Customizable virtual assistant service for specific user

**Hanyang University Department of Computer Science**

**박종민**

목차

[Customizable virtual assistant service for specific user 1](#_Toc11512024)

[**0. The purpose and description of this report** 3](#_Toc11512025)

[**1. Project overview** 4](#_Toc11512026)

[**1.1. What is the virtual assistant service?** 4](#_Toc11512027)

[**1.2. Limitation of existing products** 5](#_Toc11512028)

[**1.3. Difference between my project and existing products** 7](#_Toc11512029)

[**1.4. Motivation** 7](#_Toc11512030)

[**1.5. Goal** 7](#_Toc11512031)

[**2. description of project** 9](#_Toc11512032)

[**2.1. entire project structure** 9](#_Toc11512033)

[**2.2. client** 10](#_Toc11512034)

[**2.2.1. Text to voice, voice to text converter** 10](#_Toc11512035)

[**2.2.2. GPS module** 11](#_Toc11512036)

[**2.2.3. how does client work?** 17](#_Toc11512037)

[**2.3. server** 20](#_Toc11512038)

[**2.3.1. Django framework** 21](#_Toc11512039)

[**2.3.2. 현재 날씨/일기예보** 22](#_Toc11512040)

[**2.3.3. 다음 일정** 24](#_Toc11512041)

[**2.3.4. 식사 추천** 27](#_Toc11512042)

[**3. Future works** 29](#_Toc11512043)

[**3.1. change device from raspberry pi to smart phone** 29](#_Toc11512044)

[**3.2 recommender system for meal service** 30](#_Toc11512045)

# **0. The purpose and description of this report**

먼저 제공해 주신 양식에 따라 보고서를 작성하지 못 한 이유에 대해 간략히 설명해 보자면

우선 제공받은 양식은 기승전결이 어느 정도 갖춰진 완성된 보고서라 할 수 있습니다.

허나 그런 보고서를 쓰려 하자면 100을 계획했다면 적어도 80이상은 완벽히 구현되어 있고 나머지는 future works로 돌리는 식으로 해서 쓸 수 있다고 생각합니다.

하지만 본 프로젝트 경우에는 제가 생각했을 때 100을 계획했다면 아직 20도 채 완성되지 못한 미완성 프로젝트라고 생각합니다. 이런 상황속에서 내가 한 것은 20인데 80을 한 것처럼 보고서를 작성하는 것이 양심상 찔리고 단순히 보여주기 식의 실속 없는 문서보다는 실제로 도움이 될 수 있는 보고서를 쓰려고 합니다.

그렇기에 기승전결 형식의 깔끔한 보고서 형식이 아니라 약 3주 기간 동안 프로젝트를 진행하면서 겪었던 issue들과 나의 해결방식 더 나아가 이렇게 하면 더 좋을 것 같은 추천방식으로 서술하여 나중에 제가 겪었던 것과 비슷한 상황에 처할 때 참조할 수 있는 보고서를 쓰는 것을 목표로 합니다.

# **1. Project overview**

## **1.1. What is the virtual assistant service?**

현재 스마트폰에서 이용할 수 있는 구글의 assistant, 애플의 시리, 삼성의 빅스비 같은 인공 지능 비서 서비스를 지칭합니다.

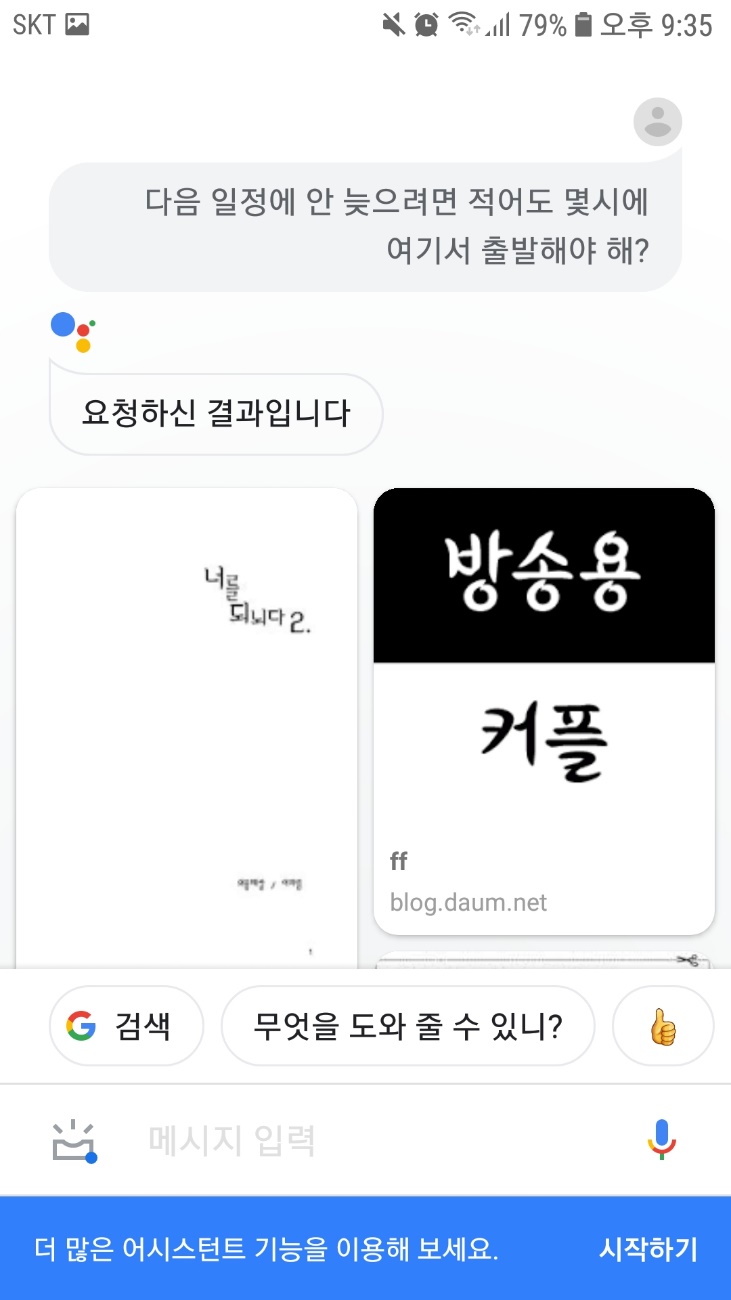
이들의 특징으로는 실제 개인 비서처럼 사용자가 요구하는 작업을 처리하고 인공 지능(AI) 엔진과 음성 인식을 기반으로 사용자의 음성 명령에 따라 일정관리, 이메일 전송, 식당 예약 같은 작업을 수행한다는 것입니다.

아래의 표는 현재 상용중인 가상 비서 서비스(virtual assistant service)들과 그들이 제공하는 기능들을 보여줍니다.



