

# Midtem Presentation

Capstone Design Project

이상해 C

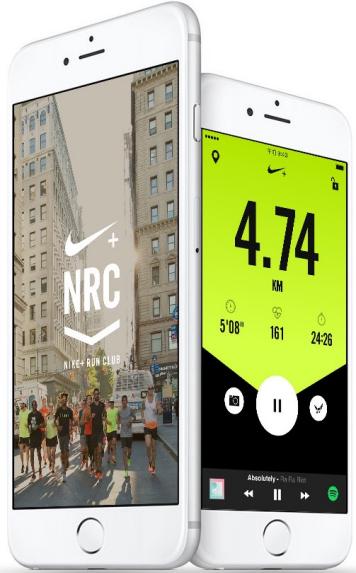
강승목, 김준석, 이혜원



# Index

1. Objective
2. Milestone
3. Role of each member
4. Design
5. Implementation
6. Challenges
7. Limitation
8. To-Dos
9. Demo

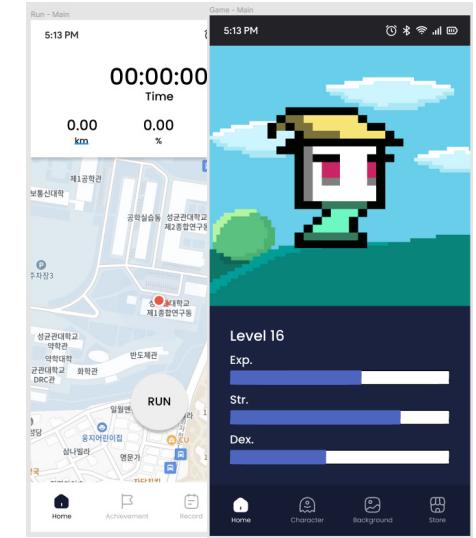
# 1. Objective



Running App

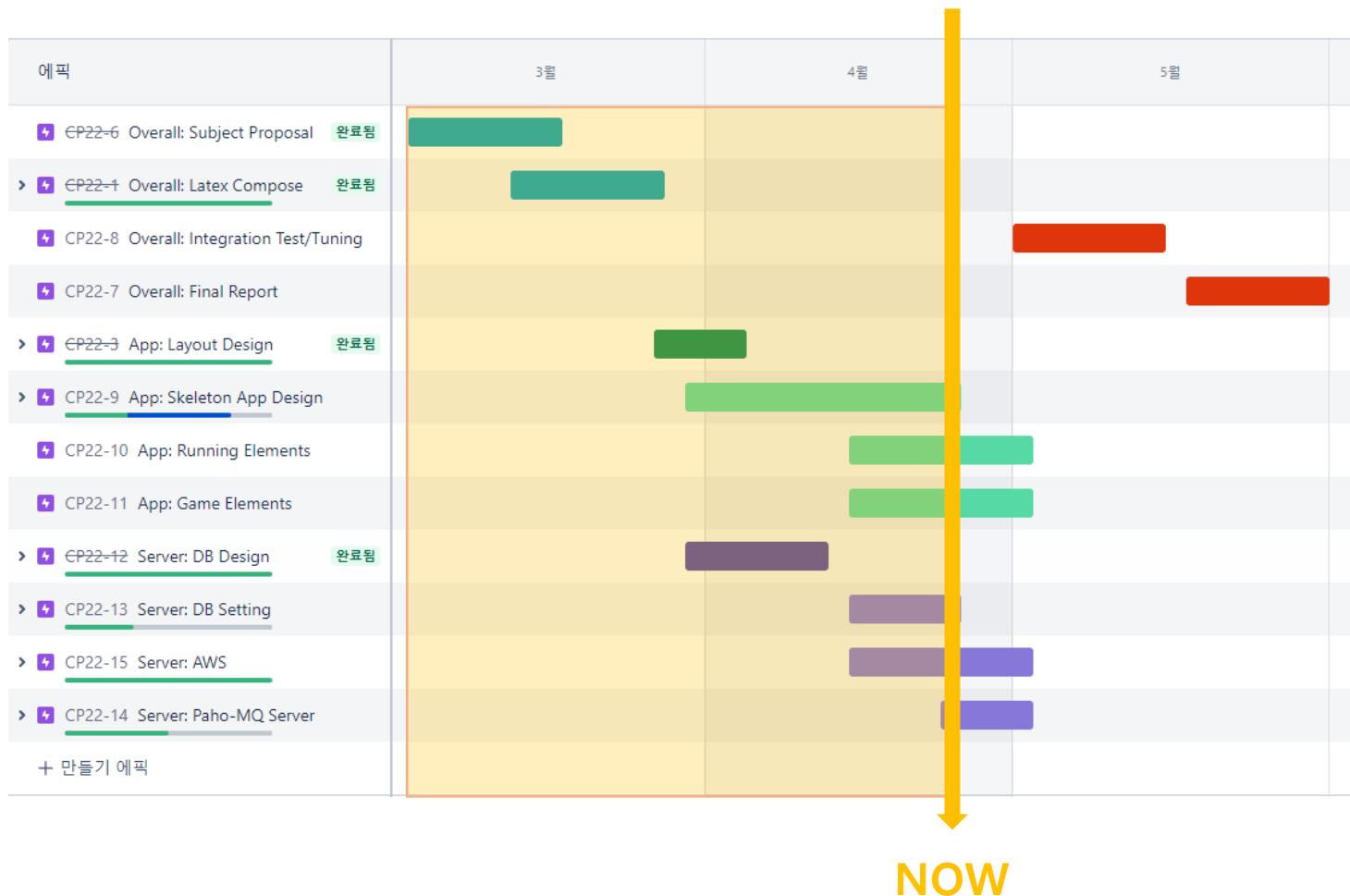


Virtual Character  
Simulation Game



Running App  
for Beginners

# 2. Milestone



### 3. Role of each member

Seungmok Kang	Hyewon Lee	Junsoek Kim	Jaeyoon Park
<b>Backend</b> <ul style="list-style-type: none"><li>• ER Diagram &gt; DB</li><li>• AWS Server</li></ul>		<b>Frontend</b> <ul style="list-style-type: none"><li>• Skeleton Code of Design</li><li>• Skeleton Code of Flow Chart</li></ul>	
<b>Frontend &amp; Backend</b> <ul style="list-style-type: none"><li>• Connecting Frontend with Backend</li><li>• Communication between Server-side and Client-side</li></ul>			

# 3. Role of each member

The screenshot shows a list of tasks assigned to members CP22-41 through CP22-32. Each task is associated with a specific feature, a server, and a member. A checkmark indicates the task is assigned to the member.

Task Description	Server	Assigned Member
FEAT : Connect Calls To DB	SERVER: PAHO-MQ SERVER	CP22-41
FEAT: User Data ER Design	SERVER: DB DESIGN	CP22-27
FEAT : Character Data ER Design	SERVER: DB DESIGN	CP22-28
FEAT : Landmark Data ER Design	SERVER: DB DESIGN	CP22-29
FEAT : 업적 Data ER Design	SERVER: DB DESIGN	CP22-30
FEAT : 포스터 Data ER Design	SERVER: DB DESIGN	CP22-31
FEAT : 테마 Data ER Design	SERVER: DB DESIGN	CP22-32

# 3. Role of each member

▼  이혜원 5개 이슈

FEAT : set up TSDB(for storing running data)

**SERVER: DB SETTING**

CP22-38 

FEAT : Connect DB to Python Server

**SERVER: DB SETTING**

CP22-40 

+ 이슈 만들기

RESEARCH : Cognito를 EC2랑 사용하는 방법

**SERVER: DB DESIGN**

CP22-36 

RESEARCH : MongoDB 대체제 있는지 조사

**SERVER: DB DESIGN**

CP22-37 

FEAT : setup RelationalDB(for storing User, Game Data)

**SERVER: DB SETTING**

CP22-39 

### 3. Role of each member

▼ **B** 김준석 5개 이슈

FEAT: Cover & Login/Register  
**APP: SKELETON APP DESIGN**  
 CP22-17 **B**

+ 이슈 만들기

FEAT: Landmarks  
**APP: SKELETON APP DESIGN**  
 CP22-21 **B**

...

