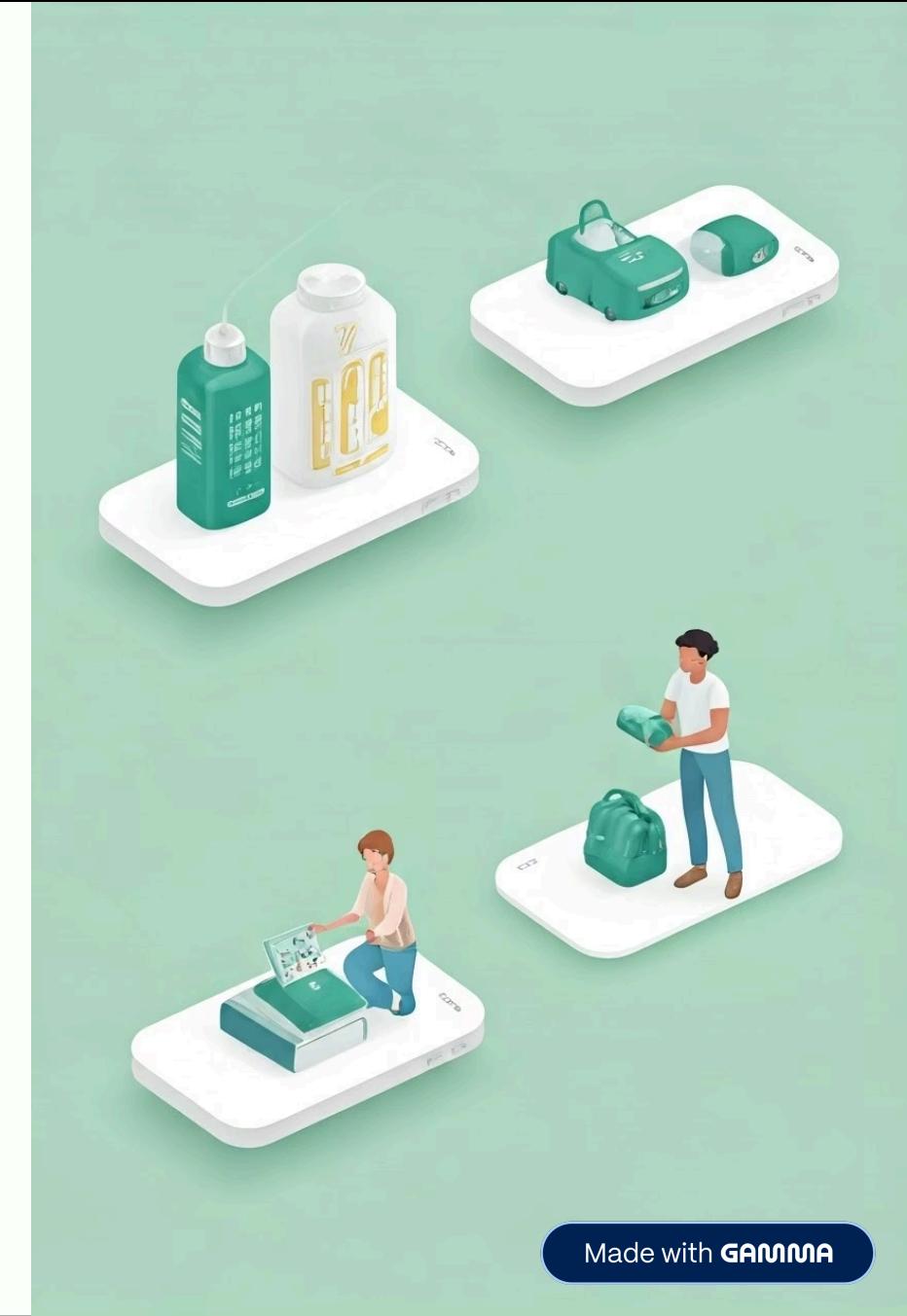


증고물품 거래 플랫폼

# RE:NT

팀원: 김한재, 박기도, 이아림, 추혜민



## Presentation 구성



### 1. RE:NT는?

주제 선정 배경

우리 팀 목표 & 차별화



### 2. 개발환경

기술스택

(FE, BE, DB, ORM)

협업도구



### 3. 서비스 & 주요 기능

판매, 대여, 무료나눔

이용자 위치 기반 상품 추천

영상 시연



### 4. 프로젝트 성과 & 보완점

프로젝트 성과

아쉬운 점

## 1. RE:NT는?



"손님 초대 때 가끔 쓸 식기세트, 꼭 사야 할까?"

"큰 맘 먹고 장만한 홈트 기구...  
옷걸이로 두기엔 아까운데..."

## 1. RE:NT는?

# RE:NT is the Next Trend

혼자 쓰기 아까운 물건, 같이 써서 가치있게



## REUSE

다시 쓰고 나누며 자원 효율성 극대화



## RECYCLE

자원·경제·환경의 지속가능한 순환



## RENT

소유보다 공유, 필요할 때만 빌려 사용

## RE:NT 프로젝트 목표



사용자 친화적 서비스



확실한 수익모델



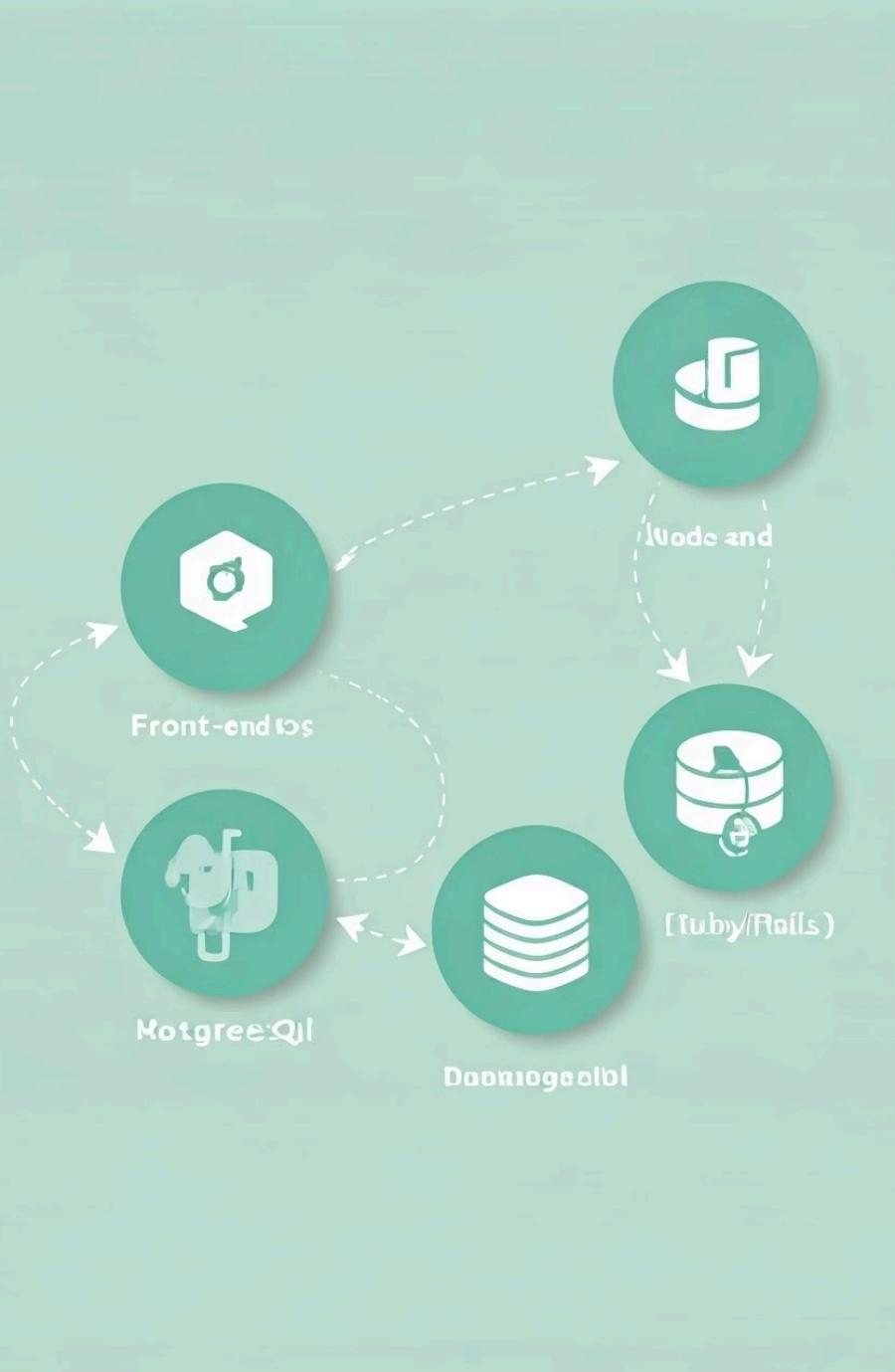
공공의 이익에 기여

안전거래 시스템 구축  
직관적 UI  
사용자 위치 기반 상품 추천

이용자간 거래 수수료를 기  
본 수익으로 확보  
공공기관&기업과의 협업 등  
공공대여 서비스 플랫폼으로  
확장 가능

자원 재활용 촉진  
나눔 커뮤니티로서 친환경적  
소비문화 형성

## 2. 개발 환경 및 기술 스택



### 프론트엔드

- **HTML/CSS/JavaScript**  
페이지 구조 및 동적 인터랙션
- **jQuery**  
AJAX 처리, DOM 조작
- **Bootstrap**  
반응형 UI 구성 및 빠른 화면 설계



