TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Final Project Report

Blockchain Technology for Smart Cities
Waste Management Scenario

Created by

Amal Jawahdou

Farah Jaouadi

Nour N.GHSAIER

Github Link :

`https://github.com/Amal146/WasteManagementHederaDapp.git`

Supervised by	Academic year	University
Prof. Mariem Thaalbi	**2024 - 2025**	**Tunis Business School**

Contents

Introduction

Context and Motivation

The rapid growth of urbanization and technological advancements has transformed the way cities operate, giving rise to the concept of **smart cities**. These are urban areas that leverage technology and data to improve efficiency, sustainability, and quality of life for their residents. However, with this transformation come significant challenges:

- **Data security and privacy**: The interconnected nature of smart cities increases the risk of cyberattacks and unauthorized data access.
- **Lack of transparency**: Centralized systems often hinder accountability in resource management and service delivery.
- **Complex coordination**: Managing various stakeholders and systems in a city demands efficient and reliable frameworks.

Blockchain technology offers a robust solution to these challenges by providing a decentralized, secure, and tamper-resistant system.

In the context of smart cities, blockchain can enhance transparency in governance, enable efficient resource allocation, and ensure the integrity of critical operations such as identity management, energy trading, and public safety coordination.

This project explores how blockchain can be effectively designed and implemented to address these challenges, ultimately contributing to the realization of smarter, more secure urban environments.

Objectives of the Project

The primary goal of this project is to design, develop, and evaluate a blockchain-based solution tailored for smart cities. This includes:

1. **Design Phase**:
    - Identifying key application areas in smart cities, such as energy management, supply chain transparency, and public services.
    - Defining the assets, stakeholders, and workflows that interact within the blockchain ecosystem.
2. **Development Phase**:
    - Implementing smart contracts to automate and secure various processes in smart city scenarios.
    - Ensuring scalability, security, and efficiency in the blockchain application.
3. **Evaluation Phase**:
    - Testing the blockchain system for performance, reliability, and usability.
    - Demonstrating the system's ability to address real-world challenges in smart cities.

Through these phases, the project aims to contribute a practical, scalable solution that leverages blockchain's potential in urban innovation.

Methodology Overview

The project follows a structured, phased methodology to ensure a comprehensive approach to problem-solving:

1. **Exploration and Analysis**:
    - Conduct a literature review to understand the intersection of blockchain and smart cities.
    - Identify use cases where blockchain provides a distinct advantage.
2. **Design and Modeling**:
    - Develop system architecture diagrams and smart contract models tailored to smart city applications.
    - Utilize tools like UML and PlantUML for clear documentation and visualization.
3. **Implementation**:
    - Build the blockchain application using appropriate platforms (e.g., Ethereum, Hyperledger).
    - Write and deploy smart contracts addressing specific use cases, such as identity management and energy trading.

4. **Testing and Validation**:
    - Test the blockchain solution in controlled environments to assess its effectiveness.
    - Gather feedback from simulated stakeholders for iterative improvements.
5. **Documentation and Presentation**:
    - Document the findings, challenges, and outcomes of the project.
    - Present the solution, highlighting its impact on smart city challenges.

This methodology ensures a thorough exploration of blockchain's potential while maintaining a focus on practical application and measurable outcomes.

Chapter 1

Design the Blockchain Solution for a Smart City

This chapter focuses on the foundational design of a blockchain solution tailored for a smart city ecosystem. It outlines the key participants, the rationale for selecting a specific blockchain type, and the necessary components such as smart contracts and assets. By establishing a well-structured blockchain design, we aim to address the complexities of urban management, enhance transparency, and foster collaboration among diverse stakeholders in a smart city environment.

1.1 Participants in the Blockchain for a Smart City

In a smart city blockchain solution, various stakeholders play critical roles. Below are the key participants categorized by their sectors:

Category	**Subcategories**
Government Entities	• Local Governments • National Governments • Regulatory Bodies • Law Enforcement and Emergency Response Agencies

Private Sector	• Technology Companies • Construction and Engineering Firms • Telecommunications Providers
Academia and Research Institutions	• Universities • Think Tanks: Research and policy organizations
Community and Citizens	• Residents • Community Organizations
Non-Governmental Organizations (NGOs)	• Advocacy • Research Groups
Investors and Financial Institutions	• Venture Capitalists • Banks
Utility Companies	• Energy • Waste Management Services

1.2 Roles of Participants in the Blockchain

Government Entities	
Local Governments	• **Validate** data related to public services (e.g., land registries, citizen identities). • **Manage** local infrastructure and services data on the blockchain. • **Track** real-time city conditions and resource usage for effective management. • **Collaborate** with other government entities and private sector partners.
National Governments	• **Coordinate** with international organizations and other countries. • **Ensure** data privacy and security standards are met. • **Provide** funding and resources for smart city projects. • **Provide** access to aggregated data for economic and policy planning.

Regulatory Bodies	• **Audit** blockchain transactions to ensure adherence to regulations and legal requirements. • **Resolve** disputes between different stakeholders by reviewing immutable transaction records. • **Certify** and validate new participants or services added to the blockchain. • **Enforce/Upgrade** regulations and standards for smart city technologies.
Law Enforcement and Emergency Response Agencies	• **Access** real-time data to respond to emergencies and crimes. • **Investigate** incidents and collect evidence using blockchain records. • **Coordinate** with other agencies to optimize response times.
Private Sector	
Technology Companies	• **Update** blockchain infrastructure to ensure smooth functionality. • **Provide** data analytics and AI solutions to optimize city operations. • **Offer** cybersecurity services to protect the blockchain network.
Construction and Engineering Firms	• **Utilize** blockchain to track construction materials and supply chains.

Telecommunications Providers	• **Provide** real-time data on network usage and connectivity within the smart city. • **Offer** cloud-based services to support smart city applications. • **Maintain** secure and seamless communication networks supporting blockchain operations.
Academia and Research Institutions	
Universities and Think Tanks	• **Conduct** research using blockchain data to enhance innovative smart city solutions. • **Analyze** trends and patterns from blockchain records to advance urban planning methodologies. • **Educate** and train future professionals in blockchain technology and its applications. • **Generate** policy recommendations based on insights from blockchain data.
Community and Citizens	
Residents	• **Access** and **verify** personal records such as identities, property ownership, and utility usage. • **Participate** in decision-making through blockchain-based voting, surveys, and feedback systems. • **Report** issues and concerns directly to city authorities.

Community Organizations	• **Advocate** for the needs and interests of community members. • **Track** and **report** on the impact of social programs and projects. • **Collaborate** with government and private entities to enhance local initiatives.
Non-Governmental Organizations (NGOs)	
Advocacy and Research Groups	• **Advocate** for transparency, accountability, and citizen participation. • **Raise** awareness about the benefits and risks of smart cities. • **Collaborate** with other stakeholders to promote equitable and sustainable urban development. • **Conduct** research on the social and environmental impacts of smart city technologies.
Investors and Financial Institutions	
Venture Capitalists and Banks	• **Support** the growth of the smart city ecosystem. • **Offer** financial products and services tailored to smart city initiatives. • **Manage** risk and assess the financial viability of smart city investments.

Utility Companies	
Energy and Waste Management Services	• **Optimize** energy distribution and consumption. • **Monitor** and **manage** waste disposal and recycling. • **Reduce** carbon emissions and promote sustainability. • **Provide** real-time data on energy usage and waste generation.

1.3 Justification for Federated Blockchain in Smart Cities

A Federated (or Consortium) Blockchain is an ideal choice for the proposed smart city project, as it effectively balances the needs for transparency and privacy among a diverse group of stakeholders. This type of blockchain allows multiple organizations to share control over the network, making it particularly suitable for scenarios where participants include government entities, private sector players, academia, and community organizations.

Definition of Federated Blockchain

A federated blockchain is a type of blockchain network that is governed by a group of pre-selected nodes or entities, rather than by a decentralized network of nodes. In a federated blockchain, the participating nodes or entities form a consortium and have the ability to make decisions about the governance and maintenance of the blockchain.[1]

Advantages of the consortium blockchain in the Smart City Context

1. **Transparency**: Public records such as energy usage, land registries, and environmental data can be securely stored on the blockchain, ensuring accountability and accessibility to all stakeholders.

2. **Privacy**: Sensitive data such as citizen identities and personal information are protected, as only authorized participants can access or modify this information.

3. **Collaboration**: Multiple organizations, including government bodies, private companies, and NGOs, can co-manage the network, ensuring fair participation and mutual trust.

