연세바로치과 스케줄 관리 시스템 - 기능 명세서 Part 1

문서 버전: 2.1 (최종)
작성일: 2025-10-21
대상: 백엔드/프론트엔드 개발자
Part: 1/2 (섹션 1~5)

Part 1 목차

다음 문서: 기능명세서 Part 2 (비즈니스 로직, 알고리즘, 보안, 성능, 테스트, 배포)

---

1. 프로젝트 개요

1.1 프로젝트 목적

연세바로치과의 월간 직원 근무 스케줄을 자동화하여 관리자의 업무 부담을 **85% 감소**(3시간 → 15분)시키고, 직원 간 형평성을 보장하며, 연차 및 오프 신청을 효율적으로 관리하는 웹 기반 시스템 개발.

1.2 주요 목표

- **자동화**: 원장 패턴 저장, 월간 일괄 배치, 자동 검증
- **형평성**: 야간 근무, 주말 근무, 연차 사용의 공정한 배분
- **효율성**: 직원 자율 신청 시스템, 실시간 알림, 한눈에 보는 대시보드
- **확장성**: 다른 치과/의원으로 확장 가능한 구조

1.3 사용자 구분

사용자	역할	주요 기능
관리자	원장, 실장	전체 스케줄 관리, 설정, 통계 조회
직원	일반 직원	연차/오프 신청 (외부 링크), 스케줄 확인
원장	진료 원장	조회 전용 (선택적)

1.4 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)
Email: Nodemailer (선택적)

배포 및 인프라

yaml

Hosting: Vercel (권장) 또는 AWS EC2
Database: Supabase (PostgreSQL) 또는 AWS RDS
Storage: AWS S3 (백업/파일)
Monitoring: Sentry
Analytics: Google Analytics 4

1.5 개발 환경 설정

bash

```
# 필수 환경 변수 (.env)
DATABASE_URL="postgresql://..."
NEXTAUTH_SECRET="..."
NEXTAUTH_URL="https://..."

# AWS S3 (백업용, 선택)
AWS_ACCESS_KEY_ID="..."
AWS_SECRET_ACCESS_KEY="..."
AWS_REGION="ap-northeast-2"
AWS_S3_BUCKET="dental-schedule-backup"

# 모니터링 (선택)
NEXT_PUBLIC_SENTRY_DSN="..."
NEXT_PUBLIC_GA_ID="..."
```

---

2. 전체 워크플로우

2.1 개선된 월간 스케줄 작성 프로세스

```
┌─────────────────────────────────────────────────────────────┐
│                    월간 스케줄 작성 워크플로우                    │
└─────────────────────────────────────────────────────────────┘

【STEP 1】 원장 스케줄 세팅 (월간 일괄) ⭐
├── 1-A. 요일별 패턴 불러오기
│   ├── ⚙️ 설정 > 원장 관리 > 요일별 패턴
│   ├── 월~토 각 요일별 원장 조합 저장
│   └── 야간 진료 여부 설정
│
├── 1-B. 달력에서 [🔄 요일 패턴 적용] 클릭
│   ├── 한 달 전체 자동 세팅 (1초 완료)
│   ├── 공휴일 자동 제외
│   └── 기존 스케줄 덮어쓰기 확인
│
├── 1-C. 예외일 수동 수정
│   ├── 특정 날짜 클릭
│   ├── 원장 추가/제거
│   └── 야간 진료 변경
│
└── 1-D. 필요 인원 자동 계산
    ├── 원장 조합 → 등급별 필요 인원
    ├── 계산 공식:
    │   • 원장 1명당: 팀장 또는 고년차 1 + 중년차 1 + 저년차 1
    │   • 야간 추가: 중년차 +1, 저년차 +1
    └── 결과 자동 표시
```

【STEP 2】 연차 및 오프 신청 오픈 ⭐
├── 2-A. 신청 링크 생성
│ ├── 📋 연차관리 > [📎 신청 링크 생성]
│ ├── 신청 기간 설정 (예: 1월 20일 ~ 1월 23일)
│ ├── 날짜별 슬롯 설정 (예: 평일 3명, 주말 1명)
│ ├── 고유 토큰 생성 (UUID)
│ └── URL 생성: https://domain.com/apply/{token}
│
├── 2-B. 모바일로 링크 공유 📱
│ ├── [공유하기] 버튼 클릭
│ │ └── Web Share API 사용 (API 등록 불필요!)
│ ├── 모바일: 카톡, 문자, 이메일 등 선택 가능
│ └── PC: 링크 자동 복사 → 수동 전송
│
└── 2-C. 직원들 자율 신청
├── 링크 접속 (로그인 불필요)
├── 이름 선택
├── 생년월일 6자리 입력 (YYMMDD)
├── [인증하기] 클릭
├── 인증 성공 시:
│ ├── [PIN 번호 설정하기] (선택)
│ ├── 실시간 슬롯 현황 확인
│ ├── 원하는 날짜 신청
│ └── 신청 취소도 가능
└── 마감일 자동 종료

