G53MDP / COMP3018

Running Tracker

Coursework 2

James Thackway
1-12-2020

Project Aim

The aim of this project is to design and develop a simple running tracker. An android application that can be used to track and store data on runs or other movement you do.

I have chosen to focus my application on the idea of activity sessions, when the user wishes to take a run of a specific length and record their timings and speed for each attempt. This would be useful for anyone that frequently takes runs of the same length or time. Therefore, my application has two states of data recording: in-session and non-session. The user must explicitly start and stop a "session" to record the session data.

Design

The application will store the time, duration and distance of each session into a database after completion. The user is also able to rate and add notes to each session. The application will also supply supplementary statistics such as average speed or unit conversion to commonly used units. For instance, while distance can be showed in both metric metres and imperial miles, while speed is generally only thought of in mph.

I have chosen to only collect minimal data from the user since storing information like location or exact route would be superfluous to the principle and would lead to many problems with identifiable and compromising data collection.

This stored session data can be parsed, viewed and edited within the application also. The user can sort the sessions with a wide range of variables and view more in-depth information on them by clicking and expanding any session.

Keeping components separated is paramount for keeping the application clean, I will create a service to do the data collection separate from the activity, which will allow the user to temporarily leave the app or lock their phone while keeping the tracker recording data.

Components

Main Activity

MainActivity.java using layout activity_main.xml

The main activity is used when the user wants to record a session. I included the actual session recording on the main activity as to allow users to quickly start a session upon opening the application. The start session button is the biggest and initial view on the screen, with a simple overview of the session underneath displaying elapsed duration, total distance and average speed in minutes and seconds, metres and miles per hour respectively. From this page the user can also view extant session data using the “history” button

The main activity binds to a service on creation, this does not start a session but prepares the necessary objects for so. The Runner object is also initialised within the main activity as to give it the correct context for use by the location listener later.

List Activity

ListActivity.java using layout activity_list.xml and list_item.xml

The list activity displays a list of all recorded sessions from the database using the style defined in “list_item.xml”. This displays the date, distance, duration and rating in a compact fashion within the page. At the top the user can choose from a total of 10 ordering options. These are lowest to highest or vice versa for the variables: date, distance, duration, rating and speed.

View Run Activity

ViewRunActivity.java using layouts activity_view_run.xml

The view run activity has two primary uses and as such two entry points. The first is to view old session data accessed from ListActivity. Here it will display the data as described by the database and allow the user to alter the rating or add/alter the notes attached. The rating value also has a graphical representation of the rating as it was upon loading the session. The session can be saved, updating the required data, or deleted.

The second use is to inspect data for a session that has just concluded. When the “stop” button is clicked on MainActivity, the user is brought here and the data is displayed as such. Here the user can add a rating and notes and save, or delete the session if it was made in error.

The session can also be exited via the OS back key, which allows the user to cancel the edit while respecting the activity lifecycle.

Service

RunnerService.java

The service is used to handle the ongoing collection of data. This task is removed from MainActivity as to allow the phone to be locked or the application to be minimised and the session can still be recorded. This may be useful for runners since they would not want to keep their phone unlocked throughout a session, and it also allows them to use different applications simultaneously such as a map or music.

Runner

Runner.java

The runner object is where the data on a session is collected and stored. It holds the locationManager and listener objects and is the soul interactor with them. It updates the time/distance once per second and is responsible for incrementing the distance correctly using the old and new location data.

I chose to separate this class from the service class as to allow it to be a direct holding class for a given session. It can be used to store sessions as a purely structural object but also contains the methods to run the collection of data. This means that old sessions could easily be loaded from the database into an instance of Runner and then data collection resumed. This separation allows for greaterflexibility with adding functionality in the future.

Location Listener

RunLocationListener.java

This class implements LocationListener and is used as a listener for the system location services. This is updated here and can be accessed by the runner object only.

Database

Initialised by DBHelper.java

The database is used to store all data on sessions. The only table is the “runs” table with each session represented with a single record.

The fields are as follows:

- _id: The primary key.
  [*INTEGER*]
- date: An automatic timestamp made on record creation.
  [*INTEGER*]
- time: The duration of the session in seconds.
  [*INTEGER*]
- distance: The distance of the session in metres.
  [*INTEGER*]
- speed: The average speed of the session in metres per second. Used for sorting purposes.
  [*FLOAT*]
- rating: The user given rating. An integer from 1 to 5 or null.
  [*INTEGER*]
- notes: 250 characters of user given notes on the session.
  [*VARCHAR*(*250*)]

Content Provider

MyContentProvider.java

The content provider handles reading and writing to and from the database. All database access goes through this class. It uses values defined in the provider contract to define the required database queries.

These are:

- Delete run: Removes the specified session from the database.
- Insert run: Appends a new session to the database.
- Update run: Updates a session in the database. Only the rating and notes fields may be updated.

Provider Contract

MyProviderContract.java

This class defines values to be used by the content provider when accessing the database. This includes table, field and other definition.

Component Diagram

This diagram (figure 1) describes the components as above and their relationships with other components and classes. For activities this means direct user movement while for others it means the communication of data.

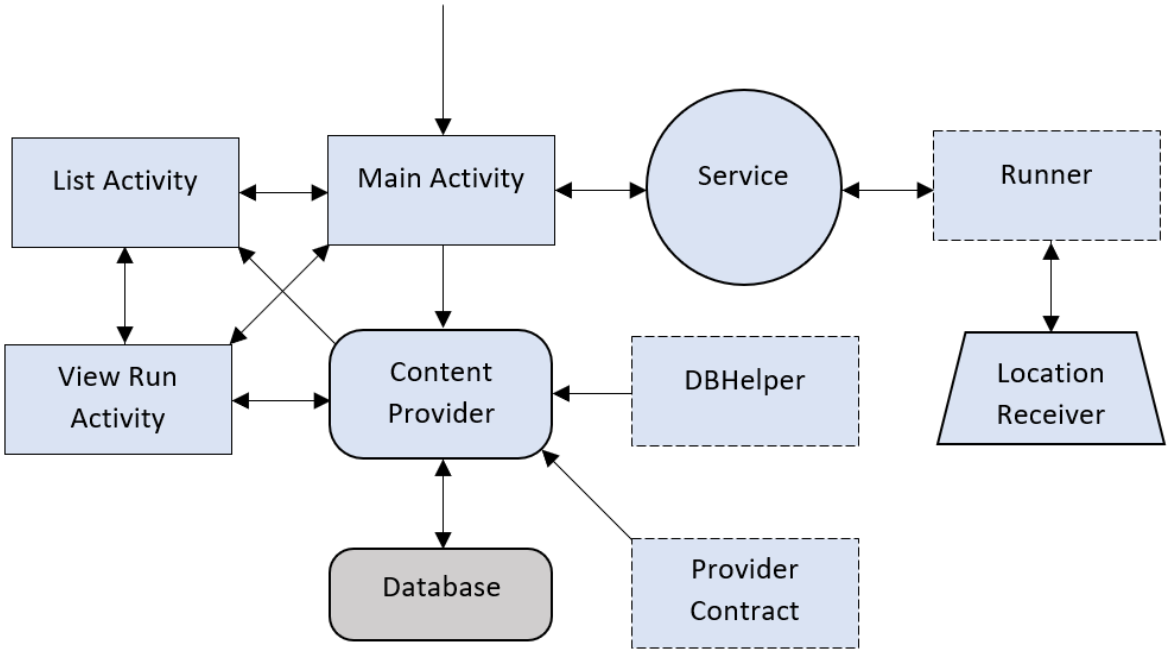

Figure 1: component diagram