API 연동 규격서

태그	설계 필수

1. 개요

- **프로젝트명**: LinkCare
- **작성일자**: 2025-10-14

2. 공통 규칙

- 모든 API는 HTTPS 프로토콜을 사용해야 함
- AccessToken 만료 시 RefreshToken으로 재발급 필요

2.1 요청/응답 형식

- **Content-Type**: `application/json; charset=utf-8`
- **인증 방식**: JWT (Header `Authorization: Bearer {token}`)
- **응답 포맷**

```
{
  "code": "SUCCESS",
  "message": "요청이 성공적으로 처리되었습니다.",
  "data": { }
}
```

2 .2 에러 응답 공통 포맷

```
{
  "code": "ERROR_CODE", // ex) 1001
  "message": "오류 설명"
}
```

에러 예시

코드	설명
INVALID_PARAM	파라미터 오류
UNAUTHORIZED	인증 실패
FORBIDDEN	권한 없음
NOT_FOUND	리소스 없음
INTERNAL_ERROR	서버 내부 오류

3. API 상세 목록

▼ Auth

A. Auth / User

1. 회원가입 (Sign Up)

■ 개요

신규 사용자가 이메일과 비밀번호를 통해 회원가입을 수행한다.

가입 시 서버는 비밀번호를 해시 처리 후 저장하고, 기본 프로필을 생성한다.

■ 요청 정보

- **URL**: `/api/auth/signup`
- **Method**: `POST`
- **Content-Type**: `application/json`

■ 요청 헤더

Key	Value	설명
`Content-Type`	`application/json`	JSON 형식 요청
`User-Agent`	`FitLink-App/1.0.0`	(선택) 앱 버전 정보

■ 요청 바디

필드명	타입	필수	설명
email	string	Y	사용자 이메일 (고유값)
password	string	Y	비밀번호 (SHA256 해시)
nickname	string	Y	닉네임
birthDate	string	N	생년월일 (YYYY-MM-DD)
gender	string	N	M / F

■ 요청 예시

```
{
  "email": "user@example.com",
  "password": "test1234",
  "nickname": "운동좋아하는펭귄",
  "birthDate": "1998-07-15",
  "gender": "M"
}
```

■ 응답 정보

✅ 성공 (201 Created)

```
{
  "code": "SUCCESS",
  "message": "회원가입이 완료되었습니다.",
  "data": {
    "userId": 12,
    "email": "user@example.com",
    "nickname": "운동좋아하는펭귄",
    "createdAt": "2025-10-14T10:12:45Z"
  }
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "DUPLICATE_EMAIL",
```

```
  "message": "이미 등록된 이메일입니다."
}
```

❌ 실패 (422 Validation Error)

```
{
  "code": "INVALID_FORMAT",
  "message": "비밀번호 형식이 올바르지 않습니다. (8자 이상, 특수문자 포함)"
}
```

---

2. 로그인 (Sign In)

■ 개요

등록된 이메일과 비밀번호를 통해 로그인하고, JWT Access/Refresh Token을 발급받는다.

■ 요청 정보

- **URL**: `/api/auth/login`
- **Method**: `POST`
- **Content-Type**: `application/json`

■ 요청 헤더

Key	Value	설명
`Content-Type`	`application/json`	JSON 요청
`User-Agent`	`FitLink-App/1.0.0`	앱 버전

■ 요청 바디

필드명	타입	필수	설명
`email`	string	Y	사용자 이메일
`password`	string	Y	비밀번호

■ 요청 예시

```
{
  "email": "user@example.com",
  "password": "test1234"
}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "로그인 성공",
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "user": {
      "userId": 12,
      "nickname": "운동좋아하는펭귄",
      "email": "user@example.com"
    }
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_CREDENTIALS",
  "message": "이메일 또는 비밀번호가 올바르지 않습니다."
}
```

❌ 실패 (403 Forbidden)

```
{
  "code": "ACCOUNT_DISABLED",
```

```
  "message": "탈퇴 또는 비활성화된 계정입니다."
}
```

3. 토큰 재발급 (Refresh Token)

■ 개요

만료된 Access Token을 갱신하기 위한 API.

Refresh Token이 유효해야 하며, 새 Access Token을 반환한다.

■ 요청 정보

- **URL**: `/api/auth/refresh`
- **Method**: `POST`
- **Content-Type**: `application/json`

■ 요청 헤더

Key	Value	설명
`Authorization`	`Bearer {refreshToken}`	Refresh Token 포함
`Content-Type`	`application/json`	JSON 요청

■ 요청 예시

```
{}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "토큰이 갱신되었습니다.",
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIs..."
```

```
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_REFRESH_TOKEN",
  "message": "유효하지 않은 Refresh Token입니다."
}
```

---

4. 로그아웃 (Logout)

■ 개요

현재 로그인 세션을 종료하고, 서버에서 Refresh Token을 폐기한다.

■ 요청 정보

- **URL**: `/api/auth/logout`
- **Method**: `POST`
- **Content-Type**: `application/json`

■ 요청 헤더

Key	Value	설명
`Authorization`	`Bearer {accessToken}`	로그인된 사용자 Access Token
`Content-Type`	`application/json`	JSON 요청

■ 요청 예시

```
{}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "로그아웃이 완료되었습니다."
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

---

5. 회원 탈퇴 (Delete Account)

■ 개요

회원의 데이터를 삭제하거나 비활성화 상태로 전환한다.

■ 요청 정보

- **URL**: `/api/users/me`
- **Method**: `DELETE`

■ 요청 헤더

Key	Value	설명
`Authorization`	`Bearer {accessToken}`	사용자 인증 토큰

■ 요청 예시

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
```

```
  "message": "회원 탈퇴가 완료되었습니다."
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "로그인이 필요한 요청입니다."
}
```

▼ Samsung Health

B. Samsung Health

1. Samsung Health 데이터 권한 연동

■ 개요

앱/워치에서 삼성 헬스 SDK Access Token을 발급받아 서버에 전달하면,

서버는 해당 토큰으로 헬스 데이터를 가져오기 위한 권한을 확인한다.

