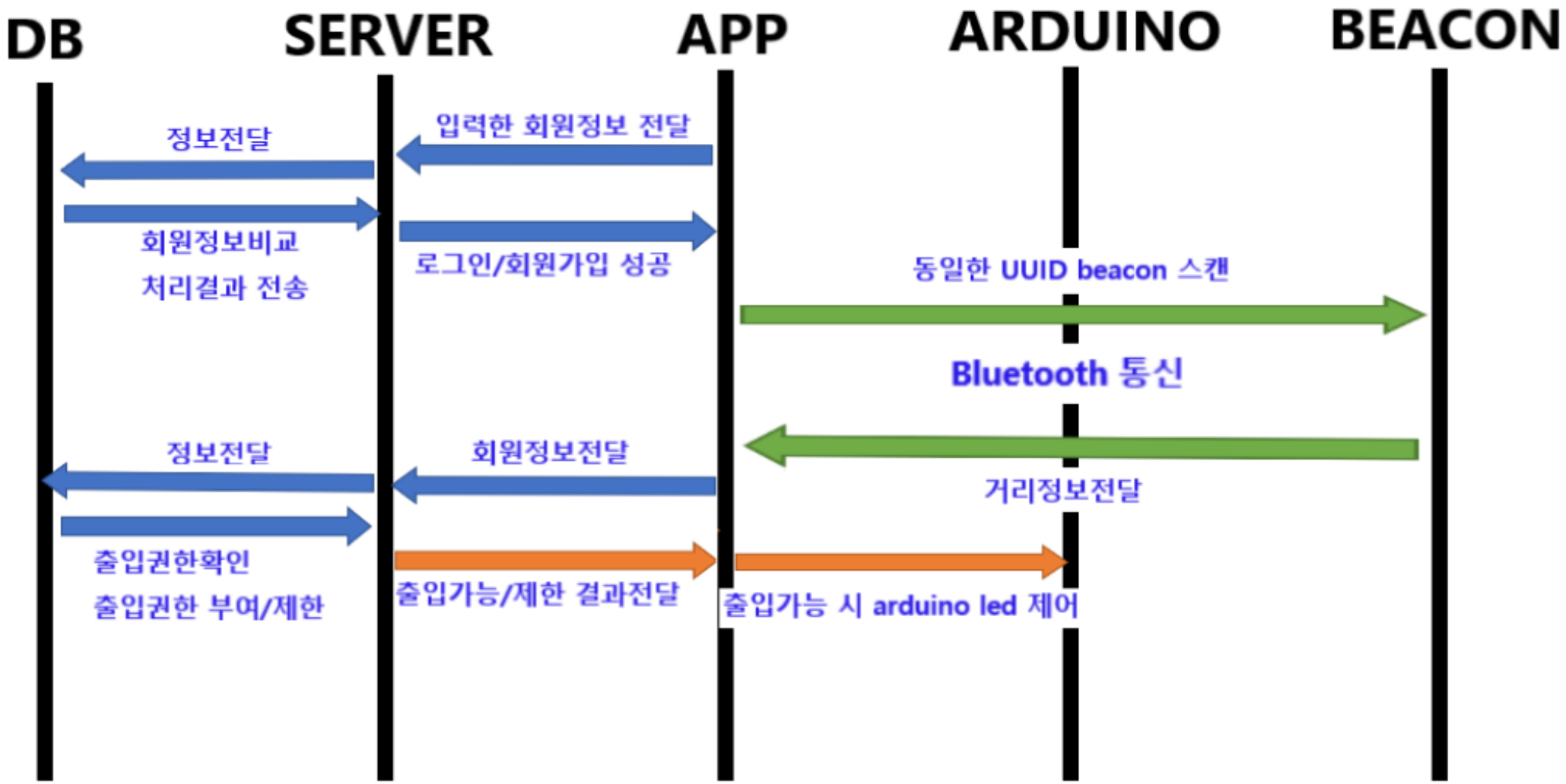
HUFSmartkey REST API 소개

발표자 : 201802719 이유진

1. 제품흐름도


DB
SERVER
APP
ARDUINO
BEACON
정보전달
입력한 회원정보 전달
회원정보비교
처리결과 전송
로그인/회원가입 성공
동일한 UUID beacon 스캔
Bluetooth 통신
정보전달
회원정보전달
거리정보전달
출입권한확인
출입권한 부여/제한
출입가능/제한 결과전달
출입가능 시 arduino led 제어