4. **Scalability and Efficiency**: Compared to public blockchains, federated blockchains are more scalable and require fewer computational resources, making them cost-effective for managing large-scale urban operations.

5. **Security**: The controlled participation of trusted entities reduces the risk of malicious activities while still leveraging the cryptographic security of blockchain.

The following figure outlines the architecture of a Federated Blockchain tailored for smart cities, illustrating the collaboration among key stakeholders and the flow of information.

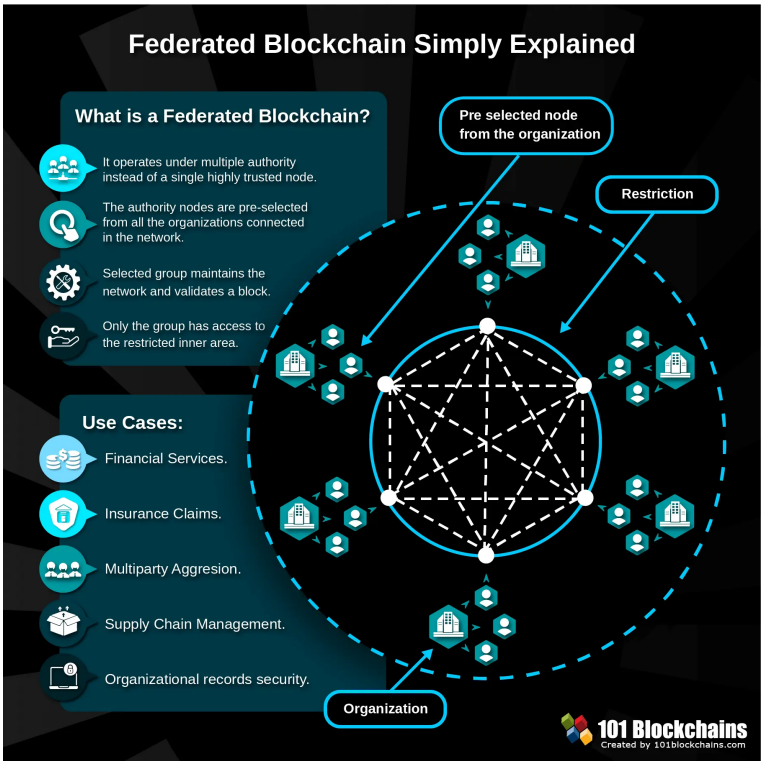

Figure 1.1: Federated Blockchain Simply Explained.
Source: 101Blockchains

1.4 Smart Contract Description

Smart contracts play a vital role in enabling secure, transparent, and automated interactions in a blockchain-based smart city solution.
Below is an overview of smart contracts tailored to each scenario, along with example functions.

1.4.1 Energy Management

Scenario: Facilitate peer-to-peer energy trading and ensure transparency in energy distribution.

Smart Contract Role: Automates energy sales, tracks consumption, and ensures fairness.
Example Functions:

- `offerEnergy(uint energyAmount, uint price)`: Allows producers to list energy for sale.
- `buyEnergy(address producer, uint energyAmount)`: Facilitates energy purchase directly between participants.
- `logTransaction(address buyer, address seller, uint energyAmount)`: Records completed transactions for transparency.

1.4.2 Supply Chain Management

Scenario: Ensure the traceability and authenticity of goods in the supply chain.
Smart Contract Role: Maintains immutable records from production to consumption.
Example Functions:

- `addGoods(string productDetails)`: Registers a product at the start of its lifecycle.
- `updateStatus(string productId, string status)`: Updates the product's status and location.
- `validateProduct(string productId)`: Verifies the product's history and authenticity.

1.4.3 Public Services

Scenario: Automate public services like waste management and procurement.
Smart Contract Role: Triggers services based on conditions and ensures transparency.
Example Functions:

- `triggerService(string serviceName)`: Initiates a public service (e.g., waste collection).
- `logServiceCompletion(string serviceName)`: Records service completion for accountability.

1.4.4 Identity Management

Scenario: Provide secure identity verification and facilitate transparent voting.
Smart Contract Role: Issues digital identities and manages secure voting.
Example Functions:

- `createIdentity(address user, string details)`: Issues a decentralized identity.

- `castVote(address voter, uint proposalId)`: Ensures secure and tamper-proof voting.

1.4.5 Transportation and Mobility

Scenario: Enable decentralized ride-sharing and autonomous vehicle coordination.
Smart Contract Role: Manages transactions and ensures communication between stakeholders.

Example Functions:

- `requestRide(address passenger, uint rideDetails)`: Passengers request a ride.
- `completeRide(address passenger, uint rideCost)`: Finalizes the ride and releases payment.

1.4.6 Data Sharing and Privacy

Scenario: Allow citizens to control access to their data and ensure secure IoT integration.
Smart Contract Role: Manages permissions and logs data transactions.

Example Functions:

- `grantAccess(address requester, string dataType)`: Gives access to specific data.
- `logDataTransfer(string dataType)`: Records the sharing of IoT data securely.

1.4.7 Real Estate and Land Registry

Scenario: Digitize property records and automate rental agreements.
Smart Contract Role: Maintains immutable records and facilitates automated agreements.

Example Functions:

- `registerProperty(address owner, string details)`: Records property ownership.
- `createRentalAgreement(address tenant, uint rentAmount)`: Automates rental transactions.

1.4.8 Public Safety and Security

Scenario: Secure evidence and coordinate emergency responses.
Smart Contract Role: Protects sensitive data and ensures efficient coordination.

Example Functions:

- `storeEvidence(string evidenceHash)`: Stores encrypted evidence securely.

- `triggerEmergencyResponse(string details)`: Initiates response actions among agencies.

1.4.9 Urban Planning and Governance

Scenario: Ensure transparency in budget allocation and citizen involvement.
Smart Contract Role: Tracks funds and facilitates proposal voting.
Example Functions:

- `allocateFunds(string department, uint amount)`: Records fund allocations.
- `castVote(address voter, uint proposalId)`: Records votes on governance proposals.

1.4.10 Waste Management

Scenario: Track waste disposal and recycling processes.
Smart Contract Role: Logs waste data and verifies compliance with regulations.
Example Functions:

- `trackWaste(string wasteId, string location)`: Updates waste journey details.
- `logRecycling(string wasteId, string recycler)`: Confirms waste recycling.

1.5 Blockchain Application Assets

Identifying the key assets in the Blockchain Application is crucial to understanding its functionality and scope. These assets serve as the foundation for transactions, interactions, and processes within the system. Below are the main assets considered:

- **Energy Units:** Quantities of energy available for trade between households or businesses.
- **Goods/Products:** Unique items in the supply chain, enabling transparent tracking.
- **Digital Identities:** Decentralized identity data ensuring secure citizen authentication.
- **Transportation Tokens:** Units for accessing ride-sharing services or autonomous vehicles.
- **Property Records:** Digital representations of property titles and ownership histories.
- **Contracts:** Smart contracts regulating services like waste collection or procurement.
- **Public Funds:** Blockchain representations of budgets for transparent allocation and monitoring.

Chapter 2

Development of the Blockchain Solution

2.1 Study of Blockchain Development Tools

What is Hyperledger Fabric?

Hyperledger is an open-source collaborative effort created to advance cross-industry blockchain technologies. It is a global collaboration, hosted by The Linux Foundation, including leaders in finance, banking, IoT, supply chain, manufacturing, and technology [2]. It follows a permissioned and private approach designed for enterprise use. It supports privacy as in only members of the network can access the system and scalability, making it an attractive option for enterprises that need to control access to data while maintaining network security and performance.

Hyperledger provides several advantages that make it a suitable choice for business:

- **Permissioned Network:** Only authorized members can access the network.
- **Modular Architecture:** Network components are customizable like consensus mechanisms, identity services, and ledger storage, which provides flexibility to meet specific business requirements.
- **Data Privacy:** Supports private channels and data collections. Enables selective data sharing among authorized participants while keeping sensitive information confidential.
- **Chaincode (Smart Contracts):** Enables automation of business processes using Chaincode. Supports familiar programming languages like Java, Golang, and Node.js, reducing the learning curve for developers.
- **Scalability:** Designed to scale horizontally, accommodating large and growing networks without performance degradation. Ideal for high transaction volumes.

What is Ethereum?

Ethereum is a global, public, and open-source platform for decentralized applications (DApps) that establishes a peer-to-peer network that securely executes and verifies application code, called smart contracts. Founded by Vitalik Buterin in 2015, it has become one of the most popular platforms for building decentralized applications.

