APPLICATION OF BLOCK CHAIN TECHNOLOGY IN DISASTER MANAGEMENT

A PROJECT REPORT

OF PROJECT-II (PROJ-IT781)

BACHELOR OF TECHNOLOGY
in
Information Technology

(From **Maulana Abul Kalam Azad University of Technology, West Bengal**)

SUBMITTED BY

Amit Shah (13000220004)
Debraj Bhattacharjee (13000220014)
Mainak Paul (13000220023)
Souvik Chattopadhyay (13000220025)

Under the Supervision of
Dr. Tanmay Bhattacharya

Department of Information Technology
Techno Main Salt Lake
Kolkata -700091

Abstract

In the face of escalating challenges posed by natural disasters and humanitarian crises, the necessity for efficient and timely relief measures has become strikingly apparent. Traditional disaster management systems grapple with critical issues such as real-time data sharing, resource allocation, and transparency. This research propounds a paradigm shift towards decentralized disaster relief, harnessing the latent potential of blockchain technology. The project's core objective is to establish a secure and transparent platform by leveraging blockchain's intrinsic attributes, namely decentralization, immutability, and smart contracts. In response to the pressing need for improved disaster management systems, our project seeks to leverage blockchain technology for decentralized and transparent disaster relief efforts. This progress report outlines the advancements made towards the development and deployment of our blockchain-based solution using Hyperledger Fabric. The abstract amalgamates the comprehensive goals and methodology of the project, emphasizing the urgent need for improved coordination, communication, and resource allocation in disaster response operations. The endeavor aspires to elevate data integrity, enhance decision-making processes, and instill accountability. Situated at the nexus of technology and humanitarianism, our project on the "Application of Blockchain Technology in Disaster Management" unfolds a pioneering narrative. It delves into the design, implementation, and impact of a decentralized blockchain system, with a core objective of enhancing real-time data sharing in disaster relief through the transformative power of blockchain technology. The project incorporates innovative approaches in missing persons' tracking, aid distribution, critical area identification, refugee relief camp mapping, disaster reporting, record-keeping, security, and continuous system improvement, promising to reshape humanitarian efforts and disaster response mechanisms fundamentally.

Keywords

Blockchain Technology · Disaster Management · Decentralization · Real-time Data Sharing · Smart Contracts · Humanitarian Technology · Hyperledger Fabric

Contents

List of Figures

Chapter – I

INTRODUCTION

1 INTRODUCTION

1. Introduction of the project:

The integration of blockchain technology into disaster management signifies a profound shift in crisis response strategies. Traditional systems often struggle with inefficiencies, lack of transparency, and delayed responses. Blockchain, renowned for its decentralized and secure nature, offers a novel solution to these challenges.

2. Objective of Industrial Project:

In response to escalating natural disasters and humanitarian crises, our project seeks to leverage blockchain technology for decentralized and transparent disaster relief efforts. The immutability of blockchain ensures that critical information is securely recorded, fostering trust and accountability among stakeholders.This feature is particularly crucial in disaster scenarios where swift and accurate information dissemination can be a matter of life and death.

3. Summary of a Report:

As part of our industrial project, we are now running algorithms using Hyperledger Fabric.The decentralized nature of blockchain technology reduces the risk of data loss or manipulation by eliminating the reliance on a central repository. By harnessing Hyperledger Fabric, we aim to revolutionize how we prepare for, respond to, and recover from crises.This progress report outlines our advancements in running a sample test network using Hyperledger Fabric, laying the groundwork for decentralized, secure, and transparent disaster relief efforts.

Chapter – II

LITERATURE SURVEY

2 Literature Survey

In recent years, the transformative potential of blockchain technology in disaster response has garnered attention, as evidenced by the work of Tapscott and Tapscott (2016) [6]. This literature review explores the decentralized nature of blockchain and its potential to enhance security and efficiency in disaster relief operations. The authors emphasize the paradigm shift that blockchain introduces in disaster management, highlighting the system's ability to operate without a central authority. This approach aligns with the overarching goal of improving the resilience and effectiveness of disaster response through innovative technological solutions.

M. Swan (2017) [11] critically examine the challenges and opportu- nities associated with blockchain in disaster management. Their work lays the foundation for understanding potential hurdles, offering valuable insights for re- searchers and practitioners. Strategies for effectively harnessing blockchain for humanitarian purposes are outlined, highlighting the importance of a thought- ful and strategic approach to maximize the benefits of this technology in the context of disaster response.

Casey (2018) [12] delves into the realm of smart contracts and their piv- otal role in automating and streamlining coordination efforts during disaster response. The focus on transparent and self-executing agreements is crucial for the efficient distribution of aid packages and the management of critical ar- eas. Casey's work underscores the practical application of blockchain through smart contracts, offering a promising avenue for improving the coordination and deployment of resources in disaster-stricken areas.

X Zhang and co. (2018) [15] contribute to the literature by examining blockchain's role in community engagement during disaster response. Their work navigates ethical considerations and adopts a community-centric approach. This perspective aligns with the broader goal of creating inclusive and ethically sound disaster response frameworks.

The concept of real-time data sharing in disaster scenarios takes center stage in the work of Mazi and Kohli (2018) [14]. Their exploration emphasizes the immediacy and transparency required in disaster management. The alignment with the core objectives of disaster response projects underscores the significance of timely information exchange. Blockchain's potential to facilitate real-time data sharing emerges as a key consideration in addressing the dynamic nature of dis- asters, enabling quicker decision-making and more effective response strategies. The synergies between blockchain and the Internet of Things (IoT) in disaster scenarios are explored by Ichiwaka and co. (2018) [16]. Their work provides insights into comprehensive solutions for real-time monitoring and response, informing considerations for future technological integration. By examining the intersection of blockchain and IoT, the authors contribute to a more holistic understanding of how emerging technologies can collaborate to improve the effectiveness of disaster management strategies.

Chapter – III

DEFINITION OF THE PROBLEM WITH THE MODULES AND FUNCTIONALITIES

3 DEFINITION OF THE PROBLEM WITH THE MODULES AND FUNCTIONALITIES

3.1. Problem Statement:

- **Context:** Escalating natural disasters and humanitarian crises demand a paradigm shift in disaster relief strategies.
- **Challenge:** Conventional systems struggle with real-time data sharing, resource allocation, and transparency.
- **Objective:** Advocate for a decentralized disaster relief paradigm using blockchain technology. [1]

3.2. Modules and Functionalities:

Step 1: Missing People Log

- **Input:** Name, last known location, description.
- **Process:** Smart contracts in Solidity securely log andupdate missing people's information.
- **Output:** Secure, transparent, and up-to-date list of missing people [1].

Step 2: Disaster Reporting and Management

- **Input:** Disaster details (name, location, timestamp, description, critical areas).
- **Process:** Smart contracts manage disasters, ensuring transparency, traceability, and security.
- **Output:** Efficient disaster response and management [3].

Step 3: Aid Package Distribution

- **Input:** Recipient information, aid package details.
- **Process:** Smart contracts coordinate aid packagedistribution, tracking movement.
- **Output:** Efficient aid distribution to affected areas. [5]

Step 4: Critical Area Management

- **Input:** Area name, description, severity.
- **Process:** Smart contracts add/remove critical areas,prioritize response efforts.
- **Output:** List of active critical areas for effective response [7].

Step 5: Refugee Relief Camp Mapping

- **Input:** Refugee and camp information.
- **Process:** Smart contracts assign refugees to camps,manage allocations.
- **Output:** Details of relief camps and active refugees [9].

Step 6: Medical Record Management

- **Input:** Medical Records (Name, Address).
- **Process:** Blockchain secures and stores medical records with immutability.
- **Output:** Comprehensive and secure medical records for injured individuals [11].

Step 7: Continuous Monitoring and Improvement

- **Input:** Real-time system data, user feedback, emerging technologies.
- **Process:** Continuous monitoring for irregularities, real-time data up- dates, user feedback analysis.
- **Output:** Well monitored, up-to-date, and ever improving blockchain based humanitarian system [13].

