Midtem Presentation

Capstone Design Project

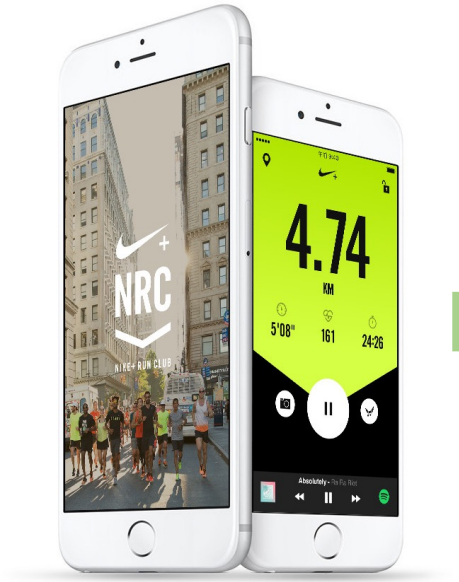
이상해 C

강승목, 김준석, 이혜원

Index

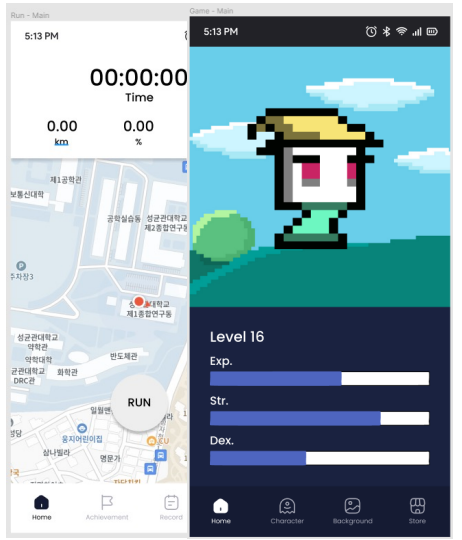
1. Objective
2. Milestone
3. Role of each member
4. Design
5. Implementation
6. Challenges
7. Limitation
8. To-Dos
9. Demo

1. Objective



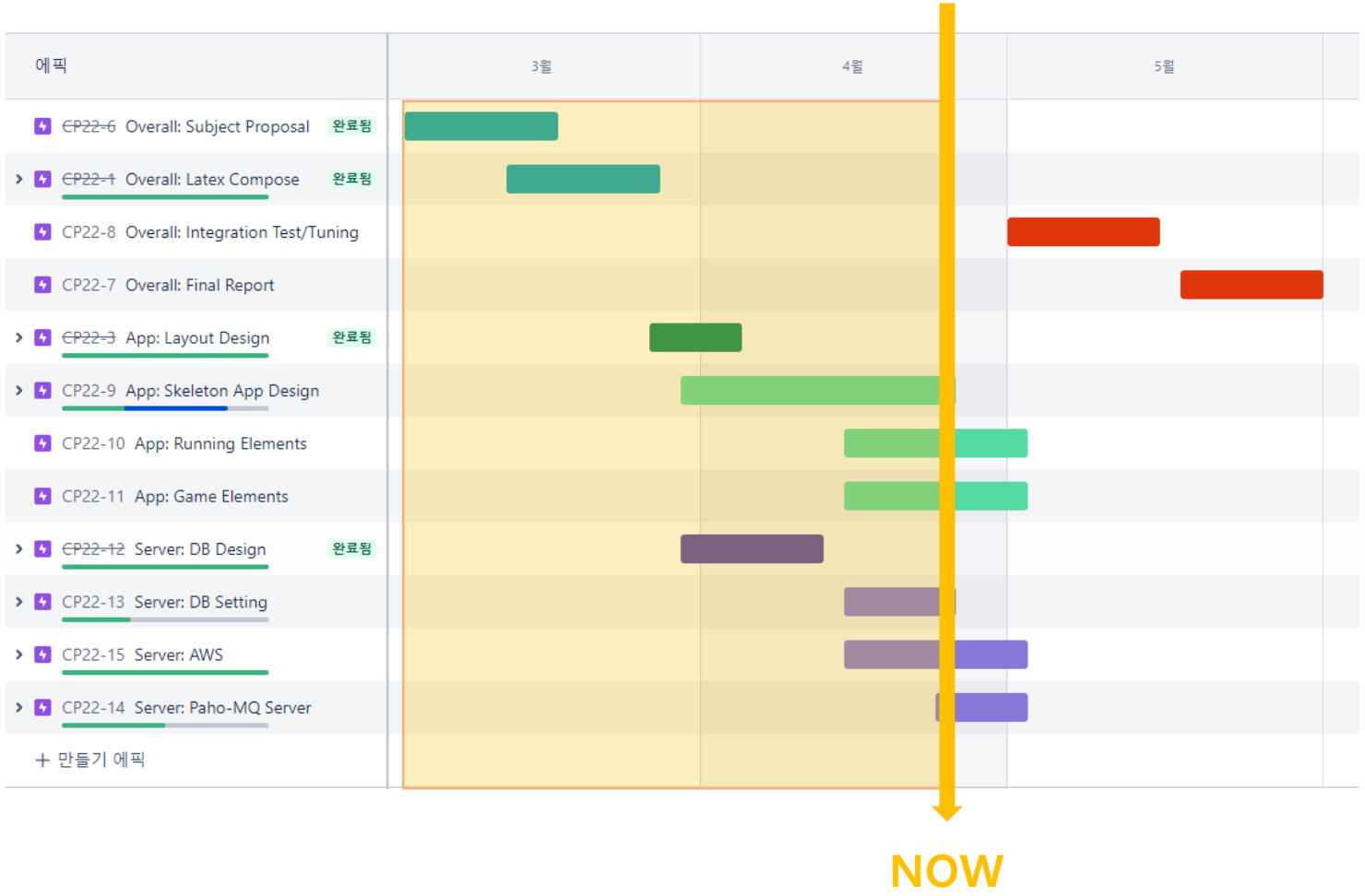
Running App

Virtual Character
Simulation Game



Running App
for Beginners

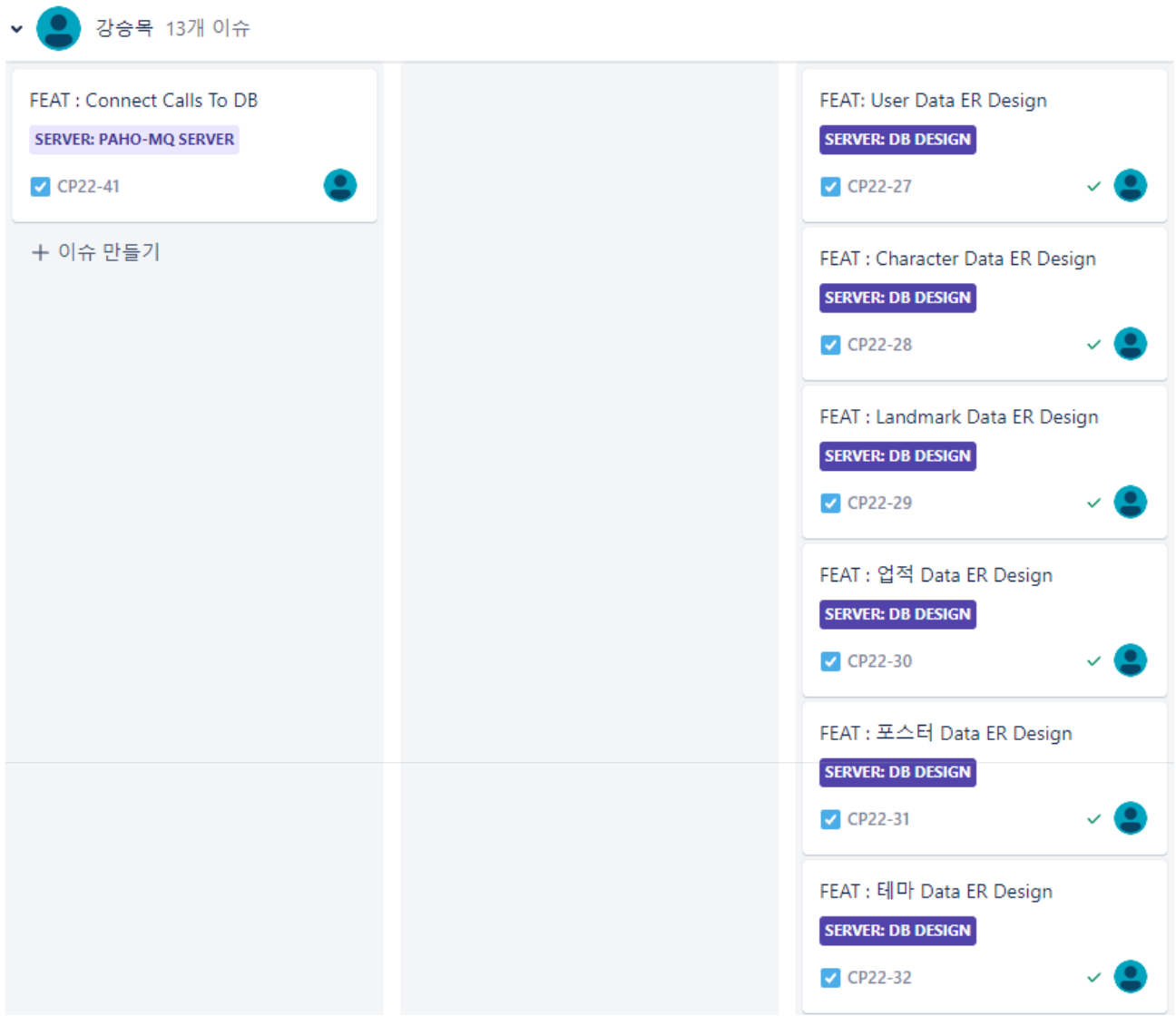
2. Milestone


에픽
3월
4월
5월
~~CP22-6~~ Overall: Subject Proposal 완료됨
~~CP22-1~~ Overall: Latex Compose 완료됨
CP22-8 Overall: Integration Test/Tuning
CP22-7 Overall: Final Report
~~CP22-3~~ App: Layout Design 완료됨
CP22-9 App: Skeleton App Design
CP22-10 App: Running Elements
CP22-11 App: Game Elements
~~CP22-12~~ Server: DB Design 완료됨
CP22-13 Server: DB Setting
CP22-15 Server: AWS
CP22-14 Server: Paho-MQ Server
+ 만들기 에픽
NOW

