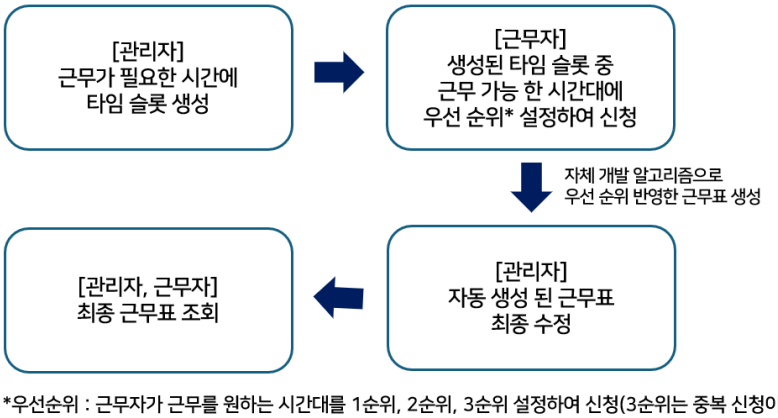
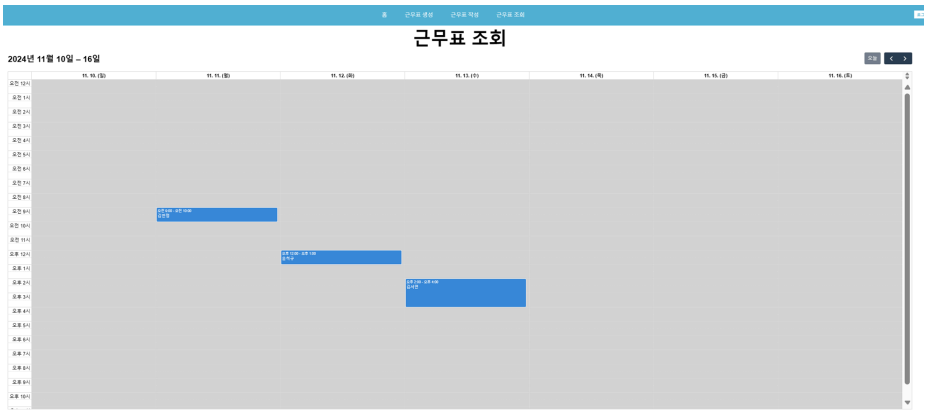
오픈소스SW프로젝트 보고서

학생 팀별 작성용

과제 수행원 현황						
수행 학기	■ 2024-2					
프로젝트명	■ Shiftmate -아르바이트 근무 일정 관리 서비스					
팀명	■ stackOverflowers					
	학과	학번	성명	성별	연락처	**E-mail**
팀장	경제학과	2021110806	김민정	여	010-8794-9031	qwer10897@naver.com
팀원	통계학과	2020113296	윤석규	남	020-2968-5218	kehahahaaaa@gmail.com
	산업시스템공학과	2021112379	김서연	여	010-3538-2391	amyk021224@naver.com
	생명과학과	2020111670	지경민	여	010-9204-1672	jenny1672@naver.com
지도교수	교과목명	■ 오소스SW프로젝트				
	소속	■ SW교육원				
	성명	■ 박효순 교수				

프로젝트	
1. 프로젝트 개요	* 개발동기 및 목적 스케줄 기반 아르바이트에서 파트타임 근무자들의 근무 일정을 수동으로 관리하는 과정에서 발생하는 비효율성과 불편함을 해결하기 위해 본 프로젝트를 시작하게 되었다. 근무 일정이 매번 변동되는 파트타임 근무 환경에서는 근무자들이 선호하는 시간대와 실제 근무 일정이 일치하지 않는 경우가 빈번하다. 고용주가 근무표를 수동으로 작성하는 과정에서 근무자들의

	요청이 반영되지 않거나 오류가 발생해, 원치 않는 비용이 발생하거나 근무자 만족도가 저하되는 상황이 생기기도 한다. 이러한 문제를 해결하고자 본 서비스는 파트타임 근무자들이 선호 시간대를 반영해 공정하고 효율적으로 근무표를 생성하는 자동화된 시스템을 개발하는 것을 목적으로 한다. 이를 통해 고용주와 근무자 모두가 만족할 수 있는 근무 환경을 조성하고, 수동 관리에서 발생하는 시간 낭비와 오류를 최소화하고자 한다. * 개발목표 및 범위: 본 프로젝트의 목표는 파트타임 근무자와 고용주의 요구를 모두 충족시키는 유연하고 공정한 근무 일정 관리 시스템을 개발하는 것이다. ① 관리자의 일정 관리 기능 - 근무 시간 설정: 근무가 필요한 시간대를 관리자가 등록할 수 있도록 한다. - 근무 신청 승인 및 거절 : 근무자들의 희망 근무 신청을 우선순위에 따라 승인 혹은 거절 할 수 있도록 한다. - 근무 시간 수정 및 확정: 등록된 시간대를 수정하거나 확정하여 최종 근무표를 확정할 수 있고 확정된 일정을 근무자와 공유할 수 있다. ② 근무자의 일정 관리 기능 - 희망 근무 시간 신청: 근무자들은 원하는 근무 시간대를 신청할 수 있다. - 근무 일정 확인: 근무자는 확정된 근무표를 통해 신청한 근무가 반영됐는지, 전체 일정표는 어떤지 확인할 수 있다. ③ 선호도에 따른 근무 배정 알고리즘 - 근무자들의 희망 근무 시간을 우선순위로 고려하여 최대한 공정하게 근무 일정을 배정하는 알고리즘을 구현한다.
2. 최종 결과물 소개	스케줄 조정이 잦은 단순 아르바이트 생을 대상으로 효율적으로 근무 일정 배정 및 관리를 할 수 있도록 도와주는 웹서비스 ShiftMate이다. 핵심 기능은 희망하는 근무 일정 자동 배정 기능과 근무 일정 관리 기능이다.  *우선순위 : 근무자가 근무를 원하는 시간대를 1순위, 2순위, 3순위 설정하여 신청(3순위는 중복 신청이 가능) 대표 기능은 알고리즘을 통하여 근무자가 가능한 한 근무를 원하는 시간대에 근무를 할 수 있도록 자동으로 스케줄을 배정해주는 기능이다.



근무표 조회 예시 화면으로, 자동으로 근무표가 완성된 후 근무자는 위와 같은 화면을 통하여 자신의 근무 확정 시간을 확인할 수 있게 된다.

3. 프로젝트 추진내용

3.1 추진 배경

1) 개발 배경 및 필요성 **:**

- 개발 배경 : 다양한 업종의 파트타임 근무자들은 유동적인 근무 일정을 가지고 근무하는 경우가 많다. 영화관, 카페, 병원 등에서 주간 또는 월간 단위로 근무 일정을 조정해야 하는 경우가 일반적이며, 고용주가 수동으로 근무자의 선호 시간을 취합하고 근무표를 작성하는 과정에서 많은 불편함이 발생한다.
- 데이터를 기반으로 최대한 많은 사람들의 요구를 충족시킬 수 있는 효율적인 스케줄을 자동으로 생성하는 프로그램이 필요하다. 또한, 효율적인 인사 관리와 공정한 스케줄링을 지원하는 HR 솔루션에 대한 관심이 높아지고 있다.

2) 선행기술 및 사례 분석**:**

-

서비스 이름	특징	다른 서비스와 차별점	링크	우리 프로젝트와 차이점
시프티(Shiftee)	근무표 작성, 휴가,연차 관리 전자결재, 전자 계약 근태, 급여관리	사내 메세지 대량발송 기능	https://shiftee.io/ko	근무자-관리자 근무 일정 매칭 시스템 부재
플렉스(Flex)	- 근무시간을 입력하면 게이지 바 형태로 시각화. - 출퇴근 기록,주당 근무시간 확인 - 시차, 고정, 선택, 교대 근무 선택	슬랙, 구글캘린더 등 외부 공유 기능 탑재	https://about.flex.team/features/time-tracking	

	기능			
핀플(pinpl)	- 관리자 입장: 출퇴근 현황 관리 연차,야근, 특근,출장 결재 재택근무지 설정 근로계약서, 서약서 작성 각종 증명서 발급 -근무자 입장 : 연차,야근, 특근,출장 신청 간편한 출퇴근 체크 근로시간 관리 재택 출퇴근 영역 추가.	WIFI 기반 출퇴근(출퇴근 지역 설정), 간편한 연차 결재.	https://pinpl.biz/	
위펄슨(wepers on)	-인사 관리를 카드형태로 구현 -조직도 시각화	각종 휴가,연차,출퇴근 등 통계자료 시각화 대시보드	https://www.weperson.com/ko	
듀팅(dutying)	-일반 간호사, 수간호사 용 근무표 작성 -간호사 근무표 자동생성 서비스 -연속 근무 일수 3일 이하 등 제약조건 설정가능	선호 시간 신청 및 신청근무 반영/미반영 기능 표시	https://www.dutying.net/	간호사 직종만을 타겟으로 함. 근무자들이 원하는 근무시간을 우선순위에 따라 설정할 수 있는 기능 X. 이전 근무 신청 이력을 반영해 지난 근무 신청에서 원하는 근무가 배정되지 않은 근무자를 우선 배정하는 기능 X.

