캡스톤 디자인

출차 시간 알림 서비스

이 리 로

Android 송하은

Device 유종현

Server 윤덕우 윤영서

Role of Team Member

캡스톤 디자인 6팀

송하은

2020112133

기획 및 디자인

안드로이드 개발

유종현

2020112139

하드웨어

AWS 환경 구축

윤덕우

2020112148

DB 설계

아파트 및 차량 API

윤영서

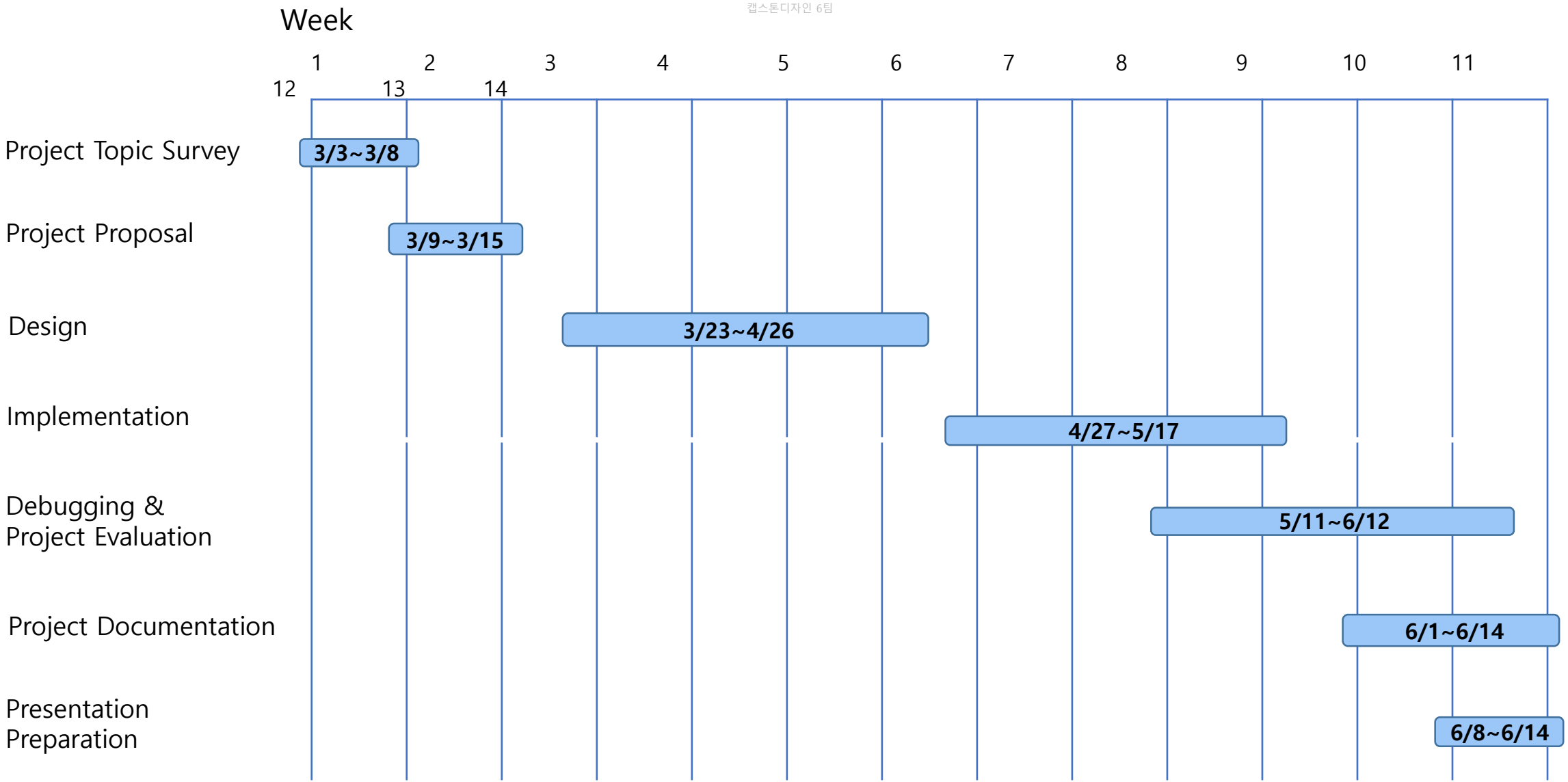
2020112155

영상 편집

회원 및 신고하기 API

Project Schedule

캡스톤디자인 6팀



BackGround

캡스톤 디자인 6팀

'사건반장' 제주대학교병원 주차장에서 이중주차 시비, 피해자 측 "암 치료 중인 환자를 죽으라며 28 차례 들이받아"

진병훈 기자 | 승인 2018.12.10 16:44 | 댓글 제보하기 | 반론요청

전국 자동차 등록대수 2500만대 돌파…2명당 1대 보유

친환경차 등록 비중 5%…반도체 수급난에 신규등록은 감소

2022.04.13 국토교통부

우리나라 자동차 등록대수가 2500만대를 돌파했다. 국민 2명 중 1명은 자동차를 보유하고 있는 셈이다.

국토교통부는 2022년 1분기 자동차 누적 등록대수가 2507만대로 전 분기 대비 0.6%(15만 9000대) 증가했다고 13일 밝혔다. 인구 2.06명당 1대의 자동차를 보유하고 있는 것이다.

수도권

"주차딱지 왜 붙여" 아파트 주차장 입구 막은 입주민 벌금형

이중구 기자 구독 + 입력 2021.07.08 10:36 수정 2021.07.08 10:40

재판부, 벌금 150만원 선고

많이 본 뉴스

1 [단독] 김만배도 입 열었다 "뇌물 영수, 200억 부동산 요구"

2 살인 피해 유족들에 물었다...'해자에 사형 선고된다면?"