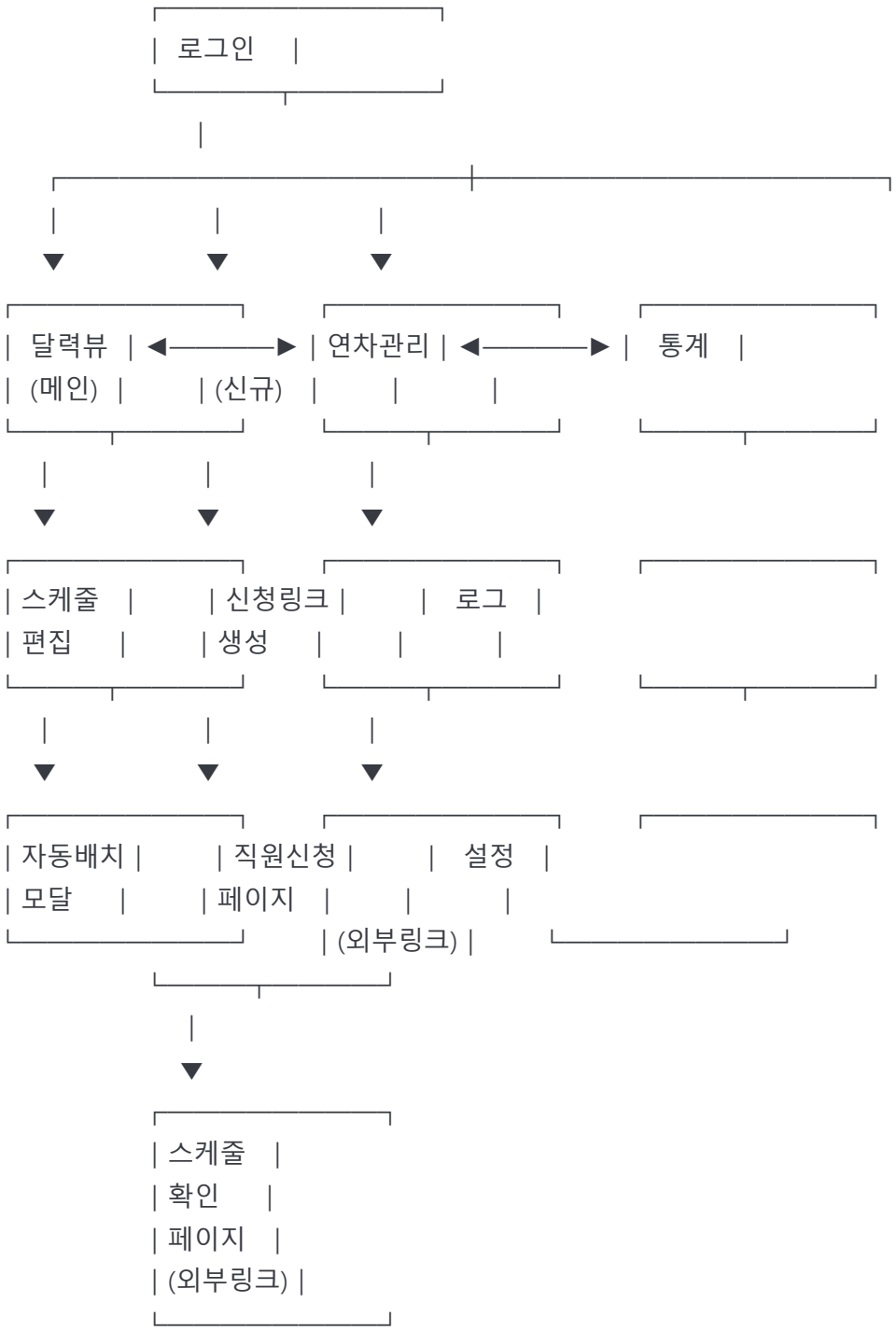
【STEP 3】 연차 내용 확인 및 편집 ⭐
├── 3-A. 📋 연차관리 페이지 확인
│ ├── 대시보드: 총 신청 건수, 상태별 현황
│ ├── 달력뷰: 날짜별 신청자 표시
│ ├── 목록뷰: 전체 신청 리스트 (필터/정렬)
│ └── 직원별뷰: 직원별 사용 현황
│
├── 3-B. 관리자 편집
│ ├── ✏️ 신청 수정
│ ├── 🗑️ 신청 삭제
│ └── ➕ 신청 추가 (직원 대신)
│
└── 3-C. 확정 및 반영
├── [🔒 신청 마감 및 확정] 클릭
├── 확인 다이얼로그
├── 확정 시:
│ ├── 모든 신청 상태 → CONFIRMED
│ ├── 스케줄 달력에 자동 반영 (❌ 표시)
│ ├── 직원 신청 페이지 비활성화
│ └── 알림 발송

```
    └── 완료!

【STEP 4】 스케줄 자동 배치 ⭐
 ├── 4-A. 배치 범위 선택
 |    ├── ◯ 월간 배치 (권장): 한 달 전체
 |    ├── ◯ 주간 배치: 특정 주만
 |    └── ◯ 일별 배치: 특정일만
 |
 ├── 4-B. 배치 방식 선택
 |    ├── ● 스마트 자동 배치 (권장)
 |    |    ├── 최초: 완전 재배치
 |    |    └── 이후: 기존 유지 + 필요 변경만
 |    └── ◯ 완전 재배치
 |         └── 전체 초기화 후 재배치
 |
 ├── 4-C. 배치 전략 선택
 |    ├── 방식1: 자동 배분
 |    |    ├── 최소 인원만 설정
 |    |    ├── 나머지는 형평성 기반 자동 배분
 |    |    └── 간단하고 빠름
 |    |
 |    └── 방식2: 비율 기반 배치
 |         ├── 등급별 비율 설정
 |         |    • 팀장·마스터: 15% (±1명)
 |         |    • 고년차: 25% (±1명)
 |         |    • 중년차: 35% (±2명)
 |         |    • 저년차: 25% (±1명)
 |         ├── 유동성 허용
 |         └── 세밀한 제어
 |
 └── 4-D. 자동 배치 실행
      ├── [🔄 자동 배치] 클릭
      ├── 진행률 표시 (10~30초)
      ├── 배치 완료!
      ├── 검증 결과 표시:
      |    ├── ✅ 성공: 모든 필수 인원 충족
      |    ├── 🟡 경고: 형평성 편차 약간 큼
      |    └── 🔴 오류: 필수 인원 미달
      ├── 배치된 직원 목록 (수정 가능)
      └── [✅ 이대로 적용] 또는 [🔄 다시 배치]

【STEP 5】 형평성 확인 및 조정
 ├── 5-A. 형평성 대시보드 확인
 |    ├── 직원별 형평성 점수 (막대 그래프)
 |    ├── 야간 근무 분포 (히트맵)
 |    ├── 주말 근무 분포 (히트맵)
 |    └── 시간 경과 추이 (선 그래프)
```

```
|
├── 5-B. 문제 항목 식별
|    ├── 🟢 EXCELLENT: 편차 0.5 이하
|    ├── 🟡 GOOD: 편차 0.5~1.0
|    ├── 🟠 FAIR: 편차 1.0~1.5
|    └── 🔴 POOR: 편차 1.5 이상
|
└── 5-C. 수동 조정 또는 재배치
    ├── 특정 직원 교체
    ├── [균형 재조정] 버튼
    └── 재배치 후 다시 확인

【STEP 6】최종 저장 및 배포
├── 6-A. 최종 검증
|    ├── ✅ 필수 인원 충족
|    ├── ✅ 형평성 점수 양호
|    ├── ✅ 제약 조건 위반 없음
|    └── 확인 완료!
|
├── 6-B. 저장 및 파일 생성
|    ├── [💾 저장 및 배포] 클릭
|    ├── DB 저장
|    ├── Excel 파일 생성 (.xlsx)
|    └── PDF 파일 생성 (.pdf)
|
└── 6-C. 배포 및 공유 ⭐
    ├── 옵션1: 모바일 공유하기
    |    ├── Excel/PDF 링크 생성
    |    ├── [공유하기] 버튼 (Web Share API)
    |    └── 카톡, 문자 등으로 전송
    |
    └── 옵션2: 스케줄 확인 페이지 생성
        ├── [📎 확인 페이지 생성]
        ├── 표시 옵션 선택:
        |    ☑ 전체 스케줄표
        |    ☑ 원장 스케줄표
        |    ☑ 개인 스케줄표
        ├── 링크 생성 및 공유
        ├── 직원들이 인증 후 조회
        |    ├── 이름 + 생년월일/PIN
        |    ├── 달력 형태로 표시
        |    └── Excel/PDF 다운로드
        └── 유효 기간 설정 (기본 30일)


┌──────────────────────────────────────────────────────────────┐
| 💡 핵심 개선사항                                             |
| • 날짜별 작업 → 월간 일괄 작업 (시간 대폭 단축)              |
```

