



DALHOUSIE UNIVERSITY

CSCI6313 Introduction to Blockchains

ASSIGNMENT – 1 PART - B

Git Repository Link: <https://github.com/KrishnaVaibhav/Blockchain-Assignment-Part2>

Name: Krishna Vaibhav Yadlapalli

Banner ID: B00968432

Table of Contents

Assignment-1 Part-B.....	3
Smart Contract:	3
Truffle Contract Deployment:	3
Explanation of Index.js:	4
Output	5
References	7

Assignment-1 Part-B

Smart Contract:



```

1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 contract documents432 {
5     struct Document432 {
6         string docHash;
7         string content;
8     }
9
10    mapping(string => Document432) private documents;
11
12    function storeDocument(string memory id, string memory content, string memory docHash) public {
13        documents[id] = Document432(docHash, content);
14    }
15
16    function getDocument(string memory id) public view returns (string memory, string memory) {
17        return (documents[id].content, documents[id].docHash);
18    }
19 }
20

```

Figure (1): Solidity Code for smart Contract.

- document432: The name of the smart contract.
- struct Document432: A structure to hold document data (content and its hash).
- mapping: A storage mechanism that links a document ID to a Document.
- storeDocument: A function to save a document and its hash.
- getDocument: A function to retrieve a document and its hash.

Truffle Contract Deployment:

Truffle initiated:



```

1 module.exports = {
2     networks: {
3         development: {
4             host: "127.0.0.1",
5             port: 7545,
6             network_id: "*"
7         }
8     },
9     compilers: {
10        solc: {
11            version: "0.8.0"
12        }
13    }
14 };
15
16

```

Figure (2): Truffle config file.

