Prompt Design

WorkFlow - AI 기반 개발을 위한 단계별 프롬프트 설계

버전: 1.0
작성일: 2026-02-02
작성자: Development Team
문서 상태: Draft

목차

---

1. 문서 개요

1.1 문서 목적

본 문서는 AI(Claude)가 WorkFlow 시스템을 단계별로 구현할 수 있도록 구조화된 프롬프트를 제공한다. 각 프롬프트는 독립적으로 실행 가능하며, 순차적으로 진행하여 완성된 시스템을 구축한다.

1.2 프롬프트 설계 원칙

- **독립성**: 각 프롬프트는 다른 프롬프트에 의존하지 않고 실행 가능
- **명확성**: 모호함 없이 구체적인 요구사항 명시
- **점진성**: 작은 기능부터 구현하여 최종 완성으로 진행
- **검증 가능**: 각 단계의 완료 기준이 명확

1.3 사용 방법

1. 각 Phase의 프롬프트를 순서대로 복사
2. Claude에게 프롬프트 입력
3. 생성된 코드 확인 및 테스트
4. 다음 프롬프트로 진행

---

2. 개발 마일스톤

전체 개발 로드맵 (4주)

```
Week 1: Phase 1-2 (프로젝트 초기화 + 인증)
 ├── Day 1-2: 프로젝트 구조 및 Supabase 연동
 ├── Day 3-4: 근로자/관리자 인증 시스템
 └── Day 5-7: DB 스키마 구축 및 테스트

Week 2: Phase 3 (출퇴근 기능)
```

```
├── Day 8-10: GPS 기반 출퇴근 체크인/체크아웃
├── Day 11-12: 근태 기록 조회 및 캘린더 UI
└── Day 13-14: 에러 처리 및 오프라인 모드

Week 3: Phase 4 (관리자 대시보드)
├── Day 15-17: 실시간 출퇴근 현황 대시보드
├── Day 18-19: 근로자 관리 CRUD
└── Day 20-21: 근태 기록 조회 및 필터링

Week 4: Phase 5 (급여 관리)
├── Day 22-24: 급여 계산 로직 및 급여대장 생성
├── Day 25-26: 엑셀 내보내기 기능
└── Day 27-28: 통합 테스트 및 배포 준비
```

---

3. Phase 1: 프로젝트 초기화

3.1 Milestone 1: Flutter 프로젝트 생성

프롬프트:

```
WorkFlow 안드로이드/iOS/웹 앱을 위한 Flutter 프로젝트를 생성해주세요.

요구사항:
- 앱 이름: WorkFlow
- 패키지명: com.taekyungholdings.workflow
- Flutter 3.19 이상 사용
- Material Design 3 적용
- 모노레포 구조로 설정 (근로자 앱, 관리자 웹 공통 코드 공유)

프로젝트 구조:
```

workflow/ ├── packages/ | ├── core/ # 공통 모델, 유틸리티 | ├── supabase_client/ # Supabase 연동 | └── ui_components/ # 공통 UI 위젯 ├── apps/ | ├── worker_app/ # 근로자 앱 (iOS/Android) | └── manager_web/ # 관리자 웹 └── pubspec.yaml

```
필요한 주요 dependencies:
- flutter_riverpod: 2.5.1 (상태 관리)
- supabase_flutter: 2.5.0 (백엔드)
- geolocator: 11.0.0 (GPS)
- permission_handler: 11.3.0 (권한)
- intl: 0.19.0 (날짜/시간)
- freezed: 2.5.2 (불변 모델)
- json_serializable: 6.8.0 (JSON)

각 패키지와 앱의 pubspec.yaml을 완성하고, 기본 main.dart 파일을 생성해주세요.
```

기대 산출물:

- `pubspec.yaml` (루트, packages, apps)
- `apps/worker_app/lib/main.dart`
- `apps/manager_web/lib/main.dart`

- `packages/core/lib/core.dart`
- `README.md` (프로젝트 설명)

3.2 Milestone 2: Supabase 연동 설정

프롬프트:

```
Supabase를 WorkFlow 프로젝트에 연동하는 초기 설정을 구현해주세요.

요구사항:
- packages/supabase_client 패키지에 Supabase 클라이언트 래퍼 구현
- 환경 변수 관리 (.env 파일)
- 싱글톤 패턴으로 Supabase 클라이언트 관리
- 에러 핸들링 포함

구현해야 할 파일:
1. packages/supabase_client/lib/src/supabase_service.dart
   - SupabaseService 클래스 (싱글톤)
   - initialize() 메서드
   - client getter

2. packages/supabase_client/lib/src/exceptions.dart
   - 커스텀 예외 클래스 (SupabaseException)
   - 에러 타입별 분류

3. .env.example
   - SUPABASE_URL
   - SUPABASE_ANON_KEY

4. packages/supabase_client/lib/supabase_client.dart
   - export 파일

Riverpod Provider로 SupabaseService를 제공하도록 구현해주세요.
```

기대 산출물:

