CSCI 6313

Introduction to Blockchains

ASSIGNMENT – 1

Part B

Name: Kuldeep Rajeshbhai Gajera

Banner ID: B00962793

Email ID: kuldeepg@dal.ca

Github Link: https://github.com/K-D521/Smart-Contract---Storage

Contents

Task:

create a smart contract to support the storage of documents with assurance of integrity. The smart contract should have the following methods:

- Store document: The document is stored on the blockchain and its hash code is stored also.
- Get document: The document and its hash code are retrieved from the blockchain.

The hash code of the retrieved document is calculated and then compared to the hash code retrieved from the blockchain. If all is OK, both the document and the hash code are returned. Compile and deploy your smart contract on a testbed Ethereum blockchain. Create a Dapp (Distribute Application that uses blockchain smart contracts) that invokes the application.

Answer:

This smart contract, named `**SecureDocumentStorage**`, is to facilitate the secure storage of documents on the blockchain. It ensures the integrity of stored documents by using cryptographic hash functions. The contract includes the following features:

1. **Document Structure**: Defines a `Document` structure containing the document's content and its hash.

2. **saveDocument Function**: This function allows users to store a document by providing its content. It calculates the hash of the content and stores both the content and the hash in a mapping, indexed by the user's address.

3. **retrieveDocument Function**: This function retrieves stored documents, computes and verifies the document's hash to ensure integrity, and returns the document content, stored hash, and a boolean indicating if the integrity is intact.

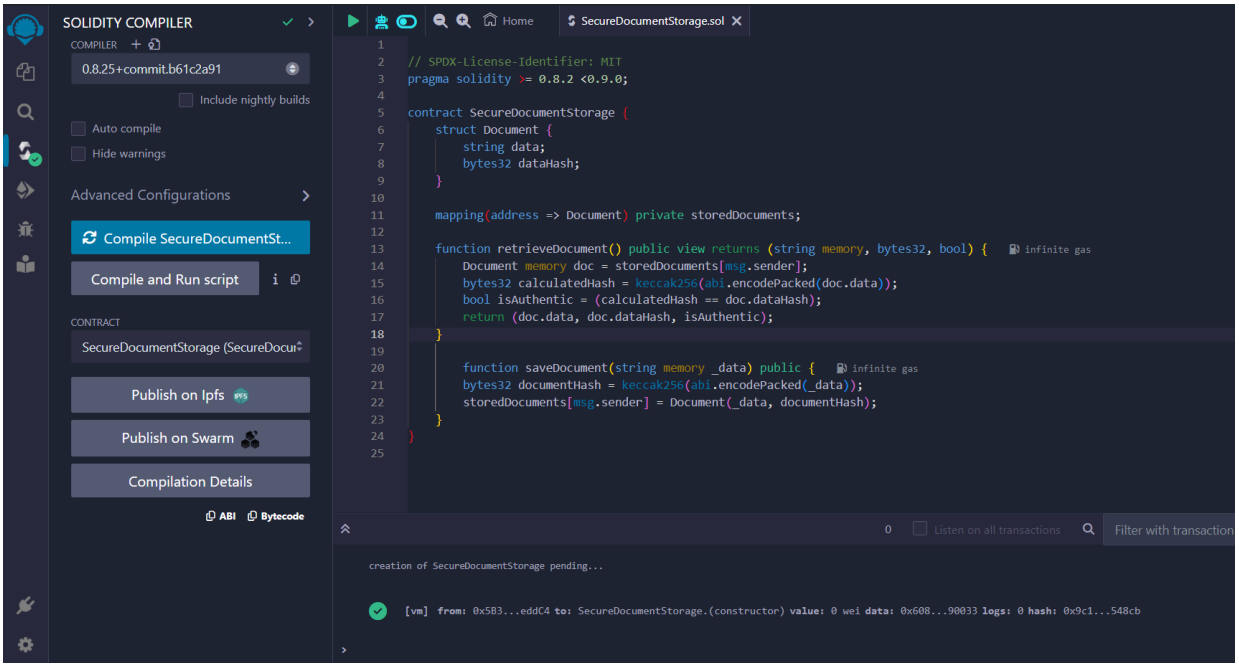

Figure 1: Smart contract created and compiled in Remix.

