



# DALHOUSIE UNIVERSITY

## CSCI6313 Introduction to Blockchains

### ASSIGNMENT – 1 PART - B

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.

MNEMONIC		HD PATH	
flock taste east artefact laundry usage cotton possible awkward offer trigger crouch		m44'60'0'0account_index	
ADDRESS	BALANCE	TX COUNT	INDEX
0xfea5dECa69999C646980c9C3dc5f16FDdC4b7988	100.00 ETH	0	0
0x51103A275789EaD30a6693eA35B52Cfaccf26986	100.00 ETH	0	1
0x74676bbe62D187db1B66Ce27C29aAABb243e9F3	100.00 ETH	0	2
0x9B6cD09fa84f93cDdFd29BA7fd6356e0ce02B15f	100.00 ETH	0	3
0x4892033bd55A2e576bE5DeD0CE97Cb4880201197	100.00 ETH	0	4
0xb8427aC349b250913f6626Dcc9A60B1Bb71B4974	100.00 ETH	0	5
0xCe000181754c9e00EBF41E55a5F274595cAB3115	100.00 ETH	0	6

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 components:

- 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:** Displays the 'index.js' file. The code defines a 'deploy' function that uses 'web3.eth.Contract' to create a contract instance and 'storeDocument' to save documents. The function is called with 'documents432.json' as the document ID.
- Terminal Panel:** Shows the output of the deployment command. It includes the transaction hash, block number, gas used, gas price, value sent, and total cost. The output is as follows:
 

```

      Deploying 'documents432'
      -----
      > transaction hash: 0x94681526a4470a0dde9543fa65a637a4fb327b5c2191fdbabb7d95f8b60735270
      > Blocks: 0
      > contract address: 0x9870683dc99A94A1cc8e3D3E7964975E5D4bff771
      > block number: 2
      > block timestamp: 1716416161
      > account: 0xdF83f917f2A808E20AeF41B3F3a36188fA4A0365
      > balance: 99.99752182208441141
      > gas used: 418422 (0x66276)
      > gas price: 3.276301845 gwei
      > value sent: 0 ETH
      > total cost: 0.00137087677058859 ETH

      > Saving artifacts
      -----
      > Total cost: 0.00137087677058859 ETH

      Summary
      =====
      > Total deployments: 2
      > Final cost: 0.00247817714558859 ETH
      
```

## Output

KRISHNA VAIBHAV YADLAPLLI

[illegible]

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.

## References

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