

연세바로치과 스케줄 관리 시스템 - 완전 개발 계획서 v3.0

작성일: 2025-10-22

버전: 3.0 (최종 통합)

작성자: 프로젝트 분석팀

문서 유형: 개발자용 통합 문서

목차

- 프로젝트 전체 분석
- 누락된 기능 및 페이지 파악
- 전체 파일 구조 및 설계
- 데이터 흐름 및 로직 분석
- 개발 단계별 실행 계획
- Prisma 스키마 설계
- 핵심 API 명세
- 구현 우선순위별 상세 가이드

1. 프로젝트 전체 분석

1.1 문서 분석 요약

문서명	주요 내용	완성도	상태
기능명세서 Part1	기본 워크플로우, 핵심 기능, 데이터 구조	90%	✓ 완료
기능명세서 Part2	비즈니스 로직, 알고리즘, 보안	85%	✓ 완료
화면구성 및 설계	UI/UX, 컴포넌트 구조, 라우팅	80%	✓ 완료
추가 및 수정 사항	변경사항, 새 기능, 제거 기능	100%	✓ 완료

1.2 프로젝트 목적

연세바로치과의 월간 직원 근무 스케줄을 자동화하여:

- 시간 절감:** 3시간 → 15분 (85% 감소)
- 형평성 보장:** 야간/주말 근무의 공정한 배분
- 효율성 향상:** 직원 자율 신청, 실시간 알림
- 확장성:** 다른 치과/의원으로 확장 가능

1.3 기술 스택

프론트엔드

```
yaml

Framework: Next.js 14 (App Router)
Language: TypeScript 5.x
UI Library: Tailwind CSS 3.x, shadcn/ui
State Management: Zustand, React Query
Charts: Recharts
Date Handling: date-fns
Validation: Zod
```

백엔드

```
yaml

Runtime: Node.js 20.x
Framework: Next.js API Routes
ORM: Prisma 5.x
Database: PostgreSQL 16
Authentication: NextAuth.js 4.x
File Storage: AWS S3 (백업용)
```

배포

```
yaml

Hosting: Vercel
Database: Supabase (PostgreSQL)
Storage: AWS S3
Monitoring: Sentry
```

1.4 핵심 개선 사항

★ 새로 추가할 핵심 기능

```
typescript
```

1. 원장 요일별 패턴 저장 시스템

위치:  설정 > 원장 관리 > 요일별 패턴

목적: 한 번 설정으로 매달 재사용

효과: 원장 스케줄 입력 시간 95% 단축

2. 연차 자율 신청 시스템 (최우선 기능)

- 신청 링크 생성 (관리자)
- 직원용 신청 페이지 (로그인 불필요)
- 실시간 슬롯 관리 (동시성 제어)
- 관리자 확정 기능

효과: 관리자가 연차 수집하러 돌아다니지 않음

3. 연차관리 전용 페이지

- 대시보드 (전체 현황)
- 달력뷰 (날짜별 신청 현황)
- 목록뷰 (신청 목록)
- 직원별뷰 (개인별 신청 내역)

4. 월간/주간 자동 배치

- 월간 일괄 배치 (30일 한번에)
- 주간 부분 배치
- 스마트 배치 (기존 유지 + 필요한 것만 변경)

효과: 날짜별 작업 → 월간 일괄 작업으로 전환

5. 알림 센터

- 프로그램 내 실시간 알림
- 연차 신청 알림
- 마감 임박 알림
- 배치 완료 알림

6. Web Share API 연동

- 모바일에서 카톡/문자로 쉽게 공유
- 별도 API 등록 불필요

✖ 제거할 기능

typescript

1. 연차 입력 **모드** (달력 드래그)

제거 이유: 자율 신청 시스템으로 대체

2. 연차-휴무 붙이기 설정

제거 이유: 너무 복잡, 실제 사용 빈도 낮음

3. 연속 휴무 설정

제거 이유: 불필요한 제약

4. 이메일 알림

제거 이유: 프로그램 내 알림으로 충분

5. 복잡한 패턴 분석 화면

제거 이유: 사용하지 않는 기능

2. 누락된 기능 및 페이지 파악

2.1 ☒ 기존 문서에 있는 페이지

- ☒ /login - 로그인
- ☒ /setup - 최초 설정
- ☒ /calendar - 달력뷰 (메인)
- ☒ /statistics - 통계
- ☒ /settings/* - 설정 (직원, 원장, 규칙 등)
- ☒ /logs - 활동 로그

2.2 ★ 추가해야 할 페이지 (누락됨)

2.2.1 연차관리 페이지군 (신규)

```

└─ src/app/(dashboard)/leave-management/
  └─ page.tsx // 연차관리 대시보드 ★
    └─ 현황 요약, 신청 링크 생성, 통계
  └─ calendar-view/
    └─ page.tsx // 연차 달력뷰 ★
      └─ 날짜별 신청 현황 시각화
  └─ list-view/
    └─ page.tsx // 연차 목록뷰 ★
      └─ 전체 신청 목록 (필터/검색)
  └─ staff-view/
    └─ page.tsx // 연차 직원별뷰 ★
      └─ 개인별 신청 내역
  └─ period-setup/
    └─ page.tsx // 신청 기간 설정 ★
      └─ 링크 생성, 슬롯 설정, 공유

```

2.2.2 직원용 연차 신청 페이지 (외부 접근)

```

└─ src/app/(public)/leave-apply/[token]/
  └─ page.tsx // 연차 신청 메인 페이지 ★
    └─ 실시간 슬롯 표시
    └─ 날짜 선택
    └─ 유형 선택 (연차/오프/반차)
    └─ 신청 제출
  └─ layout.tsx // 공개 레이아웃
    └─ 로그인 불필요 설정
  └─ success/
    └─ page.tsx // 신청 완료 페이지
      └─ 신청 내역 확인

```

2.2.3 스케줄 확인 페이지 (외부 접근)

```

src/app/(public)/schedule-view/[token]/
├── page.tsx // 스케줄 조회 페이지 ★
│   ├── 간단 인증 (이름 + 생년월일)
│   ├── 개인 스케줄 조회
│   └── Excel/PDF 다운로드
└── auth.tsx // 간단 인증 컴포넌트

```

2.2.4 알림 센터

```

src/app/(dashboard)/notifications/
├── page.tsx // 알림 센터 전체뷰 ★
│   ├── 전체 알림 목록
│   ├── 필터 (읽음/안읽음)
│   └── 일괄 읽음 처리

```

2.2.5 원장 패턴 설정 페이지

```

src/app/(dashboard)/settings/doctors/pattern/
├── page.tsx // 요일별 패턴 설정 ★
│   ├── 요일별 원장 조합 저장
│   ├── 야간 진료 여부
│   └── 패턴 미리보기

```

2.3 ★ 추가해야 할 컴포넌트

typescript

src/components/

```
├── leave-management/ // 연차관리 전용 ★
│   ├── Dashboard.tsx // 대시보드 메인
│   ├── CalendarView.tsx // 달력 형태
│   ├── ListView.tsx // 목록 형태
│   ├── StaffView.tsx // 직원별 뷰
│   ├── PeriodSetup.tsx // 신청 기간 설정
│   ├── ApplicationCard.tsx // 신청 카드
│   ├── SlotStatus.tsx // 슬롯 현황
│   └── ConfirmModal.tsx // 확정 모달
│
├── public-apply/ // 직원 신청용 ★
│   ├── DateSelector.tsx // 날짜 선택 (캘린더)
│   ├── TypeSelector.tsx // 유형 선택 (연차/오프)
│   ├── RealTimeStatus.tsx // 실시간 슬롯 현황
│   ├── ApplicationForm.tsx // 신청 폼
│   ├── SuccessMessage.tsx // 완료 메시지
│   └── ConfirmationModal.tsx // 확인 모달
│
├── notifications/ // 알림 ★
│   ├── NotificationBell.tsx // 헤더 벨 아이콘
│   ├── NotificationPanel.tsx // 드롭다운 패널
│   ├── NotificationItem.tsx // 개별 알림
│   ├── NotificationBadge.tsx // 읽지 않음 배지
│   └── NotificationFilter.tsx // 필터 (타입별)
│
└── pattern/ // 원장 패턴 ★
    ├── DayPatternEditor.tsx // 요일별 편집기
    ├── PatternPreview.tsx // 미리보기
    ├── PatternApplyButton.tsx // 적용 버튼
    └── PatternList.tsx // 저장된 패턴 목록
```

