신일제약 실적관리 시스템 개발 요약 문서 (v3.1)

문서 버전: 3.1 (2025-07-30)

1. 개요 및 설계 철학

1.1. 문서의 목적

본 문서는 신일제약 실적관리 시스템의 **아키텍처, 핵심 기능, 데이터 구조, 그리고 주요 로직을 종합적으로 정리**한 공식 개발 기획 문서입니다. 시스템의 안정적인 유지보수, 신규 개발자의 온보딩, 그리고 향후 기능 확장을 위한 핵심 가이드라인으로 활용하는 것을 최우선 목적으로 합니다.

1.2. 시스템 설계 철학

본 시스템은 다음 세 가지 핵심 철학을 바탕으로 설계되었습니다.

1. **데이터 무결성 (Data Integrity):** 사용자가 입력한 원본 데이터는 절대 변경되지 않습니다. 모든 수정 및 변경 사항은 별도의 테이블(`performance_records_absorption`)에 이력으로 기록되어, 데이터의 정합성을 보장하고 모든 변경 과정을 추적할 수 있습니다.
2. **로직의 중앙화 (Centralized Logic):** 복잡한 데이터 처리, 집계, 계산 로직은 최대한 프론트엔드에서 분리하여 데이터베이스의 **뷰(View)와 함수(RPC)에 중앙화**합니다. 이를 통해 프론트엔드는 UI 표시에만 집중할 수 있어 코드의 복잡성이 감소하고 유지보수성이 향상됩니다.
3. **상태 기반 UI (State-Driven UI):** 모든 데이터와 UI 컴포넌트는 명확한 '상태'(`status` , `action` 등)를 가집니다. 이 상태 값에 따라 UI가 동적으로 변화하여, 사용자에게 현재 진행 상황을 명확하게 인지시키고 오작동을 방지합니다.

1.3. 개발 환경 및 기술 스택

1.3.1. 프론트엔드 (Frontend)

- **프레임워크**: Vue.js 3.5.13 (Composition API)
- **UI 라이브러리**: PrimeVue 4.2.0 (Aura 테마)
- **빌드 도구**: Vite 6.2.4
- **패키지 관리**: npm
- **라우팅**: Vue Router 4.5.0
- **상태 관리**: Vue 3 Composition API (Pinia 미사용)
- **HTTP 클라이언트**: Supabase JavaScript Client 2.49.4
- **추가 라이브러리**:
    - XLSX 0.18.5 (엑셀 처리)
    - File-saver 2.0.5 (파일 다운로드)
    - JSZip 3.10.1 (압축 파일 처리)

- html2canvas 1.4.1 (화면 캡처)
- jsPDF 3.0.1 (PDF 생성)
- CKEditor 5.41.4.2 (리치 텍스트 에디터)
- Quill 2.0.3 (텍스트 에디터)
- VueDraggable 4.1.0 (드래그 앤 드롭)

1.3.2. 백엔드 (Backend)

- **BaaS 플랫폼**: Supabase
- **데이터베이스**: PostgreSQL 15.x
- **인증**: Supabase Auth (Email/Password)
- **파일 저장소**: Supabase Storage
- **실시간 기능**: Supabase Realtime
- **Edge Functions**: Deno (TypeScript)

