연세바로치과 스케줄 관리 시스템 - 기능 명세서 Part 2

문서 버전: 2.1 (최종)
작성일: 2025-10-21
대상: 백엔드/프론트엔드 개발자
Part: 2/2 (섹션 3.3.3~11)

Part 2 목차

이전 문서: 기능명세서 Part 1

---

3. 핵심 기능 명세 (계속)

3.3.3 직원 신청 페이지 (외부 링크)

URL 구조:

```
https://dental-schedule.com/apply/{token}
```

페이지 구성:

tsx

```
// src/app/apply/[token]/page.tsx
```

인증 성공 후:

tsx

🏥 연세바로치과 연차/오프 신청

✅ 인증 완료: 김철수님

📅 신청 가능 기간: 2025-01-20 ~ 01-23
⏰ 남은 시간: 2일 14시간 35분

⚙️ [PIN 번호 설정하기] 👤 [내 정보] 🚪 [로그아웃]

📋 신청 현황:

날짜	요일	슬롯 현황	상태		
2월 1일	토	●●○ 2/3	[신청]		
2월 2일	일	●○○ 1/3	[신청]		
2월 3일	월	●●● 3/3	마감		
2월 4일	화	●○○ 1/3	[신청]		
2월 5일	수	●●○ 2/3	✅신청완료		
2월 6일	목	○○○ 0/3	[신청]		
...	...	...	...		

✅ 내 신청 내역:
• 2월 5일 (수) - 연차
신청일시: 2025-01-21 14:23
[❌ 취소하기]

💡 신청한 날짜는 언제든 취소할 수 있습니다

📊 내 연차 현황:
• 총 연차: 15일
• 사용: 3일
• 잔여: 12일

```
|                                                    |
└────────────────────────────────────────────────────┘
```

PIN 번호 설정 모달:

```tsx
┌──────────────────────────────────────────────────┐
│  🔐 PIN 번호 설정                                │
├──────────────────────────────────────────────────┤
│                                                  │
│  다음부터는 PIN 번호로 빠르게 인증할 수 있습니다!  │
│                                                  │
│  새 PIN 번호 (4~6자리 숫자):                     │
│  ┌──────────────────────────┐                    │
│  │ [_ _ _ _ _ _]            │                    │
│  └──────────────────────────┘                    │
│                                                  │
│  PIN 번호 확인:                                  │
│  ┌──────────────────────────┐                    │
│  │ [_ _ _ _ _ _]            │                    │
│  └──────────────────────────┘                    │
│                                                  │
│  ⚠️ 중요:                                        │
│  • PIN 번호는 숫자만 입력 가능합니다             │
│  • PIN을 잊으면 생년월일로 인증할 수 있습니다    │
│  • 관리자는 PIN을 볼 수 없습니다 (암호화 저장)   │
│                                                  │
│  [나중에 하기]  [설정 완료]                      │
│                                                  │
└──────────────────────────────────────────────────┘
```

신청 목록 (휴일 처리 포함):

```tsx
```

```
┌──────────────────────────────────────────────────────────────────────────────┐
│ 📋 신청 현황:                                                                │
│                                                                              │
│ ⚠️ 오프는 주 2일까지만 신청 가능합니다 (주4일제 기준)                        │
│                                                                              │
│ ┌──────────┬──────┬──────────┬──────────┬──────┬──────┐                      │
│ │ 날짜     │ 요일 │ 슬롯 현황 │ 상태     │      │      │                      │
│ ├──────────┼──────┼──────────┼──────────┼──────┼──────┤                      │
│ │ 2월 1일  │ 토   │ ●●○ 2/3  │ [신청]   │      │      │                      │
│ │ 2월 2일  │ 일   │ 🔒 휴무일 │ 신청불가 │      │      │                      │
│ │ 2월 3일  │ 월   │ ●●● 3/3  │ 마감     │      │      │                      │
│ │ 2월 4일  │ 화   │ ●○○ 1/3  │ [신청]   │      │      │                      │
│ │ 2월 5일  │ 수   │ ●●○ 2/3  │ ✅신청완료 │      │      │                      │
│ │ 2월 6일  │ 목   │ ○○○ 0/3  │ [신청]   │      │      │                      │
│ │ 2월 9일  │ 일   │ 🔒 휴무일 │ 신청불가 │      │      │                      │
│ │ 2월 10일 │ 월   │ 🔒 설날연휴 │ 신청불가 │      │      │                      │
│ │ ...      │ ...  │ ...      │ ...      │      │      │                      │
│ └──────────┴──────┴──────────┴──────────┴──────┴──────┘                      │
│                                                                              │
│ 💡 일요일과 공휴일은 자동으로 휴무일입니다                                   │
│                                                                              │
│ ──────────────────────────────────────────────────────────────────────────── │
│                                                                              │
│ ✅ 내 신청 내역:                                                             │
│ • 2월 5일 (수) - 연차                                                        │
│   신청일시: 2025-01-21 14:23                                                │
│   [❌ 취소하기]                                                              │
│                                                                              │
│ 📊 내 오프 신청 현황:                                                        │
│ • 이번 주 (2/3~2/9): 1일 신청 (1일 더 가능)                                  │
│ • 다음 주 (2/10~2/16): 0일 신청 (2일 가능)                                   │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘
```

신청 확인 다이얼로그:

tsx

```
┌──────────────────────────────────────────────────────────────┐
│ ✅ 연차/오프 신청 확인                                       │
├──────────────────────────────────────────────────────────────┤
│                                                              │
│ 다음 날짜에 신청하시겠습니까?                                │
│                                                              │
│ 📅 날짜: 2025년 2월 5일 (수)                                 │
│ 📝 유형: ⚫ 연차  ⚪ 오프                                     │
│                                                              │
│ ⚠️ 오프 신청 시 주의사항:                                    │
│ • 주 2일까지만 신청 가능 (주4일제 기준)                      │
│ • 이번 주 (2/3~2/9): 1일 신청 완료                           │
│ • 이 신청으로 이번 주 총 2일 (한도 충족)                     │
│                                                              │
│ ⚠️ 신청 후에도 마감 전까지 취소할 수 있습니다                │
│                                                              │
│ [취소]  [신청하기]                                           │
│                                                              │
└──────────────────────────────────────────────────────────────┘
```

