졸업작품 최종보고서

'저 여기서 내려요' 앱

지도교수

유준범 교수님

컴퓨터공학과 201212186 황보규

컴퓨터공학과 201411271 박상우

컴퓨터공학과 201611271 심준석

목차

Graduation Project

1. 선정 배경 및 목적

1) 선정 배경

i) 일상 생활에 가장 밀접한 관련이 있는 지하철 이용에 불편한 점을 선정.

ii) 내가 다른 사람보다 먼저 승차했음에도 운에 따라 좌석에 앉지 못하는 경우.

iii) 앞에 앉아 있는 승객이 언제 내릴지 알 수 있으면 좋겠다는 아이디어 창출.

2) 목적

i) 승객들의 좌석 편의.

ii) 어느 좌석 앞에서 대기하고 있으면 빨리 앉을 수 있을지 쉽게 알 수 있도록 함.

iii) 좌석 새치기 방지.

2. 개발 내용 및 목표

1) 개발 내용

i) 안드로이드 기반의 어플리케이션.

ii) 서버 및 클라이언트를 구현하여 서버는 각 클라이언트들 간의 접속에 필요한 정보를 제공할 수 있게 구현함.

iii) 실시간 지하철 도착 API를 JSON 정보로 받아 위치를 파악.

iv) 하차역을 서로 볼 수 있는 네트워크 통신 구현.

v) 사용자가 쉽게 지하철 역 검색 / 좌석 등록을 할 수 있는 UI 구현.

vi) 좌석 등록 동기를 부여할 포인트 제도 구현.

2) 개발 목표

i) 여러 사용자가 사용해도 누락 없이 하차역 표시.

ii) 여러 사용자가 좌석을 선택할 때 중복되지 않도록 함.

Iii) 실시간 좌석 조회 구현.

iv) 포인트 부정 획득을 방지하기 위한 대책 구현.

3. 개발 환경

- OS : Microsoft Windows
- IDE – Eclips, Android Studio
- Language – Java
- 서버 데이터베이스 : MariaDB

4. 요구사항 분석

1) 기능적 요구사항

회원가입	1.1 ID 입력 1.2 ID 중복확인 1.3 비밀번호 입력, 비밀번호 확인 1.4 집 / 회사와 가까운 지하철역들을 즐겨찾기로 설정 1.5 회원가입 완료
로그인	2.1 ID, PW 입력 2.2 유효한 ID, PW인지 확인 2.3 로그인 완료
로그아웃	3.1 로그아웃 버튼 선택 3.2 로그아웃 완료
지하철 차량 선택	4.1 지하철 역 검색 4.2 지하철 역 선택 4.3 사용자 본인이 탈 예정 / 타고 있는 지하철 차량 선택
하차 역 등록	5.1 좌석에 앉아 있는 사용자가 현재 자신의 좌석을 확인 (지하철 내부의 칸 번호를 통해 자신의 위치 확인) 5.2 사용자가 내릴 지하철 역 선택 5.3 지하철역 선택 및 확인 창 표시 5.4 5.1에서 선택한 좌석에 5.3에서 선택한 지하철역을 하차역으로 등록 5.5 일정 포인트 획득 및 등록 결과 창 표시

하차 역 조회	6.1 하차 역 조회 버튼 선택 6.2 보유 포인트가 충분한지 확인 6.3 하차 역 조회 선택을 재확인하는 창 표시 6.4 기존에 색이 입혀지지 않았던 지하철 열차 객실과 객실 내부의 좌석들 색 변화 6.4.1 곧 하차하는 승객들이 많은 객실 순서대로 다른 색 표현 (하차하는 승객들이란 하차역을 앱에 등록한 사용자들 지칭. 앱을 사용하지 않지만 탑승하고 있는 승객들을 지칭하지 않는다.) 6.4.2 하차역이 등록된 좌석들에 한하여 하차역까지 남은 거리에 따라서 다른 색으로 표현. 6.5 보유 포인트에서 일정 차감되면서 조회 완료
하차 역 변경	7.1 하차 역 변경 버튼 선택 7.2 사용자가 내릴 지하철역 검색(5.2과 동일) 7.3 지하철역 선택 및 확인 창 팝업(5.3과 동일) 7.4 변경 사항 저장
하차 역 재 조회	8.1 하차 역 재 조회 버튼 선택 8.2 하차 역 조회 시간 잔여 여부 확인 8.3 재 조회 완료

2) 비기능적 요구사항

i) Privacy

- 사용자가 등록된 하차역을 명시적으로 표기하지 않고, 남은 역 수에 따라서 다른 색으로 표시.

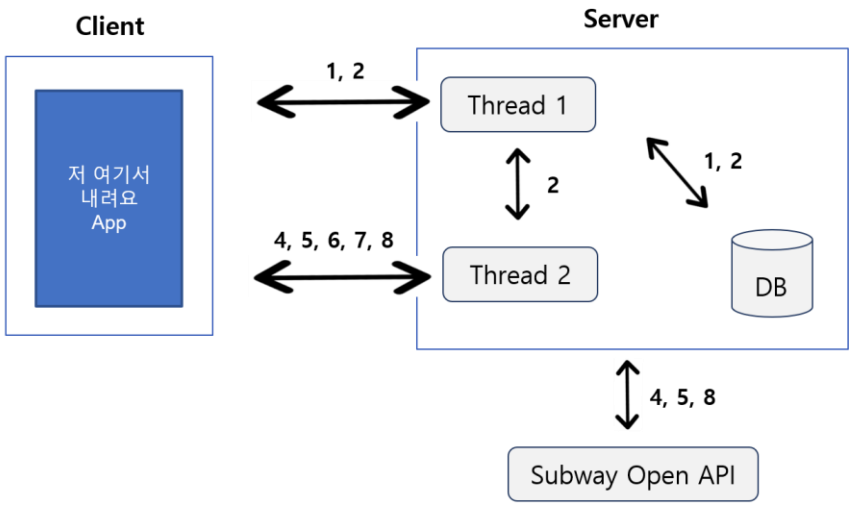
ii) Ethical

- 포인트 부정 획득을 방지하기 위해 실제로 지하철을 타고 있을 때만 포인트 획득이 가능하도록 설정.

5. 디자인

1) Architecture Diagram

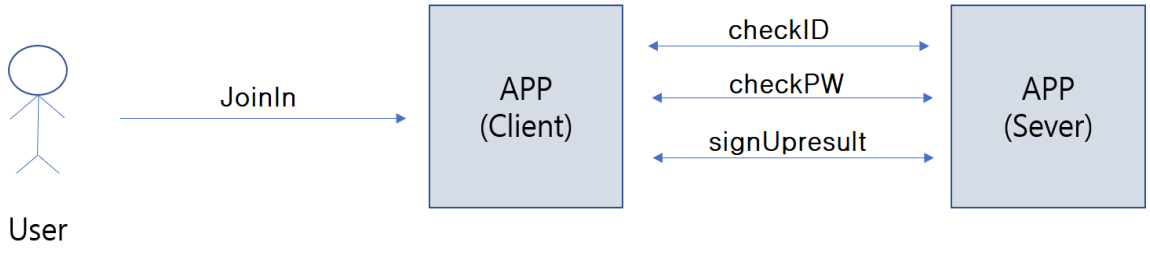
아키텍처 다이어그램



- 크게 클라이언트와 서버 두 가지로 구성되어 있음.
- 회원가입, 로그인 담당의 Thread 1과 회원 정보를 저장하는 DB.
- 지하철 차량선택, 등록된 좌석 조회, 좌석 등록, 등록된 좌석 재 조회를 담당하는 Thread2.
- 실시간 지하철 도착 정보를 제공하는 Subway Open API

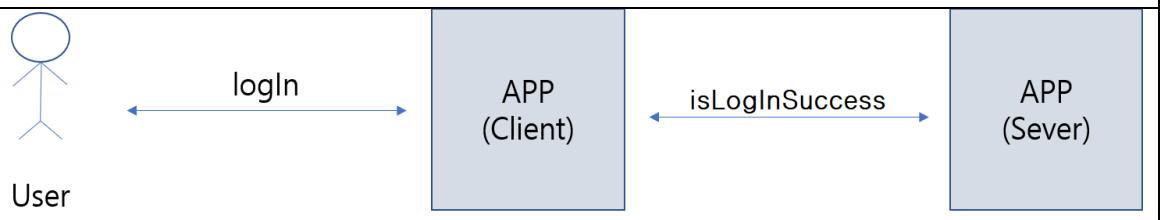
2) Interface Diagram

1. 회원가입



- 유저가 회원가입 버튼을 누름.
- ID 중복 확인, PW 일치 확인을 한 후 계정 생성.

2. 로그인

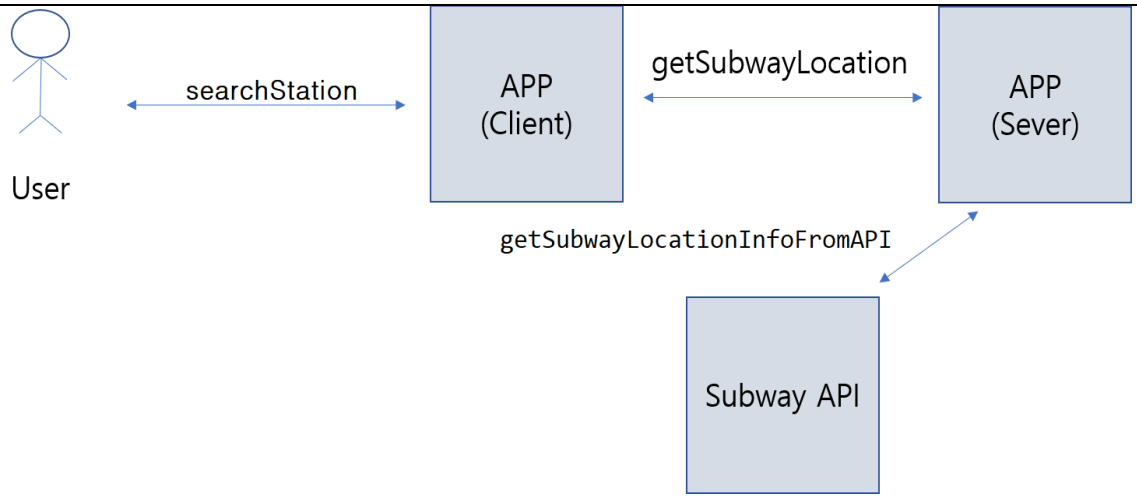


- 유저가 로그인을 버튼을 누르고, 서버에서 로그인 정보가 맞는지 체크 후 로그인.

3. 로그아웃

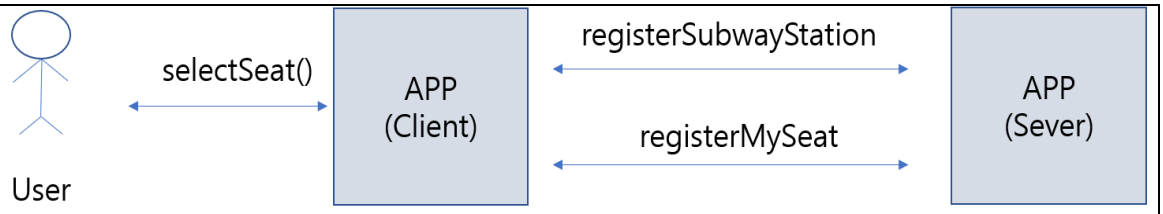
- 서버를 거칠 필요 없이 클라이언트 내에서 로그아웃이 진행.

