

POPBOB

팝업 스토어, 전시회 정보 제공 / 장소 대여 웹 서비스



KIC Campus Project

INDEX

- 
- 01 기획서
 - 02 개발 환경
 - 03 설계
 - 04 화면 구현
 - 05 시연
 - 06 소감

01 기획서

팀 소개

프로젝트 기획

프로젝트 일정

미션과 목표

About us



박*람

담당 : 팀장(일정 조율, 의견 조율 등), DB설계,
Chat GPT 구현



김*연

담당 : DB설계, Chat GPT 구현



조*현

담당 : DB작성, 정보 찾기



이*준

담당 : 로그인 / 회원가입, 정보찾기



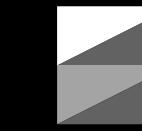
이*영

담당 : 전체 List, View Page



01 기획서

프로젝트 기획



01 기획서

사이트 정의

전시회, 팝업 스토어 정보 제공,
장소 대여 서비스 제공 플랫폼

타겟층

전시회, 팝업 스토어 정보를 얻고자 하는 2030,
개최를 원하는 기업 및 단체

사이트 목적

팝업, 전시회 정보와 더불어 행사 장소
임대 서비스로 기업 및 단체 고객 유치

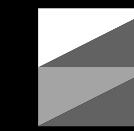
사이트 경쟁력

쉬운 접근성으로 팝업, 전시회 정보 확인

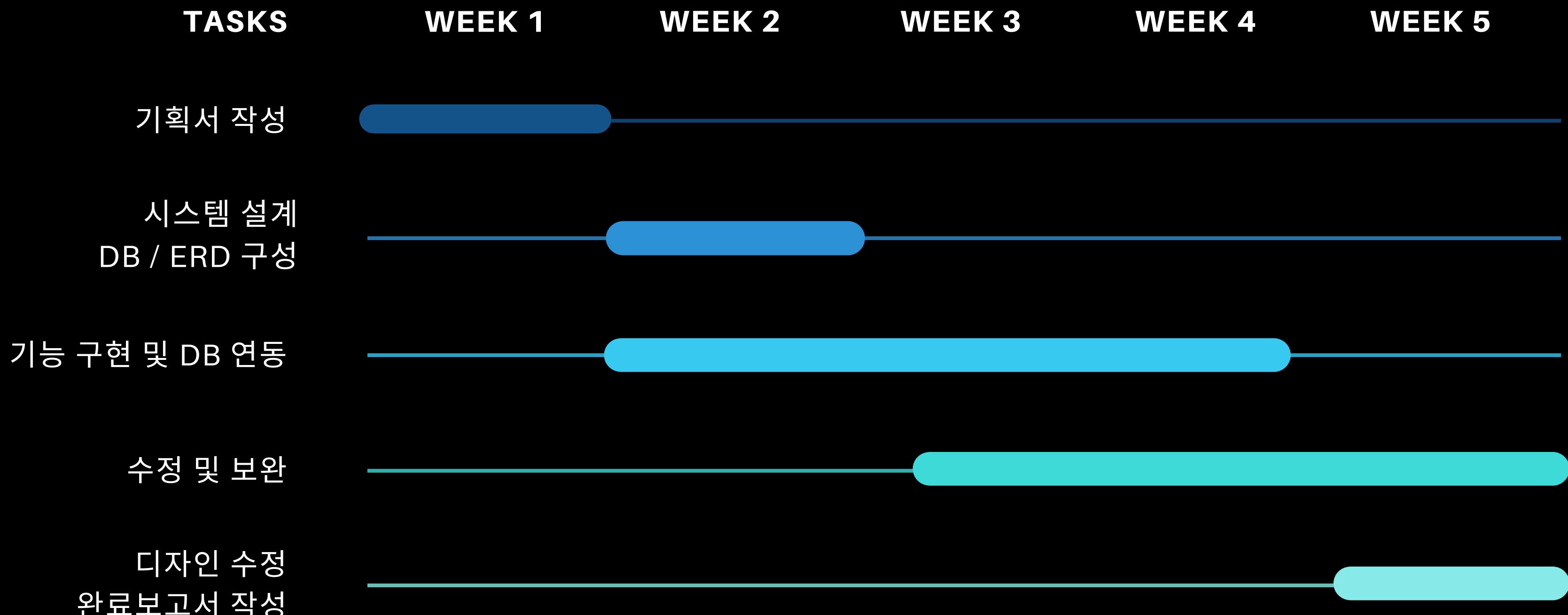
도메인

www.popbob.com

프로젝트 일정



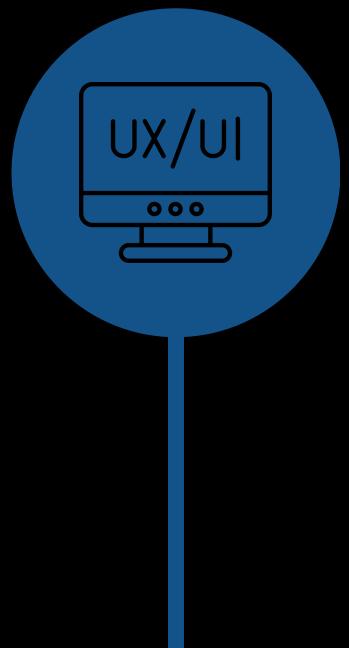
01 기획서



MISSION

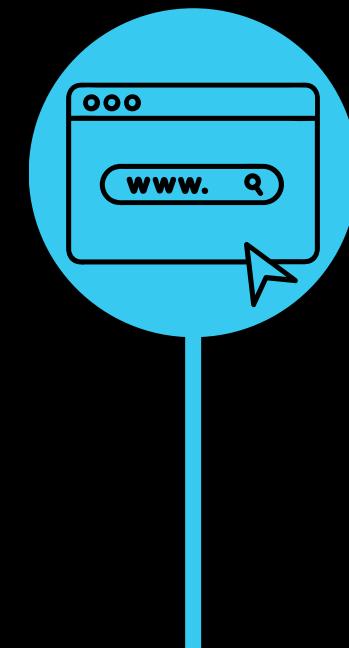
팝업스토어 / 전시회의 정보 제공 및 해당 행사 장소 임대

GOAL



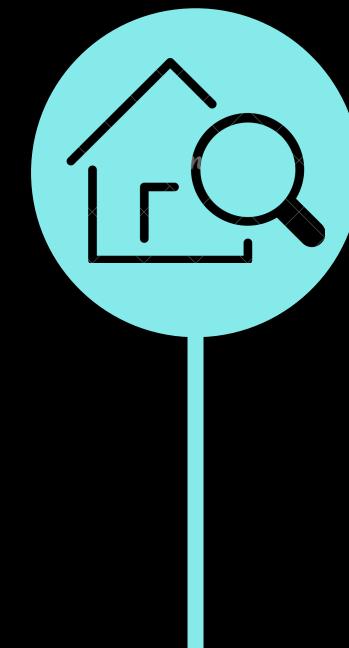
01

직관적이고 사용자
편의적인 UI 구현



02

각 행사의 개최 정보,
해당 장소 임대 정보 제공



03

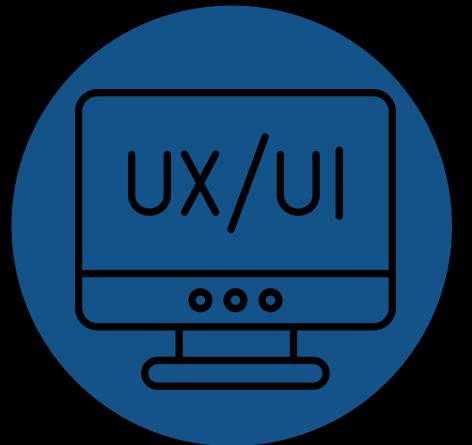
각종 행사가 개최될
장소를 임대할 수 있는
서비스 제공

프로젝트 목표



01 기획서

01



직관적이고 사용자
편의적인 UI 구현

배경

다양의 정보를 사용자에게 피로감 없이 전달해야 함
사용자가 원하는 정보를 쉽고 빠르게 얻을 수 있도록 도와야 함

주요 내용

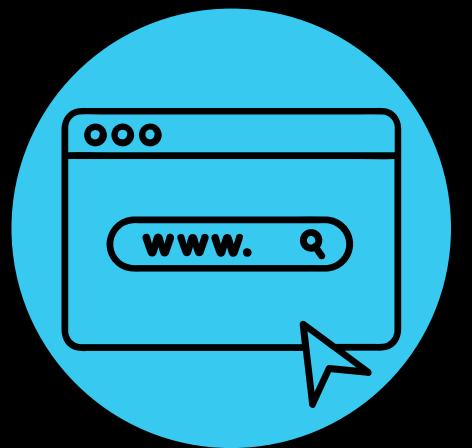
최소화 된 메뉴로 깔끔한 첫인상을 심어주는 메인 페이지
사이트 내의 메뉴 구성을 단일화 해 사용자들이 혼매지 않게 함
게시물의 썸네일과 같이 이미지를 활용해 정보가 한눈에 들어오게 함

