Assignment 1: Live positioning view

Introduction

So far, you have developed apps in Android Studio that focused on writing, sending and displaying text, viewing EMF sensor and Motion sensor data and monitoring battery status. In this first assignment, we are going to experience app development in the industry with an open source projects to improve a Positioning app developed by last year students. The project is available on github (https://github.com/openpositioning/DataCollectionTeam2) and we would like for you to follow a gitflow workflow to add a live view of the Position of the user using outdoor maps (based on the maps SDK) and indoor maps (ground overlay)

The objectives for Assignment 1 are the following:

1. **Show the position of the user as it collects data using the "Outdoor" map types (https://developers.google.com/maps/documentation/android-sdk/overview):**
    a. **Show the current position and direction of movement in a map using the normal (MAP_TYPE_NORMAL) and satellite (MAP_TYPE_SATELLITE) map types.**
    b. **Show the PDR trajectory/path of the user as it walks.**
    c. **Allow the user to change the map type (optional)**
    d. **Show the received GNSS positions and its positioning errors (optional)**
2. **Show indoor information for the target buildings as the user walks in their boundaries.**
    a. **The buildings are:**
        - **The Nucleus building,**
        - **The Noreen and Kenneth Murray Library**
        - **The Fleeming Jenkin Building (optional)**
        - **The Faraday Building (optional)**
    b. **Switch to an indoor map (we recommend to use GroundOverlay) as the user walks inside the boundary.**
    c. **Allow the user to switch the floor displayed for the building**
    d. **Show an indication of available indoor maps (optional).**
    e. **Automatically change the floor using the barometer information (optional)**

