Project Report: Pyramid Cards

Michael Yang, Zichen Chang, Kevin Chu, Tai Jing, Zhiwei Li

EECE 571G Blockchain Software Engineering
April 9, 2024

Abstract

The Pyramid Cards project is the first of many steps to integrate blockchain technology with the online gaming, and lottery is a vested sector. In essence, Pyramid Cards intends to benefit from decentralized system technology to create a higher level of transparency, fairness, and user responsiveness in card collection games. By combining solidity smart contracts with Chainlink's Verifiable Random Function , Pyramid Cards created a transparent platform with complete draw records.

Pyramid Cards offer a secure, unalterable platform, with every card game draw being transparent and verifiable. It will not only guarantee the fairness of the game but also redefine the industry standard for trust in gaming. Our DApp is designed to bring collectors, players, and blockchain enthusiasts a fun and entertaining experience while offering a fair and user-friendly gaming ecosystem.

The github link to the project is: `https://github.com/Will-Li-zw/571G`

Contents

1 Introduction

The consistent improvement of the blockchain technology has inspired various innovative decentralized applications in different sectors such as the online gaming and lottery. Our "Pyramid Cards" project aims to combine the fun of card collection with the transparency and security of blockchain technology. Our vision is to create a DApp that not only offers an exciting and rewarding user experience but also sets a new standard for transparency and fairness in the industry. We believe that users should have full rights to stay informed about the games they play and the platforms they engage with. The mission of Pyramid Cards is to take advantage of the power of blockchain to ensure true randomness, fairness, and an enjoyable gaming experience for all users.

2 Problem Statement

The global makert for online lottery and card collection is vast. According to the IMARC Group report, the global online lottery market was approximately USD 11.0 billion in 2023 and it is estimated to reach about USD 16.76 million by 2029[2]. However, the existing systems often lacks transparency and fairness because usually their probabilities are concealed, leaving users ignorant of their true chance of obtaining rare items or winning. Such grey zone may lead to user exploitation, where platforms might easily adjust probabilities to maximize their profits. For example, by initially making it easy for users to acquire new cards and then making it increasingly difficult to complete a collection, users are incentivized to irrationally spend money on collecting the last card.

In some countries, although regulations require the disclosure of relevant probabilities, these figures are not always guaranteed to correspond to actual chances, given the lack of access to internal code. This discrepancy raises doubt about trust and fairness, and deters potential users from investing time and resources into a system that might be inherently biased against them.

Actually, these issues not only harm users but also hinder the healthy growth and adoption of online gaming industry. As more users become aware of the potential for unfair practices, The reputation and sustainability of the industry would be significantly compromised.

3 Market Review

Blockchain is having a significant impact on the gaming industry. Different from centralized services, true asset ownership is a popular concept in games. Under this model, players may own their in-game assets, which they can then sell or trade because they have genuine versions. As a result, the crypto-gaming industry has soared, leading to potential reshape of gaming economies and player experiences[1]. A notable example of this trend is Axie Infinity, a game where players can genuinely own, sell, and trade their in-game assets as NFTs, reaching over 2.7 million daily active users in 2022[4].

Also, blockchain technology is transforming digital lottery draws. By using blockchain and open-source algorithms, innovative players like adessoDraws provide more transparency than traditional draws. This strategy boosts trustworthiness and substantially reduces digital lotteries' possible fraud, which have long plagued traditional electronic draw systems. Due to this transparent and fair nature, such Blockchain-based lotteries are increasingly gaining popularity and major enterprises are exploring this area. For instance, Tabcorp is examining distributed ledger technology for Australia's national lotteries[3], aiming to enhance transaction integrity and bolster their digital product offerings.

4 Solution Overview

Based on the former successful cases, the Pyramid Cards aim to bring transparency and fairness to online card collection and lottery games by leveraging blockchain technology . The core mechanism of our DApp derives from the smart contracts that manage card distribution, ensuring that every card draw is verifiable, transparent, and consequently tamper-proof.

Key features of our DApp:

- **Transparent Odds:** Utilizing smart contracts, Pyramid Cards openly displays the probabilities of each card, ensuring that users are fully informed about their chances of winning or collecting rare cards, which helps to establish reasonable expectation and decision.

- **Blockchain-Based Fairness:** The DApp relies on Chainlink's Verifiable Random Function (VRF) to generate randomness for card draws. This guarantees the fairness of randomness and provides proof of integrity directly on the blockchain.

- **Rewarding Collection Mechanism:** Players can collect card sets to redeem prizes, including Ethereum tokens, which provides users with an incentive of ultimate reward

beyond collection value.

- **Peer-to-peer Interaction Among Users:** Players have complete control over their collected items and can trade (in future versions) or redeem cards based on their strategies.

Use case:

- **Transparent and Fair Online Reward:** The features of Pyramid Cards ensure users can trust that the outcomes are genuinely random and unbiased. This use case targets users who enjoy online gaming but are concerned about the fairness and integrity of current platforms.
- **Card Collection Game for Collectors:** Players who value digital collectibles will find Pyramid Cards an reliable and fair platform. Users can collect, trade, and compete with their digital cards, similar to popular physical card games like Pokemon, Yu-Gi-Oh, etc.
- **Branding and Promotions:** Various brands, celebrities, and IP owners could partner with this platform to launch exclusive branded digital collectibles. For example, a sports team could release limited-edition player cards that fans can collect, trade, and potentially redeem for limited edition autographed jerseys.

In addition to a fair and enjoyable gaming experience, our DApp also has broader implications for the online gaming industry. By demonstrating a more transparent and user-centric model, we aspire to set a new standard and drive the adoption of blockchain-based gaming solutions.

As we grow, we envision to expand our business modes, such as partnerships with digital artists for exclusive card designs, and the integration of NFT technology to enhance the value and ownership of digital assets within our ecosystem.

In summary, our DApp is dedicated to eradicating the unfairness and exploritation in online gaming and lottery by harnessing the power of blockchain. We hope this is a starting point to revolutionize the experience within this industry.

5 Technical Implementation

5.1 Overview

This project consists of two part: Back End services and Front End presentation. The recommended node service version for this project to run is $\geq$ node 18. Under the node.js

environment, the Back End is enabled through Hardhat and Foundry developing tools. Hardhat is used for smart contract developmen and deployment. It provides a robust development environment with features like built-in testing frameworks, debugging tools, and automated contract deployments. Foundry is utilized for deploying and managing the app. It simplifies the testing process by simplifying local chain mocking and monitoring contract interactions.

