TRD (Technical Requirement Document)

WorkFlow - 기술 요구사항 명세서

버전: 1.0
작성일: 2026-02-02
작성자: Engineering Team
문서 상태: Draft

목차

---

1. 문서 개요

1.1 문서 목적

본 문서는 WorkFlow 시스템의 기술적 요구사항, 아키텍처, 제약 조건을 정의하여 개발팀이 일관된 기술 결정을 내리고 안정적인 시스템을 구축할 수 있도록 한다.

1.2 대상 독자

- 백엔드/프론트엔드 개발자
- DevOps 엔지니어
- QA 엔지니어
- 기술 아키텍트

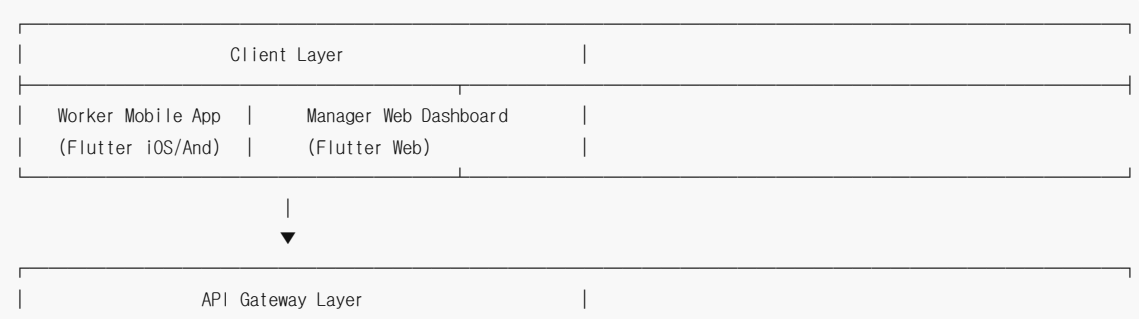
1.3 참조 문서

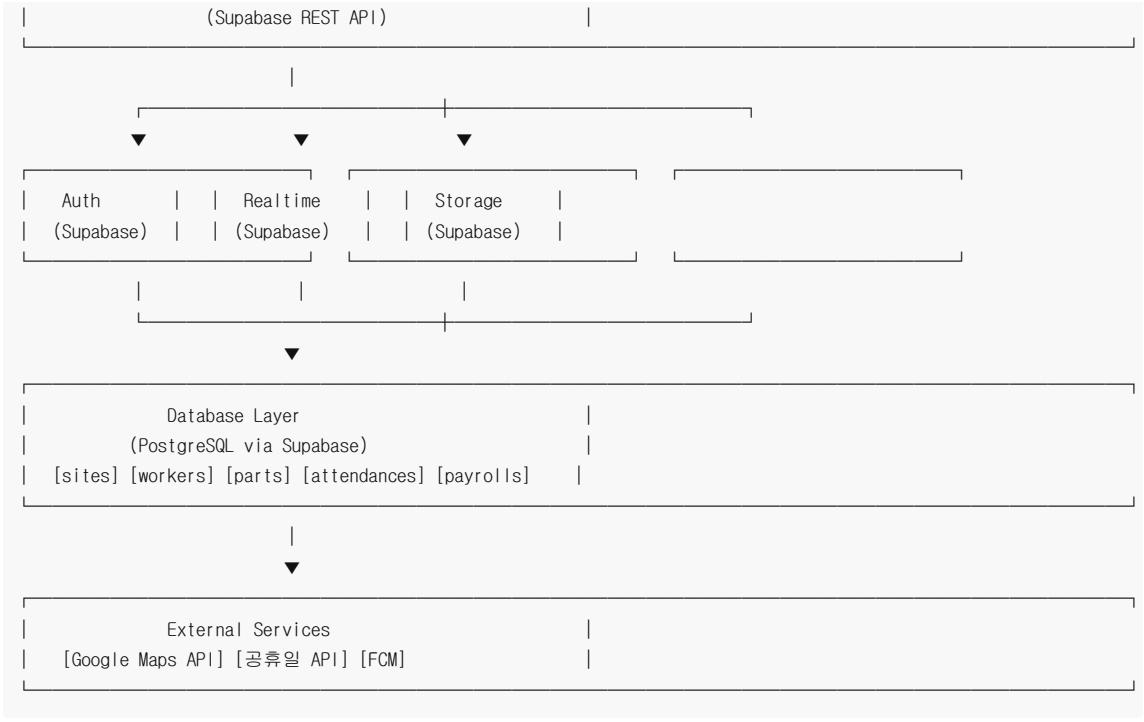
- PRD (Product Requirement Document)
- Flutter 공식 문서
- Supabase 공식 문서

