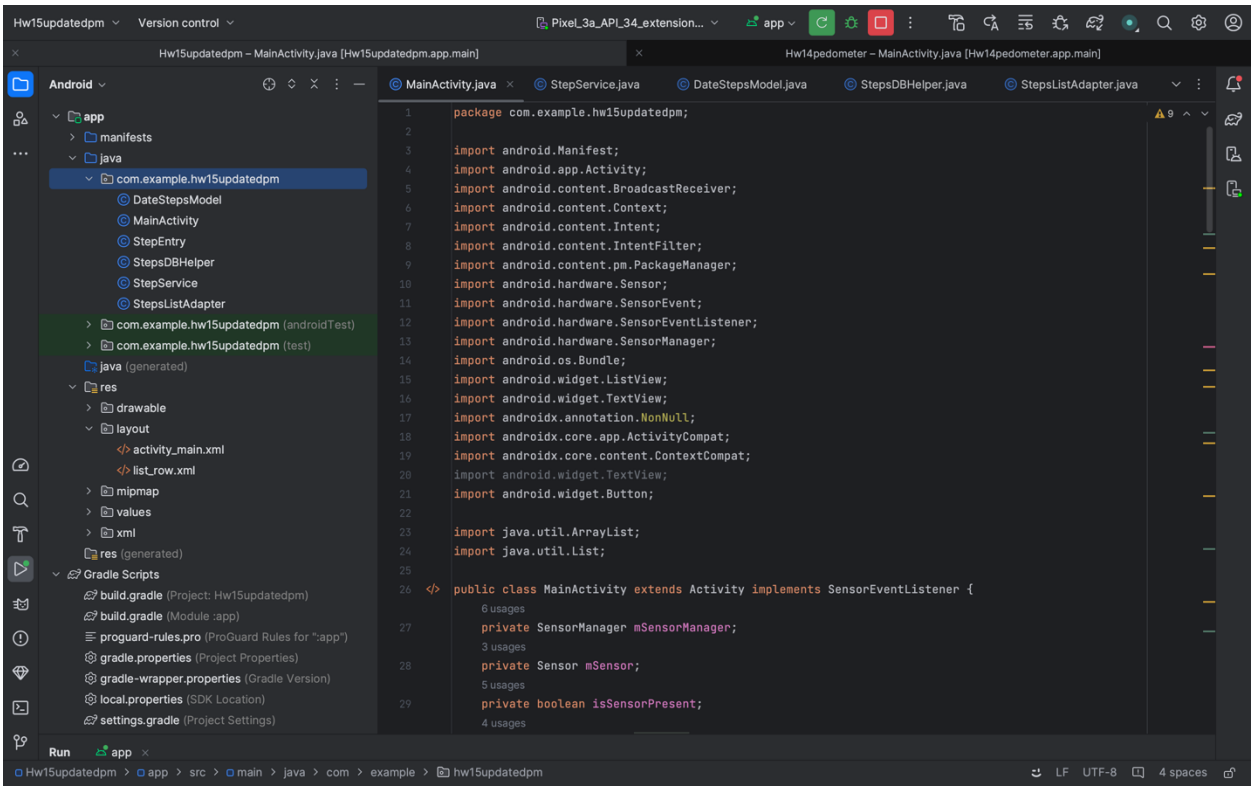
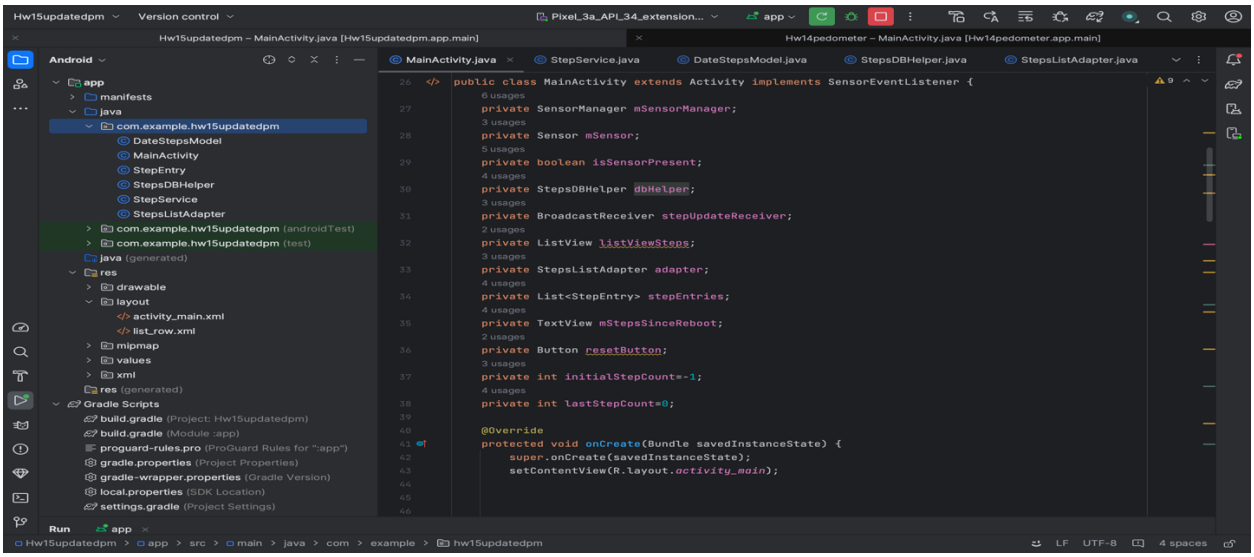
The improved pedometer app with the custom activity detection algorithm has a design issue: the step data are pulled from the SQLite database only at the start of the app. So, if you have run the app for a few minutes with multiple steps, the display the main activity won't change at all. To see the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

