

# PORTFOLIO

기술을 쌓아올리는 테트리스형 개발자

김현수

jjc5989@naver.com

+82 010-2420-5989



GitHub



# Kim Hyeonsu

개발자 | 김현수



**Birth.** 1998.10.30

**E-mail.** jjc5989@naver.com

**Tel.** 010.2420.5989

나를 한 마디로  
표현한다면?

흡수는 스펜지처럼,  
쌓기는 테트리스처럼.  
실전에 강한 신입 개발자입니다.

## 학력사항

- 2014.03~2017.02 금호고등학교 졸업  
2017.03~2025.08 고려대학교 세종캠퍼스 졸업예정

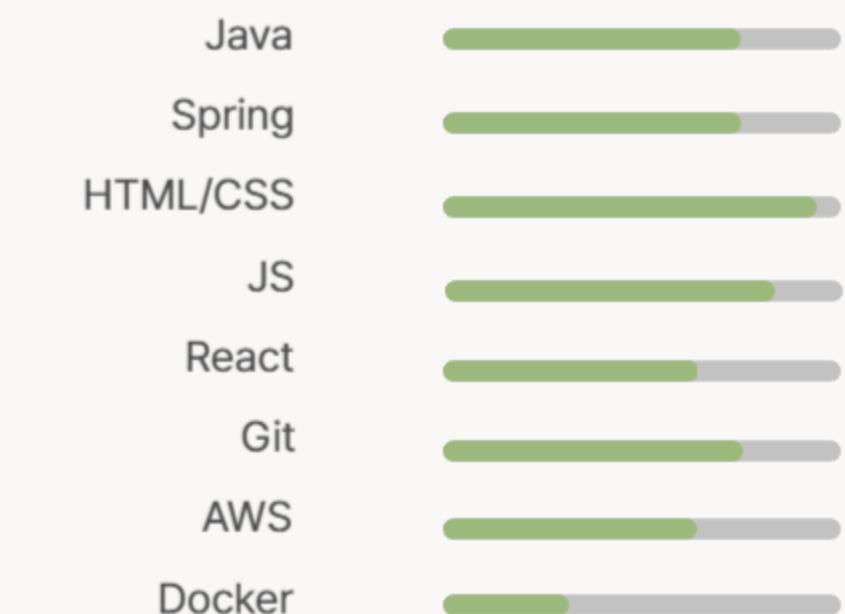
## 경력사항

- 2020.12.17~2022.01.18 IMC큐브(AE)

## 자격사항

- 2024.09.10 정보처리기사  
2021.10.01 SQLD  
2020.12.22 ADsP

## 보유스킬



## 어학사항

- 2024.09.08 TOEIC SPEAKING(IH)  
2023.10.29 TOEIC(880)

# CONTENTS

## SWS

01

맞벌이 부부를 위한 자녀들의 실시간 위치를  
제공하는 등하원 시스템



## BidCast

02

실시간 스트리밍 서비스를 겸비한 경매 플랫  
폼



# 01. SWS(SAFETY WAY SEOUL)

## 프로젝트 개요

기간: 2025.04 - 2025.05

SWS는 서울시의 어린이집 통학버스를 이용하는 자녀들의 부모들을 위해  
자녀의 실시간 위치를 확인할 수 있도록 도와주는 위치 기반 모바일 앱입니다.

실시간 위치 공유, 통학버스 운행상태, 어린이집 정보 등 실용성과 안정성을  
중시한 프로젝트입니다.

## 역할 - 팀장

- 팀원 간 업무 분담 및 일정 관리
- 프로젝트 기획 및 전체 서비스 구조 설계 주도
- ERD 설계 및 DB구조 정의
- 사용자 인증, 쿠키 기반 로그인 등 공통 기능 구현
- 실시간 좌표 전송 및 지도 위 위치 표시 기능 구현
- AWS를 활용한 서버 배포 및 도메인 연결