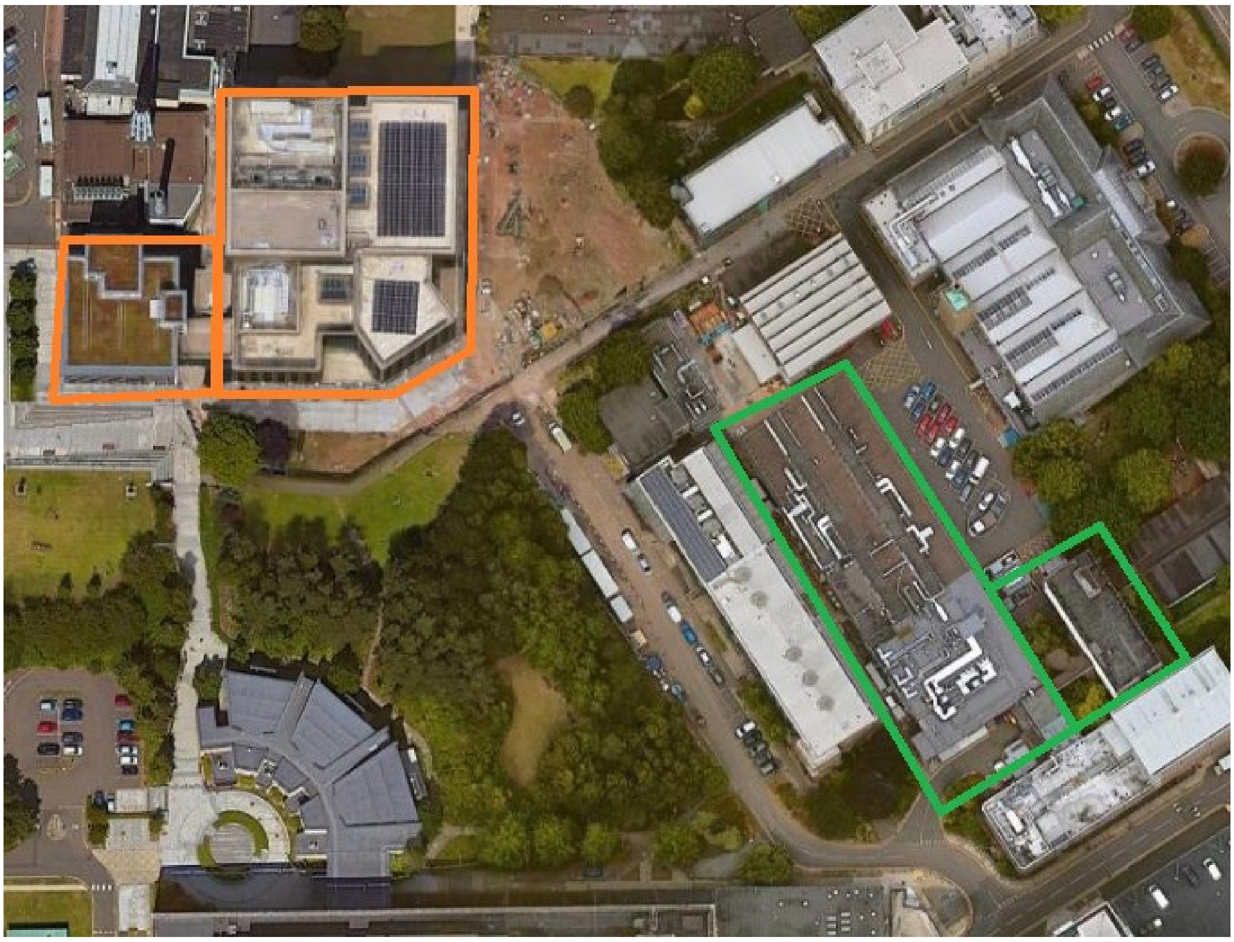


Figure 1. Areas of interest

Assignment 1 Marking Guidelines:

Sr. No.	Criterion	Weightage
1	Demonstration of App Features	35%
2	Code Quality (Reusability, adoption of modular approach, good use of comments, proper alignment, etc.)	30%
3	Documentation (User's Guide & Programmer's Guide)	35%

We would like for the code to be kept in git to keep track of the work of each student, and to use a gitflow workflow to create individual features/merge requests that if approved in the open source project could be integrated adding you to the list of collaborators and showcasing your work for future employers. For a reference of gitflow
https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Important Note: You are to develop the application by utilizing the android studio version, SDK version and Gradle version indicated on Learn for consistency (Course Information > Guidelines for Installing Android Studio on your PC).

Assessment 1
Embedded and Wireless Systems
Demonstration of Application Features
Guidelines

These are the guidelines for your application demonstration. You are to upload a 5-10 minute video demonstrating the working of your application. You may choose to make a split-screen video with one side showing the movement of the user and the other side showing the response of the application.

Demo Video Assessment

The demo video assessment marking criteria is as follows:

- Demonstrating key features of the application addressing its key and distinct features.
- Addressing any distinct features that set the application aside from others developed by researchers or available in the market.
- This should not last more than 5-10 minutes. Imagine presenting your application to a customer or an investor.

Assignment 1 – Documentation (Programmer's & User's Guide)

You need to submit your code and apk through the Learn system. Documentation also must be submitted for this assignment which would include a user's guide and a programmer's guide. The aim of the user guide is to provide instructions to the user on how to use the app and details on the different features in the app. You are encouraged to also demonstrate how your app is different from those already in the market (in the User's guide). The aim of the programmer's guideis to explain the functionality and code implementation of your live positioning view improvement to other android programmers so that they could understand the working principles. You are encouraged to also include references of what has been done in literature and how your app is different from that (in the Programmer's guide).

Programmer's guide

Following bullet points should be covered and explained in your document regarding the new features:

1. Application Interface and functionality introduction (20%)
2. How to add indoor maps to the interface (25%)
3. User information used to achieve functionality of the app (25%)
4. Principal methods/listeners implemented in the code (20%)
5. Elaborate on coding style (e.g. commenting, modularity, naming conventions used, etc.) (4%)
6. Extra features (if applicable) and its realization should be explained for marking purpose only (4%)
7. References (if any) (2%)

User's guide

Following bullet points should be covered and explained:

1. Details of the core features that are in the app and what they do (include screenshots) (50%)
2. Instructions on how to use the new features of the app and how to navigate them (include screenshots) (50%)

General rules for the programmer's and user's guide:

1. Each guide should be contained on **two pages** of A4, flexible layout, within reason.
2. Any number of words, but font size not smaller than Times size 10.
3. Each guide cannot exceed two pages of A4, anything beyond will be discarded before marking. No title pages, no appendices will be looked at.
4. Graphs / Charts / Tables can be incorporated – but EVERYTHING has to be contained on two pages of A4.
5. Word processed documents most welcome, equivalent word-processed forms acceptable.
6. The user's guide must be written in layman's terms for users to understand.

Hints on Getting Started

This document should you give you a general idea on how to get started. For this assignment you will base your work in an open source data collection app available at https://github.com/openpositioning/PositionMe, where you will aim to add features to the project.