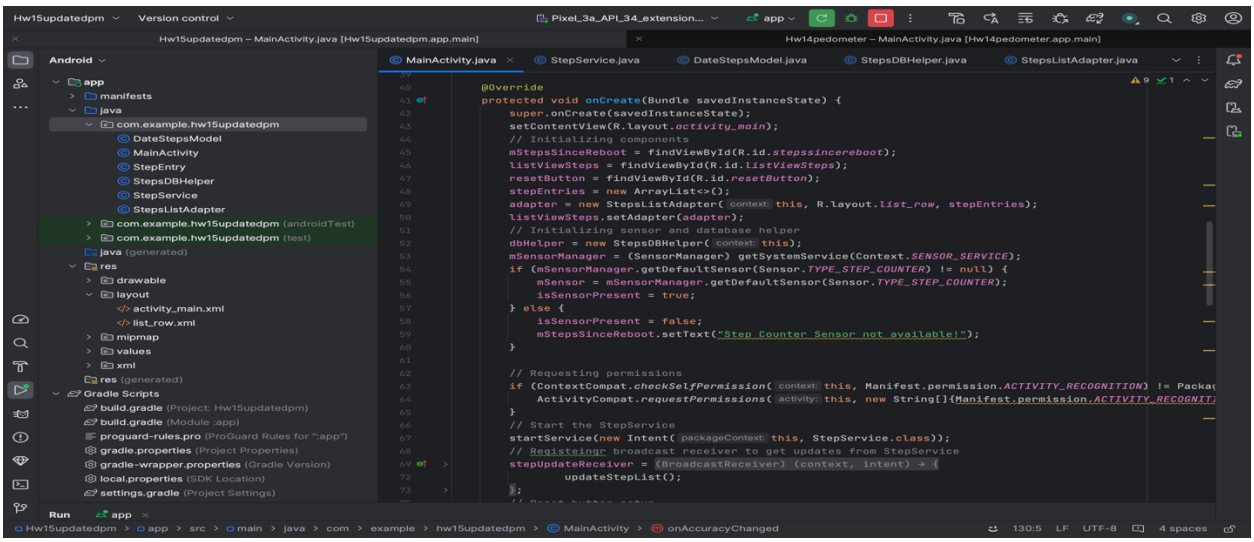
To address this issue, modify the app so that whenever a new batch of steps are recorded, the display on the main activity would be updated. Alternatively, you may add a refresh button on the main activity layout to retrieve the data and display it.

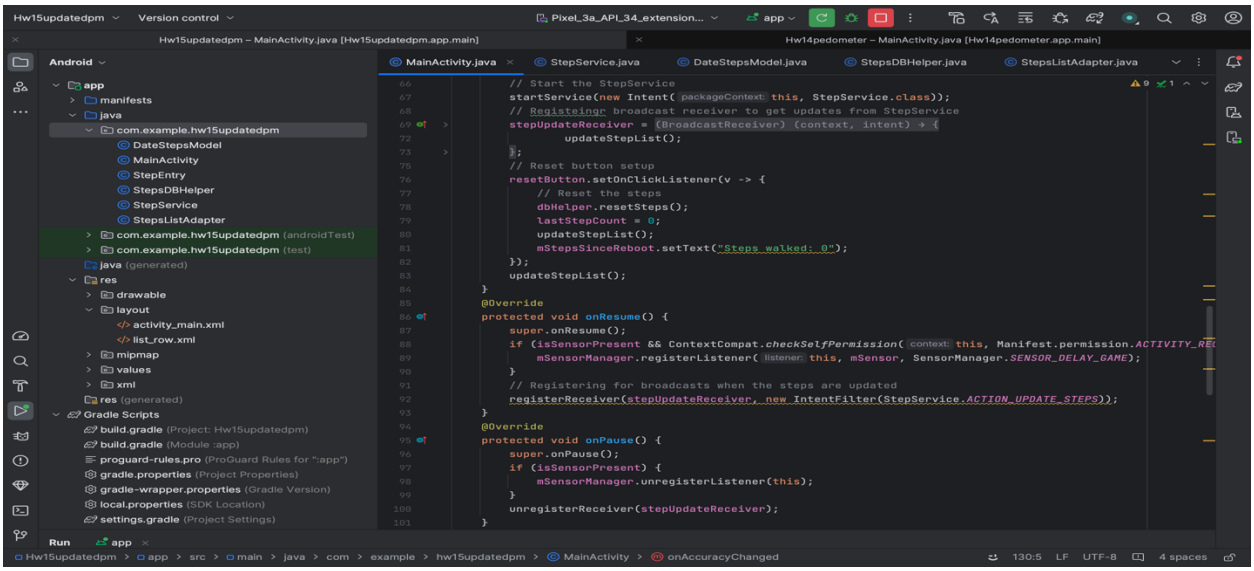
The app actually has another issue: the step count is not as accurate as the step counter. Do some experiments (you will need to carry your phone and walk for say, 100 steps, and compare what is recorded in your app with respect to the actual steps you made) and tune the 3 thresholds used for step counting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

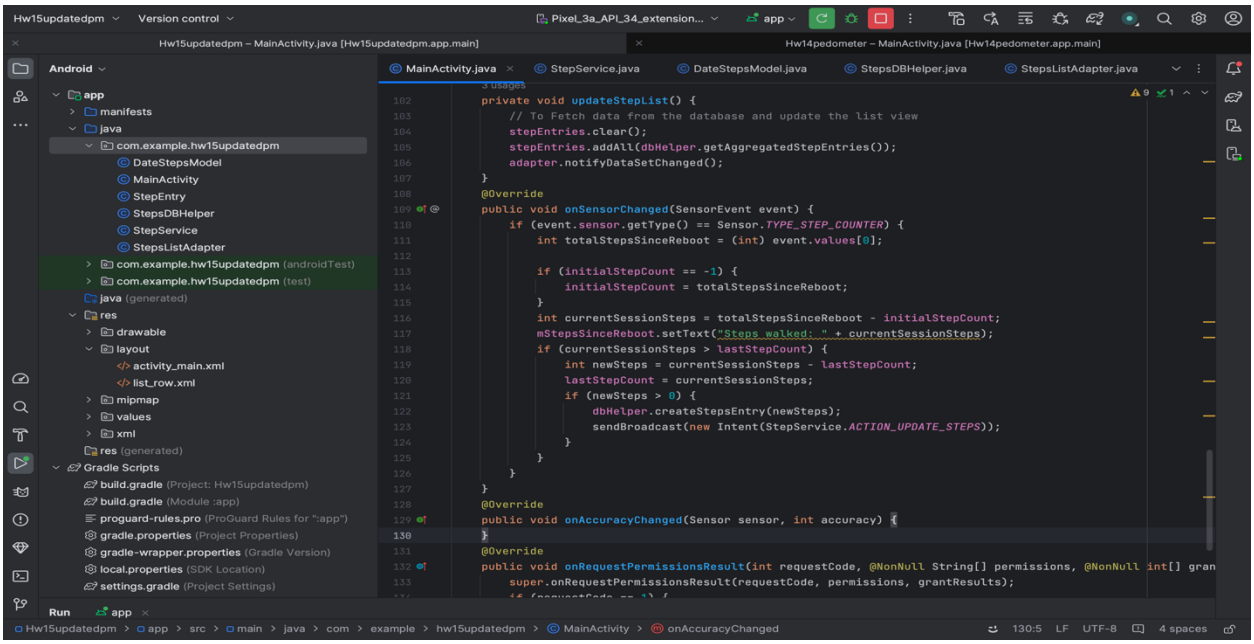
Mainactivity. Java : The 'MainActivity.java' code provided is for an Android step counting app. It loads UI components, such as TextViews and a reset button, and checks for the step counter sensor's availability. The app seeks the necessary permission for activity recognition and launches a background 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.

