Mobile Computing Report

Find my Car

Student Dorian Dascalu

1. About the project

The project presented in this report is representing by an application dedicated for car localization and guidance of the driver.
The application will provide to the user the possibility to save the location of the car, in order to be easily found and recognized inside difficult areas.

The location will be provided by the GPS and guidance to the location by maps services. Additionally to this, on board camera and accelerometer will be used to have more flexible ways to save the location of the car.

All the data will be saved on a cloud in order to have a history of locations available.

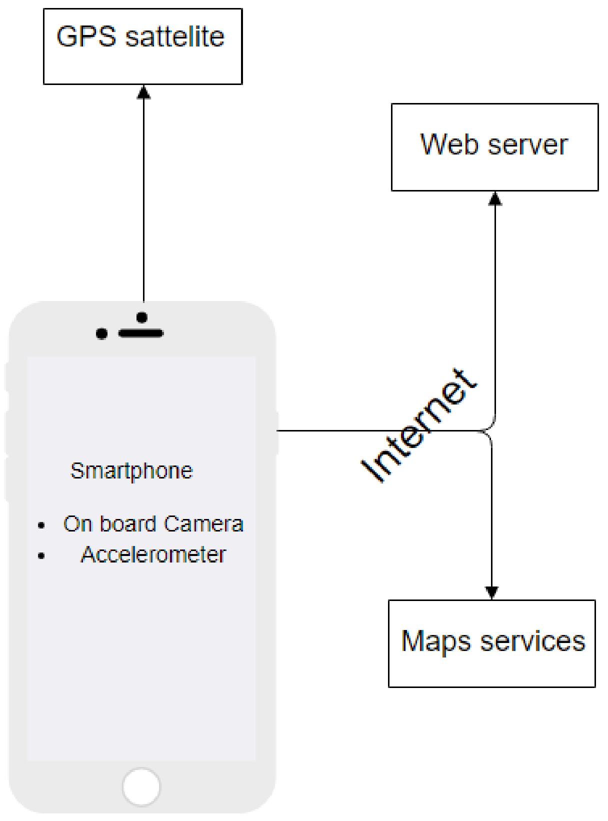
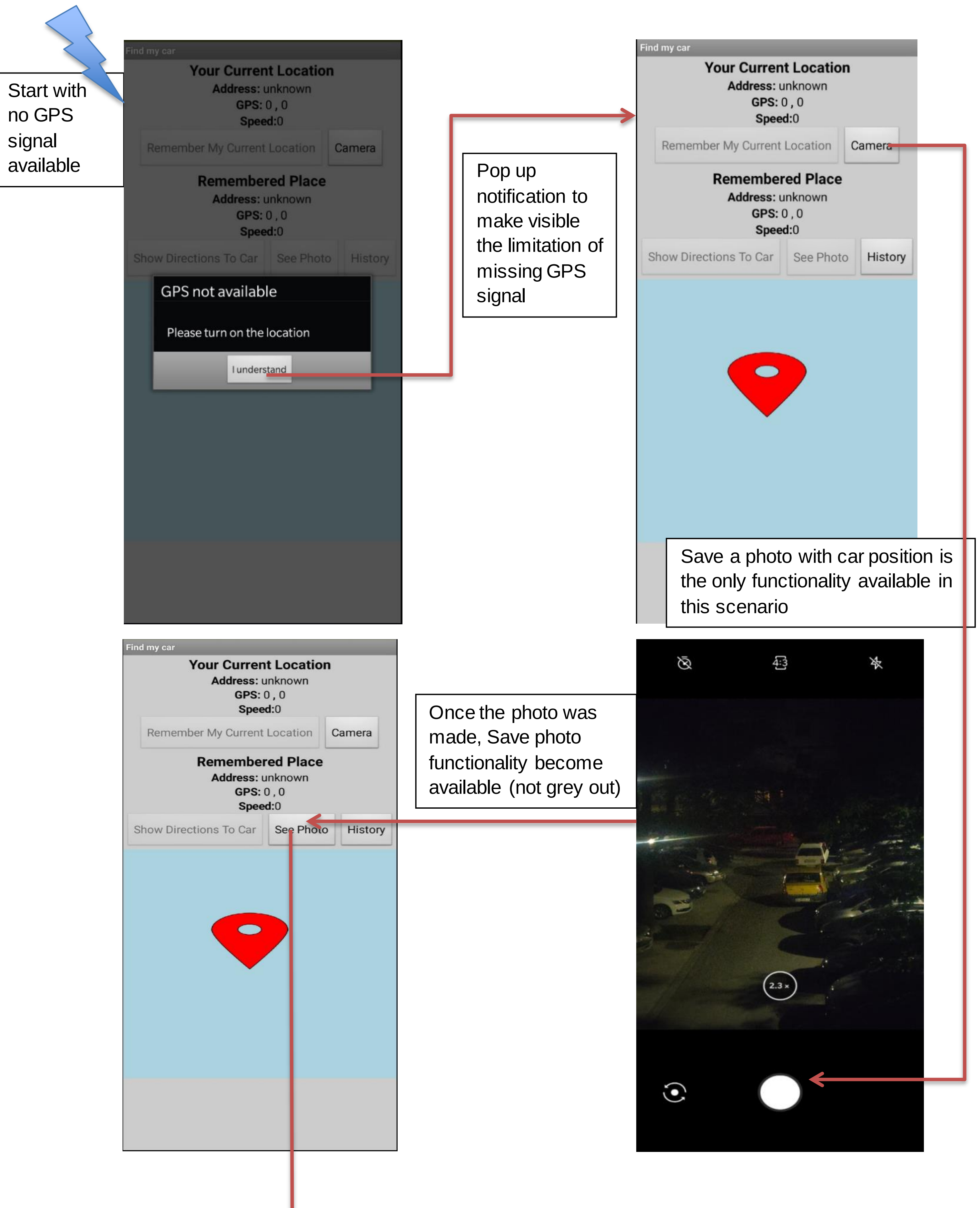


