

# ASK2LIVE에 오신 것을 환영합니다.

• Live 3000000명



종료



허승



한정서



최교진



조영후



## 소개

### ASK2LIVE

ASK2LIVE는 “짜임새 있는 Live Q&A” 서비스입니다.

#### • 플랫폼 특징

1. 호스트가 특정 주제로 세션 오픈을 예약
2. 평소 관심 있었던 게스트들이 사전질문을 등록  
→ 호스트의 매끄러운 진행을 도울 수 있는 큐시트 역할
3. 라이브 중 게스트의 질문들은 세션에 참여한 모든 사람들에게 공유되어 흐름을 놓치지 않고 참여 가능
4. 음성이 부담스러운 유저를 위한 텍스트 질문, 채팅 기능 구비



메인 화면



세션 예약하기



사전 질문하기



라이브 화면

## 답변완료

### Frontend

### Backend

Q. 음성 스트리밍 서비스를 어떻게 구현하였나요?

- 음성 스트리밍 기술을 지원하는 Agora SDK (Software Development Kit) 사용
- ASK2LIVE에 필요한 기능 구현 과정 속 2가지 문제
  - 애코 : 발언자의 목소리가 자신의 디바이스에서 다시 재생되는 문제
    - 아고라 공식 문서의 API와 메소드 학습
    - 컴포넌트 내의 적절한 곳에 API와 메소드들을 배치
  - 권한 : 음성 권한을 부여하는 기능 구현 문제
    - 아고라 슬랙 커뮤니티에서 비슷한 기능에 관심이 있는 다른 유저들의 질문들을 참고
    - 아고라에서 지원하는 RTM 기능을 활용
    - 음성 권한을 부여할 때 해당 게스트에게 메시지를 전송하여 음성 스트리밍 기능을 활성화
    - 라이브에 참여한 모든 유저들이 해당 게스트가 음성 권한을 획득했음을 인식.

Q. React를 사용하셨는데, 상태 관리는 어떻게 하셨나요?

### Redux

- “같은 코드가 여러 컴포넌트에서 중복되지 않게 하자!”
- 컴포넌트의 depth가 깊고 서로 유기적으로 결합되어있는 데이터를 한 페이지에 그려야하기 때문에 전역적인 state 관리가 필요
  - Flux 디자인 패턴의 단방향 데이터 흐름을 활용한 Redux 라이브러리로 State를 관리하기로 결정. 러닝커브가 높지만 컴포넌트의 depth가 깊어짐에 따라 props 전달이 어려워 Redux를 도입
  - Redux 이점
    - (1) 컴포넌트 내부에서 호출했던 Api를 분리시킴으로써 비즈니스 로직을 컴포넌트와 분리 가능
    - (2) 서로 떨어져있는 컴포넌트 사이에서 전역적으로 state 를 사용하는 것이 가능
    - (3) 같은 api를 여러 컴포넌트에서 호출해야 할 때 전역적으로 관리 -> 코드의 중복 최소화

Q. 실제 사용자를 만나 보셨나요?

“정글 Ask Me Anything 진행”

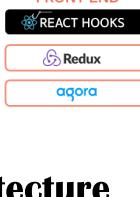


접속자 16명

- 사용성에 대한 피드백 및 UI/UX의 개선할 점 확인
  - 발언자의 Avatar 주변에 음성이 송출되고 있다는 점을 알 수 있도록 음파 추가
  - 텍스트/음성 질문 영역을 보다 직관적으로 수정 등



FRONT END

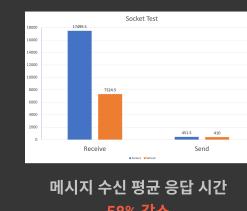


BACK END



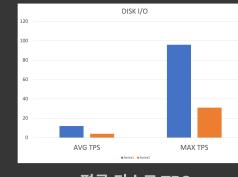
## Architecture

열심히 일하겠습니다. 😊



메시지 수신 평균 응답 시간

58% 감소



평균 디스크 TPS

67% 감소

“100명의 유저를 버텨보자!”

- REST API, WebSocket 기반 챕팅 구현 완료 후 부하 테스트를 위해 Locust 활용
- 가상의 유저 100명이 서비스를 이용하는 것을 가정 <로그인 → 세션 조회 → ... → 라이브 진입 & 이탈> 전체 플로우를 스크립트화
- 테스트 후 부하 최소화를 위해 서버 투닝
  - Gunicorn 워커 개수 증가
  - Nginx Connection 늘리기
  - EC2 스케일업

“DB I/O를 최소화하자.”

- 기존 챕팅 방식
  - (1) 브라우저가 보낸 메시지를 DB에 저장
  - (2) 전체 메세지를 DB에서 다시 조회해서 챕팅 그룹에 Broadcast

#### - 문제점

- 많은 양의 메시지가 전송될 경우 DB I/O Time이 늘어나면서 보낸 메세지와 받은 메세지의 Latency ↑

#### - 개선

- 메세지 데이터를 저장할 Cache Memory로 Redis 활용
- 브라우저에서 메시지가 전송되면 서버에서는 받은 메시지를 그대로 다시 Broadcast
- 유저가 챕팅방 첫 진입 시 Redis에 메세지가 존재하면 MySQL을 안 거치고 전달
- Redis에 데이터가 없을 경우에만 DB에서 데이터를 조회하여 전달

디자이너 한수영 님과의 협업이 진행된 프로젝트입니다.