실시간 업데이트:

```typescript
```

```tsx
// src/app/apply/[token]/page.tsx

'use client'

import { useEffect, useState } from 'react'
import { useParams } from 'next/navigation'

interface SlotStatus {
  date: string
  dayOfWeek: number
  current: number
  max: number
  isFull: boolean
  isHoliday: boolean  // ⭐ 휴일 여부
  holidayName?: string  // ⭐ 공휴일명
}

interface WeeklyOffCount {
  weekStart: string
  weekEnd: string
  count: number
  maxAllowed: number
}

export default function ApplyPage() {
  const { token } = useParams()
  const [slotStatus, setSlotStatus] = useState<SlotStatus[]>([])
  const [weeklyOffCounts, setWeeklyOffCounts] = useState<WeeklyOffCount[]>([])
  const [selectedType, setSelectedType] = useState<'ANNUAL' | 'OFF'>('ANNUAL')

  // 실시간 슬롯 현황 업데이트 (3초마다)
  useEffect(() => {
    const fetchStatus = async () => {
      const res = await fetch(`/api/apply/status/${token}`)
      const data = await res.json()

      setSlotStatus(data.slotStatus)
      setWeeklyOffCounts(data.weeklyOffCounts)
    }

    fetchStatus()
    const interval = setInterval(fetchStatus, 3000)

    return () => clearInterval(interval)
  }, [token])
```

```
  // 신청 가능 여부 체크
  const canApply = (slot: SlotStatus): {
    canApply: boolean
    reason?: string
  } => {
    // 휴일 체크
    if (slot.isHoliday) {
      return {
        canApply: false,
        reason: slot.holidayName || '휴무일',
      }
    }

    // 슬롯 마감 체크
    if (slot.isFull) {
      return {
        canApply: false,
        reason: '마감',
      }
    }

    // 오프 주간 제한 체크
    if (selectedType === 'OFF') {
      const weekInfo = getWeekInfo(slot.date)
      const weeklyCount = weeklyOffCounts.find(w =>
        w.weekStart === weekInfo.start && w.weekEnd === weekInfo.end
      )

      if (weeklyCount && weeklyCount.count >= weeklyCount.maxAllowed) {
        return {
          canApply: false,
          reason: `이번 주 오프 한도 초과 (${weeklyCount.count}/${weeklyCount.maxAllowed})`,
        }
      }
    }

    return { canApply: true }
  }

  const handleApply = async (date: string) => {
    const slot = slotStatus.find(s => s.date === date)

    if (!slot) return

    const check = canApply(slot)

    if (!check.canApply) {
```

```
      alert(check.reason)
      return
    }

    // 확인 다이얼로그
    const confirmed = await showConfirmDialog({
      date: slot.date,
      type: selectedType,
      weeklyOffCount: selectedType === 'OFF'
        ? getWeeklyOffCount(slot.date)
        : undefined,
    })

    if (!confirmed) return

    // 신청 API 호출
    try {
      const res = await fetch('/api/apply/submit', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({
          tempToken: getTempToken(),
          date: slot.date,
          leaveType: selectedType,
        }),
      })

      const data = await res.json()

      if (data.success) {
        alert('신청이 완료되었습니다!')
        fetchStatus() // 새로고침
      } else {
        alert(data.error)
      }
    } catch (error) {
      alert('신청 중 오류가 발생했습니다')
    }
  }

  return (
    <div className="apply-page">
      {/* 유형 선택 */}
      <div className="leave-type-selector">
        <label>
          <input
            type="radio"
```

```
        value="ANNUAL"
        checked={selectedType === 'ANNUAL'}
        onChange={(e) => setSelectedType(e.target.value as 'ANNUAL')}
      />
      연차
    </label>
    <label>
      <input
        type="radio"
        value="OFF"
        checked={selectedType === 'OFF'}
        onChange={(e) => setSelectedType(e.target.value as 'OFF')}
      />
      오프
    </label>
  </div>

  {/* 오프 안내 */}
  {selectedType === 'OFF' && (
    <div className="off-notice">
      ⚠ 오프는 주 2일까지만 신청 가능합니다 (주4일제 기준)
    </div>
  )}

  {/* 슬롯 목록 */}
  <div className="slot-list">
    {slotStatus.map((slot) => {
      const check = canApply(slot)
      const dayOfWeekText = ['일', '월', '화', '수', '목', '금', '토'][slot.dayOfWeek]

      return (
        <div
          key={slot.date}
          className={`slot-item ${!check.canApply ? 'disabled' : ''}`}
        >
          <div className="date">
            {format(new Date(slot.date), 'M월 d일')} ({dayOfWeekText})
          </div>

          <div className="status">
            {slot.isHoliday ? (
              <>🔒 {check.reason}</>
            ) : slot.isFull ? (
              <>⚠ 마감</>
            ) : (
              <>{'⚫'.repeat(slot.current)}{'⚪'.repeat(slot.max - slot.current)} {slot.current}/{slot.max}</>
            )}
```

```
            </div>

            <button
              onClick={() => handleApply(slot.date)}
              disabled={!check.canApply}
            >
              {check.canApply ? '신청' : check.reason}
            </button>
          </div>
        )
      })}
    </div>

    {/* 주간 오프 현황 */}
    {selectedType === 'OFF' && (
      <div className="weekly-off-status">
        <h3>📊 내 오프 신청 현황</h3>
        {weeklyOffCounts.map((week) => (
          <div key={week.weekStart} className="week-info">
            • {week.weekStart} ~ {week.weekEnd}: {week.count}일 신청
            ({week.maxAllowed - week.count}일 더 가능)
          </div>
        ))}
      </div>
    )}
  </div>
 )
}
```

신청 로직:
```typescript
// src/app/api/apply/submit/route.ts

export async function POST(request: Request) {
  const { tempToken, date, leaveType } = await request.json()

  // 1. 임시 토큰 검증
  const tokenData = await verifyTempToken(tempToken)

  if (!tokenData) {
    return NextResponse.json(
      { success: false, error: '인증이 만료되었습니다' },
      { status: 401 }
    )
  }

  // 2. 신청 링크 확인
  const link = await prisma.applicationLink.findUnique({
    where: { id: tokenData.linkId },
    include: { slotLimits: true },
  })

  if (!link || !link.isActive) {
    return NextResponse.json(
      { success: false, error: '신청 기간이 아닙니다' },
      { status: 400 }
    )
  }

  // 3. 날짜 유효성 확인
  const targetDate = new Date(date)

  if (targetDate < link.startDate || targetDate > link.endDate) {
    return NextResponse.json(
      { success: false, error: '신청 가능한 날짜가 아닙니다' },
      { status: 400 }
    )
  }

  // 4. ⭐ 휴일 체크 (일요일 또는 공휴일)
  const dayOfWeek = targetDate.getDay()
  const isHoliday = await prisma.holiday.findFirst({
    where: {
      clinicId: link.clinicId,
```

```
    date: targetDate,
  },
})

if (dayOfWeek === 0 || isHoliday) {
  return NextResponse.json(
    { success: false, error: '휴무일에는 신청할 수 없습니다' },
    { status: 400 }
  )
}

// 5. ⭐ 오프 신청 제한 체크 (주 2일)
if (leaveType === 'OFF') {
  const weekStart = startOfWeek(targetDate, { weekStartsOn: 1 }) // 월요일 시작
  const weekEnd = endOfWeek(targetDate, { weekStartsOn: 1 })

  const offCountThisWeek = await prisma.leaveApplication.count({
    where: {
      linkId: link.id,
      staffId: tokenData.staffId,
      leaveType: 'OFF',
      leaveDate: {
        gte: weekStart,
        lte: weekEnd,
      },
      status: { not: 'CANCELLED' },
    },
  })

  if (offCountThisWeek >= 2) {
    return NextResponse.json(
      {
        success: false,
        error: '오프는 주 2일까지만 신청할 수 있습니다',
        details: {
          weekStart: format(weekStart, 'M월 d일'),
          weekEnd: format(weekEnd, 'M월 d일'),
          currentCount: offCountThisWeek,
          maxAllowed: 2,
        }
      },
      { status: 400 }
    )
  }
}

// 6. 슬롯 제한 확인
```

```
const slotLimit = link.slotLimits.find(sl =>
  isSameDay(sl.date, targetDate)
)

if (!slotLimit) {
  return NextResponse.json(
    { success: false, error: '신청할 수 없는 날짜입니다' },
    { status: 400 }
  )
}

// 7. 현재 신청 수 확인
const currentCount = await prisma.leaveApplication.count({
  where: {
    linkId: link.id,
    leaveDate: targetDate,
    status: { not: 'CANCELLED' },
  },
})

if (currentCount >= slotLimit.maxSlots) {
  return NextResponse.json(
    { success: false, error: '신청 인원이 마감되었습니다' },
    { status: 400 }
  )
}

// 8. 중복 신청 확인
const existing = await prisma.leaveApplication.findFirst({
  where: {
    linkId: link.id,
    staffId: tokenData.staffId,
    leaveDate: targetDate,
    status: { not: 'CANCELLED' },
  },
})

if (existing) {
  return NextResponse.json(
    { success: false, error: '이미 신청한 날짜입니다' },
    { status: 400 }
  )
}

// 9. 신청 생성
const application = await prisma.leaveApplication.create({
  data: {
```

```
      linkId: link.id,
      staffId: tokenData.staffId,
      leaveDate: targetDate,
      leaveType,
      status: 'PENDING',
      ipAddress: request.headers.get('x-forwarded-for') || 'unknown',
      userAgent: request.headers.get('user-agent') || 'unknown',
    },
  })

  // 10. 알림 생성 (관리자에게)
  await createNotification({
    userId: link.createdBy,
    type: 'INFO',
    category: 'LEAVE',
    title: '새로운 연차/오프 신청',
    message: `${tokenData.staffName}님이 ${format(targetDate, 'M월 d일')}에 ${leaveType === 'ANNUAL' ?
'연차' : '오프'}를 신청했습니다`,
    actionUrl: `/leave-management`,
  })

  // 11. 활동 로그
  await createActivityLog({
    userId: tokenData.staffId,
    action: 'LEAVE_APPLICATION_CREATED',
    targetType: 'LeaveApplication',
    targetId: application.id,
  })

  return NextResponse.json({
    success: true,
    applicationId: application.id,
  })
}
```

