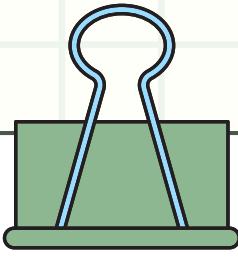
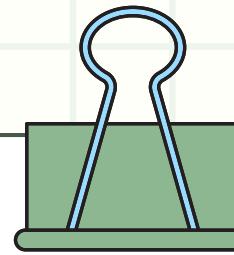


TEAM PROJECT



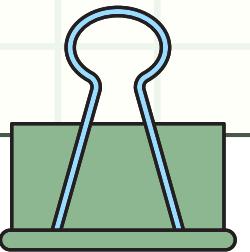
# 팀 프로젝트 쇼핑몰 - CLODI

KDT 개발자 과정



# 목차

- 01**      **프로젝트 개요**
- 02**      **팀 구성원 소개**
- 03**      **프로젝트 기획서**
- 04**      **UI 흐름도**
- 05**      **개발 환경**
- 06**      **프로젝트 진행 일정**
- 07**      **E-R 다이어그램**
- 08**      **주요 코드**
- 08**      **Q&A**



# 01 프로젝트 개요

## 프로젝트의 목적 및 목표

### 📌 프로젝트 목적

웹 서비스 개발의 전 과정을 실습하고 팀 협업을 통해 실무 개발 환경에 대한 이해를 높이는 데 목적이 있습니다.

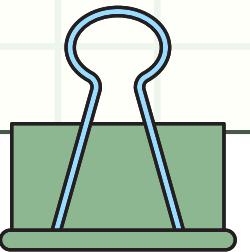
### 📌 프로젝트 목표

- 쇼핑몰 핵심 기능(로그인, 회원가입, 상품, 주문, 결제, 챗봇 등) 구현
- Spring Boot와 JSP 기반 웹 어플리케이션 개발
  - Git을 활용한 협업 및 형상관리 경험
- 보고서 및 산출물 정리 후 포트폴리오로 활용

## 프로젝트의 중요성

### 📌 프로젝트 중요성

실제 상용 웹 서비스와 유사한 구조를 가진 쇼핑몰 시스템을 통해, 웹 개발에 필요한 핵심 기술과 협업 프로세스를 실습 할 수 있는 중요한 프로젝트입니다



## 02 팀원 소개

김영민(0713)

- 이메일 인증 기반 회원가입, 로그인 기능 구현
- 아이디/비밀번호 찾기, 이메일 인증 기반 비밀번호 재설정 기능 개발
- 회원정보 수정(이메일, 비밀번호 등) 기능 구현
- 관리자 등록 FAQ 기반의 챗봇 기능 구현

이한민

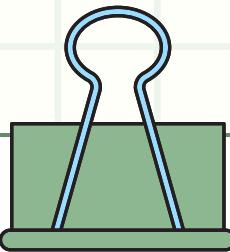
- 공지사항
- 결제(카카오API 활용)
- 관리자 대시보드(회원·입점 신청에 대한 관리)

지승택

- 상품조회 및 상품 후기
- 장바구니
- 이미지 업로드 및 관리(imagekit 연동)

김영민(1201)

