

CHALKAG

사진 작가와 모델이 소통하는 플랫폼

[프로젝트 GitHub](#)

[프로젝트 Notion](#)

INFINITYSTONE

목차

- 팀원 소개 및 역할 분담
- 프로젝트 목적
- 프로젝트 주요 기능
- 프로젝트 설계
- 주요 기능 시연 및 코드 설명
- QnA



팀원 소개 및 역할 분담



팀원1

팀장
회원 서비스
관리자 서비스
결제 API
1:1 채팅



팀원2

팀원
글, 댓글, 답글 페이지
글 페이지 처리
이미지 업로드



팀원3

팀원
메인 컨트롤러
SMS 인증 API
소셜 로그인 API



팀원4

팀원
게시글 서비스
댓글 서비스
게시판 필터 검색



팀원5

팀원
게시글 서비스
신고 서비스
게시판 필터 검색



정석진

팀원
회원 관련 페이지
관리자 페이지
신고 기능

프로젝트 기대 효과

- Spring MVC 패턴에 대한 이해 확립
- API에 대한 이해와 활용 능력 향상
- Git 활용 능력 및 이를 통한 협업 능력 향상
- 설계 회의를 통한 의사소통 능력 향상
- 코드 취합을 통한 코드 리뷰 능력 향상
- 협업을 위한 주석 작성의 습관화
- 오류 탐색 및 오류 발생 시 해결 능력 향상

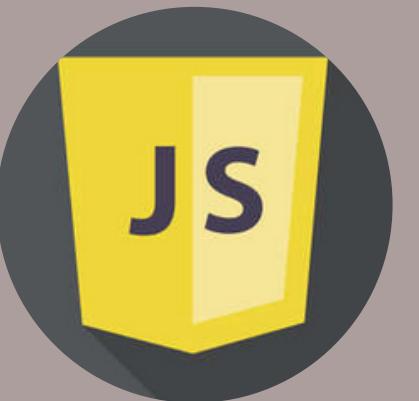
프로젝트 목적

- 사진 작가와 모델을 매칭하고 서로의 정보를 공유할 수 있는 플랫폼을 제공
 - Spring MVC 패턴에 대한 이해 확립
 - API의 이해와 활용
 - 기존 프로젝트를 Spring 으로 이관함으로써 Spring 구조 이해
-

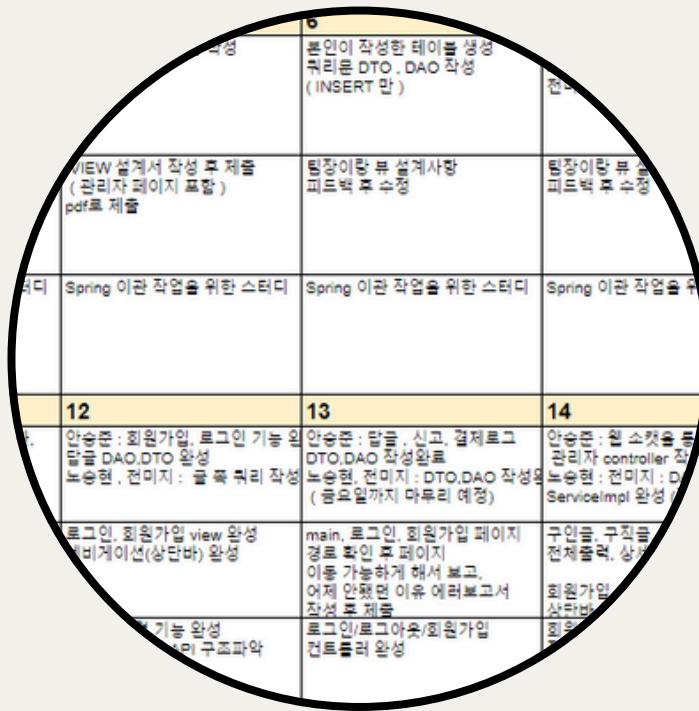
프로젝트 주요 기능

- 실시간 1:1 채팅
- 게시글 필터 검색
- 답글 작성, 수정
- 관리자 메뉴

개발 환경

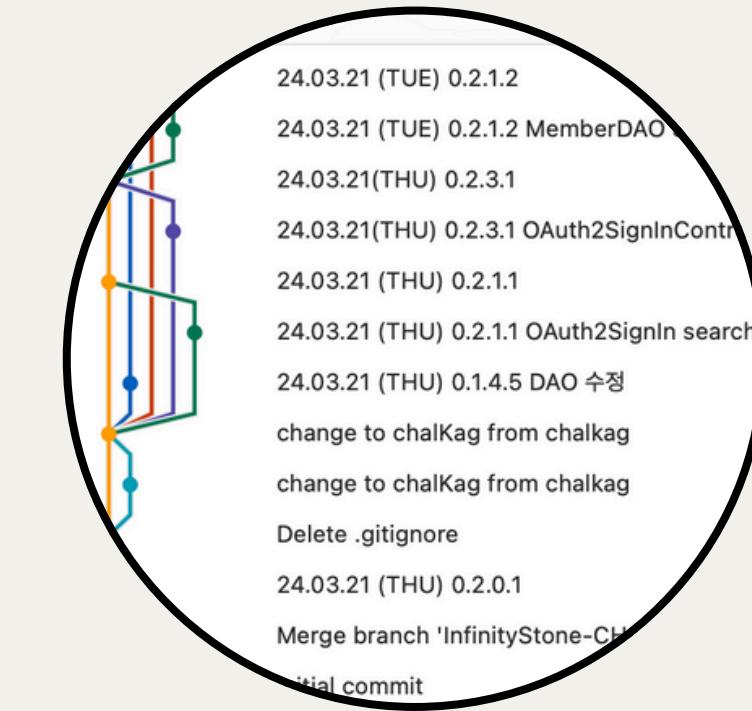


협업 관리



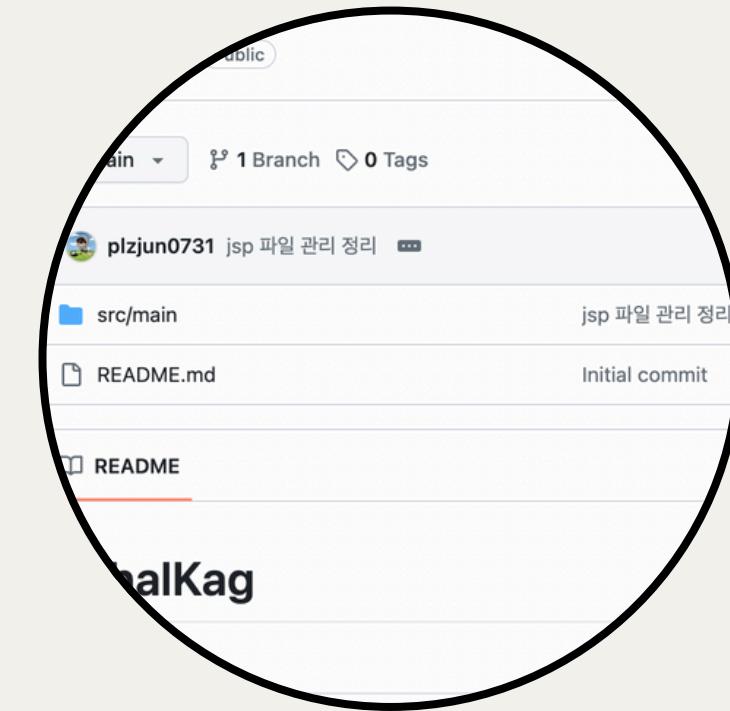
일정 관리

공유 스프레드 시트를 활용해서
파트 별로 작업 내역 작성



버전 관리

0.0.0.1 버전으로 시작
각 팀원들에게 고유번호 부여
개인 버전 관리



버전 관리

기능 구현 완료 시 취합
이후 버전 증가

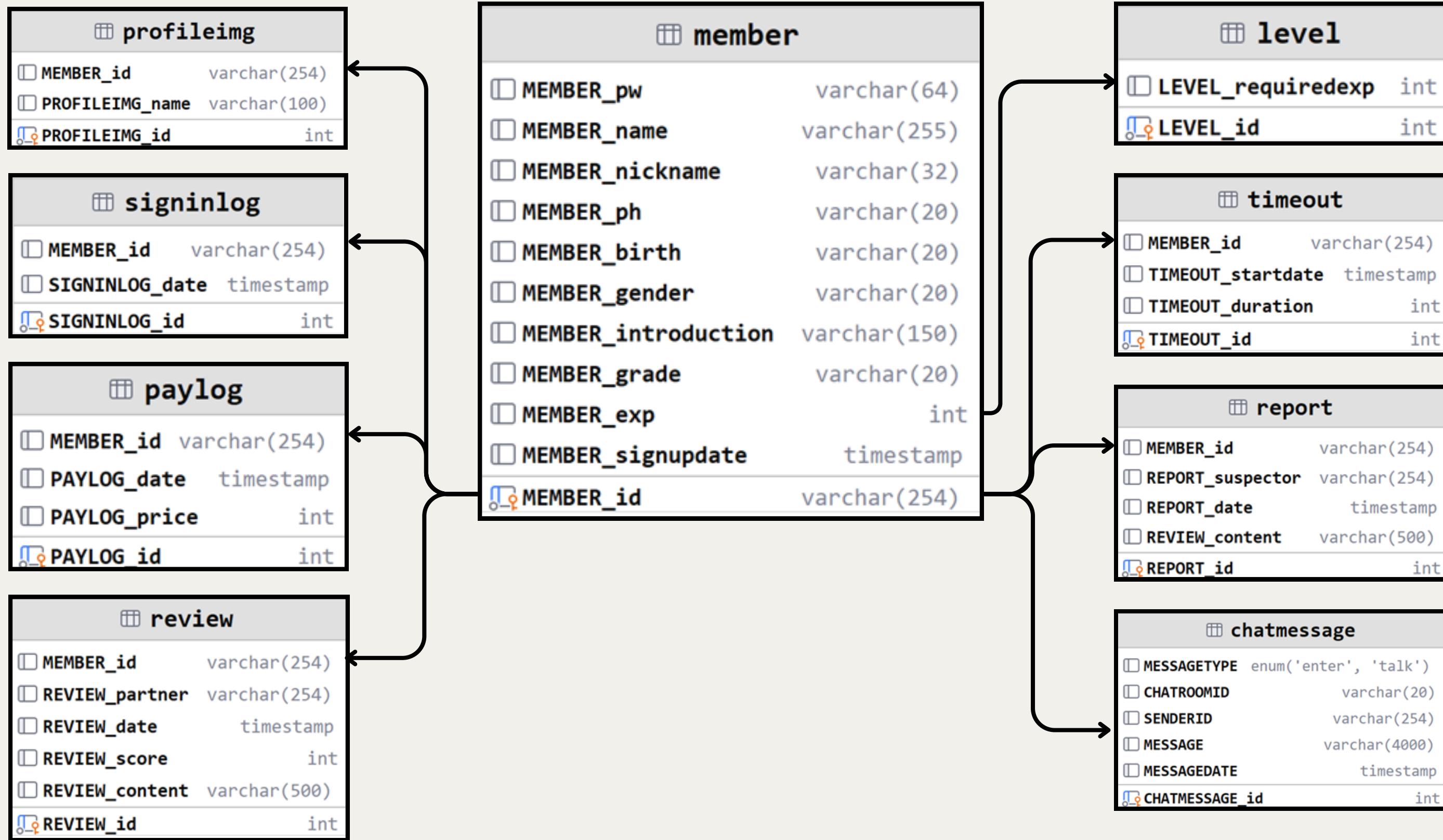
프로젝트 개발 일정

<h2><CHAL KAG></h2>							
INFINITYSTONE		PROJECT MANAGER: 안승준 , 김도연 , 김성민 , 노승현 , 전미지 , 정석진				START DATE: 2024.02.27	
TASK	START DATE	COMPLETION DATE	PROJECT DATA				
			Week 1	Week 2	Week 3	Week 4	Week 5
설계			02.27	03.05	03.12	03.19	03.26
데이터 설계	24.02.27 (TUE)	24.03.04 (MON)					
컨트롤러 설계	24.02.27 (TUE)	24.03.10 (SUN)					
뷰 설계	24.02.27 (TUE)	24.03.10 (SUN)					
Logic Process 작성	24.02.27 (TUE)	24.03.10 (SUN)					
데이터 설계	24.02.27 (TUE)	24.03.10 (SUN)					
컨트롤러 이관	24.03.06 (WED)						
채팅 로직 설계							
결제 API 로직 설계							
기능구현							
로그인, 회원가입 기능 구현	24.03.11 (MON)	24.03.12 (TUE)					
구직글, 구인글 목록 출력	24.03.11 (MON)	24.03.14 (THU)					
프로필 사진 업로드 기능 추가	24.03.18 (MON)	24.03.19 (TUE)					
SNS 로그인 기능 구현	24.03.18 (MON)	24.03.20 (WED)					

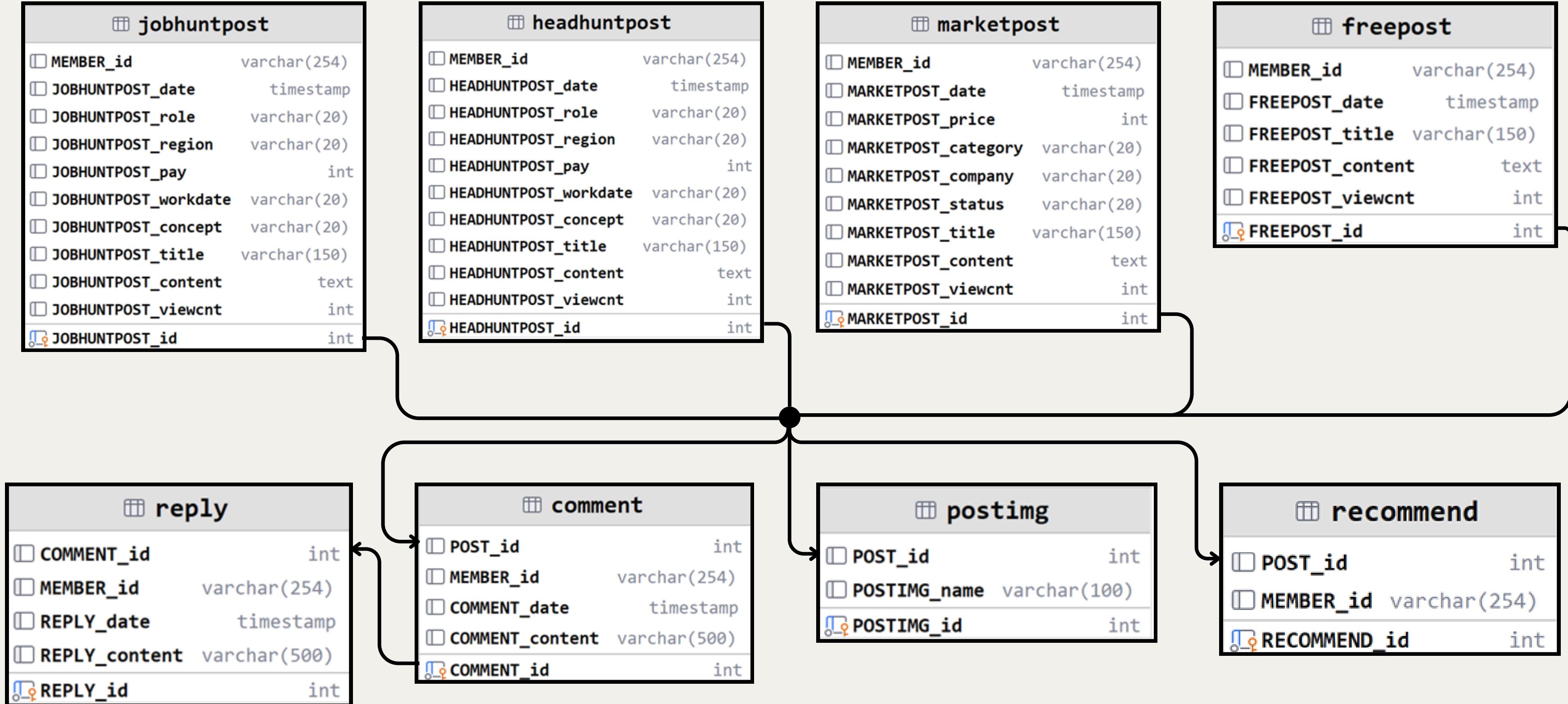
공유 스프레드 시트를 활용해서
프로젝트 개발 일정 관리

[프로젝트 설계 자세히 보기](#)

ERD - 회원



ERD - 게시글



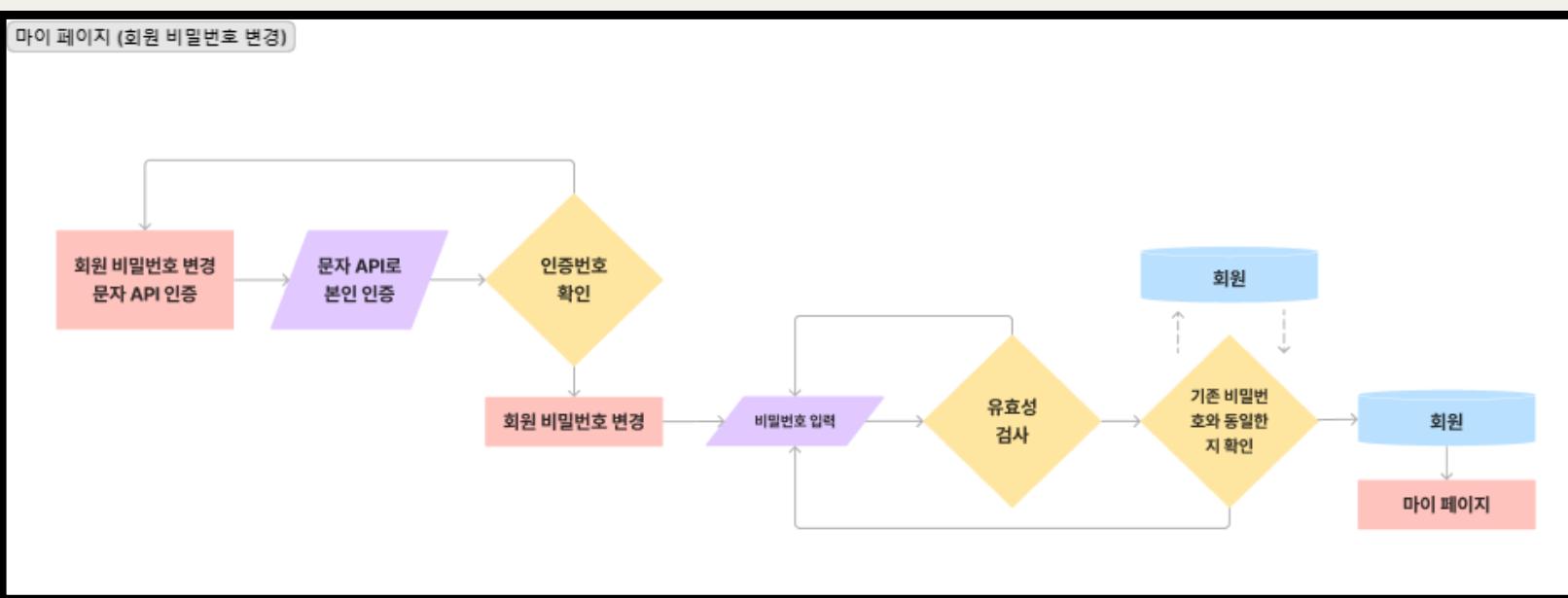
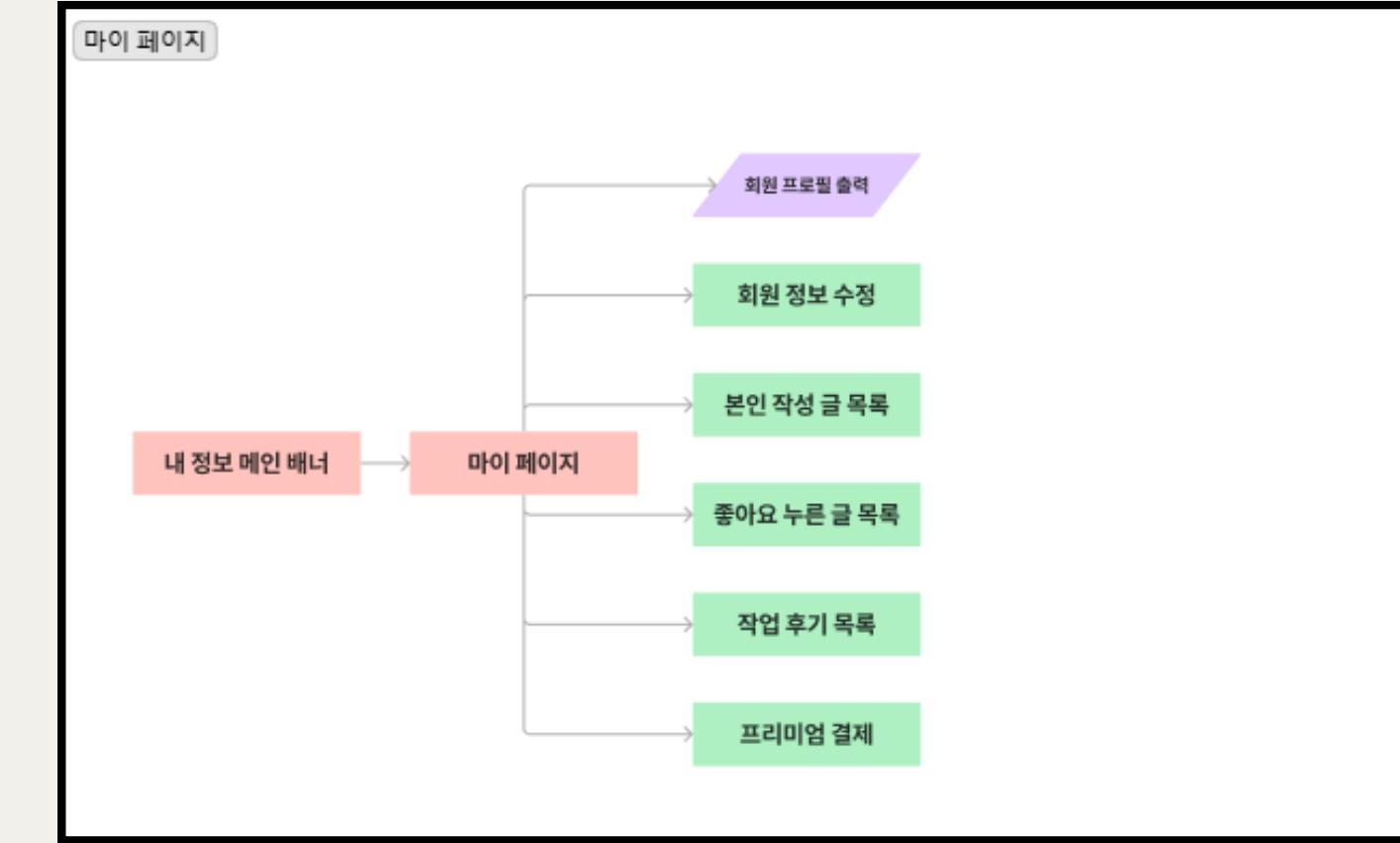
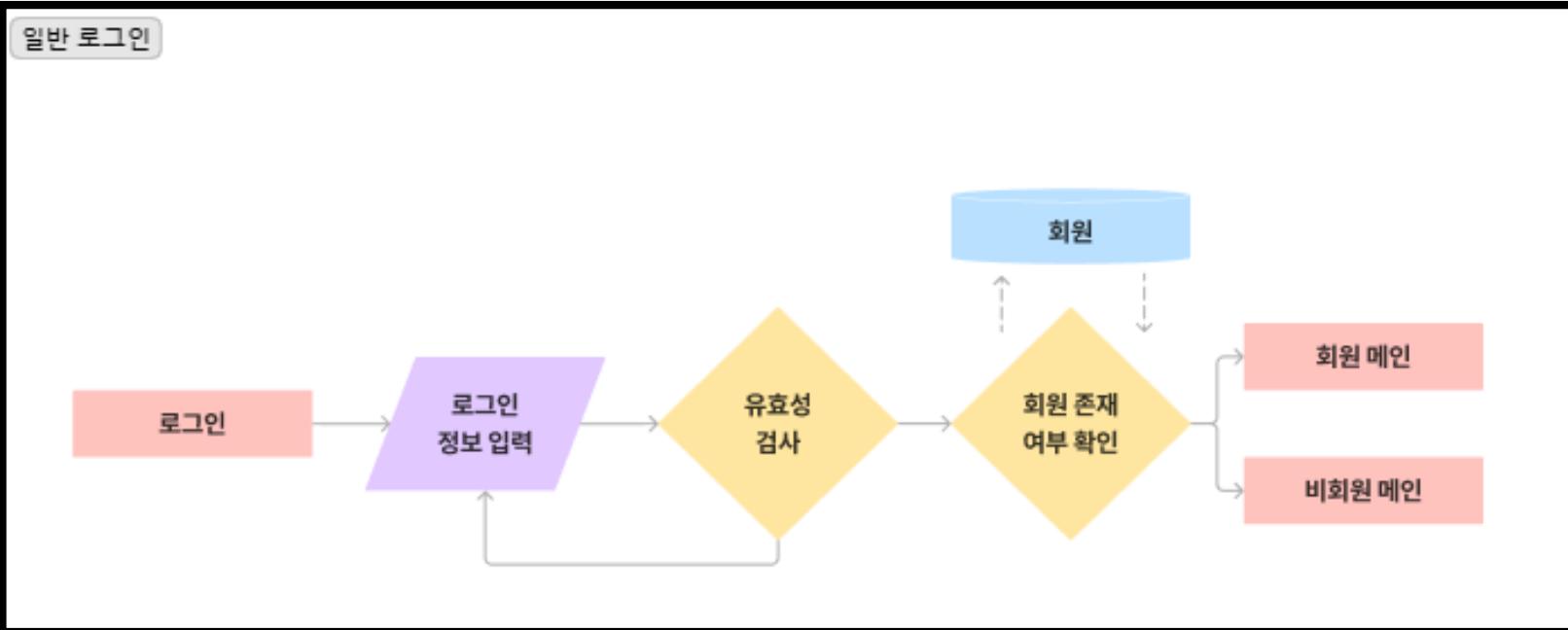
기능 정의서 (1/3)

기능 정의서 (2/3)