As for the Front End, this project adopts the state-of-the-art frontend developing framework React with web3.js module. React is a popular JavaScript library for building user interfaces, developed and maintained by Facebook. It's known for its simplicity, efficiency, and flexibility, making it a preferred choice for building modern web applications. At its core, React utilizes a component-based architecture, where UIs are composed of reusable and encapsulated components. Each component manages its own state, allowing for better organization and easier maintenance of code. React's declarative nature enables developers to describe how the UI should look at any given time, and React efficiently updates and renders the components when the underlying data changes. Together with the Web3.js module, the interaction to our deployed contract on-chain would be driven smoothly through the cooperation with backend smart contract API. The technology stack is shown in the figure 1:

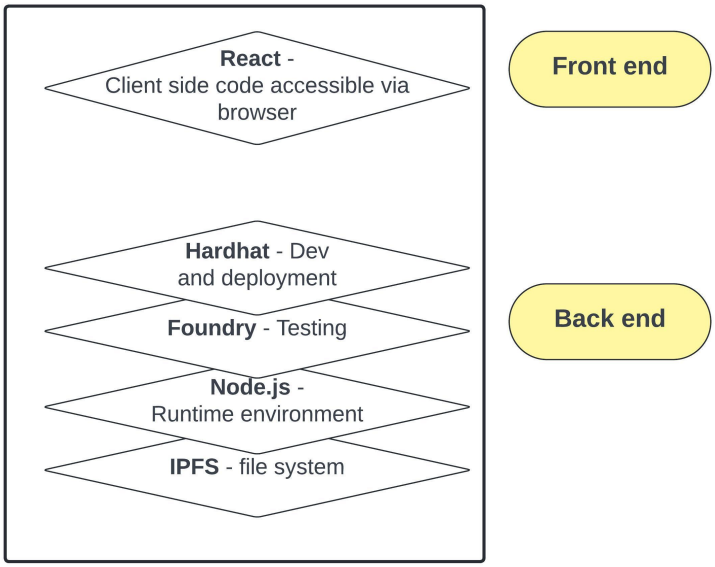

Figure 1: Tech Stack

Since March 11, 2024, we have made significant progress in development. By March 16, 2024, we completed the initial interface prototype and established the skeleton of our back-end architecture. Subsequently, on March 21, 2024, we finished developing the core smart contracts and APIs for seamless communication between the front-end and back-end

systems. The development and testing of finalized smart contracts was completed on March 24, 2024. Finally, we concluded the user interface enhancements and deployed the finished application on March 29, 2024.

5.2 Smart Contract Details

The follows are critical functions within the Pyramid Cards smart contract:

- **Card Distribution:**

  By sending enough Ether to the smart contract, users gain chances to draw cards from any available card pool. When a user request a card draw, the smart contract will send a request to Chainlink's Verifiable Random Function system to generate a random number. This random number determines the card ID that the user receives, based on predefined probabilities for each card. The obtained card is then allocated to the user's collection. This process ensures that each card draw method is unique, unpredictable, and that they can be verified on the blockchain, which maintains the game's integrity.

  Main functions includes:

  - `addBalance()`: Enables users to add a balance to their account, which is used for drawing cards. This function takes an ether value and updates the user's draw chances accordingly.
  - `redeemChance(uint256 id)`: Allows users to redeem a specific number of identical cards for a draw chance. This function decreases the card quantity and updates the user's draw balance.
  - `drawRandomCard(string memory collection)`: Users can draw a random card from a specified collection. The function interfaces with Chainlink VRF to ensure fair and random selection.
  - `fulfillRandomWords(uint256 requestId, uint256[] memory randomWords)`: This is a callback function for the Chainlink VRF that assigns a randomly drawn card to the user's collection based on the generated random number.

- **Transparent Odds:**

  To ensure that each card's probability of drawing is transparent, we implement smart contracts ensuring that the odds are also publicly verifiable and immutable. Such a design is a vital part of the gaming experience since we at Pyramid Cards DApp are

dedicated to ensuring the highest level of fairness and transparency. It operates as mentioned below:

- **Odds Encoding:** The probabilities of each card type in the deck are explicitly coded into the smart contracts and deployed onto the blockchain. As a result, there is no way of hiding the likelihood of drawing any particular card hence a player can review the transparency and all probabilities after deployment, This process reassures the player that there are no invisible mechanisms or changes made after deployment that could affect the draw without their knowledge or favor unfairness.
- **Public Verification:** An independent player may access and verify the smart contract including the encoded odds. This structure of design ensures Pyramid Cards to be free of deception and manipulation giving all participants a zero percentage chance in gaming.
- **Immutability of the Probabilities:** The encoded odds in the smart contracts cannot be changed without deploying an entirely different contract, which is visible to all community members. Once launched, the probability of each card becomes permanent.

Here are some main functions to realize that:

- `createCollections(string memory name, uint256[] memory probabilities, string[] memory urls)`: Admin function to create new card collections with associated probabilities and image URLs.
- `getChanceArray(string memory collection)`: Returns an array of cumulative probabilities for each card in a collection, aiding in determining card draw odds.
- `getUserCollection()`: Provides users with the ability to view their card inventory, including the IDs and quantities of cards they own.

- **Reward Mechanism:**

The Reward Mechanism is designed to create an incentivizing process that is transparent, fair, and verifiable through the blockchain. Here is how it works:

- **Set Completion Rewards:** As a player, users will earn rewards when they collect a complete set of cards. When gaining a complete card collection, a player can redeem the corresponding rewards from the smart contract.

  - **Transparent Reward Criteria:** The criteria for receiving rewards are also transparently included in the smart contract, so that everyone can read them.

  Relevant functions:

  - `redeemAward(string memory awardName)`: Users can redeem a specified award by submitting a collection of required cards. This function checks the user's collection for the required cards and quantities.
  - `createCollections(string memory name, uint256[] memory probabilities, string[] memory urls)`: This is also used for setting up the rewards associated with a certain card set.
  - `getAllRewards()`: Provides information about all available rewards, including required cards and quantities, for the front-end display.

