



Home

About

Contact



팝업스토어가 한눈에

# POPIC

POPIC은 분산된 팝업 정보를 통합하고, 예약·대기 기능으로 현장 대기 시간을 최소화합니다.

정지현

김아영

임유림

지원준



Learn More →



# 목차

- 01 프로젝트 개요
- 02 프로젝트 팀 구성 및 역할
- 03 프로젝트 수행 절차 및 방법
- 04 프로젝트 시연 및 기능소개
- 05 비즈니스 및 마케팅 전략
- 06 자체 평가 및 개인 의견
- 07 Q&A



Home

Overview

Contact



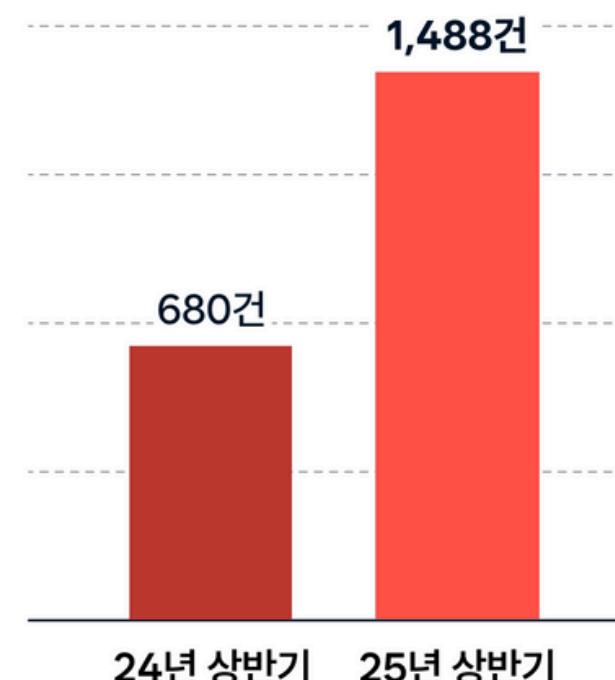
# 프로젝트 개요

Learn More A white button with rounded corners containing the text "Learn More" and a small white arrow pointing to the right.



## 2025년 팝업스토어

상반기 팝업스토어 오픈 수



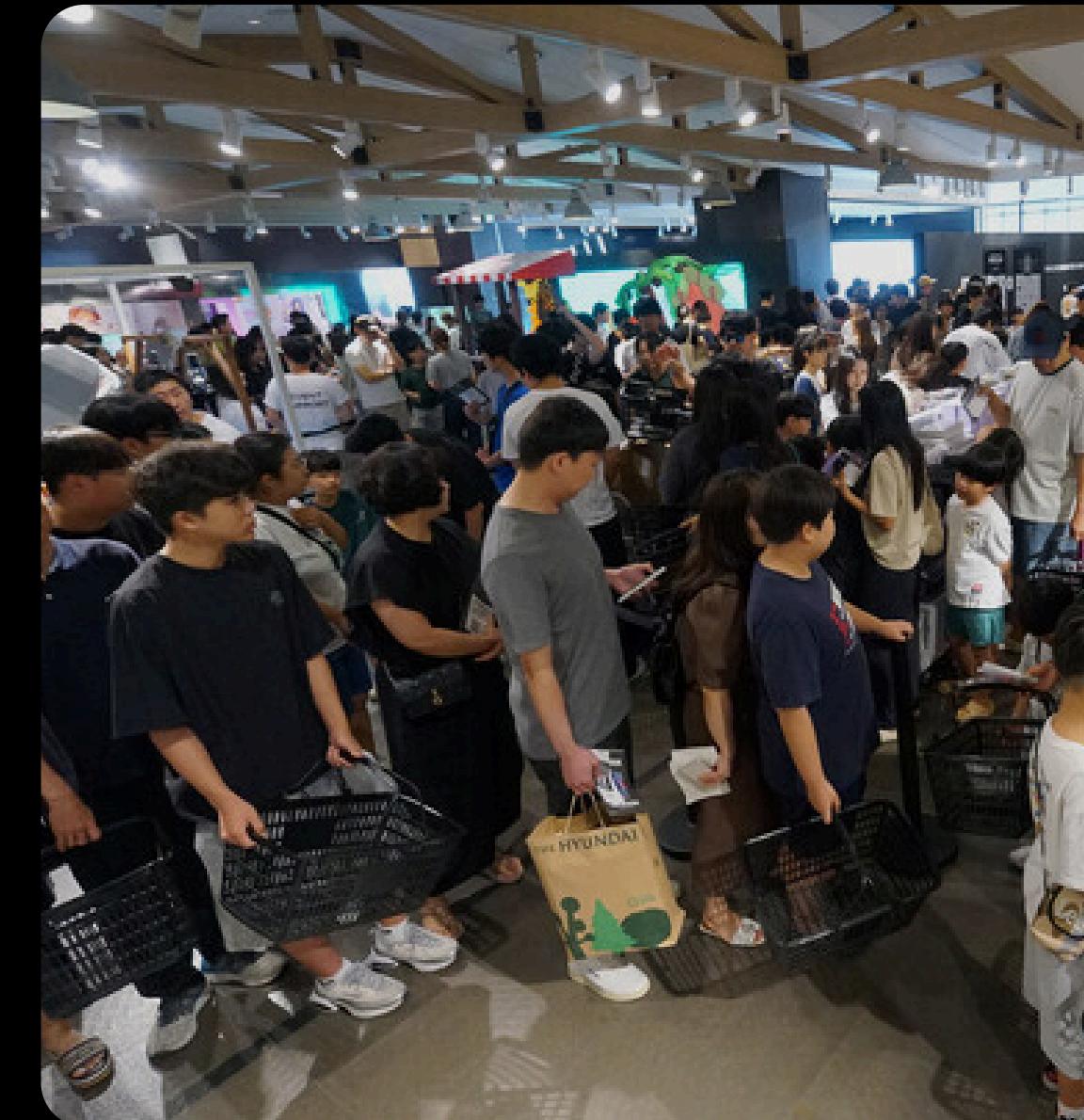
### 2025 상반기 팝업스토어 1,488개, 전년 대비 2배 ↑

유통·패션·IT 등 업종 전반이 체험형 마케팅을 확대하며, 도심 핵심 상권 중심으로 출점이 빠르게 늘고 있다. 다만 정보가 플랫폼·SNS에 파편화되어 이용자는 일정·위치·혼잡도·체험 내용 등을 한 곳에서 비교·예약하기 어렵다.

### 2시간 대기, 10분 체험... 현장 대기 불편 '폭증'

인기 팝업 현장에선 몇 시간을 기다리고도 체험은 수분에 그치거나, 새벽부터 줄을 섰지만 물량 소진으로 구매조차 못 하는 사례가 잇따른다. 실시간 대기 인원·예상 대기시간 안내가 부족하고, 수기 줄 관리·동시 입장 등으로 대기열 왜곡과 과밀이 빈발한다.