3 11년 전 은하 3호 이틀 만에 추락했는데, 北 발사체 인양은 왜

4 푸틴, 핵 재앙 공포 노렸나... 호우카댐 파괴로 원전 비상

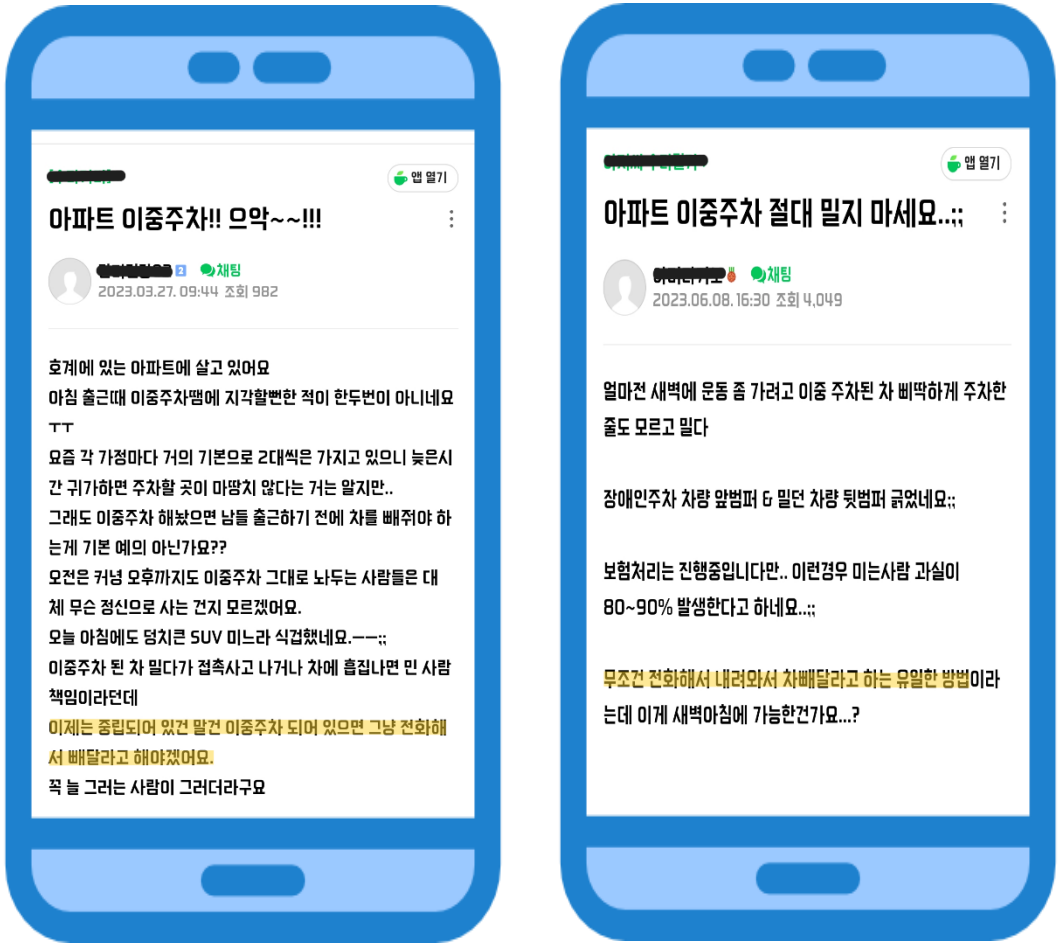
5 한국에는 개가 너무 많다

사회 많이 본 뉴스

자동차 등록대수에 비해 협소한 주차공간으로 이중주차 불가피한 상황 & 이중주차로 인한 갈등상황 심각

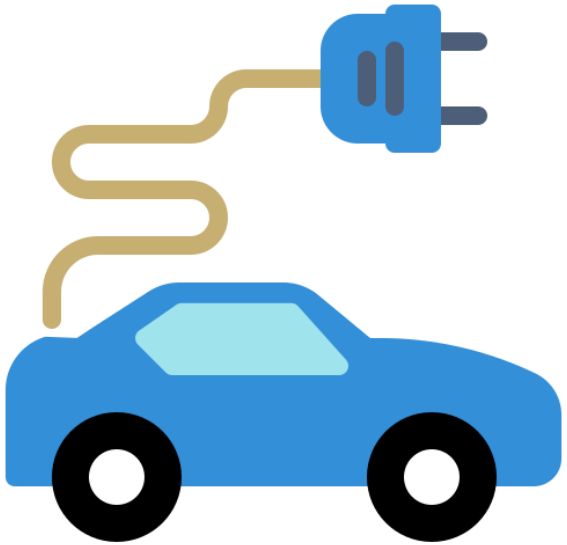
BackGround

캡스톤 디자인 6팀



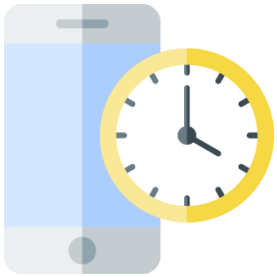
많은 사람들이 이중주차된 차량을 직접 밀기를 꺼림

전기차의 경우 중립주차가 안되는 차종이 많고
무거운 무게로 인해 밀기가 힘듦



Solution

캡스톤 디자인 6팀



출차 시간 인지

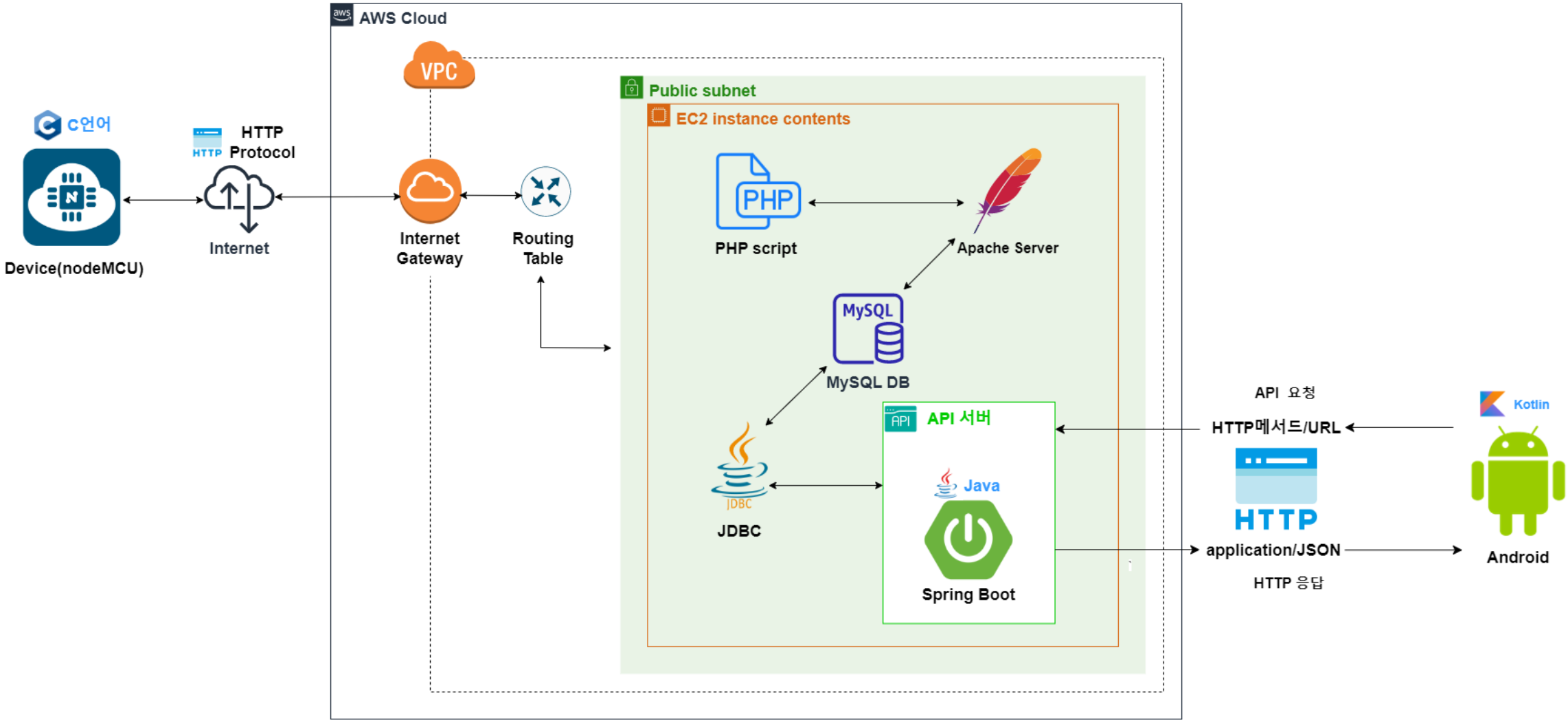
이중 주차

RFID를 이용한 주변 차량 출차 시간 알림 서비스

‘이리로’

System Overview

캡스톤 디자인 6팀



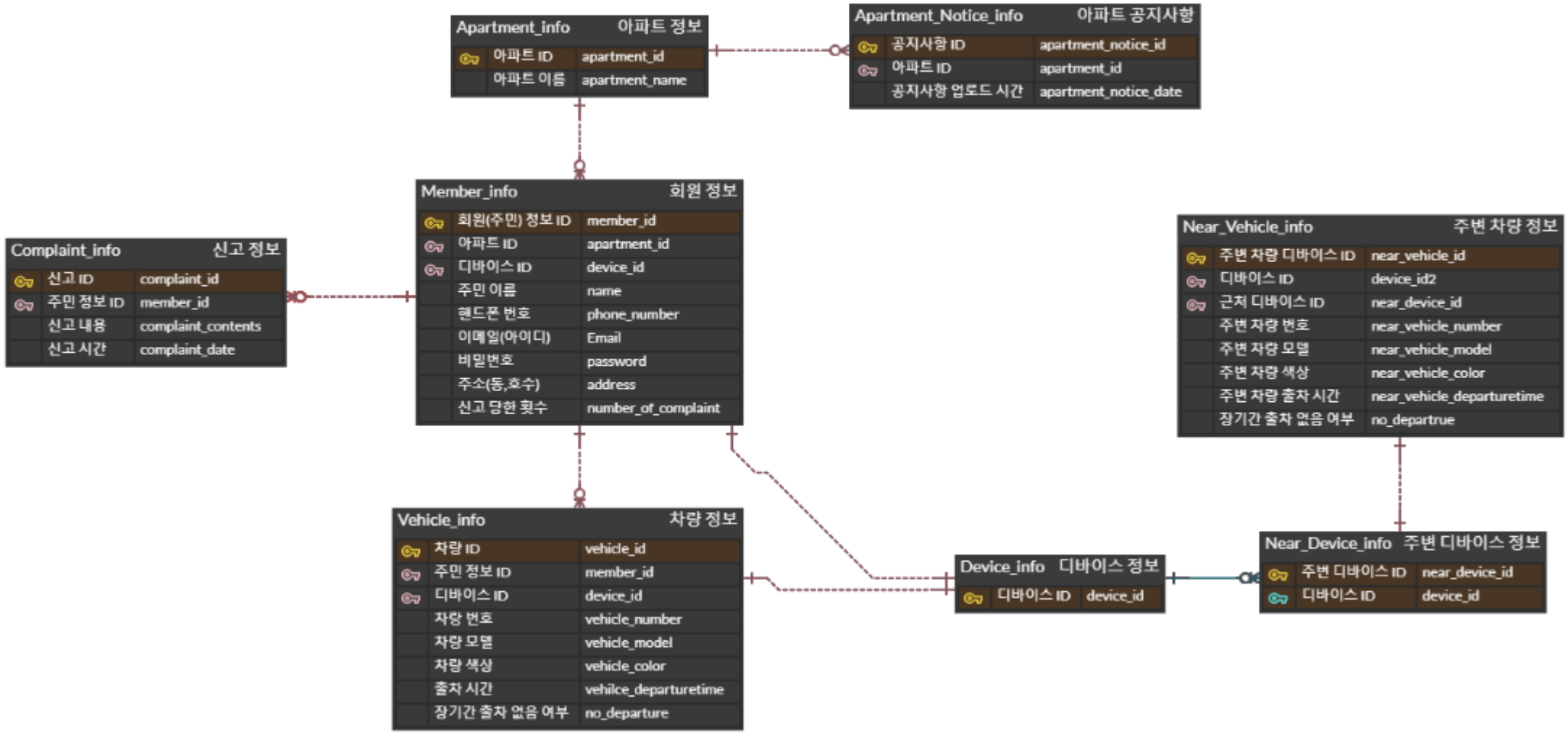
기능 명세서

캡스톤 디자인 6팀

화면	분류	기능 목록	순위	URI	Method
로그인	회원가입	로컬 회원가입	2	/members	POST
	로그인	로컬 로그인	2	/members/{id}	POST
메인페이지	신고 내역	신고 내역 조회 - 횟수	3	/members/{id}/complaints	GET
		신고 내역 조회 - 사유	3	/members/{id}/complaints	GET
	출차 시간	주소 조회	2	/members/{id}	GET
		출차 잔여 시간 조회	2	/vehicle/departuretime	GET
		출차 시간 조회	2	/vehicle/departuretime	GET
		출차 시간 등록 - 시간 등록	2	/vehicle/departuretime	POST
		출차 시간 등록 - 장기 주차 등록	2	/vehicle/departuretime	POST
	공지사항	아파트 공지사항 조회	3	/apartments/{id}	GET
주차 시작	출차 시간	출차 시간 등록	1	/vehicle/departuretime	PUT
	출차 정보	주변 차량 출차 정보 조회 - 차량 색	1	/nearvehicles	GET
		주변 차량 출차 정보 조회 - 차종	1	/nearvehicles	GET
		주변 차량 출차 정보 조회 - 차량 번호	1	/nearvehicles	GET
		주변 차량 출차 정보 조회 - 출차 시간	1	/nearvehicles	GET
		차량 조회 새로고침	1	/nearvehicles	GET
마이페이지	회원정보	이메일	3	/members/{id}	GET
		차량 번호	3	/members/{id}	GET
		주소	3	/members/{id}	GET
		누적 신고	3	/members/{id}	GET
	공지사항	아파트 공지사항 조회	3	/apartments/{id}	GET
	신고하기	신고하기 - 신고 대상	3	/members/{id}/complaints	POST
		신고하기 - 신고 사유	3	/members/{id}/complaints	POST
	디바이스	수신 시 LED 표시	3		

ERD

캡스톤 디자인 6팀



디자인

캡스톤 디자인 6팀

Color System

LOGO

Typeface

Typeface/Inter/title

Typeface/Inter/title

Typeface/Inter/title

I.A

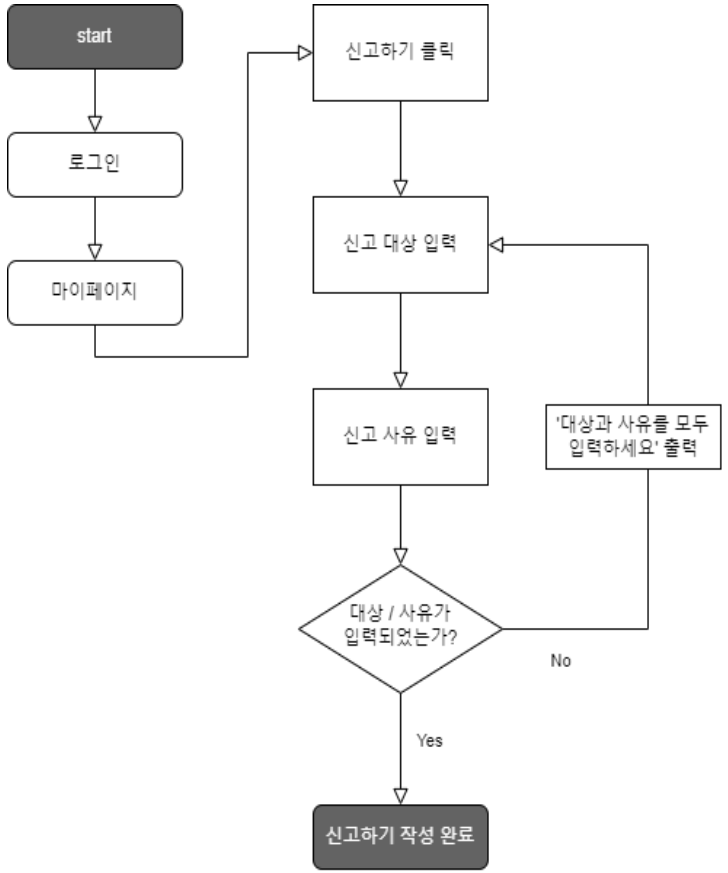
캡스톤 디자인 6팀


스플래시
회원가입 및 로그인
차량 정보 입력
아파트 정보 입력
홈
출차 시간
잔여 시간
신고 내역
출차 시간
신고 횟수
신고 사유
출차 시간 등록
출차 시간
공지사항
장기 주차 설정
주차시작
출차 시간 등록
출차 시간
장기 주차 설정
차량 출차 정보 조회
주변 차량 출차 시간
차종
차량 색
차량 번호
출차 시간
새로고침
마이페이지
회원 정보
이메일
차량 번호
주소
누적 신고 횟수
고객센터
공지사항
신고하기
신고 대상
신고 사유

