COMP4442 Service and Cloud Computing

Group Project

Name	Student ID	Email
Cheung Sui Wing	21027547D	21027547D@connect.polyu.hk
Lau Man Chun	21027257D	21027257D@connect.polyu.hk
Kwong Chun Him	21028468D	21028468D@connect.polyu.hk
Cheng Chi Kit	21028079D	21028079D@connect.polyu.hk

Table of Contents

Group tasks performed by each member

Cheung Sui Wing

- Spark script
- Database SQL, upload data script, initSQL
- Flask App: summary bar chart and pie chart
- Setup/deploy the app in AWS S3, EC2, RDS, EMR
- Report Document

Lau Man Chun

- Summary table/ fetch SQL/ filter etc.
- Monitor record page, bug fix
- Doc

Kwong Chun Him

- Monitor function and monitor page
- Monitor record page, record database/SQL
- Report Document

Cheng Chi Kit

- Testing
- Deploy the app in AWS
- Report Document

AWS App link:

http://web-env.eba-qttvhmqs.us-east-1.elasticbeanstalk.com/

Folder structure:

```
.
├── data-after-spark            # data after running the AWS EMR
├── DB                          # Folder for code related to DB/Spark
│   ├── connection.py           # code of connecting to DB
│   ├── initDB.sql              # query to create database and table
│   ├── Spark_aws_EMR.py        # Spark code on AWS EMR
│   ├── spark_colab.ipynb       # Spark code on Google Colab (for testing,debug etc)
│   ├── gen_data_for_monitor.py # generate data and insert to monitor table
│   ├── readData.py             # upload given speed data to DB
│   └── upload_data_to_DB.py    # read the csv in data-after-spark and upload to DB
├── detail-record               # original data set
├── FlaskApp                    # Folder for Flask app code
│   ├── website                 # website source code
│   │      ├── static           # All css/js/images put here
│   │      ├── templates        # All html files put here
│   │      ├── __init__.py      # Flask init code
│   │      └── views.py         # code for define URL route
│   └── application.py          # main.py
├── website.zip                 # zip file used to upload to AWS Elastic Beanstalk
├── Demo Video.mp4              # Demo video
├── Report.pdf                  # Report document
└── ...
```

System Specification

Operation system

AWS EC2:

- Python 3.8 running on 64bit Amazon Linux 2/3.3.12

In Local:

- Windows 10 / Google Colab

Programming language with version

Python 3.8

MySQL 8.0.28

System software

In AWS:

- AWS Spark,
- AWS EMR,
- AWS S3,
- AWSElastic BeanStalk,
- AWSElastic BeanStalk Loadbalancer

In Local:

- Visual Studio code
- Google colab
- MySQL workbench 8.0

Required packages

certifi==2020.6.20
click==8.1.2
colorama==0.4.4
Flask==2.1.1
itsdangerous==2.1.2
Jinja2==3.1.1
MarkupSafe==2.1.1
mysql-connector==2.2.9
numpy==1.22.3
pandas==1.4.2
py4j==0.10.9.3
PyMySQL==1.0.2
pyspark==3.2.1
python-dateutil==2.8.2
pytz==2022.1
six==1.16.0
Werkzeug==2.1.1
wincertstore==0.2

Functional modules

Key functions

Summary Web Page

The Summary page includes all the functions below, it is for viewing the driving behavior of the people in a selected amount of date and time or viewing a specific person's driving behavior.

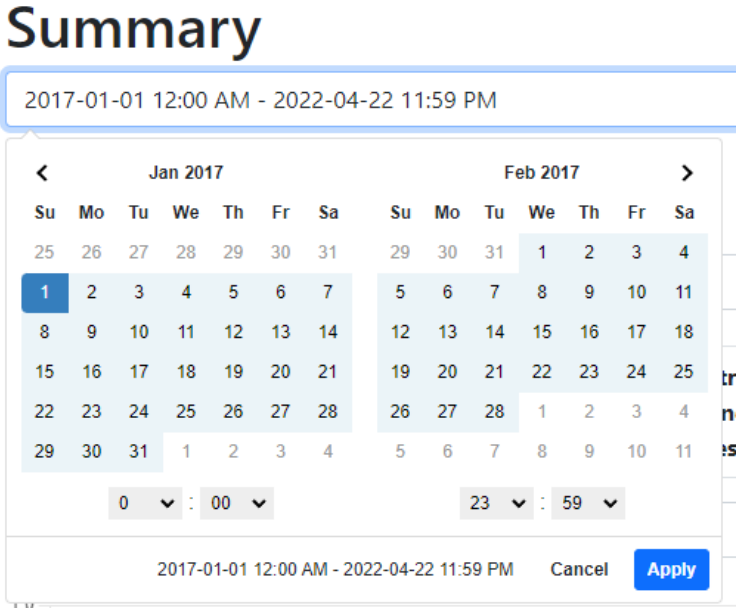
- **Summary data table**

  Show the summary data according to the the selecting day in table form

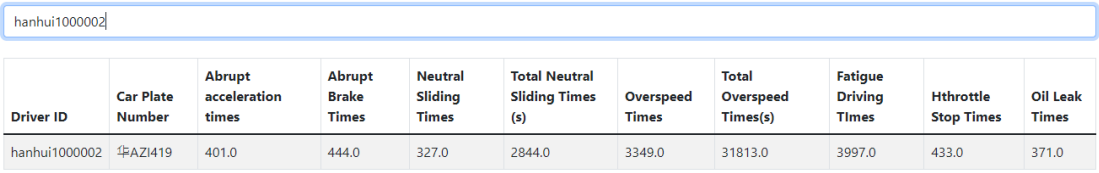
Driver ID	Car Plate Number	Abrupt acceleration times	Abrupt Brake Times	Neutral Sliding Times	Total Neutral Sliding Times (s)	Overspeed Times	Total Overspeed Times(s)	Fatigue Driving Times	Hthrottle Stop Times	Oil Leak Times
hanhui1000002	皖AZI419	401.0	444.0	327.0	2844.0	3349.0	31813.0	3997.0	433.0	371.0
haowei1000008	皖A709GB	321.0	314.0	255.0	2659.0	2635.0	25522.0	3204.0	312.0	318.0
likun1000003	皖AVM936	341.0	354.0	291.0	3043.0	3043.0	28728.0	3552.0	347.0	376.0
panxian1000005	皖AX542C	395.0	434.0	330.0	2930.0	3530.0	33946.0	4307.0	417.0	441.0
shenxian1000004	皖ADJ750	374.0	356.0	297.0	2810.0	3126.0	31494.0	3767.0	383.0	366.0
xiexiao1000001	皖AEB132	264.0	261.0	248.0	2525.0	2320.0	23434.0	2720.0	314.0	253.0
xiezhi1000006	皖A6CU11	255.0	310.0	254.0	2074.0	2535.0	23942.0	2931.0	312.0	279.0
zengpeng1000000	皖AZQ110	340.0	344.0	272.0	2894.0	2762.0	25479.0	3274.0	284.0	337.0
zouan1000007	皖A58M83	360.0	385.0	315.0	2997.0	3179.0	31248.0	3594.0	389.0	385.0
duxu1000009	皖AT75H8	238.0	284.0	247.0	2632.0	2300.0	22338.0	2814.0	264.0	248.0

- **Date and time Selection**

  Select data and time for data to be shown and for generating bar chart and pie chart.

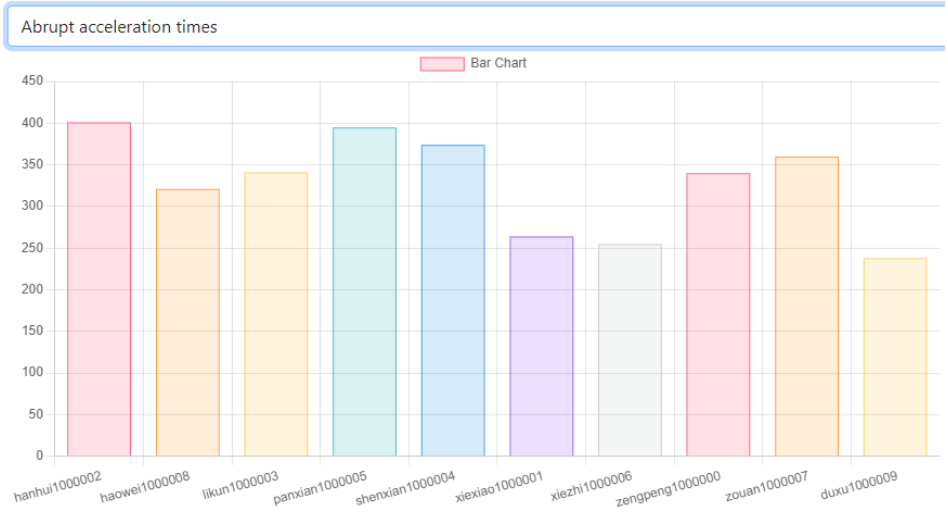


- **Filter**
  Find driving behavior according to DriverID or Car Plate Number, etc. (only filter the data existing in the table)

