TUGAS AKHIR – IF184802

METODE *CROWDSOURCING* MENGGUNAKAN *ETHEREUM* UNTUK MELINDUNGI PRIVASI PADA LINGKUNGAN DINAMIS

M. Hazdi Kurniawan
05111640000072

Dosen Pembimbing
Bagus Jati Santoso, S.Kom., Ph.D.
Dr. Radityo Anggoro, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2020

TUGAS AKHIR – IF184802

METODE *CROWDSOURCING* MENGGUNAKAN *ETHEREUM* UNTUK MELINDUNGI PRIVASI PADA LINGKUNGAN DINAMIS

M. Hazdi Kurniawan
05111640000072

Dosen Pembimbing I
Bagus Jati Santoso, S.Kom., Ph.D.

Dosen Pembimbing II
Dr. Radityo Anggoro, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya
2020

[*Halaman ini sengaja dikosongkan*]

FINAL PROJECT – IF184802

CROWDSOURCING METHOD USING ETHEREUM WITH PRIVACY PRESERVATION IN MOBILE ENVIRONMENT

M. Hazdi Kurniawan
05111640000072

Supervisor I
Bagus Jati Santoso, S.Kom., Ph.D.

Supervisor II
Dr. Radityo Anggoro, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya
2020

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

METODE *CROWDSOURCING* MENGGUNAKAN *ETHEREUM* UNTUK MELINDUNGI PRIVASI PADA LINGKUNGAN DINAMIS

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S-1 Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:
M. Hazdi Kurniawan
NRP: 05111640000072

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Bagus Jati Santoso, S.Kom., Ph.D. .....
NIP. 198611252018031001 (Pembimbing 1)

Dr. Radityo Anggoro, S.Kom., M.Sc. .....
NIP. 198410162008121002 (Pembimbing 2)

SURABAYA
Juni 2020

[*Halaman ini sengaja dikosongkan*]

METODE *CROWDSOURCING* MENGGUNAKAN *ETHEREUM* UNTUK MELINDUNGI PRIVASI PADA LINGKUNGAN DINAMIS

Nama Mahasiswa	:	M. Hazdi Kurniawan
NRP	:	051116 40000 072
Departemen	:	Teknik Informatika Fakultas Teknologi Elektro dan Informatika Cerdas – ITS
Dosen Pembimbing 1	:	Bagus Jati Santoso, S.Kom., Ph.D.
Dosen Pembimbing 2	:	Dr. Radityo Anggoro, S.Kom., M.Sc.

Abstrak

Metode crowdsourcing adalah proses di mana suatu individu atau kelompok dapat bertukar ide, layanan, atau uang secara daring atau langsung. Permasalahan dalam tugas akhir ini adalah seringnya terjadi kerusakan atau modifikasi informasi pada penggunaan metode crowdsourcing dengan menggunakan sistem tersentralisasi. Dengan menyimpan seluruh data pengguna dalam satu node, menjadikan sistem ini rentan terhadap serangan. Kerusakan atau modifikasi informasi dapat mengganggu proses pertukaran dalam metode crowdsourcing dan merugikan banyak pihak.

Blockchain bersifat terdesentralisasi dan terdistribusi. Setiap data yang disimpan di blockchain menjadi immutable atau tidak mungkin untuk diubah. Dengan menerapkan metode crowdsourcing dengan menggunakan teknologi blockchain, diharapkan dapat menyelesaikan masalah kerusakan informasi.

Sistem kriptografi yang digunakan pada blockchain membuat data yang disimpan sangat aman. Tetapi semakin aman data memiliki trade-off yaitu blockchain tidak dapat mengeksekusi banyak transaksi setiap detik. Selain itu, bahasa pemrograman, library, dan aplikasi yang digunakan dalam tahap pengembangan masih kurang dalam dokumentasi.

Kata kunci: Crowdsourcing, Blockchain, Data sensitif, Terdesentralisasi, Terdistribusi

CROWDSOURCING METHOD USING ETHEREUM WITH PRIVACY PRESERVATION IN MOBILE ENVIRONMENT

Student Name	: M. Hazdi Kurniawan
Registration Number	: 051116 40000 072
Department	: Department of Informatics Faculty of Intelligent Electrical and Informatics Technology – ITS
First Supervisor	: Bagus Jati Santoso, S.Kom., Ph.D.
Second Supervisor	: Dr. Radityo Anggoro, S.Kom., M.Sc.

Abstract

Crowdsourcing is a process where an individual or groups can exchange ideas, services, or money online or directly. The problem in this final project is the frequent damage or modification of information about the use of crowdsourcing methods using a centralized system. By storing all user data in one node, making this system vulnerable to attacks. Damage or modification of information can disrupt the exchange process in the crowdsourcing method and is detrimental to many parties.

Blockchain is decentralized and distributed. Any data stored on the blockchain is immutable or impossible to change. By applying crowdsourcing method using blockchain technology, it is expected to solve the problem of information damage.

The cryptographic system used on the blockchain makes the data stored very safe. But the more secure the data has a trade-off that is the blockchain cannot do many transactions every second. In addition, the programming languages, libraries, and applications used in the development phase are lacking in documentation.

Keywords: Crowdsourcing, Blockchain, Sensitive data, Decentralized, Distributed

KATA PENGANTAR

Puji syukur penulis ucapkan kepada Tuhan Yang Maha Esa atas pimpinan, penyertaan, dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

METODE *CROWDSOURCING* MENGGUNAKAN *ETHEREUM* UNTUK MELINDUNGI PRIVASI PADA LINGKUNGAN DINAMIS

Pengerjaan Tugas Akhir ini dilakukan untuk memenuhi salah satu syarat meraih gelar Sarjana di Departemen Informatika Fakultas Teknologi Elektro dan Informatika Cerdas Institut Teknologi Sepuluh Nopember.

Dengan selesainya Tugas Akhir ini diharapkan apa yang telah dikerjakan penulis dapat memeberikan manfaat bagi perkembangan ilmu pengetahuan terutama di bidang teknologi informasi serta bagi diri penulis sendiri selaku peneliti.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama penulis mengerjakan Tugas Akhir maupun selama menempuh masa studi antara lain:

1. Terimakasih kepada Allah SWT, di mana penulis masih diberi kesempatan, kesehatan dan umur untuk menempuh kuliah di sini dan menjalani hidup dengan baik.
2. Ibu dan Bapak penulis yang selalu memberikan perhatian do'a, dorongan, dan juga kasih sayang agar lebih semangat menempuh kuliah dan segera menyelesaikan Tugas Akhir ini.
3. Bapak Bagus Jati Santoso, S.Kom., Ph.D. selaku Dosen Pembimbing I yang telah membimbing penulis selama penyelesaian Tugas Akhir ini.
4. Bapak Dr. Radityo Anggoro, S.Kom., M.Sc. selaku dosen pembimbing II yang telah memberikan ilmu kepada penulis.
5. Teman-teman angkatan 2016 Informatika ITS yang telah menemani perjuangan penulis selama masa perkuliahan.