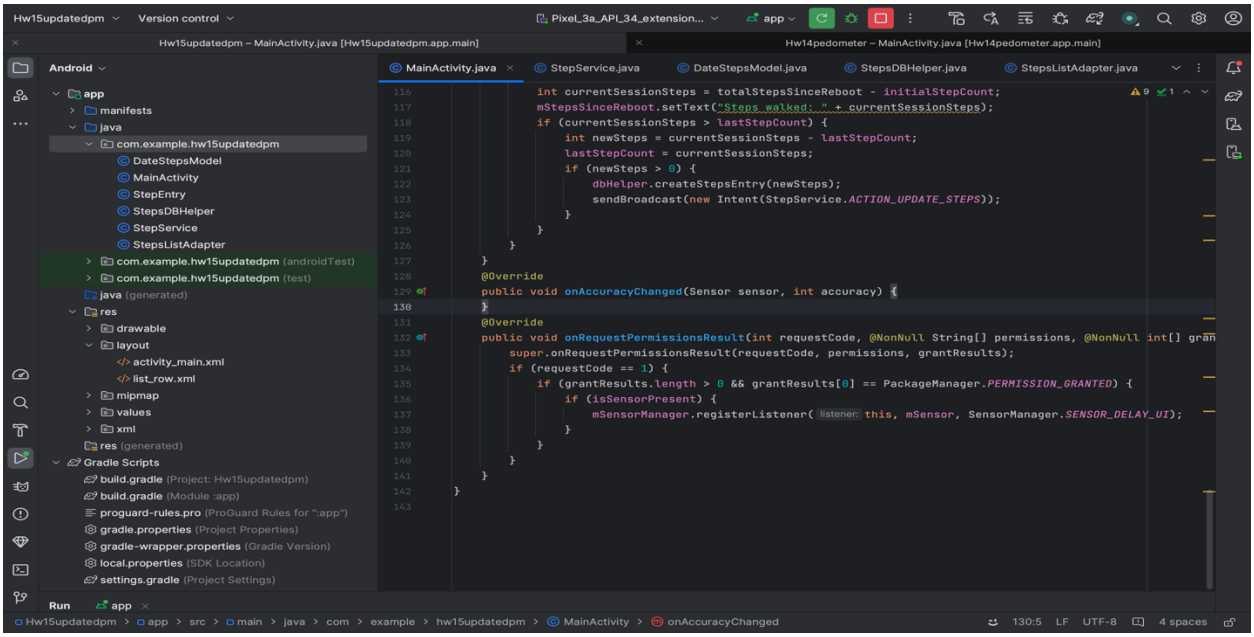



Hw15updatedpm
Version control
Pixel_3a_API_34_extension...
app
Hw15updatedpm – MainActivity.java [Hw15updatedpm.app.main]
Hw14pedometer – MainActivity.java [Hw14pedometer.app.main]
Android
MainActivity.java
StepService.java
DateStepsModel.java
StepsDBHelper.java
StepsListAdapter.java
public class MainActivity 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 List<StepEntry> stepEntries;
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
LF
UTF-8
4 spaces

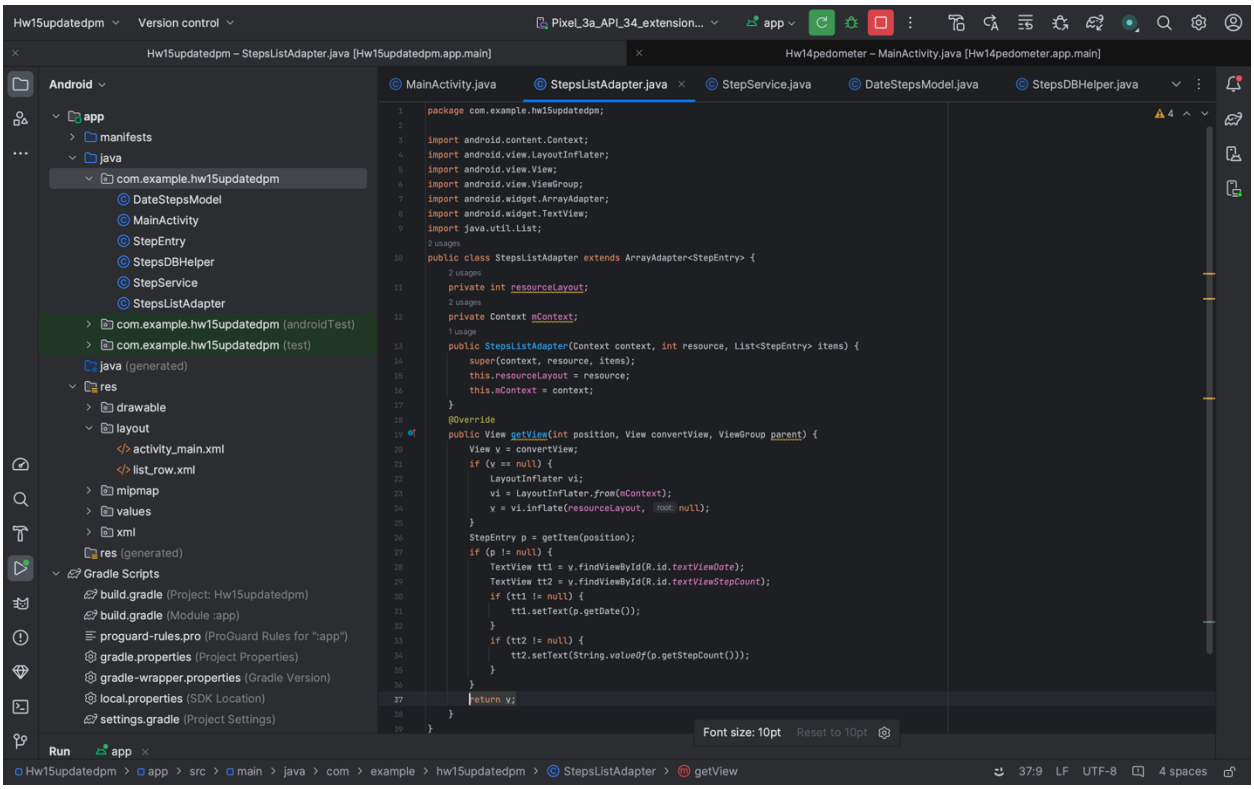

Hw15updatedpm
Version control
Pixel_3a_API_34_extension...
app
Hw15updatedpm – MainActivity.java [Hw15updatedpm.app.main]
Hw14pedometer – MainActivity.java [Hw14pedometer.app.main]
Android
MainActivity.java
StepService.java
DateStepsModel.java
StepsDBHelper.java
StepsListAdapter.java
@Override
protected 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(context: this, R.layout.list_row, stepEntries);
listViewSteps.setAdapter(adapter);
// Initializing sensor and database helper
dbHelper = new StepsDBHelper(context: this);
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
// Start the StepService
startService(new Intent(packageContext: this, StepService.class));
// Registeingr broadcast receiver to get updates from StepService
updateStepList();
Run
app
130:5
LF
UTF-8
4 spaces


