2020 오픈소스전문프로젝트 11조

박수창 교수님

제출일: 2020/06/09

2014040003 윤민호  
2014040017 현승호  
2014040019 하태연  
2018076051 김주은  
2018076052 옥보라

스마트 버스 시스템

최종 결과 보고서

1. 서비스 개요---------------------------------------------------------------- 2
2. 사용센서 ------------------------------------------------------------------- 2
3. 시스템 아키텍쳐 (System Architecture) ---------------------------------- 3
4. 운영단계(Operation Steps) ------------------------------------------------ 4
5. 기대효과 ------------------------------------------------------------------- 5
6. 1차 데이터 수집 ---------------------------------------------------------- 6
7. 수집한 데이터에 대한 분석 계획 ---------------------------------------- 6
8. 1차 수집 데이터에 대한 분석 결과 ------------------------------------- 6
9. 피드백 후 새 데이터 분석 ---------------------------------------------- 12
10. 파이썬 Keras를 통한 데이터 학습 -------------------------------------- 14
11. 분석 데이터 이용 계획 -------------------------------------------------- 17
12. 어플리케이션 제작 진행 상황 ------------------------------------------ 18
13. 데이터베이스 제작 상황 ------------------------------------------------- 28
14. MLP학습을 통한 인공지능과의 연계 결과 ----------------------------- 30
15. 참고문헌 및 자료 -------------------------------------------------------- 35

**목차**

**1. 서비스 개요**

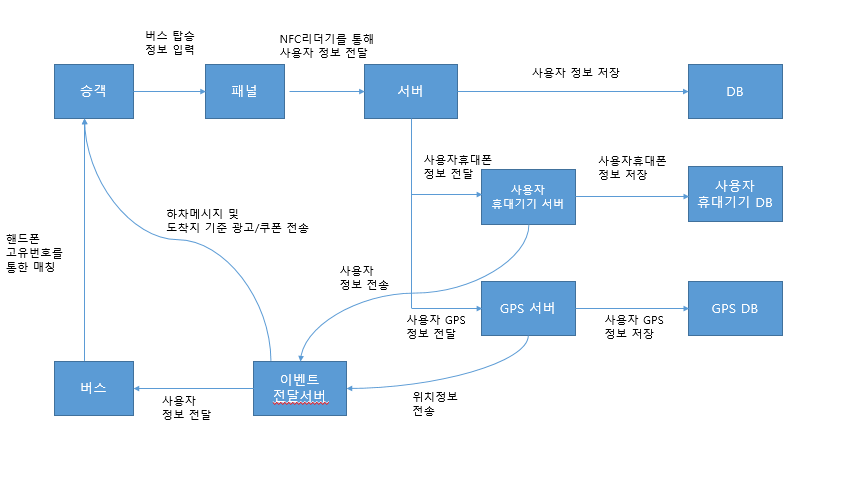
서비스 명 : 사전 결제 시스템을 활용한 스마트 버스 승하차 서비스

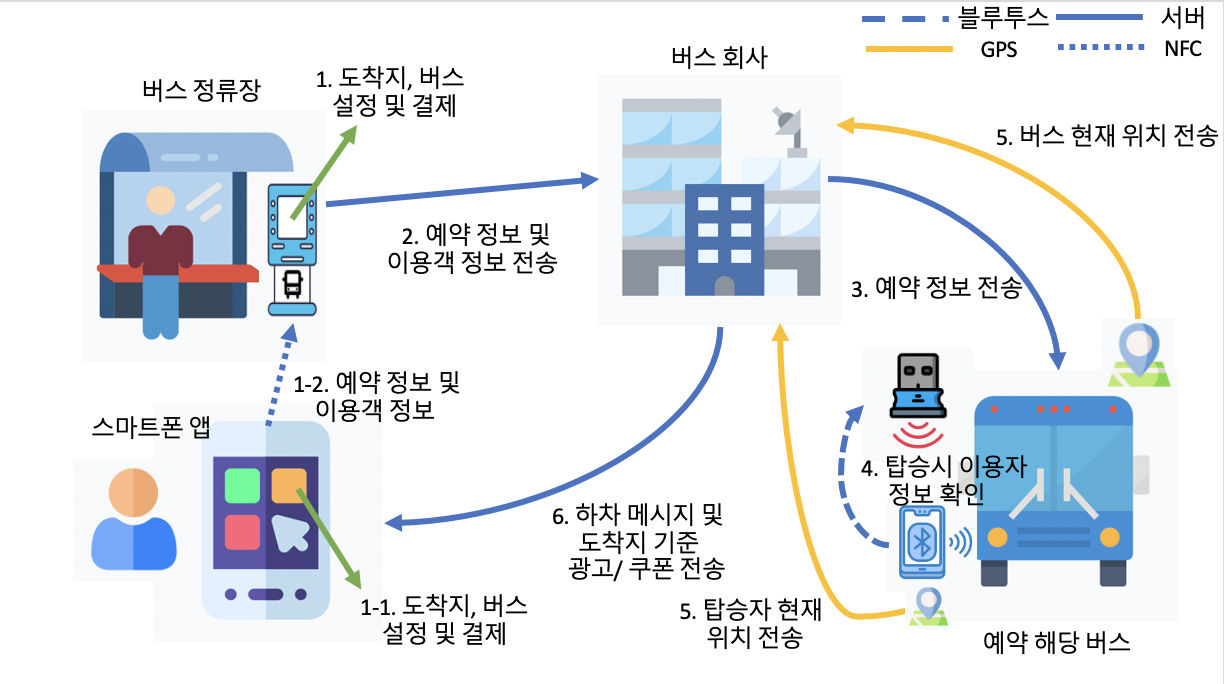
스마트폰에 다양한 센서들이 생기면서 어플의 기능도 그만큼 다양해지고 있다. 이러한 스마트폰의 장점을 이용하여 현재의 버스 이용 시스템을 통신채널을 기반으로 한 완전히 새로운 체제로 변경 하고자 한다. 서비스 이용자는 스마트폰 어플리케이션이나 버스 정류장의 패널을 통해 출발하고자 하는 정류장의 버스 정보를 확인한다. 이용자가 확인 할 수 있는 정보로는 버스의 도착 예정시간, 현재 위치, 버스의 혼잡도이다. 그 후 이용자는 탑승하고자 하는 버스를 선택한다. 그리고 어플이나 패널을 통해 교통비 결제를 진행한다. 이후 이용자가 서비스에 대해 취소나 변경을 원할 시 패널이나 어플에서 언제든 가능하다. 버스에 탑승할 때에 버스에 있는 블루투스 인증 단말기를 통해 탑승자가 결제를 끝낸 사람인지 확인을 하여 버스 기사는 무임승차자를 구별 할 수 있다. 탑승중인 버스는 GPS 데이터를 통해 구별이 가능하다. 이용객이 하차할 시에는 하차 메시지 및 도착지 근처의 광고/쿠폰 등이 송신되며 별도의 하차 태그 없이 하차 가능하며 하차 후 승,하차정보는 버스회사 DB에 저장되어 추후 정류장 개선 정보로 이용 가능하다.

**2. 사용센서**

1. NFC: 'Near Field Communication'의 약자로 가까운 거리에서 무선 데이터를 주고받는 통신 기술을 의미하며, NFC 기능이 있는 스마트폰을 기기에 접촉하기만 하면 자동으로 블루투스 페어링이 된다. 복잡한 설정 과정 없이 쉽고 빠르게 스마트폰과 블루투스 기기를 연결한다. NFC 접촉시 스마트폰의 NFC 기능이 켜져 있어야 하고, 잠금화면이 해제되어 있어야 한다. NFC 접촉시 스마트폰 연결을 위한 어플이 자동으로 실행될 수 있다.
2. 블루투스: 블루투스(Bluetooth)는 휴대폰, 노트북, 이어폰·헤드폰 등의 휴대기기를 서로 연결해 정보를 교환하는 근거리 무선 기술 표준을 뜻한다. 주로 10미터 안팎의 초단거리에서 저전력 무선 연결이 필요할 때 쓰인다.
3. GPS: 위성에서 보내는 신호를 수신해 사용자의 현재 위치를 계산하는 위성항법시스템이다. 항공기, 선박, 자동차 등의 내비게이션장치에 주로 쓰이고 있으며, 최근에는 스마트폰, 태블릿 PC등에서도 많이 활용되는 추세다. 기기의 위치를 가장 정확하게 특정 지을 수 있는, 사실상 현대의 유일무이한 위치 결정 시스템이다.

**3. 시스템 아키텍쳐 (System Architecture)**





**4. 운영단계 (Operation Steps)**

이 Operation Steps은 사용자가 어플 사용자인 것을 전제로 한다.

**Operation Steps 1. 정보 기입**

* 1. 어플 사용자의 경우 어플을 통해 출발 정류장, 도착 정류장, 탑승할 버스번호를 사전에 어플을 통해 기입한다.
  2. 어플 비사용자의 경우 버스정류장에 설치된 패널을 통해 도착 정류장, 탑승할 버스번호를 패널에 정보를 기입한다.
  3. 초행길이라 도착 정류장을 모르는 경우 주변 지역 혹은 건물을 적을 시 가장 가까운 정류장이 패널에 뜨게 된다.

**Operation Steps 2. 결제**

* 1. 어플 사용자의 경우 어플을 통해 결제하거나 패널을 통해 결제한다.
  2. 어플 비사용자의 경우 버스 정류장에 설치된 패널을 통해서만 결제가 가능하다.

**Operation Steps 3. 패널을 통한 정보 전송**

결제가 완료된 후에는 패널에 부착된 NFC 리더기에 사용자의 핸드폰을 가져다 댐으로써 출발정류장, 도착정류장, 사용자의 GPS정보, 탑승한 버스번호, 핸드폰의 고유번호를 버스회사 서버로 전송한다. 전송된 정보는 버스회사의 DB에 저장된다.

**Operation Steps 4. 버스회사에서 버스로 정보 전송**

버스회사에서는 전송된 정보 중 핸드폰의 고유번호, 결제자의 수를 해당 정류장 기준 가장 가까운 선택된 번호의 버스에게 전송한다. 이 때 핸드폰의 고유 번호는 결제확인의 수단으로 전송되며 결제자의 수는 버스가 태울 사람이 없는 정류장에 의미없이 정차하는 것을 막기 위해 전송된다..

