종합설계 프로젝트 수행 보고서

프로젝트명	애자일 성과 분석 서비스
팀번호	**S1-3**
문서제목	수행계획서( ) **2**차발표 중간보고서( ) **3**차발표 중간보고서( ) **4**차발표 중간보고서(**O**) 최종결과보고서( )

2023.06.07

팀원 : 최정훈

(팀장)

권순호

박현준

염종협

지도교수 : 전광일 교수

문서 수정 내역

작성일	대표작성자	버전**(Revision)**	수정내용	
2022.02.23	최정훈 (팀장)	1.0	수행계획서	최초작성
2023.03.20	권순호	2.0	2차발표자료	설계서추가
2023.06.07	박현준	3.0	4차발표자료	시험결과추가
		4.0	최종결과보고서	시험결과 수정

문서 구성

진행단계	프로젝트 계획서 발표	중간발표**1** **(3**월**)**	중간발표**2** **(5**월**)**	학기말발표 **(6**월**)**	최종발표 **(10**월**)**
기본양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식	계획서 양식
포함되는 내용	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~3) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~4) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~5) 참고자료	Ⅰ. 서론 (1~6) Ⅱ. 본론 (1~7) 참고자료	Ⅰ Ⅱ Ⅲ

이 문서는 한국산업기술대학교 컴퓨터공학부의

“종합설계” 교과목에서 프로젝트 “애자일 성과 분석

서비스” 을 수행하는

(S1-3, 최정훈, 박현준, 권순호, 염종협)들이 작성한 것으로

사용하기 위해서는 팀원들의 허락이 필요합니다.

목 차

Ⅰ. 서론

1. 작품선정 배경 및 필요성

인적 자원은 대부분의 조직에서 소중한 자산이다. 인적 자원을 효율적으로 관리하는 것은 기업의 성장에 있어 무엇보다 중요한 요소라고 할 수 있다.

하지만, 대다수의 성과 관리 시스템은 아주 복잡하고 이해하기 어렵다. 2013년 SHRM(Society for Human Resource Management)의 통계에 따르면, 오직 23%만이 회사의 성과 관리 시스템을 보통 이상이라고 응답하였다. 성과를 독려하기 위해 제안된 방식이 오히려 과도한 경쟁을 부추기거나, 의욕을 떨어뜨리게 된 것이다.

2. 기존 연구/기술동향 분석

기존에 존재하는 성과분석 서비스인 LemonBase 프로그램은 간편하고 직관적인 UI/UX를 제공하고 성과 관리에 필요한 주요 기능을 저렴한 가격에 이용할 수 있다는 장점을 가지고 있었지만, 1:N 커뮤니케이션이 아닌 1:1 커뮤니케이션만 가능하다는 단점과, 개인 역량 분석 지표가 너무 부족하다는 단점이 있다.

3. 개발 목표

애자일 방법론을 사용하는 기업을 대상으로 하는 성과 분석 서비스를 개발하여, 보다 객관적인 성과 분석을 제공하여 기업의 인적 의사결정에 도움이 될 수 있도록 하는 것이 목표이다.

4. 팀 역할 분담

최정훈(팀장)	- UI / UX 디자인 - 프론트엔드 개발
권순호	- 백엔드 개발 - DB 테이블 설계
박현준	- 백엔드 개발 - 개발 인프라 구축 (DevOps) - 클라우드 서비스

염종협	- UI / UX 디자인 - 로고 및 배경화면 디자인

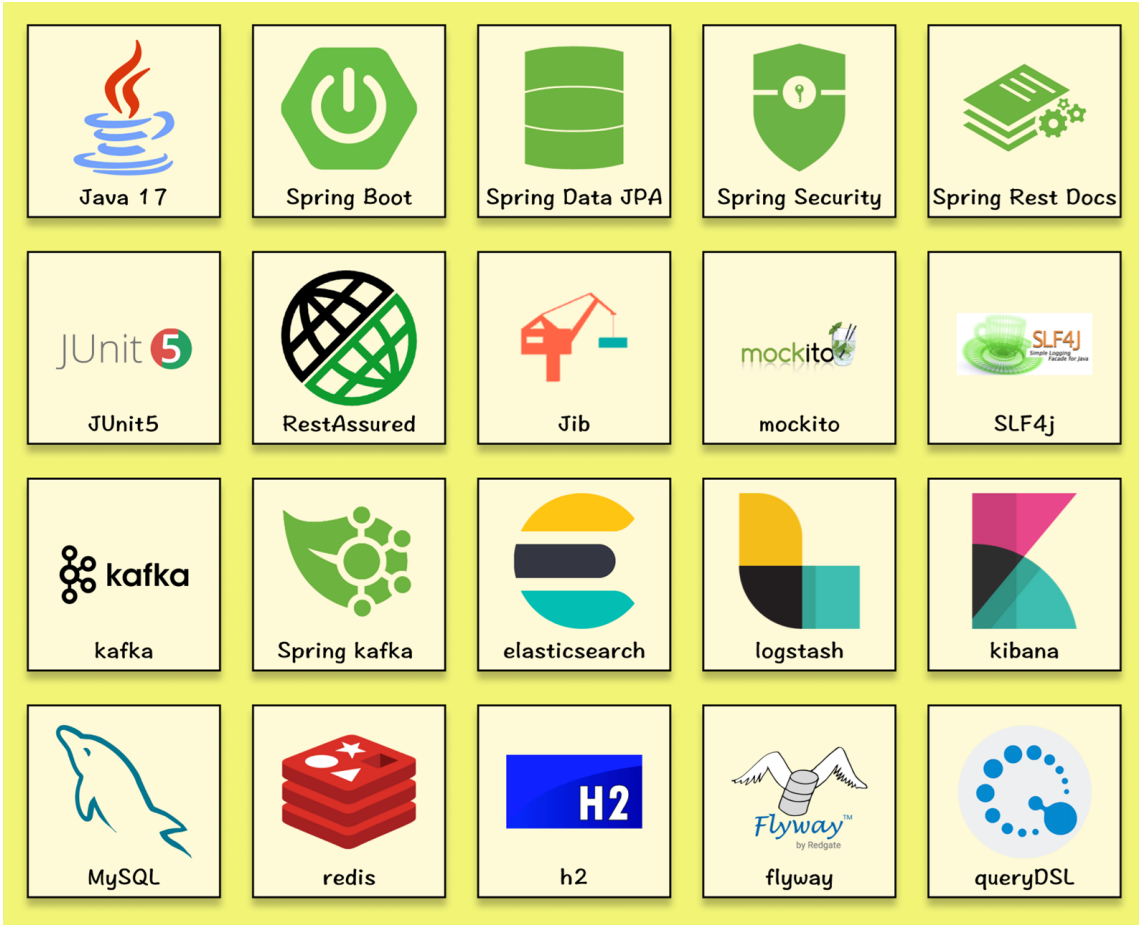
5. 개발 일정

활동	12	01	02	03	04	05	06	07	08	09
요구사항 및 분석	■									
시스템 상계 설계 및 시나리오 구성		■	■							
UI/UX 디자인		■	■	■						
프론트엔드 개발		■	■	■	■	■				
백엔드 개발		■	■	■	■	■				
시스템 테스트 및 개발						■	■	■		
최종 데모									■	
최종 보고서 작성										■

