UCLA | MS (COMPUTER SCIENCE) | FALL 2015

Carma

If you are Texting, you are not Driving!

Archana Ramakrishnan
Under the guidance of Professor Majid Sarrafzadeh

12/1/2015

GIT: https://github.com/arcVoila/FinalProject_Carma/

Table of Contents

Abstract

The US Department of Transportation estimates that reaching for a phone distracts a driver for 4.6 seconds, or the equivalent of the length of a football field, if the vehicle is traveling at 55 MPH [1]. The National Safety Council estimates that around 1,600,000 accidents are caused by texting during driving alone per year [2]. Reports suggest that texting while driving is 6 times more likely to cause an accident than driving intoxicated [3]. Human behavior is much harder to change than technology. Hence, there have been several attempts to curb texting during driving using technology.

We propose an inexpensive and indigenous technological solution to this problem that involves a hardware device similar to the one used in DUI control and a Mobile Android Smartphone. We also try to find a workaround for the inherent urgency that the user experiences when he/she receives a text by allowing the user to view a list of safe locations near to his/her current location, where he/she could park the phone and read the text-message.

The target market for this application will be the general consumers who can install this application on their phone or maybe even on their loved ones' phone. Parents of teenagers could be potential high-frequency users of this application. The app allows users to choose which apps to block. Any app installed on the phone can be blocked by this app. The user can also choose three emergency contacts and the numbers of these contacts will be displayed on the app block screen. The user can also preselect the type of place he would like to drive to in order to stop his/her car and text. For example, the user can preselect that he/she would like to halt at a restaurant and he/she can navigate to the nearest restaurant from the block screen.

Because our app promises to help the consumer build good driving *karma*, we have come up with a name that befits this situation and we call our safe-driving application as '***Carma'***.

1. Recap of prototype

- The prototype that we developed for our class last Quarter used Bluetooth to identify the car driver.
- On receiving a signal from the Bluetooth beacon, the android application installed on the driver's device would block the texting application on his/her phone.
- This prototype used the distance between the Bluetooth device connected to the ODB2 port of the car and the driver's cell phone as an indicator to distinguish between the driver and the co-passenger in the same car.

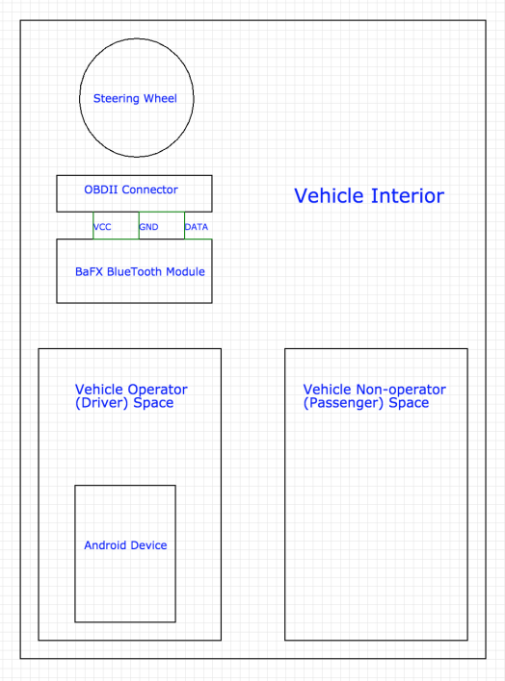

Figure 1: System Physical Schematic

1.1 Deliverables of the Prototype

All experiments were done in a classroom setting.

- Prototype app distinguished between the driver and the co-passenger.
- Blocked Messaging if the phone was used by the person in the driver's seat.
- Unblocked messaging if the phone is used by a co-passenger.
- On clicking the Safe Spot button, displayed Google maps of the current vicinity.

While the Bluetooth model worked well in a classroom environment, it did not perform as well in a car. A private company called 'Cell Control' also sells a product that claims to have the same feature as the product we proposed in class. The reviews on the app store for this app are listed below. While people have appreciated the concept, the

reviews for the app are not at all encouraging. With Carma, we plan to provide better features set for the same principle of blocking distractions during driving.