4. 지하철 차량 선택



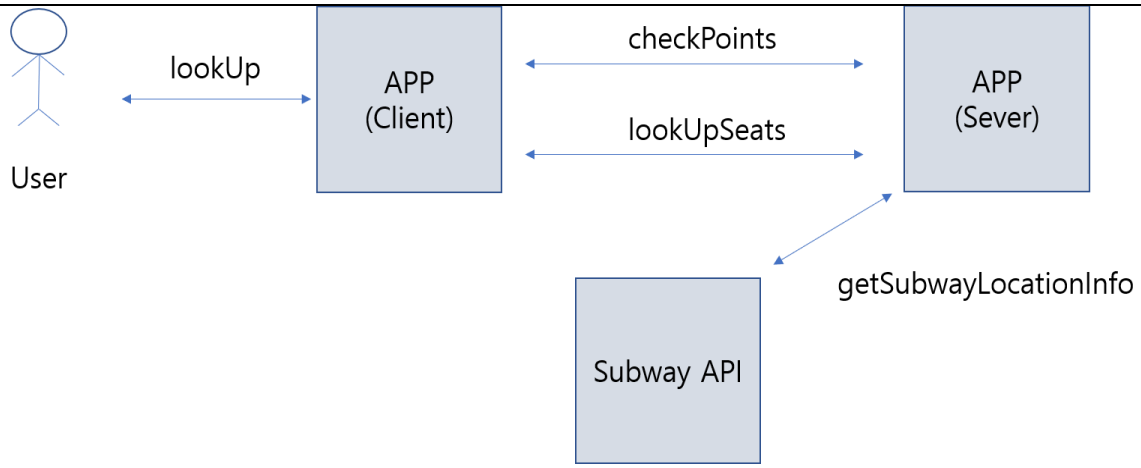
- 유저가 지하철 역을 검색하면 Subway API에서 지하철이 어디 있는지 정보를 받아와 해당 지하철을 선택.

5. 하차 역 선택



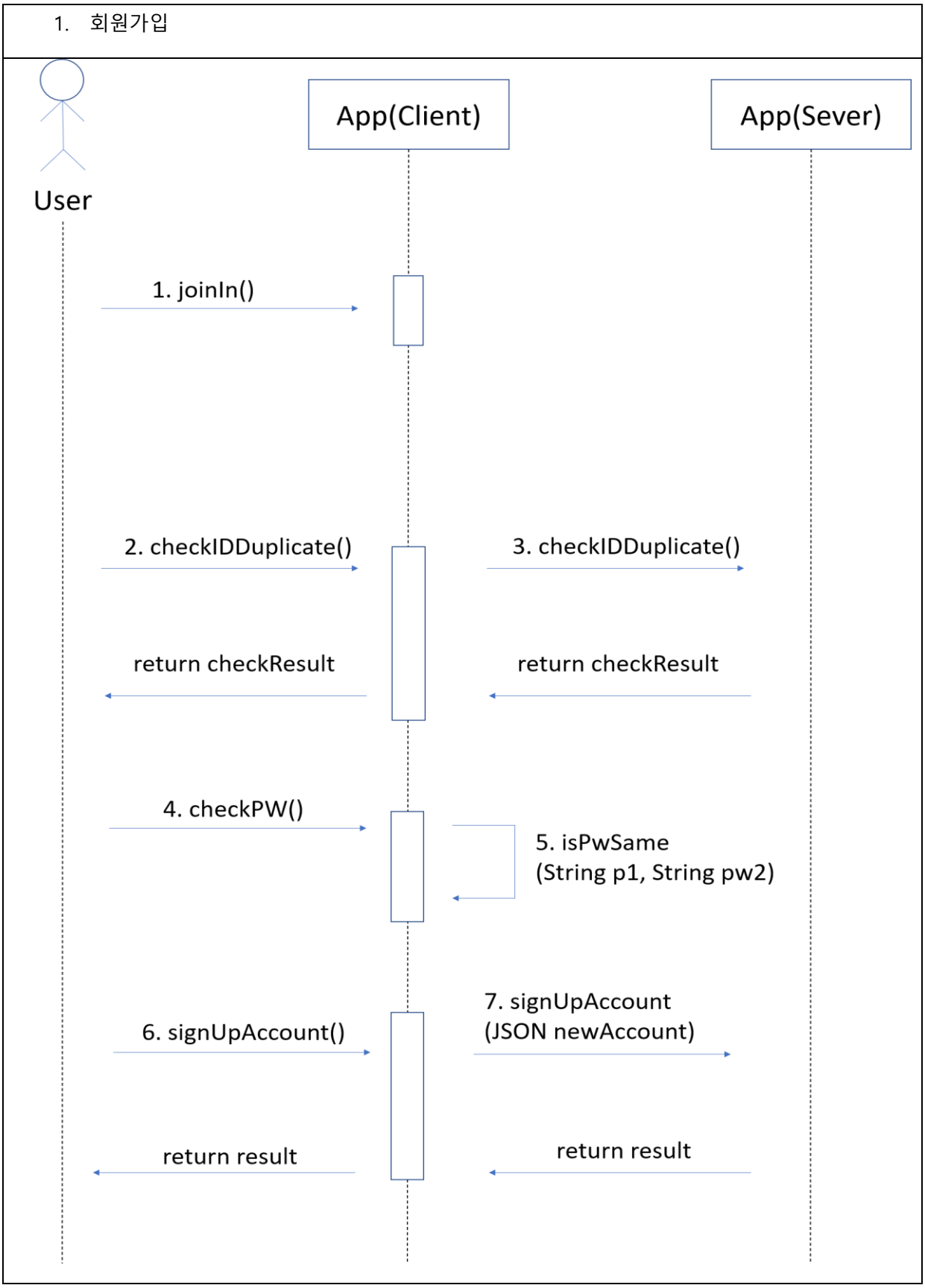
- 유저가 좌석을 선택하면 서버에 좌석이 등록된다.

6. 하차 역 조회

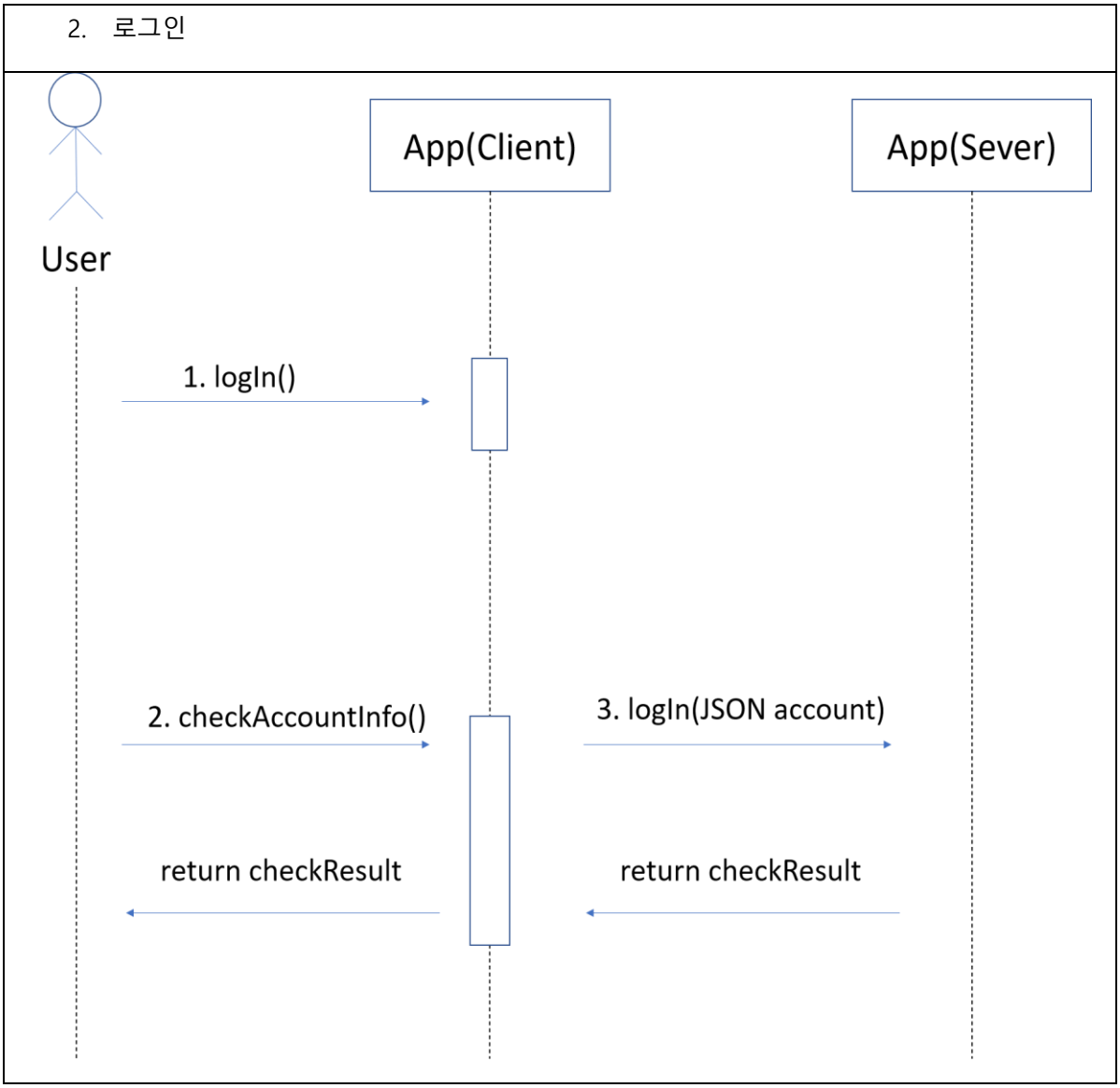


- 유저가 검색 버튼을 누르면 서버에서 해당 계정의 잔여 포인트를 계산, 포인트가 충분하면 등록된 좌석들을 조회할 수 있다.
- 해당 열차의 좌석을 조회하려면 실시간 지하철 위치 정보가 필요하여 Subway API에서 정보를 받아온다.

