최종결과보고서

딥 러닝을 활용한 음악 Beat Note 자동 생성 서비스가 있는 VR 리듬게임

Beat Make & Crush

VR rhythm game with Beat Note automatic generation service using deep learning

Ver. 1.1

2019. 12. 03

한국외국어대학교

정보통신공학과

Team VRR

문서 정보

구 분	소 속	성 명	날 짜	서 명
작성자	한국외국어대학교	박영준	2019. 12. 03	
	한국외국어대학교	문명기	2019. 12. 03	
	한국외국어대학교	김세진	2019. 12. 03	
	한국외국어대학교	조동철	2019. 12. 03	
	한국외국어대학교	이호찬	2019. 12. 03	
검토자	한국외국어대학교	박영준	2019. 12. 03	
	한국외국어대학교	문명기	2019. 12. 03	
	한국외국어대학교	김세진	2019. 12. 03	
	한국외국어대학교	조동철	2019. 12. 03	
	한국외국어대학교	이호찬	2019. 12. 03	
사용자				
승인자	한국외국어대학교	홍진표	2019. 12. 03	

머리말

본 문서는 자사의 '딥 러닝을 활용한 음악 Beat Note 자동 생성 서비스가 있는 VR 리듬게임'의 서비스 소개와 시스템을 구축하기 위한 사업계획과 설계방법을 기술한다.

개정 이력

버전	작성자	개정일자	개정 내역	승인자
1.0	박영준	2019. 12. 03.	초안 작성	
	문명기			
	김세진			
	조동철			
	이호찬			
	검토자	박영준		
1.1	박영준	2019. 12. 19.	요구사항 재 구성 및 기술 추가	
	문명기			
	김세진			
	조동철			
	이호찬			
	검토자	박영준		

목 차

1 개요

본 장에서는 딥 러닝을 활용한 음악 Beat Note 자동 생성 서비스가 있는 VR 리듬게임 VRR 에 대한 목적과 범위, 용어 및 약어를 제시한다.

1.1 목적

본 프로젝트는 키넥트를 이용해 VR 리듬게임을 컨트롤러 없이 손과 발로 Play 할 수 있는 VRR 게임을 소개 및 설계하고 딥러닝을 활용한 음악 Beat Note 자동 생성 서비스를 제공하기 위해 필요한 딥러닝 모델을 설계, 구현한다. 또한 게임 play 기록과 점수, 자신이 만든 Beat Note 가 저장 되어있는 것을 확인할 수 있는 Web Platform 을 설계한다.

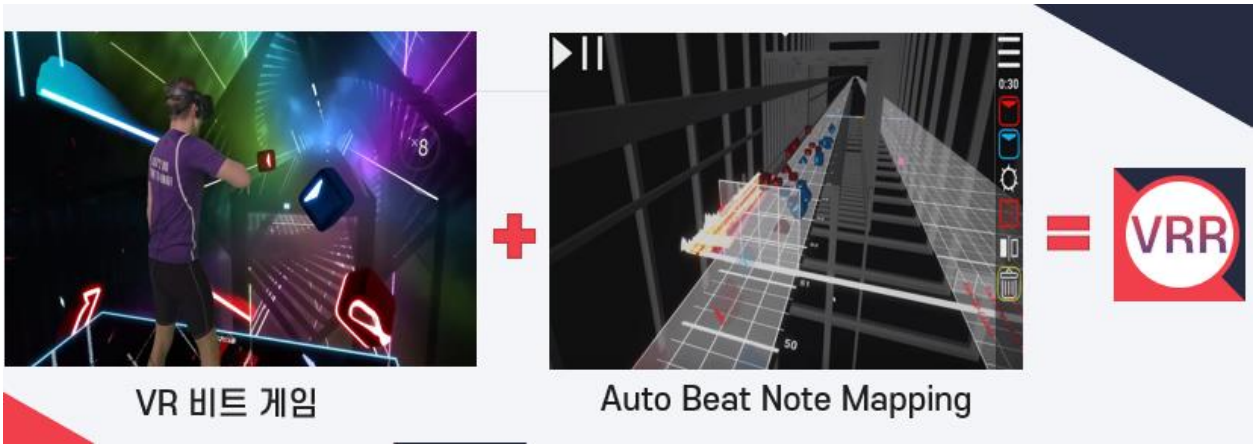
이를 통해 사용자가 VR 리듬게임을 즐기면서 자신이 Play 하고 싶은 Note 가 없을 경우 음원 파일을 Web Platform 에 올리고 설계한 딥러닝 모델을 통해 게임에서 필요한 Beat Note 가 생성되어 음악을 전혀 모르는 사용자여도 자신이 Play 하고자 하는 음악을 VR 리듬게임으로 즐길 수 있다.

1.2 용어 및 약어

구성 요소	설명
VRR	VR 리듬게임의 약어로 팀명이다.
Beat Crush	자사의 게임 이름으로 키넥트를 이용해 몸으로 게임하며 Beat Note 를 치는 형식의 'Crush'를 의미한다.
비트 (Beat)	음악의 박자, 패턴과 같은 리듬을 의미한다.
비트 노트 맵 (Beat Note map)	음원을 기반으로 만들어진 게임 플레이에 필요한 리듬게임 악보이다.
비트 노트 (Beat Note)	만들어진 Beat Note map 에 있는 하나 하나의 박자를 의미
Oculus Rift DK2	Oculus Rift 는 VR 기기 이름, Development Kit 의 약자이다

2 서비스 개요

VRR 서비스는 키넥트와 VR 기기를 이용한 컨트롤러 없이 손 발을 사용한 VR 리듬게임을 제공하며, Beat Note 자동 생성 서비스를 통해 사용자가 원하는 음악을 VR 리듬게임으로 Play 할 수 있는 환경을 제공한다.



2.1 서비스 기획 배경

리듬게임이란 쉽게 말해 '컨트롤러를 통한 빠르고 정확한 조작'이라는 액션 게임적인 요소에 '악기를 조작한다' 라는 시뮬레이션 요소를 더한 장르다. 우리는 이런 음악게임은 '리듬게임', '리듬비트게임', '리듬비트액션', '리듬액션게임'이라고 부리고 있다.

이러한 리듬게임은 비디오 게임 초창기인 80 년대부터 VR 이 상용화된 현재까지 지속적으로 개발되어 왔으며 앞으로도 많은 리듬이 만들어질 전망이다.

이렇게 계속 발전되어 온 리듬게임의 비트 노트를 아직 개발자들이 수작업으로 만들고 있다. 새로운 음악이 나오면 사용자는 노트를 제작할 때까지 기다려야 하고 개발자가 노트를 만들지 않으면 결국 원래 있던 Note 만 Play 할 수 있다. 이를 자동화하기 위해 우리는 딥러닝을 활용해 음원을 넣으면 Beat Note 를 자동으로 만들어주는 서비스를 기획하고자 하였다.

또한 VR 기기의 보급이후 아직 컨트롤러를 이용해 조작하는 게임이 대부분이지만 VRR 게임은 컨트롤러 없이 손과 발을 이용해 게임의 재미와 자유도를 극대화하고자 하였다.

[그림] 비트게임 예시

2.2 서비스 소개

자사의 서비스 'Beat Crush'는, VR(Virtual Reality)기기와 키넥트의 모션인식 기술을 이용하여 **Player 의 Action 으로 날아오는 Beat Note 를 부수는 게임 서비스**이다.

2 인 play 모드는 경쟁하며 콤보에 따른 패널티를 부과해 점수를 높이는 있는 새로운 방법의 VR 리듬게임 서비스이며 키넥트의 모션인식 기능을 활용해 단순히 Beat 를 치는 것에 국한되지 않고 벽을 피하는 등 움직이는 요소로 재미를 극대화한다.

deep learning 기술을 활용해 어떤 음악이던 재미있는 리듬 게임으로 Play 할 수 있도록 **Beat Note 를 자동으로 생성**해 주는 서비스가 있다. 매일 똑같은 Beat Note 를 Play 하는 것은 금방 질리기 쉽고, 흥미가 떨어지기 쉽지만 자신이 Play 하고 싶었던 음악을 Beat Note 로 만들 수 있기 때문에 끊임없이 새로운 리듬게임을 Play 할 수 있다

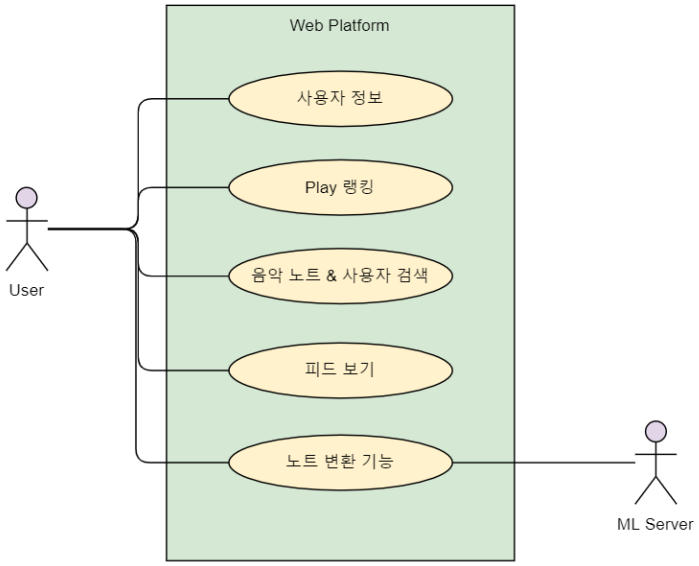
생성된 나만의 Beat Note 는 자사의 VRR Server 를 통해 다른 Player 들과 공유할 수 있고 기록, 정보 등을 확인할 수 있는 웹 플랫폼을 제공하며, 웹 플랫폼을 통해 게임 기록, 사용자 랭킹, 생성한 Beat Note 목록 등을 확인할 수 있는 **편리한 웹 서비스**도 제공한다.

