

A cozy living room scene featuring a round wooden coffee table in the foreground. On the table sits a modern, cylindrical lamp with a warm glow, a dark mug, a small bowl of cookies, and some papers. In the background, a light-colored sofa is adorned with a blue and a yellow cushion. To the right, a wooden chair with a woven seat is visible. The overall atmosphere is warm and inviting.

LISTENING WITH

FINAL REVIEW

JUNE 2022



INDEX

1. Review

1.1 Keyword Reviewal

1.2 Team Replacement

2. Overview

2.1 Technical Information

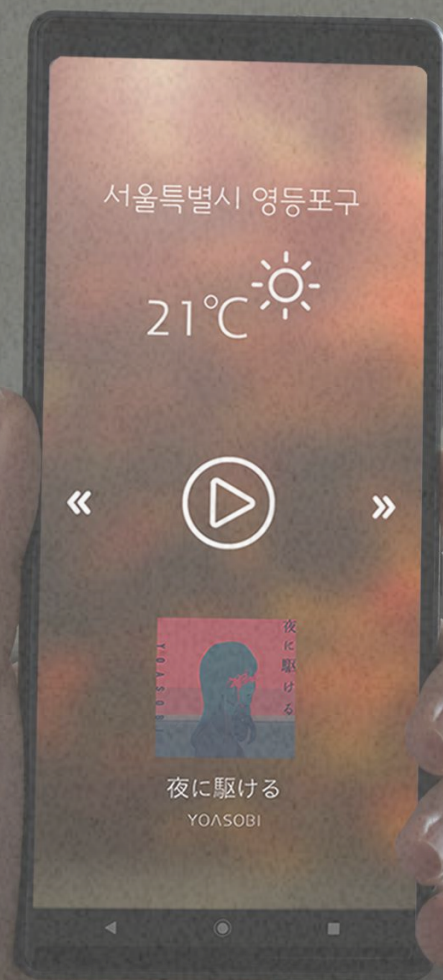
2.2 Application Workflow

2.3 Prototyping

3. Conclusion

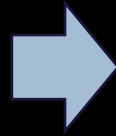
3.1 Pros&Cons

3.2 Follow-up Task



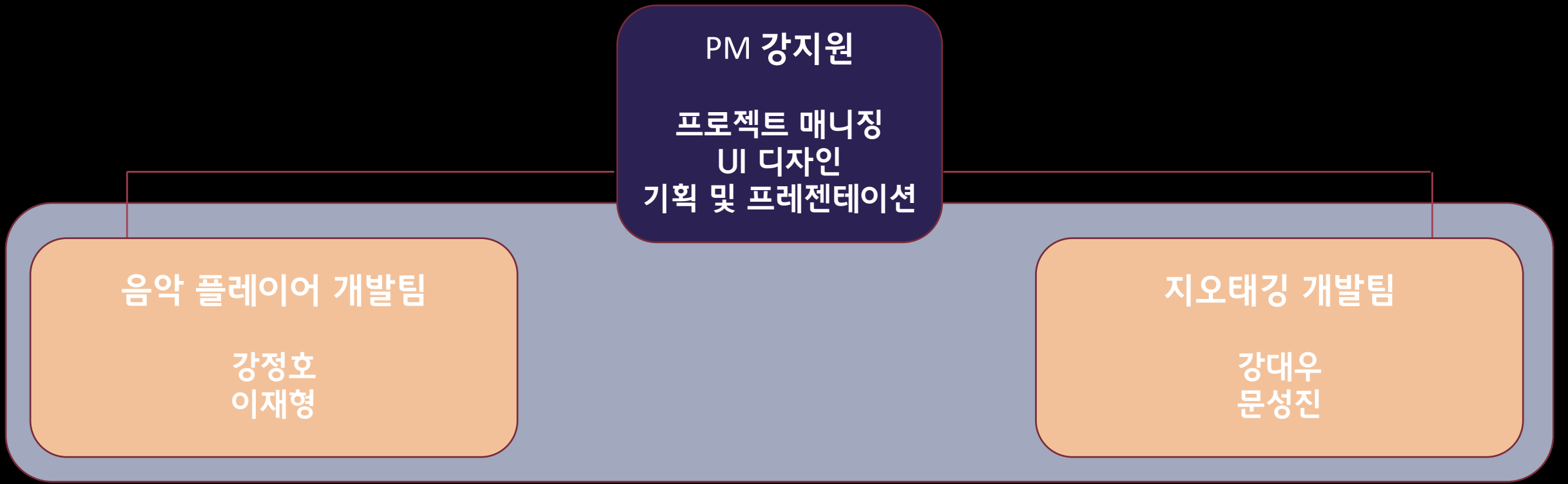
REVIEW

Parameter 분석으로
생성한 DB를
날씨 및 시간정보와 비교하여