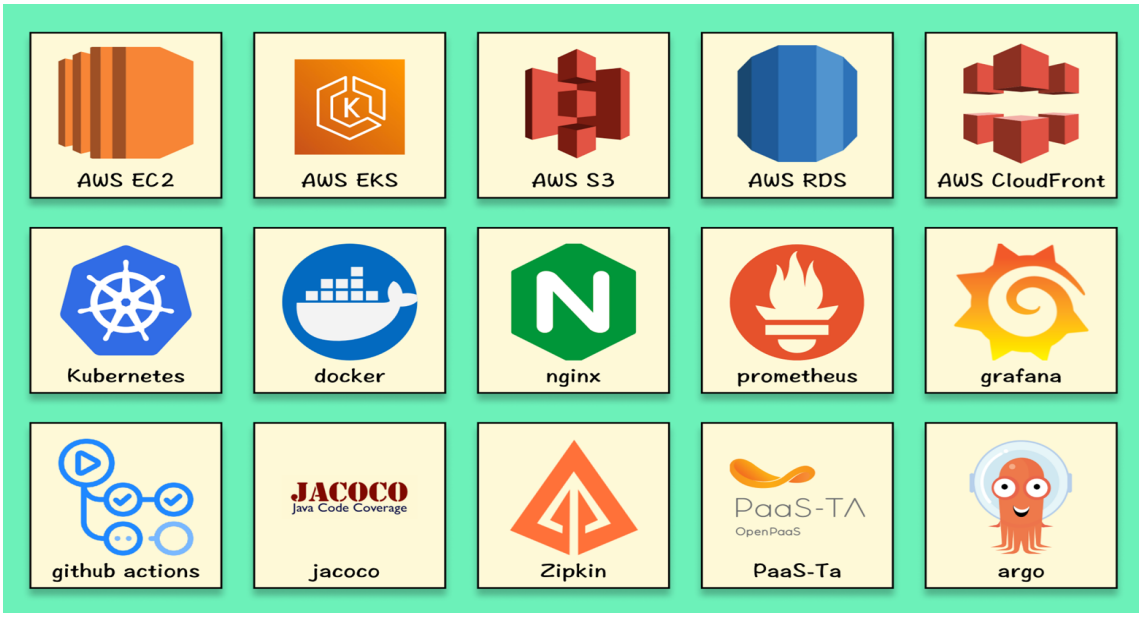
6. 개발 환경

기술 스택 - FRONTEND

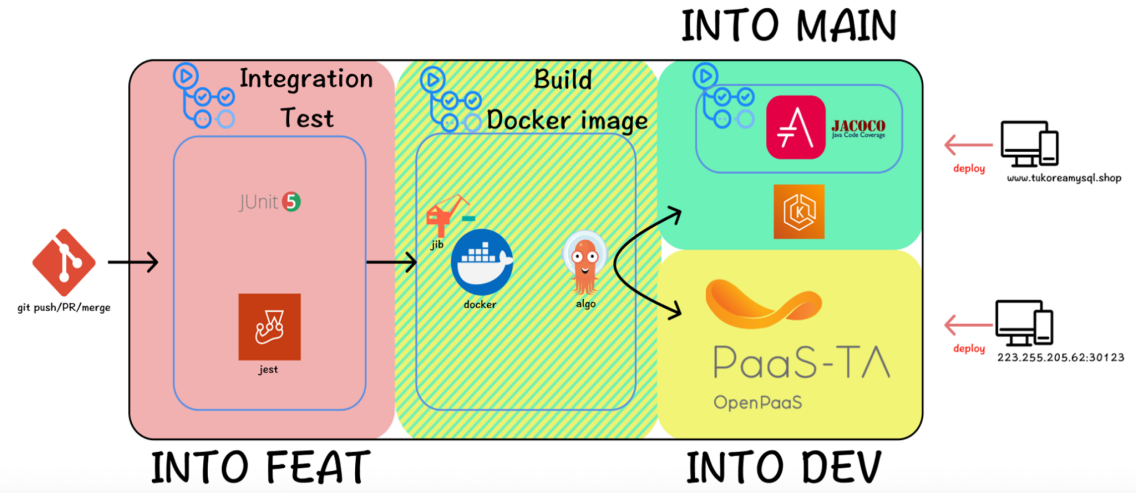
기술 스택 - BACKEND



기술 스택 - INFRA/DevOps



CI/CD FLOW



Ⅱ. 본론

1. 개발 내용

프로젝트는 여러 개의 스프린트(n주차)로 이루어져 있고, 스프린트는 여러 개의 태스크로 이루어짐을 전제로 한다.

프로젝트, 스프린트, 태스크 관련 개발 내용은 다음과 같다.

초기 팀장이 회원가입 후, 팀장 권한을 부여받고 팀을 생성한다(팀 생성은 팀장 권한만 가능). 팀 생성 후, 팀원 초대 혹은 팀원이 팀 참가 신청 가능하다. 팀장(프로젝트 관리자)이 프로젝트 생성 후, 스프린트 담당자 배정 및 스프린트 가치를 설정한다. 스프린트 담당자가 태스크의 가치를 설정하고 태스크 담당자 배정 혹은 프로젝트 참여자가 직접 태스크 선택한다. 태스크는 해결 or 미해결(True or False)로 나뉘고 task 해결 갯수에 따라 스프린트 진행도가 변한다.

이슈 관련 개발 내용 및 순서도는 다음과 같다.

1. 태스크 담당자가 이슈 생성
2. 답변자가 답변 달기
3. 필요시 질문자가 대댓글 달기
4. 완료되면 질문자가 이슈 close
5. 이슈에 관련된 사람들의 관련 역량 점수 업데이트

또한, 팀원마다 개인의 역량을 그래프로 시각화하여 볼 수 있고, 역량의 강점과 약점을 확인할 수 있다.

2. 문제 및 해결방안

2-1 기존 성과관리 시스템이 조직의 성장을 방해하는 이유

2013년 SHRM(Society for Human Resource Management)는 HR 전문가들에게 그들이 속한 회사의 성과관리 시스템이 얼마나 질적으로 만족스러운지를 물었다. 평균 이상이라 응답한 사람은 오직 23%에 불과했다. CEB(the Corporate Executive Board)에 따르면 조사대상 기업 매니저의 95%가 그들의 성과관리 제도를 불신하고, HR 헤드의 90%는 현재의 성과관리 시스템으로 본디 의도했던 구성원에 대한 실질적인 정보, 역량을 정확하게 파악할 수 없다고 여기고 있었다.

오늘날, 대다수 회사의 성과관리 제도는 매우 번거롭고, 복잡하고, 구성원이 제대로 이해하기도 어렵다. 조직, 인사관리 부서는 그러한 기형적인 성과관리 시스템을 행정적으로 통제하고 관리하는 것에 비생산적인 시간을 쏟아붓는다. 많은 회사는 관리자가 구성원을 중간등급으로 평가하려 하는 경향을 보이는 것(중심화 현상)을 파악했다. 어떤 기업의 임원은 이에 대해 이렇게 말한다. “3이외의 모든 등급은 관리자가 추가작업해야 하는 부담이 있다. 1~2를 주면 그 이유를 명확히 파악해 다른 구성원에게 정당화해야 하며, 4~5 등 낮은 등급을 주면 당사자를 납득시키고, 성과 개선 계획을 별도로 작성해야 하기에 부담이 된다.”

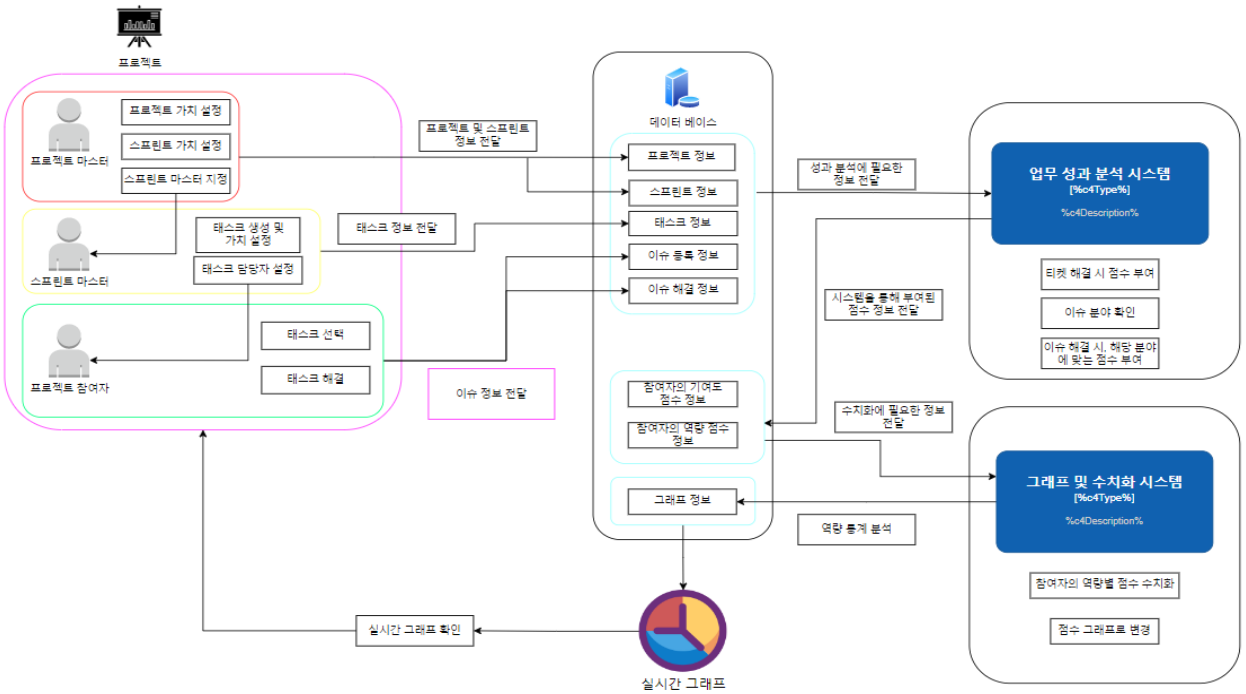
일련의 중심화 경향을 해소하기 위해 몇몇 선도 기업은 관리자에게 등급의 강제할당을 의무화하는 정책을 펼쳤다. 문제는, 이 시스템 아래에서 매니저와 구성원은 모두 지치고, 오히려 업무 동기를 상실하고 있다는 것이다. 성과를 독려하기 위해 고안된 방식이 사실상 그 반대의 효과를 자극하고 있는 것이다. 해외에서는 이러한 문제점을 인식하고 성과 관리 시스템을 전면 재설계하여 연 200만의 시간 낭비를 줄였다는 사례도 있다.

2-2 애자일 성과 분석 시스템의 장점

위에서 말한 문제점들은 애자일 성과 분석 시스템으로 해결할 수 있다. 애자일은 프로젝트 관리 및 소프트웨어 개발에 대한 반복적인 접근 방식으로, 고객에게 가치를 더 빠르고 덜 복잡하게 제공할 수 있다. 애자일 방식은 모든 것에 집중하는 대신 더 작은 공략 가능한 단위로 작업을 제공한다. 이를 통해, 고객의 요구에 좀 더 반응적으로 대응할 수 있게 된다. 또한, 정량적이고 세밀한 평가를 통해 인적 의사결정에 반영할 수도 있고, 팀간의 협력 문화를 배양할 수 있게 된다.