- 상품 관리
- 판매자 인증
- 배송 및 매출 관리



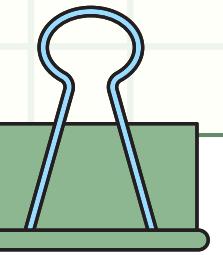
# 03 프로젝트 기획서

## 필요 기능 정리

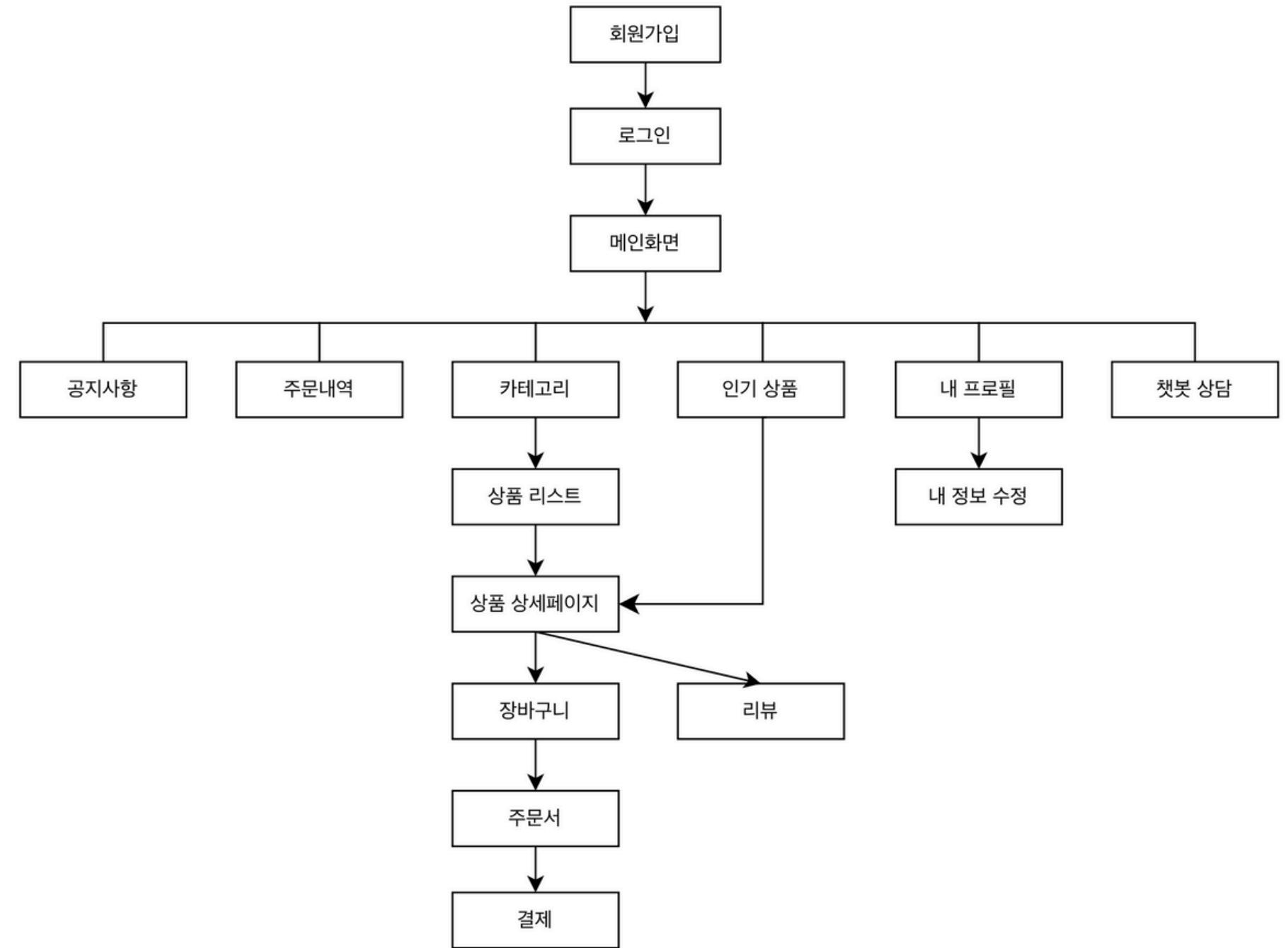
실제 프로젝트를 진행하면서 작업하게 될 구현물과는  
상이 할 수 있으나 전체적인 틀을 잡기 위한 기획서

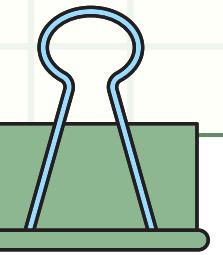
실제 프로젝트 구현 중 주의해야하는 점을 정리

프로젝트 명	쇼핑몰(클로디)
프로젝트 참여자	김영민(010713), 이한민, 지승택, 김영민(1201)
프로젝트 기간	2025.06.02 ~ 2025.06.30
프로젝트 목적	웹 서비스 개발의 전 과정을 실습하고 팀 협업을 통해 실무 개발 환경에 대한 이해를 높이는 데 목적이 있습니다.
구현기능	<b>로그인</b> - DB에 있는 아이디와 비밀번호를 비교 후 로그인 <b>회원가입</b> - 이메일 인증을 통한 회원가입 구현 <b>아이디 및 비밀번호 찾기</b> - 이메일 인증 후 DB에 들어있는 아이디, 비밀번호 추출 <b>마이페이지</b> - 로그인 정보를 불러 정보 확인 및 수정 <b>공지사항</b> - 관리자만 작성 및 수정, 삭제 <b>챗봇</b> - 사용자가 입력한 내용에서 키워드 추출 후 답변 <b>장바구니 및 결제</b> - 상품 선택 후 장바구니로 정보 이동 - 장바구니안에 있는 물품 결제로 데이터 이동 <b>리뷰</b> - 사용자가 구매했던 물품만 리뷰 작성 가능(수정, 삭제) <b>회원관리</b> - 관리자 계정으로 로그인 후 구매자 판매자 정보 관리 <b>챗봇 FAQ 관리</b> - 자주 묻는 키워드들을 등록 후 답변 등록 및 수정, 삭제 <b>입점 신청 관리</b> - 판매자가 입점 신청 후 관리자가 확인해야 입점 가능 <b>상품 관리</b> - 판매자가 등록한 상품 관리(등록, 수정, 삭제) <b>월별 매출 확인</b> - 구매자가 구매한 상품들을 전체, 카테고리별 매출 확인 <b>주문내역 관리</b> - 구매자가 구매한 상품 주문내역 관리 <b>배송 관리</b> - 주문된 상품 배송 상태 변경
설계 주의점	정기적인 테스트를 통해 각자 구현한 기능들이 충돌하지 않도록 예방하고 시스템에 영향을 주지 않는지 확인해야 한다. DB 구성 및 기능 구현할 때 충분한 논의가 필요하다.