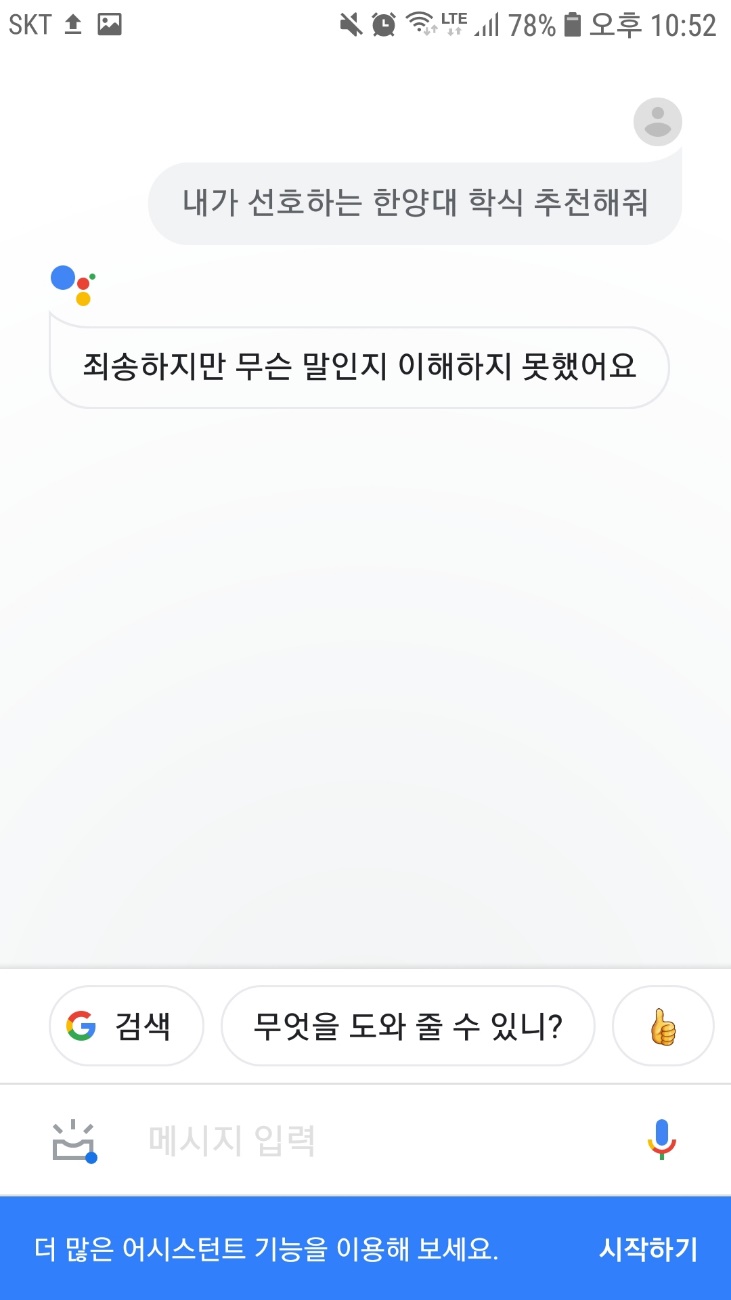
Reference : <http://www.dbpia.co.kr/Journal/ArticleDetail/NODE07241110>

## **1.2. Limitation of existing products**



현재 위치에서 다음 일정 장소까지 늦지 않게 도착할 수 있는 가장 늦은 출발시각과 같은

구체적이고 어려운 질문에 대답을 못합니다.



사용자의 기호에 따른 점심 추천과 같은 사용자 사적인 개인정보가 제공되기 어려운 환경에선

적절한 서비스를 제공할 수 없습니다.

## **1.3. Difference between my project and existing products**

이러한 기존 인공지능 비서 서비스들은 특정 사용자가 아닌 범용적인 사용자를 대상으로 서비스를 진행하기 때문에 사용자의 특성을 최대한 이용한 맞춤형 service라기 보다는 현재는 단순히 범용적으로 사람들 사이에서 많이 사용하는 기능들만 제공하고 있습니다.

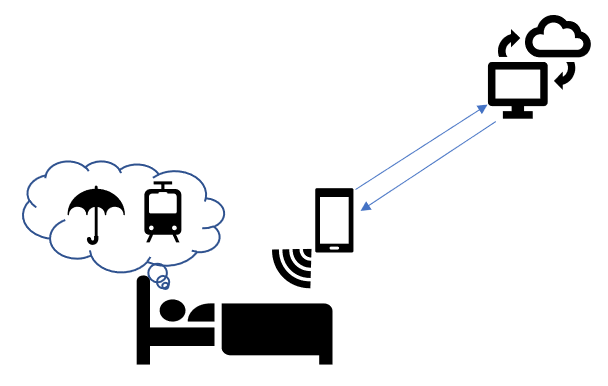
이러한 한계점에 착안하여 본 프로젝트는 특정 개인에게 최적화되어 있는 비서 서비스를 만드는 것을 목표로 합니다.

## **1.4. Motivation**

이 프로젝트를 시작하게 된 배경에 대해 이야기하자면 먼저 기왕 프로젝트를 하는 김에 나에게 좀 더 도움이 되는 프로그램을 만들어 보자는 생각에서 시작하게 되었습니다. 그러던 중 개인적으로 아침잠이 많아서 아침에 해야 할 일들을 도와주는 프로그램을 만들면 조금이라도 더 늦잠을 잘 수 있지 않을까 하는 게으른 생각에서 프로젝트를 시작하게 되었습니다.

## **1.5. Goal**

프로젝트 초기에 계획했던 최종 목표는 아래와 같습니다.



아침에 스마트폰 알람에 맞춰서 스마트폰 내의 virtual assistant client가 자동적으로 실행되어 cloud 서버와 서로 통신하면서 일기예보, 오늘의 스케쥴, 지각하지 않으려면 적어도 몇 시에 집에 나가야 하는지 같은 것을 브리핑 해주는 서비스를 만드는 것을 목표로 잡았습니다.

허나 여러 번의 중간 발표와 최종발표 때 받은 피드백을 바탕으로 vision을 좀 더 크게 잡아 새롭게 목표를 설정해 보았습니다.

New goal

아침 브리핑뿐만 아니라 마치 개인집사가 존재하는 것처럼 하루 일과를 시간 단위로 세세하게 계획해주는 “smart”한 비서 서비스로 개량하는 것을 목표로 하였습니다.

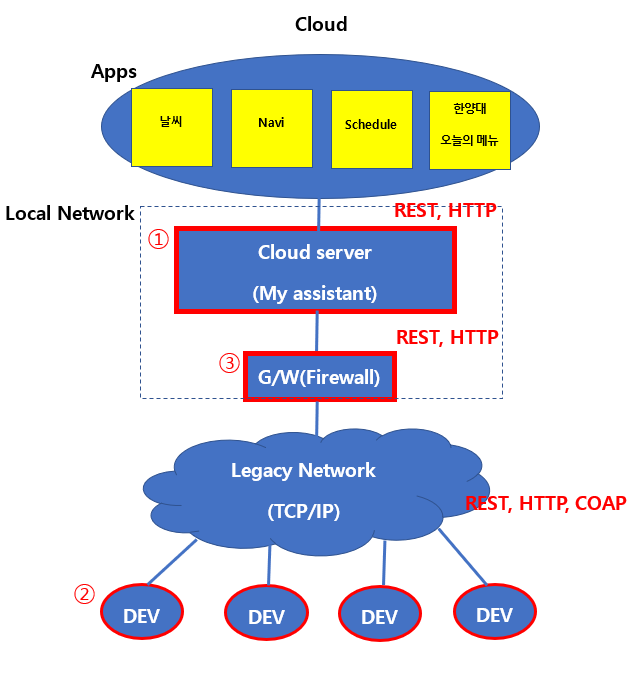
이렇게 할 수 있는 근거로는 우선 내 개인 일정을 모두 열람 및 수정할 수 있는 권한을 줄 수 있고 또한 내가 좋아하는 음식, 장소, 취미 등을 내가 직접 입력하여 인공지능 시스템이 이 정보들을 바탕으로 좀 더 정확한 맞춤형 서비스를 제공할 수 있기 때문입니다.