최적한 악곡을 큐레이션하는
음악 애플리케이션



사전 정의된 Playlist를
날씨 및 계절정보와 비교하여
재생하는
음악 애플리케이션

Keyword Reviewal

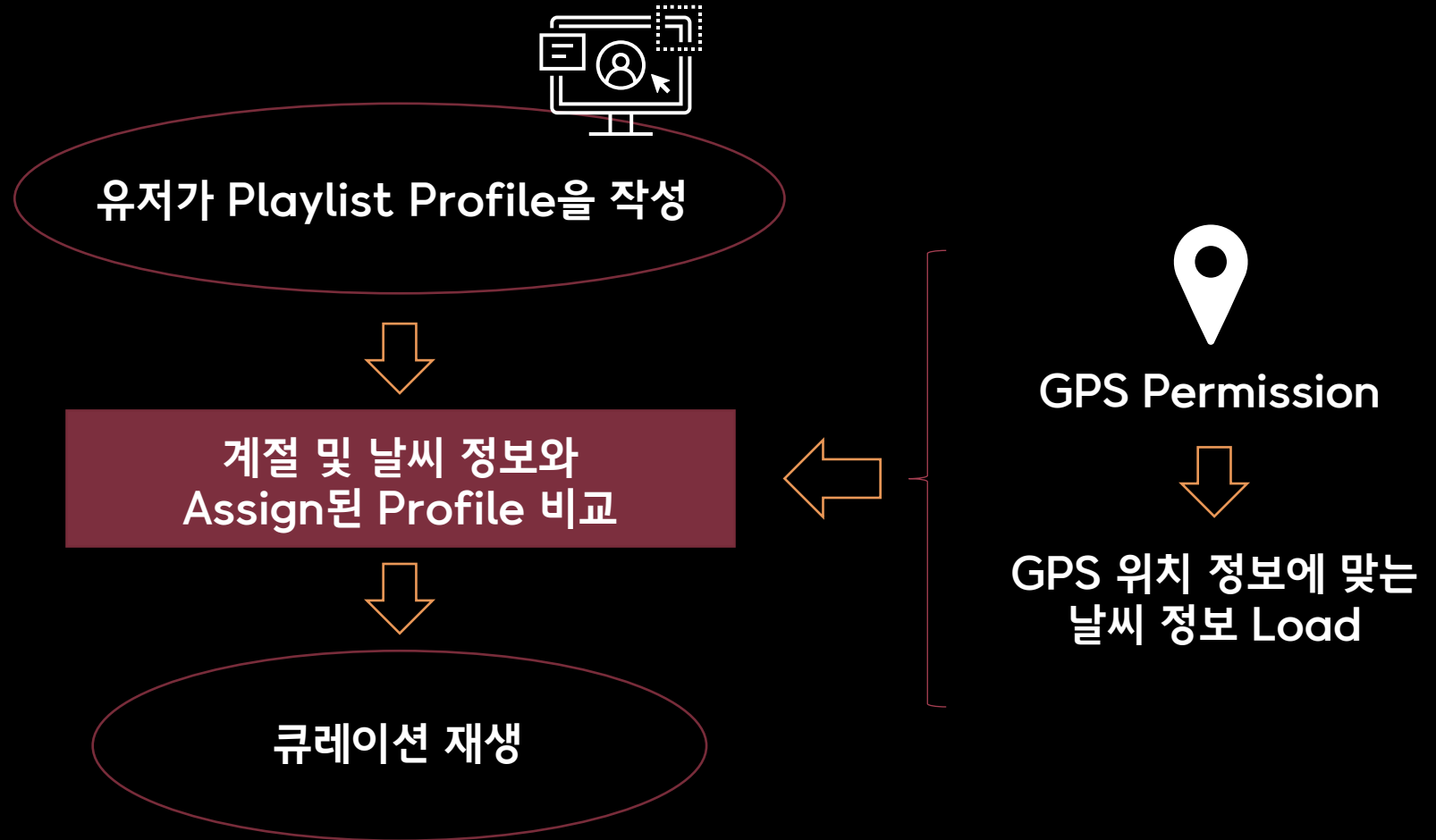


Team Replacement



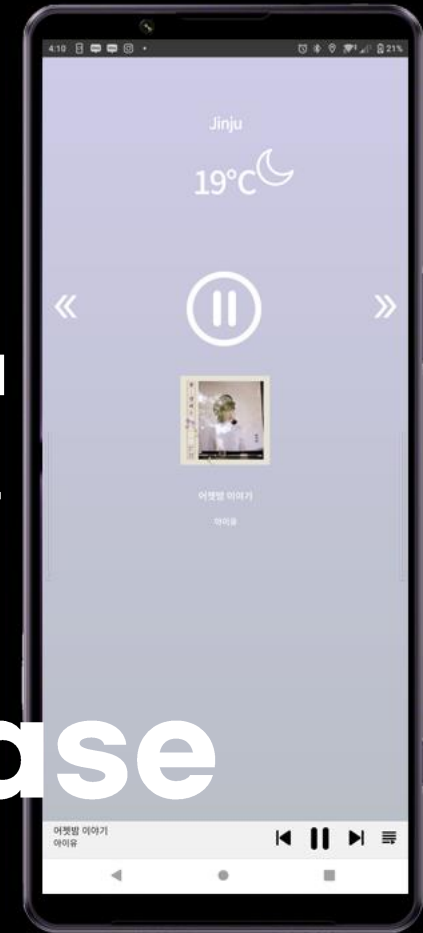


애플리케이션 기동



Application Workflow

API
JSON Calendar
MediaPlayer
bindService Background
NotificationPlayer
GPSTransfer
LocationManager
Database



Technical Information

기상청 단기예보 API를 활용하여 날씨정보 취득

dataType는 Gson 라이브러리 파싱을 위해
JSON으로 명시

Android의 **Calendar**에 기한 지오태깅

X, Y 좌표는 Android의
LocationManager에서 취득후 변형
(**GPSTransfer class** 참고)

```
@Expose
@SerializedName("totalCount")
private int totalCount;
@Expose
@SerializedName("items")
private Items items;
@Expose
@SerializedName("dataType")
private String dataType;

public int getTotalCount() { return totalCount; }

