# NFT Card Game Documentation

## Abstract

Our NFT-based card game, Rondo, is built on the AVAX blockchain using Solidity. NFT-based card games represent a significant innovation in the gaming industry, combining the unique attributes of non-fungible tokens (NFTs) with strategic gameplay. Unlike traditional digital card games, where assets are centrally controlled and owned by the game developer, NFT-based card games allow players to truly own, trade, and sell their digital cards. These cards are verifiably scarce and unique, thanks to blockchain technology, enhancing the sense of ownership and investment for players.

Rondo leverages the ERC1155 standard for token management and includes functionalities for player registration, token creation, and battle mechanics. This document provides a comprehensive overview of the smart contract, including its structure, functions, and interaction logic. Additionally, it covers the front-end and back-end development aspects, ensuring a comprehensive understanding of the entire project.

## Introduction

The Rondo NFT card game is designed to bring a novel and engaging gaming experience to the blockchain ecosystem. By utilizing non-fungible tokens (NFTs), players can own unique digital assets that represent different game characters with distinct attributes. The game combines strategic gameplay with the benefits of blockchain technology, such as transparency, security, and true ownership of in-game assets.

Rondo aims to create a fully decentralized and player-driven environment where every in-game action is recorded on the blockchain, ensuring transparency and fairness. The game's unique characters, each represented by an NFT, offer players a sense of ownership and investment in the game. This combination of blockchain technology and engaging gameplay mechanics distinguishes Rondo from traditional card games.

The project includes both front-end and back-end components. The front end is responsible for providing a user-friendly interface for players to interact with the game. It is developed using modern web technologies like React, ensuring responsiveness and an intuitive user experience. The back end, powered by a smart contract written in Solidity, ensures the integrity and security of the game logic and data. The smart contract handles critical operations such as player registration, token minting, and battle resolution, leveraging the ERC1155 token standard to manage multiple token types efficiently.

By using decentralized storage solutions like IPFS for storing metadata, Rondo ensures that the information remains tamper-proof and accessible. This document provides an in-depth look into the development process, including the smart contract's architecture, the technologies used in the front and back end, and the role of NFTs in the game.

# Methodology

The development of Rondo involves several key components:

## Smart Contract Development

The smart contract, written in Solidity, is the backbone of the Rondo game. It manages player registration, token creation, and battle mechanics. The contract is based on the ERC1155 standard and incorporates additional functionalities for game-specific requirements.

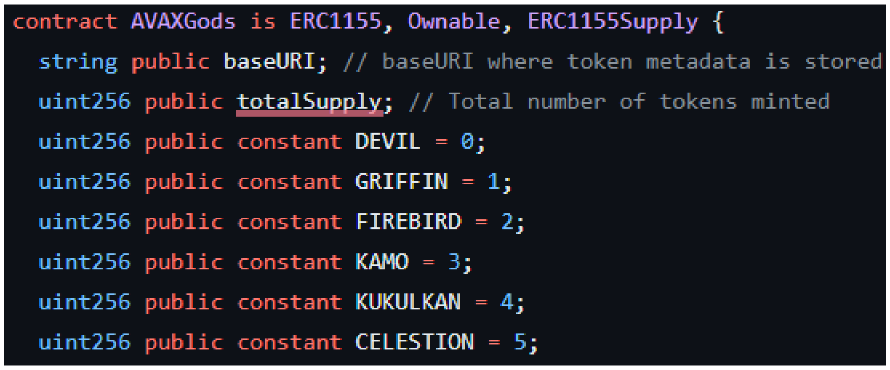
1. Imports and Inheritance: The contract imports OpenZeppelin libraries for ERC1155 token standards, ownership management, and supply tracking. It inherits from `ERC1155`, `Ownable`, and `ERC1155Supply` to utilize these functionalities.

2. State Variables: Key state variables include:

- `baseURI`: Stores the base URI for token metadata.

- `totalSupply`: Tracks the total number of tokens minted.

- Constants representing different game tokens and their maximum strengths, such as `DEVIL`, `GRIFFIN`, `FIREBIRD`, `KAMO`, `KUKULKAN`, and `CELESTION`.



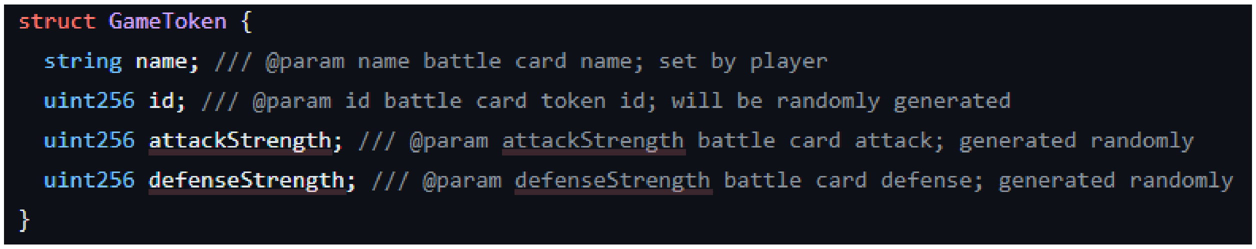
```
contract AVAXGods is ERC1155, Ownable, ERC1155Supply {
  string public baseURI; // baseURI where token metadata is stored
  uint256 public totalSupply; // Total number of tokens minted
  uint256 public constant DEVIL = 0;
  uint256 public constant GRIFFIN = 1;
  uint256 public constant FIREBIRD = 2;
  uint256 public constant KAMO = 3;
  uint256 public constant KUKULKAN = 4;
  uint256 public constant CELESTION = 5;
```

