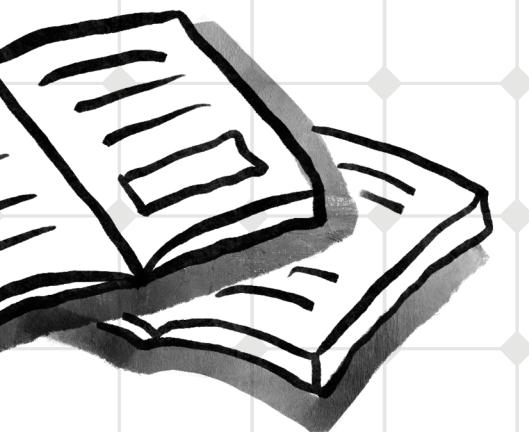


The image shows a decorative banner with stylized Korean text. The top part features the word "살뺀조" in large, bold, black-outlined letters with a hand-drawn texture. Below it, a single vertical character "(" is positioned. The bottom part contains several words arranged in two rows: "온도과식단기로" and "칼로리케산서비스". Each word is composed of large, rounded, black-outlined letters with a hand-drawn texture. The entire banner is set against a light gray background with a subtle grid pattern.



# 목차

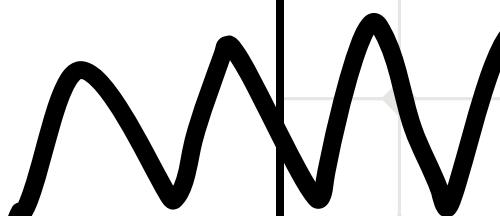
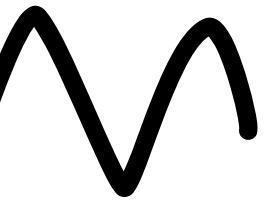
1. 팀원 소개 및 역할
2. 프로젝트 개요
3. 개발환경
4. 프로젝트 진행 과정
5. 코드리뷰
6. 자체평가 및 느낀점
7. Q&A

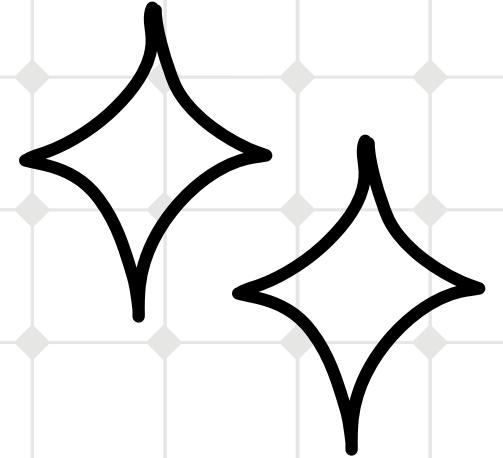
1

임원님

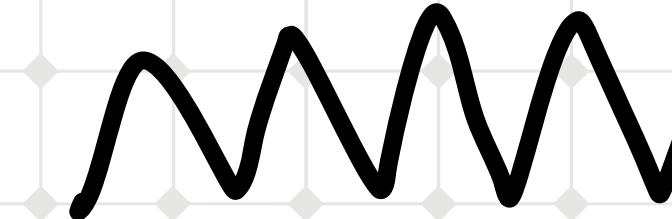
여행

소개





# OUR TEAM



김형진

회원가입,로그인  
깃관리,스웨거 작성  
요구사항 명세서

정종철

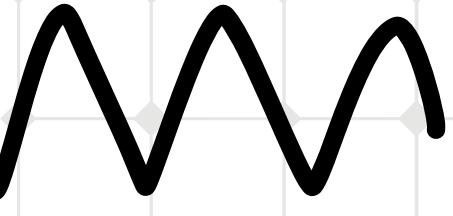
플로우차트,  
팔로우

정영우

메인  
기능 명세서

진영호

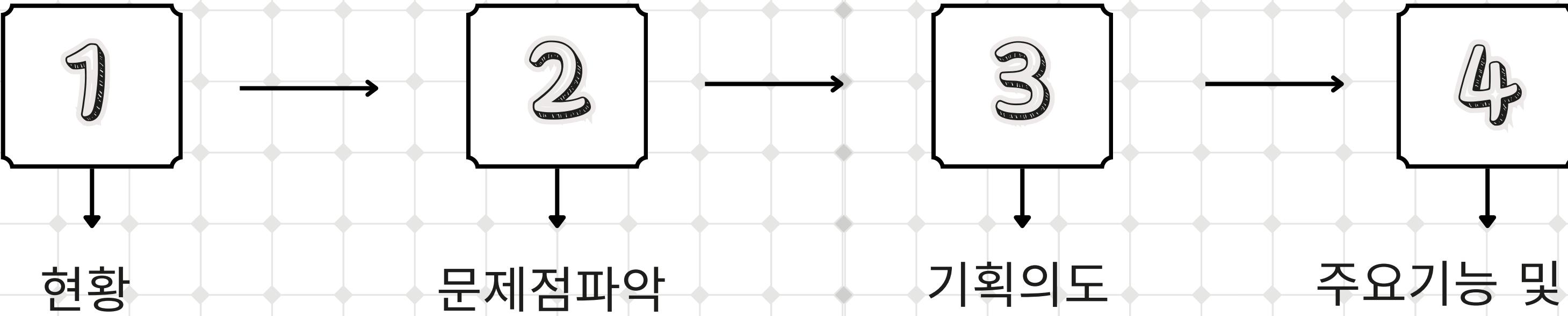
회원정보, 캘린더,  
DB설계



2

프로젝트 개요

# 프로젝트 개요



- 현재 운동 및 식단 기록, 칼로리 계산 앱들은 사용성과 편의성이 부족합니다.
- 복잡한 인터페이스와 번거로운 데이터 입력으로 이용하기 어려운 문제가 있습니다.

- 기존 앱들은 사용자들의 일상적인 요구를 충족시키지 못합니다.
- 복잡한 인터페이스와 번거로운 데이터 입력으로 사용자들의 이용이 어렵습니다.

- 직관적이고 간편한 사용자 경험을 제공하여 사용자들이 쉽게 운동 및 식단을 기록하고 칼로리를 계산할 수 있는 앱을 개발합니다.

- 간편한 기록: 직관적인 인터페이스를 통해 사용자가 운동과 식단을 쉽게 기록할 수 있습니다.
- 칼로리 계산: 입력한 식단 정보를 기반으로 자동으로 칼로리를 계산합니다.
- 타겟층: 건강에 관심 있는 모든 사용자를 대상으로 합니다. 특히, 바쁜 현대인들에게 적합합니다.
- 우리의 강점: 사용자 편의성에 초점을 맞춘 앱입니다.