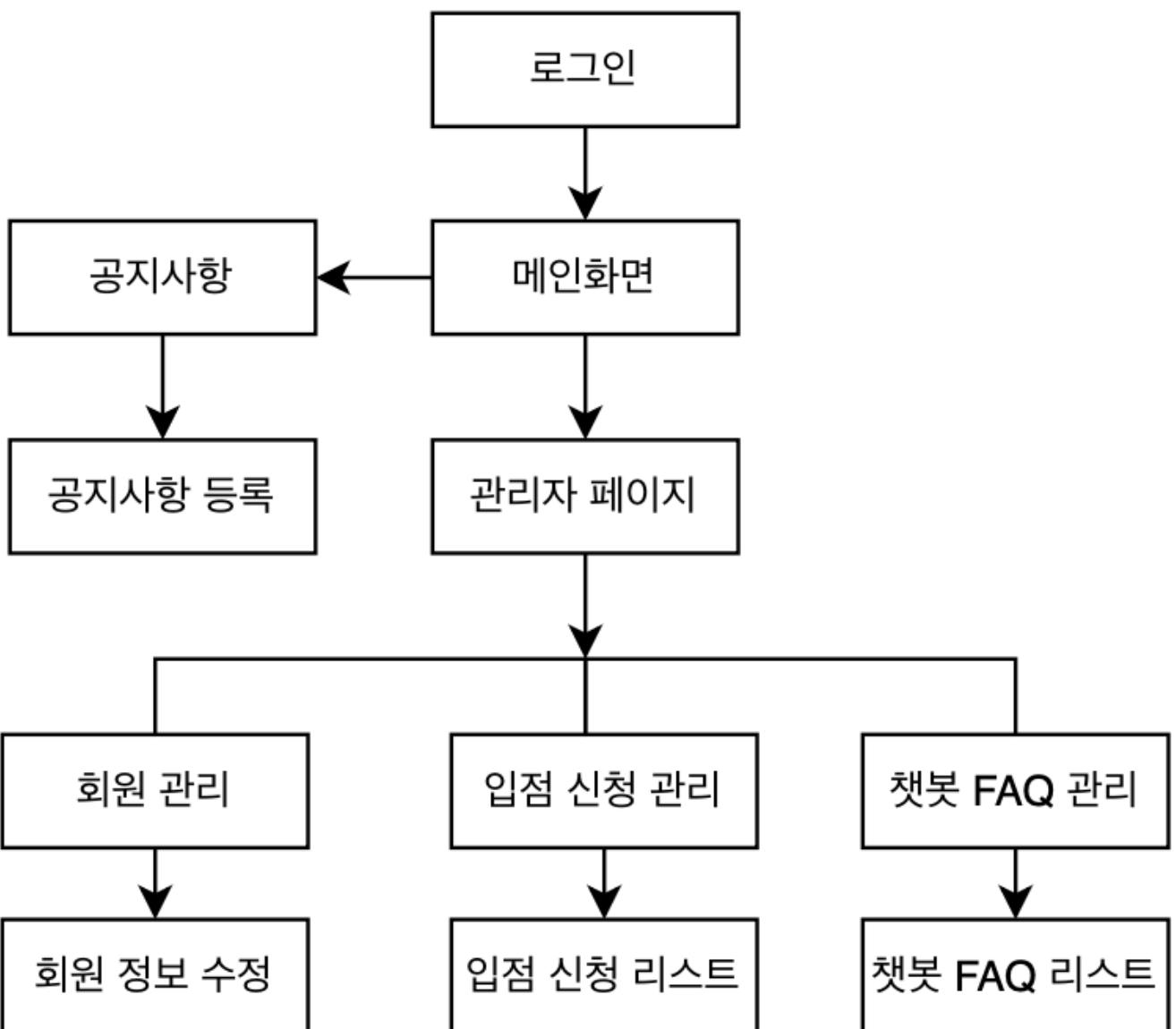


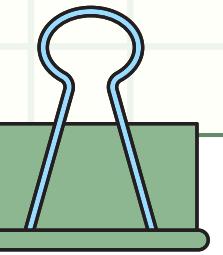
# 04 UI 흐름도 - 구매자



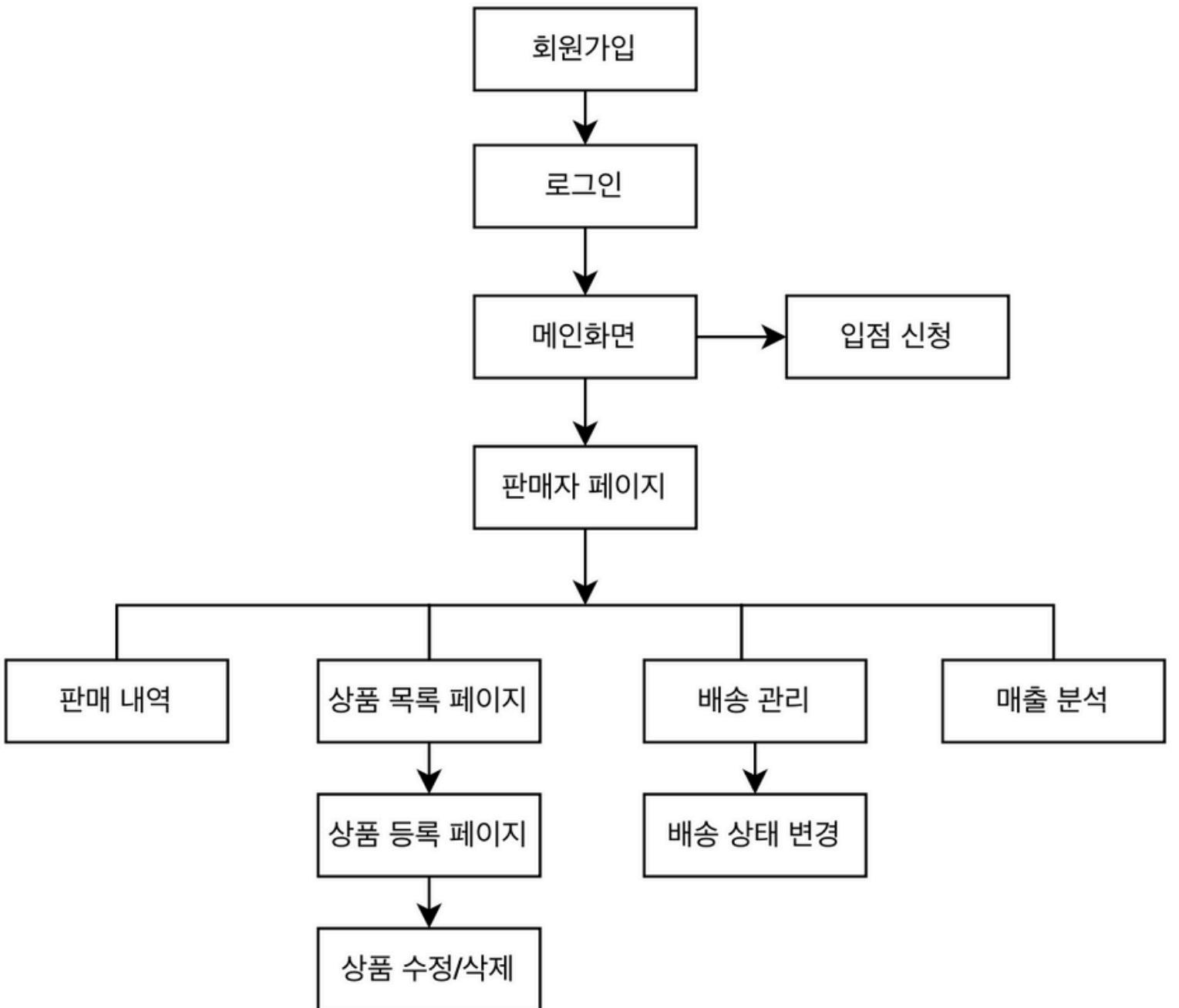


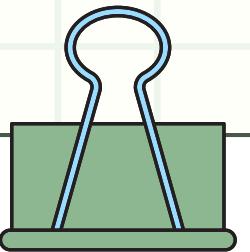
# 04 UI 흐름도 - 판매자





# 04 UI 흐름도 - 관리자





# 05 개발 환경

## 협업 툴



*imagekit.io*

## 제작기간

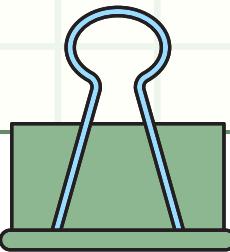
2025.06.02 ~ 2025.06.30

개발 언어 : Java, JSP, JPA

프레임 워크 : Spring boot Framework

DB : Oracle

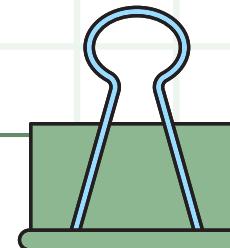
IDE: Spring Tool Suite 4 (STS4)