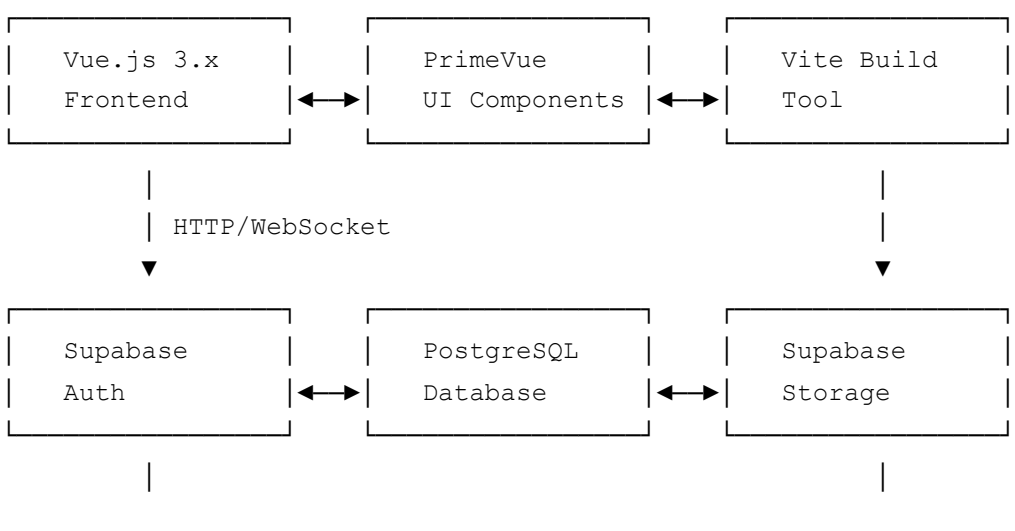
1.3.3. 개발 도구

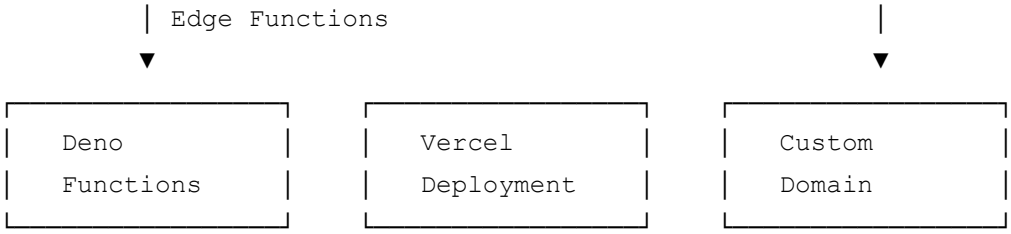
- **IDE**: Visual Studio Code
- **버전 관리**: Git
- **코드 품질**: ESLint 9.22.0
- **코드 포맷팅**: Prettier 3.5.3
- **형식 검사**: TypeScript (선택적)
- **API 테스트**: Supabase Dashboard
- **Vue 개발 도구**: Vite Plugin Vue DevTools 7.7.2

1.3.4. 배포 환경

- **프론트엔드 배포**: Vercel
- **백엔드 서비스**: Supabase Cloud
- **도메인**: 사용자 정의 도메인 (선택적)
- **SSL 인증서**: 자동 (Vercel/Supabase 제공)

1.4. 시스템 아키텍처





1.5. 개발 환경 설정

1.5.1. 필수 요구사항

- **Node.js**: 18.x 이상 (Vite 6.x 요구사항)
- **npm**: 9.x 이상
- **Git**: 2.x 이상
- **브라우저**: Chrome, Firefox, Safari, Edge (최신 버전)
- **메모리**: 최소 4GB RAM (대용량 엑셀 파일 처리 시)

1.5.2. 환경 변수 설정

```
# .env.local
VITE_SUPABASE_URL=your_supabase_project_url
VITE_SUPABASE_ANON_KEY=your_supabase_anon_key
VITE_APP_TITLE=신일제약 실적관리 시스템
VITE_APP_VERSION=3.0.0
```

1.5.3. 개발 서버 실행

```
# 의존성 설치
npm install

# 개발 서버 실행
npm run dev

# 프로덕션 빌드
npm run build

# 빌드 미리보기
npm run preview

# 코드 린팅
npm run lint

# 코드 포맷팅
npm run format
```

1.6. 데이터베이스 설계 원칙

1.6.1. 테이블 명명 규칙

- **일반 테이블**: `snake_case` (예: `performance_records`)
- **매핑 테이블**: `entity_entity_assignments` (예: `client_company_assignments`)
- **뷰**: `entity_details_view` (예: `review_details_view`)
- **함수**: `action_entity` (예: `calculate_absorption_rates`)

1.6.2. 컬럼 명명 규칙

- **기본 컬럼**: `snake_case` (예: `created_at`, `updated_at`)
- **상태 컬럼**: `status` (예: `review_status`, `user_status`)
- **액션 컬럼**: `action` (예: `review_action`)
- **수정 컬럼**: `field_modify` (예: `prescription_qty_modify`)
- **추가 컬럼**: `field_add` (예: `company_name_add`)

1.6.3. 인덱스 전략

- **기본키**: 모든 테이블에 `id` (UUID) 기본키
- **외래키**: 참조 무결성을 위한 인덱스
- **조회 최적화**: `review_status`, `settlement_month` 등 자주 조회되는 컬럼
- **복합 인덱스**: `(settlement_month, company_id, review_status)` 등

---

2. 핵심 데이터 흐름 (Data Flow Lifecycle)

하나의 실적 데이터가 생성되어 최종 정산되기까지의 여정은 다음과 같습니다.

