3. Design

RIDE MAP

STUDENT NO.	22112353
NAME	김동근
EMAIL	ehdrms3535@naver.com

[Revision history]

Revision date	Version #	Description	Author
MM/DD/YYYY	0.00	Type brief description here	Author name
05/06/2025	0.01	전반적인 내용구성	김동근

= Contents =

1. Introduction

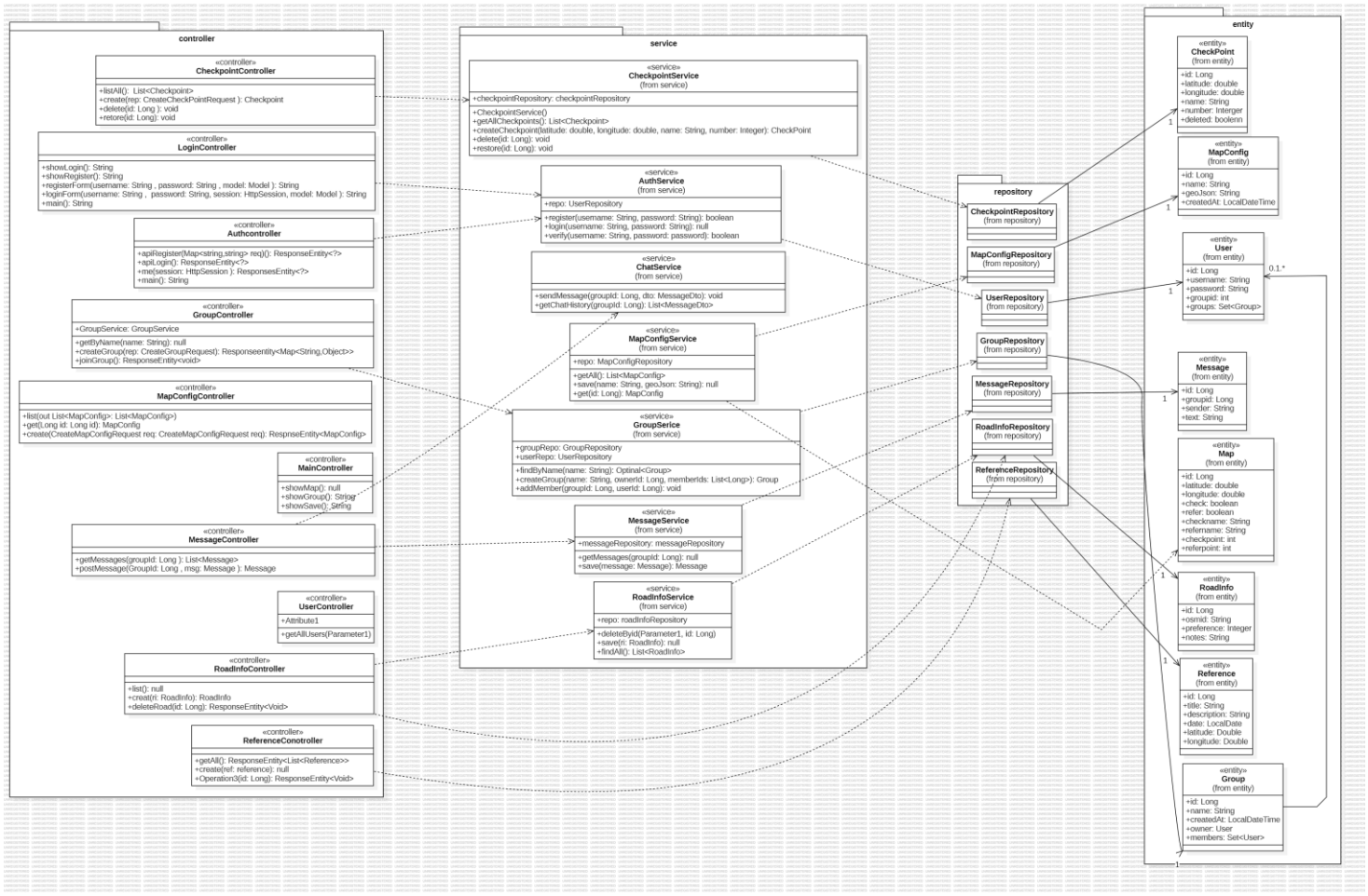
최근 자전거, 전동 킥보드, 스쿠터, 오토바이 등 **비자동차형 이동 수단**의 활용이 점점 증가하고 있습니다. 이들은 가벼운 여행이나 일상 속 산책, 출퇴근 등 다양한 상황에서 유용하게 쓰이고 있지만, **도로 상태나 갈 수 있는 경로, 가볼 만한 장소에 대한 정보가 부족해 불편함**을 겪는 경우가 많습니다.

