Project Report On

Driver Behaviour and Usage Based Insurance (UBI)

Submitted by

Harsh Shinde 17u008

Shardul Nazirkar 17u024

Under the Guidance of

Prof. P. S. Wawage

In Partial fulfilment of

Bachelor of Technology

[B.Tech Information Technology]

[2020-2021]

At

Department of Information Technology

Vishwakarma Institute of Information Technology, Pune 411048

Affiliated to

Savitribai Phule Pune University, Pune

Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Information Technology, Pune 411048

CERTIFICATE

This is to certify that the work entitled "**DRIVER BEHAVIOUR AND USAGE-BASED INSURNACE (UBI)**" is a bonafide work carried out by Mr. Shardul Nazirkar, Mr. Harsh Shinde in partial fulfilment of the award of Bachelor of Technology in Information Technology, Savitribai Phule Pune University, Pune, during the year 2021. The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Technology Degree.

Prof. Pawan S Wawage
Project Guide

Prof. P.Futane   Prof.Vivek.Deshpande
Head, IT Department Director, VIIT, Pune

Date:

Examiners: 1 . . . . 2. . . . .

Place: Pune

Acknowledgement

We take this opportunity to thank Head of the Department Prof. P.Futane and our project guide Prof. Pawan.S.Wawage for their valuable guidance and providing all the necessary facilities, which were indispensable in the completion of this project report. We are also thankful to all the staff members of the Department of Information Technology, VIIT, Pune for their valuable time, support, comments, suggestions and persuasion. We would also like to thank Mr.Pawan S Wawage, Mr.Pankaj Shinde (TechTrail Technologies Pvt.Ltd) and the institute for providing the required infrastructure, internet and library facility.

Harsh Shinde 17u008

Shardul Nazirkar 17u024

ABSTARCT

In a country like India the scope of Usage-based insurance is a lot but it’s almost non-existent. As a further step, VIIT along with Tech-trail technologies have taken an initiative to help build a system that will help us overcome the following problem.

The purpose of this project is to develop a machine learning model that will help us calculate the safety-scores of drivers/customers based on where and how the vehicle has been driven. We plan to achieve this using the data collected from sensors that are present on our smartphones. The project follows agile methodology. The tools and technologies used are: Python for building the machine learning model, Firebase for real-time Database, Google cloud platform for model deployment, Visual Studio Code for building and implementing machine learning algorithm, and Postman for API Services along with flask.

Contents

List of Figures

List of Tables

Abbreviations

- DB : Driver behaviour
- UBI : Usage-based insurance
- PHYD: Pay-how-you –drive

Chapter 1

Introduction

1.1 Motivation

In today's day and age, having the right kind of knowledge is so important, but more so data. It's not the quantity of information that counts, but the kind of information you are able to acquire and the reason why you want to get it. That's why data is so important. The world is built upon data, so it makes sense to learn how to use it so that you can benefit the world in many ways. Hence, the aim of our project is to collect data relating to driving a vehicle and make a classification system that gives us the driver's behaviour and profile during driving. This data and classification system in turn can be applied to real world problems and one such being Usage Based Insurance (UBI).

1.2 Need of Driver behaviour profiling and usage-based insurance

Usage-based insurance, also referred to as pay-as-you-drive, is a type of auto insurance that, depending on the specific insurer's program, can measure how far a vehicle is driven, where it's driven, and/or how it's driven. We aim to achieve this using the data collected from sensors that are present on our smartphones. Usage Based Insurance has been widely proven to be effective and enticing for both the insurers and insured in many different countries. However, this system is very rarely used and almost non-existent in India, but it is on the track in future. Hence, the application of our project has great potential in India.

Work seamlessly with android devices – The system is integrated with an android application and customer's safety score and trip details are stored within the app so as to calculate to insurance premiums for the user.

1. Cheaper insurance – Usage-based insurance is usually denoted as Pay-How-You-Drive (PHYD) model, is useful for providing cheaper insurance to customers who have good safety scores instead only taking into consideration other statistical factors like(age, gender, marital status etc.).

2. Driver/Driving Monitoring – This system will help us to identify driver behaviour of driver/customer based on factors like aggressive braking, sudden turns, speeding and cornering that will help us in avoiding vehicular accidents due to road rage.

The main purpose of UBI will be to bring a more pragmatic and commensurate system to discover the risk where customer who have exhibited aggressive driving behaviour need to pay a higher premium.

1.3 Brief Introduction to DB and UBI System

In the case of UBI, the risk factors and variables being evaluated overlay the traditional rating methods—typically, age, gender, marital status, credit and driving records, and insurance scores—which have proven over time to have predictive capabilities. UBI extends the range of relevant data and makes it possible to evaluate risk based on each individual's real-world driving behaviour, inclinations toward safe or dangerous driving practices, trip characteristics, the current condition of the vehicle, and the locale of vehicle operation, weather factors, and more. Combined with traditional rating variables, UBI proves a better assessment of risk, insurers could offer the best drivers the maximum discount and still maintain a reasonable level of profitability. Deploying the appropriate technologies for UBI and analytical tools to acquire and evaluate driving data generates expenses—including cost of preparation, manpower, technology and equipment, logistics, and support.

1.4 Reason behind Driver Behaviour System

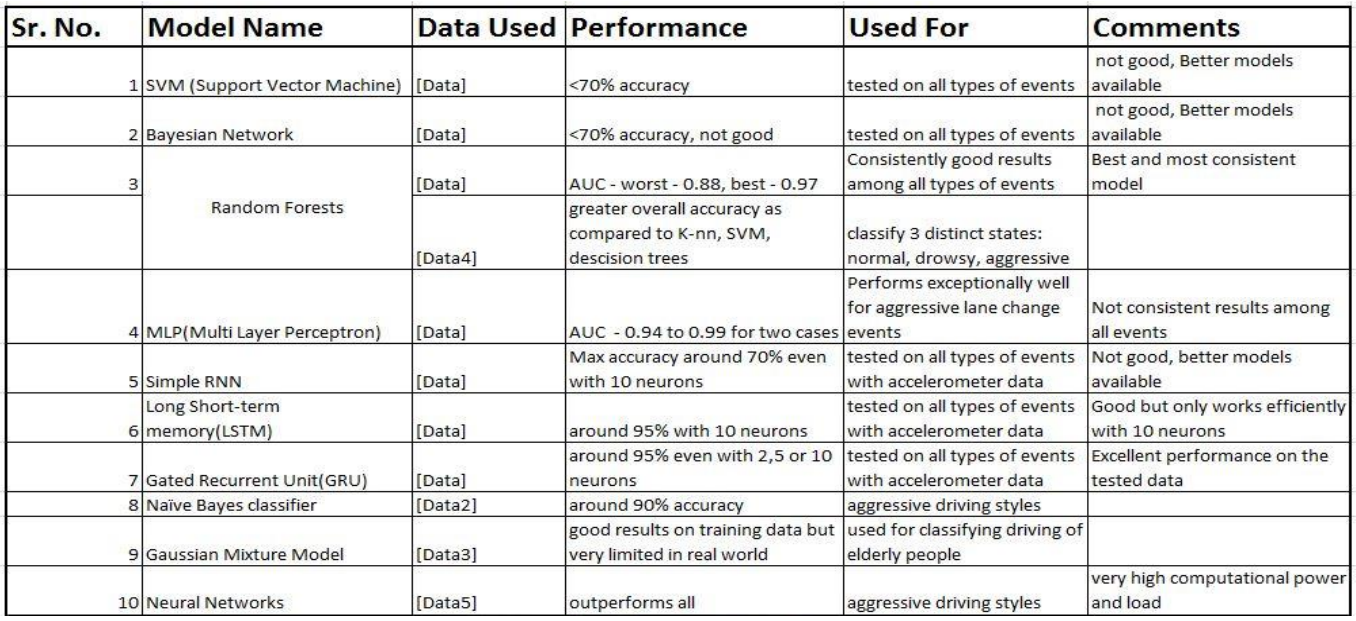
1. Attracting low-risk drivers
2. Enhancing customer loyalty
3. Reducing claims costs
4. Increasing the number of potential touch-points per year

Chapter 2

Literature Survey

2.1 Literature Review

Table No 2.1: Comparison table of pros and cons of Machine learning models



Sr. No.	Model Name	Data Used	Performance	Used For	Comments
1	SVM (Support Vector Machine)	[Data]	<70% accuracy	tested on all types of events	not good, Better models available
2	Bayesian Network	[Data]	<70% accuracy, not good	tested on all types of events	not good, Better models available
3	Random Forests	[Data]	AUC - worst - 0.88, best - 0.97	Consistently good results among all types of events	Best and most consistent model
		[Data4]	greater overall accuracy as compared to K-nn, SVM, descision trees	classify 3 distinct states: normal, drowsy, aggressive	
4	MLP(Multi Layer Perceptron)	[Data]	AUC - 0.94 to 0.99 for two cases	Performs exceptionally well for aggressive lane change events	Not consistent results among all events
5	Simple RNN	[Data]	Max accuracy around 70% even with 10 neurons	tested on all types of events with accelerometer data	Not good, better models available
6	Long Short-term memory(LSTM)	[Data]	around 95% with 10 neurons	tested on all types of events with accelerometer data	Good but only works efficiently with 10 neurons
7	Gated Recurrent Unit(GRU)	[Data]	around 95% even with 2,5 or 10 neurons	tested on all types of events with accelerometer data	Excellent performance on the tested data
8	Naïve Bayes classifier	[Data2]	around 90% accuracy	aggressive driving styles	
9	Gaussian Mixture Model	[Data3]	good results on training data but very limited in real world	used for classifying driving of elderly people	
10	Neural Networks	[Data5]	outperforms all	aggressive driving styles	very high computational power and load

2.2 Review of existing System

While brainstorming during system requirements and functionalities we came across various research papers related to DB and UBI. Upon survey we find out that random forest algorithm was found to be the best among the algorithms that are mentioned in the table above. None of the systems had cloud integration in their system so our goal was to integrate our system with any of the cloud platforms. After comparing various platforms we found out that google cloud platform was found to efficient with our System. Google app engine was the most efficient way to deploy the machine learning model on the cloud platform. We also performed visual data analysis on existing vehicle data sets to observer the trends in data. We have applied Time-Series graph on the available raw-datasets and tried to make sense out of the graphs visualised. We have used comparison between accelerometer vs gyroscope data collected from sensors of the smartphone.