Hw15updatedpm
Version control
Pixel_3a_API_34_extension...
app
Hw15updatedpm – MainActivity.java [Hw15updatedpm.app.main]
Hw14pedometer – MainActivity.java [Hw14pedometer.app.main]
Android
MainActivity.java
StepService.java
DateStepsModel.java
StepsDBHelper.java
StepsListAdapter.java
// Start the StepService
startService(new Intent(packageContext: this, StepService.class));
// Registeingr broadcast receiver to get updates from StepService
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: this, Manifest.permission.ACTIVITY_RE
mSensorManager.registerListener(listener: this, 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
130:5
LF
UTF-8
4 spaces

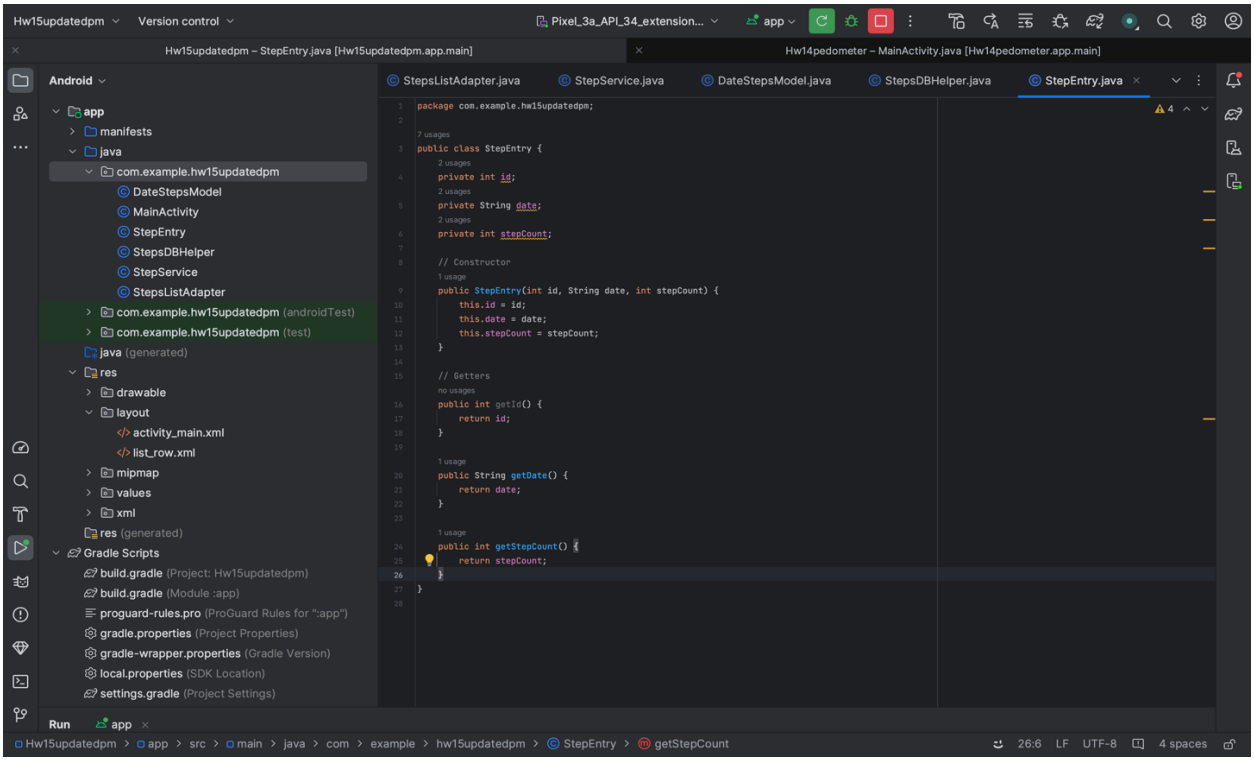




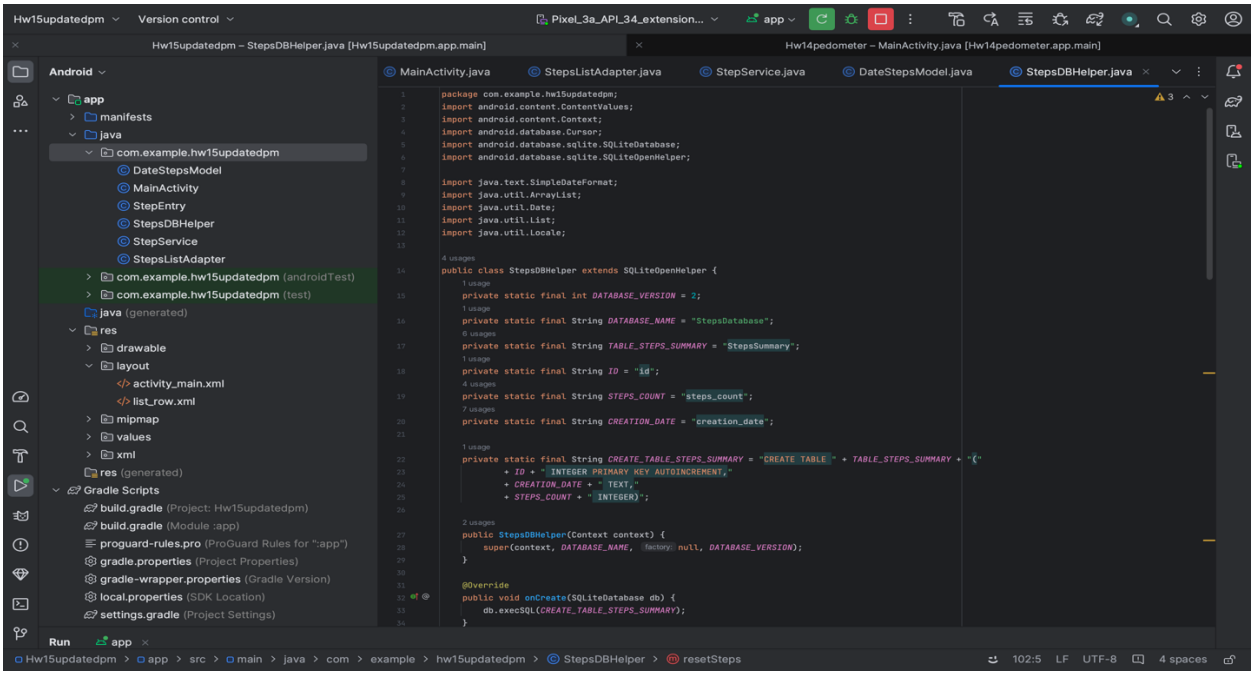
StepsListAdapter.java : In Android, the 'StepsListAdapter' class is used to populate a 'ListView' with step count data. It extends ArrayAdapterStepEntry, a general adapter for dealing with lists of 'StepEntry' objects. It inflates the layout for each list item in its 'getView' method and sets the date 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.