+ 이슈 만들기

FEAT: Running Home  
**APP: SKELETON APP DESIGN**  
 CP22-18 ✓ **B**

FEAT: Achievements  
**APP: SKELETON APP DESIGN**  
 CP22-19 ✓ **B**

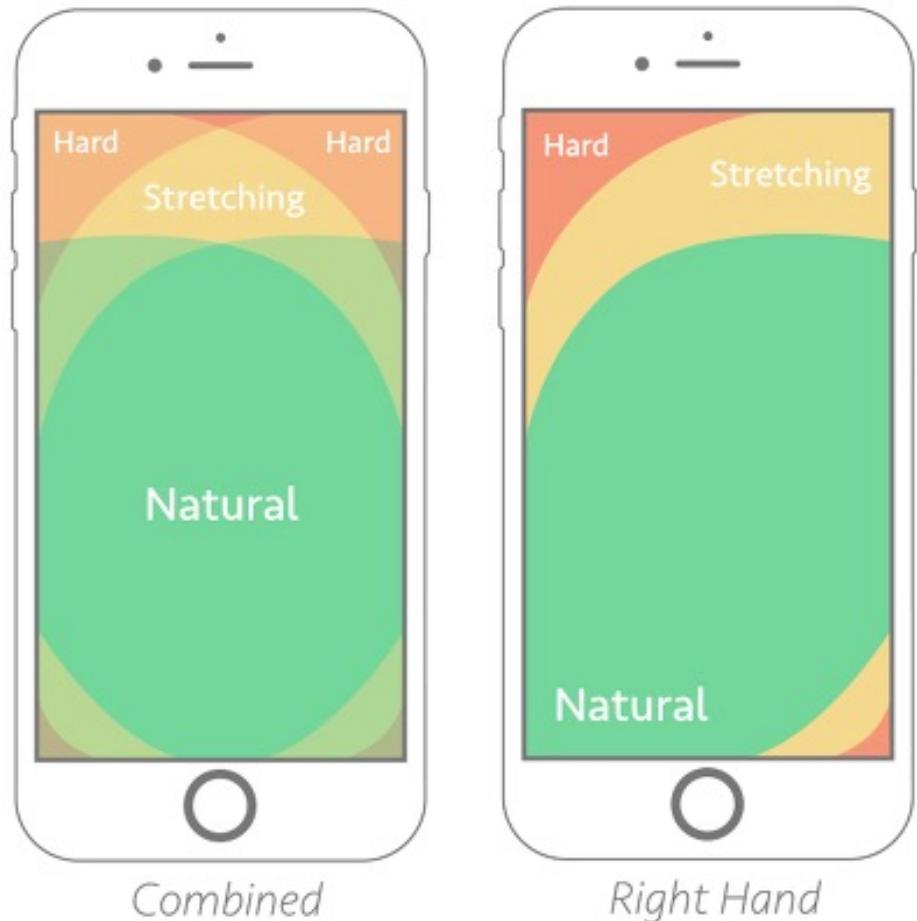
FEAT: My Records  
**APP: SKELETON APP DESIGN**  
 CP22-20 ✓ **B**

# 3. Role of each member

▼ JP Jaeyoon Park 6개의 이슈

FEAT: Settings APP: SKELETON APP DESIGN <input checked="" type="checkbox"/> CP22-26	FEAT: Character Home APP: SKELETON APP DESIGN <input checked="" type="checkbox"/> CP22-22	FEAT: Layout design APP: LAYOUT DESIGN <input checked="" type="checkbox"/> CP22-16
+ 이슈 만들기		
FEAT: Gacha APP: SKELETON APP DESIGN <input checked="" type="checkbox"/> CP22-23	FEAT: Character List APP: SKELETON APP DESIGN <input checked="" type="checkbox"/> CP22-24	FEAT: Background Poster APP: SKELETON APP DESIGN <input checked="" type="checkbox"/> CP22-25

# 4. Design - UX/UI



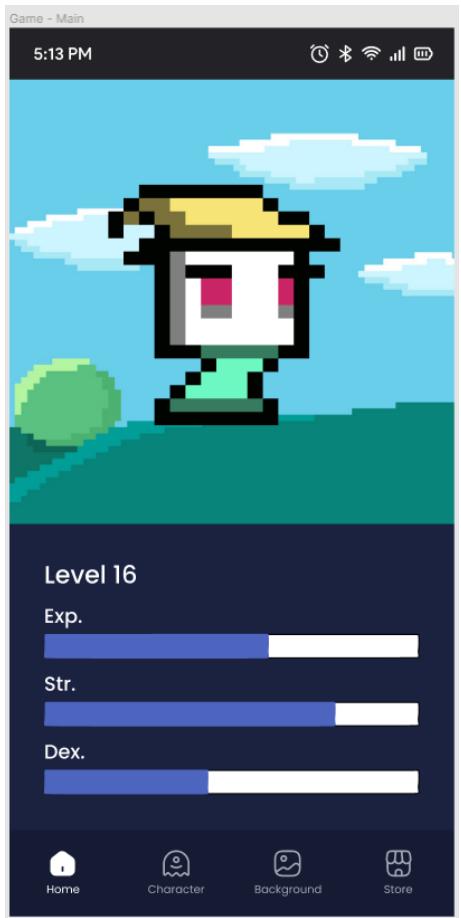
## The Thumb Zone

- Important links
  - ▶ easy-to-reach zone
- Unimportant links
  - ▶ hard-to-reach zones

