2025-1 블록체인 최종보고서

|  |  |  |  |
| --- | --- | --- | --- |
| **작품명** | **블록체인을 활용한 출석 보상 시스템** | | |
| **팀 원** | **학 번** | **성 명** | **역할/주요업무** |
| 2021810008 | 김도현 | 팀장 및 주요 기능 개발, 피피티 제작 |
| 2019250005 | 김대홍 | 발표 및 부분개발 |
| 2022810033 | 배유화 | 보고서 작성 및 부분개발 |
| 2020810041 | 손영준 | 개발 총괄 및 시연영상설명 |
| 2021810033 | 심다정 | 주요 기능 개발 및 보고서 작성, 피피티 제장 |

**2025.06.04**

**목 차**

# 과제 개발팀 및 역할 분담

# 개발시스템(Target System) 명세

* 1. 개발시스템의 목적
  2. 개발시스템의 예상 사용자 및 사용자가 느끼는 예상 효용
  3. 개발시스템 개요도
  4. 개발시스템의 주요기능 개발된 내용 설명
  5. 개발시스템의 범위(참고한 외부소스& 업그레이드 내용 )
  6. 유사시스템의 존재 여부와 기존 유사시스템과의 장단점 비교
  7. 상호평가 (각 팀원이 다른 팀원을 100점 만점 기준평가한  평가표 (팀원별 평균 포함)

# 부 록

R-1. 참고문헌 및 참고사이트

R-2. 프로그램 소스(각 기능별/모듈별 파일명 설

# 과제 개발팀 및 역할분담

|  |  |
| --- | --- |
| 팀원 | 역할 |
| 김도현 | 개발환경 구축, 출석체크 기능 코드 작성, 통합 테스트 |
| 김대홍 | 개발환경 구축, 스마트 계약 개발 코드 작성, 발표 |
| 배유화 | 개발환경 구축, 토큰 지급/보상 로직 코드 작성, 문서 작성 |
| 손영준 | 개발환경 구축, 백엔드 서버 코드 작성, 통합 테스트 |
| 심다정 | 개발환경 구축, DB 연동 코드 작성, 통합 테스트, 문서 작성 |

# 개발시스템(Target System) 명세

* 1. 개발시스템의 목적

최근 비대면 학습과 온라인 교육의 확산으로 출석 관리의 신뢰성과 효율성에 대한 문제가 대두되고 있다. 기존 출석 시스템은 중앙 서버에 의존하고 있어 조작 가능성이 존재하며, 별도의 보상 체계가 없어 학습자의 자발적인 참여를 유도하기 어려운 구조이다. 이러한 문제를 해결하기 위해 본 시스템은 블록체인의 투명성과 불변성을 활용하여 신뢰할 수 있는 출석 보상 시스템을 구축하는 것을 목적으로 한다.

