

MUSIC PLAYER

이정임



Index

1 뮤직플레이어 소개 p. 3-8

2 CODE p. 9-19

3 후기 p. 20-21



1. 뮤직플레이어 소개

p. 3-8

1. 뮤직플레이어 소개

사용 기술



front-end

HTML



CSS



JavaScript



back-end

node.js



Express



MySQL



1. 뮤직플레이어 소개

디자인 기획



사용자의 편의성

깔끔하고 직관적인 UI를 사용하여 사용자가 기존 뮤직플레이어와 비슷한 느낌을 가질 수 있도록 노력하였습니다.

노을 디자인

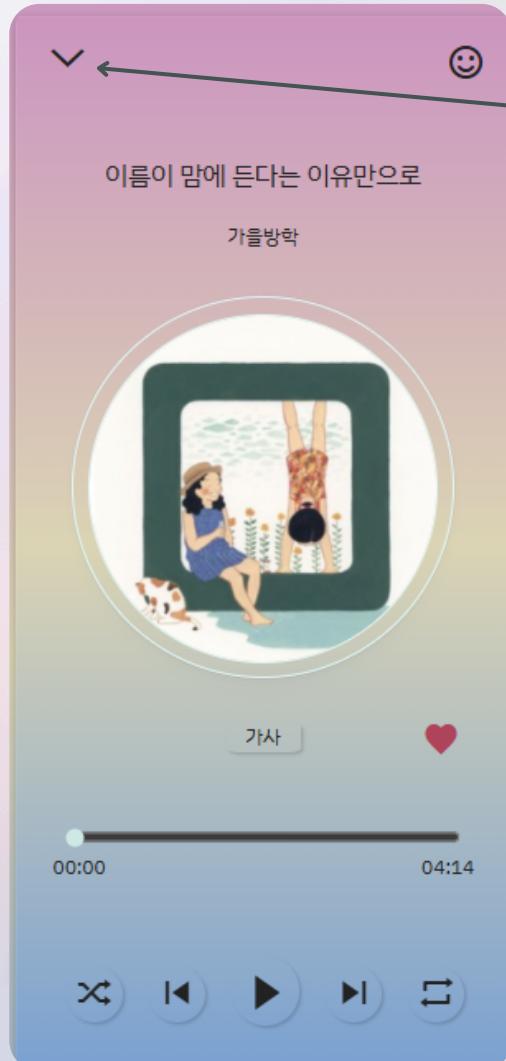
해가 지는 노을을 표현한 밝은 느낌의 파스텔 톤 디자인입니다.
사용자의 눈이 편안할 수 있게 연한 색을 위주로 사용하였습니다.

디자인 포인트

커버와 버튼을 전체적으로 둥글게 만들어 귀여운 느낌이 들도록 디자인 하였습니다.

1. 뮤직플레이어 소개

화면 소개



메뉴 버튼 클릭 시

재생목록에 있는 노래 리스트가 나옵니다.

원하는 노래를 골라 클릭하면,

해당 노래가 재생됩니다.

