PROJECT REPORT

Lottery Dapp

Decentralized Application

Instructor:

Mr. Yongchang He

Members:

Hai Nam Nguyen – 000520322 – nguyen0465@saskpolytech.ca

Cong Chi Tai Nguyen - 000516006 - nguyen6169@saskpolytech.ca

Xuan Hieu Nguyen – 000518043 – nguyen8191@saskpolytech.ca

Table of Contents

1. Problem Definition

Nowadays, it's necessary to maintain fairness and trust among participants in the Lottery. Thus, ensuring the integrity of lottery draws and protecting against **cyber-attacks, fraud, and manipulation** are top concerns. However, this system has been facing a huge amount of information technology challenges including **security, transaction handling, data privacy, and user experience**. Those challenges arise while handling transaction volumes leading to finding out the solution for **scalability, compliance,** mitigating the impact of **system failures** and ensuring **uninterrupted operations**. Addressing these challenges is important for lottery operators to keep the integrity, security, and reliability of our systems, enabling trust and confidence among participants.

For those reasons, we would like to create something new to leverage three main specifications of Blockchain:

- **Transparency**: Blockchain offers an unchangeable, transparent ledger of transactions. Every participant can inspect that the lottery procedure is fair and free from fraud or manipulation.
- **Decentralization**: traditional lotteries are frequently run by a centralized organization. Blockchain-based lotteries can work in a decentralized way, eliminating the requirement for a central authority and reducing corruption and manipulation.
- **Smart Contracts**: The lottery process from ticket sales to prize distribution can be run automatically using smart contracts, which are self-executed with the conditions of the agreement put into code. This will reduce administrative expenses and guarantee lottery execution fairness.

2. Contract Design

- Name of the contracts:
  - **Lottery**
  - **DataConsumerV3**

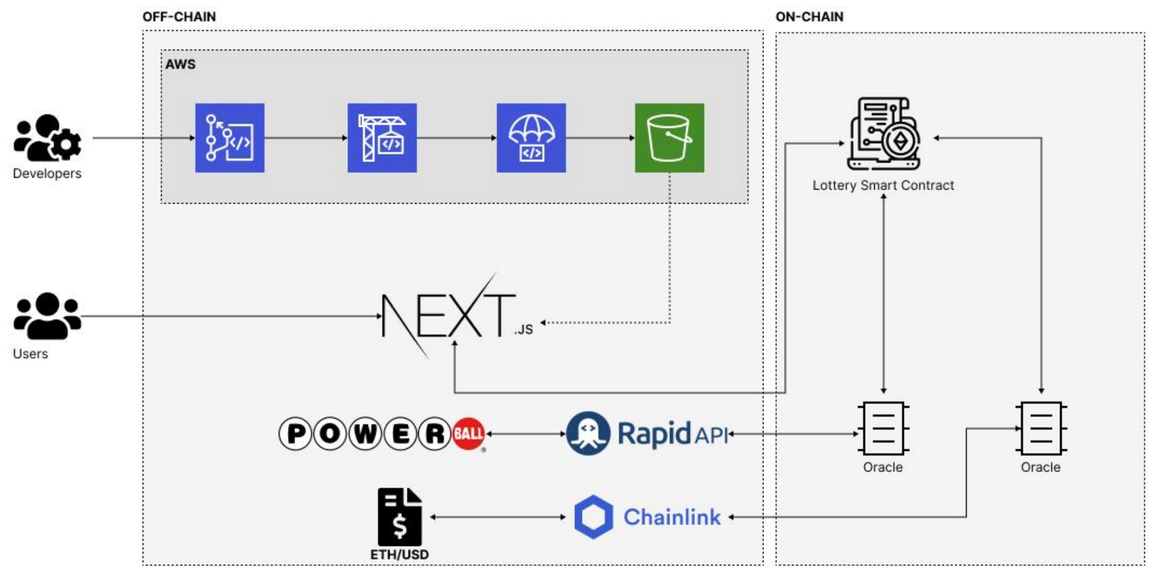
- Interfaces and libraries that the contracts will be use:
    - **ILotteryDataInterface**
    - **IDataConsumerInterface**
    - **Chainlink Data Feeds**
- The Lottery contract is designed to ensure a fair and transparent lottery system. It includes the following functions:
    - **enter()**: This a public function, meaning anyone can call it, and it is marked as payable to allow it to receive Ether along with the function call
    - **random()**: Define a private function and marked as view, which means it does not modify the contract state, and it returns a random number as a uint.
    - **pickWinner()**: This function is used to select a random winner from the list of players and then transfer the contract 80% balance to the winner
    - **getPlayers()**: This function allows anyone to view the list of players in the lottery
    - **pickPowerballWinner()**: This function is used to select a Powerball winner from the list of players and then transfer the contract 20% balance to the winner
    - **modifier restricted()**: This modifier restricts the access to pickWinner() and pickPowerballWinner() functions, allowing only the manager to call them
- DataConsumerV3 contract to connect our contract to asset pricing data for the ETH / USD feed, and Powerball result via RapidAPI by the Immediate-read pattern.
    - **getPowerballResult()**: Retrieves the latest Powerball result from RapidAPI.