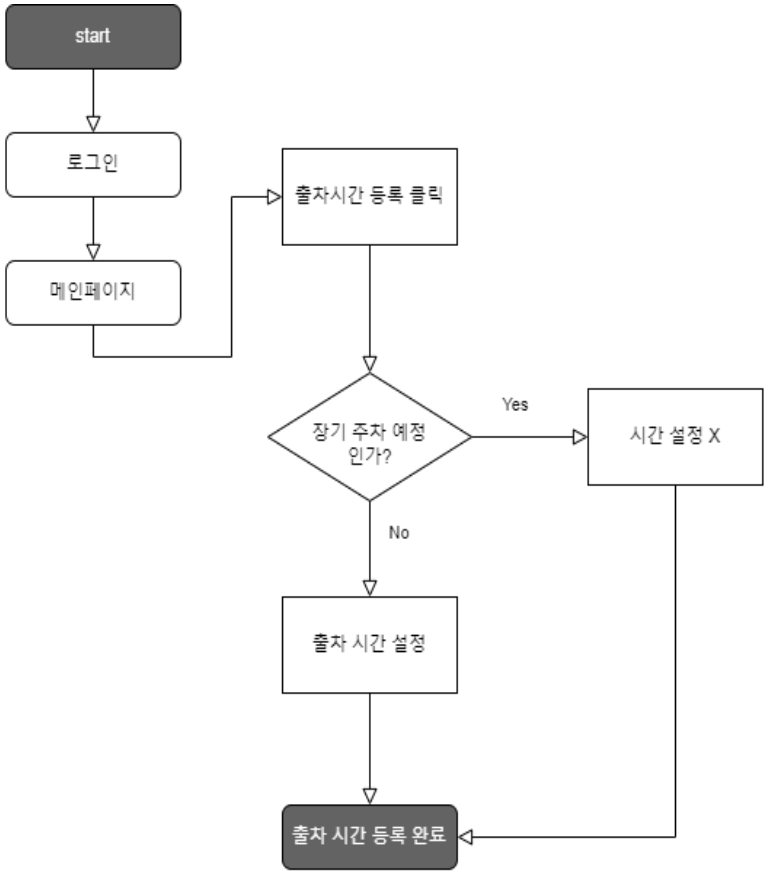
Flow Chart

캡스톤 디자인 6팀

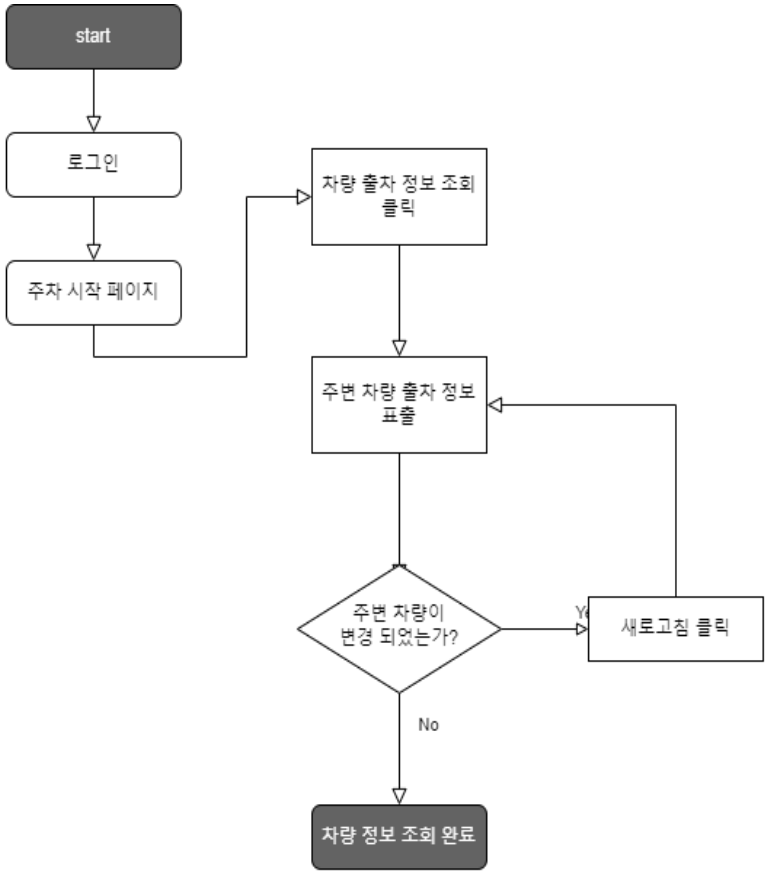
#신고하기 Flow



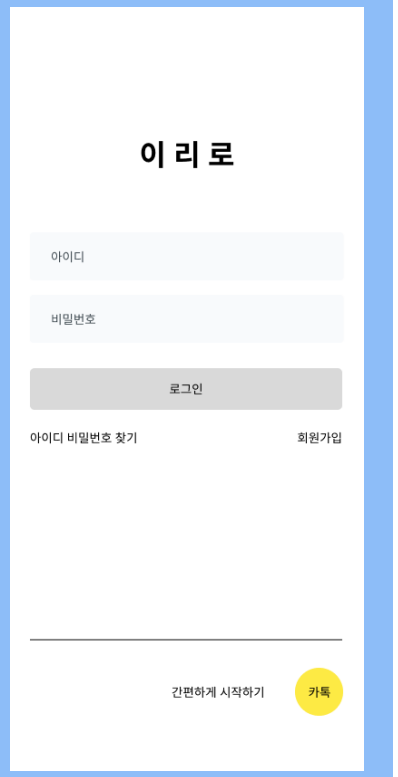
#출차 시간 등록 Flow

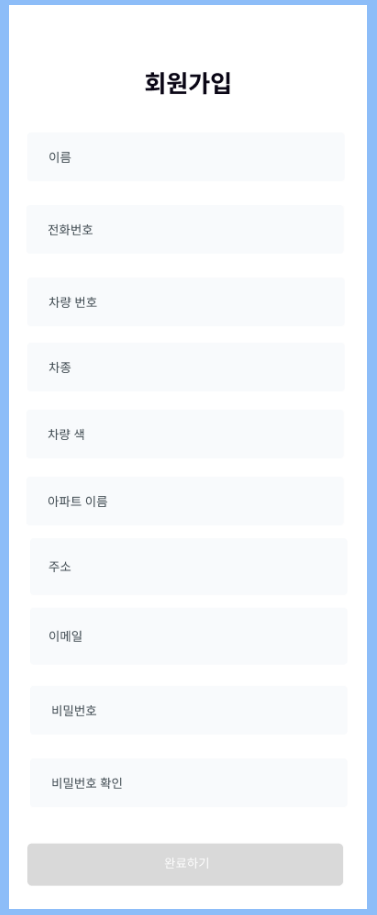


#주변 차량 조회Flow

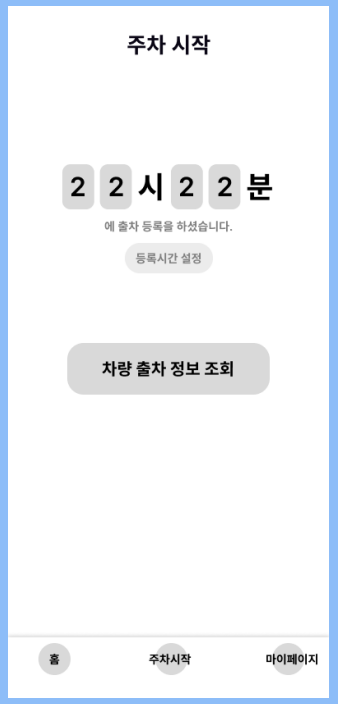


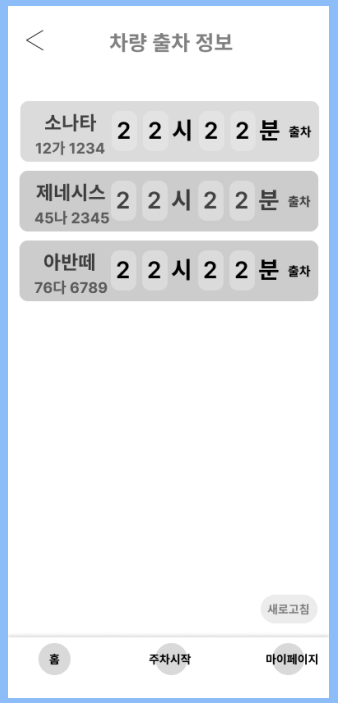
Wireframe

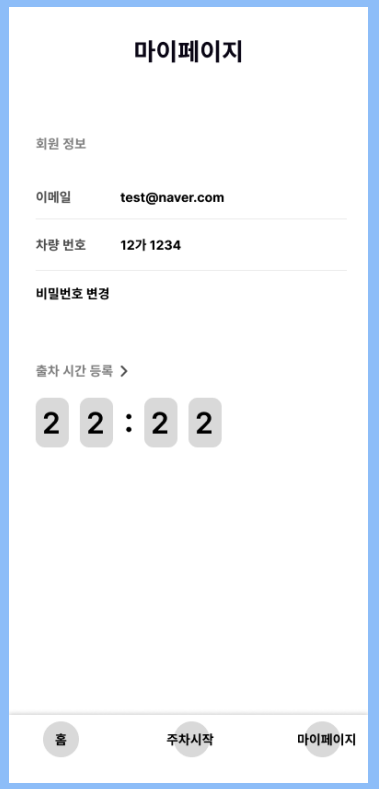

이 리 로
아이디
비밀번호
로그인
아이디 비밀번호 찾기
회원가입
간편하게 시작하기
카톡

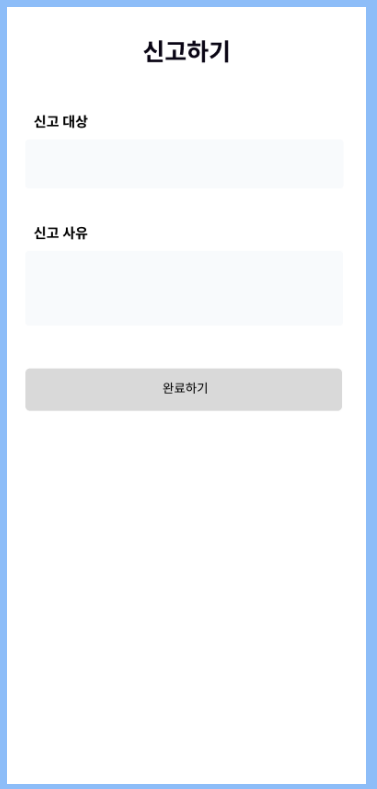

회원가입
이름
전화번호
차량 번호
차종
차량 색
아파트 이름
주소
이메일
비밀번호
비밀번호 확인
완료하기


오전 8시 30분
출발 예정
1시간 19분
남은 시간
서로를 위해 시간을 지켜주세요
오늘 하루도 힘내세요!
홈
주차 시작
마이페이지


주차 시작
2 2 시 2 2 분
에 출차 등록을 하셨습니다.
등록시간 설정
차량 출차 정보 조회
홈
주차시작
마이페이지


차량 출차 정보
소나타
12가 1234
2 2 시 2 2 분 출차
제네시스
45나 2345
2 2 시 2 2 분 출차
아반떼
76다 6789
2 2 시 2 2 분 출차
새로고침
홈
주차시작
마이페이지


마이페이지
회원 정보
이메일 test@naver.com
차량 번호 12가 1234
비밀번호 변경
출차 시간 등록 >
2 2 : 2 2
홈
주차시작
마이페이지


신고하기
신고 대상
신고 사유
완료하기

기술 스택

1. Device

- Arduino IDE
  - Development on - nodeMCU
- nodeMCU esp8266
  (wifi client)