2.3 서비스 기능

2.3.1 Web Platform Use Case Diagram

❒ 반영된 요구사항

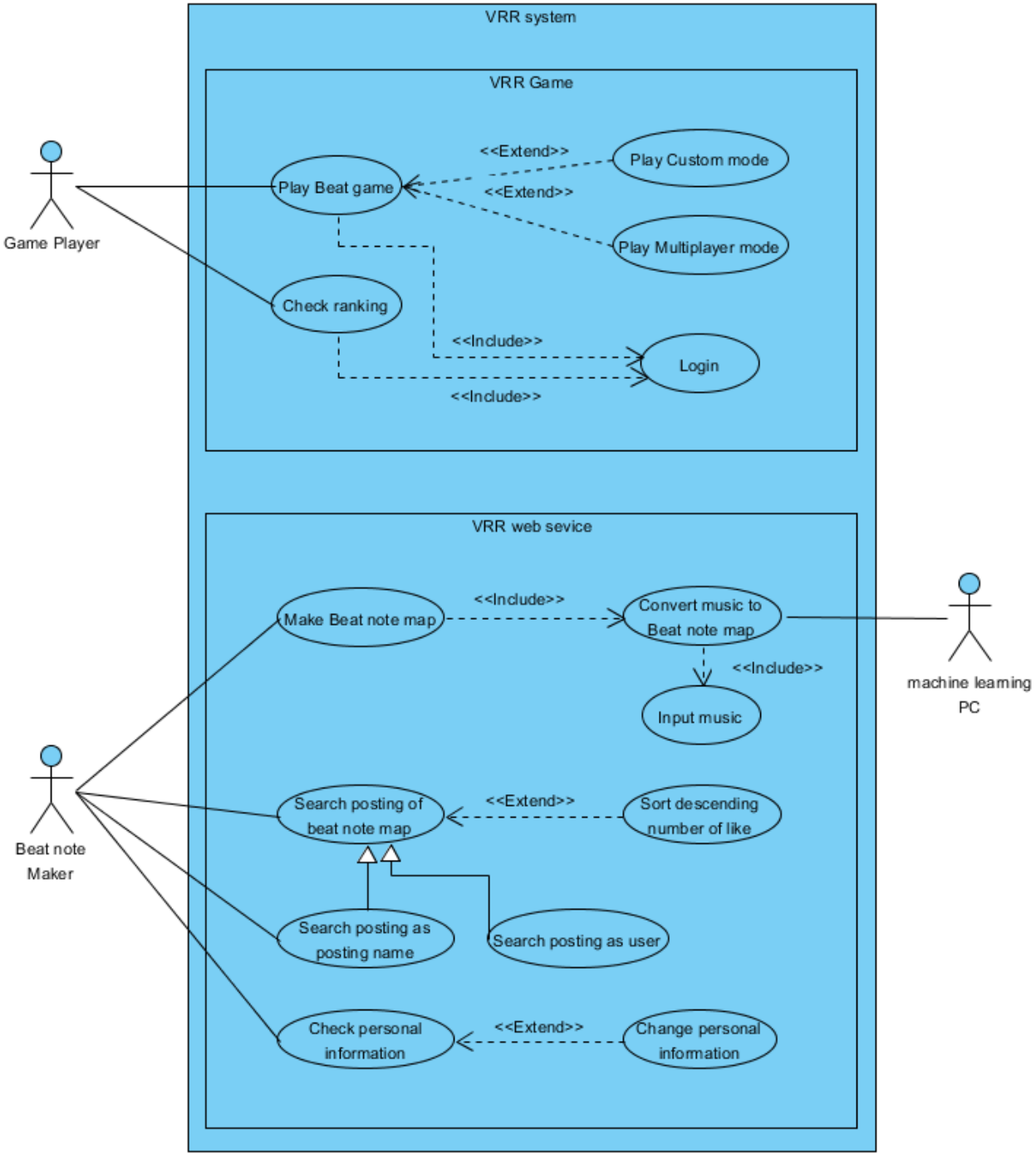
[SFR-I-01~14] 인터페이스 어플리케이션 부 요구사항



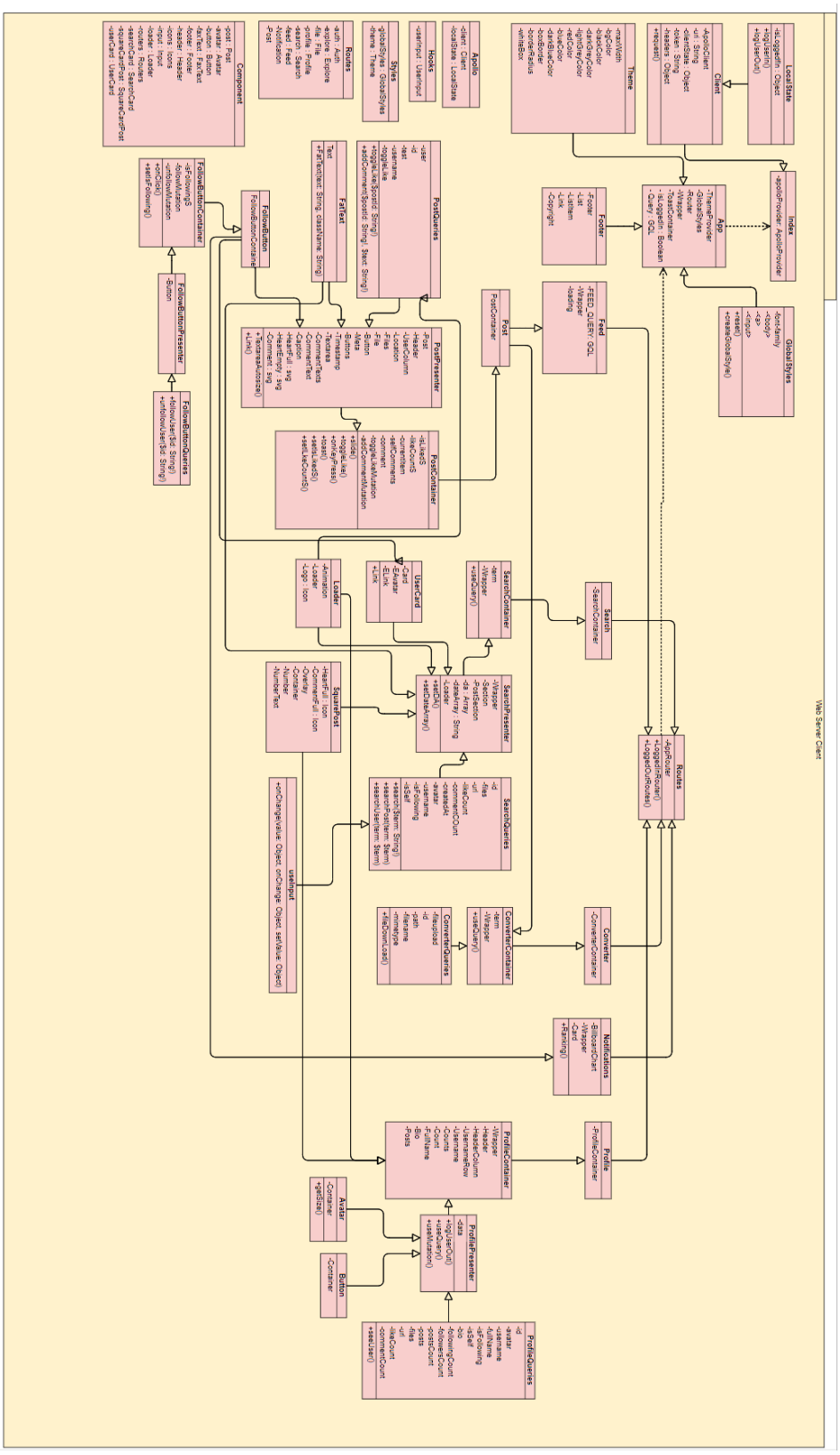
2.3.2 VRR System Use Case Diagram

❒ 반영된 요구사항

[SFR-I-01~14] 인터페이스 어플리케이션 부 요구사항



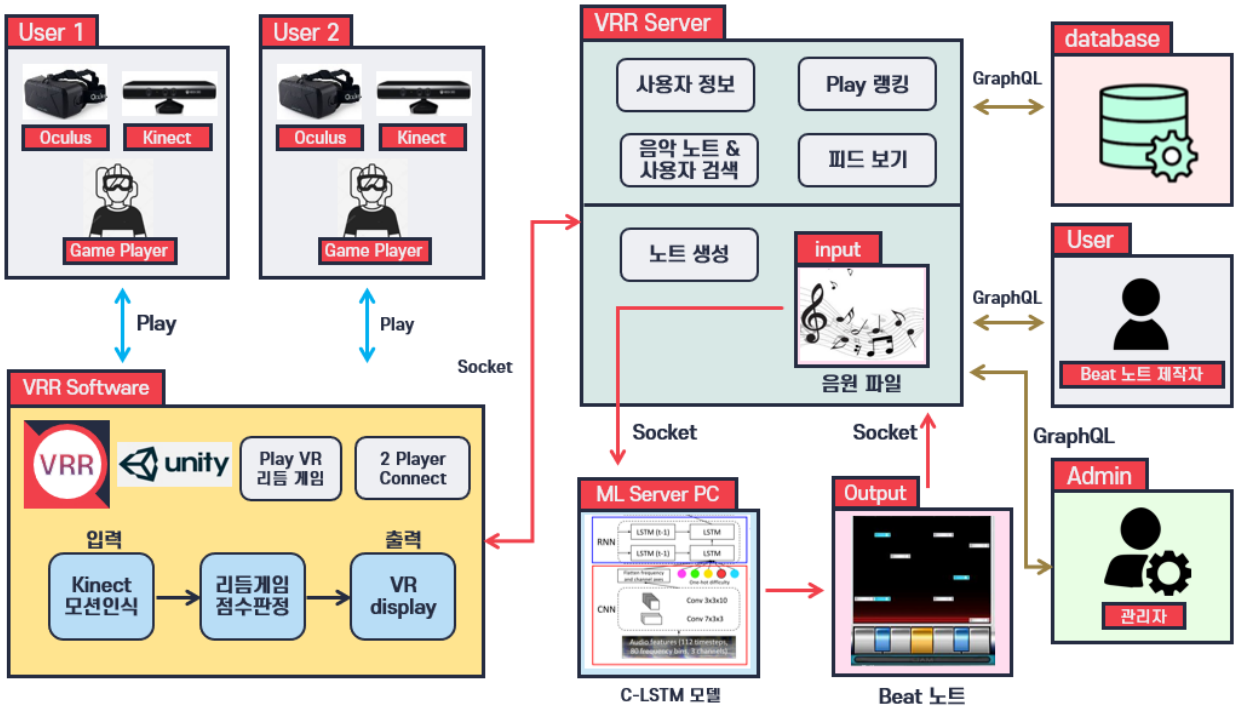
2.3.3 Web Platform Class Diagram



3 시스템 구성

3.1 시스템 구성도

본 시스템의 구성요소에는 User, VRR Software, VRR Web App, VRR Server 가 포함된다.

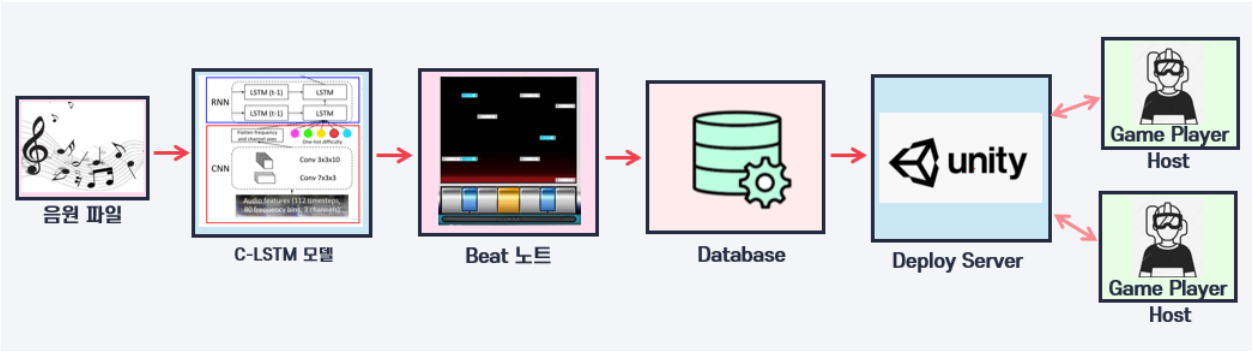


[그림] VRR 시스템 구성도

구성 요소	설명
User	User 는 VRR Game Service 를 받는 대상이다.
VRR Software	VRR Software 는 VRR Game 프로그램이다.
VRR Server	Web App 과 Software 사이에 필요한 Server 이다.
ML Server	비트 노트 자동 생성에 필요한 CNN 모델의 머신러닝 Server 이다.
Database	Database 는 VRR System 과 정보를 주고받는다.

언급한 구성요소에 대한 대략적인 설명을 위와 같다.

3.2 기능 흐름도

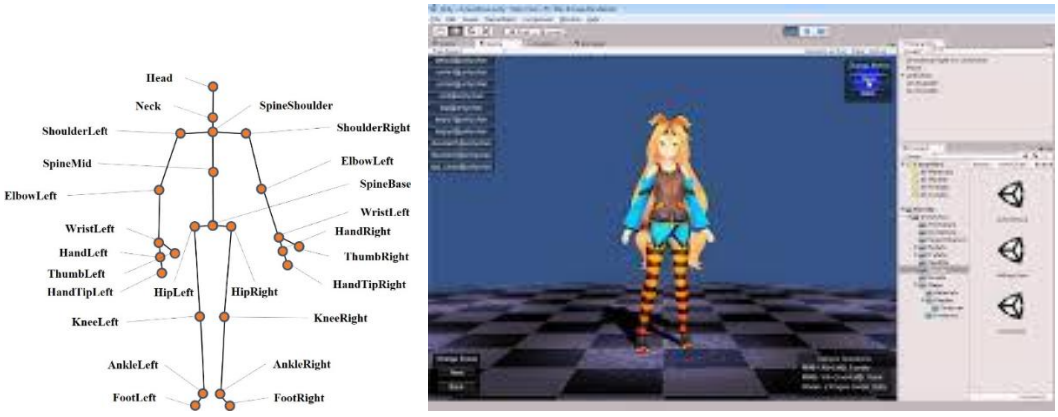


[그림] 기능 흐름도

4 시스템 상세 설계

4.1 VRR 게임 Play

VRR 게임은 Kinect 를 이용해 컨트롤러 없이 손과 발로 Beat Note 를 치며 게임을 할 수 있는 요구사항을 만족시키기 위해 Kinect BodySourceManager 를 이용해 Unity 에 Kinect 의 움직임을 받아온 후 Kincet Avater 의 Unity-Chan Asset 을 활용해 실제 움직이는 Avater 를 Unity 에 띄웠다.



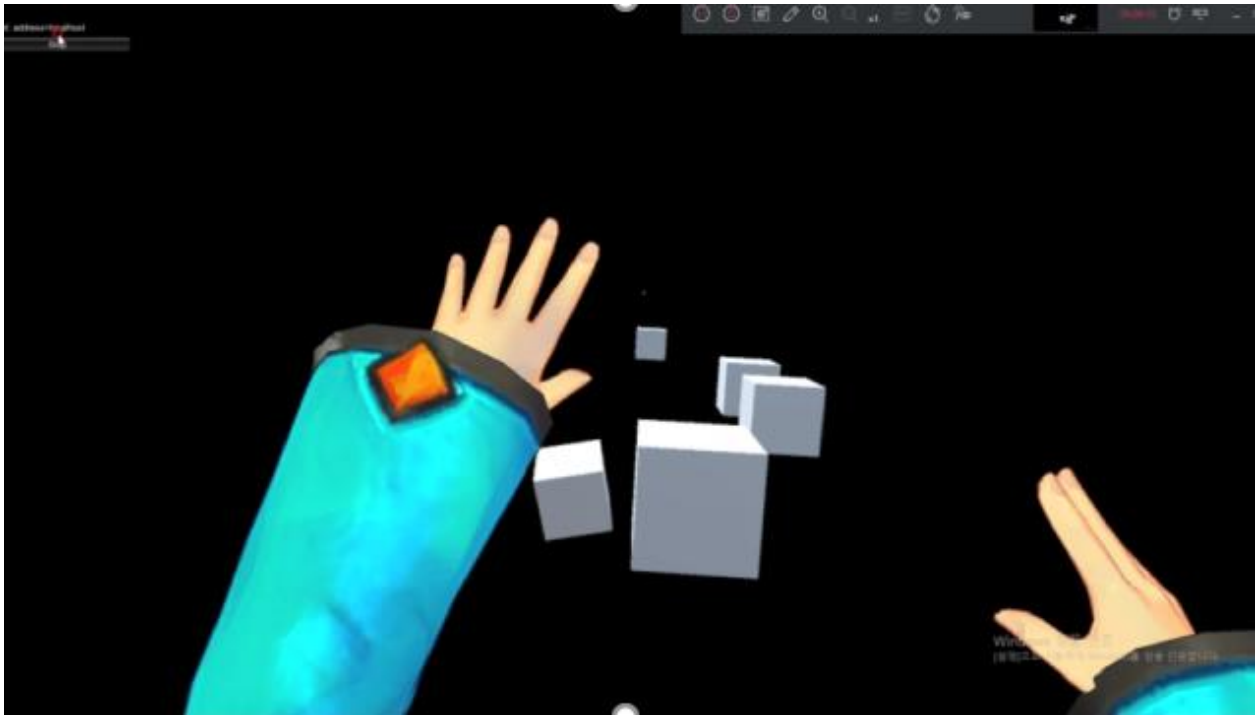
[그림] Kinect Body → Unity Avater 화

kinect asset 을 사용하여 인체의 25 개의 골격 구조를 받아와 골격 구조와 해당 골격의 위치정보를 Unity 가상공간에 배치한다. 25 개의 골격 중에서 hand-tip-right 와 hand-tip-left 를 이용하여 가상공간의 인터페이스를 조작할 수 있다. mesh collider 를 이용하면 두 object 가 충돌한 지점은 contactPoint 함수로 위치 좌표를 받아와서 사용할 수 있다.

[그림] User 사용 장비

만들어진 Beat Note 들이 Oculus VR 기기에 display 되고 KInect 를 통해 동작을 입력 받아 날아오는 Note 를 치면서 게임을 할 수 있다. 틀리지 않고 Note 를 연속으로 치게 되면 점수가 더 많이 올라가고 점수, 랭킹이 데이터베이스에 저장된다.

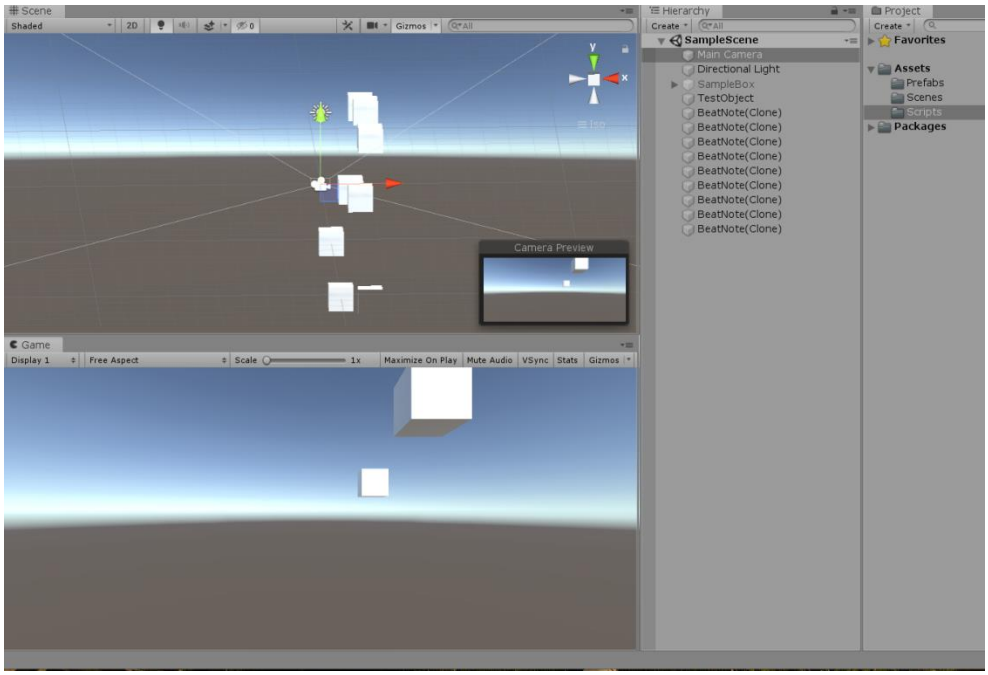
Beat Note 는 전처리 된 음원 데이터 파일을 받아 Unity 에 3D 공간화 시켜 Beat Note 화 하였고 게임을 시작하면 음악에 맞는 적절한 Note Map 이 Loading 되어 VR 로 Display 된다.



[그림] Beat Note 생성 및 게임 Play

BeatNoteMap 데이터를 통해 받은 BeatNote 들의 정보를 List 에 저장한다. 그리고 update()함수(매 frame 마다 실행되는 함수)에서 각 BeatNote 들의 첫 번째 인자 값인 time 을 time.time(프로그램 시작부터 현제까지의 시간)과 비교하면서 BeatNote 를 생성하고 생성위치는 BeatNote 들의 두 번째, 세 번째인자인 lineIndex 와 lineLayer 를 통해 Transform.Translate(x,y,z) 함수를 통하여 배치한다. 그리고 생성된

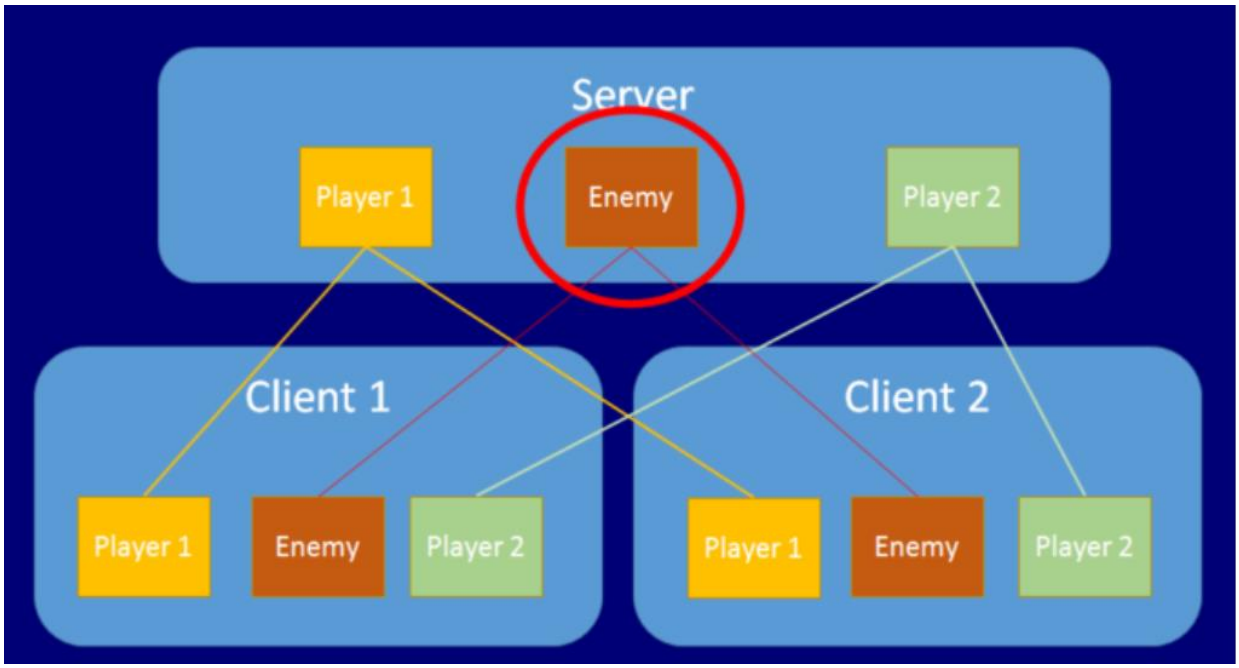
GameObject 들을 새로운 List 에 저장하고 반복문을 통하여 List 의 모든 GameObject 들의 위치를 변화시켜주면 BeatNote 들이 Player 방향으로 이동한다.



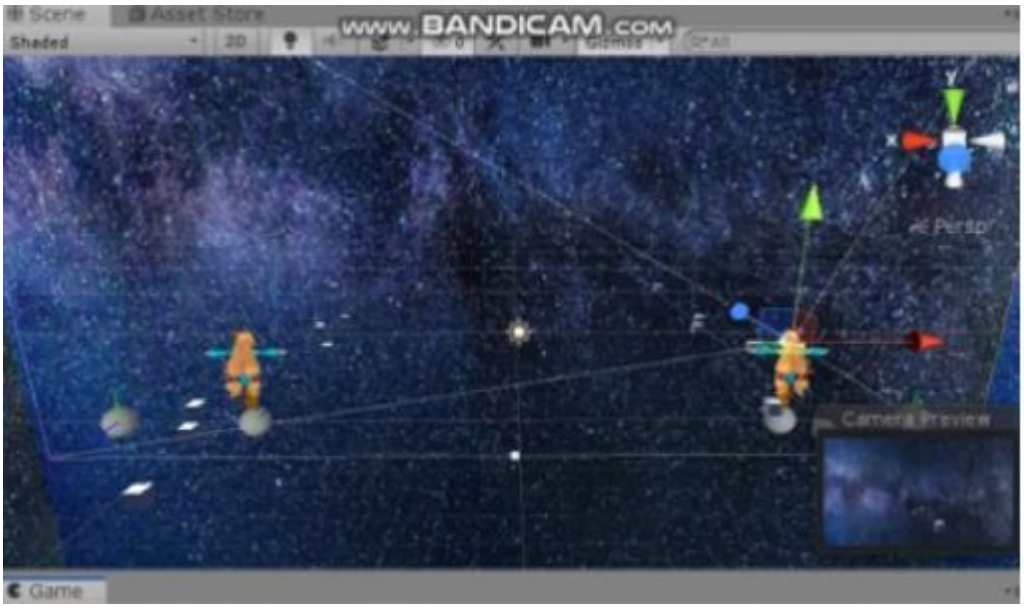
[그림] Beat Note Map 객체

4.2 2 인 Play Mode

게임은 여러 명과 같이하며 경쟁하면 재미를 극대화할 수 있다. VRR 서비스는 멀티플레이어 게임을 구현하기 위해 Unity 의 Mirror Networking Asset 를 사용하여 하나의 서버 환경을 두고 여러 Host 들이 접속하여 객체에 대한 권한을 얻어 게임을 Play 한다.



