Where is my space ?

3. Design

학번 : 22212043
이름 : 서민지
E-Mail : mjlove6630@naver.com

[Revision history]

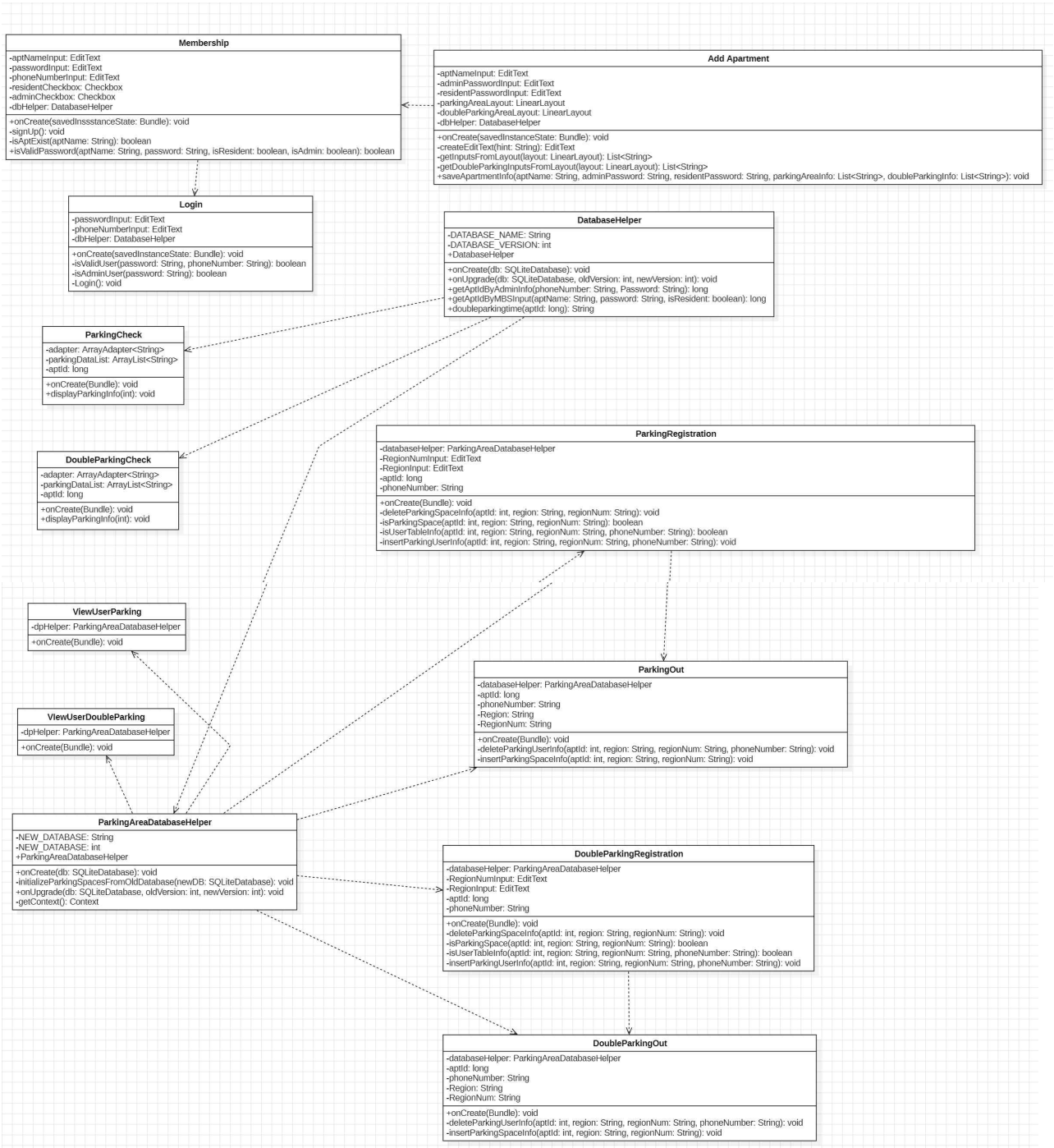
Revision date	Version #	Description	Author
05/21/2024	1.00	First Draft	

= Contents =

1. Introduction

WIMS 앱은 현대시대의 주차 문제를 해결하기 위해 만들어졌다. 현재 아파트에 사는 많은 사람들은 자신의 차량을 보유하고 있다. 즉 한 집 당 하나 이상의 차량을 가지고 있는 것은 흔한 일이다. 하지만 아파트에 주차를 할 수 있는 주차장의 공간은 한정적이기에 주차 문제가 발생하게 된다. 그 중 이중 주차 문제는 다른 사람들이 공간을 편리하게 사용하지 못할 뿐만 아니라 이중 주차를 한 사람들의 정보를 모르는 경우 연락을 할 수 없기 때문에 개인적인 일상까지 피해가 가게된다.
WIMS는 아파트 주차공간을 예약, 조회, 취소 할 수 있고 더해 이중 주차 가능한 구역을 선정하여 가능 시간대에만 등록할 수 있도록 하고, 이중 주차를 한 주민들의 정보를 공유할 수 있기 때문에 위에 언급한 문제를 해결할 수 있다.
하지만 고려해야하는 사항들이 있다. 사용자의 권한에 따라 사용되는 기능을 구별해야하며 관리자가 제시한 이중 주차 시간대 설정을 확실히 적용해야한다. 이러한 기능을 명확히 하기 위하여 Design단계에서 class, sequence, state machine 다이어그램을 만들 것이다. Diagrams과 Implementation requirements는 앱이 올바르게 구현되도록 돕고 개발과정에서 커뮤니케이션을 더 원활히 할 수 있도록 해줄 것이다.