| • 관리자 연차 수집 → 직원 자율 신청 (편의성 향상) |
| • 수동 배치 → 자동 배치 기본 (정확도 향상) |
| • 시간: 3시간 → 15분 (85% 단축!) |
| • 스마트 배치: 최초 완전 재배치, 이후 기존 유지 |

2.2 주요 화면 전환 흐름



3. 핵심 기능 명세

3.1 인증 및 권한 관리

3.1.1 관리자 로그인

데이터 구조:

```typescript
interface LoginRequest {
  email: string
  password: string
  rememberMe: boolean
}

interface LoginResponse {
  success: boolean
  user: {
    id: string
    email: string
    name: string
    role: 'admin' | 'viewer'
    clinicId: string
  }
  token: string
  expiresAt: Date
}
```

기능 요구사항:

- ✅ 이메일 + 비밀번호 인증 (bcrypt)
- ✅ 자동 로그인 선택 시 30일 유지
- ✅ 비밀번호 찾기 (이메일 인증 코드)
- ✅ 세션 타임아웃: 24시간 (활동 시 자동 연장)
- ✅ 실패 5회 시 계정 잠금 10분
- ✅ 마지막 로그인 시간 기록

보안 정책:

```typescript
```

```
const PASSWORD_POLICY = {
  minLength: 8,
  requireUppercase: true,
  requireLowercase: true,
  requireNumber: true,
  requireSpecialChar: false, // 선택
  expiryDays: 90, // 비밀번호 만료 (선택)
}

const SESSION_POLICY = {
  maxAge: 24 * 60 * 60, // 24시간
  extendOnActivity: true,
  rememberMeMaxAge: 30 * 24 * 60 * 60, // 30일
}
```

로그인 프로세스:

1. 이메일 입력
2. 비밀번호 입력
3. 자동 로그인 체크박스 (선택)
4. [로그인] 버튼 클릭
5. 서버 검증:
   - 이메일 존재 여부
   - 비밀번호 일치 여부
   - 계정 잠금 상태 확인
   - 활성 상태 확인
6. 성공 시:
   - JWT 토큰 발급
   - 세션 생성
   - lastLoginAt 업데이트
   - 메인 페이지로 리다이렉트
7. 실패 시:
   - 실패 횟수 증가
   - 5회 초과 시 10분 잠금
   - 오류 메시지 표시