3.3. Future Enhancement:

- **Plan:** Run and deploy smart contracts in Hyperledger Fabric.
- **Objective:** Enhance the system further with emerging technologies[15]

Chapter – IV

SOFTWARE DESIGN

WORK FLOW DIAGRAM

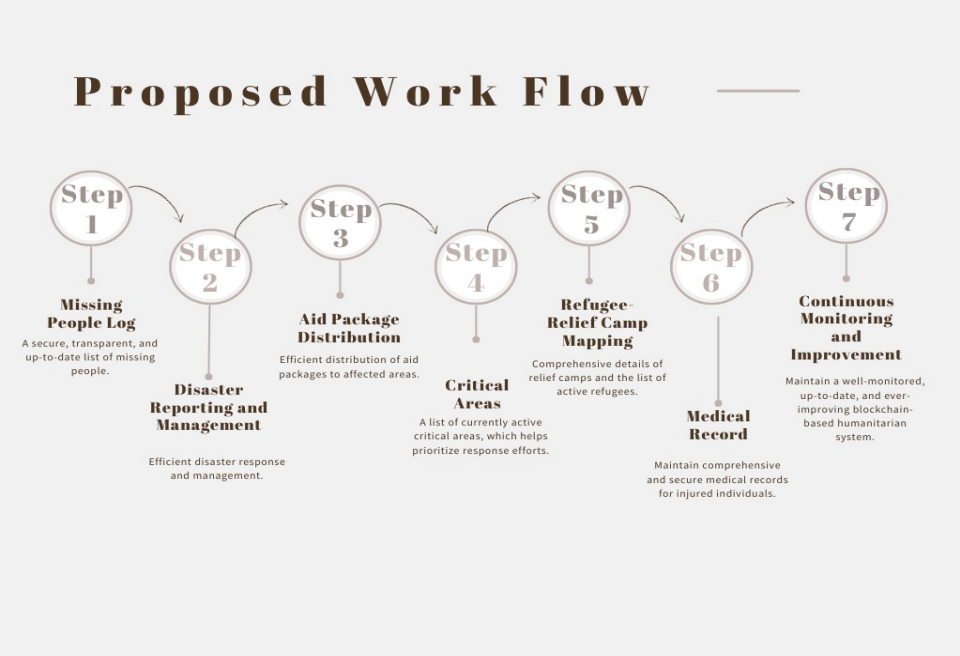


Fig 1: Work Flow Diagram

The blockchain-based disaster management system is intricately designed with seven modules, each implemented as a Solidity smart contract. The "Missing People Log" module ensures transparency by structuring details of missing persons, allowing their addition through emitted events. The "Disaster Reporting and Management" module tracks and resolves disaster reports, capturing crucial details. "Aid Package Distribution" manages aid package distribution, recording sender, recipient, and delivery status. The "Critical Area Management" module handles critical zones based on severity. "Refugee Relief Camp Mapping" efficiently maps refugees to camps, ensuring transparent organization. The "Medical Record Management" module securely stores medical data. The "Continuous Monitoring and Improvement" module, restricted to an owner, identifies improvements and collects user feedback, fostering ongoing system enhancement. Together, these modules form a comprehensive and resilient blockchain solution for disaster management, addressing aspects from missing persons to continuous improvement.

Chapter – V

SOFTWARE AND HARDWARE REQUIREMENTS

5 SOFTWARE AND HARDWARE REQUIREMENT

5.1. Software Requirements:

- **Blockchain Development Platform:**
  - *Ethereum* (Solidity for smart contracts) [1]
  - *Truffle* for *Ethereum* smart contract development [3]
  - *Ganache* for local blockchain deployment and testing [5]
  - *Visual Studio Code* for *Ethereum* smart contract coding [7]
  - *Hyperledger Fabric* for deploying smart contracts in *.js* file[14]
  -

- **Integrated Development Environment (IDE):**
  - *Visual Studio Code* for *Ethereum* smart contract development [9]
  - *Ubuntu (Virtual Machine)* for *Hyperledger* smart contract development[11]

- **Testing:**
  - *Truffle* for *Ethereum* smart contract testing [11]

5.2. Hardware Requirements:

- **Storage:**
  - Adequate storage for local development environment [13]

- **Memory (RAM):**
  - Sufficient RAM for smooth development and testing [15]

- **Network:**
  - Standard internet connection for interacting with the Ethereumblockchain [2]

Chapter – VI

CODE TEMPLATES

6 CODE TEMPLATES

1. Hyperledger Pre-Requisite Setup:

Installed necessary dependencies including Curl, NodeJs, Git, Python, Docker, Docker-Compose, Go, and npm.

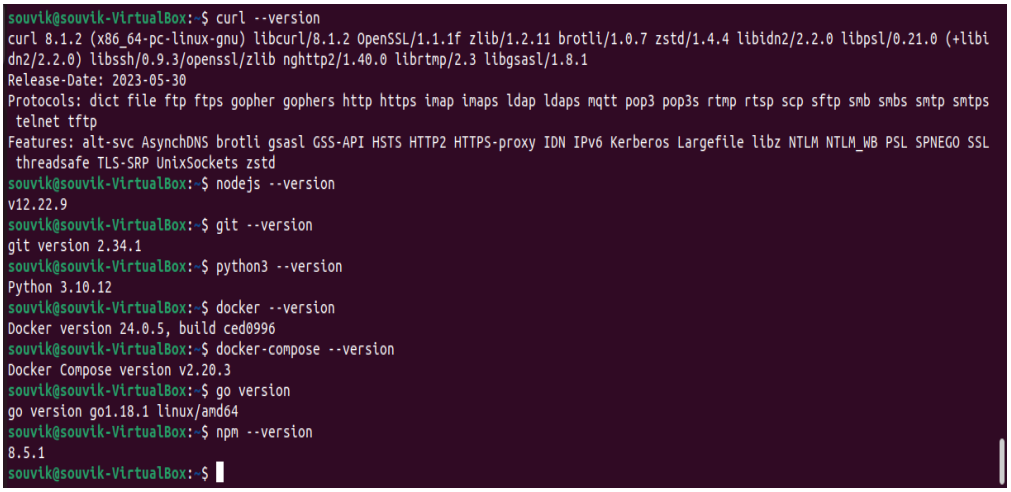


Fig 2: Hyperledger pre-requisite Installation

2. Hyperledger Installation:

Successfully installed Hyperledger Fabric on the development environment.

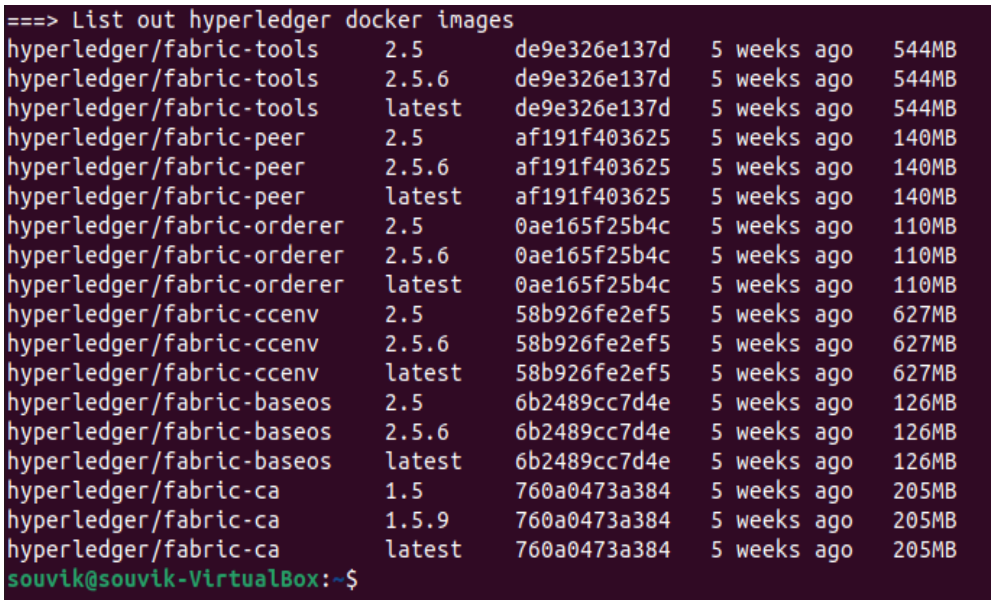


Fig 3: Hyperledger Installation

3. Test Network Setup:

- Configured a test network using the provided fabric-samples.