### 백엔드

- **Java (Eclipse IDE 사용)**  
서버 로직, Servlet/JSP 기반 웹 애플리케이션 개발
- **JSP / Servlet**  
사용자 요청 처리, 동적 페이지 렌더링
- **MyBatis**  
Java 객체와 SQL 간 매핑 (ORM 프레임워크)
- **Apache Tomcat**  
WAS (Web Application Server)로 JSP/Servlet 구동



### 데이터베이스

- **MariaDB**  
데이터 저장 (회원, 상품, 주문, 신고, 리뷰 등)
- **HeidiSQL**  
MySQL 시각화 도구로 DB 관리 및 쿼리 테스트



### 외부 연동 및 API

- **카카오 지도 API**  
물품 거래 위치 및 사용자 주소
- **소셜 로그인 (카카오, 네이버)**  
간편 로그인
- **Toss 결제 API**  
결제 및 환불 시뮬레이션



### 협업 도구

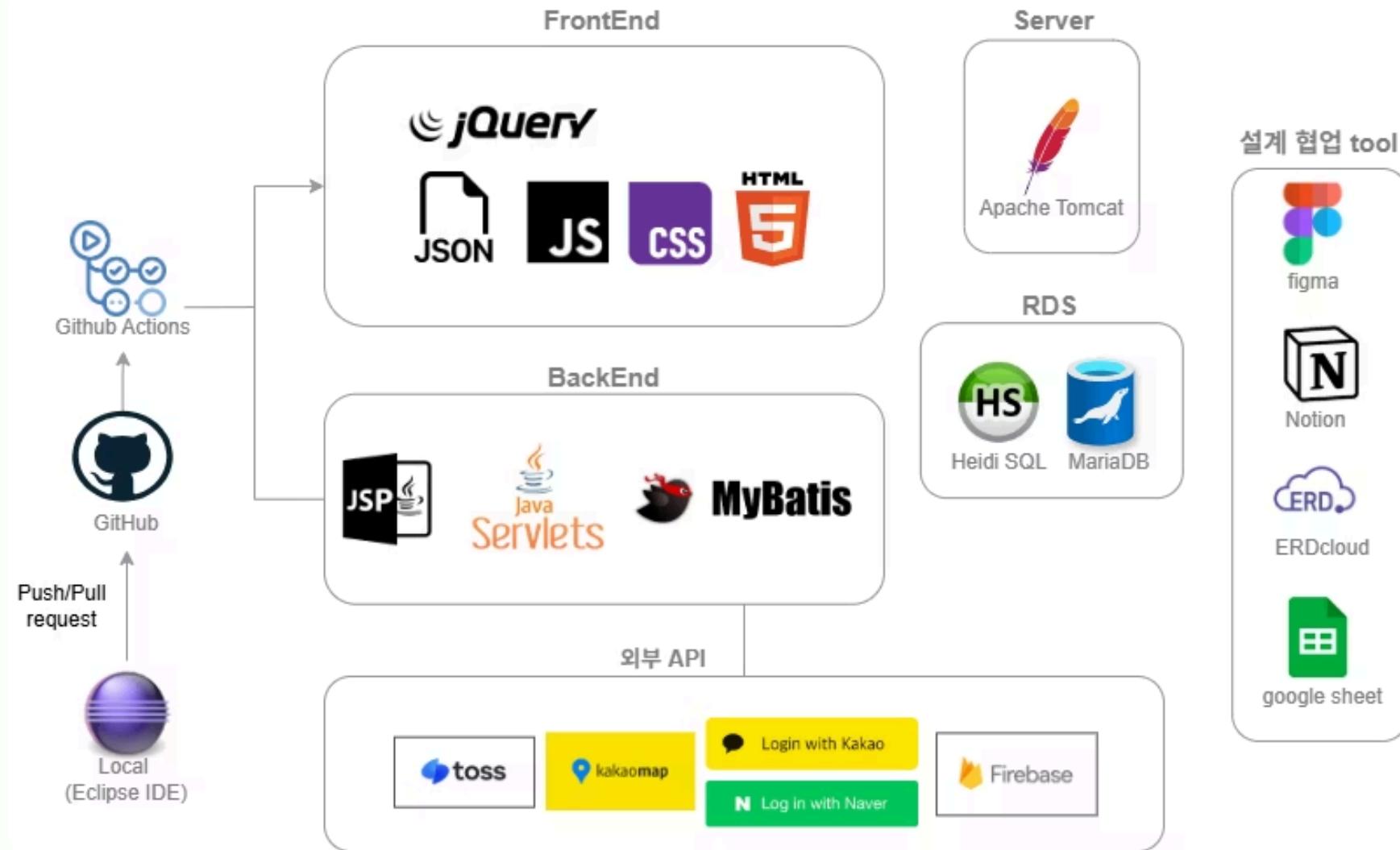
- **Git, GitHub**  
버전 관리, 협업 브랜치 운영
- **Notion**  
일정 관리, 회의록, 업무 기록



### 프로젝트 설계

- **Figma**  
화면 디자인 및 웨어프레임 제작
- **ERDcloud**  
ERD 설계 및 테이블 관계 설계
- **Google Sheet**  
기능정의서, JSP정의서 작성, 공유

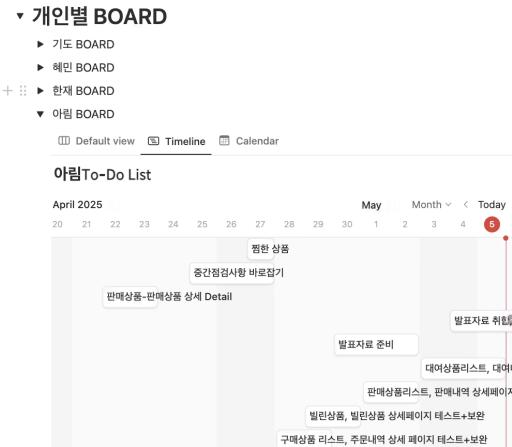
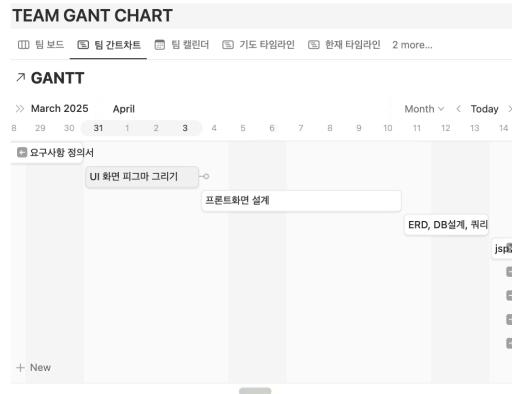
## 2. 개발 환경 및 기술 스택



# 일정관리 협업

The screenshot shows a Notion page with the title "RE:NT". It features several cards and a Gantt chart.

- Links:**
  - ERD링크
  - 깃허브 레포지토리 링크
  - JSP정의서
  - 피그마 링크
- PAGES:**
  - 서비스 소개
  - 팀 소개
  - 참조
  - 회의록
  - 컨벤션
  - 문서
- TEAM GANTT CHART:** A Gantt chart showing tasks for March 2025 and April. Tasks include "UI 화면 피그마 그리기", "프론트화면 설계", and "ERD, DB설계, 퀴리 jsp".



## 팀원이 함께 작성, 공유한 문서 아카이빙

- ERD cloud
- GitHub
- Google sheet  
(프로젝트요구정의서, JSP정의서 등)
- Figma 화면설계
- 각종 회의록 등