3. 시험 시나리오

시스템 구성도

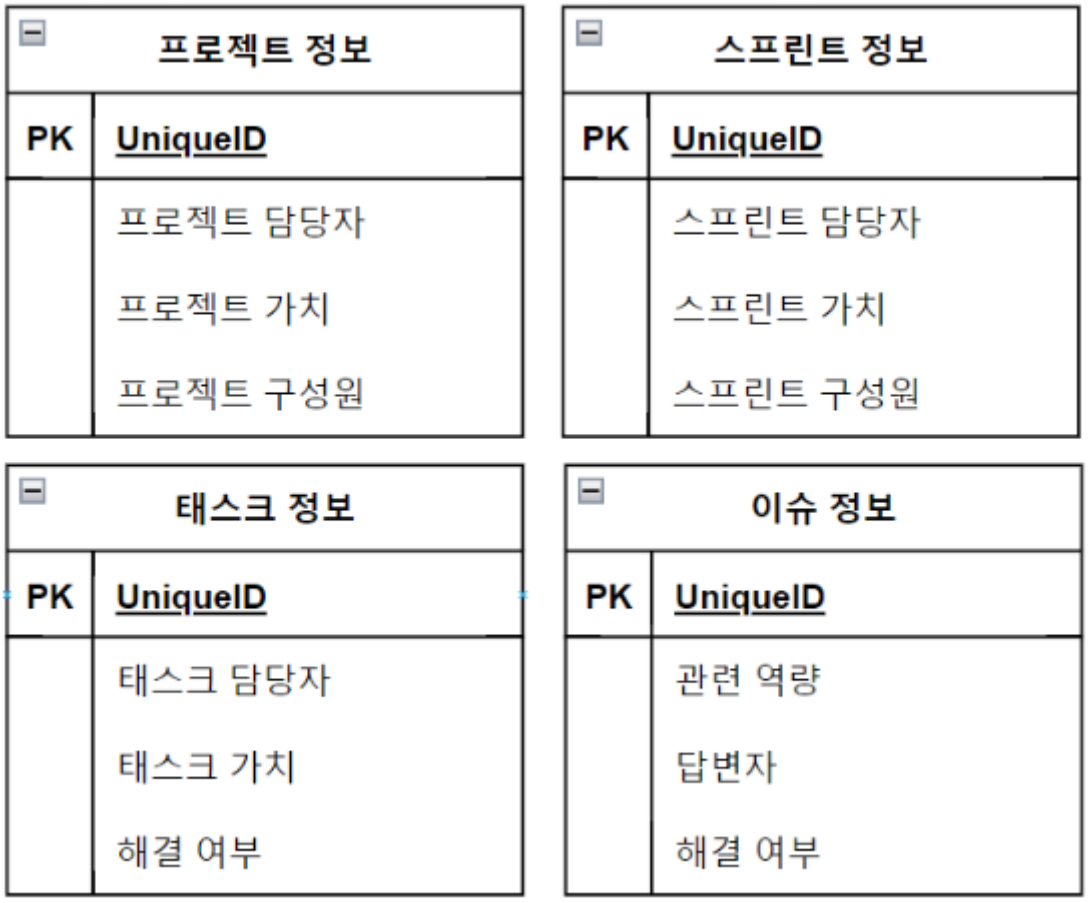


프로젝트에서는 프로젝트 마스터, 스프린트 마스터, 프로젝트 참여자 등으로 구성되어 있다. 프로젝트 마스터는 프로젝트 생성 및 프로젝트와 각 스프린트마다의 가치 점수를 부여한다. 또한, 스프린트 마스터를 설정할 수 있다. 스프린트 마스터는 스프린트 마다의 태스크를 생성하고, 생성된 태스크를 직접 프로젝트 참여자에게 할당할 수 있다. 또한, 태스크의 가치 점수를 부여한다. 프로젝트 참여자는 할당된 태스크를 수행하거나 혹은 직접 태스크를 선택할 수 있고, 이슈 발생시 조직 구성원들간의 논의를 위해 이슈 등록을 할 수 있다. 이 모든 정보들은 데이터베이스에 저장된다.

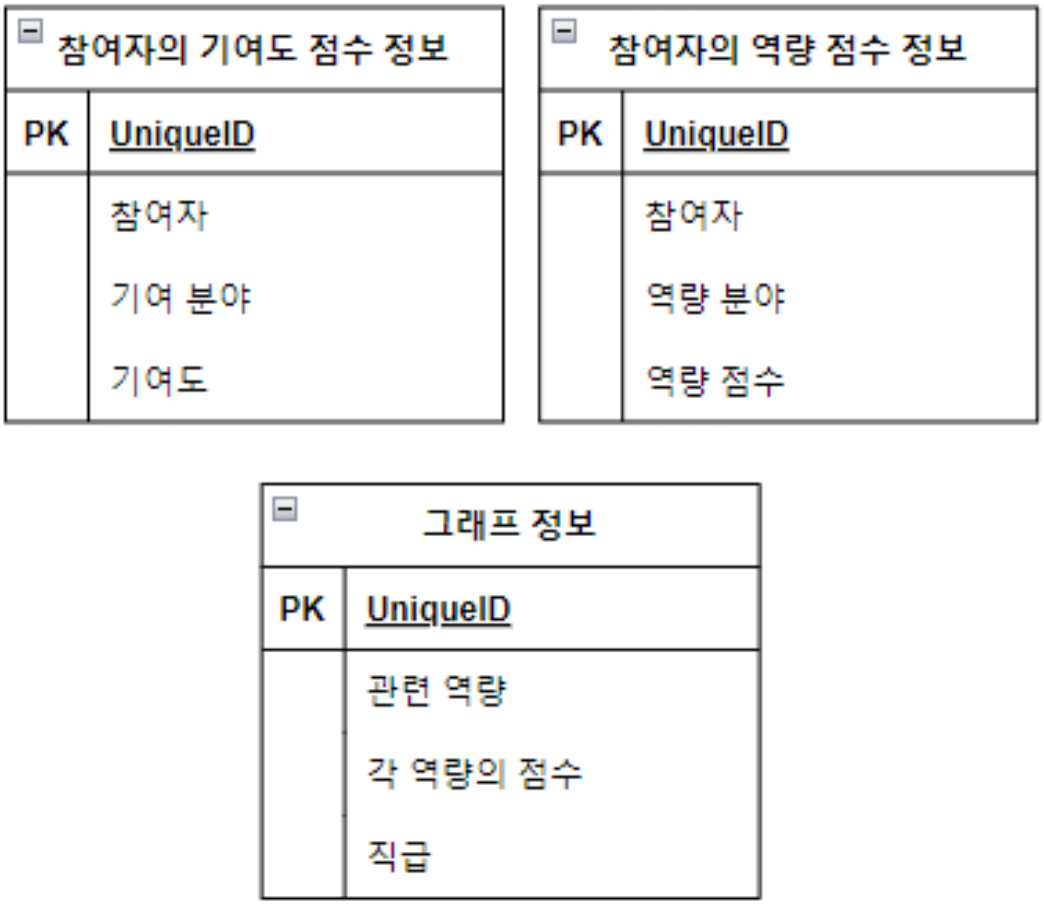
데이터베이스는 프로젝트, 스프린트, 태스크들의 정보를 저장한다. 또한, 성과 분석에 필요한 정보들을 업무 성과 분석 시스템에 전달하면, 이슈에 맞는 태그 점수를 최신화하고 각 프로젝트 참여자에 대한 정보를 갱신한다.
이 정보는 다시 데이터베이스에 저장된다. 그 정보를 활용하여 다시 그래프 및 수치화 시스템에 정보를 전달하여 프로젝트 참여자의 역량별 점수를 수치화하여 점수 그래프로 변경한다. 이 정보를 조직 구성원들이 필요시, 언제든지 볼 수 있도록 데이터베이스에 저장한다.

4. 상세 설계

DB테이블은 다음과 같이 설계한다.

