Mobile Computing (2021 Fall)

PaceMaker

2016-10454 이지원
2016-18221 이동현
2016-19985 서성호
2016-13919 이다운

목차

A table of Contents

목차

A table of Contents

Introduction *PaceMaker*

Problems

It is not easy to run with other people in the same place at the same time.

- To meet at the same time, appointment in advance is needed.
- It can be difficult to find a track where everyone can run together.
- Due to the COVID-19, it is burdensome to gather together.

Introduction *PaceMaker*

Goal

Give the user a chance to run together whenever they want

Make it feel like a track wherever users run and don't care about each other's location

Relieve the burden of gathering due to COVID-19

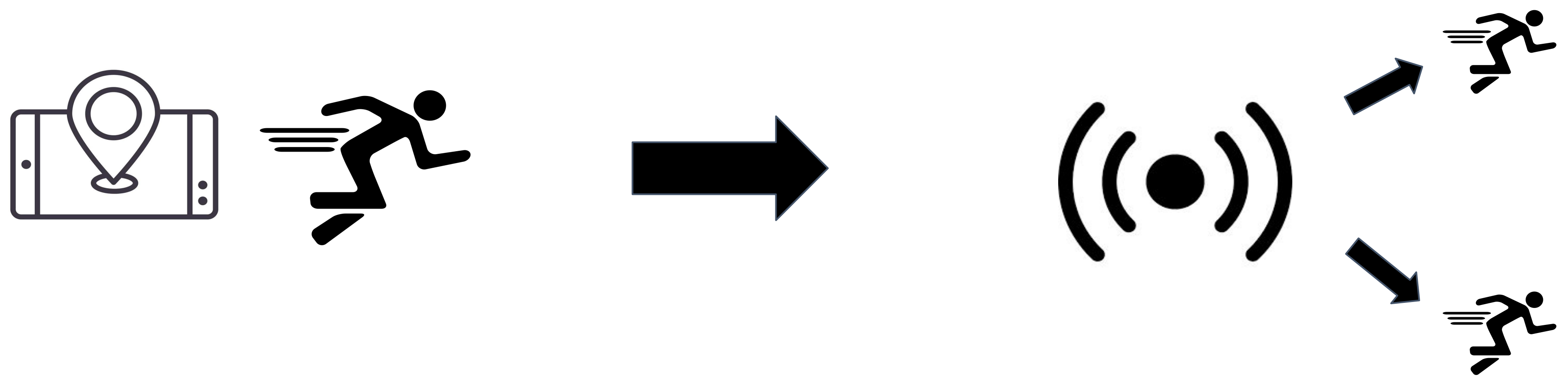
and *PaceMaker*

Introduction *PaceMaker*

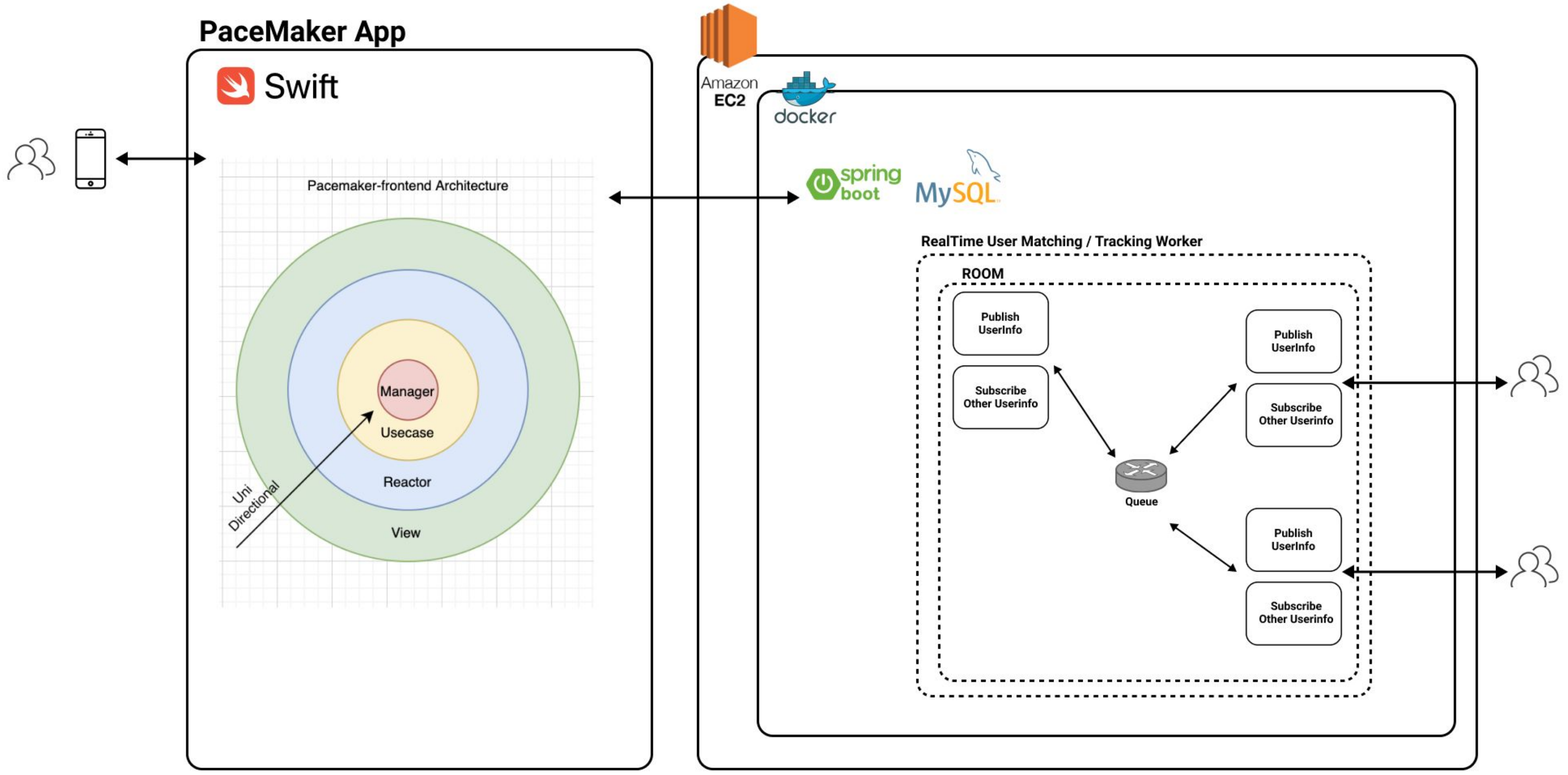
Key Solution

Match people who want to run by implementing a task queue

Running competitions by tracking users' GPS in real time

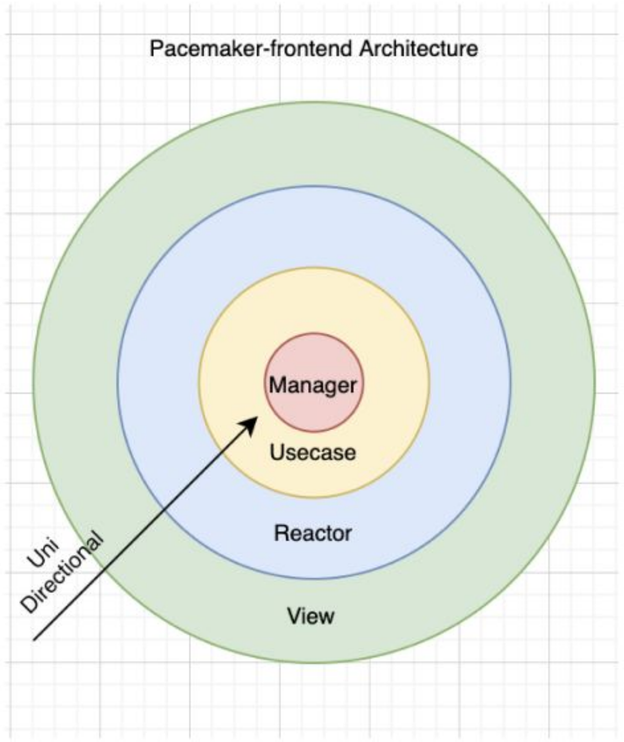


Architecture Overview



Architecture Overview

Swift



Client

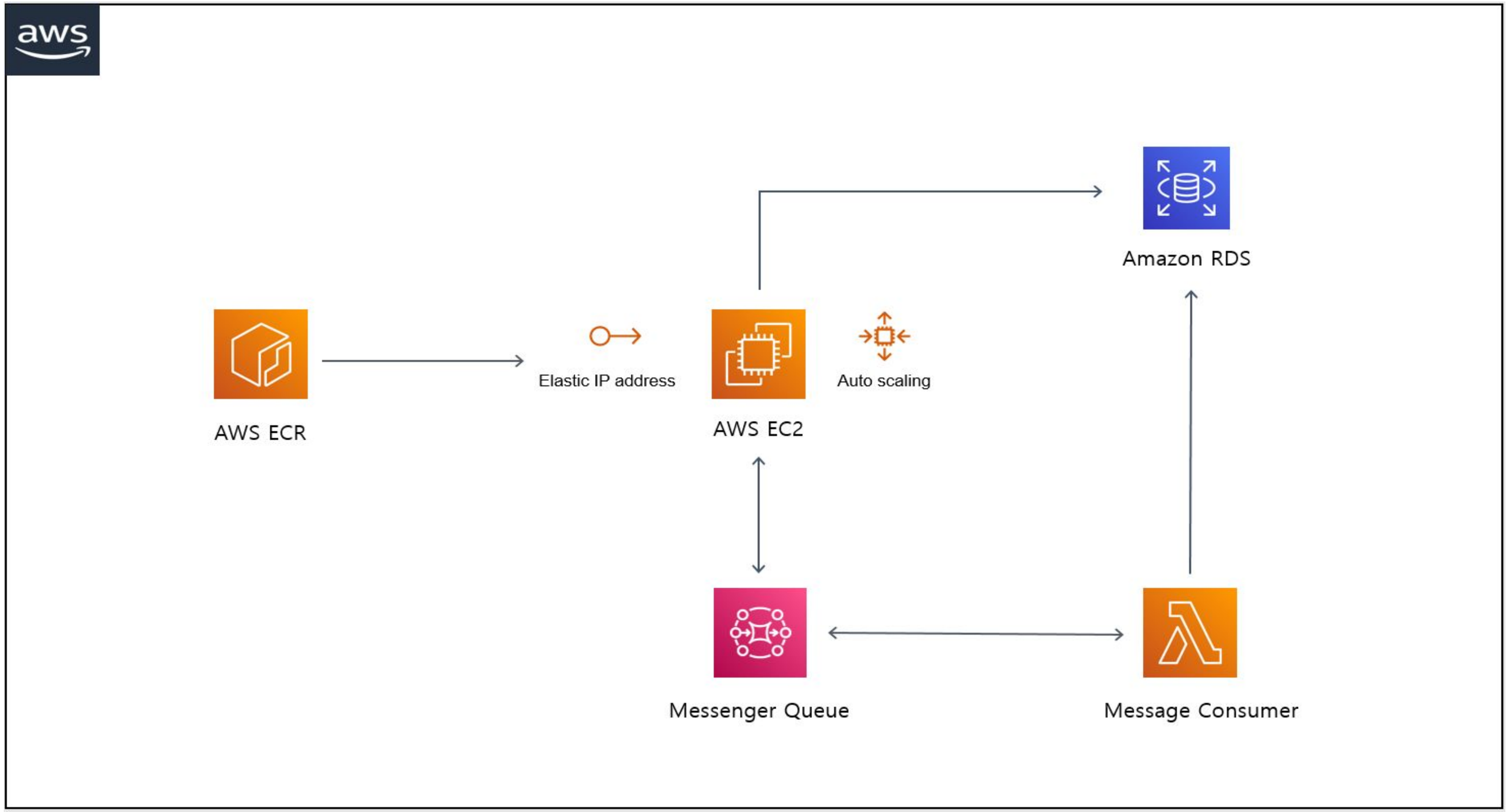
MVVM model

Uni directional hierarchy (ViewController, Reactor, UseCase, Manager)

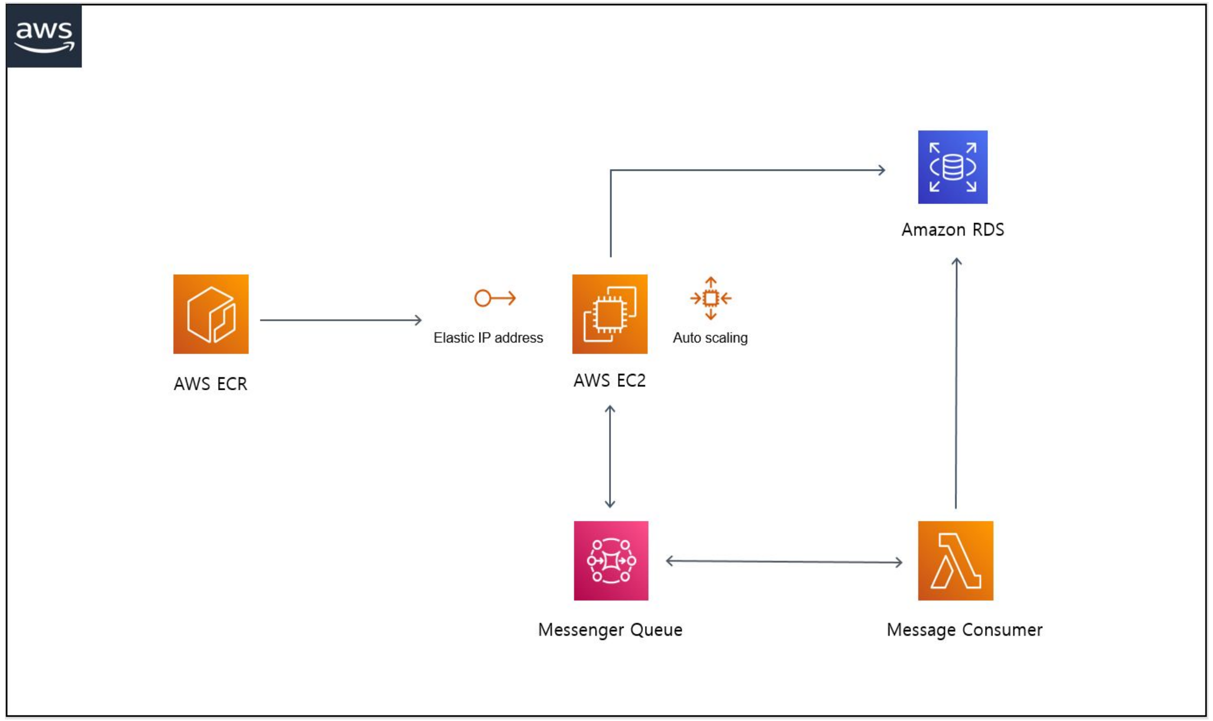
Map & Current Location & Running Route (CoreLocation, MapKit)