이름이 맘에 든다는 이유만으로
가을방학

취미는 사랑
가을방학

속마음
찰총

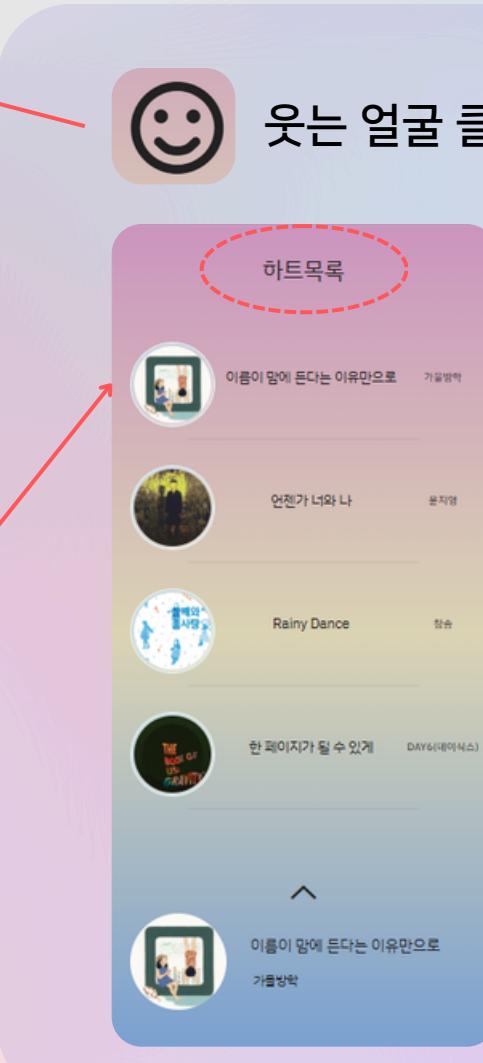
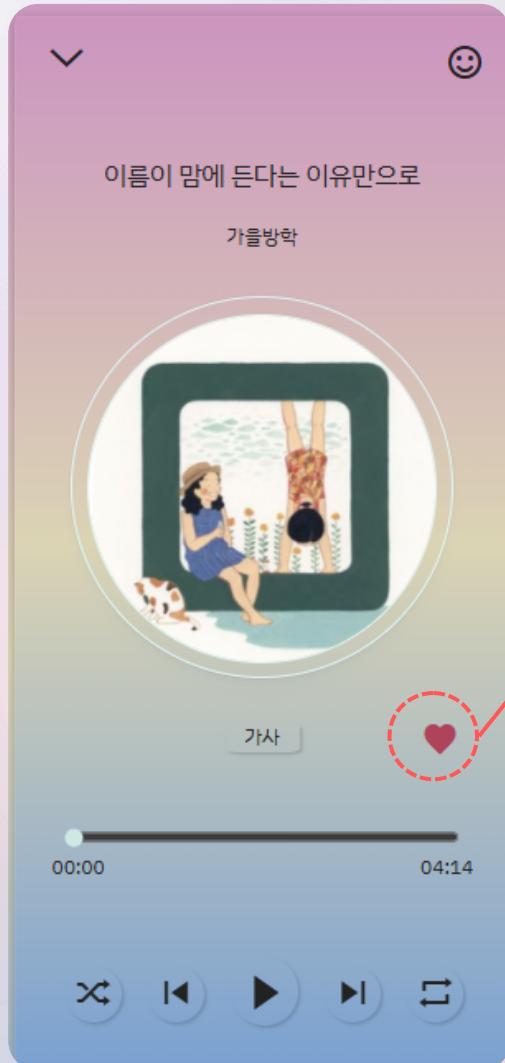
언젠가 너와 나
문지영

이름이 맘에 든다는 이유만으로
가을방학

취미는 사랑
가을방학

1. 뮤직플레이어 소개

화면 소개



웃는 얼굴 클릭 시

하트목록에 있는
노래 리스트가 나옵니다.

하트목록에는 하트를 눌렀던
노래들이 쌓여있습니다.

하트목록에서 노래를 클릭하면
하트목록 내의 음악이 순서대로
재생됩니다.

하트를 다시 누르면
하트가 해제 되며,
해당 리스트에서는 사라집니다.

