

CSCI 6313

ASSIGNMENT – 1 B

**By
Judith Kurian
(B00940475)**

Table of Contents

Smart Contract Algorithm.....	3
API for Smart Contract	4
Screenshots:	9
Validation on DAPP Execution Result.....	23

Smart Contract Algorithm

Algorithm for DocumentStorage Contract

Initialization

1. Define Structure:

- Define a `Document` structure with fields `content` (string) and `hash` (bytes32).

2. State Variables:

- Initialize a mapping `documents` that maps an address to a `Document`.

3. Events:

- Define `DocumentStored475` event with parameters: `user` (address) and `hash` (bytes32).
- Define `DocumentRetrieved475` event with parameters: `user` (address), `content` (string), and `hash` (bytes32).

Functions

Store Document

1. Function Declaration:

- Define a public function `storeDocument475` that accepts a string parameter `_content`.

2. Compute Hash:

- Compute the keccak256 hash of `_content` and store it in a variable `hash`.

3. Store Document:

- Create a `Document` instance with `_content` and `hash`.
- Store this document in the `documents` mapping with `msg.sender` as the key.

4. Emit Event:

- Emit the `DocumentStored` event with `msg.sender` and `hash`.

Get Document

1. Function Declaration:

- Define a public view function `getDocument475` that returns a tuple containing:
 - `content` (string)
 - `storedHash` (bytes32)
 - `integrity` (bool)

2. Retrieve Document:

- Retrieve the document associated with ``msg.sender`` from the ``documents`` mapping and store it in a variable ``doc``.

3. Compute Hash:

- Compute the keccak256 hash of ``doc.content`` and store it in a variable ``calculatedHash``.

4. Verify Integrity:

- Compare ``calculatedHash`` with ``doc.hash`` and store the result in ``integrityCheck`` (boolean).

5. Return Values:

- Return ``doc.content``, ``doc.hash``, and ``integrityCheck``.

Summary of Events and Data Flow

1. Storing a Document:

- User calls ``storeDocument`` with document content.
- Contract computes the hash of the content.
- Contract stores the document content and hash in the ``documents`` mapping.
- Contract emits ``DocumentStored`` event.

2. Retrieving a Document:

- User calls ``getDocument``.
- Contract retrieves the stored document for the user.
- Contract computes the hash of the stored content.
- Contract verifies if the stored hash matches the computed hash.
- Contract returns the document content, the stored hash, and the integrity check result.

API for Smart Contract

The API for the ``DocumentStorage`` smart contract consists of the available functions and events that can be interacted with. Here is a detailed description:

Functions

1. storeDocument

- Description: Stores a document for the caller and emits an event with the document's hash.
- Parameters:
 - ``_content`` (string): The content of the document to be stored.

- Returns: None

- Solidity Signature:

```
function storeDocument(string memory _content) public
```

2. getDocument

- Description: Retrieves the stored document for the caller, along with its hash and integrity check.

- Parameters: None

- Returns:

- `content` (string): The content of the stored document.

- `storedHash` (bytes32): The hash of the stored document.

- `integrity` (bool): A boolean indicating whether the stored document's hash matches the computed hash of the content.

- Solidity Signature:

```
function getDocument() public view returns (string memory content, bytes32 storedHash, bool integrity)
```

Events

1. DocumentStored

- Description: Emitted when a document is stored.

- Parameters:

- `user` (address, indexed): The address of the user who stored the document.

- `hash` (bytes32): The hash of the stored document.

- Solidity Signature:

```
event DocumentStored(address indexed user, bytes32 hash)
```

2. DocumentRetrieved

- Description: Emitted when a document is retrieved. This event is not present in the current implementation but is useful for tracking document retrievals.

- Parameters:

- `user` (address, indexed): The address of the user who retrieved the document.

- `content` (string): The content of the retrieved document.

- `hash` (bytes32): The hash of the retrieved document.

