

TUGAS AKHIR - EC224801

SISTEM DATA SHARING BERBASIS *BLOCKCHAIN* UNTUK *AUDIO PLAYER* DI *METaverse*

Aaron Christopher Tanhar

NRP 0721 19 4000 0055

Dosen Pembimbing

Mochamad Hariadi, S.T., M.Sc., Ph.D.

NIP 19691209 199703 1 002

Dr. Surya Sumpeno, S.T., M.Sc.

NIP 19690613 199702 1 003

Program Studi Strata 1 (S1) Teknik Komputer

Departemen Teknik Komputer

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2023



TUGAS AKHIR - EC224801

SISTEM DATA SHARING BERBASIS *BLOCKCHAIN* UNTUK *AUDIO PLAYER* DI *METaverse*

Aaron Christopher Tanhar

NRP 0721 19 4000 0055

Dosen Pembimbing

Mochamad Hariadi, S.T., M.Sc., Ph.D.

NIP 19691209 199703 1 002

Dr. Surya Sumpeno, S.T., M.Sc.

NIP 19690613 199702 1 003

Program Studi Strata 1 (S1) Teknik Komputer

Departemen Teknik Komputer

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2023

[Halaman ini sengaja dikosongkan]



FINAL PROJECT - EC224801

***BLOCKCHAIN DATA SHARING SYSTEM FOR AUDIO
PLAYER IN METAVERSE***

Aaron Christopher Tanhar

NRP 0721 19 4000 0055

Advisor

Mochamad Hariadi, S.T., M.Sc., Ph.D.

NIP 19691209 199703 1 002

Dr. Surya Sumpeno, S.T., M.Sc.

NIP 19690613 199702 1 003

Undergraduate Study Program of Computer Engineering

Department of Computer Engineering

Faculty of Intelligent Electrical and Informatics Technology

Sepuluh Nopember Institute of Technology

Surabaya

2023

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KALKULASI ENERGI PADA ROKET LUAR ANGKASA BERBASIS *ANTI-GRAVITASI*

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Teknik pada Program Studi S-1
Teknik Komputer Departemen Teknik Komputer Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh: Aaron Christopher Tanhar
NRP. 0721 19 4000 0055

Disetujui oleh Tim Penguji Tugas Akhir:

Mochamad Hariadi, S.T., M.Sc., Ph.D.
NIP: 19691209 199703 1 002

(Pembimbing I)

.....

Dr. Surya Sumpeno, S.T., M.Sc.
NIP: 19690613 199702 1 003

(Pembimbing II)

.....

Dr. Galileo Galilei, S.T., M.Sc.
NIP: 18560710 194301 1 001

(Penguji I)

.....

Friedrich Nietzsche, S.T., M.Sc.
NIP: 18560710 194301 1 001

(Penguji II)

.....

Alan Turing, ST., MT.
NIP: 18560710 194301 1 001

(Penguji III)

.....

Mengetahui,
Kepala Departemen Teknik Komputer FTEIC - ITS

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T.
NIP. 19700313 199512 1 001

SURABAYA
Bulan, 2023

[Halaman ini sengaja dikosongkan]

APPROVAL SHEET

KALKULASI ENERGI PADA ROKET LUAR ANGKASA BERBASIS *ANTI-GRAVITASI*

FINAL PROJECT

Submitted to fulfill one of the requirements for obtaining Engineering degree at Undergraduate Study
Program of Computer Engineering Department of Computer Engineering Faculty of Intelligent
Electrical and Informatics Technology Sepuluh Nopember Institute of Technology

By: Aaron Christopher Tanhar
NRP. 0721 19 4000 0055

Approved by Final Project Examiner Team:

Mochamad Hariadi, S.T., M.Sc., Ph.D.
NIP: 19691209 199703 1 002

(Pembimbing I)

.....

Dr. Surya Sumpeno, S.T., M.Sc.
NIP: 19690613 199702 1 003

(Pembimbing II)

.....

Dr. Galileo Galilei, S.T., M.Sc.
NIP: 18560710 194301 1 001

(Penguji I)

.....

Friedrich Nietzsche, S.T., M.Sc.
NIP: 18560710 194301 1 001

(Penguji II)

.....

Alan Turing, ST., MT.
NIP: 18560710 194301 1 001

(Penguji III)

.....

Acknowledged,
Head of Computer Engineering Department ELECTICS - ITS

Dr. Supeno Mardi Susiki Nugroho, S.T., M.T.
NIP. 19700313 199512 1 001

SURABAYA
Month, 2023

[Halaman ini sengaja dikosongkan]

PERNYATAAN ORISINALITAS

Yang bertanda tangan dibawah ini:

Nama Mahasiswa / NRP : Aaron Christopher Tanhar / 0721 19 4000 0055
Departemen : Teknik Komputer
Dosen Pembimbing / NIP : Mochamad Hariadi, S.T., M.Sc., Ph.D. / 19691209 199703 1 002

Dengan ini menyatakan bahwa Tugas Akhir dengan judul ”SISTEM DATA SHARING BERBASIS *BLOCKCHAIN* UNTUK *AUDIO PLAYER* DI *METaverse*” adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, Mei 2021

Mengetahui
Dosen Pembimbing

Mahasiswa

Mochamad Hariadi, S.T., M.Sc., Ph.D.
NIP. 19691209 199703 1 002

Aaron Christopher Tanhar
NRP. 0721 19 4000 0055

[Halaman ini sengaja dikosongkan]

STATEMENT OF ORIGINALITY

The undersigned below:

Name of student / NRP : Aaron Christopher Tanhar / 0721 19 4000 0055
Department : Computer Engineering
Advisor / NIP : Mochamad Hariadi, S.T., M.Sc., Ph.D. / 19691209 199703 1 002

Hereby declared that the Final Project with the title of "*BLOCKCHAIN DATA SHARING SYSTEM FOR AUDIO PLAYER IN METAVERSE*" is the result of my own work, is original, and is written by following the rules of scientific writing.

If in future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with provisions that apply at Sepuluh Nopember Institute of Technology.

Surabaya, Mei 2021

Acknowledged
Advisor

Student

Mochamad Hariadi, S.T., M.Sc., Ph.D.
NIP. 19691209 199703 1 002

Aaron Christopher Tanhar
NRP. 0721 19 4000 0055

[Halaman ini sengaja dikosongkan]

ABSTRAK

Nama Mahasiswa : Aaron Christopher Tanhar
Judul Tugas Akhir : SISTEM DATA SHARING BERBASIS *BLOCKCHAIN* UNTUK *AUDIO PLAYER* DI *METaverse*
Pembimbing : 1. Mochamad Hariadi, S.T., M.Sc., Ph.D.
2. Dr. Surya Sumpeno, S.T., M.Sc.

Metaverse merupakan sebuah seperangkat ruang virtual, tempat seseorang dapat membuat dan menjelajah dengan pengguna internet lainnya yang tidak berada pada ruang fisik yang sama dengan orang tersebut. Pengaplikasiannya kerap kali menggunakan *blockchain* sebagai solusi *decentralized*. Metaverse sendiri tentunya memerlukan adanya output audio, yang diwujudkan oleh Unreal Engine 5 dengan fitur metasoundnya.

Penggabungan keduanya dapat dicapai dengan menggunakan *smart contract*. Maka di penelitian ini akan dibuat sistem yang dapat mewujudkan hal tersebut dengan bantuan *Ethereum Smart Contract*, NFT untuk menyimpan metadata, dan web3.storage IPFS untuk menyimpan file sumber audio untuk metasound.

Dengan adanya sistem terdesentralisasi tersebut juga diharapkan terwujudnya *interoperability* agar metaverse ini dapat digunakan dan berkomunikasi dengan platform blockchain lainnya.

Kata Kunci: *Metaverse, Audio, Blockchain, Ethereum, NFT*

[Halaman ini sengaja dikosongkan]

ABSTRACT

Name : Aaron Christopher Tanhar
Title : *BLOCKCHAIN DATA SHARING SYSTEM FOR AUDIO PLAYER IN META-VERSE*
Advisors : 1. Mochamad Hariadi, S.T., M.Sc., Ph.D.
2. Dr. Surya Sumpeno, S.T., M.Sc.

The Metaverse is a set of virtual spaces, where one can create and browse with other internet users who are not in the same physical space with that person. The application often uses blockchain as a solution decentralized. The Metaverse itself certainly requires an audio output, which is realized by Unreal Engine 5 with its metasound feature.

Merging the two can be achieved by using smart contracts. Then in This research will create a system that can make this happen with the help of Ethereum Smart Contract, NFT to store metadata, and IPFS web3.storage to store audio source files for metasound.

With the existence of a decentralized system, it is also hoped that interoperability will be realized. ability to make this metaverse usable and communicate with other blockchain platforms.

Keywords: Rocket, Anti-gravity, Energy, Space.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur kehadiran Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque.

Penelitian ini disusun dalam rangka Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Keluarga, Ibu, Bapak dan Saudara tercinta yang telah Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero.
2. Bapak Nikola Tesla, S.T., M.T., selaku Quisque ullamcorper placerat ipsum. Cras nibh.
3. Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl.

Akhir kata, semoga Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus.

Surabaya, Mei 2023

Aaron Christopher Tanhar

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

ABSTRAK	i
ABSTRACT	iii
KATA PENGANTAR	v
DAFTAR ISI	vii
DAFTAR GAMBAR	xi
DAFTAR TABEL	xiii
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Permasalahan	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
2 TINJAUAN PUSTAKA	3
2.1 Penelitian Terdahulu	3
2.1.1 Sistem Transaksi Antar Player Pada Game Multiplayer Wisata Bromo Menggunakan Blockchain	3
2.2 <i>Blockchain</i>	3
2.3 <i>Ethereum</i>	4
2.4 <i>Smart Contract</i>	5
2.5 <i>Ethereum Transaction</i>	5
2.6 <i>InterPlanetary File System</i>	6
2.7 <i>Solidity</i>	6
2.8 <i>Crypto Wallet</i>	7
2.9 <i>Unreal Engine 5</i>	8
2.10 <i>Ethereum Virtual Machine</i>	9
2.11 <i>ERC-721</i>	9
2.12 Minting Token	10

2.13 Metaverse	11
2.14 Interoperabilitas	11
2.15 <i>Blueprint</i>	12
3 METODOLOGI	15
3.1 Software	15
3.1.1 Remix IDE	15
3.1.2 Ganache	16
3.1.3 Unreal Engine 5	18
3.2 Desain Sistem	18
3.3 Create Wallet Account	19
3.4 Create <i>Smart Contract</i>	21
3.4.1 Kontrak Import dan Inheritance	23
3.4.2 Konstruktor	23
3.4.3 Fungsi Mint	23
3.4.4 Fungsi getAllTokens	23
3.4.5 Fungsi getAllTokenURIs	23
3.4.6 Fungsi _beforeTokenTransfer dan _burn	24
3.4.7 Fungsi supportsInterface dan tokenURI	24
3.5 Prepare Metadata with IPFS	24
3.5.1 Web3 Storage	26
3.6 Mint NFT	27
3.7 Create Unreal Engine 5 Project	28
3.7.1 Runtime Audio Importer	29
3.7.2 Emergence	29
3.8 Integrate Smart Contract with Unreal Engine 5	30
3.8.1 Read Method	33
4 HASIL DAN PEMBAHASAN	35
4.1 Skenario Pengujian	35
4.1.1 Hasil Pembuatan Wallet Account	35
4.1.2 Hasil dan Pengujian Method Smart Contract	35
4.1.3 Hasil Pengujian Estimasi Gas	39
4.1.4 Hasil Pembuatan Metadata dengan IPFS	41
4.1.5 Hasil Mint NFT	43

4.1.6	Hasil Pembuatan Project Pada Unreal Engine 5	45
4.1.7	Hasil Integrasi Kontrak dengan Unreal Engine 5	46
4.2	Evaluasi Pengujian	50
5	PENUTUP	51
5.1	Kesimpulan	51
5.2	Saran	51
	DAFTAR PUSTAKA	53
	BIOGRAFI PENULIS	55

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

3.1	Tampilan <i>Remix IDE</i>	16
3.2	Tampilan <i>Ganache</i>	18
3.3	Tampilan <i>Unreal Engine 5</i>	18
3.4	Desain Sistem	19
3.5	Tampilan Metamask	20
3.6	Tampilan Akun Goerli di Metamask	21
3.7	Antarmuka Deployment di Remix IDE	24
3.8	Standar NFT OpenSea	25
3.9	Tampilan web3.storage	26
3.10	Tampilan di menu file web3.storage	27
3.11	Tampilan Remix IDE untuk deployed contracts	27
3.12	Tampilan Asset yang disediakan Emergence	30
3.13	Tampilan Asset Kontrak Emergence	31
3.14	Tampilan Deployed Contracts dari Emergence	32
3.15	Tampilan Blueprint Read Method	33
4.1	Transaksi blockchain di Ganache	39
4.2	Tampilan IPFS nft-1.json yang sudah diupload	42
4.3	Tampilan OpenSea yang berhasil membaca metadata	43
4.4	Etherscan dari hasil transaction mint NFT	44
4.5	Tampilan project unreal engine 5	45
4.6	Tampilan blueprint final	46
4.7	Tampilan contract ABI di Unreal Engine 5	47
4.8	Tampilan blueprint untuk pemanggilan method <code>tokenURI</code>	48
4.9	Tampilan blueprint untuk load metadata	49

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

4.1	Hasil Pengujian Method <code>mint</code>	35
4.2	Hasil Pengujian Method <code>getAllTokens</code>	36
4.3	Hasil Pengujian Method <code>getAllTokensURI</code>	36
4.4	Hasil Pengujian Method <code>tokenURI</code>	37
4.5	Hasil Pengujian Deploy Smart Contract	40
4.6	Hasil Pengujian Deploy Smart Contract di <i>Ganache</i>	40
4.7	Hasil Pengujian Gas Mint Smart Contract	41
4.8	Hasil Pengujian Gas Mint di <i>Ganache</i>	41
4.9	Hasil Pengujian Deploy ke Goerli berdasarkan Ether	43
4.10	Hasil Pengujian ABI Smart Contract di UE5	47
4.11	Hasil Pengujian Pemanggilan method menggunakan Blueprint	47
4.12	Hasil Pengujian Load Metadata Audio di UE5	49
4.13	Hasil Pengujian Pemutaran Audio di Unreal Engine 5	49

[Halaman ini sengaja dikosongkan]

BAB I

PENDAHULUAN

1.1 Latar Belakang

Munculnya teknologi *virtual reality* (VR) dan pengembangan platform metaverse telah mengarah pada terciptanya pengalaman online yang imersif di mana pengguna dapat berinteraksi satu sama lain dan terlibat dalam berbagai aktivitas, apalagi setelah Mark Zuckerberg membeli *oculus* (Naz et al., 2019). Salah satu aktivitas tersebut adalah memainkan alat musik sebagai karakter virtual, yang dapat dicapai melalui penggunaan teknologi *motion capture* dan simulasi instrumen virtual.

Di era saat ini, blockchain sangat penting bagi industri musik dan metaverse karena menyediakan solusi untuk beberapa masalah yang dihadapi kedua industri tersebut. Blockchain dapat membantu industri musik menangani masalah seperti keaslian, transparansi, dan pengelolaan royalti. Informasi kepemilikan dan hak cipta musik dapat dilacak secara terbuka dan tidak dapat diubah dengan teknologi blockchain. Ini menjaga kekayaan intelektual para pencipta dan memastikan bahwa royalti yang seharusnya mereka terima didistribusikan dengan adil. Selain itu, blockchain memungkinkan pencipta musik untuk menjual musik mereka secara langsung kepada penggemar mereka dengan menggunakan NFT (Non-Fungible Token), yang menawarkan pengalaman interaktif dan keuntungan baru dalam hal monetisasi.

Metaverse sangat penting untuk menciptakan pengalaman digital yang lebih kaya, inklusif, dan terhubung. Metaverse membawa potensi baru dalam bidang hiburan, interaksi sosial, ekonomi digital, pendidikan, dan banyak lagi. Dengan perkembangan teknologi dan adopsi metaverse yang semakin luas, kita dapat mengantisipasi perubahan besar dalam cara kita berinteraksi dan mengalami dunia digital.

Kemudian untuk layanan-layanan pemutar musik juga memiliki beberapa implementasi sistem *centralized*. Contohnya spotify, soundcloud, dan lain-lain. Soundcloud sendiri memiliki kelemahan pada sisi kepemilikan atau *copyright*. Sistem *decentralized* merupakan salah satu solusi dari hal ini, dikarenakan sifatnya yang *immutable*. Salah satu contohnya adalah *Audius*. Audius sendiri menggunakan sistem *decentralized* dan source musiknya disimpan di IPFS. Akan tetapi untuk menggunakan Audius harus menggunakan *embedded iframe* yang artinya hanya support untuk penggunaan di *Web Application* saja.

Adapula antarmuka dari audio itu sendiri adalah metaverse yang salah satu contohnya adalah Unreal Engine 5. Salah satu fitur dari Unreal Engine 5 yaitu *Metasound* yang dapat melakukan pengolahan audio dengan mudah bagi para pengembang. Unreal Engine 5 juga memiliki fitur yakni *API blockchain* sehingga dapat mengeksekusi *smart contract*. Fitur ini dinamakan Web3.UE. Web3.Unreal adalah plugin *open source* yang dibuat untuk pengembang game dan komunitas untuk membantu mereka yang bekerja dengan Unreal Engine untuk mengintegrasikan fungsionalitas *blockchain* ke dalam game mereka. Kedua hal ini dapat dikombinasikan dan digunakan untuk pengembangan metaverse.

Metaverse telah digambarkan sebagai iterasi baru dari internet yang menggunakan headset

VR, teknologi *blockchain*, dan avatar dalam integrasi baru dunia fisik dan *virtual*. (Dwivedi et al., 2022). Dalam proposal penelitian ini, diusulkan untuk merancang dan mengimplementasikan sistem berbagi data berbasis *blockchain* untuk *musical player* di *metaverse* dengan *NFT*. Sistem ini akan menggunakan *smart contract* dan solusi penyimpanan terdesentralisasi untuk memungkinkan pengguna berbagi dan mengakses data musik dari *metaverse* dengan cara yang aman dan transparan.

1.2 Permasalahan

Di dunia *virtual metaverse*, kebutuhan akan sistem yang aman dan efisien untuk berbagi data terkait musik semakin meningkat. Saat ini, data ini sering disimpan dalam *database* terpusat yang rentan terhadap pelanggaran keamanan dan penyensoran, dan dapat menyulitkan pengguna untuk mengakses dan mengontrol data mereka sendiri. Ketidakmampuan berinteroperabilitas dalam ekosistem *blockchain* dapat menyebabkan fragmentasi data dan aset, keterbatasan fungsionalitas dan kasus penggunaan, penurunan skalabilitas, likuiditas terbatas, fragmentasi pasar, serta kurangnya kolaborasi dan standardisasi. Interoperabilitas sangat penting untuk mewujudkan potensi penuh teknologi *blockchain* dan memungkinkan pertukaran nilai yang lancar di berbagai jaringan. Karena belum ada sistem yang ada untuk melakukan hal serupa layaknya *audius* dan *spotify* di *metaverse*, maka penelitian ini juga mencetuskan hal yang belum ada pada saat proposal ini diajukan.

1.3 Tujuan

Penelitian ini memiliki tujuan untuk membuat sistem yang dapat melakukan sharing data untuk audio player yang ada di *Metaverse* menggunakan platform *blockchain*, agar pengguna *metaverse* dapat menggunakan platform music yang diintegrasikan dengan *audio player* di *Unreal Engine 5*. Penelitian ini juga bertujuan mengembangkan *metaverse* yang bersinergi dengan web3.0 dan menerapkan *interoperability*.

1.4 Batasan Masalah

Untuk memfokuskan permasalahan yang diangkat maka dilakukan pembatasan masalah. Batasan - batasan masalah tersebut diantaranya:

1. Jenis *Blockchain* yang digunakan adalah *Ethereum*.
2. Jaringan *testnet blockchain* yang digunakan adalah *Goerli* dan *Sepolia*
3. Metadata audio yang digunakan berformat *JSON*.

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

2.1.1 Sistem Transaksi Antar Player Pada Game Multiplayer Wisata Bromo Menggunakan Blockchain

Penelitian oleh Reza Putra Pradana berjudul Sistem Transaksi Antar Player Pada Game Multiplayer Wisata Bromo merupakan implementasi teknologi *Blockchain* untuk sistem transaksi antar player dalam bentuk game dengan mengujikan latensi serta performa waktu eksekusi transaksi hingga biaya gas yang dibutuhkan untuk pengembangan game tersebut. Dalam penelitian ini *Blockchain* yang digunakan salah satunya adalah Ethereum untuk pengujiannya. Adanya biaya gas kaitannya erat dengan proses *development* dari game itu sendiri karena melalui biaya gas dapat diketahui berapa lama proses transaksi dilakukan sehingga analisis lebih lanjut diperlukan untuk mendapat formula yang pas terkait biaya gas dengan kebutuhan *development* dari game tersebut.

2.2 Blockchain

Blockchain adalah sebuah teknologi yang mendasari sistem terdesentralisasi yang memungkinkan pertukaran informasi dan aset digital dengan aman, transparan, dan terverifikasi. Dalam konsepnya, *blockchain* adalah jenis database terdistribusi yang memungkinkan para peserta jaringan untuk mencatat, menyimpan, dan memverifikasi transaksi tanpa memerlukan otoritas pusat. Teknologi ini telah menarik minat dari berbagai sektor, termasuk keuangan, logistik, kesehatan, energi, dan banyak lagi, karena memiliki potensi untuk mengatasi masalah keamanan, transparansi, dan efisiensi dalam sistem tradisional. (Nakamoto, 2008)

Salah satu karakteristik utama dari *blockchain* adalah sifat terdistribusinya. Artinya, database *blockchain* dikelola oleh jaringan peer-to-peer yang terdiri dari berbagai peserta yang saling berinteraksi tanpa otoritas pusat yang mengendalikan seluruh jaringan. Setiap peserta dalam jaringan memiliki salinan lengkap dari database yang diperbarui secara real-time. Hal ini memberikan keamanan dan ketahanan yang tinggi karena tidak ada satu titik kegagalan tunggal yang dapat menyebabkan kegagalan seluruh sistem.