Code Configuration Management : Github

Architecture Overview



Architecture Overview



Server

Kotlin & Spring boot

Dockerize

ECR : Docker Image Store

EC2 : Application Cloud Computing Instance

RDS : User, Match, History Store

MQ(Messenger Queue) : Matching, Polling

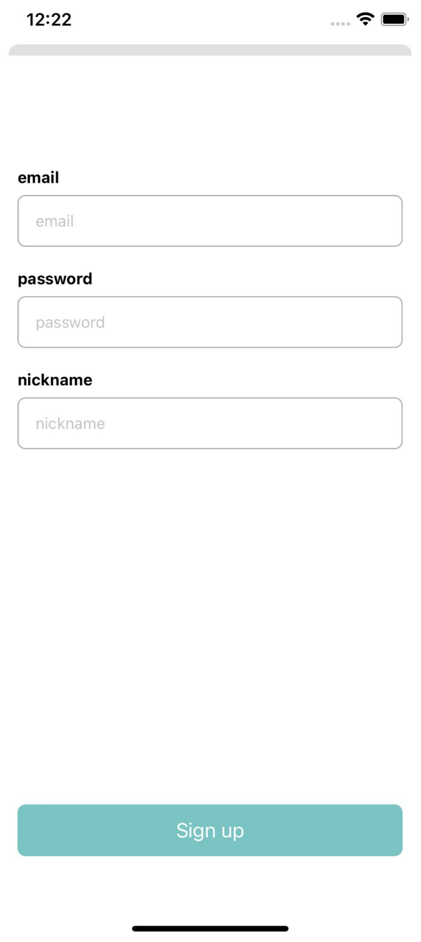
MQ Consumer : Matching

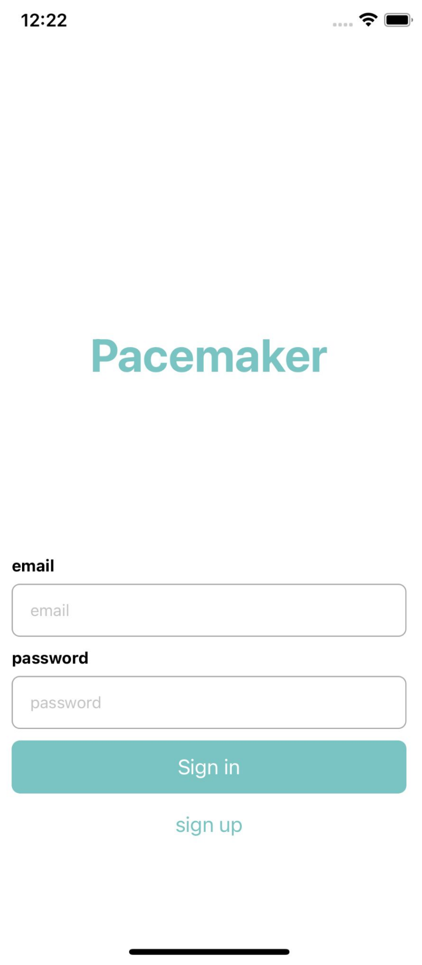
Code Configuration Management : Github

UI / UX

Login Flow

- Sign up
- Sign in
- Session control with oAuth (jwt token)

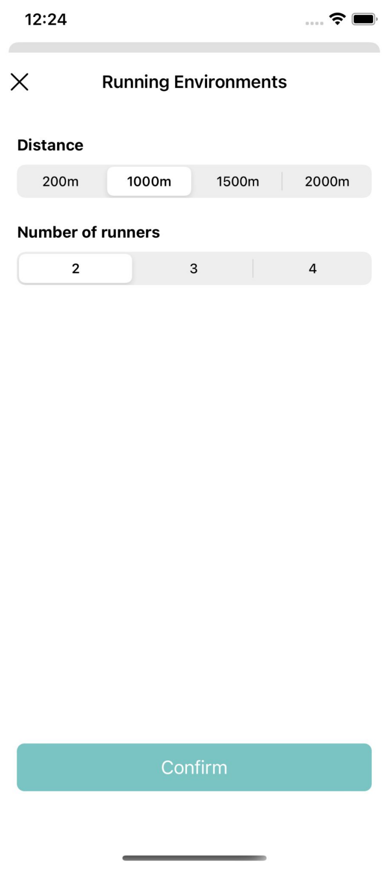




UI / UX

Main

- Match environment
- Match start
- Match Information polling with http request (interval 1)


