# 3. Design


**MMD(My mommy's Drivers)**

- 21912050 박진규
- jinkyu57505750@gmail.com

# [ Revision history ]

| Revision date | Version # | Description | Author |
|---|---|---|---|
| MM/DD/YYYY | 0.00 | Type brief description here | Author name |
| 06/02 2023 | 0.01 | First Document | 박진규 |
| | | | |
| | | | |
| | | | |
| | | | |

= Contents =

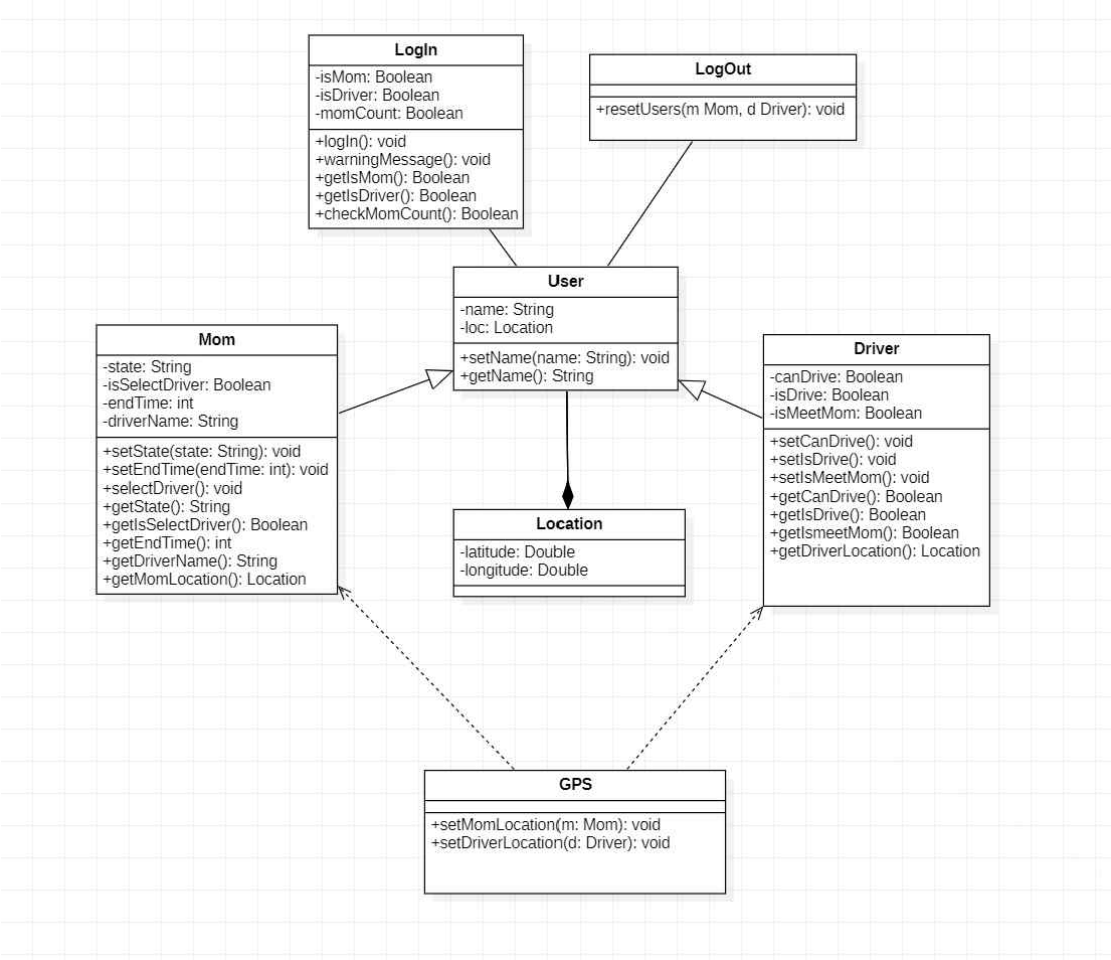
# 1. Introduction

본 문서는 소프트웨어 설계의 Design단계의 관한 문서이다. 현재 진행하고있는 MMD(My Mommy's Driver) 프로젝트의 Conceptualization , Anaysis 단계를 지나 세 번째 단계로, 진행중인 시스템 속 포함되어있는 클래스와 클래스들 서로의 연관관계, 상속관계를 나타내고 클래스들의 속성과 메소드 등 사용되는 클래스들의 전반적인 정보를 시각화하여 보여주는 Class Diagram을 작성한다. 이를 통해 시스템에서 사용되는 클래스들을 이해할 수 있으며 시스템의 전반적인 흐름을 이해할 수 있을 것이다.