기대 효과

깔끔하고 사용하기 편리한 사이트라는 인식을 줌
다양의 정보를 제공하는 사이트의 특성 상 줄 수 있는 피로감을 최소화 함



02



각 행사의 개최 정보,
해당 장소 임대 정보 제공

배경

팝업 스토어 / 전시회의 개최 정보와
해당 행사 대여 장소의 정보를 원하는 사이트 사용자들의 니즈 파악

주요 내용

한국문화정보원의 '공연전시 정보 조회 서비스 API'를 사용한 정보 제공
각종 행사 개최 장소의 대여 가능 여부, 해당 장소 예약 일정 제공

기대 효과

여러 행사의 개최 정보를 한 곳에서 모아 볼 수 있는 통합적 서비스 제공
팝업 스토어와 전시회 관련 소비자와 공급자 모두 편리한 정보 습득 가능



03



각종 행사가 개최될 장소를
임대할 수 있는 서비스 제공

배경

행사 개최 장소의 임대 / 대여를 원하는 기업 간의 B2B 서비스 전용 플랫폼

주요 내용

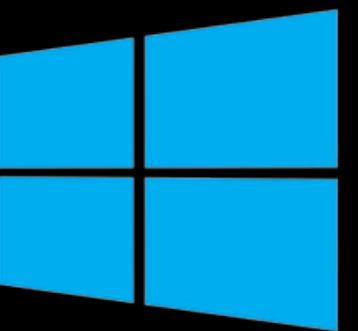
임대 가능한 장소의 위치 정보와 예약 가능한 시간대 등 일정 정보 제공
정보 제공에서 나아가 실제로 예약 가능한 시스템 구현
장소 예약 현황 확인 가능 서비스 제공

기대 효과

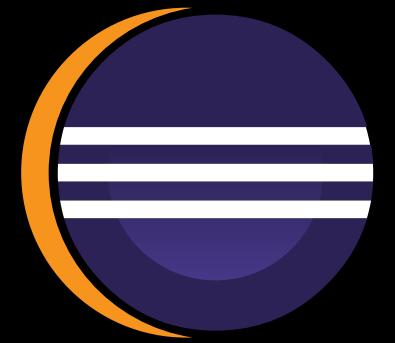
기업 간 편리한 임대 서비스 프로세스로 시간 및 비용 절감 효과
접근이 용이한 온라인 예약 서비스로 많은 서비스 이용자 유치에 유리

02 개발 환경

운영체제 / 개발 도구



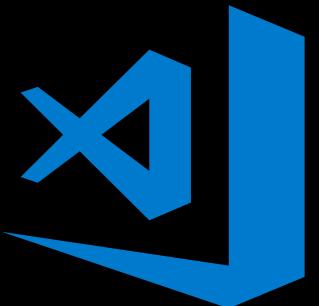
Windows 10



eclipse



macOS

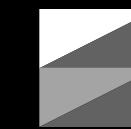


Visual Studio Code



02 개발환경

개발 언어 / 프레임워크 라이브러리



활용한 API

카카오 로그인

kakaomap

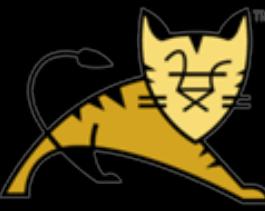
FullCalendar

CHAT GPT

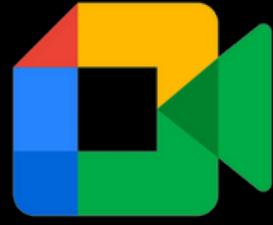


02 개발환경

웹서버 / DBMS / 협업 툴



DISCORD



Google Meet



GitHub



02 개발환경

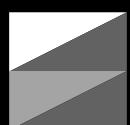
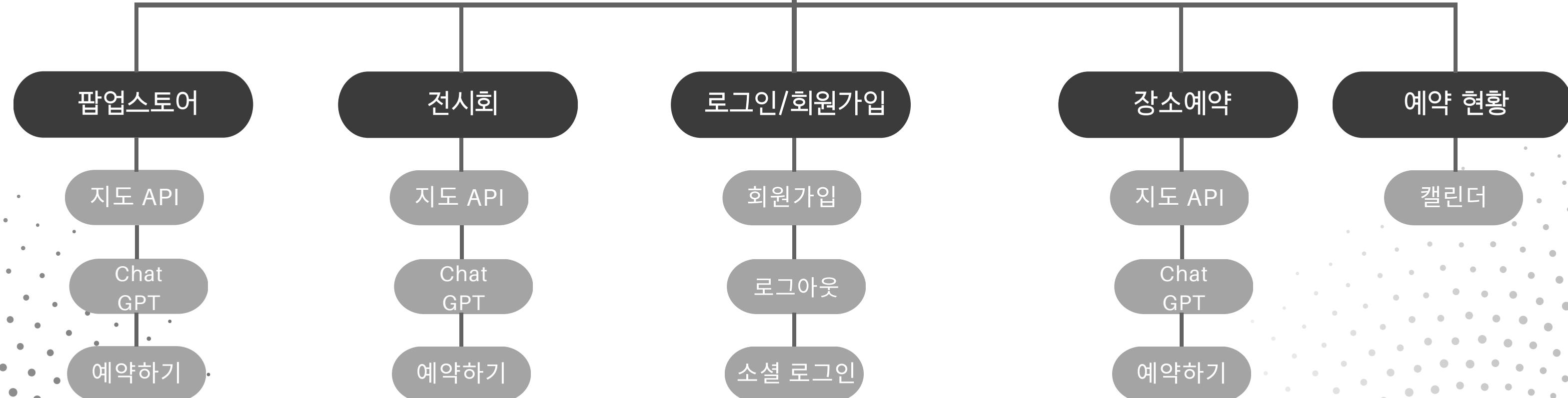
03 설계