public void setTotalCount(int totalCount) { this.totalCount = totalCount; }

public int getNumOfRows() { return numOfRows; }

public void setNumOfRows(int numOfRows) { this.numOfRows = numOfRows; }

public int getPageNo() { return pageNo; }

public void setPageNo(int pageNo) { this.pageNo = pageNo; }

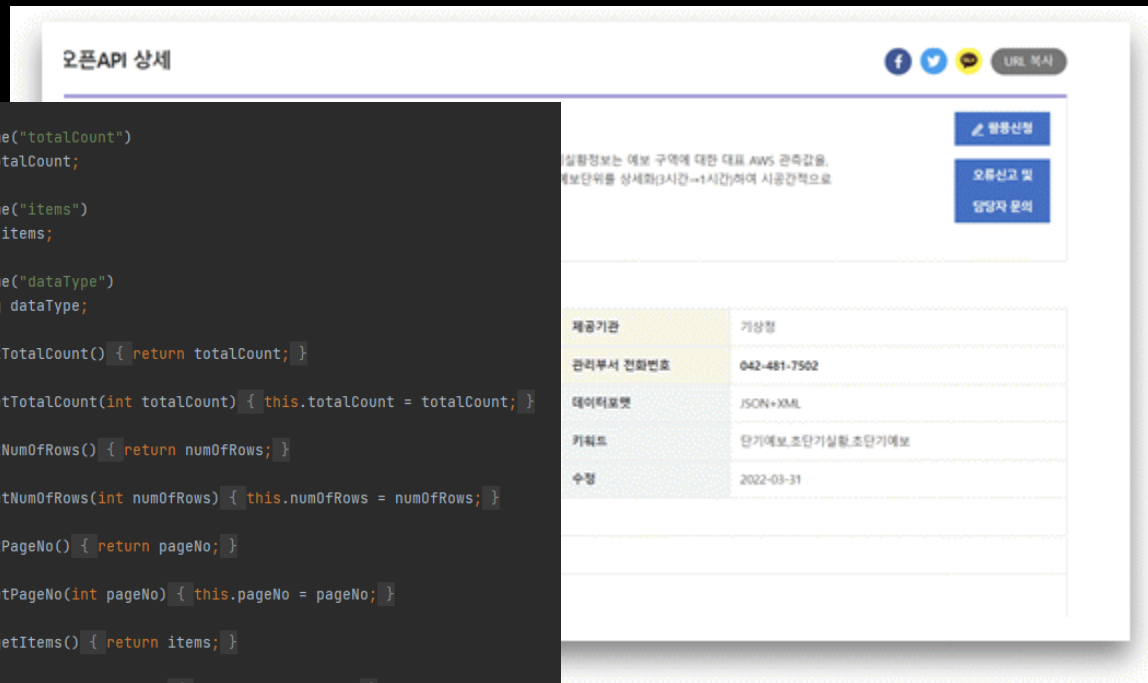
public Items getItems() { return items; }

public void setItems(Items items) { this.items = items; }

public String getDataType() { return dataType; }

public void setDataType(String dataType) { this.dataType = dataType; }

@Override
public String toString() {
    return "Body{" +
        "numOfRows=" + numOfRows +
        ", pageNo=" + pageNo +
        ", totalCount=" + totalCount +
        ", items=" + items +
        ", dataType=" + dataType + '\n' +
        '}';
}
```