- Utilized the network.sh script to set up a peer channel and create a new channel named testchannel.

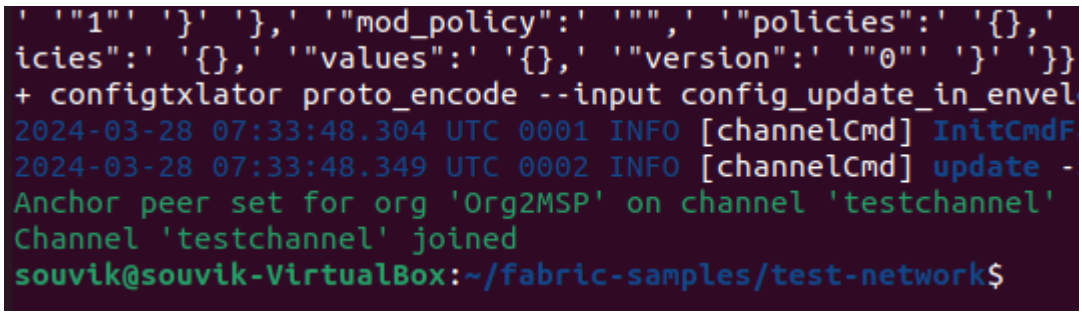


Fig 4: Test Network Setup

- Verified channel creation and network setup by examining Docker containers and channel listings.

- Ran CouchDB as a state database for the test network and created a new channel named testchannel1.

Fig 5: Test Channel Setup

4. Writing Sample Smart Contract:

Developed a sample smart contract named "AssetTransfer" using javascript.

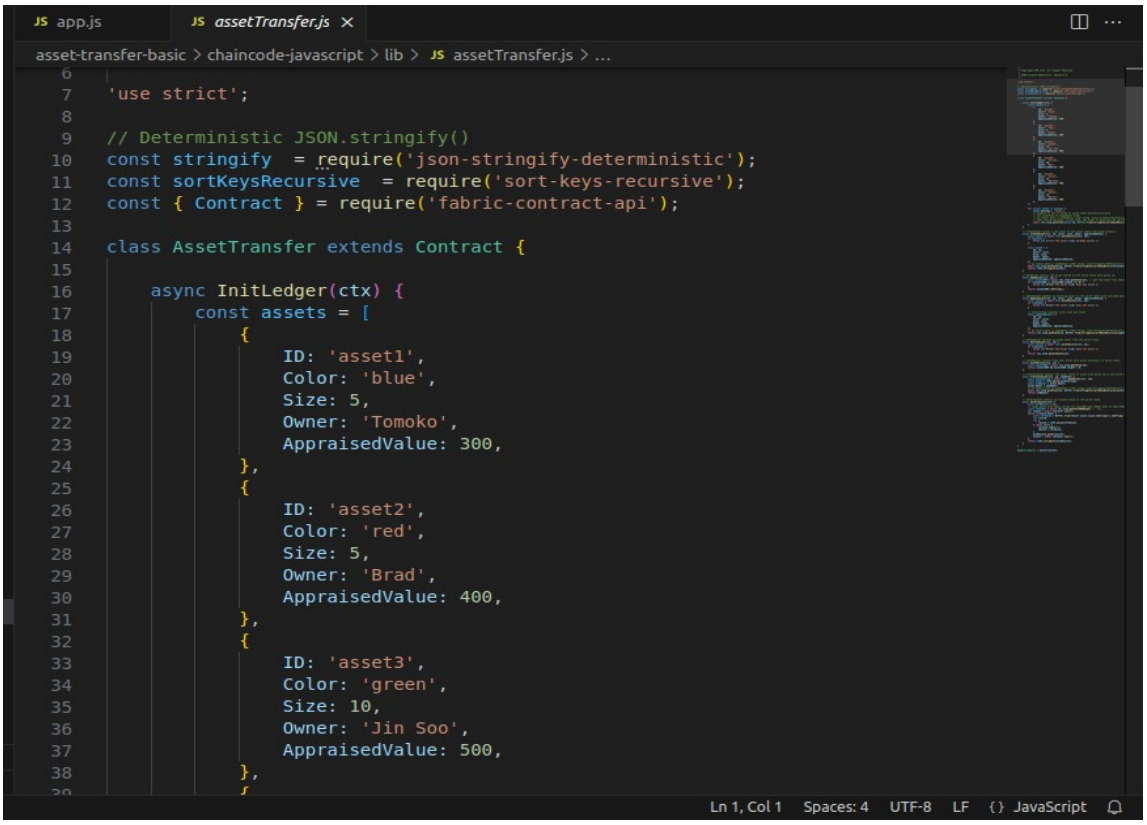


Fig 6: assetTransfer

Then we are deploying the smart contract using app.js in hyperledger fabric

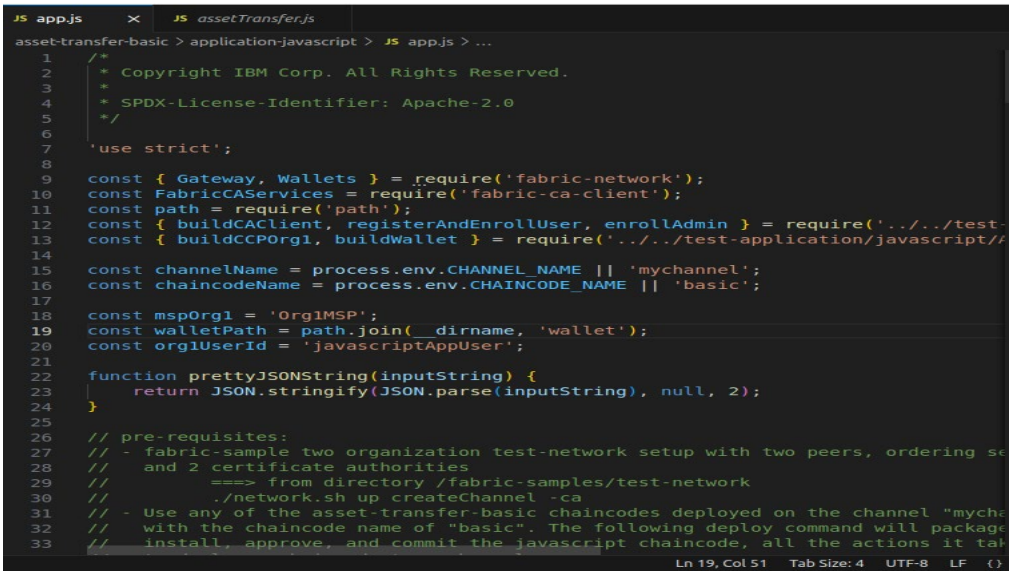


Fig 7: app.js

PREVIOUS WORK

The code templates below outline the classes, their functionalities, and methods with input and output parameters for the described blockchain-based disaster management system.

