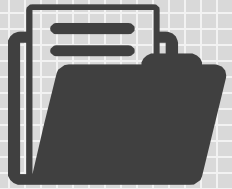
2022 SmileGate Winter :// Dev.Camp



PMP

Team Ottogi

Team Ottogi 

Project Management Plan

FRONT END : 김현우 허다은
BACK END : 김수찬 박규현 백종인

2022 SmileGate Winter :// Dev.Camp

PMP

주제

팀 목표

Work Flow

개인 계획

팀 마일스톤

1. 주제

Discord Clone Coding

Discord ?

- ✓ 음성, 채팅, 화상통화 등을 지원하는 인스턴트 메신저
- ✓ 대한민국에서는 주로 온라인 게임을 즐기는 사람들이 많이 이용하는 편이며, 게임용 메신저의 대명사.
  - ***실시간 채팅***
  - ***실시간 음성***
  - ***개별 서버 단위의 메신저***

주제 선정 이유

- 팀원들이 평소에도 대부분 사용해본 서비스로 이용 경험이 다수 있다.

- 채팅서버, 시그널링 서버, 알림 서버, 인증 서버, 상태관리 서버 등 기능에 있어 복합적으로 이루어져 있고, 다양한 기능의 구현을 통해 성장을 원하는 팀원들의 니즈에 적합하다.

- 채팅, 음성, 화상대화 등 각 기능을 구현 하는데 있어서 팀원들이 기존에 접해보지 못했던 아키텍처와 기술 스택들이기 때문에 도전 목표에 적합하다.

2. 팀 목표

팀 목표

“TEAM GOAL 🎯”

1. 중요한 것은 꺾이지 않는 마음!

- 캠프 기간 동안 중도탈락 없이 프로젝트 끝까지 마무리하기

2. 기억은 희미해지지만 기록은 희미해지지 않는다.

- 이슈 상황이 생겨서 문제 해결을 하거나 구현을 위해 필요한 지식을 학습한 일련의 과정들에 대해서 문서화하여 회고 할 수 있는 자료 만들기

3. 도전은 경험을, 경험은 기회를

- 경험이 없는 아키텍처, 디자인 패턴에 대해서 이해하고 적용하기

3. Work Flow

Meeting

Weekly Planning

- 일요일 1시 정기회의 (오프라인)
- 일주일 간 작업과정, 결과물 공유 & 발표
- 다음 정기회의까지 프로젝트 진행 방향 논의

Meeting

Daily Planning

- Slack Bot을 활용한 데일리 스크럼
- 코어타임(매주 화, 목 22:00 ~ 24:00) 온라인
- 팀원들의 일간 작업 진행정도 파악 가능

화면공유를 통하여 딴짓 방지

Question

- 오늘 기분은 어떠시나요?
- 오늘 할 일 적기
- 어제 하던 일 중 남은 일 적기
- 팀원과 공유하고 싶은 내용이 있다면?
- 현재 가장 직면한 큰 문제는 무엇인가요?
- 팀원들에게 하고 싶은 말

Trouble Shooting

- Git을 활용해 지속적으로 이슈 공유
- 이슈 있으면 주말 회의 또는 코어 타임 발언 및 팀원과 논의
- 슬랙 이슈방에 질문 및 의문점 공유

Development Rules

- 변수 네이밍의 경우 BE, FE 모두 일관되게 작성
  (진행하면서 협의)
- 코드는 Depth 2 이하로 지향
- Console 출력 보단 Logging 사용
- Commit convention 통일