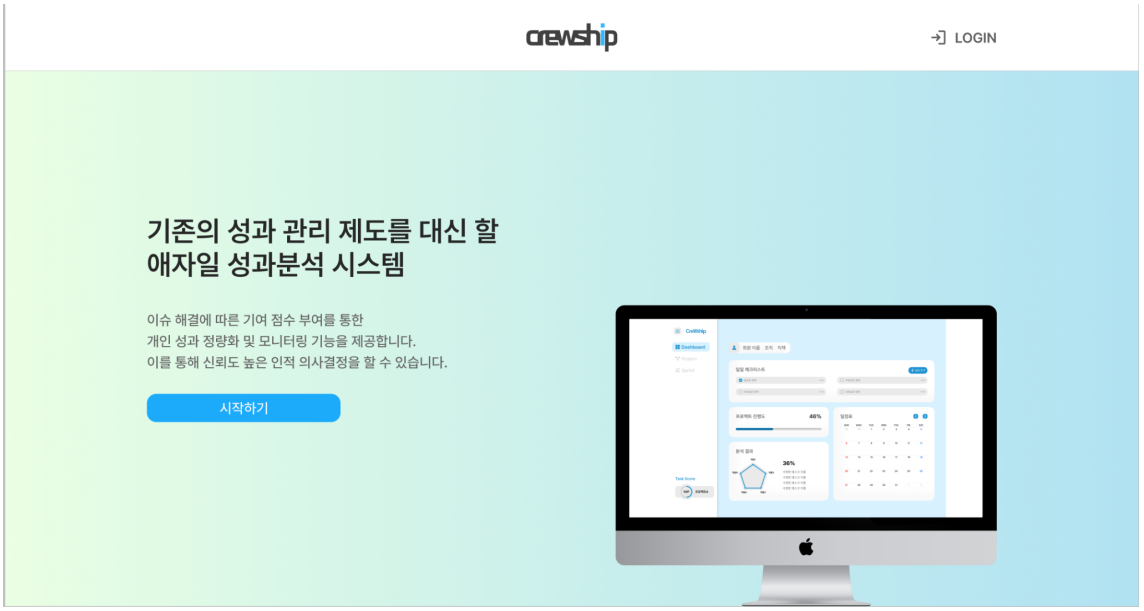


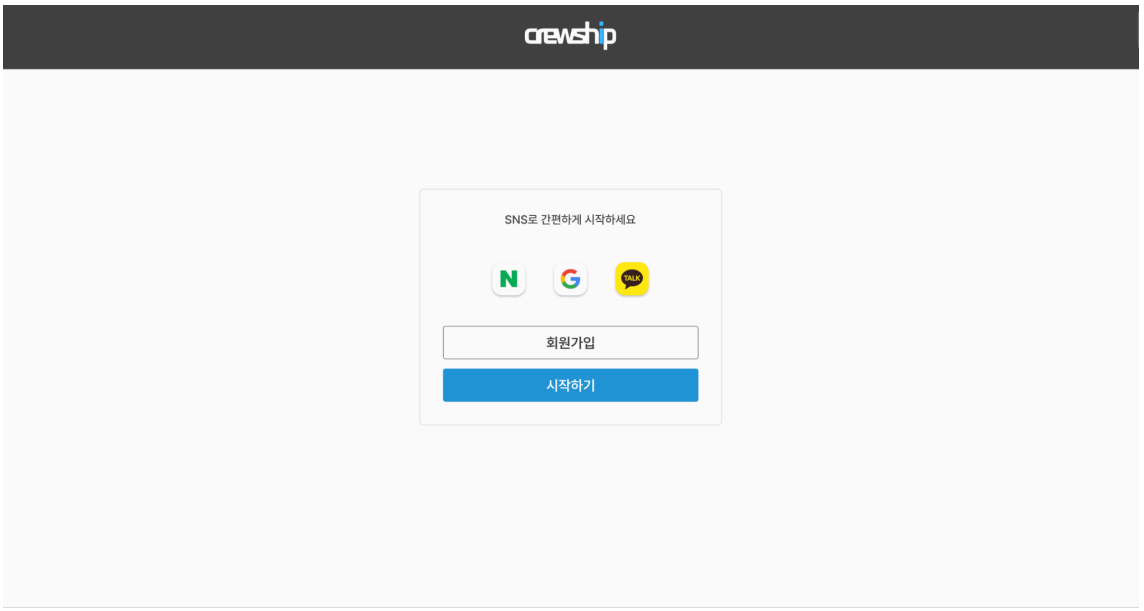
DB테이블은 크게 프로젝트에 대한 정보와 구성원들의 정보로 나뉘어진다. 프로젝트 관련 정보는 다음과 같이 프로젝트와 스프린트, 태스크의 정보가 있다. 프로젝트와 스프린트는 각각의 담당자와 가치 그리고 구성원을 저장한다. 태스크는 구성원이 아닌 해결 여부를 저장한다. 또한, 이슈에는 관련 역량과 답변자, 그리고 해결 여부를 저장한다. 이슈가 해결이 되면 답변자의 관련 역량 점수가 최신화된다.



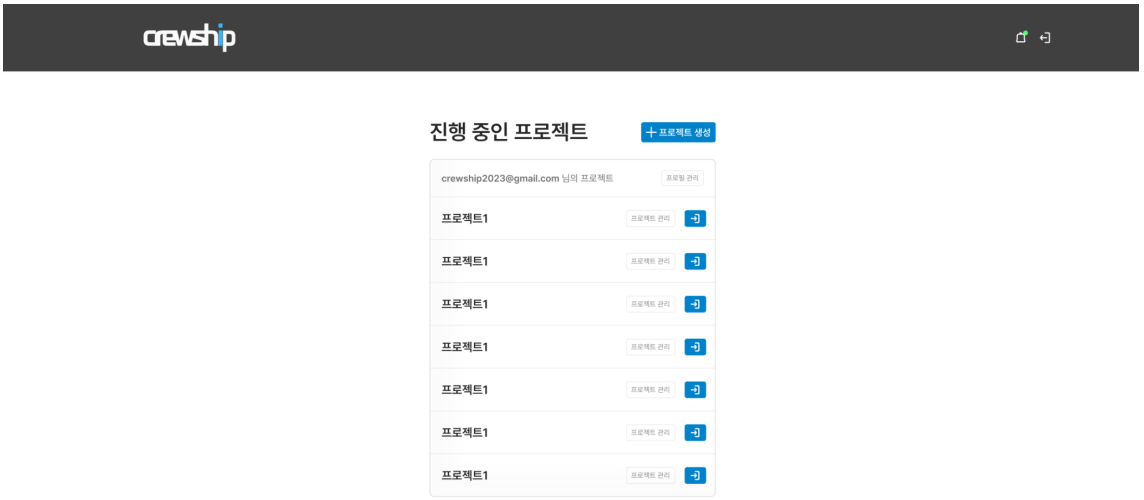
다음은 구성원들에 대한 정보이다. 구성원들에 대한 정보에는 참여자의 기여도 점수와 역량 점수 정보, 그리고 그래프 정보가 있다. 참여자의 기여도 점수 정보에는 참여자와 기여 분야 그리고 기여도로 구성되어 있다. 참여자의 역량 점수 정보는 비슷하게 참여자와 역량 분야, 역량 점수로 구성되어 있다. 그래프 정보는 위 두 정보를 사용하여 나타낼 수 있는데, 관련 역량과 그 역량에 맞는 점수와 직급으로 구성되어 있다.

