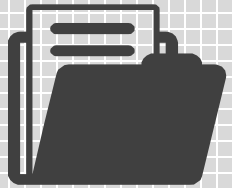
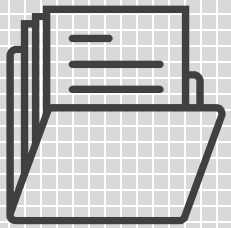
2022 Smilegate Winter :// Dev.Camp



Mid



Team Ottogi

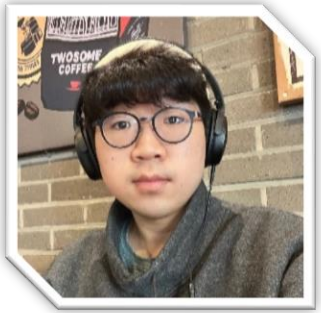
# Team Ottogi


## 중간 공유회

FRONT END : 김현우 허다은
BACK END : 김수찬 박규현 백종인

# Ottogi 팀 소개

*저희 팀원을 소개합니다!*



FE / 김현우

FE / 허다은

BE / 김수찬

BE / 박규현

BE / 백종인

# 2022 Smilegate Winter :// Dev.Camp

PMP

주제

목표

Work Flow

Architecture

진행 상황

Dev Flow

주제

# Discord Clone Project

# 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.

# 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

# 개인목표 - 김수찬

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

## Current

- ✓ 협업 능력의 부족, 지저분한 코드 작성
- ✓ 비효율적인 변수 선언
- ✓ 일회성이 강한 Function 및 Object

→

## GOAL

- ✓ 마크다운의 활용, 기능 설명 및 버전 관리
- ✓ 컨벤션 및 합의된 변수명을 활용
- ✓ 직관적 git commit
- ✓ Function들의 기능 세분화(유지보수의 용이)

# 개인목표 - 백종인

*새로운 개념에 대한 효과적 학습 방식 정립, 편한 유지보수가 뒤따르는 클린코드 작성 습관 기르기*

## Current

- ✓ 처음으로 접한 기술, 개념에 대해 학습보다 사용과 구현을 우선시하는 경향
- ✓ 개념의 확실한 정립 부족으로 인한 학습과정의 반복
- ✓ 프로젝트 진행 시 구현에 우선시 하여, 유지보수에 용이한 코드 작성의 부족

→

## GOAL

- ✓ 모르는 개념의 효과적 학습 방식 정립
  - 노션을 통한 학습 내용 기록
- ✓ 클린 코드 작성 습관 기르기
  - 코드 Depth가 2를 넘지 않도록 작성
  - 테스트 코드 작성, 테스트를 통한 코드 점검

# 개인목표 - 박규현

*MSA & TDD의 적용, 깨끗한 코드의 작성, 문제 해결과정에 대한 문서화 능력 향상*

## Current

- ✓ UI와 도메인 로직 분리 X
- ✓ 지저분한 코드 작성
- ✓ BACKEND 경험 부족
  - REST API, DB에 대한 경험 X

→

## GOAL

- ✓ MVC, MVVM패턴의 적용을 통한 UI, 도메인 분리
- ✓ Airbnb style guide를 통한 클린한 코드 작성
  - 올바른 변수, 함수명 작성
- ✓ Mocha를 이용한 TDD 적용
  - 기능 단위 개발 및 테스트 코드 작성
- ✓ 문제 해결과정에 대한 문서화, 복습, 정리 능력 향상

# 개인목표 - 허다은

*협업을 통해 성장하고 팀과 함께 나아가기, 디자인 패턴의 적용을 통해 효율적인 설계 추구*

## Current

- ✓ 일회적인 이슈 해결
- ✓ 구체적이지 않은 폴더 구조 및 컴포넌트 정리 방식
- ✓ 페어 프로그래밍, 코드 리뷰에 대한 경험 부족
- ✓ 팀 프로젝트 협업 경험의 부족

→

## GOAL

- ✓ 이슈 해결과정의 문서화를 통한 이슈 관리법 정립
- ✓ 디자인 패턴 적용, 효율적 디렉토리 구조 설계
- ✓ Git commit convention, git-flow의 적용
  - 협업을 위한 GitHub 사용법 갖추기
- ✓ 페어프로그래밍, 코드리뷰의 적용
  - 시각의 다양화, 논리적 견해를 가진 코드 작성

# 개인목표 - 김현우

*GitHub를 통한 효율적 프로젝트 관리, 꾸준한 기록, 회고하는 습관 기르기*

## Current

- ✓ GitHub를 통한 프로젝트 관리 부족
- ✓ 웹 프론트엔드 아키텍처에 대한 이해 부족
- ✓ 개발과정에서 구현을 우선 시 하는 경향, 과정에 대한 기록의 부족, 회고 어려움