1. `[생성]` **사용자가 실적 입력**:
    - 사용자가 UI를 통해 실적 정보를 입력하고 저장합니다.
    - 데이터는 `performance_records` 테이블에 저장됩니다.
    - 이때, `review_status` 는 **'대기'** 상태가 됩니다.
2. `[동기화]` **관리자가 검수 시작**:
    - 관리자가 '실적 검수' 화면에서 '불러오기'를 실행합니다.
    - `performance_records` 의 '대기' 상태 데이터를 '검수중'으로 변경합니다.
    - 동시에 `performance_records_absorption` 테이블에 복사하여 검수 작업을 진행합니다.
3. `[변경]` **관리자가 검수 진행**:
    - 관리자는 `performance_records_absorption` 의 데이터를 기반으로 수정, 추가, 삭제 작업을 수행합니다.
    - **수정**: `..._modify` 필드에 변경 값이 저장되고, `review_action` 은 '수정'이 됩니다.
    - **추가**: `..._add` 필드에 업체/거래처 정보가 저장되고, `review_action` 은 '추가'가 됩니다.
    - **삭제**: `review_action` 이 '삭제'로 업데이트됩니다. (실제 행 삭제 X)

4. **[확정] 관리자가 검수 완료**:
    - 관리자가 '검수 상태 변경' 버튼을 통해 검수를 마칩니다.
    - `performance_records_absorption`의 `review_status`와 원본 `performance_records`의 `review_status`가 모두 **'완료'**로 변경됩니다.
5. **[활용] 데이터 분석 및 공유**:
    - `review_status`가 '완료'된 데이터만이 '흡수율 분석'과 '정산내역서 공유' 화면에 집계될 자격을 얻습니다.
    - 모든 데이터 조회는 복잡한 테이블들을 미리 결합해놓은 `review_details_view`를 통해 이루어집니다.

---

3. 주요 기능 및 사용자 시나리오

3.1. 실적 검수 (`AdminPerformanceReviewView.vue`)

- **기능 목표**: 관리자가 사용자의 실적을 효율적으로 검토하고, 오류를 수정하며, 최종 데이터를 확정하는 인터페이스를 제공합니다.
- **사용자 시나리오**:
    1. 관리자는 '정산월'을 선택하고 '실적 정보 불러오기' 버튼을 클릭합니다.
    2. 시스템은 '대기' 상태의 모든 실적을 화면에 로드하고 '검수중' 상태로 전환합니다.
    3. 관리자는 각 행을 검토하며, 특정 행의 '수정(✎)' 버튼을 눌러 인라인 편집 모드로 전환합니다. 수량, 수수료율 등을 변경하고 '저장(✓)' 버튼을 누릅니다.
    4. 특정 행 아래에 새로운 실적을 추가하고 싶으면, '+' 버튼을 눌러 새 행을 만들고 정보를 입력 후 저장합니다.
    5. 검수를 마친 행들을 체크박스로 선택한 후, '검수 상태 변경' 버튼을 눌러 '완료' 상태로 확정합니다.

3.2. 흡수율 분석 (`AdminAbsorptionAnalysisView.vue`)

- **기능 목표**: 검수가 완료된 데이터를 기반으로, 제품별 처방액 대비 실제 매출(흡수율)을 분석하여 영업 성과를 다각도로 평가합니다.
- **사용자 시나리오**:
    1. 관리자는 '정산월', '업체', '병의원' 등 원하는 필터 조건을 설정합니다.
    2. '검수 완료 불러오기'를 클릭하여 조건에 맞는 '완료' 상태의 실적 데이터를 조회합니다.
    3. '흡수율 분석' 버튼을 클릭하면, 시스템은 백그라운드(DB 함수)에서 각 실적의 도매/직거래 매출을 합산하고 처방액과 비교하여 흡수율(%)을 계산한 후, 테이블의 '합산액' 및 '흡수율' 열을 업데이트합니다.