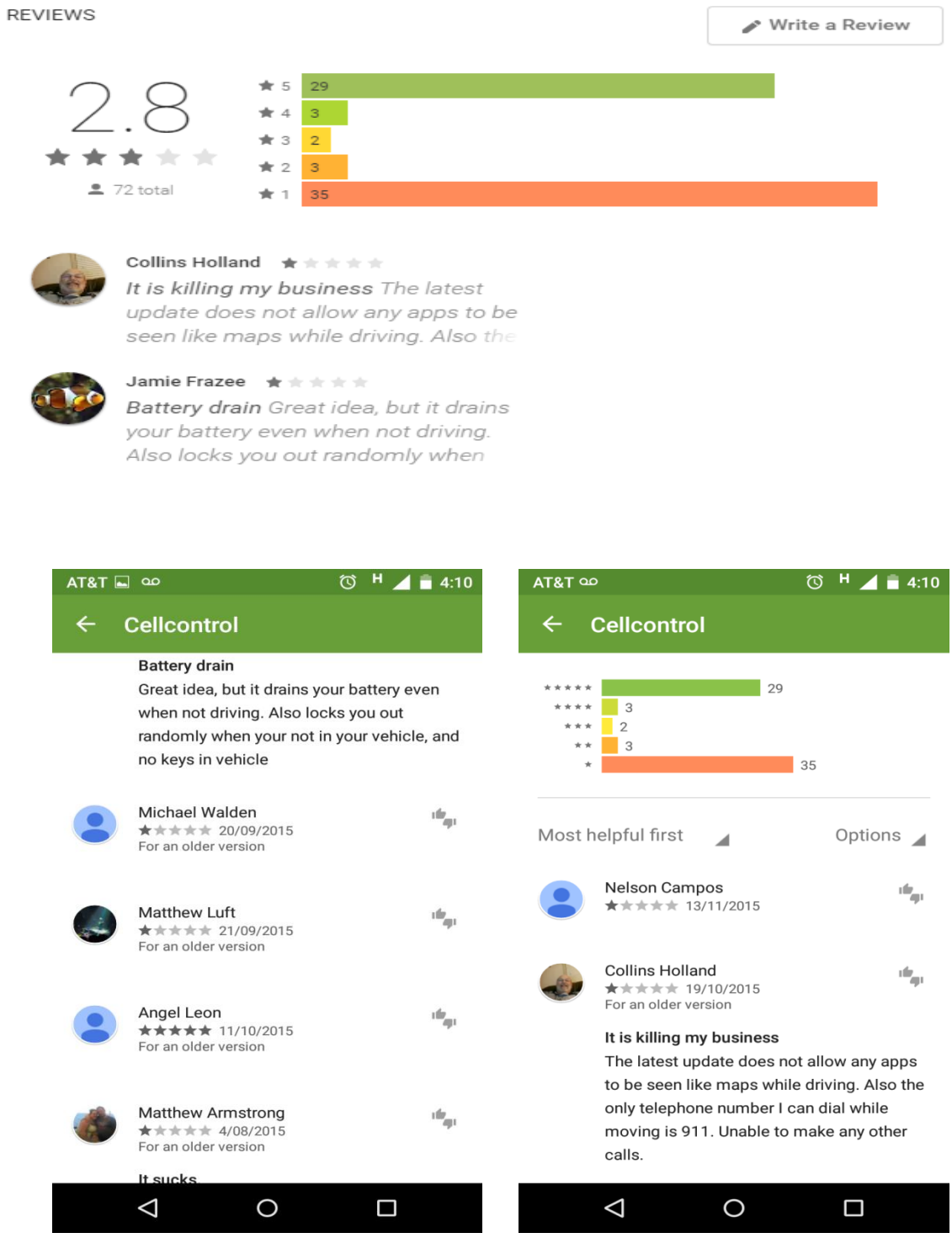

Figure 2: Reviews for Cell Control

The main issues listed above in the App reviews match the ones that we identified with Bluetooth. Sending signals for the bluetooth piconet leads to draining of the battery. Disturbance and interference also causes bluetooth to give inconsistent results while identifying the driver and co-passenger. Listed below are some of the most critical drawbacks of using bluetooth.

1.2 Problems with Bluetooth

1. Can be circumvented by switching off the Bluetooth on the driver's phone.
2. Distance accuracy (for distinguishing the driver from the co-passenger) can vary depending on interference from other Bluetooth devices and the build of the car.
3. The app uses system level android calls to hear for the Bluetooth beacon. This can drain the battery of the phone quickly and hence would not the preferred by users.
4. Messaging can be unblocked if the driver stretches his arm to read the messages, which could pose a severe threat to the driver.

In order to overcome these issues, we have come up with a new technique to integrate the android capabilities with the car. We use the technique used in Ignition Interlock Device (IID) for DUI cases. Using this method, we plan to eliminate all the drawbacks of Bluetooth and have a hardwired technique to ensure that the phone of the driver is blocked.

Since the installation of IID is currently being done for several DUI cases, we can be sure that such a solution is feasible. In the future, this could be added as a feature to the newly made cars. The technique is explained in detail in the further sections.

