Trakr - location recorder

GitHub Username: vokod

Trakr

Description

Trakr is a simple app to record the user’s geographical position and path during outdoor activities like walking, hiking, running, biking, boating, motorbike riding, driving or even flying.

Users can record tracks, view them, analyze the elevation profile, export them as GPX file, and synchronize between 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.

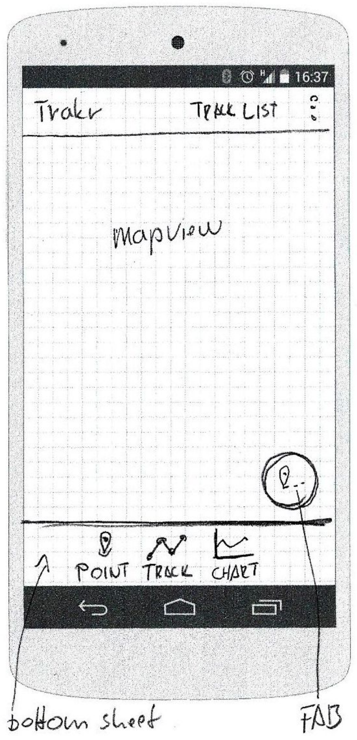
Features

- Records geographical 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

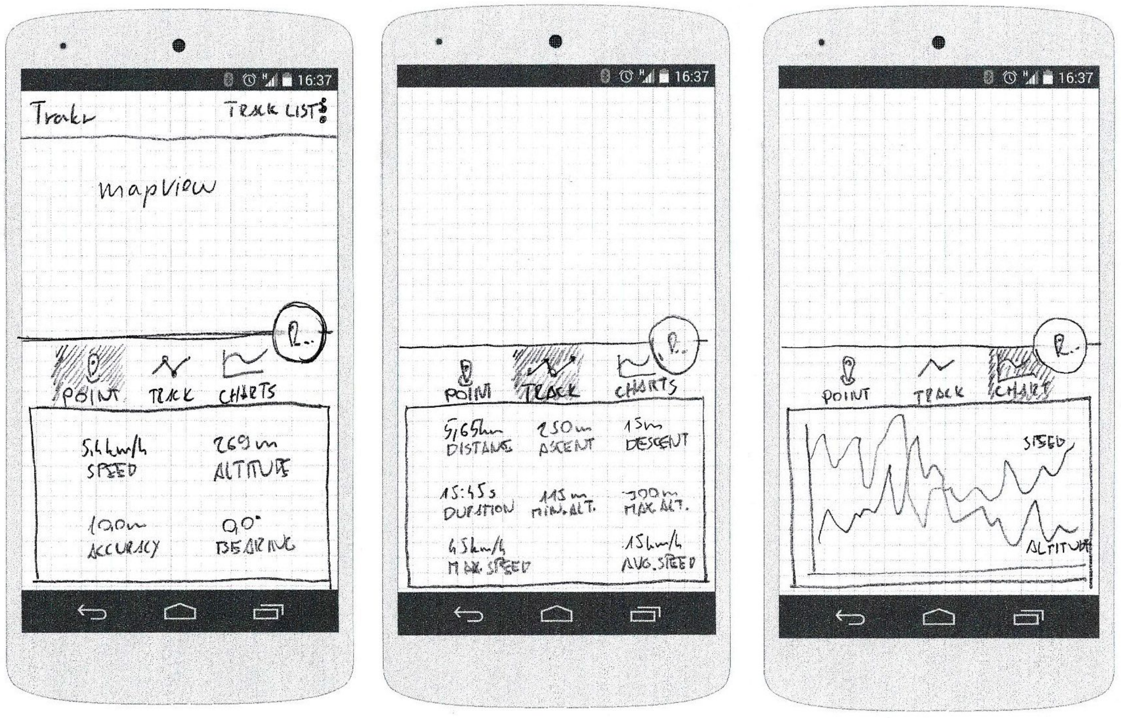
Screen 1 - recorder screen with bottom sheet collapsed

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.