이번 문서에서 작성할 두 번째 내용으로는 Sequence Diagram이다. 이름에서 알 수 있듯이 Sequence, 즉 순서를 보여주는 다이어그램이다. 객체와 객체들의 어떤 순서로 상호작용을 하는지 나타내는 다이어그램으로, 시스템이 어떤 흐름으로 동작하는지를 알 수 있다. 지금까지 작성해두었던 유즈케이스를 이 시퀀스 다이어그램으로 표현할 것이며, 이를 통해 각 유즈케이스의 흐름을 파악할 수 있어 내가 만들고자 하는 프로그램의 실질적인 동작 흐름을 이해할 수 있을 것이다.

마지막으로는 State Machien Diagram을 작성할 것이다. 이 다이어그램을 한글로 번역하면 상태 머신 다이어그램이다. 주요 키워드로 보이는 것이 상태이다. 즉, 객체의 상태를 보이는 다이어그램이다. 각 객체들은 어떠한 특정 상태를 갖게되며, 이 상태는 어떠한 이벤트들로 변화할 수 있는 것이다. 요약하자면, 프로그램을 실행하고 각 이벤트 별로 객체들이 어떠한 상태로 변화하게되는지를 시각화하는 다이어그램이다.

2. Class diagram




-Class Diagram

– LogIn

| Attributes |
| --- |
| - isMom: Boolean : 사용자가 Mom인지 나타내는 변수<br>- isDriver: Boolean : 사용자가 Driver인지 나타내는 변수<br>- momCount : Boolean : Mom의 수 |
| Methods |
| + logIn(): void : 로그인<br>+ warningMessage(): void : Mom이 이미 선택되어 있는 상태에서 유저가 Mom을 선택했을시 보내는 경고메시지<br>+ getIsMom(): Boolean : isMom 반환<br>+ getIsDriver(): Boolean : isDriver 반환<br>+ checkMomCount(): Boolean : momCount 반환 |

첫 화면에서 Mom과 Driver 두 버튼 중 하나를 선택을하면 logIn메소드가 실행이 되며 선택한 역할의 변수가 True로 바뀌게 된다.

– LogOut

| Attributes |
| --- |
| |
| Methods |
| -resetUsers(m: Mom, d: Driver): void : 모든 과정이 끝나고 유저들의 상태를 초기화하는 메소드 |

모든 과정이 끝나고 유저들의 상태를 초기화 후 첫 화면으로 돌아가게 된다.

– Location

| Attributes |
| --- |
| -latitude: Double : 위도<br>- longitude: Double : 경도 |
| Methods |
| |

유저가 가지게 될 위치정보

-User

| Attributes |
| --- |
| -name: String : 유저의 이름<br>-loc: Location : 유저의 위치정보 |
| Methods |
| -setName(name: String): void : 유저 이름 저장<br>-getName(): String : 유저 이름 반환 |

유저의 이름과 위치정보를 저장하는 클래스

-Mom

| Attributes |
| --- |
| -state: String : Mom이 Driver에게 알릴 특이사항<br>-isSelectDriver: Boolean : Mom이 Driver을 선택한 상태인지의 대한 변수<br>-endTime: int : Mom의 퇴근시간<br>-drvierName: String : Mom이 선택한 Driver의 이름 |
| Methods |
| +setState(state: String): void : state를 저장<br>+setEndTime(endTime: int): void : endTime을 저장<br>+selectDriver(); void : driver을 선택<br>+getState(): String : state를 반환<br>+getEndTime(): int : endTime을 반환 |

| |
|---|
| +getIsDriver(): Boolean : isDriver을 반환<br>+getDriverName(): String : driverName을 반환<br>+getMomLocation(): Location : Location을 반환 |

Mom의 기본정보를 저장하는 클래스이며 User클래스를 상속받는다

-driver

| Attributes |
|---|
| -canDrive: Boolean : Driver가 그 날 운행 가능한지 설정하는 변수<br>-isDrive: Boolean : Driver가 운행중인지 나타내는 변수<br>-isMeetMom: Boolean : Driver가 Mom을 만났는지 나타내는 변수 |
| **Methods** |
| +setCanDrive(): void : canDrive를 설정<br>+setIsDrive(): void : isDrive를 설정<br>+setIsMeetMom(): void : isMeetMom을 설정<br>+getCanDrive(): Boolean : canDrive를 반환<br>+getIsDrive(): Boolean : isDrive를 반환<br>+getIsMeetMom(): Boolean : isMeetMom을 반환<br>+getDriverLocation(): Location : Location을 반환 |