메뉴 구조도

스토리보드

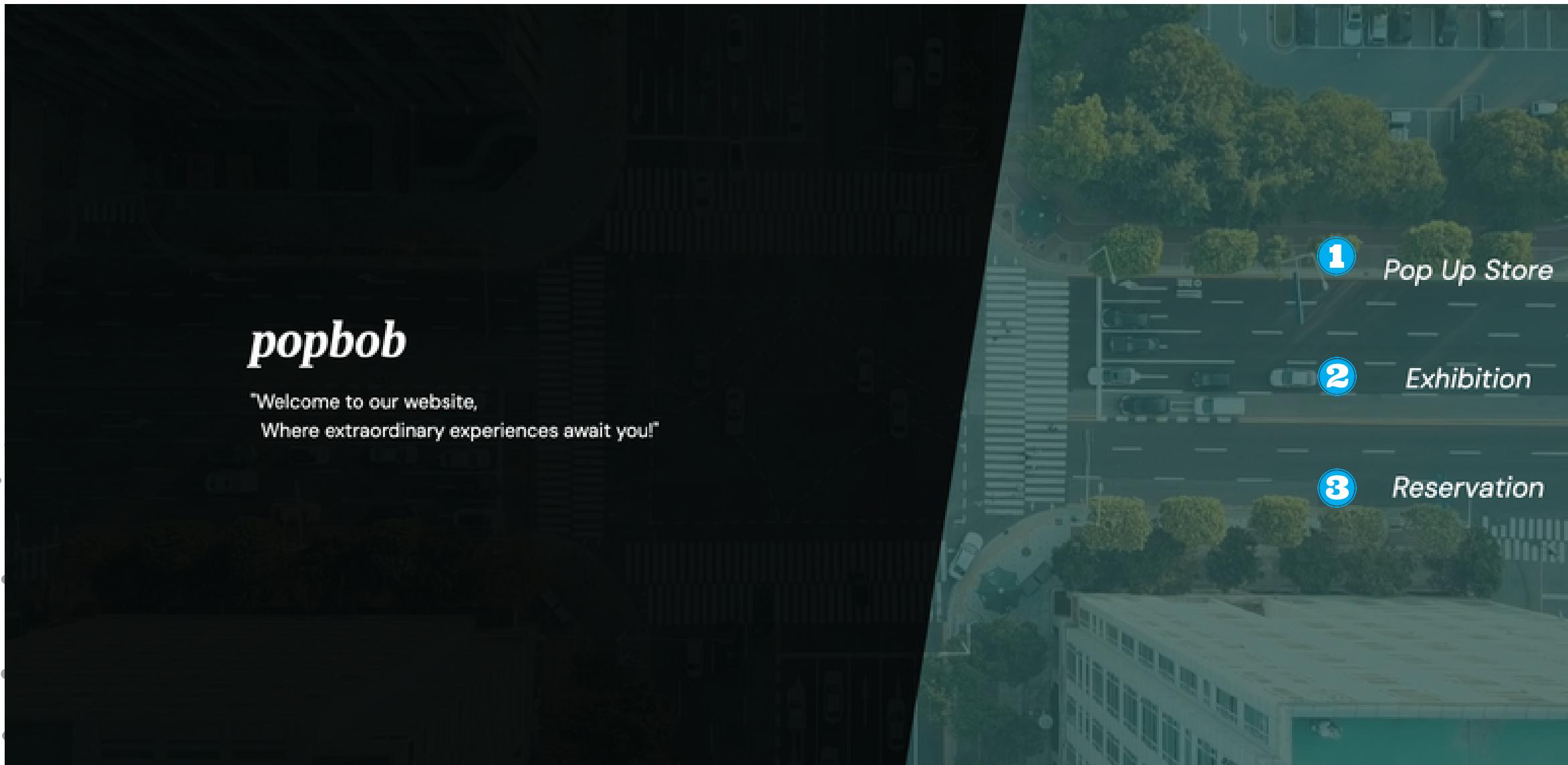
ERD

www.popbob.com



03 설계

메뉴 구조도



Main Page

- ❶ POP – UP Store List 페이지 이동
- ❷ 전시회 List 페이지 이동
- ❸ 장소 예약 및 예약 현황 확인 페이지 이동

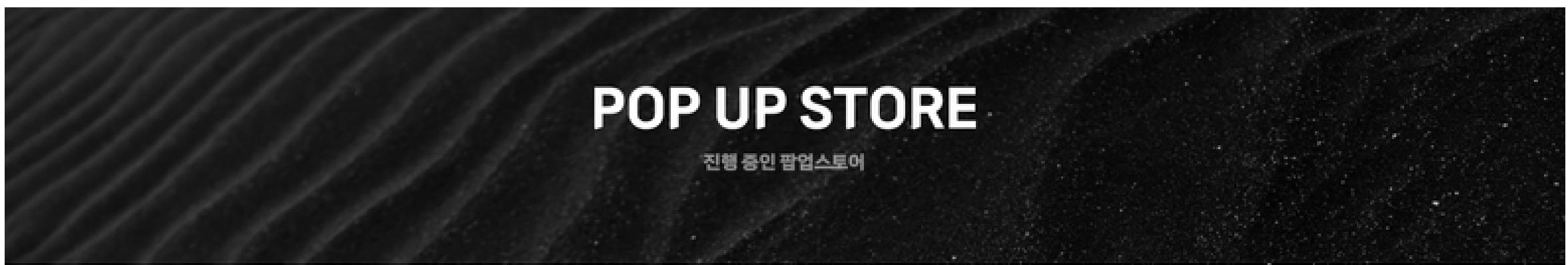


03 설계

스토리보드

POP UP LIST Page

① POP – UP Store View 페이지 이동



누데이크 성수점
상시
View

위글위글 하우스 도산
상시
View

금네 플레이타운
종료시 알림
View

한맥Xteddy 하우스
23.6.23(금) ~ 23.7.6(목)
View



03 설계

스토리보드



누데이크 성수점

가격 : 무료

성수핫플 누데이크 성수점
뉴진스 클라보로 유명해진 누데이크, 뉴진스 OMG 컴백을 기념해 팝
를 개시!
도산점도 인기가 많지만 성수점도 인기가 정말 많아요!

지하 1층 카페공간,
2, 3층 젠터 몬스터,
4층 탐버린즈로 구성되어 있습니다!

다양하고 특별한 디저트
신기하고 멋진 인테리어, 굽즈도 판매하고 있어요!

- 1 [상세지도](#) 2 [Chat](#)

POP UP View Page

- ① POP – UP Store 위치를
표시하는 카카오 맵 표시
- ② POP – UP Store에 관하여 간단한
질문을 할 수 있는 Chat GPT 기반 서비스



