SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF COMPUTER SCIENCE

AncientSociety

BLOCKCHAIN AND DISTRIBUTED LEDGER TECHNOLOGIES

Professors:
Claudio Di Ciccio

Students:
Lorenzo Ciarpaglini

Academic Year 2023/2024

Contents

1 Introduction

1.1 Presentation of the DApp

AncientSociety is a web browser game built on blockchain technology. The game's objective is to build and expand an ancient city by leveling up buildings, trading with other users, engaging in activities through tools, and earning tradable tokens through liquidity pools. It aims to merge traditional infrastructure seen in today's Web 2.0 games with the capabilities of blockchain technology, thus granting the game unique features. These include the ability to purchase NFTs on a decentralized marketplace that supports multiple currencies and chains, staking these NFTs to earn tokens and enhance city activities, and increasing the liquidity pool to encourage the use of the game's token.

The use of blockchain technology fosters player trust by enabling the acquisition of genuine assets that can be traded outside the game, in addition to tradable tokens that can be exchanged for other cryptocurrencies. The game also incorporates various mechanisms to deflate the token, thereby maintaining its higher value.

This project is a reimagining of a game I personally developed between 2022 and 2023, which is still playable at the following link: https://www.ancientsociety.io/game.

1.2 Overview

- **Background:** An introduction to the fundamental concepts upon which AncientSociety is based is provided in Section 2.
- **Presentation of the Context:** This part delineates the game's objectives and offers a clear explanation, as detailed in Section 3.
- **Software Architecture:** Described in Section 4, this section outlines the software architecture, gives an overview of the game components, examines each component in detail, and reflects on the design choices made.
- **Implementation:** The front-end presentation and the tools used are elaborated in Section 5.
- **Known Issues and Limitations:** Discussions on the limitations and identified issues within the protocol are presented in Section 6, highlighting the constraints and areas for improvement.

1.3 Team presentation

My team consists solely of myself, Lorenzo Ciarpaglini. Initially, my goal was to understand which Web 2.0 architecture to use as a baseline, onto which I could then

integrate blockchain technology. Secondly, I focused on determining which smart contracts were most suitable for the game, and more importantly, which blockchain was best suited to provide adequate playability for the players. Finally, I designed the entire infrastructure, including the front-end, back-end, database, and blockchain interaction.

2 Background

2.1 Blockchain

Blockchain technology [1] represents a revolutionary approach to securely and transparently conducting transactions without the need for centralized authority. It is fundamentally a distributed ledger that records transactions across a network of computers, ensuring that each transaction is immutable and transparent to all participants. This technology operates on the principles of cryptography, ensuring secure transactions, and a consensus mechanism, which validates transactions and adds them to the blockchain.

The essence of blockchain lies in its ability to provide a secure, immutable, and decentralized record of transactions. Each block in the chain contains a number of transactions, and every new block links to the previous block, forming a chain of blocks or blockchain. The security of these transactions is guaranteed through the use of cryptographic hashing and digital signatures, making it nearly impossible to alter historical data without the consensus of the network.

Blockchain technology is not just about recording financial transactions but is a versatile platform that supports a wide range of applications, including supply chain management, digital voting, and decentralized finance (DeFi). The introduction of smart contracts, self-executing contracts with the terms of the agreement directly written into lines of code, further extends the capabilities of blockchain, enabling automated, secure, and tamper-proof transactions.

The distributed nature of blockchain means that it does not rely on a single point of failure, enhancing the security and robustness of the system. Transactions on a blockchain are validated by a network of nodes through consensus mechanisms such as Proof of Work (PoW) or Proof of Stake (PoS), ensuring that each transaction is verified and recorded without bias.

Key Characteristics of Blockchain Technology:

- **Distributed:** The ledger is maintained across multiple nodes, ensuring no single point of control.
- **Immutable:** Once recorded, the data in any given block cannot be altered without affecting the entire chain.
- **Secured:** Cryptography ensures the security of transactions and the privacy of participants.
- **Decentralized:** Operates without a central authority, enhancing trust and security.

In summary, blockchain technology offers a new paradigm for transactional and data management systems, characterized by decentralization, immutability, security,

and transparency. Its potential applications are vast and varied, promising to revolutionize numerous industries by providing a secure, efficient, and transparent method for conducting transactions.

2.1.1 Consensus

Consensus mechanisms [2] are essential to the operation of blockchain technologies, underpinning the process through which transactions are verified and new blocks are added to the blockchain. These algorithms facilitate a decentralized, fault-tolerant system that ensures the security and integrity of the blockchain. Among the plethora of consensus algorithms, Proof of Work (PoW) and Proof of Stake (PoS) stand out due to their distinct methodologies and implementations.

Proof of Work (PoW) is the seminal consensus algorithm employed by Bitcoin, mandating that nodes, referred to as miners, engage in solving computationally demanding puzzles. This process, which consumes a considerable amount of energy, serves to validate transactions and mint new blocks, thereby securing the network. Miners are incentivized through rewards, aligning their interests with the network's overall security. The robustness of PoW lies in its deterrence against alterations of the blockchain: modifying past transactions would necessitate redoing the work of the subsequent blocks, a feat that is computationally infeasible.

Proof of Stake (PoS) [3], on the other hand, offers a less energy-intensive alternative by allocating block validation rights based on the stake or ownership of the cryptocurrency within the network. Validators are selected based on their stake, with larger stakes increasing the chances of being chosen for block validation. This method not only conserves energy but also enables quicker transaction validations compared to PoW. However, it introduces a level of centralization, as those holding more significant stakes wield greater influence over the validation process.

Beyond PoW and PoS, the blockchain ecosystem has evolved to include other consensus algorithms [4], each addressing specific needs.

The selection of a consensus algorithm is pivotal, influencing a blockchain's security, efficiency, and degree of decentralization. While PoW is revered for its security measures, PoS is celebrated for its sustainability and scalability, underscoring the blockchain community's commitment to innovation and the improvement of consensus mechanisms. The trade-offs inherent in each consensus model fuel ongoing research aimed at refining these technologies to ensure they remain secure, decentralized, and adaptable to a myriad of applications.

2.1.2 Ethereum

Ethereum [5] signifies a departure from Bitcoin's transaction-centric model to a more flexible Account (or Balance) Model, moving away from transaction-based exchanges

to direct value transfers. Central to this ecosystem is the Ethereum Virtual Machine (EVM), which facilitates the standardized execution of smart contracts across all nodes. This standardization ensures compatibility and provides a reliable environment for executing smart contracts, which are pieces of code that automate the execution of agreements. These contracts are executed at a cost quantified in gas, a metric that remains stable irrespective of Ether's market price, ensuring that the computational efforts of the nodes are appropriately compensated.

The advent of smart contracts has elevated Ethereum beyond a mere platform for cryptocurrency, enabling a wide array of decentralized applications (dApps). Ethereum meticulously tracks the balance of each account, allowing for straightforward exchanges of value. The execution of smart contracts incurs a cost, calculated in gas, to compensate for the computational resources utilized by the network's nodes.

Originally, Ethereum employed a Proof-of-Work (PoW) consensus mechanism, akin to Bitcoin, but differentiated by unique aspects such as the Ethash algorithm and the use of gas to manage and regulate the execution of smart contracts. However, the escalating difficulty of PoW computations highlighted concerns regarding scalability and environmental impact.

A significant evolution in Ethereum's history is "The Merge," an upgrade transitioning the network from PoW to Proof of Stake (PoS). This shift markedly reduced the network's energy consumption and introduced a new layer of consensus, where validators play a pivotal role in securing and maintaining the network's integrity. In the PoS system, validators, chosen based on their ETH stake, contribute to the network's energy efficiency and security. The mechanism of selecting validators based on their stake not only conserves energy but also democratizes the network's consensus process.

2.1.3 Smart contracts

Smart contracts [6], as autonomous computer programs running on the blockchain, revolutionize the way contracts are negotiated by automating the agreement processes. These contracts are immutable and decentralized, which guarantees their security and trustworthiness. Once a smart contract is deployed on the blockchain, its code cannot be altered, making it ideal for transactions and contractual relationships that value transparency and impartiality.

The elimination of intermediaries by smart contracts reduces both the cost and duration of negotiations significantly. Furthermore, smart contracts can enforce contract conditions automatically, providing incentives for compliance and enabling the creation of decentralized governance systems. These systems facilitate decision-making in a transparent and democratic manner across the network.

Smart contracts self-execute, with the terms of agreements embedded within their code, and automatically enforce conditions when predetermined criteria are met. This characteristic, along with the decentralized execution on a secure network, underscores

their reliability. The advent of smart contracts on the Ethereum platform marked a pivotal development, extending blockchain applications beyond mere currency transactions.

These contracts, once compiled into EVM (Ethereum Virtual Machine) Bytecode, associate specific costs with each instruction, reflecting in fees for both deployment and execution proportional to their complexity.

The Ethereum ecosystem categorizes tokens based on their utility and exchangeability, serving various purposes from acting as currency and commodities to providing utility and security within the digital economy. Smart contracts facilitate these transactions and interactions, significantly extending the capabilities of traditional financial instruments and enabling a fully decentralized digital economy. The deployment of these contracts involves their compiled code running on the blockchain, offering the option to include the source code at the discretion of the developer.

Smart contracts are not signed but can be invoked either locally, simulating execution without incurring costs, or globally, executing on the blockchain with associated costs and immutable outcomes.