이러한 문제를 해결하기 위해, 사용자들이 **도로 환경, 여행 루트, 주요 랜드마크, 숨겨진 명소 등**의 정보를 서로 공유하고 활용할 수 있다면, 이동의 효율성과 즐거움 모두 크게 향상될 것입니다

이번 문서는 Analysis 다음 단계인 Design 단계로 실제로 사용하는 기능들을 구체화하고 개발 또는 그 이후에 사용할 내용들을 더 작은 부분들까지 구체화를 하는 단계의 내용(아래의 내용)들이 작성된다.

1. 기능 분해 및 책임 할당
    - Use Case 상세화: 각 핵심 기능(회원 관리, 경로 저장/조회, QR 코드 생성/로드, 그룹 공유 등)에 대해 시나리오별 순서도/시퀀스 다이어그램 작성
    - 컴포넌트 모듈화: 최대한 low-coupling/high-cohesion을 유지하도록 서비스, 컨트롤러, 리포지토리, 유틸리티 모듈로 분리
2. 데이터베이스 설계
    - ERD(Entity-Relationship Diagram) 작성: User, Map, Route, CheckPoint, Reference, Group, QRCode 테이블 간 관계 정의
    - 인덱스/제약조건 설계: 외래키, 유니크 제약, 검색 성능을 위한 인덱스 후보 선정
3. API 인터페이스 정의
    - RESTful 엔드포인트 목록 작성 (HTTP 메서드, URL, 요청/응답 JSON 스펙)
    - 요청·응답 DTO 스펙(필드, 유효성 검증) 상세 설계
4. UI/UX 흐름 설계
    - 주요 화면(Screen) 설계: 로그인/회원가입, 지도 뷰, 경로 작성·저장, QR 코드 스캔·로드, 그룹 관리 등
5. 클래스·모듈 설계
    - 시퀀스 다이어그램: 사용자 요청 처리 흐름, 내부 서비스 호출, DB 접근까지 상세 플로우

2. Class diagram



1) CheckPoint

Attributes
+ id : Long 체크포인트 번호 + latitude : double x좌표 + longitude : double y좌표 + name : String 체크포인트 이름 + number : Integer 체크포인트 순서를 위한 번호 + deleted : Boolean 삭제 여부
Methods

2) MapConfig

attributes
+ id : Long 지도 번호 + name : String 지도 이름 + geoJson : String 지도 정보 + createdAt : LocalDateTime 생성 시간
Methods

3) User

attributes
+ id : Long 유저 번호 + username : String 유저 이름 + password : String 유저 비밀번호 + groupid : int 유저 그룹아이디 + groups : Set<Group> 그룹 내 집합
Methods

4) Message

attributes
+ id : Long 메시지 번호 + groupid : Long 그룹 아이디 + sender : String 전송자 아이디 + text : String 전송내용
Methods

5) Map

attributes
+ id : Long 지도 이름 + latitude : double x좌표 + longidude : double y좌표 + check : Boolean 체크포인트 유무 + refer : Boolean 선호도 유무 + cheackname : String 체크포인트 이름 + refername : String 선호 이름 + checkpoint : int 체크포인트 번호 + referpoint : int 선호도 번호
Methods

6) Reference

attributes
+ id : Ling 선호 도로 번호 + title : String 선호 도로 이름 + description : String 선호 이유 + data : LocalDate 지정 날자 + latitude : Double x좌표 + longitude : Double y좌표
Methods

7) RoadInfo

attributes
+id: Long 도로 정보 번호 +osmid: String osm 번호 + preference : integer 선호 도로 정도 + notes : String 각주
Methods

8) Group

attributes
+ id : Long 그룹 번호 + name : String 그룹 이름 + createdAt : LocalDateTime 생성 시간 + owner : User 그룹장 + members : Set<User> 그룹 내 집합
Methods

9) CheckpointContoller

attributes
Methods
+ listAll() : List<Checkpoint> 체크포인트 리스트
+ create(rep: CreateCheckPointRequest) : Checkpoint 체크포인트 생성
+ delete(id : Long) : void 체크포인트 삭제
+ restore(id : Long) : void 체크포인트 저장

10) LoginController

attributes
Methods
+showLogin() : String 로그인 페이지 호출
+ShowRegister():String 회원가입 페이지 호출
+registerForm(username:String,password:String,model:Model):String 회원가입 DB에 저장
+loginForm(username:String,password:String,session:HttpSession,model:Model):String DB에서 회원 정보 읽어와 회원 확인
+main(): String 메인 함수 호출

11) AuthController

attributes
Methods
+apiRegister(Map<string,string> req)(): ResponseEntity<?> 회원가입 API주소에서 엔티티 호출
+apiLogin(): ResponseEntity<?> 로그인 API주소에서 엔티티 호출
+me(in session:HttpSession): ResponsesEntity<?> 로그인한 객체 엔티티 생성
+main(): String 메인 함수 호출

12) GroupController

attributes
+ GroupService: GroupService 그룹 서비스 호출
Methods
+ getByName(in name:String) 그룹 이름 호출 + createGroup(in rep:CreateGroupRequest): Responseentity<Map<String,Object>> 그룹 생성 + joinGroup(): ResponseEntity<void> 그룹 참가