03 설계

스토리보드

POP UP View Page

- ① POP – UP Store 위치를 표시하는 카카오 맵 표시

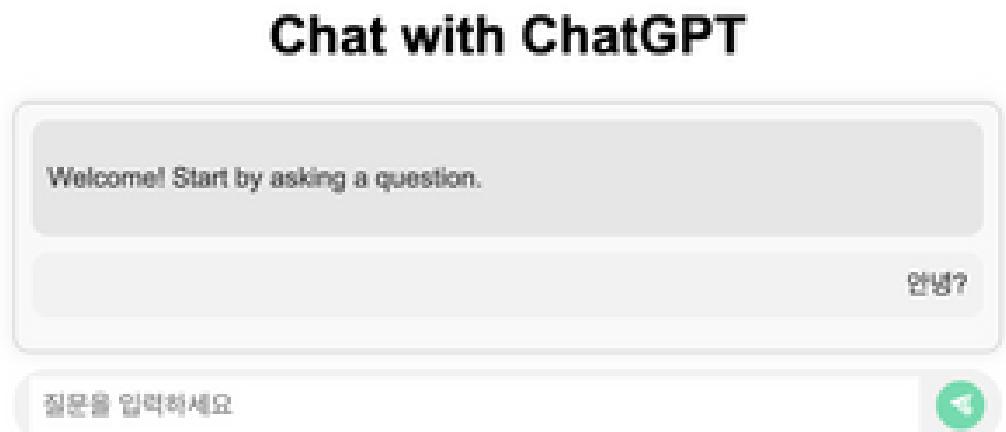


03 설계

스토리보드



03 설계



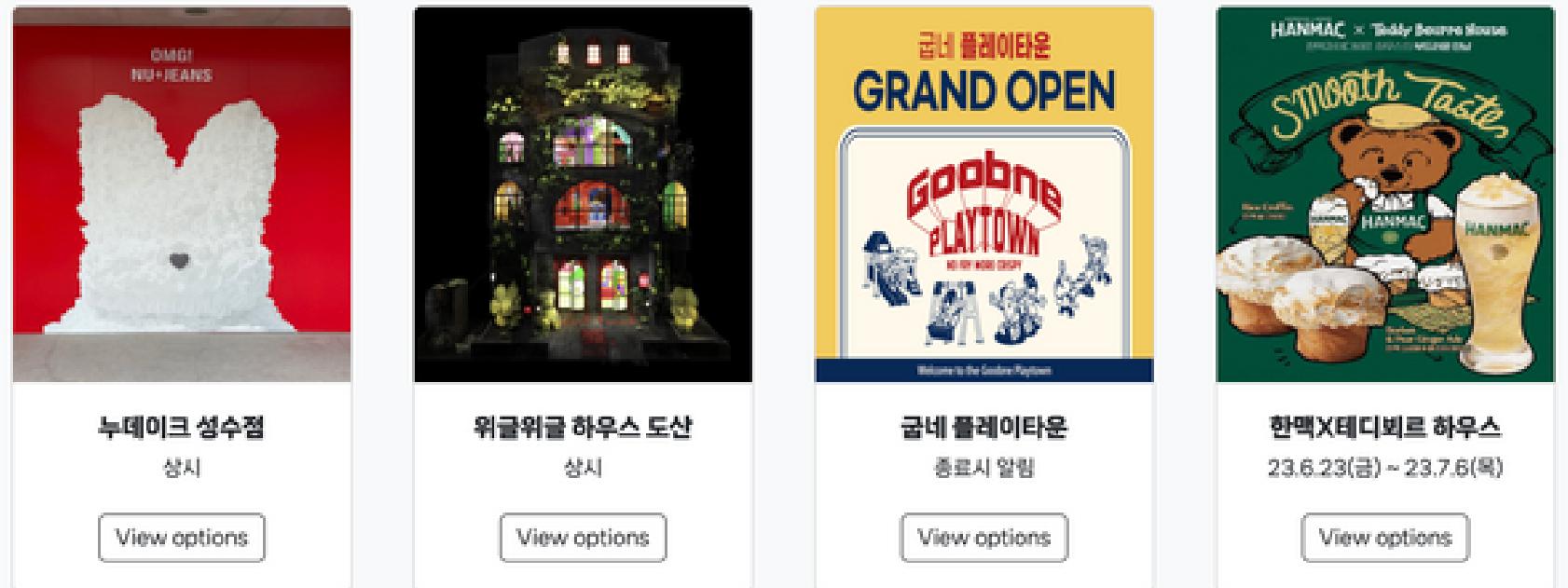
POP UP View Page

- ② POP – UP Store에 관하여 간단한 질문을 할 수 있는 Chat GPT 기반 서비스

스토리보드

POP UP View Page

① 팝업 스토어 최신 목록



① POP – UP Store 게시물을 최신순으로 표시

popbob © 문의 / 전화 02.538.3644

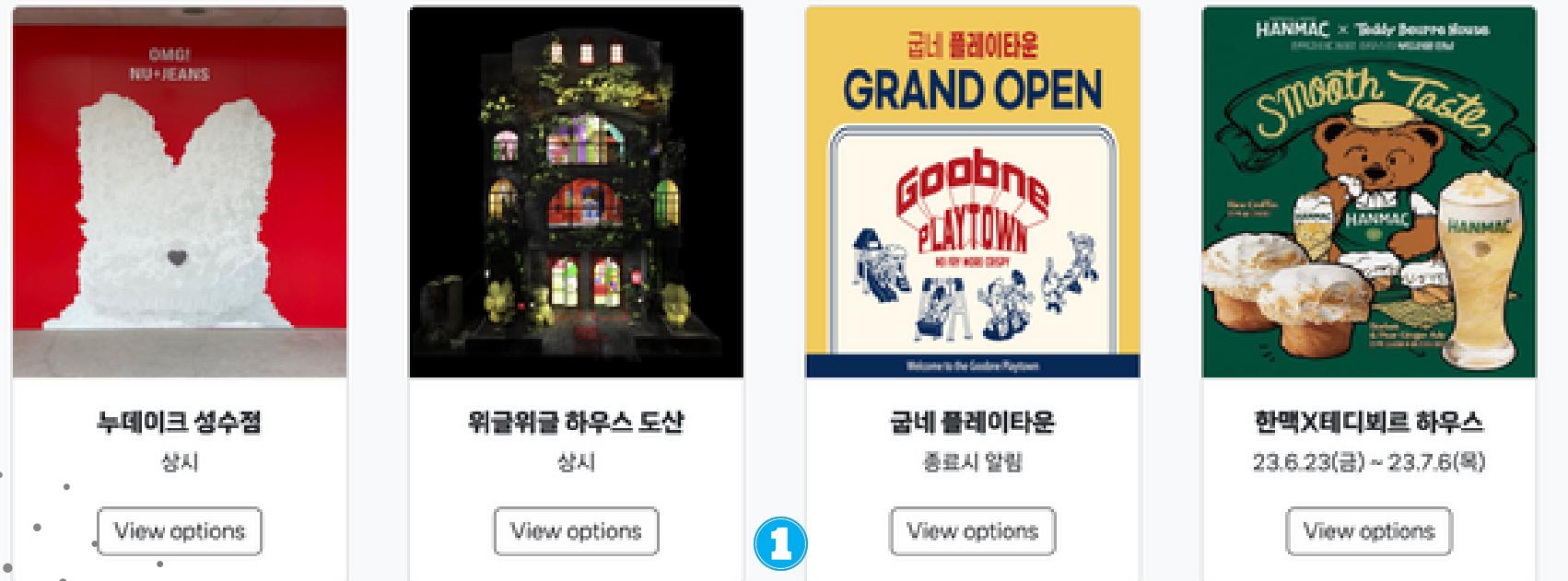


03 설계

스토리보드

전시회 View Page

① 팝업 스토어 최신 목록



popbob © 문의 / 전화 02.538.3644

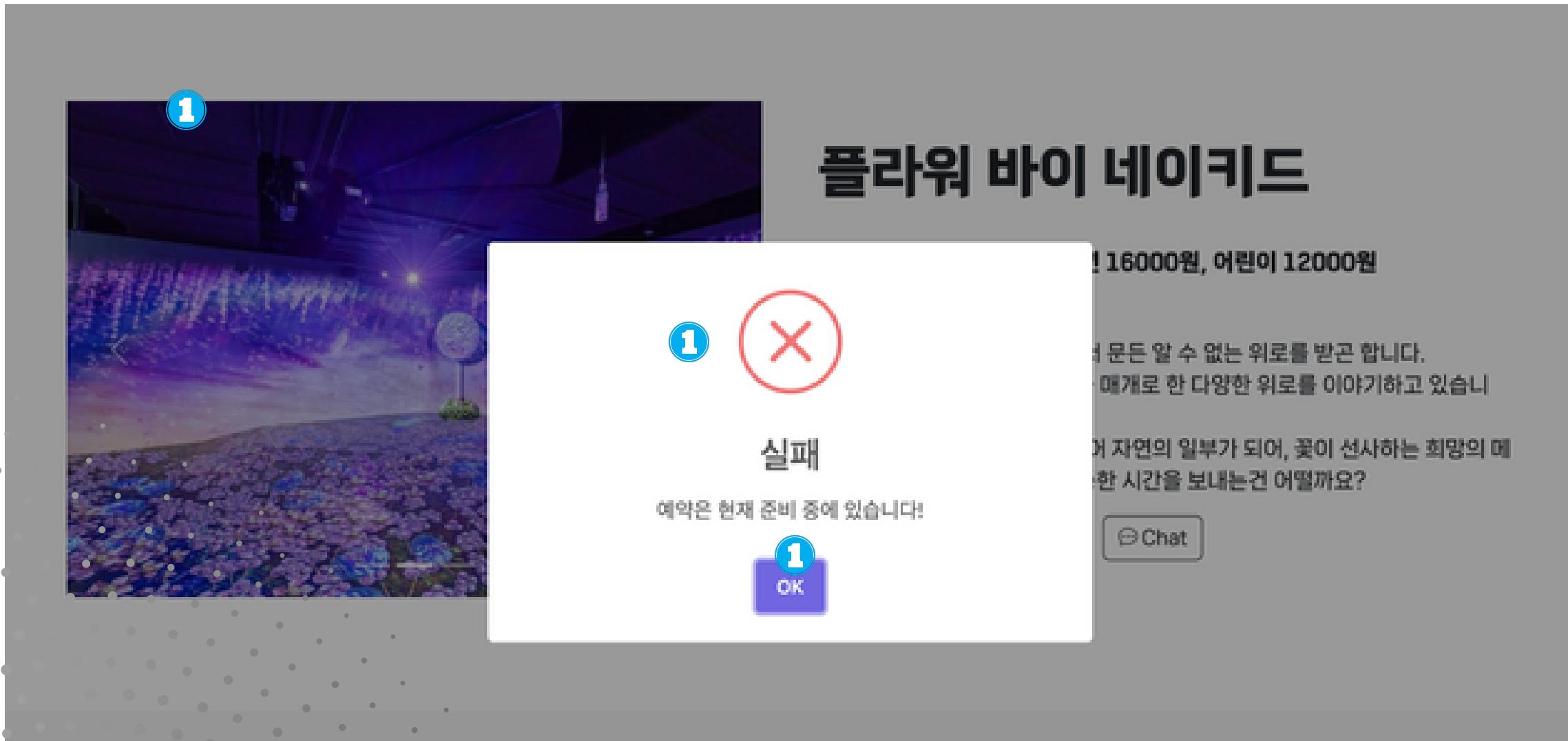
- ① 추후 전시업체와 협업하여 직접 예약을 할 수 있도록 기능 추가 예정



03 설계

스토리보드

전시회 View Page



- ① 추후 전시 업체와 협업하여 직접 예약을 할 수 있도록 기능 추가 예정



03 설계

스토리보드



popbob

아이디

비밀번호

1 LOGIN **2** 회원가입이 필요하신가요? [회원가입](#)

3 카카오 로그인

Login page

- ①** 로그인 : 아이디 / 비밀번호 입력 후, 성공 시 예약 페이지로 이동
- ②** 회원가입 : 회원가입 페이지로 이동
- ③** 카카오 로그인 : 카카오로 로그인