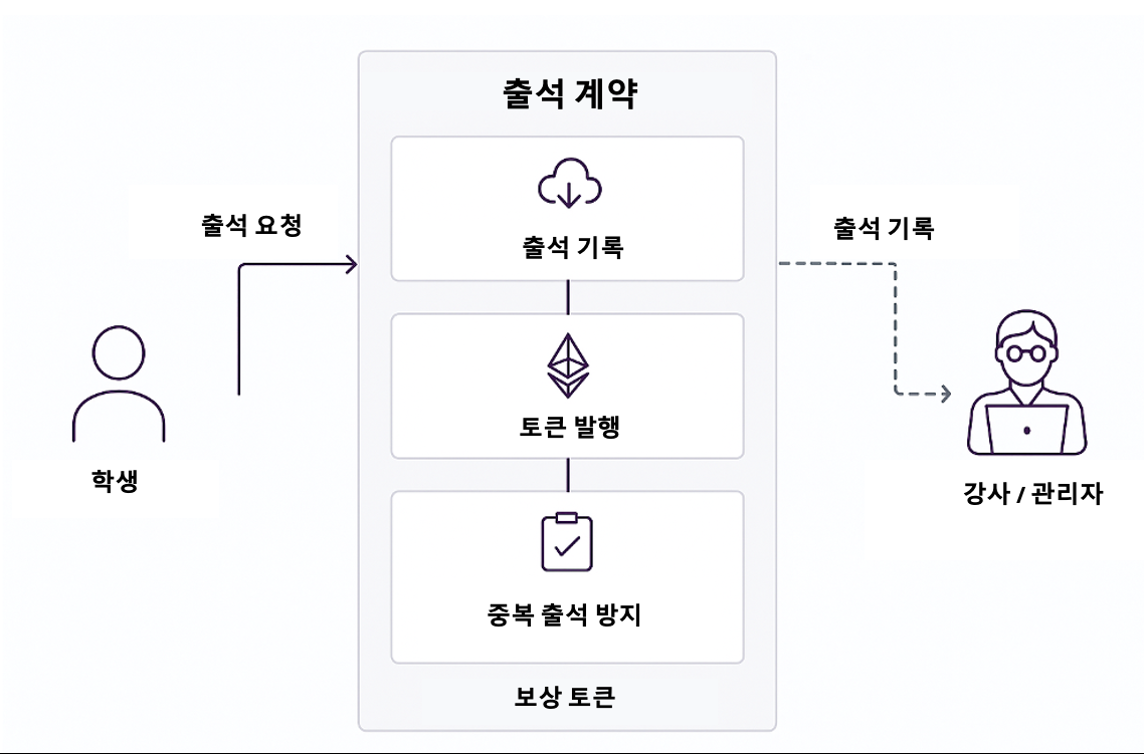
본 시스템은 스마트 계약(Smart Contract)을 기반으로 하여 사용자의 출석 내역을 블록체인에 자동으로 기록하고, 정해진 조건을 만족하면 ERC-20 기반 토큰을 자동 지급하는 구조로 설계되었다. 사용자는 단순히 출석 버튼을 클릭하는 것만으로 출석이 블록체인에 기록되며, 일정 횟수 출석 시 보상 토큰을 지급받게 된다. 이 과정은 조작이 불가능한 블록체인 상에서 이루어지므로 강사나 운영자 또한 공정하고 투명한 출석 관리가 가능하다.

이처럼 본 시스템은 탈중앙화된 구조를 통해 출석 관리의 신뢰성을 확보하고, 보상 기능을 통해 학습자의 자발적인 참여와 지속적인 학습을 유도하는 것을 핵심 목적으로 한다.

* 1. 개발시스템의 예상 사용자 및 사용자가 느끼는 예상 효용

예상 사용자 : 온라인 강의 수강생, 강사, 교육기관 관리자

예상 효용:

* 수강생은 간단한 출석 버튼 클릭만으로 출석 기록이 블록체인에 자동 저장되고 토큰 보상을 받을 수 있어 학습 참여 동기가 증대된다.
* 강사는 조작 불가능한 기록을 바탕으로 정확하고 공정한 출석 관리를 수행할 수 있다.
* 관리자는 데이터 위·변조 방지 및 기록 투명성 확보를 통해 신뢰성 높은 학사 관리를 실현할 수 있다.
  1. 개발시스템 개요도
  2. 개발시스템의 주요기능 개발된 내용 설명

2.4.1 사용자 기능

|  |  |
| --- | --- |
| 기능 | 구성 내용 |
| 지갑 연결 | MetaMask 지갑을 연결하여 개인 계정 주소확보 |
| 출석 일자 입력 | YYYYMMDD 형식의 출석 일자를 사용자가 직접 입력 |
| 출석 요청 | 버튼 클릭 시 스마트컨트랙트에 트랜잭션 요청 |
| 출석 횟수 조회 | 현재까지 출석한 횟수를 실시간 표시 |
| 보상 잔액 확인 | ERC-20 토큰 기준 개인 보상 잔액 표시 |

2.4.2 관리자 기능

|  |  |
| --- | --- |
| 기능 | 구성 내용 |
| 출석 기준 설정 | 몇 회 출석마다 보상을 지급할지 N값 설정 |
| 보상 수량 설정 | 지급할 토큰 양(단위: wei)을 설정 |
| 학생 출석/잔액 조회 | 임의의 주소 입력 시 해당 유저 출석 횟수, 보상 토큰 잔액 확인 |
| 허용된 호출자 등록 | AttendanceHub 컨트랙트를  AttendanceLog에 등록 (최초 1회) |

2.4.3 스마트 컨트랙트 기능 (Solidity)

