ASHESI UNIVERSITY

IMPLEMENTING PRIVACY-BY-DESIGN IN AN IOT-CONTROLLED INSURANCE SYSTEM

THESIS PROJECT

B.Sc. Computer Science

HODO A. S.

2021

ASHESI UNIVERSITY

PRIVACY-BY-DESIGN FOR INTERNET OF THINGS(IOT): IMPLEMENTING PRIVACY-BY-DESIGN IN AN IOT-CONTROLLED INSURANCE SYSTEM

THESIS CAPSTONE

PROJECT

Capstone Project submitted to the Department of Computer Science, Ashesi University, in partial fulfilment of the requirements for the Bachelor of Science degree in Computer Science award.

Albert Senyo Hodo

2021

DECLARATION

I hereby declare that this capstone is the result of my original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

....

Candidate's Name:

....

Date:

....

I hereby declare that the preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University.

Supervisor’s Signature:

....

Supervisor’s Name:

....

Date:

....

Acknowledgements

First of all, I like to appreciate the Holy Spirit for His guidance throughout this project.

To all the people whose encouragement and academic advice helped me undertake this project, I declare my earnest appreciation. I wish to express my gratitude to my supervisor Mr Francis Gatsi who guided me through this project.

I would also like to thank my family and friends who supported me and offered deep insights into the study. I wish to extend my special thanks to Mr William Albert Kyei, whose contribution and patience was helpful in this project.

I wish to acknowledge the help provided by the Faculty of the Computer Science department from Ashesi University, whose feedback and input was instrumental in the completion of this project. Lastly, to all my classmates who supported me during this project, I say thank you.

Abstract

Usage-Based Insurance (UBI) is an emerging trend in the insurance industry where insurance premiums packages of vehicles are calculated by the driving behaviour of its drivers to be more personalized. However, traditional IoT systems do not give users full capabilities to authorize and/or deauthorize the collection of user data at any point in time. This creates a problem of the absence of user autonomy in IoT systems. This is problematic because law enforcing documents such as the General Data Protection Regulation (GDPR) and other documents by Engineering Institutions such as the ARM AI Manifesto states that user autonomy and user data rights must be considered when developing such systems, and if we do not start now, valuable technologies such as these will not be adopted in certain parts of the world. The project seeks to tackle this problem by testing the feasibility of a design-by-privacy approach in an IoT-controlled Insurance, as well as study the effects such a system will have on the systems that are highly dependent on data. In this study, a stripped-down version of the entire IoT-Controlled Insurance was created with privacy-by-design, and the results were captured and analyzed. The results prove that it is feasible, and more studies must be done to enhance the efficiency of the system when data is interrupted by the user.

Table of Contents

Chapter 1: Introduction

1.1 Background

Internet of Things (IoT) is a term used to describe the interconnectivity of smart objects(things) that collect data, process the data, communicate with each other, and actuate other objects in the environment. The integration of the Internet of Things into various sectors is improving lives daily. An example of such an industry is the insurance sector. Intelligent Transportation Systems (ITS), Usage-Based Insurance(UBI) etc. are a few examples of new trends in the car insurance business[1]. UBI calculates insurance premiums based on sensor data collected from OBD systems and is used for a more personalized policy. Its main benefits are reduced car accidents, lower premiums, and better risk calculation, better cost savings for companies and a lot more.[2]

1.2 Problem Statement

The major problem with the UBI and other IoT controlled insurance systems is the user privacy concerns[2], [3]. Most IoT systems that involve humans lack robust user privacy features that can ensure users' safety and privacy. Even though IoT is based on constrained networks and systems, it is fundamentally still information Technology(IT). All the risks and attack scenarios in the past concerning IT must be reconsidered in these systems[4]. The main challenge is the privacy aspect because IoT does not present the same interfaces or protocols to the typical IT systems. Users cannot precisely tell what data is being taken from them and have little or no say in it after the systems have been deployed.

1.3 Motivation

Ethical issues about the IoT controlled environment might prevent the adoption of this technology in many countries. Users might also not patronize such systems if robust privacy profiles are not created for them because they would not want the general public, government agencies or insurance companies having their location and other personal data[2].

With the emergence of digital laws and ethical regulations such as the GDPR, which is the core of Europe's digital privacy legislation, such rights and other ethical considerations must be factored in when developing IoT systems. When we consider section 3 of the GDPR that talks about human Rights to Rectification and Erasure[5], most or all IoT systems do not have robust features to exercise such rights to their full capabilities, and this is significant. This research also is motivated by the Arm AI manifesto that provides guidelines for working with AI and user data and provides frameworks for engineers to use and take responsibility for user privacy and AI ethical issues[6]. Suppose IoT system designers don't begin to include such features. In that case, there is the risk of IoT systems not being adopted in most parts of the world as other governments are also coming up with similar Digital Rules.

1.4 Aims and Objectives

This research aims to find out how we can implement privacy-by-design in an IoT-controlled insurance system and find out the effects that privacy by design will have on such a system. This will be done by borrowing the website design concept that allows users to get access to certain features of websites even if they did not accept cookies and the users' data is not stored. This research will also find efficient machine learning algorithms that can be used on sensor data from the vehicle that does not directly relate to the user or can be used against the user.

This will be done by focusing on non-driver dependent data so as to provide users with different privacy and control options that would not be affected by the core data used in this research.

The research will be focused on these **objectives**;

- To find out how we can implement privacy-by-design in an IoT controlled insurance system
- To find out the effects that privacy by design will have on such a system

The **objectives** will be addressed by the following questions in the research;

1. What are the mechanisms for implementing privacy by design in an IoT-controlled insurance system?
2. What is the effect of implementing privacy-by-design on the accuracy of classifying a user as risky or not risky?

Chapter 2: Literature Review

Research in the field of IoT controlled Insurance has been conducted over the years. It has mainly taken different forms, ranging from Usage-Based Insurance, Intelligent Transportation System, Driver Behaviour Recognition etc. [2], [7]. One interesting aspect is that, aside from proving concepts and theories, no user privacy model has been implemented yet.

2.1 Existing Literature

Advancement in UBI has provided different theories on how to mitigate the privacy issues concerned with the general model that collects data, transfers it to a service provider, which then calculates the driving score and sends it to the insurance companies. The general solution tried to mitigate the issues about privacy by allowing the service provider to handle all the data and only send statistics and a score to the insurance company. The general model has lots of benefits; it improves driving behaviour over time, allows the company to save money due to better risk calculation, etc. [2]. That is a good step, but now the GDPR, Ghana Data Protection Act(DPA) and others that are being developed by countries state that it is not enough and makes us responsible for the user's data protection rights. Users cannot control their data flow; they sometimes do not have access to their own data and cannot access their rights to be forgotten[5].

The diagram below shows the general model. Solutions have been proposed to the problem of the general model but have not been implemented yet. [2], [8] Researchers have proposed that the data be processed locally in the vehicle and aggregate data sent to the insurer for premium calculation.[8]proposed a privacy-enhancing architecture called PEAR's which highlights the

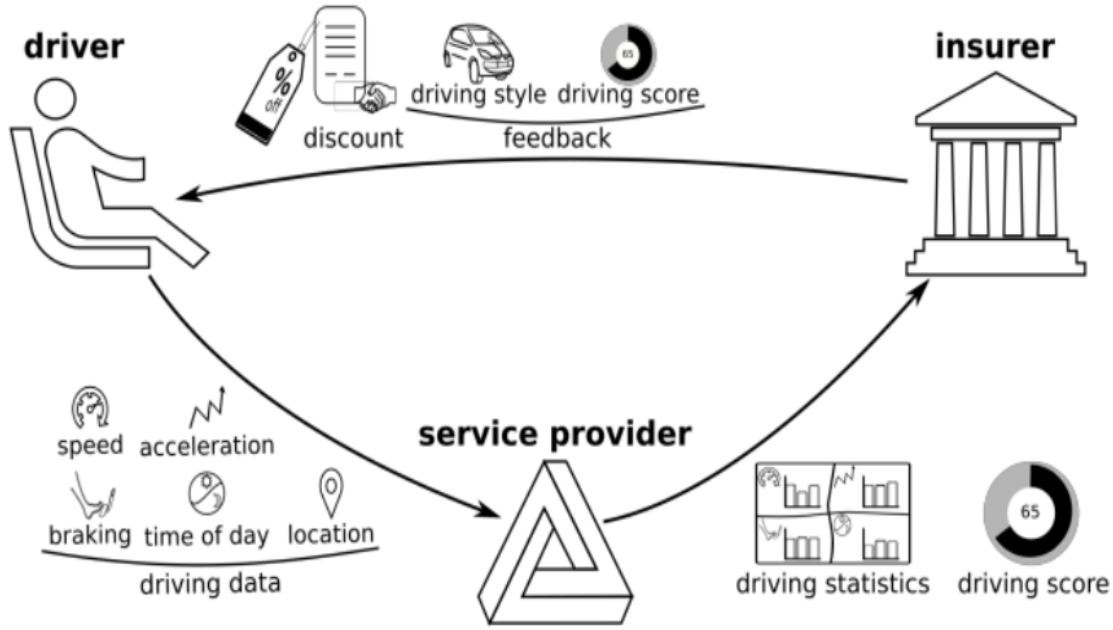


Figure 2.1: Privacy-enhanced architecture

importance of design architecture in developing privacy by design solutions.