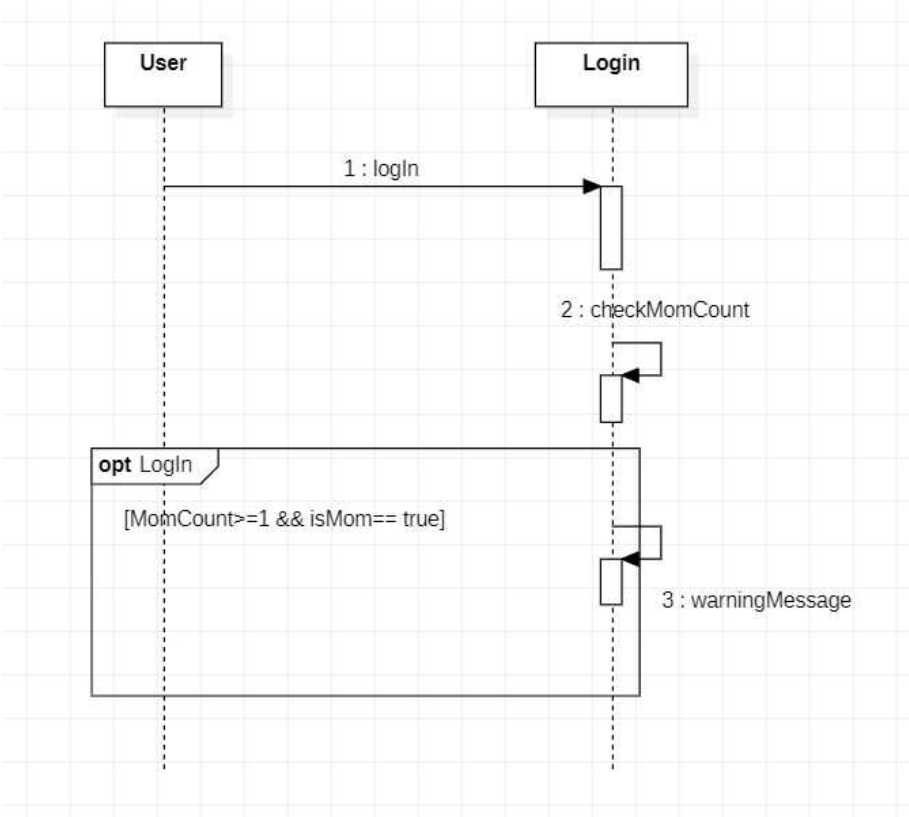
Driver의 기본정보를 저장하는 클래스이며 User클래스를 상속받는다

-GPS

| Attributes |
|---|
| |
| **Methods** |
| +setMomLocation(m: Mom): void : Mom의 위치정보를 저장<br>+setDriverLocation(d: Driver): void : Driver의 위치정보를 저장 |

