UDC: 004.652.4.056.55:004.414.23
Professional paper

DESIGNING USE CASE DIAGRAMS AND SCENARIOS FOR CREATING THE BLOCKCHAIN SYSTEM FOR BIG DATA MANAGEMENT

Hakan REXHEPI[1*], Avni RUSTEMI[1], Sazana ILAZI[1], Florim IDRIZI[1], Grela AJVAZI[1], Hirijete IDRIZI[2]

[1]Department of Informatics, Faculty of Natural Sciences and Mathematics, University of Tetova, N. Macedonia*
[2]Faculty of Agriculture and Biotechnology, University of Tetova, N. Macedonia
[*]Corresponding Author: e-mail: h.rexhepi320057@unite.edu.mk

Abstract

Smart contracts programming plays an important role nowadays in creating blockchain systems. Blockchain technology is one of the technologies that offer hope for overcoming problems related to the preservation of identity, privacy, transparency, and, above all, the immutability of data. The creation of blockchain systems is a challenge that researchers are facing, especially with big data management, due to the high maintenance costs. Currently, however, blockchain technology is finding a lot of use when combined with centralized databases, effectively replicating a blockchain database and network. Through the paper, we will try to clarify the main challenges in blockchain programming, and the reasons for not implementing blockchain programming for big data management. We will present examples of smart contract programming in the Solidity programming language, and we will make analyses and comparisons with centralized object-oriented systems. Part of the work will also be the design of use case diagrams and scenarios for the implementation of blockchain systems in big data management. The main purpose of this paper is to arouse some interest among young developers, to deal with programming in Solidity and the creation of blockchain systems for big data management.

Keywords: smart contract, blockchain technology, Solidity programming language, data immutability.

1. Introduction

The creation of decentralized systems nowadays is a challenge that researchers are facing, due to the decentralized way of managing and storing data, which is challenging, especially when it comes to big data. Blockchain technology, even though it is marking a technological revolution in the field of programming and data management, is again challenging and is facing numerous problems. Among the many challenges, it is worth highlighting the lack of interest of researchers in dealing with the problems and implementation of blockchain in big data, especially for data management and the creation of systems for their management (Rustemi, Dalipi, Atanasovski, & Risteski, 2023a). Blockchain is proving to be successful in creating systems or software modules, where the immutability of data is required, and the same can be verified more easily in the future. This is because blockchain technology is characterized by a distributed ledger, in which the data of the entire blockchain network is stored, as well as all the nodes that are connected to the network, contain a copy of this ledger, which makes it impossible for the data to update or insert new data without being confirmed by all the nodes that are in the blockchain network. Any data that is entered into the blockchain network cannot be deleted or changed, which represents the immutability of the blockchain as a unique characteristic that distinguishes it from centralized systems (G V, B S, L M, B, & H B, 2022). When we talk about the creation of decentralized systems based on blockchain technology, the use of smart contracts is inevitable. Smart contracts are code sequences, which enable the execution of agreements between two parties in digital form, based on predefined conditions. They enable automatic execution of the code, which guarantees the reliability of the system and

the fulfillment of predefined conditions between the two involved parties. Smart contracts above all are useful to facilitate the process of data verification, to facilitate and assist in the fulfillment of obligations between the parties included in the contract, and the automatic execution of code sequences without the intervention of a third party, something which makes them safer and more reliable for use in everyday life during the creation of blockchain systems for data management (Saini, Roy, Chelliah, & Patel, 2021). Compared to traditional contracts, smart contracts are efficient and faster, they are immutable and cost-effective, they work with digital signatures and in virtual spaces, and they do not require third-party authorization. Smart contracts are compatible with the Ethereum platform, as well as the Solidity programming language. Each execution of them on the Ethereum platform has its own cost, which is calculated with Ether (Gugnani, Godfrey, & Sadhya, 2022).
Decentralized systems have begun to be used in various fields, although various challenges are encountered during practical implementations, due to the characteristics of the blockchain. However, very important during the creation of decentralized systems, is the definition of the main actors who will use the system. Then the definition of auxiliary equipment for the realization of the system and an analysis and long-term strategic plan. The definition of use case diagrams and scenarios is also very important in the realization of the system, up to the definition of the functional and non-functional requirements of the system. Also, the definition of the conceptual model, the architecture of the system operation, and the detailed analysis of the practical implementation are necessary for the successful implementation of blockchain systems (Rustemi, Dalipi, Atanasovski, & Risteski, 2023b).
Through this paper, we will first describe the blockchain system, its main characteristics, to the advantages and disadvantages compared to traditional systems. Then we will design use case diagrams for the main actors in a big data management system, describing also use case scenarios for the same. We will briefly describe all the necessary equipment for the implementation of the blockchain system and, using the Solidity programming language, we will give examples of smart contract implementation, graphically describing their execution in the blockchain network.