---

2. 시스템 아키텍처

2.1 전체 아키텍처





2.2 데이터 플로우

2.2.1 출퇴근 체크인 플로우

```
1. 근로자 앱에서 "출근" 버튼 탭
2. GPS 위치 수집 (Geolocator 패키지)
3. 사업장 GPS 좌표와 거리 계산
4. 거리 검증 (100m 이내?)
   - 성공: Supabase DB에 출근 기록 저장
   - 실패: 에러 메시지 표시
5. 실시간 업데이트 (Realtime)
6. 관리자 웹 대시보드 자동 갱신
```

2.2.2 급여대장 생성 플로우

```
1. 관리자 웹에서 "급여대장 생성" 버튼 클릭
2. Supabase Edge Function 호출
3. 해당 월의 출퇴근 기록 조회 (PostgreSQL)
4. 근로자별 집계:
   - 총 근무시간 = SUM(퇴근시간 - 출근시간)
   - 파트별 시급 조회
   - 급여 계산 = 시급 × 근무시간
5. 결과를 JSON으로 반환
6. 클라이언트에서 엑셀 파일 생성 (excel 패키지)
7. 다운로드
```

2.3 아키텍처 패턴

Frontend (Flutter):

- **Clean Architecture** 적용
  - Presentation Layer: UI (Widgets)
  - Domain Layer: Business Logic (Use Cases)
  - Data Layer: Repository, Data Sources

Backend (Supabase):

- **Serverless Architecture**
  - Edge Functions for 복잡한 비즈니스 로직
  - Database RLS (Row Level Security) for 데이터 접근 제어
  - Realtime Subscriptions for 실시간 업데이트

---

3. 기술 스택

3.1 Frontend

3.1.1 Core Framework

기술	버전	용도	선택 이유
Flutter	3.19+	UI Framework	단일 코드베이스로 Web/iOS/Android 지원
Dart	3.3+	프로그래밍 언어	Flutter 공식 언어, 강타입 지원

3.1.2 State Management

기술	버전	용도	선택 이유
Riverpod	2.5+	상태 관리	테스트 용이, 확장성 우수

3.1.3 주요 패키지

패키지	버전	용도
supabase_flutter	2.5+	Supabase 클라이언트
geolocator	11.0+	GPS 위치 수집
permission_handler	11.3+	권한 관리
excel	4.0+	엑셀 파일 생성
intl	0.19+	날짜/시간 포맷팅
freezed	2.5+	불변 모델 생성
json_serializable	6.8+	JSON 직렬화

3.2 Backend

3.2.1 Core Infrastructure

기술	용도	선택 이유
Supabase	통합 백엔드 플랫폼	빠른 개발, 실시간 기능, 인증 내장
PostgreSQL	관계형 데이터베이스	강력한 쿼리 성능, 트랜잭션 지원
PostgREST	자동 REST API	DB 스키마에서 API 자동 생성

3.2.2 Supabase Features

기능	용도
Auth	전화번호/이메일 인증
Database	PostgreSQL 호스팅

Realtime	WebSocket 실시간 업데이트
Storage	파일 저장 (향후 확장)
Edge Functions	서버리스 함수 (Deno)
Row Level Security	행 단위 접근 제어

3.3 External APIs

API	용도	비용
Google Maps Geolocation API	GPS 좌표 검증	무료 (월 40,000건)
한국천문연구원 공휴일 API	공휴일 조회	무료
Firebase Cloud Messaging	푸시 알림 (Post-MVP)	무료