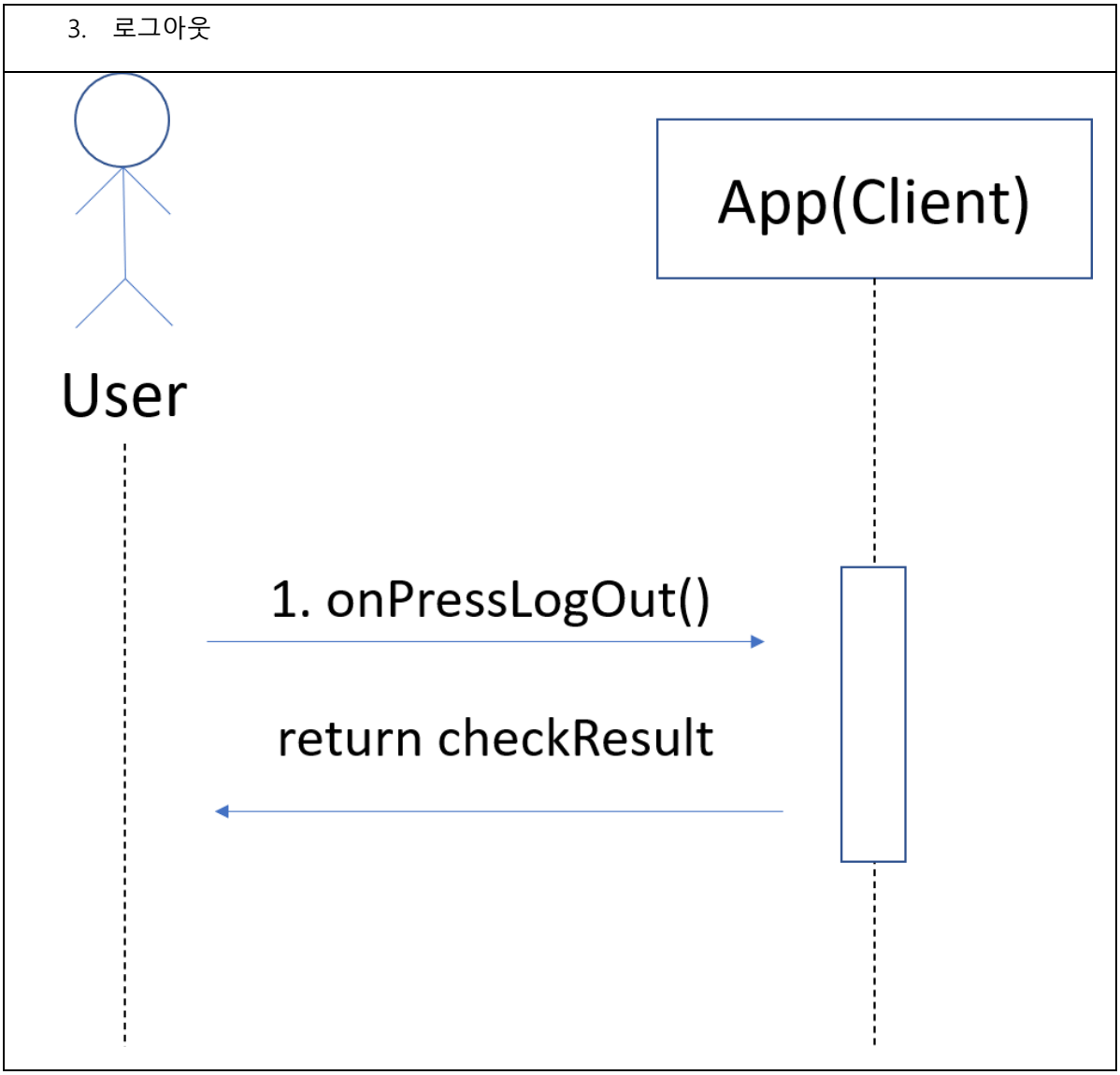
3) System Sequence Diagram



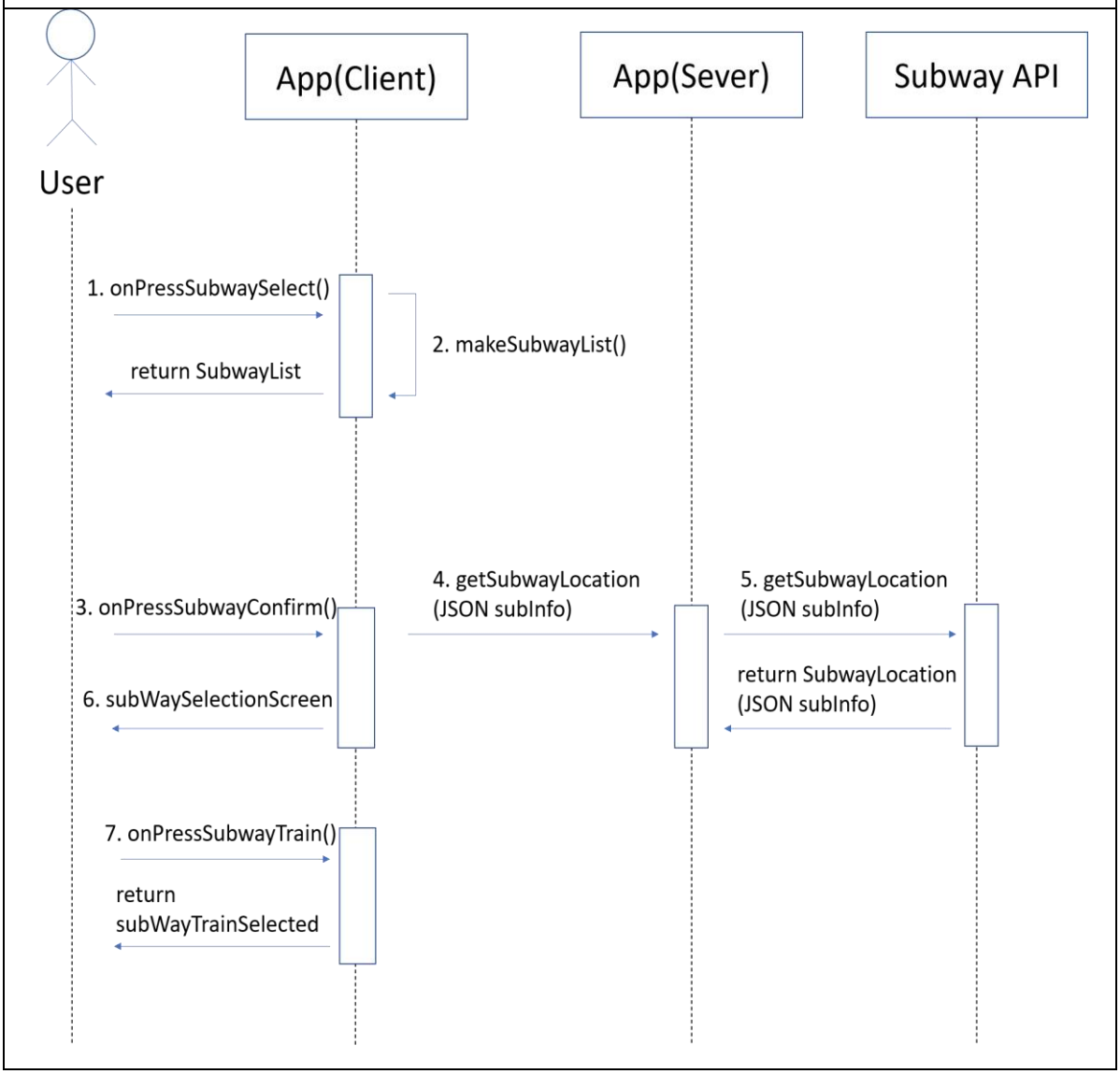
2. 로그인


User
App(Client)
App(Sever)
1. logIn()
2. checkAccountInfo()
3. logIn(JSON account)
return checkResult
return checkResult