In terms of classifying the data gotten from the sensors, [7] in one research, car sensor data was collected to find which data best explained the driver behaviour based on the car's performance. Over 50 sensor data was collected and tested with various algorithms.

Popular machine learning models were tested with the data. From the testing done, the top 3 were Random Forests, Decision Tree and Gradient Boosting with over 95%, with Random Forests being the highest reaching 97.5%, while the others performed poorly on the kind of vehicular data provided. The top 10 sensor data were chosen based on high correlation. More tests were done, and the top 3 sensor data that was representative of the drivers' performance and was difficult to manipulate by car owners were identified [7]. The research did not consider user involvement and how the algorithm would act if users decided to stop the flow of data for some time or chose to hold on to some sensor data.

2.2 Focus of Study

In this research, the main aim is to protect the users' data. From the literature review and much research, if this research implements a way to allow users control over the data, there would be a great tradeoff between privacy and efficiency, hence affecting the program's usefulness if users do not allow data to be sent. At the core, if a model is created such that specific data needs to be collected and the user chooses not to allow the flow of such data, then the system would not function. In that regard, the research solves this issue by borrowing the logic from websites that use cookies. This analogy was used; "If a user chooses to use the system and give out all data necessary, then the program works. However, if the user does not choose to give out sensitive data, then the program should still work based on general data that does not relate to the user directly or cannot be used to deduce the users' personal data." Hence the program would not compromise so much on efficiency since it should provide a fair idea of the users driving behaviour indirectly and still protects the user's privacy.

Chapter 3: Methodology

For the research, a prototype system was built to implement and validate the concept of privacy by design. Machine learning algorithms were tested with the data. The data was also paused at different times to simulate the users stopping data flow, and the effects were studied.

3.1 Architectural Design

The standard three (3) layered IoT architecture was used in this study. Ie.

- The IoT device layer - The client-side. This consists of sensors and actuators.
- The IoT gateway layer - The server-side. Operations on pre-processing and aggregation.
- The IoT platform layer - Connecting clients and operations as well as further analysis.[9]

IoT architecture is also divided into four stages. These are; a. The sensor and actuator stage. b. Data acquisition stage, c. Edge IT and d. Data Center and Cloud. The first two stages will be done by the car On-Board Diagnostics (OBD II) system. The already existing car sensors will be used, and the OBD system will capture the data and present it to a module that is attached to the port. The module does some pre-processing on the data to aggregate and filter the data for security purposes. That forms the IoT device layer. In this research, a car OBD simulator will be used to manage cost and accessibility. The module then sends the aggregated data to the mobile phone of the user via Bluetooth. This is the second layer which consists of the edge IT stage. The App on the phone allows the user to visualize the data, know what data is being sent, and control the flow of the data. The information is then sent to the data centre stage, which falls in the IoT platform layer. This is where the machine learning algorithms work on the data to calculate the driving score and behaviour category to be presented to the insurance company. The research is done in 3 parts.

Simulating IoT system. This is the first stage of the study. This is where the OBD simulator was used to generate the data, which was aggregated and sent to the phone. The phone's App made it possible for the user to visualize the data, allow control, select what profile of data they wanted to be sent or stopped the App totally. The internet was used to transfer the data to the cloud for the machine learning algorithms and a final transfer of information to the insurance company.

Sensor data selection and machine learning testing. The second stage of the research. At this stage, the sensor data was divided into direct and indirect user-dependent data, respectively. Several machine learning models were tested, and the correlation threshold was set to 0.98. A supervised quantitative algorithm and a classification algorithm were chosen for the system based on performance. IF the user allowed all the necessary data, the first algorithm was used, but if the user did not allow the flow of some of the data, then the classification algorithm was used to give a categorical sense of what the user driving metrics is.

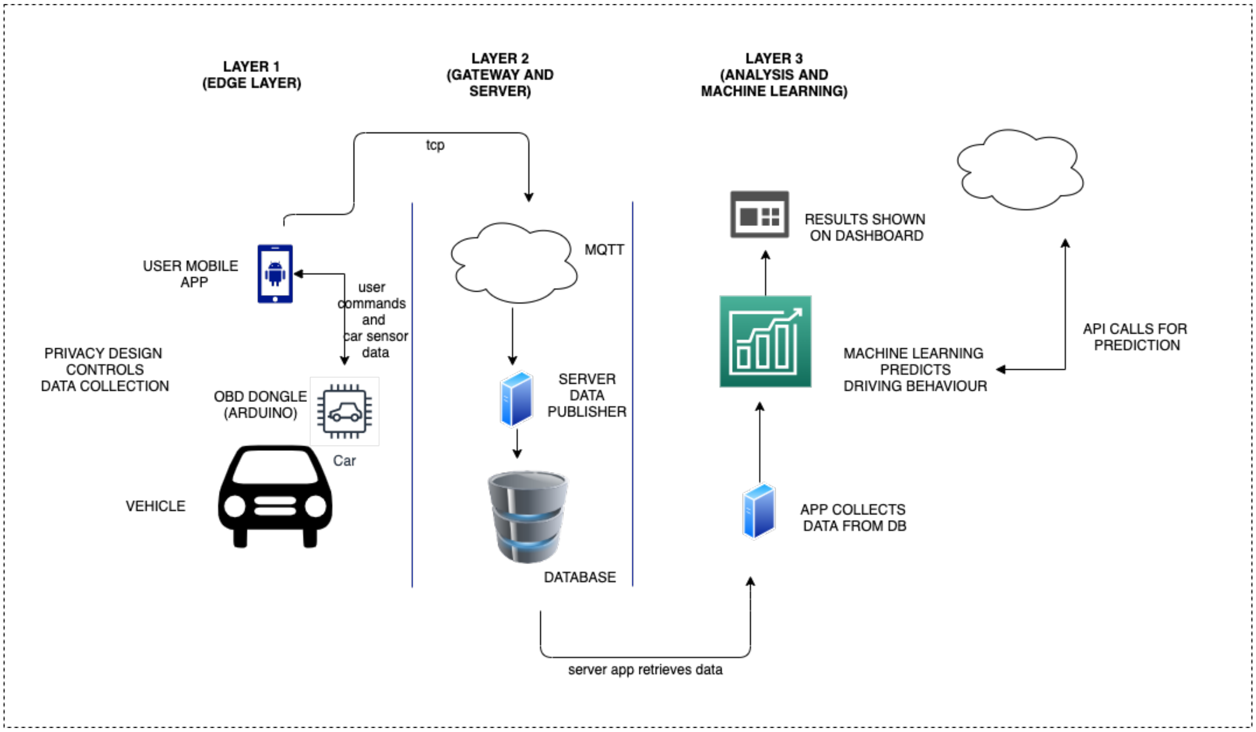


Figure 3.1: Overview of the proposed system

Studying the effects of privacy implementation on the system. This is the third stage of the research, where the efficiency of the system is tested and analyzed. At this stage, we checked the effects of the kind of sensor data that the system received to check its effects on the model. The second part of the security, which is allowing users to control the data, is also analyzed, and the effects were documented. It is at this stage that we tested how much control the user could have on the flow of data for the system to work.

