INTRODUCE

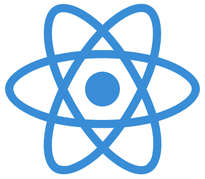
리액트 뮤직플레이어/채팅 웹앱 포트폴리오

다양한 인터랙티브 모션과 간단한 로그인/채팅창을 만들기 위한 백엔드 공부용 토이프로젝트입니다.
주요 기능은 **뮤직플레이어/구글아이디 로그인/채팅창**입니다.

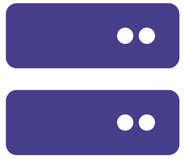
INDEX

리액트 뮤직플레이어 목차

리액트 뮤직플레이어 주요기능



리액트를 이용하여 동적인 이벤트를
처리하고 사용자와 상호작용



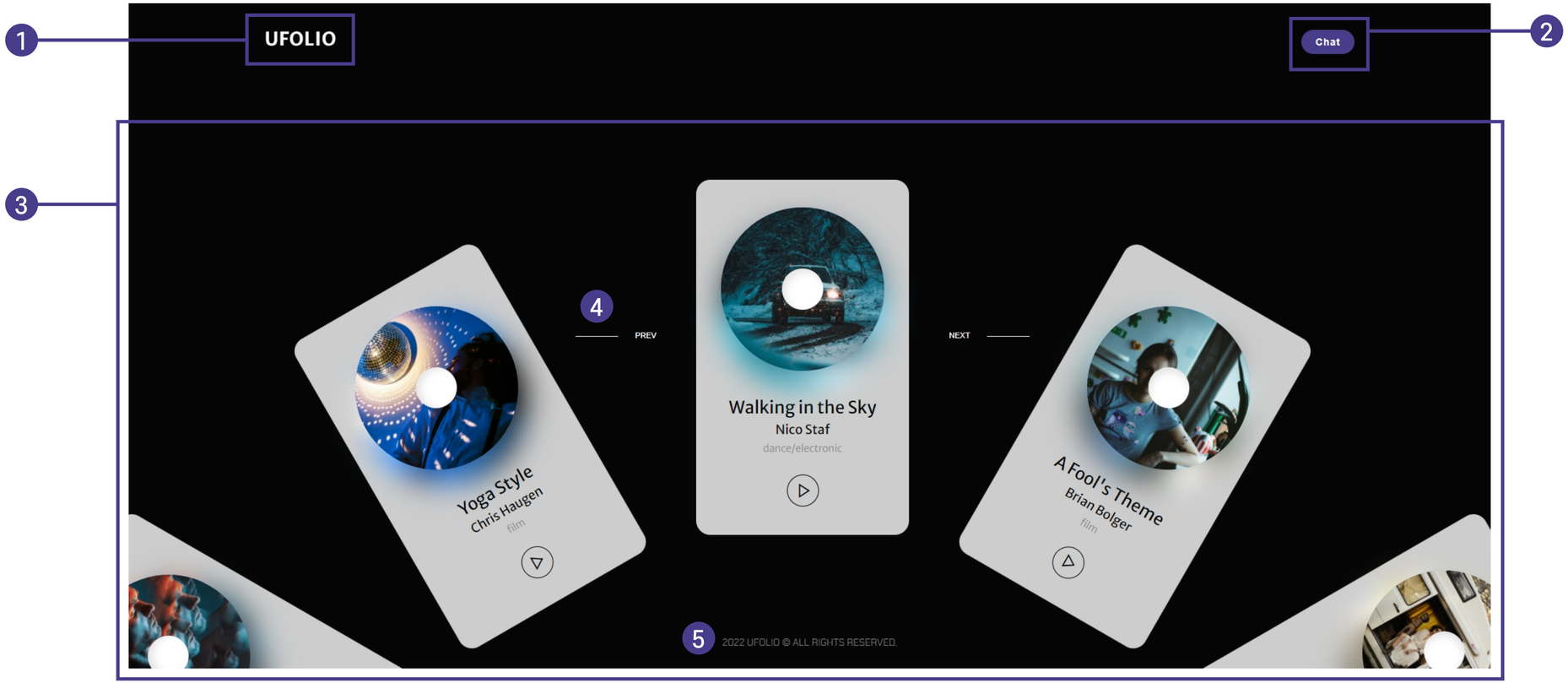
Firebase를 이용하여 데이터를 저장,
실시간 업데이트를 통해 **채팅창 구현**

