고급웹 프로그래밍

기말 프로젝트 보고서

다이어트 기록 및 도우미 사이트

이름 : 이성은

학번 : 60212770

1. 서비스 개요

이 프로젝트는 "다이어트 기록 및 도우미 사이트"로, 로그인한 사용자가 자신의 기본 신체 정보(나이, 성별, 체중, 신장)와 운동 기록, 식단 기록을 관리하고, 이 정보를 이용하여 하루 칼로리 소모 및 섭취 현황을 계산해 주는 웹 서비스입니다.

주요 기능은 다음과 같습니다.

- 로컬/소셜(카카오, 네이버) 로그인 및 회원가입
- 기본 신체 정보(나이, 성별, 체중, 신장) 입력/수정
- 운동 기록 등록/수정/삭제, 운동별 METs 기반 칼로리 계산
- 한 번의 운동 세션에 여러 운동을 N:M 구조로 기록
- 식단(식사 종류, 칼로리, 날짜, 음식 사진) 등록/수정/삭제
- 하루 기준 섭취 칼로리, 운동 소모 칼로리, 순 소모 칼로리 계산 및 저장- 운동 기록별 댓글 등록/수정/삭제

2. 전체 시스템 구조

2-1. 클라이언트 (EJS 템플릿)

화면은 EJS 템플릿으로 구성되며, 각 템플릿에서 폼 전송(method="post")과 JavaScript fetch()를 함께 사용하여 Express 서버와 통신합니다.

- login.ejs

  로컬 로그인 폼 (아이디, 비밀번호)

  카카오/네이버 소셜 로그인 링크

- register.ejs

  회원가입 폼 (아이디, 비밀번호, 나이, 성별, 체중, 신장)

- profile.ejs

  로그인 사용자의 기본 신체 정보를 수정하는 화면

  소셜 로그인 최초 이용 시 필수 입력 페이지

- health.ejs

운동 선택 버튼, 운동 시간 선택, 세부사항 입력
운동 기록 목록, 각 기록별 댓글 목록 및 댓글 입력 폼 상단 "오늘의 칼로리 현황"
영역에서 AJAX 로 /health/calorie 호출 후 결과 표시

- food.ejs
식단 등록 폼 (식사 종류, 칼로리, 날짜, 이미지 여러 장 업로드)
"오늘 섭취 칼로리" 표시 및 식단 기록 목록 및 수정/삭제 버튼
(fetch()로 PUT/DELETE 요청)

2-2. 서버 (Express + Passport + Sequelize)

app.js 에서 Express 애플리케이션을 생성하고 다음과 같이 설정합니다.
- 뷰 엔진: EJS
- 정적 파일 디렉터리: public/
- 미들웨어: morgan, express.json, express.urlencoded, cookie-parser, express-session- 인증: passport.initialize(), passport.session()
- 라우터 연결:

/ → routes/auth.js (로그인, 회원가입, 소셜 로그인, 프로필, 환영 페이지)
/health → routes/health.js (운동 기록 + 댓글 + 칼로리 계산) /food
→ routes/food.js (식단 기록 + 이미지 업로드)

passportConfig(passport)를 통해 로컬, 카카오, 네이버 전략을 등록하고, 인증에 성공하면 req.user 정보를 기존 세션 기반 코드와 호환되도록 req.session.userId, req.session.user 등에 동기화합니다.

데이터베이스는 Sequelize ORM 을 사용하여 MySQL 과 연결하며, 서버 시작 시 Exercise 테이블에 기본 운동(METs 값 포함)을 초기 데이터로 삽입합니다.

3. 코드 구성 (폴더별 파일 및 역할)

3-1. 프로젝트 루트

- app.js
Express 서버 초기화 및 미들웨어 설정
passport 설정 및 세션 연동

라우터(auth, health, food) 연결

Sequelize sync 및 Exercise 초기 데이터 삽입

3-2. routes/

- routes/auth.js

POST /register : 로컬 회원가입 처리 (아이디 중복 체크 후 User 생성)

POST /admit : passport local 전략으로 로그인, 성공 시 /welcome 리다이렉트

GET /auth/kakao, /auth/naver : 소셜 로그인 시작

GET /auth/kakao/callback, /auth/naver/callback : 소셜 로그인 콜백 처리

GET /welcome : 로그인 후 환영 페이지

GET/POST /user/profile : 회원 기본 정보(나이, 성별, 체중, 신장) 조회 및 수정

GET /logout : 로그아웃 및 세션 해제

- routes/health.js

GET /health : 운동 기록 목록 및 댓글 목록 조회, 운동 선택 버튼에 필요한 Exercise 목록 조회

POST /health : 운동 기록 등록, Health + Health_Exercise 생성 및 칼로리 계산

PUT /health/:id : 운동 기록 수정 (여러 운동/시간 수정 반영)