- Solidity Signature:

```
event DocumentRetrieved(address indexed user, string content, bytes32 hash)
```

Example Usage

Storing a Document

To store a document, you would call the `storeDocument` function with the document content as a parameter.

```
contractInstance.storeDocument("This is my document content.");
```

Retrieving a Document

To retrieve a document, you would call the `getDocument` function, which returns the content, hash, and integrity status.

```
(string memory content, bytes32 storedHash, bool integrity) =  
contractInstance.getDocument();
```

ABI (Application Binary Interface)

The ABI is a JSON representation of the contract's interface, which can be used to interact with the contract from a web3-enabled application (like a DApp). Here is the ABI for the `DocumentStorage` contract:

```
[  
  {  
    "inputs": [  
      {  
        "internalType": "string",  
        "name": "_content",  
        "type": "string"  
      }  
    ],  
    "name": "storeDocument",  
    "outputs": [],  
    "stateMutability": "nonpayable",  
    "type": "function"  
  },  
  {  
    "inputs": [],  
    "name": "getDocument",  
    "outputs": [  

```

```
{
  "internalType": "string",
  "name": "content",
  "type": "string"
},
{
  "internalType": "bytes32",
  "name": "storedHash",
  "type": "bytes32"
},
{
  "internalType": "bool",
  "name": "integrity",
  "type": "bool"
}
],
"stateMutability": "view",
"type": "function"
},
{
  "anonymous": false,
  "inputs": [
    {
      "indexed": true,
      "internalType": "address",
      "name": "user",
      "type": "address"
    },
    {
      "indexed": false,
```

```
        "internalType": "bytes32",
        "name": "hash",
        "type": "bytes32"
    }
],
"name": "DocumentStored",
"type": "event"
},
{
    "anonymous": false,
    "inputs": [
        {
            "indexed": true,
            "internalType": "address",
            "name": "user",
            "type": "address"
        },
        {
            "indexed": false,
            "internalType": "string",
            "name": "content",
            "type": "string"
        },
        {
            "indexed": false,
            "internalType": "bytes32",
            "name": "hash",
            "type": "bytes32"
        }
    ],
```



```

    "name": "DocumentRetrieved",
    "type": "event"
  }
]

```

This ABI can be used with web3.js or ethers.js to interact with the `DocumentStorage` smart contract deployed on an Ethereum-compatible blockchain.

Screenshots:

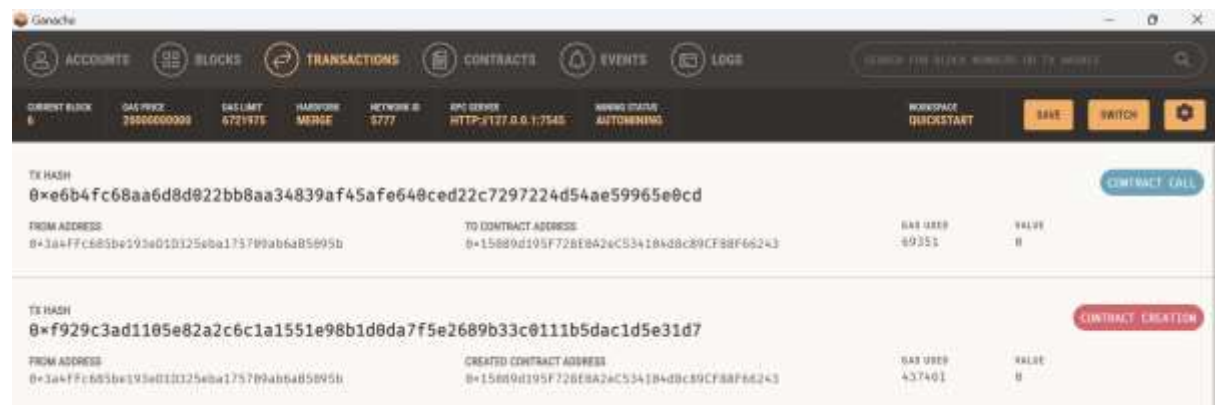


Figure 1: Ganache Transactions

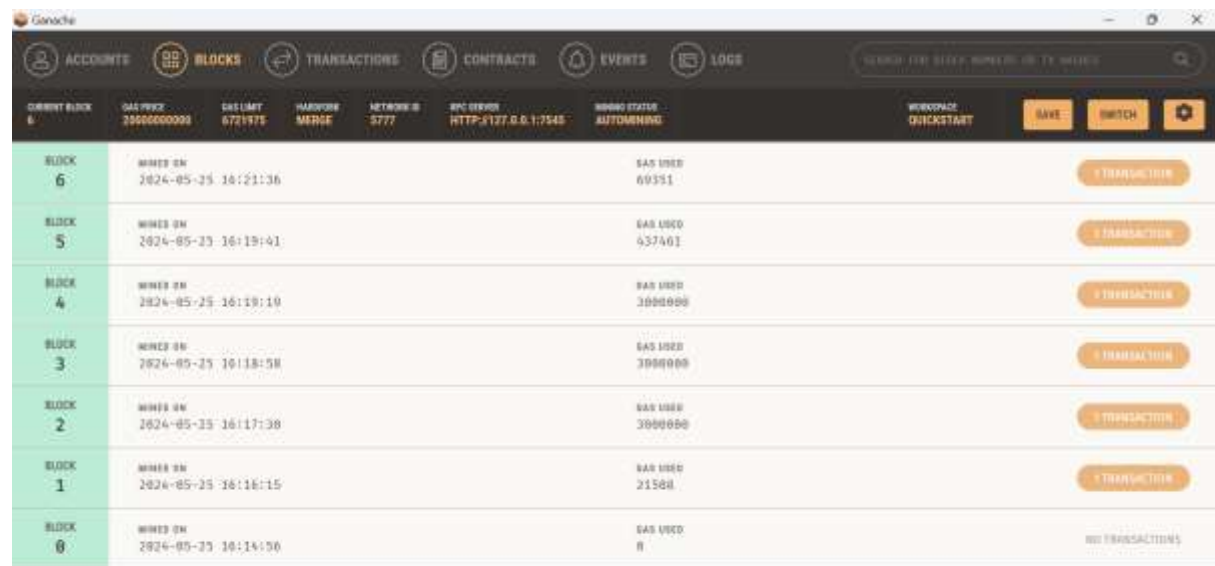


Figure 2: Ganache Blocks

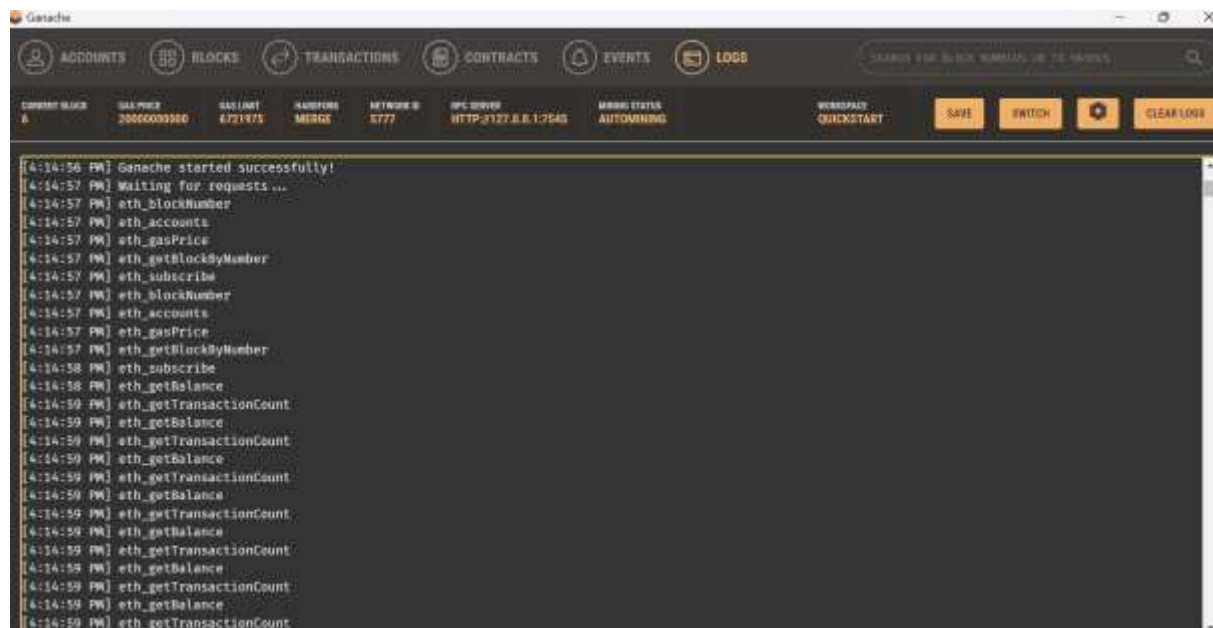


Figure 3: Ganache Logs

```
● PS C:\Personal Pro\Academics\Masters\Blockchain\Assignment 1> node index.js
0x3a4FFc685be193eD1D325eba175709ab6aB5095b
ContractBuilder {
  _emitter: EventEmitter {
    _events: Events <[Object: null prototype] {}> {},
    _eventsCount: 0,
    maxListeners: 9007199254740991
  },
  config: {
    handleRevert: false,
    defaultAccount: undefined,
    defaultBlock: 'latest',
    transactionBlockTimeout: 50,
    transactionConfirmationBlocks: 24,
    transactionPollingInterval: 1000,
    transactionPollingTimeout: 750000,
    transactionReceiptPollingInterval: undefined,
    transactionSendTimeout: 750000,
    transactionConfirmationPollingInterval: undefined,
    blockHeaderTimeout: 10,
    maxListenersWarningThreshold: 100,
    contractDataInputFill: 'data',
    defaultNetworkId: undefined,
    defaultChain: 'mainnet',
    defaultHardfork: 'london',
    defaultCommon: undefined,
    defaultTransactionType: '0x2',
    defaultMaxPriorityFeePerGas: '0x9502f900',
    enableExperimentalFeatures: {
      useSubscriptionWhenCheckingBlockTimeout: false,
      useRpcCallSpecification: false
    },
    transactionBuilder: undefined,
```

Figure 4: DAPP Execution 1