13) MapConfigController

attributes
Methods
+ list(out List<MapConfig>:List<MapConfig>) 지도 정보 리스트 호출 + get(in Long id:Long id): MapConfig 지도 정보 호출 + create(in CreateMapConfigRequest req:CreateMapConfigRequest req): RespnseEntity<MapConfig> 지도 정보 엔티티 생성

14) MainController

attributes
Methods
+ showMap() 지도 생성 페이지 이동 + showGroup(): String 그룹 페이지 이동 + showSave(): String 저장 페이지 이동

15) MessageController

attributes
Methods
+ getMessages(in groupId:Long): List<Message> 그룹내 메시지 수신
+ postMessage(in GroupId:Long , in msg:Message): Message 그룹내에 메시지 전송

16) UserController

attributes
Methods
+ getAllUsers(int Parameter1) 유저 번호 전부 호출

17) RoadInfoController

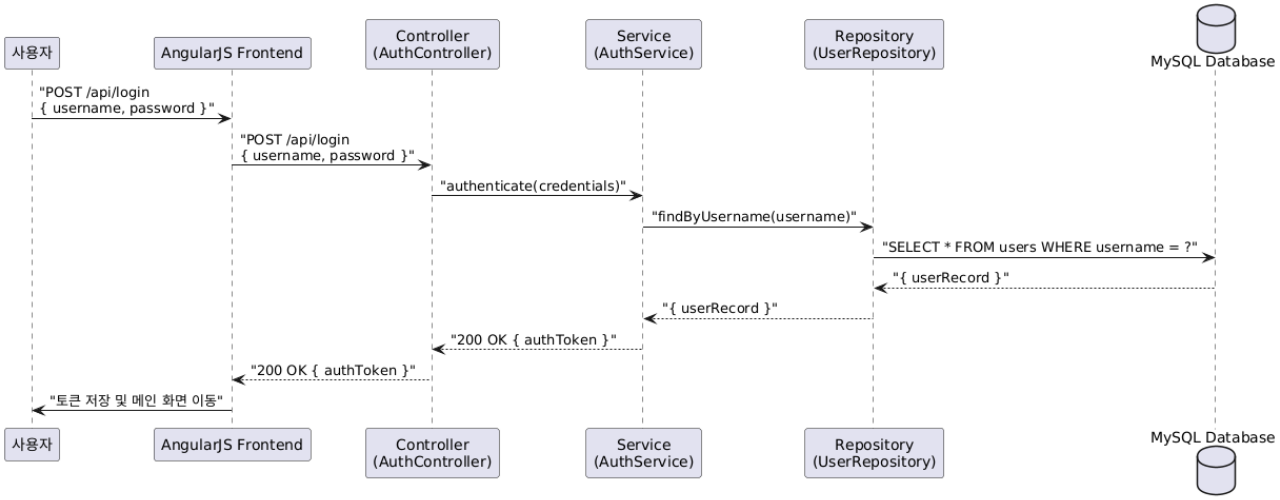
attributes
Methods
+ list():ResponseEntity<List<RoadInfo>> 도로 정보 전부 호출
+ creat(in ri:RoadInfo): RoadInfo 도로 정보 엔티티 생성
+ deleteRoad(in id:Long): ResponseEntity<Void> 도로 정보 엔티티 삭제

18) ReferenceController

attributes
Methods
+ getAll(): ResponseEntity<List<Reference>> 선호도 전부 호출 + create(in ref:reference) 선호도 생성 + Operation(id:Long): ResponseEntity<Void> 선호도 엔티티 작성

