SISTEM DATA SHARING UNTUK SCORE PADA GAME WIDOW WATERFALL MENGGUNAKAN ETHEREUM BLOCKCHAIN

SKRIPSI

oleh :
MUHAMMAD NAUFAL FIRDAUS
NIM. 16650013

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2020

LEMBAR PENGAJUAN

SISTEM DATA SHARING UNTUK SCORE PADA GAME WIDOW WATERFALL MENGGUNAKAN ETHEREUM BLOCKCHAIN

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
MUHAMMAD NAUFAL FIRDAUS
NIM. 16650013

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2020

LEMBAR PERSETUJUAN

SISTEM DATA SHARING UNTUK SCORE PADA GAME WIDOW WATERFALL MENGGUNAKAN ETHEREUM BLOCKCHAIN

SKRIPSI

oleh:
MUHAMMAD NAUFAL FIRDAUS
NIM. 16650013

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal : 08 Desember 2020

Dosen Pembimbing I	Dosen Pembimbing II
Yunifa Miftachul Arif, M.T	Hani Nurhayati, M.T
NIP. 198306162011011004	NIP. 197806252008012006

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

LEMBAR PENGESAHAN

SISTEM DATA SHARING UNTUK SCORE PADA GAME WIDOW WATERFALL MENGGUNAKAN ETHEREUM BLOCKCHAIN

SKRIPSI

Oleh:
MUHAMMAD NAUFAL FIRDAUS
NIM. 16650013

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 23 Desember 2020

Susunan Dewan Penguji :			**Tanda Tangan**
Penguji Utama	:	Fachrul Kurniawan, M.MT NIP. 197710202009121001	( )
Ketua Penguji	:	Fresy Nugroho, M.T NIP. 197107222011011001	( )
Sekertaris Penguji	:	Yunifa Miftachul Arif, M.T NIP. 198306162011011004	( )
Anggota Penguji	:	Hani Nurhayati, M.T NIP. 197806252008012006	( )

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini,

Nama	: Muhammad Naufal Firdaus
NIM	: 16650013
Jurusan	: Teknik Informatika
Fakultas	: Sains dan Teknologi
Judul Skripsi	: Sistem data *sharing* untuk score pada *game widow waterfall* menggunakan *Ethereum Blockchain*

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambilan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila di kemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan saya tersebut.

Malang, 08 Desember 2020
Yang membuat pernyataan,

Muhammad Naufal Firdaus
NIM.16650013

HALAMAN MOTTO

“Jangan lari dari masalah, hadapi dan nikmati prosesnya”

HALAMAN PERSEMBAHAN

الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ

Puji syukur kehadirat Allah, shalawat dan salam bagi Rasul-Nya

Penulis persembahkan sebuah karya ini kepada:

Kedua orang tua penulis tercinta, Bapak Abdul Rofik dan Ibu Lailatul Muttahidah yang telah mendidik penulis dan mempuyai jasa bagi penulis yang tak terhingga.

Dosen pembimbing penulis Bapak Yunifa Miftachul Arif, M.T dan Ibu Hani Nurhayati, MT yang telah dengan sabar membimbing jalannya penelitian skripsi ini dan selalu memberikan stimulus positif untuk tetap semangat menjalani setiap tahap ujian skripsi

Seluruh dosen Teknik Informatika UIN Maulana Malik Ibrahim Malang dan seluruh guru-guru penulis yang telah membimbing dan memberikan ilmunya yang sangat bermanfaat

Keluarga Teknik Informatika kelas A 2016 dan Andromeda (Teknik Informatika angkatan 2016) yang telah memberikan semangat dan doanya.

Orang-orang yang tak bisa penulis sebutkan satu per satu yang selalu memberikan semangat dan motivasinya kepada penulis untuk menyelesaikan skripsi ini.

KATA PENGANTAR

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

Puji dan syukur penulis panjatkan ke hadirat Allah subhanahu wa ta'ala yang telah melimpahkan rahmat dan hidayahNya kepada kita, sehingga penulis bisa menyelesaikan skripsi dengan tepat waktu, yang kami beri judul "Sistem Data Sharing Untuk Score Pada Game Widow Waterfall Menggunakan Ethereum Blockchain". Tujuan dari penyusunan skripsi ini guna memenuhi salah satu syarat untuk bisa menempuh ujian sarjana komputer pada Fakultas Sains dan Teknologi (FSAINTEK) Program Studi Teknik Informatika di Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang. Didalam pengerjaan skripsibini telah melibatkan banyak pihak yang sangat membantu dalam banyak hal. Oleh sebab itu, disini penulis sampaikan rasa terima kasih sedalam-dalamnya kepada:

1. Prof. Dr. Abdul Haris, M. Ag selaku Rektor Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.
2. Dr. Sri Harini, M.Si, Selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang. Dr. Chayo Crysdian, Selaku Ketua Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Ilsam Negeri (UIN) Maulana Malik Ibrahim Malang.
3. Yunifa Miftachul Arif, M.T Selaku Dosen Pembimbing I yang telah membimbing dalam penyusunan skripsi ini hingga selesai.
4. Hani Nurhayati, M.T selaku Dosen Pembimbing II yang telah membimbing dalam penyusunan skripsi ini hingga selesai.

5. Fatchurrohman, M.Kom, Selaku Dosen Wali yang senantiasa memberikan banyak motivasi dan saran untuk kebaikan penulis.
6. Fachrul Kurniawan, M.MT dan Fresy Nugroho, M.T selaku Dosen Penguji dengan sikap Profesional telah menguji seluruh proses ujian skripsi penulis.
7. Seluruh Dosen serta Staff Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.
8. Keluarga tercinta serta kerabat yang senantiasa memberikan doa dan dukungan semangat kepada penulis.
9. Sahabat-sahabat seperjuangan yang tiada henti memberi dukungan dan motivasi kepada penulis serta target bersama untuk lulus skripsi dan wisuda bersama
10. Semua pihak yang telah banyak membantu dalam penyusunan skripsi ini yang tidak bisa penulis sebutkan namanya.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih terdapat kekurangan dan penulis berharap semoga skripsi ini bisa memberikan manfaat kepada para pembaca khususnya bagi penulis secara pribadi.

Malang, 23 Desember 2020

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Firdaus, Muhammad Naufal. 2020. **Sistem Data Sharing Untuk *Score* Pada *Game Widow Waterfall* Menggunakan *Ethereum Blockchain*.** Skripsi. Jurusan Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing (I) Yunifa Miftachul Arif,.M.T, (II) Hani Nurhayati, M.T.

Kata Kunci : Sistem Data *Sharing*, *Ethereum Blockchain*, Sistem Skoring

Sistem data *sharing* untuk *score* adalah sebuah sistem dimana semua *score* yang diperoleh oleh *player* akan di urutkan berdasarkan perolehan poin *player*. Semakin banyak poin yang didapat maka *player* tersebut akan berada pada puncak *score* teratas *leaderboard*. Sistem *scoring* tersebut sudah umum digunakan dalam game *multiplayer*. pada *game* sendiri, banyak metode yang digunakan untuk sistem *scoring* maupun penyimpanan data jangka panjang. Pada penelitian *game widow waterfall* ini sistem tersebut akan di integrasikan dengan jaringan *Ethereum Blockchain*. *Ethereum blockchain* dalam *game* ini berfungsi sebagai penyimpanan data *score* pada *game widow waterfall* dengan media transaksi dalam jaringan tersebut. Transaksi yang sukses menandakan bahwa data sudah terkirim dan tersimpan didalam jaringan. Dari hasil penelitian didapatkan bahwa transaksi sukses dilakukan dan tervalidasi pada jaringan *ethereum blockchain.*

ABSTRACT

Firdaus, Muhammad Naufal. 2020. **Data Sharing System for Scoring in Widow Waterfall Game Using the Ethereum Blockchain.** MiniThesis. Department of Informatics Engineering Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University of Malang. Supervisor (I) Yunifa Miftachul Arif,.M.T, (II) Hani Nurhayati, M.T.

Keywords **:** Data Sharing System, Ethereum Blockchain, Scoring System

The data sharing system for scores is a system where all the scores obtained by the player will be sorted according to the player's points earned. The more points you get, the player will be at the top of the leaderboard score. This scoring system is commonly used in multiplayer games. In the game itself, many methods are used for scoring systems and long-term data storage. In this research, the widow waterfall game will be integrated with the Ethereum Blockchain network. The Ethereum blockchain in this game functions as a score data store in the widow waterfall game with transaction media in the network. A successful transaction indicates that the data has been sent and stored on the network. The results showed that the transaction was successful and validated on the ethereum blockchain network.

ملخص البحث

فردوس، محمد نافل .2020 .**نظام تبادل البيانات لتسجيل على لعبة شلال الأرملة اطروحه .قسم هندسة المعلوماتية، كلية العلوم .Ethereum Blockchain** باستخدام والتكنولوجيا، جامعة مولانا مالك إبراهيم الحكومية الإسلامية مالانغ .المشرف(1) يونيفا مفتاهول عارف،. م .ت (2).هاني نورهيتي، م .ت

: نظام تبادل البيانا ,Ethereum Blockchain ,نظام التسجيل: الكلمات الرئيسيه

نظام تبادل البيانات للدرجة هو نظام حيث سيتم فرز جميع الدرجات التي حصل عليها اللاعب على أساس نقاط اللاعب المكتسبة .كلما حصل اللاعب على نقاط أكثر، سيكون في قمة قائمة المتصدرين .نظام التهديف يستخدم عادة في ألعاب متعددة اللاعبين .في اللعبة نفسها، وتستخدم العديد من الأساليب لأنظمة التهديف، فضلا عن تخزين البيانات على المدى الطويل . Ethereum في هذا البحث لعبة شلال أرملة، سيتم دمج النظام مع شبكة في هذه اللعبة بمثابة تخزين درجة البيانات في Ethereum blockchain .بلوكشين لعبة شلال الأرملة مع وسائل الإعلام المعاملات في الشبكة .تشير الحركات الناجحة إلى أن البيانات قد تم إرسالها وتخزينها على الشبكة .من نتائج البحث التي تم الحصول عليها أن يتم تنفيذ المعاملات الناجحة والتحقق من صحتها ethereum blockchain على شبكة

BAB I

PENDAHULUAN

Pada Bab Pendahuluan ini Menjelaskan tentang latar belakang masalah, identifikasi masalah, batasan masalah, tujuan penelitian, manfaat penelitian serta sistematika penulisan. Berikut penjelasan mengenai bagian-bagian tersebut.

1.1 Latar Belakang

Perkembangan teknologi jaringan komputer pada era ini semakin canggih dan cepat. *Blockchain* merupakan salah satunya. *Blockchain* sendiri ditemukan oleh (Nakamoto, 2009) dan dirilis ke publik pada tahun 2009 bersamaan dengan terciptanya mata uang digital *bitcoin*. (Fauzan, 2018) menjelaskan bahwa teknologi *Blockchain* diciptakan untuk merombak skema sirkulasi perantara dalam melakukan transaksi. Transaksi antara A dan B bisa terjadi tanpa perantara, dalam waktu lebih singkat, biaya lebih murah, dan bahkan jauh lebih aman dibandingkan transaksi yang ditawarkan bank atau institusi serupa lainnya. Menurut (Mutar & AL-Huseiny, 2019) *Blockchain* memungkinkan seseorang untuk melakukan transaksi atau hal lainnya tanpa diperlukannya pihak ke 3 yang sering menimbulkan permasalahan. Dengan tidak adanya pihak ke 3 memungkinkan transparasi yang lebih baik. Pada umumnya *blokchain* hanya digunakan untuk menyimpan mata uang digital, tetapi *blokchain* juga berpotensi untuk digunakan dan dikembangkan di luar lingkup tersebut.

Seiring dengan berkembangnya teknologi *Blockhchain* dari tahun ke tahun, *Ethereum* pun terbentuk. *Ethereum* adalah salah satu dari banyak pengembangan jaringan *blockchain* yang sukses. Diusulkan oleh (Buterin, 2014), *Ethereum* pada

dasarnya berfungsi untuk menyimpan mata uang digital sama seperti jaringan *blockchain* lain namun dapat diprogram untuk kebutuhan lain yang artinya *ethereum* membuka kesempatan bagi para pengembang dari berbagai bidang untuk bisa memanfaatkan dan mengembangkan jaringan tersebut dan membuat produk mereka sendiri.

Salah satu pemanfaatan jaringan *Ethereum blockchain* adalah pada pembuatan *Game*. Pada pembuatan game khususnya *game multiplayer*, data *sharing* adalah hal yang penting dikarenakan beberapa data tersebut diperlukan untuk menampilkan informasi yang penting kepada pemain, seperti misalnya Informasi *Score* atau poin yang didapatkan oleh masing-masing pemain. Informasi *Score* pemain diperlukan karena hal tersebut dapat memunculkan sportifitas persaingan antar pemain, sehingga para pemain berlomba untuk mendapatkan nilai *Score* tertinggi dan mengalahkan pemain lain. Menurut (Studi et al., 2019) Umumnya data disimpan dalam *database* yang disimpan secara menggunakan *cloud server* pusat yang berjalan pada game tersebut yang dapat menjadi ancaman terhadap privasi data pengguna karena pengguna tidak mengetahui bagaimana pengelolaan dan pendistribusian data oleh pihak *cloud*. Privasi merupakan hak pengguna untuk menjaga kerahasiaan dan kontrol atas informasi yang dimiliknya ketika diberikan kepada pihak lain.

Berbeda ketika data tersebut disimpan dalam *Ethereum blokchain*. Data tersebut akan ada dan dapat dilihat selama jaringan *Ethereum blockchain* masih ada. Karena data tersebut tersimpan dalam sebuah blok yang terenkripsi dan dapat dilihat serta divalidasi oleh semua yang terhubung ke dalam jaringan *blockchain*. Ketika ada

data baru yang disimpan maka data tersebut akan disimpan dalam blok yang terhubung oleh blok-blok sebelumnya sehingga memiliki tingkat kemanan tinggi. *Ethereum blokchain* tidak memiliki server pusat, melainkan masing masing komputer menjadi *server* dan *clientnya* sendiri (Buterin, 2014). Dan ketika ada *system crash*, data tersebut masih ada dalam jaringan karena data tersebut sudah disimpan dan didistribusikan ke seluruh jaringan. Maka dari itu dipilihnya Jaringan *Ethereum Blokchain* untuk digunakan sebagai data *sharing Score* pemain pada *game Widow Waterfall*.

Dalam Penelitian ini penulis berharap, pemanfaatan teknologi *Blockchain* dapat diaplikasikan kedalam berbagai macam produk tidak hanya *Game* saja. Karena masih banyak peluang terbuka bagi pengembang untuk mengembangkan jaringan *Ethereum blokchain* yang mungkin akan bermanfaat bagi masyarakat luas

1.2 Identifikasi Masalah

Berdasarkan dari latar belakang yang dijelaskan diatas, maka dapat diidentifikasi permasalahan sebagai berikut :

1. Bagaimana Mendapatkan data *Score Game Widow Waterfall* menggunakan *Ethereum Blockchain* ?

1.3 Tujuan Penelitian

Adapun tujuan dalam penelitian ini adalah

1. Untuk mendapatkan data *Score Game Widow Waterfall* menggunakan *Ethereum Blockhchain*

1.4 Manfaat Penelitian

Adapun manfaat yang dapat dihasilkan dalam penelitian ini adalah :

1. Untuk mengetahui cara kerja *Ethereum Blockchain* ketika digunakan untuk mendapatkan data *Score* pada *Game Widow Waterfall*

1.5 Batasan Penelitian

Berikut batasan penelitian dibuat penulis guna membatasi sistem yang diusulkan, yaitu sebagai berikut :

1. *Game* merupakan game bertipe *Multiplayer*
2. Dimainkan oleh Pemain yang memiliki akun *Ethereum*
3. Data yang akan didapatkan adalah nilai *Score* yang kumpulkan oleh Pemain

BAB II

TINJAUAN PUSTAKA

Pada bab ini di kemukakan tentang hal-hal teori -teori yang berkaitan dengan permasalahan dan ruang lingkup dalam penelitian ini.

2.1 Penelitian Terkait

Penelitian yang dilakukan oleh (Dinh et al., 2018), dengan judul *Untangling Blockchain : A data processing view of blockchain system* berfokus pasa analisis sistem dari *blockchain* itu sendiri. Fokus tersebut terbagi menjadi 4 bagian yaitu distribusi, kriptografi, protokol konsensus dan kontrak pintar (*smart contract*). Analisis dilakukan di dalam 3 jaringan sistem blockchain utama yaitu, *Ethereum, Parity* dan *Hyperledger Fabric*. Dari analisis yang dilakukan di dalam 3 jaringan *blockchain* tersebut menunjukkan bahwa adanya kesenjangan kinerja besar dan desain arsitektur antara *blockchain* dan sistem *database*.

Penelitian yang dilakukan dengan judul *Simulating a Blockchain Network with SimBlock* menghasilkan sebuah sistem yang bernama *SimBlock*. *SimBlock* sendiri merupakan simulator jaringan *blockchain* yang mensimulasikan jaringan *peer to peer* dari *blokchain* publik seperti *Bitcoin* yang terdiri dari ribuan *node* serta parameter yang terkait dengan *blockchain* dan jaringannya yang dapat dikonfigurasi secara fleksibel. Dengan hasil strategi bagaimana pemilihan *node* tetangga yang lebih baik dan menilai pengaruh dari relai jaringan (Banno & Shudo, 2019)

Penelitian yang dilakukan oleh (Fauzan, 2018) dengan judul *Teknologi blockchain dan peranannya dalam era digital* menunjukkan bahwa teknologi *blockchain* menyederhanakan proses proses konvensional yang menghabiskan

banyak waktu dan membutuhkan banyak pihak. Teknologi *blockchain* dianggap lebih unggul dikarenakan bisa diterapkan dan diimplementasikan ke banyak disiplin keilmuan maupun sektor industri. Dengan versi *Blockchain* 2.0 yang memungkinkan pertukaran nilai tanpa pihak ke tiga. Dengan tidak adanya pihak ke 3, maka kecepatan dalam transfer data ataupun pertukaran nilai akan semakin cepat karena tidak harus melewati sistem kontrol(*server*).

Penelitian yang dilakukan oleh (Hu et al., 2019) dengan judul *A Delay Tolerant Payement Scheme Base on the Ethereum Blockchain* mengusulkan bahwa diperlukan skema pembayaran digital berbasis *blockchain* yang dapat memberikan layanan yang baik dan andal. Dengan memanfaatkan keuntungan dari jaminan verifikasi dari teknologi *blockchain* untuk verifikasi transaksi keuangan dan memanfaatkan kontrak pintar untuk manajemen layanan yang aman.

Penelitian yang dilakukan oleh (Jani, 2018) dengan judul *An Overview of Ethereum and its Comparison with Bitcoin* mengatakan bahwa terciptanya *Bitcoin* sebagai teknologi *blockchain* pertama adalah sebagai alat konsesus terdistribusi. Seiring dengan perkembangan teknologi *blockchain* maka muncul beberapa aplikasi umum dari teknologi *blockchain* yang menggunakan inti aset dari *bitcoin* sendiri diantaranya adalah harus mewaliki mata uang khusus, kepemilikan properti, nama koin yang tidak bisa di tukar, keamanan, dan kontrak pintar.

2.2 Landasan Teori

Sub bab ini menjelaskan tentang teori teori yang digunakan dalam penelitian terkait.

2.2.1 Ethereum

Ethereum adalah sebuah *blockchain* yang dapat diprogram, memungkinkan siapa saja untuk menulis *smart contract* dan aplikasi terdesentralisasi di mana mereka dapat membuat aturan sesuai dengan keinginan sendiri untuk kepemilikan, format transaksi dan fungsi transisi negara Seiring dengan perkembangan *ethereum*, kontrak pintar tidak hanya digunakan untuk pembayaran otomatis, tetapi kontrak pintar dapat dikembangkan dan di sesuaikan dengan kebutuhan dari masing masing pihak yang berkepentingan (Buterin, 2014).

Menurut (Jani, 2018), *Ethereum* sendiri merupakan jaringan *peer* to *peer* publik dari mesin virtual yang dapat digunakan oleh setiap pengembang untuk membuat atau menjalankan aplikasi terdistribusi(*DApps*) yang memiliki mata uang digital sendiri yang disebut dengan *ether*. *Dapps* sendiri merupakan sebutan untuk program atau aplikasi yang diciptakan atau dikembangkan dengan memanfaatkan jaringan *blockchain Ethereum* dengan memanfaatkan kontrak pintar(*smart contract*). *Ethereum* menggunakan *blockchain* publik terdesentralisasi untuk menyimpan, melaksanakan, dan melindungi kontrak pintar ini secara kriptografis. Setiap komputer dalam jaringannnya akan mengunduh mesin virtual kecil yang berguna untuk sinkronisasi dengan jaringan *blokchain ethereum* yang akan terus melakukan ekseskusi kontrak pintar(*smart contract*).

(Jani, 2018) juga menjelaskan bahwa jaringan dalam ethereum dengan mudah memberikan keamanan, dan daya komputasi yang diperlukan untuk melakukan perancangan pengembangan sebuah aplikasi. Kontrak pintar dapat diprogram untuk melakukan hal tertentu seperti akan melakukan sebuah eksekusi jika ada kondisi yang terpenuhi atau kontrak pintar tersebut dibuat hanya untuk menyimpan sebuah nilai. Di dalam jaringan *ethereum* terdiri dari objek yang disebut "akun", dengan setiap akun memiliki alamat dengan panjang 20 *byte* dan informasi transaksi untuk menunjukkan blok sebelumnya untuk membedakan nilai dan informasi antar akun. Setiap akun memiliki 4 *field* antara lain :

1. *Nonce* : Penghitung yang digunakan untuk memastikan setiap transaksi hanya dapat di proses sekali.
2. Saldo Ether saat ini.
3. Kode kontrak akun, jika memang ada kontak pintar yang memberikan kondisi tertentu.
4. Dompet Akun (kosong secara *default*).

"*Eter*" adalah bahan *crypto* utama di dalam jaringan *Ethereum* dan digunakan untuk membayar biaya transaksi. Perlu diperhatikan bahwasannya "kontrak" atau *smart contract* dalam *ethereum* tidak boleh dilihat sebagai sesuatu yang harus "dipatuhi" atau "dipenuhi", melainkan lebih seperti "agen otonom" yang hidup di dalam lingkungan eksekusi jaringan *ethereum*. Memiliki kontrol langsung atas keseimbangan *eter* dan kunci untuk melacak nilai. Istilah "transaksi" yang digunakan dalam jaringan *ethereum* merujuk pada paket data yang di tanda tangani

yang menyimpan pesan untuk dikirim dari akun yang dimiliki secara eksternal. Sebuah "transaksi" mengandung :

1. Penerima pesan
2. Tanda tangan yang mengidentifikasikan bahwa pengirim.
3. Jumlah eter untuk ditransfer dari pengirim ke penerima.
4. Bidang data (opsional).
5. Nilai *StartGas* : mewakili jumlah masksimum langkah komputasi yang dapat dilakukan eksekusi transaksi.
6. Nilai *GasPrice* : mewakili biaya yang dibayar pengirim perlangkah untuk melakukan komputasi.

Dari beberapa list yang disebutkan tadi, 3 yang pertama adalah bidang standar yang dimiliki oleh setiap jaringan mata uang digital. *StartGas* dan *GasPrice* sangat penting bagi *ethereum* sebagai sistem penolakan untuk mencegah terciptanya *loop* tidak terbatas yang nantinya akan mempengaruhi sistem komputasi lainnya. Unit dasar perhitungannya adalah disebut dengan "*gas*". Setiap langkah perhitungan mempunyai biaya 1 *gas* atau setiap *byte* data mempunyai biaya 5 gas. Tetapi ada beberapa komputasi operasi yang membutuhkan jumlah gas yang banyak karena tingkat komputasi yang lebih sulit atau banyaknya data yang harus disimpan sebagai bagian dari fitur akun. Maksud dari sistem biaya didalam jaringan ethereum adalah sebagai biaya perawatan dan pemeliharaan jaringan *blockchain ethereum* secara professional untuk setiap sumber daya, perhitungan, bandwith, dan penyimpanan. Karena setiap transaksi yang mengarah ke jaringan *etherereum*

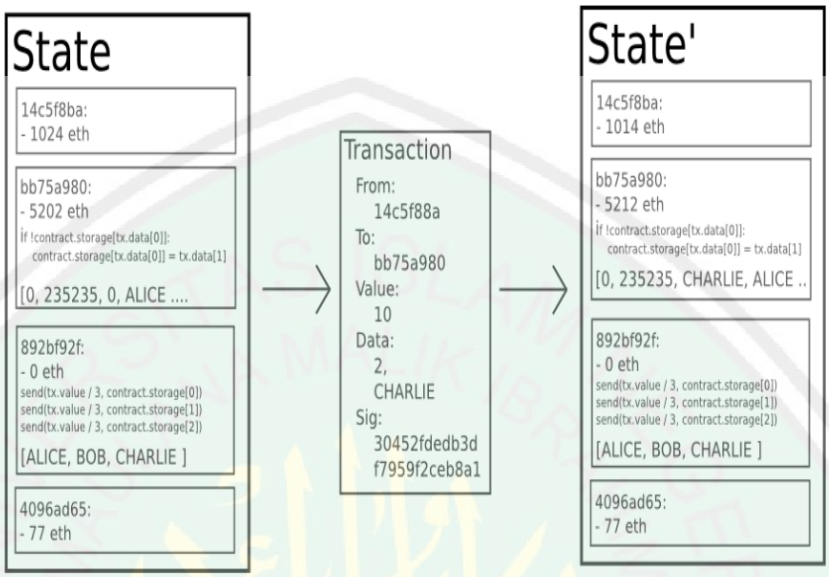
mengonsumsi sejumlah besar sumberdaya yang ada sehingga harus memiliki biaya *gas* yang proposional dengan kenaikan (Jani, 2018).

