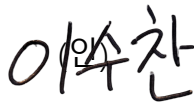


창의융합종합설계 최종 결과보고서

팀 명칭	COCA			
책 임 자	성 명	이수찬		
	소 속	컴퓨터소프트웨어공학과		
	학 년	4	학 번	20181402
작품명칭	COCA(Collaborative Calendar) : 그룹 및 개인 일정 관리 및 공유 서비스			
개발기간	2024년 2월 1일 ~ 2024년 6월 19일			
참여학생	학번	이름	전공	
	20190823	이상헌	컴퓨터소프트웨어공학과	
	20200979	임희열	컴퓨터소프트웨어공학과	
	20210923	이채연	컴퓨터소프트웨어공학과	
<p>본인은 소프트웨어공학심화프로그램 종합설계에 대한 최종결과보고서 및 작품을 첨부와 같이 제출합니다.</p> <p>별첨: 아래 내용을 포함한 CD 혹은 USB 메모리</p> <ul style="list-style-type: none">- 제안서, 결과 보고서- 수업 시간 발표 자료- 모든 회의록- 개발 작품에 대한 관련 자료(동영상, 소스 파일등) <p>2024년 06월 19일</p> <p>책임자 이수찬 </p> <p>소프트웨어공학심화프로그램 PD 귀하</p>				

< 목 차 >

1. 개요	1
1-1. 문제 정의	1
1-2. 목표	2
2. 요구사항 및 기능명세	3
3. 시스템 구조도	5
4. 시스템 설계도	6
4-1. 클래스 다이어그램 설계	6
5. 시스템 구성 및 환경	13
5-1. 배치 다이어그램	13
5-2. 개발 환경	13
6. 구현 기술	14
6-1. 빈 일정 찾기 알고리즘	14
7. 주요 자료 구조 또는 DB 구조	15
7-1. 데이터베이스 구조	15
7-2. RESTful API JSON 데이터 통신 구조	16
8. 주요 함수 명세	17
9. 사용자 매뉴얼	19
10. 역할 분담	23
11. 팀원별 프로젝트 감상문	24
12. 팀원별 프로젝트 포트폴리오	29

1. 개요

1-1. 문제 정의

본 프로젝트는 업무나 학업, 일상 생활 등에서 일정과 관련하여 발생할 수 있는 문제들을 아래와 같이 정의함으로써 해당 문제들을 소프트웨어를 통해 해결하고자 기획하게 되었다.

1. 일정 전달에서 발생할 수 있는 오류 가능성과 불편함

- 구두 전달 : 발음 혹은 유사 단어로 인한 전달 오류, 기억에 의존되는 오류
- 공지 전달 : 확인자가 직접 공지를 찾아가거나 공지를 주기적으로 확인해야 하는 불편

본 프로젝트는 그룹 관리자가 일정 생성 시 그룹원들의 일정에 자동으로 추가됨으로써 위와 같은 오류 가능성과 불편함을 해결하고자 함.

2. 상대방의 연락 가능 여부 확인의 필요성

회의나 발표, 강의 등 중요한 일정을 수행하는 중 전화나 알림 메시지가 오는 등으로 인해 업무에 방해되는 경우들에 대해 방지해야 할 필요성을 느낌.

중요한 일정과 같은 연락을 받기 어려운 일정에 대해 사용자 상태 표시를 통해 연락자가 사전에 확인할 수 있도록 함으로써 문제를 해결하고자 함.

3. 그룹에서 생성된 일정 변경에 대비를 위한 반복 확인의 불편함

그룹 관리자가 기존에 있던 일정을 변경할 가능성이 존재한다. 이에 따라 그룹 참가자들은 일정 변경 가능성을 대비하기 위해 지속적이고 반복적으로 일정 변동 공지를 확인해야한다.

따라서 본 프로젝트는 그룹 캘린더 서비스를 통해 변경된 일정에 대해 쉽게 확인할 수 있도록 개발하고자 한다.

4. 그룹의 관리자의 새로운 참가자 초대 의 번거로움

기존 시스템들은 그룹의 관리자 혹은 이미 참가한 사용자가 초대 기능을 통해야만 새로운 사용자를 그룹에 포함시킬 수 있다. 이러한 방식은 폐쇄적인 방식이며, 해당 프로젝트는 개방적인 방식으로 참가자가 그룹을 찾아 참가하도록 개선하여 기존의 번거로움을 해소하고자 한다.

1-2. 목표

COCA 시스템은 사용자 중심의 일정 관리 및 그룹 협업 플랫폼을 제공함으로써 일상생활, 학업, 업무 등 다양한 분야에서의 효율적인 일정 관리와 소통을 목표로 합니다. 본 시스템은 사용자가 직면할 수 있는 일정 관리의 불편함을 해소하고, 개인 및 그룹 단위의 일정 공유와 관리를 원활하게 하여 개인의 생산성 향상과 그룹 간 협업을 촉진합니다.

소프트웨어의 주요 기능을 통해 COCA 시스템은 사용자가 개인 및 그룹 일정을 효율적으로 관리하고, 다른 사용자들과의 협업과 소통을 원활하게 하는 플랫폼을 제공함으로써, 사용자의 생산성과 협업 효율을 극대화하는 것을 목적으로 합니다.

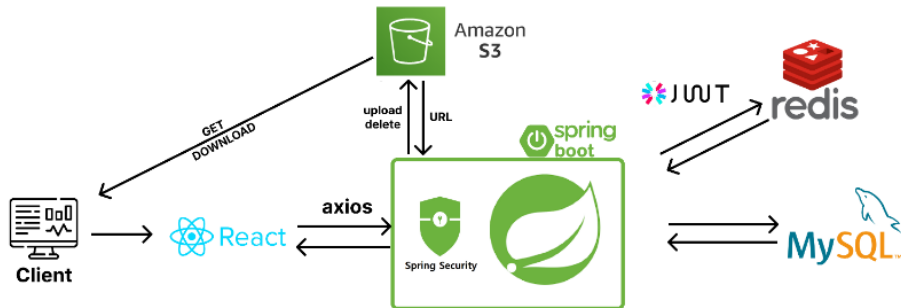
2. 요구사항 및 기능명세

항번	서브시스템명	유즈케이스 다이어그램 식별자	유즈케이스명	유즈케이스 식별자	유즈케이스 명세 식별자
1	회원관리	UCD-01	회원 가입	UC-01	UCS-01
2			로그인	UC-02	UCS-02
3			로그아웃	UC-03	UCS-03
4			개인 정보 조회	UC-04	UCS-04
5			개인 정보 수정	UC-05	UCS-05
6			회원 탈퇴	UC-06	UCS-06
7	캘린더 관리	UCD-02	캘린더 목록 조회	UC-07	UCS-07
8	공통 일정 관리	UCD-03	빈 일정 찾기	UC-08	UCS-08
9	개인 일정 관리	UCD-04	개인 일정 등록	UC-09	UCS-09
10			개인 일정 목록 조회	UC-10	UCS-10
11			개인 일정 상세 정보 조회	UC-11	UCS-11
12			개인 일정 수정	UC-12	UCS-12
13			개인 일정 삭제	UC-13	UCS-13
14	그룹 일정 관리	UCD-05	그룹 일정 등록	UC-14	UCS-14
15			개인 일정 가져오기	UC-15	UCS-15
16			그룹 일정 목록 조회	UC-16	UCS-16
17			개인 일정으로 저장하기	UC-17	UCS-17
18			그룹 일정 상세 정보 조회	UC-18	UCS-18
19			그룹 일정 수정	UC-19	UCS-19

20			그룹 일정 삭제	UC-20	UCS-20
21	그룹 관리	UCD-06	그룹 등록	UC-21	UCS-21
22			그룹 검색	UC-22	UCS-22
23			그룹 상세 정보 조회	UC-23	UCS-23
24			그룹 초대	UC-24	UCS-24
25			그룹 수정	UC-25	UCS-25
26			그룹 삭제	UC-26	UCS-26
27	그룹 공지 관리	UCD-07	그룹 공지 등록	UC-27	UCS-27
28			그룹 공지 조회	UC-28	UCS-28
29			그룹 공지 수정	UC-29	UCS-29
30			그룹 공지 삭제	UC-30	UCS-30
31	친구 관리	UCD-08	친구 등록	UC-31	UCS-31
32			친구 일정 조회	UC-32	UCS-32
33			친구 목록 조회	UC-33	UCS-33
34			친구 수정	UC-34	UCS-34
35			친구 삭제	UC-35	UCS-35
36	요청 관리	UCD-09	요청 등록	UC-36	UCS-36
37			요청 목록 조회	UC-37	UCS-37
38			요청 수정	UC-38	UCS-38
39			요청 삭제	UC-39	UCS-39
40	회원 인증	UCD-10	회원 토큰 생성	UC-40	UCS-40
41			회원 토큰 인증	UC-41	UCS-41
42			회원 토큰 삭제	UC-42	UCS-42

