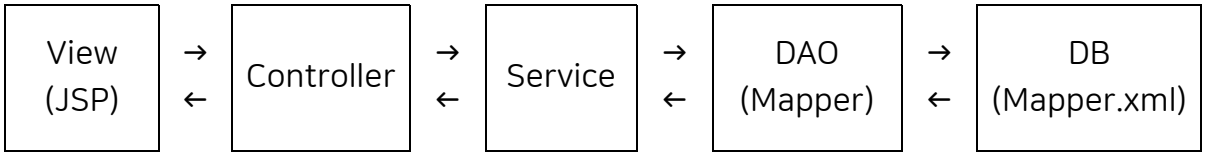
프로젝트 기술서

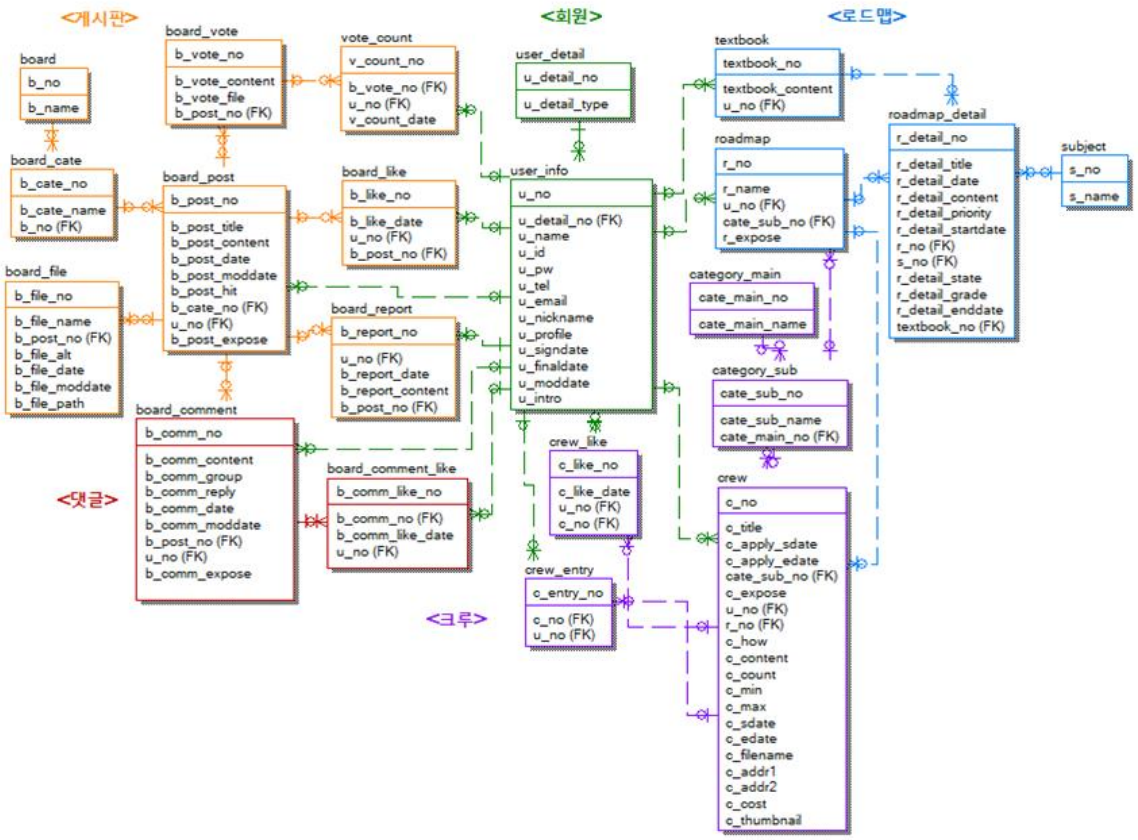
프로젝트 명	Triple A		
포트폴리오 주소	https://github.com/kmhjn33/WEB_pjt.git		
프로젝트 개요	- 사용자가 직접 학습 계획을 작성하고 Drag&Drop으로 진행 상태를 변경하며 관리할 수 있는 스케줄러(칸반 보드) 기능을 갖춘 중·고등학생 대상의 학습 정보 공유 커뮤니티 - 코로나로 인해 원격 수업이 진행된 이후로 중·고등학생의 자기주도적 학습 능력과 또래 간 의사소통이 감소했다는 조사 결과를 바탕으로 학습 스케줄러와 소통 공간을 제공하는 웹 서비스를 제작함		
개발기간	2021.11.11 - 2021.12.06	투입인원	5명
주요 기술	- Java - Spring Boot - MyBatis - MVC2 모델 적용 - JavaScript / jQuery / Ajax - Oracle 11g - 오픈 API (카카오 지도, 구글 이메일 인증)		
본인의 역할	- 팀장으로서 전체 작업 스케줄 관리, 업무 분배 - 게시판 영역의 DB 설계 - 회원 전용 게시판의 게시글/댓글 읽기, 쓰기, 수정, 삭제, 추천 기능 구현 - 게시글 검색, 정렬, 분류, 페이징 구현 - 쿠키를 이용한 게시글 조회수 중복 불가 처리 - 로드맵(해당 프로젝트의 학습 스케줄 관리 컨텐츠)에서 HTML요소에 Drag&Drop 이벤트 발생 시 데이터 베이스 UPDATE 구현 - 마이페이지 내 게시글 관리의 내가 쓴 글/댓글, 추천한 게시물 리스트 확인, 해당 페이지로 이동		