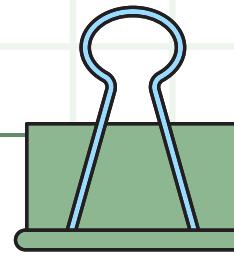


# 06 프로젝트 진행 일정

단계	기간	작업 내용
프로젝트 기획	6/2(월) ~ 6/4 (수)	요구사항 분석, 전체 일정 수립
설계	6/5(목) ~ 6/6(금)	ERD 작성, 기술 스택 선정
개발 - 1단계	6/9(월) ~ 6/13(금)	DB 구축, 회원관리 및 기본 기능 구현
개발 - 2단계	6/16(월) ~ 6/20(금)	주요 기능 개발 (주문, 결제, 챗봇 등)
테스트	6/23(월) ~ 6/25(수)	기능 점검, 버그 수정
보고서 작성	6/25(수) ~ 6/30(월)	프로젝트 결과 보고서 작성 및 제출

# 07 E-R 다이어그램





# 08 주요 코드

이메일 인증 기반  
회원가입 코드의 일부

```
@PostMapping("/join")
public String joinForm(@Valid UserDto userDto, BindingResult bindingResult, Model model, HttpSession session) {

    // 이메일 인증 확인
    String verifiedEmail = (String) session.getAttribute("emailVerified");
    if (verifiedEmail == null || !verifiedEmail.equals(userDto.getEmail())) {
        model.addAttribute("emailError", "이메일 인증을 완료해주세요.");
        model.addAttribute("userDto", userDto);
        return "user/join";
    }

    // 1. 기본 유효성 검사
    if (bindingResult.hasErrors()) {
        // 에러가 있으면 에러 메시지 Map 생성해서 모델에 담음
        model.addAttribute("errors", getErrors(bindingResult));
        model.addAttribute("userDto", userDto);
        return "user/join";
    }

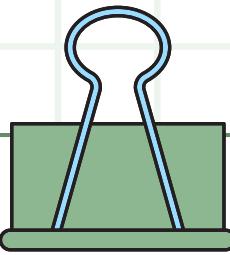
    // 2. 중복 검사 (bindingResult에 에러 추가)
    userService.isDuplicate(userDto, bindingResult);

    // 3. 중복 검사 후 에러가 있으면 다시 뷰로
    if (bindingResult.hasErrors()) {
        model.addAttribute("errors", getErrors(bindingResult));
        model.addAttribute("userDto", userDto);
        return "user/join";
    }

    // 4. 회원가입 진행
    System.out.println("회원가입 처리 시작");
    userService.join(userDto);
    System.out.println("회원가입 완료 후 리다이렉트");
    // 회원가입 후 세션에서 인증 정보 제거
    session.removeAttribute("emailVerified");

    System.out.println("emailVerified after remove: " + session.getAttribute("emailVerified"));

    return "redirect:/login";
}
```



# 08 주요 코드

이메일 인증을 통한 비밀번호 찾기  
코드의 일부

```
// STEP 1 - 인증번호 발송
@PostMapping("/find-password/send-code")
public String sendCode(@RequestParam("id") String id,
                      @RequestParam("email") String email,
                      Model model) {
    Optional<UserEntity> optionalUser = userRepository.findById(id);

    if (optionalUser.isEmpty() || !optionalUser.get().getEmail().equals(email)) {
        model.addAttribute("error", "아이디와 이메일이 일치하지 않습니다.");
        return "user/find-password";
    }

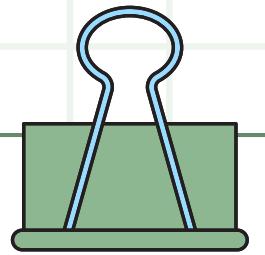
    // 인증번호 생성 & 저장
    String code = String.format("%04d", new Random().nextInt(10000));
    verificationCodes.put(email, code);

    // 이메일 전송
    emailService.sendEmail(email, "비밀번호 재설정 인증번호", "인증번호: " + code);
    model.addAttribute("email", email);
    model.addAttribute("id", id);
    model.addAttribute("message", "인증번호가 전송되었습니다.");
    return "user/find-password";
}

// STEP 2 - 인증번호 확인
@PostMapping("/find-password/verify-code")
public String verifyCode(@RequestParam("id") String id,
                       @RequestParam("email") String email,
                       @RequestParam("code") String code,
                       Model model) {
    String savedCode = verificationCodes.get(email);

    if (savedCode != null && savedCode.equals(code)) {
        verificationCodes.remove(email); // 일회용
        model.addAttribute("id", id);
        model.addAttribute("email", email);
        return "user/reset-password";
    }

    model.addAttribute("error", "인증번호가 일치하지 않습니다.");
    return "user/find-password";
}
```