- `packages/supabase_client/lib/src/supabase_service.dart`
- `packages/supabase_client/lib/src/exceptions.dart`
- `packages/supabase_client/lib/supabase_client.dart`
- `.env.example`

3.3 Milestone 3: 데이터베이스 스키마 구축

프롬프트:

```
Supabase PostgreSQL 데이터베이스에 WorkFlow의 테이블 스키마를 생성하는 SQL 마이그레이션 파일을 작성해주세요.

필요한 테이블:
1. sites (사업장)
   - id (UUID, PK)
   - name (VARCHAR, NOT NULL)
```

```
   - latitude (DECIMAL)
   - longitude (DECIMAL)
   - radius (INTEGER, default 100)
   - created_at, updated_at (TIMESTAMP)

2. parts (파트/직급)
   - id (UUID, PK)
   - name (VARCHAR, UNIQUE, NOT NULL)
   - hourly_wage (INTEGER, NOT NULL)
   - daily_wage (INTEGER, nullable)
   - description (TEXT)
   - created_at, updated_at (TIMESTAMP)

3. workers (근로자)
   - id (UUID, PK)
   - site_id (UUID, FK → sites)
   - part_id (UUID, FK → parts)
   - name (VARCHAR, NOT NULL)
   - phone (VARCHAR, UNIQUE, NOT NULL)
   - role (VARCHAR, CHECK: 'worker' or 'manager')
   - is_active (BOOLEAN, default TRUE)
   - created_at, updated_at (TIMESTAMP)

4. attendances (출퇴근 기록)
   - id (UUID, PK)
   - worker_id (UUID, FK → workers)
   - check_in_time (TIMESTAMP, NOT NULL)
   - check_in_latitude, check_in_longitude (DECIMAL)
   - check_out_time (TIMESTAMP, nullable)
   - check_out_latitude, check_out_longitude (DECIMAL, nullable)
   - work_hours (DECIMAL, 자동 계산)
   - status (VARCHAR, CHECK)
   - notes (TEXT)
   - created_at, updated_at (TIMESTAMP)

5. payrolls (급여대장)
   - id (UUID, PK)
   - worker_id (UUID, FK → workers)
   - year_month (VARCHAR, YYYY-MM)
   - total_work_hours (DECIMAL)
   - total_work_days (INTEGER)
   - base_salary (INTEGER)
   - total_salary (INTEGER)
   - is_finalized (BOOLEAN)
   - finalized_at (TIMESTAMP, nullable)
   - created_at, updated_at (TIMESTAMP)
   - UNIQUE(worker_id, year_month)

추가 요구사항:
- 모든 테이블에 적절한 인덱스 생성
- updated_at 자동 갱신 트리거
- work_hours 자동 계산 트리거 (attendances)
```

```
- Row Level Security (RLS) 정책 설정:
  * workers: 근로자는 자신만, 관리자는 같은 사업장 조회
  * attendances: 근로자는 자신만, 관리자는 같은 사업장 조회

SQL 마이그레이션 파일을 순서대로 작성해주세요.
```

기대 산출물:

- `supabase/migrations/001_create_tables.sql`
- `supabase/migrations/002_create_indexes.sql`
- `supabase/migrations/003_create_triggers.sql`
- `supabase/migrations/004_setup_rls.sql`

3.4 Milestone 4: 도메인 모델 생성

프롬프트:

```
WorkFlow의 도메인 모델을 Freezed를 사용하여 구현해주세요.

packages/core/lib/src/models/ 경로에 다음 모델들을 생성:

1. site_model.dart
   - Site 클래스
   - 필드: id, name, latitude, longitude, radius, createdAt, updatedAt
   - fromJson, toJson 메서드

2. part_model.dart
   - Part 클래스
   - 필드: id, name, hourlyWage, dailyWage, description, createdAt, updatedAt

3. worker_model.dart
   - Worker 클래스
   - 필드: id, siteId, partId, name, phone, role, isActive, createdAt, updatedAt
   - WorkerRole enum (worker, manager)

4. attendance_model.dart
   - Attendance 클래스
   - 필드: id, workerId, checkInTime, checkInLatitude, checkInLongitude,
           checkOutTime?, checkOutLatitude?, checkOutLongitude?,
           workHours?, status, notes?, createdAt, updatedAt
   - AttendanceStatus enum (present, absent, leave, holiday)

5. payroll_model.dart
   - Payroll 클래스
   - 필드: id, workerId, yearMonth, totalWorkHours, totalWorkDays,
           baseSalary, totalSalary, isFinalized, finalizedAt?, createdAt, updatedAt

모든 모델에 Freezed 어노테이션 적용:
- @freezed
- @JsonSerializable()
- copyWith, fromJson, toJson 자동 생성
```

```
packages/core/lib/core.dart에서 모든 모델 export 하도록 설정해주세요.
```

기대 산출물:

- `packages/core/lib/src/models/*.dart` (5개 모델)
- `packages/core/lib/core.dart`
- `build.yaml` (Freezed 설정)