DELETE /health/:id : 운동 기록 삭제 (연관 Health_Exercise, 댓글도 함께 삭제)

POST /health/:id/comments : 댓글 등록

PUT /health/:healthId/comments/:commentId : 댓글 수정

DELETE /health/:healthId/comments/:commentId : 댓글 삭제

GET /health/calorie : BMR, 운동 소모 칼로리, 섭취 칼로리를 종합해 오늘 칼로리 현황 JSON 응답

recalcTodayCalorieForUser(userId) : 하루 기준 순 소모 칼로리를 재계산하여 User_Calorie_Log 에 저장하는 헬퍼

- routes/food.js

GET /food : 사용자의 식단 기록 조회 및 오늘 섭취 칼로리 합계 계산

POST /food : multer 로 이미지 업로드 후 FoodLog 생성

PUT /food/:id : 식단 종류/칼로리 수정 DELETE /food/:id : 식단 삭제 식단 등록/수정/삭제 후 health.recalcTodayCalorieForUser(userId)를 호출하여 오늘 칼로리 현황에 즉시 반영

3-3. views/

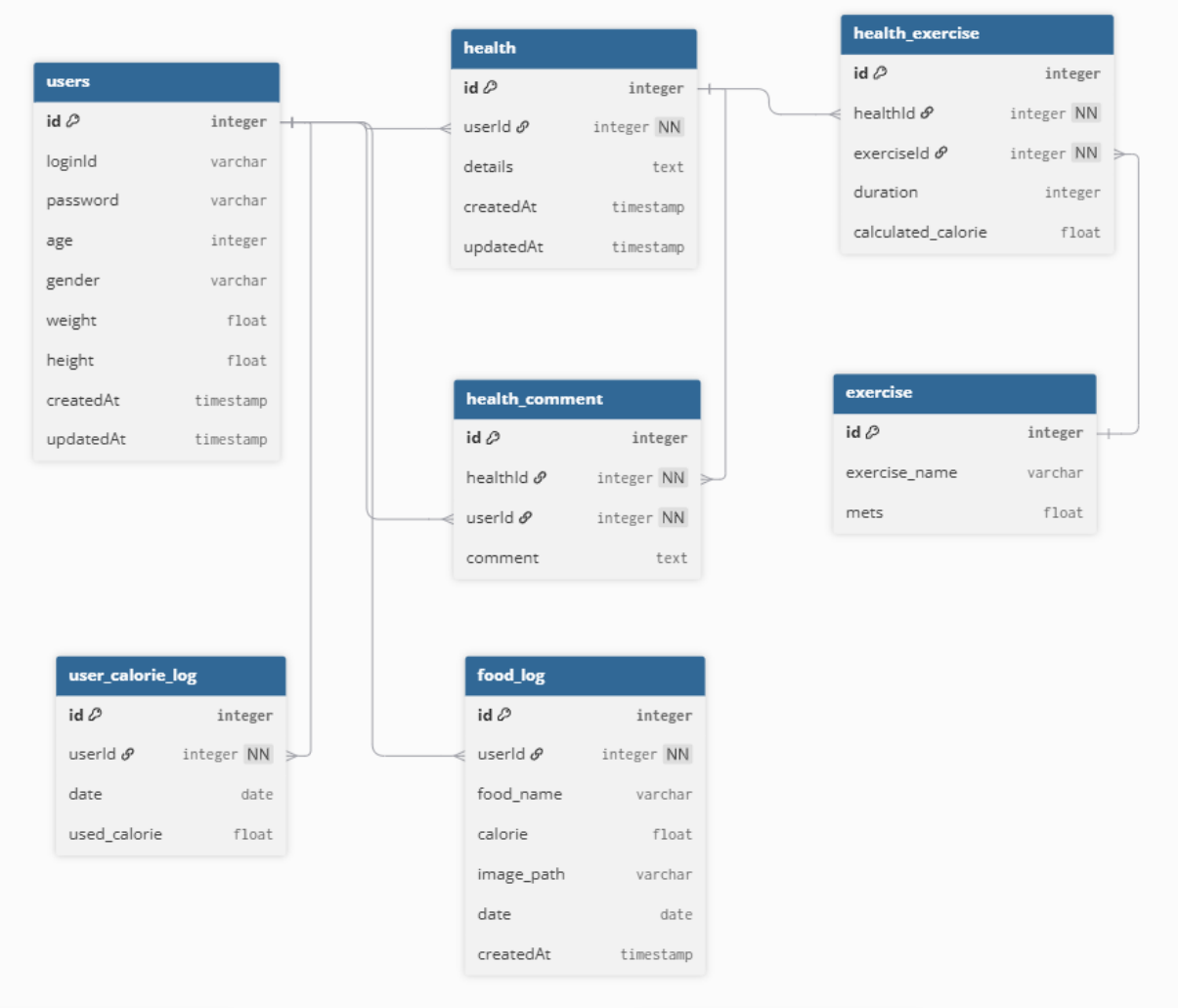
- login.ejs, register.ejs, profile.ejs, welcome.ejs
  인증 및 회원 정보 관련 화면 템플릿
- health.ejs
  운동 선택, 운동 기록 목록, 댓글, 오늘의 칼로리 현황 표시 템플릿
- food.ejs
  식단 등록 및 목록 표시 템플릿, 이미지 썸네일 렌더링

