Telematics

Jerome Aganon, Jabneel Cruz, Jaewoong Kang, Achille Mattapullut

March 28th, 2017

Telematics

Project Build Website: http://Semjerome.github.io

Project Official website: http://www.autaccreport.com/wordpresstest/

Declaration of Joint Authorship

We, the Galaxy Noise, confirm that this work submitted for assessment is our own and is expressed in our own words. Any uses that was used in this documents that came from any other authors were acknowledged where their works were used. A list of references are included in this document and the work breakdown can be find on Overall Description Section 2.3.

Galaxy Noise: Jerome Aganon, Jabneel Cruz, Jaewhoong Kang, Achille Mattapullut

Date: February 6, 2017

Proposal

Submission January 17, 2017

Prepared by Galaxy Noise (Jerome, Jabneel, Jaewoong)

Computer Engineering Technology Student

https://semjerome.github.io

Executive Summary

As students in the Computer Engineering Technology program, we will be integrating the knowledge and skills we have learned from our program into this Internet of Things themed capstone project. This proposal requests the approval to build the hardware portion that will connect to a database as well as to a mobile device application. The internet connected hardware will include a custom PCB with sensors and /or actuators for detecting car collision and save the captured image/video in a black box The database will store the timestamp and the folder location of the video that were saved in the blackbox. The mobile device functionality will include notifying the car owner of an accident happening and if possible, an image or video capture of the accident and will be further detailed in the mobile application proposal. We will be collaborating with the following company/department Engineering Club. The hardware will be completed in CENG 317 Hardware Production Techniques independently and the application will be completed in CENG 319 Software Project. These will be integrated together in the subsequent term in CENG 355 Computer Systems Project as a member of a 2 or 3 student group.

Background

The problem solved by project is to not to prevent any accident but to get evidence if an accident happened. Having a dash cam does not mean that if an accident happened there would be footage of what happened. Sometimes, accident happens on the side or back of the car. In the event that such thing happened, the people will rely on the story of a witness or the victims. Sometimes the people will try to change their story in order for them to get away with it. If such thing happens, the police will have to look around and see if a CCTV has captured the accident.

I have searched for prior art via Humber's IEEE subscription selecting "My Subscribed Content" and have found and read which provides insight into similar efforts.

The first journal that we found presents " the approach to use smartphones as a multi sensor platform in a field operational test respectively a naturalistic driving study". (Pfriem & Gauterin, 2014)

The second journal talks about optical camera used for vehicles. (Nguyen, Islam, & Jang, 2016)

For the last journal, it then talks about inevitable accidents that can happen with motorcycles and automobiles. (Savino, Giovannini, Fitzharris, & Pierini, 2016)

In the Computer Engineering Technology program we have learned about the following topics from the respective relevant courses:

- Java Docs from CENG 212 Programming Techniques In Java
- Construction of circuits from CENG 215 Digital And Interfacing Systems,
- Rapid application development and Gantt charts from CENG 216 Intro to Software Engineering,
- Micro computing from CENG 252 Embedded Systems,
- SQL from CENG 254 Database With Java,
- Web access of databases from CENG 256 Internet Scripting; and,
- Wireless protocols such as 802.11 from TECH152 Telecom Networks.

This knowledge and skill set will enable me to build the subsystems and integrate them together as my capstone project.

Methodology

This proposal is assigned in the first week of class and is due at the beginning of class in the second week of the fall semester. My coursework will focus on the first two of the 3 phases of this project: Phase 1 Hardware build. Phase 2 System integration. Phase 3 Demonstration to future employers.

Phase 1 Hardware build

The hardware build will be completed in the fall term. It will fit within the CENG Project maximum dimensions of 12 13/16" x 6" x 2 7/8" (32.5cm x 15.25cm x 7.25cm) which represents the space below the tray in the parts kit. The highest AC voltage that will be used is 16Vrms from a wall adaptor from which +/- 15V or as high as 45 VDC can be obtained. Maximum power consumption will be 20 Watts.

Phase 2 System integration

The system integration will be completed in the fall term.

Phase 3 Demonstration to future employers

This project will showcase the knowledge and skills that I have learned to potential employers.

The tables below provide rough effort and non-labour estimates respectively for each phase. A Gantt chart will be added by week 3 to provide more project schedule details and a more complete budget will be added by week 4. It is important to start tasks as soon as possible to be able to meet deadlines.

Labour Estimates	**Hrs**	**Notes**
Phase 1		
Writing proposal.	9	Tech identification quiz.
Creating project schedule. Initial project team meeting.	9	Proposal due.
Creating budget. Status Meeting.	9	Project Schedule due.
Acquiring components and writing progress report.	9	Budget due.
Mechanical assembly and writing progress report. Status Meeting.	9	Progress Report due (components acquired milestone).

PCB fabrication.	9	Progress Report due (Mechanical Assembly milestone).
Interface wiring, Placard design, Status Meeting.	9	PCB Due (power up milestone).
Preparing for demonstration.	9	Placard due.
Writing progress report and demonstrating project.	9	Progress Report due (Demonstrations at Open House Saturday, November 12th, 2016 from 10 a.m. - 2 p.m.).
Editing build video.	9	Peer grading of demonstrations due.
Incorporation of feedback from demonstration and writing progress report. Status Meeting.	9	30 second build video due.
Practice presentations	9	Progress Report due.
1st round of Presentations, Collaborators present.	9	Presentation PowerPoint file due.
2nd round of Presentations	9	Build instructions up due.
Project videos, Status Meeting.	9	30 second script due.
Phase 1 Total	**135**	
Phase 2		
Meet with collaborators	9	Status Meeting
Initial integration.	9	Progress Report
Meet with collaborators	9	Status Meeting
Testing.	9	Progress Report
Meet with collaborators	9	Status Meeting
Meet with collaborators	9	Status Meeting
Incorporation of feedback.	9	Progress Report
Meet with collaborators	9	Status Meeting
Testing.	9	Progress Report
Meet with collaborators	9	Status Meeting

Prepare for demonstration.	9	Progress Report
Complete presentation.	9	Demonstration at Open House Saturday, April 8th, 2017 10 a.m. to 2 p.m.
Complete final report. 1st round of Presentations.	9	Presentation PowerPoint file due.
Write video script. 2nd round of Presentations, delivery of project.	9	Final written report including final budget and record of expenditures, covering both this semester and the previous semester.
Project videos.	9	Video script due
Phase 2 Total	**135**	
Phase 3		
Interviews	TBD	
Phase 3 Total	**TBD**	
Material Estimates	**Cost**	**Notes**
Phase 1		
RaspBerry Pi 3 Starter Kit	$89.99	https://www.amazon.com/Vilros-Raspberry-Ultimate-Starter-Kit-Clear/dp/B01CYWE20U
Pi Camera Module with Case	$38.79	https://www.amazon.com/Raspberry-Pi-Camera-Module-Megapixel/dp/B01ER2SKFS/ref=sr_1_2?s=pc&ie=UTF8&q 2&keywords=raspberry+pi+camera
Piezo Buzzer Element (Vibration Sensor)	$5.19	Canada Robotix
LED	$0.50	Canada Robotix
USB GPS Dongle.	$50	Amazon
Phase 1 Total	**>$200.00**	
Phase 2		
Materials to improve functionality, fit, and finish of project.		

Phase 2 Total	**TBD**
Phase 3	
Off campus colocation	<$100.00 An example: .
Shipping	*TBD*
Tax	*TBD*
Duty	*TBD*
Phase 3 Total	**TBD**

Concluding remarks

This proposal presents a plan for providing an IoT solution for mini computer and black box for car. This is an opportunity to integrate the knowledge and skills developed in our program to create a collaborative IoT capstone project demonstrating my ability to learn how to support projects such as the initiative described by our professor. I request approval of this project.

Abstract (Executive Summary)

In today's society, car has been part of our daily life. Based on National Safety Council, around 27,000 accidents occur everyday in United States alone but only 0.3 percent of it result in a fatality. The problem solved by our project is not to prevent any accident, but to get evidence if an accident happened. Having a dash cam does not mean that if an accident happened there would be footage of what happened. Sometimes, accident happens on the side or back of the car. In the event that such thing happened, the people will rely on the story of a witness or the victims. Sometimes the people will try to change their story in order for them to get away with it. If such thing happens, the police will have to look around and see if a CCTV has captured the accident. With our project, it will help the police or the Insurance Company to determine who is at fault if an accident occurs without any hassle. And at the same time, it will encourage the driver to abide the road rules since this project can be used against them if they do not.

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this documentation is to give a detailed description of the requirements for the ”Telematics” hardware and all of the software that comes with it. It will explain system constraints, system interface and interactions with other external applications. This documentation is primarily intended for our professor and his approval as well as reference for developing the system.

1.2 Scope

The ”Telematics” is a hardware that helps the police, insurance companies and drivers to determine the cause of a car incident. The hardware will have 3 cameras (one on each side and one on the rear). Although in this prototype, it will only have one camera because the raspberry pi3 can only support 1 raspberry pi camera at at ime. It will also have an Android application where the user can review any previous incident that has happened and add more information if needed.

Upon installing the hardware, the user has to register through the official website and enter the appropriate information and product key. From then on, the hardware will be able to send any information to the database once it is connected to the internet. The hardware does not need to maintain a network connection in order to work. Once the car starts, it will start recording and will replace the old file with a new file until it detects an accident. Once the hardware detects an incident, it will lock the file and will not be able to be overwritten until it is successfully uploaded to the server. Once the hardware is connected to the network, it will upload the video to the server and any information related to incident such as date and the location of the incident.

Furthermore, the user will be able to access the information to through their Android phone. From the moblie application, they will need to log in in order to retrieve the information related to their account. The user can then review the incident and add information about the incident such as, the car and driver’s information. If the user does not have an android phone, they can do everything that was mentioned above through the official website.