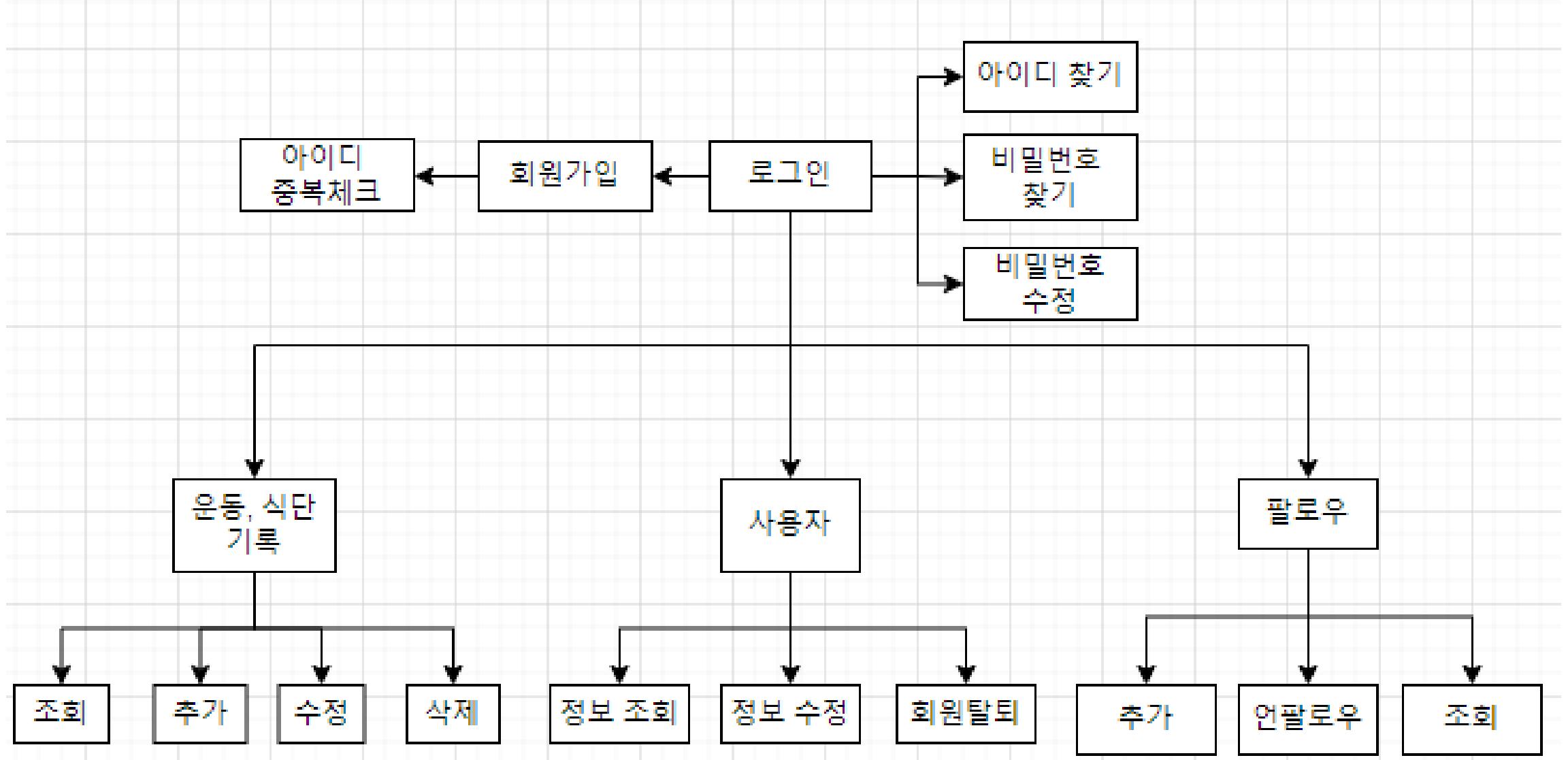
# 요구사항 명세서

요구사항ID	구분	기능ID	기능 상세 설명	필요데이터	비고
login	[POST] /login	getLogin	로그인 처리	아이디 / 비밀번호	비밀번호(영어,숫자,특수기호 조합) 최소 8자리
	[POST] /login/make	postMakeID	회원가입 정보 저장	아이디 / 비밀번호 / 이메일 / 이름 성별 / 나이 / 신장 / 몸무게	1.비밀번호(영어,숫자,특수기호 조합) 최소 8자리 2. 성별 - MALE,FEMALE로 값 넘겨주기
	[POST] /login/make/dupl	getduplID	아이디 중복 확인	아이디	
	[POST] /login/findid	getFindID	아이디 찾기	이름 / 이메일	
	[POST] /login/findpw	getFindPW	아이디 / 이메일 가입정보 확인	아이디 / 이메일	
	[PUT] /login/findpw	putFindPW	비밀번호 변경	새 비밀번호 / 새 비밀번호 확인	비밀번호(영어,숫자,특수기호 조합) 최소 8자리
main	[POST] /main/record	getRecord	사용자 7일 운동량 요약 확인	사용자 아이디	
			지난주, 이번주 운동량 비교		
			운동목표 확인		
	[POST] /main/record/insert	postRecord	운동목표 입력	사용자 아이디 / 운동이름 / 운동 시간	
	[PUT] /main/record	putRecord	운동목표 수정	사용자 아이디 운동이름,운동시간,소모 칼로리	
	[DELETE] /main/record	delRecord	운동목표 삭제	사용자 아이디/ 운동이름 or 음식이름	
	[POST] /main/eat	getDiet	섭취 음식 확인	사용자 아이디	
	[DELETE] /main/eat	delDiet	섭취음식 삭제	사용자아이디 / 섭취 음식이름	
	[POST] /main/eat/insert	postDiet	섭취 음식추가	사용자 아이디 섭취 음식이름, 섭취 칼로리	
calender	[GET] /calendar	Calendar	월별 총 운동량/섭취 칼로리 정보	사용자 아이디/월 정보	
			일별 운동량/섭취 칼로리 정보	사용자 아이디/일 정보	
follow	[GET] /follow/followers	getFollowers	팔로워 조회	사용자 아이디	
	[GET] /follow/followings	getFollowings	팔로잉 조회	사용자 아이디	
	[POST] /follow/following	postFollowing	팔로잉	사용자 아이디 / 상대 아이디	
	[DELETE] /follow/unfollowing	delUnfollowing	언팔로잉	사용자 아이디 / 상대 아이디	
user	[POST] /user/info	postInfo	회원 정보 조회	사용자 아이디	
	[POST] /user/update	postUpdate	회원 정보 수정	사용자 비밀번호 / 수정할 값	
	[PUT] /user/delete	putDelete	회원 탈퇴	사용자 아이디 / 비밀번호	