2. New Model

- In order to fix the problems with Bluetooth, we have come up with a more robust, hard-wired solution.
- We draw inspiration from the Ignition Interlock Device (IID) which is installed in the cars of drivers who are found to be serial offenders of driving under influence (DUI).

Figure 3: Ignition Interlock Device

2.1 Working of IID

1. The driver blows into the mouthpiece.
2. The fuel cell inside the ignition interlock device analyzes the blood alcohol concentration (BAC).
3. If the BAC is above the pre-determined limit, the fuel cell triggers a relay to stay open and the car will not start.
4. If the BAC is within the pre-determined limit, the fuel cell triggers a relay to close and the car will start.

2.2 Leveraging IID for Carma

1. Instead of the **mouthpiece**, we plan to attach the **driver's phone** to the same set-up.
2. There would be a one-to-one match between the driver's phone and the circuitry. A particular circuitry will be configured to accept only a particular phone. This means that the system cannot be fooled to accept a different phone.
3. The relay circuit would be completed only if the phone is attached; which means the driver will not be able to start the car unless he/she plugs in the phone to the outlet.
4. Once the phone is attached to the outlet, it will block texting on the phone.

2.3 Test Cases

1. **Driver plugs the phone off when the car is in motion.**

    - Under no circumstances should a running car be made stop abruptly. But, at the same time, the driver should have a motivation/obligation to plug the phone back in.
    - For this, we propose that, if the driver plugs off the phone, then our system would honk the car for a predetermined time interval.
    - Honking is considered socially rude and offensive in most parts of America and would divert attention towards the driver. We think that this would be a strong deterrent to the driver and he/she would keep the phone plugged in.

2. **Driver loses the Phone.**

    - Because the presence of the phone is required to start the car, it becomes essential to insure events such as theft, damage or loss of phone.
    - We plan to provide the driver a master key/code that can be used a limited number of times to start the car without the presence of the phone.
    - The codes will be limited in number in order to avoid misuse by using these codes to circumvent the system in question. In future, we can also charge the users for this code.

3. **Driver stops at a Stop Sign.**

    - Since the car engine is still running, the system will behave the same way as in point (1) if the phone is unplugged. Meaning the car would honk.

4. **Driver parks/stops the car.**

    - After the car engine is turned off, the driver can safely remove the phone from the associated circuitry.
    - After the phone is plugged off, the driver will be able to access his/her messaging app after an interval of one minute.

5. **Driver wants to urgently text/read messages.**

    - If the driver wants to urgently read his/her text messages or respond to the messages, then the driver can press the safe spot option on the plugged in phone and the app will show the nearby areas where the driver can park the car and respond to the messages.
    - Further down the line, this feature can be made intelligent by learning from frequent routes of the driver and/or from predetermined choices of the driver.

2.4 Advantages of the New Model

1. Will completely eliminate the interference and the distance calibration issues caused by Bluetooth.
2. Driver cannot cheat by holding the phone at an angle or by holding it at a distance (e.g. by holding it in the stretched arm).
3. No hassle of differentiating between the driver and the co-passenger. The driver's phone is attached to the car and hence the car can be uniquely identified. This makes the system foolproof.
4. Because the car will not start till the phone is linked, this system will also protect against car theft.
5. Phone will be charged continuously and hence there will be no danger of the battery getting drained or phone getting switched off.
6. Idea is unique and patentable.

2.5 Main Components of the Model

- Hardware Component (IID, communication with the Car)
- Android App (Blocking and other features)
- Communication (between the phone and the hardware)

The next sections discuss the Android App Component of Carma.

3. Android App

3.1 Blocking of Apps – Changes from Android Version 5 to 5.1

The android app developed for Spring 2015 class had only one property, it blocked the messaging (default texting) on the Android phone. The blocking worked on Android version 5 (Android Lollipop). This version was released in November 2014 and Android 5.1 update was sent to android devices starting from March 2015.

