Summer Training Report

on

Content Distribution Decentralized Application Platform

A Project Report submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by
Abdul Rahim
Roll no: CO20301

Under the supervision of
Dr. R.B. Patel
Department of Computer Science & Engineering
CCET (Degree Wing), Chandigarh

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)

Government Institute under Chandigarh (UT) Administration, Affiliated to Panjab University , Chandigarh

Sector-26, Chandigarh. PIN-160019

July, 2023

Department of Computer Sc. & Engineering

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled "CONTENT DISTRIBUTION DEENTRALISED APPLICATION PLATFORM", in fulfillment of the requirement for the award of the degree Bachelor of Engineering in Computer Science & Engineering, submitted in CSE Department, Chandigarh College of Engineering & Technology (Degree wing) affiliated to Punjab University, Chandigarh, is an authentic record of my/our own work carried out during my degree under the guidance of Dr. R.B. Patel. The work reported in this has not been submitted by me for award of any other degree or diploma.

Date : 21 December, 2023 Abdul Rahim

Place : Chandigarh CO20301

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)
Government Institute under Chandigarh (UT) Administration | Affiliated to Panjab University , Chandigarh
Sector-26, Chandigarh. PIN-160019 | Tel. No. 0172-2750947, 2750943
Website: www.ccet.ac.in | Email: principal@ccet.ac.in | Fax. No. :0172-2750872

Department of Computer Sc. & Engineering

CERTIFICATE

This is to certify that the Project work entitled "CONTENT DISTRIBUTION DEENTRALISED APPLICATION PLATFORM" submitted by **Abdul Rahim** bearing roll no. **CO20301** in fulfillment for the requirements of the award of Bachelor of Engineering Degree in Computer Science & Engineering at Chandigarh College of Engineering and Technology (Degree Wing), Chandigarh is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the project has not been submitted to any other University/Institute for the award of any Degree.

Date : 21 December, 2023
Place : Chandigarh

Dr. R.B. Patel
Deptt. of CSE
CCET(Degree Wing), Chandigarh

Department of Computer Sc. & Engineering

ACKNOWLEDGEMENT

It is my proud privilege and duty to acknowledge the help and guidance received from my professors in preparing this project. It would not have been possible to prepare this report in this form without their valuable help, cooperation, and guidance.

I feel deeply honoured in expressing my sincere thanks to “Dr. R.B. Patel” for giving me this opportunity and guiding me throughout my internship to get a good understanding of Blockchain Technology and valuable insights leading to matured clarity in the industry.

I wish to record my sincere gratitude to Faculty members, Mentors, and Academic cells for their constant support and encouragement in the preparation of this report. The guidance on internship opportunities were very helpful us in giving the necessary background information and inspiration.

Their contributions and technical support in preparing this report are greatly acknowledged. Last but not the least, we wish to thank our parents for financing our studies in this college as well as for constantly encouraging us to learn to engineer. Their personal sacrifice in providing this opportunity to learn engineering is gratefully acknowledged.

ABSTRACT

Piracy of digital content in the expanse of cyberspace has emerged as a formidable challenge, posing a significant obstacle for law enforcement agencies worldwide. Keeping track of copies of content published online is genetically impossible, since a copy is indistinguishable from the other, hence it is easy to copy and republish content illegally. Traditional methods, such as the use of 'registration key' systems for software, have aimed to curtail the circulation of unauthorized copies. However, the success of these approaches has been limited since authentication code for registration key can be changed in the software, also, such approaches are limited to software only.

In this project we demonstrate a novel approach of publishing content on the internet using blockchain technology. We bind articles with ERC 1155 token which helps us to keep track of published copies. While our illustrative example focuses on articles, it's essential to emphasize the versatility of this approach, as it can be extended to encompass a wide array of digital assets and content types. This report delves into the intricate details of our Content Distribution Decentralized Application Platform, elucidating its architecture, functionalities, and the transformative impact it holds in combating digital content piracy.

Table of Contents

Introduction

The vast expanse of the internet has significantly transformed the way we create, share, and consume digital content. However, this unprecedented freedom has also given rise to a pervasive and challenging issue – digital piracy. Piracy on the internet refers to the unauthorized reproduction and distribution of digital content, such as music, movies, software, and other creative works. This illicit activity poses multifaceted problems that reverberate across various sectors and impact both content creators and consumers. One of the central challenges stems from the inherent nature of digital assets, where perfect copies can be easily replicated and disseminated with minimal effort. Unlike physical goods, digital content lacks the constraints of scarcity, making it susceptible to widespread and indiscriminate duplication. This ease of reproduction not only devalues the intellectual property of creators but also disrupts established distribution models, affecting industries ranging from entertainment to software development.

Law enforcement agencies and content creators grapple with the complex task of identifying and prosecuting individuals engaged in digital piracy. The borderless nature of the internet makes tracking and apprehending offenders challenging, often resulting in a cat-and-mouse game between authorities and those seeking to exploit digital content without proper authorization. Furthermore, the financial implications of digital piracy are significant. Content creators, including musicians, filmmakers, authors, and software developers, rely on the revenue generated from their work. Piracy erodes this income stream, jeopardizing the livelihoods of creative professionals and stifling innovation by reducing the financial incentives for producing high-quality content. Traditional methods of combating piracy, such as employing digital rights management (DRM) technologies or issuing registration keys for software, have encountered limited success. These approaches often face circumvention by tech-savvy individuals, and their effectiveness diminishes over time as methods of unauthorized reproduction evolve.

Our goal in this project is to leverage blockchain technology to store and manage ownership information of articles. Blockchain uses decentralised ledger technology to store transaction related information. Although blockchain in its barebones form only stores transactions, but recent advancements in blockchain technology namely Ethereum uses smart contracts. These smart contracts are extremely versatile and can be used in a wide array of applications including management of ownership information. The network can be queried if a certain user owns an article using a set of addresses, one belongs to the user and the other to the article itself. Also, we can execute transactions on the network to change this ownership information.

Blockchain Technology

Introduction

In 2008, a pseudo-anonymous person by the name Satoshi Nakamoto proposed a peer-to-peer electronic cash system hence introducing us to blockchains. The proposed system would process transactions between parties without the need of any financial institution. In this type of system, the transactions are not verified by a single centralised entity like a bank but a lot of participant computers which run calculations separately to verify that nobody is cheating, these computers then vote if they think that transactions are valid. And if they agree and reach consensus, the transactions are validated.

A blockchain is a data-structure similar to a singly linked list, but instead of nodes they have blocks. A node may contain any information, but a block contains multiple timestamped data pieces often representing transactions. And in contrast to memory pointers pointing to next node in a linked list. They have block hashes (i.e. result of some hash function applied on a block) in each block which refers to previous block.

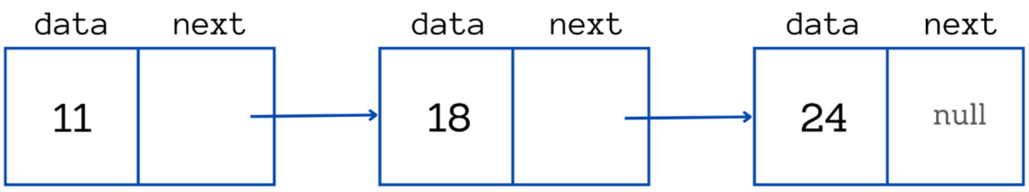


Fig. 1: A linked list

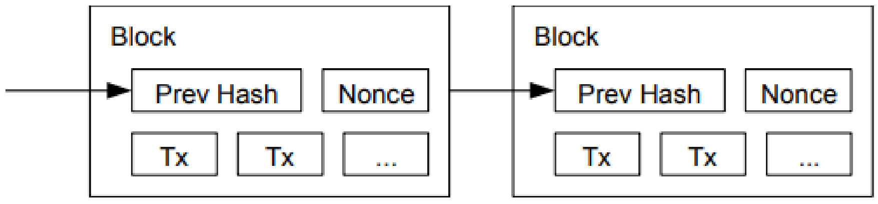


Fig. 2: A blockchain, each block contains multiple transactions, prev. hash (hash of last block) and other metadata

The benefit of having previous hashes in each block is that if let's say someone tries to tamper records than he must change nth block to transfer himself an arbitrary amount of money. But then the hashes of all the blocks after the nth block will change indicating that the records have been tampered with. This effect appears because, if the input data changes even slightly, its hash changes completely.

More formally, A blockchain is a public database that is shared across multiple computers across a network. For this reason, it is also called distributed ledger technology. A blockchain is decentralised distributed ledge that stores time series data usually related to transactions in blocks, these blocks are cryptographically connected to each other. Each block contains multiple transactions. Although blockchains can be used to store anything but we generally store things which require consensus (trusted agreement) usually monetary information for e.g. Alice has 50$ and Bob has 1$. A block contains many transactions. A Block is added at the head of chain at fixed intervals of time (called epoch). Data already added in a previous block cannot be changed without changing all subsequent blocks which require consensus of the entire network.

Every computer in the network must agree upon each new block and the chain as a whole. These computers are known as nodes. Nodes ensure that everyone interacting with the network has same data. To accomplish this, distributed agreement, blocks need a consensus mechanism. Which can be either proof-of-work or proof-of-stake. In case of Ethereum it is proof-of-stake and for bitcoin it is proof-of-work.

Merkel Trees

Since, we are storing blocks on each computer in the network redundantly, and in real world applications transactions occur at rates of hundreds of transactions per second. Although note that this does not relate to block creation time since that is called epoch which is usually 10-15 seconds. At each epoch all available transactions are compiled in a block and this block is added in the network to be verified by peers. Now we don't want to store each transaction in a block since that is costly in terms of space, we store them using a data structure called Merkel tree.

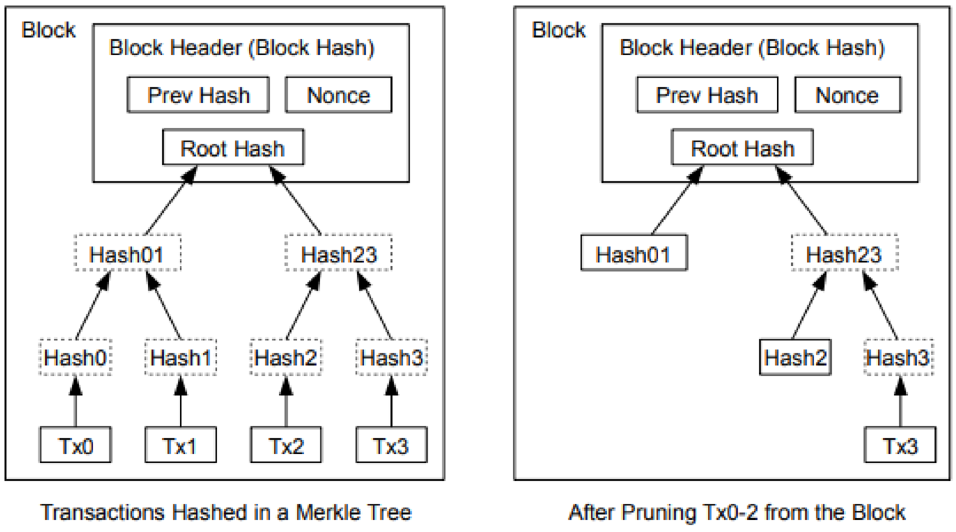


Fig. 3: Transaction are stored in merkel tree, with only the root hash stored in the block to save storage space

A merkel tree is a data structure that stores hashes in nodes, each parent node is the hash of its 2 child nodes. The leaves of merkel tree contain transactions. The benefit of this approach is that even if we remove a part of the tree for e.g. here, we have removed the left subtree, the root hash remains same. Hence, computers in the network need not store all the transactions saving disk space.