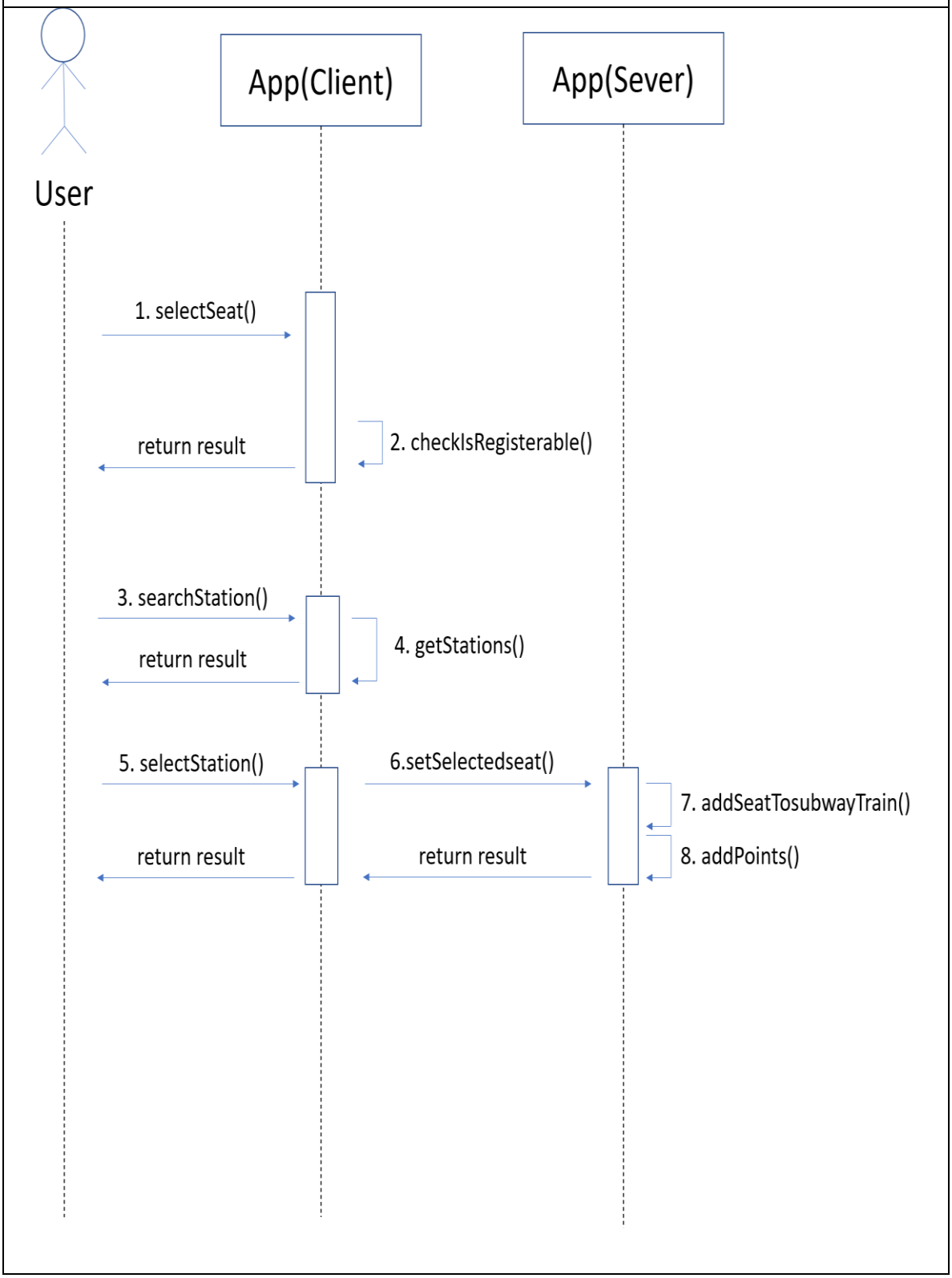
3. 로그아웃



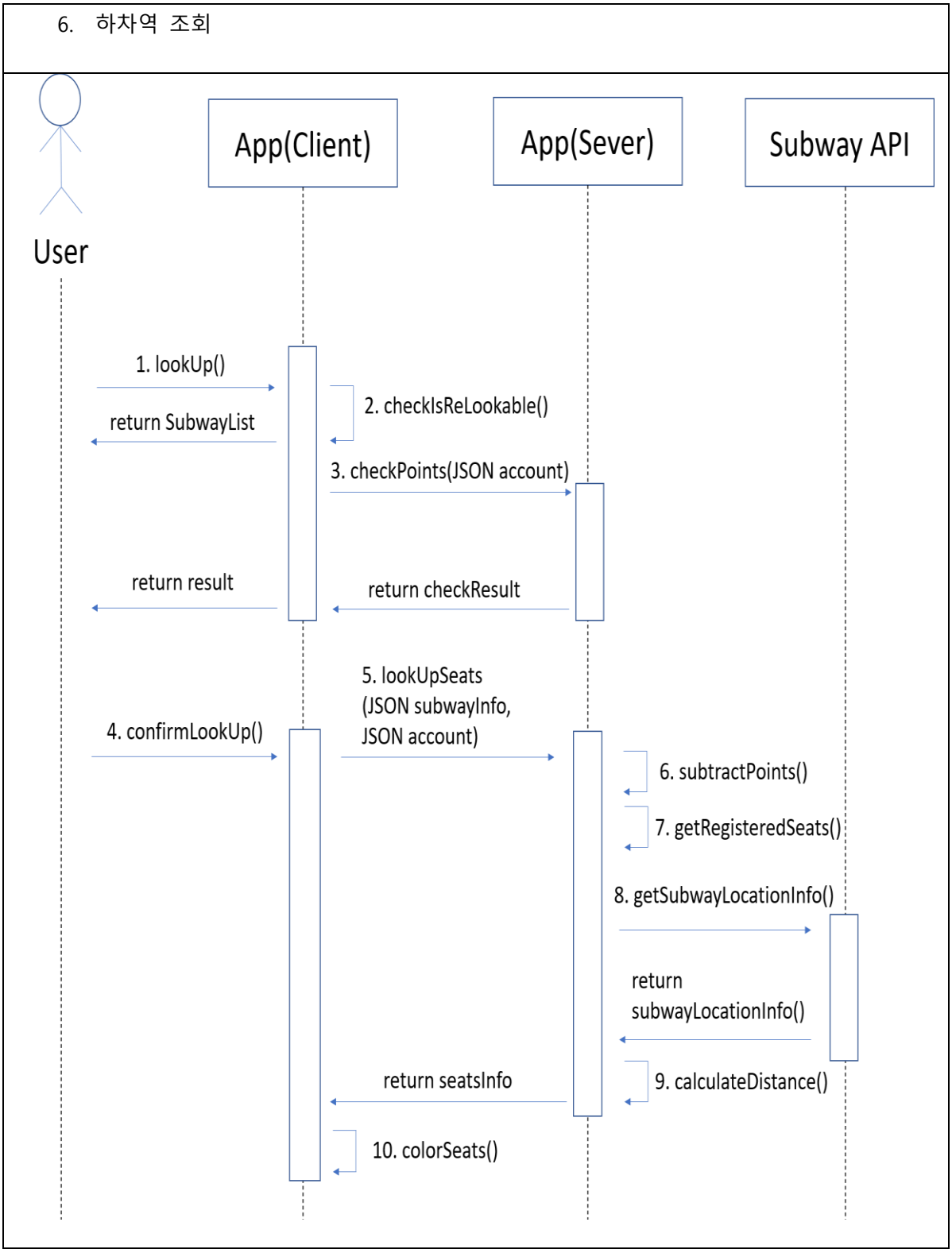
4. 지하철 차량 선택


User
App(Client)
App(Sever)
Subway API
1. onPressSubwaySelect()
2. makeSubwayList()
return SubwayList
3. onPressSubwayConfirm()
4. getSubwayLocation
(JSON subInfo)
5. getSubwayLocation
(JSON subInfo)
return SubwayLocation
(JSON subInfo)
6. subWaySelectionScreen
7. onPressSubwayTrain()
return
subWayTrainSelected

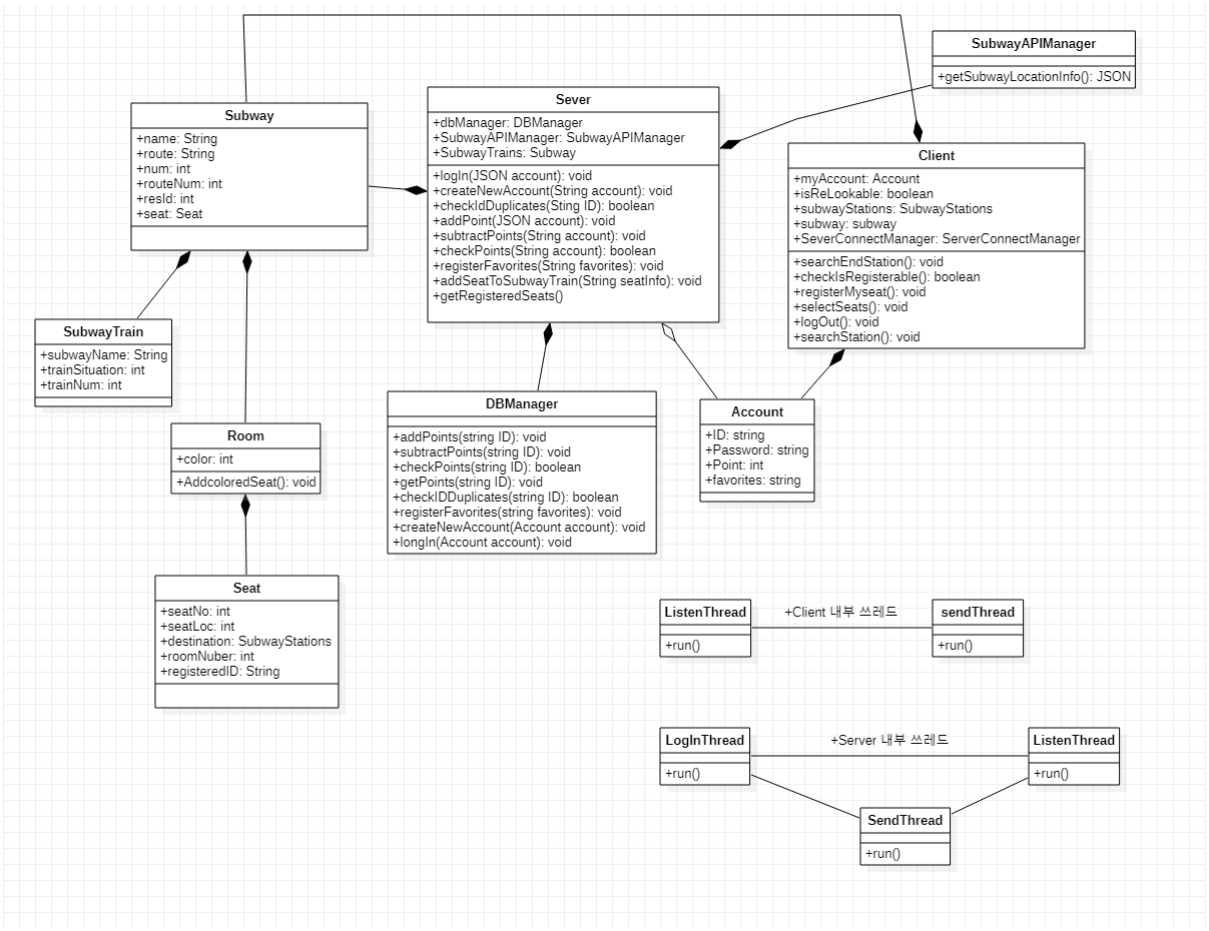
5. 하차역 선택


User
App(Client)
App(Sever)
1. selectSeat()
return result
2. checkIsRegisterable()
3. searchStation()
return result
4. getStations()
5. selectStation()
return result
6.setSelectedseat()
return result
7. addSeatTosubwayTrain()
8. addPoints()

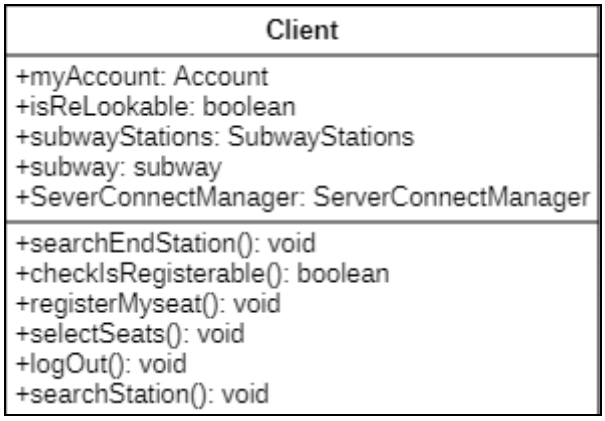
6. 하차역 조회


User
App(Client)
App(Sever)
Subway API
1. lookUp()
2. checkIsReLookable()
return SubwayList
3. checkPoints(JSON account)
return result
return checkResult
5. lookUpSeats
(JSON subwayInfo,
JSON account)
4. confirmLookUp()
6. subtractPoints()
7. getRegisteredSeats()
8. getSubwayLocationInfo()
return
subwayLocationInfo()
return seatsInfo
9. calculateDistance()
10. colorSeats()

4) Class Diagram

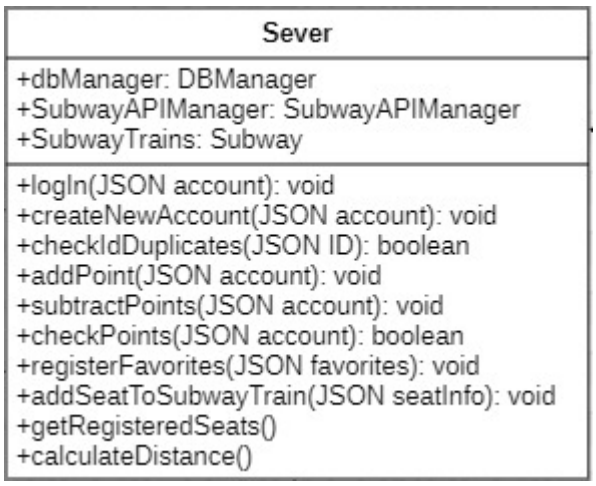


i) Client



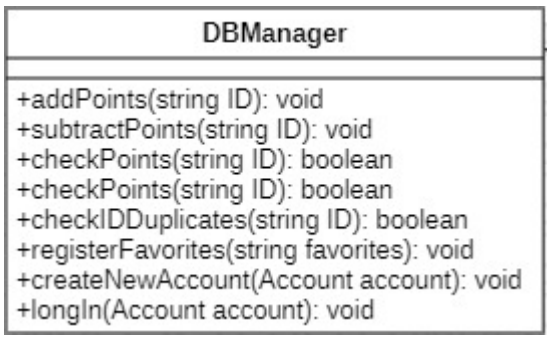
- myAccount : 내 계정 정보
- isReLookable : 재조회
- SubwayStations : 지하철 역 정보
- Subway : 차량 정보
- SeverConnectManager : 메인 서버와 연결
- seatchEndStation : 도착역 검색
- checkIsRegisterable : 등록 가능한 좌석 확인
- registerMyseat : 내 좌석 등록
- selectSeats : 좌석 선택
- logOut : 로그아웃
- seatchStation : 역 검색

ii) Sever



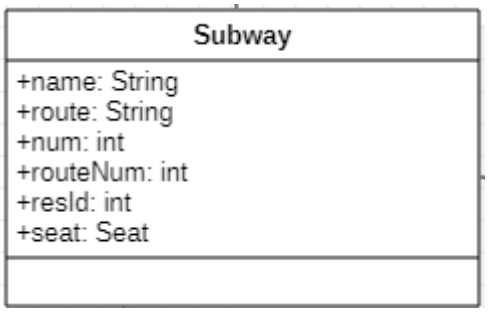
- dbManager : 유저 정보를 저장한 DB를 관리하는 Manager.
- SubwayAPIManager : 실시간 지하철 도착 정보를 JSON으로 받기 위한 Manager.
- SubWayTrains : 지하철 차량 정보.
- logIn : 로그인
- createNewAccount : 회원가입
- checkIdDuplicates : ID 중복 확인
- addPoint : 포인트 획득
- subtractPoints : 포인트 차감
- registerFavorites : 즐겨찾기 등록
- addSeatToSubwayTrain : 지하철 차량 좌석 추가
- getRegisterdSeats : 등록된 좌석.
- calculateDistance : 도착지까지의 거리 계산.