---

4. Phase 2: 인증 시스템 구현

4.1 Milestone 5: 근로자 SMS 인증

프롬프트:

```
근로자용 전화번호 기반 SMS 인증 시스템을 구현해주세요.

구현 위치: apps/worker_app/

요구사항:
1. 인증 Repository 구현
   - lib/features/auth/data/auth_repository.dart
   - sendOtp(String phone) → Future<void>
   - verifyOtp(String phone, String token) → Future<Worker>
   - signOut() → Future<void>
   - currentUser → Stream<Worker?>

2. 인증 상태 관리
   - lib/features/auth/providers/auth_provider.dart
   - Riverpod StateNotifier 사용
   - AuthState (초기/로딩/인증됨/오류)

3. 로그인 화면 UI
   - lib/features/auth/presentation/login_screen.dart
   - 전화번호 입력 (한국 국가코드 +82 고정)
   - SMS 인증번호 입력
   - 유효성 검증 (전화번호 형식)
   - 로딩 인디케이터
   - 에러 메시지 표시

4. 메인 앱 진입점 수정
   - lib/main.dart
   - 인증 상태에 따라 로그인 or 메인 화면 표시
   - ProviderScope 설정

Material Design 3 스타일 적용:
- 큰 버튼 (최소 48dp 높이)
- 명확한 색상 (Primary, Error)
- TextField 아웃라인 스타일

에러 처리:
- 네트워크 오류
```

```
- 잘못된 인증번호
- 타임아웃
- 미등록 사용자
```

기대 산출물:

- `apps/worker_app/lib/features/auth/data/auth_repository.dart`
- `apps/worker_app/lib/features/auth/providers/auth_provider.dart`
- `apps/worker_app/lib/features/auth/presentation/login_screen.dart`
- `apps/worker_app/lib/main.dart` (수정)

4.2 Milestone 6: 관리자 이메일 로그인

프롬프트:

```
관리자용 이메일/비밀번호 로그인 시스템을 구현해주세요.

구현 위치: apps/manager_web/

요구사항:
1. 인증 Repository
   - lib/features/auth/data/auth_repository.dart
   - signInWithEmail(String email, String password) → Future<Worker>
   - signOut() → Future<void>
   - currentUser → Stream<Worker?>
   - 관리자 권한 확인 (role == 'manager')

2. 인증 상태 관리
   - lib/features/auth/providers/auth_provider.dart
   - Riverpod StateNotifier
   - AuthState 관리

3. 로그인 화면 UI (웹 최적화)
   - lib/features/auth/presentation/login_screen.dart
   - 이메일 입력 (유효성 검증)
   - 비밀번호 입력 (마스킹)
   - "로그인 유지" 체크박스
   - 로그인 버튼
   - 에러 메시지 표시

4. 메인 앱 진입점
   - lib/main.dart
   - 인증 상태 기반 라우팅
   - 관리자 권한 확인

웹 UI 스타일:
- 중앙 정렬된 로그인 카드
- 반응형 레이아웃 (최대 400px 너비)
- 깔끔한 폼 디자인
- Material Design 3

에러 처리:
```

```
- 잘못된 이메일/비밀번호
- 관리자 권한 없음
- 네트워크 오류
```

기대 산출물:

- `apps/manager_web/lib/features/auth/data/auth_repository.dart`
- `apps/manager_web/lib/features/auth/providers/auth_provider.dart`
- `apps/manager_web/lib/features/auth/presentation/login_screen.dart`
- `apps/manager_web/lib/main.dart` (수정)

---

5. Phase 3: 출퇴근 기능 구현

5.1 Milestone 7: GPS 위치 서비스

프롬프트:

```
GPS 기반 위치 수집 및 검증 서비스를 구현해주세요.

구현 위치: packages/core/lib/src/services/

요구사항:
1. 위치 서비스
   - location_service.dart
   - getCurrentPosition() → Future<LatLng>
   - checkPermission() → Future<bool>
   - requestPermission() → Future<bool>
   - calculateDistance(LatLng point1, LatLng point2) → double (미터)
   - isWithinRadius(LatLng userLocation, LatLng siteLocation, double radius) → bool
   - GPS 정확도 검증 (<50m)
   - 타임아웃 처리 (10초)

2. 위치 모델
   - models/lat_lng.dart
   - LatLng 클래스 (Freezed)
   - latitude, longitude 필드

3. 위치 예외 처리
   - exceptions/location_exception.dart
   - PermissionDeniedException
   - LocationTimeoutException
   - LowAccuracyException
   - OutOfRangeException

4. Riverpod Provider
   - providers/location_provider.dart
   - LocationService 제공

Geolocator 패키지 사용:
- 위치 정확도: LocationAccuracy.high
- 타임아웃: Duration(seconds: 10)
- Haversine 공식으로 거리 계산
```

```
에러 처리:
- 권한 거부
- 위치 서비스 비활성화
- 타임아웃
- 정확도 낮음
```

기대 산출물:

- `packages/core/lib/src/services/location_service.dart`
- `packages/core/lib/src/models/lat_lng.dart`
- `packages/core/lib/src/exceptions/location_exception.dart`
- `packages/core/lib/src/providers/location_provider.dart`

