banghyunmin1999@gmail.com

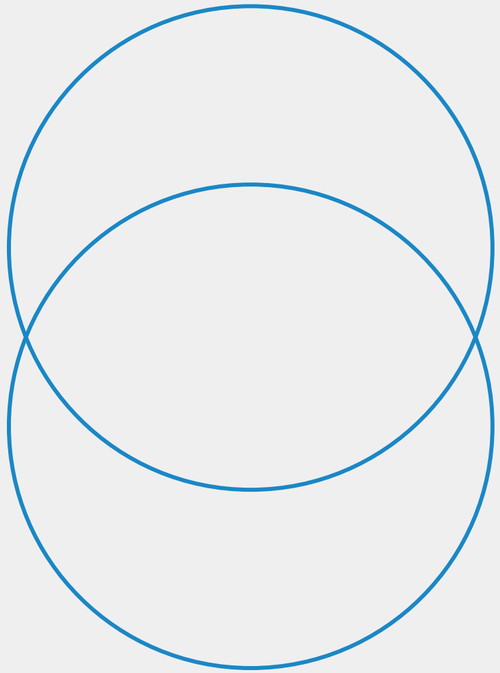
Solidity

스마트 컨트랙트 개선 과제 발표

2025.05

스마트금융과 방현민

INDEX



백엔드 개발 담당: 방현민

프론트엔드 개발 담당: 김진모

개요

Solidity 기반 경매 스마트 컨트랙트 개발

Remix IDE를 활용한 브라우저 기반 개발 및 테스트

JavaScript VM 환경에서 로컬 블록체인 시뮬레이션 수행

입찰, 낙찰자 판단, 출금, 경매 종료 기능 포함

관리자 전용 기능과 일반 사용자 구분 구현

struct, mapping, modifier 등 Solidity 주요 기능 활용

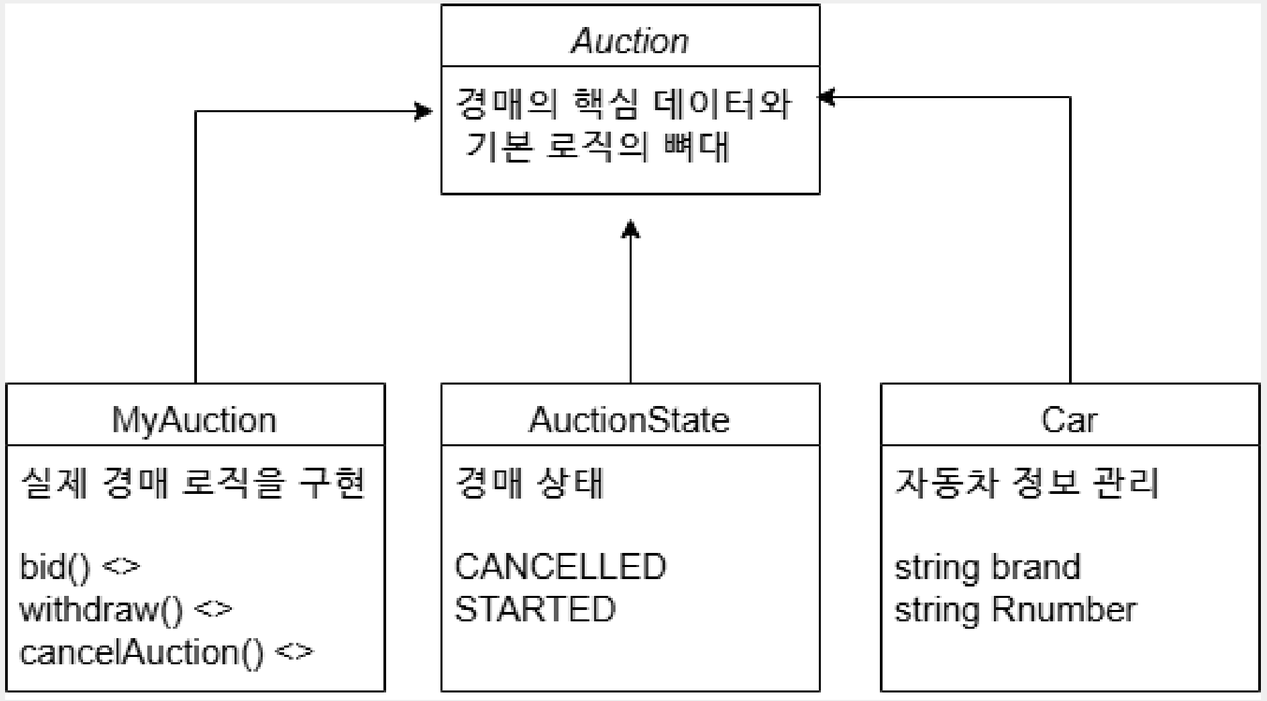
프론트엔드는 팀원(김진모) 담당

🛠 개발 및 운영 가이드 및 사용 기술

항목	내용
개발 언어	Solidity 0.8.0
개발 도구	Remix IDE (브라우저 기반)
실행 환경	JavaScript VM (내장 블록체인)