12:24

Running Environments

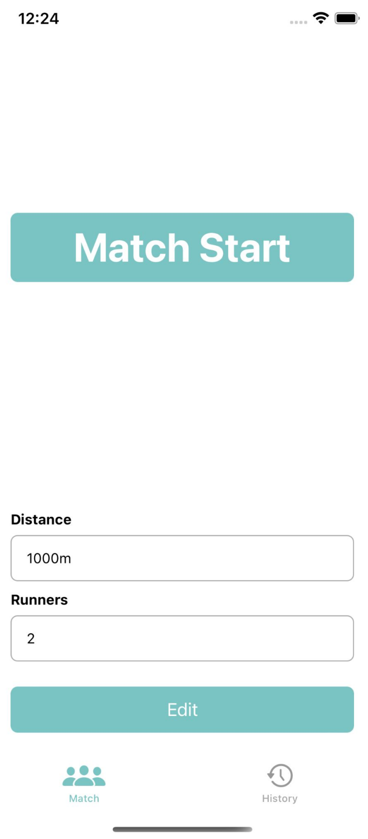
Distance

200m | 1000m | 1500m | 2000m

Number of runners

2 | 3 | 4

Confirm


12:24

Match Start

Distance

1000m

Runners

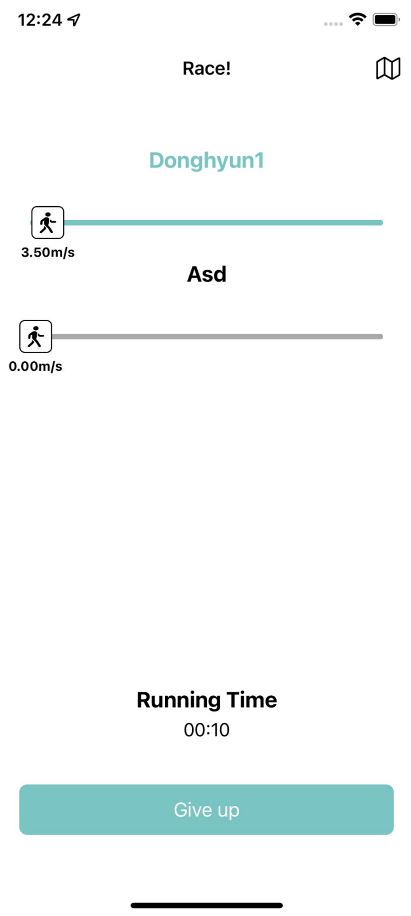
2

Edit

UI / UX

Match

- Match with others
- Time, speed, distance with Core Location
- Match result



12:24

Race!

Donghyun1

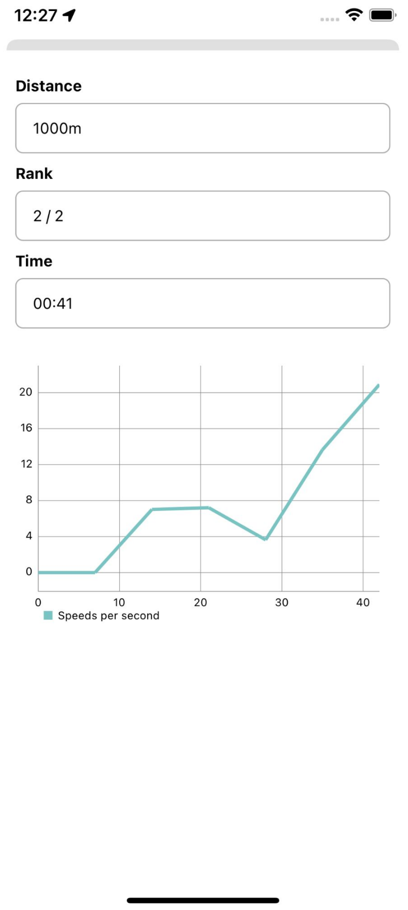
3.50m/s

Asd

0.00m/s

Running Time

00:10



12:27

Distance: 1000m

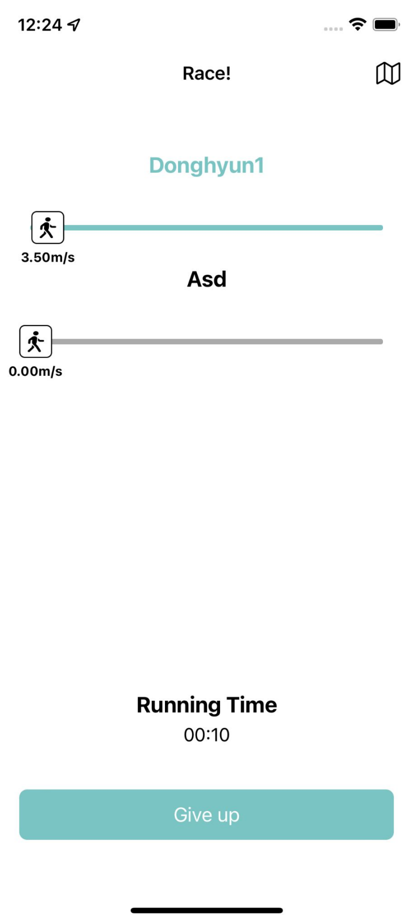
Rank: 2 / 2

Time: 00:41

UI / UX

Map

- Show route with MapKit
- Speed, distance with Core Location



12:24

Race!