5.2 Milestone 8: 출근 체크인 기능

프롬프트:

```
근로자 앱의 출근 체크인 기능을 구현해주세요.

구현 위치: apps/worker_app/lib/features/attendance/

요구사항:
1. 출퇴근 Repository
   - data/attendance_repository.dart
   - checkIn(String workerId, LatLng location) → Future<Attendance>
   - getTodayAttendance(String workerId) → Future<Attendance?>
   - 중복 체크인 방지

2. 출퇴근 상태 관리
   - providers/attendance_provider.dart
   - AttendanceState (idle/checking-in/checked-in/error)
   - 오늘 출근 기록 캐싱

3. 메인 화면 UI
   - presentation/home_screen.dart
   - 상단: 사용자 이름, 사업장명
   - 중앙: 큰 "출근" 버튼 (초록색, 최소 120dp)
   - 하단: 오늘 출근 정보 (출근 시간 or 미출근)
   - 로딩 인디케이터
   - 에러 스낵바

4. 출근 플로우
   - 출근 버튼 탭
   - 위치 권한 확인 → 권한 요청 다이얼로그
   - GPS 위치 수집 → 로딩 표시
   - 사업장 거리 검증 → 오류 시 다이얼로그
   - DB 저장 → 성공 피드백 (진동 + 메시지)
   - 출근 완료 상태로 전환

UI/UX:
- 버튼 비활성화 (로딩 중, 이미 출근)
```

```
- 명확한 상태 피드백
- 햅틱 피드백 (진동)
- 시간 포맷: HH:MM:SS (한국 시간)

에러 처리:
- 위치 권한 거부
- GPS 오류
- 사업장 반경 밖
- 중복 체크인
- 네트워크 오류
```

기대 산출물:

- `apps/worker_app/lib/features/attendance/data/attendance_repository.dart`
- `apps/worker_app/lib/features/attendance/providers/attendance_provider.dart`
- `apps/worker_app/lib/features/attendance/presentation/home_screen.dart`

5.3 Milestone 9: 퇴근 체크아웃 기능

프롬프트:

```
근로자 앱의 퇴근 체크아웃 기능을 구현해주세요.

구현 위치: apps/worker_app/lib/features/attendance/

요구사항:
1. Repository 확장
   - data/attendance_repository.dart
   - checkOut(String attendanceId, LatLng location) → Future<Attendance>
   - 출근 기록 존재 확인
   - work_hours 자동 계산 (DB 트리거)

2. 상태 관리 확장
   - providers/attendance_provider.dart
   - 퇴근 상태 추가
   - 근무 시간 실시간 계산

3. UI 업데이트
   - presentation/home_screen.dart
   - 출근 완료 후 "퇴근" 버튼 표시 (파란색)
   - 현재 근무 시간 실시간 표시 (타이머)
   - 퇴근 완료 다이얼로그
     * 출근 시간
     * 퇴근 시간
     * 총 근무 시간
     * 확인 버튼

4. 퇴근 플로우
   - 퇴근 버튼 탭
   - 출근 기록 확인 → 없으면 오류
   - GPS 위치 수집
   - 사업장 거리 검증 (경고만, 강제 가능)
```

```
   - DB 업데이트
   - 근무 요약 다이얼로그
   - 메인 화면 복귀

특별 처리:
- 위치 오류 시 "강제 퇴근" 옵션 (관리자 승인 필요 플래그)
- 근무 시간 포맷: "N시간 M분"
- 자정 넘어가는 근무 처리

에러 처리:
- 출근 기록 없음
- 이미 퇴근함
- 위치 오류 (경고)
```

기대 산출물:

- `apps/worker_app/lib/features/attendance/data/attendance_repository.dart` (확장)
- `apps/worker_app/lib/features/attendance/providers/attendance_provider.dart` (확장)
- `apps/worker_app/lib/features/attendance/presentation/home_screen.dart` (수정)
- `apps/worker_app/lib/features/attendance/presentation/widgets/checkout_summary_dialog.dart`

5.4 Milestone 10: 근태 기록 조회

프롬프트:

```
근로자 앱의 근태 기록 조회 기능을 구현해주세요.

구현 위치: apps/worker_app/lib/features/attendance/

요구사항:
1. Repository 확장
   - data/attendance_repository.dart
   - getMonthlyAttendances(String workerId, int year, int month) → Future<List<Attendance>>
   - getAttendanceDetail(String attendanceId) → Future<Attendance>

2. 상태 관리
   - providers/attendance_history_provider.dart
   - 월별 기록 캐싱
   - 선택된 월 상태 관리

3. 근태 기록 화면 UI
   - presentation/attendance_history_screen.dart
   - 상단: 월 선택기 (이전/다음 달 버튼)
   - 중앙: 캘린더 뷰
     * 출근 날짜: 초록색 점
     * 결근 날짜: 빨간색 점
     * 휴무 날짜: 회색 배경
   - 하단: 월 통계
     * 총 출근 일수
     * 총 근무 시간
   - 날짜 탭 시 상세 화면 이동
```

```
4. 상세 화면
   - presentation/attendance_detail_screen.dart
   - 날짜
   - 출근 시간 / 퇴근 시간
   - 총 근무 시간
   - GPS 위치 (작은 지도)
   - 메모 (있는 경우)

UI 컴포넌트:
- table_calendar 패키지 사용
- Material Design 3 카드 스타일
- 스켈레톤 로딩

에러 처리:
- 로딩 실패
- 빈 데이터
```