Struktur dasar dari *blockchain* adalah serangkaian blok yang saling terhubung melalui kriptografi. Setiap blok berisi sejumlah transaksi atau data yang telah diverifikasi dan ditandatangani secara digital. Setiap kali ada transaksi baru atau perubahan data, blok baru akan ditambahkan ke rantai, menciptakan jejak waktu dan konsistensi kronologis. Kunci keamanan dalam *blockchain* adalah hash, yaitu fungsi kriptografi yang menghasilkan nilai unik untuk setiap blok, serta tautan ke blok sebelumnya.

Mekanisme konsensus adalah bagian penting dalam *blockchain* untuk mencapai kesepakatan tentang keabsahan transaksi dan menjaga integritas jaringan. Beberapa mekanisme konsensus yang umum digunakan termasuk Proof of Work (PoW), Proof of Stake (PoS), dan Delegated Proof of Stake (DPoS). Mekanisme ini memastikan bahwa para peserta dalam jaringan

mencapai kesepakatan sebelum transaksi atau perubahan data dapat dianggap sah. (Swan, 2015)

Keuntungan utama dari *blockchain* meliputi transparansi, keamanan, dan efisiensi. Transparansi terjadi karena semua transaksi yang terjadi di dalam *blockchain* dapat dilihat oleh semua peserta jaringan, mengurangi risiko manipulasi dan kecurangan. Keamanan terjadi karena setiap transaksi harus diverifikasi oleh jaringan secara kolektif, dan setiap blok dalam rantai memiliki tautan dengan blok sebelumnya menggunakan kriptografi. Efisiensi terjadi karena tidak ada perantara atau otoritas pusat yang memperlambat proses transaksi, sehingga memungkinkan pertukaran aset dan informasi secara langsung antara para peserta.

Meskipun *blockchain* pertama kali dikenal melalui Bitcoin, cryptocurrency pertama yang dibangun di atas teknologi ini, namun sekarang *blockchain* digunakan dalam berbagai konteks dan aplikasi. Selain transaksi keuangan, *blockchain* juga digunakan dalam pengelolaan rantai pasok, pemilu elektronik, manajemen data medis, sertifikasi aset digital, dan banyak lagi. Dengan menerapkan prinsip-prinsip desentralisasi, transparansi, dan keamanan, *blockchain* menghadirkan potensi untuk mengubah cara kita mempertukarkan nilai dan membangun sistem yang lebih efisien dan aman. (Tapscott & Tapscott, 2016)

2.3 *Ethereum*

Ethereum adalah sebuah platform *blockchain* terdesentralisasi yang memungkinkan pengembangan dan eksekusi aplikasi terdistribusi yang disebut smart contract. Dikembangkan oleh Vitalik Buterin pada tahun 2013, *Ethereum* memperluas konsep dasar *blockchain* dengan memperkenalkan kemampuan untuk menjalankan kode pemrograman yang kompleks di dalam *blockchain*. Platform ini menggunakan mata uang digital yang disebut *Ether (ETH)* untuk memfasilitasi transaksi dan menjalankan *smart contract*. (Wood, 2014)

Smart contract di *Ethereum* adalah program komputer yang dieksekusi secara otomatis ketika kondisi yang telah ditentukan terpenuhi. Mereka memungkinkan pihak-pihak yang terlibat dalam sebuah kesepakatan untuk menjalankan transaksi atau interaksi bisnis tanpa memerlukan perantara. Dengan demikian, *Ethereum* memungkinkan pelaksanaan kontrak yang transparan, terotomatisasi, dan tidak dapat diubah. (Antonopoulos & Wood, 2018)

Salah satu fitur utama *Ethereum* adalah Turing-completeness, yang berarti platform ini dapat menjalankan hampir semua jenis aplikasi terdistribusi. Hal ini memungkinkan pengembang untuk membangun aplikasi yang lebih kompleks dan beragam, termasuk permainan, pasar digital, sistem keuangan terdesentralisasi, identitas digital, dan banyak lagi. (Burniske & Davis, 2019)

Ether (ETH) adalah mata uang digital yang digunakan di dalam jaringan *Ethereum*. *Ether* tidak hanya berfungsi sebagai alat pembayaran dalam transaksi, tetapi juga digunakan untuk menjalankan *smart contract*. Setiap kali *smart contract* dieksekusi, pemilik *smart contract* harus membayar biaya dalam bentuk *Ether* yang dikenal sebagai gas. Biaya gas ini membantu mencegah serangan spam dan memastikan bahwa pengguna platform menggunakan sumber daya jaringan dengan bijak. (Diedrich, 2017)

Selain itu, *Ethereum* juga memiliki kemampuan untuk menciptakan token *ERC-20*, yang memungkinkan pengguna untuk membuat dan mengelola token mereka sendiri di dalam ekosistem *Ethereum*. Token ini digunakan untuk berbagai tujuan, termasuk dalam ICO (Initial Coin Offering) sebagai sarana penggalangan dana untuk proyek baru, atau sebagai aset digital yang dapat diperdagangkan di bursa kripto.

Ethereum juga mendukung konsep DAO (*Decentralized Autonomous Organization*), yang memungkinkan komunitas untuk mengorganisir diri mereka sendiri dan mengambil keputusan berdasarkan prinsip demokratis dengan menggunakan *smart contract*. DAO memungkinkan anggota komunitas untuk memiliki suara dalam pengelolaan dan penggunaan sumber daya kolektif tanpa keberadaan otoritas pusat. (Ozer, 2018)

2.4 Smart Contract

Smart contract adalah program komputer yang berjalan secara otomatis ketika kondisi yang telah ditentukan terpenuhi. Mereka beroperasi di dalam platform *blockchain*, seperti *Ethereum*, dan berfungsi untuk mengeksekusi transaksi atau interaksi bisnis tanpa memerlukan perantara. (Antonopoulos & Wood, 2018)

Smart contract menggunakan bahasa pemrograman yang telah ditentukan, seperti *Solidity* pada *Ethereum*, untuk menggambarkan logika bisnis yang ingin dijalankan. Mereka menyimpan perjanjian atau kesepakatan antara pihak-pihak yang terlibat dan mengatur aliran dana atau aset digital. Setiap kali kondisi yang telah ditentukan terpenuhi, *smart contract* secara otomatis menginisiasi tindakan yang telah ditetapkan.

Keuntungan utama dari *smart contract* adalah keandalan dan keamanan yang tinggi. Karena mereka dieksekusi di dalam platform *blockchain*, setiap transaksi dan tindakan yang dilakukan oleh *smart contract* tercatat secara permanen dan tidak dapat diubah. Ini memastikan transparansi dan meminimalkan risiko manipulasi atau kecurangan.

Selain itu, *smart contract* juga menghilangkan kebutuhan akan perantara atau pihak ketiga dalam transaksi bisnis. Dengan mengotomatiskan proses, *smart contract* mengurangi biaya dan waktu yang terlibat dalam penyelesaian transaksi. Mereka juga meningkatkan kepercayaan antara pihak-pihak yang terlibat karena semua detail dan ketentuan kesepakatan tercatat dengan jelas di dalam kode program.

Namun, penting untuk memperhatikan bahwa *smart contract* tidak sempurna dan masih dapat rentan terhadap kerentanan atau kesalahan dalam kode program. Karena itu, pengembang dan pengguna *smart contract* perlu melakukan pengujian yang cermat dan perhatian terhadap aspek keamanan. (Burniske & Davis, 2019)

2.5 Ethereum Transaction

Transaksi adalah paket data yang ditandatangani untuk mengirim Ether dari satu akun ke akun lain atau ke kontrak, memanggil metode kontrak, atau menyebarkan kontrak baru. Sebuah transaksi adalah ditandatangani menggunakan ECDSA (Elliptic Curve Digital Signature Algorithm), yang merupakan digital algoritma tanda tangan berdasarkan ECC. Sebuah transaksi berisi penerima pesan, sebuah tanda tangan yang mengidentifikasi pengirim dan membuktikan niat mereka, jumlah Ether yang akan ditransfer, jumlah maksimum langkah komputasi yang diizinkan untuk dilakukan oleh eksekusi transaksi (disebut batas gas), dan biaya yang bersedia dibayar oleh pengirim transaksi untuk masing-masing langkah komputasi (disebut harga gas). Jika niat transaksi adalah untuk memanggil metode kontrak, itu juga berisi data input, atau jika tujuannya adalah untuk menyebarkan kontrak, maka itu bisa kode inisialisasi. Produk gas yang digunakan dan harga gas disebut transaksi biaya. Untuk mengirim Ether atau menjalankan metode kontrak, perlu menyalurkan transaksi ke jaringan. Pengirim perlu menandatangani transaksi dengan kunci pribadinya (Prusty, 2017)

2.6 InterPlanetary File System

IPFS (InterPlanetary File System) adalah sebuah protokol dan sistem distribusi file yang dirancang untuk menyimpan dan mengakses konten secara terdesentralisasi. Tujuan utama dari IPFS adalah untuk mengatasi beberapa keterbatasan yang ada dalam sistem penyimpanan file tradisional, seperti ketergantungan pada server pusat, keterbatasan bandwidth, dan risiko kehilangan data.

Dalam *IPFS*, setiap file dan blok data diberikan alamat yang unik berdasarkan kontennya. Alih-alih menggunakan alamat berbasis lokasi, seperti *URL* dalam sistem web tradisional, *IPFS* menggunakan alamat berbasis konten yang disebut *CID (Content Identifier)*. Hal ini memungkinkan pengguna untuk mengakses konten secara langsung dari jaringan IPFS tanpa bergantung pada lokasi fisik file tersebut disimpan. (Benet, 2014)

IPFS menggunakan konsep distribusi *peer-to-peer*, di mana setiap peserta dalam jaringan *IPFS* berfungsi sebagai titik penyimpanan dan penyebaran konten. Setiap kali pengguna meminta file, *IPFS* mencari konten tersebut secara otomatis dari *node* yang paling dekat dan mempercepat proses pengunduhan dengan memanfaatkan koneksi *peer-to-peer*.

Keuntungan dari *IPFS* termasuk keandalan dan ketahanan terhadap kehilangan data. Karena setiap file mendapatkan alamat berdasarkan kontennya, jika ada duplikat file yang identik di jaringan, maka file tersebut hanya akan disimpan satu kali. Selain itu, *IPFS* juga mendukung mekanisme verifikasi integritas data menggunakan teknologi hash. (batiz2018mastering)

2.7 Solidity

Solidity adalah bahasa pemrograman yang digunakan untuk menulis *smart contract* di platform *Ethereum*. *Solidity* dirancang untuk menggabungkan fitur-fitur dari bahasa pemrograman seperti *JavaScript* dan *C++* serta memperluas kemampuan mereka untuk memenuhi kebutuhan pengembangan aplikasi terdesentralisasi.

Solidity mendukung pengembangan aplikasi yang berjalan di atas platform *Ethereum* dengan menggunakan *smart contract*. Dalam *Solidity*, pengembang dapat mendefinisikan struktur data, fungsi, dan logika bisnis yang akan dieksekusi oleh *smart contract*. Pengembang juga dapat menetapkan aturan untuk penggunaan aset digital, seperti token ERC-20 atau ERC-721, serta mengatur interaksi antara *smart contract* dan pengguna atau *smart contract* lainnya. (“Solidity Documentation”, n.d.)

Kelebihan *Solidity* adalah kemampuannya untuk mengelola aset digital dan melaksanakan perjanjian bisnis secara aman dan andal. *Solidity* memastikan bahwa *smart contract* berjalan sesuai dengan yang diharapkan dan mencegah terjadinya manipulasi atau kegagalan sistem. Selain itu, *Solidity* juga menyediakan fasilitas pengujian dan pemecahan masalah yang memungkinkan pengembang untuk mengidentifikasi dan memperbaiki kesalahan dalam kode *smart contract*. *Solidity* memiliki sintaks yang mirip dengan bahasa pemrograman lainnya seperti *JavaScript*, sehingga pengembang dengan pengetahuan pemrograman umum dapat dengan mudah mempelajari dan menggunakan bahasa ini.

Solidity mendukung berbagai fitur yang penting dalam pengembangan *smart contract*. Beberapa fitur tersebut antara lain:

1. Tipe Data: *Solidity* mendukung berbagai jenis tipe data seperti bilangan bulat, string, boolean, array, dan struktur data yang memungkinkan pengembang untuk mengelola dan

memanipulasi data dengan lebih efisien.

2. *Contract*: Solidity memungkinkan pengembang untuk mendefinisikan *smart contract* yang berfungsi sebagai entitas utama dalam aplikasi terdesentralisasi. *Smart contract* ini dapat mengatur logika bisnis, menyimpan data, mengelola aset digital, dan berinteraksi dengan *smart contract* lainnya.
3. Fungsi dan Modifikator: *Solidity* memungkinkan pengembang untuk mendefinisikan fungsi dan modifikator yang memungkinkan pengontrolan akses, verifikasi parameter, dan menambahkan logika tambahan dalam eksekusi *smart contract*.

Solidity adalah bahasa pemrograman yang matang dan terus berkembang dengan komunitas pengembang yang aktif. Dukungan yang kuat dari komunitas Ethereum dan dokumentasi yang kaya menjadikan Solidity sebagai pilihan yang populer dalam pengembangan aplikasi terdesentralisasi.

2.8 *Crypto Wallet*

Crypto wallet, atau dompet kripto, adalah sebuah perangkat lunak atau layanan yang digunakan untuk menyimpan, mengelola, dan berinteraksi dengan aset kripto. Dalam dunia kriptografi, aset kripto seperti Bitcoin, Ethereum, atau altcoin lainnya tidak benar-benar disimpan dalam bentuk fisik seperti uang tunai konvensional. Sebagai gantinya, kepemilikan aset kripto ini dicatat dalam bentuk transaksi digital yang tercatat di dalam blockchain.

Crypto wallet berfungsi sebagai alat untuk mengakses, mengelola, dan mengendalikan aset kripto pengguna. Dengan menggunakan crypto wallet, pengguna dapat melakukan transaksi seperti mengirim dan menerima aset kripto, melacak saldo aset, dan memonitor riwayat transaksi. Crypto wallet juga dapat memberikan fitur keamanan seperti enkripsi data dan tanda tangan digital untuk memastikan keamanan dan keotentikan transaksi. (Noauthor, 2017)

Selain itu, crypto wallet juga menyediakan alamat publik yang dapat digunakan oleh orang lain untuk mengirimkan aset kripto ke wallet tersebut. Alamat publik ini bersifat terbuka dan dapat dibagikan kepada siapa pun tanpa mengungkapkan informasi rahasia seperti kunci pribadi.

Seiring dengan berkembangnya industri kripto, crypto wallet juga telah mengalami evolusi. Saat ini, ada banyak jenis wallet yang muncul dengan fitur-fitur yang berbeda. Misalnya, beberapa wallet memiliki fitur pertukaran terintegrasi yang memungkinkan pengguna menukar satu jenis aset kripto dengan yang lain. Ada juga wallet yang mendukung staking, yaitu mengunci aset kripto untuk mendapatkan imbalan.

Keamanan crypto wallet sangat penting, karena jika kunci pribadi dicuri atau hilang, akses ke aset kripto tidak dapat dipulihkan. Oleh karena itu, pengguna crypto wallet perlu mengambil langkah-langkah keamanan yang tepat, seperti menggunakan wallet resmi yang terpercaya, membuat cadangan kunci pribadi (backup), dan mengaktifkan fitur keamanan tambahan seperti otentikasi dua faktor (2FA). (Antonopoulos, 2014)

Pemilihan jenis crypto wallet tergantung pada preferensi pengguna, kebutuhan keamanan, dan kenyamanan akses. Penting untuk memilih wallet yang andal, aman, dan kompatibel dengan aset kripto yang ingin Anda simpan.

2.9 *Unreal Engine 5*

Unreal Engine 5 adalah sebuah mesin permainan (game engine) yang dikembangkan oleh Epic Games. Ini adalah generasi terbaru dari Unreal Engine, yang dirancang untuk memberikan pengalaman pengembangan dan visual yang luar biasa. Unreal Engine 5 memiliki sejumlah fitur baru yang memungkinkan pengembang untuk menciptakan pengalaman permainan yang lebih realistis dan imersif.

Salah satu fitur unggulan dari Unreal Engine 5 adalah "Nanite", teknologi rendering yang memungkinkan penggunaan geometri film-quality secara real-time. Nanite memungkinkan pengembang untuk menggunakan model 3D yang sangat detail dengan jutaan poligon tanpa perlu khawatir tentang kinerja permainan. Ini memberikan kebebasan yang lebih besar dalam menciptakan lingkungan yang kaya dan mendetail.

Selain itu, Unreal Engine 5 juga menghadirkan "Lumen", sebuah sistem pencahayaan global yang dinamis. Lumen memungkinkan pencahayaan yang realistis dan interaksi cahaya yang kompleks dalam permainan. Dengan Lumen, pengembang dapat menciptakan efek pencahayaan yang realistis seperti bayangan dinamis, refleksi, dan pencahayaan global yang real-time. (Games, 2022b)

Unreal Engine 5 juga menyediakan alat pengembangan yang lebih mudah digunakan, termasuk alat visual scripting yang kuat yang disebut "Blueprints", yang memungkinkan pengembang untuk membuat logika permainan tanpa harus menulis kode. Ini membuat pengembangan permainan menjadi lebih cepat dan lebih mudah, terutama bagi pengembang yang tidak memiliki latar belakang pemrograman yang mendalam.

Dengan kombinasi fitur-fitur ini, Unreal Engine 5 memberikan platform yang kuat bagi pengembang permainan untuk menciptakan pengalaman permainan yang mengesankan dengan visual yang spektakuler dan gameplay yang mendalam. (Games, 2022a)

Unreal Engine 5 adalah salah satu mesin permainan yang berperan penting dalam pembangunan dan pengembangan metaverse. Metaverse adalah sebuah konsep yang merujuk pada dunia virtual yang terhubung secara global, di mana pengguna dapat berinteraksi dengan lingkungan digital yang luas, bertemu dengan orang lain, dan menjalankan berbagai aktivitas secara virtual.

Unreal Engine 5 menyediakan sejumlah fitur yang mendukung pembangunan metaverse. Salah satu fitur utama adalah "World Partition", yang memungkinkan pengembangan lingkungan digital yang sangat luas tanpa mengorbankan kinerja permainan. Dengan fitur ini, dunia virtual yang luas dapat dibagi menjadi beberapa bagian yang dapat dimuat secara terpisah, sehingga hanya bagian yang relevan yang dimuat ke memori saat diperlukan. Hal ini memungkinkan lingkungan metaverse yang lebih besar dan lebih padat dengan detail tanpa mengorbankan kinerja permainan.

Selain itu, Unreal Engine 5 juga mendukung teknologi streaming yang canggih, yang memungkinkan pengguna untuk mengalami lingkungan metaverse yang terhubung secara mulus tanpa adanya waktu unduh yang lama. Dengan menggunakan teknologi streaming ini, konten dapat dimuat secara dinamis saat pengguna menjelajahi metaverse, memungkinkan akses instan ke lingkungan baru dan interaksi yang lancar.

Fitur-fitur visual Unreal Engine 5 juga sangat relevan dalam menciptakan pengalaman metaverse yang menarik. Misalnya, Nanite yang telah disebutkan sebelumnya memungkinkan penggunaan model 3D yang sangat detail, memperkaya lingkungan metaverse dengan objek-objek yang realistis. Selain itu, Lumen juga membantu menciptakan efek pencahayaan yang

realistis, memberikan atmosfer yang mendalam dan nuansa yang kaya pada lingkungan metaverse.

Dengan kombinasi dari semua fitur ini, Unreal Engine 5 memberikan alat yang kuat bagi pengembang untuk membangun metaverse yang menarik, interaktif, dan memikat. Dalam metaverse, pengguna dapat menjelajahi dunia virtual yang luas, berinteraksi dengan objek dan orang lain, serta melakukan berbagai aktivitas, seperti bermain permainan, menghadiri acara, dan bahkan melakukan transaksi ekonomi digital.

2.10 *Ethereum Virtual Machine*

Proses dibalik berjalannya *Smart Contract* pada *Ethereum* diatur dalam suatu mesin komputasi yang diberi nama *Ethereum Virtual Machine*. Semua penyebaran serta eksekusi dari *Smart Contract* berlangsung dengan melewati mesin virtual ini. Transaksi transfer nilai sederhana dari satu EOA ke EOA lainnya tidak perlu melibatkannya, tetapi yang lainnya akan melibatkan pembaruan status yang dihitung oleh *EVM*. Pada tingkat tinggi, *EVM* yang berjalan di *Blockchain Ethereum* dapat dianggap sebagai komputer terdesentralisasi global yang berisi jutaan objek yang dapat dieksekusi, masing-masing dengan penyimpanan data permanennya sendiri.

2.11 *ERC-721*

ERC-721, yang juga dikenal sebagai Non-Fungible Token (NFT) Standard, adalah sebuah standar kontrak pintar (smart contract) yang dikembangkan di jaringan Ethereum. Standar ini memungkinkan pembuatan dan pengelolaan token non-fungible, yang merupakan jenis token digital yang unik dan tidak dapat saling dipertukarkan satu sama lain. ERC-721 memainkan peran penting dalam memfasilitasi ekonomi digital dan pasar NFT yang sedang berkembang.

Token non-fungible adalah token yang mewakili kepemilikan unik atas suatu aset digital atau fisik. Berbeda dengan token kripto fungible seperti Bitcoin atau Ethereum, yang dapat dipertukarkan satu sama lain dengan nilai yang sama, setiap token non-fungible memiliki karakteristik, nilai, dan identitas yang berbeda. Misalnya, token non-fungible dapat mewakili seni digital, koleksi digital, barang-barang virtual dalam permainan, sertifikat kepemilikan, atau bahkan aset fisik yang diwakili secara digital.

Salah satu fitur utama dari ERC-721 adalah kemampuan untuk menciptakan dan melacak kepemilikan aset digital unik. Setiap token ERC-721 memiliki nomor identifikasi unik yang disebut token ID. Token ID ini membedakan setiap token dan memastikan bahwa tidak ada token lain dalam kontrak ERC-721 yang memiliki token ID yang sama. Dengan menggunakan token ID, pengguna dapat dengan mudah mengidentifikasi dan membuktikan kepemilikan mereka atas aset digital yang direpresentasikan oleh token tersebut.