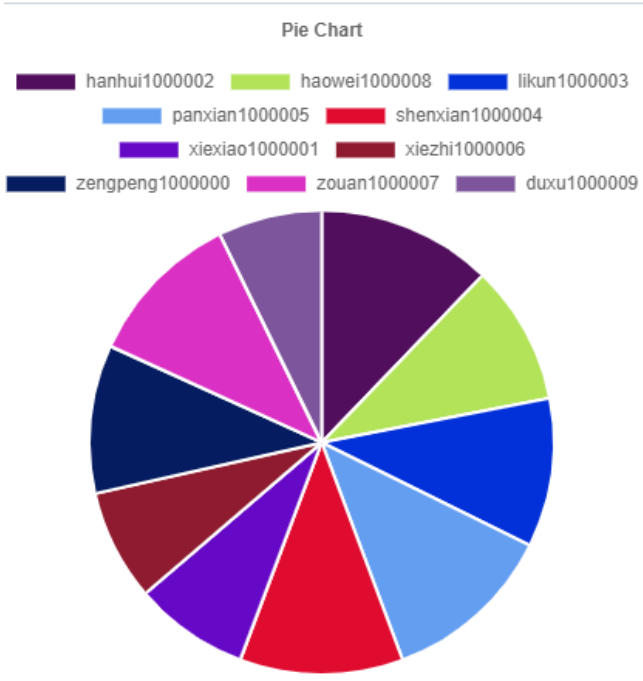


Driver ID	Car Plate Number	Abrupt acceleration times	Abrupt Brake Times	Neutral Sliding Times	Total Neutral Sliding Times (s)	Overspeed Times	Total Overspeed Times(s)	Fatigue Driving TImes	Hthrottle Stop Times	Oil Leak Times
hanhui1000002	华AZI419	401.0	444.0	327.0	2844.0	3349.0	31813.0	3997.0	433.0	371.0

- **Generating Bar Chart**
  Generate a Bar Chart according to the data of each or every driver, which include abrupt acceleration times, abrupt brake times, overspeed times, etc.

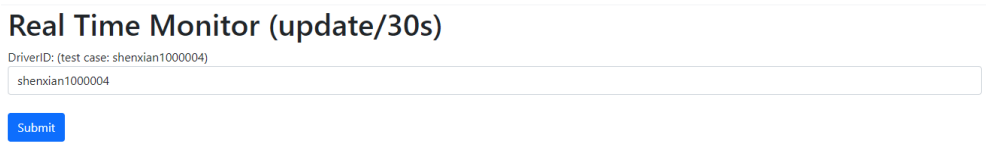


- **Generating Pie Chart**
  Generate a Pie Chart according to the data of each or every driver, which include abrupt acceleration times, abrupt brake times, overspeed times, etc.

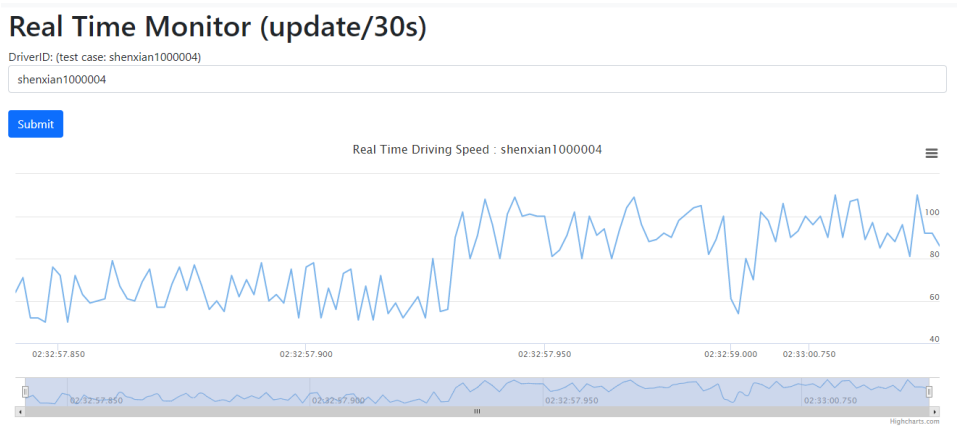


Monitor Web Page

- Can search the driver ID that you want to monitor.

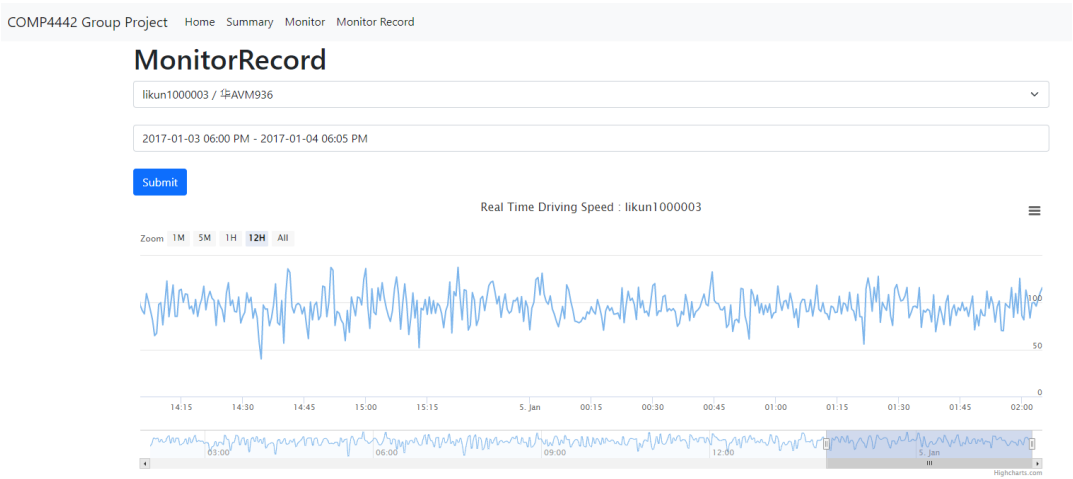


- Show the real time driving speed of this driver (update the data every 30 seconds), when the speed of driver high than 80 will alert “speed too fast” message.



Monitor Record Page

Show the Monitor Record according to the selected driver and select time period (show the speed data from the dataset provided)



Spark

- File for using in EMR: DB/Spark_aws_EMR.py
- File for using in development (Google Colab) : DB/ spark_colab.ipynb
- We hard code the S3 input output path, no need to use sys.args.

Basic logic

- First, read all 10 text data
- Then, Filter out all the record that do no have any special behaviour
  e.g.:

```
where isRapidlySpeedup > 0
OR isRapidlySlowdown > 0
OR isNeutralSlide > 0
OR isNeutralSlideFinished > 0
OR isOverspeed > 0
OR isOverspeedFinished > 0
OR isFatigueDriving > 0
OR isHthrottleStop > 0
OR isOilLeak > 0)
```

  In our case, only count the line that have more than 8 records

```
counts = text_file.map(lambda line: line.split(",")).filter(lambda line: len(line)>8)
```

- Then use driver, carPlateNumber, Time as a key to sum up all other behaviour primary in that day.
- Final output as a single csv file

Reference: Huawai Driving Behavior Data Analysis [1]

Upload data to database

- Path: DB/upload_data_to_DB.py
- Upload csv data generate by the spark function

Generate data for real time monitor

- Path: DB/ gen_data_for_monitor.py
- Generate random data for real time monior function

Relationship among them

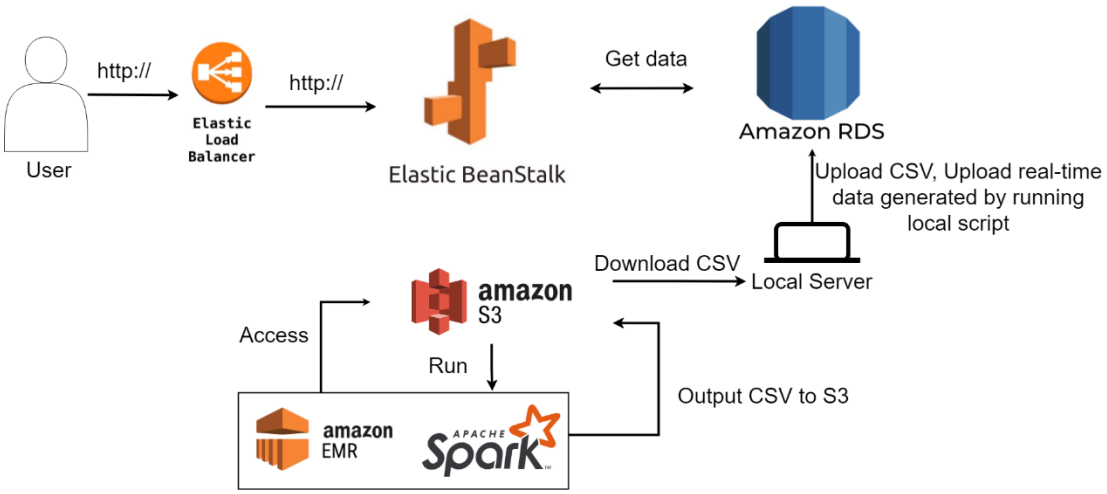
Summary and Moniotr are the fontend function

Summary includes functions that are Data and time selection, Filter, Generating Bar Chart, and Generating Pie Chart, they all are configed into a single page for easier browsing. The data used is come from spark processing.