Fig 1 System high overview with interactions

2. Application wireframe

Keeping in mind the possibility of losing GPS signal due to different factors, the application will provide mechanism of saving the position in the both scenarios (GPS signal On/Off)



Start with no GPS signal available

Find my car

Your Current Location
Address: unknown
GPS: 0 , 0
Speed:0

Remember My Current Location | Camera

Remembered Place
Address: unknown
GPS: 0 , 0
Speed:0

Show Directions To Car | See Photo | History

GPS not available

Please turn on the location

I understand

Pop up notification to make visible the limitation of missing GPS signal

Find my car

Your Current Location
Address: unknown
GPS: 0 , 0
Speed:0

Remember My Current Location | Camera

Remembered Place
Address: unknown
GPS: 0 , 0
Speed:0

Show Directions To Car | See Photo | History

Save a photo with car position is the only functionality available in this scenario

Find my car

Your Current Location
Address: unknown
GPS: 0 , 0
Speed:0

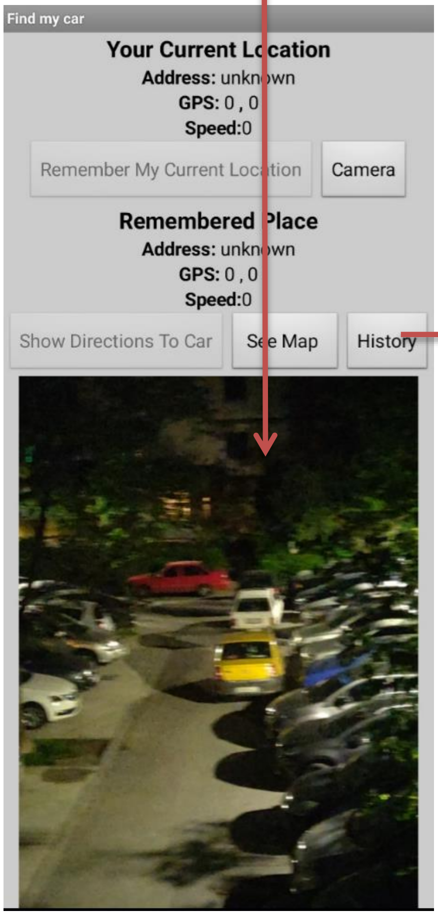
Remember My Current Location | Camera

Remembered Place
Address: unknown
GPS: 0 , 0
Speed:0

Show Directions To Car | See Photo | History

Once the photo was made, Save photo functionality become available (not grey out)