2.2.1.1 Cara Kerja Ethereum

Berikut cara kerja *ethereum* menurut (Jani, 2018) antara lain :

1. Periksa apakah transaksi telah dilakukan dengan baik, sebagai contoh apakah jumlah *eter* dan biaya gas memiliki nilai yang tepat. Tanda tangan valid, dan memeriksa apa angka yang dikirim akan sama ketika sudah sampai di akun penerima. Jika tidak maka transaksi akan dikembalikan.
2. Hitung biaya transaksi sebagai *StartGas* dan *GasPrice* dan tetukan alamat pengiriman dari tanda tangan.kurangi biaya saldo akun pengirim dan naikkan nilai penerima. Jika tidak ada saldo yang cukup untuk dibelanjakan, maka transaksi tidak bisa diteruskan dan akan dikembalikan.
3. Insiasi *Gas = GasStart* dan mengurangi sejumlah gas per-*byte* untuk membayar *byte* dalam transaksi.
4. Transfer nilai transaksi dari akun pengirim ke akun penerima. Jika belum ada akun penerima maka cari akun lain atau membuatnya sendiri. Jika akun pertama adalah kontrak, jalankan kode kontrak untuk menyelesaikan transaksi sampai kehabisan biaya *gas*.
5. Jika transfer nilai gagal karena pengirim tidak memiliki cukup ruang atau tidak cukupnya gas untuk melakukan eksekusi, kembalikan semua perubahan keadaan kecuali pembayaran biaya dan alihkan biaya ke akun penambang.

6. Kalau tidak, kembalikan biaya untuk semua *gas* yang tersisa ke pengirim dan kirim biaya yang dibayarkan untuk *gas* yang dikonsumsi penambang.

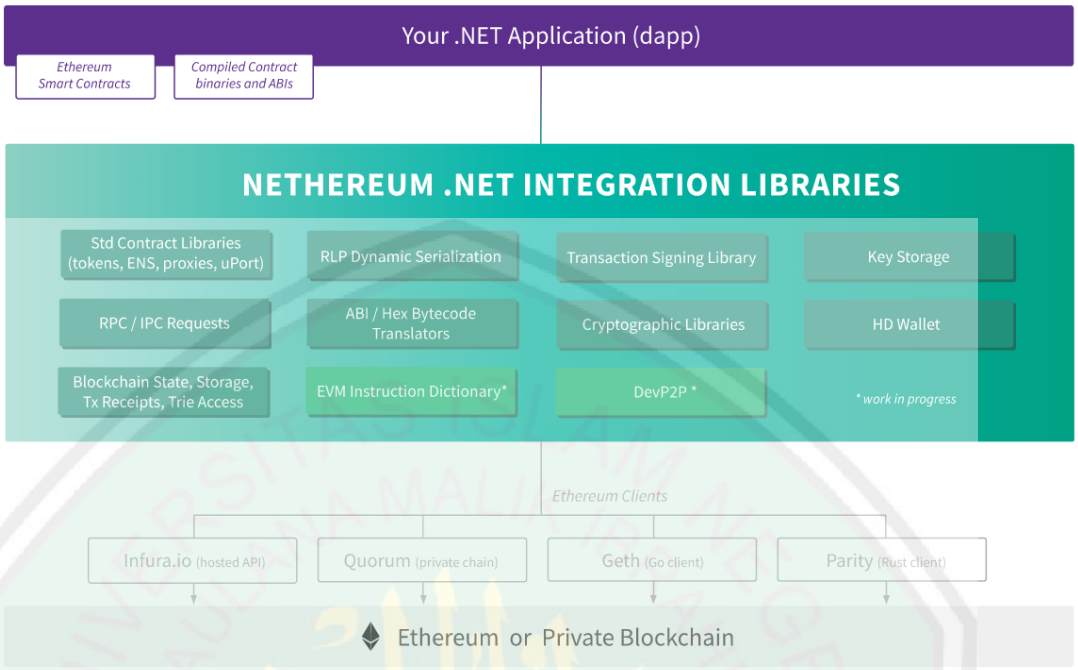


Gambar 2.1 Transaksi padaEthereum
Sumber : (Jani, 2018)

Jika tidak ada kontrak yang dibuat di akun penerima transaksi, maka total biaya transaksi akan sama dengan *GasPrice* yang disediakan dikalikan dengan panjang transaksi dalam *byte*, dan data yang dikirim bersamaan dengan transaksi akam menjadi tidak relevan.

2.2.2 Nethereum

Dikutip dari dokumentasi Nethereum pada Laman http://docs.nethereum.com/, *Nethereum* Adalah pustaka / *library* integrasi *.Net* untuk *Ethereum*. Menyederhanakan akses dan interaksi kontrak pintar dengan *node* jaringan *ethereum* publik maupun jaringan *ethereum* yang membutuhkan perizinan seperti *Geth*, *Parity* atau *quorum*.



Gambar 2.2 Nethereum
Sumber : (Nethereum.com)

Nethereum dikembangkan dengan *netstandard* 1.1, *net451* dan juga *library portable*, sehingga kompatibel denga semua sistem operasi (*Windows*, *Linux*, *MacOs*, *Android* dan *OSX*). Adapun beberapa Fitur dari *Nethereum* antara lain :

1. Fitur *JSON RPC/IPC*
2. Manajemen *Geth API*
3. Manajemen *Parity API*
4. Integrasi *Quorum*
5. Interaksi Kontrak Pintar (*smart contract*) yang disederhanakan
6. Integrasi *Unity* 3D
7. Encoding dan *decoding* dari *ABI* ke *.Net*
8. Transaksi, *RLP* dan penandatanganan pesan, verifikasi dan pemulihan akun

9. Pustaka library untuk kontrak standar pengujuan *Token*, *ENS* dan *Uport*
10. Penyimpanan kunci menggunakan standar *Web3*, kompatibel dengan *Geth* maupun *Parity*
11. Siklus hidup akun disederhanakan untuk di kelola oleh pribadi atau berdiri sendiri (transaksi yang ditandatangani).
12. Intersepsi panggilan *RPC* yang rendah
13. Pembuatan kode layanan kontrak pintar
14. *Hd wallet*

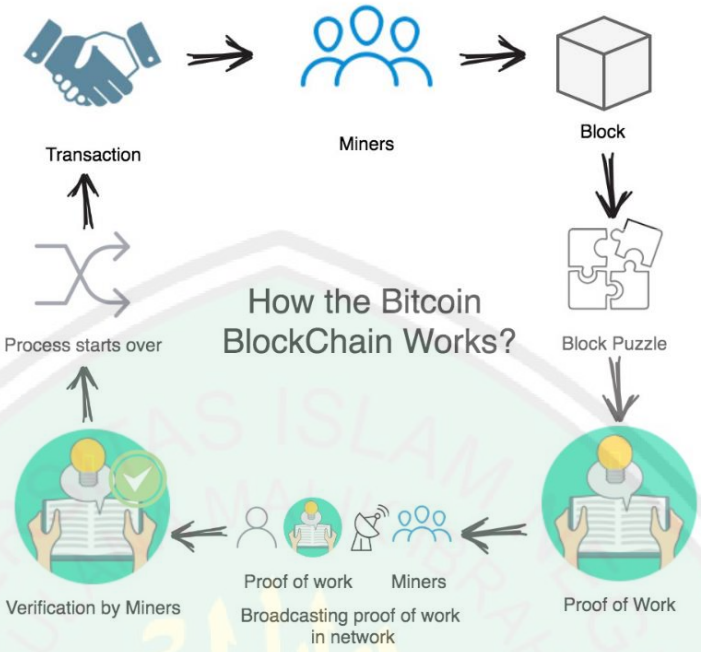
2.2.3 Blockchain

Blockchain adalah landasan berdirinya teknologi *bitcoin*, *NIST* (institut nasional dan Teknologi) telah menyatakan bahwa *blockchain* di ibaratkan sebuah buku besar digital yang terdistribusi dari transaksi transaksi yang ditandatangani secara kriptografis yang dikelompokkan ke dalam blok-blok. Setiap blok berkaitan secara kriptografis dengan blok yang sebelumnya setelah validasi dan menjalani keputusan konsensus. Ketika blok baru ditambahkan, blok lama menjadi lebih sulit di modifikasi. Blok baru direplikasi di semua salinan buku besar dalam jaringan, dan setiap masalah atau konflik diselesaikan secara otomatis menggunakan aturan yang ditetapkan (Liu, 2019).

Dari penjelasan diatas, maka pada dasarnya *blockchain* dapat diibaratkan sebagai basis data dengan struktur khusus dari blok rantai. Data dienkripsi untuk menjaga privasi dan disimpan dalam sebuah blok. Struktur rantai dan mekanisme validasi dan konsensus memastikan integrasi data. Salinan berulang dan penyimpanan terdistribusi sehingga meningkatkan kemanan data.

(Saini, 2019) menjelaskan Ada 4 prinsip dasar yang mendasari teknologi *blockchain*. Prinsip prinsip tersebut adalah Basis data Terdistribusi, transmisi *peer to peer*, Transparansi, dan logika komputasi.

1. Basis data terdistribusi : Setiap entitas pada *blockchain* memiliki akses ke seluruh *database* dan riwayat lengkapnya. Tidak ada entitas tunggal yang mengendalikan informasi dan setiap entitas dapat memverifikasi catatan dari mitra transaksinya secara langsung, tanpa perantara. *Blockchain* pada dasarnya buku besar yang terbuka dan didistribusikan. Buku besar ini mencatat semua transaksi antara dua entitas secara efisien dan dengan mudah dapat diverfikasi satu sama lain.
2. Transmisi *peer to peer* : Semua yang berkepentingan dan terlibat dalam sistem berjalan pararel dan *sharing resources*.
3. Transparansi : semua entitas yang berada pada jaringan bisa mengetahui dan menverifikasi transaksi yang telah atau yang sedang berlangsung.
4. Logika Komputasi : Transaksi dalam *blockchain* dapat diartikan dengan logika komputasi karena adanya buku besar digital yang bisa diprogram. Ada kemungkinan untuk mengatur aturan dan algoritma dimana algoritma ini dapat secara otomatis melakukan transaksi di berbagai entitas atau *node*

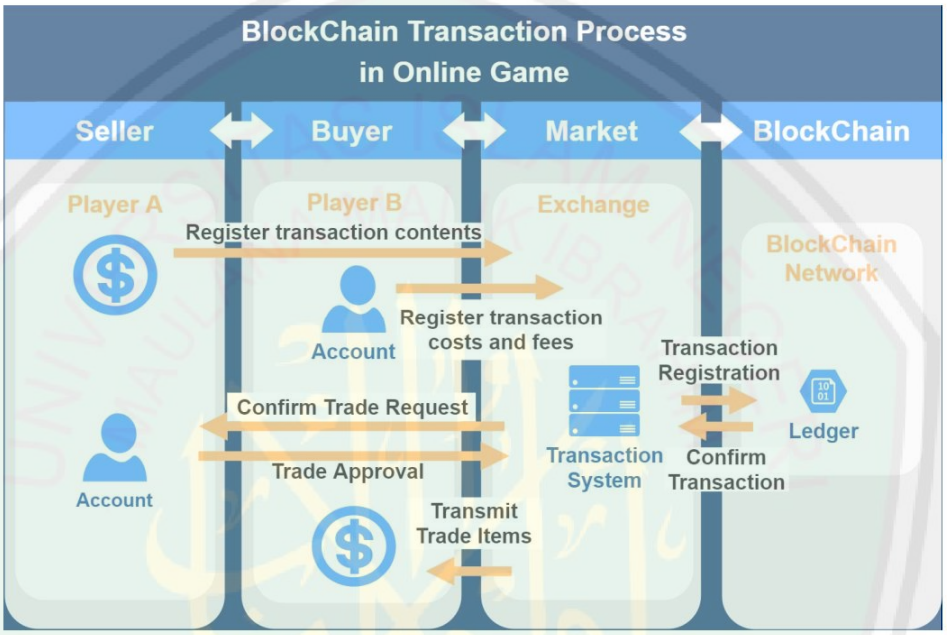


Gambar 2.3 Cara Kerja Blockchain
Sumber : (Byun & Byun, 2018)

Semua catatan transaksi yang pernah dilakukan akan di simpan di dalam entitas yang di sebut blok. Blok blok tersebut saling terhubung satu sama lainnya sehingga membentuk rantai blok (*chain*). Satu blok bisa menyimpan beberapa transaksi, sehingga ketika suatu blok sudah penuh, maka akan membuat blok baru yang tersambung dengan blok sebelumnya sehingga semua data transaksi akan terus terhubung satu sama lainnya.

Menurut (Dinh et al., 2018) Jaringan blockchain terpisah menjadi 2 yaitu jaringan blockchain publik dan jaringan blockchain pribadi. Publik blockchain bisa diakses dan semua orang bisa berpartisipasi didalamnya, sedangkan Private blockchain atau blockchain pribadi hanya bisa diakses oleh peserta yang dikenal dan dipercaya seperti Grup industri atau grup perusahaan yang dimiliki oleh payung perusahaan. Adapun beberapa keuntungan menggunakan jaringan *blockhain* ialah

proses transaksi menjadi lebih cepat, transaksi lebih murah karena tidak adanya pihak ketiga yang mengatur segala transaksi, lebih transparan karena semua orang yang berada di jaringan bisa melihat dan mevalidasi transaksi yang sudah tersebar, dan jaminan keamanan yang lebih baik.



Gambar 2.4 Blockchain dalam game online
Sumber : (Byun & Byun, 2018)

(Byun & Byun, 2018) menjelaskan Pada **Gambar 2.4** Ketika transaksi terjadi pada jaringan *blockchain* sebagai berikut :

(1) Penjual mendaftarkan jumlah token yang ingin dijual ke sistem perdagangan dalam game dan jumlah uang game yang sesuai dengan nilai token. Sistem memeriksa untuk melihat apakah penjual memiliki cukup bahan bakar dan token. Jika itu tidak cukup, sistem tidak mendaftarkan perdagangan token.

(2) Pembeli meminta pembelian dengan harga yang ingin ia beli dan sistem memeriksa apakah ia memiliki uang permainan yang ia tawarkan, termasuk

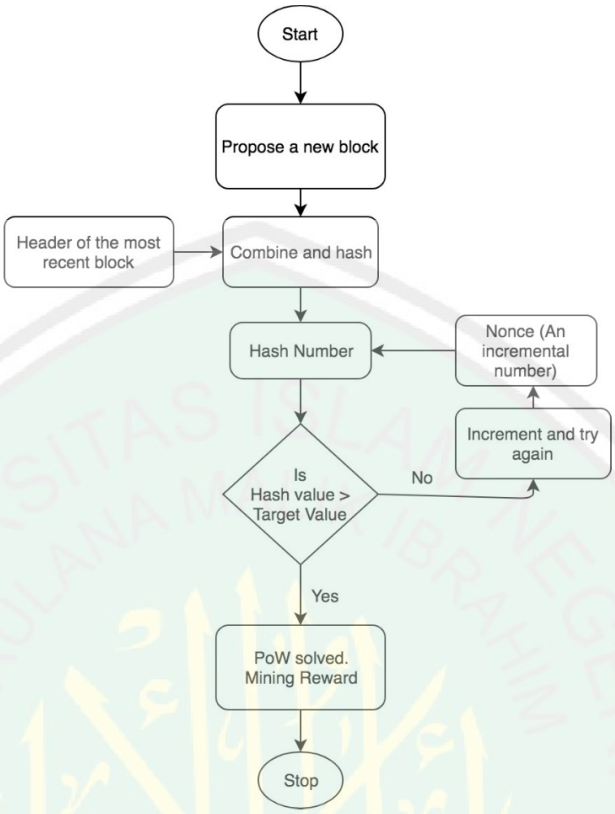
biayanya. Permintaan pembelian terdaftar dalam sistem jika pembeli memenuhi persyaratan.

(3) Penjual mengonfirmasi permintaan pembelian dan menyetujui transaksi. Token dikirim ke pembeli menggunakan bahan bakar penjual. Pada saat ini, beberapa biaya transaksi pembeli dibayarkan kepada penjual dan sisanya dikumpulkan dalam sistem permainan. Ini memberikan kompensasi kepada penjual untuk beberapa bahan bakar yang digunakan untuk dijual dan ini merupakan peluang bagi perusahaan game untuk memulihkan sebagian dari koin mereka yang telah mereka bagikan.

(4) Jika suatu transaksi didaftarkan secara normal dalam rantai blok, token dikirim ke akun pembeli. Jika transaksi tidak terdaftar secara normal dalam rantai blok, uang permainan yang disajikan oleh pembeli dikembalikan kecuali biaya transaksi yang sudah dibayarkan. Penjual harus mendaftar ulang transaksi jika dia ingin dijual kembali.

2.2.4 Proof Of Work (PoW)

Pada dasarnya *Proof of Work* seperti yang dikutip dari laman *ledger.com* adalah algoritma konsensus jaringan *blockchain* yang merupakan model konsensus yang mendasari *bitcoin*. *Bitcoin* merupakan mata uang kripto pelopor penggunaan *Proof of work*. *Proof of* Work merupakan konsensus untuk menuju kesepakatan didalam jaringan *blockchain* untuk mengkonfirmasi transaksi dan memproduksi blok baru pada rantai jaringan.



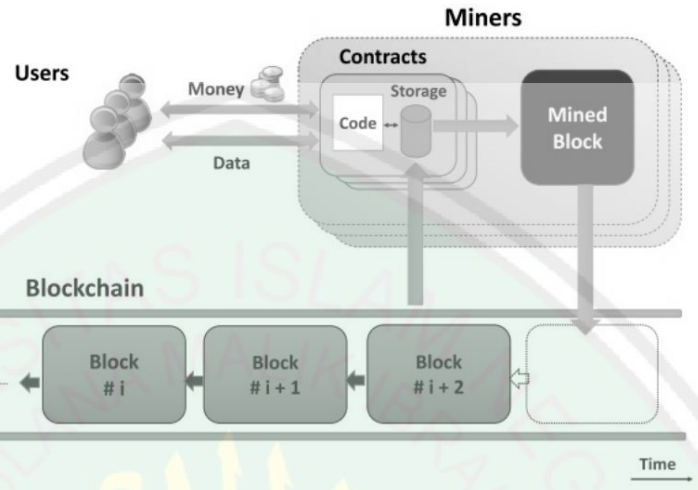
Gambar 2.5 Flow Chart Proof Of Work
Sumber : (Ghimire & Selvaraj, 2019)

Dengan *proof of work*, para penambang berkompetisi dengan penambang lainnya untuk mengkonfirmasi transaksi yang kemudian akan mendapatkan imbalan. Penambang adalah partisipan yang belomba untuk memecahkan masalah komputasi yang rumit yang nantinya akan di sebarkan ke jaringan *blockchain*. Kemungkinan penambang menyelesaikan masalah komputasi dan membuat blok baru di tentukan oleh kekuatan kompitasi masing masing komputer (Ghimire & Selvaraj, 2019).

2.2.5 Smart Contract

(Kalra et al., 2018) menjelaskan bahwa Smart *contract* merupakan sebuah baris kode komputer yang didalamnya mengandung persyaratan atau kondisi

tertentu yang dimana menentukan berhasilnya sebuah transaksi antar dua belah pihak yang bersangkutan.



Gambar 2.6 Smart Contract
Sumber : (Alharby & Moorsel, 2017)

Smart contract berjalan secara otomatis di jaringan *blockchain* publik dimana semua transaksi tercatat dan terduplikasi sehingga memiliki tingkat keamanan yang tinggi. *Smart contract* juga mengapus adanya pihak ketiga seperti bank, atau perantara lainnya yang menyebabkan adanya isu kepercayaan atau ketidakpercayaan terhadap pihak ketiga tersebut. Bahasa pemrograman yang di pakai untuk menulis *smart contract* adalah bahasa pemrograman *Solidity* yang dipengaruhi oleh bahasa pemrograman *C++*, *Phyton*, serta *Javascript* dan di desain untuk menargetkan jaringan *Blockhain Ethereum*. Untuk pengujian, pengembangan serta *compile smart contract* bisa menggunakan *online web based IDE* yaitu *Remix* (Alharby & Moorsel, 2017).

2.2.6 Score

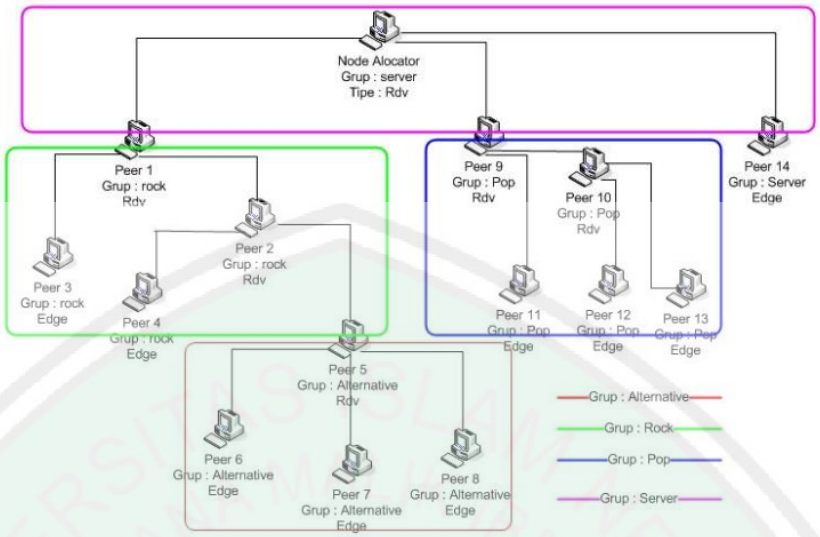
Sistem Scoring yang digunakan dalam segala jenis game dapat memiliki pengaruh yang cukup besar pada kepuasan pemain selama bermain game. Penilaian

berperan sebagai jenis umpan balik positif dan sistem penghargaan yang mampu mendorong pemain menuju tantangan yang lebih besar. Penilaian *Score* didasarkan pada ukuran objektif seperti jumlah kemenangan, banyaknya poin yang didapat, *Health bar*, atau tingkat akurasi yang sukses, dan didapat secara otomastis dan dihitung selama pertandingan (Lee et al., 2017).

Score juga merupakan titik tingkat keberhasilan para pemain dalam menyelasaikan tantangan -tantangan, misi -misi yang ada di dalam *game* atau juga sebagai penentuan *Ranking* pada suatu permainan yang melibatkan banyak orang seperti game *multiplayer*. *Score* pada *game* juga dapat digunakan untuk menentukan tingkat kesulitan yang akan diberikan kepada para pemain yang tentu saja mempunyai tujuan agar para pemain baik pemula ataupun yang berpengalaman dapat menikmati tantangan yang disajikan dalam *game* tersebut.

2.2.7 Data Sharing

(Shiddiqi et al., 2010) menjelaskan bahwa Peer to Peer (P2P) adalah metode untuk mengkoneksikan dua komputer untuk membentuk jaringan yang sederhana. Pengkoneksian ini dimaksudkan untuk membagi resource yang ada dikomputer yang satu dengan komputer yang lain. Konsep pembangunan jaringan Peer to Peer ini kemudian diterapkan untuk membangun sebuah aplikasi yang berkemampuan untuk mengkoneksikan secara Peer to Peer komputer yang terhubung dengan jaringan internet. Kelebihan jaringan komputer dengan menerapkan Data Sharing adalah. dapat memback-up data atau menyimpan data, menghemat waktu dalam proses pengiriman, dan dapat mengurangi biaya duplikasi data.



Gambar 2.7 jaringan peer -to-peer
Sumber : (Shiddiqi et al., 2010)

BAB III

METODOLOGI PENELITIAN

Pada bab ini menjelasakan tentang perancangan sistem implementasi teknologi *Blockchain* berbasis *Ethereum* dalam *Game Widow Waterfall* untuk sistem *Score*. Analisa dan perancangan sistem dibuat untuk mempermudah proses pembuatan *Game Widow Waterfall*.