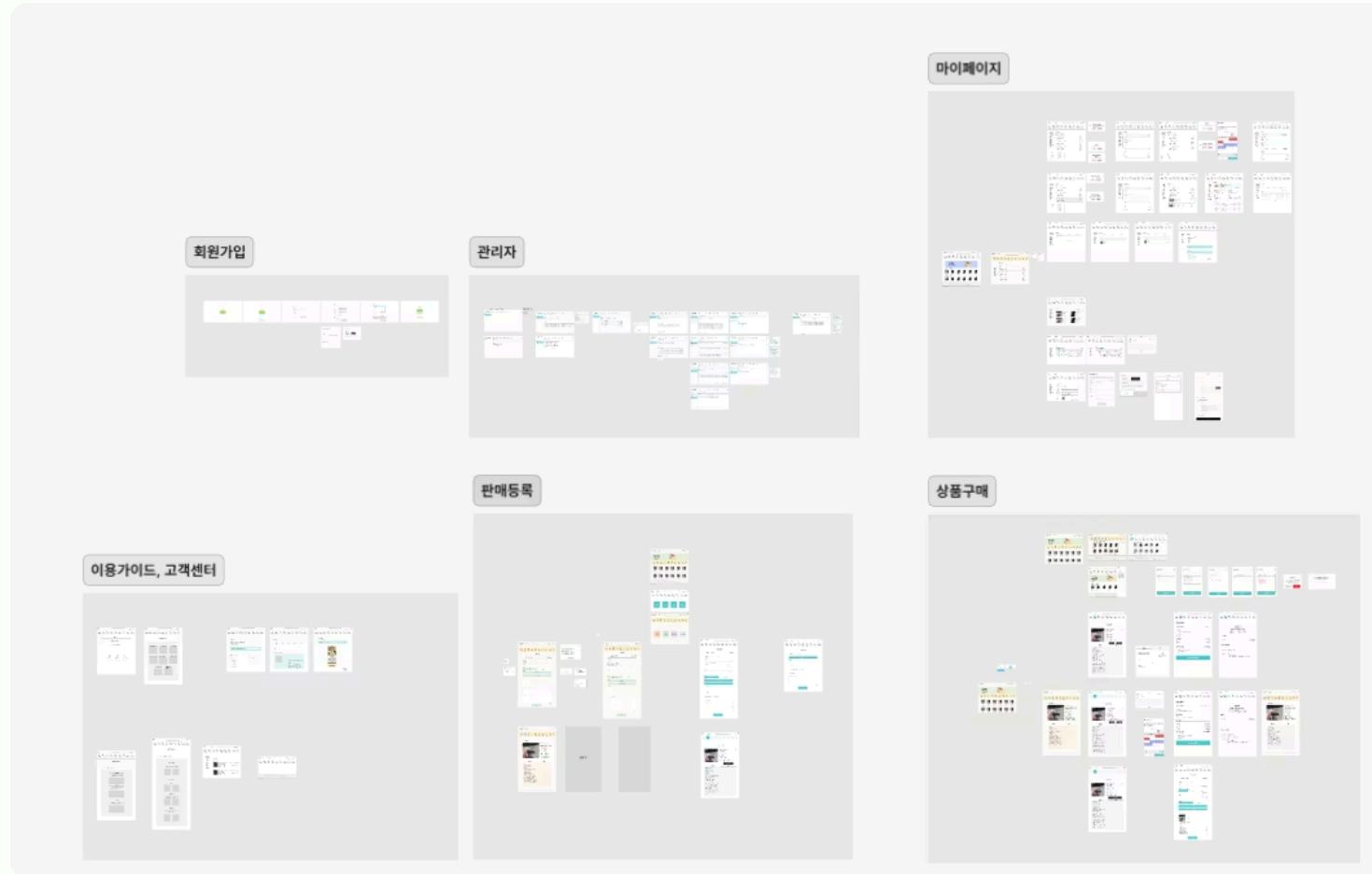
## 팀 간트차트 일정관리

- 팀 전체 일정
- 팀원별 BOARD 및 일정

→ 팀별 전체 일정과 팀원별 진행사항을 한 곳에서 공유해, 프로젝트 전체 진행상황을 효율적으로 파악

Notion LINK

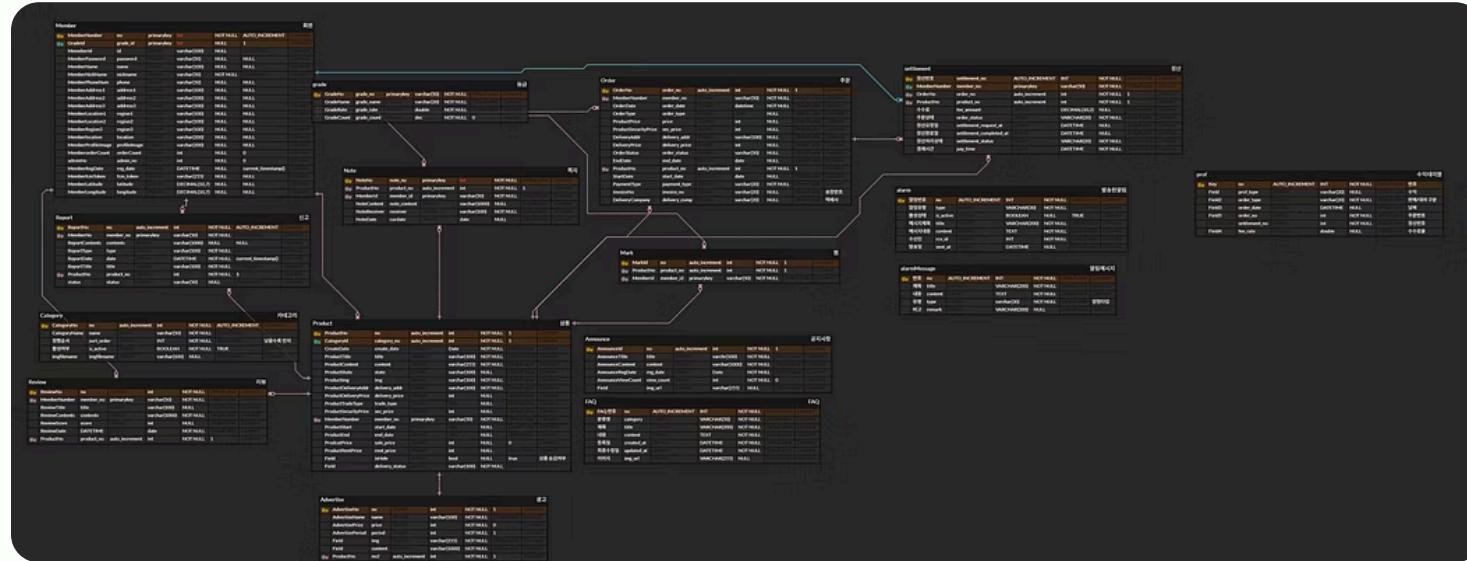
# ₩ 화면 설계



피그마(Figma)를 활용해 전체 화면 설계를 공유하였으며,  
서비스 전반의 화면 구조를 시각화하여 개발의 초석을 마련하였습니다.  
**80개의 화면을 기반으로 구성하였습니다**



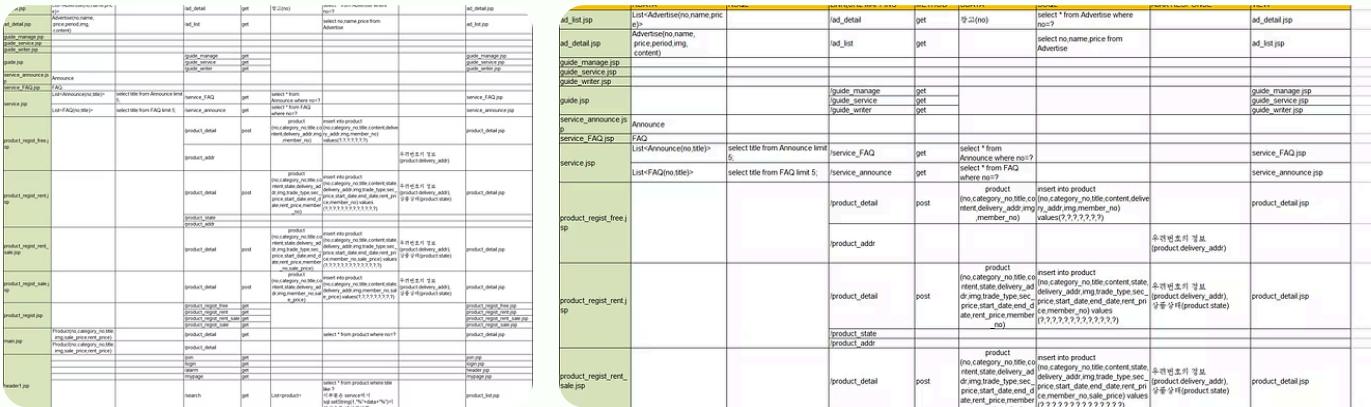
## 데이터베이스 설계



데이터베이스는 물품, 사용자, 거래, 리뷰 테이블 등 **총 16개**의 테이블을 기반으로 구성했습니다.



# jsp 정의서



프로젝트 설계 단계에서 각 JSP 화면과 서비스 흐름을 정의하고 서로 정리하여 역할과 데이터 흐름을 명확히 했습니다.

**Re:nt**는 서블릿 101개, JSP 78개로 구성된 안정적인 시스템입니다.



# GitHub 이용한 분산협업

## Network graph

Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.



팀원이 역할 분담해 각 기능을 독립적으로 개발하고 하나의 메인 브랜치로 머지함으로써 안정성을 확보했습니다.

모두 적극적으로 기여하고, 충돌 없이 머지하는 과정을 통해 협상 관리의 중요성과 효율적인 협업 방식을 경험했습니다.

### 3. 서비스&주요기능

## 👤 액터별 서비스 정리



### 상품 등록, 주문된 상품 배송

판매, 대여, 나눔의 방식으로

거래 상품을 등록할 수 있습니다.

등록 후 카테고리 및

검색에서 노출됩니다.

### 쪽지 기능

거래 전 물품에 대해 궁금한 점을

1:1로 판매자에게

상세히 문의할 수 있습니다.

### 결제 시스템

토스 페이먼츠를 통해  
'결제 버튼 → 인증 → 완료'

최소한의 클릭으로

간편 결제를 완료 가능합니다

### 찜, 게시물공유, 상품리뷰, 평점

회원은 관심있는 상품을 '찜'하고, 링크를 공유할 수 있고,  
상품 구매자는 구매한 상품에 대해 리뷰를 남길 수 있습니다.

구매/대여 전, 찜과 리뷰 등 다른 사용자의 평가를 통해  
신뢰도를 사전에 확인할 수 있습니다.

찜과 리뷰는 마이페이지에서 확인 및 관리합니다.

### 검색 기능

카테고리, 추천, 위치, 최신 필터링으로  
원하는 물품을 빠르게 찾을 수 있습니다.

다양한 옵션 및 직접 검색으로

사용자 취향에 맞춘 검색이 가능합니다.



### 회원 정보 및 주문내역 관리

전체 회원 목록, 신고 상품,

주문 내역과 정산 상태를

실시간으로 확인할 수 있습니다

조회 후, 구매자가 신고한 상품에 대한 노출 처리가 가능합니다

판매자에게 상품의 정산 처리가 가능합니다.

### 공지 및 메시지 등록·수정 관리

상품 및 FAQ 카테고리와

알림 메시지, 공지사항 등을

등록·수정하여 시스템 정보를

최신 상태로 유지합니다.



# RE:NT 거래 프로세스 흐름

## 판매자 상품 등록

판매자는 상세 정보와 함께  
상품을 등록합니다

1

## 구매자 상품검색, 상호작용

상품 검색, 상품 좋아요, 공유(링크, 카톡),  
쪽지, 신고 기능을 활용합니다.

2

## 판매자 발송 관리

구매자가 상품을 결제하면  
판매자가 상품의 배송을 시작합니다.