User can switch between Photo or Map perspective


Find my car

Your Current Location

Address: unknown

GPS: 0 , 0

Speed:0

Remember My Current Location | Camera

Remembered Place

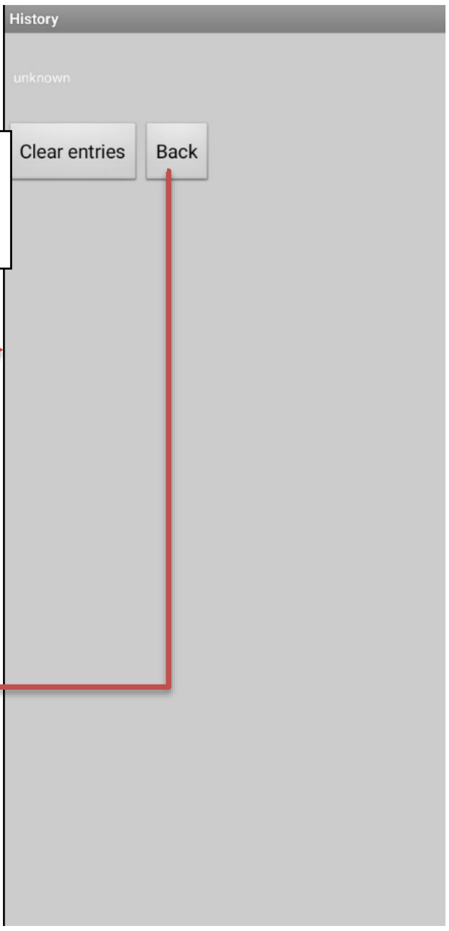
Address: unknown

GPS: 0 , 0

Speed:0

Show Directions To Car | See Map | History

History will not provide useful information since location is unknown

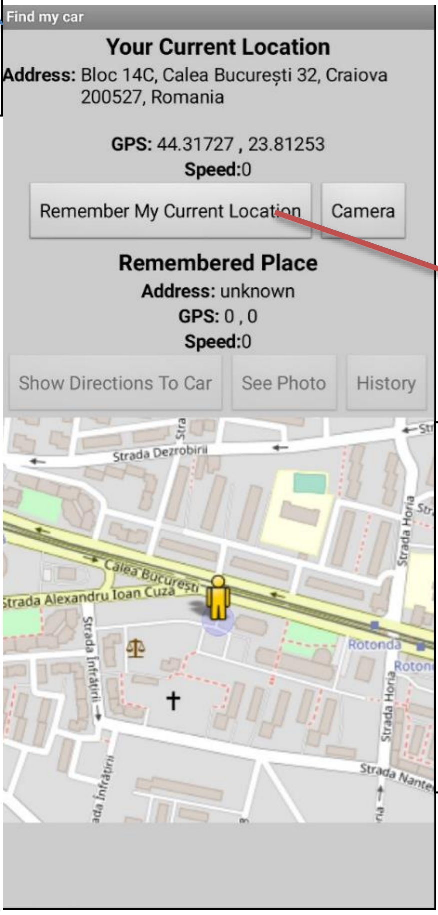

History

unknown

Clear entries | Back

User can return or clear the entries

Start with GPS signal available


Find my car

Your Current Location

Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

GPS: 44.31727 , 23.81253

Speed:0

Remember My Current Location | Camera

Remembered Place

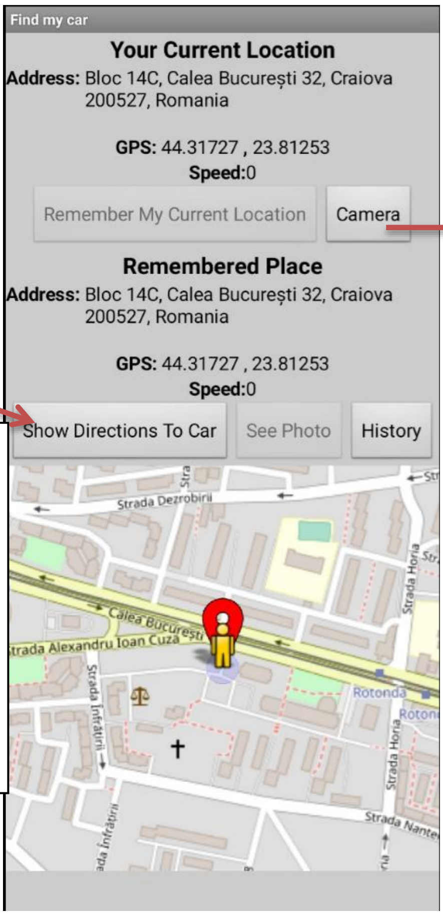
Address: unknown

GPS: 0 , 0

Speed:0

Show Directions To Car | See Photo | History

Once the location is available, driver can save his location, leading to :
- saving and placing a marker on the map
- allowing to show direction with maps services
- saving the location on web data bases


Find my car

Your Current Location

Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

GPS: 44.31727 , 23.81253

Speed:0

Remember My Current Location | Camera

Remembered Place

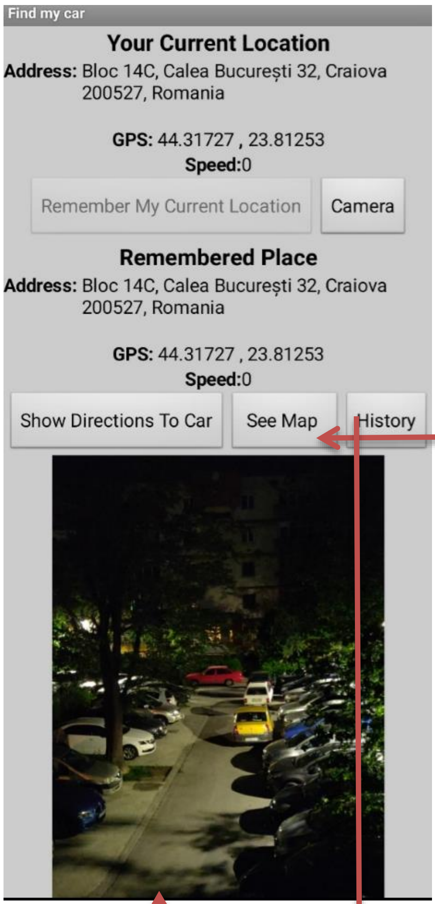
Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