■ **기본 구성 및 흐름**



김희진 | 010-7793-9833 | kmhjn33@gmail.com

■ DB ERD



■ **게시판 정렬, 분류, 검색, 페이징 처리**

✓ View

- 게시판 번호를 Pathvariable로 전달
- 검색 분류, 검색어, 카테고리 분류, 정렬 기준, 현재 페이지를 쿼리스트링으로 전달

```
/board/list/1?sort=view&searchCate=title&searchWord=등급&page=1&b_cate_no=1
```

✓ Controller

- 데이터들을 받아서 Service로 넘겨주고 리턴 받은 Map을 Model에 태움

✓ ServiceImpl

- 현재 게시판의 카테고리를 List로 가져옴
- 검색/분류/정렬이 적용된 전체 글 개수를 가져옴
  => 페이지 계산에 이용
- 검색/분류/정렬/현재 페이지에 맞는 게시글을 List로 가져옴
- 게시판 카테고리와 게시글을 Map에 담아서 Controller로 리턴

김희진 | 010-7793-9833 | kmhjn33@gmail.com

✓ Mapper

```
<!-- ============================ 검색 조건 ================================ -->
<sql id="boardSearch">
    <if test="searchCate != ''">
        <choose>
            <when test="searchCate.equals('title')">
                and bp.b_post_title like '%'||#{searchWord}||'%'
            </when>
            <when test="searchCate.equals('content')">
                and bp.b_post_content like '%'||#{searchWord}||'%'
            </when>
            <when test="searchCate.equals('writer')">
                and ui.u_nickname like '%'||#{searchWord}||'%'
            </when>
        </choose>
    </if>
</sql>
<!-- ============================ //검색 조건 ================================ -->

<!-- ============================ 카테고리 분류 ================================ -->
<sql id="boardCategorizing">
    <if test="b_cate_no != 0">
        and bp.b_cate_no = ${ b_cate_no }
    </if>
</sql>
<!-- ============================ //카테고리 분류 ================================ -->

<!-- ============================ 정렬 조건 ================================ -->
<sql id="boardSort">
    <choose>
        <when test="sort.equals('date')">
            order by bp.b_post_date desc
        </when>
        <when test="sort.equals('view')">
            order by bp.b_post_hit desc, bp.b_post_date desc
        </when>
        <when test="sort.equals('like')">
            order by count(distinct bl.b_like_no) desc, bp.b_post_date desc
        </when>
    </choose>
</sql>
<!-- ============================ //정렬 조건 ================================ -->
```

```
<!-- 게시글 갯수 가져오기 -->
<select id="selectBoardListCount" resultType="int">
        select count(*)
        from   board_post bp, user_info ui
        where  bp.u_no = ui.u_no and
               bp.b_cate_no in (select b_cate_no from board_cate where b_no = ${b_no}) and
               bp.b_post_expose = 'Y'
                  <!-- 검색 적용 -->
               <include refid="boardSearch" />
               <!-- 카테고리 분류 적용 -->
               <include refid="boardCategorizing" />
</select>
```

```
<!-- 전체 게시글 가져오기 -->
<select id="selectBoardList" resultType="com.site.team1.vo.BoardListPost">

    select postList.*
    from (
            select post.*, rownum as rnum
            from (
                    select bp.b_post_no as b_post_no,
                           bp.b_post_title as b_post_title,
                           bp.b_post_date as b_post_date,
                           bp.b_post_hit as b_post_hit,
                           bcate.b_cate_name as b_cate_name,
                           count(distinct bl.b_like_no) as b_like_count,
                           count(distinct (case when bcomm.b_comm_expose='Y' then 1 end)) as b_comm_count,
                           ui.u_nickname as u_nickname

                    from   board_post bp, board_cate bcate, board_like bl, user_info ui, board_comment bcomm

                    where  bp.b_cate_no = bcate.b_cate_no and
                           bp.b_post_no = bl.b_post_no(+) and
                           bp.b_post_no = bcomm.b_post_no(+) and
                           bp.u_no = ui.u_no and
                           bp.b_cate_no in (select b_cate_no from board_cate where b_no = ${b_no}) and
                           bp.b_post_expose = 'Y'
                           <!-- 검색 적용 -->
                           <include refid="boardSearch" />
                           <!-- 카테고리 분류 적용 -->
                           <include refid="boardCategorizing" />

                    group by bp.b_post_no, bp.b_post_title, bp.b_post_date, bp.b_post_hit, ui.u_nickname, bcate.b_cate_name

                    <!-- 정렬 적용 -->
                    <include refid="boardSort" />

                ) post
            ) postList
    <![CDATA[
    where postList.rnum >= ${boardPage.startRow} and postList.rnum <= ${boardPage.endRow}
    ]]>

</select>
```