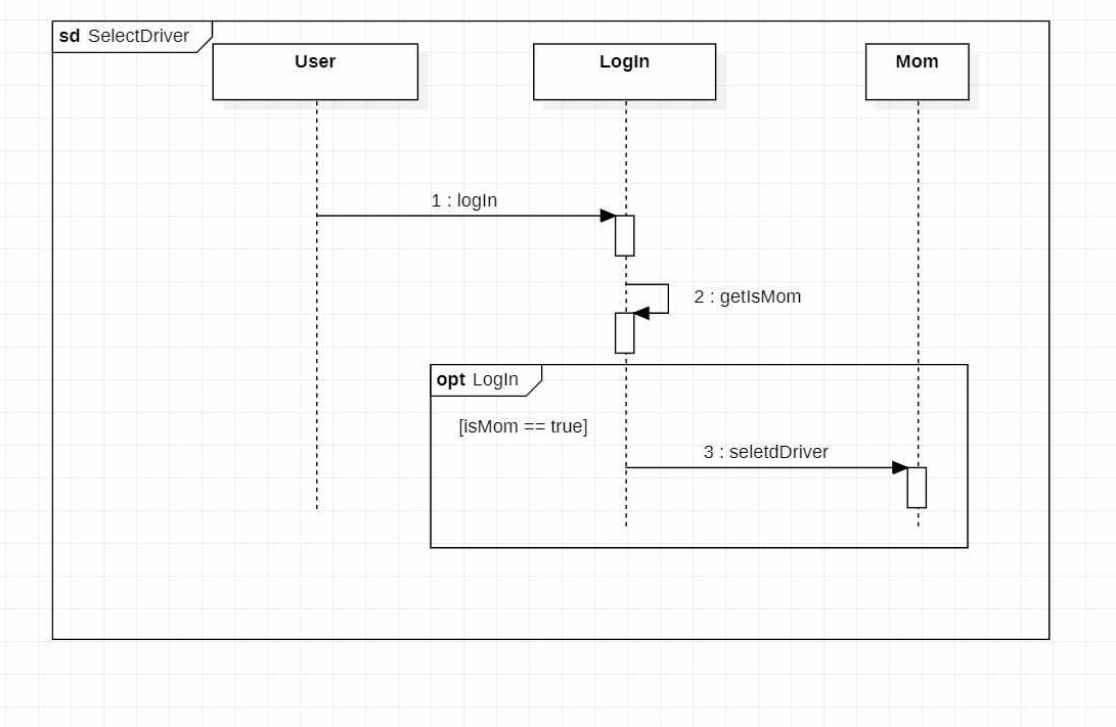
유저의 위치정보를 받아 저장하는 클래스

3. Sequence diagram



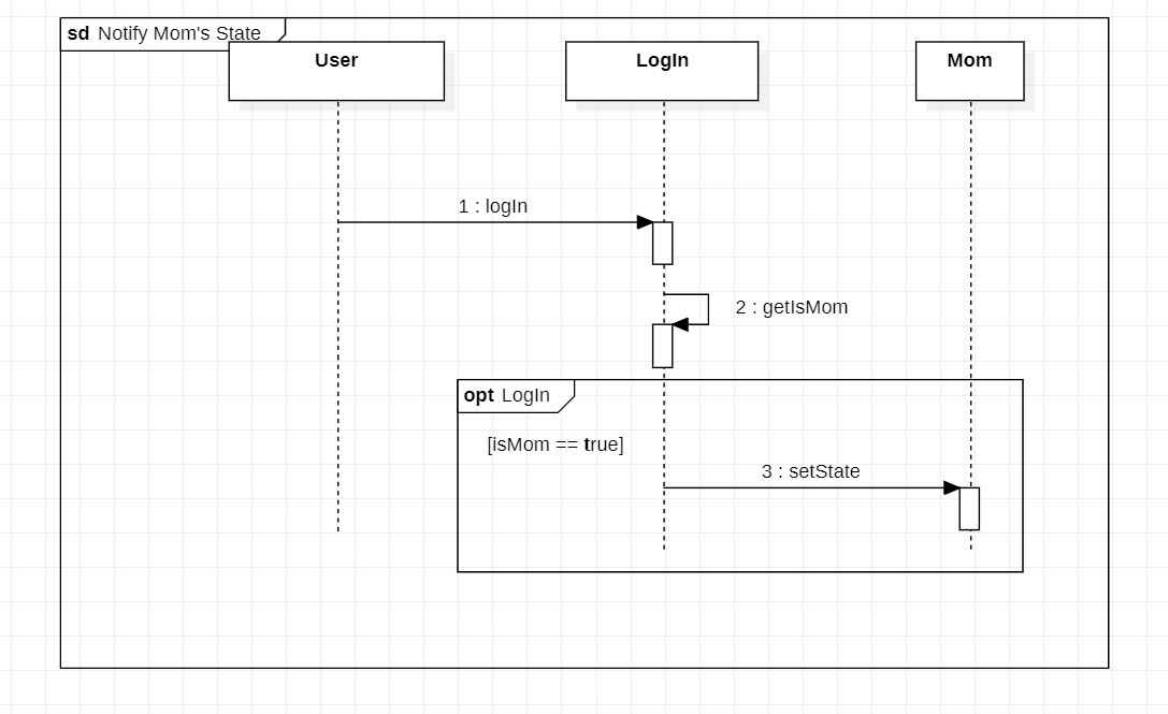

Use Case : Log In

- 로그인클래스의 로그인 메소드를 실행하고, Mom으로 선택한 유저가 있을 수 있으니 checkMomCount메소드를 통해 Mom을 선택한 유저가 있는지 확인 후, 만약 있다면 경고 메시지를 출력한다.



