CMPE 492

Blockchain Based Token Rental System

Serdar Akol 2017400126

Elif Nur Akalın 2018400312

Advisor:

Can Özturan

TABLE OF CONTENTS

1. INTRODUCTION 1
1.1. Broad Impact 1
1.2. Ethical Considerations 1
2. PROJECT DEFINITION AND PLANNING 3
2.1. Project Definition 3
2.2. Project Planning 3
2.2.1. Project Time and Resource Estimation 3
2.2.2. Success Criteria 4
2.2.3. Risk Analysis 5
2.2.4. Team Work 5
3. RELATED WORK 6
4. REQUIREMENTS SPECIFICATION 8
5. DESIGN 10
5.1. Information Structure 10
5.2. Information Flow 11
5.3. System Design 12
5.4. User Interface Design 12
6. IMPLEMENTATION AND TESTING 15
6.1. Implementation 15
6.2. Testing 16
6.3. Deployment 17
7. RESULTS 19
8. CONCLUSION 20
REFERENCES 21

1. INTRODUCTION

1.1. Broad Impact

Rentable contract systems are one of the most promising uses of blockchain technology. It may start a new era in several markets such as real estate and automotive along with this it is also applicable on many domains, reserving a hotel room, equipment rental, machinery rental or tool rental.

Rentable contact systems provide a decentralized platform, like other blockchain technologies, that enables users to rent their assets in a transparent and secure way. With the rules and conditions already determined in the rentable contact systems, the agreements and verification procedures get automated. It may allow peer-to-peer car renting, house renting etc and reduce the need for car or house rental companies. With this way, users may rent their assets without the need of a third party. This increases the utilization of assests. It also puts another dimension to consumer behaviors having assets and renting them, since renting gets feasible to a great extent.

In this graduation project we wanted to develop a rentable blockchain contract system prototype, that shows the potential of this technology and its impact on various industries.

1.2. Ethical Considerations

As in any technology, the most potential concern is privacy, like personal data, payment, history and rental history. The security measures to protect user data are well considered and in place while developing our project.

Another ethical consideration is the potential impact on traditional rental companies and their employees. The rentable contract system projects may result in job losses in

such industries. This gives another task of research of such negative outcomes and ways to reduce them. This technology should be conducted in a socially responsible manner.

In the environmental aspect, this project may lead to increased use of shared assets and may reduce the demand for new products, since it increases the utilization. This mitigates the human and technological impact on the environment. But on the other hand the energy demands of blockchain projects are also usually high. These two aspects should be weighted carefully. Our project should be conducted in an environmentally responsible manner.

2. PROJECT DEFINITION AND PLANNING

2.1. Project Definition

Our graduation project aims to develop a rentable blockchain contract system that utilizes ERC-721 tokens to enable renting of various assets. The potential impact of this technology on various industries such as real estate, hospitality services, automotive, and equipment rental will be explored.

The one bottleneck of this project is developing an efficient algorithm for the problem of intersecting time intervals i.e. requesting to rent an asset while it is rented for the same time interval or some portion of the same time interval. This algorithm will prevent possible conflicts. We are committed to develop an effective solution that meets any needs of our users.

While providing a platform for a peer-to-peer rental contract system and eliminating the need for the third party rental companies, our project aims to consider all ethical considerations. Sensitive data will remain secure.

By using the blockchain technology and ERC-721, we believe that our project may have a huge impact on the rental industry and benefit users from a lot of domains, since it can utilize any kind of asset, while ensuring security.

2.2. Project Planning

2.2.1. Project Time and Resource Estimation

We planned a comprehensive yet not too complex project, therefore we have estimated that the development time is 4 months for 2 developers, taking phases into account such as the literature survey, architecture, backend development, frontend

development, testing and reporting.

As we finalize our project, we can establish that our progress aligns with our initial project planning. We implemented a blockchain based smart contract and we developed a web page for members to interact with the smart contract. Additionally, we integrated with Metamask to let user interaction with the smart contract be smooth.