- **Deployment:**

  The deployment of the contract is done with the help of Hardhat developing environment. By providing the proper initial parameters, our contract was successfully deployed on the Sepolia test net. The admin of the contract is the deployer by default.

- **Other resources:**

  In our project, we have leveraged the InterPlanetary File System (IPFS) to store and retrieve images for our card sets, ensuring a decentralized storage solution compatible with our project.

5.3 Front-end Details

Our blockchain application employs modern front-end technologies to deliver a user-centric experience. At its core, it is built using React.js, a widely adopted JavaScript library for creating dynamic user interfaces, complemented by Material-UI for streamlined and visually appealing design elements. Being one of the most popular front-end frameworks, React.js is a powerful library for building dynamic and interactive user interfaces, utilizing a component-based architecture to promote reusability and maintainability of code. React's virtual DOM and efficient reconciliation algorithm allow our website to have blazing-fast rendering time and smooth transition between different UI components.

To ensure a maintainable codebase, clean separation of concerns, and effective management of complex application states, we implement a state management system for a unidirectional data flow paradigm. This approach facilitates seamless data flow and ensures

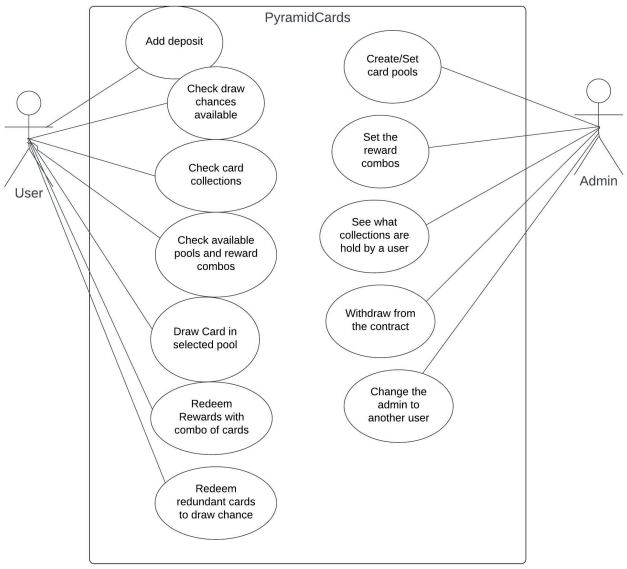

Figure 2: Use Case Diagram

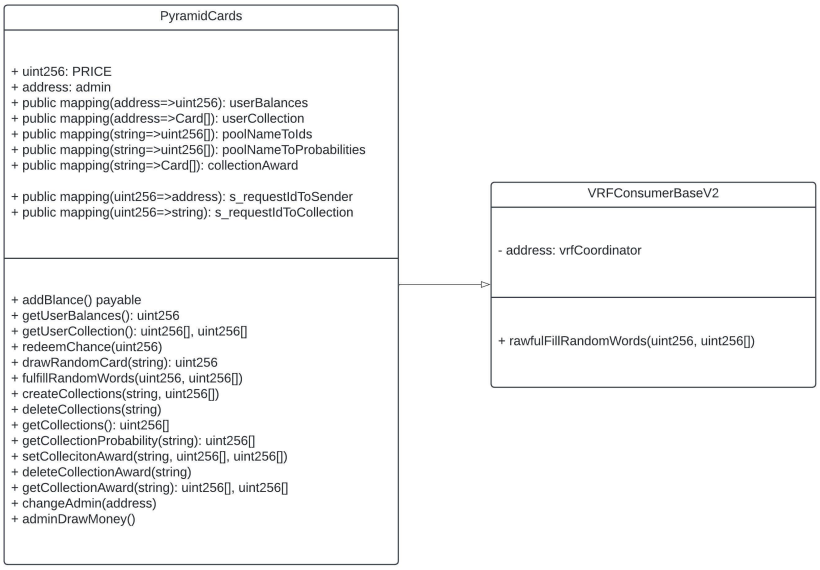

Figure 3: UML class Diagram

coherence across disparate components. The tool we used to build this system is Redux, a state management library written in TypeScript. Redux enables our front-end to have a single source of data across pages, creating a simplified data management system. It also allows us to normalize our data structure. The data communicated to the front-end are lists but with Redux we could transform and normalize the data which improves the ability to query and update data across different parts of our app.

Moreover, scalability and performance are central considerations in our front-end architecture. Techniques such as asynchronous data fetching and optimized rendering are implemented to enhance responsiveness, minimize load times, and facilitate future development. For instance, asynchronous data fetching enables efficient retrieval of data from external sources without blocking the main thread, while optimized rendering techniques ensure smooth user interactions by minimizing unnecessary re-renders. These strategies not only improve the user experience but also lay a solid foundation for scalability as the application grows.

5.3.1 Detailed Implementation:

To achieve modularity and enhance user experience, we write our front-end in separate components each responsible for their own functionality and interface. With this approach, our front-end system is easily tested, maintained, and scaled. Here is the list of the component and a brief description of the purpose of each:

- AccountInformation.tsx: displays the user's number of available draws and their collection of cards. Allows the user to redeem cards, update the collection, and purchase draw chances.
- ActivateDeactivate.tsx: connects and disconnects to/from MetaMask to the Pyramid app.
- AdminPage.tsx: creates the admin page for adding new card collections, assigning rewards, and withdrawing balance.
- DrawCard.tsx: lets the user to select a card pool, show the card odds, draw cards, and purchase more draws.
- HomePage.tsx: defines the homepage of the Pyramid app. Produces navigation links to "My Account", "Draw Card", "Store", and potentially the Admin page if the user logs in from a recognized admin address.
- LoginPage.tsx: the first page the user sees when opening the Pyramid app. Asks the user to log in with their Metamask account to start using the app.
- MyAccount.tsx: container that calls and generates AccountInformation.
- SectionDivider.tsx: style component that defines a clear separation in the UI.
- SignMessage.tsx: enables the user to sign a message with their connected Metamask.

- Store.tsx: displays the available rewards and what cards are required for redeeming the reward. Also shows if the user's inventory has sufficient cards to redeem.
- WalletStatus.tsx: shows the key information of the user's wallet, including account address and balance.