2.4 ★ 추가해야 할 API 엔드포인트

typescript

src/app/api/

```
|— doctor-pattern/           // 원장 패턴 ★
|   |— route.ts              // GET (조회), POST (생성)
|   |— [id]/
|       |— route.ts          // PUT (수정), DELETE (삭제)
|       |— apply/
|           |— route.ts       // POST (달력에 패턴 적용)
|
|— leave-apply/              // 직원 신청 (공개) ★
|   |— [token]/
|       |— route.ts          // GET (페이지 데이터)
|       |— submit/
|           |— route.ts       // POST (신청 제출)
|       |— status/
|           |— route.ts       // GET (실시간 슬롯)
|       |— my-application/
|           |— route.ts       // GET (내 신청 내역)
|
|— leave-management/         // 연차 관리 (관리자) ★
|   |— route.ts              // GET (전체 현황)
|   |— period/
|       |— route.ts          // POST (기간 생성)
|       |— [id]/
|           |— route.ts       // PUT (수정), DELETE (삭제)
|           |— confirm/
|               |— route.ts    // POST (확정) ★
|               |— reopen/
|                   |— route.ts // POST (재오픈)
|   |— calendar-view/
|       |— route.ts          // GET (달력 데이터)
|   |— list-view/
|       |— route.ts          // GET (목록 데이터)
|   |— staff-view/
|       |— route.ts          // GET (직원별 데이터)
|       |— [id]/
|           |— route.ts       // PUT (수정), DELETE (삭제)
|
|— auto-assign/
|   |— monthly/              // 월간 배치 ★
|       |— route.ts          // POST
|   |— weekly/               // 주간 배치 ★
|       |— route.ts          // POST
|   |— daily/                // 일별 배치
|       |— route.ts          // POST
```

```

└─ notifications/           // 알림 ★
  └─ route.ts               // GET (목록)
    └─ unread/
      └─ route.ts           // GET (안 읽은 것만)
    └─ [id]/
      └─ read/
        └─ route.ts         // PUT (읽음 처리)
      └─ route.ts           // DELETE (삭제)
  └─ read-all/
    └─ route.ts             // PUT (모두 읽음)

```

3. 전체 파일 구조 및 설계

3.1 완전한 프로젝트 디렉토리 구조

```

yonsei-dental-schedule/
├─ package.json
├─ next.config.js
├─ tsconfig.json
├─ tailwind.config.ts
├─ .env.local
├─ .gitignore
├─ README.md
├─
├─ prisma/
│  ├─ schema.prisma        ★ 업데이트 필요 (신규 테이블 5개)
│  ├─ seed.ts
│  └─ migrations/
├─
├─ public/
│  ├─ logo.png
│  ├─ icon-192.png
│  ├─ icon-512.png
│  └─ manifest.json
├─
└─ src/
   └─ app/
      ├─ layout.tsx
      ├─ page.tsx
      └─ globals.css
      └─
      └─ (auth)/
         └─ login/
            └─ page.tsx
         └─ setup/

```

└─ page.tsx

└─ (dashboard)/

└─ layout.tsx

└─ calendar/

└─ page.tsx

└─ leave-management/ ★ 신규 디렉토리

└─ page.tsx

└─ calendar-view/

└─ page.tsx

└─ list-view/

└─ page.tsx

└─ staff-view/

└─ page.tsx

└─ period-setup/

└─ page.tsx

└─ statistics/

└─ page.tsx

└─ notifications/ ★ 신규

└─ page.tsx

└─ settings/

└─ page.tsx

└─ staff/

└─ page.tsx

└─ doctors/

└─ page.tsx

└─ pattern/ ★ 신규

└─ page.tsx

└─ rules/

└─ page.tsx

└─ holidays/

└─ page.tsx

└─ fairness/

└─ page.tsx

└─ notifications/ ★ 수정

└─ page.tsx

└─ backup/

└─ page.tsx

└─ logs/

└─ page.tsx

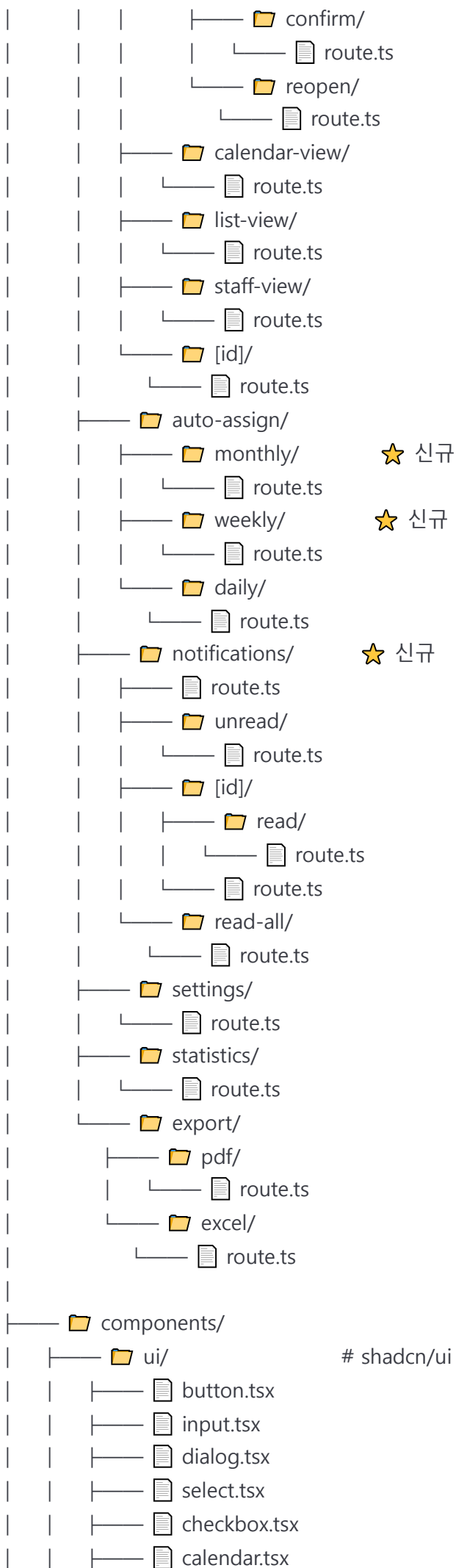
```
├── (public)/
│   ├── layout.tsx
│   ├── leave-apply/
│   │   ├── [token]/
│   │   │   ├── page.tsx
│   │   │   └── success/
│   │   │       └── page.tsx
│   │   └── schedule-view/
│   │       ├── [token]/
│   │       │   ├── page.tsx
│   │       │   └── auth.tsx
│   └── api/
│       ├── auth/
│       │   ├── [...nextauth]/
│       │   │   └── route.ts
│       │   └── schedule/
│       │       ├── route.ts
│       │       └── [id]/
│       │           └── route.ts
│       ├── staff/
│       │   ├── route.ts
│       │   └── [id]/
│       │       └── route.ts
│       ├── doctors/
│       │   └── route.ts
│       ├── doctor-pattern/
│       │   ├── route.ts
│       │   ├── [id]/
│       │   │   └── route.ts
│       │   └── apply/
│       │       └── route.ts
│       ├── leave-apply/
│       │   ├── [token]/
│       │   │   └── route.ts
│       │   ├── submit/
│       │   │   └── route.ts
│       │   ├── status/
│       │   │   └── route.ts
│       │   └── my-application/
│       │       └── route.ts
│       ├── leave-management/
│       │   ├── route.ts
│       │   ├── period/
│       │   │   ├── route.ts
│       │   │   └── [id]/
│       │   │       └── route.ts
```

★ 신규 라우트 그룹
(인증 불필요)

★ 신규






★ 신규

★ 신규









└─ ...