기대 산출물:

- `apps/worker_app/lib/features/attendance/data/attendance_repository.dart` (확장)
- `apps/worker_app/lib/features/attendance/providers/attendance_history_provider.dart`
- `apps/worker_app/lib/features/attendance/presentation/attendance_history_screen.dart`
- `apps/worker_app/lib/features/attendance/presentation/attendance_detail_screen.dart`

---

6. Phase 4: 관리자 대시보드

6.1 Milestone 11: 실시간 출퇴근 현황 대시보드

프롬프트:

```
관리자 웹의 실시간 출퇴근 현황 대시보드를 구현해주세요.

구현 위치: apps/manager_web/lib/features/dashboard/

요구사항:
1. Dashboard Repository
   - data/dashboard_repository.dart
   - getTodayAttendanceSummary(String siteId) → Future<AttendanceSummary>
   - subscribeToAttendanceUpdates(String siteId) → Stream<Attendance>
   - Supabase Realtime 구독

2. AttendanceSummary 모델
   - packages/core/lib/src/models/attendance_summary.dart
   - totalWorkers: int
   - checkedInCount: int
   - notCheckedInWorkers: List<Worker>
   - checkedInWorkers: List<WorkerWithAttendance>

3. 상태 관리
   - providers/dashboard_provider.dart
   - 실시간 업데이트 반영
   - Realtime 스트림 구독
```

```
4. 대시보드 UI
   - presentation/dashboard_screen.dart
   - 상단 헤더
     * 오늘 날짜, 사업장명
     * 새로고침 버튼
   - 중앙 통계 카드
     * "출근 N명 / 전체 M명" (큰 숫자)
     * 진행률 바
   - 하단 탭 뷰
     * 미출근 (빨간색)
     * 근무 중 (초록색)
     * 퇴근 완료 (회색)
   - 각 탭: 근로자 목록 테이블
     * 이름, 파트, 출근시간, 상태

5. 실시간 업데이트
   - 새 출근 기록 → 자동 UI 갱신
   - 애니메이션 효과
   - 소리 알림 (옵션)

웹 레이아웃:
- 반응형 그리드
- 카드 기반 디자인
- Material Design 3
- 데이터 테이블 사용

에러 처리:
- Realtime 연결 실패
- 데이터 로딩 오류
```

기대 산출물:

- `apps/manager_web/lib/features/dashboard/data/dashboard_repository.dart`
- `packages/core/lib/src/models/attendance_summary.dart`
- `apps/manager_web/lib/features/dashboard/providers/dashboard_provider.dart`
- `apps/manager_web/lib/features/dashboard/presentation/dashboard_screen.dart`

6.2 Milestone 12: 근로자 관리 CRUD

프롬프트:

```
관리자 웹의 근로자 관리 기능을 구현해주세요.

구현 위치: apps/manager_web/lib/features/workers/

요구사항:
1. Workers Repository
   - data/workers_repository.dart
   - getWorkers(String siteId) → Future<List<Worker>>
   - createWorker(WorkerCreateDto) → Future<Worker>
   - updateWorker(String id, WorkerUpdateDto) → Future<Worker>
```

```
   - deleteWorker(String id) → Future<void>
   - 전화번호 중복 체크

2. DTO 모델
   - packages/core/lib/src/models/worker_dto.dart
   - WorkerCreateDto (name, phone, partId, siteId)
   - WorkerUpdateDto (name?, partId?)

3. 상태 관리
   - providers/workers_provider.dart
   - 근로자 목록 캐싱
   - 검색/필터 상태

4. 근로자 관리 화면
   - presentation/workers_screen.dart
   - 상단: 검색바, "근로자 등록" 버튼
   - 중앙: 근로자 목록 테이블
     * 이름, 전화번호, 파트, 상태 (재직/퇴사)
     * 액션: 수정, 삭제 버튼
   - 검색: 이름, 전화번호
   - 필터: 파트별, 재직/퇴사

5. 근로자 등록/수정 다이얼로그
   - presentation/widgets/worker_form_dialog.dart
   - 이름 입력 (한글, 2-10자)
   - 전화번호 입력 (010-XXXX-XXXX, 중복 체크)
   - 파트 선택 (드롭다운)
   - 유효성 검증
   - 저장/취소 버튼

6. 삭제 확인 다이얼로그
   - presentation/widgets/delete_confirm_dialog.dart
   - 경고 메시지
   - 확인/취소

웹 UI:
- DataTable 위젯 사용
- 페이지네이션 (50개씩)
- 정렬 가능 컬럼
- Modal 다이얼로그

에러 처리:
- 전화번호 중복
- 유효성 검증 실패
- 삭제 실패 (출퇴근 기록 있음)
```