iii) DBManager



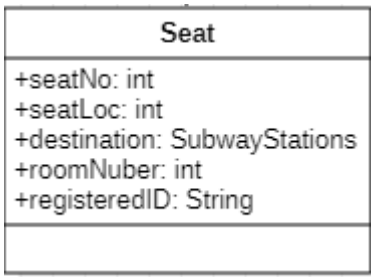
- addPoints : 포인트 적립
- subtractPoints : 포인트 차감
- checkPoints : 잔여 포인트 확인
- checkIDDuplicates : ID 중복 확인
- registerFavorites : 즐겨찾기 등록
- createNewAccount : 회원가입
- logIn : 계정 로그인

iv) Subway



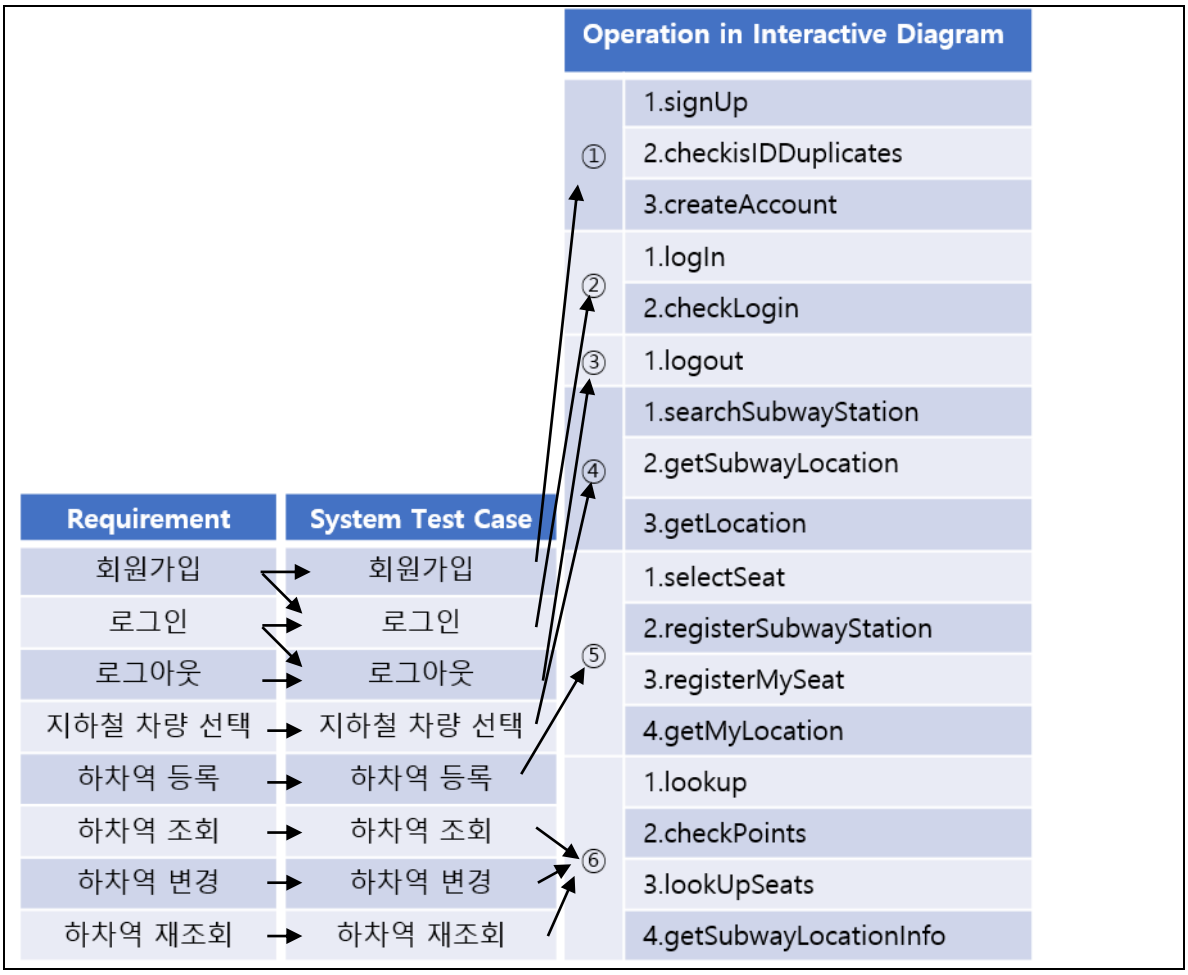
- name : 지하철 이름
- route : 지하철 노선
- num : 지하철 차량 번호
- routeNum : 상행 / 하행
- resId : 지하철 고유 번호
- seat : 좌석 정보

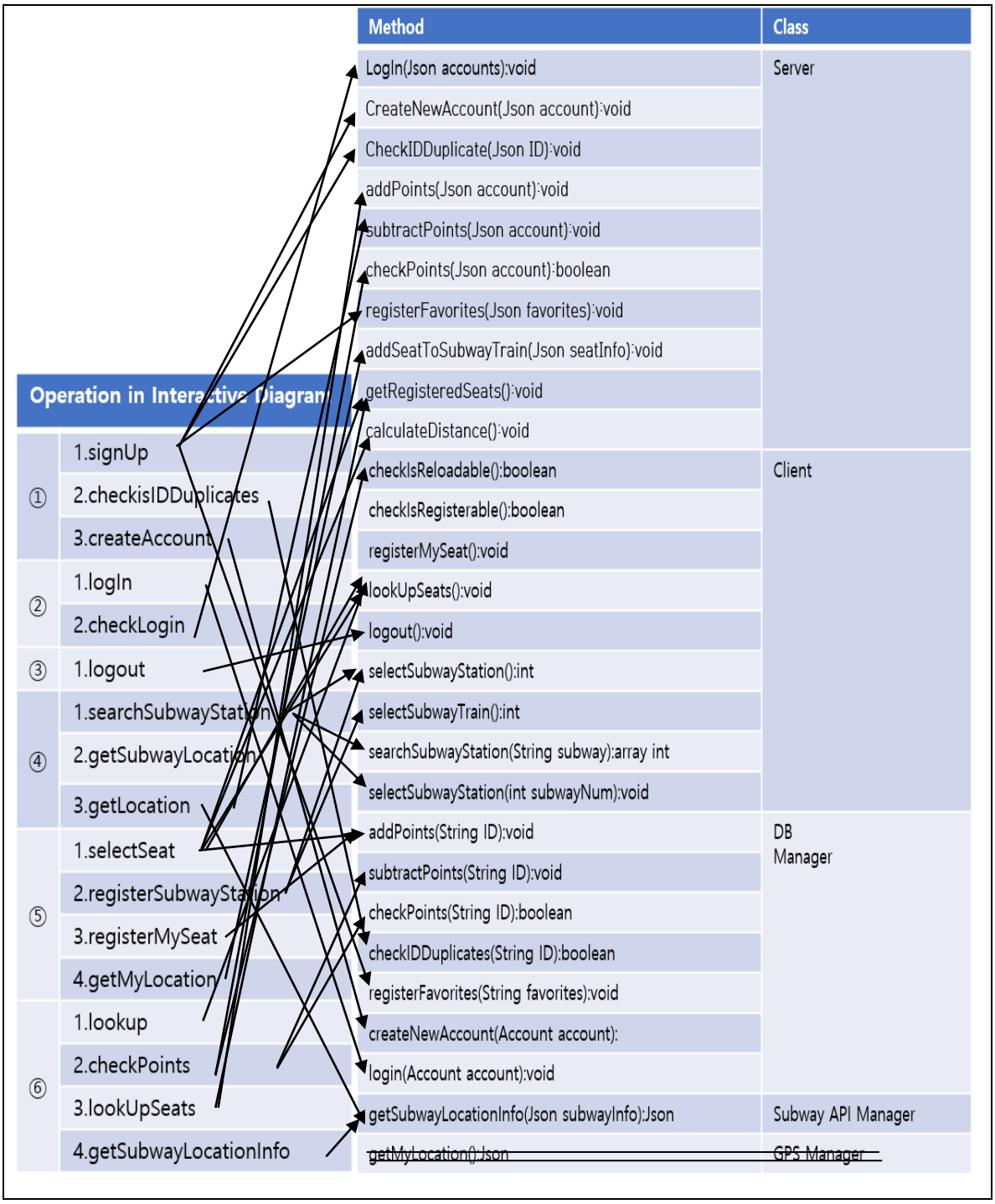
v) Seat



- seatNo : 좌석 번호
- seatLoc : 좌석 위치
- destination : 도착역
- roomNumber : 객실 칸 번호
- registeredID : 등록된 좌석의 ID

5) Traceability Matrix


Operation in Interactive Diagram
1.signUp
2.checkisIDDuplicates
3.createAccount
1.logIn
2.checkLogin
1.logout
1.searchSubwayStation
2.getSubwayLocation
3.getLocation
1.selectSeat
2.registerSubwayStation
3.registerMySeat
4.getMyLocation
1.lookup
2.checkPoints
3.lookUpSeats
4.getSubwayLocationInfo
①
②
③
④
⑤
⑥
Requirement
System Test Case
회원가입
회원가입
로그인
로그인
로그아웃
로그아웃
지하철 차량 선택
지하철 차량 선택
하차역 등록
하차역 등록
하차역 조회
하차역 조회
하차역 변경
하차역 변경
하차역 재조회
하차역 재조회


Operation in Interactive Diagram
① 1.signUp
2.checkisIDDuplicates
3.createAccount
② 1.logIn
2.checkLogin
③ 1.logout
④ 1.searchSubwayStation
2.getSubwayLocation
3.getLocation
⑤ 1.selectSeat
2.registerSubwayStation
3.registerMySeat
4.getMyLocation
⑥ 1.lookup
2.checkPoints
3.lookUpSeats
4.getSubwayLocationInfo
Method
Class
LogIn(Json accounts):void
CreateNewAccount(Json account):void
CheckIDDuplicate(Json ID):void
addPoints(Json account):void
subtractPoints(Json account):void
checkPoints(Json account):boolean
registerFavorites(Json favorites):void
addSeatToSubwayTrain(Json seatInfo):void
getRegisteredSeats():void
calculateDistance():void
Server
checkIsReloadable():boolean
checkIsRegisterable():boolean
registerMySeat():void
lookUpSeats():void
logout():void
selectSubwayStation():int
selectSubwayTrain():int
searchSubwayStation(String subway):array int
selectSubwayStation(int subwayNum):void
Client
addPoints(String ID):void
subtractPoints(String ID):void
checkPoints(String ID):boolean
checkIDDuplicates(String ID):boolean
registerFavorites(String favorites):void
createNewAccount(Account account):
login(Account account):void
DB Manager
getSubwayLocationInfo(Json subwayInfo):Json
Subway API Manager
getMyLocation():Json
GPS Manager