테스트 방식	컨트랙트 직접 배포 및 실행
프론트엔드 구현	React 기반 DApp (김진모)

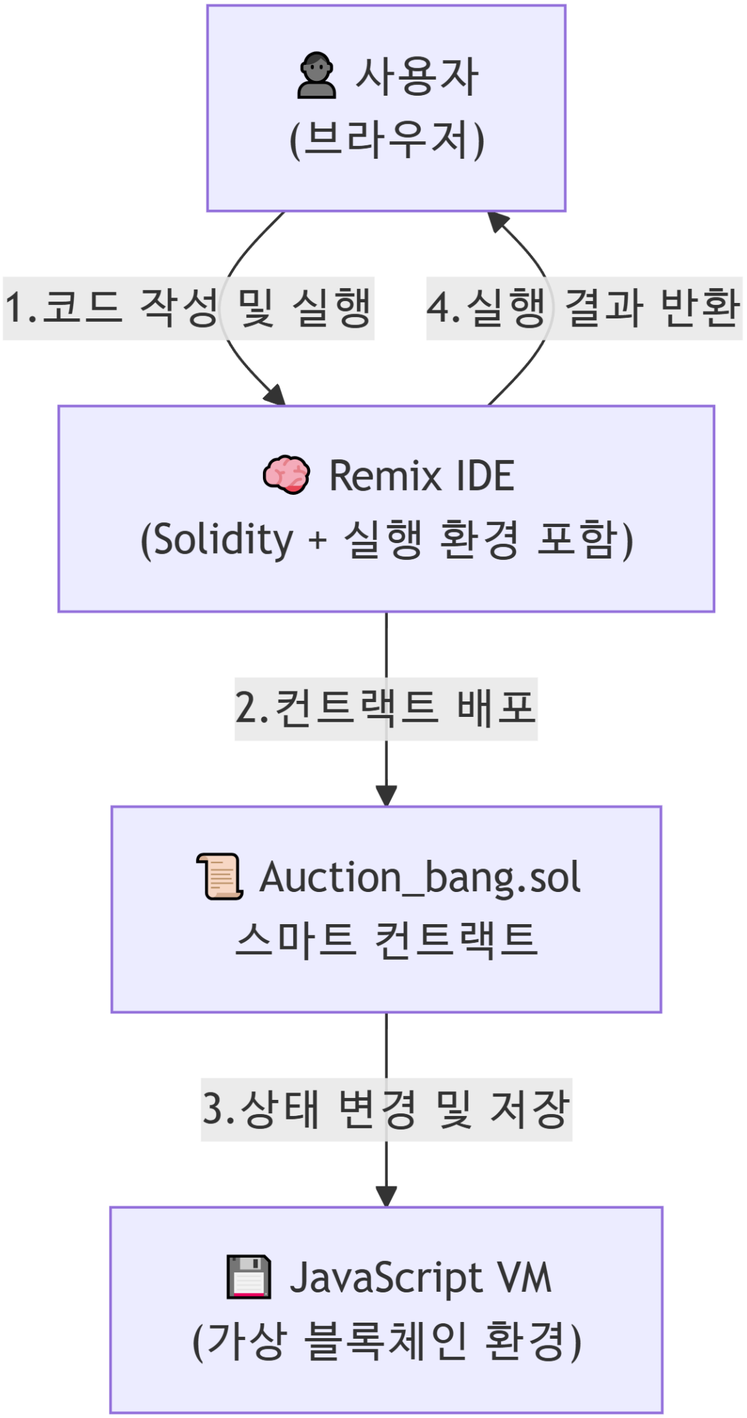
항목	설명
경매 시작	startAuction() 호출 (소유주만 가능)→ 경매 기간 및 차량 정보 입력 필요
입찰	bid() 함수 호출→ ETH 전송 포함→ 최고가보다 높은 경우만 입찰 반영됨
입찰금 회수	withdraw() 함수 호출→ 이전 최고입찰자가 입찰금 반환 가능
경매 종료	endAuction() 함수 호출→ 경매 마감 시간 이후 소유주가 호출하여 입찰금 수령 가능
경매 중단	cancelAuction() 호출 (소유주만 가능)→ 경매 상태를 CANCELLED로 변경
이벤트 로깅	emit Start(), emit Bid(), emit End() 등 로그 발생→ Web3.js에서 추적 가능
보안 관리	onlyOwner, an_ongoing_auction 등의 modifier로 접근 제어 및 경매 상태 체크
배포 환경	테스트: Remix (JavaScript VM)