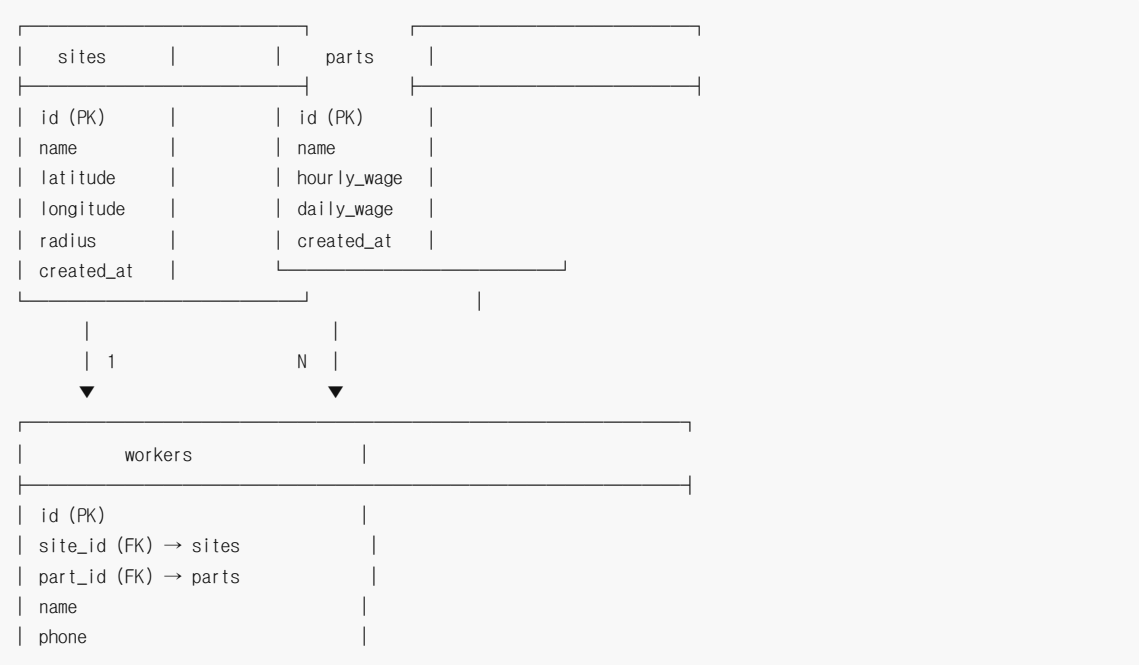
3.4 DevOps & Tools

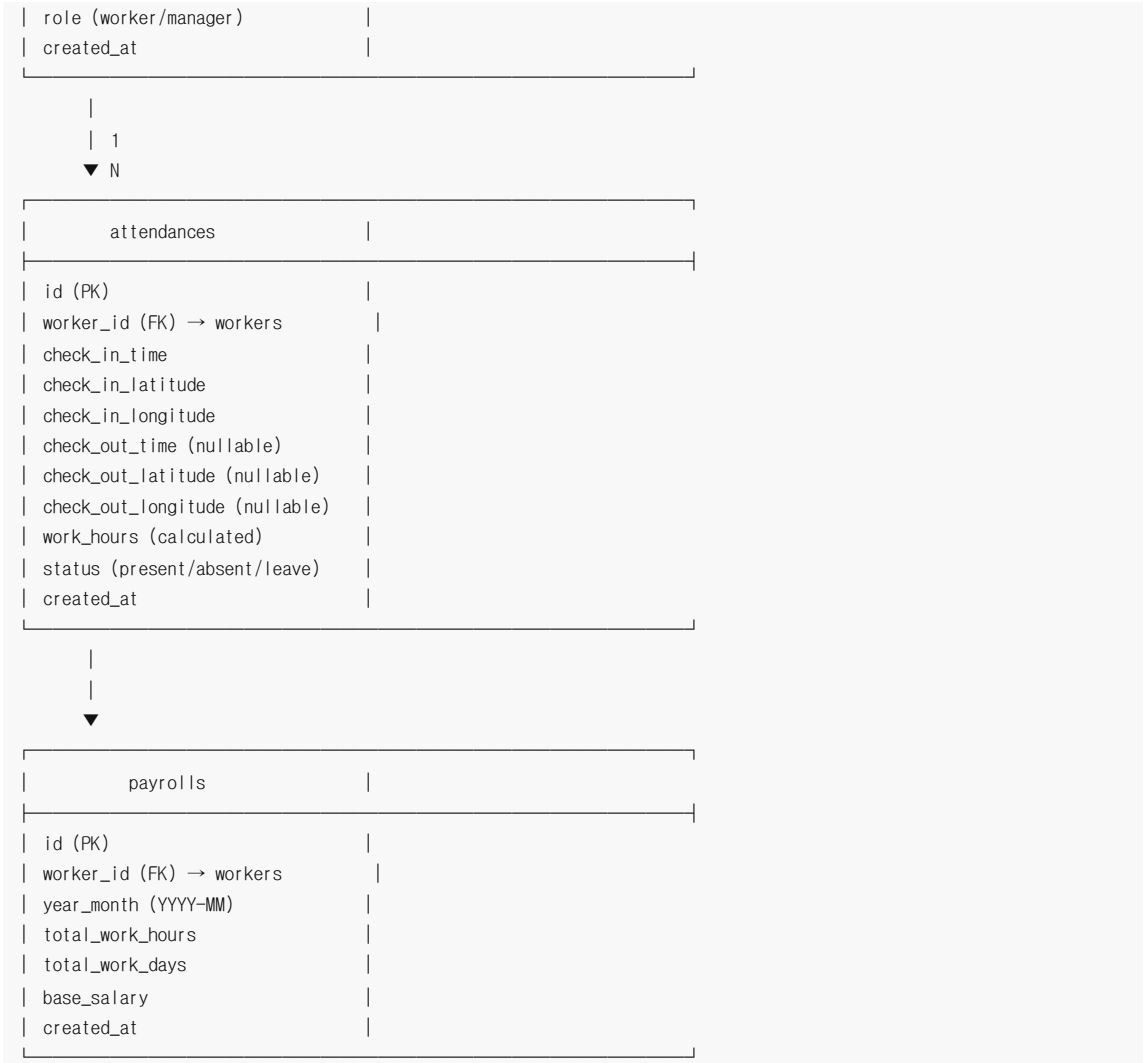
도구	용도
GitHub	버전 관리
GitHub Actions	CI/CD
Vercel	Web 호스팅
App Store Connect	iOS 배포
Google Play Console	Android 배포
Sentry (Optional)	에러 트래킹

---

4. 데이터베이스 설계

4.1 ERD (Entity Relationship Diagram)





4.2 테이블 스키마

4.2.1 sites (사업장)

```sql
CREATE TABLE sites (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name VARCHAR(255) NOT NULL,
  latitude DECIMAL(10, 8) NOT NULL,
  longitude DECIMAL(11, 8) NOT NULL,
  radius INTEGER DEFAULT 100, -- 미터 단위
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 인덱스
CREATE INDEX idx_sites_name ON sites(name);
```

4.2.2 parts (파트/직급)

```sql
CREATE TABLE parts (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
```

```
  name VARCHAR(100) NOT NULL UNIQUE,
  hourly_wage INTEGER NOT NULL, -- 시급 (원)
  daily_wage INTEGER, -- 일급 (원, nullable)
  description TEXT,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 인덱스
CREATE INDEX idx_parts_name ON parts(name);
```

4.2.3 workers (근로자)

```
CREATE TABLE workers (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  site_id UUID REFERENCES sites(id) ON DELETE CASCADE,
  part_id UUID REFERENCES parts(id) ON DELETE SET NULL,
  name VARCHAR(100) NOT NULL,
  phone VARCHAR(20) UNIQUE NOT NULL,
  role VARCHAR(20) CHECK (role IN ('worker', 'manager')) DEFAULT 'worker',
  is_active BOOLEAN DEFAULT TRUE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 인덱스
CREATE INDEX idx_workers_site_id ON workers(site_id);
CREATE INDEX idx_workers_part_id ON workers(part_id);
CREATE INDEX idx_workers_phone ON workers(phone);
CREATE INDEX idx_workers_role ON workers(role);
```