1. 코드설명

1. SignUp

```
@csrf_exempt
def client_list(request, format=None): # app --(identification, password, phone_number, name)--> server
    if request.method == 'GET': # 전체조회
        query_set = ClientData.objects.all()
        serializer = ClientDataSerializer(query_set, many=True)
        return JsonResponse(serializer.data, safe=False)

    elif request.method == 'POST': # 회원가입_test완료
        identification = request.POST.get("identification", "")
        password = request.POST.get("password", "")
        phone_number = request.POST.get("phone_number", "")
        name = request.POST.get("name", "")

        print('identification = ' + identification + 'password = ' + password + 'phone_number = ' + phone_number + 'name= ' + name) # 서버쪽 터미널에 띄움
        myuser = ClientData.objects.filter(identification=identification)

        if myuser: # db에 저장되어있으면 -> id중복
            print("duplicated id, signUp failed") # for server debugging
            return JsonResponse({'code':'400', 'msg':'duplicated id'}, status=400)
        else: # new client면 -> db저장
            form = ClientData(identification=identification, password=password, phone_number=phone_number, name=name)
            form.save()
            print("signUp success") # for server debugging
            return JsonResponse({'code':'201', 'msg':'signup success'}, status=201) # app으로 보내는 msg
```

client로 부터 data를 받아
맞는 변수에 저장한다.

db에 저장되어있는지 확인

1. 있다면 -> 중복이므로 회원가입 실패
2. 없다면 -> 새로운 회원이므로 회원가입 성공

2. LOGIN

```
@method_decorator(csrf_exempt, name='dispatch')
def login(request, format=None): # app --(identification, password)--> server --(allowed_area)--> app
    if request.method == "GET":
        return render(request, 'client_data/login.html')

    elif request.method == 'POST':
        identification = request.POST.get("identification", "")
        password = request.POST.get("password", "")
        myuser = ClientData.objects.filter(identification=identification, password=password)

        print("identification = " + identification + " password" + password)

        if myuser:
            print("login success")

        obj = ClientData.objects.get(identification=identification)
        phone_number = obj.phone_number
        name = obj.name

        print("identification = " + identification + " password" + password)
        print("phone:" + phone_number + "name:" + name)
```

로그인 시 client가 입력한 정보들을 POST로 받아와 변수에 저장합니다..

이때, 사업자인지 guest인지 확인하기 위하여 소상공인 db에 로그인한 client가 있는지 확인합니다.

3. door open - 사업자

```python
@csrf_exempt
def door_open(request, format=None): # app --(id, uuid)--> server
    if request.method == "GET":
        return render(request, 'client_data/login.html')
    if request.method == 'POST':
        id = request.POST.get("id", "")
        uuid = request.POST.get("uuid", "")

        print("<door_open> id = " + id + " uuid" + uuid)
        if(int(id) == 0): # 사업자 -> 바로 문 열어준다.
            # from .ctr_servo import run_servo
            # run_servo(1) # run servo Motor
            arduino.write([1])
            data = arduino.read()
            print(data)

            return JsonResponse({'code':'201', 'msg':'true'}, status=201)
```

client가 출입을 요청할 때, 만약 이 client가 사업자라면(id==0) 바로 문을 열어주고, led를 켜줍니다.(문 열림 체크)

3. door open - guest

만약 guest라면, 출입요청시 서비스타임이 만료되었는지 확인해야합니다. db에 저장되어있는 guest qr check시간과 현재시각을 비교하여 만약 2시간이 지났다면 db에서 삭제하고, 출입권한부여를 하지 않습니다.

```
elif(int(id) > 0): # guest일 때 -> 현재시각과 service start 한 시간 비교
    now = round(time.time())
    print("now time is:" + str(now))
    con = sqlite3.connect("db.sqlite3")
    cursor = con.cursor()
    start_time = cursor.execute("SELECT start_time FROM small_business_businessdata WHERE id='%d'" %(int(id))).fetchall()[0][0]
    if(now - int(start_time) >= 7200): # service time 이 두시간 이상일 때
        db = cursor.execute("DELETE FROM small_business_businessdata WHERE id='%d'" %(int(id)))
        print("service time done")
        return JsonResponse({'code':'201', 'msg':'false'}, status=201) # service time done
    else:
        # from .ctr_servo import run_servo
        # run_servo(1) # run servo Motor
        arduino.write([1])
        data = arduino.read()
        print(data)
        return JsonResponse({'code':'201', 'msg':'true'}, status=201) # door open
else :
    return JsonResponse({'code':'400', 'msg':'door not open'}, status=400)
```