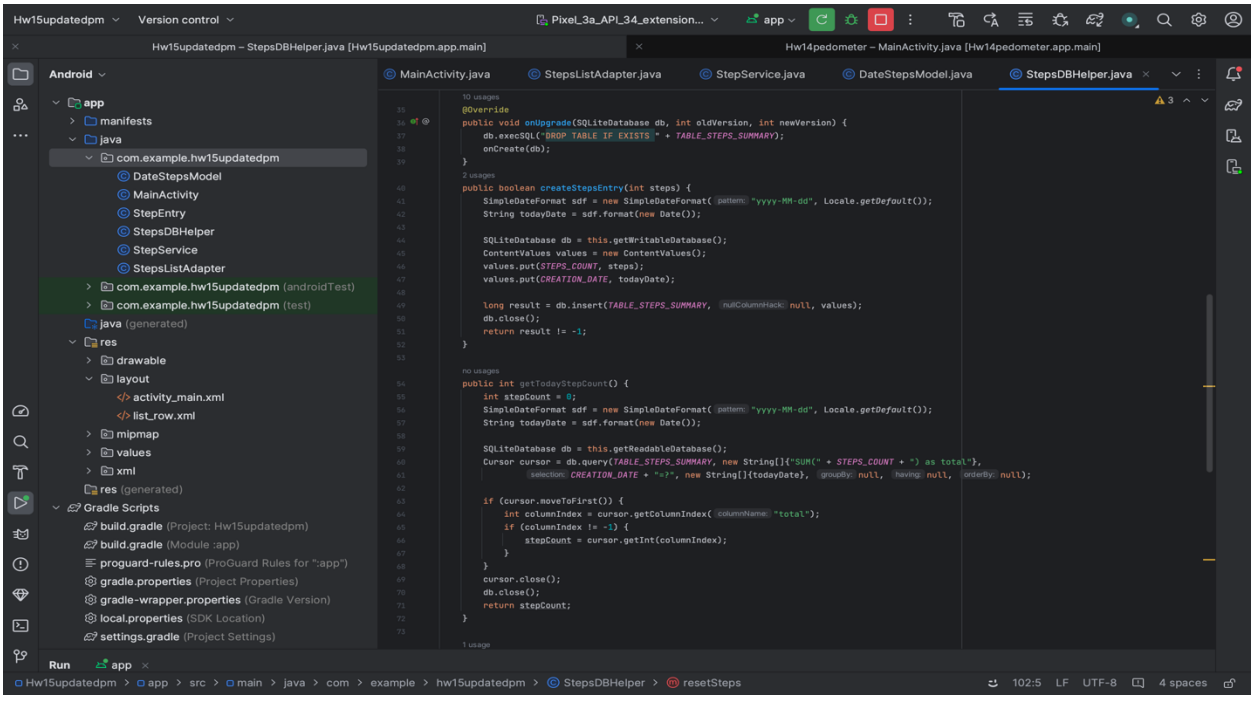


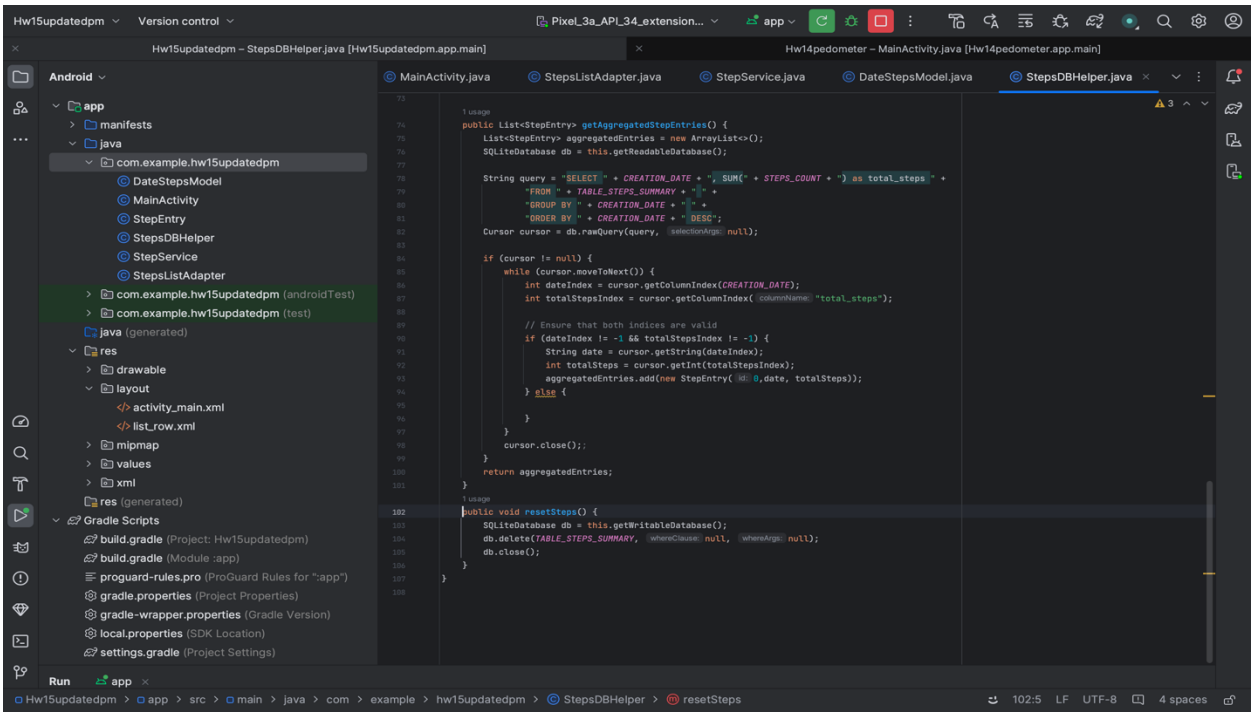
StepEntry : The 'StepEntry' class represents a step count entry with ID, date, and step count attributes. It is used to encapsulate step count data and provides getters for accessing these characteristics, aiding systematic step count storage and retrieval.