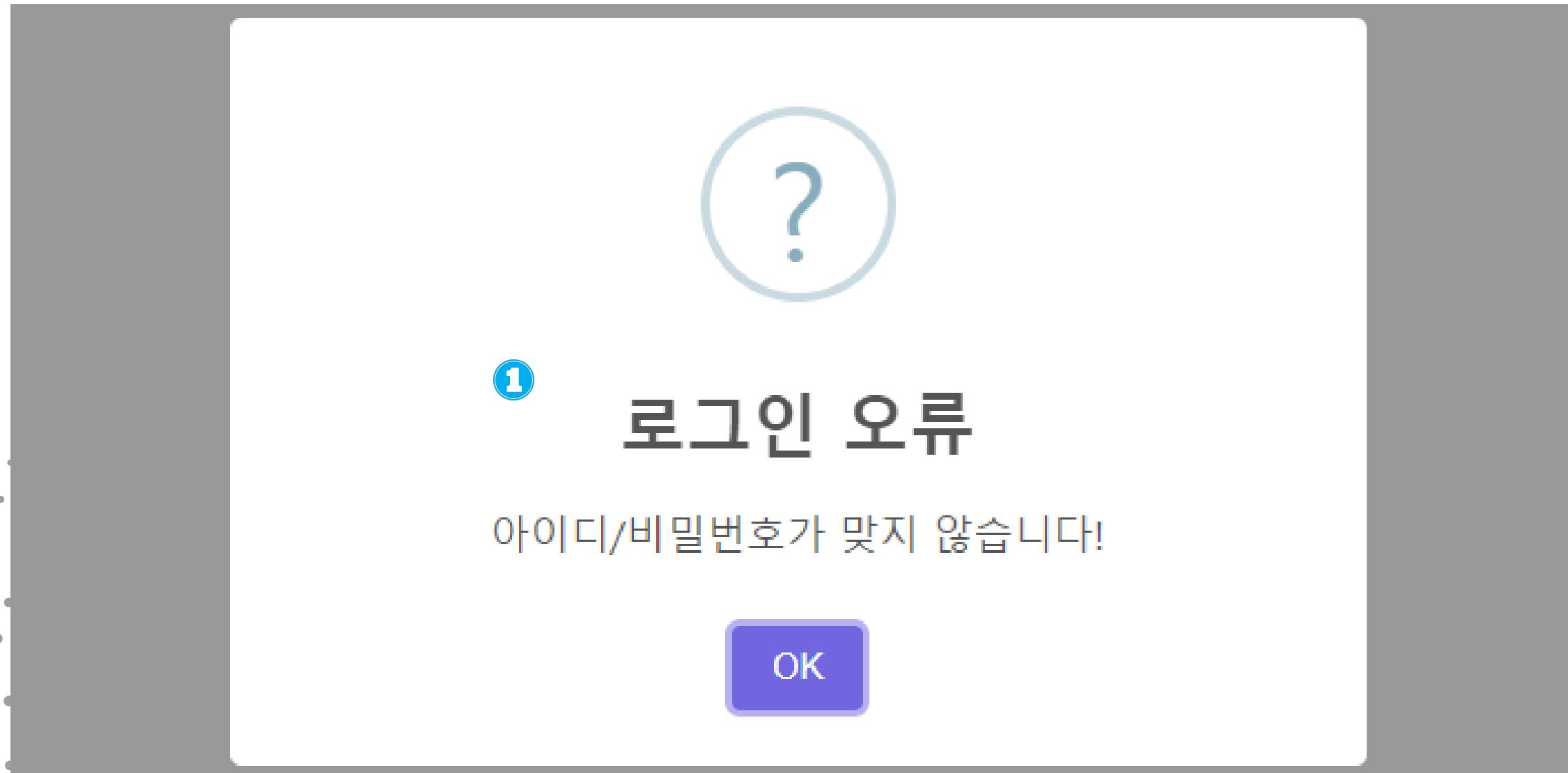


03 설계

스토리보드

Login page – 에러

- ① 정보를 잘못 입력하면 Alert 창 표시

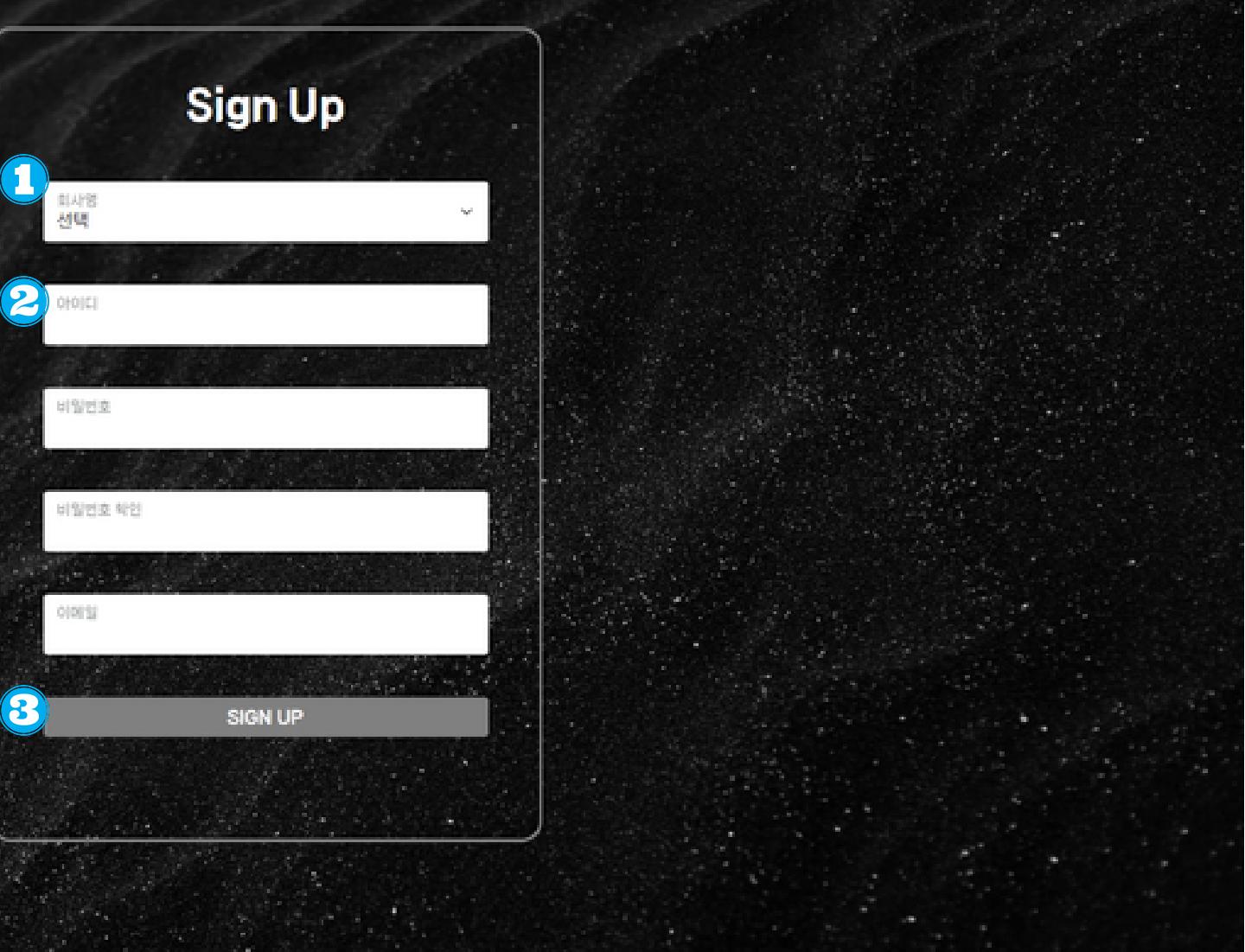


03 설계

스토리보드

회원가입

- ❶ 회사명 선택 : 옵션기능으로 select 선택
- ❷ 아이디 / 비밀번호 / 이메일 입력 : 정규식을 설정하여 그에 맞게 입력하도록 만듦
- ❸ 회원가입 버튼 : 위 조건 만족 시 회원가입 완료



The image shows a 'Sign Up' form interface. At the top, it says 'Sign Up'. Below that, there are four input fields with corresponding labels: 1. 회사명 선택 (Company Selection) with a dropdown arrow icon; 2. 아이디 (ID); 3. 비밀번호 (Password); and 4. 이메일 (Email). At the bottom is a large grey 'SIGN UP' button with white text.



03 설계

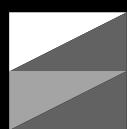
스토리보드

The screenshot shows a reservation interface. At the top left is a user icon labeled 'popbob' with a blue number '1'. To its right are navigation links: 'Home', 'Reservation', and 'OverView'. At the top right is another user icon with a blue number '2', followed by the text '환영합니다 cpname님!', a 'Logout' button, and a small profile picture. The main title 'Reservation' is centered above a sub-section titled '현재 예약 가능한 장소'. Below this is a card for a room listing:

- 청담 앤틱 테마 행사장**
-
- 가격 : 1,650,000**
- 위치 : 서울시 강남구 도산대로55길 26**
- 이용 가능한 기간 : 2023-08-03~2023-11-13**
- 예약하기** (button with a blue number '3')

예약 리스트 페이지

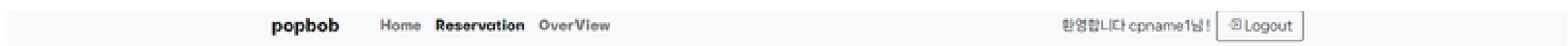
- ❶ 상단 메뉴바 : 각 메뉴 클릭시 해당 메뉴로 이동
- ❷ 사용자 문구 + 로그아웃 : 로그인 시, 사용자 아이디를 읽어서 표시 / 로그아웃 버튼 누를 시, 로그아웃
- ❸ 예약하기 : 버튼 클릭 시, 예약 뷰 페이지로 이동



03 설계

스토리보드

예약 뷰 페이지



청담 앤틱 테마 행사장

가격 : 1,650,000



① 장소
청담 앤틱 테마 행사장

회사명
(주)회사1

이메일
cpname1@naver.com

② 전화번호
(-미포함) 전화번호를 입력해주세요.

③ 시작 날짜
년-월-일

④ 예약하기

⑤ 마지막 날짜
년-월-일

⑥ 상세지도

Chat

- ❶ 장소 / 회사명 / 이메일 : 로그인한 사용자의 정보를 가져옴
- ❷ 전화번호 입력 : 정규식에 맞게 입력
- ❸ 시작 날짜 / 마지막 날짜 : 날짜 선택, 시작 날짜를 선택하면 마지막 날짜는 그 이전 날짜를 고를 수 없게 함
- ❹ 예약하기 버튼 : 조건 만족 시 예약 완료
- ❺ 상세지도 : 누르면 해당 장소가 모달 형식의 지도로 표시
- ❻ Chat : 질문을 받아주는 chat AI로 이동

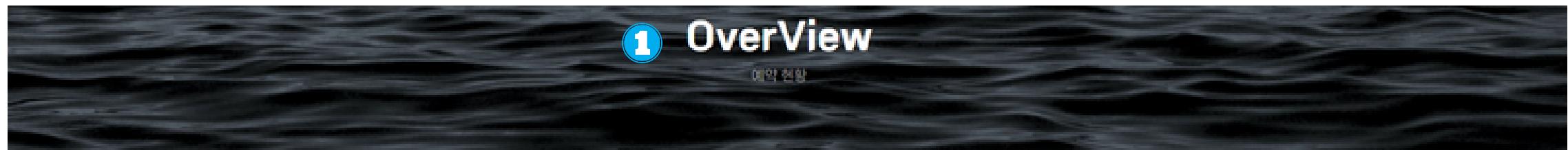


03 설계

스토리보드

① OverView

예약 현황



OverView 페이지

- ① 전체 예약 현황을 볼 수 있음
- ② 월 / 주 / 일 별로 표시 : 각 일정을 월 / 주 / 일에 맞게 보여준다.
- ③ 예약 완료된 일정들이 캘린더에 표시됨