예를 들어 하루 일과를 모두 캘린더에 등록해 둔다면 개인집사는 이를 바탕으로 미래를 예측해 최적의 동선 및 일정을 수정하여 제공해 줄 수 있습니다. 가령 점심시간 전 후로 itbt에서 수업을 듣는다면 이 근처의 학식 메뉴를 사용자 기호에 맞게 추천 해 줄 수 있고,

만약 다른 지역에서 일 또는 공부하고 있을 경우 그 근처 식당의 메뉴를 추천 더 나아가 예약해주는 것까지 생각해 볼 수 있습니다.

# **2. description of project**

## **2.1. entire project structure**

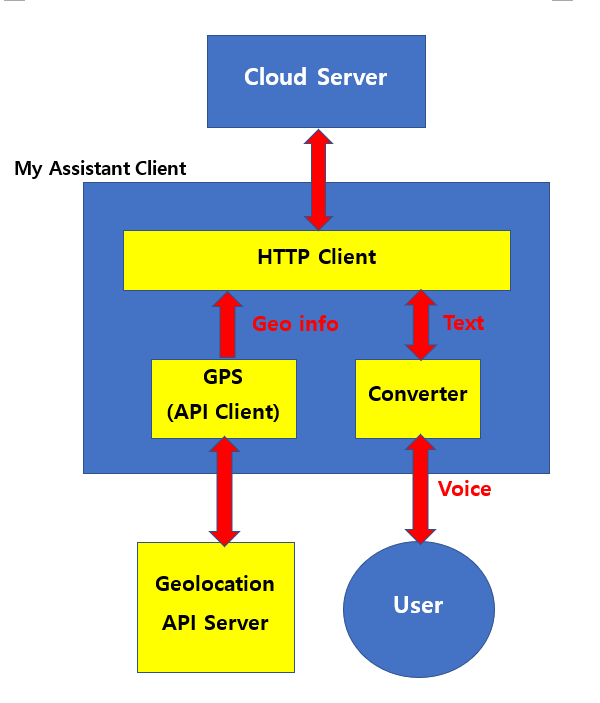


기본적으로 제 프로젝트는 수 많은 third party API에 의해 동작하게 되어있습니다.

기존에 제공되는 API 서비스들을 조합하여 저만의 서비스를 만들어 사용자에게 제공하는 것입니다. 클라우드 서버는 restful하게 설계되어 있습니다. 즉 http를 사용합니다. 하지만 http가 아닌 다른 프로토콜을 사용하는 다양한 디바이스에서도 서비스를 이용할 수 있도록 네트워크 앞 단에 게이트웨이를 두는 구조를 취하고 있습니다. 게이트 웨이의 존재 덕분에 클라우드 서버는 오직 http만 고려하여 설계하면 되고 또한 1차적으로 게이트웨이(+firewall)에서 악의적인 사용자의 패킷들을 걸러줄 수 있기 때문에 보안적으로 이점이 있다고 생각합니다.

먼저 component를 server, client, gateway로 크게 3부분으로 나누었는데 현재 구현 되어있는 건 server와 client뿐입니다.

## **2.2. client**



### **2.2.1. Text to voice, voice to text converter**

먼저 사용자가 음성으로 명령어를 입력하면 이를 내부적으로 다시 text로 변환하는 형식으로 설계하였습니다. 또한 서버로부터 받은 text 응답을 다시 voice로 변환하여 사용자에게 응답하는 방식입니다. 결론만 말하면 현재는 converter없이 text로 명령을 입력하고 응답을 받는 형태로 구현 되어있습니다. 허나 조사해 본 결과 google API service를 이용하면 이를 쉽게 구현할 수 있을 것 같습니다.

Cloud text-to-speech

<https://cloud.google.com/text-to-speech/?hl=ko>

Cloud speech-to-text

<https://cloud.google.com/speech-to-text/?hl=ko>

### **2.2.2. GPS module**

GPS 모듈은 google geolocation API를 이용하여 위도 경도 좌표를 얻어냈습니다.

이 기능이 필요한 이유는 사용자의 현재 위치를 기반으로 서비스를 제공하고 싶었기 때문입니다.

예를 들어 날씨 같은 경우에는 사전에 지정된 위치의 날씨가 아닌 동적으로 사용자의 현재 위치의 기상정보를 알려주고 싶었습니다. 즉 해외에 나가게 되더라도 서버와의 연결만 되어 있다면 해외 날씨를 추가적인 변경없이 제공받고 싶었습니다.

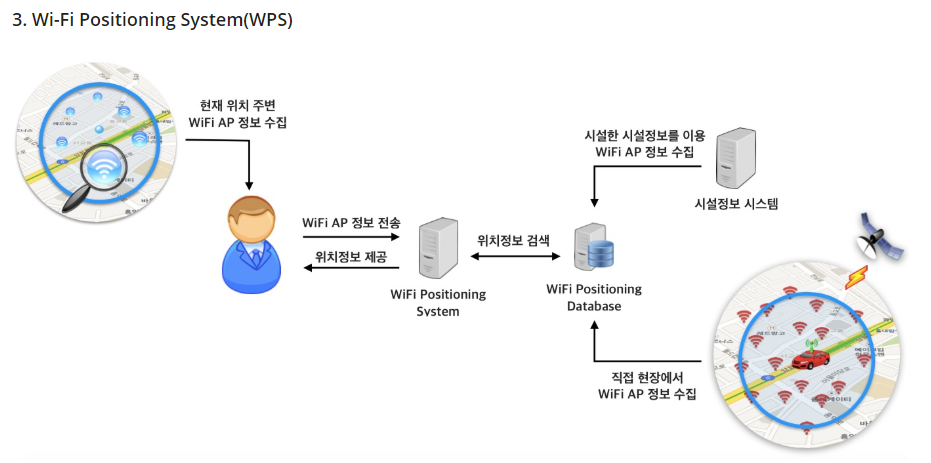
또한 현재 위치를 기준으로 근처의 식당이나 현재 위치에서 캘린더에 등록 되어있는 다음 약속 장소 까지의 경로 및 걸리는 시간을 계산하기 위해서는 geo location information이 필수적 이였습니다.

하지만 문제는 raspberry pi에 기본적으로 GPS 모듈이 제공되지 않고 추가적으로 돈을 내고 구입하기에는 비용이 부담되었습니다. 그렇다고 IP주소를 통해 geo location을 추정하기에는 오차가 1km가 넘어가서 만족스럽지 않았습니다.

가장 best 해결책은 사용자 device를 GPS기능이 기본적으로 있는 고사양의 스마트폰이나 다른 h/w로 변경하는 것이지만 제한된 자원으로 이를 해결하기 위해 공부하던 중 WIFI positioning system이란걸 알게 되었습니다.

#### 2.2.2.1. Wi-Fi positioning system

간략하게 WIFI positioning system에 대해 설명하자면



그림과 같이 사전에 현장에서 직접 WIFI AP 정보를 수집하여 DB에 저장합니다.

후에 사용자가 자신의 주변의 WIFI AP정보들을 서비스 제공자에게 보내게 되면 사전에 수집해 놨던 AP정보들을 바탕으로 사용자의 위치를 추정해서 그 geo location 정보를 반환해 주는 시스템을 일컫습니다.

Reference : <https://www.sphinfo.com/google-maps-geolocation-api/>

이러한 WIFI positioning system을 구글해서는 API 형태로 제공하고 있었습니다.

#### 2.2.2.2. Google geolocation API

아래와 같이 body에 WIFI information을 담아 해당 URL로 request를 보내면 json형식으로 위치정보와 정확도를 사용자에게 응답해줍니다.



세부적인 API 사용방법을 보고서에 기술하는 것보다는 구글에서 제공하는 document 직접 보는게 더 정확하고 도움이 될 것 같아 링크만 남깁니다.

<https://developers.google.com/maps/documentation/geolocation/intro>

이제 본격적으로 API를 이용하기 위해 http body에 들어가야 할 AP list를 구하는 과정에 대해 이야기하겠습니다.

