

따봉캐시

- 블록체인 기반 리뷰 관리 시스템 -

2018007610 안효성

2021072524 장연

CONTENTS

1. 중간 발표내용 요약
2. 서비스 Flow
3. UI 디자인 및 기능 상세
4. 기술 스택 선정
5. 데모 영상
6. 주요 코드 설명

01. 중간 발표내용 요약

문제점

- 악성 소비자 문제:

- 별점 약속 후 댓글 미작성으로 점주 부담 증가

- 시스템 부재:

- 기존 배달 플랫폼은 리뷰 계약 관리 시스템 없음

해결책

블록체인 기반 리뷰 계약 시스템:

- 스마트 계약으로 리뷰 작성 및 보상 절차 자동화
- 가게는 **주문 건수마다 일정 금액**을 지불
- 따봉 수에 따라 토큰 보상 자동 분배

장점

- 먹튀 방지로 점주 부담 감소
- 솔직한 리뷰로 신뢰성 있는 문화 형성
- 공정하고 투명한 생태계 구축

01. 중간 발표내용 요약

목표 시장

주요 고객:

- B2B 고객: 배달 플랫폼(배달의 민족, 요기요, 쿠팡이츠).
- B2C 고객: 소비자(리뷰 작성 및 보상 수령), 점주(신뢰성 있는 리뷰로 평판 개선).

시장 특징:

- 배달의 민족, 요기요, 쿠팡이츠 3사가 약 97%의 점유율을 차지.
- 소비자는 리뷰의 진정성과 보상 시스템에 큰 관심을 가짐.

시장 규모

배달 산업 규모:

- 2024년 약 **25조 원** 예상

잠재 고객 기반:

- 배달 플랫폼 내 활성 점포 수: 약 **50만 개**
- 플랫폼 사용자 수: 약 **3,000만 명** 활성 소비자.

성장 추세:

- 배달 플랫폼 연평균 성장률 약 **10%-15%**.
- 리뷰와 보상 시스템 수요 증가로 인해 새로운 기술 도입 가능성 증대.

경쟁 분석

기존 경쟁자:

- 배달의 민족, 요기요, 쿠팡이츠 등 배달 플랫폼의 내장 리뷰 기능.
- 리뷰 보상 플랫폼(여기어때, 야놀자 등).

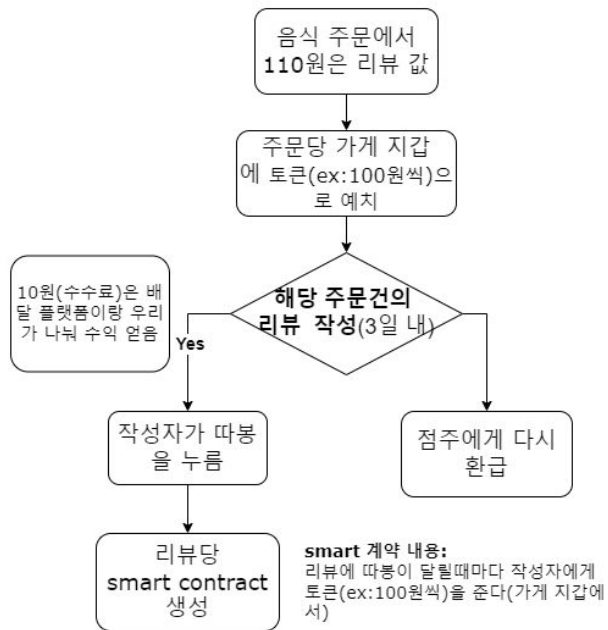
차별화된 강점:

- **블록체인 기술**로 투명성과 변조 방지 실현.
- **따봉 기반 보상 체계**로 공정성과 참여율 확보.
- **데이터 투명성**으로 점주·소비자·플랫폼 간 신뢰 구축.