6. 소스 소개

1) 회원가입

```
public class JoinInActivity extends AppCompatActivity {


    String joinID;
    String joinPW1;
    String joinPW2;
    String searchtext="";
    String homeStationNum="";
    String workStationNum="";


    Boolean isIDDuplicate = true;
    Boolean isPWSame = false;

    TextView joiningID;
    TextView joiningPW1;
    TextView joiningPW2;
    TextView homeText;
    TextView workText;




    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_join_in);


        Button back_button2 = (Button)findViewById(R.id.backButton2);
        Button x_button2 = (Button)findViewById(R.id.xButton2);
        Button checkPW = (Button)findViewById(R.id.identical_button);
        Button checkID = (Button)findViewById(R.id.duplicate_button);
        Button joininButton = (Button)findViewById(R.id.joinin_button_confirm);
        Button homeButton = (Button)findViewById(R.id.home_button);
        Button workButton = (Button)findViewById(R.id.work_button);

        joiningID =(TextView) findViewById(R.id.idText);
        joiningPW1 =(TextView) findViewById(R.id.password1Text);
        joiningPW2 =(TextView) findViewById(R.id.password2Text);
        homeText=(TextView)findViewById(R.id.homeText);
        workText=(TextView)findViewById(R.id.workText);


        //뒤로가기 버튼
        back_button2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
        x_button2.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View v) {
                finish();
            }
        });

        //아이디 중복방지 버튼.
        checkID.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                joinID = joiningID.getText().toString();
                //여기에 서버 확인
                JSONObject jsonObj  = JsonWrapper.wrapSignInID(joinID);
                ServerConnectManager scMgr = new ServerConnectManager();
                scMgr.requestQuery(jsonObj.toString());
                while(scMgr.getQueryResult().equals(""));
                boolean isDuplicate = scMgr.getQueryResult().equals("Duplicate");
                String resultText = isDuplicate? "아이디 중복입니다":"아이디
사용가능합니다";
                isIDDuplicate = isDuplicate;
                Toast.makeText(getApplicationContext(),
resultText,Toast.LENGTH_SHORT).show();

            }
        });

        //비번 일치 확인 버튼
        checkPW.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                joinPW1 = joiningPW1.getText().toString();
                joinPW2 = joiningPW2.getText().toString();
                if(joinPW1.equals(joinPW2)){
                    isPWSame =true;
                    Toast.makeText(getApplicationContext(),"비밀번호 일치.
",Toast.LENGTH_LONG).show();
                } else{
                    Toast.makeText(getApplicationContext(),"비밀번호 불일치.
",Toast.LENGTH_LONG).show();
                }
            }
        });



        //회원가입 버튼. 만일을 위해 아이디 중복확인(서버연결 미구현) 및 비번 일치확인
과정을 다시 걸치고 가입을 한다.
        joininButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                joinPW1 = joiningPW1.getText().toString();
                joinPW2 = joiningPW2.getText().toString();
                joinID = joiningID.getText().toString();

                if(!isIDDuplicate && isPWSame &&joinPW1.equals(joinPW2)){//ID
중복확인도 나중에 포함
                    JSONObject jsonObj =
JsonWrapper.wrapSignUpAccount(joinID,joinPW1,homeStationNum,workStationNum);
```

```
                    ServerConnectManager scMgr = new ServerConnectManager();
                    scMgr.requestQuery(jsonObj.toString());
                    while(scMgr.getQueryResult().equals(""));
                    String signUpResult = scMgr.getQueryResult();
                    if(signUpResult.equals("ACCOUNT CREATED")) {
                        Toast.makeText(getApplicationContext(), "회원가입 완료! ",
Toast.LENGTH_LONG).show();
                        finish();
                    }else{
                        //회원가입 오류 메시지
                    }
                }
            }
        });

        //홈 즐겨찾기 검색 버튼 누를 경우
        homeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                searchtext = homeText.getText().toString();
                Intent intent = new
Intent(getApplicationContext(),FavoriteSearchActivity.class);

                intent.putExtra("searchWord",searchtext);
                intent.putExtra("favorite","home");
                startActivityForResult(intent, 105);
            }
        });
        workButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                searchtext = workText.getText().toString();
                Intent intent = new
Intent(getApplicationContext(),FavoriteSearchActivity.class);

                intent.putExtra("searchWord",searchtext);
                intent.putExtra("favorite","work");
                startActivityForResult(intent, 105);
            }
        });

    }

    //즐겨찾기 검색 후 지정하고 돌아왔을 경우.
    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode==105){//login 성공적으로 되서 다시 돌아왔을경우
isloggedin=true 로 바꾸어 진행이 가능하게 한다.
            if(data.getStringExtra("favorite").equals("home")){
               homeStationNum=data.getStringExtra("stationNum");
                homeText.setText(data.getStringExtra("stationName"));
            }else if(data.getStringExtra("favorite").equals("work")){
                workStationNum=data.getStringExtra("stationNum");
                workText.setText(data.getStringExtra("stationName"));
```

```
            }

        }

    }
}
```

2) 로그인

```
public class LogInActivity extends AppCompatActivity {

    String currentID;
    String currentPW;
    TextView loginID;
    TextView loginPW;
    MyApplication myApp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_log_in);

        myApp = (MyApplication)getApplication();

        loginID =(TextView) findViewById(R.id.idText);
        loginPW =(TextView) findViewById(R.id.passwordText);
        Button join_button = (Button)findViewById(R.id.joinin_button);
        Button login_button = (Button)findViewById(R.id.login_button);
        Button back_button1 = (Button)findViewById(R.id.backButton1);
        Button x_button1 = (Button)findViewById(R.id.xButton1);

        //회원가입 버튼 누를경우 회원가입창으로 넘어간다.
        join_button.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v){
                    Intent intent = new
Intent(getApplicationContext(),JoinInActivity.class);
                    startActivityForResult(intent, 103);
            }
        });

        //로그인 버튼누를경우. 서버에서 확인후(아직 미구현) 로그인 된 상태로 첫화면으로
진행한다.
        login_button.setOnClickListener(new View.OnClickListener(){
            public void onClick(View v){
                currentID = loginID.getText().toString();
                currentPW = loginPW.getText().toString();
                Log.d("test",currentID);
                //서버에서 확인과정 거치고
                JSONObject jsonObj  =
JsonWrapper.wrapSignInAccount(currentID,currentPW);
                ServerConnectManager scMgr = new ServerConnectManager();
                scMgr.requestQuery(jsonObj.toString());
                while(scMgr.getQueryResult().equals(""));
                boolean isLogInSuccess = (!scMgr.getQueryResult().equals("LOGIN
FAIL"));
                if(isLogInSuccess) {
```

```
                    //로그인 성공시
                    //집 , 직장으로 등록된 역 가져오기;
                    JSONObject obj;
                    Vector<String> favorites = new Vector<String>();
                    try {
                        obj = new JSONObject(scMgr.getQueryResult());
                        favorites.add((String)obj.get("FAVORITE1"));
                        favorites.add((String)obj.get("FAVORITE2"));
                        //myApp.setHomeStation((String)obj.get("FAVORITE1"));
                       //myApp.setWorkStation((String)obj.get("FAVORITE2"));
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }

                    myApp.setLoggedIn(true);
                    myApp.setLoggedId(currentID);
                    myApp.setHomeStation(favorites.get(0));
                    myApp.setWorkStation(favorites.get(1));

                    finish();
                }else{
                    //로그인 실패시
                    Toast.makeText(getApplicationContext(), "로그인 실패. 아이디와
비밀번호 확인해주세요 ", Toast.LENGTH_LONG).show();
                }
            }
        });
```

3) 로그아웃

```
//로그아웃 버튼
    logoutButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            myApp.setLoggedIn(false);
            myApp.setLoggedId("");
            myApp.setEndStationNumber(0);
            myApp.setSelectedCarLocation(0);
            myApp.setSelectedSeat(0);
            myApp.setCarNumber(0);
            myApp.setPayed(false);
            myApp.setTrainNumber(0);
            myApp.setStationNumber(0);
            myApp.setRoute(0);

            Toast.makeText(getApplicationContext(),"로그아웃
성공하셨습니다.",Toast.LENGTH_LONG).show();
        }
    });

}
```

4) 지하철 차량 선택

```
public class SearchEndStationActivity extends AppCompatActivity {
    MyApplication myApp;

    boolean homeStationAdded=false;
    boolean workStationAdded=false;
    String searchText;
    SubwayAdapter adapter;
    String[][]
station={{"장암","1007000709"},{"도봉산","1007000710"},{"수락산","1007000711"},{"마
들","1007000712"},{"노원","1007000713"},{"중계(한국성서대)","1007000714"},{"하계(을
지원 을지병원)","1007000715"},{"공릉(서울과학기술대)","1007000716"},
          {"태릉입구","1007000717"},
{"먹골","1007000718"},{"중화","1007000719"},{"상봉(시외버스터미널)","1007000720"},{"
면목(서일대입구)","1007000721"},{"사가정(녹색병원)","1007000722"},{"용마산","10070007
23"},{"중곡","1007000724"},{"군자(능동)","1007000725"},

{"어린이대공원(세종대)","1007000726"},{"건대입구","1007000727"},{"뚝섬유원지","100700
0728"},{"청담(한국금거래소)","1007000729"},{"강남구청","1007000730"},{"학동","100700
0731"},{"논현","1007000732"},{"반포","1007000733"},{"고속터미널","1007000734"},{"내
방","1007000735"},

{"아수","1007000736"},{"남성","1007000737"},{"숭실대입구(살피재)","1007000738"},{"상
도","1007000739"},{"장승배기","1007000740"},{"신대방삼거리","1007000741"},{"보라매",
"1007000742"},{"신풍","1007000743"},{"대림(구로구청)","1007000744"},{"남구로","10070
00745"},

{"가산디지털단지(마리오아울렛)","1007000746"},{"철산","1007000746"},{"광명사거리","10
07000748"},{"천왕","1007000749"},{"온수(성공회대입구)","1007000750"},{"까치울","1007
000751"},{"부천종합운동장","1007000752"},{"춘의","1007000753"},{"신중동","1007000754
"},

{"부천시청","1007000755"},{"상동","1007000756"},{"삼산체육관","1007000757"},{"굴포천
","1007000758"},{"부평구청","1007000759"}};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_search_end_station);

        myApp=(MyApplication)getApplication();

        Intent passedIntent = getIntent();

        //Toast.makeText(getApplicationContext(),""+carNumber+" "+seatNumber+"
"+trainNumber, Toast.LENGTH_LONG).show();

        final TextView textView  =(TextView)findViewById(R.id.searchtext);
        Button searchButton = (Button)findViewById(R.id.searchbutton);
        Button backButton = (Button)findViewById(R.id.backButton1);
        Button xButton = (Button)findViewById(R.id.xButton1);
        Button homeButton = (Button)findViewById(R.id.homeshortcut);
        Button workButton = (Button)findViewById(R.id.workshortcut);
```

```
        ListView listView = (ListView)findViewById(R.id.subwayListView);

        adapter = new SubwayAdapter();


        searchButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                adapter.clear();
                searchText  = textView.getText().toString();
                for(int i=0;i<station.length;i++){
                    if(station[i][0].contains(searchText)){

                        adapter.addItem(new
SubwayItem(station[i][0],Integer.parseInt(station[i][1]),R.drawable.sevenline,stat
ion[i-1][0],1));
                        adapter.notifyDataSetChanged();
                    }
                }
            }
        });



        backButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
        xButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
        homeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                for(int i=0;i<station.length;i++){
                    if(station[i][1].equals(myApp.getHomeStation())){

                        adapter.addItem(new
SubwayItem(station[i][0],Integer.parseInt(station[i][1]),R.drawable.sevenline,stat
ion[i-1][0],1));
                        adapter.notifyDataSetChanged();
                    }
                }
            }
        });

        workButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                for(int i=0;i<station.length;i++){
                    if(station[i][1].equals(myApp.getWorkStation())){

                        adapter.addItem(new
```

```
SubwayItem(station[i][0],Integer.parseInt(station[i][1]),R.drawable.sevenline,stat
ion[i-1][0],1));
                        adapter.notifyDataSetChanged();
                    }
                }
            }
        });

        listView.setAdapter(adapter);

        //해당하는 도착 역 선택
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {

                SubwayItem item = (SubwayItem) adapter.getItem(position);
                boolean rightPath;
                if(myApp.getRoute()==0){
                    rightPath= (item.getNum()<myApp.getStationNumber());
                }else{
                    rightPath= (item.getNum()>myApp.getStationNumber());
                }

                if(rightPath){
                    Toast.makeText(getApplicationContext(),item.getName()+"역 선택",
Toast.LENGTH_LONG).show();
                    myApp.setEndStationNumber(item.getNum());
                    Intent intent1 = new Intent(getApplicationContext(),
EndActivity.class);

                    JSONObject json =
JsonWrapper.wrapSeatInfo(myApp.getLoggedId(),myApp.getTrainNumber(),myApp.getCarNu
mber(),item.getNum(),myApp.getSelectedCarLocation(),myApp.getSelectedSeat());
                    ServerConnectManager scMgr = new ServerConnectManager();
                    scMgr.requestQuery(json.toString());

                    startActivityForResult(intent1, 510);
                }else{
                    Toast.makeText(getApplicationContext(),"방향과 맞지 않는 역
선탁하셨습니다.", Toast.LENGTH_LONG).show();
                }


            }
        });

    }



    class SubwayAdapter extends BaseAdapter {
        ArrayList<SubwayItem> items = new ArrayList<SubwayItem>();

        @Override
        public int getCount() {
            return items.size();
```

```
        }

        public void addItem(SubwayItem item){
            items.add(item);
        }

        @Override
        public Object getItem(int position) {
            return items.get(position);
        }

        @Override
        public long getItemId(int position) {
            return position;
        }

        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            SubwayItemView view = new SubwayItemView(getApplicationContext());

            SubwayItem item = items.get(position);
            view.setName(item.getName());
            view.setImage(item.getResId());
            //view.setRoute(item.getRoute());
            return view;
        }
        public void clear(){items.clear();}
    }
}
```