3. Contract Architecture

The Lottery application will have the architecture as in the figure below and use the latest technologies such as Amazon Web Services (AWS) for the DevOps, NextJS for the front-end application, and Smart Contract, Oracle on Ethereum blockchain for the logic application:

The Lottery application will use modern technologies like Amazon Web Services (AWS) or DevOps, NextJS for the front-end. It will also utilize Smart Contracts deployed on the

Ethereum blockchain to manage the logic. Additionally, an Oracle will be integrated into the system to fetch external data and interact with the Smart Contracts on-chain.



For DevOps, the team will use AWS CodePipeline, CodeCommit, CodeBuild, and CodeDeploy to manage Continuous Integration and Continuous Delivery (CI/CD). This process is dedicated to the front-end application, which will be deployed to AWS Simple Storage Services (S3) and hosted as a static website.

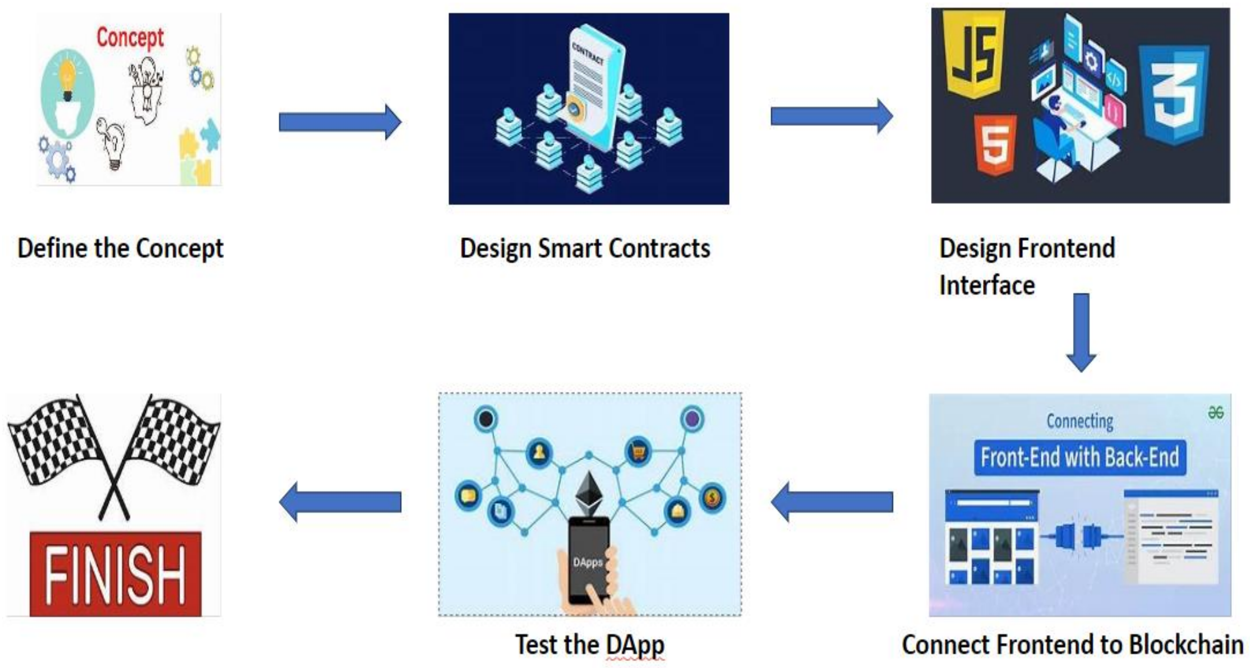
The front-end application will be developed using the NextJS framework, which leverages the React library as its core. It will establish communication with the on-chain application logic through a smart contract named Lottery Smart Contract, storing crucial data from the front-end application such as player addresses, randomly chosen numbers, and the list of players in the Lottery poll.

An on-chain Oracle will invoke APIs from Rapid API to fetch data from Powerball in the United States of America. This data will be securely stored on-chain and retrieved by the Lottery Smart Contract whenever the logic triggers to determine the winner, who will receive 20% of the total winning pool.

4. Implementation

We followed 6 stages to develop this application:

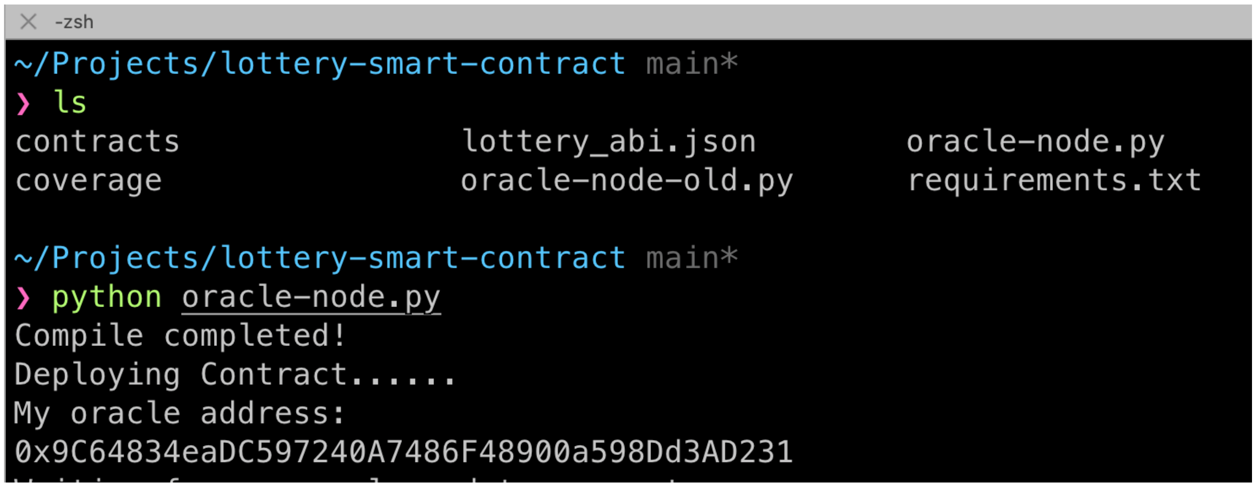
- **Configure CI/CD**: to leverage the strength of AWS CI/CD, we created a pipeline to automate building, deploying our frontend and backend and decreasing manual process in our development.
- **Define the Concept**: we defined the purpose and functionality of the DApp. What problem does it solve? Who are the target users? Then we will find the suitable Blockchain platform for Dapp's requirements.
- **Design Smart Contract**: we wrote smart contracts to define the rules and logic of the DApp using Solidity.
- **Design Frontend interface:** frontend was designed by using NextJS framework, which leverages the React library as its core.
- **Connect Frontend to Blockchain:** after finishing the frontend, we integrated it to the Blockchain backend using Web3py.
- **Test Dapp**: finally, we tested its functions on local network to make sure everything was running as we expected.



5. Testing

For testing this application, we followed these steps:

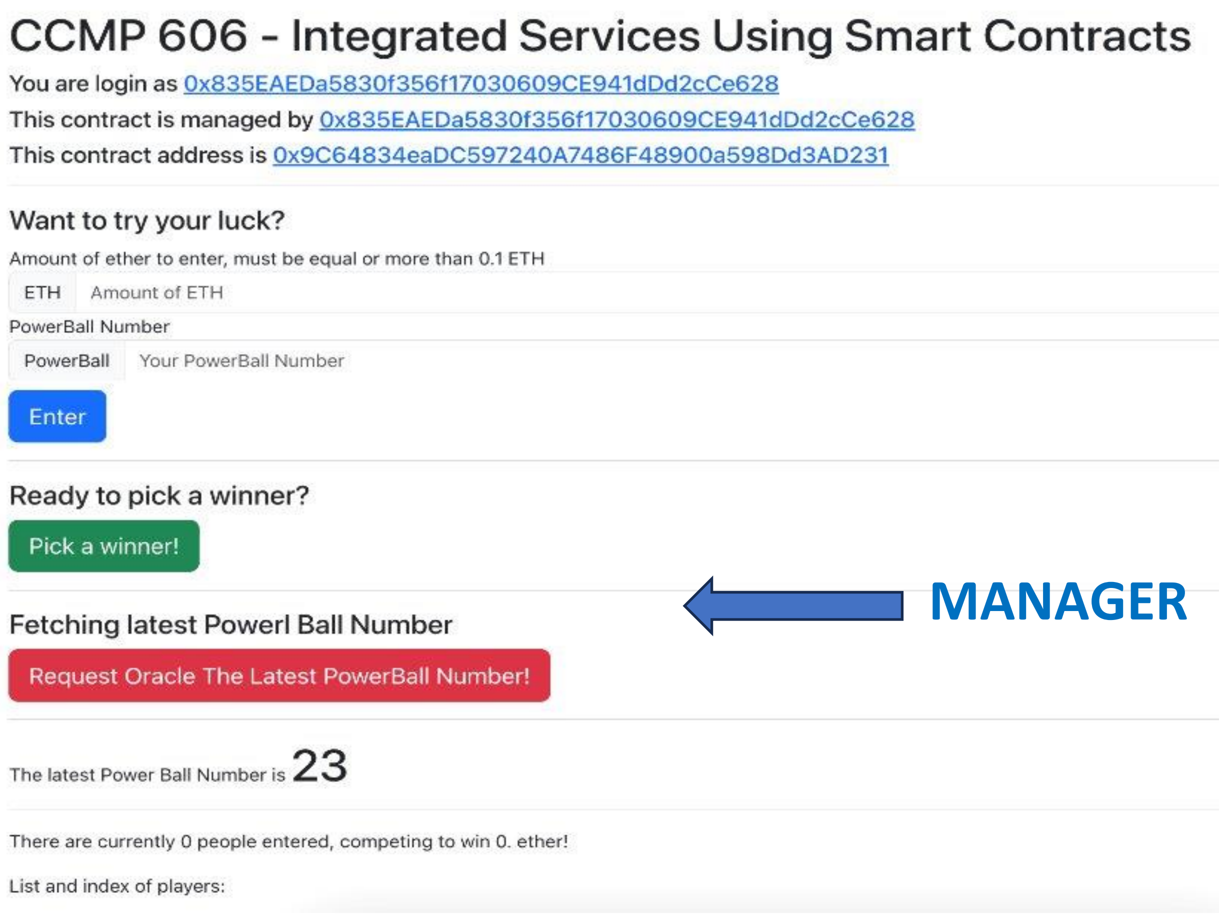
- **Deploy contract in local network**.



```
-zsh
~/Projects/lottery-smart-contract main*
❯ ls
contracts                lottery_abi.json         oracle-node.py
coverage                 oracle-node-old.py       requirements.txt

~/Projects/lottery-smart-contract main*
❯ python oracle-node.py
Compile completed!
Deploying Contract......
My oracle address:
0x9C64834eaDC597240A7486F48900a598Dd3AD231
```