두시간이 지나지 않았다면 led를 켜줍니다(문열림 체크)

4. guest qrcode

```python
@csrf_exempt
def first_qr_scan(request, format=None): # app --(store, allowed_data)--> server --(id)--> app
    if request.method == 'POST':
        store = request.POST.get("store", "")
        allowed_area = request.POST.get("allowed_area", "")

        print("from app) store: " + store + ", allowed_area: " + allowed_area)

        start_time = str(round(time.time()))
        print("start time is : " + start_time)

        con = sqlite3.connect("db.sqlite3")
        cursor = con.cursor()
        db = cursor.execute("INSERT INTO small_business_businessdata (store, allowed_area, start_time) VALUES ('%s', '%s', '%s')" %(store, allowed_area, start_time))
        id = cursor.execute("SELECT id FROM small_business_businessdata WHERE start_time='%s'" %(start_time)).fetchall()[0][0]
        con.commit()
        con.close()
        print("db insert result id:" + str(id))

        if (id >= 0):
            return JsonResponse({'code':'201', 'msg': str(id)}, status=201)
        else:
            return JsonResponse({'code':'400', 'msg':'store into db as guest failed'}, status=400)
```

client가 qr code를 찍으면 POST로 server에게 가게이름과 가게에서 미리 설정해둔 guest 출입가능 구역을 보내줍니다.

이제 guest도 출입가능한 구역이 생겼기 때문에 소상공인 db에 guest로 추가를 해줍니다.

guest는 시간제한이 있으므로 qr을 시간을 같이 db에 저장해줍니다.