GPS: 44.31727 , 23.81253

Speed:0

Show Directions To Car | See Photo | History

Save photo functionality now saves also the location automatically


Find my car

Your Current Location

Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

GPS: 44.31727 , 23.81253

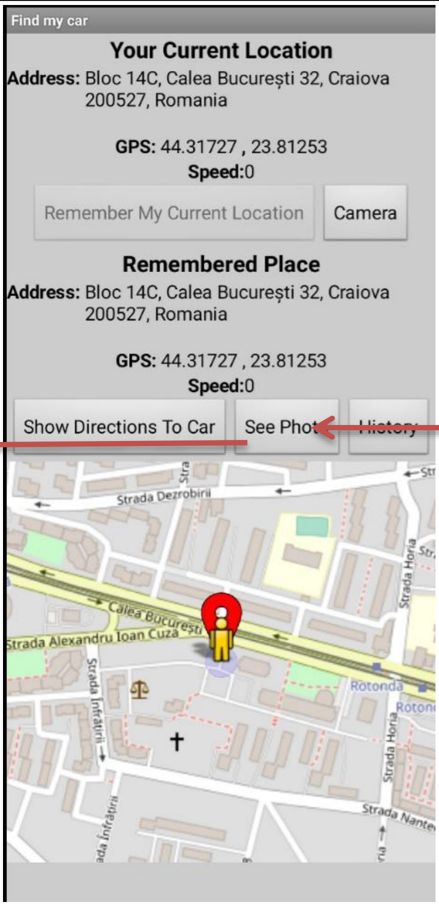
Speed:0

Remembered Place

Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

GPS: 44.31727 , 23.81253

Speed:0


Find my car

Your Current Location

Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

GPS: 44.31727 , 23.81253

Speed:0

Remembered Place

Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

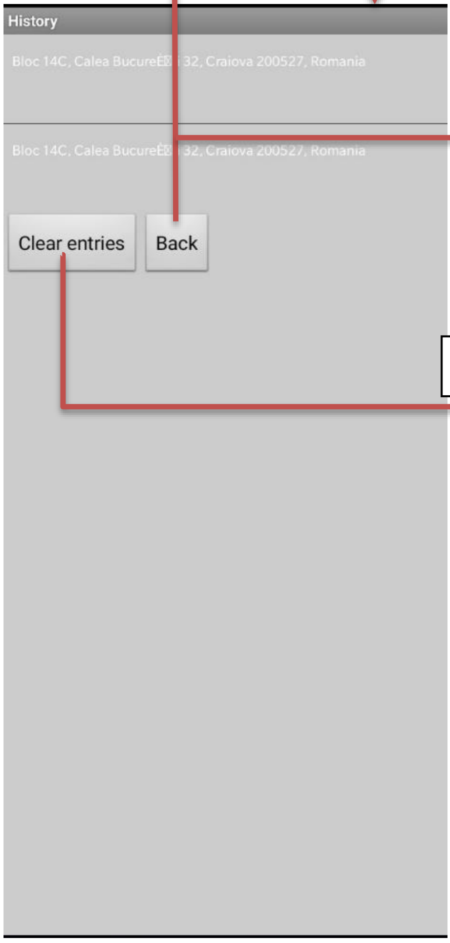
GPS: 44.31727 , 23.81253

Speed:0

Possibility to swap as described above

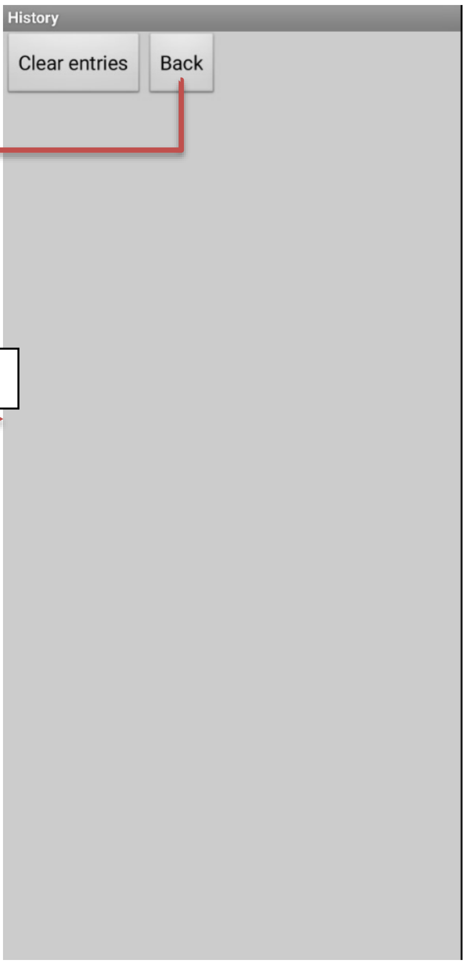
Back from where the software came

Two location saved (one normal and one from camera)