2. Class diagram



1) Membership

Attributes
aptNameInput: 아파트 이름
passwordInput: 사용자 비밀번호
phoneNumberInput: 사용자 전화번호
residentCheckbox: 주민인지 확인
adminCheckbox: 관리자인지 확인
dbHelper: DatabaseHelper class 접근
Methods
signUp(): void: 멤버십 가입할 때 사용자의 여러 입력 처리들 관리
isAptExist(aptName:String): boolean: 아파트 이름으로 아파트가 존재하는지 확인
isVaildPassword(aptName:String,password:String,isResident:boolean,isAdmin:boolean): boolean: 입력된 정보가 해당 아파트의 사용자 비밀번호와 일치하는지 확인

2) App Apartment

Attributes
aptNameInput: 아파트 이름
adminPasswordInput: 관리자 비밀번호
residentPasswordInput: 주민 비밀번호
parkingAreaLayout: 일반 주차 정보 입력 레이아웃
doubleParkingLayout: 이중 주차 정보 입력 레이아웃
dpHelper: DatabaseHelper class 접근
Methods
createEditText(hint:String): EditText: EditText 생성
getInputsFromLayout(layout:LinearLayout): List<String>: 입력된 일반 주차 정보 가져오기
getDoubleParkingInputsFromLayout(layout:LinearLayout): List<String>: 입력된 이중 주차 정보 가져오기
saveApartmentInfo(aptName:String,adminPassword:String,residentPassword:String, parkingAreaInfo:List<String>,doubleParkingInfo:List<String>): void: 입력된 아파트 정보 저장하기

3) Login

Attributes
passwordInput: EditText: 사용자 비밀번호
phoneNumberInput: EditText: 사용자 전화번호
dbHelper: DatabaseHelper class 접근
Methods
isValidUser(password:String,phoneNumber:String): boolean: 가입된 사용자인지 확인
isAdminUser(password:String): boolean: 관리자인지 판단
Login(): void: 로그인 할 때 사용자의 여러 입력 처리들 관리

4) DatabaseHelper

Attributes
DATABASE_NAME: String: 데이터베이스 이름 DATABASE_VERSION: int: 데이터베이스 버전
Methods
DatabaseHelper(): SQLite 데이터베이스 접근 getAptIdByAdminInfo(phoneNumber:String,Password:String): long: 사용자 정보로 아파트 id 가져오기 getAptIdByMBSInput(aptName:String,password:String,isResident:boolean): long: 아파트 이름으로 아파트 id 가져오기 doubleparkingtime(aptId:long): String: 아파트 id로 이중 주차 가능 시간 가져오기

5) ParkingAreaDatabaseHelper

Attributes
NEW_DATABASE: String: 데이터베이스 이름 NEW_DATABASE: int: 데이터베이스 버전
Methods
ParkingAreaDatabaseHelper(): SQLite 데이터베이스 접근 initializeParkingSpacesFromOldDatabase(newDB:SQLiteDatabase): void: 주차 구역 정보와 이중 주차 구역 정보 가져오기

6) (Double)Parking Registration

Attributes
databaseHelper: ParkingAreaDatabaseHelper: 데이터베이스 접근 RegionNumInput: EditText: 일반 주차 구역 이름 RegionInput: EditText: 일반 주차 가능 구역 번호 aptId: long: 아파트 id phoneNumber: String: 사용자 전화번호
Methods
deleteParkingSpaceInfo(aptId:int,region:String,regionNum:String): void: 사용자가 등록한 아파트 구역를 데이터 베이스에서 삭제하기 isParkingSpace(aptId:int,region:String,regionNum:String): boolean: 사용자가 입력한 주차 구역 정보가 데이터베이스에 있는지 확인하기 isUserTableInfo(aptId:int,region:String,regionNum:String,phoneNumber:String): boolean: 사용자가 입력한 주차 구역이 이미 등록되어 있는지 확인하기 insertParkingUserInfo(aptId:int,region:String,regionNum:String,phoneNumber:String): void: 사용자의 정보를 등록한 아파트 구역과 함께 데이터베이스에 저장하기

7) (Double)Parking Out

Attributes
databaseHelper: ParkingAreaDatabaseHelper: 데이터베이스 접근 aptId: long: 아파트 id phoneNumber: String: 사용자 전화번호 Region: String: 주차 구역 이름 RegionNum: String: 주차 구역 번호
Methods
deleteParkingUserInfo(aptId:int,region:String,regionNum:String,phoneNumber:String): void: 취소할 주차 구역 정보와 사용자 정보를 데이터베이스에서 삭제 insertParkingSpaceInfo(aptId:int,region:String,regionNum:String): void: 취소된 주차 구역 정보를 데이터 베이스에 추가