### 핵심 기술

SpringBoot, WebSocket, Mybatis, Mysql

### 보조 기술

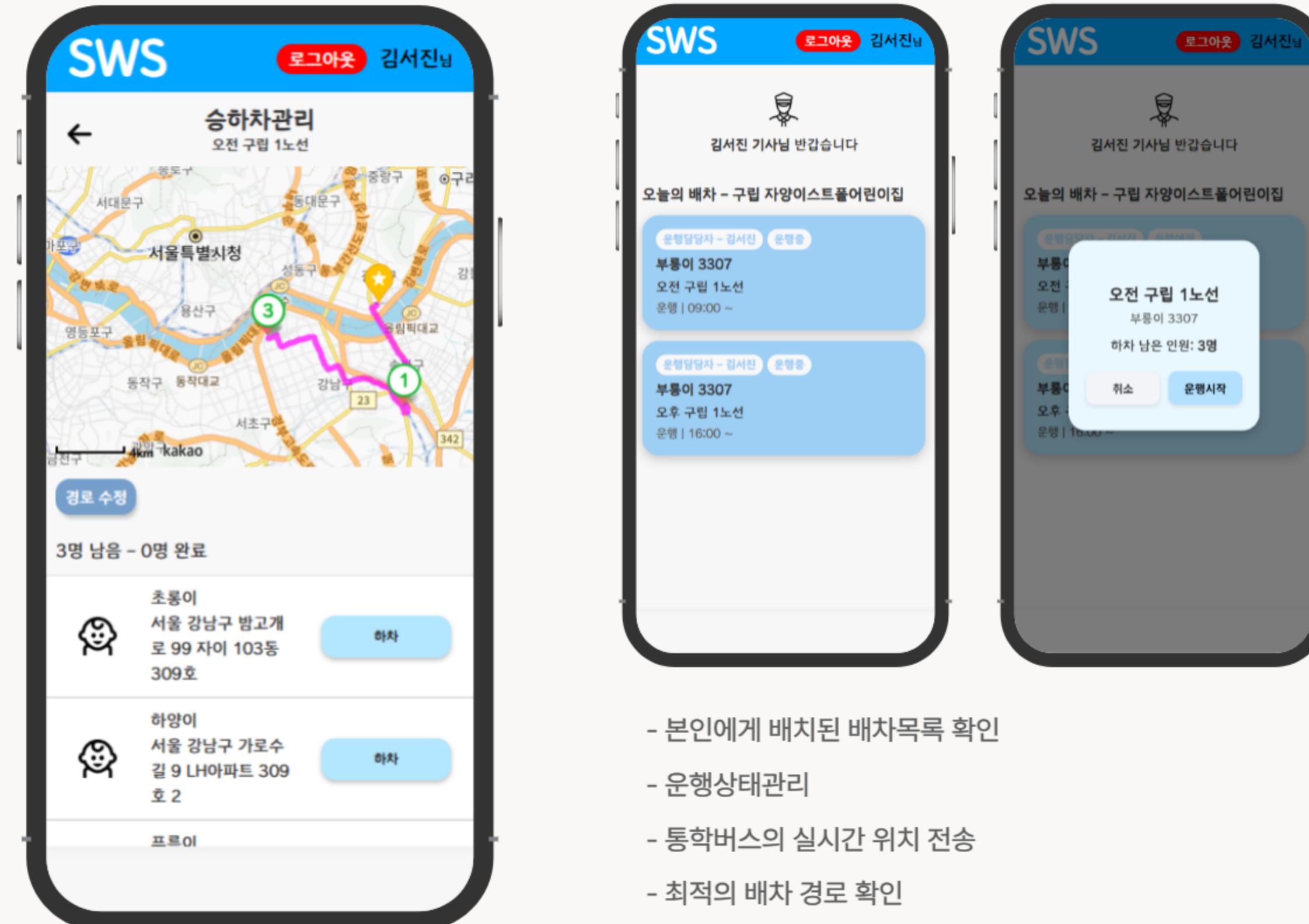
KakaoMap API, Spring Scheduler