Technical Information

Android에서 제공하는 **MediaPlayer**를 활용

Service와 Activity간의 Interaction 위해 **bindService** 사용.
NotificationPlayer에 따른 조작 가능

Artist, Album, Title, Track 등의 value를 설정

Database에 플레이리스트 보존. Field는 id, name, count, musiclist

재생시 musiclist의 값을 input하여
MusicFile 객체를 생성. MusicFile을 포함하는
List를 Service에 전송

```
@Database(entities = {Playlist.class}, version = 1, exportSchema = false)
public abstract class RoomDB extends RoomDatabase {
    private final String TAG = RoomDB.class.getSimpleName();
    private static String DATABASE_NAME = "database";
    private static RoomDB instance;

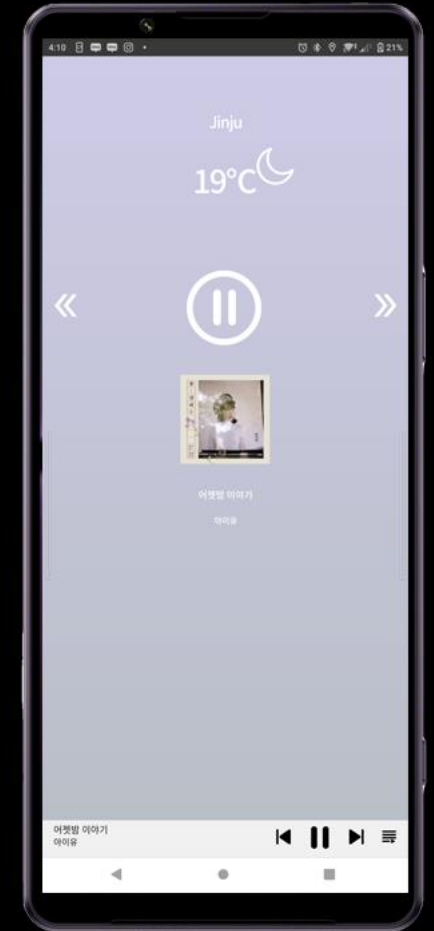
    public static final int NUMBER_OF_THREADS = 4;
    public static final ExecutorService databaseWriteExecutor
        = Executors.newFixedThreadPool(NUMBER_OF_THREADS);

    public synchronized static RoomDB getInstance(Context context){
        if(instance == null){
            instance = Room.databaseBuilder(context.getApplicationContext(), RoomDB.class, DATABASE_NAME)
                .allowMainThreadQueries()
                .fallbackToDestructiveMigration()
                .addCallback(setInitialRoomDatabaseCallback)
                .build();
        }
        return instance;
    }
}

public abstract PlaylistDao playlistDao();

private static final RoomDatabase.Callback setInitialRoomDatabaseCallback =
    new RoomDatabase.Callback(){
        @Override
        public void onCreate(@NonNull SupportSQLiteDatabase db) {
            super.onCreate(db);
            databaseWriteExecutor.execute(()->{
                PlaylistDao playlistDao = instance.playlistDao();

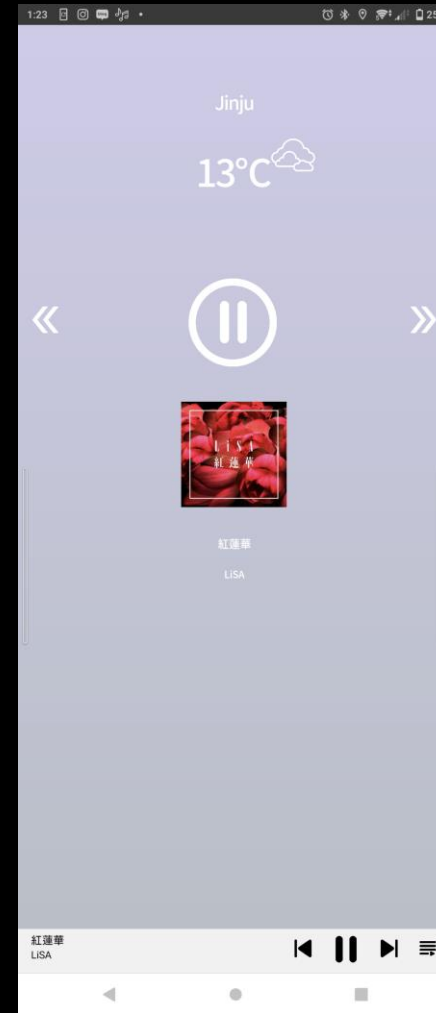
                Playlist playlistBasic1 = new Playlist("봄", 0);
                Playlist playlistBasic2 = new Playlist("여름", 0);
                Playlist playlistBasic3 = new Playlist("가을", 0);
                Playlist playlistBasic4 = new Playlist("겨울", 0);
                Playlist playlistBasic5 = new Playlist("비", 0);
```



