Sheep Game Analysis

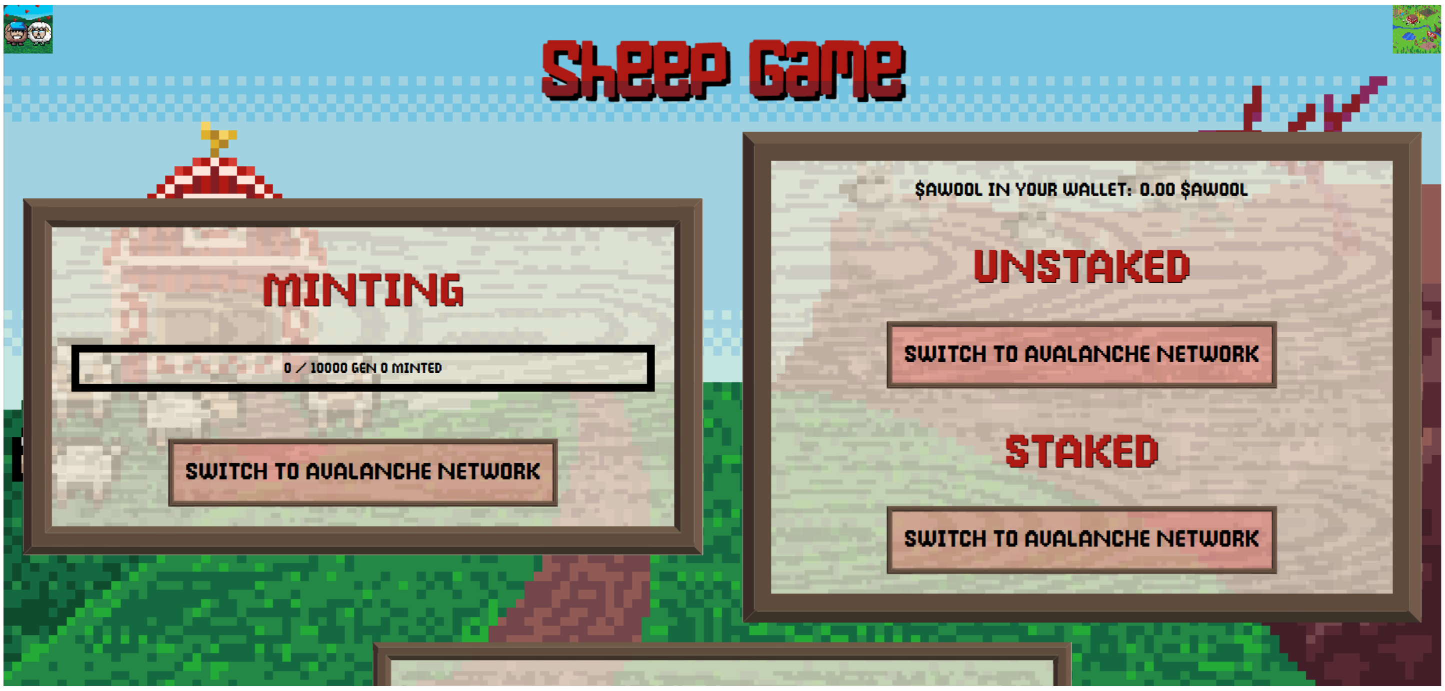
Ayush Singh Rathore

[ayyush1738 (Ayush Singh Rathore) (github.com)](github.com)

1. **Introduction:**

I am the best fit for this role because I have relevant experience in the domain of Web3.0. I have worked on various chains and different wallets in past, I am very much fascinated with the concept of DE-FI. My first project which I built in the ETH-India hackathon was a DE-FI platform which uses credit based informal lending and it won the polygon prize track in 2022.

Coming to the challenge which was shared to me I really liked the idea of the project, to run the project I have switched my node version from latest to 16.20.0. I can see a "Sheep Game" with the functionality of minting and staking the "Woolf" NFT on Avalanche Blockchain, it is an interesting pay-to-earn game.