### 인프라 · 협업 도구

AWS EC2, S3, Aurora RDS, Nginx, Github

# 핵심기능

기사앱

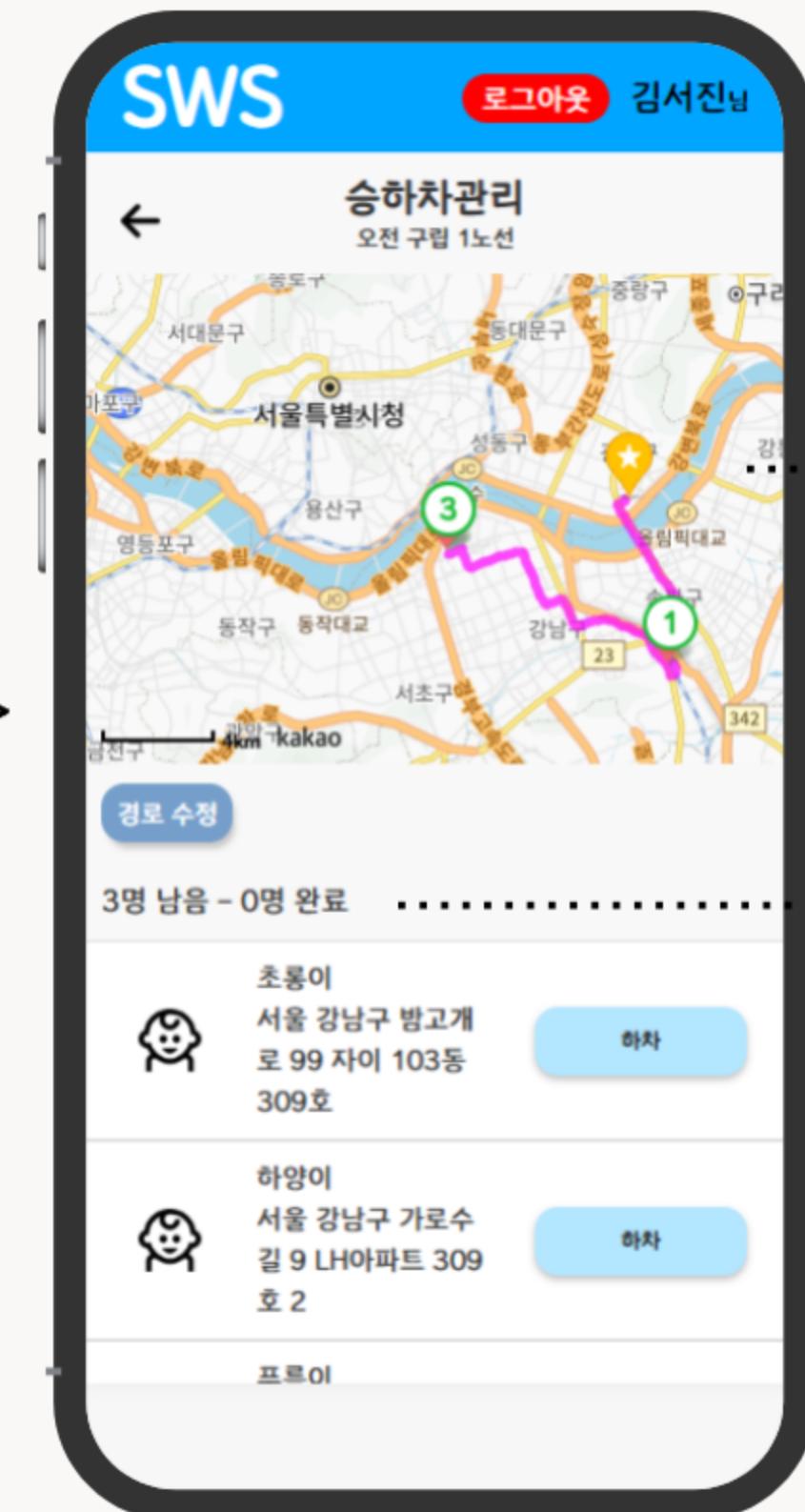


# 핵심기능

기사앱



어린이집 통학버스의  
실시간 위치 전송 시작