2. Related works

More and more attempts are being made to digitize the services in various public and private institutions, and with this to facilitate the procedures for generating and verifying documents, as well as to speed them up and to be able to provide transparent services at any time. Although in the centralized systems, there are more and more attempts to introduce different features of artificial intelligence, blockchain technology, and different encryption algorithms, with the sole purpose of increasing the efficiency, security, and reliability of the system, despite many issues that have to do with identity, privacy and data security remain as issues that are discussed and are not being implemented through centralized systems.
Unlike centralized systems, blockchain systems for big data management are characterized by the classification of services into private, public, and consortium. Services are transparent, data is immutable. Different cryptographic mechanisms are used for data encryption and decryption, which makes data decryption impossible. Despite the advantages, blockchain systems are characterized by more limitations and shortcomings during practical implementations. Blockchain systems are the target of various attacks, as a result of the transparency of the various services they offer. Lack of standardization of services, as a result of the lack of standardization of smart contracts. Lack of blockchain databases, and maintaining services for a longer time is very expensive. Most of the data is stored in file storage, or in the blockchain network itself, or they also use centralized databases with blockchain features. The process of transferring data from centralized systems to blockchain ones is difficult and faces numerous

problems. And above all, there is a lack of staff, programmers, and researchers who deal with blockchain programming and big data management, because blockchain technology is used more for economic benefits, and in particular with cryptocurrencies (Rustemi, Atanasovski, & Risteski, 2023).

Unlike centralized systems, where practical implementation requires a programming language, database, and system design, blockchain systems need more additional tools for implementation. The implementation of the blockchain network is made possible through the Ethereum Virtual Machine, which represents the simulation of the blockchain network, offering a real environment for the implementation of virtual activities that must be part of the blockchain network. First is the frontend part, which represents the part with which users will have direct contact. Here are included different user interfaces with which the blockchain system is designed. For design, the same tools as the centralized systems can be used, including HTML, CSS, Node.js, and the newest versions. After completing the specified user interface, they initiate the execution of the smart contract, which represents the backend part of the system. The coding of the smart contract on the Ethereum platform is done using the Solidity programming language. There are a large number of applications that offer free testing and execution, to test and optimize their code. It should be taken into account that each real execution in the blockchain network has its own cost. Truffle Suite is a framework that simulates the blockchain network, which consists of several parts, but among the most important is Ganache. All accounts, blocks, transactions, contracts, and activities performed by a user are presented in Ganache. For each realization of the transaction, the receiver's address, the price of the execution of the transaction in Gas, Network ID, the status of the mining process, etc. are given through Ganache (Truffle Suite, n.d.). To carry out transactions on the Ethereum platform, you must confirm and connect the blockchain wallet to the blockchain network. One of the most used wallets is also Metamask, which is connected to our Ethereum address, every transaction is confirmed by the same and a certain amount of Ether is paid (MetaMask, n.d.). Very important is the use of RemixIDE, which is an integrated development environment, for testing, optimizing, and correcting the smart contract code, as well as for calculating in advance the cost of executing the smart contract in the blockchain network (Remix, n.d.). And in the end, the definition of blockchain storage is very important, for storing data for a longer period. Among the blockchain databases used for practical implementations are: Interplanetary file storage (IPFS), which is a file storage based on the peer-to-peer protocol, BigChainDB, Swarm, Cassandra, ChainifyDB, CovenantSQL, Modex Blockchain Database (BCDB), Postchain (Rustemi, Atanasovski, & Risteski, 2023). Figure 1 describes the main components of the blockchain system, starting from the frontend, backend, ethereum network, and up to the blockchain storage where the data is stored for a longer period.