- AttendanceToken.sol

|  |  |
| --- | --- |
| 함수 | 기능 |
| mint(address, amount) | 출석 기준 충족 시 보상 지급 |
| balanceOf(address) | 토큰 잔액 확인 |
| transfer/transferFrom | 토큰 전송 (표준 ERC-20 기능 포함 ) |

- AttendanceLog.sol

|  |  |
| --- | --- |
| 함수 | 기능 |
| logAndReward(uint date) | 출석 처리 + 보상 지급 트리거 |
| setRewardEvery(uint) | 출석 간격 설정 |
| setRewardAmount(uint) | 보상 토큰 양 설정 |
| rewardAmount() | 현재 설정값 확인용 view 함수 |

- AttendanceHub.sol

|  |  |
| --- | --- |
| 함수 | 기능 |
| logAttendaceFor(address, date) | 특정 유저의 출석 내역 기록 |
| getAttendanceCount(address) | 출석한 총 일수 반환 |
| hasCheckedln(user, date) | 중복 출석 여부 확인용 |
| setHub(address) | 호출 권한이 있는 Hub 주소 설정 (최초 1회만 허용) |

* 1. 개발시스템의 범위(참고한 외부소스& 업그레이드 내용)

2.5.1 출석 횟수 기반 보상 시스템 구현

단순 출석 기록뿐 아니라, 일정 횟수마다 보상을 자동 지급하는 구조

2.5.2 2개 스마트컨트랙트의 모듈화

AttendanceHub, AttendanceLog, AttendanceToken을 분리하여 실행

2.5.3 중복 출석 방지 로직

사용자-날짜 매핑을 통해 동일한 날짜의 중복 출석 불가

2.5.4 프론트엔드 연동 및 시각적 UI 구현

메타마스크 연동. (HTML/JS)

2.5.5 참고한 외부 소스

OpenZeppelin Contracts v4.9.3

→ ERC-20 토큰 구현 및 소유권 제어(Ownable) 기능 적용

→ 보안성과 신뢰성이 검증된 스마트컨트랙트 코드 기반

이더리움 개발자 문서 및 튜토리얼

→ 스마트컨트랙트 패턴, 프론트엔드-컨트랙트 연동 구조 참조

기초 출석 시스템 예제 코드

→ 스마트컨트랙트를 이용한 출석 기록 구조의 기초 로직 활용

* 1. 유사시스템의 존재 여부와 기존 유사시스템과의 장단점 비교

|  |  |  |
| --- | --- | --- |
| 항목 | 기존 시스템(LMS,  출석 앱) | 블록체인 기반 출석 보상 시스템 |
| 보상 시스템 | 없음 (출석 후 점수 부여) | 일정 출석마다 ERC-20 토큰 자동 지급 |
| 출석 기록 방식 | 웹 기반 출석 버튼 또는 자동 입력 | 블록체인 스마트컨트랙트에 기록 |
| 출석 검증 방식 | 서버 기록 기준 또는 수동 검토 | Metamask 지갑 연동 + 스마트 컨트랙트 처리 |
| 출석 인증 | 단순 로그인 | Metamask서명 (지갑 주소) |
| 사용자 신뢰성 | 서버 관리자에 의존 | 탈중앙화 구조 기반 |

* 1. 작품보완점 및 목표구현 정도

본 시스템은 출석 체크와 보상 지급이라는 기본 목적을 충실히 구현하였으며, 출석 내역이 블록체인에 안전하게 저장되고, 일정 횟수 출석 시 ERC-20 토큰이 자동으로 지급되는 핵심 기능을 성공적으로 구현하였다. MetaMask 지갑을 통한 사용자 인증, 스마트컨트랙트 호출, 실시간 잔액/출석횟수 반영 등 DApp의 기본적인 구조도 잘 반영되어 있다.

다만, 사용자 인터페이스(UI) 측면에서는 시각적 완성도나 사용자 친화적인 기능이 다소 미흡한 부분이 있다. 예를 들어, 출석 가능한 날짜의 캘린더 입력 기능, 출석 이력 리스트, 잔여 보상 횟수 표시 등 **시각적 피드백 요소**가 부족해 추가 개선이 필요하다.

