기술 스택 및 시스템 아키텍처

> *대학 출석 관리 시스템의 기술 스택과 아키텍처 개요*

목차

기술 스택 요약

Frontend

기술	버전	용도
Next.js	14.2.32	React 프레임워크 (App Router)
TypeScript	5.6.3	정적 타입 언어
React	18.3.1	UI 라이브러리
Tailwind CSS	3.4.13	유틸리티 CSS 프레임워크
html5-qrcode	2.3.8	QR 코드 스캔
qrcode	1.5.4	QR 코드 생성

Backend

기술	버전	용도
Node.js	20+	서버 런타임
Next.js API Routes	14.2.32	RESTful API
Supabase JS	2.45.4	데이터베이스 클라이언트
jose	6.1.0	JWT 인증
bcryptjs	3.0.2	비밀번호 해싱
Zod	3.25.76	스키마 검증

Database

기술	용도
PostgreSQL 15+	메인 데이터베이스 (Supabase)
Row Level Security (RLS)	데이터 접근 제어

Realtime Subscriptions	실시간 데이터 동기화

위치 추적 & 센서

기술	용도
Kalman Filter (kalmanjs)	GPS 노이즈 필터링
PDR (Pedestrian Dead Reckoning)	실내 위치 추적 (자체 구현)
Haversine Formula	GPS 거리 계산 (자체 구현)
GPS-PDR Fusion	실내/실외 융합 위치 추적
Environment Detector	실내/실외 자동 감지
Browser Geolocation API	GPS 좌표 수집
DeviceMotion API	가속도계, 자이로스코프

개발 도구

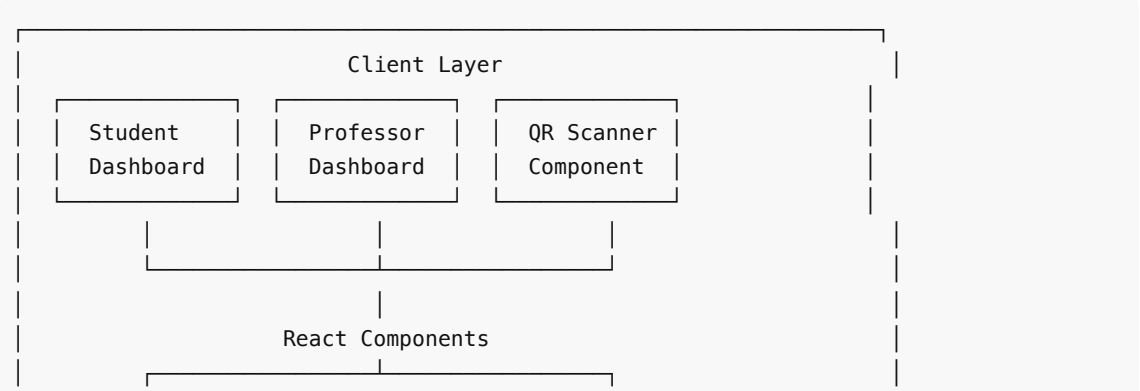
기술	용도
ESLint	코드 린팅
Playwright	E2E 테스트
TypeScript Compiler	타입 체크