HUFSmartkey APP, Retrofit2 소개

발표자 : 201602560 이재성

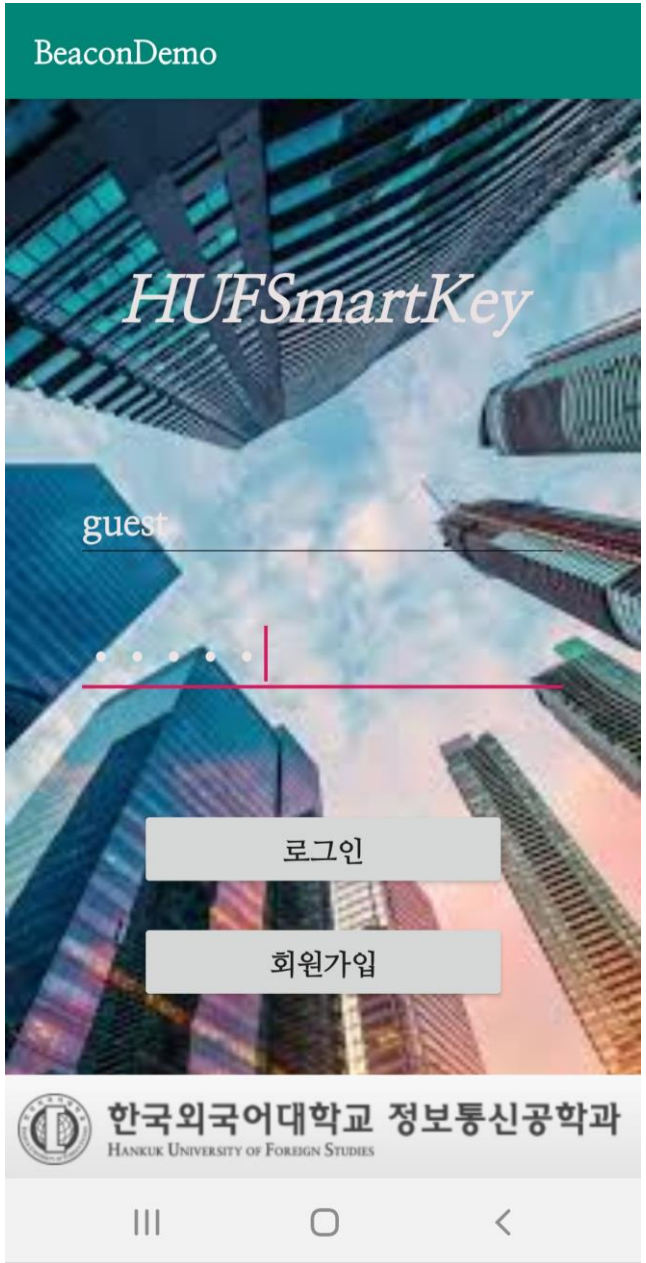
HUFSmartKey APP 소개

1. GUEST

1. SIGN UP

정상적인 Sign up이 완료되면, Toast 메시지로 회원가입 완료 문구와 아이디를 띄워 줌.

2. LOGIN



- ✓ 입력한 계정이 DB에 존재하면 로그인 성공.
- ✓ 입력한 계정이 DB에 존재하지 않으면 로그인 실패 -> Toast 메시지를 띄워 줌.

3. LOCK

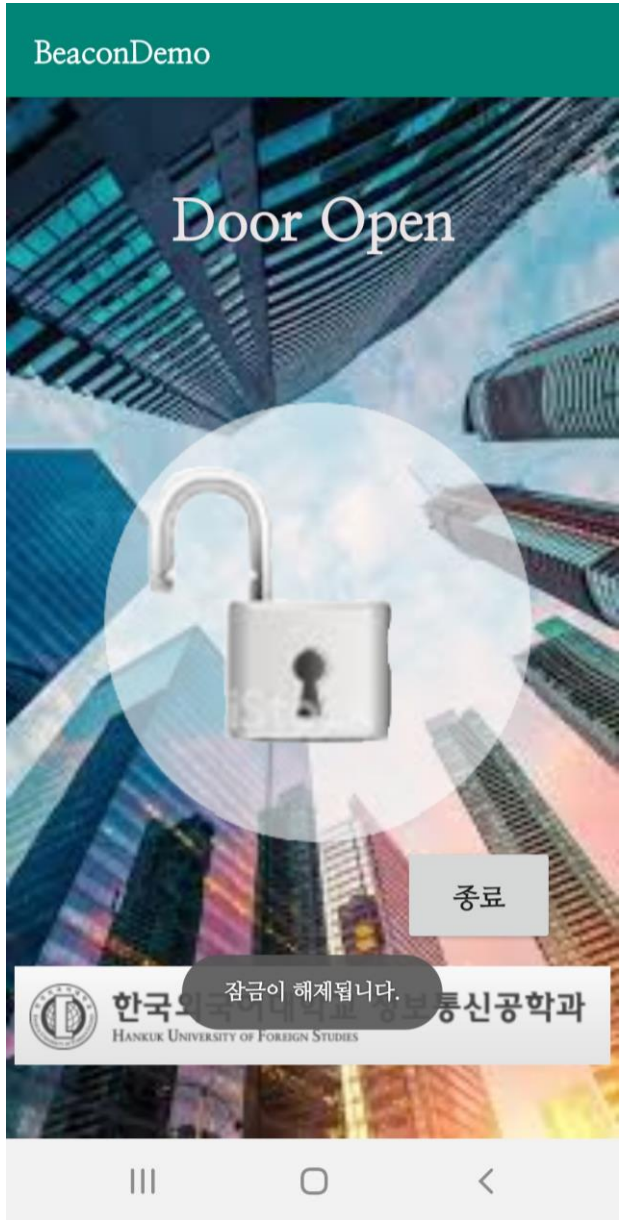
✓ Guest는 현재 접근할 수 있는 구역이 없음.

✓ 가게 내 QR CODE인증 필요.