1.3 Definitions, acronyms, and abbreviations

Term	Definition
Telematics	Name of the hardware.
User	Someone who interacts with the web or software appplication
GPS	Global Positioning System
PHP file	PHP is a script language and interpreter that is freely available and used primarily on Linux Web servers. It lets the mobile application retrieve and add infomration from and into the database.
Database	a structured set of data held in a computer, especially one that is accessible in various ways. We used SQL databse for this project.
Java	Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is the main programming language that we used fro creating mobile application
Android Studio	Android Studio is the integrated development environment (IDE) from google. It was used for creating the mobile application.
IDE	is a software application that provides comprehensive facilities to computer programmers for software development.
Official Website	http://autaccreport.com/

2. Software Requirements Specification (SRS)

2.1 Technology Introduction

2.1.1 Purpose The purpose of our software is to provide an automated service for drivers that notifies them of any accidents or damages that may have occurred to their car. With the use of mobile application, the owner will have the data of the location and video feed of the accident. It will also serve as a data storage regarding the accident for the owner. They will be able to input additional information about the driver, the car, as well as information needed for the insurance company. A website will be available for the user to register an account for them to use the mobile application.

2.1.2 Product Overview The product will have piezo vibration sensors that will detect if there are any collisions detected. Raspberry Pi camera module will also be used to record video. Then the Raspberry Pi will process the video file and location information that will be sent to the web server that will be used by the mobile application.

2.1.3 Targeted Audience Group: Our target audience is any person that has a car and insurance company for they will be able to use the video feed as a proof of an incident.

2.2 Product Information

2.2.1 Main Functionality The main functionality of our system is to provide an incident report on user's mobile device in case of an accident. Information that the user saves on the application can then be used by them for their insurance company. The hardware will provide the video feed as well as the location of the report through the use of gps.

2.2.2 Extra Requirements Internet connectivity is required for the hardware device for it to be able to send the report, video, and location to the database. The Raspberry Pi will be powered by having it connected to the car itself.

2.2.3 Best Performance To achieve best performance of the product, we recommend sharing a stable wifi internet connection with the device.

2.3 Overall Description

2.3.1 System Interface System interface for our project includes: collision detector hardware, database server, and mobile application. If an accident is detected from the hardware, it sends an accident report to the server. The user through the use of mobile application in which he/she can retrieve the video file and location of the accident can then access this report. The user will also be able to delete the report (in case of false report), as well as add information to the report.

2.3.2 Database We are going to use the GoDaddy's database to store the information about the user, the accident, the driver they had an accident with and the car that the other driver was using. For the user infomration, we are going to store the username password and its user's id. For the password, we are going to encrypt the user's password using MD5 to protect the password just in case someone tries to hack our system. In accident table, we are going to the store the reporrtId, incident date, video name, longitude and latitude. In the driver table, we are going to store the other driver's license, first name,, last name, gender and insurance number. In the car table, we are going to store the other driver's car information such as plate number, brand, model and the year it was manufactured. (Jerome)

A separate database will be created for car makes and models, which is then going to be used for the mobile application. It will be used for drop-down spinners of both car makes and models. (Jabneel)

2.3.3 Hardware Our hardware is a collision detector. Once the car has start up, the hardware will start running immediately. It will start recording as soon as it is start up. It will have loop and lock functionality. With the loop functionality, the hardware will replace the old video with a new video, which prevents the hardware from using too much memory. The lock functionality will lock up the video file if the hardware detects any collision. It will lock up the file until the hard is connected to the Internet.(Jerome)

Once it is connected, it will upload the file to the server and store the date that the video was captured, the location and the name of the video. It will also have a GUI(Graphic USer Interface so that it will be

easy to use for new user.(Jaewoong)

2.3.4 Mobile Application Our mobile application is called AutAcc Report and is exclusive to Android. With the mobile app, the user is able to retrieve any incident related to the user. Once it is able to retrieve an incident report, the user will be able to add more information about the driver that he/she had an accident with and the car he/she was using. The application has playback functionality where it can watch the video that was captured during the incident. It also has map functionality where it will pinpoint the exact location of the incident by using Google map. (Jabneel)

The mobile app also has functionality where the user can contact his/her insurance agent where he/she can report the incident. User accounts and having access to different reports is also a requirement. (Jerome)

2.3.5 Web Application For our project, we will also provide a website. The front-end design and user interface of the website will be worked on by Jabneel and Achille. Usability and design of the website will require. Developed by (Jabneel) and user testing done by (Achille).

In the website, the user will be able to register. If the user is already registered, he/she will be able to review the information related to his/her account. He/she will also be able to delete an incident or modify any information related to that incident. The user can also change their password. This will be worked on by (Jerome).

2.4 Future Consideration

2.4.1 Operating Environment To fully utilize this product you will need an Android device (Tablet or phone) on the OS android 4.4 KitKat and above (API 19 and above). This product does not require constant internet connection. An Internet connection is only required when you need to synchronize the hardware with the database.

2.4.2 Safety Consideration Safety precaution that has to be considered when using the hardware:

Figure 1: alt text

- When using the hardware, make sure that the hardware is properly plugged in before turning on the car.
- Make sure that the hardware is secured and in place where there is no direct contact with sunlight.

2.4.3 Future Additions We are planning on creating a new case for the hardware since we are planning on adding a touchscreen capability on our hardware for portability and easy usability.

2.5 Project Schedule and Progress Status

2.5.1 Schedule

2.5.2 Status Report

[31/1/2017] Report by Jabneel Recent Project Activities: Our group, Telematics, have been working on our database, web server, website template, mobile application, as well as hardware improvement. We are currently on schedule in regards to our technical report and updating our Github.

The progress of work for each group member is as follows: Jabneel - I have been working on setting up a database for car models and car makes that will be used for the mobile application. We have recently bought the VPS Hosting so there is no database that has been set up yet, but I have done some research and found repositories that have databases that use JSON (most of our SQL format). I am also currently working on our website template. Jerome - He has bought the VPS hosting and been working on setting up and giving access to all the members so we can start with using/creating our database and website. Issues with GoDaddy customer service caused some delays during the weekend. Jaewoong - He is assigned to update the vibration sensor into a g-sensor which is what most dashcam uses. He is also assigned to buy a GPS module which is going to be used by the hardware to send coordinates/address to the database.

Financial Updates: Jerome and I have bought the VPS Hosting which costs around $260.00 and having access to it for 6 months. Jaewoong, on the other hand, has to spend money for the g-sensor as well as the GPS module.

[14/02/2017] Report by Jaewoong and Achille This is the reporting status of troubleshooting progress for our group, Telematics.

We are currently trying making the programs for GPS and server/database for Raspberry PI. In addition, we are also creating the GUI and website too.

Jaewoong Kang - I am mostly concentrating on GUI programming. We already set up several programs for each function such as recording video, catching event signal from sensor, and saving video files with specified event information. Now we need to merge these functions with Threading. Therefore, I also doing this at the same time. I have little problem caused by the broken part of Raspberry Pi, but it won't affect to programming, and we already decided to exchange this to Jabneel's device.

Achille Mattapullut - He is currently helping with the hardware part, In addition, he is also helping setting up the GPS on new device. He is mainly trying to finish up on the website front end. He had little delay caused by his job interview, but now he can come back to his part then it is expected to be able to catch up as our schedule.

Jabneel - He currently focuses on website security by using WordPress hash. He is also trying to finish his main works, PHP program for communication between mobile and database.

Jerome - He currently concentrates on the GPS receiving programming for Raspberry Pi. He is also helping making codes for server and Raspberry Pi communication by using Python.

Delay Issue and Troubleshooting: We have been a slight delay issues such as exchanging device caused by broken part issue, but we already start to set it up and it won't be long delay. In addition, we currently finished around 50% of technical report, so we will focus on finishing this in next week.

[07/03/2017] Report by Jabneel Recent Project Activities: Our group has recently finished setting up our website where the user can register an account (used for website and application). We also imported the database from Jerome's web host to ours. Hardware has recently gained access to GPS

module. The car models and brands have been imported into the Application instead of having to access server yo use it. This is to reduce coupling. There have been recent delays with database and progress in the application but this gave space to finishing the website.

The progress of work for each group member is as follows:
Jabneel- I am currently working on the integration of the website's registration system and the application. We want the app to have a link to the website's registration page then use that account for both application and website. Since WordPress encrypts the password of the user in the wp database, in the mobile app, we're gonna have to find a way to match user's input with what's in the database. Jerome will be helping me with this problem. Jerome- He has been working with the integration of the database as well as the website. Database to website's integration is working properly. The user can register and log in on the website and it is properly stored in the database. Since we recently imported the database from the previous Web host, the mobile application's links to PHP files should be changed. Jaewoong- He is working on hardware's integration with the database. Information that the hardware detects and creates should be sent to the database server. He is currently working on GUI for the sensor program. Achille- He has been doing some testing, proof reading and implementation on the website and database. This week, he will be doing more testing with the website's integration with the database after the hardware is able to send information to the database.

Financial Update: There is no financial update since my last email.

[20/03/2017] Report by Achille Achille Mattapullut - is currently helping with the hardware part, I am mainly helping setting up the GPS. I am rewriting some codes to make it compatible with our project. And finishing up on the website front end.

Jabneel Cruz - added password security by using WordPress hash and also updated our php file for android so that it will have password security too

Jerome Aganon - is currently working to add php file on our server so we can retrieve information from the raspberry pi.

Jaewoong Kang - will write a separate progress report for hardware.

Delay Issue: We are still behind in term of hardware. Pi2 camera connection part is broken.

[20/03/2017] Report by Jaewoong This is the reporting status of troubleshooting progress for our group, Telematics.

We are currently trying making the programs for GPS and server/database for Raspberry PI. In addition, we are also creating the GUI and website too.

Jaewoong Kang - I am mostly concentrating on GUI programming. We already set up several programs for each function such as recording video, catching event signal from sensor, and saving video files with specified event information. Now we need to merge these functions with Threading. Therefore, I also doing this at the same time. I have little problem caused by the broken part of Raspberry Pi, but it won't affect to programming, and we already decided to exchange this to Jabneel's device.

Achille Mattapullut - He is currently helping with the hardware part, In addition, he is also helping setting up the GPS on new device. He is mainly trying to finish up on the website front end. He had little delay caused by his job interview, but now he can come back to his part then it is expected to be able to catch up as our schedule.

Jabneel - He currently focuses on website security by using WordPress hash. He is also trying to finish his main works, PHP program for communication between mobile and database.

Jerome - He currently concentrates on the GPS receiving programming for Raspberry Pi. He is also helping making codes for server and Raspberry Pi communication by using Python.