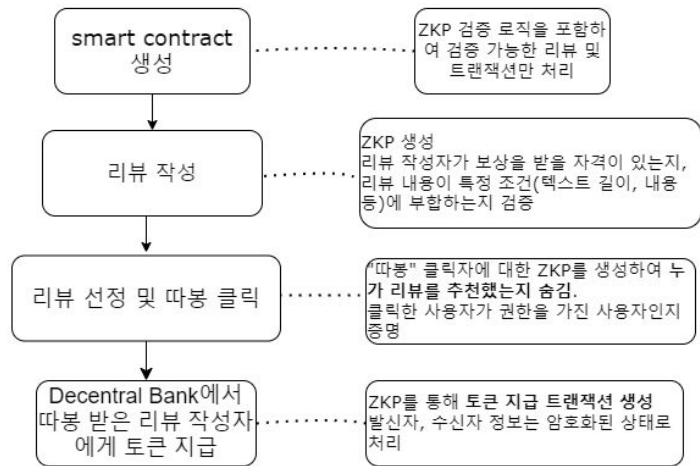
02. 서비스 플로우

Flow Chart

As-Is



To-Be



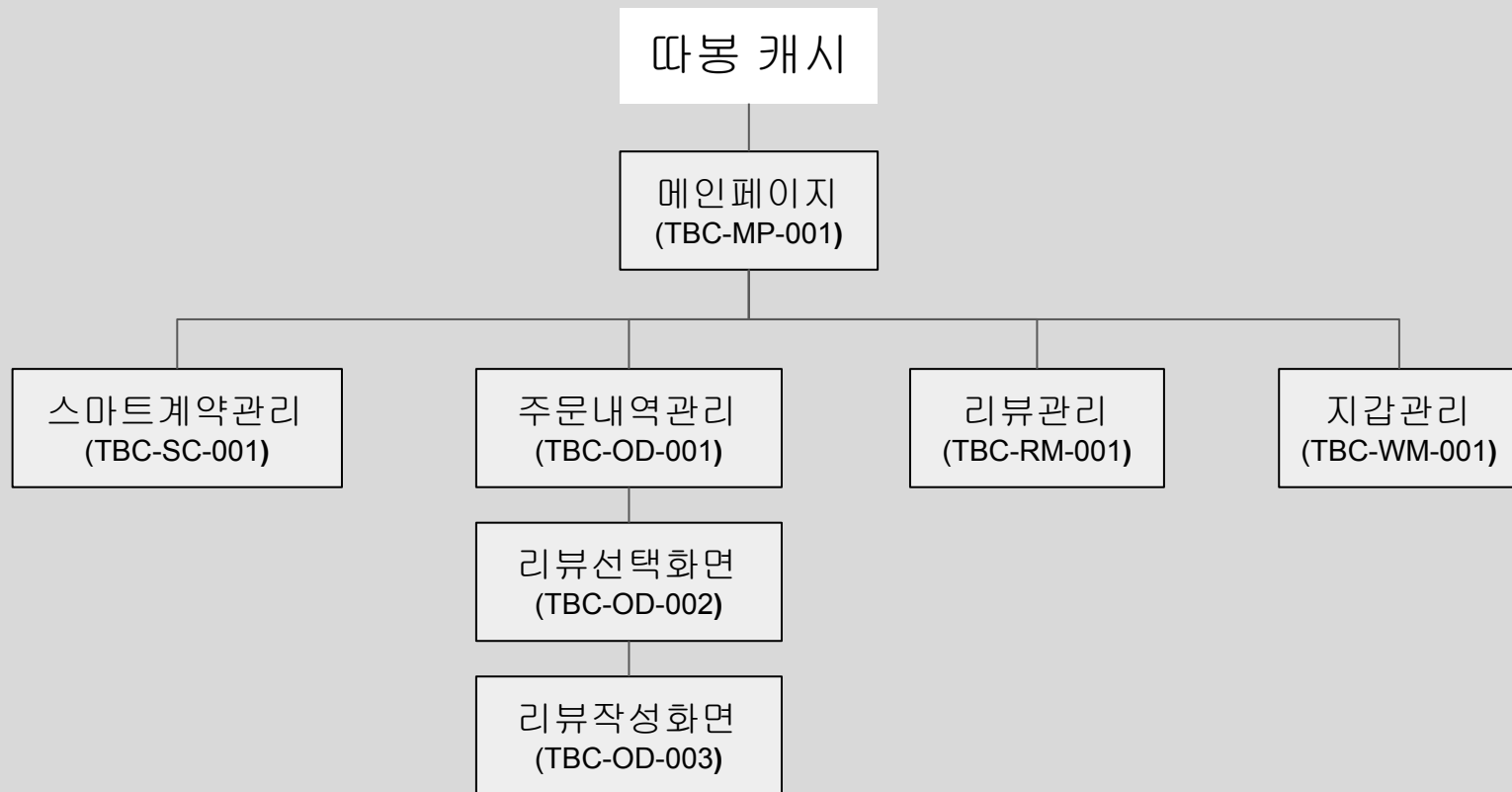
- 음식점 점주는 Bank에 토큰을 예치
- Decentral Bank에서 일정 수수료 공제 후 리뷰어에게 토큰 지급

```

// 좋아요 기능
function likePost(address user2) public {
    require(user2 != address(0), "Invalid user address");

    uint256 reward = 10 * (10**uint256(tether.decimals()));
    require(
        Ddabbong.balanceOf(address(this)) >= reward,
        "Not enough tokens in DecentralBank"
    );
    Ddabbong.transfer(user2, reward);
    emit Liked(msg.sender, user2);
}
    
```