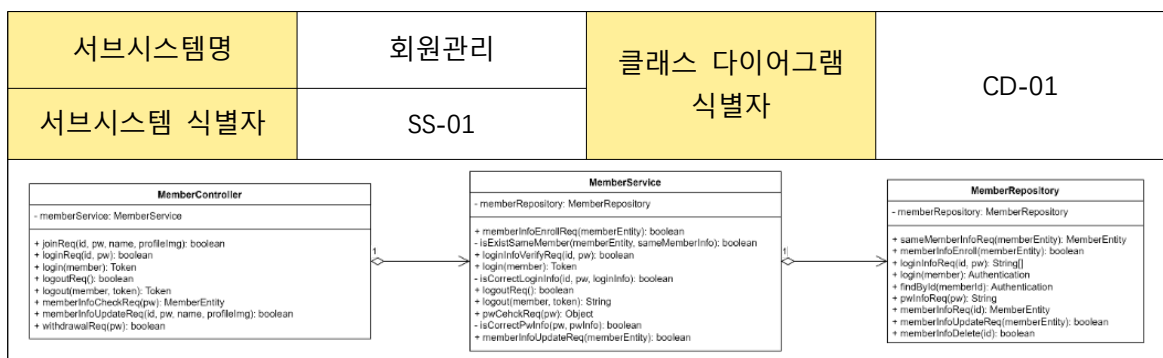
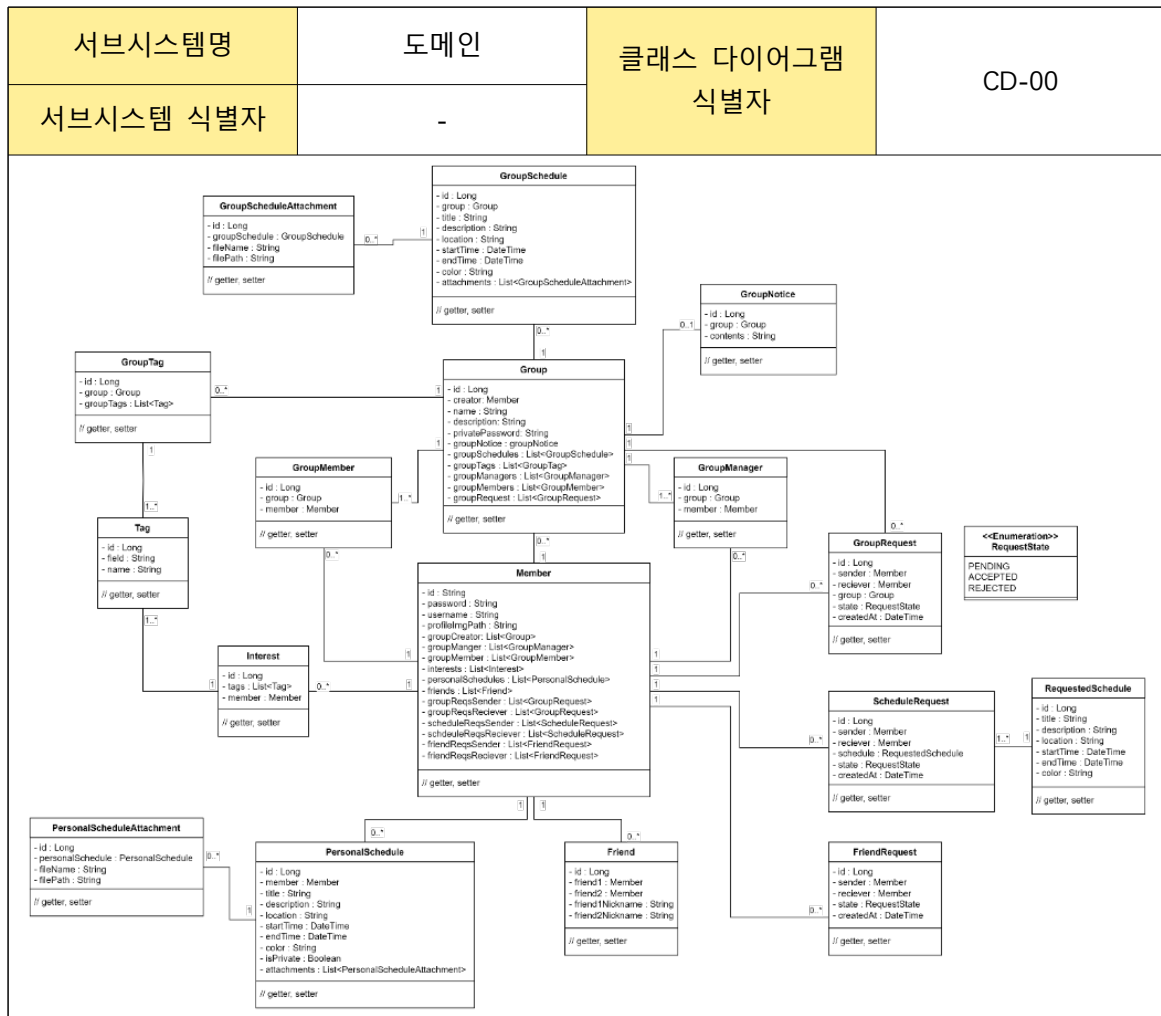
3. 시스템 구조도

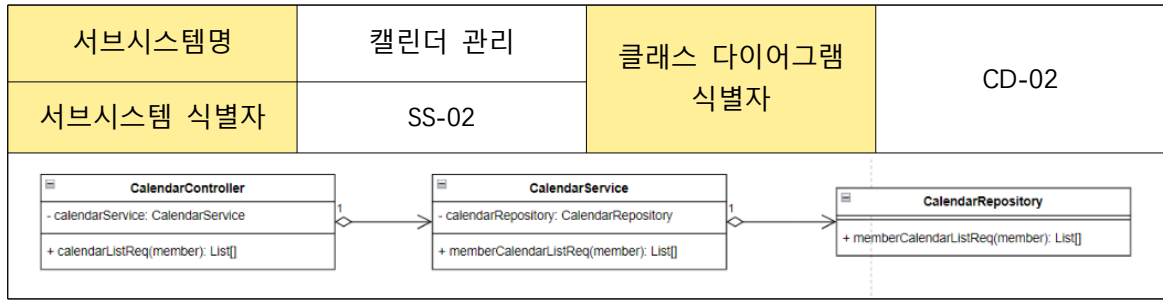
시스템 구조도



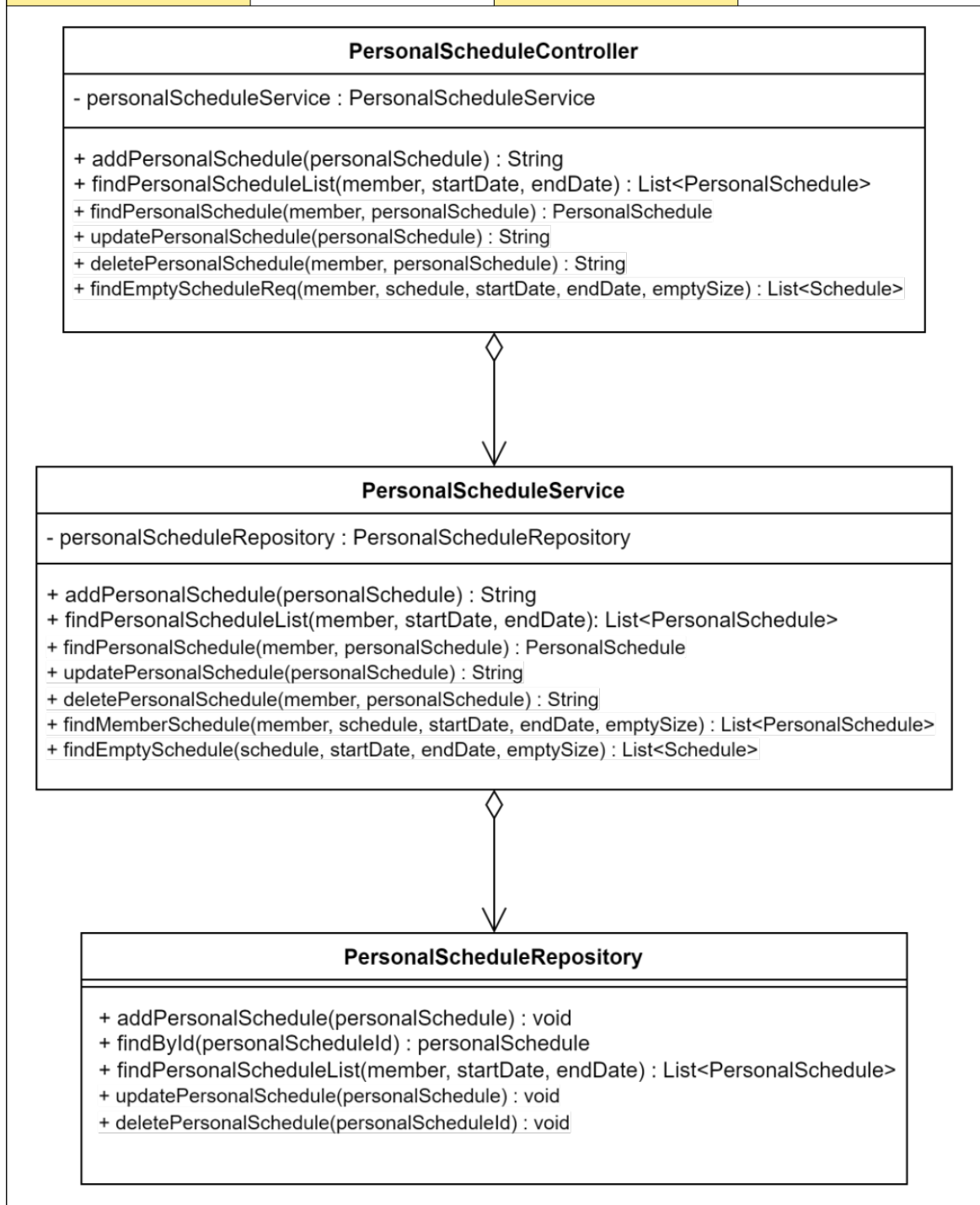
4. 시스템 설계도

4-1. 클래스 다이어그램 설계

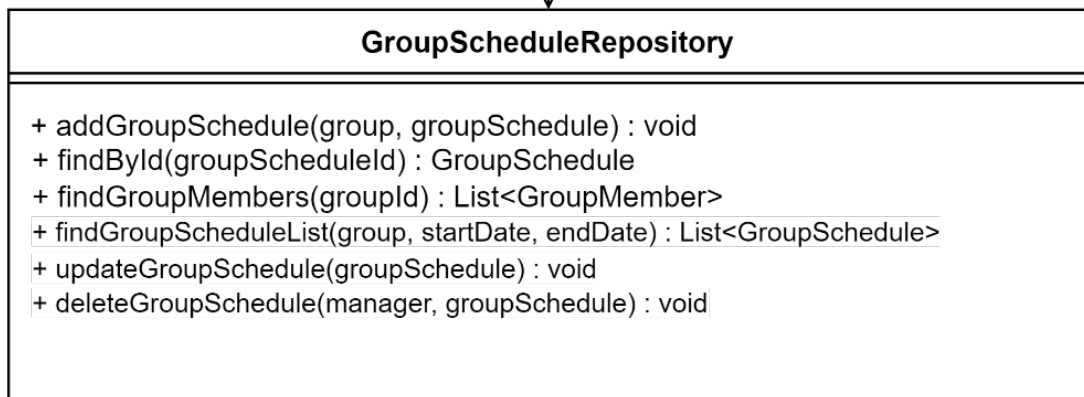
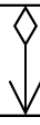
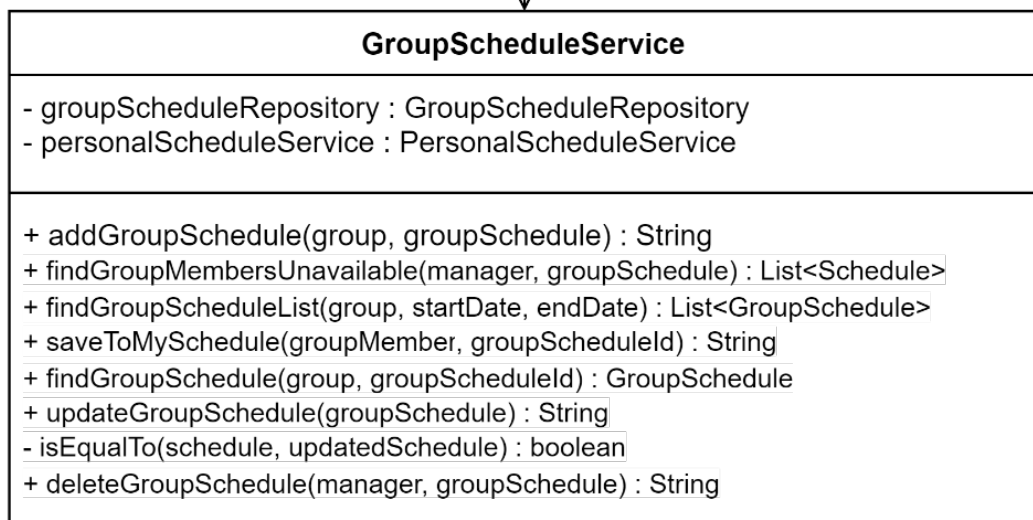
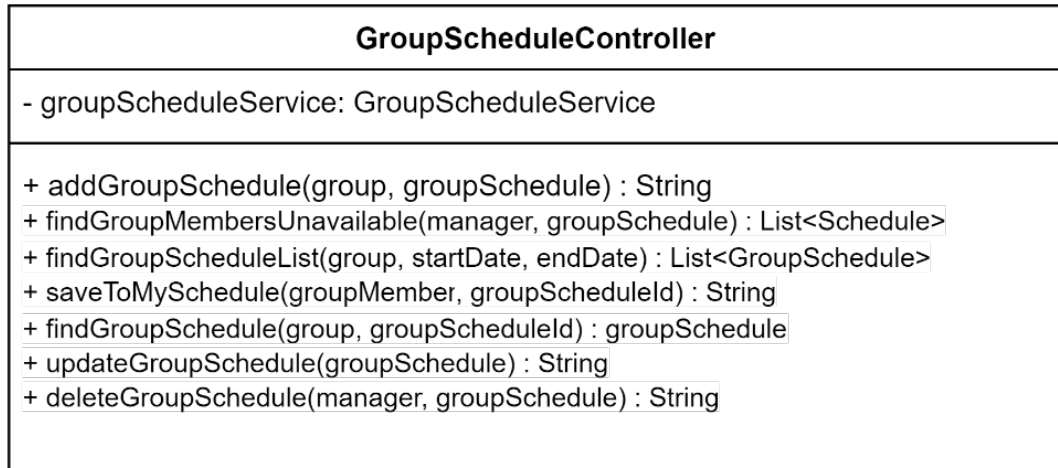