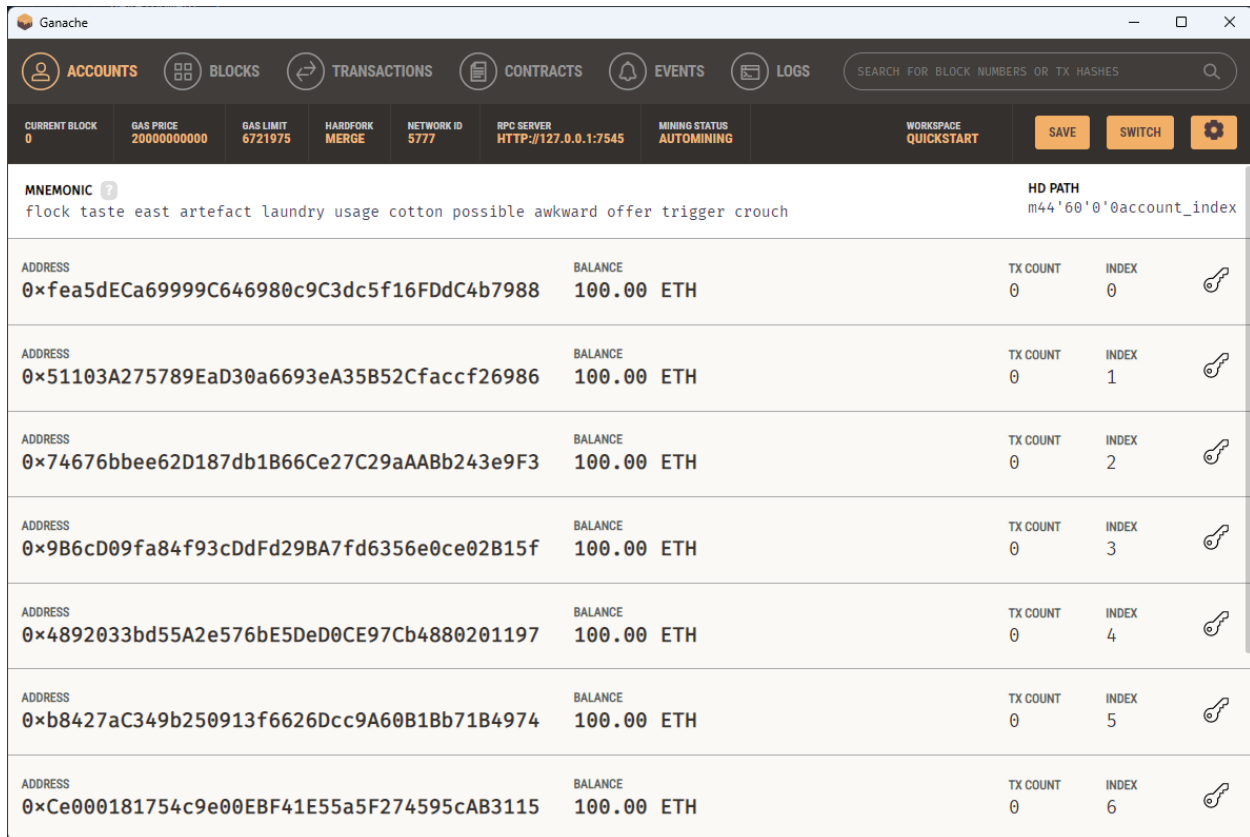


Figure (3): Deployed Blockchain locally using Ganache.

```

57  async function storeDocument432(docId, content) {
58      const docHash = web3.utils.sha3(content);
59      const accounts = await web3.eth.getAccounts();
60      const result = await contract.methods.storeDocument(docId, content, docHash).send({ from: accounts[0], gas: 3000000 });
61      console.log('[B00968432] - Document stored successfully.');
```

```

62      console.log('[B00968432] - Document Hash:', docHash);
63      console.log('[B00968432] - Transaction:', result);
64  }
65
66  async function getDocument432(docId) {
67      const result = await contract.methods.getDocument(docId).call();
68      console.log('[B00968432] - Document Content:', result[0]);
69      console.log('[B00968432] - Document Hash:', result[1]);
70      console.log('[B00968432] - Transaction:', result);
71  }

```

Figure (4): Code to store and get document with hash of the document.

Explanation of Index.js:

- Connect to Ganache: new Web3('http://127.0.0.1:7545') connects to the local blockchain provided by Ganache.
- Contract ABI and address: contractABI is the contract's interface, and contractAddress is the deployed contract's address taken from the json file in the deploy file.
- Functions:
 - storeDocument432: Stores a document's content and its hash.
 - getDocument432: Retrieves a document's content and its hash.
- Command-line arguments:
 - store: Stores a document.
 - get: Retrieves a document.

The screenshot shows the VS Code interface with the following details:

- Explorer Panel:** Displays the project structure. The 'contracts' folder is expanded, showing 'documents432.json' and 'migrations432.json'. The 'migrations' folder is also expanded, showing '1_initial_migration.js' and '2_deploy_contracts.js'.
- Main Editor:** Shows the 'index.js' file. The code defines a 'contract' variable and a 'deploy' function that uses 'web3.eth.Contract' and 'truffle.deploy' to deploy the contract.
- Terminal Panel:** Shows the output of the deployment command. It includes the transaction hash, block details, and the final cost of the deployment.

Output

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows the project structure. The 'contracts' folder is expanded, showing 'documents432.sol' and 'migrations'. The 'migrations' folder is also expanded, showing '1_initial_migration.js' and '2_deploy_contracts.js'.
- Main Editor:** Displays the 'index.js' file. The code defines a 'storeDocument432' function that takes 'docId' and 'content' as arguments. It uses 'web3' to interact with the blockchain, including getting accounts and sending transactions.
- Terminal:** Shows the command 'node index.js store "documnet432" "Student ID: B00968432"' being executed. The output shows the transaction hash '0x929ae483e6633db446472ece2240d312aef146825c0983301d5fda3fce3804' and the gas price '3191344709n'.

KRISHNA VAIBHAV YADLAPLLI

The screenshot shows the VS Code editor with the following components:

- Explorer:** Displays the project structure. Folders include 'build', 'contracts', 'migrations', and 'node_modules'. Files include 'documents432.json', 'Migrations432.json', 'contracts', 'documents432.sol', 'Migrations432.sol', 'migrations', '.gitkeep', '1_initial_migration.js', '2_deploy_contracts.js', 'package-lock.json', 'package.json', and 'truffle-config.js'.
- index.js:** Contains the implementation of the `storeDocument432` function. It uses `web3.eth.Contract` to interact with the `Document432` contract. The function takes `docId` and `content` as arguments, calculates a document hash, and sends a transaction to the contract.
- Terminal:** Shows the command `D:\U\Sem - #\Blockchain\LI>node index.js get "document432"` and its output. The output lists document details for ID `B00968432`, including its hash, gas price, and transaction details.

Figure (7): Document retrieved successfully.

Both the store document and get document functions are working successfully, **the hash of the document before and after getting the document are the same as shown in figures 6 and 7** maintaining the integrity of the file.

Git Repository Link: <https://github.com/KrishnaVaibhav/Blockchain-Assignment-Part2>.

References

- [1] Drchrisliu, “DRCHRISLIU/6313-IPFs,” *GitHub*, <https://github.com/drchrisliu/6313-ipfs> (accessed May 22, 2024).