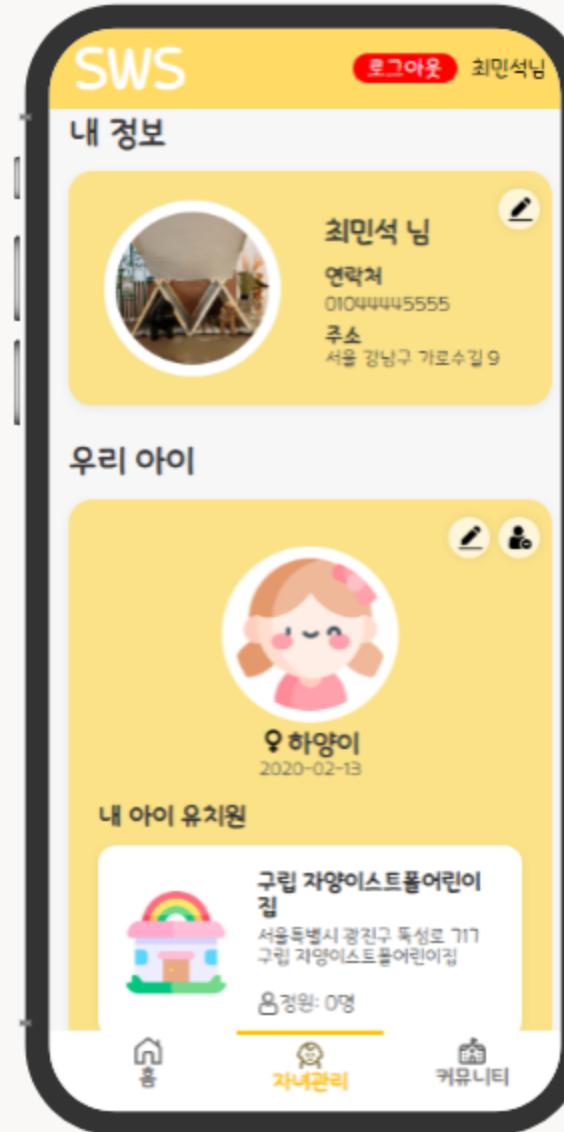
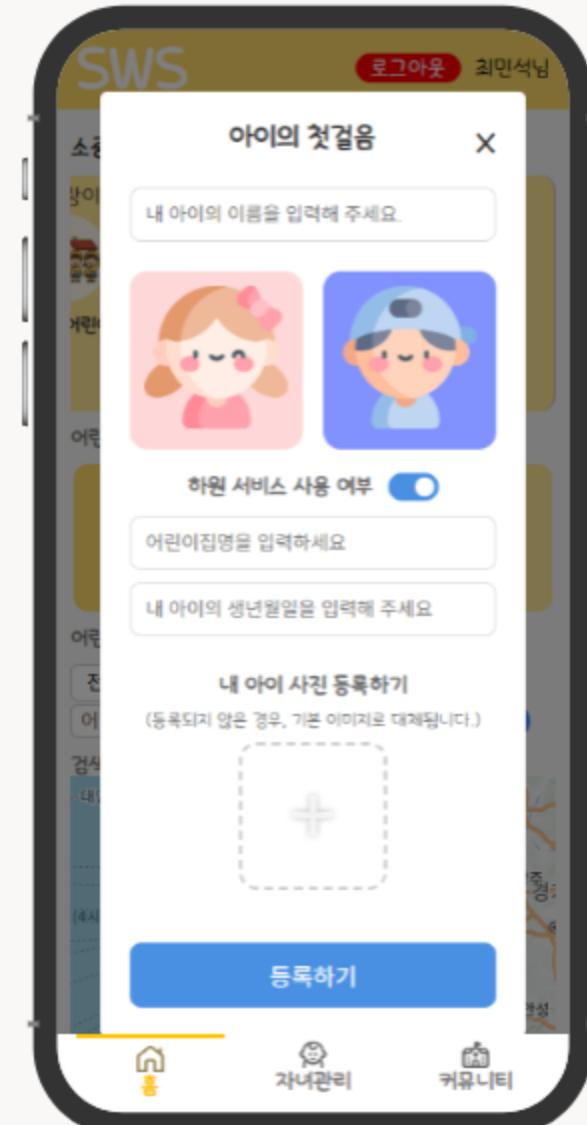
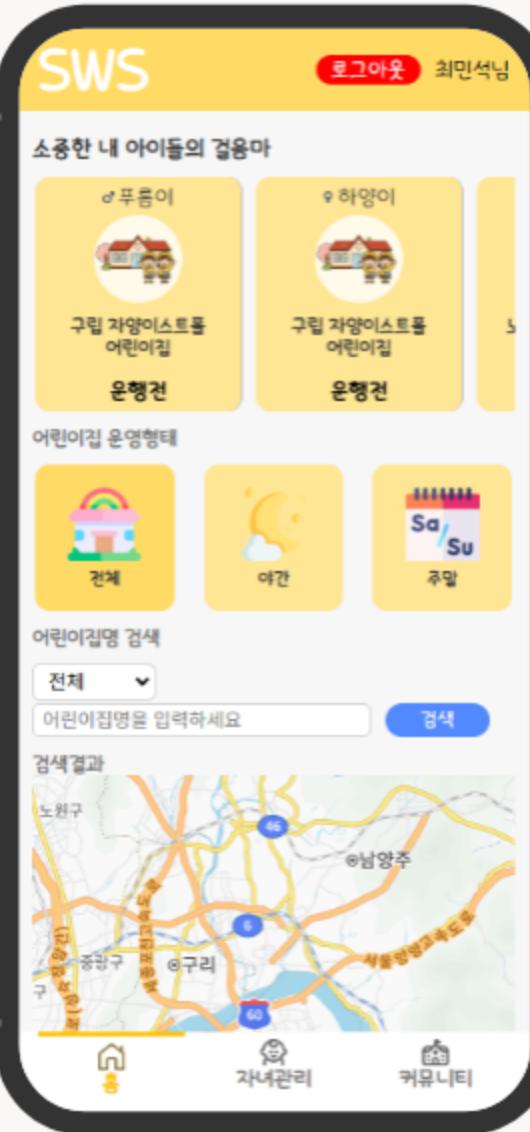
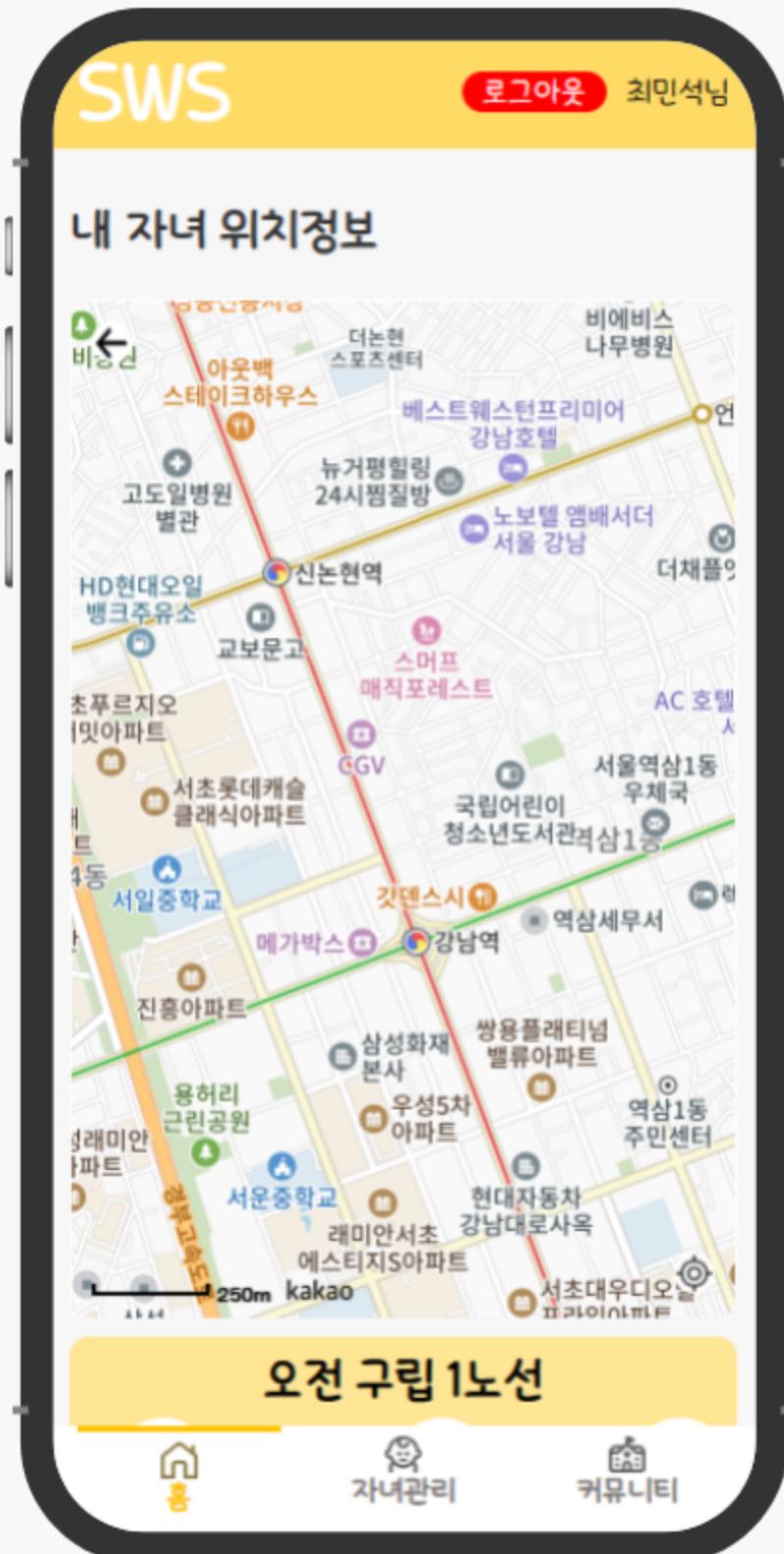