→

## GOAL

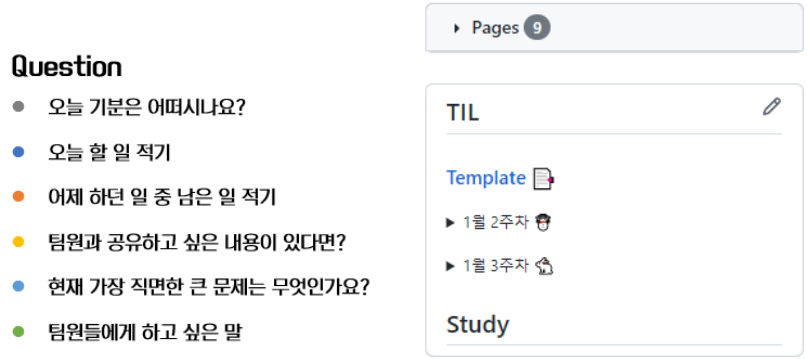
- ✓ GitHub issue, pull request 활용
- ✓ Design pattern + storybook 적용
  - ▪ 컴포넌트의 재활용성, 확장성 고려
- ✓ 꾸준한 기록, 회고하는 습관 만들기

# Work Flow

# Work Flow

## Meeting

Daily Planning




- Slack Bot을 활용한 데일리 스크럼

  평일 매일 오전 11시 까지 공유!
  팀원들의 일간 작업 진행정도 파악 가능

- 매일 본인 TIL을 Wiki에 작성 및 공유하기

  프로젝트에 꾸준히 참여함을 유도하고, 팀 목표 포기하지 않기를 도모하기 위함!

- 코어타임(매주 화, 목 22:00 ~ 24:00) 온라인

  실시간 화면 공유로 각자 코딩하고, 이슈 있을 시 실시간으로 공유

# Meeting

Weekly Planning

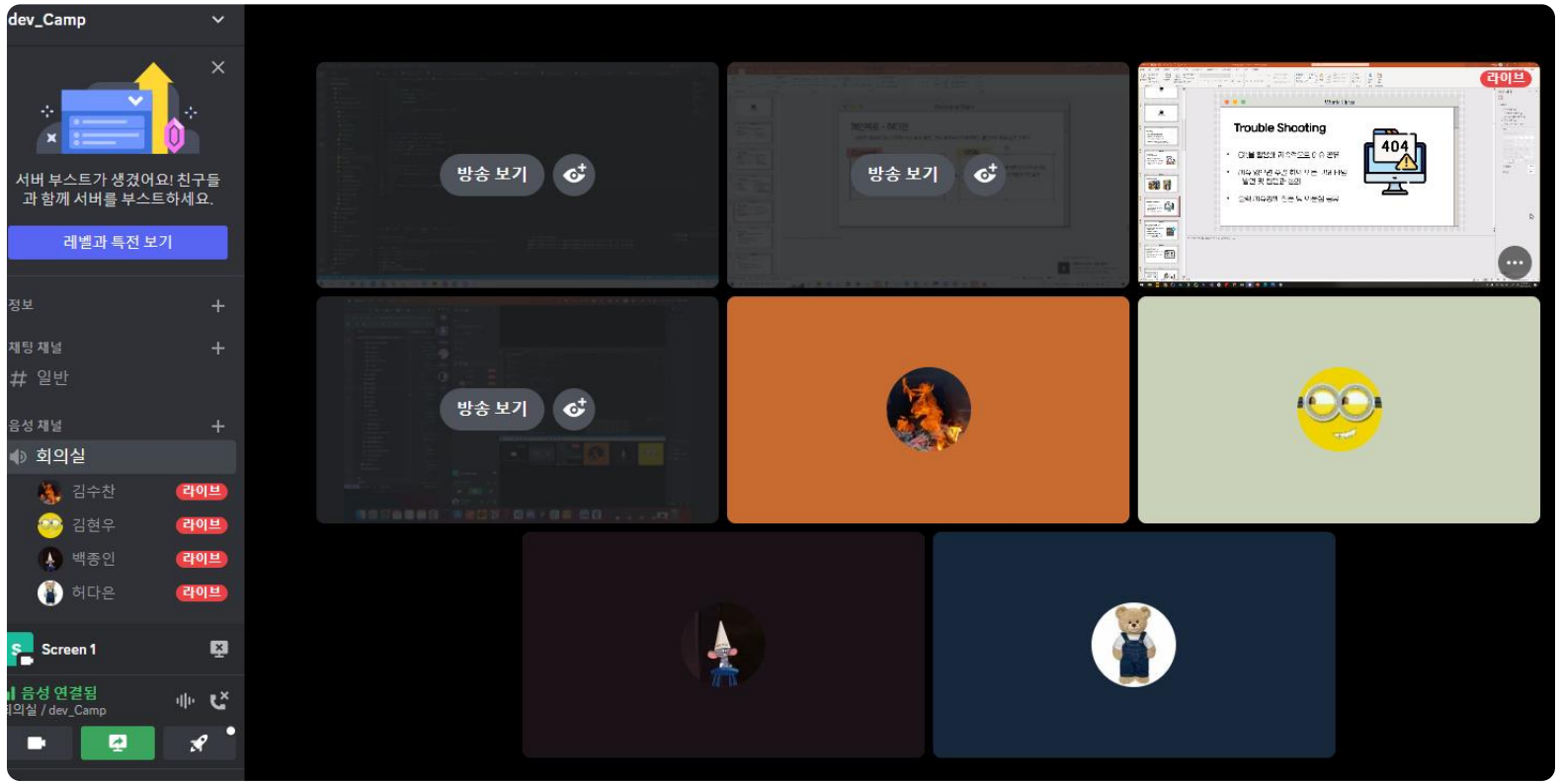
- 일요일 1시 정기회의 (오프라인)
- 일주일 간 작업과정, TIL 팀원간 점검
- 다음 정기회의까지 각자 목표 정하기