The Monitor Web Page shows the real-time driving data generated by using the local script.

System architecture

(System architecture to show how you orchestrate the AWS services.)



Deployment procedures

Operations in AWS

1. Create storage Buckets

Name	AWS Region	Access
comp4442-group-project	US East (N. Virginia) us-east-1	Objects can be public

2. Upload the python script and data to the bucket

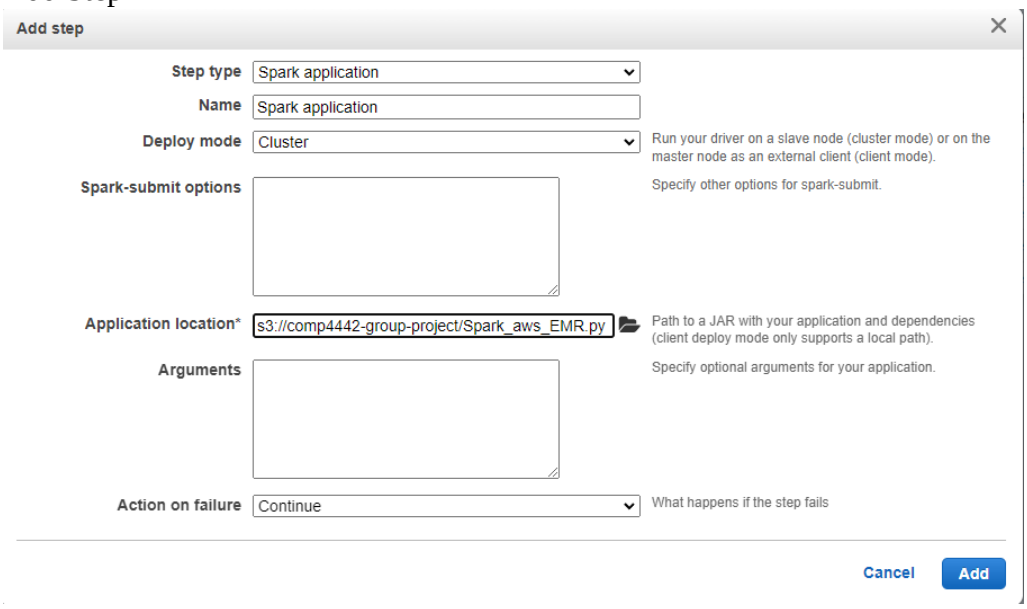
Name	Type
data/	Folder
output/	Folder
Spark_aws_EMR.py	py

data/ storing the 10 orginal data
output/ use to store the csv output after running the python script
and the **python script**

3. Setup EMR

Name	ID
comp4442-project	j-KQ1WYQNIRTF4

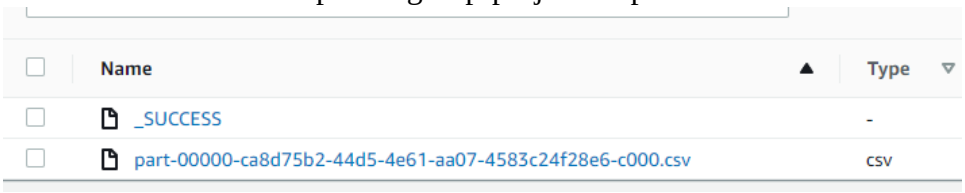
4. Add Step



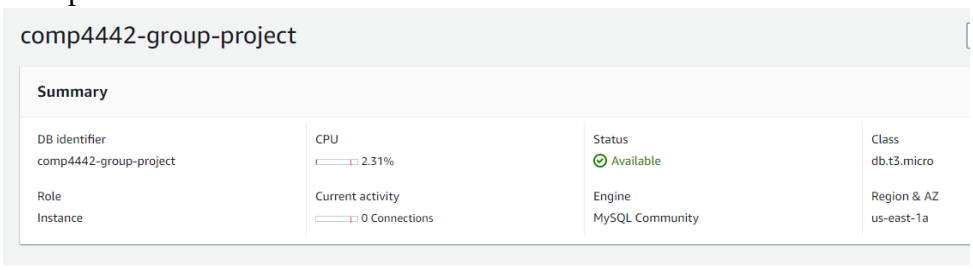
(hard code path, no args are needed)

5. Wait for complete

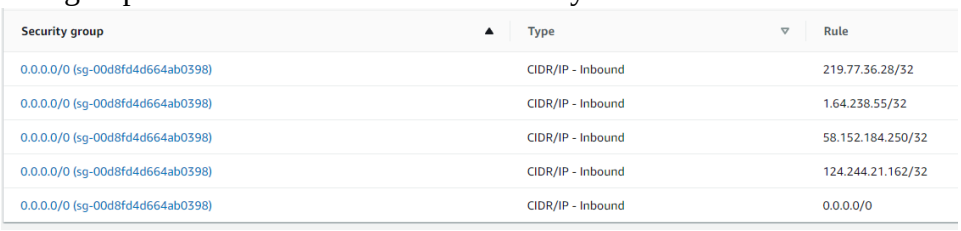
6. Find the csv in S3://comp4442-group-project/output/



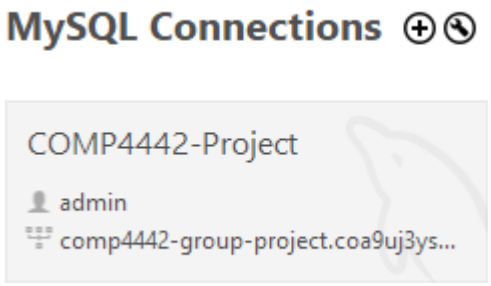
7. Download the csv and store in local
8. Set up AWS RDS database



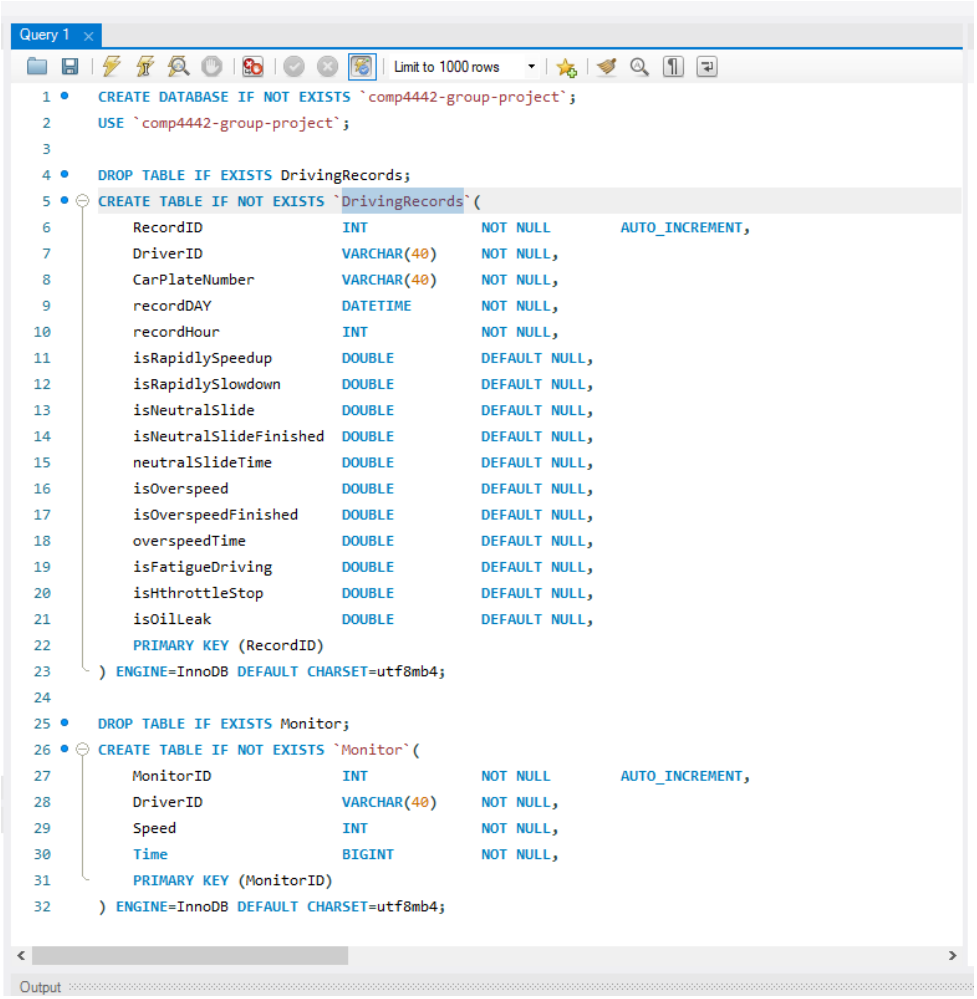
9. Add groupmate's IP and 0.0.0.0/0 to security Inbound rule