3.3.4 연차관리 대시보드 ⭐

위치: 메인 메뉴 > 📋 연차관리

페이지 구성:
```tsx
┌──────────────────────────────────────────────────────────────────────────┐
│  📋 연차/오프 관리                          [🔗 신청 링크 생성]  │
├──────────────────────────────────────────────────────────────────────────┤
│                                                                          │
│  대상 월: [2025년 ▼] [2월 ▼]                                          │
```


📊 대시보드
총 신청
35건
확정
35건
대기 중
0건
취소
2건
가장 많이 신청된 날짜:
• 2월 10일 (토): 3명 (슬롯 마감)
• 2월 17일 (토): 3명 (슬롯 마감)
• 2월 5일 (수): 3명 (슬롯 마감)
보기 방식:
● 달력뷰 ○ 목록뷰 ○ 직원별뷰
【달력뷰】
월 화 수 목 금 토 일
1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
...
범례:
1~2명 신청 ⚠ 슬롯 마감 ✅ 0명 (여유)

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                                                                              │
│                                                                              │
│  선택된 날짜: 2월 5일 (수)                                                   │
│                                                                              │
│  신청자 목록: (3명 / 슬롯 마감)                                              │
│  ┌──────────────────────────────────────────────────────────────────────┐    │
│  │                                                                      │    │
│  │  | 이름   | 유형 | 신청일시       | 상태 | 동작 |                   │    │
│  │  |--------|------|----------------|------|------|                   │    │
│  │  | 김철수 | 연차 | 01-21 14:23    | 확정 | [✏️][🗑️] |              │    │
│  │  | 이영희 | 오프 | 01-21 15:10    | 확정 | [✏️][🗑️] |              │    │
│  │  | 박민수 | 연차 | 01-22 09:30    | 확정 | [✏️][🗑️] |              │    │
│  └──────────────────────────────────────────────────────────────────────┘    │
│                                                                              │
│  [➕ 신청 추가]  [✅ 선택 확정]  [❌ 선택 취소]                              │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘
```

목록뷰:

```tsx
┌──────────────────────────────────────────────────────────────────────────────┐
│  📋 연차/오프 관리 - 목록뷰                                                  │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│  필터:                                                                       │
│  날짜: [전체 ▼]  직원: [전체 ▼]  상태: [전체 ▼]                              │
│  정렬: [날짜순 ▼]                                                            │
│                                                                              │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘
┌──────────────────────────────────────────────────────────────────────────────┐
│                                                                              │
│                                                                              │
│  ┌──────────────────────────────────────────────────────────────────────┐    │
│  │                                                                      │    │
│  │  | ☐ | 날짜     | 이름   | 유형 | 신청일시     | 상태 | 동작 |       |    │
│  │  |---|----------|--------|------|--------------|------|------|       |    │
│  │  | ☐ | 2월 1일  | 김철수 | 연차 | 01-21 14:23  | 확정 | [✏️][🗑️] |  |    │
│  │  | ☐ | 2월 1일  | 이영희 | 오프 | 01-21 15:10  | 확정 | [✏️][🗑️] |  |    │
│  │  | ☐ | 2월 2일  | 박민수 | 연차 | 01-22 09:30  | 확정 | [✏️][🗑️] |  |    │
│  │  | ☐ | 2월 3일  | 최지훈 | 연차 | 01-20 18:45  | 확정 | [✏️][🗑️] |  |    │
```

```
|  | ☐ | 2월 3일 | 정수진 | 오프 | 01-21 11:20 | 확정 | [✏️][🗑️] |  |
|  | ... | ... | ... | ... | ... | ... | ... | ... |  |
|
└──────────────────────────────────────────────────────────────────────────────┘

|
|                                                    |
|  ☐ 전체 선택  35개 중 0개 선택됨                                  |
|                                                    |
|  [✅ 일괄 확정] [❌ 일괄 취소] [📥 Excel 다운로드]                  |
|                                                    |
└──────────────────────────────────────────────────────────────────────────────┘
```

직원별뷰:

```tsx
┌──────────────────────────────────────────────────────────────────────────────┐
|  📋 연차/오프 관리 - 직원별뷰                                                  |
├──────────────────────────────────────────────────────────────────────────────┤
|                                                    |
|
┌──────────────────────────────────────────────────────────────────────────┐
|
|  | 이름  | 총연차 | 사용 | 잔여 | 2월신청 | 상태 | 상세보기 |  |
|
├──────────────────────────────────────────────────────────────────────────┤
|
|  | 김철수 | 15 | 3 | 12 | 1건 | 확정 | [보기] |  |
|  | 이영희 | 15 | 2 | 13 | 2건 | 확정 | [보기] |  |
|  | 박민수 | 15 | 4 | 11 | 1건 | 확정 | [보기] |  |
|  | 최지훈 | 12 | 1 | 11 | 1건 | 확정 | [보기] |  |
|  | 정수진 | 15 | 5 | 10 | 2건 | 확정 | [보기] |  |
|  | ... | ... | ... | ... | ... | ... | ... |  |
|
└──────────────────────────────────────────────────────────────────────────┘

|
|                                                    |
|  선택된 직원: 김철수                                          |
|                                                    |
|  📆 2025년 신청 내역 타임라인:                                    |
|                                                    |
|  ┌──────────────────────────────────────────────────────────────────┐
|
|  | ● 2월 5일 (수) - 연차 [확정]                  |  |
|  |   신청: 01-21 14:23                  |  |
|  |   확정: 01-24 09:00                  |  |
|  |                                  |  |
|  | ○ 3월 15일 (금) - 연차 [계획]                 |  |
```

```
|   |                                   |       |
|   | ● 1월 10일 (목) - 연차 [완료]          |       |
|   └──────────────────────────────────────────────────────────────┘
|
|                                       |
└──────────────────────────────────────────────────────────────────────
```

3.3.5 연차 확정 및 스케줄 반영

```typescript
// src/app/api/leave/applications/confirm/route.ts

export async function POST(request: Request) {
  const { applicationIds, confirmAll, month } = await request.json()

  let applications: LeaveApplication[]

  if (confirmAll) {
    // 월 전체 확정
    applications = await prisma.leaveApplication.findMany({
      where: {
        clinicId,
        status: 'PENDING',
        leaveDate: {
          gte: startOfMonth(new Date(month)),
          lte: endOfMonth(new Date(month)),
        },
      },
      include: { staff: true },
    })
  } else {
    // 선택된 신청만 확정
    applications = await prisma.leaveApplication.findMany({
      where: {
        id: { in: applicationIds },
        clinicId,
      },
      include: { staff: true },
    })
  }

  const results = {
    confirmed: 0,
    failed: 0,
    errors: [],
  }
```

```
// 트랜잭션으로 처리
await prisma.$transaction(async (tx) => {
  for (const app of applications) {
    try {
      // 1. 신청 상태 변경
      await tx.leaveApplication.update({
        where: { id: app.id },
        data: {
          status: 'CONFIRMED',
          confirmedAt: new Date(),
          confirmedBy: userId,
        },
      })

      // 2. 스케줄에 반영 (연차 표시)
      const schedule = await tx.schedule.findUnique({
        where: {
          clinicId_scheduleDate: {
            clinicId,
            scheduleDate: app.leaveDate,
          },
        },
      })

      if (schedule) {
        // 기존 스케줄에서 해당 직원 제거 (배치되어 있다면)
        await tx.staffAssignment.deleteMany({
          where: {
            scheduleId: schedule.id,
            staffId: app.staffId,
          },
        })
      }

      // 3. 연차 사용 개수 증가
      if (app.leaveType === 'ANNUAL') {
        await tx.staff.update({
          where: { id: app.staffId },
          data: {
            annualLeaveUsed: {
              increment: 1,
            },
          },
        })
      }

      results.confirmed++
```

```
      } catch (error) {
        results.failed++
        results.errors.push({
          applicationId: app.id,
          reason: error.message,
        })
      }
    }
  })