기존 선행기술들의 경우 대부분이 알고리즘을 통해 자동으로 근무 배정을 관리해주는 기능이 없었으며, 이 기능이 있는 듀팅의 경우에도 근무 우선순위를 반영해 근무자를 우선 배정하는 기능이 없었다. 또한, 근무가 배정되는 기준을 근무자들이 알 수 없기 때문에 근무 관리의 공정성이 떨어진다는 문제가 있었다.

이를 보완하고자 이전 근무 신청 이력을 반영해 원하는 근무가 배정되지 않았던 근무자를 우선 배정하도록 알고리즘의 기준을 설계하고 사용자들이 근무가 배정되는 기준을 알 수 있도록 메인 홈 페이지에 기준을 제시하고자 한다.

3) 요구사항 분석 **:**

- 기능관련 요구사항 : 수행해야하는 기능, 성능, HW/SW 사양
  - 관리자가 근무자가 필요한 시간대에 타임 슬롯을 생성하는 기능
  - 근무자가 생성된 타임슬롯에 우선순위를 설정하여 근무를 신청하는 기능
  - 알고리즘을 통해 근무자의 신청 우선순위를 가능한 한 많이 반영할 수 있는 근무표를 생성하는 기능
  - 최종 생성된 근무표를 관리자와 근무자가 확인 할 수 있도록 하는 기능
  - 권장 사양 : Window10 이상, CPU Intel i5 이상, 8GB 이상 메모리
- 데이터 관련 요구사항
  - 관리자의 근무표 생성 : 메뉴에서 날짜 정보를 클릭하고, 마우스 드래그 이벤트를 통하여 근무자가 필요한 시간대 데이터를 입력한다.
  - 근무자의 근무 신청 : 메뉴에서 우선순위를 선택하고 생성된 타임 슬롯을 클릭하면 해당 사용자와 함께 우선순위 정보 데이터가 저장된다.
  - 관리자와 근무자의 클릭 및 마우스 드래그 이벤트를 통하여 입력 된 시간 정보는 json 형식으로 저장된다.
  - 관리자와 근무자의 근무표 조회 : 알고리즘을 통해 최종 배정된 시간대가 json 형식으로 저장되면, 이를 근무표 조회 페이지에서 읽어서 캘린더에 출력한다.
- 데이터 보관 기간: 회원 정보의 경우, 탈퇴 등 회원의 요청이 있을 경우 삭제한다. 근무 일정에 관한 정보는 이후 급여 관리를 대비하여 해당 달이 지나고 2개월까지 유지한다.
- 인터페이스 관련 요구사항 : 다른 시스템등 외부로부터 입력되는 입력 데이터의 유형과 접근 방법 등
  - 근무 일정 생성 시 필요한 관리자 및 근무자 시간 데이터 JSON 형식으로 저장
  - 출력 데이터 접근: 사용자는 웹 인터페이스를 통해 근무 일정을 확인.
- 사용자 관련 요구사항

잠재 사용자는 스케줄이 고정되지 않거나 변동이 잦은 아르바이트 근무자 및 관리자이다. 여러 형태의 파트 타임 근무가 있으나 본 프로젝트에서는 특정 기업 혹은 매장에 고용되어 지속적인 근무를 약속했으나, 상대적으로 짧고 파편적인 근무시간 때문에 변동이 잦은 근무자들을 잠재 사용자로 설정했다.

근무자, 관리자 모두에게 근무 일정을 배정하는 것은 시간이 많이 드는 일이다. 관리자는 근무자들의 다른 일정과 요구 사항을 최대한 반영하여 운영에 차질이 없도록 일정을 배정해야 하고, 근무자들은 만약 근무 일정 중 다른 일정이 생겼을 때, 다른 근무자들에게 직접 연락하고 부탁하여 일정 변경을 요청하고, 이를 다시 관리자에게 알리는 복잡한 과정이 필요하다.

일정 배정의 비효율성이 아르바이트 근무자와 관리자에게 가장 필요한 기능이라고 생각했고, 이를 효율적이고 합리적으로 해결할 수 있는 자동 근무 시간 배정 알고리즘을 고안했다.

4) 설계의 현실적 제한요소 도출 **:**

- 시간적 제약 : 본 프로젝트는 약 **3**달 정도로 설계 시간이 한정되어 있다. 일반적으로 근태관리의 공통적인 기능은 크게 근무시간 관리, 인사 관리, 휴가 및 연차 관리, 급여 관리 **4**가지이지만, 본 프로젝트에서는 근무 선호 시간대를 반영한 최적 근무시간표 배정 시스템을 구축하는 데 집중하는 대신, 휴가 및 연차 관리, 급여 관리 기능은 배제되어 있다.
- 기술적 제약 : 근무 일정표 알고리즘 설계 시 모든 근무자의 선호도를 반영할 수는 없다는 제약이 존재한다. 예를 들어 이전 스케줄에서 우선순위가 잘 반영 되지 않은 경우를 고려하여 다음 스케줄을 짜는데, 이때 어느 시점까지 고려를 해야 할 지가 명확하지 않다. 모든 요청을 반영하는 것은 불가능하기에 우선순위를 정할 기준을 최대한 합리적으로 마련하기 위한 논의를 충분히 가지고, 피드백을 통해 기준의 완성도를 높일 것이다.

3.2 프로젝트 구현과정

1) 기능 정의 **:** 핵심 기능 및 하위 기능은 다음과 같다.

1. 근무 시간 자동 배정 기능-알고리즘

2. 근무표 생성 기능

관리자)

a. 근무 일정 생성 기능: 근무표를 생성해야할 기간의 시작일, 종료일, 시작 시간, 종료 시간, 시간 단위 등을 설정해 근무 일정을 생성할 수 있다.
b. 근무 시간대 설정 기능: 근무자들이 필요한 시간대를 드래그하여 타임 슬롯을 만들 수 있다.
c. 근무표 작성 마감 기한 설정 기능
d. 근무표 접근 가능 근무자 추가 및 삭제 기능: 관리자가 근무표를 생성할 때 근무자 아이디를 검색해 근무표에 접근할 수 있는 근무자를 추가하거나 삭제할 수 있다.
e. 근무표 초기화 및 저장 기능: 초기화 버튼을 눌러 설정해놓은 근무 타임 슬롯을 삭제할 수 있고 저장 버튼을 통해 근무표를 생성할 수 있다.

3. 근무 신청 기능

근무자)

a. 우선순위에 따른 희망 근무 신청: 근무자가 원하는 근무 시간을 우선 순위대로 타임 슬롯을 클릭하여 근무를 신청할 수 있다.

4. 근무 일정 관리 기능

-관리자)

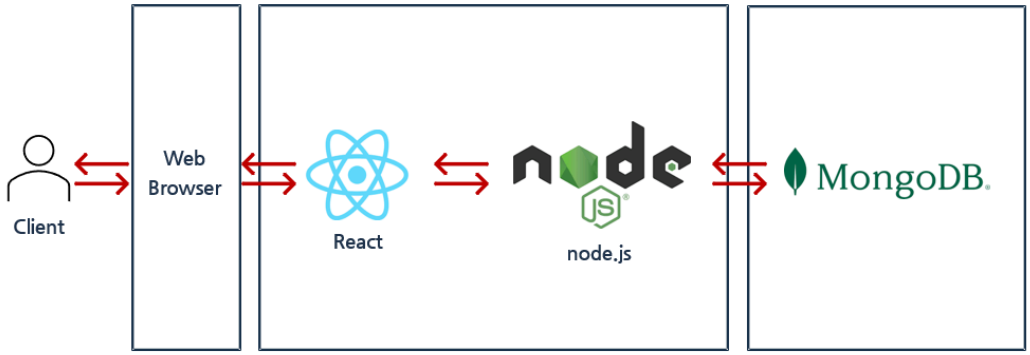
a. 최종 근무표 승인 및 수정 기능: 알고리즘이 배정한 근무일정을 관리자가 최종적으로 승인 및 수정 할 수 있다.
b. 전체 근무 일정 조회 기능