서비스시스템명	개인 일정 관리	클래스 다이어그램	CD-03
서비스시스템 식별자	SS-03	식별자	



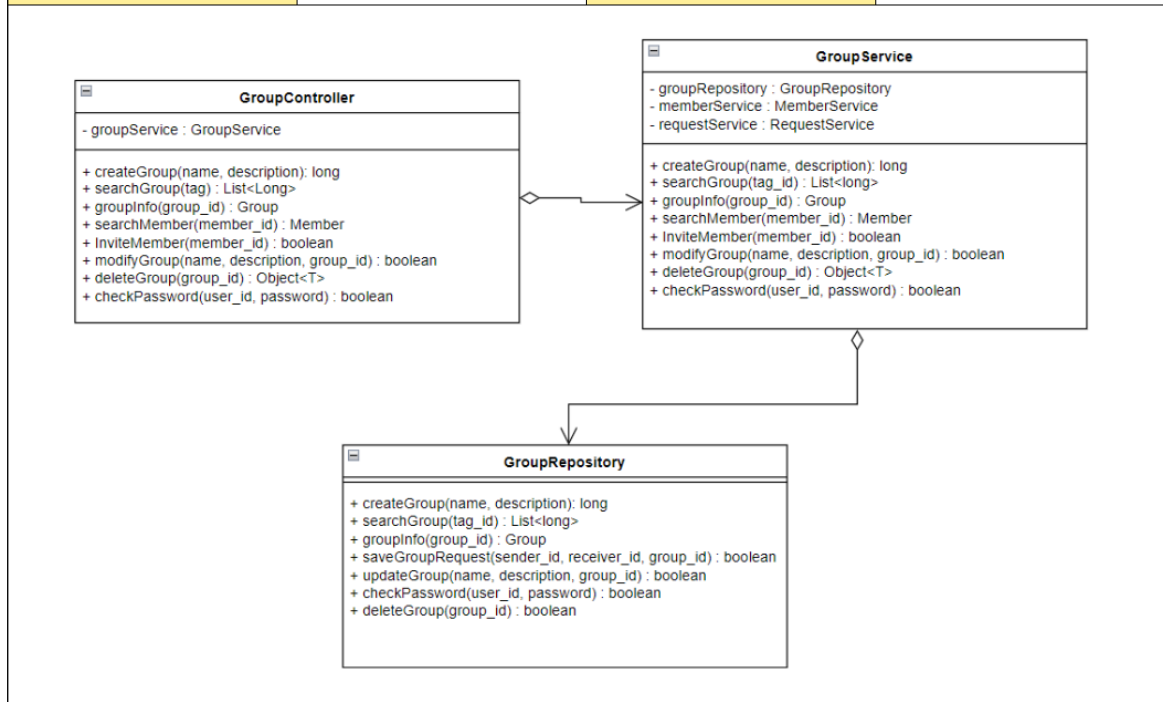
서비스시스템명	개인 일정 관리	클래스 다이어그램	CD-04
서비스시스템 식별자	SS-04	식별자	

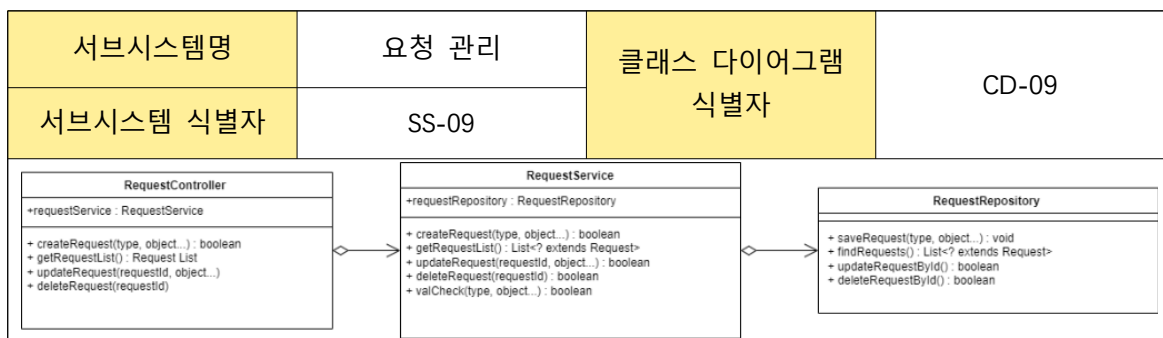
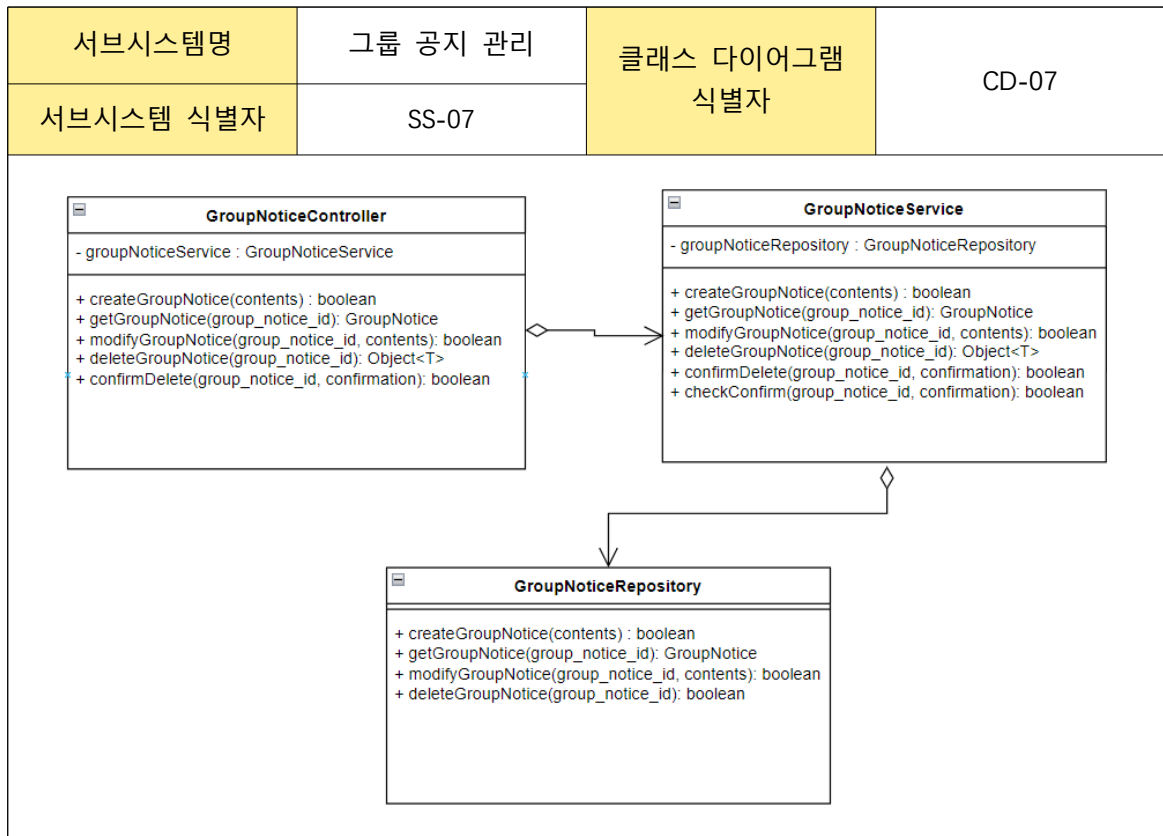


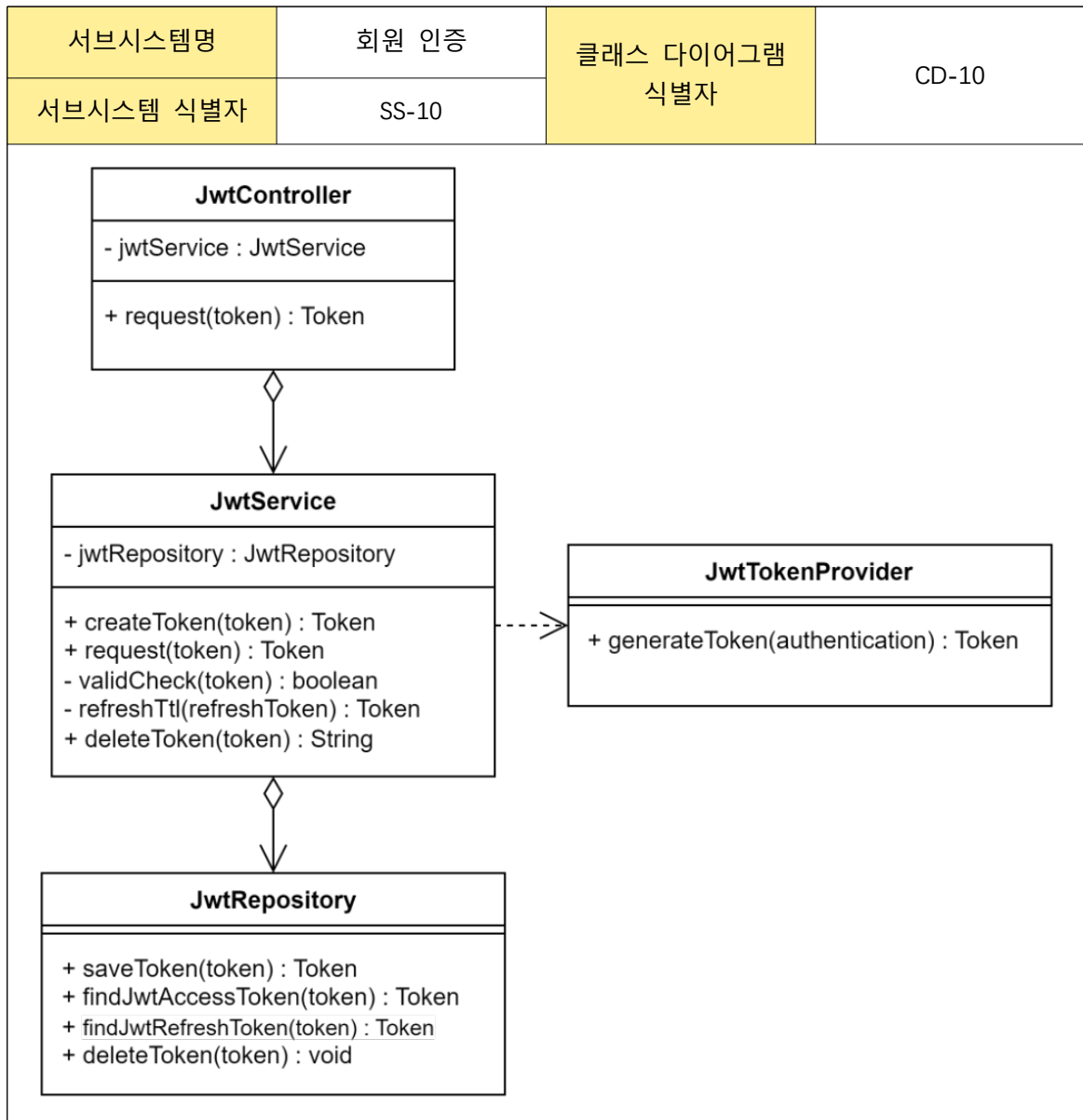
서비스시스템명	그룹 일정 관리	클래스 다이어그램	CD-05
서비스시스템 식별자	SS-05	식별자	