아키텍처 구조(BE)


Auction
경매의 핵심 데이터와
기본 로직의 뼈대
MyAuction
실제 경매 로직을 구현
bid() <>
withdraw() <>
cancelAuction() <>
AuctionState
경매 상태
CANCELLED
STARTED
Car
자동차 정보 관리
string brand
string Rnumber

전체 구성도(BE)

구성 요소	설명
사용자 (브라우저)	Remix IDE를 통해 컨트랙트 실행
Remix IDE	Solidity 코드 작성, 컴파일, 배포 및 함수 실행
Auction 스마트 컨트랙트	경매 기능 구현 (입찰, 종료 등)
JavaScript VM	Remix 내장 블록체인, 상태 저장 및 테스트 트랜잭션 처리



@스마트금융과

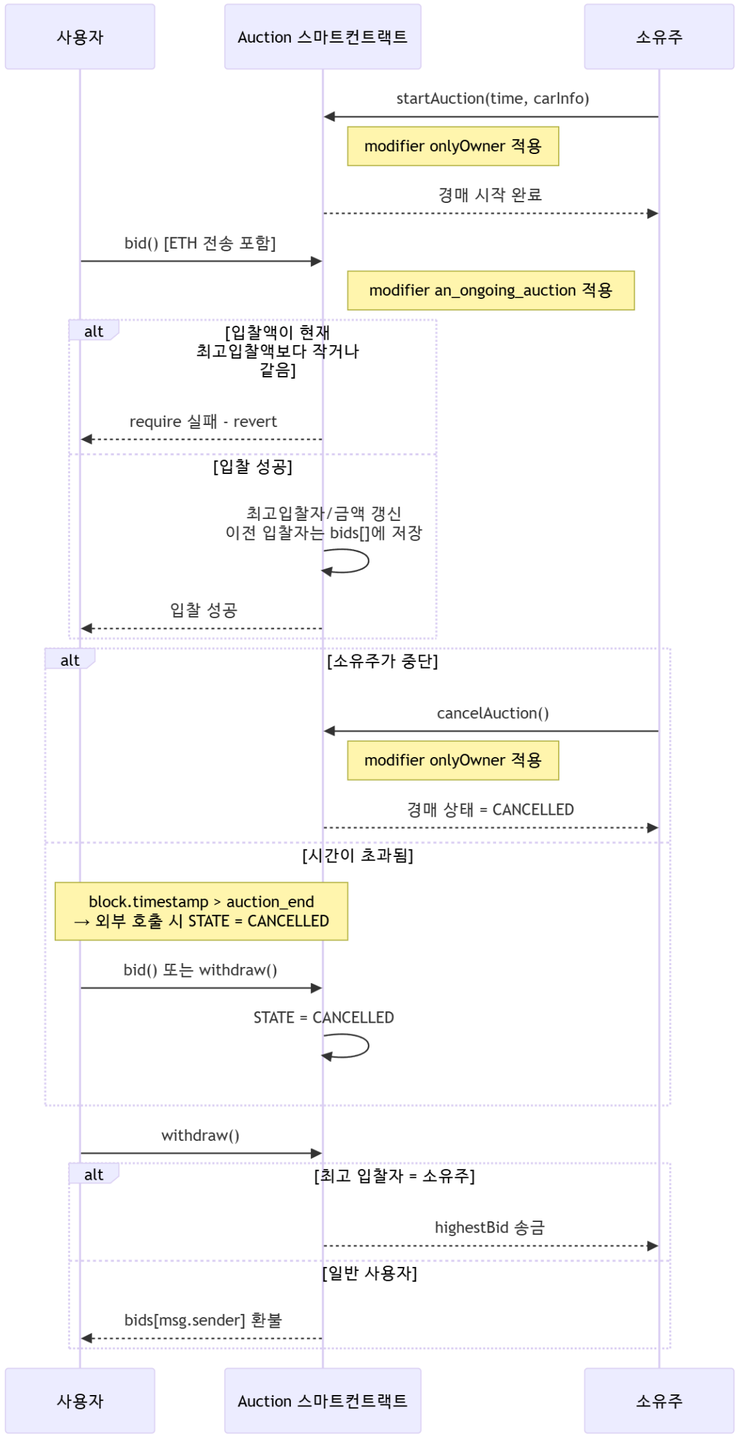
시퀀스 다이어그램(BE)

소유주가 차량 정보와 시간 입력 → 경매 시작

사용자가 입찰기존 최고가보다 낮으면 실패 (revert)높으면 최고가 갱신 및 저장

경매 종료소유주가 직접 종료하거나시간이 지나고 사용자가 함수 호출 시 자동 종료 처리됨

출금소유주는 최고 입찰금 수령나머지 입찰자는 환불가능



주요 함수 설명 및 과제 방향(BE)

코드 내 잠재적 문제점 파악

기존 코드 요소들을 적극 활용

개선 및 기능 강화

문제점 1 - 소유자 자금 인출 제한 부재

분석 대상

경매 소유자가
남은 자금을 회수하는 함수

문제 현상

현재 로직 상, 경매 소유자가
컨트랙트 내 자금을
시점이나 횟수에 관계없이
인출할 가능성 존재

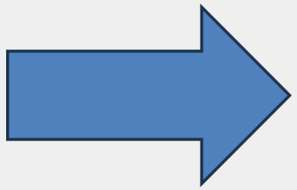
개선 사항

1. 경매 종료 후
2. 최고 입찰자의 입찰 금액만
3. 한 번만 인출 가능하도록 제한

개선 사항: 경매 종료 후

방법	참고	기능
end_auction modifier 신규 정의	기존 '경매 진행 중 확인' modifier 구조 활용	경매가 실제로 종료되었는지 (시간 경과 또는 취소) 확인

```
// 경매가 진행 중인지 확인하는 modifier
modifier an_ongoing_auction() {
    require(block.timestamp <= auction_end, "Auction has ended");
    _;
}
```



```
// 경매가 끝났는지 확인하는 modifier ( 시
modifier end_auction() {
    require(block.timestamp > auction_end
    _;
}
```