3-4. models/

Sequelize 모델로 User, Health, Exercise, Health_Exercise, Health_comment, FoodLog, User_Calorie_Log 등을 정의합니다.

특히 Health_Exercise 모델을 통해 Health – Exercise 사이를 N:M 구조로 연결하고, 각 운동의 duration(분)과 calculated_calorie 를 함께 저장하여, 한 번의 운동 세션에서 여러 운동을 기록할 수 있게 한 것이 추가 기능의 핵심입니다.

4. 데이터베이스 설계



4-1. 주요 테이블

- User

id, loginId, password, age, gender, weight, height, createdAt, updatedAt

모든 기능의 기준이 되는 회원 정보 테이블이며, BMR 계산에 필요한 신체 정보를 포함합니다.

- Health (운동 기록)

id, userId(FK), details, createdAt, updatedAt

한 번의 운동 세션마다 한 행이 생성되며, userId 를 통해 User 와 1:N 관계를 형성합니다.

Exercise (운동 마스터) id, exercise_name, mets 운동 종류별 METs 값을 정의하는 마스터 테이블로, 운동 소모 칼로리 계산에 사용됩니다.

- Health_Exercise (운동– 세션 N:M + 칼로리)

id, healthId(FK), exerciseId(FK), duration, calculated_calorie

한 Health(운동 세션)에 여러 Exercise 를 연결할 수 있도록 하는 N:M 연결 테이블입니다.
동시에 각 운동을 몇 분 했고 그로 인해 몇 kcal 를 소모했는지 저장합니다.

- Health_comment (운동 댓글)

id, healthId(FK), userId(FK), comment, createdAt, updatedAt

한 운동 기록에 여러 댓글을 달 수 있는 1:N 구조입니다.

- FoodLog (식단 기록)

id, userId(FK), food_name, calorie, image_path, date, createdAt
사용자가 먹은 식단 이름과 직접 입력한 칼로리, 업로드된 이미지 경로, 식사 날짜를 저장합니다.

- User_Calorie_Log (하루 순 소모 칼로리 로그)

id, userId(FK), date, used_calorie recalcTodayCalorieForUser()에서 계산한 하루 순 소모 칼로리를 날짜별로 기록합니다.

4-2. 관계 요약

- User – Health : 1:N, 한 사용자는 여러 운동 기록을 가질 수 있음
- Health – Exercise : N:M, Health_Exercise 를 통한 다대다 관계
- User – FoodLog : 1:N, 한 사용자는 여러 식단 기록을 가질 수 있음
- User – User_Calorie_Log : 1:N, 한 사용자의 일별 칼로리 로그를 여러 건 저장
- Health – Health_comment : 1:N, 한 운동 기록에 여러 댓글 연결

이 구조를 통해 운동 기록(Health/Health_Exercise/Exercise), 식단 기록(FoodLog), 하루 요약(User_Calorie_Log)을 유연하게 확장할 수 있으며, METs 기반 칼로리 계산, 다중 운동 기록, 이미지 포함 식단 기록 기능을 모두 지원합니다.

5. 사용자인터렉션별 요청/응답 흐름

각 인터랙션은 다음 형식을 따릅니다.