6. Teman-teman saya yang telah menemani perjuangan penulis selama masa perkuliahan.
7. Serta pihak-pihak lain yang tidak dapat disebutkan di sini yang telah banyak membantu penulis dalam menyusun Tugas Akhir ini.

Penulis mohon maaf apabila masih ada kekurangan pada Tugas Akhir ini. Penulis juga mengharapkan kritik dan saran yang membangun untuk pembelajaran dan perbaikan di kemudian hari. Semoga melalui Tugas Akhir ini penulis dapat memberikan kontribusi dan manfaat yang sebaik-baiknya.

Surabaya, Juni 2020

M. Hazdi Kurniawan

DAFTAR ISI

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan masalah, batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan buku Tugas Akhir ini.

1.1 Latar Belakang

Crowdsourcing merupakan sebuah metode yang popular dimana setiap individual atau sekelompok orang dapat memperoleh barang atau jasa, seperti ide, layanan servis, dan uang. Metode ini biasanya melibatkan pengguna internet untuk menari dan membagi pekerjaan untuk memberikan hasil yang komulatif.

Akan tetapi penerapan metode ini dengan menggunakan sistem yang tersentralisasi atau terpusat dapat meningkatkan kemungkinan dari perusakan atau modifikasi informasi. Penerapan metode *crowdsourcing* dengan menggunakan teknologi *blockchain* yang terdesentralisasi dan terdistribusi dapat terhindar dari masalah tersebut. *Blockchain* yang bersifat terdesentralisasi dan terdistribusi membuat data yang telah disimpan di *blockchain* bersifat *immutable* yang berarti sangat sulit bagi peretas untuk mengubah informasi yang sudah disimpan di *blockchain*. *Smart contract* yang terdapat pada *blockchain* juga dapat mengurangi biaya transaksi dan tidak lagi membutuhkan aplikasi pihak ketiga untuk melakukan transaksi. [1]

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana menghindari perusakan informasi pada metode *crowdsourcing*?

2. Bagaimana memberikan waktu layanan yang maksimal pada metode *crowdsourcing*?

1.3 Batasan Permasalahan

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Aplikasi yang digunakan adalah layanan berbasis web dan memerlukan aplikasi tambahan bernama *Metamask* sebagai jembatan untuk mengakses *Blockchain*.
2. Teknologi *Blockchain* yang digunakan adalah *Ethereum* dengan mengguakan bahasa pemrograman *solidity*.

1.4 Tujuan Pembuatan Tugas Akhir

Tujuan dari pembuatan Tugas Akhir ini adalah mengimplementasikan metode *crowdsourcing* dengan menggunakan teknologi *blockchain* untuk menjaga keamanan dan informasi pengguna.

1.5 Manfaat Tugas Akhir

Tugas Akhir ini diharapkan dapat dapat memberikan waktu layanan yang maksimal, meningkatkan keuntungan, mengurangi penggunaan energi bagi pemberi layanan, dan meminimalisir terjadinya perusakan informasi pada penggunakan metode *crowdsourcing*.

1.6 Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan Tugas Akhir ini yaitu:

1.6.1 Penyusunan Proposal Tugas Akhir

Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal tugas akhir ini berisi tentang penjelasan mengenai pendahuluan dari tugas akhir yang dibuat, dan tinjauan pustaka yang digunakan sebagai referensi pendukung pada pembuatan tugas akhir.

1.6.2 Studi Literatur

Pada tahap ini dilakukan pencarian informasi dan studi literatur yang relevan untuk dijadikan referensi dalam melakukan pengerjaan Tugas Akhir. Informasi didapatkan dari materi-materi yang berhubungan dengan metode *crowdsourcing* dan teknologi *blockchain*.

1.6.3 Implementasi Perangkat Lunak

Implementasi perangkat lunak pada tugas akhir ini dilakukan dengan menggunakan *Ethereum* sebagai teknologi *Blockchain*, bahasa pemrograman yang digunakan yaitu *javascript* dengan menggunakan *library* react pada front-end dan *Solidity* pada back-end aplikasi.

1.6.4 Pengujian dan Evaluasi

Pengujian pada Tugas Akhir ini dilakukan dengan mendeploy *smart contract* ke *Ethereum Virtual Machine* (EVM), dan *javascript* yang bekerja sebagai *front-end* sekaligus sebagai *server-side* yang dapat berinteraksi dengan *smart contract*.

1.6.5 Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini merupakan laporan secara lengkap mengenai Tugas Akhir yang telah dikerjakan baik dari sisi teori, rancangan, maupun implementasi sehingga memudahkan bagi pembaca dan juga pihak yang ingin mengembangkannya lebih lanjut. Sistematika penulisan buku Tugas Akhir secara garis besar antara lain:

Bab I Pendahuluan

Bab ini berisi penjelasan latar belakang, rumusan masalah, batasan masalah dan tujuan pembuatan Tugas Akhir. Selain itu, metodologi pengerjaan dan sistematika penulisan laporan Tugas Akhir juga dijelaskan di dalamnya.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab ini berisi kajian teori atau penjelasan metode, algoritma, *library*, dan *tools* yang digunakan dalam pembuatan tugas akhir.

Bab III Desain dan Perancangan

Bab ini berisi penjelasan tentang rancangan dari sistem yang akan dibangun.

Bab IV Implementasi

Bab ini berisi implementasi dari rancangan yang telah dibuat pada bab sebelumnya. Implementasi disajikan dalam bentuk *pseudocode* disertai dengan penjelasannya.

Bab V Pengujian dan Evaluasi

Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai hasil percobaan yang telah dilakukan.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menjelaskan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

[*Halaman ini sengaja dikosongkan*]

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dijelaskan mengenai dasar teori yang menjadi dasar pengerjaan Tugas Akhir ini.