6	구직글	구직글 작성	<p>구직글 제작 일자 구직글 사진 업로드 구직글 하면 친해 구직글 하면 지역 선택 구직글 하면 페이 인증 구직글 내용 영역 선택 구직글 내용 일자 구직글의 모든 내용은 필수 시장이므로 유효성 검사</p>	<p>회원 아이디 구직글 작업 구직글 작업 지역 구직글 작업 페이 구직글 작업 날짜 구직글 활영 컨셉 구직글 제목 구직글 내용 구직글 조회수 게시글 사진 이름</p>		<p>writeFreePost</p> <p>자유글 작성</p>	<p>자유글 제작 일자 자유글 사진 업로드 자유글 내용 자유글 제작 일자 자유글 내용 자유글 제작 페이 자유글 내용 자유글 조회수 게시글 사진 이름</p>	<p>회원 아이디 자유글 제작 자유글 내용 회원 아이디 자유글 새로운 제작 자유글 내용 게시글 새로운 사진 이름</p>	
	구직글 수정		<p>구직글 새로운 제작 일자 구직글 사진 업로드 구직글 하면 친해 주기 혹은 삭제 구직글 하면 지역 선택 구직글 새로운 작업 지역 구직글 새로운 페이 구직글 새로운 활영 컨셉 구직글 새로운 내용 구직글 새로운 사진 이름</p>	<p>회원 아이디 구직글 새로운 작업 구직글 새로운 작업 구직글 새로운 작업 지역 구직글 새로운 작업 페이 구직글 새로운 활영 컨셉 구직글 새로운 내용 구직글 새로운 사진 이름</p>		<p>updateFreePost</p> <p>자유글 수정</p>	<p>자유글 새로운 제작 일자 자유글 사진 업로드 자유글 내용 자유글 제작 일자 자유글 내용 자유글 제작 페이 자유글 내용 자유글 조회수 게시글 사진 이름</p>	<p>회원 아이디 자유글 새로운 제작 자유글 내용 게시글 새로운 사진 이름</p>	
	구직글 삭제	구직글 삭제	구직글 아이디를 통해 삭제	구직글 아이디		<p>deleteFreePost</p>	<p>자유글 제작 자유글 작성 시간 자유글 내용 자유글 조회수 자유글 좋아요 수 자유글 제작 회원 아이디 회원 닉네임 자유글 작성 시간 자유글 내용 자유글 좋아요 수 자유글 조회수 프리미엄 회원의 글 광고 사진</p>	<p>회원 아이디 회원 닉네임 자유글 작성 시간 자유글 내용 자유글 좋아요 수 자유글 조회수 프리미엄 회원의 글 광고 사진</p>	
7	구직글 목록	구직글 목록	<p>구직글 대표사진 (첫번째 사진) / 작성자 / 작성시간(elapseTime) / 글 내용 / 좋아요 수 / 조회수 구직글 작성 버튼 프리미엄 페이 작업 날짜 (단위) 활영 컨셉 작업 지역 제작으로 검색 내용으로 검색 제작 + 내용으로 검색 10개씩 10개 페이지 출력</p>	<p>게시글 사진 이름 구직글 아이디 회원 아이디 회원 닉네임 자유글 작성 시간 자유글 내용 자유글 좋아요 수 자유글 조회수 프리미엄 회원의 글 광고 사진</p>		<p>freePostList</p> <p>자유글 목록</p>	<p>글 순서 / 자유글 대표사진 (첫번째 사진) / 작성자 / 작성시간(elapseTime) / 글 내용 / 좋아요 수 / 조회수 제작으로 검색 내용으로 검색 제작 + 내용으로 검색 10개씩 10개 페이지 출력</p>	<p>게시글 사진 이름 구직글 아이디 회원 아이디 회원 닉네임 자유글 작성 시간 자유글 내용 자유글 좋아요 수 자유글 조회수 프리미엄 회원의 글 광고 사진</p>	
	구직글 자세히보기	구직글 자세히보기	<p>구직글 제작 회원 아이디 회원 닉네임 구직글 작성 시간 구직글 작성 구직글 페이 구직글 작업 날짜 구직글 활영 컨셉 구직글 내용 구직글 조회수 구직글 좋아요 수 수정 및 삭제 회원 사이드 바 광고 / 프리미엄 회원의 글 5개 출력 (내가 작성 한 글일 때는 수정 , 삭제 버튼)</p>	<p>구직글 제작 회원 아이디 회원 닉네임 구직글 작성 시간 구직글 작성 구직글 페이 구직글 작업 날짜 구직글 활영 컨셉 구직글 내용 구직글 조회수 구직글 좋아요 수 프리미엄 회원의 글 광고 사진</p>		<p>freePostSingle</p> <p>자유글 자세히 보기</p>	<p>글 순서 / 자유글 대표사진 (첫번째 사진) / 작성자 / 작성시간(elapseTime) / 글 내용 / 좋아요 수 / 조회수 제작으로 검색 내용으로 검색 제작 + 내용으로 검색 10개씩 10개 페이지 출력</p>	<p>게시글 사진 이름 구직글 아이디 회원 아이디 회원 닉네임 자유글 작성 시간 자유글 내용 자유글 좋아요 수 자유글 조회수 프리미엄 회원의 글 광고 사진</p>	
	내가 쓴 구직글 보기	내가 쓴 구직글 보기	<p>글 순서 / 구직글 대표사진 (첫번째 사진) / 작성자 / 작성시간(elapseTime) / 글 내용 / 좋아요 수 / 조회수 구직글 작성 버튼 작업 날짜 (단위) 활영 컨셉 작업 지역 제작으로 검색 내용으로 검색 제작 + 내용으로 검색 10개씩 10개 페이지 출력</p>	<p>게시글 사진 이름 구직글 아이디 회원 아이디 회원 닉네임 구직글 작성 시간 구직글 작성 구직글 페이 구직글 작업 날짜 구직글 활영 컨셉 구직글 내용 구직글 좋아요 수 구직글 조회수 프리미엄 회원의 글 광고 사진</p>		<p>myFreePostList</p> <p>내가 쓴 자유글 보기</p>	<p>글 순서 / 자유글 대표사진 (첫번째 사진) / 작성자 / 작성시간(elapseTime) / 글 내용 / 좋아요 수 / 조회수 제작으로 검색 내용으로 검색 제작 + 내용으로 검색 10개씩 10개 페이지 출력</p>	<p>게시글 사진 이름 구직글 아이디 회원 아이디 회원 닉네임 자유글 작성 시간 자유글 내용 자유글 좋아요 수 자유글 조회수 프리미엄 회원의 글 광고 사진</p>	

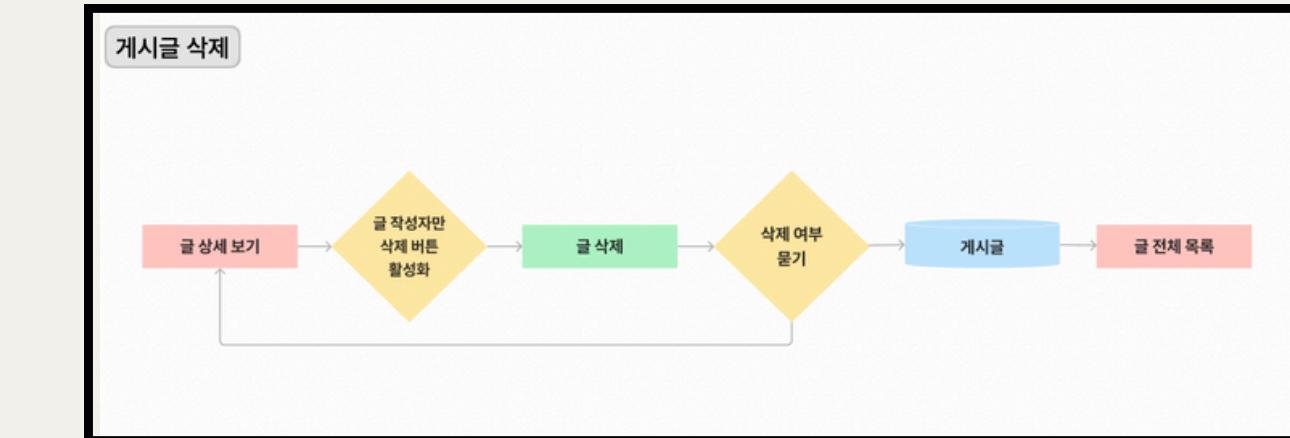
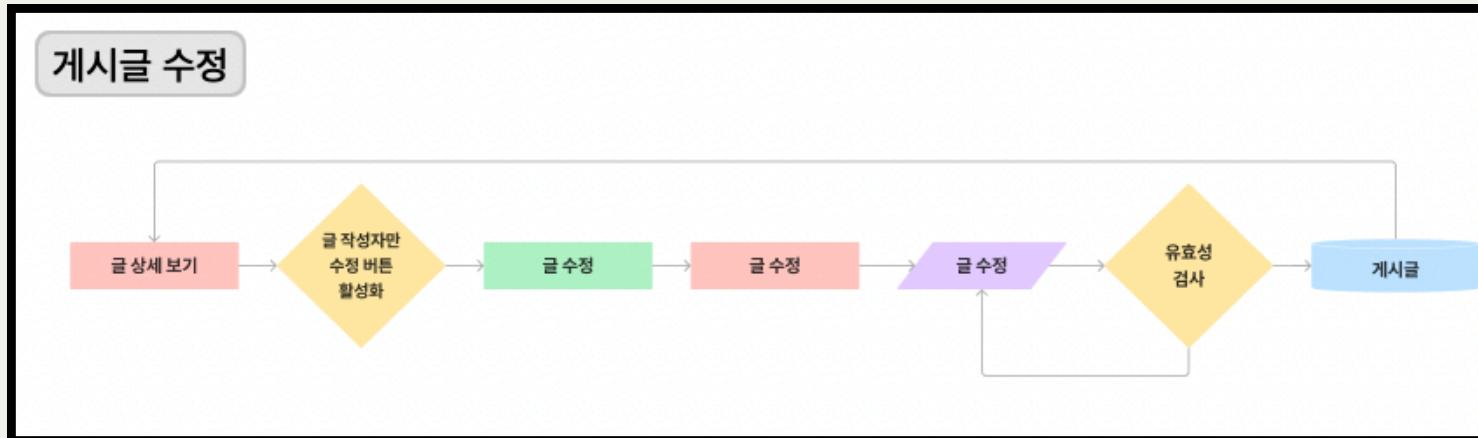
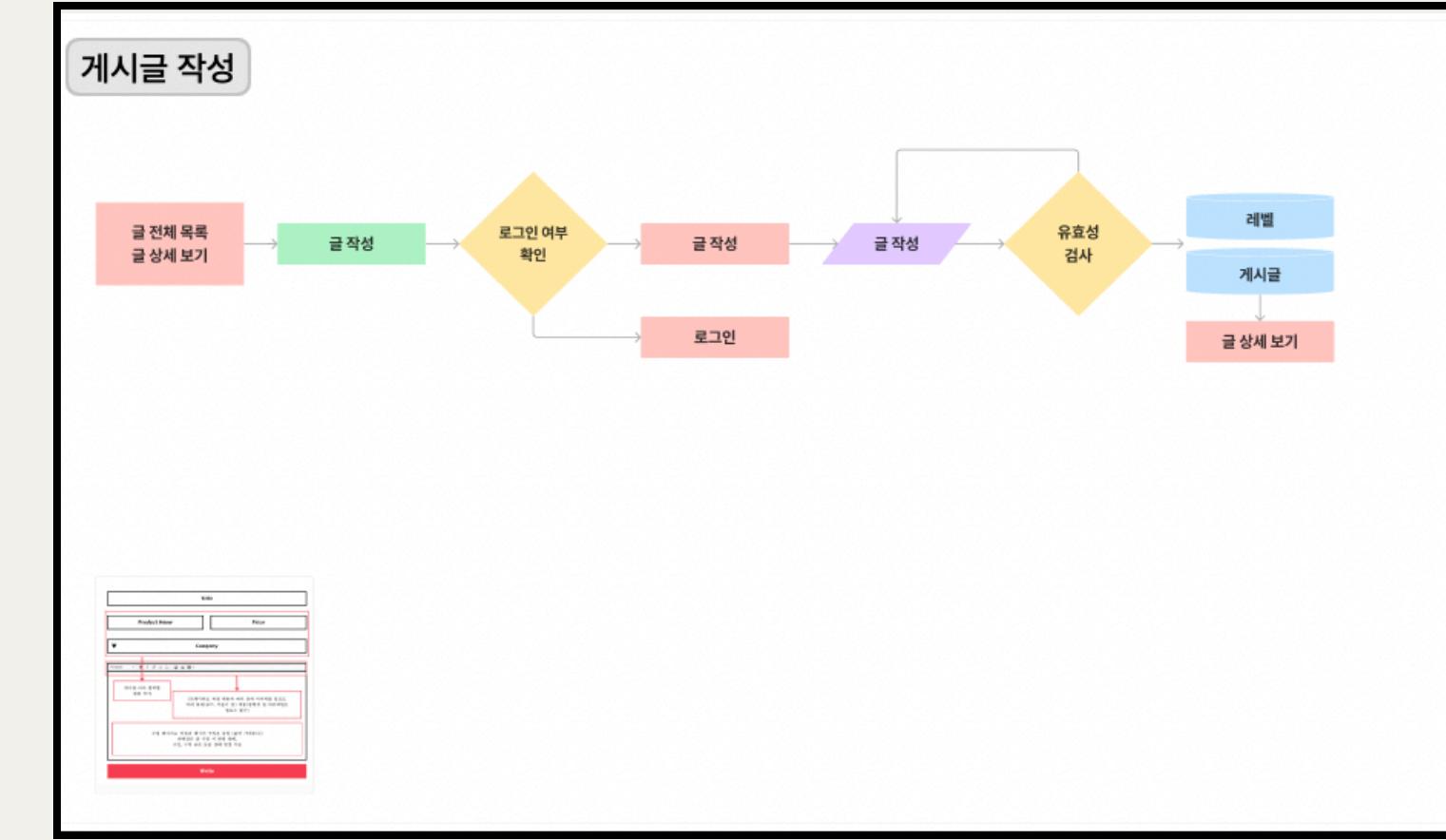
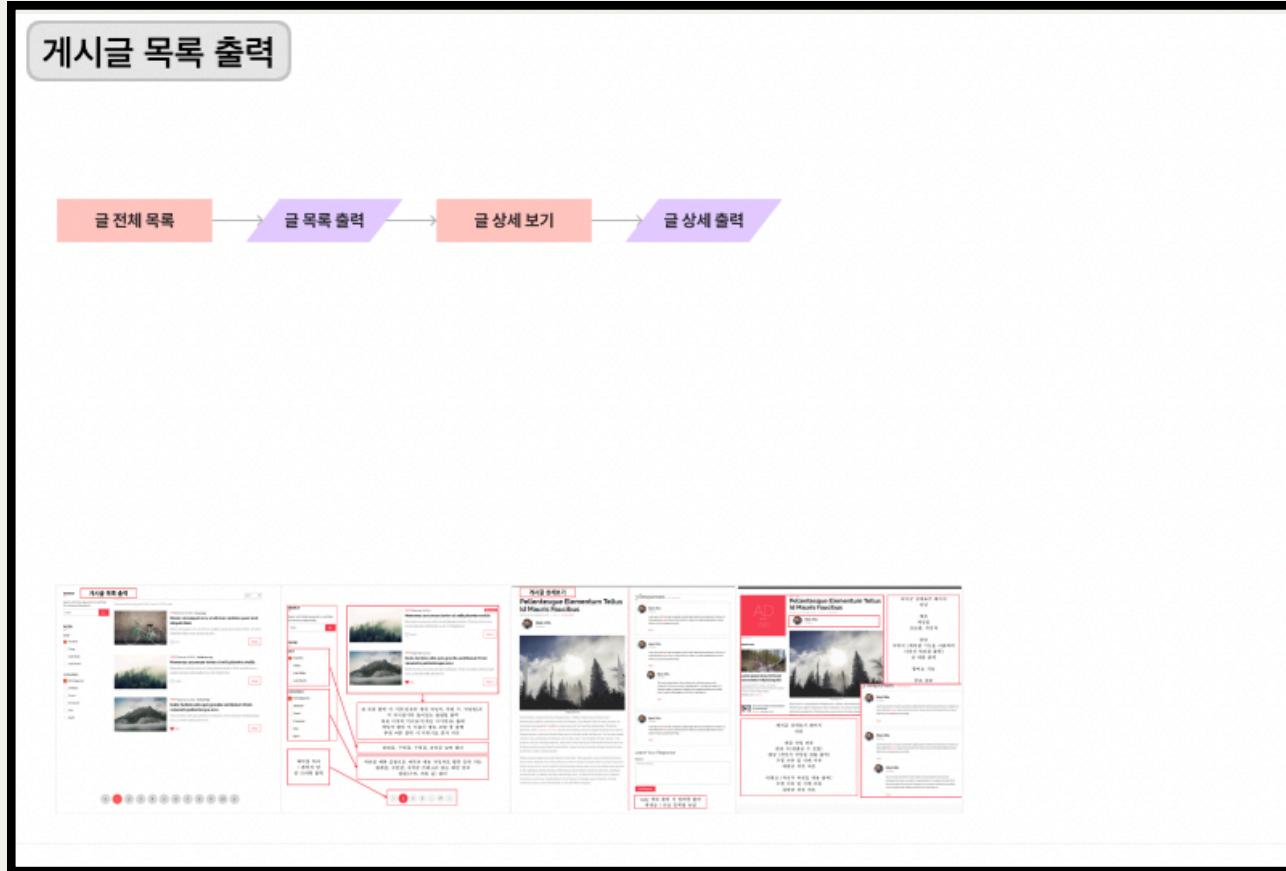
기능 정의서 (3/3)

Logic Process - 회원가입

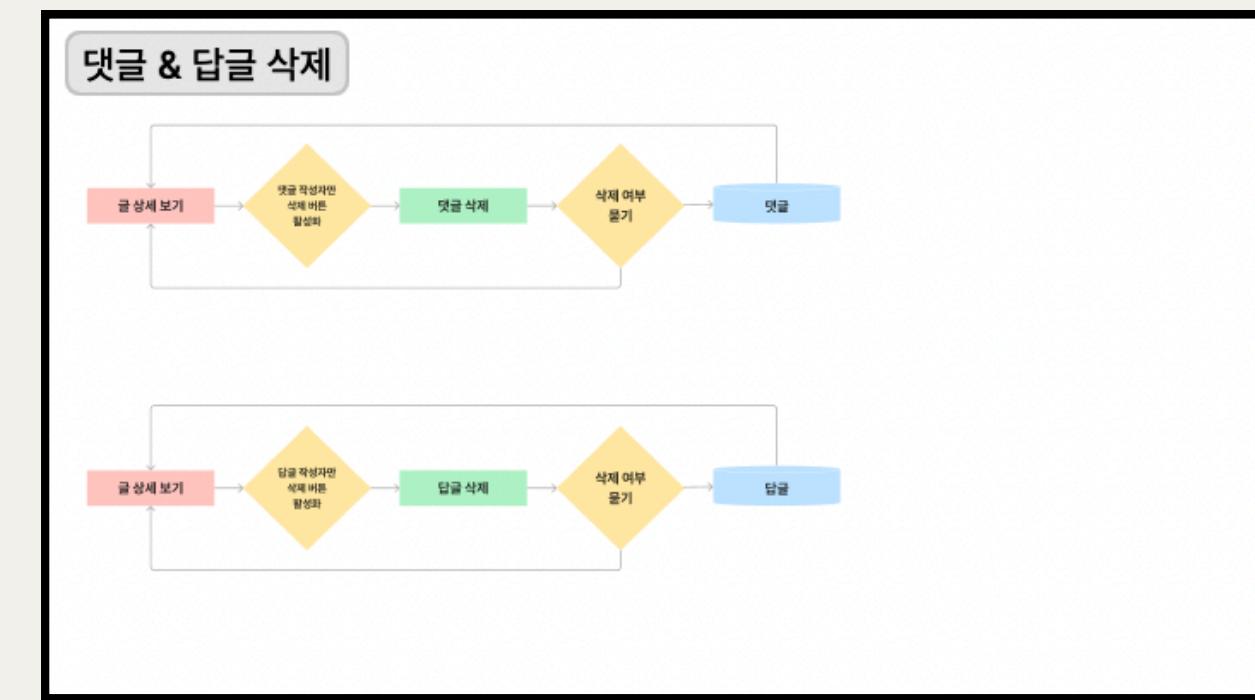
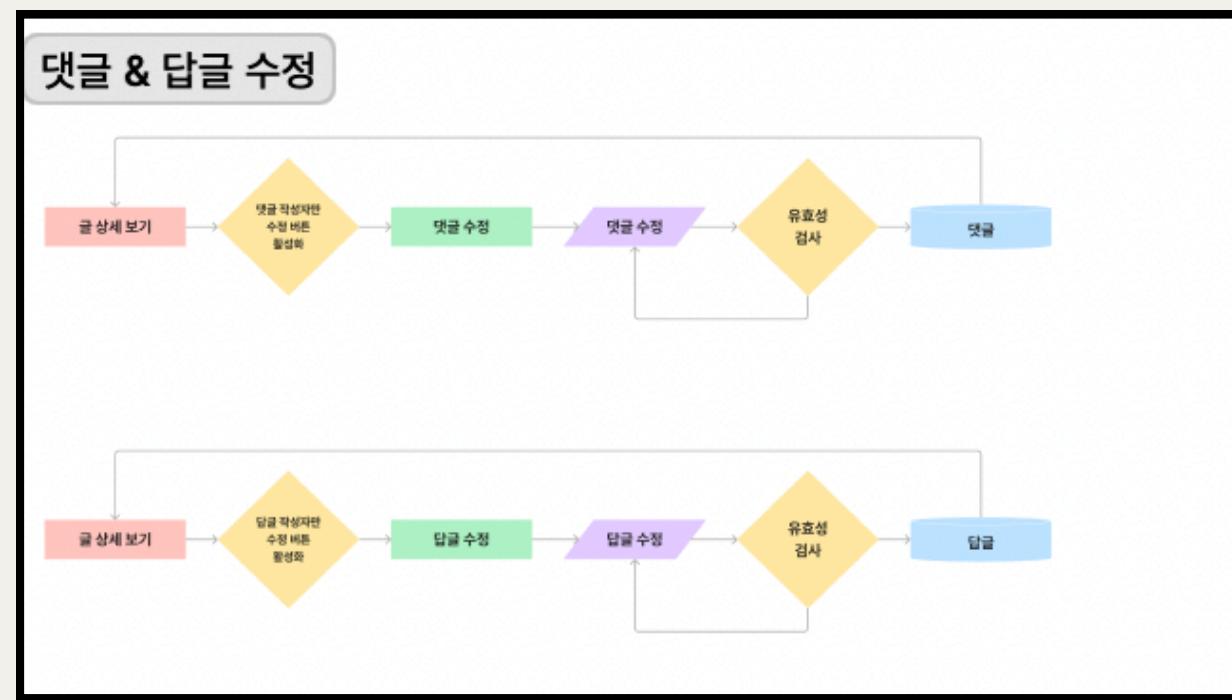
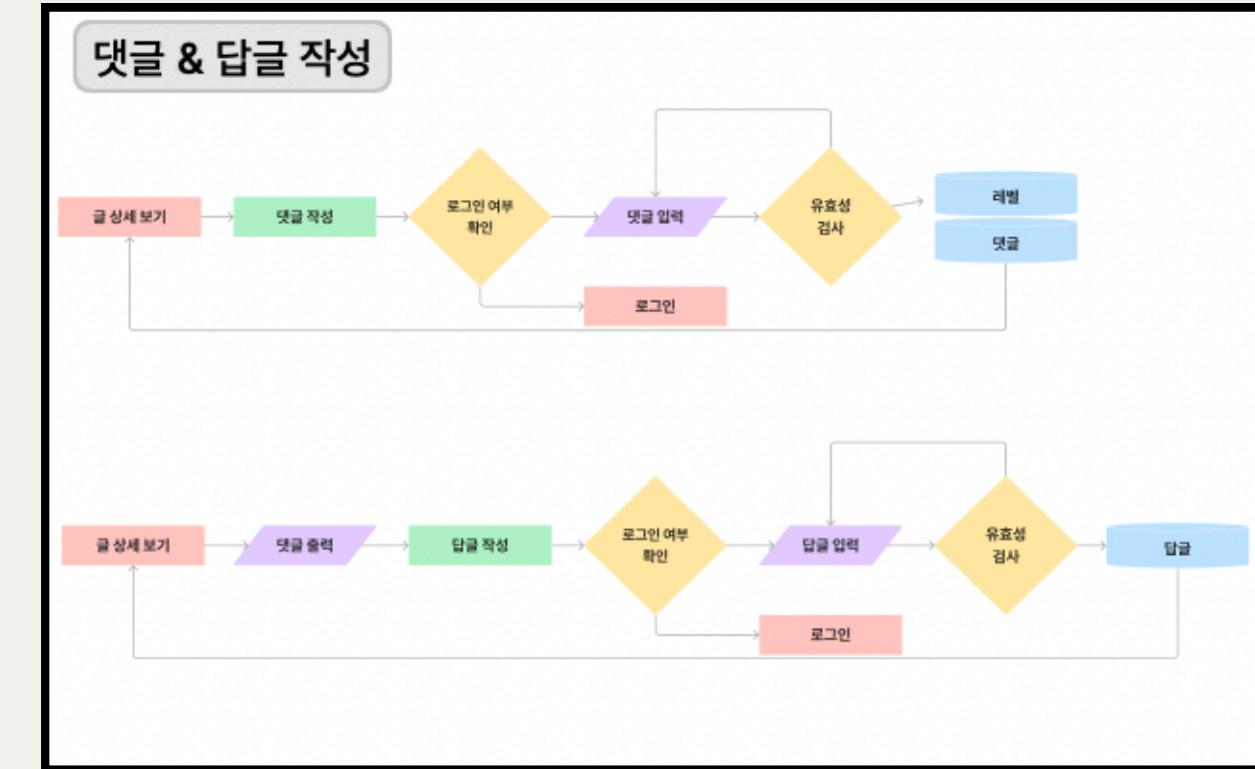
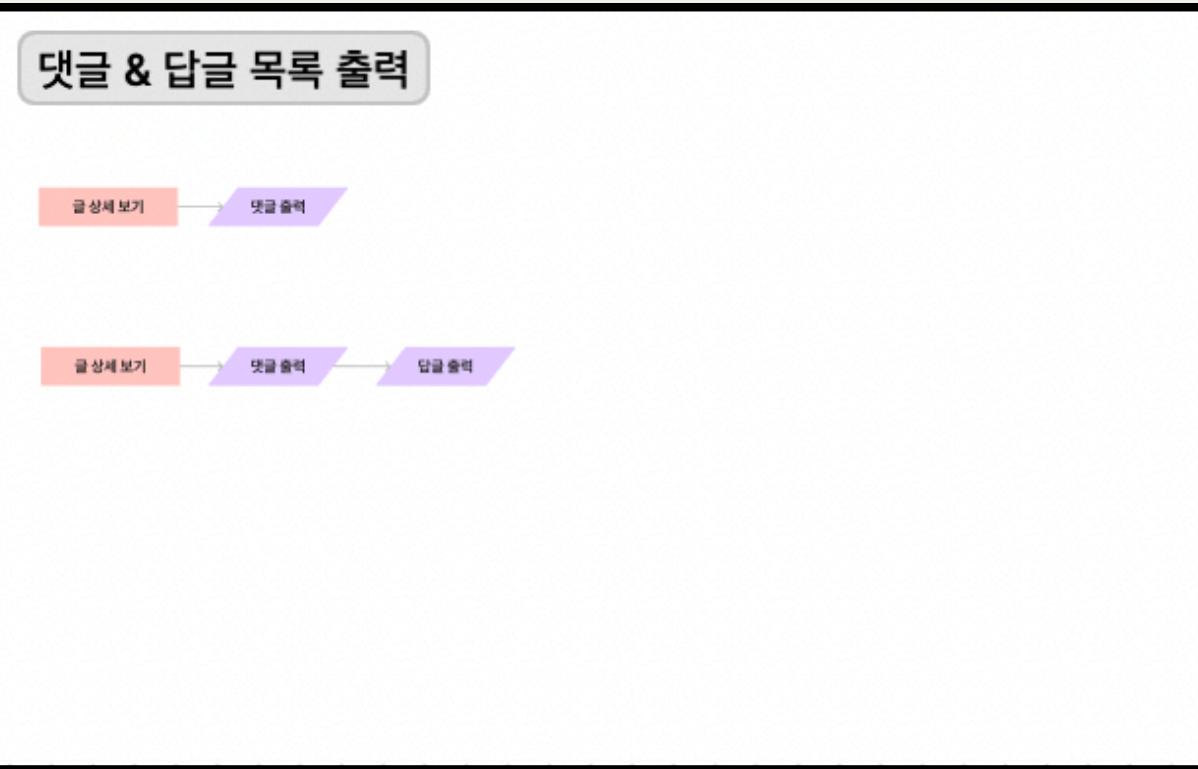


[프로젝트 설계 자세히 보기](#)