3.3. 정산내역서 공유 (`AdminSettlementShareView.vue`)

- **기능 목표**: 최종 확정된 실적을 **업체별로 깔끔하게 합산**하여 보여주고, 각 업체 담당자가 자신의 정산 내역을 조회할 수 있도록 공유 상태를 제어합니다.

- **사용자 시나리오**:
    1. 관리자는 '정산월'을 선택합니다.
    2. 시스템은 해당 월의 '완료'된 모든 실적을 업체별로 자동 합산하여, 업체당 한 줄의 요약 데이터를 보여줍니다. (총 처방액, 총 지급액 등)
    3. 관리자는 각 업체의 '공유' 체크박스를 선택/해제하고 '저장' 버튼을 눌러 공유 상태를 업데이트합니다.
    4. 공유가 활성화된 업체의 담당자는 자신의 계정으로 로그인하여 해당 월의 정산 내역을 조회할 수 있게 됩니다.

---

4. 최신 테이블 구조 개요

4.1. 핵심 테이블

- **`performance_records`** : 사용자 실적 원본 데이터 (review_status: '대기', '검수중', '완료')
- **`performance_records_absorption`** : 관리자 검수 및 분석용 테이블
- **`companies`** : 업체 정보 (user_type: 'admin', 'user')
- **`clients`** : 거래처(병의원) 정보
- **`products`** : 제품 정보 (commission_rate_a, commission_rate_b, commission_rate_c)
- **`pharmacies`** : 약국 정보
- **`wholesale_sales`** : 도매 매출 데이터
- **`direct_sales`** : 직거래 매출 데이터
- **`settlement_months`** : 정산월 관리
- **`settlement_share`** : 정산내역서 공유 설정
- **`notices`** : 공지사항 (is_pinned, view_count)
- **`performance_evidence_files`** : 실적 증빙 파일

4.2. 매핑 테이블

- **`client_company_assignments`** : 거래처-업체 매핑 (commission_grade)
- **`client_pharmacy_assignments`** : 거래처-약국 매핑

---

5. Database Functions 개요

5.1. 핵심 비즈니스 로직 함수들

- **`calculate_absorption_rates`** : 흡수율 분석 실행 (도매/직거래 매출액 계산)
- **`get_settlement_summary_by_company`** : 정산내역서 공유용 업체별 합산 데이터 조회
- **`create_settlement_summary`** : 정산월별 요약 데이터 생성

5.2. 변경사항 감지 함수들

- `check_for_updates` : 정산월별 변경사항 확인 (검수, 매핑, 매출 데이터 변경)
- `check_performance_changes` : 실적 데이터 변경사항 확인 (추가, 수정, 삭제 건수)

5.3. 데이터 조회 함수들

- `get_absorption_analysis_details` : 흡수율 분석 화면의 상세 데이터 조회
- `get_distinct_companies_from_analysis` : 업체 필터 목록 조회
- `get_distinct_clients_from_analysis` : 거래처 필터 목록 조회
- `get_distinct_settlement_months_from_analysis` : 정산월 필터 목록 조회

5.4. 디버깅 함수들

- `debug_absorption_calculation` : 흡수율 계산 디버깅용
- `debug_absorption_rates` : 흡수율 분석 디버깅용 (상세 매칭 정보)
- `debug_distribution_ratios` : 분배 비율 디버깅용