10. Setup MySQL workbench



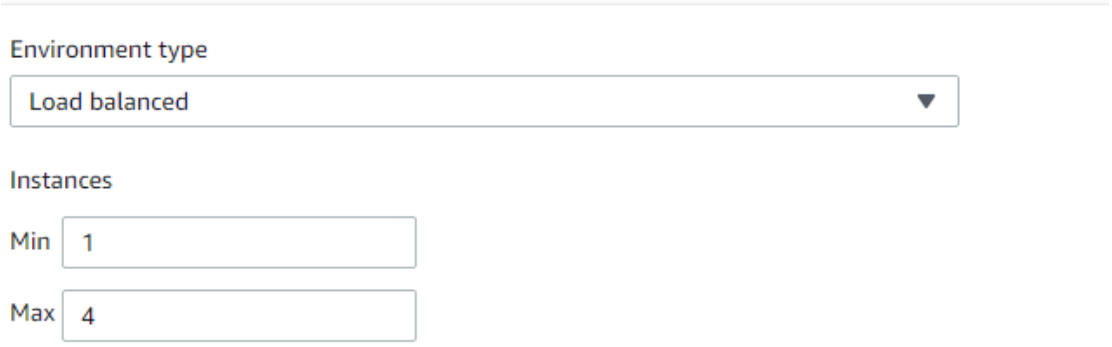
11. Copy the create table query in DB/initDB.sql and execute it to create DrivingRecords and Monitor tables.



12. Run upload_data_to_DB.py to upload the csv data to DB
    python upload_data_to_DB.py

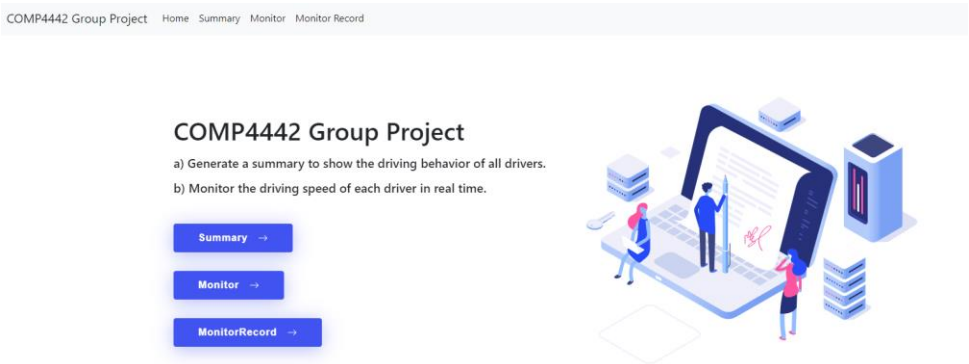
13. Develop the Flask application locally
    python application.py

14. Load balancer



15. Deploy the Flask app to AWS Elastic Beanstalk

url: http://web-env.eba-qttvhmqs.us-east-1.elasticbeanstalk.com/



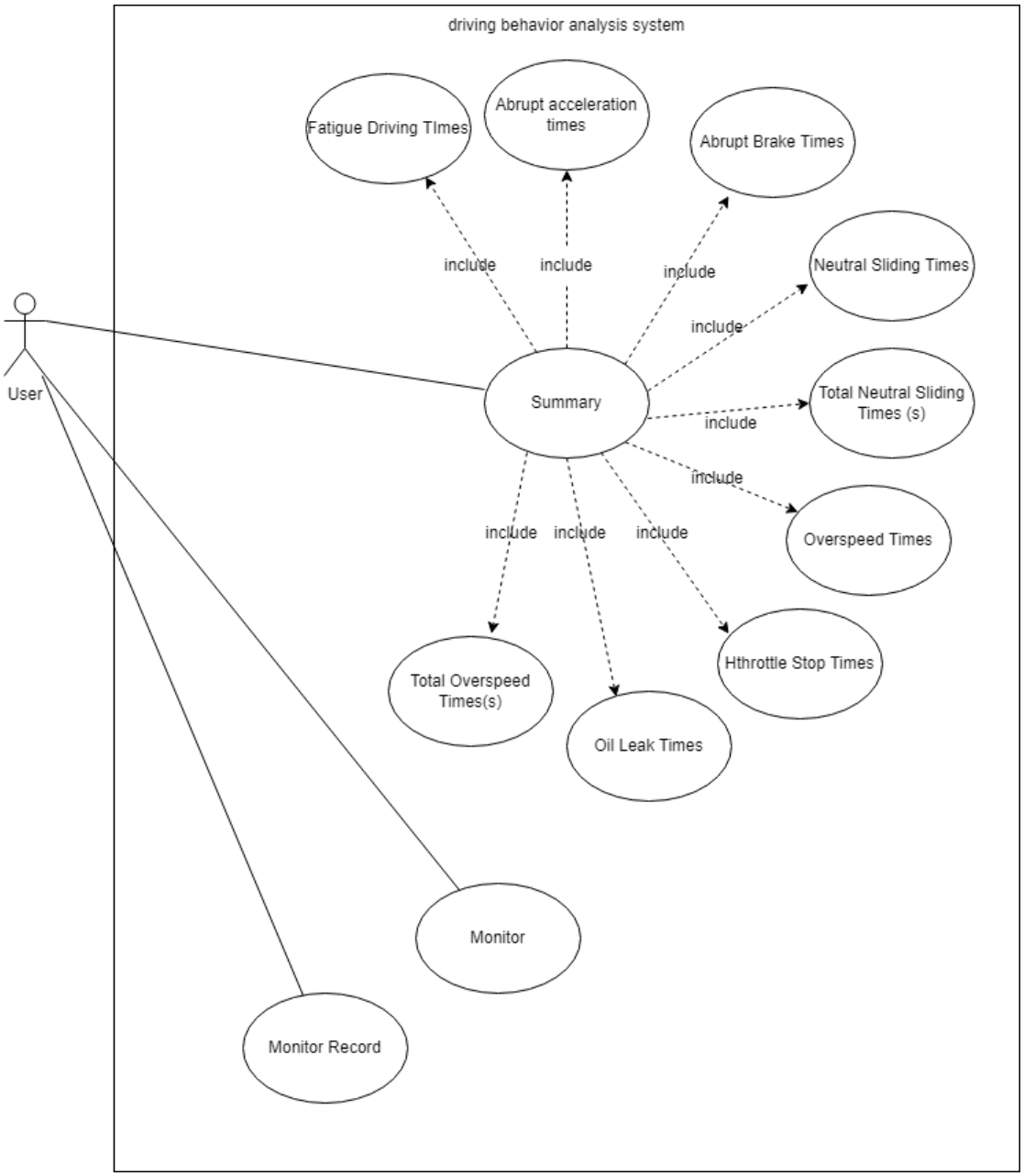
Scripts to be executed locally

- DB/ upload_data_to_DB.py
  Use once to upload the csv data to database

- DB/ gen_data_for_monitor.py
  Run this script to generate fake data for real time monitor function