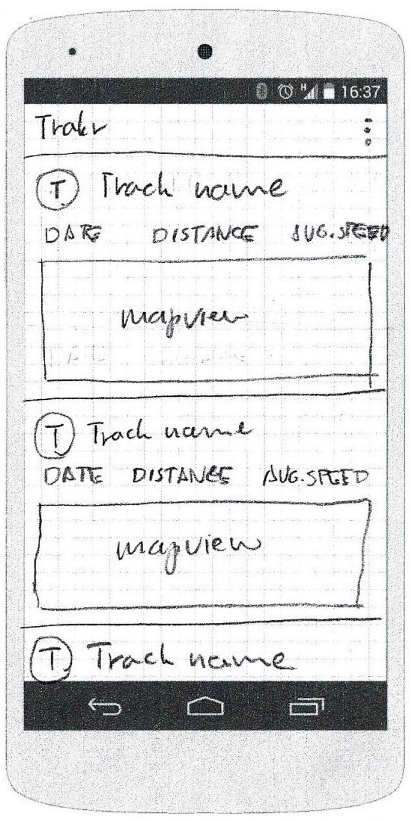
Screen 1 - recorder screen with bottom sheet expanded

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.



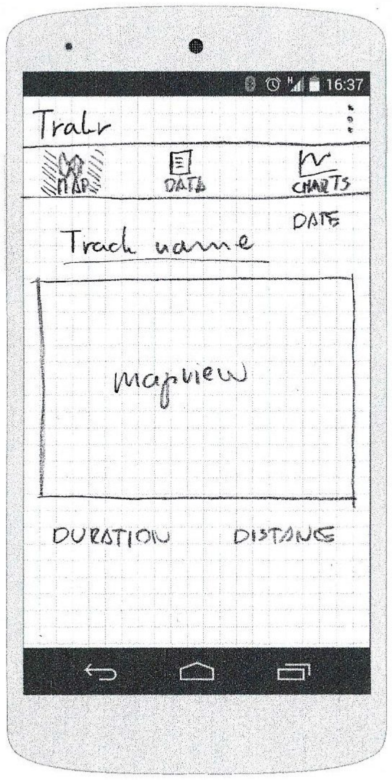
Screen 2 - Track list

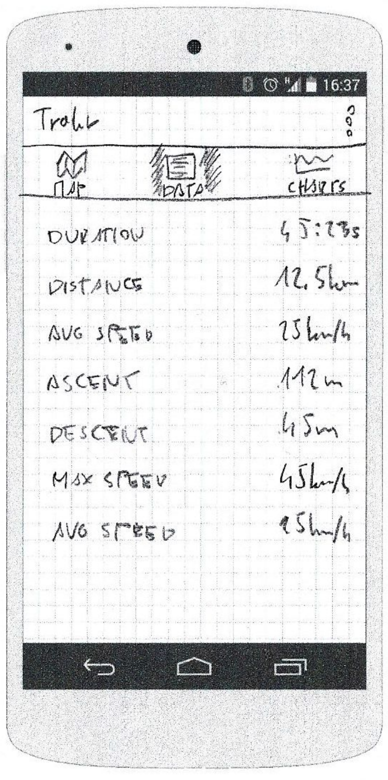
List of the previously recorded tracks.
Track items are shown in a Recyclerview. Each item contains basic data and a small mapview with the polyline of the track.