```

    },
    providers: {
      HttpProvider: [class HttpProvider extends Web3BaseProvider],
      WebsocketProvider: [class WebsocketProvider extends SocketProvider]
    },
    _requestManager: Web3RequestManager {
      _emitter: EventEmitter {
        _events: [Events <Complex prototype>],
        _eventsCount: 2,
        maxListeners: 9007199254740991
      },
      _provider: HttpProvider {
        clientUrl: 'http://127.0.0.1:7545/',
        httpProviderOptions: undefined
      },
      useRpcCallSpecification: undefined
    },
    _subscriptionManager: Web3SubscriptionManager {
      requestManager: Web3RequestManager {
        _emitter: [EventEmitter],
        _provider: [HttpProvider],
        useRpcCallSpecification: undefined
      },
      registeredSubscriptions: {
        logs: [class LogsSubscription extends Web3Subscription],
        newPendingTransactions: [class NewPendingTransactionsSubscription extends Web3Subscription],
        newHeads: [class NewHeadsSubscription extends Web3Subscription],
        syncing: [class SyncingSubscription extends Web3Subscription],
        pendingTransactions: [class NewPendingTransactionsSubscription extends Web3Subscription],
        newBlockHeaders: [class NewHeadsSubscription extends Web3Subscription]
      },
      tolerateUnlinkedSubscription: false,
      _subscriptions: Map(0) {}
    }
  }
}