# Meeting

Daily Planning

## Discord

- 코어타임 화요일, 목요일 22:00~24:00
- 온라인 모각코, 팀회의 등 팀 활동 진행



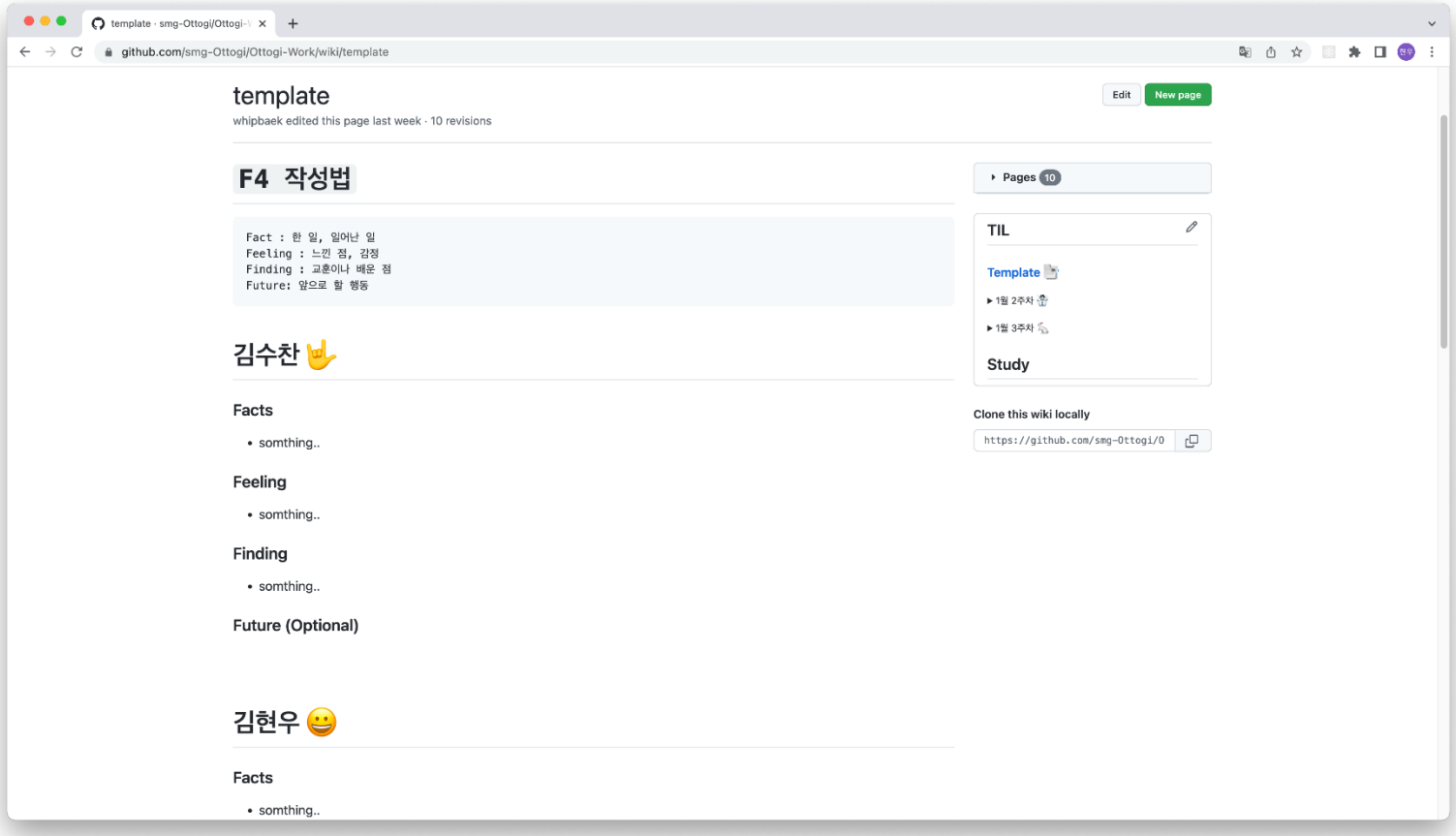
# Work Flow

## Meeting

Daily Planning

### TIL 작성

- GitHub Wiki 기록
- F4 작성법 적용



# Meeting

Weekly Planning

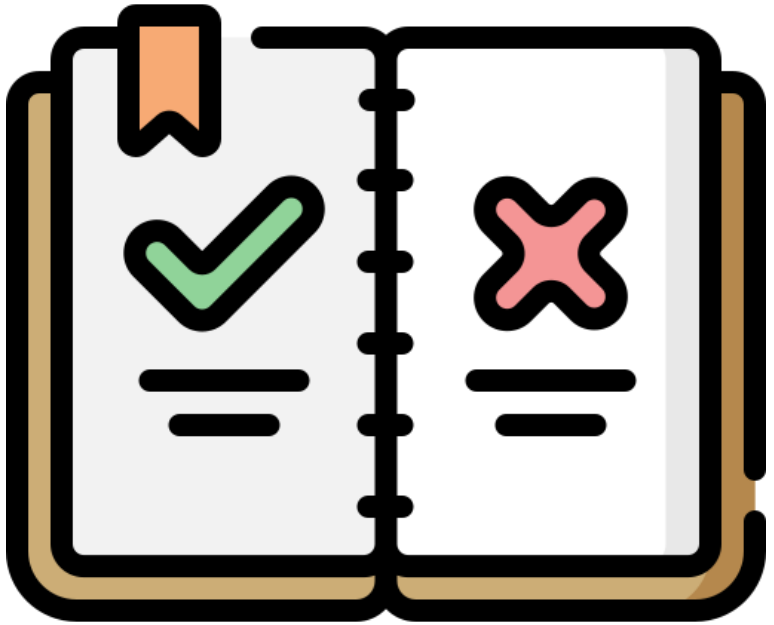
01.08 정기모임 😊

01.15 정기모임 🎵

# Ground Rules

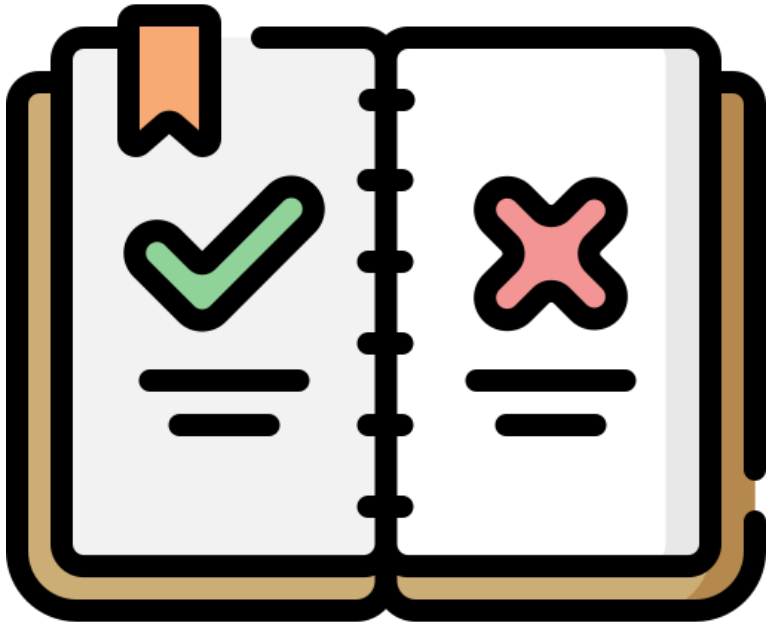
주의할 점

- 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!