- Code used in Android 5 is as shown below. In this case, the function getRunningTasks gave a list of activities that are currently active on the phone. Getting the topActivity would return the foreground app.
- I would check if this activity was the messaging activity and if it is, then open the blocking activity on top of it to prevent the user from accessing the messaging app.

```
final List<ActivityManager.RunningTaskInfo> taskInfo
=mActivityManager.getRunningTasks(1);
final ComponentName componentName = taskInfo.get(0).topActivity;
final String[] foreGroundPackage = new String[1];
foreGroundPackage [0] = componentName.getPackageName();
```

Figure 4: Code in Android 5 for Blocking

- However, starting Android version 5.1, topActivity(0) returns the running process, which my case was the activity that ran Carma. As a result, my code broke for Android 5.1 and I was not able to block apps.
- This is because starting Android 5.1, the security settings became much more stringent and it blocked an app from getting the details about other apps running on the phone.
- A hack to this is to use Android UsageStatsManager, which gives statistics about the usage of different apps on the phone.
- In order for the app to use this package, it is necessary that the user gives the app permission for usage access.
- When the use installs the app, a prompt asks if the user agrees to allow usage access to the app. The app will work correctly ONLY if the user grants this permission.
- The code that asks for this permission and the screens that are displayed on installation are shown in Figures 5, 6 and 7 respectively.

```
Intent intent = new Intent(Settings.ACTION_USAGE_ACCESS_SETTINGS);
startActivity(intent)
```

Figure 5: Code that asks for User Permission

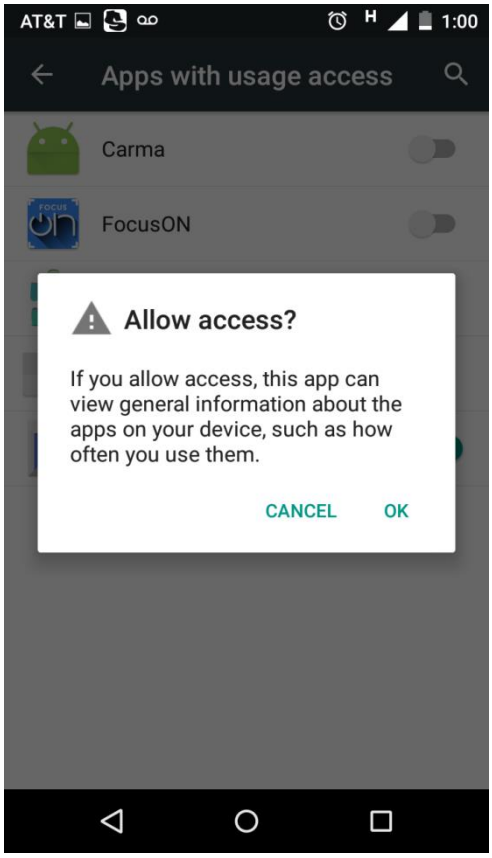

Figure 6: Request for Access

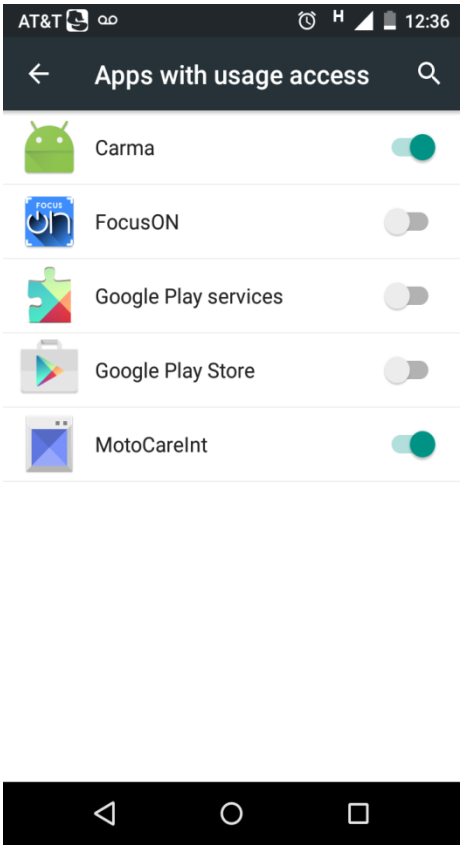

Figure 7: Granted Access to Carma

For android versions 5.1 and above, I used the code given in Figure 8 to get the foreground activity. After getting the foreground activity, a check is performed to see if any of the blocked activities are in the foreground. If yes, then the Block Activity is called on top of the foreground activity.

Currently Carma is backward compatible with version 5.

```
List<UsageStats> appList = usm.queryUsageStats(UsageStatsManager.INTERVAL_DAILY,  time - 1000*1000, time);
        if (appList != null && appList.size() > 0) {
            SortedMap<Long, UsageStats> mySortedMap = new TreeMap<Long, UsageStats>();
            for (UsageStats usageStats : appList) {
                mySortedMap.put(usageStats.getLastTimeUsed(), usageStats);}
        }
```

Figure 8: Code for getting the foreground activity in Android V > 5.1

3.2 Listing all installed apps and allowing user to choose the apps to block.

- This is the feature that was really important in terms of ease of use and display. The UI for the feature is displayed in Figure 9. In order to make the apps seem familiar, the app displays a list of apps with their icon, name and a checkbox to select the apps to block.
- In order to save the user's preferences after the app is closed, the app uses SharedPreferences to store the choices.
- Unlike other apps in the app store, the user does not have to click a submit or a block button. In order to block or unblock an app, the user only has to check or uncheck the checkbox.
- Carma runs a service in the background which keeps pinging for the foreground app every 5 seconds. In case any of the checked app is opened, then the app is blocked if the mode is DRIVING.
- Because the app runs a service to check if the blocked apps are open, the blocking will work even if the app is removed from the list of background apps.