Delay Issue and Troubleshooting: We have been a slight delay issues such as exchanging device caused by broken part issue, but we already start to set it up and it won't be long delay. In addition, we currently finished around 50% of technical report, so we will focus on finishing this in next week.

2.6 Hardware Build Instructions

2.6.1 Introduction The project will be able to detect car collision and save the recorded video into a file server to be used by the android application. The goal of this project is to notify the car owner of an accident happening. The owner should be able to access the video of the recorded accident using their mobile application. We are using a vibration sensor (buzzer) to detect an impact signal that will be sent to the raspberry pi which then would run the camera to record a 20 second video that will be sent to the

Figure 2: alt text

server. This video file can then be accessesed by the owner by using an android application. Take note that for now, our current production build is only sending the video file to the owner's email address.

2.6.2 System Diagram

Equipment	Quantity	Cost With tax
Raspberry Pi 3 Starter Kit	1	$89.99
Pi Camera Module	1	$38.79
Piezo Buzzer Element (Vibration Sensor)	2	$5.19
Camera Module Case	1	$25.28
1 Mega Ohm Resistor	1	$0.35
330 Ohm Resistor	1	$0.25
Full Sized Breadboard	1	$8.81
USB GPS Dongle	1	$50.00
LED	1	$0.25
Case	1	$17.00
TouchScreen	1	$71.80
Total		$282.04

2.6.3 Bills of Material / Budget

Todo	time Required
Raspberry Pi 3 power up	(30 mins) I will take around 30 min to set up the raspberry Pi. You have to install the OS and connect it to the internet.
PCB Soldering	1~2 hours for schematics and soldering. Printing the PCB could take 1 hour. And 30 min to solder it.
Project Wiring	15~30 mins
Importing Codes	1 hour, depends if you have all the required packages for using the camera and accessing email.

Figure 3: alt text

Running Code	5-10 mins, depends on the length of the video. The longer it is, the more time it needs to perform

2.6.4 Time Commitment

2.6.5.1 Mechanical Assembly This shows the circuit you have to build for the sensor to detect a vibration coming from tapping it. The LED will blink if the video file was sent to your email.

JP1 --- 2 is for the inner side of the buzzer. Positive. --- 1 outer part of the buzzer. Negative.

Following the diagram and placing each components in a breadboard would ensure a working vibration sensor program

Connect the particular camera module that I listed above to your Raspberry Pi 3.

Assembling the Bicolor Case for 7inch LCD #### Assembling Instruction

1. Assemble the Raspberry Pi on #1 board using #9, #10, and #11 (through the halls #2 on board #1)
2. Assemble board #2, touchscreen, and board #1 using #5, #6, and #7 ..a. Insert each #5 bolts through all six halls of board #2 ..b. Insert each #5 bolts into #6 bearings ..c. Assemble Touchscreen and above board #2 which is already assembled with #5 bolts ..d. Insert each #5 bolts into #6 bearings again ..e. Assemble board #1 and Touchscreen & board #2 which are already assembled in above steps ..f. Tighten the all #5 bolts all #7 nuts
3. Make the Screen Holder using #3, #4, #7, and #8 ..a. Assemble the headers of #8 bolts into each furrow of each side in #4 bars ..b. Assemble sides of two #4 bar and the furrows on #3 panels ..c. Tighten the #7 nuts

Product Detail Site : http://www.waveshare.com/product/Bicolor-Case-for-7inch-LCD.htm

Figure 4: alt text

Figure 5: alt text

Programming on Raspberry Pi

2.6.5.2 Hardware Application Programming In Raspberry Pi, we designed a GUI program which is separated by several parts; recording a video, catching an event signal from vibration sensor, uploading the video file, uploading the information in database, and multithreading code. We chose Python as programming language for these program because of several its benefits. • Python is very easy to learn • Support various systems and platforms • A card sized microcomputer, especially Raspberry Pi support to use many various resources such as Camera and many sensors. • There are a lot of available resources and open sources for Python

Python is littler slower and is not powerful at high graphic and memory intensive tasks, it is still great for our small device.

2.6.5.2.1 Setting for Pi Camera We already explained how to connect Pi Camera to Raspberry Pi, but in order to control the Pi camera by python, you must install python-picamera also. ~~~~ sudo apt-get update sudo apt-get install python-picamera ~~~~

If you are currently using Python3 package, you can install with below command ~~~~ sudo apt-get install python3-picamera ~~~~

2.6.5.2.2 Setting for Touch Screen You can buy Raspberry Pi LCD Touch Display (https://www.raspberrypi.org/pr pi-touch-display). It has great compatibility with Raspberry Pi and it is also easy to install and use. However, we chose the different touch screen, WaveShare LCD(c) which had IPS panel and more resolution with similar price, and it required below setting to use.

First, open “config.txt” file which is located in the root of Raspberry Pi TF card Second, add these 4 lines in “config.txt” file ~~~~ max_usb_current=1 hdmi_group=2 hdmi_mode=87 hdmi_cvt 1024 600 60 6 0 0 0 ~~~~

Third, save config.txt file and reboot Raspberry Pi Note: You must make sure that there are no spaces on either side of the equal sign. If the screen is displayed with not full size of touch

screen, it means that configuration is not completed properly. Link for more Information: http://www.waveshare.com/wiki/7inch_HDMI_LCD_(C)

2.6.5.2.3 Setting for GPS Receiver To use GPS USB receiver, you also should install GPS on raspberry Pi with following commands, ~~~~ sudo apt-get upgrade sudo apt-get install gpsd gpsd-clients python-gps ~~~~

If you finished installation of GPS, you also must modify the gpsd.hotplug file for setting serial port to access to the GPS. You can use the lines below to set up GPS serial port. ~~~~ nano /lib/udev/gpsd.hotplug

Scroll down the document and add chmod a+rw $DEVNAME above the line that says gpsdctl $ACTION $DEVNAME

Press CTRL and O together to save Press CTRL and X together to exit the file

/etc/init.d/gpsd restart gpsd /dev/ttyUSB0 -F /var/run/gpsd.sock ~~~~

If the operating system of your Raspberry Pi is the Raspberian Jessie not Wheezy, you wouldn't be albe to see any texts in gpsd.hotplug file. In this case, you can set up your Raspberry Pi by following method,

• Type following command on terminal ~~~~ sudo nano /etc/default/gpsd ~~~~

• Modify the gpsd as following lines START_DAEMON="true" GPSD_OPTIONS="n" DEVICES="/dev/ttyUSB0" USBAUTO="false" GPSD_SOCET="/var/run/gpsd.sock

• Save this, then restart the GPS with below command ~~~~ sudo /etc/init.d/gpsd restart ~~~~

If you finish setting it up, you can check GPS working with cgps -s command

2.6.5.2.4 Setting up Qt Design for GUI Programming There are various open source frameworks package for GUI programming by Python, but in this project, we will make GUI by PyQt, one of the most popular and powerful GUI packages. In order to use PyQt, you have to download and install it using following command ~~~~ sudo apt-get install python-qt4 ~~~~ In addition, we also need to install several tools, and you can install them with following command ~~~~ sudo apt-get install pytqt-dev-tools ~~~~

Reference video link: https://www.youtube.com/watch?v=ZcfUjxji-YA

2.6.5.2.5 Programming to Record Event Video and Catch Event

```
while True:
            GPIO.output(18,GPIO.HIGH)

            if isTrig == True:
              trig();
              #stopVideo()

            result = GPIO.input(5)
            if result == 1:
                print("Vibrated\n********************************\nRecording Video")
                isTrig =True
                vidName = time.strftime("%d%m%Y")+time.strftime("%I%M%S")
                recVideo(vidName)
            sleep(1)
            hname = vidName+"video.h264"
            mname = vidName+"video.mp4"
            print("Converting video file h264 to MP4");
            commands = "sudo MP4Box -add" + " "+hname + " " +mname
            os.system(commands)
                print("Sending video file to email");
            sendAlert(vidName)
            print("File sent");
```

~~~~~~

2.6.5.2.6 Programming to Upload a Video File into Server

We also have to make the code for uploading video files into server.

In this project, we will transfer our files to server by using Python Ftp Library
~~~~~~

import ftplib ~~~~ Basic command to connect to the server is, ~~~~ ftp_connection = ftplib.FTP(server, username, password) ~~~~ After above, we should define where the transferred file needs to be stored. ~~~~ ftp_connection.cwd("Target Directory Path") ~~~~ Finally, the below codes are for reading source and sending it into server. ~~~~ myfile = open("path/file", 'rb') ftp_connection.storbinary('STOR file', myfile) myfile.close() ~~~~ Note: ftplib is a built-in module in Python, so you don't have to install it

2.6.5.2.7 Programming to Insert Data into MySQL This application must update the information of database too whenever video files are uploaded into server. For this application, we need only INSERT method.