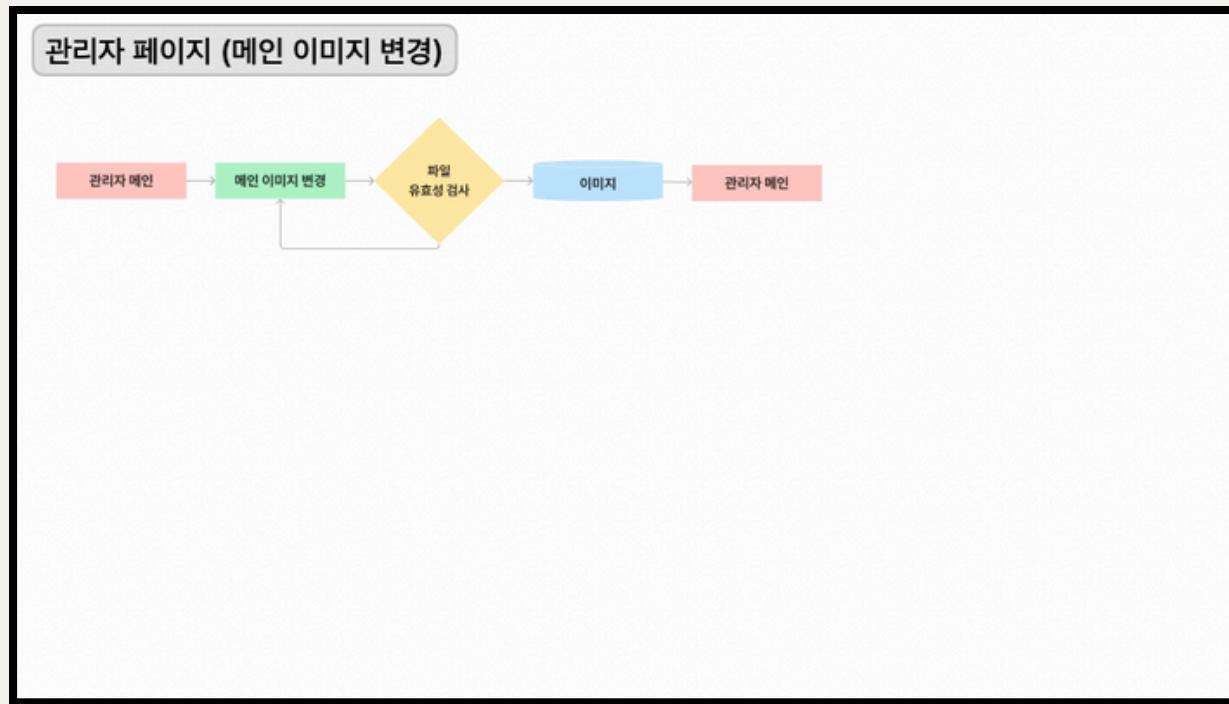
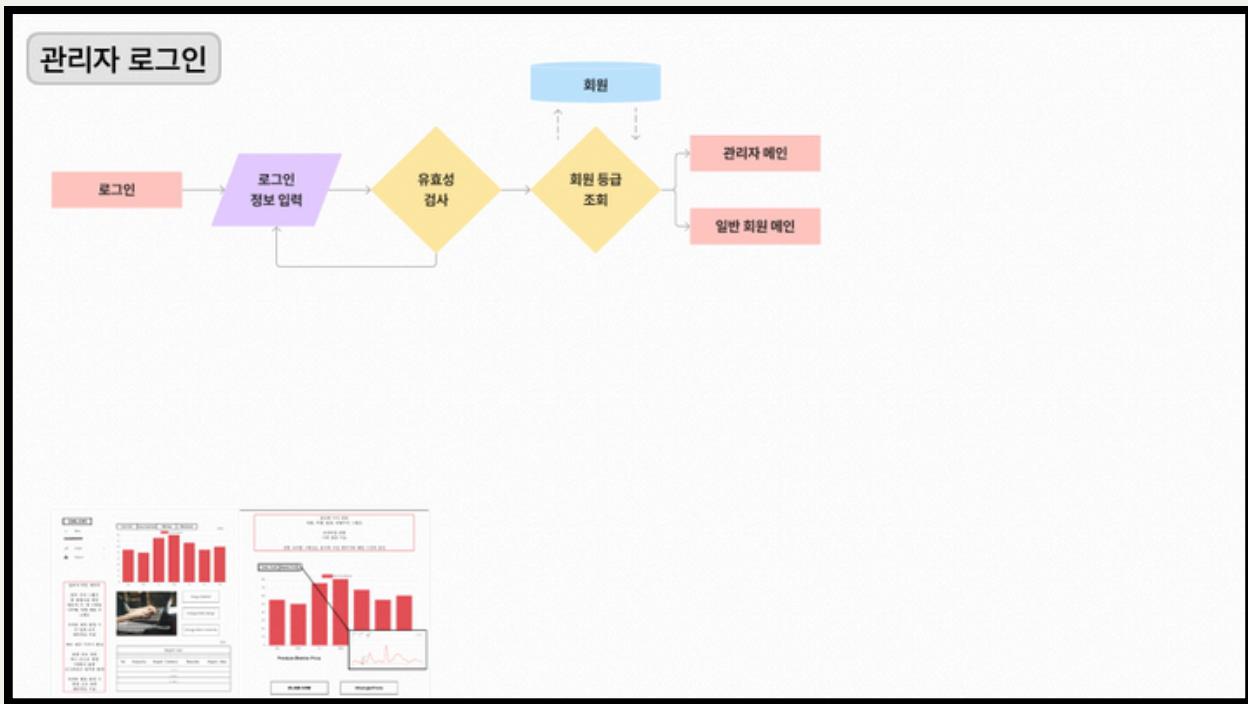
Logic Process - 게시글



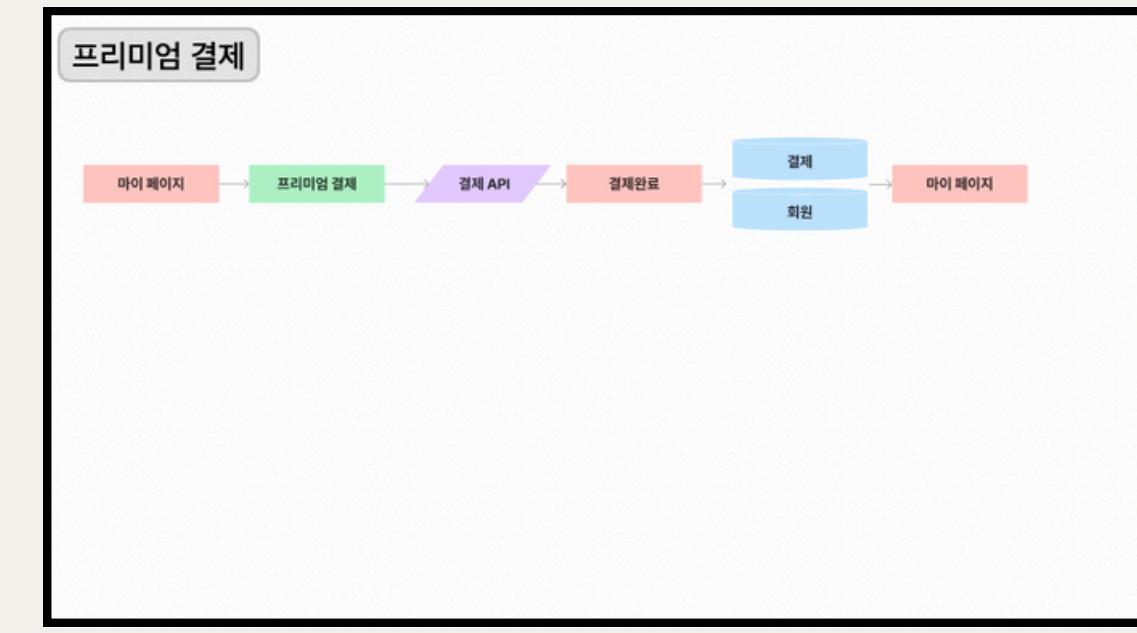
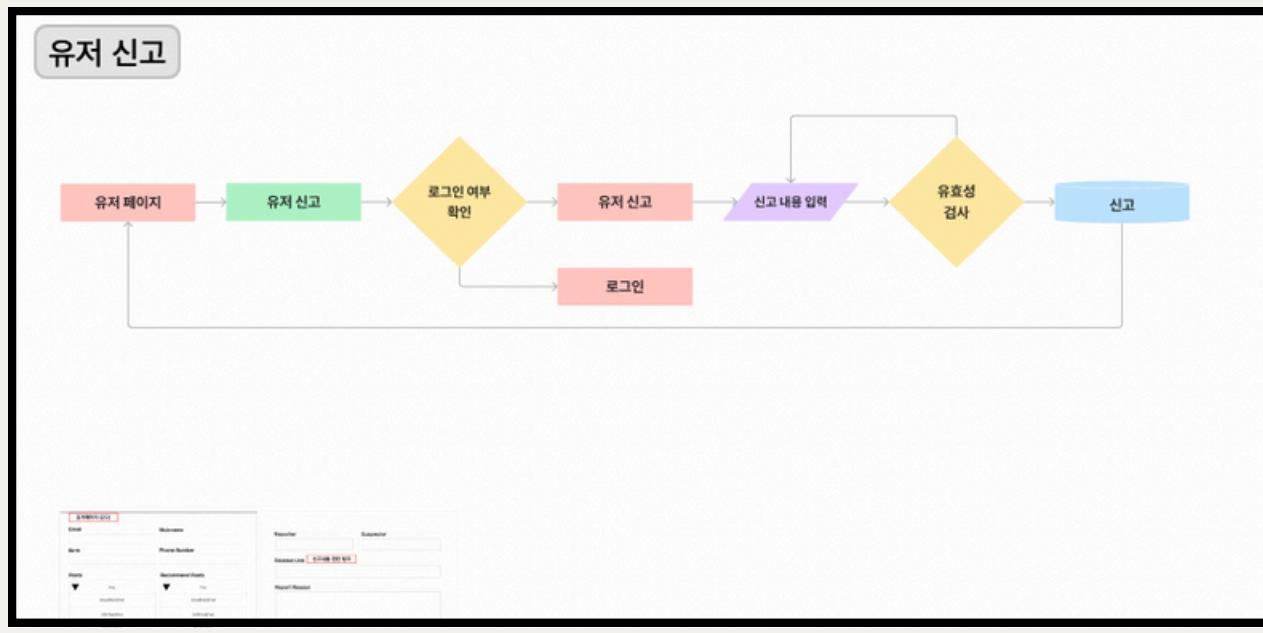
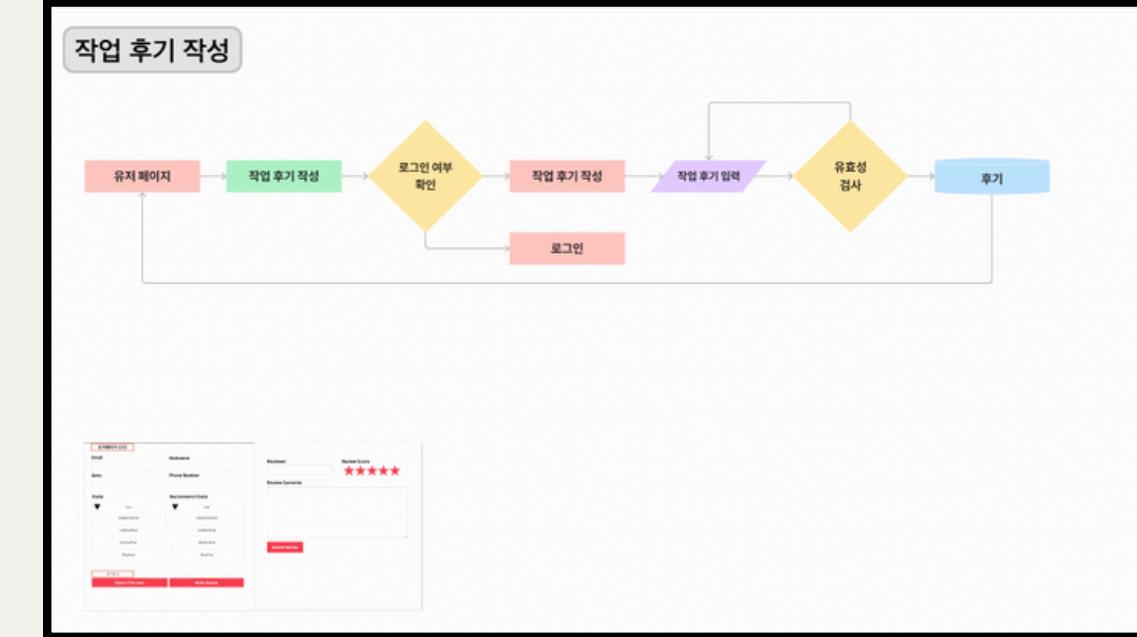
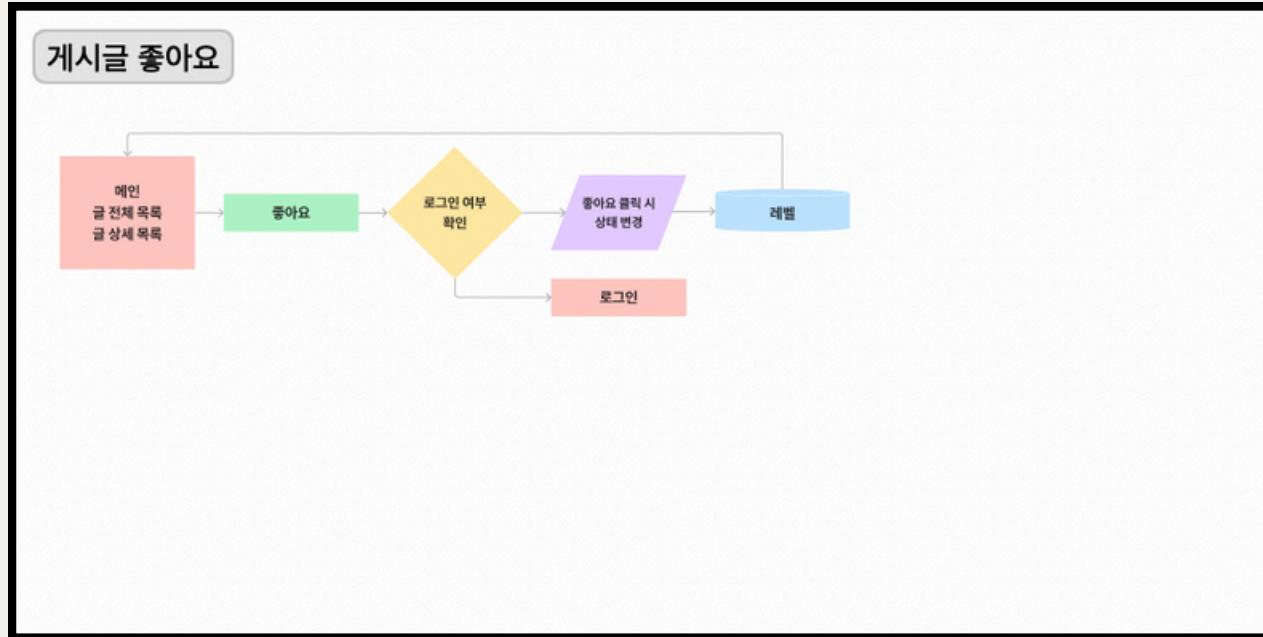
Logic Process - 댓글



Logic Process - 관리자



Logic Process - 유저



주요 기능 시연 및 코드 설명

Spring Security 를 활용한 소셜 로그인 (1/6)

signIn.jsp



SecurityConfig.java

```
@Bean  
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
  
    http  
        .csrf((csrf) -> csrf.disable());  
  
    http  
        .formLogin((auth) -> auth.disable());  
  
    http  
        .httpBasic((basic) -> basic.disable());  
  
    http  
        .oauth2Login((oauth2) -> oauth2  
            .userInfoEndpoint(userInfoEndpointConfig) -> userInfoEndpointConfig  
                .userService(customOAuth2UserService)  
                .defaultSuccessUrl("/oauth2SignIn"));  
  
    http  
        .authorizeHttpRequests((auth) -> auth  
            .anyRequest().permitAll());
```

filterChain 메서드를 사용해
SecurityFilterChain 빈을 정의
사용자가 요청한 URI에 oauth2 가 포함되면
customOAuth2UserService 로 이동

Spring Security 를 활용한 소셜 로그인 (2/6)

CustomOAuth2UserService.java

```
@Service  
public class CustomOAuth2UserService extends DefaultOAuth2UserService {  
    //DefaultOAuth2UserService OAuth2UserService의 구현체  
  
    private final UserRepository userRepository;  
  
    public CustomOAuth2UserService(UserRepository userRepository) {  
  
        this.userRepository = userRepository;  
    }
```

OAuth2UserService 의 구현체인
DefaultOAuth2UserService 를 확장해
커스텀 OAuth 2.0 사용자 서비스를 정의

```
public interface UserRepository extends JpaRepository<UserEntity, String> {  
    UserEntity findByEmail(String member_id);  
}
```

Spring Data JPA 를 사용해 DB 와 상호 작용하는 데
사용되는 사용자 리포지토리를 정의

Spring Security 를 활용한 소셜 로그인 (3/6)

CustomOAuth2UserService.java

```
@Override  
public OAuth2User loadUser(OAuth2UserRequest userRequest) throws OAuth2AuthenticationException {  
    OAuth2User oAuth2User = super.loadUser(userRequest);  
    System.out.println(oAuth2User.getAttributes());  
  
    String registrationId = userRequest.getClientRegistration().getRegistrationId();  
    OAuth2Response oAuth2Response = null;  
    if (registrationId.equals("naver")) {  
        oAuth2Response = new NaverResponse(oAuth2User.getAttributes());  
    }  
    else if (registrationId.equals("google")) {  
        oAuth2Response = new GoogleResponse(oAuth2User.getAttributes());  
    }  
    else if (registrationId.equals("kakao")) {  
        oAuth2Response = new KakaoResponse(oAuth2User.getAttributes());  
    }  
    else {  
        return null;  
    }  
}
```

NaverResponse.java

```
@Override  
public String getPh() {  
    return attribute.get("mobile").toString().replace("-", "");  
}
```

KakaoResponse.java

```
@Override  
public String getPh() {  
    String originalNumber = ((Map<String, Object>) attribute.get("kakao_account")).  
    String formattedNumber = formatPhoneNumber(originalNumber);  
    return formattedNumber;  
}  
  
public static String formatPhoneNumber(String phoneNumber) {  
    // 전화번호에서 숫자만 남기기  
    String cleanedNumber = phoneNumber.replaceAll("[^0-9]", "");  
    // 010 형식으로 재구성  
    String formattedNumber = "010" + cleanedNumber.substring(cleanedNumber.length() - 8);  
    return formattedNumber;  
}
```

부모 클래스의 loadUser 메서드를 실행
OAuth2 공급자로부터 사용자 정보를 가져옴
일반 사용자와 같은 테이블에서 관리하기 때문에
DB 의 저장 양식에 맞춰서 속성 값 반환

Spring Security 를 활용한 소셜 로그인 (4/6)

CustomOAuth2UserService.java

```
String memberEmail= oAuth2Response.getEmail();
UserEntity existData = userRepository.findByEmail(memberEmail);
String role = "ROLE_USER";
if (existData == null) {

    UserEntity userEntity = new UserEntity();

    UUID uuid = UUID.randomUUID();
    String temporaryPassword = uuid.toString().substring(0, 8);
    System.out.println("temporaryPassword 확인 ["+temporaryPassword+"]");

    userEntity.setUsername(oAuth2Response.getName());
    userEntity.setBirth(oAuth2Response.getBirthYear()+oAuth2Response.getBirthDay());
    userEntity.setEmail(oAuth2Response.getEmail());
    userEntity.setNickname(oAuth2Response.getNickname());
    userEntity.setPh(oAuth2Response.getPh());
    userEntity.setPw(temporaryPassword);
    userEntity.setGender(oAuth2Response.getGender());
    userEntity.setRole(role);

    userRepository.save(userEntity);
}

return new CustomOAuth2User(oAuth2Response, role);
```

oAuth2Response.getEmail() 을 사용해
인증된 사용자의 이메일을 가져옴

userRepository 의 findEmail 메서드를 활용
해당 이메일을 가진 사용자가 있는지 확인 후
존재하지 않는다면 새로운 UserEntity 를 생성

비밀번호는 랜덤 UUID 를 생성 후 저장
CustomOAuth2User 객체를 생성하고 이를 반환

```
public class CustomOAuth2User implements OAuth2User {
    private final OAuth2Response oAuth2Response;
    private final String role;

    public CustomOAuth2User(OAuth2Response oAuth2Response, String role) {
        this.oAuth2Response = oAuth2Response;
        this.role = role;
    }
}
```

OAuth2User 인터페이스를 구현

Spring Security 를 활용한 소셜 로그인 (5/6)

UserEntity.java

```
@Entity  
@Getter  
@Setter  
@Table(name = "MEMBER")  
public class UserEntity {  
  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    @Transient  
    private String id;  
  
    @Transient  
    private String role;  
  
    @Column(name="MEMBER_name")  
    private String username;  
  
    @Id  
    @Column(name="MEMBER_id")  
    private String email;  
  
    @Column(name="MEMBER_pw")  
    private String pw;  
  
    @Column(name="MEMBER_nickname")  
    private String nickname;  
  
    @Column(name="MEMBER_ph")  
    private String ph;  
  
    @Column(name="MEMBER_birth")  
    private String birth;  
  
    @Column(name="MEMBER_gender")  
    private String gender;  
  
}
```

member
ABC MEMBER_id
ABC MEMBER_pw
ABC MEMBER_name
ABC MEMBER_nickname
ABC MEMBER_ph
ABC MEMBER_birth
ABC MEMBER_gender
ABC MEMBER_introduction
ABC MEMBER_grade
123 MEMBER_exp
⌚ MEMBER_signupdate

@Table 어노테이션 활용
UserEntity 클래스를 MEMBER 테이블과 맵핑

@Column 어노테이션 활용
Entity 클래스의 필드를 테이블의 컬럼과 맵핑

@Id 어노테이션 활용
email 필드를 엔티티의 기본키로 설정

Spring Security 를 활용한 소셜 로그인 (6/6)

SecurityConfig.java

```
http
    .oauth2Login((oauth2) -> oauth2
        .userInfoEndpoint((userInfoEndpointConfig) -> userInfoEndpointConfig
            .userService(customOAuth2UserService))
        .defaultSuccessUrl("/oauth2SignIn"));
```

SecurityContextHolder 를 이용
현재 사용자의 인증 정보에 접근

사용자의 정보 중 이메일을 활용
MemberService 의 selectOne 메서드 호출
session 에 사용자의 정보를 저장

MemberDAO.java

```
else if (memberDTO.getSearchCondition().equals("OAuth2SignIn")) { // 소셜 로그인
    Object[] args = {memberDTO.getMemberId()};
    result = jdbcTemplate.queryForObject(SELECTONE_OAUTH2SIGNIN, args, new OAuth2SignInRowMapper());
    System.out.println("MemberDAO(selectOne) Out로그 = [" + result + "]");
    return result;
}
```

```
// 소셜 로그인
private static final String SELECTONE_OAUTH2SIGNIN = "SELECT MEMBER_id, "
    "MEMBER_grade " +
    "FROM MEMBER " +
    "WHERE MEMBER_id = ?";
```

OAuth2SignInController.java

```
@Controller
public class OAuth2SignInController {

    @Autowired
    private MemberService memberService;

    @Autowired
    private SignInLogService signInLogService;

    @RequestMapping("/oauth2SignIn")
    public String myEndpoint(HttpSession session, MemberDTO memberDTO, SignInLogDTO signInLogDTO) {
        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

        // 인증된 사용자가 CustomOAuth2User 객체인지 확인
        if (authentication.getPrincipal() instanceof CustomOAuth2User) {
            // CustomOAuth2User 객체로 캐스팅하여 사용자 정보에 접근
            CustomOAuth2User customOAuth2User = (CustomOAuth2User) authentication.getPrincipal();

            // 사용자 정보 활용
            String email = customOAuth2User.getEmail();

            // 사용자 정보 반환
            System.out.println("[OAuth2SignInController] email : " + email);

            memberDTO.setSearchCondition("OAuth2SignIn");
            memberDTO.setMemberId(email);

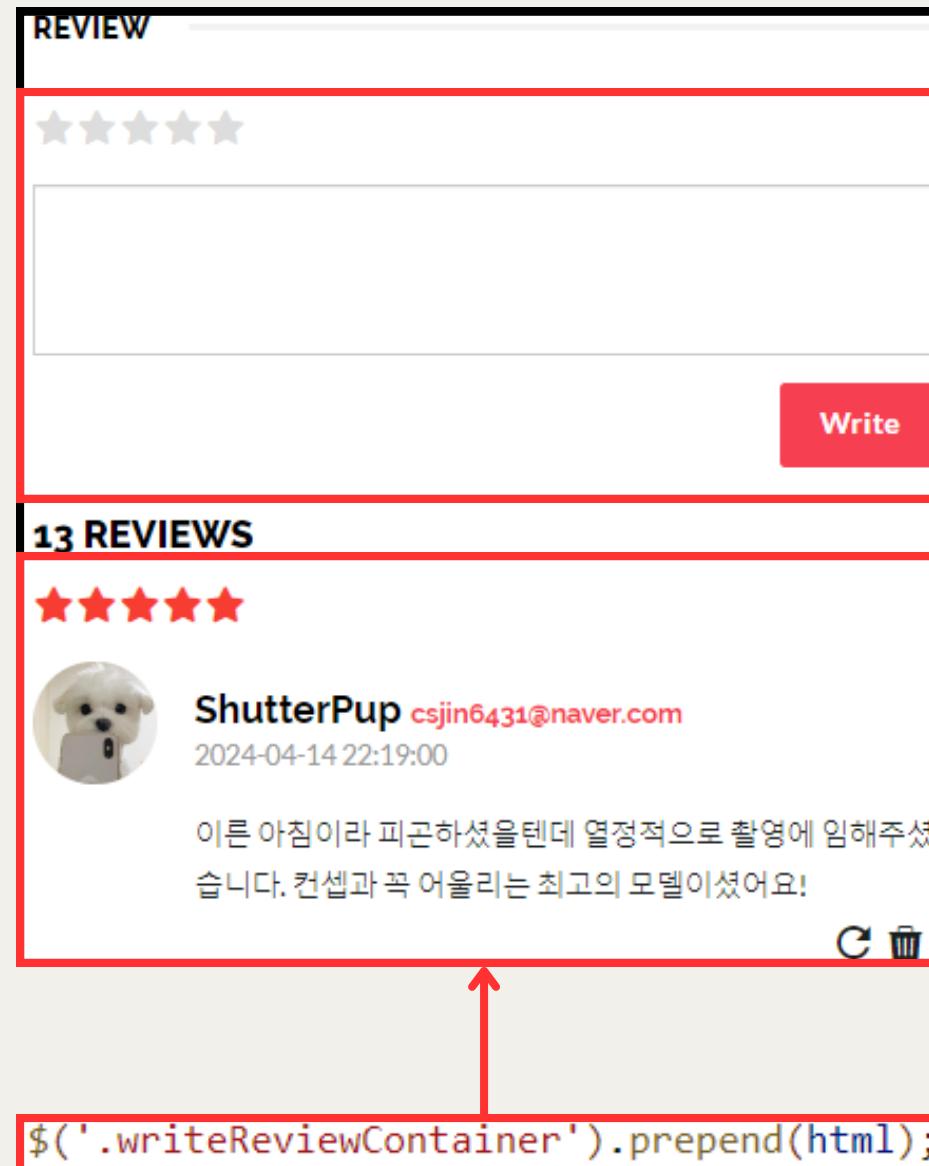
            // 기존에 있는 멤버 서비스를 활용하여 ID와 해당 유저의 등급을 세션에 저장
            MemberDTO result = memberService.selectOne(memberDTO);

            signInLogDTO.setMemberId(result.getMemberId());
            signInLogService.insert(signInLogDTO);

            session.setAttribute("member", result.getMemberId());
            session.setAttribute("memberGrade", result.getMemberGrade());
        }
        return "redirect:/main";
    }
}
```

후기 작성 (1/3)

memberPage.jsp



writeReview.js

```
$.ajax({
  type: 'POST',
  url: '/writeReview',
  data: $('#writeReviewForm').serialize(), // 폼 데이터 전송
  datatype: 'json',
  success: function (reviewData) {
    if (reviewData) {
      $('input[name^="reviewScore"]').prop('checked', false); // 별점 초기화
      $('textarea[name="reviewContent"]').val(''); // 리뷰 내용 지우기
      $('#emptyReview').text("");
    }
  }
});
```