■ 요청 정보

- **URL**: `/api/health/samsung/connect`
- **Method**: `POST`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Content-Type`	`application/json`	JSON 요청
`User-Agent`	`FitLink-App/1.0.0`	앱/워치 정보

요청 바디

필드명	타입	필수	설명
`accessToken`	string	Y	Samsung Health SDK에서 발급된 Access Token
`deviceId`	string	N	워치/기기 식별자 (선택)
`scopes`	string[]	N	요청할 헬스 데이터 종류 (`heart_rate`, `steps`, `sleep` 등)

요청 예시

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "deviceId": "galaxy-watch6-1234",
  "scopes": ["heart_rate", "steps", "sleep"]
}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "삼성 헬스 연동 권한 확인 완료",
  "data": {
    "deviceId": "galaxy-watch6-1234",
    "scopesGranted": ["heart_rate", "steps", "sleep"],
    "connectedAt": "2025-10-14T15:30:00Z"
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_HEALTH_TOKEN",
  "message": "유효하지 않은 Samsung Health Access Token입니다."
}
```

❌ 실패 (403 Forbidden)

```
{
  "code": "INSUFFICIENT_SCOPE",
  "message": "요청한 헬스 데이터 권한이 허용되지 않았습니다."
}
```

2. 헬스 데이터 조회

■ 개요

서버가 삼성 헬스 SDK Access Token을 사용하여 실제 헬스 데이터를 조회한다.

- 서버가 직접 삼성 API 호출
- JWT 없음, 서버가 Access Token으로만 호출 가능

■ 요청 정보

- **URL**: `/api/v1/health/samsung/data`
- **Method**: `GET`
- **Query Parameter**: `deviceId=<워치ID>&type=<데이터종류>&start=<YYYY-MM-DD>&end=<YYYY-MM-DD>`

요청 예시

```
{}
```

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <Samsung SDK Access Token>`	SDK에서 발급된 토큰

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "데이터 조회 완료",
  "data": [
    { "date": "2025-10-01", "steps": 10234 },
    { "date": "2025-10-02", "steps": 8750 },
    { "date": "2025-10-03", "steps": 11200 }
  ]
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_HEALTH_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

❌ 실패 (404 Not Found)

```
{
  "code": "DATA_NOT_FOUND",
  "message": "해당 기간에 대한 헬스 데이터가 없습니다."
}
```

▼ User Profile

C. 유저 프로필 API

1. 유저 프로필 조회 API

■ 개요

로그인한 유저가 자신의 프로필 정보를 조회할 수 있는 API.

- 인증: 서버 JWT Access Token 필요

■ 요청 정보

- **URL:** `/api/users/me`
- **Method:** `GET`
- **Content-Type:** `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	서버 로그인 토큰
`Content-Type`	`application/json`	JSON 요청

■ 요청 예시

```
{}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "유저 프로필 조회 성공",
  "data": {
    "userId": 24,
    "email": "user@example.com",
    "nickname": "FitRunner",
    "profileImageUrl": "https://cdn.fitlink.com/users/24/profile.png",
    "createdAt": "2025-10-01T12:00:00Z",
    "updatedAt": "2025-10-10T15:00:00Z",
    "healthLinked": true,
    "healthScopes": ["heart_rate", "steps", "sleep"]
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

2. 유저 프로필 수정 API

■ 개요

유저가 자신의 닉네임, 프로필 이미지 등을 수정할 수 있는 API.

- 인증: 서버 JWT Access Token 필요

■ 요청 정보

- **URL**: `/api/users/me`
- **Method**: `PUT`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	서버 로그인 토큰
`Content-Type`	`application/json`	JSON 요청

요청 바디

필드명	타입	필수	설명
`nickname`	string	N	닉네임 변경
`profileImageUrl`	string	N	프로필 이미지 URL 변경
`bio`	string	N	자기소개

■ 요청 예시

```
{
  "nickname": "HealthChampion",
```

```
  "profileImageUrl": "https://cdn.fitlink.com/users/24/new-profile.png",
  "bio": "매일 운동 기록 중!"
}
```

◾ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "유저 프로필 수정 성공",
  "data": {
    "userId": 24,
    "nickname": "HealthChampion",
    "profileImageUrl": "https://cdn.fitlink.com/users/24/new-profile.png",
    "bio": "매일 운동 기록 중!",
    "updatedAt": "2025-10-14T15:50:00Z"
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "INVALID_NICKNAME",
  "message": "닉네임은 2~20자 사이여야 합니다."
}
```

▼ Friend

3. 친구 목록 조회

■ 개요

로그인한 유저의 친구 목록을 조회하는 API.

- 인증 필요: 서버 JWT Access Token
- 친구의 닉네임, 프로필 이미지, 상태(온라인/오프라인) 등 반환 가능

■ 요청 정보

- **URL**: `/api/friends`
- **Method**: `GET`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	서버 로그인 토큰
`Content-Type`	`application/json`	JSON 요청

■ 요청 예시

```
{}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "친구 목록 조회 성공",
  "data": [
    {
      "friendId": 12,
      "nickname": "RunnerA",
```

```
    "profileImageUrl": "https://cdn.fitlink.com/users/12/profile.png",
    "status": "online"
   },
   {
    "friendId": 34,
    "nickname": "CyclistB",
    "profileImageUrl": "https://cdn.fitlink.com/users/34/profile.png",
    "status": "offline"
   }
  ]
}
```


❌ 실패 (401 Unauthorized)


```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```


4. 친구 요청 보내기

■ 개요

특정 유저에게 친구 요청을 보내는 API.

- 인증 필요: 서버 JWT Access Token
- 요청 상대 유저는 이미 가입되어 있어야 함

■ 요청 정보

- **URL**: `/api/friends/request`
- **Method**: `POST`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	서버 로그인 토큰
`Content-Type`	`application/json`	JSON 요청

요청 바디

필드명	타입	필수	설명
`friendId`	integer	Y	요청할 상대 유저 ID

요청 예시

```
{
  "friendId": 34
}
```

◾ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "친구 요청 전송 완료"
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "ALREADY_FRIEND",
  "message": "이미 친구로 등록되어 있습니다."
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

5. 친구 요청 수락/거절

■ 개요

받은 친구 요청을 수락 또는 거절하는 API.