2.2.2. Success Criteria

We have determined the following success criteria:

- Security: Our system must be secure, and should in no way allow attacks such as stealing the tokens in the process of renting, or revealing sensitive information.
- Scalability: Our system must be able to scale to millions of users, and perform quickly under high volumes of demand.
- Satisfaction: Our system must satisfy users in a way that it must be user friendly and cost-effective so that the customers keep using the platform.

Results on success criteria:

- Security: Our system is secure and in every steps the authorization on the actions are checked logically.
- Scalibility: We focused on delivering a product that meets the requirements and customer needs as much as we can in a limited time period. We planned implementing more efficient data structures and algorithms infuture, they are not met yet.
- Satisfaction: Our system meets most of the needs of the customers but we are also aware of the need of a better user experience on our front-end and on the interface.

2.2.3. Risk Analysis

There are some potential risks including the following:

- Security risks: We must implement a system that is impenetrable against attacks. Failures would cause lawsuits and damage in reputation.
- Legal risks: We must implement rental contracts that are in accordance with local laws.

2.2.4. Team Work

Our team consists of two developers competent in different aspects of the project. While Elif Akalın is more experienced in developing blockchain based contracts, Serdar Akol's experience is more focused on project planning and frontend development. Therefore we form a suitable team that is capable of successfully concluding the project in due time.

3. RELATED WORK

The intersecting interval problem is kind of a bottleneck for our project. It deals with the problem of multiple users requesting to rent a token for the same time intervals or overlapping time intervals. This could occur in our user cases frequently, since there is no central authority to resolve the conflict.

Implementing an efficient algorithm for managing intersecting time intervals is necessary and would solve the problem mentioned above. This is also necessary for the trust for our project.

As the name also refers, the problem is overlapping intervals. It has various applications such as scheduling, resources allocation, database management etc. Common approaches to detect the overlap include the data structures such as binary search tree, red black trees or interval trees graphs which can store the intervals and detect the overlaps efficiently. The algorithms to find overlap involve brute-force search, sweep line algorithm, divide and conquer algorithm and so on.

This problem also rises in blockchain projects frequently. There are researches regarding this problem specific to blockchain based projects such as off-chain verification, side chains and layer-2 solutions.

In blockchain-based projects this problem is solved commonly by combination of both traditional solutions like red black tree and blockchain specific solutions like off-chain protocols, smart contracts and decentralized file storage systems.

Red-black tree is a self-balancing binary search tree that helps keep the tree height as low as possible, which leads to faster lookup times. Root and leaves are black and if a node is red then its children are black. Every path from a given node to its leaf descendants contains the same number of black nodes. Every node is either red or

black. Insertion and deletion causes the tree to be reconstructed while obeying the rules of red black tree.

Off-chain algorithm is used commonly with red-black trees. They are not executed on the chain itself, but rather on a different server. On the other hand, smart contract algorithm executes the checks of interval intersection on the blockchain.

Off-chain algorithms are fast, cost efficient and flexible but there is centralization of the data, it lacks the security and transparency. Smart chain algorithms are slow and cost expensive but they are more secure, transparent and decentralized.

We will implement our project in a smart contract manner. On smart contract systems, the intervals are divided into subintervals recursively and subintervals are checked for the intersection. Here, we will try to use and apply the ideas in the back of red black trees. [1], [2], [3], [4], [5].

4. REQUIREMENTS SPECIFICATION

The system must meet the following functionality:

- All standard ERC-721 functions should be provided.
- Anyone should be able to rent their tokens.
- The contract should provide a way to temporarily transfer the ownership of the tokens.
- The contract should provide a way to do exchanges of the tokens, therefore permanently transferring the ownership of the tokens.
- The contract should override some of the standard ERC-721 functions, such as transfer functions.
- The contract should perform feasibly under the load of users on the scale of millions.
- The contract should implement a data structure that has low complexity for search, add, edit, delete and detect collision operations to hold the intervals of time of rental periods.
- The contract functions should be efficient so that the gas fees of the functions are reasonable.
- The contract should support different kinds of rental periods such as minutes, hours, days, months etc.