2.2 Application Domain

In the evolving landscape of blockchain technology, a myriad of application domains stands out, each harnessing the decentralized, secure, and transparent nature of blockchain to innovate and transform traditional systems:

- **Decentralized Finance (DeFi) [7]:** Blockchain facilitates a plethora of financial services including, but not limited to, peer-to-peer lending, automated trading, and asset management, revolutionizing traditional finance with transparency and efficiency.
- **Gaming and Virtual Assets:** Blockchain introduces a new paradigm in gaming, enabling true ownership of in-game assets, provably fair gaming mechanisms, and decentralized gaming economies, thereby enhancing player engagement and trust.
- **Digital Identity and Privacy:** Through blockchain, individuals gain unprecedented control over their digital identities, allowing for secure and selective sharing of personal data, thereby mitigating risks associated with data breaches and identity theft.
- **Decentralized Social Networks:** Offering an alternative to traditional social media platforms, blockchain-based social networks prioritize user privacy, resist censorship, and protect against data misuse, promoting freedom of expression and data sovereignty.

- **Transparent Supply Chains:** Blockchain technology ensures the authenticity of products by providing a transparent, unalterable ledger for tracking the provenance and lifecycle of goods from origin to consumer, fostering trust and sustainability in supply chains.
- **Secure Data Storage:** Leveraging decentralized networks for data storage ensures enhanced security, privacy, and data integrity, offering a resilient alternative to centralized cloud storage solutions.
- **Innovations in Healthcare:** Blockchain empowers the healthcare sector with secure patient data management, enhances interoperability among healthcare providers, and ensures the authenticity of pharmaceuticals, improving patient care and privacy.
- **Transparent Voting Mechanisms:** By facilitating secure, anonymous, and verifiable voting processes, blockchain technology promises to restore trust in electoral systems and democratize voting, making it accessible and transparent.
- **Intellectual Property and Creative Rights:** Blockchain serves as a robust platform for creators to register, manage, and monetize their intellectual property, ensuring creators' rights are protected and royalties are fairly distributed.
- **Decentralized Energy Grids:** Blockchain technology paves the way for peer-to-peer energy trading, allowing individuals to buy, sell, or exchange renewable energy directly, promoting sustainability and energy independence.

2.2.1 NFTs

Non-Fungible Tokens (NFTs) [8] represent a breakthrough in the digital representation of ownership and authenticity. Unlike cryptocurrencies and other fungible tokens where each token is identical and interchangeable, NFTs are unique digital assets that signify ownership of specific items or content, such as digital art, collectibles, and even real estate in virtual worlds. The uniqueness and non-interchangeability of NFTs make them ideal for certifying the originality and ownership of digital and physical assets in a transparent and immutable manner on the blockchain.

ERC-721, one of the first standards for representing non-fungible digital assets on the Ethereum blockchain, lays the foundation for the creation and trading of NFTs. It defines a set of rules that a token must follow to be considered an ERC-721 token, ensuring interoperability across the ecosystem. Each ERC-721 token is distinct, with metadata that can link to external information, thereby proving the token's uniqueness and ownership. This standard allows developers to create sophisticated digital collectibles and virtual assets, opening up vast opportunities for artists, gamers, and content creators to monetize their creations directly and securely.

2.2.2 Tokens

Tokens [9] in the blockchain context represent digital assets or units of value that are created and managed on a blockchain platform. These tokens can serve various functions, from representing a stake in a decentralized application (dApp), acting as a medium of exchange within an ecosystem, to symbolizing ownership of real-world or digital goods. The flexibility and programmability of tokens enable the creation of complex economic systems and incentive structures within blockchain projects, facilitating everything from digital currencies and utility tokens to asset-backed tokens and more.

ERC-20 is a widely adopted standard for fungible tokens on the Ethereum blockchain. It defines a common list of rules and functions that an Ethereum token contract must implement, ensuring compatibility with the broader Ethereum ecosystem, including exchanges, wallets, and other smart contracts. Fungibility implies that each token is identical to another in value and function, making ERC-20 tokens ideal for creating cryptocurrencies, voting tokens, staking tokens, and more. The ERC-20 standard has played a pivotal role in the widespread adoption of Ethereum as a platform for developing and launching a diverse range of tokens and decentralized applications.

2.2.3 Stakers

Staking smart contracts [10] are specialized types of smart contracts in the blockchain ecosystem designed to facilitate staking mechanisms. Staking involves locking up a certain amount of cryptocurrency within a smart contract to support the operations of a blockchain network, such as transaction validation or network security in Proof of Stake (PoS) systems. In return for their contribution, stakeholders are typically rewarded with additional tokens, proportional to the amount and duration of their stake. These contracts automate the staking process, rewards distribution, and other related mechanisms, ensuring transparency, security, and efficiency in the staking operations.

ERC-900 is a proposed standard interface for staking contracts on the Ethereum blockchain, aimed at providing a consistent API for staking tokens. This standard allows for the creation, management, and rewards distribution of staked tokens, offering a flexible framework that supports a wide range of staking models.

2.2.4 Liquidity Pools

Liquidity pools [11] are foundational components of decentralized finance (DeFi) ecosystems on blockchain platforms, facilitating automated and permissionless trading of digital assets. These pools are essentially smart contracts that hold reserves of two or more tokens, creating a market for these tokens that allows users to trade directly with the pool rather than with individual counterparts. The prices of the tokens in a liquidity

pool are determined by a constant mathematical formula (e.g., the constant product formula in the case of Uniswap), ensuring that the pool remains balanced and that the token price adjusts according to supply and demand dynamics.

Participants, known as liquidity providers, contribute an equal value of two tokens to the pool and in return receive liquidity tokens, which represent their share of the total liquidity pool. These liquidity tokens can be redeemed for the underlying assets at any time. In addition to facilitating trading, liquidity pools offer incentives for liquidity providers by distributing a portion of the trading fees to them based on their share of the pool. This mechanism incentivizes the provision of liquidity, ensuring the availability of assets for trading and contributing to the overall efficiency and liquidity of the DeFi market.

3 Presentation of the context

3.1 Aim of the DApp

3.1.1 Gameplay

The objective of the DApp is to address a gap left by traditional Web 2.0 games, which is the inability to transfer or sell assets earned within the game to the outside world. AncientSociety introduces a video game that not only retains high playability but also enables players to acquire assets they can freely trade or sell on external marketplaces to anyone looking to join the game with already farmed assets. Furthermore, the player has personal ownership of the asset in their own wallet, making it an irremovable resource without their explicit consent. This feature significantly enhances the game's trustworthiness in the eyes of potential players.

3.1.2 Play-to-earn

Owners have several avenues for revenue generation, with the primary methods being the production of tokens via their NFTs, which can be exchanged in specialized liquidity pools for MATIC, and by upgrading their buildings (NFTs) to sell them on the marketplace at determined prices, thereby earning cryptocurrencies such as ETH or MATIC upon sale.

The economic model of the shop is anchored in the principles of the free market. However, given that Opensea—a renowned NFT marketplace—is the primary platform for selling NFTs, the DApp imposes an additional commission on purchases made through this marketplace. This extra charge is allocated towards enhancing the liquidity pool and acquiring other tokens (ANCIEN), thereby reducing the token's supply and augmenting its value.

Naturally, the value and rarity of a building directly influence its potential selling price, allowing owners to command higher prices for more prestigious and scarce buildings.

3.2 NFTs

Buildings are one of the NFTs within the video game and serve as the primary means for playing and leveling up your city. As NFTs, these buildings are implemented using ERC-721 contracts, adhering to the standards proposed by EIP [12]. The NFTs vary in type and rarity, with each type enabling players to generate the resources necessary for leveling up, enhancing resource production, and more. Among these, we have the Townhall, Lumberjack, and Stonemine.

Certain NFTs unlock various activities within the city, through which players can discover items, recipes, and tools essential for further city development and for

acquiring increasingly rare resources. Examples include the Fisherman's Hut, Miner, and Farmer.

The final NFT category is Land, the rarest of all game NFTs, as it does not produce resources or unlock any activities. Instead, it allows for the staking of resource buildings, such as the Lumberjack and Stonemine, thereby boosting their productivity. In return, the landowner can impose a tax on players who choose to relocate their buildings to these lands.

3.2.1 Metadata

All buildings come equipped with metadata to be displayed on marketplaces such as Opensea, in addition to providing information about the building, including its rarity, unique traits, level, and other details. The method employed for the generation and storage of these metadata involves a script on a dedicated server that dynamically generates building information, such as its level, the amount of resources being produced at the moment, and other dynamic information. This type of metadata is utilized by all buildings.

3.3 Stakers

AncientSociety requires assurance that the player intends to retain their buildings and must continue to possess them for as long as they wish to use them in the game. To achieve this, the NFTs must be staked in specific contracts, which are inspired by the standard proposed by ERC-900. The purpose of these contracts is to "lock" the NFTs, preventing the player from sending, exchanging, or selling the buildings while they intend to use them in the game. When the player decides to trade or sell them, they must unstake the buildings from the staking contracts, after which the buildings will no longer be present in the game, cease to produce resources, and no longer unlock the related activity in the game.

During the staking phase, the player temporarily transfers ownership of the NFT to an external contract, which cannot transfer the NFT to anyone other than the actual owner. Similarly, lands also require a staking contract to allow the owner to use them within the game. Moreover, we must prevent the player from unstaking the land while there are active contracts, i.e., with other buildings staked on it.

To facilitate this, AncientSociety utilizes "land lease contracts," which are essentially signatures made by the land owner, specifying the start and end of the "lease contract" for that land. Upon expiration, all buildings on that land will be automatically removed. The lease contract can only be terminated by the land owner, provided there are no buildings on the land.

3.4 Resources

In the video game, several resources are crucial for city development, including ANCIEN, wood, and stone. Among these, ANCIEN is particularly intriguing for players as it not only serves as an in-game resource but can also be converted into an actual token, implemented following the ERC-20 standard. This conversion process is facilitated through the signature of a wallet known as the "signer," generating a "voucher" that the player can redeem on the ANCIEN contract itself.

This voucher specifies various details such as the receiver of the tokens and the amount. To secure these details and provide additional information, the ANCIEN contract also incorporates EIP-712 [13]. This allows for hashing and signing the voucher to ensure its authenticity and security. After being used, the voucher is stored on a parallel contract that acts as storage for already used vouchers.