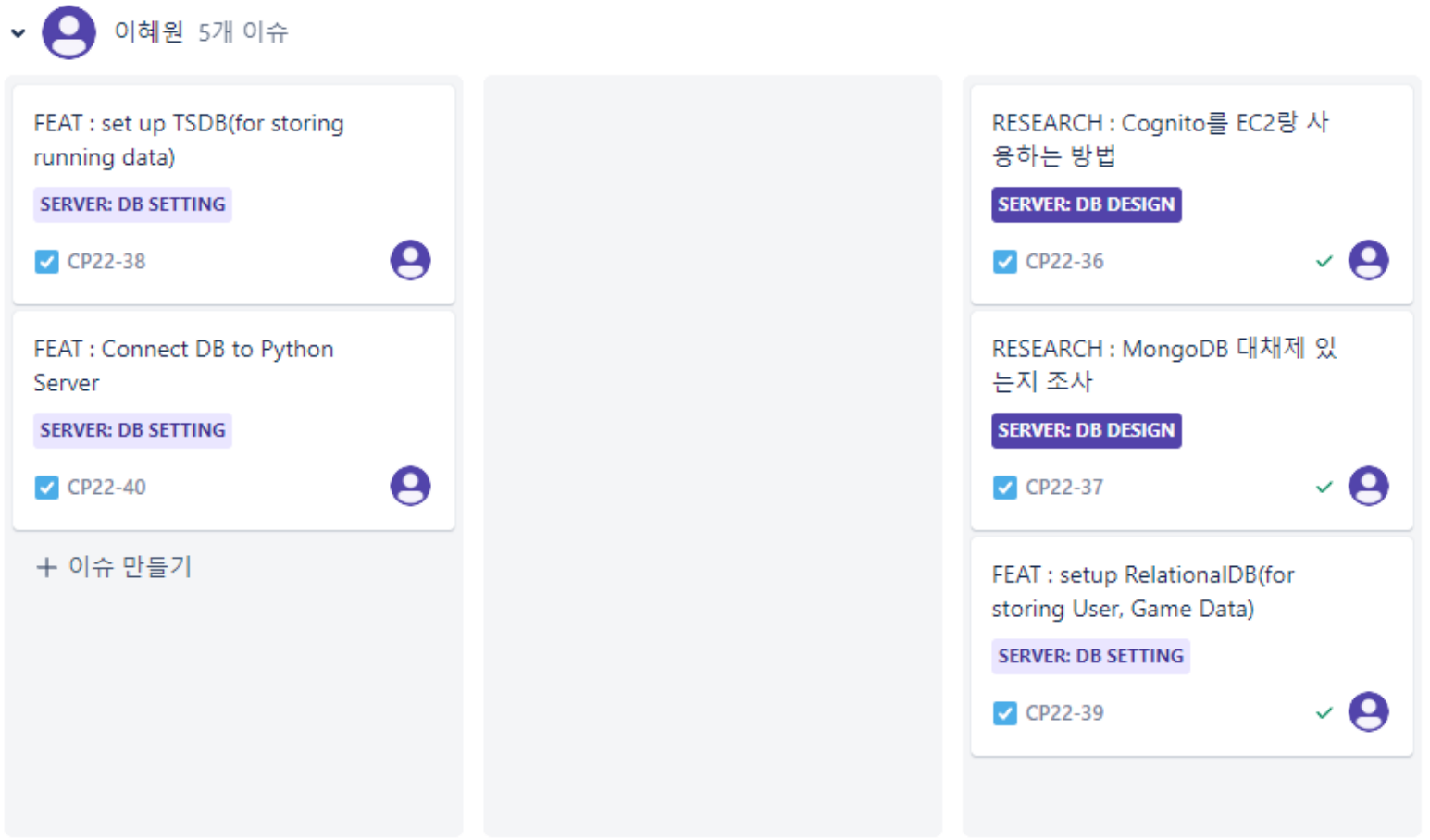
3. Role of each member

Seungmok Kang	Hyewon Lee	Junsoek Kim	Jaeyoon Park
Backend • ER Diagram > DB • AWS Server		**Frontend** • Skeleton Code of Design • Skeleton Code of Flow Chart	
Frontend & Backend • Connecting Frontend with Backend • Communication between Server-side and Client-side			