-근무자)

a. 근무 일정 승인 확인 및 전체 일정 조회 기능

2) 구체적인 설계안 **:**

- 전체 시스템 설계



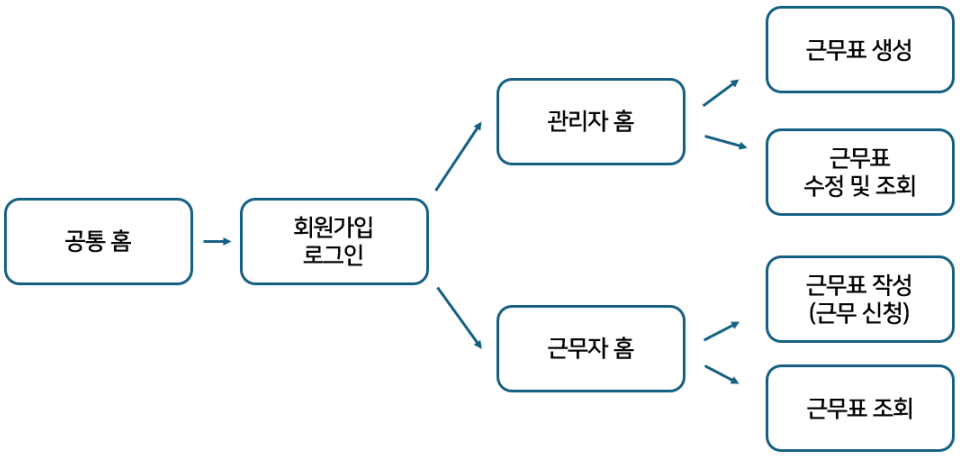
[그림1] 아키텍쳐

- 시스템 구조
  - Admin: 관리자 모델
  - Worker: 근무자 모델
  - 공통 필드: id, name, password, phone
- 기능별 상세 설계

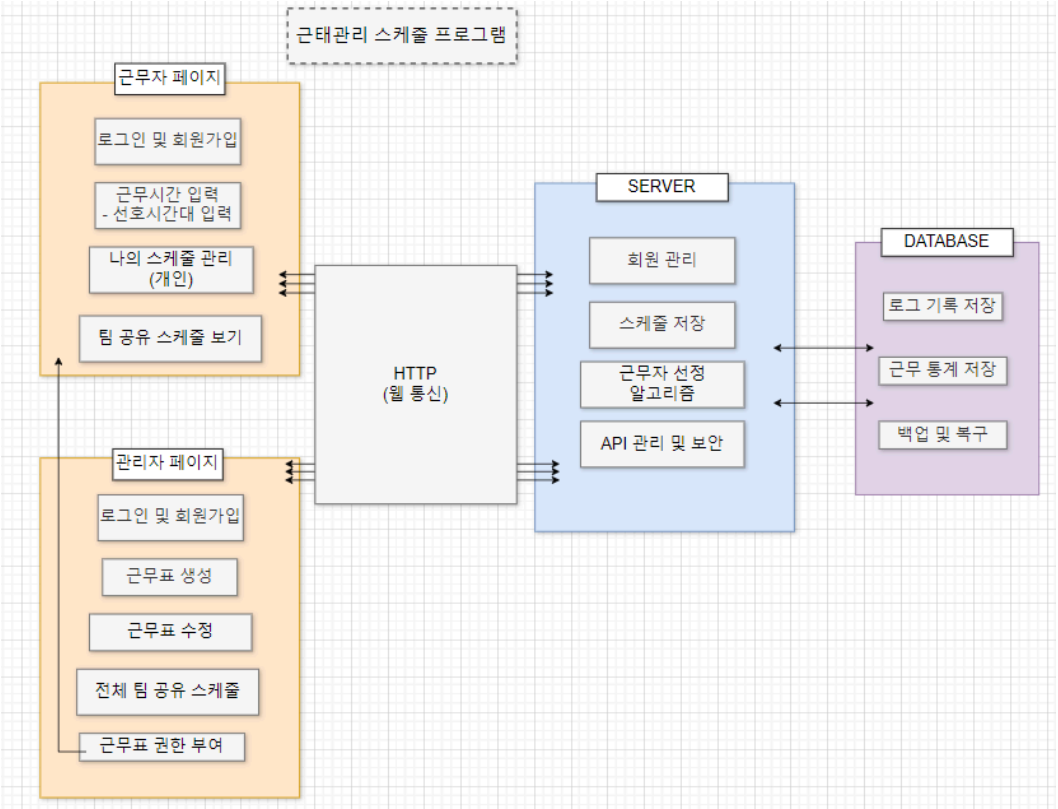
회원가입	로그인	로그아웃
-사용자 입력 데이터 검증 -중복 사용자 확인 (Admin, Worker) -새 사용자 생성 및 저장	-userType에 따른 모델 선택 -사용자 존재 여부 확인 -비밀번호 검증 -세션 생성	-세션 제거 -쿠키 삭제 -클라이언트 응답
*보안 사항: 비밀번호 암호화를 위한 해싱 처리 (모델 레벨에서 처리) 중복 가입 방지 세션ID를 쿠키로 관리	*세션 관리: *저장 정보: userId, userType *역할별 리다이렉션: 관리자: /admin/main 근무자: /worker/main	