3.1 Analisis dan Perancangan Game

Analisa diperlukan dalam sebuah *game* atau aplikasi untuk mengetahui permasalahan permasalahan yang ada sehingga dapat menentukan kebutuhan kebuthan yang ada untuk mengatasi permasalahan tersebut yang akan membantu memudahkan dalam proses pembuatan *game*. Pada penelitian ini, penulis akan membangun sebuah *game Multiplayer* yang mengambil Tema Wisata Alam Coban Rondo dengan memanfaatkan Teknologi *Ethereum Blockhcain* yang diterapkan pada Sistem *Score Game*.

3.2 Storyboard

Game Widow Waterfall merupakan *multiplayer game* dengan genre *adventure game*. Tokoh utama pada *game* ini merupakan wisatawan yang sedang berada di wisata Coban Rondo yang harus mengumpulkan poin tertinggi yang tersebar sepanjang wisata Coban Rondo. Didalam Al - Qur'an dijelaskan bahwa manusia juga perlu untuk berwisata sebagai bentuk wujud syukur kepada allah sebagaimana di paparkan dalam Surah Al-Mulk Ayat 15 :

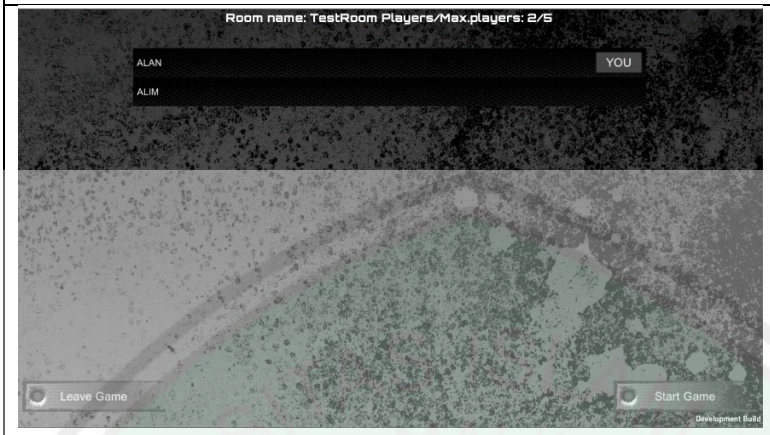
هُوَ الَّذِي جَعَلَ لَكُمُ الْأَرْضَ ذَلُولًا فَامْشُوا فِي مَنَاكِبِهَا وَكُلُوا مِن رِزْقِهِ ۖ وَإِلَيْهِ النُّشُورُ

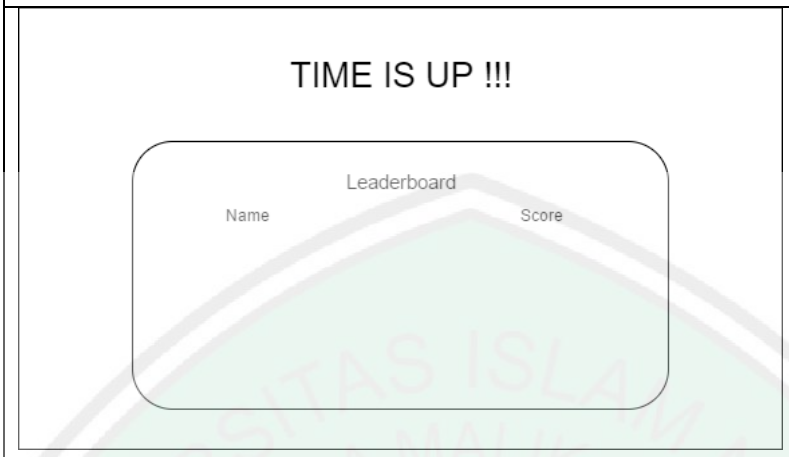
Artinya : Dialah Yang menjadikan bumi itu mudah bagi kamu, maka berjalanlah di segala penjurunya dan makanlah sebahagian dari rezeki-Nya. Dan hanya kepada-Nya-lah kamu (kembali setelah) dibangkitkan.(Q.S . Al-Mulk : 15)

Para pemain diharuskan mengumpulkan poin tertinggi dengan cepat karena waktu yang diberikan terbatas dan juga bersaing dengan pemain lain. Jika waktu yang ditentukan sudah berakhir maka pemenang akan ditentukan dengan banyaknya poin yang didapat. *Score* yang di dapat masing masing pemain akan ditampilkan didalam *Leaderboard* yang terlihat oleh semua pemain.

Tabel 3.1 Storyboard Game

Scene	Keterangan
	Menu awal pada *game* dimana para pemain diharuskan untuk mengisi nama, alamat dompet *ethereum*, dan *private key*-nya untuk melihat saldo *ethereum* yang mereka punya.

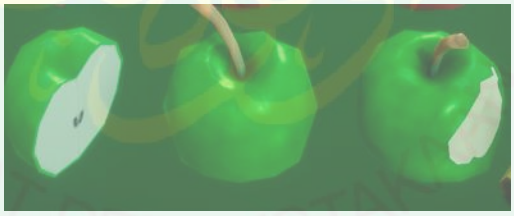
Scene	Keterangan
	Tampilan ketika masing masing pemain berada pada satu ruang sebelum masuk dalam permainan.
	Ketika pemain menekan tombol *start game* maka akan menuju kedalam tampilan awal *game*.
	Tampilan ketika para pemain sedang mencari dan mengumpulkan poin yang berupa apel

Scene	Keterangan
	Gambar ketika waktu sudah habis dan menyelesaikan misinya dan akan menampilkan hasil *score* yang pemain dapatkan

3.3 Deskripsi karakter dan item

1. Item (Apel)

Sebuah *item* didalam *game* yang berfungsi untuk menambah poin untuk para pemain yang mengambilnya. 1 apel bernilai 5 poin ketika pemain dapat mengambilnya



Gambar 3.1 Apel dalam game

2. Pemain

Merupakan tokoh utama yang akan dimainkan didalam *game* ini. Memiliki tugas untuk menyelesaikan misi untuk mendapatkan poin terbanyak yang berupa buah apel yang terbesar didalam game. Pemain dengan poin terbanyak akan menjadi pemenang.

Gambar 3.2 Karakter Pemain

3.4 Desain Interface Game

Interface merupakan tampilan desain dari sebuah aplikasi yang berfungsi sebagai perantara antara *user* dengan aplikasi tersebut.

a. *Desain Interface Icon*

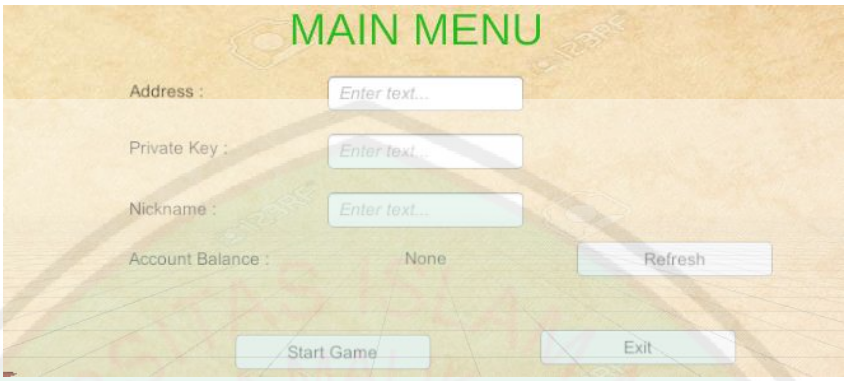
Merupakan sebuah desain yang merupakan ikon identitas dari *game* tersebut. Berikut *icon* dari *game Widow Waterfall*.

Gambar 3.3 icon game

b. Desain *Interface Main Menu*

Merupakan desain *Main Menu* ketika pemain membuka *game Widow Waterfall*. Di dalam tampilan *main menu* tersebut pemain diharuskan mengisi beberapa input yang berguna untuk menampilkan informasi yang dimiliki pemain tersebut yaitu *Address* dompet *Ethereum*, *Private Key Ethereum*,

nickname. Dibawah dari inputan yang harus di isi terdapat 3 tombol yaitu *Start Game*, *Exit* dan *Refresh*



Gambar 3.4 Desain Interface Menu utama

c. Desain *Leaderboard Score*

Fungsi dari *leaderboard* adalah untuk menampilkan hasil *Score* yang diperoleh oleh para pemain.

Gambar 3.5 Desain Leaderboard Score pemain

d. Desain Interface Lobby Game

Merupakan desain Lobby Game ketika Player berada pada satu Ruang (room). Lobby game dibutuhkan karena untuk mempertemukan semua pemain dalam satu Ruang dan pada penelitian ini Game Widow Waterfall bertipe Multiplayer Online.

Gambar3.6 Desain Interface Lobby Game

e. Desain *Interface* Permainan

Merupakan Gambaran Tampilan bagaimana *game* tersebut akan dimainkan oleh pemain. Di dalam *game* ini pemain diharuskan untuk berlomba-lomba menyelesaikan misi sehingga mendapatkan *Score* tertinggi dan mengalahkan pemain lain.

Gambar3.7 Desain Latar Permainan

3.5 Perancangan Data Sharing

Pada Penelitian *Game Widow Waterfall* menggunakan *Photon* 2 dan *Blockchain* sebagai *Platform Networking Engine*. *Photon Networking* sendiri

memiliki banyak cabang produk yang bisa digunakan untuk berbagai macam device dan platform. Produk yang akan digunakan dalam penelitan ini adalah *PUN* (Photon Unity Networking).*PUN* menggunakan server pusat dari *Photon Networking* Untuk mengatur arus data sharing yang berjalan. Jaringan *Blockhchain* yang digunakan pada penelitian ini adalah jaringan *Blockchain Ethereum*. Karena sifat dari jaringan *Ethereum* adalah jaringan publik dan *peer to peer*, maka semua komputer yang terhubung akan mengetahui data apa yang sedang di bagikan dalam jaringan tersebut. Fungsi *PUN* dan *Ethereum Blockchain* dalam *Game* ini adalah untuk memungkinkan bermain *Game Widow Waterfall* dengan *Multiplayer Online* dan digunakan juga sebagai *Data Sharing* antara sistem dan pemain. Kombinasi dari Multiplayer Online degan *Photon Unity Networking (PUN)* dan *Blockchain Ethereum* dimana ketika permainan dimulai, segala pergerakan dan Interaksi yang dilakukan oleh pemain lain akan di sinkronisasikan dengan *Photon Network* dan setiap *score* yang dihasilkan pemain akan disimpan dan di proses dengan *Blockchain Ethereum*

Data yang dibagikan (Sharing) dan output dari data *sharing* dalam *Game Widow Waterfall* adalah *Score* pemain. *Score* yang dimaksud dalam game ini merupakan semua poin yang didapat oleh pemain ketika pemain mengambil objek berupa Buah Apel yang tersebar dalam game.

Gambar 3.8 alur data score

Untuk perhitungan poin sistem *score* yang akan ditampilkan pada *leaderboard game* adalah sebagai berikut :

$$PlayerPointScore = Point / Time \quad (3.1)$$

Definisi :

PlayerPointScore : Hasil poin yang didapat pemain

Point = Jumlah poin yang dikumpulkan

Time : waktu yang dibutuhkan pemain untuk mengumpulkan poin

Dari hasil *PlayerPoint* yang didapat oleh semua pemain, kemudian sistem akan mencari nilai terbesar dengan menggunakan rumus berikut :

$$Score = MAX\ (AllPlayerPointScore) \quad (3.2)$$

Definisi :

Score : hasil perhitungan semua poin pemain

MAX : Mencari nilai tertinggi

AllPlayerPoint: semua poin yang didapat para pemain

Rumus yang digunakan adalah dengan mencari nilai *MAX* dari semua poin yang didapat para pemain. Untuk Setiap buah Apel yang dikumpulkan pemain adalah =+5 poin. Poin yang didapatkan lalu disimpan oleh sistem yang kemudian di tampilkan kembali kepada pemain dalam bentuk *Leaderboard Score* tertinggi antar pemain. Kemudian untuk menghindari kesamaan poin masing masing pemain,

maka digunakanlah variabel waktu (time) sebagai pembeda sekaligus penentu Score tertinggi dalam leaderboard.

Tabel 3.2 contoh perhitungan Score

Pemain	Point	Time	PlayerPointScore
Player 1	50	10,1 detik	4,9
Player 2	30	12 detik	2,5
Player 3	50	10 detik	5

Pada contoh perhitungan **tabel 3.2** terdapat 3 orang pemain yang masing masing memiliki poin tersendiri. Kemudian digunakan rumus *MAX* untuk mencari nilai tertinggi dari PlayerPointScore maka hasilnya seperti **tabel 3.3** dibawah :

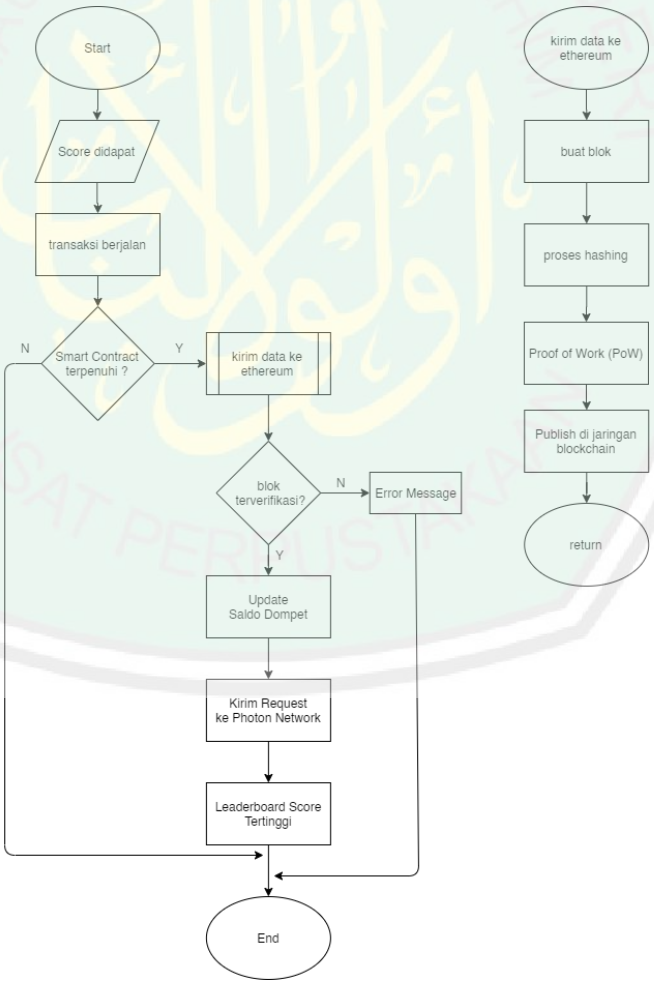
Tabel 3.3 Hasil Perhitungan MAX

Pemain	Point	Time	Score
Player 3	50	10 detik	5
Player 1	50	10,1 detik	4,9
Player 2	30	12 detik	2,5

Hasil perhitungan MAX pada **tabel 3.3** menunjukkan bahwa Player 3 mendapatkan Score tertinggi dari 2 pemain lain. meskipun Player 1 dan Player 3 memiliki Point yang sama, tetapi hasil *Score* mereka berbeda dikarena perbedaan waktu yang mereka dapatkan Dengan ditampilkannya data *sharing* yang berupa nilai *score* tertinggi dalam *Leaderboard Game*, akan memunculkan sportifitas persaingan bagi pemain lain dan game terasa mengasikkan karena pemain lain akan mencari poin sebanyak mungkin untuk mengalahkan *score* tertinggi.

3.6 Blockchain dalam Game

Dalam *game Widow Waterfall* teknologi *Blockchain* digunakan sebagai penyimpanan data *Score* para pemain. Para pemain akan mengumpulkan poin yang tersebar dalam game, kemudian poin tersebut diambil dan disimpan di dalam jaringan *Ethereum blockchain* dalam bentuk sebuah transaksi, *Ethereum Blockhchain* digunakan selain karena hanya jaringan *Blockchain* dari *Ethereum* saja yang mendukung integrasi dalam *Game Engine Unity* 3D dengan menggunakan *library Nethereum*, juga sifat jaringan *Ethereum* yang dapat di program sehingga dapat disesuaikan dengan kebutuhan para pengembang.



Gambar 3.9 Blockchain untuk Sistem Game

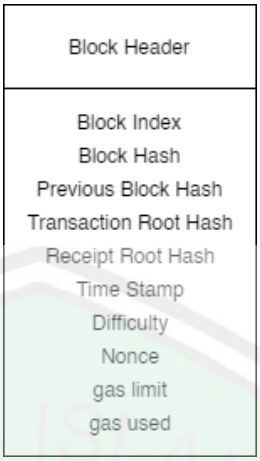
Blockchain juga mendukung transparasi serta penyimpanan data jangka panjang pada jaringannya, artinya data tersebut bisa dilihat oleh semua komputer yang terhubung ke jaringan selama masih ada komputer yang terhubung pada jaringan tersebut. Keamanan data tetap terjaga karena adanya proses enkripsi yang meliputinya. tidak seperti jaringan server pusat pada umumnya yang hanya bisa mengakses data ketika menggunakan jaringan server pusat tersebut. Berikut alur penerapan teknologi *Blockchain* dalam data *sharing* Sistem *Score* pada *game Widow Waterfall* :

a. Proses terbentuknya *Blockchain* karena kesepakatan transaksi antara pihak A dan pihak B dimana kesepakatan awal tersebut membentuk yang bernama *Smart Contract. Smart Contract* berisi sebuah kode komputer yang sudah memenuhi syarat kesepakatan transaksi antara kedua belah pihak.

b. Ketika syarat yang berada di dalam *Smart Contract* sudah terpenuhi maka transaksi pertama akan terjadi.

c. Syarat Tersebut adalah ketika pemain sudah mengumpulkan semua *Score* atau waktu habis, maka Transaksi terjadi.

d. Proses pembuatan blok dilakukan setelah smart contract berjalan, gambar dibawah merupakan Struktur dari blok yang akan di publikasi ke jaringan Blockchain Ethereum :



Gambar 3.10 Struktur Blok Ethereum

Penjelsan gambar diatas sebagai berikut :

1. *Block Index*: nomor *index* dari block
2. *Block Hash : Hash* dari block *header*
3. *Previous Block Hash* : nilai *Hash* dari *block* sebelumnya
4. *Transaction Root Hash* : nilai *Hash* yang dibuat setiap eksekusi transaksi
5. *Receipt Root Hash* nilai *Hash* dari detail transaksi yang dilakukan
6. *TimeStamp* : Waktu dari pembuatan blok
7. *Difficultly* : tingkat kesulitan untuk menambangnya.
8. *Nonce* : nilai integer yang ditentukan oleh penambang untuk menentukan nilai *Hash* yang valid.
9. gas *limit* : batas gas yang di tetapkan untuk *blok* tersebut
10. *gas used* : jumlah gas yang digunakan oleh semua transaksi di blok tersebut

e. Ketika Transaksi awal terjadi, sistem akan menggunakan hashing *SHA*-256 untuk dimasukkan di dalam transaksi. *SHA*-256 merupakan algoritma *hashing* yang akan mengubah sebuah inputan menjadi pesan 256-*bit*. Proses awal dari hashing *SHA*-256 adalah *Message Padding* dimana penyisipan angka 1 dan

penambahan *bit* 0 untuk membuat pesan kongruen dengan 448 *modulo* 512. Kemudian tahap kedua adalah *Parsing* yang merupakan proses di baginya sebuah pesan yang sebelumnya sudah di *padding*, menjadi N buah *block* 512 *bit*. Proses selanjutnya pemecahan pada masing masing blok menjadi 16 buah *word* 32 *bit* yang diperluas menjadi 64 *word* yang disebut sebagai *Message Expansion*. 64 *word* yang sudah terbentuk kemudian diberi label, lalu diproses dengan fungsi *SHA*-256 yang akan mendapatkan 8 variabel yang diberi nilai awal untuk masing masing fungsi. Hasil akhir proses *hashing SHA*-256 diperoleh dari penggabungan 8 variabel yang sudah dikomputasi.

f. Transaksi akan disimpan di blok awal jika tidak terjadi transaksi baru.

g. Jika transaksi baru terbentuk, maka akan dilakukan lagi proses *hashing SHA*-256 yang nantinya akan terbentuk blok baru.

h. Transaksi dalam hal ini adalah nilai *Score* player yang dikirim kedalam jaringan *Blockchain*.

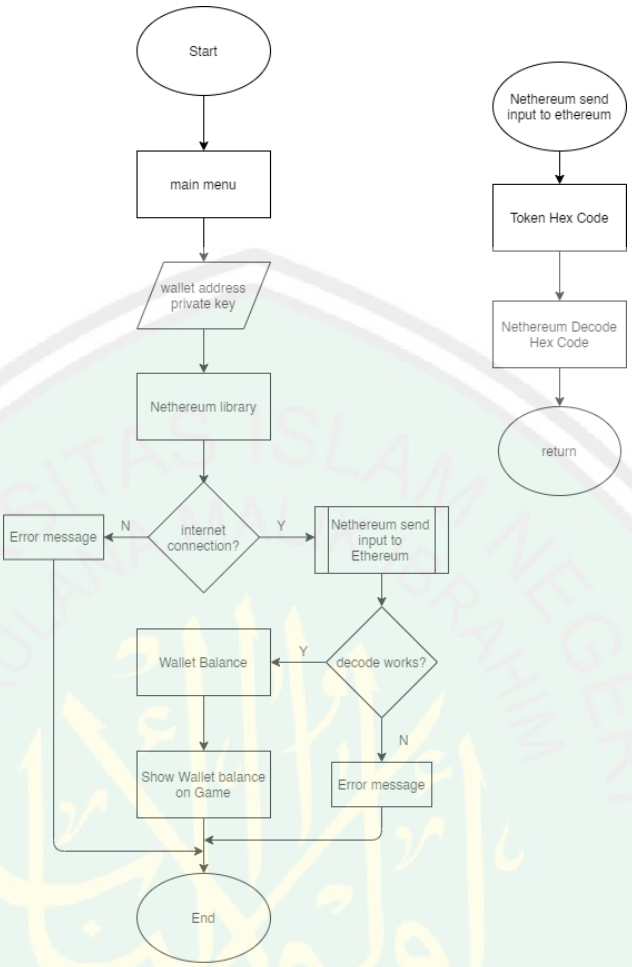
i. Data sudah tersimpan dalam jaringan *Blockchain Ethereum*

j. Transaksi dikatakan berhasil jika sudah terverifikasi dalam jaringan *blockchain*.

Gambar 3.11 Contoh transaksi berhasil

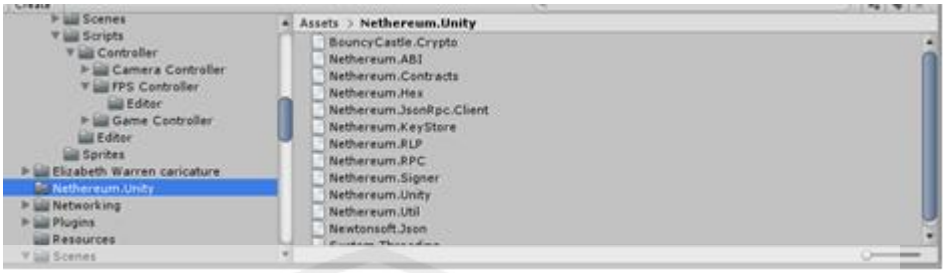
Gambar diatas merupakan contoh transaksi yang berhasil dengan alamat transaksi 0x41ab802efc29fxx dengan Time Stamp 13 hari 6 jam dengan biaya transaksi 20 *Gwei*, jumlah *maximum gas* yang dibayar sebesar 8.000.000 dengan nilai *nonce* transaksi 3.

Dalam Game, *Blockchain* selain digunakan untuk penyimpanan data *Score* pada *Game Widow Waterfall*, Juga digunakan dalam mengakses Informasi data sisa saldo dompet penggunanya. Gambar dibawah menunjukkan alur untuk mengakses informasi data sisa saldo pemain :

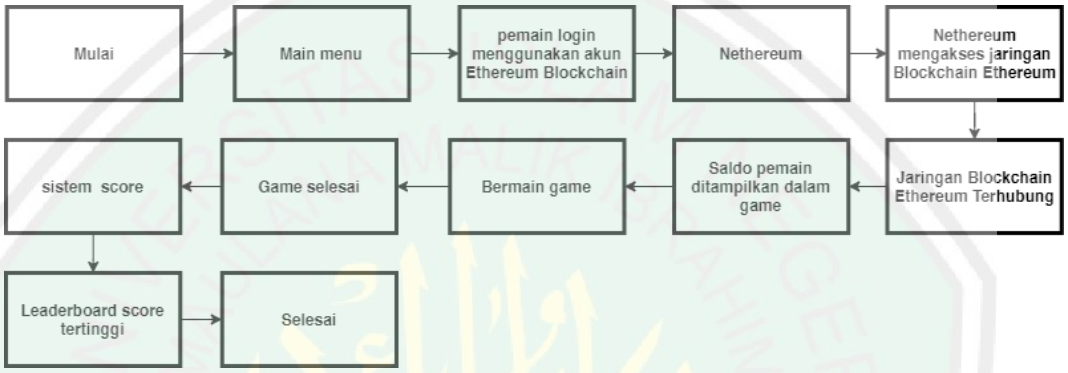


Gambar 3.12 Akses Informasi Wallet Balance

Ketika berada pada *main menu game*, pemain akan menggunakan informasi *Wallet Address* dan *Private key Wallet* untuk menampilkan sisa saldo dompet yang mereka miliki dalam game. *Library Nethereum* digunakan untuk mengakses informasi *Wallet Address* dan *Private key* pengguna jaringan *Blockchain* dengan menggunakan jaringan internet. Kemudian sisa saldo akan ditampilakan sehingga pemain dapat mengetahui berapa banyak mata uang digital eter (ETH) dalam *game*. Karena ketika sistem mengirim data *Score* pada jaringan *blockchain*, saldo dari pemain akan berkurang karena proses transaksi yang terjadi.