- 인증 필요: 서버 JWT Access Token

■ 요청 정보

- **URL**: `/api/friends/request/:requestId`
- **Method**: `PUT`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	서버 로그인 토큰
`Content-Type`	`application/json`	JSON 요청

요청 바디

필드명	타입	필수	설명
`action`	string	Y	`accept` 또는 `decline`

요청 예시

```
{
  "action": "accept"
}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "친구 요청 수락 완료"
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "INVALID_ACTION",
  "message": "accept 또는 decline만 가능합니다."
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

6. 친구 삭제

■ 개요

등록된 친구를 삭제하는 API.

- 인증 필요: 서버 JWT Access Token

■ 요청 정보

- **URL**: `/api/v1/friends/:friendId`
- **Method**: `DELETE`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	서버 로그인 토큰
`Content-Type`	`application/json`	JSON 요청

■ 요청 예시

```
{}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "친구 삭제 완료"
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

❌ 실패 (404 Not Found)

```
{
  "code": "FRIEND_NOT_FOUND",
  "message": "삭제할 친구가 존재하지 않습니다."
}
```

▼ Group

D. 모임 API

1. 모임 생성

■ 개요

사용자가 운동 모임을 생성하는 API.

- 인증 필요: 서버 JWT Access Token
- 모임 이름, 설명, 참여 제한 인원, 운동 유형 등을 설정 가능

■ 요청 정보

- **URL**: `/api/groups`
- **Method**: `POST`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	서버 로그인 토큰
`Content-Type`	`application/json`	JSON 요청

요청 바디

필드명	타입	필수	설명
`name`	string	Y	모임 이름
`description`	string	N	모임 설명
`maxMembers`	integer	N	최대 참여 인원
`category`	string	N	운동 유형 (`running` , `cycling` , `yoga` 등)

요청 예시

```
{
  "name": "아침 러닝 모임",
  "description": "매일 아침 7시에 5km 러닝",
  "maxMembers": 10,
```

```
  "category": "running"
}
```

■ 응답 정보

✅ 성공 (201 Created)

```
{
  "code": "SUCCESS",
  "message": "모임 생성 완료",
  "data": {
    "groupId": 101,
    "name": "아침 러닝 모임",
    "description": "매일 아침 7시에 5km 러닝",
    "maxMembers": 10,
    "category": "running",
    "creatorId": 24,
    "createdAt": "2025-10-14T16:20:00Z"
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "INVALID_INPUT",
  "message": "모임 이름은 2~50자 사이여야 합니다."
}
```

2. 모임 목록 조회

■ 개요

내가 속한 모 목록을 조회하는 API.

- 인증 필요: 서버 JWT Access Token

■ 요청 정보

- **URL**: `/api/groups`
- **Method**: `GET`

요청 예시

```
{}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "모임 목록 조회 성공",
  "data": [
    {
      "groupId": 101,
      "name": "아침 러닝 모임",
      "description": "매일 아침 7시에 5km 러닝",
      "category": "running",
      "currentMembers": 5,
      "maxMembers": 10
    },
    {
      "groupId": 102,
      "name": "저녁 사이클링 모임",
      "description": "퇴근 후 10km 라이딩",
      "category": "cycling",
      "currentMembers": 3,
      "maxMembers": 8
```

```
    }
  ]
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

3. 모임 상세 조회

■ 개요

모임 정보를 상세히 조회하고, 멤버 리스트를 확인하는 API.

- 인증 필요: 서버 JWT Access Token

■ 요청 정보

- **URL**: `/api/groups/{groupId}`
- **Method**: `GET`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	서버 로그인 토큰

요청 예시

```
{}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "모임 상세 조회 성공",
  "data": {
    "groupId": 101,
    "name": "아침 러닝 모임",
    "description": "매일 아침 7시에 5km 러닝",
    "category": "running",
    "currentMembers": 5,
    "maxMembers": 10,
    "members": [
      { "userId": 24, "nickname": "FitRunner", "profileImageUrl": "..." },
      { "userId": 12, "nickname": "RunnerA", "profileImageUrl": "..." }
    ]
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

❌ 실패 (404 Not Found)

```
{
  "code": "GROUP_NOT_FOUND",
  "message": "해당 모임이 존재하지 않습니다."
}
```

4. 모임 참여

■ 개요

사용자가 특정 모임에 참여 요청을 보내는 API.

- 인증 필요: 서버 JWT Access Token
- 참여 요청 상태는 서버에서 관리 (`pending` , `accepted`)

■ 요청 정보

- **URL**: `/api/groups/{groupId}/join`
- **Method**: `POST`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	서버 로그인 토큰
`Content-Type`	`application/json`	JSON 요청

요청 바디

```
{
  "groupId" : 5
}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "모임 참여 요청 완료",
  "data": {
    "groupId": 101,
    "requestStatus": "pending",
    "requestedAt": "2025-10-14T17:10:00Z"
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "ALREADY_MEMBER",
  "message": "이미 참여한 모임입니다."
}
```

▼ Notification

E. 알림

1. 알림 목록 조회

■ 개요

사용자가 받은 알림(Notification) 목록을 조회하는 API.

친구 요청, 모임 초대, 운동 제안 등 다양한 유형의 알림을 페이지 단위로 불러온다.

- 인증 필요: 서버 JWT Access Token
- 알림 유형별 필터링 및 읽음 여부 필터링 가능

■ 요청 정보

- **URL**: `/api/notifications`
- **Method**: `GET`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	서버 로그인 토큰
`Content-Type`	`application/json`	JSON 요청

요청 파라미터

Key	Type	필수	설명
`page`	int	❌	페이지 번호 (기본값: 0)
`size`	int	❌	페이지 크기 (기본값: 10)
`isRead`	boolean	❌	읽음 여부 필터링 (true / false)
`type`	string	❌	알림 유형 필터 (`FRIEND_REQUEST`, `GROUP_INVITE`, `HEALTH_ALERT` 등)

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "알림 목록 조회 성공",
  "data": {
    "notifications": [
      {
        "id": 101,
        "type": "FRIEND_REQUEST",
        "title": "새 친구 요청",
        "content": "민사빈님이 친구 요청을 보냈습니다.",
        "isRead": false,
        "createdAt": "2025-10-15T10:21:00Z"
      },
      {
        "id": 99,
        "type": "GROUP_INVITE",
        "title": "모임 초대",
        "content": "'러너스 클럽'에서 당신을 초대했습니다.",
```

```
      "isRead": true,
      "createdAt": "2025-10-14T08:13:20Z"
    }
  ],
  "page": 0,
  "size": 10,
  "totalElements": 25
 }
}
```

❌ 실패 (401 Unauthorized)

```
{
 "code": "INVALID_TOKEN",
 "message": "유효하지 않은 토큰입니다."
}
```

❌ 실패 (500 Internal Server Error)

