# Trakr - location recorder

**GitHub Username**: vokod

# Trakr

## Description

Trakr is a simple App to record your position and path for outdoor activities like walking, hiking, running, biking, boating, motorbike riding, driving or even flying.

You can record all your tracks, view them, analyze the elevation profile, export them as GPX file, and synchronize between your devices.

## Intended User

Trakr is useful for anybody who wants to review the geographic details of a certain activity. The possibility to export recorded tracks as GPX files makes it a great substitute for sports-tracking apps - with less power consumption.

## Features

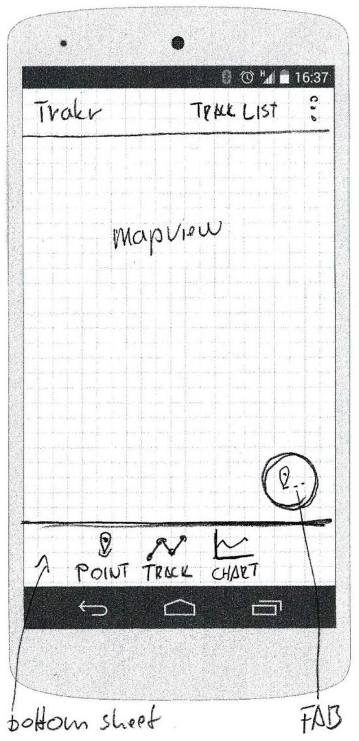
- Records geographics position and altitude.
- Shows the users position and track on a map.
- Shows relevant information about currently recorded track, like GPS coordinates, maximum speed, average speed, elapsed time, height and other useful metrics.
- Shows speed graph
- Shows altitude graph
- Previously recorded tracks can be analyzed on a map, with speed and altitude graphs.
- Accuracy settings can be tailored to suit your needs.
- Export tracks as GPX files.
- Synchronise recorded tracks across multiple devices.

## User Interface Mocks

### Screen 1 - recorder screen with bottom sheet collapsed

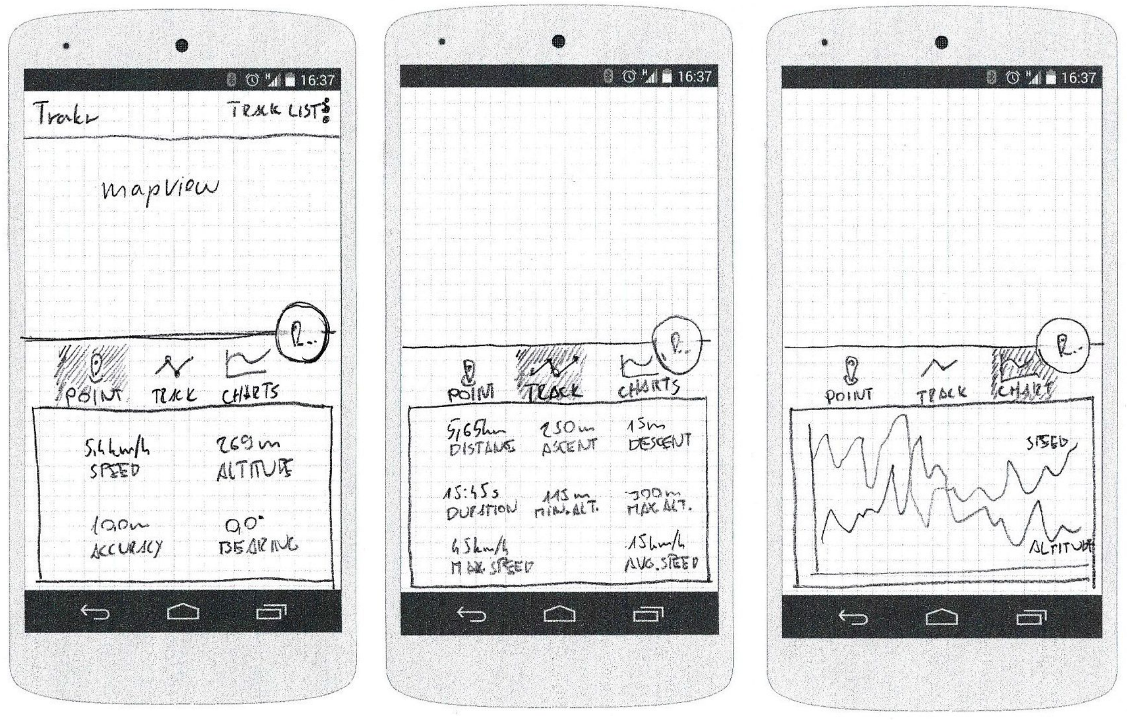
The toolbar contains an overflow menu with an always showing item: Track list. The menu contains another item: Settings.
The screens main feature is a mapview. The user’s location is in center.