**Operation Steps 5. 서비스 변경 및 취소**

5-1. 사용자가 서비스 변경을 원하는 경우 패널을 통해 탑승할 버스의 번호를

변경 가능하다.

5-2. 사용자가 서비스 취소를 원하는 경우 패널을 통해 취소가 가능하며, 취소없이 정류장을 벗어나거나 탑승하지 않을 시에는 버스의 GPS 정보와 사용자의 GPS정보를 대조해 탑승하지 않았을 시에는 변경 및 취소 알람을 전송. 알람 전송 후 일정시간 대답이 없을 시에는 취소로 간주하고 결제한 금액을 환불한다.

**Operation Steps 6. 버스 탑승**

버스가 도착하고 사용자가 버스에 탑승하려 할 시 사전에 전송된 사용자 핸드폰의 고유번호를 기준으로 버스 입구에서 블루투스 인증 단말기를 통해 대조 후 이상 없을 시 탑승한다. 탑승 후 탑승중인 버스는 GPS 데이터를 통해 구별이 가능하다.

**Operation Steps 7. 버스 하차 메시지 및 광고/쿠폰 전송**

사용자의 GPS정보와 도착정류장의 GPS정보가 가까워지면 버스회사에서 하차 메시지 및 도착지 근처의 광고/쿠폰 등을 사용자의 핸드폰에 송신한다.

**Operation Steps 8. 버스 하차**

버스가 정류장에 도착하면 사용자는 별도의 하차 태그에 태그 없이 하차 후 GPS를 이용해 버스와 사용자의 위치가 멀어지면 하차한 것으로 간주하고 사용자의 GPS 수집을 중단한다.

**5. 기대효과**

1. 버스 탑승의 간소화: 정류장에서 결제를 미리 하여 버스 탑승에 필요한 시간을 줄인다. 이로 인해 버스가 도착하면 이용객들은 버스에 탑승하기만 하면 되므로 버스 정류장의 복잡함도 줄일 수 있다.
2. 불필요한 정차가 없음: 버스 정류장에 해당 버스의 추가 탑승자가 있는지 미리 파악이 가능하므로 승,하차를 하려는 사람이 없는 경우에는 불필요하게 정차를 함으로 인한 교통방해를 줄일 수 있다.
3. 승,하차 데이터 누적: 하차태그를 찍지 않아도 자동적으로 승객의 자세한 승,하차 정보를 판단하고 데이터로 보관하여 새로운 노선이나 정류장을 건설 할 때 참고 자료로 가용 가능하다.
4. 하차 알림: 하차 예정인 정류장에 가까워지면 알림을 받으므로 잘못 내리거나 내리지 못하는 경우를 줄일 수 있다.
5. 초행길 도우미: 처음 가는 지역에 가더라도 정확한 정류장에서 승,하차를 할 수 있으므로 쉽게 길을 찾을 수 있다.

**6. 1차 데이터 수집**

데이터 수집은 제작하고자 하는어플의 특수성을 고려하여 새로운 데이터를 모으기 위해서는 어플 제작 및 서비스 제공부터 수집이 가능하기에 기존 버스 시스템을 통해 모인 공공데이터를 이용하였다. 사용 한 공공데이터는 서울시에서 제공하는 서울시의 모든 버스 정류장 및 버스들의 시간대별 승하차 인원에 대한 공공 데이터이다.

- **공공데이터 파일명:** 2020년\_버스노선별\_정류장별\_시간대별\_승하차\_인원\_정보(04월)

**- 서울특별시 제공 서울시 버스노선별 정류장별 시간대별 승하차 인원 정보 공공데이터 링크:**

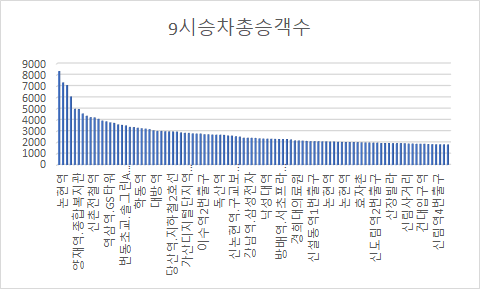
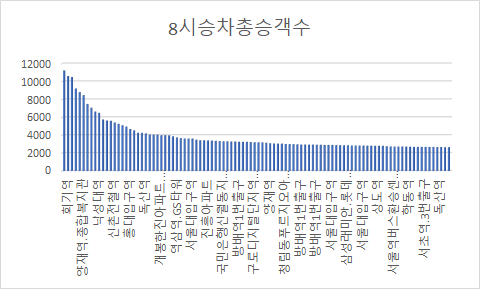
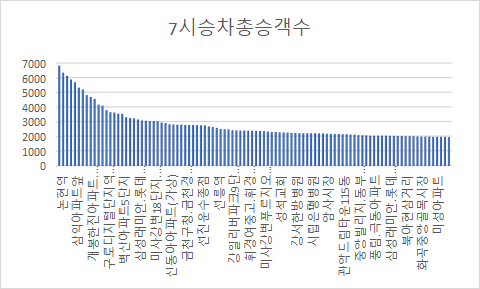
<https://data.seoul.go.kr/dataList/OA-12913/S/1/datasetView.do>

**7. 수집한 데이터에 대한 분석 계획**

데이터에 대한 분석은 크게 두가지로 나누어진다. 첫째는 서비스 이용자(대표적으로 버스 이용 시민)에게 맞춤 서비스를 제공하기 위한 데이터 분석이며 두번째는 개발 된 시스템을 사용할 버스회사에게 제공될 데이터 분석이다.

**8. 1차 수집 데이터에 대한 분석 결과**

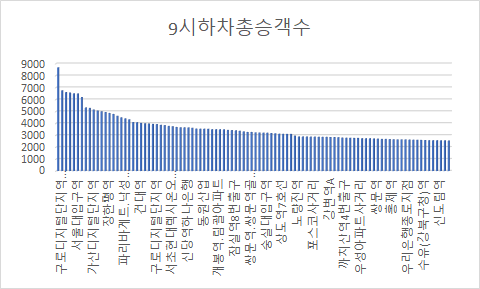
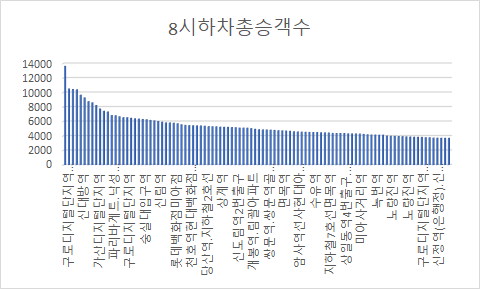
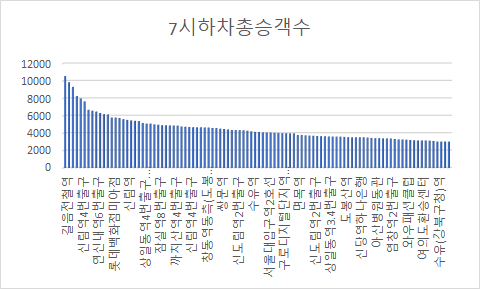
1. 오전 7시 – 오전 9시 (승차)

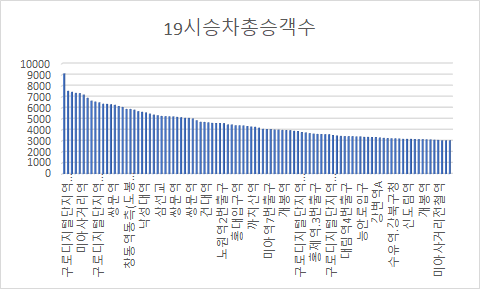
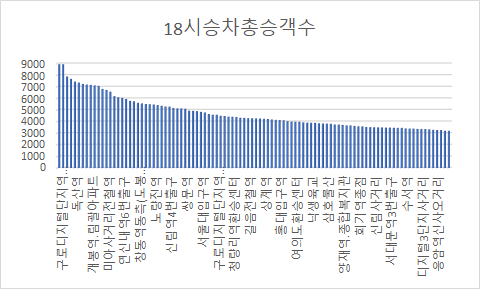


* 1. 직장과 학교로 가기 위해 탑승
     + - 이 때 높은 이용률을 기록하는 정류장들은 사람들의 거주지일 확률이 높음
  2. 활용 방안
     + - 인근 지역에 생활편의시설을 확충하여 주민들의 삶의 질을 향상시킬 수 있다.
       - 주변 사업체와의 제휴를 통해 서비스 이용자에게 부가서비스(할인) 제공

Ex) 마트, 헬스장 등

1. 오전 7시 – 오전 9시 (하차), 오후 6시 – 오후 7시 (승차)