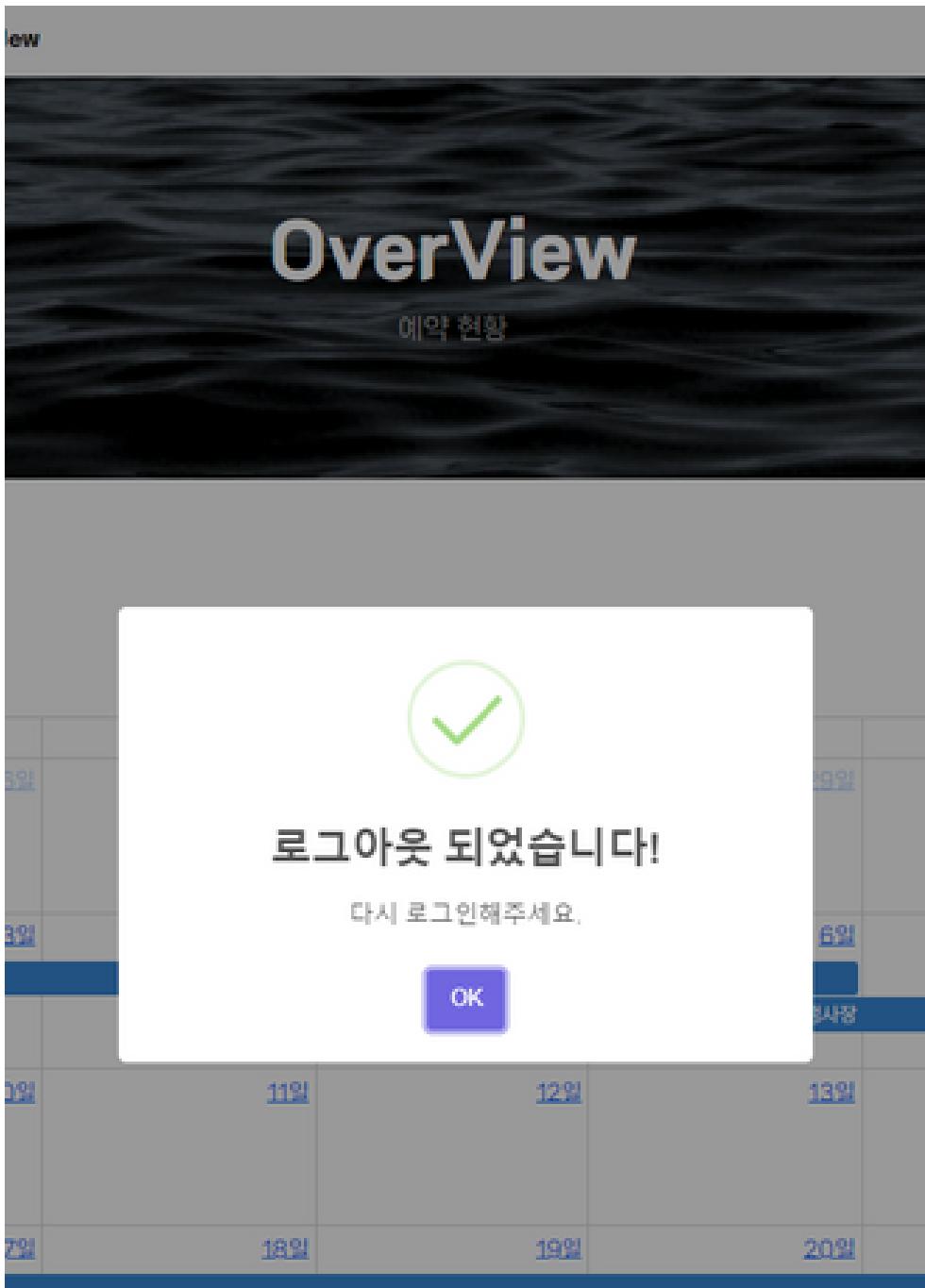
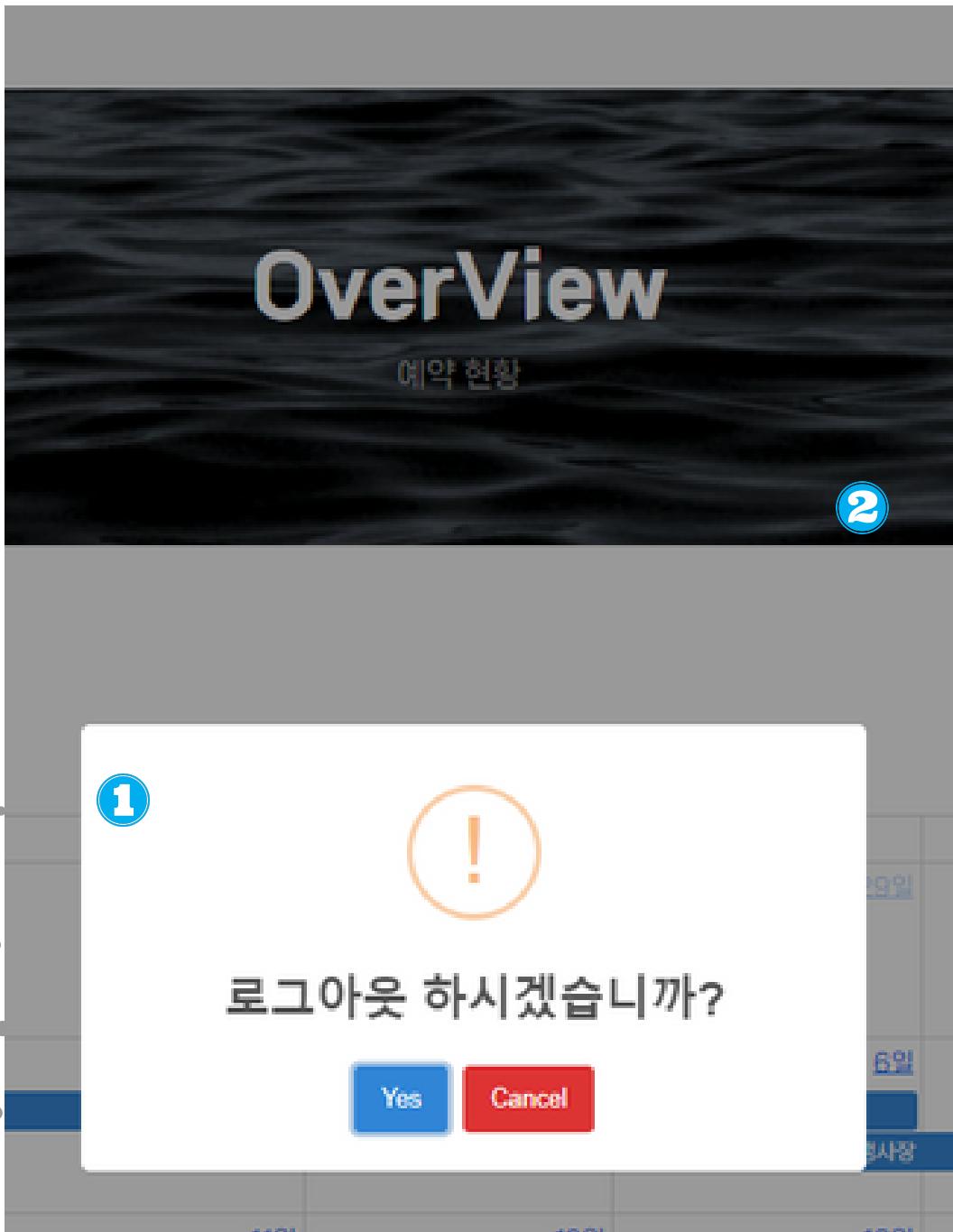


03 설계

스토리보드

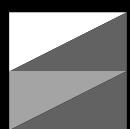
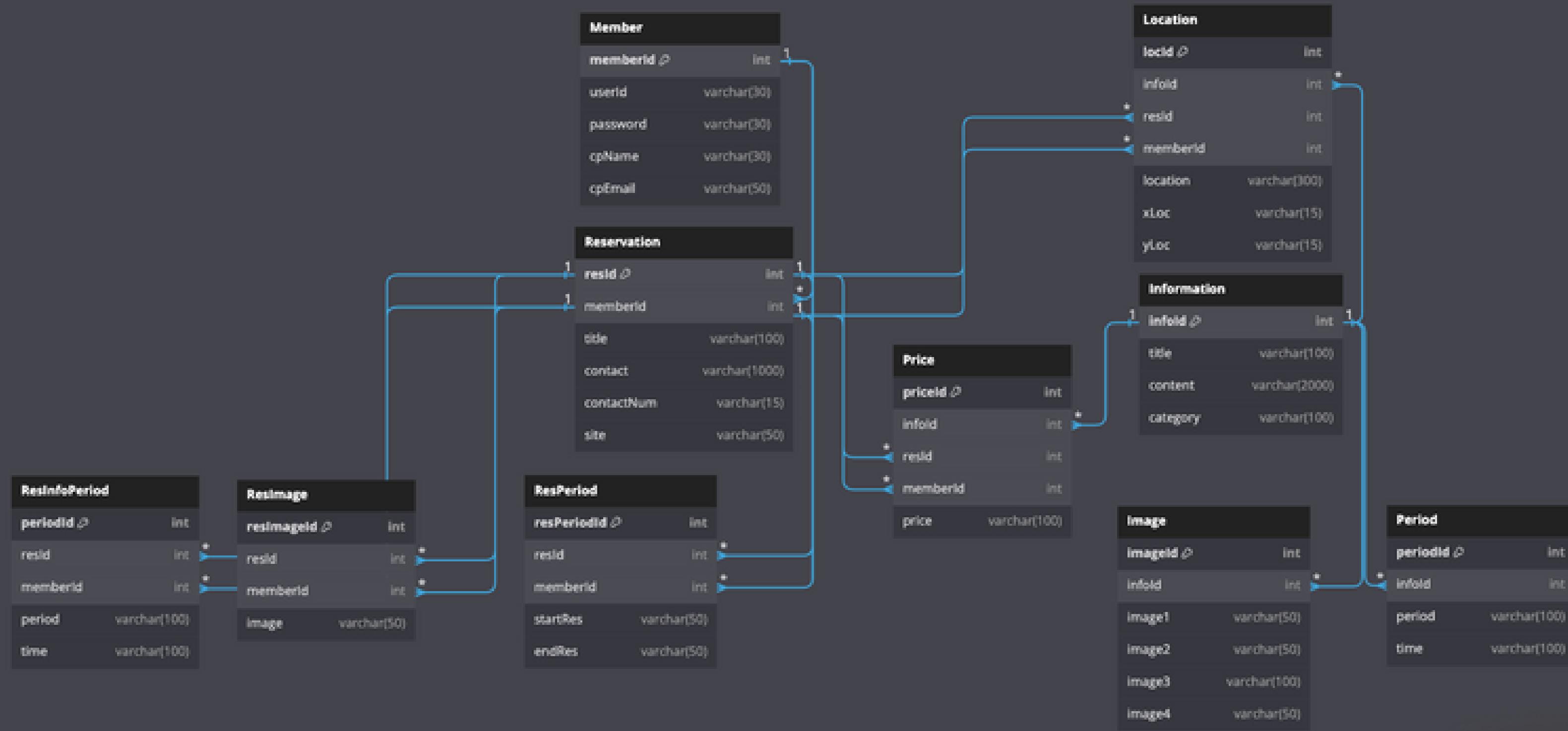
로그아웃