03. UI 디자인 및 기능 상세

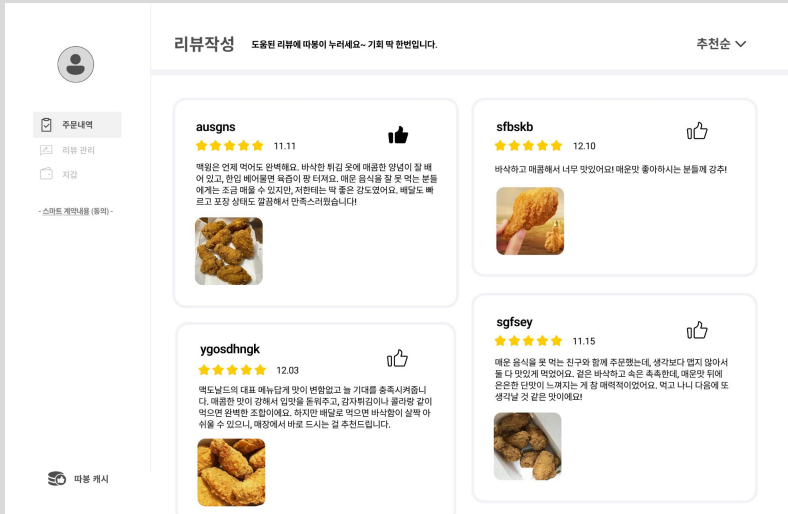


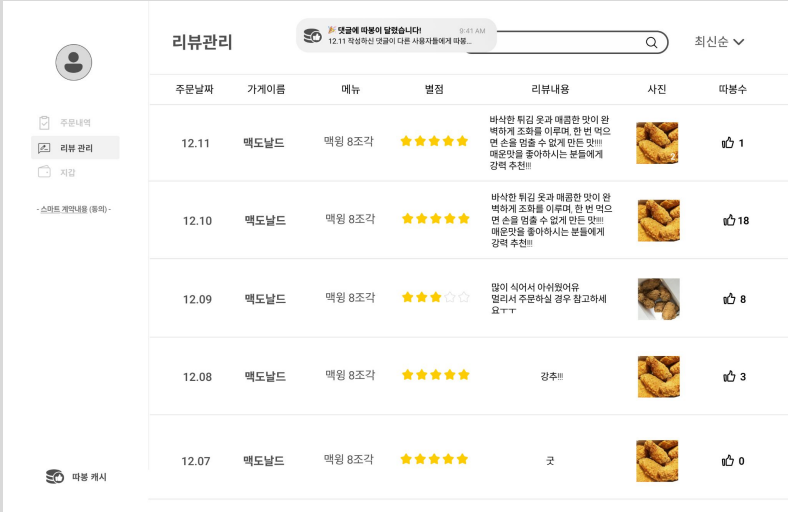

화면(보고서) ID	TBC-SC-001	화면(보고서)명	스마트 계약 관리
화면(보고서) 경로	메인페이지 화면 - 스마트계약관리		
화면(보고서) 구성	<div><div><div><div><div><div></div><div>주문내역</div></div><div><div></div><div>리뷰 관리</div></div><div><div></div><div>지급</div></div></div><div><div>- 스마트 계약내용 (종료) -</div></div></div><div><div><div>스마트 계약 약관 동의</div><div><div>스마트 계약 서비스 공지 및 이용 약관</div><div><p>저희 마봉 케시 스마트 리뷰 보상 시스템을 이용해 주셔서 감사합니다. 서비스를 이용하기 전에 아래 약관을 꼼꼼히 읽어 주시기 바랍니다. 리뷰 작성 또는 마봉(종아요) 참여 시 아래 약관에 동의한 것으로 간주됩니다.</p><div><div>1. 서비스 설명</div><div>이 서비스는 블록체인 스마트 계약을 기반으로 공정하고 투명한 리뷰 보상 환경을 제공합니다. 스마트 계약을 통해 작성된 리뷰와 마봉 데이터는 블록체인에 기록되어 데이터의 투명성과 변조 불가능성을 보장합니다.</div><div>2. 데이터 수집 및 저장</div><div><div>· 수집 내용</div><div><div>· 주문 ID: 리뷰 자격 검증에만 사용됩니다.</div><div>· 리뷰 내용 및 관련 메타데이터: 분산 스토리지 네트워크(IPFS)에 저장됩니다.</div><div>· 마봉 기록: 보상 분배에 사용됩니다.</div></div><div>· 데이터 사용 목적</div><div><div>· 리뷰 검증, 마봉 수 통계, 보상 지급에 사용됩니다.</div><div>· 사용자의 사전 동의 없이 다른 목적으로 활용되지 않습니다.</div></div><div>3. 보상 정책</div><div><div>· 작성한 리뷰는 다음 조건에 따라 보상이 지급됩니다.</div><div><div>a. 마봉 제한</div><div><div>· 마봉은 현재 주문한 제품의 리뷰에만 가능합니다.</div><div>· 동일 제품 내 리뷰에 대해서만 마봉을 누를 수 있습니다.</div></div><div>b. 마봉 수 제한</div><div><div>· 주문 건수에 따라 마봉 가능 수가 제한됩니다.</div><div>(예: 주문 1건일 경우 총아요 1회 가능)</div></div><div>c. 보상 지급</div><div><div>· 스마트 계약으로 논지워 마봉 소를 기호으로 지워오로 보상을 보배한다. J3</div></div></div></div></div><div><div>동의하고 계속하기</div></div></div></div></div></div></div></div></div>		<div><div>11:11</div><div>스마트 계약 약관 동의</div><div><div>스마트 계약 서비스 공지 및 이용 약관</div><div><p>저희 마봉 케시 스마트 리뷰 보상 시스템을 이용해 주셔서 감사합니다. 서비스를 이용하기 전에 아래 약관을 꼼꼼히 읽어 주시기 바랍니다. 리뷰 작성 또는 마봉(종아요) 참여 시 아래 약관에 동의한 것으로 간주됩니다.</p><div><div>1. 서비스 설명</div><div>이 서비스는 블록체인 스마트 계약을 기반으로 공정하고 투명한 리뷰 보상 환경을 제공합니다. 스마트 계약을 통해 작성된 리뷰와 마봉 데이터는 블록체인에 기록되어 데이터의 투명성과 변조 불가능성을 보장합니다.</div><div>2. 데이터 수집 및 저장</div><div><div>· 수집 내용</div><div><div>· 주문 ID: 리뷰 자격 검증에만 사용됩니다.</div><div>· 리뷰 내용 및 관련 메타데이터: 분산 스토리지 네트워크(IPFS)에 저장됩니다.</div><div>· 마봉 기록: 보상 분배에 사용됩니다.</div></div><div>· 데이터 사용 목적</div><div><div>· 리뷰 검증, 마봉 수 통계, 보상 지급에 사용됩니다.</div><div>· 사용자의 사전 동의 없이 다른 목적으로 활용</div></div></div></div></div><div><div>동의하고 계속하기</div></div></div></div>
	주요 항목 설명		<div><div>- 주요 버튼</div><div>1. smart contract 기반 리뷰 보상 시스템 약관 동의</div><div>2. 데이터 수집 및 저장 동의</div><div>3. 보상 정책 및 책임 동의</div><div>4. 동의하고 계속하기</div><div>: 메인페이지(TBC-MP-001) 화면 이동</div></div>
주요 체크 사항	기능 제약 사항 N/A		