# 기능 명세서

페이지 명	주기능	상세기능	설명
로그인/회원가입	로그인	로그인	사용자계정으로 로그인 하는 기능 입니다.
	회원가입	회원가입	새로운 계정을 만드는 기능 입니다.
	아이디 찾기	아이디/비밀번호 찾기	사용자가 계정의 아이디나 비밀번호를 잊은 경우
	비밀번호 찾기	아이디/비밀번호 찾기	사용자가 계정의 아이디나 비밀번호를 잊은 경우
메인	운동 정보/ 섭취 칼로리 확인	운동명 운동량 소모 칼로리 섭취칼로리확인	일일 운동량과 음식 섭취정보를 보여줍니다.
	운동 정보/	운동명 운동량	사용자가 운동 정보와 섭취 칼로리 정보를
	운동 정보/	운동명 운동량 소모 칼로리	지정한 날짜의 운동 정보와 섭취 칼로리 정보를
	운동 정보/	사용자 운동정보	사용자가 운동 정보와 섭취 칼로리 정보의
	운동 정보/	지난주와 운동정보등을	지난주와 이번주를 비교하여 운동 량 소모 칼로리와
운동일지	월별 캘린더	월별 운동표시	캘린더에 월별 소모 칼로리와 섭취칼로리 표시합니다.
	일자표시	일자 운동표시	캘린더에 날짜를 선택시 그 일자에 일일 섭취 칼로리와
친구 및 소셜	친구 조회	친구 리스트 확인	사용자의 친구 목록을 보여주는 기능 입니다.
	친구 추가	친구추가 기능	사용자에게 새로운 친구를 추가 해주는 기능 입니다
	친구 삭제	친구 삭제 기능	사용자의 친구를 삭제 하는 기능 입니다.
	친구 정보 확인	운동정보 확인	친구의 운동정보 및 공개된 프로필을
사용자 계정 관리	회원정보 표시	회원정보 표시	사용자의 정보를 표시하는 기능 입니다.
	회원정보수정	회원정보수정	이름, 이메일, 비밀번호등의 개인 정보를
	회원정삭제	회원정삭제	사용자 계정을 삭제하는 기능 입니다.

# FLOWCHART

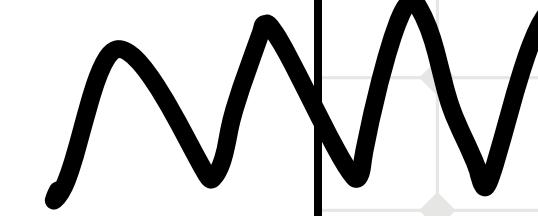
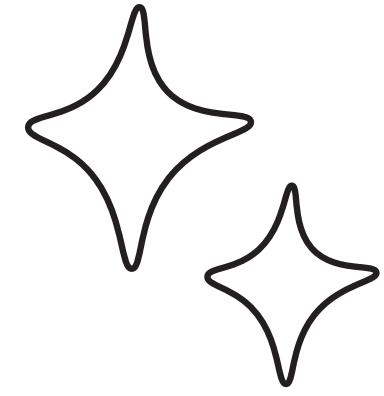
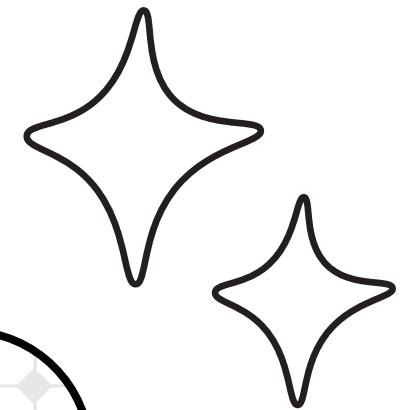
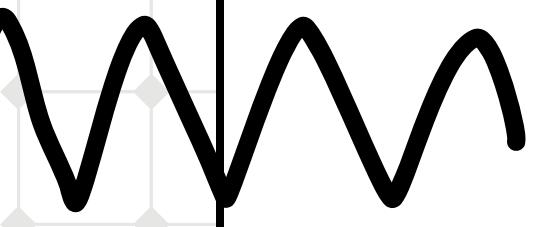


# 데이터베이스 설계



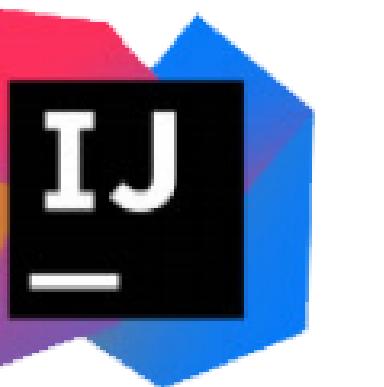
3

개발환경



기술스택

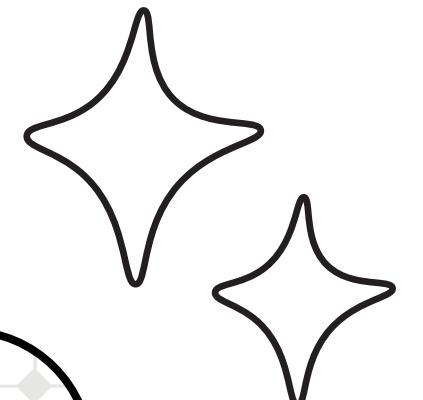
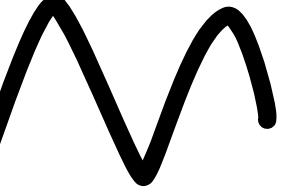
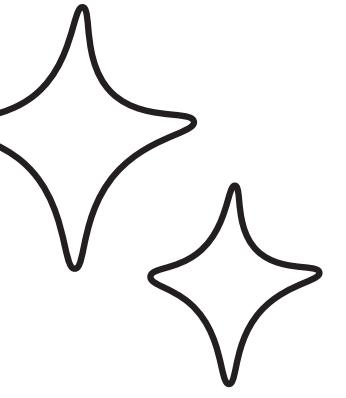
협업툴