Fig. 2.1 Accelerometer Main Data

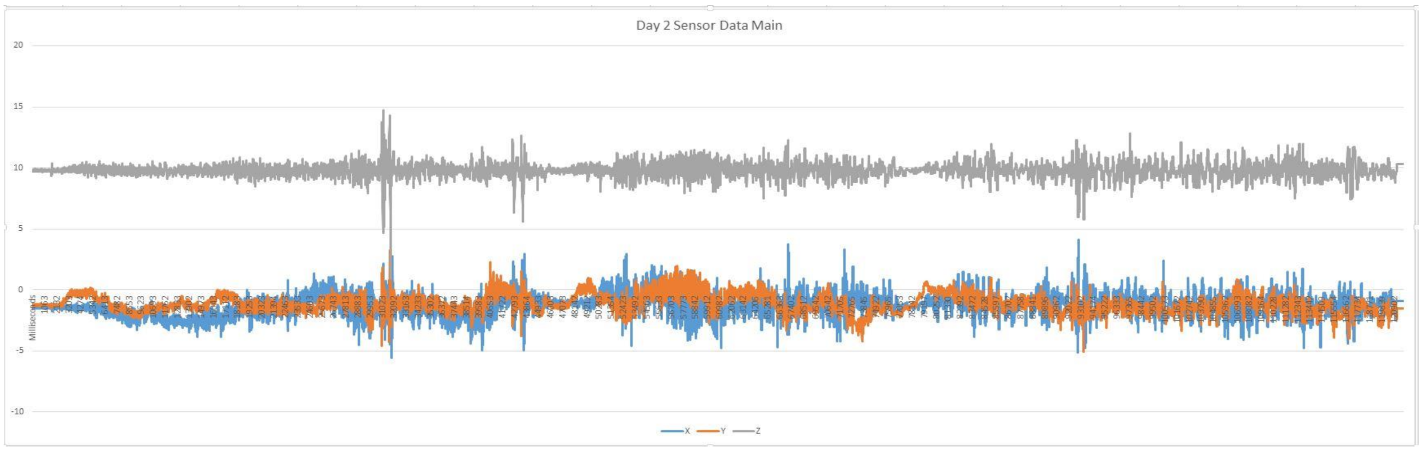
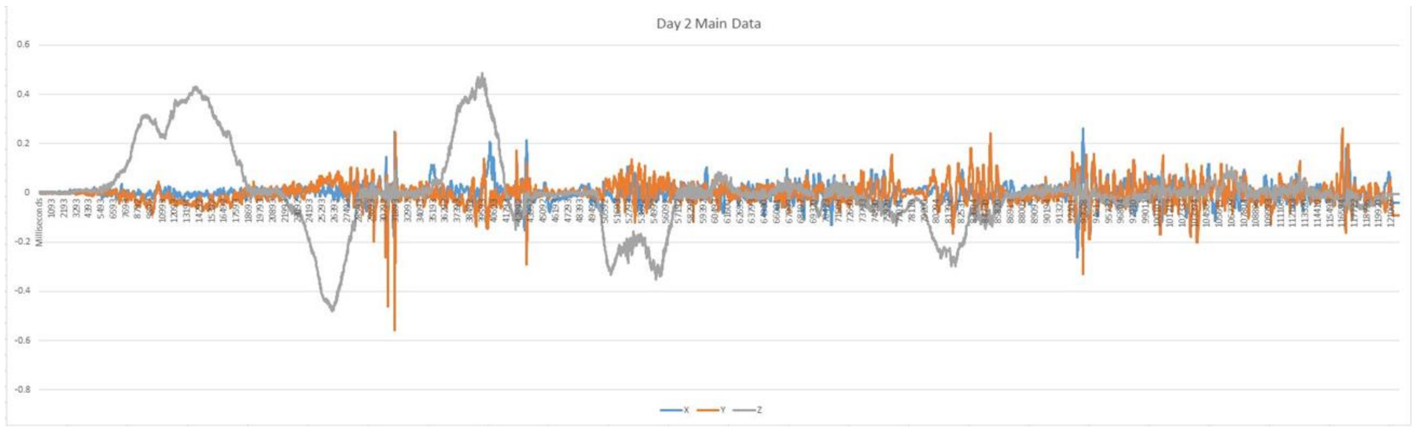


Fig. 2.2 Gyroscope Main Data



2.2.1 System Functions

User/Customer Functions

- Login / Sign-up to the system
- Start new trip
- View trip history
- View safety-score

Chapter 3

Project Statement

3.1 Purpose behind the Project

In a country like India Driver behaviour and usage-based insurance is almost non-existent. The purpose of this project is to develop and implement an android application using a machine learning algorithm, which will not only help the users to provide insurance benefits but also to identify driver behaviour as aggressive or normal there by reducing the risk of road accidents .

3.2 Decision of Scope

This project aims to develop a system which will include several functions like Sensors used for Smartphone Sensors (Accelerometer, Gyroscope, and Magnetometer) to collect data, Driver Behaviour Profiling into Safe or Aggressive, Giving a safety score to the driver for that trip, Using above data for Usage Based Insurance maybe monthly or quarterly and Platform – Smartphone for collecting and cleaning data. Depending on Algorithm load, deploying the model on smartphone using android and cloud platform in order to handle large amounts of data.

3.3 Methodology for solving this proposed theme

3.3.1 Proposed system Architecture

In our system, we are following this architecture. From mobile app the function is pinged into the system. The mobile app will be responsible for collecting the data. The safety score and driver behaviour will be displayed in the mobile app. The safety score is calculated using api's on cloud server and the whole model is deployed using the flask application.

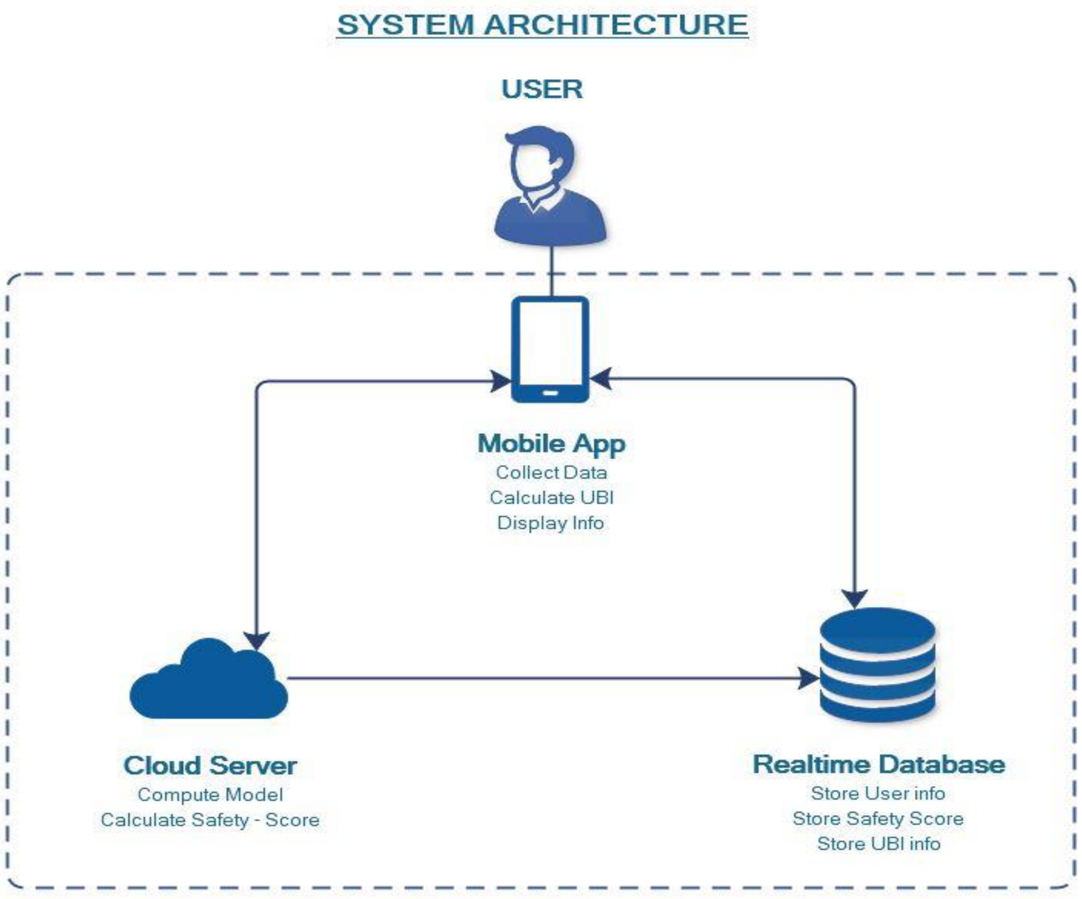


Fig. 3.1: Proposed System Architecture

3.3.2 In-depth Working of system

1. **User**: Users need to register in the app in order to avail the services provided by the system. Once registered, users have their own unique credentials to login to further access facilities. User will be able to view UBI details and will receive all the alerts and updates.

2. **App**: The app will collect the sensor data and will send the sensor data to the cloud.

3. **Cloud server**: The machine learning model on the cloud will calculate the safety score and send it to the app.

4. **Database**: The database will be responsible for storing real-time data of the user and other data related to UBI.