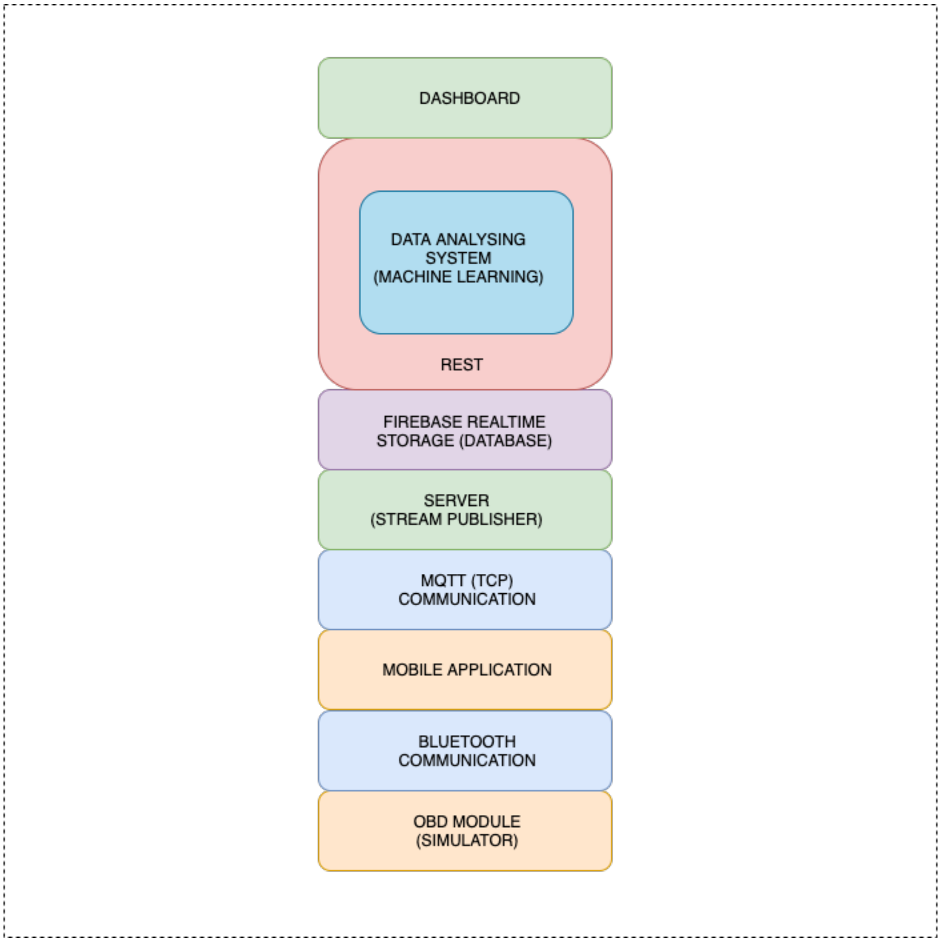

DASHBOARD
DATA ANALYSING SYSTEM (MACHINE LEARNING)
REST
FIREBASE REALTIME STORAGE (DATABASE)
SERVER (STREAM PUBLISHER)
MQTT (TCP) COMMUNICATION
MOBILE APPLICATION
BLUETOOTH COMMUNICATION
OBD MODULE (SIMULATOR)

Figure 1.2: High-level architectural design of the system

3.2 System Design

Mobile Application

The mobile application is in 2 versions. Version one(1) is without the privacy design, and version two (2) is with the privacy design alongside a stripped-down version of the functionality to basic requirements to serve as a proof of concept.

Version 1 has the following requirements, namely:

1. Retrieve vehicular data from the OBD simulator
2. Allow user to view the data being collected
3. Periodically send the data to a server

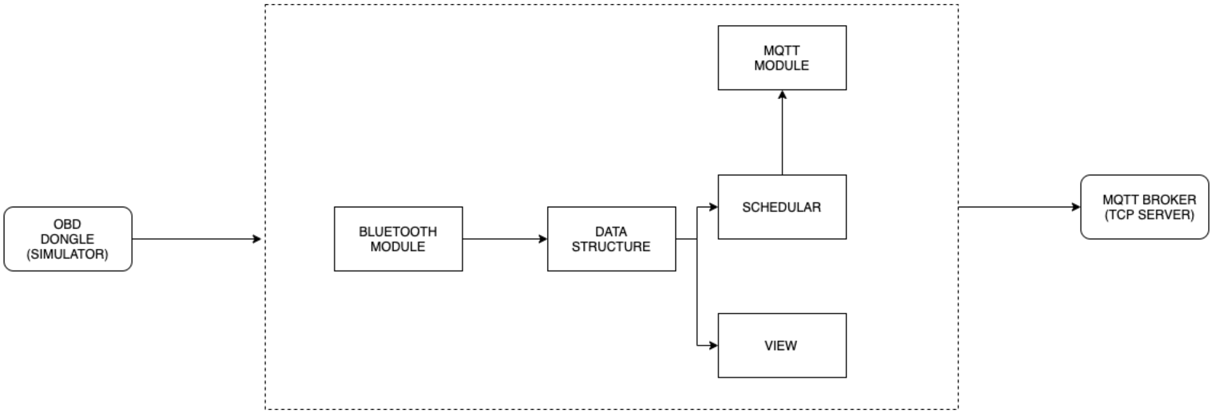


Figure 3.3: high-level architecture of mobile App (version 1)

Version 2 provides the same requirements as version 1 but also allows the privacy design that gives the user the opportunity to select and filter what data is collected and when it can be collected.

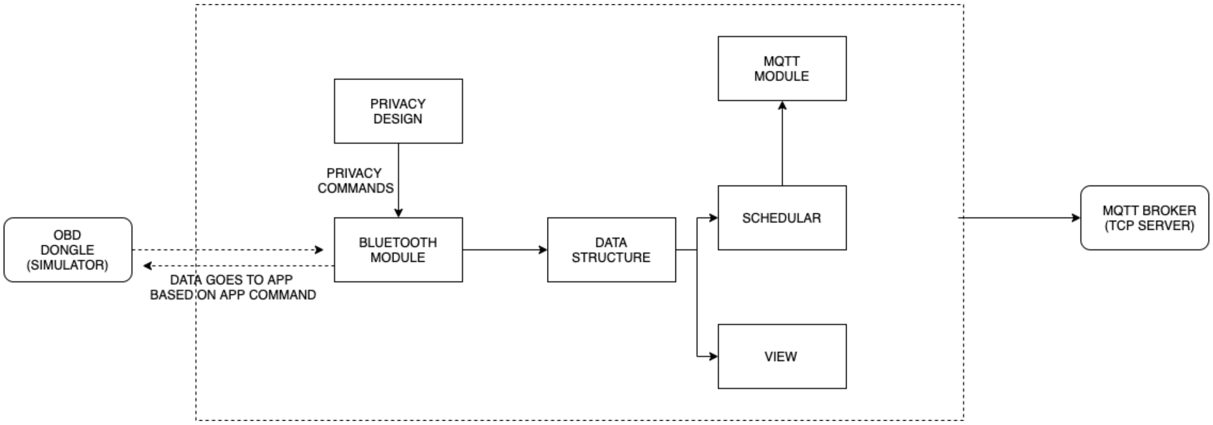


Figure 3.4: high-level of architecture of mobile App (version 2)

Driver Privacy Design

The data privacy design aspect is the main section of version 2 of the App. Its main focus is to allow the user to filter and choose what data is sent to the cloud to allow them to exercise their rights. Various methods were analyzed, and the best one was chosen. The application will have the functionality to allow the user to toggle radio buttons that will send signals to the Arduino to block or interrupt the specified data from flowing.

Vehicle Data Publisher & Stream Publisher

The Vehicle data publisher is the aspect of the system that passes data from all users to the cloud. It is designed to send the data to the MQTT server, which is a broker that allows clients to subscribe to the data to get them in real-time. The vehicle data publisher is part of the mobile application.

On the other hand, the stream publisher is a stand-alone server application that subscribes to the MQTT broker and collects the data upon arrival and takes it to the Firebase Realtime Database.

Driving Behaviour Analysis

The analysis of the driving style is the part of the system designed to analyze the data being collected to determine the driving style. Labelled data will be collected from an appropriate data source, and a machine learning model will be used to predict new data based on the deep learning it does from the dataset.