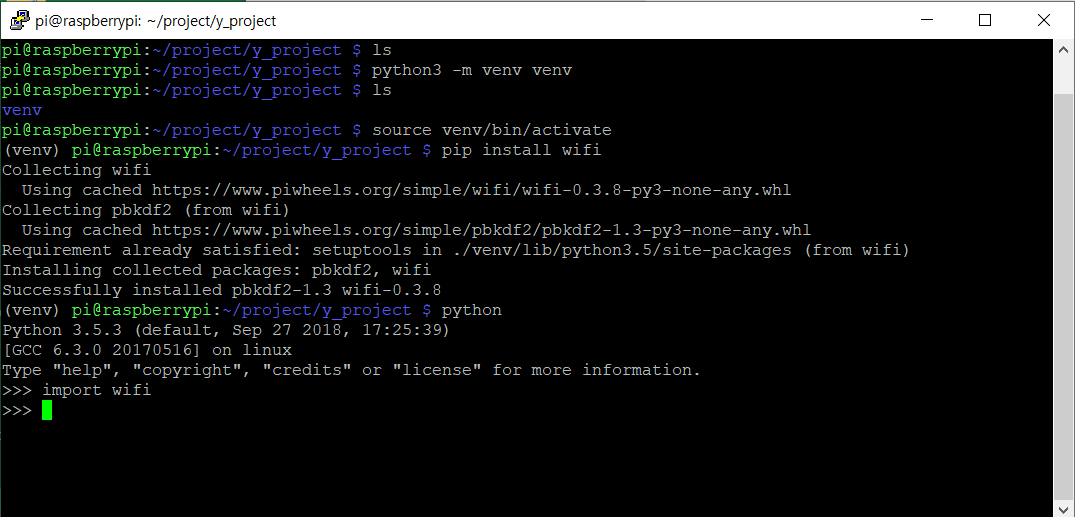
처음에는 리눅스 명령어인 wiliest 명령어를 이용하여 출력되는 결과를 프로그램에서 읽어서 이를 parsing한 후 사용할까 하였는데 일단 다른 사람이 이러한 일을 하는 프로그램(library)를 만들어 두지 않았을까? 하는 생각에 찾아보았습니다.

조사해 본 결과 open source library인 python wifi 라이브러리가 있었습니다.

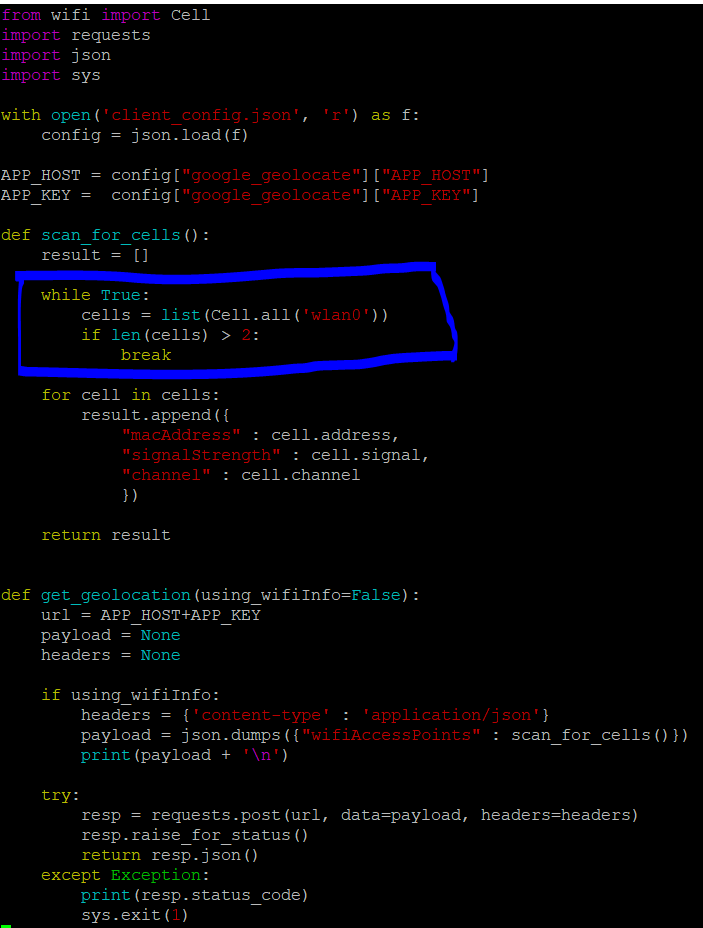
#### 2.2.2.3. Open source library python wifi

Wifi provides a command line wrapper for iwlist and /etc/network/interfaces that makes it easier to connect the WiFi networks from the command line. The wifi command is also implemented as a library that can be used from Python.

#### 2.2.2.4. 라이브러리 설치 및 적용(command line)



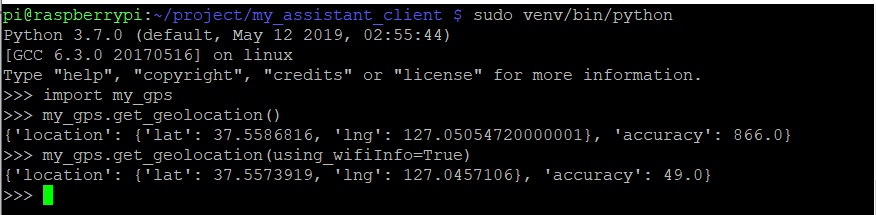
#### 2.2.2.5. source code



파란색 표시된 부분은 무선랜 인터페이스 특성상 신호의 세기나 주변 WIFI 개수가 수시로 바뀌기 때문에 이러한 걸로 생기는 오차를 최소화 하기위해 여러 번 측정한 후 평균값을 취하는 작업이 필요합니다. 실제로 최근 값에 가중치를 두어 계산을 하지만 단순히 본 과제에서는 일정 대수 이상의 무선 ap mac address만 있으면 상당히 높은 정확도를 얻을 수 있기에 스캐닝 이후 발견된 ap의 개수가 2개 이상이면 해당 WIFI info를 사용하는 것으로 naive하게 이 문제를 해결하였습니다.

#### 2.2.2.6. Simple test

아래는 모듈을 파이썬 인터프리터에서 import 한 후 해당 function을 실행시켜 geo location 정보를 받아온 모습입니다.



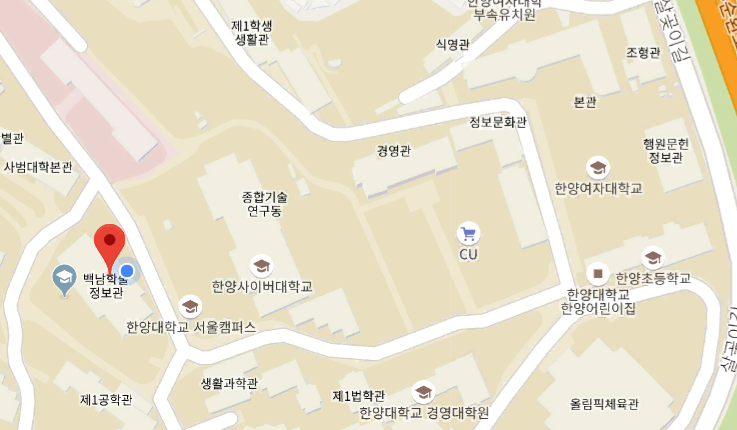
**Without WIFI info**



WIFI Info를 사용하지 않았을 경우

오차가 꽤 크다는 걸 볼 수 있습니다.

**With wifi info**



WIFI info를 사용했을 시 보시는 바와 같이 오차가 거의 없다는 걸 알 수 있습니다.