5.4 Testing and Results

We use Foundry for unit testing in our project. The test cases include:

- **Customer Function Test**
  1. Add Balance: User can add multiple draw chances to their balance.
  2. Add Balance Zero Amount: Tests addBalance function with no value received.
  3. Add Balance Non-Multiple Amount: Checks addBalance function with values not a multiple of unit price.
  4. Redeem Chance: User can redeem four identical cards for one draw chance.
  5. Redeem Insufficient Cards: Ensures user cannot redeem a card if there aren't enough cards.
  6. Check User Collection: Verifies users can accurately view their card inventory.
- **Admin Function Test**
  1. Invalid Create Collection: Tests admin's ability to create card collections with invalid inputs.
  2. Admin Rights Transfer: Tests transfer of admin rights to a new address.
  3. Withdraw Contract Money: Checks admin's ability to withdraw money from the contract.
  4. Check Admin Account: Verifies the functionality to check if a user is an admin.
- **Draw Card VRF Function Test**
  1. Draw No Pool: Ensures user cannot draw if the card pool does not exist.
  2. Random Request: Checks the recording of request IDs to the VRF coordinator.
  3. Random Draw for Once: Tests the addition of a random card into the user's collection.

4. Random Draw 5 Times: Assesses the random drawing mechanism over multiple iterations.

The test result is listed in Figure 4

```
➜  571G git:(main) forge test --match-path test/PyramidCards.t.sol -v
[⠒] Compiling...
No files changed, compilation skipped

Running 14 tests for test/PyramidCards.t.sol:PyramidCardsTest
[PASS] testAddBalance() (gas: 43368)
[PASS] testAddBalanceFailWithNonMultipleOfPrice() (gas: 17842)
[PASS] testAddBalanceFailWithZeroAmount() (gas: 11398)
[PASS] testChangeAdmin() (gas: 28304)
[PASS] testCreateCollections() (gas: 929654)
[PASS] testDrawNoPool() (gas: 46704)
[PASS] testGetUserCollection() (gas: 140269)
[PASS] testIsAdmin() (gas: 17327)
[PASS] testRandomDraw5times() (gas: 1381288)
[PASS] testRandomDrawforOnce() (gas: 1016649)
[PASS] testRandomRequest() (gas: 967848)
[PASS] testRedeemChance() (gas: 164038)
[PASS] testRedeemChanceRevertInsufficientCards() (gas: 85087)
[PASS] testWithDrawContractMoney() (gas: 61741)
Test result: ok. 14 passed; 0 failed; 0 skipped; finished in 4.58ms

Ran 1 test suites: 14 tests passed, 0 failed, 0 skipped (14 total tests)
```

Figure 4: Test Result of Foundry

6 User Interface and Experience

The Pyramid Cards platform offers an intuitive user interface and a smooth user experience. We will first introduce the interface designed for admin use and explain how to implement various admin functions. This includes how to operate to create new card collections and new awards, how to withdraw savings from smart contracts, etc. Following that, we will describe what interface and functions users can experience after entering the Pyramid Cards platform, as well as how to interact with smart contracts.

- **Admin Interface and Functions**

1. **Admin Creates New Card Collection and Award:** From Figure 5, we can see that the admin interface primarily provides a page with functions exclusive to admins. Marked as sections 1, 2, and 3, these sections allow the admin to create new card collections and new awards. Admins simply enter the collector name and award name, then set the card's picture style and probability (the sum of

all probabilities must equal 1 for each collection) before clicking "SUBMIT" to create a new card collection and award.

2. **Admin Withdraws Balance from Smart Contract:** Only the Admin can withdraw savings from the smart contract (Figure 5). This function is crucial for Pyramid Cards to earn revenue and continue providing services to users.

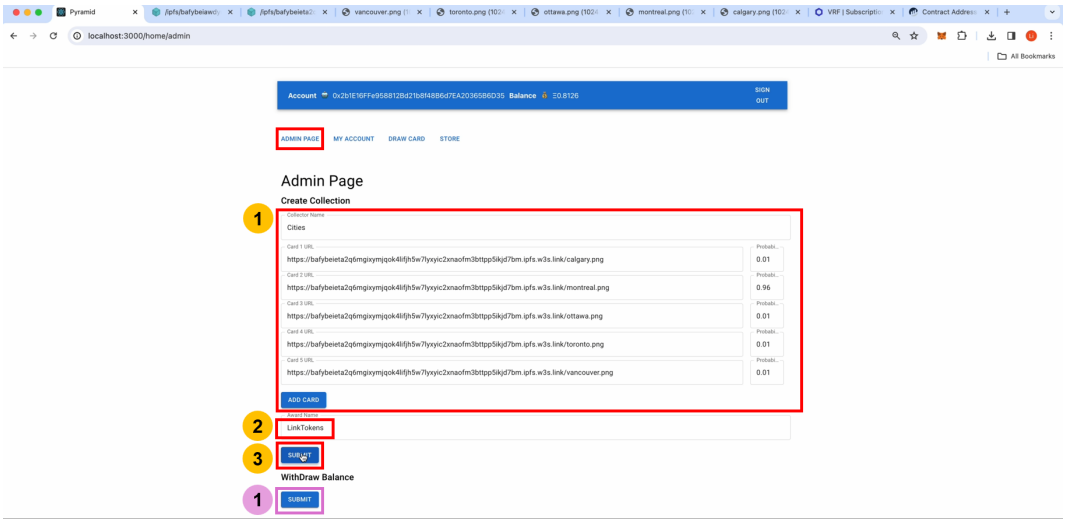

Figure 5: Admin Interface and Functions

- **User Interface and Experience**

1. **User Login:** Upon entering Pyramid Cards, users encounter the following Interface. Users need to click "LOGIN WITH METAMASK" to link their MetaMask wallet with the Pyramid Cards platform (Figure 6).