The `**SecureDocumentStorage**` contract was compiled and deployed using Remix[1] and a local Ethereum testbed via Ganache[2]. The deployment involved selecting an account from Ganache[2] and using Remix[1] to deploy the contract. Once deployed, users can interact with it to securely store and verify documents.

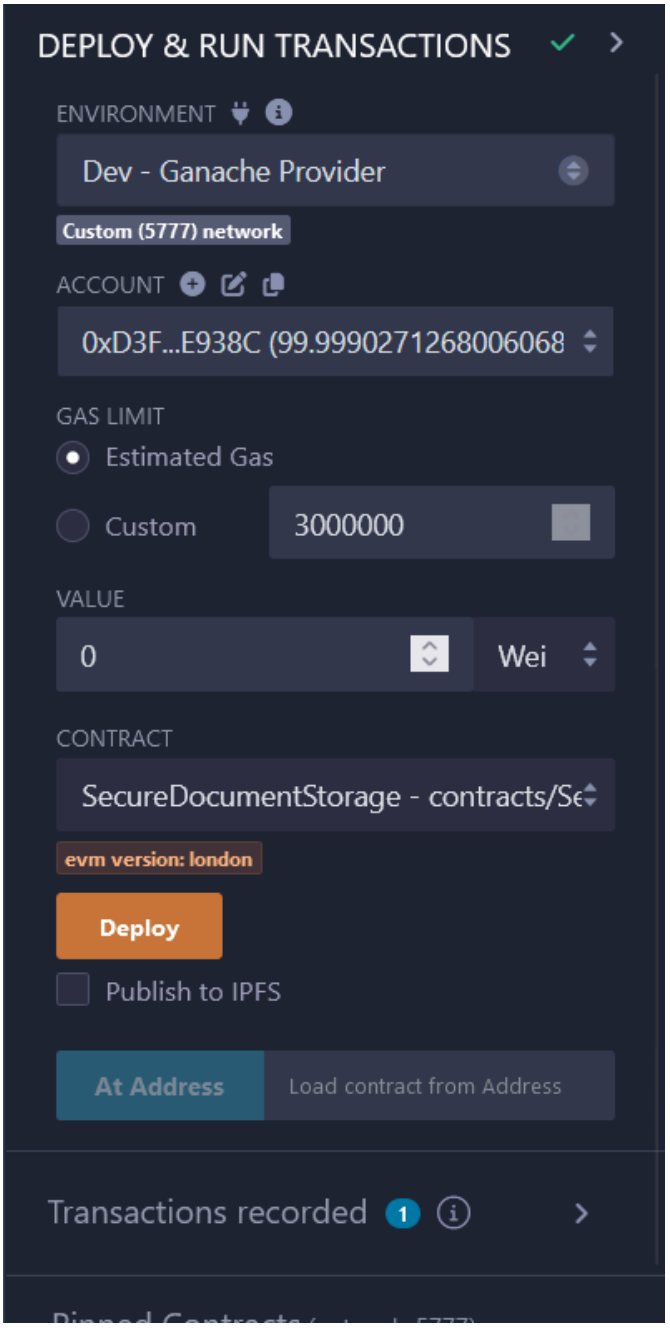

Figure 2: SecureDocumentStorage.sol smart contract deployed locally using Ganache and Remix.