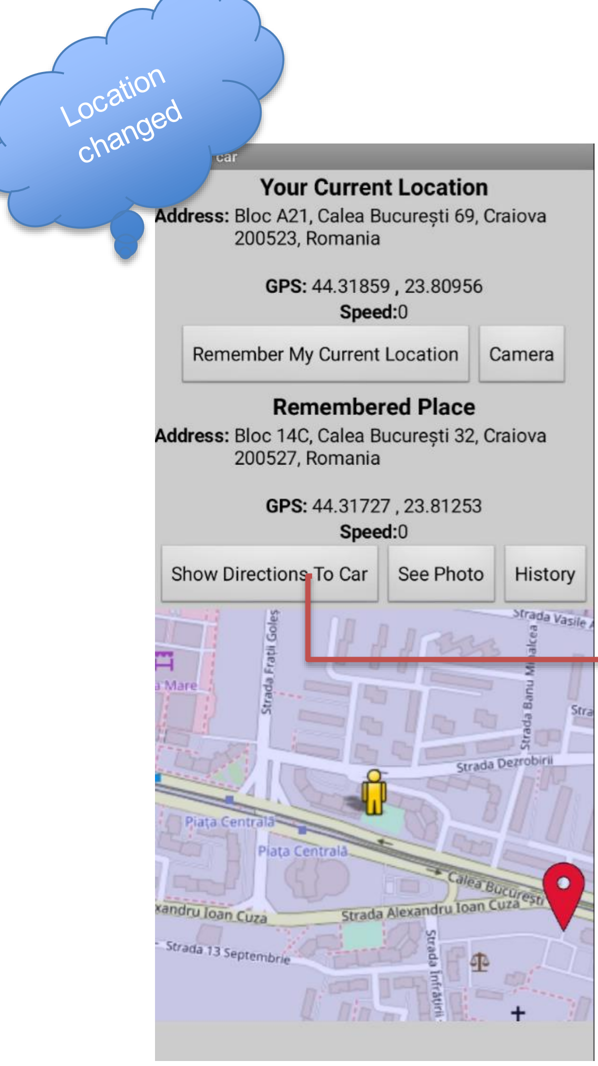

History

Bloc 14C, Calea Bucure�� 32, Craiova 200527, Romania

Bloc 14C, Calea Bucure�� 32, Craiova 200527, Romania


History

Clear entities will erase the data



car

Your Current Location

Address: Bloc A21, Calea București 69, Craiova 200523, Romania

GPS: 44.31859 , 23.80956
Speed:0

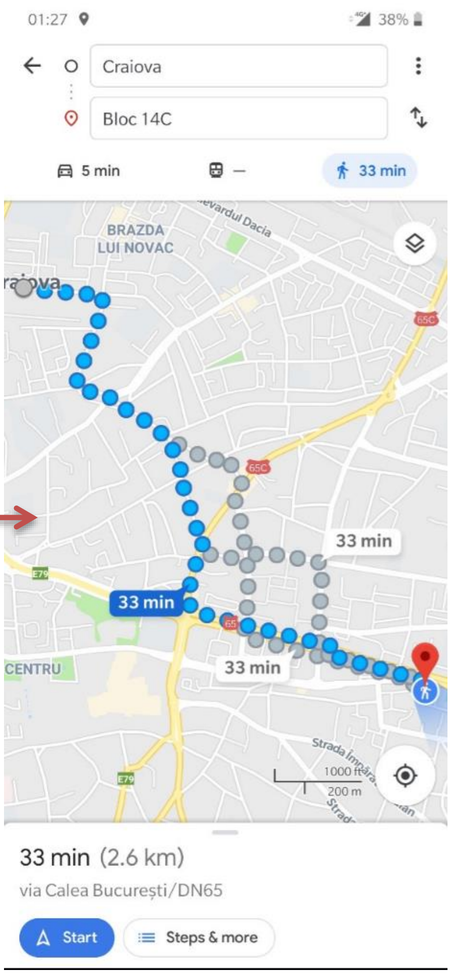
Remember My Current Location | Camera

Remembered Place

Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

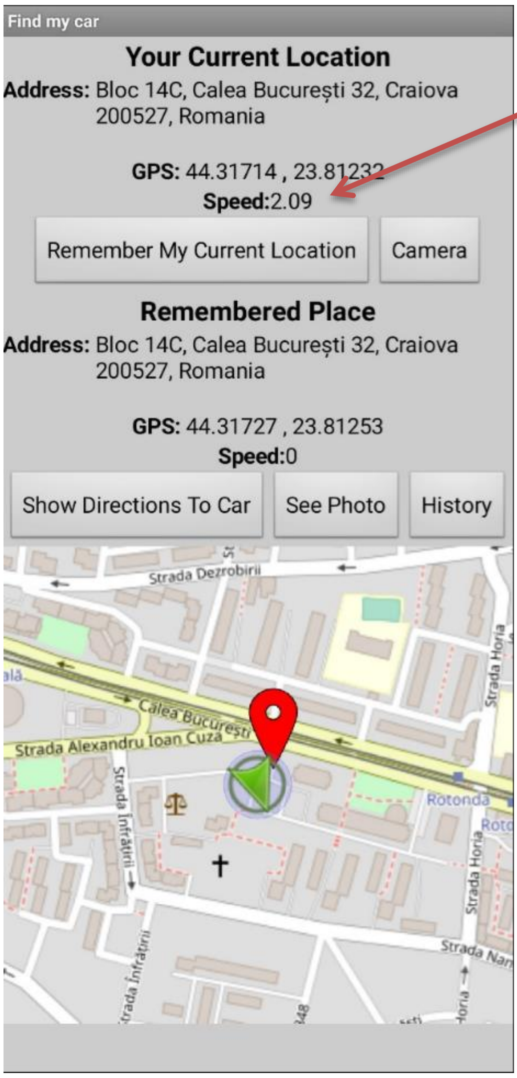
GPS: 44.31727 , 23.81253
Speed:0

Show Directions To Car | See Photo | History



33 min (2.6 km)
via Calea București/DN65

Show direction is using google maps functionality in order to provide guidance and traffic options