Ethereum offers several benefits:

- **Decentralization:** Ethereum operates where no single entity has control.
- **Smart Contracts:** Ethereum introduced smart contracts, which are self-executing agreements with predefined rules. They automate processes, reducing the need for intermediaries.
- **Open-Source Platform:** Enables developers worldwide to collaborate, contribute, and build solutions.
- **Ethereum Virtual Machine (EVM):** The EVM allows developers to write and deploy code in multiple programming languages, such as Solidity and Vyper, making the platform accessible and versatile.
- **Programmability:** Developers can create smart contracts and DApps for use cases.

What is Hedera?

Hedera is a decentralized, open-source, proof-of-stake public ledger that utilizes the leaderless, asynchronous Byzantine Fault Tolerance (aBFT) hashgraph consensus algorithm. It is governed by a collusion-resistant, decentralized council of leading enterprises, universities, and Web3 projects from around the world. Hedera is built differently from other blockchains. It has:

- **High throughput:** Hedera can process thousands of transactions per second (TPS) thanks to its asynchronous Byzantine Fault Tolerance (aBFT) consensus algorithm.
- **Fast Finality:** Transactions on Hedera achieve finality within seconds, ensuring that once a transaction is processed, it is irreversible.
- **Low and Predictable Fees:** Hedera is known for its minimal transaction fees, which are typically less than a cent. The fees are also predictable, meaning users can anticipate the costs involved in transaction processing without the risk of sudden fluctuations.
- **Scalability and Reliability:** Hedera's architecture is designed for scalability, allowing it to handle an increasing number of transactions as the network grows. It achieves this through its unique consensus algorithm.

Comparison of Hyperledger Fabric, Ethereum, and Hedera

Feature	**Hyperledger Fabric**	**Ethereum**	**Hedera**
Network Type	Permissioned, private	Public, decentralized	Public, decentralized
Consensus Mechanism	Pluggable (e.g., Raft, Kafka)	Proof of Stake (PoS)	Asynchronous Byzantine Fault Tolerance (aBFT)
Privacy and Security	Private channels, data collections	Public (with optional privacy features like zk-SNARKs)	High security, fair transaction ordering, private timestamps
Governance Model	Centralized (controlled by consortium)	Decentralized (community-driven)	Decentralized, governed by a council of enterprises, universities, and Web3 projects
Smart Contracts	Chaincode (supports Go, Java, Node.js)	Solidity (smart contract language)	Smart contracts (Solidity, but limited support)
Throughput	High (2000+ transactions per second)	Low (20 transactions per second)	High (up to 10,000 transactions per second)
Scalability	Modular, highly scalable for enterprise applications	Scalability improvements with Ethereum 2.0	Highly scalable with low latency and high throughput
Reliability	High (suitable for enterprise use)	Decentralized but subject to congestion and gas fees	High (due to leaderless consensus and aBFT)
Transaction Fees	Variable, depending on implementation	Gas fees (variable, subject to congestion)	Low, predictable fees (less than a cent)
Finality	Finality within a block (variable based on consensus)	Finality within blocks (but can be delayed)	Fast finality (seconds)

Table 2.1: Comparison between Hyperledger Fabric, Ethereum, and Hedera

2.1.1 Choosing the Right Tool

For a smart city implementing a nested blockchain structure, Hedera would be the most suitable choice for the underlying blockchain platform. Its high throughput, fast finality, and Byzantine Fault Tolerance (aBFT) ensure that the system can efficiently handle the vast number of transactions generated across multiple interconnected services in a smart city environment, such as transportation, energy management, and public safety. Hedera's scalability and low transaction fees make it well-suited for handling real-time data exchange and ensuring seamless interactions between different layers of the smart city infrastructure. Additionally, Hedera's decentralized governance model aligns well with the transparent and secure nature required for smart city applications, offering both reliability and speed. While Ethereum is also a strong contender for decentralized applications, especially for governance and public services, Hedera's performance and security make it a better fit for the overall blockchain backbone of a nested blockchain architecture, where multiple layers of services need to interact with minimal latency and maximum security. .

2.2 Designing a Blockchain Solution for Waste Management in a Smart City

Rapid population growth and urbanization have significantly accelerated waste generation, making effective waste management a critical challenge for smart cities. This growth intensifies issues such as illegal dumping, low recycling rates, and a lack of transparency in tracking waste disposal. Modern waste management strategies are increasingly adopting advanced technologies, including IoT, big data analytics, machine learning, and blockchain, which offers unique advantages. Blockchain ensures information security and integrity without requiring centralized guarantees and provides an immutable, transparent system to monitor waste from collection to recycling or disposal. It enables accountability among stakeholders through smart contracts, which can enforce penalties for illegal dumping or offer incentives for recycling. Token-based systems further encourage citizen participation, enhancing efficiency and reducing environmental harm. By integrating blockchain into waste management, smart cities can design innovative mechanisms that address global waste challenges and foster trust [3]

2.2.1 System Architecture

To illustrate the design and workflow of the blockchain-based waste management system, the architecture diagram provides an overview of key components, interactions, and processes.

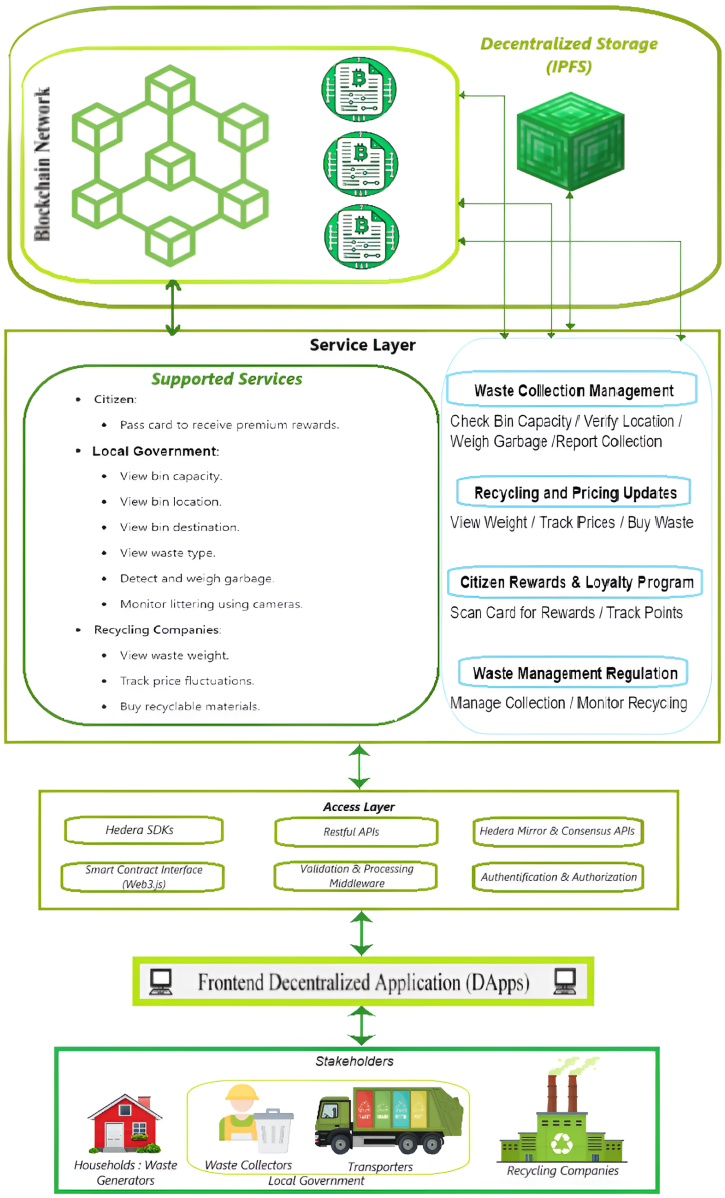

Figure 2.1: High-Level Architecture of the Blockchain-Based Waste Management System

2.2.2 Participants and Roles

- **Citizens (Households: Waste Generators)**
  Roles:
  - Generate waste.
  - Participate in reward programs.
  - Recycle.

- **Waste Collectors (Local Government)**
  Roles:
  - Collect waste.
  - Check bin capacity.
  - Verify garbage locations.
- **Transporters**
  Roles:
  - Transport waste from collection points to recycling or disposal facilities.
- **Recycling Companies**
  Roles:
  - View waste weight.
  - Track price fluctuations.
  - Purchase recyclable materials.
- **Local Government**
  Roles:
  - View bin data (capacity, location, destination, and waste type).
  - Detect and weigh garbage.
  - Monitor littering using cameras.
  - Enforce waste management regulations.