Selain itu, token ERC-721 memungkinkan transfer kepemilikan antara pengguna. Pemilik token dapat mentransfer token ERC-721 kepada orang lain, baik melalui proses penjualan, pertukaran, atau hadiah. Dalam kontrak ERC-721, terdapat fungsi-fungsi yang mendukung transfer kepemilikan, seperti fungsi `transferFrom`, `approve`, dan `safeTransferFrom`. Hal ini memungkinkan terjadinya perdagangan dan pertukaran aset digital unik antara pengguna, menciptakan pasar yang dinamis dan ekonomi yang berkembang di dalam ekosistem NFT.

Selain itu, ERC-721 menyediakan fungsi-fungsi tambahan untuk mengelola dan mengakses informasi terkait dengan token non-fungible. Dalam kontrak ERC-721, pengembang da-

pat menyertakan metadata yang menggambarkan atribut-atribut unik dari aset yang direpresentasikan oleh token. Metadata ini dapat berisi informasi seperti nama aset, deskripsi, gambar, URL, atau atribut khusus lainnya. Dengan adanya metadata, pengguna dapat dengan mudah mengeksplorasi dan memahami karakteristik serta keaslian aset digital yang diwakili oleh token ERC-721.

Standar ERC-721 telah digunakan dalam berbagai aplikasi dan platform NFT yang populer. Misalnya, aplikasi pasar NFT seperti OpenSea, Rarible, dan SuperRare menggunakan standar ini untuk memfasilitasi perdagangan dan pertukaran aset digital. Selain itu, permainan blockchain seperti CryptoKitties dan Decentraland juga menggunakan ERC-721 untuk mewakili barang-barang unik dalam permainan. Standar ini memberikan kerangka kerja yang jelas dan terstandarisasi bagi pengembang untuk menciptakan dan mengelola token NFT secara efisien di jaringan Ethereum. (Shirley, 2018)

2.12 Minting Token

Dalam konteks smart contract, "minting token" merujuk pada proses penciptaan dan penerbitan token baru di jaringan blockchain. Proses ini penting dalam menciptakan dan memperluas ekosistem token yang digunakan dalam aplikasi dan platform blockchain.

Proses minting token biasanya dilakukan menggunakan smart contract yang mengikuti standar token seperti ERC-20 atau ERC-721 di jaringan Ethereum. Kontrak pintar ini memuat fungsi khusus yang memungkinkan pemanggilan untuk menciptakan token baru dan menetapkan atribut-atributnya. Proses minting ini biasanya dilakukan oleh pemilik smart contract atau oleh pihak lain yang memiliki hak akses dan wewenang yang sesuai.

Dalam smart contract berstandar ERC-20, fungsi mint atau fungsi serupa digunakan untuk menciptakan token baru. Fungsi ini menerima parameter seperti jumlah token yang akan dicetak dan alamat pemilik awal token. Setelah fungsi tersebut dipanggil, smart contract akan menciptakan token baru dan menetapkan atribut-atributnya, seperti simbol, nama, jumlah total pasokan, dan pemilik awal.

Dalam smart contract berstandar ERC-721, proses minting token juga melibatkan fungsi mint atau fungsi serupa. Namun, dalam konteks token non-fungible (NFT), setiap token memiliki identitas dan karakteristik yang unik. Oleh karena itu, proses minting dalam smart contract ERC-721 melibatkan penetapan atribut khusus untuk setiap token yang diciptakan, seperti metadata yang menggambarkan aset yang direpresentasikan oleh token tersebut.

Selama proses minting token, catatan transaksi yang mencatat penciptaan dan penerbitan token baru juga dicatat secara permanen dalam blockchain. Ini memastikan keaslian dan jejak kepemilikan token tersebut. Informasi ini dapat diakses dan diverifikasi oleh semua peserta jaringan, memberikan transparansi dan kepercayaan dalam ekosistem token.

Proses minting token memiliki implikasi yang luas dalam ekosistem blockchain. Ini memungkinkan penciptaan token baru untuk digunakan dalam berbagai aplikasi, seperti aplikasi keuangan, pasar NFT, game blockchain, identitas digital, dan banyak lagi. Dengan minting token, pengembang dapat menciptakan ekonomi digital yang dinamis dan inovatif di atas teknologi blockchain. (Shirley, 2018)

2.13 Metaverse

Metaverse adalah istilah yang digunakan untuk menggambarkan dunia virtual yang terhubung secara digital, yang dihuni oleh pengguna dari berbagai lokasi di seluruh dunia. Konsep Metaverse mirip dengan realitas virtual, tetapi dengan tambahan elemen interaktif yang memungkinkan pengguna untuk berinteraksi dengan lingkungan virtual dan antara satu sama lain.

Metaverse bertujuan untuk menciptakan pengalaman yang imersif dan menyatu dengan menggunakan teknologi seperti realitas virtual (VR), augmented reality (AR), dan mixed reality (MR). Di dalam Metaverse, pengguna dapat mengontrol avatar mereka, menjelajahi lingkungan virtual, berinteraksi dengan objek dan orang lain, serta terlibat dalam berbagai aktivitas, seperti bermain game, berkomunikasi, berbelanja, atau bahkan bekerja. (Kietzmann et al., 2011)

Konsep Metaverse telah mendapatkan perhatian yang signifikan dalam beberapa tahun terakhir, terutama dengan perkembangan teknologi seperti blockchain, digital assets, dan NFT (Non-Fungible Tokens). Teknologi-teknologi ini memungkinkan pembangunan dan pertukaran aset digital di dalam Metaverse, termasuk properti virtual, koleksi seni digital, karakter game, dan banyak lagi. Pengguna dapat memiliki, memperdagangkan, dan menggunakannya di dalam lingkungan Metaverse.

Selain itu, Metaverse juga mencakup elemen sosial yang kuat. Pengguna dapat berinteraksi dengan pengguna lainnya melalui avatar mereka, berkomunikasi melalui suara atau teks, membentuk komunitas, dan bahkan melakukan kegiatan sosial seperti menghadiri konser, pameran seni, atau acara virtual lainnya. Konsep Metaverse mendukung kolaborasi, eksplorasi, dan kreasi bersama di dalam dunia virtual.

Namun, perlu dicatat bahwa saat ini, Metaverse masih dalam tahap perkembangan dan implementasi yang terbatas. Ada berbagai platform dan proyek yang berusaha membangun dan mewujudkan visi Metaverse, seperti Decentraland, Cryptovoxels, Somnium Space, dan lainnya. Perkembangan teknologi dan adopsi yang lebih luas akan menjadi kunci untuk mewujudkan potensi penuh Metaverse di masa depan. (Castronova, 2005)

2.14 Interoperabilitas

Interoperabilitas blockchain merujuk pada kemampuan berbagai jaringan blockchain untuk berkomunikasi, berbagi data, dan bekerja bersama secara mulus. Ini mencakup kemampuan untuk mentransfer aset digital antara berbagai platform blockchain, memvalidasi dan mengonfirmasi transaksi lintas rantai, serta mengintegrasikan fungsi dan aplikasi blockchain yang berbeda.

Interoperabilitas blockchain sangat penting dalam mengatasi beberapa tantangan yang dihadapi oleh ekosistem blockchain saat ini. Dalam ekosistem blockchain yang terfragmentasi, setiap jaringan blockchain biasanya beroperasi secara terpisah dengan protokol, struktur data, dan kontrak pintar yang berbeda. Hal ini mengakibatkan adanya silo data yang tidak kompatibel, hambatan dalam berbagi informasi, dan kesulitan dalam mentransfer aset antara jaringan blockchain yang berbeda. (De Angelis & Panzieri, 2019)

Dengan adanya interoperabilitas blockchain, berbagai jaringan blockchain dapat terhubung dan berinteraksi secara langsung, memungkinkan aliran data yang mulus dan transfer aset lintas rantai. Ini memungkinkan pengguna untuk memanfaatkan manfaat dan fungsionalitas dari berbagai jaringan blockchain, meningkatkan skalabilitas, efisiensi, dan fleksibilitas.

Ada beberapa pendekatan yang digunakan untuk mencapai interoperabilitas blockchain. Salah satu pendekatan umum adalah penggunaan protokol jembatan (bridge protocol) yang

memungkinkan komunikasi dan transfer aset antara berbagai blockchain. Protokol ini memfasilitasi penciptaan pintu gerbang (gateways) yang memungkinkan aset yang ada di satu blockchain dipindahkan ke blockchain lain, diubah menjadi format yang kompatibel, dan diteruskan ke penerima di jaringan lain. (Swanson, 2016)

Selain itu, protokol standar juga dapat digunakan untuk mencapai interoperabilitas. Protokol standar mendefinisikan aturan dan format yang sama untuk berbagai blockchain, memungkinkan mereka untuk beroperasi dengan cara yang konsisten dan saling berkomunikasi. Contohnya adalah ERC-20 (Ethereum Request for Comments 20), yang adalah standar token pada blockchain Ethereum yang digunakan secara luas dalam pengembangan token di berbagai jaringan.

Interoperabilitas blockchain juga dapat dicapai melalui penggunaan teknologi seperti sidechain dan cross-chain. Sidechain adalah blockchain terpisah yang terhubung dengan blockchain utama, memungkinkan transfer aset dan data di antara keduanya dengan mekanisme yang aman dan terpercaya. Cross-chain, di sisi lain, mengacu pada kemampuan untuk mentransfer aset dan informasi lintas rantai secara langsung antara blockchain yang berbeda. (Brunnhofer & Pregenzer, 2019)

Dengan adanya interoperabilitas blockchain, ekosistem blockchain dapat tumbuh dan berkembang secara kolektif. Ini membuka peluang untuk pengembangan aplikasi dan layanan yang lebih luas, peningkatan likuiditas pasar aset digital, dan kolaborasi yang lebih baik antara berbagai proyek blockchain. (Huang & Chen, 2021)

2.15 *Blueprint*

Blueprint adalah sistem pemrograman visual yang ada di Unreal Engine 5. Dengan menggunakan Blueprint, pengembang permainan dapat membuat logika permainan, fungsi, dan interaksi antarmuka pengguna tanpa menulis kode secara langsung. Blueprint menggunakan node dan aliran kerja yang terhubung untuk menggambarkan alur logika permainan secara visual.

Dalam Unreal Engine 5, Blueprint telah mengalami beberapa pembaruan dan peningkatan fitur. Salah satu fitur utama dari Blueprint di Unreal Engine 5 adalah peningkatan kecepatan dan efisiensi. Blueprint sekarang dapat dieksekusi lebih cepat, yang menghasilkan waktu loading yang lebih pendek dan kinerja yang lebih baik secara keseluruhan.

Selain itu, Unreal Engine 5 juga memperkenalkan fitur baru seperti Visual Scripting, yang memungkinkan pengembang menggunakan Blueprint untuk membuat efek visual yang kompleks, seperti partikel dan animasi. Fitur ini memberikan fleksibilitas yang lebih besar dalam menciptakan dunia game yang memukau secara visual. (Games, 2022a)

Blueprint di Unreal Engine 5 juga mendukung fitur-fitur seperti modular scripting, debugging, dan reusable components. Pengembang dapat membagikan dan menggunakan Blueprint yang sudah dibuat sebelumnya, menghemat waktu dan usaha dalam pengembangan permainan.

Dalam hal dokumentasi dan dukungan, Unreal Engine 5 menyediakan panduan dan tutorial yang komprehensif tentang penggunaan Blueprint. Pengembang dapat mengakses dokumentasi resmi Unreal Engine, forum pengguna, dan sumber daya online lainnya untuk mempelajari lebih lanjut tentang Blueprint dan mendapatkan bantuan jika diperlukan.

Secara keseluruhan, Blueprint di Unreal Engine 5 memberikan alat yang kuat dan fleksibel bagi pengembang permainan untuk mewujudkan visi mereka tanpa harus menjadi ahli dalam pemrograman. Dengan antarmuka visual yang intuitif, pengembang dapat menciptakan penga-

laman permainan yang menarik dan inovatif. (Games, 2021)

[Halaman ini sengaja dikosongkan]

BAB III

METODOLOGI

Penelitian ini dilaksanakan sesuai desain sistem berikut beserta implementasinya. Desain sistem merupakan konsep dari pembuatan dan perancangan infrastruktur dan kemudian diwujudkan dalam bentuk blok-blok alur yang harus dikerjakan. Pada bagian implementasi merupakan pelaksanaan teknis untuk setiap blok pada desain sistem.

3.1 Software

Sejumlah software digunakan sebagai pendukung pengerjaan Tugas Akhir ini. Software yang berhubungan dengan Ethereum Blockchain diantaranya Remix, Metamask, Ganache, dan Unreal Engine 5. Kemudian dari segi platform, yang digunakan adalah IPFS dan Plugin-plugin yang ada di Unreal Engine 5 serta blueprints-nya. Sementara dari Ethereum Blockchain, bahasa pemrograman yang digunakan adalah Solidity.

3.1.1 Remix IDE

Remix adalah sebuah lingkungan pengembangan (IDE) yang populer dan terbuka untuk kontrak pintar (smart contract) Ethereum yang ditulis dalam bahasa Solidity. Remix menyediakan antarmuka yang intuitif dan lengkap untuk mengedit, menguji, dan menerapkan kontrak pintar secara langsung dari browser web tanpa perlu mengunduh atau menginstal perangkat lunak tambahan.

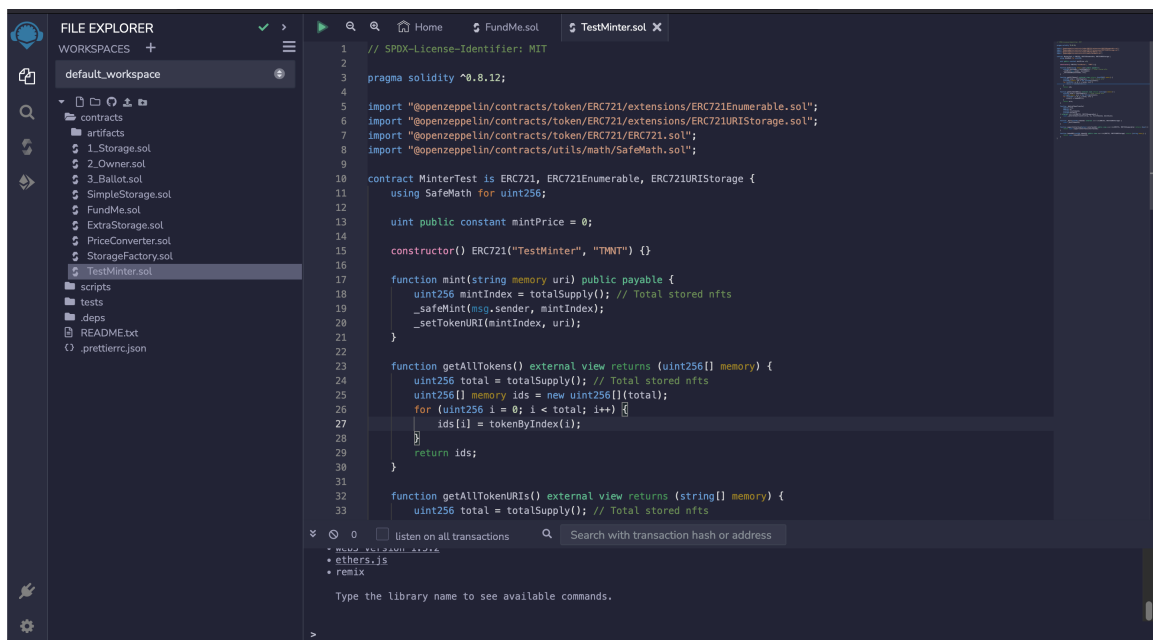
Remix menyediakan berbagai fitur yang memudahkan pengembangan kontrak pintar dalam bahasa Solidity. Beberapa fitur utama dari Remix meliputi:

1. Editor Solidity: Remix menyediakan editor Solidity yang terintegrasi, yang memungkinkan pengembang untuk menulis, mengedit, dan memeriksa sintaksis dan kesalahan dalam kontrak pintar mereka. Editor ini juga menawarkan fitur seperti penyorotan sintaksis, penjadaran otomatis, dan saran kode (code completion) untuk mempercepat pengembangan.
2. Penjelajah File: Remix memungkinkan pengembang untuk melihat struktur proyek mereka dengan menggunakan penjelajah file terintegrasi. Pengembang dapat dengan mudah menjelajahi dan mengatur file kontrak pintar, library, dan dependensi lainnya.
3. Pemecah Masalah (Debugger): Remix dilengkapi dengan pemecah masalah yang kuat yang memungkinkan pengembang untuk menguji dan menganalisis langkah-demi-langkah eksekusi kontrak pintar. Dengan pemecah masalah, pengembang dapat memverifikasi logika dan perilaku kontrak pintar serta menemukan dan memperbaiki bug dengan lebih efisien.
4. Simulasi dan Uji: Remix menyediakan fitur simulasi dan pengujian yang memungkinkan pengembang untuk menjalankan dan menguji kontrak pintar dalam lingkungan virtual

sebelum diterapkan di jaringan Ethereum yang sebenarnya. Fitur ini membantu pengembang untuk mengidentifikasi dan memperbaiki masalah sebelum kontrak pintar diterapkan secara nyata.

5. Integrasi Jaringan: Remix dapat terhubung ke berbagai jaringan Ethereum, termasuk jaringan pengembangan lokal seperti Ganache atau jaringan Ethereum utama. Ini memungkinkan pengembang untuk menguji dan menerapkan kontrak pintar pada berbagai lingkungan dengan mudah.

Remix telah menjadi salah satu alat yang sangat populer dan diandalkan dalam komunitas pengembangan Ethereum. Antarmuka yang kuat dan berfitur lengkap, serta kemudahan penggunaan langsung dari browser, menjadikan Remix sebagai pilihan favorit bagi pengembang untuk mengembangkan, menguji, dan menerapkan kontrak pintar dalam bahasa Solidity.



Gambar 3.1: Tampilan *Remix IDE*.

3.1.2 Ganache

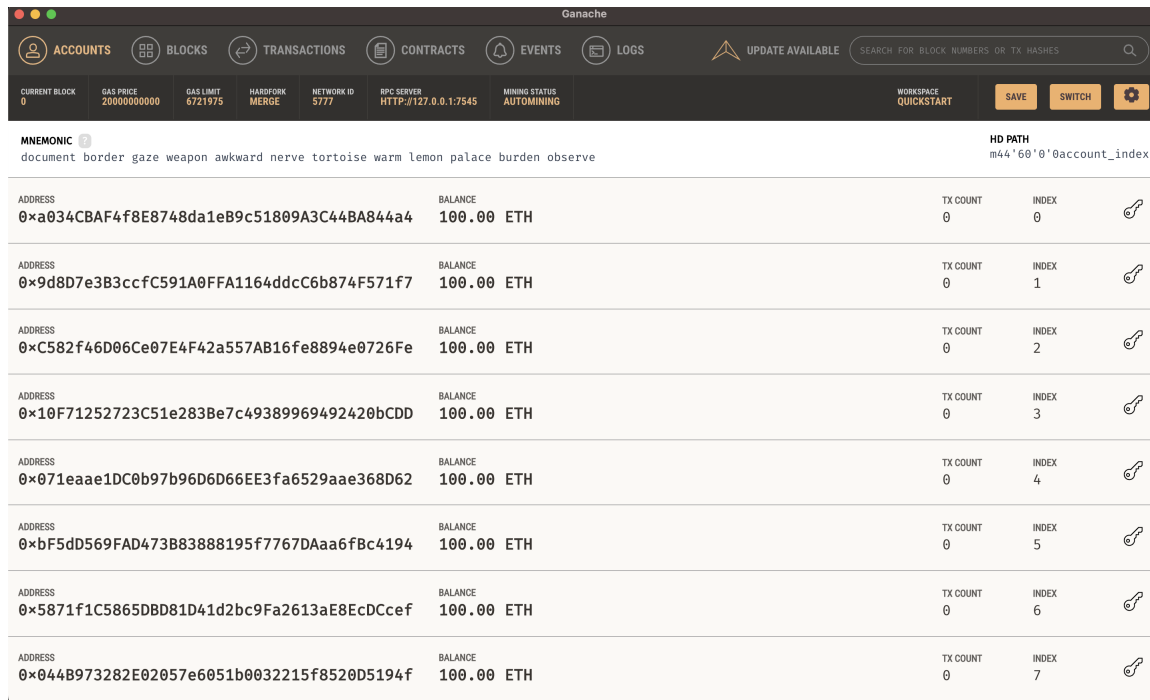
Ganache adalah sebuah perangkat lunak yang digunakan dalam pengembangan aplikasi berbasis blockchain, khususnya pada platform Ethereum. Ini adalah salah satu alat yang sangat populer di antara pengembang karena menyediakan lingkungan pengujian lokal yang mudah digunakan dan fleksibel.

Ganache awalnya dikenal dengan nama "TestRPC" sebelum berganti nama menjadi "Ganache". Perangkat lunak ini memungkinkan pengembang untuk melakukan pengembangan, pengujian, dan debugging aplikasi blockchain secara lokal tanpa perlu terhubung ke jaringan Ethereum yang sebenarnya. Dengan menggunakan Ganache, pengembang dapat membuat jaringan Ethereum pribadi mereka sendiri di mesin lokal mereka.

