Manon JACQUET – 3064101

**User Interface**

Explanations:

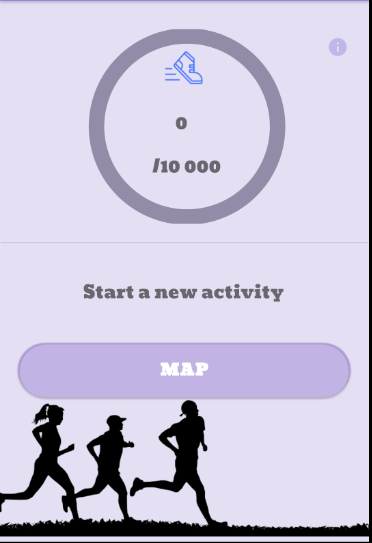
This app is a Tracker app. The purpose of this app is to count your steps and track your activity (running, walking …). At the end of the tracking, it will give you statistics about your average speed or distance covered.

Home Page/ Menu :



This is the first page you can see. When you tap on the screen you have access to the second activity.

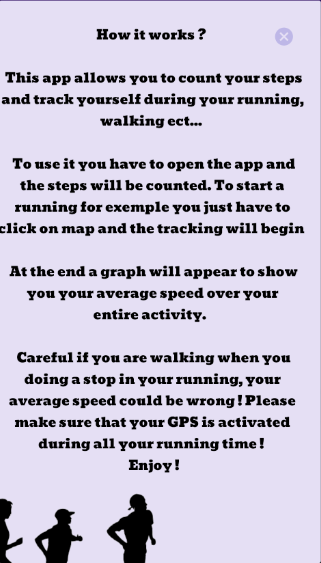
Next page:



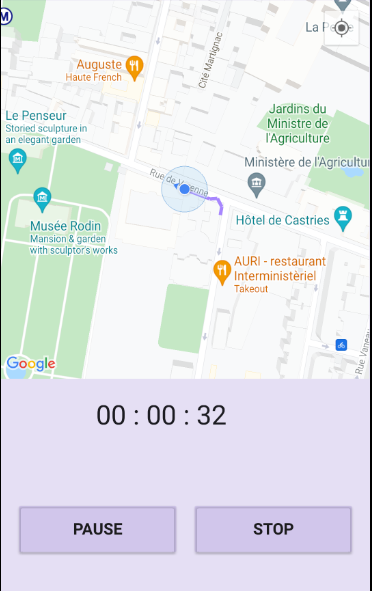
This is the next page of the app: When you are here, your step will be counted.

You can also start tracking with the map button.

When you click on the icon on the upper right corner you have access to information about the app and how it works as you can see on the following image.



Then, when you click on the “map”button, the tracking begin :



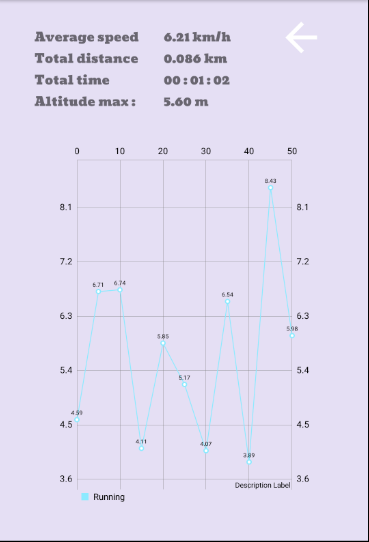
Here you can see the tracker. The user is the blue pointer and the path is drawn behind him.

Under the map you can see the stopwatch. And just below you have 2 buttons.

“Pause”: pauses the time and the location search of the user. It becomes “resume”. Clicking on the new button the time and location search resumes



When you click on “stop”, the timer and location are stopped, and you have access to your statistics.



Here you can see your statistics.

The graph shows your speed versus the time

The arrow points to the activity with the pedometer.

**Documentation**

To design this app, the instructions were to use a pedometer, draw a map that allows the tracking of the user and include then a graph with statistics. I have created a home page with a simple image. This activity is there to help me manage location permission. The location authorization is requested from the user in the following activity, but if the user refuses the location authorization, instead of closing the application, the app returns to this home page. I think it is better (more intuitive) for using the application.

