Coursework 4 Report

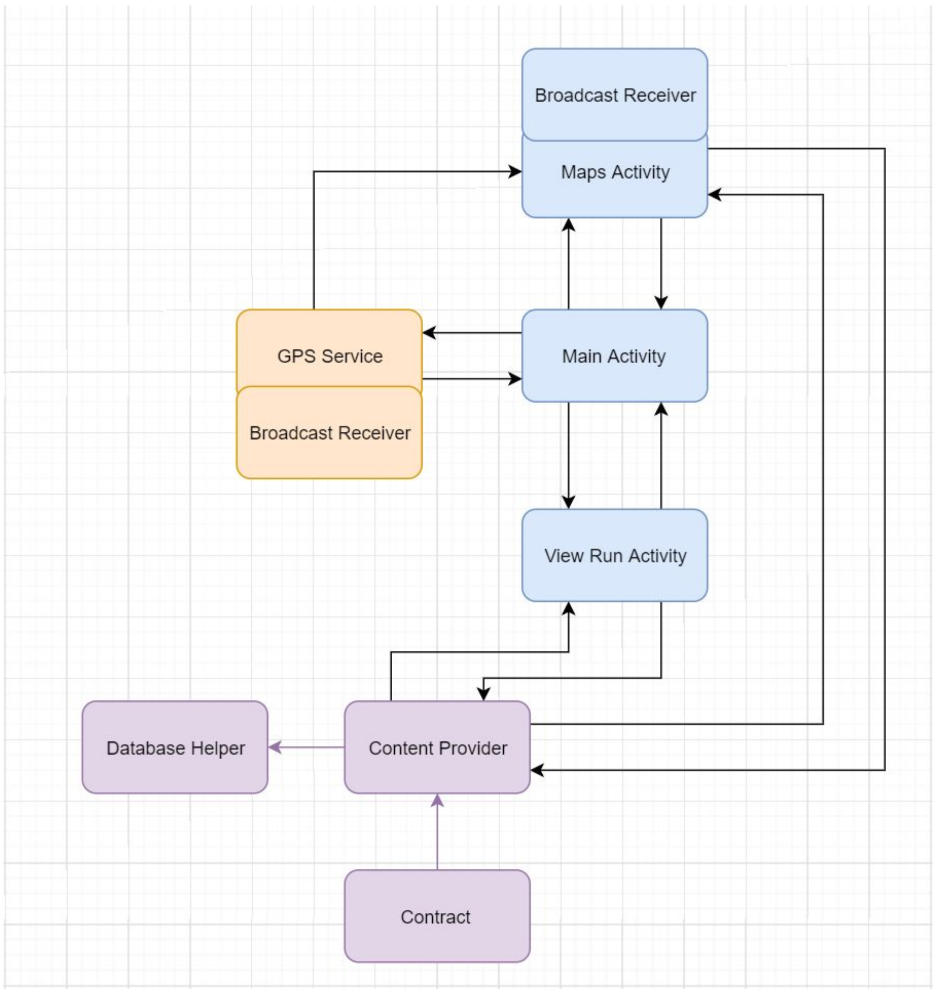
Overview of system

Once the user opens the system he is greeted with a listView which if empty displays nothing and if it has records displays the records which the user can click to check their statistics. There is also 3 buttons, 2 to re-order the list view in descending order based on either time or distance covered in that session. The last button is the start tracking run button which when pressed will open the maps activity. Once the map activity is opened, tracking of the activity has begun and data is updated each time a change in location is registered.There are some textViews which display to the user basic running statistics about his run such as

- Current Speed
- Average Speed
- Time
- Distance

There is also a map which logs the currents movements to a map so the user can actually see a map with trails leading from his starting location to his current location. Once the user has finished their run, they press the stop tracking button which returns them to the main page. Now on this main page they can choose to view their run if necessary which will also allow them to edit certain values of their run. E.g. the user can choose whether the activity was a walk or a run as well as allowing them to give the activity a name by which they can recognise it.

Architecture of the application



System components (description of each activity/class/service)

Database Content Provider / Content Provider Contract

The Content Provider Contract class is used to establish a contract between the content provider and all the other classes which need to use it. The class is used to store constant value names which are unlikely to change such as database table field names (e.g. GPS_AVG_SPEED, GPS_DISTANCE). The Contract class also contains definitions for the URI's. Which when used in conjunction with the content provider it allows for an abstractions to the database. Furthermore

DBHelper

THe DBHelper class is used to create the database tables as well as update them as necessary by incrementing the version number.