4.2.4 attendances (출퇴근 기록)

```
CREATE TABLE attendances (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  worker_id UUID REFERENCES workers(id) ON DELETE CASCADE,

  -- 출근 정보
  check_in_time TIMESTAMP WITH TIME ZONE NOT NULL,
  check_in_latitude DECIMAL(10, 8) NOT NULL,
  check_in_longitude DECIMAL(11, 8) NOT NULL,

  -- 퇴근 정보 (nullable)
  check_out_time TIMESTAMP WITH TIME ZONE,
  check_out_latitude DECIMAL(10, 8),
  check_out_longitude DECIMAL(11, 8),

  -- 계산 필드
  work_hours DECIMAL(5, 2), -- 근무시간 (시간 단위)

  -- 근태 상태
  status VARCHAR(20) CHECK (status IN ('present', 'absent', 'leave', 'holiday'))
    DEFAULT 'present',

  -- 메타데이터
  notes TEXT,
```

```
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- 인덱스
CREATE INDEX idx_attendances_worker_id ON attendances(worker_id);
CREATE INDEX idx_attendances_check_in_time ON attendances(check_in_time);
CREATE INDEX idx_attendances_status ON attendances(status);

-- 복합 인덱스 (급여 계산용)
CREATE INDEX idx_attendances_worker_month
  ON attendances(worker_id, EXTRACT(YEAR FROM check_in_time), EXTRACT(MONTH FROM check_in_time));
```

4.2.5 payrolls (급여대장)

```
CREATE TABLE payrolls (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  worker_id UUID REFERENCES workers(id) ON DELETE CASCADE,
  year_month VARCHAR(7) NOT NULL, -- YYYY-MM 형식

  -- 집계 데이터
  total_work_hours DECIMAL(8, 2) NOT NULL,
  total_work_days INTEGER NOT NULL,

  -- 급여 계산
  base_salary INTEGER NOT NULL, -- 기본급
  overtime_pay INTEGER DEFAULT 0, -- 연장수당 (Post-MVP)
  holiday_pay INTEGER DEFAULT 0, -- 공휴일수당 (Post-MVP)
  total_salary INTEGER NOT NULL, -- 총 급여

  -- 메타데이터
  is_finalized BOOLEAN DEFAULT FALSE,
  finalized_at TIMESTAMP WITH TIME ZONE,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),

  -- 유니크 제약 (한 근로자당 한 달에 하나)
  UNIQUE(worker_id, year_month)
);

-- 인덱스
CREATE INDEX idx_payrolls_worker_id ON payrolls(worker_id);
CREATE INDEX idx_payrolls_year_month ON payrolls(year_month);
CREATE INDEX idx_payrolls_is_finalized ON payrolls(is_finalized);
```

4.3 Database Functions

4.3.1 근무시간 자동 계산 (Trigger)

```
CREATE OR REPLACE FUNCTION calculate_work_hours()
RETURNS TRIGGER AS $$
BEGIN
  IF NEW.check_out_time IS NOT NULL THEN
    NEW.work_hours := EXTRACT(EPOCH FROM (NEW.check_out_time - NEW.check_in_time)) / 3600.0;
  END IF;
  RETURN NEW;
END;
```

```
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_work_hours
BEFORE INSERT OR UPDATE ON attendances
FOR EACH ROW
EXECUTE FUNCTION calculate_work_hours();
```

4.3.2 updated_at 자동 갱신 (Trigger)

```
CREATE OR REPLACE FUNCTION update_updated_at()
RETURNS TRIGGER AS $$
BEGIN
  NEW.updated_at = NOW();
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- 모든 테이블에 적용
CREATE TRIGGER update_sites_updated_at BEFORE UPDATE ON sites
  FOR EACH ROW EXECUTE FUNCTION update_updated_at();

CREATE TRIGGER update_parts_updated_at BEFORE UPDATE ON parts
  FOR EACH ROW EXECUTE FUNCTION update_updated_at();

CREATE TRIGGER update_workers_updated_at BEFORE UPDATE ON workers
  FOR EACH ROW EXECUTE FUNCTION update_updated_at();

CREATE TRIGGER update_attendances_updated_at BEFORE UPDATE ON attendances
  FOR EACH ROW EXECUTE FUNCTION update_updated_at();

CREATE TRIGGER update_payrolls_updated_at BEFORE UPDATE ON payrolls
  FOR EACH ROW EXECUTE FUNCTION update_updated_at();
```

---

5. API 명세

5.1 인증 API

5.1.1 근로자 로그인 (SMS 인증)

```
POST /auth/v1/otp
Request:
{
  "phone": "+821012345678"
}

Response:
{
  "message": "OTP sent"
}
```