  // 4. 신청 링크 비활성화 (선택)
  if (confirmAll) {
    await prisma.applicationLink.updateMany({
      where: {
        clinicId,
        year: new Date(month).getFullYear(),
        month: new Date(month).getMonth() + 1,
      },
      data: {
        isActive: false,
        closedAt: new Date(),
        closedBy: userId,
      },
    })
  }

  // 5. 알림 발송
  await createNotification({
    userId,
    type: 'SUCCESS',
    category: 'LEAVE',
    title: '연차 확정 완료',
    message: `${results.confirmed}건의 연차가 확정되었습니다`,
  })

  return NextResponse.json({
    success: true,
    confirmed: results.confirmed,
    failed: results.failed,
    errors: results.errors,
    scheduleUpdated: true,
  })
}
```

3.4 스케줄 자동 배치 시스템 ⭐

3.4.1 배치 설정 UI

```tsx
┌──────────────────────────────────────────────────────────────┐
│ 🤖 스케줄 자동 배치                              [닫기 X]  │
├──────────────────────────────────────────────────────────────┤
│                                                              │
│ 대상 월: 2025년 2월                                          │
│                                                              │
│ 【1】 배치 범위                                              │
│ ⚫ 월간 배치 (권장)                                          │
│   → 한 달 전체를 한 번에 배치 (10~30초)                      │
│ ⚪ 주간 배치                                                 │
│   → 특정 주만 배치: [1주차 ▼]                                │
│ ⚪ 일별 배치                                                 │
│   → 특정 날짜만: [2월 5일 ▼]                                 │
│                                                              │
├──────────────────────────────────────────────────────────────┤
│                                                              │
│ 【2】 배치 방식                                              │
│ ⚫ 스마트 자동 배치 (권장)                                   │
│   • 최초: 완전 재배치                                        │
│   • 이후: 기존 유지 + 필요 변경만                            │
│   • 직원 혼란 최소화                                         │
│ ⚪ 완전 재배치                                               │
│   • 전체 초기화 후 재배치                                    │
│   • 최적의 형평성                                            │
│                                                              │
├──────────────────────────────────────────────────────────────┤
│                                                              │
│ 【3】 등급별 배치 전략                                       │
│ ⚪ 방식1: 자동 배분 (최소 인원만 설정)                       │
│ ⚫ 방식2: 비율 기반 배치                                     │
│                                                              │
│ 팀장·마스터:                                                 │
│ • 필수 최소: [1]명                                           │
│ • 비율: [15]%                                                │
│ • ☑ 유동성 허용: ±[1]명                                      │
│                                                              │
│ 고년차:                                                      │
│ • 비율: [25]%                                                │
│ • ☑ 유동성 허용: ±[1]명                                      │
```

```
│                                              │
│  중년차:                                      │
│  • 비율: [35]%                                │
│  • ✅ 유동성 허용: ±[2]명                      │
│                                              │
│  저년차:                                      │
│  • 비율: [25]%                                │
│  • ✅ 유동성 허용: ±[1]명                      │
│                                              │
│  [💾 비율 설정 저장]                           │
│                                              │
│
─────────────────────────────────────────────────
│
│                                              │
│  【4】 추가 옵션                               │
│  ✅ 형평성 우선 고려                            │
│  ✅ 주말 근무 허용                              │
│  ✅ 특별 조건 존중                              │
│                                              │
│
─────────────────────────────────────────────────
│
│                                              │
│  📊 예상 결과:                                 │
│  • 배치 대상: 22일                             │
│  • 필요 인원: 약 220명 (연인원)                  │
│  • 예상 시간: 15~20초                           │
│                                              │
│  [🔄 자동 배치 시작]  [취소]                     │
│                                              │
└────────────────────────────────────────────────
```

배치 진행 화면:

```tsx
┌────────────────────────────────────────────────
│  ⏳ 스케줄 자동 배치 중...                       │
├────────────────────────────────────────────────
│                                              │
│  ████████████████░░░░░░░░░ 65%                          │
│                                              │
│  진행 상황:                                    │
│  ✅ 제약 조건 수집 완료                          │
│  ✅ 연차/휴무 확인 완료                          │
│  ⏳ 14일 배치 중... (22일 중)                    │
│  ⏸ 대기: 형평성 점수 계산                        │
```

```
┌──────────────────────────────────────────────────────────┐
│  ⏸ 대기: 검증                                            │
│                                                          │
│  예상 남은 시간: 약 8초                                  │
│                                                          │
│  💡 잠시만 기다려주세요...                               │
│                                                          │
└──────────────────────────────────────────────────────────┘
```

배치 완료 화면:

```tsx
┌──────────────────────────────────────────────────────────┐
│  ✅ 자동 배치 완료!                                      │
├──────────────────────────────────────────────────────────┤
│                                                          │
│  📊 배치 결과:                                           │
│  • 배치 완료: 22일 / 22일                                │
│  • 총 배치 인원: 218명 (연인원)                          │
│  • 소요 시간: 18초                                       │
│                                                          │
│                                                          │
├──────────────────────────────────────────────────────────┤
│                                                          │
│  ✅ 검증 결과:                                           │
│  • 필수 인원 충족: 22/22일 ✅                            │
│  • 형평성 점수: 평균 0.8 (양호) 🟢                       │
│  • 제약 조건 위반: 0건 ✅                                │
│                                                          │
│  🟡 경고사항:                                            │
│  • 김철수: 야간 근무 4회 (평균 3회, +1)                  │
│  • 이영희: 주말 근무 3회 (평균 2회, +1)                  │
│                                                          │
│  💡 경고는 무시해도 되며, 필요 시 수동 조정 가능합니다   │
│                                                          │
│                                                          │
├──────────────────────────────────────────────────────────┤
│                                                          │
│  다음 작업:                                              │
│  • 달력에서 배치된 직원 확인                             │
│  • 필요 시 수동 조정                                     │
│  • 형평성 대시보드 확인                                  │
│                                                          │
│  [✅ 이대로 적용] [🔄 다시 배치] [❌ 취소]               │
│                                                          │
└──────────────────────────────────────────────────────────┘
```

```

#### 3.4.2 배치 알고리즘

**전체 흐름:**
```typescript
// src/lib/auto-assign/engine.ts

async function autoAssignSchedule(
  config: AutoAssignConfig
): Promise<AutoAssignResult> {

  // 1. 초기화
  console.log('🚀 자동 배치 시작')
  const startTime = Date.now()

  // 2. 데이터 수집
  console.log('📊 데이터 수집 중...')
  const dates = await getTargetDates(config)
  const schedules = await getSchedules(dates)
  const staff = await getActiveStaff()
  const leaves = await getLeaveApplications(dates)
  const constraints = await getConstraints()

  // 3. 배치 방식 결정
  const isFirstTime = schedules.every(s => s.staffAssignments.length === 0)
  const useFullReassign = config.method === 'FULL' || isFirstTime

  if (useFullReassign) {
    console.log('🔄 완전 재배치 모드')
    return await fullReassign(schedules, staff, leaves, constraints, config)
  } else {
    console.log('💡 스마트 배치 모드 (기존 유지)')
    return await smartReassign(schedules, staff, leaves, constraints, config)
  }
}
```

완전 재배치:
```typescript
async function fullReassign(
  schedules: Schedule[],
  staff: Staff[],
  leaves: LeaveApplication[],
  constraints: Constraint[],
  config: AutoAssignConfig
): Promise<AssignResult> {
```
```

```
const results = {
  success: [],
  failed: [],
  warnings: [],
}

// 1. 기존 배치 초기화
await prisma.staffAssignment.deleteMany({
  where: {
    scheduleId: { in: schedules.map(s => s.id) },
  },
})

// 2. 각 날짜별 배치
for (const schedule of schedules) {
  console.log(`📅 ${format(schedule.scheduleDate, 'M월 d일')} 배치 중...`)

  // 2.1. 연차/휴무자 제외
  const availableStaff = staff.filter(s =>
    !isOnLeave(s, schedule.scheduleDate, leaves)
  )

  // 2.2. 필요 인원 계산
  const required = calculateRequiredStaff(
    schedule.doctors.length,
    schedule.hasNightShift
  )