└─  layout/

- └─  Header.tsx
- └─  Sidebar.tsx
- └─  Footer.tsx
- └─  NotificationBell.tsx ★ 신규
- └─  DashboardLayout.tsx







└─  calendar/

- └─  CalendarView.tsx
- └─  CalendarGrid.tsx
- └─  CalendarCell.tsx
- └─  DayDetailPopup.tsx
- └─  MonthNavigator.tsx
- └─  PatternApplyButton.tsx ★ 신규





└─  leave-management/ ★ 신규 디렉토리

- └─  Dashboard.tsx
- └─  CalendarView.tsx
- └─  ListView.tsx
- └─  StaffView.tsx
- └─  PeriodSetup.tsx
- └─  ApplicationCard.tsx
- └─  SlotStatus.tsx
- └─  ConfirmModal.tsx





└─  public-apply/ ★ 신규 디렉토리








- └─  DateSelector.tsx
- └─  TypeSelector.tsx
- └─  RealTimeStatus.tsx
- └─  ApplicationForm.tsx
- └─  SuccessMessage.tsx
- └─  ConfirmationModal.tsx







└─  notifications/ ★ 신규 디렉토리









- └─  NotificationPanel.tsx
- └─  NotificationItem.tsx
- └─  NotificationBadge.tsx
- └─  NotificationFilter.tsx
























└─  pattern/ ★ 신규 디렉토리

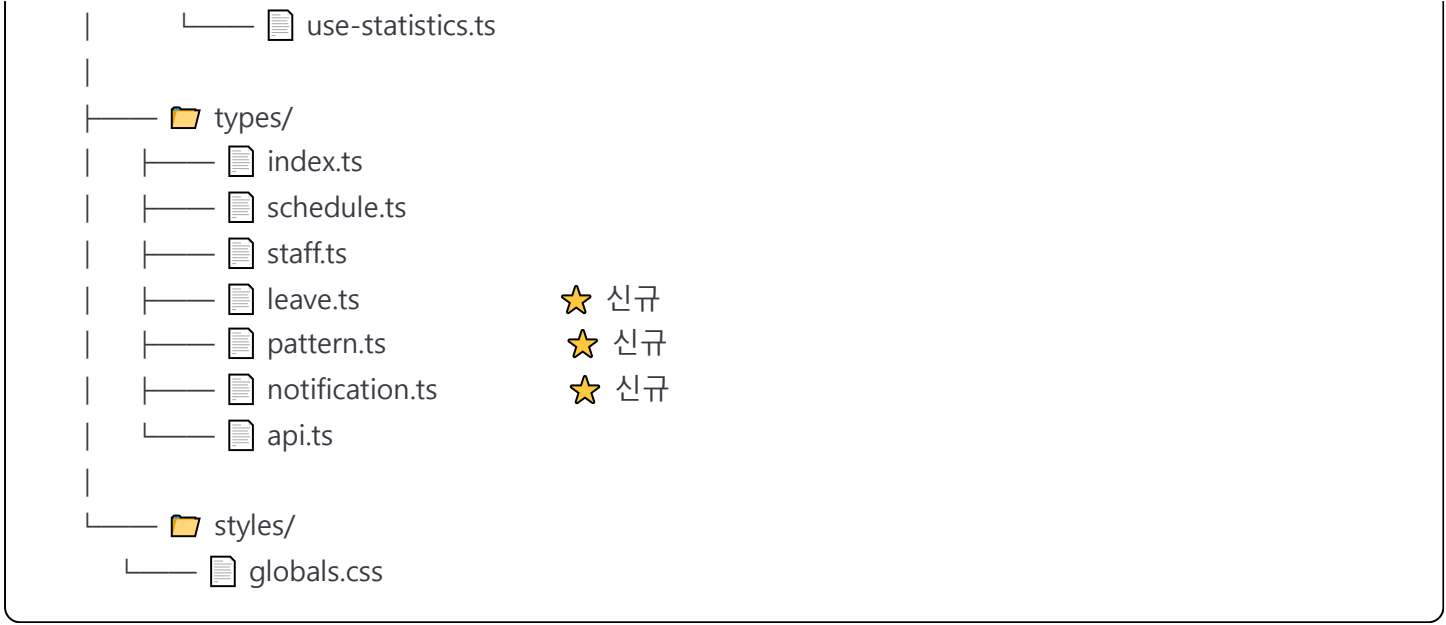
- └─  DayPatternEditor.tsx
- └─  PatternPreview.tsx
- └─  PatternApplyButton.tsx
- └─  PatternList.tsx

- └─  schedule/
 - └─  EditPanel.tsx
 - └─  DoctorSelector.tsx
 - └─  StaffAssignment.tsx
 - └─  AutoAssignModal.tsx ★ 수정 (월간/주간 추가)
 - └─  ValidationResult.tsx
 - └─  FairnessChart.tsx

- └─  statistics/
 - └─  Overview.tsx
 - └─  StaffWorkload.tsx
 - └─  FairnessReport.tsx
 - └─  MonthlyTrend.tsx
 - └─  ExportOptions.tsx

- └─  settings/
 - └─  StaffManager.tsx
 - └─  DoctorManager.tsx
 - └─  RuleSettings.tsx
 - └─  HolidayManager.tsx
 - └─  FairnessSettings.tsx
 - └─  NotificationSettings.tsx ★ 수정
 - └─  BackupRestore.tsx

- └─  lib/
 - └─  prisma.ts
 - └─  auth.ts
 - └─  utils.ts
 - └─  date-utils.ts
 - └─  algorithms/
 - └─  auto-assign.ts
 - └─  monthly-assign.ts ★ 신규
 - └─  weekly-assign.ts ★ 신규
 - └─  fairness-calculator.ts
 - └─  validation.ts
 - └─  services/
 - └─  schedule-service.ts
 - └─  leave-service.ts ★ 신규
 - └─  pattern-service.ts ★ 신규
 - └─  notification-service.ts ★ 신규
 - └─  export-service.ts
 - └─  hooks/
 - └─  use-schedule.ts
 - └─  use-leave-management.ts ★ 신규
 - └─  use-notifications.ts ★ 신규
 - └─  use-pattern.ts ★ 신규
 - └─  use-auto-assign.ts



4. 데이터 흐름 및 로직 분석

4.1 전체 데이터 흐름도



날짜/유형 선택 → 신청



StaffLeaveApplication 생성



LeaveSlotStatus 업데이트 (트랜잭션)



관리자에게 실시간 알림 



관리자 → 연차관리 페이지에서 확인



[확정] 클릭



Schedule DB에 연차 정보 반영



신청 페이지 자동 닫힘

【3. 자동 배치 흐름】

관리자 → [월간 자동 배치] 클릭



배치 옵션 선택:

- 스마트 배치 (기존 유지)
- 완전 재배치



알고리즘 실행:

- ① 연차 직원 제외
- ② 등급별 필요 인원 계산
- ③ 형평성 점수 계산
- ④ 최적 배치 수행



진행률 실시간 표시 (0% → 100%)



배치 결과 검증



Schedule DB 업데이트



형평성 재계산 및 표시

【4. 알림 흐름】

이벤트 발생:

- 연차 신청
- 슬롯 마감 임박
- 배치 완료
- 시스템 알림

↓
Notification DB 생성
↓
실시간 푸시 (WebSocket/SSE)
↓
헤더 🛎 아이콘에 뱃지 표시
↓
사용자 클릭 → 알림 패널 열림
↓
[읽음] 처리

4.2 핵심 비즈니스 로직

4.2.1 월간 자동 배치 알고리즘

typescript

/**

* 월간 자동 배치 로직

*

* 입력:

* - targetMonth: Date (2025-01-01)

* - mode: 'smart' | 'full' (스마트/완전재배치)

* - considerLeave: boolean (연차 고려 여부)

*

* 출력:

* - 30일치 배치 결과

* - 형평성 점수

* - 경고 메시지

*/

async function monthlyAutoAssign(
 targetMonth: Date,
 mode: 'smart' | 'full',
 considerLeave: boolean
) {

// 1. 해당 월의 모든 날짜 가져오기

const dates = getAllDatesInMonth(targetMonth)

// 2. 각 날짜별 스케줄 정보 조회

const schedules = await prisma.schedule.findMany({
 where: {
 date: { gte: startOfMonth, lte: endOfMonth }
 },
 include: { staff: true, doctors: true }
})

// 3. 연차 정보 조회 (확정된 것만)

const leaves = await prisma.staffLeaveApplication.findMany({
 where: {
 period: { targetMonth, status: 'confirmed' }
 }
})

// 4. 스마트 배치인 경우: 기존 배치 유지

let staffToAssign = []
if (mode === 'smart') {
 // 기존에 배치된 직원은 유지
 staffToAssign = getUnassignedStaffOnly(schedules)
} else {
 // 완전 재배치: 모두 초기화
 await clearAllAssignments(schedules)
 staffToAssign = await getAllActiveStaff()

```
}
```

```
// 5. 날짜별 반복 배치
```

```
for (const date of dates) {  
  const schedule = schedules.find(s => isSameDay(s.date, date))  
  if (!schedule) continue
```

```
// 5-1. 해당 날짜의 연차 직원 제외
```

```
const leavingStaff = leaves  
  .filter(l => l.applications.some(a => isSameDay(a.date, date)))  
  .map(l => l.staffName)
```

```
const availableStaff = staffToAssign.filter(  
  s => !leavingStaff.includes(s.name)  
)
```

```
// 5-2. 등급별 필요 인원 계산
```

```
const requirements = calculateRequirements(  
  schedule.doctors,  
  schedule.hasNightShift  
)
```

```
// 5-3. 형평성 기반 배치
```

```
const assigned = assignStaffWithFairness(  
  availableStaff,  
  requirements,  
  date  
)
```

```
// 5-4. DB 업데이트
```

```
await prisma.schedule.update({  
  where: { id: schedule.id },  
  data: {  
    staff: { connect: assigned.map(s => ({ id: s.id })) },  
    leaveCount: leavingStaff.length,  
    leaveStaff: leavingStaff  
  }  
})
```

```
// 5-5. 진행률 업데이트 (SSE)
```

```
emitProgress(dates.indexOf(date) / dates.length * 100)  
}
```

```
// 6. 전체 형평성 재계산
```

```
const fairness = await recalculateFairness(targetMonth)
```

```
// 7. 검증
```

```
const validation = validateMonthlySchedule(schedules)
```

```
return {  
  success: true,  
  assigned: schedules.length,  
  fairness,  
  warnings: validation.warnings  
}  
}
```

4.2.2 실시간 슬롯 관리 로직

typescript

```
/**
```

```
* 연차 신청 시 슬롯 관리
```

```
* 동시성 제어 포함
```

```
*/
```

```
async function submitLeaveApplication(
```

```
  token: string,
```

```
  staffName: string,
```

```
  applications: { date: Date, type: string }[]
```

```
) {
```

```
  // 트랜잭션으로 동시성 제어
```

```
  return await prisma.$transaction(async (tx) => {
```

```
    // 1. 신청 기간 확인
```

```
    const period = await tx.leaveApplicationPeriod.findUnique({
```

```
      where: { linkToken: token }
```

```
    })
```

```
    if (!period || period.status !== 'open') {
```

```
      throw new Error('신청 기간이 아닙니다')
```

```
    }
```

```
    // 2. 각 날짜별 슬롯 확인 및 예약
```

```
    for (const app of applications) {
```

```
      const slot = await tx.leaveSlotStatus.findUnique({
```

```
        where: {
```

```
          periodId_date: {
```

```
            periodId: period.id,
```

```
            date: app.date
```

```
          }
```

```
        },
```

```
        // 비관적 잠금
```

```
        lock: 'pessimistic_write'
```

```
      })
```

```
      if (!slot || slot.isFull) {
```

```
        throw new Error(`${format(app.date, 'M/d')}는 마감되었습니다`)
```

```
      }
```

```
    // 슬롯 업데이트
```

```
    await tx.leaveSlotStatus.update({
```

```
      where: { id: slot.id },
```

```
      data: {
```

```
        usedSlots: { increment: 1 },
```

```
        remainingSlots: { decrement: 1 },
```

```
        isFull: slot.usedSlots + 1 >= slot.maxSlots,
```

```
        applicants: { push: staffName }
```

```

    }
  })
}

// 3. 신청 내역 저장
const application = await tx.staffLeaveApplication.create({
  data: {
    periodId: period.id,
    staffName,
    applications,
    ipAddress: getClientIP(),
    source: 'staff'
  }
})

// 4. 관리자에게 알림
await tx.notification.create({
  data: {
    userId: period.createdBy,
    type: 'leave_apply',
    title: '새로운 연차 신청',
    content: `${staffName}님이 ${applications.length}건 신청`
  }
})

return application
})
}

```

4.2.3 형평성 계산 로직

typescript

/**

* 형평성 점수 계산

* 야간 근무, 주말 근무, 연차 사용 고려

*/

async function calculateFairnessScore(

 staffId: string,

 month: Date

) {

 const schedules = await prisma.schedule.findMany({

 where: {

 date: { gte: startOfMonth(month), lte: endOfMonth(month) },

 staff: { some: { id: staffId } }

 }

 })

 // 1. 야간 근무 횟수

 const nightShiftCount = schedules.filter(s => s.hasNightShift).length

 // 2. 주말 근무 횟수

 const weekendCount = schedules.filter(s => {

 const day = getDay(s.date)

 return day === 0 || day === 6

 }).length

 // 3. 연차 사용 일수

 const leaveCount = schedules.reduce((sum, s) =>

 sum + (s.leaveStaff.includes(staffId) ? 1 : 0), 0

)

 // 4. 전체 직원 평균과 비교

 const allStaff = await prisma.staff.findMany({ where: { isActive: true } })

 const avgNightShift = await getAvgNightShift(month)

 const avgWeekend = await getAvgWeekend(month)

 const avgLeave = await getAvgLeave(month)

 // 5. 편차 계산

 const nightShiftDeviation = Math.abs(nightShiftCount - avgNightShift)

 const weekendDeviation = Math.abs(weekendCount - avgWeekend)

 const leaveDeviation = Math.abs(leaveCount - avgLeave)

