APPENDIX

Appendix A. Solidity Code

GitHub can be obtained at the following link : https://github.com/urahardja/EESP

GitHub

EESP ⇒ Master ⇒ Controllers ⇒ Diplomas

```
'use strict';
require('dotenv').config();
const fs = require('fs').promises;
const { Web3Storage, File } = require('web3.storage');
const { ethers } = require('ethers');
const formidable = require('formidable');

const DiplomasContract = require('../contracts/Diplomas.json').abi;

const LOCAL_DEVELOPMENT = process.env.LOCAL_DEVELOPMENT;
const LOCAL_WALLET_PRIVATE_KEY = process.env.LOCAL_WALLET_PRIVATE_KEY;
const LOCAL_CONTRACT_ADDRESS = process.env.LOCAL_CONTRACT_ADDRESS;
const ALCHEMY_NETWORK = process.env.ALCHEMY_NETWORK;
const ALCHEMY_API_KEY = process.env.ALCHEMY_API_KEY;
const WALLET_PRIVATE_KEY = process.env.WALLET_PRIVATE_KEY;
const CONTRACT_ADDRESS = process.env.CONTRACT_ADDRESS;
const WEB3STORAGE_TOKEN = process.env.WEB3STORAGE_TOKEN;

const createContract = () => {
  const provider = LOCAL_DEVELOPMENT
    ? new ethers.getDefaultProvider('http://127.0.0.1:8545')
    : new ethers.providers.AlchemyProvider(ALCHEMY_NETWORK, ALCHEMY_API_KEY);

  const wallet = LOCAL_DEVELOPMENT
    ? new ethers.Wallet(LOCAL_WALLET_PRIVATE_KEY, provider)
    : new ethers.Wallet(WALLET_PRIVATE_KEY, provider);

  return LOCAL_DEVELOPMENT
    ? {
        r: new ethers.Contract(
          LOCAL_CONTRACT_ADDRESS,
          DiplomasContract,
          provider
        ),
        rw: new ethers.Contract(
          LOCAL_CONTRACT_ADDRESS,
          DiplomasContract,
          wallet
        )
      }
    : {
        r: new ethers.Contract(CONTRACT_ADDRESS, DiplomasContract, provider),
        rw: new ethers.Contract(CONTRACT_ADDRESS, DiplomasContract, wallet)
      };
};
const contracts = createContract();

const parseForm = (form) => {
```

```
  return new Promise((resolve) => {
    parse(form, (error, fields, files) => {
      if (error)
        return { status: 'Error', message: 'Error parsing uploaded file' };

      if (!fields.name)
        return {
          status: 'Error',
          message: 'Name required'};

      if (!fields.NIM)
        return {
          status: 'Error',
          message: 'NIM required'
        };

      if (!files.file)
        return { status: 'Error', message: 'No file uploaded' };

      resolve({
        name: fields.name,
        NIM: fields.NIM,
        filePath: files.file.path
      });
    });
  });
};

const onRootCidReady = (hash) => {
  throw hash;
};
const web3storage = new Web3Storage({ token: WEB3STORAGE_TOKEN });

exports.validate = async (DocInfo) => {
  const file = await fs.readFile(filePath);
  const metadata = {
    DocInfo.eesp_hash,
    DocInfo.date_added
  };
  const files = [
    new File([file], 'diploma'),
    new File([JSON.stringify(metadata)], 'metadata')
  ];

  const eesp_hash = addToIPFS(files);
  const contractResult = await contracts.rw.add(eesp_hash);

      return{
        status: 'OK',
        message: 'File uploaded successfully',
        data: contractResult
      };
    };
};
```

```
exports.verify = async (req, res) => {
  const { filePath } = await parseForm(req, res, {
    checkName: false,
    checkNIM: false
  });

  const file = await fs.readFile(filePath);
  const files = [new File([file], 'diploma')];

  web3storage
    .put(files, { wrapWithDirectory: false, onRootCidReady })
    .catch(async (hash) => {
      const contractResult = await contracts.r.get(hash);

      res.send({
        status: 'OK',
        message: contractResult ? 'File is authentic' : 'File is not authentic',
        data: contractResult
      });
    });
};

exports.get = async (fileHash) => {
  const contractResult = await contracts.r.get(fileHash);
  return{
    status: 'OK',
    message: `Data found`,
    data: {
      eesp: contractResult.eespHash,
      eesp: contractResult.dateAdded
    }
  };
};

exports.get = async (eespHash) => {
  const Result = await getIPFS(eespHash);
  return{
    status: 'OK',
    message: `Data found`,
    data: {
      eesp: result
    }
  };
};
```

GitHub

EESP ⇒ Contract ⇒ Contracts ⇒ Diplomas

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.7.0;

contract assets {
  mapping (string => string) eespHash;

  function add(string memory _cid) public {
```

```
    eespHash[_cid] = _cid;
  }

  function get(string memory _cid) view public returns (string memory) {
    return eespHash[_cid];
  }
}
```

Appendix B. Simulation Results

Simulation Results can be obtained at the following link :
https://docs.google.com/document/d/1pZwU1Jg6n2GuXTxMsJbiZloHMKG5v06ahKPBrXLwUog/edit?usp=sharing