▶ 하차 최적경로 안내

▶ 어린이 하차 상태 관리

# 핵심기능

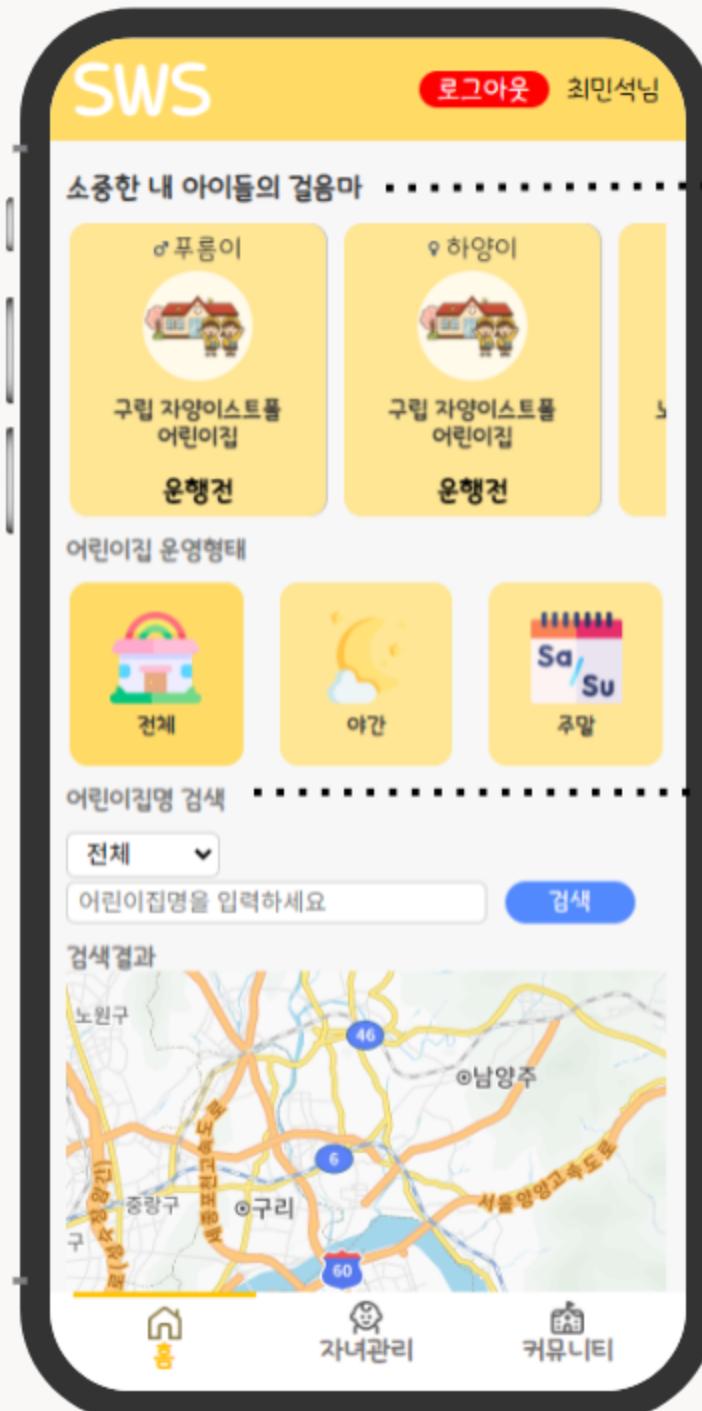
유저앱



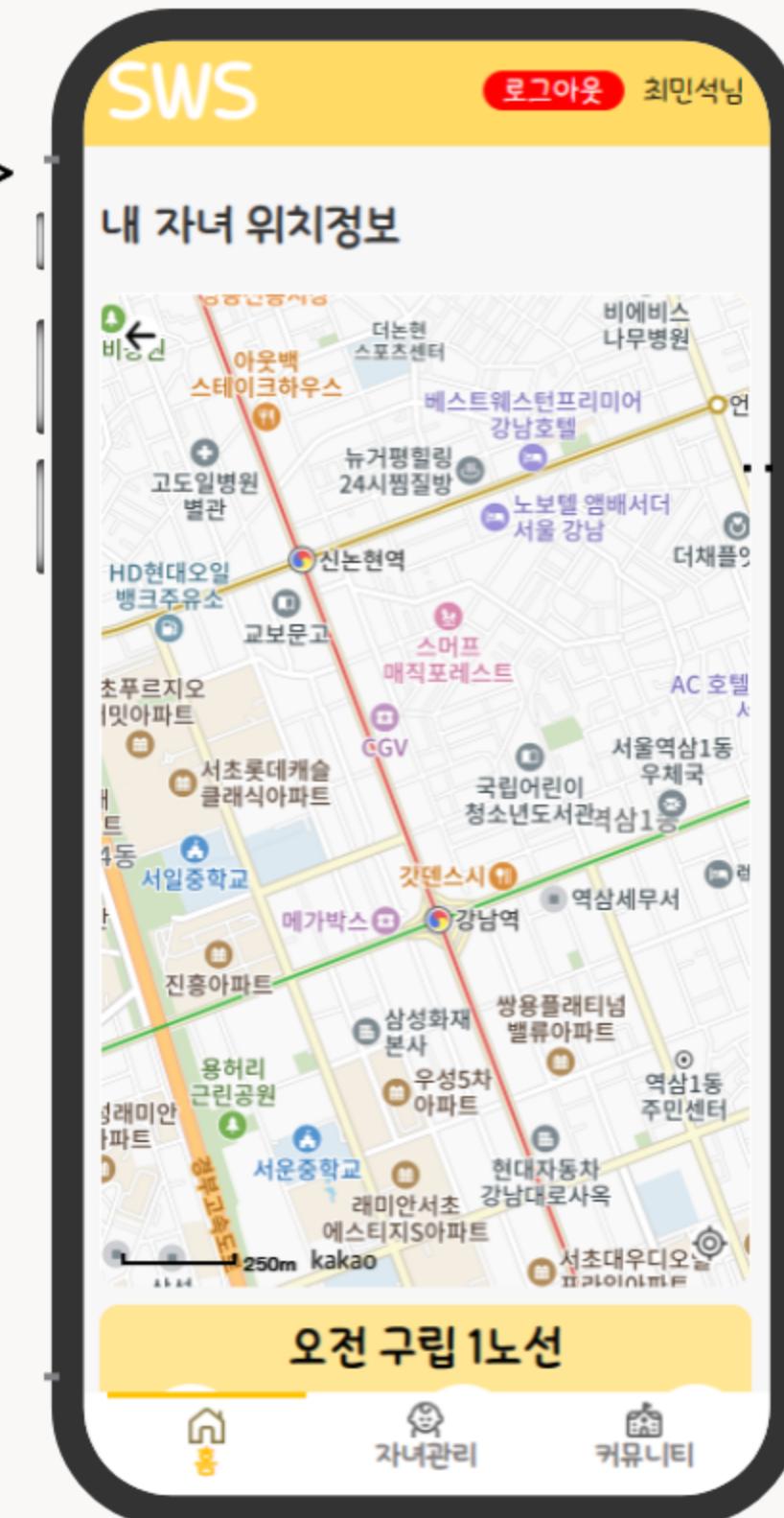
- 어린이집 통학버스의 실시간 위치 확인
- 내 자녀의 하차여부 상태 확인
- 어린이집 검색 및 위치확인
- 내 자녀 등록
- 내 정보 및 자녀 정보 관리

# 핵심기능

유저앱



내 자녀 하차 상태 확인



어린이집 통학버스 실시간 위치 확인

유형별 어린이집 검색 기능

# 기술구현 포인트

```
// ✅ 매일 새벽 0시에 실행
@Scheduled(cron = "0 0 0 * * *") at 00:00 ± KR-HS
public void scheduledUpdate() {
    System.out.println("⌚ 주기적 Kinder 데이터 수집 시작");
    registKinderAPI();
}
```

## 어린이집 정보 업데이트 자동화

- Spring Scheduler를 활용해 매일 0시에 서울시 어린이집 API로부터 최신 데이터를 자동 수집하고, 기존 데이터와 비교해 변경된 정보만 업데이트되도록 구현

```
if (!closed.isBlank() || zipcode.isBlank() || phone.isBlank()) continue;
if(kinderMapper.existsByLaAndLo(latitude,longitude)>0) continue;
```

## 실시간 위치 정보 맵 구현

- 웹소켓을 이용하여 기사 애플리케이션에서 전송한 위치 정보(위도, 경도)를 중계 서버가 수신하여 DB에 저장한 뒤, 이를 유저 애플리케이션에 실시간으로 전달

- 유저 애플리케이션은 최초에 DB에 저장된 기존 경로 데이터를 리스트 형식으로 불러오고, 이후 실시간으로 수신된 위치 정보를 이어붙여 지도에 경로를 시각화

```
const socket : SockJS = new SockJS("https://swssocket.kro.kr/wss");
const stompClient = Stomp.over(socket);