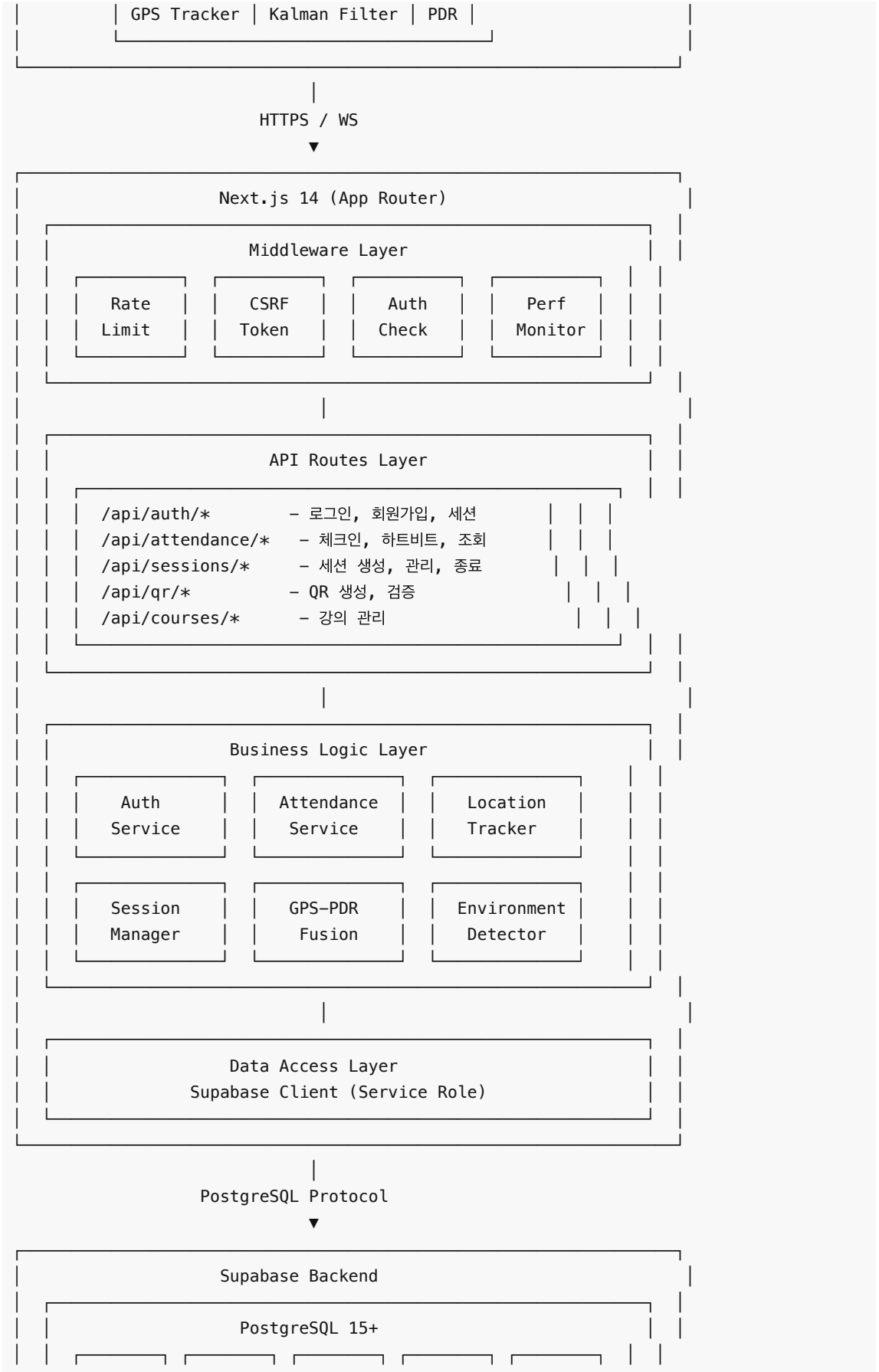
배포

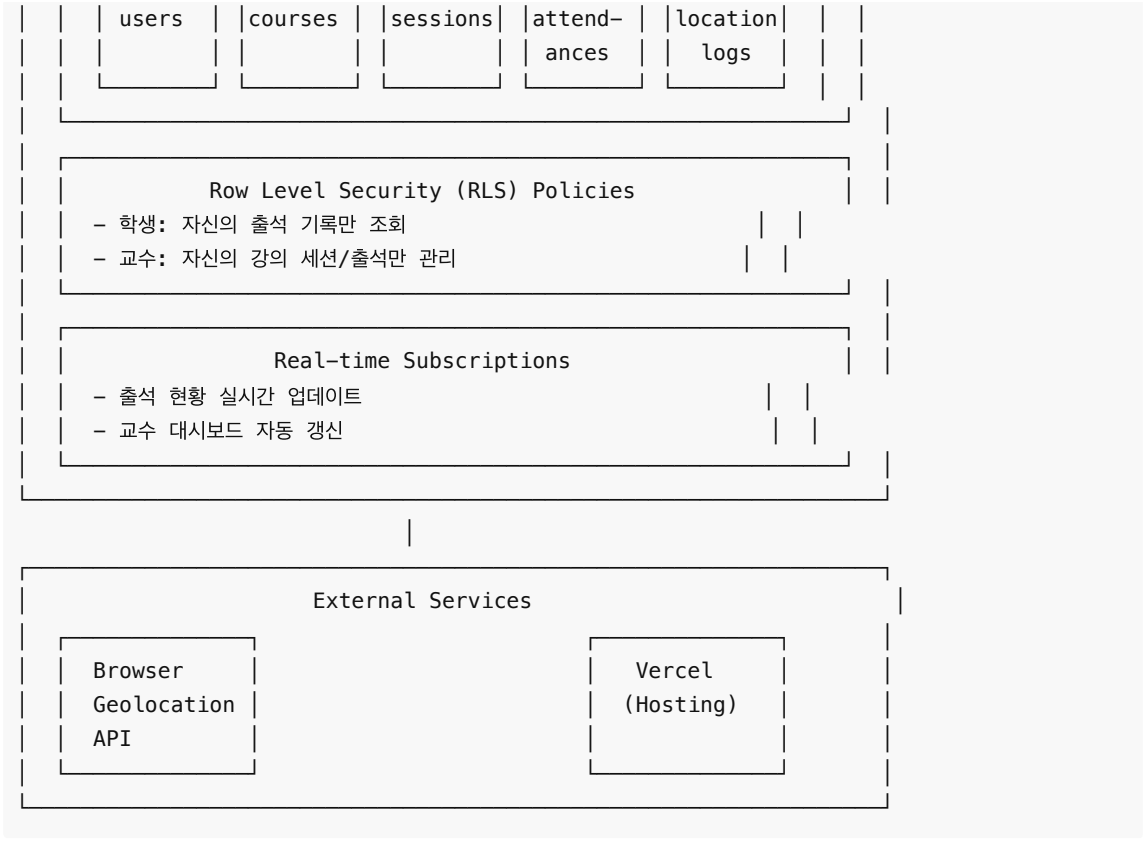
기술	용도
Vercel	호스팅 플랫폼
GitHub	버전 관리 및 CI/CD

시스템 아키텍처

전체 구조도




GPS Tracker | Kalman Filter | PDR
HTTPS / WS
Next.js 14 (App Router)
Middleware Layer
Rate Limit
CSRF Token
Auth Check
Perf Monitor
API Routes Layer
/api/auth/* – 로그인, 회원가입, 세션
/api/attendance/* – 체크인, 하트비트, 조회
/api/sessions/* – 세션 생성, 관리, 종료
/api/qr/* – QR 생성, 검증
/api/courses/* – 강의 관리
Business Logic Layer
Auth Service
Attendance Service
Location Tracker
Session Manager
GPS-PDR Fusion
Environment Detector
Data Access Layer
Supabase Client (Service Role)
PostgreSQL Protocol
Supabase Backend
PostgreSQL 15+



핵심 기술 상세

1. 위치 추적 시스템

GPS-PDR 융합 시스템

```
// lib/fusion/gps-pdr-fusion.ts
- GPS 데이터 + PDR 데이터 융합
- Kalman Filter로 노이즈 제거
