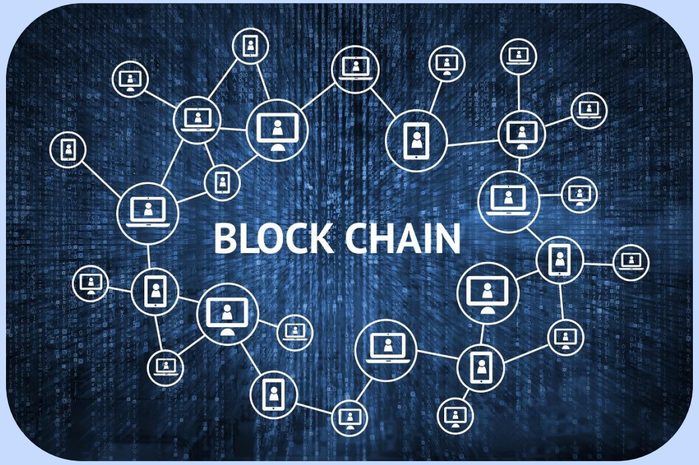
네트워크 최신기술

스마트 컨트랙트와 이를 활용하는 웹 서비스 개발
Secret-Contract (가제)

국민대학교 소프트웨어학부
20162347 김창규

프로젝트 소개

→



구두/문서로 작성/관리/이행되는
기존 계약 방식

블록체인을 활용해 안전하게
작성/관리/이행되는 계약 방식

소스 파일 및 동작 환경

[개발 환경]

운영체제	MS Windows 11 (10↑)
작성언어	HTML
	Javascript(+ Node.js)
	Solidity
IDE	VS Code

[Module]

Web3 (index.html)
http / fs / url (server.js)
Bootstrap 4.6.1 (index.html & style.css)

[실행 환경]

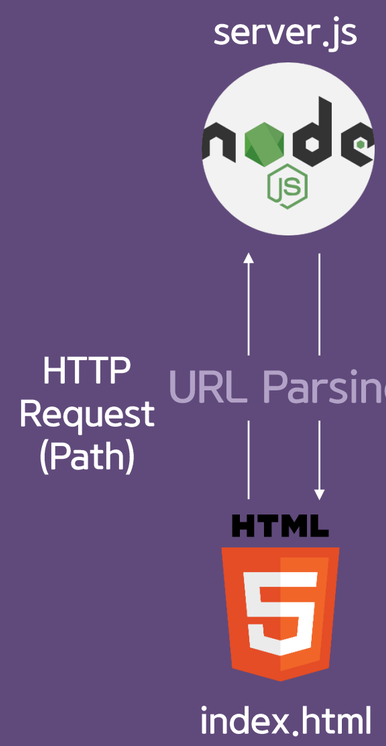
Server	localhost:8080
Client	localhost:8080/index.html

[디렉토리 구조]

```
node_modules/
└ web3/
  └ dist/
    └ web3.min.js
      web3.min.js.map
      (web3 제어 모듈)
  ...
```

Kim-Changgyu.html	(웹 서비스 소스코드)
Kim-Changgyu.sol	(스마트 컨트랙트 소스코드)
Kim-Changgyu.pdf	(발표 자료)
server.js	(Node.js 웹 서버 실행 파일)
style.css	(기본 CSS 설정 파일)
package*.json	(패키지 의존성)

[동작 흐름]



실행 방법 및 가이드

서버 실행

node server.js

클라이언트 실행

웹 브라우저에서 링크 접속
localhost:8080/Kim-Changgyu.html

클라이언트 요청

UI를 통한 각 함수(기능) 사용

Solidity (스마트 컨트랙트) 소스 코드 - 구조체 및 변수

```
//SPDX-License-Identifier: UNLICENSED

pragma solidity >= 0.8.13;

contract SecretContract {
    struct Instance {
        address A;                              // A
        address B;                              // B
        string content;                         // 계약 내용
        string date;                            // 작성일자자
    }

    address public owner;                       // 스마트 컨트랙트 소유주
    mapping (uint => Instance) contracts;       // 어드레스별 계약서 관리
    uint counter;                               // 전체 계약서 수수
    uint commision;                             // 수수료

    constructor () {
        owner = msg.sender;
        counter = 0;
        commision = 10000000000000000;          // 기본 설정 : 0.01 Ether
    }
```