8) (Double)Parking Check

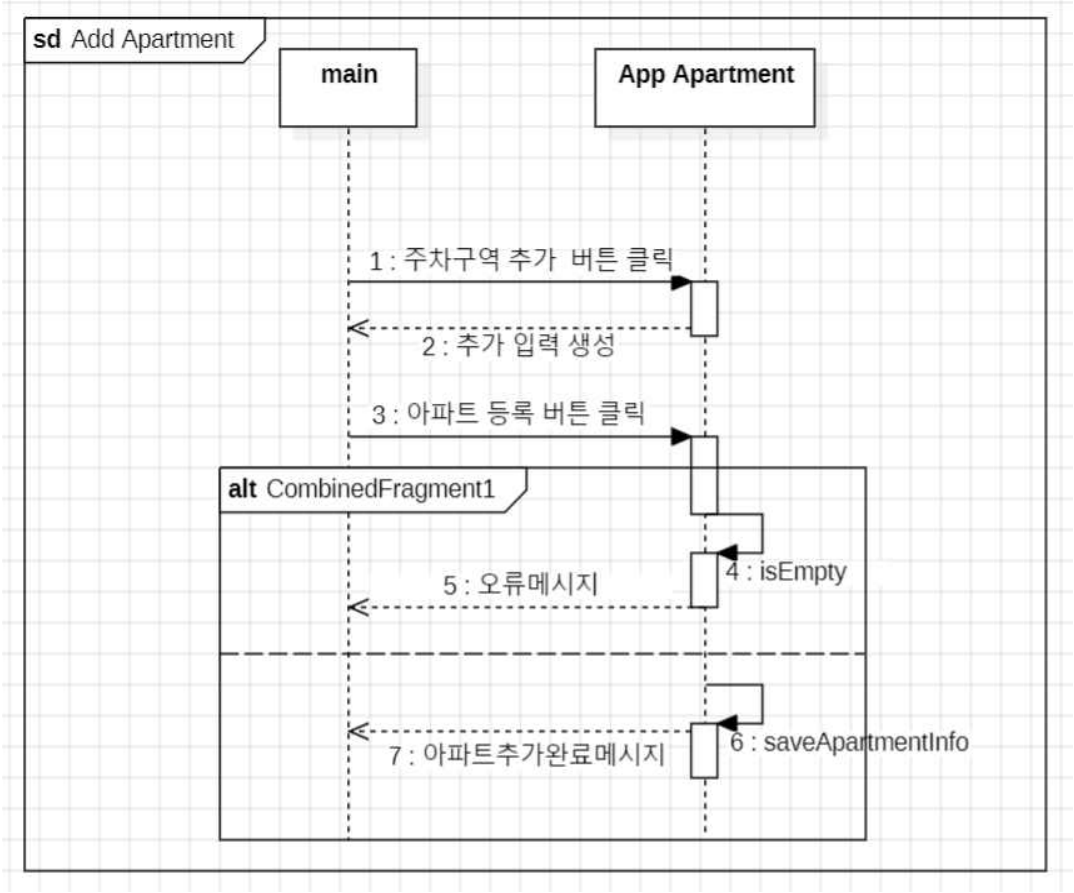
Attributes
adapter: ArrayAdapter<String>: ListView에 데이터를 표시하기 위한 어댑터 parkingDataList: ArrayList<String>: 주차 정보를 저장하는 ArrayList aptId: long: 아파트 id
Methods
displayParkingInfo(int): void: 주차 정보를 조회하고 표시

9) ViewUser(Double)Parking

Attributes
dpHelper: ParkingAreaDatabaseHelper: 데이터 베이스 (사용자 테이블) 접근
Methods
onCreate(Bundle): void: 아파트 id에 해당하는 주차 사용자 정보 조회하여 텍스트뷰에 표시