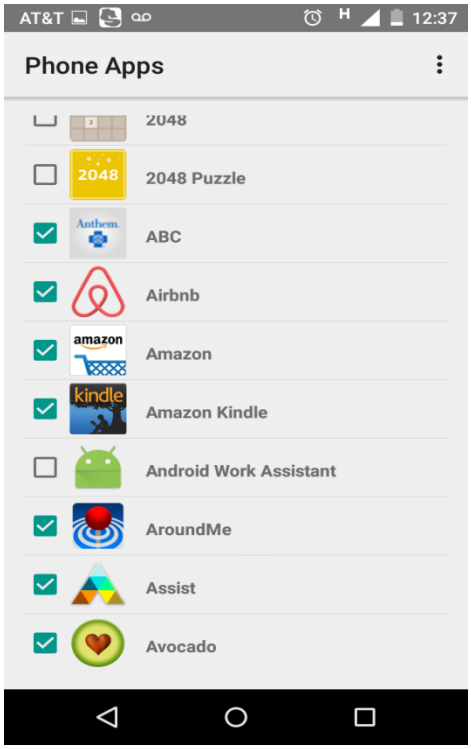

Figure 9: Select App to Block

3.2 Add Emergency Contacts to the block screen

- The app gives an option to select three contacts from the contact list to be used as emergency contact on the block screen.
- The Figure 10 and 11 show the selection of contacts and the maximum number of contacts that can be selected as emergency contacts.
- The selection is stored in a shared preference so that it can be retrieved when the app is reopened.
- If all three contacts are selected, then all three are displayed on the block screen. In case, fewer numbers of contacts are selected, then the contact button on the block screen is dynamically updated.

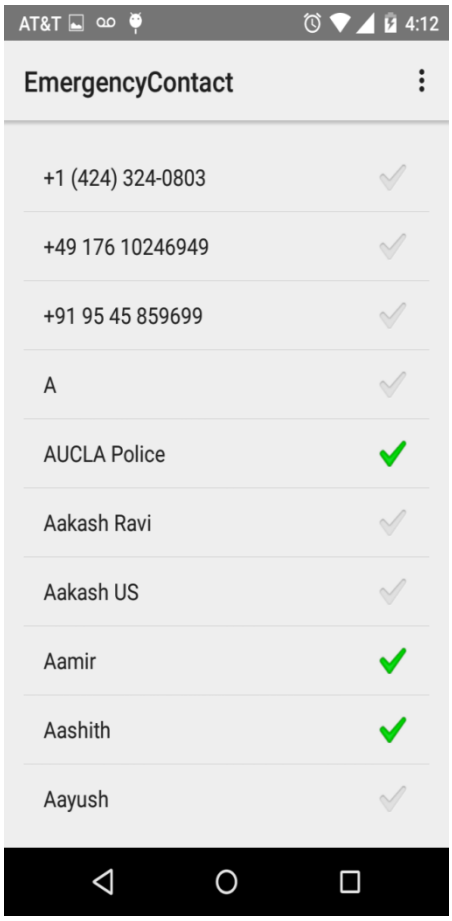

Figure 10: Select three contacts as emergency contacts.

