

금융 데이터 기반  
티켓 할인 예매 사이트



# TICKET PONG

Performers

황인식 손지연 양재식 박우정 이예리 송운정

When

2024.03.13 - 2024. 04. 17



Ticket Pong

FRONT-END DEVELOPMENT PROJECT



# Contents

---

**01.**

## 프로젝트 기획

- 선정 이유
- 기대 효과

**02.**

## 진행 프로세스

- 개발 일정
- 개발 환경

**03.**

## 프로젝트 상세

- 사이트맵/워크플로우
- 시스템 아키텍처

**04.**

## 서버 설계

- ERD
- 서버 구성

**05.**

## 프로젝트 시연

- 시연 영상

**06.**

## 프로젝트 기능

- 기능 소개
- 코드 설명

## Overview

티켓 예매 사이트의 독점 방지를 위한 예매사이트 분산의 필요성과 편의성을 목표로 시작한 프로젝트로,  
문화생활에 대한 접근성을 높일 수 있는 정보 집약적 서비스를 구현하였습니다.

01

안전한 예매를 위한  
사용자 인증 기기 등록



02

저렴한 관람을 위한  
카드 별 할인 혜택 제공



03

접근성이 용이한  
정보 집약적 서비스



04

위치 기반 서비스로  
지역별 공연 마케팅 효과



# Project Timeline



## Dev Tools

---

### 프론트엔드

HTML, CSS, JavaScript, React



---

### 백엔드

Node.js + express, MySQL, MariaDB



---

### 협업툴

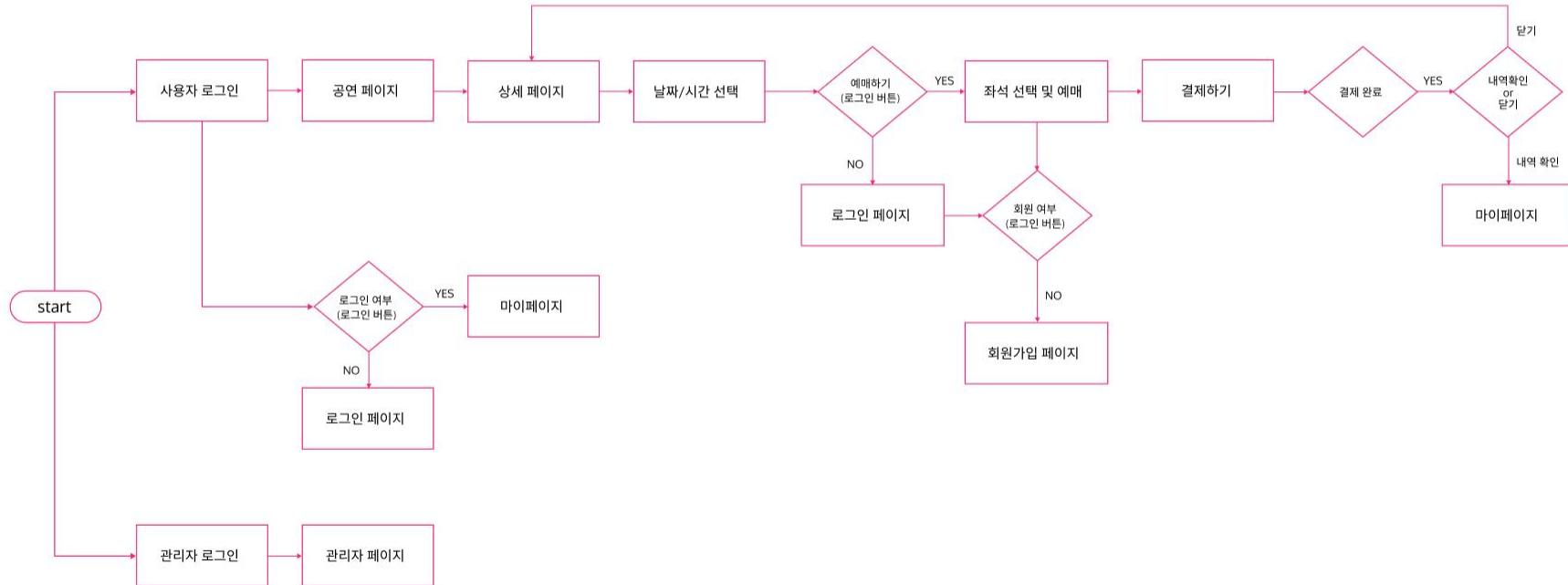
VSCode, GitHub, Figma



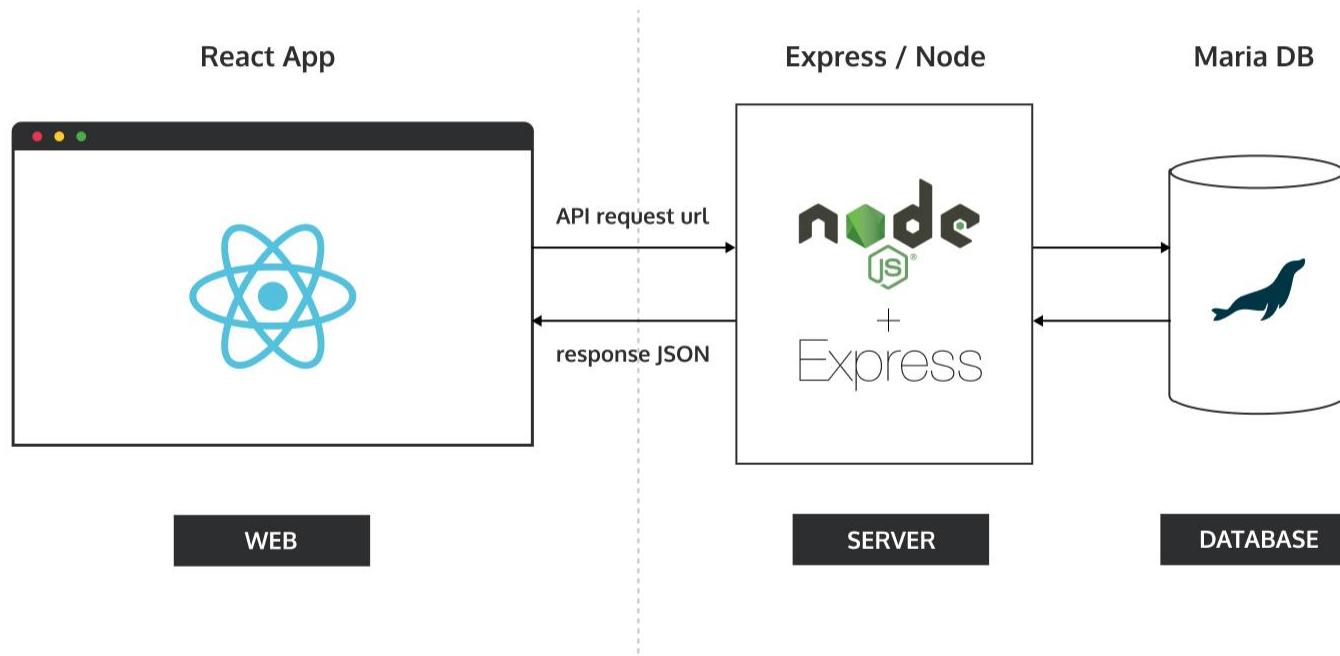
## < Site Map >



## < Work Flow >



## < Architecture >



# < ERD 설계 >

- DEVICES : 인증기기 테이블

PK - device\_id FK - user\_id

- MEMBER : 회원 정보 테이블

PK - user\_id

- MANAGE : 관리자 정보 테이블

PK - manage\_id

- RESERVATION : 예매 정보 테이블

PK - imp\_uid FK - mt20id, manage\_id, mt10id, user\_id

- DISCOUNT : 할인율 테이블

PK - code

- PERFORMANCE : 공연 정보 테이블

PK - mt20id FK - manage\_id, mt10id

- PERFORMANCEHALL : 공연시설 테이블

PK - mt20id

- REVIEW : 후기 정보 테이블

PK - pre\_id FK - imp\_uid, mt20id, manage\_id, mt10id, user\_id

- RECOMMAND : 추천 정보 테이블

PK - recommand\_id FK - pre\_id, user\_id

- BOXOFFICE : 예매상황판 테이블

PK - mt20id FK - mt20id2

DEVICES : 인증기기			
기기ID	device_id	문자	VARCHAR(255)
아이디	user_id	문자	VARCHAR(255)
기기명	device_name	문자	VARCHAR(255)
macaddress	macaddress	문자	VARCHAR(255)
등록일시	reg_date	일시	DATE

DEVICES : 인증기기

MEMBER : 회원			
아이디	user_id	문자	VARCHAR(255)
이름	user_name	문자	VARCHAR(255)
비밀번호	user_password	문자	VARCHAR(255)
이메일	user_email	문자	VARCHAR(255)
휴대폰번호	user_phone	문자	VARCHAR(255)
주소	address	문자	VARCHAR(255)
상세주소	detailAddress	문자	VARCHAR(255)