개선 사항:최고 입찰자의 입찰 금액만

문제	참고	변경
uint balance = address(this).balance; (컨트랙트 전체 잔액)	기존 코드의 highestBid 변수 활용	인출 금액을 call{value: highestBid}로 수정

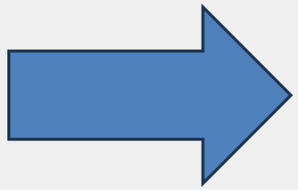
```
(bool success, ) = payable(auction_owner).call{value: balance}("");
```

```
(bool success, ) = payable(auction_owner).call{value: highestBid}("");
```

개선 사항: 단 1회 인출

방법 1	방법 2	방법 3
bool internal isWithdraw = true; 상태 변수 도입	함수 실행 시 isWithdraw = false; 로 설정	require(isWithdraw, ...) 구문을 사용하여 isWithdraw 상태 확인

```
bool internal  isWithdraw = true;
```



withdrawRemainingFunds 함수 내부

```
isWithdraw = false;
```

문제점 2 & 개선 - 최고 입찰자 인출 방지

문제 현상

기존 withdraw 함수 로직 상, 최고 입찰자도 자신의 입찰금 인출 가능

개선 방안

활용: 기존 코드의 highestBidder 상태 변수
방법: 신규 modifier 정의
조건: 함수 호출자가 최고 입찰자가 아니어야 함 (require(msg.sender != highestBidder, ...))
적용: 정의된 modifier를 기존 withdraw 함수에 추가
적용후 최고 입찰자를 제외한 나머지 입찰자들은 입찰한 금액을 인출 가능하며, 추가입찰도 가능

문제점 3 & 개선 - 입찰 로직 수정

문제 현상

최고 입찰 금액과 상관없이
낮은 금액으로 입찰 가능한 문제
(최고 입찰액 갱신 로직 문제)

원인

bid 함수에서 highestBid 갱신 시,
추가 입찰 금액(msg.value)만 반영

개선 사항

기존 입찰 금액(bids[msg.sender]) +
추가 입찰 금액(msg.value)을
합산하여 갱신하도록 변경 (highestBid
= bids[msg.sender] + msg.value;)