The implemented requirements on the smart contract:

- All standard ERC-721 functions should be provided.
- Anyone should be able to rent their tokens.
- The contract should provide a way to temporarily transfer the ownership of the tokens. This is implemented in the way of creating a rental contract that gives legal rights to use of the asset.
- The contract should provide a way to do exchanges of the tokens, therefore per-

manently transferring the ownership of the tokens. Already implemented in ERC-721.

- The contract should perform feasibly under the load of users on the scale of millions.
- The contract should support different kinds of rental periods such as minutes, hours, days, months etc.

5. DESIGN

5.1. Information Structure

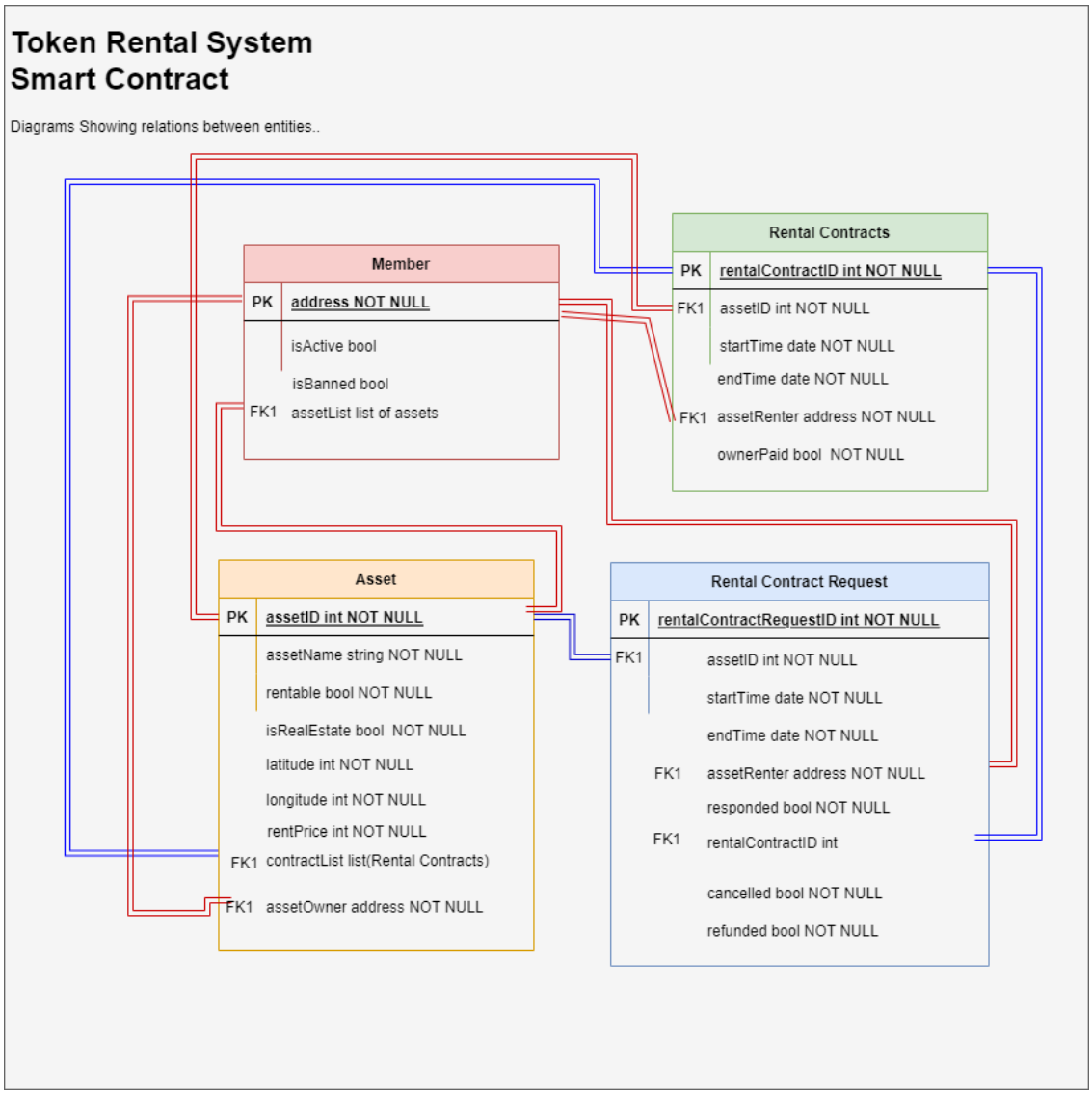

Figure 5.1. ER Diagram

5.2. Information Flow

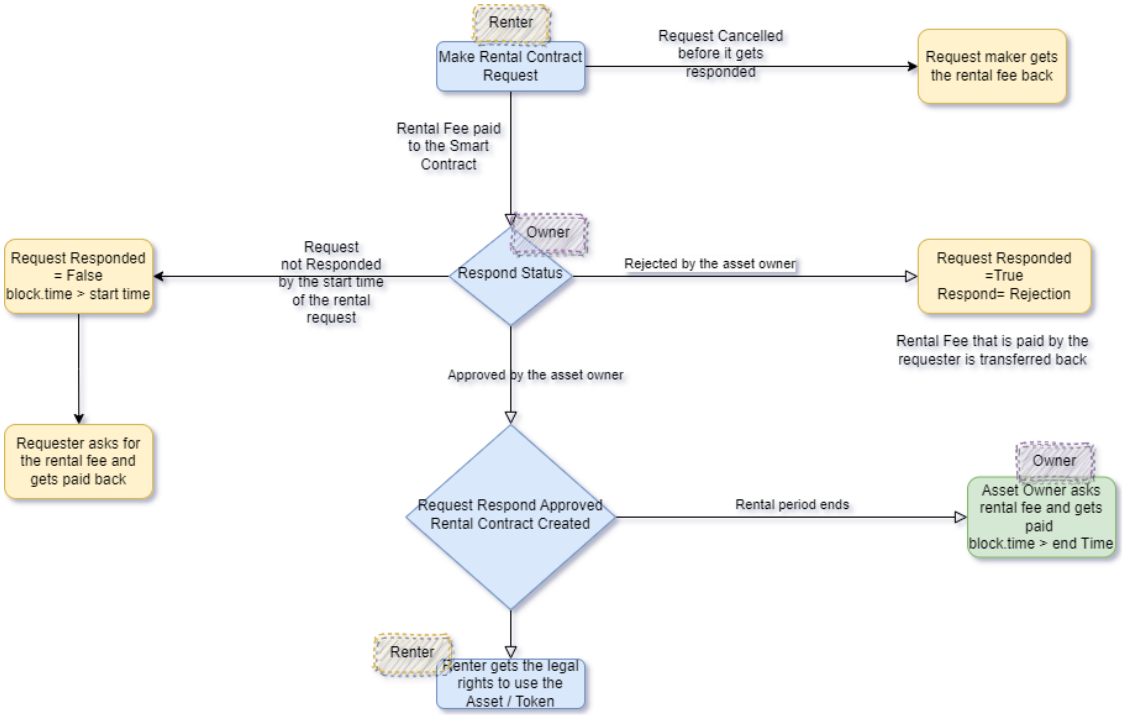

Figure 5.2. Workflow of a rental contract agreement

5.3. System Design