5. Prototype 구현

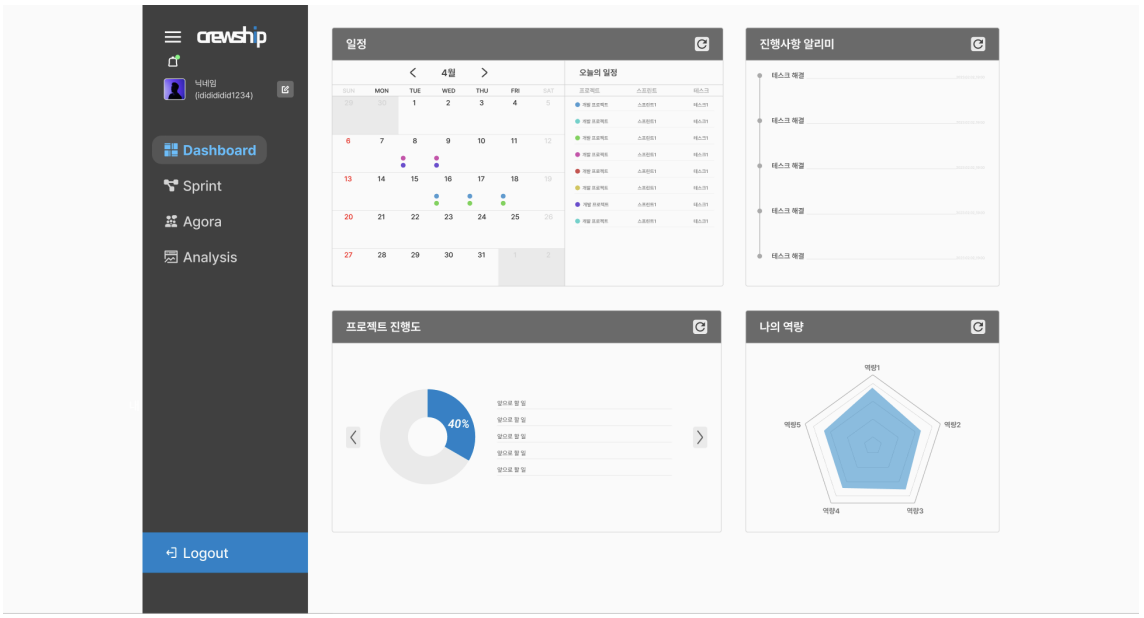




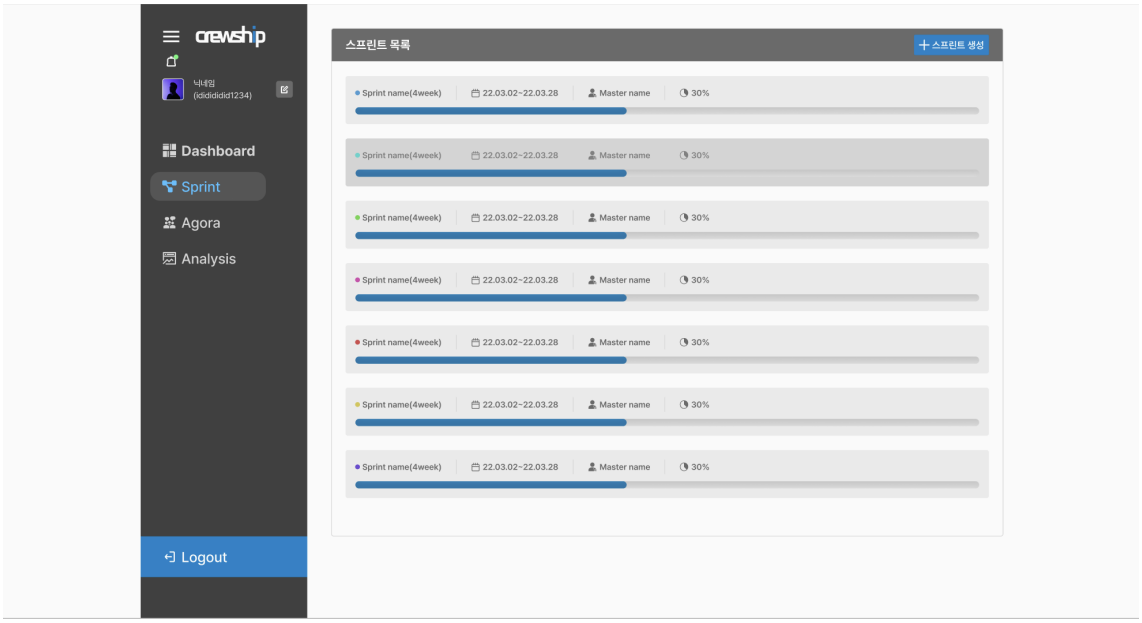
- 별도의 회원가입 없이 소셜 로그인을 통해 로그인 진행

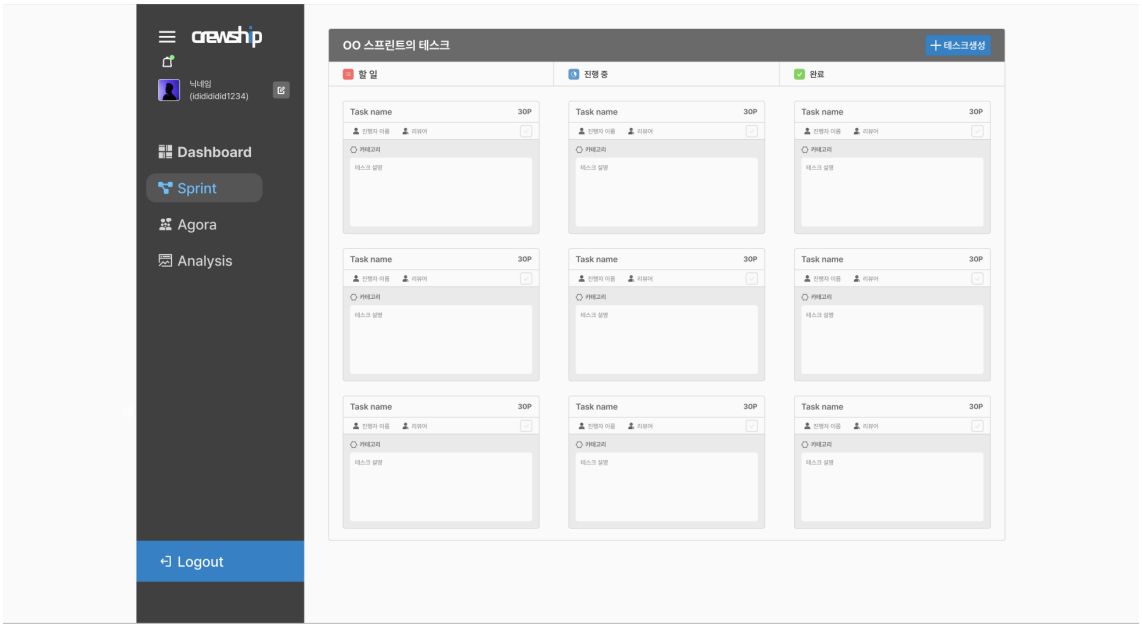


- 로그인 후 본인이 참여중인 프로젝트 목록 확인
- 프로젝트 생성 및 초대 기능 구현

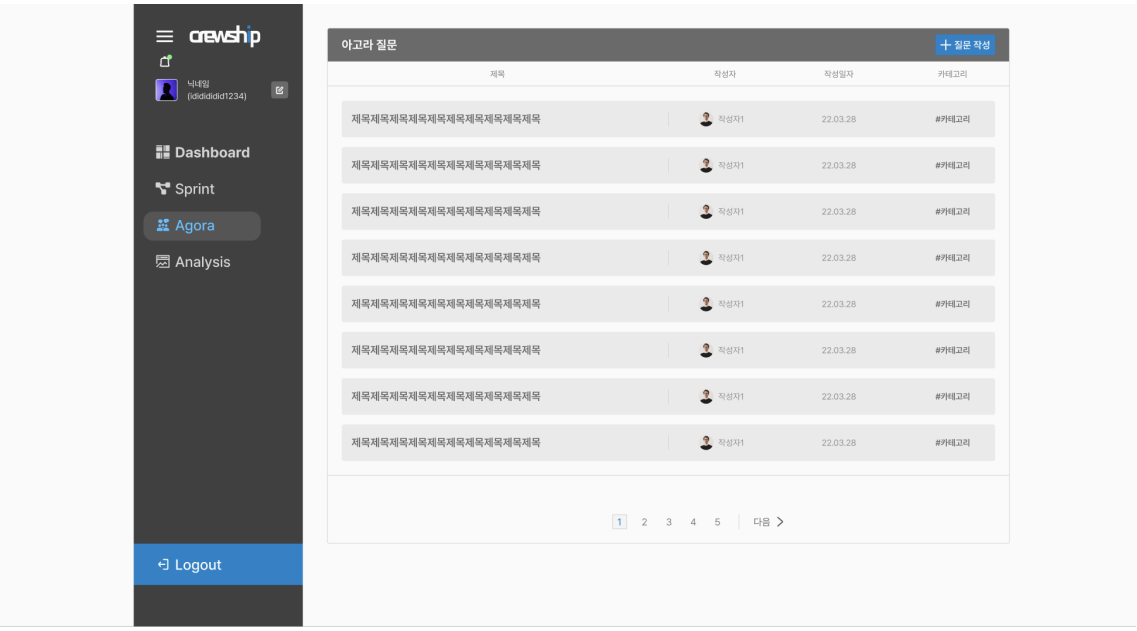


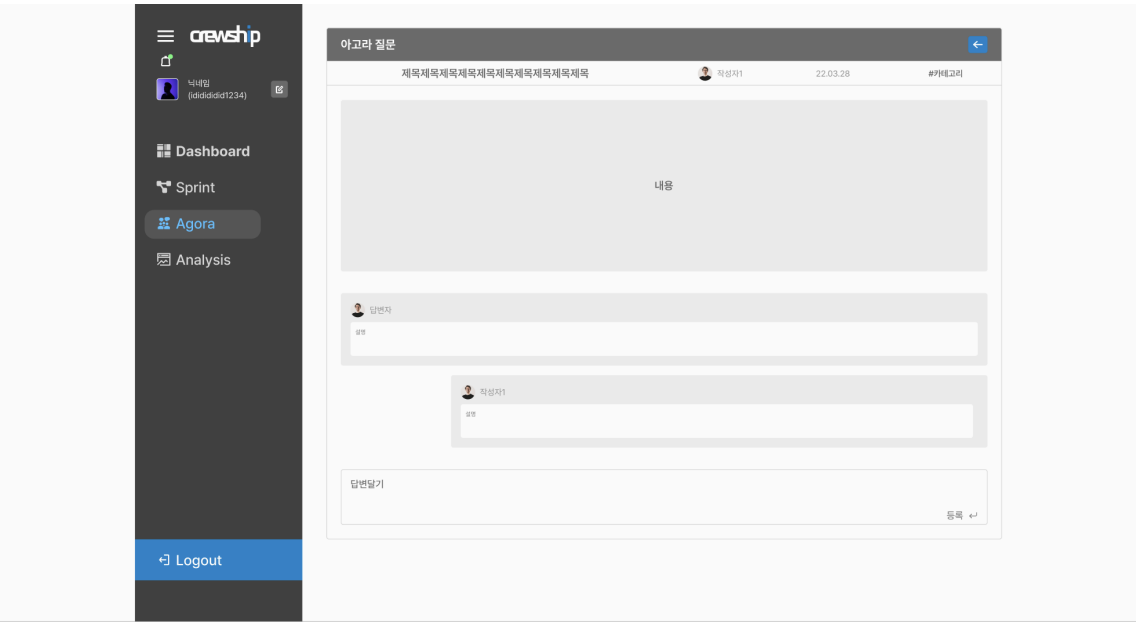
- 메인 화면 (대시 보드)
- 기본적인 일정 관리, 프로젝트 진행도, 본인의 참여 역량(그래프)





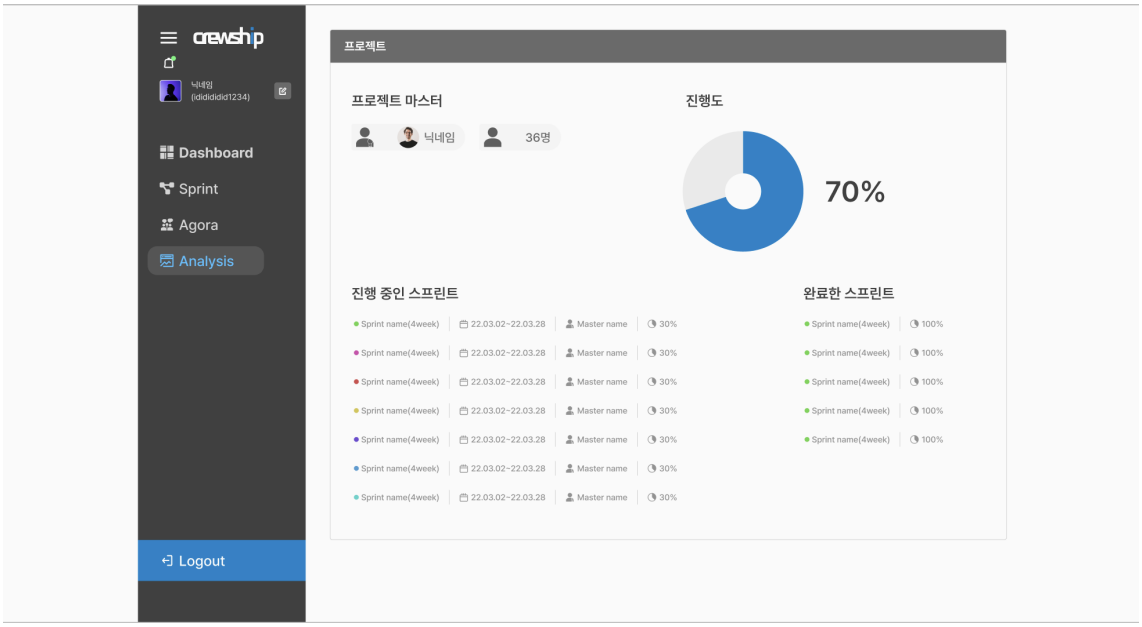
- 스프린트의 경우 프로젝트 기간과 스프린트 주기에 맞춰 자동으로 생성
- 스프린트 내에서 필요한 태스크 생성
- 태스크는 진행도에 따라 3가지로 분류하여 관리