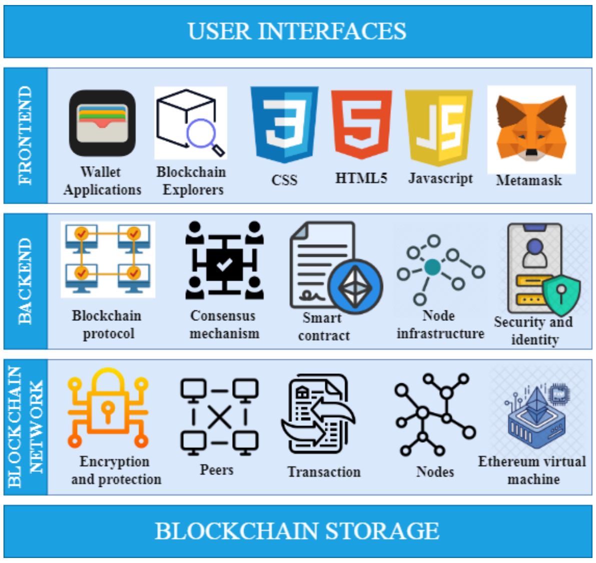
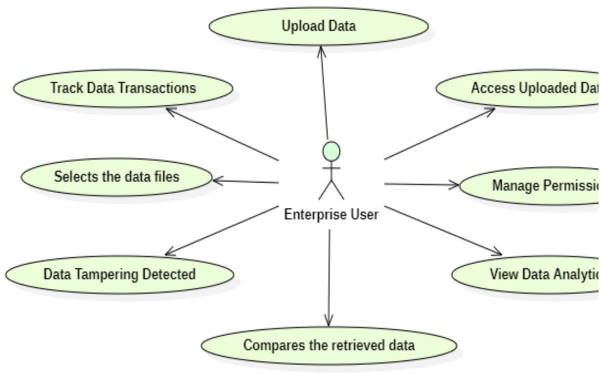


Figure 1. Overview of the main components of the blockchain system

3. Use case diagrams and scenarios

The most efficient method for system design, with which the functionalities of the system, in general, are shown until its final generalization, which is otherwise known as Model-Driven Architecture (MDA), is the Unified Modeling Language (UML), which consists of several diagrams that specify the structure and behavior of the system in general. UML diagrams can also be used for the design of smart contracts, their functionality, and their connection during execution, with the aim of more successful validation and verification of the system (Jurgelaitis, čeponienė, & Butkienė, 2022). As the most important actors in the use of the blockchain system for big data management, there are many, however the most important are data providers, users of the system through different companies, data consumers, and developers in the creation and maintenance of the system structure, regulators and auditors as well as the regular users of the system that they use for different purposes, depending on the nature of the company that offers such a system.



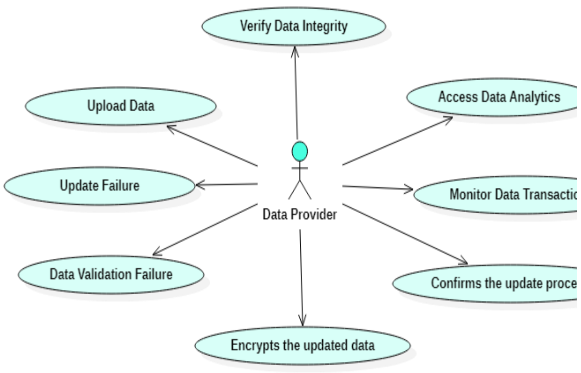
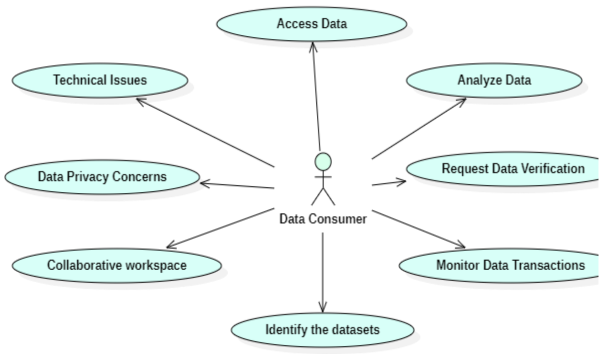