# Dev Flow

# 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

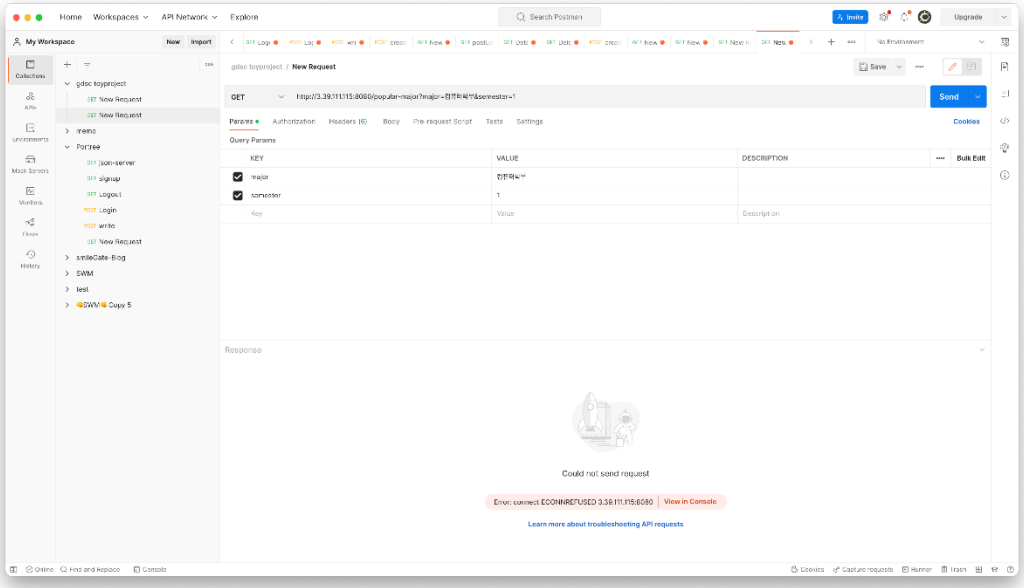
*AngularJS Commit Convention 준수*

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

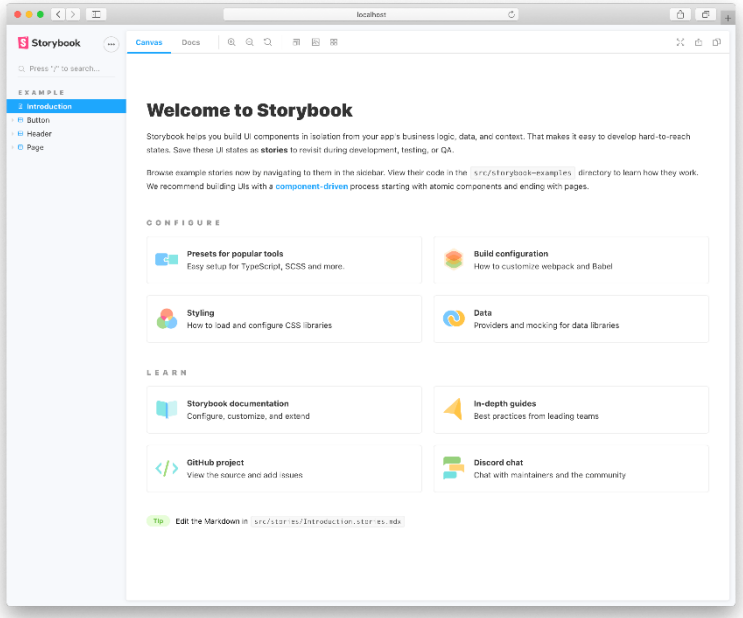
# Testing Tool

FE/BE 개발 AI 사용할 테스트 도구

POSTMAN




Storybook

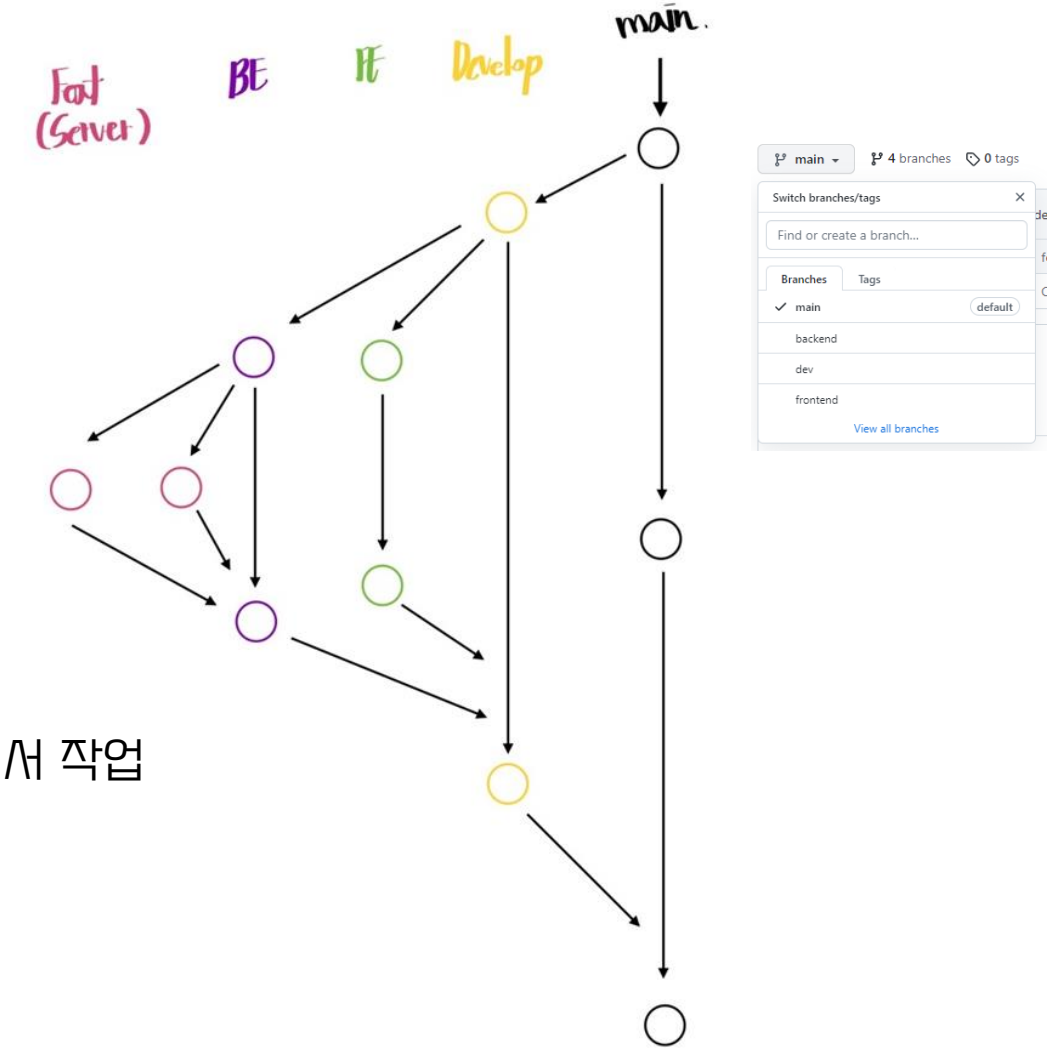


# Version Managing

- main (배포)
  - Develop(개발)
    - Front End
    - Back End
      - Feat(Server)