서비스시스템명	그룹 관리	클래스 다이어그램	CD-06
서비스시스템 식별자	SS-06	식별자	

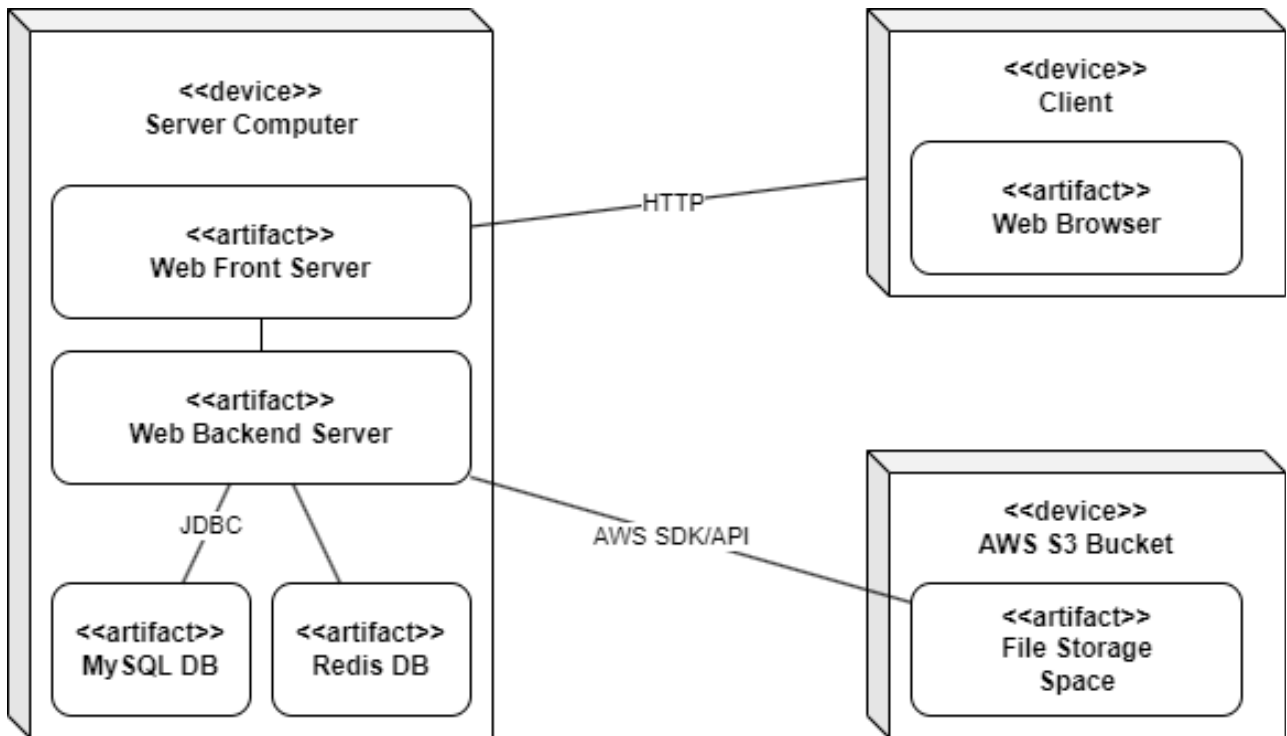






5. 시스템 구성 및 환경

5-1. 배치 다이어그램



5-2. 개발 환경

구분	개발 환경
운영체제	Windows 11, MacOS 13, Linux
개발언어	Java, JavaScript
IDE	IntelliJ, Visual Studio Code
클라우드 서비스	AWS S3 Bucket
백엔드 프레임워크	Spring
데이터베이스	MySQL, Redis
프론트엔드 라이브러리	React