2.2.3 Assets

- **Waste Bins**
  Description: Physical containers for waste collection with data regarding bin capacity, location, waste type, and weight.
- **Waste Collection Data**
  Description: Data related to waste collection, including time, location, weight, and type of waste collected.
- **Recycling Materials**
  Description: Materials that can be recycled, including metals, plastics, and paper, tracked through weight and price data.

- **Citizen Rewards Data**
  Description: Data regarding citizen participation in reward programs, including points, rewards received, and actions like recycling.

- **Waste Management Regulations**
  Description: Regulations related to waste disposal, recycling guidelines, and penalties for illegal dumping or non-compliance.

- **Price Data**
  Description: Data tracking price fluctuations for recyclable materials, used by recycling companies for purchase decisions.

- **Littering Detection Data**
  Description: Data from cameras or sensors monitoring public spaces for littering, used by local governments for enforcement.

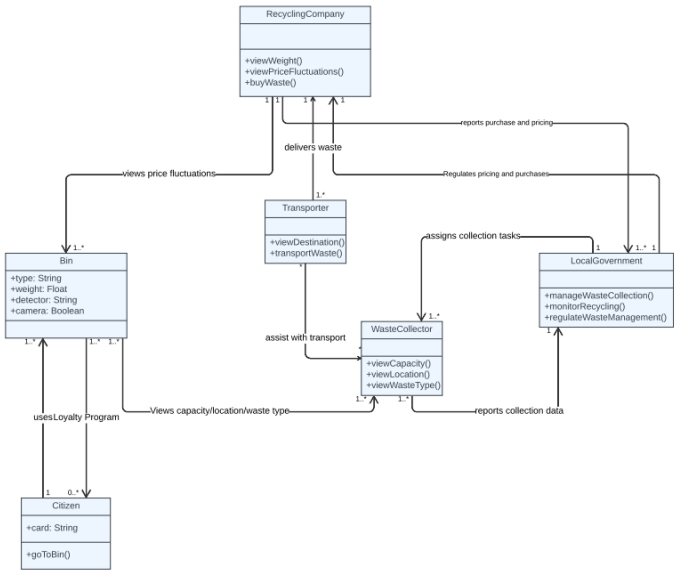

Figure 2.2: UML Diagram of Waste Management System

2.2.4 Smart Contracts for Waste Management and Recycling

The integration of Smart Contracts plays a vital role in automating waste management and recycling processes. Several Solidity smart contracts have been developed to support waste collection, recycling, and incentivization within the ecosystem. These contracts provide transparency, security, and efficiency by enforcing rules and tracking data on a decentralized blockchain. Below are key smart contracts and their functionalities:

Waste Collection and Bin Management

This contract manages waste bin data, including capacity, weight, and location, allowing citizens, governments, and recycling companies to monitor bin status and track waste collection efficiency.

Key Functions:

- `checkBinCapacity`: Returns available capacity in a bin.
- `weighGarbage`: Updates bin weight after waste is added.
- `reportCollection`: Marks a collection as complete.

```
function checkBinCapacity(string memory binID) public view returns (uint) {
    return bins[binID].capacity - bins[binID].currentWeight;
}

function weighGarbage(string memory binID, uint weight) public returns
    (bool) {
    require(bins[binID].capacity >= bins[binID].currentWeight + weight,
        "Bin capacity exceeded");
    bins[binID].currentWeight += weight;
    return true;
}

function reportCollection(string memory binID) public returns (bool) {
    bins[binID].isCollected = true;
    bins[binID].currentWeight = 0;
    return true;
}
```

Recycling Verification and Compliance

This contract ensures recycling companies comply with regulations by tracking and verifying waste collection activities, promoting transparency and adherence to environmental standards.

Key Functions:

- `manageCollection`: Allows logging of waste collection events.
- `verifyRecycling`: Verifies if recycling companies comply with regulations.
- `getWasteCollectionDetails`: Retrieves details about a specific waste collection.

Recycling Materials and Price Tracking

This contract rewards citizens for recycling by tracking waste disposal and awarding loyalty points that can be redeemed for rewards, fostering greater engagement in waste management.

Key Functions:

- `addMaterial`: Adds new recyclable materials.
- `updateMaterialPrice`: Adjusts prices for materials.
- `logRecycling`: Logs recycling contributions by users.

```
function manageCollection(string memory binID, bool status) public returns
    (bool) {
     bins[binID].isCollected = status;
     return true;
}

function verifyRecycling(bool compliance) public pure returns (bool) {
     require(compliance, "Recycling standards not met");
     return true;
}

function getWasteCollectionDetails(string memory binID) public view returns
    (bool isCollected, uint currentWeight, string memory location) {
     Bin memory bin = bins[binID];
     return (bin.isCollected, bin.currentWeight, bin.location);
}
```

Loyalty and Rewards Program

A loyalty program is integrated to incentivize citizens for their recycling efforts. By throwing recyclable waste, citizens earn loyalty points which they can redeem for rewards.

Key Functions:

- `logWasteThrown`: Logs waste thrown by a citizen, earning loyalty points.
- `redeemLoyaltyPoints`: Allows citizens to redeem accumulated points.

```
function logWasteThrown(address _citizen, uint256 _materialId, uint256
    _weight) public {
     require(_materialId < materialCount, "Invalid material ID");
     require(_weight > 0, "Weight must be greater than zero");

     // Track the waste thrown by the citizen
```

```
    citizens[_citizen].totalWasteThrown += _weight;

    // Calculate loyalty points: 10 points per 1kg of waste thrown
    uint256 pointsEarned = _weight * 10;
    citizens[_citizen].loyaltyPoints += pointsEarned;

    emit WasteThrownLogged(_citizen, _materialId, _weight);
    emit LoyaltyPointsUpdated(_citizen, citizens[_citizen].loyaltyPoints);
}

function redeemLoyaltyPoints(address _citizen, uint256 _points) public {
    require(citizens[_citizen].loyaltyPoints >= _points, "Not enough
        loyalty points");
    citizens[_citizen].loyaltyPoints -= _points;
    emit LoyaltyPointsUpdated(_citizen, citizens[_citizen].loyaltyPoints);
}
```

Waste Management Regulation

This contract manages waste collection activities, including tracking the total waste collected, the amount of waste recycled, and verifying recycling processes. It allows governments or regulatory authorities to monitor the collection status and ensure that the recycling company complies with regulations. The contract ensures transparency by logging waste collection events and the recycling process for verification.

Key Functions:

- `manageCollection`: Logs a waste collection, including the amount of waste collected, the amount recycled, the collection location, and the recycling company.
- `verifyRecycling`: Verifies whether the recycling company complied with the recycling regulations.
- `getWasteCollectionDetails`: Retrieves the details of a specific waste collection by collection ID.
- `getTotalWasteCollections`: Returns the total number of logged waste collections.

```

    function manageCollection(
        string memory _location,
        uint256 _totalWasteCollected,
        uint256 _totalWasteRecycled,
        address _recyclingCompany
    ) public onlyOwner {
```

```
8           require(_recyclingCompany != address(0), "Invalid recycling
                company address");
9           wasteCollections[wasteCollectionCount] = WasteCollection(
10              block.timestamp,
11              _location,
12              _totalWasteCollected,
13              _totalWasteRecycled,
14              _recyclingCompany,
15              false
16          );
17          emit WasteCollectionLogged(
18              wasteCollectionCount,
19              block.timestamp,
20              _location,
21              _totalWasteCollected,
22              _totalWasteRecycled,
23              _recyclingCompany
24          );
25          wasteCollectionCount++;
26      }
27
28      function verifyRecycling(uint256 _collectionId, bool
            _recyclingVerified) public onlyOwner {
29          require(_collectionId < wasteCollectionCount, "Invalid
                collection ID");
30          WasteCollection storage collection =
                wasteCollections[_collectionId];
31          collection.recyclingVerified = _recyclingVerified;
32          emit RecyclingVerificationCompleted(_collectionId,
                _recyclingVerified);
33      }
34
35      function getWasteCollectionDetails(uint256 _collectionId) public
            view returns (
36          uint256, string memory, uint256, uint256, address, bool
37      ) {
38          require(_collectionId < wasteCollectionCount, "Invalid
                collection ID");
39          WasteCollection storage collection =
                wasteCollections[_collectionId];
40          return (
41              collection.collectionDate,
42              collection.collectionLocation,
43              collection.totalWasteCollected,
44              collection.totalWasteRecycled,
45              collection.recyclingCompany,
```

```
46                 collection.recyclingVerified
47             );
48         }
49
50         function getTotalWasteCollections() public view returns (uint256) {
51             return wasteCollectionCount;
52         }
53     }
```

Listing 2.1: WasteManagementRegulation Contract

These smart contracts not only help streamline the waste collection and recycling process but also encourage greater citizen participation by offering rewards. Additionally, the use of blockchain ensures that all data, including waste collection reports and recycling contributions, is immutable and transparent, fostering trust among all stakeholders.