 // 6. 종합 점수 (낮을수록 좋음)

 const score =

 nightShiftDeviation * 2.0 + // 야간 가중치 높음

 weekendDeviation * 1.5 +

 leaveDeviation * 1.0

// 7. 등급 판정

```
let grade: 'EXCELLENT' | 'GOOD' | 'FAIR' | 'POOR'
if (score < 0.5) grade = 'EXCELLENT'
else if (score < 1.0) grade = 'GOOD'
else if (score < 1.5) grade = 'FAIR'
else grade = 'POOR'

return {
  staffId,
  month,
  nightShiftCount,
  weekendCount,
  leaveCount,
  nightShiftDeviation,
  weekendDeviation,
  leaveDeviation,
  totalScore: score,
  grade
}
```

5. 개발 단계별 실행 계획

5.1 전체 개발 일정 (10주)

10주 개발 로드맵

Week 1-2: 기초 설정 + 핵심 UI (2주)
Week 3: 원장 패턴 저장 (1주) ★
Week 4-5: 연차 자율 신청 시스템 (2주) ★ 최우선
Week 6: 월간/주간 자동 배치 (1주) ★
Week 7: 알림 시스템 (1주) ★
Week 8: 기존 기능 통합 + 테스트 (1주)
Week 9: 성능 최적화 + 완성도 (1주)
Week 10: 배포 + 교육 (1주)

5.2 주차별 상세 계획

Week 1-2: 기초 설정 및 핵심 UI (2주)

목표: 프로젝트 세팅 + 기본 화면 구현

✅ 완료 기준:

- Next.js 프로젝트 생성 및 기본 설정
- Prisma 스키마 정의 (★ 신규 테이블 5개 포함)
- 로그인 화면
- 대시보드 레이아웃 (헤더, 사이드바)
- 달력뷰 기본 구조

📁 작업 파일:

- package.json (의존성 설정)
- next.config.js
- tailwind.config.ts
- prisma/schema.prisma ★
- src/app/layout.tsx
- src/app/(auth)/login/page.tsx
- src/app/(dashboard)/layout.tsx
- src/app/(dashboard)/calendar/page.tsx
- src/components/layout/*
- src/components/calendar/CalendarView.tsx

🎯 핵심 작업:

1. 프로젝트 초기화

```
bash

npx create-next-app@latest yonsei-dental-schedule --typescript --tailwind --app
cd yonsei-dental-schedule
npm install prisma @prisma/client next-auth zod date-fns
npx prisma init
```

2. Prisma 스키마 정의 (신규 테이블 포함)

- User, Staff, Doctor (기존)
- Schedule, Settings (기존)
- DoctorPattern ★
- LeaveApplicationPeriod ★
- StaffLeaveApplication ★
- LeaveSlotStatus ★
- Notification ★

3. NextAuth.js 설정

- Credentials Provider

- JWT 세션

4. 기본 레이아웃

- Header (로고, 사용자 정보, 알림 벨)
- Sidebar (메뉴)
- Footer

5. 달력 그리드 구현

- 월간 뷰 (7x5 그리드)
- 날짜 셀 기본 구조

Week 3: 원장 패턴 저장 기능 (1주) ★

목표: 요일별 원장 패턴 저장 및 적용

✅ 완료 기준:

- 원장 패턴 설정 페이지 완성
- 패턴 저장/수정/삭제 기능
- 달력에 패턴 적용 기능 (1클릭 30일 세팅)

📁 작업 파일:

```
- src/app/(dashboard)/settings/doctors/pattern/page.tsx ★
- src/components/pattern/DayPatternEditor.tsx ★
- src/components/pattern/PatternPreview.tsx ★
- src/components/pattern/PatternList.tsx ★
- src/app/api/doctor-pattern/route.ts ★
- src/app/api/doctor-pattern/[id]/route.ts ★
- src/app/api/doctor-pattern/apply/route.ts ★
- src/lib/services/pattern-service.ts ★
- src/lib/hooks/use-pattern.ts ★
```

🎯 핵심 작업:

1. 요일별 원장 선택 UI

- 월~토 각 요일별 탭
- 원장 체크박스 선택
- 야간 진료 토글
- 저장 버튼

2. 패턴 저장 API

typescript

```
POST /api/doctor-pattern
Body: {
  dayOfWeek: 1, // 월요일
  doctors: ['id1', 'id2'],
  hasNightShift: true
}
```

3. 패턴 적용 버튼 (달력 화면)

- [요일 패턴 적용] 버튼
- 한 달 전체 스케줄 생성
- 기존 스케줄 덮어쓰기 확인
- 공휴일 자동 제외

4. 30일 일괄 스케줄 생성 로직

typescript

```
async function applyPatternToMonth(month: Date) {
  const patterns = await getPatterns()
  const dates = getAllDatesInMonth(month)

  for (const date of dates) {
    const dayOfWeek = getDay(date)
    const pattern = patterns[dayOfWeek]

    await createOrUpdateSchedule({
      date,
      doctors: pattern.doctors,
      hasNightShift: pattern.hasNightShift
    })
  }
}
```

Week 4-5: 연차 자율 신청 시스템 (2주) ★ 최우선

목표: 직원 자율 신청 + 관리자 관리 페이지

✅ 완료 기준:

- 신청 링크 생성 기능
- 직원용 신청 페이지 (로그인 없음)
- 실시간 슬롯 관리 (동시성 제어)

- 관리자 연차관리 페이지 (대시보드, 달력뷰, 목록뷰, 직원별뷰)
- 확정 기능 (Schedule DB 반영)
- Web Share API 연동

📁 작업 파일 (많음):

직원용 신청 페이지:

- src/app/(public)/leave-apply/[token]/page.tsx ★
- src/app/(public)/leave-apply/[token]/success/page.tsx
- src/app/(public)/layout.tsx
- src/components/public-apply/DateSelector.tsx ★
- src/components/public-apply/TypeSelector.tsx ★
- src/components/public-apply/RealTimeStatus.tsx ★
- src/components/public-apply/ApplicationForm.tsx ★
- src/components/public-apply/ConfirmationModal.tsx ★

관리자 연차관리:

- src/app/(dashboard)/leave-management/page.tsx ★
- src/app/(dashboard)/leave-management/calendar-view/page.tsx ★
- src/app/(dashboard)/leave-management/list-view/page.tsx ★
- src/app/(dashboard)/leave-management/staff-view/page.tsx ★
- src/app/(dashboard)/leave-management/period-setup/page.tsx ★
- src/components/leave-management/Dashboard.tsx ★
- src/components/leave-management/CalendarView.tsx ★
- src/components/leave-management/ListView.tsx ★
- src/components/leave-management/StaffView.tsx ★
- src/components/leave-management/PeriodSetup.tsx ★
- src/components/leave-management/ConfirmModal.tsx ★

API:

- src/app/api/leave-apply/[token]/route.ts ★
- src/app/api/leave-apply/[token]/submit/route.ts ★
- src/app/api/leave-apply/[token]/status/route.ts ★
- src/app/api/leave-management/route.ts ★
- src/app/api/leave-management/period/route.ts ★
- src/app/api/leave-management/period/[id]/confirm/route.ts ★

서비스 & 훅:

- src/lib/services/leave-service.ts ★
- src/lib/hooks/use-leave-management.ts ★

🎯 핵심 작업:

Day 1-3: 신청 기간 설정 및 링크 생성

// 1. 신청 기간 생성 UI

- 대상 월 선택
- 신청 기간 (시작일~종료일)
- 날짜별 슬롯 설정 (평일 3명, 주말 1명)
- [링크 생성] 버튼

// 2. 링크 생성 API

POST /api/leave-management/period

Body: {

targetMonth: '2025-02-01',

startDate: '2025-01-20',

endDate: '2025-01-23',

maxDaysPerPerson: 10,

allowTypes: ['annual', 'off'],

slotSettings: {

weekday: 3,

weekend: 1

}

}

Response: {

id: 'xxx',

linkToken: 'abc123',

shareableLink: 'https://domain.com/leave-apply/abc123'