3

판매자도 쪽지 답변을 통해  
구매자와 수시로 소통합니다.

4

## 구매자 리뷰, 별점 작성

상품 구매자는 구매한 상품에 대해  
리뷰와 별점을 남길 수 있습니다

5

## 관리자 주문 및 정산 처리

관리자는 거래 내역을 확인하고 판매자에  
게 정산을 진행합니다.

신고상품에 대한 노출 조정을 진행합니다.



# 기술적 특화 요소

## ✓ 데이터 연동

- 회원가입부터 상품 주문, 정산까지 MyBatis 기반으로 안정적이고 신뢰성 있는 데이터 연동을 구현
- 관리자 승인 및 사용자 알림까지 연결되는 자동 업데이트로 관리·처리 효율을 극대화

## ✓ 결제

- Toss API 연동으로 외부 인증 기반의 안전한 결제를 제공
- 비동기 처리와 최소 클릭 설계로 사용자 경험을 최적화
- 서비스 내에서 환불 가능 여부(상품 발송 전 상태)를 통합 관리하며, 사용자가 취소요청 시 신속하게 환불을 처리

## ✓ 실시간 알림

- FCM(Firebase Cloud Messaging)을 통해 회원가입, 결제 완료, 대여 지연 등 주요 정보가 실시간으로 사용자에게 전달
- 정확한 정보 제공으로 사용자 경험(UX)을 강화

# 상품등록



### 상품 판매 등록

제목  
사용감 거의 없는 어쿠스틱 기타

카테고리  
기타

판매가  
50.000원

거래방식  
직거래 (선택)

배송비  
2,500원

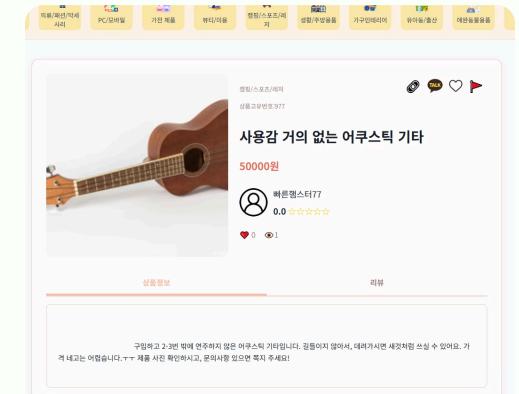
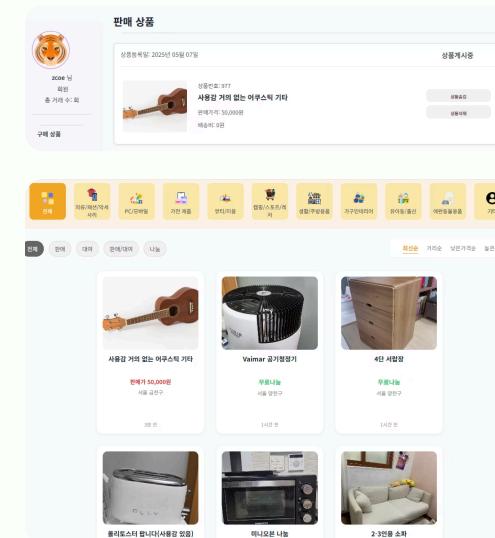
상품 상태  
사용감 처음

상품 설명  
언락상하고 2~3번 사용한 어쿠스틱 기타입니다. 사용감 거의 없고 악주에 문제 없어요. 가격 대고는 어울습니다. 사진 확인하시고 물의사항 있으면 물어보세요!

상품 이미지 (최대 5장)

필수 안내에 동의합니다

[등록하기](#)



## 상품등록 프로세스 흐름

- 메인 페이지(/main) 상단의 '상품등록' 클릭 → 'productRegister.jsp'에서 상품등록 유형 선택
- 상품등록페이지 호출됨 → 상품게시글 제목, 내용, 가격, 배송비, 상품상태 등 상세사항 기입하고 '등록하기' 버튼 클릭
- 등록 완료시 서버로 결과 전송 → 성공 시 서버에서 판매상품(product 테이블)에 저장



# 상품등록-트러블 슈팅 및 추후 보완점

## 트러블 슈팅

### ▼ 1.이미지 한번에 2개이상 등록이 불가.

#### 원인

Cos라이브러리를 활용한 이미지 등록시에는 한번에 하나의 이미지만 등록이 가능하다.

#### 해결

input을 각각 다른 name의 (img1~img5)로 등록

### ▼ 2.대여 상품에 대한 시작/종료일이 없음

#### 원인

거래유형이 '대여'임에도 대여 시작/종료일을 입력하지 않을 시에 서버에 해당 필드에 유효성 검사를 하지않아 null로 입력

#### 해결

거래유형이 '대여'일 때 <input name="startDate">,<input name="endDate">를 필수입력으로 jsp에서 처리 및 서버단에서 유효성 검사를 추가.

### ▼ 3.배송지 자동입력

#### 원인

기존에는 마이페이지에 등록된 배송지 정보를 기반으로 거래 위치를 가져왔기 때문에, 배송지가 없는 경우 사용자가 직접 마이페이지로 이동해 정보를 등록해야 하는 번거로움이 발생함

#### 해결

**카카오맵 API**를 활용하여 사용자의 현재 위치의 위도와 경도를 자동으로 가져오고, 이를 통해 별도 입력 없이 거래 위치가 자동 지정되도록 개선함.

## 추후 보완점

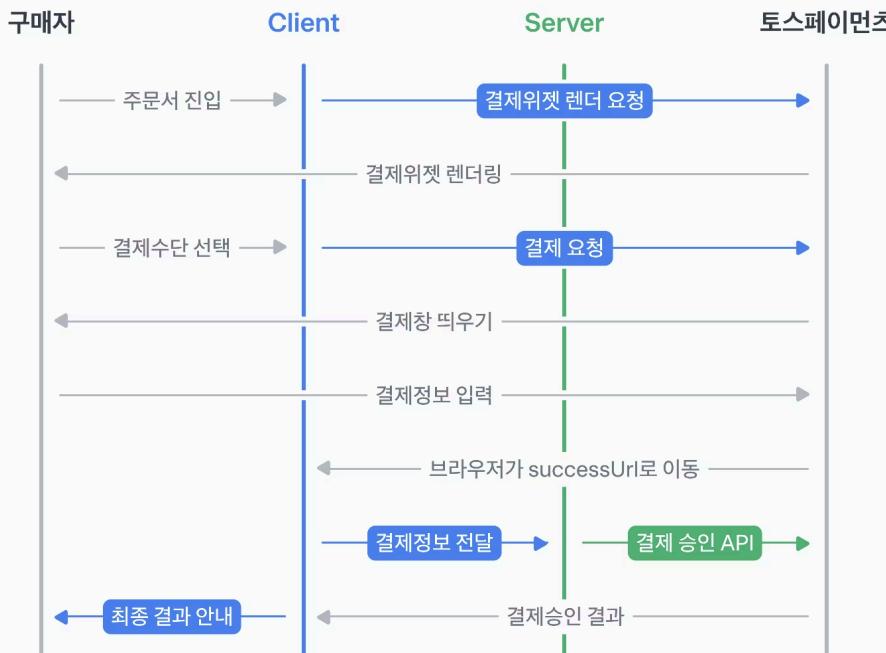
### ▼ 1.이미지 등록시에 한번에 여러개 등록

- 현재는 Cos라이브러리를 통해 이미지 등록을 하여 한번에 여러개의 이미지를 등록하는 것이 불가능함.
- Apache Commons FileUpload을 활용하여 Servlet에서 이미지 파일을 읽어 오게되면 한번에 여러개의 이미지를 추가할 수 있음.

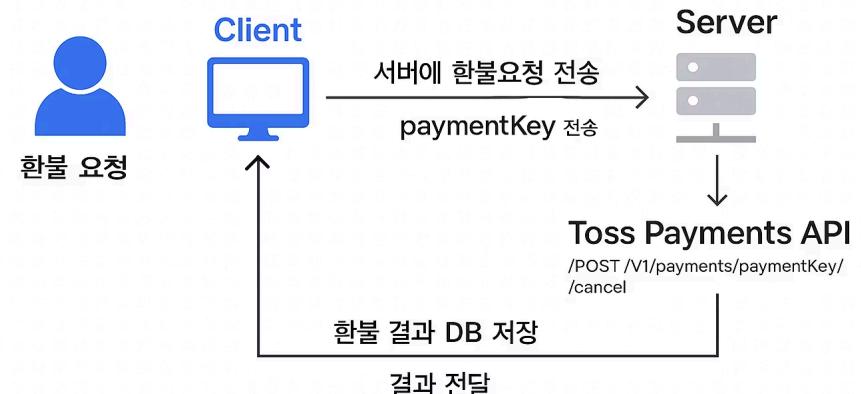
### ▼ 2.등록시 알림처리

- 현재는 등록시 상품상세로 넘어가기만 함
- 알림API를 활용하여 등록시에 등록한 사람에게 알림이 가능 추가

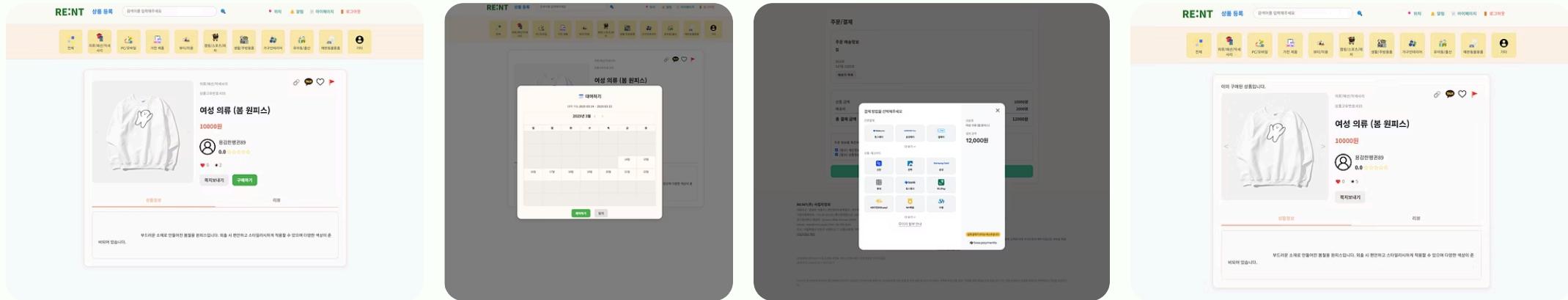
# 결제



# 환불



# 결제



## 결제 프로세스 흐름

- 상품상세페이지 (productDetail.jsp) → 사용자가 '구매하기' 버튼 클릭
- 결제 페이지 (orderSell.jsp) → 상품 금액과 배송비 확인 후 '결제하기' 버튼 클릭
- Toss Payments 결제창 호출 → payment.requestPayment() 함수를 통해 비동기 결제 진행
- 결제 완료 시 서버로 결과 전송 → 성공 시 서버에서 주문 정보(order 테이블)에 저장