MEMBER : 회원

MANAGE : 관리자			
아이디	manage_id	문자	VARCHAR(255)
이름	manage_name	문자	VARCHAR(255)
비밀번호	manage_password	문자	VARCHAR(255)
화면번호	manage_phone	문자	VARCHAR(255)
권한	manage_auth	문자	VARCHAR(255)
직급	manage_part	문자	VARCHAR(255)

MANAGE : 관리자

DISCOUNT : 할인율			
코드	code	문자	VARCHAR(255)
카드사	card_name	문자	VARCHAR(255)
할인율	discountRate	숫자	INT

DISCOUNT : 할인율

RESERVATION : 예매정보			
예매ID	imp_uid	문자	VARCHAR(255)
승인ID	mt20id	문자	VARCHAR(255)
아이디	manage_id	문자	VARCHAR(255)
공연ID	mt10id	문자	VARCHAR(255)
예약시작ID	mt10id	문자	VARCHAR(255)
예약종료ID	mt10id	문자	VARCHAR(255)
아이디	user_id	문자	VARCHAR(255)
예약일자	reg_date	일시	DATE
예약시간	reg_time	시간	TIME
선택시작	selecttime_start	일시	TIME
선택종료	selecttime_end	일시	TIME
선택인원	people	숫자	INT

RESERVATION : 예매정보

REVIEW : 후기			
후기ID	pre_id	문자	VARCHAR(255)
제작사ID	imp_uid	문자	VARCHAR(255)
공연ID	mt20id	문자	VARCHAR(255)
아이디	managed_id	문자	VARCHAR(255)
공연시작ID	mt10id	문자	VARCHAR(255)
아이디	user_id	문자	VARCHAR(255)
제작사	prestitle	문자	VARCHAR(255)
내용	content	문자	TEXT
등록일자	predate	일시	DATETIME
별점수	prestar	문자	VARCHAR(20)
추천수	recommend	숫자	INT

REVIEW : 후기

RECOMMAND : 추천			
추천ID	recommended_id	문자	VARCHAR(255)
후기ID	pre_id	문자	VARCHAR(255)
아이디	user_id	문자	VARCHAR(255)
추천여부	recommended_state	불린	BOOLEAN

RECOMMAND : 추천

PERFORMANCE : 공연			
공연ID	mt20id	문자	VARCHAR(255)
아이디	manage_id	문자	VARCHAR(255)
공연시작ID	mt10id	문자	VARCHAR(255)
제작사ID	pre_id	문자	VARCHAR(255)
제작사	prfName	문자	VARCHAR(255)
장르	genre	문자	VARCHAR(255)
장소	place	문자	VARCHAR(255)
개인장장	prfIndiv	불린	BIT
단체장장	prfGroup	불린	BIT
티켓판매가	prfTicket	불린	BIT
장장	genre	문자	VARCHAR(255)
관람장장	updateDate	문자	VARCHAR(255)
장장수정	updateTime	일시	TIME
장장상태	status	문자	VARCHAR(255)
장장설명	desc	문자	VARCHAR(255)
장장별명	alias	문자	VARCHAR(255)
장장별명	telno	문자	VARCHAR(255)
장장주소	addr	문자	VARCHAR(255)

PERFORMANCE : 공연

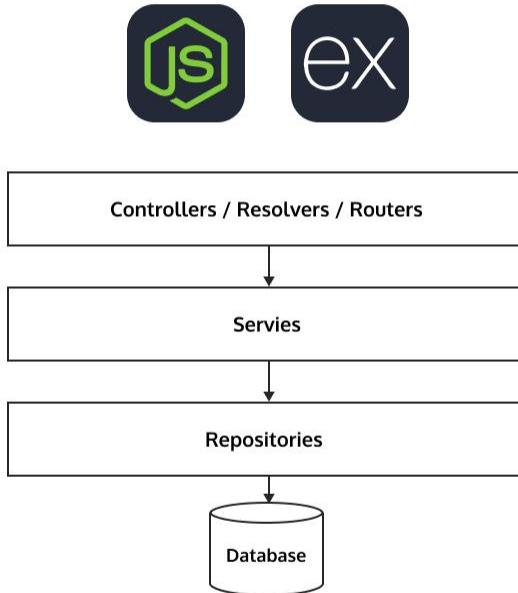
BOXOFFICE : 예매상황판			
승리ID	mt20id	문자	VARCHAR(255)
승리ID	mt20id2	문자	VARCHAR(255)
승리	rum	숫자	INT
승리	prfCount	문자	VARCHAR(255)

BOXOFFICE : 예매상황판

PERFORMANCEHALL : 공연시설			
공연시설ID	mt20id	문자	VARCHAR(255)
공연시설명	chName	문자	VARCHAR(255)
위도	lo	문자	VARCHAR(255)
경도	la	문자	VARCHAR(255)
전화번호	telno	문자	VARCHAR(255)
시도	sidonm	문자	VARCHAR(255)
주소	addr	문자	VARCHAR(255)

PERFORMANCEHALL : 공연시설

# Server



## 유저 요청

요청 응답

비즈니스로직, 바디,  
데이터 전달받음

DB연결, 쿼리 커넥션

```

app.use("/main", mainRouter);

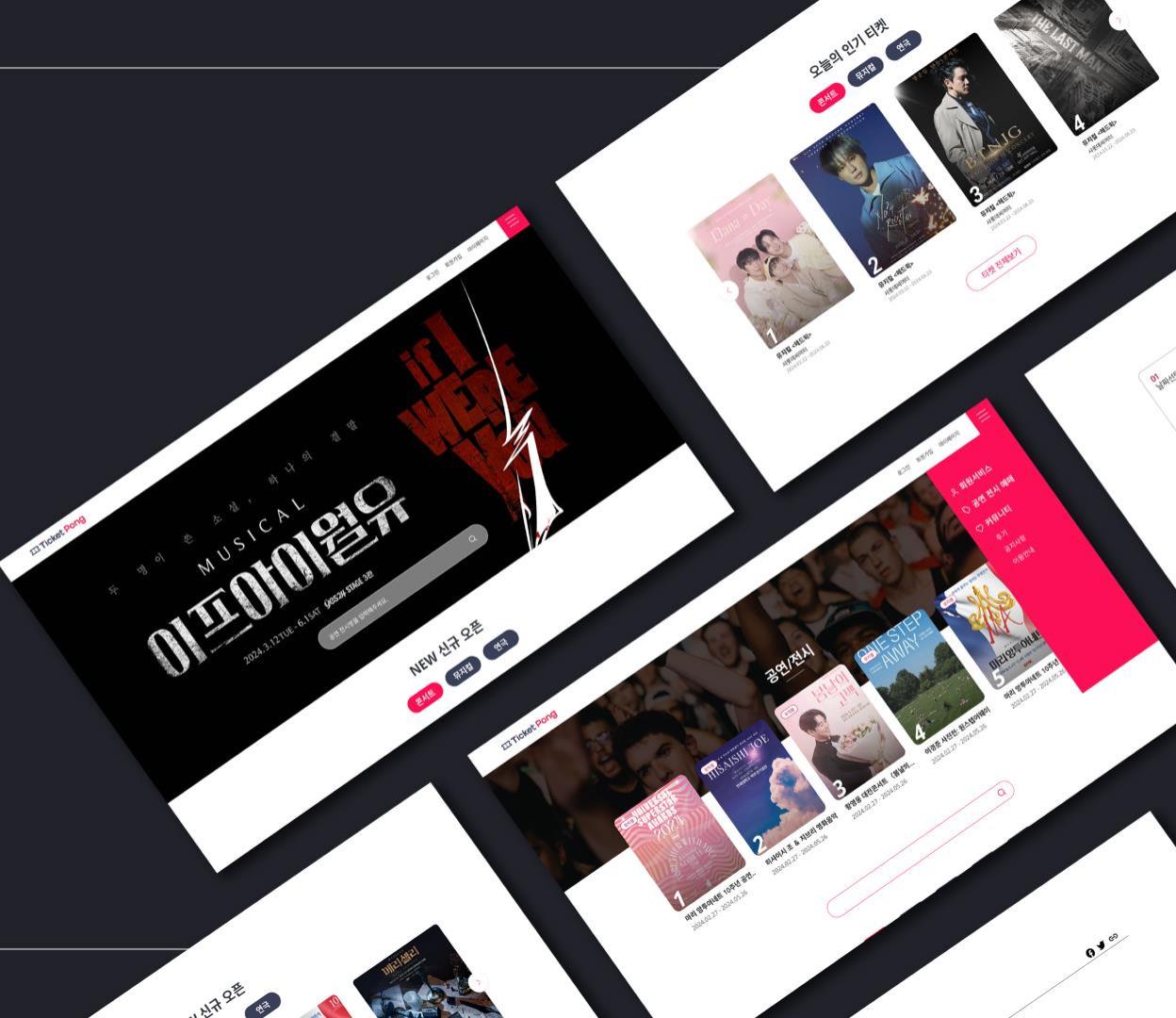
// post member main page
router.post("/member", memberMain.postMember);

// member postmain page
const postMember = async (req, res) => {
  const { id } = req.body;
  try {
    const result = await memberService.postMember(id);
    if (result) {
      res.status(200).json(result);
    } else {
      res.status(400).send("member main post failed");
    }
  } catch (error) {
    console.log(error);
    res.status(500).send("member main post failed");
  }
};

// member postmain page
const postMember = (id) => {
  const sql = `SELECT * FROM member WHERE user_id = ?`;
  let params = [id];

  return new Promise((resolve, reject) => {
    dbconn.db.query(sql, params, (err, rows) => {
      if (err) {
        console.log(err);
        resolve(false);
      } else {
        resolve(rows[0]);
      }
    });
  });
};
  