[그림] Mirror 의 작동 원리



[그림] Mirror 를 통한 2 인 Play 구현

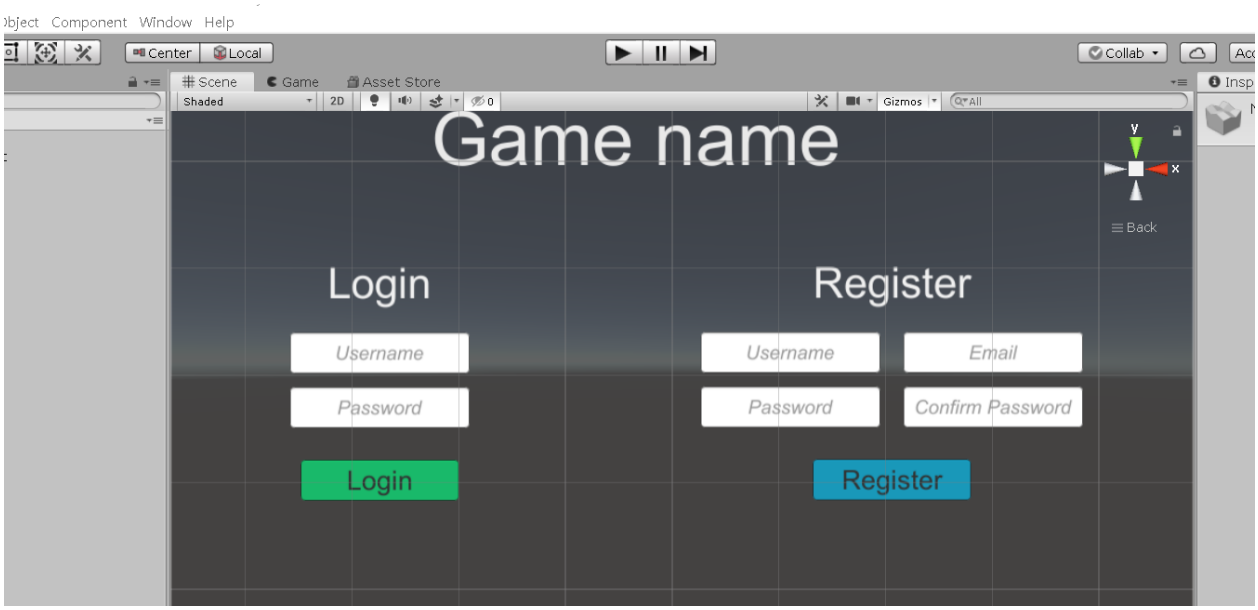
Unity 2 인 플레이를 하기위해 TCP 로 구현되어있는 Unity Mirror Asset 을 사용하여 Server-Client 방식으로 구현했다. 이때, 여러 Host 들은 Deploy Unity Server 로 부터 자신에게 할당된 객체에 대한 조작 권한을 얻고, 권한을 얻지 못한 객체에 대해서는 조작 접근이 불가능하다.

VRR 게임에서는 Deploy Unity Server 로부터 각자의 캐릭터의 조작 권한을 얻어 조작하며 이때 할당된 권한으로 충돌 검사를 해서 여러 비트들에 대하여 물리적 적용을 할 수 있다.

networkidentity : 오브젝트의 고유 id 부여(서버에서 spawn 된 object 관리하기 위함, 사용자의 authority 부여), 이 component 가 있어야만 networkmanager 에 의해 동기화 및 관리됨, client 의 환경에서 authority 를 확인하여 해당 client player 만 object 를 제어가능.

networkTransform : 오브젝트의 움직임 동기화, 각 클라이언트와 서버에서 오브젝트의 transfrom 을 동기화하여 모든 환경에서 위치 동기화

networkmanager : 서버 및 클라이언트의 모든 object(prefabs)들을 생성 및 관리하며 실행이전에 등록된 object 들을 관리 등록되어있지 않으면 동기화 x, player object 들을 생성해주며 각 object 의 authority 를 구분하여 관리(서버에서는 모든 object 를 제어할 수 있다.



[그림] 로그인 시스템과 회원가입

게임을 플레이한 이후에 로그인을 통해 서버 DB 와 연동해 플레이어의 정확성 및 콤보 등 결과들을 표시 및 저장해준다. 점수 판정 범위와 콤보를 적용한 점수를 측정해 Unity UI 를 통해 VR 화면에 출력해주며 2 인 플레이를 한 경우 상대방과 점수를 비교해 경쟁할 수 있도록 설계한다.

4.3 VRR Note 자동 생성 Model

VRR 게임에서 원하는 Note 가 없을 때 음원을 입력으로 넣으면 Beat Note Map 이 자동으로 생성되도록 Deep Learning Model 을 설계하였다.

❒ 반영된 요구사항

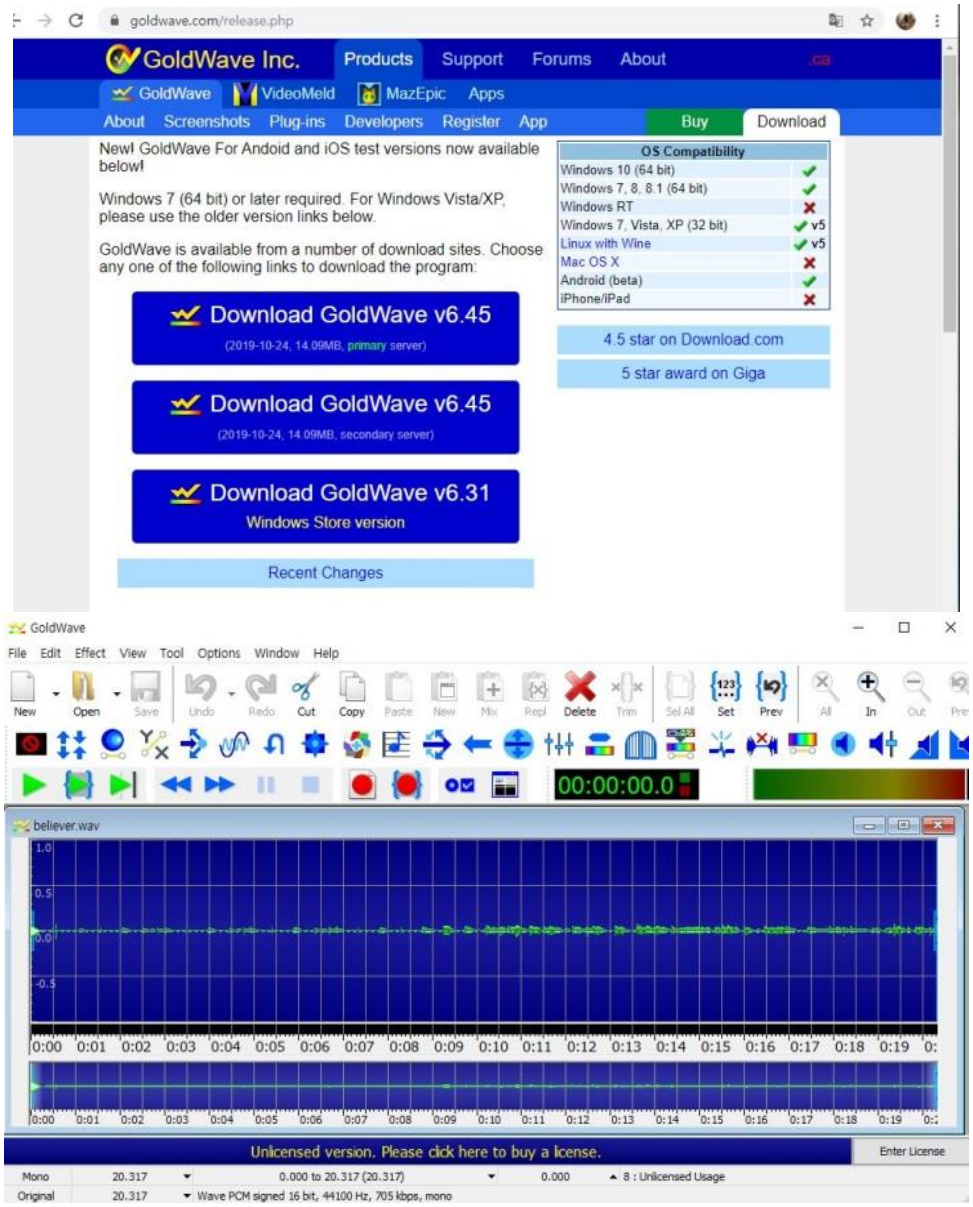
[SFR-V-02] 전처리 2 – STFT 적용 및 스케일링

[SFR-V-06] 최적화 알고리즘

4.3.1 전처리 1 – 음악파일 변환

음악파일을 스테레오 pcm 오디오로 디코딩 하고 두 채널을 평균화하여 단조로운 표현을 만든다. Pcm 을 지원하는 Wav 파일형태로 인코딩, 디코딩한 후 이 때 2 라인(채널)로 나오는 파형의 평균화 하여 매끄러운 파형을 획득한다.

Gold wave 프로그램을 이용하여 44.1khz 의 wav 음악의 인코딩 및 디코딩에 이용한다.

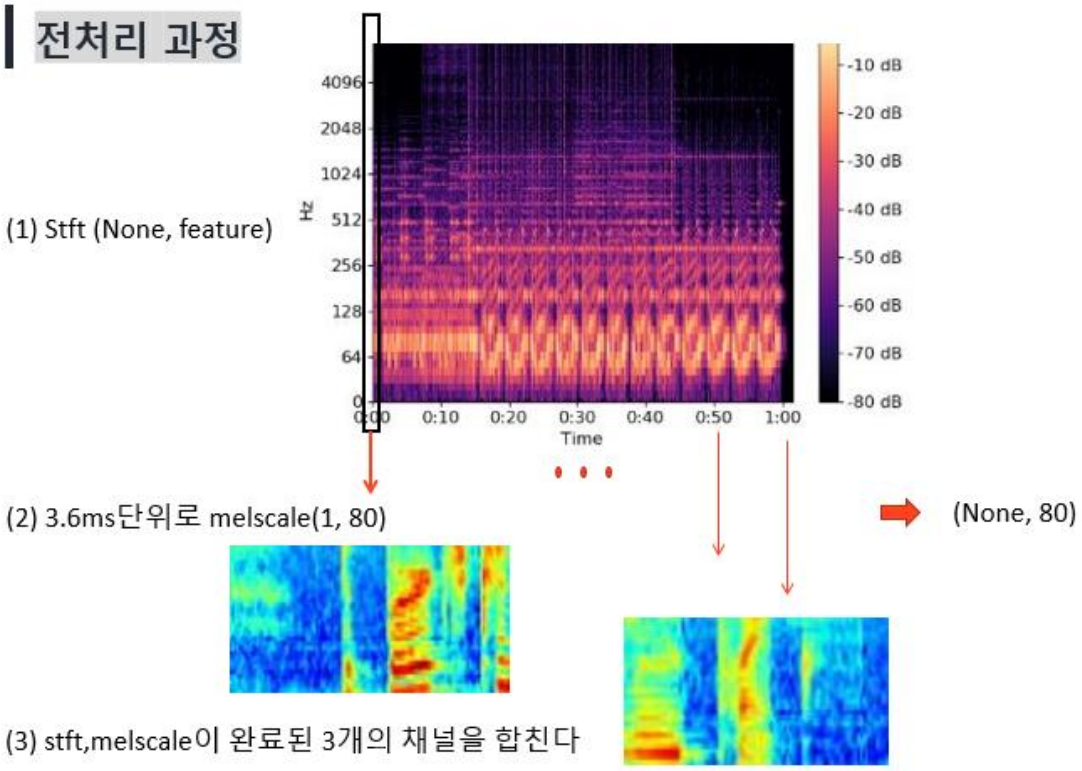


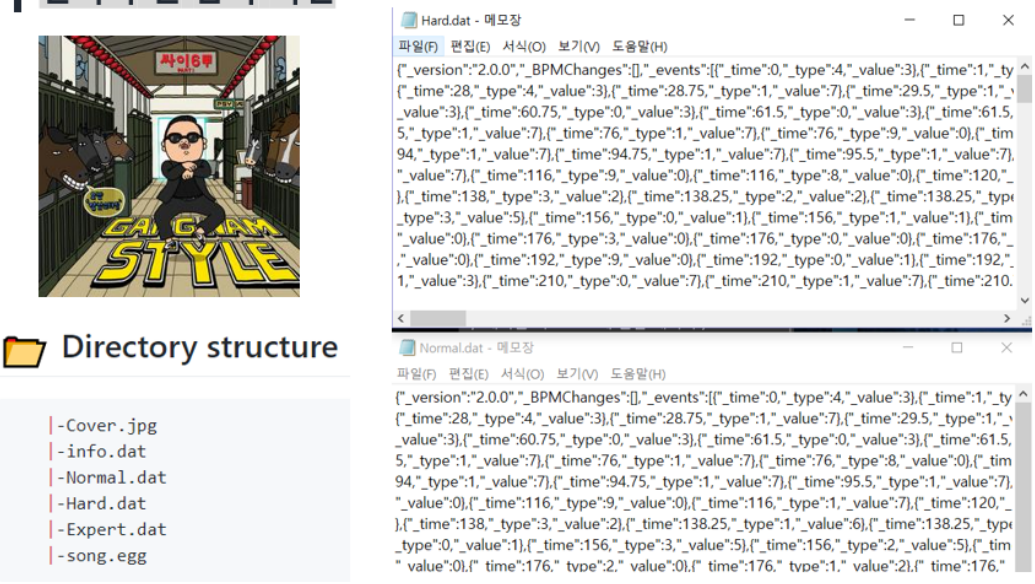
4.3.2 전처리 2 – STFT 적용 및 스케일링

23, 46, 93ms 의 윈도우 길이와 10ms 스텝을 이용하여 다중시간, 단기간 푸리에 변환(STFT)를 계산한다. 단기간 윈도우는 음정 간의 관계 맥락을 제공하고 큰 윈도우는 멜로디와 리듬 같은 높은 수준의 맥락을 제공한다.

STFT 를 적용한 후 2 또는 3 차원의 스펙토그램을 획득한다.

얻게 된 스펙토그램의 데이터의 양이 많기 때문에 mel scale 을 적용해 80 주파수 대역으로 줄이고 로그 스케일링 해준다.





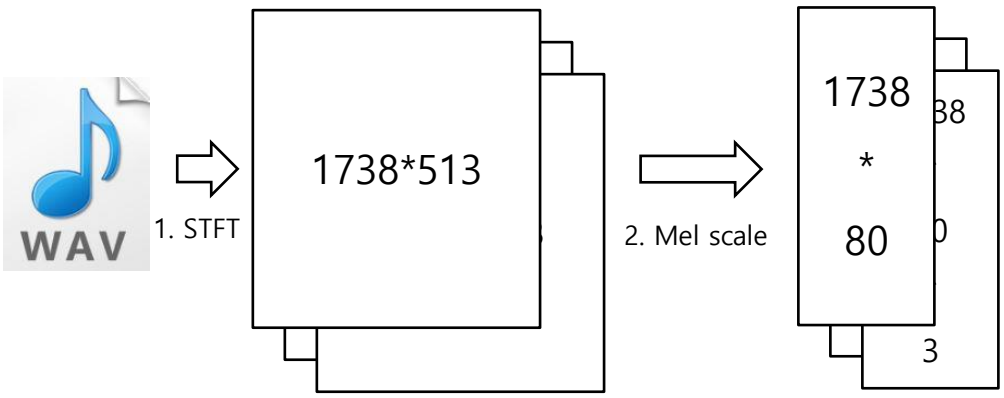
코드 설명

```
class preprocess:
    def __init__(self, music):
        self.music = music

    def stft(self, music, n_fft_size):
        self.music = music
        samples, sample_rate = librosa.load(path= music, sr = 44100, mono = True, duration = 3)
        D = np.abs(librosa.stft(samples, n_fft=n_fft_size,  hop_length=160))
        return D

    def mel_scale(self):
        data = [0]*3
        channel = self.three_channel()
        for k in range(0,len(channel)):
            T = np.transpose(channel[k])
            mfccs = np.matrix(librosa.feature.mfcc(y = T[0], n_mfcc=80))
            mel_scale = np.transpose(mfccs)
            for i in range(1,len(T)):
                mfccs = librosa.feature.mfcc(y = T[i], n_mfcc=80)
                mfccs_t = np.transpose(mfccs)
                mel_scale = np.vstack((mel_scale, mfccs_t))
            arr = np.array(mel_scale)
            data[k] = arr.tolist()
        return data
```

코드 설명

```
def three_channel(self):
    #368, 736, 1488
    music = self.music
    D1 = self.stft(music,368)
    D2 = self.stft(music,736)
    D3 = self.stft(music,1488)
    #return len(D1),len(D2), len(D3)
    return [D1,D2,D3]
```

```
def preprocess_output(self):
    melscale_data = self.mel_scale()
    channel_one = [melscale_data[0][0]]
    channel_two = [melscale_data[1][0]]
    channel_three = [melscale_data[2][0]]

    T = np.vstack((channel_one, channel_two, channel_three))
    tmp = np.vstack((channel_one, channel_two, channel_three))
    new_T = np.vstack(([T],[tmp]))

    for i in range(1,int(len(melscale_data[0])/2)):
        channel_one = [melscale_data[0][i]]
        channel_two = [melscale_data[1][i]]
        channel_three = [melscale_data[2][i]]
        tmp = np.vstack((channel_one, channel_two, channel_three))
        new_T = np.vstack((new_T,[tmp]))
    return new_T
```



1. 1738(time)*513(feature)*3(윈도우개수) (stft 적용)
2. 1738(time)*80(feature)*3((윈도우개수)) (mel scale 적용)

-시간을 20 초로 했을 때 1738 개의 행이 생성됩니다

4.3.3 파형 feature 추출 및 손실 함수 정의

파형 feature 추출

CNN 에서 이용할 수 있도록 파형에서 특색 값을 추출한다.