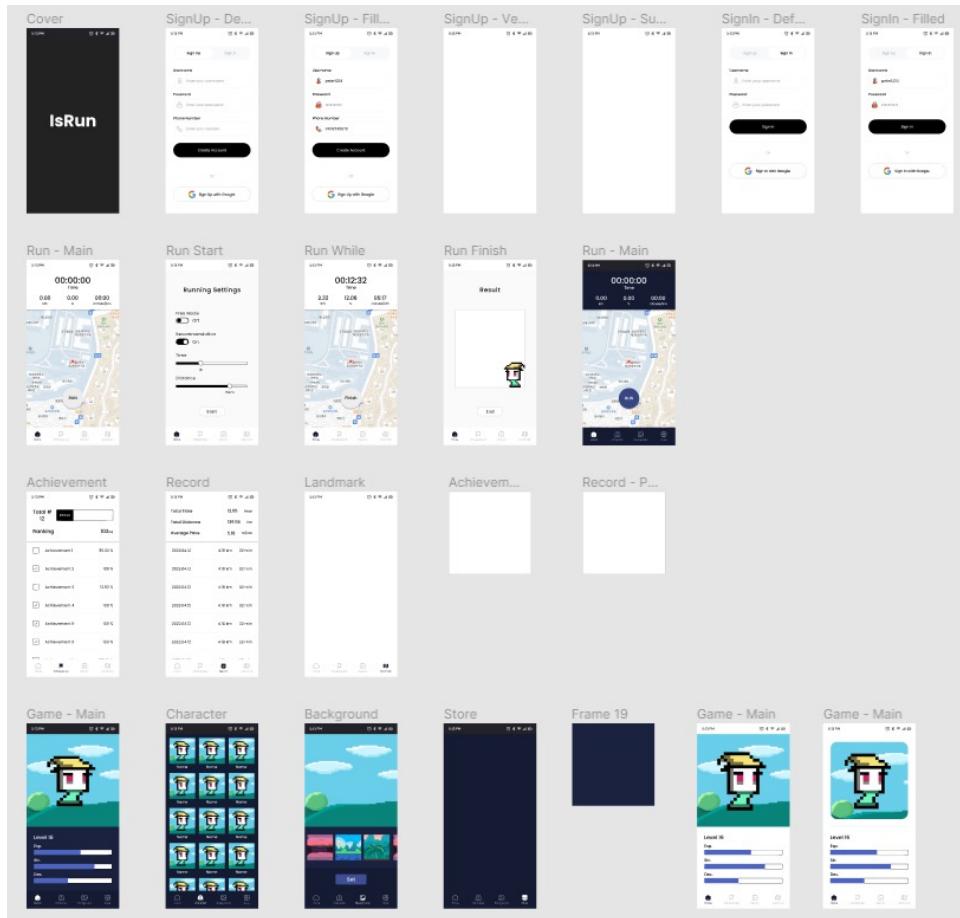
# 4. Design - UX/UI



Running



Gaming

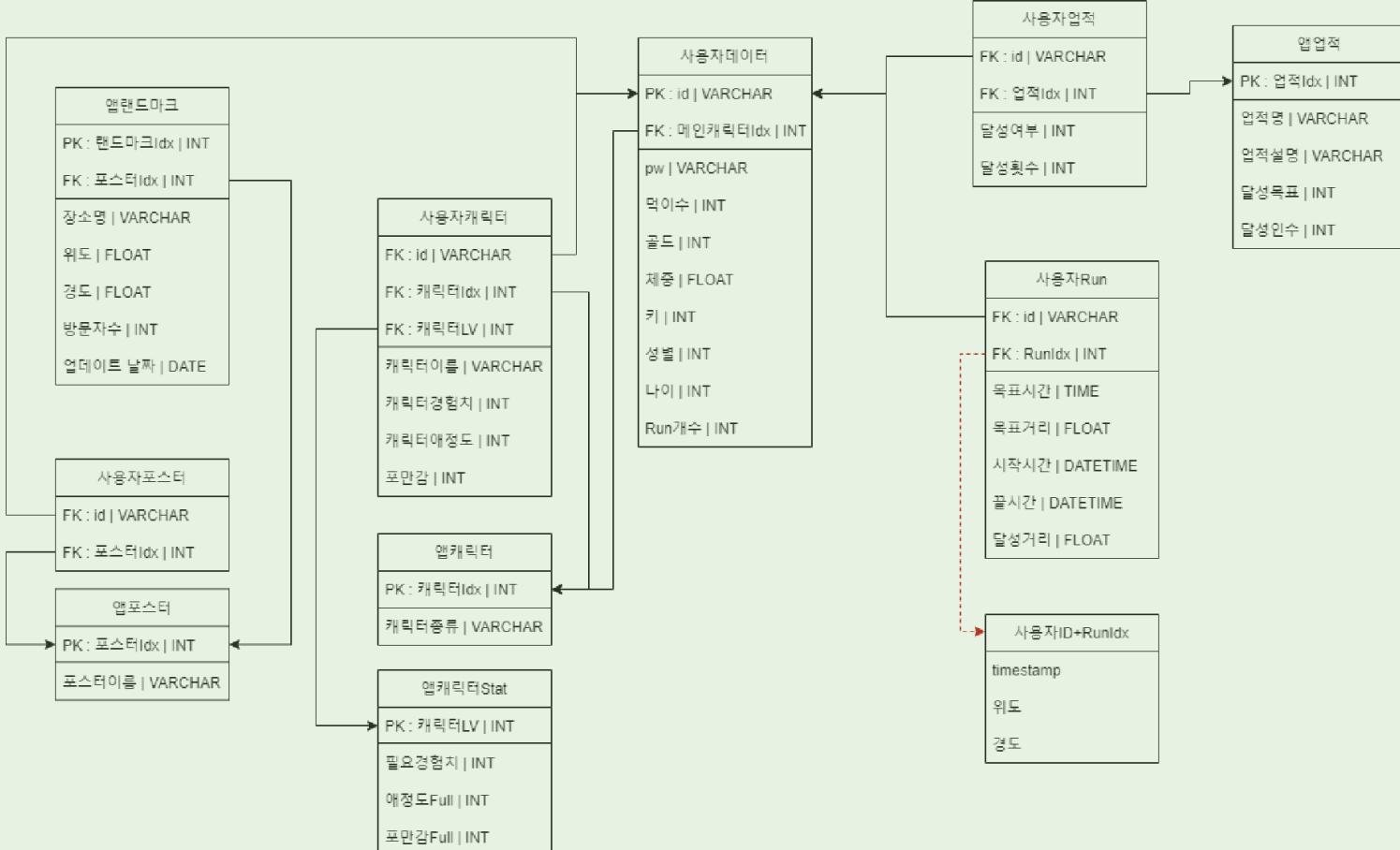


Overall

# 4. Design - Server



# 4. Design - DB



# 5. Implementation – frontend



## IDE

- Android studio

## Language

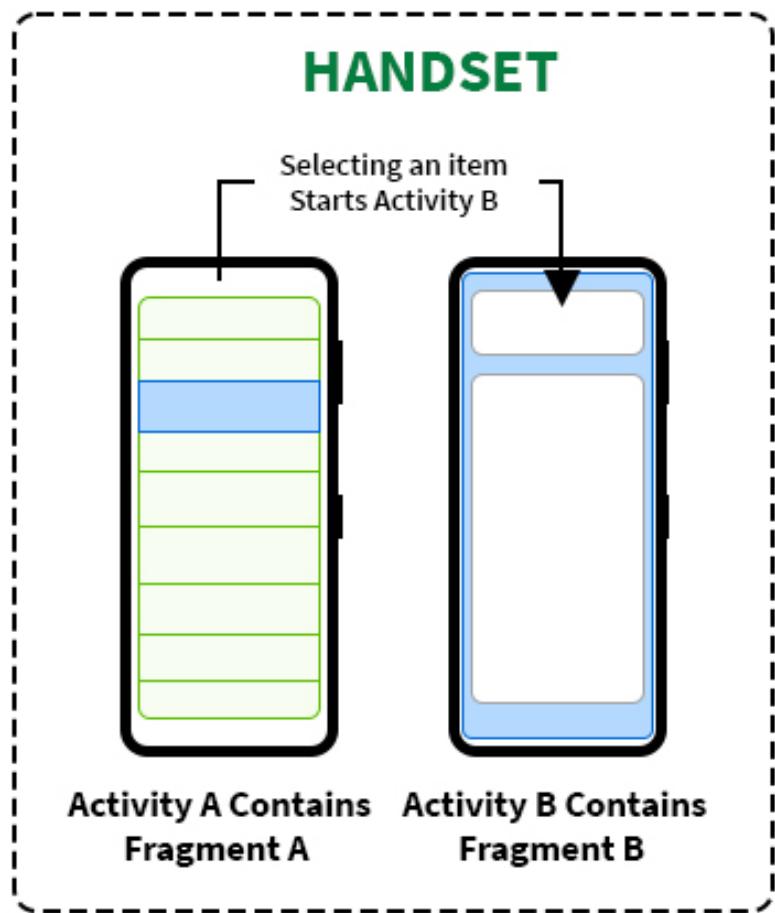
- Kotlin

## Tools

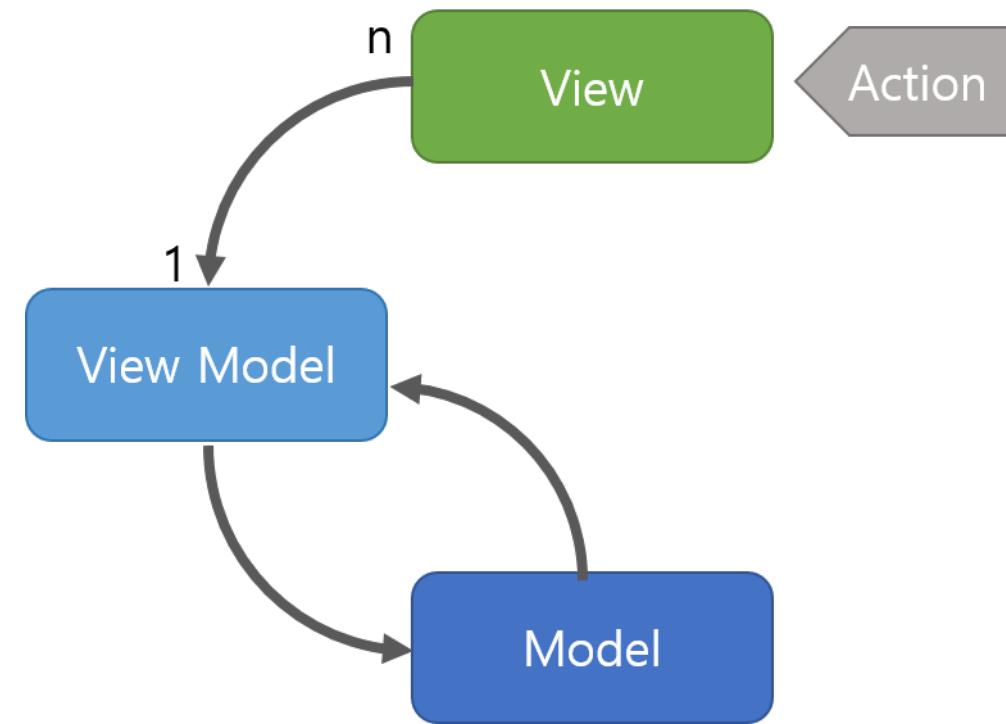
- Fragment
- MVVM
- Navigation

# 5. Implementation - frontend

## - Fragments -

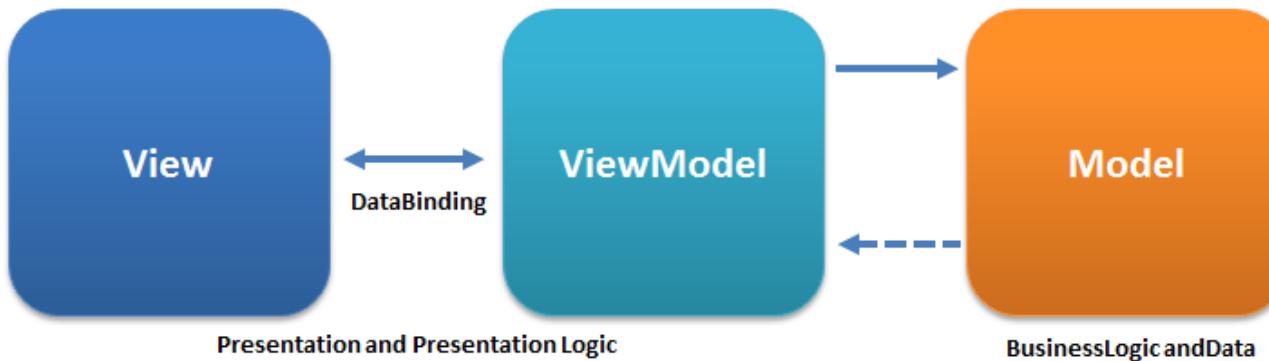