- 프로젝트 진행과정에서 필요한 질문을 위한 Agora 페이지

- StackOverflow의 질의응답 기능에서 착안하여 진행한 페이지로, 질문에 대해 자유롭게 댓글을 달 수 있고, 댓글 활동은 기여 점수에 포함됨



- 최종 분석 페이지
- 본인의 진행 사항과 역량을 확인 가능
- 5가지 역량에 맞춰 본인이 어느정도 기여했는지 비교 분석 가능
- 다른 프로젝트 참여자와의 그래프 비교를 통해 본인의 역량 확인 가능

6. 시험/ 테스트 결과

번호	테스트 항목	테스트 내용	결과
1	로그인	소셜 로그인을 통한 로그인이 잘 진행되는가?	O
2 - 1	프로젝트 생성	프로젝트 생성이 구현되었는가?	O
2 - 2		프로젝트에 지원하는 기능이 구현되었는가?	O
2 - 3		프로젝트에 팀원 초대가 진행되는가?	O
3 - 1	메인 화면 (대시 보드)	일정 관리를 위한 달력과 알리미(리스트)기능이 구현되었는가?	O
3 - 2		프로젝트 진행도와 역량 그래프가 구현되었는가?	O
4 - 1	Agora (질의 응답 게시판)	게시판 기능이 구현되었는가?	O
4 - 2		댓글 기능이 구현되었는가?	X
5 - 1	스프린트 / 태스크	프로젝트 기간과 스프린트 주기에 맞춰 스프린트가 자동으로 생성되었는가?	O
5 - 2		태스크 생성이 구현되었는가?	O
5 - 3		태스크는 진행상황에 맞춰 3단계로 관리가 되는가?	△
6 - 1	성과 분석	5가지 역량이 제대로 반영된 그래프가 구현되었는가?	△
6 - 2		프로젝트 진행도가 표시된 그래프가 구현되었는가?	O
6 - 3		다른 팀원들과 비교분석 가능한 PivotTable이 구현되었는가?	X

7. Coding & DEMO

가. 주요 코드

(1) Member

```
public class AuthService {  Complexity is 4 Everything is cool!
    1 usage
    private final MemberQueryRepository memberQueryRepository;

    5 usages  Hyeonjun Park
    public UUID getLoginUserId() {
        return ((LoginUser) (SecurityContextHolder.getContext().getAuthentication().getPrincipal()))
                .getMemberId();
    }

    4 usages  Hyeonjun Park +1
    public Member getLoginUserEntity() {  Complexity is 3 Everything is cool!
        return memberQueryRepository
                .findById(getLoginUserId())
                .orElseThrow(EntityNotFoundException::new);
    }
}
```

UUID를 반환하는 함수다.

SecurityContextHolder.getContext().getAuthentication().getPrincipal()을 사용하여 현재 인증된 사용자의 principal 객체를 LoginUser 타입으로 얻는다. 이에 대해 getMemberId() 메소드를 호출하여 로그인 사용자의 멤버 ID를 나타내는 UUID를 가져와서 반환한다.

(2) Staff

```
2 usages  Hyeonjun Park
public List<StaffResponse> findAllApplyByTeamId(final Long teamId) {
    final List<Staff> staffs = staffQueryRepository.findAllByTeamIdApplyFilter(teamId);
    return mapToList(staffs);
}
```

특정 팀 ID에 대해 해당 팀에 속한 staff들의 목록을 StaffResponse 객체의 목록으로 변환하여 반환하는 메소드다.

```
2 usages  Hyeonjun Park
public List<StaffResponse> findAllInvite() {
    final UUID loginUserId = authService.getLoginUserId();
    final List<Staff> staffs = staffQueryRepository.findAllByMemberIdInviteFilter(loginUserId);
    return mapToList(staffs);
}
```

findAllInvite는 현재 로그인된 사용자에게 전송된 초대에 대한 staff 목록을 가져오는 메소드다.

```
2 usages  Hyeonjun Park
public List<StaffResponse> findAllTeamStaff(final Long teamId) {
    final List<Staff> staffs = staffQueryRepository.findAllActiveStaffByTeamId(teamId);
    return mapToList(staffs);
}
```

findAllTeamStaff는 특정 팀에 속한 모든 staff의 목록을 가져오는 메소드다.

```
2 usages  Hyeonjun Park
public List<StaffResponse> findAllMyTeam() {
    final UUID loginUserId = authService.getLoginUserId();
    List<Staff> staffs = staffQueryRepository.findAllActiveStaffByMemberId(loginUserId);
    return mapToList(staffs);
}
```

findAllMyTeam는 현재 로그인된 사용자가 속한 모든 팀의 스태프 목록을 가져오는 메소드다.

(3) Team

```
Hyeonjun Park
public IdResponse<Long> create(final TeamRequest dto) {
    final Member loginUser = authService.getLoginUserEntity();
    final Team team = teamRepository.save(teamMapper.toEntity(dto, loginUser));
    staffService.createProjectLeaderStaff(team);
    return new IdResponse<>(team.getId());
}
```

TeamRequest DTO를 기반으로 새로운 team을 생성하고, 생성된 team과 관련된 추가 작업을 수행하는 메소드다.

(4) Project

```
public ProjectResponse create(final ProjectRequest dto) {
    final Staff manager = staffLoader.getEntity(dto.getManagerId());
    final Team team = teamLoader.getEntity(dto.getTeamId());
    final Project project = projectRepository.save(projectMapper.toEntity(dto, team, manager));
    sprintService.createInitial(project);
    return projectMapper.toResponse(project);
}
```

ProjectRequest DTO를 기반으로 새로운 Project를 생성하고, 관련된 추가 작업을 수행 후, ProjectResponse 객체로 변환하여 반환하는 메소드다.

```
public List<ProjectResponse> findAllByTeamId(final Long teamId) {  Complexity is 3 Everything is cool!
    return projectQueryRepository.findAllByTeamId(teamId).stream()  Stream<Project>
            .map(projectMapper::toResponse)  Stream<ProjectResponse>
            .toList();
}
```

teamId를 통해 해당 팀에 소속한 모든 Project 목록들을 보여주는 메소드다.

(5) Sprint

```java
public void createInitial(final Project project) {  Complexity is 7 It's time to do something...
    if (project.getEndAt() == null) {
        return;
    }
    int round = 1;
    LocalDate datePivot = LocalDate.from(project.getStartAt());
    final LocalDate endDate = LocalDate.from(project.getEndAt());
    final Integer sprintDays = project.getSprintDays();
    while (!getNextDays(datePivot, sprintDays).isAfter(endDate)) {
        sprintRepository.save(
                new Sprint(
                        project,
                        round,
                        datePivot,
                        getNextDays(datePivot,  sprintDays: sprintDays - 1),
                        project.getManager()));
        datePivot = getNextDays(datePivot, sprintDays);
        round++;
    }
    if (!datePivot.isEqual(endDate)) {
        sprintRepository.save(
                new Sprint(project, round, datePivot, endDate, project.getManager()));
    }
}
```

프로젝트의 시작일과 종료일을 기반으로 초기 스프린트를 생성한다. 스프린트의 시작일과 종료일은 스프린트 일수(sprintDays)를 기준으로 계산한다. 스프린트를 생성하고 저장한 후, 마지막 스프린트의 종료일이 프로젝트의 종료일과 일치하지 않을 경우 추가적인 스프린트를 생성하여 저장한다.

```java
private LocalDate getNextDays(final LocalDate datePivot, final Integer sprintDays) {
    return datePivot.plusDays(sprintDays);
}
```

주어진 날짜(datePivot)로부터 일정 일 수(sprintDays)를 더한 다음 날짜를 반환하는 getNextDays 메소드다.

(6) Task

```
public List<TaskSummary> findAllByCondition(final TaskSearchCondition condition) {
    return taskQueryRepository.findAllByCondition(condition).stream() Stream<Task>
            .map(taskMapper::toSummary) Stream<TaskSummary>
            .toList();
}
```

주어진 조건에 해당하는 모든 task의 요약 정보를 가져오는 메소드다.

```
public TaskResponse findById(final Long id) {
    return taskMapper.toResponse(getEntity(id));
}
```

주어진 ID에 해당하는 task의 상세 정보를 가져오는 메소드다.

```
public List<TaskSummary> findAllMyTask(final MyTaskCondition condition) {
    final Staff activeStaff = staffLoader.getActiveStaff(condition.getTeamId());
    return findAllByCondition(condition.toSearchCondition(activeStaff.getId()));
}
```

현재 사용자의 task를 조회하는 메소드다.

(7) question

```
2 usages  Hyeonjun Park +1
public List<QuestionResponse> findAllByWriterId(final Long writerId) {
    final List<Question> questions = questionQueryRepository.findAllByWriterId(writerId);
    return mapToList(questions);
}
```

주어진 작성자 ID에 해당하는 모든 question의 목록을 가져오는 메소드다.

```
Hyeonjun Park +1
public QuestionResponse findById(final Long Id) {
    final Question question = getEntity(Id);
    return questionMapper.toResponse(question);
}
```

주어진 questionID에 해당하는 question의 상세 정보를 가져오는 메소드다.

```
2 usages  Soon Ho Kwon
public List<QuestionResponse> findAllByTeamId(final Long teamId) {
    final List<Question> questions = questionQueryRepository.findAllByTeamId(teamId);
    return mapToList(questions);
}
```

주어진 teamID에 해당하는 모든 question들의 목록을 반환하는 메소드다.