1. 뮤직플레이어 소개

화면 소개



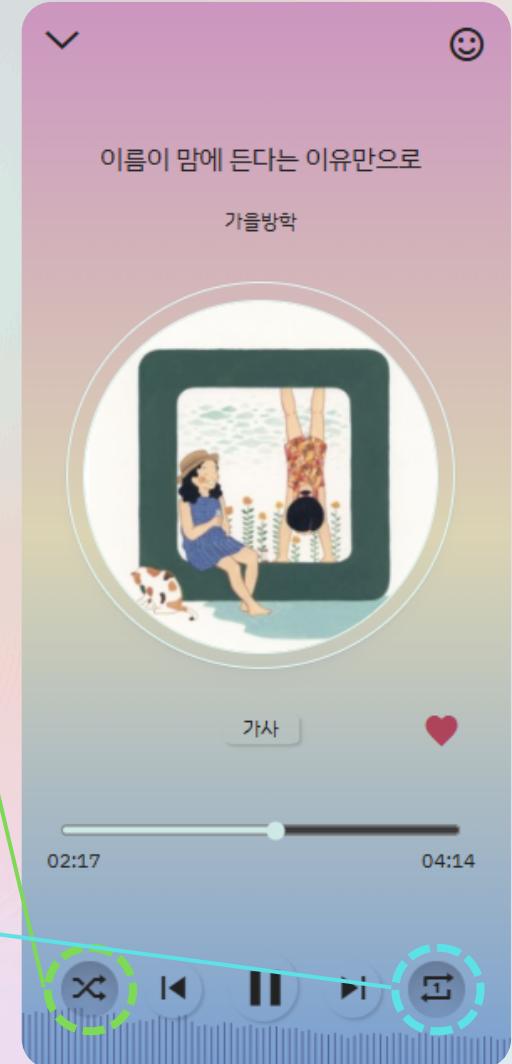
음악 재생 기능

랜덤버튼을 클릭하면 다음곡부터
랜덤으로 플레이 됩니다.

반복 재생 버튼을 클릭하면
현재 재생 중인 음악을
반복 재생합니다.

하단 영역에 재생 중인 음악의
파형이 생성됩니다.

버튼들이 현재 눌려 있다는
느낌을 주기 위해 버튼을
안으로 넣은 듯한 효과를
주었습니다.





2. CODE

p. 9–19

2. CODE

FRONT-END



```
1 class MusicPlayer {  
2     constructor(id) {  
3         this.id = id;  
4         this.music = [];  
5         this.likeData = [];  
6         this.Index = 0;  
7         this.likeIndex = 0;  
8         this.isLike = false;  
9     }  
}
```

class의 constructor

music에는 빈 배열을 만들어 놨습니다.
DB와 연동하여 배열에
music list가 추가됩니다.

likeData에는 좋아요를 누른 곡들이
추가됩니다.

재생목록의 Index와 하트목록의 likeIndex를
달리하여 음악을 재생할 때,
해당 Index를 가지고 옵니다.

isLike를 false로 기본 설정하여
isLike가 true일 때
likeIndex를 이용합니다.

2. CODE

FRONT-END



```
1 const musicPlayer = new MusicPlayer("musicPlayer");
2 musicPlayer.control();
```



```
1 async control() {
2     try {
3         const response = await fetch("http://kkms4001.iptime.org:45170/data");
4         const data = await response.json();
5         this.music = data.map((song) => ({
6             id: song.id,
7             singer: song.singer,
8             title: song.title,
9             img: song.img,
10            song: song.song,
11            text: song.text,
12            like: song.like
13        }));
14         this.likeData = this.music.filter(song => song.like === "1");
```

control

fetch를 사용하여 DB에 저장해 놓았던 노래 목록을 받아옵니다.

DB에 들어있는 music data를 map함수를 이용하여 this.music에 노래 목록을 넣었습니다.

filter함수를 사용하여 this.music에 안의 like가 1(true)인 노래를 this.likeData에 넣었습니다.

2. CODE

FRONT-END



```
1 if (this.isLike == false) {  
2     const nowIndex = this.music[this.Index];  
3     document.getElementById("title").innerHTML = `${nowIndex.title}`;  
4     document.getElementById("singer").innerHTML = `${nowIndex.singer}`;  
5     document.getElementById("musicCover").innerHTML = ``;  
6     document.getElementById('musicBox').innerHTML = `<audio src="../music/${nowIndex.song}" id="${nowIndex.song}"></audio>`;  
7     this.updateUI(nowIndex);  
8     this.visual(nowIndex);  
9     mediaPlayer.timeupDate(nowIndex);  
10 }
```

이름이 맘에 든다는 이유만으로

가을방학



main재생 화면 출력

현재 재생 중인 노래를 main화면에 표시하는 코드입니다.
배열 안에 있는 music에서 현재 인덱스 번호를 가지고 온 후,
Index의 title, singer, img, song을 화면에 출력합니다.