SCSS를 사용하여 코드의 가독성을
높여 기존 CSS의 단점을 보완

Firebase를 이용하여 사용자의
인증 및 관리하고 구글 로그인 기능 구현

MAIN PAGE STRUCTURE

메인 페이지 구조 설명



1. Header.js
2. Chat.js
3. Panels.js
4. Btns.js
5. Footer.js

메인 페이지의 구조는 Header, Footer, Panels(MusicPlayer), Btns, Chat 으로 구성되어 있으며
프로젝트 특성상 모션도 사이트의 중요한 요소이기 때문에 반응형대신 적응형으로 작성되었습니다.

FUNCTIONS DETAIL

기능 상세설명 2-1

간단하게 만든 JSON 파일을 저장후, axios를 이용하여 JSON 데이터 추출.
재생 버튼 클릭시 음악과 이미지 애니메이션이 재생/정지 구현.
또, 양 옆의 버튼을 클릭하면 뮤직플레이어의 프레임이 30deg 씩 rotate가 됨.

1 axios를 이용해 데이터 호출

albums.json의 데이터 값은
{title, artist, janre, img, mp3} 로 구성.

```
useEffect(()=>{
      axios.get(`${path}/dbs/albums.json`).then(data=>{
         setAlbums(data.data.data);
      })
   },[])
```

2 Panels.js 함수 정의

initMusic() :
오디오, 재생/정지 버튼, 재생목록 모션 초기화 함수 정의

playMusic() :
재생 버튼에 class="on"이 포함되어있을때,
버튼을 누르면 오디오, 버튼, 재생목록 모션 제거.
포함되지 않을때 버튼을 누르면 오디오, 모션 재생 함수 정의

3 Btns(props)

버튼을 클릭하면 뮤직 플레이어의
프레임이 좌우로 30도씩 이동.

```
<div className="btnPrev" onClick={plus}><span>PREV</span></div>
<div className="btnNext" onClick={minus}><span>NEXT</span></div>
```

4 Btns.js 함수 정의

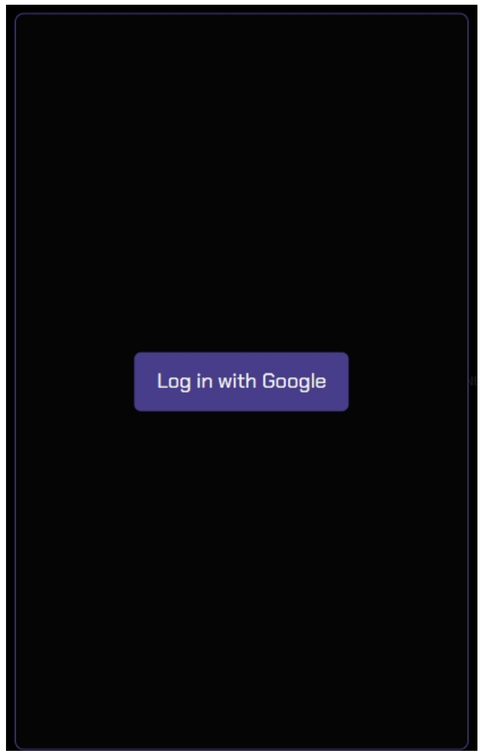
plus() :
Prev 버튼 클릭시 ++index가 되면서 왼쪽으로 30도씩 rotate되는 함수 정의.
동시에 모든 음악과 모션 재생 정지.