2.1 Deskripsi Umum Permasalahan

Permasalahan yang diangkat pada Tugas Akhir ini adalah rusaknya informasi pada penggunaan metode *crowdsourcing* pada aplikasi tersentralisasi. Sifat terdesentralisasi dan *immutable* pada *blockchain* dapat menjadi solusi pada masalah kerusakan informasi pada metode *crowdsourcing*. Dan dengan adanya *smart contract* yang ada pada *blockchain*, *smart contract* dapat berfungsi seperti aplikasi pihak ketiga, jadi pengguna tidak perlu lagi mengeluarkan banyak biaya tambahan untuk setiap transaksi.

Dengan menerapkan metode *crowdsourcing* dengan menggunakan teknologi *blockchain*, penulis berharap dapat menyelesaikan masalah kerusakan informasi yang sering terjadi. Selain itu, beberapa factor juga diperhitungkan untuk mendapatkan waktu layanan yang maksimal bagi para penyedia layanan. [1]

2.2 *Crowdsourcing*

Crowdsourcing adalah proses yang menggunakan banyak individu yang berbeda (kerumunan / *crowd*) untuk melakukan atau memberikan layanan baik berupa ide, waktu, keahlian, atau uang. Orang yang terlibat pada metode ini biasanya bekerja sebagai freelancer yang dibayar, atau orang yang ingin mengerjakan tugas-tugas kecil secara sukarela. Misalnya, pengemudi yang melaporkan kecelakaan pada aplikasi lalu lintas, atau insiden jalan lainnya untuk memberikan informasi terkini kepada pengguna aplikasi lainnya. [2]

2.3 *Blockchain*

Blockchain adalah sistem penyimpanan data digital berisikan catatan yang terhubung melalui kriptografi. Teknologi *blockchain* kini telah dimanfaatkan oleh berbagai sektor, salah satunya untuk transaksi mata uang kripto seperti *Bitcoin*. Fitur keamanan yang digunakan oleh blockchain pada dasarkan adalah *public-key / asymmetric cryptography* dan fungsi *hash*, yang peranannya dalam *blockchain* akan dijelaskan pada subbab berikut.

1. Public-Key Cryptography

Blockchain menggunakan *public-key cryptography* untuk mengamankan pertukaran informasi antar pengguna dengan mengotentikasi transaksi melalui tanda tangan digital. Selama proses tanda tangan, penandatangan menandatangani kunci dengan menggunakan kunci pribadi, sedangkan kunci publik digunakan untuk memverifikasi bahwa tanda tangan itu valid.

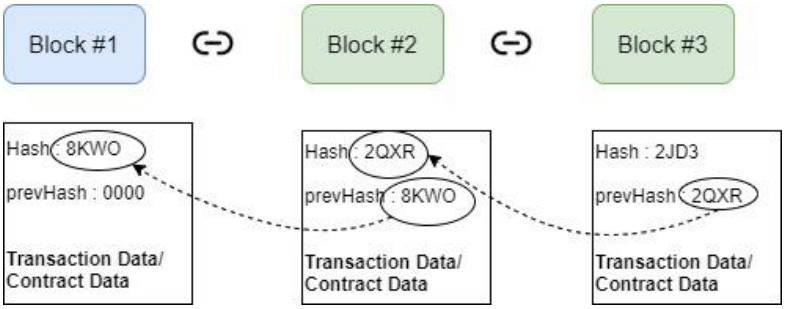
Public-key cryptography juga penting untuk setiap dompet pengguna, dimana dompet tersebut merupakan wadah untuk kunci pribadi yang menyimpan beberapa file dan data sederhana. Dengan demikian, dalam sistem blockchain setiap pengguna memiliki dompet yang dikaitkan dengan alamat publik (biasanya hash dari kunci publik pengguna) dan kunci pribadi yang akan dibutuhkan oleh pemilik dompet untuk menandatangani transaksi. Sebagai contoh, dalam blockchain seperti *Bitcoin*, setiap transaksi dikirim ke alamat publik penerima dan ditandatangani dengan kunci pribadi pengirim. Untuk mengirim *bitcoin*, pemilik transaksi perlu menunjukan kepemilikan kunci pribadi. Untuk memverifikasi transaksi tersebut, penerima *bitcoin* akan memverifikasi tanda tangan digital pada transaksi menggunakan kunci publik pengirim

2. Fungsi *hash*

Fungsi *hash* seperti *SHA-256* atau *Scrypt* biasanya digunakan oleh *blockchains* karena mudah diperiksa, tetapi

sangat sulit untuk dipalsukan, sehingga memungkinkan tanda tangan digital yang pengguna *blockchain* perlukan untuk mengautentikasi diri mereka sendiri atau data mereka di depan orang lain.

Fungsi hash juga digunakan oleh *blockchain* untuk menyambungkan setiap blok. Setiap blok tersebut dihubungkan dalam urutan berdasarkan urutan kronologis, yang berisi hash dari blok sebelumnya, dan hash blok tersebut. Fungsi hash dalam blockchain juga digunakan untuk menghasilkan alamat pengguna (kunci publik / kunci pribadi) atau untuk memperpendek alamat publik. [3]



Gambar 2.1 Data pada setiap block

2.4 *Ethereum*

Ethereum adalah platform komputasi terdistribusi publik berbasis *blockchain*, dan open source. Seperti *blockchain* lainnya, *Ethereum* memiliki *cryptocurrency* asli yang disebut *Ether* (ETH). Penulis memilih *Ethereum* sebagai teknologi *blockchain* yang dipakai karena *Ethereum* dapat menjalankan *smart contract*, yang memungkinkan untuk memodifikasi setiap transaksi sesuai kode program yang dibuat. [4]

Ethereum menyediakan mesil virtual, yaitu *Ethereum Virtual Machine* (EVM) yang dapat mengeksekusi skrip menggunakan jaringan internasional dari node publik atau jaringan lokal dari node pribadi. Set instruksi pada mesin virtual berbeda dengan yang lainnya seperti *Turing-complete* pada skrip *Bitcoin*. EVM menggunakan ‘gas’ yang merupakan mekanisme penetapan harga internal, digunakan untuk mengurangi spam, dan mengalokasikan sumber daya di jaringan.