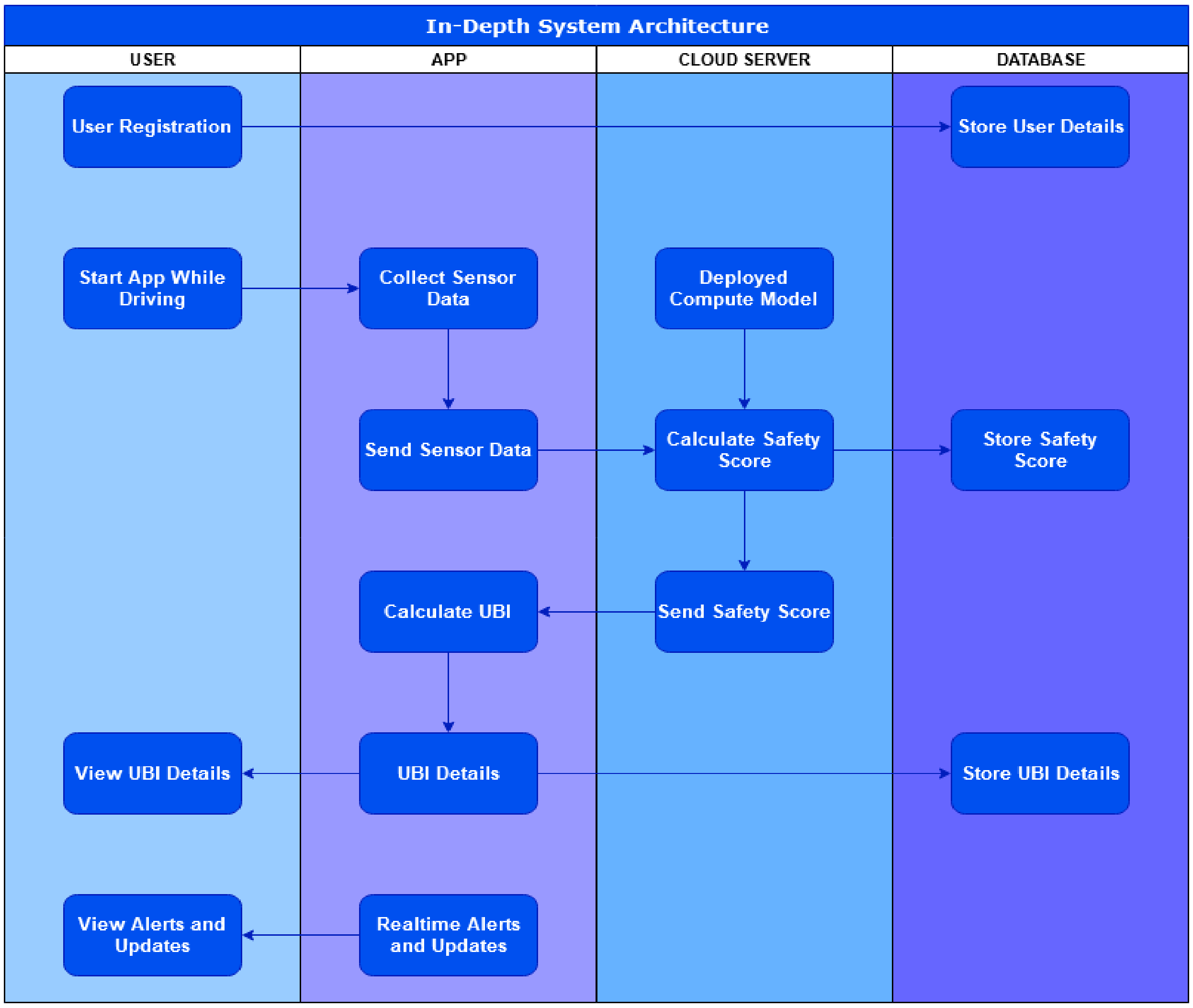


Fig. 3.2 In-Depth System Architecture

3.3.3 Proposed System for UBI

1. Every instance starts safety score with 10 out of 10.

2. For every 5min interval of data, we get a list of 0's (Normal) and 1's (Aggressive).

3. We calculate the percentage of 1's in the list.

4. Depending on the percentage of 1's we subtract from the safety score.

Percentage of 1's (Aggressive Instances)	Deduction from 10
>20%	-1
>25%	-2
>30%	-3
>35%	-4
>40%	-5
>50%	-6

Table 3.1 Proposed Deduction for Safety Score

1. Data is collected in intervals of 5 mins from start of the trip to end.

2. Score calculated for first interval using previous mentioned scoring system. (For eg. Let score be 6).

3. Then score is calculated for the next interval and average is calculated. (For eg. Score for 2nd interval is 7, therefore the overall score becomes 6+7/2 = 6.5).

4. Then average of previous overall and new interval score is calculated. (For eg. Score for 3rd interval is 8, therefore the overall score becomes 6.5+8/2 = 7.25).

5. The score for every trip can be calculated using the way in previous slide.

6. This score for all the trips in the time period of 1 month can be averaged out and hence we will get the average safety score for that month.

7. Depending on whether the insurance premium is to be paid month or yearly we can further average the scores of every month to get the score for a year.

8. Discount in the premium for the customer can be given using these scores using the below proposed system:

Score	Discount
6	5%
7	10%
8	12.5%
9	15%
10	20%

Table 3.2 Premium Discount System

Chapter 4

System Requirement and Specification

4.1 Software requirements specifications:

4.1.1 Introduction

This document is recommended to be viewed by the development team and other stakeholders of the system.

The developer and the tester are intended to read those sections of the document which Include operating environment, software interfaces, and design and implementation constraints. The users are exclusively supposed to read user documentation.

This is used as a way for making sure all the stakeholders of the system will have complete and clear understanding about the requirements of the system.

4.1.2 User Classes and Characteristics

There are two users in the system:

User :

- Start android app
- Sign-up
- Login
- Mount smart device in the vehicle
- Start journey and record data
- End journey
- Unmount smart device

System/app :

- The application will record data using sensors present on the smartphone.
- Data will be cleaned in the app
- Execution of machine learning algorithm on cloud servers
- Driver behaviour classification and safety score calculation

4.1.3 Operating Environment

The system is an android application that can work on any android device. It just requires a stable internet connection to avail all the facilities.

4.1.4 External Interface Requirements

User Interfaces: User interface will be an android application which consists of a Register and Login option. Where users need to register on the system before booking starting the trip.

Hardware Interfaces:

Smartphone with an Internet Connection.

Software Requirements:

Tools: Postman, flask, firebase, Android studio, Google app engine, Visual Studio

4.1.5 Functional Requirement

- View History (button)-When clicked on the button Driving info along with session as per the user should be displayed.
- Google map - Accurate map from the user’s location to the destination should be precise.

- Login - New user id and password or incorrect when entered shouldn't login into the system .Once logged in should open different UI depending on the user and admin.

- Sign-up - All the requirements for login & password with the inculcated rules should be satisfied .If not, the user entry for the same shouldn't be considered. New entry should be saved in the specific domain only.

- New trip - Once clicked should open google map to search the destination. In order to start the journey the user should select destination for directions.

4.1.6 Other Non-functional Requirements

• **Performance Requirements:**

1. Ability to maintain the data and train the model without crashing and classify driver behaviour along with safety score.

2. Quick recovery time.

3. The response time for safety score calculation will not be more than 2 seconds.

• Security Requirements:

1. System should not grant authentication to any unauthorised person.

2. The system should not be vulnerable to the security attacks.

• Software Quality Attributes:

The system ensures the following software quality attributes:

1. Reliability

2. Efficiency

3. Security

4. User-friendliness

5. Flexibility

4.1.7 Product Perspective

The product is supposed to be open source. It is android application integrated with machine learning model along with Google cloud platforms Google app engine and firebase for safety score calculation, driver behaviour classification and storing the data in real-time database.

- User account: The system allows the user to create their accounts in the system and provide features of updating and viewing profiles.

- Number of users being supported by the system: Though the number is precisely not mentioned but the system is able to support a number of users at a time.

- Search: search is simply a l search engine for searching location on Google map for directions.

4.1.8 Product Function

Some major product functionalities of the system are as follows:

This project can be divided into 4 main features namely:

- Data collection: Users will mount their smartphones in the vehicle. Upon start of journey data will be collected using sensors that are present on the smartphone.

- Data cleaning: Data collected through sensors will cleaned in the app.

- Execution of machine learning algorithm: Machine learning algorithm will be executed on input data on cloud servers.

- Driver behaviour calculation: Driver behaviour classification and safety score is calculated and data is stored in the database.