출처 : 조선일보, 이뉴스 투데이





### 팝업 정보의 파편화

운영 기간이 짧고 채널은 분산  
팝업 정보가 흩어져 최신성·신뢰성 저하

### 정보 통합 & 표준화

팝업 정보를 한 곳으로 통합  
실시간 업데이트로 최신성 및 신뢰성 확보

### 현장 대기의 비효율

인기 팝업 장시간 대기·선착순 품절·헛걸음  
혼잡·불만·안전 리스크 증가

### 줄 없는 방문 경험

사전 예약 및 대기 시스템으로  
대기 시간 최소화 및 불필요한 헛걸음 방지





## 팝업은 PoPIC

분산된 팝업 정보를 한곳에 모아 실시간으로 간편하고,  
사전 예약·대기 기능으로 줄 서지 않고 시간을 아끼는 차별화된  
방문 경험을 제공합니다.



이수진 22세 | 학생

### 팝업마다 줄 서서 기다리는 게 너무 불편해요.

인기 팝업을 방문할 때마다 대기줄이 길고, 입장까지 얼마나 남았는지 알 수 없어요.

→ 현장 대기 상황을 미리 확인하고, 효율적으로 방문할 수 있는 서비스를 원한다.



김민수 35세 | 브랜드 매니저

### 여러 채널에 팝업을 따로 홍보하는 게 너무 번거로워요.

브랜드별로 여러 팝업을 기획하지만, 인스타·홈페이지·커뮤니티 등 각 채널에 따로 홍보해야 해서 관리가 복잡해요.

→ 한 플랫폼에서 등록과 홍보를 동시에 할 수 있는 시스템을 원한다.



## 1. 회원가입 및 로그인

비회원은 팝업 탐색만 가능  
예약·리뷰 시 로그인 필요 안내  
SNS 간편가입 또는 회원가입 후 로그인

## 2. 메인페이지 탐색

이달의 추천, 테마, 종료 임박 팝업 등  
카테고리로 탐색 가능

## 3. 북마크 관리

북마크 등록/삭제로 관심 팝업 관리

## 6. 리뷰 및 문의

체크인 후 리뷰, 문의 작성 가능

## 5. 현장 체크인

현장 도착 시 앱 내 체크인 버튼 클릭  
→ QR 생성 → 운영자 스캔 인증

## 4. 예약 및 대기 등록

팝업 상세 페이지에서 날짜·시간 선택 후  
예약 또는 대기 신청  
마이페이지 > 내 팝픽에서 예약 현황 확인

**1. 회원가입 및 승인**

사업자등록번호 입력 후 승인 절차 완료  
→ 운영자 계정 생성 및 로그인

**2. 메인페이지 진입**

운영자 정보 및 등록한 팝업 목록,  
예약 현황, 리뷰 현황 확인 가능

**3. 팝업관리**

팝업스토어 등록 및 수정,  
상세정보를 통한 리뷰, 문의 관리

**5. 현장관리**

사용자 현장 대기 상태 확인 → 호출 및 입장 처리  
QR 스캔을 통한 체크인 관리

**4. 예약관리**

사용자가 예약한 내역 확인 및 관리  
예약 승인, 취소, 시간대별 예약 현황 모니터링



Home

Roles

Contact



# 프로젝트 팀 구성 및 역할

Learn More A white button with rounded corners and a thin red border, containing the text "Learn More" and a small white right-pointing arrow icon.