- BackEnd 는 각자 Server 기능에 맞게 Push

- Front는 Atomic Design에 의거하여 기능별로 나뉘어서 작업

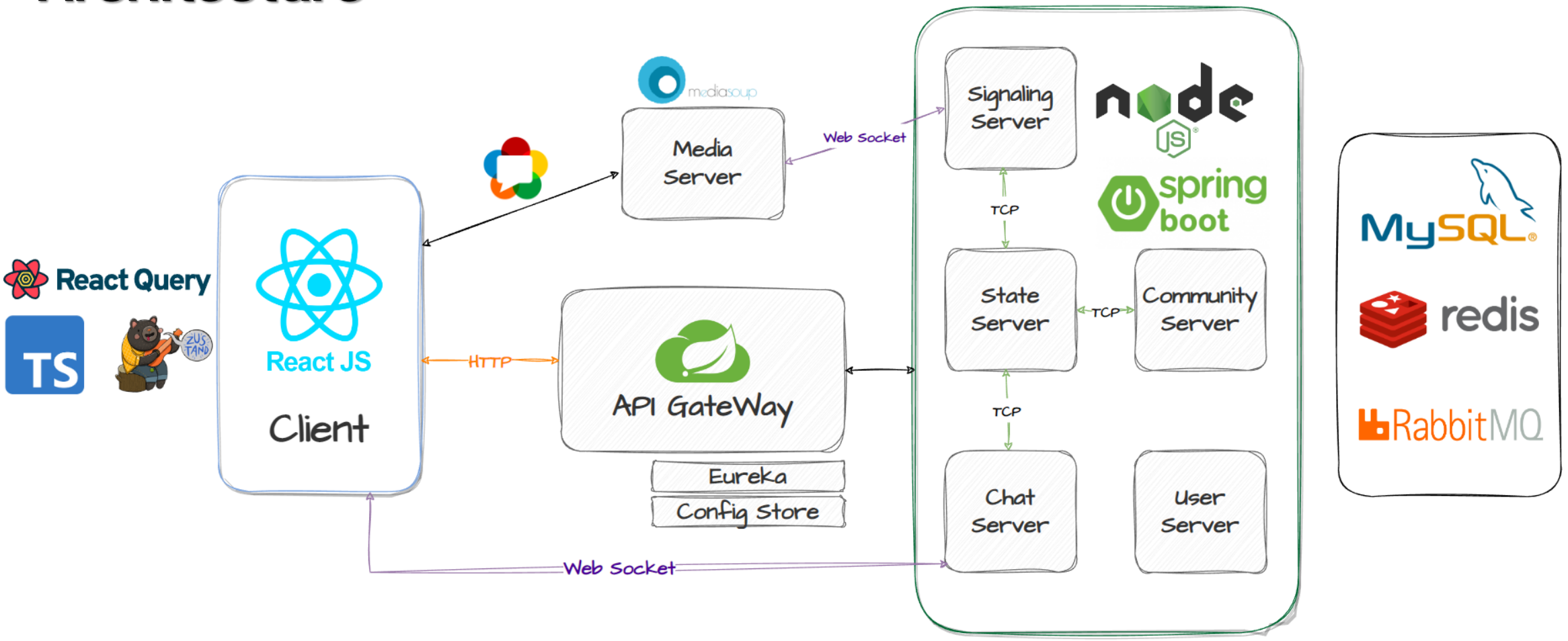


# Trouble Shooting

- Git을 활용해 지속적으로 이슈 공유
  Git Issue 활용
- 이슈 있으면 주말 회의 또는 코어 타임 발언 및 팀원과 논의
- Slack 이슈방에 질문 및 의문점 공유