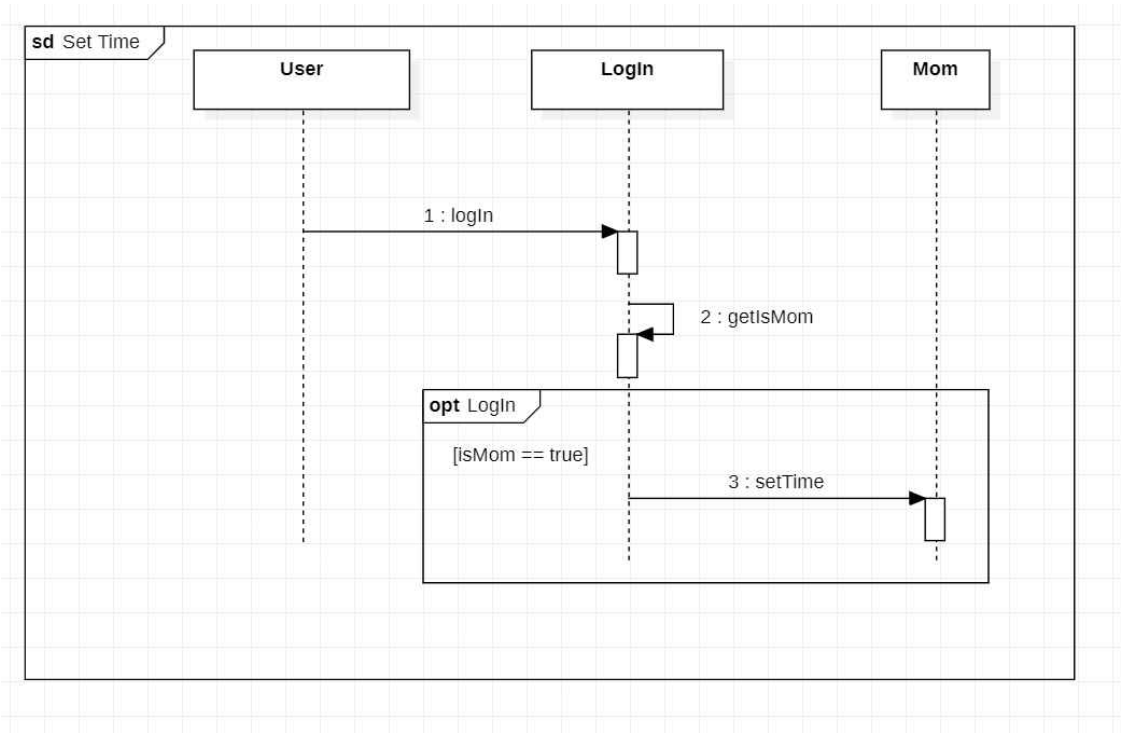

Use Case :SelectDriver

– 로그인 후, getIsMom을 통해 Mom임을 확인하고 selectDriver메소드를 통해 현재 가용한 Driver을 선택하게 된다.