The driving behaviour will also be analyzed before and after the introduction of privacy by design to determine the effects the privacy aspect has on it.

Chapter 4: Experimental Setup and Implementation

4.1 Language, Tools and Technologies

The development of the system was separated into various tasks. The OBD module simulator was created in python, and the data was sent to an ATmega2560. From there, data sent to the phone via Bluetooth and to the cloud for analysis. A more detailed explanation of the technology used is presented here:

- OBD Module

  Languages- Arduino, Python

  IDE- Arduino, Visual Studio Code

  Hardware - ATmega2560, one breadboard, 1 Bluetooth module, three male to female wires and two male to male wires
- Mobile App(v 1 & 2)

  Languages – Arduino

  IDE – Android studio
- Server

  Languages – Python

  IDE – Visual Studio Code
- Database

  Languages – NoSQL

  Engine – Firebase Realtime Database Engine
- Analysis

  Languages- Python (Scikit learn library)

  IDE – Jupyter Lab and Spyder (Anaconda environment)
- Dashboard

  Languages – HTML, CSS, JS, PYTHON (bootstrap, jQuery, flask)

  IDE – Visual Studio Code

4.2 Implementation

OBD Simulator (Arduino Module)

An OBD emulator was purchased from the Mac App store, but it lacked the necessary documentation and tutorials needed for it to be used; hence a simple simulator was created in python to send generated data to the Arduino module. The Arduino then sends the data every 10 seconds via Bluetooth to a paired mobile phone based on the command it receives from the phone.

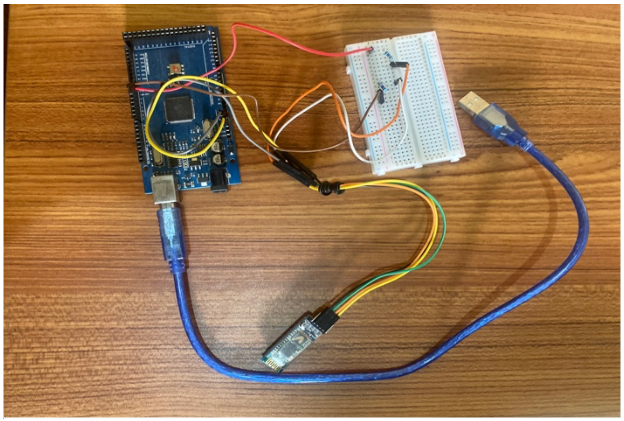


Figure 4.1: Arduino module setup

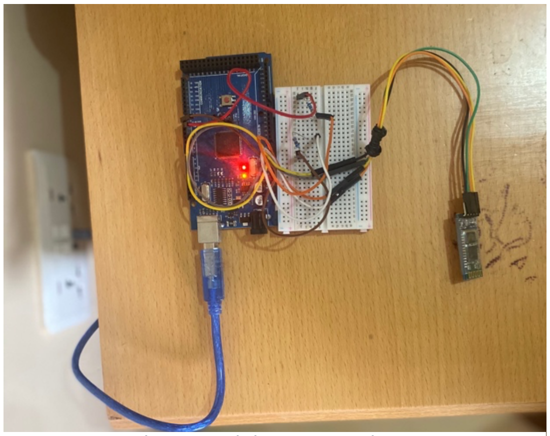
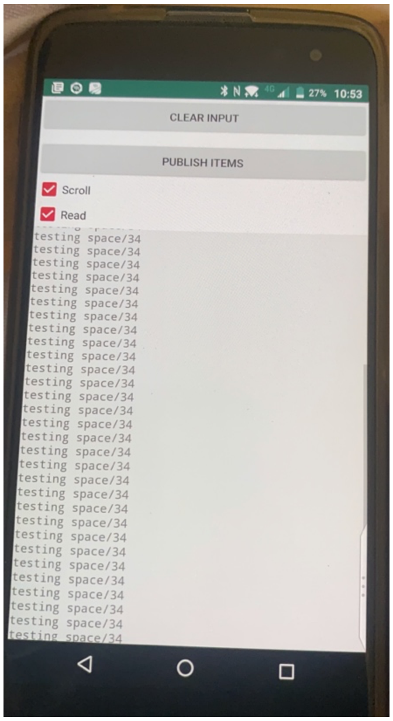


Figure 4.2: Arduino module connected to power

Mobile Application

The mobile app is implemented using Android SDK; the minimum SDK version to run the App is Android API-level 27. The mobile phone that runs the App must have Bluetooth and internet to execute the tasks. Version 1 of the app receives data from the OBD simulator and displays it. It also sends the data periodically to the MQTT broker via the internet. Version 2 of the application has a design that gives the user the ability to control the data that is collected and sent over the network.



```
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
testing space/34
```

Figure 4.3: version 1 of mobile App in the testing phase

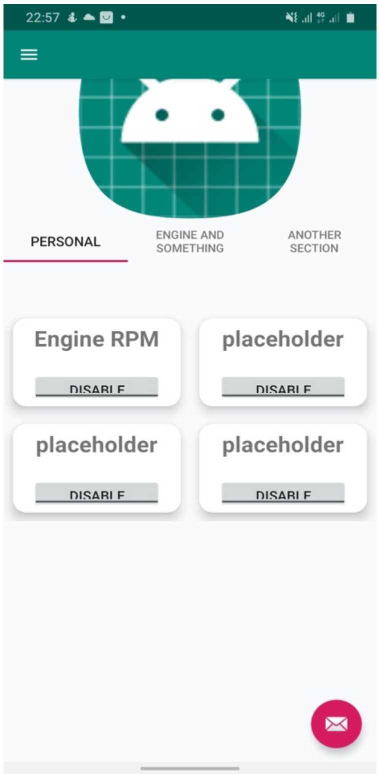


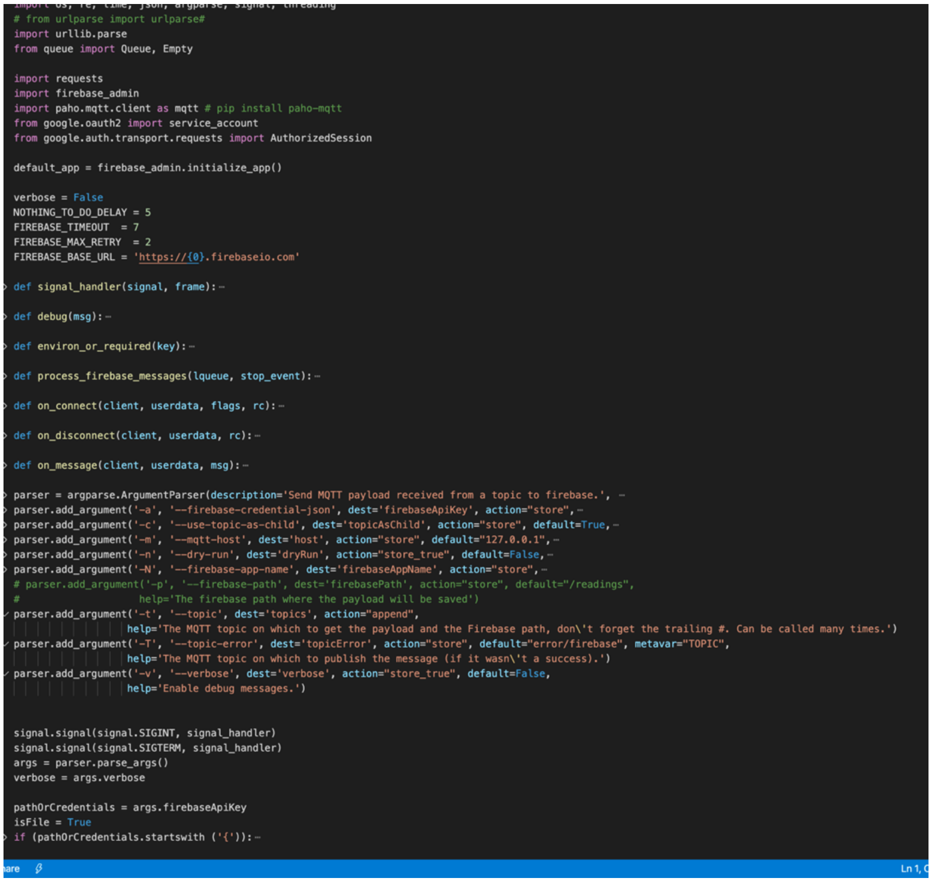
Figure 4.4: version 2 of android app

MQTT Listening Server

The Stream publisher (listening server), in this case, is the server that is implemented. Data from the phone is sent to an MQTT broker, and the server subscribes to the necessary topic and listens for uploaded data. As soon as data is sent to the broker, it takes the data and sends it to the firebase Realtime Database. It was created in python using the Pyrabase library and MQTT library. Part of the code was obtained from