Sheep Game
MINTING
0 / 10000 GEN 0 MINTED
SWITCH TO AVALANCHE NETWORK
$AWOOL IN YOUR WALLET: 0.00 $AWOOL
UNSTAKED
SWITCH TO AVALANCHE NETWORK
STAKED
SWITCH TO AVALANCHE NETWORK

Software Requirements Specification:

1.1 Purpose

The purpose of Sheep Game is to provide a decentralized gaming experience on the Avalanche blockchain, involving interactions between ERC-20 and ERC-721 tokens.

1.2 Scope

Sheep Game allows players to mint and manage Sheep (ERC-721 NFTs) and Wolves, participate in staking, and engage in token interactions such as $WOOL earnings and taxes.

1.3 Definitions, Acronyms, and Abbreviations

- **NFT**: Non-Fungible Token
- **ERC-20**: Ethereum Request for Comments 20 (Token Standard)
- **ERC-721**: Ethereum Request for Comments 721 (NFT Token Standard)
- **$WOOL**: Token used within Sheep Game for staking, earnings, and taxes
- **AVAX**: Avalanche native token

2. Overall Description

2.1 Product Perspective

Sheep Game operates as a decentralized application (DApp) on the Avalanche blockchain, interacting with smart contracts for minting, staking, and economic activities involving $WOOL tokens.

2.2 Product Functions

- **Minting**: Players can mint Sheep and Wolves using AVAX or $WOOL, each with unique traits and properties.
- **Staking**: Sheep can be staked in the Barn to earn $WOOL tokens, subject to risks of Wolves attempting to steal accumulated earnings.
- **Taxation**: Wolves earn a portion of taxed $WOOL from Sheep activities based on their Alpha scores.
- **Interactions**: Sheep can claim $WOOL earnings and Wolves can stake, claim taxes, and attempt to steal newly minted NFTs.

2.3 User Classes and Characteristics

- **Players**: Individuals participating in Sheep Game, minting and managing Sheep and Wolves.
- **Developers**: Responsible for managing the smart contracts, game mechanics, and community engagement.

3. Specific Requirements

3.1 Functional Requirements

- **Minting Requirements**:
  - Players can mint Gen 0 Sheep using AVAX, TRACTOR, or JOE.
  - Minting costs are specified based on token ID ranges.
- **Staking Requirements**:
  - Sheep can be staked in the Barn to accumulate $WOOL.
  - Wolves attempt to steal accumulated $WOOL if Sheep are unstaked.
- **Taxation and Earnings**:
  - Wolves earn a share of taxed $WOOL based on their Alpha scores.
  - Sheep receive 80% of claimed $WOOL, with 20% taxed if left in the Barn.
- **Interactions**:
  - Wolves can stake, claim taxes, and attempt to steal newly minted Sheep or Wolves based on their Alpha scores.

3.2 Non-Functional Requirements

- **Performance**: Transactions should have low fees and sub-second finality on the Avalanche blockchain.
- **Security**: Smart contracts must be secure against vulnerabilities such as reentrancy attacks.
- **Scalability**: The game should handle a large number of transactions and users seamlessly.

3.3 External Interface Requirements

- **Contract Addresses**: Provided for Sheep/Wolf NFT, Barn/Gang Staking, and $WOOL token.
- **APIs**: Interfaces with Avalanche blockchain for transaction processing and data retrieval.

4. System Features

4.1 Feature 1: Minting

- **Description**: Allows players to mint Sheep and Wolves using specified tokens.
- **Priority**: High
- **Dependencies**: Contract deployment and token availability.

4.2 Feature 2: Staking and Earnings

- **Description**: Sheep can be staked in the Barn to earn $WOOL tokens.
- **Priority**: High
- **Dependencies**: Contract integration and token economics.

4.3 Feature 3: Taxation and Wolves

- **Description**: Wolves earn a portion of taxed $WOOL based on their Alpha scores.
- **Priority**: Medium

- **Dependencies**: Smart contract implementation and alpha score calculations.

5. Other Non-Functional Requirements

5.1 Documentation Requirements

- **User Documentation**: Guidelines for players on how to interact with Sheep Game.
- **Technical Documentation**: Comprehensive documentation for developers on smart contract functionalities and API interactions.

5.2 Legal and Regulatory Requirements

- **Compliance**: Ensure adherence to blockchain regulations and guidelines.

6. Appendices

6.1 Glossary

- Definitions of key terms and concepts used within the document.