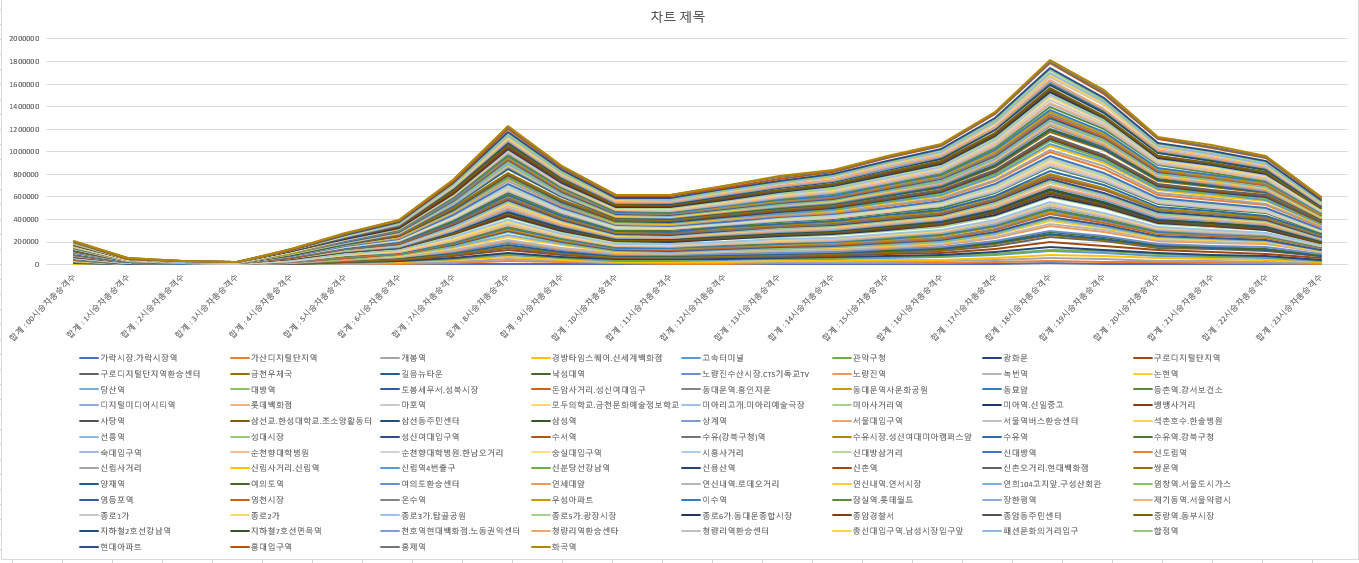




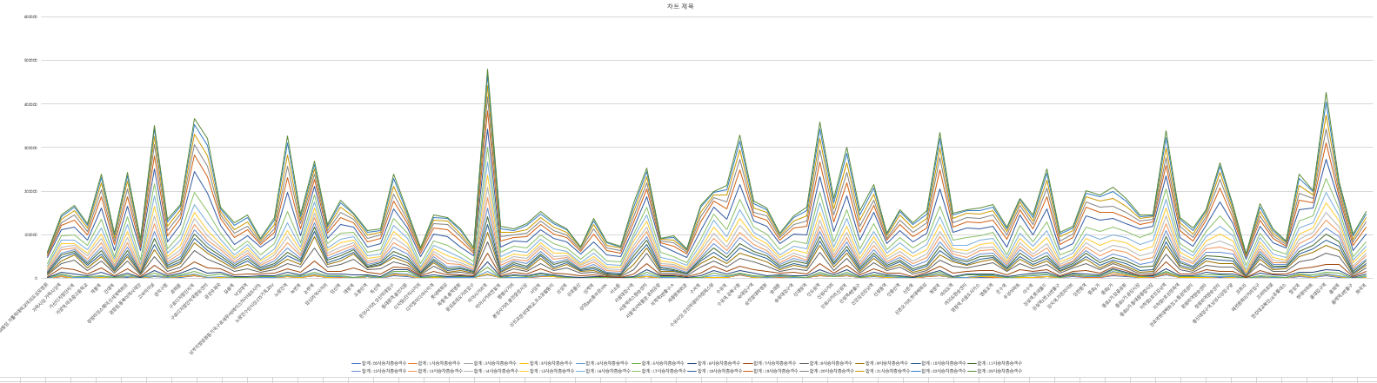
* 1. 하루 중 가장 높은 이용률을 보이는 시간대 (평균 9천 - 1만명)
     + - 유동인구가 많은 지역(사업체 밀집 / 번화가 / 학교)
  2. 활용 방안
     + - 경제적 가치가 높은 지역으로 향후 사업체 투자 유치나 상권 부흥에 활용
       - 주변 사업체와의 제휴를 통해 서비스 이용자에게 부가서비스(할인) 제공

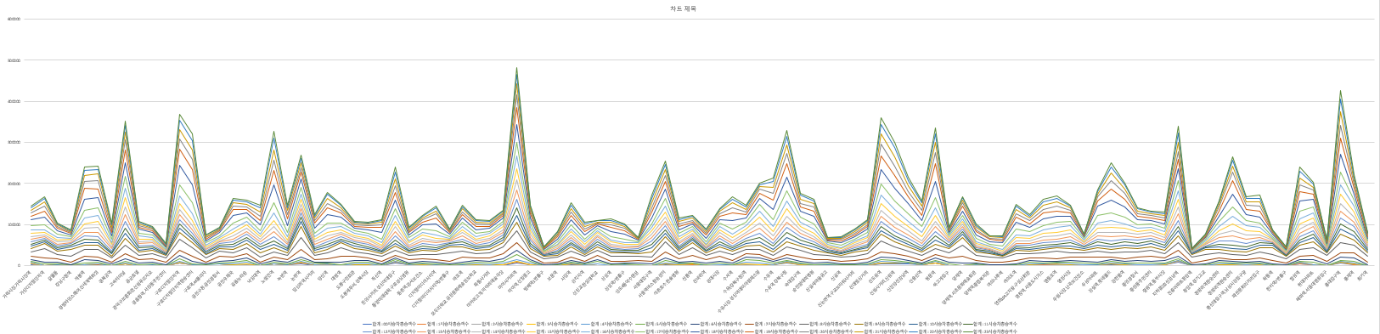
Ex) 식당, 의류판매점, 카페 등

1. 승하차율이 높은 지역
   * + - 노선을 증설하여 교통혼잡을 줄이는 효과
2. 승하차율이 저조한 지역
   * + - 거주 인구 또는 유동인구가 적은 지역으로 이용률에 비해 많은 노선이 배치된 경우 배차간격을 늘리고 노선 수를 줄여 비용을 절감하는 효과



1. 각 정류장들의 시간대 별 승객수를 분석
   1. 8시와 18시에 승객수가 가장 많음
   2. 출퇴근, 등하교 시간으로 유동 인구가 가장 많음





1. 8시와 18시의 정류장별 승객수를 다시 분석
2. 평균적으로 승객수가 많은 곳과 적은 곳을 선별
   1. 주로 \_\_\_\_역이 수치가 높음
      * + 지하철역 주변으로 유동인구가 많고 상가가 밀집해 있음

=> 주변 상가들의 광고창(맛집 추천 서비스) 운영 및 할인 쿠폰 제공 등의 서비스를 지원할 수 있음

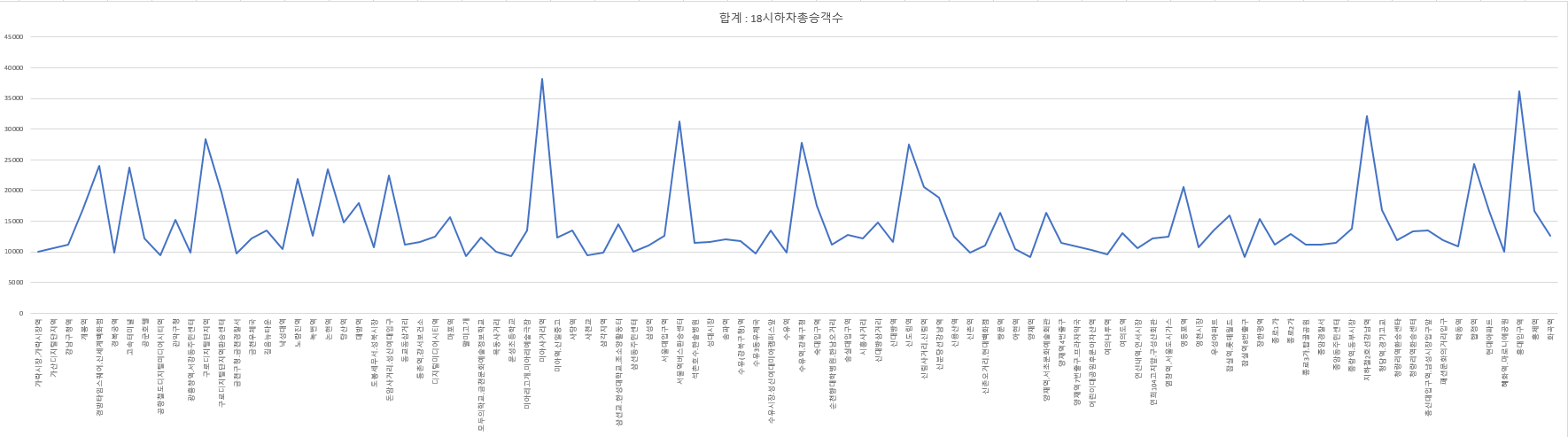
* + - * 환승이 편리함

=> 환승역까지의 길찾기를 지원할 수 있음

* 1. 추후에 노선 변경, 버스 수 조정에 필요한 데이터 사용 가능



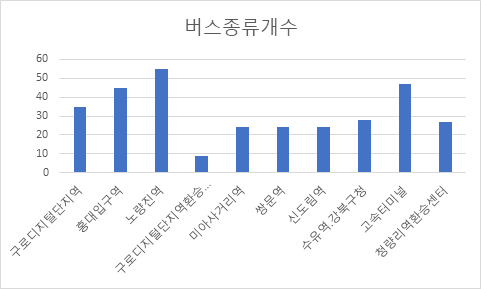
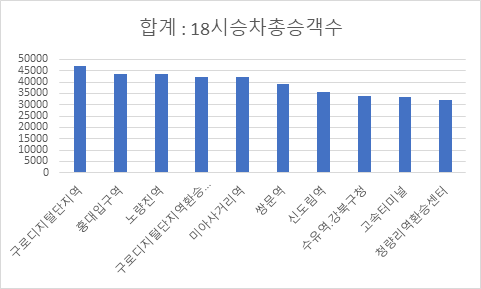
-8시 승차인원이 상위100인 버스정류장-



-18시 하차인원이 상위100인 버스정류장-

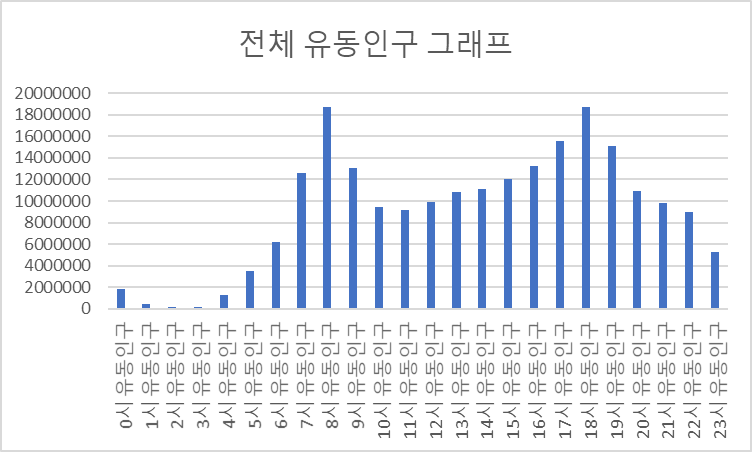
위의 그래프를 기준으로 분석한 결과

1. 8시 승차그래프의 값과 18시 하차 그래프의 값이 비슷한 경우 해당 정류장은 주거지역 근처로 분석 할 수 있다.
2. 또한 8시의 승차그래프의 인원보다 18시 하차그래프의 인원이 더 높은 버스 정류장은 상권이 발달한 지역으로 분석할 수 있다.
   1. 위의 분석한 결과들을 토대로 주거지역과 상권이 발달한 지역을 산정할 수 있는데 이 데이터들을 가지고 주거지역 근처의 편의시설들과 상권이 발달한 지역 근처의 음식점들과 제휴를 맺어 버스를 자주 이용하는 승객들에 한해 각 제휴 업체들에 대한 할인 쿠폰 등의 서비스를 제공할 수 있다.
3. 시내 버스 운행에 있어서 사람이 많이 탑승하는 시간에 버스가 상대적으로 적게오면 삶의 질이 떨어지므로 가장 유동인구가 많은 오후 6시의 정류장 승차 인구와 각 정류장으로 오는 버스를 조사함.