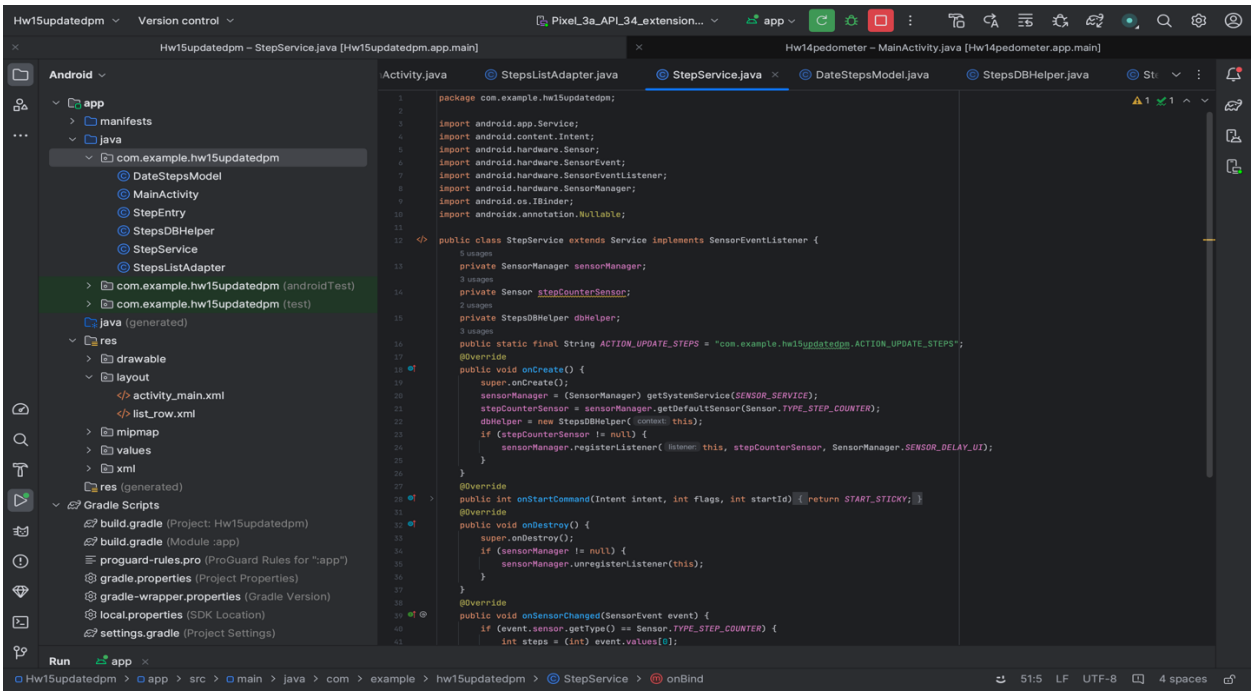
StepsDBHelper.java : The class is a required component of every Android app, acting as a SQLite database helper for managing step count data. It creates and maintains the 'StepsSummary' database table, which has columns for ID, step count, and creation date. This class contains critical methods for the 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.

