**All-in-one app**

**-Mobile computing-**

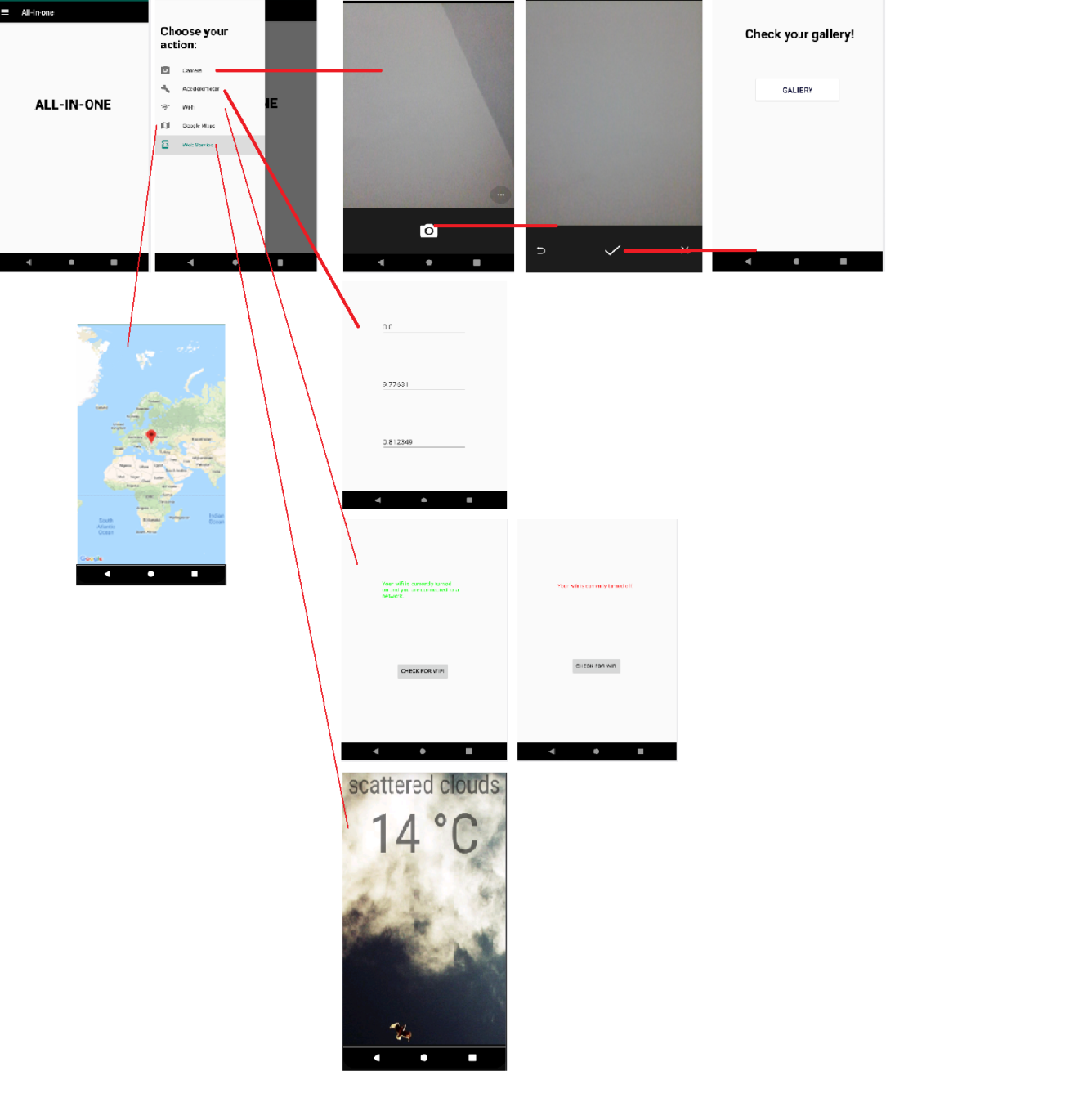
**Student: Cristina-Larisa Alexe**

**Group: C.EN.4.S1**

1. The project is an application that will allow the user to interact with some of the basic apps of the Android. The application will provide the user with 5 different types of actions (that’s where all-in-one name comes) which will be fun to interact with:

* Wi-Fi notifications;
* Camera activities;
* Accelerometer;
* Google Maps access;
* Weather information;

1. Application Wireframe



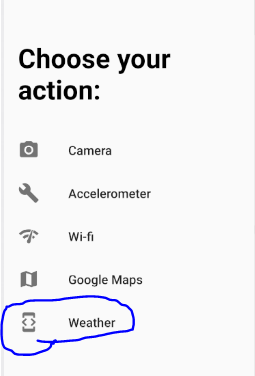
1. **Server side functionality**

The weather side of the application has a simple and minimalist user interface, showing exactly what we need to know about the weather conditions. Manifest file will contain all the information about the application such as the activities and the application permissions.