Gambar 3.13 Library nethereum pada unity



Gambar 3.14 Alur Blockchain ketika game terpasang

Pada **Gambar 3.14** adalah alur teknologi *blockchain* yang digunakan pada game widow waterfall ketika game sudah terpasang. **Gambar 3.14** juga menggambarkan penyesuaian *game Widow Waterfall* dengan karakter dari jaringan *Ethereum Blockchain* yang membutuhkan koneksi internet untuk mengaksesnya. Ketika pada main menu game pemain akan *login* menggunakan akun *ethereum* mereka yang nantinya akan menampilkan sisa saldo yang mereka miliki. ketika informasi sisa saldo pemain sudah dapat ditampilkan dalam game, maka jaringan *Blockchain Ethereum* sudah terhubung dengan bantuan *library Nethereum* yang sudah di integrasikan dalam *game*.

Pemain kemudian melakukan misi dalam game yaitu mengumpulkan poin sebanyak mungkin. Nantinya poin tersebut akan masuk dalam sistem *score* untuk penentuan Score terbanyak dari para pemain.dan pengiriman data *Score* dengan

menggunakan sisa saldo pemain ke dalam jaringan *Blockchain Ethereum* sebagai transaksi yang valid.

3.7 Perancangan Pengujian

Skenario Pengujian dilakukan untuk mengetahui bahwa jaringan *Ethereum blockchain* dan Sistem yang akan dibuat berjalan dengan baik dan sesuai rencana yang ditulis. Pengujian juga diperlukan untuk melihat performa jaringan *Ethereum blockchain* ketika digunakan dalam *Game Widow Waterfall*.

3.7.1 Pengujian Koneksi

Pengujian Koneksi dimaksudkan apakah Jaringan *Ethereum Blockhcain* sudah terhubung dengan *Game Engine Unity* 3D dan *Game Widow Waterfall* berjalan dengan baik. Pengujian akan dilakukan dengan terhubung dalam koneksi intenet. Hasil pengujian pada **tabel 3.4** akan dibahas pada bab selanjutnya.

Tabel 3.4 Pengujian Koneksi

No	Skenario Pengujian	Hasil Pengujian
1	Pada Console Unity menampilkan pesan Koneksi Berhasil, serta Token Hex Code dari saldo Pemain	
2	Ketika Pemain menekan Button Refresh yang berada di Game, maka akan menampilkan Informasi Sisa saldo pemain	

3.7.2 Pengujian Sharing Data

Pengujian Sharing data dimaksudkan apakah data yang sudah di *share* (dibagikan) dalam sistem dapat digunakan dan ditampilkan dengan baik. Hasil pengujian pada **tabel 3.5** akan dibahas pada bab selanjutnya.

Tabel 3.5 Pengujian Sharing data

No	Skenario Pengujian	Hasil Pengujian
1	Pemain bisa melihat sisa saldo dompet mereka dalam game	
2	Score tercatat sebagai transaksi valid dalam jaringan Ethereum Blockhchain	
3	Para Pemain bisa melihat hasil Score yang mereka dapat dalam Game	

3.7.3 Pengujian Transaction Speed (Kecepatan Transaksi)

Pada proses pengujian kecepatan transaksi dimaksudkan untuk melihat seberapa cepat waktu yang dibutuhkan dalam validasi sebuah transaksi. Adapun hal hal yang mempengaruhi kecepatan validasi sebuah transaksi adalah Jumlah *block* dalam sebuah jaringan, *gas price* atau *gas limit*, dan bobot datanya. Semakin Banyak blok yang berada dalam jaringan *ethereum blockchain*, maka semakin lama juga proses validasinya. *gas price* atau *gas limit* juga berpengaruh dalam kecepatan pembuatan nilai *hash*, semakin besar *gas price* dan *gas limit* yang diberikan maka semakin cepat juga proses validasi transaksi sebuah transaksi. Semakin besar bobot data yang akan di kirim dalam jaringan ethereum blockhcain maka akan semakin

lama juga proses validasnya. Dari ketiga variabel tadi yaitu banyaknya blok, bobot data dan *gas price* atau *gas limit*, karena tidak memungkinkan untuk mengubah bobot data dan banyaknya blok yang ada, maka digunakanlah pengaturan jumlah *Gas limit* dalam penelitian *game widow waterfall*. (Arif et al., 2020) menyatakan bahwa rumus untuk menghitung kecepatan proses validasi sebuah transaksi adalah sebagai berikut :

$$\bar{T} = \frac{1}{n} \sum_{i=1}^{n} Ti \qquad (3.3)$$

Keterangan :

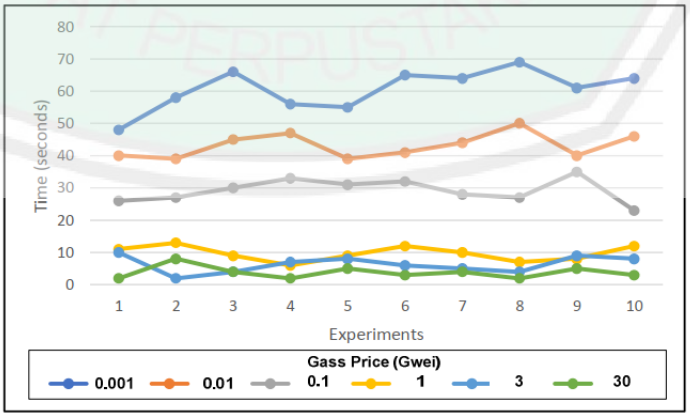
$\bar{T}$ = rata – rata waktu transaksi

Ti = Waktu yang dibutuhkan untuk memproses sebuah transaksi

n = Banyaknya percobaan

i = transaksi

Kemudian setelah didapatkannya rata rata waktu transaksi menggunakan rumus diatas maka akan menghasilkan grafik perbandingan waktu berdasarkan *gas price* atau *gas limit* yang telah ditentukan seperti gambar dibawah ini :



Gambar 3.15 Grafik Perbandingan
Sumber : (Arif et al., 2020)

3.7.4 Estimasi Biaya Pengembangan

Pengujian Estimasi Biaya Pengembangan dimaksudkan untuk menghitung estimasi biaya yang dikeluarkan untuk perawatan atau pengembangan pada *game widow waterfall*. Hal tersebut didapatkan dengan cara menghitung biaya setiap transaksi yang dilakukan. Biaya transaksi pada *Ethereum Blockchain* ditentukan oleh jumlah *Gas* yang digunakan saat pembuatan transaksi. Semakin banyak *gas* yang digunakan, maka akan semakin besar juga biaya yang dikeluarkan untuk setiap transaksi yang dilakukan. Berikut perhitungan yang digunakan dalam menentukan estimasi biaya yang dikeluarkan pada setiap transaksi yang dilakukan.

$$T_{Cost} = Gas\ Price\ x\ Estimate\ Gas\ used \quad (3.4)$$

Keterangan :

T_{Cost} = Biaya setiap transaksi

Estimate Gas Used = jumlah gas untuk proses validasi

Gas price = biaya setiap gas yang digunakan

T_{Cost} tersebut kemudian akan dikonversikan dari Gwei ke dalam Rupiah (Rp). Kemudian dilakukan perhitungan estimasi skenario *player* maupun *developer* melakukan transaksi sebanyak 15 kali yang akan menghasilkan perbandingan rata-rata biaya yang dikeluarkan dalam per tahun yang akan dibahas pada bab selanjutnya.

BAB IV

PENGUJIAN DAN EVALUASI

Pada bab ini dijelaskan terkait implementasi dan evaluasi terhadap sistem yang telah dirancang sebelumnya. Hal tersebut dilakukan untuk membuktikan efektifitas sistem data *sharing* untuk score pada *game widow waterfall* menggunakan *Ethereum Blockchain.*

4.1 Implementasi

Pada tahap ini *Ethereum Blockchain* akan di Implementasikan pada *Game Widow Waterfall* untuk membuat Sistem Data *Sharing Score*. Dalam bab ini juga dibahas proses implementasi serta hasil yang akan didapat. Skenario Pengujian pada bab III digunakan sebagai acuan keberhasilan pada penelitian ini. Implementasi dilakukan untuk menerapkan langkah-langkah yang sudah ditentukan sebelumnya.

4.2 Script

Komponen paling utama untuk membangun sebuah sistem data *sharing* pada *game widow waterfall* menggunakan *Ethereum Blockchain* adalah *script* atau *Code.Visual Studio Code* digunakan sebagai *Script* editor implementasi *Ethereum Blockchain* dalam Game. Bahasa pemrograman *C#* digunakan untuk menulis baris program dalam sistem data *sharing* pada *game widow waterfall* menggunakan *Ethereum Blockhchain*.

4.3 Perangkat lunak yang digunakan

Perangkat lunak yang digunakan untuk pembuatan sekaligus pengujian game widow waterfall sebagai berikut :

Operating System	Windows 10 Home
Game Engine	Unity3D 2018 v3.14f1
Scrip Editor	Visual Studio Code 1.48.2
UI Editor	Adobe Photoshop CC 2017
3D Modeller	Blender 3D v2.8
Ethereum Wallet	Metamask

4.4 Perangkat Keras yang digunakan

Perangkat Keras yang digunakan (*Hardware*) dalam pengujian penelitian game widow waterfall menggunakan *Ethereum Blockchain* sebanyak 5 Hardware. Hal tersebut mengingat *game multiplayer* membutuhkan pengujian dari beberapa *device* yang berbeda karena untuk mengetahui apakah aplikasi benar benar dapat berjalan dengan baik atau tidak. Berikut 5 perangkat keras yang digunakan dalam penelitian game widow waterfall :

1. Perangkat 1

Gadget	Laptop
Merk	Asus
Operating Sytem	Windows 10 Home
CPU	Intel Core i7-6700HQ
GPU	Nvidia GTX 950M

RAM	16 GB DDR4

2. Perangkat 2

Gadget	Laptop
Merk	Hp
Operating Sytem	Windows 10 Home
CPU	Intel Core i3-5005U
GPU	AMD Radeon R5 M330
RAM	6 GB DDR3

3. Perangkat 3

Gadget	Laptop
Merk	Asus
Operating Sytem	Windows 10 Home
CPU	Intel Core i3-5005U
GPU	Nvidia GeForce 930M
RAM	4 GB

4. Perangkat 4.

Gadget	Smartphone
Merk	Samsung Galaxy A50s
Operating Sytem	Andorid 9.0
CPU	Exynos 9611
GPU	Mali-G72 MP3
RAM	4 GB

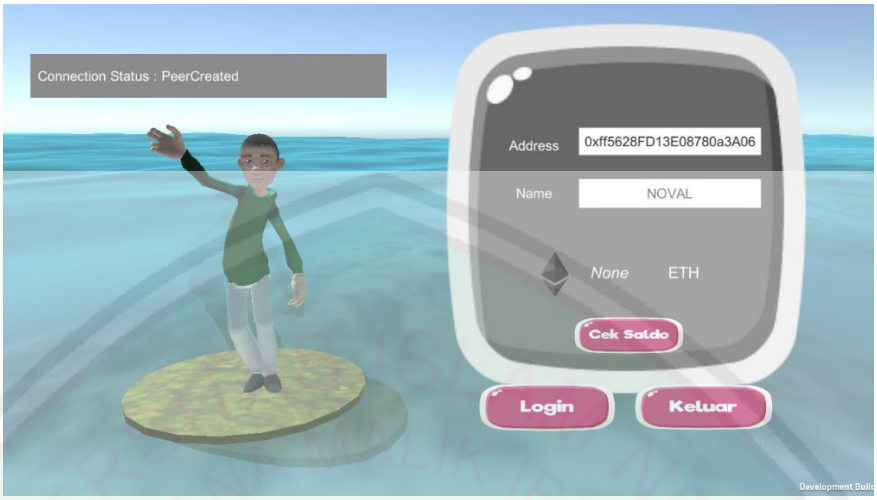
5. Perangkat 5

Gadget	Smartphone
Merk	Samsung Galaxy A6+
Operating Sytem	Andorid 8.0
CPU	Qualcomm Snapdragon 450
GPU	Adreno 506
RAM	3 GB

4.5 Implementasi Skenario Game

Implementasi Skenario *Game* adalah penerapan tampilan dari *Game* yang telah dibuat menggunakan metode yang dibahas pada BAB 3 Berikut skenario *game* widow waterfall ketika game berjalan.

1. Login Main Menu



Gambar 4.1 Login Main Menu

Gambar diatas merupakan tampilan *Login Main* menu pada g*ame widow waterfall*. Pemain diharuskan mengisi nama sebagai *nickname* ketika game berjalan serta mengisi alamat dompet *ethereum* untuk mengetahui sisa saldo *ethereum* yang nantinya akan digunakan sebagai biaya pengirim data untuk score pada jaringan *blokchain* ketika player menyelesaikan *game widow waterfall*.

2. Lobby Game

Gambar 4.2 Lobby Game

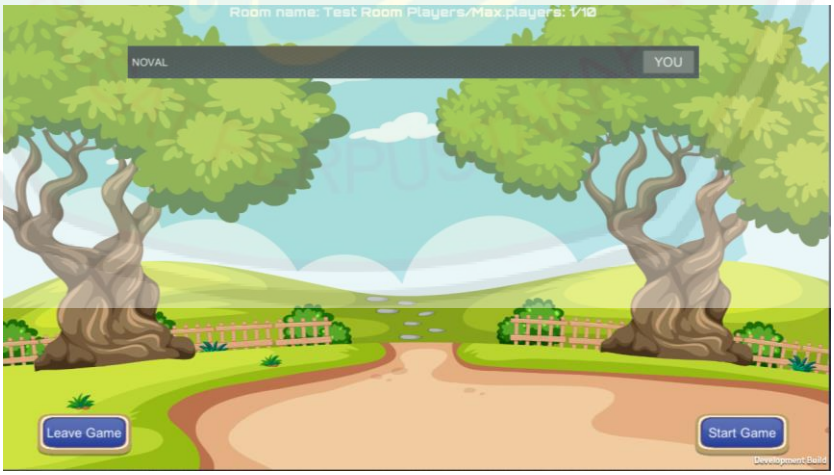
Gambar diatas adalah *lobby game* sebelum para pemain masuk dalam satu *Room* (ruang tunggu game). Setelah pemain berhasil *login*, maka akan diarahkan

ke *UI lobby game*. Di didalam *lobby game* terdapat 3 pilihan, yaitu *Create Room*, *Join Random Room*, dan *Show Room list*.

3. Create Room

Gambar 4.3 Create Room

Gambar 4.3 merupakan tampilan ketika *player* menekan tombol *create Room* yang kemudian akan diarahkan ke tampilan seperti diatas. *Player* diharuskan mengisi nama *Room* dan jumlah maksimal *player* yang bisa masuk dalam satu *room*. Kemudian menekan tombol *Create* untuk membuat *room*.

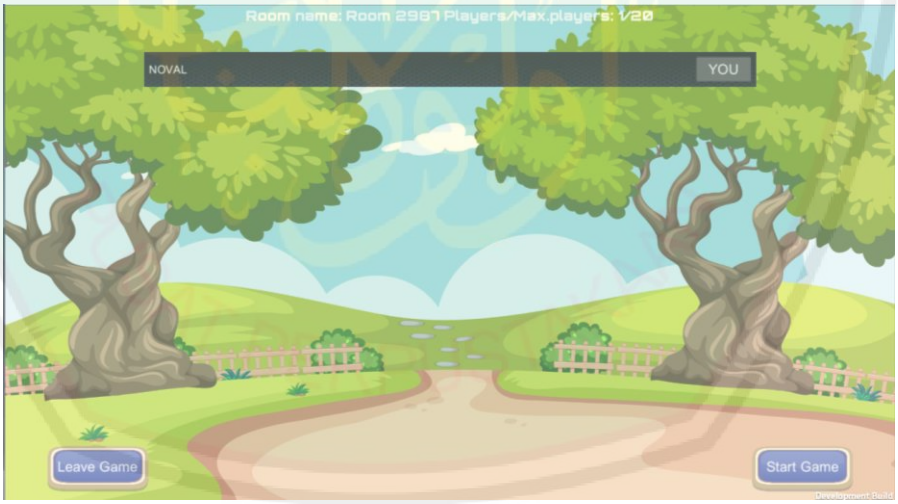


Gambar 4.4 Masuk di dalam Room

4. Join Random Room

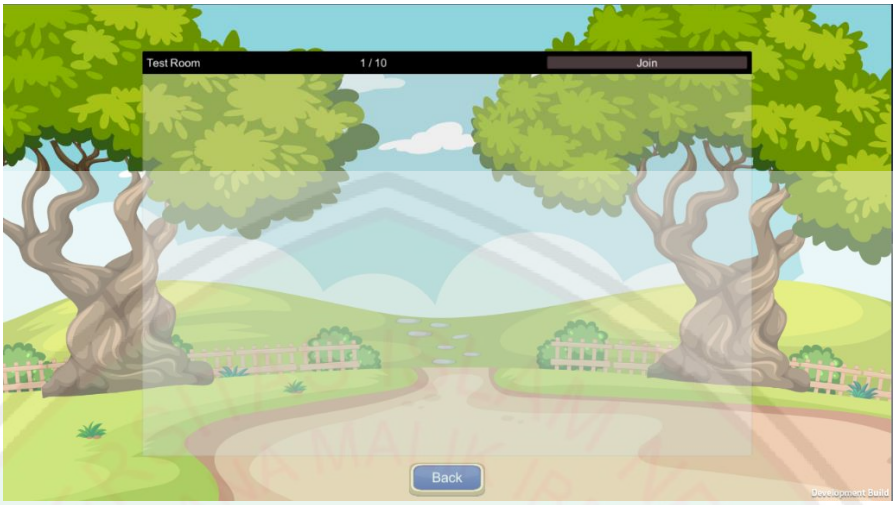
Gambar 4.5 Join Random Room

Ketika *Player* menekan tombol *Join Random Room* maka secara otomatis akan bergabung dengan *Room* yang ada. dan ketika tidak ada *room* yang terbuat, maka *Join Random Room* akan membuat *Room* sendiri secara otomatis.



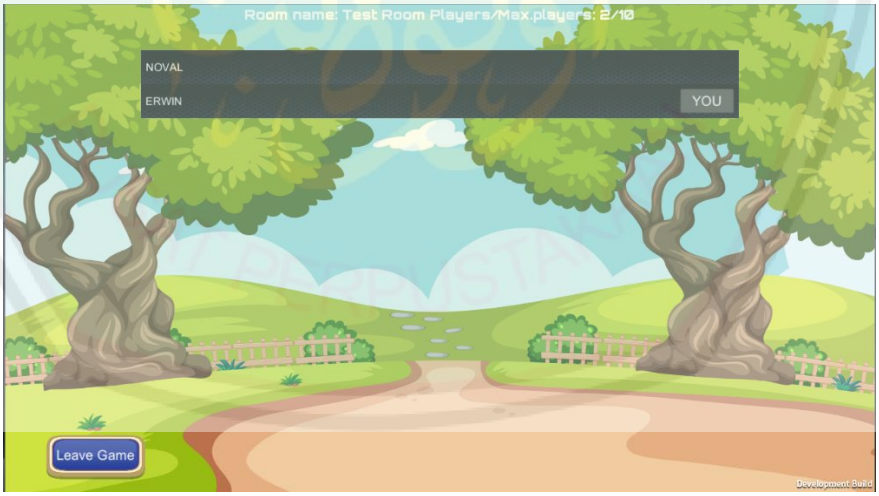
Gambar 4.6 Masuk dalam Random Room

5. Show Room List



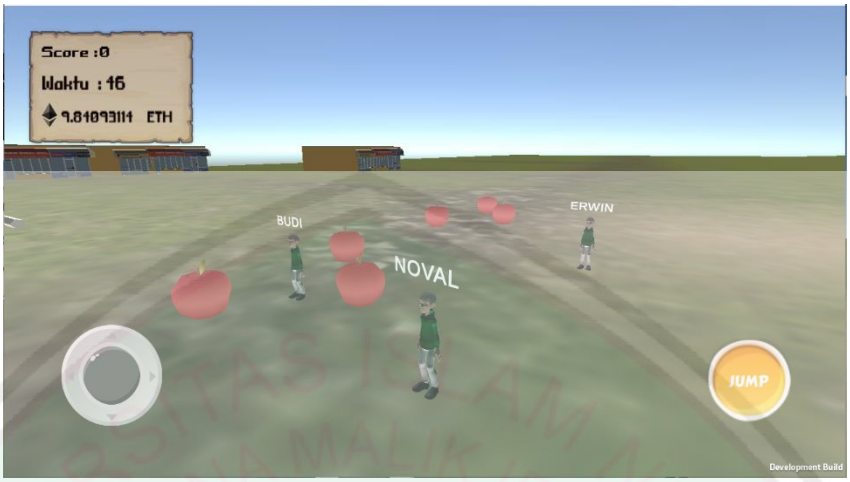
Gambar 4.7 Show Room list

Gambar 4.7 diatas merupakan tampilan ketika *player* menekan tombol *Show Room List*. Didalamnya terdapat informasi mengenai nama dari *Room* tersebut, jumlah *player* yang telah bergabung dan maksimal *player* yang bisa ditampung dalam *Room* tersebut serta tombol *join* untuk bergabung dengan *Room* tersebut.



Gambar 4.8 Masuk dalam Room yang telah dibuat

6. Tampilan ketika bermain Game



Gambar 4.9 Tampilan ketika berada dalam permainan

Ketika *Player* sudah masuk dalam permainan, akan tampil di setiap pemain nama atau *nickname* dari setiap pemain, saldo *Ethereum* yang mereka miliki, sisa waktu permainan dan score yang mereka dapat. Semua hal tersebut ditunjukkan pada gambar 4.9.

7. Tampilan Leaderboard Score

Gambar 4.10 Tampilan Leaderboard

Setelah semua *player* menyelesaikan game dan telah mengumpulkan buah apel yang tersebar didalam game untuk mendapatkan poin, maka selanjutnya adalah menampilkan poin yang diperoleh dari masing masing *player*. Pada Gambar 4.10

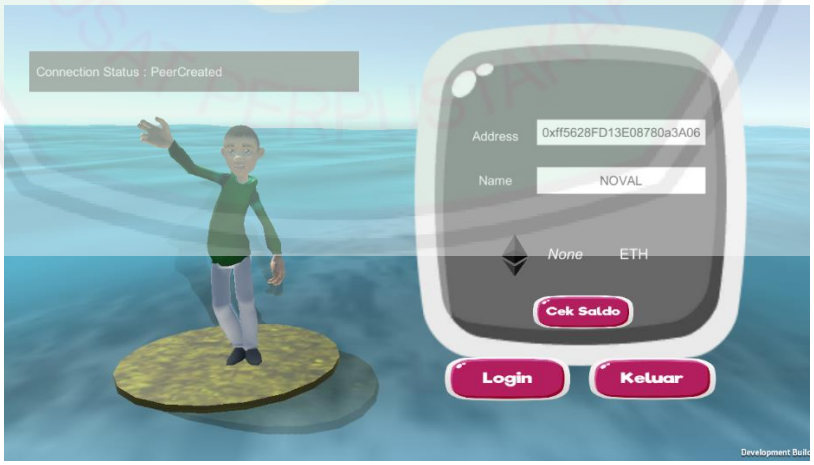
diatas merupakan tampilan *Leaderboard score* dari para pemain yang telah mengumpulkan *score*. *Player* dengan poin yang tertinggi akan berada pada urutan paling atas dan seterusnya sampai *player* dengan poin terendah.

4.6 Uji Coba

Uji Coba dilakukan untuk mengetahui performa sistem yang telah dibuat apakah akan berjalan dengan baik. Pengujian akan melihat bagaimana Jaringan *Ethereum Blockchain* akan terhubung dan berjalan dengan baik dalam game, serta bagaimana data sharing yang score yang akan menampilkan hasil score pemain sesuai urutan posisi poin tertinggi.