leet

4

프로젝트 진행과정



# 프로젝트 개발 일정

일 월 화 수 목 금 토



S

코스피

코드부

김형진

로그인,회원가입

# 구현기능

로그인, 회원가입  
아이디/비밀번호 찾기  
CORS  
INTERCEPTOR

## LoginController

로그인, 회원가입, 아이디 중복체크,  
아이디/비밀번호 찾기 가능

PUT

/login/findpw  
비밀번호 수정

POST

/login/findpw  
비밀번호 찾기

POST

/login  
로그인

POST

/login/make  
회원가입

POST

/login/make/dupl  
아이디 중복체크

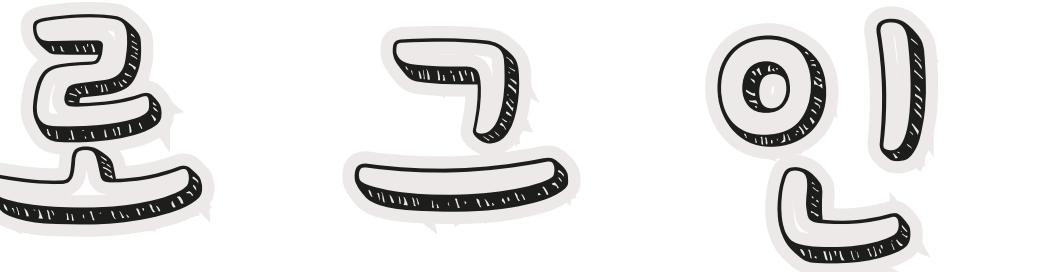
POST

/login/findid  
아이디 찾기

# CORS / INTERCEPTOR

```
@Configuration  
@RequiredArgsConstructor  
public class CorsConfig implements WebMvcConfigurer { ⚡ simple (0%)  
  
    private final LoginInterceptor loginInterceptor;  
  
    no usages 🔍 김형진  
    @Override  
    public void addCorsMappings(CorsRegistry registry) { ⚡ simple (0%)  
        registry.addMapping( pathPattern: "/**")  
            .allowedOrigins("*")  
            .allowedHeaders("*")  
            .allowedMethods("*");  
    }  
  
    no usages 🔍 김형진  
    @Override  
    public void addInterceptors(InterceptorRegistry registry) { ⚡ simple (0%)  
        registry.addInterceptor(loginInterceptor)  
            .addPathPatterns("/**")  
            .excludePathPatterns("/login/**");  
    }  
}
```

```
@Component  
@RequiredArgsConstructor  
public class LoginInterceptor implements HandlerInterceptor { ⚡ simple (18%)  
    private final UserRepository userRepository;  
  
    no usages 🔍 김형진  
    @Override  
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {  
  
        if (request.getRequestURI().contains("swagger")  
            || request.getRequestURI().contains("v3")) {  
            return true;  
        }  
  
        String authHeader = request.getHeader(s: "Authorization");  
        if (authHeader != null && authHeader.startsWith("Basic ")) {  
            String credentials = new String(Base64.getDecoder().decode(authHeader.substring(beginIndex: 6)));  
            String[] values = credentials.split(regex: ":", limit: 2);  
            String userid = values[0];  
            String password = values[1];  
  
            User user = userRepository.findByIdAndPassword(userid, password);  
  
            if (user != null) {  
                return true;  
            }  
        }  
        return false;  
    }  
}
```



```
public User findByUser(String id, String pw) { ⚡ simple (25%)  
    User user = loginRepository.findByIdAndPassword(id, pw);  
  
    if (user == null) {  
        throw new LoginException(ErrorCode.NOTFOUND);  
    }  
    if (user.getResign().equals(Resign.Y)) {  
        throw new LoginException(ErrorCode.INFONOTFOUND);  
    }  
  
    return user;  
}
```

아이디/패스워드를  
받아서 DB 검색을  
하고 검색된 값에  
따라 에러처리 또는  
유저 정보 리턴

# 회원가입

중복 관련 에러 처리, 비밀번호 정  
규식 및 비밀번호 체크, null 에러  
처리 후 사용자 정보 DB 저장

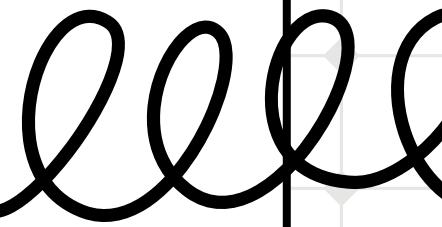
```
if (!dupeid.isEmpty()) {  
    throw new LoginException(ErrorCode.DUPLICATIONID);  
} else if (dupemail != null) {  
    throw new LoginException(ErrorCode.DUPLICATIONEMAIL);  
} else if (dupnickname != null) {  
    throw new LoginException(ErrorCode.DUPLICATIONNICKNAME);  
} else if (dumphonenum != null) {  
    throw new LoginException(ErrorCode.DUPLICATIONPHONENUM);  
} else if (user.getId() == null) {  
    throw new LoginException(ErrorCode.IDCHECK);  
} else if (user.getPassword() == null) {  
    throw new LoginException(ErrorCode.PASSWORDCHECK);  
} else if (user.getEmail() == null) {  
    throw new LoginException(ErrorCode.EMAILCHECK);  
} else if (user.getName() == null) {  
    throw new LoginException(ErrorCode.NAMECHECK);  
} else if (user.getGender() == null) {  
    throw new LoginException(ErrorCode.GENDERERROR);  
} else if (user.getPhonenumber() == null) {  
    throw new LoginException(ErrorCode.PHONENUMCHECK);  
} else if (user.getBirthdate() == null) {  
    throw new LoginException(ErrorCode.BIRTHDATECHECK);  
} else if (user.getPassword().length() < 8  
        || user.getPassword().length() > 15) {  
    throw new LoginException(ErrorCode.PASSWORDSIZE);  
} else if (!user.getPassword().equals(user.getPasswordch())) {  
    throw new LoginException(ErrorCode.PASSWORDDIFFERENT);  
}  
  
User dbuser = loginRepository.save(user);  
  
return dbuser;
```

# ID/PW 찾기

```
@Operation(summary = "아이디 찾기")
@PostMapping("/findid")
public ResponseEntity<String> getFindID(@RequestBody UserFindId userFindId){ ⚡ simple (0%)
    String id = loginService.findID(userFindId.getName(),userFindId.getEmail());
    return ResponseEntity.status(HttpStatus.OK).body(id);
}

no usages ✎ 김형진
@Operation(summary = "비밀번호 찾기")
@PostMapping("/findpw")
public ResponseEntity<String> getFindPW(@RequestBody UserFindPassword userFindPassword){ ⚡ simple (0%)
    String check = loginService.findPW(userFindPassword.getId(),userFindPassword.getEmail());
    return ResponseEntity.status(HttpStatus.OK).body(check);
}
```

ID/PW 찾을 때  
필요한 정보를 입  
력 받고 DB에 값  
이 있는지 없는지  
확인



# PW 수정

```
@Operation(summary = "비밀번호 수정")
@PutMapping("/findpw")
public ResponseEntity<String> putFindPW(@RequestBody @Valid PasswordCheck passwordCheck){ ⚡ simple(0)
    User dbuser = loginService.updatePW
        (passwordCheck.getId(),passwordCheck.getPassword(),passwordCheck.getPasswordCheck());
    return ResponseEntity.status(HttpStatus.ACCEPTED).body("변경된 비밀번호 = "+dbuser.getPassword());
}
```

사용자 정보 확인 뒤 비밀번호를 변경하는 부분

코 드리 브

장 야 우

매인

# 메인

운동/음식 기록  
조회 추가 수정 삭제 기능

## RecordController 운동기록 조회/추가/삭제 기능

PUT /main/record  
운동기록 수정

POST /main/record  
운동기록 조회

DELETE /main/record  
운동기록 삭제

POST /main/record/insert  
운동기록 추가

## DietController 음식기록 조회/추가/삭제 기능

PUT /main/eat  
음식기록 수정

POST /main/eat  
음식기록 조회

DELETE /main/eat  
음식기록 삭제

POST /main/eat/insert  
음식기록 추가

# 메인 - 소모, 섭취 칼로리 조회

일일 운동명과 운동시간  
그로 인해 소모된 칼로리를 계산하여  
출력합니다.  
그리고 이번주의 소모칼로리와 지난주의  
소모칼로리를 출력하여 줍니다.  
또한 같은 방법으로 일일 섭취 음식과 섭취  
칼로리와 지난주와 이번주 섭취 칼로리를  
보여 줍니다.

```
@Operation(summary = "운동기록 조회")
@PostMapping(@RequestMapping)
public ResponseEntity<String> Week(@RequestBody RecordDto recordDto) {
    String text = recordService.Week(recordDto.getId());

    return ResponseEntity.status(HttpStatus.OK).body(text);
}

public String Week(String id) { ⚡ simple (62%)
    StringBuilder daysBuilder = new StringBuilder();

    List<String> ename = recordRepository.name(id);
    List<Integer> daytime = recordRepository.time(id);
    List<Integer> daytotal = recordRepository.calories(id);
    // 이번주 소모 칼로리
    Integer week = recordRepository.findEMinById(id);

    LocalDate now = LocalDate.now();
    LocalDate endOfLastWeek = now.with(TemporalAdjusters.previous(DayOfWeek.SUNDAY));
    LocalDate startOfLastWeek = endOfLastWeek.with(TemporalAdjusters.previous(DayOfWeek.MONDAY));
    // 지난주 운동 정보
    Integer last = recordRepository.findCalculatedEMinByLastWeekAndId(startOfLastWeek.atStartOfDay(), endOfLastWeek.atStartOfDay(), id);
    // 지난주 운동기록이 없을 경우
    if (last == null) {
        last = 0;
    }
    // 이번주 운동기록이 없을 경우
    if (week == null) {
        week = 0;
    }
}
```