  // 2.3. 배치 전략에 따라 선발
  let selected: Staff[]

  if (config.assignmentStrategy === 'RATIO') {
    // 비율 기반
    selected = selectByRatio(
      availableStaff,
      required,
      config.ratioConfig,
      schedule.scheduleDate
    )
  } else {
    // 자동 배분
    selected = selectByAuto(
      availableStaff,
      required,
      schedule.scheduleDate
    )
```

```
    }

    // 2.4. 검증
    const validation = validateAssignment(selected, required, constraints)

    if (!validation.isValid) {
      results.failed.push({
        date: schedule.scheduleDate,
        reason: validation.errors,
      })
      continue
    }

    if (validation.warnings.length > 0) {
      results.warnings.push(...validation.warnings)
    }

    // 2.5. DB 저장
    await prisma.staffAssignment.createMany({
      data: selected.map(s => ({
        scheduleId: schedule.id,
        staffId: s.id,
      })),
    })

    results.success.push(schedule.scheduleDate)
  }

  // 3. 형평성 점수 재계산
  await recalculateFairnessScores(config.year, config.month)

  const endTime = Date.now()
  const duration = (endTime - startTime) / 1000

  console.log(`✅ 배치 완료! (${duration}초)`)

  return {
    success: true,
    schedulesAssigned: results.success.length,
    schedulesFailed: results.failed.length,
    warnings: results.warnings,
    duration,
  }
}
```

스마트 재배치:

```typescript
async function smartReassign(
  schedules: Schedule[],
  staff: Staff[],
  leaves: LeaveApplication[],
  constraints: Constraint[],
  config: AutoAssignConfig
): Promise<AssignResult> {

  const results = {
    kept: [],
    changed: [],
    failed: [],
    warnings: [],
  }

  // 1. 각 날짜별로 문제 식별
  for (const schedule of schedules) {
    const existingAssignments = schedule.staffAssignments

    // 1.1. 문제 확인
    const issues = identifyIssues(schedule, leaves, constraints)

    if (issues.length === 0) {
      // 문제 없음: 기존 유지
      results.kept.push(schedule.scheduleDate)
      continue
    }

    console.log(`⚠️ ${format(schedule.scheduleDate, 'M월 d일')}: ${issues.length}개 문제 발견`)

    // 1.2. 문제 해결
    try {
      // 연차자 제거
      const leaveStaffIds = leaves
        .filter(l => isSameDay(l.leaveDate, schedule.scheduleDate))
        .map(l => l.staffId)

      await prisma.staffAssignment.deleteMany({
        where: {
          scheduleId: schedule.id,
          staffId: { in: leaveStaffIds },
        },
      })

      // 빈 자리 채우기
      const currentStaff = await prisma.staffAssignment.findMany({
```

```
      where: { scheduleId: schedule.id },
      include: { staff: true },
    })

    const required = calculateRequiredStaff(
      schedule.doctors.length,
      schedule.hasNightShift
    )

    const shortage = calculateShortage(currentStaff, required)

    if (shortage.total > 0) {
      // 추가 배치 필요
      const additional = selectAdditionalStaff(
        staff,
        currentStaff.map(a => a.staff),
        shortage,
        schedule.scheduleDate,
        leaves
      )

      await prisma.staffAssignment.createMany({
        data: additional.map(s => ({
          scheduleId: schedule.id,
          staffId: s.id,
        })),
      })
    }

    results.changed.push(schedule.scheduleDate)

  } catch (error) {
    results.failed.push({
      date: schedule.scheduleDate,
      reason: error.message,
    })
  }
}

// 2. 형평성 재확인
const fairnessIssues = await checkFairness(config.year, config.month)

if (fairnessIssues.length > 0) {
  results.warnings.push(...fairnessIssues)
}

console.log(`✅ 스마트 재배치 완료!`)
```

```
  console.log(`  • 유지: ${results.kept.length}일`)
  console.log(`  • 변경: ${results.changed.length}일`)

  return {
    success: true,
    schedulesKept: results.kept.length,
    schedulesChanged: results.changed.length,
    schedulesFailed: results.failed.length,
    warnings: results.warnings,
  }
}
```

비율 기반 선발:
```typescript
function selectByRatio(
  availableStaff: Staff[],
  required: RequiredStaff,
  ratioConfig: RatioConfig,
  date: Date
): Staff[] {

  // 1. 등급별로 분류
  const byRank = {
    leader: availableStaff.filter(s => s.rank === 'LEADER'),
    senior: availableStaff.filter(s => s.rank === 'SENIOR'),
    intermediate: availableStaff.filter(s => s.rank === 'INTERMEDIATE'),
    junior: availableStaff.filter(s => s.rank === 'JUNIOR'),
  }

  // 2. 총 필요 인원
  const totalNeeded = required.leader.min + required.intermediate.min + required.junior.min

  // 3. 비율에 따른 인원 계산
  const targets = {
    leader: Math.round(totalNeeded * ratioConfig.leader.ratio / 100),
    senior: Math.round(totalNeeded * ratioConfig.senior.ratio / 100),
    intermediate: Math.round(totalNeeded * ratioConfig.intermediate.ratio / 100),
    junior: Math.round(totalNeeded * ratioConfig.junior.ratio / 100),
  }

  // 4. 유동성 적용
  const ranges = {
    leader: {
      min: Math.max(required.leader.min, targets.leader - ratioConfig.leader.flexibility),
      max: targets.leader + ratioConfig.leader.flexibility,
    },
```

```
    senior: {
      min: Math.max(0, targets.senior - ratioConfig.senior.flexibility),
      max: targets.senior + ratioConfig.senior.flexibility,
    },
    intermediate: {
      min: Math.max(required.intermediate.min, targets.intermediate - ratioConfig.intermediate.flexibility),
      max: targets.intermediate + ratioConfig.intermediate.flexibility,
    },
    junior: {
      min: Math.max(required.junior.min, targets.junior - ratioConfig.junior.flexibility),
      max: targets.junior + ratioConfig.junior.flexibility,
    },
  }

  // 5. 각 등급별 선발 (형평성 점수 기반)
  const selected: Staff[] = []

  // 5.1. 팀장·마스터
  const leaderCount = clamp(targets.leader, ranges.leader.min, ranges.leader.max)
  const leaders = selectByFairness(byRank.leader, leaderCount, date)
  selected.push(...leaders)

  // 5.2. 고년차
  const seniorCount = clamp(targets.senior, ranges.senior.min, ranges.senior.max)
  const seniors = selectByFairness(byRank.senior, seniorCount, date)
  selected.push(...seniors)

  // 5.3. 중년차
  const intermediateCount = clamp(targets.intermediate, ranges.intermediate.min, ranges.intermediate.max)
  const intermediates = selectByFairness(byRank.intermediate, intermediateCount, date)
  selected.push(...intermediates)

  // 5.4. 저년차
  const juniorCount = clamp(targets.junior, ranges.junior.min, ranges.junior.max)
  const juniors = selectByFairness(byRank.junior, juniorCount, date)
  selected.push(...juniors)

  return selected
}
```

형평성 기반 선발:
```typescript
function selectByFairness(
  candidates: Staff[],
  count: number,
  date: Date
```

```
): Staff[] {

  // 1. 각 직원의 형평성 점수 계산
  const scored = candidates.map(staff => ({
    staff,
    score: calculateFairnessScore(staff, date),
  }))

  // 2. 점수 낮은 순 정렬 (근무 적게 한 사람 우선)
  scored.sort((a, b) => a.score - b.score)

  // 3. 상위 N명 선택
  return scored.slice(0, count).map(s => s.staff)
}

function calculateFairnessScore(
  staff: Staff,
  targetDate: Date
): number {

  const year = targetDate.getFullYear()
  const month = targetDate.getMonth() + 1

  // 해당 월의 근무 통계
  const stats = getStaffWorkStats(staff.id, year, month)

  // 평균 대비 편차
  const avgNight = getAverageNightShifts(year, month)
  const avgWeekend = getAverageWeekendShifts(year, month)

  const nightDev = Math.abs(stats.nightShifts - avgNight)
  const weekendDev = Math.abs(stats.weekendShifts - avgWeekend)

  // 가중 평균 (야간 3, 주말 2)
  const score = (nightDev * 3 + weekendDev * 2) / 5

  return score
}
```

3.5 형평성 관리 ⭐

3.5.1 형평성 점수 계산

데이터 구조:

```typescript
interface FairnessScore {
  id: string
  staffId: string
  staffName: string
  year: number
  month: number

  metrics: {
    nightShifts: {
      count: number
      average: number
      deviation: number
      percentile: number
    }
    weekendShifts: {
      count: number
      average: number
      deviation: number
      percentile: number
    }
  }

  totalScore: number
  grade: 'EXCELLENT' | 'GOOD' | 'FAIR' | 'POOR'
  calculatedAt: Date
}
```

계산 공식:
```typescript
async function calculateFairnessScores(
  year: number,
  month: number
): Promise<FairnessScore[]> {

  // 1. 모든 활성 직원 조회
  const staff = await prisma.staff.findMany({
    where: {
      clinicId,
      isActive: true,
    },
  })

  // 2. 해당 월의 모든 스케줄 조회
  const startDate = new Date(year, month - 1, 1)
  const endDate = endOfMonth(startDate)
```

```
const schedules = await prisma.schedule.findMany({
  where: {
    clinicId,
    scheduleDate: {
      gte: startDate,
      lte: endDate,
    },
  },
  include: {
    staffAssignments: {
      include: { staff: true },
    },
  },
})

// 3. 각 직원별 통계 계산
const stats = staff.map(s => {
  let nightShifts = 0
  let weekendShifts = 0

  for (const schedule of schedules) {
    const assigned = schedule.staffAssignments.find(a => a.staffId === s.id)

    if (assigned) {
      // 야간 근무
      if (schedule.hasNightShift) {
        nightShifts++
      }

      // 주말 근무 (토요일=6, 일요일=0)
      if (schedule.dayOfWeek === 0 || schedule.dayOfWeek === 6) {
        weekendShifts++
      }
    }
  }

  return {
    staffId: s.id,
    staffName: s.name,
    nightShifts,
    weekendShifts,
  }
})

// 4. 평균 계산
const avgNight = mean(stats.map(s => s.nightShifts))
```

```
  const avgWeekend = mean(stats.map(s => s.weekendShifts))

  // 5. 각 직원의 점수 계산
  const scores = stats.map(s => {
    const nightDev = Math.abs(s.nightShifts - avgNight)
    const weekendDev = Math.abs(s.weekendShifts - avgWeekend)

    // 가중 평균 (야간 3, 주말 2)
    const totalScore = (nightDev * 3 + weekendDev * 2) / 5

    // 등급 산정
    let grade: FairnessGrade
    if (totalScore < 0.5) grade = 'EXCELLENT'
    else if (totalScore < 1.0) grade = 'GOOD'
    else if (totalScore < 1.5) grade = 'FAIR'
    else grade = 'POOR'

    // 백분위 계산
    const nightPercentile = calculatePercentile(s.nightShifts, stats.map(x => x.nightShifts))
    const weekendPercentile = calculatePercentile(s.weekendShifts, stats.map(x => x.weekendShifts))

    return {
      staffId: s.staffId,
      staffName: s.staffName,
      year,
      month,
      metrics: {
        nightShifts: {
          count: s.nightShifts,
          average: avgNight,
          deviation: nightDev,
          percentile: nightPercentile,
        },
        weekendShifts: {
          count: s.weekendShifts,
          average: avgWeekend,
          deviation: weekendDev,
          percentile: weekendPercentile,
        },
      },
      totalScore,
      grade,
      calculatedAt: new Date(),
    }
  })

  // 6. DB 저장
```

```
  await prisma.$transaction(
    scores.map(score =>
      prisma.fairnessScore.upsert({
        where: {
          staffId_year_month: {
            staffId: score.staffId,
            year: score.year,
            month: score.month,
          },
        },
        create: score,
        update: score,
      })
    )
  )

  return scores
}
```

3.5.2 형평성 대시보드

```tsx
┌──────────────────────────────────────────────────────────────────────────────┐
│  📊 형평성 분석                                    2025년 2월  │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│  【전체 요약】                                                               │
│                                                                              │
│   ┌──────────────┐  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐     │
│   │ 우수         │  │ 양호         │  │ 보통         │  │ 개선필요     │     │
│   │ 🟢 12명      │  │ 🟡 5명       │  │ 🟠 2명       │  │ 🔴 1명       │     │
│   │ 60%          │  │ 25%          │  │ 10%          │  │ 5%           │     │
│   └──────────────┘  └──────────────┘  └──────────────┘  └──────────────┘     │
│                                                                              │
│                                                                              │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│  【직원별 형평성 점수】                                                      │
│                                                                              │
│  (막대 그래프 - 낮을수록 좋음)                                               │
│                                                                              │
│  김철수 ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓ 0.5 🟢                                        │
│  이영희 ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓ 0.6 🟡                                        │
│  박민수 ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓ 0.3 🟢                                        │
```

| 최지훈  0.8 🟡 |
| 정수진  0.4 🟢 |
| 강민지  1.0 🟠 |
| ... |

[상세 보기] [Excel 다운로드]

【야간 근무 분포】

(히트맵 - 색이 진할수록 많음)

	1주	2주	3주	4주	
김철수	●	●	○	○	2회 🟢
이영희	●	●	●	○	3회 🟢
박민수	●	○	○	○	1회 🟢
최지훈	●	●	●	●	4회 🟡 (평균 대비 +1)
정수진	●	○	●	○	2회 🟢
...					

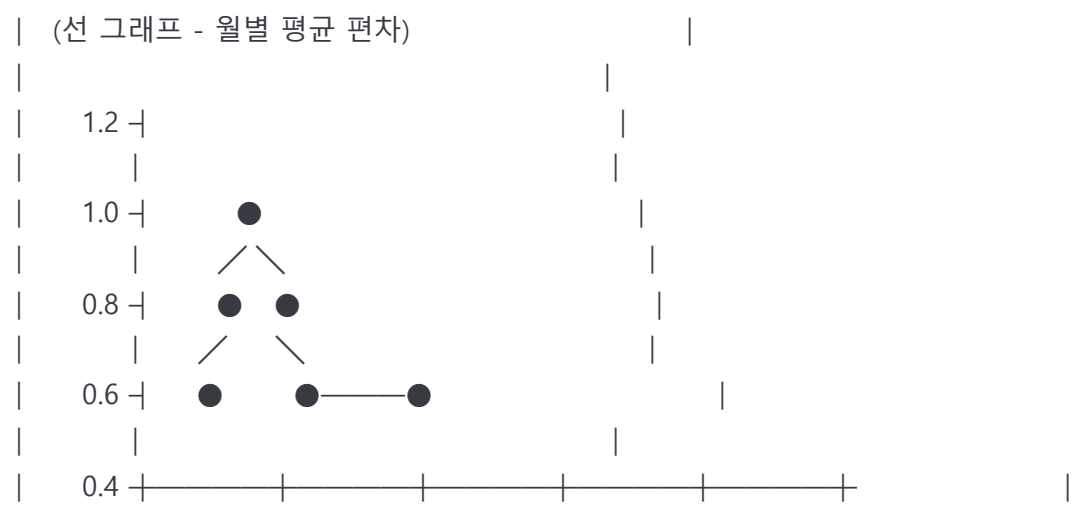
평균: 2.5회 표준편차: 0.8

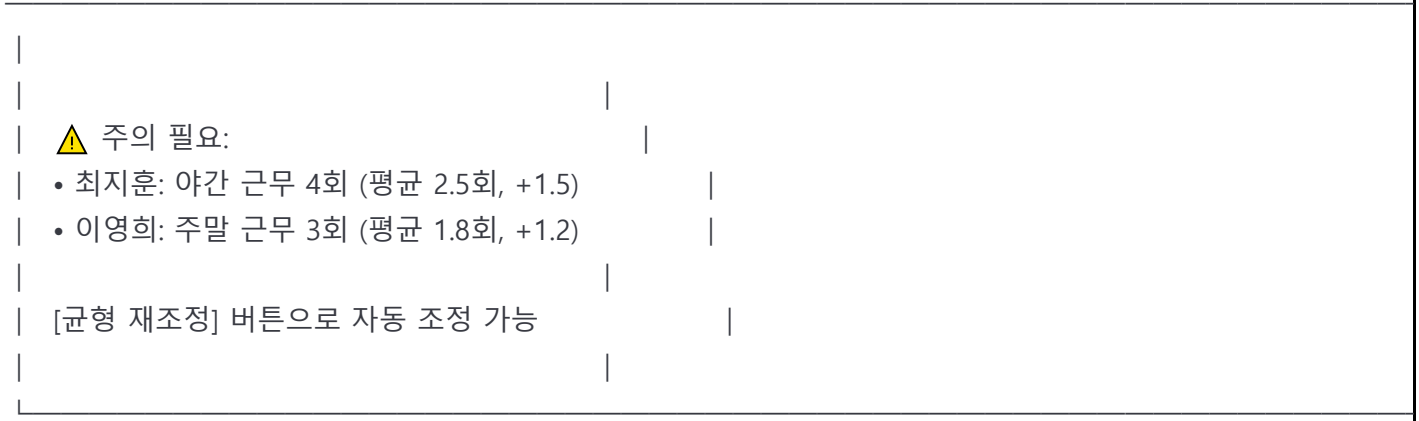
【주말 근무 분포】

	1주	2주	3주	4주	
김철수	●	○	○	○	1회 🟢
이영희	●	●	●	○	3회 🟡 (평균 대비 +1)
박민수	●	○	○	○	1회 🟢
최지훈	●	●	○	○	2회 🟢
정수진	●	○	●	○	2회 🟢
...					

평균: 1.8회 표준편차: 0.7

【시간 경과 추이】