Find my car

Your Current Location

Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

GPS: 44.31714 , 23.81232
Speed:2.09

Remember My Current Location | Camera

Remembered Place

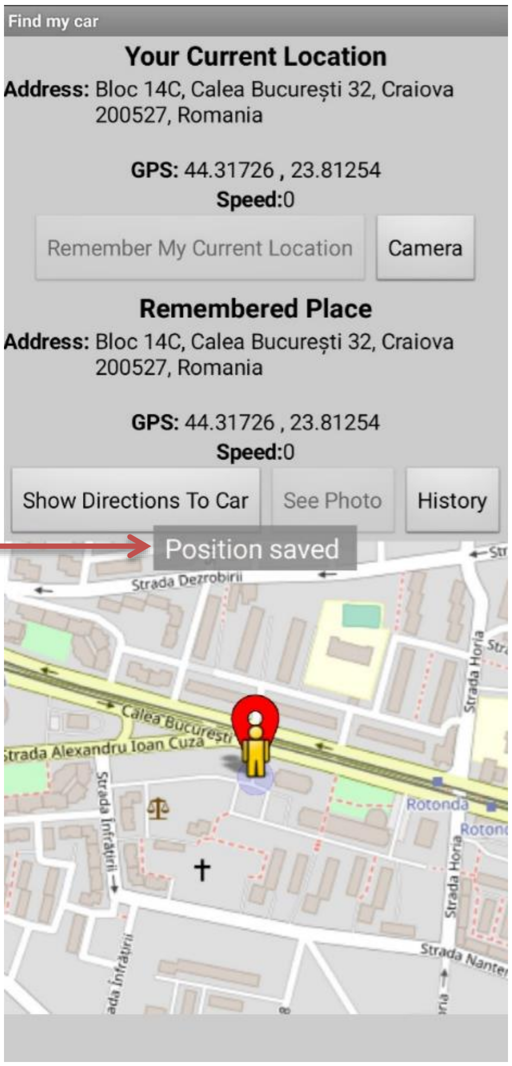
Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

GPS: 44.31727 , 23.81253
Speed:0

Show Directions To Car | See Photo | History

Application is capable to measure also the speed of current/saved location

Position will be saved if the device will be shaken



Find my car

Your Current Location

Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

GPS: 44.31726 , 23.81254
Speed:0

Remember My Current Location | Camera

Remembered Place

Address: Bloc 14C, Calea București 32, Craiova 200527, Romania

GPS: 44.31726 , 23.81254
Speed:0

Show Directions To Car | See Photo | History

Position saved

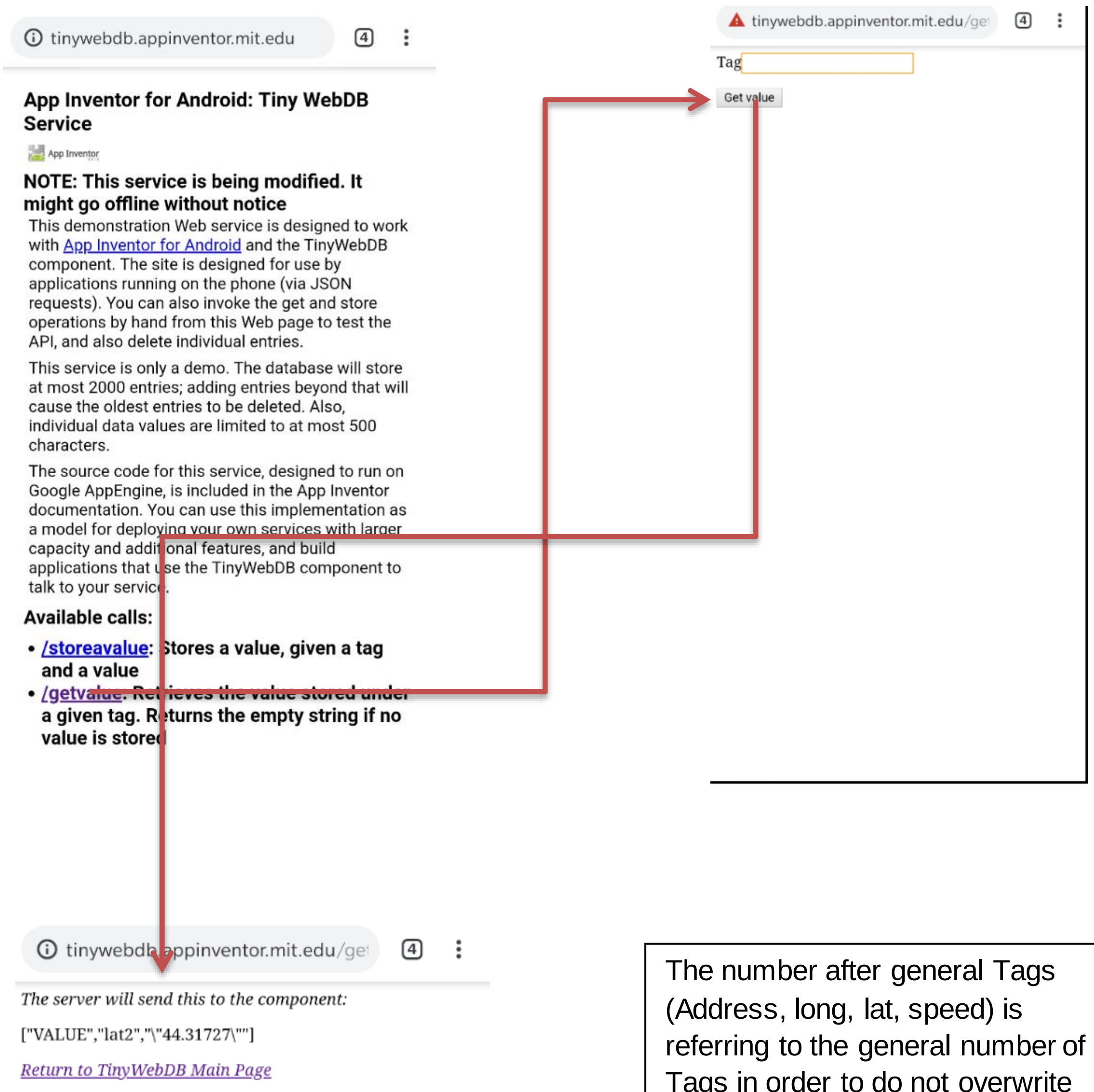
Fig 2 Application wireframe

3. Server Side Functionality

Implementation of cloud based web server consist in MIT App Inventor included plugin called “Tiny Web DB“ with API : http://tinywebdb.appinventor.mit.edu