## - MVVM Design Pattern -



# 5. Implementation - frontend

## - MVVM Design Pattern -



### Characteristic

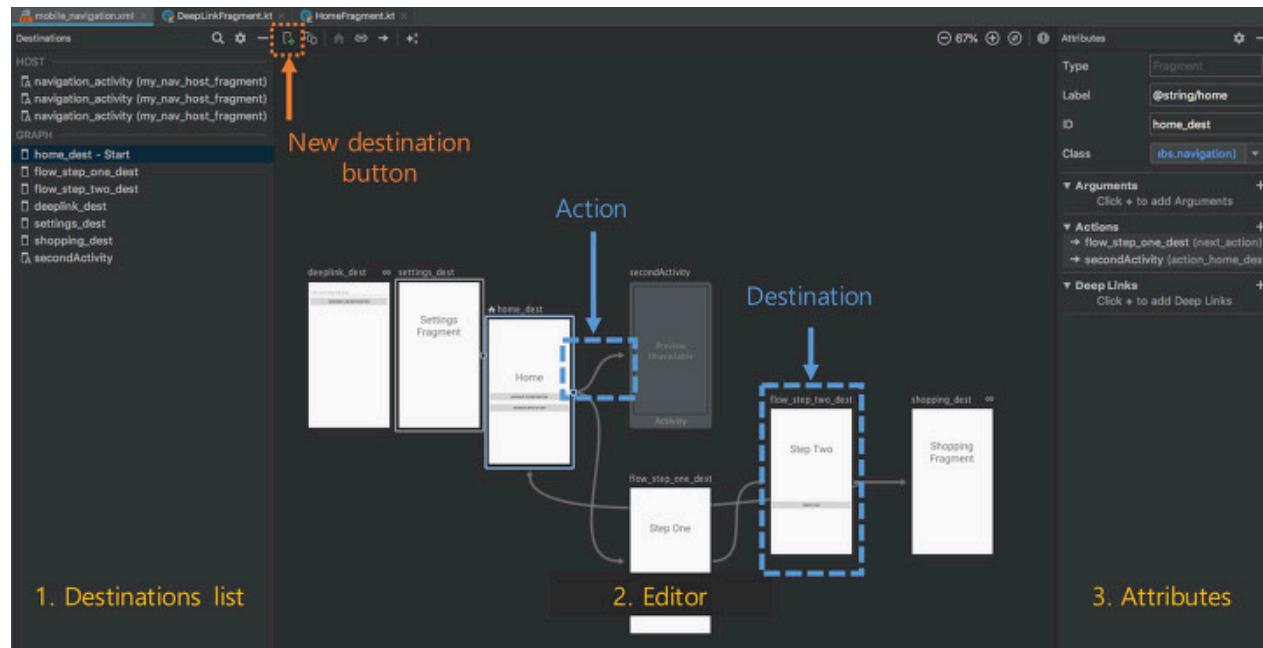
- View Model(1) : View(n)
- Command & Data Binding Pattern
  - ➡ **No Dependency** between View and View Model

### Pros & Cons

- Pros: **No Dependency** between View and View Model
  - ➡ Easy Modularization
- Cons: **Not easy to construct!!**