Instance (암호 계약서의 구조체)

- A : 갑, 고용주, 사용자 등의 어드레스를 저장
- B : 을, 공급자 등의 어드레스를 저장
- content : 계약서의 본문에 해당하는 내용을 저장
- date: 계약서 작성일을 저장

owner, contracts, counter, commision는 주석과 동일

생성자(constructor)

- 스마트 컨트랙트 소유주, 전체 계약서 개수, 수수료 등 Deploy 시 초기 설정값 지정

Solidity (스마트 컨트랙트) 소스 코드 - 계약서 조회 및 등록

```
// 계약서 조회
function getContract() view public returns (address[] memory, address[] memory, string[] memory, string[] memory) {
    address[] memory _A = new address[](counter);
    address[] memory _B = new address[](counter);
    string[] memory _content = new string[](counter);
    string[] memory _date = new string[](counter);

    for(uint i = 0; i < counter; i++) {
        _A[i] = contracts[i].A;
        _B[i] = contracts[i].B;
        _content[i] = contracts[i].content;
        _date[i] = contracts[i].date;
    }

    return (_A, _B, _content, _date);
}

// 계약서 등록
function addContract(address _A, address _B, string memory _content, string memory _date) public payable {
    if (msg.value != commision) {
        revert();
    }

    contracts[counter++] = Instance(_A, _B, _content, _date);
}
```

getContract() (전체 계약서 조회 함수)

- Input : 없음
- Output : 4개의 배열 (A, B, Content, Date)
- mapping과 counter를 통해 전체 계약서 읽기
- 4개의 배열로 변환 후 리턴

addContract(A, B, Content, Date) (계약서 등록 함수)

- Input : Address A, B, String Content, Date + msg.value
- Output : 없음
- 호출자가 송금한 금액이 수수료와 같은 지 체크
- 같다면 계약서 관리 변수에 암호 계약서 추가

Solidity (스마트 컨트랙트) 소스 코드 - 소유주, 잔고, 수수료 조회 및 회수(출금)

```
// 등록 수수료 조회
function getCommision() view public returns (uint) {
    return commision;
}

// 소유주 조회
function getOwner() view public returns (address) {
    return owner;
}

// 계약 잔고 조회
function getBalance() view public returns (uint) {
    return address(this).balance;
}

// 전액 잔고 회수
function withdraw() public payable {
    require(msg.sender == owner);

    if (!payable(owner).send(address(this).balance)) {
        revert();
    }
}
```

getCommision() (계약서 등록 수수료 조회)
- Input : 없음 / Output : uint 타입의 Wei 단위 수수료

getOwner() (스마트 컨트랙트 소유주 조회)
- Input : 없음 / Output : 소유주의 어드레스

getBalance() (스마트 컨트랙트 잔고 조회)
- Input : 없음 / Output : Wei 단위의 스마트 컨트랙트의 잔고

withdraw() (스마트 컨트랙트 잔고 회수)
- Input : 없음 / Output : 없음
- 스마트 컨트랙트 소유주 어드레스로 잔고 송금

Node.js 웹 서버 소스 코드

```
'use strict';

var http = require("http");
var fs = require('fs');
var url = require("url")

async function gateWayPage(req, res) {
    var fname = "./" + url.parse(req.url).pathname;
    console.log(fname)

    fs.readFile(fname, async function (err, data) {
        if(err) {
            res.writeHead(404, {'Content-Type': 'text/html'});
            return res.end("404 Not Found");
        }

        res.writeHead(200);
        res.write(data);
        return res.end();
    });
}
http.createServer(gateWayPage).listen(8080);
```

- use strict : 엄격(타이트)한 설정 (디버깅, 속도 유리)
- http, fs, url : HTTP 통신을 통해 url parsing으로 웹 서버 파일 기준의 경로로 파일 접근
- Response의 Content-Type을 HTML 파일로 지정
  즉, localhost:8080/<HTML_파일명>.html 로 렌더링 가능
- 정상적으로 경로 내 파일이 존재하면 Status Code 200 반환
  로드에 실패한 경우에는 Status Code 404 반환