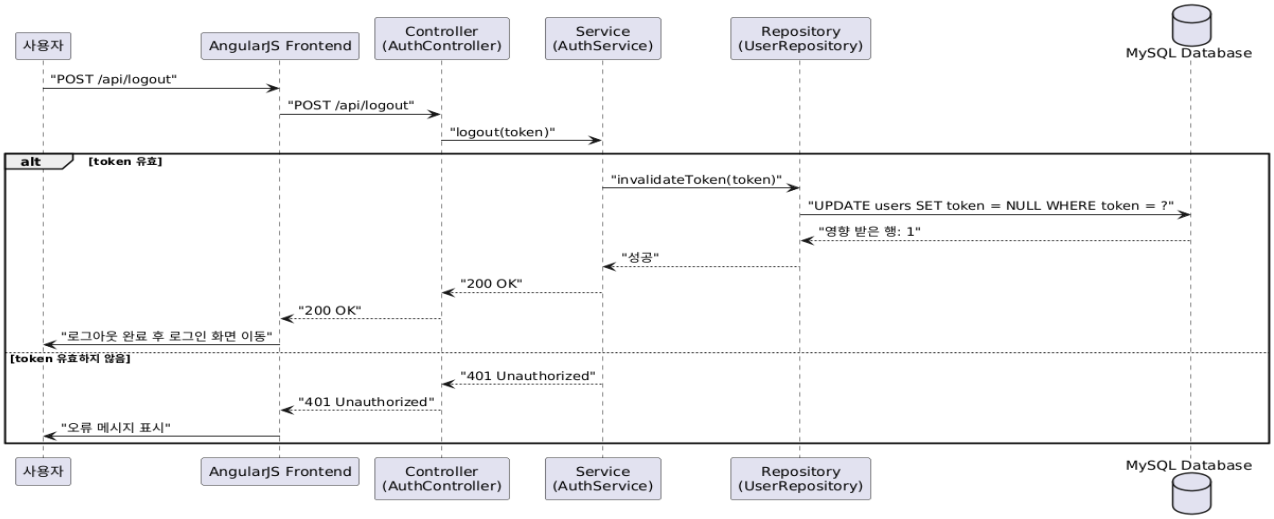
3. Sequence diagram

- login



사용자가 로그인 정보를 입력하고 로그인 버튼을 누르면 Angular Frontend에서 값을 받아 AuthController로 값을 넘겨준다. 그 후 AuthService에서 로그인을 시도를 할 때 User Repository가 DB에서 값을 받아 결괏값을 순차대로 넘겨준다. 로그인 성공 시 main page 이동을 요청하고 실패 시 error 메시지를 띄운다