4. 출입 권한

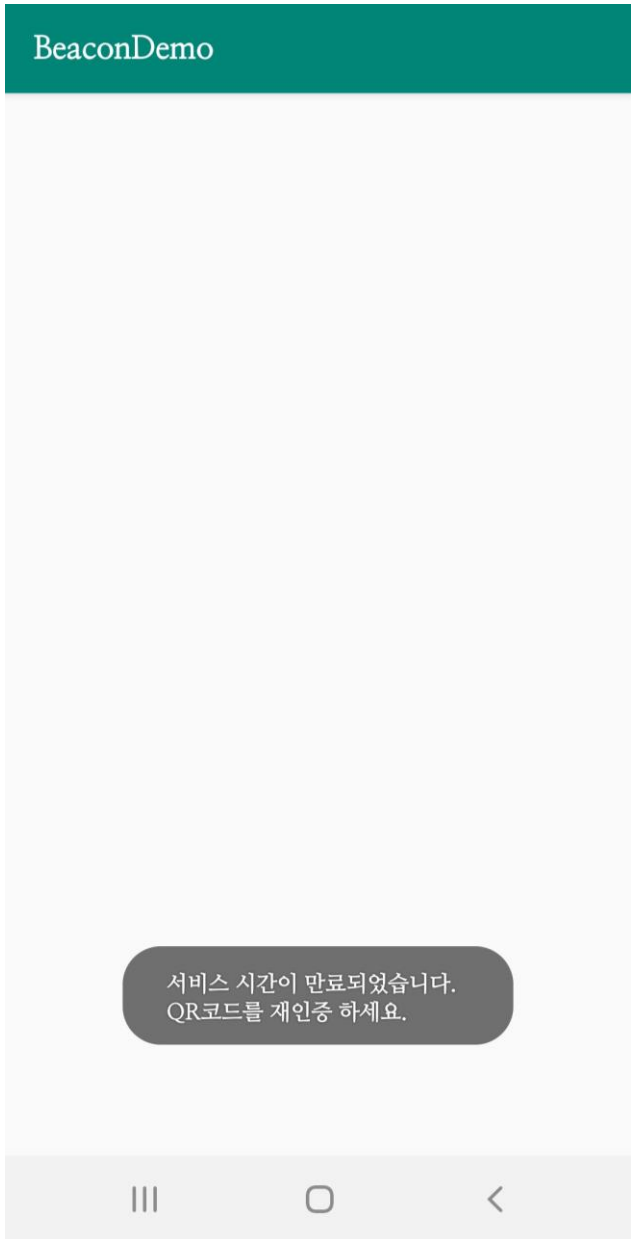
- ✓ QR Code 인증 후 출입 가능한 리스트 표시
- ✓ 출입하고자 하는 구역을 입력 -> 자물쇠 버튼 클릭하여 출입 권한 요청

5. UNLOCK



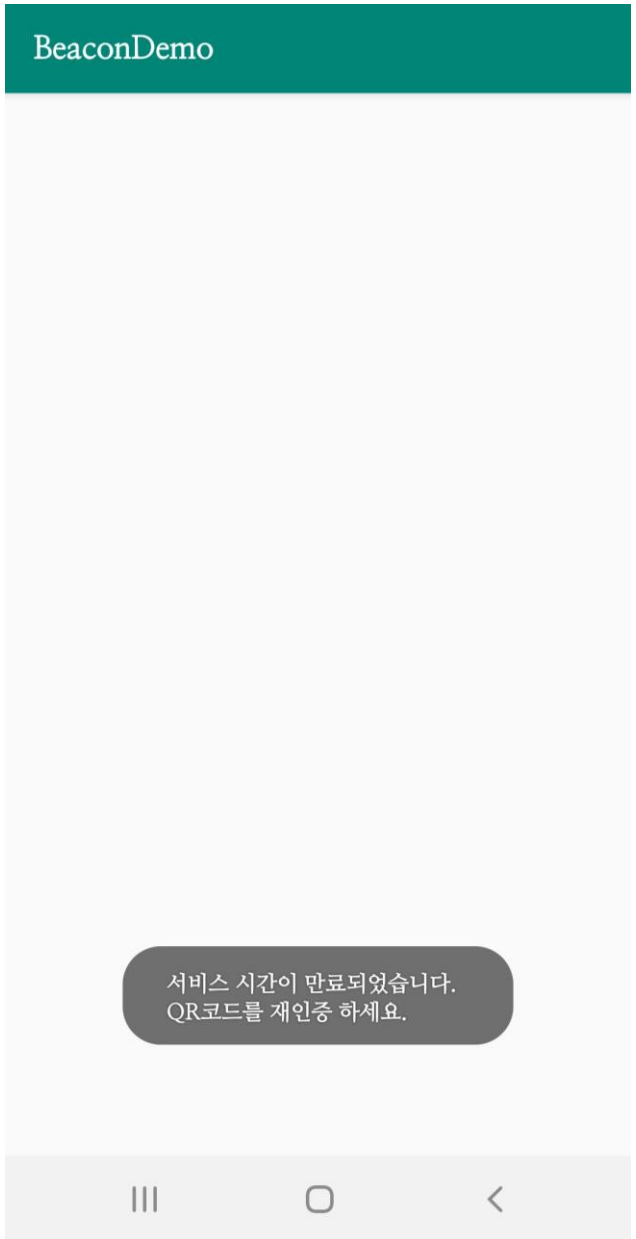
✓ 비콘과 사용자의 거리가 10cm 이내이고, QR Code 인증 후 2시간 이내이면 unlock