- **Deploy Frontend**



CCMP 606 - Integrated Services Using Smart Contracts

You are login as 0x835EAEDa5830f356f17030609CE941dDd2cCe628

This contract is managed by 0x835EAEDa5830f356f17030609CE941dDd2cCe628

This contract address is 0x9C64834eaDC597240A7486F48900a598Dd3AD231

Want to try your luck?

Amount of ether to enter, must be equal or more than 0.1 ETH

ETH Amount of ETH

PowerBall Number

PowerBall Your PowerBall Number

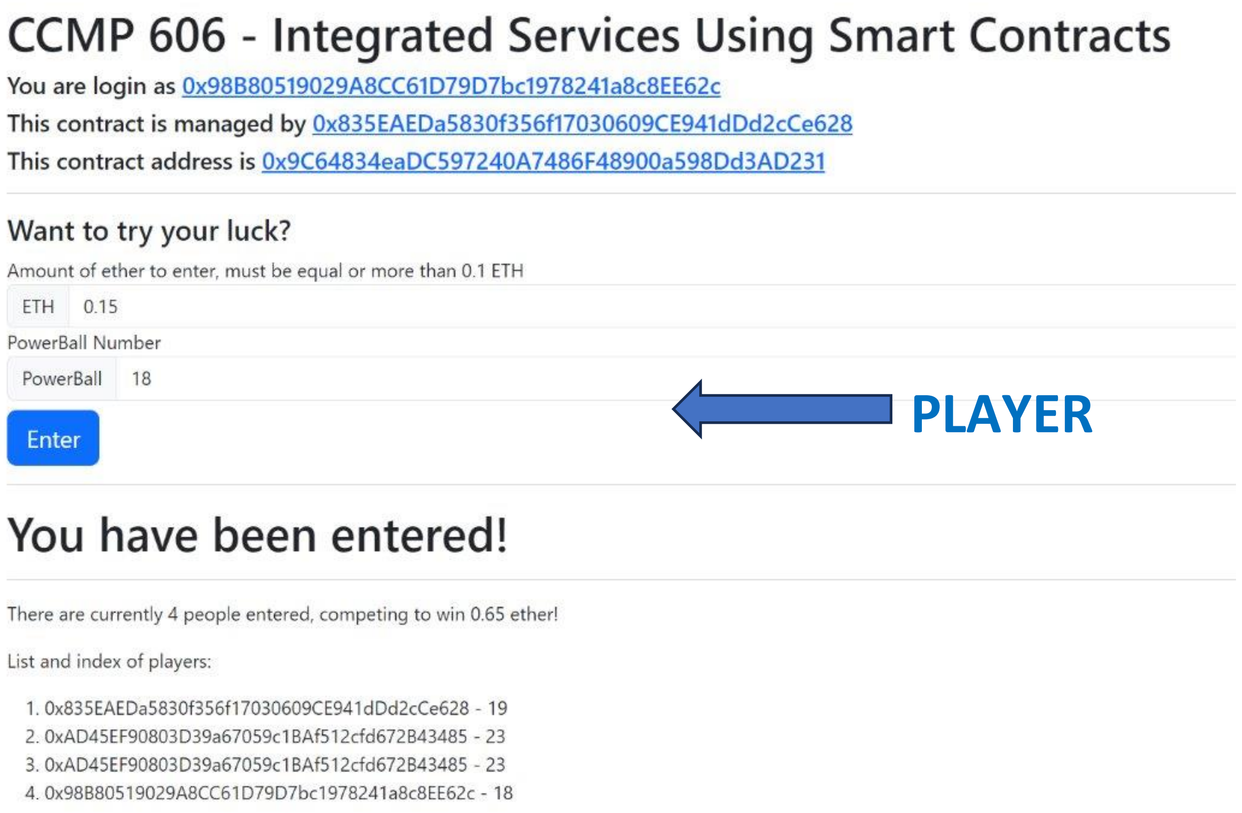
Ready to pick a winner?

Fetching latest Powerl Ball Number

The latest Power Ball Number is 23

There are currently 0 people entered, competing to win 0. ether!