- 실내/실외 자동 전환
- 정확도 30~50% 향상
```

동작 방식:

1. **GPS 우선 모드** (실외)
   - Kalman Filter (kalmanjs)로 GPS 노이즈 제거
   - 정확도: ±5-10m
2. **PDR 보조 모드** (실내)
   - 가속도계 + 자이로스코프 (DeviceMotion API)
   - Step Detection + Heading Estimation
   - 정확도: ±2-5m (단거리)
3. **환경 자동 감지**
   - GPS 신호 품질 모니터링

- 실내/실외 자동 판단
- 모드 자동 전환

Haversine 거리 계산

```
// lib/utils/geo.ts
- 지구 곡률을 고려한 정밀 거리 계산
- 오차: ±0.5% 이내
- 허용 반경: 기본 50-100m
```

2. 인증 & 보안 시스템

JWT 인증

```
// lib/auth.ts
- HttpOnly Cookie 저장 (XSS 방지)
- jose 라이브러리 사용
- 만료 시간: 7일
- bcryptjs로 비밀번호 해싱
```

Rate Limiting

```
// lib/middleware/rate-limit.ts
- Sliding Window 알고리즘
- 메모리 기반 (in-memory)
- 로그인: 5회/분
- 체크인: 10회/분
- QR 생성: 20회/시간
```

CSRF Protection

```
// lib/middleware/csrf.ts
- Double Submit Cookie 패턴
- 모든 POST/PUT/DELETE 요청 검증
```