정지현

프로젝트 총괄 관리



지원준

DB 설계 및 토큰 관리



김아영

Figma를 활용한 UI 시안 제작



임유림

회원가입 및 계정 CRUD 구현

팝업 예약·대기 및 결제 기능 구현

팝업 상세 페이지 및 승인·반려 처리

공통 컴포넌트 및 메인 페이지 구축

예약·대기 현황 및 리뷰 조회

카카오 지도 API

마이페이지 프로필 RUD 구현

게시글 글 CRUD 기능 및 관리

예약자·대기자 조회 및 검색 기능

팝업 CRUD 및 카카오맵 지오코딩 API

예약 확인 모달 및 대기 페이지 구현

QR 생성 및 체크인

북마크 CD 구현

리뷰·문의 CRUD 및 운영상태 표시

CSS 스타일링 및 발표자료 제작

소셜 로그인(구글·네이버·카카오) 인증

토큰 만료 관리

계정 정지·승인



Home

Procedure

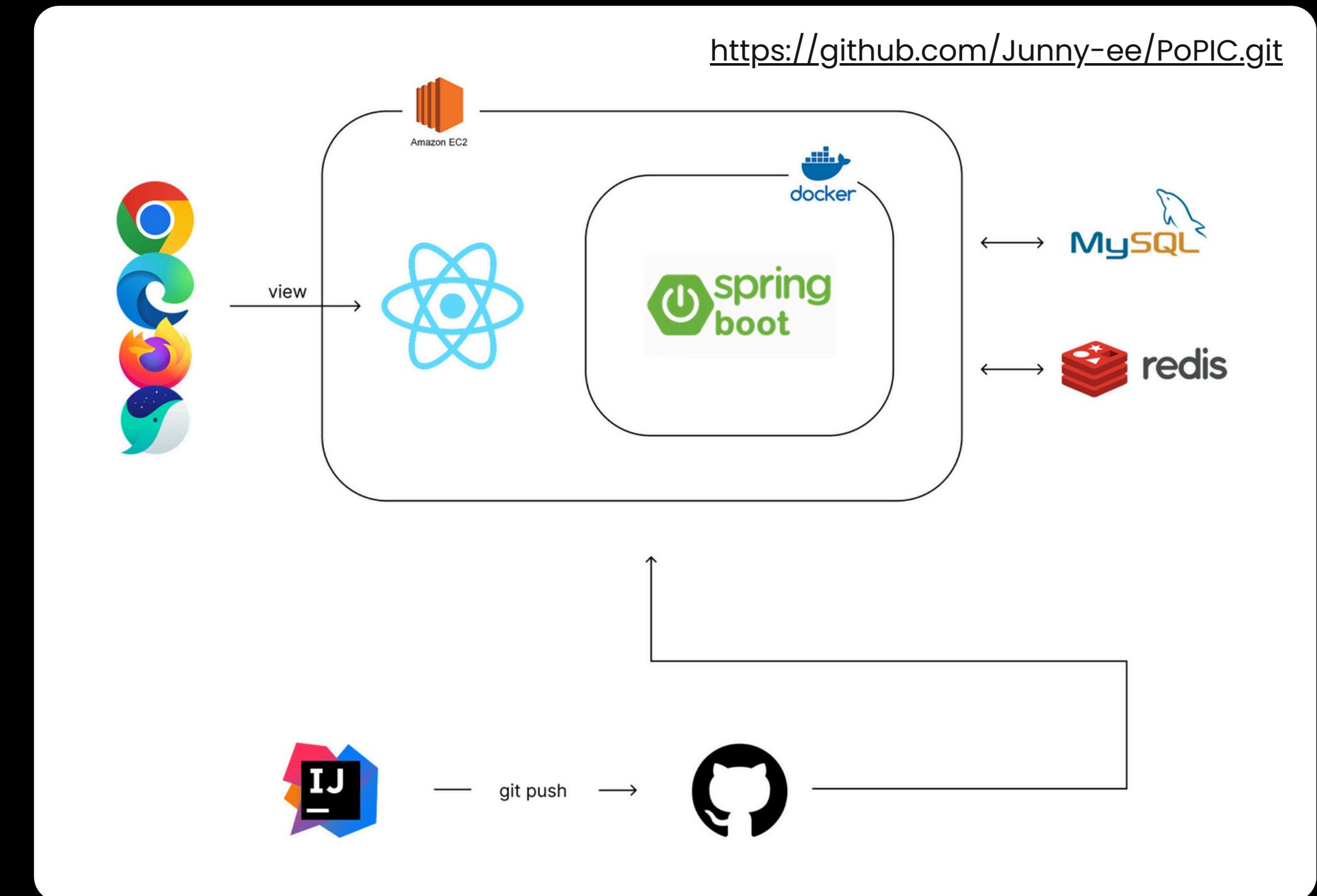
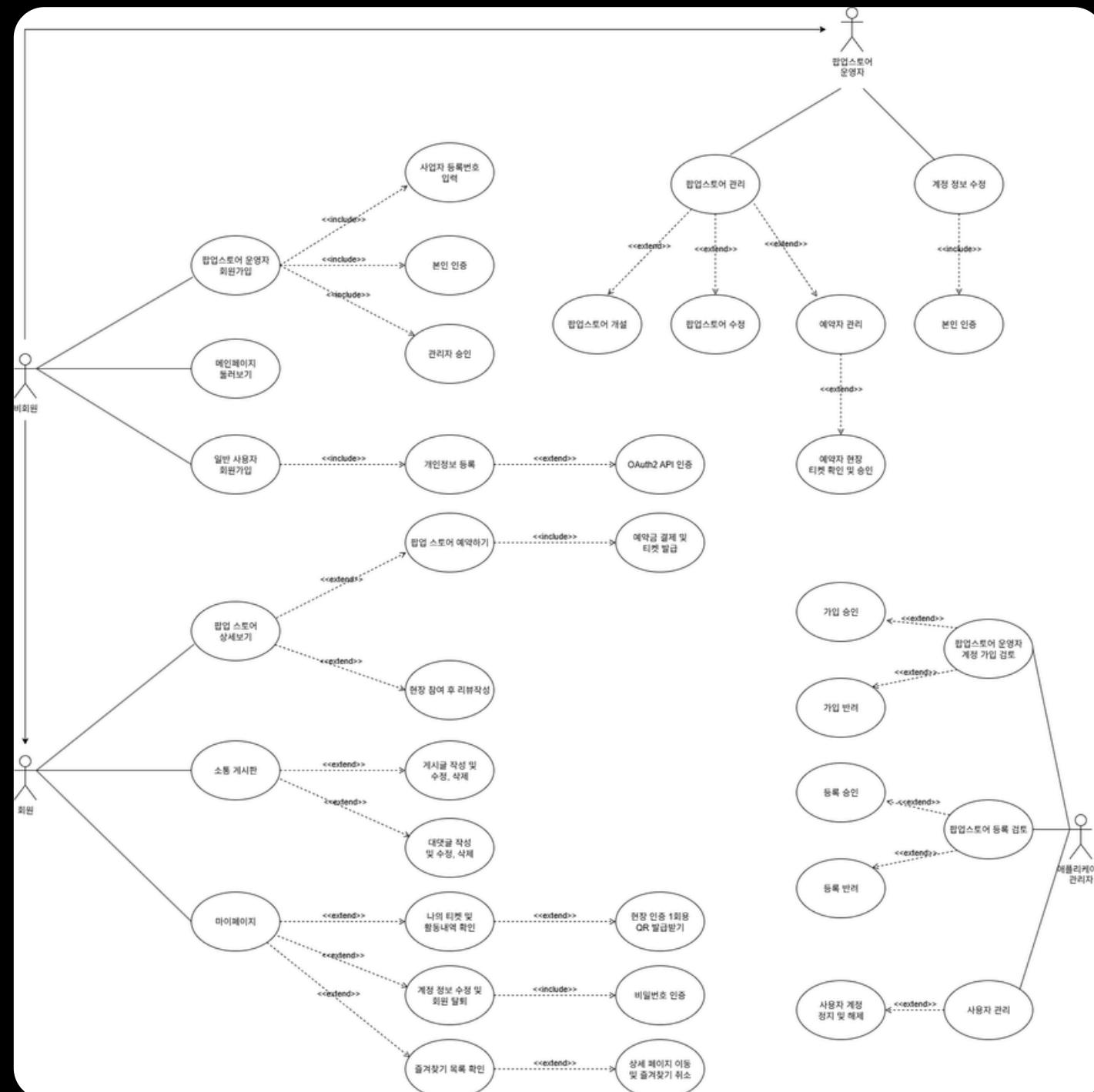
Contact



# 프로젝트 수행 절차 및 방법

Learn More ➔







Home

Feature

Contact



# 프로젝트 시연 및 기능소개

Learn More ➔



Home

Demo

Contact



# 프로젝트 시연 영상

**DB 트랜잭션**

연관 데이터 정합성 유지

다단계 작업 원자성 보장

M:N 관계 무결성 확보

운영 및 예약 데이터 신뢰성 강화

**API**

OAuth2 로그인 (카카오·네이버·구글)

토스페이먼츠 결제 처리 API 연동

사업자등록번호 검증 API 연동

카카오맵 API(지도 표시·주소→좌표 변환)

**Redis**

QR 코드 임시 저장(인메모리 캐시)