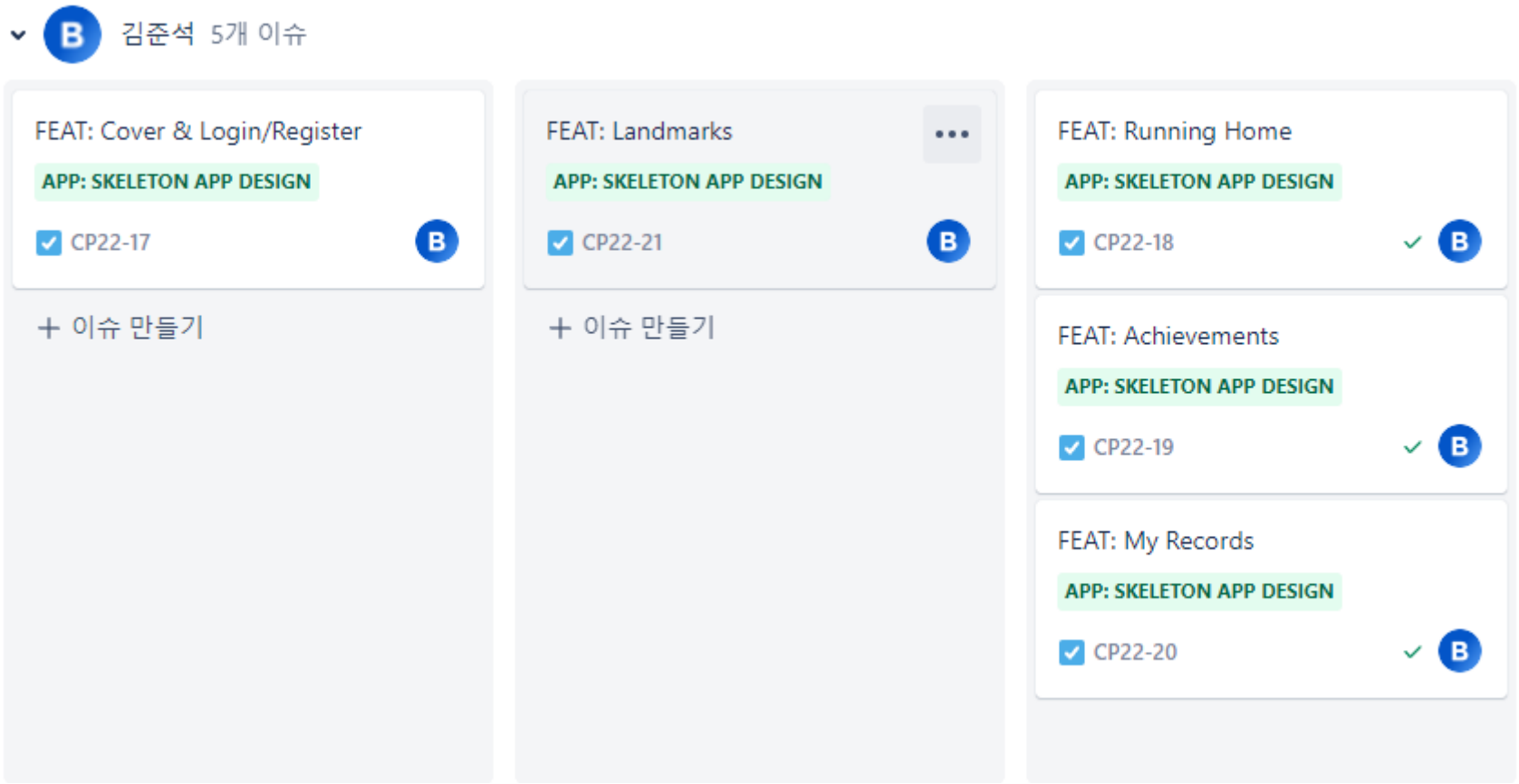
3. Role of each member



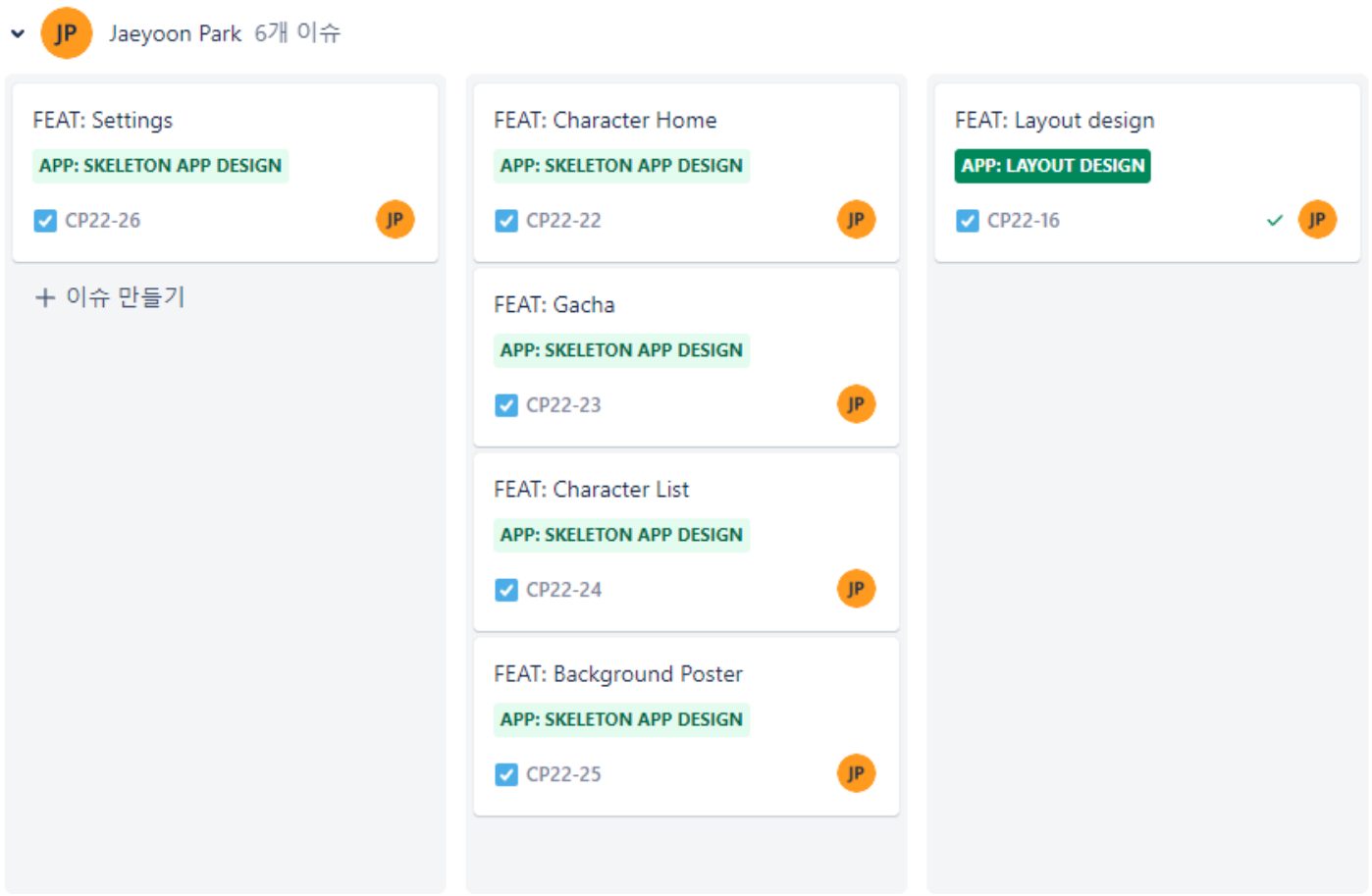
3. Role of each member



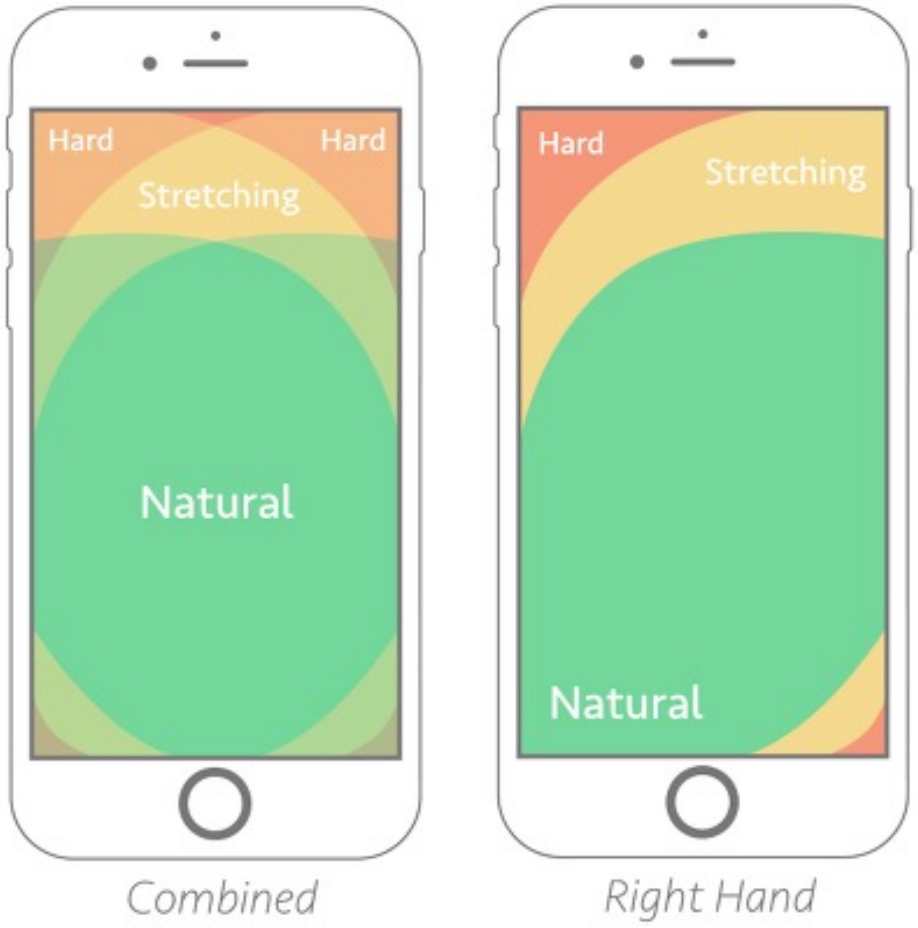
3. Role of each member



3. Role of each member



4. Design - UX/UI

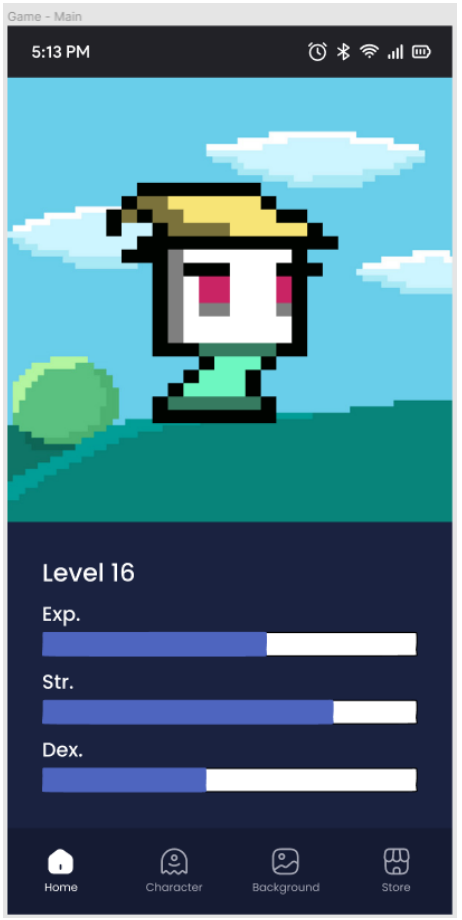


The Thumb Zone

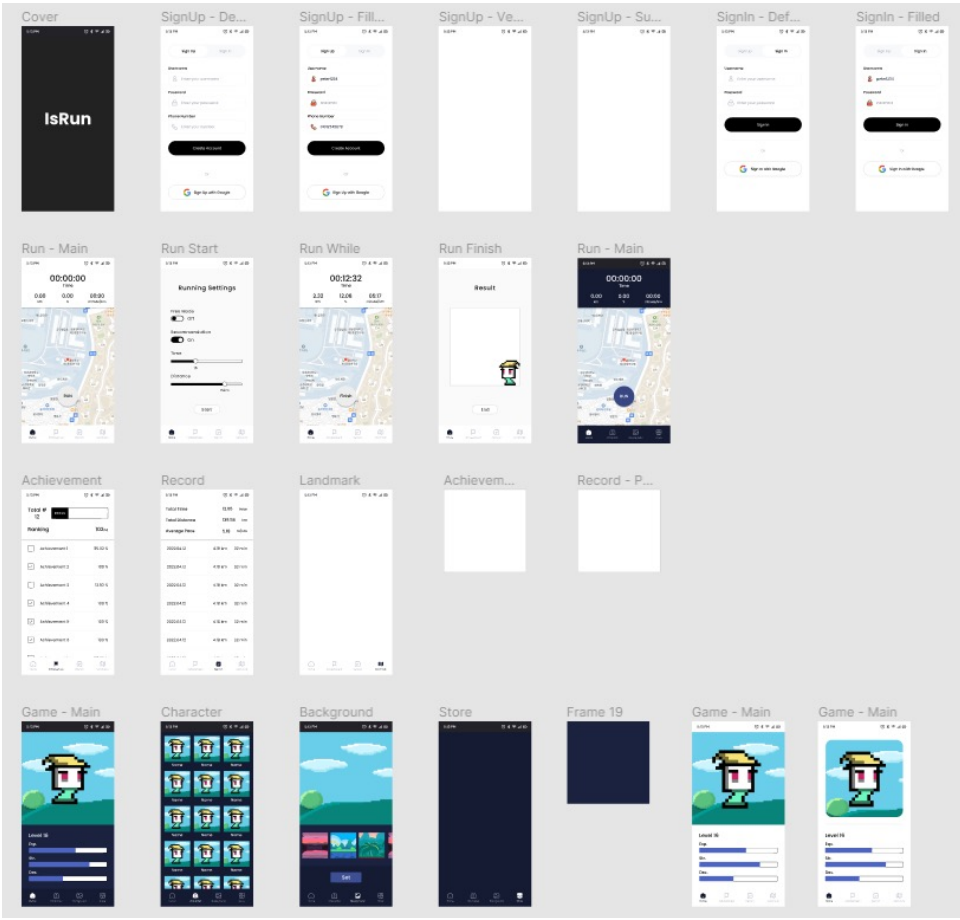
- Important links
  - easy-to-reach zone
- Unimportant links
  - hard-to-reach zones