기대 산출물:

- `apps/manager_web/lib/features/workers/data/workers_repository.dart`
- `packages/core/lib/src/models/worker_dto.dart`
- `apps/manager_web/lib/features/workers/providers/workers_provider.dart`
- `apps/manager_web/lib/features/workers/presentation/workers_screen.dart`

- `apps/manager_web/lib/features/workers/presentation/widgets/worker_form_dialog.dart`
- `apps/manager_web/lib/features/workers/presentation/widgets/delete_confirm_dialog.dart`

6.3 Milestone 13: 출퇴근 기록 조회 및 관리

프롬프트:

```
관리자 웹의 출퇴근 기록 조회 및 관리 기능을 구현해주세요.

구현 위치: apps/manager_web/lib/features/attendance_records/

요구사항:
1. Attendance Records Repository
   - data/attendance_records_repository.dart
   - getAttendances(String siteId, DateTime startDate, DateTime endDate) → Future<List<Attendance>>
   - getAttendancesByWorker(String workerId, DateTime startDate, DateTime endDate) →
Future<List<Attendance>>
   - exportToExcel(List<Attendance> attendances) → Future<Uint8List>

2. 상태 관리
   - providers/attendance_records_provider.dart
   - 날짜 범위 필터
   - 근로자 필터
   - 정렬 상태

3. 출퇴근 기록 화면
   - presentation/attendance_records_screen.dart
   - 상단 필터
     * 날짜 범위 선택 (시작일 ~ 종료일)
     * 근로자 선택 (드롭다운, 전체/개별)
     * 파트 필터
     * 상태 필터 (출근/결근/휴무)
   - 중앙: 출퇴근 기록 테이블
     * 날짜, 이름, 파트, 출근시간, 퇴근시간, 근무시간
     * 정렬 가능 (날짜순, 이름순, 시간순)
   - 하단: 엑셀 내보내기 버튼

4. 상세 보기 다이얼로그
   - presentation/widgets/attendance_detail_dialog.dart
   - 근로자 정보
   - 출퇴근 시간
   - GPS 위치 지도
   - 메모

5. 엑셀 내보내기
   - excel 패키지 사용
   - 컬럼: 날짜, 이름, 파트, 출근, 퇴근, 근무시간
   - 헤더 스타일 (볼드)
   - 자동 컬럼 너비

웹 UI:
- 필터는 상단 카드로 배치
```

```
- DataTable + 페이지네이션
- 다운로드 버튼 (Material Icon)

에러 처리:
- 날짜 범위 검증 (최대 3개월)
- 데이터 없음
- 엑셀 생성 실패
```

기대 산출물:

- `apps/manager_web/lib/features/attendance_records/data/attendance_records_repository.dart`
- `apps/manager_web/lib/features/attendance_records/providers/attendance_records_provider.dart`
- `apps/manager_web/lib/features/attendance_records/presentation/attendance_records_screen.dart`
- `apps/manager_web/lib/features/attendance_records/presentation/widgets/attendance_detail_dialog.dart`

---

7. Phase 5: 급여 관리 시스템

7.1 Milestone 14: 급여 계산 로직

프롬프트:

````
급여 계산 로직을 Supabase Edge Function으로 구현해주세요.

구현 위치: supabase/functions/calculate-payroll/

요구사항:
1. Edge Function (Deno/TypeScript)
   - index.ts
   - 입력: { siteId: string, yearMonth: string (YYYY-MM) }
   - 출력: PayrollData[]

2. 계산 로직
   - 해당 월의 모든 출퇴근 기록 조회
   - 근로자별 집계:
     * 총 근무시간 = SUM(work_hours)
     * 출근일수 = COUNT(DISTINCT DATE(check_in_time))
   - 근로자의 파트 조회 → 시급
   - 기본급 계산 = 시급 × 총 근무시간 (반올림)
   - (MVP: 연장/심야/공휴 수당은 0)

3. SQL 쿼리 최적화
   - JOIN으로 한 번에 조회
   - 인덱스 활용
   - 집계 쿼리 효율화

4. 에러 처리
   - 잘못된 년월 형식
   - 데이터 없음
   - DB 오류

5. 응답 형식
```typescript
````
````

```
interface PayrollData {
  workerId: string;
  workerName: string;
  partName: string;
  totalWorkHours: number;
  totalWorkDays: number;
  hourlyWage: number;
  baseSalary: number;
  totalSalary: number; // MVP에서는 baseSalary와 동일
}
```

Deno 환경 설정:

- import_map.json 사용
- Supabase 클라이언트 연동
- CORS 허용

```
**기대 산출물:**
- `supabase/functions/calculate-payroll/index.ts`
- `supabase/functions/calculate-payroll/import_map.json`

---

### 7.2 Milestone 15: 급여대장 생성 및 관리

**프롬프트:**
```

관리자 웹의 급여대장 생성 및 관리 기능을 구현해주세요.