```

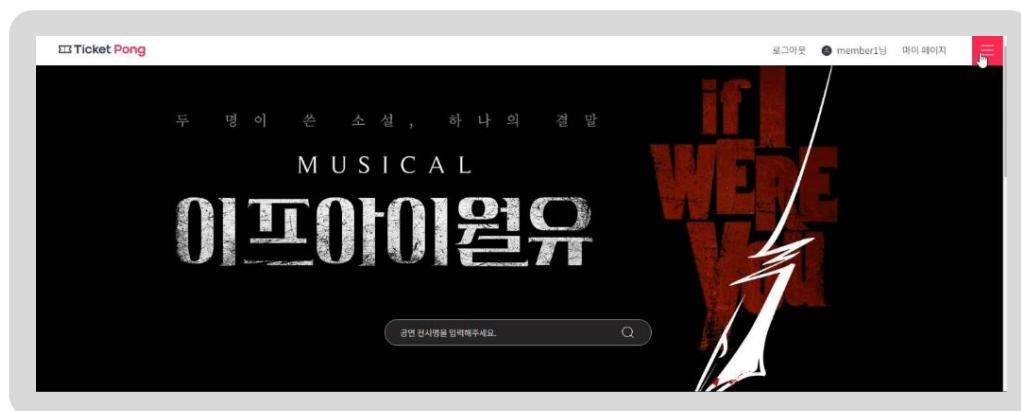
# Feature Detail



## 기능 핵심 코드

### < Header 기능 설명 >

- **useEffect**와 **axiosWithAuth**를 이용, 사용자 인증 토큰을 포함하여 로그인 구현
- 로그아웃 버튼 클릭 시 **handleLogout** 함수 실행
- mousedown 이벤트가 발생했을 때 NavMenu가 오픈되도록 이벤트 리스너 등록

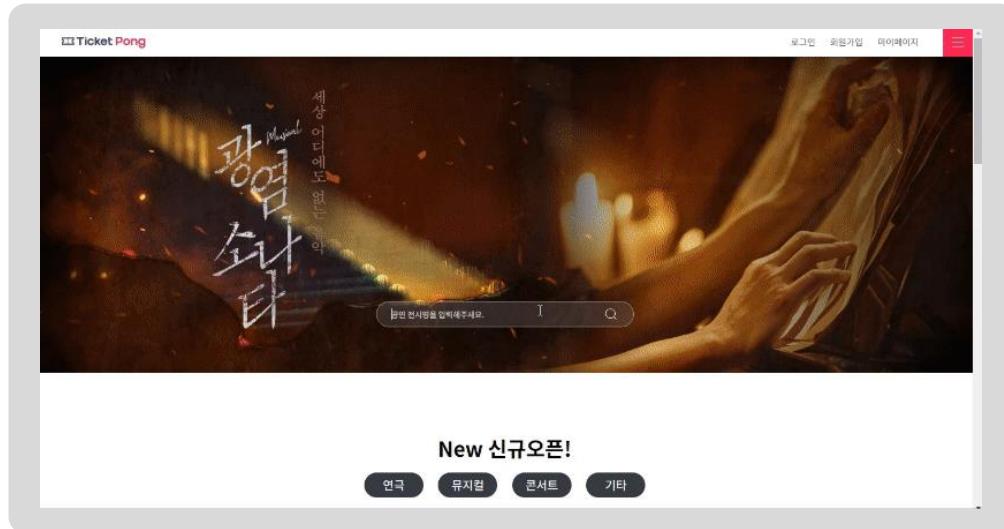


```
useEffect(() => {
  const fetchLoginStatus = async () => {
    try {
      const response = await axiosWithAuth().get(
        "http://localhost:8080/login/profile"
      );
      const { id, isLoggedIn } = response.data;
      if (isLoggedIn) {
        setUserId(id);
        setIsLoggedIn(true);
      }
    } catch {
      console.log("로그인 상태를 확인하는 동안 오류 발생:");
    }
  };
  fetchLoginStatus();
}, []);
```

```
const handleLogout = async () => {
  try {
    const response1 = await axios.get("http://localhost:8080/logout");
    const response2 = await axios.get(
      "http://localhost:8080/manage/manageLogout"
    );
    if (response1.status === 200 && response2.status === 200) {
      setIsLoggedIn(false);
      setUserId("");
      localStorage.removeItem("token");
      navigate("/");
    } else if (response1.status === 500 && response2.status === 500) {
      console.error("로그아웃 요청 실패:", response2.statusText);
    } else {
      console.error("서버에서 오류가 발생했습니다.");
    }
  } catch (error) {
    console.error("로그아웃 요청 중 에러 발생:", error);
  }
};
```

## < 배너 슬라이드 & 검색창 기능 설명 >

- isActive와 setInterval을 이용하여 이미지 페이드 효과 구현
- 검색어 입력시 keyword가 업데이트되며 엔터/아이콘 클릭 시 handleSearch 함수 호출
- fetch 함수를 사용하여 서버에 요청을 보낸 후 response.json으로 변환



```
const SearchResult = () => {
  const location = useLocation();
  const queryParams = new URLSearchParams(location.search);
  const keyword = queryParams.get("keyword");
  const [searchResults, setSearchResults] = useState([]);

  useEffect(() => {
    if (keyword) {
      fetch(`http://localhost:8080/searchBar?keyword=${keyword}`)
        .then((response) => {
          if (!response.ok) {
            throw new Error("검색 결과 가져오기 실패");
          }
          return response.json();
        })
        .then((searchData) => {
          setSearchResults(searchData);
        })
        .catch((error) => {
          console.error("검색 결과를 가져오는 중에 오류 발생:", error);
        });
    }
  }, [keyword]);
```

```
const SearchBar = ({ isHomepage }) => {
  const navigate = useNavigate();
  const [keyword, setKeyword] = useState("");

  const handleChange = (event) => {
    setKeyword(event.target.value);
  };

  const handleSearch = () => {
    if (keyword.trim()) {
      navigate(`/searchresult?keyword=${keyword}`);
    }
  };

  const handleKeyDown = (event) => {
    if (event.key === "Enter") {
      handleSearch();
    }
  };
};
```

## 기능 핵심 코드

### < 카테고리 슬라이더 기능 설명 >

- 모든 데이터는 `useEffect`와 `axios`를 통해 MariaDB에 연결하여, URL을 통해 출력
- 데이터 내부에서 카테고리를 가져왔을 때, `getDisplayedData` 함수를 통해 버튼별로 분류
- `categoryButton0`이 눌릴 때, `handleCategoryChange` 함수를 실행하며, 분류별로 출력

The screenshot shows a user interface for movie recommendations. At the top, there's a banner with the text "New 신규오픈!" and four buttons: 연극 (highlighted in red), 뮤지컬, 콘서트, and 기타. Below the banner, there are two movie cards on the left and three on the right, each with a thumbnail, title, and showtimes.