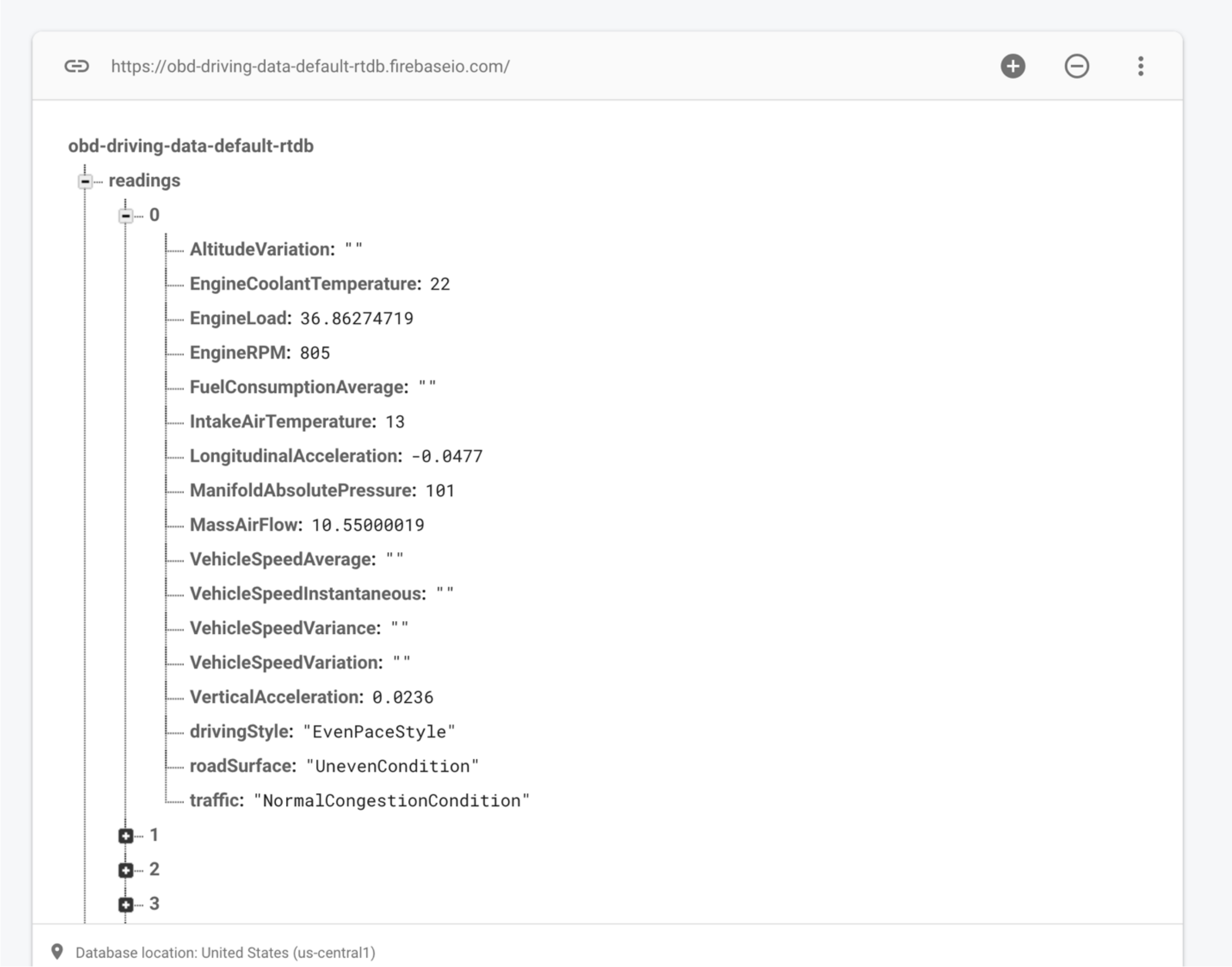
GitHub [10] under the GNU General Public License by the Free Software Foundation.

The database is the online Realtime Database from Firebase.



```python
import os, re, time, json, argparse, signal, threading
# from urlparse import urlparse#
import urllib.parse
from queue import Queue, Empty

import requests
import firebase_admin
import paho.mqtt.client as mqtt # pip install paho-mqtt
from google.oauth2 import service_account
from google.auth.transport.requests import AuthorizedSession

default_app = firebase_admin.initialize_app()

verbose = False
NOTHING_TO_DO_DELAY = 5
FIREBASE_TIMEOUT  = 7
FIREBASE_MAX_RETRY  = 2
FIREBASE_BASE_URL = 'https://{0}.firebaseio.com'

def signal_handler(signal, frame):⋯

def debug(msg):⋯

def environ_or_required(key):⋯

def process_firebase_messages(lqueue, stop_event):⋯

def on_connect(client, userdata, flags, rc):⋯

def on_disconnect(client, userdata, rc):⋯

def on_message(client, userdata, msg):⋯

parser = argparse.ArgumentParser(description='Send MQTT payload received from a topic to firebase.', ⋯
parser.add_argument('-a', '--firebase-credential-json', dest='firebaseApiKey', action="store",⋯
parser.add_argument('-c', '--use-topic-as-child', dest='topicAsChild', action="store", default=True,⋯
parser.add_argument('-m', '--mqtt-host', dest='host', action="store", default="127.0.0.1",⋯
parser.add_argument('-n', '--dry-run', dest='dryRun', action="store_true", default=False,⋯
parser.add_argument('-N', '--firebase-app-name', dest='firebaseAppName', action="store",⋯
# parser.add_argument('-p', '--firebase-path', dest='firebasePath', action="store", default="/readings",
#                     help='The firebase path where the payload will be saved')
parser.add_argument('-t', '--topic', dest='topics', action="append",
                    help='The MQTT topic on which to get the payload and the Firebase path, don\'t forget the trailing #. Can be called many times.')
parser.add_argument('-T', '--topic-error', dest='topicError', action="store", default="error/firebase", metavar="TOPIC",
                    help='The MQTT topic on which to publish the message (if it wasn\'t a success).')
parser.add_argument('-v', '--verbose', dest='verbose', action="store_true", default=False,
                    help='Enable debug messages.')


signal.signal(signal.SIGINT, signal_handler)
signal.signal(signal.SIGTERM, signal_handler)
args = parser.parse_args()
verbose = args.verbose

pathOrCredentials = args.firebaseApiKey
isFile = True
if (pathOrCredentials.startswith ('{')):⋯
```

Figure 4.5: snippet from the stream publisher server code



```
https://obd-driving-data-default-rtdb.firebaseio.com/

obd-driving-data-default-rtdb
  readings
    0
      AltitudeVariation: ""
      EngineCoolantTemperature: 22
      EngineLoad: 36.86274719
      EngineRPM: 805
      FuelConsumptionAverage: ""
      IntakeAirTemperature: 13
      LongitudinalAcceleration: -0.0477
      ManifoldAbsolutePressure: 101
      MassAirFlow: 10.55000019
      VehicleSpeedAverage: ""
      VehicleSpeedInstantaneous: ""
      VehicleSpeedVariance: ""
      VehicleSpeedVariation: ""
      VerticalAcceleration: 0.0236
      drivingStyle: "EvenPaceStyle"
      roadSurface: "UnevenCondition"
      traffic: "NormalCongestionCondition"
    1
    2
    3

Database location: United States (us-central1)
```