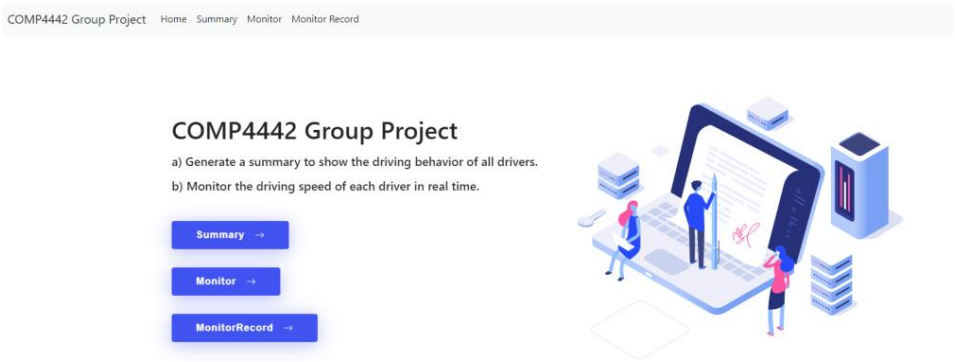
Testing reports for different functions

Use cases

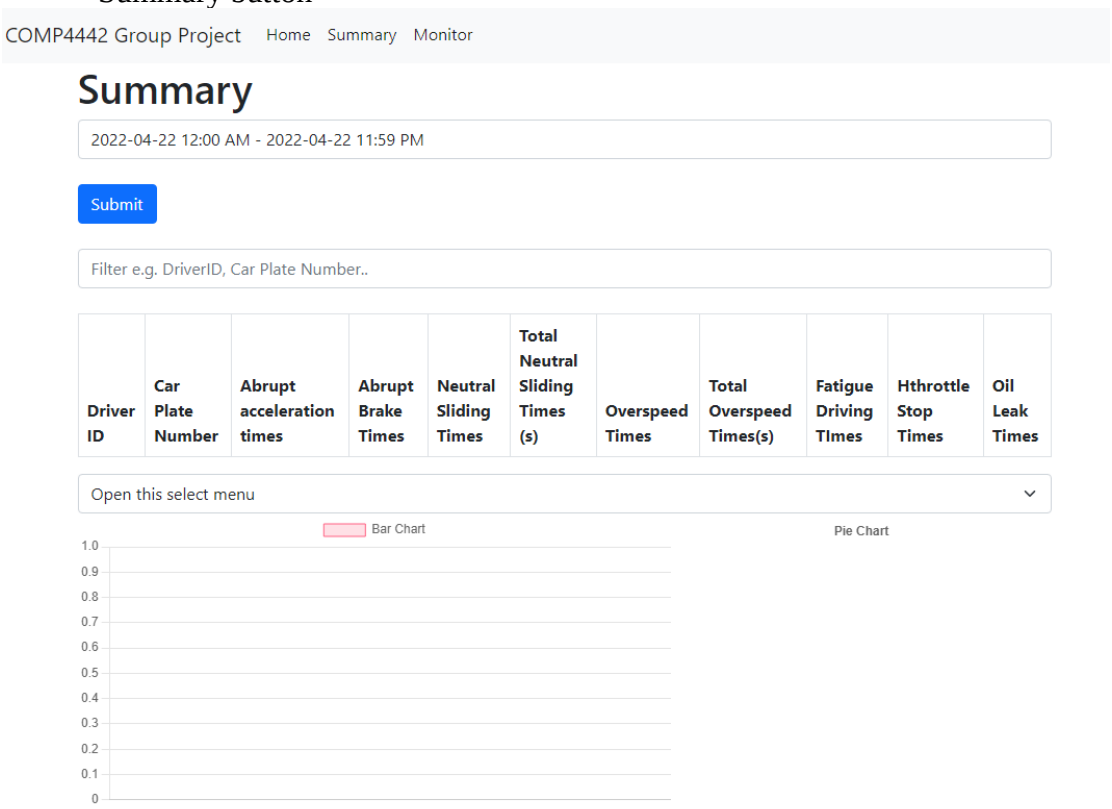


Screenshots of testing results

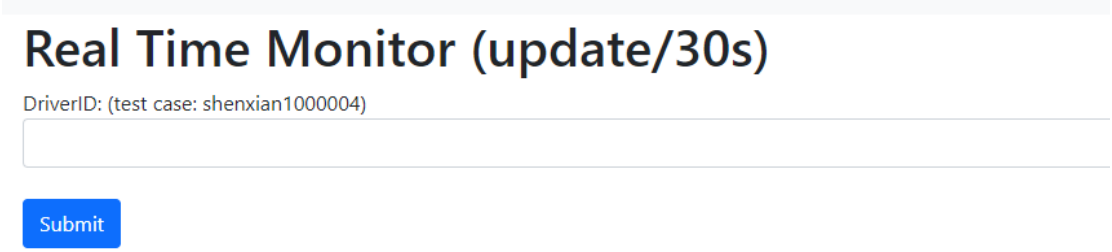
- Home Page



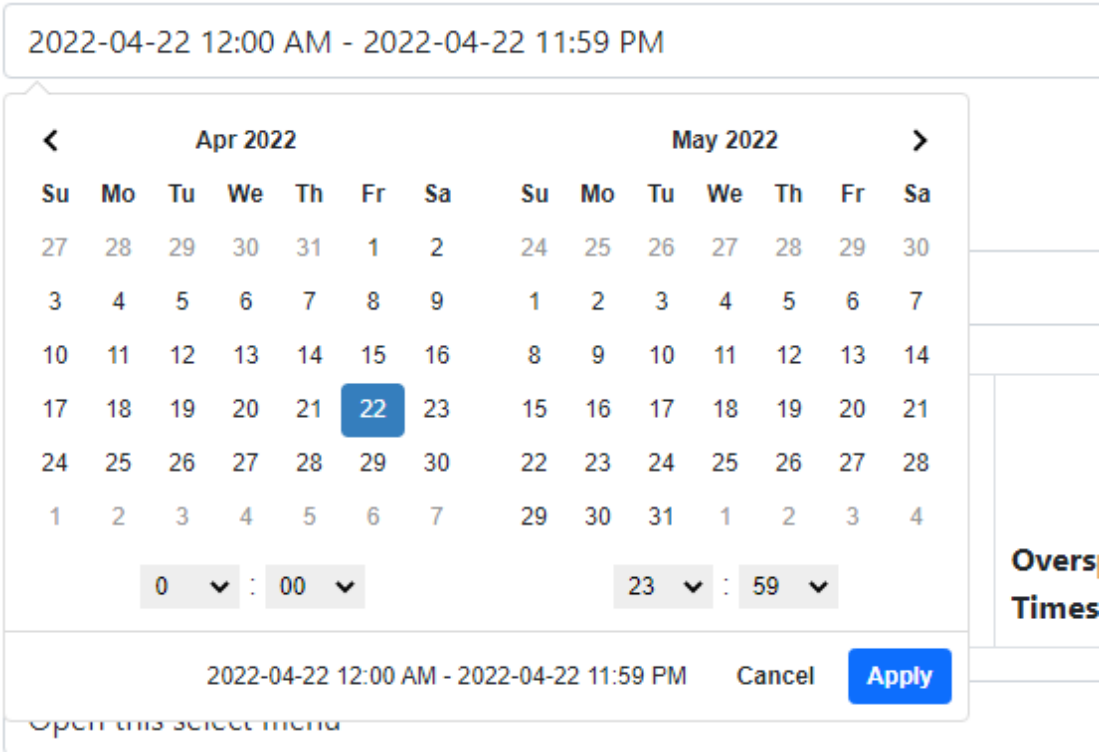
- Summary button



- Monitor button

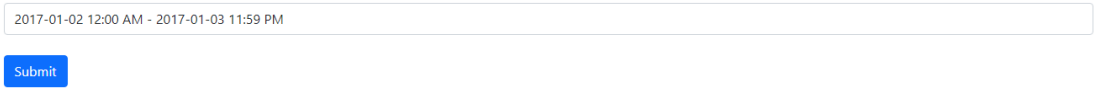


- Summary-Date and time button



- Summary-Choosing date and time

- Summary-Submit button



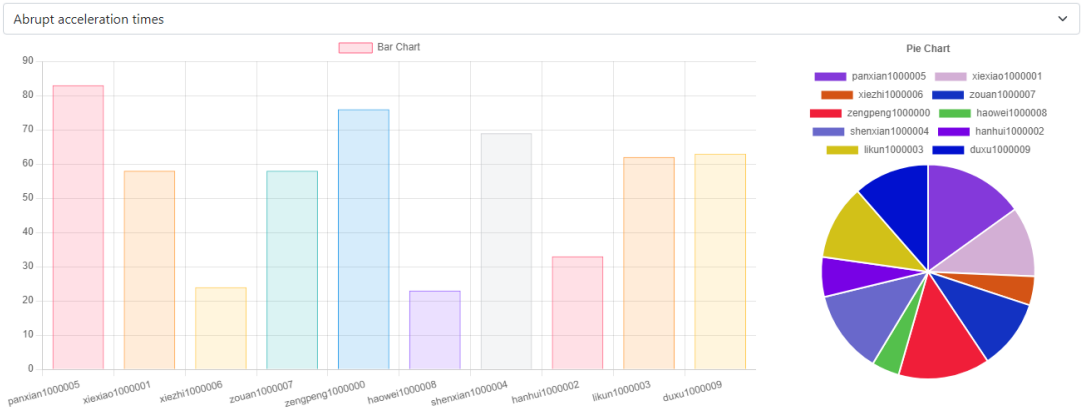
Filter e.g. DriverID, Car Plate Number..