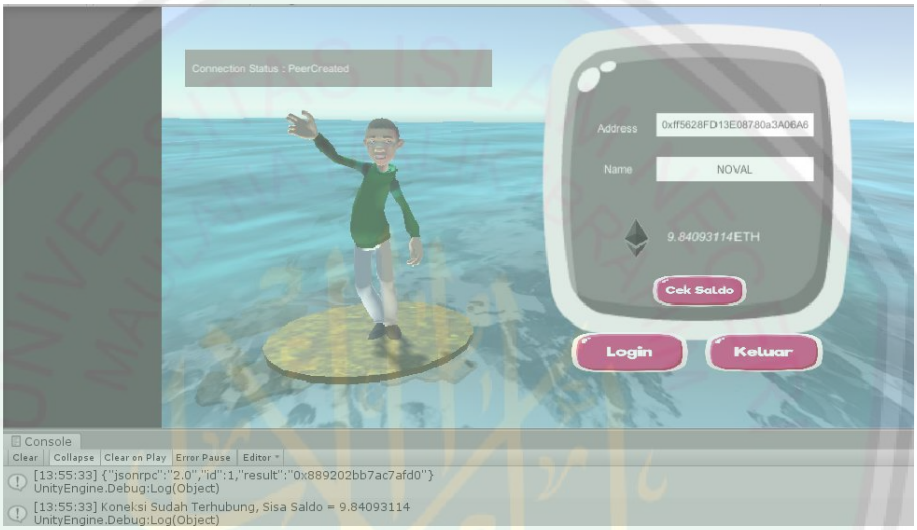
4.6.1 Uji Koneksi Ethereum Blockchain dengan Unity

Pada pengujian ini, dilakukan untuk mengetahui apakah jaringan *Ethereum Blockchain* sudah terhubung di dalam game. Hal terebut berguna untuk menampilkan sisa saldo (*balance*) *Ethereum* yang dimiliki *player*. Saldo tersebut nantinya akan digunakan untuk mengirim dan menyimpan data score yang diperoleh player ke dalam jaringan *Ethereum Blockchain.*



Gambar 4.11 Main Menu

Pada **Gambar 4.12** *Player* memasukkan *Address* dari dompet *Ethereum* yang dimiliki, kemudian menekan button Cek saldo**.** Apabila koneksi jaringan *Ethereum Blockchain* sudah terhubung maka di dalam *console unity* akan manmpilkan pesan "Koneksi Sudah terhubung, Sisa Saldo = 9.84093114" dan tampil sisa saldo yang dimilik di dalam game seperti pada **Gambar 4.13.**



Gambar 4.12 Koneksi sudah Terhubung

Baris *script* yang digunakan dalam mengkoneksikan jaringan *Ethereum Blockchain* pada penelitian *Game Widow Waterfall* sebagai berikut:

```
public void GetAccountBalance(){
    StopCoroutine(getAccountBalanceCoroutine);
    getAccountBalanceCoroutine = GetAccountBalanceCoroutine();
    StartCoroutine(getAccountBalanceCoroutine);
}
2 references
public IEnumerator GetAccountBalanceCoroutine(){
    var getBalanceRequest = new EthGetBalanceUnityRequest(networkUrl);
    yield return getBalanceRequest.SendRequest
    (InputAccountText.text, Nethereum.RPC.Eth.DTOs.BlockParameter.CreateLatest());

    if(getBalanceRequest.Exception == null){
        var balance = getBalanceRequest.Result.Value;
        uiTextEtherBalance.text =
        Nethereum.Util.UnitConversion.Convert.FromWei(balance, 18).ToString("n8");
        Debug.Log("Koneksi Sudah Terhubung, Sisa Saldo = " + uiTextEtherBalance.text);
      }
      else
        {
            Debug.Log ("Koneksi Gagal, Coba Lagi");
            Debug.Log
            ("RW: Get Account Balance gave an exception : " + getBalanceRequest.Exception.Message);
        }

}
```

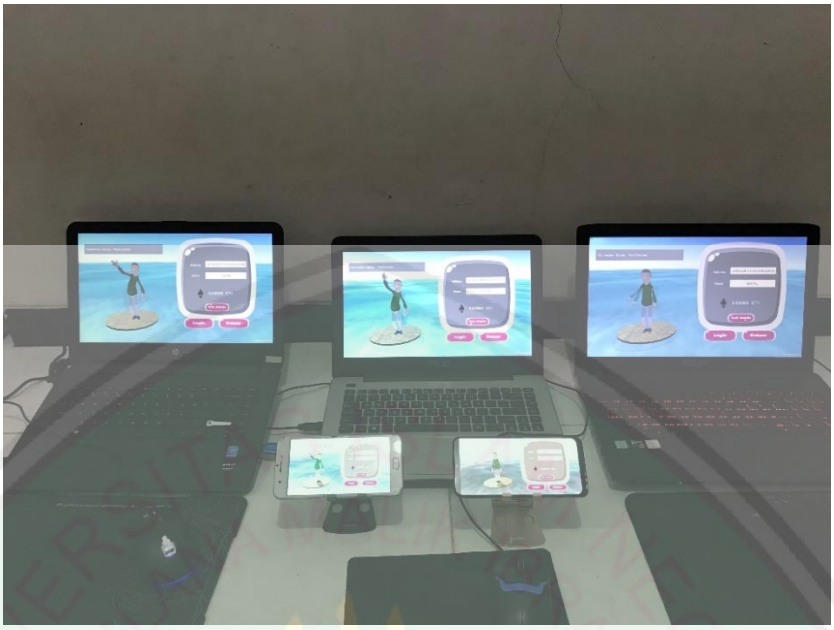
Gambar 4.13 Script Koneksi Ethereum

Pengujian dilakukan dengan menggunakan 5 Perangkat berbeda. Pengujian dinyatakan berhasil ketika berubahnya tampilan informasi saldo dari "*None*" ke "9.84093114 ETH". Angka "9.84093114" merupakan sisa saldo (*balance*) dari dompet *Ethereum* yang sudah dipersiapkan untuk penelitian ini dan sudah dikonversikan dari *Token HexCode* "0x889202bb7ac7afd0" ke dalam bentuk *String* yang bisa terbaca oleh sistem. Berikut data hasil percobaan koneksi dapat dilihat pada **Tabel 4.1**

Tabel 4.1 Uji Koneksi

Test Koneksi	**Perangkat 1**	**Perangkat 2**	**Perangkat 3**	**Perangkat 4**	**Perangkat 5**
Test 1	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Test 2	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Test 3	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Test 4	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Test 5	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Test 6	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Test 7	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Test 8	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Test 9	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Test 10	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil

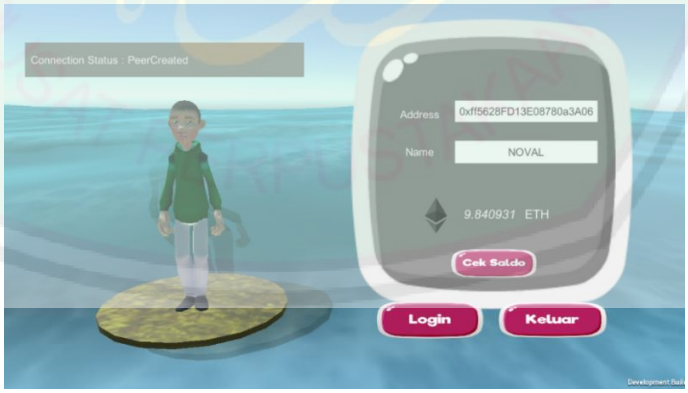
Terlihat pada tabel diatas , pengujian test koneksi *game* dengan jaringan *Ethereum blockchain* dengan 10 kali percobaan pada tiap perangkat menghasilkan tingkat keberhasilan yang tinggi.



Gambar 4.14 Pengujian Koneksi dengan 5 perangkat

4.6.2 Pengujian Login

Pengujian yang selanjutnya adalah pengujian *login* kedalam *game*. Percobaan pengujian dimulai dengan mengisi *nickname* yang akan digunakan didalam *game*. Langkah teresebut dilakukan setelah *player* mengisi *Address* (alamat) dari dompet *Ethereum* yang dimiliki.



Gambar 4.15 Menu Login pada game

Baris *script* yang digunakan Untuk bagian *Login* pada penelitian *Game Widow Waterfall* sebagai berikut:

```
    0 references
    public void OnLoginButtonClick(){
    string playerName = playerNameInput.text;
    if(!string.IsNullOrEmpty(playerName)){
        PhotonNetwork.LocalPlayer.NickName = playerName;
        PhotonNetwork.ConnectUsingSettings();
    }else{
        Debug.Log("playername invalid");
    }
}

//photonCallback
2 references
public override void OnConnected() {
    PhotonNetwork.LoadLevel(1);
}

4 references
public override void OnConnectedToMaster(){
    Debug.Log(PhotonNetwork.LocalPlayer.NickName+ "Login berhasil");
}
```

Gambar 4.16 Script Login

Pengujian dinyatakan berhasil dengan masuknya data *nikcname player* kedalam *server* dan berubahnya tampilan *UI game* yang semula dari *Main Menu* ke *UI Lobby Game* seperti pada gambar dibawah ini :

Gambar 4.17 UI Lobby

Data hasil percobaan *login* yang dilakukan dengan 5 perangkat yang berbeda dapat dilihat pada tabel berikut. Percobaan dinyatakan berhaasil jika *player* bisa *login* ke dalam *Game*.

Tabel 4.2 Uji Login

Pengujian Login	**Perangkat 1**	**Perangkat 2**	**Perangkat 3**	**Perangkat 4**	**Perangkat 5**
Login 1	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Login 2	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Login 3	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Login 4	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Login 5	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Login 6	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Login 7	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Login 8	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Login 9	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Login 10	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil

Dapat dilihat pada tabel diatas bahwa hasil dari uji login pada *game widow waterfall* menggunakan 5 perangkat yang berbeda dengan tiap perangkat melakukan 10 kali percobaan berhasil dengan hasil *player* dapat masuk ke dalam *lobby game.*

Gambar 4.18 Pengujian Login dengan 5 perangkat

4.6.3 Pengujian Create Room

Pada pengujian ketiga ini dilakukan dengan cara membuat *room lobby game* atau *room master*. Pada *game Widow waterfall* pemain dapat membuat *room custom* sesuai keinginan *player* dan membuat *room* secara *otomatis*. Pengujian ini dilakukan setelah *player* sudah masuk dalam *lobby game*.

```
public void OnCreateRoomButtonClicked(){
    string roomName = roomNameInput.text;
    if(string.IsNullOrEmpty(roomName)){
        roomName = "Room " + Random.Range(1000,10000);
    }
 RoomOptions roomOptions = new RoomOptions();
 roomOptions.MaxPlayers = (byte)int.Parse(maxPlayerinput.text);

    PhotonNetwork.CreateRoom(roomName, roomOptions);
}
```

Gambar 4.19 Script Create Custom Room

Pada **Gambar 4.19** merupakan *script* yang digunakan untuk membuat *custom room*. Player diharuskan mengisi apa nama *room* (*Room Name*) yang akan dibuat dan jumlah maksimal *player* (*Max Player*) yang akan masuk dan bermain dalam satu waktu bersamaan. Berikut tampilan *UI Create Custom Room* :

Gambar 4.20 Create Custom Room

Pemain juga dapat membuat *Room otomatis* ketika tidak ditemukannya *room* yang ada pada game atau pencarian *room* tidak ditemukan. Berikut Script yang digunakan untuk pembuatan room otomatis :

```
public override void OnJoinRandomFailed(short returnCode, string message){
    Debug.Log(message);

    string roomName = "Room " + Random.Range(1000, 10000);

    RoomOptions roomOptions = new RoomOptions();
    roomOptions.MaxPlayers = 20;

    PhotonNetwork.CreateRoom(roomName, roomOptions);
}
```

Gambar 4.21 Auto Create Room

Pengujian ini dinyatakan berhasil ketika *Room* berhasil terbuat dan *player* yang membuat *room* tersebut atau *Room Master* dapat masuk ke dalamnya. Saat *Room* berhasil dibuat maka tampilan *UI* dalam *game* akan berubah seperti pada gambar berikut :

Gambar 4.22 Room telah terbuat

Pada gambar di atas merupakan *room* yang telah berhasil dibuat. Dengan *Room Name* = TestRoom dan *Max Player* = 10 serta menampilkan siapa saja yang bergabung dalam *room* tersebut. Hal tersebut menandakan bahwa *room* telah berhasil terbuat dan *Room Master* menunggu *node* (*player* lain) masuk ke dalam *room* yg telah dibuat. Berikut data pengujian *Create Roo*m yang menggunakan 5 perangkat berbeda. Pengujian dikatakan sukses jika perangkat berhasil membuat *Room*.

Tabel 4.3 Uji Create Room

Create Room	Perangkat 1	Perangkat 2	Perangkat 3	Perangkat 4	Perangkat 5
Create 1	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Create 2	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Create 3	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Create 4	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Create 5	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Create 6	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil

Create Room	Perangkat 1	Perangkat 2	Perangkat 3	Perangkat 4	Perangkat 5
Create 7	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Create 8	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Create 9	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Create 10	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil

Dapat dilihat pada tabel diatas bahwa hasil dari uji *create* pada *game widow waterfall* menggunakan 5 perangkat yang berbeda dengan tiap perangkat melakukan 10 kali percobaan berhasil dengan hasil *player* dapat membuat *room* dan masuk kedalam *room game* tersebut.

Gambar 4.23 Pengujian Create dengan 5 perangkat

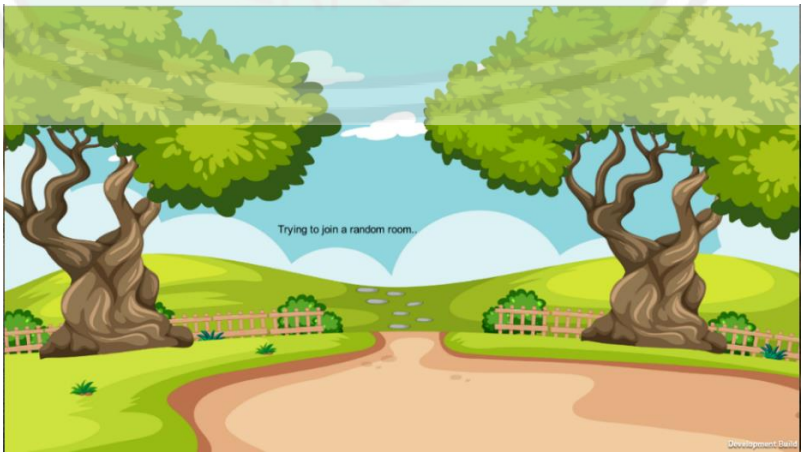
4.6.4 Pengujian Join Room

Pengujian ke empat dilakukan dengan percobaan masuk ke dalam *room* yang telah dibuat oleh *player (node)* lain. proses ini dilakukan setelah *player login* ke dalam *game*. Pengujian ini juga memeriksa apakah *player (node*) satu dengan yang lainnya sudah terhubung. Berikut baris *Script* yang digunakan untuk *Join Room* pada penelitian *game Widow Waterfall*.

```
public void OnJoinRandomRoomButtonClicked(){
    ActivatePanel(JoinRandomRoom_Panel.name);
    PhotonNetwork.JoinRandomRoom();
}
```

Gambar 4.24 Script Join Random Room

Pada penelitian *game widow waterfall*, pencarian room dapat dilakukan dengan 2 cara yaitu secara *random* dan melihat *list room* yang terbuat. Ketika player memilih untuk *join random room* maka sistem akan secara acak memasukkan *player* ke dalam *room player* lain yang terdaftar di *server* pada saat *game* tersebut di mainkan. Hal ini membuat proses join *room* lebih singkat dan *player* tidak perlu bersusah payah untuk mencari dan *join room* yang ada di dalam *game*. Berikut Tampilan *UI* pada *game widow waterfall* ketika proses pencarian *Join Random Room* bisa di lihat pada gambar di bawah ini.



Gambar 4.25 Proses pencarian Join Random Room

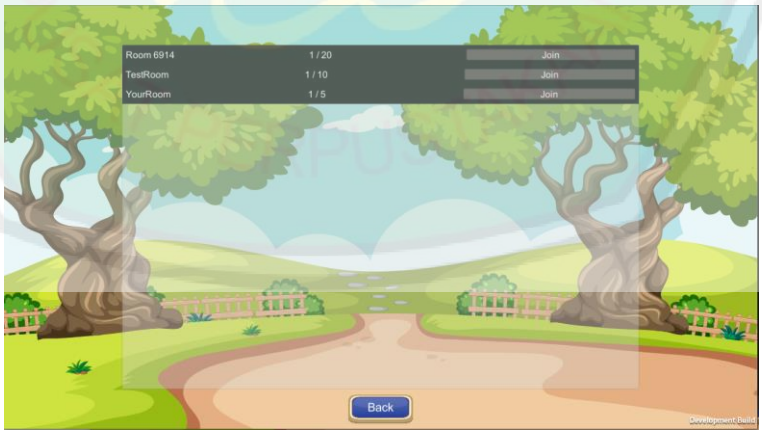
Ada 2 kemungkinan pada proses pencarian *Join Random Room* ketika sudah selesai, pertama *Room Master* (*room host*) di temukan, kedua *Room Master* (*room host*) *tidak ditemukan*. Pada saat *Room master* ditemukan, *player* (*node*) akan secara otomatis masuk kedalam *Room Master* tersebut, tetapi ketika *Room Master* tidak ditemukan, *player* akan secara otomatis melakukan proses *creating random room*.

Cara selanjutnya untuk bergabung dalam *Room Master* (*room host*) yaitu dengan melihat *List Room* yang sudah terbuat. Berikut baris *Script* yang digunakan dalam *List Room* pada penelitian *game widow waterfall*.

```
public void OnShowRoomListButtonClicked(){

    if(!PhotonNetwork.InLobby){
        PhotonNetwork.JoinLobby();
    }

    ActivatePanel(RoomList_Panel.name);
}
```

Gambar 4.26 Script Room List

Pada saat player menekan *button Show Room List* yang berada di *lobby game*, tampilan *UI* akan berubah seperti pada gambar berikut.



Gambar 4.27 Script Room List

Seperti pada **gambar 4.24** merupakan opsi kedua *player* bisa bergabung ke dalam *Room Master* (*room host*). Pada *UI* tersebut terdapat informasi berupa nama

Room yang tersedia, jumlah maksimal player yang dapat bergabung, dan tombol join untuk masuk ke dalam *Room*. Baik dengan cara *Join Random Room* ataupun dengan *Room List*, ketika player berhasil masuk ke dalam room maka tampilan *UI game* akan berubah seperti gambar dibawah.

Gambar 4.28 Bergabung ke Room Host

Berikut data pengujian *Join Random Room* dan *Room List* yang menggunakan 5 perangkat berbeda. Pengujian dikatakan sukses jika perangkat berhasil masuk ke dalam *Room Master* (room host).

Gambar 4.29 join room dari 5 player

Pada gambar diatas menunjukkan bahwa data *sharing nickname* antara player berjalan dengan baik. *Player* dapat melihat *nickname* satu sama lainnya pada satu *room* yang sama.

Tabel 4.4 Uji Join Random Room

Join Room	**Perangkat 1**	**Perangkat 2**	**Perangkat 3**	**Perangkat 4**	**Perangkat 5**
Join 1	Berhasil	Berhasil	Berhasil	Gagal	Berhasil
Join 2	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Join 3	Gagal	Berhasil	Berhasil	Berhasil	Berhasil

Join Room	Perangkat 1	Perangkat 2	Perangkat 3	Perangkat 4	Perangkat 5
Join 4	Berhasil	Berhasil	Gagal	Berhasil	Berhasil
Join 5	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil

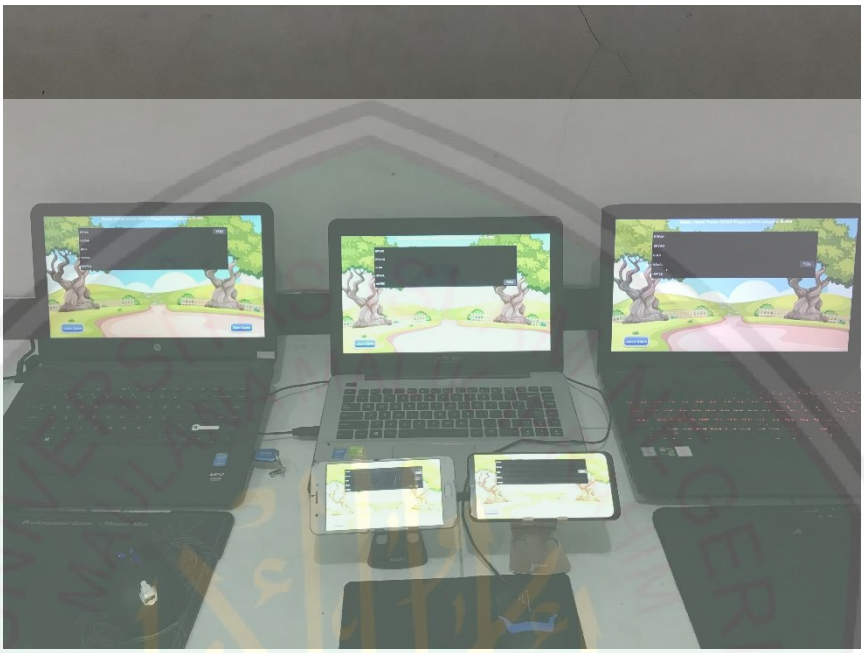
Tabel 4.5 Uji Room List

Join Room	Perangkat 1	Perangkat 2	Perangkat 3	Perangkat 4	Perangkat 5
Join 1	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Join 2	Berhasil	Gagal	Berhasil	Berhasil	Berhasil
Join 3	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Join 4	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil
Join 5	Berhasil	Berhasil	Berhasil	Berhasil	Berhasil

Pada 2 tabel di atas terlihat pada beberapa percobaan yang dilakukan sebagian bersar berhasil terhubung dan masuk ke dalam *Room* yang sudah dibuat. Adapun beberapa kemungkinan gagalnya *Join Room* yang sudah dibuat antara lain sebagai berikut :

1. *Time Out*, artinya ketika 2 atau lebih perangkat melakukan pencarian *room* yang terlalu lama di dalam jaringan yang terhubung dan akhirnya semua perangkat yang terhubung membuat *room* secara bersamaan.
2. Koneksi Internet tidak stabil ataupun tidak adanya jaringan internet (*Offline*) sehingga tidak terhubung ke jaringan sistem.

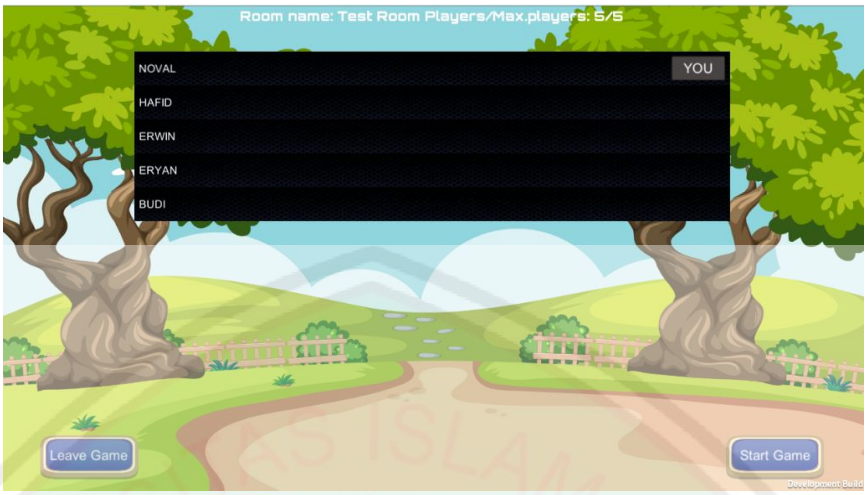
3. Versi *game* yang berbeda antra *Room Mast*er dengan *Node* yang akan bergabung.