```

Figure 5: DAPP Execution 2

```

    _subscriptions: Map(0) {}
  },
  _accountProvider: {
    signTransaction: [Function: signTransactionWithContext],
    create: [Function: createWithContext],
    privateKeyToAccount: [Function: privateKeyToAccountWithContext],
    decrypt: [Function: decryptWithContext],
    recoverTransaction: [Function: recoverTransaction],
    hashMessage: [Function: hashMessage],
    sign: [Function: sign],
    recover: [Function: recover],
    encrypt: [Function: encrypt],
    wallet: Wallet(0) [
      _accountProvider: [Object],
      _addressMap: Map(0) {},
      _defaultKeyName: 'web3js_wallet'
    ],
    privateKeyToAddress: [Function: privateKeyToAddress],
    parseAndValidatePrivateKey: [Function: parseAndValidatePrivateKey],
    privateKeyToPublicKey: [Function: privateKeyToPublicKey]
  },
  _wallet: Wallet(0) [
    _accountProvider: {
      create: [Function: createWithContext],
      privateKeyToAccount: [Function: privateKeyToAccountWithContext],
      decrypt: [Function: decryptWithContext]
    },
    _addressMap: Map(0) {},
    _defaultKeyName: 'web3js_wallet'
  ],
  syncWithContext: false,
  _functions: {
    'storeDocument475(string)': { signature: '0x4a2aa1d7', method: [Function (anonymous)] },

```

Figure 6: DAPP Execution 3

```

_functions: {
  'storeDocument475(string)': { signature: '0x4a2aa1d7', method: [Function (anonymous)] },
  'getDocument475()': { signature: '0x89a4ccb7', method: [Function (anonymous)] }
},
_overloadedMethodAbis: Map(2) {
  'storeDocument475' => [ [Object] ],
  'getDocument475' => [ [Object] ]
},
_methods: {
  storeDocument475: [Function (anonymous)],
  'storeDocument475(string)': [Function (anonymous)],
  '0x4a2aa1d7': [Function (anonymous)],
  getDocument475: [Function (anonymous)],
  'getDocument475()': [Function (anonymous)],
  '0x89a4ccb7': [Function (anonymous)]
},
_events: {
  'DocumentRetrieved(address,string,bytes32)': [Function (anonymous)],
  DocumentRetrieved: [Function (anonymous)],
  '0xb208da2131a1e4fa14f5e918eb0e88312d516dfa74cb624cfc395b5f8c9ebc5b': [Function (anonymous)],
  'DocumentStored(address,bytes32)': [Function (anonymous)],
  DocumentStored: [Function (anonymous)],
  '0x0b5964e012751f4988419a5f0dfdd1d3c3721eda68dd04aabc19b94cc1f4d0cf': [Function (anonymous)],
  allEvents: [Function (anonymous)]
},
_jsonInterface: [
  {
    anonymous: false,
    inputs: [Array],
    name: 'DocumentRetrieved',
    type: 'event',
    signature: '0xb208da2131a1e4fa14f5e918eb0e88312d516dfa74cb624cfc395b5f8c9ebc5b'
  }
],

```

Figure 7: DAPP Execution 4

```

    },
    {
      anonymous: false,
      inputs: [Array],
      name: 'DocumentStored',
      type: 'event',
      signature: '0x0b5964e012751f4988419a5f0dfdd1d3c3721eda68dd04aabc19b94cc1f4d0cf'
    },
    {
      inputs: [Array],
      name: 'storeDocument475',
      outputs: [],
      stateMutability: 'nonpayable',
      type: 'function',
      signature: '0x4a2aa1d7',
      methodNameWithInputs: 'storeDocument475(string)',
      constant: false,
      payable: false
    },
    {
      inputs: [],
      name: 'getDocument475',
      outputs: [Array],
      stateMutability: 'view',
      type: 'function',
      signature: '0x89a4ccb7',
      methodNameWithInputs: 'getDocument475()',
      constant: true,
      payable: false
    }
  ],
  _errorsInterface: [],
  _address: '0x15089d195F728E0A2eC534104d8c89CF88F66243',

```

Figure 8: DAPP Execution 5


```

_errorsInterface: [],
_address: '0x15089d195F728E0A2eC534104d8c89CF88F66243',
options: {
  address: [Getter/Setter],
  jsonInterface: [Getter/Setter],
  gas: undefined,
  gasPrice: undefined,
  from: '0x3a4FFc685be193eD1D325eba175709ab6aB5095b',
  input: undefined,
  data: undefined
},
context: Web3 {
  _emitter: EventEmitter {
    _events: [Events <Complex prototype>],
    _eventsCount: 1,
    maxListeners: 9007199254740991
  },
  config: {
    handleRevert: false,
    defaultAccount: undefined,
    defaultBlock: 'latest',
    transactionBlockTimeout: 50,
    transactionConfirmationBlocks: 24,
    transactionPollingInterval: 1000,
    transactionPollingTimeout: 750000,
    transactionReceiptPollingInterval: undefined,
    transactionSendTimeout: 750000,
    transactionConfirmationPollingInterval: undefined,
    blockHeaderTimeout: 10,
    maxListenersWarningThreshold: 100,
    contractDataInputFill: 'data',
    defaultNetworkId: undefined,
    defaultChain: 'mainnet',