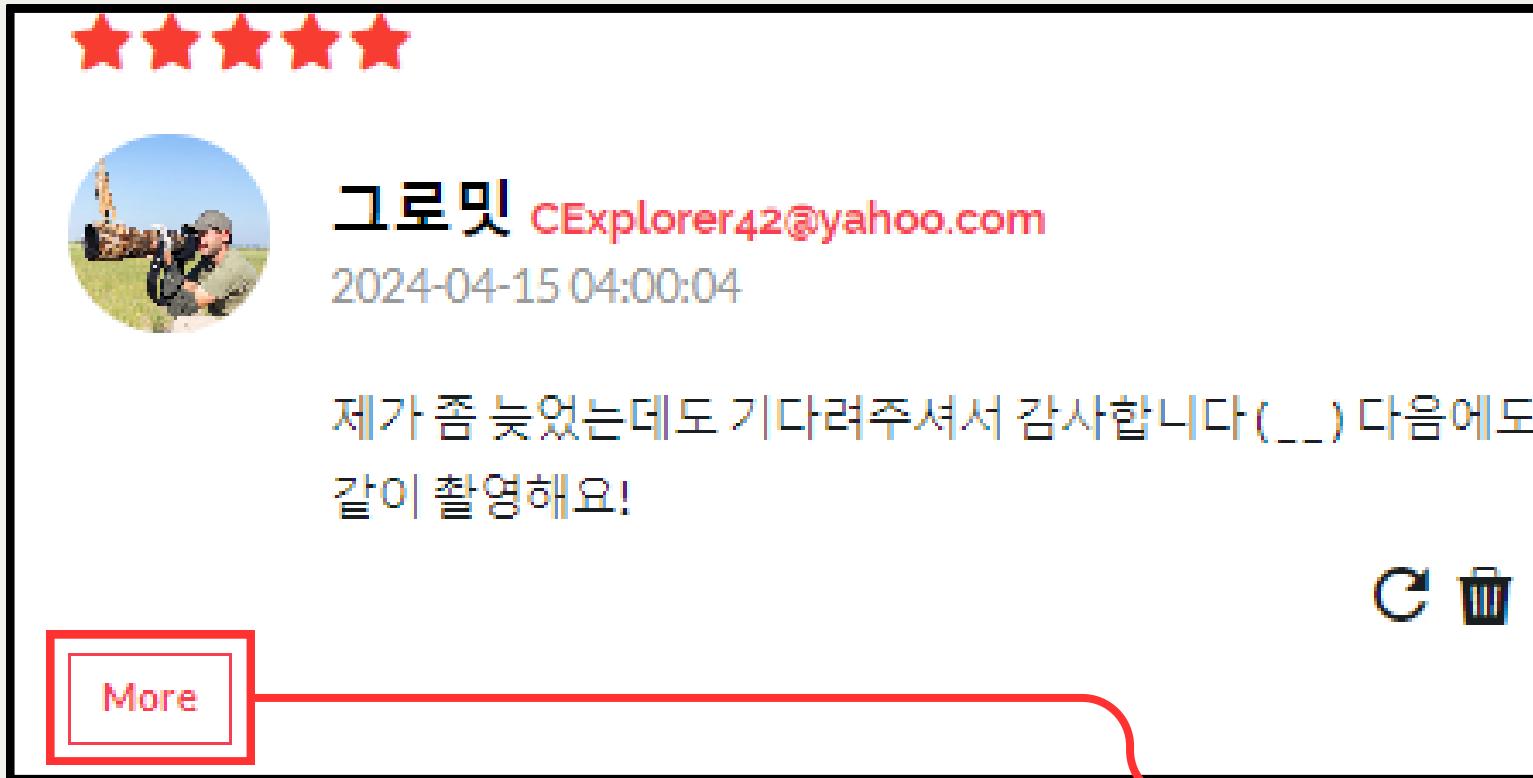
WriteReviewController.java

```
@RequestMapping(value = "/writeReview")
public @ResponseBody String writeReview(ReviewDTO reviewDTO, HttpSession session) {
  System.out.println("WriteController In로그 = [" + reviewDTO + "]");
  reviewDTO.setMemberId((String) session.getAttribute("member"));
  reviewService.insert(reviewDTO);
  return gson.toJson(reviewService.selectOne(reviewDTO));
}
```

사용자에게 입력받은 데이터를 ajax 를 사용하여
비동기로 Controller 에 요청하고 DB 에 저장한 후
저장된 데이터를 즉시 응답으로 받아 화면에 출력

후기 작성 (2/3)

memberPage.jsp



더보기 클릭 시 삭제된 후기 개수를 고려하여
가져올 데이터 시작지점을 계산하고
ajax를 통해 Controller로 필요한 데이터를
전달하며 비동기 요청

moreReview.js

```
var deletedCnt = 0;
var reviewStart = 10; // 초기값 설정

$.ajax({
    url: "/deleteReview",
    success: function (response) {
        deletedCnt++;
        //삭제된 데이터가 있을경우 고려해서 다음 10개를 가져옴
        reviewStart -= deletedCnt;
        $.ajax({
            type: 'GET',
            url: '/moreReview',
            dataType: 'json',
            data: {
                reviewPartner: $('#reviewPartner').val(),
                reviewStart: reviewStart
            }
        })
    }
})
```

후기 작성 (3/3)

UpdateReviewController.java

```
@RequestMapping(value = "moreReview")
public @ResponseBody String moreReview(ReviewDTO reviewDTO) {
    System.out.println("UpdatereviewController In로그 [" + reviewDTO + "]");
    reviewDTO.setReviewPartner(reviewDTO.getReviewPartner());
    // 가져올 데이터 시작지점
    reviewDTO.setReviewStart(reviewDTO.getReviewStart());
    // 가져올 데이터 개수
    reviewDTO.setReviewCnt(10);

    return gson.toJson(reviewService.selectAll(reviewDTO));
}
```

View 로 부터 전달받은 데이터를 바탕으로
더보기를 클릭 시 화면에 출력하는
후기글을 쿼리로 추출하여 View 로 응답

ReviewDAO.java

```
private static final String SELECTALL = "SELECT REVIEW_score, "
        "(SELECT PROFILEIMG_name " +
        "FROM PROFILEIMG " +
        "WHERE PROFILEIMG.MEMBER_id = REVIEW.MEMBER_id " +
        "ORDER BY PROFILEIMG_id DESC " +
        "LIMIT 1) AS PROFILEIMG_name, " +
        "(SELECT COUNT(*) " +
        "FROM REVIEW " +
        "WHERE REVIEW_partner = ?) AS REVIEW_totalCnt, " +
        "REVIEW.MEMBER_id, " +
        "MEMBER.nickname, " +
        "REVIEW_partner, " +
        "REVIEW_content, " +
        "REVIEW_date, " +
        "REVIEW_id " +
        "FROM REVIEW " +
        "INNER JOIN MEMBER ON REVIEW.MEMBER_id = MEMBER.MEMBER_id " +
        "WHERE REVIEW_partner = ? " +
        "ORDER BY REVIEW_id DESC " +
        "limit ?, ?";
```

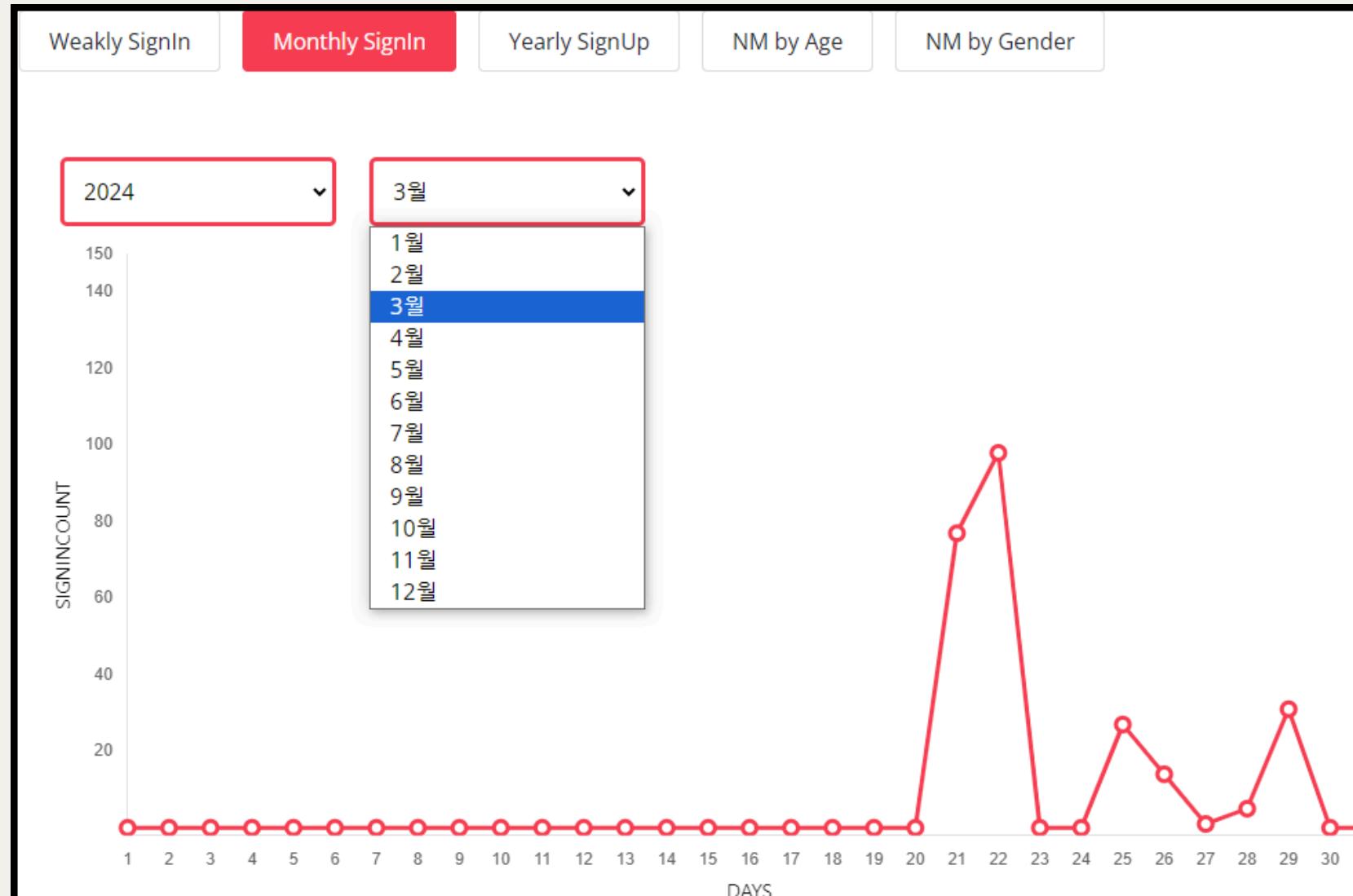
moreReview.js

```
// 다음 10개           // 삭제된 개수 초기화
reviewStart += 10;      deletedCnt = 0;
```

후기글을 추가적으로 출력 후 다음
더보기를 위한 변수값 변경

그래프 출력 (1/2)

adminMain.jsp



drawChart.js

```
// 바 차트 그리기
function drawBarChart(chartId, labels, data, xTitleLabel, yTitleLabel, ticksOptions) {
    // 라인 차트 그리기
    function drawLineChart(chartId, labels, data, max, xTitleLabel, yTitleLabel) {
        handleSelectChange();
        // 년, 월 선택시 그래프 변화
        function handleSelectChange() {
            const selectedYear = document.getElementById('year').value;
            const selectedMonth = document.getElementById('month').value;

            $.ajax({
                type: "GET",
                url: "/signInCountByYearMonthDate"
            })
            .done(function(response) {
                const signInCounts = response.data;
                const chartData = signInCounts.map(item => ({ day: item.date, count: item.signInCount }));
                drawLineChart(chartId, labels, chartData, max, xTitleLabel, yTitleLabel);
            })
            .fail(function(error) {
                console.error("AJAX error: " + error);
            });
        }
        document.getElementById('year').addEventListener('change', handleSelectChange);
        document.getElementById('month').addEventListener('change', handleSelectChange);
    }
}
```

사용자가 값을 선택할 때마다 새로운 데이터를 가져오기 위해
서버로 비동기 요청 진행

그래프 출력 (2/2)

AdminDAO.java

```
// 관리자) 월 날짜별 로그인 수. 안승준
private static final String SELECTALL_SIGNINCOUNTBYYEARMONTHDATE = "WITH RECURSIVE DATERANGE AS ( " +
    "SELECT DATE_FORMAT(CONCAT(? , '-' , LPAD(?, 2, '0') , '-01') , '%Y-%m-%d') AS DATE " +
    "UNION ALL " +
    "SELECT DATE_ADD(DATE, INTERVAL 1 DAY) " +
    "FROM DATERANGE " +
    "WHERE DATE < LAST_DAY(DATE) " +
    ")" +
    "SELECT " +
    "DAY(DATERANGE.DATE) AS DATE, " +
    "COALESCE(COUNT(MEMBER.MEMBER_id), 0) AS SIGNINCOUNT " +
    "FROM DATERANGE " +
    "LEFT JOIN SIGNINLOG ON DATE(SIGNINLOG.SIGNINLOG_date) = DATERANGE.DATE " +
    "LEFT JOIN MEMBER ON SIGNINLOG.MEMBER_id = MEMBER.MEMBER_id AND MEMBER.MEMBER_grade != 'ADMIN' " +
    "GROUP BY DATE " +
    "ORDER BY DATE";
```

재귀 쿼리 선언
임시테이블(자기자신)
참조해서 반복
반복 종료 조건

입력 받은 년도와 월에 따라서 해당 월의 마지막일까지 날짜를
생성하는 임시 테이블을 DATARANGE로 선언

DATARANGE 테이블의 날짜와 SIGNINLOG 테이블의 날짜가
같은 데이터를 카운트해서 선택

DATE	SIGNINCOUNT
21	1
22	0
23	63
24	78
25	11
26	22
27	7
28	0
29	0
30	12
31	20

신고 관리 (1/3)

adminReportList.jsp

	No	Reporter
<input checked="" type="checkbox"/>	1	CExplorer@yahoo.com
<input checked="" type="checkbox"/>	2	CExplorer@yahoo.com
<input checked="" type="checkbox"/>	3	CExplorer@yahoo.com
<input checked="" type="checkbox"/>	4	csjin@naver.com
<input checked="" type="checkbox"/>	5	csjin@naver.com
<input checked="" type="checkbox"/>	6	csjin@naver.com
<input checked="" type="checkbox"/>	7	csjin@naver.com

reportStateUpdate.js

```
// 모든 체크된 체크박스 요소를 선택하고 각각의 값을 배열에 추가
$('input[type=checkbox][name=reportId]:checked').each(function () {
    selectedReportIds.push($(this).val());
});

→ selectedStateUpdate?selectedReportIds=
```

AdminReportStateController.java

```
@RequestMapping(value = "/selectedStateUpdate",
// 선택된 신고 ID가 있는 경우, SQL 쿼리를 생성
if (reportIdList.size() > 0) {
    for (int i = 0; i < reportIdList.size(); i++) {
        reportIdSb.append("\'" + reportIdList.get(i) + "\'");
        if (i + 1 < reportIdList.size()) {
            reportIdSb.append(",");
        }
    }
    SELECTEDSTATEUPDATESQL = "(" + reportIdSb.toString() + ")";
```

```
// click 이벤트 리스너 제거
$checkboxTh.off('click.DT');

// keypress 이벤트 리스너 제거
$checkboxTh.off('keypress.DT');

// selectstart 이벤트 리스너 제거
$checkboxTh.off('selectstart.DT');
```

ReportDAO.java

```
private static final String UPDATE_SELECTEDSTATEDELETED = "UPDATE REPORT "
    + "SET "
    + "    REPORT_state = 'DELETED' "
    + "WHERE "
    + "    REPORT_id IN ";
UPDATE_SELECTEDSTATEDELETED + reportDTO.getSelectedStateUpdateSQL()
```

템플릿 테이블의 DOM 을 조작해서 체크박스 컬럼을 제작

현재 화면에
체크된 체크박스들의 ID 를
배열에 저장해
Controller 로 전달하고,
체크된 신고 내역들이
전부 삭제될 수 있도록
필요한 쿼리를 제작해서
전체 삭제

신고 관리 (2/3)

adminReportSingle.jsp

Report Details	
Date	2024-04-15 16:13:06
Reporter	CExplorer@yahoo.com
Suspector Email	ewz1125@gmail.com
Suspector Nickname	월레스
Textarea	계속 게시판에 도배합니다. 제재해 주세요.
Hold Period	<input type="radio"/> 1 day <input type="radio"/> 7 days <input type="radio"/> 30 days <input checked="" type="radio"/> 90 days
User Hold Reject	

reportStateUpdate.js

```

// 신고를 통해 이미 정지된 회원인지 확인
$.ajax({
  type: 'POST',
  url: '/reportStateCheck',
  data: $('#reportStateHoldForm').serialize(), // 폼 데이터 전송
  success: function (result) {
    if (result === 1) {
      // 기존에 정지된 회원이 아니면 정지상태로 변경
      reportStateHold();
    } else {
      swal("WARNING", "이미 정지된 사용자입니다.", "error",
        button: "OK"
    }
  }
});

```

정지 버튼을 누르면 ajax를 통해 비동기로 이미 정지된 회원인지 확인 후, 정지된 상태가 아니라면 정지 상태로 변경하기 위한 요청 진행

AdminReportStateController.java

```

// 정지된 회원인지 selectOne 으로 탐색 후 아닐경우 1 정지된 회원일 경우 0 반환
memberDTO.setSearchCondition("myPage");
MemberDTO result = memberService.selectOne(memberDTO);
if (result.getMemberGrade().equals("TIMEOUT")) {
  System.out.println("AdminReportListController Out로그");
  return 0;
}
System.out.println("AdminReportListController Out로그");
return 1;

```

신고 관리 (3/3)

reportStateUpdate.js

```

function reportStateHold() {
    // 폼 가져오기
    var form = document.getElementById('reportStateHoldForm');

    // 폼 제출
    form.submit();
}

<form id="reportStateHoldForm" action="reportStateHold" method="post"
      class="form-horizontal">
    <!-- reportState를 HOLD로 바꾸기 위한 hidden value -->
    <input type="hidden" id="reportId" name="reportId"
          value="${reportSingle.reportId}">

    <div class="row form-group">
        <div class="col col-md-3"><label
            class=" form-control-label">Date</label></div>
        <div class="col-12 col-md-9"><input type="text" id="text-input"
            name="text-input" class="form-control"
            value="${reportSingle.reportDate}" readonly>
        </div>
    </div>
    <div class="row form-group">
        <div class="col col-md-3"><label
            class=" form-control-label">Reporter</label></div>
        <div class="col-12 col-md-9"><input type="email" id="reporter"
            name="reporter" class="form-control"
            value="${reportSingle.memberId}" readonly>
        </div>
    </div>
    <div class="row form-group">
        <div class="col col-md-3"><strong
            class=" form-control-label">Suspector Email</Strong></div>
        <div class="col-12 col-md-9"><input type="email" id="memberId"
            name="memberId" class="form-control"
            value="${reportSingle.reportSuspector}"
            style="font-weight: bold;" readonly></div>
    </div>

```

AdminReportStateController.java

```

@RequestMapping(value = "/reportStateHold", method = RequestMethod.POST)
public String reportStateHold(TimeOutDTO timeOutDTO, MemberDTO memberDTO) {

    System.out.println("AdminReportListController In로그");
    // 회원 정지 정보 저장
    if (!timeOutService.insert(timeOutDTO)) {
        System.out.println("AdminTimeOutListController reportUpdate failed");
    }

    memberDTO.setSearchCondition("timeOut");
    // 회원의 등급 정지로 변경
    if (!memberService.update(memberDTO)) {
        System.out.println("AdminTimeOutListController memberUpdate failed");
    }

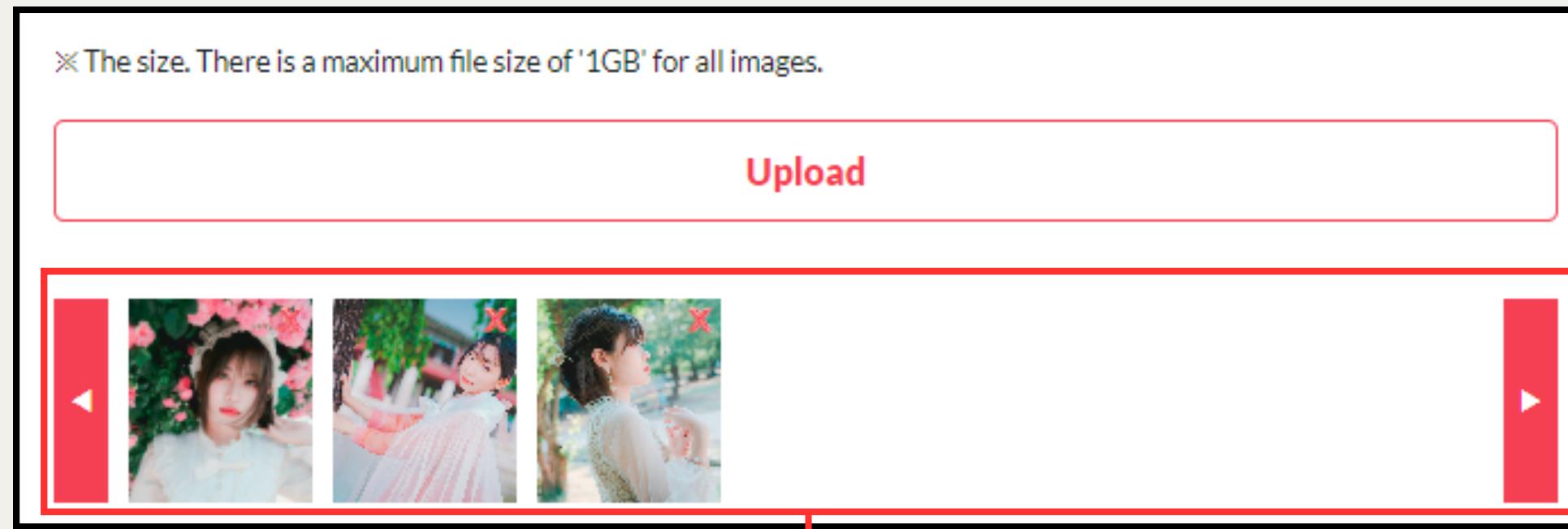
    System.out.println("AdminReportListController Out로그");

    return "redirect:adminReportList";
}

```

정지 테이블에 회원의 정보를 저장하고
회원의 등급을 정지로 변경

글 작성 시 이미지 업로드(1/3)



업로드한 이미지를 슬라이드 형식으로 미리보기

업로드 할 수 있도록 multipart 속성 설정

writeHeadHuntPost.jsp

```
<form action="/writeHeadHuntPost" method="post" onsubmit="return validateForm(event)" enctype="multipart/form-data">
```

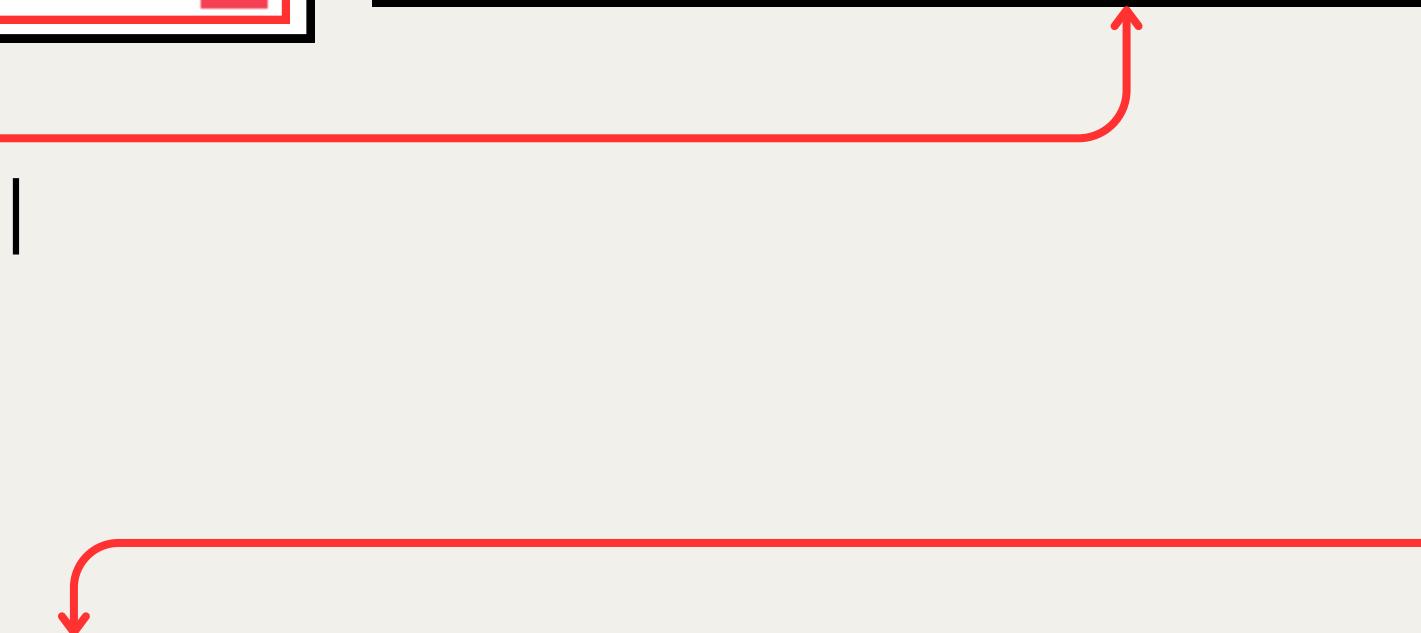
previewImg.js

```
let filesArray = []; // 이미지 업로드 빈 배열 설정

imgWrapper.appendChild(img); // 이미지 정보
imgWrapper.appendChild(deleteButton); // 이미지 삭제 버튼
imgWrapper.appendChild(upButton); // 이미지 위치 ◀ 버튼
imgWrapper.appendChild(downButton); // 이미지 위치 ▶ 버튼
imageContainer.appendChild(imgWrapper); // div 태그에 imgWrapper 를 삽입

// 선택된 파일을 filesArray에 추가
filesArray.push(file); // 배열에 파일 삽입

updateSliderButtonsVisibility(); // 슬라이드 버튼 가시성 업데이트
```



글 작성 시 이미지 업로드(2/3)

WriteHeadHuntPostController.java

```
String uploadDir = this.getClass().getResource("").getPath();
System.out.println("WriteHeadHuntPostController Log01 = [" + uploadDir + "]");

uploadDir = uploadDir.substring(1, uploadDir.indexOf("chalKag")) + "chalKag/src/main/resources/static/postImg";
System.out.println("WriteHeadHuntPostController Log02 = [" + uploadDir + "]");

// Add the user-written post
if (!headHuntPostService.insert(headHuntPostDTO)) {
    System.out.println("WriteHeadHuntPostController insertion of post failed");
}

headHuntPostDTO.setSearchCondition("maxPostId");
HeadHuntPostDTO maxPostIdDTO = headHuntPostService.selectOne(headHuntPostDTO);
headHuntPostDTO.setHeadHuntPostId(maxPostIdDTO.getHeadHuntPostId());

System.out.println(maxPostIdDTO.getHeadHuntPostId());

postImgDTO.setPostId(headHuntPostDTO.getHeadHuntPostId());

for (MultipartFile file : files) {
    if (file != null && !file.isEmpty()) {
        String originalFilename = file.getOriginalFilename();
        String extension = FilenameUtils.getExtension(originalFilename);
        String newFilename = UUID.randomUUID().toString() + "." + extension;
        String filePath = uploadDir + File.separator + newFilename;
        File newFile = new File(filePath);
        postImgDTO.setPostImgName(newFilename);
        if (!postImgService.insert(postImgDTO)) {
            System.out.println("WriteHeadHuntPostController insertion of image failed");
        }
        try {
            file.transferTo(newFile);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

이미지를 절대 경로에 저장하기 위해
저장 폴더의 경로 설정

이미지 테이블에 데이터 저장 시,
해당글의 번호를 가져와 같이 저장

여러 개의 이미지 파일을 저장하기에
반복문을 사용하여
다수의 이미지를 데이터를 저장
이때 저장되는 같은
이미지의 파일이 저장되는 것을
방지하기 위해
UUID 클래스로 파일명 변경 후 저장

글 작성 시 이미지 업로드(3/3)

HeadHuntPostDAO.java

```
private static final String INSERT_HEADHUNTPOST = "INSERT INTO HEADHUNTPOST ( "
    + "MEMBER_id, "
    + "HEADHUNTPOST_title, "
    + "HEADHUNTPOST_content, "
    + "HEADHUNTPOST_role, "
    + "HEADHUNTPOST_region, "
    + "HEADHUNTPOST_pay, "
    + "HEADHUNTPOST_workDate, "
    + "HEADHUNTPOST_concept, "
    + "HEADHUNTPOST_viewcnt "
    + ")VALUES(?,?,?,?,?,?,?,?,?,0) ";
// 사용한 테이블 : 구인글 테이블
// 사용한 컬럼 (작성 내용) : 회원 아이디(회원 테이블), 제목, 내용, 작성 시간, 직업, 작업 지역, 작업 페이, 작업 날짜, 촬영 컨셉, 조회수
// 구인글 아이디는 테이블 생성시 AUTO_INCREMENT를 사용해 2001번부터 자동 증감하게 설정
```

// 게시글 아이디 최대값 가져오는 쿼리문
private static final String SELECTONE_MAXPOSTID = "SELECT MAX(HEADHUNTPOST_id) FROM HEADHUNTPOST";

글을 먼저 저장 후 최대 값을 가져옴

PostImgDAO.java