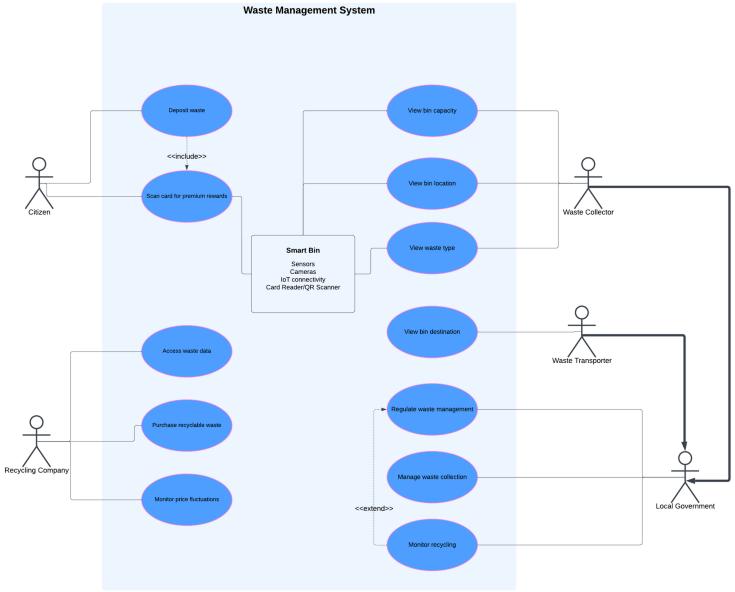

Figure 2.3: Use Case Diagram Illustrating Key Interactions in the Waste Management System

2.3 Development and Integration of The Solution with Hedera Hashgraph

The development phase was centered around transforming the system design into a fully functional blockchain application. This involved testing, deploying the application, and integrating it with the Hedera Hashgraph network for enhanced decentralization and scalability.

2.3.1 Hedera Integration

Hedera Hashgraph: Integrated the application with Hedera Hashgraph to leverage its decentralized consensus and distributed ledger for improved scalability, security, and low transaction costs. Hedera Hashgraph was used to manage accounts and provide decentralized consensus.

2.3.2 Hardhat for Contract Deployment:

Hardhat was used for smart contract testing and deployment. It provided an efficient environment for compiling, deploying, and verifying smart contracts on the Ethereum Virtual Machine (EVM). This ensured the contracts were functional and ready for interaction with the Hedera network.

2.3.3 API Development and Testing:

To facilitate communication with the blockchain, **Node.js** was used to create API requests that interacted with the deployed contracts on Hedera. **Bruno** was employed to test these API endpoints, ensuring their reliability and performance before full deployment. Additionally, **HashScan** was utilized to enable real-time tracking of transactions, monitoring account activity, and analyzing token performance. This integration ensured comprehensive visibility into blockchain operations, enhancing transparency and operational efficiency.

The following screenshots showcase API testing results from BRUNO, highlighting the endpoints' performance, reliability, and interaction with the deployed smart contracts.

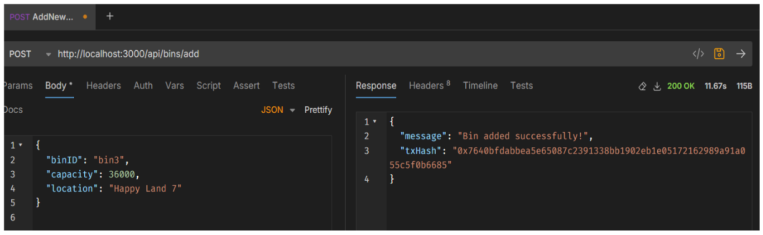

Figure 2.4: AddNewBin API Endpoint for Bin Management

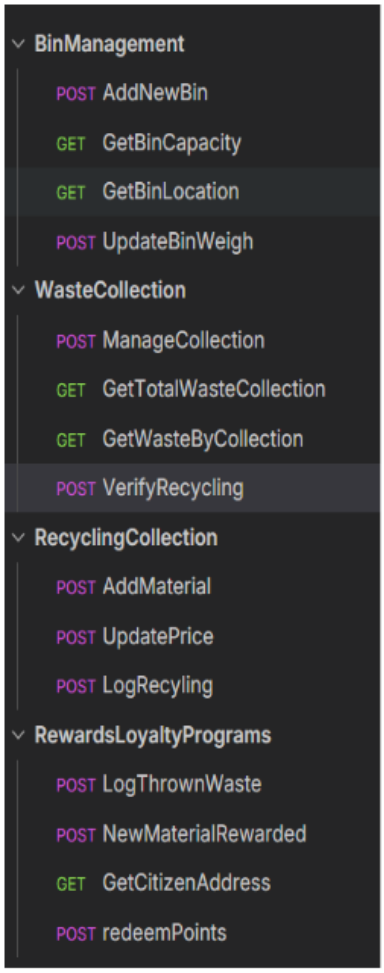

Figure 2.5: API Endpoints Overview from BRUNO for Waste Management Blockchain

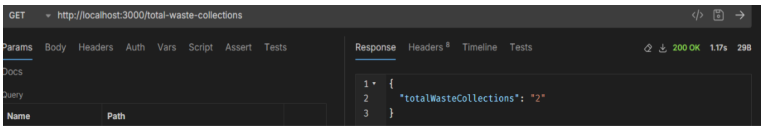

Figure 2.6: GetTotalWasteCollection API for Waste Collection

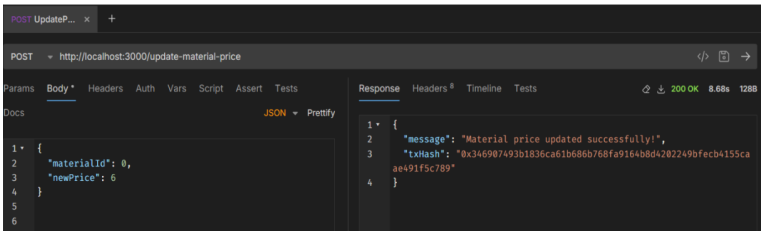

Figure 2.7: Update Price API for Recycling Collection

2.3.4 Frontend Application (React DApp):

Once the APIs were developed and tested, they were integrated with the frontend application, which was built using **React**. This decentralized application (DApp) enabled users to interact with the smart contracts via the API endpoints, providing a seamless interface for real-time interactions with the blockchain.

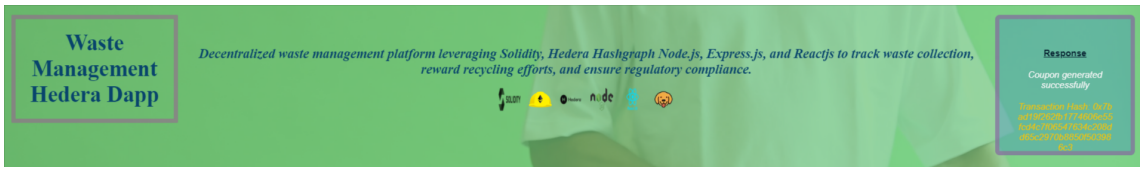

Figure 2.8: React-Based Frontend: Seamless Interaction with Hedera Blockchain for Waste Management.

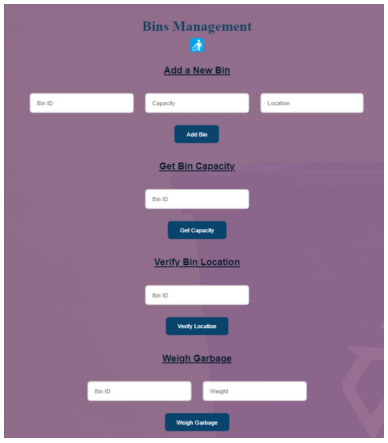

Figure 2.9: React-Based Frontend: Seamless Interaction with Hedera Blockchain for Waste Management.