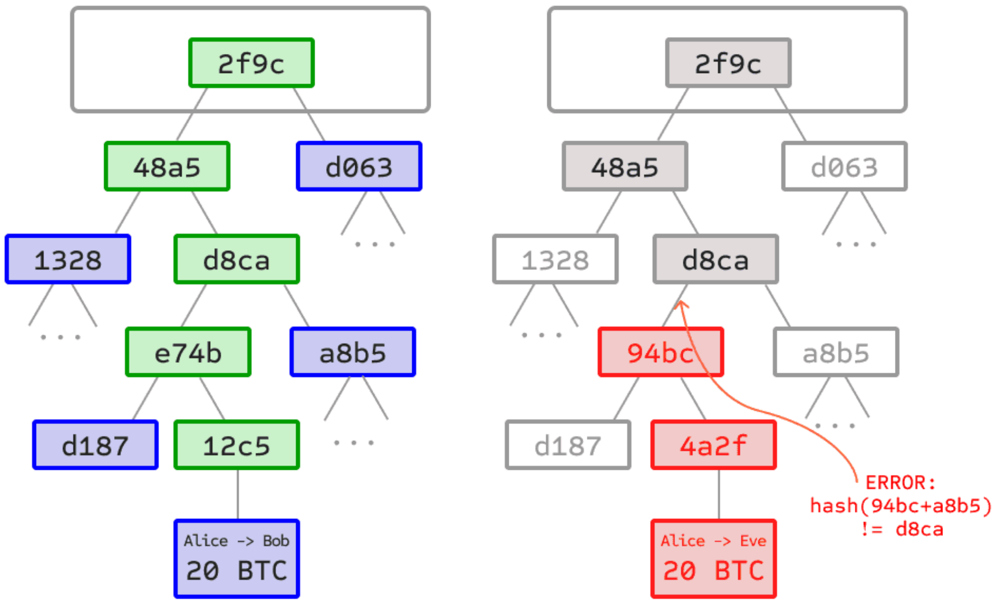


Fig. it suffices to present only a small number of nodes in a Merkle tree to give a proof of the validity of a branch, any attempt to change any part of the Merkle tree will eventually lead to an inconsistency somewhere up the chain

Mining Process and Consensus Mechanism

The consensus mechanism that bitcoin uses is called proof of work. The proof-of-work involves scanning for a value called nonce that when hashed (such as with SHA-256) with other block data, the hash begins with a predefined number of zero bits. The average work required is exponential in the number of zero bits required and can be verified by executing a single hash by other participants. This process is called mining. Miners incrementing a nonce in the block until a value is found that gives the block's hash the required zero bits.

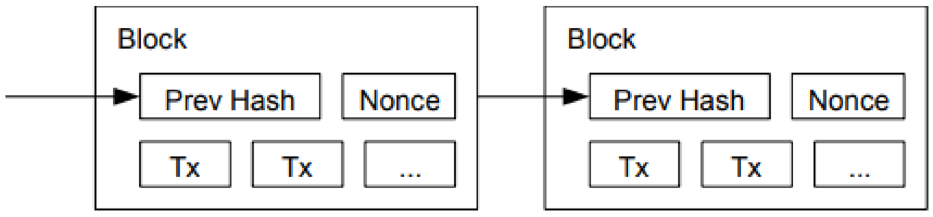


Fig. 4: Miners continuously change the nonce value until they obtain a hash required hash

The difficulty of mining is adjusted by changing the number of zeroes required at the beginning of the hash. This ensures that blocks are generated at a constant rate which is 10 minutes in case of bitcoin and 10-15 seconds in case of Ethereum.

After the CPU puts in the effort to meet the proof-of-work requirement, the block becomes resistant to changes. Changing the block would mean redoing the work, and since subsequent blocks are linked to it, altering one block requires redoing all the blocks that follow. If most of the computer processing power belongs to honest participants, their honest chain will grow faster, surpassing any competing chains. To alter a past block, an attacker would need to redo the proof-of-work for that block and all the subsequent blocks, then catch up with and surpass the work done by the honest participants.

As a convention, the first transaction in a block is a special transaction that initiates a new coin owned by the block creator. This provides an incentive for nodes to support the network and offers a way to initially circulate coins without a central authority. The continuous addition of a fixed number of new coins is similar to gold miners investing resources to introduce gold into circulation. In our case, it involves expending CPU time and electricity. The incentive can also be supported by transaction fees. Once a predetermined number of coins have entered circulation, the incentive can shift entirely to transaction fees, making the system completely free from inflation.

Another consensus mechanism that is used in blockchains is called proof of stake. Unlike Proof of Work (PoW), which relies on computational work and mining, PoS selects validators to create new blocks based on the amount of cryptocurrency they hold and are willing to "stake" as collateral. In a PoS system, participants, also known as validators, are required to lock up a certain amount of cryptocurrency as collateral – this is referred to as "staking." The idea is that validators with a higher stake have a higher chance of being chosen to create the next block. If a validator attempts to validate fraudulent transactions or compromise the network, they risk losing their staked coins. This economic incentive encourages responsible participation and discourages bad actors. PoS is considered more energy-efficient compared to PoW since it doesn't involve computing hashes repeatedly to find nonce, which often requires significant computational power.

Etherium Blockchain

Introduction

Introduced by Vitalik Buterin in 2015, Ethereum represents a decentralized, open-source blockchain that facilitates the creation of smart contracts and decentralized applications (DApps). Etherium is decentralised blockchain with smart contract functionality. Ether is the native cryptocurrency of the platform which allows anyone to deploy permanent and immutable decentralised applications onto it, with which users can interact. Decentralised financial applications provide financial instruments which do not directly rely on financial intermediaries like brokerages, exchanges, banks. These facilities borrowing against cryptocurrency or lending them out for interest. Many cryptocurrencies utilize the ERC 20 token standard on top of etherium blockchain. Buterin argued that blockchain technology could benefit from other applications besides money, it needed a more robust language for application development (Turing complete). Attaching real world assets such as stocks and property to the blockchain.

Ethereum, like Bitcoin, operates on blockchain technology, but it takes the concept a step further. While Bitcoin's scripting language is designed for specific transaction functionalities, Ethereum incorporates a Turing-complete programming language. This allows developers to create complex smart contracts—self-executing contracts with coded terms and conditions—enabling a broader spectrum of decentralized applications. Ether, the native cryptocurrency of the Ethereum platform, serves multiple purposes. Firstly, it acts as a fuel for executing operations on the Ethereum network, preventing misuse by requiring users to pay for computational resources. Secondly, Ether is used as an incentive for miners who validate transactions and secure the network. This dual-role mechanism provides economic stability to the Ethereum ecosystem. Ethereum's infrastructure supports the development of decentralized applications, offering a decentralized and secure environment for users. DApps leverage the Ethereum blockchain for transparent and trust-less execution of operations, ranging from gaming and social networking to decentralized exchanges.

Ethereum virtual machine is the runtime environment for transaction execution in etherium. EVM is stack based. EVM is deterministic, given a pre transaction state and a transaction, each node produces the same post transaction state, enabling network consensus. gas is unit of account and is used in calculation of transaction fee. Each type of operation which may be performed by the EVM is hardcoded with a certain amount of gas cost, which is intended to be roughly proportional to the monetary value of resources expended. if at any point the transaction does not have enough gas to perform the next operation, the transaction is reverted and the sender is refunded unused gas. EVM's instruction set is Turing complete. Ethereum's smart contracts are written in high level language (usually solidity), and then compiled down to EVM byte code which is then executed by every node in the blockchain to produce a new state upon which the transaction is successful.

Popular uses of etherium include creation of fungible tokens (ERC 20) for cryptocurrencies built on top of etherium blockchain, non-fungible tokens (ERC 720) for NFT artworks, and semi fungible tokens (ERC 1155) for selling items which may not be unique. Etherium allows for the creation of unique and indivisible tokens called non fungible tokens.

The components of Etherium blockchain:

1. Blockchain:

Think of the blockchain as a digital history book for Ethereum. It's a chain of blocks, and each block contains a reference to the one before it, creating a clear order of events.

2. ETH (Ether):

Ether is like the money of Ethereum. People use Ether to pay each other for getting things done on the platform, like running computer code.

3. EVM (Ethereum Virtual Machine):

The EVM is like a giant virtual computer that everyone using Ethereum agrees on. People can ask this computer to run different codes, and when it does, it changes its state.

4. Nodes:

Nodes are like the real-world computers that keep track of the EVM's state. They talk to each other to share information about what's happening on the EVM. Any user can ask these nodes to run some code.

5. Accounts:

Accounts are where Ether is stored. Users can create accounts, put Ether in them, and send Ether to others. All of this is kept in a big table that's part of the EVM's overall state.

6. Transactions:

A transaction is like a formal request for the EVM to do something. When this request is completed, it's called a transaction. Users can broadcast these requests, but for them to actually affect the EVM, they need to be checked, executed, and confirmed by another computer (node).

Some examples of transactions:

a. Sending Ether from my account to Alice's account.
b. Putting a piece of code into the EVM.
c. Running the code of a specific smart contract on the EVM.

7. Blocks:

Because there are lots of transactions, they're grouped together into blocks. Each block holds many transactions.

8. Smart Contracts:

Smart contracts are like reusable pieces of code that people can publish into the EVM. Anyone can ask the EVM to run this code, and because of this, they're also called Decentralized Apps or dapps. It's like having mini-programs on Ethereum for things like games, markets, or financial tools.

Decentralised Applications (Dapp)

A decentralized application (DApp) is like a special kind of app that works on a decentralized network. It combines a smart contract (a set of rules) with a user interface that you interact with. On Ethereum, these smart contracts are open and transparent, so you can use ones that others have made.

A DApp runs its backend code on a decentralized network of computers, unlike regular apps where the code is on central servers. DApps can have interfaces written in any language, just like regular apps. They can even use decentralized storage, making them more independent.

- Decentralized: DApps operate on Ethereum, a public and decentralized platform where no one person or group controls everything.
- Deterministic: DApps do the same thing no matter where they're used.
- Turing complete: DApps can do any action given the necessary resources.
- Isolated: DApps run in a special environment called Ethereum Virtual Machine, so if there's a mistake, it won't mess up the whole network.