결론을 말하자면 이 라이브러리가 사용법이 간단하여 바로 적용하기는 쉽지만 현재 관리되고 있지 않는 라이브러리기에 확실한 동작을 보장받을 수 없어 사용하는 걸 적극 추천하지는 않습니다.

대신 python scapy 모듈을 사용할 것을 추천합니다.

위키피디아의 설명을 인용하면 scapy는 다음과 같습니다.

**Scapy** is a [packet](https://en.wikipedia.org/wiki/Packet_(information_technology)) manipulation tool for [computer networks](https://en.wikipedia.org/wiki/Computer_network),[[3]](https://en.wikipedia.org/wiki/Scapy#cite_note-pysysadmin-3)[[4]](https://en.wikipedia.org/wiki/Scapy#cite_note-mob-app-sec-4) written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) by Philippe Biondi. It can forge or decode [packets](https://en.wikipedia.org/wiki/Network_packet), send them on the wire, capture them, and match requests and replies. It can also handle tasks like scanning, [tracerouting](https://en.wikipedia.org/wiki/Traceroute), probing, [unit tests](https://en.wikipedia.org/wiki/Unit_testing), attacks, and [network discovery](https://en.wikipedia.org/wiki/Network_detector).

자세한 사용방법은 라이브러리를 관리하는 커뮤니티 주소를 남기겠습니다.

[https://scapy.readthedocs.io/en/latest/#](https://scapy.readthedocs.io/en/latest/)

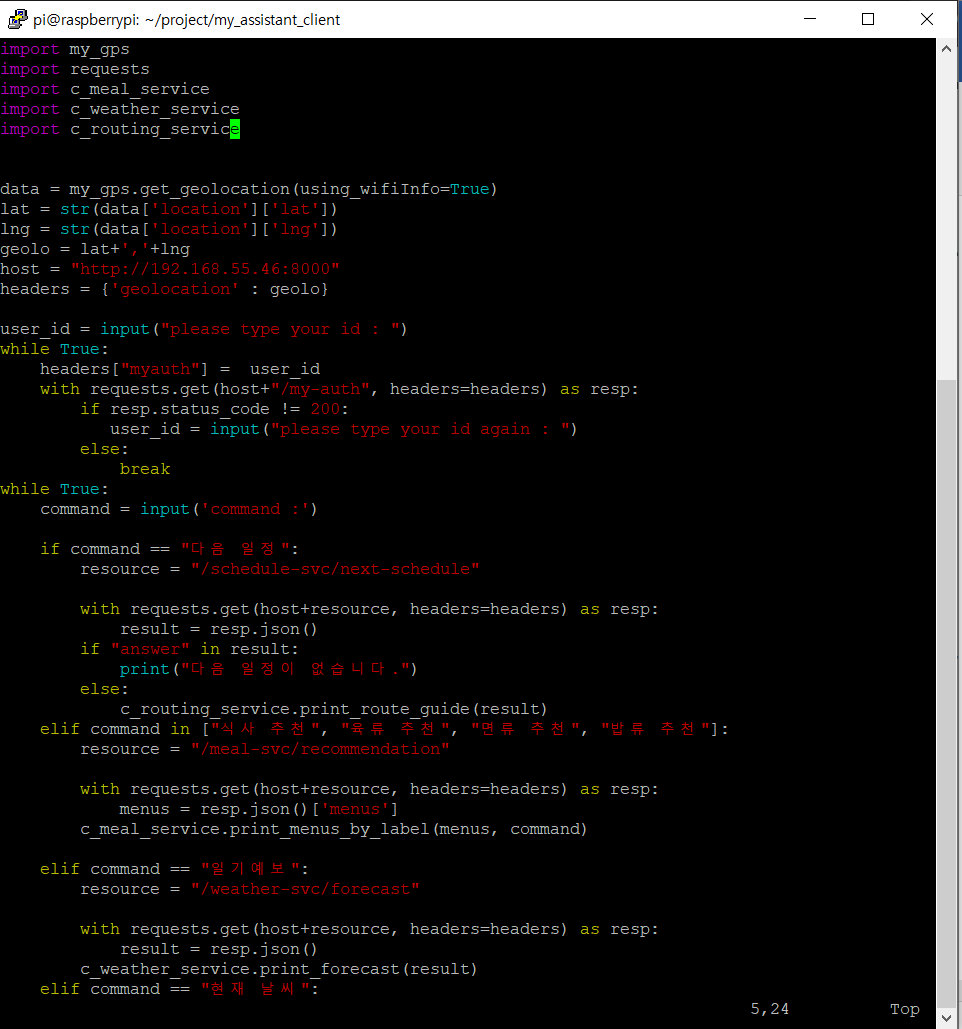
### **2.2.3. how does client work?**

클라이언트 안의 세부적인 모듈 설명은 여기까지 하고 전체적으로 사용자 입력을 받는 방식과 명령을 처리하는 법에 대해 설명하겠습니다.

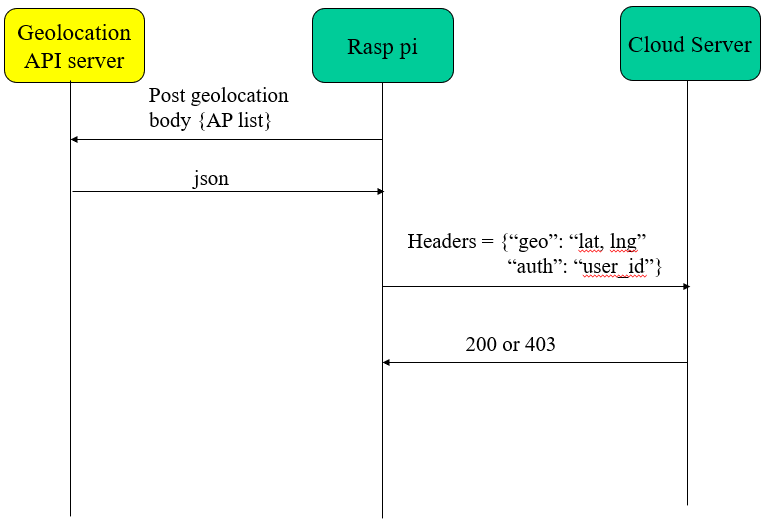
우선 동작방식은 임의의 명령어를 입력하면 그것에 해당하는 적절한 명령을 찾는 시스템이 아니라 사전에 text 형식으로 정해져 있습니다. 일기예보를 알고 싶으면

날씨 알려줘, 일기예보 알려줘 이런 것이 아니라

정확하게 사전에 프로그램에 정의 되어있는 “일기 예보” 라는 걸 정확하게 입력해야 합니다.



클라이언트가 실행되면 우선 GPS정보부터 구해 놓습니다. 그 후 클라우드 서버에 로그인 하기위해서 사용자 아이디를 입력 받습니다. 해당 아이디가 클라우드 서버내 데이터베이스에 존재하면 접속을 허락하는 식이다. 아직 로그인과 인증 시스템에 대해서 자세하게 공부하지 못해 이렇게 간략하게 구현하였습니다. 제가 한 방식으로 로그인을 구현하는 걸 추천하지 않습니다.



