Runt - The MDP Running Tracker

James Scully (14304469)
psyjs20@nottingham.ac.uk

Contents

1 Preamble

The project was written in Kotlin to simplify development; namely data-classes, simplified listeners, lambdas and anonymous classes. It does however use Java for the ContentProvider and Database, as the author is much more versed with these types of classes in Java.

The application itself allows for the importing of GPX (albeit very verbose) data, tracking of running and annotating each trip through ratings and comments on milestones throughout the run.

2 Terminology

2.1 Trip

A *Trip* refers to a run; as the user could use this for running/walking/journeys, we decided to name it trip. It is a dataclass found as Trip.kt. It holds the following:

id	-	The ID of the Trip
name	-	Name of the Trip
notes	-	Contains JSON data for various settings of a trip, such as rating, comments on trackpoints, etc. This is used so additional functionality could be added later.

2.2 Trackpoint

Trackpoint refers to the data-class Trackpoint.kt, which stores data on a point during the run, such as:

tID	-	Trip ID it is associated with
seq	-	Sequence number of the point in the run
lat/lng	-	Latitude / longitude coordinate of the point
elev*	-	Unused; elevation at the point.
time	-	Seconds since the Unix epoch

** unused due to permissions needed from Google for Elevations API. Can however be gathered through importing GPX data.*

2.3 Movement

This is an array of **Trackpoints**, which is held by a **Trip** object. This makes it easy to get the first and last of a Trips trackpoints, for estimating journey length, distance and starting / end dates.

3 Activities navigation

Runt contains the following activities:

IntroActivity
This is where we prompt the user of what the apps purpose is, and whether they accept to our permissions that we need; storage access and location.

MainActivity
This acts as the main menu, where users can decide whether they want to start a new run, view their previous trips or exit the app.

ListTripsActivity
This displays the users each trip with data such as how far they ran and in how long. It will initially contain one test trip, but is where the users can find and import GPX data (the Import button on the map).

ViewTripActivity
This is where the user can view their trip in detail; it shows the route they took through Google Maps, the rating they chose for the trip and comments made on each of the milestones in the run. Clicking on a milestone will highlight it on the map.

TrackActivity
This is where the user starts and stops their run, they can view where they are and been on the map, with updates to how fast they are running, how long for and how far.

You can view the Navigation diagram in the Appendix.

4 TrackActivity and TrackService

Two of the main components in the application would be how the application tracks the user during their run, and updates the activity to show this.
The way these two work together are as follows:

The user starts the activity. At this point, we start and bind to the service, but **don't** start tracking the user yet. It is at this point we receive the IBinder object to the service.

Then when the user presses start, we invoke the startTracking() function on the service, which will create a LocationListener that will register location updates and - whilst we're bounded - add Trackpoints to our **activities** trackpoint buffer. This allows the map to be updated for each location update, so the user can see where they've been and where they are.

Once the user navigates away from the activity, in the case that they put it into their pocket or check another app, we unbind from the service and move the service to the foreground (since Android Q, we need to keep a persistent notification for foreground). Each location update from here will be added to the services own Trackpoint buffer.

Once the user comes back to the application, we will bind back to the activity. This removes the service from the foreground, as we do not need it whilst they are using our app.
When this happens, the BOUNDED variable will be updated in the service and once we receive our binder, we retrieve the services buffer (which is then cleared) and we add each new Trackpoint to the activities buffer aswell as the map.

5 Appendix