5.1.2 OTP 검증

```
POST /auth/v1/verify
Request:
{
  "phone": "+821012345678",
```

```
  "token": "123456"
}

Response:
{
  "access_token": "eyJhbGc...",
  "refresh_token": "...",
  "user": { ... }
}
```

5.1.3 관리자 로그인

```
POST /auth/v1/token?grant_type=password
Request:
{
  "email": "manager@example.com",
  "password": "securepassword"
}

Response:
{
  "access_token": "eyJhbGc...",
  "refresh_token": "..."
}
```

5.2 출퇴근 API

5.2.1 출근 체크인

```
POST /rest/v1/attendances
Headers:
  Authorization: Bearer {token}

Request:
{
  "worker_id": "uuid",
  "check_in_time": "2026-02-02T07:00:23+09:00",
  "check_in_latitude": 37.5665,
  "check_in_longitude": 126.9780
}

Response:
{
  "id": "uuid",
  "worker_id": "uuid",
  "check_in_time": "2026-02-02T07:00:23+09:00",
  "status": "present"
}
```

5.2.2 퇴근 체크아웃

```
PATCH /rest/v1/attendances?id=eq.{attendance_id}
Headers:
  Authorization: Bearer {token}

Request:
{
```

```
  "check_out_time": "2026-02-02T17:00:45+09:00",
  "check_out_latitude": 37.5665,
  "check_out_longitude": 126.9780
}

Response:
{
  "id": "uuid",
  "check_out_time": "2026-02-02T17:00:45+09:00",
  "work_hours": 10.0
}
```

5.3 근태 조회 API

5.3.1 오늘 출근 현황

```
GET /rest/v1/rpc/get_today_attendance?site_id={site_id}
Headers:
  Authorization: Bearer {token}

Response:
{
  "total_workers": 50,
  "checked_in": 47,
  "not_checked_in": [
    { "id": "uuid", "name": "김하역" },
    ...
  ]
}
```

5.3.2 근로자별 출퇴근 기록

```
GET /rest/v1/attendances?worker_id=eq.{worker_id}&order=check_in_time.desc
Headers:
  Authorization: Bearer {token}

Response:
[
  {
    "id": "uuid",
    "check_in_time": "2026-02-02T07:00:23+09:00",
    "check_out_time": "2026-02-02T17:00:45+09:00",
    "work_hours": 10.0
  },
  ...
]
```

5.4 급여 API

5.4.1 월간 급여 계산 (Edge Function)

```
POST /functions/v1/calculate_payroll
Headers:
  Authorization: Bearer {token}

Request:
{
  "site_id": "uuid",
```

```
  "year_month": "2026-01"
}

Response:
{
  "payrolls": [
    {
      "worker_id": "uuid",
      "worker_name": "김하역",
      "part_name": "일용직",
      "total_work_hours": 220.5,
      "total_work_days": 22,
      "base_salary": 3042900,
      "total_salary": 3042900
    },
    ...
  ]
}
```

5.4.2 급여대장 저장

```
POST /rest/v1/payrolls
Headers:
  Authorization: Bearer {token}

Request:
[
  {
    "worker_id": "uuid",
    "year_month": "2026-01",
    "total_work_hours": 220.5,
    "total_work_days": 22,
    "base_salary": 3042900,
    "total_salary": 3042900
  },
  ...
]

Response: 201 Created
```

---

6. 보안 요구사항

6.1 인증 및 권한

6.1.1 Row Level Security (RLS) 정책

workers 테이블:

```sql
-- 근로자는 자신의 정보만 조회 가능
CREATE POLICY worker_select_own
ON workers FOR SELECT
USING (auth.uid() = id);

-- 관리자는 같은 사업장의 근로자 조회 가능
CREATE POLICY manager_select_site
ON workers FOR SELECT
USING (
```

```
  role = 'manager' AND
  site_id IN (
    SELECT site_id FROM workers WHERE id = auth.uid()
  )
);

-- 관리자만 근로자 등록/수정/삭제
CREATE POLICY manager_modify
ON workers FOR ALL
USING (
  EXISTS (
    SELECT 1 FROM workers
    WHERE id = auth.uid() AND role = 'manager'
  )
);
```