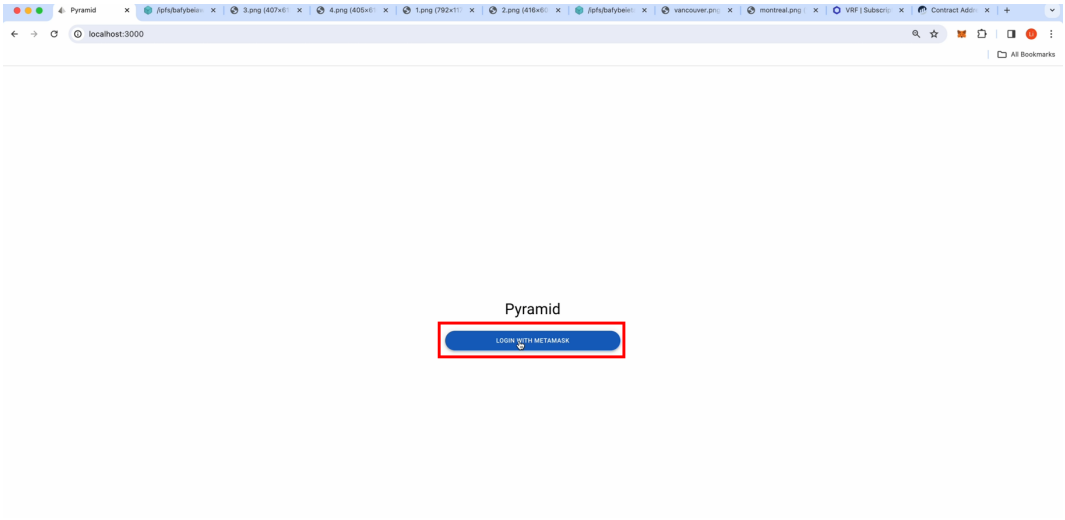

Figure 6: User Login

2. **User Buys Draws:** Welcome to Pyramid Cards, where users can do three things in this interface: in "MY ACCOUNT" check how many types of cards are collected in their account, go to "DRAW CARD" to start drawing their favorite cards, and visit "STORE" to view and redeem store awards. If it's the user's first visit to Pyramid Cards, we suggest going directly to "DRAW CARD" to purchase draw chances and enjoy collecting cards. As shown below (Figure 7), users just need to enter the number of draws they wish to buy in the "Quick Add Draws" section, then press "CONFIRM". At this point, the user's wallet will transact with the smart contract.

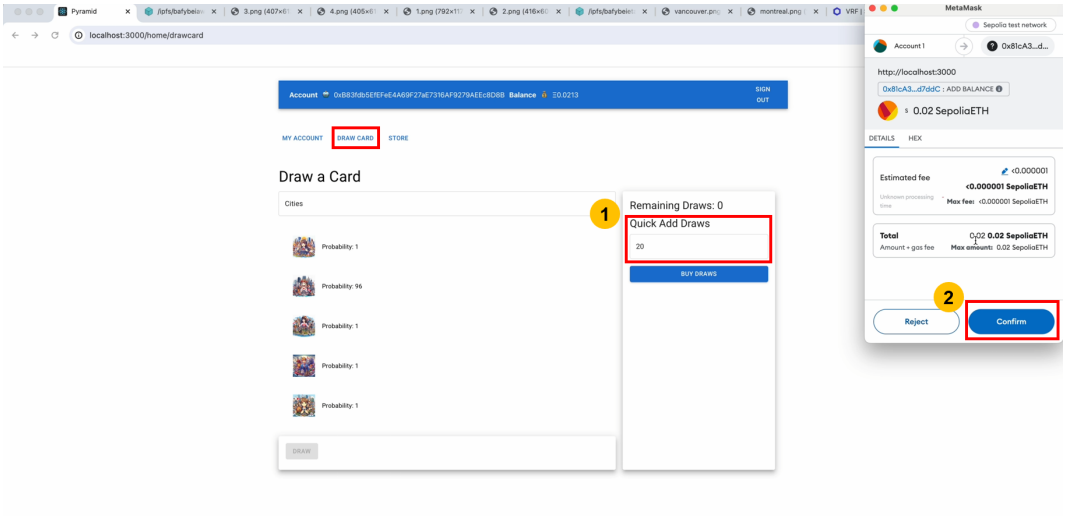

Figure 7: User Buys Draws

3. **User Draws a Card:** After the transaction is complete, the "Remaining draws" will display the number of card draws just purchased. Users can then freely choose their favorite card collection and, after selection, can see the probability of each card in the collection. Once the user selects a collection to draw from and clicks "DRAW" followed by "CONFIRM", they can start drawing a card (Figure 8). After successfully drawing a card, users can see which card they have drawn from the system prompt (Figure 9).