3.5 Marketplaces

The game features various marketplaces for minting a variety of NFTs, items, recipes, and tools crucial for gameplay, as well as for the sale of NFTs by players themselves.

3.5.1 Mint

There are two distinct marketplaces for minting. The first is used to directly mint individual NFTs through two separate mechanisms. The first method allows NFTs to be received by players on a whitelist, enabling them to obtain buildings for free. This whitelist is implemented using a Merkle Tree [14], which generates a data structure that is efficient in terms of space and time for verifying the inclusion of a specific wallet in the whitelist. The second method is a traditional mint, allowing players to purchase a specific building by paying with the network's native cryptocurrency (MATIC), offering the option to select the quantity of NFTs to purchase in a single transaction.

The second marketplace is more complex, enabling the purchase of not only individual buildings but also bundles of different buildings, along with off-chain elements like items and tools. This marketplace supports the use of various currencies, including the chain's cryptocurrency (MATIC) and other tokens or crypto wrappers from different CHAINS (such as WETH), maintaining real-time pricing.

3.5.2 Oracle

The real-time acquisition of accurate market values for various cryptocurrencies on a specific blockchain is facilitated through the deployment of Chainlink oracles [15]. These oracles play a critical role in bridging the gap between the decentralized digital environment and real-world data by fetching the live market prices of cryptocurrencies.

This feature is crucial for ensuring the game's economic mechanisms reflect current market conditions, thereby enhancing the realism and fairness of in-game transactions.

Moreover, the dynamic aspects of the game, such as the minting of buildings and the distribution of in-game items, are regulated by an off-chain event handler. This component is designed to monitor the marketplace contract for specific triggers continuously. Upon detecting an event, it verifies the details of the NFTs to be minted or items to be released and executes these operations based on the current state of the game and the blockchain. This sophisticated interaction between on-chain and off-chain elements underscores the game's innovative approach to integrating real-time data for a seamless and immersive gaming experience.

3.5.3 Opensea

Regarding the sale of NFTs, a specific marketplace has not been developed; instead, the use of Opensea [16], a well-known marketplace for NFT sales, is encouraged. Through Opensea, Ancient Society earns a commission used to increase the liquidity pool by purchasing and burning ANCIEN tokens, thereby reducing their circulation and increasing their value.

3.6 Uniswap

We have utilized Uniswap to establish a liquidity pool [17], facilitating the trading of ANCIEN tokens with MATIC by the players. This liquidity pool is powered by the contributions of the players themselves. To discourage players from draining the liquidity pool, various game mechanisms have been implemented that incentivize players to deposit both ANCIEN and MATIC, thereby potentially increasing the value of the ANCIEN token.

One such mechanism allows players to add a chosen amount of ANCIEN/MATIC to the pool. Depending on the volume of tokens contributed, players are granted access to various bonuses; the larger the contribution, the higher the quality of the bonus, which can include receiving NFTs for use within the game or for sale on the marketplace.

After depositing tokens into the liquidity pool, the respective address is monitored for a period of 72 hours by an off-chain event handler. This ensures that the player does not withdraw liquidity within this timeframe. Upon completion of this period, players are awarded a corresponding bonus.

Functioning of a Liquidity Pool:

Liquidity pools [18], such as the one implemented on Uniswap, operate on the principle of automated market makers (AMMs), enabling trading between crypto assets in a decentralized and non-custodial manner. Participants, or liquidity providers, pool two assets at a predetermined ratio to create a market. Prices are determined

algorithmically based on the relative supply of the two assets in the pool, ensuring liquidity and enabling immediate trades without the need for a traditional buyer-seller matching system. This innovative approach to trading and liquidity provision on blockchain platforms like Uniswap exemplifies the synergy between technological advancements and strategic game design, offering players not just a gaming experience but an opportunity to engage with real-world economic systems.

4 Software Architecture

4.1 Front-end

To develop a dynamic front-end, **React** [19] was employed. For the Game component, the front-end's initial steps involve connecting to either **Metamask** [20] or **Coinbase Wallet** [21], bypassing the need for traditional login procedures with email and password. Instead, users are automatically logged in using the wallet with which they sign a welcome message. This message is then used by the back-end to identify the address responsible for the signature, as the signed message and the message to be signed together invariably lead back to the signer's public address. Following this, the front-end receives a **JWT (JSON Web Token)** [22] from the back-end, which is thereafter utilized to authenticate various API calls.

Regarding marketplace login, the front-end simply needs to connect to the Metamask wallet, select the appropriate chain for the transaction, and then invoke the marketplace contract by calling the desired function.

Furthermore, during gameplay, the front-end's role extends beyond merely displaying information returned from the back-end; it also involves invoking functions of various smart contracts, such as those for staking or ERC-20 for token withdrawal. To accomplish this, the front-end once again relies on necessary information returned from the back-end upon calling specific APIs.

4.2 Back-end

For the back-end, **Node.js** [23] was utilized. The back-end's key responsibility includes verifying the authenticity of the wallet address during the login process. For new users not previously registered, it creates a corresponding record in the **database (DB)**. Otherwise, it generates a **JWT (JSON Web Token)** containing the address used for login, which authorizes the user until the JWT expires.

During gameplay, the back-end is tasked with checking which **NFTs** the user owns, identifying which are staked, the volume of unclaimed resources produced, and the quantity and types of resources, items, tools, and recipes in the inventory. It also checks for any offers on the game's internal marketplace, dedicated solely to off-chain elements like items, tools, or recipes, and verifies the ownership of lands for the user. Moreover, it manages all in-game activities that do not involve specific interactions with the blockchain, which will not be detailed further due to their complexity and being outside the scope of this report and course.

In the NFT minting phase, the back-end verifies whether the user is on the whitelist and, if so, generates and returns the necessary **Merkle proof** to confirm the presence of the public address on the whitelist.

Lastly, for the main marketplace, it provides the front-end with all currently

available offers, the supported blockchains, the usable currencies along with their respective contracts and amounts.

4.3 Database

For the database, a **relational database** was employed, specifically **MySQL** [24]. The database is crucial for maintaining all information related to the video game, such as **items**, **tools**, and **recipes**, as well as activities users can engage in through the **buildings** they have staked. It also holds data pertinent to the blockchain, including which buildings are currently in the wallet, which are staked, the amount of **ANCIEN** on the blockchain, whether ANCIEN has been deposited into the **liquidity pool**, and all information related to the marketplace such as available **currencies**, the **contract address** of oracles, and which **NFTs** are being distributed along with their corresponding offers.

4.4 Smart contracts

The diverse smart contracts developed for the application were written in **Solidity** [25]. Multiple smart contracts have been deployed, including an **ERC721** for each building and land with their respective staking contracts, an **ERC20** for ANCIEN tokens, and a contract for the marketplace. The following sections will highlight the key aspects of all contracts.

Buildings and Lands

These contracts inherit from the **ERC721Enumerable** contract, a variant of the **OpenZeppelin's EIP-721** standard, enriching the classic ERC721 functionalities. Buildings incorporate two minting functions; the first adheres to the EIP specification, being a payable function that requires the address to which NFTs should be issued and the quantity the user wishes to purchase. The second minting function allows users on a whitelist to receive free NFTs, utilizing a **MerkleProof** for this purpose. Additionally, an airdrop function is utilized by an off-chain event handler for the acquisition of a bundle on the marketplace, evaluating which buildings are in the bundle and airdropping them accordingly. Lands, however, lack mint functions as they can only be purchased through the marketplace.

Staking Contracts

Inspired by **EIP-900**, staking contracts implement stake and unstake functions, generate *Staked* and *Unstaked* events, and provide information on the total number of buildings staked and which buildings are staked for a specific address if any. For lands, the staking contract differs slightly; in addition to stake and unstake functions, the landowner

can create a contract specifying several details essential for informing a user wishing to place their city on a land. These details are stored in a *ContractLand struct* with properties such as *idLand, owner, expireTime, fee*, and *active*. Creation or deletion of a contract is contingent upon back-end approval and certain conditions, like the land being staked and no active contracts existing.

ANCIEN

The game's token, ANCIEN, implements an **ERC-20** standard. Minting of this token requires back-end approval, which verifies the amount generated by the user's townhall or purchased in the game's internal marketplace, adjusts the minting amount, and, similar to the land staking contract, generates a voucher containing details like the ANCIEN amount to be minted and the recipient address.

VoucherStorage

This contract was developed solely to maintain all vouchers generated for both ANCIEN and land contracts, ensuring the integrity and traceability of transactions and contractual agreements within the game environment.

4.5 Event handlers

Event handlers constitute a crucial component of the application while also representing a potential bottleneck. Their roles are multifaceted:

Transfer:

When a user transfers a building or land from one wallet to another, whether through a sale on Opensea, a simple transfer, or immediately after minting, an event handler is tasked with capturing the corresponding *Transfer* event and updating the NFT's owner in the database.

Stake/Unstake:

For user actions involving staking or unstaking a building or land, an event handler updates this information in the database. This update allows the asset to appear in the user's city, begins resource production or unlocks related activities by updating the database.

Land Contract Creation/Deletion:

Creating or deleting a contract for a land requires database updates to enable new contract creation or prevent it while allowing other users to place their cities on the land to boost production.

ANCIEN Mint/Burn:

If a user wishes to create a voucher for minting ANCIEN, the voucher is stored in the database until used. Upon usage, it is recorded in the user's history, enabling the generation of new vouchers for future minting. Conversely, if a user decides to use ANCIEN within the game, they can burn the ANCIEN, and the event handler captures the *Burn* event, increasing the ANCIEN storage in the game.

Liquidity Checks:

To encourage players to increase the liquidity pool, a mechanism allows them to receive blessings, granting various in-game bonuses based on the MATIC/ANCIEN amount added to the pool. The event handler captures liquidity increases and lists users for blessing receipt. It also monitors for decreases in pool position by withdrawing MATIC/ANCIEN, in which case it captures the *DecreaseLiquidity* event and revokes the player's eligibility for blessings.