Smart contracts are like the brains behind DApps. They are bits of code that live on the Ethereum blockchain and work exactly as they are programmed. Once they are on the network, you can't change them. DApps are decentralized because they follow the rules in these contracts, not one person or company.

Benefits of DApp Development:

- Zero Downtime: DApps on the blockchain are always available. Bad actors can't target individual DApps with attacks.
- Privacy: You don't need to share real-world identity for DApp use.
- Resistance to Censorship: No one can stop users from using DApps or interacting with the blockchain.
- Data Integrity: Data on the blockchain is secure and can't be tampered with.
- Trustless Computation: Smart contracts execute predictably without relying on a central authority.

Drawbacks of DApp Development:

- Maintenance: Changing DApps on the blockchain is hard, making updates challenging.
- Performance Overhead: Running every transaction on every computer for security makes the system slower.
- Network Congestion: Too many transactions from one DApp can slow down the whole network.
- User Experience: Setting up tools to interact with the blockchain securely can be complex.
- Centralization: Some DApps might end up looking like centralized services, losing the advantages of blockchain.

Etherium Virtual Machine

The Ethereum Virtual Machine (EVM) is like a powerful computer shared by thousands of connected computers running Ethereum software. It's not a physical entity but a crucial part of Ethereum that ensures consistent and unchangeable operation. While Bitcoin uses a distributed ledger to track transactions, Ethereum takes it a step further. It's like a distributed state machine, a big data structure that holds accounts, balances, and a machine state that changes according to rules. The EVM defines these rules.

The EVM acts like a math function: it takes an old state (S) and a set of transactions (T) as input and produces a new valid state (S'). In Ethereum, the state is a massive structure storing all accounts and balances.

$$T(S) \rightarrow S'$$

Transactions are signed instructions from accounts. There are two types: those resulting in message calls and those creating contracts. Contract creation leads to a new account executing smart contract bytecode when called by another account.

The EVM executes as a stack machine with a depth of 1024 items. Each item is a 256-bit word. During execution, it uses transient memory and storage trie for contracts. Smart contract bytecode consists of EVM opcodes performing operations like XOR, AND, ADD, etc., along with blockchain-specific operations.

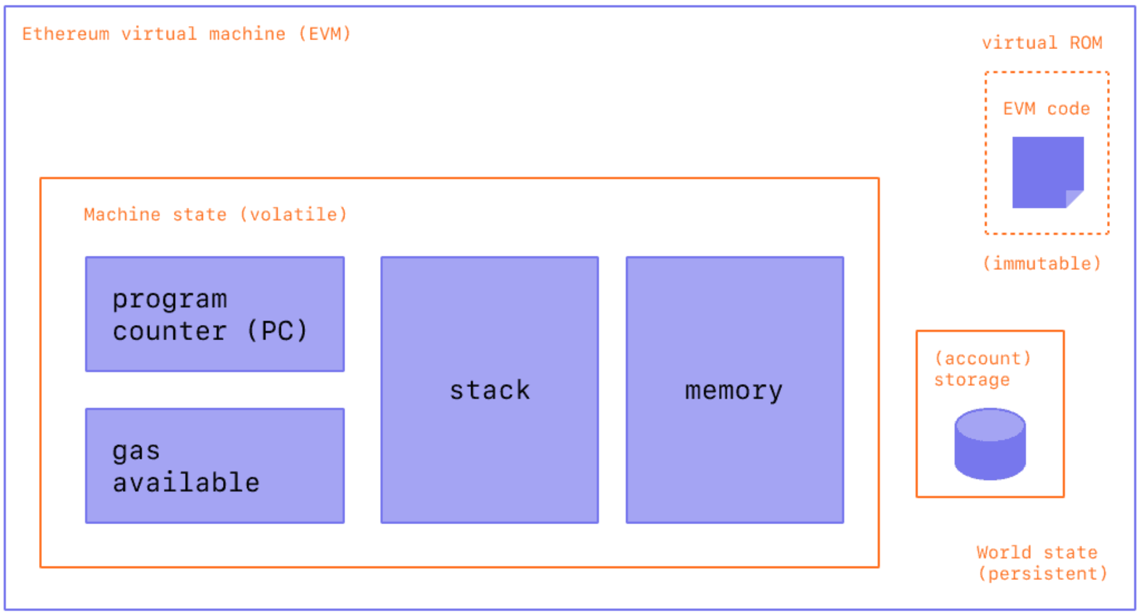


Fig. Block Diagram of Etherium Virtual Machine (EVM)

Tokens in Ethereum

In Ethereum, tokens are digital assets created and managed on the Ethereum blockchain. They represent value or ownership of specific assets and can be used for various purposes within the Ethereum ecosystem. There are several types of tokens, but three main categories are ERC-20 tokens, ERC-721 tokens (Non-Fungible Tokens - NFTs), and ERC-1155 tokens. Let's delve into each:

1. ERC-20 Tokens:

    - Function: ERC-20 is a widely adopted standard for fungible tokens on the Ethereum blockchain. Fungible tokens are interchangeable, meaning each token is identical to every other token of the same type.
    - Application: ERC-20 tokens have broad applications, including Initial Coin Offerings (ICOs), decentralized finance (DeFi) platforms, and various decentralized applications (dApps). They are used for creating utility tokens, stablecoins, and representing shares or ownership in projects.

    Key Characteristics:

    - Transferability: Each token is identical and can be exchanged on a one-to-one basis.
    - Divisibility: ERC-20 tokens are divisible into smaller units, allowing for microtransactions.
    - Interoperability: ERC-20 tokens can be easily traded on decentralized exchanges (DEXs) that support the standard.

2. ERC-721 Tokens (Non-Fungible Tokens - NFTs):

    - Function: ERC-721 is a standard for creating unique, non-fungible tokens. Each token is distinct and cannot be replaced or exchanged on a one-to-one basis with other tokens.
    - Application: NFTs are used to represent ownership or proof of authenticity for digital or physical assets. They find applications in digital art, collectibles, virtual real estate, and gaming items.

    Key Characteristics:

    - Uniqueness: Each token has a distinct value, making it suitable for representing individual assets.
    - Ownership: ERC-721 tokens represent ownership of a specific asset, providing a transparent and secure way to track ownership history.
    - Indivisibility: NFTs cannot be divided into smaller units, maintaining the uniqueness of each token.

3. ERC-1155 Tokens:

    - Function: ERC-1155 is a multi-token standard that allows a single contract to manage both fungible and non-fungible tokens. It provides flexibility for developers to create different types of assets within the same contract.
    - Application: ERC-1155 tokens are used in applications where a combination of fungible and non-fungible assets is required, such as in gaming, virtual goods, and tokenized assets.

    Key Characteristics:

    - Efficiency: ERC-1155 is more gas-efficient than deploying separate contracts for each token type.
    - Versatility: It allows the creation of a wide range of assets within a single contract, reducing complexity.
    - Cost-Effective: ERC-1155 tokens can be more cost-effective in terms of transaction fees compared to managing multiple contracts.

Overview of technologies used

Unity Engine

Unity is a powerful and versatile game development engine used to create a wide range of interactive experiences, including video games, simulations, architectural visualizations, and more. It provides developers with a comprehensive set of tools and features that simplify the process of designing, building, and deploying applications across multiple platforms.

In our project we have built our web application in unity, unity was chosen because I am already familiar with it and also because it is simple to work with. It treats components in software as game objects. Then C# scripts can be attached to these game objects to define their behaviour. There are also various kind of animations that you can do with it. So, in our applications we implemented articles as game objects. The scripts attached to these game objects decided weather they are available to the user or not based on ownership, to do this these scripts used thirdweb SDK. You can directly import thirdweb SDKs code into your script. And then you can use its functions. Unity is also an excellent tool to build user interfaces. These are built using a special type of game object called canvas.

Unity supports cross-platform development, allowing developers to create applications for various platforms, including PC, mobile devices (iOS and Android), consoles, augmented reality (AR), virtual reality (VR), and web browsers. Unity employs a powerful graphics engine capable of rendering high-quality 2D and 3D graphics. It supports advanced features like real-time global illumination, high-dynamic-range rendering, and post-processing effects. The built-in physics engine enables the simulation of realistic physics interactions within the game environment. This is crucial for creating lifelike animations, character movements, and object interactions. Unity Asset Store is an extensive marketplace where developers can find pre-built assets, tools, and plugins to enhance their projects. This accelerates development by providing ready-made solutions for common tasks.

Unity uses the C# programming language for scripting. C# is widely used in the software development industry, and its integration with Unity allows for efficient and structured coding. Unity includes a robust animation system that facilitates the creation of complex character animations, cinematic sequences, and interactive cutscenes. It supports both 2D and 3D animations. Unity provides networking capabilities for multiplayer game development. It supports various networking architectures, including client-server and peer-to-peer models. Unity includes features for collaboration and version control, allowing multiple developers to work on the same project simultaneously. It integrates with version control systems like Git. Developers import assets such as 3D models, textures, audio files, and animations. These assets are organized in the project hierarchy.

Unity's physics engine is utilized to simulate realistic interactions between objects. Developers can apply colliders, rigidbodies, and joints to achieve desired physics behavior. Developers write scripts in C# to define the behavior of game objects. Unity's scripting API provides access to various functions and events. Optimization is performed to ensure smooth performance on target platforms. This includes optimizing graphics, code, and other resources. Developers build the project for specific platforms. Unity generates executable files or packages that can be deployed to devices or distributed through app stores. Unity has a large and active community of developers, artists, and designers. The Unity Asset Store plays a significant role in the ecosystem by offering a vast selection of assets, plugins, and tools created by both Unity Technologies and third-party developers. This collaborative environment fosters knowledge-sharing and accelerates development through the reuse of assets and solutions. Unity's versatility, ease of use, and extensive feature set make it a popular choice for game development and interactive applications. Whether creating a small indie game or a large-scale enterprise application, Unity provides the tools and resources needed to bring creative visions to life.

We have used unity’s webGL platform functionality to deploy our application on the web. Unity's WebGL (Web Graphics Library) deployment option enables developers to export and run their Unity projects directly in web browsers without the need for plugins. This feature is particularly advantageous for creating online games, interactive simulations, and applications that users can access through standard web browsers. WebGL builds from Unity are essentially web-compatible versions of the Unity runtime, and they leverage standard web technologies such as HTML5, JavaScript, and WebGL itself.

When developers choose to build for WebGL, Unity compiles the game's code into JavaScript, which can be executed within a web browser. This approach eliminates the need for users to install additional plugins or applications to run Unity content, enhancing accessibility and user engagement. The WebGL deployment option supports a wide range of browsers, including Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge, making Unity projects widely accessible across various platforms and devices.

Unity's WebGL builds retain the graphical and interactive capabilities of the original projects, allowing developers to deliver rich, immersive experiences directly through a web browser. This includes support for 3D graphics, animations, physics simulations, and other Unity features. However, it's essential for developers to consider performance optimizations when targeting the web platform, as browser-based execution imposes certain limitations compared to native applications.

