Mobile Computing Report

BestApp Mobile Application

Student: Dumitru Alexandra-Georgiana

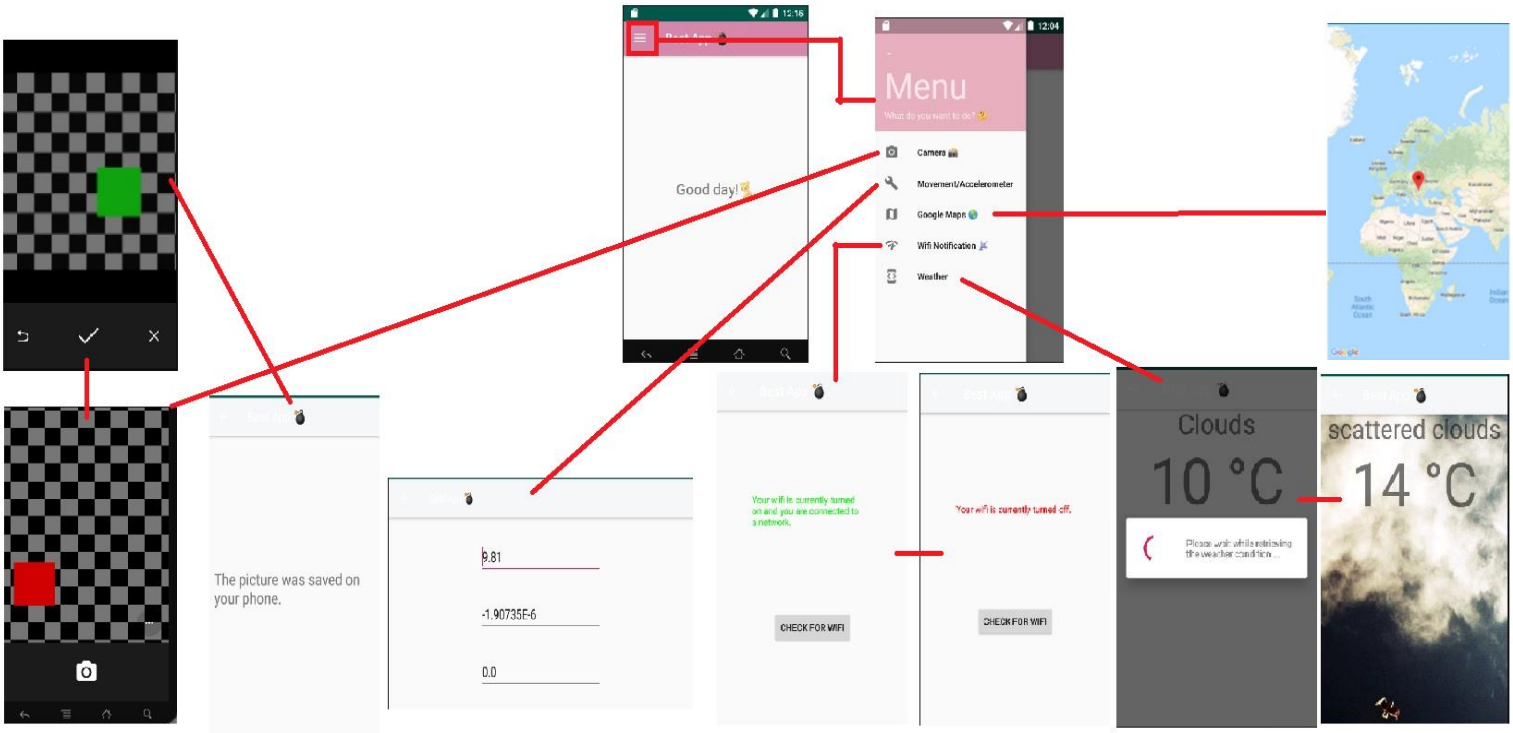
CEN 4.H1

1. About the project

The project is an all-in-one basic Android application that can help the user on day-to-day tasks. It doesn't really have a main purpose but it can do a lot of cool stuff.
The sensors I have decided to use are the following:

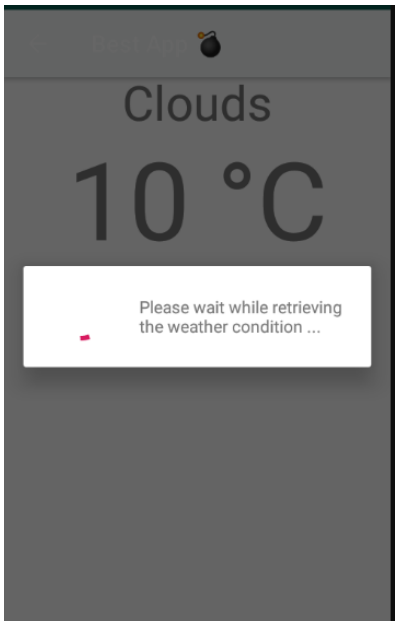
- Camera, which enables the user to take photos, decide whether or not to keep them and save them on the device.
- Location, for when using mobile data or wifi in Maps.
- Accelerometer to measure dynamic to sense movement.
- Wifi

2. Application wireframe



3. Server side functionality

For the server side functionality, I have decided to make a weather application.



First and foremost, app dependencies had to be added. The first one, Volley, a library that makes HTTPS requests easier and Glide, a library that makes displaying images easier.
I added them in the Gradle Scripts, in build.gradle (Module : App).

```
implementation 'com.android.volley:volley:1.1.0'
implementation 'com.github.bumptech.glide:glide:3.7.0'
```

Then, I had to request Internet Permission and Access Network State Permission.

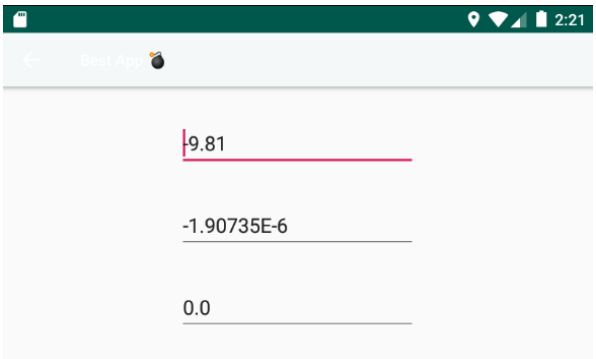
```
<!--Request Internet connection-->
<uses-permission android:name="android.permission.INTERNET" />
<!--Request to check Internet connection state-->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

I also added in layout(the .xml) two TextViews, to display the temperature and weather description, and one ImageView to show a background according the weather condition, to have the GUI ready.

```
<ImageView
    android:id="@+id/weatherbackground"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