구현 위치: apps/manager_web/lib/features/payroll/

요구사항:

1. Payroll Repository
    - data/payroll_repository.dart
    - calculatePayroll(String siteId, String yearMonth) → Future<List>
    - savePayroll(String siteId, String yearMonth, List data) → Future
    - getPayroll(String siteId, String yearMonth) → Future<List>
    - finalizePayroll(String siteId, String yearMonth) → Future
2. PayrollData 모델
    - packages/core/lib/src/models/payroll_data.dart
    - Edge Function 응답과 동일 구조
3. 상태 관리
    - providers/payroll_provider.dart
    - 계산 중/완료 상태
    - 급여 데이터 캐싱
4. 급여 메뉴 화면
    - presentation/payroll_screen.dart

    - 상단: 년월 선택기 (기본값: 이전 달)
    - "급여대장 생성" 버튼
    - 중앙: 급여대장 테이블
        - 이름, 파트, 출근일수, 총 근무시간, 시급, 기본급, 총 급여
        - 합계 행 (하단)
    - 하단: "엑셀 내보내기", "확정" 버튼
5. 생성 플로우
    - 년월 선택
    - "급여대장 생성" 클릭
    - Edge Function 호출 → 로딩 표시
    - 결과 테이블 표시
    - 수동 수정 가능 (인라인 편집)
    - "저장" 버튼으로 DB 저장
6. 확정 기능
    - "확정" 버튼 클릭
    - 확인 다이얼로그
    - is_finalized = true 설정
    - 확정 후 수정 불가 (읽기 전용)

웹 UI:

- 년월 선택: 드롭다운 또는 DatePicker
- 테이블: 정렬/필터 가능
- 금액: 천 단위 콤마, 우측 정렬
- 합계 행: 볼드, 배경색

에러 처리:

- 미래 월 선택 불가
- 계산 실패
- 이미 확정됨

```
**기대 산출물:**
- `apps/manager_web/lib/features/payroll/data/payroll_repository.dart`
- `packages/core/lib/src/models/payroll_data.dart`
- `apps/manager_web/lib/features/payroll/providers/payroll_provider.dart`
- `apps/manager_web/lib/features/payroll/presentation/payroll_screen.dart`

---

### 7.3 Milestone 16: 급여대장 엑셀 내보내기

**프롬프트:**
```

급여대장을 엑셀 파일로 내보내는 기능을 구현해주세요.

구현 위치: apps/manager_web/lib/features/payroll/

요구사항:

1. Excel Service
    - packages/core/lib/src/services/excel_service.dart
    - generatePayrollExcel(List data, String yearMonth, String siteName) → Future
2. 엑셀 파일 구조
    - 시트명: "급여대장"
    - 헤더 (1행):
        - A: 이름
        - B: 파트
        - C: 출근일수
        - D: 총 근무시간
        - E: 시급
        - F: 기본급
        - G: 총 급여
    - 데이터 행 (2행~)
    - 합계 행 (마지막):
        - A: "합계"
        - C-G: SUM 수식
3. 스타일링
    - 헤더: 볼드, 배경색 (회색), 중앙 정렬
    - 금액 셀: 천 단위 콤마, 우측 정렬
    - 합계 행: 볼드, 배경색 (연한 노랑)
    - 자동 컬럼 너비
4. 다운로드 기능
    - presentation/payroll_screen.dart 수정
    - "엑셀 내보내기" 버튼
    - 파일명: `급여대장_YYYY-MM_사업장명.xlsx`
    - 웹: anchor 다운로드
    - 성공 스낵바

excel 패키지 사용:

- Excel.createExcel()
- Sheet.appendRow()
- CellStyle (bold, backgroundColor, numberFormat)

에러 처리:

- 엑셀 생성 실패
- 빈 데이터

```
**기대 산출물:**
- `packages/core/lib/src/services/excel_service.dart`
- `apps/manager_web/lib/features/payroll/presentation/payroll_screen.dart` (수정)

---
```

```
## 8. 통합 및 마무리

### 8.1 Milestone 17: 네비게이션 및 라우팅

**프롬프트:**
```

앱의 네비게이션과 라우팅을 구현해주세요.

근로자 앱:

- lib/core/router/app_router.dart
- go_router 패키지 사용
- 라우트:
  - / → 로그인 or 홈
  - /home → 메인 화면 (출퇴근)
  - /history → 근태 기록
  - /profile → 프로필 (설정)
- 하단 네비게이션 바 (홈, 기록, 프로필)

관리자 웹:

- lib/core/router/app_router.dart
- 라우트:
  - / → 로그인 or 대시보드
  - /dashboard → 대시보드
  - /workers → 근로자 관리
  - /attendance → 출퇴근 기록
  - /payroll → 급여 관리
- 사이드 네비게이션 드로어

인증 가드:

- 미인증 시 로그인 페이지로 리다이렉트
- 관리자 권한 확인

에러 처리:

- 404 페이지

```
**기대 산출물:**
- `apps/worker_app/lib/core/router/app_router.dart`
- `apps/worker_app/lib/core/widgets/bottom_nav_bar.dart`
- `apps/manager_web/lib/core/router/app_router.dart`
- `apps/manager_web/lib/core/widgets/side_nav_drawer.dart`

---

### 8.2 Milestone 18: 오프라인 모드 및 동기화

**프롬프트:**
```

근로자 앱의 오프라인 모드와 동기화 기능을 구현해주세요.