Gambar 4.30 Pengujian Join dengan 5 perangkat

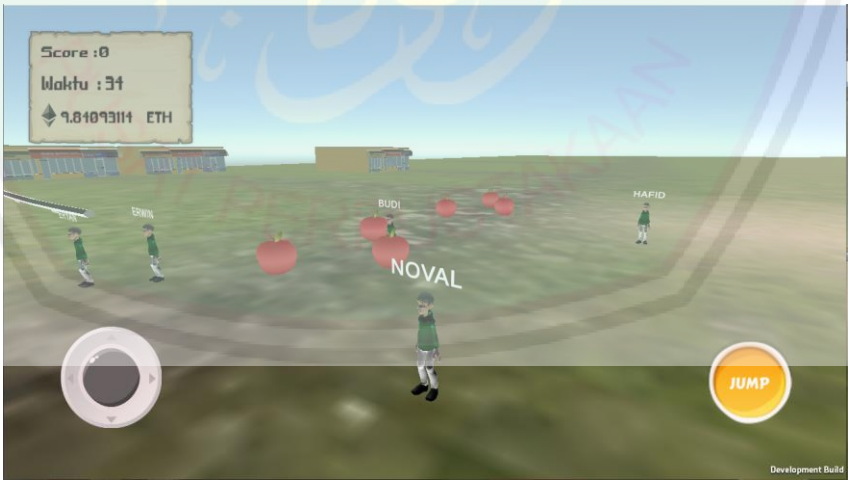
4.6.5 Pengujian Muliplayer

Pada penelitian *game widow waterfall* ini menggunakan *Photon 2* yang berfungsi untuk memungkinkan bermain secara *multiplayer* dan juga mensinkronisasikan setiap data *sharing* yang berjalan pada game yang nantinya akan di kirim ke dalam jaringan *Ethereum Blockchain* yang sudah di koneksikan sebelumnya. *Photon 2* memilki banyak produk yang kompatibel dengan berbagai macam perangkat. Penelitian ini menggunakan *PUN* (*Photon Unity Networking*) sebagai *networking engine*.



Gambar 4.31 Tampilan Room Master

Pada Gambar diatas merupakan *UI* dari *Master R*oom. Hanya *master room* yang dapat menekan *button start*. pada *UI* tersebut tampil semua *Nickname Player* yang akan bermain bersama. Pengujian selanjutnya adalah pengujian *multiplayer*. Dilakukan dengan menekan *button Start Game* oleh *Room Master*. Pengujian ini berguna untuk melihat apakah ke 5 perangkat tersebut dapat masuk dan bermain bersama ke dalam *game*.



Gambar 4.32 Semua player masuk ke dalam game

Gambar diatas adalah tampilan semua *player* berhasil masuk ke dalam *game*. Pengujian menggunakan 5 perangkat berbeda yang sudah di sebutkan sebelumnnya. Berikut data hasil pengujian *multiplayer* yang bisa dilihat pada **tabel**

4.6. Pengujian dinyatakan berhasil ketika semua perangkat berhasil masuk kedalam *game*. Berikut data percobaan *Multiplayer* menggunakan 5 perangkat berbeda.

Tabel 4.6 Uji Multiplayer

Perangkat	Nickname	Spawn ke dalam Game
1	Noval	Berhasil
2	Hafid	Berhasil
3	Erwin	Berhasil
4	Eryan	Berhasil
5	Budi	Berhasil

Dapat dilihat pada tabel diatas bahwa hasil dari uji *Multiplayer* pada *game widow waterfall* menggunakan 5 perangkat yang berbeda dengan hasil *player* dapat masuk atau *spawn* ke dalam arena *game.*

Gambar 4.33 Pengujian Multiplayer dengan 5 perangkat

4.6.6 Pengujian Ambil Poin

Pengujian ini dilakukan dengan *player* mengambil buah apel yang akan memberikan *player poin score*. Setiap apel yang diambil akan memberikan poin *score* sebesar 10 kepada *player*. Player yang mendapatkan poin terbanyak adalah pemenangnya dan akan berada pada posisi nomor 1 pada *leaderboard score* para *player*. Data poin apel inilah yang nantinya akan dikirim ke dalam jaringan *Ethereum Blockchain* sebagai sebuah transaksi yang sukses. Berikut baris *script* yang digunakan untuk mengambil poin dari apel yang tersebar di dalam arena *game widow waterfall.*

```
private void OnTriggerEnter(Collider other){
        if (other.tag == "coinnya") {
        Debug.Log("kena koin");
            if (ScoreText != null) {
                Debug.Log("UI ketemu"+ScoreText);
            }
            tempscore += points;
            Debug.Log(tempscore);
            scorenya.text = tempscore.ToString();
            //GameManager.instance.OnLogin(login, true, false);
            FindObjectOfType<InitPlayerEntry>().ClickAdd("AddKill", points);
        }
}
```

Gambar 4.34 Script Ambil Poin

Gambar dibawah ini menampilkan *score* yang didapatkan oleh *player* pada awal game adalah bernilai “0”.

Gambar 4.35 Score Player bernilai 0

Kemudian ketika player mengambil apel, maka *score* player yang semula benilai “0” berubah menjadi “10” seperti terlihat pada gambar dibawah ini.

Gambar 4.36 Score Player bernilai 10

Pengujian menggunakan 5 perangkat dan nickname yang sudah disebutkan pada sub bab serta pengujian sebelumnya dan dilakukan dengan mengambil apel yang tersebar pada arena *game*. Kolom ambil poin bernilai berhasil jika *player* mengambil apel yang kemudian skornya bertambah. Kolom *Score* akan bernilai poin yang didapat *player* ketika sedang melakukan pengujian. Berikut data percobaan ambil poin menggunakan 5 perangkat berbeda.

Tabel 4.7 Uji Ambil Poin

Perangkat	Nickname	Ambil Poin	Score
1	Noval	Berhasil	70
2	Haped	Berhasil	20
3	Erwin	Berhasil	40
4	Eryan	Berhasil	10
5	Budi	Berhasil	30

Dapat dilihat pada tabel diatas bahwa hasil dari uji Ambil Poin pada *game widow waterfall* menggunakan 5 perangkat yang berbeda dengan hasil *player* dapat mengambil apel tersebut dan kemudian mendapatkan nilai poin yang sesuai dengan banyaknya apel yang diambil.

Gambar 4.37 Pengujian Ambil Poin dengan 5 Perangkat

4.6.7 Pengujian Data Sharing Nickname

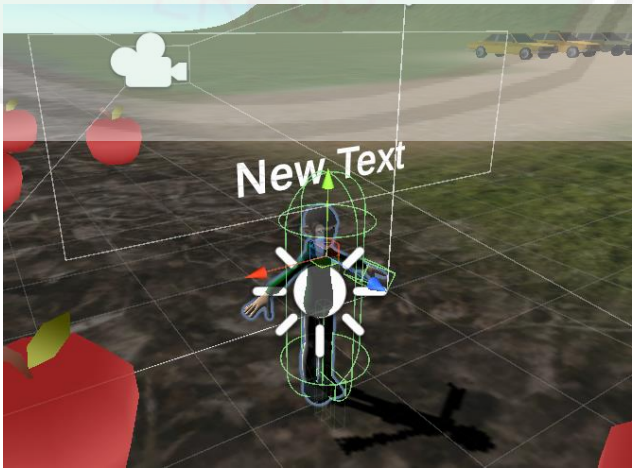
Pengujian selanjutnya adalah data *sharing nickname player*. Pengujian ini dilakukan untuk melihat apakah *nickname player* dapat terlihat oleh *player* lain. Ketika nama *player* terlihat oleh *player* lain maka proses data *sharing nickname player* berhasil. Pada pengujian ini kombinasi *Photon networking engine* dan jaringan *ethereum blockchain* di uji keberhasilannya, *photon* akan mensinkronisasikan data *nickname* dari *node ethereum blockchain* yang sudah terhubung dalam jaringan *peer to peer*.

Node yang dimaksud ialah *player* yang sedang bermain di dalam *game*. Dari data *nickname* masing masing *node* akan di distribusikan ke dalam jaringan

yang terhubung lalu akan ditampilkan kepada *node* lain. sehingga semua *node* yang sudah terhubung bisa melihat *nickname*-nya satu sama lain. hal tersebut penting karena mengingat pada penelitian *game widow waterfall* merupakan tipe *game multiplayer* sehingga bisa membedakan *player* satu dengan *player* lain. Gambar dibawah menampilkan bahwa player atau *node* dengan *nickname* NOVAL dapat melihat *nickname* dari *Node* lain.

Gambar 4.38 Pengiriman data nickname

Script data *sharing nickname* akan dimasukkan ke dalam *game object player* yang sudah dipersiapkan sebelumnya di dalam *unity* sebagai karakter utama dalam *game* seperti terlihat pada gambar dibawah ini.



Gambar 4.39 Object karakter dalam game

Berikut baris *script* untuk data *sharing nickname player* pada penelitian *game widow waterfall*.

```
void setPlayerUI(){

    playerNameText.text = photonView.Owner.NickName;


}
```

Gambar 4.40 Script Sharing Nickname

Seperti terlihat pada baris *script* diatas, data *nickname player* didapat dari *Photon engine* mengambil data *nickname* dari *owner node* jaringan *ethereum blockchain* yang bersangkutan. Setelah didapat maka akan ditampilkan pada *game obeject* bertipe *Text* (*string*) ke dalam *game* yang dapat terlihat oleh semua pemain. Pengujian menggunakan 5 perangkat dan nickname yang berbeda. Berikut data pengujian *data sharing nickname* yang sudah didapatkan. Kolom *spawn* bernilai berhasil ketika *player* dapat muncul di dalam *game* setelah button *start game* di tekan oleh *Room Master*. Kolom tampilan bernilai sesuai ketika tampilan nama dari masing masing *player* sama dengan *nickname player* dan dapat dilihat oleh *player* lain. Berikut data percobaan data *sharing Nickname* menggunakan 5 perangkat berbeda.

Tabel 4.8 Uji Sharing Nickname

Perangkat	Nickname	Spawn	Tampilan
1	Noval	Berhasil	Sesuai
2	Haped	Berhasil	Sesuai
3	Erwin	Berhasil	Sesuai
4	Eryan	Berhasil	Sesuai

Perangkat	Nickname	Spawn	Tampilan
5	Budi	Berhasil	Sesuai

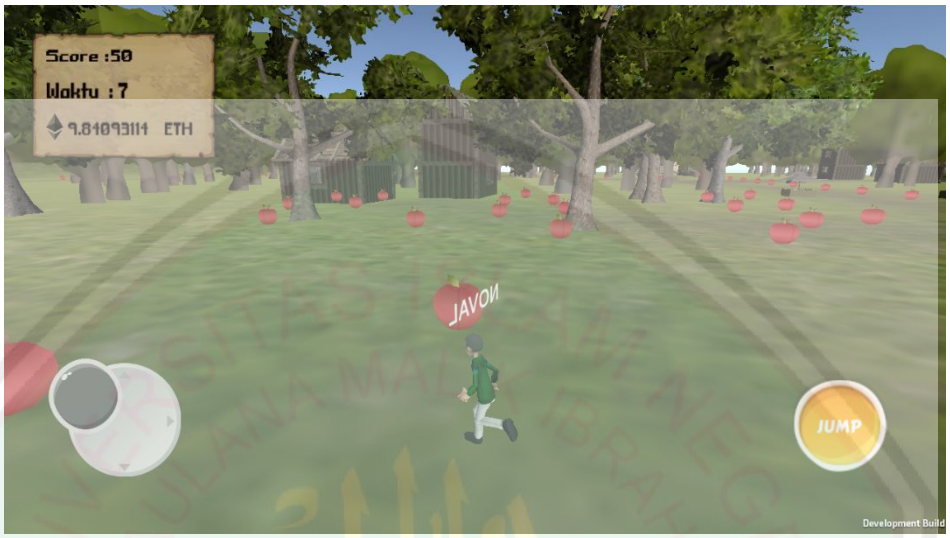
Dapat dilihat pada tabel pengujian diatas bahwa hasil dari Uji *data sharing nickname player* berhasil dilakukan dengan hasil ke 5 player dapat *spawn* ke dalam *game* dan tampilan dari *nickname player* sesuai dengan nama yang tercantum ketika *login game*.

Gambar 4.41 Pengujian Data Sharing Nickname dengan 5 Perangkat

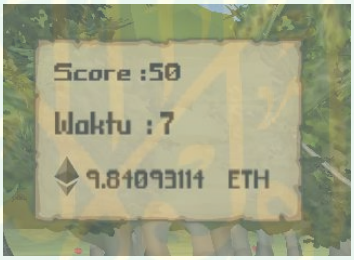
4.6.8 Pengujian Data Sharing Saldo

Pengujian berikutnya adalah data *sharing* saldo. Pengujian ini berfungsi untuk mengetahui data sisa saldo yang sudah ditampilkan pada *login main menu* dapat diambil dan ditampilkan dalam permainan. Data tersebut berguna bagi *player* untuk mengetahui sisa saldo yang akan digunakan untuk pengiriman dan penyimpanan data *score* ke dalam jaringan *Ethereum Blockchain*. Informasi sisa saldo *player* berada pada pojok kiri atas UI player bersama dengan informasi waktu

game yang tersisa, dan *score* yang didapat. Berikut tampilan *UI player* yang menampilkan sisa saldo yang dimiliki *player*



Gambar 4.42 Tampilan UI player.



Gambar 4.43 Player information

Gambar diatas merupakan tampilan dari beberapa Informasi yang dimiliki oleh *player* pada tampililan *UI* nya. Diketahui bahwa sisa salado yang tertera pada *UI player* bernilai " 9.84093114 ETH". Berikut dibawah ini baris *script* yang digunakan dalam *data sharing* sisa saldo pada *game widow waterfall*.

```
void Start()
{
        if(PhotonNetwork.IsConnected){
        int randomPoint = Random.Range(-10,10);
        GameObject Player =
        PhotonNetwork.Instantiate(playerPrefabs.name, new Vector3(randomPoint, 0f, randomPoint),Quaternion.identity);

        GetAccountBalance();

        }
    }
```

Gambar 4.44 Data Sharing Sisa saldo.

Data saldo tersebut diambil dari data informasi saldo yang tertera pada login *main menu game*. Data diambil dari jaringan *ethereum blockchain* ketika terkoneksi

dengan unity. Sistem akan mengambil informasi sisa saldo yang dimiliki oleh *player* yang kemudian ditampilkan di dalam main menu. Ketika sudah berada didalam *game* data tersebut di bagikan kepada *node* yang sudah *login* sesuai dengan alamat dompet (*wallet*) yang dimilikinya.

Berikut data pengujian menggunakan 5 perangkat dan *nickname* yang sudah disebutkan pada sub bab serta pengujian sebelumnya dan dilakukan dengan *player masuk* di dalam game. Kolom *Spawn* bernilai berhasil jika *player* berhasil masuk ke dalam arena *game*. Kolom tampilan bernilai sesuai jika data *sharing* sisa saldo berhasil tampil ke dalam *game*.

Tabel 4.9 Uji Saldo

Perangkat	Nickname	Spawn	Tampilan
1	Noval	Berhasil	Sesuai
2	Haped	Berhasil	Sesuai
3	Erwin	Berhasil	Sesuai
4	Eryan	Berhasil	Sesuai
5	Budi	Berhasil	Sesuai

Dapat dilihat pada tabel pengujian diatas bahwa hasil dari Uji data *sharing* Saldo player berhasil dilakukan dengan hasil ke 5 player dapat spawn ke dalam *game* dan tampilan dari saldo *player* sesuai dengan informasi *saldo* yang tercantum pada *main menu game*.

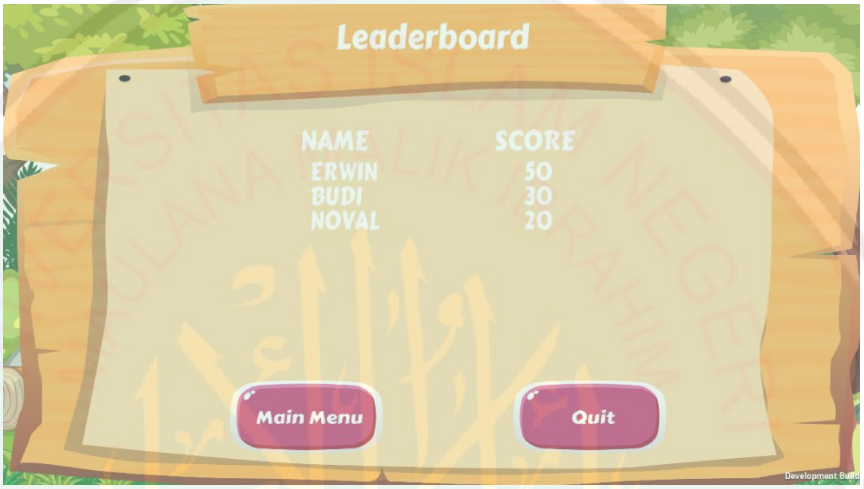
Gambar 4.45 Pengujian Data Sharing Saldo dengan 5 perangkat

4.6.9 Pengujian Data Sharing Score

Pengujian selanjutnya adalah Data *Sharing Score* dan pengurutan posisi *Leaderboard player* berdasarkan banyaknya poin yang didapat *player*. Semua *player* akan mengetahui berapa poin yang di dapat oleh *player* lainnya. *Player* dengan poin terbanyak akan berada pada posisi teratas puncak *leaderboard score* yang kemudian membuat *player* lain ingin mengalahkannya. Data *score* diambil dari misi *game* yaitu mengumpulkan buah apel yang bernilai poin sebanyak mungkin didalam *game*. Data tersebut nantinya akan dibagikan pada semua *node* yang terhubung kedalam jaringan *ethereum blockchain*. dengan dibagikannya data tersebut maka semua *node* mengetahui score *node* lain. data yang ditampilkan juga sudah disimpan terlebih dahulu didalam jaringan *Ethereum blockchain* pada saat pengujian *deploy ethereum contract*. *Scoreboard* atau *Leaderboard* ini bersifat sementara, yang artinya setiap permainan dalam *game* ini akan menghasilkan posisi puncak *player* yang berbeda. Dengan diterapkannya sistem *leaderboard* yang

seperti itu, maka semua *player* memeliki kesempatan untuk menduduki posisi puncak *score player* lain.

Pengujian dilakukan setelah waktu yang diberikan dalam game sudah habis dan *Deploy Ethereum Contract* sudah berhasil serta tampilan *UI* dalam di dalam *game* berubah seperti gambar di bawah ini.



Gambar 4.46 UI Leaderboard Score.

Berikut data pengujian menggunakan 5 perangkat dan *nickname* yang sudah disebutkan pada sub bab serta pengujian sebelumnya dan dilakukan dengan *player* mengambil beberapa buah apel sebagai poin dan bermain di dalam *game* sampai waktu habis. Kolom Tampilan bernilai sesuai jika data *sharing score* berhasil tampil ke dalam *game* dan *player* lain dapat melihatnya. Kolom *Score* benilai poin yang didapat para *player*.

Tabel 4.10 Uji Data Sharing Score

Perangkat	Nickname	Tampilan	Score
1	Noval	Sesuai	80
2	Haped	Sesuai	60
3	Erwin	Sesuai	90

Perangkat	Nickname	Tampilan	Score
4	Eryan	Sesuai	70
5	Budi	Sesuai	110

Seperti terlihat pada tabel diatas bahwa semua *player* dapat melihat *score player* lain. sesuai dengan apa yang didapatkan ketika didalam *game*. Kemudian setelah itu sistem akan mencari *player* dengan poin peringkat tertinggi dengan *script* seperti gambar dibawah berikut ini.

```
public void SortGameObject()
{
    foreach (MPPlayer player in playerListEntries.OrderByDescending(p => p.Kills).ToArray())
    {
        player.ObjForScorepanel.GetComponent<RectTransform>().SetAsLastSibling();
    }
}
```

Gambar 4.47 Script Data Sharing Score.

Pada gambar diatas merupakan baris *script* yang digunakan dalam menentukan urutan posisi *player* di *leaderboard* berdasarkan hasil *score* yang didapat. Kemudian di dapatkan data *Leaderboard* seperti pada tabel dibawah dan gambar berikut ini.

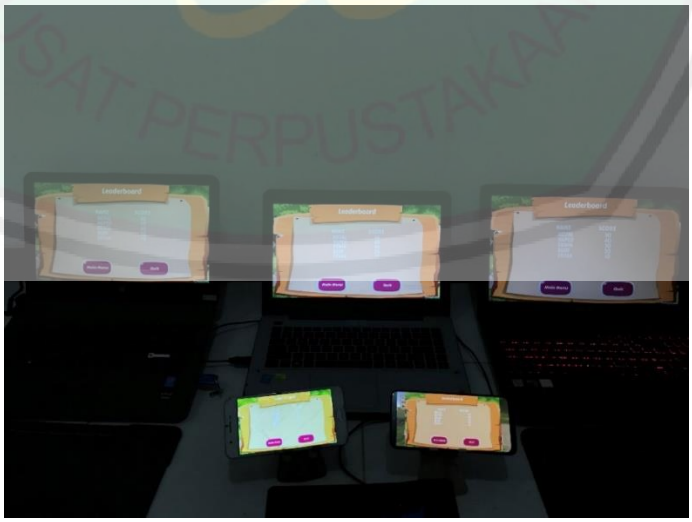
Tabel 4.11 Hasil Pengurutan Score

Peringkat	Nickname	Tampilan	Score	Perangkat (Device)
1	Budi	Sesuai	110	5
2	Erwin	Sesuai	90	3
3	Noval	Sesuai	80	1
4	Eryan	Sesuai	70	4

Peringkat	Nickname	Tampilan	Score	Perangkat (Device)
5	Haped	Sesuai	60	2

Gambar 4.48 Hasil Pengurutan Score Tertinggi.

Dapat dilihat pada pengujian diatas bahwa hasil dari uji data *sharing* *Score player* berhasil dilakukan dengan hasil ke 5 *player* dapat melihat *score* satu sama lainnya berserta posisi urutan sesuai dengan banyaknya poin yang dimiliki *player* tersebut.



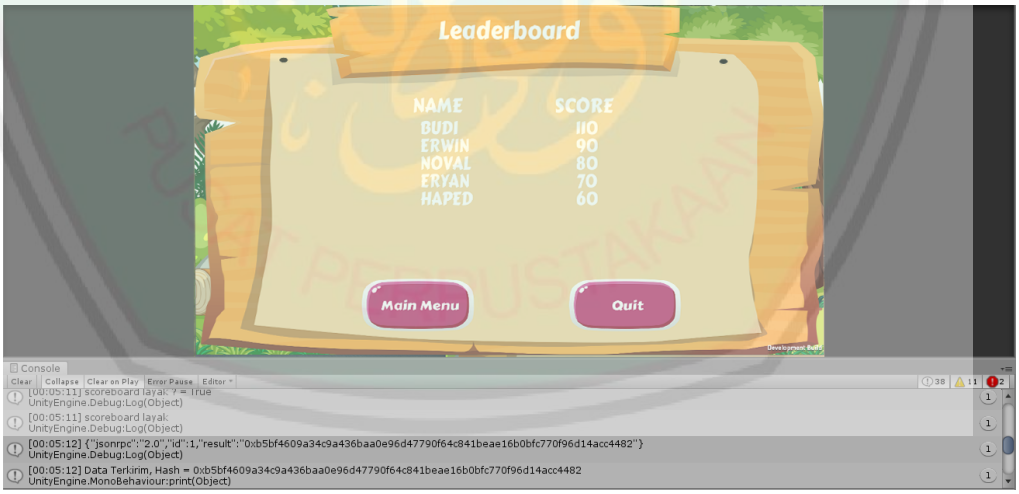
Gambar 4.49 Pengujian Data sharing Score dengan 5 Perangkat

4.6.10 Pengujian Deploy Ethereum

Pengujian ini dilakukan untuk mengetahui apakah implementasi sistem yang telah dibuat dan teknologi *Ethereum Blockchain* dapat berjalan dengan baik. Pengujian dilakukan dengan cara player bermain di dalam arena game sampai waktu bermain habis, ketika waktu sudah habis maka secara otomatis sistem akan mengirim data *score* ke dalam jaringan *ethereum blockchain*. Berikut baris script yang digunakan untuk menentukan waktu bermain atau *TimeRemain*.

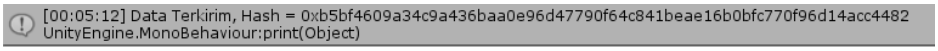
```
void Start () {
    StartCoroutine("LoseTime");
    timeLeft = 60;

}

// Update is called once per frame
0 references
void Update () {
 countdownText.text =("Waktu  : "+timeLeft);
 if(timeLeft == 0){
    StopCoroutine("LoseTime");
    countdownText.text="Waktu Habis";
    UIscore.transform.position = poss.position;


    }
}
```

Gambar 4.50 Score TimeRemain



Gambar 4.51 Pesan Console Unity ketika data terkirim

Ketika data score sudah *terkirim* maka pada *console unity* akan muncul pesan atau kode *hash* dari data tersebut.seperti pada Gambar 4.43 diatas, di dalam *Console Unity* terdapat *Log* pesan seperti pada gambar dibawah berikut ini.