2.5 *Solidity*

Solidity adalah bahasa pemrograman yang dibuat oleh tim *Ethereum*. Bahasa ini merupakan bahasa pemrograman tingkat tinggi, berorientasi objek, dan dibuat khusus untuk membuat *smart contract* pada berbagai platform *blockchain*, salah satunya *Ethereum*. *Solidity* didasarkan oleh C++, Python, dan JavaScript. [5]

2.6 *Smart Contract*

Smart Contract atau Kontrak Pintar merupakan protokol komputer yang berfungsi untuk memfasilitasi, memverifikasi secara digital yang ditulis melalui kode program. *Smart contract* bekerja tanpa melalui pihak ketiga dan memiliki proses transaksi yang kredibel sehingga tidak bisa di retas ataupun diubah. Dengan mengunakan *smart contract*, kita dapat melakukan pertukaran uang, properti, saham atau apapun secara transparan dan tanpa perantara. [6]

2.7 *JavaScript*

JavaScript (JS) merupakan bahasa pemrograman tingkat tinggi, multi-paradigma, dan dikompilasi secara *real-time*. Bersama dengan HTML dan CSS, javasript juga merupakan salah

satu teknologi inti dari *World Wide Web*. Pada umumnya *Javascipt* digunakan pada web browser untuk menciptakan halaman web yang menarik dan interaktif serta menerapkan berbagai fungsi pada halaman web. [7]

2.8 React.js

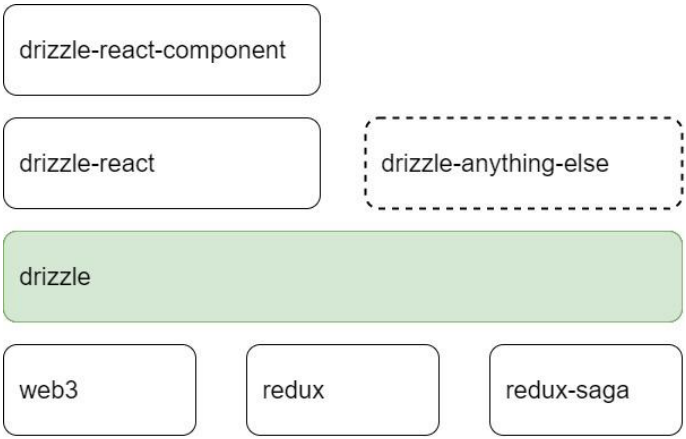
React.JS adalah sebuah *library* javascript yang bersifat *open source* untuk membangun *User Interface* yang dibuat oleh Facebook [8]. ReactJS hanya mengurusi semua hal yang berkaitan dengan tampilan dan logika disekitarnya. ReactJS dapat mendesain tampilan sederhana untuk setiap level dalam aplikasi, sehingga dapat digunakan untuk membuat dan mengembangkan pembuatan aplikasi berbasis web. Pada Tugas Akhir ini, react yang digunakan adalah react dengan versi 16.13.1.

2.9 Truffle

Truffle merupakan salah satu lingkungan pengembangan untuk blockchain yang menggunakan EVM. Truffle mempermudah pengembang dApps untuk mengompilasi, menguji, men-*debug*, dan men-*deploy smart contract* pada dApps. [9]

2.10 Drizzle

Drizzle merupakan *library* JavaScript yang mempermudah untuk memproses data dari jaringan blockchain. Drizzle didasarkan pada react redux, sehingga drizzle memiliki akses ke seluruh alat pengembangan yang tersedia pada react redux, hal ini mempermudah kita dalam mengambil dan mengirim data ke jaringan blockchain. Drizzle juga menangani sinkronisasi pada *smart contract*, data transaksi, dan lainnya. [10]



Gambar 2.2 Ilustrasi Drizzle [10]

Drizzle memiliki beberapa bagian yang dapat digunakan secara terpisah, yaitu drizzle-react-component dan drizzle-store. *Library* drizzle-store sudah termasuk dalam drizzle-react-component. Drizzle-store berfungsi untuk mengambil data dari jaringan blockchain dengan menggunakan *library* web3, dan drizzle-react-component berfungsi untuk memberikan visualisasi terhadap data yang diambil oleh drizzle-store. Drizzle-react merupakan *library* tua dari drizzle, digunakan hanya untuk penggunaan react dengan versi 16.2 kebawah. Karena versi react yang digunakan adalah versi 16.13.1 maka *library* drizzle dan drizzle-react-component akan digunakan pada Tugas Akhir ini.

2.11 Ganache

Ganache merupakan EVM yang memungkinkan pengembang dApps untuk mengembangkan, melakukan uji coba, dan men-deploy dApps pada jaringan Ethereum atau Codra di jaringan lokal / pribadi. [11]

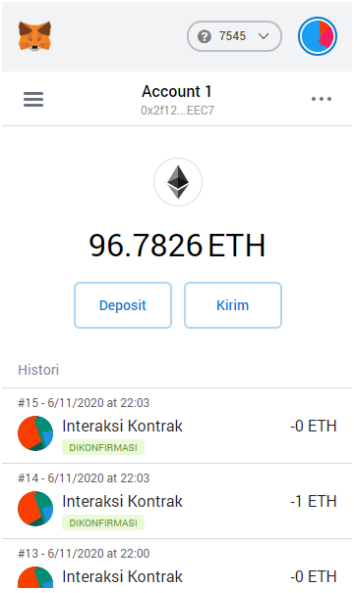
Ganache UI merupakan aplikasi desktop yang mendukung teknologi Ethereum dan Codra. Selain itu, ganache juga memiliki

versi lain hanya untuk teknologi Ethereum dan hanya dapat digunakan menggunakan terminal yaitu ganache-cli.

Ganache akan memberikan beberapa akun dengan *ether* yang sudah tersedia pada setiap akun, akun tersebut dapat digunakan untuk melakukan testing kepada aplikasi, banyaknya akun dapat diubah sesuai keinginan pengembang aplikasi. Ganache UI menggunakan port 7545, port ini akan digunakan ketika menyambungkan ganache dengan metamask pada browser.

2.12 Metamask

Metamask merupakan aplikasi tambahan yang berfungsi untuk menyambungkan browser yang digunakan ke jaringan Ethereum. Metamask dapat disambungkan dengan EVM pribadi seperti Ganache dengan cara meng-impor seed dan port yang diberikan oleh Ganache. [12]



Gambar 2.3 Tampilan Metamask

Metamask memberikan informasi ke website yang memerlukan metamask, informasi tersebut berupa jaringan yang digunakan, jumlah *ether*, dan alamat publik pengguna. Metamask tidak menyebarkan alamat pribadi, dan hanya pemilik akun yang dapat melihat alamat pribadi tersebut dengan menggunakan kata sandi.