# 08 주요 코드

키워드 추출을 통한 챗봇  
코드의 일부

```
// 질문 받는 POST 요청
@PostMapping("/ask")
public String ask(@RequestParam("question") String question, Model model, HttpSession session) {

    UserEntity loginUser = (UserEntity) session.getAttribute("loginUser");
    String userId = loginUser.getId();
    // userId가 없는 경우 기본 답변 처리
    if (userId == null) {
        return "/login";
    }

    String answer = chatbotService.getAnswer(question,userId);

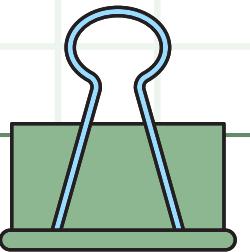
    // 세션에 저장된 대화 리스트 가져오기 (없으면 새로 생성)
    List<FaqDto> chatHistory = (List<FaqDto>) session.getAttribute("chatHistory");
    if (chatHistory == null) {
        chatHistory = new ArrayList<>();
    }

    // 새 질문과 답변을 FaqDto 객체로 만들어 리스트에 추가
    FaqDto newEntry = new FaqDto();
    newEntry.setKeyword(question); // 질문을 keyword 필드에 저장
    newEntry.setAnswer(answer); // 답변을 answer 필드에 저장
    chatHistory.add(newEntry);

    // 다시 세션에 저장
    session.setAttribute("chatHistory", chatHistory);

    // 모델에도 대화 리스트 전달
    model.addAttribute("chatHistory", chatHistory);
}

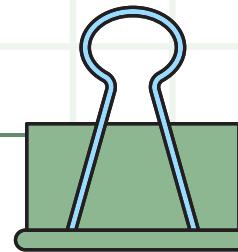
return "chatbot/chatbot";
}
```



# 08 주요 코드

챗봇 기능 중 하나로  
주문내역 키워드 입력 시  
자신의 주문내역 출력  
코드의 일부

```
public String getAnswer(String userInput, String userId) {  
    // "주문내역", "내 주문" 등 키워드 감지  
    if (containsIgnoreWhitespace(userInput, "주문내역") || containsIgnoreWhitespace(userInput, "내 주문")) {  
        return getOrderHistory(userId);  
    }  
  
    List<FaqEntry> faqList = faqRepository.findAll();  
    int no = 1;  
    for (FaqEntry entry : faqList) {  
        if (containsIgnoreWhitespace(userInput, entry.getKeyword())) {  
            return entry.getAnswer();  
        }  
    }  
  
    return "죄송합니다. 질문을 이해하지 못했습니다.";  
}  
  
public String getOrderHistory(String userId) {  
    List<OrderSummaryDTO> summaries = ordersRepository.findOrderSummaries(userId);  
  
    if (summaries.isEmpty()) {  
        return "주문 내역이 없습니다.";  
    }  
  
    StringBuilder sb = new StringBuilder("고객님의 최근 주문 내역입니다:<br>");  
    for (OrderSummaryDTO summary : summaries) {  
        sb.append("- 주문번호: ").append(summary.getOrderGroup())  
        .append(", 날짜: ").append(summary.getOrderDate())  
        .append("<br>상품 내역<br>");  
  
        List<OrderDetailDTO> details = orderDetailRepository.findDetailsByOrderGroup(summary.getOrderGroup());  
        for (OrderDetailDTO detail : details) {  
            sb.append("상품명: ").append(detail.getProductName())  
            .append(", 수량: ").append(detail.getQuantity())  
            .append("<br>");  
        }  
        sb.append("<br>");  
    }  
    return sb.toString();  
}
```