3. 실시간 모니터링

Heartbeat 시스템

```
// lib/realtime/heartbeat-manager.ts
- 30초마다 위치 전송
- 2회 연속 이탈 시 자동 조퇴
- 실시간 위치 추적
```

성능 모니터링

```
// lib/monitoring/web-vitals.ts
- Web Vitals 측정 (FCP, LCP, CLS, FID, TTFB, INP)
- 구조화된 JSON 로깅
- 콘솔 기반 로깅
```

데이터 플로우

출석 체크 플로우

```
학생 QR 스캔
    │
    ▼
GPS 위치 수집 (3회 샘플링)
    │
    ▼
Kalman Filter 적용 (kalmanjs)
    │
    ▼
환경 감지 (실내/실외)
    │
    ├─ 실외 → GPS 우선
    └─ 실내 → PDR 보조
    │
    ▼
거리 검증 (Haversine)
    │
    ├─ 개발 환경 → 모든 거리 통과
    ├─ 프로덕션 + 허용 범위 내 → 출석 성공
    └─ 프로덕션 + 허용 범위 외 → 출석 실패
    │
    ▼
DB 저장 (출석 기록 + 위치 로그)
    │
    ▼
실시간 대시보드 업데이트 (Supabase Realtime)
```

QR 코드 생성 플로우

```
교수 QR 생성 요청
    │
    ▼
Rate Limit 체크 (20/hour)
    │
    ▼
세션 생성 (DB)
    │
    ├─ session_id (UUID)
    ├─ qr_code (암호화된 문자열)
    ├─ expires_at (5분 후)
```

```
    └─ classroom_location (위도, 경도, 반경)
    │
    ▼
QR 코드 이미지 생성 (qrcode 라이브러리)
    │
    ▼
클라이언트 반환
```

보안 아키텍처

다층 보안 구조

```
┌─────────────────────────────────────────────────────────┐
│ Layer 1: Network Security                               │
│  - HTTPS Only (Vercel 자동)                             │
│  - CORS 설정                                            │
└─────────────────────────────────────────────────────────┘
                          ▼
┌─────────────────────────────────────────────────────────┐
│ Layer 2: Application Security                           │
│  - Rate Limiting (Sliding Window, in-memory)            │
│  - CSRF Protection (Double Submit Cookie)               │
│  - XSS Prevention (Zod 검증 + 입력 새니타이제이션)      │
└─────────────────────────────────────────────────────────┘
                          ▼
┌─────────────────────────────────────────────────────────┐
│ Layer 3: Authentication & Authorization                 │
│  - JWT (HttpOnly Cookie, jose)                          │
│  - bcrypt Password Hashing                              │
│  - Role-based Access Control                            │
└─────────────────────────────────────────────────────────┘
                          ▼
┌─────────────────────────────────────────────────────────┐
│ Layer 4: Database Security                              │
│  - Row Level Security (RLS)                             │
│  - SQL Injection Prevention (Supabase Client)           │
│  - Encrypted Connections (PostgreSQL SSL)               │
└─────────────────────────────────────────────────────────┘
```

보안 기능 상세

위협	방어 방법	구현 위치
XSS	Zod 스키마 검증 + 입력 새니타이제이션	`lib/schemas/*`, `lib/utils/sanitize.ts`
CSRF	Double Submit Cookie	`lib/middleware/csrf.ts`
SQL Injection	Supabase Client (자동 파라미터화)	모든 DB 쿼리
Brute Force	Rate Limiting (in-memory)	`lib/middleware/rate-limit.ts`

Session Hijacking	HttpOnly + Secure Cookie	`lib/auth.ts`
Unauthorized Access	RLS + JWT 검증	Supabase + Middleware

