## Android Fundamentals Project Self-Evaluation

**Instructions:** Once you’ve completed your Final Project, please respond to the questions below. This is a chance for you to briefly explain to the grader your thought-process during development. Once you are done, include this with the source code and accompanying files you are submitting. Then, give yourself a pat on the back for making a great app!

# Questions about Required Components

## Permissions

**Please elaborate on why you chose the permissions in your app.**

|  |
| --- |
| android.permission.ACCESS\_FINE\_LOCATION  This permission is needed to access location/gps data.  android.permission.INTERNET  android.permission.WRITE\_EXTERNAL\_STORAGE  android.permission.ACCESS\_NETWORK\_STATE  These permissions are required by google maps (SupportMapFragment). |

## Content Provider

**What is the name of your Content Provider, and how is it backed? (For example, Sunshine’s Content Provider is named WeatherProvider backed by an SQLite database, with two tables: weather and location.)**

|  |
| --- |
| My content provider is named TracksProvider and it is backed by a Sqlite database. The database has two tables; tracks and geolocations. The table tracks contains info about a track (name, type, created timestamp and last uploaded to runkeeper timestamp), and the table geolocations contains all the coordinates a track is built up by. |

**What backend does it talk to? (For example, Sunshine talks to the OpenWeatherMap API.)**

|  |
| --- |
| It is possible to upload a track to RunKeeper. I use Health Graph API, provided by RunKeeper, to upload tracks. The upload is handled by an intent service named RunKeeperService. Documentation on Health Graph API can be found here: <http://developer.runkeeper.com/healthgraph/overview>  I use Retrofit (<https://square.github.io/retrofit>) to talk to Health Graph API. |

**If your app uses a SyncAdapter, what is it called? What mechanism is used to actually talk over the network? (For example, Sunshine uses HttpURLConnection to talk to the network, but your app may use a third-party library to do the talking.)**

|  |
| --- |
| N/A |

**What loaders/adapters are used?**

|  |
| --- |
| I use a CursorLoader to load all tracks into my adapter (derived from CursorAdapter). The adapter is used by a ListView. See TracksFragment. |

## User/App State

**Please elaborate on how/where your app correctly preserves and restores user or app state. (See rubric for examples on this question)**

|  |
| --- |
| When an activity is displayed, the same activity appears on rotation.  User text input is preserved on rotation. |

# Questions about Optional Components

Answer the questions that are applicable to your final project

## Notifications

**Please elaborate on how/where you implemented Notifications in your app:**

|  |
| --- |
| When a track is being tracked I create a persistent notification. If the notification is clicked I open up the track detail view (TrackDetailsActivity and TrackDetails\*Fragment). When no track is being tracked the notification is removed. |

## ShareActionProvider

**Please elaborate on how/where you implemented ShareActionProvider:**

|  |
| --- |
| The ShareActionProvider is implemented using a menu-resource and in the code I set the share intent (see onCreateOptionsMenu).  The data being shared is a GPX-file with track name etc. and all coordinates. The GPX-file is created by my content provider (see openFile) and the file is available through the uri …/tracks/{id}/{filename}.gpx. On the share intent I set the GPX-file uri using the key EXTRA\_STREAM. I also set the track name using the keys EXTRA\_SUBJECT and EXTRA\_BODY in case the user wants to attach the GPX-file to a mail. |

## Broadcast Events

**Please elaborate on how/where you implemented Broadcast Events:**

|  |
| --- |
| I use LocalBroadcastManager to send local broadcast events. When a track is uploaded to RunKeeper an event is sent to notify any listeners whether the upload succeeded or failed.  Events are sent from RunKeeperService (see broadcastSuccess and broadcastFailure) and in TracksDetailsInfoFragment I listen for events (see onResume and onPause). |

## Custom Views

**Please elaborate on how/where you implemented Custom Views:**

|  |
| --- |
| N/A |