4. Design - UX/UI

Running



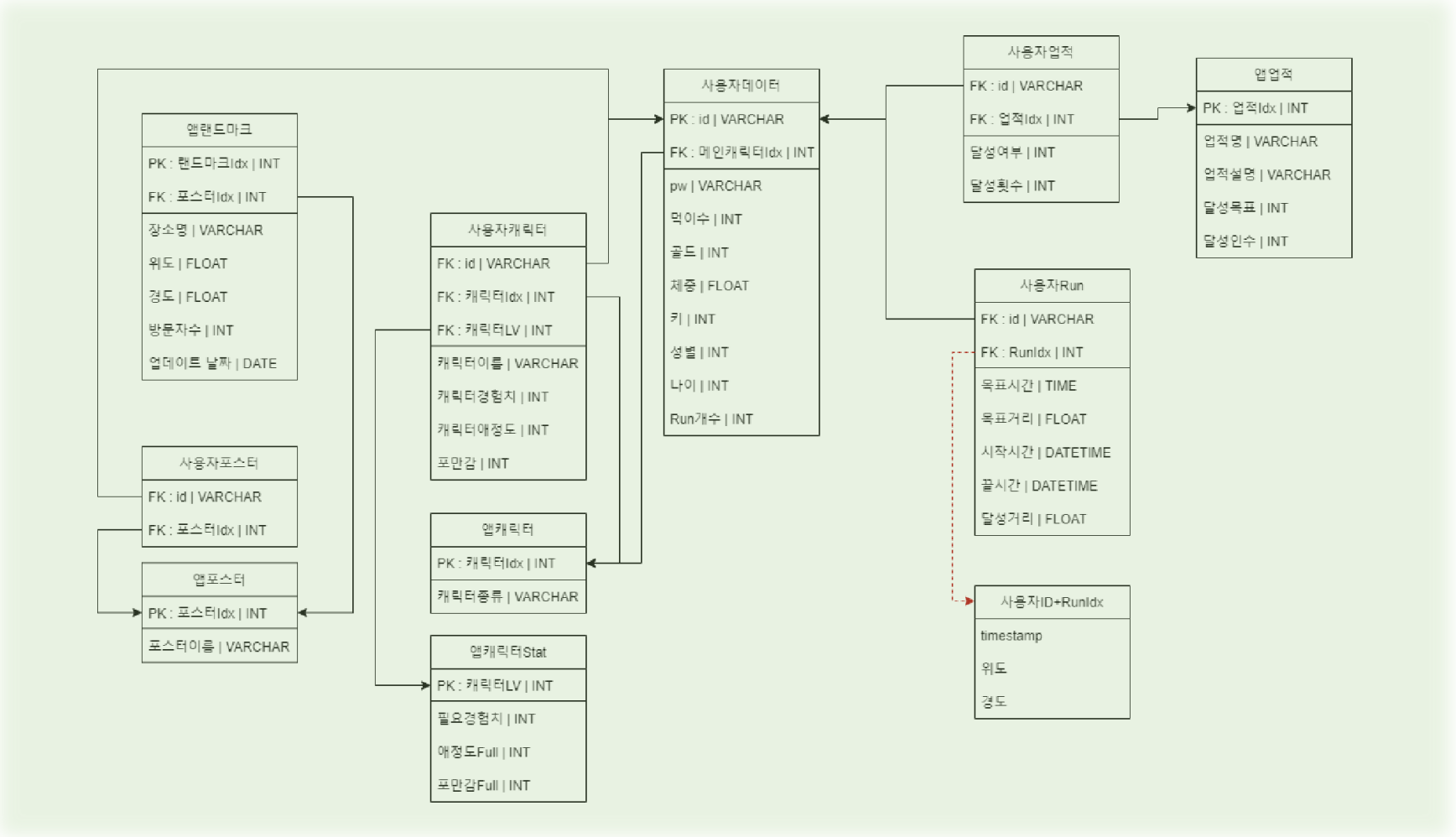
Gaming



Overall

4. Design - Server

4. Design - DB


앱랜드마크
PK : 랜드마크Idx | INT
FK : 포스터Idx | INT
장소명 | VARCHAR
위도 | FLOAT
경도 | FLOAT
방문자수 | INT
업데이트 날짜 | DATE
사용자포스터
FK : id | VARCHAR
FK : 포스터Idx | INT
앱포스터
PK : 포스터Idx | INT
포스터이름 | VARCHAR
사용자캐릭터
FK : id | VARCHAR
FK : 캐릭터Idx | INT
FK : 캐릭터LV | INT
캐릭터이름 | VARCHAR
캐릭터경험치 | INT
캐릭터애정도 | INT
포만감 | INT
앱캐릭터
PK : 캐릭터Idx | INT
캐릭터종류 | VARCHAR
앱캐릭터Stat
PK : 캐릭터LV | INT
필요경험치 | INT
애정도Full | INT
포만감Full | INT
사용자데이터
PK : id | VARCHAR
FK : 메인캐릭터Idx | INT
pw | VARCHAR
먹이수 | INT
골드 | INT
체중 | FLOAT
키 | INT
성별 | INT
나이 | INT
Run개수 | INT
사용자업적
FK : id | VARCHAR
FK : 업적Idx | INT
달성여부 | INT
달성횟수 | INT
앱업적
PK : 업적Idx | INT
업적명 | VARCHAR
업적설명 | VARCHAR
달성목표 | INT
달성인수 | INT
사용자Run
FK : id | VARCHAR
FK : RunIdx | INT
목표시간 | TIME
목표거리 | FLOAT
시작시간 | DATETIME
끝시간 | DATETIME
달성거리 | FLOAT
사용자ID+RunIdx
timestamp
위도
경도