구간 길이와 파형의 크기에 대한 feature 를 추출하고

시간, 좌우 위치, 상하 위치, 노트 색깔, 자르는 방향의 정보를 가진 노트와 매칭하여 데이터를 얻는다. (이 때 사용하는 노트는 one-hot 인코딩을 하여 사용한다.)

이 후 CNN 에서 feature 추출을 잘하는 layer 를 테스트한 후 적용하고 리듬과 음정의 전후 관계를 파악하기 위해 RNN 을 적용한다.

손실 함수 정의_ CNN, RNN

전체 유닛에서 다중분류, 정확도 향상을 위해 크로스 엔트로피 손실함수를 이용한다.

$$E = -\sum_{k} t_k log y_k$$

4.3.4 활성화 함수, 최적화 알고리즘

활성화 함수 정의

CNN 에서 ReLU 함수, Softmax 함수를 이용하여 3 차원 커널 값을 채널과 주파수 축의 2 차원 값으로 바꾼다. 그 후 RNN 에서 활성화 함수로 하이퍼볼릭 탄젠트를 이용하여 출력층에 들어갈 값을 구한다. 비선형함수를 활성화 함수로 이용하여 층을 구성한다.

$$\begin{aligned} tanh(x) &= 2\sigma(2x) - 1 \\ &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \\ tanh'(x) &= 1 - tanh^2(x) \end{aligned}$$

하이퍼 볼릭 탄젠트 공식

$$f(x) = max(0, x)$$

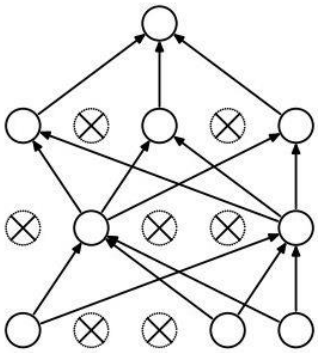
ReLU 함수 공식

$$f(\vec{x})_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \quad \text{for } i = 1, \ldots, K$$

Softmax 함수 공식

최적화 알고리즘, 함수

Overfitting 을 해결하기 위해 dropout 을 사용한다. 기본적으로 드롭아웃의 비율만큼 유닛 및 뉴런에서의 출력을 높이는 방법을 이용하거나 가중치 규제를 이용하여 해결할 것이다
Underfitting 을 해결하기 위해 튜닝과정을 거칠 것이고 여기서 적절한 함수를 찾고, 손실값 그래프를 시뮬레이션하여 최대 퍼포먼스 지점의 에포크 지점을 찾고 훈련의 조기 종료 등의 방법을 적용한다.
또한 dropout 의 비율, 시점, 빈도 등을 고려하여 최적의 퍼포먼스를 기대한다.

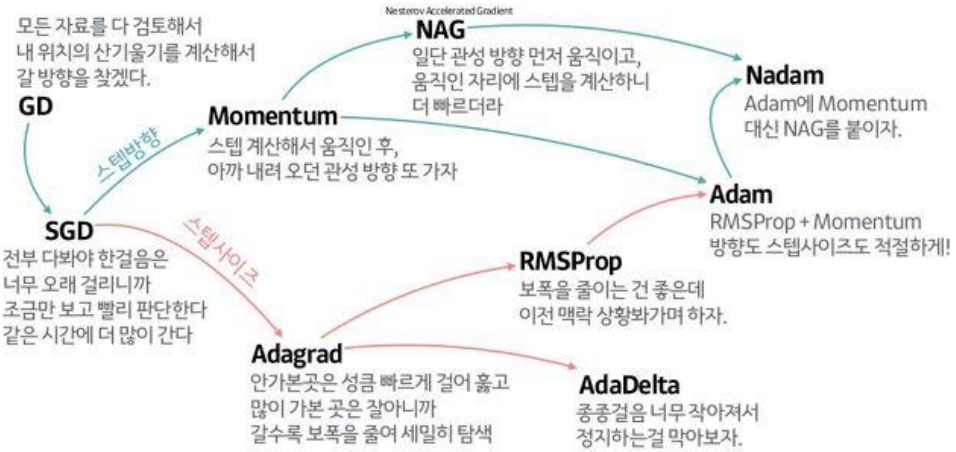


드롭아웃 방식

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

L2 규제(Lidge regression) 공식

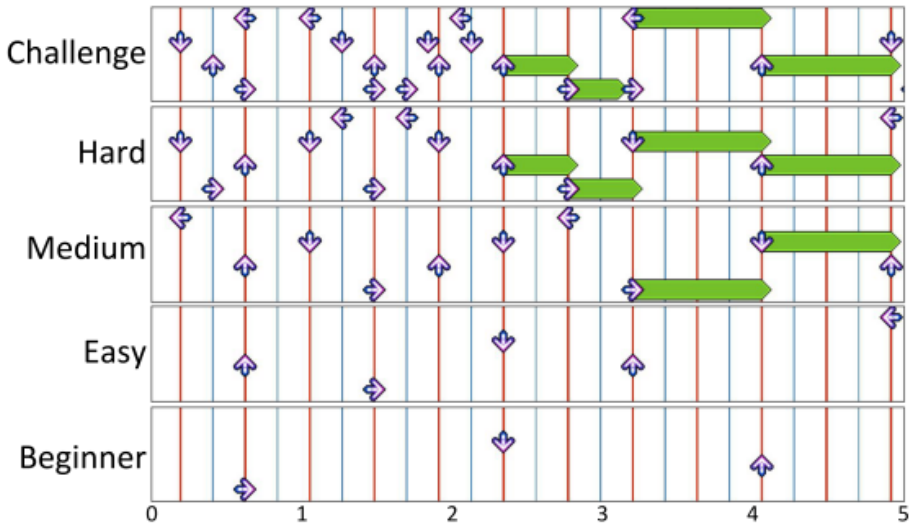
학습 속도를 빠르고 안정적으로 만들기 위해 optimizer 를 변경해 가며 가장 높은 정확도가 나오는 optimizer 를 적용할 것이다.



4.3.5 DataSet Mirroring

Chris Donahue, Zachary C. Lipton ,Julian McAuley Dance Dance Convolution.
In ICML, 2017.
에 따르면, 학습에 필요한 데이터양을 늘리기 위해 왼쪽/오른쪽, 위쪽/아래쪽을 미러링하여 데이터를 2 배로 늘렸다. 이러한 dataset mirroring 으로 성능이 상당히 향상되었다.

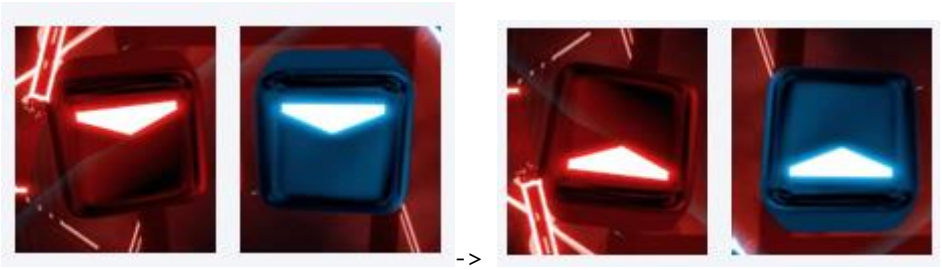
예를 들어, 왼쪽은 -> 오른쪽으로 바꾸고, 위쪽은 아래쪽으로 바꾸는 식으로 dataset mirroring 하여 같은 step 위치에 방향만 거꾸로 하여 새로운 데이터를 만들었다. 이는 level 별로 (Beginner,Easy, Medium,Hard,Challenge) 각각 2 배로 증가시켰다.



– Chris Donahue, Zachary C. Lipton, Julian McAuley(2017)

따라서, 본 프로젝트에서도 부족한 dataset 을 늘리기 위해 동일한 왼쪽, 오른쪽, 위쪽, 아래쪽을 가진 beat note 에도 mirroring 을 적용하여 dataset 을 2 배로 늘릴 것이다.

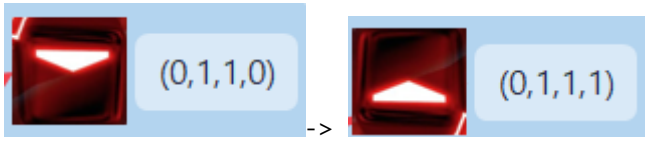
<왼쪽, 오른쪽, 위쪽, 아래쪽 예시>



Dataset mirroring 의 예시: <아래쪽, 아래쪽> 이 <위쪽, 위쪽>으로 변환된다.

구현 방법:

(색깔, 행 위치, 열 위치, 자르는 방향)으로 구성된 beat map 의 txt 파일에서



아래 예시와 같이 마지막 인덱스인 '자르는 방향' 인덱스 값만 0 에서 1 로 바꿔준다

4.3.6 CNN_LSTM 모델 선정 이유

CNN 모델

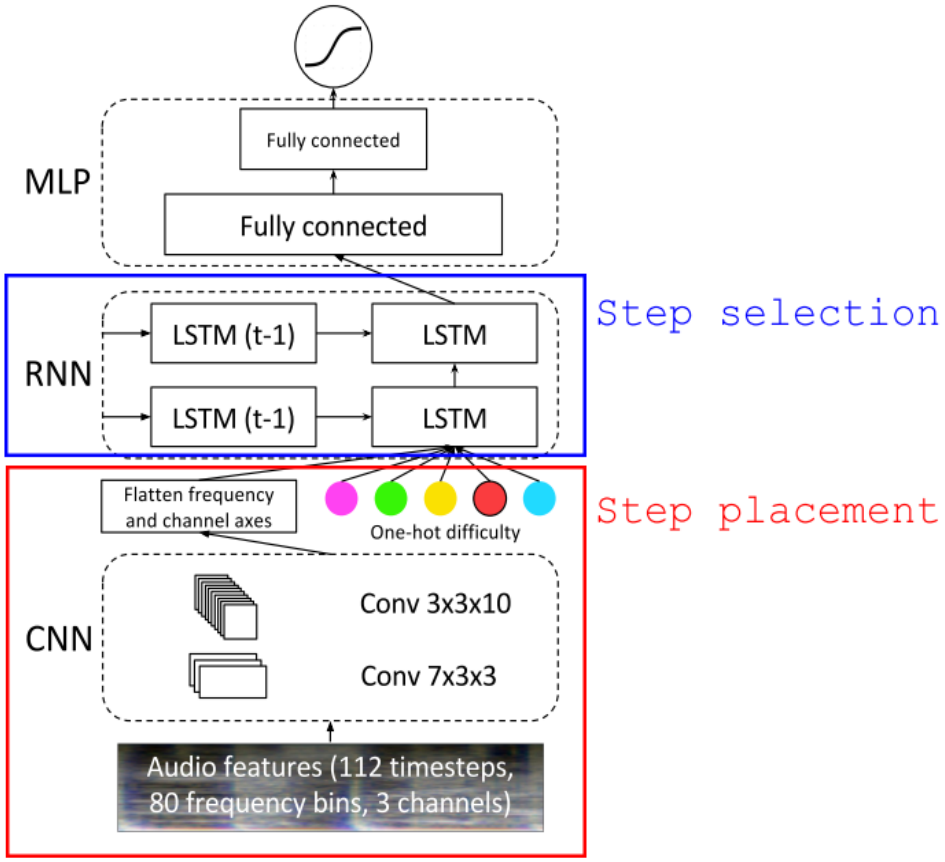
<CNN 모델 선택 이유>
Schl¨uter, Jan and B¨ock, Sebastian. Improved musical on-
set detection with convolutional neural networks.
In ICASSP, 2014.
에 따르면, On-set detection 은 음이 변하는 특징구간을 감지하는 것으로, 오디오파일에 맞춰 비트 매핑을 하기 위해 필요하다. 전처리 된 오디오파일에서 on-set detection 을 하는데, CNN 을 사용한 모델이 가장 높은 성능을 보였다.

	Precision	Recall	F-score
RNN [10, 5]	0.892	0.855	0.873
CNN [1]	0.905	0.866	0.885
+ Dropout	0.909	0.871	0.890
+ Fuzziness	0.914	0.885	0.899
+ ReLU	**0.917**	**0.889**	**0.903**
SuperFlux [5]	0.883	0.793	0.836

- Schl ¨uter & B ¨ock(2014)

리듬에 따라 배치된 박스를 'beat note'라 하는데, 음악마다 'beat note'의 지도를 만들 때 필요한 기능을 2 가지로 나눴다.
Step placement 와 step selection 이다.
Step placement 는 언제 beat note 가 올지 시간을 정하는 것이고,
Step selection 은 어떤 방향의 beat note 가 올지 방향을 결정하는 것이다.

아래 그림 <C-LSTM 의 전체 구조>과 같이
Step placement 는 CNN 으로 구성하고, step selection 은 LSTM 모델로 구현한다.
CNN 과 LSTM 은 서로 연결된 형태인데, CNN 의 output 이 LSTM 의 input 으로 구성된다.



<C-LSTM 의 전체 구조>

Step placement 을 위한 CNN 의 input, layer 구성, output 은 다음과 같이 구성된다.

Input: n 개의 timesteps x 80 개의 주파수 x 3 개의 채널 100 개의 timestep 만큼 한꺼번에 CNN 에 집어넣는다.

(-Timesteps 는 음원 파일의 길이에 따라 다르다. 예를 들어, 200 초의 음원파일을 20 초단위로 쪼개서 10 개의 토큰으로 나눈다.10 개의 토큰은 순차적으로 1 개씩 들어간다. 1 개의 토큰을 timestep 으로 나타내면, 대략, 1738 개가 된다. n=1738 timsteps 가 된다.

- 3 개의 채널은 학습단위가 된다. 23 은 20, 3 / 46 은 20,20,6 / 93 은 20,20,20,20,13 단위로 쪼개져서 학습하려 했지만, 단위가 20 이므로 그 안에서 나머지 값인 3,6,13 을 쪼개기 어려우므로, 23 단위로 쪼갠다. 23 은 23, 46 은 23,23 93 은 23 23 23 23 단위로 쪼개어 학습하도록 한다.

따라서, 23 은 23 토큰이 1 번 ,46 은 2 번 , 93 은 4 번 연속으로 학습한 뒤

lstm 의 parameter 값을 업데이트한다.)

layer1 은 7 wide 의 시간 x 3 wide 의 주파수 x3 개의 channel = 7x3 의 filter kernel 10 개로 구성되고,

layer2 는 3x3x10 의 filter kernel 20 개로 구성된다.

매 filter kernel 이후 3x3 max pooling 을 거친다.

```
# input place holders
X = tf.placeholder(tf.float32, [None, 100, 80, 3])
Y = tf.placeholder(tf.float32, [None, 100])

# Layer 1 input shape=(?, 100, 80, 3)
W1 = tf.Variable(tf.random_normal([7, 3, 3, 10], stddev=0.01))
L1 = tf.nn.conv2d(X, W1, strides=[1, 1, 1, 1], padding='SAME')
# Conv (?, 100, 78, 10)
L1 = tf.nn.relu(L1)
L1 = tf.nn.max_pool(L1, ksize=[1, 1, 3, 1], strides=[1, 1, 3, 1], padding='VALID')
# Pool (?, 100, 26, 10)

# Layer 2 input shape=(?, 100, 26, 10)
W2 = tf.Variable(tf.random_normal([3, 3, 10, 20], stddev=0.01))
L2 = tf.nn.conv2d(L1, W2, strides=[1, 1, 1, 1], padding='SAME')
# Conv  (?, 100, 24, 20)
L2 = tf.nn.relu(L2)
L2 = tf.nn.max_pool(L2, ksize=[1, 1, 3, 1], strides=[1, 1, 3, 1], padding='VALID')
# Pool  (?, 100, 8, 20)

# Flattening (except for time-axis)
L2_flat = tf.reshape(L2, [-1, 100, 8 * 20])
print('L2_flat after flattening: {}'.format(L2_flat))
```

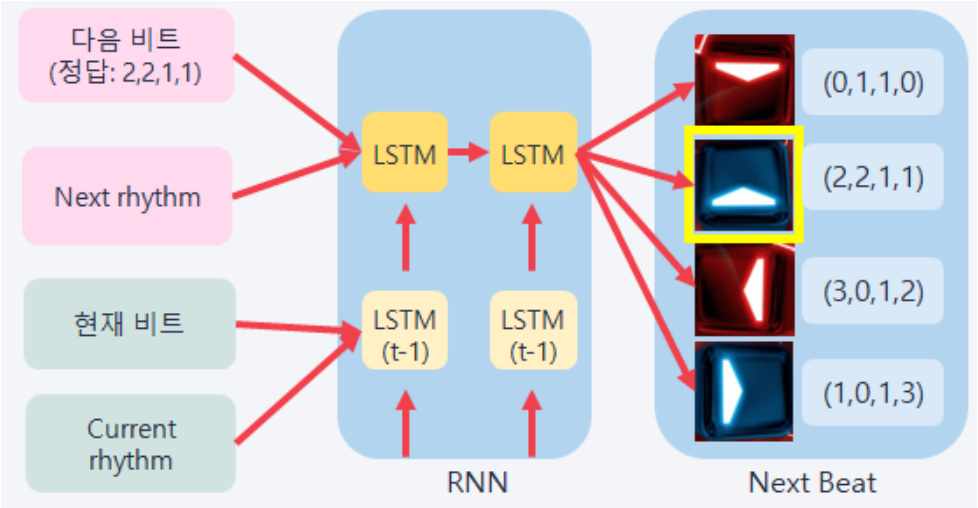
Output: 하나의 sigmoid function 으로 구성되어 구간에 beat note 가 들어가는지 아닌지를 판별한 결과값이 산출된다.
이 결과값은 LSTM 에 넣기 위해 시간 축은 보존한 상태에서 flattening 을 한다.
역할: max pooling 은 frequency 에서만 실행한다.