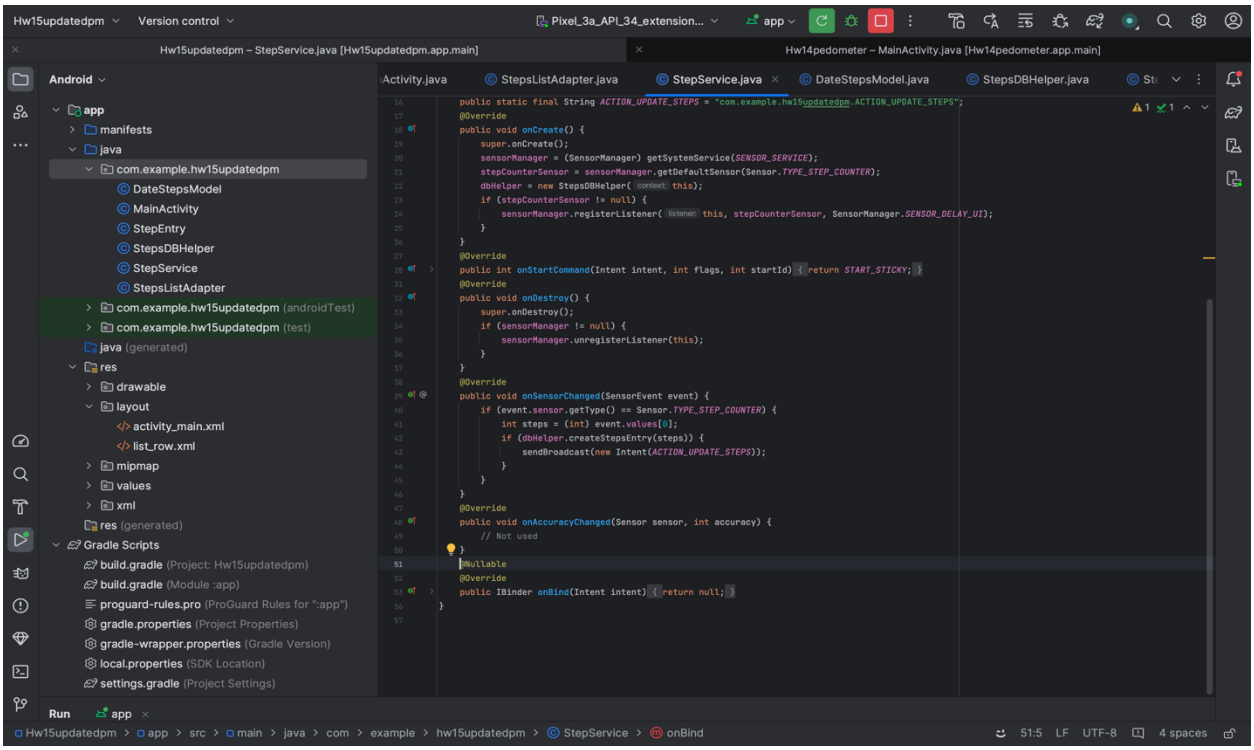




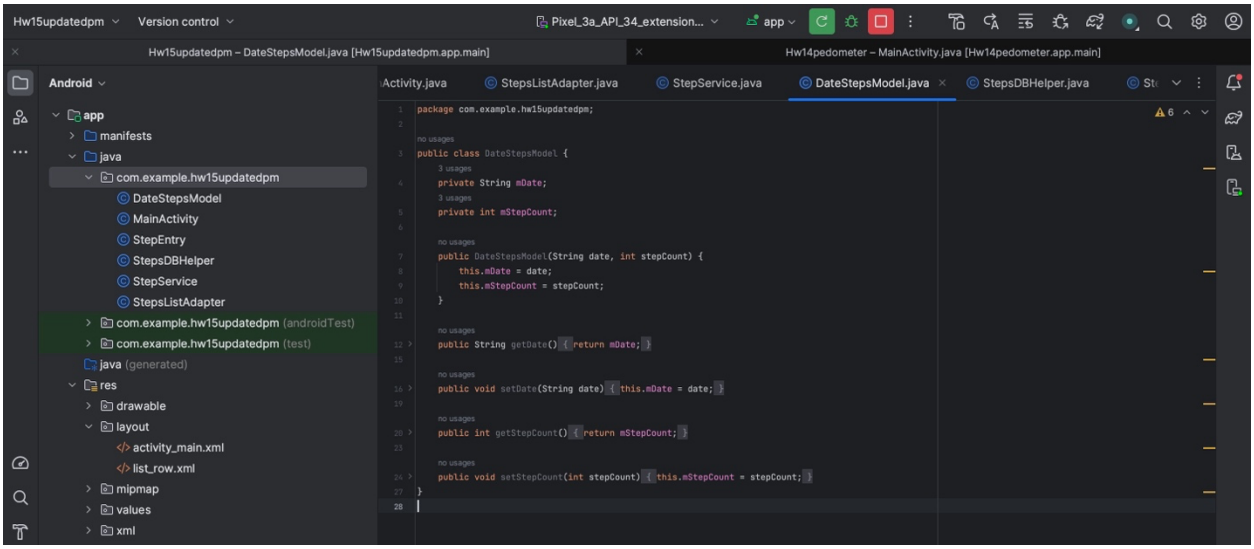


StepService.java : The 'StepService' class is an Android service that tracks steps using the step counter sensor. It adds the sensor listener to the 'onCreate' method in order to observe step events. When step events occur, the 'StepsDBHelper' records the step count in the database and transmits an update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