5. Implementation – frontend

IDE

- Android studio

Language

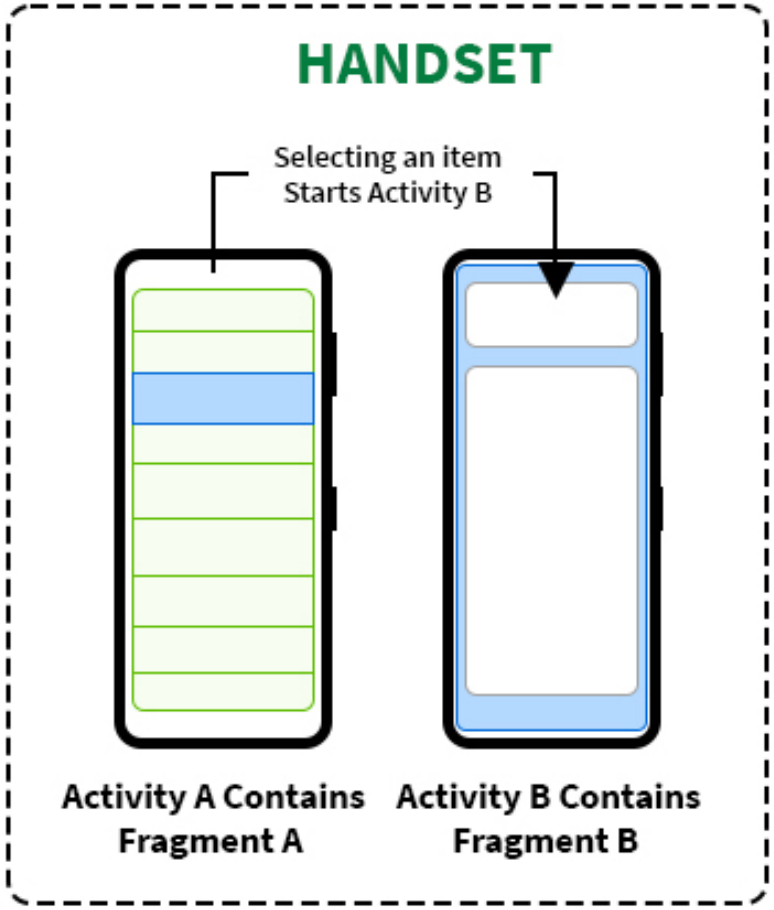
- Kotlin

Tools

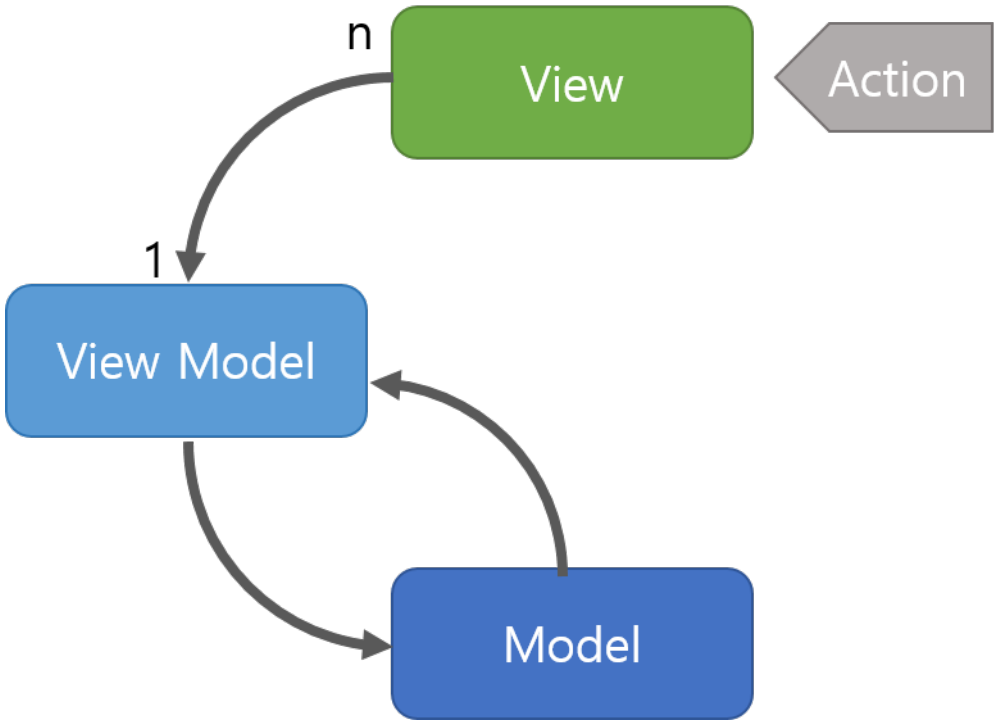
- Fragment
- MVVM
- Navigation

5. Implementation - frontend

- Fragments -

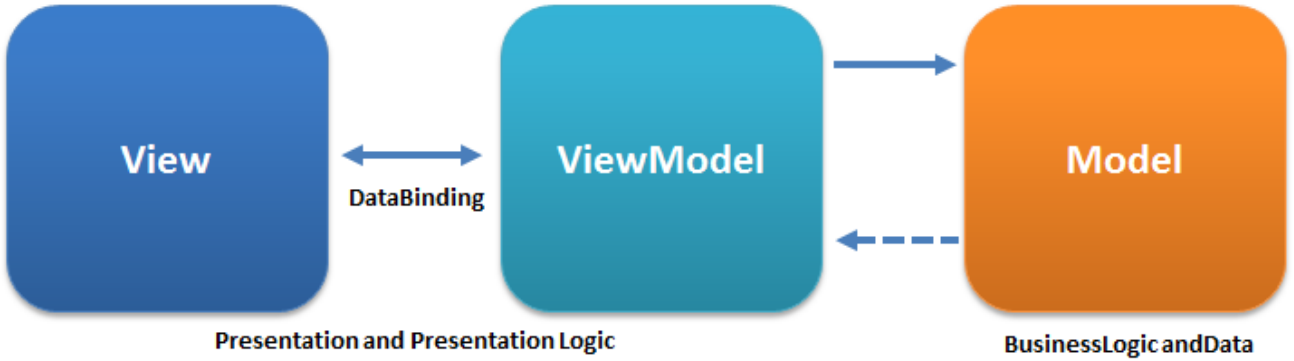


- MVVM Design Pattern -



5. Implementation - frontend

- MVVM Design Pattern -



Characteristic

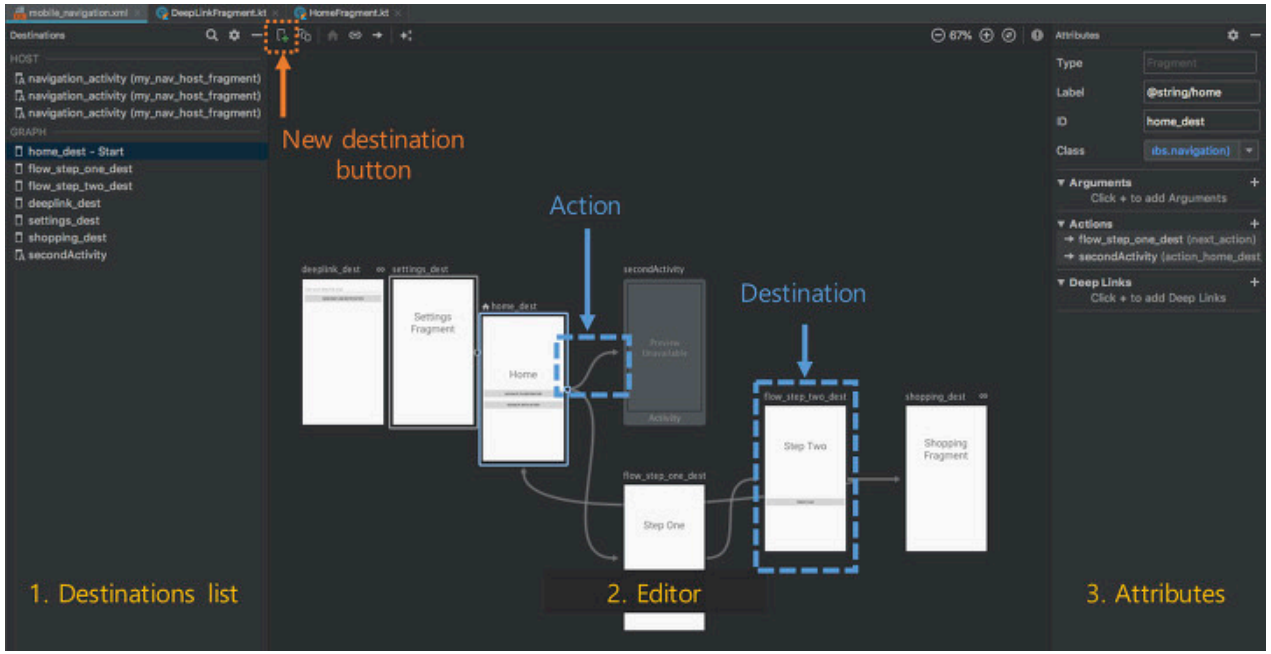
- View Model(1) : View(n)
- Command & Data Binding Pattern
  ➡ No Dependency between View and View Model