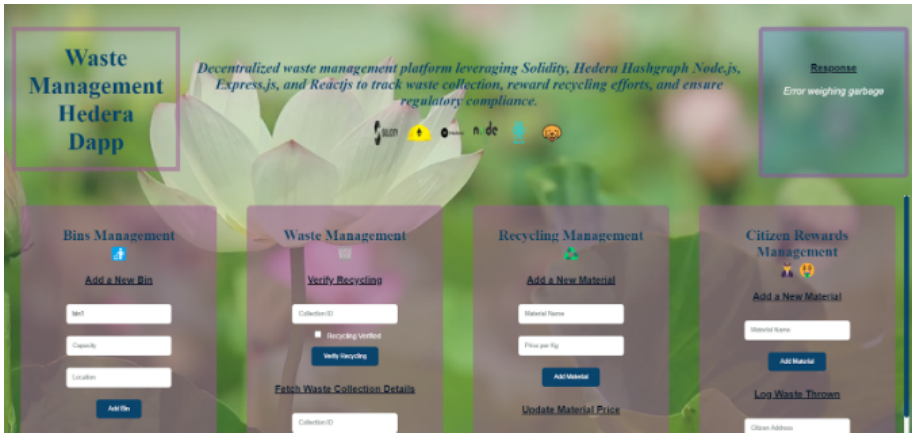

Figure 2.10: React-Based Frontend: Seamless Interaction with Hedera Blockchain for Waste Management.

Chapter 3

Blockchain Evaluation

3.1 Defining the KPIs

3.1.1 Definition of Key Performance Indicators

A Key Performance Indicator (KPI) is a measurable value that demonstrates how effectively an individual, team, or system is achieving a business objective. In the context of blockchain, KPIs are used to evaluate the performance, efficiency, security, and scalability of the blockchain network and its applications. These indicators provide insights into the success of blockchain implementation and help in decision-making processes for optimization and improvements.

KPI in Smart City Blockchain In blockchain-based systems, KPIs are essential for assessing the network's overall health and performance, particularly in complex applications such as waste management or supply chain financing. These metrics offer a comprehensive overview of critical aspects such as transaction throughput, data integrity, energy efficiency, and security. By monitoring these KPIs, stakeholders can ensure that the blockchain platform operates optimally, supports high levels of transparency, and maintains a secure, scalable environment. Understanding and analyzing these performance indicators allow for informed decisions about system improvements, risk management, and resource allocation, driving continuous innovation and operational excellence.

3.1.2 System Performance

- **Transaction Throughput (TPS):** Number of transactions processed per second, indicating the blockchain's ability to handle large volumes of transactions.
- **Latency:** Only authorized members can access the network.
- **Scalability:** Ability to handle an increasing number of users or transactions, ensuring that the blockchain can grow with demand.

- **Block Time:** Average time it takes for a new block to be added to the blockchain, impacting transaction confirmation times.

3.1.3 Data Integrity & Security

- **Data Immutability:** Ensures that once data is written to the blockchain, it cannot be altered, protecting the integrity of the data.
- **Error Rate:** Percentage of failed transactions or smart contract executions, reflecting the reliability of the system.
- **Security Breaches:** Number of attempted or successful breaches, indicating the vulnerability of the blockchain.
- **Hash Rate:** The computational power dedicated to securing the blockchain; a higher hash rate enhances the network's security.

3.1.4 Operational Efficiency

- **Energy Consumption:** The energy required to operate the blockchain infrastructure, important for sustainability considerations.
- **Network Fees:** The costs paid by users to process transactions, which can indicate network congestion or demand for processing capacity.
- **Transaction Volume:** Total number of transactions processed over a period, indicating how active the blockchain network is.
- **Transaction Value:** The total value of assets moved across the blockchain, which shows the economic utility of the network.

3.1.5 Smart City Integration

- **Interoperability:** The ability to integrate with other smart city services such as IoT devices and surveillance cameras, enabling broader system coordination.
- **Real-Time Monitoring:** Utilization of real-time data for decision-making and enforcement in waste management operations and other smart city services.

3.1.6 Transaction Metrics

- **Active Addresses:** The number of unique addresses involved in transactions, indicating the engagement level and user participation in the network.

- **Total Addresses:** The overall number of unique addresses on the blockchain, reflecting the network's adoption and growth.
- **Decentralized Transactions per Second (DTPS):** Transaction speed combined with the level of decentralization, offering insights into both efficiency and fairness of the blockchain's consensus mechanism.
- **Network Value to Transactions (NVT):** Ratio of the market capitalization of the blockchain to its transaction volume, providing insight into the blockchain's value relative to its transaction activity.

3.1.7 Network Health Metrics

- **Block Size:** The average size of blocks in the blockchain, with larger blocks capable of handling more transactions but requiring more storage and processing.
- **Nakamoto Coefficient:** Measures the number of entities that would need to collude to disrupt the blockchain's consensus mechanism, with a higher number indicating greater decentralization and resilience.

3.1.8 Decentralization Metrics

- **Gini Coefficient:** Measures inequality in resource distribution within the blockchain network, with a lower score indicating more equitable distribution of tokens or resources.
- **Nakamoto Coefficient:** As mentioned, it measures the decentralization of the network's control. More participants are needed to collude, the higher the coefficient.

3.1.9 Transparency & Risk Management

- **Transparency:** Blockchain provides visibility and tracking of information/documents, reducing the risk of fraud and credit risk by ensuring that all transactions are visible and traceable by all parties.
- **Risk Transfer:** Blockchain reduces the credit risk traditionally associated with transactions, especially for SMEs, by providing a reliable and transparent record of transactions.
- **Traceability:** Blockchain ensures that all information or documents are traceable from one participant to another, enhancing security and reducing fraud.
- **Fast Response:** Blockchain's ability to facilitate near-instantaneous execution of processes like invoicing and payments through smart contracts, improving efficiency and unlocking working capital faster.

3.2 Evaluation Study

Link to Hedera Hash Scan Dashboard.

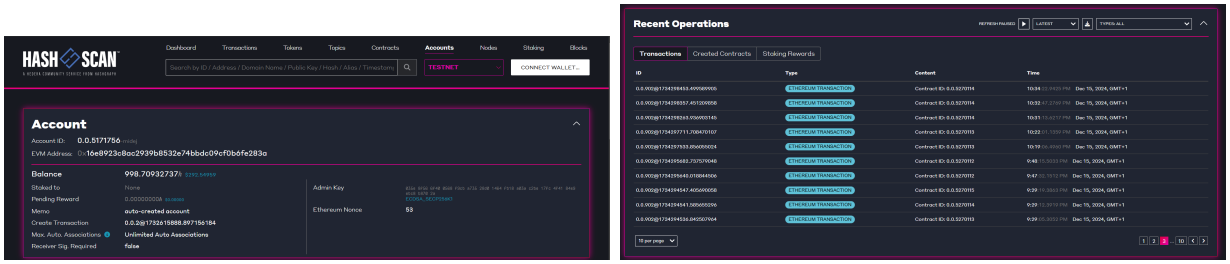

Figure 3.1: Hash scan dashboard

Additionally, we loaded data from HashScan using Python scripts and various libraries such as Pandas and Matplotlib. The data extraction process involved API calls to HashScan, transforming the data into structured formats suitable for analysis. Using these libraries, we generated insightful graphs and visualizations, enabling a deeper understanding of data trends and supporting decision-making processes.

You can find below a snapshot of our python code:

```
# Import necessary libraries
import pandas as pd

# Load and inspect the dataset
file_path = '../transactions.csv'
data = pd.read_csv(file_path)
# Load CSV data
df = pd.read_csv('hashscan.csv')


# Display the first few rows and dataset info
data.head(), data.info()

import matplotlib.pyplot as plt

# Calculate total gas used per transaction
data['TotalGasUsed'] = data['GasLimit'] * data['MaxFeePerGas']

# Aggregate data for analysis
total_gas_by_sender =
    data.groupby('From')['TotalGasUsed'].sum().sort_values(ascending=False)
average_gas_fee = data['MaxFeePerGas'].mean()
```

Below are some of the generated graphs:

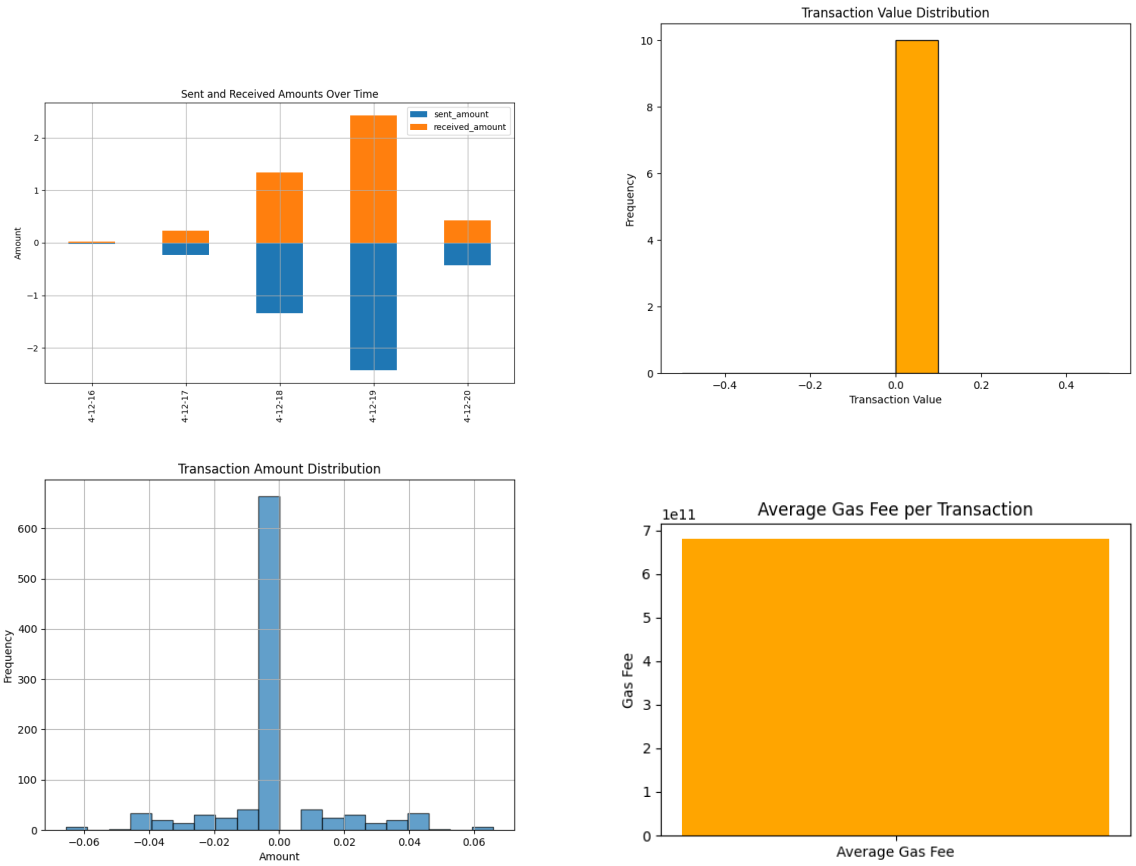

Figure 3.2: Generated Data Visualizations

3.3 Challenges and Limitations

Challenges:

Contract Linking: One of the main challenges in implementing smart contracts for waste management is ensuring proper contract linking. Waste collection and recycling processes need to be accurately recorded and connected across multiple entities such as the government, recycling companies, and citizens. This requires secure and transparent communication between these parties and seamless integration of multiple smart contracts.

Limitations:

Lack of Libraries for Visualization: Currently, there are limited libraries and tools available to graph transactions or visualize data on the blockchain for real-time monitoring and reporting. This makes it difficult to present waste collection and recycling data in an easily understandable graphical format for stakeholders.

Absence of Readily Available Software: Another limitation is the absence of comprehensive, ready-made software solutions for visualizing or analyzing transactions data.

Chapter 4

Nested Blockchain Solution

4.1 Nested Blockchain Design

4.1.1 Main Blockchains:

Purpose: The Waste Management blockchains acts as the central hub, orchestrating all key functions for waste collection, recycling compliance, material pricing, and user incentives.

Key Features:

- **Bin Management:** Add, monitor, and manage bins (e.g., location, weight, capacity). Trigger notifications for full bins and report collections.
- **Material Pricing:** Track prices for recyclable materials dynamically. Ensure transparency in reward calculation based on material contributions.
- **Citizen Rewards Integration:** Forward citizen recycling data for loyalty points calculation and pass this data to the nested coupons generation blockchain to generate coupons.
- **Waste Collection Management:** Monitor compliance with recycling regulations. Enable detailed tracking of waste streams.

4.1.2 Nested Blockchain: CouponsGenerator

Purpose: The **Coupon Generator** blockchain is dedicated to incentivizing citizens through loyalty points based on their recycling activities. It processes individual user contributions and seamlessly integrates with the main blockchain to provide a cohesive reward system.

Key Features:

- **Recyclable Material Management:** Materials are stored with a unique ID and name for identification and tracking. Only the contract owner (main blockchain or admin) can add new materials.

- **Citizen Recycling Contribution Tracking:** Tracks the total weight of waste thrown by each citizen. Records loyalty points earned based on contributions.
- **Loyalty Points System:** Citizens earn 10 points per kilogram of waste recycled. Points are cumulative and can be redeemed for rewards through the blockchain.
- **Event Logging:** Emits events to ensure transparency and traceability of key actions:
  - **MaterialAdded:** When a new recyclable material is introduced.
  - **WasteThrownLogged:** When a citizen logs their waste contributions.
  - **LoyaltyPointsUpdated:** Whenever loyalty points are earned or redeemed.
- **Ownership Restriction:** The `onlyOwner` modifier ensures that critical functions, such as adding materials, are restricted to authorized users (e.g., admin or main blockchain).
- **Reward Redemption:** Citizens can redeem accumulated points for rewards. Redemption logic can be expanded to include token transfers, discounts, or partnerships with third-party vendors.

4.1.3 Integration of Main and Nested Blockchains

The **Main Blockchain** (Rewards_LoyaltyPrograms) and **Nested Blockchain** (CouponsGenerator) work together to deliver an efficient waste management system:

- The main blockchain gathers data from various sources, such as bin statuses, recycling compliance, and material pricing, and forwards relevant information to the nested blockchain.
- The nested blockchain processes this data to calculate loyalty points for citizens, keeping a record of their contributions.
- Both blockchains share data seamlessly, enabling a transparent, traceable, and incentivized waste management ecosystem.