# Architecture

# Architecture



# Architecture 업무 분담

## Front

**김현우**
**허다은**

UI / UX 설계

디스코드 디자인 시스템 구현

WebSocket 연동

## Back

**김수찬**

WebRTC, Signaling Server

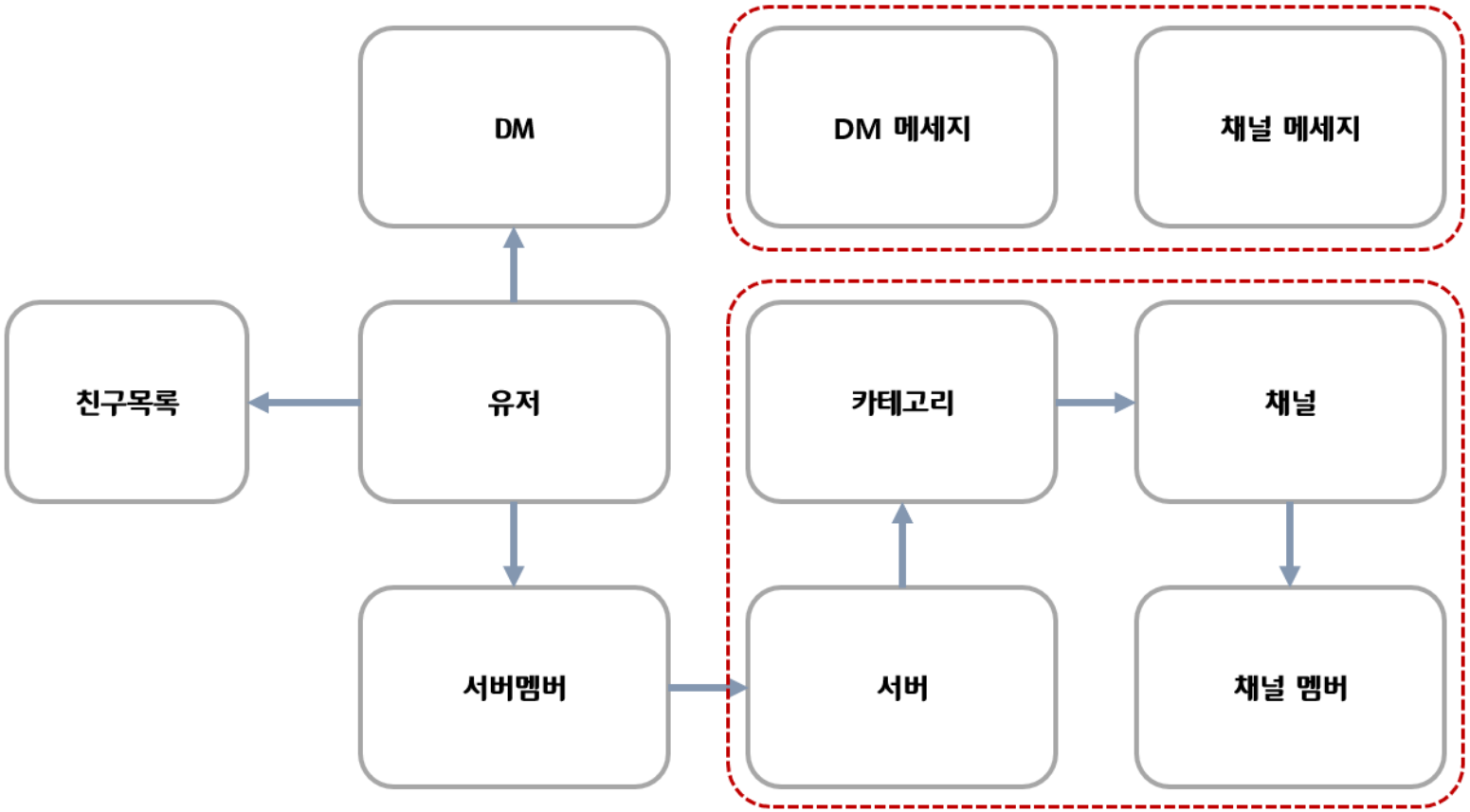
**박규현**

Community, State Server

**백종인**

API Gateway,
User, Chat Server

# Database ERD



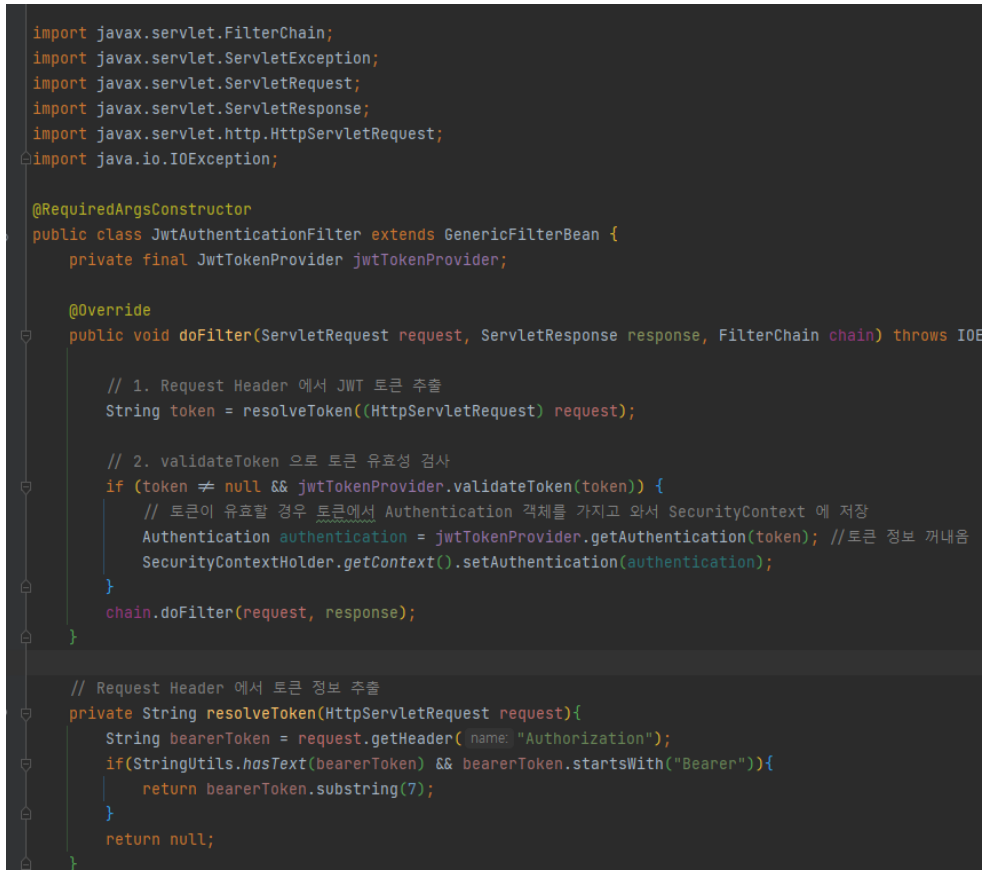
# Database ERD

# 진행 상황

# BACKEND 진행상황

## MSA 구조 채택하여 각기 서버 개발

- 유저 서버
  - DB 생성, 회원가입 로그인 및 JWT 인증
- 커뮤니티 서버
  - 채팅서버 및 시그널링 서버와 통신 학습
- WebRTC
  - Mediasoup 활용하여 튜토리얼 진행



```java
import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;

@RequiredArgsConstructor
public class JwtAuthenticationFilter extends GenericFilterBean {
    private final JwtTokenProvider jwtTokenProvider;

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOE

        // 1. Request Header 에서 JWT 토큰 추출
        String token = resolveToken((HttpServletRequest) request);

        // 2. validateToken 으로 토큰 유효성 검사
        if (token != null && jwtTokenProvider.validateToken(token)) {
            // 토큰이 유효할 경우 토큰에서 Authentication 객체를 가지고 와서 SecurityContext 에 저장
            Authentication authentication = jwtTokenProvider.getAuthentication(token); //토큰 정보 꺼내옴
            SecurityContextHolder.getContext().setAuthentication(authentication);
        }
        chain.doFilter(request, response);
    }

    // Request Header 에서 토큰 정보 추출
    private String resolveToken(HttpServletRequest request){
        String bearerToken = request.getHeader( name: "Authorization");
        if(StringUtils.hasText(bearerToken) && bearerToken.startsWith("Bearer")){
            return bearerToken.substring(7);
        }
        return null;
    }
```