## orderSell.jsp

```
async function requestPayment() {
    // 필수 체크 (배송지, 약관)
    if (!배송지 || !약관체크완료) {
        alert("필수 항목을 확인해주세요.");
        return;
    }

    // 결제 요청 (Toss Payments)
    await payment.requestPayment({
        method: "CARD",
        amount: { currency: "KRW", value: price },
        orderId: orderId,
        orderName: productTitle,
        successUrl: window.location.origin + "/success?deliveryAddr=...",
        failUrl: window.location.origin + "/fail",
        customerName: memberName,
        customerMobilePhone: memberPhone
    });
}
```

## orderSuccess.java

```
@WebServlet("/success")
public class OrderSuccess extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse res) throws
    ServletException, IOException {
        // 결제 성공 후 Toss로부터 받은 정보 수신
        String paymentKey = req.getParameter("paymentKey");
        String orderId = req.getParameter("orderId");
        // ... 기타 파라미터 저장
        //confirm 서블릿으로 포워딩하여 결제 검증 수행
        req.getRequestDispatcher("/confirm").forward(req, res);
    }
}
```

## paymentConfrim.java

```
@WebServlet("/confirm")
public class PaymentConfirm extends HttpServlet {
    protected void doPost(HttpServletRequest req, HttpServletResponse res) throws
    ServletException, IOException {
        // 결제 검증 요청 (paymentKey + orderId + amount)
        String jsonBody = "{ ... }"; // 결제 정보 JSON
        HttpURLConnection conn = (HttpURLConnection) new
        URL("https://api.tosspayments.com/v1/payments/confirm")
        .openConnection();
        // 인증 및 요청 세팅 생략...

        int status = conn.getResponseCode();

        if (status == 200) {
            // 결제 승인 성공 시 DB에 주문 정보 저장
            Order order = new Order(...);
            orderService.insertOrder(order);

            // 판매자에게 알림 전송
            Alarm alarm = new Alarm(...);
            fcmService.sendAlarm(alarm);

            // 상세 페이지로 이동
            res.sendRedirect("productDetail?no=" + productNo + "&paid=true");
        } else {
            // 결제 실패 처리
            req.getRequestDispatcher("/payment/fail.jsp").forward(req, res);
        }
    }
}
```

# 결제-트러블 슈팅 및 추후 보완점

## 트러블 슈팅

### ▼ 1.결제 완료 후 화면이 오래 멈춤

#### 원인

successUrl에서 바로 confirm서블릿으로 연결할 경우, 네트워크 지연으로 인해 화면 응답이 늦어짐

#### 해결

중간에 orderSuccess 서블릿을 두어 사용자에게 먼저 결제 성공 안내를 보여주고, 내부적으로 confirm으로 포워딩하여 검증 처리

### ▼ 2. 주문 날짜가 null 이 되거나 상품의 기본 대여 가능 날짜로 처리됨

#### 원인

사용자가 선택한 대여 날짜가 여러 JSP와 서블릿을 거치며 paymentConfirm 서블릿까지 제대로 전달되지 않음

#### 해결

날짜를 세션에 저장해 전달하고, paymentConfirm에서 사용 후 세션에서 제거하여 누락 없이 안정적으로 처리

### ▼ 3.결제 검증 실패(403,404 응답)

#### 원인

paymentKey, orderId 등 Toss Payments에서 요구하는 필수 값이 누락되거나 헤더 설정이 잘못됨

#### 해결

Secret Key를 Base64로 인코딩하여 인증 정보를 구성하고, Content-Type, Authorization 등의 헤더를 정확하게 설정

## 추후 보완점

### ▼ 1.결제 결과에 대한 비동기 처리 도입

- 현재는 결제 검증 및 DB 저장을 동기적으로 처리중, 네트워크나 외부 API지연시에 UX가 저하될 수 있음
- Toss에서 제공하는 webhook기능을 도입하여 서버에서 비동기적으로 결제결과를 수신 및 처리

### ▼ 2.결제 예외 상황에 대한 상세 로깅 강화

- 현재는 검증 실패 시 단순 실패 페이지로 이동
- 실패사유(응답코드, 메시지)를 서버 로그에 명확히 기록하고, 관리자용 알람 또는 대시보드 제공



# 환불

**빌린 상품**

zoe 님 회원 총 거래 수: 2회

구매 상품 판매 상품 템플 상품 목지갑 나의 리뷰 나의 신고목록 광고 관리 개인정보 수정 배송지 관리

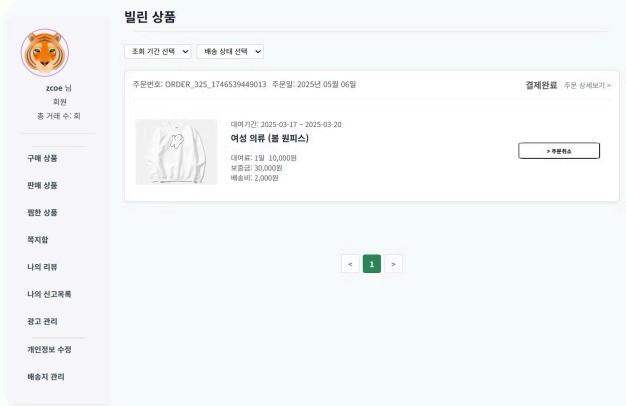
조회 기간 선택 배송 상태 선택

주문번호: ORDER\_325\_1746539449013 주문일: 2025년 05월 06일

결제완료 주문 상세보기 >

대여기간: 2025-03-17 ~ 2025-03-20  
여성 의류 (봄 원피스)  
대여료: 1일 10,000원  
보증금: 30,000원  
배송비: 2,000원

< 1 >



**빌린 상품**

zoe 님 회원 총 거래 수: 2회

구매 상품 판매 상품 템플 상품 목지갑 나의 리뷰 나의 신고목록 광고 관리 개인정보 수정 배송지 관리

조회 기간 선택 배송 상태 선택

주문번호: ORDER\_325\_1746539449013 주문일: 2025년 05월 06일

결제완료 주문 상세보기 >

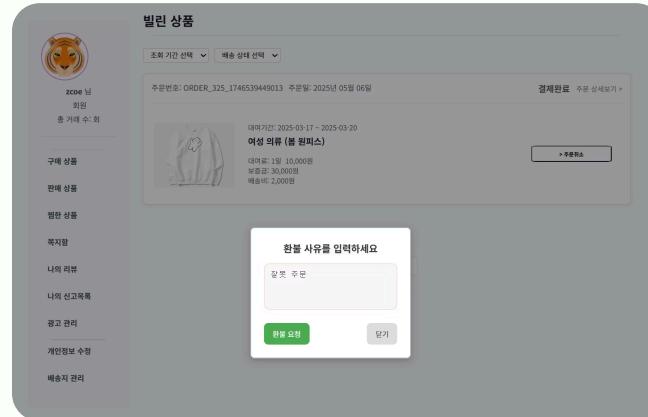
대여기간: 2025-03-17 ~ 2025-03-20  
여성 의류 (봄 원피스)  
대여료: 1일 10,000원  
보증금: 30,000원  
배송비: 2,000원

▶ 투명창

**환불 상유를 입력하세요**

잘못 주문

환불 요청 닫기



**빌린 상품**

zoe 님 회원 총 거래 수: 2회

구매 상품 판매 상품 템플 상품 목지갑 나의 리뷰 나의 신고목록 광고 관리 개인정보 수정 배송지 관리

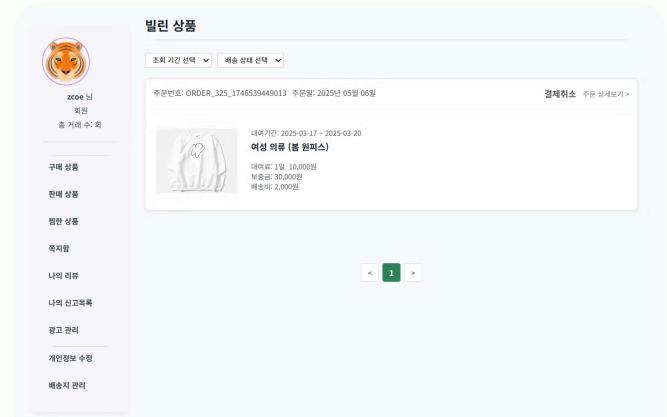
조회 기간 선택 배송 상태 선택

주문번호: ORDER\_325\_1746539449013 주문일: 2025년 05월 06일

결제취소 주문 상세보기 >

대여기간: 2025-03-17 ~ 2025-03-20  
여성 의류 (봄 원피스)  
대여료: 1일 10,000원  
보증금: 30,000원  
배송비: 2,000원