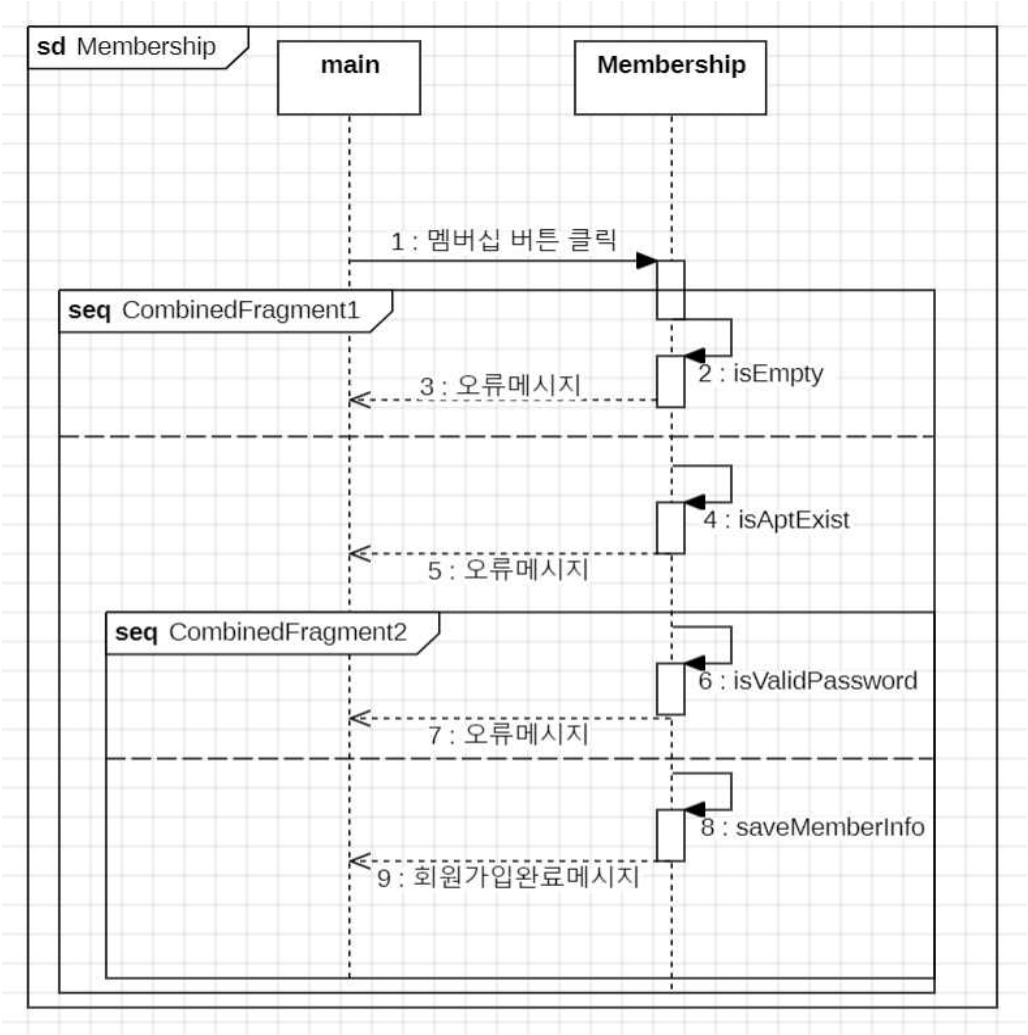
3. Sequence diagram

1) Add Apartment



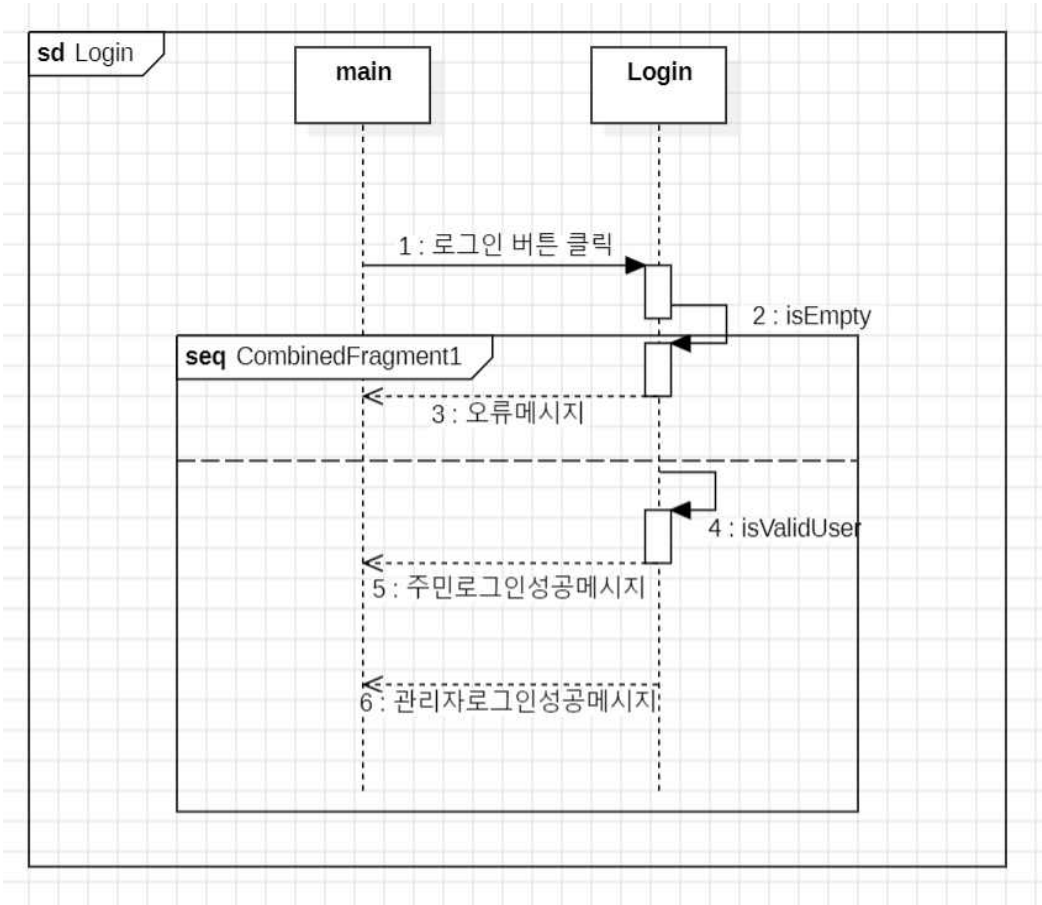
위 그림은 시스템을 실행한 후 아파트를 등록하는 기능을 수행할 때를 표현한 Sequence Diagram이다.
메인 화면에서 아파트 등록 버튼을 선택하여 아파트를 등록하는 화면으로 이동한다.
등록할 아파트에 대해 입력한 후 등록하기 버튼을 누르면 아파트의 정보가 데이터 베이스에 저장되고 완료되었다는 메시지가 뜬다.
입력란에 아무것도 입력하지 않고 등록하기 버튼을 누르면 오류메시지가 뜬다.

2) Membership



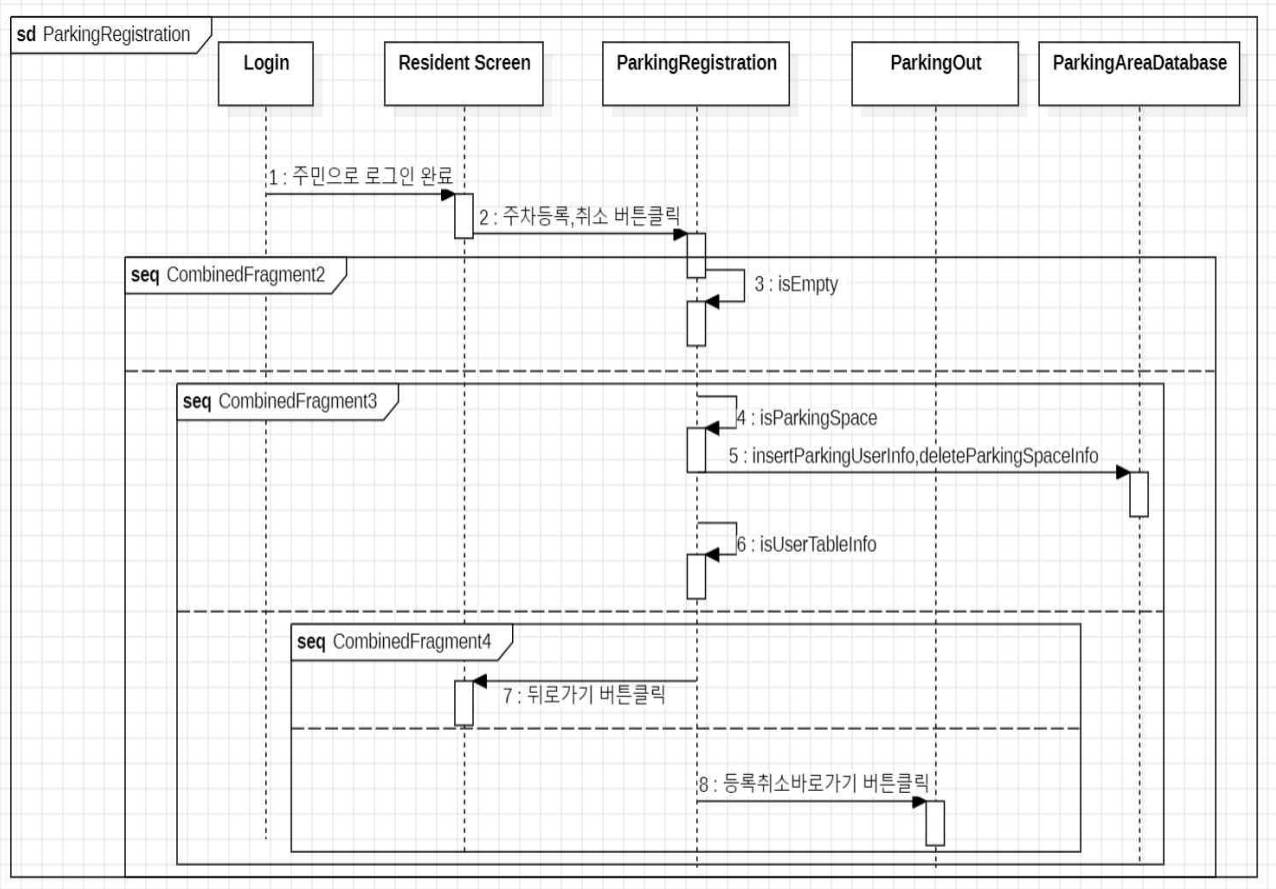
위 그림은 시스템을 실행한 후 회원가입을 하는 기능을 수행할 때를 표현한 Sequence Diagram이다. 메인 화면에서 멤버십 버튼을 클릭하면 멤버십 화면으로 이동하게 된다. 사용자는 자신의 정보와 아파트 이름을 작성하고 가입하기 버튼을 누른다.
회원가입이 완료되면 회원의 정보를 데이터 베이스에 저장하고 회원가입 완료 메시지를 출력한다.
하지만 입력란에 아무 정보를 작성하지 않은 경우, 아파트가 존재하지 않는 경우, 비밀번호가 일치하지 않는 경우는 상황에 맞는 오류 메시지를 출력한다.