Unity provides various settings and options during the WebGL export process to tailor the build to specific requirements. Developers can adjust compression settings, choose the level of graphics quality, and manage other parameters to balance performance and visual fidelity. Additionally, Unity's WebGL builds allow for integration with web-based APIs, enabling developers to incorporate online features, multiplayer functionality, and other web services into their projects seamlessly.

One of the notable advantages of Unity's WebGL deployment is its ability to leverage the extensive Unity Asset Store. Developers can access a wealth of assets, plugins, and tools from the Asset Store, enhancing their web-based projects with ready-made content and functionalities. This contributes to a streamlined development process and enables developers to focus on creating compelling user experiences rather than building everything from scratch.

ThirdWeb SDK

The integration of ThirdWeb API within Unity projects encompasses a robust set of features that seamlessly introduce blockchain capabilities into the gaming environment. This technological endeavour involves the utilization of key components outlined in the API documentation. The ThirdwebManager, operating as a MonoBehaviour, acts as a central hub for managing the ThirdwebSDK. By attaching this script to a GameObject and employing DontDestroyOnLoad(), the SDK persists throughout the game's lifecycle. It offers a customizable and convenient means of initializing and configuring the SDK based on specified settings.

Serving as the entry point to the Unity SDK, the ThirdwebSDK provides developers with access to a multitude of functionalities. Whether initialized through the ThirdwebManager or instantiated manually, the SDK opens avenues for diverse blockchain-related operations within Unity projects.

The API introduces various Prefabs, including the Connect Wallet Button, NFT Renderer, and Contract Deployer. These prefabs streamline common operations such as wallet connection, NFT rendering, and smart contract deployment, simplifying the integration of blockchain features into Unity applications. The documentation provides insight into generic methods for reading and writing smart contracts. The Read and Write methods facilitate interaction without relying on specific extensions. Extensions, represented by Solidity interfaces, unlock additional functionalities based on implemented interfaces such as ERC721, ERC1155, and ERC20.

We have used ERC1155 token standard from the arsenal of options thirdweb sdk provides us, the reason to choose ERC1155 tokens is that these are semi-fungible tokens, what it means is that they are not unique like ERC721 tokens, hence their copies are indistinguishable. Which is perfect to distribute something like a book or a software copy.

Specific functionalities for ERC1155, such as checking NFT balances, transferring NFTs between wallets, and managing approval statuses, are detailed. Methods like TotalCount and TotalSupply provide a comprehensive view of NFT counts and total supplies in a collection.

When employing the Marketplace smart contract, developers gain access to additional top-level functionalities. By creating a contract instance and specifying the marketplace contract type, Unity projects can benefit from specific marketplace features. Contract Events guide developers on retrieving occurrences of specific events emitted during a defined time period. Wallet Actions encompass operations like authentication, wallet connection and disconnection, balance retrieval, and more.

ThirdWeb also provides options to integrate with a wide range of modern crypto wallets such as metamask. In our project we would primarily be testing on metamask although you can use any wallet of your choice. MetaMask is a browser extension and cryptocurrency wallet that allows users to securely manage Ethereum-based assets and interact with decentralized applications (DApps) directly through their web browser. It simplifies blockchain interactions, offers user-friendly features, and supports custom networks and token management.

Ethereum and ERC1155 token standard

We have already discussed extensively about blockchains and etherium. The reason why we choose etherium is because of smart contract functionality. Which allows us to implement custom property. For example, consider a hypothetical asset, the owner ship of this asset can be implemented in the blockchain using ERC 1155 token standard. The hypothetical asset is just an abstraction, so now you can imagine to represent anything in place of this hypothetical asset.

In our project we have used goerli testnet, which is a dummy blockchain built for testing purposes. It is supposed to test applications to be deployed on etherium for debugging purposes. Hence it mimics functionality of etherium such as smart contract functionality. Although one might change the network with just click of a button, but the reason developers use test net is because the currency used in it is not real hence, they can experiment and test their code. Goerli test net provides all the functionality that one might use on etherium hence making it one of the most reliable test net available in the market.

Design and Architecture of the system

Understanding transaction execution on blockchain

Developers commonly implement Ethereum transactions using the Web3.js library, the process begins with the initialization of a Web3 instance, a JavaScript library commonly employed for Ethereum blockchain interaction. This instance connects to an Ethereum node, often hosted by services like Infura or MetaMask. Developers initiate transactions by creating a transaction object containing crucial details, such as the sender's and recipient's addresses, the amount of Ether to be transferred, and gas-related parameters like gas price and limit. Prior to transaction execution, the sender's Ethereum account needs to be unlocked using the private key associated with that account.

Subsequently, the transaction undergoes the signing process, where the transaction object is signed using the sender's private key. This step ensures the integrity and authenticity of the transaction. The signed transaction object is then submitted to the Ethereum network using the sendSignedTransaction method provided by Web3.js. This initiates the transaction process on the blockchain.

During this process, developers can implement event listeners to capture essential events such as transactionHash, receipt, and error. The transactionHash event provides the unique identifier for the transaction on the Ethereum blockchain, allowing users to track its status on blockchain explorers. The receipt event furnishes additional details about the completed transaction, including the block number, gas used, and, in the case of smart contract deployment, the contract address. On the other hand, the error event is triggered if any issues or errors arise during the transaction execution.

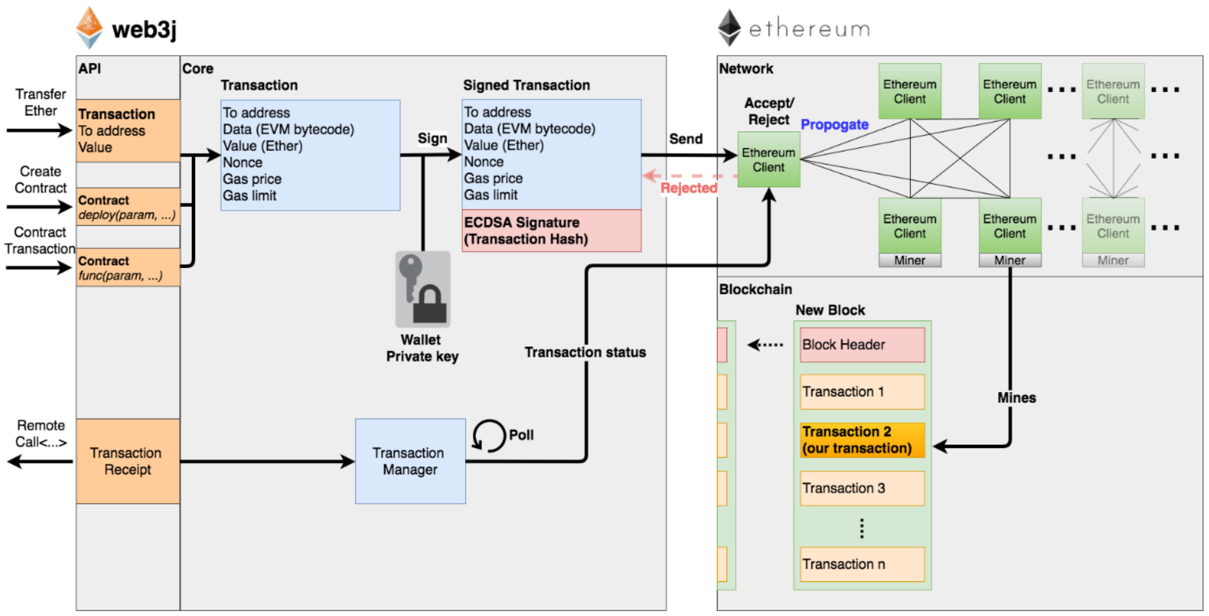


Fig. Transaction execution block diagram

Transaction management in our system

Interaction with the blockchain in our project is handled by thirdweb SDK. Under the hood it also works on model discussed above. The thirdweb-unity-bridge.js is a JavaScript file that handles integration of unity engine with etherium network end to end. It provides functionalities, such as checking NFT balances, transferring NFTs between wallets, and managing approval statuses. Methods like TotalCount and TotalSupply provide a comprehensive view of NFT counts and total supplies in a collection.

The benefit of this approach is that the functionality of blockchain integration is abstracted away. Hence, we could give more of our time and effort to designing the system, researching, organising the components and writing optimal scripts for the components.

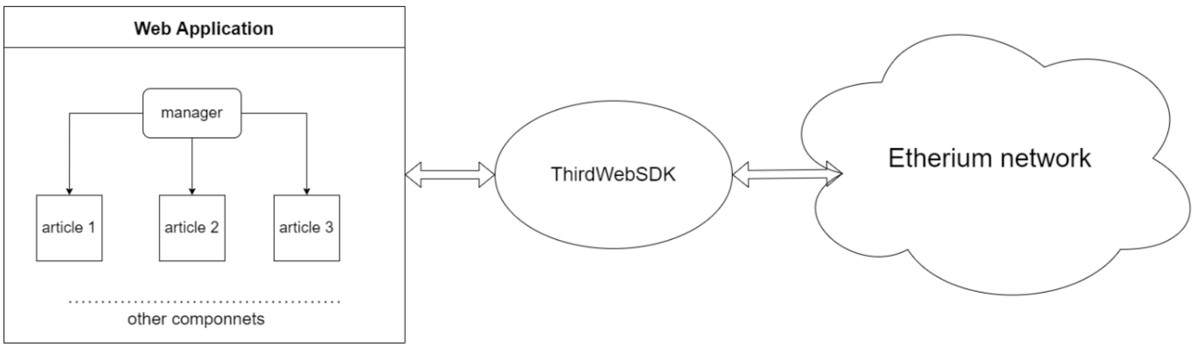
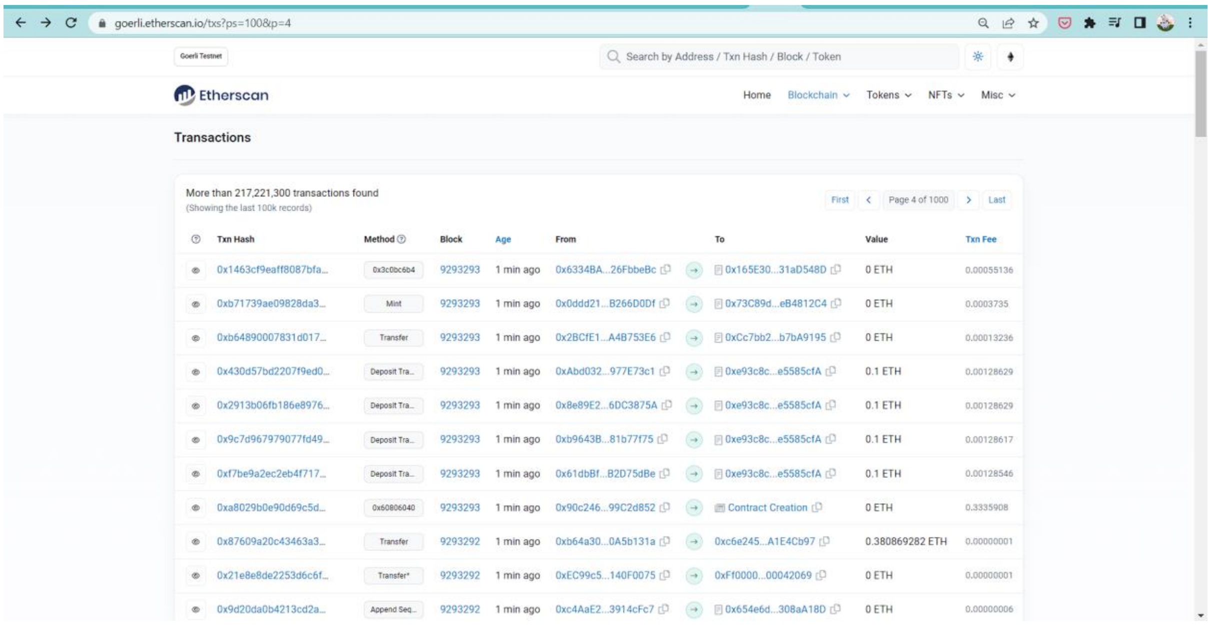