Driver ID	Car Plate Number	Abrupt acceleration times	Abrupt Brake Times	Neutral Sliding Times	Total Neutral Sliding Times (s)	Overspeed Times	Total Overspeed Times(s)	Fatigue Driving TImes	Hthrottle Stop Times	Oil Leak Times
panxian1000005	甘AX542C	83.0	76.0	69.0	554.0	667.0	6378.0	824.0	85.0	84.0
xiexiao1000001	甘AEB132	58.0	53.0	57.0	580.0	493.0	4553.0	556.0	86.0	46.0
xiezhi1000006	甘A6CU11	24.0	40.0	28.0	255.0	251.0	2443.0	321.0	34.0	25.0
zouan1000007	甘A58M83	58.0	59.0	63.0	552.0	529.0	5155.0	561.0	64.0	60.0
zengpeng1000000	甘AZQ110	76.0	74.0	62.0	646.0	654.0	6320.0	754.0	75.0	82.0
haowei1000008	甘A709GB	23.0	19.0	16.0	150.0	155.0	1902.0	241.0	18.0	18.0
shenxian1000004	甘ADJ750	69.0	61.0	63.0	664.0	603.0	5618.0	708.0	63.0	75.0
hanhui1000002	甘AZI419	33.0	37.0	36.0	350.0	282.0	2714.0	354.0	37.0	27.0
likun1000003	甘AVM936	62.0	57.0	57.0	699.0	553.0	5274.0	649.0	45.0	70.0
duxu1000009	甘AT75H8	63.0	86.0	70.0	758.0	609.0	6112.0	787.0	64.0	60.0

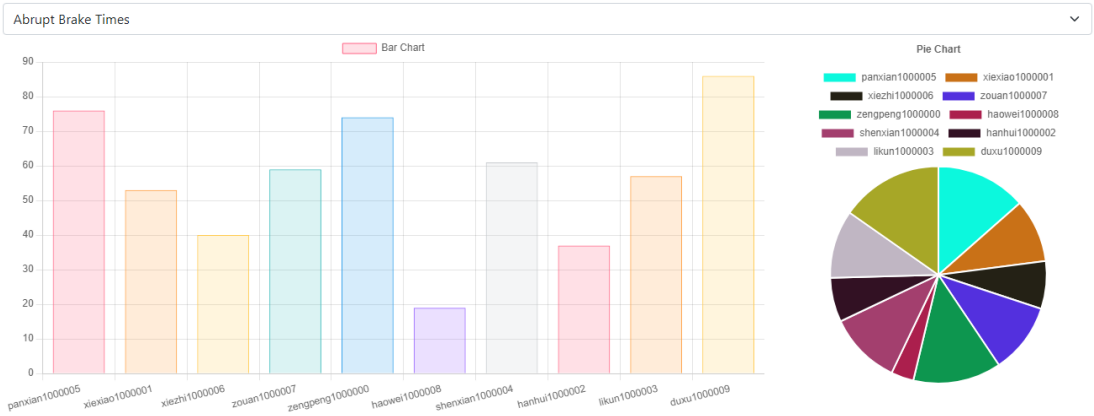
- Summary-Filter

Driver ID	Car Plate Number	Abrupt acceleration times	Abrupt Brake Times	Neutral Sliding Times	Total Neutral Sliding Times (s)	Overspeed Times	Total Overspeed Times(s)	Fatigue Driving TImes	Hthrottle Stop Times	Oil Leak Times
xiexiao1000001	甘AEB132	58.0	53.0	57.0	580.0	493.0	4553.0	556.0	86.0	46.0

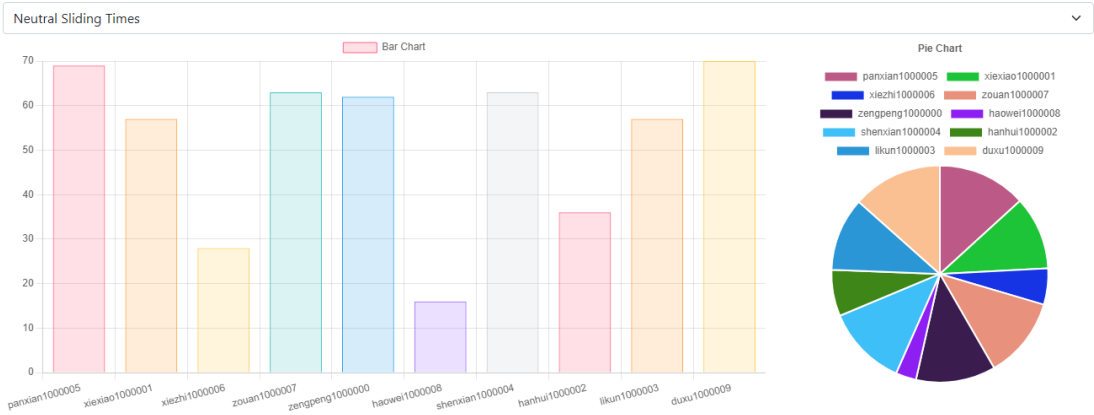
- Summary-Abrupt acceleration times



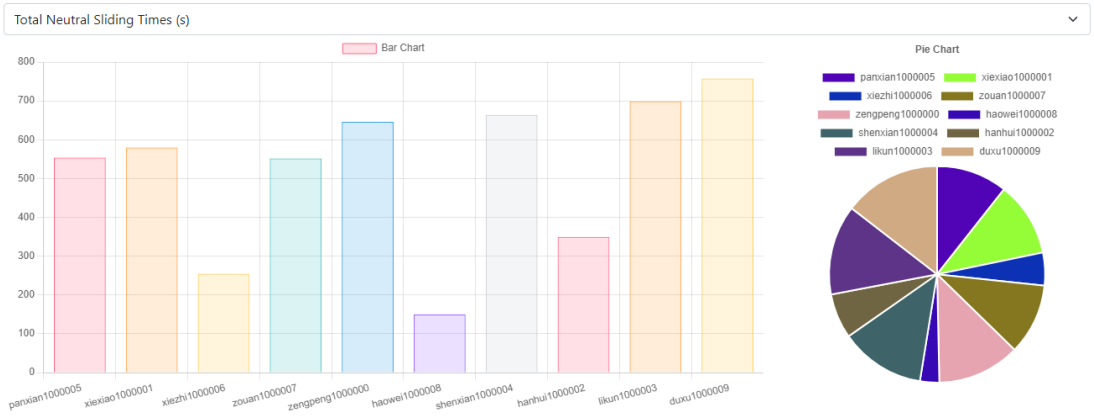
- Summary-Abrupt Brake Times



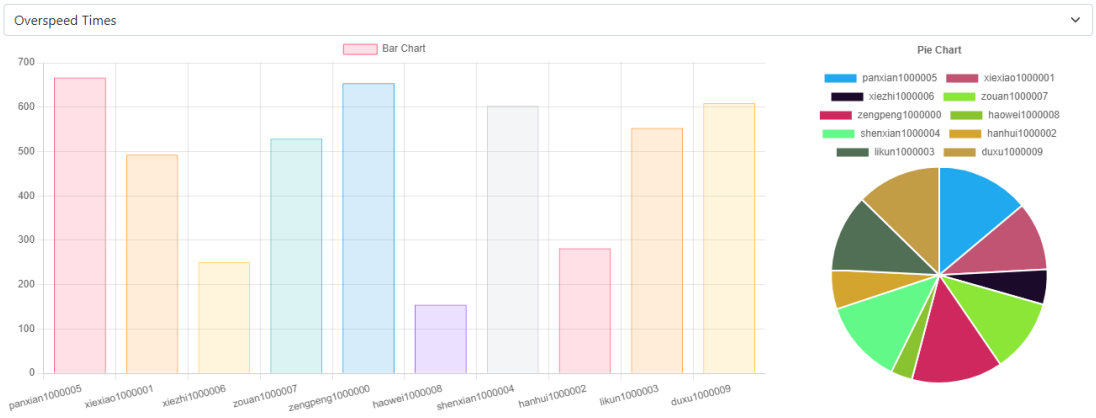
- Summary-Neutral Sliding Times



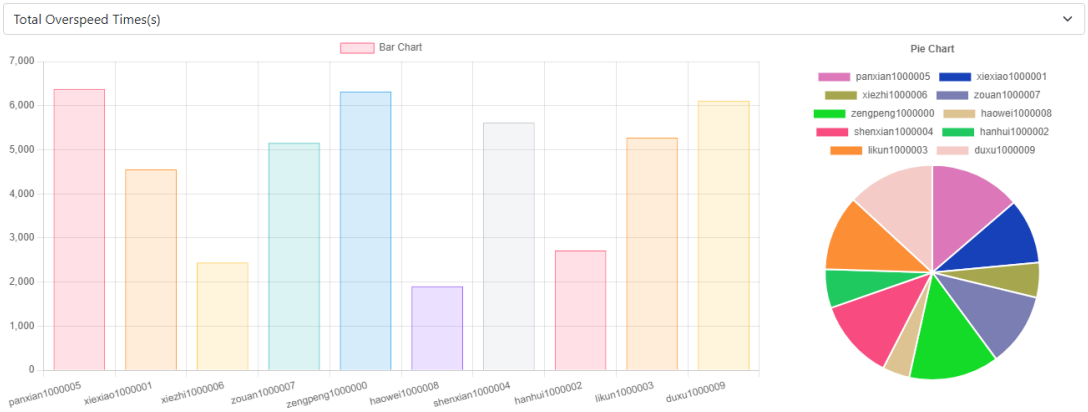
- Summary-Total Neutral Sliding Times (s)