다음 분석은 위의 그래프를 토대로 분석한 내용이다.

1. 18시 승차
   1. 구로디지털단지역 47219명 승차 버스종류 35개
   2. 홍대입구역 43587명 승차 버스종류 45개
   3. 노량진역 43450명 승차 버스 종류 55개
   4. **구로디지털단지역환승센터 42425명 승차 버스종류 9개**
   5. 미아사거리역 42388명 승차 버스종류 24개
   6. 쌍문역 39237명 승차 버스종류 24개
   7. 신도림역 35436명 승차 버스종류 24개
   8. 수유역.강북구청 33671명 승차 버스종류 28개
   9. 고속터미널 333232명 승차 버스종류 47개
   10. 청량리역환승센터 32112명 승차 버스종류 27개
2. 유동인구가 가장 많은 오후 6시에 대체적으로 27개~55개의 종류의 버스가 오는데 구로디지털단지역환승센터만 오직 버스 종류가 9개고 승차인구는 42425명으로 많다. 실조사 후에 운행 경로 수정이 필요한지 확인해야 좋을 것 같다.



1. 위의 그래프를 보면 하루 유동인구 중 8시와 18시가 제일 많다. 그 중에서도 앞뒤 시간과 차이가 많이 나는 8시를 기준으로 분석을 우선 진행하고자 한다.
2. 8시 기준 유동인구 제일 많은 5개의 정거장의 유동인구수와 배치 버스 수
   1. 구로디지털단지역 85434명 35개
   2. 신도림역 66803명 51개
   3. 노량진역 65731명 55개
   4. 구로디지털단지역환승센터 65485명 9개
   5. 미아사거리역 59913명 52개
3. 구로디지털단지환승센터 정류장의 경우 유동인구는 많으나 그에 비해 버스의 수가 부족하므로 해당하는 노선의 해당 시간대 버스 배차 간격을 좁게 하거나 다양한 버스의 배치가 필요하다 분석.
4. 또한 구로디지털단지역 정류장의 경우도 다른 정류장에 비해 월등히 높은 유동인구 수치를 보이나 상대적으로 버스의 수가 부족하므로 해당 노선의 해당 시간대 버스의 배차 간격을 좁게 하거나 추가적인 버스 노선의 배치가 필요하다 분석.

**9. 피드백 후 새 데이터 분석**

피드백 후 주요적으로 사용할 데이터를 버스 승하차 데이터가 아닌 스마트폰의 가속도 센서를 이용하여 수집한 데이터로 변경하였다. 이는 사용자가 버스를 예약 하였을 때 제시간에 버스 정류장까지 도착 할 수 있는가에 대한 분석을 하기 위함이다. 사용자의 상태를 크게 3가지 (걷기, 뛰기, 정지)로 나누었으며 이에 대한 데이터를 다양한 방식으로 수집하여 분석을 진행하였다.

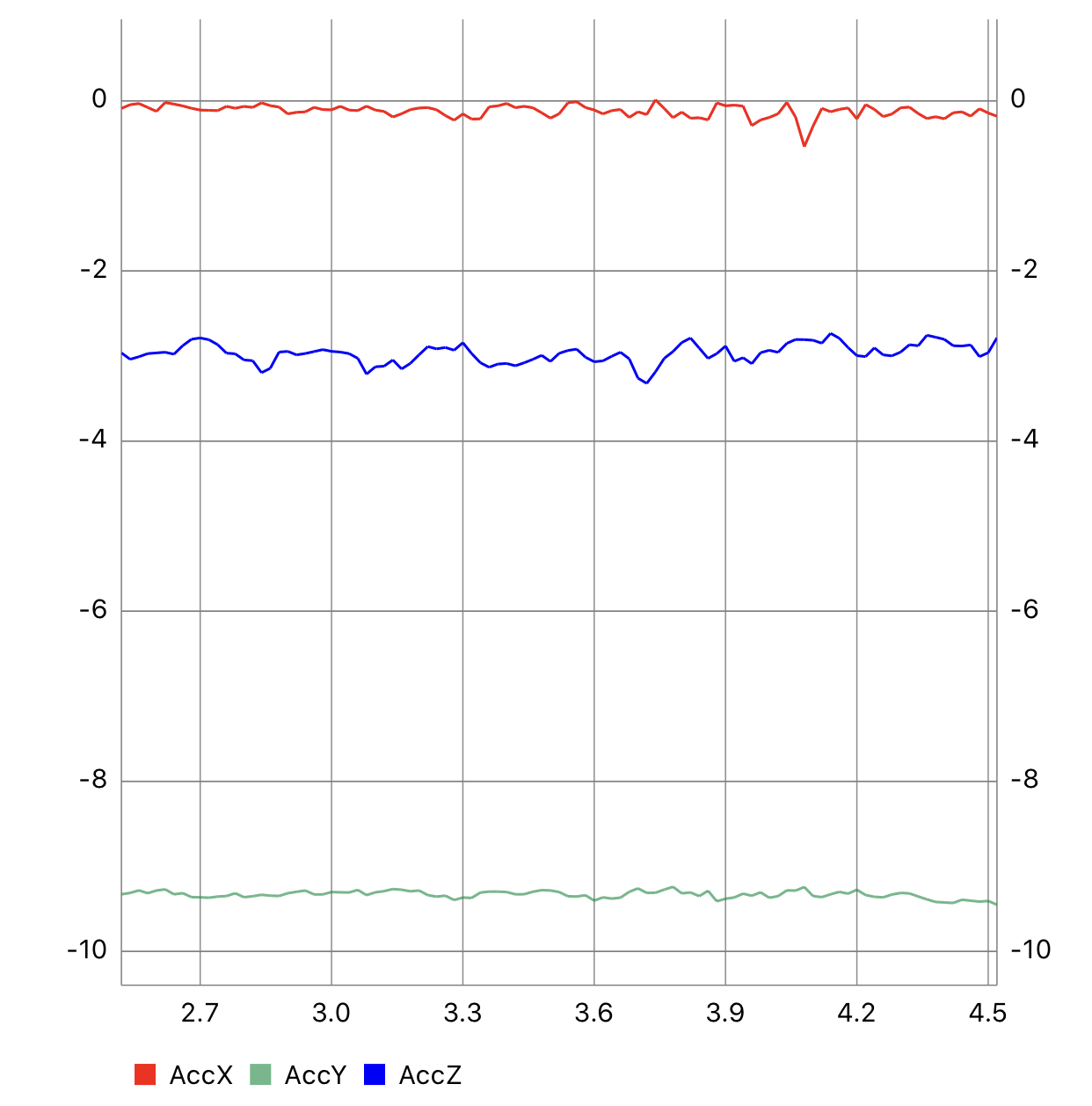
**- 센서 데이터 수집 어플을 이용한 분석**

|  |  |
| --- | --- |
| 걷다가 뛴 상태 | 뛰다가 걸은 상태 |
|  |  |
| 걷다가 멈춘 상태 | 뛰다가 멈춘 상태 |
|  |  |

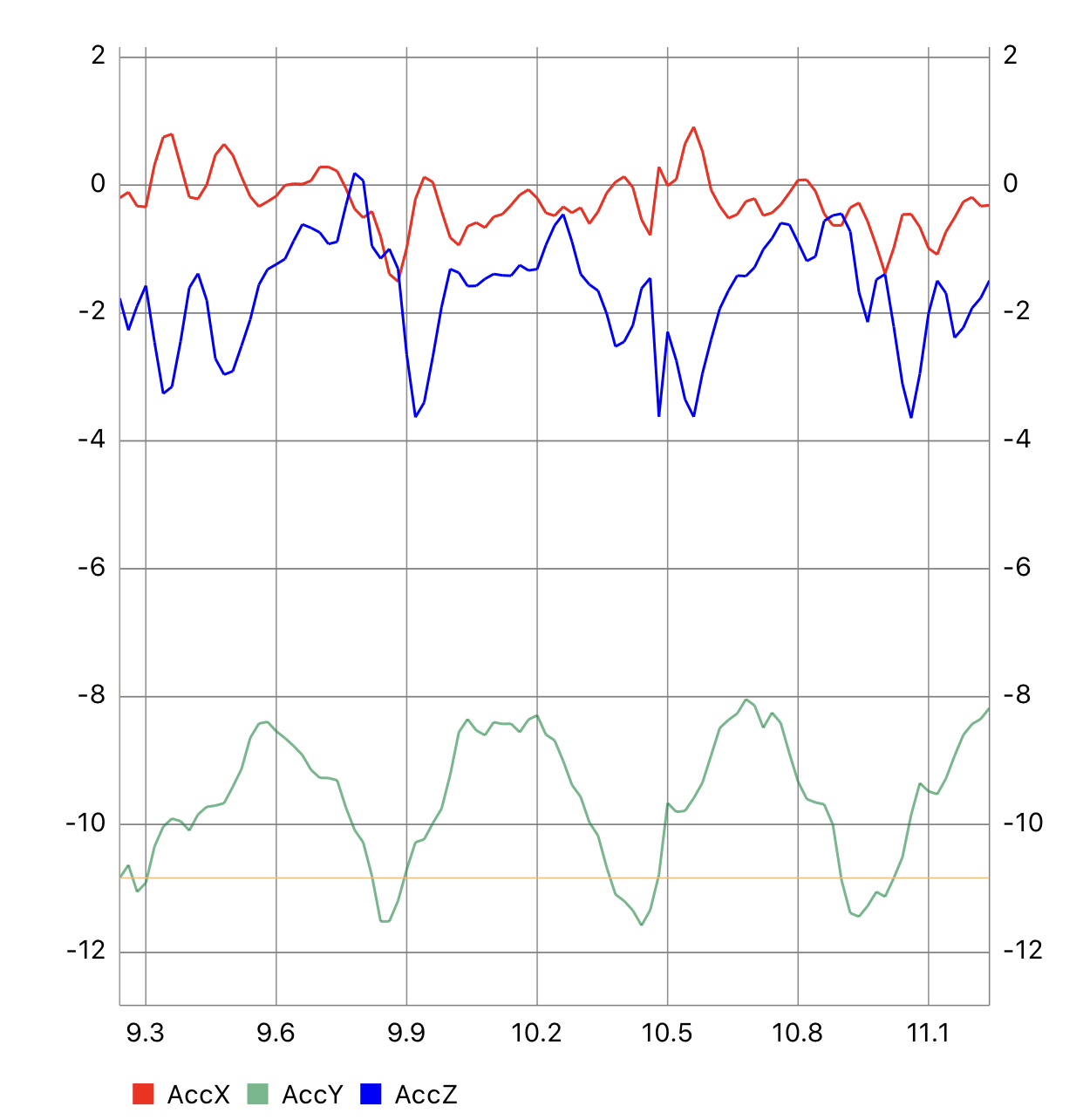
위의 네 가지 상태는 휴대폰 어플에 있는 가속도 센서를 통해 직접 실험한 사진들이다. 위의 사진들을 보면 뛴 상태, 걸은 상태, 멈춘 상태 세가지의 상태를 알아볼 수 있는데 가장 폭이 큰 경우가 뛴 상태고 폭이 중간 정도인 것이 걸은 상태 1자에 가까운 폭을 가진 상태가 멈춘 상태로 분석할 수 있다. 이와 같은 가속도의 데이터를 인공지능에게 학습시킴으로써 인공지능은 휴대폰을 들고 있는 사용자의 상태를 뛰는 상태, 걷는 상태, 멈춘 상태의 총 3가지 상태를 판단 할 수 있게 된다.