Berikut adalah beberapa fitur utama yang ditawarkan oleh Ganache:

1. Jaringan Ethereum Lokal: Ganache menyediakan jaringan Ethereum lokal yang dapat dijalankan pada mesin pengembang. Jaringan ini berjalan di dalam lingkungan sandbox yang aman, yang memungkinkan pengembang untuk membuat dan menguji kontrak pintar, mentransaksikan token, dan menjalankan semua operasi yang terkait dengan Ethereum tanpa risiko kehilangan dana atau interaksi dengan jaringan utama.
2. Blockchain Deterministik: Salah satu fitur kunci dari Ganache adalah kepastian (determinism) dari blockchain yang dihasilkan. Ini berarti bahwa setiap kali Ganache dijalankan dengan pengaturan yang sama, blockchain akan memiliki keadaan awal yang identik. Ini memungkinkan pengembang untuk menguji dan mengulangi skenario dengan hasil yang konsisten, yang sangat penting dalam pengujian dan debugging aplikasi blockchain.
3. Mode Pembuatan Blok: Ganache menawarkan beberapa mode pembuatan blok yang dapat disesuaikan sesuai dengan kebutuhan pengembangan. Mode ini memungkinkan pengembang untuk mengendalikan bagaimana blok dibuat dan transaksi dikonfirmasi. Misalnya, pengembang dapat mengatur blok agar hanya dibuat setiap kali ada transaksi, atau mengatur kecepatan pembuatan blok agar sesuai dengan kecepatan transaksi yang diinginkan.
4. Akun dan Kunci Pribadi: Ganache menyediakan sejumlah akun Ethereum yang siap digunakan dalam pengembangan. Setiap akun dilengkapi dengan kunci pribadi yang terkait, sehingga pengembang dapat menguji fungsi-fungsi yang melibatkan transaksi, seperti pengiriman Ether atau eksekusi kontrak pintar.
5. Alat Pengembangan: Ganache dilengkapi dengan berbagai alat pengembangan yang membantu dalam memahami dan menganalisis aktivitas blockchain. Ini termasuk penjelajah blok (block explorer) yang memungkinkan pengembang untuk melihat detail transaksi dan status kontrak pintar, serta alat untuk melacak log dan debug kontrak pintar.

Ganache dapat digunakan dalam berbagai skenario pengembangan, mulai dari pengujian kontrak pintar, pengembangan aplikasi terdesentralisasi, hingga pengujian keamanan. Dengan menggunakan Ganache, pengembang dapat dengan mudah membuat dan mengelola jaringan Ethereum lokal mereka sendiri, mempercepat siklus pengembangan, dan memastikan kualitas dan keandalan aplikasi blockchain yang dikembangkan.



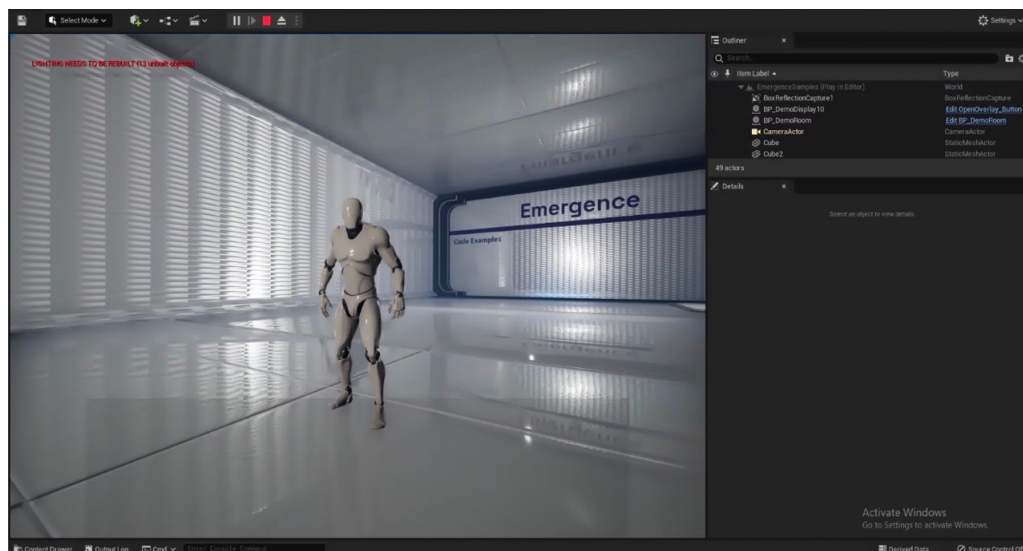
The screenshot shows the Ganache application interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below these tabs, there is a search bar and a table of account information. The table has columns for ADDRESS, BALANCE, TX COUNT, and INDEX. The accounts listed are:

ADDRESS	BALANCE	TX COUNT	INDEX
0xa034CBAF4f8E8748da1eB9c51809A3C44BA844a4	100.00 ETH	0	0
0x9d8D7e3B3ccfC591A0FFA1164ddcC6b874F571f7	100.00 ETH	0	1
0xC582f46D06Ce07E4F42a557AB16fe8894e0726Fe	100.00 ETH	0	2
0x10F71252723C51e283Be7c49389969492420bCDD	100.00 ETH	0	3
0x071eaae1DC0b97b96D6D66EE3fa6529aae368D62	100.00 ETH	0	4
0xbF5dD569FAD473B83888195f7767DAaa6fBc4194	100.00 ETH	0	5
0x5871f1C5865DBD81D41d2bc9Fa2613aE8EcDCcef	100.00 ETH	0	6
0x044B973282E02057e6051b0032215f8520D5194f	100.00 ETH	0	7

Gambar 3.2: Tampilan *Ganache*.

3.1.3 Unreal Engine 5

Unreal Engine yang digunakan adalah versi 5.0.3. Unreal Engine digunakan untuk pengimplementasian *Metaverse* yang mana dibuat sebuah lingkungan virtual sederhana seperti berikut.



Gambar 3.3: Tampilan *Unreal Engine 5*.

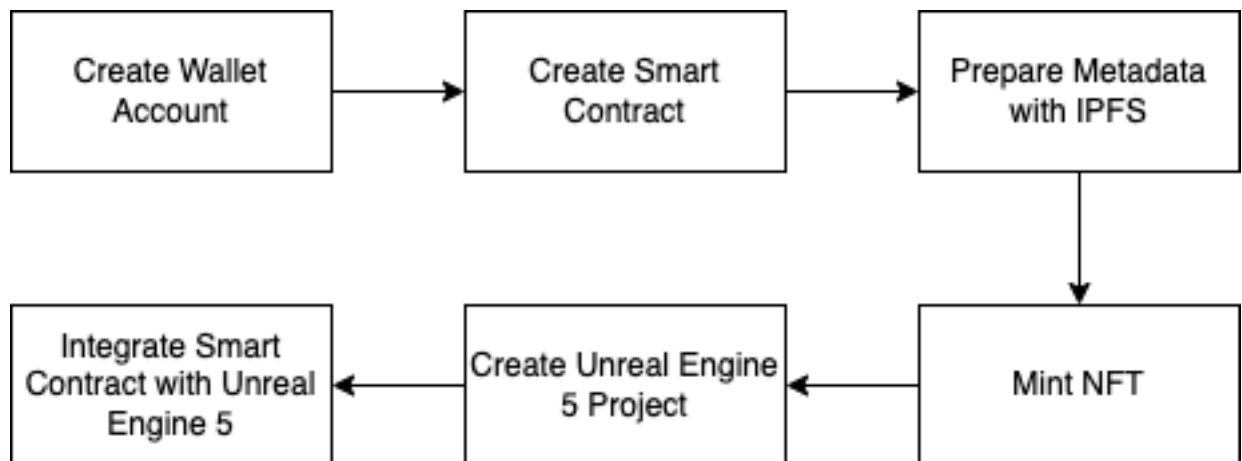
3.2 Desain Sistem

Tugas akhir ini dikerjakan dengan menggunakan beberapa service dan setiap service ini memiliki langkah-langkah tersendiri dalam pengerjaannya. Untuk data *raw* dari audionya dis-

impan di IPFS menggunakan Web3 Storage. Lalu untuk transmisi datanya menggunakan NFT yang akan di mint dengan menggunakan protokol *ERC-721*. Proses pembuatan NFT dilakukan dengan menggunakan smart contract yang ditulis dengan bahasa *Solidity*

Penulisan *Smart contract* dengan menggunakan *solidity* dilakukan dengan menggunakan *Remix IDE*. *Smart contract* yang ditulis diberikan beberapa *method* tambahan yang akan membantu proses pengembangan dan integrasi ke *Unreal Engine 5*

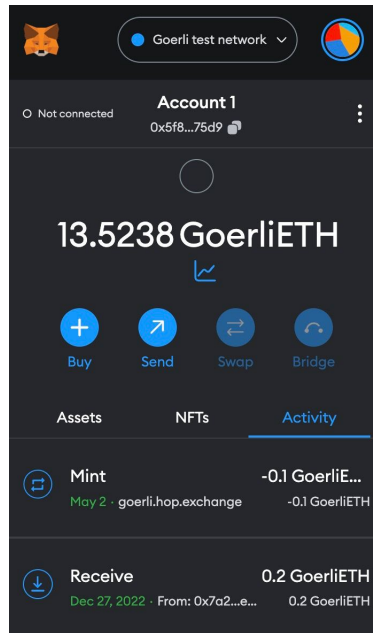
Setelah *smart contract* ditulis, *smart contract* dideploy. Kemudian dilakukan mint token dengan menggunakan tampilan antarmuka dari *Remix IDE*. Setelah sudah dilakukan deployment maka selanjutnya dilakukan pengintegrasian dengan *Unreal Engine 5*. NFT yang sudah di-mint dapat diperoleh dengan menggunakan *method smart contract* yang sudah ditulis sebelumnya. *Metadata* yang tersimpan di NFT akan digunakan dan file audio yang terkandung didalamnya akan di-play.



Gambar 3.4: Desain Sistem

3.3 Create Wallet Account

Proses pertama yang dilakukan adalah pembuatan akun wallet Ethereum dengan bantuan Metamask. Metamask merupakan sebuah plugin browser yang berfungsi sebagai dompet penyimpanan Ethereum. Plugin ini dapat berinteraksi dengan Blockchain Ethereum yang didalamnya banyak terdapat aplikasi yang terdesentralisasi. Jaringan Ethereum dalam Metamask sangat beragam. Secara defaultnya, jaringan terdiri dari 2 jenis yakni Mainnet dan juga Testnet.



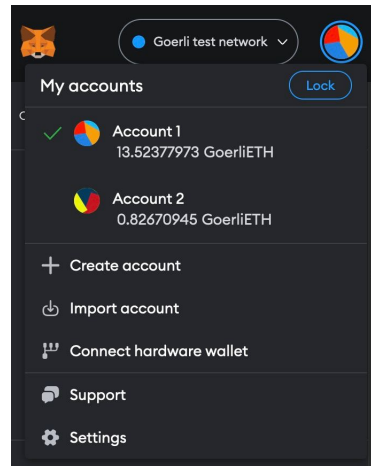
Gambar 3.5: Tampilan Metamask

Mainnet mengharuskan user untuk mengisi dompet Metamask dengan mata uang Ether asli, sehingga Metamask ini bisa digunakan untuk transaksi sehari-hari melewati jaringan Mainnet ini. Sedangkan jaringan Testnet merupakan jaringan berbentuk perco- baan yang biasa di- gunakan untuk pengembangan aplikasi yang ingin diintegrasikan dengan Metamask. Pilihan Testnet ini juga tidak hanya satu, melainkan defaultnya terdapat 4 jaringan Testnet dalam Meta- mask, antara lain Ropsten, Goerli, Kovan, dan Rinkeby.

Masing-masing Testnet ini memiliki algoritma konsensus seperti Proof of Work dan Proof of Authority. Ropsten Testnet menggunakan algoritma Proof of Work dan Testnet lainnya meng- gunakan Proof of Authority. Untuk Proof of Stake dalam default plugin Metamask belum tersedia, namun Testnet dengan algoritma ini sudah tersedia seperti Kiln dan Kitsugi, dapat ditambahkan jaringannya pada Metamask.

Setelah membuat akun, akan diberikan Secret Recovery Phrase yang merupakan 12 kata unik dari Metamask sebagai cadangan untuk bisa masuk ke akun Metamask ketika password untuk login terlupakan. Secret Recovery Phrase ini dianjurkan untuk tidak disebar- kan karena pengguna lain yang tidak memiliki akses ke password bisa mengakses akun Metamask orang lain hanya dengan Recovery Secret Phrase. Dalam plugin Metamask, pengguna dibebaskan membuat sejumlah akun lewat jaringan manapun. Untuk project ini, digunakan Testnet Ropsten yang menganut algoritma Proof of Work serta jaringan lokal Blockchain yang diperoleh dari Ganache sebagai mata uang untuk pengujian Smart Contract dan aplikasi.

Pada tahap ini, akun Testnet secara default akan memiliki 0 Ether, namun akun tersebut dapat memperoleh Ether dengan mencari faucet dari Testnet nya. Faucet ini nantinya yang akan mengirimkan sejumlah Ether sehingga akun Testnet bisa memperoleh Ether tanpa melakukan komputasi apapun. Maka dengan demikian, akun ini dinamakan Externally Owned Account, yang bisa dikendalikan langsung oleh pengguna tanpa perlu dikaitkan program. Nominal Test- net yang didapat hasil dari faucet bisa dikirimkan lang- sung ke akun lain di Testnet yang sama dengan fitur transfer between account, inilah fitur yang menjelaskan Externally Owned Account (EOA).



Gambar 3.6: Tampilan Akun Goerli di Metamask

Akun wallet dari Metamask ini merupakan Ethereum Account yang dibuat dari kriptografis private dan public key. Dengan private key, membantu membuktikan bahwa transaksi benar-benar ditandatangani oleh pengirim dan mencegah pemalsuan. Public key dalam Metamask berarti sama dengan wallet address dari akun token Testnet yang digunakan (biasanya berawal dari 0x). Karena account Ethereum ini membutuhkan private key untuk menandatangani transaksi agar bisa dikonfirmasi pada ledger, maka dari itu sistem ini menggunakan jenis yang sama pada sistem Multi Signature yaitu menggunakan 2 atau lebih private key untuk menandatangani transaksi dari akun yang berbeda. Komparasi dengan sistem wallet di Indonesia saat ini menunjukkan perbedaan dimana di Indonesia, sistemnya saat ini menggunakan 2 factor authentication dengan OTP dan juga PIN untuk melakukan transaksi sehingga media nya berbeda. Tetapi pada sistem layanan uang digital di Indonesia masih bersifat sentralisasi, sedangkan pada Multi Signature karena memanfaatkan teknologi Blockchain Ethereum, bersifat desentralisasi. Pada implementasinya, Multi Signature dapat menggunakan prinsip 2FA dengan cara menggunakan device yang berbeda. Satu orang dapat mengimplementasikan sistem ini jika memiliki lebih dari satu device untuk melakukan koneksi ke wallet yang sudah memiliki sistem Multi Signature

3.4 Create *Smart Contract*

Pembuatan Smart Contract merupakan salah satu kelebihan dari Ethereum, dimana fitur didalam wallet Ether semua diprogram dalam Smart Contract sesuai dengan kebutuhan. Token yang akan digunakan adalah token dengan spesifikasi ERC-721 atau disebut juga dengan NFT (Non Fungible Token) Pertama ditentukan spesifikasi token ERC-721 yang ingin dibuat. Ini meliputi pengaturan seperti nama token, simbol, dan jumlah total token yang akan dikeluarkan. Dapat juga dipertimbangkan apakah perlu menyertakan metadata tambahan seperti deskripsi, gambar, atau atribut khusus. Kontrak pintar (smart contract) harus dibuat menggunakan bahasa pemrograman Solidity dengan library standar ERC-721 diimpor dan kontrak didefinisikan sebagai kontrak yang mengimplementasikan interface ERC-721. Fungsi-fungsi utama seperti transfer token, melihat informasi token, dan lainnya perlu diatur. Logika bisnis yang diinginkan perlu diimplementasikan pada token ERC-721. Uji dan rilis harus dilakukan untuk memastikan kontrak berfungsi sebagaimana yang diharapkan. Kontrak dapat diuji secara lokal menggunakan alat pengujian seperti Ganache atau Remix sebelum dirilis ke jaringan Ethereum.

Dalam smart contract yang dibuat, diberikan method-method tambahan seperti tokenURI,

getAllTokens, dan getAllTokenURIs.

```
1 // SPDX-License-Identifier: MIT
2
3 pragma solidity ^0.8.12;
4
5 import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.↵
  sol";
6 import "@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.↵
  sol";
7 import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
8 import "@openzeppelin/contracts/utils/math/SafeMath.sol";
9
10 contract MinterTest is ERC721, ERC721Enumerable, ERC721URIStorage {
11     using SafeMath for uint256;
12
13     uint public constant mintPrice = 0;
14
15     constructor() ERC721("TestMinter", "TMNT") {}
16
17     function mint(string memory uri) public payable {
18         uint256 mintIndex = totalSupply(); // Total stored nfts
19         _safeMint(msg.sender, mintIndex);
20         _setTokenURI(mintIndex, uri);
21     }
22
23     function getAllTokens() external view returns (uint256[] memory) {
24         uint256 total = totalSupply(); // Total stored nfts
25         uint256[] memory ids = new uint256[](total);
26         for (uint256 i = 0; i < total; i++) {
27             ids[i] = tokenByIndex(i);
28         }
29         return ids;
30     }
31
32     function getAllTokenURIs() external view returns (string[] memory) {
33         uint256 total = totalSupply(); // Total stored nfts
34         string[] memory uris = new string[](total);
35         for (uint256 i = 0; i < total; i++) {
36             uris[i] = tokenURI(i);
37         }
38         return uris;
39     }
40
41     function _beforeTokenTransfer(
42         address from,
43         address to,
44         uint256 firstTokenId,
45         uint256 batchSize
46     ) internal override(ERC721, ERC721Enumerable) {
47         super._beforeTokenTransfer(from, to, firstTokenId, batchSize);
48     }
49
50     function _burn(uint256 tokenId) internal override(ERC721, ↵
      ERC721URIStorage) {
51         super._burn(tokenId);
52     }
53 }
```

```

53
54     function supportsInterface(bytes4 interfaceId) public view override(←
        ERC721, ERC721Enumerable) returns (bool) {
55         return super.supportsInterface(interfaceId);
56     }
57
58     function tokenURI(uint256 tokenId) public view override(ERC721, ←
        ERC721URIStorage) returns (string memory) {
59         return super.tokenURI(tokenId);
60     }
61 }

```

Listing 3.1: Smart Contract untuk Mint NFT.

Listing diatas adalah kontrak smart contract Solidity yang menggunakan beberapa kontrak dari library OpenZeppelin untuk mengimplementasikan ERC-721, ERC-721Enumerable, dan ERC-721URIStorage. Berikut adalah pembahasan secara lebih rinci setiap bagian dari kode tersebut:

3.4.1 Kontrak Import dan Inheritance

Kontrak menggunakan @openzeppelin/contracts library untuk mengimpor kontrak ERC721, ERC721Enumerable, dan ERC721URIStorage. Kontrak MinterTest adalah turunan dari ketiga kontrak tersebut.

3.4.2 Konstruktor

Konstruktor digunakan untuk menginisialisasi kontrak dengan memberikan nama dan simbol untuk token ERC-721 yang dibuat. Dalam kode ini, konstruktor didefinisikan menggunakan fungsi constructor() dengan menggunakan konstruktor kontrak ERC721.

3.4.3 Fungsi Mint

Fungsi mint digunakan untuk membuat token baru dan meng-assign kepemilikannya kepada pengirim transaksi. Fungsi ini menerima parameter uri yang merupakan URI unik untuk token yang akan dibuat. Setelah token dibuat, fungsi _safeMint digunakan untuk mengalokasikan token kepada pengirim transaksi. Fungsi _setTokenURI digunakan untuk mengatur URI token.

3.4.4 Fungsi getAllTokens

Fungsi getAllTokens digunakan untuk mengembalikan array dengan daftar semua token yang ada dalam kontrak. Fungsi ini menggunakan fungsi totalSupply untuk mendapatkan total jumlah token yang ada. Kemudian, dalam loop, fungsi tokenByIndex digunakan untuk mendapatkan ID token pada indeks tertentu.

3.4.5 Fungsi getAllTokenURIs

Fungsi getAllTokenURIs digunakan untuk mengembalikan array dengan daftar URI semua token dalam kontrak. Fungsi ini juga menggunakan fungsi totalSupply untuk mendapatkan total jumlah token yang ada. Dalam loop, fungsi tokenURI digunakan untuk mendapatkan URI token pada indeks tertentu.

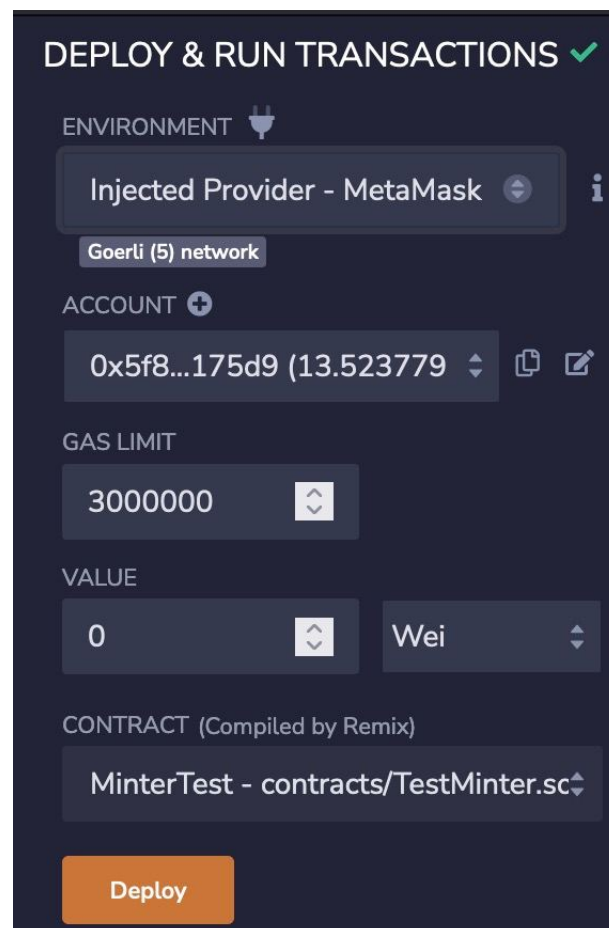
3.4.6 Fungsi `_beforeTokenTransfer` dan `_burn`

Fungsi `_beforeTokenTransfer` dan `_burn` merupakan fungsi internal yang di-override dari kontrak ERC721 dan ERC721URIStorage. Fungsi `_beforeTokenTransfer` dijalankan sebelum token transfer terjadi. Fungsi `_burn` dijalankan saat token di-burn (dihapus) dari kontrak.