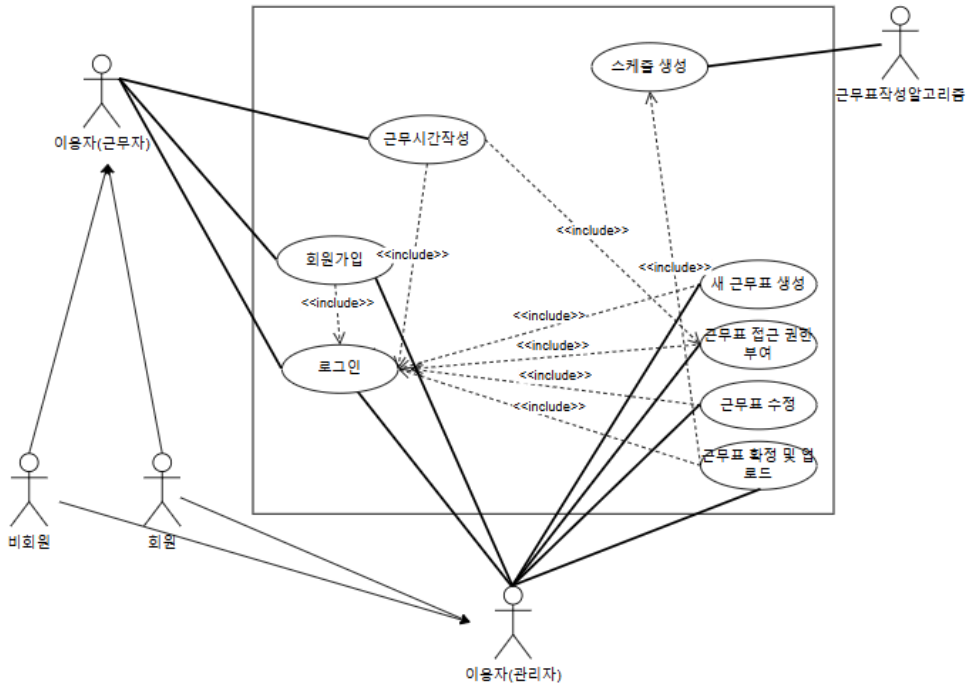
[표1] 회원 관리 기능 상세 설계

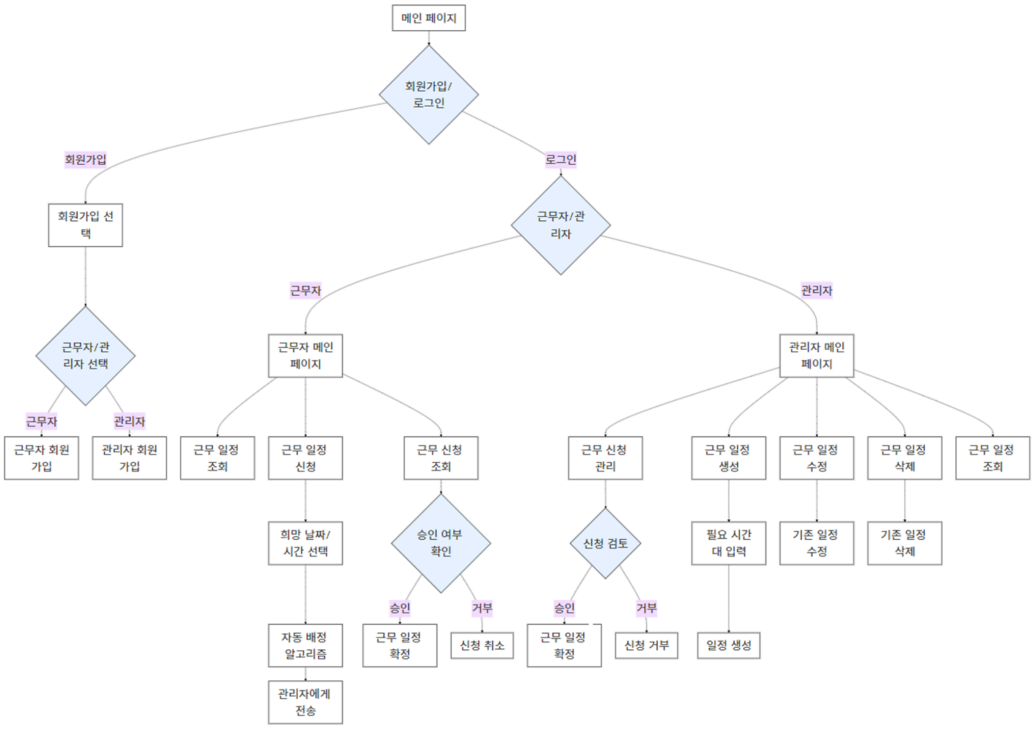
- 사이트 화면 구조


공통 홈
회원가입
로그인
관리자 홈
근무자 홈
근무표 생성
근무표
수정 및 조회
근무표 작성
(근무 신청)
근무표 조회

[그림2] UI 흐름


근태관리 스케줄 프로그램
근무자 페이지
로그인 및 회원가입
근무시간 입력
- 선호시간대 입력
나의 스케줄 관리
(개인)
팀 공유 스케줄 보기
관리자 페이지
로그인 및 회원가입
근무표 생성
근무표 수정
전체 팀 공유 스케줄
근무표 권한 부여
HTTP
(웹 통신)
SERVER
회원 관리
스케줄 저장
근무자 선정
알고리즘
API 관리 및 보안
DATABASE
로그 기록 저장
근무 통계 저장
백업 및 복구

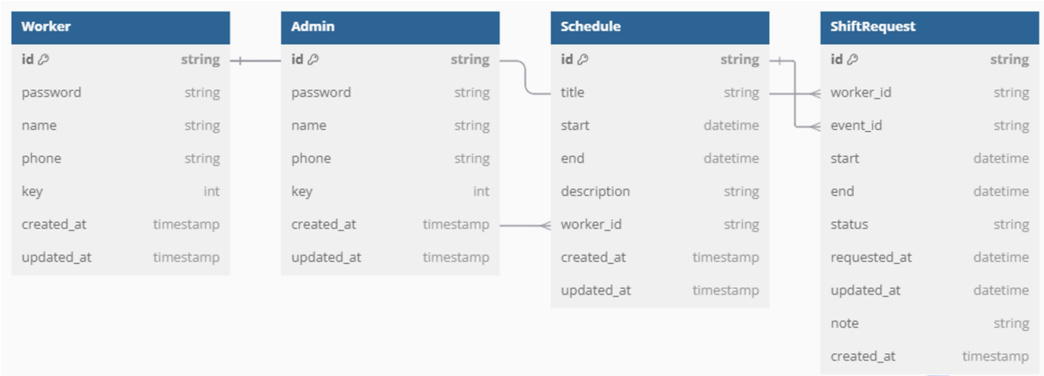

스케줄 생성
근무표작성알고리즘
이용자(근무자)
근무시간작성
<<include>>
회원가입
<<include>>
<<include>>
새 근무표 생성
<<include>>
<<include>>
근무표 접근 권한 부여
로그인
<<include>>
<<include>>
근무표 수정
근무표 확정 및 업로드
비회원
회원
이용자(관리자)


메인 페이지
회원가입/로그인
회원가입
로그인
회원가입 선택
근무자/관리자
근무자/관리자 선택
근무자
관리자
근무자 메인 페이지
관리자 메인 페이지
근무자
관리자
근무자 회원가입
관리자 회원가입
근무 일정 조회
근무 일정 신청
근무 신청 조회
근무 신청 관리
근무 일정 생성
근무 일정 수정
근무 일정 삭제
근무 일정 조회
희망 날짜/시간 선택
승인 여부 확인
신청 검토
필요 시간 대 입력
기존 일정 수정
기존 일정 삭제
승인
거부
승인
거부
자동 배정 알고리즘
근무 일정 확정
신청 취소
근무 일정 확정
신청 거부
일정 생성
관리자에게 전송

[그림]플로우차트

전체적인 흐름은 위 플로우차트와 같다, 근무자인지 관리자인지를 선택하여 회원가입 및 로그인을 하게 되면 각자의 메인페이지로 이동한다. 먼저 관리자는 근무자가 필요한 시간에 일정을 생성할 수 있다. 근무자는 생성한 일정 중 원하는 시간대를 선택하여 희망 근무를 신청한다. 알고리즘이 자동으로 배정하여 결과를 관리자에게 전송한다. 관리자는 일정을 최종 승인할 수 있고, 필요하다면 변경 가능하다. 근무자는 승인 여부 확인 후, 최종 일정 조회 페이지에서 확인할 수 있다.

- 데이터 구조 정의



[그림] E-R 다이어그램

API 명세서

Aa Index	기능	HTTP	API path
메인 페이지	첫 접속 시 메인 화면	GET	/home
회원 관리	근무자/관리자 회원가입	POST	/home/singup
	근무자/관리자 로그인	POST	/home/login
	세션 확인	GET	/home/session
근무자	로그인 후 근무자 메인 페이지	POST	/worker/main
	근무 일정 신청	POST GET	/worker/events/apply
	근무 일정 신청 승인 조회	GET	/worker/events/apply/
	근무 일정 조회	GET	/worker/events/all
	근무자 정보 조회	GET	/worker/getinfo
관리자	메인	POST	/admin/main
	근무 일정 생성	GET POST	/admin/events/create
	근무 일정 수정	GET PUT	/admin/events/edit
	근무 일정 삭제	POST	/admin/events/delete
	근무자 배정	POST	/admin/events/approve
	근무 일정 조회	GET	/admin/events/all
	관리자 정보 조회	GET	/admin/getinfo

[그림3] API 명세서

3) 구체적인 구현방법 **:**

- 최종 목표를 달성하기 위해 다양한 가능성과 대안을 고려하고 그 중 하나의 솔루션을 선택함
  - 서비스 제공 형식: 웹vs앱

    웹으로 구현 하는 방식을 선택하였다. 관리자가 주로 PC나 노트북으로 일정 조정 및 근무자 관리 기능을 수행할 가능성이 높으며, 많은 근무자의 데이터를 입력하고 수정해야 하는 경우에 웹이 적합할 것으로 판단하였기 때문이다. 물론 근무자의 경우 스케줄을 확인하는데 있어서 모바일이 편리할 수 있기 때문에 앱과 웹을 동시에 지원하는 것이 최고이긴 하겠으나, 시간상의 제약으로 웹 기반으로만 서비스를 구현하게 되었다.

- 서비스 사용자 타겟팅: 특정 직종 종사자 vs 범용적인 서비스
  범용적인 서비스로 개발하는 것을 선택하였다. 우선 개발에 참여하는 팀원들이 대학생인 관계로, 특정 직종에 대한 완벽한 이해를 하고 전문적인 직업에 대한 스케줄 관리 서비스를 구현하는 것이 어려울 것이라 판단하였기 때문이다. 또한, 범용적인 서비스로 구현을 할 시 타겟 사용자가 넓고, 추후 특정 직종에 필요한 기능을 추가하면서 서비스 범위를 확장하기에 유리하다는 장점이 있다.

■ 다양한 비교 분석 내용을 정리함

■ 비교 분석 시에 제한요소를 반드시 고려해야 함

● 선택된 설계문제의 해결방안(솔루션)과 최종 설계 결과물에 맞는 구현과 관련된 구체적인 계획을 서술

■ 주요 기능 구현 방법

먼저 일정들을 시각적으로 용이하게 표시하기 위해 달력 형식으로 일정 관리 기능을 구현. 이에 직접 달력을 구현,오픈 소스 활용, 2가지 방식이 있었음. 시간, 비용 측면에서 고려해본 결과 오픈소스의 fullcalendar라이브러리를 사용하기로 했다. fullcalendar 라이브러리는 달력 형식, 주간 형식, 일별 시간 형식이 모두 구현되어 시간별 일정을 관리하기에 편리했고, UI가 깔끔하고 보기 편하다는 장점이 있어 우리가 구현하려는 방식에 알맞다고 생각했다.