2. Server

- Java 11
- Gradle
- Springboot 2.7.12
- MySQL(5.7)
- PHP(8.0)
- Apache
- AWS EC2

3. Client

- Kotlin(1.8.20)
- Android OS(version 9 이상)
- Gradle

기술 선택 (Apache, PHP, MySQL)

1. IoT 연결성

Apache 웹 서버는

NodeMCU IoT 장치와
웹 서비스 간의 통신을

쉽게 관리할 수 있게 함

2. 데이터 처리

NodeMCU에서 수집한 데이터

→ PHP를 통해 처리

→ MySQL 데이터베이스에 저장

→

웹 서비스에서
실시간 데이터 제공

3. 접근성과 편의성

PHP와 MySQL은 웹 개발에서
널리 사용되는 언어

→ IoT 프로젝트를 더 쉽게 개발하고
관리할 수 있음

시스템 구현

캡스톤 디자인 6팀

nodeMCU

433Mhz 송수신기

RFID

통신 실패 시 예외처리

데이터 중심 설계

RESTful API

MVC 패턴

Dto, Repository, Service 패턴

통신 및 각 기능 별 예외처리

MVVM 패턴

Retrofit2

Shared Preferences

통신 실패 시 및 각 기능 별 예외처리


Device
신고하기
신고 대상
신고 사유
차량 출차 정보
주차 시작
출차 시간 등록
오전 8시 30분
차량 출차 정보 조회

Device 구성 및 통신

캡스톤 디자인 6팀

요청 쿼리 처리 php 스크립트

```php
$conn = mysqli_connect($servername, $username, $password, $dbname);

$near_id = $_GET['nearid'];
$ID = $_GET['ID'];
$sql = "INSERT INTO near_device_info(near_device_id, device_id) VALUES($near_id, $ID)";
```

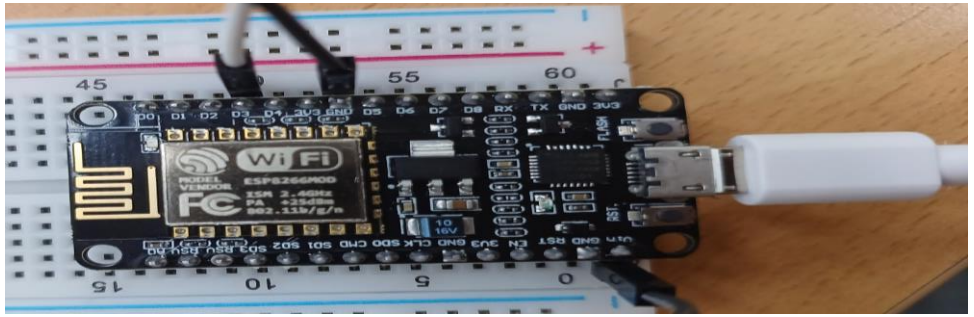
insert.php

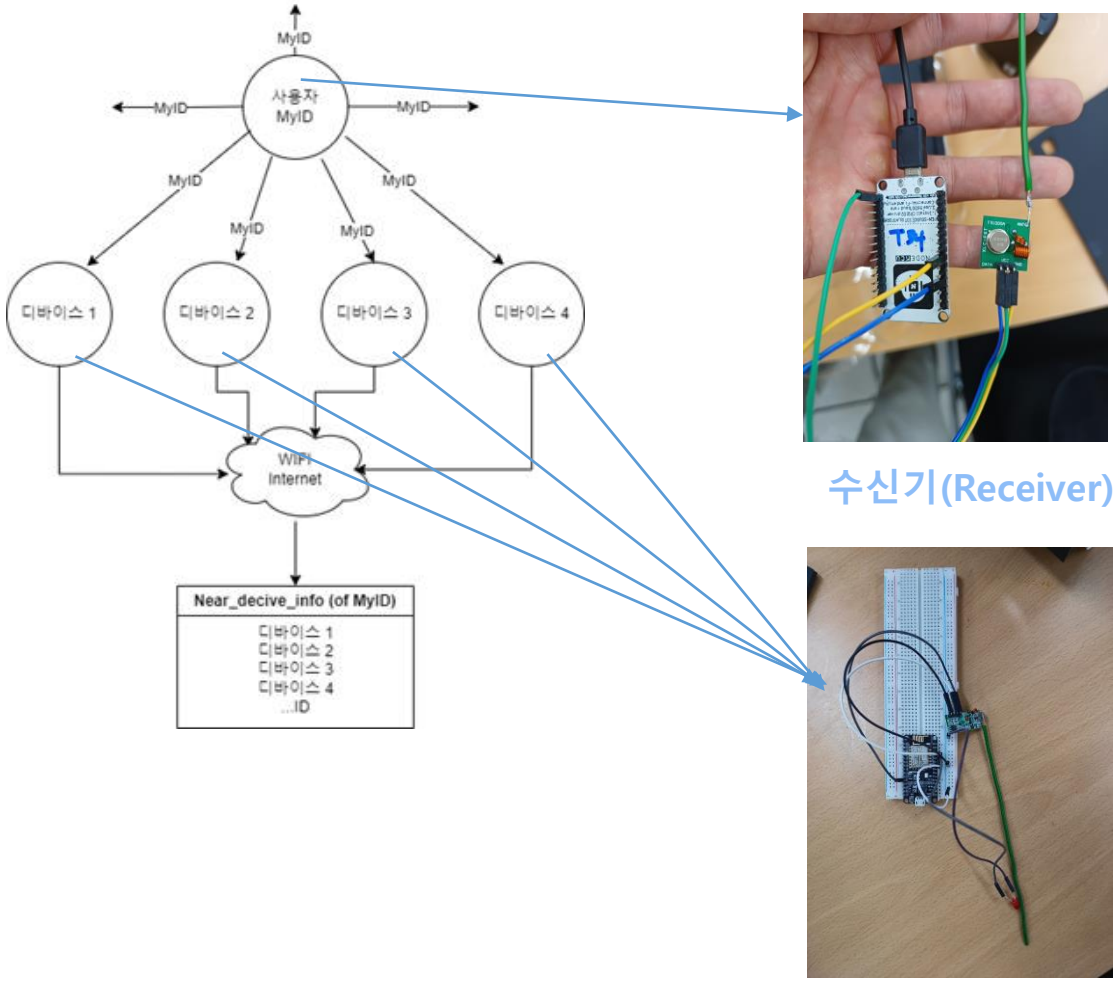
```php
$conn = mysqli_connect($servername, $username, $password, $dbname);

$near_id = $_GET['nearid'];
$ID = $_GET['ID'];
$sql = "DELETE FROM near_device_info where (near_device_id=$near_id &&device_id=$ID)";
```

delete.php

nodeMCU



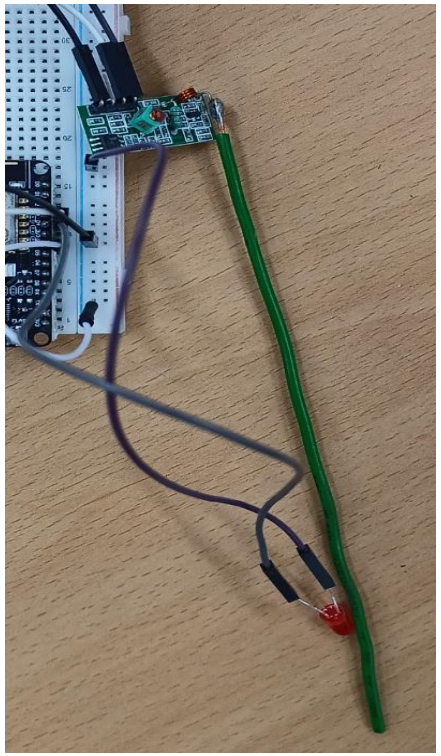


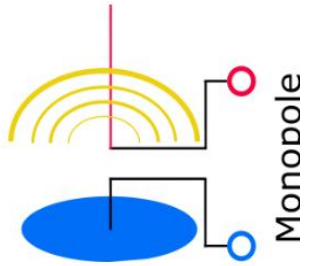
송신기(Transmitter)

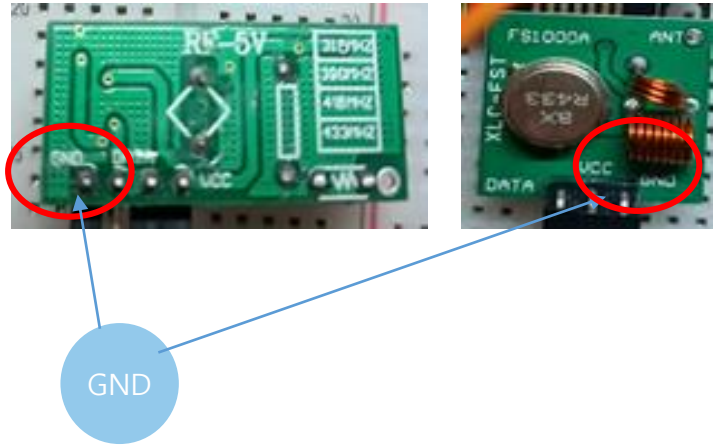
수신기(Receiver)

안테나의 구성 *Monopole Antenna*

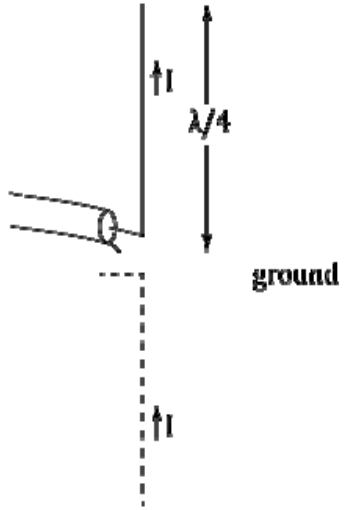
Monopole Antenna Design







Defining Antenna Length



$$c = 300 \times 10^6\ m/s$$
$$f = 433 \times 10^6\ m/s$$
$$\lambda = \frac{c}{f}$$
$$\lambda = 70cm$$
$$\lambda/4 = 17cm$$

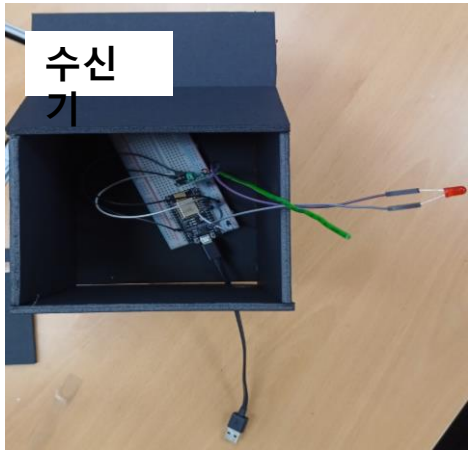
안테나의 구성

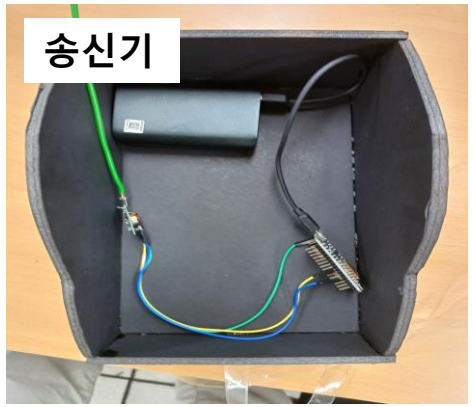
Active RFID

	Active RFID	Passive RFID
Tag battery	Yes	No
Tag power source	Internal to tag	Powered by reader via RF
Availability of tag power	Continuous	Only within field of reader
Required signal strength to tag	Low	High
Communication	Long range (100m or more)	Shot range (5m or less)
Sensor capability	Continuously monitor and record sensor input with date/time stamp	Read & transfer sensor data only when tag is powered by reader without date/time stamp