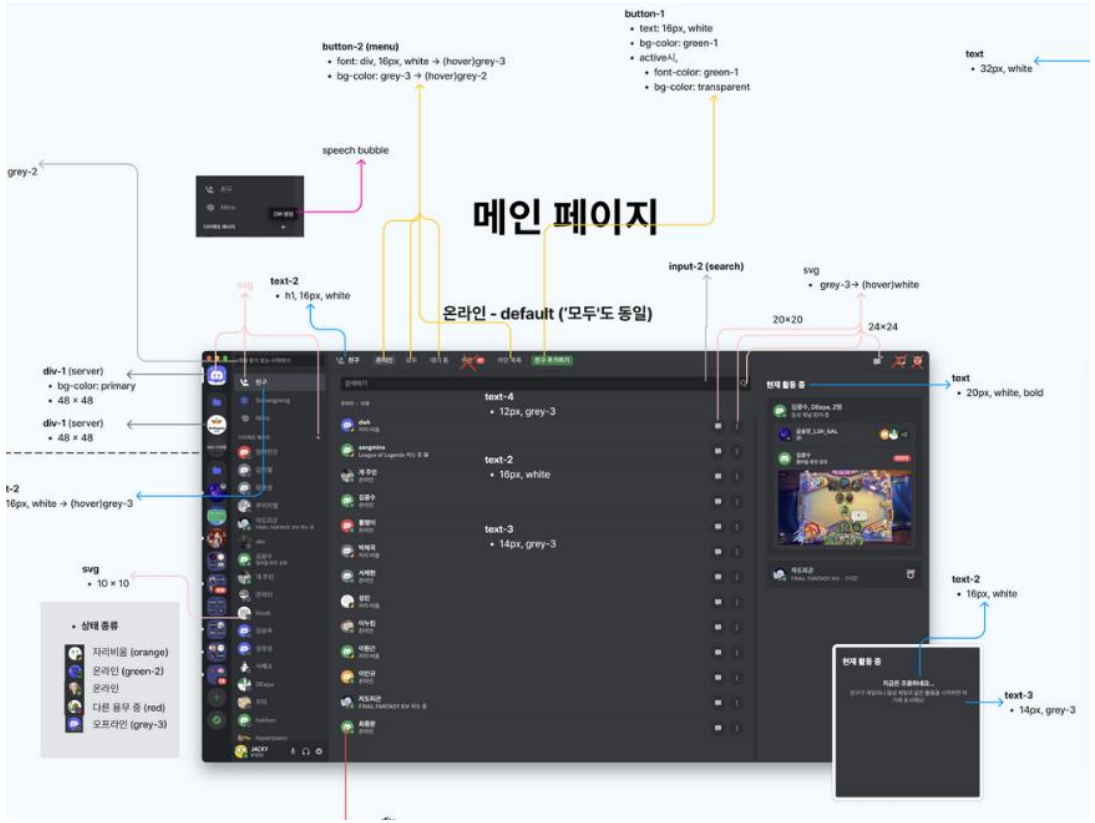
# FRONTEND 진행상황

## Atomic Design Pattern 적용

- Atom을 기준으로 컴포넌트 단위 별 세부 분석 진행
- Atomic Design Pattern 기반 React Ts 구조 설계

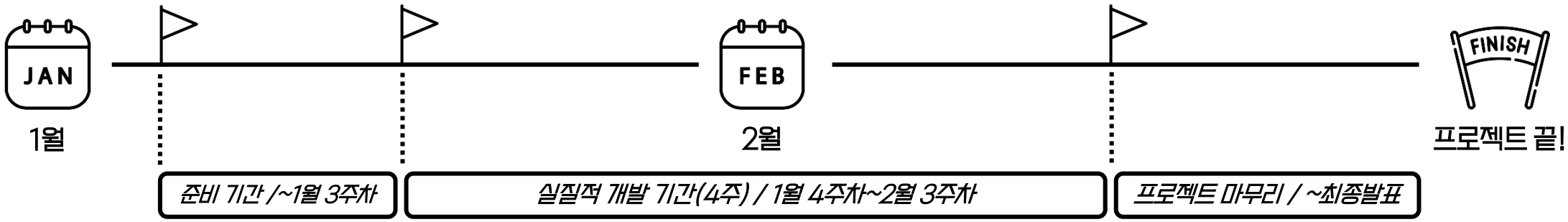
## Zustand + React Query

- recoil에서 Zustand + React Query 체제로 전환
- 관련 레퍼런스 참고 프로젝트 적용 방식 설계 및 학습 중



# TIMELINE

*스크럼 회의 / TIL 작성 / 매주 정기 회의, 코어타임 / 스터디*




- PMP 작성
- 아키텍처 설계
- Ground 룰 설정

- 주차 별 구현 목표에 맞춰 개발 진행
- 프로젝트 진행과 개인 목표의 융화

- 프로젝트 최종 발표
- 목표 달성 여부 체크

# END

감사합니다.