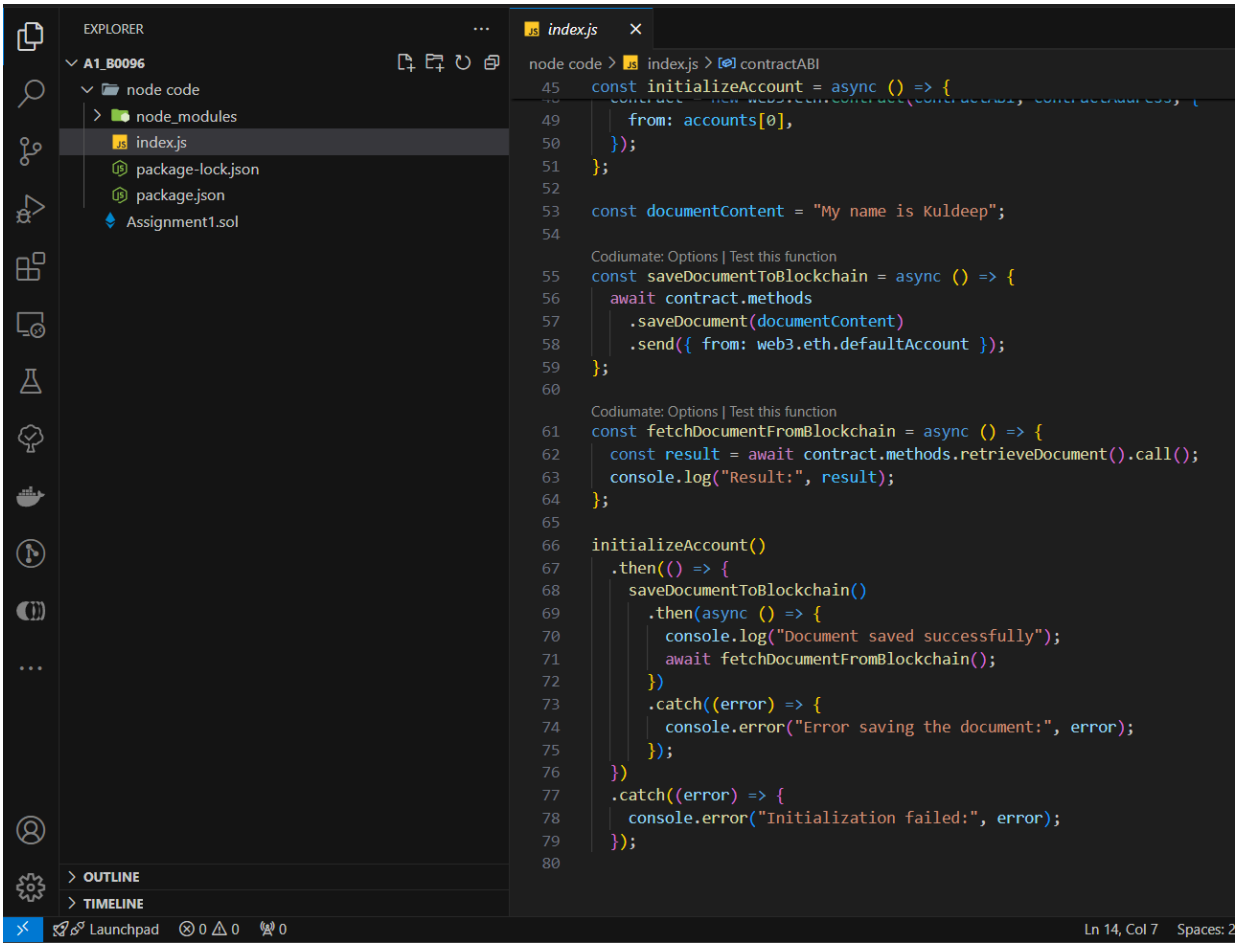

Figure 3: Node.js code

This Node.js code uses the Web3 library to interact with an Ethereum blockchain. The Web3 instance is initialized with an HTTP provider pointing to the local Ganache server.

The code defines two important variables:

contractAddress: The address of the deployed smart contract, obtained from Ganache.
contractABI: The ABI of the smart contract, obtained from the Remix compiler after successful compilation.

Three main functions are defined:

initializeAccount: Sets up the default account for transactions.

saveDocumentToBlockchain: Stores a document on the blockchain by invoking the saveDocument method of the smart contract.

fetchDocumentFromBlockchain: Retrieves a stored document from the blockchain by calling the retrieveDocument method of the smart contract.