Gambar 4.52 Kode Hash

Kode *hash* tersebut terbentuk ketika sebuah transaksi berhasil dilakukan. Kode *hash* berfungsi juga sebagai bentuk verifikasi bahwa ada data yang tersimpan pada suatu blok pada jaringan *blockchain* dan juga sebagai pembeda antara satu blok satu dengan blok lainnya.Saldo pemain akan berkurang dalam pengujian ini dikarenakan adanya biaya transaksi yang diperlukan untuk mengirim data ke dalam jaringan *Ethereum Blockchain* Berikut baris *script* yang digunakan dalam mengirim data *score* pada penelitian *game widow waterfall*.

```
public void deployEthereumContract() {
    print("Deploying contract...");

    var abi = @"[{""constant"": true, ""inputs"": [], ""name"": ""name"", ""outputs"": [ { ""name"": """", ""type"": ""string"" } ], ""payable"": false,
    var byteCode = @"60606040523415610000f57600080fd5b601260ff16600a0a6305f5e100026001819055506012600a0a6305f5e100026000260003373ffffffffffffffff

    StartCoroutine(deployContract(abi, byteCode, accountAddress, (result) => {
        print("Result " + result);
    }));

}


1 reference
public IEnumerator deployContract (string abi, string byteCode, string senderAddress, System.Action<string> callback) {

    var gas = new HexBigInteger (8000000);


    var transactionInput = contractTransactionBuilder.BuildTransaction(abi, byteCode, senderAddress, gas, null);



    var transactionSignedRequest = new TransactionSignedUnityRequest(_url, accountPrivateKey, accountAddress);


    Debug.Log("Sending Deploy contract transaction...");
    yield return transactionSignedRequest.SignAndSendTransaction(transactionInput);
    if (transactionSignedRequest.Exception == null)
    {

        callback(transactionSignedRequest.Result);
    }
    else
    {

        throw new System.InvalidOperationException("Deploy contract tx failed:" + transactionSignedRequest.Exception.Message);
    }
}
```

Gambar 4.53 Script Deploy Ethereum

Untuk deploy ke dalam jaringan *ethereum blockchain* diperlukan *ABI* dan *Byte Code*. *ABI* dan *byte code* digunakan untuk membuat sekaligus mengakses *smart contract* yang sudah terbuat ketika transaksi sudah berjalan *ABI* dan *Byte code* didapat dari *compile smart contract* yang sudah ditulis sebelumnya. Karena pada penelitian *game widow waterfall* yang dikirim kedalam jaringan *ethereum blockchain* merupakan *data score*, maka perlu ditentukan *smart contract* yang tepat

untuk penyimpanan *data score*. berikut smart contract yang sudah ditulis dalam bahasa pemrograman *Solidity* untuk *score* tertinggi.

Tabel 4.12 Smart Contract

Top Score

```
pragma solidity ^0.4.10;


contract PlayerScore {
    uint maxTopScores = 40;
    address owner;
    struct TopScore{
        address addr;
        int score;
    }
    function PlayerScore(){
        owner = msg.sender;
    }
    TopScore[] public topScores;
    mapping (address=>int) public userTopScores;
    function setTopScore(int256 score, uint8 v, bytes32 r, bytes32 s) {
        var hash = sha3(msg.sender, owner, score);
        var addressCheck = ecrecover(hash, v, r, s);
        if(addressCheck != owner) throw;
        var currentTopScore = userTopScores[msg.sender];
```

Top Score

```
if(currentTopScore < score){
    userTopScores[msg.sender] = score;
}
if(topScores.length < maxTopScores){
    var topScore = TopScore(msg.sender, score);
    topScores.push(topScore);
}else{
    int lowestScore = 0;
    uint lowestScoreIndex = 0;
    for (uint i = 0; i < topScores.length; i++)
    {
        TopScore currentScore = topScores[i];
        if(i == 0){
            lowestScore = currentScore.score;
            lowestScoreIndex = i;
        }else{
            if(lowestScore > currentScore.score){
                lowestScore = currentScore.score;
                lowestScoreIndex = i;
            }
        }
    }
```

Top Score

if(score > lowestScore){
var newtopScore = TopScore(msg.sender, score);
topScores[lowestScoreIndex] = newtopScore;
}
}
}
function getCountTopScores() returns(uint) {
return topScores.length;
}
}

Kode tersebut kemudian di *compile* di website *Remix.org* yang merupakan *Online IDE* khusus untuk membuat *smart contract custom* dalam jaringan *ethereum blockchain*. hasil *compile* dari *code* tersebut maka didapatkan *ABI* dan *Byte code* berikut yang digunakan dalam penelitian *game widow waterfall*.

Tabel 4.13 ABI Code

ABI Code
[{""constant"": true, ""inputs"": [], ""name"": ""name"", ""outputs"": [{ ""name"": "", ""type"": ""**string**"" }], ""payable"": **false**, ""stateMutability"": ""view"", ""type"": ""function"" }, { ""constant"": **true**, ""inputs"": [], ""name"": ""totalSupply"", ""outputs"": [{ ""name"": "", ""type"": ""uint256"" }], ""payable"": **false**, ""stateMutability"": ""view"", ""type"": ""function"" }, { ""constant"": **true**, ""inputs"": [], ""name"": ""pings"", ""outputs"": [{ ""name"": "", ""type"": ""uint256"" }],

ABI Code

""payable"": **false**, ""stateMutability"": ""view"", ""type"": ""function"" }, { ""constant"": **true**, ""inputs"": [], ""name"": ""INITIAL_SUPPLY"", ""outputs"": [{ ""name"": "", ""type"": ""uint256"" }], ""payable"": **false**, ""stateMutability"": ""view"", ""type"": ""function"" }, { ""constant"": **true**, ""inputs"": [], ""name"": ""decimals"", ""outputs"": [{ ""name"": "", ""type"": ""uint8"" }], ""payable"": **false**, ""stateMutability"": ""view"", ""type"": ""function"" }, { ""constant"": **false**, ""inputs"": [], ""name"": ""ping"", ""outputs"": [{ ""name"": "", ""type"": ""uint256"" }], ""payable"": **false**, ""stateMutability"": ""nonpayable"", ""type"": ""function"" }, { ""constant"": **true**, ""inputs"": [{ ""name"": ""_owner"", ""type"": ""address"" }], ""name"": ""balanceOf"", ""outputs"": [{ ""name"": ""balance"", ""type"": ""uint256"" }], ""payable"": **false**, ""stateMutability"": ""view"", ""type"": ""function"" }, { ""constant"": **true**, ""inputs"": [], ""name"": ""symbol"", ""outputs"": [{ ""name"": "", ""type"": ""**string**"" }], ""payable"": **false**, ""stateMutability"": ""view"", ""type"": ""function"" }, { ""constant"": **false**, ""inputs"": [{ ""name"": ""_to"", ""type"": ""address"" }, { ""name"": ""_value"", ""type"": ""uint256"" }], ""name"": ""transfer"", ""outputs"": [{ ""name"": "", ""type"": ""**bool**"" }], ""payable"": **false**, ""stateMutability"": ""nonpayable"", ""type"": ""function"" }, { ""inputs"": [], ""payable"": **false**, ""stateMutability"": ""nonpayable"", ""type"": ""constructor"" }, { ""anonymous"": **false**, ""inputs"": [{ ""indexed"": **false**, ""name"": ""pong"", ""type"": ""uint256"" }], ""name"": ""Pong"", ""type"": ""**event**"" }, { ""anonymous"": **false**, ""inputs"": [{ ""indexed"": **true**, ""name"": ""**from**"", ""type"": ""address"" }, { ""indexed"": **true**, ""name"": ""to"", ""type"": ""address"" }, { ""indexed"": **false**, ""name"": ""**value**"", ""type"": ""uint256"" }], ""name"": ""Transfer"", ""type"": ""**event**"" }

Tabel 4.14 Byte Code

Byte Code
60606040523415610000f57600080fd5b601260ff16600a0a6305f5e10002600181905550
601260ff16600a0a6305f5e10002600260003373ffffffffffffffffffffffffffffffffffffffff1673f
ffffffffffffffffffffffffffffffffffffffff168152602001908152602001600020819055506107 4f
80610083600039600f3006060604052600436106100995760003 57c01000000000000
0000000000000000000000000000000000000000000900463ffffffff16806306fdde031
461009e57806318160ddd1461012c5780631e81ccb2146101555780632ff2e9dc146101
7e578063313ce567146101a75780635c36b186146101d657806370a08231146101ff578
06395d89b411461024c578063a9059cbb146102da575b600080fd5b34156100a9576000
80fd5b6100b1610334565b60405180806020018281038252838181518152602001915 0
80519060200190808383600 05b838110156100f157808201518184015260208101905 0
6100d6565b50505050905090810190601f16801561011e5780820380516001836020 03
6101000a031916815260200191505b509250505060405180910390f35b341561013757
600080fd5b61013f61036d565b60405180828152602001915050604051809103 90f35b3
4156101605760008 0fd5b610168610373565b60405180828152602001915050604051 8
0910390f35b34156101895760008 0fd5b610191610379565b60405180828152602001 9
150506040518091039 0f35b34156101b257600080fd5b6101ba61038a565b6040518082
60ff1660ff16815260200191505060405180910390f35b34156101e157600080fd5b6101
e961038f565b60405180828152602001915050604051809103 90f35b341561020a57600
080fd5b610236600480803573ffffffffffffffffffffffffffffffffffffffff169060200190919050
5061049d565b60405180828152602001915050604051809103 90f35b34156102575760
0080fd5b61025f6104e6565b60405180806020018281038252838181518152602001 91
508051906020019080838360005b838110156102 9f57808201518184015260208101 90
50610284565b505050509050908101906 01f1680156102cc57808203805160018360 20

Byte Code
036101000a031916815260200191505b509250505060405180910390f35b34156102e5
57600080fd5b61031a600480803573ffffffffffffffffffffffffffffffffffffffff1690602001909
190803590602001909190505061051f565b604051808215151515815260200191505060
405180910390f35b604080519081016040528060098152602001 7f50696e67546f6b65
6e0000000000000000000000000000000000000000000000081525081565b6001548156
5b60005481565b601260ff16600a0a6305f5e1000281565b601281565b6000806012 60ff
16600a0a600102905060026000337 3ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffff
ffffffffffffffffffffffff168152602001908152602001600020548111151515610 3ed5760
0080fd5b806001600082825403925050819055508060026000337 3ffffffffffffffffffffff
ffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff168152602001908152602001600
0206000828254039250508190555060008081548092919060010191905055507f58b69f
57828e6962d216502094c54f6562f3bf082ba758966c3454f9e37b152560005460405180
828152602001915050604051809103 90a1600054915050905 65b6000600260008373fff
ffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190
815260200160002054905091905056 5b604080519081016040528060048152602001 7f
50494e4700000000000000000000000000000000000000000000000000000000815250
81565b60008073ffffffffffffffffffffffffffffffffffffffff168373ffffffffffffffffffffffffffffffffffff
fffff161415151561055c57600080fd5b600260003373ffffffffffffffffffffffffffffffffffffffff
1673ffffffffffffffffffffffffffffffffffffffff168152602001908152602001600020548211151
5156105aa57600080fd5b81600260003373ffffffffffffffffffffffffffffffffffffffff1673ffffffff
ffffffffffffffffffffffffffffff168152602001908152602001600020540360026000337 3fff
ffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190
815260200160002081905550816002600857 3ffffffffffffffffffffffffffffffffffffffff1673ff
ffffffffffffffffffffffffffffffffffff168152602001908152602001600020540160026000 85

Byte Code
73ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff1681526020019081526020016000208190555082 73ffffffffffffffffffffffffffffffffffffffff163373ffffffffffffffffffffffffffffffffffffffff167fddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef8460405180828152602001915050604051809103 90a360019050929150505600a165627a7a72305820288a1db2bd7e6b43543a4965700991ac47978b687fc8daad9f4b49911b716b8b0029

transaksi dapat berjalan ketika ada alamat pengirim dan penerima, *ABI* dan *Byte* code, serta saldo *Ether* serta *gas limit* / *gas price* yang cukup. Pengujian menggunakan 5 perangkat dan *nickname* yang sudah disebutkan pada sub bab serta pengujian sebelumnya dan dilakukan dengan bermain di dalam *game* serta mengambil poin berupa buah apel yang sudah tersebar pada arena *game* sampai waktu didalam *game* habis. Kolom *hash* akan bernilai sesuai kode *hash* yang muncul pada *console unity*. Kemudian kode *hash* tersebut akan di cek keasliannya pada pengujian selanjutnya.

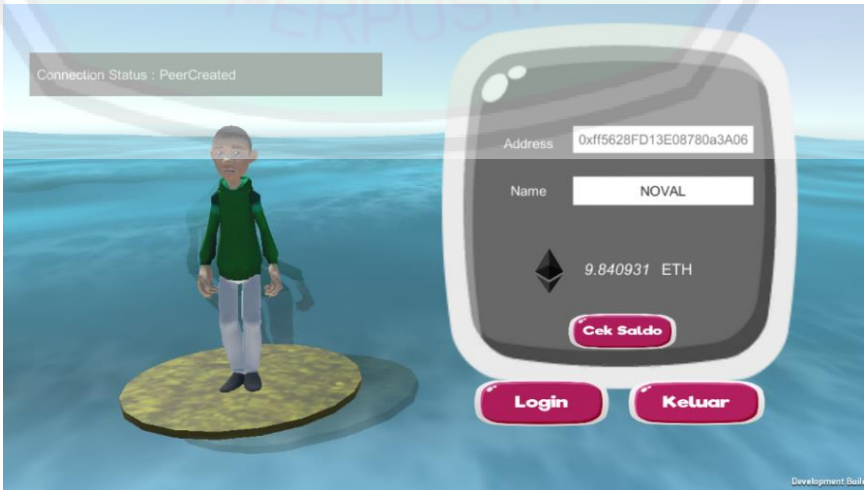
Tabel 4.15 Uji Deploy Ethereum

Perangkat	Nickname	Hash
1	Noval	0x24940877e9e8f45e3eb6c1229b2444afb575d606252707d01b266530cc69cdf2
2	Haped	0x3ef47d6df06b671deba31ef473f5fcf0b6ea80ab0920964984af6e85eea622c4

Perangkat	Nickname	Hash
3	Erwin	0x6c6f5982b0436f70b404a9c2e77f3778a2d0ccc404fc80e1c9e4856836b5155a
4	Eryan	0xf36852daa4699c3922d42e2864c9e562ee7b5a70570ecfac0764bddf4c04c05e
5	Budi	0xe2a3cd76d552eaf98af550f1ba4a34f5384895fbf62a08d71b2ac46abaeafb07

Terlihat pada tabel diatas bahwa 5 player memiliki kode *hash* yang berbeda. Hal tersebut karena setiap transaksi memiliki *unique code* untuk menyimpan informasi data yang dikirim ke dalam jaringan *ethereum blockchain*. Karena keunikan tersebut juga tingkat keamanan data didalam jaringan *ethereum blockchain* sulit untuk diretas.

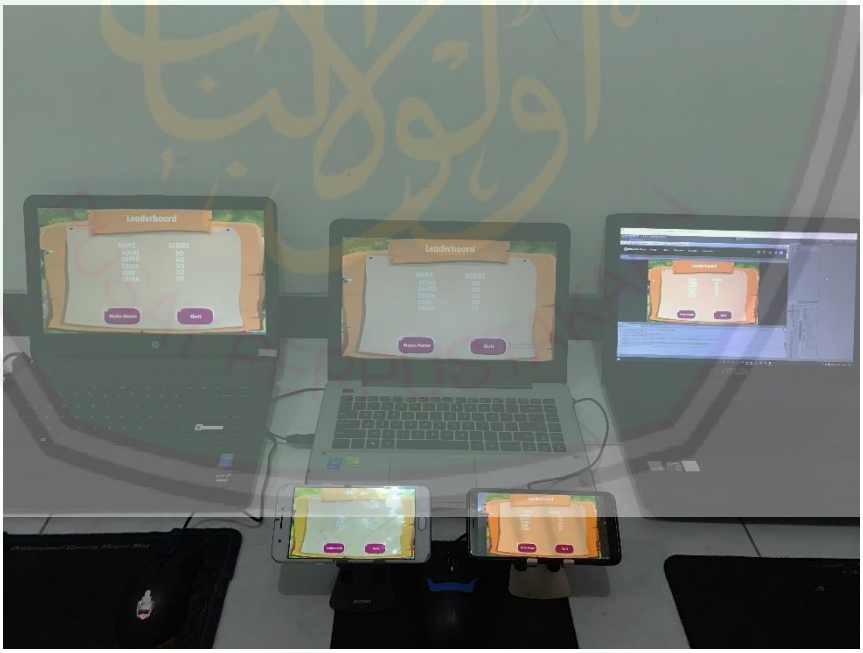
Kemudian kembali ke main menu *game* untuk melihat apakah saldo sudah berkurang dikarenakan transaksi yang terjadi. Sisa saldo *player* sebelum bermain adalah "9.840931" *ETH*. seperti yang terlihat pada gambar dibawah berikut.



Gambar 4.54 Sisa saldo sebelum bermain.

Gambar 4.55 Sisa saldo setelah bermain.

Seperti yang telihat pada **gambar 4.30** sisa salo pemain berkurang menjadi "9.830966" *ETH* setelah *player* menyelesaikan permainan. Hal tersebut menandakan bahwa transaksi pengiriman data ke dalam jaringan *Ethereum Blockchain* berhasil dilakukan.

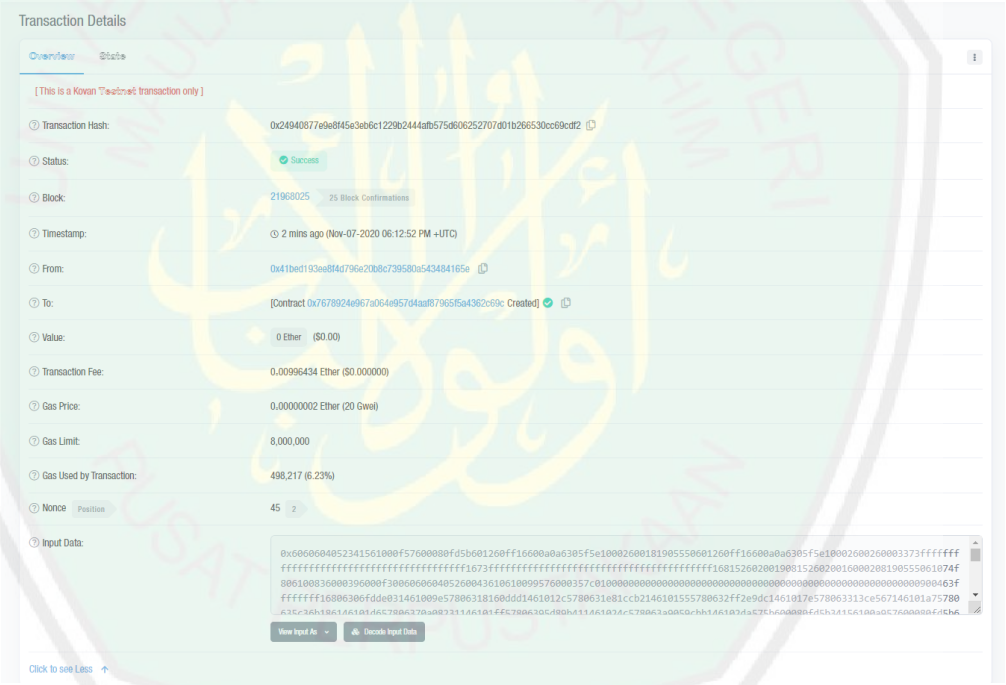


Gambar 4.56 Pengujian Deploy Ethereum dengan 5 perangkat

4.6.11 Pengujian Hash

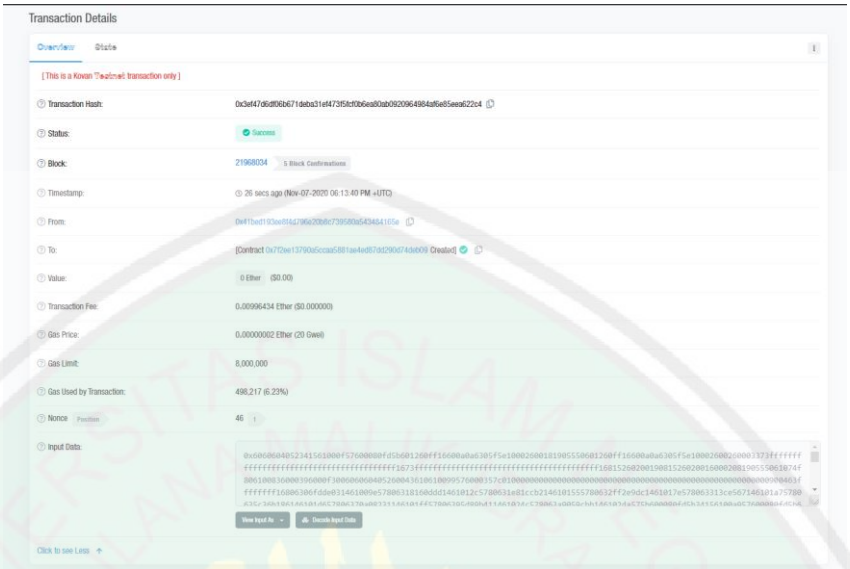
Pengujian ini dilakukan dengan bantuan situs https://kovan.etherscan.io/ untuk mengetahui apakah implementasi *Ethereum blockchain* pada penelitian ini berjalan dengan baik atau tidak. Ketika kode *hash* tersebut terdeteksi pada situs tersebut, maka transaksi pengiriman data pada jaringan *Ethereum Blockchain* berhasil. Berikut merupakan data hasil yang diperoleh menggunakan situs https://kovan.etherscan.io/ yang dapat dilihat pada beberapa gambar dibawah ini :