Marketplace:

The final event handler monitors two different marketplace contracts, one deployed on the Ethereum network and the other on the Polygon network, facilitating multi-chain payments. This handler captures *Purchase* events, verifies user payments, checks if the amount matches the selected offer, and, if affirmative, airdrops the NFTs and adds purchased items, tools, and recipes to the database.

4.6 UML diagrams

4.6.1 Component diagram

This section will show the component diagram for AncientSociety.

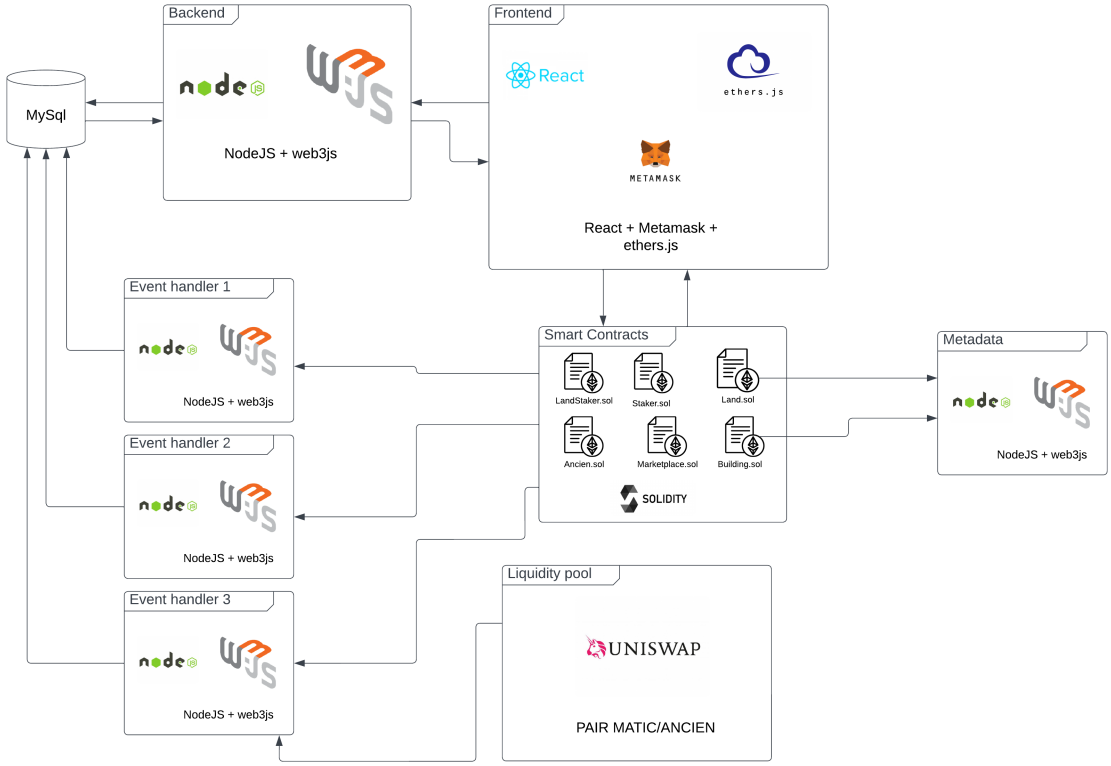


Figure 1: Component diagram

4.6.2 Deployment diagram

This section will show the deployment diagram for all AncientSociety components.

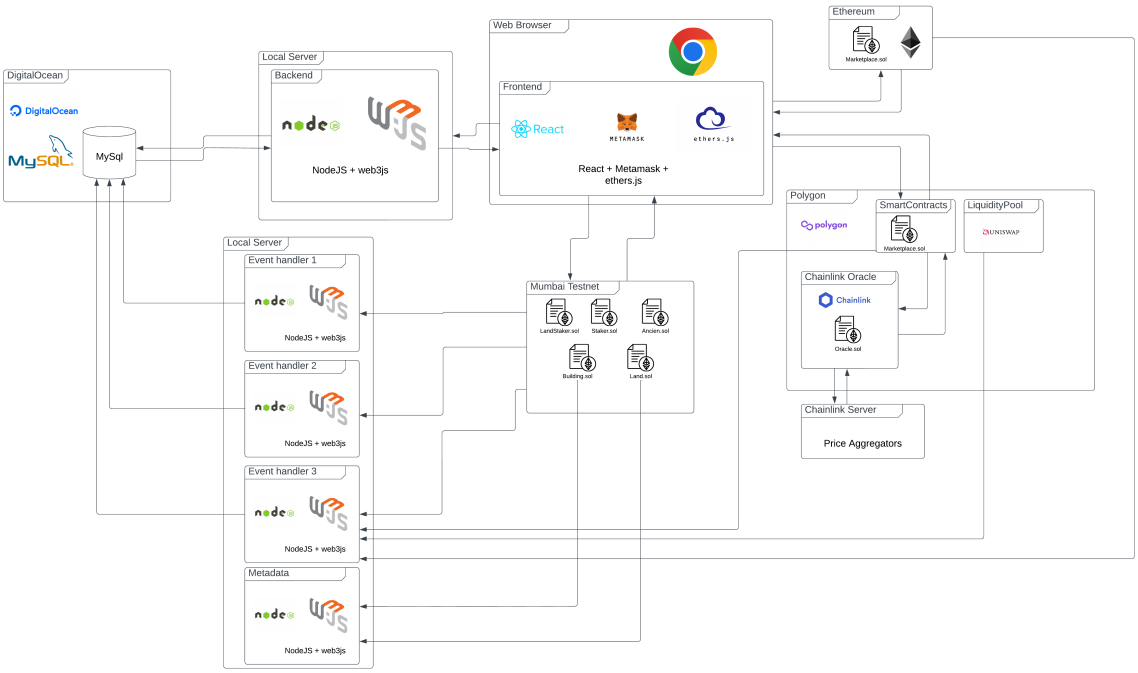


Figure 2: Deployment diagram

4.6.3 Class diagrams

This section will show class diagrams for certain AncientSociety smart contracts.