# 메인 - 소모, 섭취 칼로리 추가

오늘의 운동및 섭취 음식을 등록하는 기능으로 만약 등록시 이미 있는 정보의 경우 새로 추가로 등록하는 것이 아닌 원래 있던 정보에 합쳐서 저장합니다.

```
@Operation(summary = "운동기록 추가")
@PostMapping(@RequestMapping("/insert"))
public ResponseEntity<Record> insertwork(@RequestBody RecordDto recordDto) {
    recordDto.setRdatetime(LocalDateTime.now());
    ModelMapper mapper = new ModelMapper();
    Record record = mapper.map(recordDto, Record.class);
    Record dbrecord = recordService.regist(record);
    return ResponseEntity.status(HttpStatus.OK).body(dbrecord);
}

@Transactional
public Record regist(Record record) { ⚡ simple (25%)

    Record dbrecord = recordRepository.findByIdAndEname(record.getId(), record.getEname());

    if (dbrecord == null) {
        return recordRepository.save(record);
    } else {
        recordRepository.updateRecord(record.getId(), record.getEname(), record.getEmin());
        return dbrecord;
    }
}

@Query("SELECT emin FROM Record WHERE id = :id AND ename = :ename AND DATE_FORMAT(rdatetime, '%Y-%m-%d') = :rdatetime")
Optional<Record> selectenameday(@Param("id") String id, @Param("ename") String ename, @Param("rdatetime") String date);

//이 날짜에 운동했는지 확인
@Query("SELECT emin FROM Record WHERE id = :id AND DATE_FORMAT(rdatetime, '%Y-%m-%d') = :rdatetime")
List<Integer> selectday(@Param("id") String id, @Param("rdatetime") String rdatetime); ⚡ simple (0%)

//이미 있는 운동의 경우 그 운동에 시간 추가
@Modifying
@Query("UPDATE Record SET emin = emin + :emin WHERE id = :id AND ename = :ename AND DATE(rdatetime) = CURDATE()")
void updateRecord(@Param("id") String id, @Param("ename") String ename, @Param("emin") int emin); ⚡ simple (0%)
```

# 메인 - 소모, 섭취 칼로리 삭제

사용자가 날짜를 입력하면 해당 날짜의 정보를 삭제하는 기능으로 운동명이나 음식명을 입력하면 해당날짜에 운동명이나 음식명만 적지 않을 경우 해당 날짜의 운동 또는 음식정보를 모두 삭제 합니다.

```
@Operation(summary = "운동기록 삭제")
@DeleteMapping("{{id}}")
public String Deletework(@RequestBody RecordDto recordDto) {
    String text = recordService.delete(recordDto);
    return text;
}

@Transactional
public String delete(RecordDto recordDTO) { ⚡ mildly complex (100%)

    if (recordDTO.getEname().equals("")) {
        List<Integer> list = recordRepository.selectday(recordDTO.getId(), recordDTO.getDate());
        if (list.size() != 0) {
            recordRepository.deleteByIdAndRdatetime(recordDTO.getId(), recordDTO.getDate());
            return recordDTO.getDate() + " 의 기록을 삭제 했습니다.";
        } else {
            return "작성하신 기록은 존재하지 않습니다.";
        }
    } else {
        Optional<Record> list = recordRepository.selectenameday(recordDTO.getId(), recordDTO.getEname(), recordDTO.getDate());
        if (list.isPresent()) {
            recordRepository.deleteByIdAndEnameAndRdatetime(recordDTO.getId(), recordDTO.getEname(), recordDTO.getDate());
            return recordDTO.getDate() + " " + recordDTO.getEname() + " 의 기록을 삭제 했습니다.";
        } else {
            return "작성하신 기록은 존재하지 않습니다.";
        }
    }
    // 해당 날짜의 운동 정보 삭제
    1 usage  ↳ 김형진
    @Transactional
    @Modifying
    @Query("DELETE FROM Record WHERE id = :id AND ename = :ename AND DATE_FORMAT(rdatetime, '%Y-%m-%d') = :rdatetime")
    void deleteByIdAndEnameAndRdatetime(@Param("id") String id, @Param("ename") String ename, @Param("rdatetime") String rdatetime);

    // 해당 날짜의 모든 정보 삭제
    1 usage  ↳ 김형진
    @Transactional
    @Modifying
    @Query("DELETE FROM Record WHERE id = :id AND DATE_FORMAT(rdatetime, '%Y-%m-%d') = :rdatetime")
    void deleteByIdAndRdatetime(@Param("id") String id, @Param("rdatetime") String rdatetime); ⚡ simple (0%)
```

# 메인 - 소모, 섭취 칼로리 수정

운동 및 음식정보를 수정하는 기능입니다.  
만약 사용자가 수정 시간을 공란으로 작성  
한 경우 0으로 수정 명을 공란일 경우 null  
로 설정합니다.

```
@Operation(summary = "운동기록 수정")
@PutMapping(@Path)
public String updateWork(@RequestBody RecordDto recordDto) {
    // rename이 빈 문자열인 경우 null로 설정
    if ("".equals(recordDto.getRename())) {
        recordDto.setRename(null);
    }
    // retime이 null인 경우 0으로 설정
    if ("".equals(recordDto.getRetime())) {
        recordDto.setRetime(0);
    }

    ModelMapper mapper = new ModelMapper();
    Record record = mapper.map(recordDto, Record.class);

    String text = recordService.update(record);
    return text;
}
```

# 메인 - 소모, 섭취 칼로리 수정

수정하려는 운동명 혹은 음식명이 저장되어 있는지 판별 합니다.

판별후 수정 명, 수정시간 수정 칼로리등이 작성 되어 있는지 확인 후 수정을 합니다.