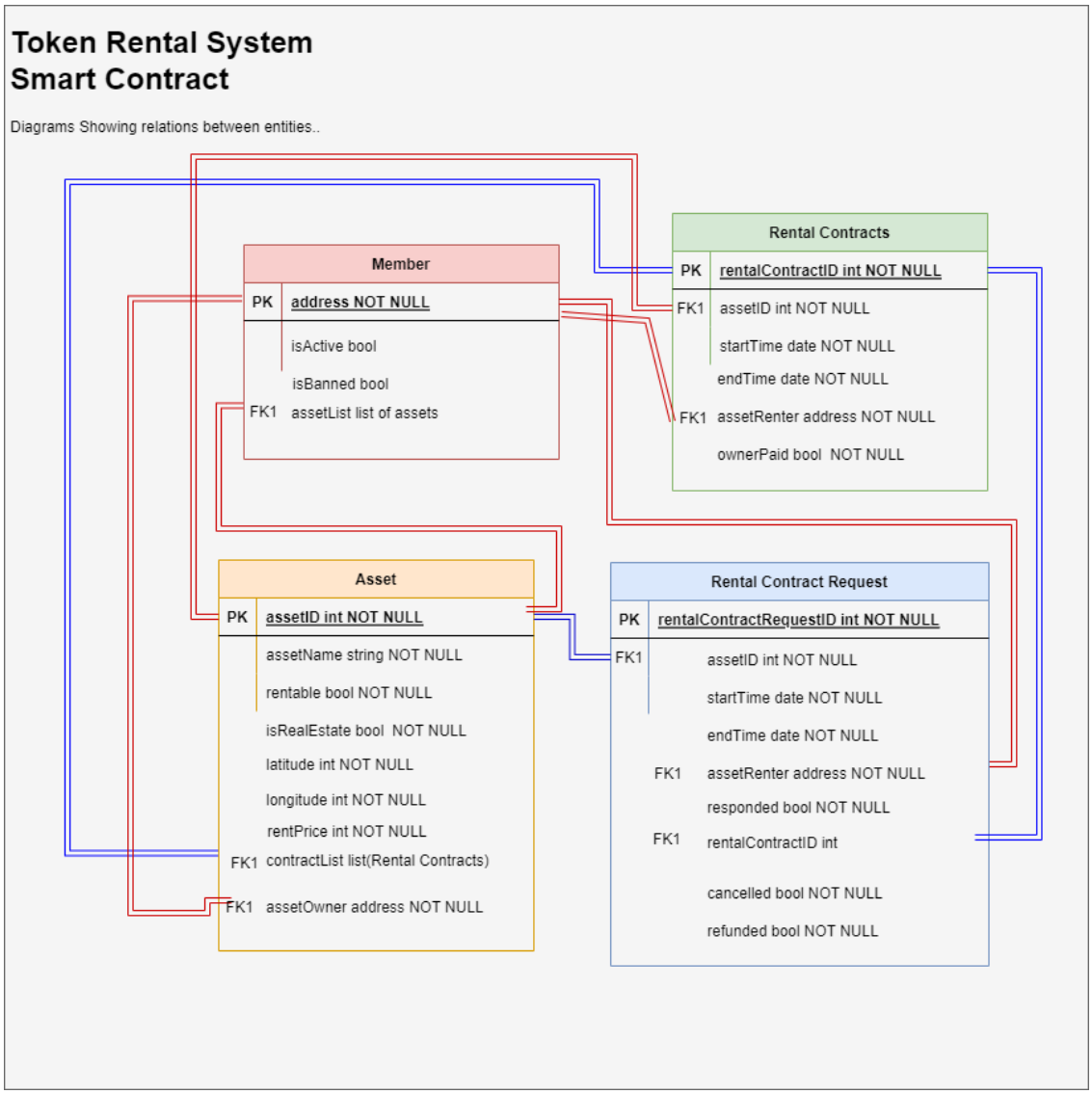

Figure 5.3. ER Diagram

5.4. User Interface Design

We were planning to implement a web application that is visually appealing and user friendly. We will connect the deployed contract via web3.js framework and build a platform that supports actions like account creation and login, dashboard displaying all rental contracts and their status, option to create a new rental contract, payment processing and transaction history, user profile management.