The bottom sheet has three pages in a viewpager setup. At its collapsed state, only the pages names/icons are visible.
Above the bottom sheet is a floating action button. Tapping it will start a recording. Tapping it again finishes the recording. During recording the FAB animates.



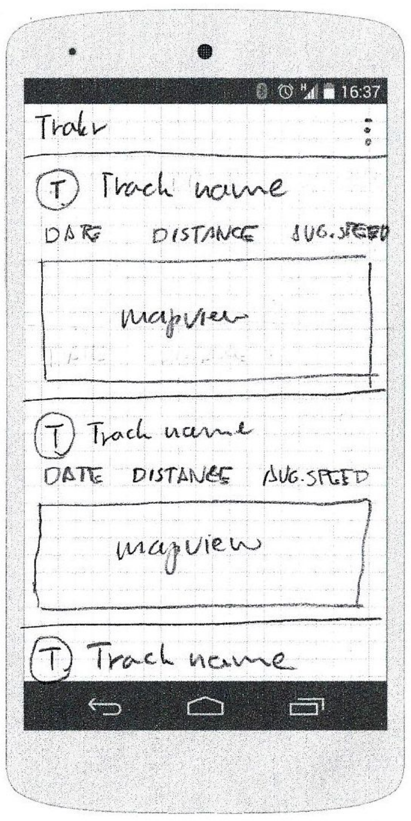
## Screen 1 - recorder screen with bottom sheet expanded

The three screen represents the three pages of the bottom sheet - Point (actual location), Track (data of the ongoing recording) and Charts (speed and elevation charts of ongoing recording) (other details like toolbar are left out from the drawings).
Track and Charts pages are empty when not recording.
When recording a polyline is drawn in real time, showing the user's location and track.