출처 :

https://www.researchgate.net/publication/265454664_RFID_technology_in_healthcare_and_mass_casualty_incidents





Device Source Code *Transmitter*

Library : RCSwitch.h

```
RCSwitch mySwitch = RCSwitch();

void setup() {
  Serial.begin(9600);
  // Transmitter is connected to Arduino Pin #0
  mySwitch.enableTransmit(0);
```

```
  mySwitch.send(ID, 24);
  Serial.println("sent!");
```

RCSwitch.h 내장 되어있는
send() 함수

Send(ID, 24) :
ID 라는 int값을 24bit로 표현

RCSwitch 라이브러리에 있는 전자파 정의

RCSwitch.cpp

C: > Users > test > Documents > Arduino > rc-switch > RCSwitch.cpp

```
58  /* Format for protocol definitions:
59   * {pulselength, Sync bit, "0" bit, "1" bit, invertedSignal}
60   *
61   * pulselength: pulse length in microseconds, e.g. 350
62   * Sync bit: {1, 31} means 1 high pulse and 31 low pulses
63   *     (perceived as a 31*pulselength long pulse, total length of sync bit is
64   *     32*pulselength microseconds), i.e:
65   *      _
66   *     | |_______________________________ (don't count the vertical bars)
67   * "0" bit: waveform for a data bit of value "0", {1, 3} means 1 high pulse
68   *     and 3 low pulses, total length (1+3)*pulselength, i.e:
69   *      _
70   *     | |___
71   * "1" bit: waveform for a data bit of value "1", e.g. {3,1}:
72   *      ___
73   *     |   |_
74   *
75   * These are combined to form Tri-State bits when sending or receiving codes.
76   */
```

433Mhz 송수신은 ASK로 이루어지므로 0, 1의 신호로 변형

수신을 위한 interrupt 구현 및 송신을 위한 프로토콜 정의

Device Source Code *RCSwitch Library*

Send()

Transmitt 함수 사용

```
for (int nRepeat = 0; nRepeat < nRepeatTransmit; nRepeat++) {
  for (int i = length-1; i >= 0; i--) {
    if (code & (1L << i))
      this->transmit(protocol.one);
    else
      this->transmit(protocol.zero);
  }
  this->transmit(protocol.syncFactor);
}
```

```
void RCSwitch::transmit(HighLow pulses) {
  uint8_t firstLogicLevel = (this->protocol.invertedSignal) ? LOW : HIGH;
  uint8_t secondLogicLevel = (this->protocol.invertedSignal) ? HIGH : LOW;

  digitalWrite(this->nTransmitterPin, firstLogicLevel);
  delayMicroseconds( this->protocol.pulseLength * pulses.high);
  digitalWrite(this->nTransmitterPin, secondLogicLevel);
  delayMicroseconds( this->protocol.pulseLength * pulses.low);
}
```

digitalWirte로 보드에 데이터 전송

getReceivedValue()

```
unsigned long RCSwitch::getReceivedValue() {
  return RCSwitch::nReceivedValue;
}
```

```
if (changeCount > 7) {    // ignore very short transmissions: no
    RCSwitch::nReceivedValue = code;
    RCSwitch::nReceivedBitlength = (changeCount - 1) / 2;
    RCSwitch::nReceivedDelay = delay;
    RCSwitch::nReceivedProtocol = p;
    return true;
}
```

```
void RCSwitch::enableReceive() {
  if (this->nReceiverInterrupt != -1) {
    RCSwitch::nReceivedValue = 0;
    RCSwitch::nReceivedBitlength = 0;
#if defined(RaspberryPi) // Raspberry Pi
    wiringPiISR(this->nReceiverInterrupt, INT_EDGE_BOTH, &handleInterrupt);
#else // Arduino
    attachInterrupt(this->nReceiverInterrupt, handleInterrupt, CHANGE);
#endif
  }
}
```

hanldeInterrupt로 저장된 nReceivedValue

Device Source Code *Receiver*

Library : ESP8266Client.h, ESP8266Wifi.h, RCSwitch.h

Interval의 차이

```
const long interval = 1000;
const long interval_d = 3000;
unsigned long previousMillis = 0;
```

삭제 요청보다 입력 요청 주기를 작게 설정

→ 정보 유실 방지

```
unsigned long currentMillis = millis();
if (currentMillis - previousMillis >= interval) {
  previousMillis = currentMillis;
  //near_device_id 송신 받은 값, device_id 내 아이디는 지정된 것
  String phpHost = host + "/insert.php?nearid=" + ID + "&ID=" + String(near_id_tmp);
```

```
unsigned long currentMillis = millis();
if (currentMillis - previousMillis >= interval_d) {
  previousMillis = currentMillis;
  String phpHost = host + "/delete.php?nearid=" + ID + "&ID=" + String(near_id);
```

쿼리문에 파라미터 전달을 위한 GET요청

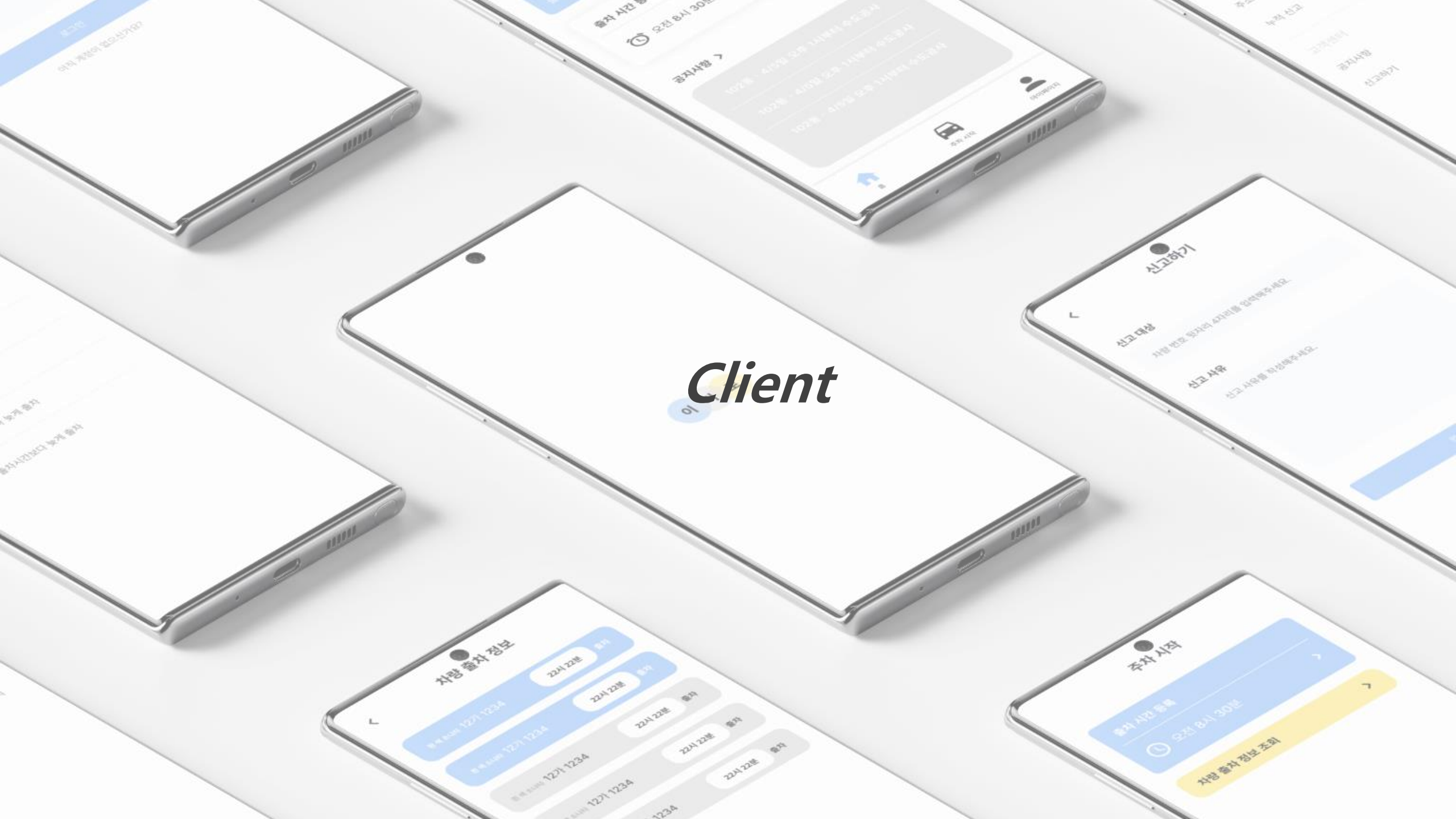
```
http.begin(client, phpHost);
http.setTimeout(1000);
int httpCode = http.GET();
```

HTTP Error 예외처리 – 송수신 모두 적용

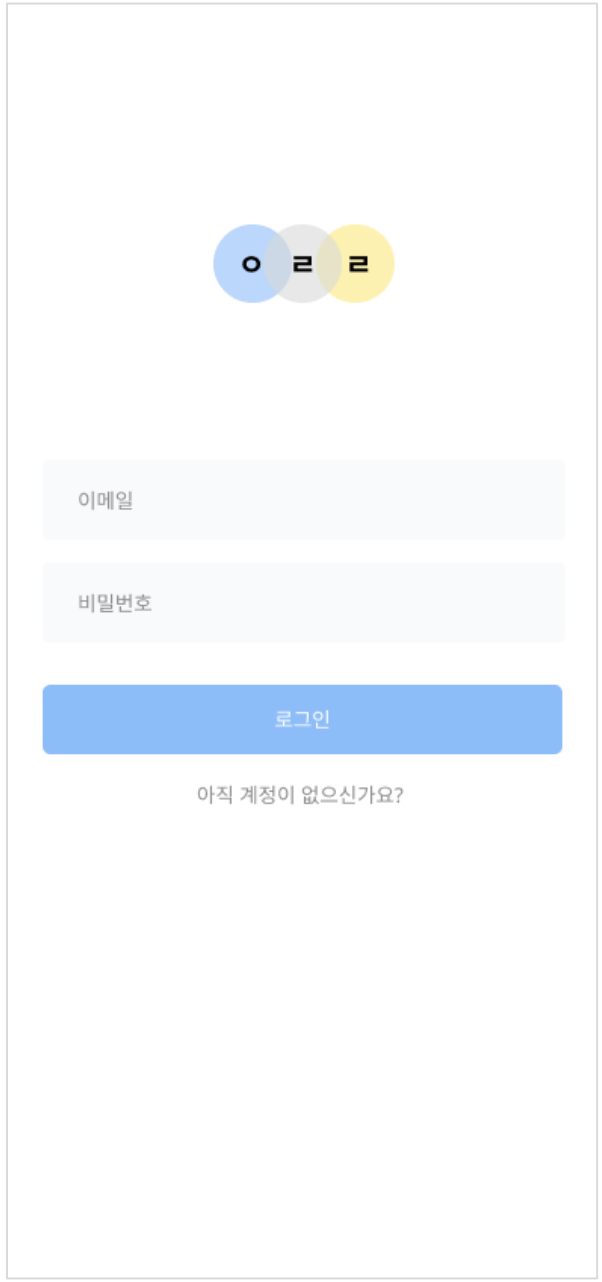
```
while(httpCode<0){
  httpCode = http.GET();
  Serial.println("not a good httpCode.");
}
```