3. Structs:

- `GameToken`: Represents the attributes of a game token, including name, ID, attack strength, and defense strength.

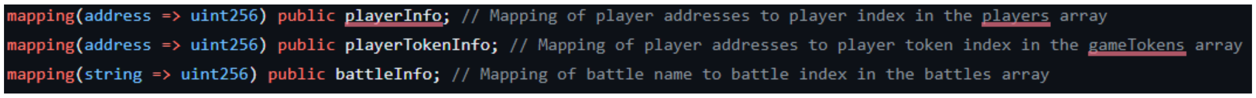
- `Player`: Stores player information such as address, name, mana, health, and battle status.

- `Battle`: Maintains the details of a battle, including its status, players, moves, and winner.



```
struct GameToken {
  string name; /// @param name battle card name; set by player
  uint256 id; /// @param id battle card token id; will be randomly generated
  uint256 attackStrength; /// @param attackStrength battle card attack; generated randomly
  uint256 defenseStrength; /// @param defenseStrength battle card defense; generated randomly
}
```

4. Mappings: Various mappings are used to link player addresses to their information, tokens, and battles. For instance, `players` map player addresses to their `Player` struct, `playerTokens` map player addresses to their tokens, and `battles` map battle IDs to their `Battle` struct.



```
mapping(address => uint256) public playerInfo; // Mapping of player addresses to player index in the players array
mapping(address => uint256) public playerTokenInfo; // Mapping of player addresses to player token index in the gameTokens array
mapping(string => uint256) public battleInfo; // Mapping of battle name to battle index in the battles array
```

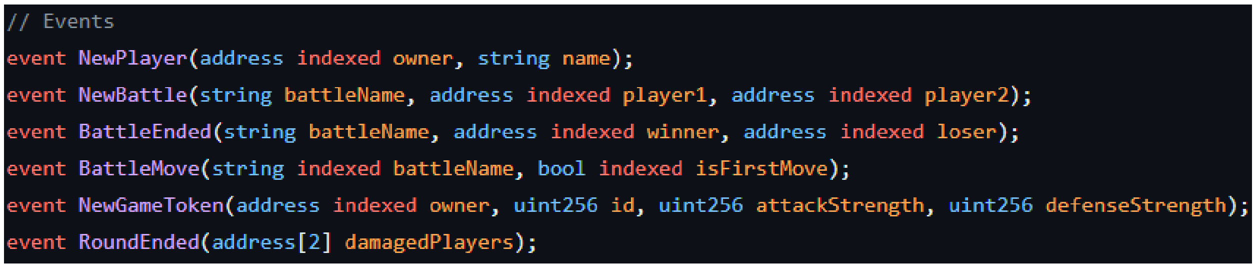
5. Functions:

- Player Functions**: Functions to register players (`registerPlayer`), retrieve player information (`getPlayer`), and manage player tokens (`mintToken`).

- Battle Functions: Functions to create and join battles (`createBattle`, `joinBattle`), register moves (`registerMove`), and resolve battles (`resolveBattle`).

- Helper Functions: Internal functions to generate random numbers (`_random`) and manage battle logic (`_calculateBattleOutcome`).

6. Events: Several events are defined to log key actions within the game, such as `PlayerRegistered`, `BattleCreated`, `BattleJoined`, and `MoveRegistered`.



```
// Events
event NewPlayer(address indexed owner, string name);
event NewBattle(string battleName, address indexed player1, address indexed player2);
event BattleEnded(string battleName, address indexed winner, address indexed loser);
event BattleMove(string indexed battleName, bool indexed isFirstMove);
event NewGameToken(address indexed owner, uint256 id, uint256 attackStrength, uint256 defenseStrength);
event RoundEnded(address[2] damagedPlayers);
```