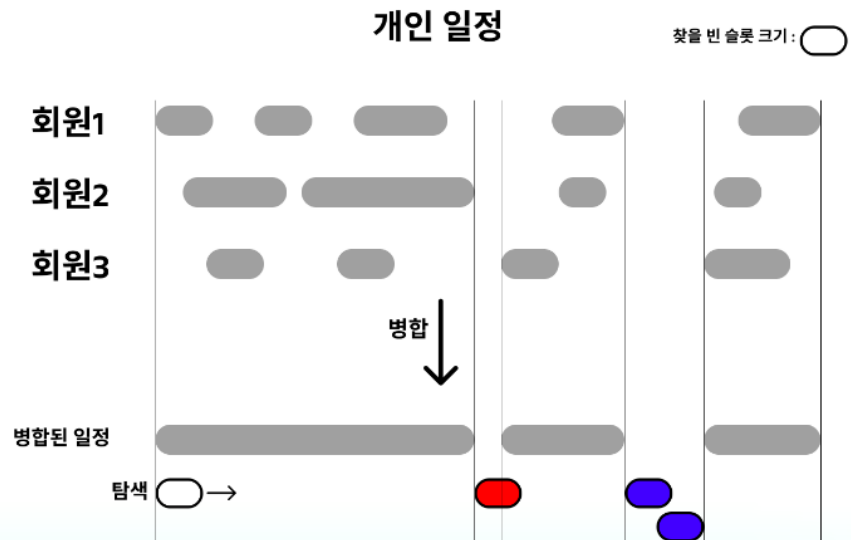
6. 구현 기술

6-1. 빈 일정 찾기 알고리즘

주요 기능

빈 일정 찾기 - 시간 단위

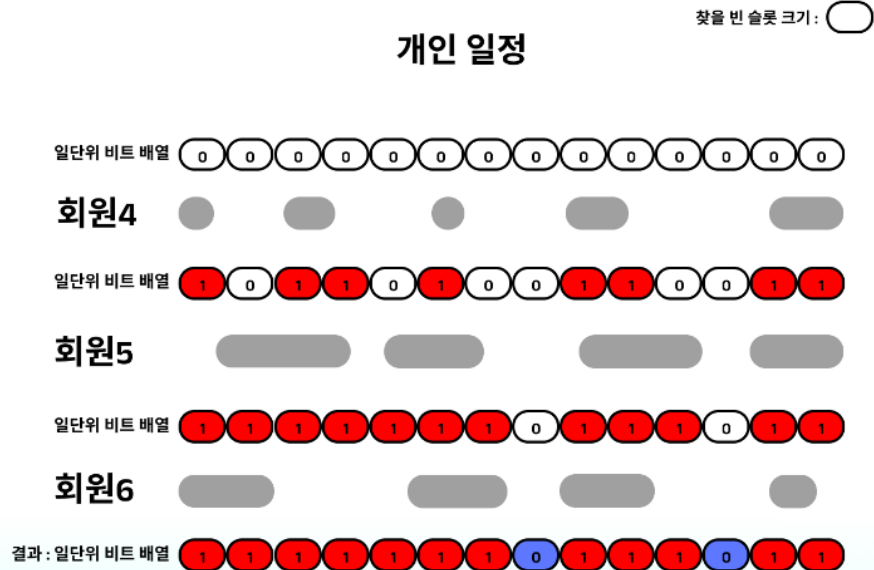
인터벌 병합 알고리즘



주요 기능

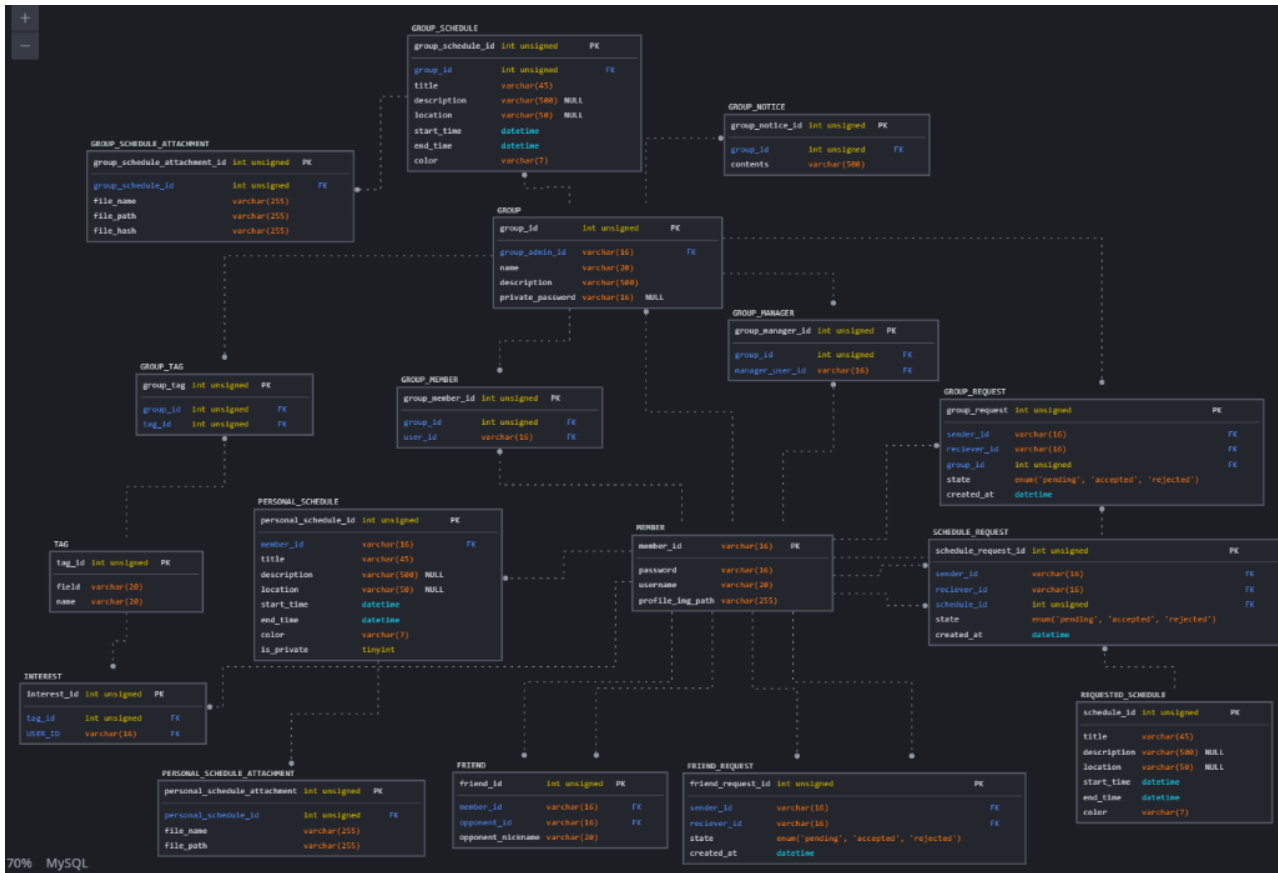
빈 일정 찾기 - 1일 단위

브루트포스 개선 탐색

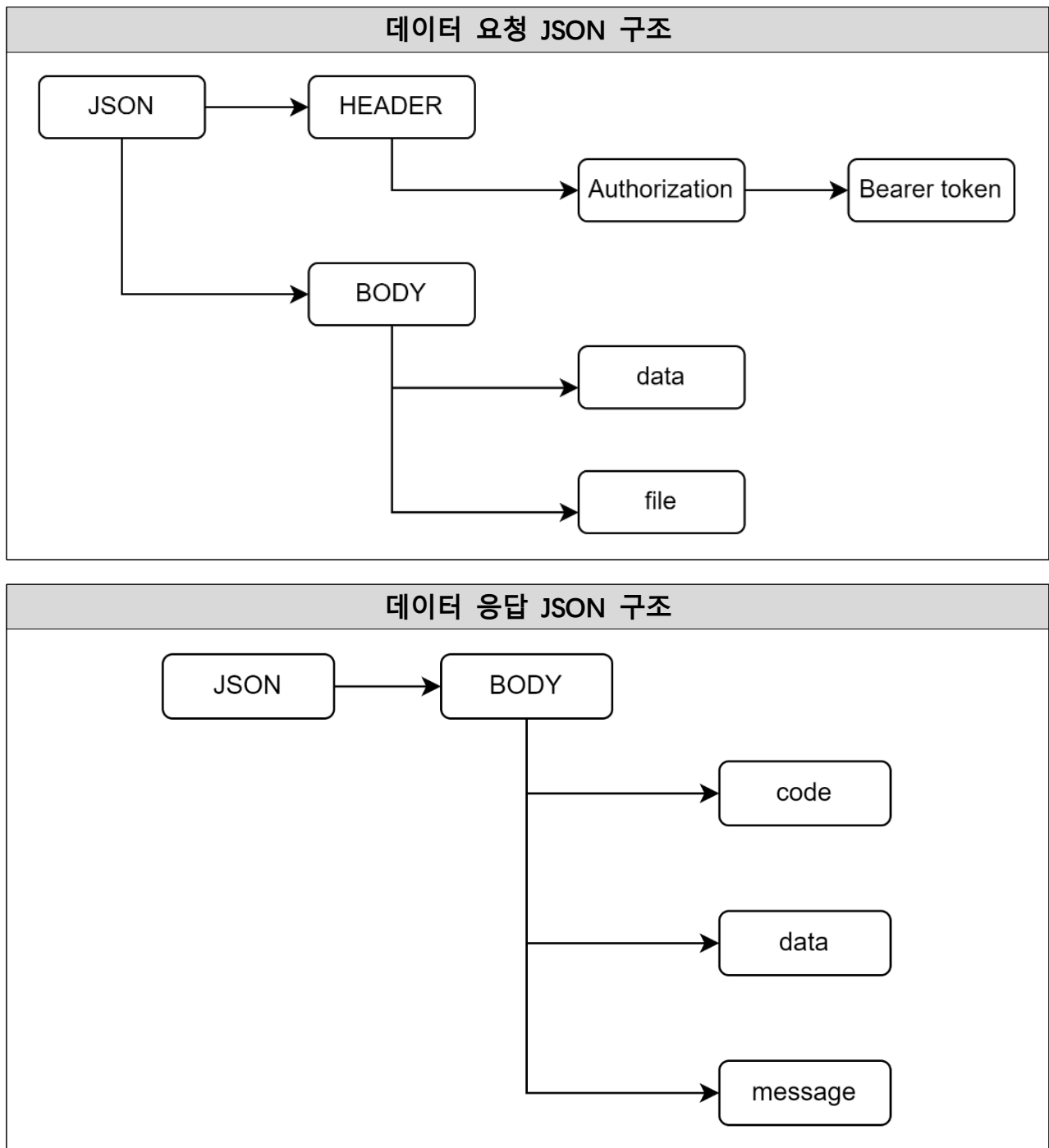


7. 주요 자료 구조 또는 DB 구조

7-1. 데이터베이스 구조



7-2. RestFul API JSON 데이터 통신 구조



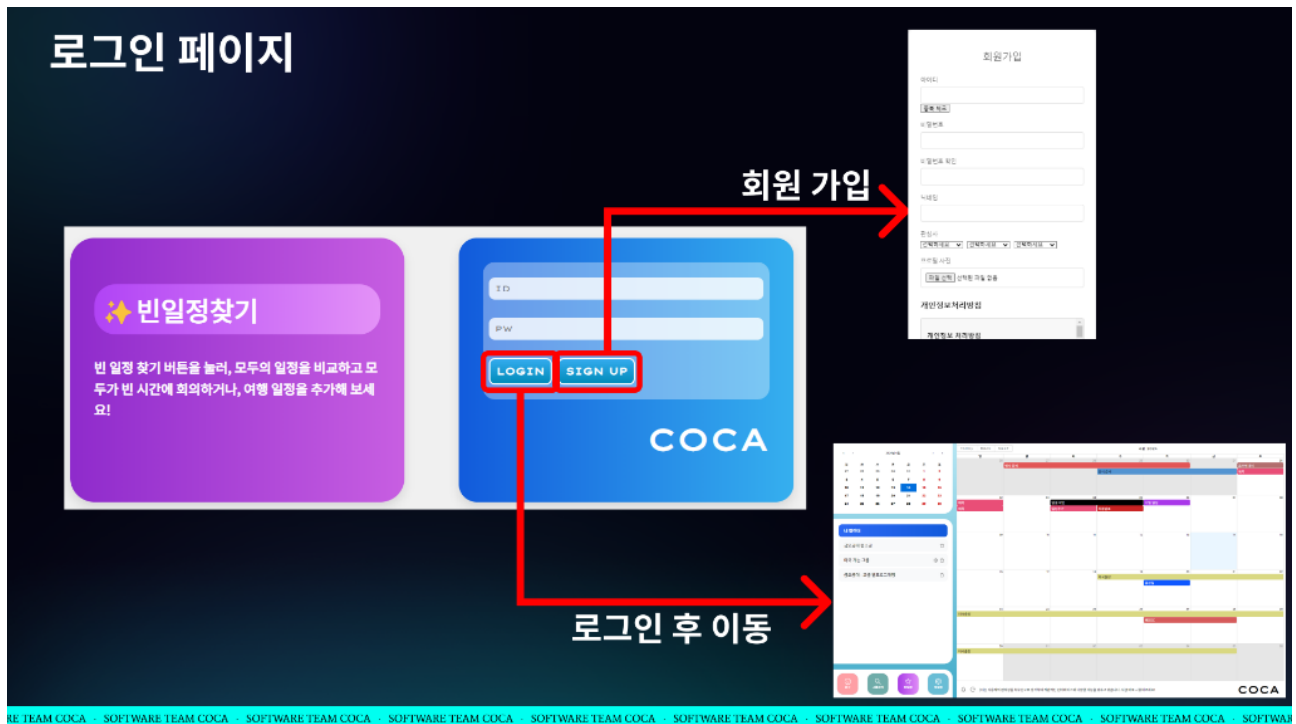
8. 주요 함수 명세

함수명	<code>private List<CommonSchedule> bruteForce(LocalDate startDate, int duration, int period, List<String> members)</code>
함수 설명	<p>각 멤버에 대해 루프를 돌면서, 주어진 기간 동안 각 날에 개인 일정이 있는지 확인합니다.</p> <p>startTime과 endTime을 이용하여 각 날의 범위를 설정합니다.</p> <p>personalScheduleRepository.findPersonalScheduleByDateRange메서드를 사용하여 각 멤버의 일정 데이터를 가져옵니다.</p> <p>만약 일정이 존재하면 해당 날을 false로 설정하여, 해당 날에 공통 일정이 불가능함을 표시합니다.</p> <p>startTime과 endTime을 하루씩 증가시키며 다음 날로 이동합니다.</p>
소스코드	
<pre>private List<CommonSchedule> bruteForce(LocalDate startDate, int duration, int period, List<String> members) { boolean daySlot[] = new boolean[period]; Arrays.fill(daySlot, true); for (String memberId : members) { LocalDateTime startTime = startDate.atTime(0, 0, 1); LocalDateTime endTime = startDate.atTime(23, 59, 59); for (int i = 0; i < period; i++) { if (daySlot[i]) { List<PersonalSchedule> personalSchedules = personalScheduleRepository.findPersonalScheduleByDateRange(memberId, startTime, endTime); if (personalSchedules != null && personalSchedules.size() > 0) daySlot[i] = false; } startTime = startTime.plusDays(1); endTime = endTime.plusDays(1); } } List<CommonSchedule> resultSchedule = new ArrayList<>(); for (int i = 0; i <= daySlot.length - duration; i++) { boolean isAvailable = true; for (int j = 0; j < duration; j++) { if (!daySlot[i + j]) { isAvailable = false; break; } } if (isAvailable) resultSchedule.add(new CommonSchedule(startDate.atStartOfDay().plusDays(i), startDate.atStartOfDay().plusDays(duration + i))); } return resultSchedule; }</pre>	