**- 가속도 센서의 증감 유형에 따른 분석**

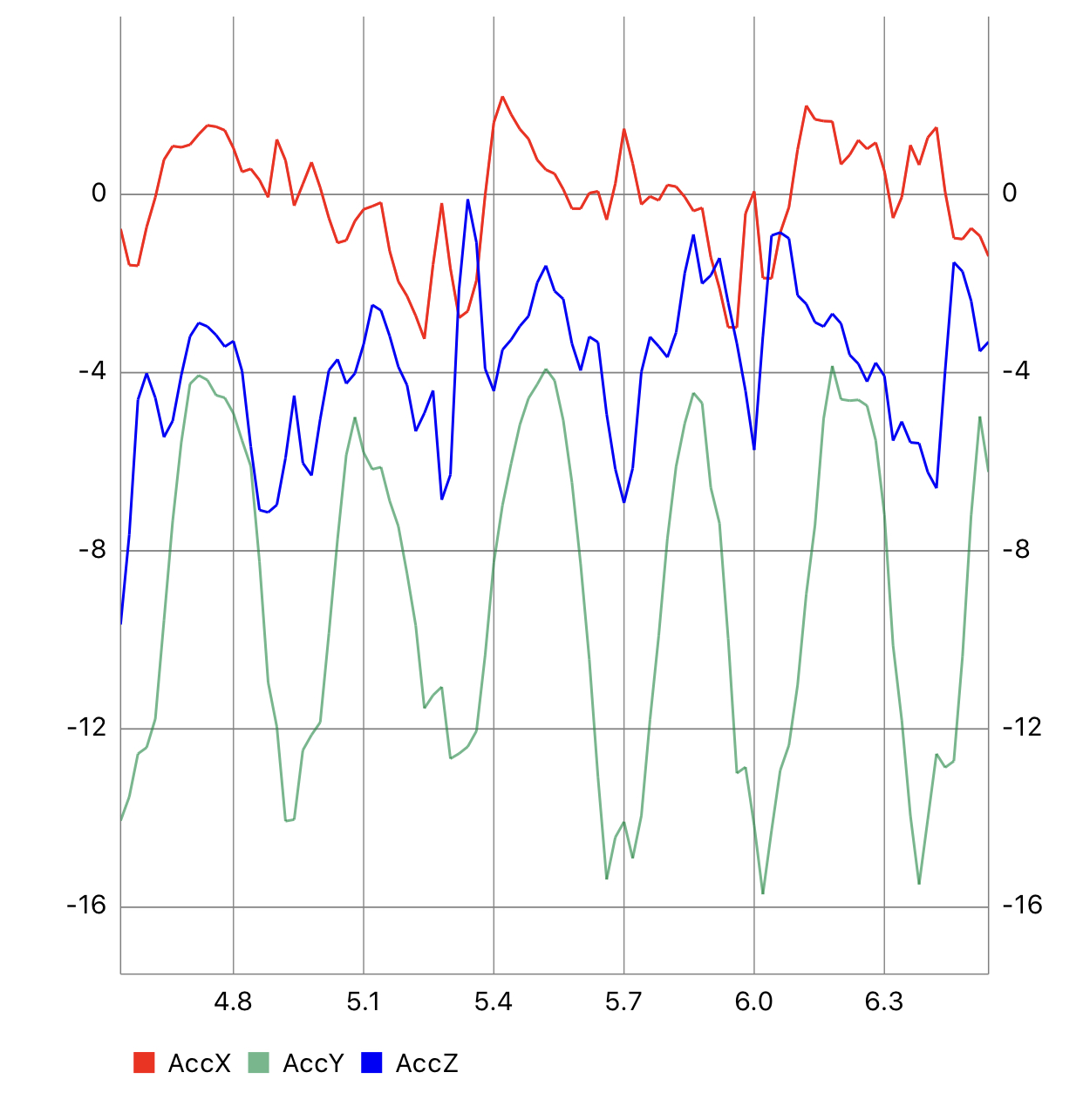
**멈춰있을 때**



**걸을 때**



**뛸 때**



정지와 걷기, 달리기에 따라 점점 가속도의 x, y, z 성분 값의 변동 폭이 커지는 양상이다.

세가지 상태의 그래프 개형을 분석했을 때 AccX와 AccY는 움직임이 격할수록 평균값이 기존보다 증가하는 경향을 보이고 AccZ는 같은 상황에서 평균값이 기존보다 감소하는 경향을 보인다.

가속도 센서는 자이로 센서가 측정한 변위로부터 가속도 성분들을 계산하는 식으로 동작하기 때문에 결과적으로 스마트폰의 가속도 센서만 활용해도 충분히 사용자의 현재 상태를 파악할 수 있을것으로 보인다. 따라서 가속도 성분의 증감 유형에 따라 사용자의 이동 상태를 판단할 수 있다.

**10. 파이썬의 Keras를 통한 데이터의 학습**

캐글을 통해 걷고 뛰는 것에 대한 가속도센서 x, y ,z 와 자이로 센서의 x, y, z의 값의 데이터를 수집했다. 이 데이터에서 걷는 것은 “0”, 뛰는 것은 “1”의 값을 가진다. 이 데이터들을 파이썬의 Keras 패키지를 통해 신경망을 구현해 학습을 진행하였다.

밑의 소스는 Sequential 모델을 사용해 레이어를 구성하고 학습을 진행한 소스이다.

-소스-

import numpy as np

import pandas as pd

import os

import numpy as np

import pandas as pd

from keras.layers import Dense, Input, BatchNormalization, Dropout

from keras.models import Sequential, Model

data = pd.read\_csv('./input/dataset.csv')

x\_train = data[['wrist', 'acceleration\_x', 'acceleration\_y', 'acceleration\_z', 'gyro\_x', 'gyro\_y', 'gyro\_z']]

y\_train = data[['activity']]

x\_train = np.array(x\_train)

y\_train = np.array(y\_train)

model = Sequential()

model.add(Dense(250, activation = 'relu', input\_shape = (7,)))

model.add(BatchNormalization())

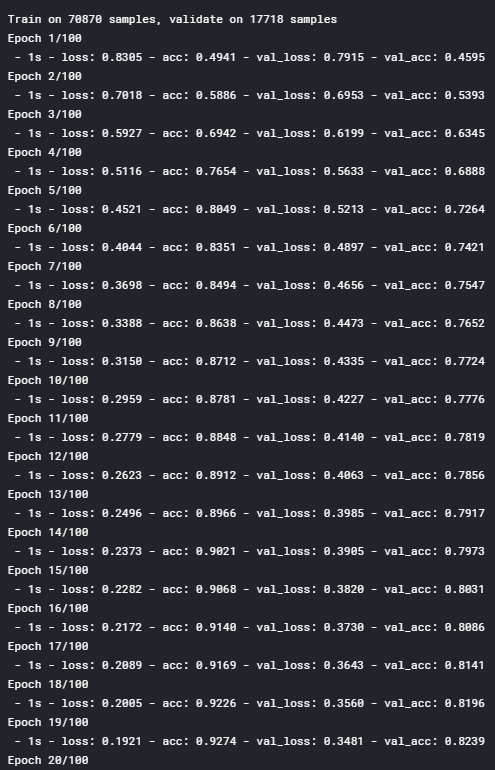
model.add(Dropout(rate = 0.25))

model.add(Dense(1, activation = 'sigmoid'))

model.compile(loss = 'binary\_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

model.fit(x\_train, y\_train, batch\_size = 40000, epochs = 100, validation\_split = 0.2, verbose = 2)

마지막 fit 메소드를 통해 model에 데이터셋을 통해 학습을 진행했다. 아래 사진을 학습한 사진이다.



위의 사진은 학습하는 것을 나타낸 사진의 일부로 총 70870개의 데이터를 학습하고 17718개의 샘플을 검증한 것을 의미한다.

loss값은 손실 값으로 결과값과의 차이를 의미하며 0에 가까울수록 좋은 모델이다.

acc는 정확도로 1에 가까울수록 좋은 모델이다.

그 후 test case인 x\_test를 생성한 후 precdict 메소드를 통해 출력된 y값을 각 클래스에 대한 판별함수로 가정하고 predict\_classes 메소드로 분류를 진행한 소스이다.