```
private static final String INSERT = "INSERT INTO POSTIMG (POST_id, POSTIMG_name) VALUES (?, ?);
```

해당 글의 ID 값을 가져올 수 있도록 최대 값을 가져오는 쿼리문을 사용

글의 ID 와 파일명을 저장

댓글 목록 출력 (1/2)



사진찍는팬더 csjin6622@gmail.com

오~ 다음번엔 요기로 촬영하러 가봐야겠군요~!



Reply List ▼

JSTL 을 사용 댓글 파일을 생성 후 호출
~PostSingle.jsp

```
<%@ taglib tagdir="/WEB-INF/tags" prefix="chalKagTags" %>
```

```
<!-- 댓글 출력 -->  
<chalKagTags:webComments />
```

webComments.tag

~PostSingleController.java

```
commentDTO.setPostId(headHuntPostDTO.getHeadHuntpostId());  
List<CommentDTO> commentList = commentService.selectAll(commentDTO);  
model.addAttribute("commentList", commentList);
```

댓글 출력 시 해당 글의 댓글이 출력될 수 있도록
상세글 출력 Controller 에 댓글 출력 서비스를 호출

로그인한 회원이 댓글 작성자일 경우 수정, 삭제, 답글 작성 버튼 출력

댓글 목록 출력 (2/2)

CommentDAO.java

```
// 댓글 전체 출력 댓글, 게시판, 회원 조인 쿼리문
private static final String SELECTALL = "SELECT "
    + " COMMENT.COMMENT_id, "
    + " COMMENT.POST_id, "
    + " COMMENT.MEMBER_id, "
    + " MEMBER.MEMBER_nickname, "
    + " COMMENT.COMMENT_date, "
    + " COMMENT.COMMENT_content, "
    + " (
        SELECT "
        + " PROFILEIMG.PROFILEIMG_name "
        + " FROM "
        + " PROFILEIMG "
        + " WHERE "
        + " PROFILEIMG.MEMBER_id = COMMENT.MEMBER_id "
        + " ORDER BY "
        + " PROFILEIMG.PROFILEIMG_id DESC "
        + " LIMIT 1 "
        + " ) AS PROFILEIMG_name "
    + " FROM "
    + " COMMENT "
    + " INNER JOIN "
    + " MEMBER ON COMMENT.MEMBER_id = MEMBER.MEMBER_id "
    + " WHERE "
    + " COMMENT.POST_id = ?";
```

댓글 출력 시
작성자의 프로필 사진을
가져올 수 있도록
프로필 테이블을 서브 쿼리로 사용

이때 한 개의 프로필 이미지가
출력할 수 있도록 LIMIT 1 작성

회원 정보는 Inner Join 으로
회원 정보를 같이 전달

댓글 작성 (1/2)

The screenshot shows a web form for writing a comment. At the top right is a button labeled "Write a comment". Below it is a text area labeled "Response *". Inside the text area, there is placeholder text "Write your response ...". At the bottom right of the text area is a red button labeled "Send Response".

webComments.tag

```
$.ajax({
    type: "POST",
    url: "/writeComment",
    data: formData, ←
    dataType: "json", // 서버로부터 응답을 JSON으로 받음
    success: function(response) {
        history.go(0);
    },
});
```

댓글 작성 시 ajax 로직으로 서버에 전달

이때 해당 글의 ID 값을 전달할 수 있도록 게시판 종류를 구분

작성 성공 시 새로고침으로 목록 출력

webComments.tag

```
<c:if test="${not empty headHuntPostSingle}">
    <input type="hidden" name="postId"
           value="${headHuntPostSingle.headHuntPostId}">
</c:if>
<c:if test="${not empty jobHuntPostSingle}">
    <input type="hidden" name="postId"
           value="${jobHuntPostSingle.jobHuntPostId}">
</c:if>
<c:if test="${not empty freePostSingle}">
    <input type="hidden" name="postId"
           value="${freePostSingle.freePostId}">
</c:if>
<c:if test="${not empty marketPostSingle}">
    <input type="hidden" name="postId"
           value="${marketPostSingle.marketPostId}">
</c:if>
```

댓글 작성 (2/2)

WriteCommentController.java

```
@Controller  
public class WriteCommentController {  
  
    @Autowired  
    private CommentService commentService;  
  
    @RequestMapping (value="/writeComment", method=RequestMethod.POST)  
    public @ResponseBody CommentDTO writeComment(CommentDTO commentDTO, HttpSession session){  
  
        commentDTO.setMemberId((String)session.getAttribute("member"));  
        boolean commentInsertResult= commentService.insert(commentDTO);  
  
        if(commentInsertResult) {  
            return commentDTO;  
        }  
        return null;  
    }  
}
```

@RequestMapping 으로 글 작성 시
post 타입의 데이터를 받은 뒤,
서비스 연결 DAO 에 값을 비동기로 전달

CommentDAO.java

```
private static final String INSERT = "INSERT INTO COMMENT (POST_id, MEMBER_id, COMMENT_content)"  
+ "VALUES (?, ?, ?);"  
  
public boolean insert(CommentDTO commentDTO) {  
    int result = jdbcTemplate.update(INSERT,commentDTO.getPostId(),commentDTO.getMemberId(),commentDTO.getCommentContent());  
    if (result <= 0) {  
        return false;  
    }  
    return true;  
}
```

댓글 수정, 삭제

webComments.tag

```
// AJAX 요청을 통해 서버에 수정된 내용 저장
$.ajax({
    url: "/updateComment",
    type: "POST",
    data: {
        'commentId': commentId,
        'commentContent': updatedContents
    },
success: function(response) {
    var responseData = JSON.parse(response);
    // 동적으로 HTML 요소 생성
    var commentContentDiv = $('

').addClass('description').text(responseData.commentContent);
    // 기존의 내용을 비우고 새로운 내용으로 변경
    $('#commentContent_' + commentId).empty().append(commentContentDiv);
    // 버튼의 텍스트를 '수정'으로 변경
    updateBtn.val('Update');
    updateBtnIcon.attr('class', 'ion-android-refresh');
},


```

webComments.tag

```
$ajax({
    url: "/deleteComment",
    type: "POST",
    data: {
        'commentId': commentId
    },
    dataType: 'text',
    success: function(response) {
        // 성공적으로 삭제되었다는 메시지를 콘솔에 출력
        console.log('삭제 완료:', response);
        // UI에서 삭제된 댓글 제거 또는 사용자에게 성공 알림
        swal({
            title: "success",
            text: "댓글이 성공적으로 삭제되었습니다.",
            type: "success"
        }, function() {
            // 댓글 삭제 후 비동기 처리로 삭제한 div를 삭제
            // 여기에 댓글 삭제 후 비동기 처리로 삭제한 div를 지우는 로직 추가
            $('#item_' + commentId).remove();
        });
    },
});
```

댓글 수정 및 삭제 시 ajax로 각각 서버에 ID와 수정할 내용을 전달
성공 시 해당 div 태그를 변경

답글 목록 출력

webComments.tag

```
var replyToggle = document.getElementById('replyListBtn_' + commentId);
if (replyElement.style.display === "none") {
    replyElement.style.display = "block"; // 요소를 보여줍니다.
    replyToggle.textContent = "Reply List ▲"; // 목록 출력시 화살표 모양 변경
}

$.ajax({
    url: '/replyList',
    method: 'GET',
    data: {
        'commentId': commentId
    },
});
```

답글 목록 출력 시 ajax 코드로 목록 데이터를 호출
innerHTML로 목록을 삽입

Reply List ▲



포로리 ezw0081@gmail.com

저도 가보고 싶네요 ㅋㅋㅋ



킹울 csjin@naver.com

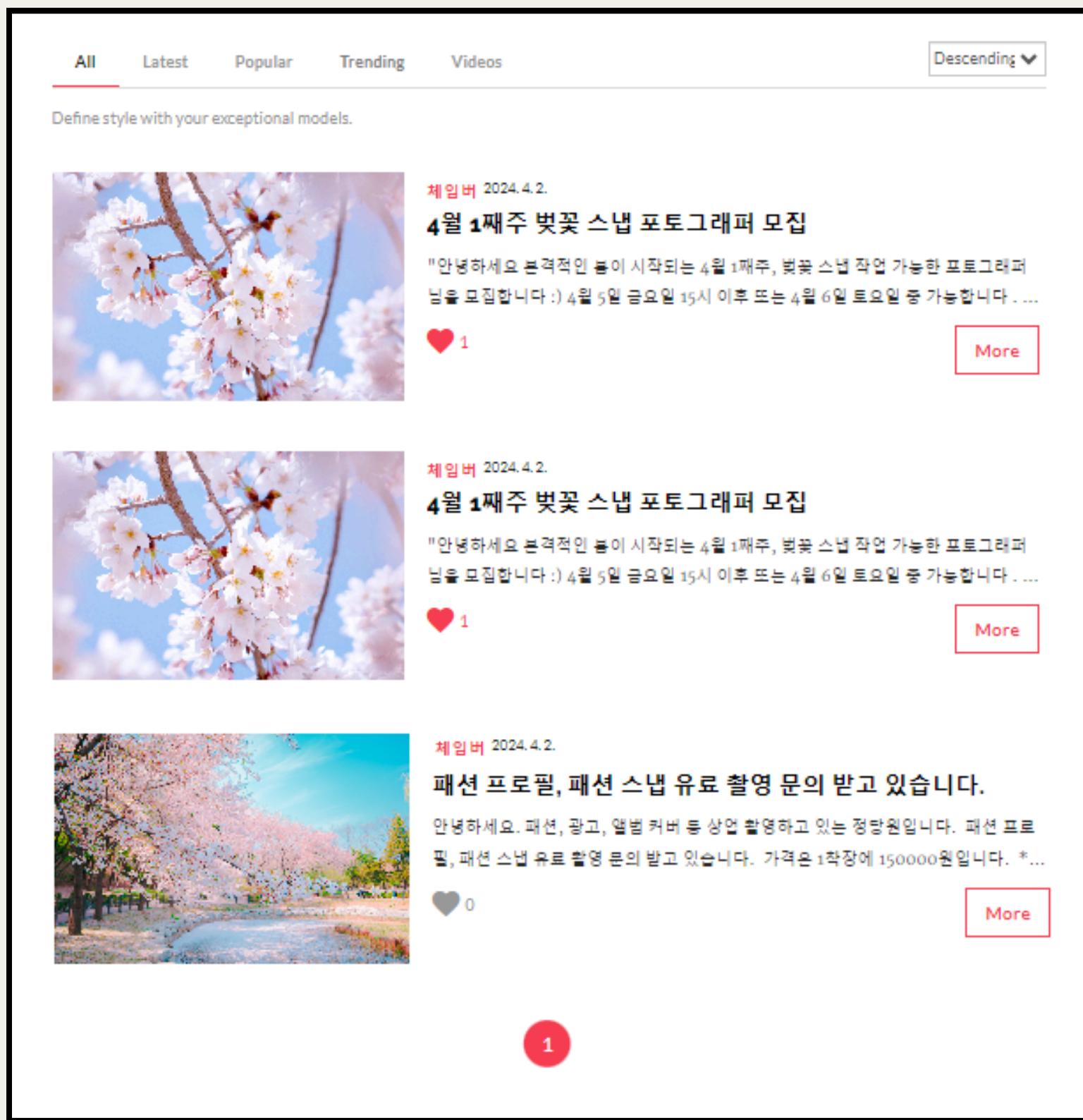
와 사진 너무 멋져요 저도 가고 싶네요 ^^



```
var itemHtml =
`<hr><div class="item" id="item_" + data[i].replyId +'">' +
'<div class="user">' +
'<figure></figure>' +
'<div class="details">' +
'<h5 class="title">' + data[i].memberNickname +
'<a href="/memberPage?memberId=' + data[i].memberId + '">' + data[i].memberId + '</a></h5>' +
'<div class="time"></div>' +
'<div class="description" style="word-break: break-all;" value="Update" id="replyContent_" + data[i].replyId +'">' + data[i].replyContent + '</div>' +
'<div style="display: flex; flex-wrap: wrap; justify-content: flex-end;">' +
'<i class="ion-android-refresh" id="replyUpdateIcon_" + data[i].replyId +
'" onclick="document.getElementById(' + data[i].replyId +
')'.click();" style="cursor: pointer; font-size: 25px; margin-right: 10px;"></i>' +
'<input type="button" style="display: none;" id="replyUpdate_" + data[i].replyId +
'" style="width: 25%; margin-right: 10px;" value="Update" onclick="replyUpdate(' + data[i].replyId + ', \'' + data[i].replyContent + '\')"' +
'<i class="ion-trash-a" id="replyDeleteIcon_" + data[i].replyId +
'" onclick="document.getElementById(' + data[i].replyId +
')'.click();" style="cursor: pointer; font-size: 25px; margin-right: 10px;"></i>' +
'<input type="button" style="display: none;" id="replyDelete_" + data[i].replyId +
'" style="width: 25%; margin-right: 10px;" value="Delete" onclick="replyDelete(' + data[i].replyId + ')"' +
'</div></div></div>`;
```

로그인한 회원이
답글 작성자일 경우
수정과 삭제 버튼 출력

글 목록 페이징 처리 (1/3)



headHuntPostList.jsp

```
<!-- 게시글 전체 갯수가 출력되게 처리 -->
<div class="row">
    <!-- 게시글 목록 출력 장소 시작 게시판마다 명이 다르다. -->
    <div id="postDatasContainer"
        data-displayreviewdata='${headHuntPostList}'></div> ←
    <!-- 게시글 목록 출력 장소 종료-->
    <!-- 페이지 이동 버튼 -->
    <!-- 페이징처리 시 1~10 까지 출력 글이 적으면 1~a 만 출력할 수 있게 처리 -->
    <div class="col-md-12 text-center">
        <ul class='pagination' id="paginationContainer">
            </ul>
    </div>
    <!-- 페이징 끝 -->
</div>
```

JS로 목록을 출력할 수 있도록
div 태그에 data-displayreviewdata 속성을 정의
HeadHuntPostListController.java에서
받아온 값을 JS로 전달

HeadHuntPostListController.java

```
headHuntPostListResult = gson.toJson(headHuntPostService.selectAll(headHuntPostDTO));
```

글 목록 페이징 처리 (2/3)

headHuntPostPagination.js

```
var memberId = $('#sessionId').val();
console.log("headHuntPostList.myRecommend : " + headHuntPostList.myRecommend);
var isAlreadyRecommended = headHuntPostList.myRecommend === 1; // 1이면 true, 그 외에는 false

console.log("memberId : " + memberId);
console.log("isAlreadyRecommended : " + isAlreadyRecommended);

if (!memberId) {
    recommendButtonHTML = `<a id="recommendBtnMessage" onclick="message()" class="love" style="margin-top:0%">
        <i class="ion-android-favorite-outline"></i>
        <div id="recommendCnt">${headHuntPostList.recommendCnt}</div>
    </a>`;
} else {
    recommendButtonHTML = `<a id="recommendBtn_${headHuntPostList.headHuntPostId}">
        class="love ${isAlreadyRecommended ? 'active' : ''}" style="margin-top:0%" data-postid="${headHuntPostList.headHuntPostId}"
        data-memberid="${memberId}"
        <i class="ion-android-favorite"></i>
        <div id="recommendCnt">${headHuntPostList.recommendCnt}</div>
    </a>`;
}

```

목록 출력 시 글 추천 버튼 추가

```
<div style="display:flex; justify-content:space-between; align-items:center; margin-bottom:10px">
    <div style="flex-grow:1; position:relative; overflow-y:scroll; height:150px; border:1px solid #ccc; padding:10px; border-radius:5px; -webkit-overflow-scrolling: touch; <!-- 웹킷 브라우저에서 스크롤 향상 -->">
        <div style="margin-bottom:10px">
            <h3>${postList.title}</h3>
            <div style="display:flex; justify-content:space-between; align-items:center">
                <div>${postList.writer}</div>
                <div>${postList.date}</div>
                <div>${postList.view}</div>
            </div>
            <div style="display:flex; justify-content:space-between; align-items:center; margin-top:10px">
                <div>${postList.comment}</div>
                <div>${postList.share}</div>
            </div>
        </div>
        <div style="border:1px solid #ccc; padding:5px; border-radius:5px; background-color:#f9f9f9; margin-top:10px">
            ${recommendButtonHTML}
        </div>
    </div>
    <div style="flex:1; text-align:right; margin-top:10px">
        <button style="border:1px solid #ccc; padding:5px; border-radius:5px; background-color:#f9f9f9; font-size:1em; margin-right:10px">More</button>
        <button style="border:1px solid #ccc; padding:5px; border-radius:5px; background-color:#f9f9f9; font-size:1em; border-left:1px solid #ccc; padding-left:10px"><i class="ion-ios-arrow-thin-right"></i></button>
    </div>
</div>
```

글 목록 페이징 처리 (3/3)

HeadHuntPostDAO.java

```
( " // 특정 구인글에 대해 로그인한 회원이 좋아요를 눌렀는지 여부를 확인
SELECT "
CASE WHEN COUNT(*) > 0 THEN TRUE ELSE FALSE END " // 좋아요를 눌렸으면 TRUE, 아니면 FALSE를 반환
FROM "
RECOMMEND " // 좋아요 테이블
WHERE "
RECOMMEND.POST_id = HEADHUNTPOST.HEADHUNTPOST_id " // 구인글에 좋아요가 눌렸는지 확인 후
AND RECOMMEND.MEMBER_id = ? " // 현재 로그인한 회원의 아이디와 일치하는 좋아요만 선택
) AS myRecommend, " // 로그인한 회원이 좋아요를 눌렀는지 중복 체크
```

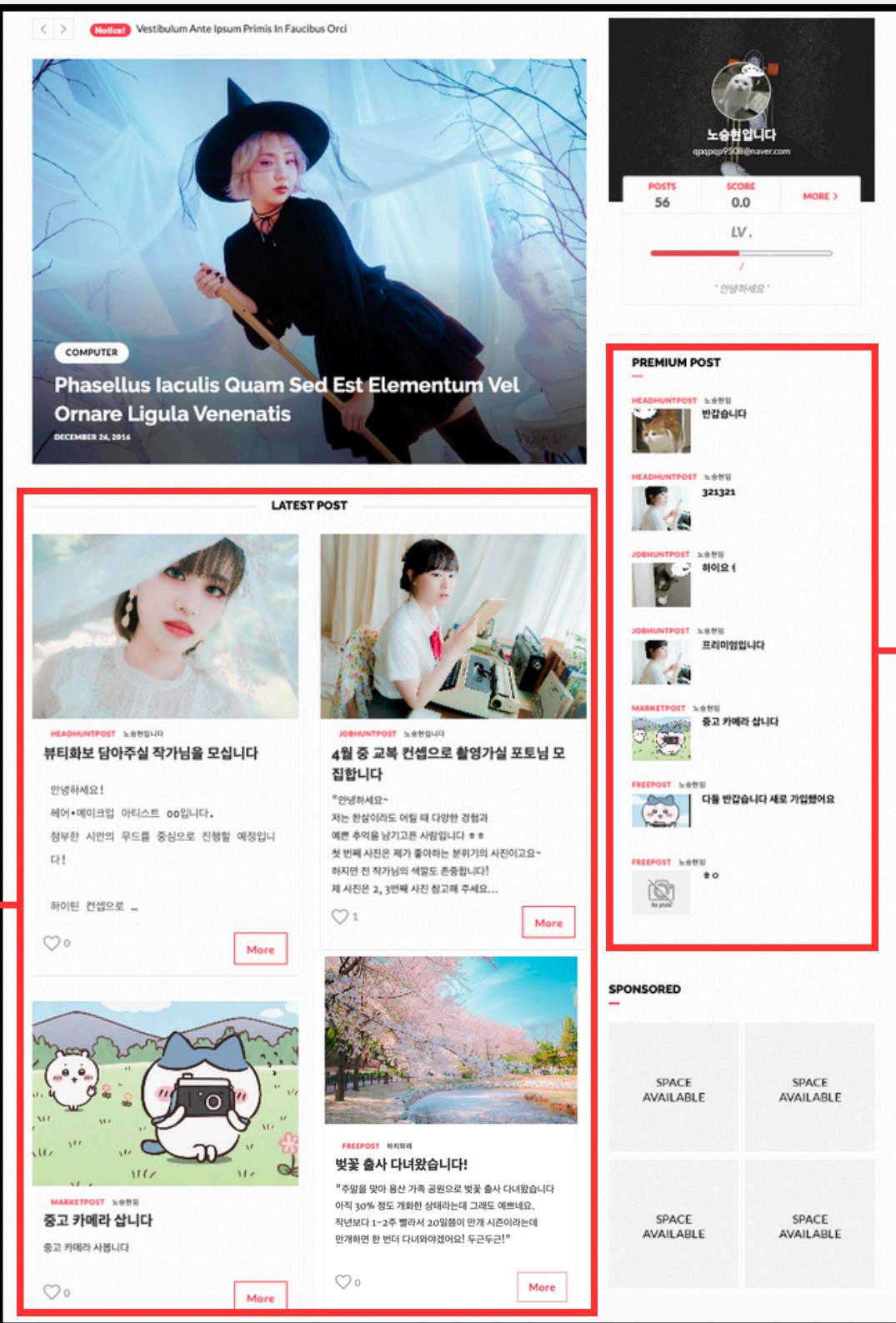
글 목록 쿼리문
해당 글의 추천 버튼을 누른 회원 정보를
서브 쿼리로 출력

```
else if (headHuntPostDTO.getSearchCondition().equals("headHuntPostList")) {
Object[] args = { headHuntPostDTO.getMemberId() };
result = jdbcTemplate.query(SELECTALL_HEADHUNTPOST, args, new HeadHuntPostRowMapper());
System.out.println("HeadHuntPostDAO(selectAll) Out로그 = [" + result + "]");
return result;
```

SelectAll 쿼리를 실행시키기 위해 로그인 한 회원의 값을 받을 수 있도록
jdbcTemplate 으로 로그인한 회원의 값을 전달

메인 페이지 출력 (1/2)

main.jsp



게시판 별
최신 글

유저 레벨순

프리미엄
회원이
작성한
최신 글

주간 추천 글



누적
추천 글

메인 페이지 출력 (2/2)

HeadHuntPostDAO.java

```
// 주간 추천순 구인글 목록 출력 (2개만 출력).전미지
private static final String SELECTALL_HEADHUNTPOSTWEEKLYBEST = "SELECT "
+ "DISTINCT " // 중복 제거 함수
+ " HEADHUNTPOST.HEADHUNTPOST_id, "
+ " HEADHUNTPOST.MEMBER_id, "
+ " MEMBER.MEMBER_nickname, "
+ " HEADHUNTPOST.HEADHUNTPOST_title, "
+ " HEADHUNTPOST.HEADHUNTPOST_content, "
+ " HEADHUNTPOST.HEADHUNTPOST_date, "
+ " ( SELECT " // 특정 구인글에 대해 로그인한 회원이 좋아요를 눌렀는지 여부를 확인
CASE WHEN COUNT(*) > 0 THEN TRUE ELSE FALSE END " // 좋아요를 눌렀으면 TRUE, 아니면 FALSE를 반환
FROM "
+ " RECOMMEND " // 좋아요 테이블
WHERE "
+ " RECOMMEND.POST_id = HEADHUNTPOST.HEADHUNTPOST_id " // 구인글에 좋아요가 눌렸는지 확인 후
AND RECOMMEND.MEMBER_id = ? " // 현재 로그인한 회원의 아이디와 일치하는 좋아요만 선택
) AS myRecommend, " // 로그인한 회원이 좋아요를 눌렀는지 중복 체크
( SELECT " // 게시글의 좋아요 수를 합산
COUNT(*) " // 해당 게시글에 대한 좋아요 수를 COUNT 함수를 사용해 합산
FROM "
+ " RECOMMEND " // 좋아요 테이블에서 가져옴
WHERE "
+ " RECOMMEND.POST_id = HEADHUNTPOST.HEADHUNTPOST_id " // 게시글 아이디와 좋아요 테이블의 게시글 아이디가 동일한 것을 선택
) AS RECOMMEND_cnt, " // 좋아요 수
( SELECT " // 대표 이미지 설정
POSTIMG.POSTIMG_name " // 게시글 이미지를 선택
FROM "
+ " POSTIMG " // 게시글 이미지 테이블
WHERE "
+ " POSTIMG.POST_id = HEADHUNTPOST.HEADHUNTPOST_id " // 게시글 아이디와 이미지 테이블의 게시글 아이디가 동일한 것을 선택
ORDER BY "
+ " POSTIMG.POSTIMG_id ASC " // 이미지 아이디를 기준으로 오름차순 정렬
LIMIT 1 " // 이미지를 1개만 가져오도록 설정
) AS POSTIMG_name " // 대표 이미지의 이름
+ "FROM "
+ " HEADHUNTPOST " // 구인글 테이블
+ "INNER JOIN "
+ " MEMBER ON HEADHUNTPOST.MEMBER_id = MEMBER.MEMBER_id " // 회원 정보와 INNER JOIN
+ "LEFT JOIN "
+ " RECOMMEND ON HEADHUNTPOST.HEADHUNTPOST_id = RECOMMEND_POST_id " // 좋아요 정보와 LEFT JOIN
+ "WHERE "
+ " HEADHUNTPOST.HEADHUNTPOST_date >= DATE SUB(NOW(), INTERVAL 1 WEEK) " // 최근 1주일간 작성된 글만 선택
+ "ORDER BY "
+ " RECOMMEND_cnt DESC, " // (1) 좋아요 수가 많은 순으로 1차 정렬 후
+ " HEADHUNTPOST.HEADHUNTPOST_date DESC " // (2) 최근 작성일 순으로 2차 정렬
+ "LIMIT 2 "; // 글을 1개만 가져오도록 설정

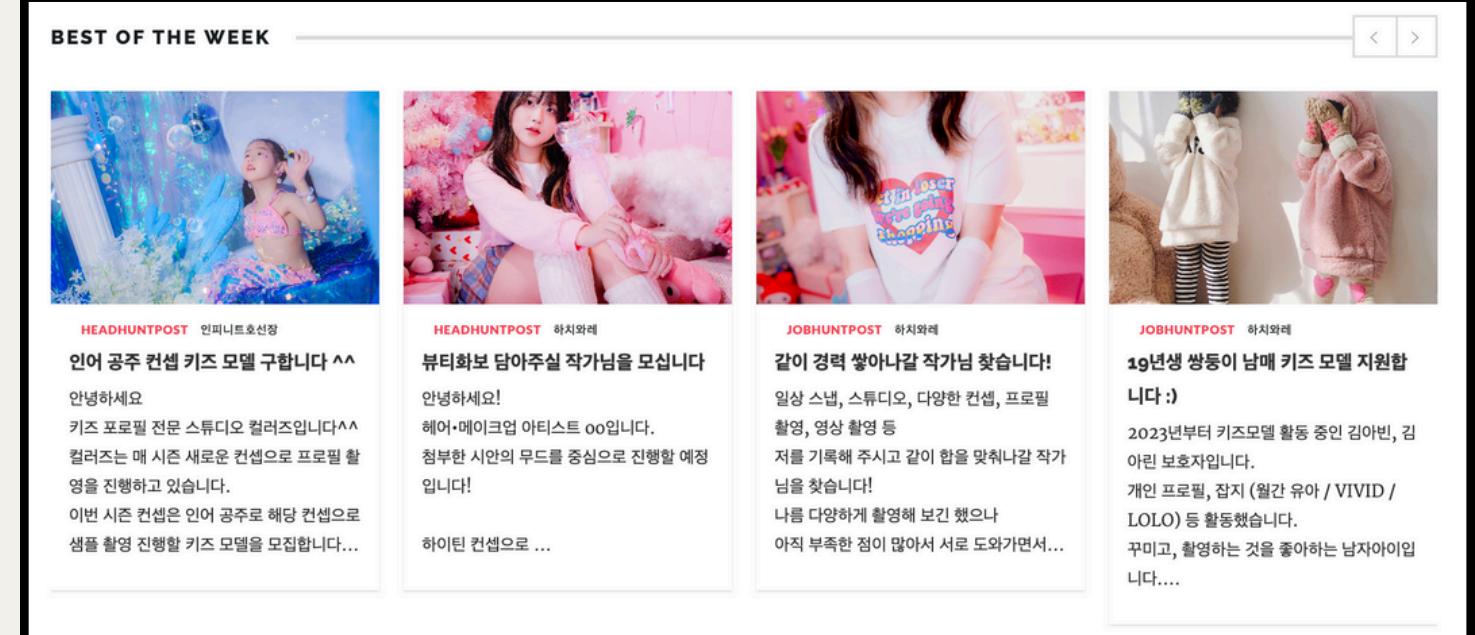
// 사용한 테이블 : 구인글 테이블, 회원 테이블, 좋아요 테이블, 게시글 이미지 테이블
// 사용한 컬럼 : 카테고리, 구인글 아이디, 회원 아이디, 회원 닉네임(회원 테이블), 제목, 내용, 작성일, 좋아요 중복 체크(좋아요 테이블), 좋아요 수(좋아요 테이블), 대표 이미지(이미지 테이블)
// 쿼리문 설명 :
// INNER JOIN을 사용해 구인글 테이블과 회원 테이블을 연결하고, 또 다른 LEFT JOIN을 사용해 구인글 테이블과 좋아요 테이블을 연결
// 게시글의 좋아요는 테이블을 따로 나누었으며 COUNT() 함수를 사용해 게시글에 대한 좋아요 수를 합산하고, 그 결과를 "RECOMMEND_cnt"라는 별칭으로 반환
// 게시글 이미지는 테이블을 따로 나누었으며 서브 쿼리를 사용해 게시글 이미지 중 대표 이미지로 보여줄 이미지를 설정하고, 그 결과를 "POSTIMG_name"라는 별칭으로 반환
```

좋아요
체크
좋아요
합산
대표
이미지

출력 조건

주간 추천 글

main.jsp



최근 1주일간 작성된 글 중
좋아요를 많이 받은 순으로 정렬한 뒤
최근에 작성된 글 2개 출력

필터 검색 (1/5)

MyBatis 프레임워크 활용 전

HeadHuntPostFilterDAO.java