HTML (웹 서비스) 소스코드 - 개요

[CSS]

UI를 위한 Bootstrap 4.6.1 사용

style.css 를 통해 HTML 코드 데코레이션

[Javascript]

메타마스크를 통한 이더리움 스마트 컨트랙트
접근 및 제어를 위해 Web3 모듈 Import
(npm install web3)

Bootstrap의 여러 기능을 위한 jQuery Import

HTML 코드만 보고 파악 가능하도록
ABI, Address 분리 X

ABI에 따른 함수 구현은 뒤에서 설명

HTML (웹 서비스) 소스코드 - 문서 로드 시 동작할 함수

```
var accounts;
var contractInstance;
var A;
var B;
var content;
var topDiv;

window.addEventListener('load', secretContract);

async function secretContract() {
    if (window.ethereum) {
        window.web3 = new Web3(window.ethereum);
        window.accounts = await ethereum.request({ method: 'eth_requestAccounts' });
        var address = "0x302D7b27BF838B0C066A08Cee0dAF1892aD4Ff08";
        var abi = [···
        contractInstance = await new window.web3.eth.Contract(abi, address);
    } else if (window.web3) {
        window.web3 = new Web3(web3.currentProvider);
    } else {
        console.log('Non-Ethereum browser detected. You should consider trying MetaMask!');
        window.web3 = null
    }
    addEvtHandlers();
}
```

[변수]

여러 함수에서 전역적으로 사용하기 위한 HTML Element를 지정

[secretContract]

HTML 로드 완료 시 실행할 함수

Web3 호환 가능 브라우저인 경우 연결

스마트 컨트랙트의 어드레스, ABI 선언

스마트 컨트랙트 객체 가져오기

↓

Input, 버튼 등 이벤트 핸들러 등록 함수 호출

HTML (웹 서비스) 소스코드 - 이벤트 핸들러 등록

```
function addEvtHandlers() {
    let btn = document.getElementById("getBalance");
    btn.addEventListener("click", getBalance);

    btn = document.getElementById("getContract");
    btn.addEventListener("click", getContract);

    btn = document.getElementById("addContract");
    btn.addEventListener("click", addContract);

    btn = document.getElementById("getCommision");
    btn.addEventListener("click", getCommision);

    btn = document.getElementById("withdraw");
    btn.addEventListener("click", withdraw);

    A = document.getElementById("A");
    B = document.getElementById("B");
    content = document.getElementById("content");
    topDiv = document.getElementById("parent");
}
```

[addEvtHandlers]

ABI에 맞는 함수를 호출하는 버튼에 클릭 이벤트 핸들러를 추가

버튼의 ID는 스마트 컨트랙트 호출 함수와 동일하게 지정

함수 호출시 ID값을 이용해 쉽게 호출 가능

Input 3개의 ID를 각각 A, B, content로 지정 후 getElement

topDiv는 전체 계약서 조회 시 목록을 동적으로 추가하기 위한 최상위 div 태그

HTML (웹 서비스) 소스코드 - getBalance / getCommision

```
async function getBalance() {
    if (contractInstance != null) {
        try {
            const value = await contractInstance.methods[this.id]().call();
            alert("현재 스마트 컨트랙트의 잔액은 " + Web3.utils.fromWei(value, 'ether') + "ETH 입니다.");
        } catch (e) {
            alert(e);
        }
    }
}

async function getCommision() {
    if (contractInstance != null) {
        try {
            const value = await contractInstance.methods[this.id]().call();
            alert("보안 계약서의 등록 수수료는 " + Web3.utils.fromWei(value, 'ether') + "ETH 입니다.");
        } catch (e) {
            alert(e);
        }
    }
}
```

[getBalance]

스마트 컨트랙트의 getBalance 함수 호출

address(this).balance를 가져와 출력

(Wei 단위를 Ether 단위로 변환)

[getCommision]

스마트 컨트랙트의 getCommision 함수 호출

commision을 가져와 출력

(Wei 단위를 Ether 단위로 변환)