정상 httpCode 결과값이 나올 때 까지 반복 요청

→ 쿼리문 요청 성공


Client
신고하기
신고 대상
신고 사유
주차 시작
차량 출차 정보 조회

구현 로그인



View

Class SignInActivity

Function onPostSignInSuccess() : 로그인 완료
onPostSignInFailure() : 예외처리

Service

Class SignInService

Function tryPostSignIn() : response 처리

Model

Class SignInReq() : 데이터 요청
SignInRes() : response
SignInResult() : response 형식 맞춤

구현 *회원가입*

View

Class SignUpActivity

Function onPostSignUpSuccess() : 회원가입 완료
onPostSignUpFailure() : 예외처리

Service

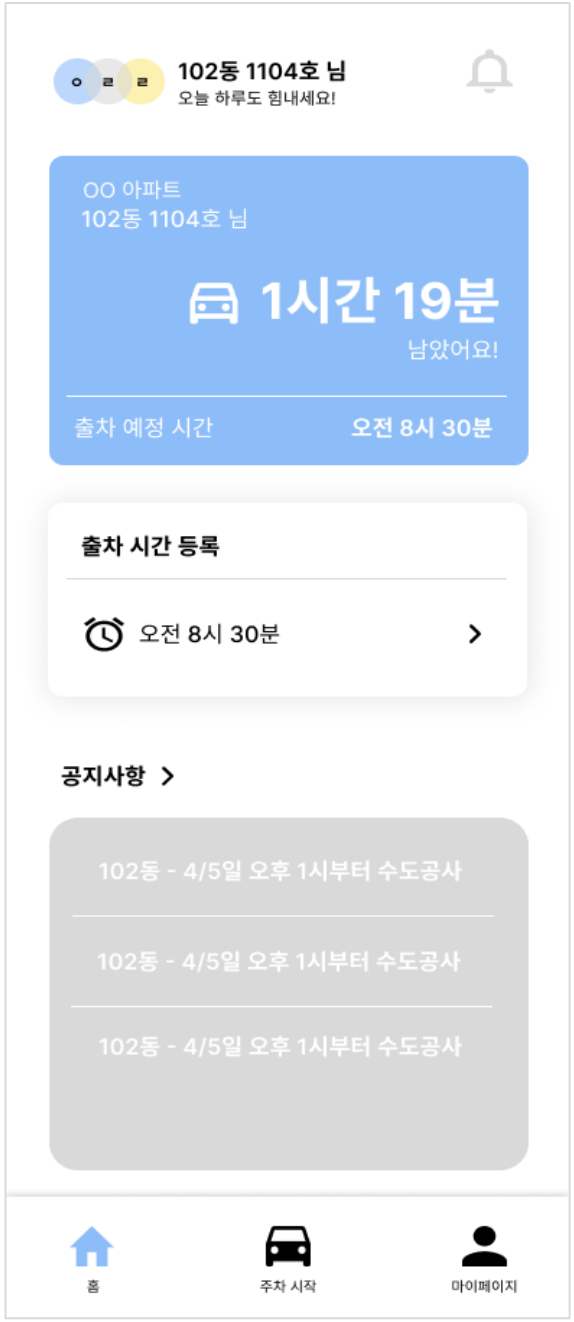
Class SignUpService

Function tryPostSignUp() : response 및 예외 처리

Model

Class SignUpReq() : 데이터 요청
SignUpRes() : response
SignUpResult() : response 형식 맞춤

구현 홈



View

Class HomeFragment

Function onGetMainSuccess() : 남은 시간 및 회원정보 조회
onGetMainFailure() : 예외처리

Service

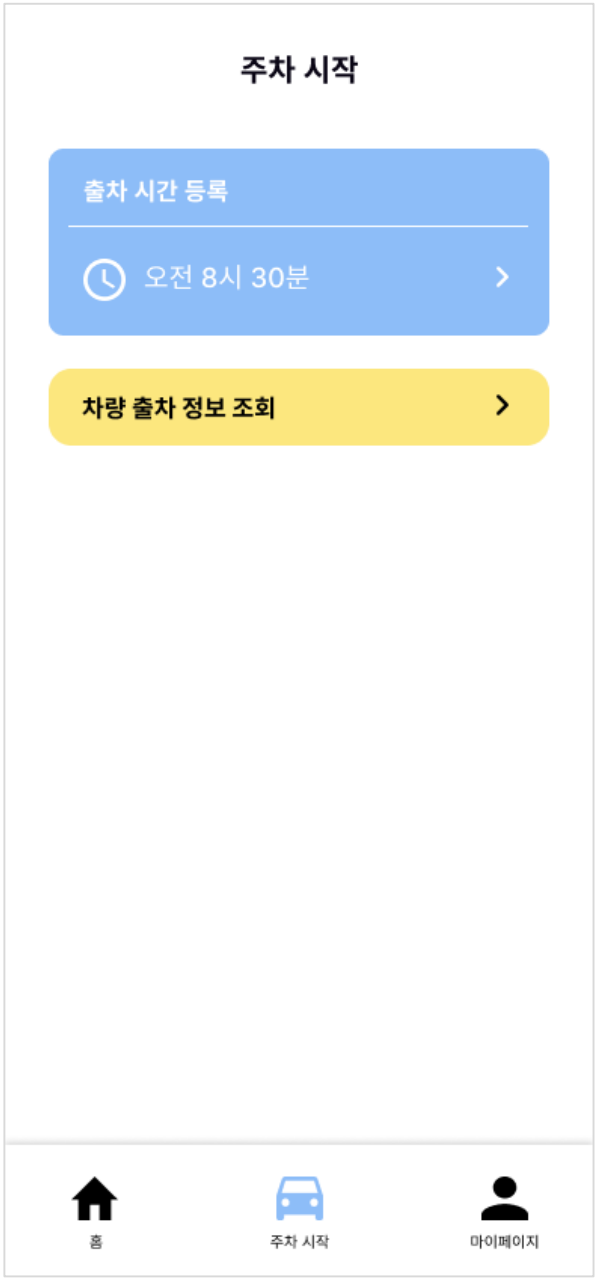
Class MainService

Function tryGetMain() : response 및 예외 처리

Model

Class MainReq() : 데이터 요청
BaseResponse() : response
MainResult() : 데이터 형식 맞춤

구현 *주차시작*



View

Class ParkFragment

Function
registerTime() : 출차 시간 설정
onPostRegisterSuccess() : 출차 시간 등록 완료
onPostRegisterFailure() : 예외처리

Service

Class ParkService

Function tryPostParkRegister() : response 및 예외 처리

Model

Class
RegisterReq() : 데이터 요청
BaseResponse() : response

구현 *차량 출차 정보*

View

Class
DepartureInfoActivity
DepareureListRVAdapter

Function
onGetNearVehicleSuccess() : 주변 차량 조회
onGetNearVehicleFailure() : 예외처리

Service

Class
ParkService

Function
tryGetNearVehicle() : response 처리

Model

Class
NearVehicleRes() : 데이터 요청
BaseResponse() : response
NearVehicleList() : response 형식 맞춤

구현 *마이페이지*

View

Class MyPageFragment

Function
onGetUserProfileSuccess() : 회원정보조회
onGetUserProfileFailure() : 예외처리
onGetLogOutSuccess() : 로그아웃
onGetLogOutFailure() : 예외처리

Service

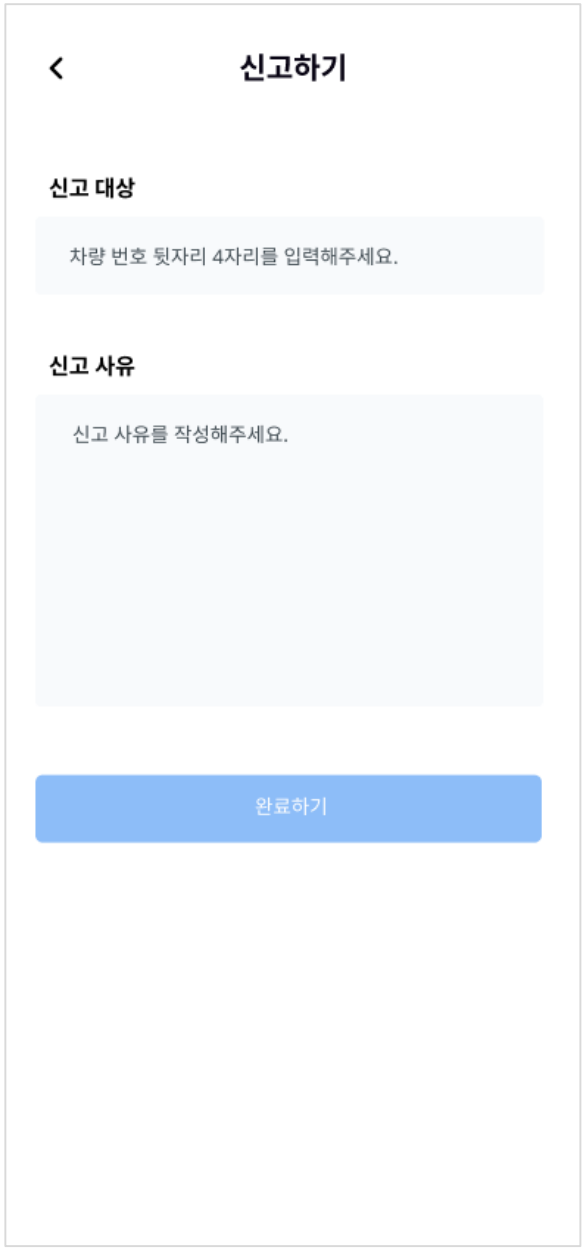
Class MyService

Function
tryGetUserProfile() : response 처리
tryGetLogOut() : response 처리

Model

Class
UserProfileRes() : response
UserProfileResult() : response 형식 맞춤
LogoutRes() : response

구현 *신고하기*



View	*Class*	DeclareActivity
	Function	onPostDeclareListSuccess() : 신고하기 완료 onPostDeclareListFailure() : 예외처리
Service	*Class*	DeclareService
	Function	tryPostDeclare() : response 처리
Model	*Class*	DeclareReq() : 데이터 요청 DeclareRes() : response

구현 *신고내역*

View

Class MemberDeclareListActivity
DeclareListAdapter

Function onGetDeclareListSuccess() : 신고 내역 조회
onGetDeclareListFailure() : 예외처리

Service

Class DeclareService

Function tryGetDeclareList() : response 처리

Model

Class DeclareListRes() : response
DeclareListResult() : response 형식 맞춤
Report(): 신고내역 Array

구현 *공지사항*

View

Class
NoticeActivity
NoticeListRVAdatper

Function
onGetNoticeListSuccess() : 공지사항 내역 조회
onGetNoticeListFailure() : 예외처리

Service

Class
NoticeService

Function
tryGetNoticeList() : response 처리

Model

Class
NoticeRes() : response
NoticeList() : 공지사항 Array


Server
신고하기
신고 대상
신고 사유
차량 출차 정보
주차 시작
출차 시간 등록
오전 8시 30분
차량 출차 정보 조회

API

apartment-controller Apartment Controller

- GET /apartment/{apartmentName} 아파트 공지사항 조회

member-controller Member Controller

- POST /members 회원가입
- GET /members/{id} 회원 정보 조회
- GET /members/{id}/complaints 신고 내용 조회
- POST /members/complaints 신고하기
- POST /members/login 로그인
- GET /members/logout 로그아웃

vehicle-controller Vehicle Controller

- GET /nearvehicles/{id} 주변 차량 정보 조회
- GET /vehicle/departuretime/{id} 출차 정보 및 주소 조회
- POST /vehicle/departuretime/{id} 출차 정보 등록
- PUT /vehicle/departuretime/{id} 출차 정보 수정