Pros & Cons

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

5. Implementation - frontend

- Navigation -



- NavHostFragment
- NavController
- Destination
- Action

5. Implementation - backend

Server Environment

- AWS EC2

Language

- Python

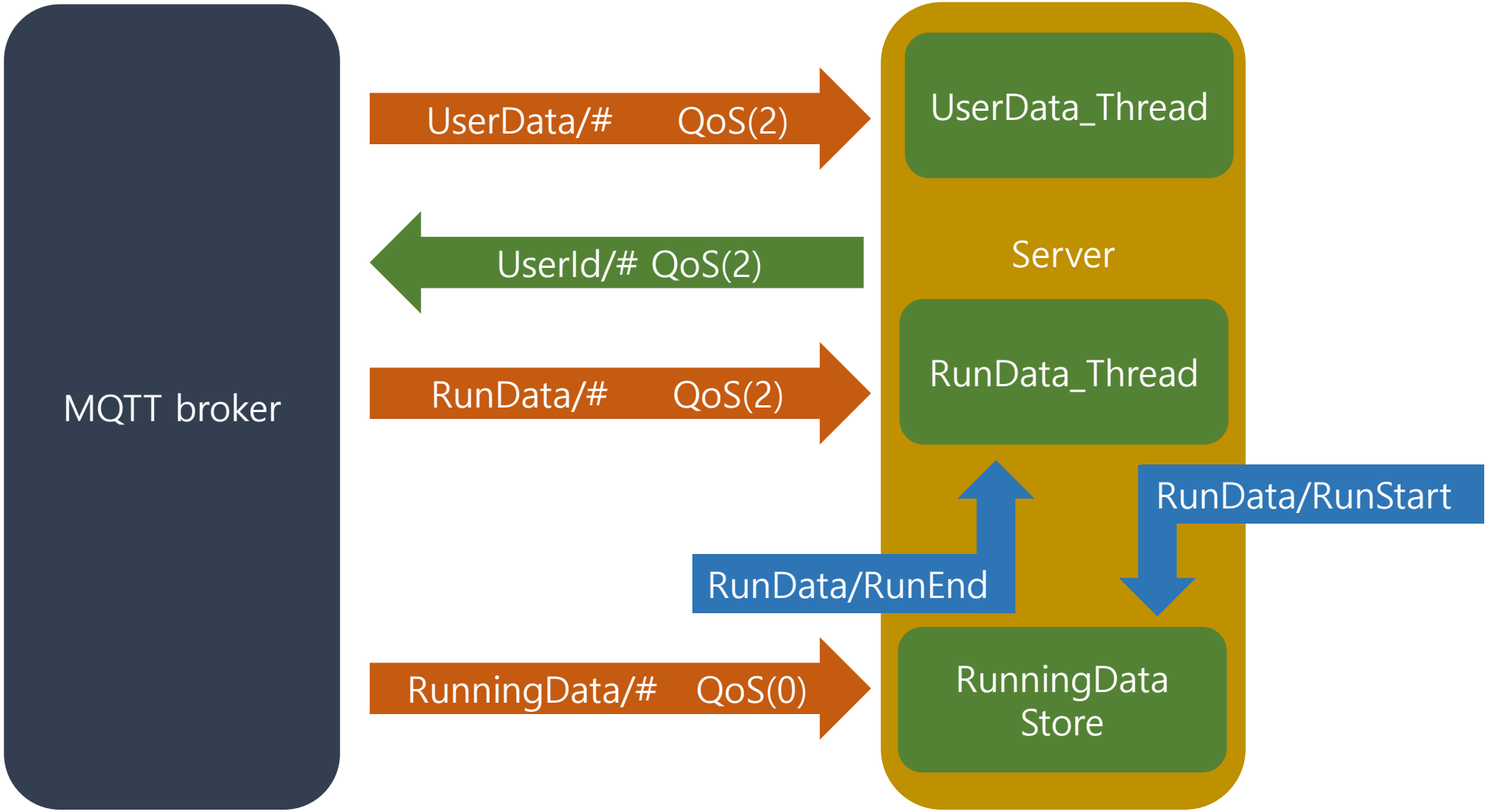
DB

- MySQL & InfluxDB

Protocol

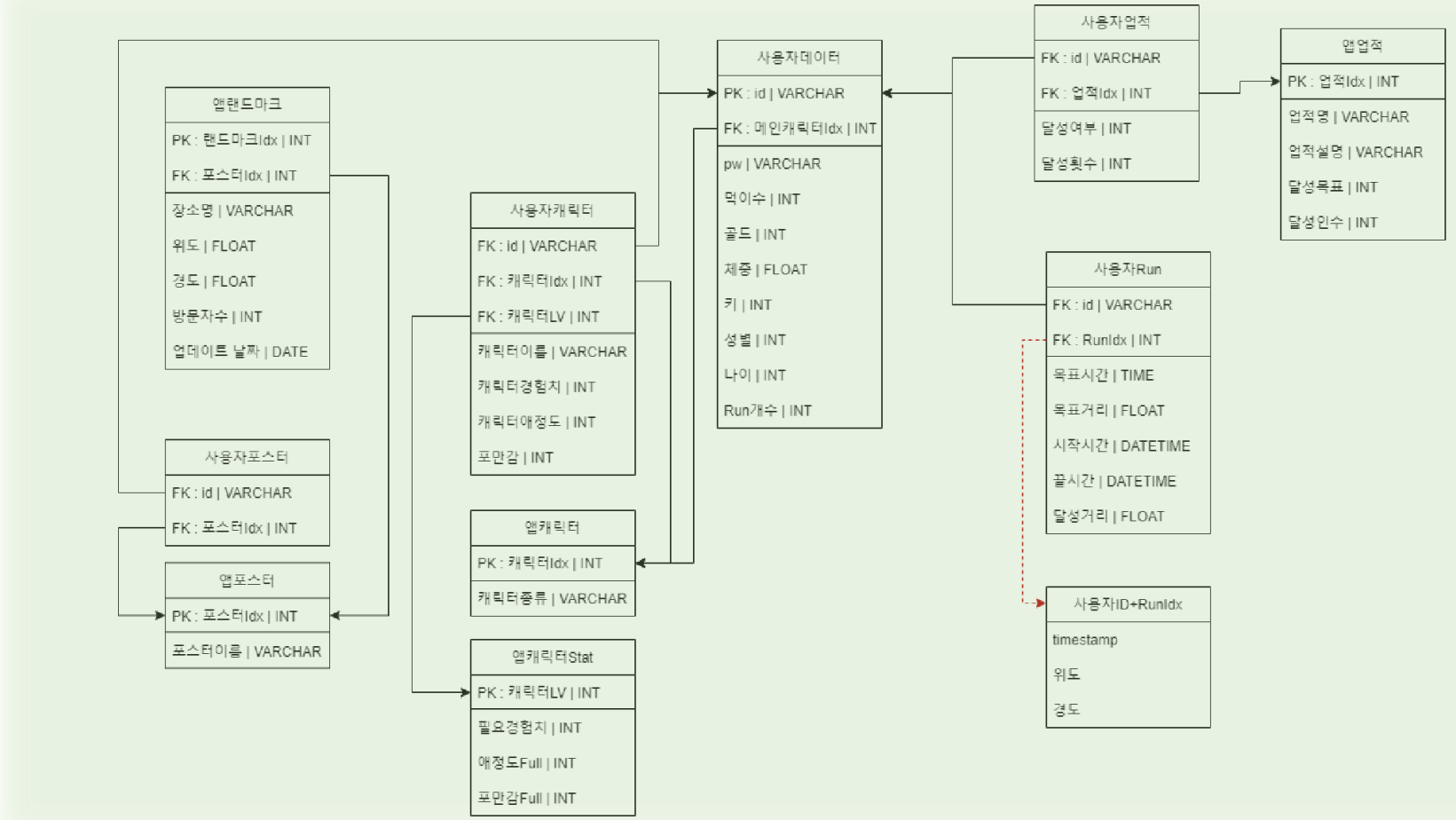
- MQTT

5. Implementation - backend

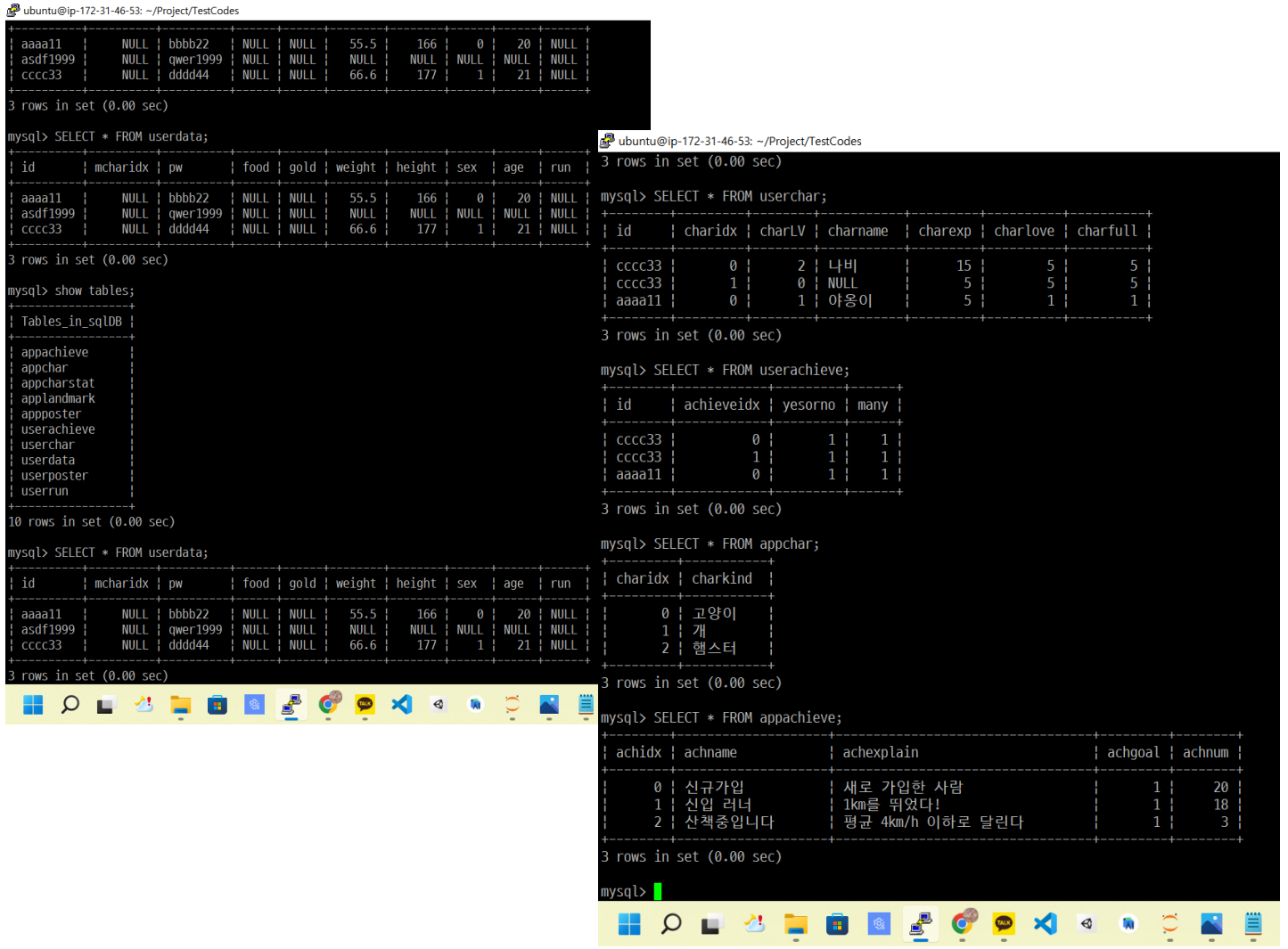


5. Implementation - backend

DB(Data Base)



5. Implementation - backend



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/GetAchives", GetAchives_message)
533      UserData_Client.message_callback_add("UserData/GetRuns", GetRuns_message)
534      UserData_Client.message_callback_add("UserData/GetLandRecommand", GetLandRecommand_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
 0
```

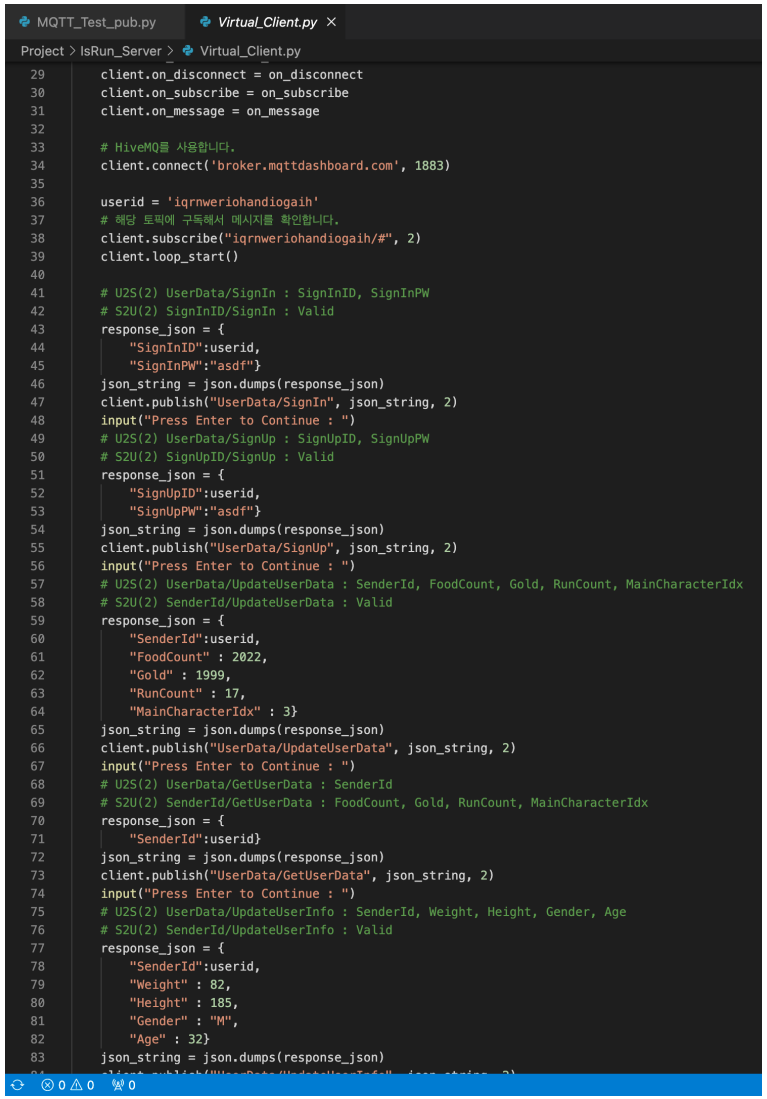
- Data Parsing for Predicted Calls
- e.g.
  UserData/SignIn
  UserData/GetRuns
  etc.

5. Implementation - backend