```
{
 "code": "SERVER_ERROR",
 "message": "서버 내부 오류가 발생했습니다."
}
```

2. 알림 읽음 처리

■ 개요

사용자가 특정 알림을 **읽음 처리**하는 API.

읽음 상태(`isRead`)를 `true`로 변경하며, 이후 알림 목록 조회 시 필터링에 반영된다.

- 인증 필요: 서버 JWT Access Token
- 읽음 처리 후 최신 상태(`updatedAt`) 반환

■ 요청 정보

- **URL:** `/api/notifications/{notificationId}/read`
- **Method:** `PATCH`
- **Content-Type:** `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	서버 로그인 토큰
`Content-Type`	`application/json`	JSON 요청

요청 경로 변수

Key	Type	필수	설명
`notificationId`	long	✅	읽음 처리할 알림의 고유 ID

요청 바디

```
{
  "isRead": true
}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "알림 읽음 처리 완료",
  "data": {
    "id": 101,
    "isRead": true,
    "updatedAt": "2025-10-15T11:12:00Z"
  }
}
```

❌ 실패 (404 Not Found)

```
{
  "code": "NOTIFICATION_NOT_FOUND",
  "message": "해당 알림을 찾을 수 없습니다."
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 토큰입니다."
}
```

❌ 실패 (500 Internal Server Error)

```
{
  "code": "SERVER_ERROR",
  "message": "서버 내부 오류가 발생했습니다."
}
```

3. 실시간 알림 소켓

■ 개요

사용자가 앱 내에서 **실시간으로 알림(Notification)** 을 받을 수 있도록 하는 WebSocket 기반 API.

친구 요청, 모임 초대, 건강 알림 등 주요 이벤트 발생 시 서버에서 즉시 클라이언트로 전송한다.

- 인증 필요: 서버 JWT Access Token
- 연결 시 Query Parameter로 토큰 전달
- 연결 유지 중, 서버는 실시간 알림 이벤트를 Push 형식으로 전송

■ 요청 정보

- **URL**: `/ws/notifications`
- **Method**: `ws` (WebSocket)
- **Protocol**: `wss://api.healthfit.app/ws/notifications`

연결 예시

```
const socket = new WebSocket(
  "wss://api.healthfit.app/ws/notifications?token=eyJhbGciOiJIUzI1NiIs..."
);
```

인증 및 연결

Key	Value	설명
`token`	`<JWT Access Token>`	WebSocket 연결 시 Query Param으로 전달
`Protocol`	`wss`	보안 WebSocket 프로토콜
`Heartbeat`	30초마다 ping/pong	연결 유지용 heartbeat 신호

■ 서버 → 클라이언트 메시지 (실시간 알림 수신)

```
{
  "event": "NEW_NOTIFICATION",
  "data": {
    "id": 112,
    "type": "GROUP_INVITE",
    "title": "모임 초대",
    "content": "'러너스 클럽'에서 당신을 초대했습니다.",
    "isRead": false,
    "createdAt": "2025-10-15T11:30:00Z"
  }
}
```

Key	설명
`event`	이벤트 타입 (예: `NEW_NOTIFICATION`)
`data.id`	알림 고유 ID
`data.type`	알림 유형 (`FRIEND_REQUEST`, `GROUP_INVITE` 등)
`data.title`	알림 제목
`data.content`	알림 본문
`data.isRead`	읽음 여부
`data.createdAt`	알림 생성 시각 (ISO 8601)

■ 클라이언트 → 서버 메시지 (읽음 처리, 확인 응답 등)

```
{
  "event": "READ_NOTIFICATION",
  "data": {
    "notificationId": 112
  }
}
```

Key	설명
`event`	클라이언트 액션 타입 (`READ_NOTIFICATION`)
`data.notificationId`	읽음 처리할 알림 ID

■ 응답 정보

✅ 성공 (연결 유지)

```
{
  "event": "CONNECTION_SUCCESS",
  "message": "WebSocket 연결이 성공적으로 설정되었습니다."
}
```

❌ 실패 (401 Unauthorized)

```
{
  "event": "ERROR",
```

```
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다. 연결이 종료됩니다."
}
```

❌ 실패 (500 Internal Server Error)

```
{
  "event": "ERROR",
  "code": "SERVER_ERROR",
  "message": "서버 내부 오류가 발생했습니다."
}
```

4. 푸시 알림 연동 (Firebase Cloud Messaging, FCM)

■ 개요

사용자가 앱을 실행 중이지 않아도, 서버에서 **중요 이벤트 발생 시 푸시 알림을 전송**할 수 있도록

Firebase Cloud Messaging(FCM)을 연동하는 API.

- **클라이언트 단말의 FCM 토큰 등록 및 관리**
- **서버에서 특정 사용자에게 알림 푸시 전송**
- **WebSocket 알림과 동기화 (읽음 처리 공유)**

■ 요청 정보 (1) FCM 토큰 등록

항목	내용
URL	`/api/v1/notifications/fcm/token`
Method	`POST`
Auth	Bearer Token (Access Token)
설명	사용자의 기기에서 발급받은 FCM 토큰을 서버에 등록

📩 Request

```
{
  "fcmToken": "eJ3Sk1KJ:APA91bGv4QlH6mCqJ8..."
}
```

📩 Response (성공)

```
{
  "status": "success",
  "message": "FCM 토큰이 정상적으로 등록되었습니다."
}
```

■ 요청 정보 (2) FCM 푸시 발송

항목	내용
URL	`/api/v1/notifications/fcm/send`
Method	`POST`
Auth	Bearer Token (Admin or System Account)
설명	특정 사용자 혹은 그룹에게 푸시 알림을 전송

📨 Request

```
{
  "userId": 42,
  "type": "GROUP_INVITE",
  "title": "모임 초대 알림",
  "body": "'러너스 클럽'에서 당신을 초대했습니다.",
  "data": {
    "groupId": 12,
    "inviteId": 88
  }
}
```

Key	설명
`userId`	수신자 사용자 ID
`type`	알림 타입 (`FRIEND_REQUEST` , `GROUP_INVITE` , `WORKOUT_ALERT` , ...)
`title`	푸시 제목
`body`	푸시 내용
`data`	추가 데이터 payload (클릭 시 특정 화면으로 이동하는 데 사용)

📩 Response (성공)

```
{
  "status": "success",
  "message": "푸시 알림이 정상적으로 전송되었습니다.",
  "notificationId": 321
}
```

❌ 실패 (400 Bad Request)