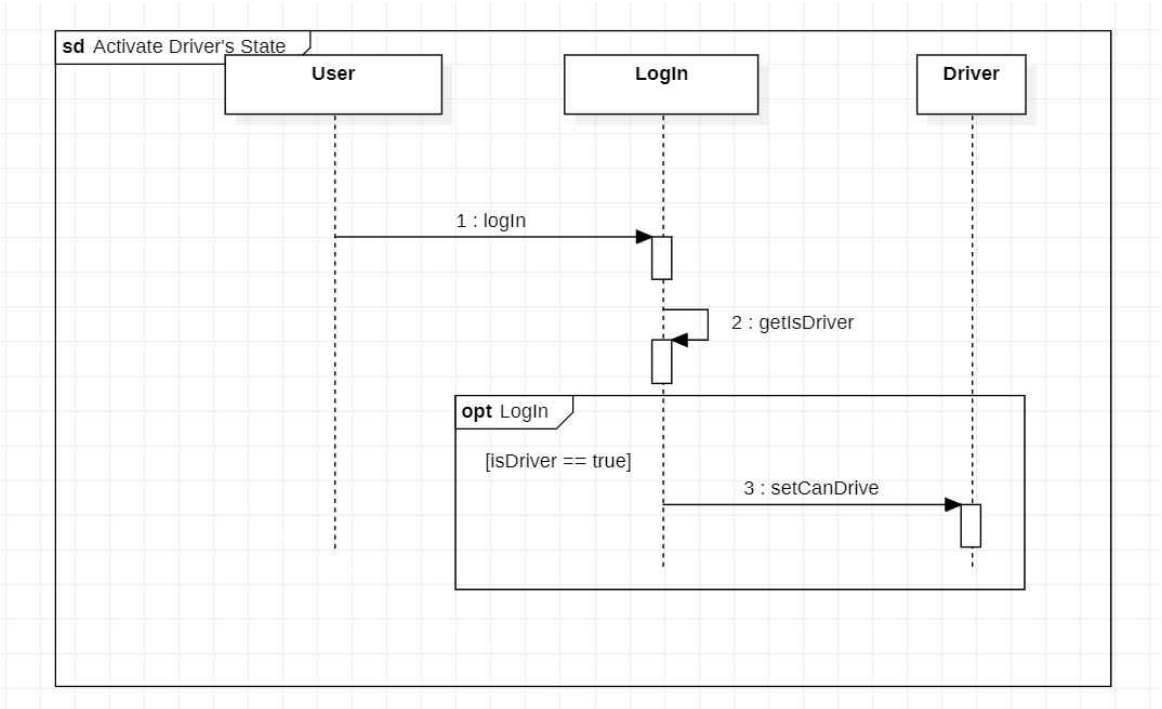

Use Case: Notify Mom's State

- Use Case: Select Driver 와 같은 구조



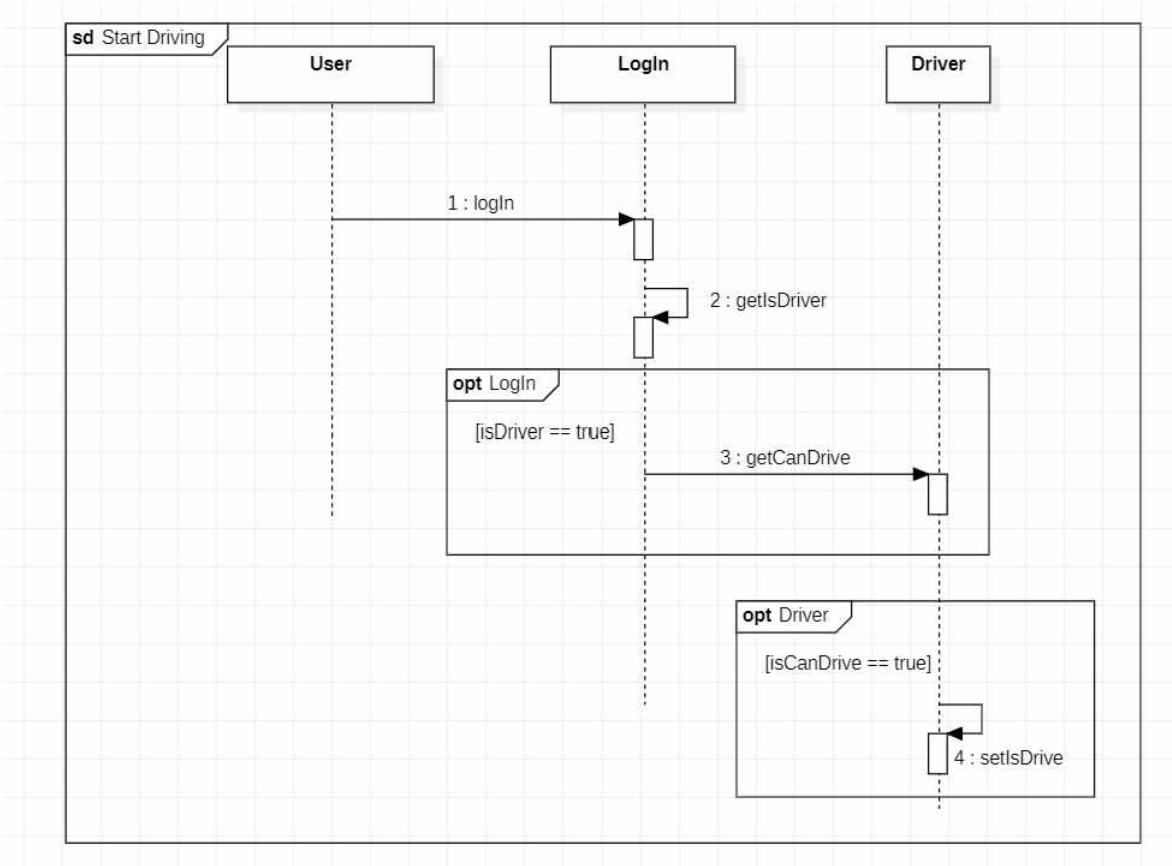

Use Case; Set Time

- Use Case: Select Driver 와 같은 구조



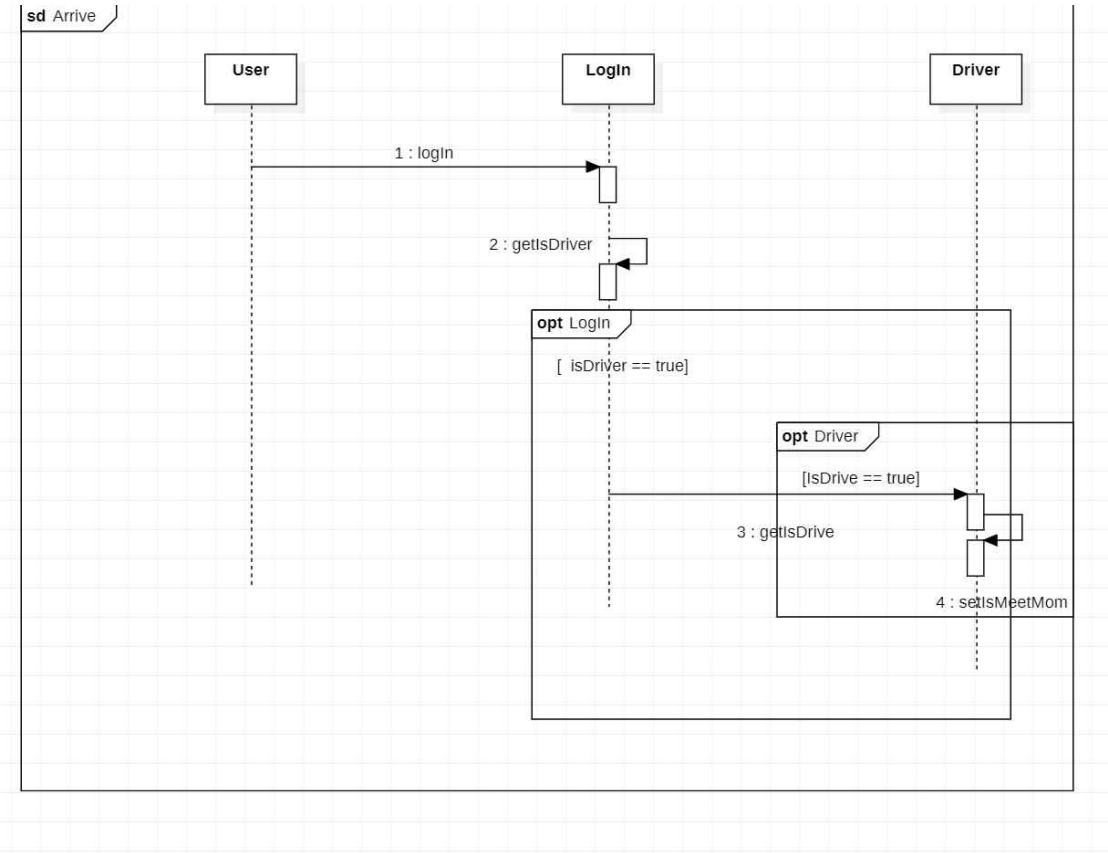

Use Case: Activate Driver's State

– 로그인 후, getIsDriver을 통해 Driver임을 확인하고 setCanDrive메소드를 통해 본인이 현재 Driving이 가능한지를 설정