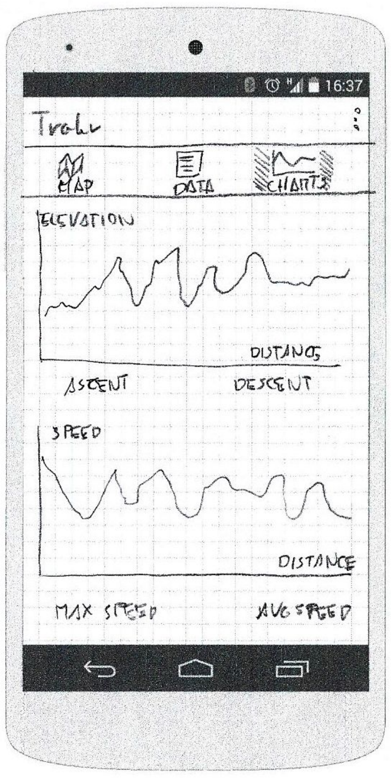


Screen 3 - Track details

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

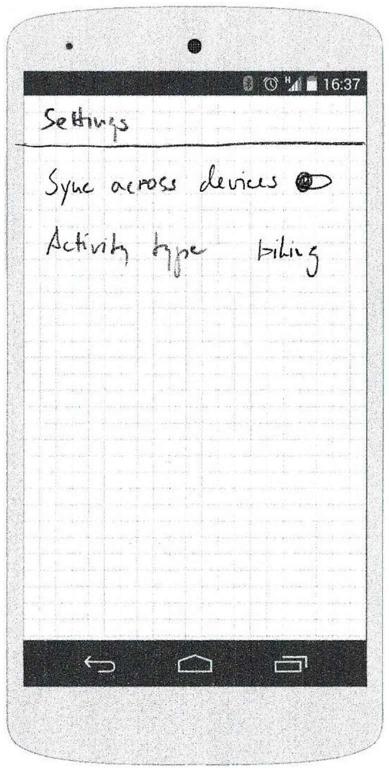






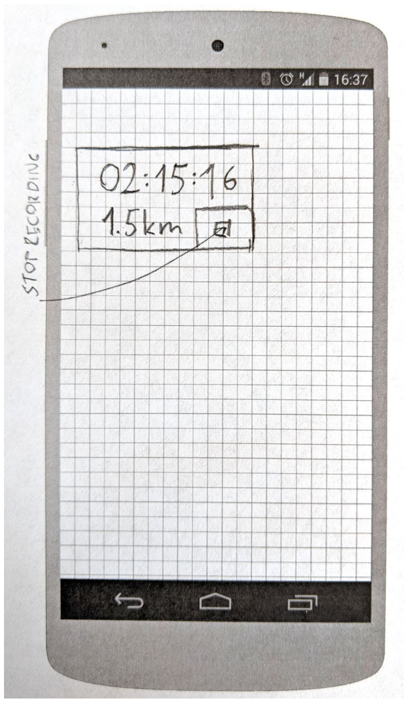
Screen 4 - Settings

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.



Screen 5 - Home screen widget

The widget shows basic info about the track currently being recorded and has a start/stop button.



Key Considerations

The app will be written in the Java Programming Language, using Android Studio 3.1.2, with Gradle 4.4, with Android Gradle Plugin 3.1.2.

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.

- Track recording cannot start if the user did not gave location permission to the app, or the location settings are inadequate. The app shall handle it gracefully, asking for the permission if not granted, and drawing the user's attention to the location settings (if not adequate), every time the app starts.
- If there is no network connection in the time of synchronisation, that might result in unexpected behaviour (the user might not be able to see tracks recorded with another device, or might not be able to upload recently recorded track to Firebase). The app signals this to the user with a toast or a snackbar, but as this is not an essential feature, won't halt the app's normal run. Instead it will try to finish the synchronisation on next app start or when the network connection is adequate (will be decided later during development of the feature).

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

- MPAndroidChart 3.0.3 for charting.
- FirebaseUI 3.3.1 android for Firebase authentication
- Firebase-database 15.0.0 for data synchronisation between the user's devices.
- Firebase-authentication 15.1.0 for user authentication

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

Play services maps 15.0.1 and Play services location 15.0.1 will be used for location recording, and location displaying in a mapview.

Design considerations and resource management

- The app won't use rich imagery. There will only be icons for which the app will use XML drawables.
- As a lot of similar data will be shown in various places throughout the app, custom views will be built. Most of the data to be shown consist of a quantity, a unit and a title or icon, but the importance (or relevance) of the data is different. Different styling will emphasize these differences.
- Styles, colors and strings will be extracted to XMLresource files.
- Icons and other visible elements will have content descriptions to support screen reader usage. Data on screen will be grouped to support single announcement while accessed with screen reader.
- The app will implement a color theme with sufficient contrast.

Required Tasks

Task 1: Project Setup

- Configure Android Studio and Github
- Setup Google Play Services project and acquire Google Maps API key
- Setup Firebase project
- Setup Play services,Firebase and MPAndroidChart Gradle dependencies

Task 2: Content provider

- Design data structure for Track and Trackpoint tables.
- Create dbHelper for database and contract for provider.
- Test content provider.

Task 3: Design and implement UI for RecorderActivity

- Implement bottom sheet with viewpager and tabinterface.
- Design and implement fragments for Point, Track and Chart tabs.

Task 4. Build track recorder service

- Track recorder service shall be a bound service that runs in the foreground always providing a notification area notification.
- Design activity-preferences like walking/running/biking/etc. Specify accuracy settings and trackpoint-record-interval.
- Thoroughly test track recorder service with various accuracy/power consumption settings.

Task 5. Implement UI for TrackListActivity

- Implement recyclerview with with cursorloader.
- Design item for recyclerview.

Task 6. Implement UI for TrackDetailActivity

- Build UI for TrackDetail Activity.
- Implement viewpager with tabbed interface.
- Implement overview, data and charts fragments.

Task 7. Implement UI for SettingsActivity

- Build UI.
- Implement settings persistence (activity-preference, sync-status).

Task 8: Build Firebase integrations

- Create Realtime Database schema.
- Build automatic sync service. The sync service will be realized with an IntentService, as it is started on a per request basis (it will run on app startup and on track recording finish).

Task 9. Build widget

- Build widget UI.
- Implement widget logic.