Chapter 5

Project Analysis and Design

5.1 Use Case Diagram

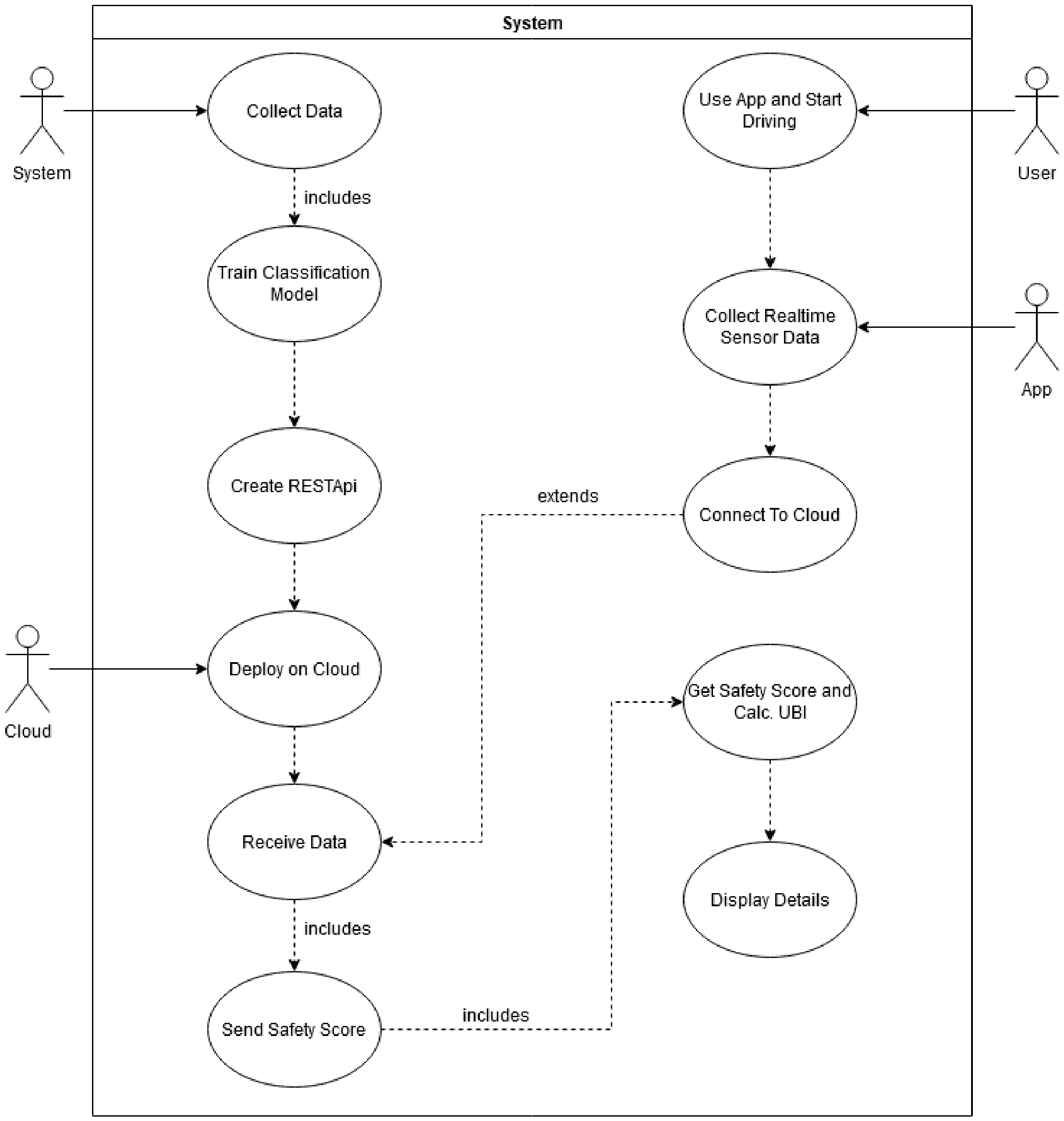
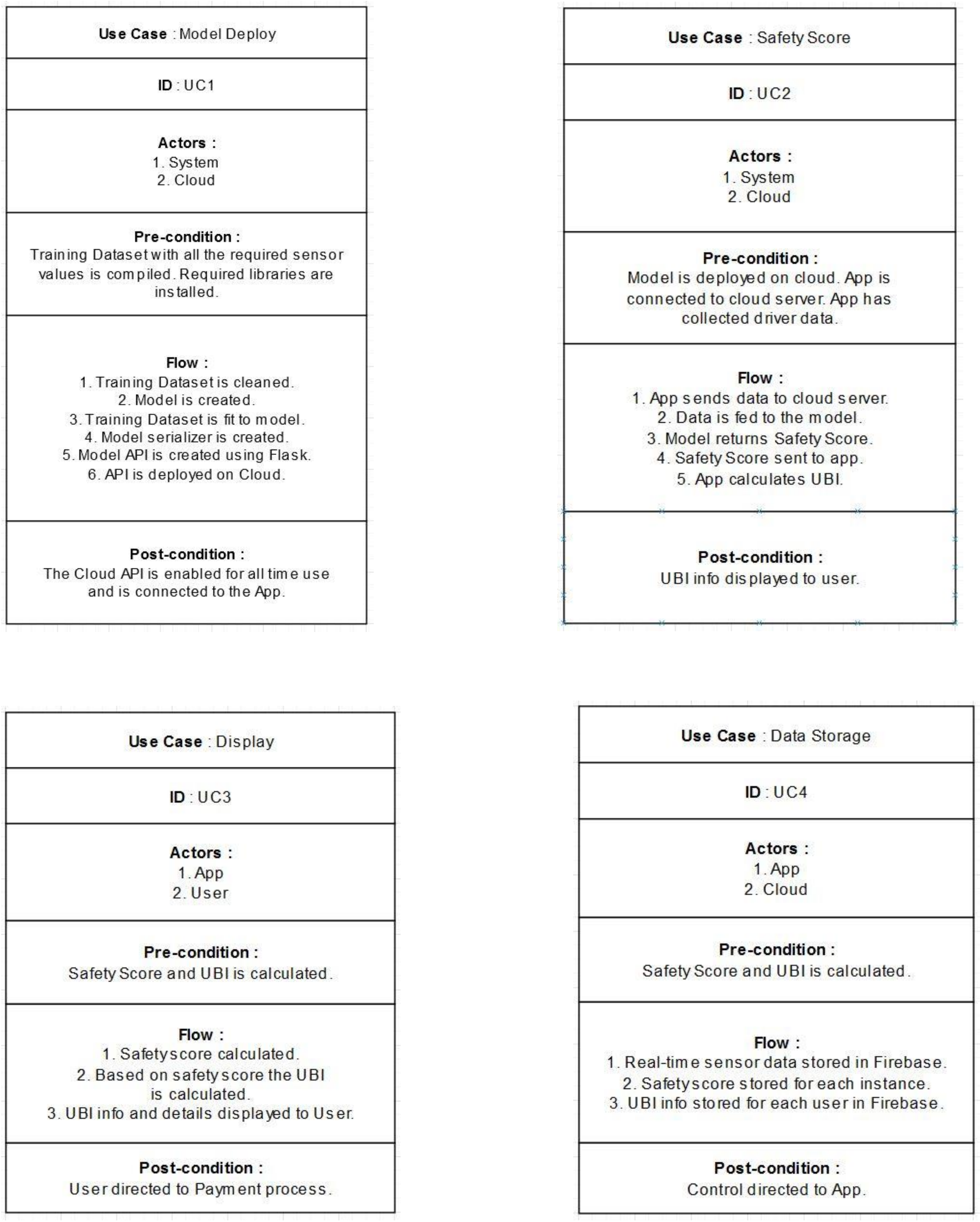


Fig. 5.1.1 Use Case Diagram



Use Case : Model Deploy
ID : UC1
Actors : 1. System 2. Cloud
Pre-condition : Training Dataset with all the required sensor values is compiled. Required libraries are installed.
Flow : 1. Training Dataset is cleaned. 2. Model is created. 3. Training Dataset is fit to model. 4. Model serializer is created. 5. Model API is created using Flask. 6. API is deployed on Cloud.
Post-condition : The Cloud API is enabled for all time use and is connected to the App.

Use Case : Safety Score
ID : UC2
Actors : 1. System 2. Cloud
Pre-condition : Model is deployed on cloud. App is connected to cloud server. App has collected driver data.
Flow : 1. App sends data to cloud server. 2. Data is fed to the model. 3. Model returns Safety Score. 4. Safety Score sent to app. 5. App calculates UBI.
Post-condition : UBI info displayed to user.

Use Case : Display
ID : UC3
Actors : 1. App 2. User
Pre-condition : Safety Score and UBI is calculated.
Flow : 1. Safety score calculated. 2. Based on safety score the UBI is calculated. 3. UBI info and details displayed to User.
Post-condition : User directed to Payment process.

Use Case : Data Storage
ID : UC4
Actors : 1. App 2. Cloud
Pre-condition : Safety Score and UBI is calculated.
Flow : 1. Real-time sensor data stored in Firebase. 2. Safety score stored for each instance. 3. UBI info stored for each user in Firebase.
Post-condition : Control directed to App.

Fig. 5.1.2 Use Case Specifications

Use Case Description : Use Case diagram shows User uses app and starts driving. The app collects realtime sensor data. This data is sent to cloud server where the safety score for driver is calculated. The system has trained a classification model which is deployed on the cloud server. The details are then displayed to the user and necessary information is stored in the database.

5.2 Activity Diagram

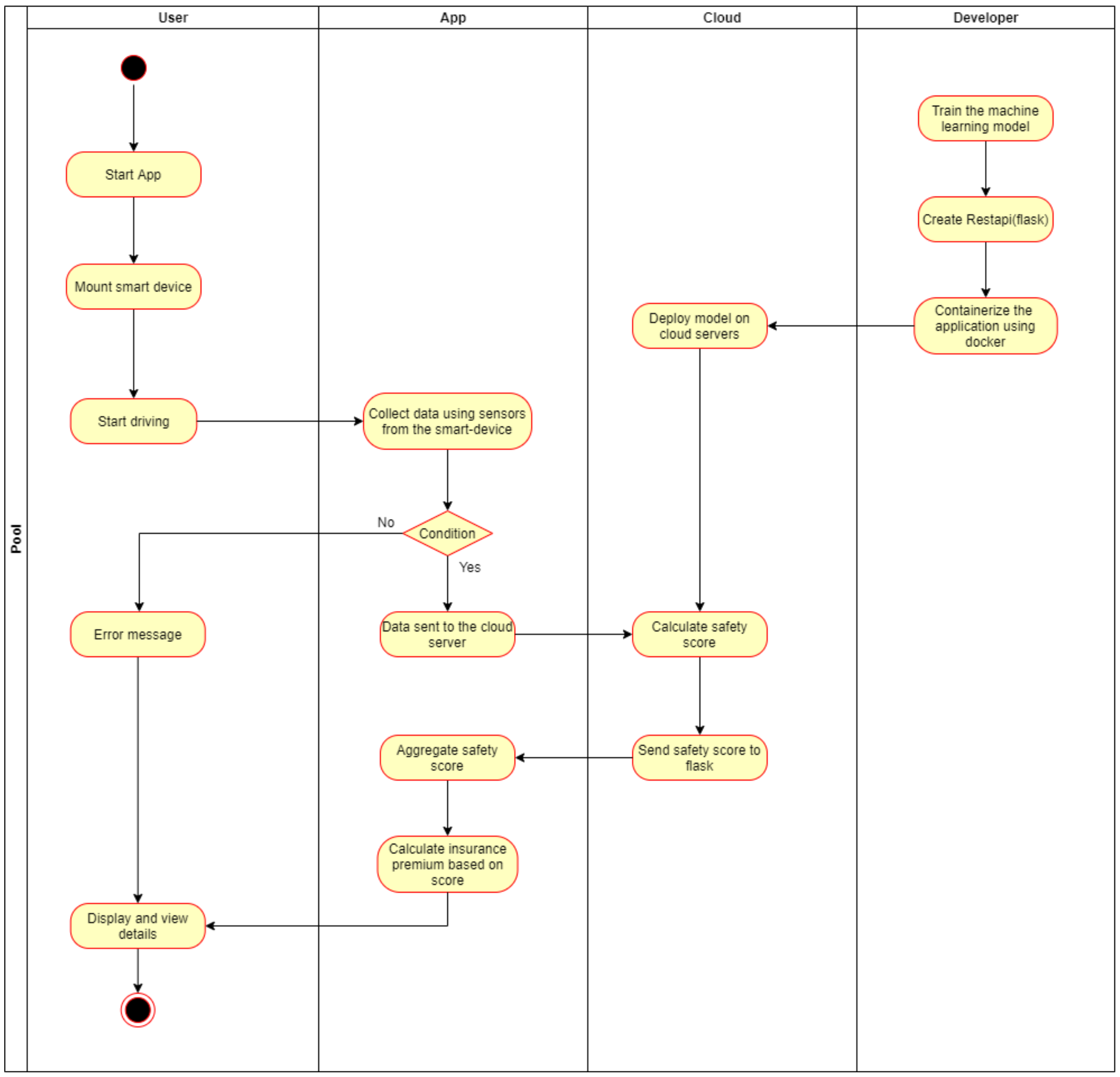

Pool
User
App
Cloud
Developer
Start App
Mount smart device
Start driving
Collect data using sensors from the smart-device
Condition
No
Yes
Error message
Data sent to the cloud server
Train the machine learning model
Create Restapi(flask)
Containerize the application using docker
Deploy model on cloud servers
Calculate safety score
Send safety score to flask
Aggregate safety score
Calculate insurance premium based on score
Display and view details