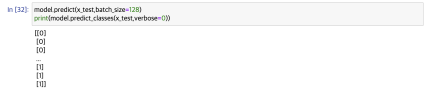
data1 = pd.read\_csv('./input/dataset1.csv')

x\_test = data1[['wrist', 'acceleration\_x', 'acceleration\_y', 'acceleration\_z', 'gyro\_x', 'gyro\_y', 'gyro\_z']]

x\_test = np.array(x\_test)

model.predict(x\_test,batch\_size=128)

print(model.predict\_classes(x\_test,verbose=0))



결과는 다음과 같이 나왔고 예측한 결과 테스트 데이터 셋의 처음에는 걷는 값인 0을 예측했고 후반에는 뛰는 값인 1을 예측했다.

**11. 분석 데이터 이용 계획**

위와 같은 데이터들을 인공지능은 학습함으로 사용자의 상태에 대한 정확한 판별을 하고 이를 사용자가 도착지까지 걸리는 시간을 계산할 때에 실시간으로 계산에 반영시켜 사용자가 얼마나 걷고 얼마나 뛰어야 하는지, 다른곳을 들럿다 가도 시간이 충분히 여유로운지 등 도착 가능성에 대한 판별을 가능토록 할 것이다.

**12. 어플리케이션 제작 진행 상황**

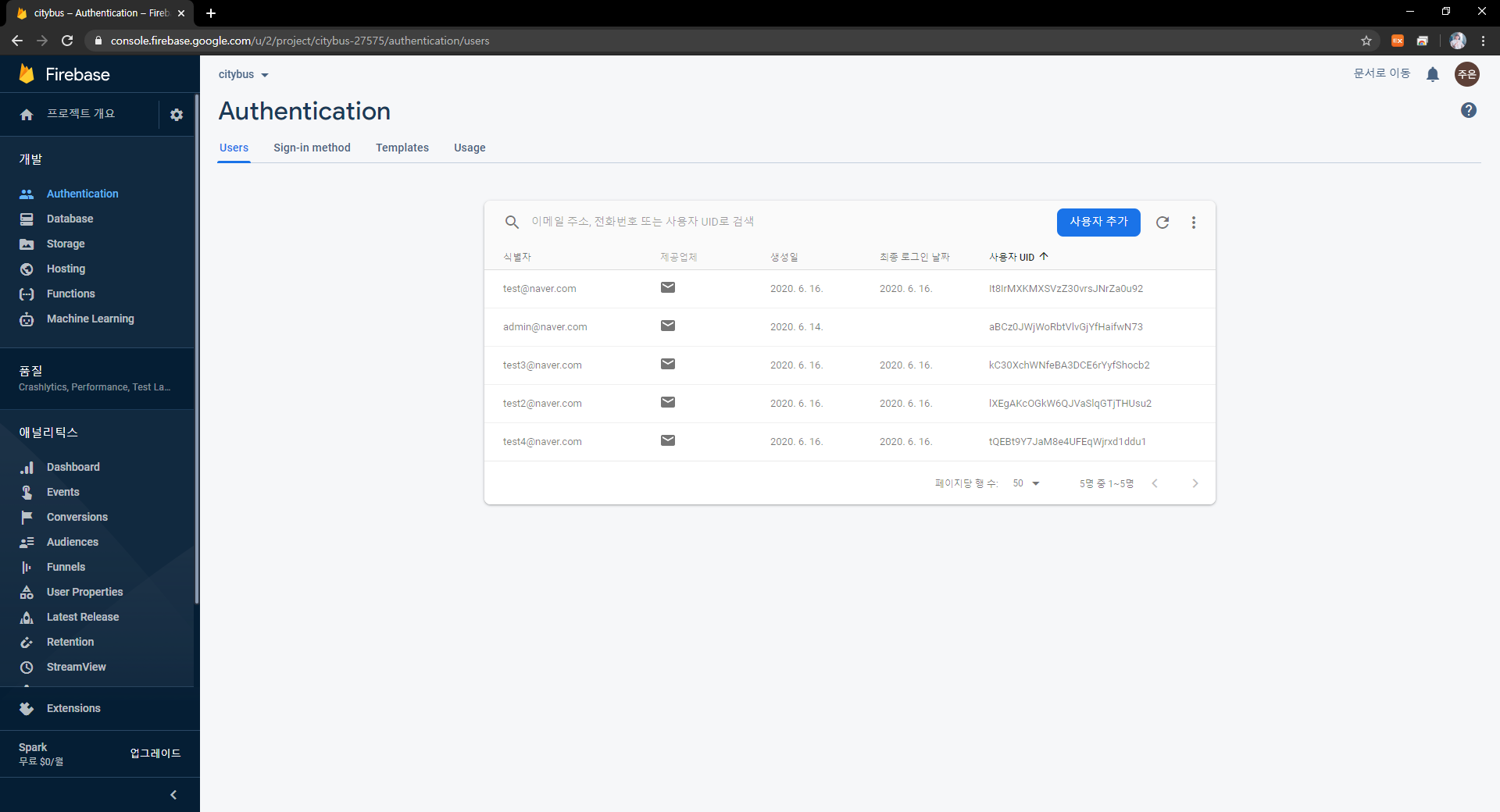
|  |  |
| --- | --- |
| **어플 화면** | **화면 설명** |
|  | **스플래시 화면**  어플을 시작하면 약 1초간 화면에 나왔다가 사라진다. |
|  | **로그인 화면**  아이디와 비밀번호를 입력하여 로그인 할 수 있으며 아이디와 비밀번호를 모두 입력해야 로그인 버튼이 활성화 된다. 잘못된 아이디나 비밀번호를 입력 했을 경우 “로그인 오류”라는 토스트 메시지가 뜬다.  ID 찾기는 아직 미구현 사항이며 PW찾기는 들어가면 비밀번호 재설정을 위한 이메일을 받을 이메일을 입력 할 수 있다.  회원가입을 누르면 회원가입 페이지로 간다. |
|  | **회원가입 화면**  이메일과 비밀번호를 입력하여 회원가입을 할 수 있으며 회원가입 버튼을 누르면 이메일과 비밀번호를 유저 데이터베이스에 업데이트 한다. 아이디의 경우 기존 유저와 같은 아이디가 입력될 경우 회원가입이 진행되지 않는다. |
|  | **비밀번호 찾기 화면**  회원가입시 입력했던 이메일 주소를 입력하면 해당 이메일로 비밀번호 재설정을 할 수 있는 이메일을 발송한다. |
|  | **티켓 결제화면**  출발위치, 버스 조회 기준 시간, 탑승 인원을 선택하고 버스를 조회 할 수 있는 화면이다. |
|  | **출발위치 선택 화면**  티켓 결제화면에서 출발위치를 클릭하면 현재 위치를 보여주고 현재위치를 기준으로 한 가까운 정류장을 표시해 준다. |
|  | **버스&도착지 선택화면**  출발위치 지정 후 버스 조회하기를 누르면 탑승할 버스 종류와 도착지를 선택 할 수 있다. 도착지는 선택한 버스가 갈수 있는 모든 정류장을 표시하며 다른 정류장은 표시하지 않는다. |
|  | **예매내용 확인 페이지**  버스 예매에 필요한 모든 정보가 선택되고 나면 예매 내역 사용자가 확인 할 수 있도록 한다. 출발지, 도착지, 결제 금액을 표시해 주며 다양한 결제 방법을 제공한다. 이후의 결제 프로세스는 다른 기업과의 협업이 필요하기에 구현하지 않았다. |
|  | **확인/변경 화면**  현재 결제되었으며 아직 탑승하지 않은 버스에 대해 변경/취소를 할 수 있는 화면이다. 표시하는 데이터는 데이터베이스에 올라와 있는 데이터를 불러와 표시한다. |
|  | **즐겨찾기 화면**  사용자가 자주가는 경로정보를 추가하여 즐겨찾기에 표시 할 수 있다. 이 페이지의 정보 또한 데이터베이스에서 불러온다. |
|  | **공지사항/이벤트 화면**  공지사항과 이벤트에 대한 정보들을 제공한다. 이 데이터는 사용자 아이디를 기반하여 제공하는 데이터가 아닌 모든 사용자에게 공통적으로 제공되는 데이터베이스에서 불러온 데이터들을 표시 해 준다. |
|  |
|  | **마이페이지 화면**  마이페이지 화면에서는 현제 결제한 티켓 중 가장 가까운 시간의 티켓을 표시 해주고 자주쓰는카드, 결제내역, 문의, 설정 페이지로 이동 할 수있도록 한다. 또한 좌측 상단의 로그아웃 버튼을 누르면 로그아웃되어 처음의 로그인 페이지로 돌아간다.  티켓을 표시하는 기능은 현재 미구현 상태이다. |
|  | **자주쓰는 카드 화면**  자주 쓰는 카드 화면에서는 빠른 결제를 위한 카드를 미리 등록 해 놓을 수 있도록 하는 화면이다. 현재 기능은 미구현 상태이다. |
|  | **결제내역 화면**  그동안 결제 한 티켓과 가격 정보들을 보여주는 화면이다. 데이터베이스에서 사용자별로 데이터를 불러 올 수 있도록 하였다. |
|  | 설정화면  설정화면에서는 어플과 사용자 정보에 대한 다양한 정보들을 수정 할 수 있도록 하였다. 상단의 이메일과 번호란에는 현재 로그인 되어있는 유저에 대한 정보를 표시한다. 현재는 이메일만 유저 정보에 맞추어 표시하도록 구현 되어있다.  푸쉬 알람 설정 옆의 스위치 버튼으로 푸쉬 설정을 할 수 있다. 현재는 미구현 상태이다.  이용약관을 누르면 이용약관을 볼 수 있는 페이지로 넘어간다.  회원탈퇴를 누르면 유저 데이터베이스에서 해당 유저에 대한 정보를 지우며 회원에서 제거한다.  로그아웃을 누르면 로그아웃되어 로그인 페이지로 돌아간다. |