[사용자 인터렉션 → 프론트 코드 요청 → 서버의 응답 → 프론트 코드 응답 수령

→ 웹 화면에 응답 결과 제시]

5-1. 회원가입 (로컬 계정)

사용자 인터렉션

- 사용자가 register.ejs 에서 아이디, 비 번호, 나이, 성별, 체중, 신장을 입력하고 "회원가입" 버튼을 클릭합니다.

프론트코드요청

- <form method="post" action="/register"> 가 POST /register 요청을 전송합니다.
- body 에는 id, password, age, gender, weight, height 값이 포함됩니다.

서버의 응답

- routes/auth.js 의 router.post('/register')에서 동일 loginId 가 있는지 User.findOne 으로확인합니다.
- 중복이 없으면 User.create(...)로 새 사용자를 생성합니다.
- "회원가입 완료!" alert 와 함께 로그인 페이지(/)로 이동하는 스크립트를 담은 HTML 응답을보냅니다.

프론트코드응답수령

- 브라우저가 응답 HTML 을 렌더링하면서 <script> 의 alert 와 location.href 실행.

웹 화면에 응답 결과 제시

- "회원가입 완료! 로그인 해주세요." 알림창이 뜬 후 로그인 페이지가 표시됩니다.

5-2. 로그인 (로컬, Passport)

사용자 인터렉션

- 사용자가 login.ejs 에서 아이디와 비밀번호를 입력하고 "로그인" 버튼을 클릭합니다.

프론트코드요청

- <form method="post" action="/admit"> 가 POST /admit 요청을 전송합니다.

서버의 응답

- routes/auth.js 의 router.post('/admit')에서 passport.authenticate('local')로 인증을 수행합니다.
- 인증 성공 시 req.login(user, ...)으로 세션에 사용자 정보를 저장하고 /welcome 으로 302 리다이렉트합니다.
- app.js 의 미들웨어에서 req.user 가 있으면 req.session.userId, req.session.user 에 로그인정보를 동기화합니다.

프론트코드응답수령

- 브라우저가 302 응답을 따라 /welcome 으로 GET 요청을 보냅니다.

웹화면에응답결과제시

- welcome.ejs 가 렌더링되어 "<아이디> 님 환 합니다!" 메시지와 함께 메뉴(운동 기록, 식단 등록, 회원 정보 수정 등)가 나타납니다.

- 5-3. 소셜 로그인 (카카오/네이버)

사용자 인터렉션

- 사용자가 login.ejs 에서 "카카오 계정으로 로그인" 또는 "네이버 계정으로 로그인" 링크를 클릭합니다.

프론트 코드 요청

- 브라우저가 GET /auth/kakao 또는 GET /auth/naver 요청을 전송합니다.

서버의 응답

- routes/auth.js 에서 passport.authenticate('kakao') 또는 'naver'를 호출하여 각 제공자의인증 서버로 리다이렉트합니다.
- 사용자가 제공자 화면에서 로그인/동의를 완료하면, 제공자가 GET /auth/.../callback?code=... 로 다시 돌아옵니다.
- 콜백 라우터에서 액세스 토큰과 프로필을 받아 User 테이블에 provider 별 계정을 찾거나 새로생성합니다.
- 인증 성공 후 /welcome 으로 리다이렉트 합니다.

프론트코드응답수령

- 브라우저가 /welcome 페이지를 요청하여 HTML 응답을 수신합니다.

웹 화면에 응답 결과 제시

- 소셜 로그인 사용자의 경우, 나이/성별/체중/신장 값이 없으면 /user/profile 로 안내하는 스크립트가 실행되어 먼저 프로필 입력 페이지로 이동하게 됩니다.

5-4. 회원 정보 수정 (/user/profile)

사용자 인터렉션

- 사용자가 "회원 정보 수정" 메뉴를 클릭하거나 소셜 첫 로그인 후 안내에 따라 /user/profile 페이지에서 정보를 입력하고 "정보 수정"을 누릅니다.

프론트코드요청

- <form method="post" action="/user/profile"> 이 POST /user/profile 요청을 전송합니다.

서버의 응답

- routes/auth.js 의 router.post('/user/profile')에서 현재 로그인된 사용자 ID 를 가져옵니다.
- 전달된 age, gender, weight, height 로 User.update(...)를 수행합니다.
- "회원 정보가 수정되었습니다." alert 후 /welcome 으로 이동하는 스크립트를 응답합니다.

프론트코드응답수령 및 결과

- 알림창 이후 환영 페이지로 돌아가며, 이후 BMR 및 칼로리 계산에 수정된 정보가 사용됩니다.