화면(보고서) ID	TBC-MP-001	화면(보고서)명	메인페이지 화면
화면(보고서) 경로	로그인 - 메인페이지 화면		
화면(보고서) 구성			주요 항목 설명
			<p>- 주요 버튼:</p> <ol style="list-style-type: none">1. 사용자 프로필 : 주문내역 (TBC-OD-001) 화면이동2. 주문내역 : 리뷰관리(TBC-RM-001) 화면이동3. 리뷰관리 : 지갑관리(TBC-WM-001) 화면이동4. 지갑 : 스마트 계약내용(동의) : 스마트계약관리 화면이동 (TBC-SC-001)
주요 체크 사항	기능 제약 사항 - N/A		

화면(보고서) ID	TBC-OD-001	화면(보고서)명	주문내역 관리																																				
화면(보고서) 경로	메인페이지 화면 - 주문내역관리																																						
화면(보고서) 구성	<div><div><div><div><div></div><div>주문내역</div></div><div><div>리뷰 관리</div><div>지갑</div></div><div><div>- 스마트 계약내용 (종목) -</div></div></div><div><div>주문내역</div><table><thead><tr><th>주문날짜</th><th>주문상태</th><th>가게</th><th>메뉴</th><th>금액</th><th>리뷰상태</th></tr></thead><tbody><tr><td>12.12 (목)</td><td>배달완료</td><td></td><td>맥도날드 맥왕 8조각 x1</td><td>11,500원</td><td>리뷰 쓰기</td></tr><tr><td>12.11 (수)</td><td>배달완료</td><td></td><td>맥도날드 맥왕 8조각 x1</td><td>11,500원</td><td>작성했던 리뷰 보기</td></tr><tr><td>12.10 (화)</td><td>배달완료</td><td></td><td>맥도날드 맥왕 8조각 x1</td><td>11,500원</td><td>작성했던 리뷰 보기</td></tr><tr><td>12.09 (월)</td><td>배달완료</td><td></td><td>맥도날드 맥왕 8조각 x1</td><td>11,500원</td><td>작성했던 리뷰 보기</td></tr><tr><td>12.08 (일)</td><td>배달완료</td><td></td><td>맥도날드 맥왕 8조각 x1</td><td>11,500원</td><td>작성했던 리뷰 보기</td></tr></tbody></table></div><div><div>마포점</div><div>계시</div></div></div></div> <div><div>11:11</div><div>주문내역</div><div>12.12 (목) · 배달완료</div><div><div></div><div>맥도날드 맥왕 8조각 x1</div><div>리뷰 쓰기</div></div><div>12.11 (수) · 배달완료</div><div><div></div><div>맥도날드 맥왕 8조각 x1</div><div>작성했던 리뷰 보기</div></div><div>12.10 (화) · 배달완료</div><div><div></div><div>맥도날드 맥왕 8조각 x1</div><div>작성했던 리뷰 보기</div></div><div>12.09 (월) · 배달완료</div><div><div></div><div>맥도날드 맥왕 8조각 x1</div><div></div></div></div> <td>주요 항목 설명</td>		주문날짜	주문상태	가게	메뉴	금액	리뷰상태	12.12 (목)	배달완료		맥도날드 맥왕 8조각 x1	11,500원	리뷰 쓰기	12.11 (수)	배달완료		맥도날드 맥왕 8조각 x1	11,500원	작성했던 리뷰 보기	12.10 (화)	배달완료		맥도날드 맥왕 8조각 x1	11,500원	작성했던 리뷰 보기	12.09 (월)	배달완료		맥도날드 맥왕 8조각 x1	11,500원	작성했던 리뷰 보기	12.08 (일)	배달완료		맥도날드 맥왕 8조각 x1	11,500원	작성했던 리뷰 보기	주요 항목 설명
	주문날짜	주문상태	가게	메뉴	금액	리뷰상태																																	
12.12 (목)	배달완료		맥도날드 맥왕 8조각 x1	11,500원	리뷰 쓰기																																		
12.11 (수)	배달완료		맥도날드 맥왕 8조각 x1	11,500원	작성했던 리뷰 보기																																		
12.10 (화)	배달완료		맥도날드 맥왕 8조각 x1	11,500원	작성했던 리뷰 보기																																		
12.09 (월)	배달완료		맥도날드 맥왕 8조각 x1	11,500원	작성했던 리뷰 보기																																		
12.08 (일)	배달완료		맥도날드 맥왕 8조각 x1	11,500원	작성했던 리뷰 보기																																		
주요 체크 사항	기능 제약 사항 N/A																																						