Fig. 5.2.1 Activity Diagram

Activity Diagram description : An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. A swim lane is a way to group activities performed by the same actor on an activity diagram or activity diagram or to group activities in a single thread. In the above diagram the different swim lanes that we have are User, App, Cloud, and Developer. The flow of the system is depicted in the above diagram.

5.3 Sequence Diagram

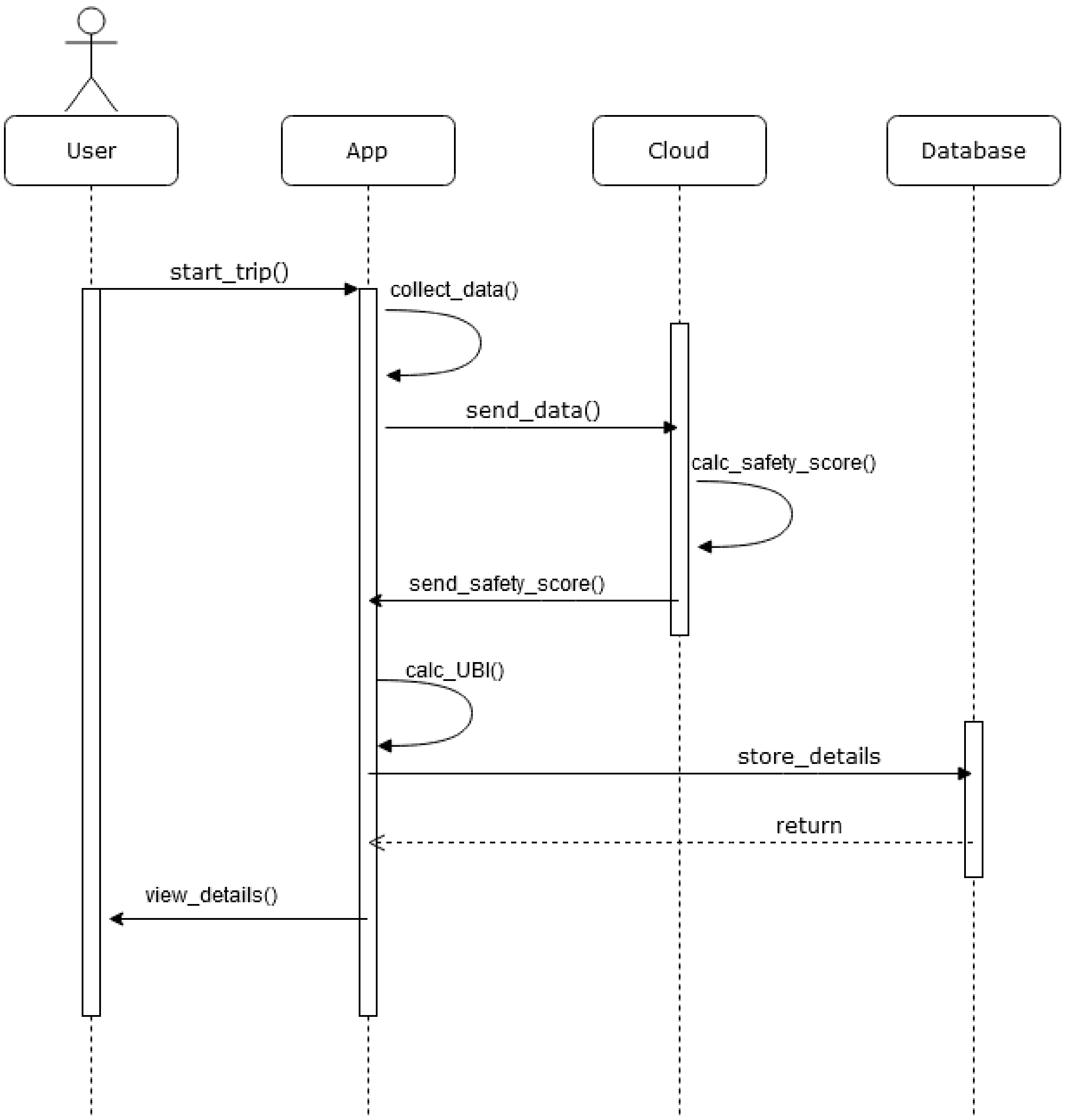


Fig. 5.3.1 Sequence Diagram

Sequence Diagram description : UML Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when. The vertical axis represents time proceedings (or progressing) down the page.

5.4 State Chart Diagram

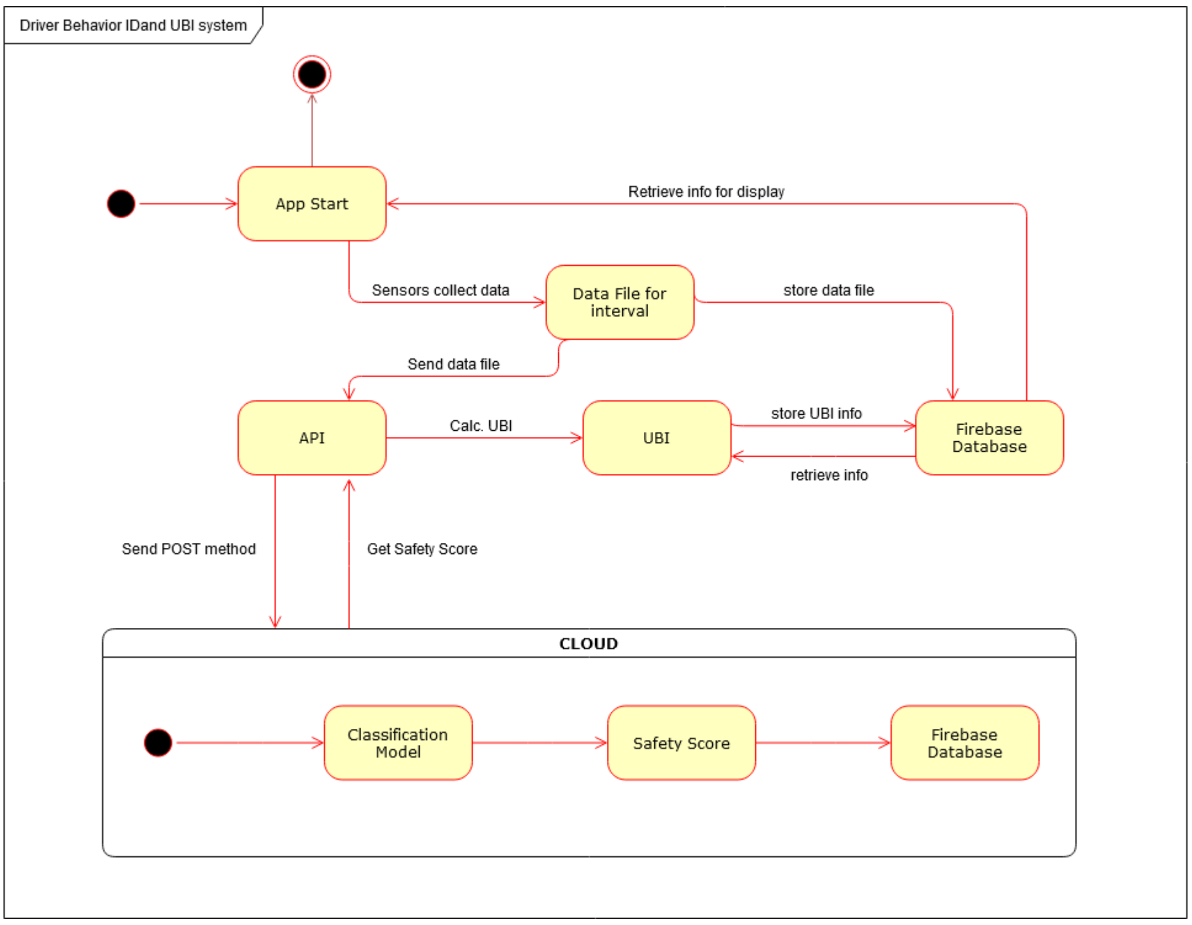


Fig. 5.4.1 State Chart Diagram

State Chart description : UML State Machine Diagrams (or sometimes referred to as state diagram, state machine or state chart) show the different states of an entity. State machine diagrams can also show how an entity responds to various events by changing from one state to another. State machine diagram is a UML diagram used to model the dynamic nature of a system. UML State Machine Diagrams (or sometimes referred to as state diagram, state machine or state chart) show the different states of an entity. State machine diagrams can also show how an entity responds to various events by changing from one state to another. State machine diagram is a UML diagram used to model the dynamic nature of a system.

5.5 Time Schedule

Task	Start	End	Status
Formation of problem statement ,scope, objective, business objective of the project and learning Phase	January	February	Completed
Literature analysis, Formation of the system architecture and Brainstorming	February	March	Completed
Developing Model, deployment on cloud	March	April	Completed
Integration of cloud and database with App.	April	May	Completed
Making the final report, collecting feedback from project guide	May	June	Completed

Table 5.5.1 Time Schedule

5.6 Team Organization

5.6.1 Team Structure

Our team consists of developers, internal mentor and external mentor.