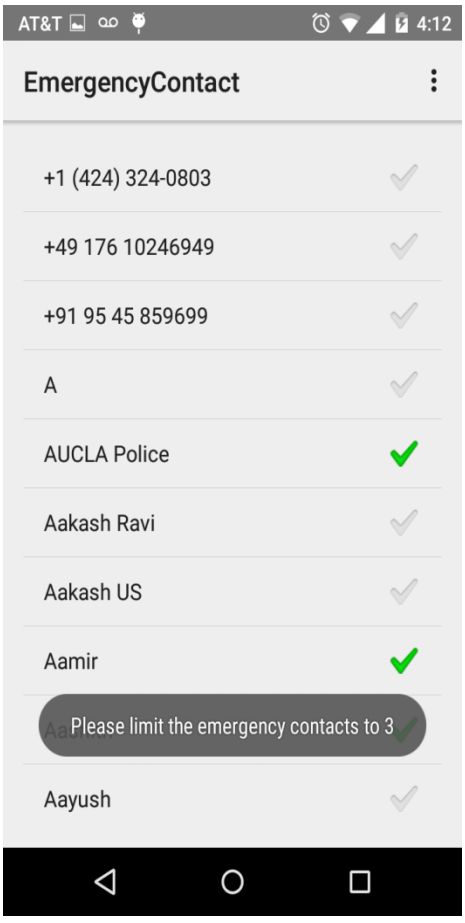

Figure 11: Maximum selection of 3 contacts

3.3 Safe Spot Search Feature

- As mentioned previously, a driver can be mentally distracted if he/she is expecting a post from someone and is unable to read it because of the blocking feature.
- Our app provides an option to find a safe spot and gives driving directions to that place. The type of place should be selected in the settings of the app.
- This place type is then invoked through a Google maps API which is integrated with our app.
- In the previous quarter, I had used Google Place Picker. While Place Picker gives a list of nearby places around the current location, the place picker API does not yet support a search by the type of place.
- The API is relatively new and work is still being done to make it much more convenient to be used in search queries and this includes adding a search by place type. In the future, if this feature is added, then Place Picker would be a better option as compared to Google Maps.
- Figures 13 and 14 show the selection of safe spot and the options displayed to the driver.