TTL 5분 만료 설정 · 자동 삭제 처리

체크인 성공 시 해당 QR 즉시 제거

누적 데이터 차단 · 메모리 사용 최적화

**비동기  
이벤트 처리**

SSE 실시간 통신 기반 데이터 갱신

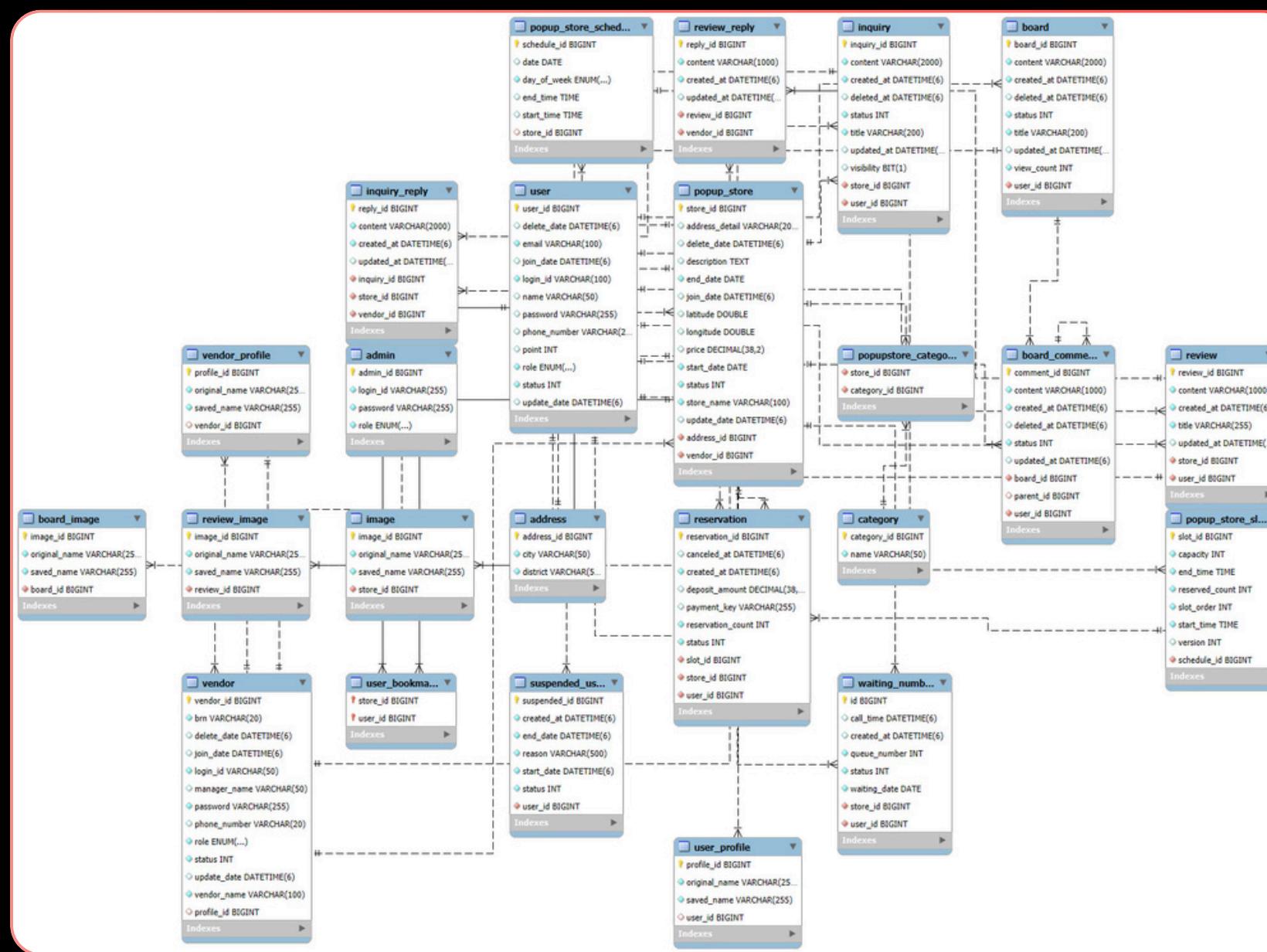
입장·예약 취소 이벤트 즉시 반영

QR 정보 자동 업데이트 및 화면 동기화

사용자 입력 변화에 따른 UI 반응형 처리



# DB 트랜잭션



01

## 연관 데이터 정합성 유지

회원, 예약, 팝업, 리뷰 간 참조 무결성을 보장하고  
외래키와 CASCADE 설정으로 데이터 불일치 방지

02

## 다단계 작업 원자성 보장

예약 생성 시 예약 등록 후 결제 정보 저장 순으로 트랜잭션 처리하며,  
중간 실패 시 전체 ROLLBACK으로 데이터 일관성 유지

03

## M:N 관계 무결성 확보

회원 ↔ 팝업(북마크), 사용자 ↔ 예약(대기·체크인) 관계를  
다대다 매팅 테이블로 관리하고, 예약 중복 방지를 위해 UNIQUE 제약 적용

04

## 운영 및 예약 데이터 신뢰성 강화

예약 상태 변경 시 일관성 유지,  
통계용 집계 테이블과 실시간 예약 테이블을 분리하여 관리



# OAuth2 로그인

```
const NaverCallback = () => {
  const [params : URLSearchParams] = useSearchParams();

  useEffect( effect: () => {
    const code : string = params.get("code");
    const state : string = params.get("state");
    const savedState : string = localStorage.getItem( key: "naver_oauth_state");

    if (state !== savedState) {
      console.error("CSRF state mismatch!");
      return;
    }
    window.location.replace(
      url: `http://localhost:8080/auth/naver/callback?code=${encodeURIComponent(code)}&state=${encodeURIComponent(state)}`;
    );
  }, [deps: []]);

  return <div>로그인 처리 중...</div>;
};

export default NaverCallback;
```

## 01 소셜 제공자 인증

구글·카카오·네이버의 권한 부여 코드 흐름으로 사용자를 인증합니다.  
최초 로그인은 프로필을 로컬 계정에 매핑합니다.

```
// 네이버 로그인
const naverLogin = () => {
  console.log("naver login");
  const NAVER_CLIENT_ID = import.meta.env.VITE_NAVER_CLIENT_ID;
  const NAVER_REDIRECT_URI = import.meta.env.VITE_NAVER_REDIRECT_URI;

  console.log("NAVER_CLIENT_ID: ", NAVER_CLIENT_ID);
  console.log("NAVER_REDIRECT_URI: ", NAVER_REDIRECT_URI);

  const handleNaverLogin = () => {
    // keep 추가
    const state : string = `${crypto.randomUUID()}:${key}`;
    localStorage.setItem("naver_oauth_state", state);

    // 로그인 유지 여부 전달용
    const naverAuthUrl : string =
      `https://nid.naver.com/oauth2.0/authorize` +
      `?response_type=code` +
      `&client_id=${NAVER_CLIENT_ID}` +
      `&redirect_uri=${encodeURIComponent(NAVER_REDIRECT_URI)}` +
      `&state=${encodeURIComponent(state)}`;

    window.location.href = naverAuthUrl;
  };
  handleNaverLogin();
};
```

## 02 토큰 발급·검증

Access/Refresh 조합으로 재발급·자동 로그인 흐름을 구성합니다.  
서버 유효성·만료 검증, 브라우저 쿠키 저장을 권장합니다.

```
public void naverCallback(@RequestParam("code") String code,
  @RequestParam("state") String state,
  @RequestParam(value = "keep", required = false, defaultValue = "false") Boolean keep,
  HttpServletResponse response) throws Exception {
  System.out.println("네이버 클릭 도착, code = " + code + ", state = " + state);

  // 1) 네이버 유저 정보
  NaverUserInfo info = naverLoginService.getUserInfo(code, state);

  // 2) 없으면 가입, 있으면 조회 (메시드명은 프로젝트에 맞게)
  User u = userService.registerOrLoginFromNaver(info);

  // 3) JWT 발급
  String access = jwtUtil.createAccessToken(u.getLogin_id(), String.valueOf(u.getRole()), u.getName());
  String refresh = jwtUtil.createRefreshToken(u.getLogin_id());

  // 로그인 유지 여부 param
  long maxAge = keep ? java.time.Duration.ofDays(14).getSeconds() : -1;

  // 4) refresh 토큰 httpOnly 쿠키로 심기
  ResponseCookie refreshCookie = ResponseCookie.from( name: "refreshToken", refresh)
    .httpOnly(true)
    .secure(false) // 배포 HTTPS면 true
    .sameSite("Lax")
    .path("/")
    .maxAge(maxAge)
    .build();
  response.addHeader(org.springframework.http.HttpHeaders.SET_COOKIE, refreshCookie.toString());

  // 소셜 정보 부록시 join 창으로 이동
  boolean need = needsMoreInfo(u);
  String base = need ? "/join" : "/main";
  String redirect = frontendBaseUrl + base
    + "?role=USER"
    + "&social=naver"
    + "&need=" + (need ? "1" : "0")
    + "&token=" + URLEncoder.encode(access, StandardCharsets.UTF_8)
    + "&name=" + URLEncoder.encode(nvl(u.getName()), StandardCharsets.UTF_8)
    + "&email=" + URLEncoder.encode(nvl(u.getEmail()), StandardCharsets.UTF_8)
    + "&phone=" + URLEncoder.encode(nvl(u.getPhone_number()), StandardCharsets.UTF_8);
  response.setStatus(302);
  response.setHeader( s: "Location", redirect);
```