구현 *회원가입, 로그인, 로그아웃, 회원정보조회*

DTO

```
UserDto ⌄ {
    address             string
                        example: 103동 103호
    apartment_name      string
                        example: 푸르지오
    device_id           string
                        example: 111
    email               string
                        example: IRIRO@capstone.com
    name                string
                        example: 김정통
    password            string
                        example: password_IRIRO
    phone_number        string
                        example: 01011112222
    pw_check            string
                        example: password_IRIRO
    vehicle_color       string
                        example: 검정색
    vehicle_model       string
                        example: 마차
    vehicle_number      string
                        example: C123123
}
```

```
LoginDto ⌄ {
    email               string
                        example: IRIRO@capstone.com
    password            string
                        example: password_IRIRO
}
```

Controller

Class **MemberController**

Method

@PostMapping("/members")
register(@RequestBody UserDto userDto): 회원가입
: memberService.register를 호출

@PostMapping("/login")
login(@RequestBody LoginDto loginDto, HttpSession session): 로그인
: memberRepository.findByEmail(), vehicleRepository.findByMemberId를 호출

@GetMapping("/logout")

logout(HttpServletRequest request): 로그아웃

@GetMapping("/members/{id}")

getMemberById(@PathVariable Long id): 회원 정보 조회

: memberService.getMemberById를 호출

적절한 상태코드와 응답 메시지를 포함한 메시지를 생성, 해당 응답 반환

구현 *회원가입, 로그인, 로그아웃, 회원정보조회*

Service

Class

MemberService,
MemberServiceImpl

Method

register(UserDto userDto)
: userDto의 정보를 기반으로 회원 가입

getMemberById(Long Id)
: 회원 id를 이용하여 회원의 이메일, 차량 번호, 주소, 신고 당한 횟수를 가져옴

Repository

Class

MemberRepository,
MemberRepositoryImpl

Method

- **ApartmentRepository.findByApartment_name(String apartment_Name):** 아파트 이름 검색
- **MemberRepository.save(Member member):** 회원 정보 저장
- **VehicleRepository.save(Vehicle vehicle):** 차량 정보 저장
- **DeviceRepository.save(Device device):** 디바이스 정보 저장
- **MemberRepository.findByEmail(String email):** 이메일로 회원 검색
- **VehicleRepository.findByMemberId(Long userId):** 회원 id로 차량 검색
- **MemberRepository.findById(Long id):** 회원 id로 회원 검색

구현 *신고하기, 신고 내용 조회*

DTO

```
ReportDto ⌄ {
    complaint_contents    string
                          example: 등록한 출차 시간 이전에 차를 빼달라고 요구
    vehicle_number        string
                          example: B123123
}
```

Service

Class

MemberService,

MemberServiceImpl

Method

- **report(ReportDto reportDto)** : 차량 번호와 내용을 입력해 신고하기
- **updateNumberOfComplaints(Long memberId)**: 회원의 신고 당한 횟수 업데이트
- **getReportInfo(Long id)** : 회원 id를 통해 신고 당한 횟수와 신고 내용, 신고 당한 날짜를 가져옴

Controller

Class

MemberController

Method

@PostMapping("/members/complaints")
report(@RequestBody ReportDto reportDto): 신고하기
: memberService.report(reportDto)를 호출

@GetMapping("/members/{id}/complaints")
getReportInfo(@PathVariable Long id): 신고 내용 조회
: memberService.getReportInfo(id)를 호출

적절한 상태코드와 응답 메시지를 포함한 메시지를 생성, 해당 응답 반환

Repository

Class

MemberRepository, MemberRepositoryImpl,

ComplaintRepository, ComplaintRepositoryImpl

Method

- **vehicleRepository.findByVehicleNumber(String vehicle_number):** 차량 번호로 차량 검색
- **memberRepository.findById(Long id):** 회원 id로 회원 검색
- **complaintRepository.save(Complaint complaint):** 신고 내용 저장
- **complaintRepository.findComplaintsByMemberId(Long memberId):** 회원 id로 신고 내용 검색

구현 *아파트 공지사항*

DTO

Class

```
ApartmentNoticeDto ⌄ {
    apartment_notice_date   string($date)
                            example: 2023-05-17
    notice                  string
                            example: 공지사항 내용
}
```

Service

Class **ApartmentNoticeService**, **ApartmentNoticeServiceImpl**

Method **getApartmentNotice(String apartmentName)**
: 해당하는 이름의 아파트의 공지사항을 ApartmentNoticeDto 객체의 리스트로 반환

Repository

Class **ApartmentNoticeRepository**, **ApartmentNoticeRepositoryImpl**

Method **findByApartmentName(String apartmentName)**
: 해당하는 이름의 아파트 공지사항을 조회

Controller

Class **ApartmentController**

Method GetMapping("/apartment/{apartmentName}")
getApartmentNotice(@PathVariable String apartmentName)
: apartmentNoticeService.getApartmentNotice를 호출
조회된 결과를 기반으로 적절한 상태 코드와 메시지 생성, 전달

구현 *출차 시간 등록 / 수정*

DTO

Class **VehicleDto**

- exitTime : 출차 시간
- isLongTermParking : 장기 주차 여부

Repository

Class **VehicleRepository**, **VehicleRepositoryImpl**

Method **findById(Long id)**
: 주어진 id에 해당하는 차량 검색

Service

Class **VehicleService**, **VehicleServiceImpl**

Method **enrollDeparturetime(Long id, LocalTime exitTime, Boolean isLongTermParking)**
: 주어진 id에 해당하는 차량의 출차 관련 정보를 등록

modifyDeparturetime(Long id, LocalTime exitTime, Boolean isLongTermParking)
: 주어진 id에 해당하는 차량의 출차 관련 정보를 수정

Controller

Class **VehicleController**

Method @PostMapping("/vehicle/departuretime/{id}")
enrollDeparturetime(id, exitTime, isLongTermParking)
: vehicleService. enrollDeparturetime을 호출
적절한 상태코드와 응답 메시지를 포함한 메시지를 생성, 해당 응답 반환

@PutMapping("/vehicle/departuretime/{id}")
modifyDeparturetime(id, exitTime, isLongTermParking)
: vehicleService. modifyDeparturetime을 호출
적절한 상태코드와 응답 메시지를 포함한 메시지를 생성, 해당 응답 반환

구현 *출차 시간 및 주소 조회*

DTO

Class **MainPageDto**

- exitTime : 출차 시간
- remainingTime : 남은 시간
- isLongTermParking : 장기 주차 여부
- apartmentName : 아파트 이름
- address : 아파트 상세 주소

Service

Class **VehicleService**,
VehicleServiceImpl

Method
- **getDeparturetime(Long id)**

: 주어진 id에 해당하는 차량의 정보, 해당 차량의 출차 정보, 차량 소유주의 주소를 조회(MainpageDto 객체로 반환)

- **formatRemainingTime(long minutes)**

: DateTime 타입의 변수를 hh:mm(String) 형태로 변환

Repository

Class **VehicleRepository**,
VehicleRepositoryImpl

Method **findByIdWithMember(Long id)**
: 주어진 id에 해당하는 차량,
해당 차량과 관련된 회원 엔티티 검색

Controller

Class **VehicleController**

Method @GetMapping("/vehicle/departuretime/{id}")
getMainPageInfo(@PathVariable Long id)
: vehicleService.getDeparturetime을 호출
적절한 상태코드와 응답 메시지를 포함한 메시지를 생성, 해당 응답 반환

구현 *주변 차량 조회*

DTO

Class **NearVehicleDto**

- vehicle_number : 차량 번호
- model : 차량 모델명
- exitTime : 출차 시간
- color : 차량 색상
- isLongTermParking : 장기 주차 여부
- isSatisfied : 이중주차 조건 만족 여부
  (본인보다 출차 시간이 늦음 -> true
  장기 주차 상태 -> true
  본인보다 출차 시간이 빠름 -> false)

Repository

Class **NearVehicleRepository**, **NearVehicleRepositoryImpl**

Method

- **deviceRepository.findById(device_id)**
  : 주어진 device의 존재 유무 검색
- **nearDeviceRepository.findByNearDeviceId(device_id)**
  : 주변의 device 중에서 주어진 id를 갖는 device의 존재 유무 검색
- **vehicleRepository.findByDeviceId(device_id)**
  : 주어진 device를 가진 차량 검색
- **nearVehicleRepository.findByDeviceId(device_id)**
  : 주어진 device를 가진 주변 차량 검색

구현 *주변 차량 조회*

Service

Class **NearVehicleService**

Method **getNearVehicle(String device_id)**
: 주어진 device_id를 갖는 차량의 주변 차량들을 조회

1. 주어진 device_id에 해당하는 Device 존재 유무 검색
2. 해당 Device의 주변 Device(NearDevice) 리스트 조회
3. near_device_id를 device_id로 갖는 차량 조회
4. 해당 차량의 출차 시간 정보 조회
5. 해당 차량과 본인의 출차 시간 정보를 비교하여 isSatisfied 변수의 값 설정
6. 출차 시간을 String (**hh:mm**) 형식으로 변환
7. 주변 차량의 출자 정보를 전달하는 NearVehicleDto 리스트 반환

각 단계마다 조회 결과에 따라 NotFound 예외처리

Controller

Class **VehicleController**

Method @GetMapping("/nearvehicles/{id}")
getNearVehicles(@PathVariable("id") String deviceId)

- nearVehicleService.getNearVehicle(deviceId) 호출
- **/nearvehicles/{id}** 경로로 GET 요청이 들어왔을 때, 해당 디바이스의 근처 차량 정보를 조회, 응답하는 기능 제공
- 발생한 상황에 따라 BAD_REQUEST, NTERNAL_SERVER_ERROR 등의 예외 발생 시킴
  (잘못된 요청 형식, 매개변수 누락, RuntimeException 등)
- 최종적으로 응답 메시지와 상태 코드를 포함한 ResponseEntity 객체 반환

구현 코드 최적화

1. Stream API

```
public List<NearVehicleDto> getNearVehicle(String device_id) {

      // RepositoryImpl 호출 -> NotFound 예외 처리

        List<NearVehicle> nearVehicles = nearVehicleRepository.findByDeviceId(device_id)
                .orElseThrow(() -> new EntityNotFoundException("Near Vehicles not found"));

        LocalTime myDepartureTime = ...

      // nearVehicles 리스트를 스트림으로 변환
        return nearVehicles.stream()

      // nearVehicle 객체를 입력으로 받아 NearVehicleDto 객체 생성, 반환
                .map(nearVehicle -> {

      // NearVehilceDto 생성 로직

                })
      // collect -> 스트림의 최종 연산
      // toList()를 사용해 수집된 NearVehicleDto 객체들을 새로운 리스트로 반환
                .collect(Collectors.toList());
    }
```

2. JPA 최적화

- 일대다 관계에 대한 최적화

IN 키워드 사용, 한 번의 쿼리로 여러 객체 조회

```
String vehicleQuery = "SELECT v FROM Vehicle v WHERE v.device.device_id IN :near_device_ids";
```

