How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

GitHub Username: msgeorgem

GeoTracker

Description

GeoTracker tracks your daily movement activities on your command. GeoTracker uses your Smartphone GPS module to track your walks, runs, rides, drives or even flights. You can choose predefined activity or create a new one and track yourself in the way you like.
Free version of GeoTracker measures distance, time, speed and of course tracks your geographic position. You can share your results with your friends via facebook, instagram.
Paid version of GeoTracker offers free version functionality and some additional functions like No ads, downloading track to a csv table version.

Intended User

GeoTracker is for anyone you wants to track their movement activities and share them with friends.

Features

List the main features:

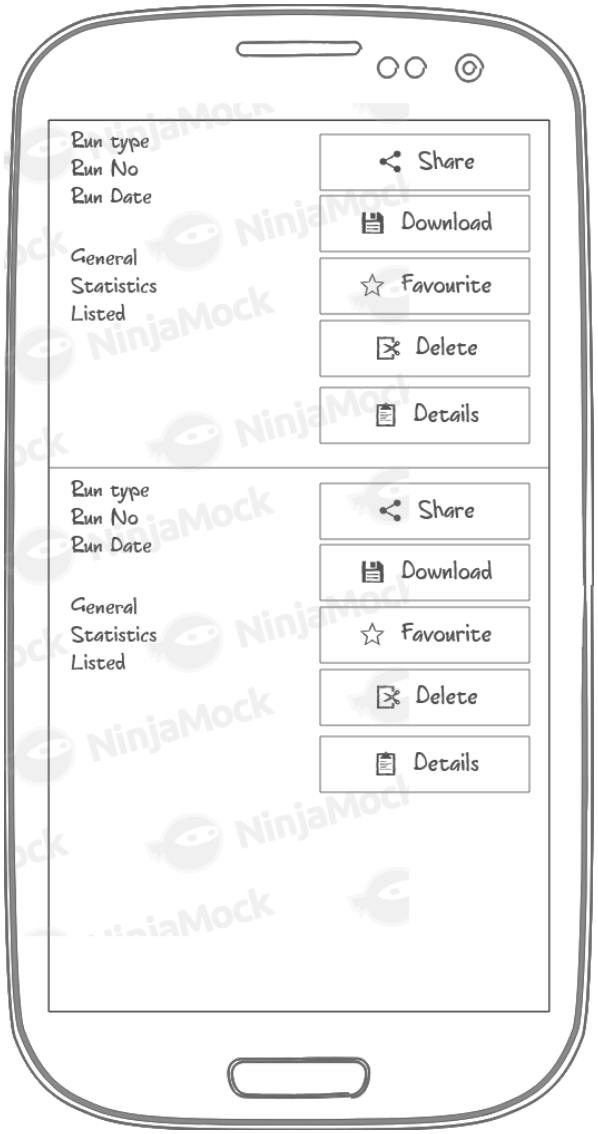
- Tracks users activity
- Saves tracks in local database
- Shares tracks with friends
- Maintenance of saved tracks(deletion, downloads)