3.4.7 Fungsi `supportsInterface` dan `tokenURI`

Fungsi `supportsInterface` dan `tokenURI` adalah fungsi yang di-override dari kontrak ERC721 dan ERC721URIStorage. Fungsi `supportsInterface` memeriksa apakah kontrak mendukung interface tertentu. Fungsi `tokenURI` mengembalikan URI token berdasarkan ID token yang diberikan.

Setelah dilakukan penulisan kode solidity maka dapat dilakukan deployment. Deployment dilakukan dengan environment *Injected Provider - MetaMask*



Gambar 3.7: Antarmuka Deployment di Remix IDE

3.5 Prepare Metadata with IPFS

Untuk melakukan persiapan metadata, maka perlu ditentukan dulu format JSON yang akan digunakan. Format JSON yang digunakan adalah format dari OpenSea yang adalah seperti berikut.


```

15     "value": "ariel"
16   }
17 ]
18 }

```

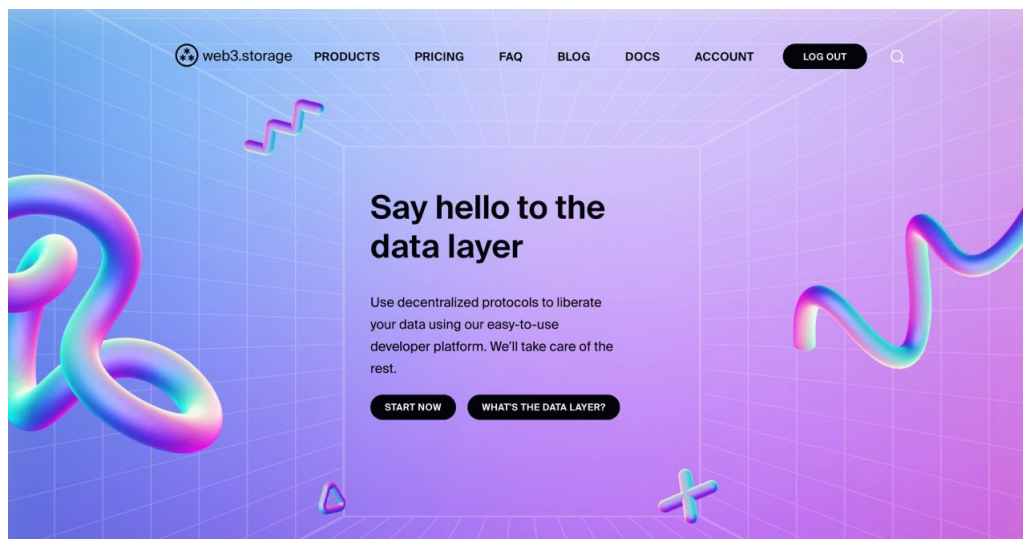
Listing 3.2: Format JSON untuk metadata NFT.

3.5.1 Web3 Storage

Service IPFS yang digunakan adalah Web3 Storage. Semua asset-asset yang akan digunakan diunggah di Web3 Storage.

web3.storage adalah kumpulan API dan layanan yang memudahkan pengembang dan pengguna lain untuk berinteraksi dengan data tanpa terikat pada lokasi fisik penyimpanan data tersebut. Layanan ini secara asli menggunakan protokol data dan identitas terdesentralisasi seperti IPFS, Filecoin, dan UCAN yang memungkinkan arsitektur dan alur kerja aplikasi yang berfokus pada verifikasi data dan pengguna.

Di inti platform ini terdapat layanan penyimpanan terhosting yang dapat digunakan untuk mengunggah dan menyimpan data agar tetap tersedia secara berkelanjutan. Platform ini juga menyediakan layanan tambahan seperti w3link dan w3name yang memudahkan pembuatan pengalaman web yang mulus dan menyenangkan dengan memanfaatkan protokol web3.



Gambar 3.9: Tampilan web3.storage

Dengan menggunakan web3.storage akan diupload file-file yang akan digunakan untuk data dari sistem. Data-data atau assets ini menggunakan audio yang berlisensi CC0.

Files

Upload +

Refresh

Sort By

Gateway

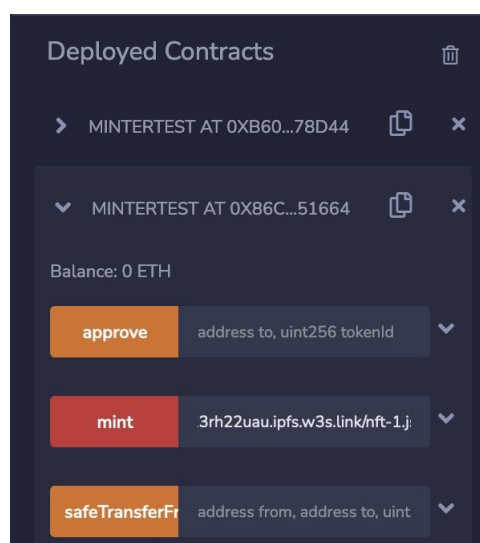
<input type="checkbox"/>	Name		CID	Complete file?	Storage Providers	Size	Date
<input type="checkbox"/>	nft-1.json	✓	bafyb...3rh22uau	Complete	Queuing...	775 B	5/26/2023
<input type="checkbox"/>	black-box-legendary-9509.mp3	✓	bafyb...6dh6xy3u	Complete	Queuing...	5.44 MB	5/26/2023
<input type="checkbox"/>	guitar-mellow-beat-20221122-128596.mp3	✓	bafyb...toawlyvm	Complete	Queuing...	6.34 MB	5/26/2023
<input type="checkbox"/>	unknown-realm-147409.mp3	✓	bafyb...j4xdxb6u	Complete	Queuing...	3.96 MB	5/26/2023
<input type="checkbox"/>	green-sky-125179.mp3	✓	bafyb...7z1nt7iy	Complete	Queuing...	2.72 MB	5/26/2023
<input type="checkbox"/>	nft.json	✓	bafyb...oxyw5xq	Complete	Stored (1)	637 B	3/28/2023
<input type="checkbox"/>	hello_world.mp3	✓	bafyb...2na5ktjm	Complete	Stored (2)	4.52 MB	3/28/2023
<input type="checkbox"/>	sample.json	✓	bafyb...fdandfdm	Complete	Stored (1)	308 B	12/27/2022
<input type="checkbox"/>	example.json	✓	bafyb...7z5dztru	Complete	Stored (1)	270 B	12/27/2022
<input type="checkbox"/>	bengpendek.wav	✓	bafyb...dvhqxogm	Complete	Stored (1)	586.04 KB	12/27/2022

Gambar 3.10: Tampilan di menu file web3.storage

Url dari audio yang sudah diupload akan diletakkan kedalam Metadata json, kemudian json tersebut akan diupload juga ke web3.storage.

3.6 Mint NFT

Di tahap ini dilakukan proses mint NFT dengan menggunakan *smart contract* yang sudah dideploy sebelumnya. Remix IDE sudah menyediakan antarmuka untuk memanggil method-method yang ada di smart contract yang sudah terdeploy.



Gambar 3.11: Tampilan Remix IDE untuk deployed contracts

Dapat juga dicek melalui etherscan. Domain etherscan yang digunakan adalah milik go-

erli karena wallet yang digunakan juga menggunakan goerli. Etherscan Goerli adalah sebuah platform yang memberikan berbagai macam layanan dan utilitas yang berguna dalam pengembangan smart contract di jaringan Goerli, yang merupakan salah satu jaringan pengujian pada Ethereum. Platform ini menyediakan berbagai fitur yang memungkinkan pengembang untuk menjelajahi, memantau, dan menganalisis smart contract dan transaksi yang terjadi di jaringan Goerli.

Salah satu kegunaan utama dari Etherscan Goerli adalah untuk menjelajahi dan menganalisis smart contract. Pengembang dapat menggunakan platform ini untuk melihat detail dari suatu smart contract, termasuk kode sumbernya, fungsi-fungsi yang ada, dan variabel-variabel yang digunakan. Informasi ini sangat penting dalam memahami dan memeriksa kecerdasan kontrak sebelum diterapkan di jaringan Ethereum utama. Selain itu, Etherscan Goerli juga menyediakan alat untuk memeriksa riwayat transaksi, sehingga pengembang dapat melacak aliran nilai dan interaksi yang terjadi pada suatu smart contract.

Selain menjelajahi dan menganalisis smart contract, Etherscan Goerli juga menyediakan utilitas untuk memantau dan melacak kinerja dan keadaan jaringan Goerli. Pengembang dapat memeriksa informasi tentang blok terbaru, hash blok, kecepatan transaksi, dan pemegang akun. Ini membantu pengembang dalam mengamati performa jaringan dan memastikan bahwa transaksi yang mereka kirimkan berjalan dengan baik di jaringan pengujian.

Selain itu, Etherscan Goerli juga memberikan fitur untuk memverifikasi smart contract. Pengembang dapat mengirimkan kode sumber smart contract mereka ke platform ini untuk diverifikasi. Dengan melakukan verifikasi, pengembang dapat memastikan bahwa smart contract yang mereka kembangkan sesuai dengan apa yang diharapkan, dan memastikan bahwa tidak ada perubahan yang tidak diinginkan atau malware yang dimasukkan ke dalam kontrak.

Selain fitur-fitur utama di atas, Etherscan Goerli juga menyediakan berbagai informasi terkait dengan jaringan Goerli, seperti harga gas, grafik transaksi, dan pemantauan kontrak yang populer. Semua informasi ini membantu pengembang dalam memperoleh pemahaman yang lebih baik tentang jaringan Goerli dan membantu mereka dalam mengembangkan dan menguji smart contract dengan lebih efektif.

Dalam pengembangan smart contract, Etherscan Goerli memberikan alat yang sangat berharga untuk menjelajahi, memantau, menganalisis, dan memverifikasi smart contract di jaringan Goerli. Dengan menggunakan platform ini, pengembang dapat memperoleh wawasan yang mendalam tentang keadaan jaringan, memeriksa dan memverifikasi kecerdasan kontrak, serta memastikan performa yang baik dalam pengujian smart contract sebelum diterapkan di jaringan Ethereum utama.

3.7 Create Unreal Engine 5 Project

Proyek Unreal Engine 5 dibuat dengan mengikuti serangkaian langkah yang penting dan harus diikuti dengan cermat. Pertama, Unreal Engine 5 diinstal dari situs resmi Epic Games dan memastikan persyaratan sistem yang diperlukan terpenuhi oleh komputer. Setelah diinstal, Unreal Engine Launcher dibuka dan proyek baru dibuat dengan memilih jenis proyek yang sesuai dengan tujuan. Setelah masuk ke Unreal Editor, pengaturan proyek dikonfigurasi seperti resolusi layar, sistem fisika, kontrol input, dan lainnya sesuai kebutuhan.

Selanjutnya, dunia (World) dibuat di mana permainan atau pengalaman berlangsung. Alat bawaan Unreal Engine 5 seperti Landscape Editor digunakan untuk membuat lingkungan yang

realistis dengan menambahkan peta, objek, karakter, dan lainnya. Selanjutnya, objek dan karakter dibuat atau diimpor ke dalam proyek. Aset dapat dibuat sendiri atau model 3D yang sudah ada diimpor dalam berbagai format seperti FBX atau OBJ. Setelah itu, animasi, tekstur, suara, dan perilaku yang diinginkan ditambahkan.

Selanjutnya, bahasa pemrograman visual bernama Blueprint digunakan untuk membuat logika permainan. Dengan Blueprint, alur permainan, skrip perilaku objek, pengaturan kamera, penanganan interaksi pemain, dan lainnya dapat dibuat. Jika memiliki pengetahuan dalam pemrograman C++, bahasa ini juga dapat digunakan dalam pengembangan proyek. Efek visual dan suara juga dapat diterapkan menggunakan fitur dan alat bawaan Unreal Engine 5, seperti pencahayaan, partikel, efek cuaca, dan suara latar.

Setelah proyek selesai dibuat, pengujian dan iterasi dilakukan. Proyek dijalankan di Unreal Editor atau mode standalone untuk melihat bagaimana proyek berjalan, bug diidentifikasi dan diperbaiki, serta pengujian dilakukan untuk memastikan kualitas dan kinerja yang baik. Terakhir, setelah proyek selesai dan telah diuji dengan baik, proyek dapat didistribusikan kepada pengguna akhir. Unreal Engine 5 menyediakan berbagai opsi distribusi, termasuk platform PC, konsol, dan virtual reality (VR), sesuai dengan kebutuhan proyek.

Proyek Unreal Engine yang dibuat sederhana saja, namun tetap dapat berinteraksi dengan menggunakan blueprints.

Kemudian perlu juga menambahkan beberapa plugin yang akan digunakan nantinya seperti Runtime Audio Importer, IPFS, dan Emergence.

3.7.1 Runtime Audio Importer

Runtime Audio Importer adalah sebuah fitur yang memungkinkan pengguna untuk mengimpor audio dalam berbagai format secara langsung saat berjalan. Fitur ini menawarkan kecepatan transkoding yang cepat dan mendukung format audio utama seperti MP3, WAV, FLAC, OGG Vorbis, dan BINK. Selain itu, fitur ini juga mendukung format RAW seperti int8, uint8, int16, uint16, int32, uint32, dan float32.

Runtime Audio Importer memiliki perilaku yang sama dengan gelombang suara biasa, termasuk SoundCue dan MetaSounds (mulai dari versi 5.2). Fitur ini juga secara otomatis mendeteksi format audio yang digunakan dan mendukung fungsionalitas streaming audio. Selain itu, pengguna juga dapat menangkap audio dari perangkat input seperti mikrofon, mengeksport gelombang suara ke file terpisah, dan menggunakan aset suara yang telah diimpor sebelumnya.

Yang menarik, Runtime Audio Importer tidak membutuhkan pustaka statis atau dependensi eksternal tertentu. Fitur ini mendukung semua perangkat yang tersedia seperti Android, iOS, Windows, Mac, Linux, dan sebagainya.

3.7.2 Emergence

Emergence adalah protokol Web3 yang ditujukan untuk pengembang game, yang terintegrasi dengan mesin permainan seperti Unreal Engine dan Unity melalui SDK berbasis Web3. Emergence memberdayakan pengembang game dengan otentikasi dompet, smart contract, sistem avatar, dan layanan inventaris NFT. Alat yang mudah digunakan dan kreatif memungkinkan pemain untuk masuk dengan mudah menggunakan dompet kripto mereka dan bermain dengan persona pilihan mereka secara transparan. Sementara itu, Emergence memberikan alat kepada pengembang game untuk memanfaatkan item NFT pengguna dan menyematkannya ke dalam permainan mereka secara mulus.

Protokol Emergence menyediakan alat yang diperlukan agar para pencipta dapat membangun dunia-dunia yang interoperabel, di mana pengguna dapat membuat persona berbasis rantai blok mereka sendiri dan memuat avatar, inventaris, dan data ke dalam dunia metaverse pilihan mereka.

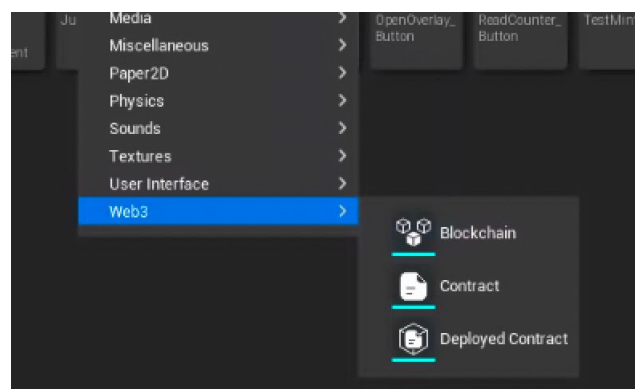
Dengan menyediakan alat seperti Layanan Inventaris, para pencipta dapat melihat potensi NFT dari segi utilitas. Bukan hanya dalam desain aslinya, tetapi juga dengan mempertimbangkan penggunaan NFT sebagai GameObjects aktual dalam dunia virtual mereka. Pengembang dapat menggunakan Layanan Inventaris Emergence untuk menyimpan metadata dinamis ke dalam NFT pengguna mereka. Hal ini membuka peluang besar untuk memikirkan tentang gaming berbasis Web3 yang baru dan apa yang dapat dicapai oleh NFT, serta bagaimana NFT tersebut berkembang melalui pengalaman metaverse. NFT lebih dari sekadar PFP (profil foto). Mereka dapat dianggap sebagai objek permainan yang persisten dan interoperabel, membawa inovasi baru dalam penceritaan, interaksi, pengalaman, dan sebagainya.

SDK Emergence menyediakan akses kode tingkat rendah dan tingkat tinggi untuk berinteraksi dengan jaringan EVM (Ethereum Virtual Machine) dan dompet. Mulai dari mengotentikasi pengguna dengan tanda tangan digital yang memungkinkan bentuk otentikasi universal baru, hingga membaca dan menulis ke smart contract. Dengan menyediakan alat seperti Sistem Avatar dan Layanan Inventaris, Emergence memberikan serangkaian fungsi agar pencipta metaverse apa pun dapat dengan mudah memanfaatkan kekuatan Web3.

3.8 Integrate Smart Contract with Unreal Engine 5

Proses integrasi smart contract dengan Unreal Engine 5 ini membutuhkan pengetahuan tentang blueprints. Proses ini dimulai dengan menyiapkan kembali proyek Unreal Engine 5 Dengan menggunakan plugin Emergence kita dapat menentukan contract dan juga menghubungkan blueprints dengan contracts yang sudah dideploy.

Plugin ini memiliki asset-asset yang akan membantu seperti asset contract dan deployed contract.



Gambar 3.12: Tampilan Asset yang disediakan Emergence

Emergence contract asset digunakan oleh berbagai metode dan objek sebagai deskriptor antarmuka pemrograman suatu kontrak. Sebagian besar datanya berasal dari ABI (Application Binary Interface) kontrak tersebut.

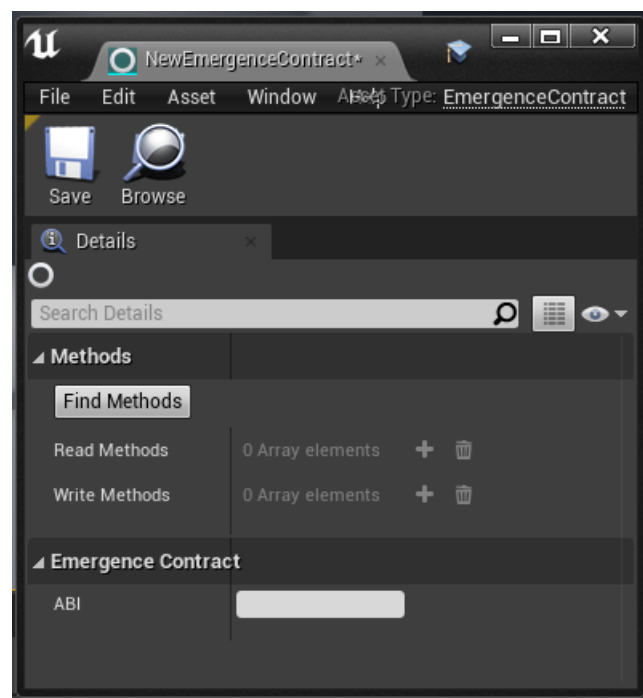
Aset Kontrak Emergence merujuk pada komponen yang digunakan oleh berbagai metode

dan objek dalam ekosistem Emergence untuk menjelaskan antarmuka pemrograman suatu kontrak. Aset ini berfungsi sebagai deskriptor yang memberikan informasi penting tentang struktur kontrak, fungsi, dan data yang terkait. Sebagian besar data yang terkait dengan aset ini berasal dari ABI kontrak, yang mendefinisikan bagaimana kontrak dapat diinteraksikan dan diakses oleh entitas eksternal.

Aset Kontrak Emergence memainkan peran penting dalam memfasilitasi integrasi dan interaksi yang lancar antara berbagai komponen dalam ekosistem Emergence. Dengan memanfaatkan ABI kontrak, pengembang dapat berkomunikasi dan berinteraksi dengan kontrak pintar secara efektif, memungkinkan mereka untuk memanggil fungsi spesifik, mengambil data, dan melakukan operasi lain yang didefinisikan dalam antarmuka kontrak.

Data yang disimpan dalam Aset Kontrak Emergence mencakup informasi seperti tanda tangan fungsi kontrak, parameter masukan dan keluaran, definisi acara, dan metadata lain yang diperlukan untuk berinteraksi dengan kontrak. Data ini memungkinkan komponen lain dalam ekosistem Emergence, seperti kontrak pintar, aplikasi, atau antarmuka pengguna, untuk memahami dan memanfaatkan fungsionalitas yang ditawarkan oleh kontrak.

Secara keseluruhan, Aset Kontrak Emergence berfungsi sebagai deskriptor yang menggambarkan antarmuka pemrograman suatu kontrak dalam ekosistem Emergence. Mereka memungkinkan integrasi, komunikasi, dan interaksi yang lancar dengan kontrak pintar dengan menyediakan informasi penting yang berasal dari ABI kontrak. Dengan memanfaatkan aset ini, pengembang dapat efektif menggunakan fungsionalitas yang ditawarkan oleh kontrak dan membangun aplikasi terdesentralisasi dan layanan inovatif dalam ekosistem Emergence.



Gambar 3.13: Tampilan Asset Kontrak Emergence

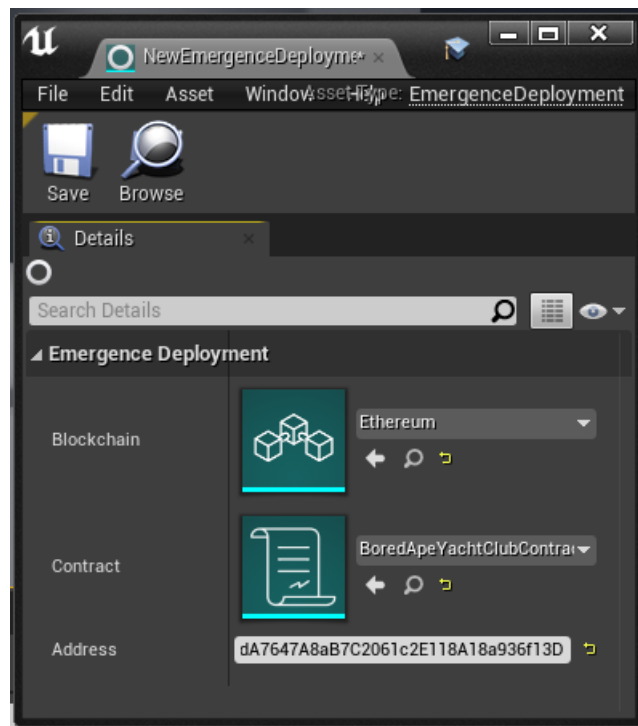
Aset Kontrak Emergence yang Dideploy digunakan oleh berbagai metode dan objek sebagai referensi ke kontrak tertentu yang telah dideploy di dalam blockchain. Aset ini terdiri dari aset Blockchain, aset Kontrak, dan alamat kontrak.