Fig. Simplified block diagram highlighting transaction management on the blockchain

Results

Deploying Smart Contract and Minting NFTs

EtherScan is a blockchain explorer for the Ethereum network, allowing users to explore and analyze transactions, addresses, and smart contracts. It provides detailed information about Ethereum addresses, transaction history, token details, and network statistics. In our project we will use ether scan to verify that our transactions are executing on the network.



goerli.etherscan.io/txs?ps=100&p=4
Goerli Testnet
Search by Address / Txn Hash / Block / Token
Etherscan
Home
Blockchain
Tokens
NFTs
Misc

Transactions

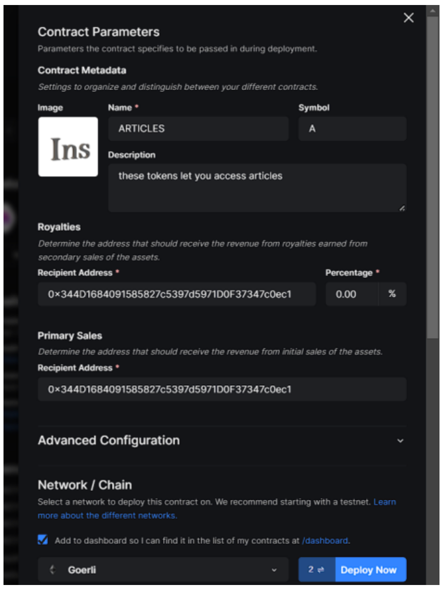
More than 217,221,300 transactions found
(Showing the last 100k records)

First | < | Page 4 of 1000 | > | Last

Txn Hash	Method	Block	Age	From	To	Value	Txn Fee
0x1463cf9eaff8087bfa...	0x3c0bc6b4	9293293	1 min ago	0x6334BA...26FbbeBc	0x165E30...31aD548D	0 ETH	0.00055136
0xb71739ae09828da3...	Mint	9293293	1 min ago	0x0ddd21...B266D0Df	0x73C89d...eB4812C4	0 ETH	0.0003735
0xb64890007831d017...	Transfer	9293293	1 min ago	0x2BCfE1...A4B753E6	0xCc7bb2...b7bA9195	0 ETH	0.00013236
0x430d57bd2207f9ed0...	Deposit Tra...	9293293	1 min ago	0xAbd032...977E73c1	0xe93c8c...e5585cfA	0.1 ETH	0.00128629
0x2913b06fb186e8976...	Deposit Tra...	9293293	1 min ago	0x8e89E2...6DC3875A	0xe93c8c...e5585cfA	0.1 ETH	0.00128629
0x9c7d967979077fd49...	Deposit Tra...	9293293	1 min ago	0xb9643B...81b77f75	0xe93c8c...e5585cfA	0.1 ETH	0.00128617
0xf7be9a2ec2eb4f717...	Deposit Tra...	9293293	1 min ago	0x61dbBf...B2D75dBe	0xe93c8c...e5585cfA	0.1 ETH	0.00128546
0xa8029b0e90d69c5d...	0x60806040	9293293	1 min ago	0x90c246...99C2d852	Contract Creation	0 ETH	0.3335908
0x87609a20c43463a3...	Transfer	9293292	1 min ago	0xb64a30...0A5b131a	0xc6e245...A1E4Cb97	0.380869282 ETH	0.00000001
0x21e8e8de2253d6c6f...	Transfer*	9293292	1 min ago	0xEC99c5...140F0075	0xFf0000...00042069	0 ETH	0.00000001
0x9d20da0b4213cd2a...	Append Seq...	9293292	1 min ago	0xc4AaE2...3914cFc7	0x654e6d...308aA18D	0 ETH	0.00000006

Fig. EtherScan is a tool that gives information about transactions in real time, available at goerli.etherscan.io

The ERC1155 smart contract is deployed through thirdWeb contract deployment functionality.



Contract Parameters

Parameters the contract specifies to be passed in during deployment.

Contract Metadata

Settings to organize and distinguish between your different contracts.

Image

Name *: ARTICLES

Symbol: A

Description: these tokens let you access articles

Royalties

Determine the address that should receive the revenue from royalties earned from secondary sales of the assets.

Recipient Address *: 0×344D1684091585827c5397d5971D0F37347c0ec1

Percentage *: 0.00 %

Primary Sales

Determine the address that should receive the revenue from initial sales of the assets.

Recipient Address *: 0×344D1684091585827c5397d5971D0F37347c0ec1

Advanced Configuration

Network / Chain

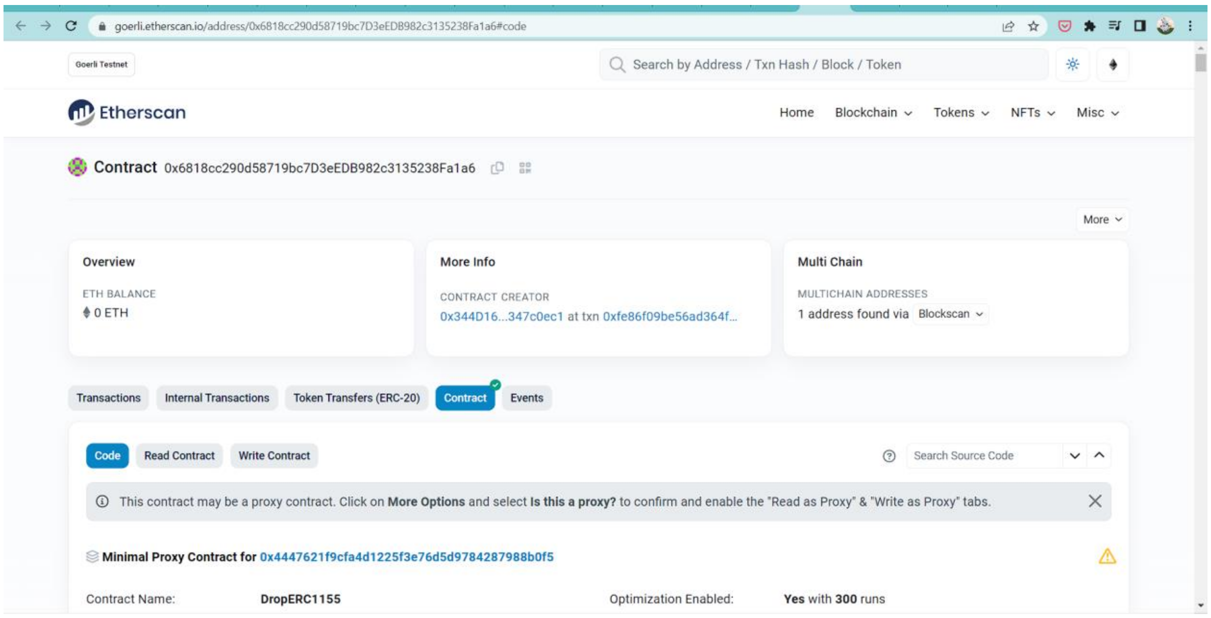
Select a network to deploy this contract on. We recommend starting with a testnet. Learn more about the different networks.

[✓] Add to dashboard so I can find it in the list of my contracts at /dashboard.

Goerli

2 ⇄ Deploy Now

Fig. We deploy our ERC1155 smart contract, the tokens would be encapsulated in this contract


goerli.etherscan.io/address/0x6818cc290d58719bc7D3eEDB982c3135238Fa1a6#code

Goerli Testnet

Search by Address / Txn Hash / Block / Token

Etherscan

Home Blockchain Tokens NFTs Misc

Contract 0x6818cc290d58719bc7D3eEDB982c3135238Fa1a6

More

Overview	More Info	Multi Chain
ETH BALANCE	CONTRACT CREATOR	MULTICHAIN ADDRESSES
0 ETH	0x344D16...347c0ec1 at txn 0xfe86f09be56ad364f...	1 address found via Blockscan

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

Code | Read Contract | Write Contract

Search Source Code

This contract may be a proxy contract. Click on **More Options** and select **Is this a proxy?** to confirm and enable the "Read as Proxy" & "Write as Proxy" tabs.

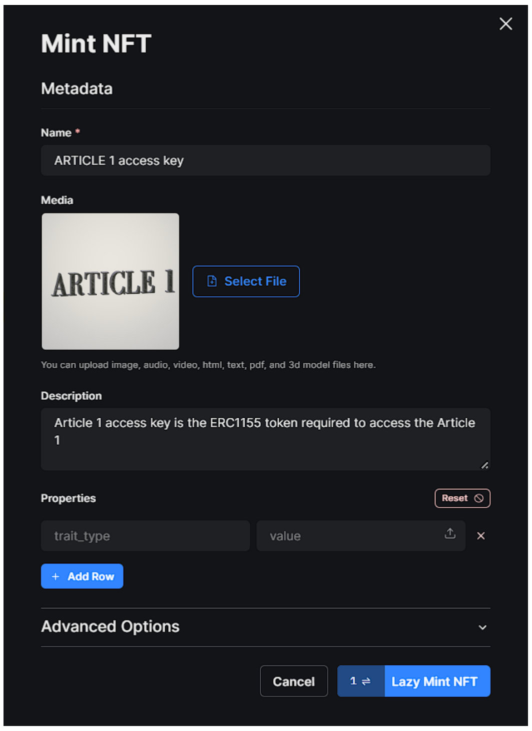
Minimal Proxy Contract for 0x4447621f9cfa4d1225f3e76d5d9784287988b0f5

Contract Name: **DropERC1155**

Optimization Enabled: **Yes** with **300** runs

Fig. Our smart contract has been deployed onto the network

Now we create 3 NFTs that will represent out articles. The ownership to these NFTs will act as the access keys in the application.


Mint NFT

Metadata

Name *

ARTICLE 1 access key

Media

Select File

You can upload image, audio, video, html, text, pdf, and 3d model files here.