2. CODE FRONT-END



```
1 displayList() {
2     // 일반 재생목록
3     const listContainer = document.getElementById(`Box2_songs`);
4     listContainer.innerHTML = '';
5     this.music.forEach((v, i) => {
6         listContainer.innerHTML += `
7             <aside id="Box2_song${i + 1}" class="Box2_song">
8                 <div class="Shadow_2">
9                     <div id="Box2_cover${i + 1}" class="Box2_cover">
10                         
11                     </div>
12                 </div>
13                 <div class="nameAndSinger">
14                     <span id="songname${i + 1}" class="songname">${v.title}</span>
15                     <span id="singername${i + 1}" class="singername">${v.singer}</span>
16                 </div>
17             </aside>`;
18     });
}
```

PlayList화면 출력

music안에 있는 노래 수에 맞게 재생 목록 화면을 동적으로 생성합니다.



```
1 setupPlaylistClick() {
2     this.music.forEach((v, i) => {
3         document.getElementById(`Box2_song${i + 1}`).addEventListener('click', () => {
4             this.Index = i;
5             this.isLike = false;
6             this.display();
7             this.playMusic();
8         });
9     });
}
```

PlayList노래 클릭 시

클릭한 노래를 Index로 지정하여 display, playMusic을 호출 합니다.

재생목록



이름이 말에 듣다는 이유만으로

기울방학



취미는 사랑

기울방학



속마을

침승



언젠가 너와 나

문지영

2. CODE

FRONT-END



```
1 playMusic() {  
2     // console.log(this.isLike);  
3     document.getElementById('playBtn').style.display = "none";  
4     document.getElementById('stopBtn').style.display = "block";  
5     // 일반 재생  
6     if (this.isLike == false) {  
7         const nowIndex = this.music[this.Index];  
8         const audio = document.getElementById(nowIndex.song);  
9         console.log(audio);  
10        if (document.getElementById('stopBtn').style.display == "block") {  
11            audio.play();  
12            audio.addEventListener('ended', () => {  
13                if (document.getElementById('oneSong').style.display == "none") {  
14                    audio.pause();  
15                    this.playMusic();  
16                }  
17                else {  
18                    this.nextSong();  
19                }  
20            });  
21        }  
22    }  
}
```

노래 play하기

현재 Index에 해당하는 노래를 재생합니다.

노래가 끝날 때,
반복 재생 중인지 판단 후,

반복 재생이라면
playMusic을 다시 호출하고,
반복 재생이 아닐 시,
nextSong을 호출합니다.

2. CODE

FRONT-END



```
1 nextSong() {
2     // like 아닌 그냥 일반 재생목록의 다음곡 재생
3     if (this.isLike == false) {
4         const nowIndex = this.music[this.nowIndex];
5         if (document.getElementById("randomBtn").classList.contains("inside")) {
6             let randomNum = (Math.floor(Math.random() * (this.music.length + 1)));
7             if (this.Index == randomNum) {
8                 this.Index = randomNum + 1;
9                 this.display();
10                this.playMusic();
11            }
12            else {
13                this.Index = randomNum;
14                this.display();
15                this.playMusic();
16            }
17        }
18        else {
19            if (this.Index < this.music.length - 1) {
20                this.Index++;
21            } else {
22                this.Index = 0;
23            }
24            this.display();
25            this.playMusic();
26        }
27    }
}
```

랜덤 재생 처리 ✖

랜덤 재생 중 이라면 Index를 randomNum으로
지정합니다.

만약 randomNum과 현재 인덱스가
같다면, randomNum+1 을 하여
현재곡과 다음곡이 겹치지 않게
설정하였습니다.

랜덤이 아닐 시 ✖

Index를 하나 증가 시켜서
music안에 있는 다음 노래를
display 및 playMusic합니다.