3) Login



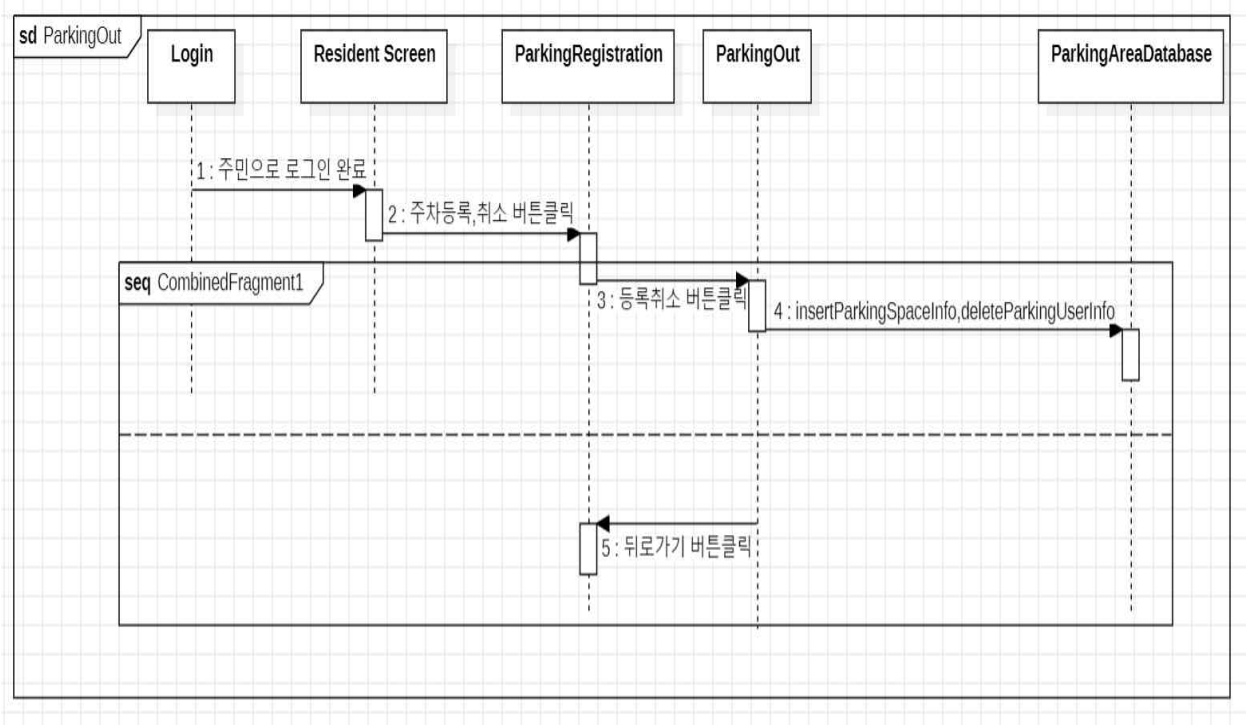
위 그림은 시스템을 실행한 후 로그인을 하는 기능을 수행할 때를 표현한 Sequence Diagram이다. 메인 화면에서 로그인 버튼을 클릭하면 로그인 화면으로 이동하게 된다. 사용자는 자신의 정보를 입력하고 로그인 버튼을 누른다. 입력된 사용자의 정보가 데이터 베이스에 있는지 확인한 후 로그인이 성공하면 성공하였다는 메시지가 뜨고 관리자로 로그인이 성공하면 관리자 화면으로, 주민으로 로그인을 성공하면 주민 화면으로 이동하게된다. 입력란에 아무것도 입력하지 않으면 오류메시지가 뜬다.