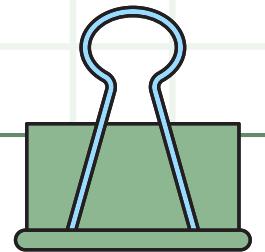
# 08 주요 코드

사용자 목록 조회 코드의 일부

```
// ===== AdminController.java =====

/**
 * [사용자 목록] - 키워드 검색이 있을 때와 없을 때를 분기.
 * 1) 키워드 O → 이름 LIKE 검색 (관리자 제외)
 * 2) 키워드 X → 전체 리스트 (관리자 제외)
 */
@RequestMapping("userList")
public String userList(@RequestParam(value = "keyword", required = false) String keyword,
                      Model model) {
    List<AdminEntity> userList;
    if (keyword != null && !keyword.trim().isEmpty()) {
        // 이름 부분 일치 검색 + role != ADMIN
        userList = adminService.findUsersByName(keyword.trim());
    } else {
        // 관리자 계정 제외 전체 조회
        userList = adminService.findAllUsersExceptAdmin();
    }
    model.addAttribute("userList", userList);
    return "admin/userList";
}

/**
 * [등급 변경] - 관리자가 선택한 멤버십 등급으로 업데이트.
 * - 등급 정책 확장 시 진입점이 되는 메서드
 * - Alert 메시지는 세션에 저장하여 Redirect 후 1회 표시
 */
@RequestMapping(value = "updateGrade", method = RequestMethod.POST)
public String updateGrade(@RequestParam("id") String id,
                         @RequestParam("grade") String grade,
                         HttpSession session) {
    adminService.updateGrade(id, grade); // Service → Repository
    session.setAttribute("alertMsg", "등급 변경이 완료되었습니다.");
    return "redirect:/admin/userList";
}
```



# 08 주요 코드

판매자 입점 신청 목록 조회  
코드의 일부

```
/*
 * [판매자 입점 신청 목록] - 관리자 권한 검증 + 목록 조회.
 * - ADMIN 외 접근 시 JS alert 후 메인으로 리다이렉트
 */
@RequestMapping("applyList")
public String viewApplyList(HttpServletRequest session, Model model, HttpServletResponse response) throws IOException {
    UserEntity loginUser = (UserEntity) session.getAttribute("loginUser");

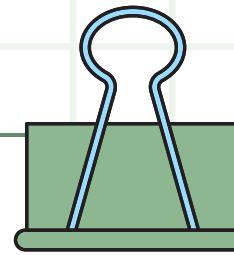
    if (loginUser == null || !"ADMIN".equals(loginUser.getRole())) {
        // 간단한 대응: 클라이언트 스크립트로 경고 후 루트 이동
        response.setContentType("text/html;charset=UTF-8");
        response.getWriter().println("<script>alert('관리자만 접근 가능한 페이지입니다.'); location.href='/';</script>");
        return null;
    }

    List<SellerRoleEntity> applyList = sellerRoleService.getAllApplications(); // 입점 신청 전체
    model.addAttribute("applyList", applyList);

    return "admin/applyList";
}

/*
 * [사용자 정보 수정] - 이메일 중복 검사 & 선택적 비밀번호 변경.
 */
@PostMapping("updateUser")
public String updateUser(@RequestParam("id") String id,
                        @RequestParam("email") String email,
                        @RequestParam(value = "passwd", required = false) String passwd,
                        RedirectAttributes ra) {

    try {
        adminService.updateUser(id, email, passwd); // Service 레이어에서 무결성 검사
        ra.addFlashAttribute("alertMsg", "수정 완료!");
    } catch (IllegalStateException e) {
        // 이미 사용 중인 이메일 등 예외 메시지를 그대로 전달
        ra.addFlashAttribute("alertMsg", e.getMessage());
    }
    return "redirect:/admin/userList";
}
```



# 08 주요 코드

관리자 페이지에서 사용자 등급변경  
코드의 일부

```
// ===== AdminServiceImpl.java =====

/**
 * [사용자 정보 수정 Service] - @Transactional 한 트랜잭션 안에서 처리.
 * 1) 이메일 중복 여부 검사 (본인 제외)
 * 2) 이메일·비밀번호 업데이트 (패스워드 입력했을 때만)
 */
@Transactional
@Override
public void updateUser(String id, String email, String passwd) {
    // 자기 자신(id) 제외 동일 이메일 존재 여부 체크
    if (adminRepository.existsByEmailAndIdNot(email, id)) {
        throw new IllegalStateException("이미 사용 중인 이메일입니다.");
    }

    // findById → Optional.orElseThrow()로 존재 검사
    AdminEntity user = adminRepository.findById(id)
        .orElseThrow();
    user.setEmail(email.trim());

    // 비밀번호는 입력이 있을 때만 변경하여 불필요한 해시 생성을 방지
    if (passwd != null && !passwd.isBlank()) {
        user.setPasswd(passwd);
    }
}

/**
 * [등급 변경 Service]
 */
@Override
public void updateGrade(String id, String grade) {
    Optional<AdminEntity> optional = adminRepository.findById(id);
    if (optional.isPresent()) {
        AdminEntity user = optional.get();
        user.setGrade(grade);
        adminRepository.save(user);
    }
}
```



