백엔드 방 관리 흐름 제안

테이블

- member
- room
- room_participant
- 일단 MySQL을 이용해 테스트 / 속도 문제가 있을 것 같기 때문에 redis 활용 검토

방 생성, 입장, 퇴장

- 기본적인 http 통신으로 진행
- 예를 들어, 클라이언트가 방을 생성하려 한다.
  서버는 room과 room_participant 테이블에 데이터를 추가하고,
  클라이언트에게 json 응답을 준다.

방 입장 후

- 웹소켓 통신으로 진행
- 기본적으로 아래 동작을 수행해야 한다
  - 웹소켓 연결 수립
    - 서버는 웹소켓 통신을 맺을 경로를 미리 지정해놓고,
    - 클라이언트는 그 경로를 통해 웹소켓 통신을 수립한다.

```
    @Override
    public void registerStompEndpoints(StompEndpointRegistry regis
try) {
        registry.addEndpoint("/connect")
            .setAllowedOrigins(
                "http://localhost:3000",
                "http://localhost:5174")
            // ws:// 말고 http:// 엔드포인트를 사용할 수 있게 해 준다.
```

```
            .withSockJS();
    }
```

```
connectWebsocket(){
        if(this.stompClient && this.stompClient.connected) return;
        // sockjs 사용하면 http엔드포인트 사용 가능.
        const sockJs = new SockJS(`${process.env.VUE_APP_API_
BASE_URL}/connect`)
        this.stompClient = Stomp.over(sockJs);
        this.token = localStorage.getItem("token");
        this.stompClient.connect({
            Authorization: `Bearer ${this.token}`
        },
            ()=>{
                // 웹소켓 통신 수립 성공시 코드...
            }
        )
    },
```

- 클라이언트와 서버 간의 메세지 송수신
  - publish: 클라이언트가 서버로 메세지를 보내는 것
  - subscribe: 클라이언트가 서버에게서 메세지를 받는 것
  - 서버는 publish/subscribe prefix를 미리 정해놓고,
  - 클라이언트는 그 경로를 통해 통신한다.

```
    @Override
    public void configureMessageBroker(MessageBrokerRegistry reg
istry) {
        // 클라이언트가 서버로 보내는 발행 요청
        // 예를 들어 /pub/**로 발행 요청이 오면, 컨트롤러 내 @MessageMa
pping 어노테이션이 붙은 메서드로 보내줌
        registry.setApplicationDestinationPrefixes("/pubish");

        // 클라이언트가 서버에게서 받을 메세지 경로
        // 클라이언트가 /topic/**를 구독하고 있으면, 메세지를 받을 수 있다
```

```
        registry.enableSimpleBroker("/topic");
    }
```

```
connectWebsocket(){
    // 웹소켓 연결 수립 후..
    ()=>{
    this.stompClient.subscribe(`/topic/${this.roomId}`, (message) => {
      const parseMessage = JSON.parse(message.body);
        this.messages.push(parseMessage);
        this.scrollToBottom();
  }, {Authorization: `Bearer ${this.token}`})})},
```

```
sendMessage(){
    if(this.newMessage.trim() === "")return;
          const message = {
        senderEmail: this.senderEmail,
        message: this.newMessage
          }
    this.stompClient.send(`/publish/${this.roomId}`, JSON.stringify(m
essage));
    this.newMessage = ""
}
```

- 그래서 배틀 방식을,
  - 준비
    - 클라이언트가 {type: READY, data: 유저번호} 메세지를 보낸다.
    - 서버는 이를 다른 클라이언트에게 전파한다.
  - 시작
    - 방장 클라이언트가 {type: START, data: .. } 메세지를 보낸다.
    - 서버는 이를 다른 클라이언트에게 전파한다.
  - 종료
    - AI 서버에서 웃음을 감지했음을 백엔드 서버로 보냈거나.

- 30초가 지났거나 등의 이유로
- 서버가 클라이언트에게 종료 메세지를 전파
- 그런데, 이때 30초가 지난 것을 프론트에서 감지해야 하는지, 서버에서 감지해야 하는지에 따라 단계가 늘어날 수 있을 듯하다