List and index of players:



CCMP 606 - Integrated Services Using Smart Contracts

You are login as 0x98B80519029A8CC61D79D7bc1978241a8c8EE62c

This contract is managed by 0x835EAEDa5830f356f17030609CE941dDd2cCe628

This contract address is 0x9C64834eaDC597240A7486F48900a598Dd3AD231

Want to try your luck?

Amount of ether to enter, must be equal or more than 0.1 ETH

ETH 0.15

PowerBall Number

PowerBall 18

Enter

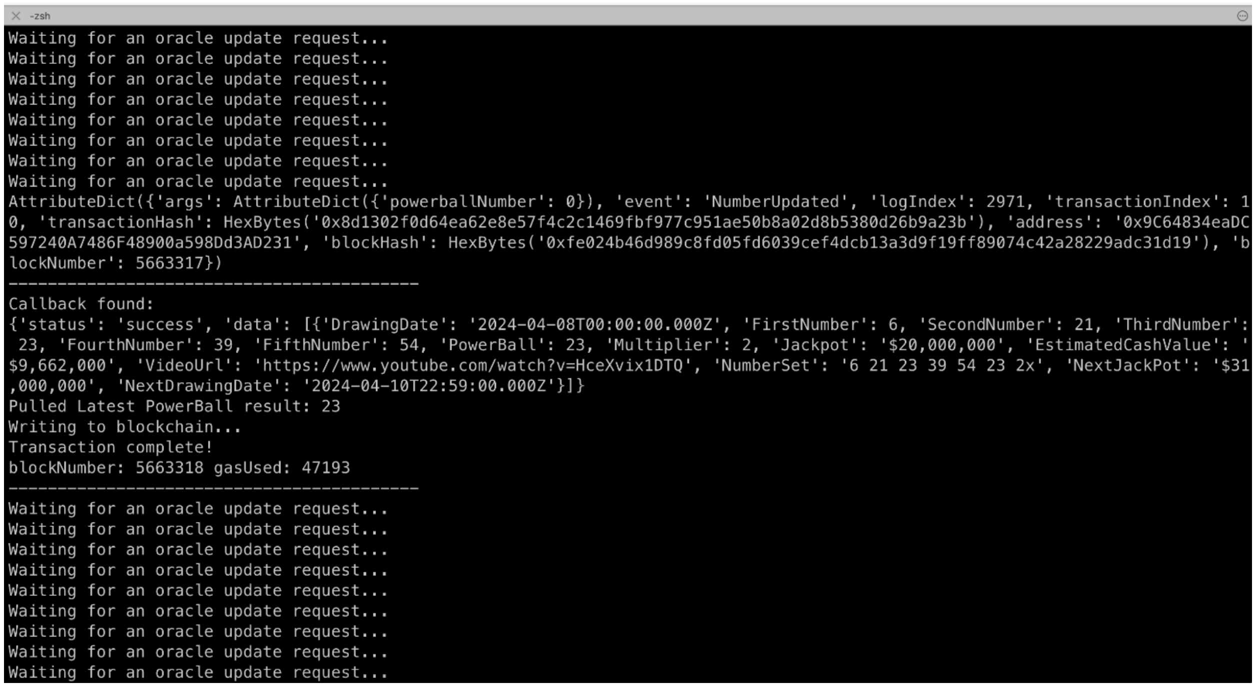
You have been entered!

There are currently 4 people entered, competing to win 0.65 ether!

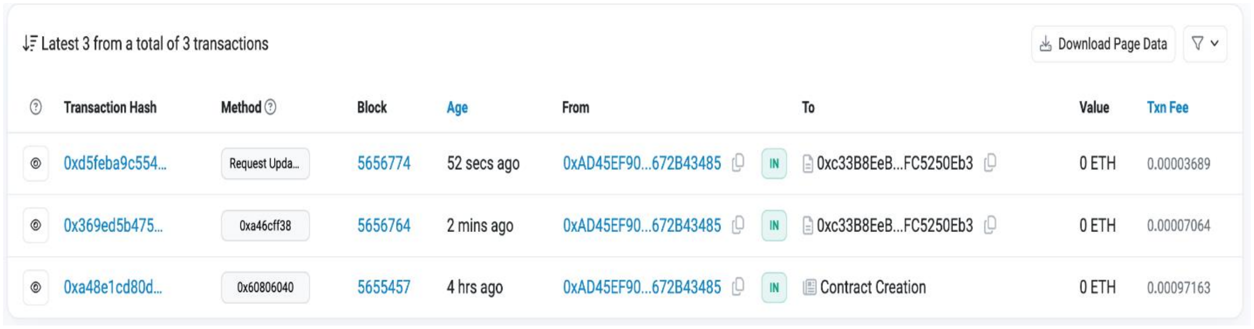
List and index of players:

1. 0x835EAEDa5830f356f17030609CE941dDd2cCe628 - 19
2. 0xAD45EF90803D39a67059c1BAf512cfd672B43485 - 23
3. 0xAD45EF90803D39a67059c1BAf512cfd672B43485 - 23
4. 0x98B80519029A8CC61D79D7bc1978241a8c8EE62c - 18