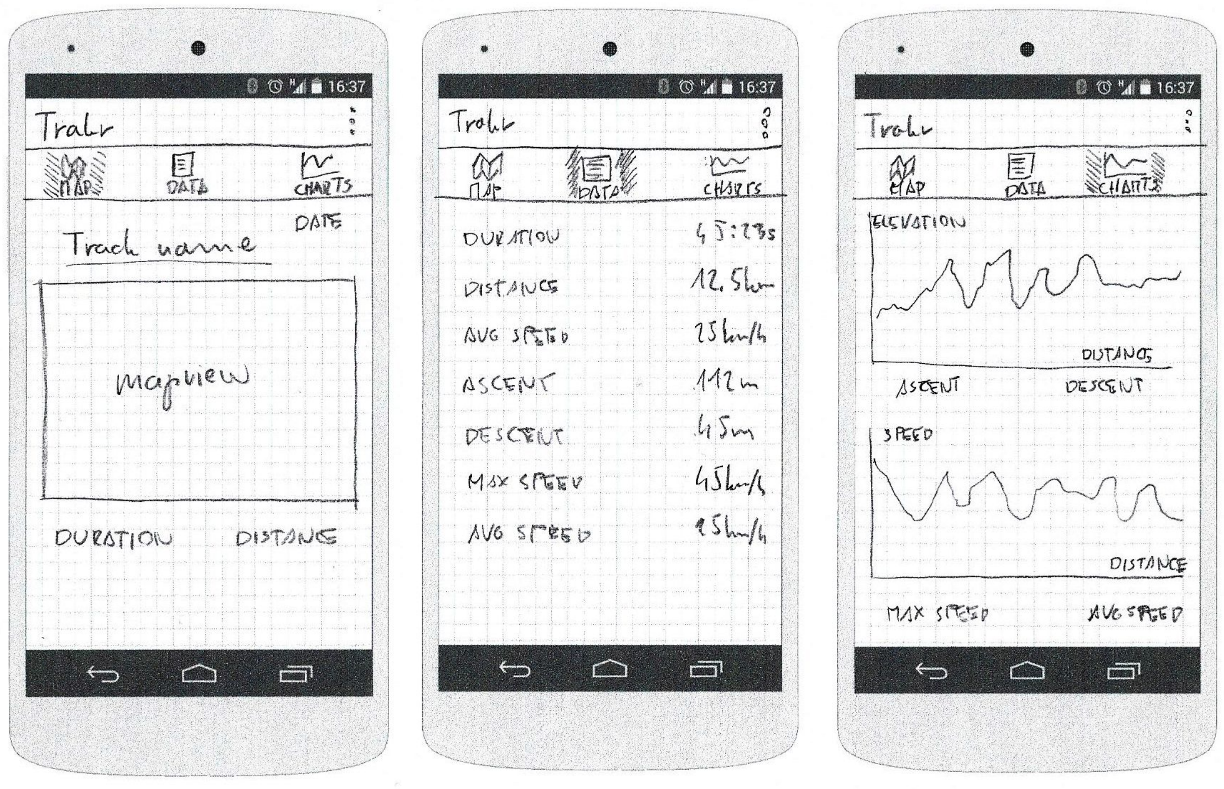
## Screen 2 - Track list

List of the previously recorded tracks.
Track items are shown in a Recyclerview. Each item contains basic data and a small mapview with the polyline of the track.



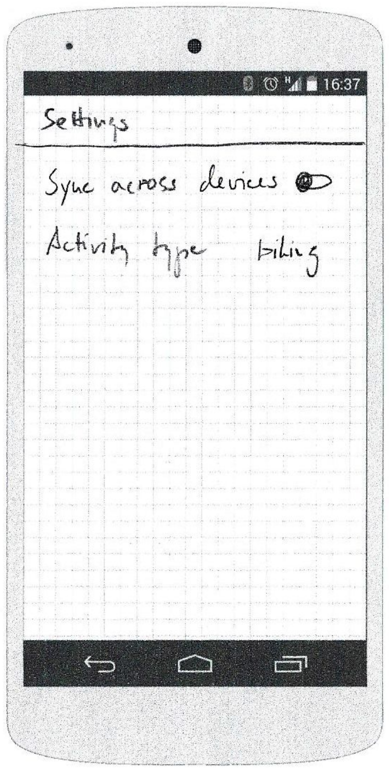
## Screen 3 - Track details

Detail view of a previously recorded track.
Data is shown in a tabbed interface with three pages. First is for a mapview (and basic data), second is for detailed data and the third is for elevation and speed charts.
The three sketches show the three pages
An overflow menu contains two options: Delete track and Export track



### Screen 4 - Settings

There are only two settings.
If the user switches on the “Sync tracks across devices” setting, she/he is transferred to a login page (Firebase UI login page) where the user have to log in/ register.
Activity type setting has options like walking, running, biking, driving etc. This has an effect on the track recording frequency and accuracy settings.



## Key Considerations

### How will your app handle data persistence?

Each track can consist of multiple (possibly thousands) trackpoints. The app will record each trackpoint in a content provider. The content provider will consist of two tables, one for the tracks and one for the trackpoints. Saving is continuous during recording.
Synchronisation between devices will be done with Firebase Realtime Database but it will be optional. If the user choose to log in to the app (with Firebase Authentication) her/his tracks will be saved to Firebase Realtime Database when the user finishes the recording. On every start, the app checks if there are unsaved tracks (tracks not uploaded to Firebase) or unloaded tracks

(tracks present in Firebase Realtime Db, but not on the device) and synchronises them. If the user chooses not to lo log in, data will only be saved to the content provider.

### Describe any edge or corner cases in the UX.

### Describe any libraries you’ll be using and share your reasoning for including them.

- MPAndroid chart for charting.
- Firebase UI android for Firebase authentication
- Firebase for data synchronisation between the user’s devices.

### Describe how you will implement Google Play Services or other external services.

Play services maps and Play services location will be used for location recording, and location displaying in a mapview.

## Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

### Task 1: Project Setup

- Configure Android Studio and Github
- Setup Google Play Services project and acquire Google Maps API key
- Setup Firebase project

### Task 2: Implement UI for Each Activity and Fragment

- Build UI for Recorder Activity
- Build UI for TrackList Activity
- Build UI for TrackDetail Activity
- Build UI for Settings Activity

## Task 3: Content provider

## Task 4: Build track recorder service

## Task 5: Build Firebase integrations

- Create Realtime Database schema
- Build automatic sync service

---

**Submission Instructions**

- After you've completed all the sections, download this document as a PDF [ File → Download as PDF ]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"