Figure 2.a. Use case diagram for enterprise user

Figure 2.b. Use case diagram for data provider



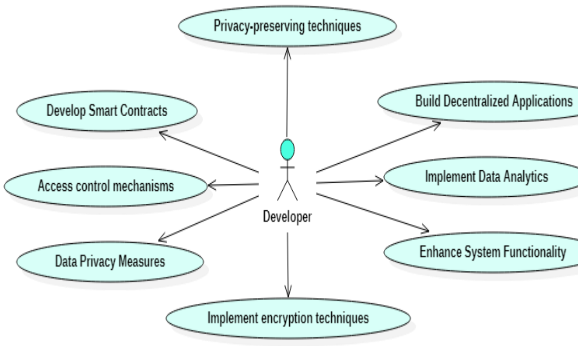

Privacy-preserving techniques
Develop Smart Contracts
Build Decentralized Applications
Access control mechanisms
Implement Data Analytics
Developer
Data Privacy Measures
Enhance System Functionality
Implement encryption techniques

Figure 2.c. Use case diagram for data consumer

Figure 2.d. Use case diagram for developer

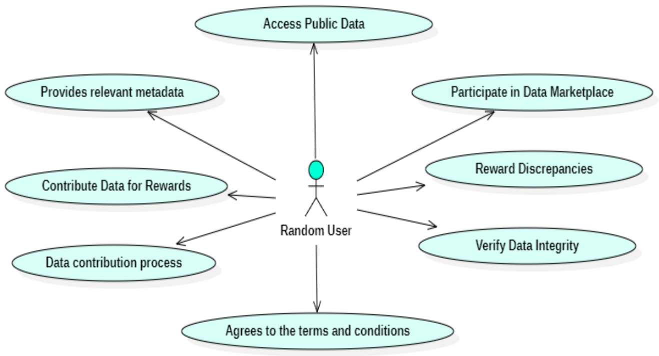

Access Public Data
Provides relevant metadata
Participate in Data Marketplace
Reward Discrepancies
Contribute Data for Rewards
Random User
Verify Data Integrity
Data contribution process
Agrees to the terms and conditions

Figure 2.e. Use case diagram for random user

Large companies, corporations, and various businesses can use blockchain systems for big data management, to create the supply chain, logistics, various customer transactions, etc. Through Figure 2.a. we have designed the use case diagram for the enterprise user. Among the most regular activities of this actor are: uploading data for longer storage or validation, the same can be accessed later, management permissions for the data uploaded to the system, where the permissions of other users are determined, tracking the execution of transactions that are carried out both from the side of customers, but also from other users who have them under their control.

Data providers are entities that influence the generation, storage, and processing of big data in the blockchain, whether for their storage, real-time access, or validation. It can be IoT devices, sensors, and any entity that produces large amounts of data, and that directly affects the real processing of big data. Among other things, data providers can verify data integrity, access data analytics, monitor data transactions, upload data to the blockchain, and encrypt the updated data. Figure 2.b. presents the use case diagram for the data provider, presenting the most important use case activities.

Data consumers are all users, corporations, or businesses that require access to data stored in the blockchain system, to carry out their analysis, research, decision-making, purchase, sale, and other purposes. , depending on the nature of the blockchain system. Among other things, consumer data can analyze data, request data verification, monitor data transactions, and take care of the system's technical issues. Figure 2.c. presents the use case diagram for the data consumer, presenting the most important use case activities.

Developers are the key actors in the creation and maintenance of the blockchain system, and in particular in the programming of smart contacts and their successful execution in the blockchain network. The developers are the practical implementers of the system, and they take care that the system is functional, and enables the provision of functional and non-functional services of

the system. Developers, among other things, must also take care of data analytics, and implement encryption techniques, privacy-preserving techniques, and control mechanisms. Figure 2.d. presents the use case diagram for the developer, presenting the most important use case activities.
Random users are the real users of the blockchain system. They can be individuals or entities that have direct interaction with the system for managing their daily work. Depending on the nature and space where the blockchain system is implemented, regular users can be researchers, investors, journalists, various public or private institutions, and many other entities. Figure 2.e. describes the use case diagram for random users, describing some of their main activities in a blockchain system.