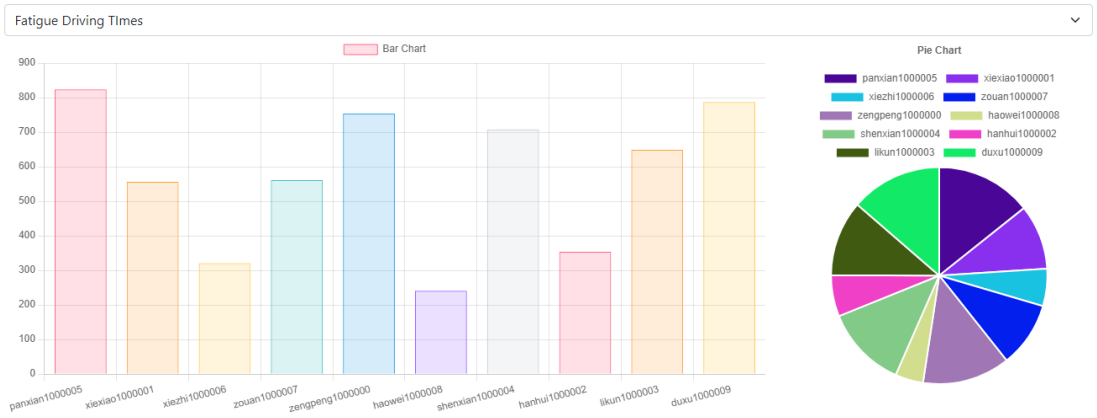
- Summary-Overspeed Times



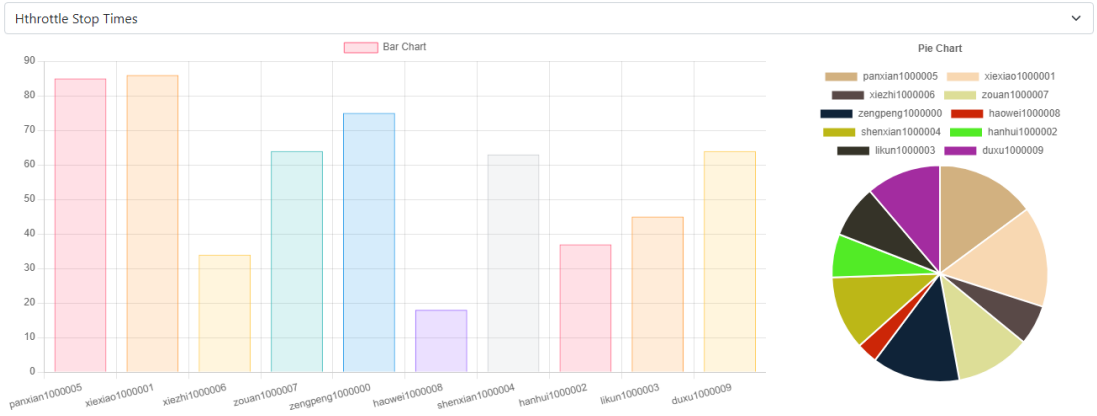
- Summary-Total Overspeed Times(s)