```python
# U2S(2) RunData/RunStart  : SenderId, RunIdx, Time_Goal, Dist_Goal, Start_Time, StartLat, StartLon
# S2U(2) SenderId/RunStart : NewPosterIdx
def RunStart_message(client, userdata, msg):
    # Parse Data
    json_data = json.loads(str(msg.payload.decode("utf-8")))
    print(json_data)
    SenderId = None
    RunIdx = None
    Time_Goal = None
    Dist_Goal = None
    Start_Time = None
    StartLat = None
    StartLon = None
    if "SenderId" in json_data.keys():
        SenderId = json_data["SenderId"]
    if "RunIdx" in json_data.keys():
        RunIdx = json_data["RunIdx"]
    if "Time_Goal" in json_data.keys():
        Time_Goal = json_data["Time_Goal"]
    if "Dist_Goal" in json_data.keys():
        Dist_Goal = json_data["Dist_Goal"]
    if "Start_Time" in json_data.keys():
        Start_Time = json_data["Start_Time"]
    if "StartLat" in json_data.keys():
        StartLat = json_data["StartLat"]
    if "StartLon" in json_data.keys():
        StartLon = json_data["StartLon"]
    # Send Response
    if None in [SenderId, RunIdx, Time_Goal, Dist_Goal, Start_Time, StartLat, StartLon]:
        print("ERROR RunStart")
    else:
        client.message_callback_add("RunningData/{}/{}".format(SenderId, RunIdx), RunningData_message)
        #======================Handle DB=======================
        # 위치 기반으로 어떤 랜드마크 보상 가능한지 확인할것.
        # 그냥 가장 가까운거 아무거나 하나 줘버릴까도 생각중.
        # UserData 테이블에서 RunCount++ 할것.
        # UserRun 테이블에 Time_Goal, Dist_Goal, Start_Time등록할것.
        # SenderID+RunIdx로 influxDB 테이블 생성할것.
        NewPosterIdx = None
        #======================================================
        # Send Response
        response_json = {
            "NewPosterIdx" : NewPosterIdx
        }
        json_string = json.dumps(response_json)
        client.publish(SenderId + "/RunStart", json_string, 2)
# U2S(2) RunData/RunEnd  : SenderId, RunIdx, End_Time, Dist_Achv, EndLat, EndLon
# S2U(2) SenderId/RunEnd : NewPosterIdx, Achv_Name[], Achv_Idx[], Achv_Disc[]
```

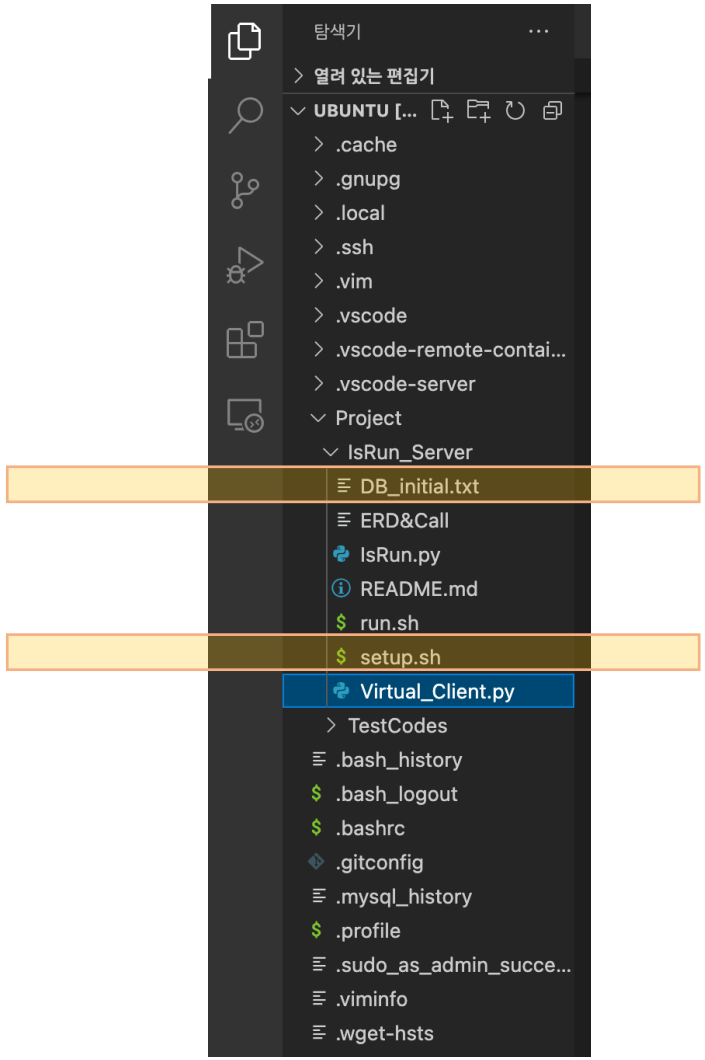
- Skeleton for Server is completed, and DB integration parts are marked for cooperation.
- Additional calls needed can easily be added to skeleton code.

5. Implementation - backend



- 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

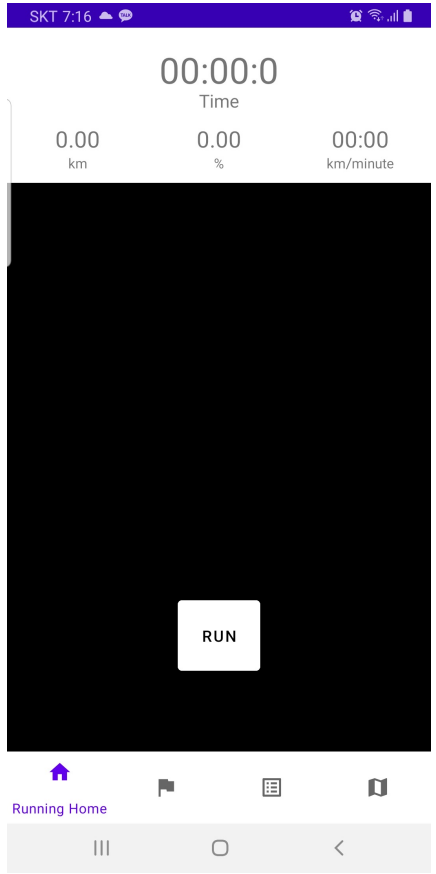
6. Challenges - frontend

- 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
```

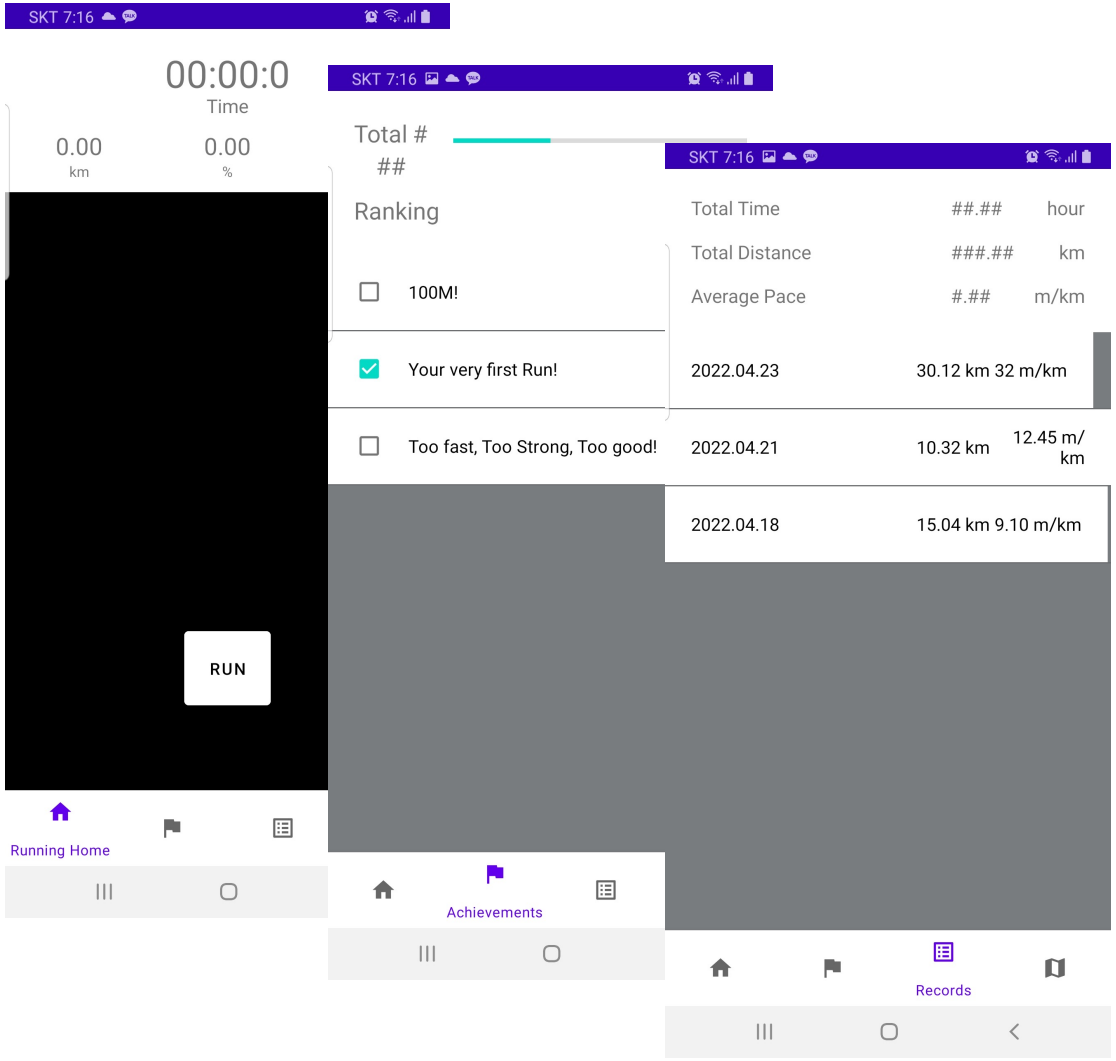
6. Challenges - backend

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 `0`과 `1`을 지원합니다. MQTT 프로토콜은 세 번째 수준의 QoS, 즉 `2`를 정의하지만 AWS IoT는 지원하지 않습니다. MQTT 프로토콜만 QoS 기능을 지원합니다. HTTPS는 값이 0 또는 1일 수 있는 쿼리 문자열 파라미터 `?qos=qos`를 전달하여 QoS를 지원합니다.