On web handling of database is done as following :

Example of handling the requests



tinywebdb.appinventor.mit.edu
tinywebdb.appinventor.mit.edu/ge
Tag
Get value
tinywebdb.appinventor.mit.edu/ge

App Inventor for Android: Tiny WebDB Service

NOTE: This service is being modified. It might go offline without notice

This demonstration Web service is designed to work with App Inventor for Android and the TinyWebDB component. The site is designed for use by applications running on the phone (via JSON requests). You can also invoke the get and store operations by hand from this Web page to test the API, and also delete individual entries.

This service is only a demo. The database will store at most 2000 entries; adding entries beyond that will cause the oldest entries to be deleted. Also, individual data values are limited to at most 500 characters.

The source code for this service, designed to run on Google AppEngine, is included in the App Inventor documentation. You can use this implementation as a model for deploying your own services with larger capacity and additional features, and build applications that use the TinyWebDB component to talk to your service.

Available calls:

- **/storeavalue: Stores a value, given a tag and a value**
- **/getvalue: Retrieves the value stored under a given tag. Returns the empty string if no value is stored**

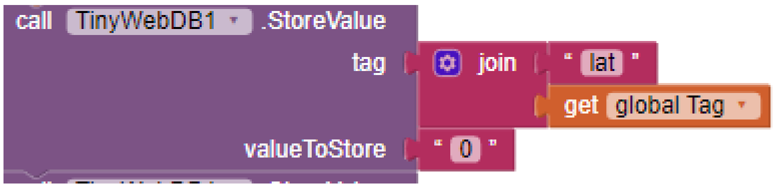
The server will send this to the component:

["VALUE","lat2","\"44.31727\""]

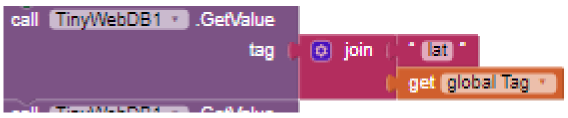
Return to TinyWebDB Main Page

The number after general Tags (Address, long, lat, speed) is referring to the general number of Tags in order to do not overwrite data

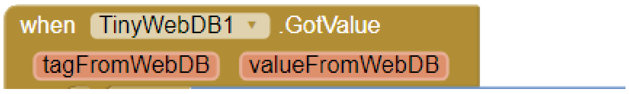
This framework has the following APIs :



Save the value "0" with tag "lat" concatenated with "global tag" to have a unique value

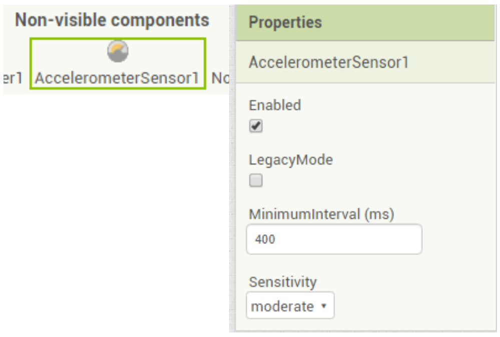


Get value from tag "lat" concatenated with "global tag". This method will invoke a a function trigger base which will return the value



4. Movement

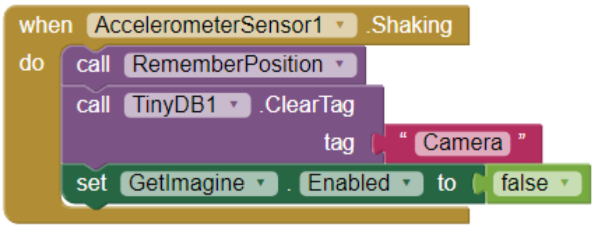
In order to use Accelerometer sensor, a non-visible component shall be added.


Non-visible components
AccelerometerSensor1

Properties
AccelerometerSensor1
Enabled ✓
LegacyMode
MinimumInterval (ms) 400
Sensitivity moderate

The application will start 400 ms refresh configuration with a moderate sensitivity.

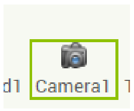
The role of the accelerometer in the application is to have a more friendly way to save the location. When shaking the device, an event is triggered and we start to save location.



When accelerometer detects a shaking action, it will call functionality to save position, and also clearing the current photo since is obsoleted

6. Images

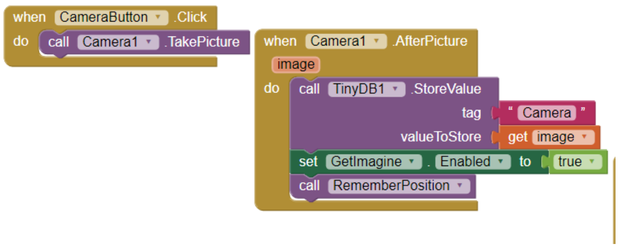
In order to use Camera, a non-visible component shall be added.