HTML (웹 서비스) 소스코드 - getContract (1)

```
async function getContract() {
    if (contractInstance != null) {
        try {
            const value = await contractInstance.methods[this.id]().call();
            console.log(value);

            while (topDiv.hasChildNodes()) {
                topDiv.removeChild(topDiv.firstChild);
            }

            if (value[0].length == 0) {
                alert("현재 스마트 컨트랙트에 등록된 계약서가 없습니다.");
                return;
            }
```

[getContract]

전체 보안 계약서 리스트를 가져오는 함수

이미 불러와진 내용을 미리 지움

스마트 컨트랙트 내에 저장된 인스턴스가
없을 경우 Alert한 뒤 함수 종료

HTML (웹 서비스) 소스코드 - getContract (2)

```
for(let i = 0; i < value[0].length; i++) {
    var a_Tag = document.createElement("a");
    a_Tag.setAttribute("href", "javascript:void(0);");
    a_Tag.setAttribute("class", "list-group-item list-group-item-action");

    var a_div_Tag = document.createElement("div");
    a_div_Tag.setAttribute("class", "d-flex w-100 justify-content-between");

    var a_div_h5_Tag = document.createElement("h5");
    a_div_h5_Tag.innerHTML = "";

    var a_div_small_Tag = document.createElement("small");
    a_div_small_Tag.setAttribute("class", "text-muted");
    a_div_small_Tag.innerHTML = value[3][i];

    a_div_Tag.appendChild(a_div_h5_Tag);
    a_div_Tag.appendChild(a_div_small_Tag);

    var a_p1_Tag = document.createElement("p");
    a_p1_Tag.setAttribute("class", "mb-1 font-weight-bold");
    a_p1_Tag.innerHTML = "A : " + value[0][i];

    var a_p2_Tag = document.createElement("p");
    a_p2_Tag.setAttribute("class", "mb-1 font-weight-bold");
    a_p2_Tag.innerHTML = "B : " + value[1][i];

    var a_small_Tag = document.createElement("small");
    a_small_Tag.innerHTML = value[2][i];

    a_Tag.appendChild(a_div_Tag);
    a_Tag.appendChild(a_p1_Tag);
    a_Tag.appendChild(a_p2_Tag);
    a_Tag.appendChild(a_small_Tag);

    topDiv.appendChild(a_Tag);
}
```

[이어서...]

document.createElement 함수를 통해 HTML 태그 동적 생성

setAttribute를 통해 속성 값 설정

appendChild를 통해 부모 / 자식 설정

value 에는 4개의 배열이 담겨 있음

value[0]
Instance의 A(계약자 어드레스)가 담긴 배열

value[1]
Instance의 B(계약자 어드레스)가 담긴 배열

value[2]
암호 계약서의 본문 내용이 담긴 배열

value[3]
계약서의 생성(등록) 날짜가 담긴 배열

HTML (웹 서비스) 소스코드 - addContract (송금)

```
async function addContract() {
    if (contractInstance != null) {
        if (A.value == "" || B.value == "" || content.value == "") {
            alert("주소 또는 내용을 입력해주세요.");
            return;
        }
        try {
            var raw_content = content.value.replaceAll(/(\n|\r\n)/g, "<br>");
            var now = new Date(+new Date() + 3240 * 10000).toISOString().replace("T", " ").replace(/\..*/, '');
            let addressA = web3.utils.toChecksumAddress(A.value);
            let addressB = web3.utils.toChecksumAddress(B.value);

            const commision = await contractInstance.methods["getCommision"]().call();
            const value = await contractInstance.methods[this.id](addressA, addressB, raw_content, now).send({from:accounts[0], value:commision});

            alert("계약서가 정상적으로 등록됐습니다.");

            A.value = "";
            B.value = "";
            content.value = "";
        } catch (e) {
            alert(e);
        }
    }
}
```

[addContract]

Input 값들이 작성됐는지, A/B가 유효한 어드레스인지 검증

입력된 Content의 포맷을 유지하기 위해 변환

날짜 및 시간을 담는 변수 사용

함수 호출 시 위에서 정의한 변수를 파라미터로 전송

스마트 컨트랙트 내용을 변경하기 때문에 send 함수
(가스가 소모되기 때문에 누가 요청하는지 from 키로 전달)