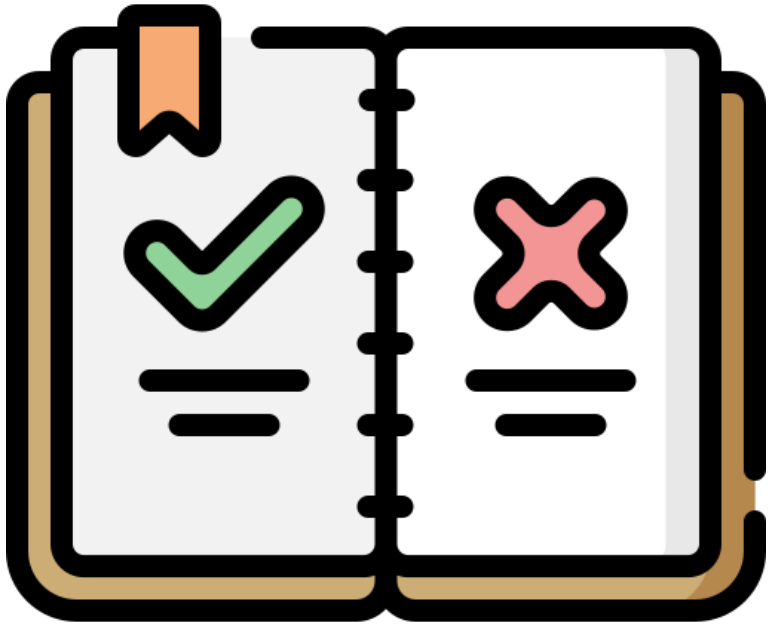
AngularJS Commit Convention 준수

Feat	새로운 기능 추가
Fix	버그 수정
Docs	문서 관련
Style	스타일 변경
Refactor	코드 리팩토링

Ground Rules

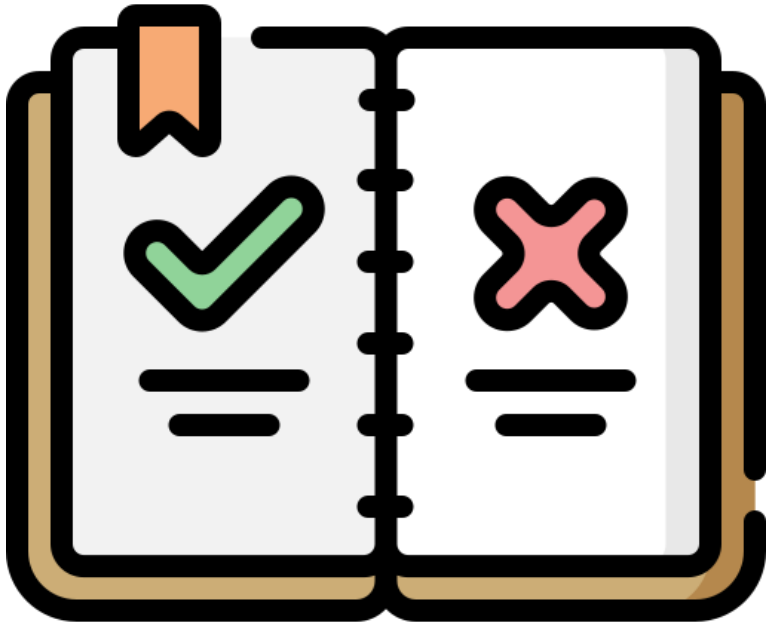
주의할 점

- 개인 일정 짤 때 팀 일정(코어타임 + 팀 회의) 반드시 고려!
- 일정이 생기면 미리미리 언급 (최소 3일전)
- 코드 짜다 "아" 하지 않기



Ground Rules 습관

- 데일리 스크럼 (하루 일과 보고)
- 회의 후 회의록 및 개발일지 작성
- 호칭은 자유롭게!
- 메신저에서 의견 내면 이모지 반응 해주기 ♡
- 에러 생기면 2시간 이상 혼자 고민 말기!



4. 개인 계획

김수찬 - 백종인 - 박규현 - 허다은 - 김현우

개인목표 - 김수찬

협업의 목적을 제대로 이해하고, 협업의 장점을 활용할 줄 아는 프로그래밍 습관 기르기

Current

- 팀 단위 프로젝트 경험은 있지만, 프로그래밍에 개인적인 성향이 많이 나타남
  협업 능력의 부족, 코드 리뷰시 해독이 필요함

- 변수들의 최적화 능력 부족 및 정리가 되지 않은 프로그래밍
  변수 선언이 효율적이지 못함

- 일회성이 강한 Function 및 Object

개인목표 - 김수찬

협업의 목적을 제대로 이해하고, 협업의 장점을 활용할 줄 아는 프로그래밍 습관 기르기

GOAL

- Markdown을 활용한 스펙 설명(기능 설명 및 버전 관리)
  필수적인 요소(필요한 input 및 return하는 output 제시)
- 컨벤션 및 합의된 변수명을 활용, 협업에 용이하게 프로그래밍 방식 변환
  git commit을 직관적으로 작업
- Function들의 기능 세분화 -> 유지보수가 용이