Then you arrive on the page with the pedometer. Here, I manage the location permission, as said above, and the pedometer. To create the pedometer, I have to use a sensor and need therefore to check if there is a good one in the phone (Step counter), thanks to a sensor manager. This sensor will start counting the steps when the activity begins. I used different methods to pick up the steps, such as “*onResume”* (begins the count of the steps)*, “onPause”* (stop the count of the steps) *and “onSensorChanged”* ( here I add the steps to the step counter).

In addition, I linked a progress bar to the step counter. I set its maximum on WHO statistics (10000 steps/per days). The bar progresses with each step of the user. Then, I manage the permission. When you install the app and open the second activity, a popup will ask the user the location permission. If the user refuses, the activity returns to the home page, but if he accepts, this authorization will be saved in the app and permission will not be requested again. We can have access to this permission, thanks to the manifest and the *FINE\_ACCESS\_LOCATION* . Then, you have access to the tracking activity (map). When you click on “map”, a map appears on your screen. It will zoom on your position.

The app can have access to the user location thanks to a *LoacationCallBack, locationRequest* and *a FusedLocationProviderClient.*  It is thanks to the location request that we can manage the time between to location request. Here, there are 5 seconds between each “refresh location”; I choose this time-frame to have a precise location without having an app that drains too much battery ( compared too one refresh/ second for example, which would use more battery ). Thanks to the merged location, the location is updated and the location recall retrieves the user’s location. Each location is collected in a list (class listpoint). Thus, I can get back the user’s location in real time. This activity is a *map Activity*. I can access the location and map because I set up a *google key* (after creating a new Google project and getting the key for my project, I got access to the google map). Then I found a way for the camera to track the user’s point. I did this, through the different points I capture. I created a *Latlng* and moved it to each new location point, then zoomed the camera to that new point. Thanks to this saved point I can then create a *polyline* (line made behind the user on the map, as he walks).

The next thing that I had to implement was the timer. the map is a *fragment,* I had different problems to solve. When I wanted to implement a Button start and Stop on the map *a findViewbyId* does not work. To prevent this trouble, I made an *OnClick* method. With the id of the button, I can stop and start the tracking and the timer. When the user wants to stop, he clicks on the “stop” button and his statistics will appear in a new activity and a graph can be seen in the meantime. To generate this graph, I have downloaded a new dependency ( that can make a draw thanks to an implementation that I find on *github*). To plot the graph, each location point in the list is recorded. The distance is calculated between each point as well as the user’s speed, with the mathematical formula : speed = distance / time. The calculated speed for each time-point allows then to generate the graph. In parallel, based on these different data, the average speed, the maximum altitude and the number of kilometers traveled, are also indicated.

When you want to return to a new activity, click on the arrow.

**References**

**Tutorial Video:**

# Android Studio Tutorial - Build a GPS App:

# <https://www.youtube.com/watch?v=_xUcYfbtfsI>

# Develop simple Step Counter in Android Studio

# <https://www.youtube.com/watch?v=CNGMWnmldaU>

# How to implement Google Maps in your Android App.

# <https://www.youtube.com/watch?v=7TIAT5zlrmc&t=214s>

# Android Simple GPS tutorial

# <https://www.youtube.com/watch?v=Xb0DTCR1H0s>

# How to add Custom Marker in Google maps in Android

# <https://www.youtube.com/watch?v=26bl4r3VtGQ>

# Introduction, download Graphview library, create empty graph : Android Graph View Tutorial

# <https://www.youtube.com/watch?v=KRUSdhfWLPI&list=PLFh8wpMiEi88ojfNpavGpMB0dtP4mvEqa>

# Circular Determinate ProgressBar with Background and Text - Android Studio Tutorial

# <https://www.youtube.com/watch?v=YsHHXg1vbcc&t=772s>

# Others :

# API Google Cloud Project

# HTML Color : <https://htmlcolorcodes.com/>

# List : <https://www.journaldev.com/31869/java-list-remove-methods-arraylist-remove>