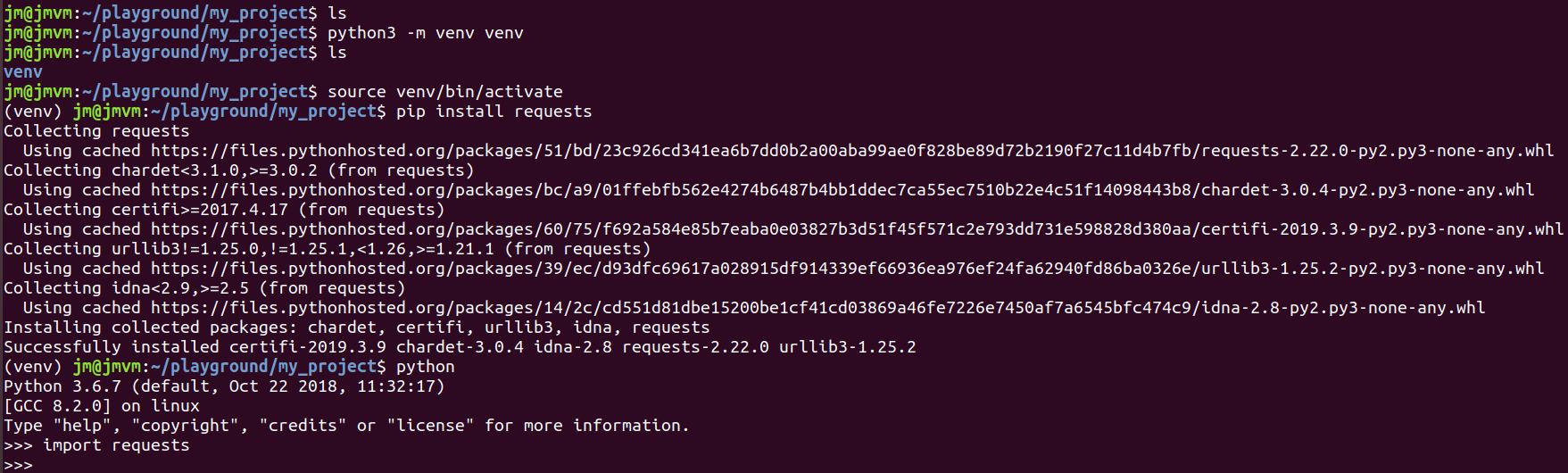
초록색 둥근 박스는 제 관리하에 있는 노드를 의미하고 노란색 둥근 박스는 외부 API Server를 의미합니다. http request body에 주변 AP list 정보를 보내면 json 형태로 GPS정보를 받고 이를 다시 헤더에 아이디와 함께 담아 cloud server로 request를 보내는 cloud서버에서는 헤더 있는 아이디가 데이터베이스 내에 존재하는 아이디이면 200을 반환하고 아니면 403 forbidden을 반환하여 응답합니다.

Test 동영상 링크: <https://youtu.be/NF-O7PxqOeE>

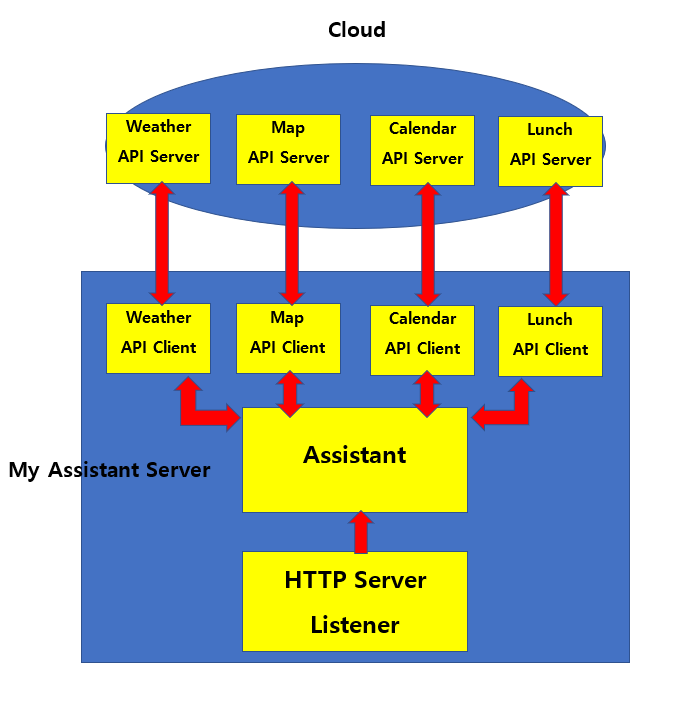
이때 http client는 python requests 모듈(라이브러리)를 이용하여 구현하였습니다. python에서 가장 인기있는 라이브러리 중 하나로 커뮤니티가 매우 활성화 되어있습니다. Python의 경우 pip라는 패키지 인스톨 메니지먼트가 존재하여 쉽게 라이브러리를 설치할 수 있어 아마 시간이 꽤 지나도 제가 기술한 라이브러리 설치 방법이 유효하게 작동할 가능성이 크지만 간혹 기기나 운영체제에 따라 라이브러리 설치가 제한되거나 방법이 다를 가능성도 존재하기 때문에 최신 official document를 참조하여 설치하는 것이 바람직하다고 생각합니다.

<https://2.python-requests.org/en/master/>

#### 2.2.3.1. Requests 라이브러리 설치 및 적용



## **2.3. server**



클라우드 서버 내부를 좀 더 자세히 보자면 먼저 http server listener가 사용자로부터의 요청을 대기합니다. 사용자로부터 request를 받으면 외부 API들을 조합하여 사용자가 원하는 답을 만들어 응답하는 서비스를 제공합니다.

현재 구현된 기능들을 하나하나 모듈별로 살펴보기 전에 cloud server를 구축할 때 사용한 python Django framework에 대해 간략하게 설명하도록 하겠습니다.

### **2.3.1. Django framework**

Django framework는 python으로 작성된 open source web application framework로 모델-뷰-컨트롤러(MVC)의 패턴을 따르고 있습니다.

자세한 설명과 사용 방법은 community의 official document를 읽어 보시는 걸 추천합니다.

<https://www.djangoproject.com/>

위의 official document 말고도 “파이썬 웹 프로그래밍, django로 배우는 쉽고 빠른 웹 개발” 이라는 책도 괜찮은 것 같습니다. 장고 그 자체에 관해서는 official 문서가 매우 좋고 정확하지만

전체적인 웹서버 제작 방식이나 python web library들의 구조들을 책에서 간략히 설명해 주기 한번 읽어보는 것도 괜찮은 것 같습니다.

일단 결론부터 말하자면 제 프로젝트에 딱 들어맞는 framework는 아니었던 것 같습니다.

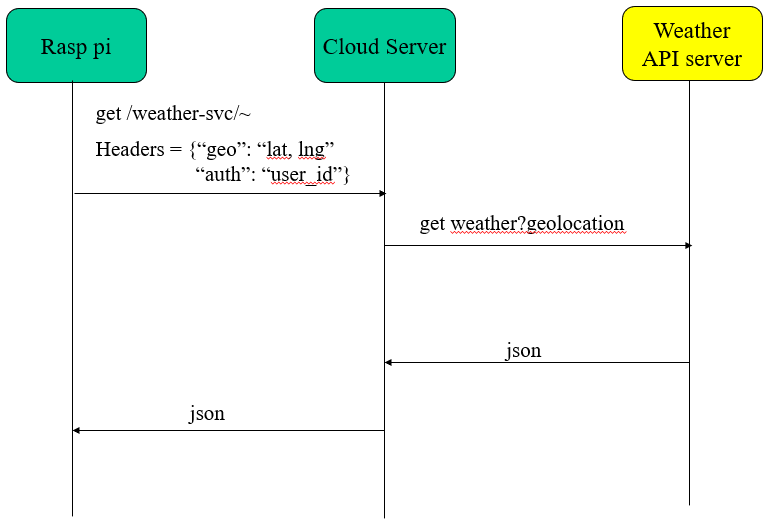
정형화된 design pattern이 존재하는 web application을 개발할 때는 Django framework를 이용하여 빠르게 구현할 수 있을 진 몰라도 제 프로젝트처럼 자유롭게 설계를 할 때는 좋은 framework는 아니라는 느낌을 받았습니다. 물론 이 부분은 제가 장고를 거의 겉 핥기 식으로 경험해 봤기에 부적절 할 수도 있습니다.