< 1 >



## 환불 프로세스 흐름

- マイページ → マイページ에서 결제완료 상태시에 결제취소 클릭
- 결제취소 모달에 모달사유 메세지 전송
- 결제완료 시 결제취소상태로 변환

## myRent.jsp

```
document.getElementById('cancelConfirmBtn').addEventListener('click', () => {
  const paymentKey = document.getElementById('cancelPaymentKey').value;
  const orderNo = document.getElementById('cancelOrderNo').value;
  const cancelReason = document.getElementById('cancelReason').value.trim();

  if (!cancelReason) {
    alert("사유를 입력해주세요.");
    return;
  }

  // 환불 요청 서버 전송
  fetch('/refund', {
    method: 'POST',
    headers: { 'Content-Type': 'application/x-www-form-urlencoded' },
    body: new URLSearchParams({
      paymentKey,
      cancelReason,
      orderNo
    })
  })
  .then(res => {
    if (!res.ok) throw new Error("환불 실패");
    return res.json();
  })
  .then(data => {
    alert("환불이 완료되었습니다.");
    window.location.href = '/myOrder';
  })
  .catch(err => {
    console.error("환불 오류:", err);
    alert("환불 실패. 관리자에게 문의하세요.");
  });
});
```

## refund.java

```
@WebServlet("/refund")
public class Refund extends HttpServlet {

  private static final String SECRET_KEY = "test_sk..."; // Toss Secret Key

  protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");

    // 요청 파라미터 수신
    // String paymentKey = ...
    // String cancelReason = ...
    // int orderNo = ...

    // Toss 환불 API 요청 구성
    HttpURLConnection conn = (HttpURLConnection) new
    URL("https://api.tosspayments.com/v1/payments/" + paymentKey +
    "/cancel").openConnection();
    conn.setRequestMethod("POST");
    conn.setRequestProperty("Authorization", "Basic " +
    Base64.getEncoder().encodeToString((SECRET_KEY + ":").getBytes()));
    conn.setRequestProperty("Content-Type", "application/json");
    conn.setDoOutput(true);

    // JSON 바디 전송
    // String jsonBody = ...
    // conn.getOutputStream().write(...);

    // 응답 처리
    int statusCode = conn.getResponseCode();
    BufferedReader br = new BufferedReader(new InputStreamReader(
    (statusCode < 400) ? conn.getInputStream() : conn.getErrorStream(), "utf-8"));

    // 응답 본문 읽기
    // StringBuilder result = ...

    if (statusCode == 200) {
      // 환불 성공 시 주문 상태 DB 업데이트
      new OrderServiceImpl().updateOrderStatus(orderNo, "결제취소");
      response.setStatus(200);
      response.setContentType("application/json;charset=UTF-8");
      response.getWriter().write(/* 응답 JSON */);
    } else {
      // 환불 실패 응답 반환
      response.setStatus(statusCode);
      response.setContentType("application/json;charset=UTF-8");
      response.getWriter().write(/* 에러 메시지 */);
    }
  }
}
```



# 환불-트러블 슈팅 및 추후 보완점

## 트러블 슈팅

### ▼ 1.환불 사유 없이 요청됨

#### 원인

사용자가 cancelReason 입력 없이 버튼 클릭 가능

#### 해결

프론트에서 cancelReason이 비어있을 경우 alert를 띄우고, fetch 중단

### ▼ 2.서버에서 paymentKey가 null로 넘어옴

#### 원인

공백이 포함된 상태로 전달되거나 name속성 누락

#### 해결

paymentKey.trim()으로 처리하고, 필드 누락여부를 if조건으로 검증

## 추후 보완점

### ▼ 1.환불 처리에 대한 사용자 알림 강화

- 환불 성공 시 alert창만 출력
- 환불 완료 후 FCM알림 또는 이메일 전송으로 사용자에게 환불 처리 이력을 실시간 안내

### ▼ 2.환불 요청 이력 관리 기능 추가

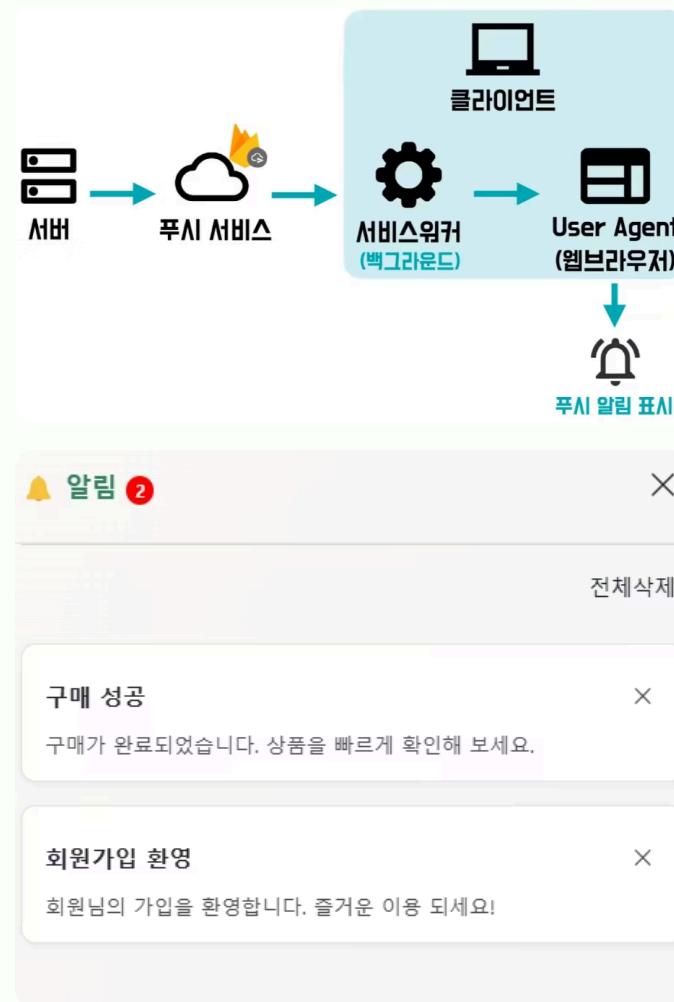
- 환불 요청과 결과는 DB에 기록되지 않음
- 환불 이력 테이블을 추가하여 사용자와 관리자가 환불 내역을 확인 가능하게 구성

### ▼ 3.환불 금액 조정 기능 고도화

- 전체 환불만 가능함
- 부분 환불, 배송비 제외 환불 등 상황별로 환불 로직 분기 및 UI개선

# API를 활용한 사용자 편의 기능

FCM을 활용한 포그라운드, 백그라운드 알림



카카오맵을 활용한 위치기반 서비스



소셜로그인 및 배송지 관리

**로그인**

로그인 및 회원가입을 시작합니다.

카카오 로그인

N 로그인

일반 회원으로 이용하기 ▼

Daum Postcode Service - Chrome

park

010-2321-2423

배송지 입력

+ 배송지 추가

집

07761

서울 강서구 가로공원로 172

영빌라 201호

**tip**  
아래와 같은 조합으로 검색을 하시면 더욱 정확한 결과가 검색됩니다.  
도로명 + 건물번호  
예) 판교역로 166, 제주 첨단로 242  
지역명(동/리) + 번지  
예) 박현동 532, 제주 영평동 2181  
지역명(동/리) + 건물명(아파트명)  
예) 분당 주공, 연수동 주공3차

설명: 다음은 api를 활용한 사용자 편의기능에 대해 소개 해드리겠습니다.

먼저 fcm을 활용한 알림입니다. 특정 시점에 서버에서 알림을 자동 또는 수동으로 전송하면 fcm에서 클라이언트에게 알림을 보내게 됩니다.

클라이언트가 로그인이 되어있지 않은 경우에도 서비스워커로 인해 백그라운드로 받을수 있고, 로그인해 있다면 실시간으로 알림을 받을 수 있습니다.

저희 서비스에서는 회원가입, 구매, 쪽지, 대여 물품 반납에 대해 알림을 전송 중입니다.

두번째는 카카오맵을 활용한 위치기반 서비스 입니다.

사용자의 위치에 따라 카카오맵으로부터 주소와 위치를 받아올수 있으며, 주소와 위치간의 변환도 가능합니다.

저희 서비스에서는, 동네 상품 추천 및 거리가 가까운 순으로 상품을 정렬하는 기능을 제공하고 있습니다.

마지막으로, 카카오 및 네이버 소셜로그인 서비스와 다음 우편번호 찾기를 통한 배송지 관리 기능을 제공하고 있습니다.