화면(보고서) ID	TBC-OD-002	화면(보고서)명	리뷰작성화면
화면(보고서) 경로	메인페이지화면 - 주문내역관리 - 리뷰작성화면		
화면(보고서) 구성			주요 항목 설명
			<p>- 주요 버튼:</p> <ol style="list-style-type: none">별 버튼 : 별 1에서 5개 까지 별점 내기사진첨부 : 음식 사진 첨부NEXT : 리뷰선택(TBC_OD-003)화면이동
주요 체크 사항	기능 제약 사항 - N/A		

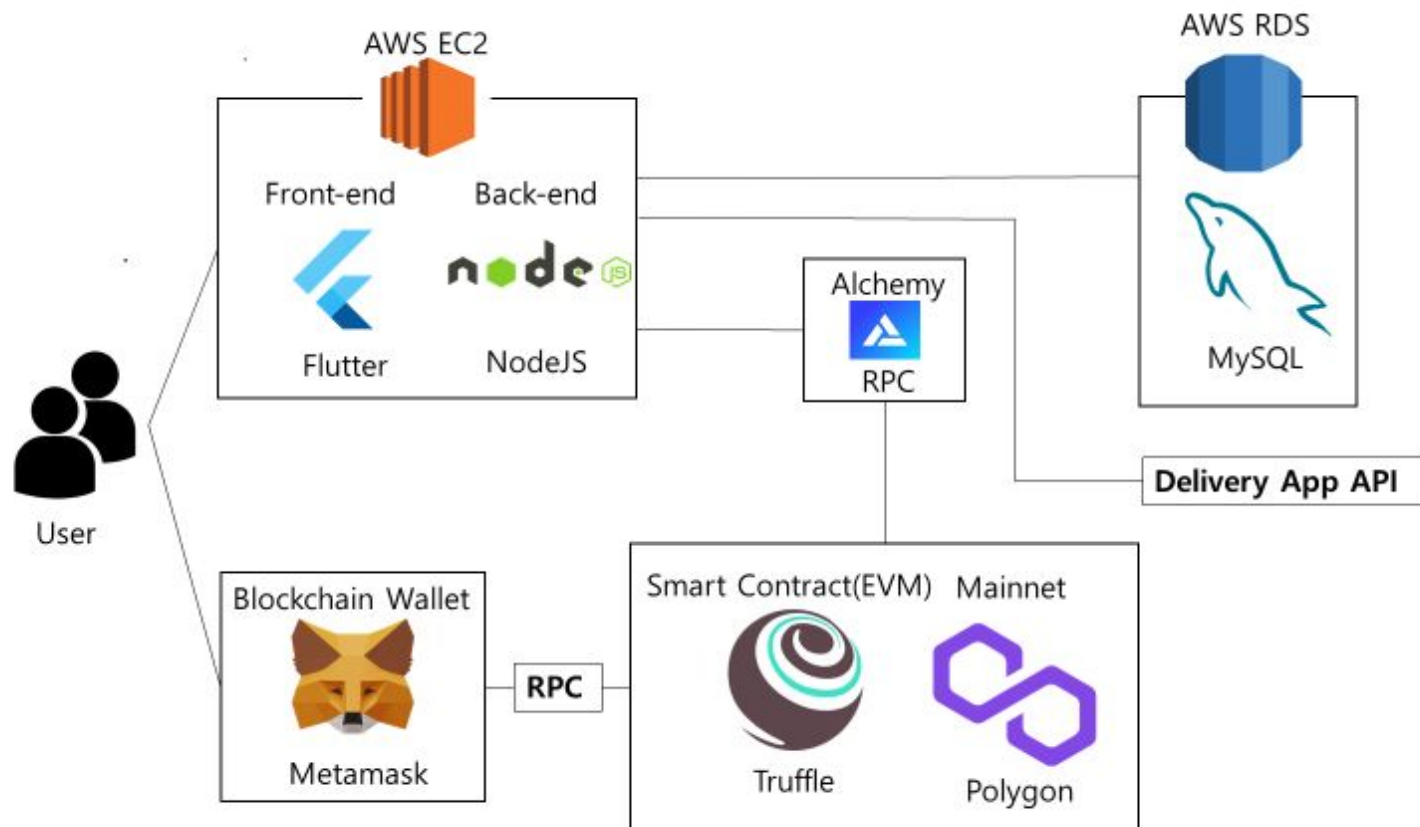
화면(보고서) ID	TBC-OD-003	화면(보고서)명	리뷰선택화면
화면(보고서) 경로	메인페이지화면 - 주문내역관리 - 리뷰작성화면 - 리뷰선택화면		
화면(보고서) 구성			<p data-bbox="1561 254 1754 287">주요 항목 설명</p> <p data-bbox="1435 341 1821 527">- 주요 버튼: 1. 따봉 : 도움된 리뷰 하나 선택해 따봉 누르기 2. 정렬 : 최신순, 따봉순에 따라 정렬</p>
주요 체크 사항	<p data-bbox="305 1013 469 1046">기능 제약 사항</p> <p data-bbox="305 1057 372 1089">- N/A</p>		