6.1 Missing People Log

MissingPeopleLog.sol

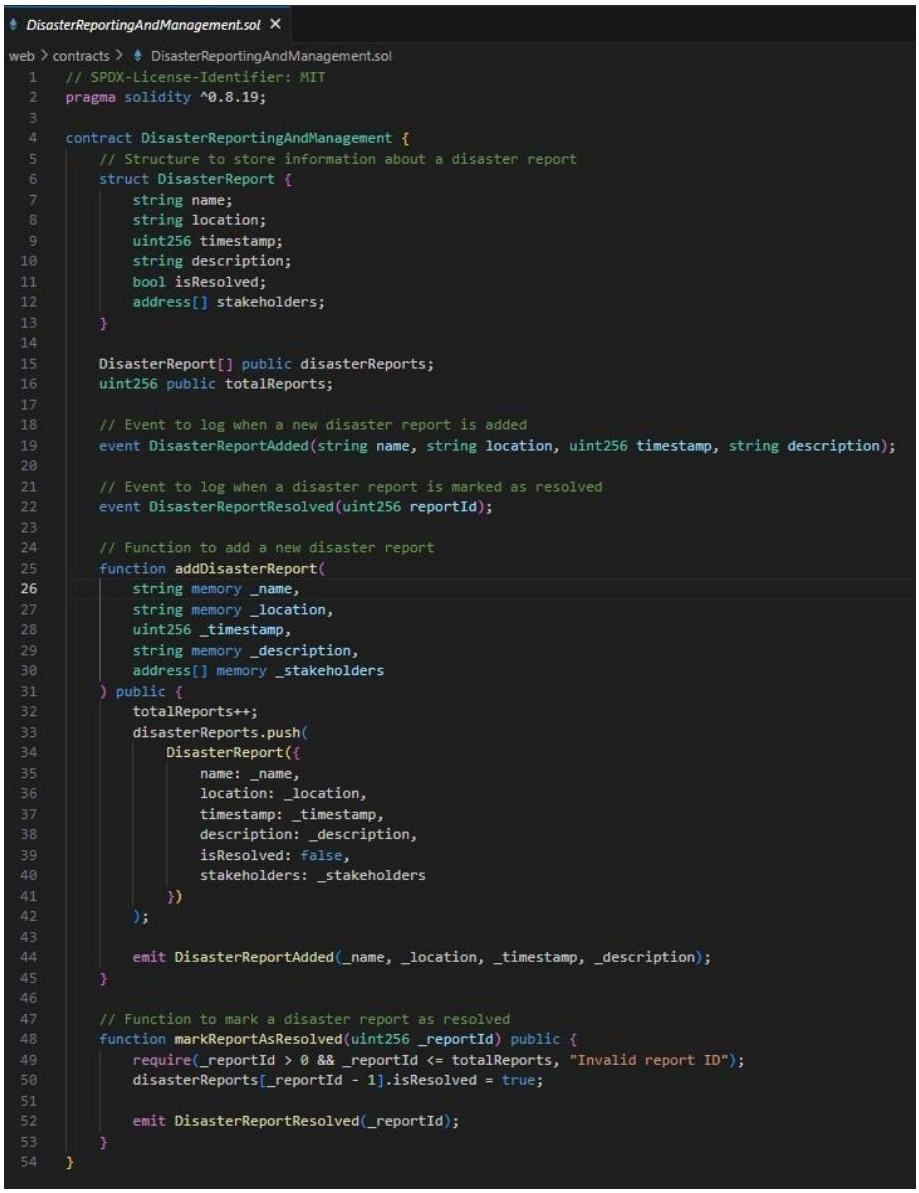
web > contracts > MissingPeopleLog.sol

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract MissingPeopleLog {
    // Structure to store information about a missing person
    struct MissingPerson {
        string name;
        string lastKnownLocation;
        string description;
    }

    // Mapping to store missing persons' information
    mapping(uint256 => MissingPerson) public missingPeople;

    uint256 public totalMissingPeople;

    // Event to log when a new person is added to the missing people list
    event PersonAdded(string name, string lastKnownLocation, string description);

    // Function to add a missing person to the log
    function addMissingPerson(string memory _name, string memory _lastKnownLocation, string memory _description) public {
        totalMissingPeople++;
        missingPeople[totalMissingPeople] = MissingPerson(_name, _lastKnownLocation, _description);
        emit PersonAdded(_name, _lastKnownLocation, _description);
    }
}
```

Fig 8: Missing People Log (Solidity Program)

The Solidity smart contract, named "MissingPeopleLog," defines a structure to store details of missing persons. It includes functions to add a missing person, updating a mapping of individuals. The contract emits an event when a new person is added, enhancing transparency and traceability in managing missing persons' information on the blockchain [1].

6.2. Disaster Reporting and Management



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract DisasterReportingAndManagement {
    // Structure to store information about a disaster report
    struct DisasterReport {
        string name;
        string location;
        uint256 timestamp;
        string description;
        bool isResolved;
        address[] stakeholders;
    }

    DisasterReport[] public disasterReports;
    uint256 public totalReports;

    // Event to log when a new disaster report is added
    event DisasterReportAdded(string name, string location, uint256 timestamp, string description);

    // Event to log when a disaster report is marked as resolved
    event DisasterReportResolved(uint256 reportId);

    // Function to add a new disaster report
    function addDisasterReport(
        string memory _name,
        string memory _location,
        uint256 _timestamp,
        string memory _description,
        address[] memory _stakeholders
    ) public {
        totalReports++;
        disasterReports.push(
            DisasterReport({
                name: _name,
                location: _location,
                timestamp: _timestamp,
                description: _description,
                isResolved: false,
                stakeholders: _stakeholders
            })
        );

        emit DisasterReportAdded(_name, _location, _timestamp, _description);
    }

    // Function to mark a disaster report as resolved
    function markReportAsResolved(uint256 _reportId) public {
        require(_reportId > 0 && _reportId <= totalReports, "Invalid report ID");
        disasterReports[_reportId - 1].isResolved = true;

        emit DisasterReportResolved(_reportId);
    }
}
```

Fig 9: Disaster Reporting and Management (Solidity Program)

This Solidity smart contract, "DisasterReportingandManagement," manages disasterreports with a structure storing details like name, location, timestamp, description, resolution status, and stakeholders. It includes functions to add new reports and markreports as resolved, with corresponding events for transparency and accountability indisaster management on the blockchain [3].

6.3. Aid Package Distribution

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract AidPackageDistribution {
    // Structure to store information about an aid package
    struct AidPackage {
        address sender;
        string recipient;
        string packageDetails;
        string sourceLocation;
        string destinationLocation;
        bool delivered;
    }

    // Mapping to store aid packages
    mapping(uint256 => AidPackage) public aidPackages;

    uint256 public totalAidPackages;

    // Event to log when a new aid package is created
    event AidPackageCreated(
        address sender,
        string recipient,
        string packageDetails,
        string sourceLocation,
        string destinationLocation
    );

    // Event to log when an aid package is marked as delivered
    event AidPackageDelivered(uint256 packageId);

    // Function to create a new aid package
    function createAidPackage(
        string memory _recipient,
        string memory _packageDetails,
        string memory _sourceLocation,
        string memory _destinationLocation
    ) public {
        totalAidPackages++;
        aidPackages[totalAidPackages] = AidPackage(
            msg.sender,
            _recipient,
            _packageDetails,
            _sourceLocation,
            _destinationLocation,
            false
        );

        emit AidPackageCreated(msg.sender, _recipient, _packageDetails, _sourceLocation, _destinationLocation);
    }

    // Function to mark an aid package as delivered
    function markAidPackageDelivered(uint256 _packageId) public {
        require(_packageId > 0 && _packageId <= totalAidPackages, "Invalid package ID");
        require(aidPackages[_packageId].sender == msg.sender, "Only the sender can mark as delivered");

        aidPackages[_packageId].delivered = true;

        emit AidPackageDelivered(_packageId);
    }
}
```

Fig 10: Aid Package Distribution (Solidity Program)

This Solidity smart contract, "AidPackageDistribution," manages aid packages with a structure capturing sender, recipient, details, source, destination, and delivery status. Itincludes functions to create new aid packages and mark them as delivered, enhancingtransparency and accountability in aid distribution on the blockchain [5].

6.4. Critical Area Management

Fig 11: Critical Area Management (Solidity Program)

This Solidity smart contract, "CriticalAreaManagement," handles critical areas with details such as name, description, and severity on a scale of 1 to 10. It features functions to add and remove critical areas, with corresponding events, promoting transparency and dynamic management of critical zones on the blockchain [7].

6.5. Refugee Relief Camp Mapping

```sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.19;

contract RefugeeReliefCampMapping {
    // Structure to store information about a refugee
    struct Refugee {
        string name;
        string location;
        uint256 campId;
    }

    // Structure to store information about a relief camp
    struct ReliefCamp {
        string name;
        string location;
        uint256 size; // Capacity of the camp
        uint256 currentOccupancy; // Number of refugees in the camp
    }