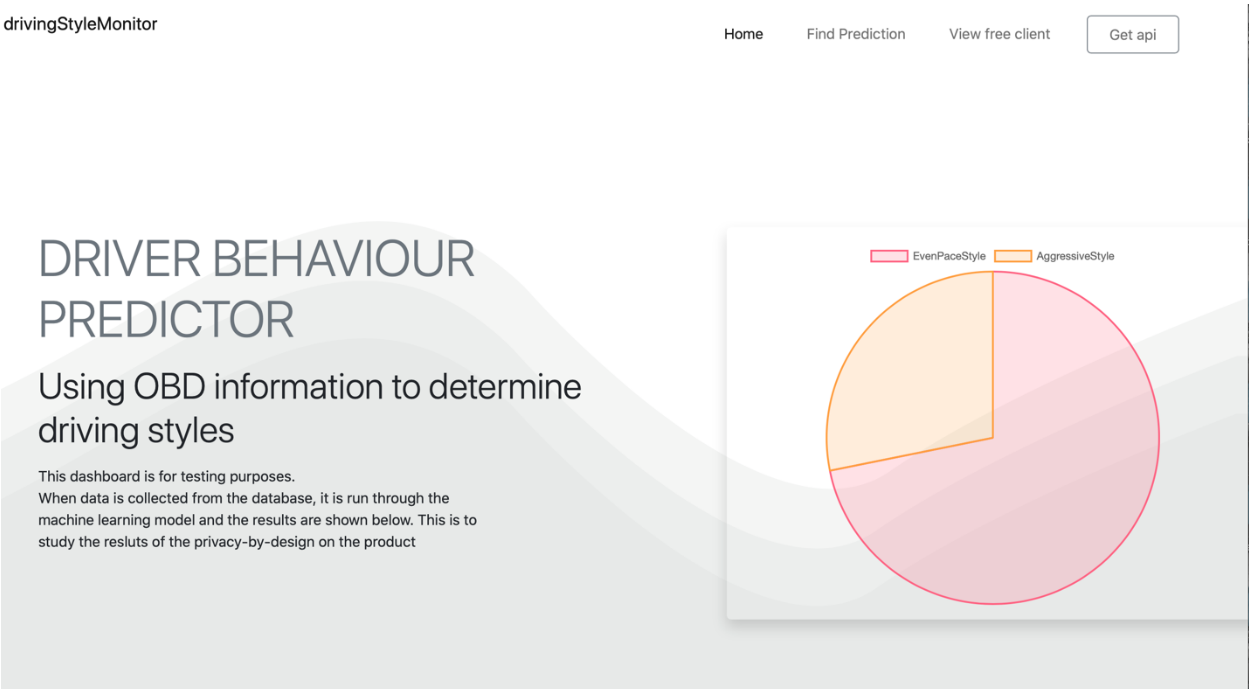
Figure 4.6: snippet from the firebase database

Driver Profiling Platform

The profiling platform is made up of the machine learning model. It predicts a user to be a safe driver or an aggressive driver and the probability of accuracy of the prediction. It uses data such as the Engine Load, Vertical Acceleration etc., to perform the prediction.

Implementation of Dashboard

The dashboard presents results for the Insurance companies as well as third-party companies that want to use the system. It shows the efficiency of the model, provides a test portal for people to test the system, shows data and driving predicted status from live cars, and API calls from the insurance companies for the results of the subscribed users in the system. The dashboard was implemented using python and the flask module. The machine learning module is serialized using the pickle library, so the model is built only once. Anytime there is an addition to the database, the data is retrieved from the firebase and goes through the prediction process and shows up on the dashboard. HTML, CSS, and bootstrap were used for the front end. ChartJS library was used in showing charts of the data on the dashboard.



DRIVER BEHAVIOUR PREDICTOR

Using OBD information to determine driving styles

This dashboard is for testing purposes.
When data is collected from the database, it is run through the machine learning model and the results are shown below. This is to study the resluts of the privacy-by-design on the product

Figure 2.7: dashboard in testing phase

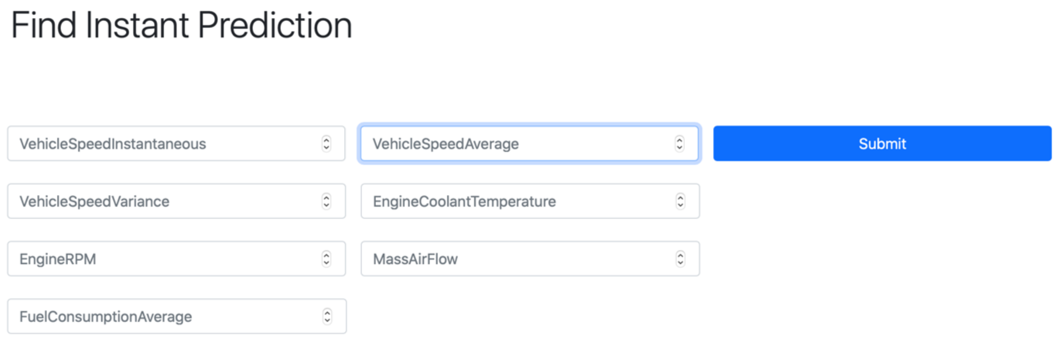
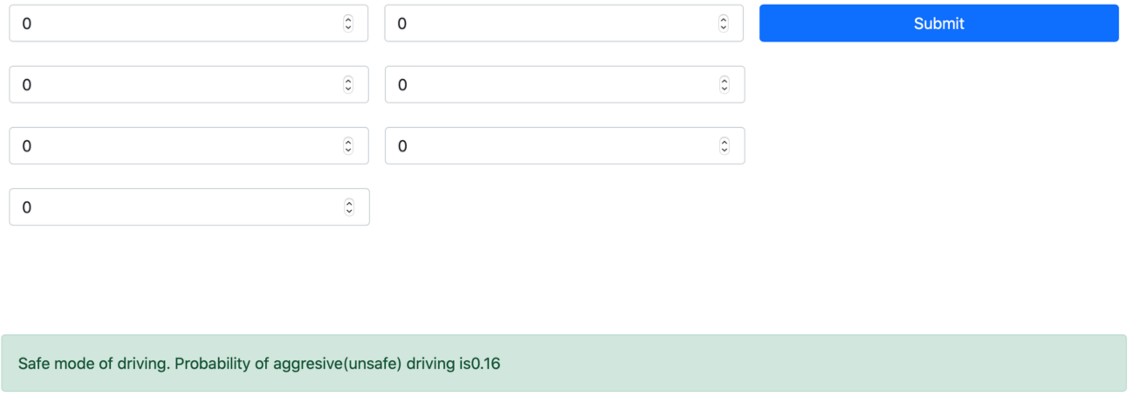