To use MySql-Python, you need to install MySQLdb module ~~~~ pip install MySQL-python ~~~~

Note: If your device doesn't have pip, you can install it with following command: ~~~~ For Python3 sudo apt-get install python3-pip For Python2 sudo apt-get install python-pip ~~~~

More information Link: https://www.raspberrypi.org/documentation/linux/software/python.md

Connecting SQL Python codes are very similar to PHP's codes. The below codes are the basic MySQLdb codes to use for database in Python: ~~~~ import MySQLdb db = MySQLdb.connect(Host Address, User ID, User Password, Database Name) cursor = db.cursor() cursor.execute("SQL Query") db.close() ~~~~

1st Line : Call the MySQLdb module from Python Libarary 2nd Line : Connect the target SQL 3rd Line : Declare cursor object using cursor() method 4th Line : Write the specified SQL Query command we want to use 5th Line : Close the database

2.6.5.2.8 Programming to make a Thread Function If the video record program start recording the video, the other program should have been waiting until this process is finished. In order to receive the event signal at the same time, we, therefore, need to correct these two functions as threading. Multithreading code in Python is very similar to other languages and its basic code is show in the following line: ~~~~ thread.start_new_thread (function, args[, kwargs]) ~~~~

kwargs is an optional dictionary of keyword arguments. For example, the threading code for the recording(location, time) and catching(port) functions are: ~~~~ thread.start_new_thread (recording, (di-

Figure 6: alt text

rectory, 60,)) thread.start_new_thread (chatching,36) ~~~~ or you can also use this, ~~~~ thread1 = recording(directory, 60) thread2 = catching(36) thread1.start() thread2.start() ~~~~ You shouldn't miss to add thread library too ~~~~ import thread ~~~~

2.6.5.2.9 GUI Programming with Qt Design We need to design a simple graphic interface because our devices don't support Keyboard input. Therefore, this GUI should involve the current status window and the basic 5 functions; starting recording video, stop recording, upload video to server, upload data into MySQL, and open the saving files directory. We already get this Qt Design when we installed PyQt.

```
1.Run Qt Design and open new project and file
2.Set up dialog size as around 400 x 300
3.Change window title to "Autacc Car Blackbox"
4.Add 5 of "Push Button" by Drag and Drop the button icon in Widget box
5.Change text of 5 buttons as functions' names
6.Add "Text Edit" with same way
7.Save this to ui format file
```

When you finish making a simple graphic user interface with Qt design, you can save a file with ui format. You should, therefore, convert it to python format, and this is the command for the converting. ~~~~ sudo pyuic4 input.ui -o output.py ~~~~

Note: If you are using PyQt5 not PyQt4, you can use below command instead of above. ~~~~ sudo pyuic5 input.ui -o output.py ~~~~

2.6.5.2.10 Programming of Event Handler for Graphic User Interface. We have made a simple GUI in previous step, but it was only for Graphical Interface such as button. Now we should add event handlers for each button. Here is the basic code for event handling. (Button's name).clicked.connet(Handler Name) For example, in case that we want call recordinfThread class

from testdrive.py whenever we clicked “Start Recording” Button, We can make these functions by adding following codes in GUI phython file 1.Add import testdrive

2.Declare the name for the recordingThread function Instance = testdrive.recordingThread(); 3.Make the connection between event handler and specified button which we made. pushButton.clicked.connet(instance)

Add all handler for each button are coded the same way.

2.6.6 PCB/Soldering We tried implementing our working circuit to a PCB but it didn’t work.

2.6.7 Unit Testing You can use these programs for testing your Camera and Vibration Sensor if they are working properly.

Camera

```
"
    from picamera import PiCamera
    from time import sleep

    camera = PiCamera()

    camera.start_preview()
    camera.start_recording('/home/pi/Desktop/Camera/video1.h264')
    sleep(5)
    camera.stop_recording()
    camera.stop_preview()
    "
```

Vibration Sensor

```
"
    from time import sleep
```

```
import RPi.GPIO as GPIO


GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.IN, pull_up_down=GPIO.PUD_DOWN )
sleep(1)


while True:
    result = GPIO.input(5)
    if result == 1:
        print("VIbrated")
    sleep(1)
"
```

2.6.8 Production Testing We wanted the video file recorded to be sent to a file server, but as of now , it is being sent to your email.

Main program

testDrive.py

```
"
from time import sleep
import RPi.GPIO as GPIO
import time
from picamera import PiCamera
from alertnoise import sendAlert
from cameraCode import initRecvideo, stopVideo, recVideo
import os


#initRecvideo()
```

```
# initialize sesnor and leds
GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.IN, pull_up_down=GPIO.PUD_DOWN )
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(18,GPIO.OUT)
sleep(1)
isTrig = False

def trig():
    print "LED on"
    GPIO.output(18,GPIO.HIGH)
    time.sleep(1)
    print "LED off"
    GPIO.output(18,GPIO.LOW)
    time.sleep(1)

print("Telematics~ running ...");
while True:
        GPIO.output(18,GPIO.HIGH)

        if isTrig == True:
          trig();
          #stopVideo()

        result = GPIO.input(5)
        if result == 1:
            print("Vibrated\n*******************************\nRecording Video")
            isTrig =True
```

```
            vidName = time.strftime("%d%m%Y")+time.strftime("%I%M%S")
            recVideo(vidName)
        sleep(1)
        hname = vidName+"video.h264"
        mname = vidName+"video.mp4"
        print("Converting video file h264 to MP4");
        commands = "sudo MP4Box -add" + " "+hname + " " +mname
        os.system(commands)
            print("Sending video file to email");
        sendAlert(vidName)
        print("File sent");
            "
```

NOTE: make sure to fill the values for raspaddress and useraddress. I removed it for obvious privacy reason.

Program that sends the recorded video file to your email

alertnoise.py

```
"
#Libraries necessary to send email using python
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText
from email.MIMEBase import MIMEBase
from email import encoders



#Email of the Raspberry Pi
raspaddress = ""
```

```
#User's Email address
useraddress = ""

#Structure of the email
msg = MIMEMultipart()
msg['From'] = raspaddress
msg['To'] = useraddress
msg['Subject'] = "Aegis Alert Message"



def alert1(vidName):
    body1 = "Movement detected in the room \n"
    filename = vidName+"video.mp4"
    openAtt = "/home/pi/Desktop/Final Presentation/"+filename
    attachment = open(openAtt, "rb")
    part = MIMEBase('application', 'octet-stream')
    part.set_payload((attachment).read())
    encoders.encode_base64(part)
    part.add_header('Content-Disposition', "attachment; filename= %s" % filename)
    msg.attach(part)
    return body1

def alert2():
    body2 = "Termperature Change Alert \n"
    return body2

def alert3():
    body3 = "Smoke Detected \n"
```

```
    return body3


def sendAlert(vidName):
    body = alert1(vidName)
    msg.attach(MIMEText(body, 'plain'))
    #Parameters for GMail Server
    server = smtplib.SMTP('smtp.gmail.com', 587)
    # Security function needed to connect to the Gmail server to protect the password.
    server.starttls()
    #Password of Raspberry Pi's Email
    server.login(raspaddress, "P@$$W0RD")
    #Sending the email
    text = msg.as_string()
    server.sendmail(raspaddress, useraddress, text)
    server.quit()
                "
```

Records the video for 20 seconds. Can be adjusted by changing the value of sleep

cameraCode.py

```
"
from picamera import PiCamera
from time import sleep


camera = PiCamera()


def initRecvideo():
        # initialize camera
        camera = PiCamera()
        camera.start_preview()
```

```
def stopVideo():
        camera.stop_preview()


def recVideo( sNum ):
    dest = '/home/pi/Desktop/Final Presentation/'+str(sNum)+'video.h264'
    camera.start_preview()
    camera.start_recording(dest)
    sleep(20)
    camera.stop_recording()
    camera.stop_preview()
    return;
"
```

2.7 Mobile Application Build Instructions

2.7.1 Introduction The mobile Application lets the user access their existing account. If the user is not registered, then they will have to register through online with the product key of their product. In mobile applicaion, the user is able to access all of the incident that they are involved with. At the same time, they are able to enter more information about the driver that they had an accident with and the car that the other drivier is using during that incident.

See section 2.5.2 System Diagram for more infomration about how mobile application interects with the server.

Reminder We removed the website name that we used for the php files for your own reproducibility. We used google maps api so make sure to visit their website to generate your OWN google_maps_key. Here is the link to the google's website for generating the google map key: https://developers.google.com/maps/documentation/javascript/get-api-key.

2.7.2 Mobile Application

MainActivity.java The opening display and login of the mobile application. Checks if the login information that the user inputs is a valid login. If it's a valid login, it then passes the information through intent to the next activity which is ReportList.java.

"

```
package galaxynoise.autaccreport;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
```

```
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;
import java.util.HashMap;

public class MainActivity extends AppCompatActivity {
    EditText username, password;
    String Name, Password;
    Context ctx=this;
    String NAME=null, PASSWORD=null, EMAIL=null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    username = (EditText) findViewById(R.id.etUserName);
    password = (EditText) findViewById(R.id.etPW);
    Button b1 = (Button) findViewById(R.id.btnLogin);
    b1.setOnClickListener(new View.OnClickListener() {
        @Override
    public void onClick (View v)
    {
        main_login();
    }
});
}

public void main_register(View v){
    startActivity(new Intent(this,ReportList.class));
}

public void main_login(){
    Name=null;
    Password=null;
    Name = username.getText().toString();
    Password = password.getText().toString();
    BackGround b = new BackGround();
    b.execute(Name, Password);
}

class BackGround extends AsyncTask<String, String, String> {
```

```
@Override
protected String doInBackground(String... params) {
    String name = params[0];
    String password = params[1];
    String data="";
    int tmp;

    try {
        URL url = new URL("http://websitename.com/app/login.php");
        String urlParams = "name="+name+"&password="+password;

        HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnec
        httpURLConnection.setDoOutput(true);
        OutputStream os = httpURLConnection.getOutputStream();
        os.write(urlParams.getBytes());
        os.flush();
        os.close();

        InputStream is = httpURLConnection.getInputStream();
        while((tmp=is.read())!=-1){
            data+= (char)tmp;
        }

        is.close();
        httpURLConnection.disconnect();

        return data;
    } catch (MalformedURLException e) {
        e.printStackTrace();
```

```
            return "Exception: "+e.getMessage();
        } catch (IOException e) {
            e.printStackTrace();
            return "Exception: "+e.getMessage();
        }
    }

    @Override
    protected void onPostExecute(String s) {
        String err=null;

        try {
            JSONObject root = new JSONObject(s);
            //JSONObject user_data = root.getJSONObject("User");
            JSONArray mainArray = root.getJSONArray("User");

            // looping through All Contacts
            for (int i = 0; i < mainArray.length(); i++) {
                JSONObject insideJsonObject = mainArray.getJSONObject(i);
                EMAIL=null;
                if (insideJsonObject != null) {
                    NAME = insideJsonObject.getString("username");
                    PASSWORD = insideJsonObject.getString("password");
                    EMAIL = insideJsonObject.getString("uid");
                    Log.d("Inside JSON uid: ", EMAIL);
                    Log.d("Inside JSON username: ", NAME);
                    Log.d("Inside JSON password: ", PASSWORD);
                }
            }
```

Figure 7: alt text

```
            } catch (JSONException e) {
                e.printStackTrace();
                err = "Exception: "+e.getMessage();
            }
            if(EMAIL!=null) {
                Toast.makeText(MainActivity.this, "Valid Login!", Toast.LENGTH_SHORT).sh
                Intent i = new Intent(ctx, ReportList.class);
                i.putExtra("name", NAME);
                i.putExtra("password", PASSWORD);
                i.putExtra("uid", EMAIL);
                i.putExtra("err", err);
                startActivity(i);
            }
            else{
                Toast.makeText(MainActivity.this, "Invalid Login or Password!", Toast.LE

            }
        }
    }
}

"
```

MainActivity, opening display and login activity for the user.