만약 수정시 이미 있는 운동명 혹은 음식명 이면 그 운동이 가지고 있던 운동시간 음식의 경우 가지고 있던 칼로리를 합쳐주는 가능 입니다.

```
@Transactional
public String update(Record record) { ⚠ very complex (200%)
    Record dbrecord = recordRepository.findByIdAndEname(record.getId(), record.getEname());
    // 운동 기록이 존재하는 경우
    if (dbrecord != null) {
        // 새로운 운동 이름과 시간이 모두 있는 경우
        if (record.getRename() != null && record.getRetime() != 0) {
            // 새로운 운동 이름이 이미 존재하는 경우
            Optional<Record> testname = recordRepository.renameTest(record.getId(), record.getRename());
            if (testname.isPresent()) {
                // 이미 있는 운동이면 합치기
                recordRepository.updateRenameretime(record.getId(), record.getRename(), record.getRetime());
                recordRepository.deleteoverlap(record.getId(), record.getEname());
                return "해당 운동명과 시간이 합쳐졌습니다.";
            } else {
                // 이미 없는 경우 새로운 운동 이름과 시간으로 업데이트
                recordRepository.updateAll(record.getId(), record.getEname(), record.getRename(), record.getRetime());
                return "해당 운동명과 시간이 변경 되었습니다.";
            }
        }
        // 새로운 운동 이름만 있는 경우
        else if (record.getRename() != null) {
            // 새로운 운동 이름이 이미 존재하는 경우
            Optional<Record> testname = recordRepository.renameTest(record.getId(), record.getRename());
            if (testname.isPresent()) {
                // 이미 있는 운동이면 시간만 업데이트
                int sumemin = recordRepository.emin(record.getId(), record.getEname());
                recordRepository.updateExistingEnameWithTime(record.getId(), record.getRename(), sumemin);
                recordRepository.deleteoverlap(record.getId(), record.getEname());
                return "해당 운동의 이름이 합쳐졌습니다.";
            } else {
                // 없는 경우 새로운 운동 이름으로 업데이트
                recordRepository.updateExistingEnameWithTime(record.getId(), record.getEname(), record.getRename());
                return "해당 운동의 이름이 변경되었습니다.";
            }
        }
        // 새로운 운동 시간만 있는 경우
        else if (record.getRetime() != 0) {
            // 운동 시간만 업데이트
            recordRepository.updatetime(record.getId(), record.getEname(), record.getRetime());
            return "해당 운동의 시간이 변경되었습니다.";
        }
    }
}
```

# 메인 - 소모, 섭취 칼로리 수정

수정하려는 운동명 혹은 음식명이 저장되어 있는지 판별 합니다.

판별후 수정 명, 수정시간 수정 칼로리등이 작성 되어 있는지 확인 후 수정을 합니다.

만약 수정시 이미 있는 운동명 혹은 음식명 이면 그 운동이 가지고 있던 운동시간 음식의 경우 가지고 있던 칼로리를 합쳐주는 가능 입니다.

```
//변경시 있는 운동명인지 확인
2 usages  ▲ DESKTOP-33SQ8Hl\n
@Query("SELECT ename FROM Record WHERE id = :id AND ename = :rename AND DATE(rdatetime) = CURDATE()")
Optional<Record> renametest(@Param("id") String id, @Param("rename") String rename); ⚪ simple (0%)\n\n
//운동 이름 시간 바꾸기
1 usage  ▲ DESKTOP-33SQ8Hl\n
@Transactional
@Modifying
@Query("UPDATE Record SET ename = :rename, emin = :retime WHERE id = :id AND ename = :ename AND DATE(rdatetime) = CURDATE()")
void updateAll(@Param("id") String id, @Param("ename") String ename, @Param("rename") String rename, @Param("retime") int retime);\n\n
//운동 시간바꾸기
1 usage  ▲ DESKTOP-33SQ8Hl\n
@Transactional
@Modifying
@Query("UPDATE Record SET emin = :retime WHERE id = :id AND ename = :ename AND DATE(rdatetime) = CURDATE()")
void updatetime(@Param("id") String id, @Param("ename") String ename, @Param("retime") int retime); ⚪ simple (0%)\n\n
//이미 있는 운동명으로 변경시 운동합치기
//운동명 변경시 운동합치기
1 usage  ▲ DESKTOP-33SQ8Hl\n
@Transactional
@Modifying
@Query("UPDATE Record SET emin = emin + :summin WHERE id = :id AND ename = :rename AND DATE(rdatetime) = CURDATE()")
void updateExistingNameWithTime(@Param("id") String id, @Param("rename") String rename, @Param("summin") int summin); ⚪ simple (0%)\n\n
//운동명 바꿀면서 합쳐질때 운동시간을 미리 저장하는 쿼리문
1 usage  ▲ DESKTOP-33SQ8Hl\n
@Query("SELECT emin FROM Record WHERE id = :id AND ename = :ename AND DATE(rdatetime) = CURDATE()")
int emin(@Param("id") String id, @Param("ename") String ename); ⚪ simple (0%)\n\n
//운동명 바꾸기
1 usage  ▲ DESKTOP-33SQ8Hl\n
@Transactional
@Modifying
@Query("UPDATE Record SET ename = :rename WHERE id = :id AND ename = :ename AND DATE(rdatetime) = CURDATE()")
void updateExistingNameWithTime(@Param("id") String id, @Param("ename") String ename, @Param("rename") String rename);\n\n
//이미 있는 운동명으로 변경 변경된 운동시간
1 usage  ▲ DESKTOP-33SQ8Hl\n
@Transactional
@Modifying
@Query("UPDATE Record SET emin = emin + :retime WHERE id = :id AND ename = :rename AND DATE(rdatetime) = CURDATE()")
void updaterenameretime(@Param("id") String id, @Param("rename") String rename, @Param("retime") int retime); ⚪ simple (0%)\n\n
//운동명 변경시 변경전 운동기록 삭제
2 usages  ▲ DESKTOP-33SQ8Hl\n
@Transactional
@Modifying
@Query("DELETE From Record WHERE id = :id AND ename = :ename AND DATE(rdatetime) = CURDATE()")
void deleteoverlap(@Param("id") String id, @Param("ename") String ename); ⚪ simple (0%)\n\n
//유효성 검사
2 usages  ▲ DESKTOP-33SQ8Hl\n
@Query("SELECT r FROM Record r WHERE r.id = :id AND r.rename = :ename AND DATE(r.rdatetime) = CURDATE()")
Record findByIdAndEname(@Param("id") String id, @Param("ename") String ename); ⚪ simple (0%)
```

코 드 러 브

진 앙 허

회원정보(조회, 변경, 탈퇴), 캘린더

# 구현기능

회원정보 조회

회원정보 수정

회원탈퇴

캘린더에 해당날짜의  
데이터 출력

한달치 칼로리 비교 후 출력

## CalenderController

하루,한달 운동/음식 칼로리 조회 가능

POST

/calender/month  
한달 운동,음식 칼로리 조회

POST

/calender/day  
하루 운동,음식 칼로리 조회

## UserContoroller

사용자 정보 확인,수정,탈퇴  
기능

PUT

/user/update  
사용자 정보 수정

PUT

/user/delete  
회원탈퇴

POST

/user/info  
사용자 정보 확인

# 회원정보변경

로그인한 회원의  
개인정보를  
변경

```
no usages
@Operation(summary = "사용자 정보 수정")
@PutMapping("/update")
public ResponseEntity<User> updateUser(@RequestBody @Valid UserDto userDto) {
    ModelMapper mapper = new ModelMapper();
    User user = mapper.map(userDto, User.class);
    User dbUser = userService.update(user);
    return ResponseEntity.status(HttpStatus.ACCEPTED).body(dbUser);
}
```

```
public User update(User user) {
    User updateuser = userRepository.findById(user.getId())
        .orElseThrow(() -> new LoginException(ErrorCode.INFONOTFOUND));

    if (user.getEmail() != null)
        updateuser.setEmail(user.getEmail());
    if (user.getPassword() != null)
        updateuser.setPassword(user.getPassword());
    if (user.getNickname() != null)
        updateuser.setNickname(user.getNickname());
    if (user.getName() != null)
        updateuser.setName(user.getName());
    if (user.getBirthdate() != null)
        updateuser.setBirthdate(user.getBirthdate());
    if (user.getPhonenumber() != null)
        updateuser.setPhonenumber(user.getPhonenumber());
    if (user.getGender() != null)
        updateuser.setGender(user.getGender());
    if (user.getHeight() != 0)
        updateuser.setHeight(user.getHeight());
    if (user.getWeight() != 0)
        updateuser.setWeight(user.getWeight());

    User dbuser = userRepository.save(updateuser);
    return dbuser;
}
```