stompClient.connect({}, () => {...});
```

```
public void sendLocation(LocationVO location) { 1 usage ± KR-HS *
    if (!isDriving || location.getRecordKey() != this.recordKey) return;
    restTemplate.postForObject(url: "https://swssocket.kro.kr/location/update",
        location, Void.class);
```

```
@Override no usages ± KR-HS
public void registerStompEndpoints(StompEndpointRegistry registry) {
    registry.addEndpoint(paths: "/wss").setAllowedOriginPatterns("*").withSockJS();}
```

# 트러블슈팅 및 해결방법

## 01. 중계 서버를 통한 실시간 위치 공유

### 01 문제 상황

기사 앱과 유저 앱이 서로 실시간 위치를 직접 주고받도록 구현했으나, 기사 앱에서 위치 전송은 정상적이었으나 사용자 앱에서 지도 경로가 갱신되지 않는 문제 발생

### 02 문제 원인

직접 연결 방식으로 진행했으나, 네트워크 환경 제약뿐 아니라 프로토콜, 포트, CORS 등으로 인해 안정적인 통신 실패

### 03 해결방안

WebSocket 중계 서버 구조를 도입하여, 서버가 사용자 간 매칭과 위치 라우팅을 담당하도록 설계

이를 통해 NAT, CORS 등 네트워크 제약을 극복하고 실시간성과 안정성을 확보

```
// 2. recordKey 채널에 실시간 브로드캐스트 (UserApp이 구독)
messagingTemplate.convertAndSend(
    destination: "/topic/location/update/" + location.getRecordKey(),
    location
);
```