가스와 별도로 계약서 등록 시의 수수료를 지불해야 하기
때문에 value 키에 수수료를 담아 전송

HTML (웹 서비스) 소스코드 - withdraw (인출/반환)

```
async function withdraw() {
    if (contractInstance != null) {
        try {
            const value = await contractInstance.methods[this.id]().send({from: accounts[0]});
            alert("스마트 컨트랙트 소유주의 어드레스로 송금 됐습니다.");
        } catch (e) {
            console.log(e);
        }
    }
}
```

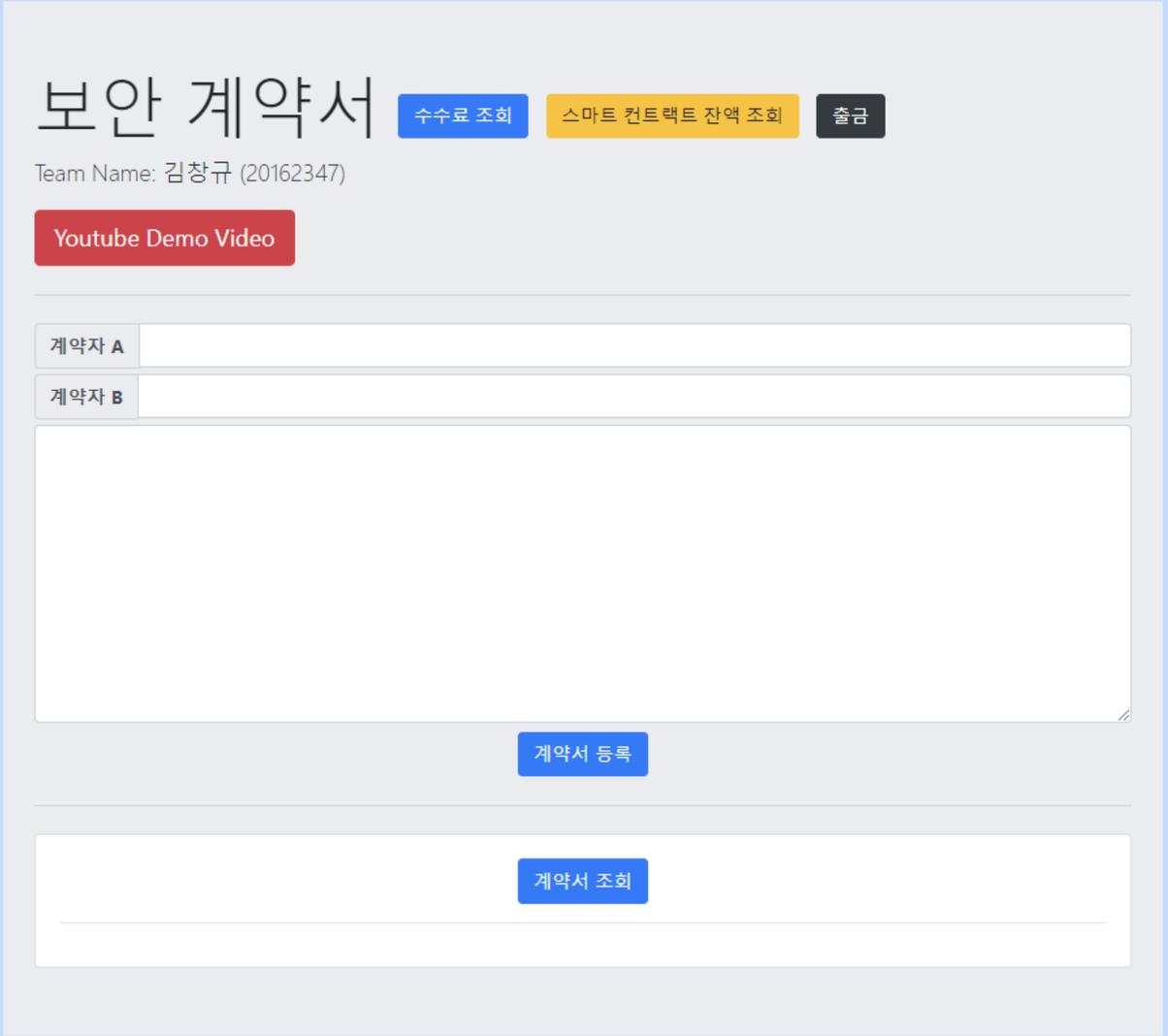
[withdraw]