```
// 사용자가 입력하는 검색 결과에 따른 쿼리문
if (filterDTO.getSearchField() != null && filterDTO.getSearchInput() != null) {
    String searchInput = filterDTO.getSearchInput().replace(" ", "").toLowerCase();
    if (filterDTO.getSearchField().equals("headHuntPostTitle")) {
        USERSEARCHSQL = "AND REPLACE(LOWER(HEADHUNTPOST_title), ' ', '') LIKE '%'||'" + searchInput + "'||'%'";
    }
    if (filterDTO.getSearchField().equals("headHuntPostContent")) {
        USERSEARCHSQL = "AND REPLACE(LOWER(HEADHUNTPOST_content), ' ', '') LIKE '%'||'" + searchInput + "'||'%'";
    }
    if (filterDTO.getSearchField().equals("memberNickname")) {
        USERSEARCHSQL = "AND REPLACE(LOWER(MEMBER_id), ' ', '') LIKE '%'||'" + searchInput + "'||'%'";
    }
    if (filterDTO.getSearchField().equals("titleAndContents")) {
        USERSEARCHSQL = "AND (REPLACE(LOWER(HEADHUNTPOST_title), ' ', '') LIKE '%'||'" + searchInput
                        + "'||%' OR REPLACE(LOWER(HEADHUNTPOST_content), ' ', '') LIKE '%'||'" + searchInput + "'||'%'))";
    }
}
```

```
class FilterRowMapper implements RowMapper<HeadHuntPostDTO> {
    @Override // mapRow 메서드 오버라이드
    public HeadHuntPostDTO mapRow(ResultSet rs, int rowNum) throws SQLException {
        // ResultSet에 저장된 데이터를 HeadHuntPostDTO 객체에 맵핑(저장)하는 메서드
        HeadHuntPostDTO data = new HeadHuntPostDTO();
        // ResultSet에 저장된 데이터를 HeadHuntPostDTO 객체에 저장
        data.setHeadHuntPostId(rs.getString("HEADHUNTPOST_id"));
        data.setMemberId(rs.getString("MEMBER_id"));
        data.setHeadHuntPostTitle(rs.getString("HEADHUNTPOST_title"));
        data.setHeadHuntPostContent(rs.getString("HEADHUNTPOST_content"));
        return data;
    }
}
```

MyBatis 프레임워크 활용 후

HeadHuntPostFilterSearch.xml

```
<if test="searchField.equals('headHuntPostTitle')">
    AND HEADHUNTPOST.HEADHUNTPOST_title LIKE CONCAT('%', #{searchInput}, '%')
</if>
<if test="searchField.equals('headHuntPostContent')">
    AND HEADHUNTPOST.HEADHUNTPOST_content LIKE CONCAT('%', #{searchInput}, '%')
</if>
<if test="searchField.equals('memberNickname')">
    AND MEMBER.MEMBER_nickname LIKE CONCAT('%', #{searchInput}, '%')
</if>
<if test="searchField.equals('titleAndContents')">
    AND (HEADHUNTPOST.HEADHUNTPOST_title LIKE CONCAT('%', #{searchInput}, '%')
          OR HEADHUNTPOST.HEADHUNTPOST_content LIKE CONCAT('%', #{searchInput}, '%'))
</if>
```

application.properties

```
mybatis.configuration.map-underscore-to-camel-case=true
```

HeadHuntPostFilterSearch.xml

```
<mapper namespace=
    "infinitystone.chalKag.biz.headHuntPost.IHeadHuntPostDAO">
    <select id="selectAll"
        resultType="infinitystone.chalKag.biz.headHuntPost.HeadHuntPostDTO">
```

필터 검색 (2/5)

headHuntPostList.jsp headHuntPostFilterSearch.js

title + contents

모델

FILTER

- DATE
 - Anytime
 - Today
 - Last Week
 - Last Month
- ROLE
 - Model
- REGION
 - Select

```
// 날짜와 카테고리에 따른 변수 업데이트 함수
function updateVariables() {
    // 검색 카테고리와 입력값
    searchField = $('#searchField').val();
    searchInput = $('#searchInput').val();
    console.log("searchInput : " + searchInput)

    minPay = $('#minPay').val();
    console.log("minPay : " + minPay);

    maxPay = $('#maxPay').val();
    console.log("maxPay : " + maxPay);

    role = $('#role').val();
    console.log("role : " + role);

    region = $('#region').val();
    console.log("region : " + region);

    startDate = $('#startDate').val();
    console.log("startDate : " + startDate);

    endDate = $('#endDate').val();
    console.log("endDate : " + endDate);

    concept = $('#concept').val();
    console.log("concept : " + concept);

    // 정렬 순서
    sortOrder = $('#sortOrder').val();
```

```
// AJAX 요청 수행 함수
function performAjaxRequest() {
    // 서버에 보낼 데이터 준비.
    const requestData = {
        fromday: minDate,
        today: maxDate,
        searchField: searchField, // search Condition(제목, 내용, 제목 + 내용)
        searchInput: searchInput, // search Condition (입력값)
        sortOrder: sortOrder, // 오름차순 정렬, 내림차순 정렬
        minPay : minPay,
        maxPay : maxPay,
        headHuntPostRole : role,
        headHuntPostRegion : region,
        startWorkDate : startDate,
        endWorkDate : endDate,
        headHuntPostConcept : concept
    };
    console.log("requestData 로그 : " + JSON.stringify(requestData));

    // jQuery를 사용한 AJAX 요청
    $.ajax({
        url: '/headHuntPostFilterSearch', // 서버의 엔드포인트 URL
        type: 'post', // 또는 'POST', 서버의 요구 사항에 따라
        data: requestData, // 서버에 보낼 데이터
        dataType: 'json',
        success: function(filterData) {
            console.log("필터 AJAX 콘솔 진입 로그 : " + JSON.stringify(filterData));
        }
    });
}
```

필터 적용 시 JavaScript 는 ajax 를 사용해 서버로 비동기 요청을 보냄

필터 검색 (3/5)

IHeadHuntPostService.java

```
public interface IHeadHuntPostService {  
    // 구인 게시판의 블더 검색 시 간드플더에서 사용할 서비스  
    // (필터검색과 관련된 비즈니스 로직을 정의하는 인터페이스)  
  
    public List<HeadHuntPostDTO> selectAll(HeadHuntPostDTO headHuntPostDTO);  
  
    public HeadHuntPostDTO selectOne(HeadHuntPostDTO headHuntPostDTO);  
  
    public boolean insert(HeadHuntPostDTO headHuntPostDTO);  
  
    public boolean update(HeadHuntPostDTO headHuntPostDTO);  
  
    public boolean delete(HeadHuntPostDTO headHuntPostDTO);  
}
```

Controller는 사용자의 요청을 전달받아
필요한 서비스 메서드 호출 후 검색 수행

인터페이스와 구현 클래스를 분리함으로써
각각을 독립적으로 개발할 수 있게 해
코드의 유연성과 확장성 증가 및
유지 보수성 향상

IHeadHuntPostServiceImpl.java

```
@Service("iHeadHuntPostService")  
// @Service : 구인 게시판의 필터검색 비즈니스 로직을 담당하는 서비스 클래스임을 명시(코드의 모듈화와 유지보수성↑)  
  
public class IHeadHuntPostServiceImpl implements IHeadHuntPostService {  
    // 구인 게시판의 필터검색 비즈니스 로직을 수행할 ServiceImpl 클래스  
  
    @Autowired // @Autowired : 구인 게시판의 필터검색 .xml과 이어진 IHeadHuntPostDAO를 의존 주입  
    private IHeadHuntPostDAO iHeadHuntPostDAO;  
  
    // 필터 검색 시 뷰에서 받은 값을 넣어서 출력  
    @Override  
    public List<HeadHuntPostDTO> selectAll(HeadHuntPostDTO headHuntPostDTO) {  
        // 필터링을 위한 매개변수를 Map으로 생성  
        Map<String, Object> map = new HashMap<String, Object>();  
        System.out.println("IMPL " + headHuntPostDTO);  
  
        // DTO 객체에서 필터링에 필요한 정보들을 추출하여 Map에 담음  
        map.put("headHuntPostId", headHuntPostDTO.getHeadHuntPostId()); // 구인글 아이디  
        map.put("headHuntPostTitle", headHuntPostDTO.getHeadHuntPostTitle()); // 제목  
        map.put("headHuntPostContent", headHuntPostDTO.getHeadHuntPostContent()); // 내용  
        map.put("headHuntPostRole", headHuntPostDTO.getHeadHuntPostRole()); // 작업 직업  
        map.put("headHuntPostRegion", headHuntPostDTO.getHeadHuntPostRegion()); // 작업 지역  
        map.put("headHuntPostConcept", headHuntPostDTO.getHeadHuntPostConcept()); // 촬영 컨셉  
  
        map.put("sortOrder", headHuntPostDTO.getSortOrder()); // 정렬  
  
        map.put("searchField", headHuntPostDTO.getSearchField()); // 검색 키워드  
        map.put("searchInput", headHuntPostDTO.getSearchInput()); // 검색어  
  
        map.put("memberId", headHuntPostDTO.getMemberId()); // 회원 아이디  
  
        System.out.println("MAP" + map);  
        System.out.println("iHeadHuntPostDAO.selectAll(map) :" + iHeadHuntPostDAO.selectAll(map)); // 검색 조건  
  
        return iHeadHuntPostDAO.selectAll(map); // 조회된 구인 게시글 리스트를 반환  
    }  
}
```

필터 검색 (4/5)

HeadHuntPostFilterSearch.xml

```
-//mybatis.org//DID Mapper 3.0//EN (doctype)
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="infinitystone.chalKag.biz.headHuntPost.IHeadHuntPostDAO">
    <select id="selectAll" resultMap="headHuntPostMap">
        <if test="searchField.equals('headHuntPostTitle')">
            AND HEADHUNTPOST.HEADHUNTPOST_title LIKE CONCAT('%', #{searchInput}, '%') </if>
        <if test="searchField.equals('headHuntPostContent')">
            AND HEADHUNTPOST.HEADHUNTPOST_content LIKE CONCAT('%', #{searchInput}, '%') </if>
        <if test="searchField.equals('memberNickname')">
            AND MEMBER.MEMBER_nickname LIKE CONCAT('%', #{searchInput}, '%') </if>
        <if test="searchField.equals('titleAndContents')">
            AND (HEADHUNTPOST.HEADHUNTPOST_title LIKE CONCAT('%', #{searchInput}, '%')
                  OR HEADHUNTPOST.HEADHUNTPOST_content LIKE CONCAT('%', #{searchInput}, '%')) </if>
        <if test="headHuntPostRole != null and headHuntPostRole != ''">
            AND HEADHUNTPOST_role LIKE CONCAT('%', #{headHuntPostRole}, '%')
        </if>
        <if test="headHuntPostRegion != null and headHuntPostRegion != ''">
            AND HEADHUNTPOST_region LIKE CONCAT('%', #{headHuntPostRegion}, '%')
        </if>
        <if test="headHuntPostConcept != null and headHuntPostConcept != ''">
            AND HEADHUNTPOST_concept LIKE CONCAT('%', #{headHuntPostConcept}, '%') </if>
        <if test="minPay != null and maxPay != null and minPay != 0 and maxPay != 0">
            AND HEADHUNTPOST_pay BETWEEN #{minPay} AND #{maxPay} </if>
        <if test="startWorkDate != null and endWorkDate != null and startWorkDate != '' and endWorkDate != ''">
            AND HEADHUNTPOST_workDate BETWEEN #{startWorkDate} AND #{endWorkDate} </if>
        <if test="fromday != null and fromday != ''">
            AND DATE(HEADHUNTPOST_date) BETWEEN #{fromday} AND #{today} </if>
    ORDER BY HEADHUNTPOST.HEADHUNTPOST_date ${sortOrder == 'asc' ? 'ASC' : 'DESC'}
```

XML에서는 사용자가 입력한 검색 조건에 따라 동적인 SQL 을 생성해 데이터 조회 후 최종 결과를 resultMap 에 정의된 DTO 객체로 매팅 후 반환

필터 검색 (5/5)

headHuntPostList.jsp

The image displays two side-by-side screenshots of the Chalkag website's search results page for 'Headhunt Post'.
Left Screenshot (Witch Theme):
- **Search Bar:** Type something here
- **Filter Options:**

- DATE:** Anytime (selected), Today, Last Week, Last Month
- ROLE:** Select
- REGION:** Select
- PAY:** Min (滑块), Max (滑块)
- WORK DATE:** Start (연도, 월, 일), End (연도, 월, 일)
- WORK CONCEPT:** Select

<- **Results:**

- 4월 17일(수요일) ~ 4월 24일(수요일) 스냅촬영 가실 모델님!
하치와레 9시간 전
- 4월 2째주 벚꽃 스냅 포토그래퍼 모집
하치와레 9시간 전
- 볼 풀장 & 물총 놀이 컨셉 페이 모델 모집
하치와레 9시간 전
- 경복궁 한복 트윈 촬영 모델 모집경복궁 한복 트윈 촬영 모델 모집
하치와레 9시간 전
- 하치와레 15시간 전
뷰티화보 담아주실 작가님을 모십니다

<- **Buttons:** FILTER RESET, More
Right Screenshot (Korean traditional Hanbok Theme):
- **Search Bar:** Type something here
- **Filter Options:**

- DATE:** Anytime, Today, Last Week (selected), Last Month
- ROLE:** Model
- REGION:** Select
- PAY:** Min (滑块), Max (滑块)
Max value: 451,000
- WORK DATE:** Start (연도, 월, 일), End (연도, 월, 일)
- WORK CONCEPT:** Select

<- **Results:**

- 4월 17일(수요일) ~ 4월 24일(수요일) 스냅촬영 가실 모델님!
하치와레 9시간 전
- 볼 풀장 & 물총 놀이 컨셉 페이 모델 모집
하치와레 9시간 전
- 경복궁 한복 트윈 촬영 모델 모집경복궁 한복 트윈 촬영 모델 모집
하치와레 2024.4.12.
- 하간 스냅 같이 해보실 분!
하간 스냅에 아직 실력이 미숙하여 좀 날이 풀리면 모델이 되어주실 분을 찾습니다 :) 최초 모델이 시연 같이 실력을 기르면 좋겠습니다 모델분은 포즈를 저는 야간 촬영 실력을..ㅎㅎ 일정은 서...
- 인피니트호선장 2024.4.11.
인어 공주 컨셉 키즈 모델 구합니다 ^^

<- **Buttons:** FILTER RESET, More

클라이언트에서는 ajax 응답을 받으면 검색 결과를 비동기 처리로 출력

게시글 사진 출력 (1/2)

jobHuntPostList.jsp



노승현입니다 0분 전

4월 중 교복 컨셉으로 촬영가실 포토님 모집합니다

"안녕하세요~ 저는 한살이라도 어릴 때 다양한 경험과 예쁜 추억을 남기고픈 사람입니다 ㅎㅎ 첫 번째 사진은 제가 좋아하는 분위기의 사진이고요~ 하지만 전 작가님의 색깔도 존중합니다! 제 ...

0

JobHuntPostDAO.java

```
JobHuntPostDTO jobHuntPostDTO = new JobHuntPostDTO(); // 새로운 JobHuntPostDTO 객체 생성
// ResultSet에 저장된 데이터를 JobHuntPostDTO 객체에 저장
jobHuntPostDTO.setPostCategory(rs.getString("POST_category")); // 카테고리
jobHuntPostDTO.setJobHuntPostId(rs.getString("JOBHUNTPOST_id")); // 구직글 아이디
jobHuntPostDTO.setMemberId(rs.getString("MEMBER_id")); // 회원 아이디
jobHuntPostDTO.setMemberNickname(rs.getString("MEMBER_nickname")); // 회원 닉네임
jobHuntPostDTO.setJobHuntPostTitle(rs.getString("JOBHUNTPOST_title")); // 제목
jobHuntPostDTO.setJobHuntPostContent(rs.getString("JOBHUNTPOST_content")); // 내용
jobHuntPostDTO.setJobHuntPostDate(rs.getString("JOBHUNTPOST_date")); // 작성일
jobHuntPostDTO.setJobHuntPostViewcnt(rs.getString("JOBHUNTPOST_viewcnt")); // 조회수
jobHuntPostDTO.setMyRecommend(rs.getInt("myRecommend")); // 좋아요 로그인한 회원이 좋아요를 눌렀는지 체크
jobHuntPostDTO.setRecommendCnt(rs.getInt("RECOMMEND_cnt")); // 좋아요 수
jobHuntPostDTO.setPostImgName(rs.getString("POSTIMG_name")); // 대표 이미지
return jobHuntPostDTO; // jobHuntPostDTO에 저장된 데이터들을 반환
```

PostImgDAO.java

```
" // 대표 이미지 설정
SELECT "
    POSTIMG.POSTIMG_name " // 게시글 이미지를 선택
FROM "
    POSTIMG " // 게시글 이미지 테이블
WHERE "
    POSTIMG.POST_id = JOBHUNTPOST.JOBHUNTPOST_id " // 게시글 아이디와 이미지 테이블의 게시글 아이디가 동일
ORDER BY "
    POSTIMG.POSTIMG_id ASC " // 이미지 아이디를 기준으로 오름차순 정렬
LIMIT 1 " // 이미지를 1개만 가져오도록 설정
AS POSTIMG name " // 대표 이미지의 이름
```

게시글 이미지는 테이블을 따로 나누었으며,
서브 쿼리를 사용해 게시글 이미지 중 대표 이미지로 보여줄 이미지를 설정하고,
그 결과를 "POSTIMG_name"라는 별칭으로 반환

게시글 사진 출력 (2/2)

JobHuntPostListController.java

```
jobHuntPostDTO.setSearchCondition("jobHuntPostList");

String jobHuntPostListResult = gson.toJson(jobHuntPostService.selectAll(jobHuntPostDTO));

model.addAttribute("jobHuntPostList", jobHuntPostListResult);

System.out.println("JobHuntPostListController Out로그");

return "jobHuntPost/jobHuntPostList";
}
```

jobHuntPostPagination.js

```
<div class="inner">
  <figure>
    <a href="/jobHuntPostSingle?jobHuntPostId=${jobHuntPostList.jobHuntPostId}">
      
    </a>
  </figure>
  <div class="details">
    <div class="detail">
      <div class="category">
        <a href="memberPage?memberId=${jobHuntPostList.memberId}" style="font-size:14px; font-weight:600;">${jobHuntPostList.memberNickname}</a>
      </div>
      &ampnbsp&ampnbsp<time>${timeString}</time>
    </div>
    <h1 style="width: 522.5px; max-height: 56px; overflow: hidden; display: -webkit-box; -webkit-box-orient: vertical; -webkit-line-clamp: 2;">
      <a href="/jobHuntPostSingle?jobHuntPostId=${jobHuntPostList.jobHuntPostId}">${jobHuntPostList.jobHuntPostTitle}</a>
    </h1>
    <p style="width: 522.5px; max-height: 52px; overflow: hidden; display: -webkit-box; -webkit-box-orient: vertical; -webkit-line-clamp: 2;">
      ${jobHuntPostList.jobHuntPostContent}
    </p>
  <footer>
```

jobHuntPostResult에 저장된 DTO 값을 addAttribute를 사용해 출력

게시글 상세보기 사진 출력 (1/2)

jobHuntPostSingle.jsp JobHuntPostSingleController.java



```
@RequestMapping("/jobHuntPostSingle")
public String jobHuntPostList(Model model, JobHuntPostDTO jobHuntPostDTO,
    // jobHuntPostDTO에 있는 정보로 게시글 내용 불러오기
    jobHuntPostDTO.setSearchCondition("jobHuntPostSingle");
    jobHuntPostDTO = jobHuntPostService.selectOne(jobHuntPostDTO);
    commentDTO.setPostId(jobHuntPostDTO.getJobHuntPostId());
    postImgDTO.setSearchCondition("jobHuntPostSingleImg");
    postImgDTO.setPostId(jobHuntPostDTO.getJobHuntPostId());

    List<PostImgDTO> postImgList = postImgService.selectAll(postImgDTO);
    List<CommentDTO> commentList = commentService.selectAll(commentDTO);
    System.out.println("postImgList : " + postImgList);
    model.addAttribute("jobHuntPostSingle", jobHuntPostDTO);
    model.addAttribute("commentList", commentList);
    model.addAttribute("postImgList", postImgList);

    return "jobHuntPost/jobHuntPostSingle";
}
```

jobHuntPostSingle.jsp

```
<div class="featured">
    <!-- 이미지 목록 캐러셀로 출력 -->
    <div class="owl-carousel">
        <c:forEach var="postImgList" items="${postImgList}">
            <figure>
                
            </figure>
        </c:forEach>
    </div>
</div>
```

이미지를 여러장 저장 가능하므로
List 탑입 사용

JSTL 아이템 태그를 이용해
사진이 저장된 리스트 출력

게시글 상세보기 사진 출력 (2/2)

JobHuntPostDAO.java

```
+ " JOBHUNTPOST.JOBHUNTPOST_id, "
+ " JOBHUNTPOST.MEMBER_id, "
+ " MEMBER.MEMBER_nickname, "
+ " MEMBER.MEMBER_introduction, "
+ " JOBHUNTPOST.JOBHUNTPOST_title, "
+ " JOBHUNTPOST.JOBHUNTPOST_content, "
+ " JOBHUNTPOST.JOBHUNTPOST_role, "
+ " JOBHUNTPOST.JOBHUNTPOST_region, "
+ " JOBHUNTPOST.JOBHUNTPOST_pay, "
+ " JOBHUNTPOST.JOBHUNTPOST_workDate, "
+ " JOBHUNTPOST.JOBHUNTPOST_concept, "
+ " JOBHUNTPOST.JOBHUNTPOST_date, "
+ " JOBHUNTPOST.JOBHUNTPOST_viewcnt, "
+ " // 회원 프로필로 보여줄 이미지 설정
SELECT "
    PROFILEIMG.PROFILEIMG_name // 프로필 이미지를 선택
FROM "
    PROFILEIMG // 프로필 이미지 테이블
WHERE "
    PROFILEIMG.MEMBER_id = JOBHUNTPOST.MEMBER_id // 회원 아이디와 프로필 이미지 테이블의
ORDER BY "
    PROFILEIMG.PROFILEIMG_id DESC // 프로필 이미지 아이디를 기준으로 내림차순 정렬
LIMIT 1 // 이미지를 1개만 가져오도록 설정
) AS PROFILEIMG_name, // 프로필 이미지의 이름
( // 게시글의 좋아요 수를 합산
SELECT "
    COUNT(*) // 해당 게시글에 대한 좋아요 수를 COUNT 함수를 사용해 합산
FROM "
    RECOMMEND // 좋아요 테이블에서 가져옴
WHERE "
    RECOMMEND.POST_id = JOBHUNTPOST.JOBHUNTPOST_id // 게시글 아이디와 좋아요 테이블의
) AS RECOMMEND_cnt // 좋아요 수
FROM "
    JOBHUNTPOST // 구직글 테이블
INNER JOIN "
    MEMBER ON JOBHUNTPOST.MEMBER_id = MEMBER.MEMBER_id // 회원 정보와 INNER JOIN
LEFT JOIN "
    RECOMMEND ON JOBHUNTPOST.JOBHUNTPOST_id = RECOMMEND.POST_id // 좋아요 정보와 LEFT JOIN
WHERE "
    JOBHUNTPOST.JOBHUNTPOST_id = ? ;
```

PostImgDAO.java

```
private static final String SELECTONE_JOBHUNTPOSTIMG = "SELECT "
    + "POSTIMG.POSTIMG_name, "
    + "POSTIMG.POSTIMG_id, "
    + "JOBHUNTPOST.JOBHUNTPOST_id "
    + " FROM POSTIMG "
    + " INNER JOIN JOBHUNTPOST ON POSTIMG.POST_id = JOBHUNTPOST.JOBHUNTPOST_id "
    + " WHERE POSTIMG.POST_id = ?";
```

PostImgDAO.java

```
else if(postImgDTO.getSearchCondition().equals("jobHuntPostSingleImg")) {
    Object[] args = {postImgDTO.getPostId()};
    result = jdbcTemplate.query(SELECTONE_JOBHUNTPOSTIMG, args, new PostImgRowMapper());
```

게시판 이미지 테이블에 있는
해당 게시글의 저장된 사진을
전부 다 출력하기 위해
JdbcTemplate에 query 메서드 사용

게시글 수정하기 (1/2)

UpdateJobHuntPostController.java

```
@RequestMapping(value = "/updateJobHuntPost", method = RequestMethod.POST)
public String updateJobHuntPost(JobHuntPostDTO jobHuntPostDTO, PostImgDTO postImgDTO, HttpSession session, @RequestParam(value = "file",
required = false) MultipartFile[] files) {
    // 사용자의 글을 수정
    jobHuntPostDTO.setMemberId((String) session.getAttribute("member"));

    System.out.println("UpdateJobHuntPostController In Log = [" + jobHuntPostDTO + "]");
    System.out.println("UpdateJobHuntPostController In Log = [" + postImgDTO + "]");

    String uploadDir = this.getClass().getResource("").getPath();
    System.out.println("UpdateJobHuntPostController Log01 = [" + uploadDir + "]");

    uploadDir = uploadDir.substring(1, uploadDir.indexOf("chakKag")) + "chakKag/src/main/resources/static/postImg";
    System.out.println("UpdateJobHuntPostController Log02 = [" + uploadDir + "]");

    jobHuntPostDTO.setSearchCondition("jobHuntPostUpdate");

    // Update the user-written post
    if (!jobHuntPostService.update(jobHuntPostDTO)) // ←
        System.out.println("UpdateJobHuntPostController Update of post failed");
    }
    postImgDTO.setPostId(jobHuntPostDTO.getJobHuntPostId());

    if (!postImgService.delete(postImgDTO)) // ←
        System.out.println("UpdateJobHuntPostController Delete of post images failed");
    }

    for (MultipartFile file : files) {
        if (file != null && !file.isEmpty()) {
            String originalFilename = file.getOriginalFilename();
            String extension = FilenameUtils.getExtension(originalFilename);
            String newFilename = UUID.randomUUID().toString() + "." + extension;
            String filePath = uploadDir + File.separator + newFilename;
            File newFile = new File(filePath);
            postImgDTO.setPostImgName(newFilename);
            if (!postImgService.insert(postImgDTO)) // ←
                System.out.println("UpdateJobHuntPostController insertion of image failed");
        }
    }
}
```

JobHuntPostDAO.java