ReportList.java After the user logins, ReportList.java handles the collection of all the reports of incidents that the user has in the database. It connects to the web server then places each report in a listview

for the user. It displays the report id, data of incident, location of the incident (longitude and latitude), and the video name. ~~~~ " package galaxynoise.autaccreport; import android.content.DialogInterface; import android.content.Intent; import android.database.Cursor; import android.os.AsyncTask; import android.os.Bundle; import android.support.design.widget.FloatingActionButton; import android.support.design.widget.Snackbar; import android.support.v7.app.AlertDialog; import android.support.v7.app.AppCompatActivity; import android.support.v7.widget.Toolbar; import android.util.Log; import android.util.SparseBooleanArray; import android.view.View; import android.widget.AdapterView; import android.widget.ListAdapter; import android.widget.ListView; import android.widget.SimpleAdapter; import android.widget.TextView; import android.widget.Toast; import org.json.JSONArray; import org.json.JSONException; import org.json.JSONObject; import java.io.IOException; import java.io.InputStream; import java.io.OutputStream; import java.net.HttpURLConnection; import java.net.MalformedURLException; import java.net.URL; import java.util.ArrayList; import java.util.HashMap; import static android.R.attr.data; import static android.R.id.list;

public class ReportList extends AppCompatActivity { ListView listIncident; public static boolean isListempty =true; Intent currIntent; //PatientAdapter patientAdapter; ArrayList> infolist = new ArrayList<>(3); private String TAG = ReportList.class.getSimpleName(); TextView tvnews; String id;

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_report_list);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    String id=null, username ;
    final Intent testIntent = new Intent(getApplicationContext(), VidActivity.class);
    Intent intent = getIntent();
    Bundle b = intent.getExtras();
    if (b != null) {
```

```
        id =  (String)b.get("uid");
        username =  (String)b.get("name");
    }

    listIncident=(ListView) findViewById(R.id.listAll);
    tvnews=(TextView)findViewById((R.id.tvnews));
//  showIncidenttList(user);
    GetInfo info = new GetInfo();
    info.execute(id);
    listIncident.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView< ?> parent, View view, int position,long arg
            view.setSelected(true);
            //Toast.makeText(ReportList.this, "Selected!", Toast.LENGTH_SHORT).show();
            HashMap<String,String> map =(HashMap<String,String>)listIncident.getItemAtPo
            String reportid = map.get("reportid");
            String idate = map.get("incidentdate");
            String longi = map.get("longi");
            String lati = map.get("lati");
            String videoName = map.get("videoName");
            //String name= map.get(TAG_NAME);
            Toast.makeText(ReportList.this, "ReportId:"+ reportid+"is selected.", Toast.
            Intent i = new Intent(ReportList.this, VidActivity.class);
            i.putExtra("reportid", reportid);
            i.putExtra("incidentdate", idate);
            i.putExtra("longi", longi);
            i.putExtra("lati", lati);
            i.putExtra("videoName", videoName);
            startActivity(i);
```

```
        }
    });
}

/**
 * Async task class to get json by making HTTP call
 */
 class GetInfo extends AsyncTask<String, Void, Void> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        // Showing progress dialog

    }

    @Override
    protected Void doInBackground(String... params) {
        String uid = params[0];
        HTTPHandler sh = new HTTPHandler();
        String data="";
        int tmp;
        // Making a request to url and getting response
        //String uri = "http://websitename.com/app/incident.php";
        //String jsonStr = sh.makeServiceCall(uri);
        try {
            URL url = new URL("http://websitename.com/app/incident.php");
            String urlParams = "uid="+uid;
```

```
    HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnectior
    httpURLConnection.setDoOutput(true);
    OutputStream os = httpURLConnection.getOutputStream();
    os.write(urlParams.getBytes());
    os.flush();
    os.close();

    InputStream is = httpURLConnection.getInputStream();
    while((tmp=is.read())!=-1){
        data+= (char)tmp;
    }
    is.close();
    httpURLConnection.disconnect();


} catch (MalformedURLException e) {
    e.printStackTrace();
    //return "Exception: "+e.getMessage();
} catch (IOException e) {
    e.printStackTrace();
   // return "Exception: "+e.getMessage();
}

//Log.e(TAG, "Response from url: " + jsonStr);

if (data != null) {
    try {
        JSONObject mainJsonObject = new JSONObject(data);
        // Log.d("JSON Data : ", mainJsonObject.toString());
```

```
infolist.clear();
JSONArray mainArray = mainJsonObject.getJSONArray("Incident");
// Log.d("JSON Array : ", mainArray.toString());

for (int i = 0; i < mainArray.length(); i++) {

    JSONObject incidentJsonObject = mainArray.getJSONObject(i);

            if (incidentJsonObject != null) {

                String reportid = incidentJsonObject
                        .getString("reportid");
                Log.d("Inside JSON LongName : ", reportid);
                String incidentdate = incidentJsonObject
                        .getString("incidentdate");
                String longi = incidentJsonObject
                        .getString("longi");
                String lati = incidentJsonObject
                        .getString("lati");
                String videoName = incidentJsonObject
                        .getString("videoName");

                HashMap<String, String> info = new HashMap<>();
                info.put("reportid", reportid);
                info.put("incidentdate", incidentdate);
                info.put("longi", longi);
                info.put("lati", lati);
                info.put("videoName", videoName);
```

```
                            // adding contact to contact list
                            infolist.add(info);
                        }
                }

            } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

        }

        return null;
    }

    @Override
    protected void onPostExecute (Void result){
        super.onPostExecute(result);
        // Dismiss the progress dialog
        /**
         * Updating parsed JSON data into ListView
         **/
        if (infolist.size() > 0) {
            ListAdapter adapter = new SimpleAdapter(
                    ReportList.this, infolist,
                    R.layout.content_report_list, new String[]{"reportid", "incidentdate
                    R.id.tvdate, R.id.tvLongi, R.id.tvLati, R.id.tvVid});
            listIncident.setAdapter(adapter);
```

Figure 8: alt text

```
        }
        else
        {
            tvnews.setText(R.string.news);
        }
    }
}
@Override
public void onBackPressed() {
    new AlertDialog.Builder(this)
            .setTitle("Exit")
            .setMessage("You want to sign out?")
            .setNegativeButton(android.R.string.no, null)
            .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener

                public void onClick(DialogInterface arg0, int arg1) {
                    ReportList.super.onBackPressed();
                }
            }).create().show();
}
```

} " ~~~~ ReportList display layout

VidActivity.java Activity for all the fragments in PageFragment.java. Mainly used for the design and display in the viewpager as well as menu options on the top right hand side of the application. ~~~~ " package galaxynoise.autaccreport; //Team name Galaxy Noise import android.content.Context; import android.content.Intent; import android.graphics.drawable.Drawable; import android.net.Uri; import android.os.Bundle; import android.support.design.widget.TabLayout;

import android.support.v4.app.Fragment; import android.support.v4.app.FragmentManager; import android.support.v4.app.FragmentPagerAdapter; import android.support.v4.content.ContextCompat; import android.support.v4.view.ViewPager; import android.support.v7.app.AppCompatActivity; import android.view.Menu; import android.view.MenuItem; import android.view.View;

public class VidActivity extends AppCompatActivity implements View.OnClickListener { int drawables[] = { R.drawable.ic_car, R.drawable.ic_driver, R.drawable.ic_location, R.drawable.ic_video };

```
String reportid;
String incidentdate;
String longitude;
String latitude;
String videoName;

@Override
protected void onCreate(Bundle savedInstanceState) {
    //requestWindowFeature(Window.FEATURE_CUSTOM_TITLE);
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_vid);
    //getWindow().setFeatureInt(Window.FEATURE_CUSTOM_TITLE, R.layout.fragment_car);

    Intent i =  getIntent();

    ViewPager viewPager = (ViewPager) findViewById(R.id.viewpager);
    viewPager.setAdapter(new SampleFragmentPagerAdapter(getSupportFragmentManager(),
            VidActivity.this));

    // Give the TabLayout the ViewPager
    TabLayout tabLayout = (TabLayout) findViewById(R.id.sliding_tabs);
    tabLayout.setupWithViewPager(viewPager);
```

```
    tabLayout.getTabAt(0).setIcon(drawables[0]);
    tabLayout.getTabAt(1).setIcon(drawables[1]);
    tabLayout.getTabAt(2).setIcon(drawables[2]);
    tabLayout.getTabAt(3).setIcon(drawables[3]);

}

public String [] getFromReport()
{
    String fromReport [] = {reportid, incidentdate, longitude, latitude, videoName};
    return fromReport;
}

public class SampleFragmentPagerAdapter extends FragmentPagerAdapter {
    final int PAGE_COUNT = 4;
    private Context context;
    private String tabTitles[] = new String[] { "Car Info","Driver", "Location","Video"
    Drawable myDrawable;
    String title;

    public SampleFragmentPagerAdapter(FragmentManager fm, Context context) {
        super(fm);
        this.context = context;
    }

    @Override
    public int getCount() {

        return PAGE_COUNT;
```

```
    }

    @Override
    public CharSequence getPageTitle(int position)
    {
        switch (position) {
            case 0:
                return tabTitles[0];
            //return tabTitles[0];
            case 1:
                return tabTitles[1];
            //return tabTitles[1];
            case 2:
                return tabTitles[2];
            //return tabTitles[2];
            case 3:
                return tabTitles[3];
        }
        return "";
    }

    @Override
    public Fragment getItem(int position) {

        return PageFragmentVid.create(position + 1);
    }
}
@Override
// Handle the Back Key
```

```
public void onBackPressed() {
    finish();
}
@Override
public void onClick(View v) {

}
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    Intent intent = null;

    switch (menu.getItemId())
    {
        //Product website
        case R.id.about:
            intent = new Intent(Intent.ACTION_VIEW,
                    Uri.parse("http://websitename.com"));
            startActivity(intent);
            break;