As per changes to the development plan, we decided to integrate ethers.js framework instead of web3.js. We developed the web application via React, prioritizing functionality rather than visual aesthetics. While the web application functions with the necessary features, we recognize the need for improvements in UX.

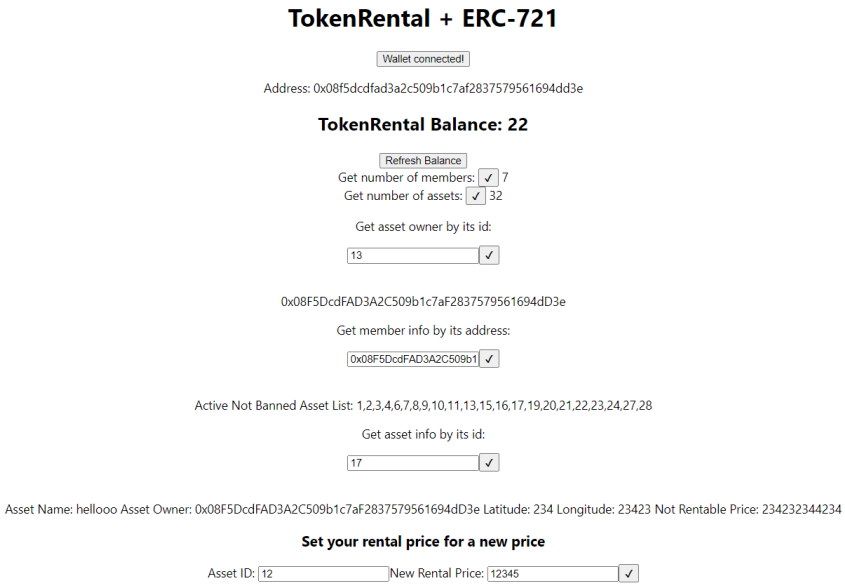

Figure 5.4. Main page of our web application

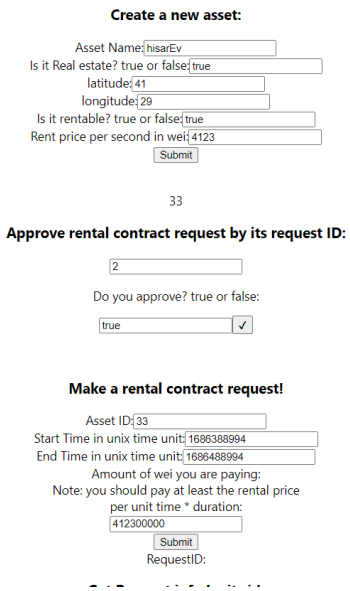

Figure 5.5. Main page of our web application continued

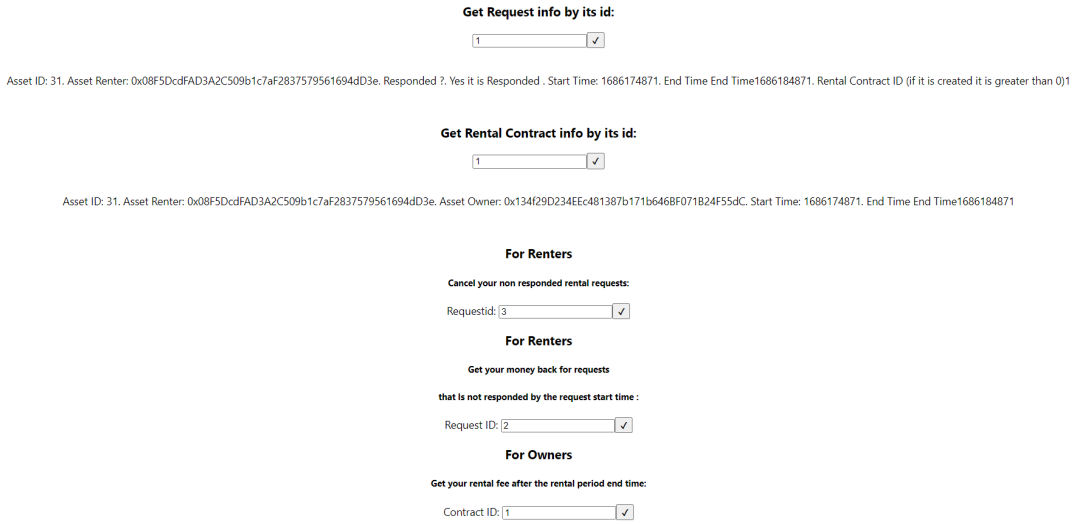

Figure 5.6. Main page of our web application continued

6. IMPLEMENTATION AND TESTING

6.1. Implementation

We are benefiting from one of the contracts provided by OpenZeppelin, ERC-721. The reason we are using ERC-721 is because of its use in non-fungible tokens (NFTs) on the Ethereum network. While each fungible token is identical to each other and therefore can be exchanged for one another, NFTs are represented by their unique identifiers (token IDs). This way they can not be traded with other tokens, or replicated. ERC-721 tokens can be used to represent digital art, collectible, game items and other unique items, in our case for example hotel rooms, which makes it suitable for our needs in a token rental system.

ERC-721 contracts consist of rules that define how non-fungible tokens are created, owned and transferred. They can be bought and sold, and provides a secure way to transfer ownership of unique items.

We have formed a plan to implement the rental contract system. First step is to create a contract that implements ERC-721. This contract defines the structure of the rentable tokens and the way of transferring ownership of tokens temporarily, or the data structure that defines the intervals of time in various duration types such as minutes, hours, days etc.

Next step was to define a rental agreement contract that defines the rental object, price, period, security deposit, terms of the rental agreement and other details.

After the contracts are implemented, the next step was to mint tokens that represent the rental objects such as hotel rooms or units like in Airbnb, or digital NFT assets.