문제점 4 & 개선 - 비활성 시 입찰 방지

문제 현상

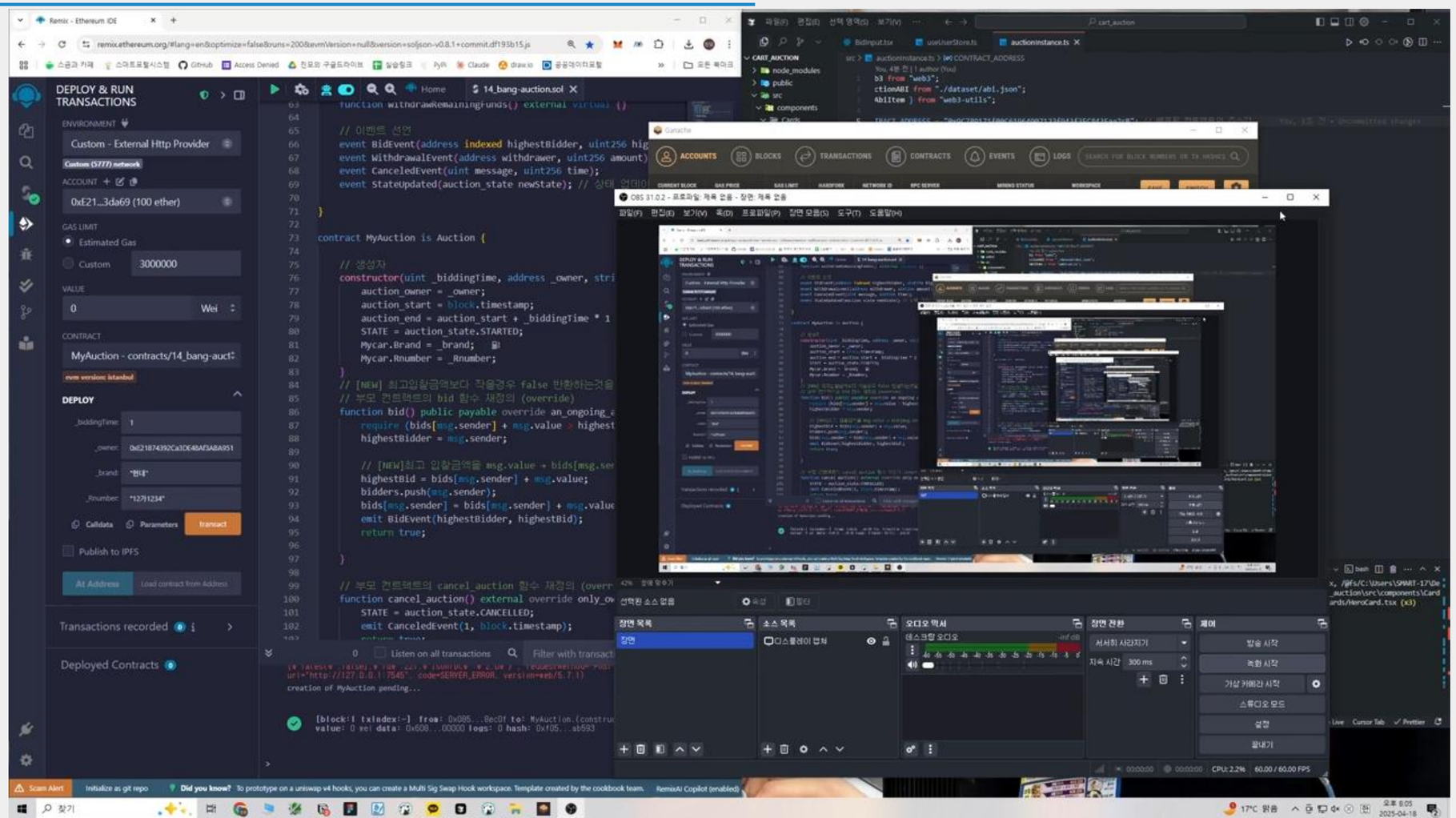
경매가 비활성화된 상태
(CANCELLED)에서도
입찰 가능

개선 방안

활용: 기존 ongoing (경매 진행 중 확인) modifier
신규 end_auction (경매 종료 확인) modifier
기존 상태 변수 (STARTED, CANCELLED)

적용: 각 modifier에 경매 상태(STATE) 확인 로직 추가
(기존코드) ongoing modifier: STARTED 상태 여부 확인 강화
end_auction modifier: 시간 경과 또는 CANCELLED 상태 확인

테스트 결과 및 주요 결과 화면



프론트엔드 개발 담당: 김진모

기존 코드 문제점 분석

모든 로직이 단일 파일에 집중되어 있어 유지보수와 확장성이 떨어졌음

→ 기능별로 디렉토리 구조를 나누는 구조적 개선 필요

.then 체이닝의 남용으로 인해 코드 가독성이 낮았음

→ async/await 기반으로 리팩토링 필요

중요 이벤트 미구현

→ 경매 상태 변화 이벤트, 입찰금 반환 이벤트 등의 생략

비동기 처리 시 예외 상황 미처리

→ 로깅 시 중복 등록, 메모리 누수 가능성 존재

프론트엔드 개발 담당: 김진모

개선 및 기능 강화 내역

역할 기반 코드 구조 분리

→디렉토리 구조: components/, store/, utils/, pages/ 등으로 분리

async/await 리팩토링

→동기적 흐름 확보 및 명확한 예외처리 (throw)

빠진 이벤트 추가 및 메모리 관리 최적화

→상태 변경 시 불필요한 호출 제거 및 이벤트 핸들러 정리

화면 렌더링과 비동기 처리의 싱크 맞춤

→Promise.all로 초기화 작업 병렬 처리

→로딩 상태를 boolean 값으로 관리하여 조건부 렌더링 적용