        //Insurance auto claims phone number
        case R.id.CAA:
```

```
    try {
        intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" + "1-800-387-26
        startActivity(intent);
    } catch (Exception e) {
    }
    break;
case R.id.TD:
    try {
        intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" + "1-800-293-49
        startActivity(intent);
    } catch (Exception e) {

    }
    break;
case R.id.RBC:
    try {
        intent= new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" + "1-877-748-722
        startActivity(intent);
    } catch (Exception e) {

    }
    break;
case R.id.Desj:
    try {
        intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" + "1-888-776-83
        startActivity(intent);
    } catch (Exception e) {

    }
```

```
            break;
        case R.id.Farm:
            try {
                intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:" + "855-209-9549
                startActivity(intent);
            } catch (Exception e) {

            }
            break;
    }
    return super.onOptionsItemSelected(menu);
}

}
```

” ~~~~

PageFragmentVid.java This is where all the data are from the database are gathered and stored to be displayed by the mobile application. The fragments call the php files required to get information such as driver’s information, car details, coordinates of incident, and video file directory. The php files that we use, transforms the data in the database into a formattable JSON file where we extract all the information from. ~~~~ ” package galaxynoise.autaccreport; /** //Team name Galaxy Noise * Created by Zaido on 2016-11-13. */ import android.Manifest; import android.app.ProgressDialog; import android.content.Context; import android.content.Intent; import android.content.pm.PackageManager; import android.location.Address; import android.location.Geocoder; import android.net.Uri; import android.os.AsyncTask; import android.os.Bundle; import android.support.v4.app.ActivityCompat; import android.support.v4.app.Fragment; import android.support.v4.content.ContextCompat; import android.util.Log; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.AutoCompleteTextView; import android.widget.Button; import android.widget.EditText; import android.widget.ListAdapter; import android.widget.MediaController; import android.widget.SimpleAdapter; import android.widget.TextView; import android.widget.Toast;

import android.widget.VideoView; import com.google.android.gms.location.LocationServices; import com.google.android.gms.maps.CameraUpdateFactory; import com.google.android.gms.maps.GoogleMap; import com.google.android.gms.maps.MapView; import com.google.android.gms.maps.MapsInitializer; import com.google.android.gms.maps.OnMapReadyCallback; import com.google.android.gms.maps.model.CameraPosi import com.google.android.gms.maps.model.LatLng; import com.google.android.gms.maps.model.MarkerOptions; import org.json.JSONArray; import org.json.JSONException; import org.json.JSONObject; import java.io.BufferedInputStream; import java.io.BufferedReader; import java.io.IOException; import java.io.InputStream; import java.io.InputStreamReader; import java.io.OutputStream; import java.net.HttpURLConnection; import java.net.MalformedURLException; import java.net.ProtocolException; import java.net.URL; import java.util.ArrayList; import java.util.HashMap; import java.util.List;

public class PageFragmentVid extends Fragment { public static final String ARG_PAGE = ”ARG_PAGE”; AutoCompleteTextView actMake; AutoCompleteTextView actModel; AutoCompleteTextView actYear;

```
EditText etPN;
EditText  etBrand;
EditText  etModel;
EditText etYear;

EditText etFname;
EditText  etLname;
EditText  etLicense;
EditText etGender;
EditText etInsurance;

String [] myData;
private int mPage;

//Boolean for checking database
public static boolean isDriverEmpty;
public static boolean isCarEmpty;
```

```
String PLATENUMBER= null;
String CARMAKE= null;
String CARMODEL= null;
String CARYEAR = null;

String DRIVERLICENSE = null, FNAME=null,LNAME = null, GENDER = null, INSURANCENUMBER = r
String reportid;
//JSON
private ProgressDialog pDialog;
private String TAG = VidActivity.class.getSimpleName();

//Arraylist
ArrayList<HashMap<String, String>> carinfo = new ArrayList<>(3);
ArrayList<HashMap<String, String>> driverinfo = new ArrayList<>(3);

Button btnCarAdd;
Button btnDriverAdd;

View view;
// Identifier for the permission request
private static final int READ_CONTACTS_PERMISSIONS_REQUEST = 1;

//hello
public static PageFragmentVid create(int page) {
    Bundle args = new Bundle();
    args.putInt(ARG_PAGE, page);
    PageFragmentVid fragment = new PageFragmentVid();
    fragment.setArguments(args);
```

```
    return fragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    mPage = getArguments().getInt(ARG_PAGE);

}

GoogleMap googleMap;
MapView mMapView;
MarkerOptions markerOptions;
TextView tvReverseGeo;
public String addressText;

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {

    VidActivity activity = (VidActivity) getActivity();
    myData = activity.getFromReport();
    reportid =myData[0];
    /*
        0 = reportid , 1 = incidentdate, 2 = longi, 3 = lati, 4 = vidname
     */
    if(mPage==1) //Car page
    {
        view = inflater.inflate(R.layout.fragment_car, container, false);
```

```
    etPN = (EditText) view.findViewById(R.id.etPN);
    etBrand= (EditText) view.findViewById(R.id.etBrand);
    etModel= (EditText) view.findViewById(R.id.etModel);
    etYear = (EditText) view.findViewById(R.id.etYear);

    actMake = (AutoCompleteTextView) view.findViewById(R.id.actMake);
    actModel = (AutoCompleteTextView) view.findViewById(R.id.actModel);
    actYear = (AutoCompleteTextView) view.findViewById(R.id.actYear);

    String carJson = loadCarJsonLocal();
    try {
        JSONObject json = new JSONObject(carJson);
    } catch (Exception e) {
        e.printStackTrace();
    }

    btnCarAdd = (Button)view.findViewById(R.id.btnCarAdd);

    btnCarAdd.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            executeAddCar();
        }
    });

    GetCar b = new GetCar();
    b.execute(reportid);
}
```

```
else if(mPage==2)//Driver
{
    view= inflater.inflate(R.layout.fragment_driver,container, false);
    String reportid = myData[0];
    etLicense= (EditText) view.findViewById(R.id.etLicense);
    etFname = (EditText) view.findViewById(R.id.etFname);
    etLname= (EditText) view.findViewById(R.id.etLname);
    etGender = (EditText) view.findViewById(R.id.etGender);
    etInsurance = (EditText) view.findViewById(R.id.etInsurance);
    btnDriverAdd =(Button)view.findViewById(R.id.btnDriverAdd);

    btnDriverAdd.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            executeDriverAdd();
        }
    });

    GetDriver g = new GetDriver();
    g.execute(reportid);
}
else if(mPage==3)
{ //location
    /*
    0 = reportid , 1 = incidentdate, 2 = longi, 3 = lati, 4 = vidname
 */
    view = inflater.inflate(R.layout.fragment_eventlocation, container, false);
    tvReverseGeo = (TextView)view.findViewById(R.id.tvReverseGeo);
```

```
mMapView = (MapView) view.findViewById(R.id.mapView);
mMapView.onCreate(savedInstanceState);

mMapView.onResume();

try{
    MapsInitializer.initialize(getActivity().getApplicationContext());
}catch(Exception e){
    e.printStackTrace();
}

mMapView.getMapAsync(new OnMapReadyCallback() {
    @Override
    public void onMapReady(GoogleMap mMap) {
        googleMap = mMap;
        if(checkPermission())
        googleMap.setMyLocationEnabled(true);

        double longi = Double.parseDouble(myData[2]);
        double lati = Double.parseDouble(myData[3]);
        /*toronto
        double lati = 43.579028;
        double longi = -79.746524; */
        LatLng event = new LatLng(lati,longi);
        markerOptions = new MarkerOptions();

        Log.d("tv: ","not set");
        new ReverseGeoCodingTask(getContext()).execute(event);
        String snippetAddress = tvReverseGeo.getText().toString();
```

```
            googleMap.addMarker(markerOptions.position(event)
                    .title("Event Location")
                    .snippet(snippetAddress));

            CameraPosition cameraPosition = new CameraPosition.Builder().target(ever
            googleMap.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosi
        }

        private boolean checkPermission()
        {
            return (ContextCompat.checkSelfPermission(getActivity(), Manifest.permis
                    == PackageManager.PERMISSION_GRANTED);
        }

    });
}
else if (mPage == 4)
{ //video
    view = inflater.inflate(R.layout.fragment_eventvideos, container, false);
    String vidname = myData[4];
    VideoView mVideoView  = (VideoView) view.findViewById(R.id.videoView);
    MediaController mediaController = new MediaController(getActivity());
    mediaController.setAnchorView(mVideoView);
    mVideoView.setMediaController(mediaController);
    String myPackage= "galaxynoise.autaccreport";
    Uri uri = Uri.parse("http://www.websitename.com/Android/"+vidname+".mp4");
    try{
       mVideoView.setVideoURI(uri);
        mVideoView.start();
```

```
        }catch(Exception e){
            Log.e("Error", e.getMessage());
            e.printStackTrace();
        }
    }

    else
    {
        view = inflater.inflate(R.layout.fragment_pager, container, false);
        TextView textView = (TextView) view;
        textView.setText("Fragment #" + mPage);
    }

    return view;
}

public String loadCarJsonLocal()
{
    String json = null;
    try {

        InputStream is = getContext().getAssets().open("car_makemodel.json");
        int size = is.available();
        byte[] buffer = new byte[size];
        is.read(buffer);
        is.close();
        json = new String(buffer, "UTF-8");

    } catch (IOException ex) {
```

```
            ex.printStackTrace();
            return null;
        }

        return json;

    }
    private class ReverseGeoCodingTask extends AsyncTask<LatLng, Void, String>{
        Context mContext;
        public ReverseGeoCodingTask(Context context){
            super();
            mContext = context;
        }

        @Override
        protected String doInBackground(LatLng... params){
            Geocoder geocoder  = new Geocoder(mContext);
            double latitude = params[0].latitude;
            double longitude = params[0].longitude;

            List<Address> addresses = null;

            try {
                addresses = geocoder.getFromLocation(latitude, longitude,1);
            } catch (IOException e) {
                e.printStackTrace();
            }

            if(addresses!= null && addresses.size() > 0 ) {
```

```
            addressText = addresses.get(0).getAddressLine(0) +"," +
                    addresses.get(0).getLocality() +"," + addresses.get(0).getPostalCode
                    ", " +addresses.get(0).getCountryName();
        }

        return addressText;