- ❶ 상단 로그아웃 버튼 누를 시에 확인창 표시
- ❷ 확인 누르면 로그아웃, 취소 누르면 변화 없음



03 설계

스토리보드

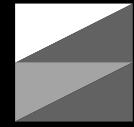


03 설계

ERD

04 화면 구현

구현 가능



구현 기능

사용자 / 메인	예약	팝업 / 전시회			
일반 로그인 (Session)	✓	회원가입 / 로그인 (일반 / 소셜)	△	팝업 / 전시회 List Page	✓
소셜 로그인 (카카오 API)	△	임대 장소 List Page	✓	팝업 / 전시회 View Page	✓
회원가입	✓	임대 장소 View Page	✓	지도 API 구현	✓
메인 - 팝업 페이지 이동	✓	장소 임대 예약	△	Chat GPT API 구현	✓
메인 - 전시회 페이지 이동	✓	예약현황 확인 (Full Calendar)	✓	전시회 예약 (사이트 이동)	△
메인 - 예약 페이지 이동	✓	지도 API 구현	✓		
		Chat GPT API 구현	✓		

```
@RequestMapping("/signup.do")
public ModelAndView signup() {
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.setViewName("signup");
    return modelAndView;
}

@RequestMapping("/signup_ok.do")
public ModelAndView signup_ok(HttpServletRequest request) {
    MemberTO signupTO = new MemberTO();

    signupTO.setUserId(request.getParameter("UserId"));
    signupTO.setPassword(request.getParameter("Password"));
    signupTO.setCpName(request.getParameter("CpName"));
    signupTO.setCpEmail(request.getParameter("CpEmail"));

    int flag = dao.signupOk(signupTO); ← Int flag에 DAO에서 처리된 값을 반환 받아 회원가입 완료 상태를 나타내는 1 또는 0을 반환

    ModelAndView modelAndView = new ModelAndView();

    modelAndView.setViewName("signup_ok");
    modelAndView.addObject("flag", flag); ← 반환된 flag값을 modelAndView.addObject("flag", flag)로 보내줌 -> signup_ok.jsp로
    return modelAndView;
}
```

회원가입 페이지를 보여준다.

회원가입시 입력된 값을 받아온다.

Int flag에 DAO에서 처리된 값을 반환 받아 회원가입 완료 상태를 나타내는 1 또는 0을 반환

반환된 flag값을 modelAndView.addObject("flag", flag)로 보내줌 -> signup_ok.jsp로



```

//정규 표현식 적용
const idRegExp = /^[a-zA-Z0-9]{6,12}$/; // 문자와 숫자로만 구성, 길이는 6 ~ 12 사이

// 영문 숫자 특수기호 포함 8자리 이상
const passRegExp = /^(?=.*[a-zA-Z])(?=.*[!@#$%^&*-])(?=.*[0-9]).{8,15}$/;

//이메일 정규식 / 이메일 주소는 영문 대소문자, 숫자, 특수문자(. _ %+-)로 구성, "@" 포함, "." 기호로 구분된 최소 2자 이상의 영문 대소문자로 된 도메인
const emailRegExp = /^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}\$/;

```

직접 입력 해야하는 정보들은 이상한 값의
입력을 막기위해 정규식을 사용

```

<!-- 비밀번호 유통 확인 및 한글문자 -->
<script>
const koreanRegExp = /[ㄱ-ㅎ]ㅏ-ㅣㅏ-ㅣ]/; // 아이디에 한글 만들었으면 하기위한 한글 정규식

$(document).ready(function() {
    // 아이디 비밀번호 유통 및 한글문자
    $('#floatingId').keyup(function() {
        let id = $(this).val();

        if (id != '') {
            if (koreanRegExp.test(id)) {
                $('#id_result').text('영문/숫자로 접속해주세요').css('color', 'red');
            } else if (idRegExp.test(id)) {
                $('#id_result').text('영문자 / 숫자로 6 ~ 12자 사이로 접속해주세요').css('color', 'red');
            } else {
                $.ajax({
                    type: 'get',
                    url: '/id_check',
                    async: true,
                    data: { id: id },
                    success: function(result) {
                        if (result == 0) {
                            $('#id_result').text('사용 가능합니다').css('color', 'blue');
                        } else if (result == 1) {
                            $('#id_result').text('중복된 아이디').css('color', 'red');
                        }
                    },
                    error: function(e) {
                        alert('[에러] : ' + e.status);
                    }
                });
            }
        } else {
            $('#id_result').text('아이디를 접속해주세요 !').css('color', 'blue');
        }
    });
});

```

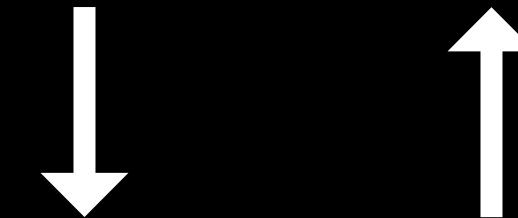
Ajax를 이용하여 실시간으로 중복체크를 적용
다른 입력정보들도 동일.



04 구현 기능

회원가입Ⅱ

```
// 회원가입  
public int signupOk(MemberTO to) {  
  
    int flag = 1;  
  
    BCryptPasswordEncoder encoder = new BCryptPasswordEncoder(); ← 회원가입시 입력한 정보를 넘겨받아서 패스워드를 Bcrypt를 사용하여 암호화  
    String encodePassword = encoder.encode(to.getPassword());  
    to.setPassword(encodePassword);  
  
    int result = mapper.signupOK(to); ← 그 후에 담은 정보들을 쿼리문에 넣어주기 위해 mapper.signupOK(to)로 보내줌 후 result 결과값 담음  
    if (result == 1) {  
        flag = 0;  
    }  
  
    return flag; ← 회원가입의 성공 / 실패를 flag 0과 1로 구분하여 다시 Controller로 반환  
}
```



```
// 회원가입시 사용하는 insert문  
@Insert("insert into member values(0, #{UserId}, #{Password}, #{CpName}, #{CpEmail})") ← DAO에서 넘겨받은 값을 쿼리문에 적용  
public int signupOK(MemberTO to);
```



04 구현 기능

회원가입Ⅲ

```
// 로그인
public boolean login(MemberTO to) {
    MemberTO storedTO = mapper.login(to);

    if (storedTO != null) {
        BCryptPasswordEncoder encoder = new BCryptPasswordEncoder();
        String storedPasswordHash = storedTO.getPassword(); // 데이터베이스에 저장된 hash 암호
        String enteredPassword = to.getPassword(); // 로그인 시 입력한 암호

        // match 메서드로 두 개의 값이 같은지 확인 가능.
        if (encoder.matches(enteredPassword, storedPasswordHash)) { ←
            // 패스워드 검증 성공
            return true;
        }
    }
    return false;
}
```

로그인 입력되는 아이디 비밀번호 값을 담고,
암호화 되어있는 패스워드 값과 입력한 패스워드
값이 일치하는지 확인



04 구현 기능

로그인

```

// POST방식으로 key - value 형태의 데이터를 요청해와온(카카오로부터)
// RestTemplate : POST방식으로 요청할때 필요한 라이브러리( http 요청을 관리해줫을 때요, HttpURLConnection같은 것(보다 좋음) )
RestTemplate rt = new RestTemplate();

HttpHeaders headers = new HttpHeaders();
headers.add("Content-type", "application/x-www-form-urlencoded;charset=utf-8"); // body 데이터가 key - value 형태라고 알려주는 것

// body데이터를 맵을 오피젝트로
MultiValueMap<String, String> params = new LinkedMultiValueMap<>();
params.add("grant_type", "authorization_code");
params.add("client_id", " ");
params.add("redirect_uri", "http://localhost:8080/kakao/callback");
params.add("code", code); // code값은 블로그로 받기때문에 code를 넣어줌

// HttpEntity<(params, headers) 이렇게 데잍으로서 params와 headers를 갖을 가지고있는 Entity가 된다.
HttpEntity<MultiValueMap<String, String>> kakaoTokenRequest = new HttpEntity<(params, headers); // HttpEntity : HTTP 메시지를 관리해 주는걸 수 있게 하든다

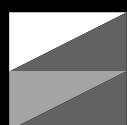
// HTTP 요청 - POST방식으로 - 그리고 response 변수를 통해 받음.
ResponseEntity<String> response = rt.exchange( // exchange라는 변수가 ResponseEntity를 오피젝트로 넘겨 보내었음. 그래서 사용
    "https://kauth.kakao.com/oauth/token", HttpMethod.POST, kakaoTokenRequest, String.class);

// Gson, Json Simple, ObjectMapper
ObjectMapper objectMapper = new ObjectMapper();
OAuthToken oauthToken = null;
try {
    oauthToken = objectMapper.readValue(response.getBody(), OAuthToken.class);
} catch (JsonMappingException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (JsonProcessingException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

System.out.println("카카오 액세스 토큰 : " + oauthToken.getAccess_token());

```

카카오 서버에 요청하여
Access Token값을 가져온다.