We were planning to come up with multiple versions of the project that deals with the transferring of the ownership of the tokens in various ways.

Instead of multiple versions, we implemented one smart contract that provides functions to create rentable tokens, namely assets, which can be virtual or physical objects such as NFTs or real estate. Our implementations also include functions to send requests to rent tokens, to approve the rental contract requests and receive payments.

When a member send a request to rent an asset, the necessary checks are made to make sure the asset is available to rent in the desired period, and the member provided enough amount of payment. If the request passes all of the necessary checks, the request is sent to the asset owner. After that, the asset owner can send their response to the request by approving or rejecting. If the request is rejected, the payment is refunded to the member who sent the request. If the request is approved, the payment will be safely held in the smart contract and the rental period will begin in due time. If the rental period successfully ends, the payment will be transferred to the asset owner's address.

Additionally we implemented view functions for each entity including Members, Assets, Rental Contracts and Rental Contract Requests to provide transparency in each step.

We implemented our functions securely, in a way that each function validates the user and does not let any unauthorized member take any actions.

6.2. Testing

We used Remix IDE on our implementation face and did always our unit tests on the smart contract. We conducted verbally user scenarios and its tests.

After implementing the web application we continued or test on web application.

Unfortunately we are not satisfied with the web application product, since we could not conducted test as we wanted. There were small bugs that could not be resolved before the deadline. However our smart contract is tested extensively.

Developments and more tests are needed for the web application and planned.

6.3. Deployment

The deployment stage consists of two steps. First we compile the smart contract code and deploy it on the Ethereum Mainnet blockchain network. In order to do so, we must pay the gas fees in Ether.