attendances 테이블:

```
-- 근로자는 자신의 출퇴근 기록만 조회/생성
CREATE POLICY worker_own_attendance
ON attendances FOR SELECT
USING (worker_id = auth.uid());

CREATE POLICY worker_create_attendance
ON attendances FOR INSERT
WITH CHECK (worker_id = auth.uid());

-- 관리자는 같은 사업장의 모든 출퇴근 기록 조회
CREATE POLICY manager_view_site_attendance
ON attendances FOR SELECT
USING (
  worker_id IN (
    SELECT w1.id FROM workers w1
    JOIN workers w2 ON w1.site_id = w2.site_id
    WHERE w2.id = auth.uid() AND w2.role = 'manager'
  )
);
```

6.1.2 데이터 암호화

- **전송 중 암호화:** HTTPS/TLS 1.3
- **저장 시 암호화:** Supabase 기본 암호화 (AES-256)
- **민감 정보:** 전화번호는 해시 없이 평문 저장 (로그인 용도)

6.1.3 접근 제어

역할	권한
근로자	자신의 출퇴근 체크인/조회, 자신의 급여 조회
관리자	같은 사업장의 모든 데이터 조회/수정, 급여대장 생성
시스템 관리자	모든 데이터 접근 (Supabase 대시보드)

6.2 데이터 보호

6.2.1 개인정보 보호

- 최소한의 개인정보 수집 (이름, 전화번호만)
- 퇴사 시 개인정보 보관 기간: 3년 (근로기준법)

- 개인정보 처리방침 준수

6.2.2 GPS 데이터 보호

- GPS 좌표는 출퇴근 검증 목적으로만 사용
- 실시간 위치 추적 불가 (출퇴근 시점만 기록)
- 좌표 데이터는 관리자만 조회 가능

6.3 취약점 대응

6.3.1 GPS 스푸핑 방지

- GPS 정확도 검증 (accuracy < 50m)
- 이상 패턴 탐지 (같은 위치에서 너무 빠른 체크인)
- 관리자 수동 검증 옵션

6.3.2 SQL Injection 방지

- Supabase의 자동 파라미터 바인딩
- ORM 사용 (직접 쿼리 작성 최소화)

6.3.3 DDoS 방어

- Supabase의 기본 Rate Limiting
- API 호출 제한: 100 req/min per user

---

7. 성능 요구사항

7.1 응답 시간

작업	목표 응답 시간	허용 한계
출퇴근 체크인	< 1초	< 2초
대시보드 로딩	< 2초	< 3초
급여대장 생성 (50명)	< 5초	< 10초
엑셀 다운로드	< 3초	< 5초

7.2 동시 사용자 지원

- **피크 시간:** 오전 7-9시, 오후 5-7시 (출퇴근 시간)
- **동시 접속:** 최대 250명 (5개 사업장 × 50명)
- **DB 커넥션:** Supabase 기본 제공 (충분)

7.3 데이터 처리량

항목	일일 예상량	월간 예상량
출퇴근 기록	500건 (250명 × 2회)	15,000건
API 호출	2,000건	60,000건
DB 쿼리	10,000건	300,000건

7.4 확장성

- **수평 확장:** Supabase 자동 스케일링
- **데이터베이스:** PostgreSQL 파티셔닝 (년도별, 향후 고려)
- **캐싱:** 파트/사업장 정보는 클라이언트 캐싱 (변경 빈도 낮음)

---

8. 인프라 및 배포

8.1 호스팅 환경

컴포넌트	호스팅 플랫폼	리전
Backend	Supabase	Seoul (ap-northeast-2)
Web	Vercel	Auto (Global CDN)
iOS App	App Store	-
Android App	Google Play	-

8.2 CI/CD 파이프라인

GitHub Actions 워크플로우:

```yaml
# .github/workflows/deploy.yml
name: Deploy WorkFlow

on:
  push:
    branches: [main]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: subosito/flutter-action@v2
      - run: flutter test

  deploy-web:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: subosito/flutter-action@v2
      - run: flutter build web
      - uses: amondnet/vercel-action@v20
        with:
          vercel-token: ${{ secrets.VERCEL_TOKEN }}

  deploy-ios:
    needs: test
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v3
      - uses: subosito/flutter-action@v2
      - run: flutter build ios --release
      - uses: apple-actions/upload-testflight-build@v1

  deploy-android:
    needs: test
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: subosito/flutter-action@v2
```

```
      - run: flutter build appbundle
      - uses: r0adkll/upload-google-play@v1
```