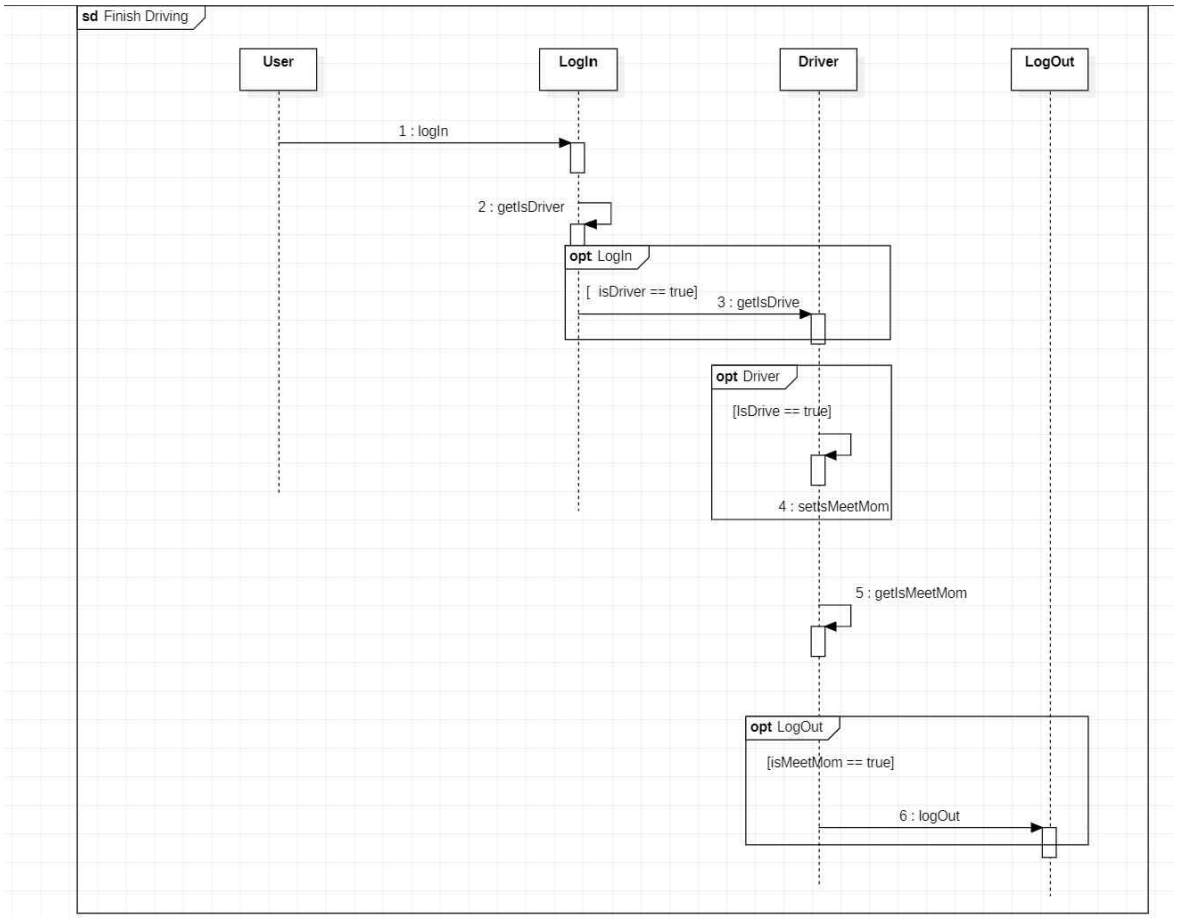

Use Case : Start Driving

- getCanDrive메소드로 Driver가 Driving가능한지를 얻어 온 후 값이 참이라면 setIsDrive메소드를 통해 Driving을 시작한다.