# 트러블슈팅 및 해결방법

## 02. 중계서버의 프로토콜 불일치 문제

### 01 문제 상황

브라우저에서 혼합 컨텐츠 오류로 WebSocket연결이 차단되어 실시간 데이터 통신 실패

### 02 문제 원인

기사 서버와 유저 서버는 HTTPS로 정상 운영되었으나, WebSocket통신은 비보안 프로토콜(ws://) 사용으로 인한 보안 정책 충돌

### 03 해결방안

- WebSocket 서버를 wss://프로토콜로 전환
- 클라이언트 WebSocket 연결 URL도 wss://로 변경하여 보안 연결 유지

그 결과, WebSocket 통신 정상화 및 실시간 데이터 송수신 안정화

```
// WebSocket 연결 URL을 보안 프로토콜 wss://로 변경
const socket : SockJS = new SockJS("https://swssocket.kro.kr/wss");
const stompClient = Stomp.over(socket);
stompClient.connect({}, () => {
    console.log("WebSocket connected");
    stompClient.subscribe(`topic/location/update/${recordKey}`, message => {...});
});
```

## 02. BIDCAST

### 프로젝트 개요

기간 : 2025.06 - 2025.06

BIDCAST는 누구나 경매를 개설하고 입찰에 참여할 수 있는, 실시간 화면 공유 기능을 포함한 영상 기반 웹 경매 플랫폼입니다. 입찰, 유찰, 낙찰 등 실제 경매 절차를 통해 협실감 있는 경매 환경을 제공하며, 실시간 채팅을 통해 참여자 간 소통과 반응을 즉각적으로 주고받을 수 있도록 구성했습니다.

### 역할 - 팀장

- 팀원 간 업무 분담 및 일정 관리
- 프로젝트 기획 및 전체 아키텍쳐 설계 주도
- ERD 설계 및 DB구조 정의
- 실시간 영상 송출 및 입찰 인터페이스 구현
- 채팅 및 경매 상태 변경(입찰, 낙찰, 유찰) 기능 구현
- AWS를 활용한 배포 및 도메인 연결
- JENKINS 기반 CI/CD 파이프라인 구축 및 자동 배포 설정



### 핵심 기술

React, SpringBoot, PostgreSQL, Mediasoup, Socket.IO

### 보조 기술

MUI, Vite, Spinner

### 인프라 · 협업 도구

AWS EC2, S3, Aurora RDS, Nginx, Jenkins, Github

# 핵심기능

게스트

경매 참여자수



소개

경매

고객센터

참여자목록 1명 시청중

화면공유 인원이 없습니다.

검색

찜

돌이

이동

실시간 채팅

실시간 채팅기능

게스트의 화면  
(화면 공유 시)

지나가는나그  
네

채팅입니다

경매 태그

현재 최고가 및 최고입찰자

현재최고가: 0원  
최고입찰자:

태그

액세서리 쥬얼리

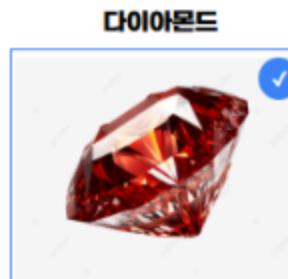
채팅을 입력해주세요

채팅

경매 종료

호스트 영상 송출/중단

경매물품 목록



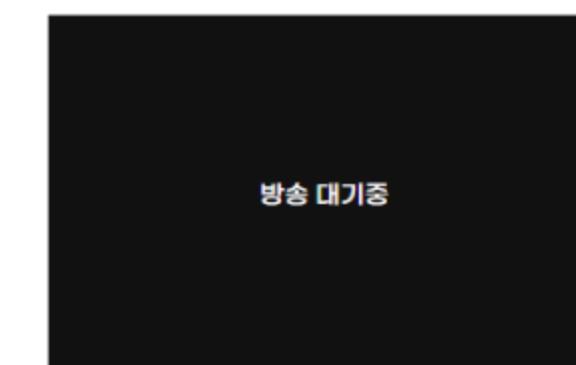
경매물품 관리

1. 경매물품선택

2. 낙찰 / 유찰

3. 경매단위 변경

- 선택
- 낙찰
- 유찰
- 경매 단위 변경



방송 대기중

호스트 영상화면

# 핵심기능

[호스트](#)

소개

경매

고객센터



현재 경매중인 물품명 &lt;-----

매물명: -

현재최고가: 0원

실시간 채팅 -----&gt;

실시간 채팅기능

방송이 중단되었습니다.

호스트 화면 &lt;-----

게스트 화면

손님 화면 송출

게스트 화면 리스트 <-----  
(화면 공유)

게스트 화면 송출/중단 -----&gt;

화면공유 인원이 없습니다.