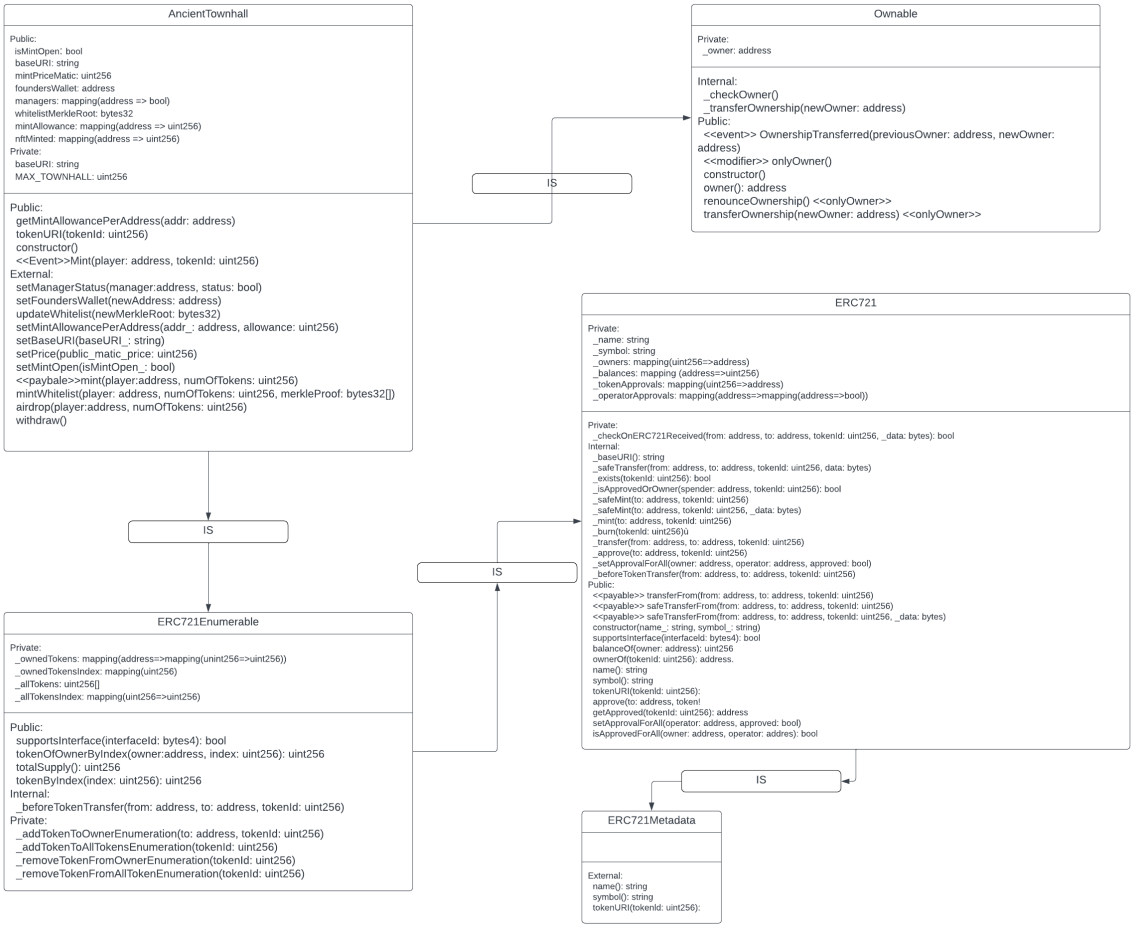


Figure 3: Building contract

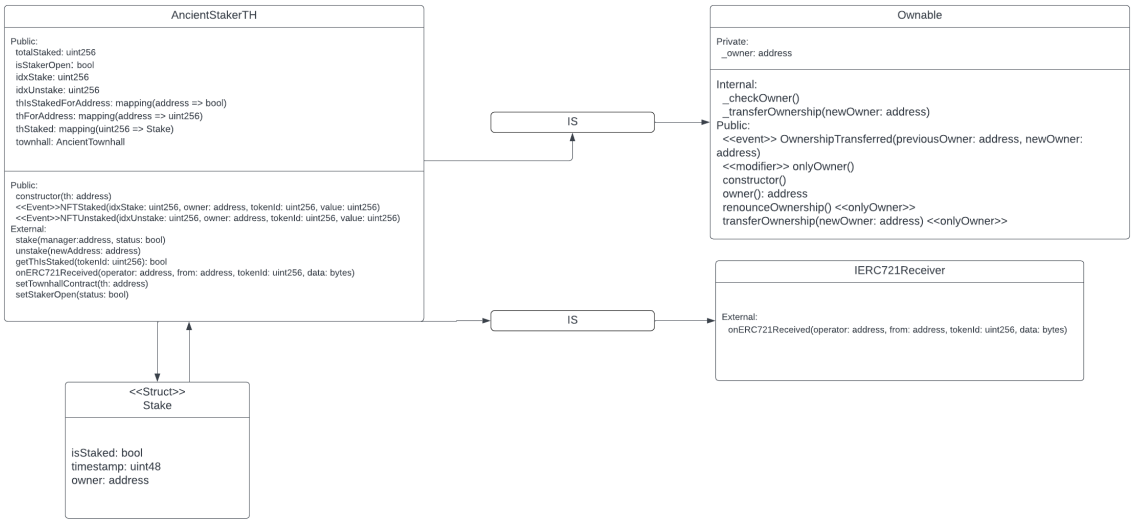


Figure 4: Staking contract

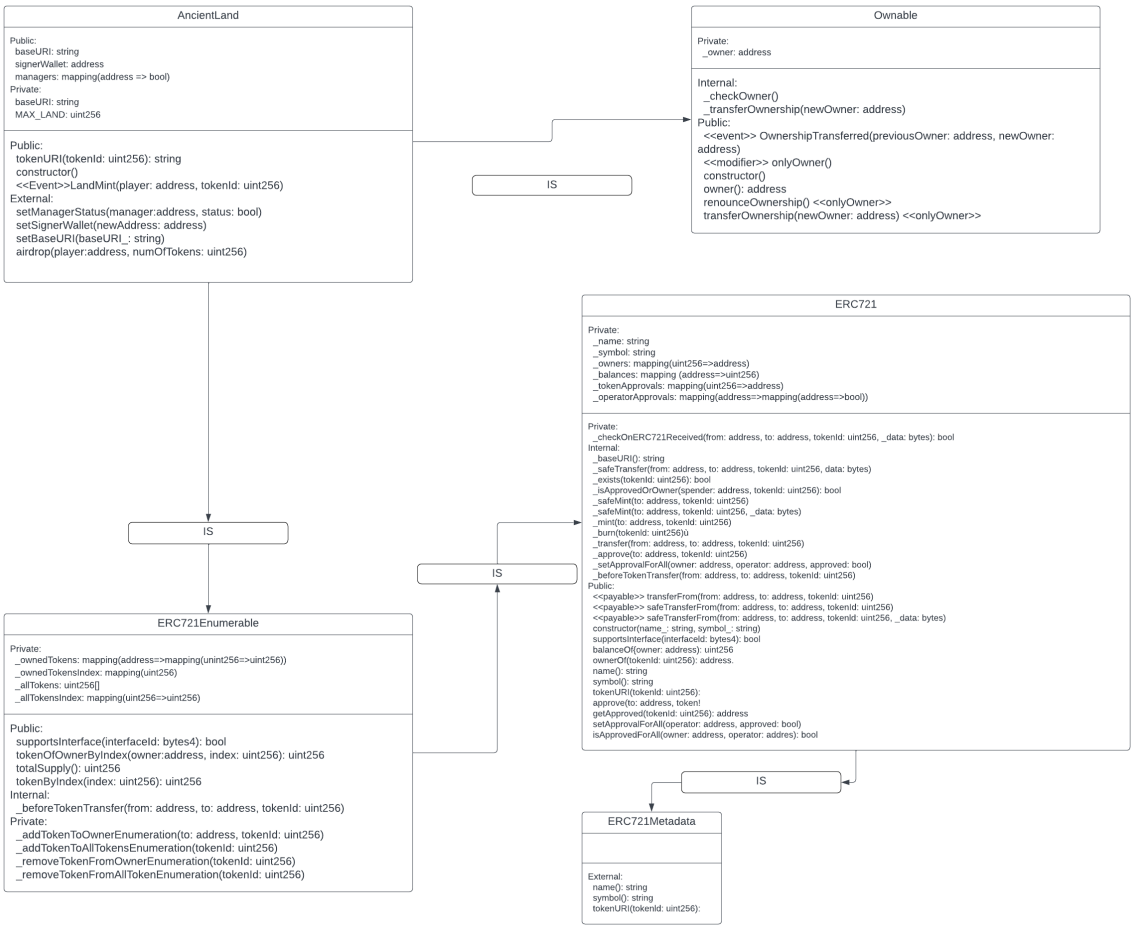


Figure 5: Land contract

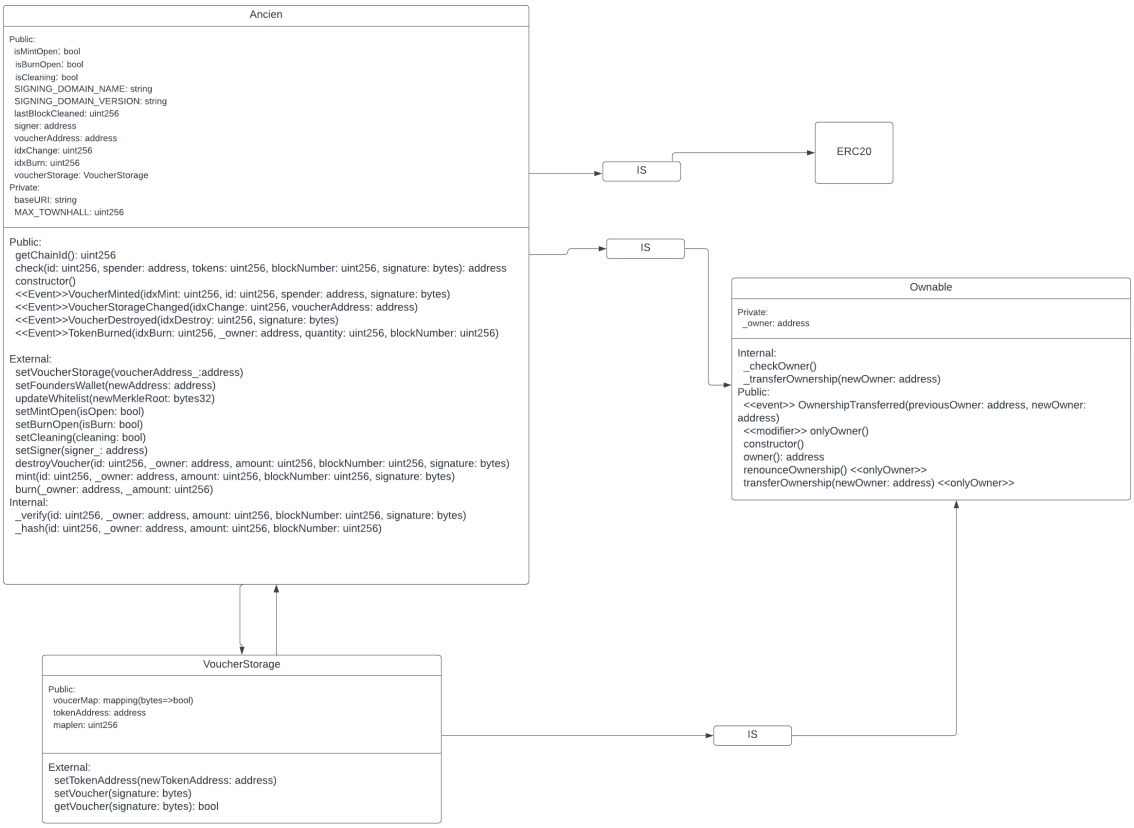


Figure 6: Ancien contract

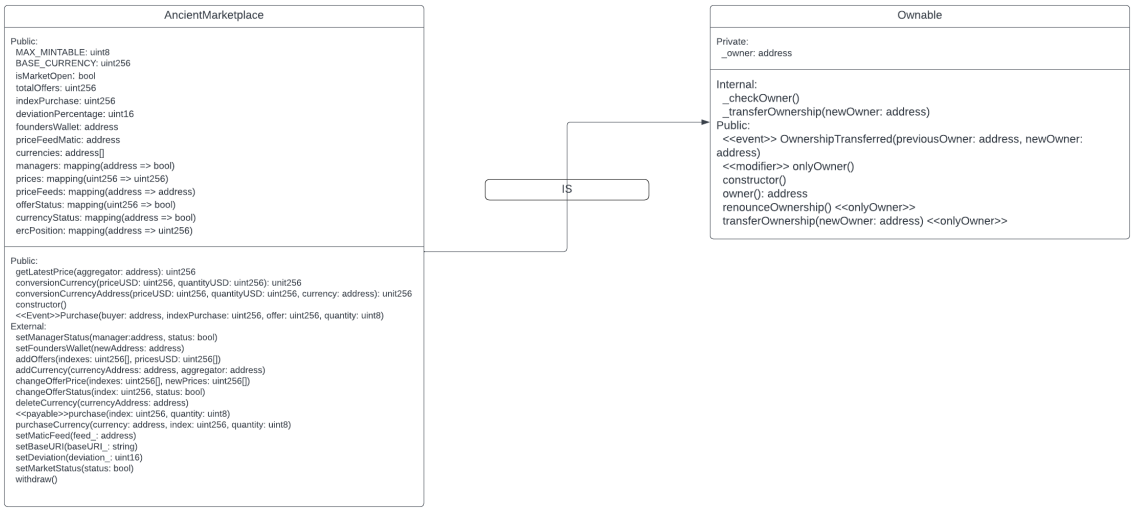


Figure 7: Marketplace contract

4.6.4 Sequence diagrams

This section will show sequence diagrams for certain AncientSociety functionalities.