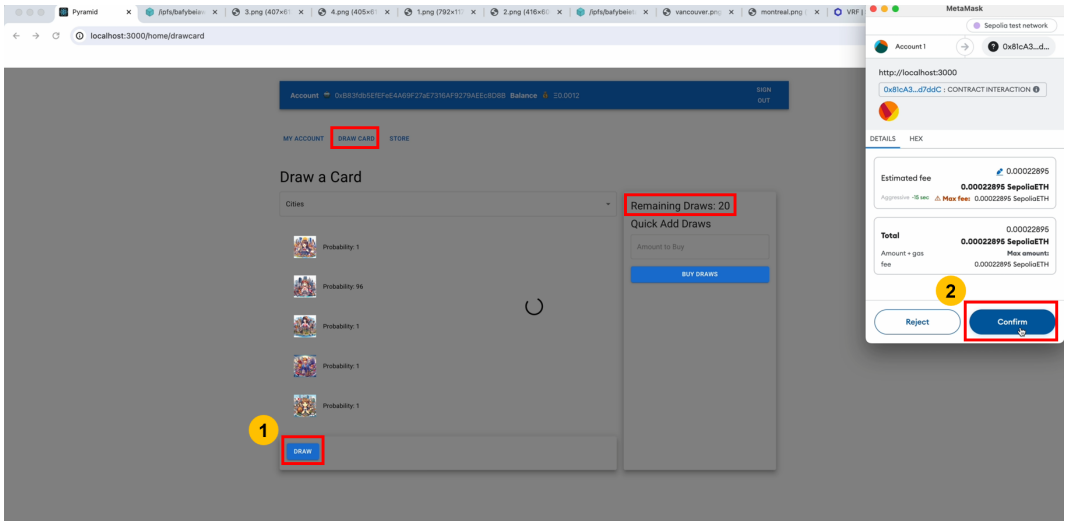

Figure 8: User Draws a Card

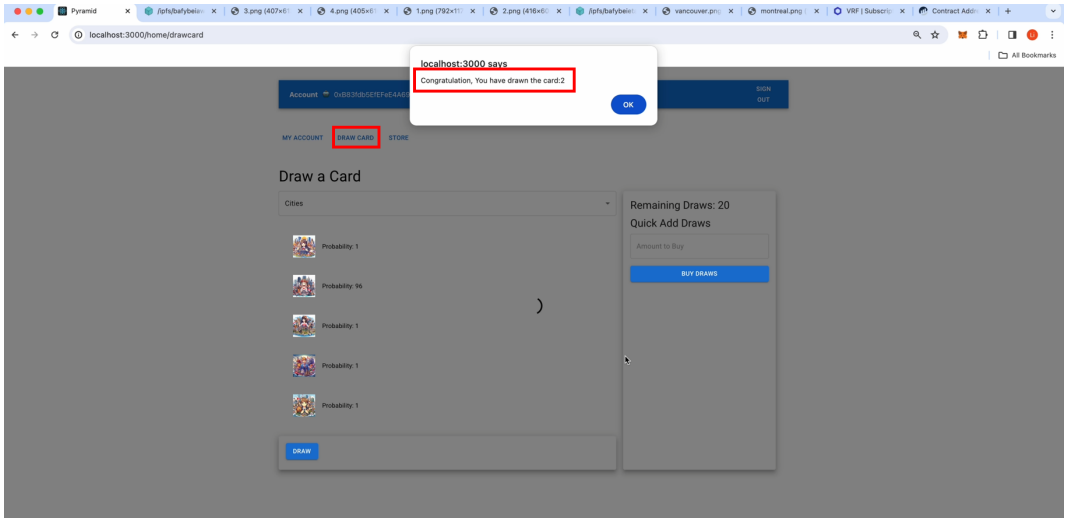

Figure 9: User Successfully Draws a Card

4. **User Redeems a Draw:** What if a user draws the same card many times? Do not need to worry, Pyramid Cards provides a "redeem a Draw" function for all users. If there are four same cards in the user's account, they can go to "MY ACCOUNT", and click "REDEEM" under that card category to exchange for a draw chance (Figure 10). After the exchange, users will see that the four same cards have been taken back by the system, and the number of draws will increase by one (Figure 11).

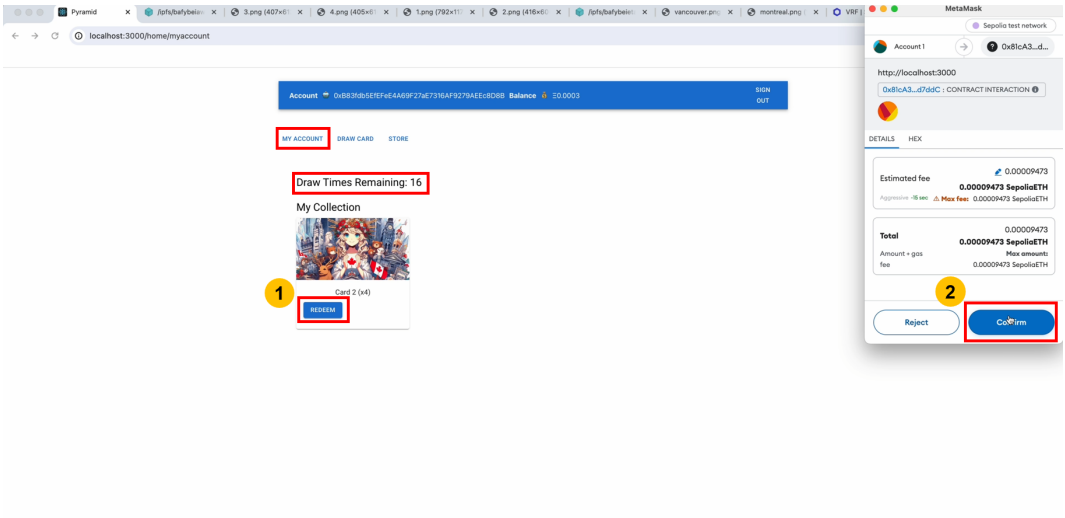

Figure 10: User Redeems a Draw

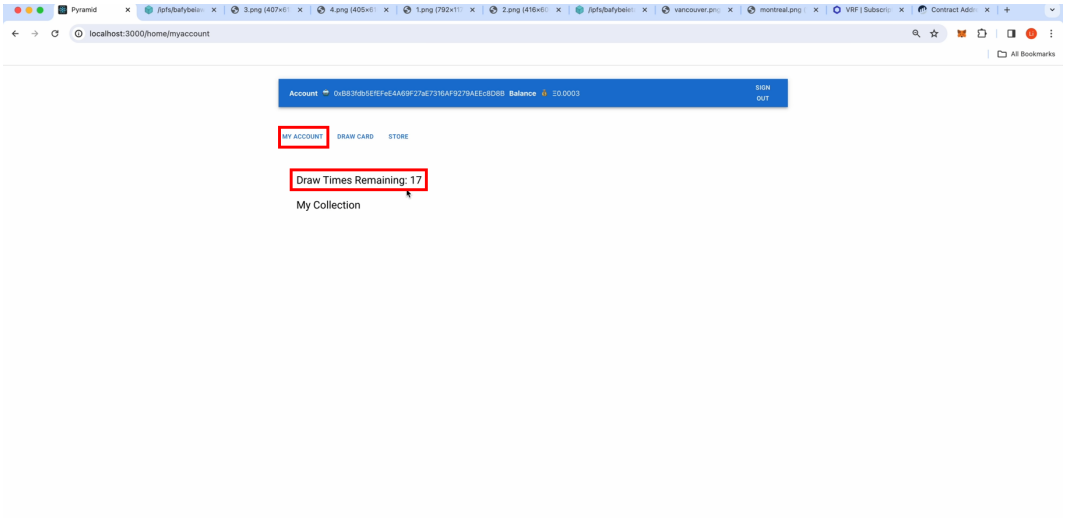

Figure 11: User's Inventory After Redeeming a Draw

5. **User Redeems a Store Award:** Congratulations, our lucky User has drawn a complete set of card collections (Figure 12)! At this point, the user can go to "STORE" to exchange for the corresponding award of that collection. Users need to click "REDEEM" on that award category and confirm to exchange a set of card collections for the corresponding award (Figure 13). After a successful exchange (Figure 14), users can go to "MY ACCOUNT" and find that the system has taken back that card collection (Figure 15).