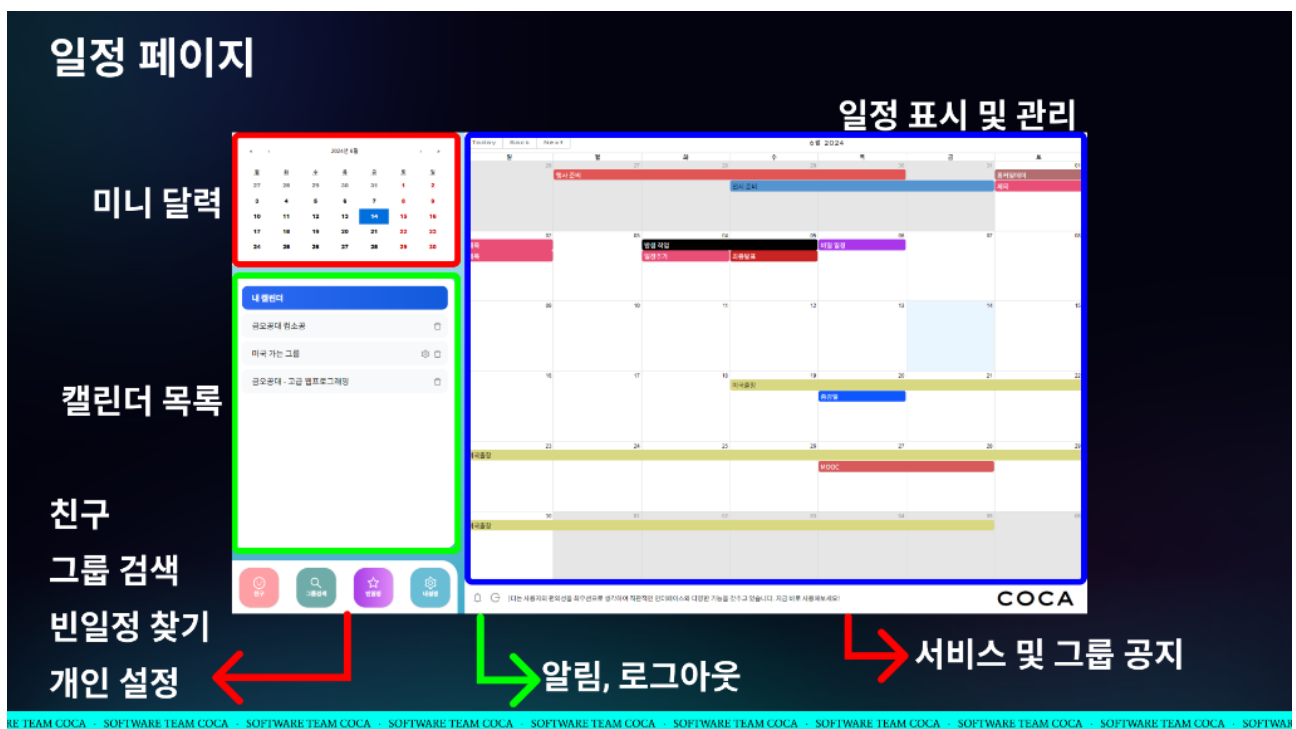
함수명	<code>private List<CommonSchedule> interval(LocalDateTime startTime, LocalDateTime endTime, int duration, List<String> members)</code>
함수 설명	IntervalMerge.intervalMerge(combined)를 사용하여 겹치는 일정을 병합합니다. 병합된 일정을 시작 시간 기준으로 정렬합니다. 병합된 일정(mergeSchedule)을 순회하며 각 빈 시간대를 찾습니다. 현재 시간(current)과 병합된 일정의 시작 시간(interval.getStart()) 사이에 빈 시간대가 있으면, 그 사이에 공통 일정을 추가합니다. current시간을 timeSlot만큼 증가시켜 다음 빈 시간대를 검색합니다. 현재 시간이 병합된 일정의 끝 시간(interval.getEnd())보다 이전이면, 현재 시간을 병합된 일정의 끝 시간으로 업데이트합니다.
소스코드	
<pre> private List<CommonSchedule> interval(LocalDateTime startTime, LocalDateTime endTime, int duration, List<String> members) { List<Interval> combined = new ArrayList<>(); final int timeSlot = 10; for (String memberId : members) { List<PersonalSchedule> personalSchedules = personalScheduleRepository.findPersonalScheduleByDateRange(memberId, startTime, endTime); if (personalSchedules != null && personalSchedules.size() > 0) { for (PersonalSchedule schedule : personalSchedules) combined.add(new Interval(schedule.getStartTime(), schedule.getEndTime())); } } List<Interval> mergeSchedule = IntervalMerge.intervalMerge(combined); Collections.sort(mergeSchedule, Comparator.comparing(Interval::getStart)); List<CommonSchedule> resultSchedule = new ArrayList<>(); //빈 일정이 담기는 리스트 LocalDateTime current = startTime; for (Interval interval : mergeSchedule) { while ((current.plusMinutes(duration).isBefore(interval.getStart()) current.plusMinutes(duration).isEqual(interval.getStart()))) { resultSchedule.add(new CommonSchedule(current, current.plusMinutes(duration))); current = current.plusMinutes(timeSlot); // 다음 빈 시간대를 검색하기 위해 시간 증가 } // 현재 시간을 다음 일정의 끝 시간으로 업데이트 if (current.isBefore(interval.getEnd())) current = interval.getEnd(); } while (current.plusMinutes(duration).isBefore(endTime) current.plusMinutes(duration).isEqual(endTime)) { resultSchedule.add(new CommonSchedule(current, current.plusMinutes(duration))); current = current.plusMinutes(timeSlot); } return resultSchedule; } </pre>	

9. 사용자 매뉴얼

로그인 페이지



일정 페이지



그룹 검색 페이지

그룹 검색 창 →

그룹 검색 결과 →

그룹 상세 정보

그룹 생성 버튼 →

그룹 참가 및 초대 버튼 →

금오공대

#소프트 #자바 #리액트 #자바스크립트 #일본 #미국 #영국 #호주 #동유럽 #자카르타 #인도네시아 #부르키나

검색

그룹명	인원
금오공대 - 고급 웹프로그래밍	2명
금오공대 컴소공	2명
금오공대 - 웹프로그래밍	1명

생성

금오공대 컴소공

[관리자] 스프링이좋은사람

금오공대 - 컴퓨터소프트웨어공학과 그룹입니다.
#자바 #공모전 #자바스크립트

그룹 초대

그룹 관리 페이지

그룹 이름

그룹 설명

그룹 공지사항

그룹 매니저 설정

그룹 관련 분야 설정

그룹 비밀번호 설정

그룹 수정

그룹 기본정보

금오공대 컴소공

금오공대 - 컴퓨터소프트웨어공학과 그룹입니다.

공지사항

그룹 매니저

매니저 > 매니저 > 추가

스프링이좋은사람 나루

해임 선택

그룹분야

자바 > 공모전 > 자바스크립트 >

비밀번호

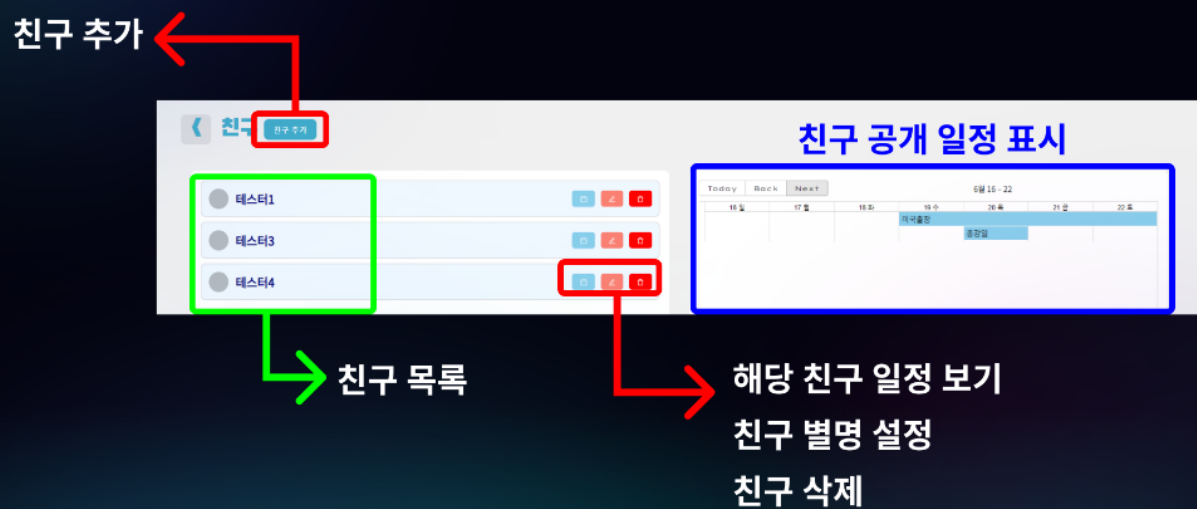
비밀번호

저장 삭제 취소

알림 페이지



친구 페이지



빈일정 찾을 회원 설정

회원 일정(파랑)
빈일정(빨강)

10. 역할 분담

팀원명	작업
이수찬	프로젝트 총괄 데이터베이스 설계 시스템 구조 설계 Spring 백엔드 기능 구현
이상현	프론트 서버 통신 구현 빈일정 찾기 알고리즘 설계 및 구현 Spring 백엔드 통신 코드 일부 및 CORS 관리 프론트 UI 일부 보조
임희열	React 웹 UI/UX 설계 및 구현 프론트 서버 통신 보조
이채연	Spring 백엔드 기능 구현 프론트 서버 통신 보조 빈일정 찾기 알고리즘 적용 첨부파일 MD5 비교 알고리즘 구현

11. 팀원별 프로젝트 감상문

11-1. 이수찬

감상문

- 프로젝트 주제 선정

COCA 프로젝트가 시작하기 전 생각해 둔 프로젝트 주제는 두 가지가 있었다. 하나는 도전성이 강하고 학교 프로젝트로서 다소 위험성이 있는 주제, 그리고 또 하나는 안정적인 주제, 이 COCA 프로젝트였다. 이 COCA 프로젝트는 내가 학교 생활하면서 시험 일정과 시험 전과 후 수업 관련 일정 공유에 차질이 있었던 경험을 토대로 나온 주제였다. 그리고 팀원이 결정되면서 팀원들의 방향성과 개인 능력들을 고려하여 이 COCA 프로젝트를 채택하게 되었다.

- 프로젝트 진행 중 문제점

우선 서비스 개발 프로젝트로서 기능이 많았던 점, 학과 일정보다 2주 앞서 진행했지만, 두 차례 서비스 내의 연관성 문제에 의한 기능 변경, 갑작스러운 학과 행사 참여 등으로 분석, 설계, 구현 기간이 많이 촉박해졌다. 이에 따라 명세에 없었지만 추가된 기능이 많았고, 설계 부족으로 작업 효율성이 떨어졌으며, 최종 발표 10분 전까지 구현을 하고 있었던 큰 문제가 발생하였다. 이로 인해 최종 발표를 제대로 준비하지 못했고, 이에 많은 고생을 해준 팀원들에게 큰 감사와 함께 미안함을 전하고 싶다.

- 프로젝트 중 도전하고 문제를 극복한 방법

우선 문제를 극복한 방법으로 팀워크를 꼽고 싶다. 팀원 간의 편안한 소통 방식과 작업의 문제가 있었던 점에 대해 이해하는 분위기, 여러 문제가 발생하여도 빠르게 받아들이고 해결 방안을 바로 모색하고자 하는 자세 등 팀워크적인 부분에서 팀원 간의 컴플레인도 없고 갈등도 없이 프로젝트를 원활히 진행할 수 있었다.

그리고 팀원들이 해보고 싶은 시도는 기능 연관성이 부족한 내용을 제외해 받아주는 편이었다. 빈 일정 찾기 알고리즘과 MD5 비교 알고리즘, AWS 버킷 CORS 권한 다운로드 설정 등이 있다. 성능적으로 기존과 새로운 시도 간의 비교가 어려울 경우, 유사한 기능들에 다르게 적용함으로써 성능 비교할 수 있었다. 또한 노선에 JSON 통신 포맷 DOC 작성을 함으로써 프론트 통신 담당자가 작업을 원활히 진행하도록 도움을 줄 수 있었다.

또한 불안했지만 생각보다 얻은 것도 많았던 도전은 학과 행사인 홈커밍 데이에 전시 참가하는 것이다. 처음에는 최종 발표를 앞둔 주말에 전시 행사는 긴박한 작업 기간에 선불리 참여하기 어려운 일정이었다. 하지만 행사에 참여함으로써 프로젝트의 임시 통합 테스트도 진행하고 사용자의 반응과 함께 예상 못했던 버그들도 발견되면서 다소 얻은 점도 많은 도전이었다.

- 프로젝트의 개선점과 향후 계획

프로젝트의 총괄인 만큼 COCA 프로젝트에 애착을 갖고 진행하였다. 하지만 긴박한 일정으로 인해 못했던 시도와 아쉬웠던 점, 하고싶었던 점이 많다. Security 시스템 강화, Docker, 클라우드 서비스를 활용한 MSA 구축, UI/UX 개선, 반응형 웹 혹은 모바일 애플리케이션 확장, 알고리즘 성능 개선, 기존 일정 서비스 연동, 테스트 코드 작성과 배포 시도, CI/CD를 통한 원활한 작업 환경 구성, 설계 문서 재작성 등을 고려하고 있다. 모두 수행하긴 어려울 것 같지만 선택적으로 개선하여 차기 버전으로 새로 개발하는 것도 좋은 프로젝트로서의 방향인 것으로 보인다.