Donghyun1

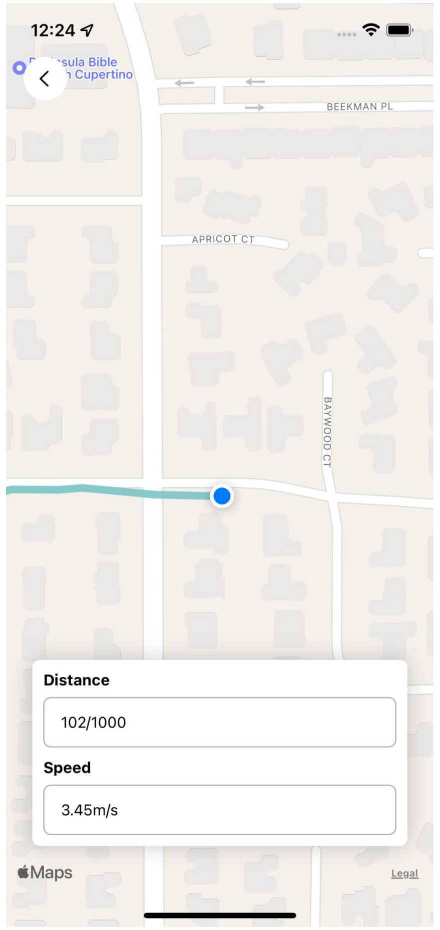
3.50m/s

Asd

0.00m/s

Running Time

00:10



12:24

Distance

102/1000

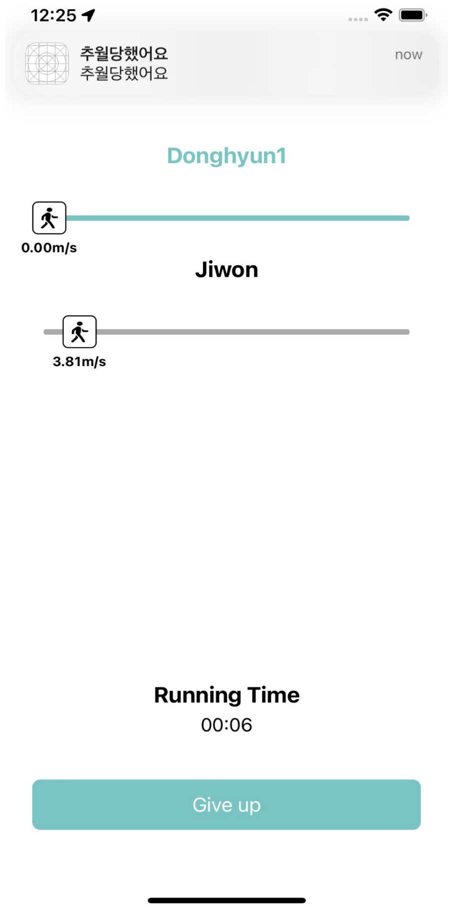
Speed

3.45m/s

UI / UX

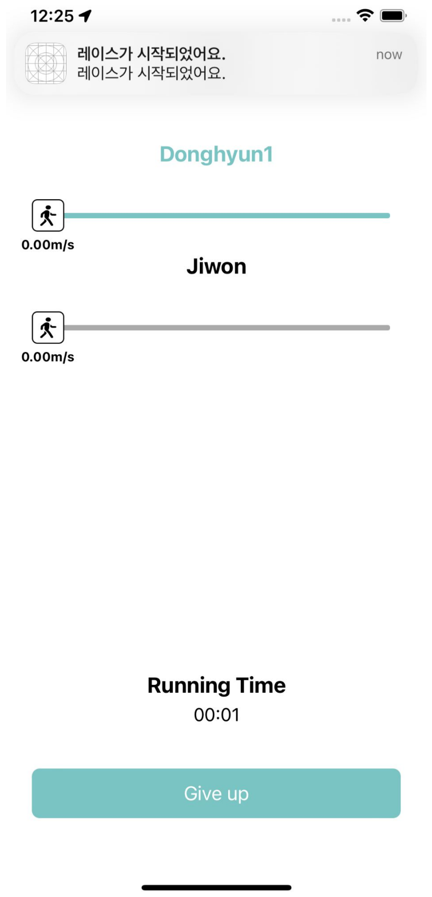
Notification

- Match event notification
- Sound, haptic, badge



12:25

추월당했어요
추월당했어요
now

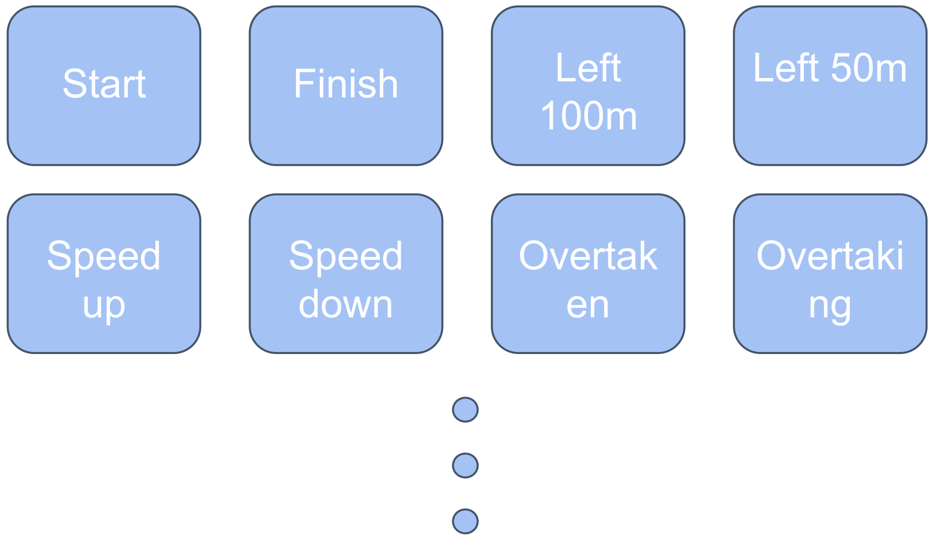


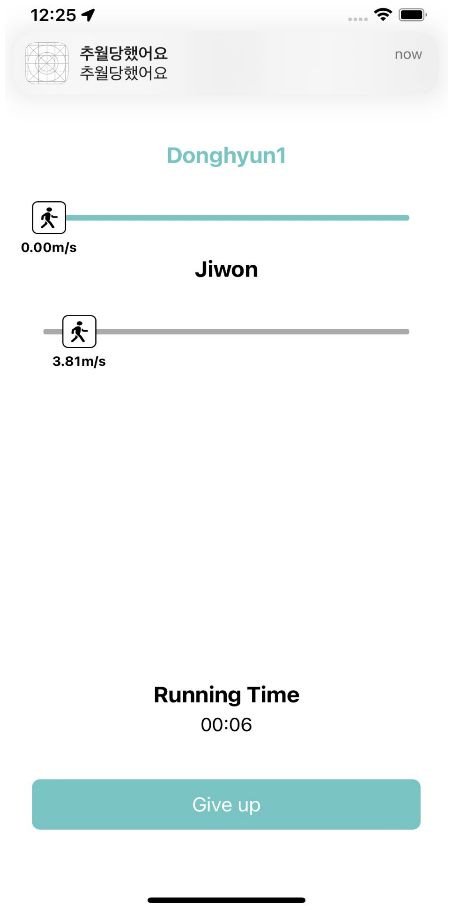
12:25

레이스가 시작되었어요.
레이스가 시작되었어요.
now

#3

UI / UX




12:25

추월당했어요
추월당했어요
now

Donghyun1

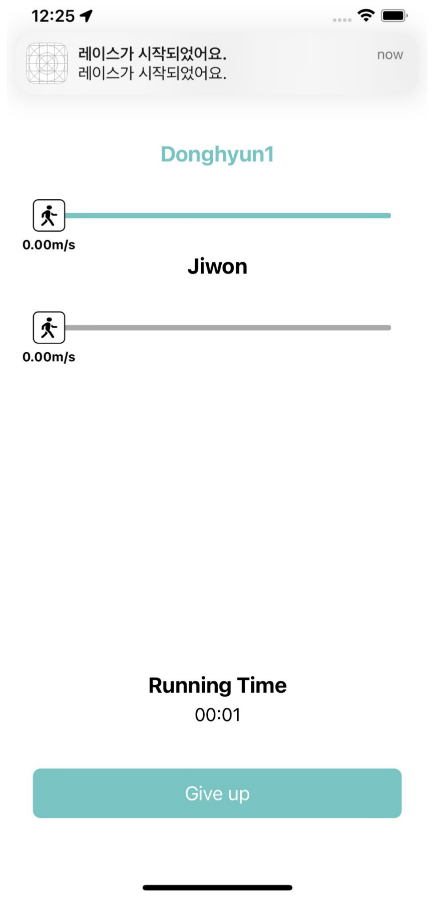
0.00m/s

Jiwon

3.81m/s

Running Time
00:06

Give up


12:25

레이스가 시작되었어요.
레이스가 시작되었어요.
now

Donghyun1

0.00m/s

Jiwon

0.00m/s

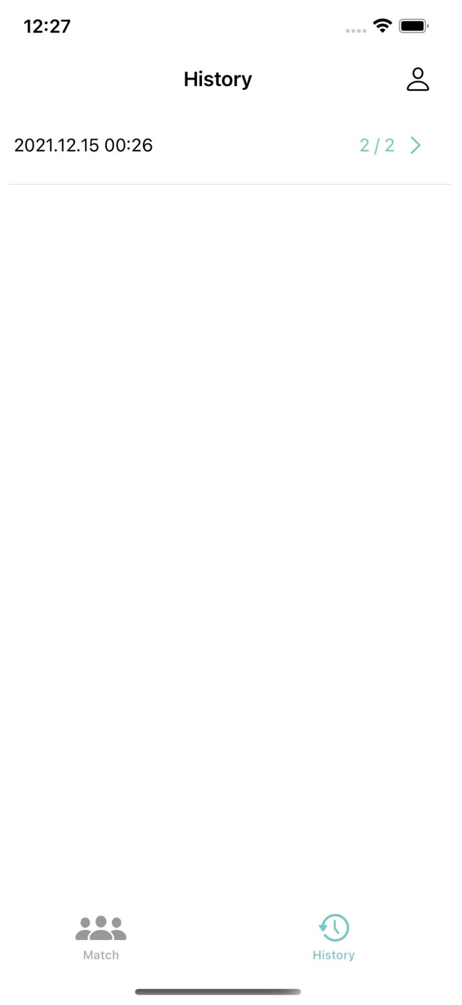
Running Time
00:01

Give up

UI / UX

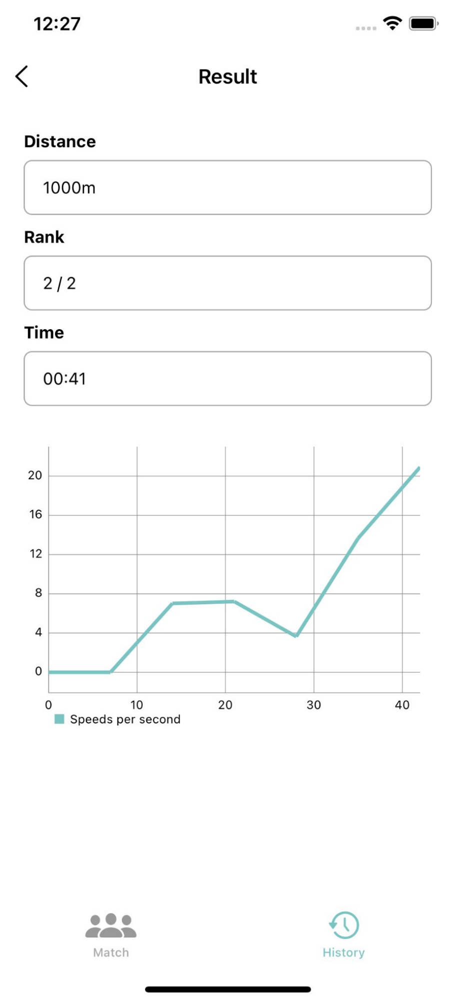
History

- List
- Detail



History

2021.12.15 00:26 2 / 2 >



Result

Distance

1000m

Rank

2 / 2

Time

00:41

🟩 Speeds per second

Challenges & Solutions

Expected challenges