# 회원탈퇴

## 회원탈퇴 진행

DB에서 삭제하지 않고 탈퇴  
여부를 수정하고 데이터를  
일정기간 보관

```
no usages
@Operation(summary = "회원탈퇴")
@PutMapping("/delete")
public ResponseEntity<String> resignuser(@RequestBody @Valid LoginInfo loginInfo) {
    String result = userService.resignuser(loginInfo.getId(), loginInfo.getPassword());
    return ResponseEntity.status(HttpStatus.ACCEPTED).body(result);
}

public String resignuser(String id, String password) {
    User user = userRepository.findByIdAndPassword(id,password);

    if (user == null){
        throw new LoginException(ErrorCode.INFONOTFOUND);
    }
    if(user.getResign().equals(Resign.Y)){
        throw new LoginException(ErrorCode.INFONOTFOUND);
    }

    user.setResign(Resign.Y);
    userRepository.save(user);

    return "☞ •••? -☆° 탈퇴 완료되었습니다. ☞ •••? -☆°";
}
```

# 캘린더 DAY

캘린더에서

해당 날짜를 누르면

해당 날짜의 데이터를 출력

19 20 21 22 23

```
@Operation(summary = "하루 운동, 음식 칼로리 조회")
@PostMapping("/day")
public ResponseEntity<String> calenderday(@RequestBody CalenderUserData calenderUserData) {
    String result = calenderService.calday(calenderUserData.getId(), calenderUserData.getDatetime());
    return ResponseEntity.status(HttpStatus.OK).body(result);
}
```

```
double dietCalories = dietCaloriesOpt.map(Double::parseDouble).orElse(0.0);
double exerciseCalories = exerciseCaloriesOpt.map(Double::parseDouble).orElse(0.0);

String dayDietCalories = String.format("%.2f", dietCalories);
String dayExerciseCalories = String.format("%.2f", exerciseCalories);

return datetime + "일의 섭취한 칼로리는 " + dayDietCalories +
        "kcal이고 운동으로 소모한 칼로리는 " + dayExerciseCalories + "kcal입니다.";
```

```
@Query(value = "SELECT d.dcalories " +
    "FROM diet d " +
    "WHERE d.id = :id AND d.ddatetime = :datetime",
    nativeQuery = true)
Optional<String> findDietCalories(@Param("id") String id, @Param("datetime") String datetime);

1 usage
@Query(value = "SELECT COALESCE(SUM(e.ecalories * r.emin), 0) " +
    "FROM record r " +
    "LEFT JOIN exercise e ON r.ename = e.ename " +
    "WHERE r.id = :id AND r.rdatetime = :datetime",
    nativeQuery = true)
Optional<String> findExerciseCalories(@Param("id") String id, @Param("datetime") String datetime);
```

# 캘린더 MONTH

캘린더에서  
한 달 동안의

운동기록과 음식기록을  
계산하여 칼로리 출력

```
public String calmonth(String id) {
    Optional<Double> dietcal = calenderRepository.dietmonth(id);
    Optional<Double> exercal = calenderRepository.exermonth(id);

    double dietmonth = dietcal.orElse( other: 0.0 );
    double exercalmonth = exercal.orElse( other: 0.0 );

    double summonth = (dietmonth - exercalmonth);
    double reversesummonth = (exercalmonth - dietmonth);

    String f2DietMonth = String.format("%.2f", dietmonth);
    String f2ExerMonth = String.format("%.2f", exercalmonth);
    String f2SumMonth = String.format("%.2f", summonth);
    String f2ReverseSumMonth = String.format("%.2f", reversesummonth);

    if (dietmonth > exercalmonth){
        return "한달동안 섭취한 음식 칼로리는 " + f2DietMonth + "kcal이며, " + "운동으로 소비한 칼로리는 " +
    }
    else {
        return "한달동안 섭취한 음식 칼로리는 " + f2DietMonth + "kcal이며, " + "운동으로 소비한 칼로리는 " +
    }

@Query(value = "SELECT SUM(dcalories) FROM diet WHERE id = :id " +
    "AND ddatetime >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)",nativeQuery = true)
Optional<Double> dietmonth(@Param("id") String id);

1 usage
@Query(value = "SELECT COALESCE(SUM(e.ecalories * r.emin), 0) FROM record r " +
    "LEFT JOIN exercise e ON r.ename = e.ename WHERE r.id = :id" +
    "| AND rdatetime >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)",nativeQuery = true)
Optional<Double> exermonth(@Param("id") String id);
```

# 한달치 데이터 출력 시연영상

HTTP restaoi01 / 캘린더 한달치 칼로리 계산값 보내주기

Save

POST http://localhost:8080/calender/month

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL JSON

```
1 {
2   ... "id": "aaa"
3 }
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize Text

1 한달동안 섭취한 음식 칼로리는 64000.00kcal이며, 운동으로 소비한 칼로리는 7894.10kcal입니다. 음식 섭취 칼로리에서 운동으로 소비한 칼로리를 빼면 56145.90kcal입니다.

5

정조찰

Follow

# 구현기능

팔로우 추가  
팔로잉 리스트 조회  
팔로이 리스트 조회  
팔로우 삭제

## FollowController 팔로우, 팔로워 기능

POST	/follow/{followerNickname}/following/{followeeNickname}	팔로우 추가
GET	/follow/{nickname}/followings	팔로잉 조회
GET	/follow/{nickname}/followers	팔로워 조회
DELETE	/follow/{followeeNickname}/unfollowing/{followerNickname}	팔로우 삭제

# ENTITY, REPOSITORY

```
public interface UserRepository extends JpaRepository<User, String> {
    10개 사용 위치 신규*
    Optional<User> findByNickname(String nickname);
```

```
public class User {

    @Column(unique = true, nullable = false)
    @Schema(title = "닉네임")
    private String nickname;
```

<code>id</code>		<code>name</code>	<code>nickname</code>	
-----------------	--	-------------------	-----------------------	--

```
public interface FollowRepository extends JpaRepository<Follow, Long> {
    2개 사용 위치 ± jungjc1117
    Optional<Follow> findByFollowerAndFollowee(User follower, User followee);
    1개 사용 위치 ± jungjc1117
    List<Follow> findByFollower(User Follower);
    1개 사용 위치 ± jungjc1117
    List<Follow> findByFollowee(User Followee);
}
```

```
@Getter
@Entity
@Table(name = "follow", uniqueConstraints = {
    @UniqueConstraint(columnNames = {"follower_id", "followee_id"})
})
@NoArgsConstructor
public class Follow {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "follower_id")
    private User follower;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "followee_id")
    private User followee;