It does not have any initialization configuration.

The role of the Camera is to provide an alternative to saving the position of the car.

Additionally to the situation when GPS signal is not available, Camera will also save the position when the location is available.



```
when CameraButton .Click
do  call Camera1 .TakePicture

when Camera1 .AfterPicture
  image
do  call TinyDB1 .StoreValue
                      tag  " Camera "
             valueToStore  get image
    set GetImagine . Enabled to true
    call RememberPosition
```

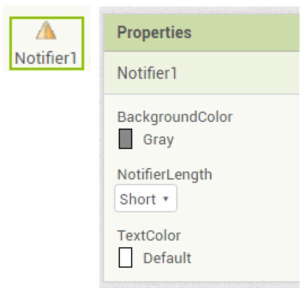
When the button is pressed, camera is taking a picture. After the photo is saved, and event will be triggered to get the data and save to database

7. Wifi Listener

The application is not using Wifi module.

8. Notifications

In order to use Notifications, a non-visible component shall be added.



Notifier1

Properties

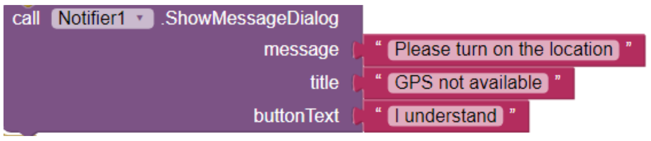
Notifier1

BackgroundColor
■ Gray

NotifierLength
Short

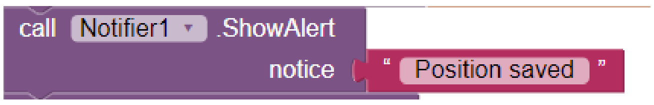
TextColor
□ Default

Default configuration will consist in font color and the duration.



```
call Notifier1 .ShowMessageDialog
                 message  " Please turn on the location "
                   title  " GPS not available "
              buttonText  " I understand "
```

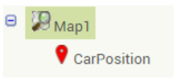
To raise a notification, you have to select proper notification format and fill the messages box



```
call Notifier1 .ShowAlert
                 notice  " Position saved "
```

9. Maps and Directions

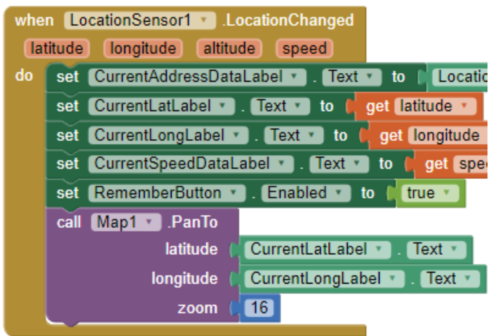
In order to use Maps, visible component shall be added.



Map1
CarPosition

Default configuration will consist in initial position and customize of interface.

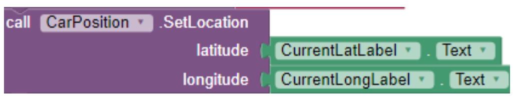
The application will show current position on mini-map from the screen when a new location is detected.



```
when LocationSensor1 .LocationChanged
  latitude  longitude  altitude  speed
do  set CurrentAddressDataLabel . Text to Locatic
    set CurrentLatLabel . Text to get latitude
    set CurrentLongLabel . Text to get longitude
    set CurrentSpeedDataLabel . Text to get spe
    set RememberButton . Enabled to true
    call Map1 .PanTo
          latitude  CurrentLatLabel . Text
         longitude  CurrentLongLabel . Text
              zoom  16
```

When location is changed, an event is raised where we are saving all the new values. Furthermore, we are setting a new location with latitude and longitude parameter and also zoom.

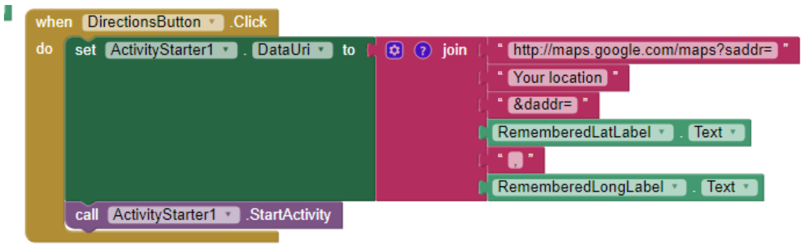
The marker is added when a new position is remembered in order to have the position of the car.



```
call CarPosition .SetLocation
          latitude  CurrentLatLabel . Text
          longitude CurrentLongLabel . Text
```

The marker will be moved to the new current position.

In order to have the directions to the position, the application will invoke Google Maps services.



```
when DirectionsButton .Click
do  set ActivityStarter1 . DataUri to  join  " http://maps.google.com/maps?saddr= "
                                             " Your location "
                                             " &daddr= "
                                             RememberedLatLabel . Text
                                             " , "
                                             RememberedLongLabel . Text
    call ActivityStarter1 .StartActivity
```

When the button is pressed, we are creating a string with current position as starting address "Your position" and destination address.
After that, application will launch the Activity and Google maps services will calculate the tracks.

Project location and information:

https://github.com/DascaluDorian/Projects/tree/master/Mobile%20Computing%20Report