```
{
  "status": "error",
  "code": "INVALID_FCM_TOKEN",
  "message": "등록된 FCM 토큰이 유효하지 않습니다."
}
```

■ 서버 내부 동작 플로우

1. 서버에서 `userId` 기준으로 등록된 `fcmToken` 조회
2. Firebase Admin SDK를 통해 푸시 메시지 전송
3. 푸시 전송 성공 시, 동일 알림을 DB에 저장
4. 사용자가 앱을 켜면 WebSocket을 통해 알림 동기화

■ 비고

- 토큰 만료 시 클라이언트가 재발급 및 `/fcm/token` 재등록 필요

- WebSocket + FCM 이중 알림 방지를 위해 `notificationId` 기반 중복 방지
- 백엔드에서 Firebase Admin SDK (`firebase-admin`) 사용

5. 알림 읽음 / 삭제 API

■ 개요

사용자가 받은 알림(푸시 or 소켓 기반)을 확인하거나 삭제하는 기능.

앱에서 알림 탭을 눌러 상세 내용을 확인했을 때 서버에 읽음 처리를 요청하거나,

알림 목록에서 특정 알림을 제거할 수 있다.

- **인증 필요:** 서버 JWT Access Token
- **알림 상태 관리:** `unread`, `read`, `deleted`

■ 요청 정보 (1) 알림 읽음 처리

항목	내용
URL	`/api/notifications/{notificationId}/read`
Method	`PATCH`
Auth	Bearer Token (Access Token)
설명	특정 알림을 읽음(`read`) 상태로 변경

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	인증 토큰
`Content-Type`	`application/json`	JSON 요청

📩 Request Example

```
{
  "notificationId": 321
}
```

📩 Response (성공 – 200 OK)

```
{
  "code": "SUCCESS",
  "message": "알림을 읽음 상태로 변경했습니다.",
  "data": {
    "notificationId": 321,
    "status": "read",
    "readAt": "2025-10-15T12:03:21Z"
  }
}
```


❌ 실패 (404 Not Found)


```
{
  "code": "NOTIFICATION_NOT_FOUND",
  "message": "존재하지 않는 알림입니다."
}
```


■ 요청 정보 (2) 알림 삭제

항목	내용
URL	`/api/notifications/{notificationId}`
Method	`DELETE`
Auth	Bearer Token (Access Token)
설명	특정 알림을 삭제(`deleted` 상태로 전환 또는 DB soft delete)

📩 Request Example


```
{
  "notificationId": 321
}
```


📤 Response (성공 – 200 OK)

```
{
  "code": "SUCCESS",
  "message": "알림이 삭제되었습니다.",
  "data": {
    "notificationId": 321,
    "status": "deleted"
  }
}
```

❌ 실패 (403 Forbidden)

```
{
  "code": "FORBIDDEN",
  "message": "본인 알림만 삭제할 수 있습니다."
}
```

▼ Character

1. 캐릭터 생성 API

■ 개요

회원가입을 완료한 사용자가 자신의 아바타 캐릭터를 생성하는 API.

테마(`themeId`), 색상 팔레트(`palette`), 바디 타입 등 선택 요소를 기반으로

에셋(이미지, 모델링 리소스)을 매핑하여 서버에 캐릭터 정보를 저장한다.

- 인증 필요: **서버 JWT Access Token**
- 캐릭터는 1인 1개 기본 생성 (이후 커스터마이징 가능)
- 생성 시 반환되는 `characterId`는 이후 꾸미기·인벤토리 등 API에서 참조

■ 요청 정보

- **URL**: `/api/character`
- **Method**: `POST`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
`Content-Type`	`application/json`	JSON 요청

요청 바디

```
{
  "themeId": "forest_theme",
  "bodyType": "fit",
  "palette": {
    "skinColor": "#F4D4B8",
    "hairColor": "#3A2E2E",
    "outfitColor": "#87C5FF"
  }
}
```

필드	타입	필수	설명
`themeId`	string	✅	캐릭터 테마 (예: forest, city 등)
`bodyType`	string	✅	캐릭터 체형 타입
`palette`	object	✅	색상 팔레트 (피부, 머리, 옷 등)

■ 응답 정보

✅ 성공 (201 Created)

```
{
  "code": "SUCCESS",
  "message": "캐릭터 생성 완료",
  "data": {
    "characterId": 1204,
    "themeId": "forest_theme",
    "palette": {
      "skinColor": "#F4D4B8",
      "hairColor": "#3A2E2E",
      "outfitColor": "#87C5FF"
```

```
    },
    "createdAt": "2025-10-15T14:20:00Z"
  }
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "INVALID_THEME",
  "message": "존재하지 않는 테마 ID입니다."
}
```

❌ 실패 (409 Conflict)

```
{
  "code": "CHARACTER_ALREADY_EXISTS",
  "message": "해당 사용자는 이미 캐릭터를 보유하고 있습니다."
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```

2. 캐릭터 상태 업데이트 API

■ 개요

사용자의 건강 데이터(운동량, 수면 시간 등)를 기반으로

캐릭터의 **상태(state)** 및 **표정(mood)** 을 자동 갱신하는 API.

서버는 삼성 헬스 등에서 전달받은 데이터를 기준으로

캐릭터의 “컨디션”이나 “기분”을 계산하고

이 상태에 맞는 표정·포즈를 실시간 반영한다.

- 인증 필요: **서버 JWT Access Token**
- 상태 자동 계산: `steps`, `workoutMin`, `sleepHrs` 등
- 결과로 `state`, `mood`, `updatedAt` 반환

■ 요청 정보

- **URL**: `/api/character/state`
- **Method**: `PATCH`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
`Content-Type`	`application/json`	JSON 요청

요청 바디

```
{
  "steps": 8250,
  "workoutMin": 45,
  "sleepHrs": 6.5
}
```

필드	타입	필수	설명
`steps`	number	✅	일일 걸음 수
`workoutMin`	number	✅	운동한 시간(분)
`sleepHrs`	number	✅	수면 시간(시간 단위)

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "캐릭터 상태가 업데이트되었습니다.",
  "data": {
    "characterId": 1204,
    "state": "active",
    "mood": "energetic",
    "updatedAt": "2025-10-15T14:45:00Z"
  }
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "INVALID_DATA",
  "message": "입력된 건강 데이터 형식이 올바르지 않습니다."
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```

❌ 실패 (404 Not Found)

```
{
  "code": "CHARACTER_NOT_FOUND",
  "message": "해당 사용자의 캐릭터를 찾을 수 없습니다."
}
```

3. 캐릭터 AI 대사 생성 API

■ 개요

캐릭터의 현재 상태(`state`, `mood`)를 기반으로

AI가 자연스러운 대사를 생성하고 저장하는 API.