```
private static final String UPDATE_JOBHUNTPOST = "UPDATE JOBHUNTPOST "
+ "SET "
+ "JOBHUNTPOST_title = ?, "
+ "JOBHUNTPOST_content = ?, "
+ "JOBHUNTPOST_role = ?, "
+ "JOBHUNTPOST_region = ?, "
+ "JOBHUNTPOST_pay = ?, "
+ "JOBHUNTPOST_workDate = ?, "
+ "JOBHUNTPOST_concept = ? "
+ "WHERE "
+ "JOBHUNTPOST_id = ? ";
// 사용한 테이블 : 구직글 테이블
// 사용한 컬럼 (수정 내용) : 제목, 내용, 직업, 작업 지역, 작업 페이, 작업 날짜, 채용 컨셉
```

수정하기 페이지로 이동하면
postImgService.delete 동작

PostImgDAO.java

```
private static final String INSERT = "INSERT INTO POSTIMG (POST_id, POSTIMG_name) VALUES (?, ?)";
private static final String DELETE = "DELETE FROM POSTIMG WHERE POST_id = ?";
```

게시글 수정하기 (2/2)

```
UpdateJobHuntPostController Log02 = [Users/noreunghyeon/Documents/workspace_infinityStone/chalKag/src/main/resources/static/postImg]
JobHuntPostDAO(update) In로그 = [JobHuntPostDTO jobHuntpostId=null, memberId=qpqpqp9508@naver.com, memberNickname=null, memberIntroduction=null,
profileImgName=null, jobHuntPostDate=null, jobHuntPostRole=Model, jobHuntPostRegion=SEOUL, jobHuntPostPay=123, jobHuntPostWorkDate=2024-04-12,
jobHuntPostConcept=pictorial, jobHuntPostTitle=하이요, jobHuntPostContent=231321○L□○L□, jobHuntPostViewcnt=null, myRecommend=0, postId=null,
postImgName=null, postCategory=null, recommendCnt=0, titleAndContents=null, minPay=0, maxPay=0, fromday=null, today=null, startWorkDate=null,
endWorkDate=null, sortOrder=null, searchField=null, searchInput=null, searchCondition=jobHuntPostUpdate]
JobHuntPostDAO(update) Out로그 = [0]
UpdateJobHuntPostController Update of post failed
UpdateJobHuntPostController Delete of post images failed
PostImgDAO(insert) Out로그 = [1]
PostImgDAO(insert) Out로그 = [1]
PostImgDAO(insert) Out로그 = [1]
PostImgDAO(insert) Out로그 = [1]
UpdateJobHuntPostController Out Log
```

수정에 실패했지만 에러가 뜨지 않아
직접 찍어본 로그

UpdateJobHuntPostController.java

```
if (!jobHuntPostService.update(jobHuntPostDTO)) {
    System.out.println("UpdateJobHuntPostController Update of post failed");
}
postImgDTO.setPostId(jobHuntPostDTO.getJobHuntpostId());

if (!postImgService.delete(postImgDTO)) {
    System.out.println("UpdateJobHuntPostController Delete of post images failed");
}
```

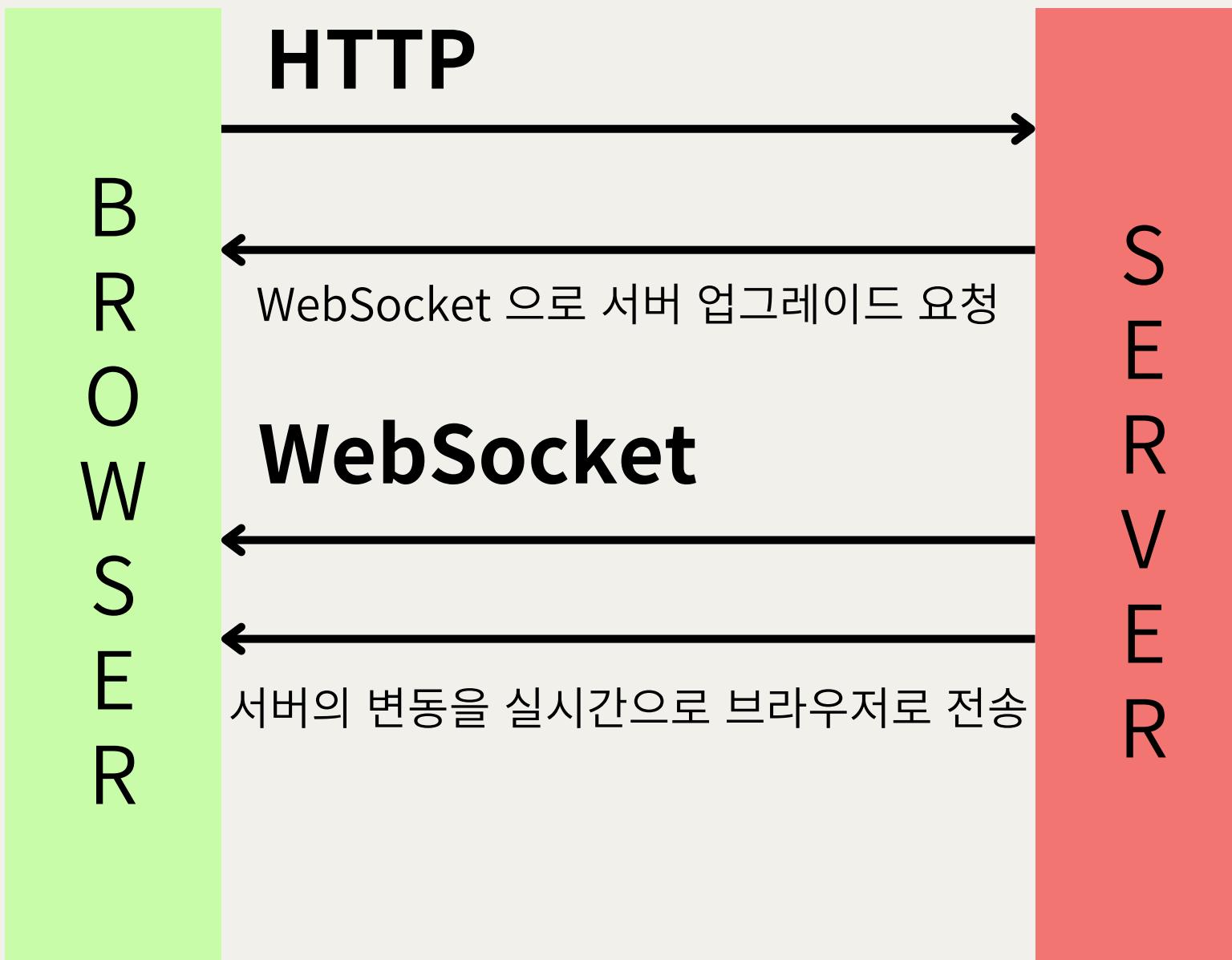
해당 포스트 멤버 아이디 값을
가져오지 못해 수정이 안되는 에러 발생

updateJobHuntPost.jsp

```
<form action="/updateJobHuntPost" method="post"
      onsubmit="return validateForm(event)"
      enctype="multipart/form-data">
    <!-- 글 번호 -->
    <input type="hidden" name="jobHuntpostId" value="${updateJobHuntPost.jobHuntpostId}" />
```

jobHuntpostId 를 넘겨서 해결

WebSocket 을 이용한 실시간 채팅 (1/6)



기존의 단방향 HTTP 프로토콜과 호환돼 양방향 통신을 제공하기 위해 개발된 프로토콜

HTTP 를 이용해 서버와 연결한 뒤
연결 상태를 유지하며
서버에 지속적으로 정보 전달

WebSocket 을 이용한 실시간 채팅 (2/6)

Stomp 

WebSocket 을 이용해
메시징 처리를 할 수 있게 도와주는 프로토콜

채팅방의 URL 을 사용자가 구독하고
같은 URL 을 구독한 사용자가
서버로 전송한 Text 를 동적으로 처리

WebSocket 을 이용한 실시간 채팅 (3/6)

build.gradle

```
implements 'org.springframework.boot:spring-boot-starter-websocket'
```

StompWebSocketConfig.java

```
package infinitystone.chalKag.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.messaging.simp.config.MessageBrokerRegistry;
import org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker;
import org.springframework.web.socket.config.annotation.StompEndpointRegistry;
import org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer;

@EnableWebSocketMessageBroker
@Configuration
public class StompWebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/stomp/chat")
            .setAllowedOrigins("http://localhost:8088")
            .withSockJS();
    }

    @Override
    public void configureMessageBroker(MessageBrokerRegistry registry) {
        registry.setApplicationDestinationPrefixes("/pub");
        registry.enableSimpleBroker("/sub");
    }

}
```

WebSocket 사용을 위한 의존 주입

EndPoint 설정

네트워크 설정에서
데이터 수신 지점을 설정하는 것이므로
자신의 어플리케이션 URL을
명확히 파악하는 것이 중요

WebSocket 을 이용한 실시간 채팅 (4/6)

RoomController.java

```
// 채팅방 개설
@RequestMapping(value = "/room", method = RequestMethod.POST)
public String createRoom(@RequestParam String name, RedirectAttributes rttr) {
    System.out.println("RoomController(room POST) In로그");
    rttr.addFlashAttribute("roomName", chatRoomDAO.createChatRoomDTO(name));
    System.out.println("RoomController(room POST) Out로그");
    return "redirect:/rooms";
}
```

RoomController 를 통해 채팅방 개설

WebSocket 을 이용한 실시간 채팅 (5/6)

sendMessage.js

```
if (writer === username) {
    console.log("로그 3");
    str = "<div class='col-6'>";
    str += "<div class='alert alert-secondary'>";
    str += "<b>" + writer + " : " + message + "</b>";
    str += "</div></div>";
    $("#msgArea").append(str);
    console.log("로그 4");
} else {
    str = "<div class='col-6'>";
    str += "<div class='alert alert-warning'>";
    str += "<b>" + writer + " : " + message + "</b>";
    str += "</div></div>";
    $("#msgArea").append(str);
}
});

//3. send(path, header, message)로 메세지를 보낼 수 있음
stomp.send('/pub/chat/enter', {}, JSON.stringify({roomId: roomId, writer: username}))
};

$("#button-send").on("click", function (e) {
    var msg = document.getElementById("msg");

    console.log(username + ":" + msg.value);
    stomp.send('/pub/chat/message', {}, JSON.stringify({roomId: roomId, message: msg.value, writer: username}));
    msg.value = '';
}),
});
```

Session에 저장된 정보와
WebSocket에 접속한
사용자 ID가 일치하는지 확인

특정 회원의 채팅방 접속을
알리는 메세지 전송

직접 작성한 메세지 또한 같은
방식으로 메세지 전송

WebSocket 을 이용한 실시간 채팅 (6/6)

STOMPChatController.java

```
@MessageMapping(value = "/chat/message")
public void message(ChatMessageDTO message) {
    template.convertAndSend("/sub/chat/room/" + message.getRoomId(), message);
}
```

@MessageMapping을 통해 WebSocket 으로 들어오는 메세지 발행

메세지가 발행되면 sub/chat/room/[roomId]로 메세지 전송

PortOne 을 이용한 결제 API (1/4)

payment.jsp

The diagram illustrates the payment process using PortOne's API. It consists of three main screens connected by arrows:

- Left Screen (User Account Boost Page):** A promotional page for "ACCOUNT BOOST". It highlights benefits like enhanced visibility, more engagements, and opportunities. A red box highlights the "boost" button at the bottom.
- Middle Screen (Payment Gateway):** A payment selection interface for "KG 이니시스". It shows various payment methods: 신용카드 (Credit Card), pay, N pay, 현대카드 (Hyundai Card), 삼성카드 (Samsung Card), KB국민 (KB国民), 비씨(페이북) (BC Pay), 신한카드 (Shinhan Card), 롯데카드 (Lotte Card), 하나Pay(하나) (Hana Pay), 능협(NH페이) (Nonghyup Pay), 씨티카드 (Citi Card), 하나Pay(외환) (Hana Pay Overseas), PAYCO, L.pay, SSGPAY, and 그외결제 (Other Payment Methods). A red arrow points from the "boost" button to this screen.
- Right Screen (Payment Confirmation):** A confirmation page for "CHALKAG BOOST" worth 9,900 원. It includes a "결제금액" (Payment Amount) section and a "렌탈하면 모바일 1년 무료" (Free mobile for 1 year if rented) offer. A red arrow points from the payment gateway screen to this confirmation screen.

The bottom of the middle screen features the text: "the new norm. More visibility leads to more interactions, and with it, more opportunities to grow".

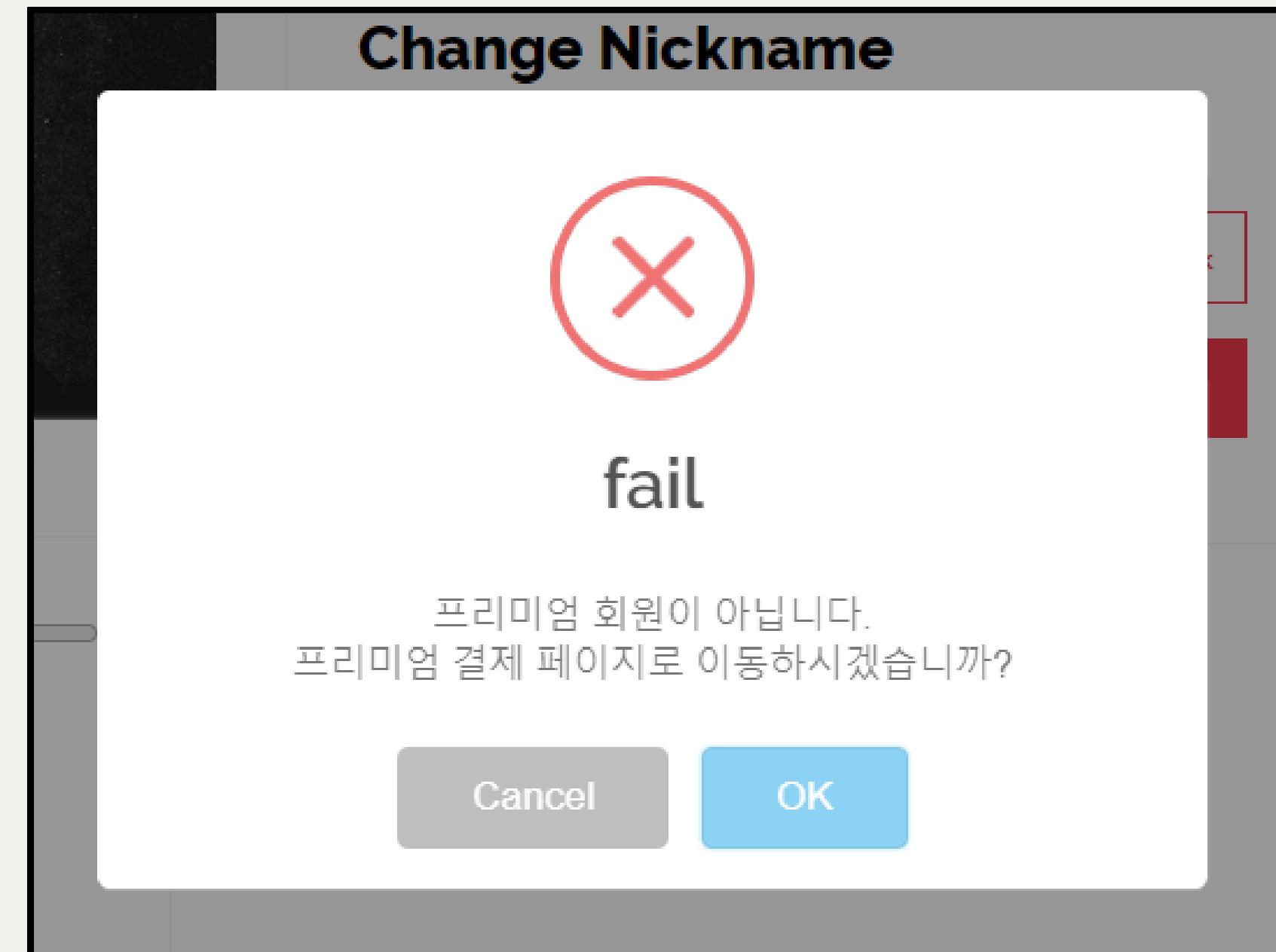
결제 방식을 특정화하지 않고 편리한 결제 제공을 위해 KG 이니시스 차용

PortOne 을 이용한 결제 API (2/4)

main.jsp



checkNickname.js



메인 페이지에 프리미엄 회원의 글 출력

무제한 닉네임 변경권 획득

PortOne 을 이용한 결제 API (3/4)

payment.js

```
console.log('로그1');
IMP.request_pay({
    pg: 'html5_inicis',
    pay_method: 'card',
    merchant_uid: Date.now(), // 상점에서 관리하는 주문 번호를 전달
    name: 'CHALKAG BOOST',
    amount: '9900',
    buyer_email: memberInfo.memberId,
    buyer_name: memberInfo.memberName,
    buyer_tel: formatPhoneNumber(memberInfo.memberPh) // - 값을 끌어줘야함
}, function (rsp) : void { // callback 로직
    if (rsp.success) {
        // [1] 서버단에서 결제정보 조회를 위해 jquery ajax로 imp_uid 전달. 안승준
        console.log(rsp);
        $.ajax({
            url: "payment",
            type: 'POST',
            dataType: 'json',
            data: {
                imp_uid: rsp.imp_uid,
                memberId: rsp.buyer_email,
                orderId: rsp.merchant_uid,
                amount: rsp.paid_amount
            }
        })
    }
})
```

결제에 필요한 정보 가공

API 요청 후
결제정보 조회를 위해
ajax로 정보 전달

PortOne 을 이용한 결제 API (4/4)

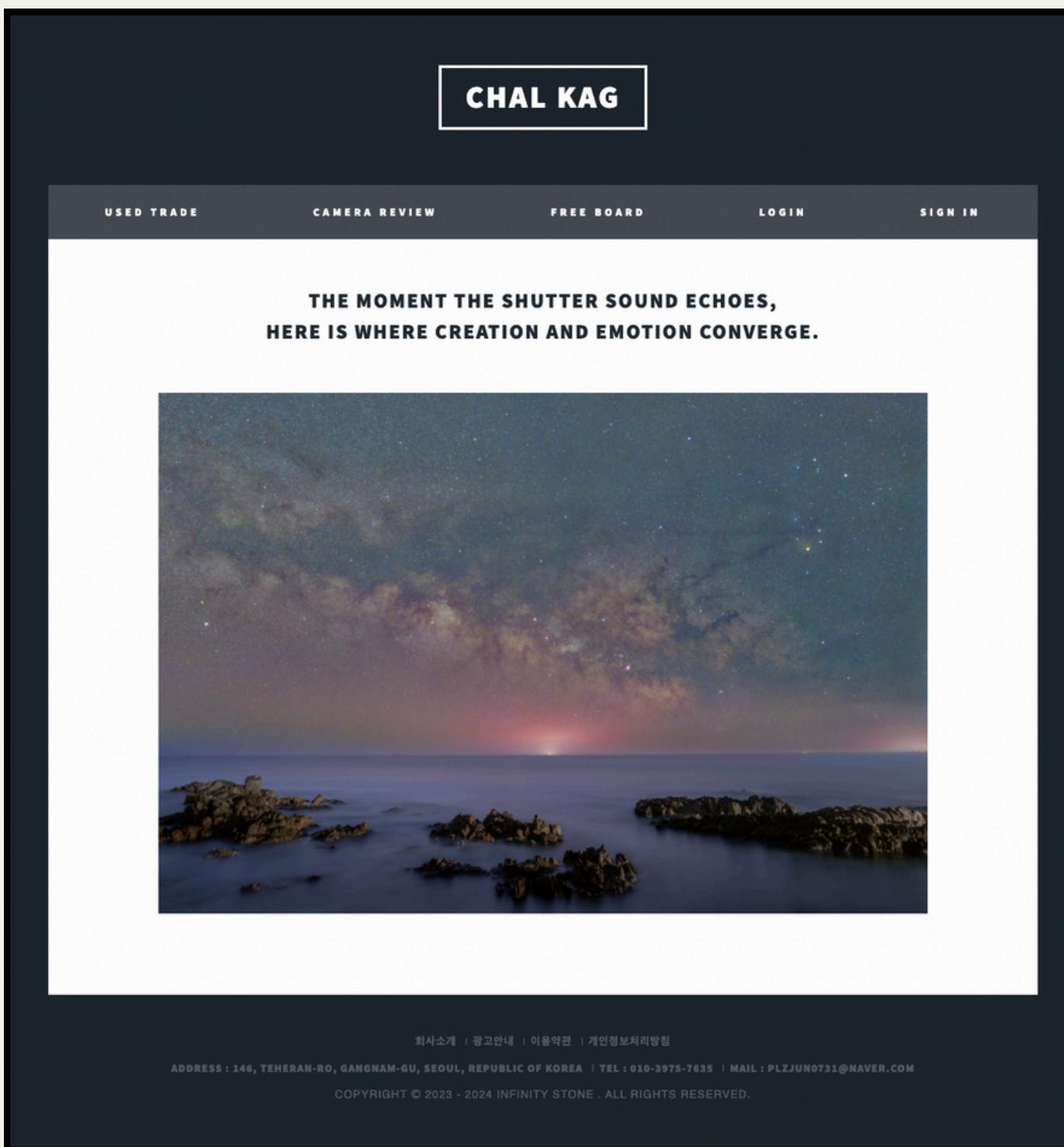
payment.js

```
}).done(function (data) : void {
    // [2] 서버에서 REST API로 결제정보 확인 및 서비스루틴이 정상적인 경우
    if (data) {
        swal({
            title: 'success',
            text: '결제가 완료되었습니다.',
            type: 'success',
            confirmButtonText: 'OK'
        }, function(isConfirmed) : void {
            if (isConfirmed) {
                window.location.href = 'main';
            }
        });
    } else {
        swal("fail", "결제를 취소했습니다", "error", {
            button: "OK",
        });
    }
});
```

비동기 처리로
결제 기록 저장 후
결제 완료 혹은 실패 알림

프로젝트 변경 및 추가 사항

View 템플릿 이관 (1/2)



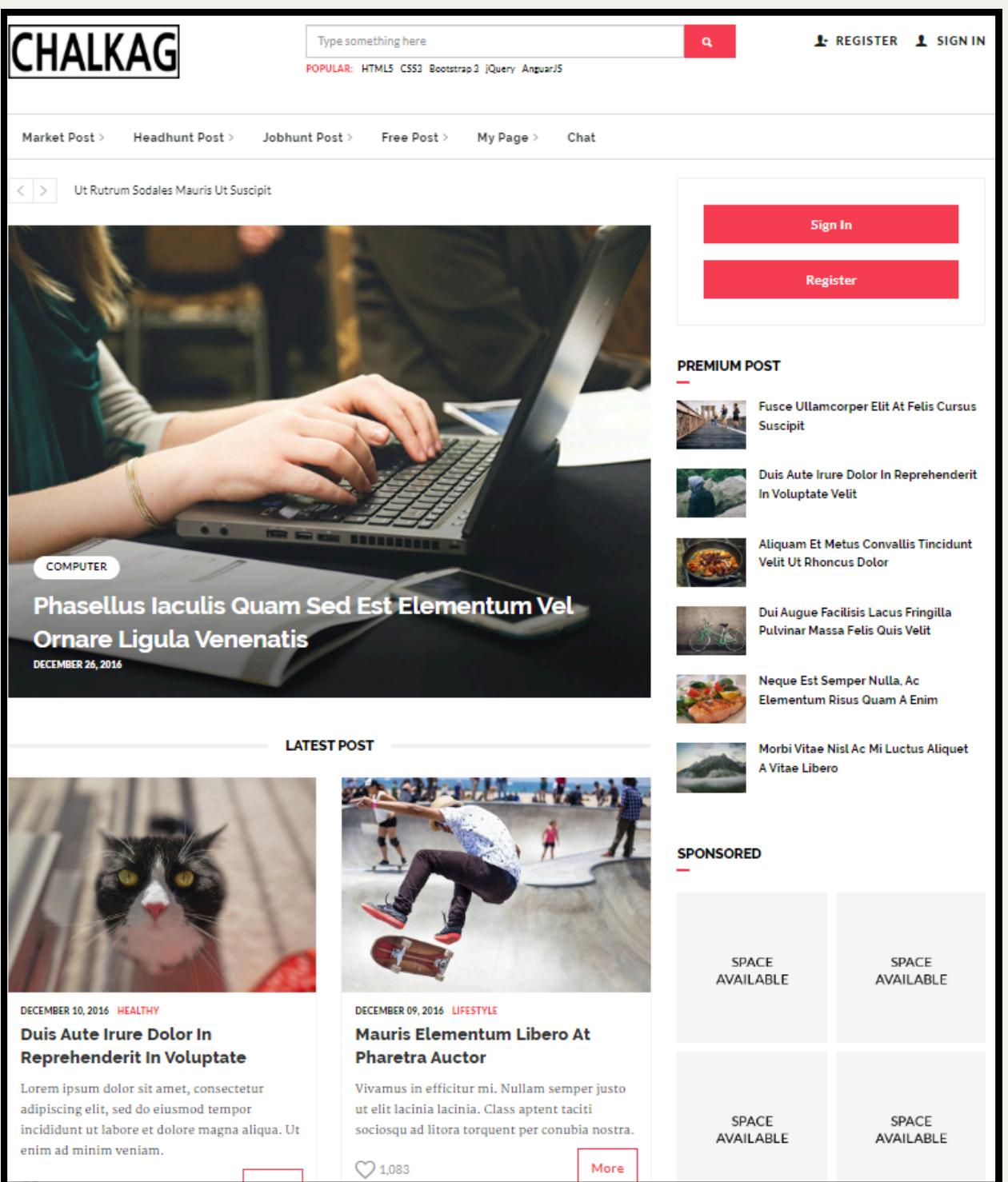
기존의 View 템플릿의 한계점 발견

기존의 템플릿은 반응형이 아니므로
직접 스타일을 추가해야 하는 불편함이 존재

이는 View 파트 개발 속도 저하로 인해
기능 개발의 디테일이 떨어진다고 판단

새로운 템플릿으로 이관 진행

View 템플릿 이관 (2/2)

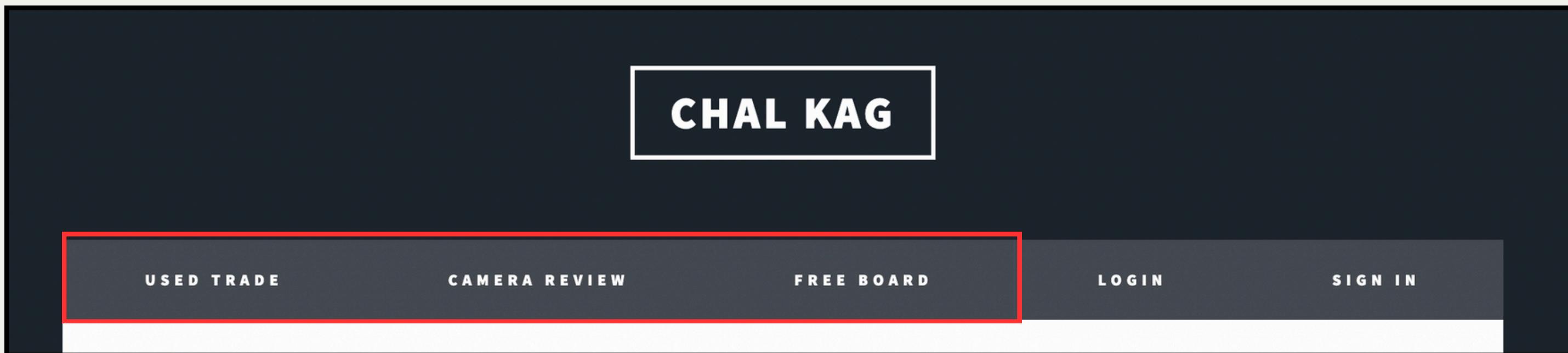


템플릿 이관 결과,

사용자들에게 익숙한 구성의 화면을 통해
사용자들이 더 많은 기능 이용 가능

스타일 추가가 최소화 되었고
개발의 진행 속도가 점진적으로 증가

사이트 내 이용 가능한 기능 추가 (1/3)



기존의 사이트는 중고거래 혹은 자유게시판을 통해 유저 간의 소통을 기대했음
하지만 사이트 자체가 단순하게 느껴진다는 피드백을 받음

또한 이런 게시판 구성은 팀원들이 프로젝트를 진행함에 있어
개발 능력 향상에 큰 도움이 되지 않는다고 판단

사이트 내 이용 가능한 기능 추가 (2/3)



A

기존의 중고거래 기능을 유지하되 팝니다, 삽니다, 무료나눔으로 구분되어 사용자들이 더 많은 거래를 경험

B.

기존의 카메라 리뷰 게시판을 자유 게시판에 통합
구인글과 구직글을 통해 사용자들끼리 작업 가능

C.

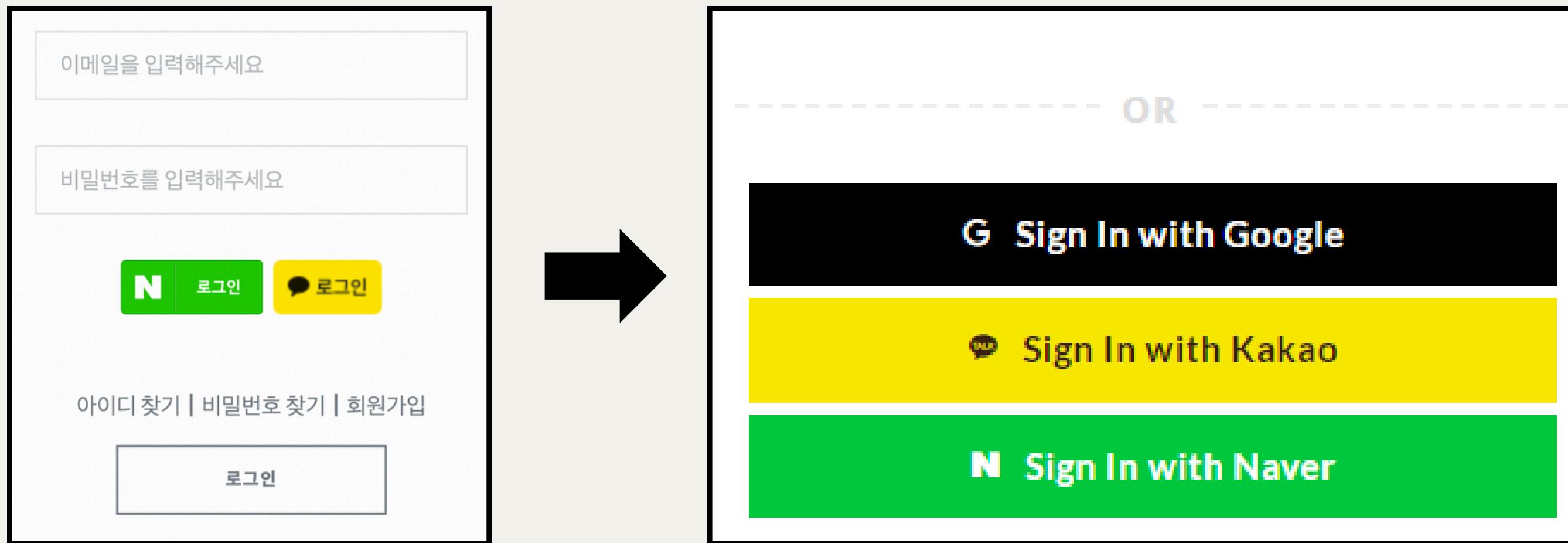
웹 소켓을 이용한 실시간 일대일 채팅 기능을 추가

사이트 내 이용 가능한 기능 추가 (3/3)