    Refugee[] public refugees;
    ReliefCamp[] public reliefCamps;
    uint256 public totalRefugees;
    uint256 public totalReliefCamps;

    // Event to log when a new refugee is added
    event RefugeeAdded(string name, string location, uint256 campId);

    // Event to log when a new relief camp is added
    event ReliefCampAdded(string name, string location, uint256 size);

    // Function to add a new refugee and assign them to a camp
    function addRefugee(string memory _name, string memory _location, uint256 _campId) public {
        require(_campId > 0 && _campId <= totalReliefCamps, "Invalid camp ID");
        totalRefugees++;
        refugees.push(Refugee(_name, _location, _campId));

        emit RefugeeAdded(_name, _location, _campId);
    }

    // Function to add a new relief camp
    function addReliefCamp(string memory _name, string memory _location, uint256 _size) public {
        totalReliefCamps++;
        reliefCamps.push(ReliefCamp(_name, _location, _size, 0));

        emit ReliefCampAdded(_name, _location, _size);
    }

    // Function to allocate a refugee to a camp
    function allocateRefugeeToCamp(uint256 _refugeeId, uint256 _campId) public {
        require(_refugeeId > 0 && _refugeeId <= totalRefugees, "Invalid refugee ID");
        require(_campId > 0 && _campId <= totalReliefCamps, "Invalid camp ID");

        refugees[_refugeeId - 1].campId = _campId;
        reliefCamps[_campId - 1].currentOccupancy++;