Find Instant Prediction

Figure 4.8: predictive test section



0 0

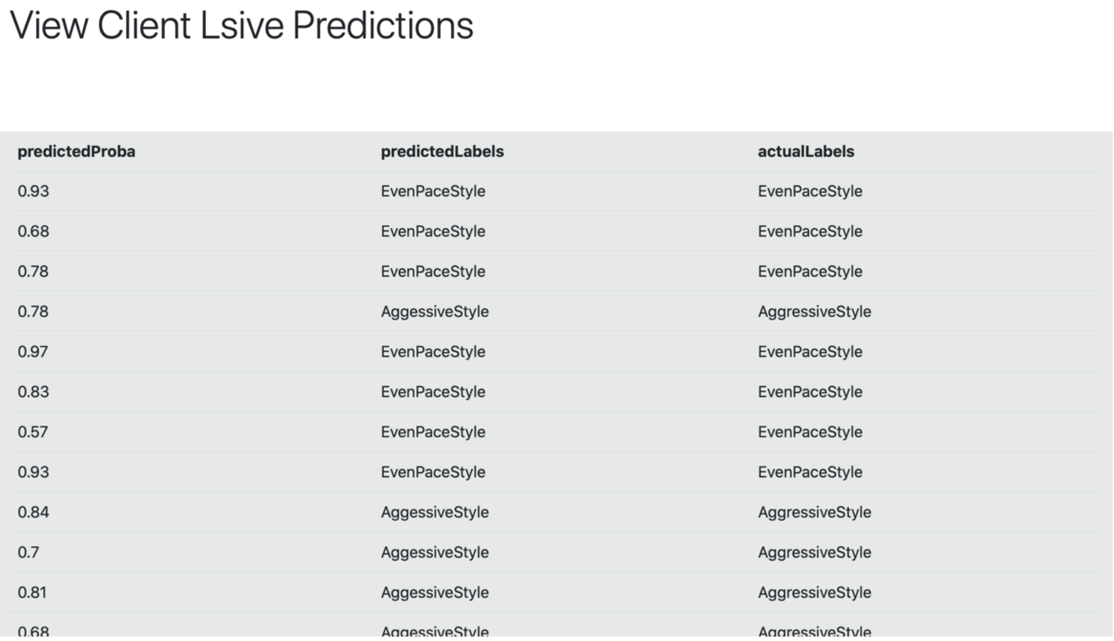
0 0

0 0

0

Safe mode of driving. Probability of aggresive(unsafe) driving is0.16

Figure 4.9: predictive test of the dashboard in testing phase



View Client Lsive Predictions

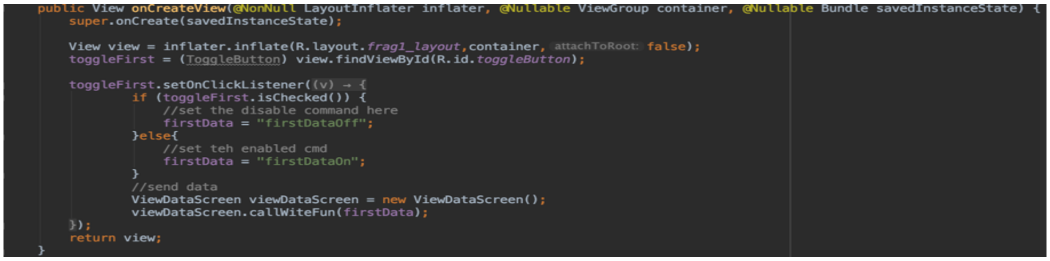
predictedProba	predictedLabels	actualLabels
0.93	EvenPaceStyle	EvenPaceStyle
0.68	EvenPaceStyle	EvenPaceStyle
0.78	EvenPaceStyle	EvenPaceStyle
0.78	AggessiveStyle	AggressiveStyle
0.97	EvenPaceStyle	EvenPaceStyle
0.83	EvenPaceStyle	EvenPaceStyle
0.57	EvenPaceStyle	EvenPaceStyle
0.93	EvenPaceStyle	EvenPaceStyle
0.84	AggessiveStyle	AggressiveStyle
0.7	AggessiveStyle	AggressiveStyle
0.81	AggessiveStyle	AggressiveStyle
0.68	AggessiveStyle	AggressiveStyle

Figure 4.10: free client predictions for driving style in testing phase.

Chapter 5: Results

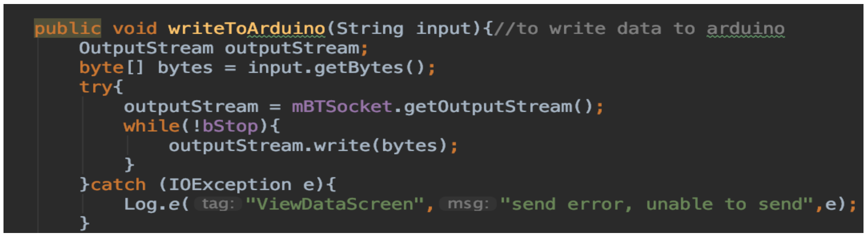
5.1 Privacy by Design

Figures 5.1 and 5.2 shows the implementation of privacy by design in the mobile app version 2. The App allows the user to turn off certain features he doesn't want to be collected as well as the ability to not send any personal data. Two options were considered. I.e. allow the data to come to the phone and, based on users preference, filter what goes to the cloud or the second option, to allow the user to control what data comes to the phone via the Arduino module by controlling it from the phone. The second option is the most secure, and that gives the user the most security and privacy control over the data. The user toggles a button either on or off, and the input is sent to the Arduino for the Arduino to prevent that data from coming in or allowing it. This reduces the risk of unwanted data being sniffed along the way. It is designed to occur even before the data is transmitted to the phone.



```
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    View view = inflater.inflate(R.layout.frag1_layout,container, attachToRoot: false);
    toggleFirst = (ToggleButton) view.findViewById(R.id.toggleButton);

    toggleFirst.setOnClickListener((v) → {
            if (toggleFirst.isChecked()) {
                //set the disable command here
                firstData = "firstDataOff";
            }else{
                //set teh enabled cmd
                firstData = "firstDataOn";
            }
            //send data
            ViewDataScreen viewDataScreen = new ViewDataScreen();
            viewDataScreen.callWiteFun(firstData);
    });
    return view;
}
```

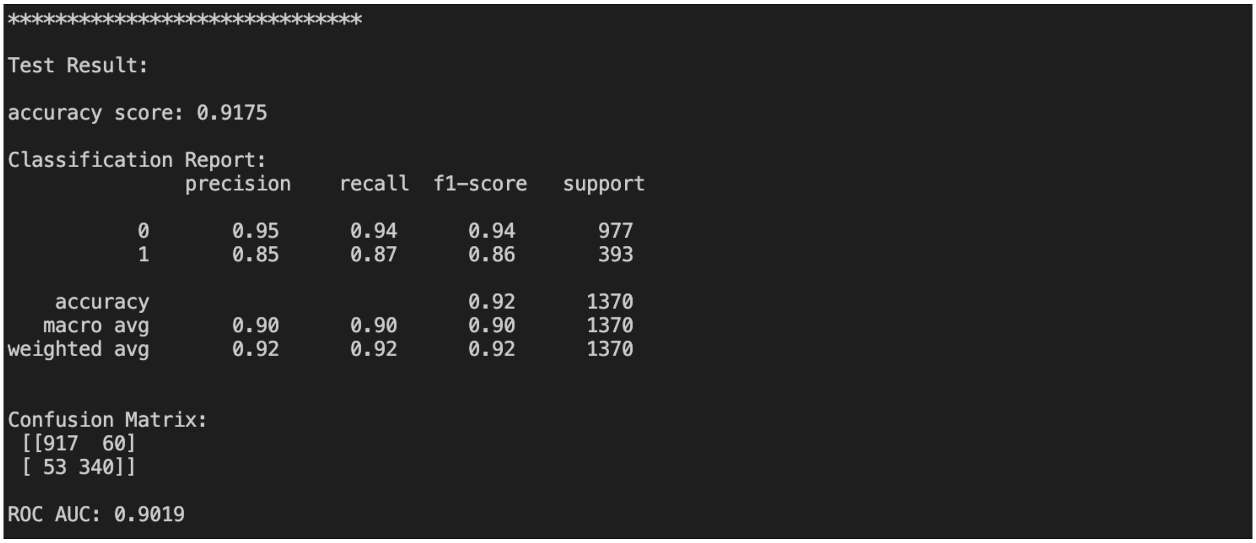
Figure 5.1: snippet of code for toggle button to turn data 'off' or 'on'



```
public void writeToArduino(String input){//to write data to arduino
    OutputStream outputStream;
    byte[] bytes = input.getBytes();
    try{
        outputStream = mBTSocket.getOutputStream();
        while(!bStop){
            outputStream.write(bytes);
        }
    }catch (IOException e){
        Log.e( tag: "ViewDataScreen", msg: "send error, unable to send",e);
    }
```

Figure 5.2: snippet of code to send data to Arduino

5.2 Performance and Accuracy



```
******************************

Test Result:

accuracy score: 0.9175

Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.94      0.94       977
           1       0.85      0.87      0.86       393

    accuracy                           0.92      1370
   macro avg       0.90      0.90      0.90      1370
weighted avg       0.92      0.92      0.92      1370


Confusion Matrix:
 [[917  60]
 [ 53 340]]

ROC AUC: 0.9019
```

Figure 5.3: Machine learning report

To evaluate the accuracy of the system's predictive part, data[11] collected from the Kaggle datasets about low-level parameters collected by the cars OBD II system through a dongle since it would be expensive to drive around with cars to collect data and label it. The dataset had 17 features and three labels; we used the driving behaviour label and, from feature selection in the machine learning process, chose seven features to use in the prediction without changing the parameters of the model except for the random state of the random classifier which was set to 50. The scikit-learn random forest classifier was used in generating the model. It generated an accuracy of 0.92 in the test data and an ROC of 0.9. From the confusion matrix, it classified 917 as safe drivers who were accurate but 60 as unsafe, which was wrong. It also classified 340 as risky driving patterns correctly and 53 incorrectly. This is a fairly good model, and because the data did not have as much data for risky driving patterns as safe patterns, it was more accurate with safe drivers with an f1-score of 0.94 for safe drivers, which is better than the f1-score of 0.86 for risky drivers.