5) 좌석 등록

```
//자리를 arraylist 로 만들어 왼쪽/중간/오른쪽 칸으로 분류하여 배분.
//자리 색깔 코딩. 포인트 낸 경우만 색 보여주고 아닌 경우에는 안보여주기.
if(payed) {
    ServerConnectManager scMgr = new ServerConnectManager();
    JSONObject jsonObj  =
JsonWrapper.wrapSubwayTrainSeatInfo(trainNumber,carNumber);
    scMgr.requestQuery(jsonObj.toString());
    while(scMgr.getQueryResult().equals(""));
    JSONArray jsonArray = null;
    try {
        //등록된 좌석정보 들어있음 키 : LOCATION - 좌석 위치(처음 중간 끝), SEATNO -
좌석 번호, DESTINATION - 123 컬러값
        jsonArray = new JSONArray(scMgr.getQueryResult());
        for (int i = 0; i< jsonArray.length();i++) {
            JSONObject obj = (JSONObject) jsonArray.get(i);

            int seatNo = (Integer) obj.get("SEATNO");
            int location = (Integer) obj.get("LOCATION");
            int destination = (Integer) obj.get("DESTINATION");

            seatSituation[location][seatNo]=destination;
        }

    } catch (JSONException e) {
```

```
        e.printStackTrace();
    }
for (int j = 0; j < 7; j++) {
    if (seatSituation[0][j] == 0)
        seatTop.get(j).setBackgroundResource(R.drawable.person1);
    else if (seatSituation[0][j] == 1)
        seatTop.get(j).setBackgroundResource(R.drawable.personred);
    else if (seatSituation[0][j] == 2)
        seatTop.get(j).setBackgroundResource(R.drawable.personorange);
    else if (seatSituation[0][j] == 3)
        seatTop.get(j).setBackgroundResource(R.drawable.persongreen);
}

....
```

5) 포인트, 등록된 좌석 조회

```
//검색 버튼. 누르면 팝업창 뜨고 결제시스템
        searchStartButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //여기에 포인트 시스템 넣으면 된다. 서버에서 payed 된 순간 시간을  재고
아이디에 payed=true 넣은 후에 시간 지나면 payed=false 되는 방식으로 해아할듯.
                ServerConnectManager scMgr = new ServerConnectManager();
                JSONObject jsonObj  =
JsonWrapper.wrapSubwayTrainInfo(myApp.isPayed(),myApp.getLoggedId(),myApp.getTrain
Number());
                scMgr.requestQuery(jsonObj.toString());
                while(scMgr.getQueryResult().equals(""));
                if(!scMgr.getQueryResult().equals("POINTNOTENOUGH")) {
                    myApp.setPayed(true);
                }
                JSONArray jsonArray = null;
//                //포인트가 충분해서 조회가 되었을 경우
                if(myApp.isPayed()) {
                    try {
                        // 이 JSONARRAY 에 0~8 인덱스에  차량의 색이 들어가있음  키 값 :
COLOR - 1 2 3 중  하나
                        jsonArray = new JSONArray(scMgr.getQueryResult());
                        for(int i=0;i<jsonArray.length()-1;i++) {
                            JSONObject obj = (JSONObject)jsonArray.get(i);
                            int roomNumber = (Integer)obj.get("ROOMNO");
                            crowdedSituation[roomNumber] = (Integer)
((JSONObject)obj).get("COLOR");
                        }

                        // 9 번째 인덱스에 포인트가 들어가있음
                        point = (Integer)
((JSONObject)jsonArray.get(jsonArray.length()-1)).get("POINT");
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }

                    searchStartButton.setBackgroundColor(Color.LTGRAY);
```

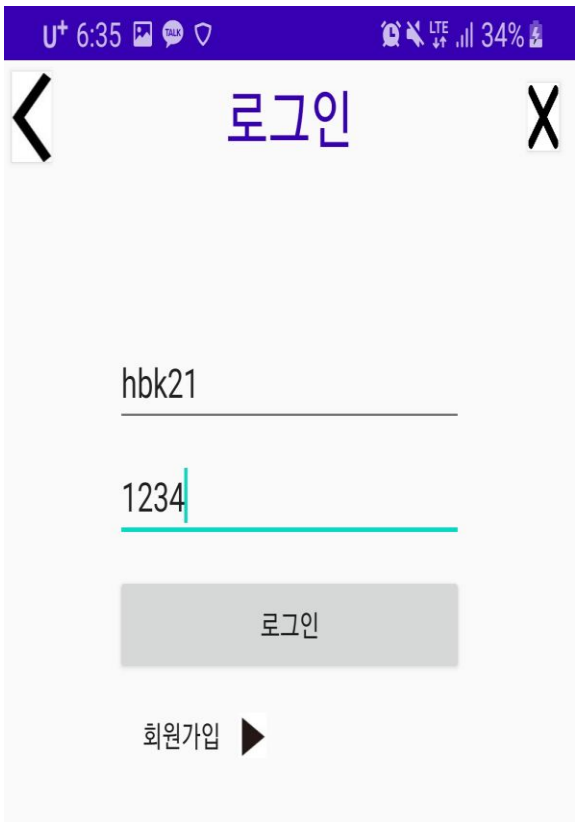
```
                    searchStartButton.setText("검색완료");
                    if(myApp.isPayedText()){
                        Toast.makeText(getApplicationContext(), "재조회 합니다.",
Toast.LENGTH_LONG).show();
                    }else{
                        Toast.makeText(getApplicationContext(), "포인트가 100 점
차감되어 " + point + "점 남았습니다", Toast.LENGTH_LONG).show();
                        myApp.setPayedText(true);
                    }

                    for (int i = 0; i < 9; i++) {
                        switch (crowdedSituation[i]) {
                            case 0:
                                break;
                            case 1:

cars.get(i).setBackgroundResource(R.drawable.subwaymapred1);
                                break;
                            case 2:

cars.get(i).setBackgroundResource(R.drawable.subwaymaporange1);
                                break;
                            case 3:

cars.get(i).setBackgroundResource(R.drawable.subwaymapgreen1);
                                break;
                        }
                    }
                }
                else{
                    Toast.makeText(getApplicationContext(), "포인트 부족으로 검색을
못합니다." , Toast.LENGTH_LONG).show();
                }
                //Log.d("time",""+payedTime);
            }
        });
```

7. 시연

1) 메인화면

- 왼쪽 상단의 로그아웃 버튼 / 중앙에 검색 창과 검색 버튼 구현.
- 검색창에 7호선 역 이름을 검색 가능.
- 7호선 역 이름 검색어 자동완성 기능 구현.
- 역 이름을 검색하면 로그인 창으로 넘어감.
- 로그아웃은 메인 화면에서만 가능.
- 로그아웃 시 "로그아웃 성공하셨습니다." 메시지가 출력되며 로그아웃.

2) 로그인 화면

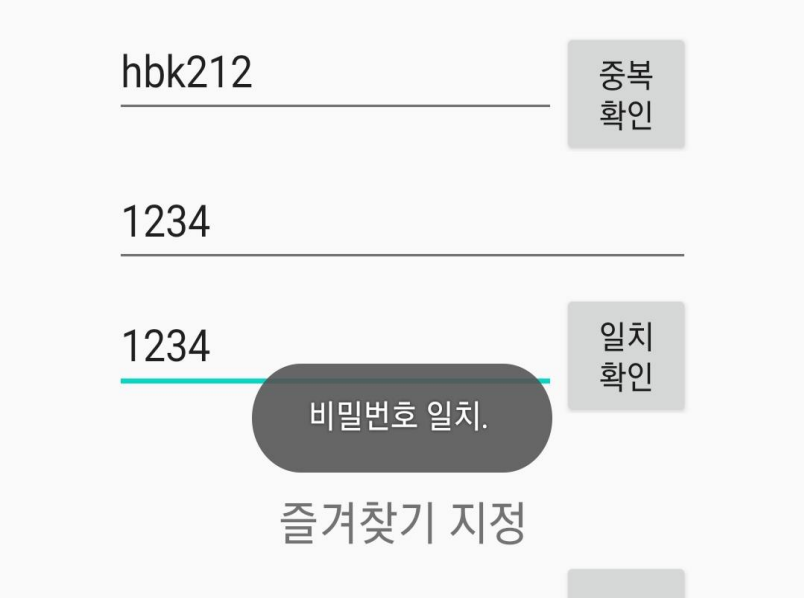


- 뒤로가기 버튼 / 창 닫기 버튼 / ID, PW 입력창 / 로그인 버튼 / 회원가입 버튼으로 구성.
- 계정이 있는 경우 ID와 PW를 입력하고 로그인.
- ID와 PW를 제대로 입력 후 로그인 버튼을 누르면 메인 화면으로 이동하고, 메시지가 뜨며 로그인 성공.
- 계정이 없는 경우 회원가입 버튼을 눌러 이동.

3) 회원가입 화면

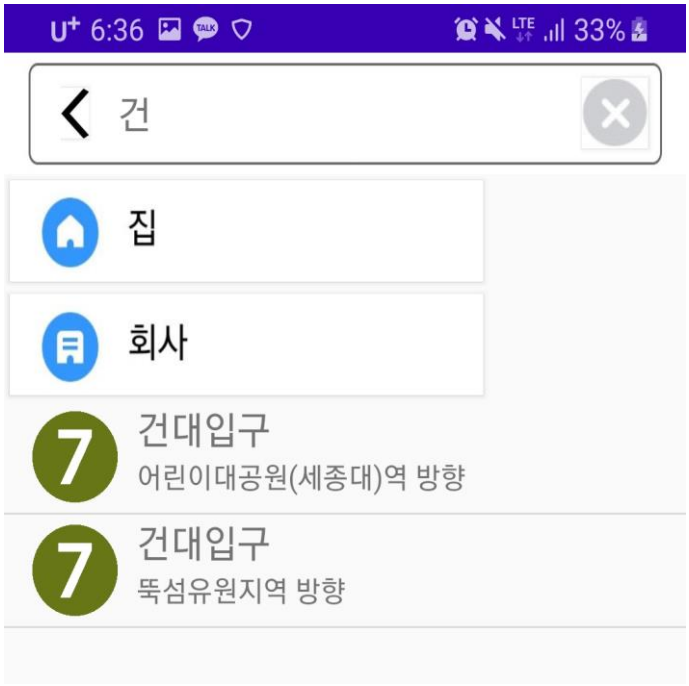
- ID 체크, PW 체크, 즐겨 찾기 세 가지의 입력창과 ID 중복 확인 버튼, PW 일치 확인 버튼, 역 검색 버튼 두 개, 회원가입 버튼으로 구성.

- ID 중복 확인 버튼을 눌러 서버에 등록되지 않은 ID이면 "아이디가 사용 가능합니다." 메시지가 출력, 중복된 아이디라면 "아이디 중복입니다." 메시지가 출력된다.