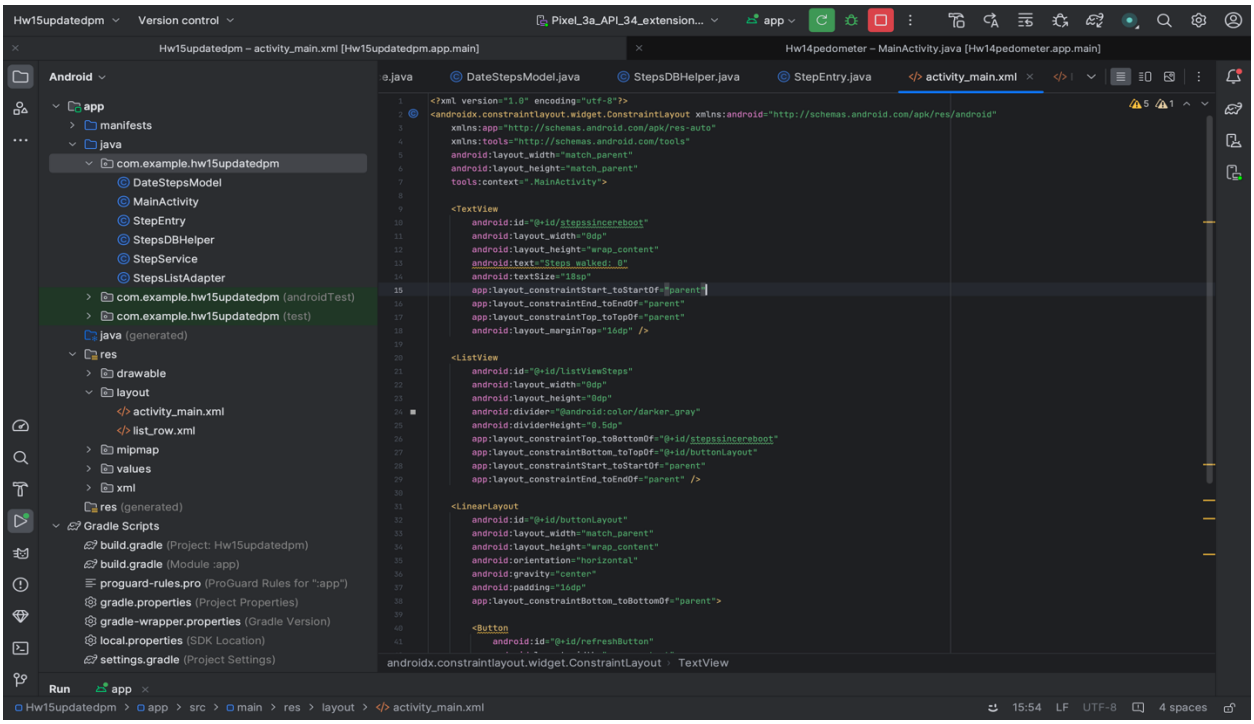


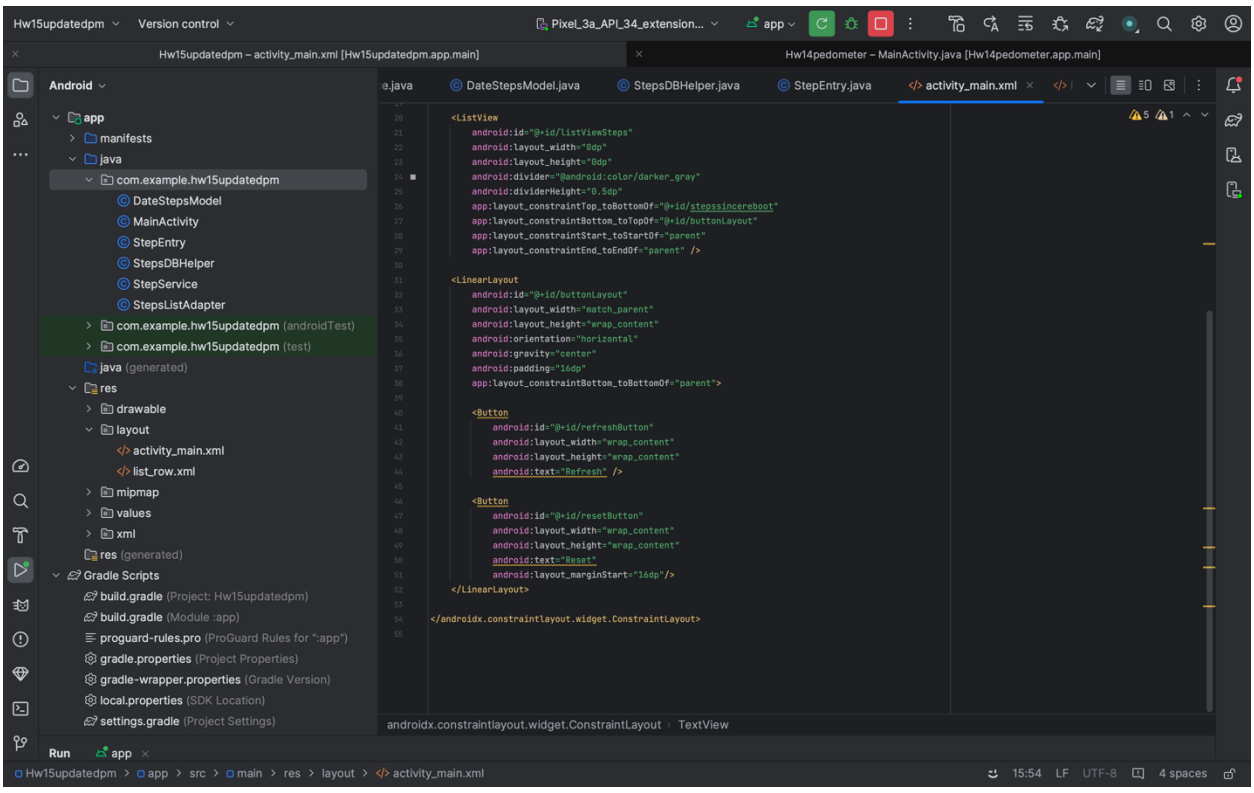


DataStepsModel.java : In the Android application, the 'DateStepsModel' class is a simple data model. It contains date-step count pairs, as well as getters and setters for both the date and the step count. Within the app's functionality, this class is often used to represent and manage step count data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

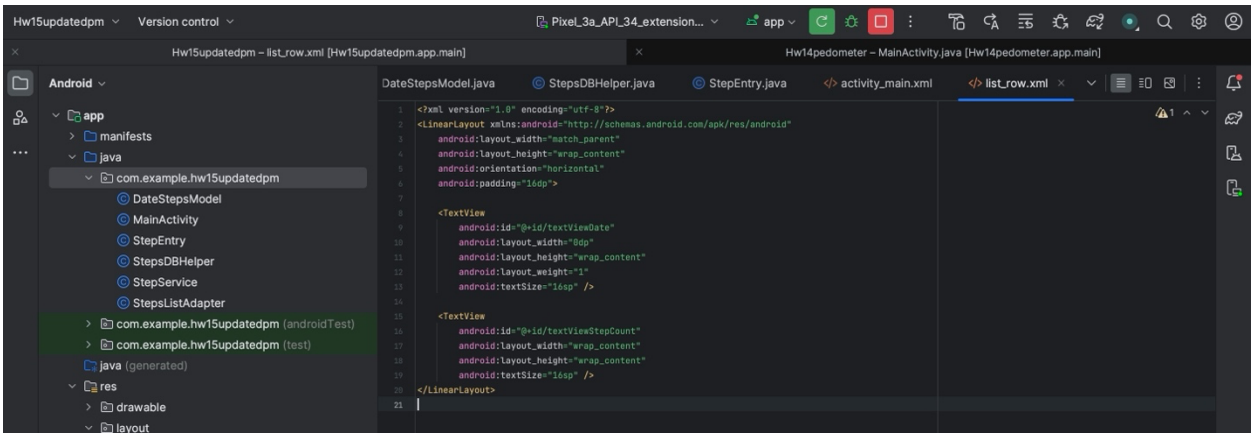


Activity_main.xml : In the Android application, the 'DateStepsModel' class is a simple data model. It contains date-step count pairs, as well as getters and setters for both the date and the step count. Within the app's functionality, this class is often used to represent and manage step count data as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

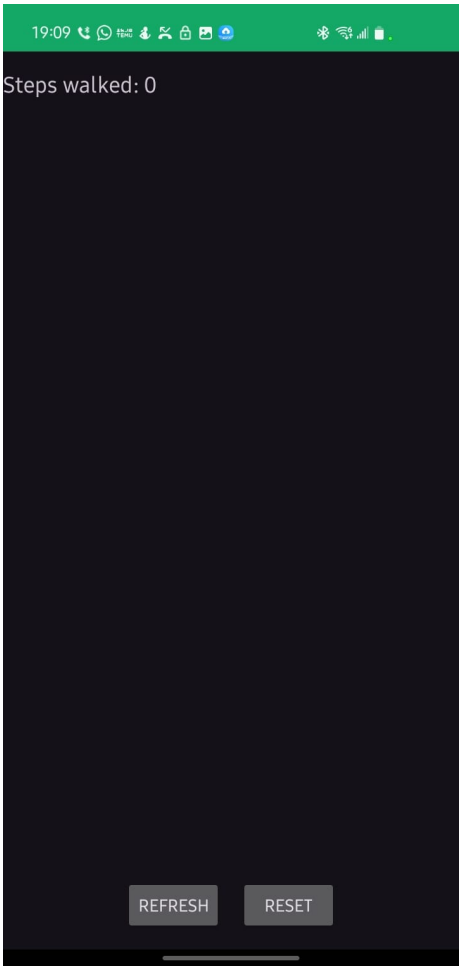


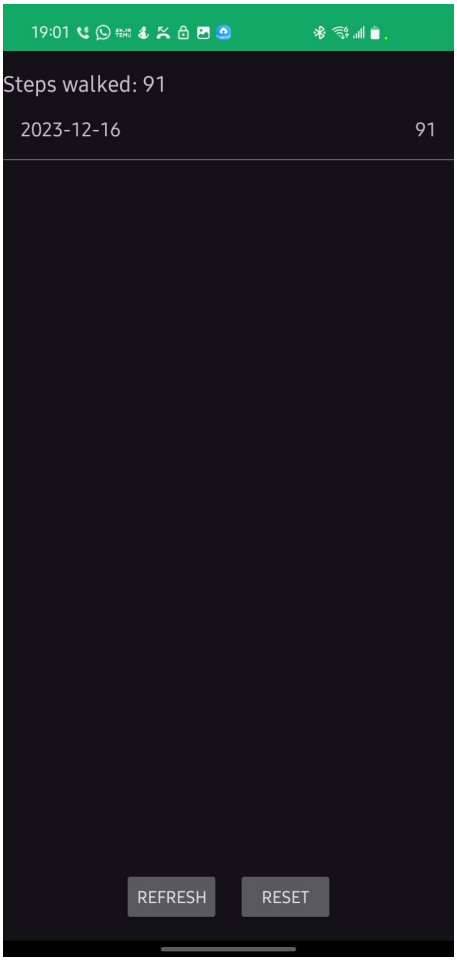


List_rows.xml : This XML layout defines a horizontal `LinearLayout` with two `TextView` elements inside it. The first `TextView` (identified by `@+id/textViewDate`) displays a date, and the second `TextView` (identified by `@+id/textViewStepCount`) displays a step count. The layout is designed for displaying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



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





SS1 mainactivity SS2 mainactivity with steps recorded.

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