# 5. Implementation - frontend

## - Navigation -



- NavHostFragment
- NavController
- Destination
- Action

# 5. Implementation - backend



PYTHON



## Server Environment

- AWS EC2

## Language

- Python

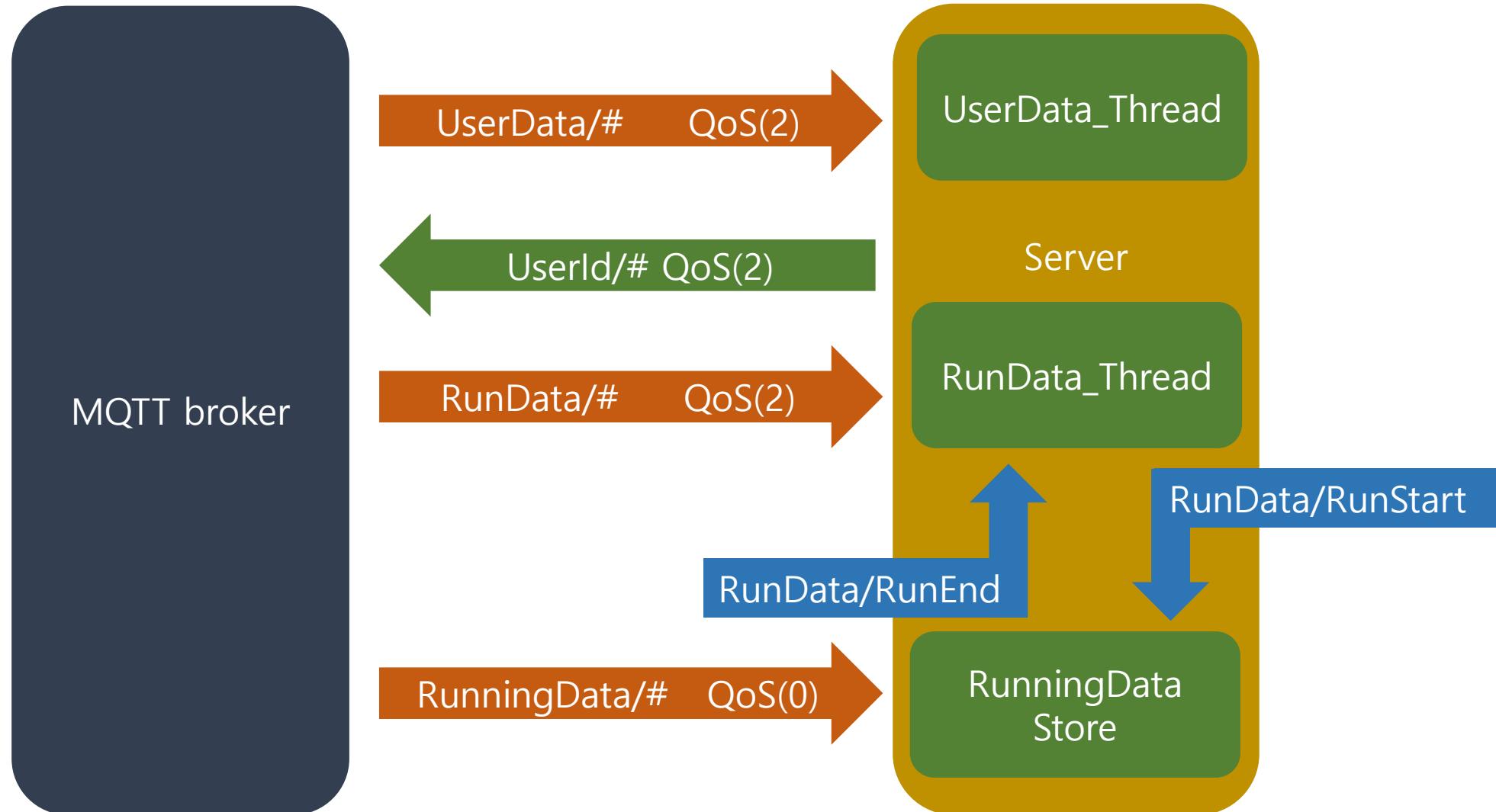
## DB

- MySQL & InfluxDB

## Protocol

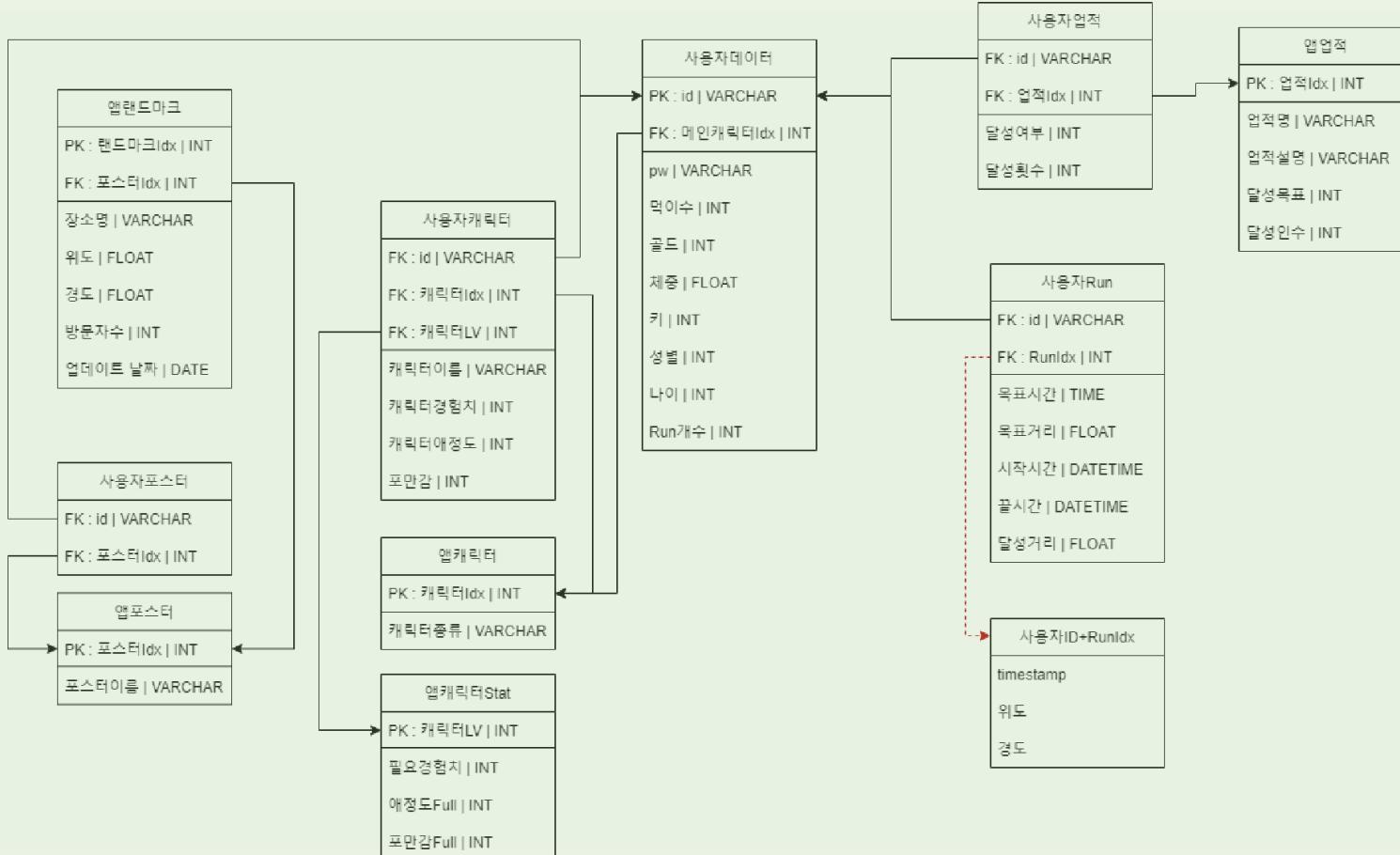
- MQTT

# 5. Implementation - backend



# 5. Implementation - backend

## DB(Data Base)



# 5. Implementation - backend

```
ubuntu@ip-172-31-46-53: ~/Project/TestCodes
+-----+
| aaaa11 |    NULL | bbbb22 |    NULL |    NULL |    55.5 |    166 |     0 |    20 |    NULL |
| asdf1999 |    NULL | qwer1999 |    NULL |
| cccc33 |    NULL | dddd44 |    NULL |    NULL |    66.6 |    177 |     1 |    21 |    NULL |
+-----+
3 rows in set (0.00 sec)

mysql> SELECT * FROM userdata;
+-----+
| id | mcharidx | pw | food | gold | weight | height | sex | age | run |
+-----+
| aaaa11 |    NULL | bbbb22 |    NULL |    NULL |    55.5 |    166 |     0 |    20 |    NULL |
| asdf1999 |    NULL | qwer1999 |    NULL |
| cccc33 |    NULL | dddd44 |    NULL |    NULL |    66.6 |    177 |     1 |    21 |    NULL |
+-----+
3 rows in set (0.00 sec)

mysql> show tables;
+-----+
| Tables_in_sqDB |
+-----+
| appachieve |
| appchar |
| appcharstat |
| applandmark |
| appposter |
| userachieve |
| userchar |
| userdata |
| userposter |
| userrun |
+-----+
10 rows in set (0.00 sec)

mysql> SELECT * FROM userachieve;
+-----+
| id | achieveidx | yesorno | many |
+-----+
| cccc33 |    0 |    1 |    1 |
| cccc33 |    1 |    1 |    1 |
| aaaa11 |    0 |    1 |    1 |
+-----+
3 rows in set (0.00 sec)

mysql> SELECT * FROM appchar;
+-----+
| id | mcharidx | pw | food | gold | weight | height | sex | age | run |
+-----+
| aaaa11 |    NULL | bbbb22 |    NULL |    NULL |    55.5 |    166 |     0 |    20 |    NULL |
| asdf1999 |    NULL | qwer1999 |    NULL |
| cccc33 |    NULL | dddd44 |    NULL |    NULL |    66.6 |    177 |     1 |    21 |    NULL |
+-----+
3 rows in set (0.00 sec)

mysql> SELECT * FROM appachieve;
+-----+
| achidx | achname | achexplain | achgoal | achnum |
+-----+
| 0 | 신규가입 | 새로 가입한 사람 | 1 | 20 |
| 1 | 신입 러너 | 1km를 뛰었다! | 1 | 18 |
| 2 | 산책중입니다 | 평균 4km/h 이하로 달린다 | 1 | 3 |
+-----+
3 rows in set (0.00 sec)

mysql>
```

## MySQL

- ▶ Implementation complete
- ▶ Sample Data inserted

# 5. Implementation - backend

```
515 # sub to UserData#
516 def UserData_Thread():
517     UserData_Client = mqtt.Client()
518     # Default
519     UserData_Client.on_connect = on_connect
520     UserData_Client.on_disconnect = on_disconnect
521     UserData_Client.on_subscribe = on_subscribe
522     # SignIn Message Handler
523     UserData_Client.message_callback_add("UserData/SignIn", SignIn_message)
524     UserData_Client.message_callback_add("UserData/SignUp", SignUp_message)
525
526     UserData_Client.message_callback_add("UserData/UpdateUserData", UpdateUserData_message)
527     UserData_Client.message_callback_add("UserData/GetUserData", GetUserData_message)
528
529     UserData_Client.message_callback_add("UserData/UpdateUserInfo", UpdateUserInfo_message)
530     UserData_Client.message_callback_add("UserData/GetUserInfo", GetUserInfo_message)
531
532     UserData_Client.message_callback_add("UserData/GetAchieves", GetAchieves_message)
533     UserData_Client.message_callback_add("UserData/GetRuns", GetRuns_message)
534     UserData_Client.message_callback_add("UserData/GetLandRecommend", GetLandRecommend_message)
535
536     UserData_Client.message_callback_add("UserData/UpdateChar", UpdateChar_message)
537     UserData_Client.message_callback_add("UserData/UpdateCharLevel", UpdateCharLevel_message)
538
539     UserData_Client.message_callback_add("UserData/GetChars", GetChars_message)
540     UserData_Client.message_callback_add("UserData/GetBackGs", GetBackGs_message)
541     UserData_Client.message_callback_add("UserData/GetGachaS", GetGachaS_message)
542     UserData_Client.message_callback_add("UserData/GetGachaF", GetGachaF_message)
543     # HiveMQ를 사용합니다.
544     UserData_Client.connect('broker.mqttdashboard.com', 1883)
545     # 해당 토픽에 구독해서 메시지를 확인합니다.
546     UserData_Client.subscribe("UserData/#", 2)
547
548     UserData_Client.loop_forever()
549
```

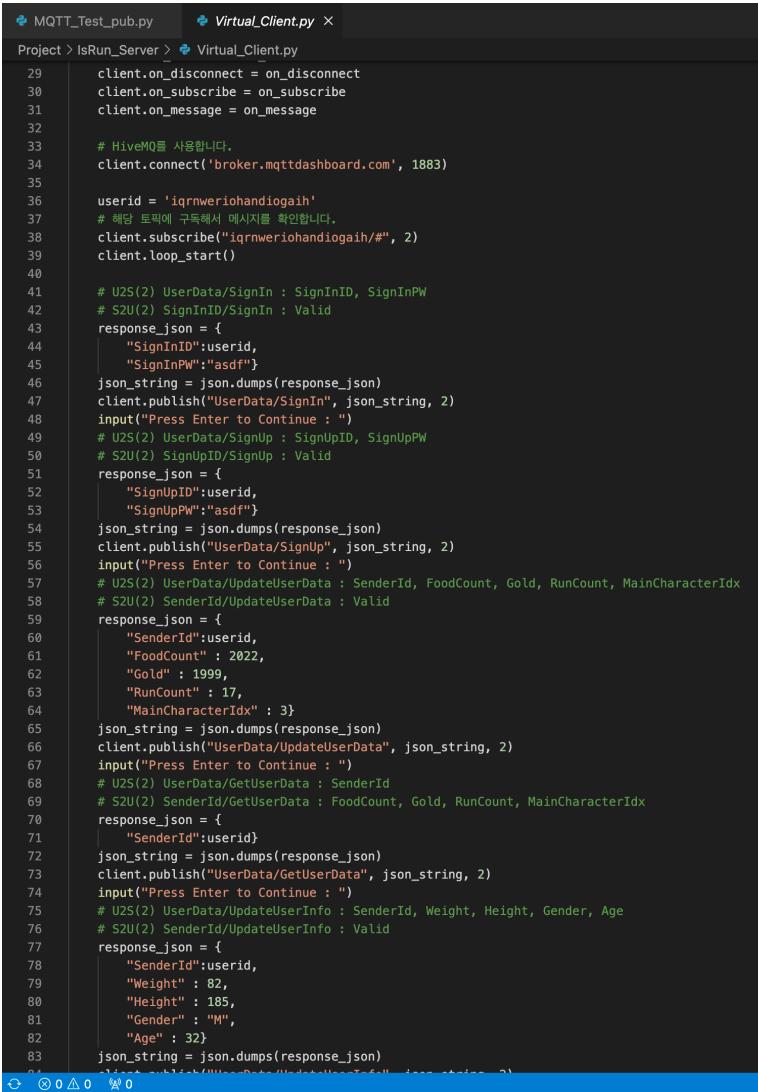
- ▶ Data Parsing for Predicted Calls
- ▶ e.g.  
UserData/SignIn  
UserData/GetRuns  
etc.