장고가 강력한 기능을 제공하는 대신 알아야하는 것들도 많고 또한 상당히 디자인 패턴이 정형화 되어있기 때문에 내가 자유롭게 할 수 있는 부분이 많이 제한되어 있다는 느낌을 받았습니다.

다음에 기회가 된다면 좀 더 제한이 없다는 flask framework를 이용해 볼 생각입니다.

### **2.3.2. 현재 날씨/일기예보**

최종 목표는 스마트폰 알람에 맞춰서 기상정보를 브리핑 해주는 것이었지만 현재는 단순히 “현재 날씨” 라는 명령어를 이용하여 현 시각 기상정보와 “일기예보”라는 명령어를 이용하여 앞으로 15시간 동안의 기상정보를 제공받는 기능이 구현되어 있습니다.



프로그램 실행 시 처음에 구해 두었던 GPS정보를 header에 담아 cloud server에서 제공하는 제가 만든 rest API를 이용하여 기상 정보를 요청합니다.

그러면 cloud server는 기상데이터를 제공하는 외부 API를 이용하여 사용자로부터 받은 위치정보에 해당하는 기상정보를 json 형태로 받아오고 이 데이터를 사용자 입맛에 맞게 가공하여 다시 json 형태로 클라이언트에게 보내게 됩니다. 응답을 받은 client는 사용자가 보기 좋게 요약하여 정보를 출력해줍니다.

Test 동영상 링크: <https://youtu.be/5rwye1u1E4s>

#### 2.3.2.1. Background

외부 기상 API는 해외 서비스인 openweathermap API를 이용하였습니다.

국내 날씨 정보를 얻을 수 있는 곳으로 기상청 날씨정보 API와 웨더아이 날씨 API가 있지만

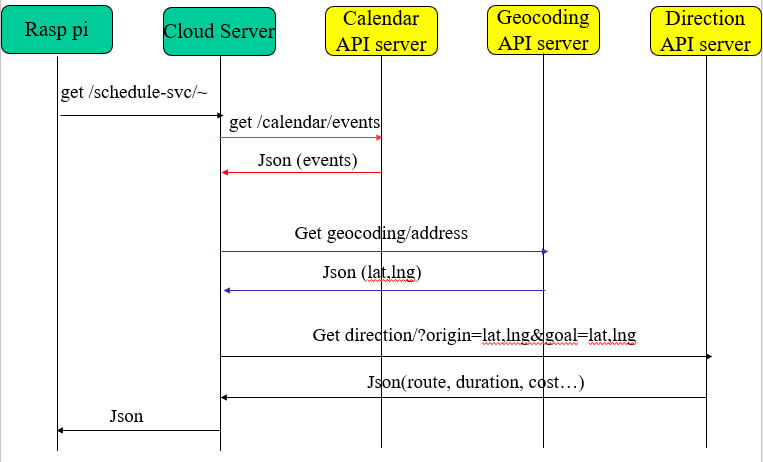
기상청의 경우 documentation이 부실하고 웨더아이의 경우 대기업에서 많이 이용하지만 기본적으로 유료사용이기에 해외 날씨 정보까지 제공하면서 무료로 사용할 수 있는 openweathermap API를 이용하였습니다.

<https://openweathermap.org/api>

### **2.3.3. 다음 일정**

이 기능은 다음 일정과 약속장소까지 걸리는 시간 및 경로를 알려주는 서비스입니다. 이것은 캘린더 API와 direction API를 조합하여 만들었습니다. 보통 캘린더에 일정을 등록하게 되면 일정 시작 시간 30분 전에 알람으로 알려주게 되지만 Direction API를 합치게 된다면 좀 더 효과적인 시간에 일정을 알려 줄 수 있게 됩니다.

예를 들어 약속장소에서 좀 떨어진 곳에서 일을 하고 있거나 공부를 하고 있어도 AI비서가 알아서 현재 위치에서 약속장소까지의 경로와 걸리는 시간을 계산하여 사용자에게 다음 일정에 늦지 않게 적절한 시간에 알람을 울려 줄 수 있습니다.



프로그램 실행 시 처음에 구해 두었던 GPS정보와 사용자 id정보를 header에 담아 서비스를 요청합니다. Cloud server는 헤더를 확인하여 사용자를 식별하고 해당 사용자의 캘린더에서 다음 일정을 가져옵니다. 이 일정에서 location 정보를 추출해 냅니다. (빨간색 화살표들로 표시된 부분입니다.) 이 location은 direction API에서 목적지 값으로 쓰이게 될 것입니다. 허나 주소를 바로 argument로 넘길 수는 없고 geocoding API를 이용하여 주소를 위도, 경도 형태의 GPS 정보로 바꿔야 합니다. (파란색 화살표 부분입니다.) 이제 최종적으로 naver에서 제공하는 direction API를 이용하여 경로를 계산하게 됩니다. 이때 argument로 현재 위치 정보와 목적지(약속장소)의 위치정보를 넘겨주면 됩니다. (검정색 화살표 부분입니다.) 그러면 최종적으로 모든 정보를 요약하여 json형태로 클라이언트에게 넘겨주고 이를 사용자에게 출력해줍니다.

Test 동영상 링크: <https://youtu.be/A9l198dhBH0>

#### 2.3.3.1. Background

##### 2.3.3.1.5 OAuth 2.0

OAuth2.0에 관하여 간략하게 예를 들어 설명하겠습니다. 어떤 사용자가 있고 제가 서비스 제공자이고 사용자의 개인정보는 구글 데이터베이스 센터에 저장되어 있다고 가정합니다.

이때 저는 구글 데이터베이스 센터에 있는 사용자의 개인정보를 이용하여 사용자에게 서비스를 제공하고 싶습니다. 이때 사용자가 저에게 구글에 있는 사용자 개인정보에 접근할 수 있는 권한을 줘야 하는데 이걸 안전하게 하는 방식을 하나의 프로토콜로 설계한 것입니다.

개인적으로 한글로 잘 정리 되어있는 자료는 없었고 OAuth wiki나 “OAuth 2 in action :OAuth 아키텍처에 대한 모든 것 “ 이라는 책의 1,2장 까지만 읽으면 대략 OAuth가 무엇이고 어떤 식으로 동작하는지 파악 하실 수 있으실 겁니다. 기본적으로 OAuth 2.0프로토콜을 이용할 때 사용자가 클라이언트를 처음부터 다 만들 필요 없이 구글에서는 자체적으로 OAuth 2.0 클라이언트를 제공하므로 안전한 통신을 위해서 API 서버 측에서 제공하는 Client를 이용하는 것이 좋을 것 같습니다. <https://developers.google.com/identity/protocols/OAuth2>

#### 2.3.3.2. Issues

##### 2.3.3.2.1 Google direction 한국 지원 x

처음에는 google direction API를 이용하려고 하였습니다. 간단히 client를 만들어 테스트했을 때 정상적으로 작동하였기 때문입니다. 허나 실제 프로그램으로 구현할 때 문제점이 발견되었는데 국내에서는 정상적으로 작동하지 않는 다는 것이었습니다. 이유는 국내법상 지도반출이 되지 않기 때문에 현재 구글에서는 경로 서비스를 국내에서 이용할 수 없었습니다.

대신에 네이버 direction API를 이용하게 되었는데 문제가 현재 자동차 경로밖에 안내하지 못합니다.