5.5. Edge Functions

- `register-user` : 사용자 회원가입 처리
- `reset-password` : 비밀번호 초기화

---

6. 향후 개선 과제 및 고려사항

- **권한 관리 고도화**: 현재는 'admin'과 'user' 역할만 존재하지만, 향후 특정 업체 담당자가 여러 업체를 관리하거나, 특정 메뉴에만 접근하는 등의 세분화된 권한 관리가 필요할 수 있습니다.
- **대시보드 기능**: 현재는 테이블 형태의 데이터 조회가 중심이지만, 주요 지표(월별 총 지급액, 상위 5개 업체 등)를 시각적인 차트와 그래프로 보여주는 대시보드 기능을 추가하여 사용성을 개선할 수 있습니다.
- **성능 최적화**: 데이터가 수십만 건 이상으로 증가할 경우, 현재의 View와 함수(RPC)의 성능을 모니터링하고 인덱스(Index) 추가, 쿼리 최적화 등의 작업이 필요할 수 있습니다.
- **사용자 경험(UX) 개선**: 엑셀 업로드/다운로드 기능의 편의성을 개선하고, 각 입력 필드에 대한 유효성 검사(Validation)를 강화하여 사용자 실수를 줄이는 방안을 고려해야 합니다.

7. 향후 개선 과제 및 고려사항

7.1. 기능 확장 및 개선

- **대시보드 기능**: 현재는 테이블 형태의 데이터 조회가 중심이지만, 주요 지표(월별 총 지급액, 상위 5개 업체 등)를 시각적인 차트와 그래프로 보여주는 대시보드 기능을 추가하여 사용성을 개선할 수 있습니다.
- **실시간 알림 시스템**: 실적 검수 완료, 새로운 공지사항 등 중요 이벤트에 대한 실시간 알림 기능을 추가하여 사용자 경험을 향상시킬 수 있습니다.

- **모바일 앱 개발**: 현재 웹 기반 시스템을 모바일 앱으로 확장하여 언제 어디서나 접근 가능한 시스템을 구축할 수 있습니다.

7.2. 성능 최적화

- **성능 최적화**: 데이터가 수십만 건 이상으로 증가할 경우, 현재의 View와 함수(RPC)의 성능을 모니터링하고 인덱스(Index) 추가, 쿼리 최적화 등의 작업이 필요할 수 있습니다.
- **캐싱 전략**: Redis 등을 활용한 데이터 캐싱 시스템을 도입하여 반복적인 쿼리 성능을 개선할 수 있습니다.
- **CDN 도입**: 정적 파일(이미지, CSS, JS)에 대한 CDN 도입으로 로딩 속도를 개선할 수 있습니다.

7.3. 사용자 경험 개선

- **사용자 경험(UX) 개선**: 엑셀 업로드/다운로드 기능의 편의성을 개선하고, 각 입력 필드에 대한 유효성 검사(Validation)를 강화하여 사용자 실수를 줄이는 방안을 고려해야 합니다.
- **접근성 개선**: 스크린 리더 지원, 키보드 네비게이션 강화 등 접근성을 개선하여 모든 사용자가 편리하게 사용할 수 있도록 해야 합니다.
- **다국어 지원**: 향후 해외 진출을 고려하여 다국어 지원 시스템을 구축할 수 있습니다.

7.4. 보안 강화

- **2단계 인증**: SMS, 이메일, OTP 앱 등을 활용한 2단계 인증 시스템을 도입할 수 있습니다.
- **감사 로그**: 모든 데이터 변경 사항에 대한 상세한 감사 로그를 기록하여 보안을 강화할 수 있습니다.
- **데이터 암호화**: 민감한 데이터에 대한 암호화를 강화하여 데이터 보안을 향상시킬 수 있습니다.

7.5. 시스템 확장성

- **마이크로서비스 아키텍처**: 현재 모놀리식 구조를 마이크로서비스로 분리하여 시스템 확장성을 개선할 수 있습니다.
- **컨테이너화**: Docker를 활용한 컨테이너화로 배포 및 운영 효율성을 향상시킬 수 있습니다.
- **자동화**: CI/CD 파이프라인 구축으로 개발 및 배포 프로세스를 자동화할 수 있습니다.

7.6. 데이터 분석 및 인사이트

- **BI 도구 연동**: Tableau, Power BI 등과 연동하여 고급 데이터 분석 기능을 제공할 수 있습니다.
- **머신러닝**: 실적 예측, 이상 패턴 감지 등 머신러닝 기능을 도입하여 비즈니스 인사이트를 제공할 수 있습니다.
- **API 확장**: 외부 시스템과의 연동을 위한 RESTful API를 확장하여 시스템 통합성을 향상시킬 수 있습니다.