```

---

### 3.6 스케줄 배포 시스템 ⭐

#### 3.6.1 스케줄 확인 페이지 생성
```typescript
// src/app/api/deploy/schedule-link/route.ts

interface ScheduleViewLinkRequest {
  year: number
  month: number
  viewOptions: {
    showFullSchedule: boolean
    showDoctorSchedule: boolean
    showPersonalSchedule: boolean
  }
  expiresInDays: number
}
```
```

```
export async function POST(request: Request) {
  const { year, month, viewOptions, expiresInDays } = await request.json()

  // 1. 토큰 생성
  const token = crypto.randomUUID()

  // 2. DB 저장
  const link = await prisma.scheduleViewLink.create({
    data: {
      token,
      clinicId,
      year,
      month,
      viewOptions,
      expiresAt: add(new Date(), { days: expiresInDays }),
      createdBy: userId,
    },
  })

  // 3. URL 생성
  const url = `${process.env.NEXTAUTH_URL}/schedule/view/${token}`

  return NextResponse.json({
    success: true,
    token,
    url,
    expiresAt: link.expiresAt,
  })
}
```

링크 생성 UI:
```tsx
┌──────────────────────────────────────────────────────────────┐
│  🔗 스케줄 확인 페이지 생성                                    │
├──────────────────────────────────────────────────────────────┤
│                                                              │
│  대상 월: [2025년 ▼] [2월 ▼]                                 │
│                                                              │
│  표시할 스케줄 선택:                                          │
│  ☑ 전체 스케줄표 (모든 직원)                                  │
│  ☑ 원장 스케줄표만                                           │
│  ☑ 개인 스케줄표 (본인 것만)                                  │
│                                                              │
│  다운로드 허용 형식:                                          │
│  ☑ Excel (.xlsx)                                            │
│  ☑ PDF (.pdf)                                               │
```

```
  |                                                  |
  |   링크 유효 기간: [30]일                          |
  |                                                  |
  |   [생성하기]  [취소]                              |
  |                                                  |
  └──────────────────────────────────────────────────┘
```

3.6.2 스케줄 확인 페이지 (외부 링크)

URL:

https://dental-schedule.com/schedule/view/{token}

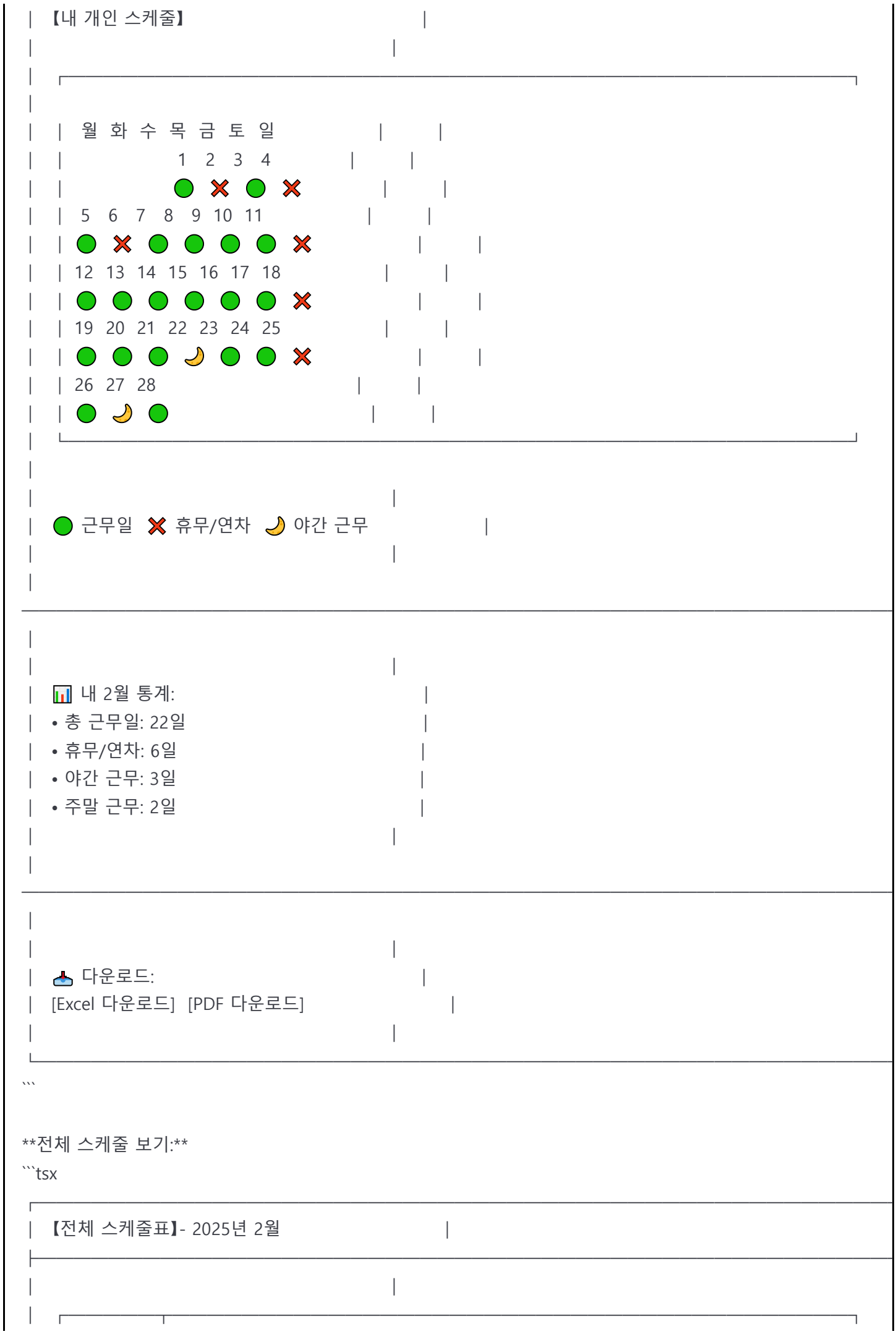
인증 전:

```tsx
┌──────────────────────────────────────────────────────────────┐
│          🏥 연세바로치과 2025년 2월 스케줄                    │
├──────────────────────────────────────────────────────────────┤
│                                                              │
│  👤 이름 선택:                                               │
│  ┌──────────────────────────────────────────────────────┐    │
│  │ 김철수                                          ▼    │    │
│  └──────────────────────────────────────────────────────┘    │
│                                                              │
│  🔐 인증:                                                    │
│  ◯ 생년월일 6자리 (YYMMDD)                                   │
│  ⬤ PIN 번호                                                  │
│  ┌──────────────────────────────────────────────────────┐    │
│  │ [_ _ _ _ _ _]                                   │    │    │
│  └──────────────────────────────────────────────────────┘    │
│                                                              │
│  [인증하기]                                                  │
│                                                              │
└──────────────────────────────────────────────────────────────┘
```

인증 후:

```tsx
┌──────────────────────────────────────────────────────────────┐
│  🏥 연세바로치과 2025년 2월 스케줄 - 김철수님                 │
│  [◀ 2025.01] [2025.02] [2025.03 ▶]                           │
├──────────────────────────────────────────────────────────────┤
│                                                              │
│  ✅ 인증 완료: 김철수님                                      │
│  👤 [내 정보] 🚪 [로그아웃]                                  │
│                                                              │
└──────────────────────────────────────────────────────────────┘

┌──────────────────────────────────────────────────────────────┐
│                                                              │
│  보기 옵션:                                                  │
│  ⬤ 내 스케줄  ◯ 전체 스케줄  ◯ 원장 스케줄                   │
│                                                              │
```




```

**전체 스케줄 보기:**

```tsx
┌──────────────────────────────────────────────────────────────────────────────┐
│ 【전체 스케줄표】- 2025년 2월                                                │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│ ┌──────────┬──────────────────────────────────────────────────────────────┐  │
```
```

```
|
|   | 날짜  |  배치 인원                          |        |
|   ├──────────┼──────────────────────────────────────────────────────────┤
|
|   | 2/1   |  박원장, 구원장, 윤원장 (야간)          |        |
|   | (토)  |  김철수, 이영희, 박민수, 최지훈, ...   |        |
|   |       |  (팀장 1, 고년차 2, 중년차 4, 저년차 4) |        |
|   ├──────────┼──────────────────────────────────────────────────────────┤
|
|   | 2/3   |  박원장, 황원장 (야간 X)               |        |
|   | (월)  |  김철수, 최지훈, 정수진, 강민지, ...   |        |
|   |       |  (팀장 1, 고년차 1, 중년차 2, 저년차 2) |        |
|   ├──────────┼──────────────────────────────────────────────────────────┤
|
|   | 2/4   |  박원장, 효원장 (야간)                 |        |
|   | (화)  |  이영희, 박민수, 정수진, 김지수, ...   |        |
|   |       |  (팀장 1, 고년차 1, 중년차 3, 저년차 3) |        |
|   | ...   |  ...                              |        |
|   └──────────┴──────────────────────────────────────────────────────────┘
|
|                                                |
|   [📥 Excel 다운로드]  [📥 PDF 다운로드]              |
|                                                |
└──────────────────────────────────────────────────────────────────────────┘
```

3.7 알림 시스템 ⭐

3.7.1 Server-Sent Events (SSE) 구현

```typescript
// src/app/api/notifications/sse/route.ts

export async function GET(request: Request) {
  const userId = await getUserIdFromRequest(request)

  if (!userId) {
    return new Response('Unauthorized', { status: 401 })
  }

  const encoder = new TextEncoder()

  const stream = new ReadableStream({
    async start(controller) {
      // 초기 연결 메시지
      controller.enqueue(
```

```
        encoder.encode('data: {"type":"connected"}₩n₩n')
      )

      // 3초마다 새로운 알림 확인
      const interval = setInterval(async () => {
        try {
          const notifications = await prisma.notification.findMany({
            where: {
              userId,
              isRead: false,
              createdAt: {
                gte: sub(new Date(), { minutes: 1 }),
              },
            },
            orderBy: { createdAt: 'desc' },
            take: 10,
          })

          if (notifications.length > 0) {
            controller.enqueue(
              encoder.encode(`data: ${JSON.stringify(notifications)}₩n₩n`)
            )
          }
        } catch (error) {
          console.error('SSE error:', error)
        }
      }, 3000)

      // 클라이언트 연결 종료 시 정리
      request.signal.addEventListener('abort', () => {
        clearInterval(interval)
        controller.close()
      })
    },
  })

  return new Response(stream, {
    headers: {
      'Content-Type': 'text/event-stream',
      'Cache-Control': 'no-cache',
      'Connection': 'keep-alive',
    },
  })
}
```

클라이언트:

```typescript
// src/components/notifications/NotificationCenter.tsx

'use client'

import { useEffect, useState } from 'react'

export function NotificationCenter() {
  const [notifications, setNotifications] = useState<Notification[]>([])
  const [unreadCount, setUnreadCount] = useState(0)

  useEffect(() => {
    // SSE 연결
    const eventSource = new EventSource('/api/notifications/sse')

    eventSource.onmessage = (event) => {
      const data = JSON.parse(event.data)

      if (data.type === 'connected') {
        console.log('✅ 알림 서버 연결됨')
        return
      }

      // 새로운 알림 수신
      if (Array.isArray(data)) {
        setNotifications(prev => [...data, ...prev])
        setUnreadCount(prev => prev + data.length)

        // 브라우저 알림 (권한 있을 경우)
        if (Notification.permission === 'granted') {
          data.forEach(notif => {
            new Notification(notif.title, {
              body: notif.message,
              icon: '/icon.png',
            })
          })
        }
      }
    }

    eventSource.onerror = (error) => {
      console.error('SSE error:', error)
      eventSource.close()
    }

    return () => {
      eventSource.close()
```

```
    }
  }, [])

  return (
    <div className="relative">
      <button onClick={() => setOpen(!open)}>
        🔔
        {unreadCount > 0 && (
          <span className="badge">{unreadCount}</span>
        )}
      </button>

      {/* 알림 목록 */}
    </div>
  )
}
```

3.8 설정 관리

3.8.1 직원 등급 명칭 관리 ⭐

```tsx
┌─────────────────────────────────────────────────────────────────────────────┐
│  ⚙️ 직원 등급 설정                                                          │
├─────────────────────────────────────────────────────────────────────────────┤
│                                                                             │
│  등급 명칭을 병원에 맞게 변경할 수 있습니다.                                │
│                                                                             │
│  레벨 1 (최상위):                                                           │
│  ┌─────────────────────────────────────────────────────────────────────┐    │
│  │  팀장·마스터                                                        │    │
│  └─────────────────────────────────────────────────────────────────────┘    │
│                                                                             │
│  레벨 2 (상위):                                                             │
│  ┌─────────────────────────────────────────────────────────────────────┐    │
│  │  고년차                                                             │    │
│  └─────────────────────────────────────────────────────────────────────┘    │
│                                                                             │
│  레벨 3 (중위):                                                             │
│  ┌─────────────────────────────────────────────────────────────────────┐    │
│                                                                             │
```

```
|   | 중년차                                  |       |
|   └──────────────────────────────────────────┘
|
|                                          |
|   레벨 4 (하위):                          |
|   ┌──────────────────────────────────────────┐
|
|   | 저년차                                  |       |
|   └──────────────────────────────────────────┘
|
|                                          |
|   💡 참고:                                |
|   • 시스템 내부에서는 레벨로 관리됩니다                |
|   • 화면에만 여기서 설정한 명칭이 표시됩니다            |
|   • 언제든지 변경할 수 있습니다                      |
|                                          |
|   [💾 저장]  [🔄 기본값으로 되돌리기]                 |
|                                          |
└──────────────────────────────────────────────────────┘
```

API:
```typescript
// src/app/api/staff/rank-settings/route.ts

export async function GET(request: Request) {
  const settings = await prisma.staffRankSettings.findUnique({
    where: { clinicId },
  })

  return NextResponse.json(
    settings || {
      leaderName: '팀장·마스터',
      seniorName: '고년차',
      intermediateName: '중년차',
      juniorName: '저년차',
    }
  )
}

export async function PUT(request: Request) {
  const { leaderName, seniorName, intermediateName, juniorName } = await request.json()

  const settings = await prisma.staffRankSettings.upsert({
    where: { clinicId },
    create: {
      clinicId,
```

```
        leaderName,
        seniorName,
        intermediateName,
        juniorName,
      },
      update: {
        leaderName,
        seniorName,
        intermediateName,
        juniorName,
      },
    })

    await createActivityLog({
      action: 'RANK_SETTINGS_UPDATED',
      details: settings,
    })

    return NextResponse.json({ success: true, settings })
}
```

Part 2 여기까지!

이제 **섹션 4, 5**를 새 파일로 작성하겠습니다.

다음 문서: Part 3 (데이터 구조, API, 비즈니스 로직, 알고리즘, 보안, 성능, 테스트, 배포)

계속 진행하시겠습니까?