* 1. 상호평가 (각 팀원이 다른 팀원을 100점 만점 기준평가한 평가표 (팀원별 평균 포함)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 평가자/  대상자 | 김도현 | 김대홍 | 배유화 | 손영준 | 심다정 | 평균 |
| 김도현 |  | 100 | 100 | 100 | 100 | 100 |
| 김대홍 | 100 |  | 100 | 100 | 100 | 100 |
| 배유화 | 100 | 100 |  | 100 | 100 | 100 |
| 손영준 | 100 | 100 | 100 |  | 100 | 100 |
| 심다정 | 100 | 100 | 100 | 100 |  | 100 |
| 팀원별 평균 | 100 | 100 | 100 | 100 | 100 | 100 |

# 부 록

R-1. 참고문헌 및 참고사이트

예제로 배우는 블록체인 – 한빛미디어

R-2. 프로그램 소스(각 기능별/모듈별 파일명 설명)

AttendanceHub.sol : 출석 로직 실행 및 보상 토큰 지급을 통합 관리하는 중앙 컨트롤러

|  |
| --- |
| // SPDX-License-Identifier: MIT  pragma solidity ^0.8.20;  /\* OpenZeppelin의 Ownable을 상속하여 관리자(owner)만 설정 변경 가능하도록 함 \*/  import "@openzeppelin/contracts@4.9.3/access/Ownable.sol";  /\* 출석 기록 인터페이스: 실제 로직은 AttendanceLog 컨트랙트에 있음 \*/  interface IAttendanceLog {  function logAttendanceFor(address user, uint256 date) external;  function getAttendanceCount(address user) external view returns (uint256);  }  /\* 보상용 ERC-20 토큰 인터페이스: 실제 토큰은 AttendanceToken 컨트랙트에서 발행 \*/  interface IAttendanceToken {  function mint(address to, uint256 amount) external;  }  /\*\*  \* @title AttendanceHub  \* @notice 출석 처리 및 보상 지급을 담당하는 중앙 컨트랙트  \* @dev 외부 출석 로그와 토큰 발행 컨트랙트를 연결하여 역할 분리  \*/  contract AttendanceHub is Ownable {  IAttendanceLog public log; // 출석 기록 컨트랙트 주소  IAttendanceToken public token; // 보상 토큰 컨트랙트 주소  uint256 public rewardEvery = 3; // 출석 보상 지급 기준 (기본: 3회마다)  uint256 public rewardAmount = 1 ether; // 보상 토큰 수량 (기본: 1 ATT, 18 decimal)  /\*\*  \* @dev 생성자에서 출석 기록 컨트랙트와 토큰 컨트랙트 주소를 받아 저장  \* @param logAddr AttendanceLog 컨트랙트 주소  \* @param tokenAddr AttendanceToken 컨트랙트 주소  \*/  constructor(address logAddr, address tokenAddr) {  log = IAttendanceLog(logAddr);  token = IAttendanceToken(tokenAddr);  }  /\*\*  \* @notice 출석 보상 지급 기준을 설정 (예: 3이면 3회마다 보상)  \* @param n 새로운 보상 간격 값 (1 이상)  \* @dev onlyOwner: 컨트랙트 배포자만 호출 가능  \*/  function setRewardEvery(uint256 n) external onlyOwner {  require(n > 0, "must be >0");  rewardEvery = n;  }  /\*\*  \* @notice 출석 시 지급할 보상 토큰 수량 설정  \* @param weiAmt 지급할 토큰 수량 (단위: wei)  \* @dev onlyOwner: 컨트랙트 배포자만 호출 가능  \*/  function setRewardAmount(uint256 weiAmt) external onlyOwner {  require(weiAmt > 0, "must be >0");  rewardAmount = weiAmt;  }  /\*\*  \* @notice 출석 기록 + 보상 지급 함수  \* @param date 출석 일자 (예: 20250604)  \* @dev 출석 기록은 AttendanceLog에 저장되며, 출석 횟수에 따라 보상 지급  \*/  function logAndReward(uint256 date) external {  // 출석 기록 (중복 방지는 AttendanceLog에서 처리)  log.logAttendanceFor(msg.sender, date);  // 누적 출석 횟수 조회  uint256 cnt = log.getAttendanceCount(msg.sender);  // 설정된 간격마다 보상 지급  if (cnt % rewardEvery == 0) {  token.mint(msg.sender, rewardAmount);  }  }  } |