LSTM(RNN) 모델

<RNN 모델 선택이유>
Chris Donahue, Zachary C. Lipton, Julian McAuley Dance Dance Convolution.
In ICML, 2017.
에 따르면, CNN 으로만 구현한 모델은 오디오파일의 시간에 따른 리듬의 변화 패턴인 rhythmic feats 을 반영하는데 한계가 있음을 지적하여 CNN 과 LSTM (RNN)을 섞은
C-LSTM 모델을 제안한다.

-Chris Donahue, Zachary C. Lipton, Julian McAuley (2017)
LSTM 은 전처리 과정에서 얻은 93ms(Rhythmic feats), 46ms(Rhythmic feats &Step feats), 23(Step feats)의 특징들을 전부 고려해서 다음 비트를 매핑하기 위해 필요하다.

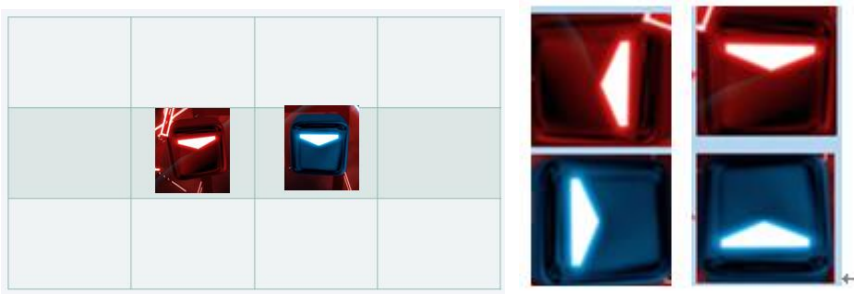
Beat note mapping 의 2 가지 과정 Step placement 와 Step selection 중
LSTM 은 Step Selection 을 위해 사용한다.



LSTM 의 역할,구성, input, output 의 개념적 설계는 다음과 같다. 상세설계는 SFR-V-08-2 에 기술하였다.
역할: LSTM 은 다음 노트가 어떤 방향(상하좌우)의 노트가 올지를 정하는 것이다.
구성: 128 개의 node 로 이뤄진 2 layer LSTM
Input : CNN 의 output 인 100 개 의 time axis output 을 LSTM 의 각 time step input 으로 넣는다.
학습 방식: 매 time step 마다 다음 step 이 어떤 방향의 노트인지를 예측한다.
LSTM 의 학습방식은 Supervised learning 방식으로 학습 시에는 미리 매핑 되어있는 beat saber 의 beat note map 을 답안지로 하여 답을 예측하고 예측 값을 조절해 나간다.
Output: 예측한 방향(왼쪽,오른쪽,아래쪽,위쪽 중 하나)의 노트

LSTM input 및 output

LSTM 모델의 추가적인 musical context 을 주기 위해 rhythmic feature 를 추가적인 input 으로 넣어준다.
Step placement 사이의 간격이 균일(uniform)하지 않기 때문에 다음 3 가지 요소를 input 에 추가해준다.
Δ-time : 지난번 note 으로부터의 시간 / 다음 note 까지의 시간 -> 2 개의 feature
Δ-beat : 지난번 note 으로부터의 비트 갯수 / 다음 note 까지의 비트 갯수 -> 2 개의 feature
beat phase : 이 note 에서 가장 가까운 note (각 beat 를 16 분 음표단위로 나눈 note subdivision 들 중)
input 은 총 길이 56 벡터가 들어가게 된다.
이번 노트의 12 position * 4 direction 의 multi class encoding (48 개)



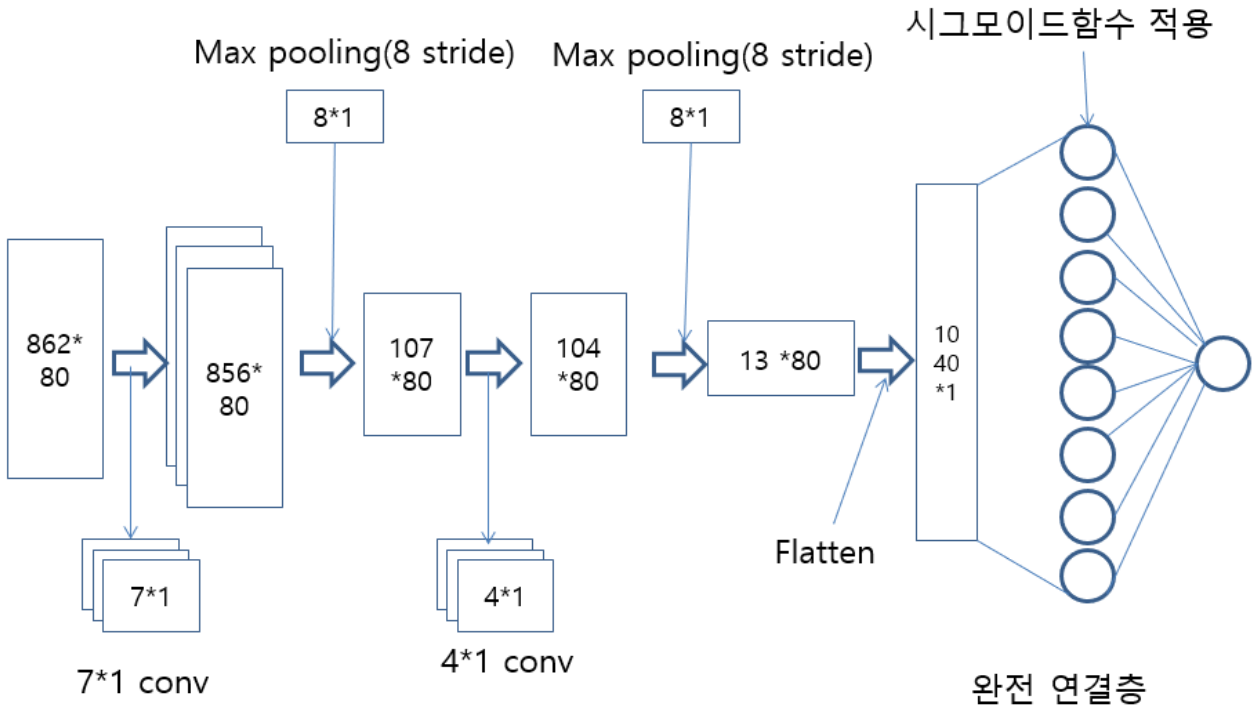
<4x3 의 12 개 position> 과 <왼,오른,위,아래 의 4 direction>
전 이전 노트와의 시간차 Δ-time (1 개) + 다음 노트와의 시간차 Δ-time (1 개)
지난번 노트로부터의 비트 갯수 Δ-beat (1 개) + 다음 노트 까지의 비트 갯수 Δ-beat (1 개)
노트가 속한 4 분음표 비트내에 있는 4 개의 16 분음표 중 가장 가까운 비트 phase 표시 (4 개)
따라서, 48*4*4 =768 개의 가능한 노트 조합이 나온다.
결과적으로, Output 은 길이 768 의 벡터가 된다.

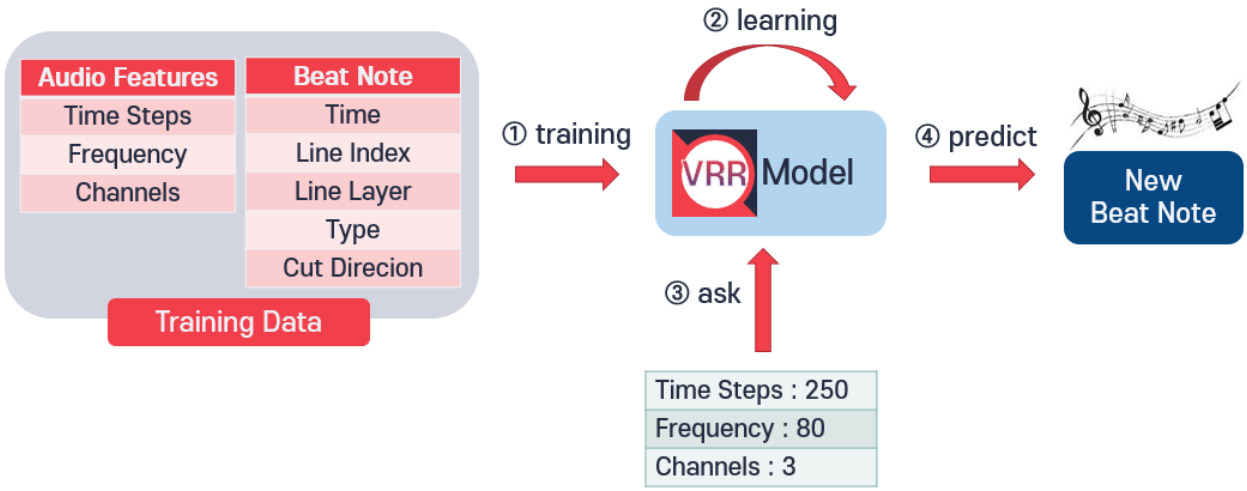
4.3.7 CNN_LSTM 모델 1

입력 데이터 량 –> mel scale 이용(513 개의 feature 값들 중에 특색을 가장 잘 보여주는 값들로 추출한다.)_실습은 80 개 feature 로 추출 또한 시간 간격을 절반으로 줄여 데이터 량을 줄임과 동시에 정확도도 높인다. (기존 20 초에서 10 초로 변경)

타깃 값의 시간 간격이 너무 길어 delay 가 생길 수 있다. -> 기존 0.5 초 간격의 타깃 데이터 간격을 10ms 으로 줄였다

1 채널 CNN



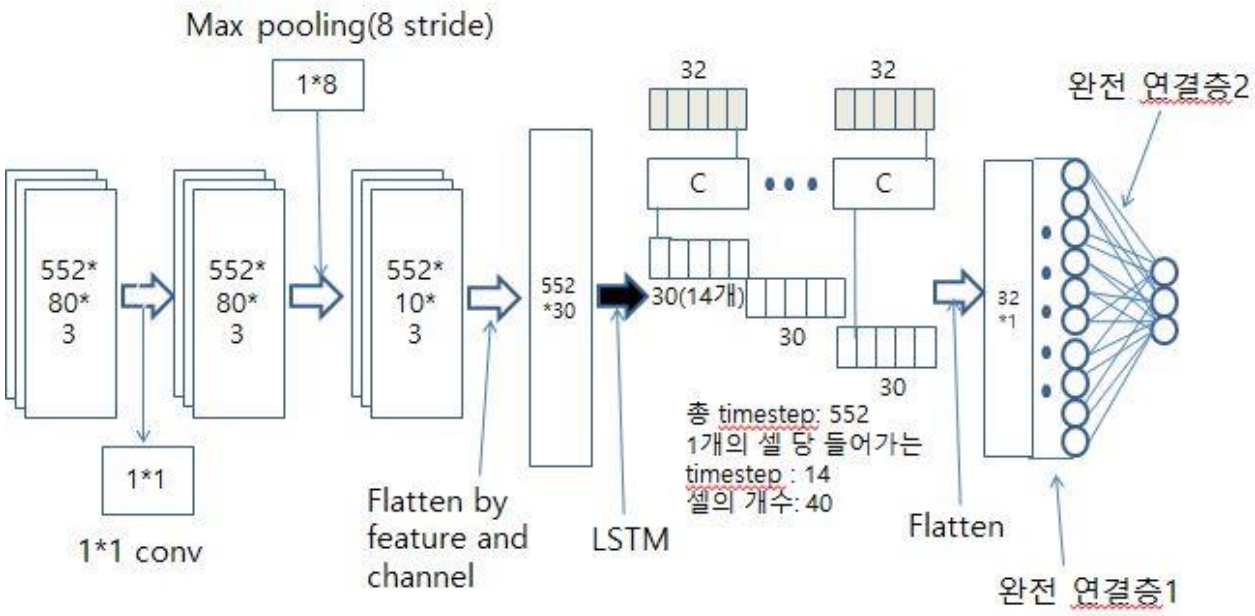


[그림] 초기 모델 설계

변경 사항

3 채널 전처리, sampling, lstm 추가, 입력 시간 단축

3 채널 CNN_LSTM 모델 1



Feature 위주의 conv, maxpooling 적용

Layer (type)	Output Shape	Param #
conv2d_64 (Conv2D)	(None, 80, 3, 1)	2
max_pooling2d_63 (MaxPooling	(None, 10, 3, 1)	0
flatten_56 (Flatten)	(None, 30)	0
reshape_36 (Reshape)	(None, 1, 30)	0
lstm_30 (LSTM)	(None, 1, 32)	8064
dropout_8 (Dropout)	(None, 1, 32)	0
flatten_57 (Flatten)	(None, 32)	0
dense_43 (Dense)	(None, 32)	1056
dense_44 (Dense)	(None, 3)	99

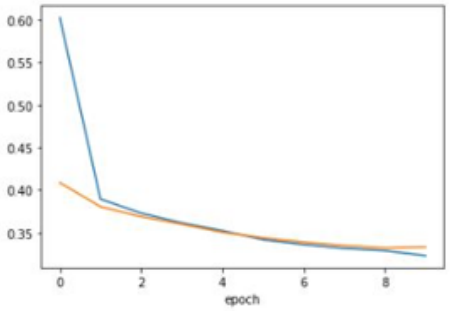
Total params: 9,221
Trainable params: 9,221
Non-trainable params: 0

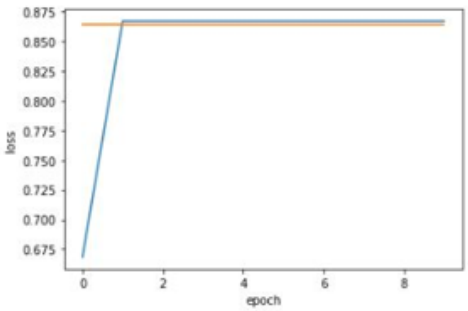
코드 설명

```
from sklearn.model_selection import train_test_split

from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import LSTM
from keras.layers import Dense, Dropout, Activation, Flatten

model = Sequential()
model.add(Conv2D(1,(1,1), activation= 'relu', padding = 'same', input_shape =(80,3,1)))
model.add(MaxPooling2D((8,1)))
model.add(Flatten())
model.add(Reshape((1,30)))
model.add(LSTM(32,batch_input_shape=(None,14,3),return_sequences=True))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(32, activation='sigmoid'))
model.add(Dense(3, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.build((None,80,3,1))
model.summary()
```

실행 결과

```
552/552 [==============================] - 2s 3ms/step - loss: 0.6027 - acc: 0.6685 - val_loss: 0.4086 - val_acc: 0.8647
Epoch 2/10
552/552 [==============================] - 0s 822us/step - loss: 0.3893 - acc: 0.8671 - val_loss: 0.3804 - val_acc: 0.8647
Epoch 3/10
552/552 [==============================] - 0s 733us/step - loss: 0.3728 - acc: 0.8671 - val_loss: 0.3687 - val_acc: 0.8647
Epoch 4/10
552/552 [==============================] - 0s 816us/step - loss: 0.3613 - acc: 0.8671 - val_loss: 0.3598 - val_acc: 0.8647
Epoch 5/10
552/552 [==============================] - 0s 772us/step - loss: 0.3522 - acc: 0.8671 - val_loss: 0.3503 - val_acc: 0.8647
Epoch 6/10
552/552 [==============================] - 0s 760us/step - loss: 0.3415 - acc: 0.8671 - val_loss: 0.3438 - val_acc: 0.8647
Epoch 7/10
552/552 [==============================] - 0s 741us/step - loss: 0.3359 - acc: 0.8671 - val_loss: 0.3386 - val_acc: 0.8647
Epoch 8/10
552/552 [==============================] - 0s 750us/step - loss: 0.3313 - acc: 0.8671 - val_loss: 0.3347 - val_acc: 0.8647
Epoch 9/10
552/552 [==============================] - 0s 742us/step - loss: 0.3287 - acc: 0.8671 - val_loss: 0.3318 - val_acc: 0.8647
Epoch 10/10
552/552 [==============================] - 0s 750us/step - loss: 0.3228 - acc: 0.8671 - val_loss: 0.3330 - val_acc: 0.8647
```