    }

    @Override
    protected void onPostExecute(String addressText) {
        // Setting the title for the marker.
        // This will be displayed on taping the marker
        // Placing a marker on the touched position
        tvReverseGeo.setText(addressText);

    }
}

/**
 car make and model reference, arthurkao, github repository,
 https://github.com/arthurkao/vehicle-make-model-data
 **/
void executeAddCar()
{
    PLATENUMBER = etPN.getText().toString();
    CARMAKE = etBrand.getText().toString();
    CARMODEL = etBrand.getText().toString();
    CARYEAR = etYear.getText().toString();
```

```
    AddCar a = new AddCar();
    a.execute(PLATENUMBER, CARMAKE, CARMODEL, CARYEAR,reportid);
}

class AddCar extends AsyncTask<String, String, String> {
    @Override
    protected String doInBackground(String... params) {
        String platenumber = params[0];
        String carmake = params[1];
        String carmodel = params[2];
        String caryear = params[3];
        String reportid = params[4];
        String data="";
        int tmp;

        try {
            URL url;

            if(isCarEmpty==true) {
                url = new URL("http://websitename.com/app/addCar.php");
                isCarEmpty=false;
            }
            else
            {
                url = new URL("http://websitename.com/app/updateCar.php");
            }

            String urlParams = "platenumber="+platenumber+"&carmake="+carmake+"&carmodel
                    +carmodel+"&caryear="+caryear+"&reportid="+reportid;
```

```
            HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection
            httpURLConnection.setDoOutput(true);
            OutputStream os = httpURLConnection.getOutputStream();
            os.write(urlParams.getBytes());
            os.flush();
            os.close();

            InputStream is = httpURLConnection.getInputStream();
            while((tmp=is.read())!=-1){
                data+= (char)tmp;
            }

            is.close();
            httpURLConnection.disconnect();

            return data;
        } catch (MalformedURLException e) {
            e.printStackTrace();
            return "Exception: "+e.getMessage();
        } catch (IOException e) {
            e.printStackTrace();
            return "Exception: "+e.getMessage();
        }
    }

    @Override
    protected void onPostExecute(String s) {
        String err=null;
```

```
try {
    JSONObject root = new JSONObject(s);
    //JSONObject user_data = root.getJSONObject("User");

    JSONObject car= root.getJSONObject("Car");

    car.put("platenumber", PLATENUMBER);
    car.put("carmake",CARMAKE);
    car.put("carmodel",CARMODEL);
    car.put("caryear",CARYEAR);
    car.put("reportid",reportid);

    Log.d("Storing to JSON plate: ", PLATENUMBER);
    Log.d("Storing to JSON make: ", CARMAKE);
    Log.d("Storing to JSON model: ", CARMODEL);



} catch (JSONException e) {
    e.printStackTrace();
    err = "Exception: "+e.getMessage();
}
if(PLATENUMBER!=null) {
    Toast.makeText(getActivity(), "Stored!", Toast.LENGTH_SHORT).show();
}
else{
    Toast.makeText(getActivity(), "Fill in plate number", Toast.LENGTH_LONG).sho

}
```

```
    }
}

//get Car
class GetCar extends AsyncTask<String, Void, Void> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        // Showing progress dialog

    }

    @Override
    protected Void doInBackground(String... params) {
        String reportid = params[0];
        HTTPHandler sh = new HTTPHandler();
        String data = "";
        int tmp;
        // Making a request to url and getting response
        //String uri = "http://websitename.com/app/incident.php";
        //String jsonStr = sh.makeServiceCall(uri);


        try {
            URL url = new URL("http://websitename.com/app/getCar.php");
            String urlParams = "reportid=" + reportid;

            HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnectior
```

```
    httpURLConnection.setDoOutput(true);
    OutputStream os = httpURLConnection.getOutputStream();
    os.write(urlParams.getBytes());
    os.flush();
    os.close();

    InputStream is = httpURLConnection.getInputStream();
    while ((tmp = is.read()) != -1) {
        data += (char) tmp;
    }

    is.close();
    httpURLConnection.disconnect();


} catch (MalformedURLException e) {
    e.printStackTrace();
    //return "Exception: "+e.getMessage();
} catch (IOException e) {
    e.printStackTrace();
    // return "Exception: "+e.getMessage();
}

//Log.e(TAG, "Response from url: " + jsonStr);

if (data != null) {
    try {
        JSONObject mainJsonObject = new JSONObject(data);
        // Log.d("JSON Data : ", mainJsonObject.toString());
```

```
JSONArray mainArray = mainJsonObject.getJSONArray("Car");
// Log.d("JSON Array : ", mainArray.toString());

for (int i = 0; i < mainArray.length(); i++) {

    JSONObject incidentJsonObject = mainArray.getJSONObject(i);

    if (incidentJsonObject != null) {

        String platenumber = incidentJsonObject
                .getString("platenumber");
        String carmake = incidentJsonObject
                .getString("carmake");
        String carmodel = incidentJsonObject
                .getString("carmodel");
        String caryear = incidentJsonObject
                .getString("caryear");

        HashMap<String, String> info = new HashMap<>();
        info.put("platenumber", platenumber);
        info.put("carmake", carmake);
        info.put("carmodel", carmodel);
        info.put("caryear", caryear);

        // adding contact to contact list
        carinfo.add(info);
    }
```

```
            }

        } catch (JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    return null;
}

@Override
protected void onPostExecute(Void result) {
    super.onPostExecute(result);
    // Dismiss the progress dialog

    /**
     * Updating parsed JSON data into ListView
     **/
    if (carinfo.size() > 0) {
        etPN.setText(carinfo.get(0).get("platenumber").toString());
        etBrand.setText(carinfo.get(0).get("carmake").toString());
        etModel.setText(carinfo.get(0).get("carmodel").toString());
        etYear.setText(carinfo.get(0).get("caryear").toString());
        isCarEmpty=false;
    }
    else
```

```
        {
            isCarEmpty=true;
        }

    }
}

void executeDriverAdd()
{
    //DRIVERLICENSE = null, FNAME=null,LNAME = null, GENDER = null, INSURANCENUMBER = nu
    DRIVERLICENSE = etLicense.getText().toString();
    FNAME = etFname.getText().toString();
    LNAME = etLname.getText().toString();
    GENDER = etGender.getText().toString();
    INSURANCENUMBER = etInsurance.getText().toString();

    AddDriver a = new AddDriver();
    a.execute(DRIVERLICENSE, FNAME, FNAME, GENDER,INSURANCENUMBER,reportid);
}

//ADDING NEW DATA FOR DRIVER
class AddDriver extends AsyncTask<String, String, String> {

    @Override
    protected String doInBackground(String... params) {
        String driverlicense = params[0];
        String fname = params[1];
        String lname = params[2];
        String gender = params[3];
```

```
String insurancenumber = params[4];
String reportid = params[5];
String data = "";
int tmp;
URL url;
try {
    if(isDriverEmpty==true) {
         url = new URL("http://websitename.com/app/addDriver.php");
        isDriverEmpty=false;
    }
    else
    {
         url = new URL("http://websitename.com/app/updateDriver.php");
    }
    String urlParams = "driverlicense=" + driverlicense + "&fname=" + fname + "&
            + lname + "&gender=" + gender + "&insuranceNumber=" + insurancenumbe

    HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnectior
    httpURLConnection.setDoOutput(true);
    OutputStream os = httpURLConnection.getOutputStream();
    os.write(urlParams.getBytes());
    os.flush();
    os.close();

    InputStream is = httpURLConnection.getInputStream();
    while ((tmp = is.read()) != -1) {
        data += (char) tmp;
    }
```

```
            is.close();
            httpURLConnection.disconnect();

            return data;
        } catch (MalformedURLException e) {
            e.printStackTrace();
            return "Exception: " + e.getMessage();
        } catch (IOException e) {
            e.printStackTrace();
            return "Exception: " + e.getMessage();
        }
    }

    @Override
    protected void onPostExecute(String s) {
        String err = null;
        //Log.e(TAG, "Response from url: " + s);
        try {
            JSONObject root = new JSONObject(s);
            //JSONObject user_data = root.getJSONObject("User");
            JSONObject car = root.getJSONObject("Driver");

            car.put("driverlicense", DRIVERLICENSE);
            car.put("fname", FNAME);
            car.put("lname", LNAME);
            car.put("gender", GENDER);
            car.put("insuranceNumber", INSURANCENUMBER);
            car.put("reportid", reportid);
```

```
        } catch (JSONException e) {
            e.printStackTrace();
            err = "Exception: " + e.getMessage();
        }
        if (DRIVERLICENSE != null) {
            Log.d("Storing to JSON plate: ", DRIVERLICENSE);
            Log.d("Storing to JSON make: ", FNAME);
            Log.d("Storing to JSON model: ", LNAME);
            Toast.makeText(getActivity(), "Stored!", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(getActivity(), "Fill in plate number", Toast.LENGTH_LONG).sho


        }
    }
}
    //get Car
    class GetDriver extends AsyncTask<String, Void, Void> {


        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            // Showing progress dialog


        }


        @Override
        protected Void doInBackground(String... params) {
            String reportid = params[0];
            HTTPHandler sh = new HTTPHandler();
```

```
String data = "";
int tmp;
// Making a request to url and getting response
//String uri = "http://websitename.com/app/incident.php";
//String jsonStr = sh.makeServiceCall(uri);


try {
    URL url = new URL("http://websitename.com/app/getDriver.php");
    String urlParams = "reportid=" + reportid;

    HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnec
    httpURLConnection.setDoOutput(true);
    OutputStream os = httpURLConnection.getOutputStream();
    os.write(urlParams.getBytes());
    os.flush();
    os.close();

    InputStream is = httpURLConnection.getInputStream();
    while ((tmp = is.read()) != -1) {
        data += (char) tmp;
    }

    is.close();
    httpURLConnection.disconnect();


} catch (MalformedURLException e) {
    e.printStackTrace();
```

```
    //return "Exception: "+e.getMessage();
} catch (IOException e) {
    e.printStackTrace();
    // return "Exception: "+e.getMessage();
}

//Log.e(TAG, "Response from url: " + jsonStr);

if (data != null) {
    try {
        JSONObject mainJsonObject = new JSONObject(data);
        // Log.d("JSON Data : ", mainJsonObject.toString());