Category	Title	Showtimes
연극	영화 <아버지자를 위하여>	2024.03.28 ~ 2024.04.07
연극	영화 <타오르는 여름 ...>	2024.03.28 ~ 2024.03.31
연극	영화 <제7회 중국희곡...>	2024.03.29 ~ 2024.03.30
원작	영화 <나는 반금련이 아니야>	2024.03.29 19:30 2024.03.30 19:30 2024.03.31 19:30
원작	영화 <국립극단 명동예술극장>	2024.03.29 19:30 2024.03.30 19:30
원작	영화 <국립극단 명동예술극장>	2024.03.30 19:30

```

const handleCategoryChange = (categoryName) => {
  setCategory(categoryName);
  setStartIndex(0);
};

const getDisplayedData = () => {
  let filteredData;
  if (category === "기타") {
    filteredData = data.filter(
      (item) => item.genremm === "무용" || item.genremm === "서양 음악(클래식)"
    );
  } else {
    filteredData = data.filter((item) => item.genremm === categoryName) || [];
  }
}

const totalDataLength = filteredData.length;

if (totalDataLength <= 4) {
  return filteredData.slice(startIndex);
} else {
  return filteredData.slice(startIndex, startIndex + 4) || [];
}
};

const displayedData = getDisplayedData();

```

```

const totalDataLength = data.filter(
  (item) =>
    item.genremm === category ||
    (category === "기타" &&
      (item.genremm === "무용" || item.genremm === "서양 음악(클래식)"))
).length;
const maxIndex = Math.ceil(totalDataLength / 4) - 1;

if (startIndex + 4 < totalDataLength) {
  setStartIndex(startIndex + 4);
} else if (
  startIndex + 4 >= totalDataLength &&
  startIndex === maxIndex * 4
) {
  setStartIndex(maxIndex * 4);
} else if (totalDataLength <= 4) {
  setStartIndex(startIndex);
}
};

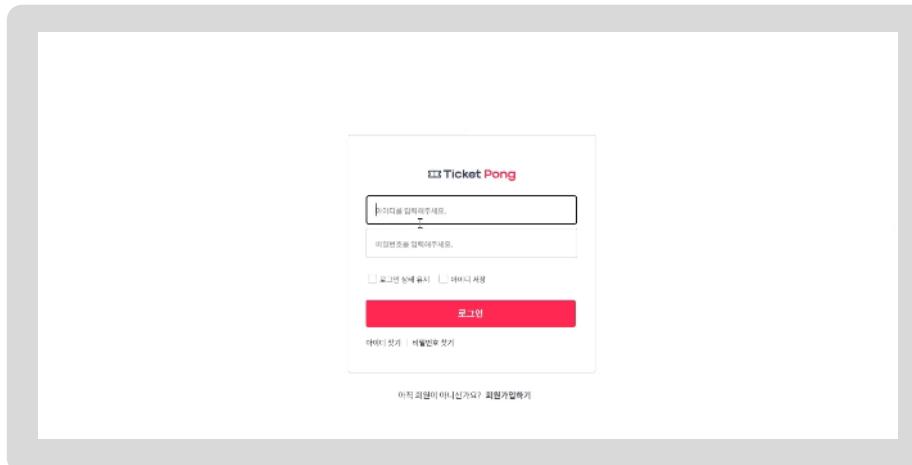
const handlePrev = () => {
  if (startIndex - 4 >= 0) {
    setStartIndex(startIndex - 4);
  }
};

```

## 기능 핵심 코드

### <로그인 기능 설명>

- useState와 axios로 로그인 내용을 backend로 전달
- 로그인 버튼 클릭시 backend 전송후 성공 시 token을 생성
- Token 생성후 localStorage에 isLoggedIn, userId를 토큰으로 저장



```

const LoginPage = () => {
  const [inputValue, setInputValue] = useState({
    id: '',
    pw: ''
  });

  const inputChangeHandler = (e) => {
    const { name, value } = e.target;
    setInputValue({
      ...inputValue,
      [name]: value,
    });
  };

  const submit = async () => {
    try {
      const response = await axios.post(url, {
        id: inputValue.id,
        pw: inputValue.pw,
      });
      if (response.status === 200) {
        if (response.data === "login failed") {
          alert("로그인 실패");
          window.location.reload();
        } else {
          const { token } = response.data;
          localStorage.setItem("token", token); // 토큰 저장
          if (localStorage.getItem("token")) {
            alert("로그인 성공");
            window.location.href = "/";
          }
        }
      } else {
        alert("로그인 실패");
        window.location.reload();
      }
    } catch (error) {
      console.log(error);
    }
  };
}

```

## < 회원가입 기능 설명 >

- useState로 변경변수, 주소검색창, 아이디, 이메일 체크 되도록 구현
- react-daum-postcode로 주소 검색 구현
- 아이디/이메일 확인을 axios로 체크한 뒤, 회원가입이 가능하도록 구현

The screenshot shows a sign-up form for 'Ticket Pong'. The fields are as follows:

- 이름(이름): 허가님은 양문나~님입니다.
- 비밀번호: 허가님은 양문나~님입니다.
- 비밀번호 확인: 허가님은 양문나~님입니다.
- 이름(성): 허가
- 이름(한글): 허가
- 연락처: 010-1234-5678
- 이메일: example@gmail.com
- 주소: 00-000 서울시 강남구 강남대로 500
- 상세주소를 입력해주세요.

A large red button at the bottom is labeled '회원가입'.

```

const SignupPage = () => [
  const [inputValue, setInputValue] = useState({
    id: '',
    pw: '',
    repw: '',
    name: '',
    phone: '',
    email: '',
    address: '',
    detailAddress: '',
  });
  const [isDaumPostcodeOpen, setIsDaumPostcodeOpen] = useState(false);
  const [isIdCheck, setIsIdCheck] = useState(false);
  const [isEmailCheck, setIsEmailCheck] = useState(false);

  const inputChangeHandler = (e) => {
    const { name, value } = e.target;
    setInputValue({
      ...inputValue,
      [name]: value,
    });
  };

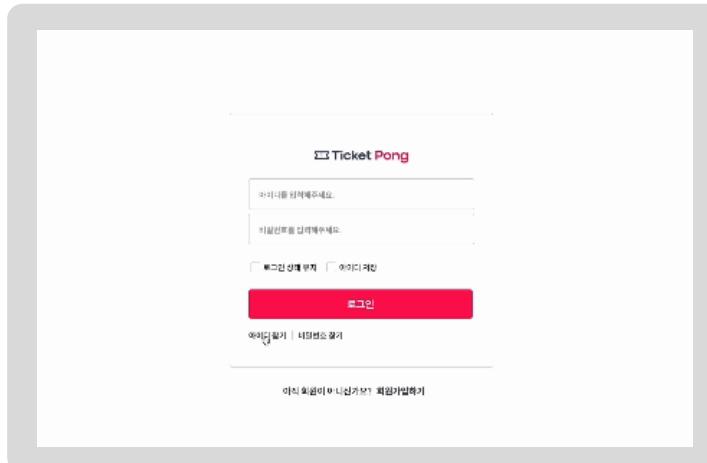
  const handleComplete = (data) => {
    const fullAddress = data.address;
    const extraAddress = data.addressType === "R" ? "" : data.bname;
    setInputValue({
      ...inputValue,
      address: fullAddress,
      detailAddress: extraAddress,
    });
    setIsDaumPostcodeOpen(false);
  };

  const openDaumPostcode = () => {
    setIsDaumPostcodeOpen(true);
  };
]

```

## < 아이디 / 비밀번호 찾기 기능 설명 >

- useState로 변수 설정
- axios로 백엔드와 연결 후 결과값 반환
- 결과값을 useNavigate로 페이지 이동과 함께 state로 결과값 전달
- 전달받은 값을 useLocation으로 받아와 변수에 저장



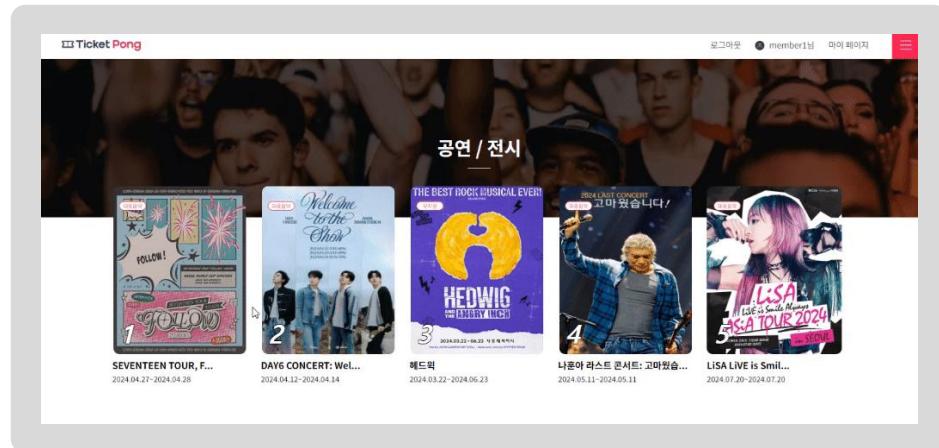
### 기능 핵심 코드

```
const navigate = useNavigate();
const handleSubmit = async (e) => {
  e.preventDefault();
  const response = await axios.post(
    "http://localhost:8080/findIdPassword/findId",
    {
      name: inputValue.name,
      email: inputValue.email,
    }
  );
  if (response.status === 200) {
    navigate("/confirmid", { state: { userId: response.data } });
  } else {
    alert("입력하신 정보가 일치하지 않습니다.");
    window.location.reload();
  }
};
```

```
const ConfirmId = () => {
  const location = useLocation();
  const userId = location.state.userId;
```

## < 박스오피스 순위 기능 >

- 모든 데이터는 useEffect와 axios를 통해 MariaDB에 연결하여, URL을 통해 출력
- 박스오피스 순위 테이블을 가져와서, 순위별로 순서대로 출력