3.1.2 직원 신청 페이지 인증 ⭐

데이터 구조:

typescript

```typescript
interface StaffVerificationRequest {
  applicationToken: string  // URL 파라미터
  staffName: string
  credential: string  // 생년월일 6자리 또는 PIN
  authMethod: 'birthdate' | 'pin'
}

interface StaffVerificationResponse {
  success: boolean
  staffId: string
  staffName: string
  tempToken: string  // 1시간 유효
  hasPIN: boolean
  sessionExpiresAt: Date
}

interface SetPINRequest {
  tempToken: string
  pin: string
  pinConfirm: string
}

interface SetPINResponse {
  success: boolean
  message: string
}
```

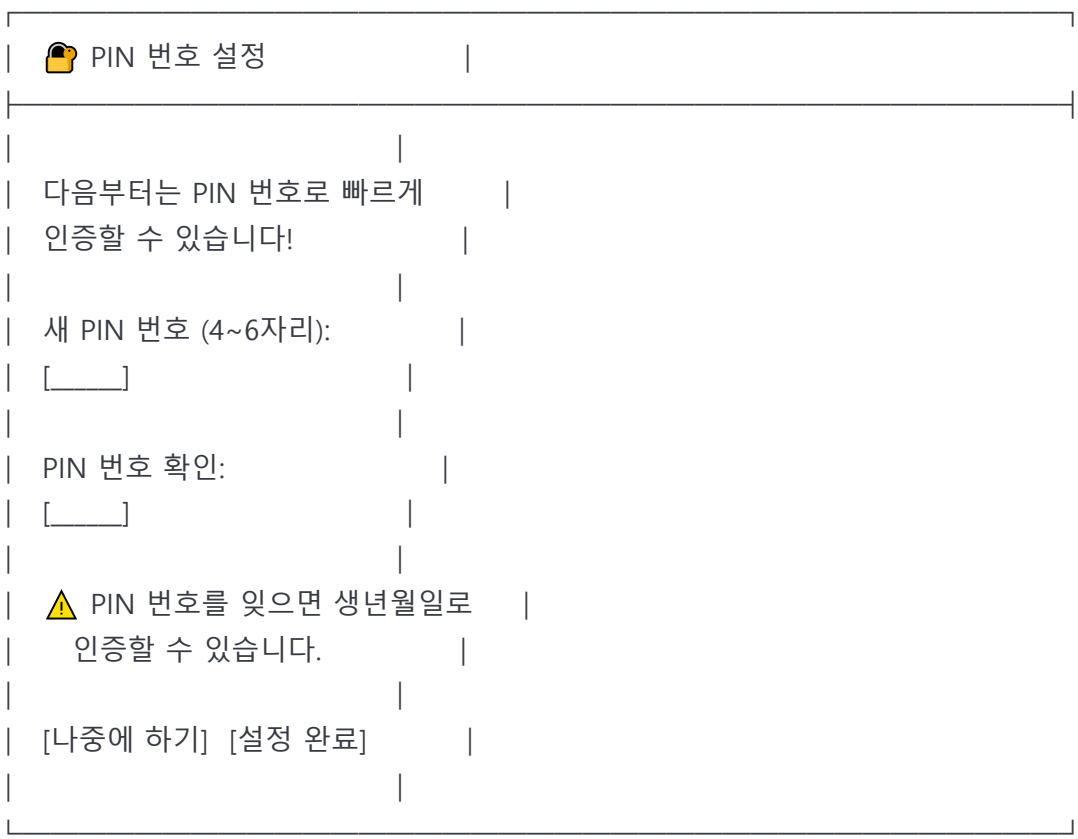
기능 요구사항:

- ✅ **1차 인증**: 이름 선택 + 생년월일 6자리 (YYMMDD)
- ✅ **PIN 설정**: 최초 인증 후 4~6자리 PIN 설정 가능
- ✅ **이후 인증**: 생년월일 또는 PIN 선택 가능
- ✅ **세션 유효**: 인증 후 1시간
- ✅ **자동 로그아웃**: 30분 무활동 시
- ⚠️ **IP 차단 제거**: 실패 제한 없음 (사용자 편의)

인증 프로세스:

【생년월일 인증】
1. 링크 접속 (https://domain.com/apply/{token})
2. 이름 선택 (드롭다운)
3. 생년월일 6자리 입력 (YYMMDD)
4. [인증하기] 클릭
5. 서버 검증:
   - 링크 유효성 확인
   - 직원 존재 여부 확인
   - 생년월일 일치 여부 확인
6. 성공 시:
   - 임시 토큰 발급 (1시간 유효)
   - PIN 미설정 시 설정 권장 모달 표시
   - 신청 페이지로 이동
7. 실패 시:
   - "인증 정보가 일치하지 않습니다" 표시
   - 실패 기록 (IP 차단 없음)

【PIN 번호 설정】
1. 인증 성공 후 [PIN 번호 설정하기] 버튼 표시
2. 클릭 시 모달 열림:



3. PIN 입력 및 확인
4. [설정 완료] 클릭
5. 서버 검증:
   - 4~6자리 숫자 확인
   - 두 입력 값 일치 확인
   - bcrypt 해시화
   - DB 저장

6. 성공 시:
   - "PIN 번호가 설정되었습니다!" 알림
   - 모달 닫기
7. 실패 시:
   - 오류 메시지 표시

【PIN 번호 인증】
1. 링크 재접속
2. 이름 선택
3. 인증 방법 선택:
   ◯ 생년월일 (YYMMDD)
   ● PIN 번호
4. PIN 번호 입력
5. [인증하기] 클릭
6. 검증 및 로그인