화면(보고서) ID	TBC-RM-001	화면(보고서)명	리뷰관리
화면(보고서) 경로	메인페이지화면 - 리뷰관리		
화면(보고서) 구성			주요 항목 설명
			<p>- 주요 버튼:</p> <ol style="list-style-type: none">조회 : 가게 이름 입력 해당 주문내역 조회정렬 : 최신순, 따봉순에 따라 정렬 <p>추가: 팝업 알림 창 작성한 리뷰에 따봉이 달렸을 때 알림 기능</p> <p>예시: “12.10일에 작성한 리뷰에 따봉이 달렸어요!”</p>

주요 체크 사항	기능 제약 사항 - N/A
----------	-------------------

화면(보고서) ID	TBC-WM-001	화면(보고서)명	지갑관리
화면(보고서) 경로	메인페이지 화면 - 지갑관리		
화면(보고서) 구성	<div><div><div><div><div><div></div><div>지갑</div></div><div><div>주요내역</div><div>리뷰 관리</div><div>지갑</div></div></div><div><div>· 스마일, 계좌내역 (종목) ·</div></div></div><div><div><div><div><div>지갑</div><div>보유 따봉 캐시</div><div><div><div><div>350</div><div>캐시</div></div></div><div><div>포인트 전환</div><div>현금 인출</div></div></div></div><div><div><div><div>획득</div><div>12.12에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10 캐시</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10 캐시</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10 캐시</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10 캐시</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10 캐시</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10 캐시</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10 캐시</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10 캐시</div></div></div></div></div></div></div></div></div>		<div><div>주요 항목 설명</div><div>- 주요 버튼: 1. 포인트 전환 : 주문할때 직접 사용할 수 있는 포인트로 전환 2. 현금 인출 : 사용자 통장으로 전환</div></div>
	<div><div><div><div><div>11:11</div><div>지갑</div></div><div><div><div><div>따봉 캐시</div><div>350</div><div>캐시</div></div><div><div>포인트 전환</div><div>현금 인출</div></div></div><div><div><div><div>획득</div><div>12.12에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10</div></div><div><div>획득</div><div>12.11에 맥도날드에서 작성했던 리뷰 통해 10 캐시 획득</div><div>+ 10</div></div></div></div></div></div></div></div>		
주요 체크 사항	기능 제약 사항 N/A		