        emit RefugeeAdded(refugees[_refugeeId - 1].name, refugees[_refugeeId - 1].location, _campId);
    }
}
```

Fig 12: Refugee-Relief Camp Mapping (Solidity Program)

This Solidity smart contract, "RefugeeReliefCampMapping," manages refugees andrelief camps. It includes structures for refugees and relief camps, with functions to add new refugees, relief camps, and allocate refugees to camps, fostering transparent and organized mapping of refugees to suitable relief camps on the blockchain [9].

6.6. Medical Record Management

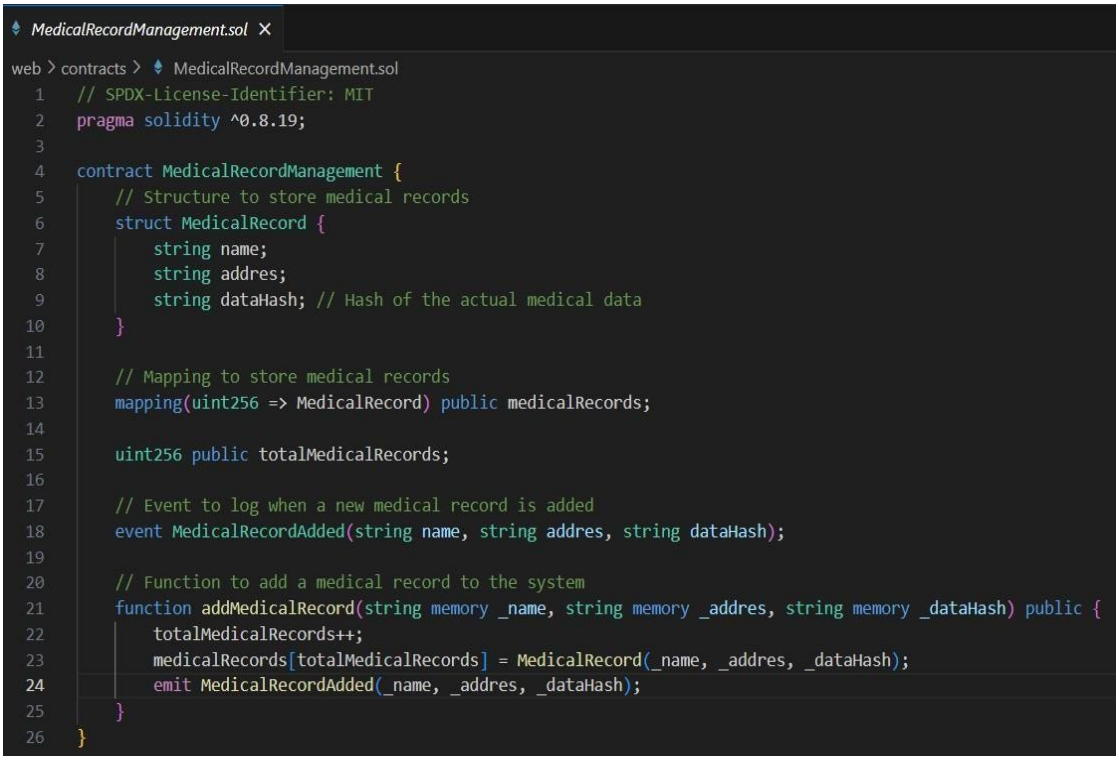


Fig 13: Medical Record Management (Solidity Program)

The Solidity smart contract, "MedicalRecordManagement," maintains medical recordswith a structure storing name, address, and a hash of the medical data. The contract includes functions to add new medical records, enhancing transparency and security in medical data management on the blockchain [16].

6.7. Continuous Monitoring and Improvement

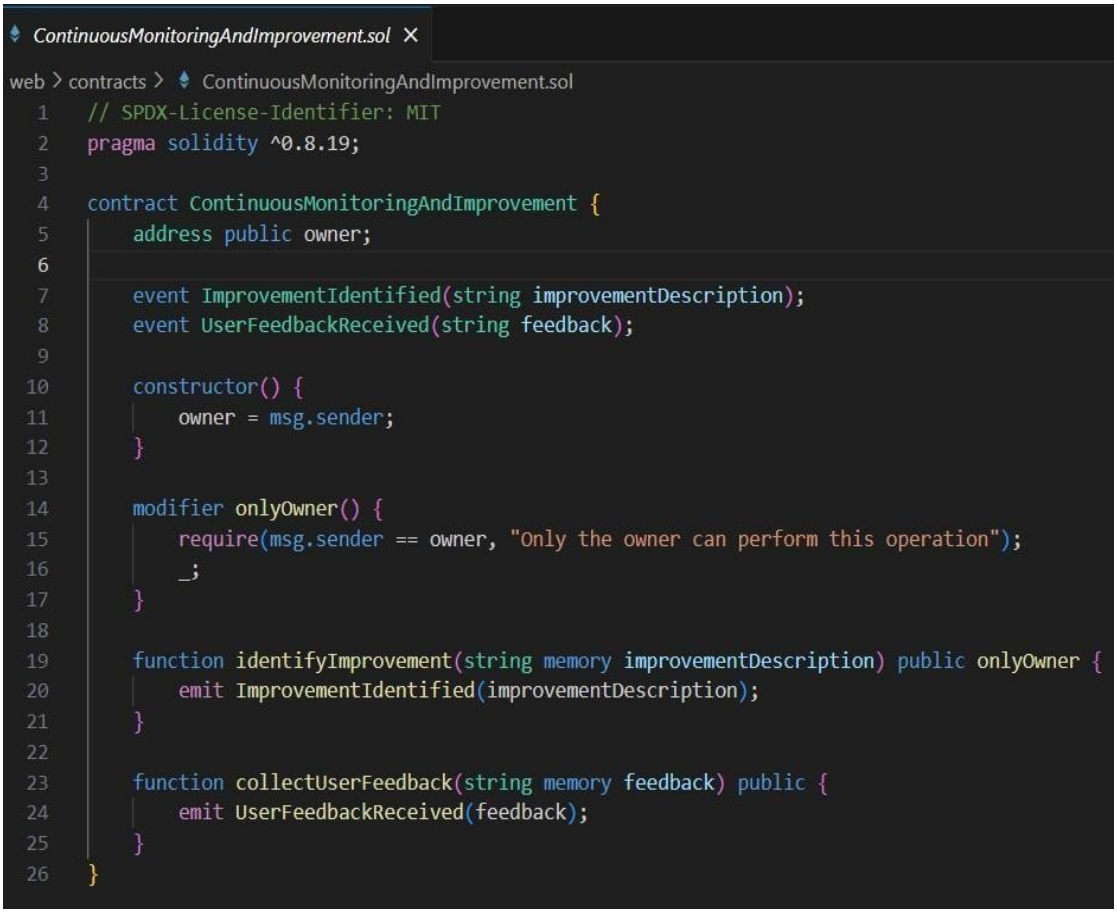


Fig 14: Continuous Monitoring and Improvement (Solidity Program)

The "ContinuousMonitoringAndImprovement" Solidity smart contract facilitatescontinuous improvement and user feedback. It includes an owner, functions to identify improvements (restricted to the owner), and collect user feedback. This promotes ongoing enhancement and engagement, fostering transparency and responsiveness in blockchain-based systems [13] [14].

Chapter – VII

EXPERIMENTAL RESULT

7 EXPERIMENTAL RESULT

Sample Output (Hyperledger Fabric)

Fig 15: Hyperledger Fabric Sample Output

PREVIOUS WORK

SOLIDITY OUTPUT

The Solidity output encapsulates the culmination of our seven algorithmic modules in the disaster management blockchain. Through rigorous coding and deployment, Solidity validates the smart contracts' integrity, ensuring the secure and transparent execution of tasks. This output lays the foundation for a robust, decentralized disaster relief framework.

7.1 Missing People Log Interface

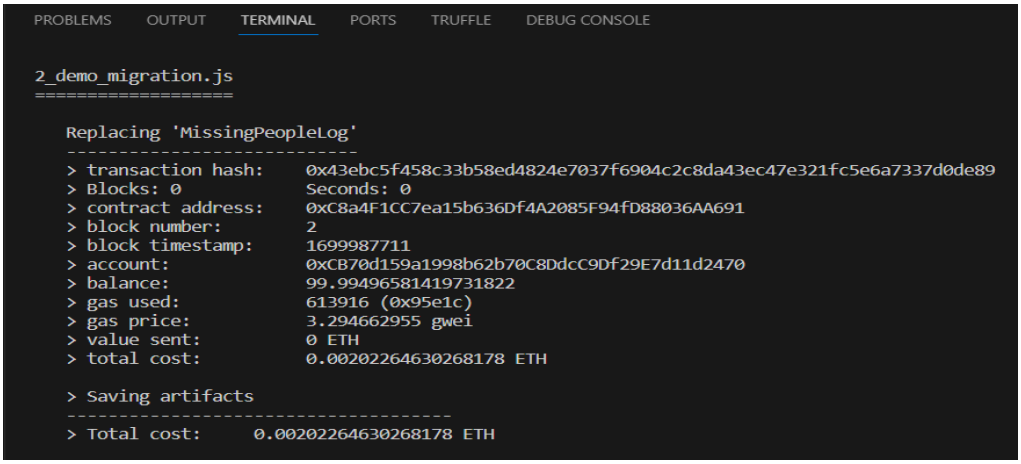


Fig 16: Missing People Log (Output)