# API - 트러블 슈팅 추후 보완점

## 트러블 슈팅

### ▼ 소셜 로그인(Kakao) 시 최초 로그인 알림 전송 로직 오류

- 최초 로그인 여부를 판단하는 로직에서 오류 발생
- `isFirstLogin(code)` 같은 메서드에서 Authorization Code를 두 번 사용함

문제점	설명
<b>Authorization Code(<code>code</code>)를 재사용</b>	<code>code</code> 는 Kakao가 주는 1회용 인증코드. <b>AccessToken</b> 발급에만 1회 사용 가능
<code>isFirstLogin(code)</code> 와 <code>KakaoLogin(code)</code> 둘 다 <code>code</code> 를 소비	내부적으로 <code>getKakaoToken(code)</code> 이 두 번 호출되며 예외 발생
<code>accessToken</code> 으로도 최초 로그인 여부 판단 시도	<code>accessToken</code> 은 사용자 정보는 알 수 있지만, "우리 서비스에 처음인지"는 Kakao가 알 수 없음

## 해결방안

- Authorization Code는 오직 한 번만 사용
- `accessToken`으로 사용자 정보 조회 후, DB에서 최초 로그인 여부 판단
- 최초 로그인 시 알림 전송 플래그를 세션에 저장
- main.jsp 진입 시 `firstLogin == true`이면 FCM 알림 전송 후 세션에서 제거

## 추후 보완점

### ▼ 추후 스프링, 리액트로 개발시 JWT기반 인증방식을 사용

- 세션 상태 관리 없이 REST API로 설계
- 최초 로그인 여부는 JWT응답 payload에 포함
- 클라이언트에서 처리후 알림 전송

### ▼ Redis와 스프링으로 fcm 성능 및 기능 확대

- Redis의 메세지큐 방식으로 성능 향상
- 스프링의 스케줄러를 이용하여 반납기간에 대하여 백그라운드에서 관리 및 알림 전송

# 마이페이지

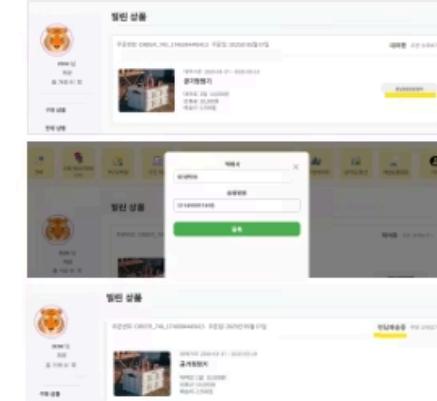
구매 /판매	구매자 (행동)	/productList 상품을 구매할 수 있음	결제함	주문취소 - 구매금액 전체 환불해야 - 상품은 (판매)게시중 상태로 가야	(택배사 처리)			구매확정 버튼 누름 ** 주문취소/환불 처리 가능여부 정해야				
	구매자 (출력회면)			/myOrder btn) 주문취소 (* 배송 시작 전까지는 주문취소 가능. 배송 시작 후 주문취소 불가)				/myOrder btn) 구매확정	/myOrder btn) 리뷰쓰러가기			
	Order/ orderStatus			결제완료	배송중		(배송완료)		거래완료			
	판매자 (출력회면)	/mySell btn) 상품등기기 btn) 상품삭제		/mySell btn) 송장번호입력								
	판매자(행동)	상품게시중			송장번호 입력함				* 정산 테이블에서 확인할 수 있어야. * 관리자에서 조회 가능하게 정산내역 넘어가야.			
빌리기 /빌려주기	빌린사람 (행동)	/productList 상품을 대여할 수 있음	결제함	주문취소 - 구매금액 전체 환불해야 - 상품은 (판매)게시중 상태로 가야	(택배사 처리)			빌리기시작 버튼 누름	*** 대여 종료는 대여시작일 기준으로 자동으로 정해짐	송장번호 입력함		
	빌린사람 (출력회면)			/myRent btn) 주문취소 (* 배송 시작 전까지는 주문취소 가능. 배송 시작 후 주문취소 불가)				/myRent btn) 빌리기시작	/myRent btn) 반납송장번호입력			/myRent btn) 리뷰쓰러가기
	Order/ orderStatus			결제완료	배송중		(배송완료)	대여중	(대여종료)		반납배송중	거래완료
	빌려준사람 (출력회면)	/myLend btn) 상품등기기 btn) 상품삭제		/myLend btn) 송장번호입력						/myLend btn) 상품회수완료		
	빌려준사람(행동)	상품게시중			송장번호 입력함					상품회수 완료 버튼 누름 * 정산 테이블에서 확인필요 * 관리자에서 조회 가능하게 정산내역 넘어가야		

구매←→판매, 빌리기←→빌려주기 분기 처리

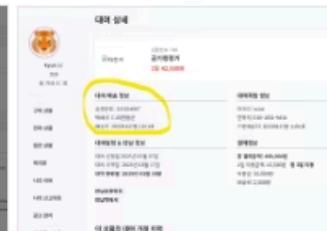
- '구매'는 주문상태(orderStatus) 4개 프로세스, '대여'는 6개 프로세스로 정의
- 각 과정에서 구매자(대여자) 및 판매자(대여판매자)에게 다음 동작을 유도하는 버튼 출력, 구현해 구매/대여 프로세스를 완성하도록 처리
- 송장번호 입력 이후에는, 해당 데이터를 각 상세페이지에서 확인할 수 있음



# '대여' 시 구매자-판매자의 마이페이지 작동 과정



빌리기 /빌려주기	빌려사람 (출력화면)		/myRent btn) 주문취소 (* 배송 시작 전까지는 주문취소 가능. 배송 시작 후 주문취소 불가)			/myRent btn) 빌리기시작		/myRent btn) 반납송장번호입력		/myRent btn) 리뷰쓰러가기
	Order/ orderStatus	결제완료	배송중	(배송완료)	대여중	(대여종료)	반납배송중	거래완료		
	빌려준사람 (출력화면)	/myLend btn) 상품승인기 btn) 상품삭제		/myLend btn) 송장번호입력				/myLend btn) 상품회수완료		





# 마이페이지 - 트러블 슈팅 및 추후 보완점

## 트러블 슈팅

- ▼ 1. 초기 버튼 구현시 <button type="button">을 제대로 명시하지 않아 JS 코드가 작동하지 않은 문제

### 원인

사용자가 cancelReason 입력 없이 버튼 클릭 가능

### 해결

프론트에서 cancelReason이 비어있을 경우 alert를 띄우고, fetch 중단

- ▼ 2. 다른 버전의 자바스크립트 코드(ex. 바닐라JS, JSTL)를 한 파일에서 사용하는 경우 제대로 작동하지 않은 문제: 한 버전의 코드로 통일 수정해 해결

### 원인

공백이 포함된 상태로 전달되거나 name속성 누락

### 해결

paymentKey.trim()으로 처리하고, 필드 누락여부를 if조건으로 검증

## 추후 보완점

- ▼ 1. 택배배송의 경우 스마트택배 API를 이용하면 자동처리되는 부분이 있어 편리성을 높일 수 있을 것 같다.
  - 환불 성공 시 alert창만 출력
  - 환불 완료 후 FCM알림 또는 이메일 전송으로 사용자에게 환불 처리 이력을 실시간 안내

- ▼ 2. 여러 테이블의 정보를 출력하는 경우가 많았는데, 테이블 구조를 개선하거나 DTO를 효율적으로 이용해보면 좋을 것 같다.
  - 환불 요청과 결과는 DB에 기록되지 않음
  - 환불 이력 테이블을 추가하여 사용자와 관리자가 환불 내역을 확인 가능하게 구성

## 대여 상품

상품등록일: 2024년 09월 20일

반납배송중 주문 상세보기 >



상품번호 : 809  
게이밍 노트북  
대여료: 1일 10,000원  
보증금: 20,000원  
배송비: 2,000원

상품반납확인

E:N:T 회원관리 카테고리관리 상품관리 정산관리 고객지원 알림관리

HOME > 정산관리 > 주문정산

정산 상태: 전체 ▾ 판매/대여 구분: 전체 ▾ 정산완료 시작: 연도-월-일 -- -- -- □ 정산완료 종료: 연도-월-일 -- -- -- □ 검색

검색된 총 정산 건수: 1건

총 수수료 금액:													
정산번호	주문번호	거래구분	회원번호	상품번호	상품명	결제일	율증가	배송비	보증금	수수료율(%)	수수료금액	최종정산금액	정산완료시간
16	4	대여	12	809	게이밍 노트북	2025-05-07	32,000 원	2,000 원	20,000 원	5 %	1,600 원	52,400 원	2025-05-07 1...

E:N:T 회원관리 카테고리관리 상품관리 정산관리 고객지원 알림관리

HOME > 정산관리 > 주문정산

정산 상태: 전체 ▾ 판매/대여 구분: 전체 ▾ 정산완료 시작: 연도-월-일 -- -- -- □ 정산완료 종료: 연도-월-일 -- -- -- □ 검색

검색된 총 정산 건수: 1건

총 수수료 금액:													
정산번호	주문번호	거래구분	회원번호	상품번호	상품명	결제일	율증가	배송비	보증금	수수료율(%)	수수료금액	최종정산금액	정산완료시간
16	4	대여	12	809	게이밍 노트북	2025-05-07	32,000 원	2,000 원	20,000 원	5 %	1,600 원	52,400 원	2025-05-07 1... 2025-05-07 17:0

## 관리자 페이지에서 CRUD + 관리 자동화 처리 방식

- CRUD 중 읽기(Read) + 업데이트(Update)에 집중된 기능
- 예: 정산 완료 상태 변경, 완료 시간 기록, orderCount 증가
- 자동화 포인트:**
  - 판매자 “정산 완료” 또는 “상품반납확인” 버튼 클릭 → 관리자 수동 작업 최소화 → DB 상태/시간 자동 업데이트



# 정산

## Servlet — DoSettlement.java

```
@WebServlet("/doSettlement")
...(중략)
boolean success = settlementService.processSettlement(settlementNo);
String completedAt = settlementService.getCompletedAt(settlementNo);
...(중략)
response.getWriter().write("{\"success\": true, \"completedAt\": \"" + completedAt + "\"}");
```