Developers :

- Shardul Nazirkar
- Harsh Shinde
- Chetan Patil

Mentors :

- Mr Pawan Wawage (Internal)
- Mr Pankaj Shinde (External)

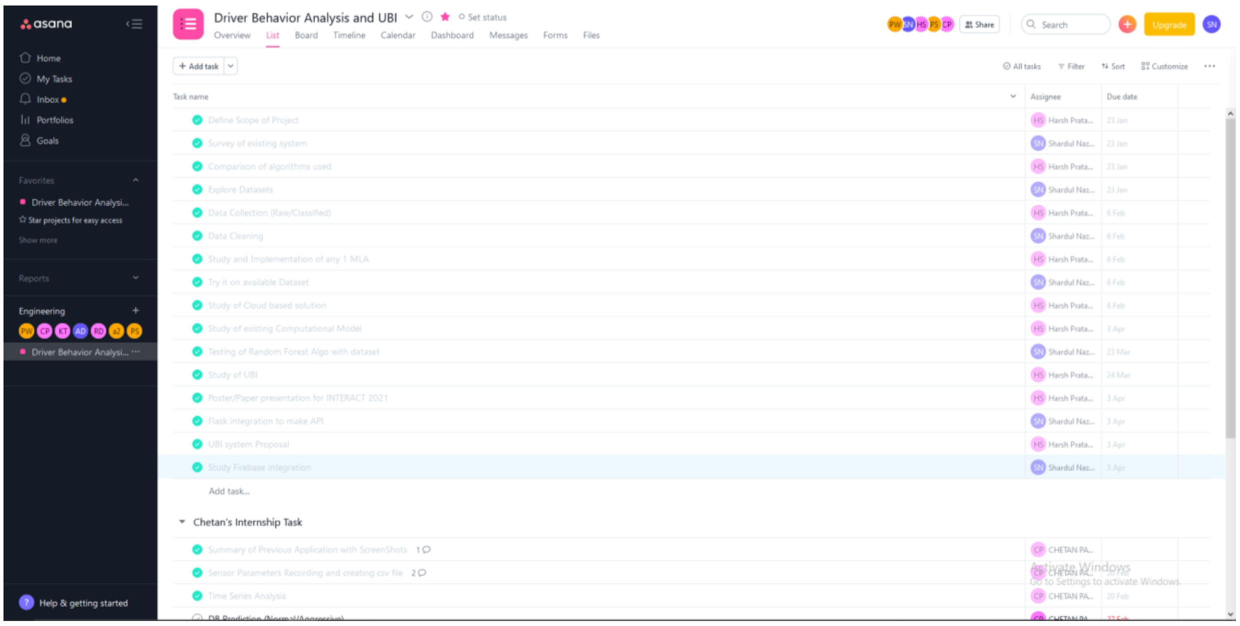
5.6.2 Agile Methodology Practice

In our project, we followed agile methodology. AGILE methodology is a practice that promotes continuous iteration of development and testing throughout the software development life cycle of the project. In the Agile model, both development and testing activities are concurrent.

- Daily Scrum Calls : In our daily scrum calls (Duration 15-30 minutes) in the presence of all developers in the project group, we discussed about the daily updates of work, roadblocks and progress regarding the same. Also, we planned the task heading up.

- Weekly Review Meeting : In our weekly review meeting, in the presence of all group members and mentors, we discussed the progress of work done in that particular week , the work that should be done in the week showing up next. Also, we discussed all possible progress and roadblock factors and solutions towards the same.

- Scrum Artefacts : We portrayed an Asana dashboard on sprint basis (1 Sprint = 2 weeks) which describes tasks to be done , by whom to be done and number of hours worked towards the same.



Task name	Assignee	Due date
Define Scope of Project	Harsh Prata...	23 Jan
Survey of existing system	Shardul Naz...	23 Jan
Comparison of algorithms used	Harsh Prata...	23 Jan
Explore Datasets	Shardul Naz...	23 Jan
Data Collection (Raw/Classified)	Harsh Prata...	6 Feb
Data Cleaning	Shardul Naz...	6 Feb
Study and Implementation of any 1 MLA	Harsh Prata...	6 Feb
Try it on available Dataset	Shardul Naz...	6 Feb
Study of Cloud based solution	Harsh Prata...	6 Feb
Study of existing Computational Model	Harsh Prata...	3 Apr
Testing of Random Forest Algo with dataset	Shardul Naz...	23 Mar
Study of UBI	Harsh Prata...	24 Mar
Poster/Paper presentation for INTERACT 2021	Harsh Prata...	3 Apr
Flask integration to make API	Shardul Naz...	3 Apr
UBI system Proposal	Harsh Prata...	3 Apr
Study Firebase integration	Shardul Naz...	3 Apr
Add task...		
Chetan's Internship Task		
Summary of Previous Application with ScreenShots 1	CHETAN PA...	
Sensor Parameters Recording and creating csv file 2	CHETAN PA...	20 Feb
Time Series Analysis	CHETAN PA...	20 Feb
DB Prediction (Normal/Aggressive)	CHETAN PA...	27 Feb

Activate Windows
Go to Settings to activate Windows.

Fig. 5.6.1 Agile methodology using Asana

5.6.3 Tools : Project Management

- Google Meet – For Weekly meetings. Extra learning sessions were held in teams , daily updates were posted through comments , buckets of the respective task were changed on basis of (to do , doing , done)
- Discord – For Daily meetings and maintaining important links and files.

- Gmail – Mail
- Git – Pushing the actual Implementation

5.6.4 Learning Sessions

Depending on the various learning phases in our project we had collaborative sessions to learn said technologies and topics.

Topics:

- Scikit learn
- Data Exploration
- Literature Analysis
- Flask
- Google Cloud
- Firebase

Chapter 6

Software Testing

6.1 Introduction

This document is a high-level overview defining our testing strategy for the application. Its objective is to test the model and integration with the cloud. It portrays a snapshot of the project as of the end of the planning phase. This document will address the different standards that will apply to the unit, integration and system testing of the specified application.

6.2 Purpose

The project aims at providing a user-friendly app for a service of Usage Based Insurance to the User. In the Driver Behaviour and UBI system, we collect mobile sensor data while driving and calculate the safety score for the driver. This safety score is in turn used to calculate UBI for that particular user.

6.3 Test Objective

The objective of our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. There will be five key functions used to manage our application: load, store, search, insert, delete. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the data. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

6.4 Process Overview

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.

2. Identify which particular test will be used to test each module.

3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.

4. Identify the expected results for each test.

5. Document the test case configuration, test data, and expected results.

6. Perform the test.

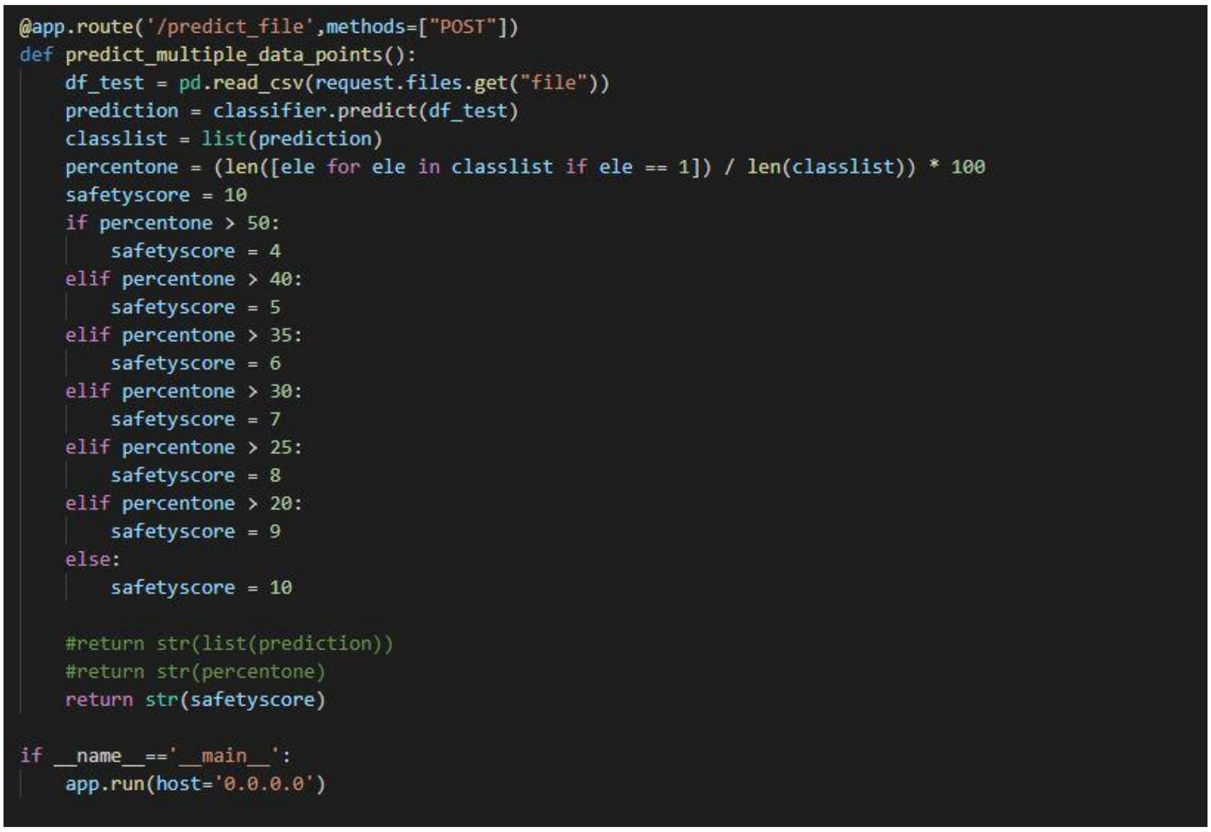
7. Document the test data, test cases, and test configuration used during the testing process.

8. Successful Unit testing is required before the unit is eligible for component integration/system testing.

9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.