**13. 데이터베이스 제작 진행 상황**

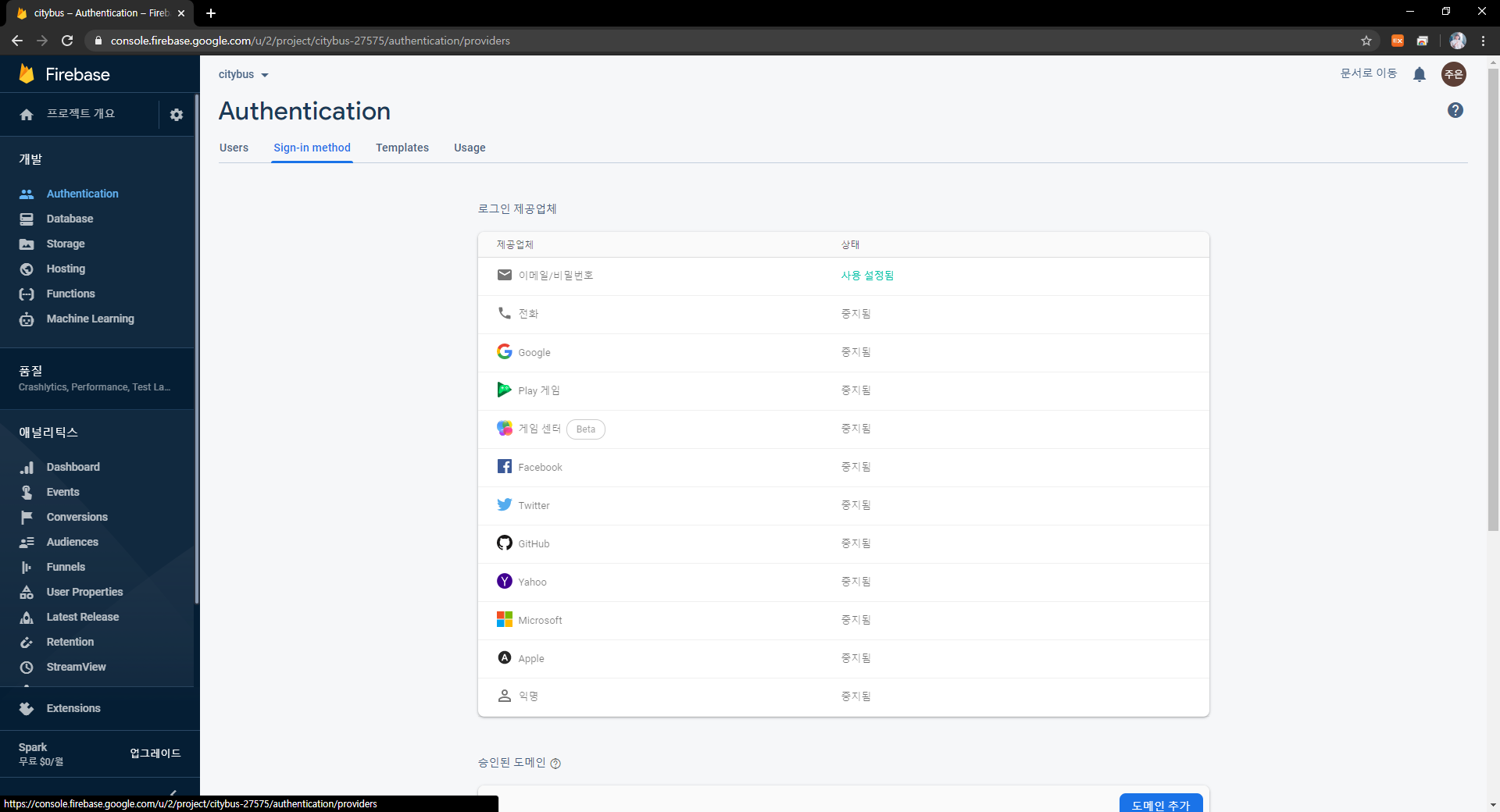
어플리케이션에 필요한 데이터베이스는 구글에서 제공하는 Firebase를 이용하였다.

Firebase를 이용하게 된 이유는 다양한 정보들에 대한 요약을 그래프와 수치로써 제공이 되어 한눈에 알아 볼 수 있고 구글에서 다양한 로그인 방법을 제공하는 등 다양한 장점들을 이용하기 위해서이다.

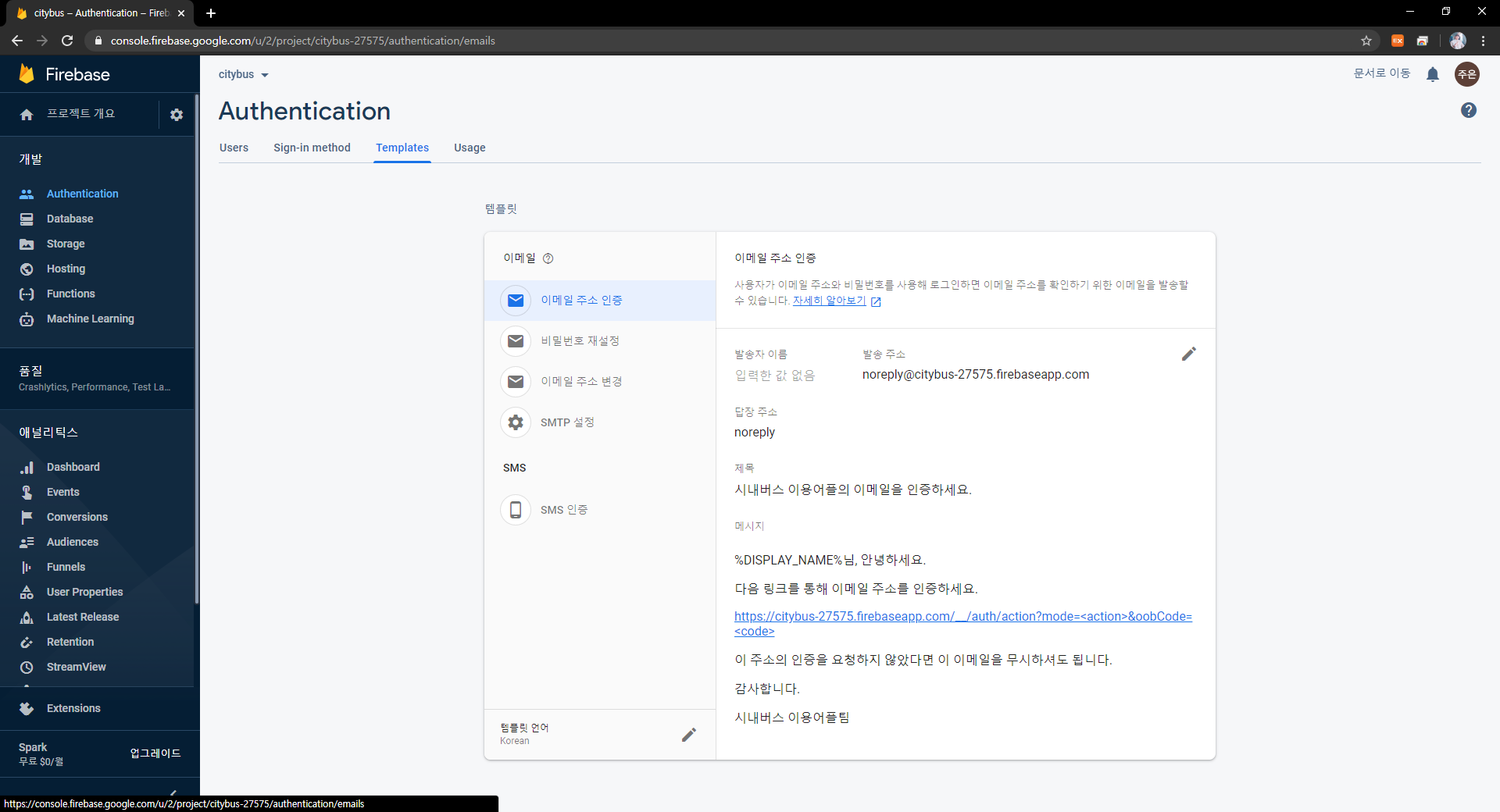
회원 관리와 연계되는 데이터는 Firebase Authentication을 이용하여 구현하였다.



위 페이지에서 회원가입된 유저들의 계정관리를 할 수있다.