- 엔티티 객체의 직접 생성

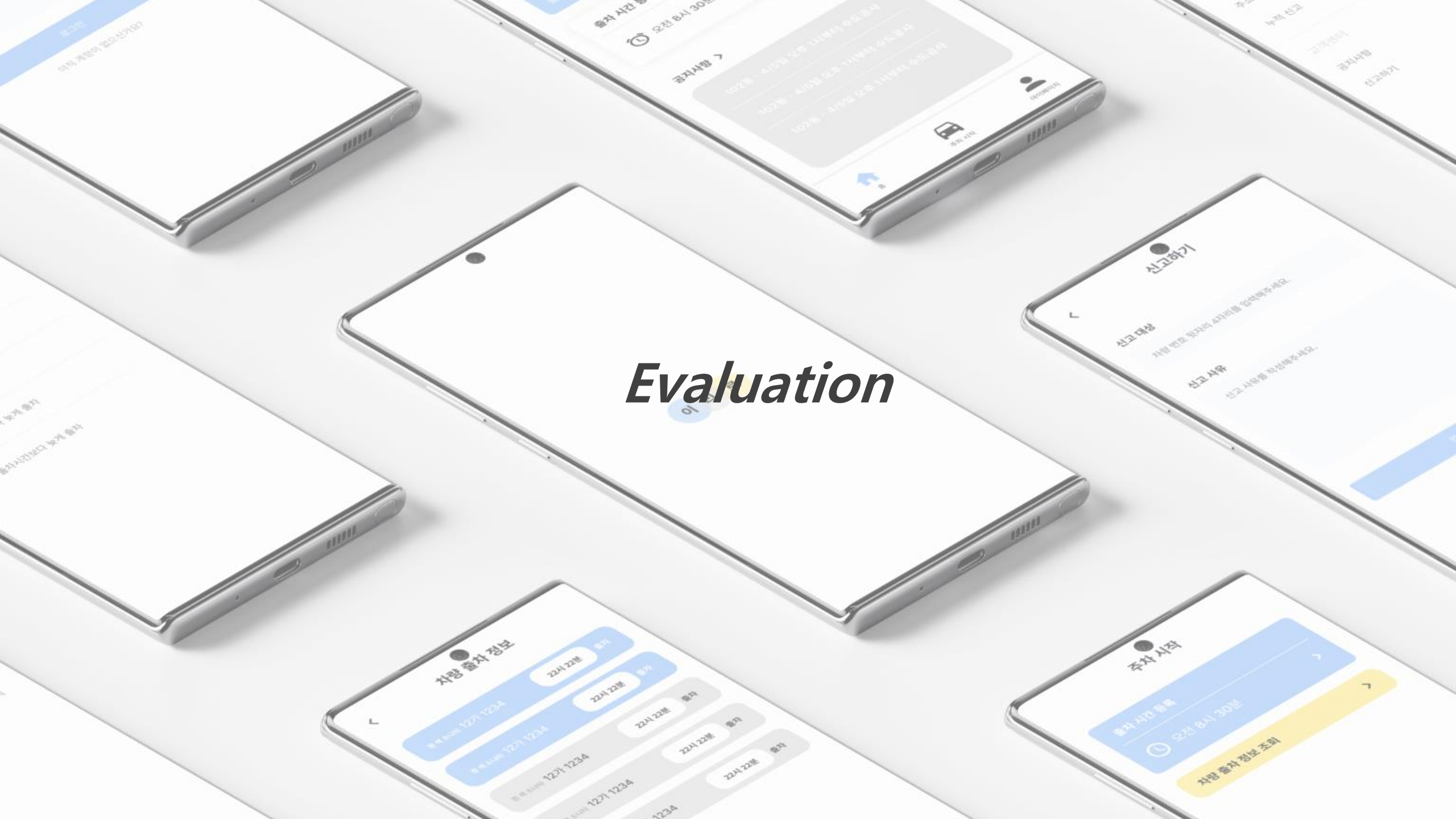
```
String vehicleQuery = "SELECT v FROM Vehicle v WHERE v.device.device_id IN :near_device_ids";

for (Vehicle vehicle : vehicles) {
            NearVehicle nearVehicle = new NearVehicle();
            nearVehicle.setNear_vehicle_number(vehicle.getVehicle_number());
            nearVehicle.setNear_vehicle_model(vehicle.getVehicle_model());
            nearVehicle.setNear_vehicle_color(vehicle.getVehicle_color());
```

- 예외 처리를 통한 로버스트한 구현

- 지연 로딩

```
  @ManyToOne(fetch = FetchType.LAZY)
      @JoinColumn(name="device_id") //FK
      private Device device;
```

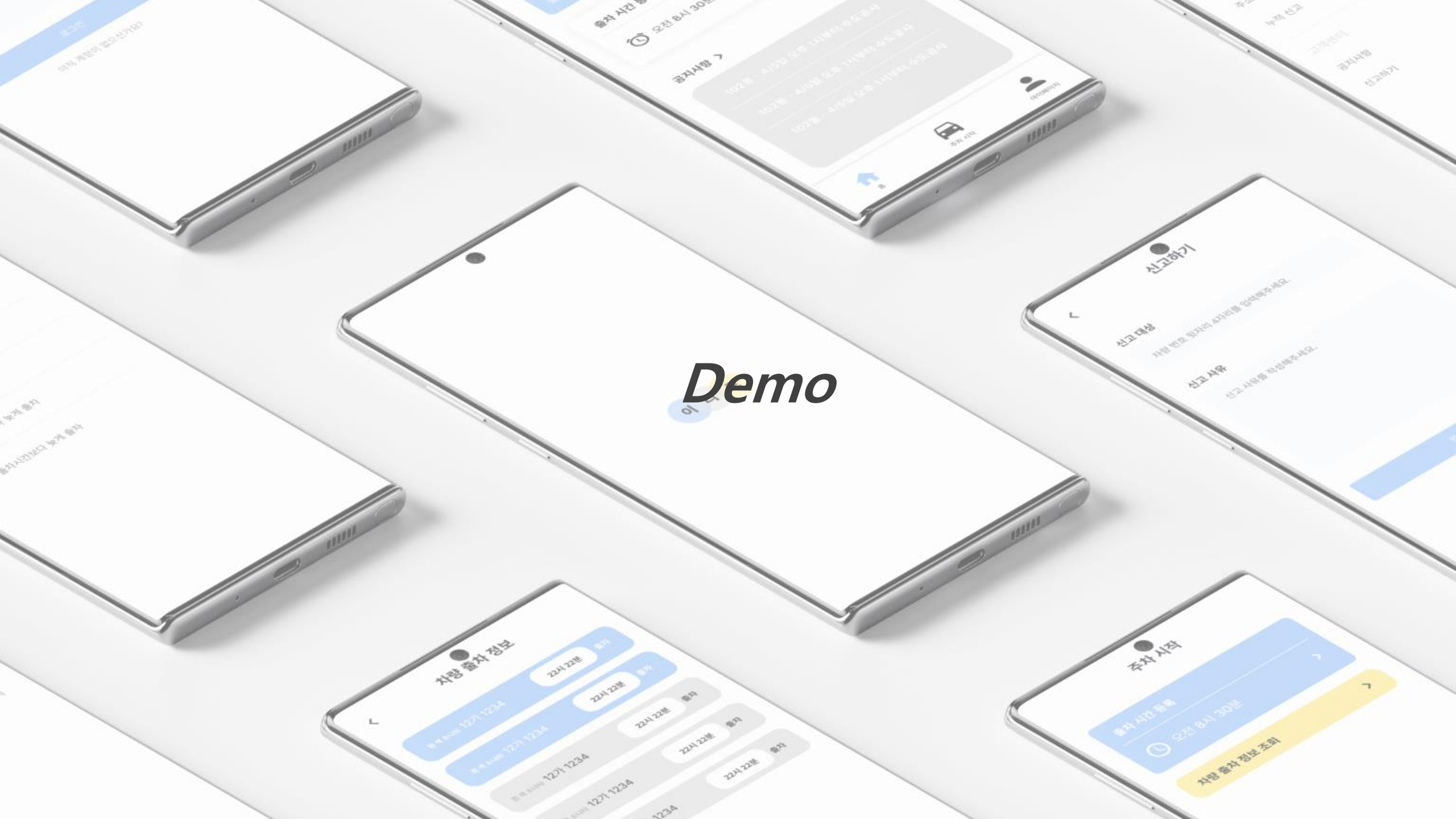

Evaluation
신고하기
신고 대상
신고 사유
차량 출차 정보
주차 시작
출차 시간 등록
오전 8시 30분
차량 출차 정보 조회

성능평가

캡스톤 디자인 6팀

분류	기능	성능평가항목	06/08	06/09	06/10	06/11	06/12	정확도
디바이스		서로 간 디바이스 인식이 가능한가?	12/12	12/12	12/12	12/12	12/12	60/60
		정확한 디바이스 id가 들어오는가?	12/12	12/12	12/12	12/12	12/12	60/60
		인식 가능한 거리는 충분한가?	11/12	11/12	10/12	11/12	12/12	55/60
		인식 거리 내 모든 디바이스 인식 가능한가?	12/12	12/12	12/12	12/12	12/12	60/60
		인식 거리 외 디바이스의 아이디는 제외가 잘 되는가?	12/12	12/12	12/12	12/12	12/12	60/60
		수신되는 기기 LED는 잘 작동하는가?	12/12	12/12	11/12	12/12	12/12	59/60

분류	기능	성능평가항목	05/25	05/30	06/01	06/06	06/08	구현여부
앱 서비스	회원가입	회원가입 시 필수 항목 작성에 대한 처리되어있는가?	O	O	O	O	O	O
	로그인	정확한 정보를 통한 로그인이 이루어지는가?	O	O	O	O	O	O
		입력한 해당 아파트로 정확히 분류하는가?	X	O	O	O	O	O
	메인페이지	출차까지 남은 시간 정보 정확한가?	O	O	X	O	O	O
		아파트 공지사항 정보 잘 보여주는가?	O	O	O	O	O	O
		회원의 주소 정보를 옳바르게 출력하는가?	O	O	O	O	O	O
		출차 정보를 등록하지 않은 경우를 고려하였는가?	O	O	O	O	O	O
	주차 시작	주변 차량에서 인식된 정보 잘 보여주는가?	O	O	O	O	O	O
		출차 시간 조건에 맞는 차량을 잘 구분하여 보여주는가?	O	O	O	O	O	O
		등록한 출차 시간에 맞게 해당 정보 변화 잘 이루어지는가?	O	O	O	O	O	O
		장기 주차 설정한 차량에 대한 처리되어있는가?	O	O	O	O	O	O
		출차 시간 등록하지 않은 차량에 대한 처리되어있는가?	O	O	O	O	O	O
		새로고침 시 인식 거리 내 새로운 차량 정보를 정확히 가져오는가?	O	O	O	O	O	O
	마이페이지	입력한 회원정보 정확히 보여주는가?	O	O	O	O	O	O
		신고 차량번호로 정확히 데이터가 입력되는가?	O	O	O	O	O	O
	통신	예외상황에 대한 처리되어있는가?	X	O	O	O	O	O
		통신 오류에 대한 처리가 되어있는가?	O	O	O	O	O	O
	신고하기	신고 내역 정보가 해당 회원에게 잘 들어오는가?	O	O	O	O	O	O
		신고 당한 횟수는 정확하게 카운트 되는가?	X	O	O	O	O	O


Demo
신고하기
신고 대상
신고 사유
차량 출차 정보
주차 시작
출차 시간 등록
오전 8시 30분
차량 출차 정보 조회

한계 및 개선사항

캡스톤 디자인 6팀

한계점

- 디바이스 간 송수신 감도가 일정하지 않음
- 송수신 감도 설정 불가
- 노이즈로 인해 통신이 일정하지 않아 다수의 송신으로 인한 통신은 부정확할 가능성이 있음
- 단일 코어 컨트롤러인 nodeMCU의 역량 부족으로 송수신기를 동시에 구현하여 디바이스의 완전체 만들기 어려움

개선사항

- 정밀한 태그 및 안테나 사용
- 다중 코어 컨트롤러 디바이스 활용