04. 기술 스택 선정



05. 데모 영상



<https://youtu.be/-DuSEwxaDUc>

06. 주요 코드

```
pragma solidity ^0.5.0;

contract Ddabbong{
    string public name="Ddabbong";
    string public symbol="Cache";
    //1 million tokens, 187의 0는 소수점을 의미하는 거라 빼고 생각
    uint256 public totalSupply=1000000000000000000000000;
    uint8 public decimals=18;
    //event는 발생시 블록체인 블록에 로그로 기록 남게 하는 친구임
    //indexed는 주소로 필터링해서 검색할 수 있음-인덱싱 필수있는 파라미터가 생기는 것
    event Transfer(
        address indexed _from,
        address indexed _to,
        uint _value
    );
    //owner인 소유자가 승인하고 spender인 사용자에게 보냄
    event Approval(
        address indexed _owner,
        address indexed _spender,
        uint _value
    );
    //이 매핑은 각 계좌의 잔액 추적용-전송하거나 업데이트 할 때 꼭 필요한 부분
    mapping(address=> uint256) public balanceOf;

    //승인과 관련된 것(누가(A) 누구(B)에게 얼마만큼의 A의 돈을 형태로 써도 된다고 했는지 기록)
    mapping(address=> mapping(address=> uint256)) public allowance;

    constructor() public{
        balanceOf[msg.sender]=totalSupply; // 초기 공급량을 배포자에게 할당
    }
    function transfer(address _to, uint256 _value) public returns (bool success){
        //보내는 사람 잔액이 보내려는 값보다 많이 있어야함.
        require(balanceOf[msg.sender]>=_value);
        balanceOf[msg.sender]-=_value; //메시지 발신자(본인잔액)은 전송할 값을 제외한 값이됨
        balanceOf[_to]+=_value; //받는 사람은 보내는 값이 더해짐
        emit Transfer(msg.sender,_to,_value); //Transfer event 함수 실행하는 곳
        return true;
    }

    //owner가 A에게 owner돈을 얼마나 저당해도 해도 되는지 승인하는 거
    function approve(address _spender,uint256 _value)public returns (bool success){
        allowance[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value);
        return true;
    }

    //제 3자가 다른사람(참수) 돈을 다른사람(참수)에게 전송하는 함수
    function transferFrom(address _from,address _to, uint256 _value) public returns (bool success){
        require(_value <=balanceOf[_from]);
        require(_value <=allowance[_from][msg.sender]);
        balanceOf[_to]+=_value; //받는 사람은 보내는 값이 더해짐
        balanceOf[_from]-=_value;
        allowance[_from][msg.sender]-=_value;
        emit Transfer(_from,_to,_value); //Transfer event 함수 실행하는 곳
        return true;
    }
}
```

```
pragma solidity ^0.5.0;
import './Rwd.sol';
import './Ddabbong.sol';
contract DecentralBank{
    string public name="Decentral Bank";
    address public owner;
    Tether public tether;
    Rwd public rwd;
    address[] public stakers;
    //스테이킹 한 사람들의 주소를 저장하기 위한용도

    mapping(address=> uint) public stakingBalance;
    //계좌별로 얼마 예금했는지 저장하기 위한 것

    mapping(address=> bool) public hasStaked;
    //스테이킹 이력(한 적 있는지)

    mapping(address=> bool) public isStaking;
    //스테이킹 진행 여부(하고 있는지)

    // 좋아요 이벤트
    event Liked(address indexed user1, address indexed user2);

    constructor(Rwd _rwd, Tether _tether) public{
        rwd=_rwd;
        tether=_tether;
        owner=msg.sender;
    }

    //staking 함수
    function depositToken(uint _amount) public {
        require(_amount > 0,'amount cannot be 0');

        //전송할 테더 토큰을 이 계약의 주소로 보냄(staking을 위해)
        tether.transferFrom(msg.sender, address(this), _amount);
        //from: 호출하는 사람 to:이 계약 주소, value: amount

        //예금액 업데이트
        stakingBalance[msg.sender]+= _amount;

        if(!hasStaked[msg.sender]){
            stakers.push(msg.sender);
        }
        //업데이트
        isStaking[msg.sender]=true;
        hasStaked[msg.sender]=true;
    }
}
```

```
//unstake tokens=>예금 출금 가능
function unstakeTokens() public {
    uint balance=stakingBalance[msg.sender];
    require(balance>0,'staking balance can't be less than 0');
    //정해진 주소로 일정량의 토큰을 전송(지금은 전부 빼도록 코딩함)
    tether.transfer(msg.sender, balance);

    //출금완료 계좌 업데이트
    stakingBalance[msg.sender]=0;

    //staking status(상태) 업데이트
    isStaking[msg.sender]=false;
}

//issue reward 스테이킹 이자
function issueTokens() public {
    //owner만 발행할 수 있음
    require(msg.sender==owner,'caller must be owner');

    for(uint i=0; i<stakers.length;i++){
        address recipient=stakers[i];
        uint balance=stakingBalance[recipient] / 9;
        if(balance>0){
            //스테이킹 한거에 9분의 1만큼 그 사람에게 리워드 전송
            rwd.transfer(recipient, balance);
        }
    }
    // 좋아요 가능
    function likePost(address user2) public {
        require(user2 != address(0), "Invalid user address");

        uint256 reward = 10 * (10**uint256(tether.decimals())); // 10 Tether 토큰
        require(
            tether.balanceOf(address(this)) >= reward,
            "Not enough tokens in DecentralBank"
        );
        tether.transfer(user2, reward);

        // 좋아요 이벤트 발생
        emit Liked(msg.sender, user2);
    }
}
```

ERC-20 표준에 맞게
토큰 발행 및 전송

06. 주요 코드

```
const Ddabong=artifacts.require('Ddabong')
const Rwd=artifacts.require('Rwd')
const DecentralBank=artifacts.require('DecentralBank')
require('chai')
.use(require('chai-as-promised'))
.should

contract('DecentralBank',([owner,customer]) =>{
  let ddabong, rwd, decentralBank
  function tokens(number){
    return web3.utils.toWei(number,'ether')
  }
  before(async ()=>{
    //계약 가져옴
    ddabong=await Ddabong.new()
    rwd=await Rwd.new()
    decentralBank=await DecentralBank.new(rwd.address, ddabong.address)

    //모든 토큰 은행에 옮기는거 테스트
    await rwd.transfer(decentralBank.address, tokens('1000000'))

    //100개 토큰 투자자에게 전송하는거
    await ddabong.transfer(customer, tokens('100'),{from: owner})

  })//여기 넣은 것은 테스트 이전, describe이전에 실행됨
  //테스트에 돌릴 코드 입력하는 곳
  describe('Ddabong Deployment',async ()=>{
    it('matches name successfully', async ()=>{
      const name=await ddabong.name()
      assert.equal(name,'Ddabong')
    })
  })
  describe('Rewark Token Deployment',async ()=>{
    it('matches name successfully', async ()=>{
      const name=await rwd.name()
      assert.equal(name,'Reward Token')
    })
  })
  describe('Decentral Bank Deployment',async ()=>{
    it('matches name successfully', async ()=>{
      const name=await decentralBank.name()
      assert.equal(name,'Decentral Bank')
    })
  })
  it('contract has tokens',async ()=>{
    let balance= await rwd.balanceOf(decentralBank.address)
    assert.equal(balance,tokens('1000000'))
  })
  })
  describe('Yield Farming', async ()=> {
    it('rewards tokens for staking', async ()=>{
      //투자자의 잔액 체크
      let result= await ddabong.balanceOf(customer)
      assert.equal(result.toString(), tokens('100'),
        'customer mock wallet balance before staking')
    })
  })
})
```

```
//고객(2번째 계정)이 받았던 100개 토큰 전부 은행에 보내고 staking 체크
await ddabong.approve(decentralBank.address,tokens('100'), {from:customer})
//승인먼저 받아야함
await decentralBank.depositToken(tokens('100'),{from: customer})

//받은 고객의 잔액 체크
result= await ddabong.balanceOf(customer)
assert.equal(result.toString(), tokens('0'),
  'customer mock wallet balance after staking 100 tokens')

//은행에 있는 잔액 체크
result= await tether.balanceOf(decentralBank.address)
assert.equal(result.toString(), tokens('100'),
  'decentral bank mock wallet balance after staking from customer')

//isstaking 체크
result= await decentralBank.isStaking(customer)
assert.equal(result.toString(), 'true','customer isStaking status after staking')

//issue tokens(토큰 발행 테스트)
await decentralBank.issueTokens({from: owner})

//소유자만 토큰 발행할 수 있다. 투자자는 안됨!

// 이 코드 안되서 아래로 바꿈. await decentralBank.issueTokens({from: customer}).should.be.rejected;
try {
  await decentralBank.issueTokens({from: customer});
  assert.fail('Customer should not be able to issue tokens');
} catch (error) {
  assert(error.message.includes('caller must be owner'),
    'Expected revert error for unauthorized call');
}

//unstake tokens
await decentralBank.unstakeTokens({from: customer})

//언스테이킹 하고 남은 잔액 체크
result= await ddabong.balanceOf(customer)
assert.equal(result.toString(), tokens('100'),
  'customer mock wallet balance after unstaking')

//은행에 있는 잔액 체크
result= await ddabong.balanceOf(decentralBank.address)
assert.equal(result.toString(), tokens('0'),
  'decentral bank mock wallet balance after staking from customer')
//is staking update
result= await decentralBank.isStaking(customer)
assert.equal(result.toString(), 'false',
  'customer is no longer Staking status after unstaking')
})
})
```