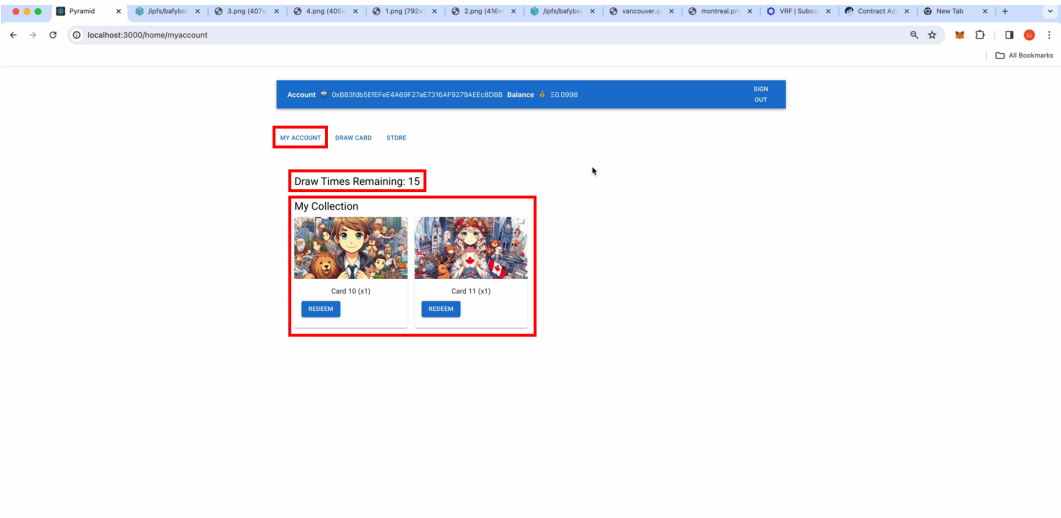

Figure 12: User's Inventory Before Redeeming a Store Award

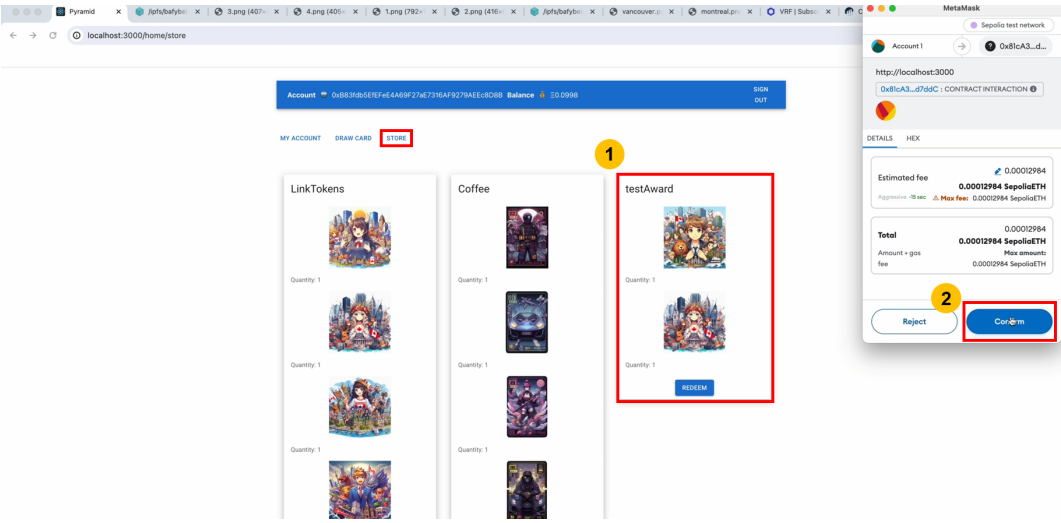

Figure 13: User Redeems a Store Award

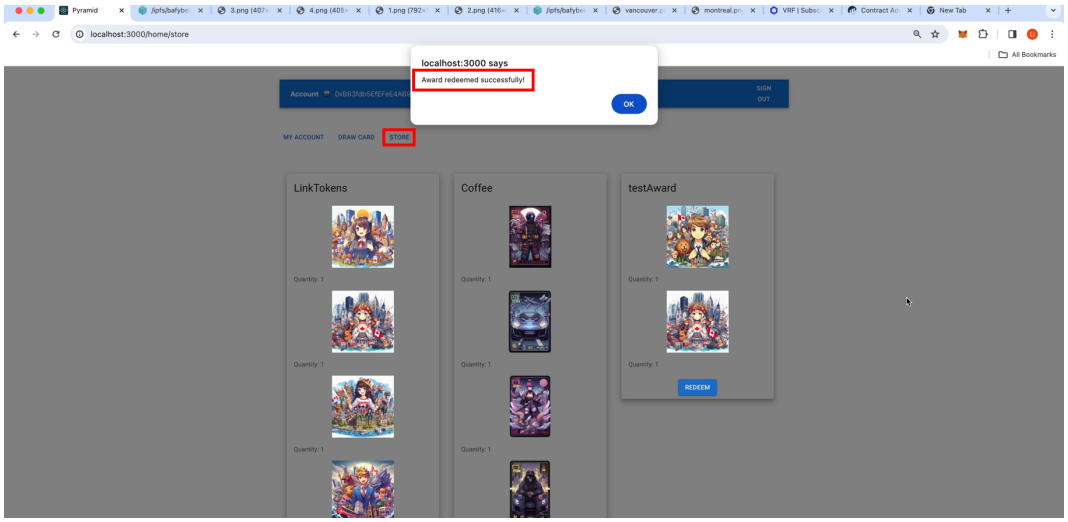

Figure 14: User Successfully Redeems a Store Award

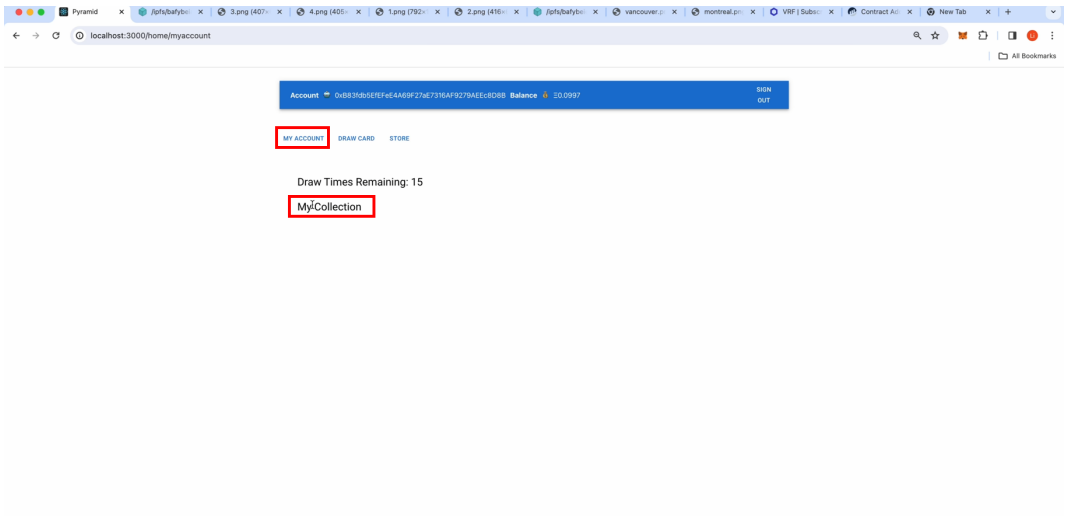

Figure 15: User's Inventory After Redeeming a Store Award

7 Discussion

7.1 Security Measures

Security is a top priority during our development process, and we have implemented a comprehensive set of measures to safeguard user data and ensure the integrity of transactions. Leveraging industry-recognized development environment Hardhat for smart contract development and testing, we conduct thorough audits to identify and address potential vulnerabilities before deployment. Utilizing the leading cryptocurrency wallet in the market Meta-

mask for user authentication, we ensure a secure gateway for users to access the Ethereum blockchain.

Key security features:

1. By integrating Redux into our DApp, we enhance security by centralizing state management, reducing the risk of data manipulation and unauthorized access. Redux enables us to implement strict access controls and enforce secure data flow patterns, bolstering protection against potential security threats. Furthermore, by maintaining a single source of truth for the application state, Redux facilitates easier auditing and monitoring, enabling swift detection and mitigation of security vulnerabilities.

2. Leveraging smart contracts enhances security by implementing tamper-resistant and transparent logic for critical operations, such as processing each transaction and managing user data. With smart contracts, cryptographic principles ensure verifiable and immutable execution of predefined rules, reducing the risk of fraud or manipulation. Moreover, the decentralized nature of smart contracts eliminates single points of failure, enhancing resilience against malicious attacks and unauthorized access.

3. With test-driven development practices in mind, our app is created with a proactive approach to security by systematically identifying and addressing vulnerabilities throughout the development lifecycle. By writing a comprehensive test suite, test-driven development methods help ensure that security considerations are integrated from the outset, minimizing the introduction of vulnerabilities during implementation. Continuous testing through test-driven development enables our developers to iteratively refine and validate security measures, leading to more robust and resilient software products.

7.2 Challenges and resolutions

- **Limitation of Array Returned in Smart Contracts:** When connecting the back-end and front-end, we encountered a limitation that our smart contract could only return a one-dimensional array to the front end, not an array of struct instances. To address this, a few functions were restructured to work within this constraint, maintaining effective communication of the smart contract with the front end.

- **Limit on Smart Contract File Size:** Another unforeseen challenge in deployment resulted from the 24kb file size limit for smart contracts. This forced us to modify our original functions to encapsulate all functionalities within a single contract due to the file size constraints.

- **Mapping Variables and Keys in Solidity:** Mapping variables in Solidity do not actually store keys; instead, they store hash identifiers of the keys, meaning we must pass them using an additional look-up table as an array to manually store and retrieve key values when the front end needs to iterate through these mappings.