암호 계약서 등록 시 마다 송금되어 쌓인 잔고를 반환

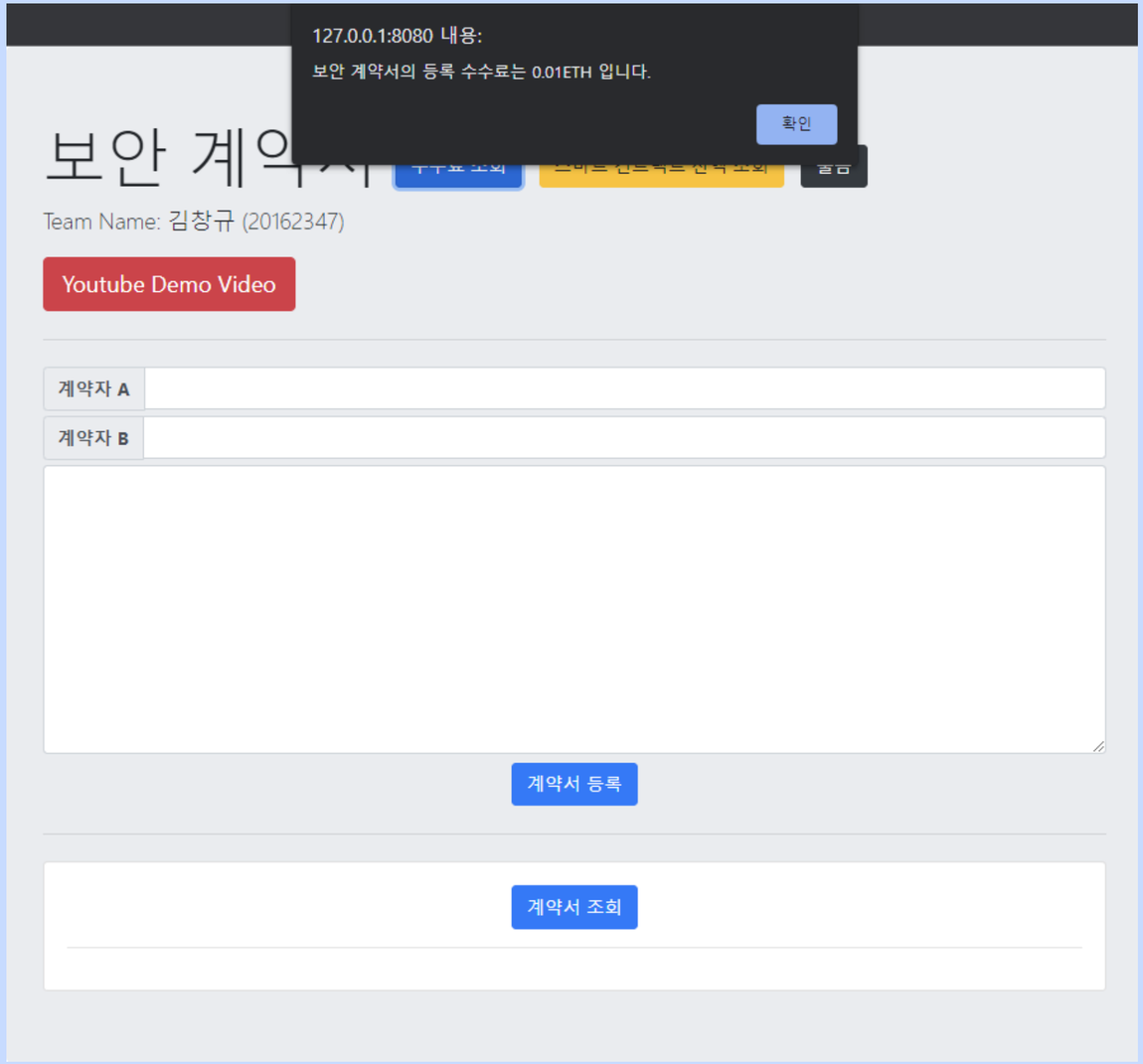
스마트 컨트랙트의 잔고를 변경시키므로 가스 소모
따라서, send() 함수와 from: accounts[0]