## 기능 핵심 코드

```
const Performance = () => [
  const URL = "http://localhost:8080/viewwall";
  const rankURL = "http://localhost:8080/viewwall/ranking";
  const [startIndex, setStartIndex] = useState(0);
  const [allPerformances, setAllPerformances] = useState([]); // 전체 공연 데이터
  const [rankedPerformances, setRankedPerformances] = useState([]); // 순위 데이터
  const [showAll, setShowAll] = useState(false);
  const [selectedCategory, setSelectedCategory] = useState("전체");
  const [geographyBased, setGeographyBased] = useState(false); // 위치 기반 필터링 여부 상태
  const [location, setLocation] = useState({
    loaded: false,
    coordinates: { lat: "", long: "" },
    error: null,
  });
  const [sortBy, setSortBy] = useState("recent");

  const handleSortBy = (event) => {
    setSortBy(event.target.value);
  };
}
```

```
const fetchRankData = async () => {
  try {
    const response = await axios.get(rankURL);
    setRankedPerformances(response.data);
  } catch (error) {
    console.log(error);
  }
};

const getRankDisplayedData = () => {
  return rankedPerformances.slice(startIndex, startIndex + 5) || [];
};

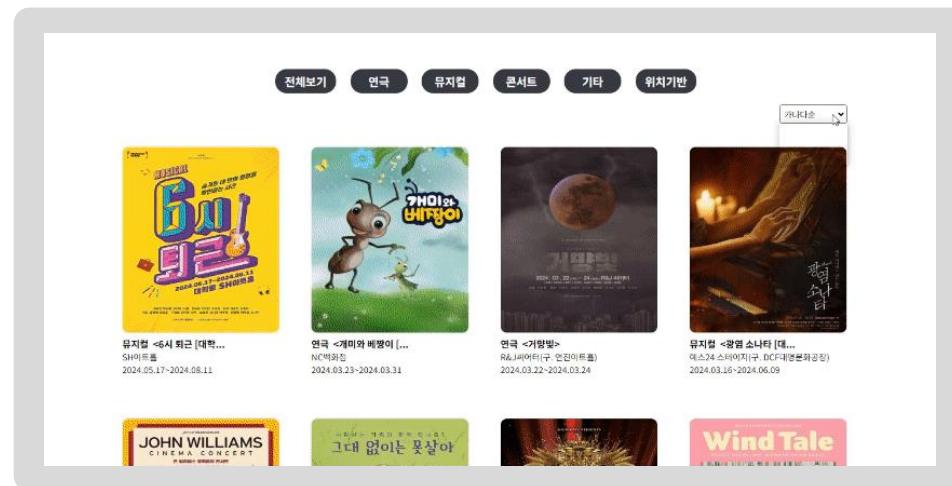
const getAllData2 = () => {
  return allPerformances || [];
};

const rankDisplayedData = getRankDisplayedData();
```

## 기능 핵심 코드

### <공연 카테고리 기능 설명 >

- 위치기반 필터링을 활성화하여 `handleGeographyChange` 함수를 호출
- 현재 사용자의 위치 정보(위도, 경도)를 가져오기 위해 브라우저의 **geolocation API**를 이용
- 선택한 카테고리에 따라 공연을 필터링하여 `getDisplayedData` 함수 호출
- 위치정보 버튼을 누를 경우, 위치 액세스를 할 경우, 사용자 주변 반경 11km 범위의 데이터를 제공
- 공연 데이터를 선택한 기준에 따라 보여주는 `filteredData`를 출력



```
useEffect(() => {
  if (!("geolocation" in navigator)) {
    onError({
      code: 0,
      message: "Geolocation not supported",
    });
  } else {
    navigator.geolocation.getCurrentPosition(onSuccess, onError);
  }
});

fetchData();
fetchRankData();
}, [1]);
```

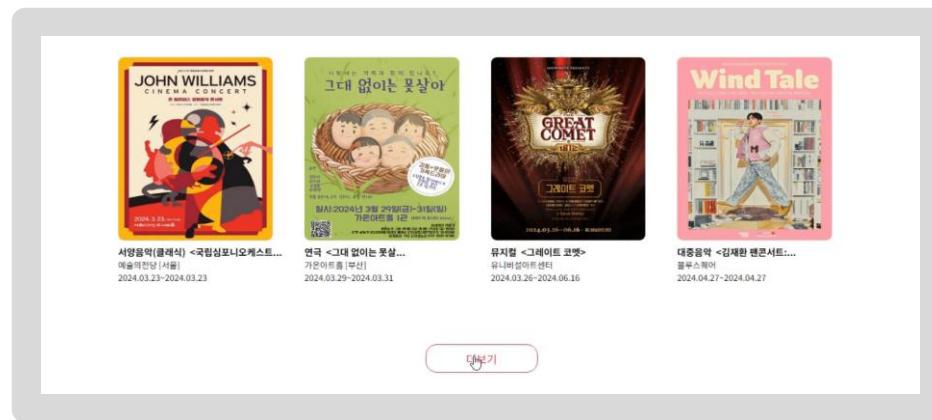
```
const getDisplayedData = () => {
  let filteredData = [];

  if (geographyBased) {
    filteredData = allPerformances.filter(item => {
      const distance = Math.sqrt(
        Math.pow(item.lat - lat, 2) + Math.pow(item.lo - long, 2)
      );
      // 0.1km 대비 반경 11km 이내
      return distance <= 0.1;
    });
  } else if (selectedCategory === "연극") {
    if (startIndex === 0) {
      filteredData = getAllData2();
    } else {
      filteredData = allPerformances.slice(startIndex, startIndex + 8);
    }
  } else {
    if (selectedCategory === "뮤지컬") {
      filteredData = allPerformances.filter(
        item => item.genreNm === "무용" || item.genreNm === "서양음악(클래식)"
      );
    } else {
      filteredData = allPerformances.filter(
        item => item.genreNm === selectedCategory
      );
    }
  }
}

return filteredData || [];
```

## < 더보기 버튼 및 데이터 정렬 기능 설명 >

- 출력하는 데이터의 갯수를 제한해놓고, 더보기버튼 클릭 시 `handleLoadMore` 함수 실행
- 출력된 모든 데이터들은 `handleSortBy` 함수에 의해 가나다, 등록순, 종료일 순으로 정렬



## 기능 핵심 코드

```
const handleCategoryChange = (category) => {
  setSelectedCategory(category);
  setStartIndex(0);
  setGeographyBased(false);
};

const handleGeographyChange = () => {
  setGeographyBased(true);
  setSelectedCategory("전체"); // 위치 기반 필터링 시 선택된 장르를 '전체'로 변경
};

const sortedData = [...displayedData].sort((a, b) => {
  if (sortBy === "prfm") {
    return a.prfm.localeCompare(b.prfm);
  } else if (sortBy === "prfpdfrom") {
    return new Date(a.prfpdfrom) - new Date(b.prfpdfrom);
  } else {
    return new Date(a.prfpdto) - new Date(b.prfpdto);
  }
});

const handleLoadMore = () => {
  setShowAll(true);
};
```

```
<S.LabelContainer>
  <select id="sortby" value={sortBy} onChange={handleSortBy}>
    <option value="prfm">가나다순</option>
    <option value="prfpdfrom">최근 등록순</option>
    <option value="prfpdto">종료일 놓은순</option>
  </select>
</S.LabelContainer>
```

## < 공연 스케줄 처리 설명 >

- 예시와 같은 데이터가 들어오면 각 요일과 시간으로 분리
- 날짜 범위가 '~'로 표기된 경우 인덱스 차이를 이용해 요일 설정
- 괄호 내부 시간 데이터 추출

The screenshot shows a ticketing interface. On the left is a poster for a performance titled '광업 속사타'. To its right is a detailed description of the event:

- 공연일자 2024.03.16 ~ 2024.06.09
- 시간 화요일(20:00), 수요일(16:00,20:00), 목요일 ~ 금요일(20:00), 토요일(15:00,19:00), 일요일(14:00,18:00), HOL(14:00,18:00)
- 장소 예스24 스테이지(구. DCF대명문화공장)
- 관람시간 1시간 40분
- 관람등급 만 13세 이상
- 문의 ☎ 02-742-9637
- 티켓가격 R석 77,000원, S석 66,000원

Below this is a schedule selection form:

01 날짜 선택	2024. 04 05	02 시간 선택	예매 가능 좌석 20시 00분																																									
<table border="1"> <tr><td>Su</td><td>Mo</td><td>Tu</td><td>We</td><td>Th</td><td>Fr</td><td>Sa</td></tr> <tr><td>31</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> <tr><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td></tr> <tr><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td></tr> <tr><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td></tr> <tr><td>28</td><td>29</td><td>30</td><td>1</td><td>2</td><td>3</td><td>4</td></tr> </table>		Su	Mo	Tu	We	Th	Fr	Sa	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	1	2	3	4	
Su	Mo	Tu	We	Th	Fr	Sa																																						
31	1	2	3	4	5	6																																						
7	8	9	10	11	12	13																																						
14	15	16	17	18	19	20																																						
21	22	23	24	25	26	27																																						
28	29	30	1	2	3	4																																						

```
// 공연 안내 정보에서 쉼표와 공백 제거
const replaced = dtguidance.replaceAll(", ", "");
// 각 섹션을 )을 기준으로 분할
const splitted = replaced.split(")");
// 섹션 배열 생성. 각 섹션에 )를 다시 추가하여 원래 형태로 복원
const sections = splitted.map((item, index) => {
  return item + ")";
});

const schedules = []; // 각 섹션의 스케줄 저장 배열
```

```
// 섹션에서 날짜 정보 추출
const daysStr = section.split("(")[0].trim();
let days = [];
if (daysStr.includes("~")) {
  // 날짜 범위가 지정된 경우
  const [startDay, endDay] = daysStr
    .split("~")
    .map((day) => day.trim());
  const startIdx = ["월요일", "화요일", "수요일", "목요일", "금요일", "토요일", "일요일"].indexOf(startDay);
  const endIdx = ["월요일", "화요일", "수요일", "목요일", "금요일", "토요일", "일요일"].indexOf(endDay);

  days = ["월요일", "화요일", "수요일", "목요일", "금요일", "토요일", "일요일", ].slice(startIdx, endIdx + 1);
}
```

```
// 괄호로 둘러싸인 일의 문자열 서치
const timesMatch = section.match(/\((.*?)\)/);
if (!timesMatch) return;
const timesStr = timesMatch[1];
const times = timesStr.split(",");
```

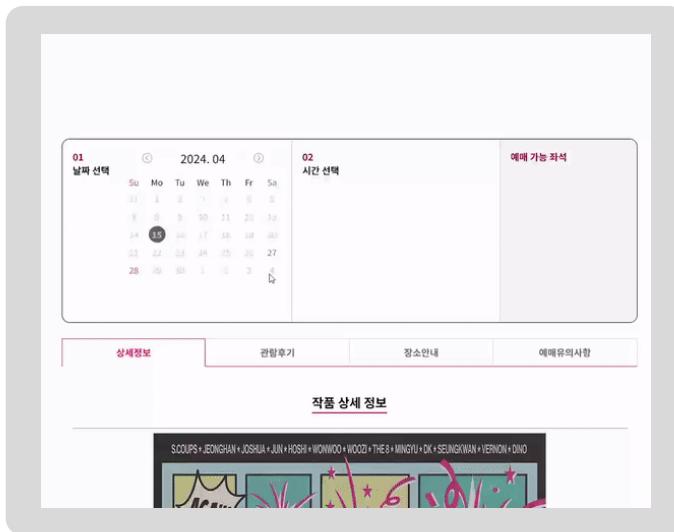
### 공연 데이터(공연시간) 예시

- 시간 화요일 ~ 금요일(19:30), 토요일 ~ 일요일(14:00,18:30), HOL(15:00)

## 기능 핵심 코드

### < 좌석 선택 및 예매 기능 설명 >

- 불러온 공연 데이터 기반 예매 가능 날짜와 시간 설정
- 기존 예매자 **회원정보 데이터와 카드 할인율 반영하여 결제 요청**
- 결제 요청 성공 시 콜백 함수 실행하여 DB에 예매 정보 저장



```

<DatePicker
  selected={selectedDate}
  onChange={(date) => {
    setSelectedDate(date);
    getShowTime(date);
  }}
  highlightDates={[dateFrom, dateTo]}
  inline
  minDate={dateFrom < dateToday ? dateToday : dateFrom}
  maxDate={dateTo}
/>

```

```

if (cardData) {
  // 할인되는 특정 카드 결제 시
  data.card = {
    direct: {
      code: cardData[0].code,
    },
  };
}

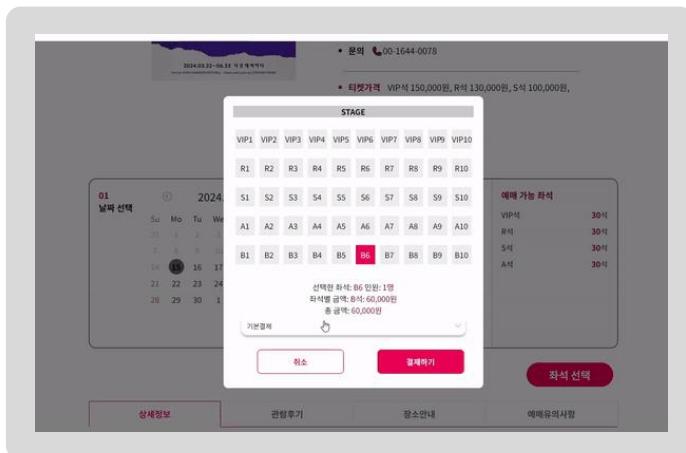
// 결제 요청 후 콜백 함수(DB 결제 내역 저장) 실행
IMP.request_pay(data, callback);

const callback = (response) => {
  if (response.success) {
    submitPayment(response);
    alert("결제 성공!");
    window.location.reload();
  } else {
    alert(`결제 실패! : ${response.error_msg}`);
  }
}

```

## < 예매 (기기 미등록/불일치 시) >

- 등록된 기기와 접속한 기기 불일치 시 예매 불가
- 기기 미등록 시 기기 등록 페이지로 이동 후 기기 등록  
(30일 이후 재등록 가능)



```

const checkMac = () => {
  if (userId && userValue.length > 0 && userValue[0].user_name) {
    if (macData) {
      onClickPayment();
    } else if (!macData) {
      alert("접속한 기기와 등록된 기기가 일치하지 않습니다!");
    } else {
      alert("기기 등록이 필요합니다.");
      navigate("/mypage#mac");
    }
  } else {
    if (!userId) {
      navigate("/login");
    } else {
      alert("잠시후 다시 시도해주세요.");
      window.location.reload();
    }
  }
};

const setMac = () => {
  const getDateStringDifference = (dateString) => {
    const today = new Date();
    const date2 = new Date(dateString);

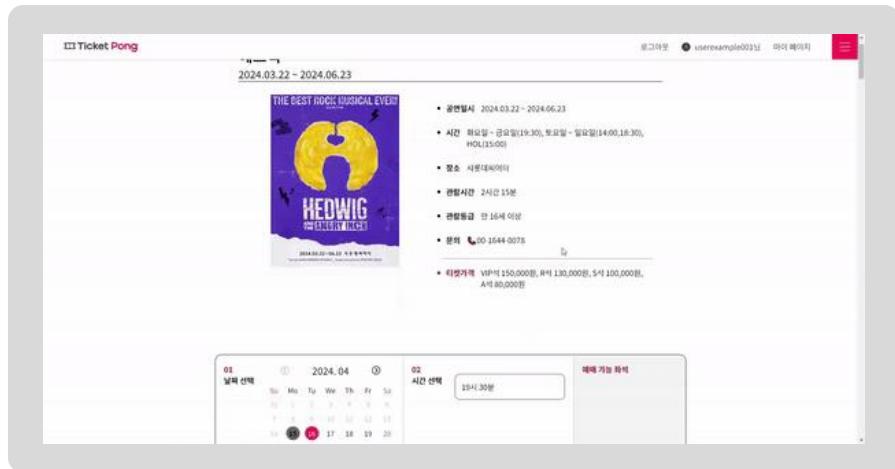
    const differenceInMs = Math.abs(today - date2);
    const differenceInDays = differenceInMs / (1000 * 60 * 60 * 24);
    return Math.floor(differenceInDays);
  };

  if (regiData) {
    if (getDateStringDifference(regiData.res_date) > 30) {
      updateMacInfo();
      alert("설정이 원료되었습니다!");
      window.location.reload();
    } else {
      alert(`${
        30 - getDateStringDifference(regiData.res_date)
      }일 후 재설정이 가능합니다.`);
    }
  } else {
    postMacInfo();
    alert("설정이 원료되었습니다!");
    window.location.reload();
  }
};

```

## < 공연 상세페이지 탭별 기능 설명 >

- 각 탭별 인덱스 부여
- 해당 공연의 리뷰 데이터 호출
- 해당 공연의 **공연 시설 위치, 경도 추출**하여 지도 표시 및 길찾기 기능 지원



```
(activeTab === 1 && selectedShowData && (
  <S.TabContentReview>
  | <S.TabContentWrapper>
  | | <TicketingReview mt20id={selectedShowData.mt20id} />
  | </S.TabContentWrapper>
  | </S.TabContentReview>
))
```

```
{activeTab === 2 && selectedShowData && (
  <S.TabContentWrapper>
  | <h2>공연장 안내</h2>
  | <hr />
  | <p>장소: {selectedShowData.fcltnm}</p>
  | <p>문의: {selectedShowData.telno}</p>
  | <PlaceMap mt10id={selectedShowData.mt10id} />
  | </S.TabContentWrapper>
)}
```

```
{dataPlace && (
  <>
  <Map>
    center={{ lat: dataPlace.la, lng: dataPlace.lo }}
    style={{|
      width: "900px", height: "500px", borderRadius: "20px",
      margin: "20px",
    |}}
  >
  <ZoomControl position={kakao.maps.ControlPosition.TOPRIGHT} />
  <MapMarker
    style={{ border: "transparent" }}
    position={{ lat: dataPlace.la, lng: dataPlace.lo }}
  ></MapMarker>
  </Map>
  <PongButton
    onClick={() => {
      window.open(
        `https://map.kakao.com/link/map/${dataPlace.fcltnm}.
        replace(
          `/[\uac00-\ud7a3()]/g,
          ...
        ), ${dataPlace.la}, ${dataPlace.lo}`
      );
    }}
  >
  길 찾기
  </PongButton>
  </>
)}
```