```

Figure 9: DAPP Execution 6


```

    defaultChain: 'mainnet',
    defaultHardfork: 'london',
    defaultCommon: undefined,
    defaultTransactionType: '0x2',
    defaultMaxPriorityFeePerGas: '0x9502f900',
    enableExperimentalFeatures: [Object],
    transactionBuilder: undefined,
    transactionTypeParser: undefined
  },
  providers: {
    HttpProvider: [class HttpProvider extends Web3BaseProvider],
    WebsocketProvider: [class WebSocketProvider extends SocketProvider]
  },
  _requestManager: Web3RequestManager {
    _emitter: [EventEmitter],
    _provider: [HttpProvider],
    useRpcCallSpecification: undefined
  },
  _subscriptionManager: Web3SubscriptionManager {
    requestManager: [Web3RequestManager],
    registeredSubscriptions: [Object],
    tolerateUnlinkedSubscription: false,
    _subscriptions: Map(0) {}
  },
  _wallet: Wallet(0) [
    _accountProvider: [Object],
    _addressMap: Map(0) {},
    _defaultKeyName: 'web3js_wallet'
  ],
  _accountProvider: {
    signTransaction: [Function: signTransactionWithContext],
    create: [Function: createWithContext],
    privateKeyToAccount: [Function: privateKeyToAccountWithContext],

```

Figure 10: DAPP Execution 7


```

transactionReceiptPollingInterval: undefined,
transactionSendTimeout: 750000,
transactionConfirmationPollingInterval: undefined,
blockHeaderTimeout: 10,
maxListenersWarningThreshold: 100,
contractDataInputFill: 'data',
defaultNetworkId: undefined,
defaultChain: 'mainnet',
defaultHardfork: 'london',
defaultCommon: undefined,
defaultTransactionType: '0x2',
defaultMaxPriorityFeePerGas: '0x9502f900',
enableExperimentalFeatures: {
  useSubscriptionWhenCheckingBlockTimeout: false,
  useRpcCallSpecification: false
},
transactionBuilder: undefined,
transactionTypeParser: undefined
},
providers: {
  HttpProvider: [class HttpProvider extends Web3BaseProvider],
  WebsocketProvider: [class WebSocketProvider extends SocketProvider]
},
_requestManager: Web3RequestManager {
  _emitter: EventEmitter {
    _events: [Events <Complex prototype>],
    _eventsCount: 2,
    maxListeners: 9007199254740991
  },
  _provider: HttpProvider {
    clientUrl: 'http://127.0.0.1:7545/',
    httpProviderOptions: undefined
  },
},

```

Figure 13: DAPP Execution 10

```

    },
    _addressMap: Map(0) {},
    _defaultKeyName: 'web3js_wallet'
  ],
  syncWithContext: false,
  _functions: {
    'storeDocument475(string)': { signature: '0x4a2aa1d7', method: [Function (anonymous)] },
    'getDocument475()': { signature: '0x89a4ccb7', method: [Function (anonymous)] }
  },
  _overloadedMethodAbis: Map(2) {
    'storeDocument475' => [ [Object] ],
    'getDocument475' => [ [Object] ]
  },
  _methods: {
    storeDocument475: [Function (anonymous)],
    'storeDocument475(string)': [Function (anonymous)],
    '0x4a2aa1d7': [Function (anonymous)],
    getDocument475: [Function (anonymous)],
    'getDocument475()': [Function (anonymous)],
    '0x89a4ccb7': [Function (anonymous)]
  },
  _events: {
    'DocumentRetrieved(address,string,bytes32)': [Function (anonymous)],
    DocumentRetrieved: [Function (anonymous)],
    '0xb288da2131a1e4fa14f5e918eb8e88312d516dfa74cb624cfc395b5f8c9ebc5b': [Function (anonymous)],
    'DocumentStored(address,bytes32)': [Function (anonymous)],
    DocumentStored: [Function (anonymous)],
    '0x0b5964e012751f4988419a5f0dfdd1d3c3721eda68dd04aabc19b94cc1f4d0cf': [Function (anonymous)],
    allEvents: [Function (anonymous)]
  },
  _jsonInterface: [
    {
      anonymous: false,