Description

Article 1 access key is the ERC1155 token required to access the Article 1

Properties

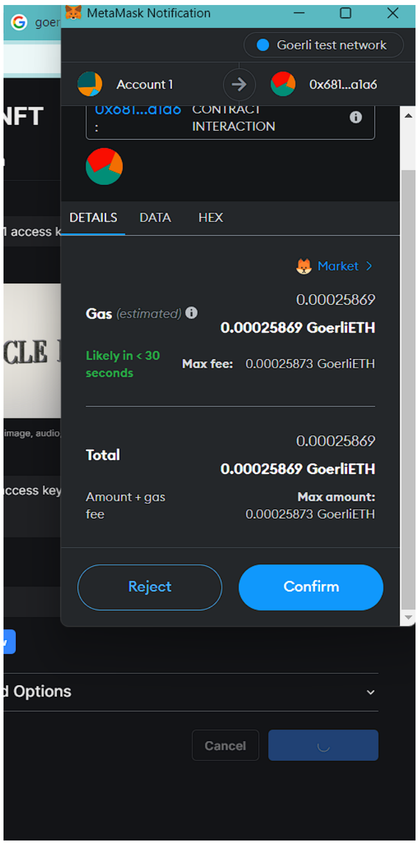
Reset

trait_type value

Add Row

Advanced Options

Cancel | 1 | Lazy Mint NFT


MetaMask Notification

Goerli test network

Account 1 → 0x681...a1a6

0x681...a1a6 : CONTRACT INTERACTION

DETAILS DATA HEX

Market

Gas (estimated) 0.00025869
0.00025869 GoerliETH

Likely in < 30 seconds — Max fee: 0.00025873 GoerliETH

Total 0.00025869
0.00025869 GoerliETH

Amount + gas fee — Max amount: 0.00025873 GoerliETH

Reject | Confirm

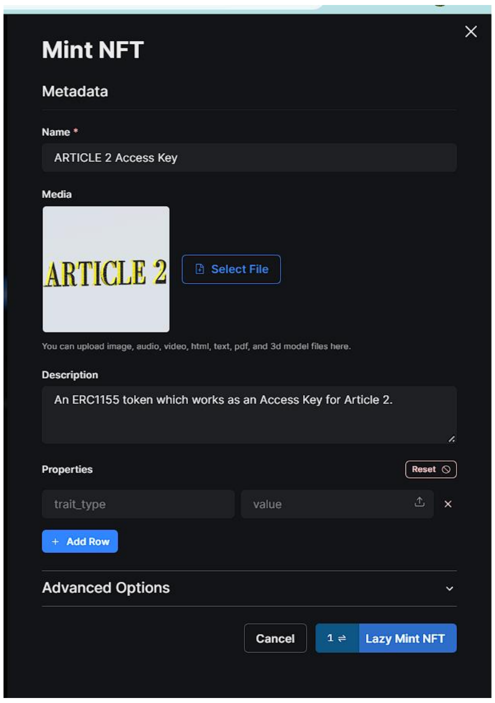
Options

Cancel

Fig. NFT for article1 is being minted here

Fig. Confirm the transaction to mint NFT

Similarly, other 2 NFTs are also minted. For each minted you can check transactions on etherscan.



Mint NFT

Metadata

Name *

ARTICLE 2 Access Key

Media

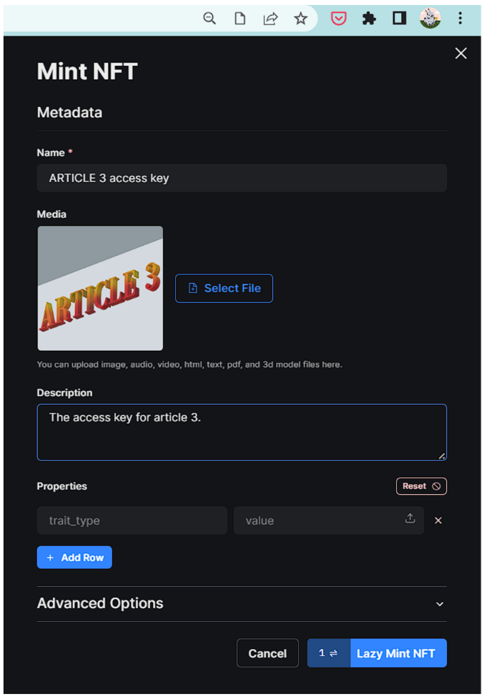
You can upload image, audio, video, html, text, pdf, and 3d model files here.

Description

An ERC1155 token which works as an Access Key for Article 2.

Properties

Fig. Minting NFT for article 2



Mint NFT

Metadata

Name *

ARTICLE 3 access key

Media

You can upload image, audio, video, html, text, pdf, and 3d model files here.

Description

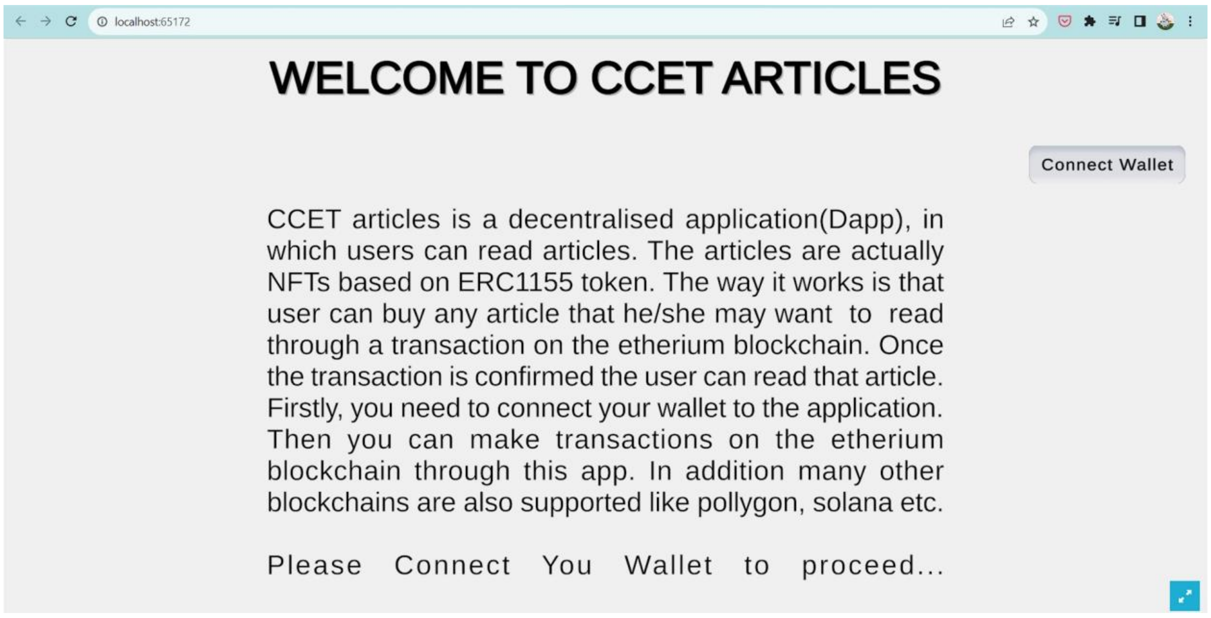
The access key for article 3.

Properties

Fig. Minting NFT for article 3

Accessing the articles in application

Now that our NFTs bounded to our articles are created, its time to open the application to see how that looks like. Like any other decentralised application, you need to connect your metamask wallet to interact with the applications. This helps in acquiring your wallet address.



localhost:65172

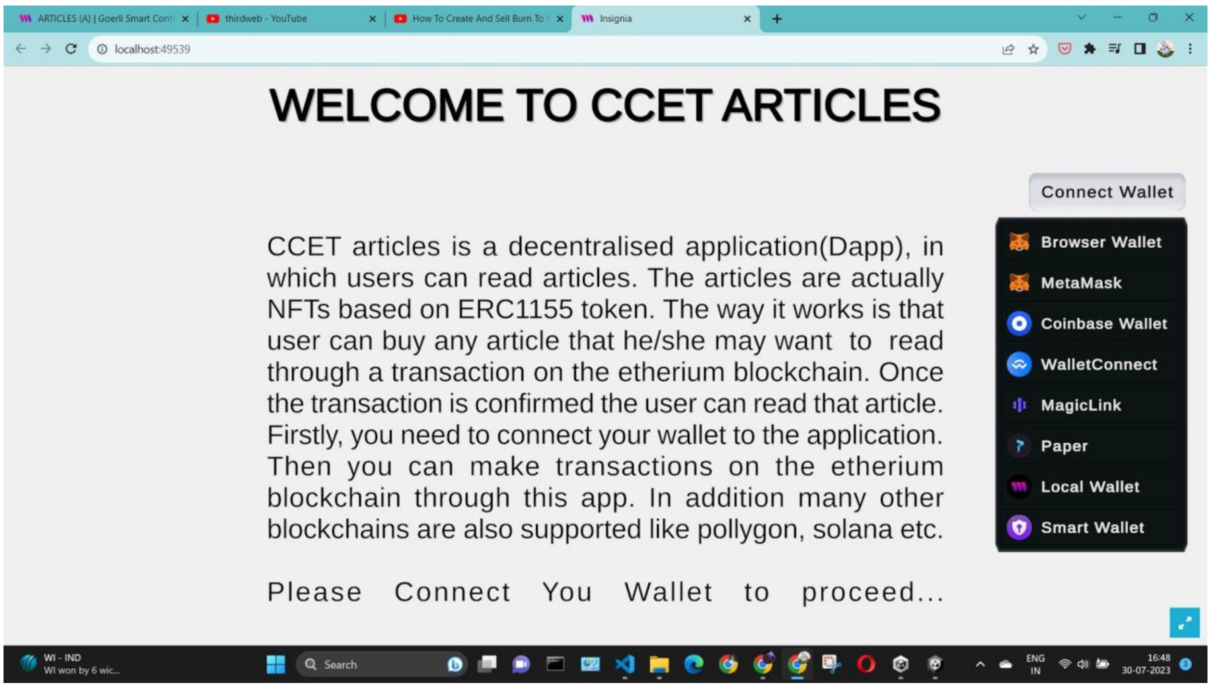
WELCOME TO CCET ARTICLES

CCET articles is a decentralised application(Dapp), in which users can read articles. The articles are actually NFTs based on ERC1155 token. The way it works is that user can buy any article that he/she may want to read through a transaction on the etherium blockchain. Once the transaction is confirmed the user can read that article. Firstly, you need to connect your wallet to the application. Then you can make transactions on the etherium blockchain through this app. In addition many other blockchains are also supported like pollygon, solana etc.

Please Connect You Wallet to proceed...

Fig. This is the user interface of the application.

Press on the connect wallet button and connect to the wallet of your choice.



WELCOME TO CCET ARTICLES

CCET articles is a decentralised application(Dapp), in which users can read articles. The articles are actually NFTs based on ERC1155 token. The way it works is that user can buy any article that he/she may want to read through a transaction on the etherium blockchain. Once the transaction is confirmed the user can read that article. Firstly, you need to connect your wallet to the application. Then you can make transactions on the etherium blockchain through this app. In addition many other blockchains are also supported like pollygon, solana etc.