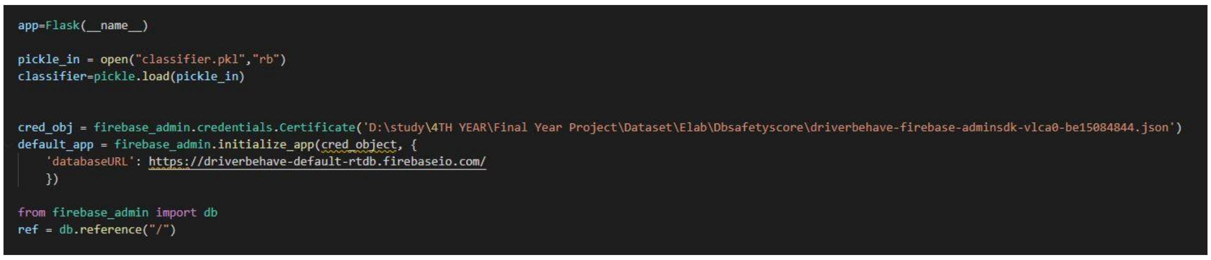
10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

6.5 Implementation



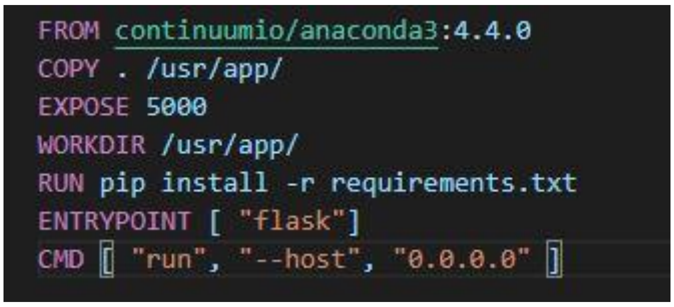
```
@app.route('/predict_file',methods=["POST"])
def predict_multiple_data_points():
    df_test = pd.read_csv(request.files.get("file"))
    prediction = classifier.predict(df_test)
    classlist = list(prediction)
    percentone = (len([ele for ele in classlist if ele == 1]) / len(classlist)) * 100
    safetyscore = 10
    if percentone > 50:
        safetyscore = 4
    elif percentone > 40:
        safetyscore = 5
    elif percentone > 35:
        safetyscore = 6
    elif percentone > 30:
        safetyscore = 7
    elif percentone > 25:
        safetyscore = 8
    elif percentone > 20:
        safetyscore = 9
    else:
        safetyscore = 10

    #return str(list(prediction))
    #return str(percentone)
    return str(safetyscore)

if __name__=='__main__':
    app.run(host='0.0.0.0')
```

Fig 6.5.1 API to calculate Safety Score



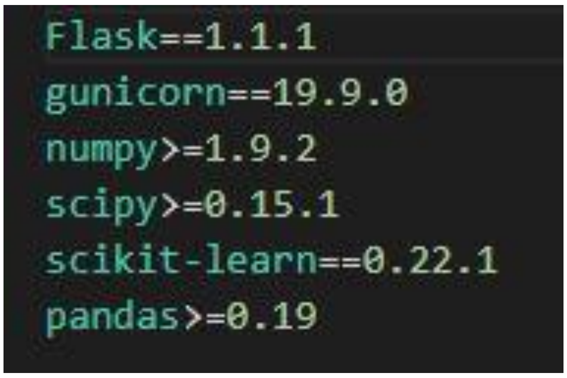
```
app=Flask(__name__)

pickle_in = open("classifier.pkl","rb")
classifier=pickle.load(pickle_in)


cred_obj = firebase_admin.credentials.Certificate('D:\study\4TH YEAR\Final Year Project\Dataset\Elab\Dbsafetyscore\driverbehave-firebase-adminsdk-vlca0-be15084844.json')
default_app = firebase_admin.initialize_app(cred_object, {
    'databaseURL': https://driverbehave-default-rtdb.firebaseio.com/
    })

from firebase_admin import db
ref = db.reference("/")
```

Fig 6.5.2 Setting up Flask, Firebase Admin



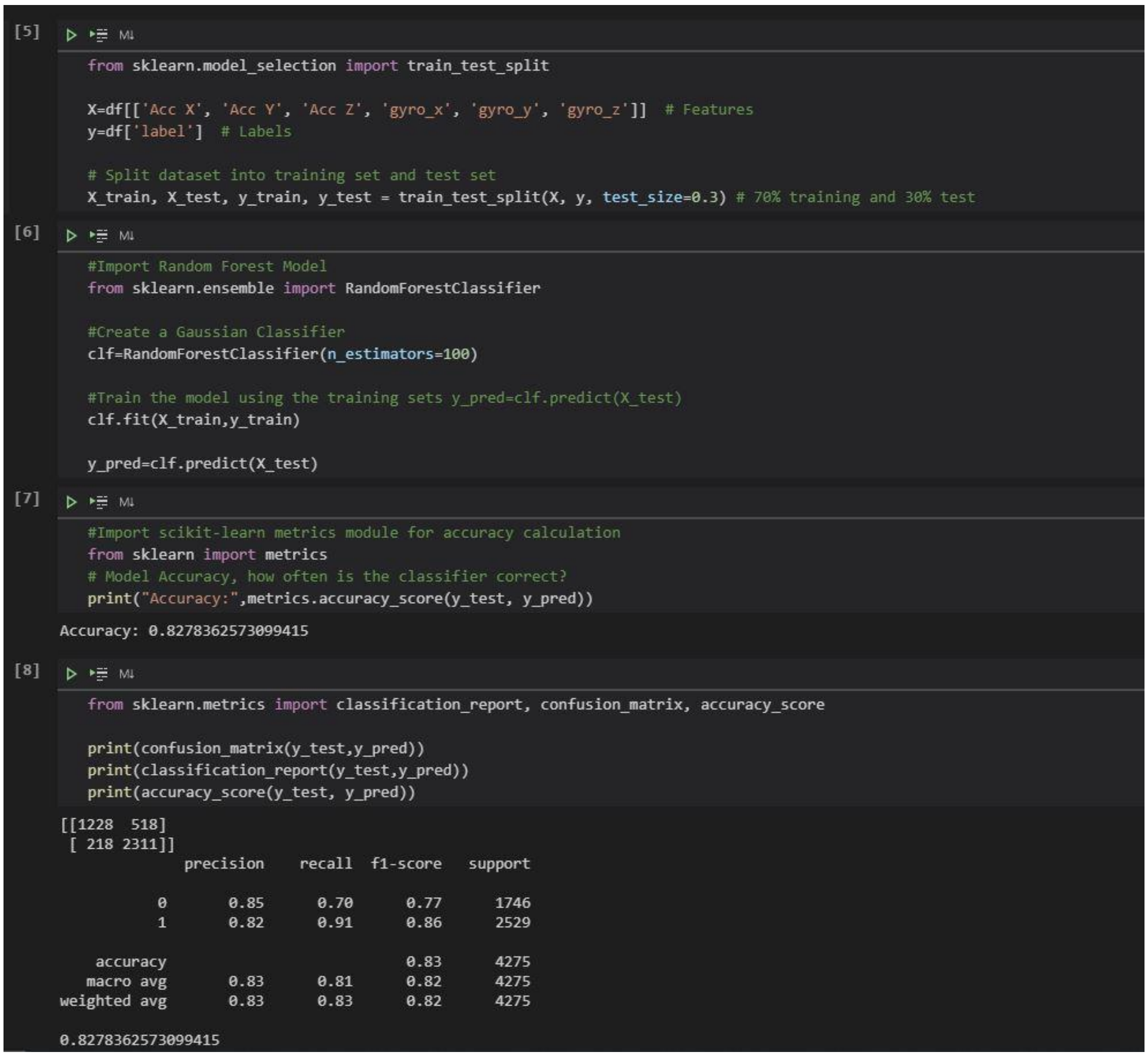
```
FROM continuumio/anaconda3:4.4.0
COPY . /usr/app/
EXPOSE 5000
WORKDIR /usr/app/
RUN pip install -r requirements.txt
ENTRYPOINT [ "flask"]
CMD [ "run", "--host", "0.0.0.0" ]
```

Fig 6.5.3 Docker File



```
Flask==1.1.1
gunicorn==19.9.0
numpy>=1.9.2
scipy>=0.15.1
scikit-learn==0.22.1
pandas>=0.19
```

Fig 6.5.4 Requirements



```
[5]
from sklearn.model_selection import train_test_split

X=df[['Acc X', 'Acc Y', 'Acc Z', 'gyro_x', 'gyro_y', 'gyro_z']]  # Features
y=df['label']  # Labels

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # 70% training and 30% test
```

```
[6]
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
```