- **Update PowerBall**



```
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
AttributeDict({'args': AttributeDict({'powerballNumber': 0}), 'event': 'NumberUpdated', 'logIndex': 2971, 'transactionIndex': 1
0, 'transactionHash': HexBytes('0x8d1302f0d64ea62e8e57f4c2c1469fbf977c951ae50b8a02d8b5380d26b9a23b'), 'address': '0x9C64834eaDC
597240A7486F48900a598Dd3AD231', 'blockHash': HexBytes('0xfe024b46d989c8fd05fd6039cef4dcb13a3d9f19ff89074c42a28229adc31d19'), 'b
lockNumber': 5663317})
-----------------------------------------
Callback found:
{'status': 'success', 'data': [{'DrawingDate': '2024-04-08T00:00:00.000Z', 'FirstNumber': 6, 'SecondNumber': 21, 'ThirdNumber':
 23, 'FourthNumber': 39, 'FifthNumber': 54, 'PowerBall': 23, 'Multiplier': 2, 'Jackpot': '$20,000,000', 'EstimatedCashValue': '
$9,662,000', 'VideoUrl': 'https://www.youtube.com/watch?v=HceXvix1DTQ', 'NumberSet': '6 21 23 39 54 23 2x', 'NextJackPot': '$31
,000,000', 'NextDrawingDate': '2024-04-10T22:59:00.000Z'}]}
Pulled Latest PowerBall result: 23
Writing to blockchain...
Transaction complete!
blockNumber: 5663318 gasUsed: 47193
-----------------------------------------
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
Waiting for an oracle update request...
```