- **Network Congestion and Transaction Delays:** The blockchain is time-consuming to interact with in some cases, specifically during deployment, transactions, and VRF subscriptions. Network congestion sometimes led to response times of 5-10 minutes, which is poor for testing and can adversely impact user experience. Although increasing the gas fee was a partial solution, the issue largely stems from the inherent mechanism of blockchain technology.
- **Interacting with Smart Contracts from the Front End:** We learned how to interact with Smart Contracts using transaction functions and parameters when it involves state-changing calls that require gas fees, whereas calling getter/view functions only requires a call function.
- **Project Setup and Dependency Management:** Project setup, dependency management, and environment configuration all led to substantial efforts that demanded careful attention to yarn and npm dependency management.

8 Conclusion

The Pyramid Cards has been successful in proving the innovative use of blockchain technology and how it can create a fair, transparent, and exciting online card collection platform. The implementation of smart contracts and Chainlink's Verifiable Random Function has enabled us to make every card draw transparent, verified, and immutable. This has not only guaranteed that every card draw made is trustworthy, but it has as well built a trustworthy relationship between the platform and all its users.

to manually store and retrieve key valuesThroughout the whole project, our team has learned a lot of experience and adaptability. We experienced many challenges from technical difficulties like integrating smart contracts with a user-friendly front end to issues, such as ensuring the complete transparency and fairness of the draw-card process. These experiences have been very valuable because they give us a better insight into the potential and risks of blockchain technology in application development. We understand how critical it is to test thoroughly, spotlight the user, and be flexible to change.

For future development, the platform still has room for further expansion and improvement. Integrating more blockchain features such as NFTs to develop individual card collections will expand the platform's capacities. Extending the card design and reward mechanisms would retain existing users and broaden the user community. Moreover, partnerships with brand owners or artists can give rise to unique content that can focus on attracting more users and clients. On the technical side, ongoing efforts to optimize transaction speeds are essential for maintaining a seamless user experience.

The Pyramid Cards project establishes an important step in the utilization of blockchain technology in developing online gaming platforms. It opens up the possibility of a future whereby digital interactions are 100% transparent and fair, increasing the user experience to new heights. As proceeding to develop our platform, we hope to use technology to build enjoyable, fair, and safe digital environments for users across the world.

References

[1] Finance Magnates. Crypto gaming: The fusion of cryptocurrencies and the gaming industry, 2022. [Online; accessed 20-Mar-2024].

[2] IMARC Group. Online lottery market report by product type, 2024. Accessed: March 22, 2024.

[3] Information Age - ACS. Next-gen lotteries may run on blockchain, 2019. [Online; accessed 20-Mar-2024].

[4] Priori Data. Axie infinity users - priori data, 2022. [Online; accessed 20-Mar-2024].

Project Repository

The GitHub link to the project repository is: `https://github.com/Will-Li-zw/571G`