Secondly, we were planning to deploy the web application. After we deploy the contract and build the web application that provides an interface to access the functions of the contract such as renting out a token, we must host it on a web server, so that it can be accessed by the public.

We tried many testnets and faced issues where many popular and efficient testnets being deprecated. After extensive research, we decided on deploying the smart contract on Sepolia Testnet.

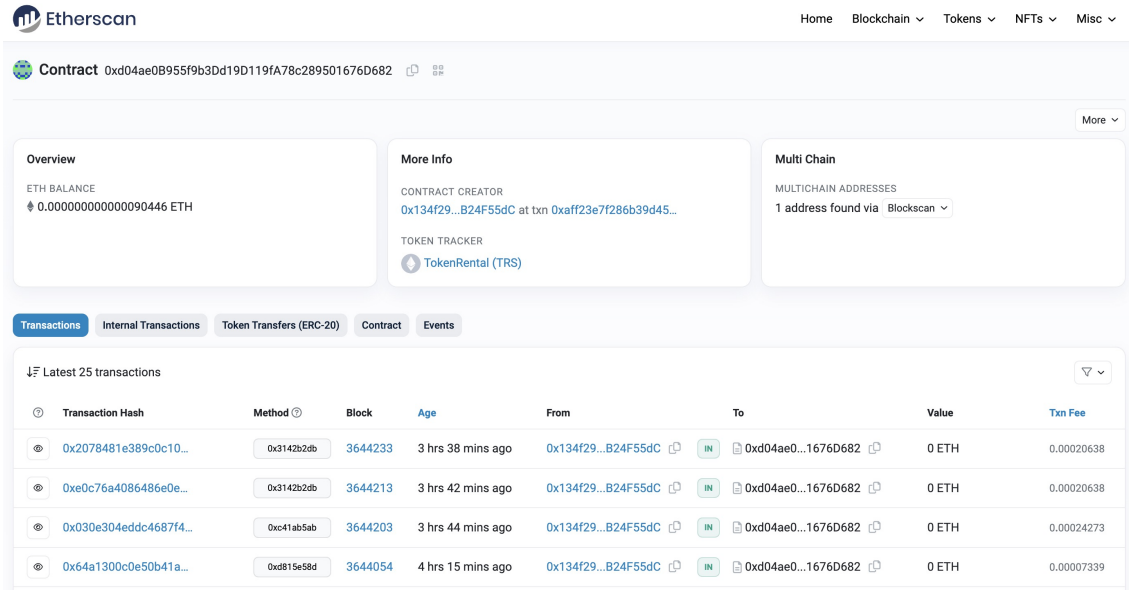

Figure 6.1. Recent transactions of our smart contract on Sepolia Testnet

7. RESULTS

After following the steps of our plan, we have created a blockchain based token rental system utilizing ERC-721 tokens. The system was developed with Solidity and deployed on Sepolia Testnet. The system is applicable on various kinds of assets, both physical and digital. It provides a secure interface.

8. CONCLUSION

In conclusion, our project addresses the transparency and security problems in rental contracts by eliminating the need for third party rental companies. The system can be utilized for different types of assets which makes it modular. The system provides fast solutions by implementing high performance algorithms and data structures. Further research and development will improve the functionalities and profitability of the system.

REFERENCES

1. Cormen, T. H., C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms, Third Edition - Interval Trees*, The MIT Press, 2009.

2. Sedgewick, R., *Algorithms in C, Part 5: Graph Algorithms*, Princeton University, 1990.

3. "Red black trees", `https://pages.cs.wisc.edu/~paton/readings/Red-Black-Trees/`.

4. "Red black trees", `https://www.cs.auckland.ac.nz/software/AlgAnim/red_black.html`.

5. "Red black trees in Solidity", `https://github.com/bokkypoobah/BokkyPooBahsRedBlackTreeLibrary`.