Two dependencies were needed : **volley**, used to make the HTTP request easier, and **glide**, used to download and display images easier. To add those libraries I modified build.gradle file. After making changes in gradle file, the project will ask a project resync which will import the two libraries. Since we needed HTTP request to retrieve the weather conditions, I asked for Internet Permission from the user. Also, I needed to know if the user has an internet connection, so I’ve asked for access network state permission as well, all these actions in AndroidManifest.xml.

Inside the weather\_activity.xml file I set a background image, and added two text views.

For the feature to function proper, MainActivity needed to be modified.

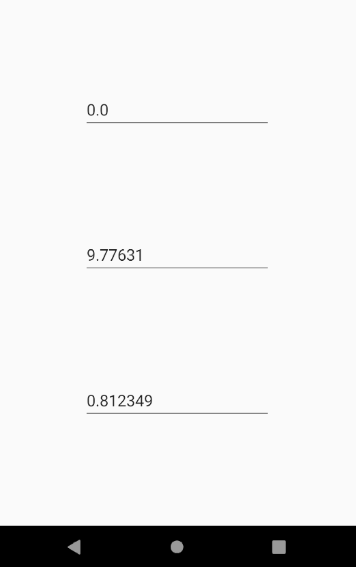
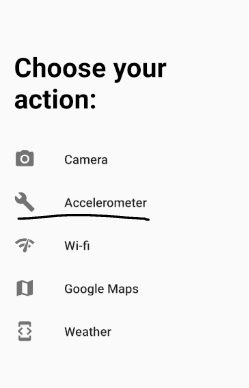


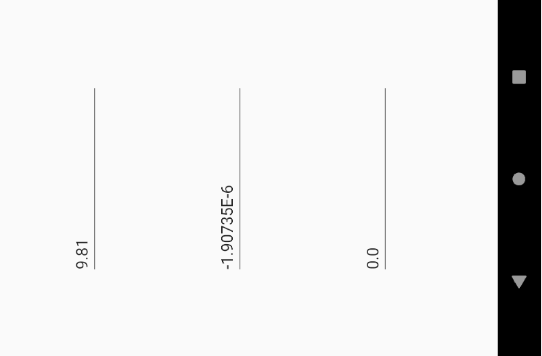
1. **Accelerometer**

For using the accelerometer, I first needed to set up the sensor. I set the main layout by invoking setContentView.

onCreate() function contains the main steps in showcasing the accelerometer functionality. First, we need to create a SensorManager and a Sensor object and register a listener to listen for changes reported by accelerometer. After having registered the listener, the function will react accordingly to the changes in values. onSensorChanged() function will be automatically called every time the sensor has any change to report.

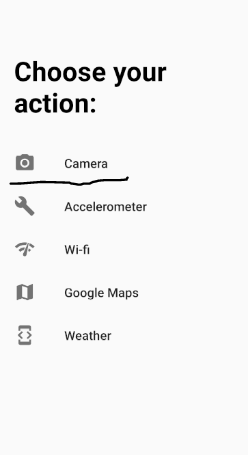
Regarding functionality design, we have three textViews that display the three axes of the accelerometer.