11-2. 이상헌

감상문

COCA 프로젝트를 진행하며 함께 달려와 줬던 팀원들에게 이 글을 바치고 싶다.

이번 프로젝트를 진행하며 기능이 수도 없이 바뀌고, 이미 했던 작업을 다시 한번 더 해야 하는 등 수많은 시행착오와 우여곡절을 겪었던 것 같다. 하지만 이 때문에 더욱더 프로젝트의 정체성을 찾을 수 있었고, 우리 팀의 역량과 협업 능력을 한층 더 발전시킬 수 있었다고 생각한다.

솔직하게 처음 프로젝트를 시작할 때는 팀원들의 강점이 명확했고 이에 단순히 어떻게든 잘 되겠지라는 생각으로 프로젝트에 임했었으나, 프로젝트가 진행되면서 계속 변경되는 기능과 이로 인한 점점 더 촉박해지는 개발 기간으로 인해 큰 압박감을 느꼈다. 반복되는 수정 작업과 새로운 문제들이 끊임없이 발생했고, 이를 해결하면서 막연하게 생각하던 나 자신을 되돌아보고 생각을 고쳐먹을 수 있게 되었다.

이번 프로젝트를 진행하면서 배운 가장 큰 교훈은 '유연성'의 중요성이다. 예기치 못한 문제들이 발생할 때마다 우리는 계획을 수정하고, 새로운 접근 방식을 시도해야 했다. 이러한 유연한 태도가 우리 팀이 어려운 상황에서도 흔들리지 않고 목표를 향해 나아갈 수 있게 해주는 원동력이 되었다고 생각한다.

또한, 서로에 대한 신뢰와 존중이 얼마나 중요한지도 다시금 깨달았다. 저번 프로젝트를 진행하면서 팀원 간의 불화가 조금 있었기에 안되면 내가 하지, 하는 마음도 어느 정도 있었으나 촉박한 개발 기간 탓에 도저히 혼자서는 해낼 수 없는 분량의 업무가 다가왔고, 큰 압박감을 느끼고 있었던 적이 있었다. 이때 팀원이 내밀어 줬던 도움의 손길은 아직도 잊을 수 없다. 먼저 도움의 손길을 내밀어 준 덕분에 나는 팀원을 더욱 신뢰하고 존중할 수 있었고, 그로 인해 그 많은 분량의 업무를 처리할 수 있었던 것 같아 너무 고맙게 생각한다.

프로젝트의 후반부에 들어서면서 우리는 더욱 긴밀하게 협력했고, 서로의 강점을 최대한 활용하면서 문제를 해결해 나갔다. 각자가 맡은 역할을 충실히 수행하면서도, 필요한 순간에는 주저 없이 서로가 서로를 도움으로써 격려했고 이를 통해 프로젝트를 성공적으로 마무리할 수 있었던 것 같다.

마지막으로, 이번 프로젝트를 통해 배운 경험과 감정, 그리고 교훈은 앞으로 우리 모두에게 큰 자산이 될 것이다. 나, 그리고 분명 다른 팀원들 또한 이번 경험을 바탕으로 앞으로도 더욱더 큰 도전을 해낼 수 있다고 믿어 의심치 않는다.

함께 노력하며 끝까지 달려와 프로젝트를 성공으로 이끌어준 모든 팀원에게 진심으로 감사의 마음을 전하며, 앞으로도 서로를 응원하며 성장해 나갈 수 있기를 바란다.

11-3. 임희열

감상문

COCA 프로젝트는 분업화가 매우 잘 이루어진 조직이었다. 조직은 크게 백엔드와 프론트엔드로, 내가 속한 프론트엔드 내에서는 UI구성과 서버통신으로 나누어졌다. 나는 UI구성과 상태를 관리하고, 서버통신 담당 팀원과 협력하여 프론트엔드 작업을 끝내는 역할을 배정받았다.

나는 기획, 설계 단계에서 와이어프레임과 UI디자인을 모두 진행해야 했고, 그만큼의 권한도 위임되었지만, 아무런 요구나 가이드라인 없이 처음부터 UI를 짚다는 것은 매우 어렵고 시간이 오래 걸리는 일이었다. 두달간 총 8번의 디자인 개편과 백엔드의 피드백을 반복해 최종 프로토타입을 기준으로 구현을 시작했다. 대규모 웹 프로젝트가 처음이었고, 디자인했던 것 그대로 구현할 수 있을지가 항상 의문이었지만, 기존 디자인과 동일하지는 않지만 흐름 자체는 계획과 거의 동일하게 구현될 수 있었다. 특히, 많은 개발자들이 이미 많은 시행착오를 겪으며 문제를 해결했고, 그 결과물로 나온 라이브러리들을 적절히 커스터마이징하여 사용함으로써 문제를 해결했기 때문에 구현 자체에는 큰 어려움을 겪지 않았다.

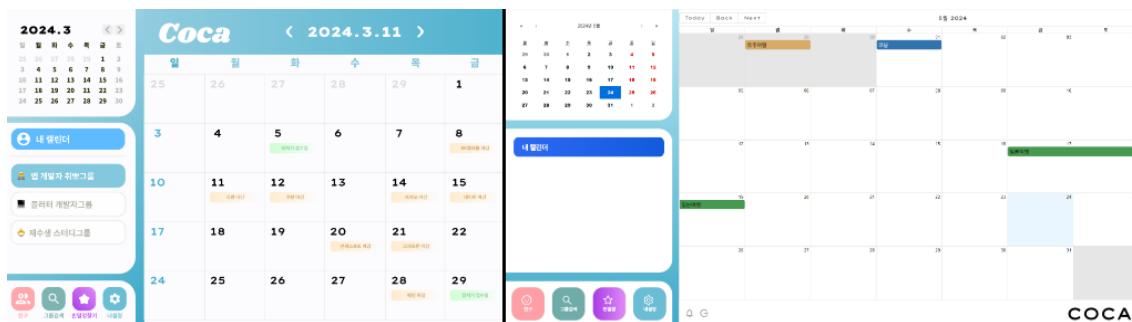


사진1) 좌: 프로토타입, 우: 실제 구현된 모습

캘린더는 어렵다. 항상 사용자의 입장이었고, 무수히 많은 캘린더를 접했기 때문에 캘린더 구현은 항상 쉽다고만 생각해 왔지만, 캘린더에는 수많은 예외와 상태가 존재했다. 매월, 매년 매번 다른 테이블을 출력하는데다, 연속된 일정은 합치고, 여러 일정이 있을 때에는 어떻게 표시하며, 일정이 두 달에 걸쳐있다면 어떻게 표시해야 하는지, 상태를 어떻게 가공하여 렌더링할지 등 직접 캘린더를 구현하는 것이 주어진 인적/시간적 자원만으로는 사실상 불가능했다. 처음부터 직접 구현한다는 것은 불가능함과 더불어서 계획한 디자인까지 도달하지 못하는 요소이기도 했고, 그래서 더욱더 주어진 라이브러리들을 탐색하고 커스터마이징하여 활용하는데 집중했다.

별도의 디자인 시스템을 만들지 않았기 때문에 그때그때 적절한 폰트와 메인컬러와 유사한 색상을 사용해 제작되었다. 제한된 시간 내에 구현하고 여러 라이브러리를 활용하다 보니, 디자인의 일관성이 어긋나는 부분이 많았고, 상대크기를 사용하는 코드의 특성상 크기의 일관성 또한 제각각인 경우가 많았다. 그래서 이번 프로젝트는 정말 기초적인 일관성만을 사용자에게 경험으로 제공하는 부분이 있었고 이를 디자인 시스템의 제작과 FigJam 사용을 통한 반응형 프로토타입 제작을 통해 해결해야 할 필요성을 크게 느낄 수 있었다.

UI와 상태를 만들면 서버통신 담당 팀원은 해당 코드를 이해하고 백엔드와 프론트엔드를 연결하는 작업을 구현하였다. 하지만, 페이지가 가지고 있는 상태만 20개가 넘는 경우도 많았기 때문에 해당

팀원이 UI 코드를 이해하는데 상당한 시간과 노력을 필요로 했을 것으로 생각한다. 그래서 상태와 시나리오를 반드시 문서화해야겠다는 필요성을 느끼기도 했다.

이번 COCA 프로젝트는 팀 구성원 모두가 실제 운영되는 웹사이트를 개발한다는 느낌으로 진행되었다. 모든 예외상황과, 세부적인 디테일까지, 단순한 과제물이라고는 볼 수 없는 4명의 노력이 투입되었고 마지막까지 이러한 초심을 유지하며 프로젝트를 완료할 수 있었다. 프로젝트를 전반적으로 기획하고 변경된 일정에도 유연하게 프로젝트를 총괄해주신 백엔드의 수찬PM님, 백엔드와 프론트엔드 두 코드를 오가며 프로젝트 완성에서 활약해주신 풀스택의 상현PL님, 좋은 상황판단력과 많은 피드백으로 프로젝트를 더 나은 방향으로 이끌어주신 백엔드의 채연PL님 모두 좋은 팀원이었음에 감사함을 느낀다.

11-4. 이채연

감상문

이번 프로젝트는 많은 우여곡절을 겪은 프로젝트였다. 우선, 요구 명세서가 교수님과 팀원들의 의견에 따라 구현 단계 직전까지 여러 번 변경되었고, 이로 인해 개발 단계로의 진입이 늦어져 지금까지 해온 프로젝트 중 가장 개발 기간이 촉박했다. 하지만 이러한 상황에서도 얻어갈 것이 참 많은 프로젝트였다.

교수님께서 우리 팀의 프로젝트 요구 명세서를 보시고 여러 종류의 기능이 혼합되어 있어 ‘공유 캘린더’라는 이름에 걸맞지 않다고 판단하셔서 ‘어떤 것을 만들고 싶은지 모르겠다’는 말씀을 하셨다. 당시 우리 프로젝트는 캘린더와 SNS가 결합된 형태였는데, 문서 요약 인공지능 기능을 제외하고 SNS 기능을 강화한 것이 문제가 되었다. 교수님의 피드백을 받은 후 팀원들과 모여 깔끔하게 떨어낼 것은 떨어내고, 공유 캘린더로서 추가할 수 있는 기능을 고민해 본 결과, 주요 기능인 ‘빈 일정 찾기’ 기능을 추가했다. 요구 명세서를 확립하기 전까지 총 5번의 수정이 있었고, 이를 통해 개발하고 싶은 시스템의 중요 포인트를 잃지 않는 것이 중요하다는 것을 깨달았다. 우리 팀은 특별한 기능에 치중하여 캘린더에서 중요시 되는 커뮤니티 기능을 계속 추가했고, 그 결과 ‘공유 캘린더’라는 중심 포인트에서 멀어졌다. 교수님의 피드백과 팀원들 간의 상의가 아니었다면 깨닫지 못했을 일이라, 귀한 경험이라고 생각한다.

또한, 개발 기간이 촉박했기 때문에 개발을 완성시키는 것에 집중한 나머지 코드의 재사용성과 수정의 용이성을 고려하지 않은 코드가 몇 가지 있었다. 추후 해당 코드를 수정할 일이 생겼을 때, 한 메소드의 수정으로 인해 다른 메소드들까지 함께 수정해야 하는 곤혹을 겪었다. 프로젝트의 마감 기한을 맞추는 것도 중요하지만, 완성도와 지속성을 위해서는 추후 코드의 수정과 확장까지 염두에 두고 작성해야 한다는 것을 다시금 느꼈다.