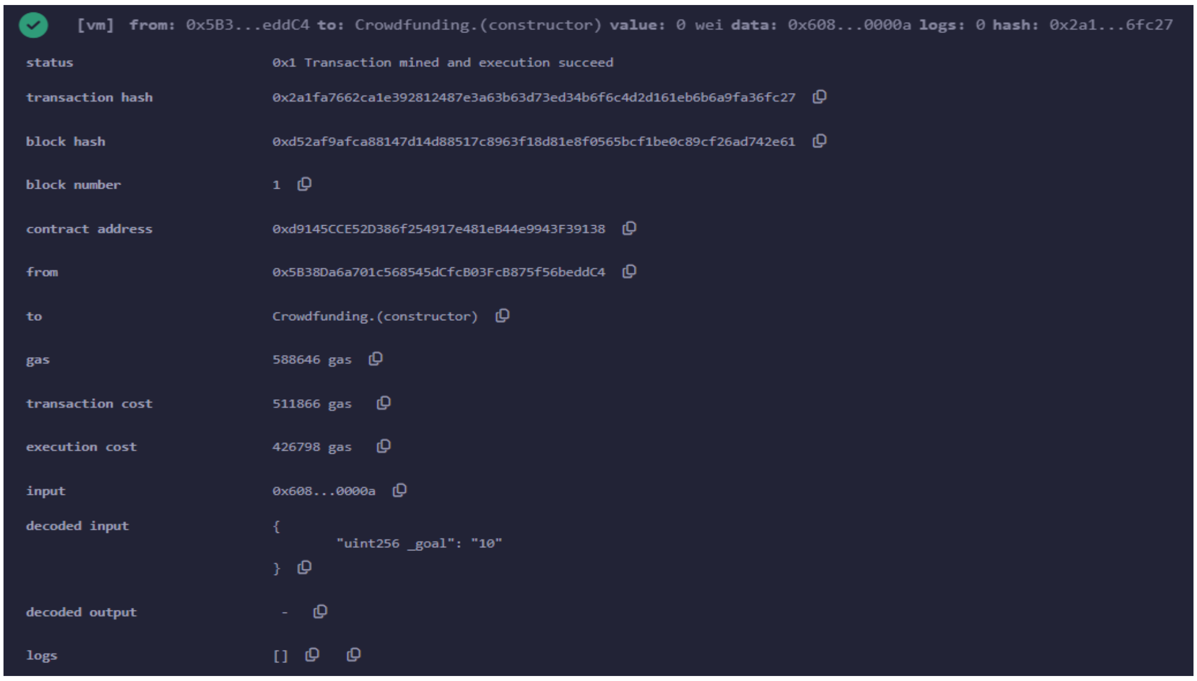
4. Smart contract deploying and testing

Smart contracts are self-executing contracts with the terms of the agreement between buyer and seller directly written into code. These contracts automatically enforce and execute the terms of an agreement when predetermined conditions are met. Smart contracts run on blockchain platforms like Ethereum, which ensures transparency, immutability, and decentralization.
Solidity is the programming language used for writing smart contracts on the Ethereum blockchain. It's specifically designed for creating smart contracts and is influenced by C++, JavaScript, and Python. Solidity allows developers to define the rules and logic of their smart contracts, specifying how they should behave under certain conditions. It's an essential tool for building decentralized applications (DApps) and implementing automated trustless transactions on the blockchain.
Here is an example of a smart contract (the code below is written, optimized, and tested using RemixIDE):
Crowdfunding Contract: This contract allows users to contribute funds towards a project and release the funds to the project owner only when a funding goal is met.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Crowdfunding {
    address public projectOwner;
    uint public fundingGoal;
    uint public totalFunds;
    mapping(address => uint) public contributions;

    event FundsContributed(address contributor, uint amount);
    event FundingGoalReached(uint totalFunds);
    event FundsReleased(address projectOwner, uint amount);

    constructor(uint _goal) {
        projectOwner = msg.sender;
        fundingGoal = _goal;
    }

    function contribute() public payable {
        require(msg.value > 0, "Contribution amount must be greater than 0");
        contributions[msg.sender] += msg.value;
        totalFunds += msg.value;
        emit FundsContributed(msg.sender, msg.value);

        if (totalFunds >= fundingGoal) {
            emit FundingGoalReached(totalFunds);
        }
    }

    function releaseFunds() public {
        require(msg.sender == projectOwner, "Only the project owner can release funds");
```