5. 서비스 만료



✓ QR Code 인증 후 2시간이 지나면 서비스 시간 만료. 문이 열리지 않음.

5. 서비스 만료



✓ QR Code 인증 후 2시간이 지나면 서비스 시간 만료. 문이 열리지 않음.

2. OWNER

1. LOGIN -> LOCK

- ✓ 사업자인 경우, 출입가능구역 리스트가 바로 표시됨.
- ✓ 출입하고자 하는 구역을 입력 -> 자물쇠 버튼 클릭하여 출입 권한 요청

2. UNLOCK

✓ unlock.

3. Restricted Area

화장실 앞에서 주차장 출입을 요구할 경우

✓ 서비스가 불가함.
✓ 문이 열리지 않음.

HUFSmartKey
Retrofit2 코드 소개

Login & Sign up

```
package com.zartre.app.beacondemo

data class forLogin(
    val code: String,
    val msg: String,
    val allowed_area: String
)
```

서버로부터 받아오는 Message에 담긴 정보입니다.

```
interface forSignupService {
    @FormUrlEncoded
//    @Headers("accept: application/json", "content-type: application/json")
    @POST("/client_data/")
    fun requestSignup(
        @Field("identification") s_id:String,
        @Field("password") s_pw:String,
        @Field("name") s_name:String,
        @Field("phone_number") s_phoneNum:String
    ) : Call<forSignup>

}
```

서버로 보내주는 정보입니다.

Login & Sign up

```
// retrofit 사용
var retrofit = Retrofit.Builder()
    .baseUrl("http://220.67.124.145:8080")
    .addConverterFactory(GsonConverterFactory.create())
    .build()

var loginservice: forLoginService = retrofit.create(forLoginService::class.java) // retrofit 객체 생성

login_btn.setOnClickListener{
    var s_id = edit_ID.text.toString()
    var s_pw = edit_PW.text.toString()

    loginservice.requestLogin(s_id, s_pw).enqueue(object: Callback<forLogin> {
        override fun onFailure(call: retrofit2.Call<forLogin>, t: Throwable) {  // 실패할 때
            Log.e("로그인", t.message)
            var dialog = AlertDialog.Builder(this@login)
            dialog.setTitle("ERROR")
            dialog.setMessage("서버와의 통신이 실패하였습니다.")
            dialog.show()
        }

        override fun onResponse(call: retrofit2.Call<forLogin>, response: Response<forLogin>) {  // 서버에서 정상 응답이 올 때
            if (response?.isSuccessful) {
                var forLogin = response.body()
                Log.d("LOGIN", "msg : " + forLogin?.msg)
                Log.d("LOGIN", "code : " + forLogin?.code)
                Log.d("LOGIN", "allowed_area : " + forLogin?.allowed_area)
                Toast.makeText(this@login, "로그인에 성공하였습니다. \n 즐거운 하루 되세요.", Toast.LENGTH_SHORT).show()

                var intent = Intent(applicationContext, lockActivity::class.java).apply {
                    putExtra(ALLOWED, forLogin?.allowed_area)
                }
                startActivity(intent)
            }
            else {
                Toast.makeText(this@login, "로그인에 실패했습니다. \n 아이디 혹은 비밀번호를 확인하세요.", Toast.LENGTH_LONG).show()
            }
        }
    })
```

retrofit 객체를 생성합니다.
로그인 버튼을 누르면 실질적인 통신이 시작됩니다.