Please Connect You Wallet to proceed...

Fig. Choose your prefferred wallet, you will be prompted to fill in your password

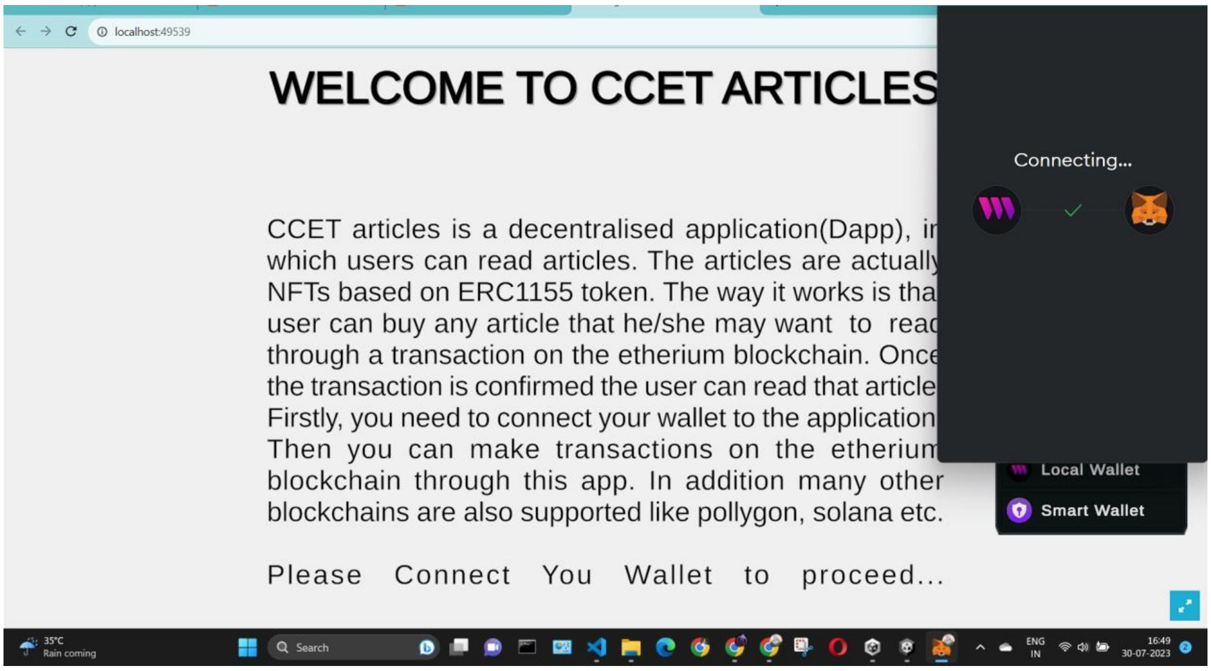
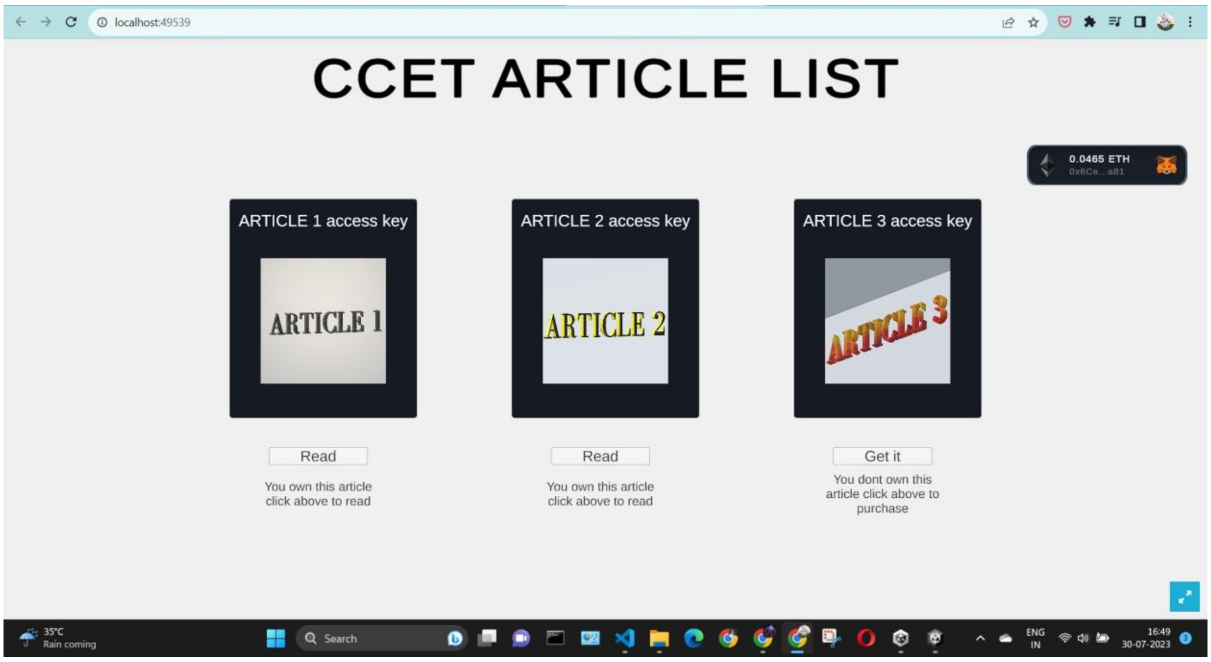


Fig. After successful connection you will be directed to article listing page

Then you are redirected to article listing screen. Here all the available articles are listed. The articles you own would have a ‘Read’ button and the articles which do not belong to your wallet address appear with a ‘Get it’ button.

This ownership information is queried live from the network to confirm that you have access to all items belonging to you.



CCET ARTICLE LIST

ARTICLE 1 access key

Read

You own this article click above to read

ARTICLE 2 access key

Read

You own this article click above to read

ARTICLE 3 access key

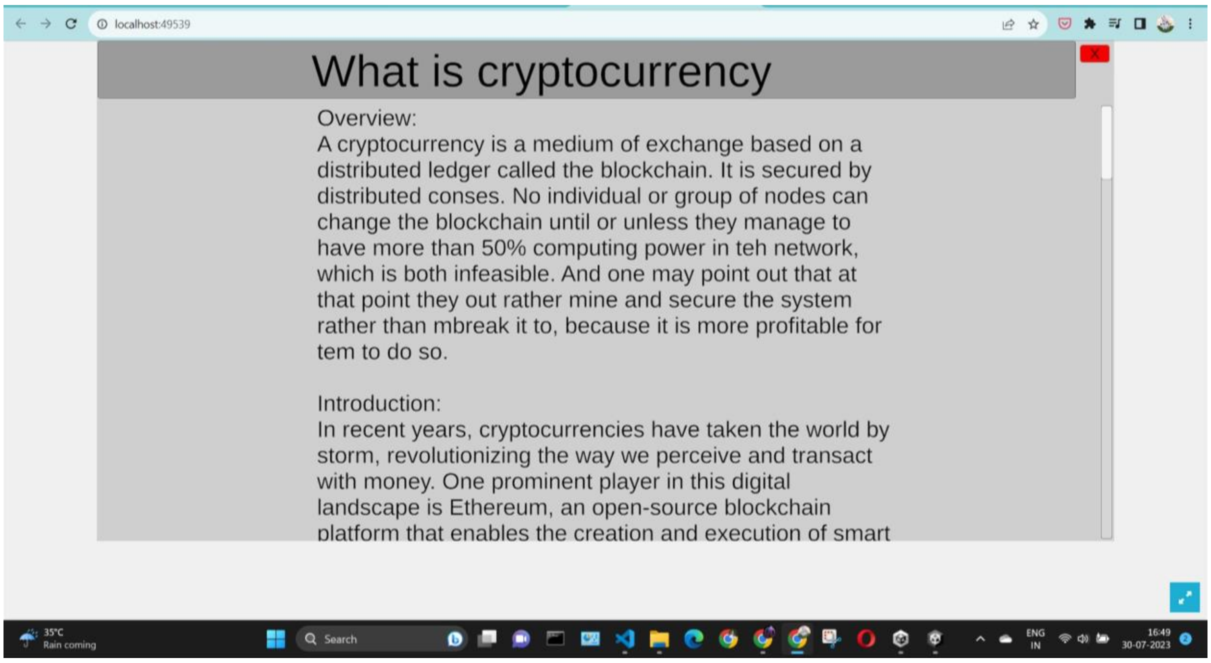
Get it

You dont own this article click above to purchase

Fig. Article listing screen

articles you own would have a ‘Read’ button and the articles which do not belong to your wallet address appear with a ‘Get it’ button.

Click the ‘Read’ button to open the articles you own.



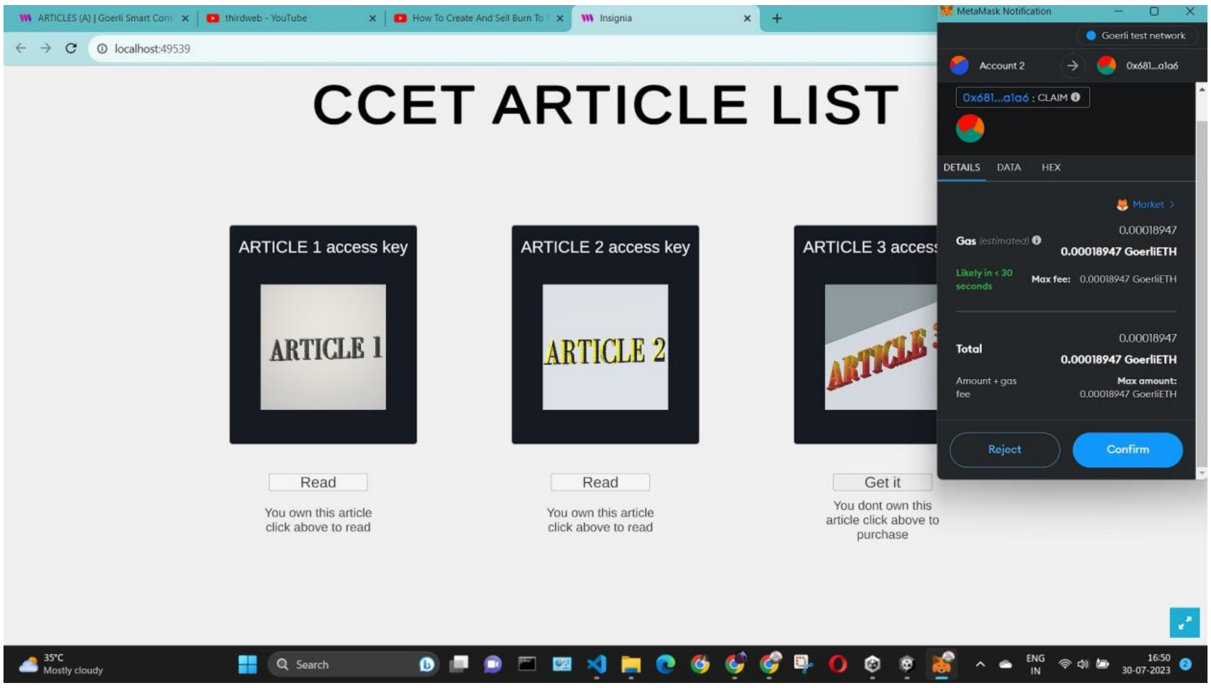
What is cryptocurrency

Overview:
A cryptocurrency is a medium of exchange based on a distributed ledger called the blockchain. It is secured by distributed conses. No individual or group of nodes can change the blockchain until or unless they manage to have more than 50% computing power in teh network, which is both infeasible. And one may point out that at that point they out rather mine and secure the system rather than mbreak it to, because it is more profitable for tem to do so.