Front-End로 미비된 기능들은 mocha/chai를 통해 Testing

06. 주요 코드

```
import React, {Component, useState} from 'react'
import './App.css'
import Navbar from './Navbar'
import Main from './Main.js'
import Web3 from 'web3'
import Ddabong from '../contracts/Ddabong.json'
import RMD from '../contracts/RMD.json'
import DecentralBank from '../contracts/DecentralBank.json'
import { BrowserRouter as Router, Route, Routes, useNavigate } from 'react-router-dom'
import OrderHistory from './OrderHistory' // 새로운 페이지
import { FaThumbUp } from 'react-icons/fa' // 박동 아이콘 추가

class App extends Component {
  async UNSAFE_componentWillMount() { //컴포넌트 마운트 시 최초 실행으로 불러오게함
    await this.loadWeb3() //메타마스크 연결
    await this.loadBlockchainData() //계좌 데이터 연결
  }

  async loadWeb3() {
    if (window.ethereum) { //윈도우 창에서 이더리움이 감지되었을때
      window.web3 = new Web3(window.ethereum)
      await window.ethereum.enable() //이더리움 활성화
    } else if (window.web3) {
      window.web3 = new Web3(window.web3.currentProvider)
    } else {
      window.alert('No ethereum browser detected! check out Metamask!')
    }
  }

  async loadBlockchainData() {
    const web3 = window.web3
    const account = await web3.eth.getAccounts() //불특정인 데이터에서 계좌 불러옴
    this.setState({account: account[0]})
    const networkId = await web3.eth.net.getId() //네트워크 ID 가져옴
    //다들 계약 불러오기
    const tetherData = Ddabong.networks[networkId]
    if (tetherData) {
      const tether = new web3.eth.Contract(Ddabong.abi, tetherData.address) //abi, 주소 가져옴
      this.setState({tether}) //상태 업데이트
      let tetherBalance = await tether.methods.balanceOf(this.state.account).call() //잔액 정보 가져옴
      this.setState({tetherBalance: tetherBalance.toString()})
    } else {
      window.alert('Error! Tether token not deployed to the network!')
    }
    //리워드 계약 불러오기
    const rwdData = RMD.networks[networkId]
    if (rwdData) {
      const rwd = new web3.eth.Contract(RMD.abi, rwdData.address) //abi, 주소 가져옴
      this.setState({rwd}) //상태 업데이트
      let rwdBalance = await rwd.methods.balanceOf(this.state.account).call() //잔액 정보 가져옴
      this.setState({rwdBalance: rwdBalance.toString()})
    } else {
      window.alert('Error! RMD token not deployed to the network!')
    }
    //decentralbank 계약 불러오기
    const decentralBankData = DecentralBank.networks[networkId]
    if (decentralBankData) {
      const decentralBank = new web3.eth.Contract(DecentralBank.abi, decentralBankData.address) //abi, 주소 가져옴
      this.setState({decentralBank}) //상태 업데이트
      let stakingBalance = await decentralBank.methods.stakingBalance(this.state.account).call() //잔액 정보 가져옴
      this.setState({stakingBalance: stakingBalance.toString()})
    } else {
      window.alert('Error! Decentral Bank token not deployed to the network!')
    }
    this.setState({loading: false}) //로딩 다했으면 false로 바꿈
  }
}
```

1. Web3.js에 기반하여 Contract의 JSON 정보를 abi로 가져옴

2. window를 통해 Metamask와 연동

<React 中>

감사합니다

https://github.com/hyosungan/blockchain_smart-contract/tree/master