The picture 7 shows the process of Login into the application. The picture 8 shows the process to Stake or Unstake a building into the staking contract.

The picture 9 shows the process of creating a new land lease contract. This contract is needed by the owner to generate the tickets and invite other user to put their cities onto his land.

The picture 10 shows the process of deleting a land lease contract. The picture 11 shows the process to generate a voucher used to mint new tokens. The picture 12 shows the process to buy on the marketplace.

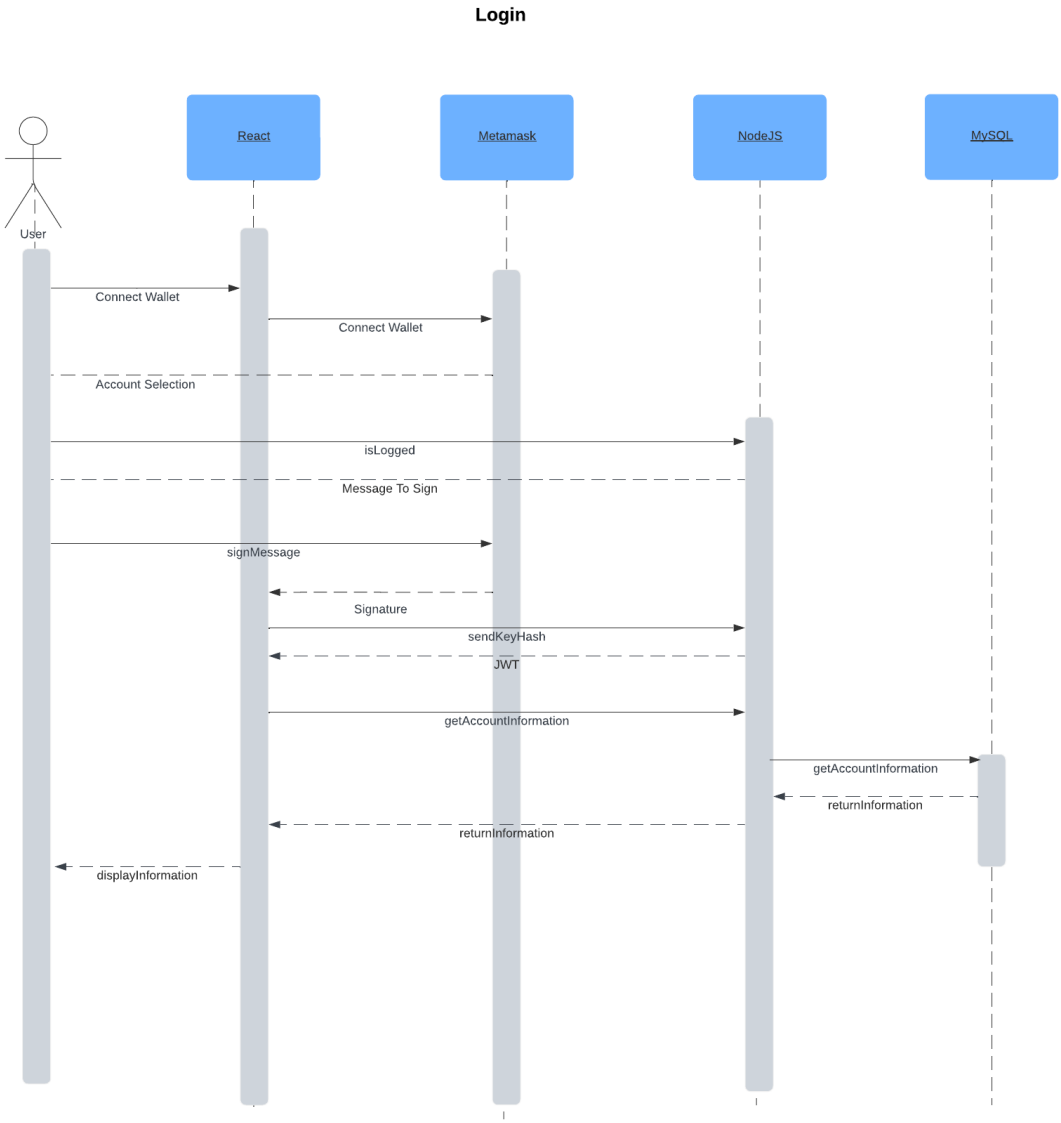


Figure 8: Login process

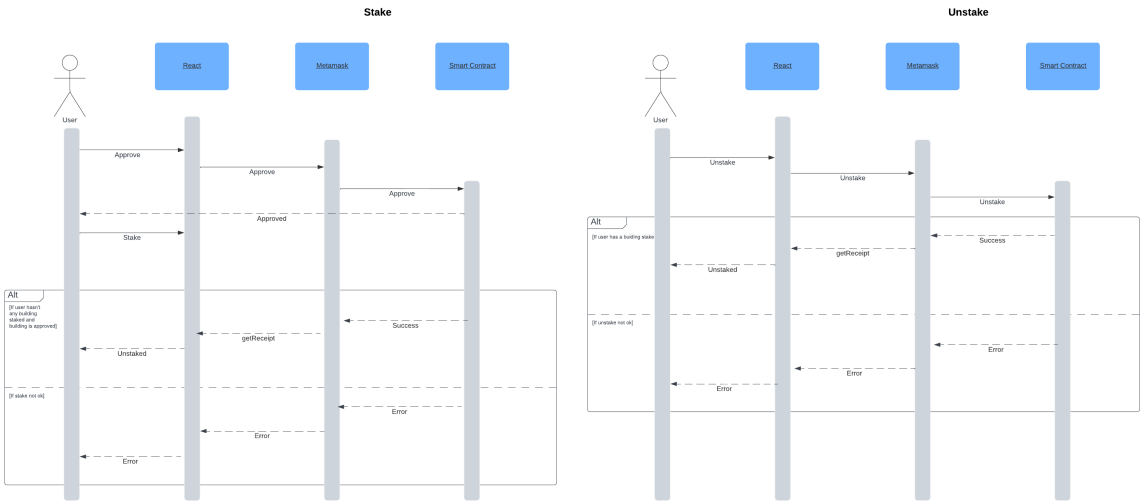


Figure 9: Processes to stake/unstake a building

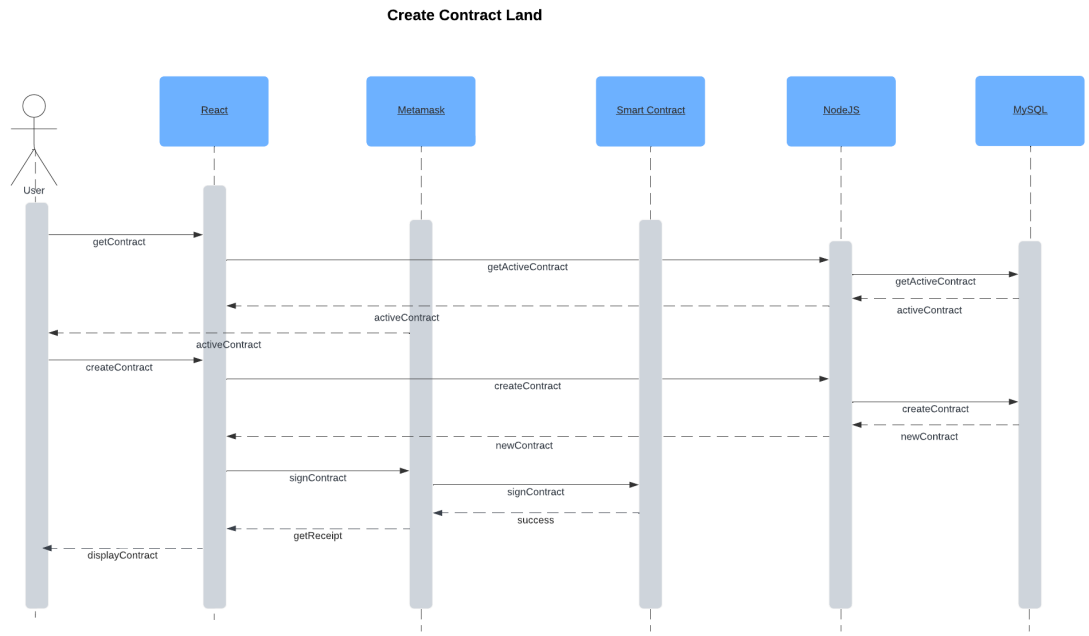


Figure 10: Process to create a land lease contract

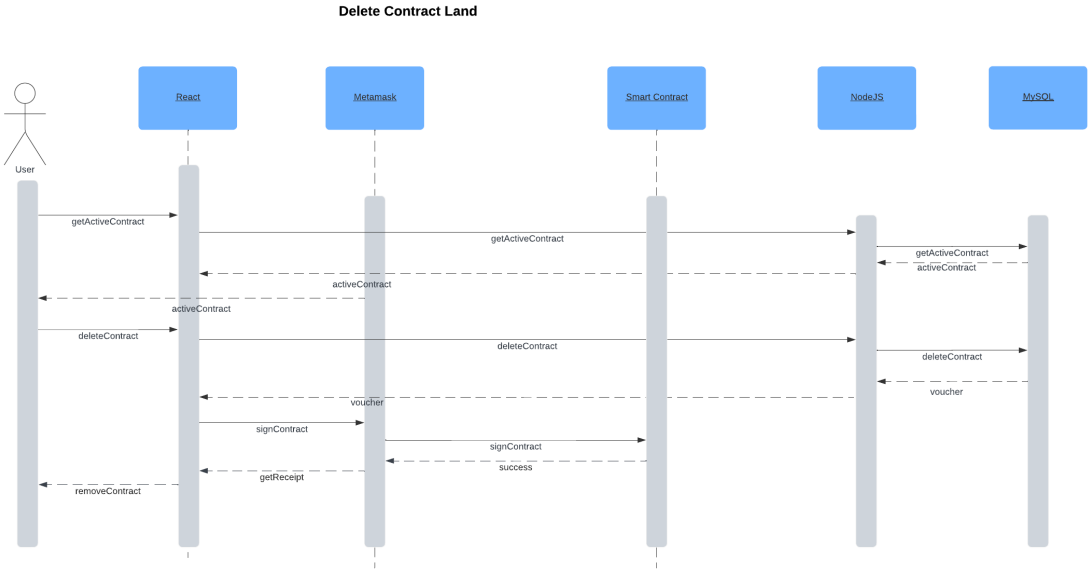


Figure 11: Process to delete a land lease contract

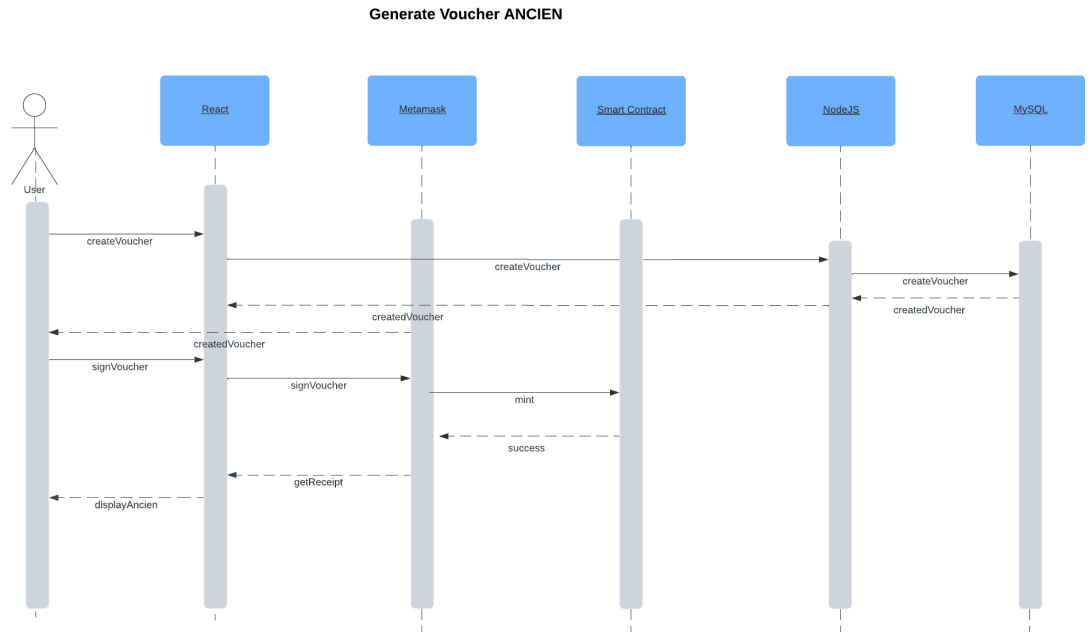


Figure 12: Process to create an Ancien voucher and mint the tokens

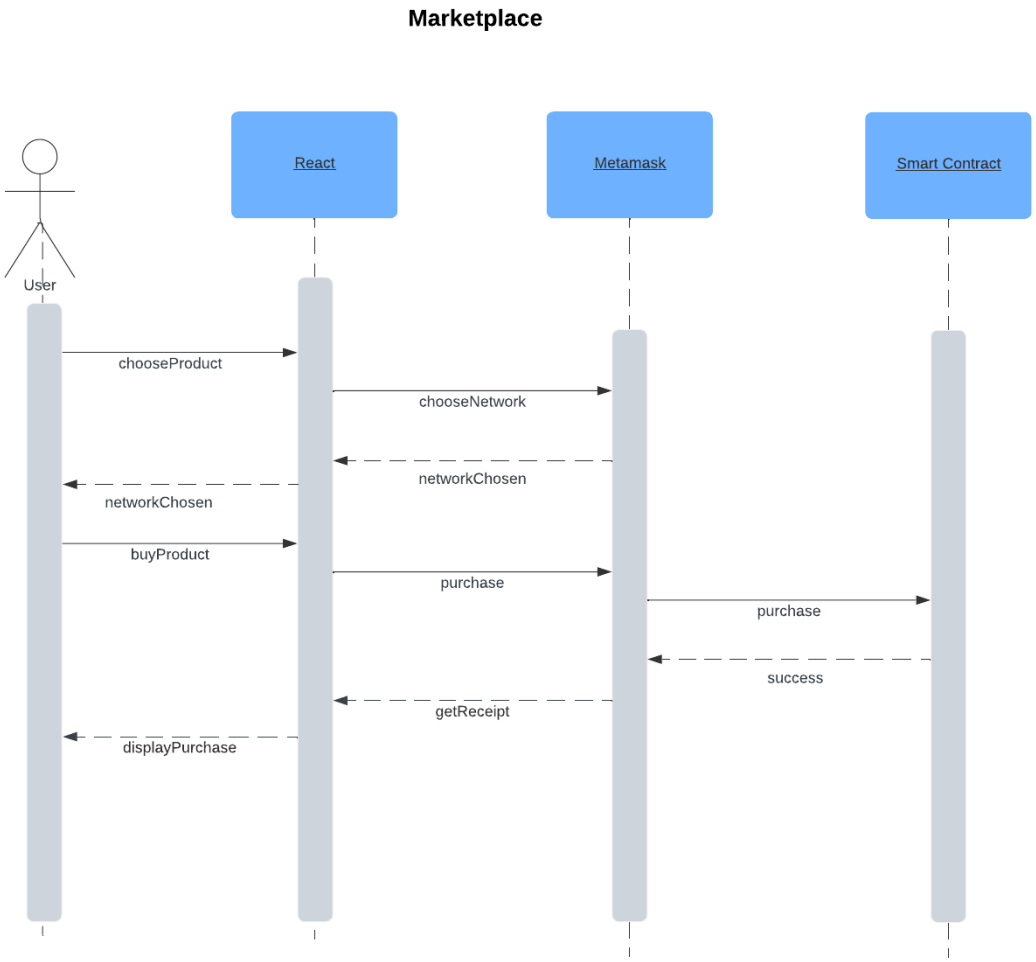


Figure 13: Process to buy on the marketplace

5 Implementation

5.1 Tools used

The following tools have been used to implement my DApp:

- **React:** a JavaScript framework used to create user interfaces, interact with metamask and communicate with the backend.
- **NodeJS:** an open-source, cross-platform, event-driven runtime system for executing JavaScript code, allowing the use of JavaScript for backend development and server-related coding. Node.js has been employed not only to construct the backend of the game but also to build all event handlers and the server to generate metadata for NFT contracts.
- **Web3.js [26]:** a JavaScript library used to connect to blockchain nodes and interact with the chain itself. This library has been leveraged by all backend servers, including event handlers, to listen for possible Events triggered by smart contracts or by the metadata server to fetch on-chain information, such as the