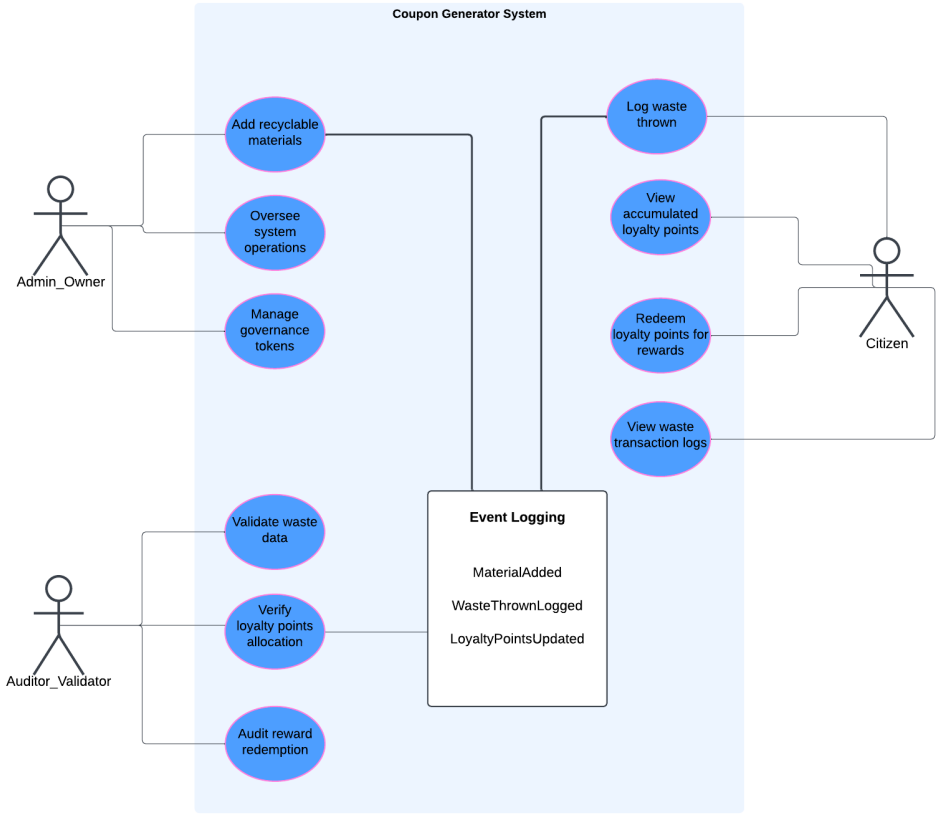

Figure 4.1: Coupon Generator System: Use-Case Diagram for Recycling Incentives

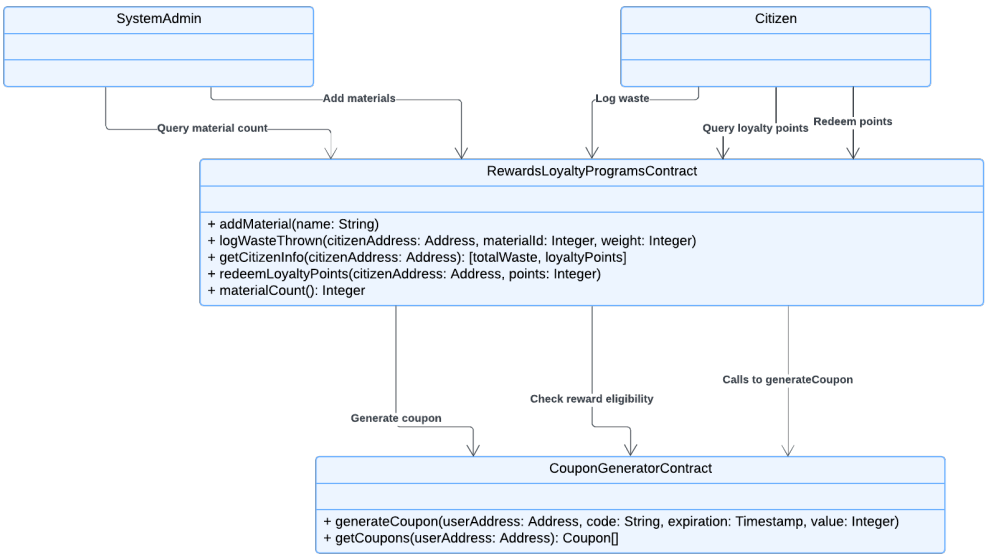

Figure 4.2: Coupon Generator System: Interaction UML Diagram

4.1.4 Updated Solidity Contract

Solidity Code:

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

contract Rewards_LoyaltyPrograms {
    struct Material {
        string name;
    }

    struct Citizen {
        uint256 totalWasteThrown;
        uint256 loyaltyPoints;
    }

    mapping(uint256 => Material) public materials;
    uint256 public materialCount;

    mapping(address => Citizen) public citizens;

    event MaterialAdded(uint256 indexed materialId, string name);
    event WasteThrownLogged(address indexed user, uint256 materialId,
       uint256 weight);
    event LoyaltyPointsUpdated(address indexed user, uint256 newPoints);

    address public owner;

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the owner can perform this
           action");
        _;
    }

    constructor() {
        owner = msg.sender;
    }

    function addMaterial(string memory _name) public onlyOwner {
        materials[materialCount] = Material(_name);
        emit MaterialAdded(materialCount, _name);
        materialCount++;
    }

    function logWasteThrown(address _citizen, uint256 _materialId, uint256
       _weight) public {
```

```
41          require(_materialId < materialCount, "Invalid material ID");
42          require(_weight > 0, "Weight must be greater than zero");
43
44          citizens[_citizen].totalWasteThrown += _weight;
45          uint256 pointsEarned = _weight * 10;
46          citizens[_citizen].loyaltyPoints += pointsEarned;
47
48          emit WasteThrownLogged(_citizen, _materialId, _weight);
49          emit LoyaltyPointsUpdated(_citizen,
                citizens[_citizen].loyaltyPoints);
50      }
51
52      function getCitizenInfo(address _citizen) public view returns (uint256,
            uint256) {
53          return (citizens[_citizen].totalWasteThrown,
                citizens[_citizen].loyaltyPoints);
54      }
55
56      function redeemLoyaltyPoints(address _citizen, uint256 _points) public {
57          require(citizens[_citizen].loyaltyPoints >= _points, "Not enough
                loyalty points");
58          citizens[_citizen].loyaltyPoints -= _points;
59
60          emit LoyaltyPointsUpdated(_citizen,
                citizens[_citizen].loyaltyPoints);
61      }
62  }
```

Listing 4.1: Main and Nested Blockchain Integration

4.1.5 Evaluation of Nested Blockchain

The bar chart displays the top receivers by gas consumption, showing the total gas used by four different entities. The entity labeled "58f1c99A5" consumed the most gas, followed by "a524AfC4," "7a817b4f3," and finally "50B9cc26," which consumed the least.

The line graph illustrates the number of transactions occurring over a period of several days, from December 16th to December 20th. The graph shows a general upward trend until a peak around December 19th, followed by a sharp decline by December 20th.

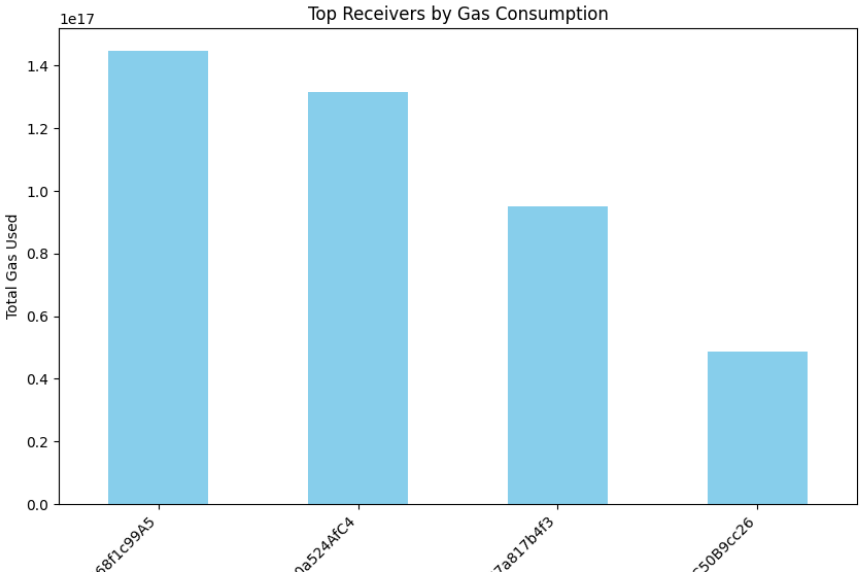

Figure 4.3: Top Gas Consumers

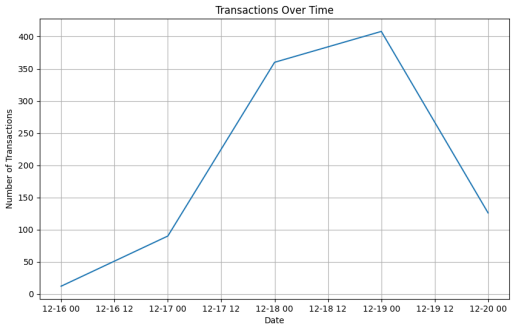

Figure 4.4: Transaction Volume Over Time

Conclusion

In conclusion, the design, development, and deployment of our smart city blockchain solution for waste management demonstrate the feasibility and efficiency of leveraging blockchain technology to address urban sustainability challenges. The nested blockchain architecture ensures data integrity, transparency, and secure transactions while enabling scalable and decentralized management of waste-related activities. Through the deployment of our waste management

scenario, we observed that the incentivization model, which rewards recycling efforts with loyalty points, not only promotes environmentally responsible behavior but also fosters trust and collaboration among stakeholders. This dynamic ecosystem facilitates active citizen participation, making environmental conservation a shared responsibility.

Future Perspectives: Looking ahead, future developments could explore integrating advanced technologies such as IoT sensors and AI-driven analytics to enhance real-time monitoring, predictive maintenance, and data-driven decision-making. Expanding the blockchain framework to include multi-stakeholder collaboration platforms could further streamline communication and coordination among municipal services, businesses, and residents. Additionally, scaling the solution to other urban domains, such as energy management and transportation, holds significant potential for fostering smarter, more sustainable cities.

Bibliography

[1] Hypertrader Dictionary. *Federated Blockchain*. Accessed: 2024-12-05. n.d. URL: https://gethypertrader.com/dictionary/federated-blockchain.

[2] HYPERLEDGER. *Whitepaper Introduction Hyperledger*. https://www.hyperledger.org/wp-content/uploads/2018/07/HL_Whitepaper_IntroductiontoHyperledger.pdf. Accessed: 2024-11-30. 2018.

[3] Peng Jiang et al. "Blockchain technology applications in waste management: Overview, challenges and opportunities". In: *Journal of Cleaner Production* 421 (2023), p. 138466. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2023.138466. URL: https://www.sciencedirect.com/science/article/pii/S0959652623026240.