Aset Kontrak Emergence yang Dideploy adalah komponen yang digunakan dalam ekosistem Emergence untuk mengidentifikasi dan berinteraksi dengan kontrak pintar yang telah dideploy di dalam blockchain. Aset ini mencakup informasi yang penting untuk mengarahkan aplikasi atau komponen lain ke kontrak yang tepat untuk berinteraksi dengannya.

Komponen utama dari Aset Kontrak Emergence yang Dideploy adalah aset Blockchain yang merujuk pada blockchain tempat kontrak dideploy, aset Kontrak yang menggambarkan kontrak pintar secara keseluruhan, dan alamat kontrak yang merupakan alamat unik yang menunjukkan lokasi kontrak di dalam blockchain.

Dengan menggunakan Aset Kontrak Emergence yang Dideploy, pengembang dapat dengan mudah mengakses kontrak pintar yang telah dideploy di dalam blockchain. Mereka dapat memanggil fungsi kontrak, membaca dan menulis data, serta berinteraksi dengan kontrak untuk menjalankan logika bisnis yang diinginkan.

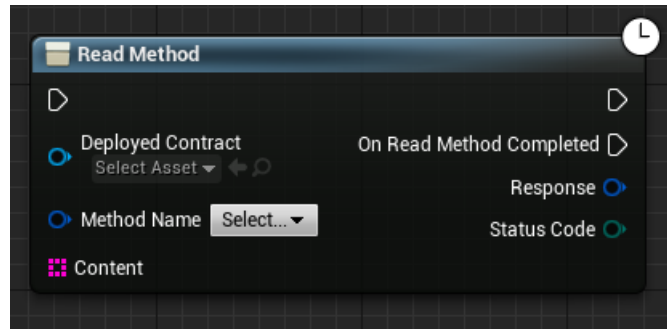
Secara keseluruhan, Aset Kontrak Emergence yang Dideploy berfungsi sebagai referensi yang mengarahkan aplikasi atau komponen lain ke kontrak pintar yang telah dideploy di dalam blockchain. Dengan menggunakan informasi yang terkandung dalam aset ini, pengembang dapat secara efisien berinteraksi dengan kontrak pintar dan memanfaatkan fungsionalitas yang ditawarkan oleh kontrak tersebut dalam ekosistem Emergence.



Gambar 3.14: Tampilan Deployed Contracts dari Emergence

Dari asset-asset tersebut akan diread oleh blueprint component dari emergence.

3.8.1 Read Method



Gambar 3.15: Tampilan Blueprint Read Method

Read method ini memiliki kegunaan untuk memanggil read method pada smart contract yang diberikan. Disini diberikan input berupa smart contract yang dideploy, nama method, dan content yaitu parameter yang akan diberikan pada method smart contract.

Kemudian dari method tersebut akan mendecode NFT yang didapat dan mendapatkan format metadata. Di metadata tersebut akan terdapat field `audio_url`. Field ini mengandung link url ke asset audio. Audio ini kemudian dapat diproses dan diplay dengan blueprints pada Unreal Engine 5.

[Halaman ini sengaja dikosongkan]

BAB IV

HASIL DAN PEMBAHASAN

Pada penelitian ini dipaparkan hasil pengujian berupa implementasi Unreal Engine 5 yang dapat menjalankan sistem dari *Smart Contract Sharing Data*, serta melakukan pengujian terkait dengan data audio yang di-load selama proses sistem berjalan hingga ke tahap pemutaran audio.

4.1 Skenario Pengujian

Pengujian dilakukan dengan melakukan langkah-langkah yang telah ditulis pada bab sebelumnya. Pengujian dimulai dengan melakukan testing method-method yang ada di smart contract. Fungsi-fungsi yang akan dicoba adalah fungsi yang dibuat sendiri untuk mempermudah proses integrasi. Pengujian dilakukan dengan melibatkan 2 buah akun yang sudah dibuat sebelumnya menggunakan metamask. Kemudian untuk method-method tersebut juga akan diuji performa dari penggunaan gas dan ETH-nya. Lalu akan dilakukan pengujian integrasi dari Unreal Engine 5. Pengujian integrasi meliputi berhasil tidaknya melakukan pemutaran audio yang sudah di-mint sebelumnya.

4.1.1 Hasil Pembuatan Wallet Account

Pembuatan wallet account dilakukan dengan Metamask. Pada platform metamask dapat dilakukan pembuatan akun dengan menginstall extension browser-nya. Setelah terbuat akunnya maka perlu digunakan faucet untuk mengisi ETH. ETH yang didapat adalah sejumlah 13 GoerliETH. ETH ini sudah lebih dari cukup untuk digunakan. Kemudian dibuat juga akun kedua dan diisi dengan GoerliETH. Akun kedua juga dapat digunakan untuk pengujian.

4.1.2 Hasil dan Pengujian Method Smart Contract

Smart Contract yang dibuat dapat digunakan untuk mint NFT. Deployment dilakukan dengan Environment Injected Provider Metamask. Metamask yang digunakan adalah akun yang sudah dibuat sebelumnya. Contract yang dibuat dideploy sudah digunakan untuk mint NFT. Untuk pengujian method mint, dilakukan dulu proses deploy dengan menggunakan remix IDE. Contract dideploy di Ganache untuk melakukan pengujian secara lokal karena apabila dilakukan deployment di Metamask maka ETH yang digunakan akan sulit untuk didapatkan kembali. Berikut ini adalah tabel dari hasil pengujian method mint.

Tabel 4.1: Hasil Pengujian Method mint

URI	Keterangan
HTTP URL yang berisi JSON	Sukses
HTTP URL random	Sukses
String random	Sukses
String kosong	Gagal

Pada bagian string kosong akan gagal karena diberikan guard sehingga jika tidak diberikan input maka akan terjadi error. Sementara untuk string random seperti string yang bukan HTTP

URL tidak gagal karena tidak diberikan guard ataupun kondisional yang melakukan pengecekan terhadap data yang diberikan.

Kemudian untuk bagian method `getAllTokens` maka akan dilakukan pengujian apakah jumlah id yang dikembalikan sesuai dengan jumlah URI yang sudah di mint.

Kegunaan dari method `getAllTokens` adalah untuk melakukan query jumlah uri yang ada di blockchain. Maka akan dilakukan pemanggilan mint terlebih dahulu sebelum melakukan pengujian `getAllTokens`.

Tabel 4.2: Hasil Pengujian Method `getAllTokens`

Jumlah Pemanggilan Mint	Panjang Nilai Balikan <code>getAllTokens</code>
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8

Dari hasil pengujian dapat dilihat hasilnya adalah sesuai ekspektasi. Ketika method mint dipanggil sejumlah 1 kali, maka panjang nilai yang dikembalikan oleh method `getAllTokens` adalah 1. Ketika method mint dipanggil sejumlah 2 kali, maka panjang nilai yang dikembalikan oleh method `getAllTokens` adalah 2. Ketika method mint dipanggil sejumlah 3 kali, maka panjang nilai yang dikembalikan oleh method `getAllTokens` adalah 3. Ketika method mint dipanggil sejumlah 4 kali, maka panjang nilai yang dikembalikan oleh method `getAllTokens` adalah 4. Ketika method mint dipanggil sejumlah 5 kali, maka panjang nilai yang dikembalikan oleh method `getAllTokens` adalah 5. Ketika method mint dipanggil sejumlah 6 kali, maka panjang nilai yang dikembalikan oleh method `getAllTokens` adalah 6. Ketika method mint dipanggil sejumlah 7 kali, maka panjang nilai yang dikembalikan oleh method `getAllTokens` adalah 7. Ketika method mint dipanggil sejumlah 8 kali, maka panjang nilai yang dikembalikan oleh method `getAllTokens` adalah 8.

Kemudian untuk pengujian method selanjutnya adalah pengujian method `getAllTokensURI`. Untuk method ini akan mengembalikan balikan array yang berisi URI yang sudah disimpan dengan method mint yang sudah pernah dipanggil sebelumnya. Hasil yang diharapkan adalah dengan memanggil method ini maka URI-URI yang sudah pernah di mint sebelumnya akan dikembalikan dalam bentuk array.

Tabel 4.3: Hasil Pengujian Method `getAllTokensURI`

Mint URI	Balikan
<code>https://bafybeu.ipfs.w3s.link/nft-1.json</code>	<code>[https://bafybeu.ipfs.w3s.link/nft-1.json]</code>
<code>https://bafybeu.ipfs.w3s.link/nft-2.json</code>	<code>[https://bafybeu.ipfs.w3s.link/nft-2.json]</code>

https://bafybeu.ipfs.w3s.link/nft-1.json, https://bafybeu.ipfs.w3s.link/nft-2.json Tidak pernah memanggil method Mint	[https://bafybeu.ipfs.w3s.link/nft-1.json, https://bafybeu.ipfs.w3s.link/nft-2.json] []
https://bafybeu.ipfs.w3s.link/nft-1.json, https://bafybeu.ipfs.w3s.link/nft-2.json, https://bafybeu.ipfs.w3s.link/nft-3.json	[https://bafybeu.ipfs.w3s.link/nft-1.json, https://bafybeu.ipfs.w3s.link/nft-2.json, https://bafybeu.ipfs.w3s.link/nft-3.json]

Dari pemanggilan method-method `getAllTokensURI` hasilnya adalah sesuai dengan yang diharapkan. Apabila dilakukan mint sebanyak 2 kali maka akan dikembalikan juga sebanyak 2 dan berisikan uri-uri yang sudah pernah di mint sebelumnya. Apabila dilakukan mint sebanyak 3 kali maka akan dikembalikan juga sebanyak 3 dan berisikan uri-uri yang sudah pernah di mint sebelumnya. Apabila dilakukan mint sebanyak 4 kali maka akan dikembalikan juga sebanyak 4 dan berisikan uri-uri yang sudah pernah di mint sebelumnya. Apabila dilakukan mint sebanyak 5 kali maka akan dikembalikan juga sebanyak 5 dan berisikan uri-uri yang sudah pernah di mint sebelumnya. Method ini akan berguna untuk melakukan pemanggilan di Unreal Engine 5. Apabila fungsi ini dipanggil oleh Unreal Engine 5 blueprint maka akan dibaca berupa array URI.

Kemudian untuk pengujian method `tokenURI` akan dilakukan dengan memanggil method mint dan kemudian menggunakan indexnya untuk memanggil method `tokenURI` ini. Method ini akan menggunakan parameter `tokenId` yang mana adalah urutan dilakukannya mint. Index ini bersifat incremental yang artinya URI pertama memiliki index 0, URI kedua memiliki index 1, URI ketiga memiliki index 3, URI keempat memiliki index 4, dan URI kelima memiliki index 5, dan seterusnya.

Berikut adalah tabel hasil pengujian method `tokenURI`

Tabel 4.4: Hasil Pengujian Method `tokenURI`

Mint-ke	URI	Parameter method tokenURI	Balikan method tokenURI
1	https://bafybeu.ipfs.w3s.link/nft-1.json	0	https://bafybeu.ipfs.w3s.link/nft-1.json
2	https://bafybeu.ipfs.w3s.link/nft-1.json	1	https://bafybeu.ipfs.w3s.link/nft-1.json
3	https://bafybeu.ipfs.w3s.link/nft-1.json	2	https://bafybeu.ipfs.w3s.link/nft-1.json
4	https://bafybeu.ipfs.w3s.link/nft-1.json	3	https://bafybeu.ipfs.w3s.link/nft-1.json
5	https://bafybeu.ipfs.w3s.link/nft-1.json	4	https://bafybeu.ipfs.w3s.link/nft-1.json

6	https://bafybeu.ipfs.w3s.link/nft-1.json	5	https://bafybeu.ipfs.w3s.link/nft-1.json
7	https://bafybeu.ipfs.w3s.link/nft-1.json	6	https://bafybeu.ipfs.w3s.link/nft-1.json
8	https://bafybeu.ipfs.w3s.link/nft-1.json	7	https://bafybeu.ipfs.w3s.link/nft-1.json
9	https://bafybeu.ipfs.w3s.link/nft-1.json	8	https://bafybeu.ipfs.w3s.link/nft-1.json
10	https://bafybeu.ipfs.w3s.link/nft-1.json	9	https://bafybeu.ipfs.w3s.link/nft-1.json
11	https://bafybeu.ipfs.w3s.link/nft-1.json	10	https://bafybeu.ipfs.w3s.link/nft-1.json
12	https://bafybeu.ipfs.w3s.link/nft-1.json	11	https://bafybeu.ipfs.w3s.link/nft-1.json
13	https://bafybeu.ipfs.w3s.link/nft-1.json	12	https://bafybeu.ipfs.w3s.link/nft-1.json
14	https://bafybeu.ipfs.w3s.link/nft-1.json	13	https://bafybeu.ipfs.w3s.link/nft-1.json
15	https://bafybeu.ipfs.w3s.link/nft-1.json	14	https://bafybeu.ipfs.w3s.link/nft-1.json
0	https://bafybeu.ipfs.w3s.link/nft-1.json	100	⌋ERROR⌋ Invalid token ID

Pada kondisi normal pemanggilan method tokenURI sesuai dengan yang dikehendaki. Pemanggilan tokenURI akan mengembalikan uri dengan index yang diberikan. Namun jika token ID yang diberikan tidak ditemukan di blockchain, maka akan terjadi error Invalid token ID. Hal ini merupakan *behavior* yang dikehendaki karena memang jika tidak ditemukan URI seharusnya dihasilkan error. Jika tidak diberikan error maka bisa terjadi *undefined behavior* pada saat digunakan di Unreal Engine 5.

Dalam konteks integrasi dengan Unreal Engine 5, jika URI tidak ditemukan atau mengembalikan hasil yang tidak valid, dapat mempengaruhi pengalaman pengguna atau proses pemrosesan data yang menggunakan URI tersebut. Oleh karena itu, dengan memberikan error "Invalid token ID" saat tidak ditemukannya URI yang sesuai, pengembang dapat dengan mudah mendebug dan menangani kasus tersebut secara programatik.

Dalam pengembangan dengan Unreal Engine 5, penting bagi pengembang untuk memahami dan menangani situasi ketika URI tidak tersedia atau tidak valid. Hal ini membantu mencegah kesalahan dalam penggunaan data dan memastikan keandalan aplikasi yang diintegrasikan dengan smart contract.

Dengan adanya error "Invalid token ID" sebagai respons saat token URI tidak ditemukan, pengembang dapat melakukan penanganan yang tepat dalam aplikasi mereka, seperti memberikan pesan kesalahan yang informatif kepada pengguna, mengambil tindakan alternatif, atau mengambil langkah pemulihan yang sesuai.

Pada pengujian selanjutnya adalah pengujian gas pada setiap method-method diatas. Pengujian ini bertujuan untuk mengukur efisiensi smart contract yang digunakan pada penelitian ini.

4.1.3 Hasil Pengujian Estimasi Gas

Pengujian ini menyajikan hasil total keseluruhan biaya gas yang diperlukan dalam melakukan proses transaksi pengiriman nominal Ether dari pengirim ke penerima. Pengujian ini dilakukan dengan cara memanggil masing-masing method yang akan diujikan lalu mencatat nilai konsumsi gasnya. Dengan menggunakan *Ganache* dan juga dilakukan pada network *Goerli* dan juga *Sepolia* yang menerapkan konsensus Proof of Work.

CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE	SAVE	SWITCH	⚙
54	2000000000	6721975	MERGE	5777	HTTP://127.0.0.1:7545	AUTOMINING	QUICKSTART			
TX HASH										
0x190da20af55d61f7c65ea00686384b4726f5cade52e25ec56681f9f74a891653										
CONTRACT CALL										
FROM ADDRESS										
0x0cf5d59795781e7f1756e818a156e5098b42c16e										
TO CONTRACT ADDRESS										
0x7c46ba1f4de683f66766546b6d113a84f47c6092										
GAS USED										
125556										
VALUE										
0										
TX HASH										
0x46de1581a37eb12fbbcd71037d2b5e2e89d7db81bf13a8f9bc45096cf1fe4620										
CONTRACT CREATION										
FROM ADDRESS										
0x0cf5d59795781e7f1756e818a156e5098b42c16e										
CREATED CONTRACT ADDRESS										
0x7c46ba1f4de683f66766546b6d113a84f47c6092										
GAS USED										
3274368										
VALUE										
0										
TX HASH										
0x16d19c8ab308250e313e06420ddd9257fd649d9534294ca8e0964a6bd688400f										
CONTRACT CALL										
FROM ADDRESS										
0x5a5f4cd0b0a70197447568cf20489d21d794ec17										
TO CONTRACT ADDRESS										
0x973Ae41360f339d37a16C989991bAcc2f95b3aB9										
GAS USED										
170992										
VALUE										
0										
TX HASH										
0x7963f2bbbf0b724f1de745514ddb0055858941833c02e80bff656b775e97f40e										
CONTRACT CALL										
FROM ADDRESS										
0x5a5f4cd0b0a70197447568cf20489d21d794ec17										
TO CONTRACT ADDRESS										
0x973Ae41360f339d37a16C989991bAcc2f95b3aB9										
GAS USED										
170992										
VALUE										
0										
TX HASH										
0xaaaf1613a9e4b4395f05fb2ee898dfedbcc96cb53d799f5ab25457839b4ab74f5										
CONTRACT CALL										

Gambar 4.1: Transaksi blockchain di Ganache

Tujuan dari pengujian ini adalah untuk mengetahui berapa besar biaya gas yang dibutuhkan untuk menjalankan proses sistem pada Smart Contract yang sudah dibuat. Proses dari pengujian ini sama seperti dengan Ganache network, hanya saja setiap proses sistem yang berjalan, lamanya waktu terkonfirmasi pada Metamask ditemukan variasi, yang mana sangat berbeda dengan Ganache yang tidak memiliki jeda. Untuk method-method yang bersifat read only atau tidak payable dan tidak melakukan write operation tidak dilakukan uji gas karena method-method tersebut tidak menggunakan gas untuk dieksekusi.

Dengan melakukan pengujian ini, pengembang dapat memperoleh pemahaman yang lebih baik tentang biaya gas yang terkait dengan menjalankan proses sistem pada Smart Contract. Informasi ini dapat digunakan untuk mengoptimalkan dan memperbaiki efisiensi penggunaan gas, serta memastikan bahwa Smart Contract berjalan secara optimal dengan biaya yang sesuai dan meminimalkan risiko transaksi yang tidak memadai.

1. Deployment Gas

Pengujian estimasi gas dimulai dari jaringan Ganache. Tabel dibawah ini menunjukkan estimasi gas yang dibutuhkan untuk melakukan proses deploy Smart Contract pada Ganache

Tabel 4.5: Hasil Pengujian Deploy Smart Contract

<i>Blockchain Network</i>	<i>Gas Used</i>	<i>Gas Price (Gwei)</i>
Ganache	3274368	2.772126468
Goerli	3275354	2.500000024
Sepolia	3275354	2.500000012

Tabel 4.6: Hasil Pengujian Deploy Smart Contract di *Ganache*

<i>Percobaan ke</i>	<i>Gas Used</i>	<i>Gas Price (Gwei)</i>
1	3274368	2.769504978
2	3274368	2.77037599
3	3274368	2.760107065
4	3274368	2.760947704
5	3274368	2.761791060
6	3274368	2.762637142
7	3274368	2.763485958
8	3274368	2.764337517
9	3274368	2.758433902
10	3274368	2.759269134
11	3274368	2.760107034
12	3274368	2.77037569
13	3274368	2.760108232
14	3274368	2.761792302
15	3274368	2.762633213

Dari hasil tabel yang didapatkan diatas, didapati bahwa penggunaan gas cukup besar karena proses melakukan deploy memang membutuhkan resource yang tidak sedikit. Gas Used pada jaringan lokal Ganache juga lebih besar daripada di jaringan Goerli dan Sepolia. Pada tabel 4.6 juga dilakukan 10 kali percobaan yang mana hasilnya dapat disimpulkan bahwa gas price di jaringan lokal ganache lebih besar daripada Goerli dan Sepolia pada saat pengujian proses deployment smart contract.

2. Minting Gas

Pengujian efisiensi gas selanjutnya adalah pengujian saat melakukan minting token atau proses pembuatan NFT untuk menyimpan data URI. Proses ini sendiri memanggil method `mint` dan karena method ini bersifat `write` dan `payable` maka penggunaan gas nya pun akan lebih banyak dibandingkan method yang bersifat `write`. Pengujian ini akan dilakukan dengan jaringan lokal ganache dan juga jaringan Goerli dan Sepolia. Berikut ini adalah table hasil pengujian efisiensi gas untuk pemanggilan method `mint` pada smart contract.

Pengujian efisiensi gas ini penting untuk memahami dan mengoptimalkan biaya yang dikeluarkan dalam proses pembuatan NFT dan penyimpanan data URI. Dengan mengetahui jumlah gas yang diperlukan, pengembang dapat mengambil langkah-langkah untuk meningkatkan efisiensi penggunaan gas dan meminimalkan biaya yang terkait dengan operasi-operasi tersebut pada smart contract.