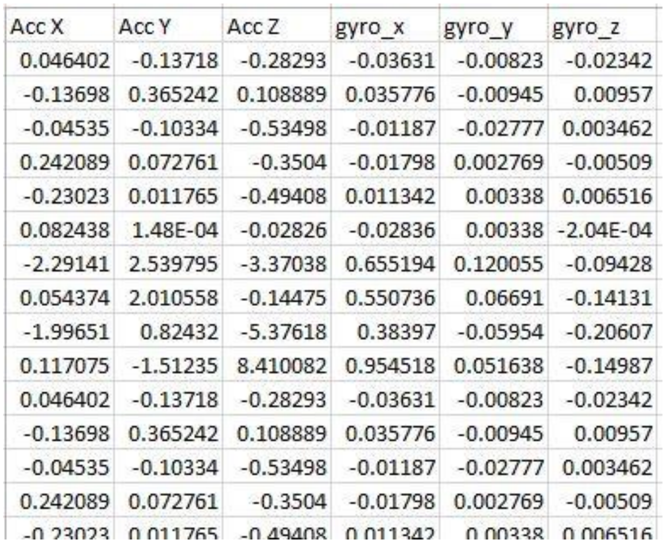
```
[7]
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.8278362573099415
```

```
[8]
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

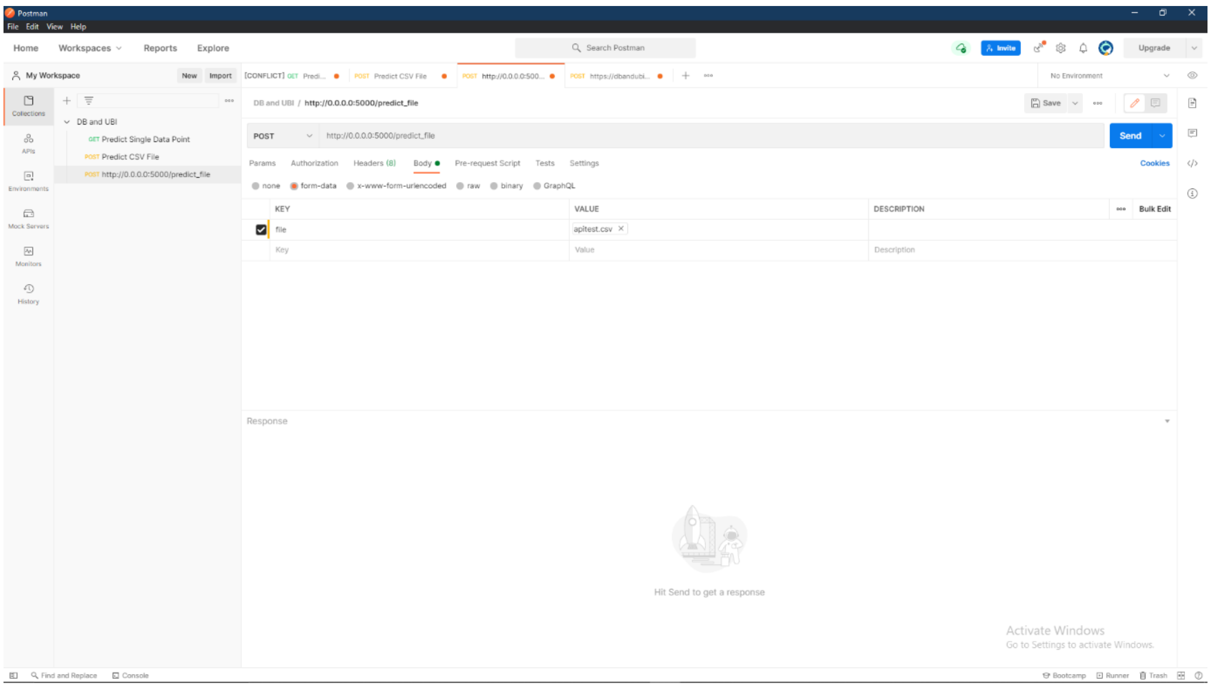
```
[[1228  518]
 [ 218 2311]]
              precision    recall  f1-score   support

           0       0.85      0.70      0.77      1746
           1       0.82      0.91      0.86      2529

    accuracy                           0.83      4275
   macro avg       0.83      0.81      0.82      4275
weighted avg       0.83      0.83      0.82      4275

0.8278362573099415
```

Fig 6.5.5 Developing the Model and Accuracy



Acc X	Acc Y	Acc Z	gyro_x	gyro_y	gyro_z
0.046402	-0.13718	-0.28293	-0.03631	-0.00823	-0.02342
-0.13698	0.365242	0.108889	0.035776	-0.00945	0.00957
-0.04535	-0.10334	-0.53498	-0.01187	-0.02777	0.003462
0.242089	0.072761	-0.3504	-0.01798	0.002769	-0.00509
-0.23023	0.011765	-0.49408	0.011342	0.00338	0.006516
0.082438	1.48E-04	-0.02826	-0.02836	0.00338	-2.04E-04
-2.29141	2.539795	-3.37038	0.655194	0.120055	-0.09428
0.054374	2.010558	-0.14475	0.550736	0.06691	-0.14131
-1.99651	0.82432	-5.37618	0.38397	-0.05954	-0.20607
0.117075	-1.51235	8.410082	0.954518	0.051638	-0.14987
0.046402	-0.13718	-0.28293	-0.03631	-0.00823	-0.02342
-0.13698	0.365242	0.108889	0.035776	-0.00945	0.00957
-0.04535	-0.10334	-0.53498	-0.01187	-0.02777	0.003462
0.242089	0.072761	-0.3504	-0.01798	0.002769	-0.00509
-0.23023	0.011765	-0.49408	0.011342	0.00338	0.006516

Fig 6.5.6 Test File structure



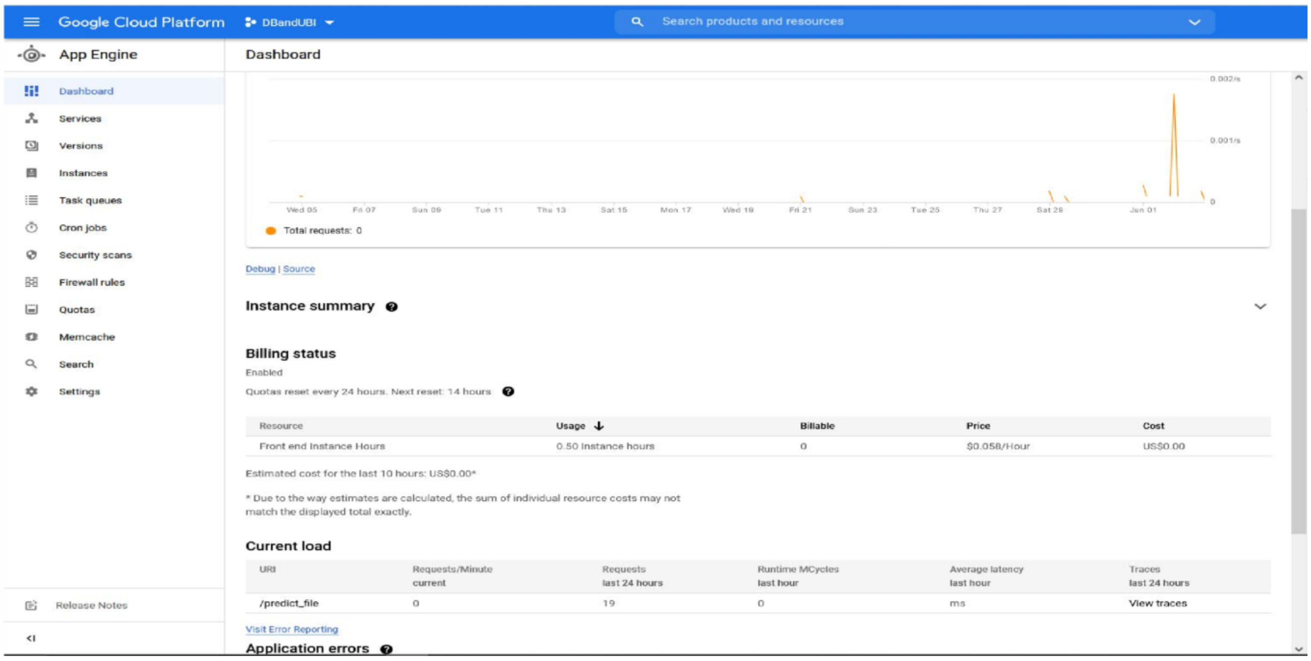
DB and UBI / http://0.0.0.0:5000/predict_file

POST http://0.0.0.0:5000/predict_file

none [✓ form-data] x-www-form-urlencoded raw binary GraphQL

	KEY	VALUE	DESCRIPTION
✓	file	apitest.csv	
	Key	Value	Description

Fig 6.5.7 Postman Implementation



Dashboard

● Total requests: 0

Instance summary

Billing status

Enabled

Quotas reset every 24 hours. Next reset: 14 hours

Resource	Usage	Billable	Price	Cost
Front end Instance Hours	0.50 Instance hours	0	$0.058/Hour	US$0.00

Estimated cost for the last 10 hours: US$0.00*

* Due to the way estimates are calculated, the sum of individual resource costs may not match the displayed total exactly.

Current load

URI	Requests/Minute current	Requests last 24 hours	Runtime MCycles last hour	Average latency last hour	Traces last 24 hours
/predict_file	0	19	0	ms	View traces

Application errors

Fig 6.5.8 Google app Engine

Chapter 7
Conclusion and Future Work

7.1 Conclusion

In this project we have successfully implemented the tasks required to develop and end to end project. The finalization of project scope gave us a clear idea of execution of the project development. We have successfully applied a classification model that has a good accuracy and is efficient and scalable. The deployment on cloud server and integrating it with app helped us to make an end to end project that is according to the modern technologies and its trends. The insights and guidance provided by our mentors helped us in carrying out the project in a systematic and organized way.

7.2 Future Scope

In our Driver Behaviour ID and UBI project, further modules can be designed as per the updates in scope, integration of the other modules in scope. The classification model can be upgraded using Deep Learning methods or ensemble methods to increase the accuracy and efficiency. The model can be further researched using different types of training data like data vectorization and enhanced hyper parameter tuning. The app can be updated with new features like drowsiness detection and real-time alerts, etc.

Chapter 8

References

1. Subramanian Arumugam and R. Bhargavi - A survey on driving behavior analysis in usage based insurance using big data [1]
2. Isaac Skog, Peter H¨andel, Martin Ohlsson, and Jens Ohlsson - Challenges In Smartphone-Driven Usage Based Insurance [2]
3. Sarah Kadhim Alluhaibi, Munaf S. Najim Al-Din, Aiman Moyaid - Driver Behavior Detection Techniques: A survey [3]
4. Jair Ferreira Junior, Eduardo Carvalho, Bruno V. Ferreira, Cleidson de Souza, Yoshihiko Suhara, Alex Pentland, Gustavo Pessin - Driver behavior profiling: An investigation with different smartphone sensors and machine learning [4]
5. Isaac Skog, Peter H¨andel, Martin Ohlsson, and Jens Ohlsson - Smartphone-Based Measurement Systems for Road Vehicle Traffic Monitoring and Usage-Based Insurance [5]
6. Eleni Mantouka, Emmanouil Barmpounakis, Eleni Vlahogianni, John Golias - Smartphone sensing for understanding driving behavior: Current practice and challenges [6]
7. https://flask.palletsprojects.com/en/2.0.x/
8. https://docs.python.org/3/
9. https://cloud.google.com/appengine
10. https://scikit-learn.org/stable/user_guide.html
11. https://www.zendrive.com/blog