4）(Double)ParkingRegistration



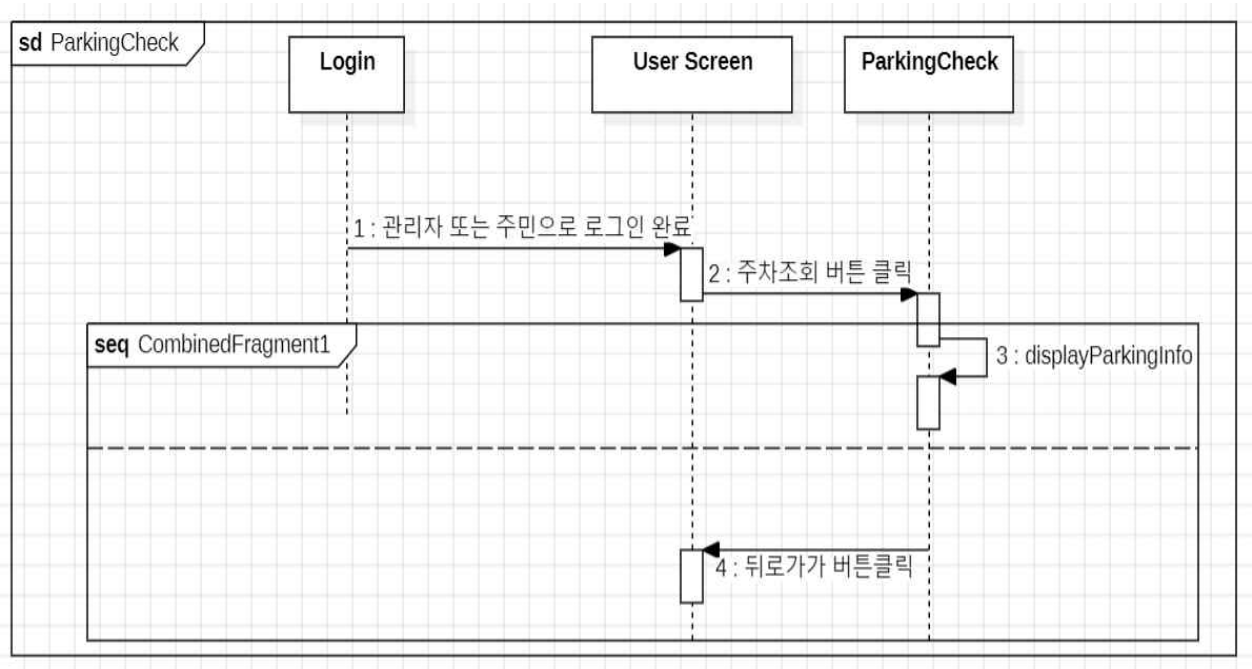
위 그림은 시스템을 실행한 후 주차를 등록할 때 수행하는 기능을 표현한 Sequence Diagram이다. 주민으로 로그인이 완료되면 주민 화면으로 이동하게 된다. 주민 화면에서 주차 등록을 누르면 주차 등록 화면으로 이동한다. 입력란에 주차할 구역의 정보를 입력하고 등록하기 버튼을 누르면 주차할 공간이 있는지 확인한 후 데이터 베이스 안에 주차 공간 테이블에서는 정보를 삭제하고 주차한 사용자정보 테이블에는 정보를 추가한다. 이후 주차가 등록되었다는 메시지를 출력한다.
뒤로가기 버튼을 누르면 주민 화면으로 되돌아가고, 등록취소 바로가기 버튼을 누르면 주차 취소 화면으로 이동하게 된다. 이중 주차 등록도 같은 방법으로 실행된다.

5）(Double)ParkingOut



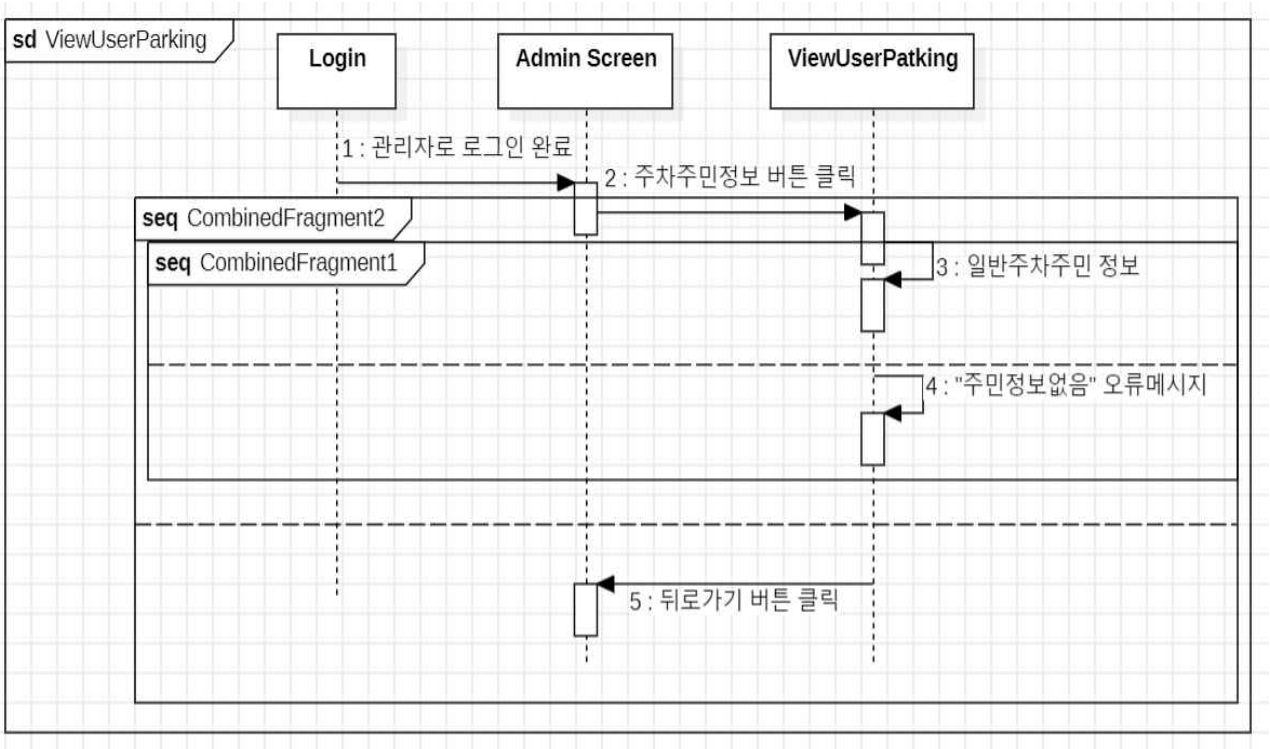
위 그림은 시스템을 실행한 후 주차를 취소할 때 수행하는 기능을 표현한 Sequence Diagram이다. 주민으로 로그인을 완료한 후 주민 화면에서 주차 등록/취소 버튼을 누르면 주차 등록 화면으로 이동하게 되고 등록 하면에서 등록 취소 버튼을 누르면 취소 화면으로 이동한다.
취소 화면에서 취소하기를 누르면 데이터 베이스에 있는 주차 정보 테이블에 주차 구역이 추가되고, 사용자 정보 테이블에서는 삭제된다. 뒤로가기 버튼을 누르면 주차 등록 화면으로 이동하게 된다.
이중 주차 취소도 같은 방법으로 실행된다.