```
            require(totalFunds >= fundingGoal, "Funding goal not reached yet");

            uint amountToRelease = totalFunds;
            totalFunds = 0;
            payable(projectOwner).transfer(amountToRelease);
            emit FundsReleased(projectOwner, amountToRelease);
        }}
```

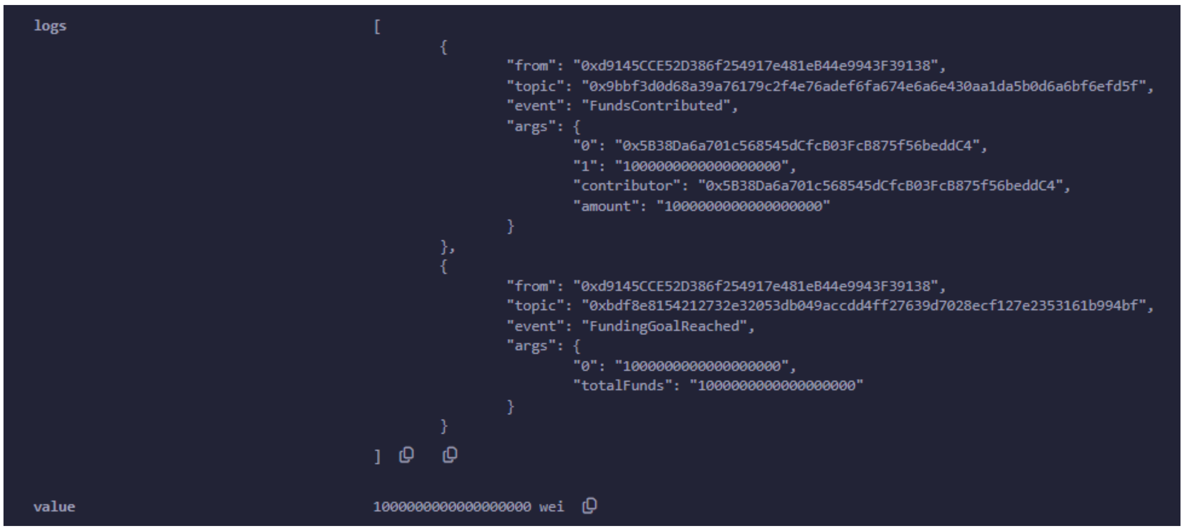
We have presented the result of the deployment process in RemixIDE in Figure 3, where, among other things, all the details of the deployment process of the smart contract are given, such as the hash code of the transaction, the address of the sender and recipient of the transaction, gas expenses, etc.