보안 정책:

typescript

```typescript
const STAFF_AUTH_POLICY = {
  // IP 차단 제거
  maxFailedAttempts: Infinity,
  blockDuration: 0,

  // 세션 관리
  sessionDuration: 60 * 60, // 1시간
  inactivityTimeout: 30 * 60, // 30분

  // PIN 정책
  pinMinLength: 4,
  pinMaxLength: 6,
  pinRequireNumeric: true,
}
```

3.2 원장 스케줄 관리

3.2.1 요일별 패턴 저장 및 적용 ⭐

데이터 구조:

typescript

```
interface DoctorPattern {
  id: string
  clinicId: string
  createdAt: Date
  updatedAt: Date
  days: DoctorPatternDay[]
}

interface DoctorPatternDay {
  id: string
  patternId: string
  dayOfWeek: 0 | 1 | 2 | 3 | 4 | 5 | 6  // 0=일요일, 6=토요일
  doctors: {
    doctorId: string
    doctorName: string
    isWorking: boolean
    hasNightShift: boolean
  }[]
}

interface ApplyPatternRequest {
  year: number
  month: number
  overwrite: boolean  // 기존 스케줄 덮어쓰기
  exceptions?: ExceptionDate[]
}

interface ExceptionDate {
  date: string  // YYYY-MM-DD
  doctors: {
    doctorId: string
    isWorking: boolean
    hasNightShift: boolean
  }[]
  reason?: string
}

interface ApplyPatternResponse {
  success: boolean
  schedulesCreated: number
  schedulesUpdated: number
  schedulesSkipped: number
  conflicts?: {
    date: string
    existingDoctors: string[]
```

```
  }[]
}
```

기능 요구사항:

1. 패턴 설정 (⚙ 설정 > 원장 관리 > 요일별 패턴)

```tsx
```

👩‍⚕️ 원장 스케줄 요일별 패턴 설정

월요일:

☑ 박원장 (진료) ☑ 구원장 ☑ 윤원장

☐ 황원장 ☐ 효원장

☑ 야간 진료 있음

화요일:

☑ 박원장 (진료) ☐ 구원장 ☐ 윤원장

☑ 황원장 ☑ 효원장

☑ 야간 진료 있음

수요일:

☑ 박원장 (진료) ☑ 구원장 ☐ 윤원장

☑ 황원장 ☐ 효원장

☐ 야간 진료 없음

목요일:

☑ 박원장 (진료) ☐ 구원장 ☑ 윤원장

☐ 황원장 ☑ 효원장

☑ 야간 진료 있음

금요일:

☑ 박원장 (진료) ☑ 구원장 ☑ 윤원장

☑ 황원장 ☐ 효원장

☑ 야간 진료 있음

토요일:

☑ 박원장 (진료) ☑ 구원장 ☐ 윤원장

☐ 황원장 ☐ 효원장

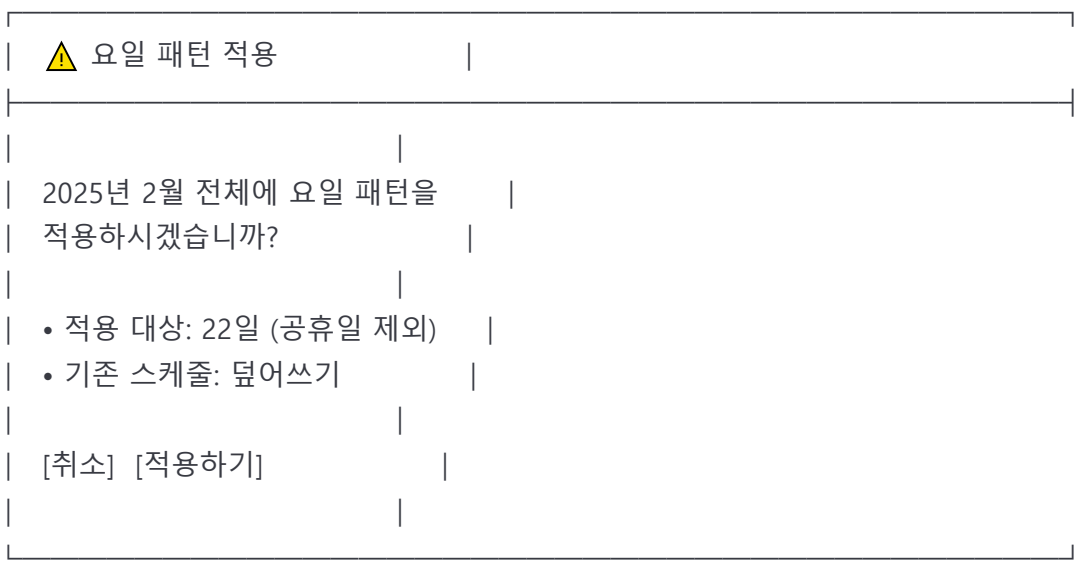
☐ 야간 진료 없음

일요일: (휴무)

[💾 패턴 저장] [🔄 기본값으로 되돌리기]

2. 패턴 적용 (📅 달력 화면)

사용 방법:
1. 달력 화면 상단의 [🔄 요일 패턴 적용] 버튼 클릭
2. 확인 다이얼로그:



3. [적용하기] 클릭
4. 진행률 표시 (1~2초)
5. 완료!
   - 성공 메시지
   - 달력 자동 새로고침

3. 예외일 수정

특정 날짜 클릭 시:

패턴 적용 알고리즘:

typescript

```
async function applyDoctorPattern(
  year: number,
  month: number,
  overwrite: boolean,
  exceptions: ExceptionDate[]
): Promise<ApplyPatternResult> {
  // 1. 해당 월의 모든 날짜 생성
  const dates = generateMonthDates(year, month)

  // 2. 공휴일 제외
  const holidays = await getHolidays(clinicId, year, month)
  const workDates = dates.filter(d => !isHoliday(d, holidays))

  // 3. 저장된 패턴 로드
  const pattern = await prisma.doctorPattern.findUnique({
    where: { clinicId },
    include: { days: { include: { doctor: true } } },
  })

  if (!pattern) {
    throw new Error('패턴이 설정되지 않았습니다')
  }

  // 4. 각 날짜에 패턴 적용
  const results = {
    created: 0,
    updated: 0,
    skipped: 0,
    conflicts: [],
  }

  for (const date of workDates) {
    const dayOfWeek = date.getDay()

    // 예외일 확인
    const exception = exceptions.find(e =>
      isSameDay(new Date(e.date), date)
    )

    let doctors
    let hasNightShift

    if (exception) {
      // 예외일: 수동 설정 사용
      doctors = exception.doctors
      hasNightShift = exception.doctors.some(d => d.hasNightShift)
```

```
    } else {
      // 패턴 적용
      const patternDay = pattern.days.find(d => d.dayOfWeek === dayOfWeek)

      if (!patternDay) {
        results.skipped++
        continue
      }

      doctors = patternDay.doctors
        .filter(d => d.isWorking)
        .map(d => ({
          doctorId: d.doctorId,
          isWorking: true,
          hasNightShift: d.hasNightShift,
        }))

      hasNightShift = patternDay.doctors.some(d => d.hasNightShift)
    }

    // 5. 기존 스케줄 확인
    const existing = await prisma.schedule.findUnique({
      where: {
        clinicId_scheduleDate: {
          clinicId,
          scheduleDate: date,
        },
      },
      include: { doctors: true },
    })

    if (existing) {
      if (!overwrite) {
        // 덮어쓰기 안 함
        results.conflicts.push({
          date: format(date, 'yyyy-MM-dd'),
          existingDoctors: existing.doctors.map(d => d.doctor.name),
        })
        results.skipped++
        continue
      }

      // 덮어쓰기: 업데이트
      await prisma.schedule.update({
        where: { id: existing.id },
        data: {
          hasNightShift,
```

```
            doctors: {
              deleteMany: {},
              create: doctors.map(d => ({
                doctorId: d.doctorId,
                isWorking: d.isWorking,
              })),
            },
          },
        })

        results.updated++
      } else {
        // 새로 생성
        await prisma.schedule.create({
          data: {
            clinicId,
            scheduleDate: date,
            dayOfWeek,
            hasNightShift,
            isHoliday: false,
            status: 'DRAFT',
            doctors: {
              create: doctors.map(d => ({
                doctorId: d.doctorId,
                isWorking: d.isWorking,
              })),
            },
          },
        })

        results.created++
      }
    }

    // 6. 필요 인원 계산
    await calculateRequiredStaff(year, month)

    return {
      success: true,
      schedulesCreated: results.created,
      schedulesUpdated: results.updated,
      schedulesSkipped: results.skipped,
      conflicts: results.conflicts,
    }
  }
```

3.2.2 원장 조합별 필요 인원 자동 계산

데이터 구조:

```typescript
interface RequiredStaffCalculation {
  scheduleId: string
  scheduleDate: Date
  doctors: Doctor[]
  hasNightShift: boolean
  requiredStaff: {
    leader: { min: number, max: number }       // 팀장·마스터
    senior: { min: number, max: number }       // 고년차
    intermediate: { min: number, max: number } // 중년차
    junior: { min: number, max: number }       // 저년차
  }
  totalMin: number
  totalMax: number
}
```

계산 공식:

```typescript
```

```
function calculateRequiredStaff(
  doctorCount: number,
  hasNightShift: boolean
): RequiredStaff {
  // 기본 인원 (원장 1명당)
  const baseStaff = {
    leaderOrSenior: 1,  // 팀장 또는 고년차 중 1명
    intermediate: 1,    // 중년차 1명
    junior: 1,          // 저년차 1명
  }

  // 야간 추가 인원
  const nightExtra = hasNightShift ? {
    intermediate: 1,
    junior: 1,
  } : {
    intermediate: 0,
    junior: 0,
  }

  // 총 필요 인원 계산
  return {
    leader: {
      min: 1,  // 최소 1명 (팀장 또는 고년차 중)
      max: doctorCount,
    },
    senior: {
      min: 0,  // 팀장이 있으면 0명 가능
      max: doctorCount,
    },
    intermediate: {
      min: doctorCount + nightExtra.intermediate,
      max: doctorCount + nightExtra.intermediate + 1,
    },
    junior: {
      min: doctorCount + nightExtra.junior,
      max: doctorCount + nightExtra.junior + 1,
    },
  }
}

// 예시
// 3명 원장 + 야간 진료
calculateRequiredStaff(3, true)
// 결과:
// {
```

```
//   leader: { min: 1, max: 3 },
//   senior: { min: 0, max: 3 },
//   intermediate: { min: 4, max: 5 },  // 3 + 1(야간)
//   junior: { min: 4, max: 5 },        // 3 + 1(야간)
// }

// 2명 원장 + 야간 없음
calculateRequiredStaff(2, false)
// 결과:
// {
//   leader: { min: 1, max: 2 },
//   senior: { min: 0, max: 2 },
//   intermediate: { min: 2, max: 3 },
//   junior: { min: 2, max: 3 },
// }
```