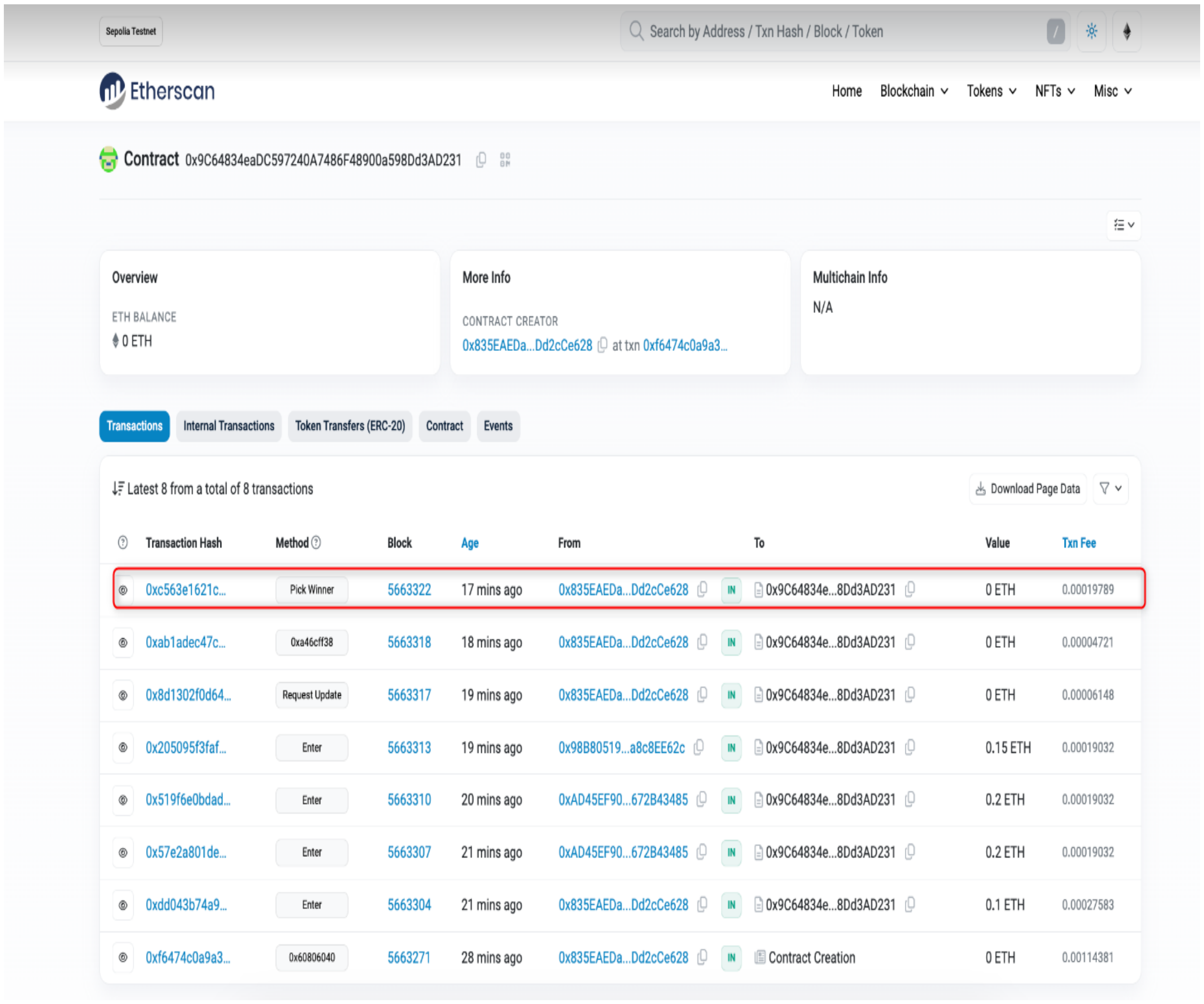


Latest 3 from a total of 3 transactions

Download Page Data

Transaction Hash	Method	Block	Age	From		To	Value	Txn Fee
0xd5feba9c554...	Request Upda...	5656774	52 secs ago	0xAD45EF90...672B43485	IN	0xc33B8EeB...FC5250Eb3	0 ETH	0.00003689
0x369ed5b475...	0xa46cff38	5656764	2 mins ago	0xAD45EF90...672B43485	IN	0xc33B8EeB...FC5250Eb3	0 ETH	0.00007064
0xa48e1cd80d...	0x60806040	5655457	4 hrs ago	0xAD45EF90...672B43485	IN	Contract Creation	0 ETH	0.00097163

- **Pick Winner**



Sepolia Testnet

Search by Address / Txn Hash / Block / Token

Etherscan

Home Blockchain Tokens NFTs Misc

Contract 0x9C64834eaDC597240A7486F48900a598Dd3AD231

Overview

ETH BALANCE

0 ETH

More Info

CONTRACT CREATOR

0x835EAEDa...Dd2cCe628 at txn 0xf6474c0a9a3...

Multichain Info

N/A

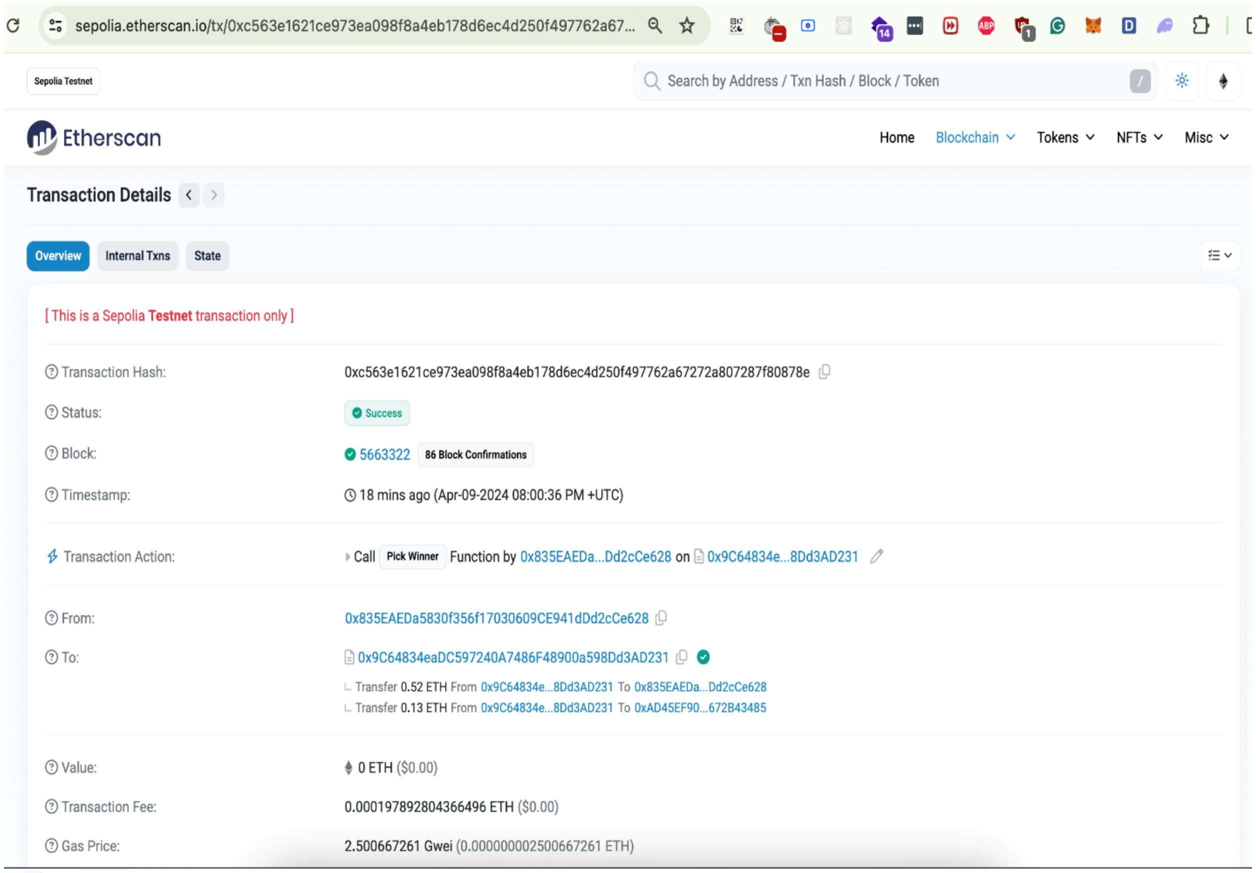
Transactions | Internal Transactions | Token Transfers (ERC-20) | Contract | Events