- 인증 필요: **서버 JWT Access Token**
- 생성 트리거:
  - 하루 1회 (매일 아침 자동 생성)
  - 또는 상태 변경 시 (`state`, `mood` 업데이트 이후)
- 생성된 대사는 캐릭터 화면 및 피드에서 표시됨

■ 요청 정보

- **URL**: `/api/character/utterance`
- **Method**: `POST`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
`Content-Type`	`application/json`	JSON 요청

요청 바디

```
{
  "characterId": 1204,
  "snapshot": {
    "state": "active",
    "mood": "energetic"
  }
}
```

필드	타입	필수	설명
`characterId`	number	✅	캐릭터 고유 ID
`snapshot`	object	✅	캐릭터의 현재 상태 정보

필드	타입	필수	설명
`snapshot.state`	string	✅	상태 (예: `active`, `tired`)
`snapshot.mood`	string	✅	기분 (예: `happy`, `sad`, `energetic`)

■ 응답 정보

✅ 성공 (201 Created)

```
{
  "code": "SUCCESS",
  "message": "AI 대사 생성 완료",
  "data": {
    "utteranceId": 3302,
    "characterId": 1204,
    "utterance": "오늘은 몸이 가볍네! 산책이라도 나가볼까?",
    "state": "active",
    "mood": "energetic",
    "createdAt": "2025-10-15T09:00:00Z"
  }
}
```

❌ 실패 (429 Too Many Requests)

```
{
  "code": "UTTERANCE_LIMIT_EXCEEDED",
  "message": "오늘은 이미 대사를 생성했습니다. 내일 다시 시도해주세요."
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```

❌ 실패 (404 Not Found)

```
{
  "code": "CHARACTER_NOT_FOUND",
  "message": "해당 캐릭터를 찾을 수 없습니다."
}
```

4. 캐릭터 꾸미기(아이템 장착) API

■ 개요

사용자가 보유 중인 아이템을 이용해

캐릭터의 **의상**, **배경**, **악세서리** 등을 꾸미는 API.

- 인증 필요: **서버 JWT Access Token**
- 포인트 또는 미션 보상으로 획득한 아이템만 장착 가능
- 캐릭터의 외형(`appearance`) 정보 업데이트
- 각 카테고리(의상/배경/악세서리)는 1개씩만 장착 가능

■ 요청 정보

- **URL**: `/api/character/equip`
- **Method**: `PATCH`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
`Content-Type`	`application/json`	JSON 요청

요청 바디

```
{
  "characterId": 1204,
```

```
  "equipItems": {
    "outfit": 2101,
    "background": 3103,
    "accessory": 4105
  }
}
```

필드	타입	필수	설명
characterId	number	✅	캐릭터 고유 ID
equipItems	object	✅	장착할 아이템 목록
equipItems.outfit	number	선택	의상 아이템 ID
equipItems.background	number	선택	배경 아이템 ID
equipItems.accessory	number	선택	악세서리 아이템 ID

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "캐릭터 꾸미기가 완료되었습니다.",
  "data": {
    "characterId": 1204,
    "equipped": {
      "outfit": {
        "itemId": 2101,
        "name": "트레이닝 셋업"
      },
      "background": {
        "itemId": 3103,
        "name": "봄의 공원"
      },
      "accessory": {
        "itemId": 4105,
        "name": "스포츠 헤드밴드"
      }
    },
```

```
    "updatedAt": "2025-10-15T15:10:00Z"
  }
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "INVALID_ITEM",
  "message": "보유하지 않은 아이템을 장착할 수 없습니다."
}
```

❌ 실패 (404 Not Found)

```
{
  "code": "CHARACTER_NOT_FOUND",
  "message": "해당 캐릭터를 찾을 수 없습니다."
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```

5. 인벤토리 조회 API

■ 개요

사용자가 보유한 모든 아이템(의상, 배경, 악세서리 등)을 조회하는 API.

캐릭터 화면 진입 시 호출되며, 현재 장착 중인 아이템 정보도 함께 제공한다.

- 인증 필요: **서버 JWT Access Token**
- 아이템 유형별(`outfit`, `background`, `accessory`) 구분

- 포인트/미션 보상으로 획득한 아이템만 포함

■ 요청 정보

- **URL:** `/api/inventory`
- **Method:** `GET`
- **Content-Type:** `application/json`

요청 헤더

Key	Value	설명
`Authorization`	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
`Content-Type`	`application/json`	JSON 요청

요청 파라미터 (선택)

Key	Type	필수	설명
`category`	string	❌	필터링용 카테고리 (`outfit` , `background` , `accessory`)

요청 예시

```
{}
```

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "인벤토리 조회 성공",
  "data": {
    "userId": 501,
    "inventory": [
      {
        "itemId": 2101,
```

```
      "name": "트레이닝 셋업",
      "category": "outfit",
      "rarity": "rare",
      "equipped": true,
      "obtainedAt": "2025-10-10T11:00:00Z"
    },
    {
      "itemId": 2102,
      "name": "러닝 후드티",
      "category": "outfit",
      "rarity": "normal",
      "equipped": false,
      "obtainedAt": "2025-10-12T08:30:00Z"
    }
  ]
 }
}
```


❌ 실패 (401 Unauthorized)


```
{
 "code": "INVALID_TOKEN",
 "message": "유효하지 않은 인증 토큰입니다."
}
```


❌ 실패 (404 Not Found)


```
{
 "code": "INVENTORY_NOT_FOUND",
 "message": "해당 사용자의 인벤토리를 찾을 수 없습니다."
}
```


▼ Dashboard

1️⃣ 주간 운동 그래프 조회

■ 개요

사용자의 일주일간 운동 데이터를 시각화용으로 조회한다.

운동 시간, 종류, 걸음수를 날짜별로 반환하며, 클라이언트에서 그래프 렌더링에 활용된다.