(8) answer

```
2 usages  Hyeonjun Park +1
public List<AnswerResponse> findAllByQuestionId(final Long memberId) {
    final List<Answer> answers = answerQueryRepository.findAllByQuestionId(memberId);
    return mapToList(answers);
}
```

해당 memberId를 가진 member가 작성한 모든 question 목록을 보여주는 메소드다.

```
Hyeonjun Park +1
public AnswerResponse findById(final Long id) {
    return answerMapper.toResponse(getEntity(id));
}
```

answerId를 통해 해당 answer의 상세 정보를 조회할 수 있는 메소드다.

(9) answerComment

```
1 usage  Hyeonjun Park +2
public List<AnswerCommentResponse> findAllByAnswerId(final Long answerId) {
    final List<AnswerComment> answerComments =
            answerCommentQueryRepository.findAllByAnswerId(answerId);
    return mapToList(answerComments);
}
```

answerId에 해당하는 answer에 달려있는 모든 answerComment들의 목록을 반환하는 메소드다.

```
Hyeonjun Park +1
public AnswerCommentResponse findById(final Long id) {
    return answerCommentMapper.toResponse(getEntity(id));
}
```

answerCommentId에 해당하는 answerComment의 상세 정보를 조회하는 메소드다.

(10) Ability

```
2 usages  Hyeonjun Park
public List<AbilityResponse> findAllByStaffId(final Long staffId) {  Complexity is 3 Everything is cool!
    final List<Ability> abilities = abilityQueryRepository.findAllByReceiverId(staffId);
    return abilities.stream().map(abilityMapper::toResponse).toList();
}
```

주어진 staffId에 해당하는 모든 ability 목록을 가져오는 메소드다.

```
2 usages  Hyeonjun Park
public AbilitySummary findSummaryByStaffId(final Long staffId) {
    final List<Ability> abilities = abilityQueryRepository.findAllByReceiverId(staffId);
    final Staff receiver = staffLoader.getEntity(staffId);
    final Map<AbilityCategory, Long> points = generatePoints(abilities);
    return abilityMapper.toSummary(receiver, points);
}
```

주어진 staffId에 해당하는 능력 요약 정보를 가져오는 메소드다.

```
2 usages  Hyeonjun Park +1
private Map<AbilityCategory, Long> generatePoints(final List<Ability> abilities) {  Complexity is 5 Everything is cool!
    final Map<AbilityCategory, Long> points = new EnumMap<>(AbilityCategory.class);
    pointMapInitialize(points);
    abilities.forEach(
            ability -> points.compute(ability.getCategory(), (k, v) -> v + ability.getPoint()));
    return points;
}
```

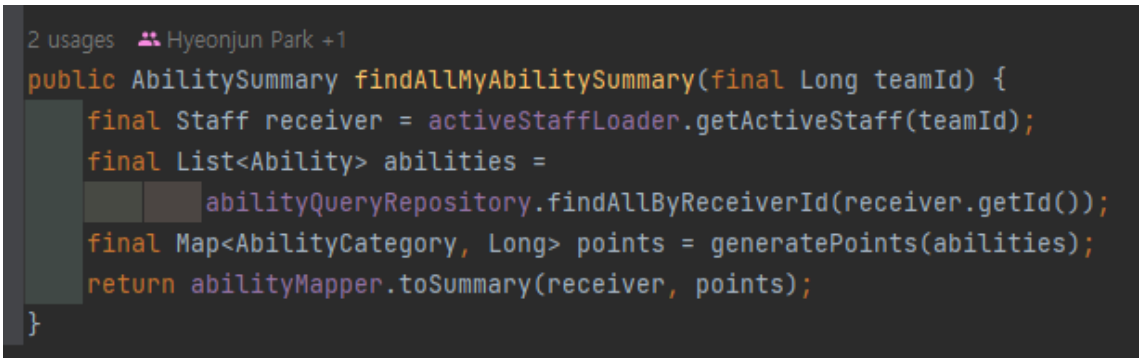
Ability 목록을 기반으로 Ability별 점수를 생성하는 메소드다.

```
1 usage  Hyeonjun Park
private void pointMapInitialize(Map<AbilityCategory, Long> points) {
    Arrays.stream(AbilityCategory.values()).forEach(category -> points.put(category, 0L));
}
```

Ability별 점수를 0L로 초기화하는 메소드다.

```
2 usages  Hyeonjun Park
public List<AbilityResponse> findAllMyAbility(final Long teamId) {  Complexity is 3 Everything is cool!
    final Long receiverId = activeStaffLoader.getActiveStaff(teamId).getId();
    final List<Ability> abilities = abilityQueryRepository.findAllByReceiverId(receiverId);
    return abilities.stream().map(abilityMapper::toResponse).toList();
}
```

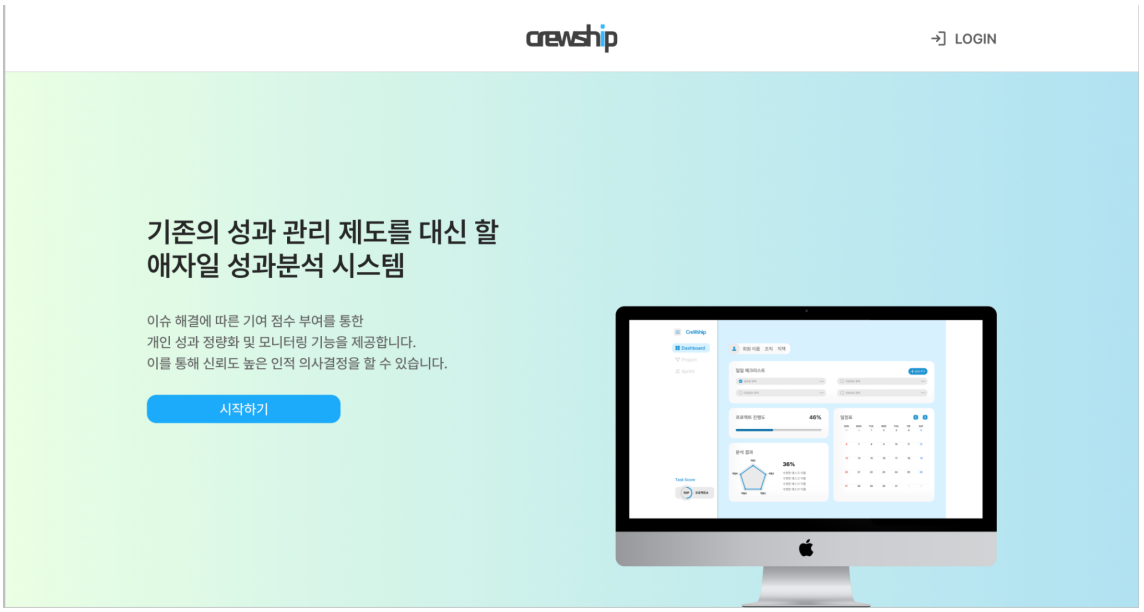
주어진 teamId에 해당하는 Ability의 목록을 가져오는 메소드다.



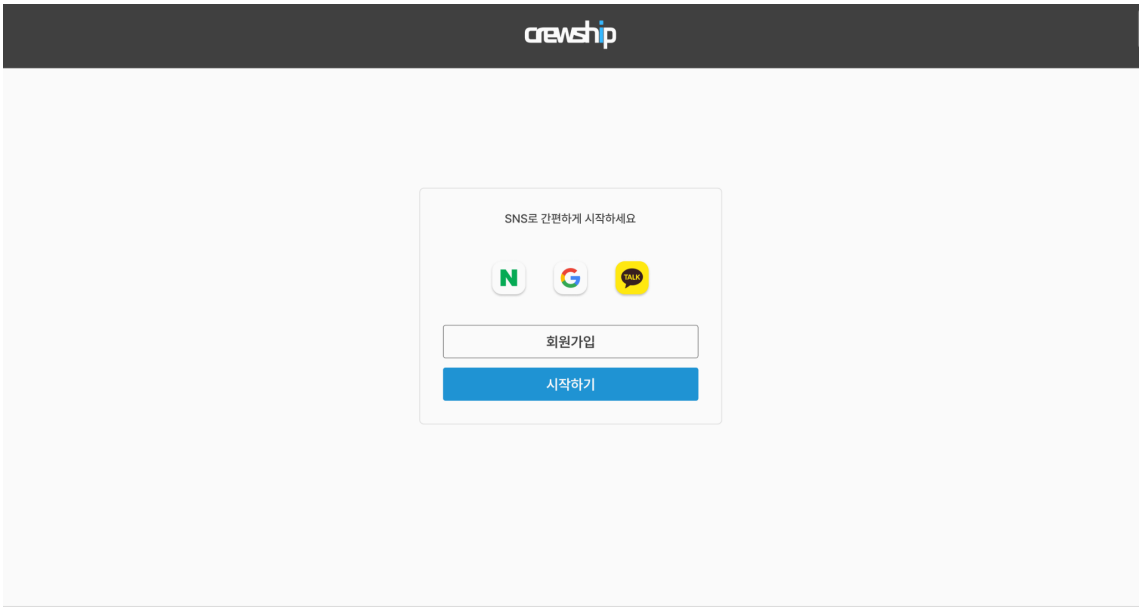
주어진 teamId에 해당하는 내 Ability의 요약 정보를 가져오는 메소드다.