경매 타이틀 <-----  
경매 참여자수 <-----

경매번호3306

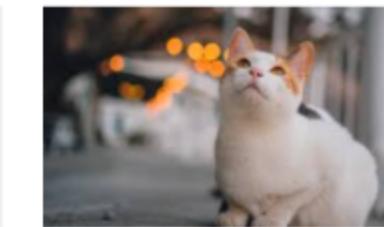
1명 시청중 가구 부동산

입찰 | 0원

채팅을 입력해주세요

채팅

다른 경매 리스트 &lt;-----



증수 예어웃습니다

주얼리물품 경매

경매번호 3307

경매물품 입찰 기능 -----&gt;

# 기술구현 포인트

## 실시간 화면 공유

웹소켓과 WebRTC(mediasoup)를 이용하여 1명의 방송자가 다수의 시청자에게 실시간 화면을 공유할 수 있는 1:N 방송 구조를 구현

- SFU방식을 활용해 방송자는 하나의 영상만 전송하고, 서버가 이를 다수의 시청자에게 효율적으로 중계

```
const rooms = Array.from(socket.rooms).filter(r => r !== socket.id)
rooms.forEach(room => {
  socket.broadcast.to(room).emit('new-producer', {
    producerId: producer.id,
    socketId: socket.id,
    // auctionId: roomId, // 방 정보 추가
    kind
  })
})
```

## 경매 기능

웹소켓을 통해 경매데이터를 실시간 송수신하고, 사용자 인터페이스와 경매로직의 상태를 동기화

- 서버에서 경매 상태 통합관리 : 선택된 물품, 입찰자 등
- 최고입찰자 변경, 입찰 실패, 유찰 처리 등 예외 사항도 고려하여 상태 변경 및 알람 구현

```
// 4) 모든 클라이언트에게 변경사항 방송
io.to(auctionId).emit("bid-update", {
  product: updatedProduct,
  bidder: {
    userKey: winnerId,
    nickname: winnerNickname,
    // 추가 정보 필요시 여기에
  }
});
```

# 트러블슈팅 및 해결방법

## 01 Mediasoup SFU 다중 스트림 동시 처리 문제

```
setSocketIdToProducerId( value: prev :{}  => {
  const existing = prev[socket.current.id] || {}
  // console.log("이미 가지고 있는 것들", existing)
  return {
    ...prev,
    [socket.current.id]: {
      ...existing,
      [kind]: id
    }
  };
});
```

### 문제 상황

여러 참가자 있는 상황에 한 명의 영상만 보이고, 다중 스트림이 정상적으로 렌더링 되지 않음

### 문제 원인

- Producer와 Consumer간 매핑이 잘못되어 스트림이 갱신되지 않음
- React State 업데이트가 비효율적이어서 UI가 새 스트림을 제대로 반영하지 못 함

### 해결 방안

- Peers 객체를 활용해 참가자별 Producer / Consumer 상태를 명확히 관리함
- SocketIdToProducerId 맵핑으로 올바른 스트림 연결 보장

# 트러블슈팅 및 해결방법

## 02 연결 종료 및 재접속 시 자원 정리 문제

```
// 해당 소켓의 모든 Transport 닫고 삭제
for (const [transportId, data] of transports) {
  if (data.socketId === socket.id) {
    data.transport.close()
    transports.delete(transportId)
  }
}

// 해당 소켓의 모든 Producer 닫고 삭제
for (const [roomId, roomProducers] of producers) {
  for (const [producerId, data] of roomProducers) {
    if (data.socketId === socket.id) {
      data.producer.close()
      roomProducers.delete(producerId)
      io.emit('user-disconnected', { ...
      })
      console.log(`Producer closed manually: ${producerId}`)
    }
  }
  if (roomProducers.size === 0) { ...
  }
}
```

### 문제 상황

사용자가 퇴장하거나 네트워크 끊김 시 미디어 트랙, Transport등이 제대로 정리되지 않아 서버 자원이 누적됨

### 문제 원인

- 서버의 disconnect 이벤트에서 Producer / Consumer, Transport 자원을 완전히 해제하지 않음
- 클라이언트도 미디어 트랙 중지 및 상태 초기화가 제대로 되지 않음

### 해결 방안

- 서버에서 disconnect 이벤트 발생 시 관련 모든 Producer, Consumer, Transport를 명확히 삭제
- 클라이언트는 미디어 트랙을 중지시키고 React 상태 초기화 수행

# 트러블슈팅 및 해결방법

## 03 실시간 입찰 및 상태 업데이트 동기화 문제

```
// 다른 유저들에게 방송
socket.to(auctionId).emit("bid-status", {
  prodKey,
  winner,
  nickname,
  status
});
```

```
// 4) 모든 클라이언트에게 변경사항 방송
io.to(auctionId).emit("bid-update", {
  product: updatedProduct,
  bidder: {
    userKey: winnerId,
    nickname: winnerNickname,
    // 추가 정보 필요시 여기에
  }
});
```

### 문제 상황

호스트와 참가자 간 입찰 데이터가 실시간으로 불일치하고, 낙찰 / 유찰 상태 알림이 지연됨

### 문제 원인

- WEBSOCKET 이벤트가 제대로 브로드캐스트 되지 않거나, 클라이언트 UI 갱신이 늦음

### 해결 방안

- 서버에서 입찰 상태가 바뀔 때 즉시 관련 이벤트를 모든 클라이언트에 송출
- 클라이언트는 이벤트 수신 시 바로 상태를 갱신하고 UI에 반영

# 감사합니다.

저는 더 나은 사용자 경험과 안정적인 시스템을 위해 끊임없이 배우고,  
꾸준히 실천하는 개발자가 되겠습니다.

김현수

jjc5989@naver.com

+82 010-2420-5989

<https://kr-hs.github.io/Fullstack-Study-241204-250625/>