5-5. 운동 기록 등록 및 조회 (Health + Health_Exercise)

사용자 인터렉션

- 사용자가 메뉴에서 "운동 기록"을 클릭합니다.
- health.ejs 에서 운동 버튼을 선택하고, 운동 시간과 세부 사항을 입력한 뒤 "운동 기록 등록"을 누릅니다. 프론트코드요청

- "운동 기록" 메뉴 클릭 시 브라우저가 GET /health 요청을 전송합니다.
- 운동 등록 폼은 <form method="post" action="/health"> 로 exerciseIds, durations, details 등을 담아 POST /health 요청을 전송합니다.

서버의 응답

- GET /health : Exercise.findAll()로 운동 목록을, Health.findAll({ include: [Health_Exercise, Health_comment, User] })로 사용자의 운동 기록과 연관된 운동/댓글을 조회하여 health.ejs 를 렌더링합니다.
- POST /health :

Health.create({ userId, details, ... })로 운동 세션을 생성합니다.

각 exerciseId 와 duration(분)에 대해 METs 와 체중을 이용해 칼로리를 계산하고, Health_Exercise 레코드를 생성합니다.

recalcTodayCalorieForUser(userId)를 호출하여 오늘 칼로리 로그를 갱신합니다. /health 로 리다이렉트합니다.

프론트코드응답수령 및 결과

- 브라우저가 새로 렌더링된 /health 페이지를 표시하고, 상단 "오늘의 칼로리 현황"은/health/calorie AJAX 로 계산 결과를 보여줍니다.
- 하단 목록에는 방금 등록한 운동 기록과 각 운동별 소모 칼로리가 표시됩니다.

5-6. 운동 기록 수정 및 삭제

(1) 운동 기록 수정

사용자 인터렉션

- 사용자가 특정 운동 기록 카드의 "수정" 버튼을 누르고, 모델에서 운동 종류/시간/세부 사항을 다시 선택한 뒤 "저장"을 클릭합니다.

프론트코드요청

- health.ejs 의 JavaScript 가 fetch('/health/:id', { method:'PUT', body: { exerciseIds,durations, details } })로 PUT 요청을 보냅니다.

서버의 응답

- routes/health.js 에서 해당 Health 를 찾고 기존 Health_Exercise 를 삭제한 뒤, 새 운동 목록을기준으로 다시 생성합니다.
- 운동별 METs 와 duration(분)을 이용해 calculated_calorie 를 재계산하고 합산합니다.
- recalcTodayCalorieForUser(userId)를 다시 호출하여 오늘 칼로리 로그를 수정합니다.- { success:true, ... } 형태의 JSON 을 응답합니다.

프론트코드응답수령 및 결과

- 응답을 받은 후 location.reload()로 페이지를 새로고침하여, 수정된 운동 기록과 칼로리가 화면에 반환 됩니다.

(2) 운동 기록 삭제

사용자 인터렉션

- 운동 기록 카드의 "삭제" 버튼을 누르고 confirm 창에서 "확인"을 클릭합니다.

프론트코드요청

- JS 에서 fetch('/health/:id', { method:'DELETE' })로 DELETE 요청을 전송합니다.

서버의응답

- Health.destroy({ where: { id, userId } })로 해당 기록을 삭제합니다.
- onDelete: 'CASCADE' 설정으로 연관 Health_Exercise, Health_comment 도 함께 삭제 됩니다.
- recalcTodayCalorieForUser(userId)를 호출해 오늘 칼로리 현황을 재계산하고 { success:true } JSON 을 반환합니다.

프론트코드응답수령 및 결과

- location.reload() 후 삭제된 기록이 목록에서 사라지고, 상단 칼로리 현황에서도 운동 소모칼로리가 줄어든 값으로 표시됩니다.

5-7. 운동 댓글 등록 / 수정 / 삭제 (Health_comment)

(1) 댓글 등록

사용자 인터렉션

- 사용자가 운동 기록 아래의 댓글 입력창에 내용을 적고 "댓글 작성" 버튼을 클릭합니다.

프론트코드요청

- <form method="post" action="/health/:id/comments"> 가 POST /health/:id/comments 요청을 전송합니다.

서버의응답

- Health_comment.create({ healthId:id, userId, comment })로 댓을 저장하고 /health 로 리다이렉트합니다.