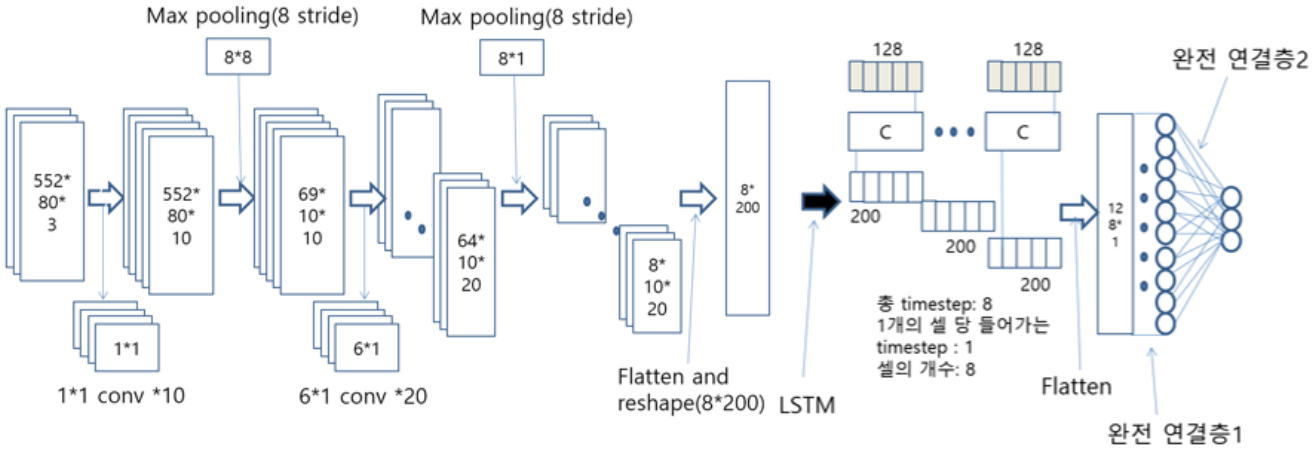


4.3.8 CNN_LSTM 모델 2

변경 사항

3 채널 전처리, sampling, lstm 추가, 입력 시간 단축

3 채널 CNN_LSTM 모델 2



Time_axis 위주의 conv, maxpooling 적용

Layer (type)	Output Shape	Param #
conv2d_59 (Conv2D)	(None, 552, 80, 10)	40
max_pooling2d_58 (MaxPooling	(None, 69, 10, 10)	0
conv2d_60 (Conv2D)	(None, 69, 10, 20)	1220
max_pooling2d_59 (MaxPooling	(None, 8, 10, 20)	0
flatten_49 (Flatten)	(None, 1600)	0
reshape_32 (Reshape)	(None, 8, 200)	0
lstm_27 (LSTM)	(None, 8, 128)	168448
flatten_50 (Flatten)	(None, 1024)	0
dense_37 (Dense)	(None, 128)	131200
dense_38 (Dense)	(None, 3)	387

Total params: 301,295
Trainable params: 301,295
Non-trainable params: 0

코드 설명

```
from sklearn.model_selection import train_test_split

from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import LSTM
from keras.layers import Dense, Dropout, Activation, Flatten

model2 = Sequential()
model2.add(Conv2D(10,(1,1), activation= 'relu', padding = 'same', input_shape =(552,80,3)))
model2.add(MaxPooling2D((8,8)))
model2.add(Conv2D(20,(6,1), activation= 'relu', padding = 'same', input_shape =(64,10,20)))
model2.add(MaxPooling2D((8,1)))
model2.add(Flatten())
model2.add(Reshape((8,200)))

model2.add(LSTM(128,batch_input_shape=(None,2,3),return_sequences=True))
model2.add(Flatten())
model2.add(Dense(128, activation='sigmoid'))
model2.add(Dense(3, activation='sigmoid'))
model2.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model2.build((552,80,3))
model2.summary()
```

4.4 Web 플랫폼

4.4.1 Sequence Diagram

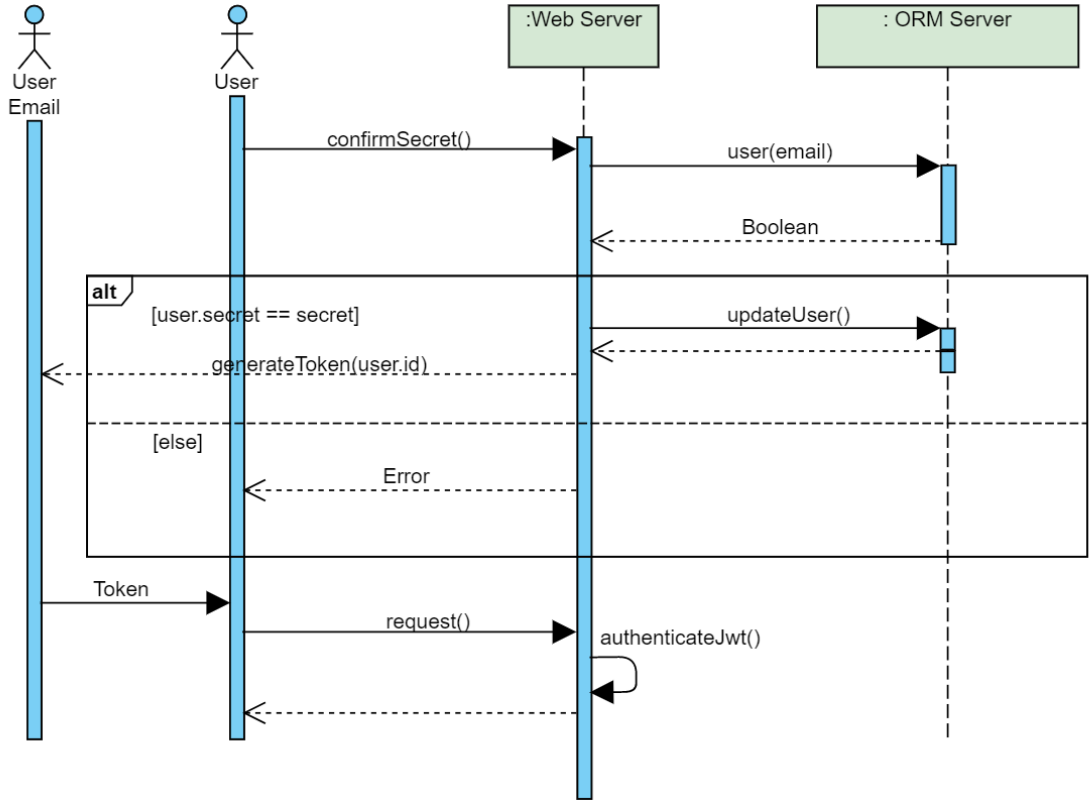
4.4.1.1 LogIn

❒ 반영된 요구사항

SFR-I-01 로그인 기능 제공

SFR-I-02 로그아웃 기능 제공

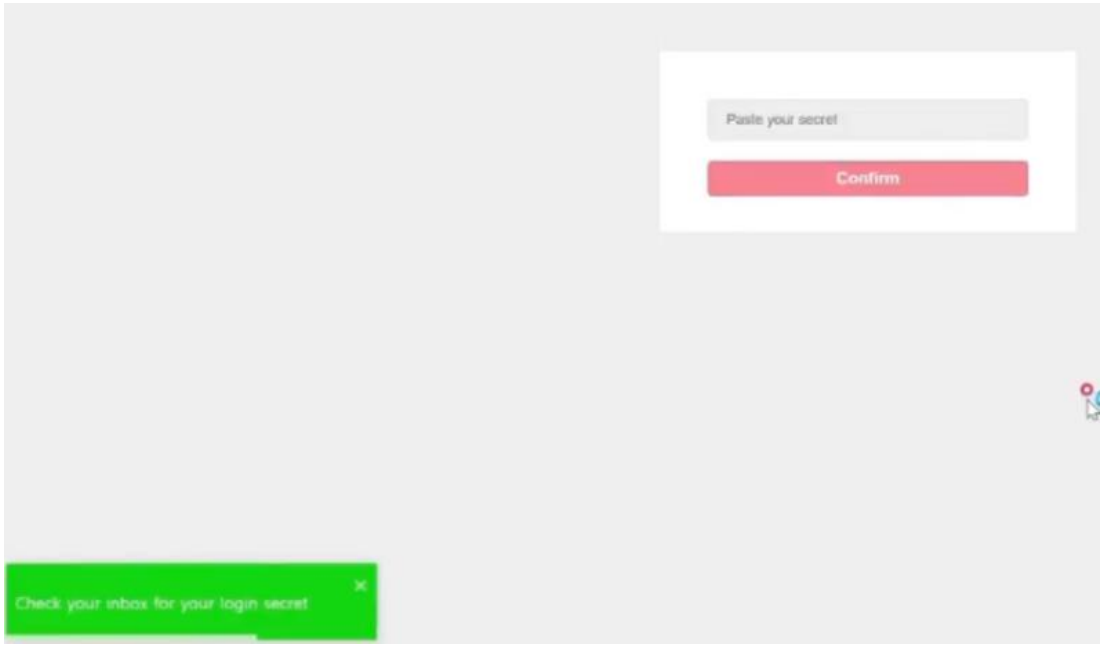
SFR-I-03 회원가입 기능 제공



받는 사람: hochan049@gmail.com

Hello! Your login secret is **best mom**.