}

// 3. Web Share API 연동

<Button onClick={async () => {

if (navigator.share) {

await navigator.share({

title: '연차 신청',

text: '2월 연차를 신청해주세요',

url: shareableLink

})

} else {

// PC: 클립보드 복사

navigator.clipboard.writeText(shareableLink)

toast('링크가 복사되었습니다')

}

>>>

{isMobile ? '공유하기' : '링크 복사'}

</Button>

Day 4-7: 직원용 신청 페이지

typescript

// 1. 토큰 기반 접근 제어

- URL에서 token 추출
- DB에서 LeaveApplicationPeriod 조회
- status === 'open' 확인
- 마감 시간 확인

// 2. 실시간 슬롯 상태 (3초마다 새로고침)

```
useEffect(() => {  
  const interval = setInterval(async () => {  
    const status = await fetch(`/api/leave-apply/${token}/status`)  
    setSlots(await status.json())  
  }, 3000)  
  
  return () => clearInterval(interval)  
}, [token])
```

// 3. 날짜 선택 UI

```
<DateSelector  
  availableDates={slots.filter(s => !s.isFull)}  
  selectedDates={selected}  
  onChange={setSelected}  
</>
```

// 4. 신청 제출 (트랜잭션)

```
POST /api/leave-apply/${token}/submit  
Body: {  
  staffName: '김철수',  
  applications: [  
    { date: '2025-02-05', type: 'annual' },  
    { date: '2025-02-06', type: 'off' }  
  ]  
}
```

// 트랜잭션으로 슬롯 동시성 제어

```
await prisma.$transaction(async (tx) => {  
  // 슬롯 확인 (비관적 잠금)  
  const slot = await tx.leaveSlotStatus.findUnique({  
    where: { ... },  
    lock: 'pessimistic_write'  
  })  
  
  if (slot.isFull) throw new Error('마감')
```

// 슬롯 업데이트

```
await tx.leaveSlotStatus.update({  
  data: { usedSlots: { increment: 1 } }  
})
```

```
})
```

```
// 신청 저장
```

```
await tx.staffLeaveApplication.create({ ... })
```

```
// 알림 생성
```

```
await tx.notification.create({ ... })
```

```
})
```

Day 8-10: 관리자 연차관리 페이지

typescript

```
// 1. 대시보드
```

- 전체 신청 **현황** (18명 신청 / 20명 중)
- 총 신청 **건수** (45건)
- 날짜별 **현황** (그래프)
- 최근 신청 목록

```
// 2. 달력뷰
```

- 날짜별 신청자 명단
- 슬롯 **현황** (5/6)
- 색상 **코드** (여유/마감)
- 클릭 시 상세 팝업

```
// 3. 목록뷰
```

- 전체 신청 **목록** (테이블)
- **필터** (날짜, 유형, 직원)
- **정렬** (날짜순, 이름순)
- 수정/삭제 버튼

```
// 4. 직원별뷰
```

- 직원별 신청 내역
- 총 신청 일수
- 승인 대기 / 확정

Day 11-14: 확정 기능

typescript

// [확정] 버튼 클릭

POST /api/leave-management/period/\${id}/confirm

// 서버 로직:

```
async function confirmPeriod(periodId: string) {
  await prisma.$transaction(async (tx) => {
    // 1. 신청 기간 상태 변경
    await tx.leaveApplicationPeriod.update({
      where: { id: periodId },
      data: {
        status: 'confirmed',
        confirmedAt: new Date()
      }
    })

    // 2. 모든 신청 내역 조회
    const applications = await tx.staffLeaveApplication.findMany({
      where: { periodId }
    })

    // 3. Schedule DB에 연차 정보 반영
    for (const app of applications) {
      for (const leave of app.applications) {
        await tx.schedule.update({
          where: {
            date: leave.date,
            /* targetMonth 조건 */
          },
          data: {
            leaveCount: { increment: 1 },
            leaveStaff: { push: app.staffName }
          }
        })
      }
    }

    // 4. 직원 신청 페이지 닫기 (status 확인)
    // 이제 신청 페이지 접속 시 "신청이 마감되었습니다" 표시
  })
}
```

Week 6: 월간/주간 자동 배치 (1주) ★

목표: 한 번에 여러 날짜 배치

✅ 완료 기준:

- 월간 자동 배치 기능
- 주간 자동 배치 기능
- 스마트 배치 vs 완전 재배포 옵션
- 진행률 실시간 표시

📁 작업 파일:

- src/components/schedule/AutoAssignModal.tsx ★ 수정
- src/app/api/auto-assign/monthly/route.ts ★
- src/app/api/auto-assign/weekly/route.ts ★
- src/lib/algorithms/monthly-assign.ts ★
- src/lib/algorithms/weekly-assign.ts ★

🎯 핵심 작업:

1. 배치 옵션 UI

typescript

// 배치 범위

- 월간 배치 (전체)
- 주간 배치 (특정 주)
- 일별 배치 (특정일)

// 배치 방식

- 스마트 배치 (기존 유지)
- 완전 재배포

// 고려 사항

- 연차 제외
- 형평성 우선

2. 월간 배치 알고리즘

- 30일치 일괄 처리
- 연차 직원 자동 제외
- 형평성 최적화
- 10~30초 내 완료

3. 진행률 표시 (SSE)

typescript

// 서버

```
async function* assignWithProgress() {  
  for (let i = 0; i < dates.length; i++) {  
    await assignDate(dates[i])  
    yield { progress: (i + 1) / dates.length * 100 }  
  }  
}
```

// 클라이언트

```
const eventSource = new EventSource('/api/auto-assign/monthly')  
eventSource.onmessage = (e) => {  
  setProgress(JSON.parse(e.data).progress)  
}
```

Week 7: 알림 시스템 (1주) ★

목표: 프로그램 내 실시간 알림

✅ 완료 기준:

- 알림 센터 (헤더 🛎 아이콘)
- 실시간 알림 푸시 (SSE)
- 알림 유형별 처리
- 읽음/삭제 기능

📁 작업 파일:

- src/components/layout/NotificationBell.tsx ★
- src/components/notifications/NotificationPanel.tsx ★
- src/components/notifications/NotificationItem.tsx ★
- src/app/(dashboard)/notifications/page.tsx ★
- src/app/api/notifications/route.ts ★
- src/app/api/notifications/unread/route.ts ★
- src/lib/services/notification-service.ts ★
- src/lib/hooks/use-notifications.ts ★

🎯 핵심 작업:

1. 알림 벨 아이콘

tsx

```

<button onClick={togglePanel}>
  <Bell />
  {unreadCount > 0 && (
    <Badge>{unreadCount}</Badge>
  )}
</button>

```

2. 실시간 푸시 (SSE 권장)

typescript

```

// 서버
GET /api/notifications/stream

// 클라이언트
useEffect() => {
  const eventSource = new EventSource('/api/notifications/stream')

  eventSource.onmessage = (event) => {
    const notification = JSON.parse(event.data)
    addNotification(notification)
    playSound()
  }

  return () => eventSource.close()
}, [])

```

3. 알림 유형

- leave_apply: 연차 신청
- slot_full: 슬롯 마감
- deadline_alert: 마감 임박
- assign_complete: 배치 완료

Week 8: 기존 기능 통합 및 테스트 (1주)

목표: 기존 기능 + 신규 기능 통합

✅ 완료 기준:

- 스케줄 편집 기능 (기존)
- 통계 페이지 (기존)
- 설정 페이지들 (기존 + 수정)

- 전체 기능 연동 테스트
- 버그 수정

핵심 작업:

1. 연차 확정 → 스케줄 반영 연동 테스트
 2. 자동 배치 → 형평성 계산 연동 테스트
 3. 전체 워크플로우 E2E 테스트
 4. 버그 수집 및 수정
-

Week 9: 성능 최적화 및 완성도 (1주)

목표: 속도 개선 + UX 향상

완료 기준:

- 월간 배치 속도 < 30초
- 실시간 슬롯 동시성 안정성
- 모바일 반응형 완성
- 로딩/에러 처리 완벽

핵심 작업:

1. DB 쿼리 최적화

- 인덱스 추가
- N+1 쿼리 제거
- 페이지네이션

2. 트랜잭션 성능

- 비관적 잠금 최소화
- 배치 처리

3. 프론트엔드 최적화

- React.memo
- useMemo, useCallback
- 코드 스플리팅

4. 에러 처리

- Error Boundary
- Toast 알림

- 로그 수집

Week 10: 배포 및 교육 (1주)

목표: 실전 투입

✅ 완료 기준:

- Vercel 배포 완료
- 실제 데이터 입력
- 사용자 교육 완료
- 공식 오픈

🎯 핵심 작업:

1. Vercel 배포

```
bash  
  
npm run build  
vercel --prod
```

2. DB 마이그레이션

```
bash  
  
npx prisma migrate deploy
```

3. 초기 데이터 입력

- 직원 정보
- 원장 정보
- 근무 규칙
- 휴업일

4. 사용자 매뉴얼 작성

5. 교육 진행 (2시간)

6. 공식 오픈

6. Prisma 스키마 설계

6.1 완전한 Prisma 스키마 (신규 테이블 포함)

```
prisma
```

```
// prisma/schema.prisma
```

```
generator client {  
  provider = "prisma-client-js"  
}
```

```
datasource db {  
  provider = "postgresql"  
  url      = env("DATABASE_URL")  
}
```

```
// =====
```

```
// 사용자 및 인증
```

```
// =====
```

```
model User {  
  id      String  @id @default(cuid())  
  email   String  @unique  
  password String  
  name    String  
  role    Role    @default(USER)  
  isActive Boolean @default(true)  
  createdAt DateTime @default(now())  
  updatedAt DateTime @updatedAt
```

```
  notifications Notification[]
```

```
  @@map("users")  
}
```

```
enum Role {  
  ADMIN  
  USER  
}
```

```
// =====
```

```
// 직원 관리
```

```
// =====
```

```
model Staff {  
  id      String  @id @default(cuid())  
  name    String  
  grade   Grade  
  phoneNumber String?  
  email   String?  
  hireDate DateTime
```

```

isActive      Boolean @default(true)
workDaysPerWeek Int    @default(5)

createdAt DateTime @default(now())
updatedAt DateTime @updatedAt

schedules Schedule[] @relation("ScheduleStaff")

@@map("staff")
}

enum Grade {
    MANAGER    // 실장급 (팀장)
    SENIOR     // 고년차
    MIDDLE     // 중년차
    JUNIOR     // 저년차
}

// =====
// 원장 관리
// =====

model Doctor {
    id      String @id @default(cuid())
    name    String
    specialty String?
    isActive Boolean @default(true)

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt

    schedules Schedule[] @relation("ScheduleDoctors")

    @@map("doctors")
}

// ★ 신규: 원장 요일별 패턴
model DoctorPattern {
    id      String @id @default(cuid())
    dayOfWeek Int    // 0=일요일, 1=월요일, ..., 6=토요일
    doctors  String[] // Doctor ID 배열
    doctorMode String // "박(진료)+구+윤" (표시용)
    hasNightShift Boolean @default(false)
    isActive Boolean @default(true)

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt

```

```

    @@unique([dayOfWeek])
    @@map("doctor_patterns")
}

// =====
// 스케줄 관리
// =====

model Schedule {
    id          String  @id @default(cuid())
    date        DateTime @unique

    // 원장 정보
    doctors      Doctor[] @relation("ScheduleDoctors")
    doctorMode    String  // "박(진료)+구+윤"
    hasNightShift Boolean @default(false)

    // 필요 인원
    requiredManager Int    @default(0)
    requiredSenior  Int    @default(0)
    requiredMiddle  Int    @default(0)
    requiredJunior  Int    @default(0)

    // 배치된 직원
    staff          Staff[] @relation("ScheduleStaff")

    // 연차 정보 ★
    leaveCount     Int     @default(0) // 연차 사용 인원
    leaveStaff      String[] @default([]) // 연차 사용 직원 명단

    // 계산 필드
    availableLeaveSlots Int @default(0) // 쉴 수 있는 인원
    actualAvailable   Int  @default(0) // 실제 배치 가능 인원

    // 상태
    isComplete      Boolean @default(false)
    isHoliday        Boolean @default(false)

    createdAt DateTime @default(now())
    updatedAt DateTime @updatedAt

    @@index([date])
    @@map("schedules")
}

// =====

```

// ☆ 신규: 연차 관리

// =====

// 연차 신청 기간

```
model LeaveApplicationPeriod {
  id          String  @id @default(cuid())
  targetMonth  DateTime // 2025-01-01 (대상 월)
  startDate    DateTime // 신청 시작일
  endDate      DateTime // 신청 마감일
  status       PeriodStatus @default(DRAFT)

  maxDaysPerPerson Int    @default(10)
  allowTypes       String[] @default(["annual", "off", "half"])

  shareableLink  String  @unique
  linkToken      String  @unique @default(cuid())

  createdBy      String
  createdAt      DateTime @default(now())
  confirmedAt    DateTime?
  confirmedBy    String?

  applications    StaffLeaveApplication[]
  slotStatuses    LeaveSlotStatus[]

  @@index([linkToken])
  @@index([targetMonth])
  @@map("leave_application_periods")
}
```

```
enum PeriodStatus {
  DRAFT    // 초안
  OPEN     // 신청 오픈
  CLOSED   // 신청 마감
  CONFIRMED // 확정 완료
}
```

// 직원 연차 신청

```
model StaffLeaveApplication {
  id          String  @id @default(cuid())
  period      LeaveApplicationPeriod @relation(fields: [periodId], references: [id], onDelete: Cascade)
  periodId    String

  staffName  String // 로그인 없으므로 이름으로 식별
  applications Json  // { date: Date, type: 'annual' | 'off' | 'half' }[]

  submittedAt DateTime @default(now())
}
```

```

ipAddress String

isEdited Boolean @default(false)
editedAt DateTime?

source ApplicationSource @default(STAFF)
memo String?

@@index([periodId])
@@index([staffName])
@@map("staff_leave_applications")
}

enum ApplicationSource {
  STAFF // 직원이 직접 신청
  ADMIN // 관리자가 수동 입력
  IMPORT // 엑셀 등에서 가져오기
}

// 날짜별 슬롯 현황
model LeaveSlotStatus {
  id String @id @default(cuid())
  period LeaveApplicationPeriod @relation(fields: [periodId], references: [id], onDelete: Cascade)
  periodId String

  date DateTime
  maxSlots Int // 월 수 있는 최대 인원
  usedSlots Int @default(0)
  remainingSlots Int // maxSlots - usedSlots
  isFull Boolean @default(false)

  applicants String[] @default([]) // 신청자 명단

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  @@unique([periodId, date])
  @@index([periodId])
  @@index([date])
  @@map("leave_slot_statuses")
}

// =====
// ★ 신규: 알림
// =====

model Notification {