개발계획 - 김수찬 HOW TO WORK (1)

1. 음성대화 채널 및 화면 & 화상대화

A. WebRTC를 이용하여 P2P 방식으로 Streaming 서비스 구현

B. TURN방식 사용으로 Device 권한문제 조정

C. Mic와 카메라(화상대화)는 권한을 받은 후 server로 데이터를 전송

D. 화면 공유의 경우, Device에 대한 접근을 조사 해봐야함

E. Streaming할 데이터는 Socket.io를 이용하여 Client - Server 통신

개발계획 - 김수찬 HOW TO WORK (2)

1. **Signaling 서버** 
    A. P2P의 기본 방식인 Mesh 방식의 문제를 해결하기 위해 SFU 혹은 MCU 방식을 사용
    B. Submit 받는 데이터를 후처리 없이 바로 전송하는 SFU방식을 사용
        a. MediaSoup에 대한 조사 혹은 Socket.io를 사용
    C. DM 서버 역시 디스코드 내 Grouping 기능을 적용하기위해서 SFU방식 사용
2. 서버 생성 시 Signaling 서버가 생성될 수 있도록 작업
3. 서버를 생성하는 작업을 수행하는 알고리즘 작업

개발계획 - 김수찬

주차 개발 계획

- **1 ~ 2주차 - 음성대화 및 화상대화 서버 생성**
  - 실시간 음성대화 시스템 생성
  - TURN 방식으로 권한 접근
  - Audio데이터 업로드
  - 화상대화 서비스 작업
  - 테스트 데이터 바탕으로 테스트 진행

- **3 주차 ~ 4주차**
  - 음성 및 화상대화 서버 생성 구현
  - Signaling 서버와 대화 서버 연결

개인목표 - 백종인

Current

- 모르는 기술이나, 개념이 나오면 사용방식만 얼추 학습하고,
  프로젝트에 적용 후에 넘어가는 방식
- 장기적으로 바라보았을 때, 한 번 개념을 제대로 정립하지 않으면 똑같은 과정을 많이
  반복하는 일이 생김
- 프로젝트를 개발하다 보면 구현에 급급하여 유지 보수에 용이한 코드 작성에서 멀어짐

개인목표 - 백종인

GOAL

- 모르는 개념을 학습할 때, 효과적으로 학습하는 방식을 정립하고 싶음
    - 모르는 개념들을 마주하고 공부할 때, 노션에 정리해가면서 학습해보고,
      정기 회의시간에 공부한 내용을 공유함으로, 잘 학습하였는지 점검해보기
- 유지 보수에 용이하고 클린한 코드 작성 습관을 기르고 싶음
    - 기능에 맞춰 테스트 코드를 작성해보고, 코드가 테스트하기에 잘 작성되었는지 점검
    - 코드 Depth가 2를 넘지 않도록 제한
    - Depth를 맞추기 위하여 기능을 세분화해서 나누고, 한 메소드가 하나의 기능만 수행할 수 있도록 코드 작성

개발계획 - 백종인 HOW TO WORK (1)

✓ 개발

A. 인증 파트와 채팅 파트 구현

B. Java를 기반으로 Spring Framework와 Spring Boot를 이용하여 개발

C. 역할에 맞게 패키지를 구분하여 개발 진행

D. 기능에 맞춰 테스트 코드 작성

E. 개발 중 모르는 내용이 있다면 우선적으로 혼자 찾아보고,
1차적으로 팀원, 2차적으로 멘토님께 여쭤보기

개발계획 - 백종인 HOW TO WORK (2)

✓ 채팅 서버

A. 유저들간 채팅위한 서버 구현

B. Spring, STOMP 기술 적용하여 구현

C. 프로젝트 규모가 커진다면 외부 Message Broker 사용 시도 해보기

✓ 유저 서버 (+ 인증)