Latest 8 from a total of 8 transactions

Download Page Data

Transaction Hash	Method	Block	Age	From		To	Value	Txn Fee
0xc563e1621c...	Pick Winner	5663322	17 mins ago	0x835EAEDa...Dd2cCe628	IN	0x9C64834e...8Dd3AD231	0 ETH	0.00019789
0xab1adec47c...	0xa46cff38	5663318	18 mins ago	0x835EAEDa...Dd2cCe628	IN	0x9C64834e...8Dd3AD231	0 ETH	0.00004721
0x8d1302f0d64...	Request Update	5663317	19 mins ago	0x835EAEDa...Dd2cCe628	IN	0x9C64834e...8Dd3AD231	0 ETH	0.00006148
0x205095f3faf...	Enter	5663313	19 mins ago	0x98B80519...a8c8EE62c	IN	0x9C64834e...8Dd3AD231	0.15 ETH	0.00019032
0x519f6e0bdad...	Enter	5663310	20 mins ago	0xAD45EF90...672B43485	IN	0x9C64834e...8Dd3AD231	0.2 ETH	0.00019032
0x57e2a801de...	Enter	5663307	21 mins ago	0xAD45EF90...672B43485	IN	0x9C64834e...8Dd3AD231	0.2 ETH	0.00019032
0xdd043b74a9...	Enter	5663304	21 mins ago	0x835EAEDa...Dd2cCe628	IN	0x9C64834e...8Dd3AD231	0.1 ETH	0.00027583
0xf6474c0a9a3...	0x60806040	5663271	28 mins ago	0x835EAEDa...Dd2cCe628	IN	Contract Creation	0 ETH	0.00114381

- **Transfer ETH to winner**



sepolia.etherscan.io/tx/0xc563e1621ce973ea098f8a4eb178d6ec4d250f497762a67...

Sepolia Testnet

Search by Address / Txn Hash / Block / Token

Etherscan

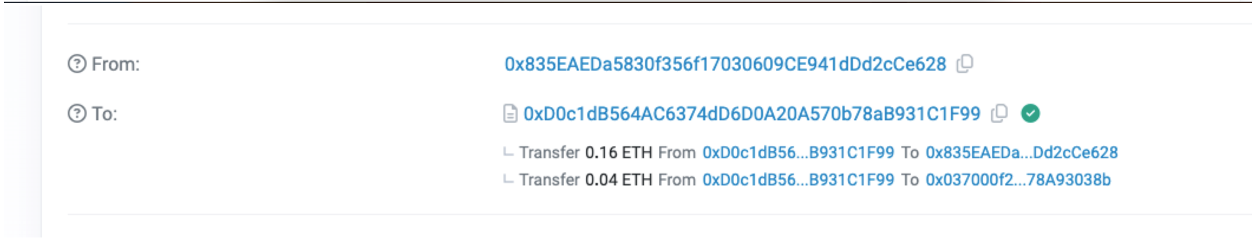
Home Blockchain Tokens NFTs Misc

Transaction Details

Overview Internal Txns State

[This is a Sepolia **Testnet** transaction only]

Transaction Hash:	0xc563e1621ce973ea098f8a4eb178d6ec4d250f497762a67272a807287f80878e
Status:	Success
Block:	5663322 86 Block Confirmations
Timestamp:	18 mins ago (Apr-09-2024 08:00:36 PM +UTC)
Transaction Action:	Call Pick Winner Function by 0x835EAEDa...Dd2cCe628 on 0x9C64834e...8Dd3AD231
From:	0x835EAEDa5830f356f17030609CE941dDd2cCe628
To:	0x9C64834eaDC597240A7486F48900a598Dd3AD231 └ Transfer 0.52 ETH From 0x9C64834e...8Dd3AD231 To 0x835EAEDa...Dd2cCe628 └ Transfer 0.13 ETH From 0x9C64834e...8Dd3AD231 To 0xAD45EF90...672B43485
Value:	0 ETH ($0.00)
Transaction Fee:	0.000197892804366496 ETH ($0.00)
Gas Price:	2.500667261 Gwei (0.000000002500667261 ETH)



From:	0x835EAEDa5830f356f17030609CE941dDd2cCe628
To:	0xD0c1dB564AC6374dD6D0A20A570b78aB931C1F99 └ Transfer 0.16 ETH From 0xD0c1dB56...B931C1F99 To 0x835EAEDa...Dd2cCe628 └ Transfer 0.04 ETH From 0xD0c1dB56...B931C1F99 To 0x037000f2...78A93038b

6. Conclusion

In conclusion, blockchain technology has several advantages that make it a desirable choice for establishing an equitable, open, and safe lottery system. Lottery organizers can use blockchain to build a decentralized, transparent, and unchangeable system that promotes participant trust, enables access to anywhere in the world, and uses smart contracts to automate procedures.