나. DEMO 시나리오

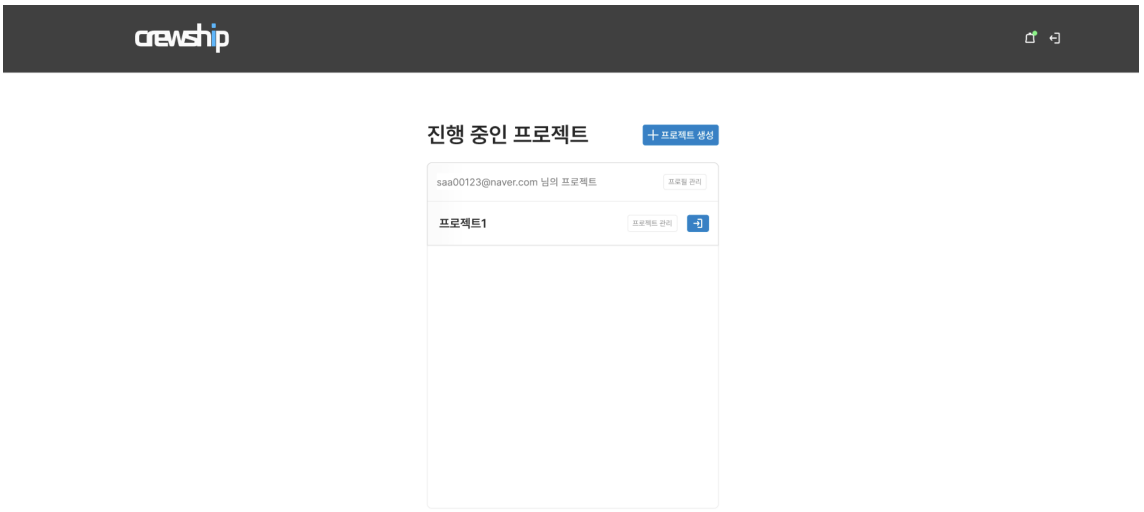
초기 화면



소셜 로그인

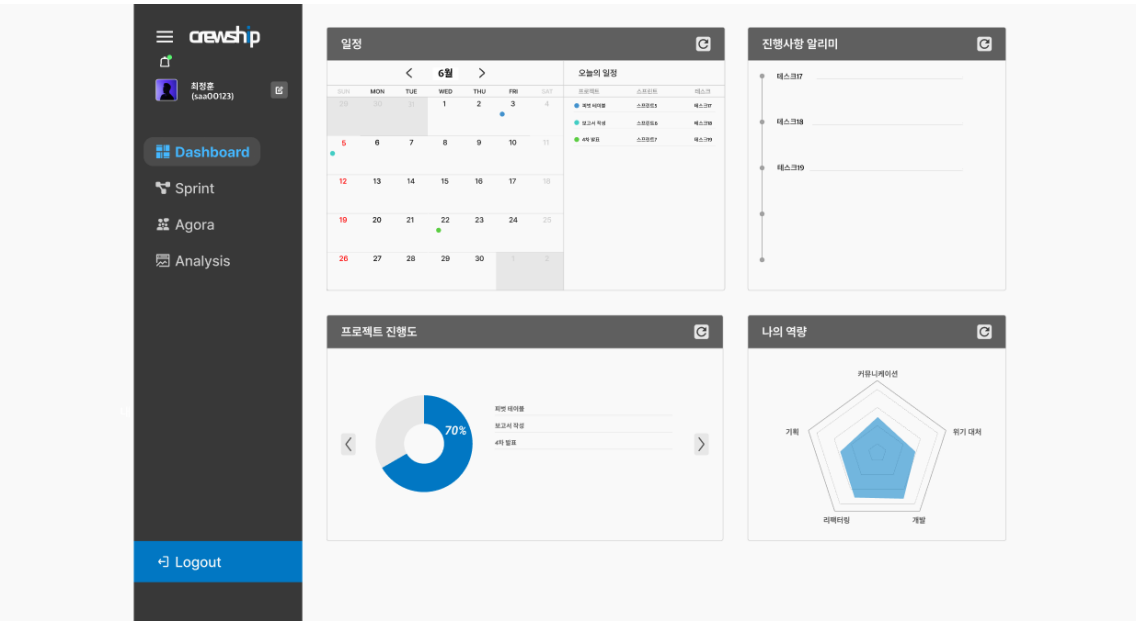


프로젝트 리스트 화면



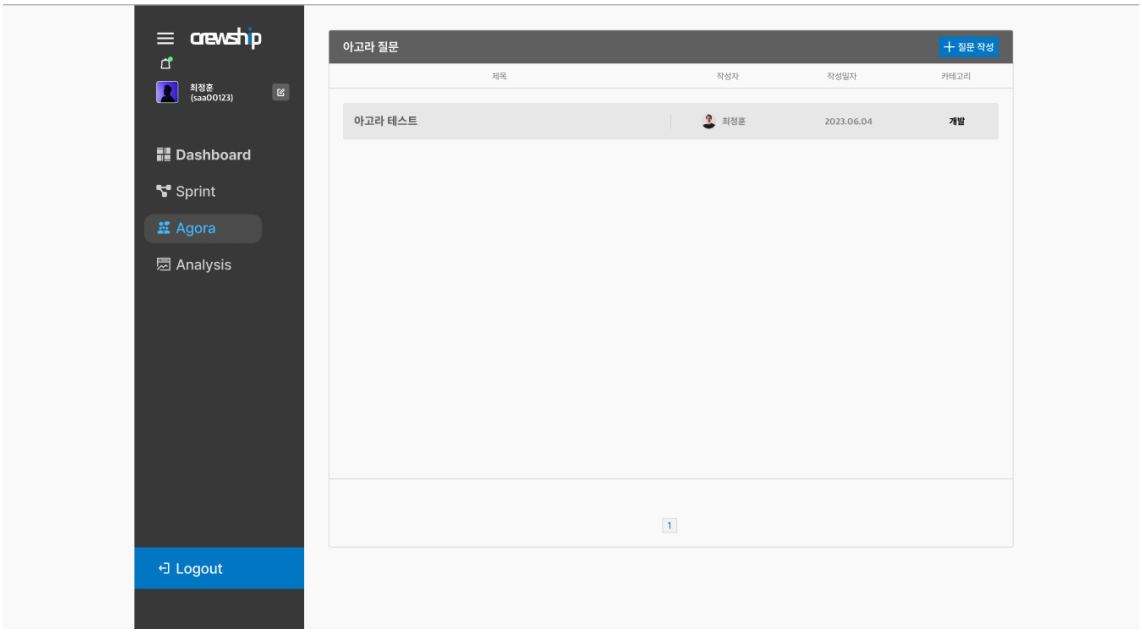
- 네이버 계정으로 로그인 (saa00123@naver.com)
- 애자일 성과 분석이라는 프로젝트 방향성에 맞게, 현재 종합설계 프로젝트 내용을 성과 분석하는 예시 프로젝트로 데모를 진행

메인 화면 (대시 보드)



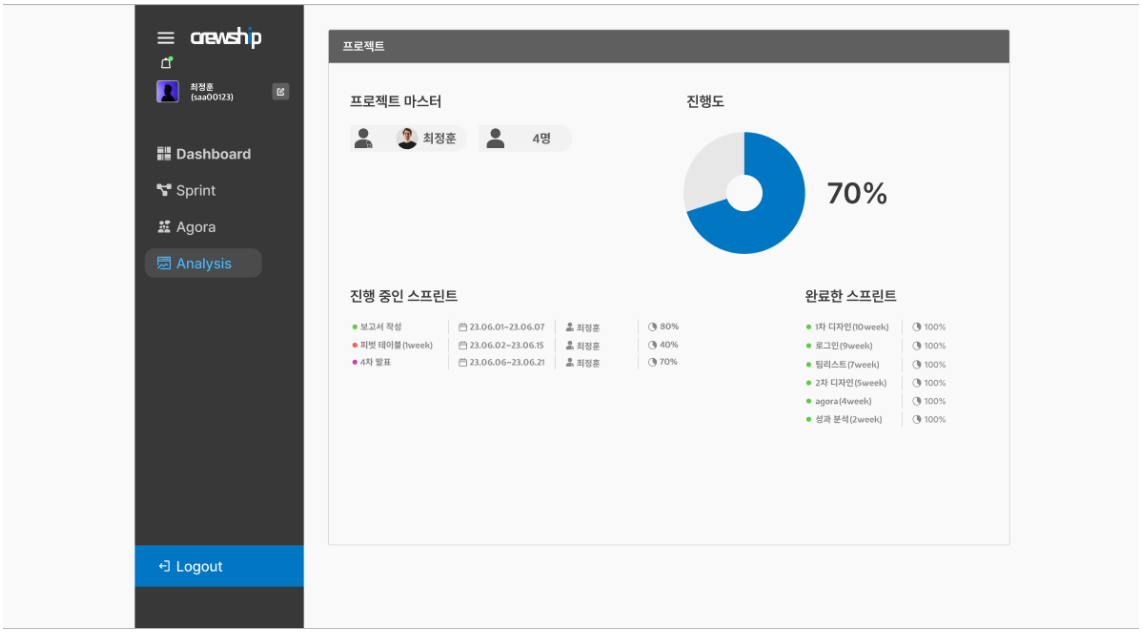
- 6월 일정에 대한 메인 화면
- 현재 프로젝트 진행도와 역량 그래프, 앞으로의 일정 등을 확인 가능

Agora (질의 응답 게시판)



- StackOverflow를 참고하여 만든 질문 게시판
- 해당 게시판에서의 질의응답은 커뮤니케이션 역량에 포함

성과 분석



- 프로젝트의 진행도와 비교분석을 위한 그래프를 확인 가능
- 현재 비교 분석 그래프의 미완성으로 인해 프로젝트 진행도와 스프린트 현황만 확인 가능한 상태
- 추후 본인의 역량 확인뿐만 아니라, 다른 팀원과의 비교분석도 가능하도록 피벗 테이블을 만들 예정