2.13 Geolib

Geolib merupakan *library* yang menyediakan operasi geospasial dasar seperti perhitungan jarak, konversi koordinat desimal ke sexadesimal dan sebaliknya, dan lainnya. Saat ini *library* hanya dapat bekerja pada dimensi 2D, artinya ketinggian tidak digunakan pada setiap fungsinya. [13]

BAB III
DESAIN DAN PERANCANGAN

Pada bagian ini akan dijelaskan desain dan perancangan sistem yang digunakan untuk menyelesaikan permasalahan pada Tugas Akhir ini.

3.1 Definisi Umum Sistem

Crowdsourcing merupakan metode yang efisien, dimana setiap permintaan yang diberikan akan dikerjakan oleh pekerja lepas yang memiliki jadwal lebih fleksibel dan bayaran yang dapat ditentukan sesuai keinginan pembuat permintaan. Pembuat permintaan biasanya membagi sebuah tugas besar yang ditransmisikan menjadi tugas-tugas yang lebih kecil agar memiliki pekerja yang banyak, hal ini akan membuat tugas tersebut menjadi lebih cepat. Setiap pekerja atau penyedia layanan tentu menginginkan keuntungan dan waktu layanan yang maksimal.

Metode *crowdsourcing* biasanya dibuat dengan menggunakan sistem terpusat, hal ini membuat aplikasi rentan terhadap serangan dari peretas. Semakin banyak pengguna yang mengakses aplikasi tersebut akan menghasilkan hasil yang lebih baik untuk metode ini, tetapi lebih banyak pengguna berarti lebih banyak informasi pribadi yang dimiliki oleh aplikasi, sehingga jika server pusat diserang akan menyebabkan masalah besar.

Sistem *blockchain* dapat menghasilkan aplikasi yang terdesentralisasi dan terdistribusi, hal ini diusulkan dapat menyelesaikan masalah dari serangan peretas. Sistem yang terdesentralisasi atau tidak terpusat membuat sistem ini tidak memiliki server utama dan menjadikan sistem ini sangat sulit untuk diretas.

Sistem yang akan dibuat dalam tugas akhir ini merupakan penerapan metode *crowdsourcing* dengan sistem terdesentralisasi atau tidak terpusat. Sistem dapat dibuat dengan menggunakan

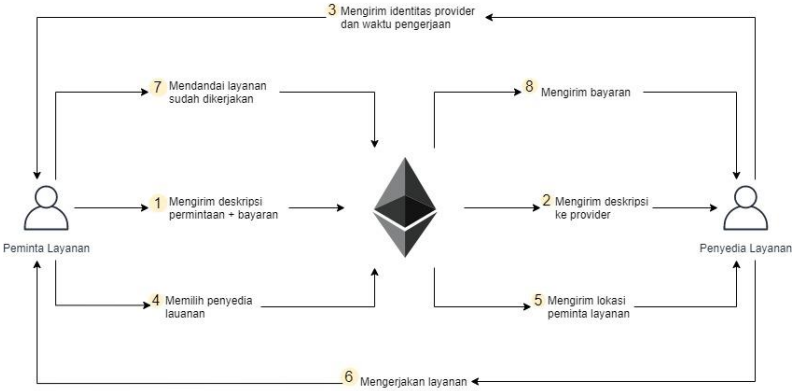
teknologi *blockchain* sebagai sistem basis data, dan juga sebagai *middleman* dengan menggunakan *smart contract* yang tersedia pada *blockchain*. Teknologi *blockchain* yang akan digunakan pada tugas akhir ini adalah *blockchain* pribadi dengan menggunakan aplikasi *Ganache* untuk membuat jaringan *blockchain* pada jaringan lokal. Tujuan akhir dari pembuatan tugas akhir ini adalah dapat menerapkan metode *crowdsourcing* terdistribusi dengan menggunakan teknologi *blockchain* untuk menghindari kerusakan dan pencurian informasi pribadi pengguna dari serangan luar.

Sistem berkomunikasi dengan *blockchain* menggunakan aplikasi tambahan yaitu *metamask*, yang perlu diunduh dahulu oleh pengguna sebelum mengakses aplikasi. *Metamask* berfungsi sebagai jembatan antara *blockchain* pribadi pada *ganache* dengan sisi antarmuka aplikasi. Setiap transaksi seperti membuat permintaan, membaca data, atau memodifikasi data pada *blockchain* hanya dapat dilakukan dengan menggunakan *metamask* yang telah tersambung ke jaringan *ganache*.

Selain itu jarak dari setiap permintaan dan waktu mulai permintaan akan diperhitungkan untuk mendapatkan waktu layanan dan keuntungan yang maksimal bagi penyedia layanan. Untuk mendapatkan waktu mulai akan digunakan *library* geolib yang tersedia pada react. Lokasi dari pembuat permintaan dan penyedia layanan akan dibandingkan untuk mendapatkan waktu mulai yang terbaik untuk penyedia layanan.

3.2 Alur Kerja Sistem

Berikut merupakan diagram tentang alur kerja sistem untuk aplikasi ini.



Gambar 3.1 Diagram alur kerja sistem

Pada Gambar 3.1 terdapat dua aktor yang tersedia pada aplikasi ini, yaitu pemohon layanan dan penyedia layanan. Pemohon layanan bekerja sebagai aktor yang membuat permintaan ke aplikasi dan penyedia layanan bekerja sebagai aktor yang mengerjakan permintaan yang dibuat oleh pemohon layanan. *Ethereum* merupakan tempat penyimpanan data atau basis data yang digunakan pada aplikasi untuk menyimpan setiap data transaksi yang dilakukan antara pemohon layanan dan penyedia layanan.

Pada gambar tersebut pemohon layanan membuat permintaan dengan memberikan deskripsi dan bayaran untuk permintaan yang akan dibuat. Untuk menghindari penipuan dari pemohon layanan, bayaran akan dikirimkan langsung saat membuat permintaan dan akan disimpan pada *Etheruem*. Bayaran tersebut hanya bisa dicairkan jika pemohon layanan membatalkan permintaan atau penyedia layanan menyelesaikan permintaan tersebut.

Setelah pemohon layanan berhasil membuat permintaan, permintaan tersebut akan diposting pada aplikasi dan dapat dilihat oleh seluruh pengguna aplikasi. Penyedia layanan yang ingin mengerjakan permintaan tersebut perlu mengirimkan waktu dan

tanggal untuk pengerjaan layanan. Setiap penyedia layanan yang mendaftar untuk mengerjakan permintaan tersebut dapat dilihat oleh pemohon layanan dan pemohon layanan akan memilih penyedia layanan yang memiliki waktu dan tanggal sesuai dengan keinginan pemohon.