# 5. Implementation - backend

```
550 # U2S(2) RunData/RunStart : SenderId, RunIdx, Time_Goal, Dist_Goal, Start_Time, StartLat, StartLon
551 # S2U(2) SenderId/RunStart : NewPosterIdx
552 def RunStart_message(client, userdata, msg):
553     # Parse Data
554     json_data = json.loads(str(msg.payload.decode("utf-8")))
555     print(json_data)
556     SenderId = None
557     RunIdx = None
558     Time_Goal = None
559     Dist_Goal = None
560     Start_Time = None
561     StartLat = None
562     StartLon = None
563     if "SenderId" in json_data.keys():
564         SenderId = json_data["SenderId"]
565     if "RunIdx" in json_data.keys():
566         RunIdx = json_data["RunIdx"]
567     if "Time_Goal" in json_data.keys():
568         Time_Goal = json_data["Time_Goal"]
569     if "Dist_Goal" in json_data.keys():
570         Dist_Goal = json_data["Dist_Goal"]
571     if "Start_Time" in json_data.keys():
572         Start_Time = json_data["Start_Time"]
573     if "StartLat" in json_data.keys():
574         StartLat = json_data["StartLat"]
575     if "StartLon" in json_data.keys():
576         StartLon = json_data["StartLon"]
577     # Send Response
578     if None in [SenderId, RunIdx, Time_Goal, Dist_Goal, Start_Time, StartLat, StartLon]:
579         print("ERROR RunStart")
580     else:
581         client.message_callback_add("RunningData/{}/{}/{}".format(SenderId, RunIdx), RunningData_message)
582         =====Handle DB=====
583         # 위치 기반으로 어떤 랜드마크 보상 가능인지 확인할것.
584         # 그냥 가장 가까운거 아무거나 하나 쥐버릴까도 생각중.
585         # UserData 테이블에서 RunCount++ 할것.
586         # UserRun 테이블에 Time_Goal, Dist_Goal, Start_Time등록할것.
587         # SenderID+RunIdx로 influxDB 테이블 생성할것.
588         NewPosterIdx = None
589         =====
590         # Send Response
591         response_json = {
592             "NewPosterIdx" : NewPosterIdx
593         }
594         json_string = json.dumps(response_json)
595         client.publish(SenderId + "/RunStart", json_string, 2)
596     # U2S(2) RunData/RunEnd : SenderId, RunIdx, End_Time, Dist_Achv, EndLat, EndLon
597     # S2U(2) SenderId/RunEnd : NewPosterIdx, Achv_Name[], Achv_Idx[], Achv_Disc[]
```

- Skeleton for Server is completed, and DB integration parts are marked for cooperation.
- Additional calls needed can easily be added to skeleton code.

# 5. Implementation - backend

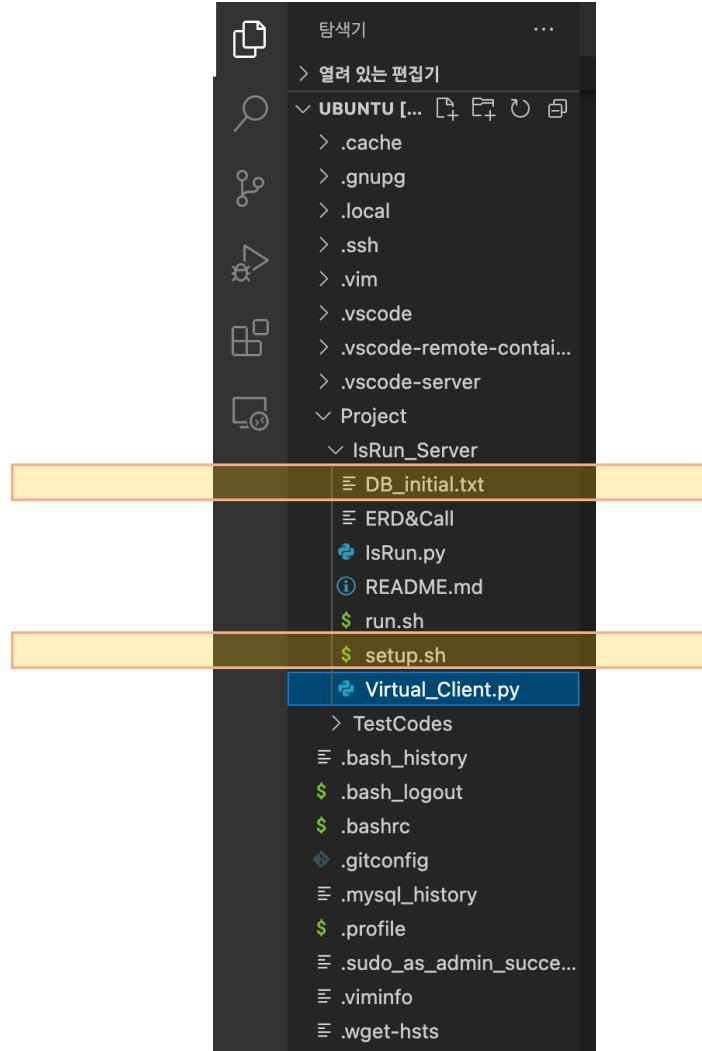


The image shows a screenshot of a code editor with two tabs open: `MQTT_Test_pub.py` and `Virtual_Client.py`. The `Virtual_Client.py` tab is active, displaying the following Python code:

```
Project > IsRun_Server > Virtual_Client.py
29     client.on_disconnect = on_disconnect
30     client.on_subscribe = on_subscribe
31     client.on_message = on_message
32
33     # HiveMQ를 사용합니다.
34     client.connect('broker.mqttdashboard.com', 1883)
35
36     userid = 'iqrnweriohandiogaih'
37     # 해당 토픽에 구독해서 메시지를 확인합니다.
38     client.subscribe("iqrnweriohandiogaih/#", 2)
39     client.loop_start()
40
41     # U2S(2) UserData/SignIn : SignInID, SignInPW
42     # S2U(2) SignInID/SignIn : Valid
43     response_json = {
44         "SignInID":userid,
45         "SignInPW":"asdf"
46     }
47     json_string = json.dumps(response_json)
48     client.publish("UserData/SignIn", json_string, 2)
49     input("Press Enter to Continue : ")
50     # U2S(2) UserData/SignUp : SignUpID, SignUpPW
51     # S2U(2) SignUpID/SignUp : Valid
52     response_json = {
53         "SignUpID":userid,
54         "SignUpPW":"asdf"
55     }
56     json_string = json.dumps(response_json)
57     client.publish("UserData/SignUp", json_string, 2)
58     input("Press Enter to Continue : ")
59     # U2S(2) UserData/UpdateUserData : SenderId, FoodCount, Gold, RunCount, MainCharacterIdx
60     # S2U(2) SenderId/UpdateUserData : Valid
61     response_json = {
62         "SenderId":userid,
63         "FoodCount" : 2022,
64         "Gold" : 1999,
65         "RunCount" : 17,
66         "MainCharacterIdx" : 3
67     }
68     json_string = json.dumps(response_json)
69     client.publish("UserData/UpdateUserData", json_string, 2)
70     input("Press Enter to Continue : ")
71     # U2S(2) UserData/GetUserData : SenderId
72     # S2U(2) SenderId/GetUserData : FoodCount, Gold, RunCount, MainCharacterIdx
73     response_json = {
74         "SenderId":userid
75     }
76     json_string = json.dumps(response_json)
77     client.publish("UserData/GetUserData", json_string, 2)
78     input("Press Enter to Continue : ")
79     # U2S(2) UserData/UpdateUserInfo : SenderId, Weight, Height, Gender, Age
80     # S2U(2) SenderId/UpdateUserInfo : Valid
81     response_json = {
82         "SenderId":userid,
83         "Weight" : 82,
84         "Height" : 185,
85         "Gender" : "M",
86         "Age" : 32
87     }
88     json_string = json.dumps(response_json)
89     client.publish("UserData/UpdateUserInfo", json_string, 2)
```

- Virtual Client that simulate client action is made to test call and response from server.
- VC can be used to verify call and response, and to test server endurance.
- RunningData stream publish rate should be tested to reduce bottleneck and packet loss.