6）(Double)ParkingCheck



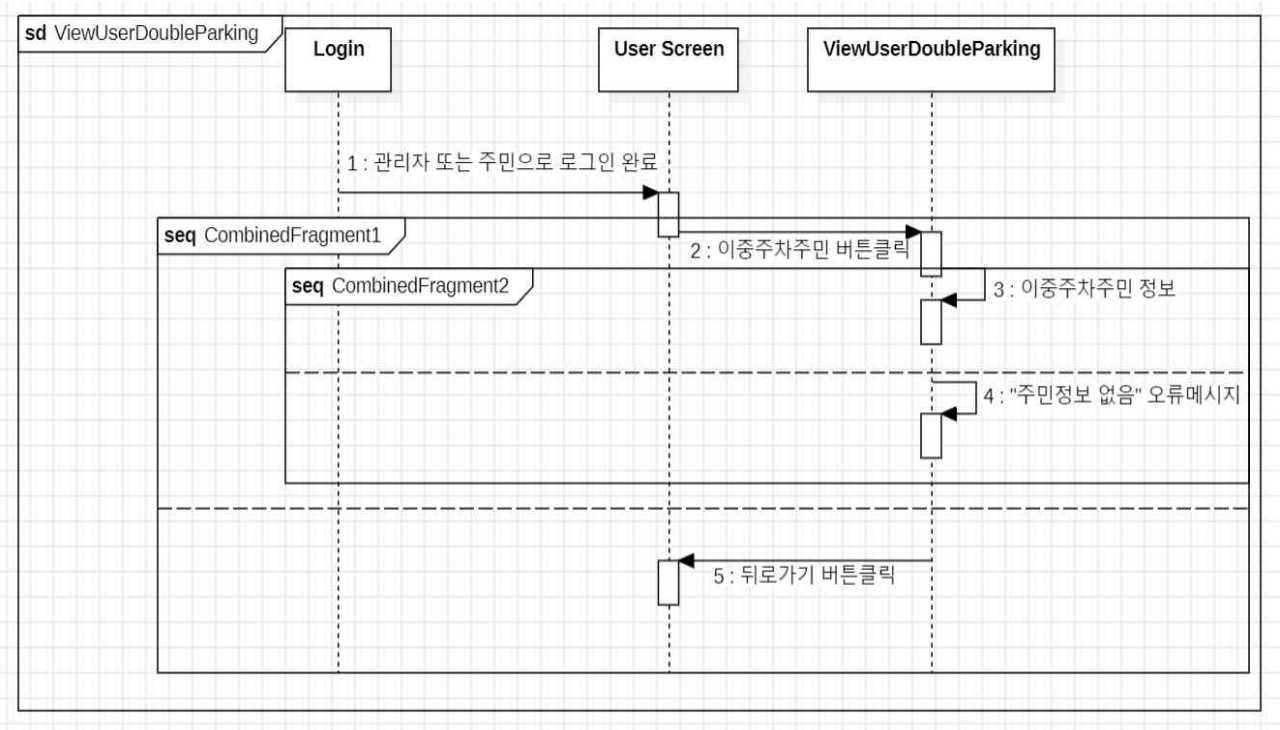
위 그림은 시스템을 실행한 후 주차 구역 정보를 조회할 때 수행하는 기능을 표현한 Sequence Diagram이다. 관리자 또는 주민으로 로그인이 성공하면 각자 화면으로 이동하게 된다. 그 화면에서 주차 조회 버튼을 클릭하면 주차 조회 화면으로 이동하고 데이터 베이스의 주차 공간 정보 테이블에서 정보를 가져와 화면에 출력한다.
뒤로 가기 버튼을 누르면 다시 주차 등록 화면으로 이동한다.
이중 주차 조회도 같은 방법으로 실행된다.