Penyedia layanan yang dipilih oleh pemohon akan diberikan informasi terkait waktu pengerjaan dan lokasi dari pemohon. Setelah penyedia layanan mengerjakan permintaan, pemohon layanan akan menandai bahwa permintaan tersebut sudah diselesaikan dan bayaran akan langsung diberikan kepada penyedia layanan.

3.3 Analisis Waktu Layanan

Pada bagian ini, analisis waktu layanan bertujuan untuk memberikan waktu pelayanan yang maksimal kepada penyedia layanan, dengan cara menentukan waktu mulai suatu permintaan berdasarkan permintaan terdekat. Daftar notasi dapat dilihat pada Tabel 3.1.

Tabel 3.1 Daftar Notasi

Notasi	Keterangan
RR	Permintaan yang tersedia
RP	Permintaan yang dilayani
rr_i	Permintaan yang tersedia ke-i
rp_i	Permintaan yang dilayani ke-i
$d_{j,k}$	Jarak antara permintaan j ke k
$tr_{j,k}$	Waktu pindah dari permintaan j ke k
est_i	Estimasi waktu mulai permintaan i

Pada tabel tersebut *RR* merupakan hasil pengelompokkan dari seluruh permintaan yang tersedia, *RP* merupakan hasil pengelopokkan dari seluruh permintaan yang akan dilayani oleh penyedia layanan yang sedang mengakses website. *RR* terdiri dari set. Setiap permintaan pada *RR* diwakili oleh rr_i dimana ($i =$

$\{1, 2, \ldots, N\}$) mewakili permintaan yang tersedia ke-i. Dan permintaan pada *RP* diwakili oleh rpi dimana ($i = \{1, 2, \ldots, N\}$) mewakili permintaan yang akan dilayani ke-i.

Setiap rp_i memiliki waktu mulai dan waktu selesai yang diwakili dengan st_i dan ft_i. Kedua variabel ini akan digunakan untuk menentukan waktu mulai dari setiap entitas dalam *RR*.

3.3.1 Analisis Waktu Perjalanan

Untuk mendapatkan waktu layanan yang maksimal, pertama diperlukan untuk mencari rp_i terdekat dari rr_i. Lalu waktu perjalanan dari rp_i terdekat ke rr_i sudah dapat ditentukan dengan

$$tr_{j,k} = \left(\frac{d_{j,k}}{200}\right) * 60$$

Pada formula diatas $tr_{j,k}$ mewakili waktu perjalanan dalam menit dari permintaan *j* ke permintaan *k*. $d_{j,k}$ mewakili jarak dari permintaan *j* ke permintaan *k*. Penulis menetapkan kecepatan rata-rata untuk setiap pengguna yaitu 12 kilometer per jam atau 200 meter per menit.

3.3.2 Analisis Estimasi Waktu

Estimasi waktu mulai (est_i) ditentukan dengan menjumlahkan waktu akhir dari permintaan terdekat (ft_i) dengan waktu perjalanan ($tr_{j,k}$).

$$est_i = ft_i + tr_{j,k}$$

Setelah mendapatkan estimasi waktu mulai, estimasi waktu selesai dari permintaan tersebut sudah dapat ditentukan dengan menjumlahkan estimasi waktu mulai dengan durasi permintaan (t_i).

$$eft_i = est_i + t_i$$

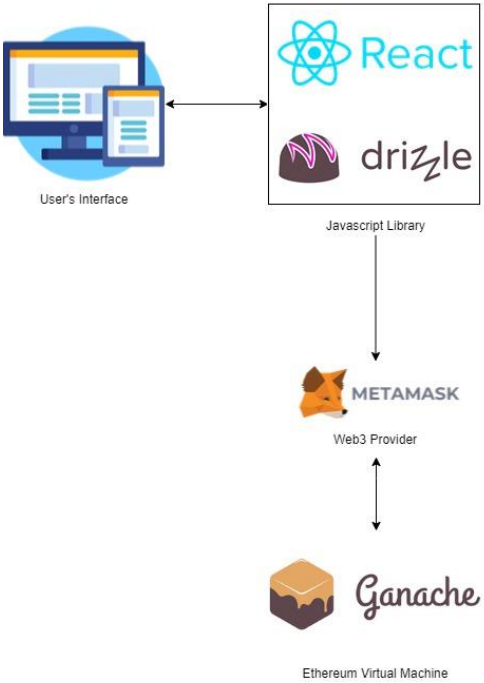
Estimasi waktu mulai dan waktu selesai dari permintaan tersebut akan dibandingkan dengan seluruh *RP*. Jika ditemukan waktu mulai dan waktu selesai dari rp_i yang tidak sesuai dengan estimasi waktu mulai dan estimasi waktu selesai, maka est_i dan eft_i akan dihitung ulang berdasarkan rp_i.

3.4 Perancangan Sistem

Bagian ini menjelaskan desain dari sistem yang akan dibangun, *virtual machine* yang digunakan, tampilan antarmuka, serta cara tampilan antarmuka berkomunikasi dengan jaringan blockchain.

3.4.1 Desain Umum Sistem

Sistem ini merupakan aplikasi terdesentralisasi yang di-*deploy* pada jaringan *blockchain* pribadi. Aplikasi ini membutuhkan aplikasi tambahan pada browser yaitu *metamask* sebelum digunakan. Berikut merupakan diagram tentang desain sistem pada aplikasi ini.



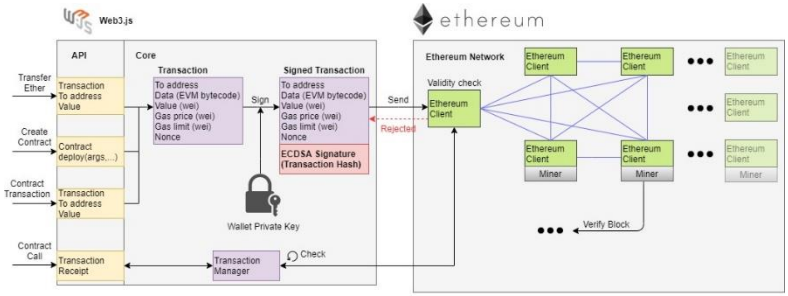
Gambar 3.2 Desain Sistem

Tampilan antarmuka pada sistem ini menggunakan *library* react.js dan drizzle. React.js berfungsi untuk memberikan tampilan interaktif kepada pengguna dan drizzle bertanggung jawab atas setiap transaksi yang dilakukan oleh pengguna. Drizzle menggunakan *library* web3.js yang digunakan untuk berkomunikasi ke jaringan *blockchain* menggunakan aplikasi tambahan yaitu *metamask*.