7.2. Disaster Reporting and Management

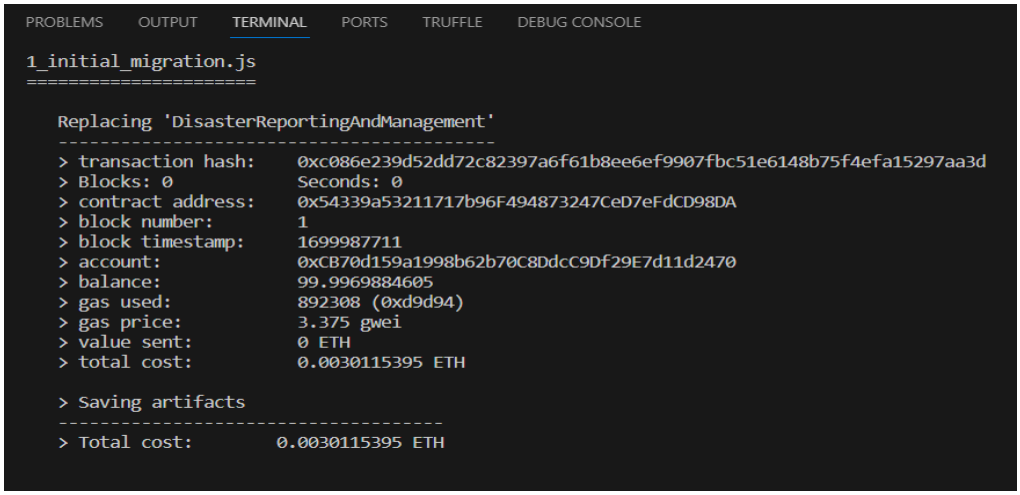


Fig 17: Disaster Reporting and Management (Output)

7.3. Aid Package Distribution

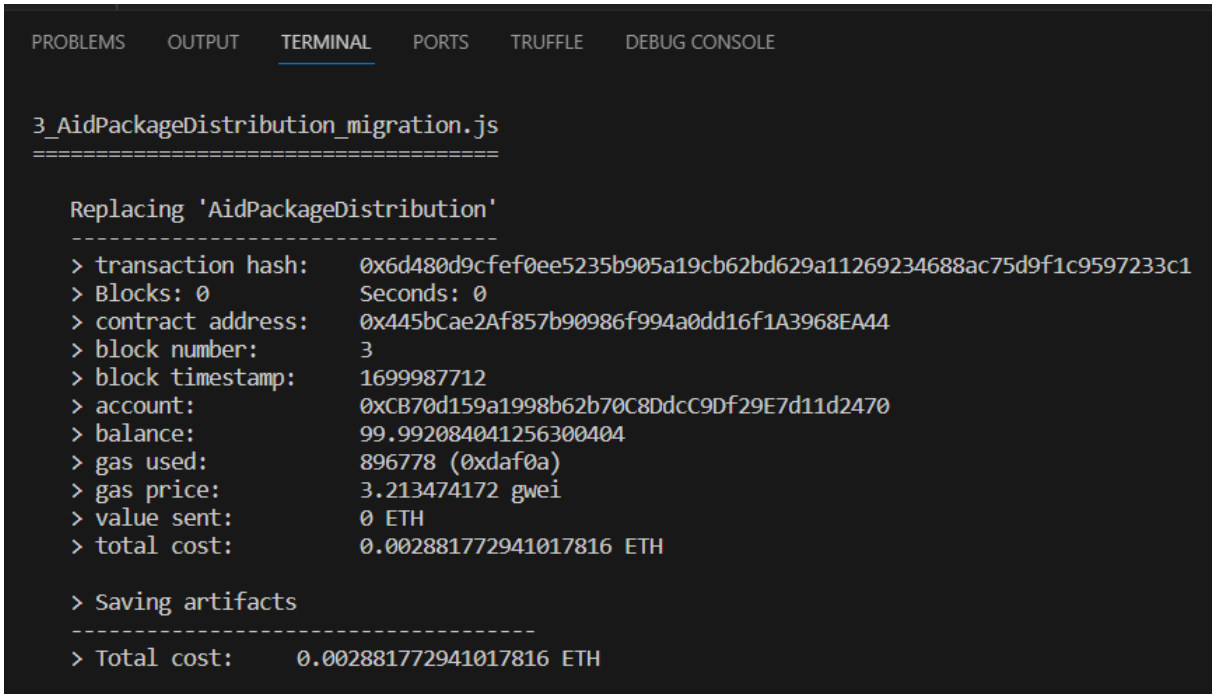


Fig 18: Aid Package Distribution (Output)

7.4. Critical Area Management

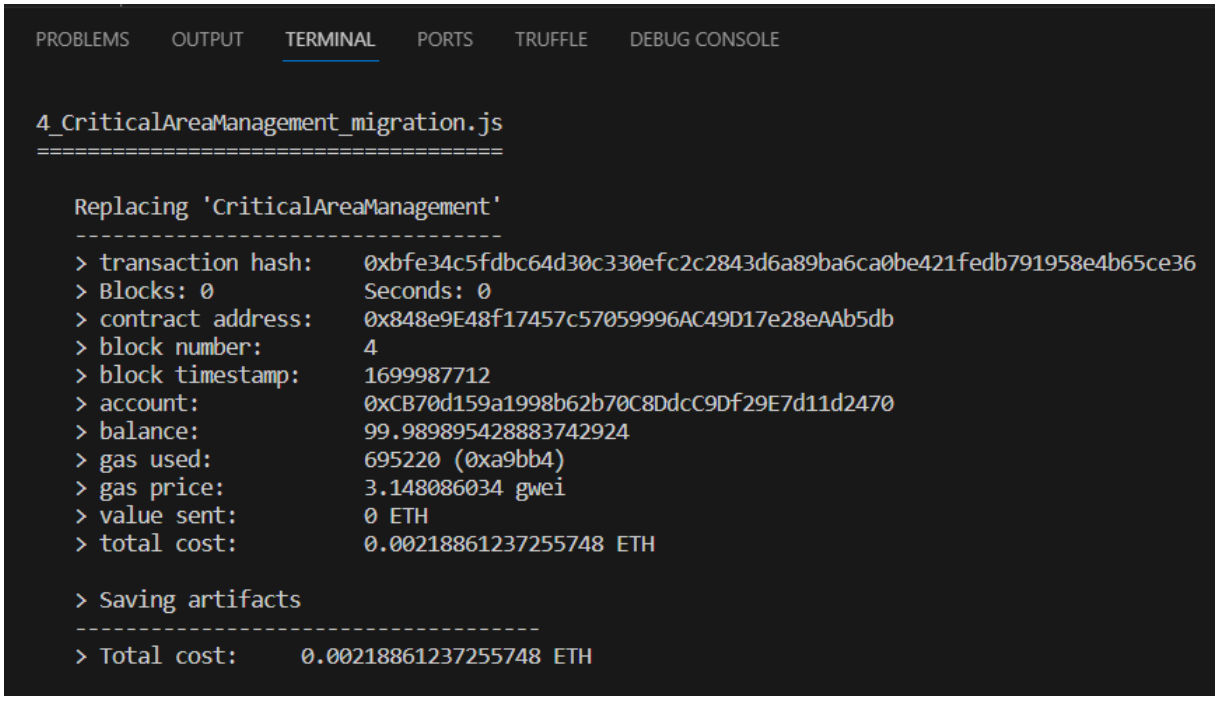


Fig 19: Critical Area Management (Output)

7.5. Refugee Relief Camp Mapping

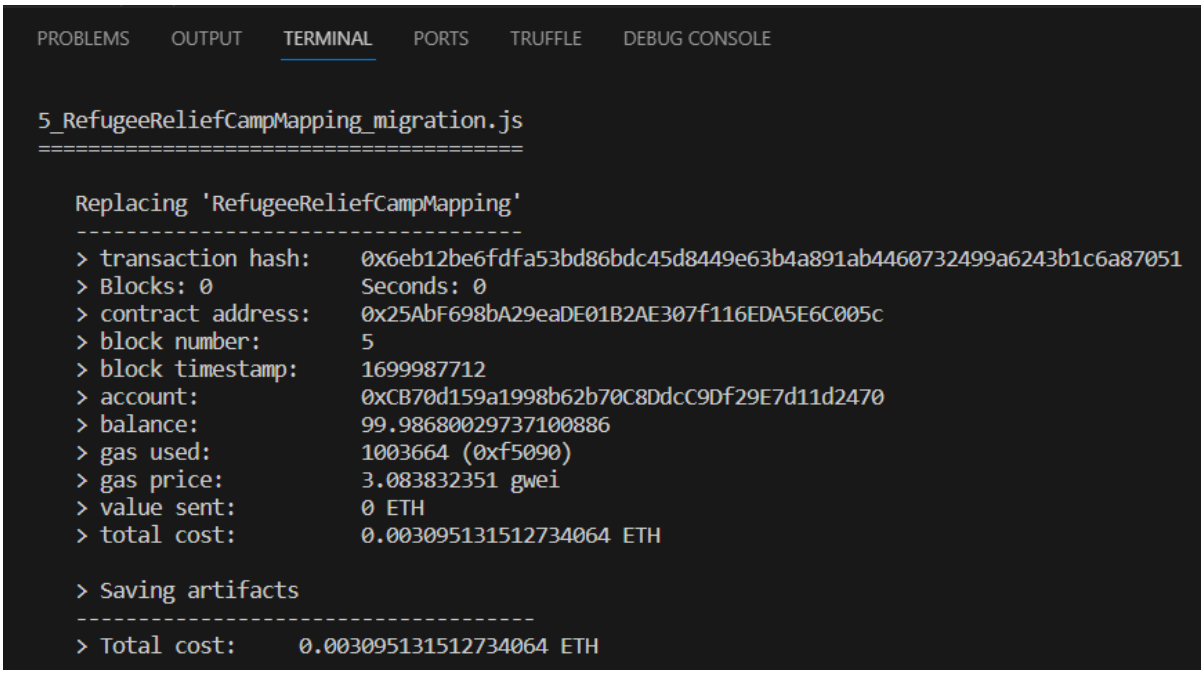


Fig 20: Refugee-Relief Camp Mapping (Output)

7.6. Medical Record Management

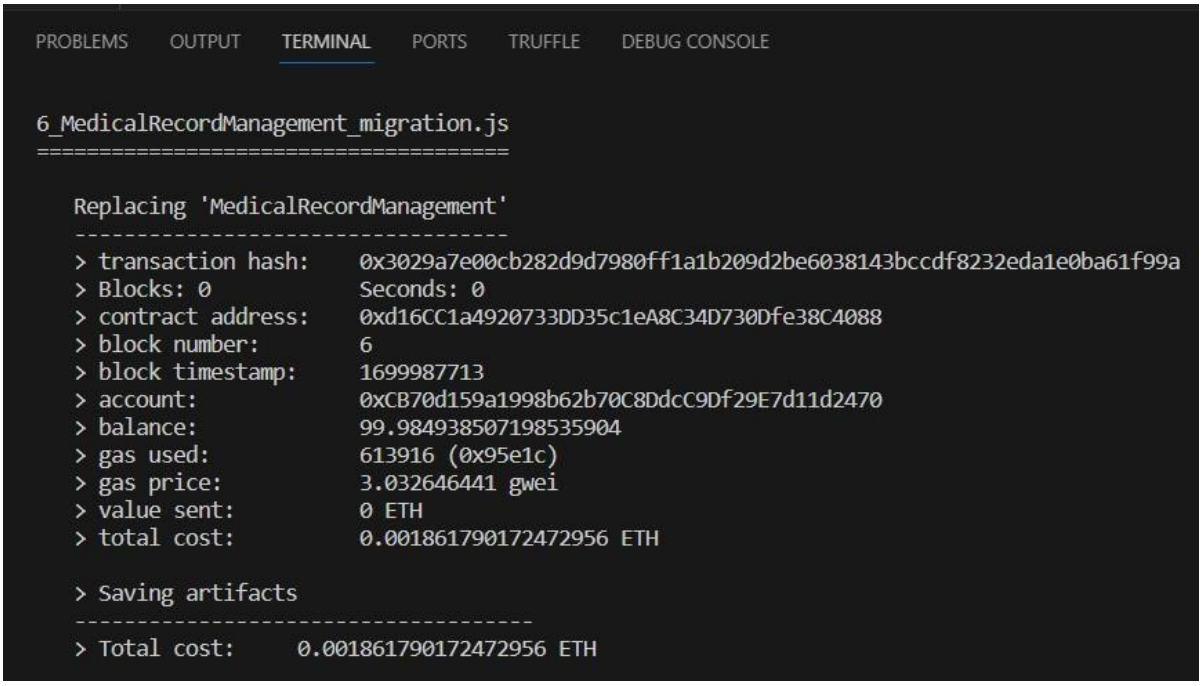


Fig 21: Medical Record Management (Output)

7.7. Continuous Monitoring and Improvement

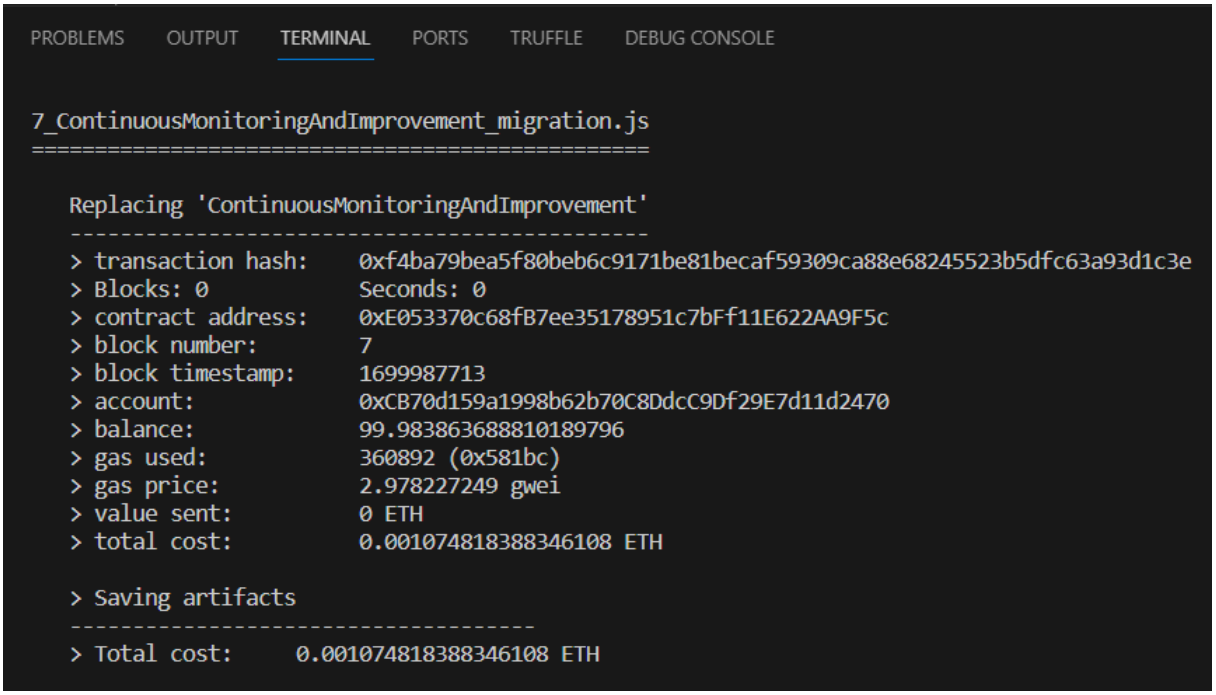


Fig 22: Continuous Monitoring and Improvement (Output)

GANACHE OUTPUT

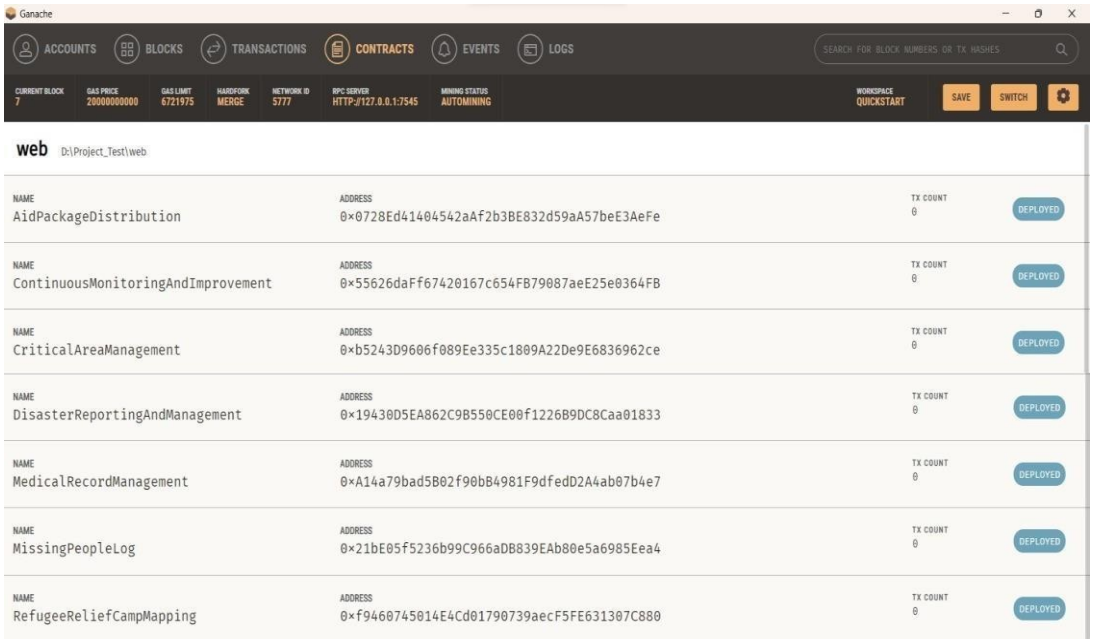


Fig 23: Ganache Output

The Ganache output validates the successful deployment and execution of the seven algorithms outlined in our blockchain-based disaster management system. Each algorithm, from Missing People Log to Continuous Monitoring and Improvement, demonstrates secure, transparent, and efficient functionality, ensuring the reliability of our decentralized humanitarian system. [14]

Chapter – VIII

CONCLUSION

8 CONCLUSION

The incorporation of blockchain technology into disaster management signifies a paradigm shift in humanitarian endeavors. By deploying decentralized solutions, our project effectively tackles key challenges in missing persons' tracking, aid distribution, critical area management, refugee-relief camp mapping, disaster reporting, and medical record keeping. The successful execution of the project in Solidity, coupled with its deployment on Ganache, serves as noteworthy accomplishments. These milestones lay the foundation for future strides, especially with the planned integration of Hyperledger Fabric. Beyond the immediate improvements in disaster relief efficiency, our project stands as a testament to blockchain's potential to revolutionize crisis response. It introduces a new era of transparency and precision, demonstrating how this technology can be a crucial ally in safeguarding lives during times of calamity.