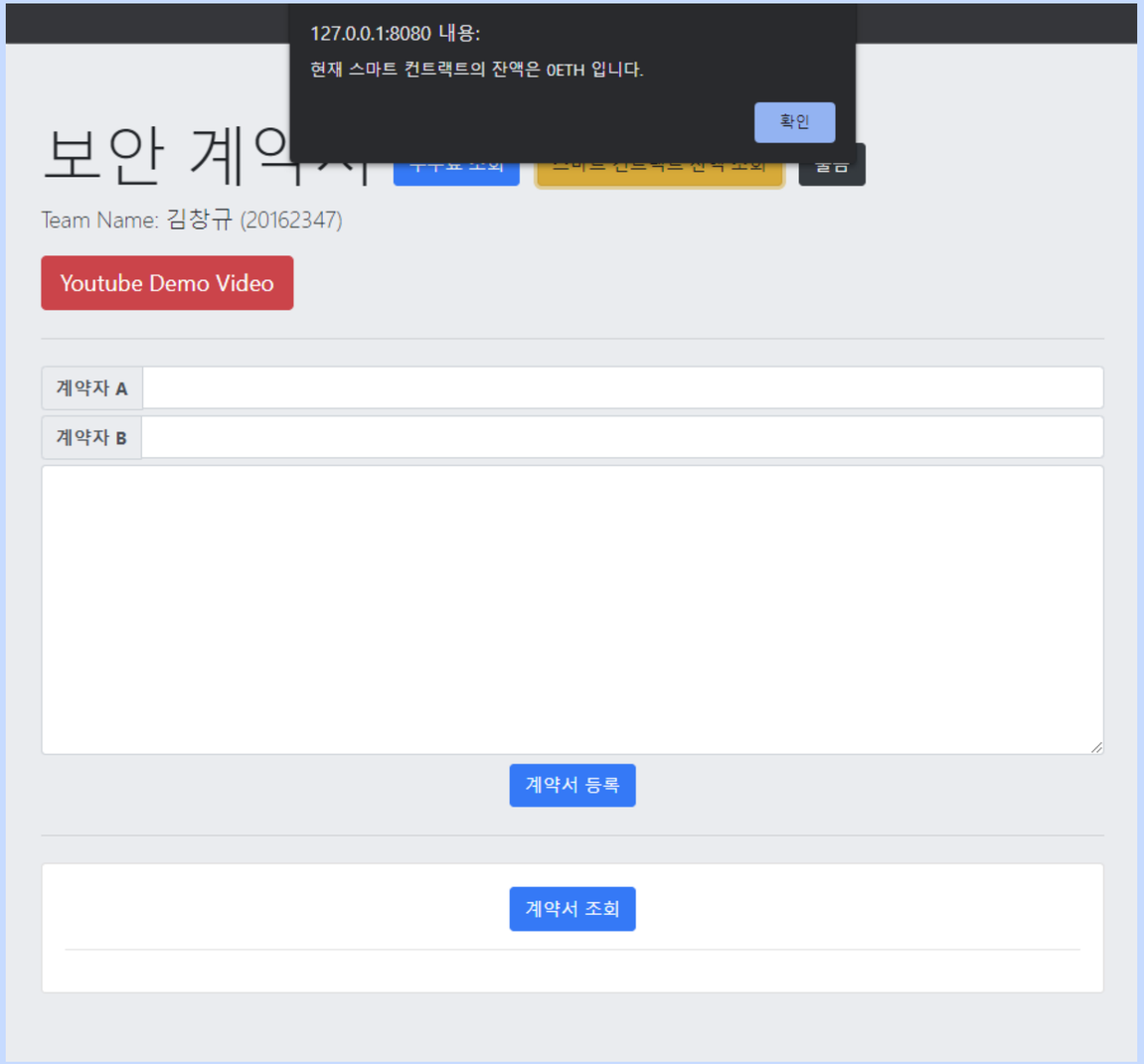
스마트 컨트랙트 내에서 msg.sender == owner
조건이 있으므로 소유주가 아닐 경우 함수 실행이 취소