디렉토리 구조

```
university-attendance-management/
├── app/                          # Next.js App Router
│   ├── api/                      # API Routes
│   │   ├── auth/                 # 인증 (로그인, 회원가입)
│   │   ├── attendance/           # 출석 (체크인, 하트비트)
│   │   ├── sessions/             # 세션 관리
│   │   ├── qr/                   # QR 코드 생성
│   │   └── courses/              # 강의 관리
│   ├── student/                  # 학생 페이지
│   ├── professor/                # 교수 페이지
│   └── auth/                     # 인증 페이지
│
├── lib/                          # 핵심 비즈니스 로직
│   ├── auth.ts                   # JWT 인증 (jose)
│   ├── supabase-admin.ts         # DB 클라이언트
│   ├── middleware/               # 미들웨어
│   │   ├── rate-limit.ts         # Rate Limiting
│   │   ├── csrf.ts               # CSRF 보호
│   │   └── performance.ts        # 성능 측정
│   ├── utils/                    # 유틸리티
│   │   ├── geo.ts                # GPS 거리 계산
│   │   ├── gps-filter.ts         # Kalman Filter
│   │   └── sanitize.ts           # XSS 방지
│   ├── fusion/                   # GPS-PDR 융합
│   │   ├── gps-pdr-fusion.ts
│   │   └── environment-detector.ts
│   ├── pdr/                      # PDR 시스템
│   │   ├── step-detector.ts
│   │   └── heading-estimator.ts
│   ├── location/                 # 위치 추적
│   │   └── location-tracker.ts
│   ├── realtime/                 # 실시간 통신
│   │   └── heartbeat-manager.ts
│   ├── schemas/                  # Zod 스키마
│   │   ├── auth.ts
│   │   ├── attendance.ts
│   │   └── qr.ts
│   └── session/                  # 세션 관리
│       └── session-service.ts
│
├── components/                   # React 컴포넌트
│   ├── qr/                       # QR 스캔/생성
│   ├── location/                 # 위치 선택
```

```
│   ├── sensors/                  # 센서 모니터
│   └── ui/                       # UI 컴포넌트
│
├── tests/                        # 테스트
│   ├── unit/                     # 단위 테스트
│   ├── e2e/                      # E2E 테스트 (Playwright)
│   └── api/                      # API 테스트
│
└── docs/                         # 문서
    ├── ARCHITECTURE.md           # 상세 아키텍처
    ├── API.md                    # API 문서
    ├── DEVELOPMENT.md            # 개발 가이드
    └── TECH_STACK.md             # 기술 스택 (이 문서)
```

환경별 설정

개발 환경 (NODE_ENV=development)

- **위치 검증**: 비활성화 (테스트 편의)
- **Rate Limiting**: 완화된 제한
- **로깅**: 상세 콘솔 로그

프로덕션 환경 (NODE_ENV=production)

- **위치 검증**: 활성화 (실제 거리 검증)
- **Rate Limiting**: 엄격한 제한
- **로깅**: 에러 및 중요 이벤트만

환경 변수

```
# Supabase
NEXT_PUBLIC_SUPABASE_URL=https://xxx.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJxxx...
SUPABASE_SERVICE_ROLE_KEY=eyJxxx...

# JWT
JWT_SECRET=your-secret-key

# 환경
NODE_ENV=development | production

# 로깅
LOG_LEVEL=debug | info | warn | error
```

성능 최적화

Frontend

- **Code Splitting**: Next.js 자동 코드 분할
- **Image Optimization**: Next.js Image 컴포넌트
- **Lazy Loading**: React.lazy() + Suspense

- **Memoization**: useMemo, useCallback

Backend

- **Database Indexing**: 주요 쿼리 컬럼 인덱싱
- **Connection Pooling**: Supabase 자동 관리
- **In-memory Caching**: Rate Limit 캐시

Network

- **HTTP/2**: Vercel 자동 지원
- **CDN**: Vercel Edge Network
- **Compression**: Gzip/Brotli 자동 압축

로깅

구조화된 로깅

```
// lib/logger/index.ts
{
  timestamp: "2025-11-10T02:46:05.734Z",
  level: "info",
  event: "checkin_success",
  data: {
    sessionId: "abc123...",
    studentId: "student1...",
    distance: 45,
    accuracy: 10
  }
}
```

로그 레벨

- **error**: 에러 발생 시
- **warn**: 경고 (느린 API, 성능 저하)
- **info**: 일반 이벤트 (출석 성공, QR 생성)
- **debug**: 디버깅 정보 (개발 환경)

참고 문서

- **상세 아키텍처**: 시스템 아키텍처 상세 설명
- **API 문서**: REST API 엔드포인트 명세
- **개발 가이드**: 로컬 개발 환경 설정
- **PDR 가이드**: PDR 시스템 튜닝