```
[vm] from: 0x5B3...eddC4 to: Crowdfunding.(constructor) value: 0 wei data: 0x608...0000a logs: 0 hash: 0x2a1...6fc27
status                      0x1 Transaction mined and execution succeed
transaction hash            0x2a1fa7662ca1e392812487e3a63b63d73ed34b6f6c4d2d161eb6b6a9fa36fc27
block hash                  0xd52af9afca88147d14d88517c8963f18d81e8f0565bcf1be0c89cf26ad742e61
block number                1
contract address            0xd9145CCE52D386f254917e481eB44e9943F39138
from                        0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                          Crowdfunding.(constructor)
gas                         588646 gas
transaction cost            511866 gas
execution cost              426798 gas
input                       0x608...0000a
decoded input               {
                                    "uint256 _goal": "10"
                            }
decoded output               -
logs                        []
```

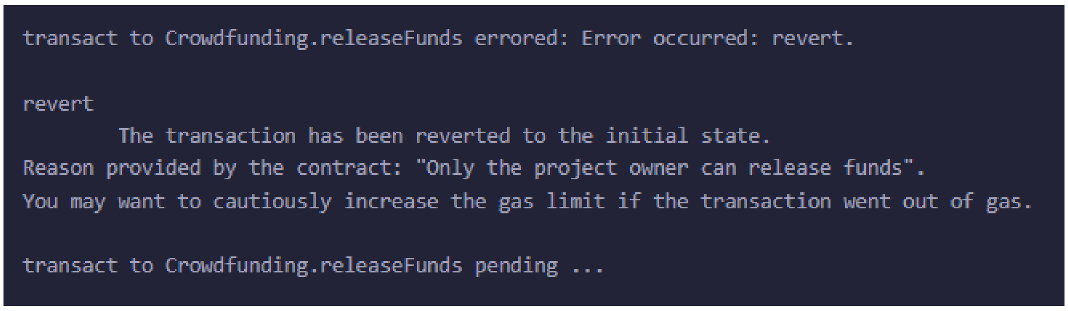
Figure 3. Crowdfunding.sol successfully compiled and deployed

We've conducted a series of tests on our Crowdfunding smart contract to ensure its functionality and security. First, we performed an ether contribution test, successfully transferring 1 ether to the smart contract from a specified address. Next, we attempted to release the funds from an address that was not the project owner, which correctly failed as expected. We then retried releasing the funds using the project owner's address, and the test passed, confirming that only the project owner can release the funds. Finally, we verified the funding goal of the smart contract using the project owner's address, ensuring that the goal was correctly set and maintained throughout the tests. These tests validate the key functionalities and access controls of the smart contract. Figures 4a, 4b, and 4c show the result of testing and validating the smart contract.



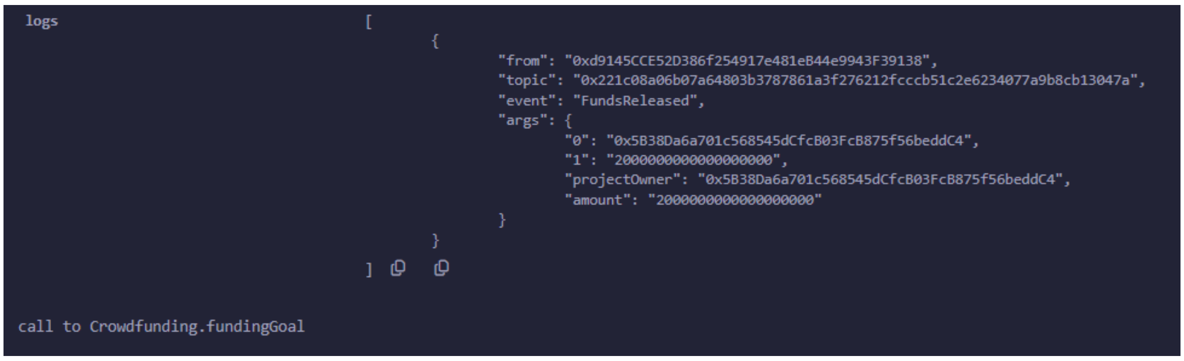
```
logs                    [
                                {
                                        "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
                                        "topic": "0x9bbf3d0d68a39a76179c2f4e76adef6fa674e6a6e430aa1da5b0d6a6bf6efd5f",
                                        "event": "FundsContributed",
                                        "args": {
                                                "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                "1": "1000000000000000000",
                                                "contributor": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                "amount": "1000000000000000000"
                                        }
                                },
                                {
                                        "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
                                        "topic": "0xbdf8e8154212732e32053db049accdd4ff27639d7028ecf127e2353161b994bf",
                                        "event": "FundingGoalReached",
                                        "args": {
                                                "0": "1000000000000000000",
                                                "totalFunds": "1000000000000000000"
                                        }
                                }
                        ]

value                   1000000000000000000 wei
```

Figure 4.a. Ether contribution test passed



```
transact to Crowdfunding.releaseFunds errored: Error occurred: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Only the project owner can release funds".
You may want to cautiously increase the gas limit if the transaction went out of gas.

transact to Crowdfunding.releaseFunds pending ...
```

Figure 4.b.releaseFunds failed because only the project owner can call this function