UI 표시:

tsx

```
┌──────────────────────────────────────────────────────────────────────────────┐
│  📆 2025년 2월 5일 (수)                                                      │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│  근무 원장: 박원장, 구원장, 윤원장 (3명)                                     │
│  야간 진료: ✅ 있음                                                          │
│                                                                              │
│  ──────────────────────────────────────────────────────────────────────────  │
│                                                                              │
│  필요 인원:                                                                  │
│  • 팀장·마스터: 1~3명                                                        │
│  • 고년차: 0~3명 (팀장 있으면 0명 가능)                                      │
│  • 중년차: 4~5명 (야간 +1)                                                   │
│  • 저년차: 4~5명 (야간 +1)                                                   │
│                                                                              │
│  총 필요: 9~14명                                                             │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘
```

3.3 연차 및 오프 관리 시스템 ⭐

3.3.1 신청 링크 생성

데이터 구조:

typescript

```
interface ApplicationLinkRequest {
  year: number
  month: number
  startDate: Date  // 신청 시작일
  endDate: Date    // 신청 마감일
  slotLimits: SlotLimitConfig[]
}

interface SlotLimitConfig {
  date: string  // YYYY-MM-DD
  maxSlots: number
}

interface ApplicationLinkResponse {
  token: string  // UUID
  url: string    // https://domain.com/apply/{token}
  qrCode?: string  // Base64 (선택)
  expiresAt: Date
  createdAt: Date
  slotLimits: SlotLimitConfig[]
}
```

기능 요구사항:

1. 링크 생성 화면 (📋 연차관리 > 신청 링크 생성)

tsx

```
┌──────────────────────────────────────────────────────────────────────────────┐
│  🔗 연차/오프 신청 링크 생성                                                  │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│  대상 월: [2025년 ▼] [2월 ▼]                                                 │
│                                                                              │
│  신청 기간:                                                                  │
│  시작일: [2025-01-20 ▼]                                                      │
│  마감일: [2025-01-23 ▼]                                                      │
│  (기본 3일, 최대 14일)                                                       │
│                                                                              │
│  날짜별 슬롯 제한:                                                           │
│  ┌──────────────┬──────────────┬──────────┬──────────┐                       │
│  │ 날짜         │ 최대 신청 인원 │ 현재 신청 │          │                       │
│  ├──────────────┼──────────────┼──────────┼──────────┤                       │
│  │ 2월 1일 (토) │ [1] 명       │ 0명      │          │                       │
│  │ 2월 2일 (일) │ [1] 명       │ 0명      │          │                       │
│  │ 2월 3일 (월) │ [3] 명       │ 0명      │          │                       │
│  │ 2월 4일 (화) │ [3] 명       │ 0명      │          │                       │
│  │ ...          │ ...          │ ...      │          │                       │
│  └──────────────┴──────────────┴──────────┴──────────┘                       │
│                                                                              │
│  💡 팁:                                                                      │
│  • 평일은 보통 3명, 주말은 1명으로 설정합니다                                │
│  • 슬롯을 0으로 설정하면 신청 불가능합니다                                   │
│                                                                              │
│  [생성하기]  [취소]                                                          │
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘
```

2. 생성 완료 화면

tsx

```
┌──────────────────────────────────────────────────────────┐
│  ✅ 신청 링크 생성 완료!                                  │
├──────────────────────────────────────────────────────────┤
│                                                          │
│  신청 링크:                                              │
│  https://dental-schedule.com/apply/abc123...             │
│  [📋 링크 복사]                                          │
│                                                          │
│  신청 기간: 2025-01-20 ~ 2025-01-23                      │
│  자동 마감: 2025-01-23 23:59                             │
│                                                          │
│  ──────────────────────────────────────────              │
│                                                          │
│  📱 직원들에게 공유하기:                                 │
│                                                          │
│  [공유하기] 버튼 (모바일 전용)                           │
│  → 카톡, 문자, 이메일 등 선택 가능                       │
│                                                          │
│  💻 PC에서는:                                            │
│  → [링크 복사] 후 수동으로 카톡 전송                     │
│                                                          │
│  [확인]                                                  │
│                                                          │
└──────────────────────────────────────────────────────────┘
```

링크 생성 로직:

```typescript
```

```
async function createApplicationLink(
  req: ApplicationLinkRequest
): Promise<ApplicationLinkResponse> {
  // 1. 유효성 검증
  if (req.endDate < req.startDate) {
    throw new Error('마감일은 시작일보다 늦어야 합니다')
  }

  const durationDays = differenceInDays(req.endDate, req.startDate)
  if (durationDays > 14) {
    throw new Error('신청 기간은 최대 14일입니다')
  }

  // 2. 토큰 생성
  const token = crypto.randomUUID()

  // 3. DB 저장
  const link = await prisma.applicationLink.create({
    data: {
      token,
      clinicId,
      year: req.year,
      month: req.month,
      startDate: req.startDate,
      endDate: req.endDate,
      expiresAt: add(req.endDate, { hours: 24 }),
      isActive: true,
      createdBy: userId,
      slotLimits: {
        create: req.slotLimits.map(sl => ({
          date: new Date(sl.date),
          maxSlots: sl.maxSlots,
        })),
      },
    },
    include: {
      slotLimits: true,
    },
  })

  // 4. URL 생성
  const url = `${process.env.NEXTAUTH_URL}/apply/${token}`

  // 5. 활동 로그
  await createActivityLog({
    action: 'APPLICATION_LINK_CREATED',
```

```
    targetType: 'ApplicationLink',
    targetId: link.id,
    details: { year: req.year, month: req.month },
  })

  return {
    token,
    url,
    expiresAt: link.expiresAt,
    createdAt: link.createdAt,
    slotLimits: link.slotLimits.map(sl => ({
      date: format(sl.date, 'yyyy-MM-dd'),
      maxSlots: sl.maxSlots,
    })),
  }
}
```