```
//구인글 상세 출력,전미지
private static final String SELECTONE_HEADHUNTPOST = "SELECT "
+ " 'HeadHuntPost' AS POST_category, " // 게시판의 카테고리 설정 "
+ " HEADHUNTPOST.HEADHUNTPOST_id, "
+ " HEADHUNTPOST.MEMBER_id, "
+ " MEMBER.MEMBER_nickname, "
+ " HEADHUNTPOST.HEADHUNTPOST_title, "
+ " HEADHUNTPOST.HEADHUNTPOST_content, "
+ " HEADHUNTPOST.HEADHUNTPOST_role, "
+ " HEADHUNTPOST.HEADHUNTPOST_region, "
+ " HEADHUNTPOST.HEADHUNTPOST_pay, "
+ " HEADHUNTPOST.HEADHUNTPOST_workDate, "
+ " HEADHUNTPOST.HEADHUNTPOST_concept, "
+ " HEADHUNTPOST.HEADHUNTPOST_date, "
+ " HEADHUNTPOST.HEADHUNTPOST_viewcnt, "
+ " (
+     SELECT "
+         PROFILEIMG.PROFILEIMG_name "
+     FROM "
+         PROFILEIMG "
+     WHERE "
+         PROFILEIMG.MEMBER_id = HEADHUNTPOST.MEMBER_id "
+     ORDER BY "
+         PROFILEIMG.PROFILEIMG_id DESC "
+     LIMIT 1 "
+ ) AS PROFILEIMG_name, "
+ " (
+     SELECT "
+         COUNT(*) "
+     FROM "
+         RECOMMEND "
+     WHERE "
+         RECOMMEND.POST_id = HEADHUNTPOST.HEADHUNTPOST_id "
+ ) AS RECOMMEND_cnt "
+ "FROM "
+ " HEADHUNTPOST "
+ "INNER JOIN "
+ " MEMBER ON HEADHUNTPOST.MEMBER_id = MEMBER.MEMBER_id "
+ "LEFT JOIN "
+ " RECOMMEND ON HEADHUNTPOST.HEADHUNTPOST_id = RECOMMEND.POST_id "
+ "WHERE "
+ " HEADHUNTPOST.HEADHUNTPOST_id = ? ";
// 사용한 테이블 : 구인글 테이블, 회원 테이블, 좋아요 테이블, 프로필 이미지 테이블
// 사용한 컬럼 (출력 내용) :
// 게시글 카테고리, 구인글 아이디, 회원 아이디, 회원 닉네임(회원 테이블), 구인글 제목, 구인글 내용,
// 구인글 직업 (모델/사진작가), 구인글 작업 지역, 구인글 작업 페이, 구인글 작업 날짜, 구인글 작업 컨셉,
// 구인글 작성일, 구인글 조회수, 게시글의 좋아요 수(좋아요 테이블), 프로필 이미지(프로필 이미지 테이블)
// 쿼리문 설명 :
// INNER JOIN을 사용해 구인글 테이블과 회원 테이블을 연결하고, 또 다른 LEFT JOIN을 사용해 구인글 테이블과 좋아요 테이블을 연결
// 프로필 이미지는 프로필 이미지 테이블을 따로 나누었으며 서브 쿼리를 사용해 회원의 프로필 이미지 중 회원정보로 보여줄 이미지를 설정하고,
// 그 결과를 "PROFILEIMG_name"라는 별칭으로 반환
// 게시글의 좋아요 수는 좋아요 테이블을 따로 나누었으며
// COUNT() 함수를 사용해 게시글에 대한 좋아요 수를 합산하고, 그 결과를 "RECOMMEND_cnt"라는 별칭으로 반환
```

다양한 기능으로 인해 기능 구현 시
서로의 상관 관계를 고민

구글 로그인 기능 추가 / 소셜 로그인 모듈화 (1/2)



구글 로그인 기능 추가로 사용자들의 편의성과 다양성 개선

구글 로그인 기능 추가 / 소셜 로그인 모듈화 (2/2)

```
@Service
public class CustomOAuth2UserService extends DefaultOAuth2UserService {
    //DefaultOAuth2UserService OAuth2UserService의 구현체

    private final UserRepository userRepository;

    public CustomOAuth2UserService(UserRepository userRepository) {
        this.userRepository = userRepository;
    }

    @Override
    public OAuth2User loadUser(OAuth2UserRequest userRequest) throws OAuth2AuthenticationException {
        OAuth2User oAuth2User = super.loadUser(userRequest);
        System.out.println(oAuth2User.getAttributes());

        String registrationId = userRequest.getClientRegistration().getRegistrationId();
        OAuth2Response oAuth2Response = null;
        if (registrationId.equals("naver")) {
            oAuth2Response = new NaverResponse(oAuth2User.getAttributes());
        } else if (registrationId.equals("google")) {
            oAuth2Response = new GoogleResponse(oAuth2User.getAttributes());
        } else if(registrationId.equals("kakao")) {
            oAuth2Response = new KakaoResponse(oAuth2User.getAttributes());
        }
        else {
            return null;
        }
        return oAuth2Response;
    }
}
```

JS를 활용한 소셜 로그인 기능에서 Spring Security 를 활용한 방법으로 구글 로그인 기능 추가 소셜 로그인 모듈화를 진행

DefaultOAuth2UserService 를 활용해서 토큰 교환을 직접적으로 진행하지 않아도 소셜 로그인이 가능하게 완성

로깅의 세부화 (1/2)

```
public MemberDTO selectOne(MemberDTO memberDTO) {
    MemberDTO data = null;
    conn = JDBCUtil.connect();

    if (memberDTO.getSearchCondition().equals("아이디찾기")) {
        try {
            pstmt = conn.prepareStatement(SELECTONE_FINDID);
            pstmt.setString(1, memberDTO.getName());
            pstmt.setString(2, memberDTO.getPh());
            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                data = new MemberDTO();
                data.setId(rs.getString("ID"));
            }
            rs.close();
        } catch (SQLException e) { 로그가 존재하지 않음
            e.printStackTrace();
        } finally {
            JDBCUtil.disconnect(pstmt, conn);
        }
    } else if (memberDTO.getSearchCondition().equals("비밀번호찾기")) {
        try {
            pstmt = conn.prepareStatement(SELECTONE_FINDPW);
            pstmt.setString(1, memberDTO.getId());
            pstmt.setString(2, memberDTO.getPh());
            ResultSet rs = pstmt.executeQuery();

            if (rs.next()) {
                data = new MemberDTO();
                data.setPw(rs.getString("PW"));
            }
            rs.close();
        } catch (SQLException e) {
    }
}
```

```
0
로그인 memberDTOMemberDTO [ ID=qpqpqp0001@naver.com, PW=1
grade=null]
```

기존 프로젝트 진행 중에는
로깅의 형식을 결정하지 않음

따라서 개발자 간의 로그 위치 및 형식이 달라
에러 발생 시 에러 확인을 위해 출력하는
단순한 로그에 그침

이는 알파테스트 시 디버깅 속도를 저하해
협업 능력을 감소

로깅의 세부화 (2/2)

```
public MemberDTO selectOne(MemberDTO memberDTO) {  
    MemberDTO result = null;  
    System.out.println("MemberDAO(selectOne) In로그 = [" + memberDTO + "]");  
    try {  
        if (memberDTO.getSearchCondition().equals("checkId")) {  
            Object[] args = {memberDTO.getMemberId()};  
            result = idbcTemplate.queryForObject(SELECTONE_CHECKID, args, new CheckIdRowMapper());  
            System.out.println("MemberDAO(selectOne) Out로그 = [" + result + "]");  
        }  
        return result;  
    } else if (memberDTO.getSearchCondition().equals("checkPh")) {  
        Object[] args = {memberDTO.getMemberPh()};  
    }  
    catch (Exception e) {  
        e.printStackTrace();  
        return null;  
    }  
    System.out.println("MemberDAO(selectOne) Error로그 = [" + memberDTO.getSearchCondition() + "]");  
    return null;  
}
```

로그 형식의 통일해 모든 팀원들이 서로 간의 로그를 확인하고 에러 발생 시 팀원들 간의 의사소통이 편리

이는 협업 능력을 향상 시키는 좋은 경험이 되어
개발 진행 속도 증가

오류 원인 분석과 해결 방안

회원가입 시 에러 (1/2)

```
-03-15T13:06:03.870+09:00 ERROR 16508 --- [nio-8088-exec-2] o.a.c.c.c.[.][/].[dispatcherServlet]    : Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request processing failed with status code 500: Internal Server Error] with root cause

java.lang.NullPointerException Create breakpoint : Cannot invoke "String.equals(Object)" because the return value of "infinitystone.chalkag.biz.member.MemberDTO.getSearchCondition()" is null AI로 설명 🤔
at infinitystone.chalkag.biz.member.MemberDAO.insert(MemberDAO.java:253) ~[main/:na] <4 내부 줄>
at org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:351) ~[spring-aop-6.1.4.jar:6.1.4]
at org.springframework.aop.framework.ReflectiveMethodInvocation.invokeJoinpoint(ReflectiveMethodInvocation.java:196) ~[spring-aop-6.1.4.jar:6.1.4]
at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:163) ~[spring-aop-6.1.4.jar:6.1.4]
at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:765) ~[spring-aop-6.1.4.jar:6.1.4]
at org.springframework.dao.support.PersistenceExceptionTranslationInterceptor.invoke(PersistenceExceptionTranslationInterceptor.java:137) ~[spring-tx-6.1.4.jar:6.1.4]
at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:184) ~[spring-aop-6.1.4.jar:6.1.4]
at org.springframework.aop.framework.CglibAopProxy$CglibMethodInvocation.proceed(CglibAopProxy.java:765) ~[spring-aop-6.1.4.jar:6.1.4]
at org.springframework.aop.framework.CglibAopProxy$DynamicAdvisedInterceptor.intercept(CglibAopProxy.java:717) ~[spring-aop-6.1.4.jar:6.1.4]
at infinitystone.chalkag.biz.member.MemberDAO$$SpringCGLIB$$0.insert(<generated>) ~[main/:na]
at infinitystone.chalkag.biz.member.MemberServiceImpl.insert(MemberServiceImpl.java:26) ~[main/:na]
at infinitystone.chalkag.controller.common.SignUpController.signUp(SignUpController.java:52) ~[main/:na] <14 내부 줄>
at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:547) ~[jakarta.servlet-api-6.0.0.jar:6.0.0] <1 내부 줄>
at jakarta.servlet.http.HttpServlet.service(HttpServlet.java:614) ~[jakarta.servlet-api-6.0.0.jar:6.0.0] <33 내부 줄>
```

회원 가입 후 DAO에서 insert를 실행하지 못하는 에러 발생
에러 메시지 확인 결과 getSearchCondition이 null임을 확인

회원가입 시 에러 (2/2)

```
public boolean insert(MemberDTO memberDTO) {
    System.out.println("MemberDAO(insert) In로그 = [" + memberDTO + "]");
    if (memberDTO.getSearchCondition().equals("signUp")) {
        if (jdbcTemplate.update(INSERT_SINGUP, memberDTO.getMemberId(), memberDTO.getMemberPw(), memberDTO.getMemberName()) < 1)
            return false;
    }
    return true;
}
System.out.println("MemberDAO(insert) Error로그 = [" + memberDTO.getSearchCondition() + "]");
return false;
}
```

로깅을 통해 Controller에서 searchCondition을 잘못 작성하는 상황 방지
모든 DAO, Controller에 In과 Out 로그를 작성
이동 요청이 정상적으로 이뤄지는지 확인

게시글 목록 보기 시 에러 (1/2)

```
1 // 필요한 변수 및 함수 선언
2 var filterData; // 필터 데이터를 저장하는 함수
3 var loadReviewData; // 데이터를 로드하는 함수
4 const dataContainer = document.getElementById('postDatasContainer'); // 데이터 컨테이너 요소를 가져옴
5 const postDatas = JSON.parse(dataContainer.getAttribute('displayReviewData'));
6 var currentPage = 0; // 첫 페이지
7 var totalPages = 0; // 총 페이지
8
9 // 현재 시간
10 const now = new Date();
```

```
✖ Failed to load resource: the server responded with a status of 404 () img14.jpg:1 ⓘ
✖ Failed to load resource: the server responded with a status of 404 () img11.jpg:1 ⓘ
✖ Failed to load resource: the server responded with a status of 404 () img10.jpg:1 ⓘ
postDatas : null headHuntPostPagination.js:6
✖ Failed to load resource: the server responded with a status of 404 () img10.jpg:1 ⓘ
✖ Failed to load resource: the server responded with a status of 404 () img14.jpg:1 ⓘ
✖ Failed to load resource: the server responded with a status of 404 () img11.jpg:1 ⓘ
```

페이지 처리시 js에서 JSON 타입으로 보내는 글 목록 데이터를 못 받아오는 오류 발생
데이터를 받아올때 로직에서 값을 받아올 수 없는 문제가 발생

게시글 목록 보기 시 에러 (2/2)

```
1 // 필요한 변수 및 함수 선언
2 var filterData; // 필터 데이터를 저장하는 함수
3 var loadReviewData; // 데이터를 로드하는 함수
4 const dataContainer = document.getElementById('postDatasContainer'); // 데이터 컨테이너 요소를 가져옴
5 const postDatas = JSON.parse(dataContainer.dataset.displayreviewdata); // 게시글 데이터를 가져옴
6 var currentPage = 0; // 첫 페이지
7 var totalPages = 0; // 총 페이지
8
9 // 현재 시간
10 const now = new Date();
11
```

getAttribute로 값을 받아오지 않고 dataset을 사용하여 문제를 해결

게시글 상세 보기 시 에러 (1/2)



The screenshot shows a Java exception stack trace in an IDE's terminal window. The error occurred at 2024-04-01T13:59:44Z. The exception is a `java.lang.NullPointerException` thrown from the `DispatcherServlet` during request processing. The cause of the exception is that `JobHuntPostDTO.getJobHuntPostId()` was called on a null object. The stack trace traces back through several layers of Spring Framework and Java base classes, ending at `Method.invoke()` and `InvocableHandlerMethod.doInvoke()`.

```
chakKag - ChakKagApplication [Spring Boot App] /Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java (2024. 4. 1. 오후 1:52:09) [pid: 29809]
jobHuntPostViewCnt=0, postId=null, postImgName=null, postCategory=JOBHuntPost, recommendCnt=0, searchCondition=null))
JobHuntPostListController Out로그
2024-04-01T13:59:44Z+09:00 ERROR 29809 --- [nio-8088-exec-1] o.a.c.c.C.[.].[dispatcherServlet] : Servlet.service() for servlet [dispatcherServlet] in
context with path [] threw exception [Request processing failed: java.lang.NullPointerException: Cannot invoke
"infinitystone.chakKag.biz.jobHuntPost.JobHuntPostDTO.getJobHuntPostId()" because "jobHuntPostDTO" is null] with root cause
java.lang.NullPointerException: Cannot invoke "infinitystone.chakKag.biz.jobHuntPost.JobHuntPostDTO.getJobHuntPostId()" because "jobHuntPostDTO" is null
at infinitystone.chakKag.controller.jobHuntPost.JobHuntPostSingleController.jobHuntPostList(JobHuntPostSingleController.java:36) ~[main/:na]
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method) ~[na:na]
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77) ~[na:na]
at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[na:na]
at java.base/java.lang.reflect.Method.invoke(Method.java:568) ~[na:na]
at org.springframework.web.method.support.InvocableHandlerMethod.doInvoke(InvocableHandlerMethod.java:259) ~[spring-web-6.1.4.jar:6.1.4]
```

게시글 상세 보기 페이지로 이동하면
JobHuntPostDTO is null 이 발생하는 에러를 확인

게시글 상세 보기 시 에러 (2/2)

```
④ @Override
public JobHuntPostDTO mapRow(ResultSet rs, int rowNum) throws SQLException {
    JobHuntPostDTO data = new JobHuntPostDTO();

    data.setPostCategory(rs.getString("POST_category"));
    data.setJobHuntPostId(rs.getString("JOBHUNTPOST_id"));
    data.setMemberId(rs.getString("MEMBER_id"));
    data.setMemberNickname(rs.getString("MEMBER_nickname"));
    data.setJobHuntPostDate(rs.getString("JOBHUNTPOST_date"));
    data.setJobHuntPostRole(rs.getString("JOBHUNTPOST_role"));
    data.setJobHuntPostRegion(rs.getString("JOBHUNTPOST_region"));
    data.setJobHuntPostPay(rs.getInt("JOBHUNTPOST_pay"));
    data.setJobHuntPostConcept(rs.getString("JOBHUNTPOST_workDate"));
    data.setJobHuntPostConcept(rs.getString("JOBHUNTPOST_concept"));
    data.setJobHuntPostTitle(rs.getString("JOBHUNTPOST_title"));
    data.setJobHuntPostContent(rs.getString("JOBHUNTPOST_content"));
    data.setJobHuntPostViewcnt(rs.getString("JOBHUNTPOST_viewcnt"));
    // data.setPostImgName(rs.getString("POSTIMG_name"));
    return data;
}
```

POSTIMG_name 은 조인해서 값을 불러오는 게 아니라
정규화로 인해 이미지 테이블이 따로 있기 때문에
rowMapper 에 없는 값을 작성해 NPE 가 뜬 상황
postImgName 삭제해 해결

프로필 이미지 출력 시 에러 (1/3)

Role	모델
Region	GYEONGGI
Pay	4,548
Work Date	2024-04-08
Concept	outdoor

4/5 ~ 4/8 일중 커플 컨셉 촬영 가능한 남, 여 모델분들 찾고 있습니다 . 과제 목적으로 작업 예정이며 추후 개인 포트폴리오로도 사용 예정입니다. 페이는 무페이로 생각하고 있으나 협의 가능하니 편히 편히 말씀해 주세요 :) 촬영 컨셉은 시안 확인 부탁드리며 문의는 채팅으로 부탁드리겠습니다.



AUTHOR

하치와레

I enjoy capturing small moments in everyday life with my camera as a hobby. I strive to capture the beauty of daily life through photos so that everyone can share in those precious moments together. Let's share these precious moments together!

게시글 상세 보기 시
프로필 이미지를 불러오지 못하는 에러 발생

프로필 이미지 출력 시 에러 (2/3)

```
HeadHuntPostListController Out로그  
HeadHuntPostDAO(selectOne) In로그 = [HeadHuntPostDTO [headHuntPostId=2020, memberId=null, memberNickname=null, profileImgName=null, headHuntPostDate=null, headHuntPostRole=null, headHuntPostRegion=null, headHuntPostPay=0, headHuntPostWorkDate=null, headHuntPostConcept=null, headHuntPostTitle=null, headHuntPostContent=null, headHuntPostViewcnt=null, postId=null, postImgName=null, postCategory=null, recommendCnt=null, searchCondition=headHuntPostSingle]]  
org.springframework.dao.IncorrectResultSizeDataAccessException: Incorrect result size: expected 1, actual 4  
at org.springframework.dao.support.DataAccessUtils.getSingleResult(DataAccessUtils.java:193)  
at org.springframework.jdbc.core.JdbcTemplate.queryForObject(JdbcTemplate.java:890)  
at infinitystone.chalKag.biz.headHuntPost.HeadHuntPostDAO.selectOne(HeadHuntPostDAO.java:414)  
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)  
at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)  
at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  
at java.base/java.lang.reflect.Method.invoke(Method.java:568)  
at org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:351)  
... . . . . .
```

프로필 이미지 변경 시 기존 이미지는 삭제하지 않고 추가로 저장되도록 설계 했으므로
프로필 이미지가 여러 장일 경우 최근에 변경된 이미지를 불러오지 못함

프로필 이미지 출력 시 예러 (3/3)

```
// 구인글 상세 출력.전미지
private static final String SELECTONE_HEADHUNTPOSTSINGLE = "SELECT "
+ "    'HeadHuntPost' AS POST_category, " // 게시판의 카테고리 설정 "
+ "    HEADHUNTPOST.HEADHUNTPOST_id, "
+ "    HEADHUNTPOST.MEMBER_id, "
+ "    MEMBER.MEMBER_nickname, "
+ "    HEADHUNTPOST.HEADHUNTPOST_title, "
+ "    HEADHUNTPOST.HEADHUNTPOST_content, "
+ "    HEADHUNTPOST.HEADHUNTPOST_role, "
+ "    HEADHUNTPOST.HEADHUNTPOST_region, "
+ "    HEADHUNTPOST.HEADHUNTPOST_pay, "
+ "    HEADHUNTPOST.HEADHUNTPOST_workDate, "
+ "    HEADHUNTPOST.HEADHUNTPOST_concept, "
+ "    HEADHUNTPOST.HEADHUNTPOST_date, "
+ "    HEADHUNTPOST.HEADHUNTPOST_viewcnt, "
+ "    ( " // 회원 프로필로 보여줄 이미지를 선택
+ "        SELECT "
+ "            PROFILEIMG.PROFILEIMG_name " // 회원의 프로필 이미지를 선택
+ "        FROM "
+ "            PROFILEIMG " // 프로필 이미지 테이블에서 가져옴
+ "        WHERE "
+ "            PROFILEIMG.MEMBER_id = HEADHUNTPOST.MEMBER_id "
+ "        ORDER BY "
+ "            PROFILEIMG.PROFILEIMG_id DESC "
+ "        LIMIT 1 " // 이미지를 1개만 가져오도록 설정
+ "    ) AS PROFILEIMG_name, " // 가져온 프로필 이미지의 이름을 "PROFILEIMG_name"으로 칭함
+ "    ( " // 게시글의 좋아요 수를 합산
+ "        SELECT "
+ "            COUNT(*) " // 해당 구인글에 대한 좋아요 수를 COUNT 함수를 사용해 합산
+ "        FROM "
+ "            RECOMMEND " // 좋아요 테이블에서 가져옴
+ "        WHERE "
+ "            RECOMMEND.POST_id = HEADHUNTPOST.HEADHUNTPOST_id " // 해당 구인글과 연관된 좋아요 수
+ "    ) AS RECOMMEND_cnt " // 좋아요 수를 "RECOMMEND_cnt"로 칭함
+ "FROM "
+ "    HEADHUNTPOST " // 구인글 테이블에서 가져옴
+ "INNER JOIN "
+ "    MEMBER ON HEADHUNTPOST.MEMBER_id = MEMBER.MEMBER_id "
+ "LEFT JOIN "
+ "    RECOMMEND ON HEADHUNTPOST.HEADHUNTPOST_id = RECOMMEND.POST_id "
+ "WHERE "
+ "    HEADHUNTPOST.HEADHUNTPOST_id = ? " ;
```

프로필 이미지 출력 시
저장되어 있는 사진 중
가장 최근에 저장된
사진 1장만 가져오도록
쿼리문을 수정

관리자 사진 변경 예러 (1/2)

```
var h2Contents = [];
document.querySelectorAll('#mainImg .item h2').forEach(function (h2) {
    h2Contents.push(h2.innerHTML);
});
console.log(h2Contents.length); // 8 출력
```

선택된 <h2> 요소들의 내용을 h2Contents 배열에 저장하고
다른 내용으로 바꾸는 과정에서 원하는 대로 바꿔지지 않아
로그를 확인해보니 4개의 <h2> 요소 밖에 없음에도 불구하고 8출력

관리자 사진 변경 예러 (2/2)

```
<div class="owl-item cloned" style="width: 750px;">
  <div class="item">
    <figure>...</figure>
    <h2>...</h2>
  </div>
</div>

> <div class="owl-item cloned" style="width: 750px;">...</div>
> <div class="owl-item" style="width: 750px;">...</div>
> <div class="owl-item" style="width: 750px;">...</div>
> <div class="owl-item" style="width: 750px;">...</div>
> <div class="owl-item active" style="width: 750px;">...</div>
> <div class="owl-item cloned" style="width: 750px;">...</div>
> <div class="owl-item cloned" style="width: 750px;">...</div>
```

확인 결과 OwlCarousel 의 loop: true 를 사용할 경우
무한 슬라이드를 위한 각각의 복제요소 생성
복제요소를 포함한 <h2> 개수가 8개

cloned 클래스를 가지고 있지 않은
<h2> 태그의 내용만 배열에 저장하여 오류 해결

```
// 기존 캐러셀 내용 중 <h2> 태그의 내용을 배열로 저장
var h2Contents = [];
// loop 가능 사용시 관리자 개발도구를 확인해보면 복제된 요소 존재
// not(.cloned)를 사용해 복제된 요소 제외하고 선택
document.querySelectorAll('.owl-item:not(.cloned) .item h2').forEach(function (h2) {
  h2Contents.push(h2.innerHTML);
});
```

기능 추가 및 추후 개발 방향

- AWS 를 이용한 실제 서버 동작
- 프리미엄 회원들의 기능 및 권한 추가
- Spring Security 를 통한 사이트 보안성 강화

THANK YOU

감사합니다



정석진

Tel. 010-6622-1689
Email. ddoljin@gmail.com
Blog. <https://varietyofit.tistory.com>
Github. <https://github.com/FullStackJinnnn>