- Matching users with **real time** data
- Implementing **matching algorithm**
- Getting current **location** & running **route** with map
- Measure Distance without advance information
- Frontend-Backend connection using **polling**
- Vibrate & sound **alarm** when event occured
- Representing **accurate** result & history

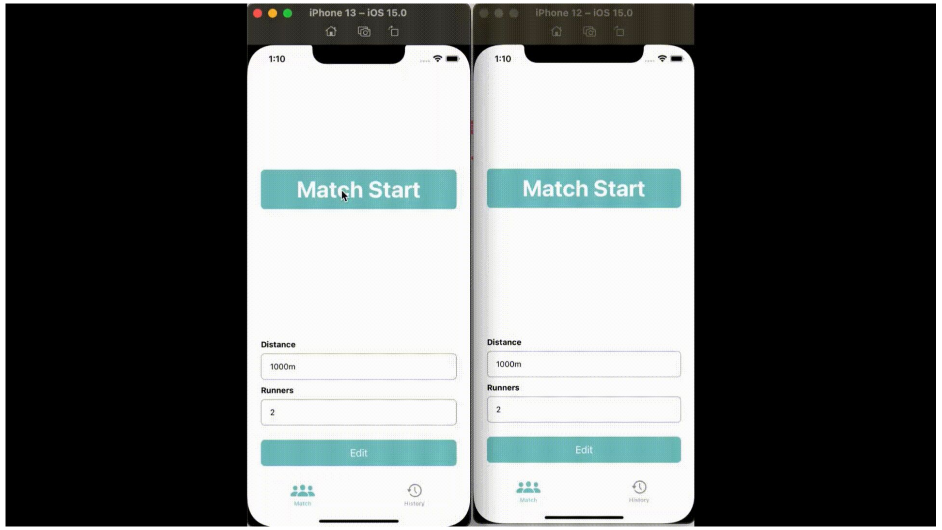
Challenges & Solutions

*Matching users with **real time** data*

Real-time matching & queueing

In the server, the consumer can match the queue in **real time** while continuously monitoring the queue.

So, in many cases, the test result *did not result in a matching delay of 2s or more.*



Challenges & Solutions

*Implementing **matching algorithm (options)***

Real-time matching algorithm

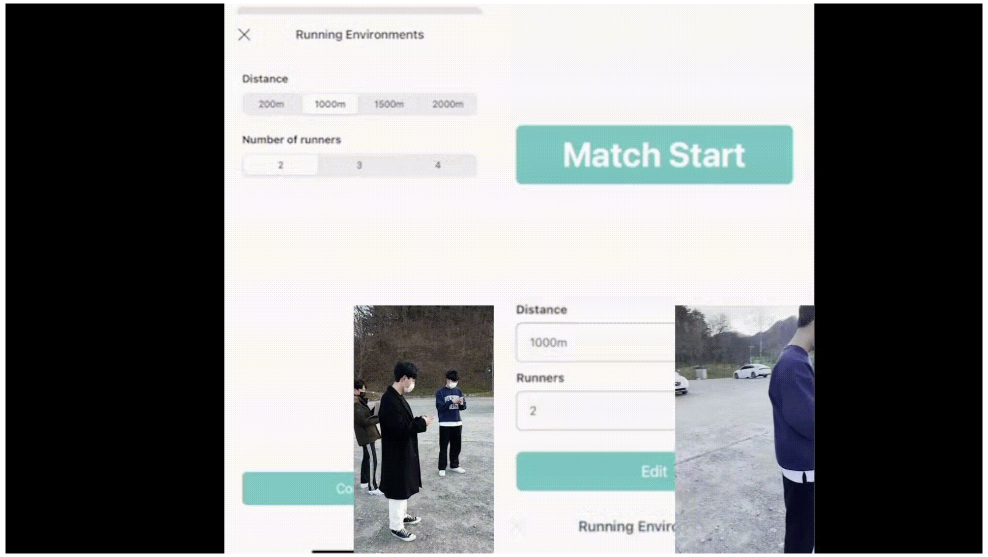
Operating with selected settings

User can select running **distance**, running **mates**.

Considering **preference** of users!

Expand and **matches** the queue according to

desired settings.