Tabel 4.7: Hasil Pengujian Gas Mint Smart Contract

<i>Blockchain Network</i>	<i>Gas Used</i>	<i>Gas Price (Gwei)</i>
Ganache	170992	2.526306790
Goerli	125580	2.500000016
Sepolia	125568	2.500000016

Tabel 4.8: Hasil Pengujian Gas Mint di *Ganache*

<i>Percobaan ke</i>	<i>Gas Used</i>	<i>Gas Price (Gwei)</i>
1	170992	2.526306790
2	170992	2.529847969
3	170992	2.533865829
4	170992	2.538424537
5	170992	2.543596897
6	170992	2.549465512
7	170992	2.556124107
8	170992	2.764337517
9	170992	2.758433902
10	170992	2.759269134
11	170992	2.526303233
12	170992	2.529832132
13	170992	2.533847239
14	170992	2.538491925
15	170992	2.543581052

Dari tabel 4.8 diatas dapat diketahui bahwasanya Gas Used pada method mint cukup besar untuk standard blockchain smart contract. Kemudian untuk pemakaian gas yang dilakukan di jaringan lokal ganache lebih besar daripada di jaringan Goerli maupun Sepolia. Hal ini memperkuat hasil dari uji deployment yang dilakukan sebelumnya karena disana juga pemakaian gas yang ada di jaringan lokal Ganache. Gas Price ditentukan oleh tingkat penggunaan jaringan blockchain tersebut, sehingga semakin banyak pengguna yang mengakses jaringan dari suatu blockchain maka akan semakin banyak juga gas price yang akan dikonsumsi.

4.1.4 Hasil Pembuatan Metadata dengan IPFS

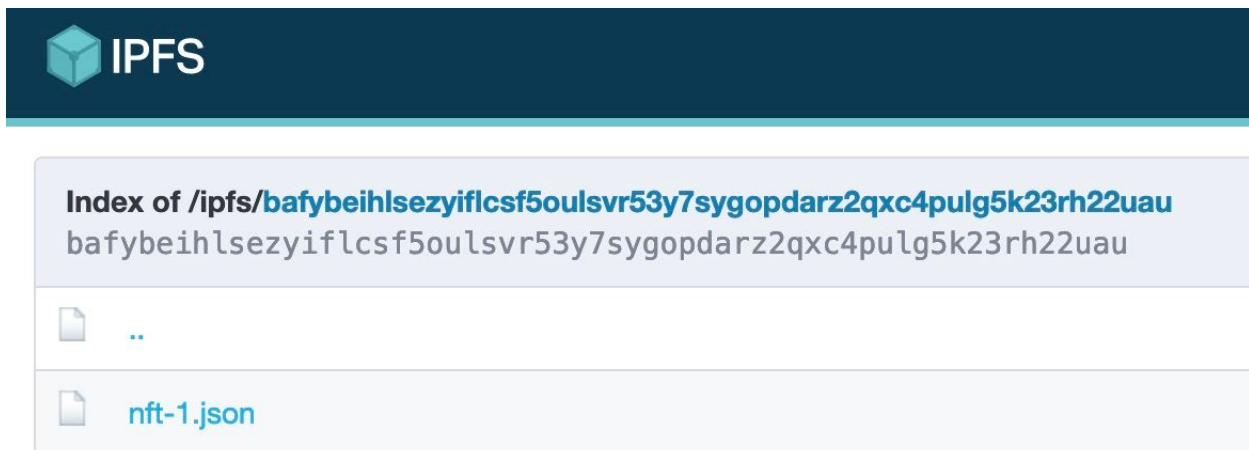
Pengujian untuk IPFS dilakukan dengan cara melakukan pencocokan metadata dengan OpenSea. Metadata yang distandarkan oleh OpenSea akan digunakan dalam sistem.

Dalam pengembangan aplikasi dengan Unreal Engine 5, penting untuk memastikan bahwa metadata yang diunggah ke IPFS sesuai dengan standar yang diterima oleh OpenSea. Hal ini penting karena kompatibilitas dengan OpenSea memungkinkan NFT yang dibuat dengan Unreal Engine 5 dapat diperdagangkan secara interoperabel di platform tersebut.

Dengan melakukan pencocokan metadata dengan standar OpenSea, pengembang dapat memastikan bahwa NFT yang dihasilkan oleh Unreal Engine 5 dapat diakses dan dikenali dengan benar oleh platform perdagangan yang kompatibel. Hal ini memungkinkan NFT tersebut untuk mendapatkan eksposur lebih luas dan meningkatkan likuiditas dan nilai jualnya.

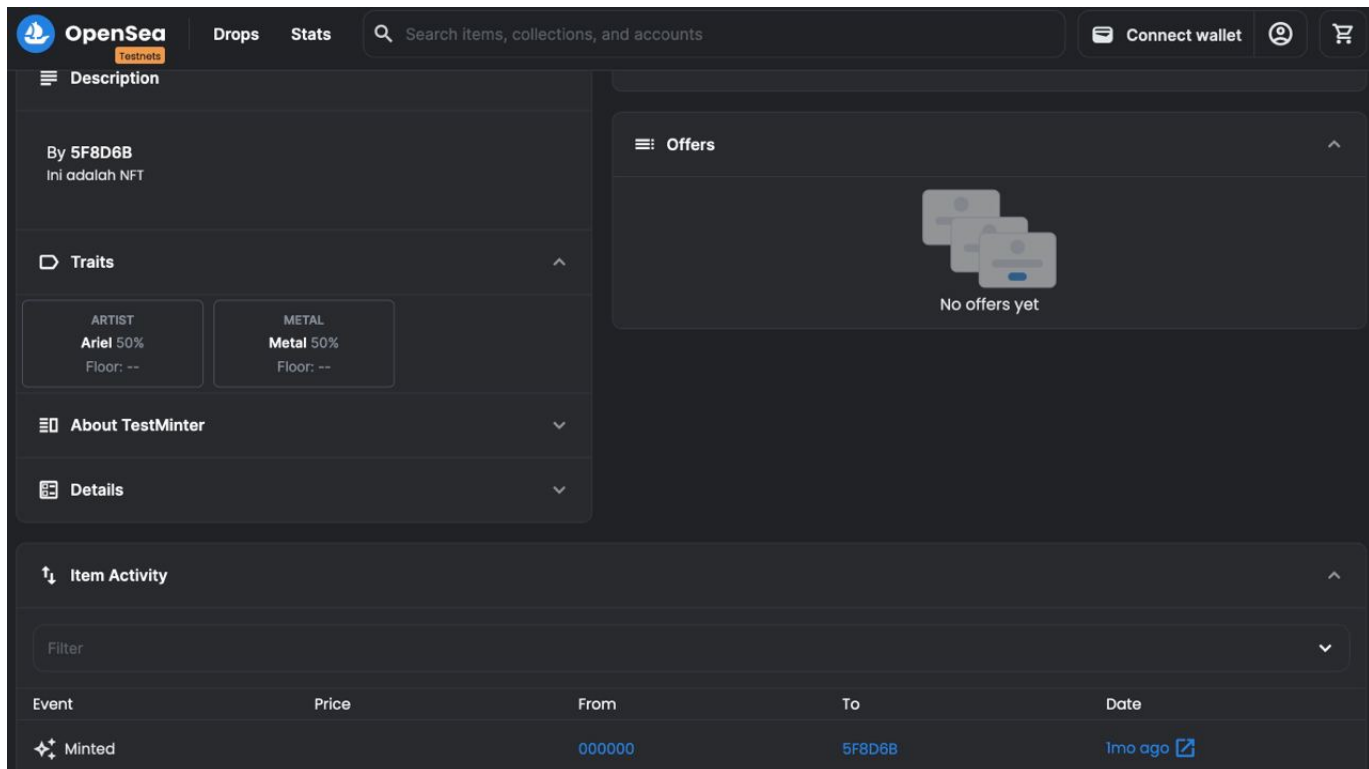
Dalam proses pengujian, metadata yang dihasilkan oleh Unreal Engine 5 akan dibandingkan dengan standar metadata yang ditetapkan oleh OpenSea untuk memastikan kesesuaian dan konsistensi. Jika metadata sesuai dengan standar OpenSea, maka NFT yang dibuat dengan Unreal Engine 5 dapat diintegrasikan dengan lancar dalam ekosistem OpenSea dan mendapatkan manfaat dari fitur-fitur yang disediakan oleh platform tersebut.

Metadata yang dibuat di IPFS sudah dapat diakses melalui gatewaynya. Contohnya untuk nft-1.json adalah sebagai berikut.



Gambar 4.2: Tampilan IPFS nft-1.json yang sudah diupload

Hasil dari pembacaan metadata pada OpenSea dengan jaringan testnet adalah dapat terbaca. Jika tidak digunakan standar metadata yang sudah menjadi konvensi maka tidak akan terbaca oleh OpenSea apa yang sudah dituliskan pada metadata JSON yang diberikan pada jaringan blockchain OpenSea dalam bentuk mint file. Kemudian untuk URI nya sendiri haruslah berupa URI valid, apabila tidak valid maka OpenSea juga tidak dapat membaca URI yang menyebabkan tidak dapat membaca metadata tersebut karena file metadata yang berupa JSON tersebut tersimpan di dalam URI yang diberikan.



Gambar 4.3: Tampilan OpenSea yang berhasil membaca metadata

Penelitian ini menggunakan metadata yang sudah distandarkan oleh OpenSea, namun tidak ada pengecekan untuk formatnya itu sendiri sehingga apabila dimasukkan format yang salah maka integrasi dengan Unreal Engine 5 akan gagal dan menyebabkan error yang detailnya akan dijelaskan pada pengujian-pengujian berikutnya.

Meskipun pengecekan format metadata tidak dilakukan dalam penelitian ini, hal tersebut merupakan langkah yang penting dalam pengembangan yang sebenarnya. Sebelum mengintegrasikan metadata dengan Unreal Engine 5, disarankan untuk memastikan bahwa metadata yang digunakan sesuai dengan format yang diharapkan untuk mencegah terjadinya kesalahan dan kegagalan integrasi yang dapat mempengaruhi fungsionalitas aplikasi.

4.1.5 Hasil Mint NFT

Pengujian ini berbeda dengan pengujian method mint yang ada pada pengujian-pengujian sebelumnya. Pada pengujian ini proses mint yang dimaksud adalah proses dimana dilakukan deployment smart contract ke jaringan blockchain Goerli. Nantinya jaringan ini akan digunakan untuk integrasi dengan Unreal Engine 5.

Deployment smart contract ke jaringan Goerli dilakukan dengan akun yang sudah berisikan cukup GoerliETH karena proses deployment juga membutuhkan Ether.

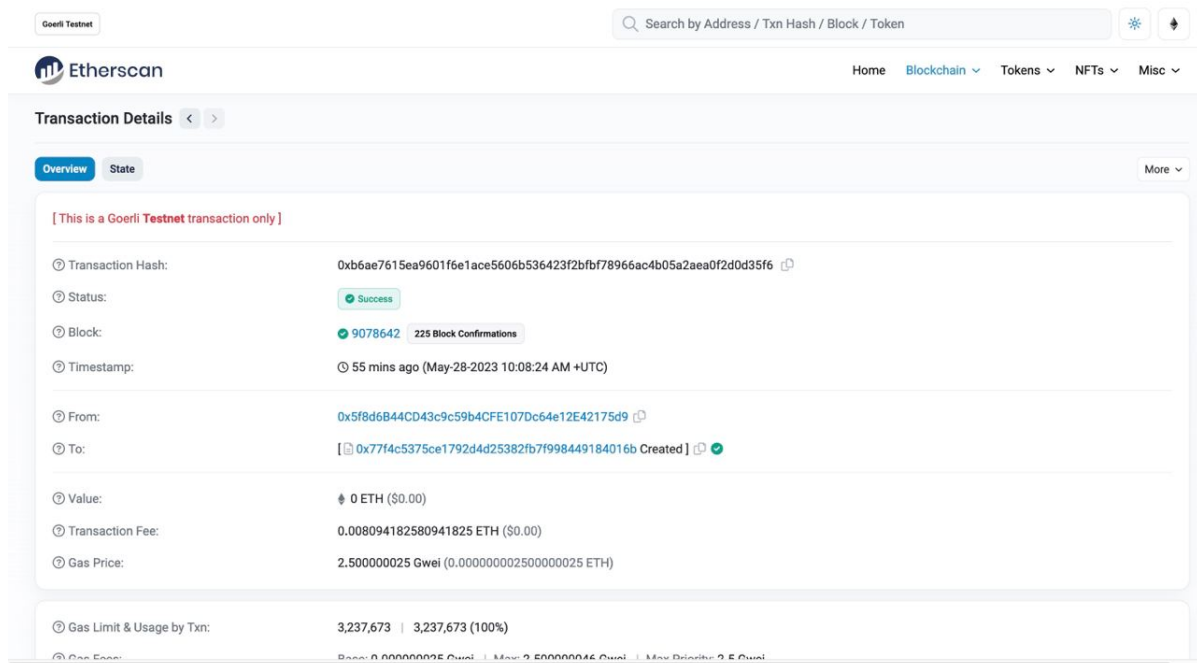
Tabel 4.9: Hasil Pengujian Deploy ke Goerli berdasarkan Ether

Jumlah GoerliETH	Keterangan
12.5	Sukses

0	Gagal
0.001	Gagal

Dari tabel 4.9 dapat disimpulkan bahwa diperlukan jumlah GoerliETH yang memadai untuk dilakukannya proses deployment smart contract. Minimum GoerliETH untuk dilakukan deployment adalah 0.01 untuk batas amannya. Namun perlu diperhatikan juga bahwa proses mint URI tetap membutuhkan GoerliETH juga sehingga perlu disiapkan GoerliETH yang cukup. GoerliETH didapat secara mining pada faucet-faucet yang tersedia di internet.

Hasil dari mint nft sudah dapat digunakan untuk integrasi Unreal Engine 5. ABI dapat dilihat pada compiler Remix IDE.



Gambar 4.4: Etherscan dari hasil transaction mint NFT

ABI yang dapat dilihat di etherscan dapat digunakan pada integrasi Unreal Engine 5. Hal ini dilakukan agar Unreal Engine 5 dapat membaca method-method apa saja yang terdapat pada smart contract. Kemudian diperlukan juga alamat contract yang sudah dideploy agar Unreal Engine 5 yang sudah mengetahui spesifikasi smart contract dapat melakukan read data yang sudah tersimpan dengan pemanggilan mint sebelumnya.

Dengan menggunakan ABI yang dapat dilihat di etherscan, Unreal Engine 5 dapat melakukan integrasi dengan smart contract yang sudah dideploy. ABI (Application Binary Interface) adalah deskripsi formal dari method dan variabel yang ada dalam smart contract. Dengan mengetahui ABI dari smart contract, Unreal Engine 5 dapat memahami dan berinteraksi dengan method-method yang terdapat dalam smart contract tersebut.

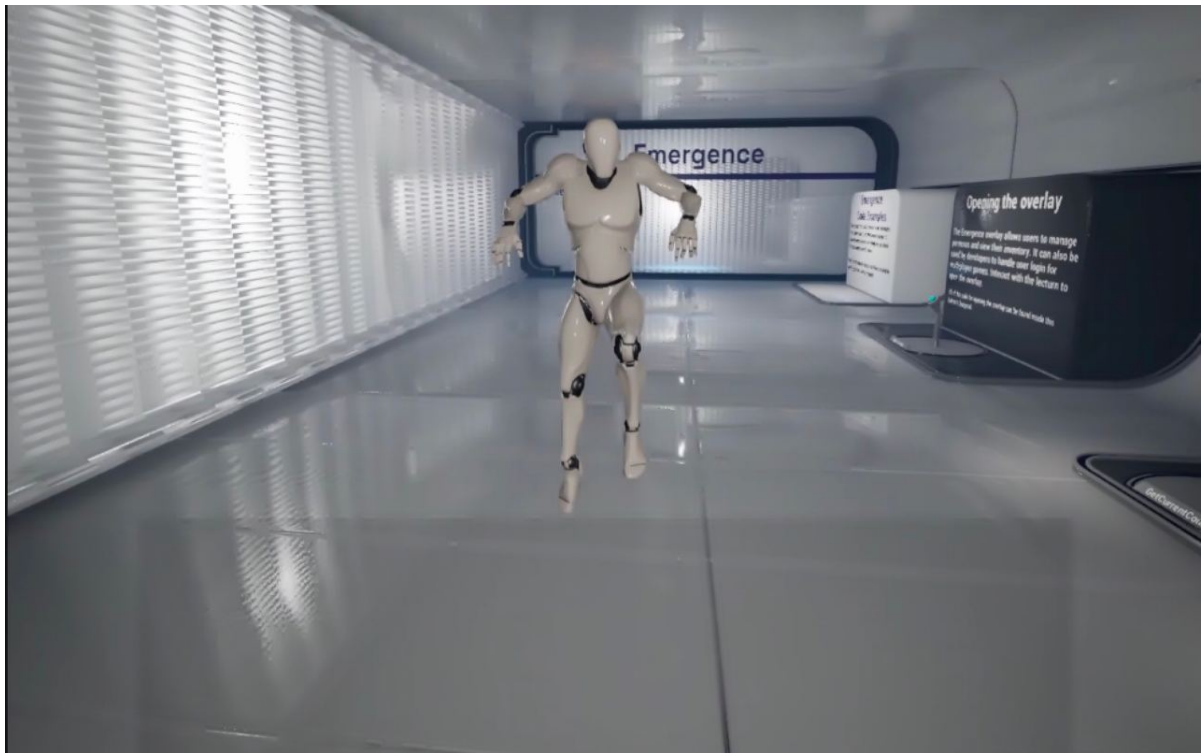
Selain itu, untuk dapat membaca data yang sudah tersimpan di smart contract, Unreal Engine 5 juga memerlukan alamat contract yang sudah dideploy di blockchain. Dengan mengetahui alamat contract, Unreal Engine 5 dapat melakukan pemanggilan ke smart contract tersebut dan mengakses data yang sudah tersimpan di dalamnya.

Proses ini memungkinkan Unreal Engine 5 untuk mengintegrasikan dan menggunakan data yang ada di dalam smart contract dalam pengembangan dan simulasi metaverse. Dengan memanggil method-method yang terdapat dalam smart contract, Unreal Engine 5 dapat membaca, memperbarui, atau menggunakan data yang ada di dalamnya sesuai dengan kebutuhan pengembangan aplikasi atau simulasi metaverse yang sedang dilakukan.

Dengan adanya integrasi ini, Unreal Engine 5 dapat memanfaatkan potensi penuh dari smart contract dalam membangun pengalaman metaverse yang lebih dinamis, interaktif, dan terkoneksi dengan teknologi blockchain.

4.1.6 Hasil Pembuatan Project Pada Unreal Engine 5

Project unreal engine 5 yang sudah dibuat dapat memberikan environment visual yang dapat melakukan visualisasi audio maupun game. Digunakan juga beberapa plugin yang dapat membantu proses integrasi.



Gambar 4.5: Tampilan project unreal engine 5

Avatar yang disediakan oleh Unreal Engine 5 dapat digunakan untuk melakukan simulasi Metaverse. Pada pengujian dilakukan operasi pergerakan avatar dan melakukan penginstalan plugin-plugin web3 yang dapat membantu proses integrasi. Selain itu, Unreal Engine 5 juga menyediakan avatar yang dapat digunakan untuk melakukan simulasi di dalam metaverse. Avatar tersebut memungkinkan pengguna untuk menciptakan representasi digital dari diri mereka sendiri atau karakter yang mereka pilih, dan berinteraksi dengan lingkungan dan pengguna lain di dalam metaverse.

Selama pengujian, dilakukan operasi pergerakan avatar untuk menguji kemampuan navigasi dan interaksi di dalam metaverse. Selain itu, pengujian juga melibatkan penginstalan plugin-plugin web3 yang dirancang khusus untuk Unreal Engine 5. Plugin-plugin ini mem-

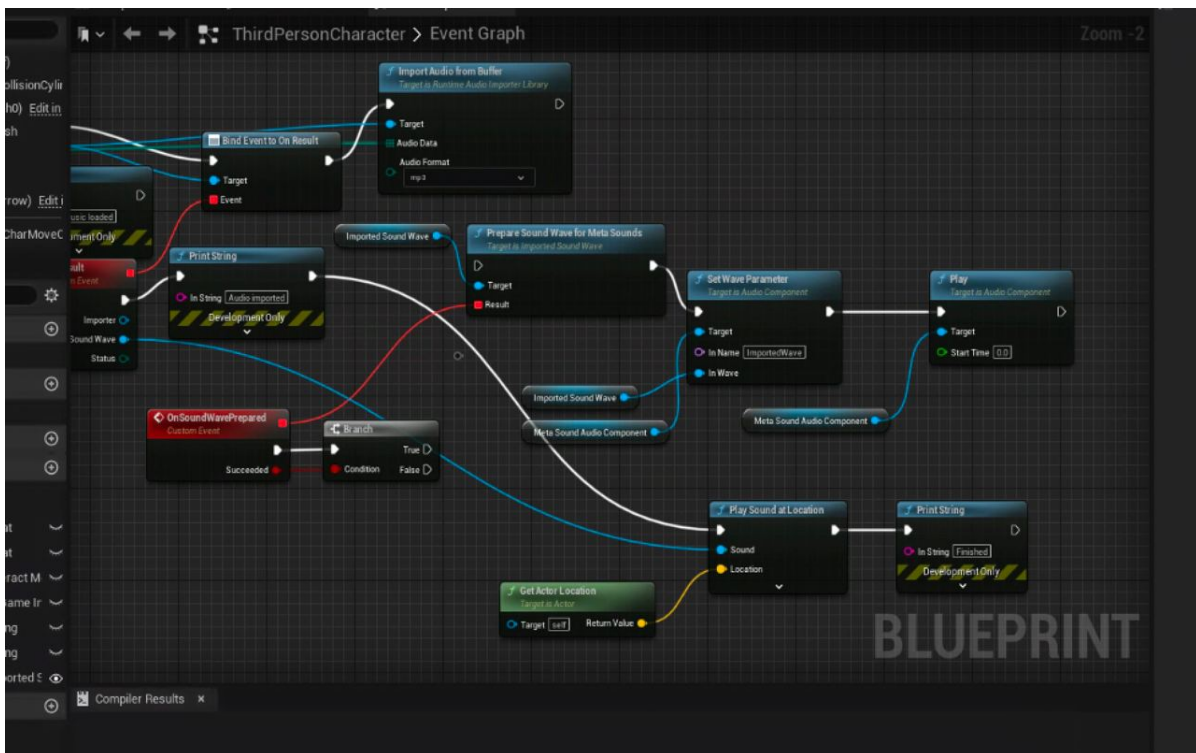
bantu proses integrasi dengan teknologi Web3, seperti blockchain dan smart contract, sehingga memungkinkan pengguna untuk terlibat dalam transaksi dan aktivitas berbasis kripto di dalam metaverse.