7) ViewUserParking



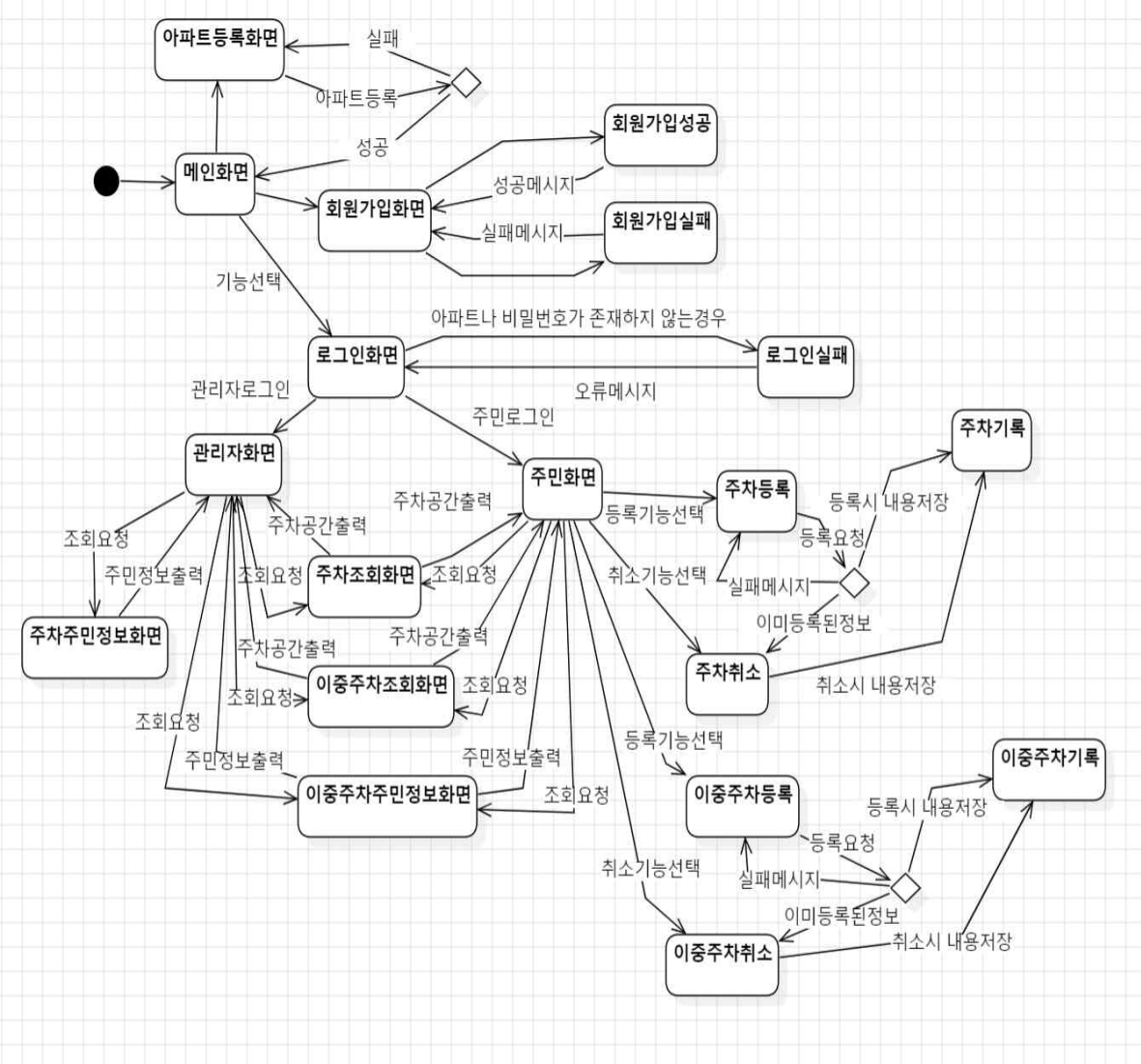
위 그림은 시스템을 실행한 후 일반 주차 구역 사용자 정보를 조회할 때 수행하는 기능을 표현한 Sequence Diagram이다. 관리자로 로그인이 완료되면 관리자 화면으로 이동하고 관리자 화면에서 주차 주민 정보 버튼을 클릭한다. 클릭하면 주차 주민 정보 조회 화면으로 이동하게 되고 일반 주차 주민 정보를 데이터 베이스에서 가져와 화면에 출력한다.
등록된 주민 정보가 없으면 오류 메시지를 출력한다.
뒤로 가기 버튼을 누르면 다시 관리자 화면으로 이동하게 된다.

8) ViewUserDoubleParking



위 그림은 시스템을 실행한 후 이중 주차 구역 사용자 정보를 조회할 때 수행하는 기능을 표현한 Sequence Diagram이다. 관리자 또는 주민으로 로그인이 완료되면 관리자 또는 주민 화면으로 이동하고 각 신분에 맞는 화면에서 주차 주민 정보 버튼을 클릭한다. 클릭하면 주차 주민 정보 조회 화면으로 이동하게 되고 이중 주차 주민 정보를 데이터 베이스에서 가져와 화면에 출력한다.
등록된 주민 정보가 없으면 오류 메시지를 출력한다.
뒤로 가기 버튼을 누르면 다시 관리자 또는 주민 화면으로 이동하게 된다.

4. State machine diagram



위는 앱의 State machine diagram으로 시스템의 각 상태들과 상태 간의 전의를 시각적으로 표현한다.
회원가입을 한 뒤는 사용자의 권한에 따라 사용할 수 있는 기능이 나뉘어 진다.
관리자로 로그인을 하면 주차 조회, 이중 주차 조회, 일반 또는 이중 주차 주민 정보 조회를 할 수 있다.
주민으로 로그인을 하면 일반 또는 이중 주차 등록과 취소, 일반 또는 이중 주차 조회, 이중 주차 주민 정보 조회를 할 수 있다.

5. Implementation requirements

1) HardWare platform requirements

Device: Android 기반의 스마트 기기 (스마트폰, 테블릿 등)

2) SoftWare platform requirements

OS : android 14.0

API: 34

SDK : 24~34

6. Glossary

state machine diagram	시스템이나 객체의 여러 상태와 상태 간 전이를 시각적으로 나타낸 다이어그램이다. 이 다이어그램은 시스템의 동작을 이해하고 설계하는 데 사용된다. 주로 상태 기계 다이어그램은 상태(state), 이벤트(event), 전이(transition)로 구성된다.
sequence diagram	객체 간 상호 작용을 시간의 흐름에 따라 나타내는 UML(Unified Modeling Language) 다이어그램의 한 종류이다. 주로 객체 간의 메시지 전달과 호출 순서를 시각적으로 보여준다.
class diagram	UML(Unified Modeling Language) 다이어그램의 하나로 시스템의 구조를 보여주는 데 사용된다. 주로 클래스, 인터페이스, 관계, 속성 등을 시각적으로 나타낸다.

7. References

https://valuefactory.tistory.com/981

https://deep-learning-challenge.tistory.com/75?category=997371