요구사항:

1. 로컬 저장소 서비스
    - packages/core/lib/src/services/local_storage_service.dart
    - SharedPreferences 사용
    - savePendingAttendance(Attendance) → Future
    - getPendingAttendances() → Future<List>
    - removePendingAttendance(String id) → Future
2. 동기화 서비스
    - packages/core/lib/src/services/sync_service.dart
    - syncPendingAttendances() → Future
    - connectivity_plus 패키지로 네트워크 감지
    - 백그라운드 자동 재시도 (1분, 5분, 10분)
3. 출퇴근 Repository 수정
    - data/attendance_repository.dart
    - 네트워크 오류 시 로컬 저장
    - 앱 시작 시 동기화 시도
4. UI 표시
    - 오프라인 배너 (상단)
    - 동기화 대기 중 표시
    - 동기화 완료 알림

에러 처리:

- 동기화 실패 (최대 3회 재시도)
- 충돌 해결 (서버 우선)

```
**기대 산출물:**
- `packages/core/lib/src/services/local_storage_service.dart`
- `packages/core/lib/src/services/sync_service.dart`
- `apps/worker_app/lib/features/attendance/data/attendance_repository.dart` (수정)
- `apps/worker_app/lib/core/widgets/offline_banner.dart`

---

### 8.3 Milestone 19: 테스트 및 배포 준비

**프롬프트:**
```

앱의 핵심 로직에 대한 단위 테스트를 작성하고 배포를 준비해주세요.

요구사항:

1. 단위 테스트
    - packages/core/test/
    - location_service_test.dart (GPS 거리 계산)
    - attendance_repository_test.dart (출퇴근 로직)

    - payroll_calculation_test.dart (급여 계산)
    - Mockito 사용
2. 위젯 테스트
    - apps/worker_app/test/
    - login_screen_test.dart
    - home_screen_test.dart
3. 배포 설정
    - Android:
        - android/app/build.gradle (버전, 서명)
        - android/app/src/main/AndroidManifest.xml (권한)
    - iOS:
        - ios/Runner/Info.plist (권한 설명)
        - ios/Runner.xcodeproj (서명)
    - Web:
        - web/index.html (메타 태그)
        - web/manifest.json
4. 환경 변수 관리
    - .env.production
    - .env.development
    - flutter_dotenv 설정
5. CI/CD
    - .github/workflows/deploy.yml
    - 자동 테스트
    - 자동 빌드

커버리지 목표:

- 핵심 로직 80% 이상

```
**기대 산출물:**
- `packages/core/test/*.dart`
- `apps/worker_app/test/*.dart`
- `android/app/build.gradle`
- `ios/Runner/Info.plist`
- `.env.production`, `.env.development`
- `.github/workflows/deploy.yml`

---

## 부록

### A. 프롬프트 실행 체크리스트

각 프롬프트 실행 후 확인 사항:
```

```
- [ ] 코드가 에러 없이 컴파일되는가?
- [ ] 요구사항이 모두 구현되었는가?
- [ ] 에러 처리가 적절한가?
- [ ] UI가 디자인 가이드를 따르는가?
- [ ] 기능이 정상 작동하는가?

### B. 다음 단계 참조 표

| 현재 Milestone | 다음 Milestone | 의존성 |
|---|---|---|
| 1 (프로젝트 생성) | 2 (Supabase) | None |
| 2 (Supabase) | 3 (DB 스키마) | Supabase 설정 |
| 3 (DB 스키마) | 4 (모델) | 테이블 구조 |
| 4 (모델) | 5 (근로자 인증) | 모델 정의 |
| 5 (근로자 인증) | 6 (관리자 인증) | 인증 Repository |
| 7 (GPS 서비스) | 8 (출근) | Location Service |
| 8 (출근) | 9 (퇴근) | 출근 Repository |
| 11 (대시보드) | 12 (근로자 관리) | Dashboard UI |
| 14 (급여 계산) | 15 (급여대장) | Edge Function |

### C. 트러블슈팅 가이드

**문제: GPS 정확도 낮음**
- 해결: 타임아웃 증가, 재시도 로직

**문제: Realtime 연결 끊김**
- 해결: 재연결 로직, 폴링 대체

**문제: 엑셀 파일 깨짐**
- 해결: UTF-8 인코딩, Byte Order Mark

---

**문서 승인:**
- [ ] Tech Lead
- [ ] Senior Developer
- [ ] QA Lead

**변경 이력:**
| 버전 | 날짜 | 작성자 | 변경 내용 |
|---|---|---|---|
| 1.0 | 2026-02-02 | Development Team | 초안 작성 |
```