## MyBatis Mapper — settlement.xml

```
<update id="updateSettlementStatus" parameterType="int">
    UPDATE settlement
    SET feeStatus = 'COMPLETE'
    WHERE settlementNo = #{settlementNo}
</update>

<update id="updateSettlementCompletedAt" parameterType="int">
    UPDATE settlement
    SET settlementCompletedAt = NOW()
    WHERE settlementNo = #{settlementNo}
</update>

<select id="selectCompletedAt" parameterType="int" resultType="string">
    SELECT DATE_FORMAT(settlementCompletedAt, '%Y-%m-%d %H:%i:%s')
    FROM settlement
    WHERE settlementNo = #{settlementNo}
</select>
```

## 각 기능의 JSP → 서블릿 → 서비스/DAO → 맵퍼 흐름

- JSP에서 사용자가 버튼을 클릭하면 →
- 서블릿(DoSettlement)이 POST 요청을 받고 →
- 서비스(SettlementServiceImpl)가 비즈니스 로직 처리 →
- DAO(SettlementDAOImpl)가 MyBatis 맵퍼를 통해 DB 업데이트 실행

# 정산 - 트러블 슈팅 및 추후 보완점

## Service – SettlementServiceImpl

```
boolean processSettlement(int settlementNo) throws Exception;
```

(수정 전)

```
// 1. 정산 상태 COMPLETE 처리  
settlementDAO.updateSettlementStatus(settlementNo);  
  
// 2. 정산 완료 시간 기록  
settlementDAO.updateSettlementCompletedAt(settlementNo);  
  
// 3. 회원번호 조회 및 등급 승급 시도  
int memberNo = settlementDAO.selectMemberNoBySettlementNo(settlementNo);  
// orderCount +1 처리  
settlementDAO.updateOrderCountPlusOne(memberNo);  
// 등급 승급 처리  
upgradeMemberGradeIfNeeded(memberNo);
```

(수정 후)

```
// 1. 정산 상태 COMPLETE 처리  
settlementDAO.updateSettlementStatus(settlementNo);  
  
// 2. 정산 완료 시간 기록  
settlementDAO.updateSettlementCompletedAt(settlementNo);  
  
// 3. 주문/상품 정보 가져오기  
Settlement settlementInfo = settlementDAO.selectSettlementInfo(settlementNo);  
Order order = settlementDAO.selectOrderInfo(settlementInfo.getOrderNo());  
Product product = settlementDAO.selectProductInfo(order.getProductNo());  
  
int buyerMemberNo = order.getMemberNo(); // 구매자  
int sellerMemberNo = product.getMemberNo(); // 판매자  
  
// 4. 각 회원의 orderCount +1  
settlementDAO.updateOrderCountPlusOne(buyerMemberNo);  
settlementDAO.updateOrderCountPlusOne(sellerMemberNo);  
  
// 5. 각 회원 등급 승급 체크  
upgradeMemberGradeIfNeeded(buyerMemberNo);  
upgradeMemberGradeIfNeeded(sellerMemberNo);
```

## 트러블슈팅

- 기존:

판매자(memberNo)만 조회하여 orderCount +1 및 등급 승급 처리

- 문제:

실제로는 구매자·판매자 모두 주문 완료 대상

→ 한쪽만 처리 시 데이터 불일치 발생

- 개선 후:

settlement → order → product 계층 조회로

→ 구매자(buyerMemberNo), 판매자(sellerMemberNo) 모두 추출

→ 각 회원별 orderCount +1, 등급 승급 로직 각각 실행

## 추후 보완 점

- 현재 테스트 환경:

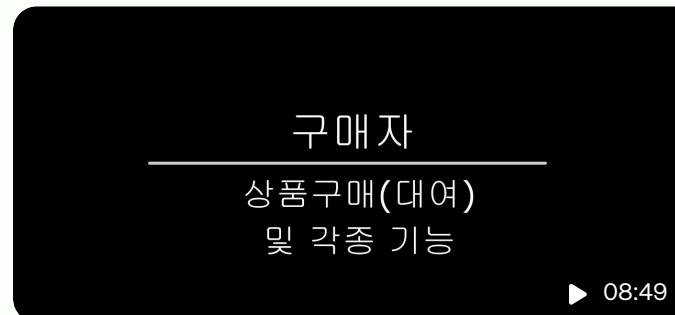
결제 API 연동 불가 → 코드에서 강제로 true 반환

- 개선 필요:

실제 결제 승인 여부를 API로 검증한 뒤 →

결제 완료 상태에서만 `settlementService.processSettlement()` 실행하도록  
수정

## 시연영상



 YouTube

RENT시연영상

KOSTA 세미프로젝트 : RE:NT



## 4. 프로젝트 성과

# ☰ 프로젝트 과정에서 느낀점 & 배운점

### 팀워크

- 명확한 역할 분담과 소통의 중요성

### 기술 습득

- 실무 API 활용 경험  
(Toss, 카카오 지도, 소셜 로그인 등)
- 백엔드-프론트엔드 연동 문제 해결 과정에서 통합적인 사고력 향상
- MyBatis 기반의 효율적인 DB 연동 설계 경험



### 시간 관리

- 점진적 개발과 일정 관리의 중요성
- 계획 수립, 우선순위 설정, 마감 관리가 프로젝트 성공에 큰 영향을 미친다는 점 깨달음

### 문제 해결

- 기술적 장애 극복 과정에서의 성장
- 문제가 발생할 때마다 멈추지 않고 해결책을 찾아나가는 태도의 중요성

# moduleId 팀원별 역할 및 자체 평가

RE:NT 팀의 역할 분담과 자체 평가를 통해 프로젝트 진행 과정을 돌아보았습니다.

팀원	담당 업무	자체 평가
이아림	<ul style="list-style-type: none"><li>마이페이지: 구매상품/대여 상품, 판매상품/대여상품</li><li>상품 및 주문 생명주기 전체 관리</li><li>프로젝트 총괄 : 팀 내 일정 조율, 우선순위 결정, 이슈 관리, Git 브랜치 전략 등</li><li>발표</li></ul>	상품 구매 이후 구매자-판매자 간의 행동을 구조화하는 과정이 있었습니다. 서비스를 만들어가는 데 꼭 필요한 일로서, 이를 구조화하고 서비스로 옮겨보는 무척 좋은 기회였던 것 같습니다. 수업에서 배운 것들을 이해하며 조금 성장했다고 느끼는 반면, 기술적으로 한 단계 도약하지 못한 점, 개인 일정관리를 효율적으로 하지 못한 점은 아쉽게 남습니다. 다음 프로젝트에서는 시간관리로 속도를 높이고 기능 구현을 목표로 잡아 이를 달성해보고 싶습니다.
박기도	<ul style="list-style-type: none"><li>마이페이지 : 신고, 리뷰, 찜, 쪽지 목록 조회, 개인정보수정, 배송지관리</li><li>상품목록 : 메인 화면, 상품 목록(필터, 정렬) 조회</li><li>API: FCM알람, 카카오 맵, 카카오&amp;네이버 소셜로그인, 다음 우편번호</li><li>피그마 및 디자인 총괄</li><li>시연영상 제작</li></ul>	처음 프로젝트인 만큼 mvc패턴 연습과 백엔드와 프론트엔드간의 데이터 전송에 중점을 두고 프로젝트를 진행하였습니다. 처음엔 CRUD에 대해서도 막연한 두려움이 있었지만, 진행하다보니 익숙해져서 재밌었습니다. 아무래도 Servlet, Jsp기반이다 보니 수작업으로 해야하는 코드들이 많아서 번거로움이 있었는데 다음에 spring으로 넘어가면 더 효율적으로 빠르게 코드 짤 수 있을 것 같습니다! api도 여러개 해봐서 너무 재밌었고 유익한 프로젝트였습니다! 팀장님, 팀원분들도 끝까지 포기하지 않고 맡은 업무를 맡은 것에 대해 감사합니다!! 다음 프로젝트도 화이팅입니다!
추혜민	<ul style="list-style-type: none"><li>관리자페이지: 회원목록조회, 카테고리 등록, 신고/정산/대여지연 목록 조회, FAQ, 공지 등록</li><li>DB부총괄 (보조 테이블): 멤버등급, 카테고리, 신고, 정산 테이블 생성</li><li>PPT 제작 : 발표용 스크립트, 시각자료 구성</li></ul>	프로젝트 이전에는 데이터베이스 간 연관관계에 대한 이해가 부족해 FK 제약, 테이블 간 조인 및 상관관계를 구현하는 데 어려움이 있었습니다. 하지만 이번 프로젝트를 통해 이러한 부분에서 한층 성장할 수 있었습니다. 또한 팀 일정은 지켰지만 개인 작업 계획 작성과 실행 관리에서는 아쉬움이 남았습니다. 이를 계기로 앞으로는 개인 일정 계획을 더 명확히 세우고 실행력을 높여 팀 내에서 더욱 책임감 있게 기여하고자 합니다.
김한재	<ul style="list-style-type: none"><li>상품등록(판매, 대여, 판매/대여, 나눔)</li><li>상품상세페이지: 대여하기, 구매하기, 리뷰, 쪽지, 신고, 공유</li><li>API: 토큰을 통한 결제 및 환불 기능</li><li>DB총괄(메인 테이블): 물품, 사용자, 거래, 리뷰 등 메인 테이블 생성 및 테스트 시나리오용 데이터 준비, FK 제약</li></ul>	이전부터 결제기능을 구현해보는 경험을 갖는게 필요하다고 생각하고 있었는데 이번 프로젝트를 통해서 경험하게 되어서 한층 더 성장한 기분이 들었습니다. 팀 프로젝트는 팀원간에 소통이 매우 중요하다고 생각하는데, 적극적으로 이야기 해주시고 프로젝트에 진지하게 임해주시는 팀원들 덕분에 제 부족한 부분들을 보완하고, 한번더 둘이켜보는 시간을 갖게되어서 좋았습니다.

# QnA

## 감사합니다