기능별 실행 화면 - 메인 화면

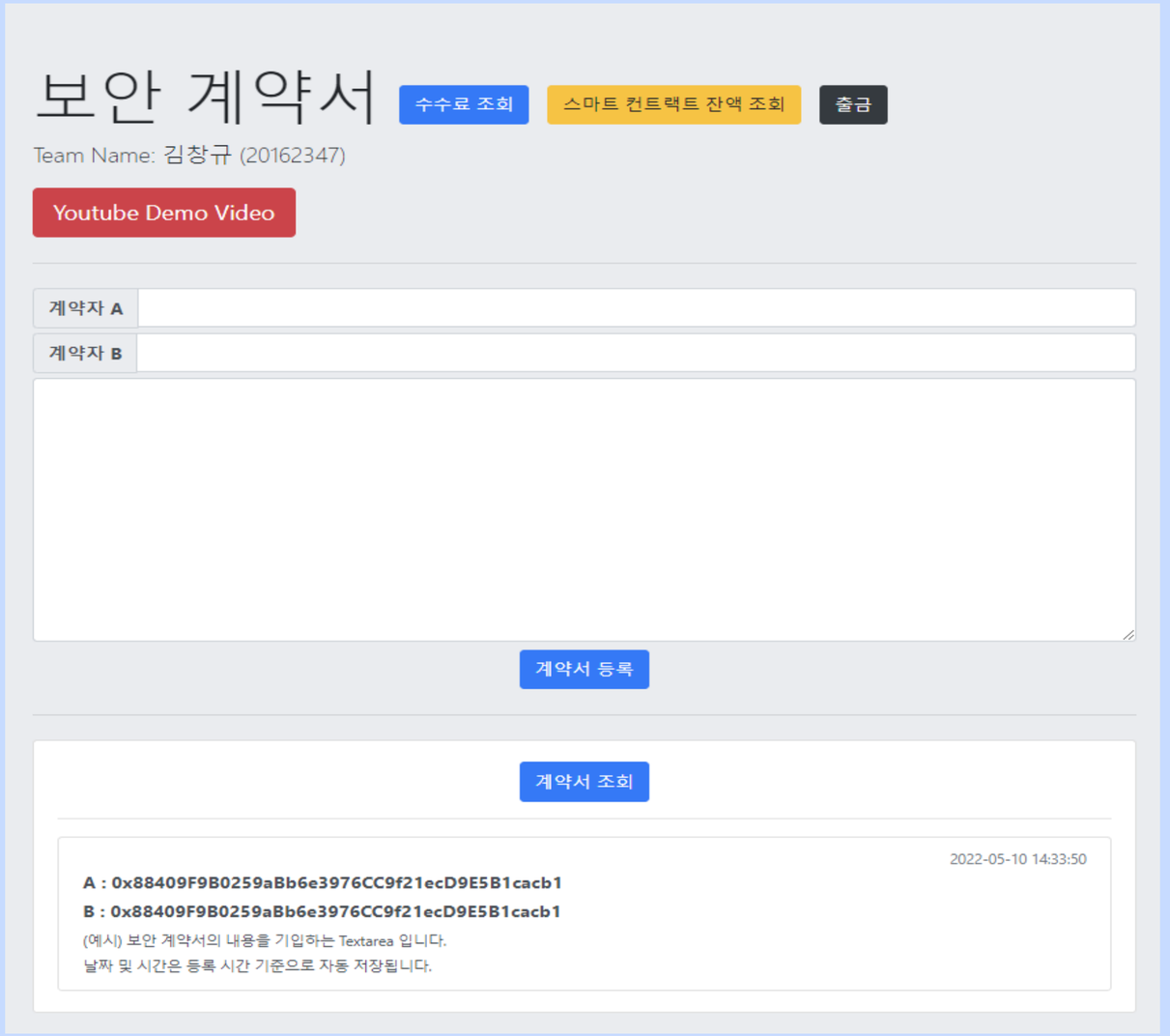


기능별 실행 화면 - 수수료 조회 및 잔액 조회





기능별 실행 화면 - 계약서 조회



기능별 실행 화면 - 계약서 등록 및 출금

가스 소모 및 추가 수수료 송금으로 인해 Youtube 시연 영상에서 확인 가능