Copy paste on the app/website login



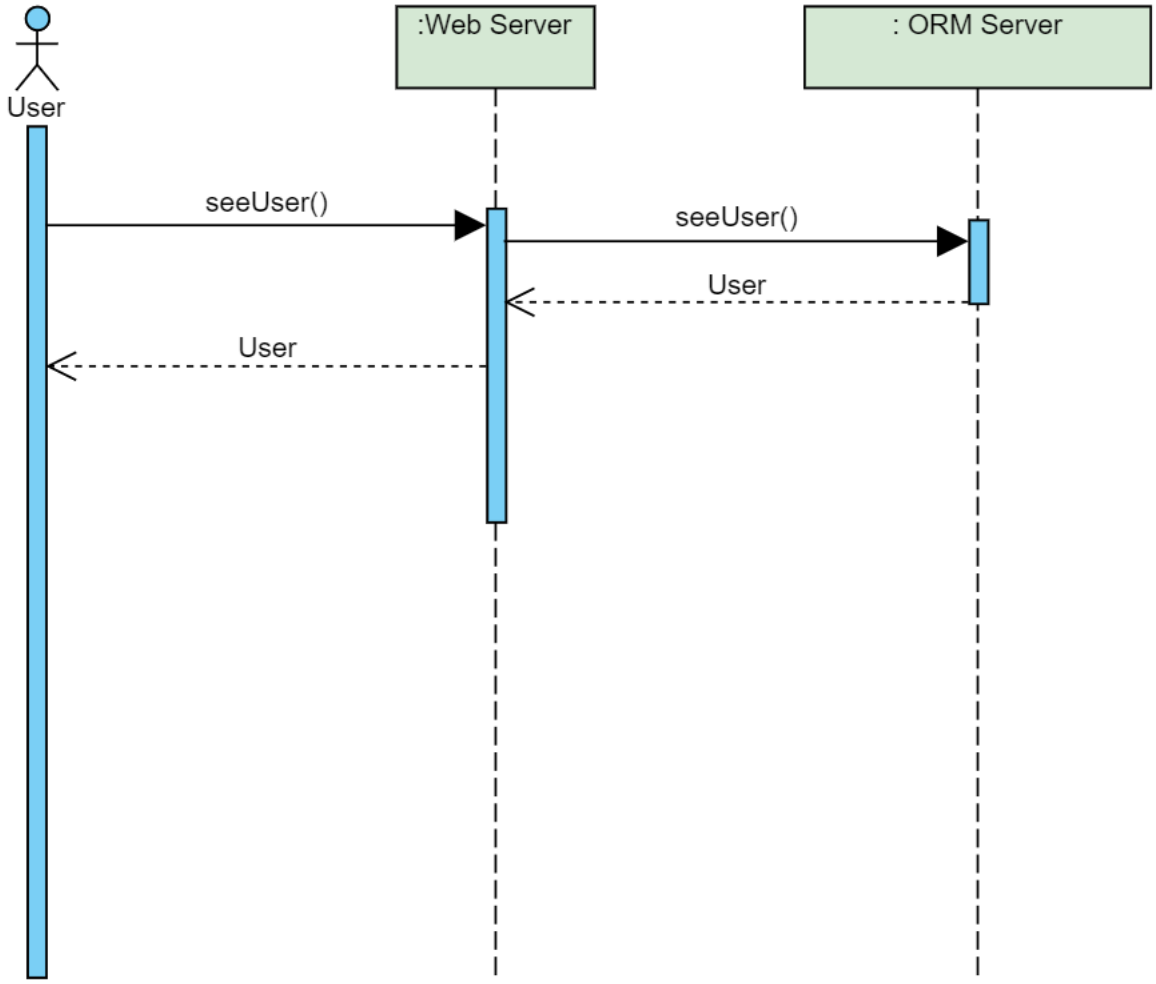
4.4.1.2 사용자 정보

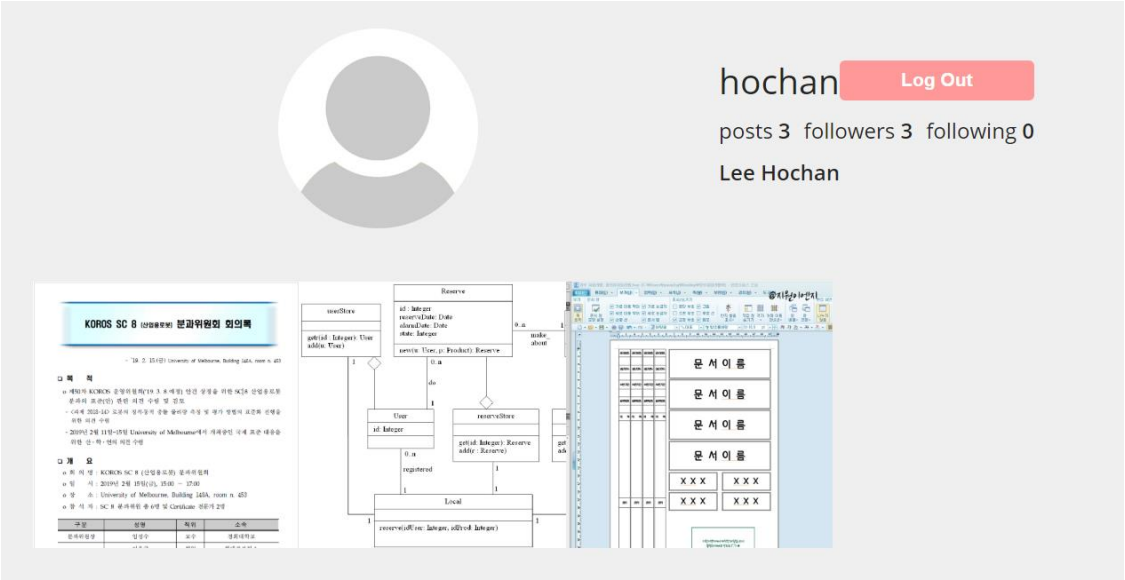
❒ 반영된 요구사항

SFR-I-07 개인정보 확인 기능 제공

SFR-I-08 탈퇴 기능 제공

SFR-I-09 개인정보 변경 기능 제공

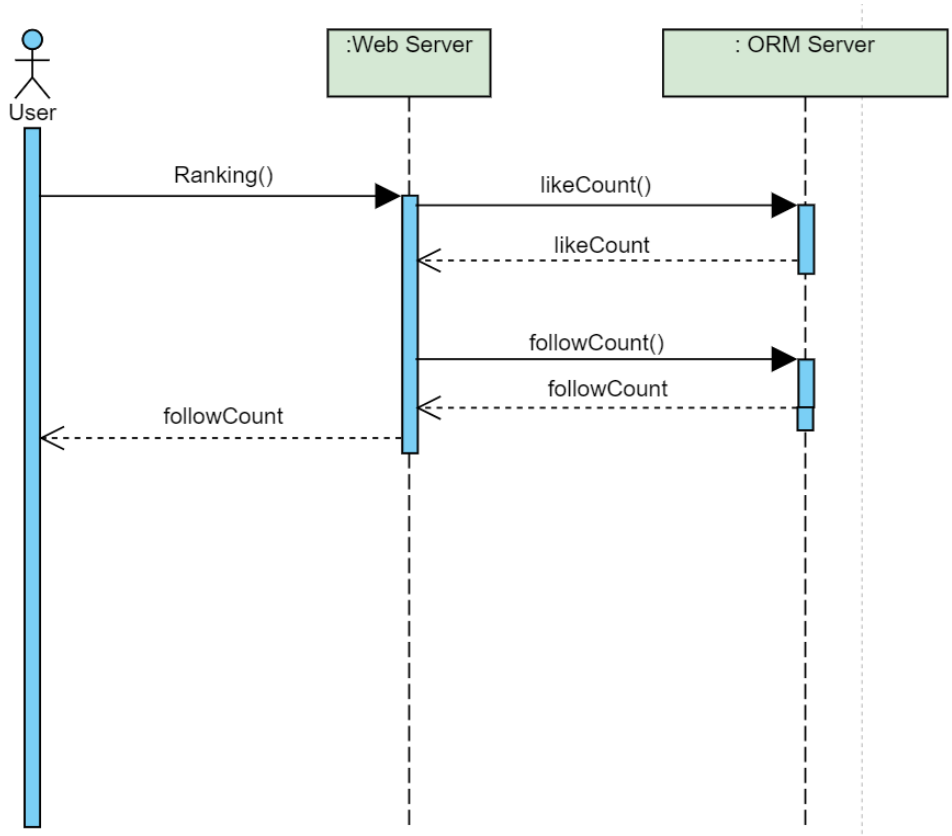




4.4.1.3 Play 랭킹

❒ 반영된 요구사항

SFR-I-05 음악 노트 게시물 댓글, 좋아요, 파일, 이미지 보기 기능 제공



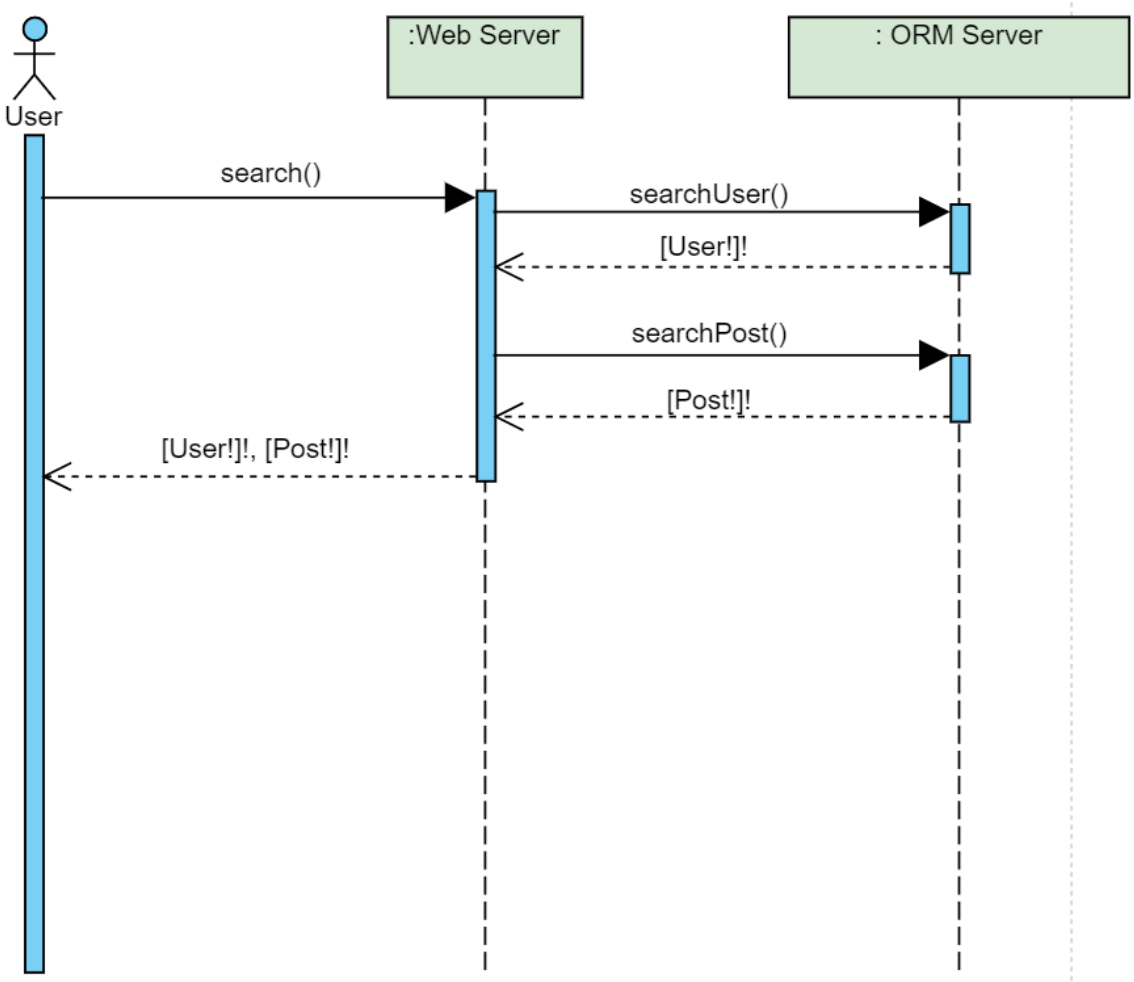
4.4.1.4 음악 노트 & 사용자 검색

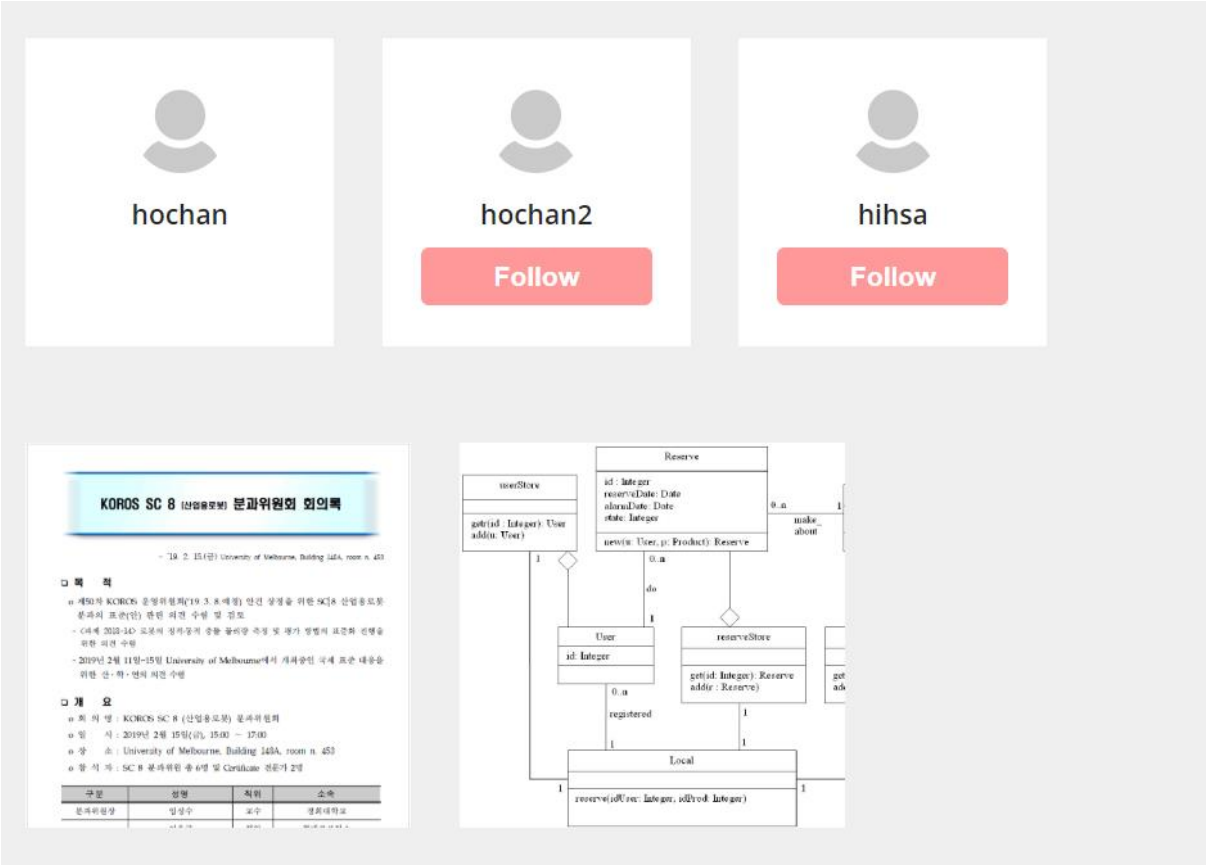
❒ 반영된 요구사항

SFR-I-04 전체 음악 노트 게시물 확인 기능

SFR-I-06 음악 노트 게시물 및 사용자 검색 기능 제공

SFR-I-12 음악 노트 공유 기능 제공

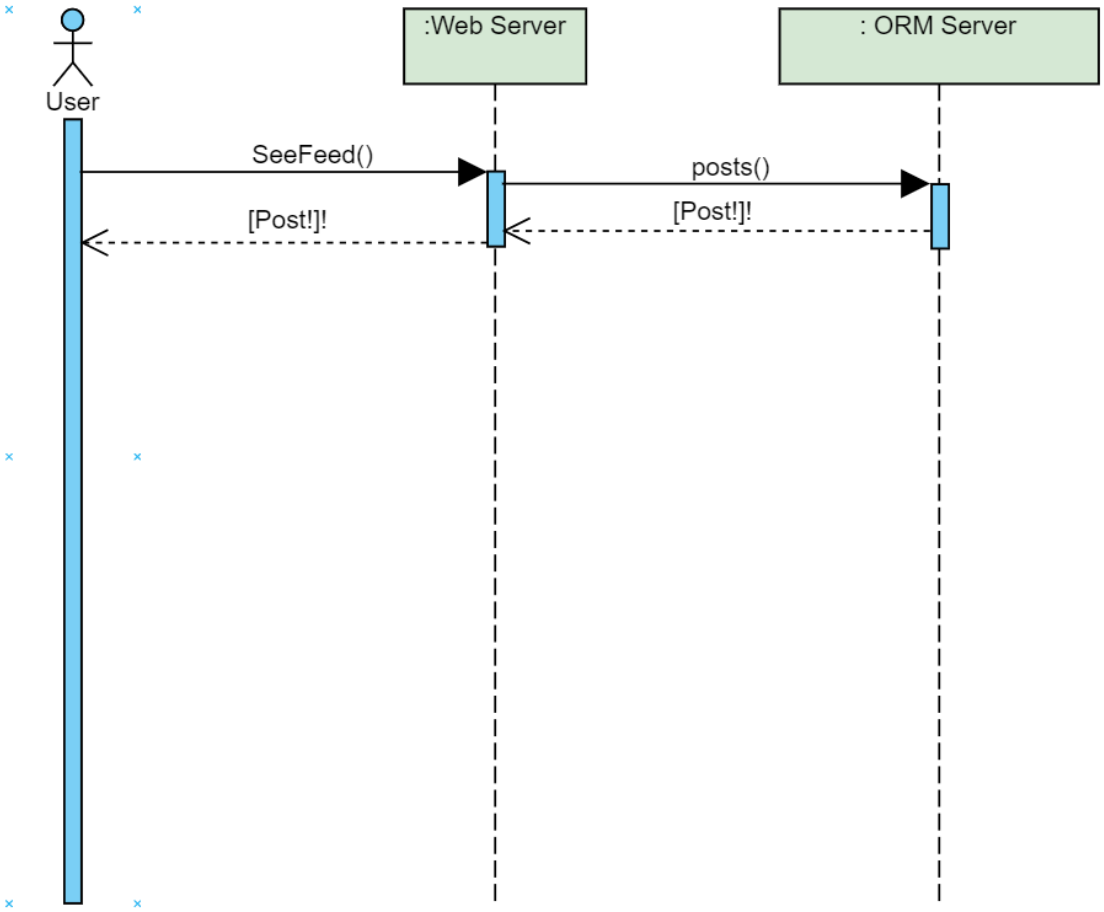



hochan
hochan2
Follow
hihsa
Follow
KOROS SC 8 (산업용로봇) 분과위원회 회의록
Reserve
id : Integer
reserveDate: Date
alarmDate: Date
state: Integer
new(u: User, p: Product): Reserve
userStore
get(id : Integer): User
add(u: User)
User
id: Integer
reserveStore
get(id: Integer): Reserve
add(r : Reserve)
registered
Local
reserve(idUser: Integer, idProd: Integer)