구글 같은 경우에는 대중교통 및 도보를 이용한 경로 안내도 추가로 제공하고 있지만 네이버는 현재는 제공하지 않고 있습니다. 네이버 클라우드 플랫폼에 안내에 따르면 조만간 제공할 예정이라고 써 있기 때문에 언젠가는 네이버에서도 자동차 경로 이외에 대중교통 경로도 안내 받을 수 있을 것이라 예상되긴 합니다.

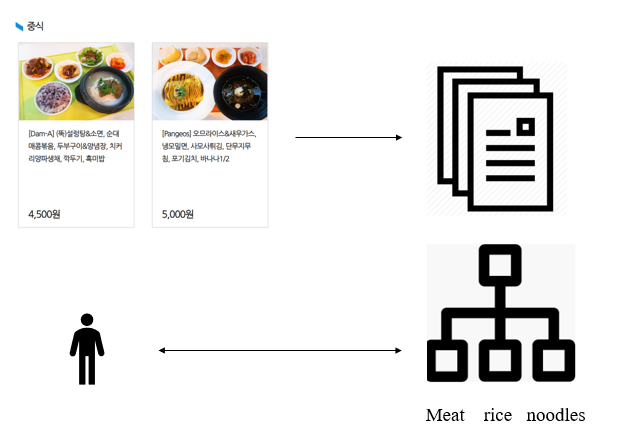
추가적으로 네이버 direction API는 출발지와 목적지 정보를 geolocation 형태로 받고 있습니다. 즉 주소 기반의 위치정보를 geocoding API를 이용하여 위도, 경도 형태로 변환할 필요가 있습니다.

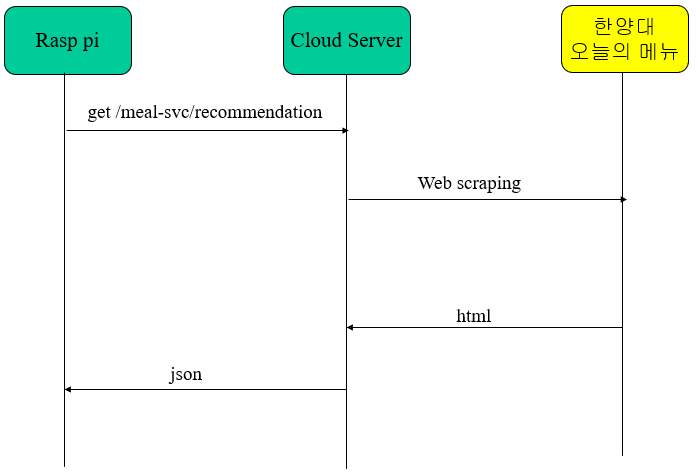
##### 2.3.3.2.2. OAuth 관련

처음 계획은 사용자를 클라우드 서버에 등록하는 과정 즉 회원 가입하는 기능까지 구현하려고 했습니다. 허나 모든 사용자 계정은 구글 캘린더와 연동이 되어야 하는데 OAuth 인증 프로세스에서 꼭 브라우저를 사용해서 사용자에게 허가를 구합니다. 예를 들어 브라우저에 구글 계정에서 로그인하고 해당 application에 권한 위임에 동의하는 지를 물어봅니다.

그래서 raspberry 상에서 회원가입 하는 과정을 보여주려면 일반 terminal이 아닌 VNC Viewer를 이용해야 하고 그 안에서 브라우저도 실행시켜야 하는데 몇 번 써본 결과 raspberry 하드웨어 상의 제한인지는 몰라도 만족스러운 반응속도를 얻기 힘들어 구현하지 않았습니다. 스마트폰이나 다른 브라우저가 원활히 실행될 수 있는 고성능 기기에서는 충분히 구현 가능 할 것으로 보입니다.

### **2.3.4. 식사 추천**





현재는 단순히 한양대 오늘의 메뉴 웹페이지에서 각 메뉴를 scraping 합니다.

각 메뉴들의 이름을 기준으로 메뉴의 종류를 육류 밥류 면류로 분류합니다.

그 후 사용자는 육류 추천해 달라고 하면 고기가 메인 메뉴인 식사를 추천해주는 형식입니다.

Test 동영상 링크: <https://youtu.be/oJx7_FCTpbc>

#### 2.3.4.1 Background

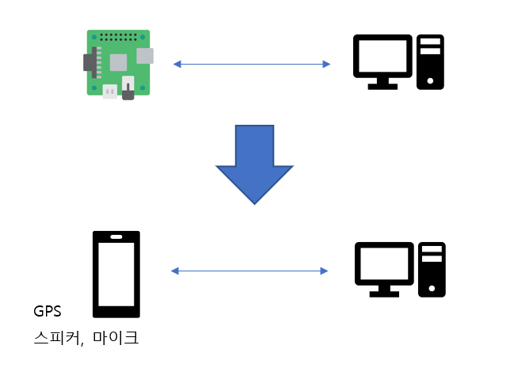
##### 2.3.4.1.1. Beautiful soup library

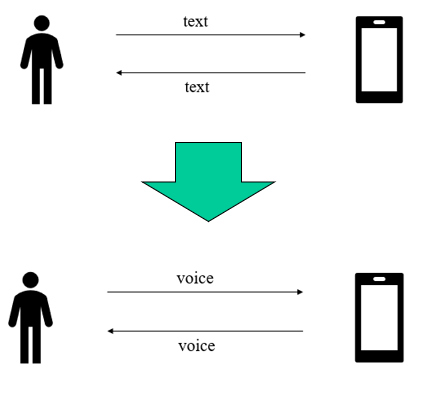
웹 scraping을 쉽게 해주는 python module(library)입니다.

<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

# **3. Future works**

## **3.1. change device from raspberry pi to smart phone**



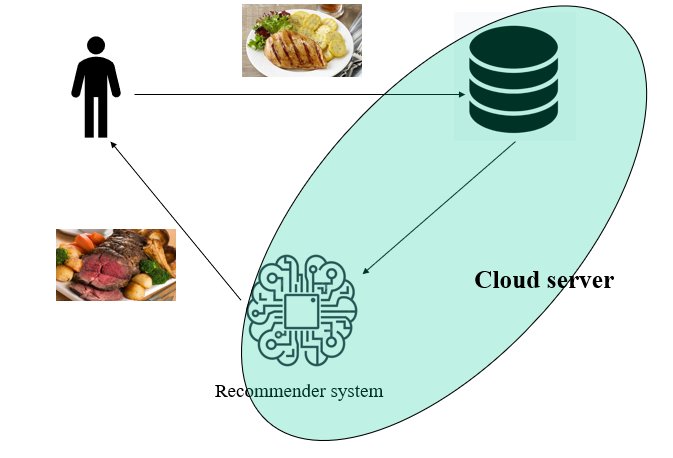


현재는 안드로이드 프로그래밍을 못해서 클라이언트 디바이스를 raspberry pi를 사용하고 있지만

이를 스마트 폰으로 대체할 생각입니다.

대체하게 되면 더 정확한 GPS정보와 스피커 마이크를 활용한 음성으로 명령을 입력 받고 응답 받게 할 생각입니다.

## **3.2 recommender system for meal service**



현재 식사 추천이 단순히 학식 메뉴 긁어와서 간략히 분류한 후 추천하는 형식인데 나중에는 사용자가 먹은 음식이나 선호하는 음식을 스마트폰 카메라로 찍어 이 데이터를 클라우드 서버에 저장하고 클라우드 서버내에서는 사용자가 선호하는 음식들을 트레이닝 셋으로 하여 추천시스템을 만들어 좀 더 사용자가 좋아할 법한 메뉴를 추천하는 식으로 현재 기능을 개선할 생각입니다.