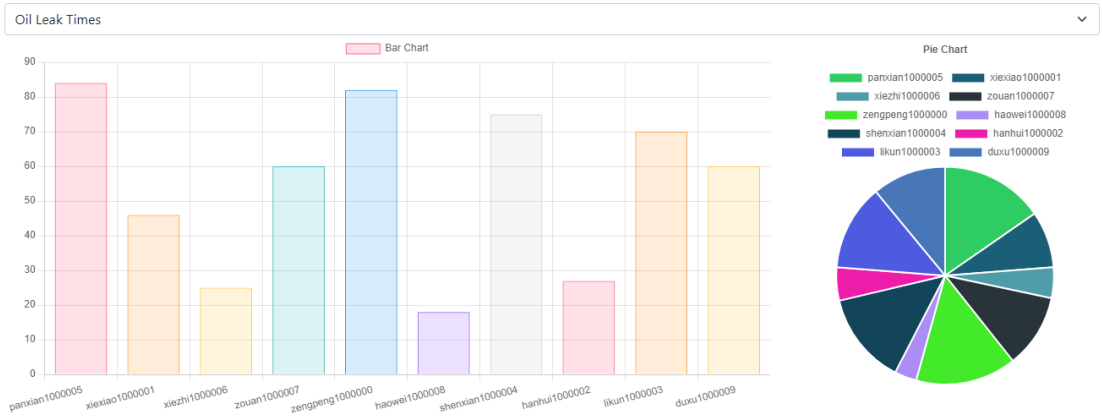
- Summary-Fatigue Driving Times



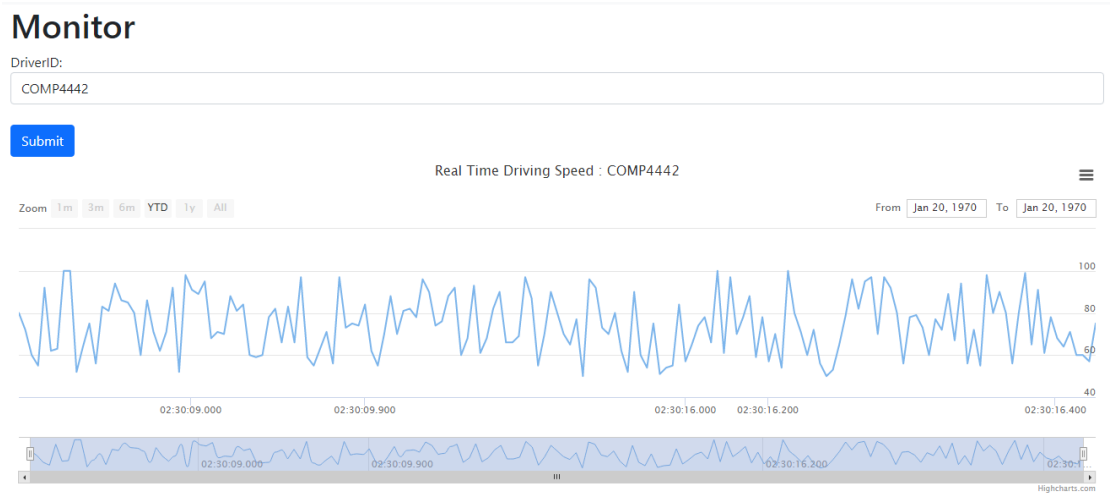
- Summary-Hthrottle Stop Times



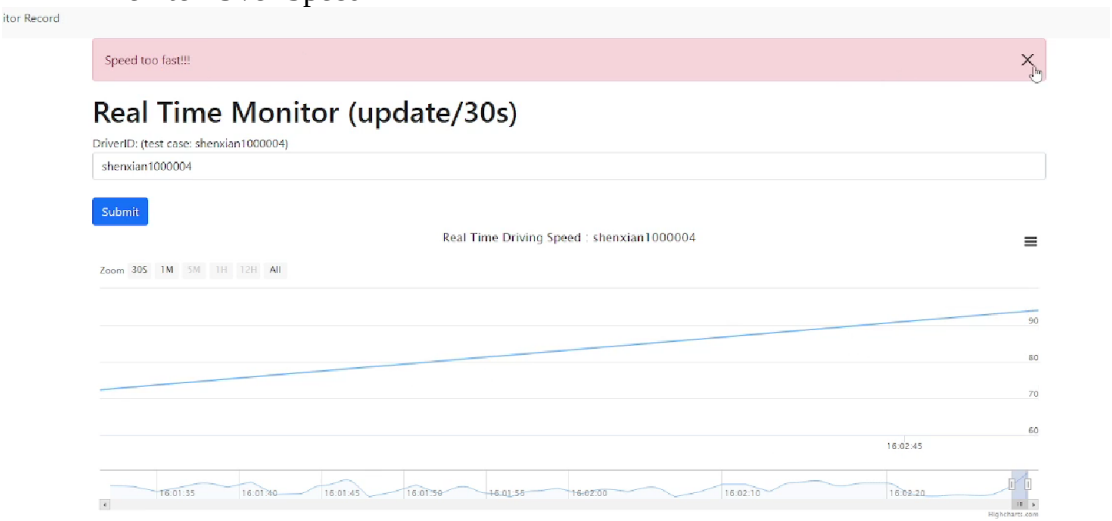
- Summary-Oil Leak Times



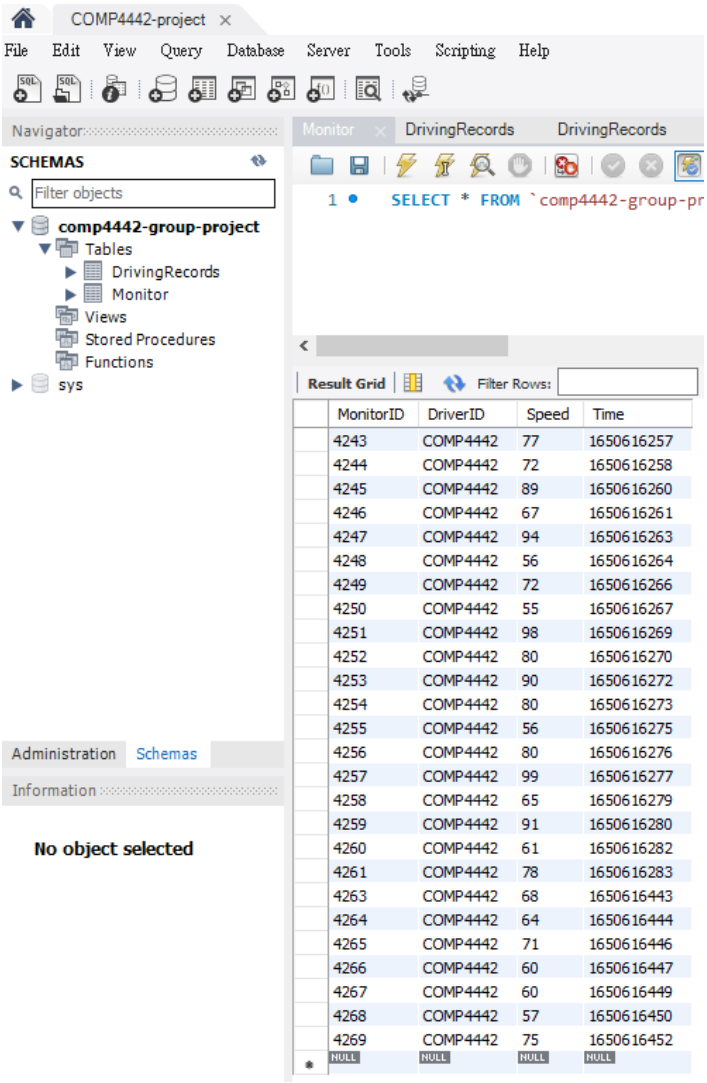
- Monitor-Submit



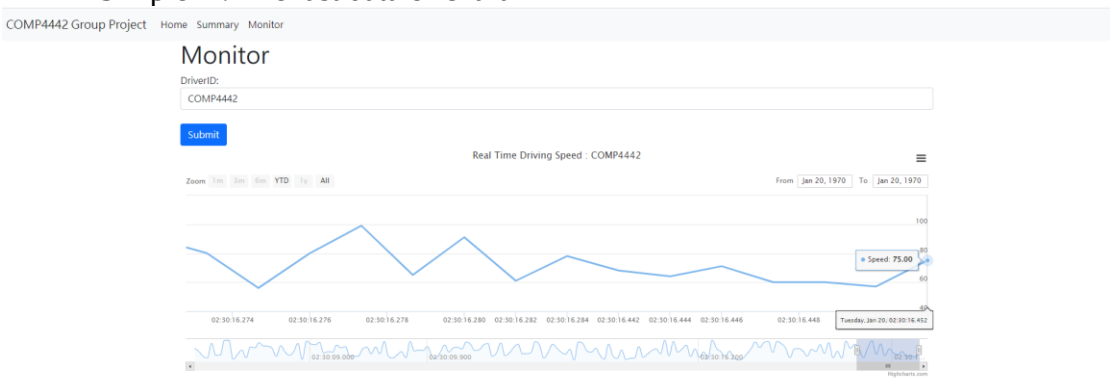
- Monitor-Over Speed



- The data of the chat should same with the database
- Data of database



- Simple 1 : The last data of chart

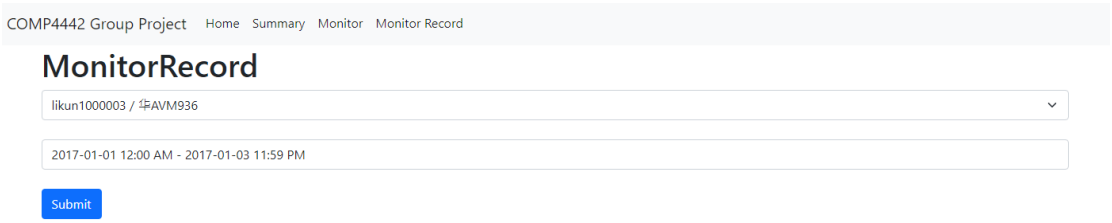


- Simple 2 : The second to last data of chart

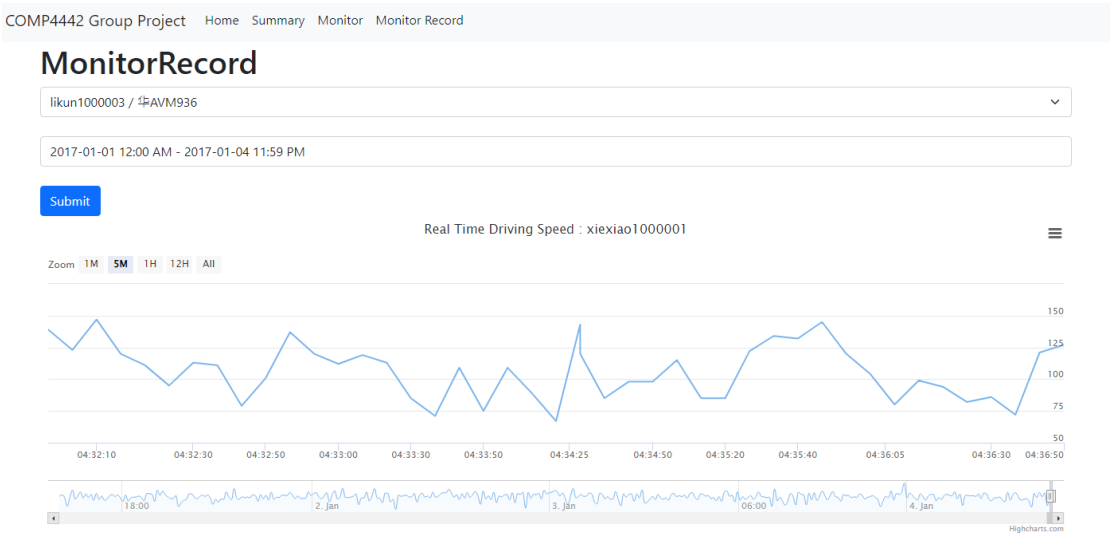
- Simple 3 : The third to last data of chart

- Result: all the data of the chat is same with the data base

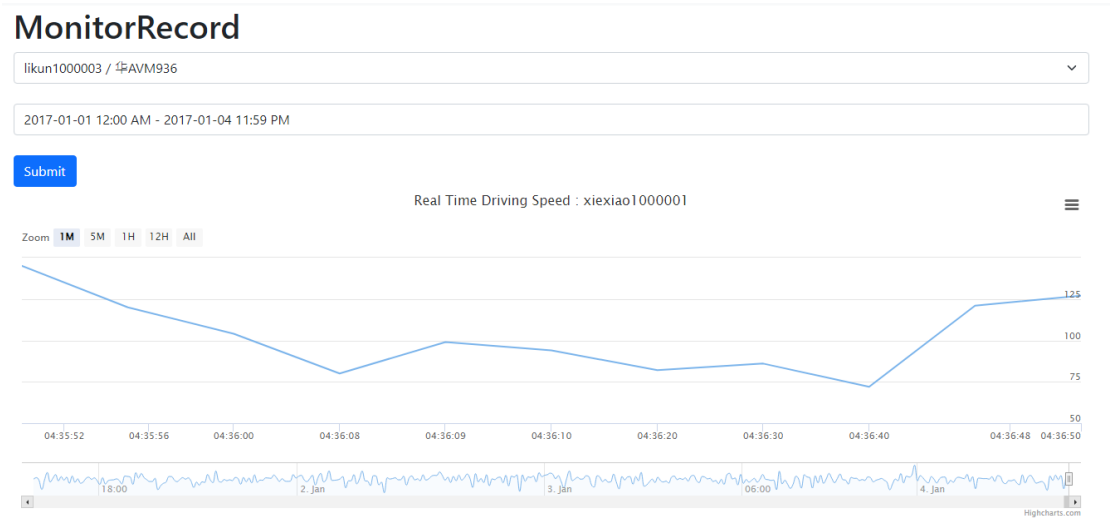
- Monitor Record



- 5M



- 1M



References:

[1] Haiwai, Driving Behavior Data Analysis, Apr 01, 2022, https://support.huaweicloud.com/intl/en-us/bestpractice-dli/dli_05_0001.html