6.2 References

- Links to relevant resources, such as contract addresses, developer documentation, and community channels.

upgrading node version 16.20.0 to 18+ with its dependencies:

Step 1: Install Node Version Manager (nvm)

Node Version Manager (nvm) is a tool that allows you to manage multiple Node.js versions on your machine.

1. **Install nvm using the installer script provided:**
2. **bash**
3. **command: curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash**
4. **Verify nvm installation: command -v nvm**
5. **You should see nvm as the output.**

Step 2: Install Node.js 18

1. **List the available Node.js versions:**
2. **bash**

3. **nvm ls-remote**
4. **Install Node.js 18: nvm install 18**
5. **Set Node.js 18 as the default version: nvm alias default 18**
6. **Verify the installation: node --version**

Step 3: Update Yarn

Ensure you have the latest Yarn version (Yarn 1.x or Yarn 2, depending on your preference).

Step 4: Update Project Dependencies

1. **Navigate to your React project directory: cd /path/to your /react/project**
2. **Update your project's dependencies : yarn install --check-files**

Analysis of current features:

Web3 Integration

SheepGame integrates with Web3 technology, enabling decentralized interactions through smart contracts on the Avalanche blockchain.

Blockchain: Avalanche

- Blockchain Choice: SheepGame operates on the Avalanche blockchain, leveraging its advantages such as low transaction fees, sub-second finality, and high throughput. These features are crucial for the game's real-time interactions and scalability.

Smart Contracts

- **ERC-721 and ERC-20 Standards:** Sheep Game utilizes ERC-721 for Sheep and Wolves, representing unique NFTs with distinct traits and behaviors. ERC-20 is employed for $WOOL tokens, serving as the in-game currency for staking, rewards, and taxation.
- **Functionality:**
  - ERC-721 (Sheep/Wolves): Each Sheep and Wolf is minted as an ERC-721 token, ensuring uniqueness and ownership verification on the blockchain. Smart contracts manage their properties and interactions, including staking and tax distributions.
  - ERC-20 ($WOOL): $WOOL tokens adhere to the ERC-20 standard, enabling fungibility and easy integration with decentralized exchanges (DEXs) and external wallets. Smart contracts govern $WOOL minting, staking rewards, and economic mechanisms like taxation.

Game Mechanics

- **Staking and Barn Mechanism:** Smart contracts manage staking functionalities where Sheep can earn $WOOL tokens based on their participation duration. Wolves, based on

their Alpha scores, participate in tax collections and potentially attempt to steal unstaked $WOOL.
- **Taxation and Alpha Scores:** Wolves with higher Alpha scores receive a larger share of taxed $WOOL, incentivizing higher participation and strategic gameplay. This mechanism is implemented via smart contracts to ensure fairness and transparency in reward distribution.

Security and Scalability

- **Smart Contract Security:** Sheep Game prioritizes smart contract security to mitigate risks such as reentrancy attacks and ensure robustness against potential vulnerabilities.
- **Scalability:** Utilizing Avalanche's consensus mechanism (e.g., Avalanche consensus), Sheep Game achieves high scalability, accommodating a large number of transactions per second and supporting a growing user base without compromising performance.

New features that could be added:

1. Sheep Breeding and Genetics

- Description: Allow players to breed their Sheep NFTs to create new offspring with combined traits. By Implementing genetic algorithms to introduce randomness and uniqueness in offspring traits.
- Benefits: Encourages long-term engagement, adds depth to gameplay through genetic strategy, and fosters a secondary market for rare genetic traits.

2. Virtual Lands

- Firstly, the concept of virtual lands can be introduced according to my analysis, which can be functioned as a booster for the players. The limited genesis land plots could be existing, each completely unique. Players can use the land to cultivate assets.

- Then, players can use farmers to boost the amount they collect on a land plot. Farmers will also be useful to speed up sheep breeding as well as protect the sheep and their offspring from the wolves.

3. DeFi Integration

- Description: Introduce decentralized finance (DeFi) functionalities such as liquidity pools for $WOOL tokens, yield farming opportunities linked to Sheep staking, or NFT-backed loans using Sheep/Wolf NFTs as collateral.
- Benefits: Expands the utility of $WOOL tokens beyond gaming, attracts DeFi enthusiasts to the ecosystem, and introduces additional revenue streams for players.