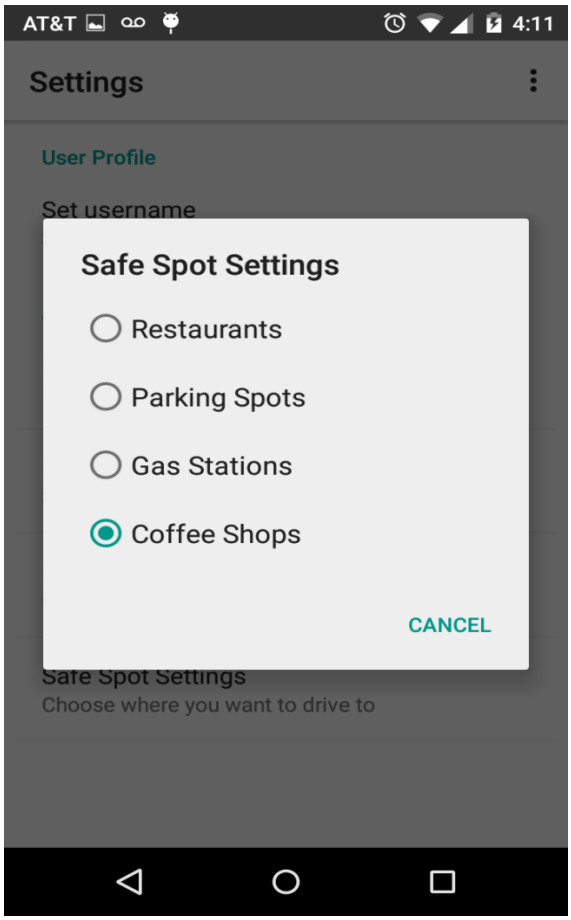

Figure 13: Select type of Safe Spot

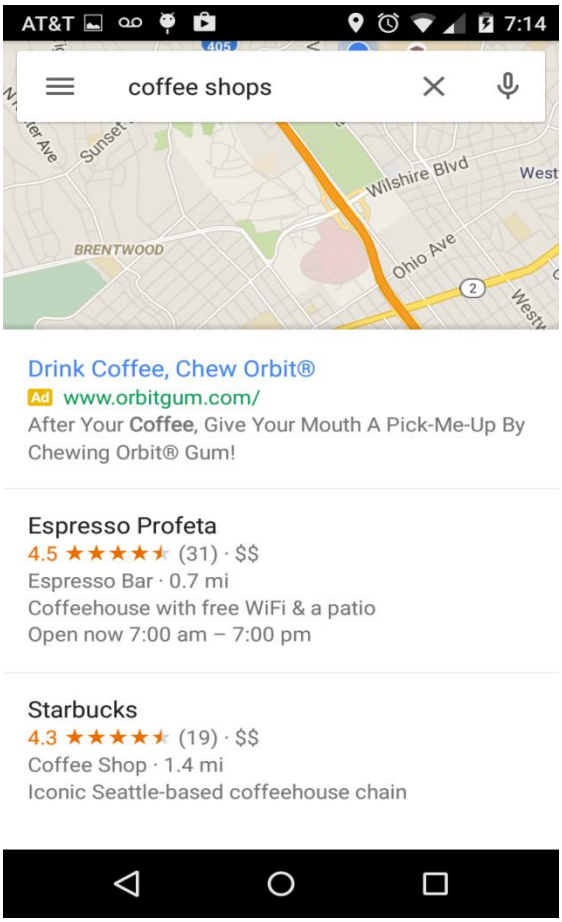

Figure 14: List displayed to user

3.4 Settings Screen

- For the settings screen, I have followed the typical template found in Android apps for ease of usability.
- Preference Manager of Android is used to create the UI shown in the figure below.
- Through the settings, the user is able to choose the emergency contacts that he wishes to call from the block screen.
- The user can also select which apps to block when the car is in motion.
- There is also an option to select the type of safe spot that the user wants to drive to once the app is blocked so that he/she can read his/her messages.
- Figure 15 shows a snapshot of the settings screen.

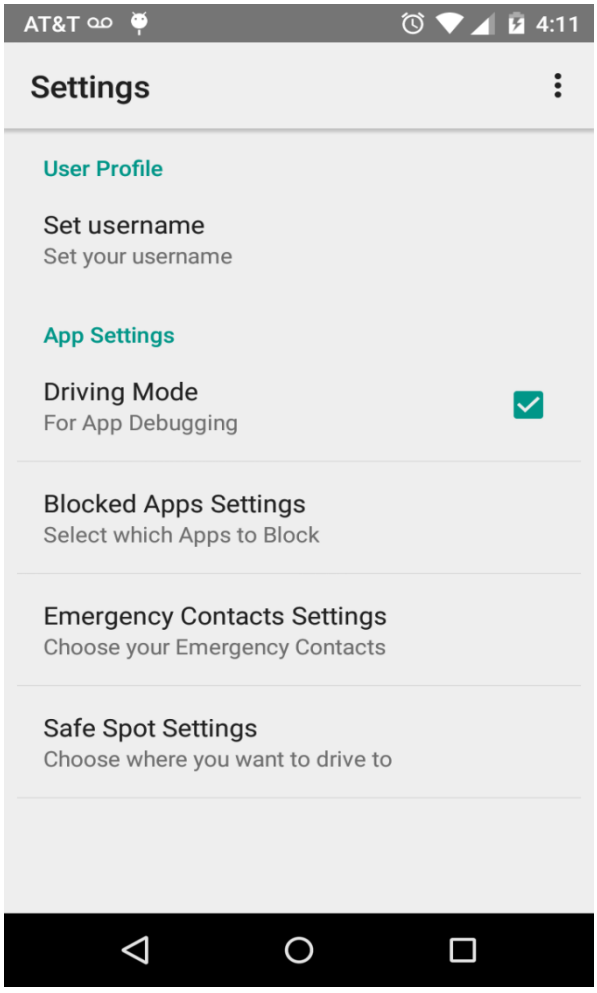

Figure 15: Settings Screen

3.5 Block Screen

- This is the screen that is drawn over when the user tries to open an app in a car which is in motion.
- This screen displays the emergency contacts and a button to safe spot search.
- This screen is displayed no matter how the app is opened; for example the messaging app can be opened through notification bar or directly. It will be blocked nevertheless.

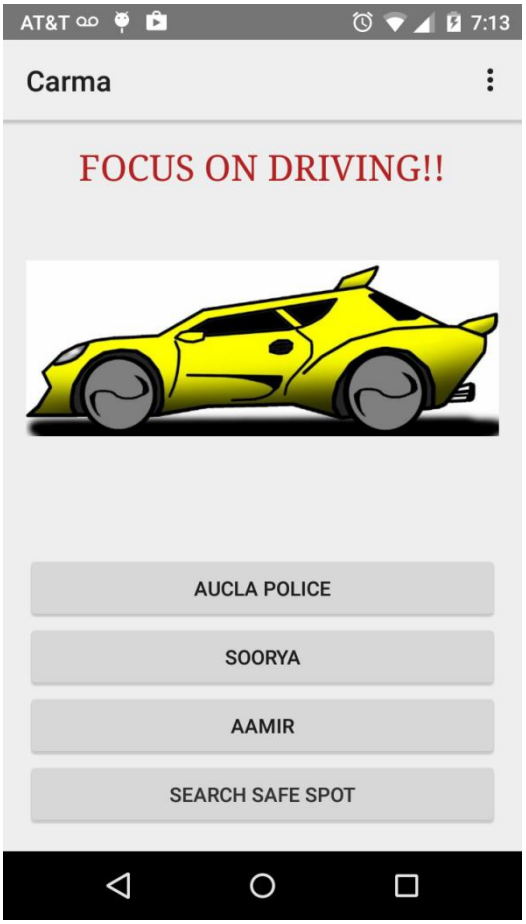

Figure 16 : Block Screen

3.6 Home Page/ Landing Screen

- This is the home page of the app and is the landing page.
- It currently dummy image of the speed summary events from the car.
- The Graph can be made using the graph API in android and I tested the skeleton code.