```

```

id      String  @id @default(cuid())
user    User    @relation(fields: [userId], references: [id], onDelete: Cascade)
userId  String

type    NotificationType
title   String
content String
data    Json?   // 추가 데이터 (링크 등)

isRead  Boolean @default(false)
readAt  DateTime?

createdAt DateTime @default(now())

@@index([userId, isRead])
@@index([createdAt])
@@map("notifications")
}

enum NotificationType {
    LEAVE_APPLY    // 연차 신청
    SLOT_FULL      // 슬롯 마감
    DEADLINE_ALERT // 마감 임박
    ASSIGN_COMPLETE // 배치 완료
    SYSTEM          // 시스템 알림
}

// =====
// 설정 관리
// =====

model Settings {
    id      String  @id @default(cuid())

    // 근무 규칙
    workDaysPerWeek Int    @default(5)

    // 등급별 비율
    managerRatio    Float  @default(0.2)
    seniorRatio     Float  @default(0.3)
    middleRatio     Float  @default(0.3)
    juniorRatio     Float  @default(0.2)

    // 배치 규칙
    mustHaveManager Boolean @default(true)

    // 형평성 규칙

```

```
maxConsecutiveWork Int @default(5)
```

```
// ✕ 제거: 연차-휴무 붙이기 관련 설정
```

```
// ✕ 제거: 연속 휴무 최소 일수
```

```
createdAt DateTime @default(now())
```

```
updatedAt DateTime @updatedAt
```

```
@@map("settings")
```

```
}
```

```
// =====
```

```
// 휴업일 관리
```

```
// =====
```

```
model Holiday {
```

```
  id      String @id @default(cuid())
```

```
  date    DateTime @unique
```

```
  name    String
```

```
  isRecurring Boolean @default(false)
```

```
  createdAt DateTime @default(now())
```

```
  updatedAt DateTime @updatedAt
```

```
  @@map("holidays")
```

```
}
```

```
// =====
```

```
// 활동 로그
```

```
// =====
```

```
model ActivityLog {
```

```
  id      String @id @default(cuid())
```

```
  userId  String
```

```
  action  String
```

```
  targetType String?
```

```
  targetId String?
```

```
  details Json?
```

```
  ipAddress String?
```

```
  createdAt DateTime @default(now())
```

```
  @@index([userId])
```

```
  @@index([createdAt])
```

```
  @@map("activity_logs")
```

```
}
```

7. 핵심 API 명세

7.1 연차 신청 API (직원용)

GET /api/leave-apply/[token]

설명: 신청 페이지 데이터 조회 인증: 불필요 (토큰으로 식별) 응답:

```
json
{
  "period": {
    "id": "xxx",
    "targetMonth": "2025-02-01",
    "startDate": "2025-01-20",
    "endDate": "2025-01-23",
    "status": "open",
    "maxDaysPerPerson": 10
  },
  "slots": [
    {
      "date": "2025-02-05",
      "maxSlots": 3,
      "usedSlots": 2,
      "remainingSlots": 1,
      "isFull": false
    }
  ]
}
```

POST /api/leave-apply/[token]/submit

설명: 연차 신청 제출 Body:

```
json
{
  "staffName": "김철수",
  "applications": [
    { "date": "2025-02-05", "type": "annual" },
    { "date": "2025-02-06", "type": "off" }
  ]
}
```

응답:

```
json
```

```
{
  "success": true,
  "applicationId": "xxx",
  "message": "신청이 완료되었습니다"
}
```

GET /api/leave-apply/[token]/status

설명: 실시간 슬롯 현황 (폴링용) **응답:** 위와 동일

7.2 연차 관리 API (관리자용)

POST /api/leave-management/period

설명: 신청 기간 생성 **인증:** 필요 **Body:**

```
json
{
  "targetMonth": "2025-02-01",
  "startDate": "2025-01-20",
  "endDate": "2025-01-23",
  "maxDaysPerPerson": 10,
  "allowTypes": ["annual", "off"],
  "slotSettings": {
    "weekday": 3,
    "weekend": 1
  }
}
```

응답:

```
json
{
  "id": "xxx",
  "linkToken": "abc123",
  "shareableLink": "https://domain.com/leave-apply/abc123"
}
```

POST /api/leave-management/period/[id]/confirm

설명: 연차 신청 확정 (Schedule DB 반영) **인증:** 필요 **응답:**

```
json
```

```
{
  "success": true,
  "confirmedCount": 45,
  "message": "확정이 완료되었습니다"
}
```

7.3 자동 배치 API

POST /api/auto-assign/monthly

설명: 월간 자동 배치 인증: 필요 Body:

json

```
{
  "targetMonth": "2025-02-01",
  "mode": "smart",
  "considerLeave": true
}
```

응답 (SSE 스트림):

```
event: progress
data: {"progress": 30, "message": "15/30 완료"}

event: complete
data: {"success": true, "assigned": 30}
```

8. 구현 우선순위별 상세 가이드

8.1 최우선 구현 항목 (Week 3-5)

1순위: 원장 패턴 저장/적용 (Week 3)

- 시간 절감 효과 최대
- 다른 기능의 기반

2순위: 연차 자율 신청 시스템 (Week 4-5)

- 핵심 기능
- 사용자 경험 개선 최대
- 가장 복잡하므로 충분한 시간 필요

3순위: 월간 자동 배치 (Week 6)

- 효율성 향상
- 연차 정보 활용

4순위: 알림 시스템 (Week 7)

- 실시간 현황 파악
- 사용 편의성










8.2 구현 순서 권장사항

근거: 의존성 및 복잡도 고려












1. **Week 1-2:** 기초 작업은 반드시 먼저
2. **Week 3:** 원장 패턴은 독립적이므로 먼저 구현 가능
3. **Week 4-5:** 연차 신청 시스템은 가장 복잡하므로 2주 확보
4. **Week 6:** 자동 배치는 연차 정보를 활용하므로 그 다음
5. **Week 7:** 알림은 다른 기능들이 완성된 후
6. **Week 8-10:** 통합, 최적화, 배포

최종 체크리스트

문서 완성도 확인

-  프로젝트 전체 분석 완료
-  누락된 페이지/기능 모두 파악
-  전체 파일 구조 정의
-  데이터 흐름 명확화
-  비즈니스 로직 상세화
-  Prisma 스키마 완성 (신규 5개 테이블 포함)
-  API 엔드포인트 전체 정의
-  10주 개발 계획 수립
-  주차별 상세 가이드 작성

추가된 핵심 기능

-  원장 요일별 패턴 저장
-  연차 자율 신청 시스템
 -  신청 링크 생성
 -  직원용 신청 페이지 (로그인 불필요)
 -  실시간 슬롯 관리 (동시성 제어)
 -  관리자 확정 기능
-  연차관리 전용 페이지
 -  대시보드
 -  달력뷰
 -  목록뷰
 -  직원별뷰

- ✓ 월간/주간 자동 배치
- ✓ 알림 센터
- ✓ Web Share API 연동

✓ 제거된 불필요 기능

- ✓ 연차 입력 모드 (드래그) 제거
- ✓ 연차-휴무 붙이기 설정 제거
- ✓ 연속 휴무 설정 제거
- ✓ 이메일 알림 제거
- ✓ 복잡한 패턴 분석 제거

다음 단계

이 문서를 기반으로 다음 작업을 진행할 수 있습니다:

1. **Prisma 스키마 파일 생성** (즉시 시작 가능)
2. **특정 기능 상세 코드 구현**
 - 연차 자율 신청 시스템 전체
 - 월간 자동 배치 알고리즘
3. **핵심 페이지 컴포넌트 생성**
 - 연차관리 대시보드
 - 직원용 신청 페이지

문서 버전: 3.0 (최종)

작성 완료: 2025-10-22

확실성 수준: [확인됨]

작업 상태:  완료

이 문서는 연세바로치과 스케줄 관리 시스템의 완전한 개발 가이드입니다.