# 5. Implementation - backend



- Env, and DB setup method is saved in `setup.sh`, `DB_initial.txt` to reduce inconvenience between changing development environment.

# 6. Challenges - frontend

- Without Fragmentation :

Code was too dirty. Main activity over 600 lines.

► Fragments, MVVM Design Pattern

- Mac vs Window

Mac had errors with build and run

► Clear build & Rebuild and Clear Invalid Caches  
or use Git for running on Window

# 6. Challenges - frontend

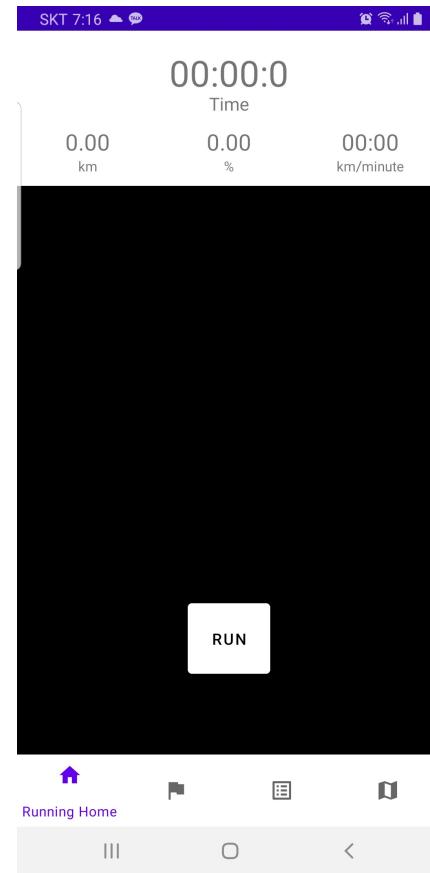
- Kakao map API :

Error code says hash-key match error,  
but same debug-hash-key.

Error page ➤

► Need to find solution.

Error code



```
2022-04-26 04:47:35.396 31857-31884/edu.skku.cs.isrun D/mali_winsys: EGLint new_window_surface(egl_winsys_display *, void *, EGLSurface, EGLConfig, egl_winsys_surface **, EGLBoolean) returns 0x3000
2022-04-26 04:47:35.397 31857-31884/edu.skku.cs.isrun W/libEGL: EGLNativeWindowType 0xe727f008 disconnect failed
```

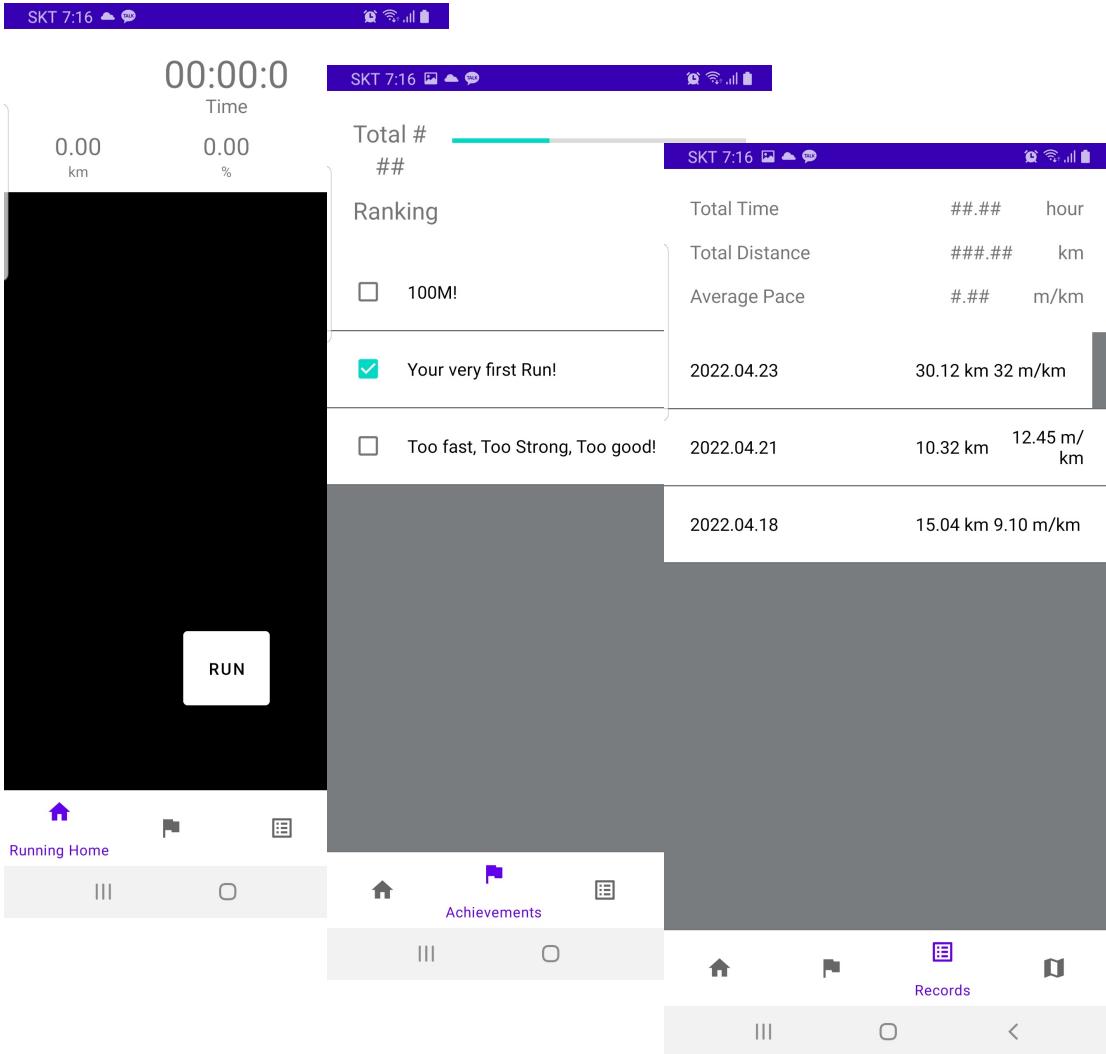
# 6. Challenges - backend

Tradeoffs in performance on EC2's limited hardware

1. Load on broker vs. Load on server (Topic wise)
2. Fast and accurate data rate vs. slow and stable data rate
3. Multithreading vs. Multiprocessing

► Finding Solution Using Virtual Client.

# 7. Limitation - frontend



- Limitation on implementing Figma Design!!

► Need to give up some details...

# 7. Limitation - backend

## MQTT 서비스 품질(QoS) 옵션

AWS IoT 및 AWS IoT Device SDK는 MQTT 서비스 품질(QoS) 수준 0과 1을 지원합니다. MQTT 프로토콜은 세 번째 수준의 QoS, 즉 2를 정의하지만 AWS IoT는 지원하지 않습니다. MQTT 프로토콜만 QoS 기능을 지원합니다. HTTPS는 값이 0 또는 1일 수 있는 쿼리 문자열 파라미터 ?qos=qos를 전달하여 QoS를 지원합니다.

이 표에서는 각 QoS 수준이 메시지 브로커에 게시된 메시지에 미치는 영향을 설명합니다.