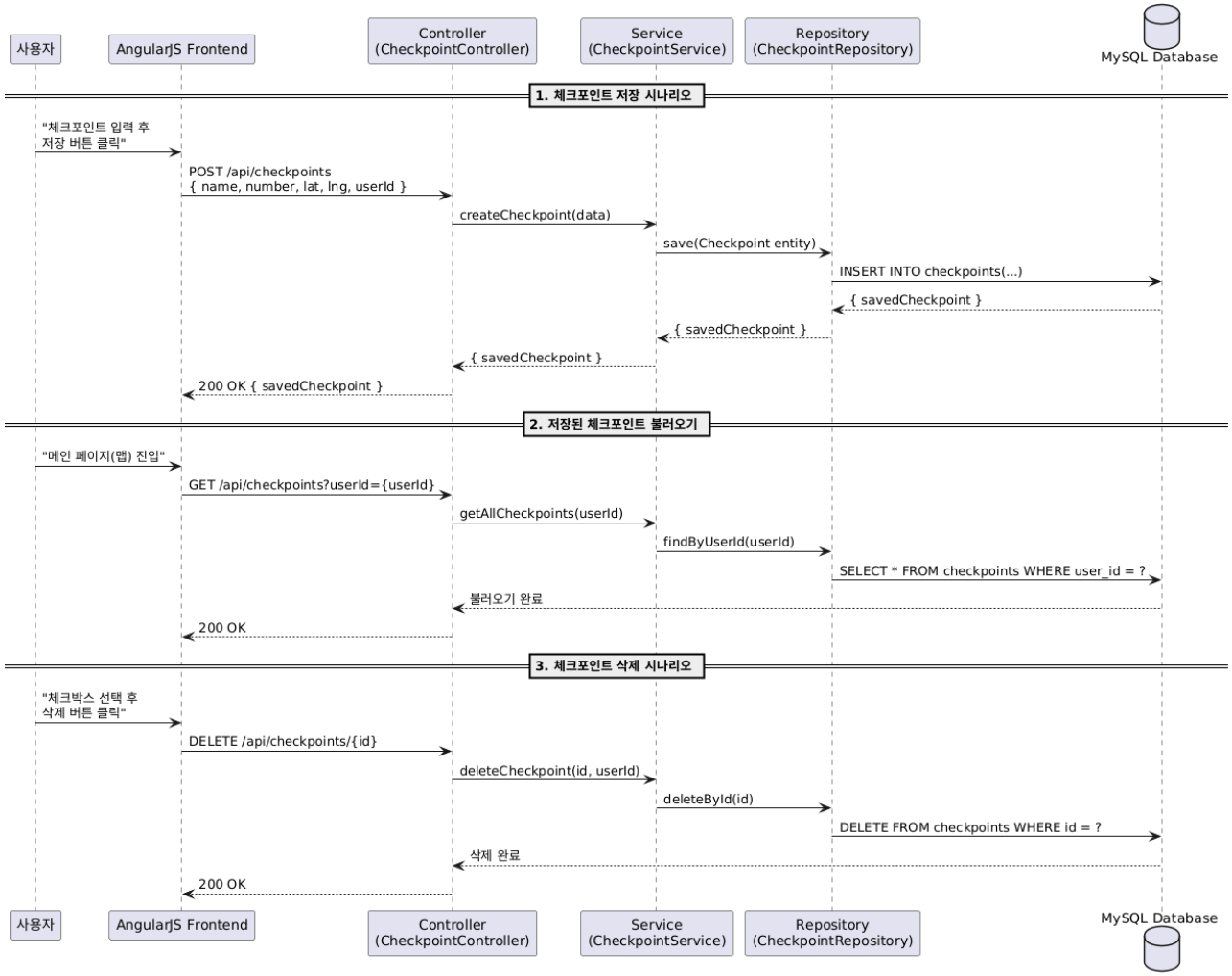
- Register



사용자가 회원가입 정보를 입력하고 회원가입 버튼을 누르면 Angular Frontend에서 값을 받아 AuthController로 값을 넘겨준다. 그 후 AuthService에서 로그인을 시도를 할 때 User Repository가 DB에서 값을 넘겨주면서 유형성 검사 및 엔티티를 생성하며 결괏값을 순차대로 넘겨준다. 로그인 성공 시 login page 이동을 요청하고 실패 시 error 메시지를 띄운다.

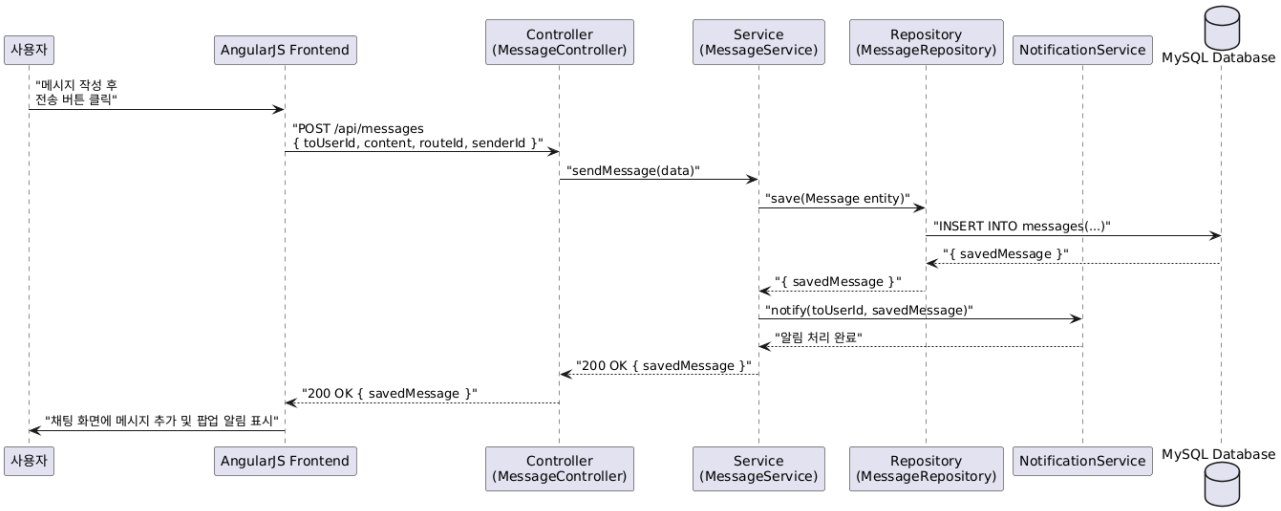
-

- Checkpoint



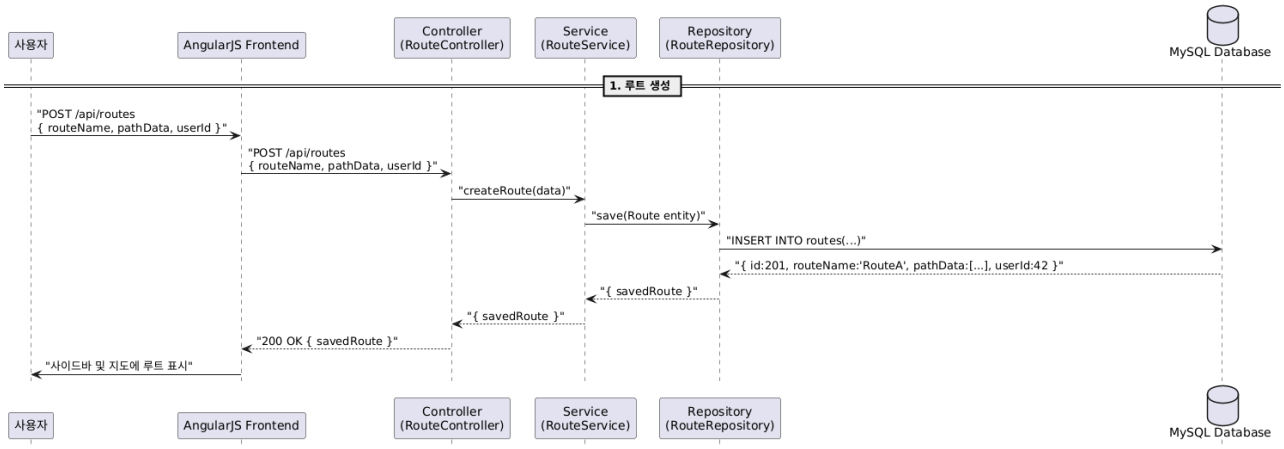
좌표를 클릭하여 좌표정보를 얻으면 좌표이름과 함께 저장한다. DB에 정상이 성공하면 저장이 완료된다. 만약 삭제를 누른다면 DB에서 완전히 삭제가 되는 것이 아니라 지도에만 뜨지 않게 Boolean 값을 false로 변환한다. 그 후 체크포인트 불러오기를 하면 boolean값이 true인 객체들만 불러온다.