2. CODE

FRONT-END



```
● ● ●  
1 // 좋아요 누른 DB 업뎃  
2     async likeDbData() {  
3         const nowIndex = this.music[this.Index];  
4         // console.log(nowIndex.id);  
5         try {  
6             const response = await fetch('http://kkms4001.iptime.org:45170/like', {  
7                 method: "POST",  
8                 headers: {  
9                     "Content-Type": "application/json",  
10                },  
11                body: JSON.stringify({ id: nowIndex.id }),  
12            );  
13            const data = await response.json();  
14            this.music = data.map((song) => ({  
15                id: song.id,  
16                singer: song.singer,  
17                title: song.title,  
18                img: song.img,  
19                song: song.song,  
20                text: song.text,  
21                like: song.like  
22            }));  
23            // console.log(this.music);  
24            this.likeData = this.music.filter(song => song.like === "1");  
25            // console.log(this.likeData);  
26        } catch (err) {  
27            console.error("에러 발생:", err);  
28        }  
29    }
```

좋아요 버튼 클릭 시



likeDbData를 호출하여
현재 재생 중인 노래의 id를 추출하
여 json형식으로 서버에 보냅니다.

서버에서 DBData를 다시 보내주면
music과 likeData를
업데이트합니다.

좋아요 버튼 해제 시



위와 같은 형식으로 DBdata를
다시 업데이트 합니다.

2. CODE

BACK-END



music		id	singer	title	img	song	text	like	
	id ↗	int	NN	1 가을방학	이름이 맘에 든다는 이유만으로 가을방학- 이름이 맘에 든다는 이유만으로.PNG 가을방학- 이름이 맘에 든다는 이유만으로.m4a	<p> 이름이 맘에 든다는 이유만으로 같은 계절을 좋아한단 것만으로<...>	1		
singer		char(100)	NN	2 가을방학	취미는 사랑	가을방학- 취미는 사랑.PNG	가을방학 - 취미는 사랑.m4a	<p> 미소가 어울리는 그녀 취미는 사랑이라 하네 만화책도 영화도 아...	
title		char(100)	NN	3 참송	속마음	참송- 속마음.PNG	참송- 속마음.m4a	<p> 여전히 말하지 못한 내 속마음과 그대와 돌아가는 이 시곗바늘 ...	
img		char(100)		4 윤지영	언젠가 너와 나	윤지영(feat.카더가든)-언젠가 너와 나.PNG	윤지영 (Yoon Jiyoung) - 언젠가 너와 나 (Feat. 카더가든) .m4a	<p> 언젠가 너와 나 중에 누굴 선택해야한다면 나는 ...	
song		char(100)	NN	5 참송	Rainy Dance	Rainydance.png	Rainy Dance.m4a	<p> 오늘은 분명히 날이 좋다 했는데 하나님을 둘러오는 구름 수상...	
text		text		6 치즈	Madeleine Love	Madeleine Love.png	Madeleine Love.m4a	<p> 오늘같이 싱그러운 날엔 길거리 차도 별로 다니지 않아 ...	
like		varchar(10)		7 치즈	Mood Indigo	치즈 - Mood Indigo.PNG	치즈 - Mood Indigo.m4a	<p> When I Fall In Love 그대와 함께 한다면 ...	
				8 DAY6(데이식스) 한 페이지가 될 수 있게	DAY6(데이식스) - 한 페이지가 될 수 있게.PNG	DAY6(데이식스) - 한 페이지가 될 수 있게.m4a	<p> 솔직히 말할게 많이 기다려 왔어 너도 그랬을 거...	1	
				9 DAY6(데이식스) 예뻤어	DAY6(데이식스)- 예뻤어.PNG	DAY6(데이식스)- 예뻤어.m4a	<p> 지금 이 말이 우리가 다시 시작하자는 건 아냐<...>	0	
				10 10cm	가끔 연락하던 애	10CM - 가끔 연락하던 애.PNG	10cm - 가끔 연락하던 애.m4a	<p> 갠 가끔 언제 만나자면 나오곤 하던 애였는데 잠...	0

DATABASE 설계

singer에는 가수, title에는 노래 제목, img에는 노래 커버, song에는 audio, text에는 가사, like에는 true, false를 넣었습니다.

2. CODE

BACK-END



```
1
2 app.get('/',(req,res)=>{
3   fs.readFile("./public/page/musicPlayer.html",'utf-8',(err,inData)=>{
4     if(err) throw err;
5     res.send(inData);
6   })
7 });
8
```

html 보내기

처음 뮤직플레이어에 접속을 하면
readFile을 사용하여 html을
보내줍니다.

```
1 app.get("/data",(req,res)=>{
2   dbInstance.query("select * from music" ,(err , dataObj)=>{
3     if(err) throw err;
4     console.log(dataObj);
5     res.json(dataObj);
6   })
7 });
8
```

DB 데이터 받아오기

select문을 사용하여
musicTable을 선택 후
dataObj를 보냅니다.