■ 게시글 조회 (중복 불가)

✓ View

- 게시판 번호를 Pathvariable로 전달
- 검색 분류, 검색어, 카테고리 분류, 정렬 기준, 현재 페이지, 선택한 게시글 번호를 쿼리스트링으로 전달

```
/board/view/2?sort=date&searchCate=&searchWord=&page=1&b_cate_no=0&b_post_no=152
```

✓ Controller

- CookieValue 어노테이션을 이용해 방문한 게시글 번호를 저장한 쿠키를 받아옴
- 이전에 생성된 쿠키가 없는 경우 : 쿠키를 생성하고 조회한 게시글 번호를 담은 후 DB의 조회수를 업데이트함
- 쿠키가 있는 경우 : 조회했던 게시글 번호가 '/' 기호와 함께 나열되어 String 타입으로 저장되어 있으므로 split 메서드로 분리해 현재 조회한 게시글 번호와 대조함. 일치하지 않는 경우 번호를 추가하고 조회수를 증가시킴, 일치할 경우 변동 없음

```java
//글 보기
@RequestMapping(value = {"/view/{b_no}", "/view"})
public String boardView(BoardMaker boardMaker, BoardPostVo boardPostVo,
                        @CookieValue(name = "visitList", required = false) Cookie cookie,
                        Model model, HttpServletResponse response) {

    String postNo = String.valueOf(boardPostVo.getB_post_no());

    //게시글 조회 목록 쿠키가 없을 경우
    if(cookie == null) {
        //쿠키 생성 후 조회 게시글 번호 추가
        cookie = new Cookie("visitList", postNo);
        //게시글 조회수 증가
        int result = boardService.boardViewCountUp(boardPostVo);

    } else {
        //게시글 조회 목록 쿠키가 있을 경우
        String[] vList = cookie.getValue().split("/");
        Boolean isVisit = false;

        //게시글 조회 목록과 현재 게시글 번호 대조
        for(String visitPost : vList) {
            if (visitPost.equals(postNo)) {
                isVisit = true;
                break;
            }//if-end
        }//for-end

        //게시글 첫 조회일 경우 조회 목록에 게시글 번호 추가, 게시글 조회수 증가
        if(!isVisit) {
            String newVisitList = cookie.getValue() + "/" + postNo;
            cookie.setValue(newVisitList);

            int result = boardService.boardViewCountUp(boardPostVo);
        }//if-end
    }//if-else-end

    //해당 글, 이전글, 다음글, 인기글, 댓글, 추천 정보 가져옴
    boardMap = boardService.boardView(boardMaker, boardPostVo);

    model.addAttribute("boardMap", boardMap);
    response.addCookie(cookie);

    return "/board/view";
}
```

✓ ServiceImpl

- 게시글 조회 화면을 구성할 때 필요한 글 내용, 댓글, 이전 글, 다음 글을 가져옴
- DB를 통해 가져온 데이터를 Map에 담아 리턴

✓ Mapper

```
<!-- 이전글 가져옴 -->
<select id="selectBoardPostPrev" resultType="com.site.team1.vo.BoardListPost">

    select bp.b_post_no, bp.b_post_title, count(distinct bcomm.b_comm_no) as b_comm_count
    from   board_post bp, board_comment bcomm
    where bp.b_post_no = bcomm.b_post_no(+) and
          bp.b_post_no = (
                          select prev
                          from (
                                  select b_post_no ,
                                         lag(b_post_no) over (order by b_post_date desc) as prev
                                  from board_post
                                  where b_cate_no in (select b_cate_no from board_cate where b_no = ${boardMaker.b_no}) and
                                        b_post_expose = 'Y'
                               )
                          where b_post_no = ${boardPostVo.b_post_no}
                        )
    group by bp.b_post_no, bp.b_post_title

</select>
```