This result was obtained after tests were done on previously generated models and found a case of overfitting, which was caused by a large percentage of the data being categorized as safe driving. Another script was written to clean the data and create a sub-data with all the aggressive behaviour and 40 per cent of all the safe driving data in order to reduce the skewness of the data to reduce the effect on the machine learning model. This improved the previous model greatly for us to arrive at the results that were attained.

5.3 Result Comparison

The results from the comparison were interesting. The study was done to examine how removing certain features of the data would affect the model to emulate the user turning off data on the phone and exercising his rights. All features for one test data was computed, and the probability and prediction were recorded. Subsequently, each feature was removed and the data was passed through the model, and the results were calculated as well. This was repeated for different scenarios until all data was removed and the empty set was passed through the model.

The results showed that each time we removed a feature, the probability of safe driving increased by a fraction and as we removed more data, the probability of safe driving kept on getting better. When we turned off all the data, the probability was almost one (1), which means the best form of safe driving. This was interpreted as the more data was removed, it meant the car was either moving at a very slow pace or was stationary, and that model predicted correctly that it was not risky driving. This meant that at the current stage, we could not rely on the model for the implementation of the privacy-by-design until something was done. After brainstorming and conducting further research, different ideas were considered, and four (4) seemed appropriate; therefore, another experiment will have to be conducted to implement them and study their performance.

5.4 Solutions from Results

As stated earlier, the analysis of the results gave rise to different solution methods to cater for the performance of the system, but they must be tested.

The first idea, which has the simplest implementation, was to generate more test data by duplicating the current test data and strategically omitting features from it until most user test scenarios have been covered and trained another model. The hypothesis is that, since it would learn from data that has all features and the same data with some features, it will give a more accurate prediction of the driving style when features are removed.

The second idea which must also be implemented and tested is to replace removed features with previously recorded features for prediction until all the features are sent again. The hypothesis is that, it would give a better prediction of driving since the removed features will be replaced by the last best-recorded features of that same driver.

The third experiment that can be conducted is to allow different drivers to dive in different ways, some safe, some aggressive and the data collected and analyzed to be used to form some form of a benchmark such that when data is removed from the set entering the model, the remaining data is run against the benchmark and deviation is calculated to determine what category best fits that driving behaviour. The disadvantage is that this would be expensive to conduct.

Finally, one of the other ideas gotten from discussions was that when data is interrupted by the user, we do not use the uninterrupted data coming through but the previous set with all the data until all access is granted again. This is to only allow the model to work on fully provided data.

Chapter 6: Summary

6.1 Conclusion

With the emergence of IoT in Insurance, developing systems to retrieve vehicular data and make informed decisions such as lower premiums and better driving styles are a few advantages. In Usage-Based Insurance's current state, it is important that privacy by design is introduced not only in Insurance IoT controlled Systems but in IoT in general. As the outcome features of this project, we came to the conclusion that privacy by design methods can be included in IoT controlled Insurance, and the current implementation does influence the outcomes of the machine learning model.

The study proved that privacy by design could be implemented, and we did that at the edge level of the system, ie. The Arduino and the phone privacy implementation. We also studied the results on the predictive capabilities of the machine learning model based on the effects of privacy by design. The current system does not perform efficiently when data is interrupted; hence further study must be done to test the four (4) hypothesis to improve the performance of the system.

6.2 Challenges

Quite a number of challenges were faced during this project. First, the OBD II ELM Emulator purchased to be used with the Arduino did not come with documentation, and no information was on their website providing steps as to how it can be used. As a solution, a log file was created with test data and a python script to pass on data as an emulator.

Getting OBD labelled data for driving behaviour was another challenge. There are not a lot of OBD collected data that has been labelled for driving behaviours.

The schedular library for android that allows background tasks only works with a minimum schedule time of 15 minutes. Since the App was needed to send data more quickly, a thread was created to send data after every 2 minutes as a solution causing the mobile application to transmit data only when the App is opened.

6.3 Future Works

Currently, the OBD Arduino dongle does not interface with an actual OBD or simulator, and test data is being sent via a log file. In future, a robust OBD Bluetooth dongle will be implemented to interface with an actual OBD II system. The mobile Apps only allow data to be transmitted when the App is opened, and it must be updated to allow it to transmit and receive data even when it is working in the background. Version two of the App does not allow the user to see graphs and performance indexes of how well they perform during trips, and this can be included to allow them to assess themselves.

For the predictive module, a better solution can be generated to reduce errors in predictions when the user controls the flow of data. The current solution is highly dependent on data and does not work well with data interruptions. Four different ideas or experiments were generated from the results to make this part of the project better. They must be implemented and tested in order to prove that privacy by design does not affect the effectiveness of the system. The ideas were;

1. The retraining of the model with duplicated partial data.
2. Filling empty feature spaces with previous data about the same driver.

3. Creating and using a benchmark with deviations to fit driving styles.
4. Using only fully accessed data for predictions.

They have been explained better in the chapter above.

Concerning the most important part of this project, the privacy by design, other design methods can be tested to make it robust enough such that the accuracy and performance of the system are not compromised.

References

[1] O. Andrisano, R. Verdone, and M. Nakagawa, "Intelligent transportation systems: the role of third-generation mobile radio networks," *IEEE Communications Magazine*, vol. 38, no. 9, pp. 144–151, Sep. 2000, doi: 10.1109/35.868154.

[2] J. Quintero and Z. Benenson, “Understanding Usability and User Acceptance of Usage-Based Insurance from Users’ View,” in *Proceedings of the 2019 2nd International Conference on Machine Learning and Machine Intelligence - MLMI 2019*, Jakarta, Indonesia, 2019, pp. 52–57, doi: 10.1145/3366750.3366759.

[3] S. Derikx, M. de Reuver, and M. Kroesen, “Can privacy concerns for insurance of connected cars be compensated?,” *Electronic Markets*, vol. 26, Dec. 2015, doi: 10.1007/s12525-015-0211-0.

[4] Y. H. Hwang, “IoT Security & Privacy: Threats and Challenges,” in *Proceedings of the 1st ACM Workshop on IoT Privacy, Trust, and Security - IoTPTS ’15*, Singapore, Republic of Singapore, 2015, pp. 1–1, doi: 10.1145/2732209.2732216.

[5] “Chapter 3 – Rights of the data subject,” *General Data Protection Regulation (GDPR)*. https://gdpr-info.eu/chapter-3/ (accessed Oct. 12, 2020).

[6] C. Herzog, E. G. Counsel, C. of A. E. W. Group, and Arm, “Engineering Ethics into AI,” *Arm Blueprint*, Nov. 06, 2019. https://www.arm.com/blogs/blueprint/arm-ai-trust-manifesto (accessed Sep. 30, 2020).

[7] W.-H. Chen, Y.-C. Lin, and W.-H. Chen, “Comparisons of Machine Learning Algorithms for Driving Behavior Recognition Using In-Vehicle CAN Bus Data,” in *2019 Joint 8th International Conference on Informatics, Electronics Vision (ICIEV) and 2019 3rd*

International Conference on Imaging, Vision Pattern Recognition (icIVPR), May 2019, pp. 268–273, doi: 10.1109/ICIEV.2019.8858531.

[8] P. Handel, J. Ohlsson, M. Ohlsson, I. Skog, and E. Nygren, “Smartphone-Based Measurement Systems for Road Vehicle Traffic Monitoring and Usage-Based Insurance,” *IEEE Systems Journal*, vol. 8, no. 4, pp. 1238–1248, Dec. 2014, doi: 10.1109/JSYST.2013.2292721.

[9] V. Gandhi and J. Singh, “IoT: Architecture, Technology, Applications, and Quality of Services,” 2019, pp. 79–92.

[10] S. Lucas, *seblucas/mqtt2firebase*. 2021.

[11] “Traffic, Driving Style and Road Surface Condition.” https://kaggle.com/gloseto/traffic-driving-style-road-surface-condition (accessed Apr. 01, 2021).