- 인증 필요: 서버 JWT Access Token
- 데이터 집계 기준: 월요일~일요일

■ 요청 정보

- **URL**: `/api/dashboard/weekly-graph`
- **Method**: `GET`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
Authorization	`Bearer <JWT Access Token>`	서버 로그인 토큰
Content-Type	`application/json`	JSON 요청

요청 파라미터 (Query)

Key	Type	필수	설명
startDate	string	❌	조회 시작일 (`yyyy-MM-dd`) / 미입력 시 이번주 기준

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "주간 운동 그래프 조회 성공",
  "data": {
    "startDate": "2025-10-06",
    "endDate": "2025-10-12",
    "dailyStats": [
      {
        "date": "2025-10-06",
```

```
      "workoutMinutes": 45,
      "steps": 7200,
      "workoutType": "cardio"
    },
    {
      "date": "2025-10-07",
      "workoutMinutes": 30,
      "steps": 5400,
      "workoutType": "yoga"
    }
  ]
 }
}
```

❌ 실패 (401 Unauthorized)

```
{
 "code": "INVALID_TOKEN",
 "message": "유효하지 않은 토큰입니다."
}
```

2️⃣ 주간 리포트 조회 (전주 대비)

■ 개요

사용자의 이번 주 운동 데이터를 전주와 비교하여

증감률(`delta`) 및 **목표 달성률(`goalRate`)**을 제공하는 API.

대시보드의 주간 리포트 카드나 통계 섹션에서 사용된다.

- 인증 필요: 서버 JWT Access Token
- 비교 기준: 최근 2주간(이번 주 vs 전주)

■ 요청 정보

- **URL**: `/api/dashboard/weekly-report`
- **Method**: `GET`

- **Content-Type:** `application/json`

요청 헤더

Key	Value	설명
Authorization	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
Content-Type	`application/json`	JSON 요청

요청 파라미터 (Query)

Key	Type	필수	설명
startDate	string	❌	이번 주 시작일 (`yyyy-MM-dd`) — 미입력 시 자동 계산

◼ 응답 정보

✅ 성공 (200 OK)


```
{
  "code": "SUCCESS",
  "message": "주간 리포트 조회 성공",
  "data": {
    "weekRange": {
      "current": "2025-10-06 ~ 2025-10-12",
      "previous": "2025-09-29 ~ 2025-10-05"
    },
    "summary": {
      "totalWorkoutMinutes": 310,
      "totalSteps": 45000,
      "goalRate": 92
    },
    "delta": {
      "workoutMinutes": "+12%",
      "steps": "-4%",
      "goalRate": "+5%"
    }
  }
```

```
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```

❌ 실패 (404 Not Found)

```
{
  "code": "NO_DATA",
  "message": "조회 가능한 주간 데이터가 없습니다."
}
```

3️⃣ 월간 리포트 조회 (AI 코멘트)

■ 개요

사용자의 한 달간 패턴을 집계하고, AI가 분석한 행동 제안을 제공하는 API.

- 매월 말 자동 생성되며, 수동 조회 가능
- 반환 데이터: 월간 운동/수면/수분 통계, 요약, AI 코멘트
- 인증 필요: 서버 JWT Access Token

■ 요청 정보

- **URL**: `/api/dashboard/monthly-report`
- **Method**: `GET`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
Authorization	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
Content-Type	`application/json`	JSON 요청

요청 파라미터 (Query)

Key	Type	필수	설명
month	string	❌	조회할 월 (`yyyy-MM`) — 미입력 시 현재 월 기준

■ 응답 정보

✅ 성공 (200 OK)


```
{
  "code": "SUCCESS",
  "message": "월간 리포트 조회 성공",
  "data": {
    "month": "2025-09",
    "totalWorkoutMinutes": 1420,
    "averageSteps": 7100,
    "totalSleepHours": 168,
    "patternSummary": "평균보다 운동 빈도가 일정하며, 수면시간은 주중 감소 경향.",
    "aiComment": "운동 리듬은 좋습니다. 주중 수면량을 30분만 늘리면 피로도가 개선될 것으로 예상됩니다."
  }
}
```


❌ 실패 (401 Unauthorized)


```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```

❌ 실패 (404 Not Found)

```
{
  "code": "NO_MONTHLY_DATA",
  "message": "해당 월의 데이터가 없습니다."
}
```

4️⃣ 목표 설정 / 조회 API

■ 개요

사용자의 운동, 수면, 수분 등 **주간 목표**를 설정하고 조회하는 API.

- 목표 설정 시 서버에 저장되며, 진행률 계산에 활용
- 조회 시 현재 진행률(`progressRate`)도 반환
- 인증 필요: 서버 JWT Access Token

■ 요청 정보

▶ 목표 조회

- **URL**: `/api/dashboard/goals`
- **Method**: `GET`
- **Content-Type**: `application/json`

▶ 목표 저장 / 수정

- **URL**: `/api/dashboard/goals`
- **Method**: `POST`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
Authorization	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
Content-Type	`application/json`	JSON 요청

요청 바디 예시 (목표 저장/수정)

```
{
  "workoutMinutesGoal": 300,
  "sleepHoursGoal": 56,
  "waterIntakeGoal": 14000
}
```

필드	타입	필수	설명
workoutMinutesGoal	number	✅	주간 운동 목표(분)
sleepHoursGoal	number	✅	주간 수면 목표(시간)
waterIntakeGoal	number	✅	주간 수분 섭취 목표(ml)

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "목표 조회/저장 성공",
  "data": {
    "workoutMinutesGoal": 300,
    "sleepHoursGoal": 56,
    "waterIntakeGoal": 14000,
    "progressRate": {
      "workoutMinutes": 74.2,
      "sleepHours": 90.5,
      "waterIntake": 60.0
    }
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "INVALID_GOAL_VALUE",
  "message": "목표 값이 유효하지 않습니다."
}
```

5️⃣ 리마인더 푸시 API

■ 개요

사용자의 목표 대비 **진행률이 낮은 경우** 넛지 알림을 전송하는 API.