■ Drag&Drop 이벤트 이후의 DB 데이터 변동

✓ View

- Drag 이벤트가 발생한 HTML요소의 id값과 Drop시 타겟이 된 HTML요소의 id값을 Ajax으로 넘겨줌

```
// 드래그 앤 드롭
// 드래그 시작
function onDragStart(ev){
   ev.dataTransfer.setData("id",ev.target.id);
};

function allowDrop(ev){
    ev.preventDefault();
};

function onDrop(ev){
    ev.preventDefault();
    var r_detail_no = ev.dataTransfer.getData("id");
    const draggableElement = document.getElementById(r_detail_no);
    const dropzone = ev.target;

    var target_id = $(ev.target).attr("class");
    if (target_id == "drop_zone") {
        dropzone.appendChild(draggableElement);
        draggableElement.setAttribute("onclick", "roadmap_detail(r_detail_no)");

        //로드맵 세부 박스 id (r_detail_no) => r_detail_no
        var detailState = $(ev.target).attr("id");
        dragDropInsert(r_detail_no, detailState);
    }

};
```

```
//로드맵 세부 드롭 후 DB 상태 업데이트
function dragDropInsert (r_detail_no, detailState) {
    var r_detail_state = '';

    switch(detailState) {
        case 'todopend' : r_detail_state = '시작 전';
            break;
        case 'doingpend' : r_detail_state = '진행 중';
            break;
        case 'donepend' : r_detail_state = '완료';
            break;
        case 'trashpend' : r_detail_state = '상태 없음';
            break;
    }

    $.ajax({
         url:"./detailroadmapDrag",
         type:"post",
         data:{
             "r_detail_no":r_detail_no,
             "r_detail_state": r_detail_state
         },
         success:function(data){
             if(data == 1) {
                //카운트 처리
                count_state();
             }
         },
         error:function(textStatus){
             alert("오류가 발생했습니다. 다시 시도해주세요.");
         }
      });//ajax-end
}
```

✓ Controller

- ResponseBody 어노테이션을 이용해 DB 업데이트 후 성공 여부 값을 넘겨줌

```
//로드맵 드래그 앤 드롭
@ResponseBody
@RequestMapping("/detailroadmapDrag")
public int detailroadmapDrag(RoadmapDetailVo roadmapDetailVo) {
    //로드맵 세부 상태 변경
    int result = roadmapDetailService.detailroadmapDrag(roadmapDetailVo);
    return result;
}
```

✓ ServiceImpl, Mapper

- DB 업데이트

■ 게시글/댓글 쓰기, 수정, 삭제

✓ View

- 게시글 : form 데이터를 post 방식으로 submit
- 댓글/답댓글 : textarea의 데이터를 Ajax으로 넘김, 답댓글일 경우 1을 세팅해서 댓글과 구분되어 저장될 수 있게끔 함

✓ Controller

- 로그인 시 세션에 저장해놓은 사용자의 고유 번호를 vo에 세팅한 후 service로 넘겨줌
- 저장된 게시글 번호를 리턴 받아 작성한 게시물의 view 페이지로 이동시킴

```
@Controller
@RequestMapping("/board")
public class BoardController {

    @Autowired
    HttpSession session;
```

```
//글 작성 완료
@PostMapping("/write")
public String boardWrite(BoardMaker boardMaker, BoardPostVo boardPostVo) {

    //세션 유저 번호 세팅
    boardPostVo.setU_no((Integer)session.getAttribute("session_u_no"));

    //글 작성
    int result = boardService.boardWrite(boardPostVo);

    return "redirect:/board/view/" + boardMaker.getB_no() + "?b_post_no=" + boardPostVo.getB_post_no();
}
```

✓ ServiceImpl

- 게시글을 저장 또는 수정한 후 selectKey를 이용해 작성된 글의 시퀀스 값을 리턴함
- 댓글을 저장 또는 수정하고 해당 글의 전체 댓글 리스트를 가져옴

✓ Mapper

```
<!-- 작성한 게시물 저장 -->
<insert id="insertBoardWrite">
    <selectKey keyProperty="b_post_no" resultType="int" order="BEFORE">
        select board_post_seq.nextval as b_post_no from dual
    </selectKey>
    insert into board_post
    values (${b_post_no},
            #{b_post_title},
            #{b_post_content},
            sysdate,
            sysdate,
            0,
            ${b_cate_no},
            ${u_no},
            'Y'
           )
</insert>
```

■ 게시글/댓글 추천

✓ View
- 추천한 게시글 번호/댓글 번호를 넘겨줌

✓ Controller
- 세션의 유저 번호를 vo에 세팅
- insert 또는 delete의 결과를 리턴함

✓ ServiceImpl
- 해당 글의 추천 정보를 불러온 후 결과를 리턴 받음
- 리턴 받은 결과가 없으면 데이터 삽입, 결과가 있을 경우 데이터 삭제