(2) 댓글 수정

사용자인터렉션

- 댓글 옆의 "댓글 수정" 버튼을 누르면 prompt 창이 뜨고, 새 내용을 입력합니다.

프론트코드요청

- fetch('/health/:healthId/comments/:commentId', { method:'PUT', body:{ comment } })로 PUT 요청을 전송합니다.

서버의응답

- Health_comment.update(...) 후 { success:true, comment, date } JSON 을 반환합니다.

(3) 댓글 삭제

사용자인터렉션

- 댓글 옆의 "댓글 삭제" 버튼을 누르고 confirm 창에서 "확인"을 클릭합니다.

프론트코드요청

- fetch('/health/:healthId/comments/:commentId', { method:'DELETE' })로 DELETE 요청을 전송합니다.

서버의응답

- Health_comment.destroy(...) 후 { success:true } JSON 을 반환합니다.

프론트코드응답수령 및 결과

- 각 경우 모두 새로고침 후 댓글 목록이 갱신되어, 추가/수정/삭제 결과가 바로 화면에 반영됩니다.

5-8. 식단 등록 / 조회 / 수정 / 삭제 (FoodLog)

사용자 인터렉션 (등록)

- 사용자가 "식단 등록" 메뉴를 클릭하고, 식사 종류/칼로리/날짜를 입력하고 음식 이미지를 여러장 선택한 뒤 "등록" 버튼을 누릅니다.

프론트코드요청

- GET /food : 페이지 로딩 시 브라우저가 /food 를 요청합니다.

- POST /food : <form method="post" enctype="multipart/form-data" action="/food"> 가 meal_type, food_name, calorie, date, images[]를 포함한 요청을 전송합니다.

서버의응답

- GET /food :

FoodLog.findAll({ where:{ userId } })로 식단 기록을 조회합니다.

오늘 날짜에 해당하는 기록만 골라 칼로리를 합산하여 todayCalorie 를 계산하고, foods + todayCalorie 로 food.ejs 를 렌더링합니다.

- POST /food : multer 로 이미지를 public/uploads 에 저장하고 파일 경로 배열을 구성합니다.

FoodLog.create({ userId, food_name, calorie, image_path, date })로 식단 기록을 추가합니다.

health.recalcTodayCalorieForUser(userId)를 호출하여 오늘 칼로리 로그를 갱신합니다. /food 로 리다이렉트 합니다.

프론트코드 응답수령 및 결과

- /food 페이지가 다시 로딩되며, 상단 "오늘 섭취 칼로리"에 오늘 날짜 식단 칼로리 합계가 표시되고, 하단 목록에 새 식단 카드와 이미지가 나타납니다.

식단 수정/삭제의 경우에도, 각 카드의 "수정", "삭제" 버튼이 fetch('/food/:id', { method:'PUT/DELETE' }) 요청을 보내고, 서버가 FoodLog 를 수정/삭제한 후 recalcTodayCalorieForUser(userId)를 호출하여 오늘 칼로리 현황에 반영하는 동일한 흐름을 따릅니다.

5-9. 오늘 칼로리 현황 조회 (/health/calorie)

사용자 인터렉션

- 사용자가 /health 페이지에 들어오거나, "다시 계산" 버튼을 클릭합니다.

프론트 코드 요청

- health.ejs 의 JS 함수 fetchCalorie()가 fetch('/health/calorie')로 GET 요청을 보냅니다.

서버의 응답

- routes/health.js 의 recalcTodayCalorieForUser(userId)에서 다음을 수행합니다.

  User 의 나이, 성별, 체중, 신장을 이용해 BMR 을 계산합니다.

  오늘 날짜 범위의 Health_Exercise 를 조회하여 운동 소모 칼로리를 합산합니다.

오늘 날짜의 FoodLog 를 조회하여 섭취 칼로리를 합산합니다.

기초대사량 기준 현재까지 소모한 칼로리 + 운동 소모 - 섭취 칼로리로 순 소모(net)를 계산합니다.

User_Calorie_Log 에 오늘 used_calorie 를 저장/갱신합니다.

{ bmr, baseBurned, exerciseBurned, burned, intake, net, date } JSON 을 반환합니다.

프론트코드응답수령 및 결과

- health.ejs 에서 응답 JSON 을 받아 bmrSpan, intakeSpan, netSpan 등 여러 요소의 내용을 업데이트합니다.
- 상단 "오늘의 칼로리 현황" 박스에 기초대사량, 운동 소모, 섭취 칼로리, 순 소모 칼로리가 한눈에 보이도록 출력됩니다.