AttendanceLog.sol : 출석 날짜를 사용자별로 안전하게 기록하고, 중복 출석 방지

|  |
| --- |
| // SPDX-License-Identifier: MIT  pragma solidity ^0.8.20;  /\*\*  \* @title AttendanceLog  \* @notice 사용자의 출석 정보를 기록하는 스마트컨트랙트  \* @dev 출석 기록은 Hub 컨트랙트를 통해서만 가능 (보안 목적)  \*/  contract AttendanceLog {  address public hub; // 출석 요청을 허용할 스마트컨트랙트 주소 (AttendanceHub)  /\*\*  \* @dev 생성자에서 기본값 0으로 설정  \* hub는 한 번만 setHub()로 설정 가능  \*/  constructor() {  hub = address(0);  }  // 출석 여부 저장: 유저 → 날짜 → true/false  mapping(address => mapping(uint256 => bool)) public hasCheckedIn;  // 사용자별 출석 날짜 목록 (날짜 배열)  mapping(address => uint256[]) private \_datesByUser;  // 출석 시 발생하는 이벤트 (프론트엔드에서 수신 가능)  event AttendanceLogged(address indexed user,  uint256 indexed date,  uint256 timestamp);  // hub 주소에서만 실행 가능하도록 제한  modifier onlyHub() {  require(msg.sender == hub, "Only Hub");  \_;  }  /\*\*  \* @notice Hub 주소 설정 (한 번만 가능)  \* @param hubAddr AttendanceHub 컨트랙트 주소  \*/  function setHub(address hubAddr) external {  require(hub == address(0), "Hub already set"); // 이미 설정된 경우 차단  require(hubAddr != address(0), "Zero addr") // 0 주소 허용 안 함  hub = hubAddr;  }  /\*\*  \* @notice 사용자의 출석 기록을 저장  \* @dev AttendanceHub에서만 호출 가능  \* @param user 출석자 주소  \* @param date 출석 날짜 (예: 20250604)  \*/  function logAttendanceFor(address user, uint256 date)  external onlyHub  {  require(date > 20200000, "Date too old"); // 과거 날짜 방지  require(!hasCheckedIn[user][date], "dup"); // 중복 출석 방지  hasCheckedIn[user][date] = true; // 출석 기록 저장  \_datesByUser[user].push(date);// 날짜 배열에 추가  emit AttendanceLogged(user, date, block.timestamp); // 이벤트 발생  }  /\*\*  \* @notice 사용자의 총 출석 횟수 반환  \* @param user 지갑 주소  \* @return 출석 횟수  \*/  function getAttendanceCount(address user)  external view returns (uint256)  {  return \_datesByUser[user].length;  }  } |

AttendaceToken.sol : ERC-20 표준 기반의 출석 보상용 토큰 발행

|  |
| --- |
| // SPDX-License-Identifier: MIT  pragma solidity ^0.8.20;  /\*\*  \* @title AttendanceToken  \* @dev ERC20 토큰으로 출석 보상 지급에 사용됨  \* OpenZeppelin 4.9.3 버전의 ERC20 및 Ownable 기능을 상속받음  \*/  /\* ⬇️ 반드시 버전 지정 (Remix/NPM 환경에서 의존성 충돌 방지) \*/  import "@openzeppelin/contracts@4.9.3/token/ERC20/ERC20.sol";  import "@openzeppelin/contracts@4.9.3/access/Ownable.sol";  contract AttendanceToken is ERC20, Ownable {  /\*\*  \* @notice 생성자: 토큰 이름과 심볼 지정  \* "Attendance Token" = 전체 이름  \* "ATTN" = 심볼 (지갑에 표시될 약칭)  \*/  constructor() ERC20("Attendance Token", "ATTN") {}  /\*\*  \* @notice 토큰 민팅 함수  \* @dev 오직 소유자(배포자 or AttendanceHub)만 호출 가능  \* @param to 토큰을 받을 주소  \* @param amount 발행할 토큰 수량 (단위: wei, 10^18 기준)  \*/  function mint(address to, uint256 amount) external onlyOwner {  \_mint(to, amount); // OpenZeppelin 내부 함수 사용  }  } |