Setiap data yang akan ditampilkan dan permintaan yang akan dilakukan oleh pengguna akan dieksekusi menggunakan drizzle. Drizzle akan mengirimkan *request* ke web3 *provider* yang digunakan, pada aplikasi ini penulis menggunakan *metamask*. Metamask berfungsi sebagai alat komunikasi antara tampilan antarmuka pengguna dengan jaringan *blockchain*.

3.4.2 Privasi Data pada Ethereum

Ethereum melindungi data privasi pengguna dengan menggunakan sistem *public-key cryptography*. Sistem ini juga merupakan sistem yang sama yang digunakan oleh *Bitcoin*. *Public-key cryptography* bekerja dengan menggunakan sepasang kunci, alamat publik atau kunci pribadi yang dapat disebarluaskan, dan kunci pribadi yang hanya diketahui oleh pemilik akun.

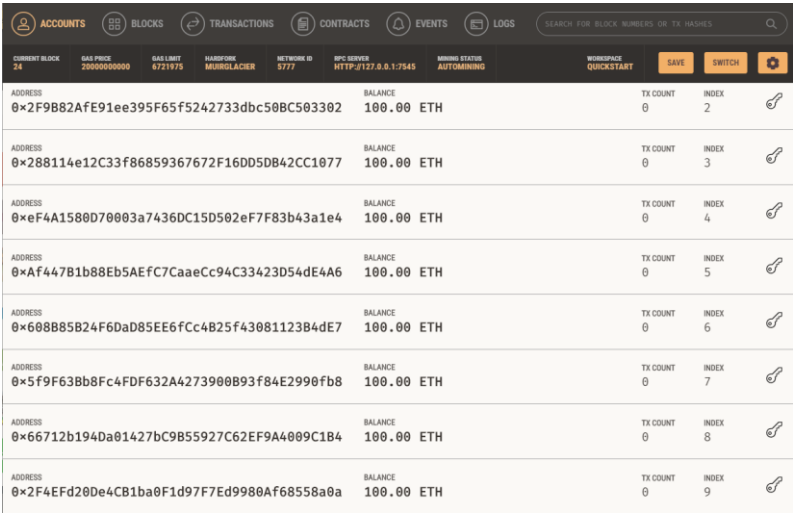


Gambar 3.3 Ilustrasi kerja *ethereum*

Dalam *public-key cryptography*, setiap pengguna memiliki kunci publik dan kunci pribadi. Alamat publik merupakan *hash* dari kunci publik pengguna, alamat ini digunakan untuk menerima transaksi. Setiap transaksi yang dilakukan oleh pengguna akan dienkripsi menggunakan kunci pribadi, dan akan diverifikasi oleh *miner* menggunakan kunci publik. Setiap transaksi yang dilakukan oleh pengguna akan ditandatangani secara digital menggunakan kunci pribadi. Kunci publik yang dimiliki pengguna berfungsi untuk memverifikasi transaksi yang telah dienkripsi menggunakan kunci pribadi, jika kedua kunci tersebut valid, maka transaksi akan disimpan pada *blockchain* dan *coin* yang digunakan dalam transaksi akan dikirim ke alamat yang dituju.

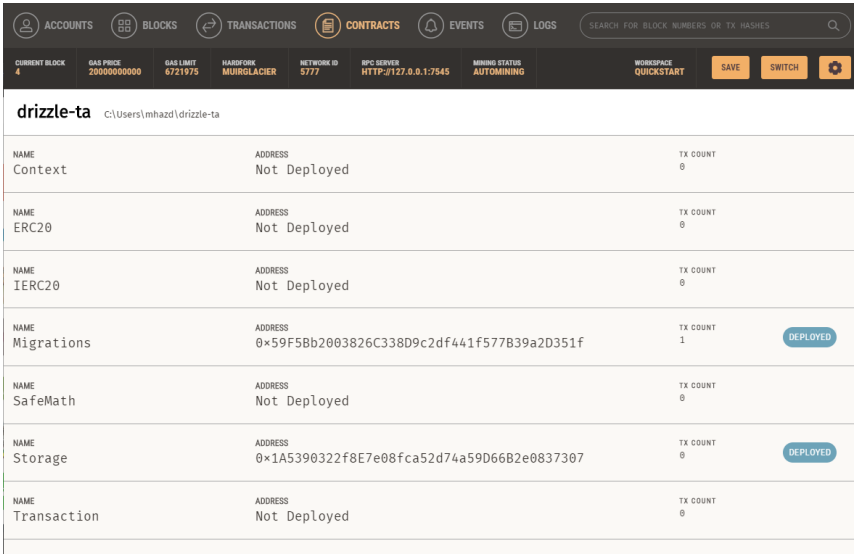
3.4.3 *Ethereum Virtual Machine*

EVM yang digunakan pada tugas akhir ini adalah *ganache*. Dibawah memungkinkan untuk node Ethereum pribadi pada jaringan lokal. Berikut merupakan tampilan antarmuka pada aplikasi *ganache*.



Gambar 3.4 Tampilan Ganache

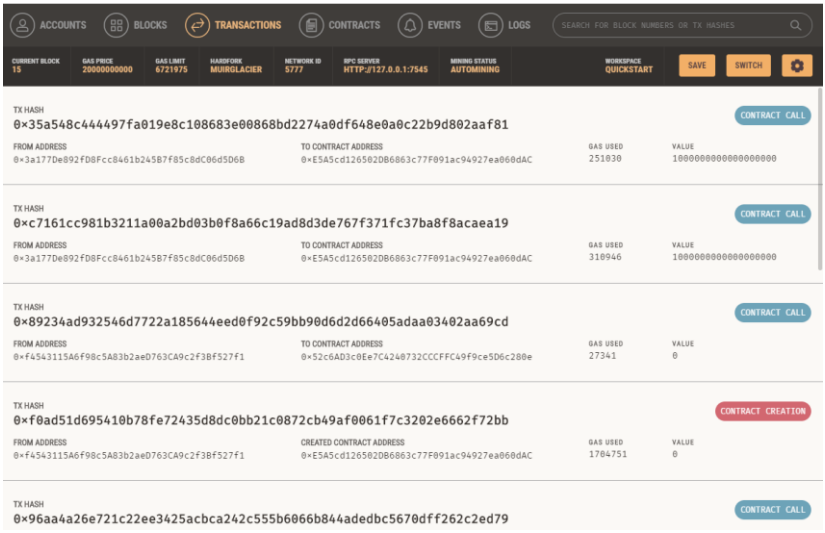
Ganache UI menggunakan port 7545 pada jaringan *localhost*. *Metamask* perlu membuat sambungan ke alamat tersebut agar dapat mengelola data pada *blockchain*. Setiap transaksi pada *ganache* di *mining* secara instan, yang berarti transaksi akan diproses lebih cepat dari jaringan Ethereum pada umumnya.