number of buildings minted or whether a building is currently staked. For event handlers, it was necessary to establish connections based on WebSocket (WSS) nodes, as bidirectional communication was required to be triggered by on-blockchain events. Otherwise, connections based on JSON-RPC are preferable.
- **Ethers.js [27]:** another JavaScript library for connecting to blockchain nodes and interacting with the chain. Ethers.js has been primarily used in the front end with React, where its purpose is to connect the user to Metamask and enable interaction with various smart contracts.
- **Metamask:** a browser extension and cryptocurrency wallet; it serves as a bridge between the user's browser and the blockchain.
- **MySQL:** a relational database used to store information about all players, including the buildings they own, their arrangement in the game, their available resources, etc.
- **Solidity:** a high-level, object-oriented programming language for developing smart contracts on various blockchains, primarily EVM-based blockchains.
- **Truffle [28]:** a development environment, testing framework, and asset pipeline for Ethereum, designed to facilitate the work of Ethereum developers. It has been used to deploy smart contracts on several blockchains.
- **Chainlink oracles:** have been utilized to convert marketplace product prices from USD to various cryptocurrencies and tokens.

5.2 GUI

This section will show the user interface for some of the AncientSociety game components and functionalities.

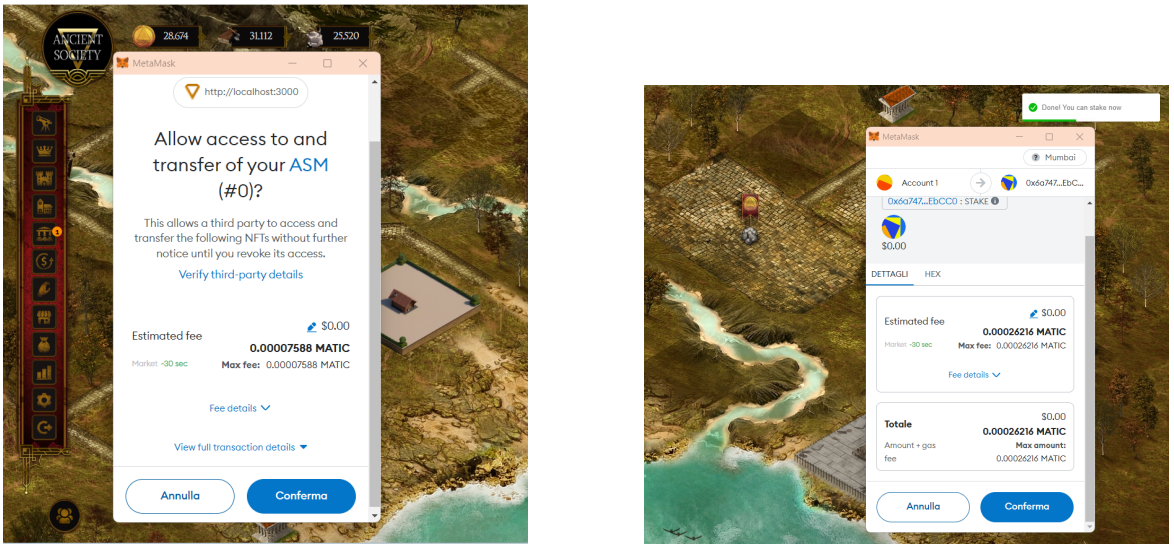
5.2.1 Login

Figure 14: Login Page

5.2.2 Game

Figure 15: Map of the game

Figure 16: List of all buildings available



(a) Approve

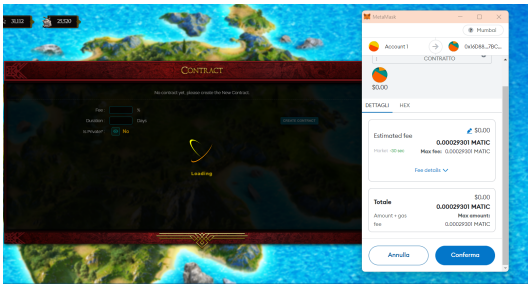
(b) Stake a Building

Figure 17: Process to stake a building. First of all the user must approve the staking contract to Transfer the NFT from the owner to the staking contract, secondly the user call the stake function to Transfer the NFT ownership.