<!--display the temperature-->
<TextView
    android:id="@+id/temperature"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp"
    android:gravity="center_horizontal"
    android:text="10 °C"
    android:textSize="90sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
<!--display the weather description-->
<TextView
    android:id="@+id/description"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="40dp"
    android:gravity="center_horizontal"
```

And I edited the MainActivity class to add the business logic.

4. Movement

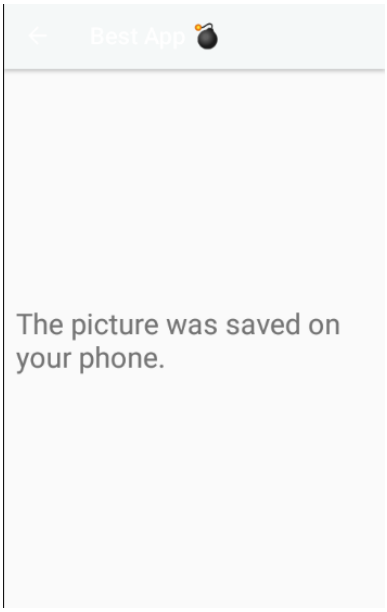


To access the data from the accelerometer, I created a SensorManager, Sensor object and register a listener that will use a broadcast receiver to listen for changes in the values reported by the accelerometer. The onSensorChanged function will be automatically called every time the sensor has any change to report. Here the textViews can be modified to display the X, Y, Z axis,

5. Images

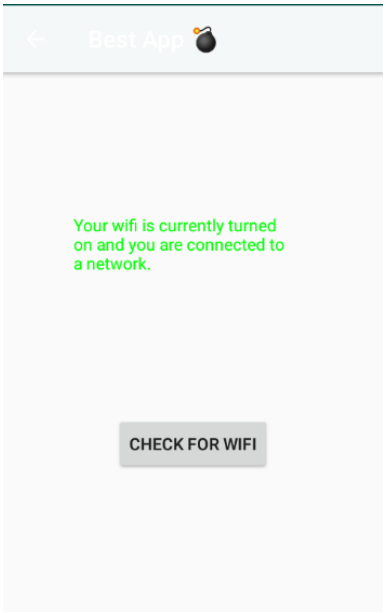
ml × | MainActivity.java × | CameraActivity.java × | weather_activity.xml × | WeatherActivity.java ×

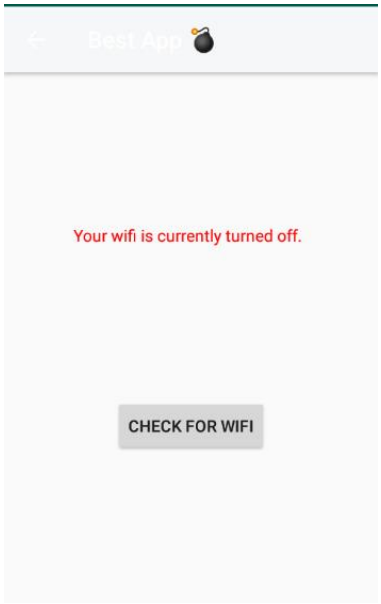
```
    static final int REQUEST_TAKE_PHOTO = 1;

    private void dispatchTakePictureIntent() {
        Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        // Ensure that there's a camera activity to handle the intent
        if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
            // Create the File where the photo should go
            File photoFile = null;
            try {
                photoFile = createImageFile();
            } catch (IOException ex) {
                // Error occurred while creating the File
            }
            // Continue only if the File was successfully created
            if (photoFile != null) {
                Uri photoURI = FileProvider.getUriForFile( context: this,
                        authority: "com.example.android.fileprovider",
                        photoFile);
                takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
                startActivityForResult(takePictureIntent, REQUEST_TAKE_PHOTO);
                galleryAddPic();
            }
        }
    }
}
```



Using the camera, the user can take a picture and save it on their device.
To take a picture and save it to the device, Permissions are needed from the user to access the camera and to read and write files to the phone's internal storage.
When the user enters the CameraActivity, the camera module must open so that they could proceed with taking a picture which would then be saved on their device. Iin the onCreate function of the activity, we call the function dispatchTakePictureIntent which dispatches the intent to take a picture using the camera. Before it starts the activity respective to that intent, it first checks if there is a camera activity that is able to resolve that intent. If there is, it creates a File object using the createImageFile function, starts the activity that uses the camera to take the picture and then calls the galleryAddPic that creates and broadcasts an intent of the ACTION_MEDIA_SCANNER_SCAN_FILE type.

6. Wifi listener





We showcase the wifi module in the app by having an activity that checks whether or not the wifi is on and connected to a network.

In order to do this, our activity will have a textView and a button. The button will act as a refresh button for the checker. The textView will display a text saying whether the phone is connected or not. We will also run the function that checks if the phone is connected, when we first enter the activity.

7. Maps and Directions

Configuring and setting up the **Maps** was easy since Google has made it very easy to import a new Activity that contains **Google Maps**.
https://developers.google.com/maps/documentation/android-sdk/start