- PW 입력 칸에 동일한 비밀번호를 입력 후, 일치 확인 버튼을 누르면 비밀번호 일치 메시지가 출력된다

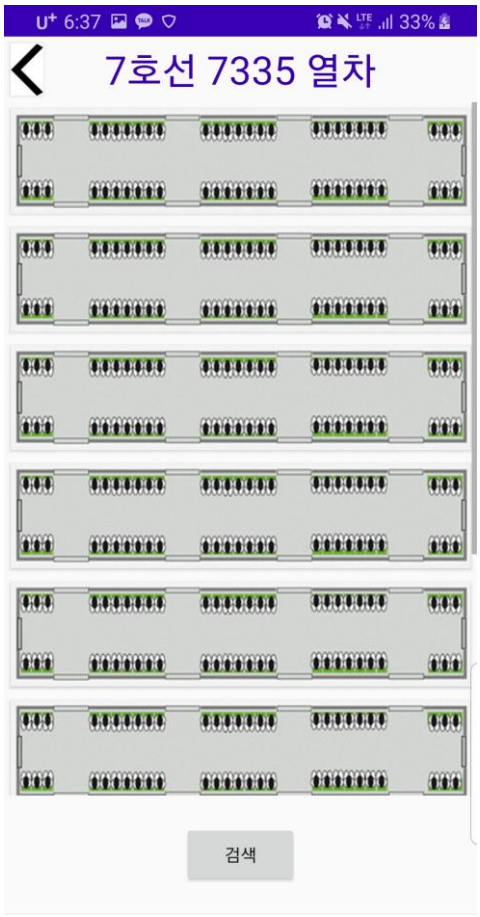
4) 열차 선택

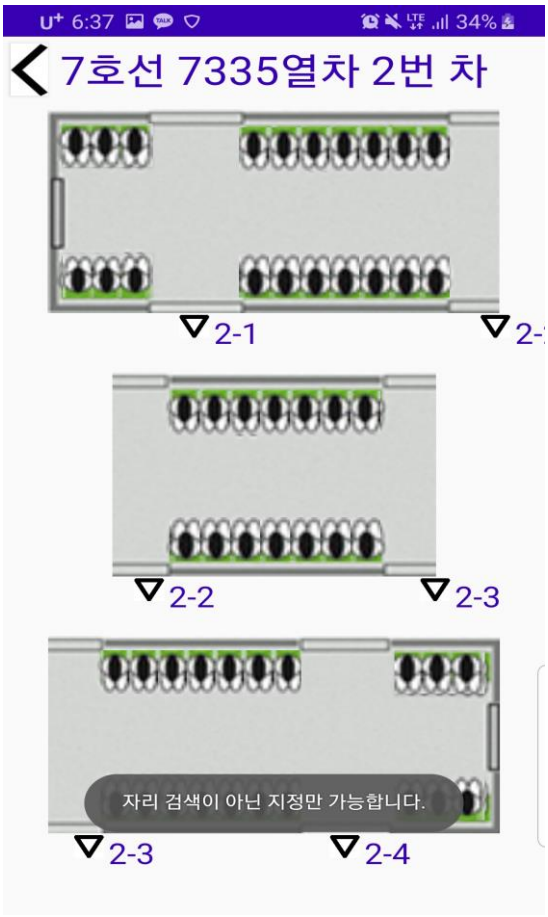


- 탑승하려는 지하철 역 검색.
- 검색하려는 지하철역명이 리스트에 자동 완성.
- 해당 역의 노선 방향을 선택할 수 있음.

- 실시간 지하철 도착 API 정보를 받아 탑승 중 / 탑승 예정인 지하철을 선택할 수 있음.
- 지하철을 선택하면 좌석 선택 화면으로 이동.

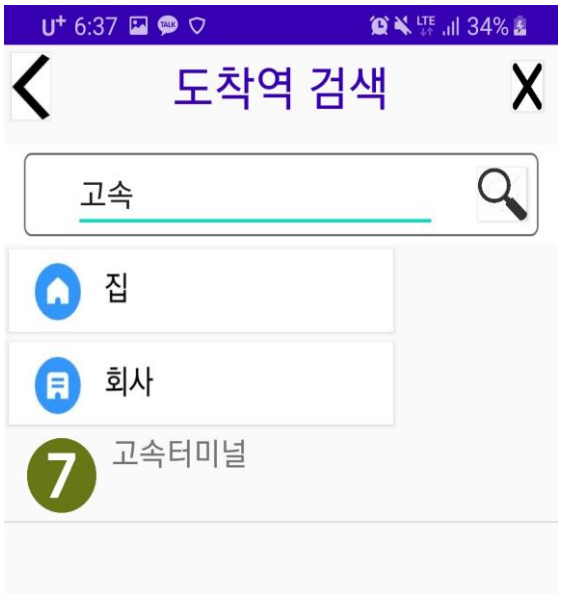
5) 좌석 선택





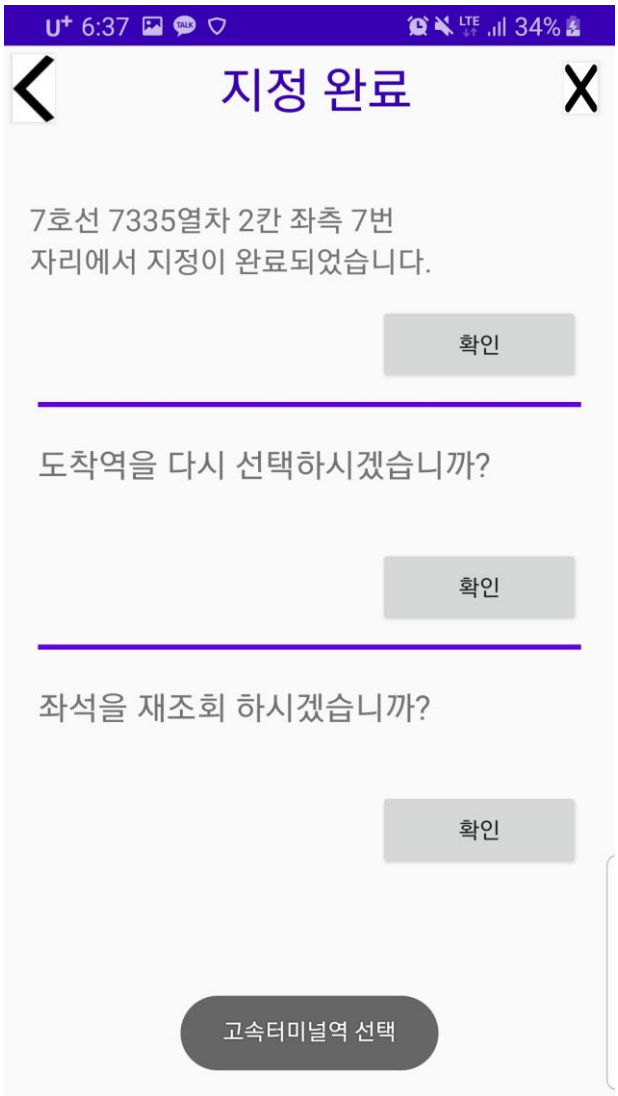
- 열차를 선택하면 해당 열차의 칸으로 나뉘어져 있음.
- 칸을 선택하면 좌석 현황 화면이 팝업.
- 좌석을 선택하면 도착 역 검색 화면으로 이동.

6) 하차 역 선택



- 자신이 내릴 역을 검색한 후 선택.

- 자신이 선택한 노선과 반대 방향인 역을 검색 / 선택하면 "방향과 맞지 않는 역을 선택하셨습니다." 메시지 출력.



- 도착역까지 선택이 됐으면 "해당 역 선택" 메시지가 팝업.
- 도착역을 잘못 설정했다면 재 지정을 할 수 있음.
- 포인트를 소모해서 등록된 좌석 현황을 재 조회할 수 있음.

8. TEST CASE / Success Criteria / 고찰

1) TEST CASE

NUM	Test 항목	Test Step	Expected result	Result
1	회원가입	1. hbk21" 입력, ID 중복확인 2. "1234" 입력, 비밀번호 일치 확인 3. "도봉산", "건대입구" 입력 4. 회원가입 완료 창 팝업 확인	ID "hbk21", PW "1234", 집 근처 역 "도봉산", 직장 근처 역 "건대입구"라는 사용자 데이터가 DB에 저장되고 새로운 계정이 생성.	◯
2	로그인	1. "hbk21", "1234" 입력 2. 로그인 결과 창 팝업 확인	로그인이 되어 앱 기능을 이용할 수 있음.	◯
3	로그아웃	1. 로그아웃 버튼 선택 2. 로그아웃 결과 창 팝업 확인	앱에서 로그아웃이 되어 기능을 이용하려 할 때 로그인 창이 팝업됨.	◯
4	지하철 차량 선택	1. "건대입구" 검색 후 선택 2. "건대입구"역으로 진입하는 열차들 중 탑승, 혹은 탑승 예정인 지하철 차량 선택 3. 지하철 차량 선택 결과 창 팝업 확인.	결과 팝업창에 자신이 선택한 지하철 차량의 정보가 제대로 적용됨.	◯
5	등록된 좌석 조회	1.등록된 좌석 조회 버튼 선택 2.조회 여부 확인 창 팝업 확인 3.보유 포인트 차감 확인, 조회 완료 확인	포인트가 설정한 수치만큼 차감 다른 사람이 등록한 좌석 조회 화면으로 이동.	◯

6	좌석 등록	1. 사용자가 앉아 있는 좌석 선택 2. "영등포 구청역" 검색 3. "영등포 구청역"선택 및 확인 창 팝업 확인 4. 1에서 선택한 좌석에 "영등포 구청역" 등록 5. 보유 포인트 증가 확인, 등록 결과 창 팝업 확인	자신이 앉아 있는 좌석이 건대입구에서 영등포 구청역까지 가는 것으로 등록됨. 포인트가 설정한 만큼 증가함.	◯
7	하차 역 변경	1. 하차역 변경 버튼 선택 2. "신촌 역 " 검색 3. "신촌 역 " 선택 및 확인 창 팝업 확인 4. 등록 결과 창 팝업 확인	"영등포 구청역"에서 "신촌 역 " 으로 정보가 갱신됨.	◯
8	좌석 재조회	1. 등록된 좌석 재조회 선택 2. 좌석 조회 시간 잔여 확인 3. 재조회 완료 팝업창 확인	남은 시간동안 4번의 화면을 다시 볼 수 있음.	△
9	Privacy	사용자가 등록된 하차역을 명시적으로 표기하지 않고, 하차역까지 남은 역 수에 따라서 다른 색으로 표시되는 것을 확인	등록된 좌석에 역 정보는 표시되지 않고 색으로만 표시됨.	◯
10	Ethical	연속으로 좌석 등록을 함.	자신이 처음에 등록했던 역에 도착할 때 까지 포인트 중복 획득 불가.	◯

2) Success Criteria

다수의 사용자가 동시에 같은 차량 내에서 하차역을 표시했을 때 누락 없이 제대로 표시되는가?	
1. 팀원 3명 + 공기계 총 4개 기기 Test	**Pass**
2. 더미데이터 100개 Test	**Pass**
최종 결과	**Success**

3) 고찰

- 포인트 부정 획득 방지를 위해 GPS API를 사용하려 했으나 지하에 있는 지하철 역 / 열차 내부에서 GPS와 Network Location 모두 불통.
- GPS를 이용한 위치 대조를 포기함. 대신 서버내에 본인이 등록한 좌석이 있다면 그 좌석에 설정된 목표역까지 도착하기 전까지는 아무리 재등록해도 포인트를 못 얻도록 처리.
- 객실 / 객실 내부 좌석 조회할 때 조금씩 지연이 걸림.
- API에서 정보를 가져와서 하차역까지 남은 수를 구하는 방식이라 API에서 지연이 걸리는 것으로 추정.
- 객실 조회할 때 좌석들을 한꺼번에 조회해서 보여주는 방식이나 서버가 일정시간마다 자동으로 갱신하는 방법으로 개선하면 반응속도가 빨라질 것으로 예상.
- TestCase 8번 좌석 재 조회 기능은 제대로 동작하나 제한 시간 기능을 넣지 못했음.
- 처음 설계 / 계획한 것과 달라진 점이 많아 아쉬웠음.