## Front-End Development

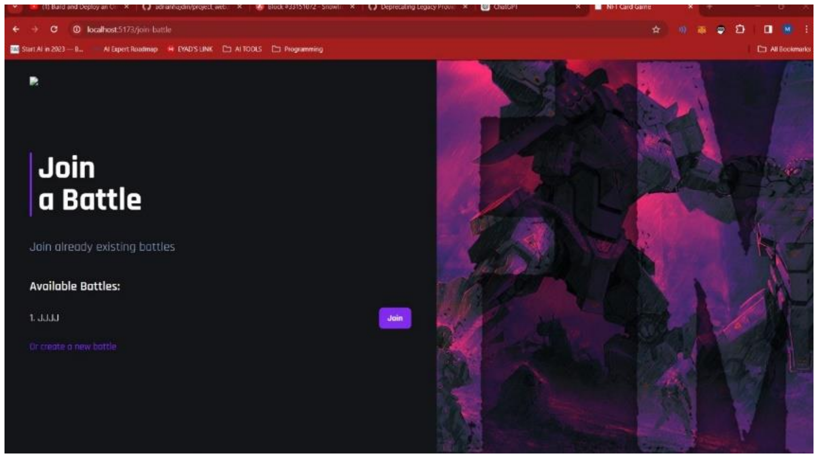
The front-end application is built with modern web technologies such as HTML, CSS, and JavaScript, with React providing a dynamic and responsive user interface. This ensures an intuitive interface for players to interact with the game seamlessly. Players can register and log in to the game using their Ethereum wallet, view and manage their NFT cards, initiate and participate in battles, and monitor game progress and outcomes. Real-time updates on battle status and results are provided, enhancing the overall gaming experience. The front end also handles transaction management, providing feedback to users about the status of their transactions on the blockchain, enabling users to:

- Register and log in to the game: Players can create an account and log in using their Ethereum wallet.

- View and manage their NFT cards: The front end displays the player's collection of NFT cards with their attributes.

- Initiate and participate in battles: Players can create new battles or join existing ones.

- Monitor game progress and outcomes: Real-time updates on the battle status and results are provided.



## Back-End Development

In the back end, using the blockchain ensures transparency and fairness, as all game actions are recorded on an immutable ledger. This eliminates the possibility of cheating and guarantees that players have true ownership of their NFT cards. Additionally, using decentralized storage solutions like IPFS for metadata ensures that the information remains tamper-proof and accessible, powered by the Ethereum blockchain, which is responsible for executing the game logic and storing data securely. The smart contract handles all critical operations, including:

- Player registration and management: Ensuring unique player registrations and maintaining player statistics.

- Token minting and attributes assignment: Minting new NFT cards with unique attributes when players register or earn them.

- Battle creation, management, and resolution: Managing the lifecycle of battles from creation to resolution, ensuring fairness and transparency.

## NFTs in Rondo

Non-fungible tokens (NFTs) play a crucial role in Rondo. Each game card is an NFT that represents a unique character with specific attributes, such as attack and defense strengths. The use of ERC1155 tokens allows for efficient management of multiple token types within the same contract.

- Minting: New NFTs are minted when players register or earn new cards. The minting process includes generating unique attributes for each token.

- Metadata: Each NFT has associated metadata, stored off-chain, that includes the character's name, image, and attributes. This metadata is linked to the token via the baseURI.

- Ownership: NFTs are owned by players and can be traded or sold on secondary markets. Ownership is tracked on the blockchain, ensuring transparency and security.

- Regarding NFTs in Rondo, each game card is represented as an NFT, crucial for character uniqueness and specific attribute representation. Solidity facilitates NFT functionality within the contract, enabling the minting of new NFTs upon player registration or earning. Metadata associated with each NFT, including name, image, and attributes, is stored off-chain and linked to the token via the baseURI. Players own these NFTs, which can be traded or sold on secondary markets, with ownership tracked securely on the blockchain.

## Discussion

The smart contract is designed to provide a seamless gaming experience while ensuring security and fairness. Key aspects discussed include:

- Randomness: The use of keccak256 hashing to generate random numbers for token attributes and battle outcomes. This ensures unpredictability and fairness in the game.

- Battle Logic: The implementation of battle mechanics, where players can choose to attack or defend, and the contract determines the winner based on their moves and token strengths. The battle logic ensures that each game is strategic and engaging.

- Player Interaction: Ensuring that players cannot manipulate the game by enforcing rules such as not allowing multiple registrations or joining multiple battles simultaneously. This maintains the integrity of the game.

- Scalability: The use of the ERC1155 standard allows for efficient management of multiple token types, making the game scalable for a large number of players and tokens. This ensures that the game can grow and accommodate more players and features over time.

## Conclusion

In conclusion, the Rondo smart contract represents a significant advancement in the realm of NFT-based gaming, offering players a compelling and secure gaming experience. By leveraging OpenZeppelin's libraries and adhering to best practices in smart contract development, Rondo ensures a fair and enjoyable experience for all players. The integration of blockchain technology provides transparency, security, and true ownership of in-game assets, setting a new standard in the gaming industry. This comprehensive documentation elucidates the contract's architecture, front-end and back-end development, and the crucial role of NFTs in shaping Rondo's ecosystem. Serving as a valuable resource for developers and stakeholders, it facilitates understanding of the implementation and potential for future enhancements. Rondo not only delivers on its promise of an engaging gaming experience but also pioneers innovation in the burgeoning NFT gaming space. As the project evolves, this documentation will continue to guide efforts in enhancing and refining the Rondo gaming experience for players worldwide.