A. 회원가입 및 로그인

B. Spring Security 활용하여 비밀번호 암호화

C. Google SMTP 사용하여 이메일 인증

D. Spring Interceptor 로 로그인 인증

E. 회원 정보 수정

F. 친구 관리

개발계획 - 백종인

주차 개발 계획

- **1주차 - 유저 서버 개발**
  - 유저관리 API
    - 회원정보 수정
    - 친구관리
  - 인증 API
    - 회원가입 및 회원가입시 이메일 인증
    - 로그인 및 로그아웃
    - 비밀번호 암호화
    - 세션과 인터셉터 활용한 로그인 유지
- **2 ~ 3주차 - 채팅서버 개발**
  - Web Socket 기반 서버간, 개인간 채팅 기능
  - STOMP 활용하여 pub / sub 구조로 개발
  - 채팅 메세지 수정 및 삭제
  - 채팅창 파일 업로드
- **4 주차 - 기능 및 통신 점검 기간**
  - 앞서서 개발한 기능들 점검 및 리팩토링
  - 다른 서버 및 프론트와의 통신 위주 테스트 진행

개인목표 - 박규현

Current

- UI와 도메인 로직 분리가 안됨
- 변수, 함수명을 혼자 알아볼 수 있게 작성
- BACK END 경험 부족(REST API, DB에 대한 경험X)

개인목표 - 박규현

GOAL

- UI와 도메인 로직의 분리(MVC 또는 MVVM패턴 참고)
- 변수, 함수명 올바르게 사용하기 (eslint이용하여 "*airbnb style guide*" 참고)
- MSA 아키텍처를 바탕으로 기능 목록 작성하기
- 작성된 기능 단위 개발 및 테스트 코드 작성 (모카(Mocha)를 이용한 TDD)
- 문제가 발생했을 때 고민하고 회고록 작성 -> 구글링을 하는 것에서 그치는 것이 아닌 문서화 시켜, 복습, 정리 능력까지 갖추는 것이 목표

개발계획 - 박규현 HOW TO WORK (1)

- **코드 작성 방법** 
  - 기능 목록 작성
  - 기능 단위 개발 -> 기능 단위 테스트
  - 디렉토리 구조화 -> 테스트
  - 코드 작성 시 발생한 이슈, 장애, 몰랐던 개념들을 바탕으로 회고록 작성

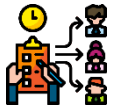
- **채널 관리 기타 서버 -> Management Server (MongoDB, redis)** 
  - 사용자(user, admin)의 서버(또는 채널) 참가, 생성, 삭제
    - 코드 인코딩 법 : 밀러의 법칙에 따른 base62 7자리 , 이미지
    - 서버 생성 시 DB 저장 (서버 명, 서버 대표 이미지, 서버 초대 코드, URL) -> 하위에 채널 목록 저장
    - 채널 생성 시 DB 저장 (채널 명, 채널 초대 코드)
    - 생성 : REST API 사용(POST)
    - 삭제 : REST API 사용(DELETE)

개발계획 - 박규현 HOW TO WORK (2)

- **채널 관리 기타 서버 -> Management Server (MongoDB, redis)** 
  - 사용자(user)의 서버 내 채널 참가
    - 채널 생성 시 DB에 저장 (DB상 서버의 하위 -> 채널 명)하여 출력
    - 채널에 참가자가 있을 경우 출력(WebRTC, Socket 에서 데이터를 받아야 하는 부분)
  - 서버 리스트, 서버 내의 채널 리스트
    - 추가로 참가자 리스트도 보여줘야 함
    - 채팅방, 서버 생성 시 연결되는 DB에서 데이터를 받아와서 출력
    - 요청: REST API 사용(GET)
- 파일(이미지, pdf 등) 저장 서버 -> File Server (MySQL) 
  - node.js의 multer 모듈 활용

개발계획 - 박규현

주차 개발 계획

- **1주차 ~ 3주차 - 관리 서버 개발** 
  - 서버관리 API (1 ~ 2 주차)
    - 서버 네비게이터
    - 서버 사용자 정보
    - 서버 내의 채널
  - 채널관리 API (3주차)
    - 채널 네비게이터
    - 채널 사용자 정보