Distance
1000m
Runners
2

Challenges & Solutions

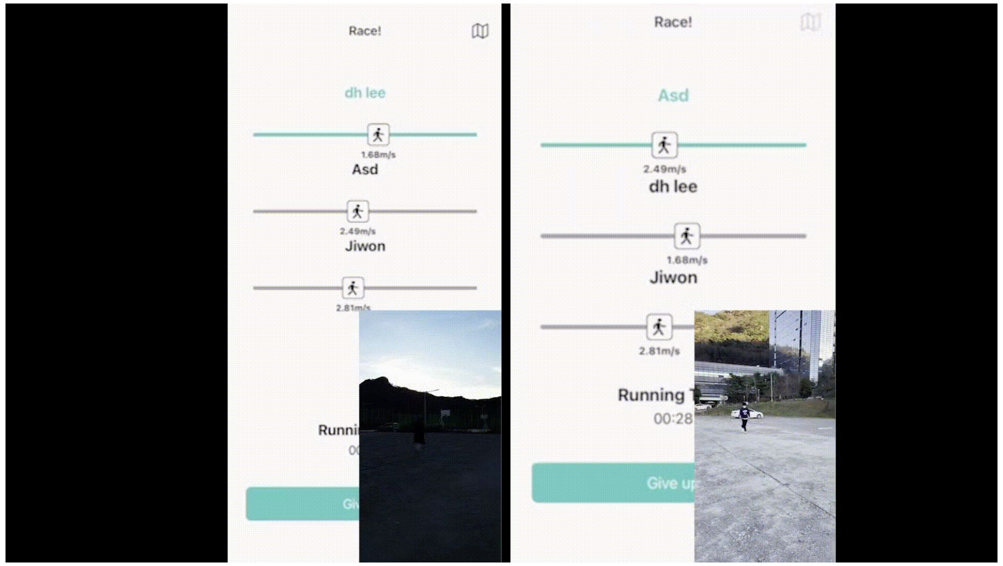
*Getting current **location** & running **route** with map*

Real time **accurate** running information

High accuracy on **straight** road

However, the moving distance accuracy is

slightly lowered on **curved** road.

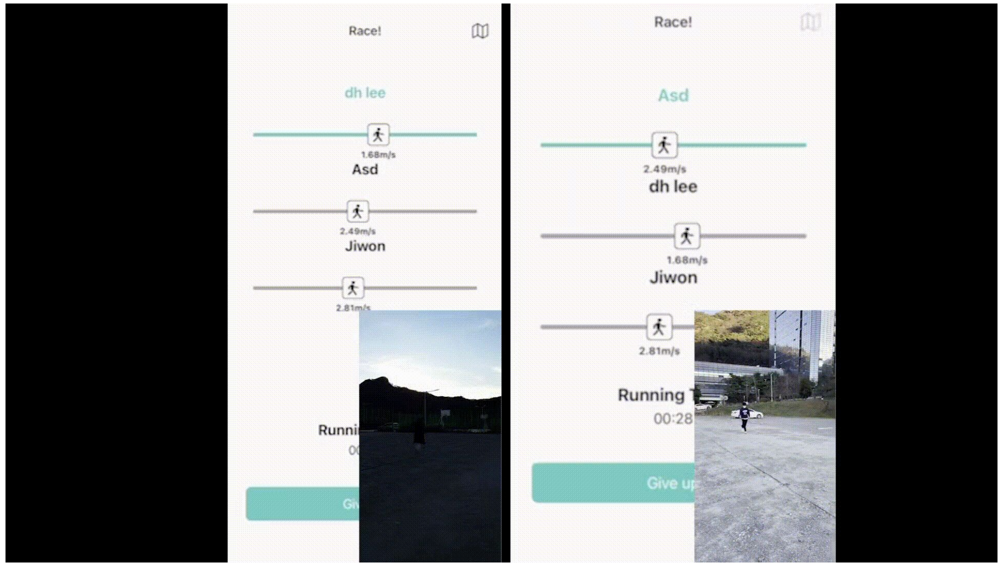


Challenges & Solutions

Frontend-Backend connection using ***polling***

It tracks the location and running information of **other users** and provides **real-time** analysis information.

It communicates through sockets at regular intervals using **polling API**.



Challenges & Solutions

*Vibrate & sound **alarm** when overtaken*

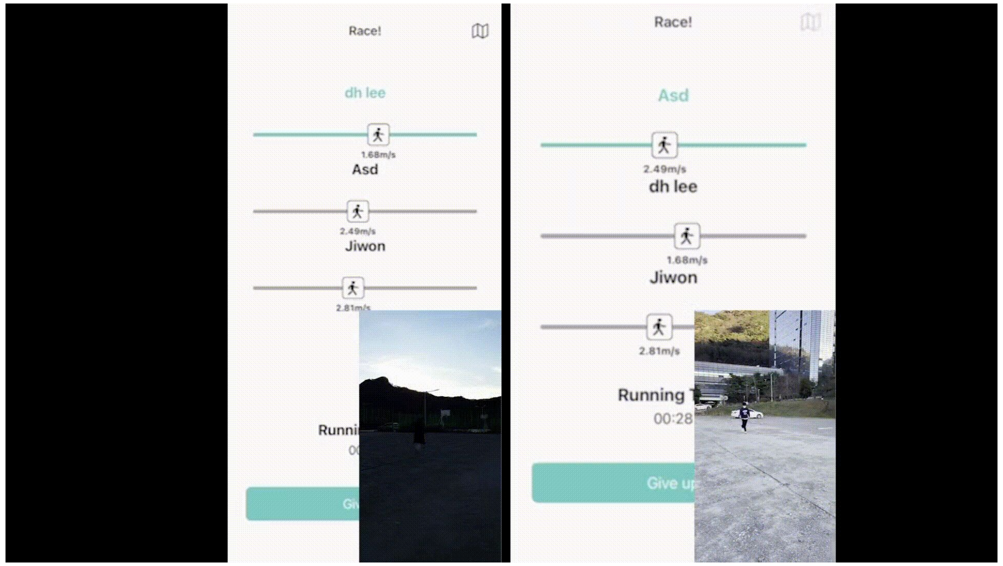
Provide **analysis running information** through

many **predefined** categories.

Using GPS, Accelerometer sensor

Vibrate & Sound push alarms

Start	Finish	Left 100m	Left 50m
Speed up	Speed down	Overtaken	Overtaking



Challenges & Solutions

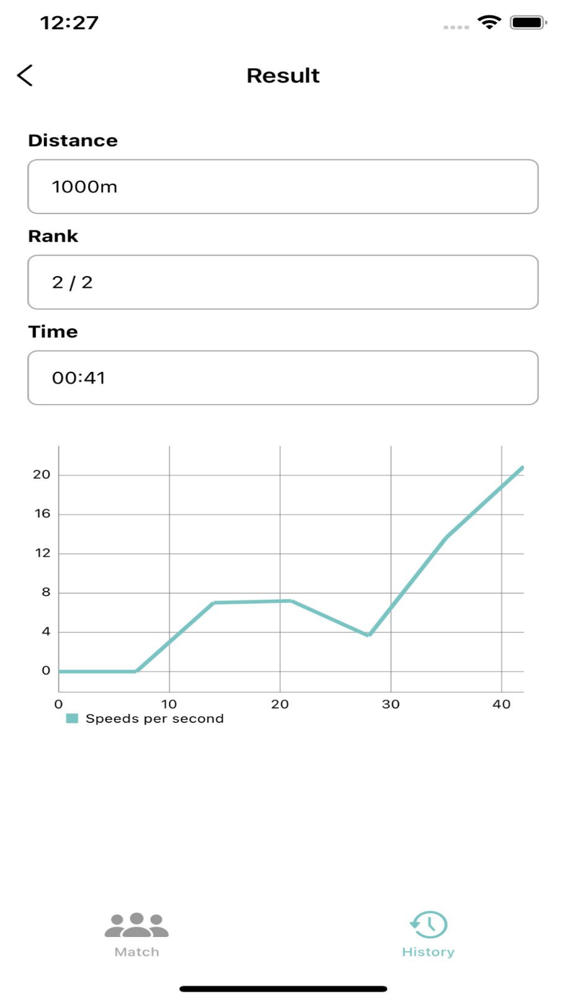
*Representing **accurate** result & history*

Result consists of rank, mean speed and graph.

