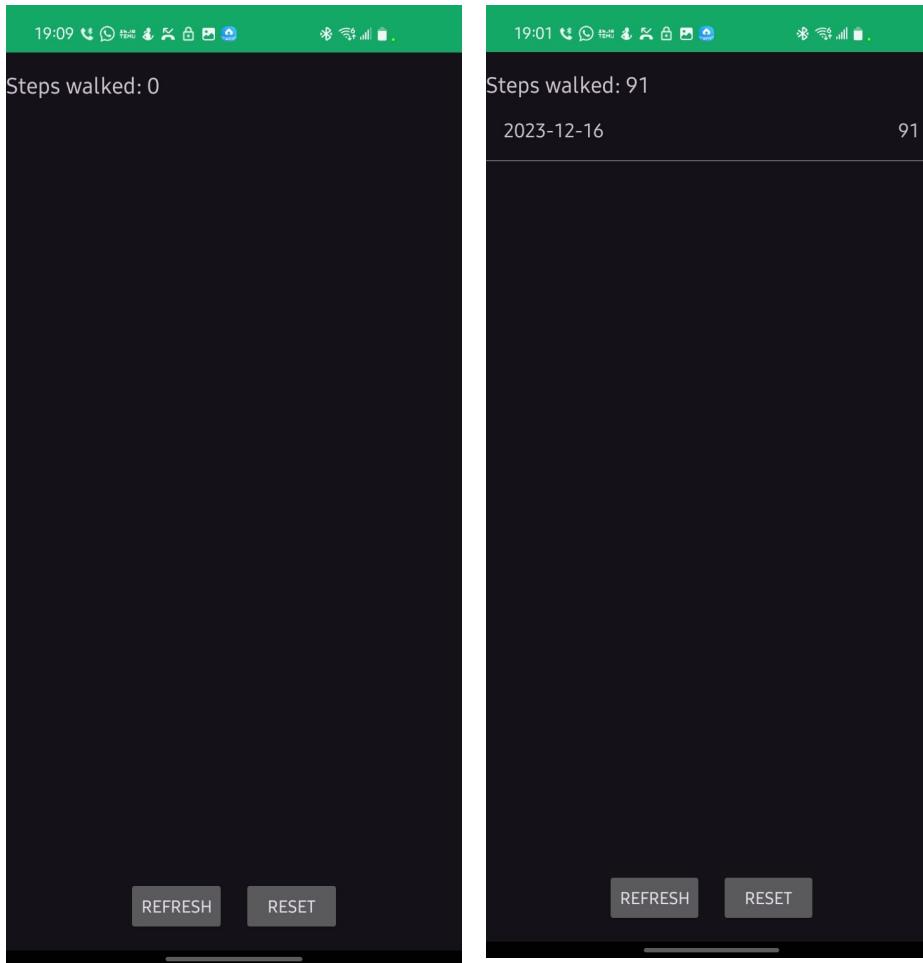
    <TextView
        android:id="@+id/textViewDate"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textSize="16sp" />

    <TextView
        android:id="@+id/textViewStepCount"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:textSize="16sp" />

</LinearLayout>

```

Below are the screenshots of the mainactivity screen and the number of steps walked :



Note : I ran this app in the android as I was crashing on the android studio emulator