- **4주차 - 파일서버 개발** 
  - Multer를 이용하여 파일(이미지, pdf..) 서버 구현
    - 파일 관리 API
  - 아키텍처 리뷰를 바탕으로 리팩토링

개인목표 - 허다은

Current

- 일회적인 이슈 해결 방식
- 구체적이지 않은 폴더 구조 및 컴포넌트 정리 방식
- 개발 협업 경험이 있으나 주로 개인 프로젝트를 진행하여 협업 경험이 부족함
- 페어 프로그래밍과 코드 리뷰 경험 역시 부재함

개인목표 - 허다은

GOAL

- 이슈가 발생했을 때 팀원들에게 질문하고 레퍼런스를 참고하고, 해결하는 과정을 노션에 기록하면서 이슈 관리법 정립하기
- 디자인 패턴을 모방하여 폴더 구조와 컴포넌트 정리 방식 세우기
- 매주 팀원과 회의에서 프로젝트 진행 상황 및 의견을 공유하여 협업을 경험하고,
  깃 커밋 컨벤션을 설정하고 깃 브랜치 전략을 사용하여 협업을 위한 깃허브 사용법 갖추기
- 팀 내 프론트엔드 개발자와의 페어 프로그래밍과 코드 리뷰를 통해 다른 관점에서 코드를 바라보고 다양한 코드를 접하여 코드 한 줄에도 논리적인 견해를 가진 코드를 작성하는 개발자 되기

개발계획 - 허다은 HOW TO WORK (1)

- **상태 관리 라이브러리** 
  - 비동기적으로 상태들을 관리해야 하므로 redux, recoil, react-query를 직접 사용하여 테스트해보고 비교한 후, 비동기 데이터 처리에 용이한지 고려하여 프로젝트에 적합한 상태 관리 라이브러리를 선택할 예정
- **이슈 해결 과정을 기록하고 공유하여 이슈 관리법 정립하기** 
  - 개발 과정에서 이슈가 발생하면 팀원들에게 질문하고 레퍼런스를 참고하여 해결하는 과정을 팀 노션에 작성한다.
  - 매주 오프라인 팀 회의에서 팀원들에게 작업 과정과 이슈 내용을 공유하고 놓친 부분은 없는지 점검한다.

개발계획 - 허다은 HOW TO WORK (2)

- **Design Pattern 적용하여 디렉터리 관리** 
  - 컴포넌트를 총 5가지 레벨로 구성된 일반 Atomic Design Pattern을 3가지 레벨로 변형한 RIDI의 디렉터리 구조를 참고하여 디렉터리를 관리한다.
  - 5가지 레벨을 3가지 레벨로 줄임으로서 pages부터 시작된 props가 atom까지 4단계가 아닌 2단계로 끝날 수 있으므로 props drilling issue를 예방할 수 있다.
  - 디렉터리 구조
    - **Atoms**
      - 단순히 한 가지 기능을 담당하는 HTML tag 하나 단위로 구성한다. (ex. input, button, ...)
      - 재사용성을 높이기 위해서 폰트, 색상, 배경, 이벤트 등을 자유롭게 설정할 수 있도록 설계한다.
      - 상위 단계에서 자유롭게 가져다쓸 수 있도록 margin, padding 속성 자제한다.
    - **Blocks**
      - 하위 단계인 atom들을 조합하여 구성한다.
    - **Pages**
      - 하나의 화면을 구성한다.
      - 하위 단계인 block들을 layout을 계산하여 배치한다.