In this document, you will learn:

- Project setup
- Adding maps
- Adding markers and lines
- Adding images to the map

Project setup

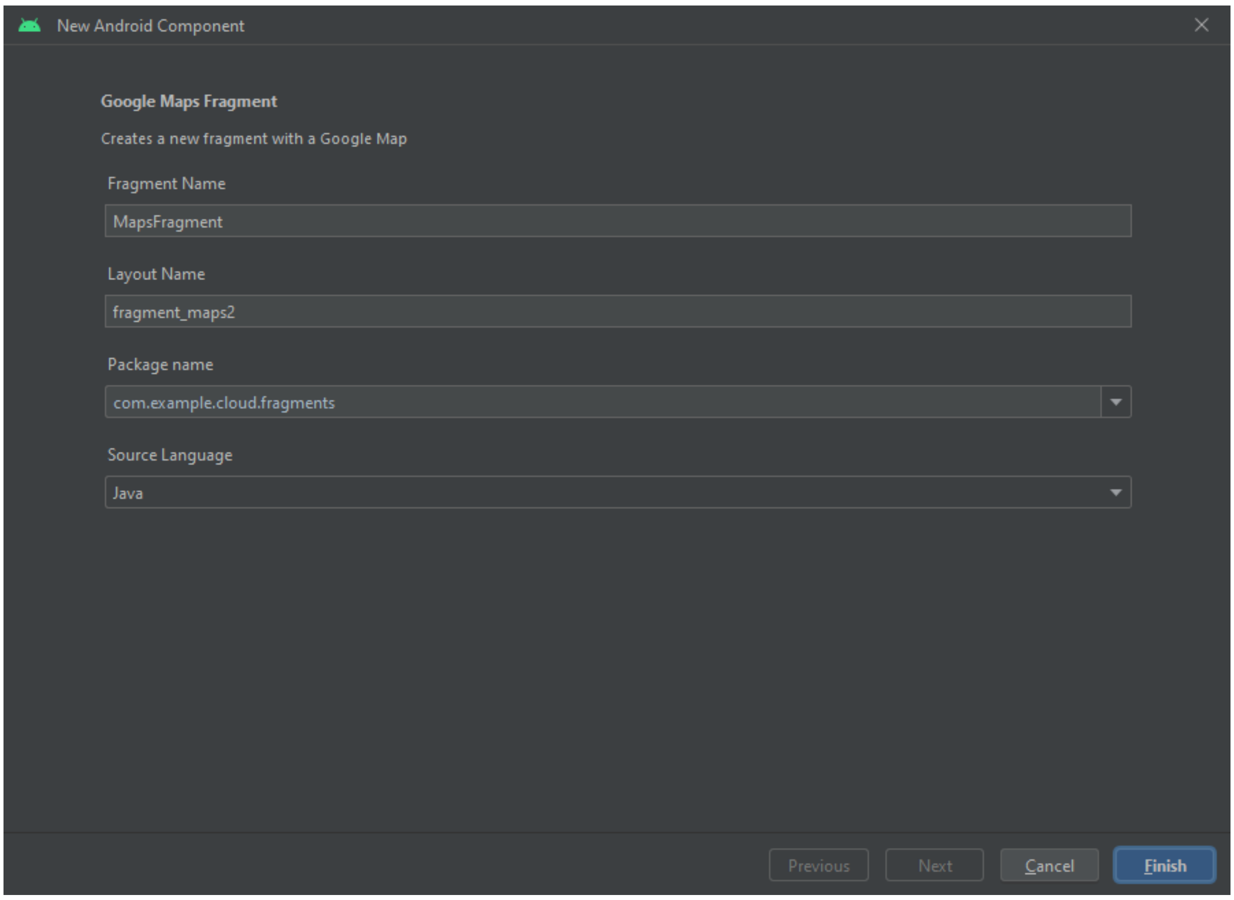
If you haven't already, we recommend you to create an account in github (you will need if you want to push code to the repository) and:

- Clone the repository (git clone https://github.com/openpositioning/PositionMe.git).
- Open the project with android studio.
- Obtain an API key for Google Maps (https://developers.google.com/maps/documentation/javascript/get-api-key)
- Signup to open positioning and obtain an api_key (https://openpositioning.org/docs)
- Modify the secrets.properties file to replace the values of:
    - MAPS_API_KEY: google maps API key
    - OPENPOSITIONING_API_KEY: user's API key from openpositioning.org
    - OPENPOSITIONING_MASTER_KEY: Master API key (ewireless)
- Build your project
- Remember not to push changes with your API keys as you will expose them to the public, ignore changes to the secrets.properties file or remove it form the index for changes detection (git update-index --skip-worktree .\secrets.properties)

Adding maps

There are several options to add a map to an android app, but one of the easiest is through the Maps SDK from google. I recommend you to have a look at the official guide (https://developers.google.com/maps/documentation/android-sdk) and in this document we will present some useful features for our task.