Use Case : Arrive

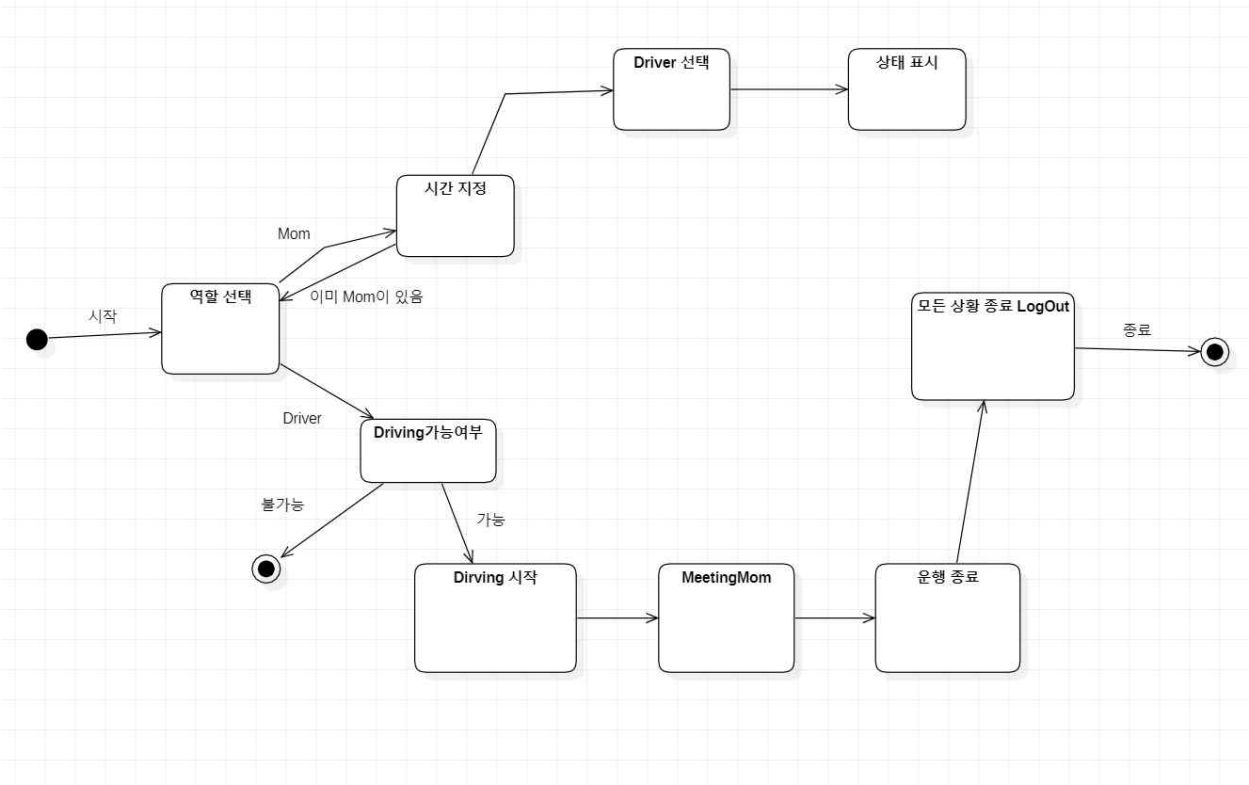
- getIsDrive메소드를 통해 현재 Driving상태중인지를 확인한 후 Mom의 근무지에 도착했을 시 setIsMeetMom메소드를 통해 isMeetMom를 true로 전환한다.




Use Case: Finish Driving

- setIsMeetMom을 통해 Mom의 근무지까지 도착한 후 , 귀가를 하면 getIsMeetMom을 통해 Mom을 만났는지의 대해 확인 후, LogOut클래스의 logOut을 진행하게 된다. 이를 통해 모든 유저들의 정보들이 초기화되며 첫 화면으로 돌아가게 된다.

4. State machine diagram




시스템을 시작하면 역할을 선택하게된다.

Mom을 선택할 시 퇴근시간을 지정할 수 있으며, 이미 Mom이 있는 경우에는 역할선택하는 화면으로 돌아가게 된다.(Driver가 잘못 선택한 경우) 퇴근시간을 지정한 경우 가용한 Driver을 선택하게 되고, Mom의 특이사항을 기재할 수 있다.

Driver을 선택한 경우 Driving가능여부를 선택할 수 있다. 불가능한 경우에는 앞으로 진행하지 못하고 종료한다.(Mom에게는 불가능한 Driver은 회색으로 처리됨.) Driving이 가능한 Driver은 Mom이 지정한 시간에 맞추어 Driving을 시작하게 되고 Mom을 만난 후, 귀가를 하면 운행을 종료하게 된다. 이후 모든 상황이 종료됨에 따라 LogOut으로 모든 유저들을 초기화 후 시스템이 종료된다.

## 5. Implementation requirements

구현 툴: 안드로이드 스튜디오
구현언어 : JAVA

## 6. Glossary

- Class Diagram : 객체지향 프로그래밍에서 소프트웨어 시스템의 구조와 클래스 간의 관계를 시각적으로 표현하는 다이어그램
- Attribute : 객체의 데이터 또는, 속성
- Opeartion : 객체의 특별한 동작이나 기능 (메소드/함수)
- Sequence Diagram : 시간 순서에 따라 객체 간의 상호작용을 시각적으로 표현하는 다이어그램
- State Machine Diagram :객체의 상태 변화를 시각적으로 표현하는 다이어그램

## 7. References

- 소프트웨어 설계 강의자료