```
logs                    [
                                {
                                        "from": "0xd9145CCE52D386f254917e481eB44e9943F39138",
                                        "topic": "0x221c08a06b07a64803b3787861a3f276212fcccb51c2e6234077a9b8cb13047a",
                                        "event": "FundsReleased",
                                        "args": {
                                                "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                "1": "2000000000000000000",
                                                "projectOwner": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                "amount": "2000000000000000000"
                                        }
                                }
                        ]

call to Crowdfunding.fundingGoal
```

Figure 4.c. Funds released by the project owner

5. Conclusion

To sum up, our study has investigated how blockchain technology and smart contracts can revolutionize the large data management industry. There has been a paradigm shift in the way large-scale data may be stored, shared, and safeguarded thanks to the programmable logic of smart contracts and the decentralization and immutability inherent in blockchain networks.
The capacity of blockchain technology to offer an unchangeable and transparent ledger for data exchanges is one of its main advantages. Blockchain networks guarantee data integrity and participant confidence by utilizing cryptographic algorithms and consensus processes. This reduces the possibility of data alteration or inappropriate access. This is especially important when it comes to big data, since the accuracy and dependability of the data are critical.
Smart contracts enable self-executing agreements depending on predetermined criteria, providing a novel method of automating data management procedures. This results in cost

savings and enhanced efficiency by streamlining workflows and lowering the need for middlemen.
As blockchain networks are decentralized, they support inclusion and data sovereignty, giving people and organizations the ability to keep ownership their data assets. Blockchain technology promotes a more democratic and fair data environment where data ownership and privacy rights are respected by doing away with central authority and intermediaries.
Nevertheless, blockchain technology has drawbacks in addition to its many advantages. To fully realize the potential of blockchain-based big data solutions, scalability, interoperability, and regulatory issues are still major obstacles that need to be overcome.

References

[1] Gugnani, P., Godfrey, W. W., & Sadhya, D. (2022). Ethereum Based Smart Contract for Event Management System. In *2022 IEEE 6th Conference on Information and Communication Technology (CICT)* (pp. 1-5). Gwalior, India. doi: 10.1109/CICT56698.2022.9997939

[2] G V, S., B S, S., M, G. L., B, V., & B, P. H. (2022). Profile - Decentralized Application for Education. In *2022 International Conference on Emerging Techniques in Computational Intelligence (ICETCI)* (pp. 76-83). Hyderabad, India. doi: 10.1109/ICETCI55171.2022.9921360

[3] Jurgelaitis, M., čeponienė, L., & Butkienė, R. (2022). Solidity Code Generation From UML State Machines in Model-Driven Smart Contract Development. *IEEE Access, 10,* 33465-33481. doi: 10.1109/ACCESS.2022.3162227

[4] Rustemi, A., Atanasovski, V., & Risteski, A. (2023). DATABASE DESIGN IN A BLOCKCHAIN-BASED SYSTEM FOR GENERATING AND VERIFYING ACADEMIC CREDENTIALS. *Journal of Electrical Engineering and Information Technologies, 8,* 93-100. doi: 10.51466/JEEIT2382210093r

[5] Rustemi, A., Dalipi, F., Atanasovski, V., & Risteski, A. (2023). A Systematic Literature Review on Blockchain-Based Systems for Academic Certificate Verification. *IEEE Access, 11,* 64679-64696. doi: 10.1109/ACCESS.2023.3289598

[6] Rustemi, A., Dalipi, F., Atanasovski, V., & Risteski, A. (2023). Towards a Conceptual Model of a Blockchain System for Automatic Generation of Academic Diplomas: Use Cases and Scenarios. In *2023 4th International Conference on Communications, Information, Electronic and Energy Systems (CIEES)* (pp. 1-6). Plovdiv, Bulgaria. doi: 10.1109/CIEES58940.2023.10378773

[7] Saini, K., Roy, A., Chelliah, P. R., & Patel, T. (2021). Blockchain 2.0: A Smart Contract. In *2021 International Conference on Computational Performance Evaluation (ComPE)* (pp. 524-528). Shillong, India. doi: 10.1109/ComPE53109.2021.9752021

[8] Truffle Suite. (n.d.). Ganache. Retrieved April 4, 2024, from https://archive.trufflesuite.com/docs/ganache/

[9] MetaMask. (n.d.). A crypto wallet & gateway to blockchain apps. Retrieved April 4, 2024, from https://metamask.io/

[10] Remix. (n.d.). The Native IDE for Web3 Development. Retrieved April 4, 2024, from https://remix.ethereum.org