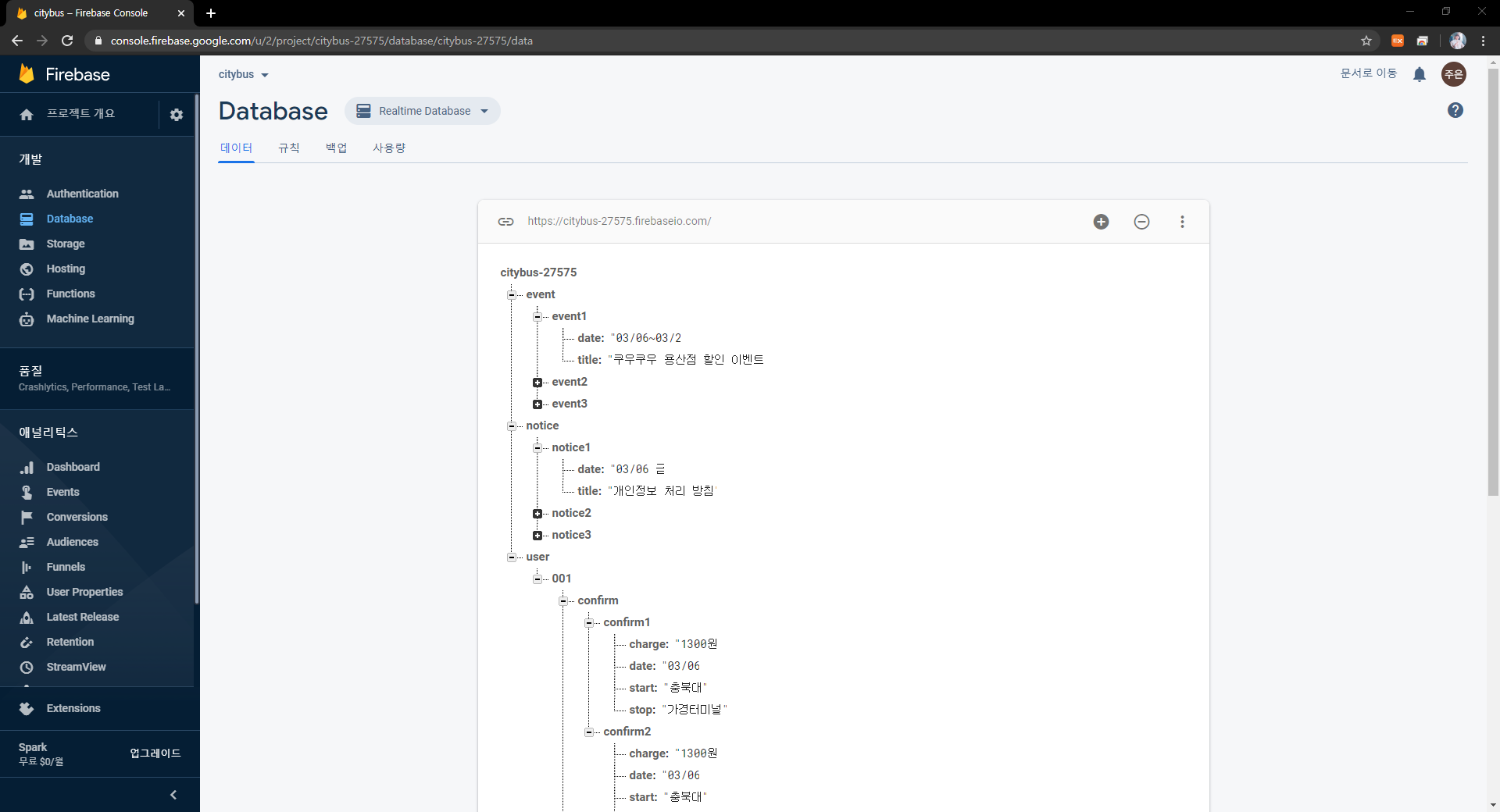


또한 Firebase에서는 다양한 로그인 방법을 제공하는데 현재는 이메일과 비밀번호를 통한 가입/로그인만 구현 하였다.



그리고 사용자 이메일 인증, 이메일, 비밀번호 변경과 관련된 이메일을 보낼 때 기본 템플릿을 지정 하였다.

유저관리를 제외한 다른 데이터들을 Firebase에서 제공하는 실시간 데이터베이스(Realtime Database)를 이용하였다.



Firebase의 Realtime Database는 SQL과 달리 테이블 형식이 아닌 트리 형식의 데이터베이스를 구성하도록 되어있다. 또한 데이터베이스의 백업을 자동적으로 할 수 있으며 데이터베이스의 사용량 등 다양한 정보를 그래프 형태로 볼 수 있다.

**14. MLP 학습을 통한 인공지능과 연계 결과**

목차10번의 내용에서 언급한 것과 같이 파이썬 서버에 인공지능을 업데이트/학습을 진행하고 어플리케이션과의 연동은 소켓통신을 이용하여 실시간으로 사용자의 상태에 대한 판단을 하고 그 결과를 어플리케이션에 표시 할 수 있도록 하였다.

아래는 파이썬 서버와 어플리케이션의 소켓통신에 대한 코드이다

import socket

import numpy as np

import pandas as pd

from keras.layers import Dense, Input, BatchNormalization, Dropout

from keras.models import Sequential, Model

host = '192.168.0.4' # 호스트 ip를 적어주세요

port = 9999 # 포트번호를 임의로 설정해주세요

data = pd.read\_csv('/input/dataset.csv')

data.info()

x\_train = data[['acceleration\_x', 'acceleration\_y', 'acceleration\_z', 'gyro\_x', 'gyro\_y', 'gyro\_z']]

y\_train = data[['activity']]

x\_train = np.array(x\_train)

y\_train = np.array(y\_train)

model = Sequential()

model.add(Dense(250, activation = 'relu', input\_shape = (6,)))

model.add(BatchNormalization())

model.add(Dropout(rate = 0.25))

model.add(Dense(1, activation = 'sigmoid'))

model.summary()

model.compile(loss = 'binary\_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

model.fit(x\_train, y\_train, batch\_size = 40000, epochs = 100, validation\_split = 0.2, verbose = 2)

while True :

server\_sock = socket.socket(socket.AF\_INET)

server\_sock.bind((host, port))

server\_sock.listen(1)

print("기다리는 중")

client\_sock, addr = server\_sock.accept()

print('Connected by', addr)

data = client\_sock.recv(1024)

num1 = data.decode("utf-8")

str1 = num1[0:7]

str2 = num1[7:14]

str3 = num1[14:21]

str4 = num1[21:28]

str5 = num1[28:35]

str6 = num1[35:42]

print(str1)

print(str2)

print(str3)

print(str4)

print(str5)

print(str6)

x\_test = np.array([[str1,str2,str3,str4,str5,str6]])

model.predict(x\_test,batch\_size=128)

data3=model.predict\_classes(x\_test,verbose=0)

print(data3[0])

if data3[0]==1 :

result=1

else :

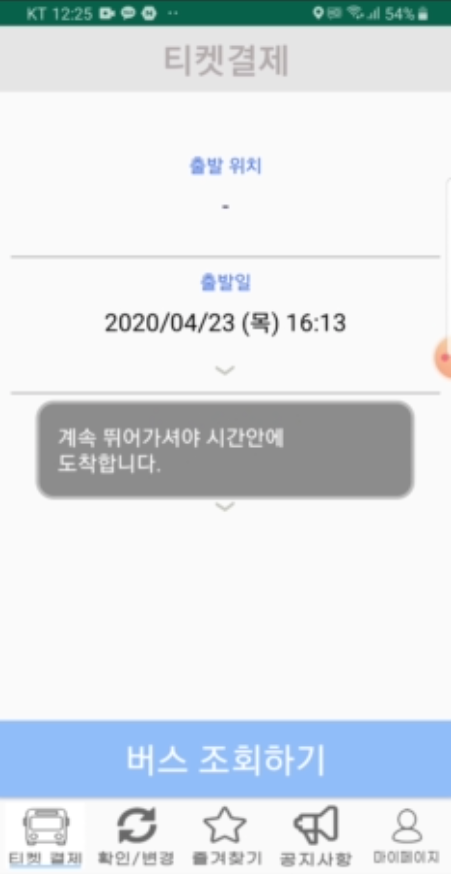
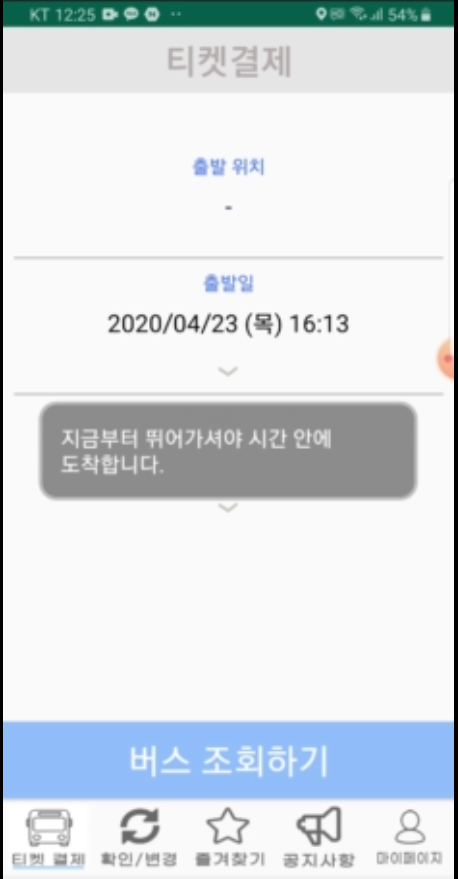
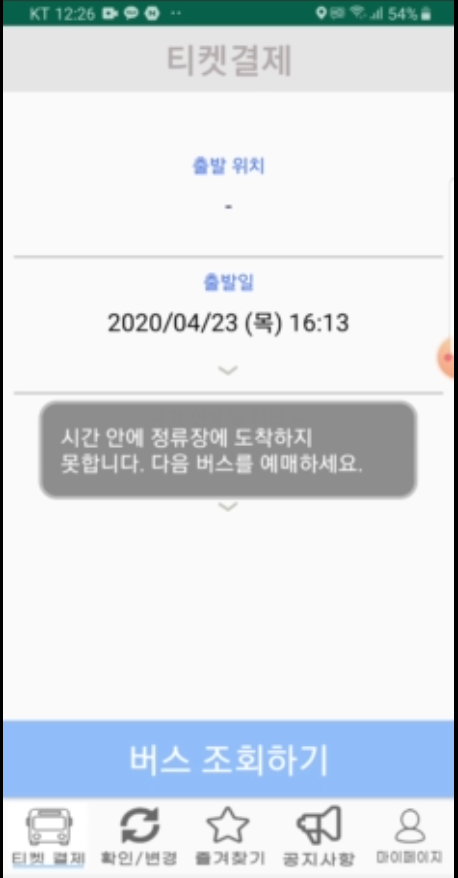
result=0

#print(data2.encode())

client\_sock.send(result.to\_bytes(4, byteorder='little'))

위 코드를 통해 사용자 핸드폰에서 가속도센서와 자이로센서의 값들을 인공지능에게 보내어 걷는 상태와 뛰는 상태에 대한 판단을 하도록 한다.

판단이후 결과값을 어플에 받아 어플에서는 버스 출발까지의 남은 시간과 사용자의 현재 상태 등을 종합적으로 판단하여 사용자가 늦지않게 도착하여 버스에 탑승 할 수 있도록 아래 이미지들과 같이 뛰어야 하는지, 걸어가도 되는지, 또는 다음 버스를 예매하여야 하는지 등 다양한 메세지를 출력한다.

**12. 참고문헌 및 자료**

<https://terms.naver.com/entry.nhn?docId=2423801&cid=51399&categoryId=51399> [네이버 지식백과] NFC (쇼핑용어사전)

<https://terms.naver.com/entry.nhn?docId=3571552&cid=59088&categoryId=59096> [네이버 지식백과] GPS (용어로 보는 IT)

<https://terms.naver.com/entry.nhn?docId=3572035&cid=59088&categoryId=59096> [네이버 지식백과] 블루투스 (용어로 보는 IT)

<http://biz.newdaily.co.kr/site/data/html/2016/10/11/2016101110036.html> 블루투스 리더기를 통한 스마트폰출입

관련 동영상: <https://www.youtube.com/watch?v=fgWzs08g3uk>

관련 자료 특허: KR20120068646A, KR20130105098A