# 토스페이먼츠 API

```
@PostMapping("/confirm")
public ResponseEntity<?> confirmPayment(@RequestBody PopupReservationDTO dto)
    try {
        PopupReservationDTO saved = reservationService.reserveSlot(
            dto.getSlot().getSlot_id(),
            dto.getUser().getUser_id(),
            dto.getPopup().getStore_id(),
            dto.getReservationCount(),
            dto.getDepositAmount(),
            dto.getPaymentKey()
        );
        return ResponseEntity.ok(saved);
    } catch (IllegalStateException e) {
        return ResponseEntity.status(HttpStatus.CONFLICT)
            .body(Map.of("message", e.getMessage()));
    }
}
```

```
// 토스페이먼츠 결제 로드 -> 위젯 초기화
useEffect( effect: () => {
    (async () => {
        const tossPayments :TossPaymentsSDK = await loadTossPayments(clientKey);
        const w :TossPaymentsWidgets = tossPayments.widgets( params: {customerKey});
        setWidgets(w);
    })();
}, deps: []);

// 결제 수단 렌더링
useEffect( effect: () => {
    (async () => {
        if (!widgets) return;
        await widgets.setAmount(amount);
        await widgets.renderPaymentMethods( params: {
            selector: "#payment-method",
            variantKey: "DEFAULT",
        });
        await widgets.renderAgreement( params: {
            selector: "#agreement",
            variantKey: "AGREEMENT",
        });
        setReady( value: true);
    })();
}, deps: [widgets, amount]);

// 결제 요청
const requestPay = async () => {
    if (!widgets) return;
    await widgets.requestPayment({
        orderId: generateRandomString(),
        orderName: `${searchParams.get("name")}` 예약`,
        successUrl:
            window.location.origin +
            `/success?people=${people}&popupId=${popupId}&slotId=${slotId}&amount=${amount}&date=${date}`,
        failUrl: window.location.origin + "/fail",
        customerName: user?.name ?? "",
        customerEmail: user?.email ?? ""
    });
};
```

```
sync function confirm() {
    const response :Response = await fetch( input: `${URL}/reservations/confirm`, init: {
        method: "POST",
        headers: {
            "Content-Type": "application/json",
            Authorization: `Bearer ${token}`,
        },
        credentials: "include",
        body: JSON.stringify(requestData),
    });

    if (!response.ok) {
        const json = await response.json().catch(() => null);
        console.error("예약 확인 실패:", response.status, json);
        navigate(`/fail?message=${json?.message || "권한 오류"}&code=${response.status}`);
        return;
    }

    // 결제 이름 가져오기
    fetch( input: `${URL}/popupStore/popupDetail?id=${popupId}`, init: {
        headers: {Authorization: `Bearer ${token}`},
    }) Promise<Response>
        .then((res :Response) => res.json()) Promise<any>
        .then((popup) => {
            setPopupName(popup.store_name);
        });

    // 슬롯 시간 가져오기
    if (date) {
        fetch( input: `${URL}/popupStore/slots?popupId=${popupId}&date=${date}`, init: {
            headers: {Authorization: `Bearer ${token}`},
        }) Promise<Response>
            .then((res :Response) => res.json()) Promise<any>
            .then((slots) => {
                const slot = slots.find(s => String(s.slot_id) === String(slotId));
                if (slot) {
                    setSlotDate(slot.schedule.date);
                    setSlotTime(slot.start_time);
                }
            });
    }
}
```

## 01 결제 위젯 렌더링

결제 금액과 수단이 자동 반영된 위젯을 통해 표준화된 UI와 안정적인 결제 환경을 제공합니다.

## 02 결제 요청 및 검증 처리

결제 요청 후 서버에서 검증과 예약 확정을 수행해 성공/실패 상태를 명확히 구분하고 데이터 일관성을 유지합니다.



# 사업자등록번호 검증 API

```
const businessNumberCheck = () => {
  const input : null = brnRef.current; // 인풋 DOM
  const clean : string = (form.brn || "").replace( searchValue: /\D/g, replaceValue: "" );

  if (clean.length !== 10) {
    input.setCustomValidity( error: "사업자등록번호 10자리를 입력해주세요." );
    input.reportValidity();
    return;
  }

  const url : string = "https://api.odcloud.kr/api/nts-businessman/v1/status"
  + "?serviceKey=u%2FoWK2f4UUhnQFKqgTJe96B2%2FNKnFcW0yEX01AFuPBGp8mH1%2B14dEoJJQCU5flSkJ%2Ba7YDazsdLnC"
  + "&brn=" + clean
  + "&format=json"
}
```

```
$.ajax({
  url,
  type: "POST",
  data: JSON.stringify( value: {b_no: [clean]} ),
  dataType: "json",
  contentType: "application/json",
  success: function (result) {
    const item = result?.data?.[0];

    if (!item) {
      input.setCustomValidity( error: "인증 결과를 확인할 수 없습니다." );
      input.reportValidity();
      return;
    }
    // 국세청 등록된 사업자
    const ok =
      typeof item?.tax_type === "string" &&
      (item.tax_type.includes("부가가치세") || item.tax_type === "부가가치세");

    if (ok) {
      input.setCustomValidity( error: "" );
      setBrnVerified( value: true );
    } else {
      input.setCustomValidity( error: "등록되지 않은 사업자 번호입니다." );
      input.reportValidity();
    }
  },
  error: function () {
    input.setCustomValidity( error: "인증 중 오류가 발생했습니다. 잠시 후 다시 시도해주세요." );
    input.reportValidity();
  }
},
```

01

## 유효성 검증

사업자번호 형식·체크섬 확인 후 외부 API로 등록/상태 조회합니다.  
검증 결과를 운영자 등록·수정 흐름에 즉시 반영합니다.

02

## 예외·보안 처리

실패 사유 안내·재시도 정책을 표준화합니다.  
번호 저장 시 마스킹·암호화로 불필요한 노출을 최소화합니다.



# 카카오맵 API

```
const initKakaoMap = () => {
  const container : HTMLElement = document.getElementById( elementId: "map");
  if (!container || !window.kakao?.maps) return;

  const options : {center: Window.kakao.maps.LatLng, level: number} = {
    center: new window.kakao.maps.LatLng(props.popup.latitude, props.popup.longitude),
    level: 3,
  };
  const map = new window.kakao.maps.Map(container, options);
  map.setZoomable(false);

  const markerPosition : Window.kakao.maps.LatLng = new window.kakao.maps.LatLng(props.popup.latitude, props.popup.longitude);

  const imageSrc : string = "/favicon.png"; // 사용할 이미지 경로
  const imageSize : Size = new window.kakao.maps.Size(40, 40); // 마커 이미지 크기
  const imageOption : {offset: Window.kakao.maps.Point} = {offset: new window.kakao.maps.Point(20, 40)}; // 이미지 기준점 (가운데)

  const markerImage : Window.kakao.maps.MarkerImage = new window.kakao.maps.MarkerImage(imageSrc, imageSize, imageOption)
  const marker : Marker = new window.kakao.maps.Marker({
    position: markerPosition,
    image: markerImage, // 여기서 이미지 적용
  });
  marker.setMap(map);

  if (window.kakao?.maps) {
    initKakaoMap();
  } else if (!document.getElementById(scriptId)) {
    const script : HTMLScriptElement = document.createElement( tagName: "script");
    script.id = scriptId;
    script.src = `https://dapi.kakao.com/v2/maps/sdk.js?appkey=${KAKAO_APP_KEY}&autoload=false`;
    script.async = true;
    document.head.appendChild(script);
    script.onload = () => window.kakao.maps.load(initKakaoMap);
  }
};
```