■ 서비스를 기능 구현을 위한 데이터 정의 및 자료 구조 정의

- 데이터 정의
  - 근무자 정보(Worker); 근무자의 회원 정보(id, name, password, phone)를 저장한다.
  - 관리자 정보(Admin): 관리자의 회원 정보(id, name, password, phone)을 저장한다.
  - 근무 일정(Event): 근무 일정에 대한 정보인 시작 시간, 종료 시간, 근무 일자, 근무 생성 시간, 업데이트 시간을 저장한다. 근무자(workerEvents)와 관리자(adminEvents)를 따로 나누어 관리한다.
  - 근무 신청(ShiftRequest): 희망하는 날짜 정보(희망일과 시작 시간,끝 시간)와 신청 근무자의 id, 근무 일정 id, 신청 상태 그d리고 알고리즘 계산에 필요한 마지막 근무일 및 시간, 근무신청이 거절된 횟수를 저장한다.

4) 최종 설계 결과물의 구현 수단

● 최종 설계 결과물의 구현 수단

■ 하드웨어 개발환경

- Microsoft Windows 11 Pro
- x64 기반 PC

■ 소프트웨어 개발환경

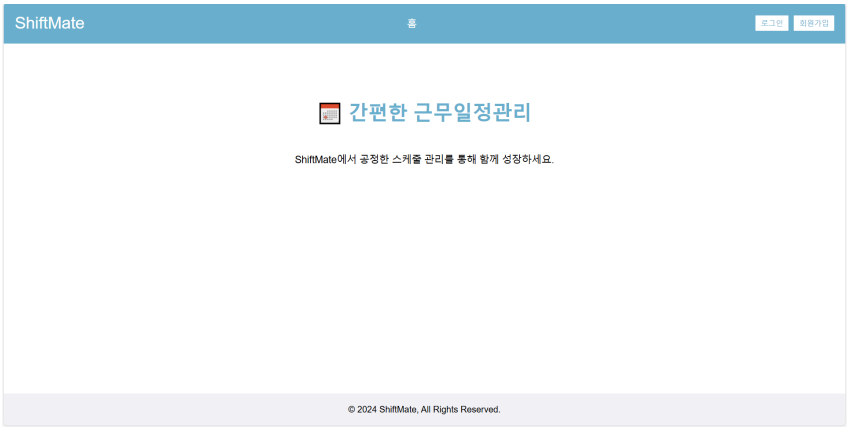
- 프론트엔드: React, javascript

	- 백엔드: node.js, javascript - IDE : VScode - 데이터베이스: mongoDB - 오픈소스: fullcalendar library

3.3 결과 분석

1) 프론트엔드

- 메인 홈 페이지

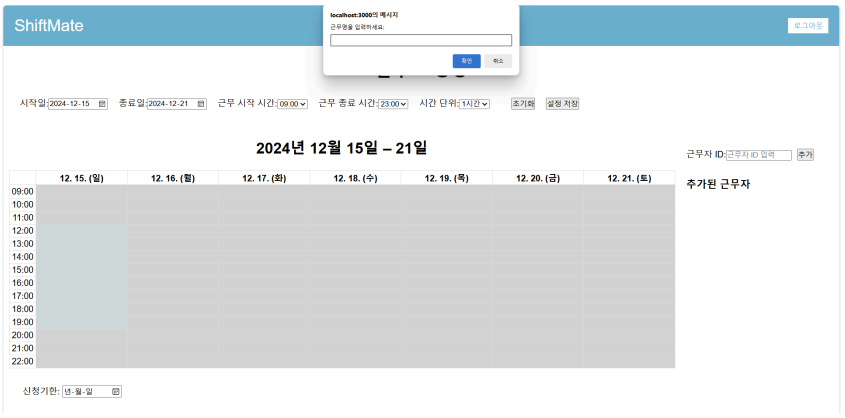


로그인 전에는 네비게이션 바 우측에 로그인과 회원가입 버튼 구현.
로그인을 하고 나면 근무자 용 메인 홈과 관리자 용 메인 홈으로 각각 나눠서 연결된다. 근무자의 경우 네비게이션 바에 홈, 근무표 신청, 근무표 조회 페이지가 구현되고 관리자는 홈, 근무표 생성, 근무표 조회 페이지가 구현되도록 하였다.

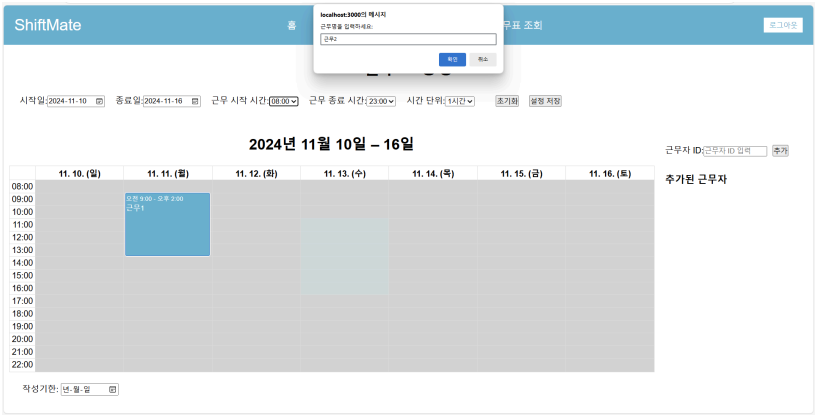
- 로그인 및 회원가입 페이지

회원가입 시 라디오 버튼을 통해 관리자, 근무자를 구분한다.

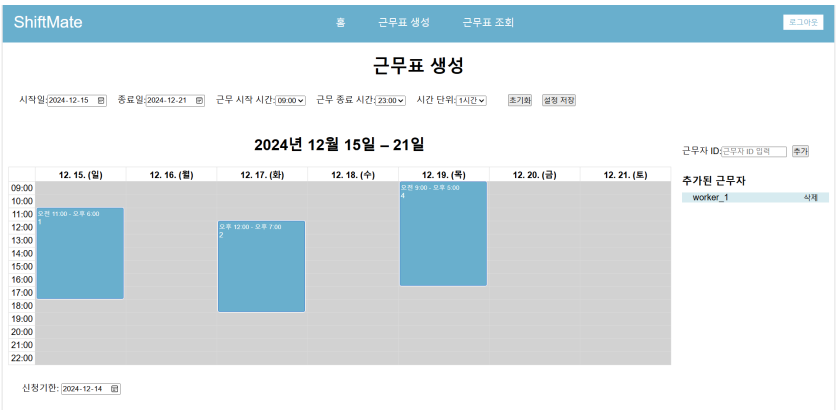
- 근무표 생성 페이지



- 페이지 상단에는 생성을 원하는 근무표의 시작일과 종료일을 선택할 수 있고 이를 선택하게 되면 캘린더 포맷이 나타나게 된다. 상단의 근무 시작 시간과 근무 종료 시간, 시간 단위를 원하는대로 설정하게 되면 이에 맞게 근무표의 열속성이 변화하게 된다. 이때 시간 단위는 1시간과 30분으로 설정 가능하다.



근무자를 필요한 시간대를 드래그하면 페이지 상단에 '근무명을 입력하세요'라는 문구와 함께 팝업이 뜨고 근무명을 입력하게 되면 해당 시간대에 타임 슬롯이 만들어지게 된다.