4. CONCLUSION & FUTURE WORK

Several reports suggest that now-a-days Texting during Driving is a greater hazard than drunk drivers [3]. An AT& T survey predicted that texting during driving kills more teens than drunk driving. In such a situation, the need for a product that prevents texting during driving cannot be stated enough.

Carma has a really good market, especially because there aren't many providers for blocking texting and even if there are, they haven't been accepted well. Carma can be targeted to parents of teens and adults alike. In the future, it is possible that the car manufacturers add this

component as a part of their build and then maybe we can greatly reduce the accidents due to texting.

Future Work

- **One change that is extremely feasible, useful and will be a great value addition is to capture the messages that arrive while the driver is driving and display a button to call the contact that messaged on the Block Screen. This would be much better than displaying pre-fixed emergency contacts. This idea struck me while I was testing the app and I looked into how it can be done. It is not very difficult and can be a highly valued feature.**
- Testing the integration with the hardware component. This will be done next quarter when the other team members are done with their parts.
- Summarizing the messages received from the notification bar and displaying it on the landing screen.
- Data that are relevant to the user's driving condition may be collected for future analysis. For example, the usual commuting route that the user is taking, the statistics of period when the user is driving, the frequency of receiving messages when the user is driving, etc.
- Since the distracted driving involving text messaging happens most among teenage drivers, it is necessary that the parents or guardians participate in the distraction control. Therefore, there is scope to develop a parental control app where will be installed on parents phone to provide some system controls.

5. ACKNOWLEDGEMENTS

For my work, I would like to acknowledge the help and guidance of Professor Majid Sarrafzadeh who has been an avid supporter and mentor through our work on this application. I would also like to acknowledge Nabil Alshurafa who was very available and accommodating for various hardware requests in Spring 2015.

This project wouldn't have been possible without my team-mates Josh and Heidi. Thanks guys for your feedback and the fun brainstorming sessions! Finally I would like to acknowledge the various classmates within our course that gave us feedback during our discussions.

6. REFERENCES

[1] Distraction.gov, 'Distracted Driving', 2015. [Online]. Available:http://distraction.gov. [Accessed: 31- May- 2015].

[2] Nhtsa.gov, 'Home | National Highway Traffic Safety Administration (NHTSA)', 2015. [Online]. Available: http://nhtsa.gov. [Accessed: 31-May- 2015].

[3] Cdc.gov, 'Teen Drivers: Get the Facts | Motor Vehicle Safety | CDC Injury Center', 2015.[Online].Available:http://www.cdc.gov/motorvehiclesafety/teen_drivers/teendrivers_factsheet.html. [Accessed: 31- May- 2015].

[4] Apple Inc., 'Driver handheld computing device lock-out', 8706143, 2014.

[5] Microsoft Corporation, 'Mobile device safe driving', 8874162, 2014.

[6] Qualcomm Incorporated, 'Preventing driver distraction', 9020482, 2015.

[7] Developer.android.com, 'BluetoothAdapter.LeScanCallback | Android Developers', 2015.[Online].Available:https://developer.android.com/reference/android/bluetooth/BluetoothAdapter.LeScanCallback.html. [Accessed: 31- May- 2015].

[8] Developer.android.com, 'LocationManager | Android Developers', 2015. [Online]. Available:http://developer.android.com/reference/android/location/LocationManager.html. [Accessed: 31- May- 2015].

[9] Google Developers, 'Google Places API Google Developers', 2015. [Online]. Available: https://developers.google.com/places/. [Accessed: 31- May- 2015].

[10] Obdedge, LLC, 'Systems, methods, and devices for policy-based control and monitoring of use of mobile devices by vehicle operators', 2010129939, 2010.

[11] Textinganddrivingsafety.com, 'Texting and Driving Statistics', 2015. [Online]. Available: http://www.textinganddrivingsafety.com/texting-and-driving-stats/. [Accessed: 31- May- 2015].

[12] Nsc.org, 'Distracted Driving: Problem of Cell Phone Distracted Driving', 2015. [Online]. Available: http://www.nsc.org/learn/NSC-Initiatives/Pages/distracted-driving-problem-of-cell-phone-distracted-driving.aspx. [Accessed: 31- May- 2015].