User Interface Mocks

Screen 1

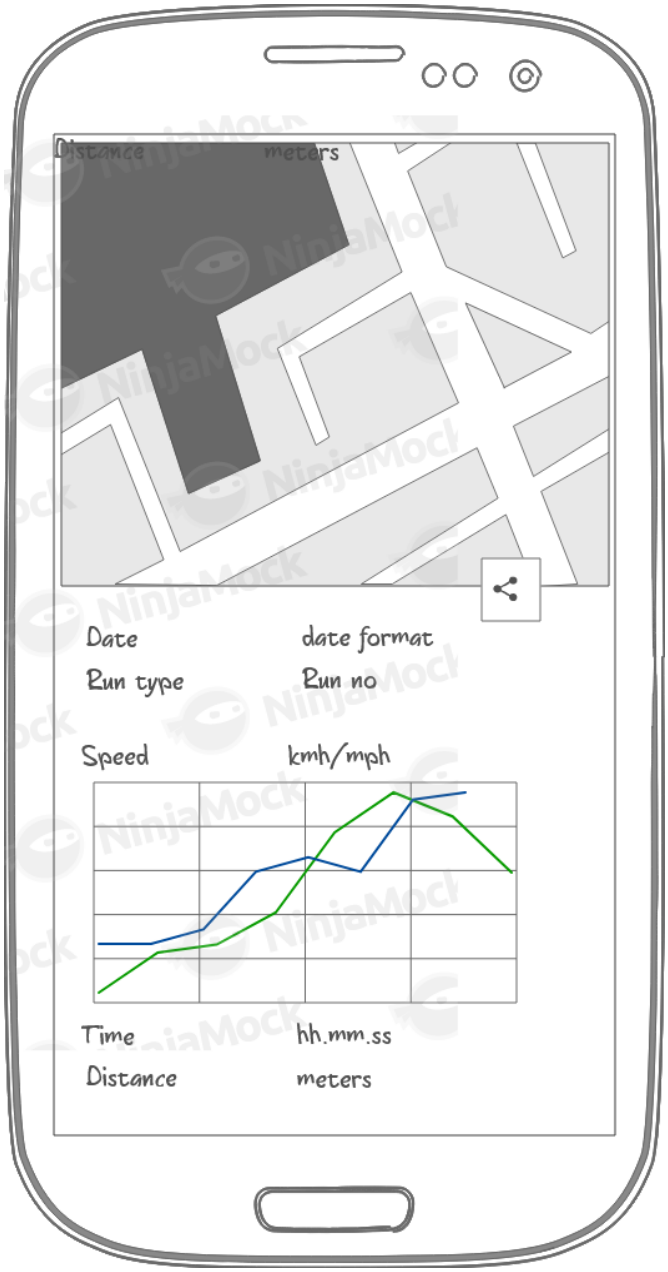
Main screen to start tracking

Screen 2



List of tracks saved after tracking

Screen 3



Details of the recorded track with map and additional statistics

Screen 4

Settings drawer

Widget will have remote control functions

Key Considerations

How will your app handle data persistence?

Sql Database with Content provider

A **corner case** may appear when user records a lot ot tracks. For instance setting WRITE/SAVE(insert in SQL DB) on 5 seconds may produce thousands if not millions of lines in table. Such huge table will cause “lazy” list on UX therefore app will have at least two tables in SQL database:

1. First table will be gathering all recorded data
2. Second table will be gathering only general data for each track.
3. AsyncTask will be used for saving in database
4. To display track list, loader and recycler view will be used. Track list will use second SMALLER table gathering only general track data(track number, total time, total distance)
5. When user decides to see the details of the recorded track the detail activity will get the data from the first BIGGER table but only for the requested track.

App will also use preferences to implement settings.

Describe any edge or corner cases in the UX.

1. On the main screen user chooses the kinds of movement to be tracked.
2. When tracking is started the main values like speed, time, distance are shown.
3. Tracking can be started from the main screen or widget(option)
4. Tracking can be stopped from the main screen or notification bar or widget(option)
5. (Option) tracker will have a timer stopping the tracker
6. Notification bar will show time elapsed and distance(option) (**corner case**: if the user forgets about the running app the notification will appear after specified time)
   To utilize this IntentService/Service will be used
7. (Option)Widget will show time elapsed, distance, average speed
8. When user exits running app the notification appears that the tracker is running.
9. User can stop the tracker from notification bar or go back to the app .
10. (Option) As an option there will be a widget having similar functions as above mentioned notification bar
11. In the list of tracks user will have options:
    - to open detail activity with map and the track is drawn with some statistics
    - to delete the track (**corner case**: if the user clicks on a delete button by accident a window will appear if he really wants to delete the track)

- to download the track in csv table (paid)
- to make track favourite
- to share the track
- (option) add background photo

12. Preferences will have option to switch themes between light and dark:
    - (this will help to be more readable/accessibility reasons also
    - Use of themes
13. App keeps all strings in a strings.xml file and enables RTL layout switching on all layouts.
14. Still not decided of the final number of functions, there are a lot more in my head!

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

App is solely written in Java
App utilizes:

- Android Studio 3.1.3
- gradle:3.1.3
- com.google.android.gms:play-services-location (tracking activities) 11.6.2
- com.google.android.gms:play-services-maps (Maps with polylines to present the recorded track) 11.6.2
- com.android.support:preference-v7-26.1.0 (implement settings)
- com.android.support:recyclerview-v7-26.1.0 (implement smooth list display)
- com.android.support:appcompat-v7-26.1.0
- com.android.support:design-26.1.0
- Com.android.support:animated-vector-drawable-26.1.0

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

I will be using location services

Next Steps: Required Tasks

Task 1: Project Setup

1. Create a database. Database with have at least two tables
    - Main table collecting all GPS reads
    - Helper table collecting main track data
2. Create a track adapter using cursor recycler adapter to present list of track in the fragment.
3. Implement location services to record and save in a database
4. Build Main activity, Detail Activity and corresponding fragments (final look not know yet)
5. Build notifications and widget(option)

Task 2: Implement UI for Each Activity and Fragment

List the subtasks. For example:

- Build UI for MainActivity
- Build UI for Detail activity
- Build UI for list of tracks
- Build UI for widget(option)

Task 3:Build settings, sharing options, notifications

- Build settings activity
- Build notifications
- Build sharing options

Task 4: Make app material

- Polish the App

Add as many tasks as you need to complete your app.

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**"