8.3 환경 분리

환경	용도	Supabase 프로젝트
Development	로컬 개발	dev-workflow
Staging	테스트	staging-workflow
Production	실서비스	prod-workflow

8.4 백업 전략

- **Database:** Supabase 자동 백업 (일일)
- **수동 백업:** 주 1회 (일요일 자정)
- **백업 보관:** 30일
- **재해 복구 목표:** RPO 24시간, RTO 2시간

---

9. 제약 조건

9.1 기술적 제약

제약 사항	영향	완화 방안
GPS 실내 정확도	건물 내 오차 발생	지오펜싱 반경 100m로 여유
네트워크 불안정	출퇴근 기록 실패	로컬 저장 후 동기화
Flutter Web 성능	초기 로딩 느림	코드 스플리팅, 지연 로딩
iOS 위치 권한	항상 허용 필요	명확한 권한 요청 UI

9.2 법적 제약

항목	요구사항	대응
개인정보보호법	동의 필수	약관 동의 화면
근로기준법	출퇴근 기록 3년 보관	DB 자동 보관
위치정보법	위치 수집 동의	GPS 권한 요청 시 설명

9.3 운영 제약

제약 사항	세부 내용
개발 기간	1개월 (MVP)
개발 인력	1인 (AI 보조)
예산	초기 무료 티어 (Supabase, Vercel)

---

10. 기술적 의사결정

10.1 주요 의사결정 기록

10.1.1 Flutter 선택

결정: Flutter를 메인 프레임워크로 사용
이유:

- 단일 코드베이스로 Web/iOS/Android 지원
- 1인 개발자에게 최적화
- Hot Reload로 빠른 개발
- Material Design 3 기본 지원

대안:

- React Native: 성숙도는 높지만 Web 지원 약함
- Native 개발: 3배의 개발 시간 필요

10.1.2 Supabase 선택

결정: Supabase를 백엔드로 사용
이유:

- 빠른 MVP 개발 (인증/DB/Realtime 내장)
- PostgreSQL 기반 (강력한 쿼리 성능)
- 무료 티어 충분 (500MB DB, 2GB 전송)
- 향후 확장 용이

대안:

- Firebase: 비용이 높고 NoSQL이라 복잡한 쿼리 어려움
- 자체 백엔드: 개발 시간 2배 이상 소요

10.1.3 Riverpod 선택

결정: Riverpod를 상태관리 라이브러리로 사용
이유:

- Provider의 개선 버전 (컴파일 타임 안정성)
- 테스트 용이
- 확장성 우수

대안:

- BLoC: 보일러플레이트 많음
- GetX: 커뮤니티 논란 있음

10.2 향후 검토 사항

항목	시기	목적
CDN 도입	MVP 이후	이미지/파일 전송 최적화
APM 도입	정식 출시 후	성능 모니터링
자체 백엔드 전환	사용자 1,000명 이상	비용 최적화

부록

A. 개발 환경 설정

A.1 필수 도구

- Flutter SDK 3.19+
- Dart SDK 3.3+
- Android Studio / Xcode
- VS Code + Flutter Extension

A.2 Supabase 설정

```
# Supabase CLI 설치
npm install -g supabase

# 로컬 개발 환경 시작
supabase start

# 마이그레이션 적용
supabase db push
```

A.3 환경 변수 (.env)

```
SUPABASE_URL=https://xxxxx.supabase.co
SUPABASE_ANON_KEY=eyJhbGc...
GOOGLE_MAPS_API_KEY=AIzaSyC...
```

B. 참조 문서

- Flutter 공식 문서: https://flutter.dev
- Supabase 공식 문서: https://supabase.com/docs
- Riverpod 문서: https://riverpod.dev
- Material Design 3: https://m3.material.io

문서 승인:

- [ ] Tech Lead
- [ ] Backend Engineer
- [ ] Frontend Engineer
- [ ] DevOps Engineer

변경 이력:

버전	날짜	작성자	변경 내용
1.0	2026-02-02	Engineering Team	초안 작성