- Message



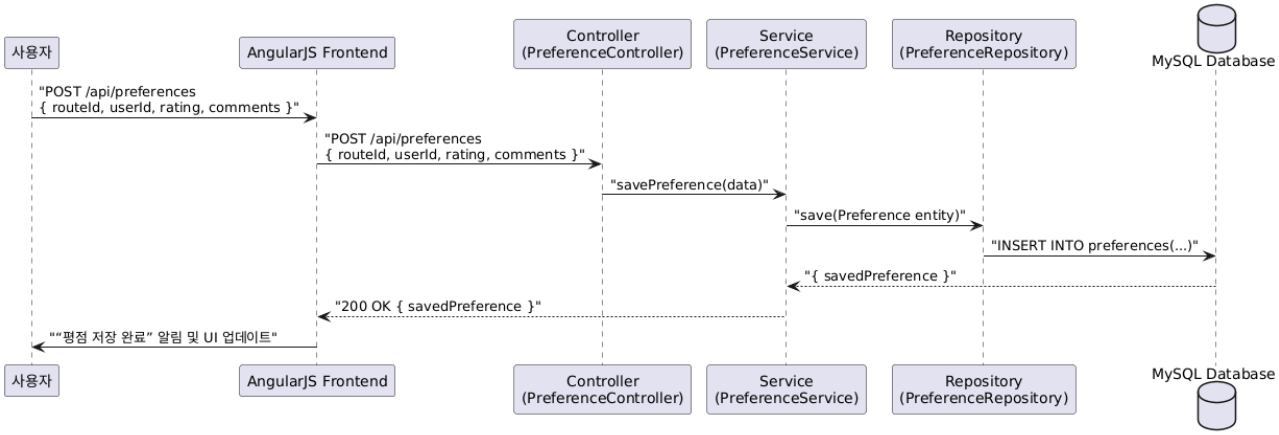
메시지 입력창에 메지시를 입력하여 전송하면 DB에 전송한 그룹명, 전송자, 시간이 DB에 기록이 되며 나중에 그 그룹에 들어가더라도 DB에 저장이 되어있어 모든 전송 메지지가 출력된다.

- Make Route



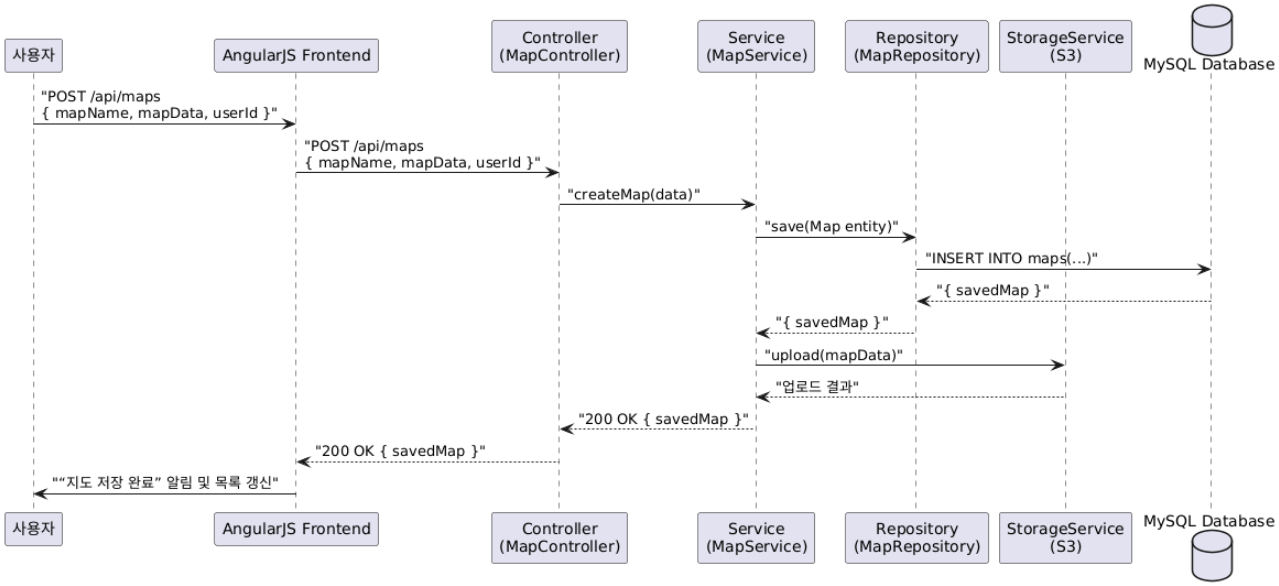
루트 생성시 지정된 체크포인트들을 읽어와서 순회하되 도로의 선호도의 정보를 이용하여 다익스트라 알고리즘을 이용해 루트를 생성하고 지도에 표시한다