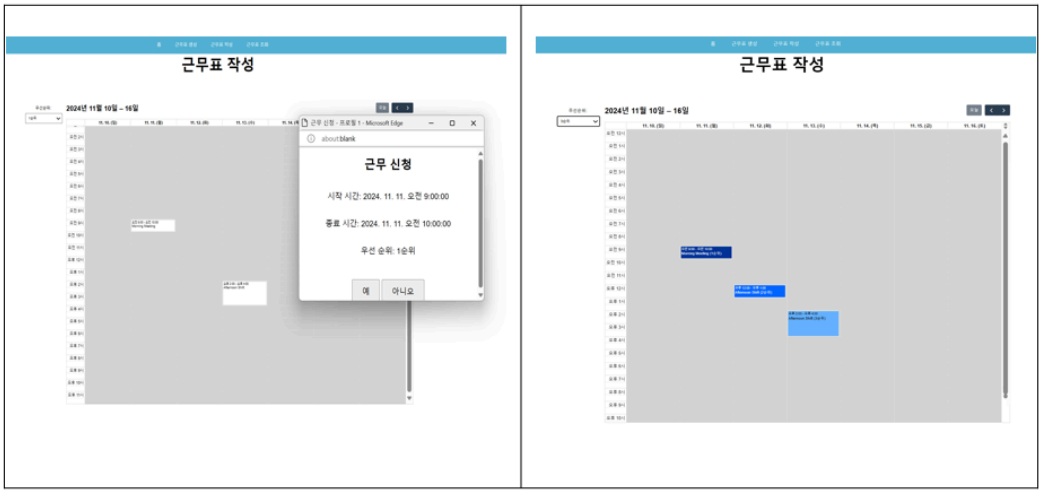
근무자들이 신청할 수 있는 기간을 근무표 아래에 있는 신청기한 기능으로 설정해줄 수 있다. 이때 신청기한은 근무표의 시작일보다 이른 날짜를 선택해야만 하도록 하였고, 시작일과 같거나 더 늦은 날짜를 선택할 시 alert를 띄우도록 하였다.

근무표 우측에는 근무표에 접근할 수 있는 근무자를 아이디로 검색해 추가하거나 삭제할 수 있는 기능을 구현해놓았다. 등록되어있지 않은 아이디거나 관리자 아이디를 입력할 경우 존재하지 않는 근무자 아이디라고 alert이 뜨도록 설정했다.

근무일정을 모두 생성하고 저장버튼을 누르면 데이터베이스에 근무표 데이터가 저장되고 근무표 생성 페이지는 리로드 되도록 만들었다.

만약 신청기한이나 근무자 아이디 등록 중 하나라도 설정하지 않은 채 저장버튼을 누르면 이를 설정하라고 alert를 띄우도록 하였다.

- 근무 신청 페이지

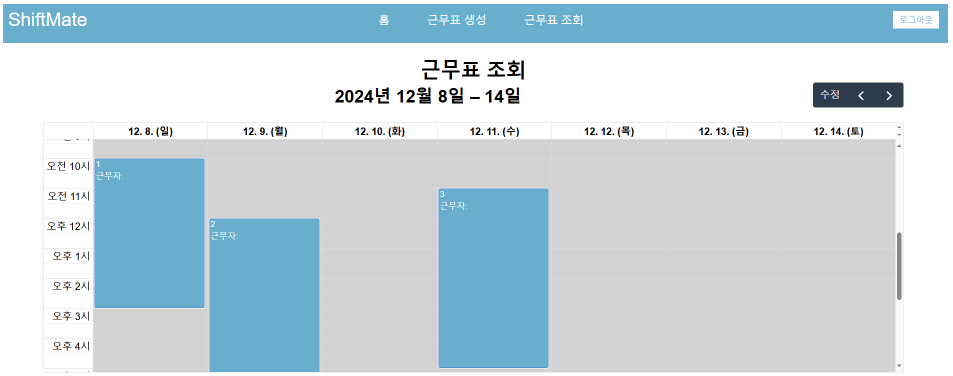


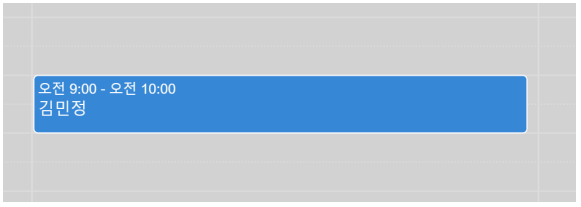
근무 신청페이지를 누르면 데이터베이스에서 worker/events/apply api를 통해 생성된 근무표 데이터를 불러온다.

왼쪽 메뉴 바에서 우선순위를 선택한 뒤, 흰 색으로 표시 된 타임 슬롯을 클릭하면 근무 신청 확인 팝업이 생성 된다.

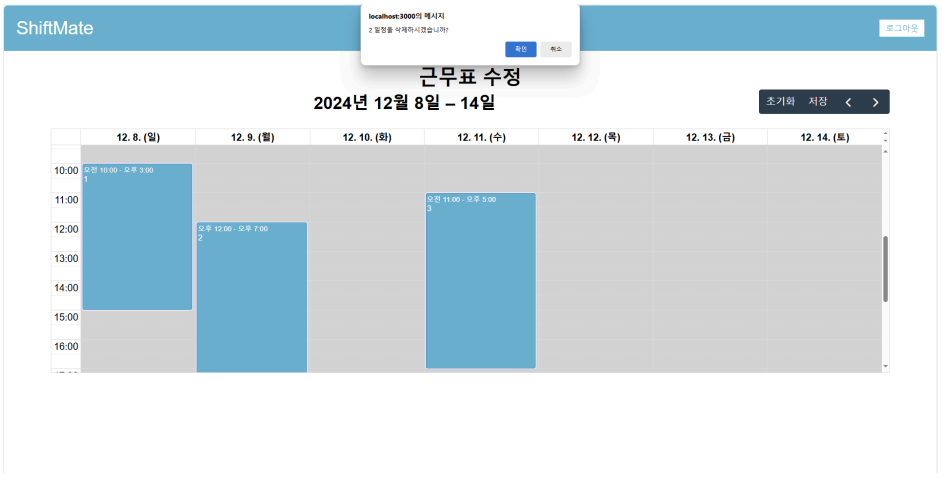
신청 완료 시 우선순위에 따라 서로 다른 색상으로 표시 된다. (1순위 - 진한 파랑, 2순위 - 파랑, 3순위 - 연한 파랑)
이때, 1,2,3순위를 최소 하나씩 신청해야하며 3순위는 2개이상 신청할 수 있도록 하였고, 이를 어기면 alert가 뜨도록 코드를 구현하였다.

- 근무표 조회 페이지





관리자가 승인되어 최종적으로 확정 된 근무표가 표시 된다.
이 때, 타임 슬롯에는 시간대와 확정 근무자 이름이 표시 된다.
관리자용 근무표 조회 페이지에서는 수정 버튼이 추가되어있고 수정 버튼을 누르면 수정 페이지로 넘어가게 된다.



근무표 수정 페이지에서는 기존 근무 일정을 누르면 삭제할 수 있고 원하는 시간대에 드래그를 해서 새로운 근무 일정을 생성할 수 있다.

초기화 버튼을 눌러 한 주의 근무표 데이터를 삭제할 수 있고 수정을 완료하고 난 후에는 저장 버튼을 눌러 저장한다.

이때 저장버튼을 누르면 근무자 조회페이지와 관리자 조회페이지에 실시간으로 업데이트 되도록 연동하였다.

2) 백엔드

-회원가입

```
// 통합 회원가입
router.post('/signup', async (req, res) => {
    try {
        const { id, name, password, phone, userType } = req.body;

        // userType에 따라 적절한 모델 선택
        const Model = userType === 'admin' ? Admin : Worker;

        // 기존 사용자 확인 (관리자와 근무자 모두 확인)
        const existingAdmin = await Admin.findOne({ $or: [{ id }, { phone }] });
        const existingWorker = await Worker.findOne({ $or: [{ id }, { phone }] });

        if (existingAdmin || existingWorker) {
            return res.status(400).json({
                message: "이미 존재하는 회원입니다",
                redirectUrl: '/home/login'
            });
        }

        // 새 사용자 생성
        const user = new Model({ id, name, password, phone });
        await user.save();

        res.status(201).json({
            message: "회원가입 성공! 로그인 해주세요.",
            redirectUrl: '/home/login',
            success: true,
        });
    } catch (error) {
        console.error('Signup error:', error);
        res.status(500).json({
            message: "서버 오류가 발생했습니다.",
            error: true,
        });
    }
```

데이터베이스의 id와 phone을 비교해 중복 회원가입 방지를 구현하였다. 회원가입 성공 시 바로 로그인 페이지로 리디렉션하였다.

-로그인

```
router.post('/login', async (req, res) => {
    try {
        const { id, password, userType } = req.body;

        const Model = userType === 'admin' ? Admin : Worker;
        const user = await Model.findOne({ id });

        if (!user) {
            return res.status(404).json({ message: "존재하지 않는 회원입니다." });
        }

        const isMatch = await user.comparePassword(password);
        if (!isMatch) {
            return res.status(400).json({
                message: "비밀번호가 일치하지 않습니다."
            });
        }

        // 세션에 사용자 정보 저장
        if (req.session) {
            req.session.userId = user._id;
            req.session.userType = userType;
        }

        const redirectUrl = userType === 'admin' ? '/admin/main' : '/worker/main';

        res.status(200).json({
            message: "로그인 성공!",
            redirectUrl
        });
    } catch (error) {
        console.error('Login error:', error);
        res.status(500).json({ message: "서버 오류가 발생했습니다." });
    }
});
```

세션에 사용자 정보를 저장하도록 구현하였고 usertype을 이용하여 로그인한 사람이 관리자인 경우 관리자의 메인 페이지로, 근무자인 경우 근무자의 메인 페이지로 이동하도록 리디렉션될 url을 분리하였다.