2. CODE

BACK-END



```
● ○ ●
1 app.post("/like", (req, res) =>{
2   // console.log(req.body.id);
3   dbInstance.query("UPDATE music SET `like` = true WHERE id= ? ", [req.body.id], (err, changeObj) =>{
4     if(err) {
5       console.log(err);
6       throw err;
7     }
8     // console.log("update successful : " , changeObj);
9   });
10
11 dbInstance.query("select * from music", (err, dataDB) =>{
12   if(err) {
13     console.log(err);
14     throw err;
15   }
16   res.json(dataDB);
17 });
18});
```

```
● ○ ●
1 app.post("/dontlike", (req, res) =>{
2   // console.log(req.body.id);
3   dbInstance.query("UPDATE music SET `like` = false WHERE id= ? ", [req.body.id], (err, changeObj) =>{
4     if(err) {
5       // console.log(err);
6       throw err;
7     }
8     console.log("update successful : " , changeObj);
9   });
10
11 dbInstance.query("select * from music", (err, dataDB) =>{
12   if(err) {
13     console.log(err);
14     throw err;
15   }
16   res.json(dataDB);
17 });
18});
```

좋아요 버튼 클릭 시

music의 해당 id를 선택하여 like를 true로 변경하여 update합니다.

업데이트 후 DB의 값을 다시 전송 합니다.

좋아요 버튼 해제 시

위와 같은 형식으로 music의 해당 id를 선택하여 like를 false로 변경하여 update합니다.

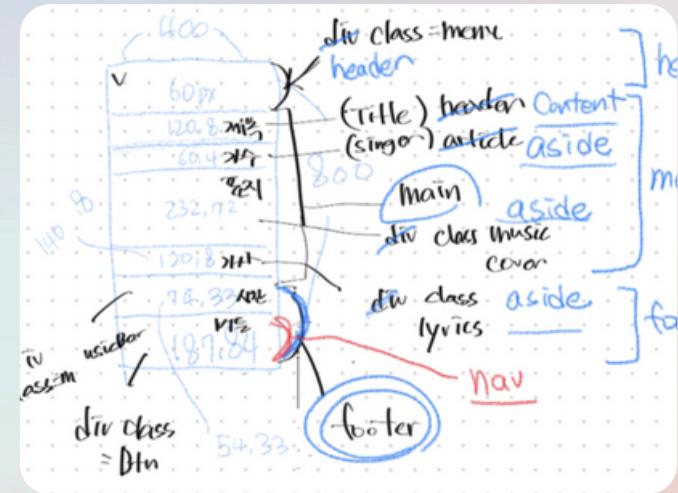
업데이트 후 DB의 값을 다시 전송 합니다.



3. 후기

p. 20-21

3. 후기



처음 Music Player를 제작할 때 노래 선곡부터 디자인, DB설계, back-end작업까지 해야 한다는 것에 두려움보다는 설렘이 다가왔습니다.

DB설계 과정에서는 likeTable을 따로 만들어서 연결을 해야 하는지, 아니면 musicTable에 같이 넣는 게 맞는지 고민을 많이 하였습니다. 고민 끝에 하나로 Table을 설계하였지만, 다음에는 Table을 연결하여 Join하는 방식을 사용해 보고 싶다고 생각하였습니다.

또한 back-end작업 과정에서는 어떻게 해야 효율적이면서 값이 꼬이지 않고 전달이 될지 고민하여 like버튼을 누를 때와 아닐 때 라우팅을 따로 작성하였는데, 이 부분도 조금 더 효율적인 방향을 찾아서 보완하고 싶다고 생각하였습니다.

이번 프로젝트로 저의 front-end와 back-end 실력이 많이 향상된 것 같아서 뿌듯하고, 모든 부분에 저의 주관이 많이 들어갔던 MusicPlayer를 무사히 완성했다는 사실에 감사합니다. 😊

감사합니다.

이정임