History preserves previous running records.

User can see **intuitive** result & history,

so they can improve their running experiences!


12:27

Result

Distance

1000m

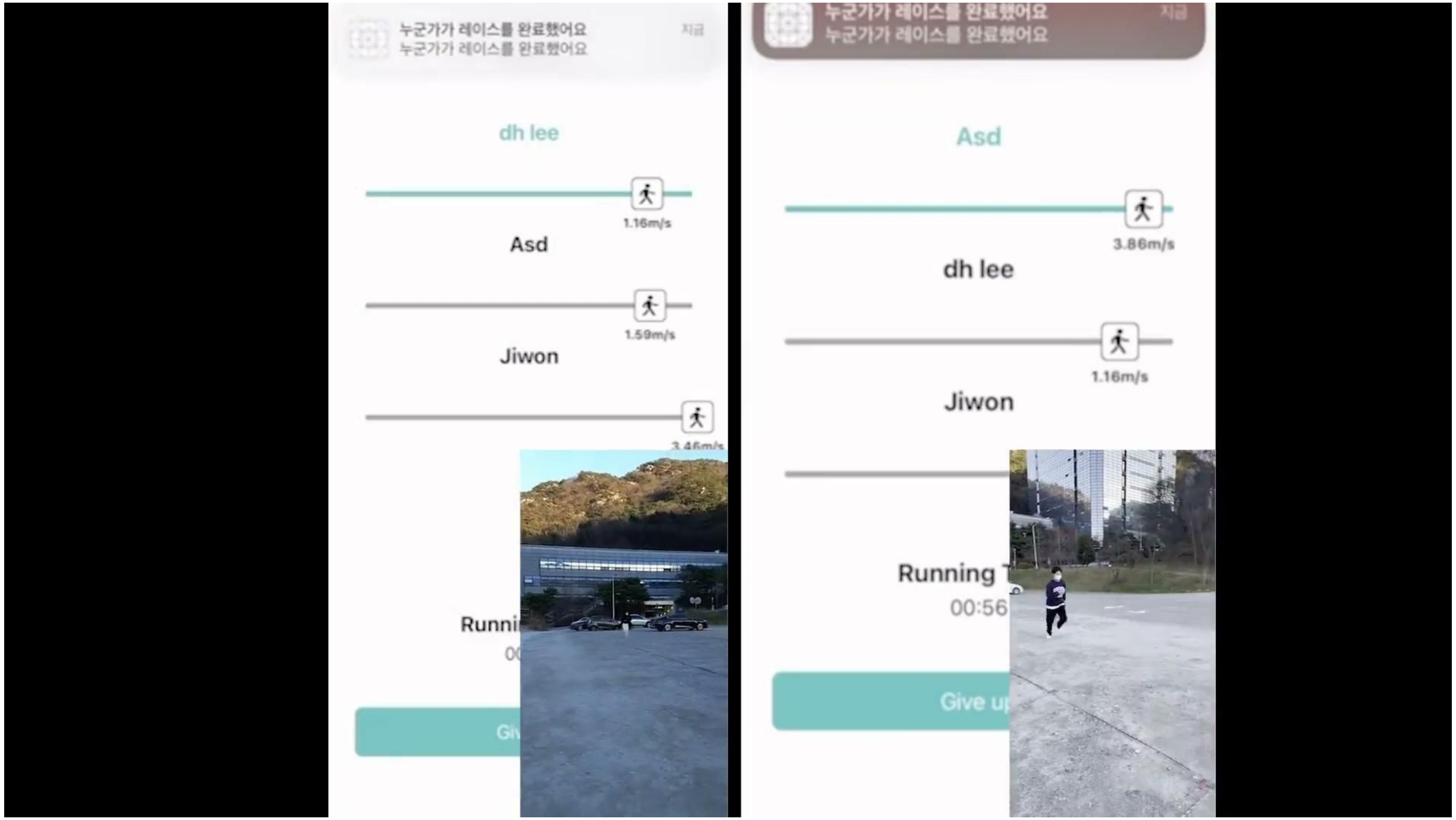
Rank

2 / 2

Time

00:41

Demo


누군가가 레이스를 완료했어요
누군가가 레이스를 완료했어요

Conclusion

- We have built an application anyone can easily find a running mate and run.
- Real-time data verification up to 1 second
- No need to select route, automatically calculated distance
- Notify events with os notification, sound, haptic
- Running history containing distance, speed, speed per seconds, visiualizing with graph

Evaluation

Resource consumption per session (measured by xcode instruments)

- 0~1% memory usage (85.7MB)
- 0~16% CPU
- Disk Reading (74.3MB)

Client-Server connection optimization

- Since we handle real-time connections with polling, there are many requests.
- However, by processing a lot of data such as push alarms and other player information through one polling, the overall performance does not fall behind.
- Calculations possible on the client, do not go through the server, reduce the network.

Project Schedule

		MileStone 2					MileStone 3		
	WEEK	2	4	6	8	10	12	14	16
✓	Project Design (plan, tech stack, ui/ux)	🟩	🟩						
✓	User / Authentication API			🟩	🟩				
✓	Login / Main Page			🟩	🟩				
✓	User Matching API				🟩	🟩	🟩		
✓	User Matching / In-Race Page				🟩	🟩	🟩		
✓	MVP Prototypes			🟦	🟦	🟦	🟦	🟦	
✓	Full-featured App (improve ui/ux, matching)							🟦	🟦
✓	Testing and Review								🟦

Role and Contributor

		What did
FrontEnd(iOS)	이동현	- Architecture, base structure design - Backend connection - In-Match UI/UX, pre-match / in-match polling - Location data handling
	이지원	- Main UI/UX - History UI/UX - Login UI/UX - Event notification
Backend	서성호	- oAuth API (using jwt token) - Match API (using MQ and RDS) - Polling API - Dockerizing and Deploy
	이다운	- User API - Polling API - History API - Local DB Test

감사합니다!

2016-10454 이지원
2016-18221 이동현
2016-19985 서성호
2016-13919 이다운