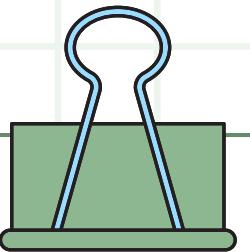
# 08 주요 코드

이메일 중복여부 확인 및  
등급변경 코드의 일부

```
// ===== AdminRepository.java =====

/**
 * [이메일 중복 검사] - "자기 자신(id) 제외" 조건 포함.
 */
boolean existsByEmailAndIdNot(String email, String id);

/**
 * [JPQL] 특정 회원 등급 업데이트.
 */
@Modifying
@Query("UPDATE AdminEntity u SET u.grade = :grade WHERE u.id = :id")
void updateUserGrade(@Param("id") String id, @Param("grade") String grade);
```



# 08 주요 코드

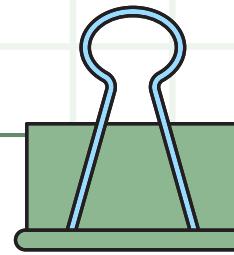
판매자 관한 검증  
코드의 일부

```
// 판매자 권한 검증 시스템
private String validateSellerAccessWithAlert(HttpServletRequest session) {
    // 1단계: 로그인 체크
    UserEntity loginUser = (UserEntity) session.getAttribute("loginUser");
    if (loginUser == null) {
        return "alert/login-required";
    }

    // 2단계: 판매자 권한 체크
    if (!"SELLER".equals(loginUser.getRole())) {
        return "alert/seller-auth-required";
    }

    // 3단계: 승인된 판매자인지 체크
    String companyName = sellerRoleService.getCompanyNameBySellerId(loginUser.getId());
    if (companyName == null) {
        return "alert/seller-approval-required";
    }

    return null;
}
```



# 08 주요 코드

상품 등록 처리 코드의 일부

```
// 상품 등록 처리
@PostMapping("/register")
public String registerProduct(@ModelAttribute("productDto") ProductRegistrationDto productDto,
                               @RequestParam(value = "productImageFile", required = false) MultipartFile
productImageFile,
                               Model model, RedirectAttributes redirectAttributes, HttpSession session) {

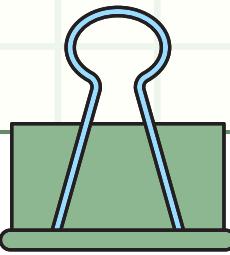
    // 권한 검증
    String validationResult = validateSellerAccessWithAlert(session);
    if (validationResult != null) {
        return validationResult;
    }

    // 회사명 자동 설정
    String companyName = getCurrentSellerCompany(session);
    productDto.setCompanyName(companyName);

    // 필드 검증
    if (productDto.getCategory() == null || productDto.getCategory().trim().isEmpty()) {
        model.addAttribute("errorMessage", "카테고리를 선택해주세요.");
        model.addAttribute("categories", productService.getAllCategories());
        model.addAttribute("sizes", productService.getAllSizes());
        return "seller/register";
    }

    if (productDto.getProductName() == null || productDto.getProductName().trim().isEmpty()) {
        model.addAttribute("errorMessage", "상품명을 입력해주세요.");
        model.addAttribute("categories", productService.getAllCategories());
        model.addAttribute("sizes", productService.getAllSizes());
        return "seller/register";
    }

    try {
        productService.registerProduct(productDto, productImageFile);
        redirectAttributes.addFlashAttribute("successMessage", "상품이 성공적으로 등록되었습니다.");
        return "redirect:/seller/register";
    } catch (Exception e) {
        model.addAttribute("errorMessage", e.getMessage());
        model.addAttribute("categories", productService.getAllCategories());
        model.addAttribute("sizes", productService.getAllSizes());
        return "seller/register";
    }
}
```



# 08 구현영상

사용자

회원가입

아이디  
dudals5608

비밀번호  
\*\*\*\*

이름  
김영민

이메일  
김영민  
경기 수원시 팔달구 고등동  
인증번호 받기

전화번호

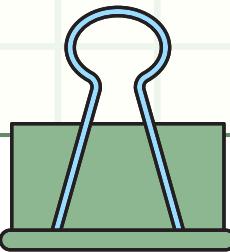
주소

우편번호  
주소 찾기

주소

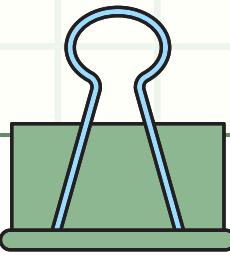
**회원가입 중~**

안  
이미 계정이 있으신가요? 로그인



# 09 질문과 답변

Q & A



감사합니다

THANK  
YOU!