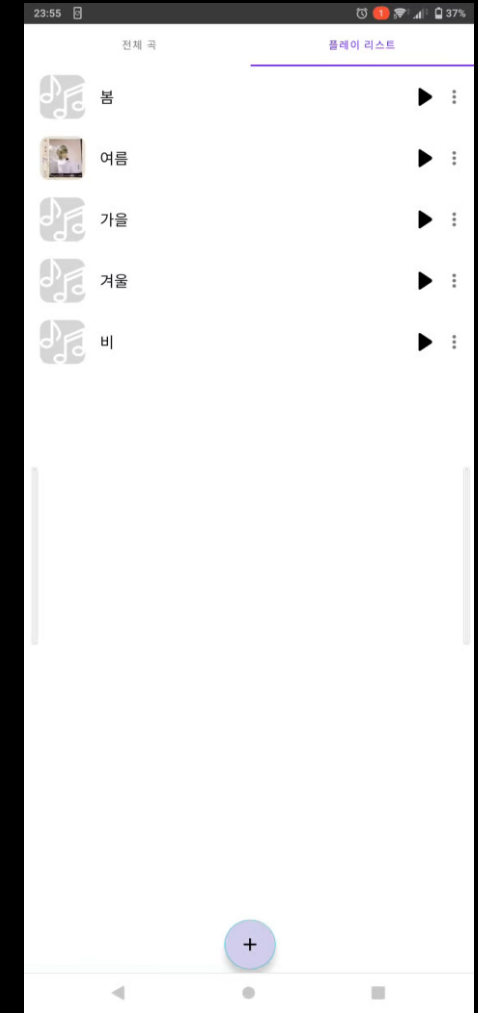
Technical Information



Idle



Player



Database

Prototyping

반응형 Design의 채용으로 다양한 Aspect Ratio와 Use Case에 대응
날씨 API를 10분 단위로 갱신하여 리소스 사용의 최적화와 함께 최신화된 정보를 제공



CONCLUSION

LISTENING WITH

계절과 날씨에 맞는 악곡을 자동으로 재생 가능

반응형 Design의 채용으로 다양한 Use Case에 대응

악곡 추가시 자동으로 Refresh되어 List-up

Pros

음장 효과 적용되지 않음

Playlist의 편집 불가

날씨 Data의 Refresh가 1일 100회에 한정

Cons

Pros&Cons

LISTENING WITH

자연어에 기반 음악 큐레이션 (ex) Title에 밤, moon, 夜 등이 포함되는 음악을 밤에 재생)

Playlist의 편집 기능 추가

날씨 Data의 Refresh Trigger를 다변화 (ex) User의 직접 갱신 가능)

가속도 센서 등과 연계하여 유저의 상태를 파악. 자연어에 기하여 음악을 큐레이션

금회 Drop된 시간대에 따른 큐레이션 기능 추가

Follow-up Task

LISTENING WITH

Project COW Music

강지원 · 강대우 · 강정호 · 문성진 · 이재형