Once you have set the google maps API key in the app you can add a **SupportMapFragment** object to the activity that will handle the map. The project already has 2 statically added fragments (fragment_correction and fragment_start_location), you could reuse the start_location to display the position and direction or you could add a new layout fragment. To add a new fragment, you can right click on the layout folder, select New > Fragment > Google Maps Fragment. This will allow you define the names of the fragment, layout and package to use:



New Android Component

Google Maps Fragment

Creates a new fragment with a Google Map

Fragment Name: MapsFragment

Layout Name: fragment_maps2

Package name: com.example.cloud.fragments

Source Language: Java

You will need to modify the layout to define the interface, buttons, textViews, etc, and in the map you will need to add markers and polylines to show the direction and path taken.

Adding markers and lines

To add a marker to the map we just need to add the Latitude and Longitude of the point of interest to the handle of the GoogleMap object. Using the onMapReady callback of the Fragment, we can set a static point like this:

```
@Override
public void onMapReady(GoogleMap googleMap) {
    // Add a marker in Sydney, Australia,
    // and move the map's camera to the same location.
    LatLng sydney = new LatLng(-33.852, 151.211);
    googleMap.addMarker(new MarkerOptions()
        .position(sydney)
        .title("Marker in Sydney"));
    googleMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
}
```

The addMarker function returns the Marker object, where we can update its position with the setPosition function, and its orientation with the setRotation function. You can explore other properties like visibility, opacity, image, or even set events for dragging, clicking, etc. For a more detailed description of the marker options I recommend the Markers tutorial at: https://developers.google.com/maps/documentation/android-sdk/marker

In a similar way, adding a line can be achieved, using Polylines, where we need to define the points of the shape, and add the object to the map like:

```
// Instantiates a new Polyline object and adds points to define a rectangle
PolylineOptions polylineOptions = new PolylineOptions()
    .add(new LatLng(37.35, -122.0))
    .add(new LatLng(37.45, -122.0))  // North of the previous point, but at the same longitude
    .add(new LatLng(37.45, -122.2))  // Same latitude, and 30km to the west
    .add(new LatLng(37.35, -122.2))  // Same longitude, and 16km to the south
    .add(new LatLng(37.35, -122.0)); // Closes the polyline.

// Get back the mutable Polyline
Polyline polyline = map.addPolyline(polylineOptions);
```

To alter the shape of the polyline after it has been added, you can call Polyline.setPoints() and provide a new list of points for the polyline. You can customize the appearance of the polyline both before adding it to the map and after it has been added to the map. See the section on customizing appearances on the tutorial for further details.

Adding images to the map

Adding an image to the map can be easily achieved using Groud overlays, these are rendered with respect to the map orientation and automatically resize with it. To add a GroundOverlay, create a GroundOverlayOptions object that defines both an image and a position. You can optionally specify additional settings that will affect the positioning of the image on the map. Once you've defined the necessary options, pass the object to the GoogleMap.addGroundOverlay() method to add the image to the map. The addGroundOverlay() method returns a GroundOverlay object; you should retain a reference to this object if you want to modify it later. Here is an example to position an image of Newark at a given latitude and longitude, with a width and height in meters:

```
LatLng newarkLatLng = new LatLng(40.714086, -74.228697);

GroundOverlayOptions newarkMap = new GroundOverlayOptions()
    .image(BitmapDescriptorFactory.fromResource(R.drawable.newark_nj_1922))
    .position(newarkLatLng, 8600f, 6500f);
map.addGroundOverlay(newarkMap);
```

If we need to show a different floor of the indoor map, we will need to add an interface to the map and update the image (and boundary) according to the different floors, we can update the image (assuming the same dimensions) with:

```
// Update the GroundOverlay with a new image of the same dimension
imageOverlay.setImage(BitmapDescriptorFactory.fromResource(R.drawable.newark_nj_1922));
```

It is also possible to handle events (click) related to the image, for more information refer to the Ground overlays tutorial at:
https://developers.google.com/maps/documentation/android-sdk/groundoverlay