- 알림 유형: 운동, 수면, 수분 등 목표별
- 주기: 목표 미달 시, 서버 스케줄러 또는 클라이언트 요청으로 트리거 가능
- 인증 필요: 서버 JWT Access Token

■ 요청 정보

- **URL**: `/api/dashboard/reminder/push`
- **Method**: `POST`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
Authorization	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
Content-Type	`application/json`	JSON 요청

요청 바디

```
{
  "userId": 501,
  "type": "workout",
  "progressRate": 42.0
}
```

필드	타입	필수	설명
userId	number	✅	사용자 고유 ID
type	string	✅	목표 유형 (`workout` , `sleep` , `water`)
progressRate	number	✅	현재 진행률 (%)

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "리마인더 푸시 전송 완료",
  "data": {
    "sentAt": "2025-10-15T06:00:00Z",
    "reminderType": "workout",
    "content": "이번 주 운동이 목표의 40%에 그쳤어요! 오늘 20분만 더 움직여볼까요?"
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "INVALID_PROGRESS_RATE",
  "message": "진행률 값이 유효하지 않습니다."
}
```

▼ Mission

1️⃣ 협동 미션 목표 생성 API

■ 개요

그룹 평균 기반으로 과훈련을 방지하면서 사용자의 주간 미션 목표를 제안하고 생성하는 API.

- 인증 필요: 서버 JWT Access Token
- 그룹 단위 주간 미션 생성
- 목표값(`missionGoal`)은 그룹 평균과 과거 기록을 참고하여 자동 계산

■ 요청 정보

- **URL**: `/api/mission/create`
- **Method**: `POST`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
Authorization	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
Content-Type	`application/json`	JSON 요청

요청 바디

```
{
  "groupId": 101,
  "missionType": "workout",
  "weekStartDate": "2025-10-13"
```

```
}
```

필드	타입	필수	설명
groupId	number	✅	그룹 고유 ID
missionType	string	✅	미션 유형 (`workout` , `steps` , `sleep`)
weekStartDate	string	✅	주간 미션 시작일 (`yyyy-MM-dd`)

■ 응답 정보

✅ 성공 (201 Created)

```
{
  "code": "SUCCESS",
  "message": "협동 미션 목표 생성 완료",
  "data": {
    "missionId": 501,
    "groupId": 101,
    "missionType": "workout",
    "missionGoal": 250,
    "weekStartDate": "2025-10-13",
    "createdAt": "2025-10-13T08:00:00Z"
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "MISSION_ALREADY_EXISTS",
  "message": "이번 주 이미 미션이 생성되어 있습니다."
}
```

2️⃣ 협동 미션 진행 현황 조회 API

■ 개요

그룹 미션의 실시간 달성률과 개인 기여도를 확인하는 API.

- 인증 필요: 서버 JWT Access Token
- 팀 단위 목표 달성률과 사용자 개인 기여도 계산
- 실시간/일일 업데이트 가능

■ 요청 정보

- **URL**: `/api/mission/{missionId}/progress`
- **Method**: `GET`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
Authorization	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
Content-Type	`application/json`	JSON 요청

요청 경로 파라미터

Key	Type	필수	설명
missionId	number	✅	조회할 미션 ID

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "협동 미션 진행 현황 조회 성공",
  "data": {
    "missionId": 501,
    "groupId": 101,
    "missionType": "workout",
    "missionGoal": 250,
    "progressRate": 72.5,
    "participants": [
      {
        "userId": 1001,
        "username": "userA",
        "contributionRate": 30.0
      },
      {
        "userId": 1002,
        "username": "userB",
        "contributionRate": 42.5
      }
    ],
    "updatedAt": "2025-10-15T10:00:00Z"
  }
}
```


❌ 실패 (401 Unauthorized)


```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```


❌ 실패 (404 Not Found)


```
{
  "code": "MISSION_NOT_FOUND",
```

```
  "message": "해당 미션을 찾을 수 없습니다."
}
```

3️⃣ 협동 미션 보상 지급 API

■ 개요

사용자가 미션 목표를 달성했을 때 **포인트/배지**를 자동 지급하는 API.

- 인증 필요: 서버 JWT Access Token
- 중복 지급 방지
- 목표 달성 여부 확인 후 지급

■ 요청 정보

- **URL**: `/api/mission/{missionId}/reward`
- **Method**: `POST`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
Authorization	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
Content-Type	`application/json`	JSON 요청

요청 경로 파라미터

Key	Type	필수	설명
missionId	number	✅	지급 대상 미션 ID

요청 바디

```
{
  "userId": 1001
}
```

필드	타입	필수	설명
userId	number	✅	보상 지급 대상 사용자 ID

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "미션 보상 지급 완료",
  "data": {
    "missionId": 501,
    "userId": 1001,
    "pointsAwarded": 100,
    "badgeAwarded": "Mission Achiever",
    "awardedAt": "2025-10-15T12:00:00Z"
  }
}
```

❌ 실패 (401 Unauthorized)

```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```

❌ 실패 (400 Bad Request)

```
{
  "code": "REWARD_ALREADY_GRANTED",
  "message": "이미 보상이 지급된 미션입니다."
}
```

4️⃣ 협동 미션 히스토리 조회 API

■ 개요

사용자가 참여한 **지난 미션 기록**을 조회하는 API.

- 회고 및 성장 추적용
- 주간 종료 후 데이터 제공
- 인증 필요: 서버 JWT Access Token

■ 요청 정보

- **URL**: `/api/mission/history`
- **Method**: `GET`
- **Content-Type**: `application/json`

요청 헤더

Key	Value	설명
Authorization	`Bearer <JWT Access Token>`	로그인된 사용자 토큰
Content-Type	`application/json`	JSON 요청

요청 파라미터 (Query)

Key	Type	필수	설명
groupId	number	❌	특정 그룹의 미션 히스토리 필터링
month	string	❌	조회 월 (`yyyy-MM`) / 미입력 시 전체 기록

■ 응답 정보

✅ 성공 (200 OK)

```
{
  "code": "SUCCESS",
  "message": "협동 미션 히스토리 조회 성공",
  "data": [
    {
      "missionId": 501,
```

```
      "groupId": 101,
      "missionType": "workout",
      "missionGoal": 250,
      "progressRate": 95.0,
      "pointsAwarded": 100,
      "badgeAwarded": "Mission Achiever",
      "weekRange": "2025-10-06 ~ 2025-10-12",
      "completedAt": "2025-10-12T23:59:59Z"
    },
    {
      "missionId": 502,
      "groupId": 101,
      "missionType": "steps",
      "missionGoal": 50000,
      "progressRate": 88.0,
      "pointsAwarded": 50,
      "badgeAwarded": null,
      "weekRange": "2025-09-29 ~ 2025-10-05",
      "completedAt": "2025-10-05T23:59:59Z"
    }
  ]
}
```


❌ 실패 (401 Unauthorized)


```
{
  "code": "INVALID_TOKEN",
  "message": "유효하지 않은 인증 토큰입니다."
}
```


❌ 실패 (404 Not Found)


```
{
  "code": "NO_HISTORY",
  "message": "조회 가능한 미션 기록이 없습니다."
}
```