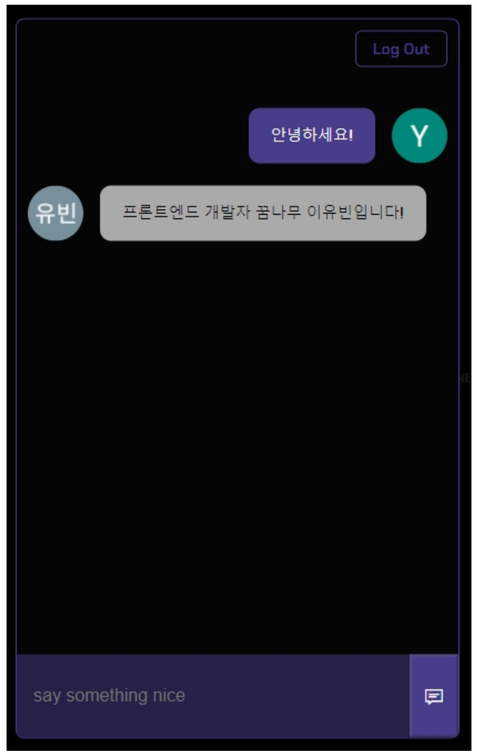
minus() :
Next 버튼 클릭시 --index가 되면서 오른쪽으로 30도씩 rotate되는 함수 정의.
동시에 모든 음악과 모션 재생 정지.

FUNCTIONS DETAIL

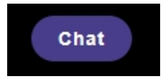
기능 상세설명 2-2

Firebase를 이용해 데이터를 저장하고 사용자를 인증/관리하며
채팅창과 로그인/로그아웃을 구현.





1 Chat 버튼 클릭시 팝업창 토글



버튼을 클릭하면 팝업창이 토글로 생성/제거가 됨.

```
<button onClick={toggleChat} className="chatBtn">Chat</button>
<aside className={isOn ? 'on' : null}>
```

Default 값으로 isOn = false 를 두고
isOn = true 일때 팝업창 생성, false일때 제거.

2 로그인/로그아웃

Firebase Authentication 빌드를 통해
구글 아이디로 사용자 인증 및 관리

```
const SignIn = ()=>{
  const signInWithGoogle = () => {
    const provider = new
firebase.auth.GoogleAuthProvider();
    auth.signInWithPopup(provider);
  }
  return (
    <>
      <button className="signIn"
onClick={signInWithGoogle}>Log in with
Google</button>
    </>
  )
}
const SignOut = ()=>{
  return auth.currentUser && (
    <button className="signOut"
onClick={() => auth.signOut()}>Log Out</
button>
  )
}
```

3 채팅창 구현

Firestore에서 컬렉션, 문서, 필드 추가후
데이터를 관리하며 실시간으로 업데이트.

4 Chat.js 함수 정의

SigIn() :
로그인 버튼을 클릭하면 구글로그인 창이 뜨면서
구글 아이디로 채팅창 접속 함수 정의.

SignOut() :
로그아웃 버튼을 클릭하면 다시 처음 로그인 창으로
바뀌는 함수 정의. 로그아웃 상태로 변화.

ChatRoom() :
사용자가 로그인이 되면 ChatRoom 생성. input창에 value 값을
입력 후 데이터를 보내면 store의 collection 필드 값을 호출하여
채팅창 구현하는 함수 정의 .

ChatMessage() :
uid에 따라 messageClass에 'sent'와 'received'를 변수에 저장.
하나의 메세지에 클래스 네임을 붙여 스타일 적용을 돕는 함수 정의.

```
const ChatMessage = (props)=>{
  const { text, uid, photoURL } = props.message;
  const messageClass = uid === auth.currentUser.uid ? 'sent' : 'received';
  return (
    <>
      <div className={`message ${messageClass}`}>
        <img src={photoURL || 'https://api.adorable.io/avatars/23/abott@adorable.png'} />
        <p>{text}</p>
      </div>
    </>
  )
}
```