## 01 카카오 지도 표시

지도 컨테이너에 카카오맵을 렌더링하고 마커로 위치를 시각화합니다.  
검색·드래그·줌 등 기본 조작으로 위치 탐색을 지원합니다.

```
// 상세주소 → 좌표 변환
const handleGeocode = useCallback(() => {
  if (
    !kakaoReady ||
    !window.kakao?.maps ||
    !window.kakao.maps.services
  ) {
    alert("카카오 지도 SDK가 아직 준비되지 않았습니다.");
    return;
  }

  if (!city || !district || !detail.trim()) {
    alert("시/구/상세주소를 모두 입력하세요.");
    return;
  }
```

## 02 주소→좌표 변환(지오코딩)

Geocoder.addressSearch로 입력 주소를 위도·경도로 변환합니다.  
결과 좌표를 지도에 반영하고 서버/폼 데이터와 동기화합니다.

```
geocoder.addressSearch(full, (results, status) => {
  if (status === window.kakao.maps.services.Status.OK && results.length > 0)
    const result = results[0];

  const hasDong = result.address?.region_3depth_name?.trim();
  const hasJibun = result.address?.main_address_no?.trim();
  const hasRoadAddr = result.road_address;

  if (!hasDong || (!hasJibun && !hasRoadAddr)) {
    alert("상세 주소를 입력하세요 (동/지번/도로명 포함).");
    return;
  }

  const nextLat = parseFloat(result.y);
  const nextLng = parseFloat(result.x);
  setLat(nextLat);
  setLng(nextLng);

  onChangeRef.current?.({
    city,
    district,
    detail: detail.trim(),
    latitude: nextLat,
    longitude: nextLng,
    addressString: [city, district].filter(Boolean).join(" "),
  });

  alert("주소 확인 완료");
} else {
  alert("주소를 다시 확인하세요.");
}
}, [city, district, detail, kakaoReady]);
```



# Redis

```

@GetMapping(value = "/generate-qr", produces = "application/json")
public ResponseEntity<Map<String, Object>> generateQR(HttpServletRequest response, @RequestParam Long reservationId) throws Exception {
    // 입장권 확인
    PopupReservationDTO reservationDTO = reservationService.findById(reservationId);

    Map<String, Object> res = new HashMap<>();
    res.put("reservation", reservationDTO);
    res.put("status", reservationDTO.getStatus() == 1 ? "OK" : "USED");

    // 직렬화
    ObjectMapper objectMapper = new ObjectMapper();
    objectMapper.registerModule(new JavaTimeModule());
    String json = objectMapper.writeValueAsString(reservationDTO);

    // 임시 토큰 생성
    String token = UUID.randomUUID().toString();
    redisTemplate.opsForValue().set(token, json, 5, TimeUnit.MINUTES);

    // QR 생성
    int width = 250;
    int height = 250;
    BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
    Graphics2D g = image.createGraphics();
    g.setColor(Color.black);
    g.fillRect(0, 0, width, height);
    String qrData = "http://192.168.23.23:8080/generate-qr/" + token;
    BitMatrix matrix = new MultiFormatWriter().encode(qrData, BarcodeFormat.QR_CODE, width, height);

    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    MatrixToImageWriter.writeToStream(matrix, "png", baos);
    String base64Image = Base64.getEncoder().encodeToString(baos.toByteArray());

    res.put("token", token);
    res.put("qrImage", "data:image/png;base64," + base64Image);
    return ResponseEntity.ok(res);
}

ObjectMapper objectMapper = new ObjectMapper();
objectMapper.registerModule(new JavaTimeModule());
PopupReservationDTO reservationDTO = objectMapper.readValue(reservationData, PopupReservationDTO.class);

// 예약 관영자와 로그인한 관영자가 동일한지 체크
if(!reservationDTO.getPopup().getVendor().getVendor_id().equals(customUserPrincipal.getId())) {
    throw new RuntimeException("권한이 없는 사용자입니다.");
}

if (reservationDTO.getStatus() != 1) {
    throw new RuntimeException("유효하지 않은 티켓입니다.");
}
reservationService.entryReservationById(reservationDTO.getReservationId());
response.put("success", true);
response.put("data", reservationData);
redisTemplate.delete(token);
return response;
}

```

01

## QR 코드 임시 저장

체크인용 QR 데이터를 **Redis** 인메모리 캐시에 임시 저장하여 데이터베이스 접근 부하를 줄이고 빠른 응답성을 제공합니다.

02

## TTL(Time To Live) 설정

각 QR 정보에 5분 TTL을 부여하여 일정 시간이 지나면 자동 삭제되도록 구성, 사용 후 잔여 데이터가 남지 않도록 관리합니다.

03

## 체크인 시 즉시 제거

사용자가 입장 또는 체크인을 완료하면 해당 QR 정보를 즉시 삭제하여 불필요한 캐시 데이터가 쌓이지 않게 처리합니다.

04

## 메모리 효율성 최적화

주기적인 만료 처리와 실시간 삭제로 **누적 데이터 최소화 및 메모리 사용 효율성**을 높입니다.



# 비동기 이벤트 처리

```
@GetMapping(value = "/qr-stream", produces = MediaType.TEXT_EVENT_STREAM_VALUE) 🌐 wonjoon
public SseEmitter streamQrStatus(@RequestParam String token) {
    SseEmitter emitter = new SseEmitter(timeout: 0L); // 타임아웃 없음
    ObjectMapper objectMapper = new ObjectMapper();

    Executors.newSingleThreadExecutor().execute(() -> {
        try {
            String lastStatusStr = null;

            while (true) {
                String reservationData = redisTemplate.opsForValue().get(token);
                String statusStr;

                if (reservationData == null) {
                    statusStr = "USED"; // 이미 스캔됨
                } else {
                    // JSON 파싱
                    JsonNode rootNode = objectMapper.readTree(reservationData);
                    int status = rootNode.path("status").asInt();
                    switch (status) {
                        case 0:
                            statusStr = "USED";
                            break;
                        case 1:
                            statusStr = "OK";
                            break;
                        case -1:
                            statusStr = "CANCELED";
                            break;
                        default:
                            statusStr = "UNKNOWN";
                    }
                }

                // 상태가 변했을 때만 전송
                if (!statusStr.equals(lastStatusStr)) {
                    emitter.send(statusStr);
                    lastStatusStr = statusStr;
                }

                // 이미 사용됐거나 취소됐으면 종료
                if ("USED".equals(statusStr) || "CANCELED".equals(statusStr))
                    break;
            }

            Thread.sleep(2000); // 2초마다 상태 체크
        }
    });

    emitter.complete();
} catch (Exception e) {
    emitter.completeWithError(e);
}

return emitter;
}
```

01

## SSE 실시간 통신

**Server-Sent Events (SSE)**를 이용해 서버 상태 변화를 클라이언트로 즉시 전송, 실시간 정보 갱신을 지원합니다.

02

## 입장·예약 취소 이벤트 반영

입장 완료, 예약 취소 등 주요 이벤트 발생 시 해당 QR 정보를 모든 클라이언트 화면에 동시에 갱신합니다.