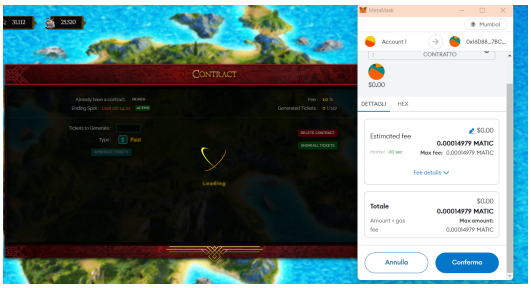
5.2.3 Lands

Figure 18: All the lands owned

Figure 19: Map of the land



(a) Create a contract



(b) Delete a contract

Figure 20: Process to create or delete a contract from a land.

(a) Create tickets and choose their price

(b) Ticket marketplace

Figure 21: Process for generating and selling tickets to users wishing to relocate their buildings onto lands to enhance their production.

5.2.4 Ancien

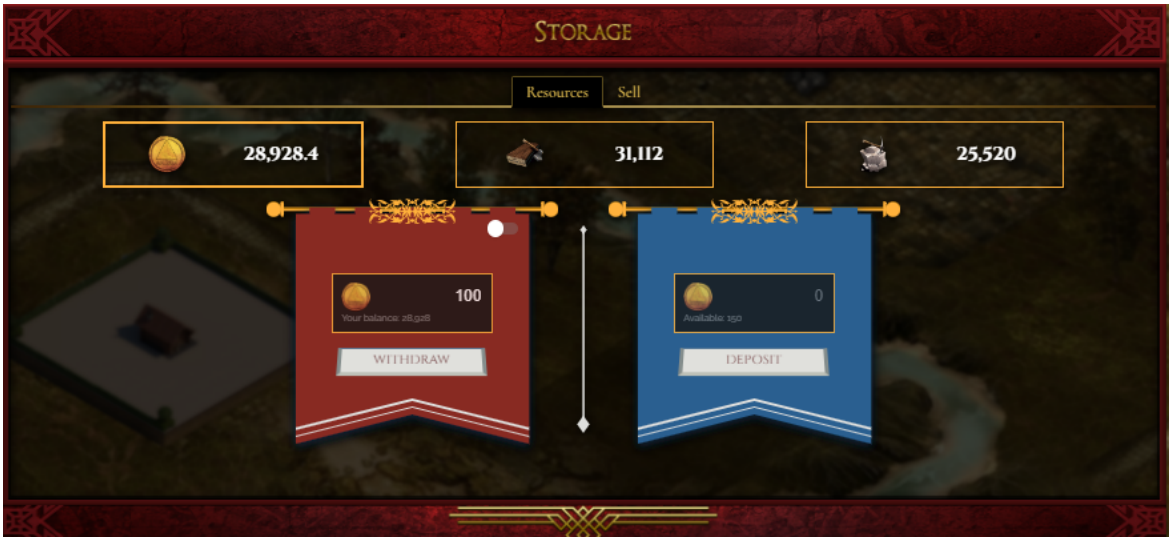


Figure 22: All the resources owned

Figure 23: Minting new tokens

6 Known issues and limitations

Ancient Society faces several limitations, some of which are briefly explained below.

Centralization

A significant level of centralization remains within the video game to ensure a comfortable user experience (UX). This centralization is manifested through the use of a Node.js back-end and a database (DB). These elements are deemed necessary due to the prohibitively high costs that would be incurred if all information currently stored in the DB were to be saved on the blockchain. Furthermore, removing the back-end would require users to execute an excessive number of transactions in each gaming session. This could lead to substantial costs, even on blockchains with low transaction fees like Polygon, and result in a poor user experience due to the transaction confirmation delay on the chain, albeit just a few seconds on platforms like Polygon.

Oracle

The downtime of an oracle poses a significant challenge in the marketplace since it prevents purchases due to the inability to convert the USD price in the DB into various cryptocurrencies. The price in USD is used to provide a stable value, unlike the volatile nature of cryptocurrencies. However, when an oracle fails, converting this value into the different currencies offered by the marketplace becomes impossible.

Event Handler

Throughout Ancient Society's history, the biggest challenge has been dealing with events not captured by the event handler, leading to missed updates in the DB and subsequent issues. Additionally, if a new user makes a purchase on the marketplace, they may not receive the purchased NFTs and in-game items. Various solutions have been explored, such as a recovery function that regularly checks whether all user information in the DB is correct. However, this approach can become a bottleneck with many users and contracts to verify.

7 Conclusions

In this report, we have introduced **Ancient Society**, a browser-based video game that aims to integrate Web 2.0 technology with blockchain technology. The primary objective of the video game is to instill greater trust among users by allowing them to own assets that are not controlled by any central authority. This empowerment also includes the ability to generate tokens that can be exchanged for cryptocurrencies, as well as to grow their NFTs and resell them on secondary marketplaces.

Furthermore, the game enables players to create a liquidity pool, affording them the freedom to determine the value of ANCIEN tokens themselves. It attempts to regulate token inflation through various mechanisms implemented within the game, thereby contributing to a dynamic and player-driven economy.

References

[1] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Bitcoin.org, 2008.

[2] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 4(3):382–401, 1982.

[3] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017.

[4] Du Mingxiao, Xiaofeng Ma, Zhe Zhang, Xiangwei Wang, and Qijun Chen. A review on consensus algorithm of blockchain. *IEEE SMC2017*, pages 2567–2572, 2017.

[5] Vitalik Buterin. Ethereum: A next-generation smart contract and decentralized application platform. *Ethereum White Paper*, 2014.

[6] Andreas M. Antonopoulos and Gavin Wood. Mastering ethereum: Building smart contracts and dapps. 2018.

[7] Fabian Schär. Decentralized finance: On blockchain- and smart contract-based financial markets, 2021.

[8] William Entriken, Dieter Shirley, Jacob Evans, and Nastassia Sachs. Erc-721 non-fungible token standard. *Ethereum Improvement Proposals*, (721), 2018.

[9] Fabian Vogelsteller and Vitalik Buterin. Erc-20 token standard. *Ethereum Improvement Proposals*, (20), 2015.

[10] Ethereum Foundation. Ethereum 2.0 staking mechanism. `https://ethereum.org/en/eth2/staking/`, 2020.

[11] Uniswap. Introduction to liquidity pools. `https://docs.uniswap.org/protocol/V2/concepts/core-concepts/pools`, 2021.

[12] Ethereum Community. Ethereum improvement proposals. `https://eips.ethereum.org/`, Accessed: 2024.

[13] Christian Reitwiessner et al. Eip-712: Typed data signing. `https://eips.ethereum.org/EIPS/eip-712`, 2018. Accessed: 2024.

[14] Vitalik Buterin. Merkle trees and merkle roots, 2014. Accessed: 2024.

[15] Chainlink. What are blockchain oracles? `https://chain.link/education/blockchain-oracles`, 2020.

[16] Opensea: Buy, sell, and discover rare digital items. https://opensea.io/. Accessed: 2024.

[17] Hayden Adams, Noah Zinsmeister, and Dan Robinson. Uniswap v2 core. https://uniswap.org/whitepaper.pdf, 2020.

[18] Guillermo Angeris and Tarun Chitra. An improved product market maker, based on the constant product formula. *arXiv preprint arXiv:2006.08806*, 2020.

[19] Artemij Fedosejev. React.js for the visual learner, 2015.

[20] MetaMask. Metamask user guide. https://metamask.io/faqs.html, 2021.

[21] Coinbase wallet: Your key to the world of crypto. https://wallet.coinbase.com/. Accessed: 2024.

[22] Michael B Jones, John Bradley, and Nat Sakimura. Json web token (jwt). https://tools.ietf.org/html/rfc7519, 2015.

[23] Stefan Tilkov and Steve Vinoski. Node.js: Using javascript to build high-performance network programs, 2010.

[24] Paul DuBois. *MySQL*. New Riders Publishing, 1999.

[25] Ethereum Foundation. Solidity programming language. https://docs.soliditylang.org/. Accessed: 2024.

[26] web3.js - ethereum javascript api. https://web3js.readthedocs.io/. Accessed: 2024.

[27] Ethers.js: Complete ethereum wallet implementation and utilities in javascript (and typescript). https://docs.ethers.io/v5/. Accessed: 2024.

[28] Truffle suite: A development environment, testing framework and asset pipeline for blockchains using the ethereum virtual machine (evm). https://www.trufflesuite.com/. Accessed: 2024.