## < 커뮤니티 전체 기능 설명 >

- 공지사항 게시판, 리뷰 전체, 이용사항 안내 조회 가능
- URL을 통해 리뷰 데이터 테이블에 연결하여 추천수, 별점, 리뷰 내용을 출력
- 리뷰에 추천 수를 표시하여 선호도가 높은 리뷰를 메인 홈페이지에 출력
- **recommendHandler** 함수를 통해 로그인되어 있다면, 추천을 줄 수 있음

**커뮤니티**

공지 사항

번호	제목	작성일자
3	개인정보 처리방침 변경안내	2024-03-19
2	3월 시스템 점검 공지	2024-03-10
1	고객지원센터 2024년 4월 휴무일 안내	2024-03-01

« < 1 > »

!정보처리방침 | 청소년보호정책 | 이용안내 | 티켓판매안내  
2-2272-4679  
12:00 ~ 13:00 (주말/공휴일 휴무)

```
const CommuReview = () => {
  const [currentPage, setCurrentPage] = useState(1);
  const URL = "http://localhost:8888/review/recommendList";
  const [data, setData] = useState([]);

  useEffect(() => {
    fetchData();
  }, [currentPage]);

  const fetchData = async () => {
    try {
      const response = await axios.get(URL);
      const newData = response.data.map((item) => ({
        ...item,
        item
      }));
      setData(newData);
    } catch (error) {
      console.error(error);
    }
  };
}
```

```
useEffect(() => {
  const getRecommendInfo = async () => {
    try {
      const response = await axios.post(
        "http://localhost:8888/review/recommend",
        {
          pre_id: preId,
          user_id: userId,
        }
      );
      const newData = response.data;
      setRecommendState(newData);
    } catch (error) {
      console.error(error);
    }
  };
  getRecommendInfo();
}, [preId, userId]);
```

```
const recommendHandler = async () => {
  try {
    const response = await axios.post(
      "http://localhost:8888/review/checkRecommend",
      {
        pre_id: preId,
        user_id: userId,
      }
    );
    if (response.data === "recommend success") {
      setRecommendState(true);
      console.log("추천 성공");
    } else {
      setRecommendState(false);
      console.log("추천 해소");
    }
  } catch (error) {
    console.error(error);
  }
};
```

## 기능 핵심 코드

### < 예매 내역 확인 및 예매 취소 기능 설명 >

- 해당 **유저의 예매 데이터**를 서버에서 가져와 렌더링
- **예매 취소 시 오늘 날짜와 예매한 공연 날짜 비교 후 관람일 이전에만 취소 가능**
- 이미 결제 취소 되거나 취소 불가한 경우 처리

마이페이지

예매 내역

최대 지난 1년의 예매 내역을 확인할 수 있습니다.						
예매일	공연명	관람일	가격	매수	처리상태	후기상태
2024-04-15	SEVENTEEN TOUR, FOLL...	2024-04-28 17:00	285000원	4	결제완료	<button>관찰</button>

나의 관람 후기 >

회원 정보 수정 >

기기 등록/수정 >

보처리방침 | 청소년보호정책 | 이용안내 | 티켓판매안내  
072-4679  
00 - 13:00 (주말/공휴일 휴무)

```
// 예매일자와 현재 날짜 비교
const isCancelable = selectDateStr > currentDateStr;

const handleCancelClick = () => {
  setIsConfirmOpen(true);
};

const handleConfirm = () => {
  const cancelSubmit = async () => {
    if (data.success.data[0] === 0) {
      alert("이미 결제 취소된 예매입니다.");
    } else {
      try {
        const response = await axios.put(
          "http://localhost:8080/reservation/cancel",
          {
            imp_uid: data.imp_uid,
          }
        );
        if (response.status === 200 || response.status === 204) {
          alert("예매가 취소되었습니다.");
          window.location.reload();
        } else {
          alert("예매 취수가 불가합니다. 문의해주세요.");
          window.location.reload();
        }
      } catch (error) {
        console.error(error);
      }
    }
  };
  // 예매 취소 동작을 수행
  cancelSubmit();
}

[isCancelable && <Button onClick={handleCancelClick}>예매취소</Button>]
```

## 기능 핵심 코드

### < 공연 관람 후 후기 작성 기능 설명 >

- 관람일 이후 후기가 작성 되지 않았을 경우에만 작성 가능
- 해당 유저가 작성한 리뷰 데이터 인덱싱하여 각 리뷰별 렌더링

**마이페이지**

**예매 내역**

최대 지난 1년의 예매 내역을 확인할 수 있습니다.

예매일	공연명	관람일	가격	매수	처리상태	후기상태
2024-04-01	아비자금 위하여	2024-04-06 20:00	60000원	1	결제완료	<span>작성하기</span>
2024-04-15	SEVENTEEN TOUR, FOLL...	2024-04-28 17:00	285000원	4	결제완료	<span>관람진</span>
2024-04-15	SEVENTEEN TOUR, FOLL...	2024-04-27 18:00	285000원	4	결제완료	<span>관람진</span>
2024-03-12	페드릭	2024-03-22 19:30	170000원	1	결제완료	<span>작성완료</span>
2024-04-15	SEVENTEEN TOUR, FOLL...	2024-04-28 17:00	285000원	4	결제완료	<span>관람진</span>
2024-04-15	SEVENTEEN TOUR, FOLL...	2024-04-28 17:00	285000원	4	결제완료	<span>관람진</span>
2024-04-14	SEVENTEEN TOUR, FOLL...	2024-04-27 18:00	90000원	1	결제취소	<span>관람진</span>

« ‹ 1 2 › »

```

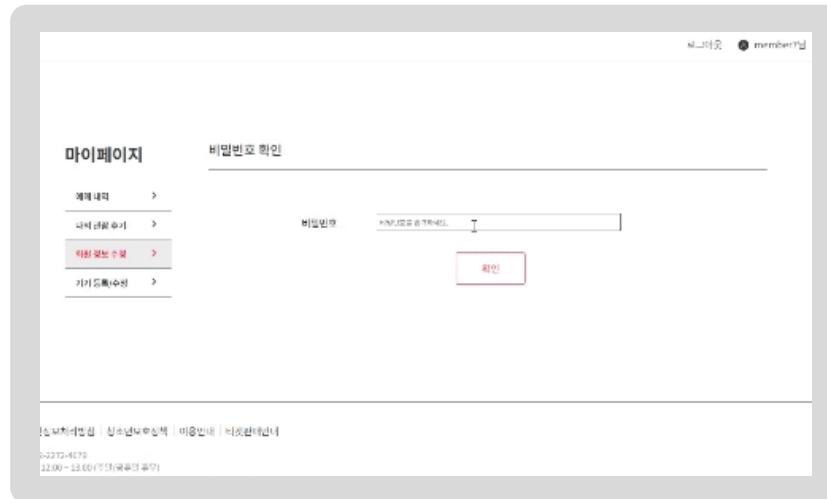
<ReviewStatus
  status={item.prestate.data[0]}
  selectdate={item.selectdate}
  onClick={() => {
    if (
      item.prestate.data[0] === 0 &&
      new Date(item.selectdate) < new Date()
    ) {
      navigate("/writereview", { state: [item] });
    }
  }}
>
{item.prestate.data[0] === 1
? "작성완료"
: new Date(item.selectdate) < new Date()
? "작성하기"
: "관람전"}
</ReviewStatus>

<Container>
{reviewList.length === 0 ? (
  <p>작성한 리뷰가 없습니다.</p>
) : (
  <ul>
    {reviewList?.slice(startIndex, endIndex).map((item, index) => (
      <HrBox key={index}>
        <ListItem key={index}>
          <div className="imageContainer">
            <Link
              to={`/reviewdetail/${item.pre_id}`}
              state={{ preId: item.pre_id }}
            >

```

## < 회원정보 수정 기능 설명 1 >

- useState로 변경되는 변수를 설정
- useEffect, axiosWithAuth함수를 이용하여 Token값 가져오기
- axiosWithAuth는 localStorage에 있는 Token값을 받아와 decode하는 함수



```
//useId, isLoggedIn 가져오기
useEffect(() => {
  fetchStatusLogin();
}, []);

const fetchStatusLogin = async () => {
  const response = await axiosWithAuth().get(
    "http://localhost:8080/login/profile"
  ); //로그인 상태 확인
  const { id, isLoggedIn } = response.data;
  if (isLoggedIn) {
    setUserId(id);
    setIsLoggedIn(isLoggedIn);
  } else {
    console.log("로그인 상태가 아닙니다.");
  }
};
```

```
const axiosWithAuth = () => {
  const token = localStorage.getItem("token");
  return axios.create({
    headers: {
      Authorization: `Bearer ${token}`,
    },
  });
};
```