- Preference



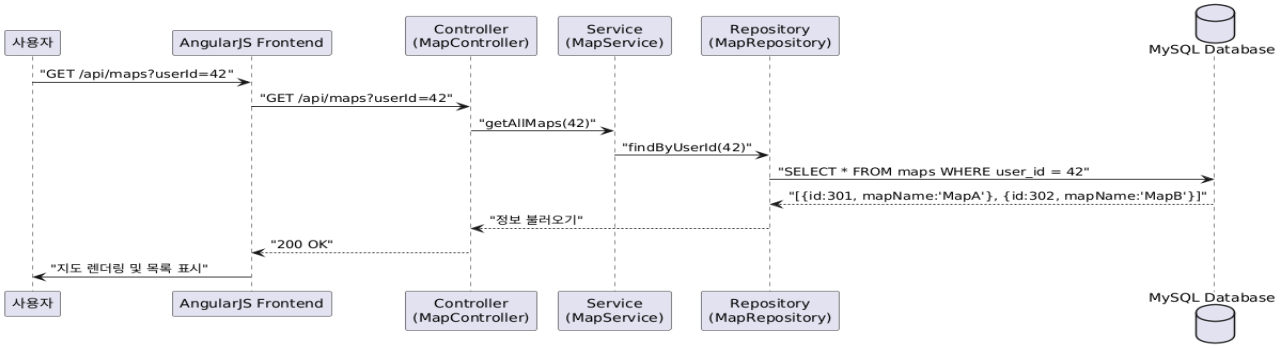
OSM으로 불러온 도로들의 중에 1개를 선택하여 이름, 이유, 선호도 정도를 설정하여 DB에 저장한다.

- Save Map



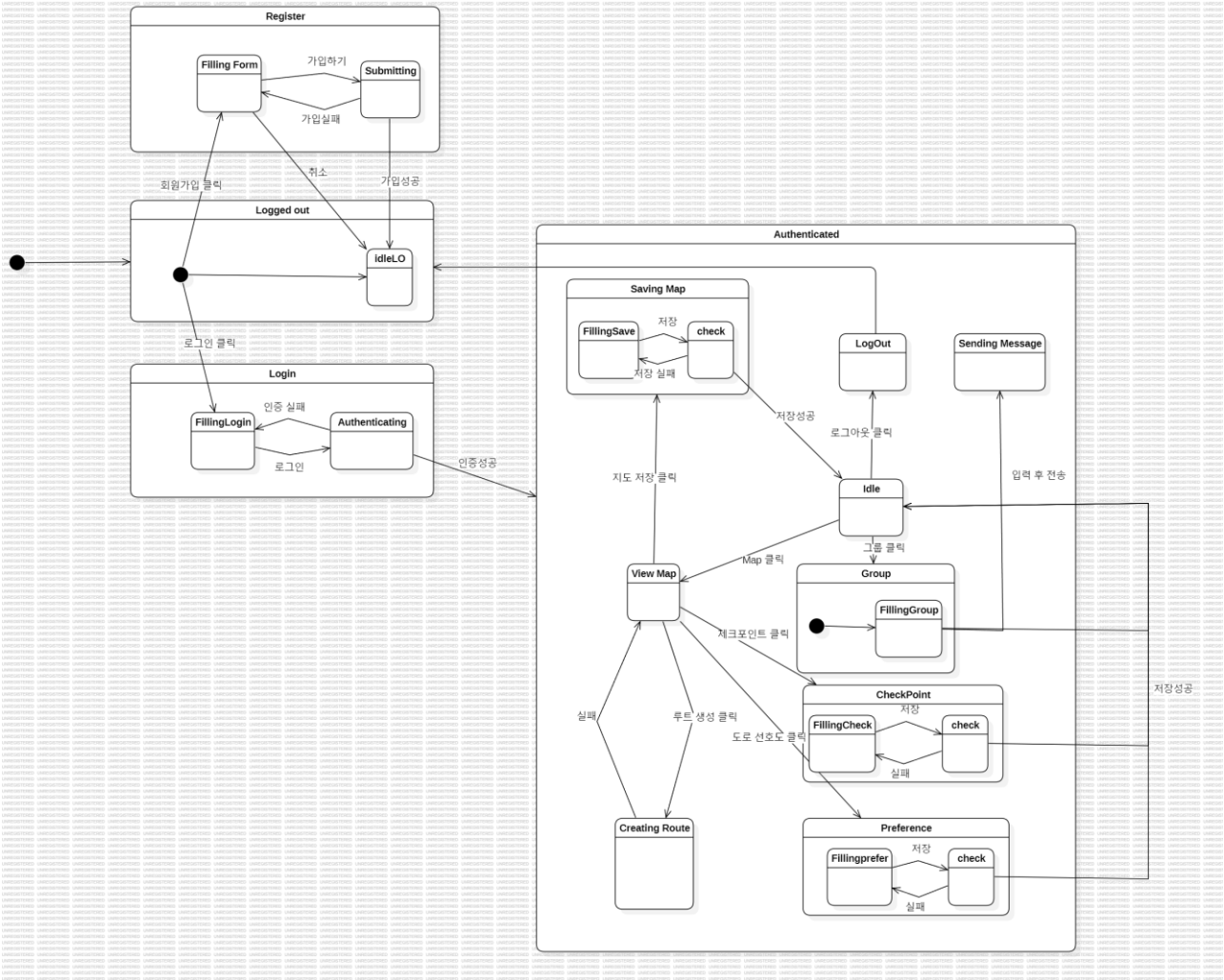
지도의 체크포인트 정보 선호도 정보 생성 루트의 정보를 생성 날짜와 함께 DB에 저장한다