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

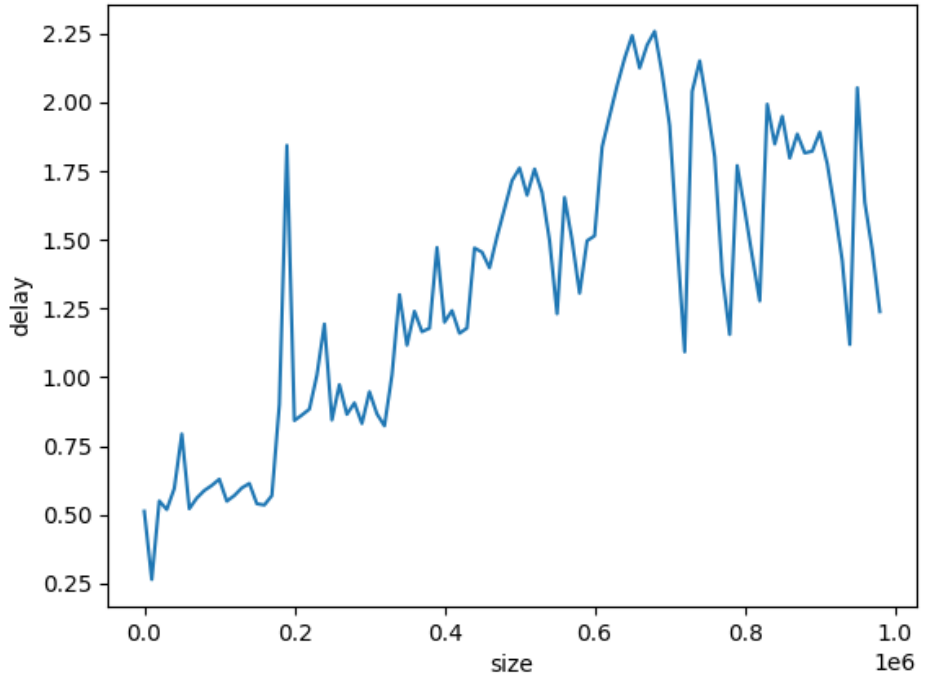
QoS 수준	메시지	설명
QoS 수준 0	0회 이상 전송됨	이 수준은 신뢰할 수 있는 통신 링크를 통해 전송되거나 누락되어도 문제 없는 메시지에 사용해야 합니다.
QoS 수준 1	한 번 이상 전송한 다음 `PUBACK` 응답이 수신될 때까지 반복적으로 전송합니다.	전송한 사람이 `PUBACK` 응답을 수신하여 성공적인 전달을 나타낼 때까지 메시지는 완료되지 않은 것으로 간주됩니다.

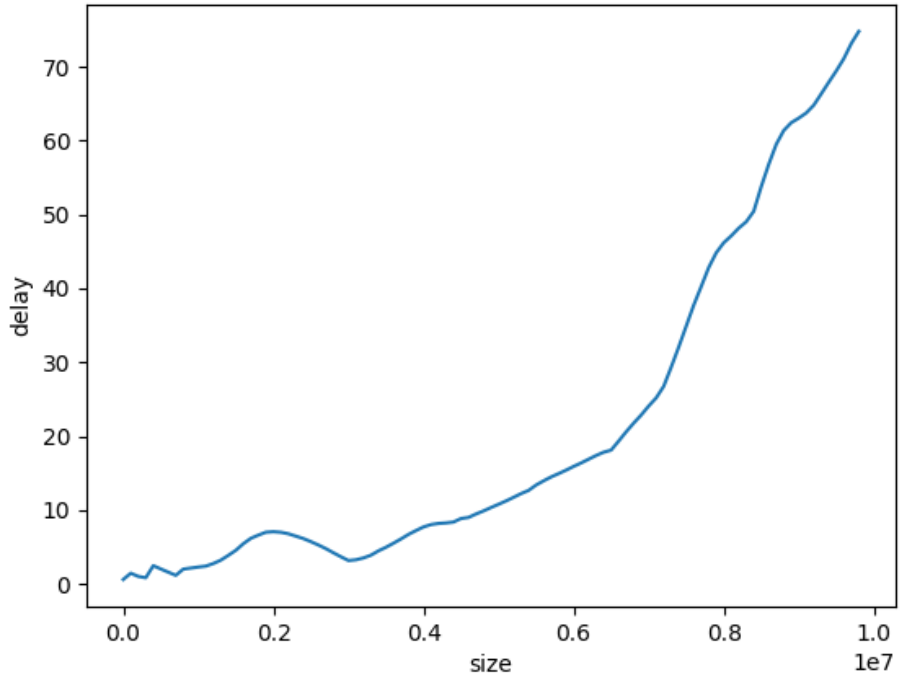
QoSes	Observations	Bevywise MQTT Route 2.0	Mosquitto 1.4.15	ActiveMQ 5.15.8	HiveMQ CE 2020.2
QoS0	Message rate(msgs/sec):	32839	32,016	573	249
	Average CPU usage(%) @above message rate:	97.93	84.29	110.44	96.68
	Average latency (ms):	1.137	1.655	1.508	2.74
	Projected message rate @100% CPU usage:	33533	37,983	518.8	257.5
QoS1	Message rate(msgs/sec):	3542.49	9488	363	118
	Average CPU usage(%) @above message rate:	95.79	89	108.82	104.16
	Average latency (ms):	0.96	0.742	1.782	3.062
	Projected message rate @100% CPU usage:	3697.67	10660	333.57	113.28
QoS2	Message rate(msgs/sec):	2649	6585	293	99
	Average CPU usage(%) @above message rate:	98.202	96.73	104.36	102.28
	Average latency (ms):	1.534	1.383	2.148	3.665
	Projected message rate @100% CPU usage:	2697.5	6807.6	280	96.7

AWS_IOT does not support QoS(2)

▶ MQTT broker : Mosquitto

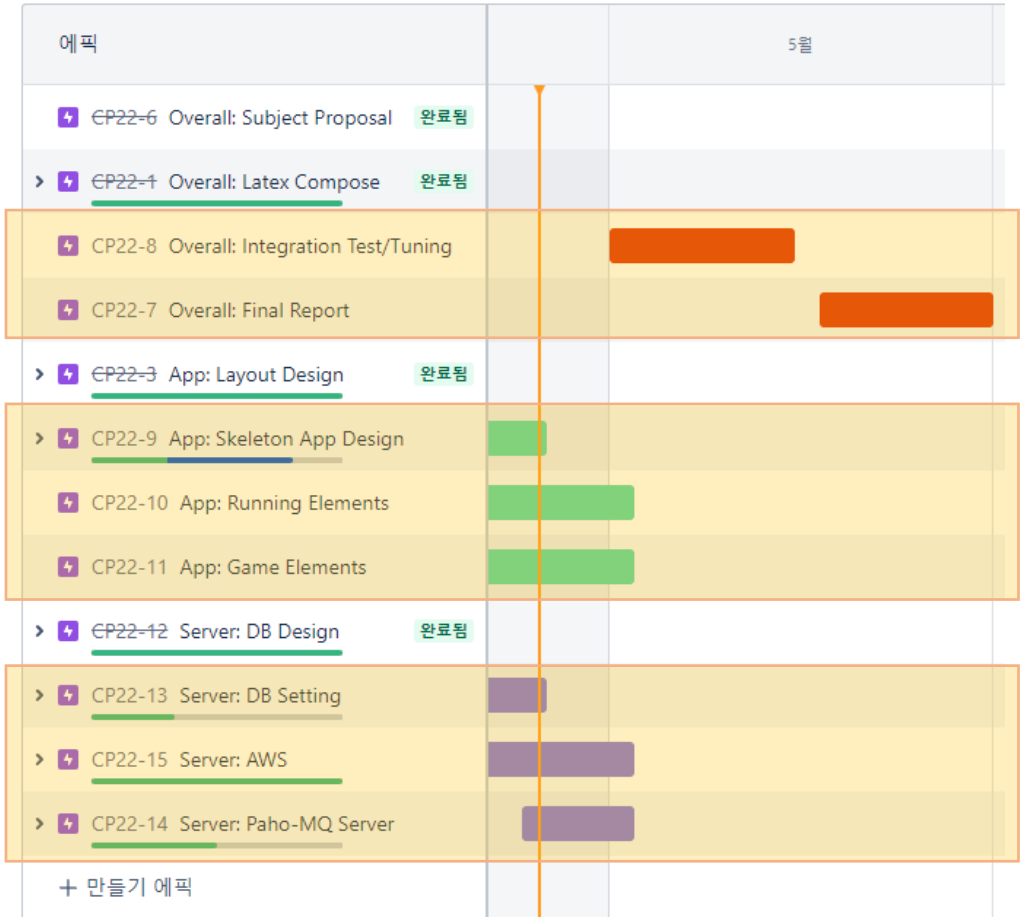
7. Limitation - backend





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

8. To-Dos



Front

- Views
- Actions

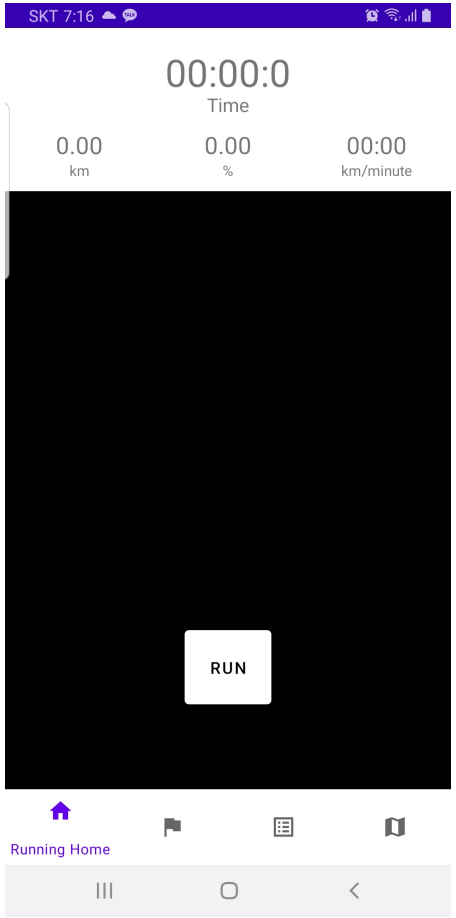
Back

- 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

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!