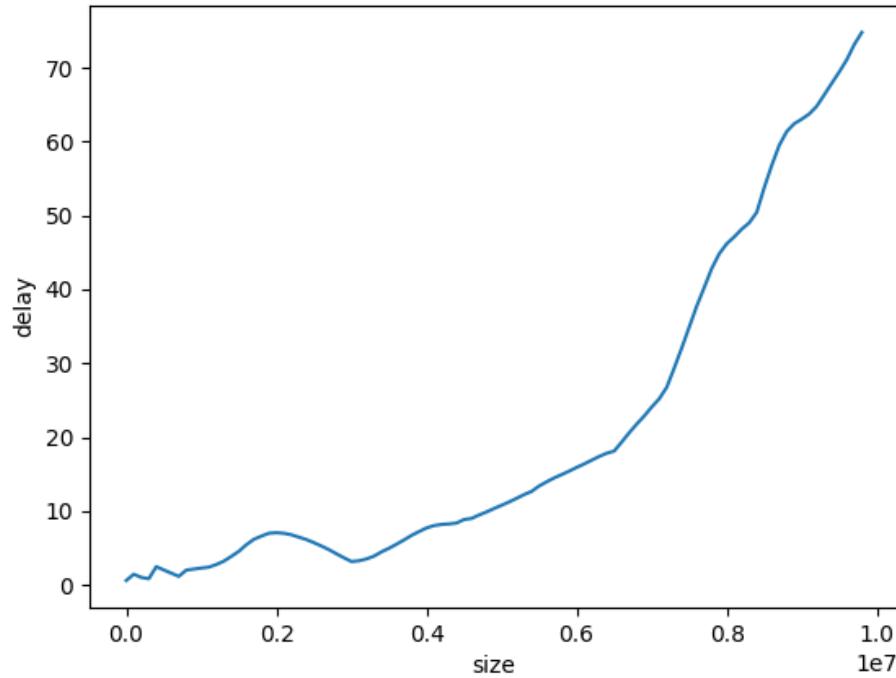
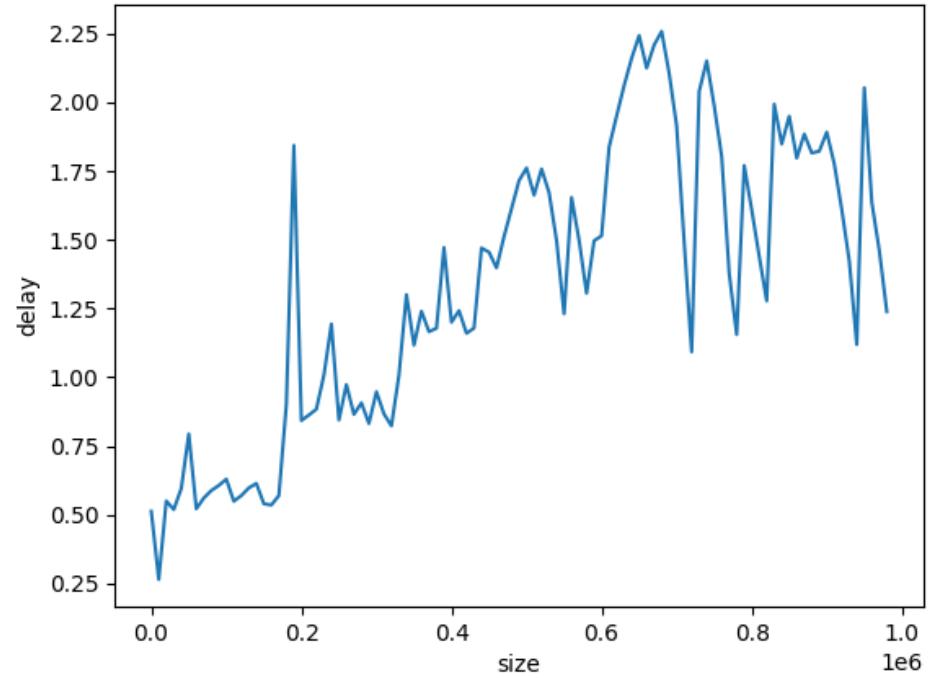
QoS 수준	메시지	설명
QoS 수준 0	0회 이상 전송됨	이 수준은 신뢰할 수 있는 통신 링크를 통해 전송되거나 누락되어도 문제 없는 메시지에 사용해야 합니다.
QoS 수준 1	한 번 이상 전송한 다음 PUBACK 응답이 수신될 때까지 반복적으로 전송합니다.	전송한 사람이 PUBACK 응답을 수신하여 성공적인 전달을 나타낼 때까지 메시지는 완료되지 않은 것으로 간주됩니다.

QoS 수준	Observations	Bevywise MQTT Route 2.0	Mosquitto 1.4.15	ActiveMQ 5.15.8	HiveMQ CE 2020.2
QoS 0	Message rate(msgs/sec): Average CPU usage(%): @above message rate: Average latency (ms): Projected message rate @100% CPU usage:	32839 97.93 1.137 33533	32,016 84.29 1.655 37,983	573 110.44 1.508 518.8	249 96.68 2.74 257.5
QoS 1	Message rate(msgs/sec): Average CPU usage(%): @above message rate: Average latency (ms): Projected message rate @100% CPU usage:	3542.49 95.79 0.96 3697.67	9488 89 0.742 10660	363 108.82 1.782 333.57	118 104.16 3.062 113.28
QoS 2	Message rate(msgs/sec): Average CPU usage(%): @above message rate: Average latency (ms): Projected message rate @100% CPU usage:	2649 98.202 1.534 2697.5	6585 96.73 1.383 6807.6	293 104.36 2.148 280	99 102.28 3.665 96.7

AWS\_IOT does not support QoS(2)

► MQTT broker : Mosquitto

# 7. Limitation - backend



Data within about 0.4 MB (400,000 chars) is suitable for packet delivery within 1 second.

# 8. To-Dos

에피	5월
▶ EP22-6 Overall: Subject Proposal 완료됨	
▶ EP22-4 Overall: Latex Compose 완료됨	
▶ CP22-8 Overall: Integration Test/Tuning	
▶ CP22-7 Overall: Final Report	
▶ EP22-3 App: Layout Design 완료됨	
▶ CP22-9 App: Skeleton App Design	
▶ CP22-10 App: Running Elements	
▶ CP22-11 App: Game Elements	
▶ EP22-12 Server: DB Design 완료됨	
▶ CP22-13 Server: DB Setting	
▶ CP22-15 Server: AWS	
▶ CP22-14 Server: Paho-MQ Server	
+ 만들기 에피	

## Front

- Views
- Actions

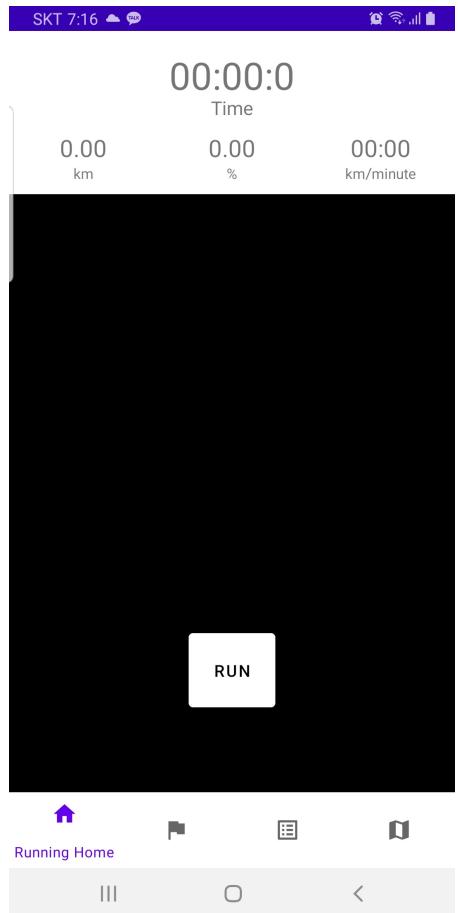
## Back

- Sync MQTT and DB

## Front & Back

- Kakao Map API
- Connect to Server

# 9. Demo



# Source

Jira: <https://capstone-2022-1.atlassian.net/jira/software/projects/CP22/boards/2/roadmap>

Figma: <https://www.figma.com/file/i9uwJMd8i3CzjL4CKxHGPO/Untitled?node-id=0%3A1>

CheeseFarm: <https://bluesoccer.net/5017>

Nike Running App: [https://www.nike.com/kr/ko\\_kr/c/nike-plus/running-app-gps/new](https://www.nike.com/kr/ko_kr/c/nike-plus/running-app-gps/new)

Pokemon Go: <https://www.instiz.net/name/45939585>

The Thumb Zone: <https://refreshstudio.tistory.com/entry/%EC%97%84%EC%A7%80%EC%98%81%EC%97%ADthe-Thumb-Zone-%EB%AA%A8%EB%B0%94%EC%9D%BC-%EC%82%AC%EC%9A%A9%EC%9E%90%EB%A5%BC-%EC%9C%84%ED%95%9C-%EB%94%94%EC%9E%90%EC%9D%B8>

Android Studio Kotlin: <https://www.pngegg.com/ko/png-xytvm>

MVVM Design Pattern: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel>

Android Fragments: <https://www.geeksforgeeks.org/difference-between-a-fragment-and-an-activity-in-android/>

Android Navigation: <https://brunch.co.kr/@oemilk/210>

AWS: <https://velog.io/@nari120/AWS-%EC%A3%BC%EC%9A%94-%EC%84%9C%EB%B9%84%EC%8A%A4>

MQTT: <https://ko.m.wikipedia.org/wiki/%ED%8C%8C%EC%9D%BC:Mqtt-hor.svg>

MySQL: <https://www.pngwing.com/ko/free-png-njuyc>

Python: <https://velog.io/@taeil77/%ED%8C%8C%EC%9D%B4%EC%8D%ACPython-Function>

# Q&A

마음대로 질문해 주세요!