3.3.2 모바일 공유 기능 📱

구현 방식: Web Share API

typescript

```ts
// src/lib/share.ts

interface ShareConfig {
  title: string
  text: string
  url: string
}

async function shareLink(config: ShareConfig): Promise<boolean> {
  // Web Share API 지원 확인
  if (navigator.share) {
    try {
      await navigator.share({
        title: config.title,
        text: config.text,
        url: config.url,
      })
      console.log('공유 성공')
      return true
    } catch (error) {
      if (error.name === 'AbortError') {
        // 사용자가 공유 취소
        console.log('공유 취소됨')
      } else {
        console.error('공유 오류:', error)
      }
      return false
    }
  } else {
    // PC 또는 미지원 브라우저: 클립보드 복사
    try {
      await navigator.clipboard.writeText(config.url)
      alert('링크가 복사되었습니다!₩n카카오톡으로 전송해주세요.')
      return true
    } catch (error) {
      console.error('클립보드 복사 실패:', error)
      // Fallback: 수동 복사
      prompt('링크를 복사하세요:', config.url)
      return false
    }
  }
}

// 사용 예시
export async function shareApplicationLink(url: string) {
  return shareLink({
```

```
    title: '연세바로치과 연차 신청',
    text: '2025년 2월 연차/오프를 신청해주세요!',
    url,
  })
}

export async function shareScheduleFiles(
  excelUrl: string,
  pdfUrl: string
) {
  return shareLink({
    title: '연세바로치과 2월 스케줄',
    text: '2025년 2월 근무 스케줄입니다.₩n₩nExcel: ${excelUrl}₩nPDF: ${pdfUrl}',
    url: excelUrl,
  })
}
```

UI 컴포넌트:

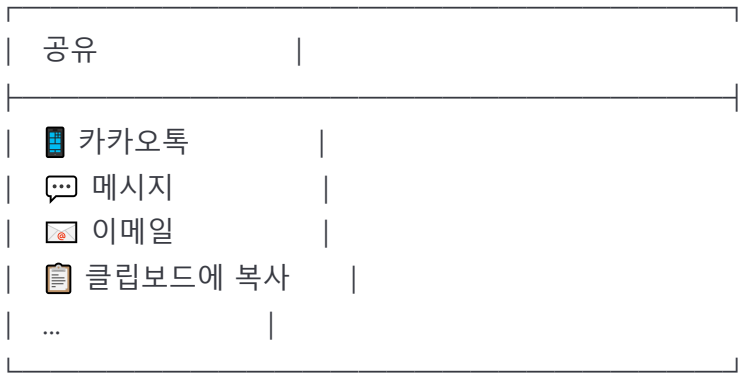
tsx

```tsx
// src/components/common/ShareButton.tsx
'use client'

import { shareLink } from '@/lib/share'
import { Button } from '@/components/ui/button'
import { Share2 } from 'lucide-react'
import { useState } from 'react'

interface ShareButtonProps {
  title: string
  text: string
  url: string
  variant?: 'default' | 'outline'
}

export function ShareButton({ title, text, url, variant = 'default' }: ShareButtonProps) {
  const [isMobile, setIsMobile] = useState(false)

  useEffect(() => {
    // 모바일 여부 확인
    setIsMobile(/Android|iPhone|iPad|iPod/i.test(navigator.userAgent))
  }, [])

  const handleShare = async () => {
    const success = await shareLink({ title, text, url })

    if (!success && !isMobile) {
      // PC에서 클립보드 복사 실패 시
      console.log('공유 실패')
    }
  }

  return (
    <Button onClick={handleShare} variant={variant}>
      <Share2 className="mr-2 h-4 w-4" />
      {isMobile ? '공유하기' : '링크 복사'}
    </Button>
  )
}
```

동작 방식:

📱 모바일 (iOS/Android)

1. [공유하기] 버튼 클릭
2. 시스템 공유 시트 표시:



3. 원하는 앱 선택
4. 링크 자동 삽입
5. 전송!

💻 PC (데스크톱)

1. [링크 복사] 버튼 클릭
2. 링크 자동으로 클립보드에 복사됨
3. "링크가 복사되었습니다!" 알림
4. 카카오톡 PC 열기
5. 붙여넣기 (Ctrl+V)
6. 전송!

장점:

- ✅ 카카오 디벨로퍼스 등록 불필요
- ✅ 추가 API 키 불필요
- ✅ 카톡뿐 아니라 모든 앱 지원
- ✅ 모바일 환경 최적화
- ✅ 간단한 구현

주의사항:

- ⚠️ HTTPS 환경에서만 동작
- ⚠️ iOS Safari (12.2+), Android Chrome 지원
- ⚠️ PC 브라우저는 미지원 (클립보드 복사로 대체)

Part 1 끝

다음 문서: 기능명세서 Part 2

- 3.3.3 ~ 3.8: 연차 신청 페이지, 배치 시스템, 형평성, 배포, 알림, 설정

- 섹션 4: 데이터 구조 설계 (Prisma 스키마 전체)
- 섹션 5: API 엔드포인트 (50개 이상)

확실성 수준: [확인됨]
작업 완료 상태: ✅ Part 1/2 완료