Dengan adanya fitur avatar dan integrasi dengan plugin-plugin web3, Unreal Engine 5 memberikan kesempatan bagi pengembang dan pengguna untuk merasakan pengalaman yang lebih mendalam dan interaktif di dalam metaverse. Pengguna dapat menjelajahi lingkungan yang dirancang dengan detail, berinteraksi dengan avatar lain, melakukan transaksi menggunakan cryptocurrency, dan terlibat dalam aktivitas-aktivitas kreatif dan sosial yang unik di dalam metaverse.

Dalam keseluruhan, Unreal Engine 5 memainkan peran penting dalam memfasilitasi simulasi metaverse dengan menyediakan fitur avatar yang dapat digunakan untuk mewakili pengguna dan integrasi dengan plugin-plugin web3. Hal ini memungkinkan pengembang dan pengguna untuk terlibat dalam pengalaman metaverse yang lebih imersif, dinamis, dan terkoneksi dengan teknologi blockchain.

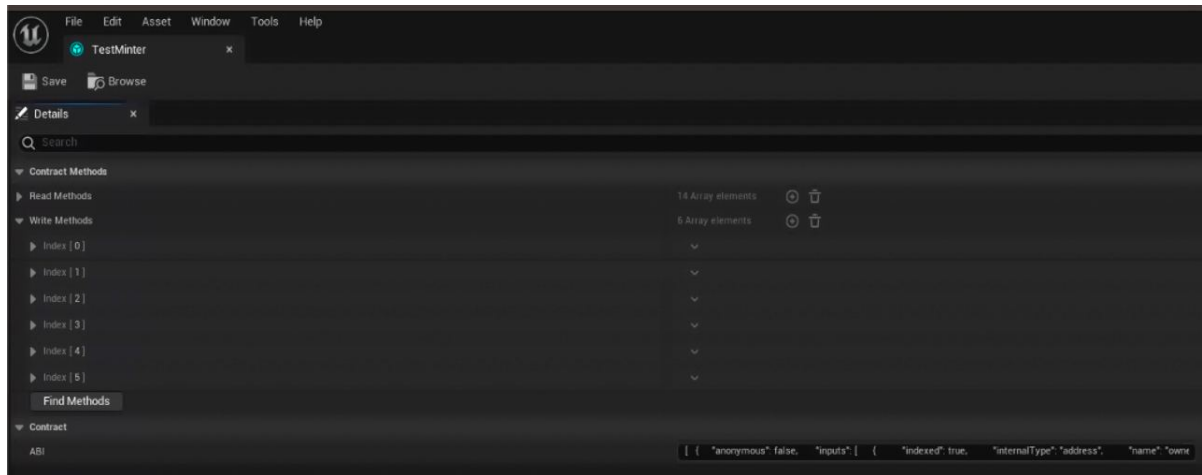
4.1.7 Hasil Integrasi Kontrak dengan Unreal Engine 5

Setelah diintegrasikan dengan blueprint maka di lingkungan virtual unreal engine 5 sudah dapat memutar audio yang didapatkan dari NFT. Blueprint yang digunakan adalah sebagai berikut.



Gambar 4.6: Tampilan blueprint final

Pengujian yang akan dilakukan adalah berhasil tidaknya integrasi dengan cara melakukan pemutaran audio dengan menggunakan metadata yang sudah di mint pada proses sebelumnya. Akan dilakukan beragam cara yakni menggunakan ABI yang tidak valid dan juga sebaliknya.



Gambar 4.7: Tampilan contract ABI di Unreal Engine 5

Tabel 4.10: Hasil Pengujian ABI Smart Contract di UE5

Contract ABI	Keterangan
Valid	Sukses
Invalid	Gagal

Dari tabel di atas, dapat disimpulkan bahwa keberhasilan integrasi tergantung pada validitas Contract ABI. Jika Contract ABI tidak valid, proses integrasi tidak dapat dilakukan karena langkah pertama yang harus dilakukan adalah menyediakan Contract ABI yang benar. Contract ABI digunakan sebagai deskripsi antarmuka dari smart contract yang akan diintegrasikan, dan kesalahan dalam Contract ABI dapat menghambat atau menyebabkan kegagalan integrasi dengan Unreal Engine 5. Oleh karena itu, penting untuk memastikan keakuratan dan kevalidan Contract ABI sebelum melanjutkan proses integrasi.

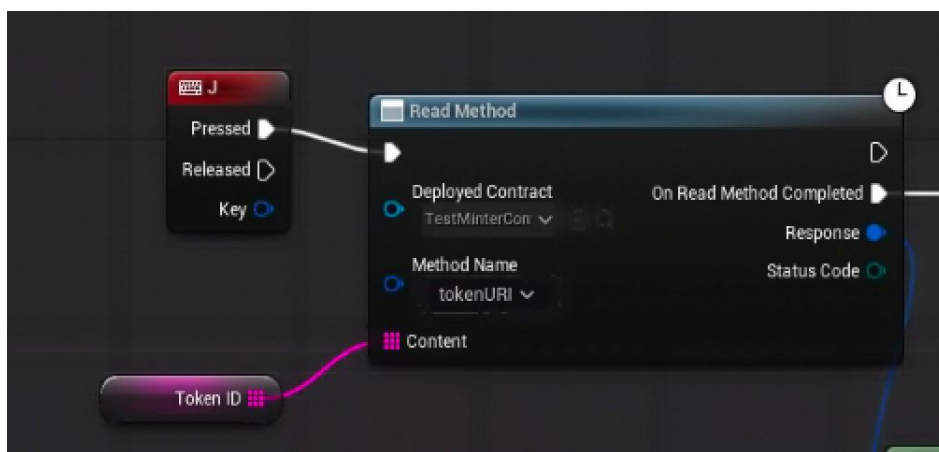
Selanjutnya adalah pengujian proses pemanggilan smart contract di Unreal Engine 5. Pada proses ini blueprint yang digunakan akan memanggil method-method read yang ada pada smart contract yang sudah dibaca dalam tahapan pemberian Contract ABI sebelumnya seperti `tokenURI`, `getAllTokens`, dan `getAllTokensURI`. Dengan memanggil method tersebut diharapkan dapat dimuat metadata JSON yang sudah di mint pada tahapan sebelumnya.

Tabel 4.11: Hasil Pengujian Pemanggilan method menggunakan Blueprint

Method	Alamat Contract	Sesuai Dengan ABI	Keterangan
tokenURI	Valid	true	Sukses
tokenURI	Invalid	true	Gagal
tokenURI	Valid	false	Gagal
tokenURI	Invalid	false	Gagal
getAllTokens	Valid	true	Sukses
getAllTokens	Invalid	true	Gagal
getAllTokens	Valid	false	Gagal
getAllTokens	Invalid	false	Gagal

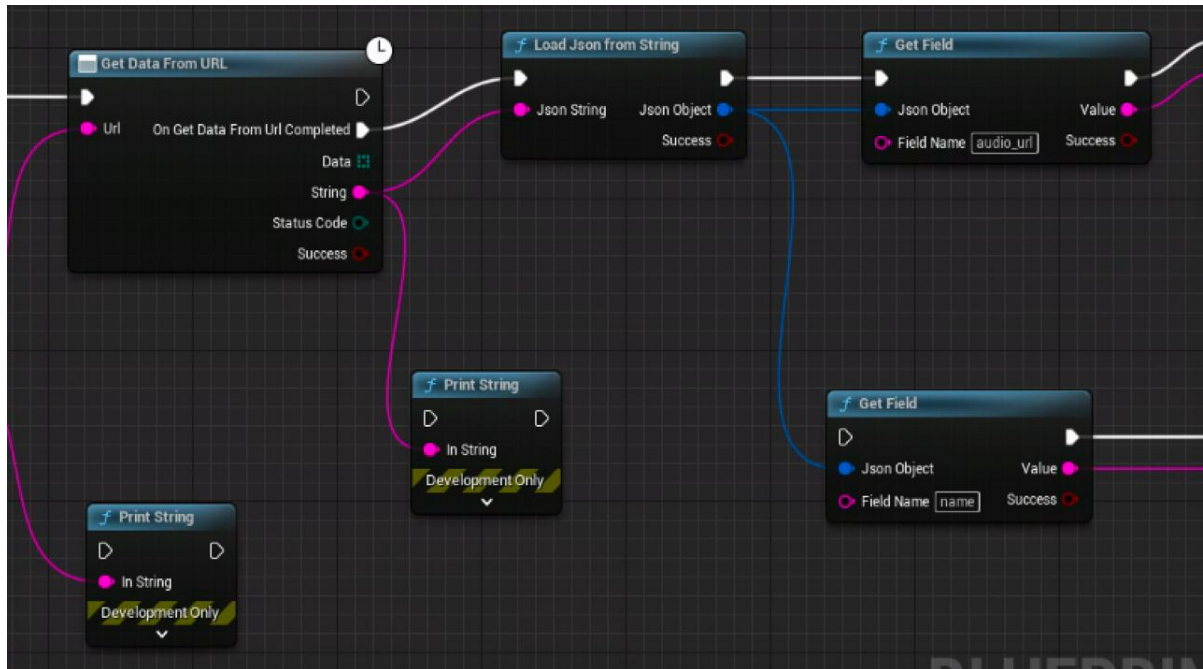
getAllTokensURI	Valid	true	Sukses
getAllTokensURI	Invalid	true	Gagal
getAllTokensURI	Valid	false	Gagal
getAllTokensURI	Invalid	false	Gagal

Dari tabel diatas dapat diketahui bahwa apabila alamat contract yang diberikan tidak valid atau tidak cocok maka akan terjadi error sehingga menyebabkan proses integrasi gagal. Jika alamat contract valid dan cocok dengan ABI dan contract juga valid maka pembacaan method akan mengembalikan nilai yang benar. Blueprint Unreal Engine 5 akan meneruskan balikan dari pemanggilan method smart contract ini untuk diproses pada blueprint-blueprint selanjutnya.



Gambar 4.8: Tampilan blueprint untuk pemanggilan method tokenURI

Blueprint-blueprint lainnya dalam Unreal Engine 5 dapat menggunakan nilai yang dikembalikan dari pemanggilan method smart contract untuk melanjutkan proses selanjutnya. Dengan demikian, integrasi yang berhasil antara Unreal Engine 5 dan smart contract memungkinkan penggunaan data blockchain dalam pengembangan aplikasi atau permainan. Hal ini membuka peluang untuk menciptakan pengalaman unik yang berbasis pada transaksi dan kepemilikan aset digital, seperti NFT. Blueprint-blueprint selanjutnya dapat menggunakan data dari smart contract untuk mengatur logika permainan, mengatur kepemilikan aset, atau bahkan memungkinkan interaksi dengan NFT yang dimiliki oleh pemain. Dengan adanya integrasi ini, Unreal Engine 5 dapat menjadi platform yang kuat untuk membangun pengalaman Metaverse yang terhubung dengan blockchain.



Gambar 4.9: Tampilan blueprint untuk load metadata

Selanjutnya adalah pengujian integrasi dengan pembacaan format metadata. Format metadata audio yang sudah dibaca pada proses sebelumnya akan diteruskan ke blueprint berikutnya. Berikut ini adalah tabel hasil pengujian proses memuat metadata.

Tabel 4.12: Hasil Pengujian Load Metadata Audio di UE5

Metadata format	Keterangan
Valid	Sukses
Invalid	Gagal

Dari tabel diatas dapat diketahui bahwa jika format metadata yang diberikan tidak valid maka akan terjadi error karena file JSON yang dibaca akan diambil key `audio_url`-nya, sehingga jika tidak terdapat key `audio_url` maka haruslah terjadi error. Ini adalah hasil yang dikehendaki.

Ketika keseluruhan data yang diberikan valid maka dapat dilanjutkan ke proses pengujian berikutnya yaitu proses pemutaran audio secara *real time*. Biasanya file audio asset pada game akan dimuat secara lokal, namun tidak dengan sistem ini dimana file yang akan diputar adalah file yang berasal dari internet.

File ini dimuat dengan membaca smart contract dan membaca audio url di dalam metadata yang ada di NFT pada smart contract. Untuk eksperimen berikut akan dilakukan dengan cara mencoba beberapa file audio yang akan di putar oleh runtime audio importer.

Tabel 4.13: Hasil Pengujian Pemutaran Audio di Unreal Engine 5

Audio format	Size	Keterangan
--------------	------	------------

mp3	5mb	Sukses
mp3	10mb	Sukses
mp3	1000mb	Gagal
wav	300kb	Sukses
wav	5mb	Sukses
wav	10mb	Sukses
flac	5mb	Sukses
flac	10mb	Sukses
ogg	5mb	Gagal
ogg	10mb	Gagal

Dari hasil pengujian diketahui bahwa format data yang disupport hanyalah mp3 saja, sementara untuk ukuran file juga berpengaruh. Semakin besar ukuran file maka waktu tunggu juga semakin lama. Akan tetapi karena file audio dimuat dalam memori jika ukuran file terlalu besar maka akan terjadi crash.

4.2 Evaluasi Pengujian

Berdasarkan pengujian yang telah dilakukan, pengujian sistem Sharing Data berbasis blockchain di metaverse dapat berjalan sebagaimana mestinya. Method *mint* pada *smart contract* hanya akan gagal apabila string yang diberikan kosong. Method *getAllTokens* akan mengembalikan array sepanjang banyaknya jumlah pemanggilan *mint*. Lalu untuk method *getTokenURI* akan mengembalikan array yang berisi URI yang sudah pernah di-*mint* sebelumnya.

Kemudian untuk pengujian *method tokenURI* diketahui bahwa *method* ini akan mengembalikan URI yang sesuai dengan urutan pemanggilan *mint*. URI tersebut disimpan dalam bentuk array sehingga apabila diberikan parameter bilangan yang tidak di dalam *range array* tersebut maka akan terjadi *error*. Kemudian terdapat juga pengujian gas dari proses *deploy smart contract* dan juga pemanggilan *mint*. Diperoleh bahwa pemakaian gas tertinggi didapatkan ketika dilakukan proses *deployment*. Lalu untuk *Gas Used* yang paling banyak digunakan di antara *test net* adalah Goerli. Lalu untuk pemanggilan *smart contract method* dengan *UE5 blueprint* didapati bahwa hanya akan sukses apabila ABI dan alamat *contract* yang diberikan valid. Lalu untuk jenis file yang didukung adalah file dengan format mp3, wav, dan flac. Sementara ogg belum mendapat *support*. Untuk ukuran file juga mempengaruhi delay daripada pemutaran audio ini, dengan file mp3 pada pengujian yang berukuran 1000mb maka akan terjadi gagal.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian yang mulai dari pengujian sistem Sharing Data Audio berbasis blockchain pada Smart Contract hingga integrasi unreal engine 5, diperoleh beberapa kesimpulan sebagai berikut:

1. *Smart contract* yang dibuat membutuhkan rerata konsumsi gas sebesar 3275354 ether untuk melakukan *deployment smart contract* di *testnet*, dan 3274368 di *Ganache*. Untuk pemanggilan *method mint* dibutuhkan gas sebesar 125574 di *testnet*
2. Sistem ini memiliki beberapa ketentuan format yang mengikat mengingat pada pengujian diperlukan data yang valid pada *JSON metadata*, yang memiliki setidaknya 1 *key audio_url*.
3. Sinergi untuk web3 tercapai dengan adanya integrasi dengan smart contract ethereum yang juga interoperable dengan blockchain lain seperti solana dan polygon.
4. Terdapat 3 file audio yang dapat dibaca oleh Unreal Engine 5 secara real time, yakni mp3, flac, dan wav.

5.2 Saran

Untuk pengembangan lebih lanjut pada penelitian ini, terdapat beberapa saran yang bisa dilakukan khususnya pada sistem data sharing berbasis blockchain ini, yang mana diantara lain:

1. Menggunakan *metasound* pada versi Unreal Engine 5.3 yang akan datang karena akan semakin erat kaitannya dengan web3 dan metaverse. Pada penelitian ini menggunakan audio player bawaan dari Unreal Engine 5.
2. Membuat sistem validasi yang lebih *robust* sehingga dapat mencegah error runtime baik pada smart contract dan juga Unreal Engine 5 terkhusus pada bagian blueprint.
3. Membuat support yang lebih pada beberapa format audio maupun format metadata yang lebih fleksibel sehingga mempermudah baik pengembang maupun pengguna.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

- Antonopoulos, A. M. (2014). *Mastering bitcoin: Unlocking digital cryptocurrencies*. O'Reilly Media.
- Antonopoulos, A. M., & Wood, G. (2018). *Mastering ethereum: Building smart contracts and dapps*. O'Reilly Media.
- Benet, J. (2014). Ipfs - content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*.
- Brunnhofner, M., & Pregonzer, A. (2019). Towards interoperability in blockchain ecosystems: State of the art and research challenges. *European Conference on Service-Oriented and Cloud Computing*, 121–135. https://doi.org/10.1007/978-3-030-32429-2_9
- Burniske, C., & Davis, V. (2019). *Ethereum for dummies*. For Dummies.
- Castronova, E. (2005). *Synthetic worlds: The business and culture of online games*. University of Chicago Press.
- De Angelis, S., & Panzieri, F. (2019). A taxonomy of blockchain interoperability. *Future Internet*, 11(6), 131. <https://doi.org/10.3390/fi11060131>
- Diedrich, H. (2017). *Ethereum: Blockchains, digital assets, smart contracts, decentralized autonomous organizations*. Apress.
- Dwivedi, Y. K., Hughes, L., Baabdullah, A. M., Ribeiro-Navarrete, S., Giannakis, M., Al-Debei, M. M., Dennehy, D., Metri, B., Buhalis, D., Cheung, C. M., Conboy, K., Doyle, R., Dubey, R., Dutot, V., Felix, R., Goyal, D., Gustafsson, A., Hinsch, C., Jebabli, I., ... Wamba, S. F. (2022). Metaverse beyond the hype: Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy. *International Journal of Information Management*, 66, 102542. <https://doi.org/https://doi.org/10.1016/j.ijinfomgt.2022.102542>
- Games, E. (2021). Unreal engine 5 [Diakses pada 12 Mei 2023]. <https://www.unrealengine.com/en-US/unreal-engine-5>
- Games, E. (2022a). *Unreal engine*. <https://www.unrealengine.com/en-US/>
- Games, E. (2022b). *Unreal engine architecture*. <https://docs.unrealengine.com/en-US/Architecture/index.html>
- Huang, X., & Chen, L. (2021). A comprehensive survey on blockchain interoperability: From theory to practice. *Journal of Network and Computer Applications*, 183, 102951. <https://doi.org/10.1016/j.jnca.2021.102951>
- Kietzmann, J., Hermkens, K., McCarthy, I. P., & Silvestre, B. S. (2011). Social media? get serious! understanding the functional building blocks of social media. *Business horizons*, 54(3), 241–251.
- Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.
- Naz, M., Al-zahrani, F. A., Khalid, R., Javaid, N., Qamar, A. M., Afzal, M. K., & Shafiq, M. (2019). Facebook buying oculus virtual-reality company for \$2 billion. *Sustainability*, 11(24), 7054.
- Noauthor. (2017). Introduction to cryptocurrency wallet security. *IACR Cryptology ePrint Archive*, 2017.
- Ozer, R. (2018). *Ethereum: The insider guide to blockchain technology, cryptocurrency and mining ethereum*. CreateSpace Independent Publishing Platform.
- Shirley, D. (2018). Erc-721 non-fungible token standard.

Solidity documentation. (n.d.).

Swan, M. (2015). *Blockchain: Blueprint for a new economy*. O'Reilly Media.

Swanson, T. (2016). *Blockchain interoperability* (tech. rep.). Oracle Corporation. <https://www.oracle.com/technetwork/topics/entarch/blockchain-interoperability-wp-5220633.pdf>

Tapscott, D., & Tapscott, A. (2016). *Blockchain revolution: How the technology behind bitcoin is changing money, business, and the world*. Penguin.

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger.

BIOGRAFI PENULIS



Aaron Christopher Tanhar adalah seorang penulis dan pengembang perangkat lunak yang lahir di Surabaya pada tanggal 8 Desember 2001. Penulis menyelesaikan pendidikannya di Institut Teknologi Sepuluh Nopember, dengan mengambil jurusan Teknik Komputer. Sejak awal masa studinya, penulis telah menunjukkan minat dan keahlian yang besar dalam bidang pengembangan perangkat lunak. Penulis aktif terlibat dalam berbagai kegiatan di kampus, termasuk menjadi anggota panitipenulis dalam acara MAGE 6 dan MAGE 7, yang merupakan acara besar dalam dunipenulis komputer dan teknologi. Ketertarikan penulis terutama terfokus pada pengembangan backend dan aplikasi. Penulis memiliki pemahaman yang mendalam tentang berbagai bahasa pemrograman dan kerangka kerja yang digunakan dalam pengembangan perangkat lunak modern. Keahliannya dalam backend development memungkinkan penulis untuk mengembangkan sistem yang kuat dan efisien. Selain itu, penulis juga memiliki ketertarikan khusus dalam topik blockchain. Dalam tugas akhirnya, penulis memilih untuk fokus pada penelitian dan pengembangan blockchain, sebuah teknologi yang memiliki potensi untuk merevolusi berbagai sektor, termasuk keuangan dan logistik. Dalam perjalanan penulisannya, penulis telah menghasilkan berbagai artikel dan tulisan teknis terkait dengan bidang pengembangan perangkat lunak. Penulis memiliki kemampuan komunikasi yang baik dan mampu menyampaikan ide-idenya dengan jelas dan terstruktur. Sebagai seorang yang berdedikasi dan bersemangat, penulis terus mengembangkan keterampilan dan pengetahuannya dalam industri teknologi yang terus berkembang. Penulis selalu berusaha untuk mengikuti perkembangan terbaru dalam dunipenulis pengembangan perangkat lunak dan mengaplikasikannya dalam pekerjaannya. Jika Anda ingin menghubungi penulis, Anda dapat menghubungi melalui email di me@aaronct.dev. Penulis sangat terbuka untuk kolaborasi dan diskusi terkait dengan pengembangan perangkat lunak dan topik terkait lainnya.

[Halaman ini sengaja dikosongkan]