GPS Service

The GPS Service class had two primary functions, one of them was implementing a location listener and retrieving data from it. And the other one was binding to the main activity and allowing its functions to be called outside of it. The service had two primary functions which were used. One of these functions when called saved data into a ContentValues and uploaded them to the database through the use of a contentResolver and the abstraction provided by the Content Provider.

And the other one was to create a notification when called. This was used by the main activity in order to create a notification upon starting a new run to track as well as cancel it once the map activity came back with an RESULT_OK. The notification flag was however changed in order to stop the default functionality of the main activity being destroyed and now instead the FLAG_UPDATE_CURRENT is used. This simply refreshes the activity without destroying the Main Activity and preventing the previous errors which broke the functionality of my Maps Activity when clicked.

The location listener "location" object was used in order to retrieve information detailing the current run such as latitude and longitude as well the current time. This data was used in order to calculate further information statistics such as average speed, total speed, total distance, current time and total time. This information was updated per every location change detection by the gps. And at each location update this data was packaged into a broadcast and send towards the Maps Activity through the use of of a Broadcast Receiver. The location listener is removed from the update cycle once onDestroy() of the service class is called.

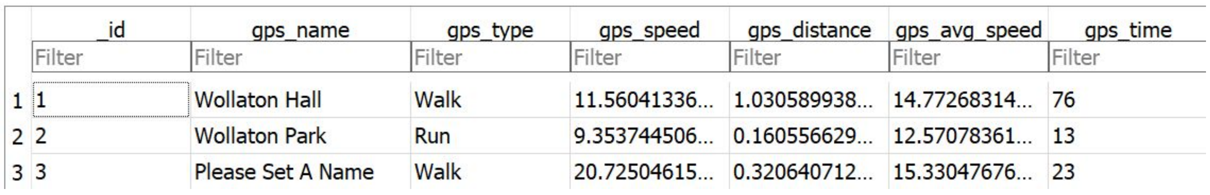
MainActivity
The main activity is used in order to start tracking runs as well as be able to look up previous runs through the use of a listView. Initially when launched it will also display a permission check to the user in order to inform them that the location permission has to be allowed in order for the application to work. The listview is populated through the use of a DataAdapter. There are also 2 buttons which allow the user to sort the runs by distance or time which are created by giving a sort order to the cursor. The start tracking button initiates a connection to the service as well as calls the startNotification() function within it. This is then bound and an Activity For Result is initiated towards the Maps Activity. Upon receiving the results back from the Maps Activity the main activity calls the service saveData() function which retrieves all current results and uploads them to the database as mentioned above in the Service section. Furthermore the service is stopped and the connection is unbound. If this fails the service is also stopped and the connection is unbound on the onDestroy() of the main.

MapsActivity
The maps activity has a registered Broadcast Receiver which is assigned an intent filter to look for. The intent filter looks for specific broadcasts, in this case "com.example.zlat.myapplication.UPDATE_MARKER". Once this broadcast has been received the data included within it is unpackaged and used to display onto text which is present on the Maps Activity screen such as
"Average speed: ? m/s" and is used to tell the user about their current statistics.

Furthermore the activity processes the latitude and longitude values by placing them into a single location, and using that location to include a marker on the map along with lines to indicate the users route for this run. There is also a stop tracking button in this activity which sends back a "RESULT_OK" intent to the Main activity.

View Run Activity
The view run activity loads data of previous runs in order to allow a user to see them. This is done by unpackaging the bundle being sent by the Main Application which contains the "id" of the run. Once the id has been retrieved a cursor is used in order to retrieve the relevant data. The cursor retrieves column fields and they are then set to textViews within the activity. The user also has a switch here in which he can toggle to say if the activity logged was a run or a walk. As well as a name field he can edit in order to give the logged activity a name. This can then be confirmed by clicking the update Run button which uses ContentValues and a contentResolver to update the values in the database. There is also a delete button which uses the "id" to delete the relevant records if the user decides to.

Data (Explanation of database)



	_id	gps_name	gps_type	gps_speed	gps_distance	gps_avg_speed	gps_time
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Wollaton Hall	Walk	11.56041336...	1.030589938...	14.77268314...	76
2	2	Wollaton Park	Run	9.353744506...	0.160556629...	12.57078361...	13
3	3	Please Set A Name	Walk	20.72504615...	0.320640712...	15.33047676...	23

The gps database table contains only information which will be showed to the user. All of the user's running data which is collected through the location manager and calculated is inserted into this table.

Initially there was also a second table which contained longitude and latitude values. This table would record the change every time the Location Manager "location" object detected a change in gps. This table had 2 "id" one which is incremented every time a new record is added into the database. And another one which was only incremented when a new run was added. This allowed for values to be kept for each run which means that further expansion could be added to the code such as displaying the user map when checking previous runs.