4.4.1.5 피드 보기

❒ 반영된 요구사항

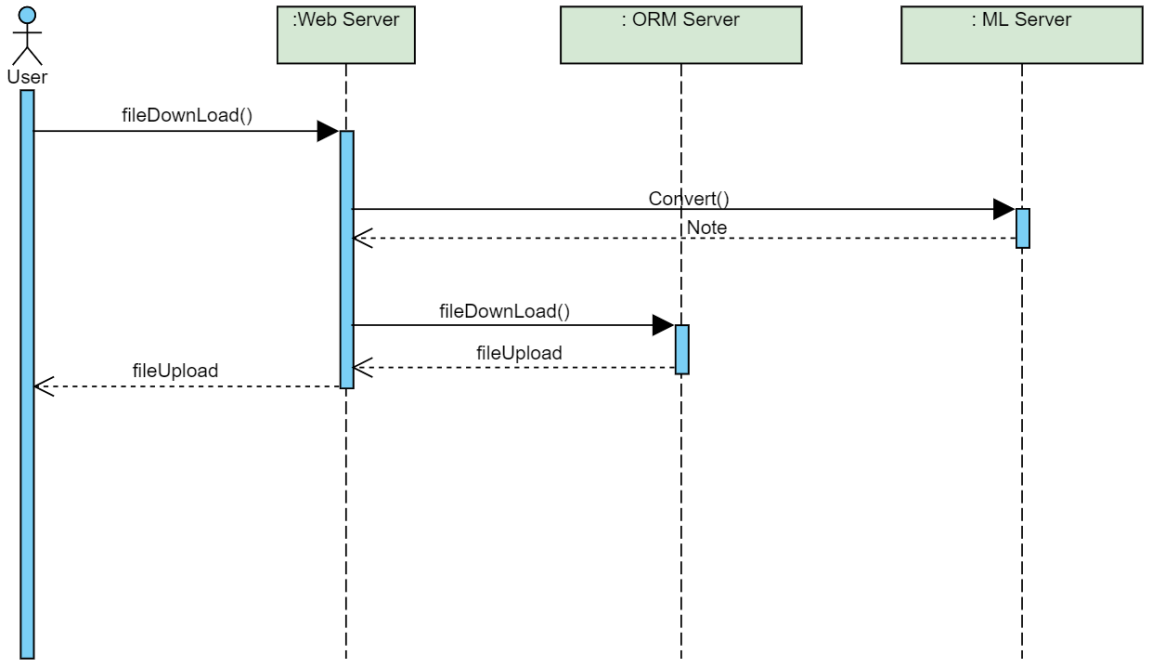
SFR-I-14 페이지 렌더링 대기상태 시 보여지는 애니메이션 제공



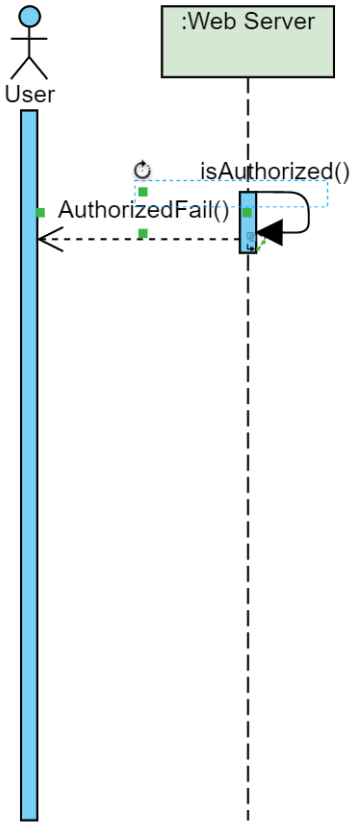
4.4.1.6 노트 변환 기능

❒ 반영된 요구사항

SFR-I-13 음악 노트로 변환 기능 제공

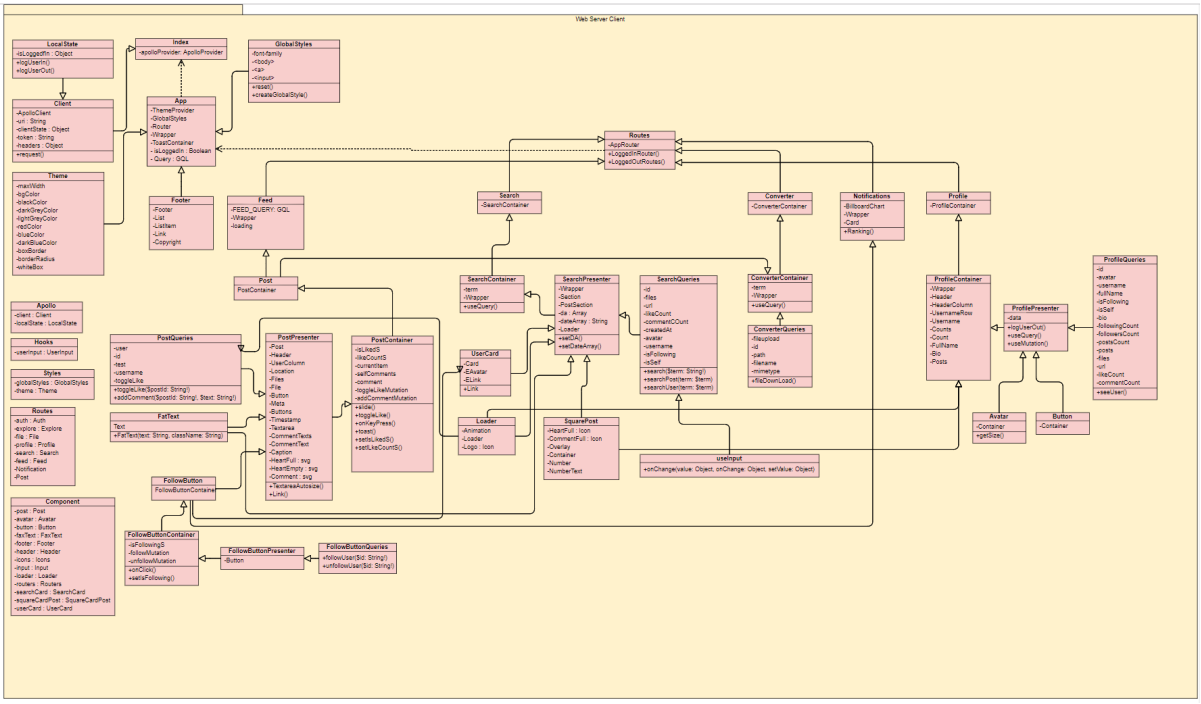


4.4.1.7 미들 웨어 권한 확인

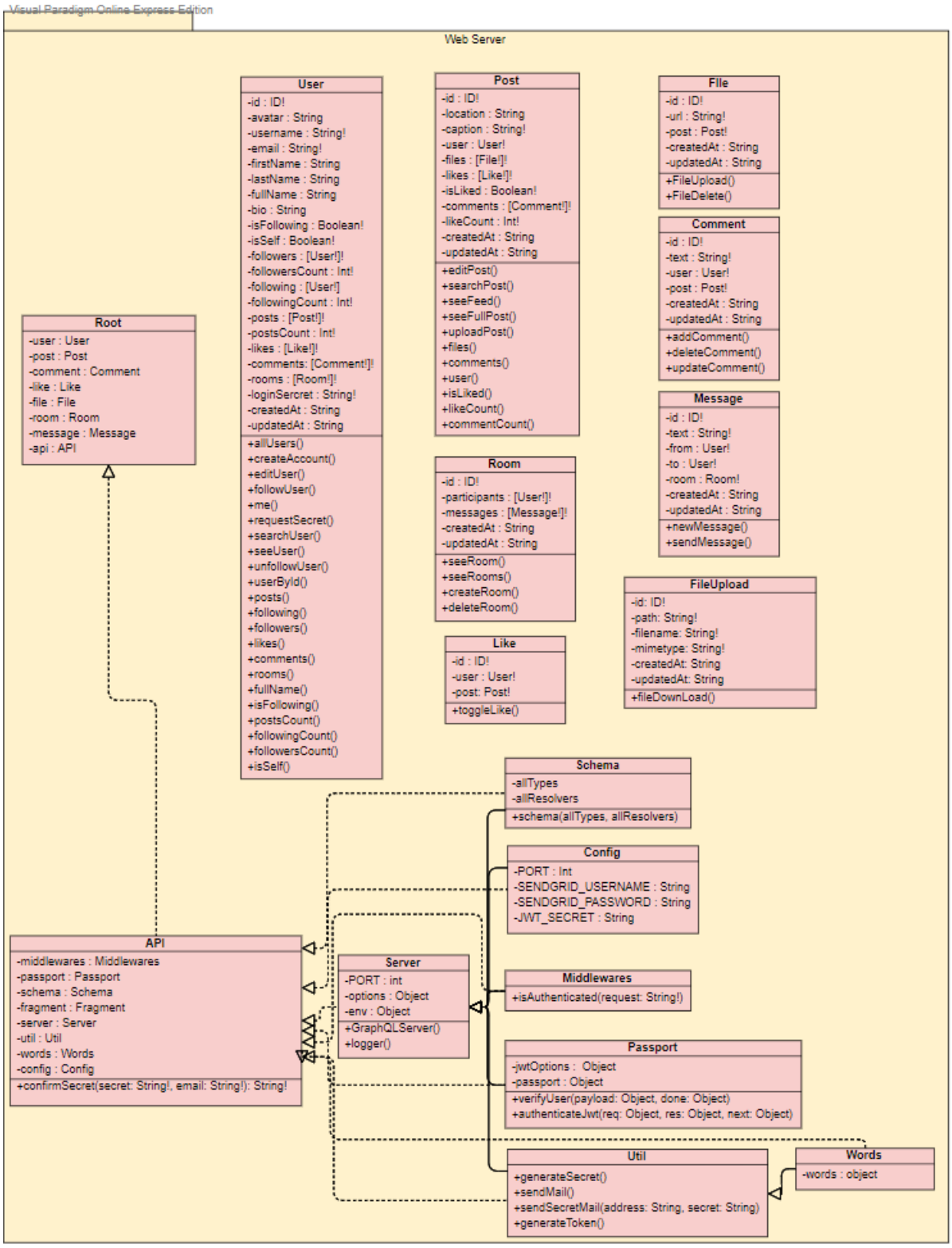


4.4.2 Class Diagram

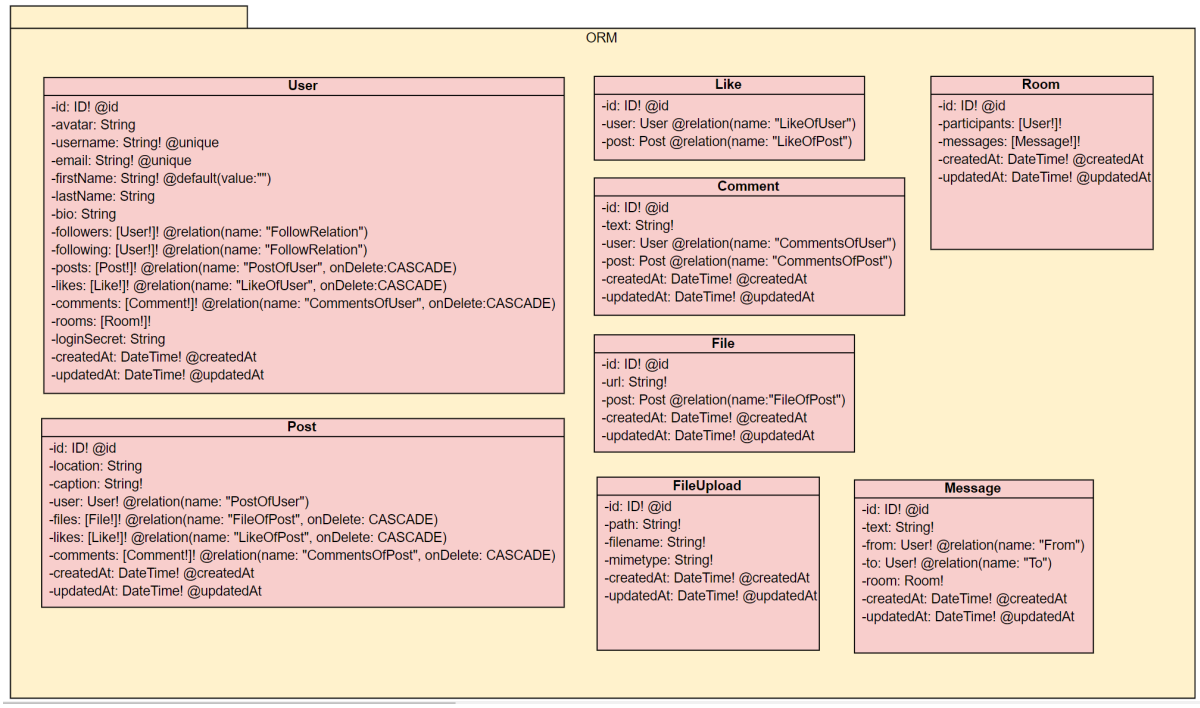
4.4.2.1 Web Server Client

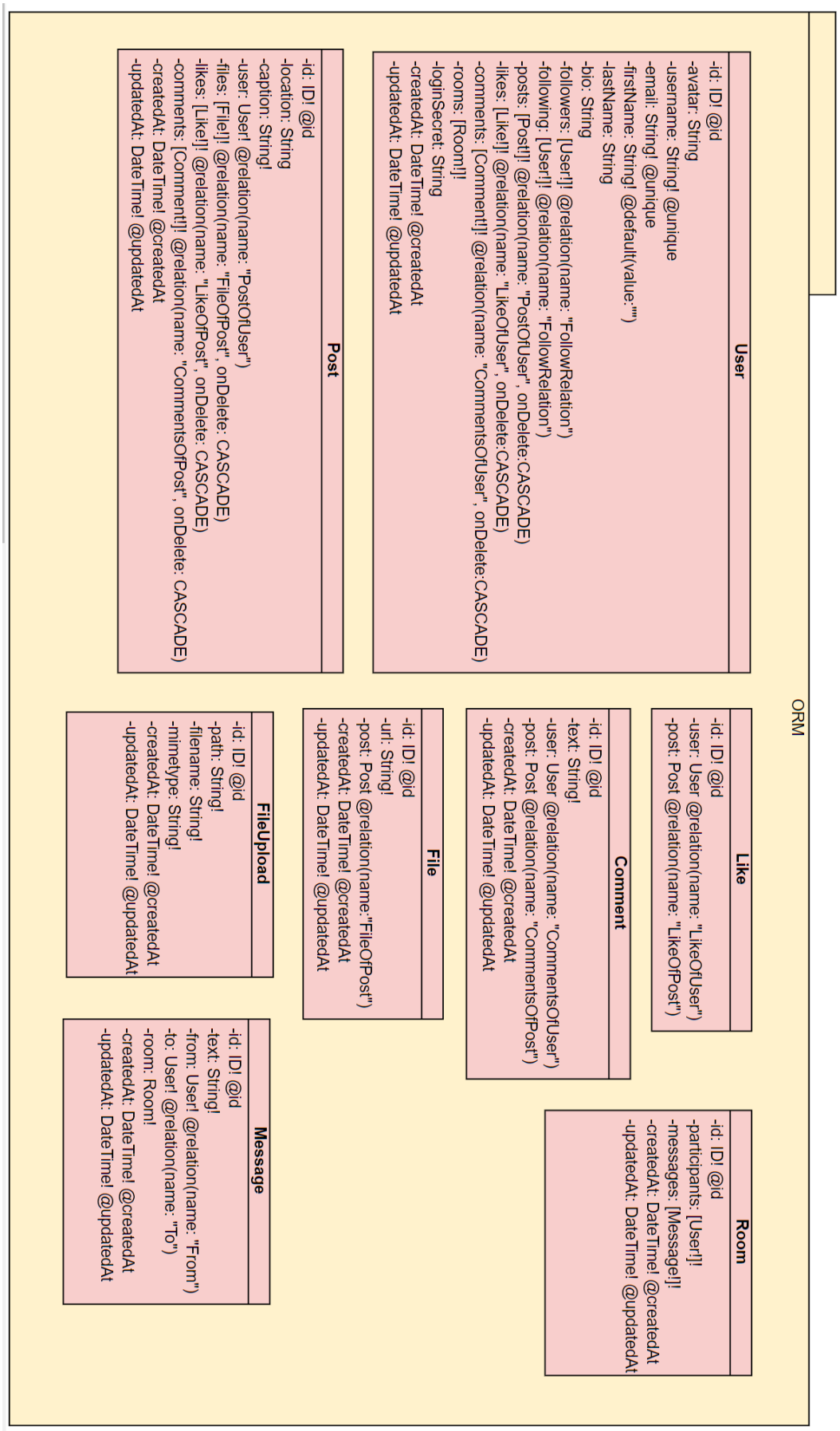


4.4.2.2 Web Server



4.4.2.3 ORM Server





5 팀원 담당업무 및 요구사항 완료

이름	업무	세부사항
박영준(팀장)	게임 개발 및 모델 설계 총괄	Unity, Deep learning Model 설계 총괄
문명기	Deep Learning Model 설계, 구현	attention mechanism 과 seq2seq 를 통한 모델 설계 및 서버 구축
김세진	게임 개발	Unity
이호찬	Web Server	웹 플랫폼 구축
조동철	Deep Learning Model 전처리 설계	전처리 및 설계 구현

5.1 요구사항 점검

아래 테이블은 System 의 상세설계 전 명세 된 요구사항을 검토하고자 유형을 기호화 Mapping 한 자료이다.

유형		요구사항 유형(구성 요소)	요구사항 기호
시스템	기능 요구사항	인터페이스 어플리케이션 부 요구사항	SFR-I
		인터페이스 게임 어플리케이션 부 요구사항	SFR-GI
		서비스 어플리케이션 부 요구사항	SFR-V
		Server 요구사항	SFR-S
		ORM Server 요구사항	SFR-O

요구사항 기호가 의미하는 바에 대해 알기 위해 다음 Table 을 참조할 수 있다.

요구사항 기호	의 미
SFR-I	'Interface application'에게 요구되는 기능을 의미하고, 크게 14 개의 요구 사항으로 구분된다.
SFR-GI	'Interface Game application'에게 요구되는 기능을 의미하고, 9 개의 요구 사항으로 구분된다.
SFR-V	'Service application'에게 요구되는 기능을 의미하고, 크게 10 개의 요구 사항으로 구분된다.
SFR-S	'Server'에게 요구되는 기능을 의미하고, 크게 29 개의 요구 사항으로 구분된다.
SFR-O	'ORM Server'에게 요구되는 기능을 의미하고, 크게 7 개의 요구 사항으로 구분된다.

5.2 요구사항 완료

유형	상세유형	요구사항 명	요구사항 ID	항목수
시스템 기능 요구사항	인터페이스 어플리케이션 부 요구사항 (SFR-I)	로그인 기능 제공	SFR-I-01	14 완료
		로그아웃 기능 제공	SFR-I-02	
		회원가입 기능 제공	SFR-I-03	
		전체 음악 노트 게시물 확인 기능	SFR-I-04	
		음악 노트 게시물 댓글, 좋아요, 파일, 이미지 보기 기능 제공	SFR-I-05	
		음악 노트 게시물 및 사용자 검색 기능 제공	SFR-I-06	
		개인정보 확인 기능 제공	SFR-I-07	
		탈퇴 기능 제공	SFR-I-08	
		개인정보 변경 기능 제공	SFR-I-09	
		좋아요 개수 높은 음악 노트 게시물 정보 제공	SFR-I-10	
		자신이 좋아하는 사용자의 음악 노트 게시물 확인 기능 제공	SFR-I-11	
		음악 노트 공유 기능 제공	SFR-I-12	
		음악 노트로 변환 기능 제공	SFR-I-13	
		페이지 렌더링 대기상태 시 보여지는 애니매이션 제공	SFR-I-14	
	인터페이스 게임 어플리케이션 부 요구사항 (SFR-GI)	키넥트 모션 인식을 이용한 인터페이스조작	SFR-GI-01	4 완료
		플레이 모드선택(커스텀, 2 인)	SFR-GI-02	
		BeatNote 들 생성과 이동	SFR-GI-03	
		플레이한 결과 종합 정보 표시	SFR-GI-04	
	서비스 어플리케이션 부 요구사항 (SFR-V)	전처리 1 – 음악파일 변환	SFR-V-01	11 완료
		전처리 2 – STFT 적용 및 스케일링	SFR-V-02	
		파형 feature 추출	SFR-V-03	
		손실 함수 정의_ CNN, RNN	SFR-V-04	
		활성화 함수 정의	SFR-V-05	
		최적화 알고리즘	SFR-V-06	
		Dataset mirroring	SFR-V-07	
		CNN 모델	SFR-V-08	
		LSTM(RNN) 모델	SFR-V-09	
		LSTM input 및 output	SFR-V-10	
		Beat Note 점수 판정 범위	SFR-V-11	
	서버 부 요구사항 (SFR-S)	댓글 생성 기능 제공	SFR-S-01	29 **완료**
		댓글 삭제 기능 제공	SFR-S-02	
		로그인 시도 시 정보 일치 여부 반환 기능 제공	SFR-S-03	

		좋아요 상태 변경 기능 제공	SFR-S-04	
		게시글 생성 기능 제공	SFR-S-05	
		게시글 수정 기능 제공	SFR-S-06	
		게시글 삭제 기능 제공	SFR-S-07	
		게시글 검색 기능 제공	SFR-S-08	
		전체 게시글 보기 기능 제공	SFR-S-09	
		내 게시글 보기 기능 제공	SFR-S-10	
		사용자 생성 기능 제공	SFR-S-11	
		모든 사용자 보기 기능 제공	SFR-S-12	
		사용자 정보 수정 기능 제공	SFR-S-13	
		사용자 팔로우 기능 제공	SFR-S-14	
		사용자 팔로우 취소 기능 제공	SFR-S-15	
		자기 자신 정보 보기 기능 제공	SFR-S-16	
		비밀 키 송신 기능 제공	SFR-S-17	
		사용자 검색 기능 제공	SFR-S-18	
		댓글 수 확인 기능 제공	SFR-S-19	
		게시글 수 확인 기능 제공	SFR-S-20	
		자신이 한 팔로우 수 확인 기능 제공	SFR-S-21	
		자신에게 팔로우 한 수 확인 기능 제공	SFR-S-22	
		대화 방 생성 기능 제공	SFR-S-23	
		대화 방 보기 기능 제공	SFR-S-24	
		대화 메시지 실시간 수신/송신 기능 제공	SFR-S-25	
		유효 JWT (JSON Web Token) 확인 기능 제공	SFR-S-26	
		Secret Key 메일 보내기 기능 제공	SFR-S-27	
		JWT(JSON Web Token) 생성 기능 제공	SFR-S-28	
		광고 기능 제공	SFR-S-29	
	ORM 서버 부 요구사항 (SFR-O)	User Type 정의	SFR-O-01	7 완료
		Post Type 정의	SFR-O-02	
		Like Type 정의	SFR-O-03	
		Comment Type 정의	SFR-O-04	
		File Type 정의	SFR-O-05	
		Room Type 정의	SFR-O-06	
		Message Type 정의	SFR-O-07	
총 항목 수: 65 개 완료				

6 프로젝트 세부일정

구분	추진 내용	3 주차	4 주차	5 주차	6 주차	7 주차	8 주차	9 주차	10 주차	11 주차	12 주차	13 주차	14 주차
계획	사업 아이디어 회의												
	제안서 작성												
분석	기술조사 및 시장 조사												
	요구사항 수집 및 분석												
설계	게임 환경 설계												
	머신러닝 모델 설계												
	Web Platform 설계												
개발	Unity 게임환경 구현												
	CRNN 설계 및 학습												
	Web Server 및 Platform												
테스트	CRNN 적용 output 확인												
	게임 오류 수정 및 디버깅												
종료	최종 발표 및 시연 준비												

7 참고문헌

[1] 박한솔.(2019) 리듬 게임 노트 자동 생성에 적합한 합성곱 신경망 설계 및 구현에 관한 연구. 가천대학교 게임대학원 석사논문

[2] Gamemeca. (2007). 리듬액션 게임의 역사 – 탄생에서 현재까지
https://www.gamemeca.com/view.php?gid=124557

[3] DEV KOREA. (2019.01). 리듬게임 노트 생성 알고리즘
http://www.devkorea.co.kr/bbs/board.php?bo_table=m03_qna&wr_id=95493

[4] Keunwoo Choi. (2016). Convolutional Recurrent Neural Networks for Music Classification
http://keunwoochoi.wordpress.com/tag/crnn/

[5] 브라우저는 어떻게 동작하는가
https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

[6] About React TimeSlicing and Suspense
https://www.youtube.com/watch?v=v6iR3Zk4oDY&feature=youtu.be&t=135

[7] graphql
https://tech.kakao.com/2019/08/01/graphql-basic/

[8] restful api 단점
https://www.slideshare.net/deview/112rest-graph-ql-relay -