    1개 사용 위치 ± jungjc1117
    public Follow(User follower, User followee) {
        this.follower = follower;
        this.followee = followee;
    }
}
```

<code>id</code>		<code>followee_id</code>		<code>follower_id</code>	
-----------------	--	--------------------------	--	--------------------------	--

# FOLLOWING

//1 사용자가 다른 사용자를 팔로우하고, 이를 데이터베이스에 저장  
1개 사용 위치 : jungjc1117

```
@Transactional  
public void follow(String followerNickname, String followeeNickname) {  
    User follower = userRepository.findByNickname(followerNickname)  
        .orElseThrow(() -> new RuntimeException("Error : Follower is not found"));  
    User followee = userRepository.findByNickname(followeeNickname)  
        .orElseThrow(() -> new RuntimeException("Error : Followee is not found"));  
    validationCheck(follower, followee);  
    Follow follow = new Follow(follower, followee);  
    followRepository.save(follow);  
}
```

```
@PostMapping("/{followerNickname}/following/{followeeNickname}")  
public ResponseEntity<String> follow(@PathVariable String followerNickname,  
                                      @PathVariable String followeeNickname) {  
    User follower = userRepository.findByNickname(followerNickname)  
        .orElseThrow(() -> new RuntimeException("Error: Follower not found"));  
    User followee = userRepository.findByNickname(followeeNickname)  
        .orElseThrow(() -> new RuntimeException("Error: Followee not found"));  
    followService.follow(followerNickname, followeeNickname);  
    String message = followerNickname + "가 " + followeeNickname + "의 친구가 되었습니다."  
  
    return ResponseEntity.ok(message);  
}
```

## FollowService.java

```
private void validationCheck(User user, User followee) {  
    if (user == followee) {  
        throw new RuntimeException("Error : Cannot follow");  
    }  
  
    followRepository.findByFollowerAndFollowee(user, followee)  
        .ifPresent(u -> {  
            throw new IllegalStateException("이미 팔로우 되었습니다.");  
        });  
}
```

POST http://localhost:8089/follow/길동1/following/길동2 Send

Params Auth Headers (8) Body ● Pre-req. Tests ● Settings Cookies

Body

Pretty Raw Preview Visualize Text

1 길동1가 길동2의 친구가 되었습니다.

#	id	followee_id	follower_id
1	2	bbb	aaa

eee

# FOLLOWING 리스트 조회

// 3 특정 사용자가 팔로잉하는 다른 사용자 목록을 가져옵니다.

```
1개 사용 위치  ↳ jungjc1117 *
@Transactional
public List<UserDto> getFollowings(String nickname) {
    User user = userRepository.findByNickname(nickname)
        .orElseThrow(() -> new RuntimeException("Error : User is not found"));
    List<Follow> followings = followRepository.findByFollower(user);
    return followings.stream() Stream<Follow>
        .map(follow -> mapToDto(follow.getFollowee())) Stream<UserDto>
        .collect(Collectors.toList());
}
```

```
@GetMapping("/{nickname}/followings")
public ResponseEntity<List<UserDto>> getFollowings(@PathVariable String nickname) {
    Optional<User> optionalUser = userRepository.findByNickname(nickname);
    if (optionalUser.isPresent()) {
        User user = optionalUser.get();
        return new ResponseEntity<>(followService.getFollowings(nickname), HttpStatus.OK);
    } else {
        return new ResponseEntity<>(HttpStatus.NOT_FOUND);
    }
}
```

## FollowService.java

```
private UserDto mapToDto(User user) {
    UserDto userDto = new UserDto();
    userDto.setId(user.getId());
    userDto.setNickname(user.getNickname());
    userDto.setName(user.getName());

    return userDto;
}
```

The screenshot shows a Postman request configuration. The method is set to GET, the URL is `http://localhost:8089/follow/길동1/followings`, and the status is 200 OK with a response time of 22 ms and a body size of 461 B. The response body is displayed in JSON format:

```
[{"id": "bbb", "password": null, "passwordch": null, "email": null, "nickname": "길동2", "name": "일길동", "gender": null, "birthdate": null, "phonenumber": null, "user_date": null, "resign": null, "height": 0.0, "weight": 0.0}]
```

# UNFOLLOWING

```
// 2 사용자가 다른 사용자를 언팔로우  
1개 사용 위치 ▲ jungjc1117  
  
@Transactional  
  
public void unfollow(String followerNickname, String followeeNickname) {  
    User follower = userRepository.findByNickname(followerNickname)  
        .orElseThrow(() -> new RuntimeException("Error : Follower is not found"));  
    User followee = userRepository.findByNickname(followeeNickname)  
        .orElseThrow(() -> new RuntimeException("Error : Followee is not found"));  
    Follow follow = followRepository.findByFollowerAndFollowee(follower, followee)  
        .orElseThrow(() -> new RuntimeException("Error : Follow is not found"));  
    followRepository.delete(follow);  
}
```

```
@DeleteMapping("/{followeeNickname}/unfollowing/{followerNickname}")  
public ResponseEntity<String> unfollow(@PathVariable String followerNickname,  
                                         @PathVariable String followeeNickname) {  
    followService.unfollow(followerNickname, followeeNickname);  
    String message = followeeNickname + "가 " + followerNickname + "와 친구관계를 삭제하였습니다.";  
    return ResponseEntity.ok(message);  
}
```

DELETE http://localhost:8089/follow/길동2/unfollowing/길동1 Send

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Body 200 OK 39 ms 313 B Save as example ...

Pretty Raw Preview Visualize Text

1 길동2가 길동1과 친구관계를 삭제하였습니다.

#	id	followee_id	follower_id
1			

6

자체평가 및

느낀점

# 자체 평가 및 느낀점

김형진

프로젝트 경험에 적어서 소통이 부족하긴 했지만 일정에 맞춰 프로젝트를 끝내서 다행이라고 생각합니다. 다음 프로젝트에서는 소통을 더 많이 해야 할 것 같습니다.

정승철

외부 환경의 변동이 많은데, 우여곡절이 많았던 기간이었다. 그러나 그것보다 나 자신의 부족함을 확인할 수 있었다. 다음에는 더 나은 결과물을 만들기 위해 노력해야겠다.

# 자체 평가 및 느낀점

정영우

소통의 중요성에 대해 알게되는 시간이였습니다.

## 진영호

1. 막연하게 설계했던 DB에 대해서 실제적으로 타켓을 정하고 시장에 출시할 목적으로 설계하여 DB의 다양한 활용방법을 생각하고 설계하게 되었습니다.
2. 기획으로 대로 프로젝트가 완성되어 기쁘지만 미처 추가하지 못한 콘텐츠와 예외처리 부분에 대해 아쉬움이 남아 다음 프로젝트 때 완성하겠습니다.
3. 작성한 코드를 팀원들과 논의하고 일정에 맞춰 프로젝트를 진행하는 경험은 좋았으나 프론트와 백엔드의 협업에 아쉬운 부분이 있었고, 지금보다 더욱 자주 만나고 연락하고 얘기를 들어주어야겠다고 생각했습니다.

7

# Q & A



THANKS,

"YOU"