Introduction:
In recent years, cryptocurrencies have taken the world by storm, revolutionizing the way we perceive and transact with money. One prominent player in this digital landscape is Ethereum, an open-source blockchain platform that enables the creation and execution of smart

Fig. On clicking Read button the article opens. You can also click the blue maximize window button for maximum view

For the articles that you do not own you can click on the ‘Get it’ button to purchase it, by giving a small amount of ether. For example, in this demonstration, we buy the third article which is about artificial intelligence.



CCET ARTICLE LIST

ARTICLE 1 access key
Read
You own this article click above to read

ARTICLE 2 access key
Read
You own this article click above to read

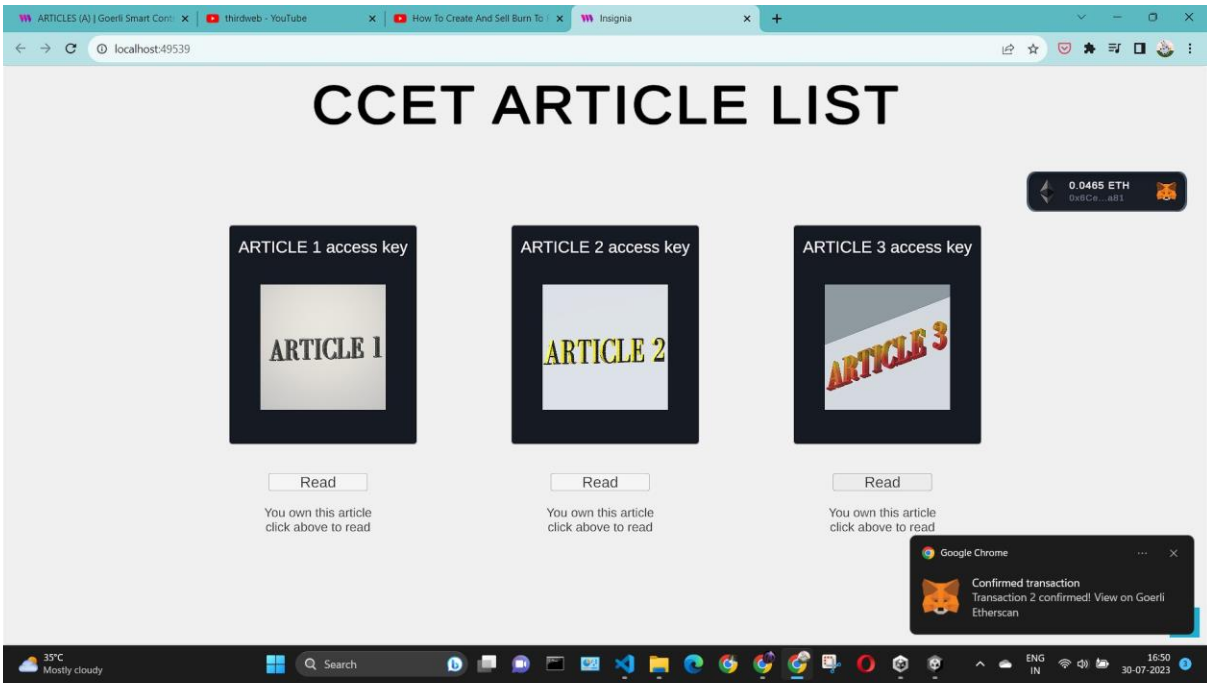
ARTICLE 3 access
Get it
You dont own this article click above to purchase

Gas (estimated) 0.00018947
0.00018947 GoerliETH
Likely in < 30 seconds
Max fee: 0.00018947 GoerliETH

Total 0.00018947
0.00018947 GoerliETH
Amount + gas fee
Max amount: 0.00018947 GoerliETH

Fig. On clicking Get it button on article3 a transaction confirmation popup will appear

Click on conform transaction, within few seconds transaction will conform and the ‘Get it’ button will change to ‘Read’ button.

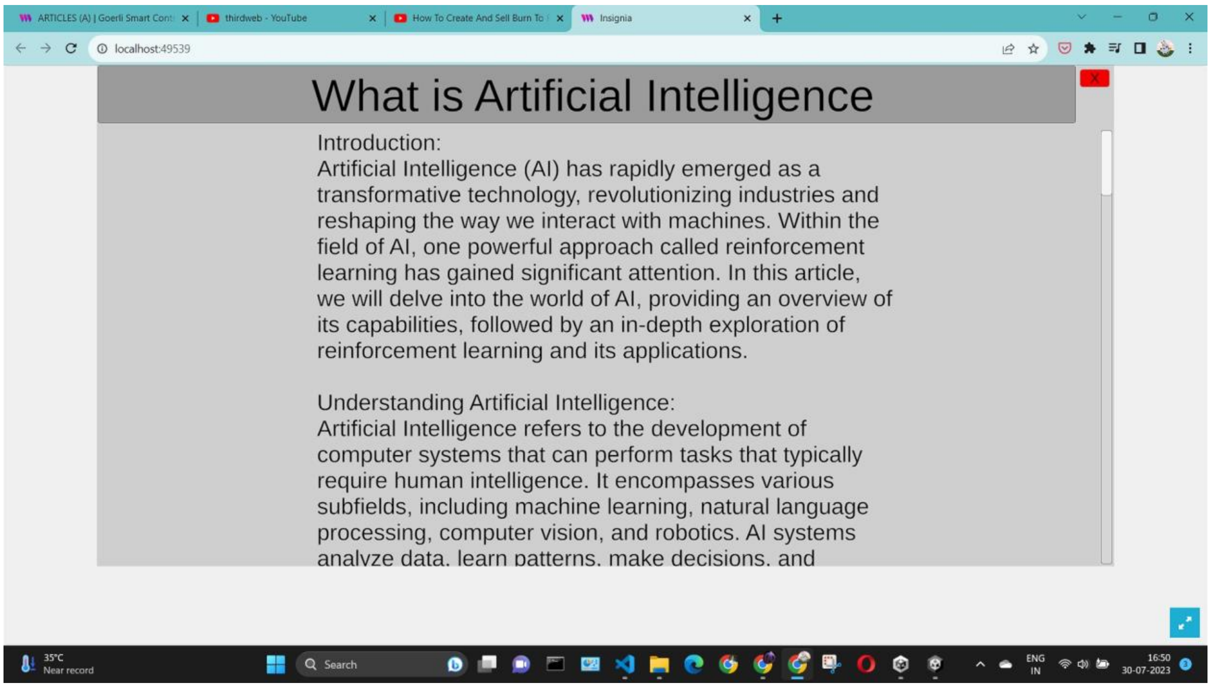


You own this article
click above to read

Confirmed transaction
Transaction 2 confirmed! View on Goerli Etherscan

Fig. Transaction has been confirmed, you can view it on etherscan

Click on the Read button to open the new article that you have purchased



What is Artificial Intelligence

Introduction:
Artificial Intelligence (AI) has rapidly emerged as a transformative technology, revolutionizing industries and reshaping the way we interact with machines. Within the field of AI, one powerful approach called reinforcement learning has gained significant attention. In this article, we will delve into the world of AI, providing an overview of its capabilities, followed by an in-depth exploration of reinforcement learning and its applications.

Understanding Artificial Intelligence:
Artificial Intelligence refers to the development of computer systems that can perform tasks that typically require human intelligence. It encompasses various subfields, including machine learning, natural language processing, computer vision, and robotics. AI systems analyze data, learn patterns, make decisions, and

Fig. The newly purchased article can now be viewed

Conclusion and Future Scope

In conclusion, this project has successfully demonstrated the effective integration of blockchain technology into content distribution platforms, showcasing its potential to enhance security, transparency, and user engagement. The meticulous research and design efforts undertaken throughout this endeavour have resulted in a robust system that leverages the Ethereum blockchain for seamless and secure transactions.

From our comprehensive research of blockchain and decentralised application systems, we have learned different approaches for application development. On top of that we have learned to design systems that integrate multi facet domains together.

Looking ahead, there are promising avenues for extending this blockchain-based content distribution technique to various other forms of digital content. The underlying principles demonstrated in this project, particularly within the Unity game environment, can be extrapolated to diverse types of media, including images, software, and more. The application of blockchain can usher in a new era of secure and verifiable content distribution across a multitude of industries.

Moreover, the integration of additional features and functionalities, such as decentralized identity management and advanced smart contract interactions, holds the potential to further enrich the capabilities of blockchain-based content distribution systems. Exploring interoperability with emerging blockchain standards and protocols could pave the way for a more interconnected and expansive ecosystem.

As technology continues to evolve, the seamless integration of blockchain into content distribution platforms stands as a testament to the adaptability and transformative potential of decentralized solutions. The groundwork laid in this project serves as a stepping stone for future innovations and developments in the dynamic landscape of blockchain-powered content distribution.

Throughout the course of this project, we have achieved significant milestones in the integration of blockchain technology into content distribution platforms. Key accomplishments include the successful implementation of a Unity game that utilizes the Ethereum blockchain for managing and distributing digital content. The incorporation of the Thirdweb SDK has facilitated secure and transparent transactions within the game environment, showcasing the practical application of blockchain in a real-world scenario.

Our achievements extend to the development of smart contracts for managing digital assets, specifically articles within the Unity game. The creation of a functional and user-friendly interface for interacting with the blockchain, coupled with the seamless integration of wallet functionalities, demonstrates a comprehensive understanding of blockchain concepts and their practical implementation.

The journey of this project has been rich with learnings across multiple domains. A deep understanding of Ethereum blockchain and the ERC-1155 token standard has been acquired, providing insights into the intricacies of decentralized applications. The utilization of the Thirdweb SDK has broadened our knowledge of blockchain integration in Unity environments and the associated methodologies for secure and efficient transaction management.

The project has enhanced our proficiency in smart contract development, allowing us to create robust and reliable digital asset management systems. The complexities of handling user wallets, authentication, and secure transactions in a decentralized environment have been thoroughly explored, contributing to a more comprehensive understanding of blockchain-based systems.

Furthermore, the project has honed our skills in system design, software architecture, and Unity game development. The hands-on experience gained in integrating diverse technologies into a cohesive and functional system has provided valuable insights into the challenges and opportunities associated with blockchain-powered content distribution platforms.