03

## QR 정보 자동 업데이트

변경된 QR 상태를 실시간으로 동기화하여 화면 새로고침 없이 최신 상태를 유지합니다.

04

## 반응형 UX 처리

사용자 입력에 따른 UI 변화를 즉시 반영하여 자연 없는 인터랙션과 자연스러운 사용자 경험을 제공합니다.



Home

Strategy

Contact



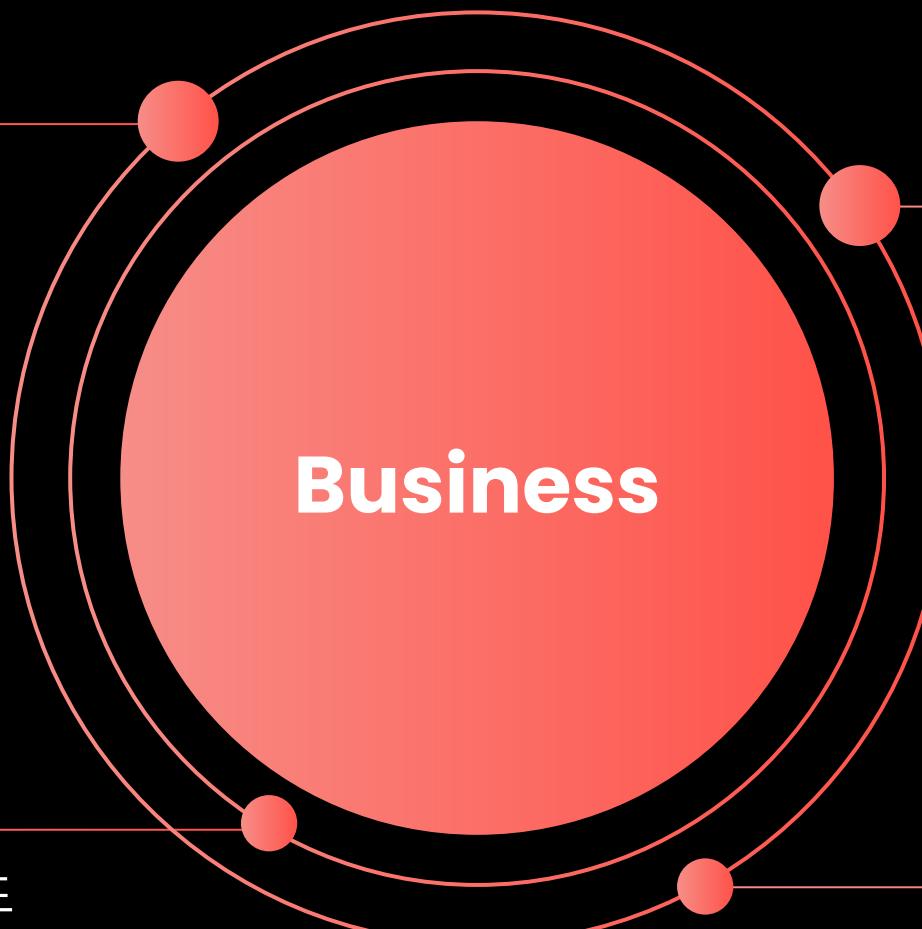
# 비즈니스 및 마케팅 전략

Learn More ➔



### 운영 효율에 따른 비용 절감

QR 체크인·예약 자동화로 **현장 인력 투입 최소화**



**Business**

### 정보 접근성 향상

흩어진 팝업 정보를 **한 플랫폼**에서 통합 조회 가능

### 재방문·충성 고객 확대

리뷰·북마크 기능으로 **재방문율 상승 유도**

### 대기 없는 방문 경험

예약·대기·체크인을 시스템화 **현장 혼잡과 불편 최소화**



### 브랜드 노출 극대화

추천·테마 큐레이션으로 **자연스러운 노출과**  
내부 트래픽 확보

### 실시간 피드백 반영

리뷰·평점을 실시간 수집·반영해 **서비스 품질 개선**

### Marketing

### 신뢰도 중심 운영

인증 리뷰 및 표준화된 시스템으로 **브랜드 신뢰 강화**

### 트렌드 데이터 활용

지역·연령·시간대별 이용 데이터를 기반으로  
**맞춤형 캠페인 전개**



## 노쇼 방지 결제 시스템

결제 선승인 방식으로 무단 예약·이탈을  
예방하고 **운영 리스크 최소화**

## AI 기반 추천·운영 지원

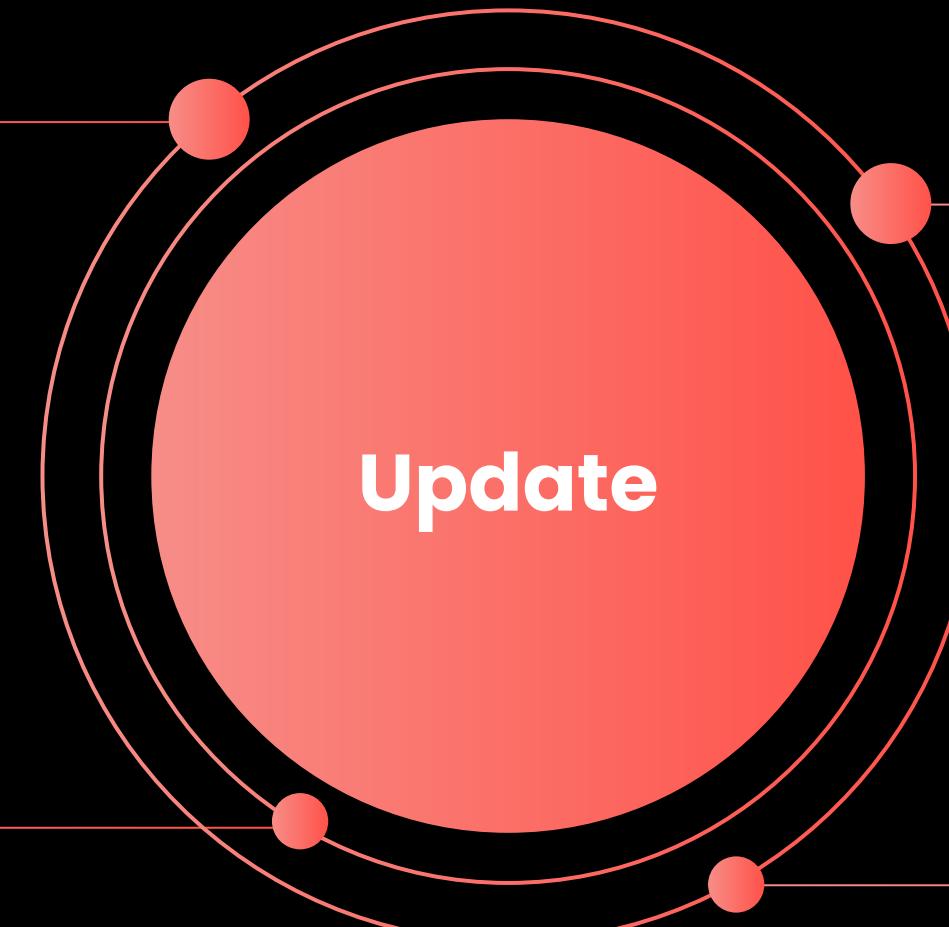
AI가 예약·리뷰 데이터를 분석해  
**맞춤형 팝업 추천 및 수요 예측 지원**

## 사용자 신고·검증 서비스

리뷰·운영 불만 접수를 즉시 처리해  
신뢰도 높은 커뮤니티 유지

## 알림 및 리마인드 기능

예약 일정·새 팝업 오픈 등 주요 이벤트를  
**자동 알림으로 제공**





Home

Reflection

Contact



# 자체 평가 및 개인 의견

Learn More ➔



정지현

새로운 API를 다루다보니 시간이 많이 소요됐었는데 공식 문서와 팀원들의 도움으로 잘 해결해낸 것 같습니다. 초반에 주제 선정 시에 의견 충돌이 있었는데 서로 대화를 통해 방향을 맞출 수 있었고 각자 주어진 일에 열심히 해주고 잘 따라준 것 같아서 수월하게 진행된 것 같습니다.