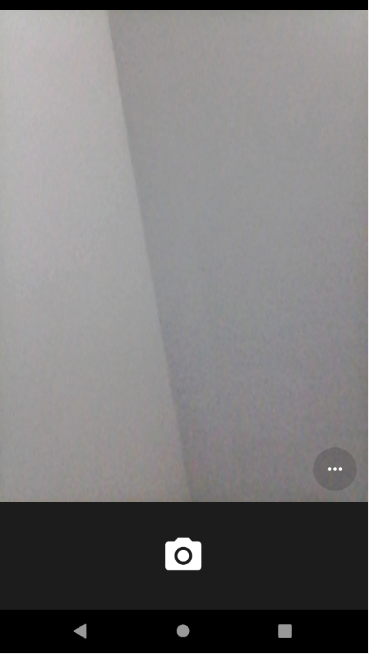


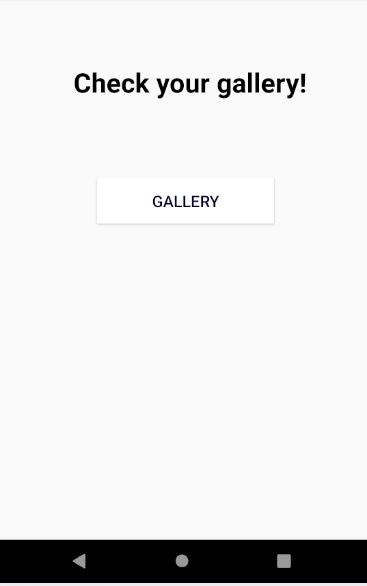


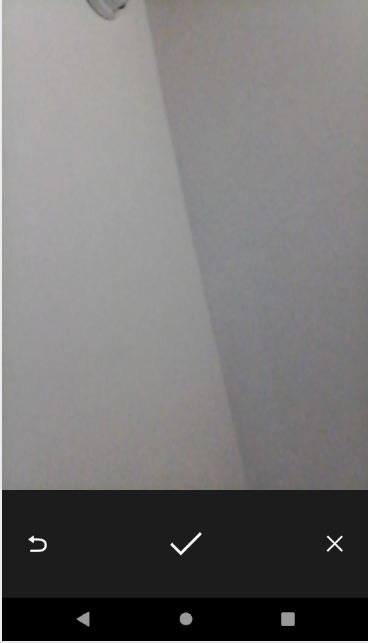
1. **Camera functionality**

The user will be able to access the camera, through the application, to take a photo that will be saved into gallery. For this to be done we need access to the camera. The application will request permission from the user, and also to read and write into the storage. I accesed the camera on a separate thread that’s launched from onCreate() function, because it can take a while and might bog down the UI thread. After calling the camera, a dispatchTakePictureIntent() function will be called, that describes what you want to do to the other applications. The app first checks if there is a camera activity that is able to solve that intent, and if there is a file object will be created by using the createImageFile function. The photo will be added to the gallery using galleryAddPic by creating and broadcasting an intent of ACTION\_MEDIA\_SCANNER\_SCAN\_FILE type.







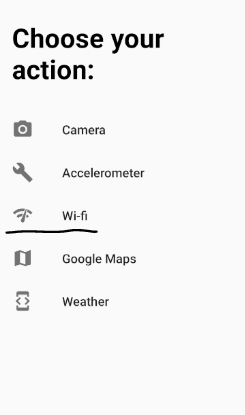
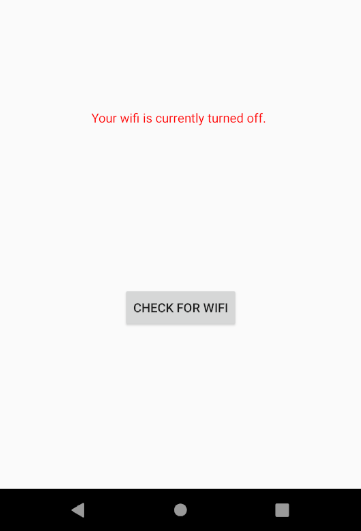
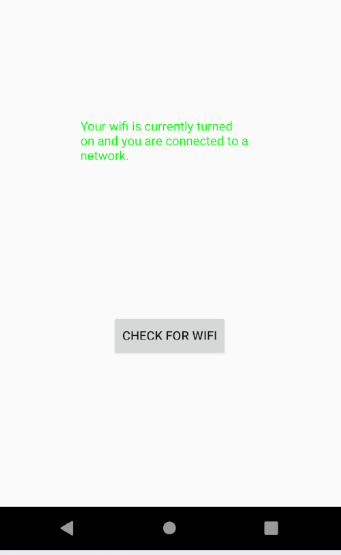


1. **Wi-Fi feature**

The Wi-Fi feature from All-in-one application simply checks if you are connected to a wi-fi or not. The design of the feature is very simple, meaning that it contains a textView which will show either you are connected to a network or not.

The main function available in the feature is checkForWifi() which will connect to an active network through connectivity manager. Then we will check if the active network is not null, if is connected or connecting and if it is wi-fi type.

If all the above conditions are true, then it is clearly that we are connected to the active network, else our network is turned off.



1. **Google Maps**

For editing Google Maps page we had available MapsActivity.java class and google\_maps\_api.xml which contains instructions on getting a Google Maps Api key that we need in order to access the Google Maps servers. The key I had for my application is restricted for Android apps. After following the information available in the file, I needed to add a marker on the map. I’ve created an object that contains the coordinated of the place, and after that the marker will be added to the map with that position using the AddMarker() function. moveCamera() function will receive the information, and move to that positon when the map is first launched.