개인목표 - 김현우

Current

- git을 통한 프로젝트 관리를 프로젝트 전반에 적용해본 경험 부족
- react 개발 시에 리액트 디자인 패턴이나, 구성 방식에 대해서 고려하여 작업해본 경험이 부족하다는 점.
- 이전에 했던 프로젝트들을 돌아보면 진행과정에서 항상 구현을 우선 시 하다보니 구현을 하는 과정에 생긴 문제 해결 과정이나 학습한 지식들에 대한 기록이 부족했었고, 관련 자료에 대한 정리 또한 부족해서 프로젝트 완성 후에 회고에 어려움을 느낌

개인목표 - 김현우

GOAL

- **GitHub issue, pull request, project board 기능 적극 활용**
  - GitHub 제공 기능을 통해 구현 관련 문제, 이슈에 대해 효과적으로 관리 기록
  - pull request 기능을 통해 코드리뷰를 받고, 관련 피드백을 남길 수 있도록 함.
- **Design pattern + storybook 적용**
  - Atomic design pattern(ridi version)의 적용과 storybook을 통해 컴포넌트의 재활용성과 확장성에 대해 고려해 진행
- **꾸준하게 프로젝트에 대해 기록하고 회고하는 습관 만들기**
  - 문제 상황 해결 과정, 특정 기능의 구현을 위한 추가 학습 발생 시 마다 기록을 할 수 있도록 습관가지기
  - 성급한 구현 보다도 새로운 아키텍처, 기술 스택에 대한 확실한 이해하기
  - 팀원과의 기록 리뷰를 통해 내 기록을 통해 경험을 충분히 이해하고 설명할 수 있는지 확인 하기

개발계획 - 김현우 HOW TO WORK (1)

- **story book, atomic design pattern(Ridi version)의 적용**
  - 페이지별 분석을 통해 최소 단위로 요소들을 쪼개기
  - atom - block - page 단위로 요소들을 조합하여 각 페이지 구성
  - storybook을 통해 각 요소별 재사용의 추구하며 더불어 요소들에 대한 테스트와 리뷰를 할 수 있도록 진행
- **상태 관리 라이브러리**
  - 실시간 업데이트 내용(채팅, 유저 관련 정보 등)을 관리하기 위해 필수적일 것
  - 이번 캠프 이전에 프로젝트 전반에 적용한 경험이 부족하여 학습에 시간이 걸리는 redux 보다는 비교적 사용이 용이한 react query 혹은 recoil를 통해 구현할 계획임.

개발계획 - 김현우 HOW TO WORK (2)

- **프로젝트 기록** 
  - 주마다 개발과정에서 발생한 트러블 슈팅과정 혹은 관련 자료, 학습내용 문서화 -> 노션에 작성후 -> 티스토리 블로그 or velog 중 하나로 올리기
  - 팀원들에게 리뷰 받고, 프로젝트 진행 + 전반적 이해도가 괜찮은 문서인지 피드백 꼭 받기

개발계획 - Front End

주차 개발 계획

- **1주차**
  - 개발 스택 선택 및 환경 구축
  - 디스코드 페이지 별 분석 후 atom, block, page 단위로 기능 구별
  - atom 단위별 기능 구현(버튼, 유저 프로필 등)
  - 로그인, 회원가입 폼 구현

- **2 ~ 3 주차**

1주차에 구현 해놓은 atom 기반으로 block, page 구현

- 내 채널(친구 관리 + 메인페이지 - 홈)
- 개별 서버(채널 커스터마이징)
- 개별 서버(채널채팅)
- 개별 서버(상세 기능)
- 개별 서버(음성 통화)
- 개인 채널(친구 관리)

-> 구현 시에 최소한 백엔드 api 연결 테스팅 가능 할 정도로 구현

개발계획 - Front End

주차 개발 계획

- **4주차**
  - 2~3주차에 구현하는 기능 리뷰 후 리팩토링 혹은 구현 마무리
  - 개인 채널(DM)
  - 개인 채널(음성 통화)
  - 개인 채널(상세 기능)

-> 우선 순위에서 뒤쪽인 기능들 구현 시작 및 기존에 구현한 기능들 코드 리뷰와 가능하다면 리팩토링 시도

5. 팀 마일스톤

Team MileStone

Step. 1

~개발 4주차 💻

- ✓ 음성대화
- ✓ 카메라
- ✓ 채팅

<추가>
- ❖ 화면 공유
- ❖ 파일 업로드

Step. 2

~프로젝트 마무리 🎉

Step 1의 구현을
통하여 실제 회의를
진행하는 것까지!

Step. 3

한계, 그 너머로! 🔥

- ✓ 그룹 추천 기능
- ✓ 친구 추천 기능
- ✓ 코드 공유 시스템

감사합니다.