- Show Map



DB에 저장되어 있는 내용들을 불러와서 지도에 표시한다.

4. State machine diagram



이 다이어그램은 자전거 경로 공유 시스템인 **RIDE MAP**의 사용자 상태 흐름을 설명하는 상태 다이어그램이다. 사용자는 애플리케이션을 처음 실행하면 **Logout** 상태로 시작한다. 이 상태에서는 회원가입 또는 로그인을 선택할 수 있다.

사용자가 회원가입을 선택하게 되면 먼저 가입 정보를 입력하는 **Filling Form** 상태로 이동하게 된다. 이곳에서 아이디 비밀번호 정보를 입력하고 '가입하기' 버튼을 누르면 **Submitting** 상태로 전환되어 가입 요청이 서버로 전송된다. 가입 요청이 실패할 경우 다시 **Filling Form** 상태로 돌아가고, 가입에 성공하면 다시 로그인 페이지로 이동하여 로그인을 하면 된다.

한편, 사용자가 **로그인(Login)**을 선택하면 **Filling Login** 상태에서 인증 정보를 입력한 후, '로그인' 버튼을 누르면 **Authenticating** 상태로 전환된다. 인증에 성공하면 사용자는 **Authenticated** 상태로 진입하며, 인증 실패 시 다시 로그인 정보를 입력하는 상태로 돌아간다.

Authenticated 상태는 사용자가 시스템에 정상적으로 로그인한 이후의 모든 기능을 사용할 수 있는 중심 상태이다. 이후 인증에 성공하면 메인 페이지로 이동하여 로그아웃, 맵, 그룹, 저장 중 하나를 선택할 수 있다. 로그아웃 버튼을 클릭하면 **Log Out** 상태로 이동하며, 그룹으로 이동 후 메시지를 입력하고 전송하면 **Sending Message** 상태로 진입한다. 또한, 사용자가 지도를 클릭하면 **View Map** 상태로 진입하

게 되며, 이곳에서 체크포인트 클릭, 경로 생성, 그룹 참여 등의 세부 기능으로 확장이 된다.

지도를 저장하고 싶을 경우 사용자는 **Saving Map** 상태로 진입한다. 이때 먼저 **Filling Save** 상태에서 저장 정보를 입력하고, '저장' 버튼을 클릭하면 **check** 상태로 넘어가 저장 요청을 확인한다. 저장이 성공하면 **Idle**로 복귀하며, 실패 시에는 다시 저장 입력 상태로 돌아간다. 체크포인트를 설정하는 과정도 유사하다. **Check Point** 상태에서 먼저 **Filling Check** 상태에서 정보를 입력하고, 이후 **check** 상태로 넘어가 저장을 시도하며, 성공 여부에 따라 전이된다.

또한, 사용자는 설정 관련 기능을 수행하기 위해 **Preference** 상태로 이동할 수 있다. 이곳 역시 정보 입력(**Filling prefer**) 후 저장 확인(**check**) 절차를 거치며, 성공 여부에 따라 상태가 달라진다. 그룹 기능의 경우에는 **Group** 상태에서 **Filling Group** 단계로 진입하여 관련 정보를 처리한다.

이처럼 RIDE MAP 시스템은 사용자의 상태 전이에 따라 다양한 기능들이 체계적으로 연결되어 있으며, 각 단계는 입력, 확인, 성공/실패 흐름으로 구성되어 있다.

5. Implementation requirements

UML - PLANT UML(PlantUM을 이용하여 diagram을 간단하게 나타냄)

OSM – 지도 API

SDK17버전

6. Glossary

AUTH – 인증을 위해 사용할 service/controller

Repository – db와의 정보교환에 사용할 클래스

Entity – db에 저장할 특성을 나타낼 클래스

7. References

Plantuml - https://www.plantuml.com/plantuml/uml/SyfFKj2rKt3CoKnELR1Io4ZDoSa700001

Uml을 제작하기 위해 사용할 페이지