-근무자 자동 배정 알고리즘

```
// 우선순위 가중치
const priorityWeights = { 1: 4, 2: 2, 3: 0 }; // 1순위: +4, 2순위: +2, 3순위: +0
const sortedRequests = requests
    .map(request => {
        const hoursSinceLastShift = (new Date() - new Date(request.lastShiftEnd)) / (1000 * 60 * 60);
        // 우선순위에 따른 가중치 추가
        score += priorityWeights[request.priority];

        // 점수 계산: 마지막 근무 경과 시간 + 거절 횟수에 따른 가산점
        let score = hoursSinceLastShift + request.rejections * 5 + priorityWeights[request.priority];;

        return { ...request.toObject(), score };
    })
    .sort((a, b) => b.score - a.score); // 높은 점수 순 정렬
```

근무자를 자동으로 배정하는 알고리즘은 다음과 같다.

1. 해당 스케줄에 등록된 근무자들에게 희망하는 근무 시간대 신청을 받는다.
2. 희망 근무 신청 마감 기한이 지나면 다음 기준을 바탕으로 근무자 별 점수를 계산한다.

기준	점수 부과	이유
우선 순위(희망 순위) (priorityWeights)	1순위: +4점 2순위: +2점 3순위: +0점	근무자의 선호도를 반영하되, 다른 요소들과 균형을 맞추기 위해 적절한 가중치 부여
거절 횟수 (rejections)	거절 1회당 +5점	거절된 근무자에게 보상 기회 제공 및 1순위 가중치(4점)보다 높아 공정성 확보
마지막 근무 후 경과 일수 (hoursSinceLastShift)	경과 일수 1일당 +1점	근무 간격을 균등하게 분배 및 오랫동안 근무하지 않은 근무자에게 우선권 부여

3. 점수가 높은 지원자 순대로 근무자를 배정한다.
4. 배정이 완료된 근무 일정은 근무자/관리자 계정의 조회 페이지에서 조회 가능하다

근무 신청에 따른 시간 배정의 예시는 다음과 같다.

1. 근무 신청 데이터 및 근무 배정 결과

	근무 신청 데이터	근무 배정 결과
1주차		
2주차		
3주차		

1주차 — 근무 신청 데이터

근무 슬롯	신청자	우선순위	점수 (설명)
11/10 10:00~14:00	worker1	1	4 (우선순위 1: +4)
11/11 14:00~18:00	worker1	3	0 (우선순위 3: +0)
11/11 14:00~18:00	worker3	2	2 (우선순위 2: +2)
11/12 12:00~16:00	worker1	2	2 (우선순위 2: +2)
11/12 12:00~16:00	worker3	1	4 (우선순위 1: +4)
11/13 10:00~14:00	worker2	1	4 (우선순위 1: +4)
11/14 14:00~18:00	worker2	2	2 (우선순위 2: +2)
11/15 12:00~16:00	worker2	3	0 (우선순위 3: +0)

1주차 — 근무 배정 결과

근무 슬롯	배정된 근무자	점수 (설명)
11/10 10:00~14:00	worker1	4 (우선순위 1: +4)
11/11 14:00~18:00	worker3	2 (우선순위 2: +2)
11/12 12:00~16:00	worker3	4 (우선순위 1: +4)
11/13 10:00~14:00	worker2	4 (우선순위 1: +4)
11/14 14:00~18:00	worker2	2 (우선순위 2: +2)
11/15 12:00~16:00	worker2	0 (우선순위 3: +0)

2주차 — 근무 신청 데이터

근무 슬롯	신청자	우선순위	거절횟수	근무후일수	총점수
11/17 10:00~14:00	worker1	1 (+4)	1 (+5)	7	16
11/17 10:00~14:00	worker3	2 (+2)	0	5	7
11/18 14:00~18:00	worker2	1 (+4)	0	4	8
11/19 12:00~16:00	worker1	2 (+2)	0	2	4
11/20 10:00~14:00	worker3	1 (+4)	0	8	12
11/21 14:00~18:00	worker2	2 (+2)	0	3	5
11/22 12:00~16:00	worker2	3 (+0)	1 (+5)	4	9

2주차 — 근무 배정 결과

근무 슬롯	배정된 근무자	우선순위	거절횟수	근무후일수	총점수
11/17 10:00~14:00	worker1	1 (+4)	1 (+5)	7	16
11/18 14:00~18:00	worker2	1 (+4)	0	4	8
11/19 12:00~16:00	worker1	2 (+2)	0	2	4
11/20 10:00~14:00	worker3	1 (+4)	0	8	12
11/21 14:00~18:00	worker2	2 (+2)	0	3	5
11/22 12:00~16:00	worker2	3 (+0)	1 (+5)	4	9

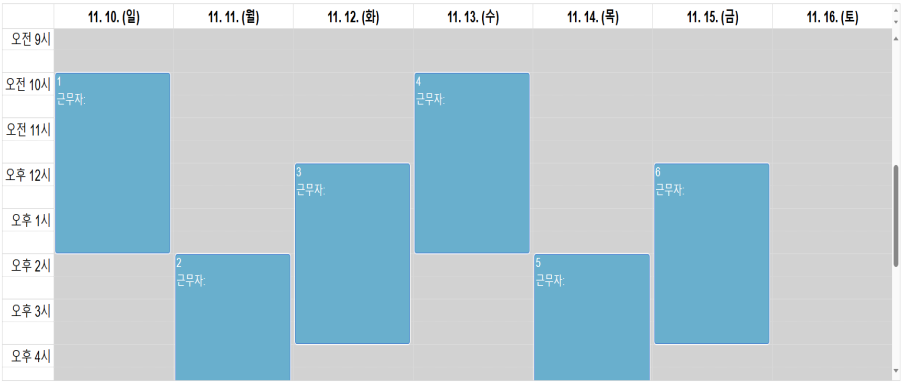
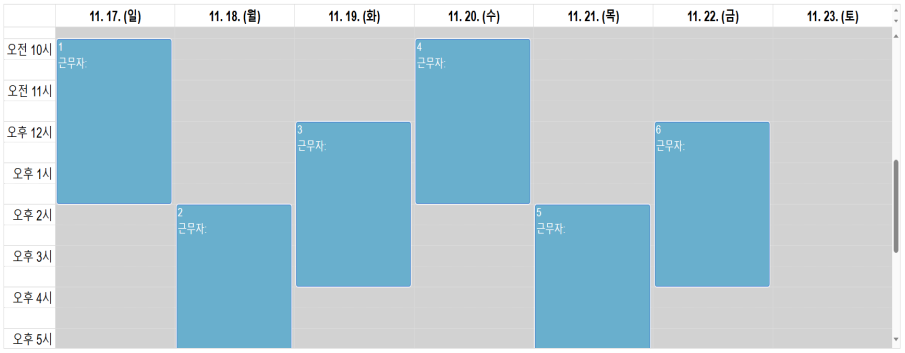
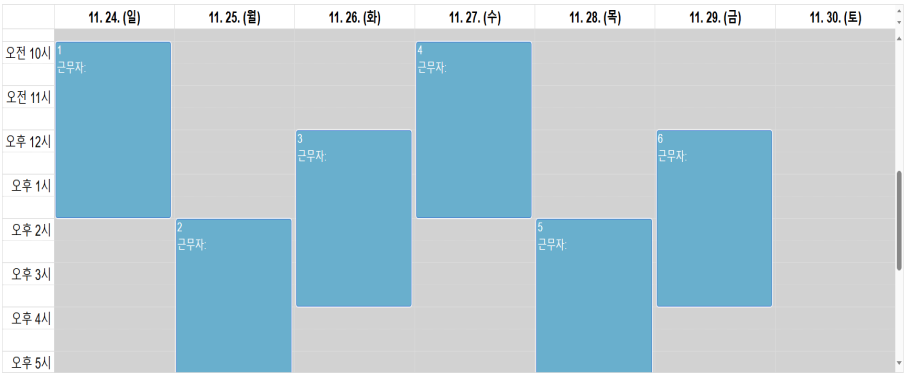
3주차 — 근무 신청 데이터