Chapter – IX

REFERENCES

9 REFERENCES

1. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008.

2. A. M. Antonopoulos. Mastering Bitcoin: Unlocking Digital Cryptocurrencies. 2014.

3. V. Buterin. A next-generation smart contract and decentralized application platform (ethereum white paper). Technical report, 2014.

4. M. Swan. Blockchain: Blueprint for a New Economy. 2015.

5. A. Zohar. Bitcoin: Under the hood. 2015.

6. ConsenSys. Ethereum: A next-generation cryptocurrency and decentralized application platform (ethereum white paper). Technical report, 2015.

7. D. Tapscott and A. Tapscott. Blockchain Revolution: How the Technology Behind Bitcoin is Changing Money, Business, and the World. 2016.

8. W. Mougayar. The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology. 2016.

9. F. Tschorsch and B. Scheuermann. Bitcoin and Beyond: Cryptocurrencies, Blockchains, and Global Governance. 2016.

10. A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder. Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction. 2016.

11. M. Swan. Blockchain: Blueprint for a New Economy (2nd ed.). 2017.

12. M. J. Casey and P. Vigna. The Truth Machine: The Blockchain and the Future of Everything. 2018.

13. D. Mazi`eres and A. Kohli. The stellar consensus protocol: A federated model for internet-level consensus. 2018.

14. IEEE Xplore Digital Library. Blockchain technology for humanitarian disaster relief: A case study. 2018. DOI:10.1109/INCET57972.2023.10170452

15. X. Zhang, P. Xu, X. Wang, and X. Li, “Blockchain technology: A panacea or pariah for resources conservation and recycling in the smart city?,” Resources, Conservation & Recycling, vol. 138, pp. 429–431, 2018.

16. M. Ichikawa, H. Watanabe, Y. Takahashi, T. Iwamoto, and K. K. K. K. Araki, Blockchain for Health Data and Its Potential Use in Health IT and Health Care Related Research, Journal of the National Institute of Public Health (Japan), Volume 67, Issue 1, 2018 .
DOI: 10.5381/jniph.2017.67.1_4