## < 회원정보 수정 기능 설명 2 >

- useState로 변경되는 변수를 설정
- useEffect, axiosWithAuth함수를 이용하여 Token값 가져오기
- 비밀번호를 입력받아 backend로 전달, 처리 값 받기
- true일 경우에만 수정페이지로 id값과 함께 이동하도록 구현



### 기능 핵심 코드

```
 {!showEditProfile ? (
    <>
      <InputContainer>
        <InputLabel>비밀번호</InputLabel>
        <Input
          type="password"
          name="password"
          value={password}
          onChange={handleChange}
          placeholder="비밀번호를 입력하세요."
        />
      </InputContainer>
      <ButtonContainer>
        <Button type="submit" onClick={editConfirm}>
          확인
        </Button>
      </ButtonContainer>
    </>
  ) : (
    <EditProfile Id={userId} />
  )}
}
```

## < 회원정보 수정 기능 설명 3 >

- useState로 변경되는 변수를 설정
- **useEffect, axios**로 backend에서 id별 정보를 가져와 저장
- value 값에 나와야 하는 값들의 변수를 넣어 화면에 출력되도록 구현
- 삭제, 수정 버튼을 누르면 그 값들이 backend로 넘어가 처리되도록 구현



```
useEffect(() => {
  // 회원 정보 가져오기 로직
  fetchUserInfo();
}, [isLoggedin]);

const fetchUserInfo = async () => {
  const response = await axios.post("http://localhost:8080/main/member", {
    id: props.id,
  });

  const newData = response.data;
  setUserInfo(prevState) => ({
    ...prevState,
    id: newData.user_id,
    name: newData.user_name,
    phoneNumber: newData.user_phone,
    email: newData.user_email,
    address: newData.address,
    detailedAddress: newData.detailAddress,
  });
};

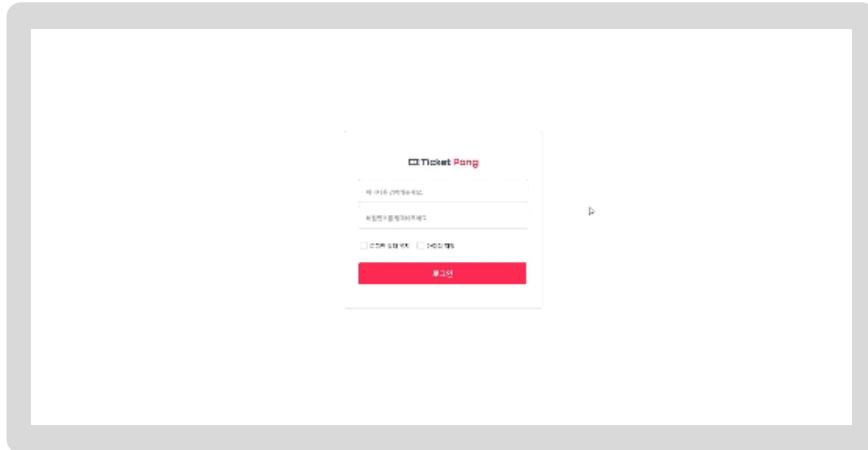
const deleteMember = async () => {
  // 회원 탈퇴 로직
  const response = await axios.delete("http://localhost:8080/main/delete", {
    id: userId,
  });

  if (response.status === 200) {
    alert("회원 탈퇴가 완료되었습니다.");
    setIsLoggedin(false);
    localStorage.removeItem("token");
  } else {
    alert("회원 탈퇴에 실패했습니다.");
  }
};
```

```
<InputContainer>
  <InputLabel>연락처</InputLabel>
  <GrayInput
    type="text"
    name="phoneNumber"
    value={userInfo.phoneNumber}
    readOnly
  />
</InputContainer>
```

## < 회원관리 화면 & Paging >

- **useEffect와 axiosWithAuth를 이용하여 로그인 핸들링**
- 회원 정보 리스트 출력
- **ITEMS\_PER\_PAGE 상수로 사용하여 데이터를 페이지별로 렌더링**



```
useEffect(() => {
  const fetchStatusLogin = async () => {
    try {
      const response = await axiosWithAuth().get("http://localhost:8080/manage/profile");
      const { id, isLoggedIn } = response.data;
      if (isLoggedIn) {
        setUserId(id);
        setIsLoggedIn(isLoggedIn);
      } else {
        console.log("로그인 상태가 아닙니다.");
        // 사용자가 로그인하지 않은 경우, 로그인 페이지로 이동
      }
    } catch (error) {
      console.error("로그인 상태 확인 중 오류가 발생했습니다.", error);
      window.location.href = '/manage'; // 로그인 페이지로 이동
    }
  };
});
```

```
/* 페이지네이션 버튼 */
<ButtonContainer>
  <button onClick={goToStartPage}>
    <MdKeyboardDoubleArrowLeft color="#999999" />
  </button>
  <button onClick={goToPrevPage}>
    <MdKeyboardArrowLeft color="#999999" />
  </button>
  <Array.from(
    { length: Math.ceil(data.length / ITEMS_PER_PAGE) },
    (_, i) => (
      <button key={i + 1} onClick={() => setCurrentPage(i + 1)}>
        {i + 1}
      </button>
    )
  )>
  <button onClick={goToNextPage}>
    <MdKeyboardArrowRight color="#999999" />
  </button>
  <button onClick={goToEndPage}>
    <MdKeyboardDoubleArrowRight color="#999999" />
  </button>
</ButtonContainer>
```

## < 공연 관리 리스트 기능 설명 >

- useState 흑을 사용하여 현재 페이지(currentPage)와 데이터를 관리
  - fetchData 함수를 호출하여 서버에서 공연 데이터를 가져옴 -----



기능 핵심 코드

```
const fetchData = async () => {
  try {
    const response = await axios.get(
      `http://localhost:8080/manage/manageMain/performance`
    );
    const newData = response.data.map((item, index) => ({
      ...item,
      number: (currentPage - 1) * 7 + index + 1,
    }));
    setData(newData);
  } catch (error) {
    console.error(error);
  }
};
```

```
ly>
.a.slice(startIndex, endIndex).map((item, index) => (
<tr>
  <Cell>{item.number}</Cell>
  <Cell>{item.mt20id}</Cell>
  <Cell>{item.prfnm}</Cell>
  <Cell>/item.genremm</Cell>
  <Cell>{item.prfpdfrom}</Cell>
  <Cell>{item.prfpdto}</Cell>
  <Cell>{item.prfstate}</Cell>
  <Cell>{item.post ? "y" : "n"}</Cell>
  <Cell>
    <Button onClick={() => onEditClick(item.mt20id)}>수정</Button>
    <Button onClick={() => performanceDelete(item.mt20id)}>삭제</Button>
  </Cell>
</tr>
)
</tbody>
</table>
</div>
```

## <공연 등록 화면 설명 >

- useState를 사용해 공연 정보, 로그인 상태를 관리
- 사용자가 "완료" 버튼을 클릭했을 때 **submit** 함수가 실행

### 기능 핵심 코드

```
const url = "http://localhost:8080/manage/manageMain/performanceAdd";

const submit = async () => {
  try {
    // 관리자 주가 처리 로직
    const response = await axios.post(url, {
      // 등록할 공연 정보
      mt20id: performance.mt20id,
      manage_id: manageId,
      mt10id: performance.mt10id,
      prfmi: performance.prfmi,
      prfpdfrom: performance.prfpdfrom,
      prfpdto: performance.prfpdto,
      prfruntime: performance.prfruntime,
      pcseguidance: performance.pcseguidance,
      genrenm: performance.genrenm,
      prfstate: performance.prfstate,
      poster: performance.poster,
      styuri: performance.styuri,
      dtguidance: performance.dtguidance,
      post: performance.post,
      prfage: performance.prfage,
    });
    json.stringify(response);
  } catch (error) {
    console.log(error);
  }
};
```

# Team Member

---



## 황인식

상세페이지(예매, 결제, 상세),  
마이페이지(예매 확인, 후기,  
사용자 기기 등록/수정),  
관리자페이지(후기), 총괄

## 손지연

DB설계/연결, 로그인, 회원가입,  
ID/PW 찾기 페이지(기능)  
관리자페이지(관리자 관리),  
회원정보 수정 페이지

## 양재식

메인페이지(슬라이드 배너/후기)  
전체보기 페이지(공연 리스트),  
커뮤니티(공지, 후기, 이용안내)

## 박우정

공연 장소, 공연, 예매상황판,  
박스오피스 DB설계,  
관리자페이지  
(회원관리, 공연관리)

## 이예리

디자인, PPT, Header, Footer,  
ID/PW 찾기 페이지(화면)  
메인페이지(상단배너+검색창)  
Search 결과 페이지

## 송운정

디자인 설계,  
스타일 가이드 작성

Thank you

---



# Q&A



---

for listening to the presentation