근무 슬롯	신청자	우선순위	거절횟수	근무후일수	총점수
11/24 10:00~14:00	worker2	1 (+4)	1 (+5)	2	11
11/24 10:00~14:00	worker3	2 (+2)	1 (+5)	4	11
11/25 14:00~18:00	worker1	1 (+4)	1 (+5)	6	15
11/26 12:00~16:00	worker3	1 (+4)	0	6	10
11/27 10:00~14:00	worker2	2 (+2)	2 (+10)	3	15
11/28 14:00~18:00	worker1	2 (+2)	0	3	5
11/29 12:00~16:00	worker1	3 (+0)	1 (+5)	4	9

3주차 — 근무 배정 결과

근무 슬롯	배정된 근무자	우선순위	거절횟수	근무후일수	총점수
11/24 10:00~14:00	worker3	2 (+2)	1 (+5)	4	11
11/25 14:00~18:00	worker1	1 (+4)	1 (+5)	6	15
11/26 12:00~16:00	worker3	1 (+4)	0	6	10
11/27 10:00~14:00	worker2	2 (+2)	2 (+10)	3	15
11/28 14:00~18:00	worker1	2 (+2)	0	3	5
11/29 12:00~16:00	worker1	3 (+0)	1 (+5)	4	9

2. 알고리즘 배정 후 최종 시간표

1주차	
2주차	
3주차	

3.5 최종 발표 관련 질문 답변

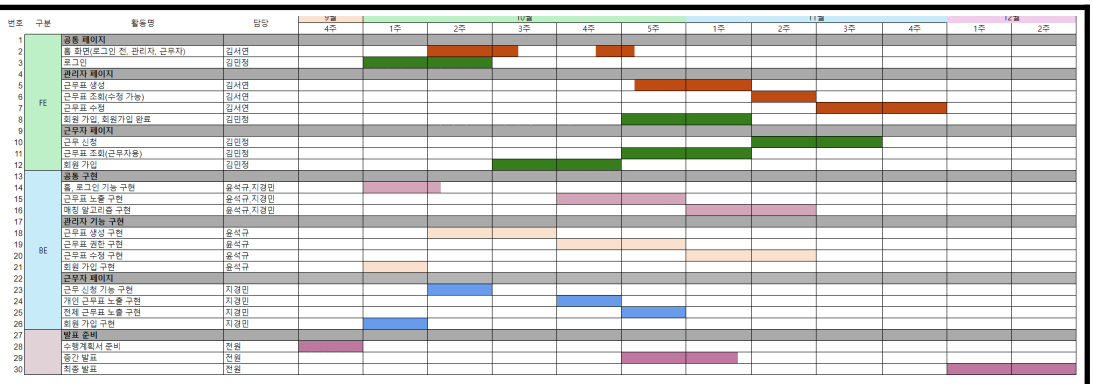
Q. 모바일 버전이 있었으면 좋을 것 같다.

A. 본 프로젝트의 팀원들 또한 공감하는 부분이다. 아르바이트 일정은 원할 때 쉽게 확인해보는 것이 중요하기 때문에 근무자 입장에서는 모바일이 더 편의성이 더 좋을 수 있다. 그러나 관리자 입장에서는 많은 근무자의 데이터를 입력하고 수정해야 하는 경우에 웹이 더 편의성이 좋을 것이라고 판단하였고, 모바일 프로그래밍에 능숙한 팀원이 없어 부득이하게 웹 서비스로 제작하게 되었다.

Q. 알고리즘은 팀원 전체가 고민하신 건가요?

	A. 알고리즘을 작성하는데 어떤 항목이 들어가면 좋을지는 팀원 전체가 고민하였다. 그러나 알고리즘 작동 방식을 구체화하고, 실제 코드를 작성한 것은 백엔드 팀원들이다. Q. 웹 페이지가 반응형인지 글씨가 너무 작은 것 같은데 추후 개선할 계획이 있는지 궁금하다. A. 아마 근무표 생성-신청 영상을 보고 질문을 주신 것 같다. 사실 컴퓨터 해상도 차이로 인해 팀원 컴퓨터 마다 UI가 다르게 나오는 상황이었는데, 프론트-백엔드 연동 오류로 인하여 급하게 글씨가 작게 나오는 팀원의 컴퓨터에서 영상을 녹화하게 되었다. 하드웨어에 맞게 유연하게 UI를 조정할 수 있어야 한다는 점에는 공감한다. Q. 시간표를 스크롤해서 보여주는게 아니라 한 화면에 다 보이도록 하실 의향은 없나요? A. 시간표가 길어질 경우 화면이 지나치게 압축되는 것을 방지하기 위하여, 다양한 화면 크기에서 대응하기 위하여 스크롤 방식을 선택하였습니다. Q. 연동이 안 된 부분이 있는데 원인이 무엇인가요? A. 데이터베이스에 근무표가 저장되는 과정에서의 문제, 혹은 프론트에서 데이터베이스의 내용을 불러오는데 문제가 있었던 것으로 추측한다. 백엔드와 프론트엔드의 연동이 팀원들이 예상했던 시간보다 오래 걸렸고 결국 최종 발표일까지 성공적으로 연동을 하지 못했다.
4. 기대효과	본 프로젝트로 기대할 수 있는 효과는 다음과 같다. 1.관리자와 근무자 간의 효율적인 일정 관리 ● 시간 절약: 일정 조율 과정에서 발생하는 불필요한 커뮤니케이션을 줄여 관리자와 근무자 모두의 시간을 절약한다. ● 오류 감소: 아르바이트와 같이 일정 변동이 잦은 경우, 자동화된 시스템을 통해 수동적인 일정 관리에서 발생할 수 있는 실수를 최소화한다. ● 근무자 만족도 향상 : 근무자의 희망 근무 시간을 반영한 일정 배정으로 근무자의 근무 만족도와 생산성 향상에 도움을 줄 수 있다. ● 근무 관리의 공평성 향상 : 알고리즘을 통해 자동으로 근무를 배정하고 이전 근무 기록에 따라 원하는 시간대에 근무하지 못한 근무자를 우선 배정해 근무 배정의 공평성을 극대화한다. 2.확장성 및 미래 발전 가능성 ● 확장성 : 향후 급여 관리, 성과 평가 등 다른 인사 시스템과의 연동 혹은 추가적인 기능 개발을 통해 종합적인 인사관리 플랫폼으로 발전할 수 있다. ● 산업 확장: 초기 아르바이트 중심에서 다양한 근무 형태와 산업군으로 확장 적용이 가능하다. 또한 선호도에 따른 시간 배정 알고리즘은 회사에서 미팅 시간을 정하거나 동아리에서 스터디 시간을 정할 때 사용하는 등 다방면으로 활용이 가능하다.

5. 팀내 협력



1) frontend - 김민정, 김서연

김서연 : 홈 화면, 근무표 생성, 관리자용 근무표 조회 페이지

김민정 : 로그인 및 회원가입, 근무 신청, 근무자용 근무표 조회 페이지

2) backend - 윤석규, 지경민

윤석규 : 근무 배정 알고리즘 및 근무 일정 CRUD 기능

지경민 : 로그인 및 회원가입 기능, 프런트 백엔드 연동

6. 참고문헌

-임세원, “인건비 낮춰라"...인력 효율화 나선 스타벅스”, 서울경제, 2024년 10월 13일 작성, 방문 주소: https://www.sedaily.com/NewsView/2DFJYQW7RR[24.11.18]

-이희조, “주휴수당 부담에 단기알바만 채용...편의점 한 곳에 6명 근무도”, 매일경제, 2024년 6월 12일 작성, 방문 주소: https://www.mk.co.kr/news/economy/11039899[24.11.18]

-시프티(Shiftee), "Shiftee", 2024년 11월 18일 작성, [온라인], 방문 주소: https://shiftee.io/ko

-플렉스(Flex), "Flex", 2024년 11월 18일 작성, [온라인], 방문 주소: https://about.flex.team/features/time-tracking

-핀플(pinpl),"pinpl", 2024년 11월 18일 작성, [온라인], 방문 주소: https://pinpl.biz/

-위펄슨(weperson), "weperson", 2024년 11월 18일 작성, [온라인], 방문 주소: https://www.weperson.com/ko

-듀팅(dutying), "dutying", 2024년 11월 18일 작성, [온라인], 방문 주소: https://www.dutying.net/

-fullcalendar, “fullcalendar”, 2024년 11월 18일 작성, [온라인], 방문 주소: https://fullcalendar.io/

	항목	세부내용	예상(달성)시기
성과 창출	Github	아르바이트 시간 배정 웹서비스 개발	2024.12
	논문게재 및 참가		

	SW등록		
	특허출원		
	시제품 (App스토어 등록)		