        JSONArray mainArray = mainJsonObject.getJSONArray("Driver");
        // Log.d("JSON Array : ", mainArray.toString());

        for (int i = 0; i < mainArray.length(); i++) {

            JSONObject incidentJsonObject = mainArray.getJSONObject(i);

            if (incidentJsonObject != null) {

                String driverlicense = incidentJsonObject
                        .getString("driverlicense");
                String fname = incidentJsonObject
                        .getString("fname");
                String lname = incidentJsonObject
                        .getString("lname");
```

```
                    String gender = incidentJsonObject
                            .getString("gender");
                    String insurancenumber = incidentJsonObject
                            .getString("insuranceNumber");

                    HashMap<String, String> info = new HashMap<>();
                    info.put("driverlicense", driverlicense);
                    info.put("fname", fname);
                    info.put("lname", lname);
                    info.put("gender", gender);
                    info.put("insuranceNumber", insurancenumber);
                    // adding contact to contact list
                    driverinfo.add(info);
                }

            }

        } catch (JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }

    return null;
}

@Override
```

```
        protected void onPostExecute(Void result) {
            super.onPostExecute(result);
            // Dismiss the progress dialog


            /**
             * Updating parsed JSON data into ListView
             **/
            if (driverinfo.size() > 0) {
                etLicense.setText(driverinfo.get(0).get("driverlicense").toString());
                etFname.setText(driverinfo.get(0).get("fname").toString());
                etLname.setText(driverinfo.get(0).get("lname").toString());
                etGender.setText(driverinfo.get(0).get("gender").toString());
                etInsurance.setText(driverinfo.get(0).get("insuranceNumber").toString())
                isDriverEmpty=false;
            }
            else
            {
                isDriverEmpty=true;
            }
        }
    }
// Called when the user is performing an action which requires the app to read the
    // user's contacts
    public void getPermissionToReadUserContacts() {
        // 1) Use the support library version ContextCompat.checkSelfPermission(...) to
        // checking the build version since Context.checkSelfPermission(...) is only ava
        // in Marshmallow
        // 2) Always check for permission (even if permission has already been granted)
        // since the user can revoke permissions at any time through Settings
```

Figure 9: alt text

Figure 10: alt text

```
        if (ContextCompat.checkSelfPermission(getActivity(), Manifest.permission.READ_CO
                != PackageManager.PERMISSION_GRANTED) {
            // The permission is NOT already granted.
            // Check if the user has been asked about this permission already and denied
            // it. If so, we want to give more explanation about why the permission is n
            // Fire off an async request to actually get the permission
            // This will show the standard permission request dialog UI
        }
    }
```

} " ~~~~ Each page fragments: Driver's Information, Car Information, Location, Video file

2.6.8 Back End

Login.php This is the php file that is going to get executed when the user tries to login with their mobile application.

```
<?php
error_reporting(0);
require "init.php";

$name = $_POST["name"];
$password = $_POST["password"];
```

Figure 11: alt text

Figure 12: alt text

```
$sql = "SELECT * FROM `User` WHERE `username`='".$name."' AND `password`='".$password."'

$result = mysqli_query($con, $sql);

$response = array();

while($row = mysqli_fetch_array($result)){
    $response[0] = array("uid"=>$row[0],"username"=>$row[1],"password"=>$row[2]);
}

echo json_encode(array("User"=>$response));

?>
```

Incident.php Once the user is able to login, this is first file that it will run in order to retrieve all of the incident related to the user's profile. It will use the user's id as reference when running the following code.

```
<?php
error_reporting(0);
require "init.php";

$uid = $_POST["uid"];



$count=0;
$sql = "SELECT * FROM `Incident` WHERE `uid`='".$uid."';";
```

```
$result = mysqli_query($con, $sql);

$response = array();

while($row = mysqli_fetch_array($result)){
    $response[$count] = array("reportid"=>$row[0],"incidentdate"=>$row[1],"longi"=>$row[
$count++;
}

echo json_encode(array("Incident"=>$response));

?>
```

GetCar.php When the user selects an incident from the list that is provided to them, this is one of the two php file that is going to get executed. This php file will retrieve any information about the car that the other driver was using during the incident. It will use the incident id as reference when searching for the car's information. If there was no recorded information about the car that the other driver was using, it will then return an empty json object.

```
<?php
error_reporting(0);
require "init.php";

$reportid = $_POST["reportid"];

$sql = "SELECT * FROM `Car` WHERE `reportid`='".$reportid."';";

$result = mysqli_query($con, $sql);
```

```
$response = array();

while($row = mysqli_fetch_array($result)){
    $response[0] = array("platenumber"=>$row[0],"carmake"=>$row[1],"carmodel"=>$row[2],'
}

echo json_encode(array("Car"=>$response));

?>
```

GetDriver.php When the user selects an incident from the list that is provided to them, this is one of the two php file that is going to get executed. This php file will retrieve any information about driver that ths user had an accident with. It will use the incident id as reference when searching for the driver's information. If there was no recorded information about the other driver, it will then return an empty json object.

```
<?php
error_reporting(0);
require "init.php";

$reportid = $_POST["reportid"];


$sql = "SELECT * FROM `Driver` WHERE `reportid`='".$reportid."';";

$result = mysqli_query($con, $sql);

$response = array();

while($row = mysqli_fetch_array($result)){
```

```
    $response[0] = array("driverlicense"=>$row[0],"fname"=>$row[1],"lname"=>$row[2],"ger
}

echo json_encode(array("Driver"=>$response));

?>
```

UpdateCar.php and AddCar.php When the user adds the car information, it will use one of thsese two application depending on the current situation. If the database does not hold any existing car infomration about the selected incident, it will use the AddCar.php. On the other hand, if the database does hold an information about the car, it will use the UpdateCar.php. We created two different php file because add and update uses different sql statement.

AddCar.php

UpdateCar.php

UpdateDriver.php and AddDriver.php When the user adds the dirver information, it will use one of thsese two application depending on the current situation. If the database does not hold any existing driver infomration about the selected incident, it will use the AddDriver.php. On the other hand, if the database does hold an information about the driver, it will use the UpdateDriver.php. We created two different php file because add and update uses different sql statement.

AddDriver.php

UpdateDriver.php ~~~~ ?php error_reporting(0); require ”init.php”;

$driverlicense = $_POST[”driverlicense”]; $fname = $_POST[”fname”]; $lname = $_POST[”lname”]; $gender = $_POST[”gender”]; $insuranceNumber = $_POST[”insuranceNumber”]; $reportid = $_POST[”reportid”];

$sql = ”UPDATE‘Driver‘SET‘driverlicense‘ =′ ”.driverlicense.”’,fname=”’.fname.”′,‘lname‘ =′ ”.lname.”’,gender=”’.gender.”′,‘insuranceNumber‘ =′ ”.insuranceNumber.”’ WHERE reportid=”’.reportid.”′;”;if(!m $sql)){ echo ’{”message”:”Unable to save the data to the database.”}’; }

?> ~~~~

Test Number	Test Case Title	test Case Purpose	Preconditions to run the test case	Steps	Expected Result
AndroidApp-01	LogIn Validation	$Verify that the Validation is working.	Startup the app	1. Run tha Application 2.Click login without entering anything	The application will ask you to enter a valid usernamne and password.
AndroidApp-02	Database Connection	Verify that the application is able to conenct through database.	Startup the app	1. Run the app. 2.Enter a valid username and password. username= jabneel password=password	Go to the next activity and retrieve any incident list related to the account.
AndroidApp-03	Incident Retrieve Information	Verify that it retrieves information related to the incident such as driver and car.	Should be on reportist class.	1. Select incident 0 on the list.	Should have an existing data about the driver and the vehicle
AndroidApp-04	Map	Verify that the map is working	Should be on Information Display	1. Select Location tab	Should pinpoint on 871 Bramble Court Mississauga

AndroidApp-05	Video	Verify that the video is working	Should be on Information Display	1. Select Video tab	Should play a video
AndroidApp-06	Change Information	Verify that the driver and car infomrattion can be change	Should be on Information Display	1. Select Car Info 2. Enter a different information 3.Save 4. Select Driver info 5. Enter a different information 6. Save	Information should be permanently changed
AndroidApp-07	Add Information	Verify that you can add car and driver information	Should be on Information Display that has no car or driver infomration	1. Select Car Info 2. Enter a different information 3.Save 4. Select Driver info 5. Enter a different information 6. Save	New information should be stored in the database

2.6.8 Mobile Application Test Case

3.Conclusion

In this report, we wrote the purpose of our project and how to reproduce it. By following the build instructions for the hardware and software, you should be able to reproduce it individually and should run on how it is supposed to. Finishing the connectivity of the hardware device to the database would ensure a working project. On the other hand, it is not possible to create a working PCB as there are some circuitry issues that has never been fixed.

4.Recommendation

When reproducing this hardware, it is recommended to get a Raspberry Pi 3 or later version. It is not recommended to use an earlier version because it does not have any built-in wi-fi or bluetooth capability.

To increase the reliability of the hardware, it is advicable to use Gravity sensor instead of vibration sensor. By using Gravity sensor, the video will only be lock when there is a disruption on the movememnt of the car. On the other hand, vibration sensor will lock the video everytime it sense a vibration.

For Android Developers, I would recommend to have and create each fragments for video, location, driver and car in separate page fragments for clear readability and easy programmability of the application.

5.Reference

References (Generated in pdf)

Nguyen, T., Islam, A., & Jang, Y. M. (2016). Region-of-interest signaling vehicular system using optical camera communications. *IEEE Photonics Journal*, *PP*(99), 1–1. https://doi.org/10.1109/JPHOT.2016.2644960

Pfriem, M., & Gauterin, F. (2014). Employing smartphones as a low-cost multi sensor platform in a field operational test with electric vehicles. In *2014 47th hawaii international conference on system sciences* (pp. 1143–1152). https://doi.org/10.1109/HICSS.2014.148

Savino, G., Giovannini, F., Fitzharris, M., & Pierini, M. (2016). Inevitable collision states for motorcycle-to-car collision scenarios. *IEEE Transactions on Intelligent Transportation Systems*, *17*(9), 2563–2573. https://doi.org/10.1109/TITS.2016.2520084