04 구현 가능

Kakao 로그인

```
// 사용자 정보 얻기
RestTemplate rt2 = new RestTemplate();

HttpHeaders headers2 = new HttpHeaders();
headers2.add("Authorization", "Bearer " + oauthToken.getAccess_token());
headers2.add("Content-type", "application/x-www-form-urlencoded; charset=utf-8");

HttpEntity<MultiValueMap<String, String>> kakaoProfileRequest2 = new HttpEntity<>(headers2);

// Http 요청 - POST방식으로 - 그대로 response 변수로 응답 받음.
ResponseEntity<String> response2 = rt2.exchange( // exchange()는 맵수가 ResponseEntity<>를 오브젝트를 넘겨 보내었음. 그래서 사용
    "https://kapi.kakao.com/v2/user/me", HttpMethod.POST, kakaoProfileRequest2, String.class);

ObjectMapper objectMapper2 = new ObjectMapper();
KakaoProfile kakaoProfile = null;
try {
    kakaoProfile = objectMapper2.readValue(response2.getBody(), KakaoProfile.class);
} catch (JsonMappingException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (JsonProcessingException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

System.out.println("카카오 아이디(번호) : " + kakaoProfile.getId());
System.out.println("카카오 이메일 : " + kakaoProfile.getKakao_account().getEmail());
System.out.println("카카오 닉네임 : " + kakaoProfile.getProperties().getNickname());
```

받은 토큰 값을 이용하여 서버
에서 로그인한 나의 정보를 가
져온다.

```
if (kakaoId != null || kakaoEmail != null ) {
    HttpSession session = request.getSession();
    session.setAttribute("kakaoId", kakaoId);
    session.setAttribute("kakaoEmail", kakaoEmail);
    session.setAttribute("kakaoNickname", kakaoNickname);
    session.setAttribute("kakaoAccessToken", oauthToken.getAccess_token());
    session.setAttribute("loginSession", "true");

    session.setMaxInactiveInterval(60 * 60);

    return "redirect:/resList.do";
}

return "redirect:/login.do";
```

받아온 정보를 다른 곳에서도
사용하기위해 세션으로 넘겨
주었음



04 구현 가능

Kakao 로그인

```
71
72 <!-- 카카오 API 코드 -->
73<script type="text/javascript"
74     src="//dapi.kakao.com/v2/maps/sdk.js?appkey=
75                                     ></script>
```

카카오 API 사용하기 위함, appkey를 받아서 입력해주면 된다.

```
// 모달이 열릴 때 지도 생성
$('#myModal').on('shown.bs.modal', function() {
    // 지도가 표시될 HTML element
    let container = document.getElementById('map');
    let options = {
        // 중심 좌표 설정
        center : new kakao.maps.LatLng(<%=xLoc%, <%=yLoc%>),
        level : 3
    // 지도의 확대 수준을 설정함 (0~14)
    };

    let map = new kakao.maps.Map(container, options); ← 지도를 표시하기 위한 옵션들
                                                ← 위 / 경도와 확대 수준을 설정 후 지도를 표
                                                ← 시한다.

    // 지도에 표시될 마크 위치 설정
    let markerPosition = new kakao.maps.LatLng(<%=xLoc%, <%=yLoc%>);
    let marker = new kakao.maps.Marker({
        position : markerPosition
    });

    marker.setMap(map); ← 지도 위의 마크를 표시
                        ← 마찬가지로 위 / 경도 설정 후 해당 위치에
                        ← 마크 표시

}); ← 닫기 버튼 누를 시 지도 모달 종료

// 모달 닫힐 때 지도 제거
$('#myModal').on('hidden.bs.modal', function() {
    let container = document.getElementById('map');
    container.innerHTML = '';
});
```



04 구현 기능

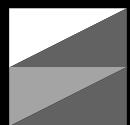
지도 API

```
<!-- fullcalendar CDN -->
<link href='https://cdn.jsdelivr.net/npm/fullcalendar@5.8.0/main.min.css' rel='stylesheet' />
<script src='https://cdn.jsdelivr.net/npm/fullcalendar@5.8.0/main.min.js'></script>
<!-- fullcalendar 번역 CDN -->
<script src='https://cdn.jsdelivr.net/npm/fullcalendar@5.8.0/locales-all.min.js'></script>
```

캘린더는 fullcalendar 사용

```
// 이벤트
events: [
  <%=sbHtml%>
]
});
// 캘린더 렌더링
calendar.render();
});
})();
</script>
</div>
</section>
```

캘린더를 형성하는 부분에서 events
에 DB에 저장된 정보를 연동시켜 예
약된 일정들을 표시



04 구현 기능

캘린더 API

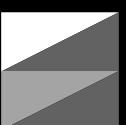
```
ArrayList<CheckTO> cal = (ArrayList<CheckTO>) request.getAttribute("cal"); ← DB에 저장된 정보 가져와서 변수  
String title = "";  
String cpName = "";  
String startDate = "";  
String endDate = "";  
  
StringBuilder sbHtml = new StringBuilder();  
  
for(CheckTO to : cal) {  
    title = to.getTitle();  
    cpName = to.getcpName();  
    startDate = to.getStartDate();  
    endDate = to.getEndDate();  
  
    // endDate에 하루를 더해주는 로직 추가  
    try {  
        SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");  
        Date parsedStartDate = dateFormat.parse(startDate);  
        Date parsedEndDate = dateFormat.parse(endDate);  
        Calendar calendar = Calendar.getInstance();  
        calendar.setTime(parsedEndDate);  
        calendar.add(Calendar.DAY_OF_MONTH, 1);  
        endDate = dateFormat.format(calendar.getTime());  
    } catch (ParseException e) {  
        e.printStackTrace();  
    }  
  
    sbHtml.append("{");  
    sbHtml.append("title: '" + cpName + "' : '" + title + "',");  
    sbHtml.append("start: '" + startDate + "',");  
    sbHtml.append("end: '" + endDate + "'");  
    sbHtml.append(",");  
}
```

DB에 저장된 정보 가져와서 변수에 집어 넣기

fullCalender에서 마지막 날이 잘리는 오류가 있어 마지막 날을 추가해주는 코드 작성

형식 정하고("yyyy-MM-dd"), 각 변수에 적용 후 Add로 1일을 추가하여 endDate에 적용

반복문으로 인하여 DB에 있는 정보를 형식에 맞게 전부 가져옴



04 구현 기능

캘린더 API

```
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

import com.example.config.ChatGptConfig;
import com.example.dto.ChatGptRequestDto;
import com.example.dto.ChatGptResponseDto;
import com.example.dto.QuestionRequestDto;

@Service
public class ChatGptService {

    private static RestTemplate restTemplate = new RestTemplate();

    public HttpEntity<ChatGptRequestDto> buildHttpEntity(ChatGptRequestDto requestDto) {
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.parseMediaType(ChatGptConfig.MEDIA_TYPE));
        headers.add(ChatGptConfig.AUTHORIZATION, ChatGptConfig.BEARER + ChatGptConfig.API_KEY);
        return new HttpEntity<>(requestDto, headers);
    }

    public ChatGptResponseDto getResponse(HttpEntity<ChatGptRequestDto> chatGptRequestDtoHttpEntity) {
        ResponseEntity<ChatGptResponseDto> responseEntity = restTemplate.postForEntity(
            ChatGptConfig.URL,
            chatGptRequestDtoHttpEntity,
            ChatGptResponseDto.class);
        return responseEntity.getBody();
    }

    public ChatGptResponseDto askQuestion(QuestionRequestDto requestDto) {
        return this.getResponse(
            this.buildHttpEntity(
                new ChatGptRequestDto(
                    ChatGptConfig.MODEL,
                    requestDto.getQuestion(),
                    ChatGptConfig.MAX_TOKEN,
                    ChatGptConfig.TEMPERATURE,
                    ChatGptConfig.TOP_P
                )
            )
        );
    }
}
```

Chat Gpt 서버로 질문을 보내기 전에
서버와 통신 및 인증 절차를 Header에 담아 전송

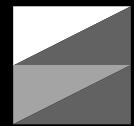


04 구현 가능

Chat GPT API

```
public class ChatGptConfig {  
    public static final String AUTHORIZATION = "Authorization";  
    public static final String BEARER = "Bearer ";  
    public static final String API_KEY = "YOUR_API_KEY";  
    public static final String MODEL = "text-davinci-003";  
    public static final Integer MAX_TOKEN = 300;  
    public static final Double TEMPERATURE = 0.0;  
    public static final Double TOP_P = 1.0;  
    public static final String MEDIA_TYPE = "application/json; charset=UTF-8";  
    public static final String URL = "https://api.openai.com/v1/completions";  
}
```

Chat Gpt 서버와 통신 및 인증 절차를 위한
토큰 및 API KEY를 Config를 통해 관리하여
다른 클래스에서도 사용 가능하게 관리.



04 구현 기능

Chat GPT API



POPBOB

05 소감

소감

박*람

우선 서로를 믿고 도와주며 함께 이끌어준 팀원분들 덕에 무사히 최종 프로젝트를 마치게 되어 감사드립니다.

이번 프로젝트를 진행하면서 여러 새로운 기술과 도구들을 익힌 것뿐만 아니라 팀원들과의 팀워크를 통해 많은 것을 배우고 성장할 수 있었습니다.

다들 너무 감사했습니다!
고생하셨어요 :) 🍀

김*연

6개월이 생각보다 빨리 간 것 같은데 다들 취뽀 성공하시길 기원합니다!!

이*준



제 자신의 실력이 많이 부족하다는 것을 뼈저리게 느낄 수 있는 프로젝트 였습니다.

하지만 덕분에 더 열심히해서 성공해보자는 욕심이 생기게 해주는 좋은 시간이였습니다.

이제 건들지 말아주세요 ..

이*영

아,,처음에는 어떻게 한 달 안에 프로젝트를 완성해야 하나 막막했었는데, 팀원들과 서로 잘 소통하면서 한 덕분에 잘 마무리할 수 있었던 것 같습니다.. 생각한 것 보다 재밌는 프로젝트였고 앞으로 개량 해나가며 지식을 쌓아가겠습니다



조*현

걱정이 많았던 프로젝트였지만 팀원들과의 커뮤니케이션을 통해 문제점을 해결하고 미흡한 부분을 보완할 수 있었습니다.

또한 프로젝트를 통해 개념을 다시 공부하는 계기가 되었고 저에게 부족한 점을 발견할 수 있어 좋은 경험이 되었습니다.



POPBOB

06 시연

Thank You