```

Figure 14: DAPP Execution 11

```
_jsonInterface: [
  {
    anonymous: false,
    inputs: [Array],
    name: 'DocumentRetrieved',
    type: 'event',
    signature: '0xb208da2131a1e4fa14f5e918eb0e88312d516dfa74cb624cfc395b5f8c9ebc5b'
  },
  {
    anonymous: false,
    inputs: [Array],
    name: 'DocumentStored',
    type: 'event',
    signature: '0x0b5964e012751f4988419a5f0dfdd1d3c3721eda68dd04aabc19b94cc1f4d0cf'
  },
  {
    inputs: [Array],
    name: 'storeDocument475',
    outputs: [],
    stateMutability: 'nonpayable',
    type: 'function',
    signature: '0x4a2aa1d7',
    methodNameWithInputs: 'storeDocument475(string)',
    constant: false,
    payable: false
  },
  {
    inputs: [],
    name: 'getDocument475',
    outputs: [Array],
    stateMutability: 'view',
    type: 'function',
    signature: '0x89a4ccb7',
```

Figure 15: DAPP Execution 12

```

    type: 'function',
    signature: '0x89a4ccb7',
    methodNameWithInputs: 'getDocument475()',
    constant: true,
    payable: false
  }
],
_errorsInterface: [],
_address: '0x15089d195F728E0A2eC534104d8c89CF88F66243',
options: {
  address: [Getter/Setter],
  jsonInterface: [Getter/Setter],
  gas: undefined,
  gasPrice: undefined,
  from: '0x3a4FFc685be193eD1D325eba175709ab6aB5095b',
  input: undefined,
  data: undefined
},
context: Web3 {
  _emitter: EventEmitter {
    _events: [Events <Complex prototype>],
    _eventsCount: 1,
    maxListeners: 9007199254740991
  },
  config: {
    handleRevert: false,
    defaultAccount: undefined,
    defaultBlock: 'latest',
    transactionBlockTimeout: 50,
    transactionConfirmationBlocks: 24,
    transactionPollingInterval: 1000,
    transactionPollingTimeout: 750000,
    transactionReceiptPollingInterval: undefined,

```

Figure 16: DAPP Execution 13


```

    Eip1193Provider: [Getter],
    SocketProvider: [Getter],
    isUint8Array: [Getter],
    uint8ArrayConcat: [Getter],
    uint8ArrayEquals: [Getter]
  },
  eth: Web3Eth {
    _emitter: [EventEmitter],
    config: [Object],
    providers: [Object],
    _requestManager: [Web3RequestManager],
    _subscriptionManager: [Web3SubscriptionManager],
    _accountProvider: [Object],
    _wallet: [Wallet],
    ens: [ENS],
    Iban: [Function],
    net: [Net],
    personal: [Personal],
    Contract: [class ContractBuilder extends Contract],
    abi: [Object],
    accounts: [Object]
  }
}
}
recippt: {
  "0": "650",
  "1": "0x00903cf98c709a9afd0d51d5fd348ba936789b2b73452753a58f859a0a579afd",
  "2": true,
  "__length__": 3,
  "content": "650",
  "storedHash": "0x00903cf98c709a9afd0d51d5fd348ba936789b2b73452753a58f859a0a579afd",
  "integrity": true
}

```

Figure 17: DAPP Execution 14

Validation on DAPP Execution Result

From [Figure 11](#) and Figure 17, the values of data hash and stored hash are the same.