The script first calls initializeAccount to set up the account, then sequentially calls saveDocumentToBlockchain to store a document, and finally fetchDocumentFromBlockchain to retrieve and verify the stored document.

Figure 4: Output of Node.js code returning JSON containing the message, contract address, and status.

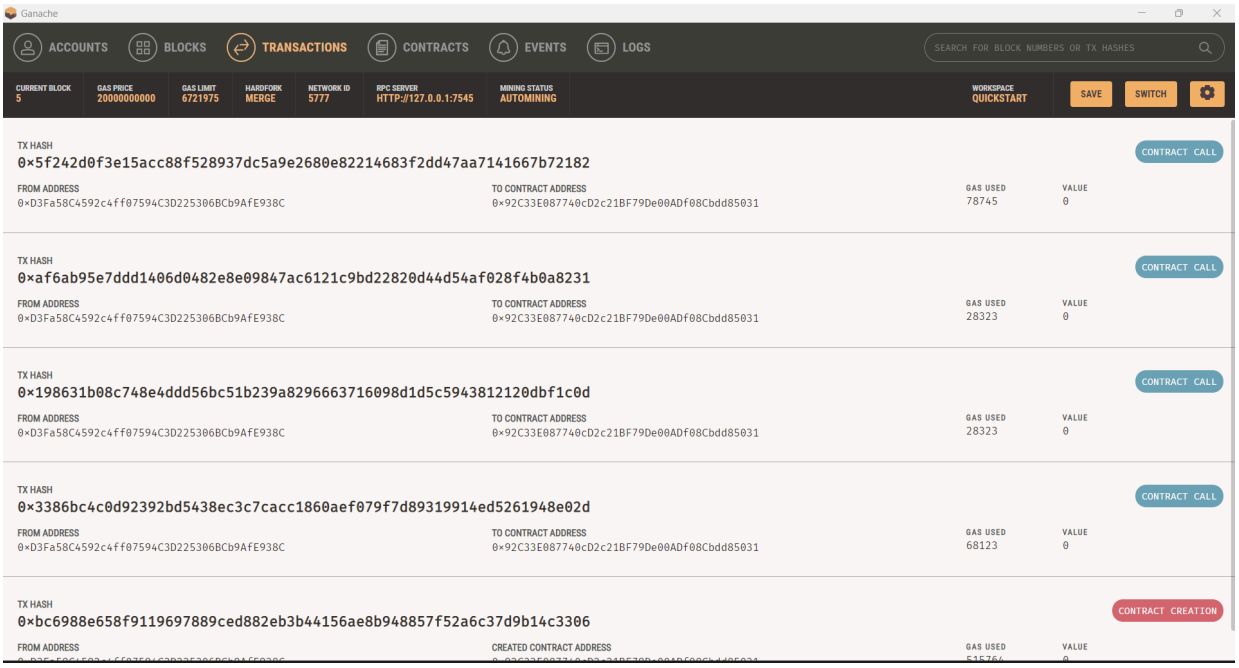

Figure 5: Transactions from Ganache

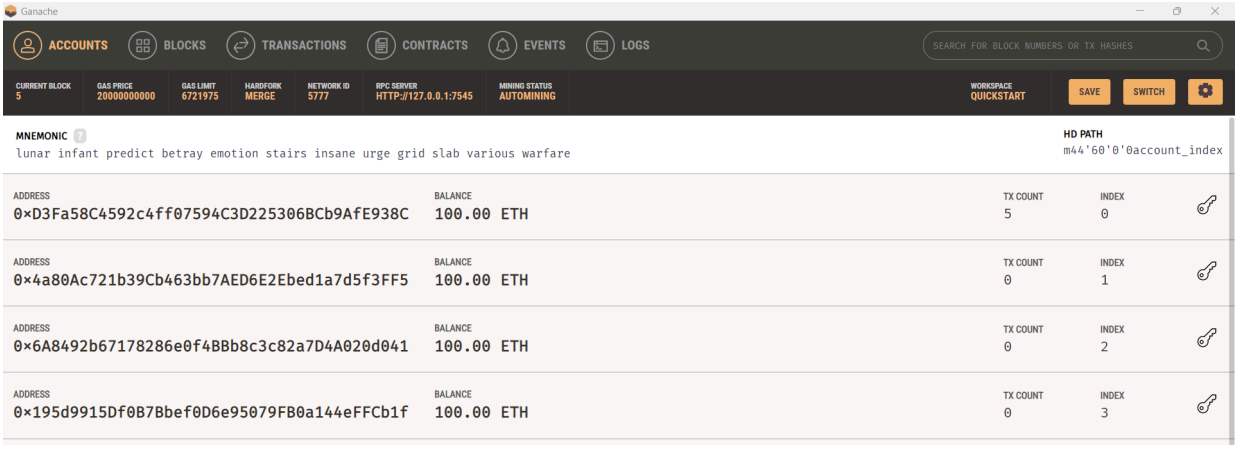

Figure 6: Accounts from Ganache

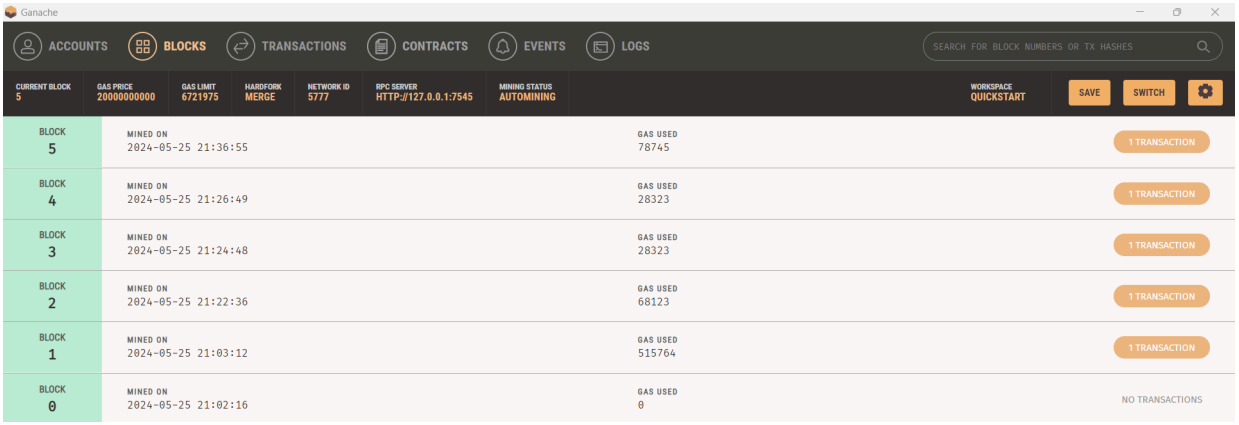

Figure 7: Blocks from Ganache

References

[1] “Remix - Ethereum IDE,” Ethereum.org. [Online]. Available: https://remix.ethereum.org/. [Accessed: 25-May-2024].

[2] “Ganache - Truffle Suite,” Trufflesuite.com. [Online]. Available: https://archive.trufflesuite.com/ganache/. [Accessed: 25-May-2024].

[3] “Home,” Solidity Programming Language. [Online]. Available: https://soliditylang.org/. [Accessed: 25-May-2024].

[4] S. Larson, “Deploying smart contracts with Truffle and Ganache,” Grizzlypeaksoftware.com. [Online]. Available: https://grizzlypeaksoftware.com/articles?id=5vWBWo4Zpi02FSVCmQunxk. [Accessed: 25-May-2024].

[5] “Npm: Web3,” npm. [Online]. Available: https://www.npmjs.com/package/web3. [Accessed: 25-May-2024].

[6] “Run JavaScript everywhere,” Nodejs.org. [Online]. Available: https://nodejs.org/en. [Accessed: 25-May-2024].