코카팀은 팀원들 간의 소통이 매우 원활했던 팀이었다. 문제가 생기면 팀원들끼리 바로 상의하여 문제를 해결하였고, 진행 상황의 공유도 원활하게 이루어졌다. 또한, 팀원 간의 업무 분담도 잘 이루어져서 분업이 잘 되어서 성공적으로 프로젝트를 마무리할 수 있었다.

이번 프로젝트는 많은 시행착오를 겪으며 많은 것을 배운 프로젝트였다. 요구 명세서의 잦은 변경과 촉박한 개발 기간 속에서도, 교수님의 피드백과 팀원들 간의 협력을 통해 프로젝트의 본질을 지켜낼 수 있었다. 또한, 코드의 재사용성과 유지보수의 중요성을 다시금 깨닫게 된 소중한 경험이었다. 코카팀의 원활한 소통 덕분에 문제를 신속하게 해결할 수 있었고, 팀워크의 중요성을 다시 한번 확인할 수 있었다. 혼자 했다면 완성시키지 못할 프로젝트를 성공적으로 마무리시킬 수 있었기에, 팀원들에게 다시금 고마움을 느꼈다.

12. 팀원별 프로젝트 포트폴리오

12-1. 이수찬

사용 기술
<ul style="list-style-type: none">• Language : Java• Framework : Spring, Spring Security• API : JPA• IDE : VSCode, IntelliJ Ultimate• 설계 Tool : draw.io• 버전 관리 : Git, GitHub• 클라우드 : AWS S3 Bucket• 데이터베이스 : MySQL, Redis• 보안 메커니즘 : JWT
주요 역할
<ul style="list-style-type: none">• 프로젝트 매니저<ul style="list-style-type: none">- 프로젝트 전체 관리- 일정 계획 수립- JSON API Document 작성 : 프론트엔드와 백엔드의 원활한 통신을 위한 API Document 작성• 시스템 배치, 데이터베이스 설계<ul style="list-style-type: none">- 프로젝트의 전체적인 시스템 구성과 배치- AWS 구성- 데이터베이스 테이블 설계• 백엔드 개발<ul style="list-style-type: none">- 로그인, 로그아웃 Spring Security 적용- JWT와 Spring Security, Redis 연동- 개인 일정 관리 시스템 개발- 그룹 시스템 개발- 친구 시스템 개발- 요청 알림 시스템 개발- 인증 시스템 개발- AWS 첨부파일 관리 시스템 개발

12-2. 이상헌

Languages

- Java
- CSS
- JavaScript

Framework

- Spring

Libraries

- React
- Ant Design, MUI, SweetAlert2
- Axios, Redux

VCS & Tools

- IntelliJ
- Visual Studio Code
- Git, Github

역할

- 빈 일정 찾기 알고리즘 제작
- 프론트엔드 통신 코드 구현
- 백엔드 컨트롤러 일부 구현 및 CORS 관리
- 프론트엔드 UI 일부 구현

개발 주요 내역

1. 빈 일정 찾기 알고리즘 제작 및 테스트

- 빈 일정 찾기 알고리즘을 Java로 개발
- 인터벌 병합 알고리즘, 해싱, 이진 검색, 브루트포스, 세그먼트 트리 알고리즘을 응용하여 총 다섯 가지 방식으로 구현
- 각 알고리즘의 수행시간과 시간복잡도를 계산하여 검증 및 테스트
- 시간별, 일자별 상황에 맞는 알고리즘으로 인터벌 병합 알고리즘과 브루트포스 알고리즘을 선택

2. 프론트엔드 통신 코드 구현

- Axios 라이브러리를 활용해 팀원이 제작한 UI에 맞도록 통신 코드 구현
- UX를 고려한 상태 관리를 위해 localStorage, Redux 등을 활용한 구현
- UX를 고려한 사용자 친화적인 예외 처리 및 데이터 바인딩 구현

3. 백엔드 컨트롤러 코드 일부 구현 및 CORS 관리

- 백엔드의 응답 내용이 프론트엔드에 종속적이지 않도록 관리
- 일부 컨트롤러 및 어플리케이션 전체에서 발생하는 CORS 에러 및 예외 처리 수정

4. 프론트엔드 UI 일부 구현

- 회원 로그인 검증, 회원 탈퇴, 비밀번호 입력 등 통신이 수반되어야 하는 UI 구현 및 제작
- UI 내 동적 렌더링 및 기타 오류 수정

12-3. 임희열

기술 스택

디자인 : Figma
디자인 환경 : M1, macOS Sonoma 14.2.1
개발 언어 : React
IDE : VSC, CursorAI
담당 업무 : React 웹 UI/UX 개발

주요 색상

주요 색상은 진하늘색을 #43ADCB 을 주요 강조색상으로 사용하고 #91CCDF와 그라데이션을 주어 전체적으로 시원한 느낌을 더하였음. 코랄 핑크 #D06B74 을 대비되는 색상으로 사용하여 적절한 색상조합을 구성하였음. linear-gradient 를 CSS 에 적용하는 방법으로 색상들을 사용하였음.

주요 폰트

Noto Sans KR, Lexend Zetta, Black Han Sans

주요 아이콘 및 컴포넌트 요소

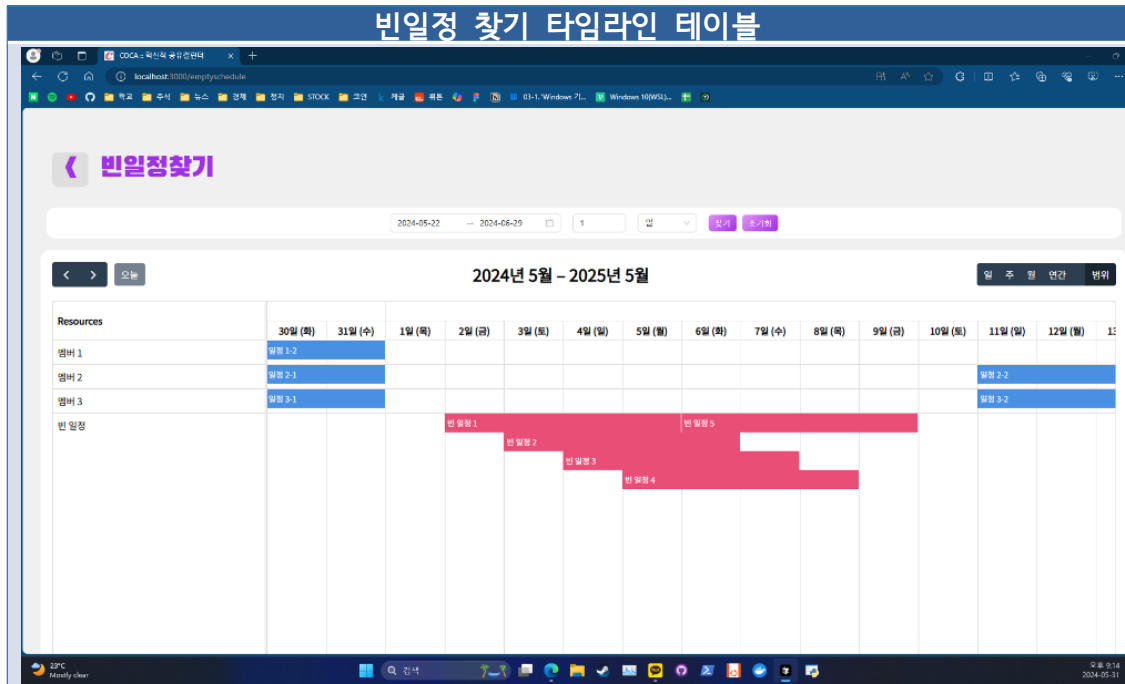
@ant-design/icons
antd
react-bootstrap
@mui/material/Pagination

CSS 스타일 전역 오염

타 컴포넌트의 CSS에 영향을 받는 것을 해결하기 위해서 CSS Module을 사용함.

메인 페이지 캘린더

react-big-calendar 라이브러리의 Calendar 를 메인 캘린더로 사용하였음. momentLocalizer를 사용하여 기본적인 한국어 구성을 따르도록 하고, onSelectSlot 함수를 통해 일자별 빈 슬롯을 선택했을 때 해당 날짜의 일정들이 표시되도록, 타 컴포넌트에 선택된 일정데이터를 넘기도록 연결하였음. 캘린더의 각 일정에 색상을 적용시키기 위해서 backgroundColor 을 상태에 저장된 일정색상으로 적용하도록 함. 좌측에 표시되는 미니 캘린더는 react-calendar 라이브러리의 RCalendar 를 사용하여 출력하였음.



COCA프로젝트의 핵심 기능인 빈일정 찾기는 fullcalendar 라이브러리의 FullCalendar 를 사용하여 타임라인 테이블을 구현함. @fullcalendar/resource-timeline 를 사용하여 기본적인 타임라인 구성을 완료하였으며, 백엔드에서 받아오는 멤버별 일정과 빈 일정을 적절히 가공하여 출력함.

fullcalendar 에서 제공되는 views 함수를 통해 일자단위로 검색했을 때와 시간단위로 검색했을 때의 커스텀 기간 뷰를 설정하였으며, 빈 일정은 이벤트가 겹치지 않고 겹칠 경우 여러 행을 사용하는 방향으로 eventOverlap 함수를 사용하여 처리함.

12-4. 이채연

기술 스택

개발 언어: JAVA, JavaScript
프레임워크: Spring
협업 도구: Git
IDE: IntelliJ, VSCode

구현 내용

빈 일정 찾기

빈 일정을 찾는 알고리즘의 기본 구조를 가져와 시스템에서 사용할 수 있는 형식으로 코드를 수정 및 적용함. 브루트포스 알고리즘의 경우 날짜를 기준으로 일정을 찾는다는 점을 이용하여 회원의 일정을 가져올 때 이미 인덱스에 일정이 있다면 그 날은 더 이상 찾지 않는 것으로 하여 일정 병합에서의 시간을 줄이는 방법으로 코드를 변경함.

회원 기능

회원가입, 로그인/아웃, 회원 정보 수정, 회원 탈퇴 등의 기본적인 기능 개발. 회원의 개인 정보를 수정 할 때 비밀번호를 한 번 더 검사하는 식의 보안을 위한 코드 작성. 통신에서도 JSON 포맷을 주고받는 방법으로 보안성을 향상

그룹 일정 기능

그룹의 일정과 관련된 기능 개발. 회원이 그룹에 속해있는지, 그룹이 존재하는지, 일정이 존재하는지 등의 여부를 판별해 정확성을 높이는 코드 작성. 그룹 일정의 파일이 수정되었을 때 파일을 전부 삭제하고 재업로드 하는 방식이 아닌, 파일의 해시코드를 데이터베이스에 미리 저장해두고 프론트에서 요청한 파일의 해시코드와 기존 파일의 해시코드를 비교하여 파일의 변경된 사항만 반영하는 코드를 작성하여 파일 업로드/삭제/수정에서 드는 비용을 줄임.

프론트 통신 작업 일부

프론트에서 백엔드로 그룹 정보 수정을 요청하는 코드 작성
빈 일정 찾기에서 적용되는 날짜 검사 코드 작성