프론트엔드와 백엔드를 분리해서 진행한 적은 처음이었는데, 협업이 훨씬 효율적이라고 느꼈습니다. 이번 프로젝트를 통해 큰 규모의 시스템이 어떻게 구성되고 동작하는지 알게되었습니다.



지원준

ZXing라이브러리를 활용해 QR 생성 기능을 구현하며 실시간 데이터 처리의 중요성을 체감했습니다. Redis를 통한 임시 데이터 저장과 만료 시간을 설정해 빠른 응답성과 서버 부하 분산을 동시에 달성했고, 이를 통해 실시간 인증 흐름의 효율적 관리 구조를 이해할 수 있었습니다.

또한 별도의 토큰 관리 기능을 통해 사용자 인증과 보안 로직을 다루며 데이터 유효성 검증과 만료 처리의 설계 중요성을 배웠습니다.

이번 프로젝트를 진행하며 새로운 경험과 팀원분들과의 즐거운 추억을 만들 수 있어서 좋았고 한단계 더 성장하는 과정이 되어 기뻤습니다.



김아영

React와 SpringBoot를 함께 사용하는 환경에서 프론트엔드와 백엔드의 구조를 나누어 개발하는 것이 처음에는 어려웠습니다.

특히 데이터 전달 과정에서 상태 관리와 API 연동 구조를 맞추는 데 시간이 걸렸지만, 공통 컴포넌트를 설계해 여러 페이지에서 재활용하면서 개발 효율이 향상되었습니다.

이 경험을 통해 프론트와 백 간의 협업 구조를 더 깊이 이해하고, 유지보수성과 확장성의 중요성을 체감할 수 있었습니다.



임유림

소셜 로그인 기능을 구현하면서 OAuth2 인증 구조를 이해하는데 어려움이 있었지만, 토큰 발급·검증 과정과 외부 API의 동작 원리를 직접 경험하며 보안성과 사용자 편의성을 함께 고려하는 방법을 배울 수 있었습니다. 또한 다양한 플랫폼(구글·네이버·카카오)의 로그인 구조 차이를 비교하며 공통 로직을 정리하고 예외 처리를 통합하는 과정을 통해 문제 해결 능력을 키울 수 있었습니다.



Home

Q&A

Contact



# Q&A

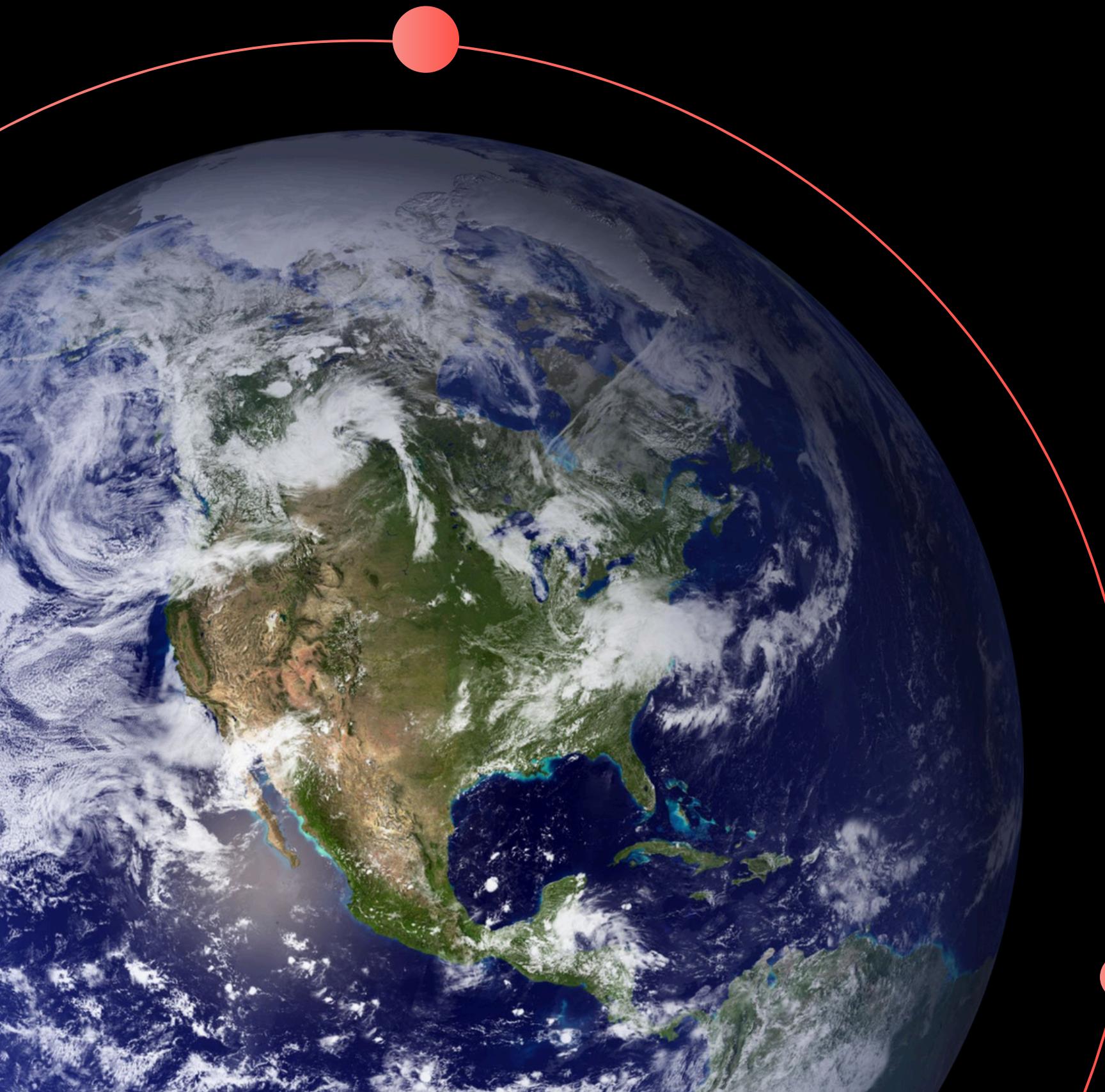
Learn More A white button with rounded corners and a thin red border, containing the text "Learn More" and a small, white, right-pointing arrow icon.



Home

About

Contact



**THANK  
YOU**