1. Perangkat 1 Nickname Noval



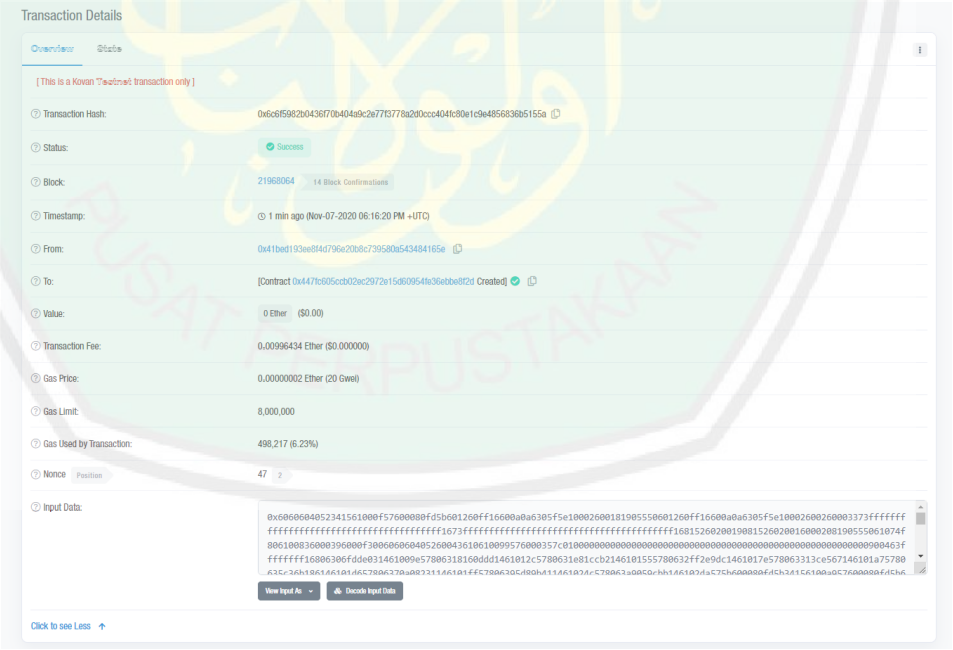
Gambar 4.57 perangkat 1

2. Perangkat 2 Nickname Haped



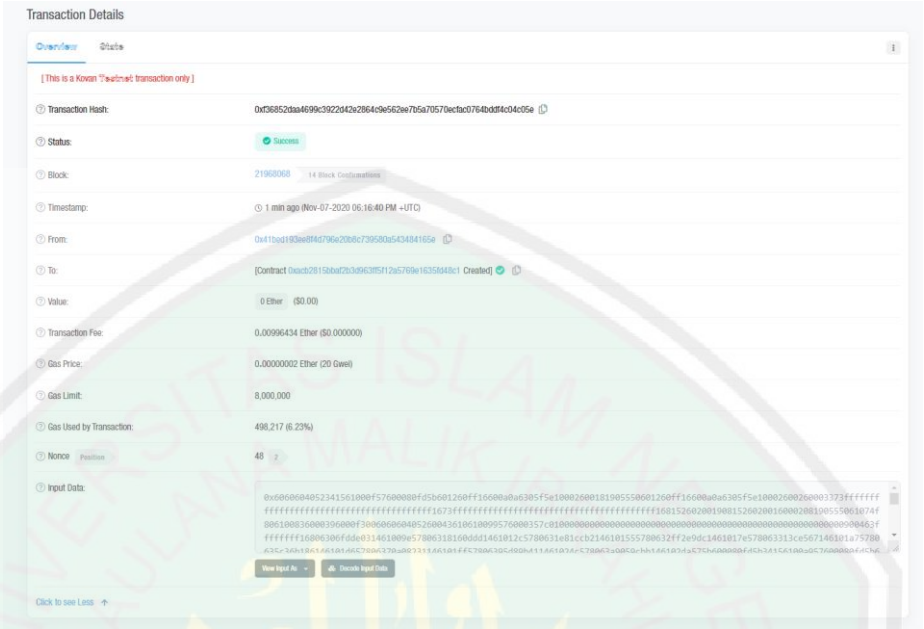
Gambar 4.58 perangkat 2

3. Perangkat 3 Nickname Erwin



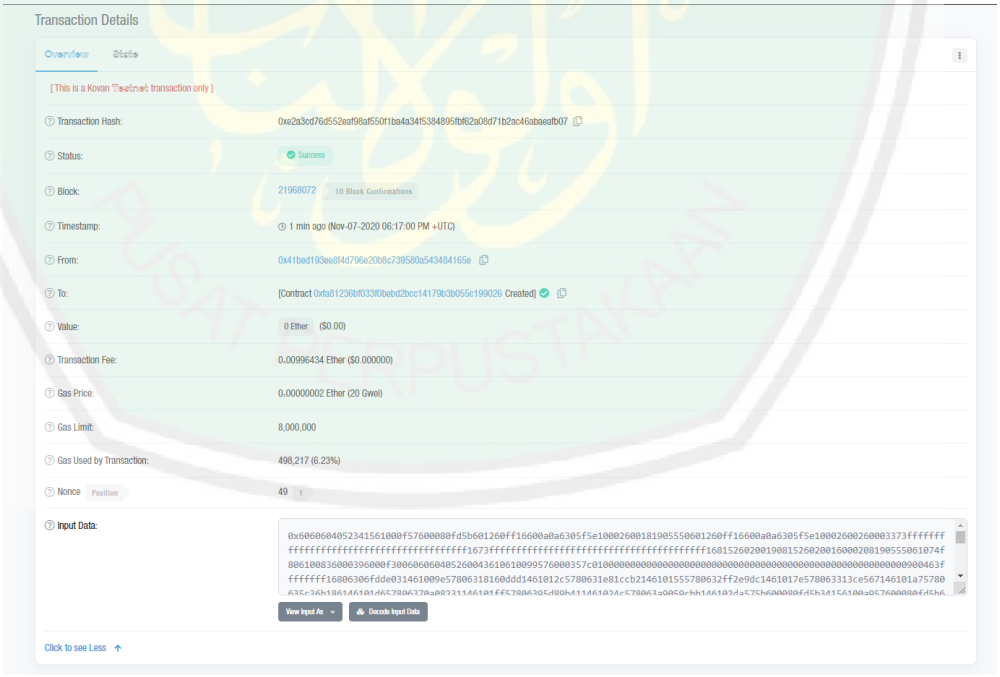
Gambar 4.59 perangkat 3

4. Perangkat 4 Nickname Eryan



Gambar 4.60 perangkat 4

5. Perangkat 5 Nickname Budi



Gambar 4.61 perangkat 5

Terlihat pada semua pengujian diatas bahwa semua kode *hash* yang keluar pada *console unity* berhasil terindentifikasi pada situs pengecekan transaksi

ethereum blockchain https://kovan.etherscan.io/. Adapun data yang dikirimkan dalam transaksi tersebut adalah alamat *hash* transaksi, status keberhasilan, alamat blok, *time stamp*, alamat pengirim, alamat penerima, nilai transaksi, biaya transaksi yang digunakan, *gas limit*, jumlah *gas* yang digunakan ketika transaksi, *gas price* dan *nonce*. Didalam gambar tersebut terlihat bahwa nilai transaksi sebesar 0 *ether* (0$). 5 transaksi pengiriman data *score* pada *etheruem blockchain* dilakukan dengan 5 *player* yang berbeda dan berhasil dilakukan, hal tersebut membuktikan bahwa implementasi pada penelitian ini sukses.

4.6.12 Pengujian Gas Limit

Pengujian ini dilakukan untuk mengetahui jumlah minimum *gas* yang diperlukan untuk mengirim sebuah transaksi ke dalam jaringan *Ethereum blockchain* pada *game widow waterfall*. hal tersebut penting mengingat keberhasilan transaksi juga dipengaruhi oleh jumlah *gas* cukup. Penentuan *gas limit* yang terbaik juga diperlukan untuk mengasilkan biaya pengembangan *game* dengan *minimum cost* sekecil mungkin. Pada tabel dibawah didapatkan hasil *limit gas* sebesar 500.000 untuk keberhasilan transaksi yang tinggi dengan minimal *gas limit* sekecil mungkin. Kolom bernilai berhasil merupakan indikator bahwa transaksi tersebut sukses, *gas limit* sesuai dan tervalidasi pada situs *https://kovan.etherscan.io/*. Kolom bernilai gagal merupakan indikator bahwa *gas limit* tidak sesuai dan nilai *hash* berhasil mucul pada console unity tetapi pada situs *https://kovan.etherscan.io/* terlihat bahwa transaksi gagal tervalidasi.

Tabel 4.16 Uji Gas Limit

Pengujian	Gas Limit			
	400.000	500.000	600.000	700.000
1	gagal	berhasil	berhasil	berhasil
2	berhasil	berhasil	berhasil	berhasil
3	gagal	berhasil	berhasil	berhasil
4	gagal	berhasil	berhasil	berhasil
5	gagal	berhasil	berhasil	berhasil
6	gagal	berhasil	berhasil	berhasil
7	berhasil	berhasil	berhasil	berhasil
8	gagal	berhasil	berhasil	berhasil
9	berhasil	berhasil	berhasil	berhasil
10	gagal	berhasil	berhasil	berhasil

Terlihat pada tabel diatas bahwa *gas limit* yang digunakan dalam penelitian *game widow waterfall* untuk *minimun cost* dan tingkat keberhasilan pengiriman transaksi ke dalam jaringan *Ethereum Blockhchain* yang tinggi sebesar 500.000 *gas limit*. Ketika menggunakan nilai dibawah dari 500.000 tingkat keberhasilan yang sangat rendah. Dengan hasil tersebut maka hasil tersebut menjadi standar *gas limit* pada penelitian *game widow waterfall.*

4.6.13 Pengujian Kecepatan Transakasi

Pada pengujian ini dilakukan untuk mengetahui seberapa cepat sebuah transaksi akan dikonfirmasi oleh situs pengecekan *Ethereum blockchain* yaitu pada situs *https://kovan.etherscan.io/*. Pengujian dilakukan dengan mengubah nilai variabel dari

gas limit yang sudah diujikan sebelumnya. Kemudian melakukan percobaan transaksi beberapa kali serta menghitung berapa rata rata waktu yang diperlukan untuk memproses sebuah transaksi. Untuk rumus yang digunakan dalam menghitung rata rata waktu sebagai berikut. :

$$\overline{T} = \frac{1}{n} \sum_{i=1}^{n} Ti \qquad (4.1)$$

Keterangan :

$\overline{T}$ = rata – rata waktu transaksi

Ti = Waktu yang dibutuhkan untuk memproses sebuah transaksi

n = Banyaknya percobaan

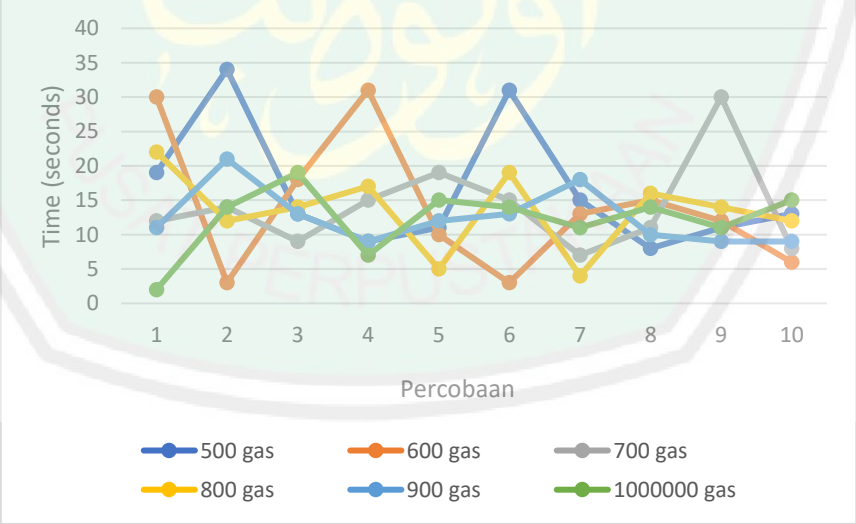
i = transaksi

Gas limit yang digunakan dalam pengecekan kecepatan transaksi dengan variabel 1.000.000, 900.000, 800.000, 700.000, 600.000 dan 500.000. nilai *gas limit* 500.000 adalah nilai minimum yang bisa digunakan untuk melakukan transaksi pada *game widow waterfall*. Jika menggunakan nilai dibawah nilai minimum dari pengujian yang dilakukan di sub bab sebelumnya menunjukkan hasil yang kurang maksimal. Berikut data hasil percobaan yang dilakukan. Nilai yang terdapat didalam setiap percobaan merupakan waktu yang dibutuhkan untuk validasi transaksi dalam satuan Detik (*Seconds*).

Tabel 4.17 Uji Kecepatan transaksi

Percobaan	**Gas Limit**					
	500k	**600k**	**700k**	**800k**	**900k**	**1.000.000**
1	19	30	12	22	11	2

Percobaan	Gas Limit					
	500k	600k	700k	800k	900k	1.000.000
2	34	3	14	12	21	14
3	13	18	9	14	13	19
4	9	31	15	17	9	7
5	11	10	19	5	12	15
6	31	3	15	19	13	14
7	15	13	7	4	18	11
8	8	15	11	16	10	14
9	11	12	30	14	9	11
10	13	6	8	12	9	15
Rata Rata	16.4 s	14.1 s	14 s	13.5 s	12.5 s	12.2 s



Gambar 4.62 Grafik perbandingan kecepatan transaksi

Seperti terlihat pada data tabel dan grafik diatas menunjukkan perbandingan waktu berdasarkan batas *Gas Limit* yang diberikan. Terlihat bahwa semakin besar

batas *Gas limit* yang diberikan, maka proses validasi transaksi juga akan semakin cepat.

4.6.14 Pengujian Estimasi Biaya Pengembangan

Pada pengujian ini dilakukan perhitungan biaya transaksi yang dikeluarkan didalam *game widow waterfall*. Biaya transaksi pada *Ethereum blockchain* tergantung jumlah *gas limit* yang digunakan, semakin besar gas yang digunakan maka semakin besar biaya juga yang dibutuhkan. Untuk perhitungan biaya yang dikeluarkan pada setiap transaksi yang dilakukan dapat menggunakan perhitungan sebagai berikut.

$$T_{Cost} = Gas\ Price\ x\ Estimate\ Gas\ used \quad (4.2)$$

Keterangan :

T_{Cost} = Biaya setiap transaksi

Estimate Gas Used = jumlah gas untuk proses validasi

Gas price = biaya setiap gas yang digunakan

Proses penghitungan biaya dimulai dari estimasi biaya yang dikeluarkan dalam satu hari, satu bulan dan satu tahun. Kemudian dilakukan penyesuaian biaya *gas* mengikuti pada pengujian sebelumnya, dan akan didapat beberapa data estimasi Biaya pengembangan. Biaya yang dikeluarkan pada estimasi biaya pengembangan merupakan biaya pengembangan game maupun pemeliharaan. Dalam *game widow waterfall*, biaya setiap transaksi data *score* yang dilakukan tidak selalu menjadi tanggung jawab pengembang, melainkan menjadi tanggung jawab *player* yang bermain game *widow waterfall*, karena data tersebut di simpan didalam jaringan *Ethereum blockchain*. pengembang akan melakukan pemeliharan dengan transaksi

yang seminim mungkin untuk memastikan bahwa *game* tetap berjalan. Sedangkan biaya transaksi data *score* yang dilakukan *player* akan di bebankan kepada *player tersebut*. Pada *game widow waterfall* biaya *default* gas price yang digunakan adalah 20 *gwei* karena sudah ditetapkan di dalam *smart contract* yang telah dibuat oleh pengembang sebelumnya, sehingga pada perhitungan ini dilakukan perubahan variabel *gas limit* yang sudah diujikan di sub bab sebelumnya. Berikut skenario estimasi dalam sehari *player* maupun *developer* melakukan 15 kali transaksi data *score* pada jaringan *ethereum blockchain*.

1. **Perhitungan Gas Limit 500.000**

Gas price: 20 *Gwei* *Estimate Gas Used* : 500.000

Gas limit : 500.000 *Transaction/year* : 15 x 365 = 5.475

T_{Cost} = *Gas Price x Estimate Gas used*

20 x 500.000 = 10.000.000

T_{Cost} = 10.000.000 *Gwei* = Rp.68.409

Estimasi biaya pengembangan = biaya transaksi x transaksi per tahun

= 68.409 x 5.475

= Rp.374.539.275

2. **Perhitungan Gas Limit 600.000**

Gas price: 20 *Gwei* *Estimate Gas Used* : 600.000

Gas limit : 600.000 *Transaction/year* : 15 x 365 = 5.475

T_{Cost} = *Gas Price x Estimate Gas used*

20 x 600.000 = 12.000.000

T_{Cost} = 12.000.000 *Gwei* = Rp.82.091

Estimasi biaya pengembangan = biaya transaksi x transaksi per tahun

= 82.091 x 5.475

= Rp.449.448.225

3. Perhitungan Gas Limit 700.000

Gas price: 20 *Gwei* *Estimate Gas Used* : 700.000

Gas limit : 700.000 *Transaction/year* : 15 x 365 = 5.475

T_{Cost} = *Gas Price x Estimate Gas used*

20 x 700.000 = 14.000.000

T_{Cost} = 14.000.000 Gwei = Rp.95.773

Estimasi biaya pengembangan = biaya transaksi x transaksi per tahun

= 95.773 x 5.475

= Rp.524.357.175

4. Perhitungan Gas Limit 800.000

Gas price: 20 *Gwei* *Estimate Gas Used* : 800.000

Gas limit : 800.000 *Transaction/year* : 15 x 365 = 5.475

T_{Cost} = *Gas Price x Estimate Gas used*

20 x 800.000 = 16.000.000

T_{Cost} = 16.000.000 Gwei = Rp.109.455

Estimasi biaya pengembangan = biaya transaksi x transaksi per tahun

= 109.455 x 5.475

= Rp.599.266.125

5. Perhitungan Gas Limit 900.000

Gas price: 20 *Gwei* *Estimate Gas Used* : 900.000

Gas limit : 900.000 *Transaction/year* : 15 x 365 = 5.475

T_{Cost} = *Gas Price x Estimate Gas used*

20 x 900.000 = 18.000.000

T_{Cost} = 18.000.000 *Gwei* = Rp.123.137

Estimasi biaya pengembangan = biaya transaksi x transaksi per tahun

= 123.137 x 5.475

= Rp.674.175.075

6. Perhitungan Gas Limit 1.000.000

Gas price: 20 *Gwei* *Estimate Gas Used* : 1.000.000

Gas limit : 1.000.000 *Transaction/year* : 15 x 365 = 5.475

T_{Cost} = *Gas Price x Estimate Gas used*

20 x 1.000.000 = 20.000.000

T_{Cost} = 20.000.000 *Gwei* = Rp.136.818

Estimasi biaya pengembangan = biaya transaksi x transaksi per tahun

= 123.137 x 5.475

= Rp.749.078.550

Dilihat dari beberapa percobaan diatas didapatkan perhitungan estimasi biaya pengembangan total yang berdasarkan biaya transaksi yang di kalikan dengan *gas limit* yang dibutuhkan. data diatas kemudian diringkas maka mendapatkan data seperti pada tabel dibawah berikut.

Tabel 4.18 Estimasi Biaya pengembangan

Gas Price	Gas Used	Biaya transaksi (Rp)	Biaya harian (Rp)	Biaya bulanan (Rp)	Biaya tahunan (Rp)
20	500.000	68.409	1,026,135	30,784,050	374.539.275
20	600.000	82.091	1.231.365	36.940.950	. 449.448.225
20	700.000	95.773	1.436.595	43.097.850	524.357.175
20	800.000	109.455	1.641.825	49.254.750	599.266.125
20	900.000	123.137	1.847.055	55.411.650	674.175.075
20	1.000.000	136.818	2.052.270	61.568.100	749.078.550

Terlihat pada tabel diatas estimasi biaya yang dikeluarkan dalam harian, bulanan dan tahunan.Terlihat bahwa pentingnya bagi developer menentukan *gas price* atau *gas limit* sekecil mungkin untuk menekan biaya pengembangan. Hal tersebut juga bermanfaat bagi *player* yang akan bermain *game* sehingga tidak mengeluarkan biaya yang banyak.

4.7 Evaluasi

Dengan semua pengujian dan percobaan diatas antara lain uji koneksi, uji Login, uji create room, uji join room, uji multiplayer, uji data sharing saldo, uji ambil poin, uji deploy ethereum contract untuk score, dan uji data sharing score, uji biaya estimasi pengembangan, uji kecepatan transaksi, uji gas limit, maka di dapatkan hasil seperti tabel dibawah ini.

Tabel 4.19 Evaluasi Pengujian

No	Skenario Pengujian	Hasil Pengujian
1	Pada Console Unity menampilkan pesan Koneksi Berhasil, serta Token Hex Code dari saldo Pemain	Koneksi unity dan ethereum berhasil terkoneksi dengan baik.
2	Ketika Pemain menekan Button Refresh yang berada di Game, maka akan menampilkan Informasi Sisa saldo pemain	Pada main menu game berhasil menampilkan sisa saldo yang dimiliki oleh player
3	Pemain bisa melihat sisa saldo dompet mereka dalam game	player berhasil melihat sisa saldo yang dimiliki ketika sedang di dalam arena game
4	Score tercatat sebagai transaksi valid dalam jaringan Ethereum Blockhchain	Kode hash terkonfirmasi pada situs pengecekan hash yaitu Kovan Etherscan
5	Para Pemain bisa melihat hasil Score yang mereka dapat dalam Game	Semua player bisa melihat hasil score satu sama lainnya.

Terlihat pada tabel diatas bahwa 5 pengujian utama berhasil dilaksanakan sesuai dengan yang diharapkan oleh penulis untuk penelitian data *sharing* untuk *score* pada *game widow waterfall* menggunakan *Ethereum Blockchain*.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian yang dilakukan merupakan penelitian implementasi blockchain dalam sistem data sharing untuk score pada game widow waterfall. Pengujian dilakukan dengan menggunakan 5 perangkat yang berbeda untuk mengetahui performa dari sistem yang telah dibangun. Pada penelitian ini kesimpulan yang didapat adalah :

1. Pengujian sistem data sharing yang dilakukan menggunakan 3 pengujian antara lain *Nickname*, sisa Saldo, dan *Score* mendapatkan hasil yang memuaskan sesuai dengan harapan penulis yaitu semua pada data sharing *nickname* semua *player* dapat melihat nama *player* lain. Kemudian pada data *sharing* sisa saldo *player* dapat melihat saldo mereka didalam arena bermain dan pada data *sharing score* didapatkan hasil *player* dapat melihat hasil *score player* lain
2. Uji *Deploy Ethereum* berhasil dilakukan dengan tingkat keberhasilan sebesar 100%. Hal tersebut juga didukung dengan semua kode *Hash* yang mucul berhasil teridentifikasi dan tervalidasi pasa situs Kovan Etherscan serta *player* dapat melihat *score* yang mereka dapatkan.
3. Dalam Pengujian .Estimasi biaya pengembangan didapatkan hasil perhitungan estimasi biaya per transaksi dalam satu tahun berdasarkan pengaturan variabel *Gas Limit* yang berbeda. Dengan biaya transaksi termurahs sebesar Rp.68.409 atau sebesar 10.000.000 *gwei* per transaksi.

4. Kecepatan validasi dalam sebuah transaksi pada jaringan *Ethereum Blockchain* ditentukan oleh berapa banyak *gas* yang dibutuhkan. semakin banyak *gas* yang dikonsumsi maka akan semain cepat waktu yang dibutuhkan untuk validasi transaksinya.

5.2 Saran

Pembuatan sistem data sharing untuk score pada game widow waterfall menggunakan ethereum blockchain ini jauh dari kata sempurna. Banyak kekurangan yang nantinya harus diperbaiki untuk kebutuhan penelitian kedepannya supaya sistem dan gamenya jauh lebih baik. Berikut beberapa saran yang bisa penulis sampaikan untuk penelitian kedepannya :

1. Mengembangkan tampilan UI serta Ux game agar jauh lebih indah dan mudah dipahami alur permainannya.
2. Menambah enviroment game agar terlihat lebih hidup, seperti menambahkan binantang ataupun npc, dll.
3. Menambahkan karakter yang dapat dimainkan seperti wisatawan dengan pakaian yang berbeda.
4. Menyesuaikan *gas limit* sehingga bisa menekan biaya pengembangan

DAFTAR PUSTAKA

Alharby, M., & Moorsel, A. van. (2017). *Blockchain Based Smart Contracts : A Systematic Mapping Study*. 125–140. https://doi.org/10.5121/csit.2017.71011

Arif, Y. M., Pradana, R. P., Nurhayati, H., Nugroho, S. M. S., & Hariadi, M. (2020). *A Blockchain-Based Multiplayer Transaction For Tourism Serious Game*. 138–143.

Banno, R., & Shudo, K. (2019). Simulating a Blockchain Network with SimBlock. *ICBC 2019 - IEEE International Conference on Blockchain and Cryptocurrency*, 3–4. https://doi.org/10.1109/BLOC.2019.8751431

Buterin, V. (2014). A next-generation smart contract and decentralized application platform. *Etherum*, *January*, 1–36. http://buyxpr.com/build/pdfs/EthereumWhitePaper.pdf

Byun, K.-R., & Byun, H.-W. (2018). Game Money Exchange System using Blockchain. *Journal of Digital Contents Society*, *19*(12), 2437–2446. https://doi.org/10.9728/dcs.2018.19.12.2437

Dinh, T. T. A., Liu, R., Zhang, M., Chen, G., Ooi, B. C., & Wang, J. (2018). Untangling Blockchain: A Data Processing View of Blockchain Systems. *IEEE Transactions on Knowledge and Data Engineering*, *30*(7), 1366–1385. https://doi.org/10.1109/TKDE.2017.2781227

Fauzan, N. I. (2018). *Abstrak*. *4*, 1–15.

Ghimire, S., & Selvaraj, H. (2019). A survey on bitcoin cryptocurrency and its mining. *26th International Conference on Systems Engineering, ICSEng 2018 - Proceedings*, 1–6. https://doi.org/10.1109/ICSENG.2018.8638208

Hu, Y., Manzoor, A., Ekparinya, P., Liyanage, M., Thilakarathna, K., Jourjon, G., & Seneviratne, A. (2019). A Delay-Tolerant payment scheme based on the ethereum blockchain. *IEEE Access*, *7*, 33159–33172. https://doi.org/10.1109/ACCESS.2019.2903271

Jani, S. (2018). *An Overview of Ripple Technology & its Comparison with Bitcoin Technology*. *January*, 1–5.

Kalra, S., Goel, S., Dhawan, M., & Sharma, S. (2018). *ZEUS: Analyzing Safety of Smart Contracts*. *February*. https://doi.org/10.14722/ndss.2018.23082

Lee, C.-I., Chen, I.-P., Hsieh, C.-M., & Liao, C.-N. (2017). Design Aspects of Scoring Systems in Game. *Art and Design Review*, *05*(01), 26–43. https://doi.org/10.4236/adr.2017.51003

Liu, X. (2019). A Small Java Application for Learning Blockchain. *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference, IEMCON 2018*, 1271–1275. https://doi.org/10.1109/IEMCON.2018.8614961

Mutar, H. A., & AL-Huseiny, M. S. (2019). Implementation of national cryptocurrency using ethereum development platform. *Periodicals of Engineering and Natural Sciences*, *7*(3), 1021–1029. https://doi.org/10.21533/pen.v7i3.634

Nakamoto, S. (2009). RAIN: A Bio-Inspired Communication and Data Storage Infrastructure. *Artificial Life*, *23*(4), 552–557. https://doi.org/10.1162/ARTL_a_00247

Saini, K. (2019). A future's dominant technology blockchain: Digital transformation. *2018 International Conference on Computing, Power and Communication Technologies, GUCON 2018*, 937–940. https://doi.org/10.1109/GUCON.2018.8675075

Shiddiqi, A. M., Abidin, A. Z., & Wibisono, W. (2010). Rancang Bangun Jaringan Peer To Peer Dengan Konsep. *Seminar Nasional Informatika*, *1*(3), 9–14.

Studi, P., Informatika, T., Informatika, J. T., Komputer, F. I., & Brawijaya, U. (2019). *Pengembangan Sistem Penyimpanan Data Sensor Menggunakan Platform Hyperledger*. *3*(5), 5154–5164.