Gambar 3.5 Menu *contracts* pada Ganache

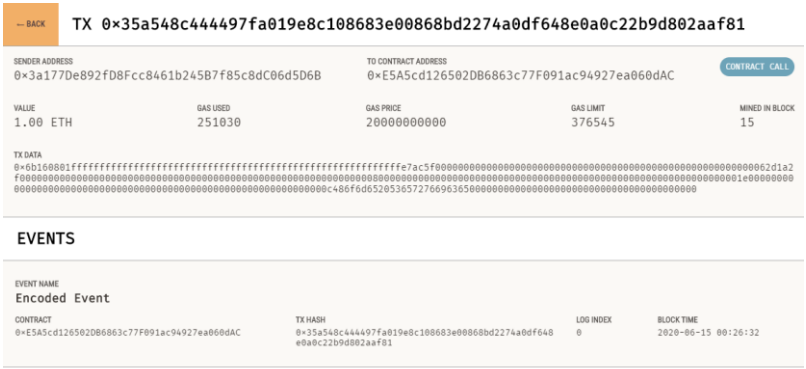
Gambar diatas merupakan menu *contracts* yang berisi daftar *smart contract* yang tersedia pada direktori beserta *smart contract* apa saja yang digunakan pada aplikasi. Storage merupakan *smart contract* yang digunakan dalam aplikasi ini, dan *Migrations* merupakan *smart contract* yang digunakan untuk menguji fungsi dasar seperti *call* dan *send*.

Setiap *smart contract* yang di-deploy memiliki alamat publik. Sama seperti alamat publik yang dimiliki setiap pengguna pada umumnya, alamat publik yang dimiliki oleh *smart contract* berfungsi sebagai alamat yang dituju saat *smart contract* tersebut digunakan. *Smart contract* tidak memiliki alamat pribadi.



Gambar 3.6 Menu *transactions* pada Ganache

Setiap transaksi yang dilakukan oleh pengguna pada aplikasi dapat dilihat pada menu *transactions* pada *ganache*. *Contract call* merupakan transaksi yang dilakukan oleh pengguna. *Contract creation* merupakan transaksi yang dilakukan oleh pengembang aplikasi yang bertujuan untuk men-*deploy smart contract* ke jaringan *blockchain.*



Gambar 3.7 Detail transaksi pada Ganache

Setiap transaksi yang dilakukan menyimpan informasi berupa hash transaksi, alamat publik dari pengguna yang melakukan transaksi, alamat publik dari *smart contract* yang digunakan, gas, dan data dari transaksi tersebut. Transaksi yang telah dikonfirmasi oleh pengguna akan disimpan pada satu blok, dan menunggu untuk diverifikasi oleh *miner* sebelum blok tersebut dihubungkan ke blok lainnya. Setelah transaksi diverifikasi, informasi tentang nomor blok yang digunakan dan waktu blok dihubungkan akan dicatat pada detail transaksi.

Beberapa transaksi memiliki *value*, seperti pada Gambar 3.7. *Value* merupakan jumlah *ether* yang dikirim bersamaan saat transaksi dilakukan, *ether* tersebut akan selalu disimpan di jaringan *blockchain* hingga suatu kondisi yang sudah ditentukan tercapai. Pada aplikasi ini, transaksi yang memiliki *value* hanya transaksi yang dilakukan oleh pemohon permintaan saat membuat suat permintaan, *value* tersebut merupakan bayaran dari permintaan tersebut. *Ether* yang telah disimpan oleh pemohon permintaan hanya bisa dicairkan jika pemohon membatalkan permintaan atau penyedia layanan menyelesaikan permintaan tersebut.

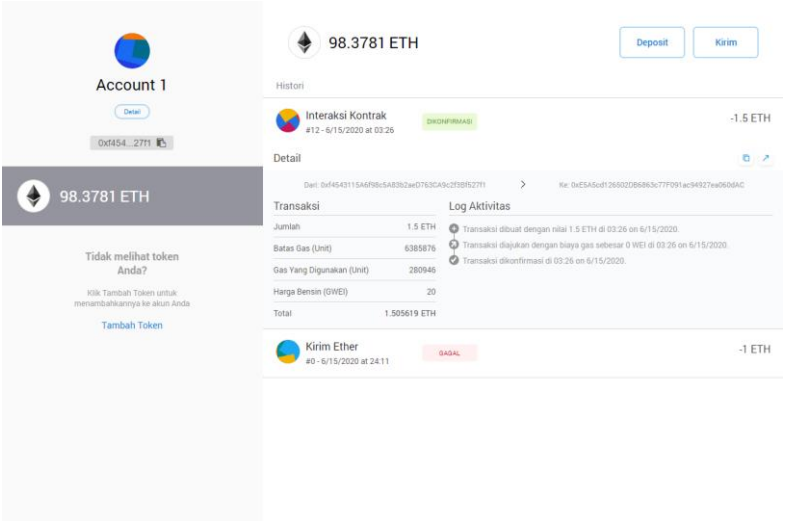
Gambar 3.8 Detail *block* pada Ganache

Setiap transaksi dilakukan, sebuah blok baru akan dibuat pada jaringan blockchain. Blok tersebut menyimpan informasi berupa hash dari blok tersebut, gas yang digunakan, waktu dan tanggal blok tersebut dikonfirmasi, dan hash dari transaksi yang disimpan pada blok tersebut. Seluruh informasi transaksi pada Gambar 3.7 disimpan pada blok diatas.

3.4.4 *Web3 Provider*

Web3 provider memungkinkan aplikasi untuk berkomunikasi dengan node Ethereum. *Web3 provider* dapat tersambung dengan node Ethereum lokal atau jarak jauh. *Web3 Provider* menyimpan informasi akun pengguna seperti alamat publik dan jumlah *ether* yang dimiliki, dan informasi ini akan digunakan pada web3.js. Selain itu, *web3 provider* juga bertanggung jawab atas setiap transaksi yang akan dilakukan oleh pengguna, dimana pengguna perlu mengkonfirmasi transaksi terlebih dahulu ke *web3 provider* sebelum transaksi tersebut dikirim ke *blockchain*.

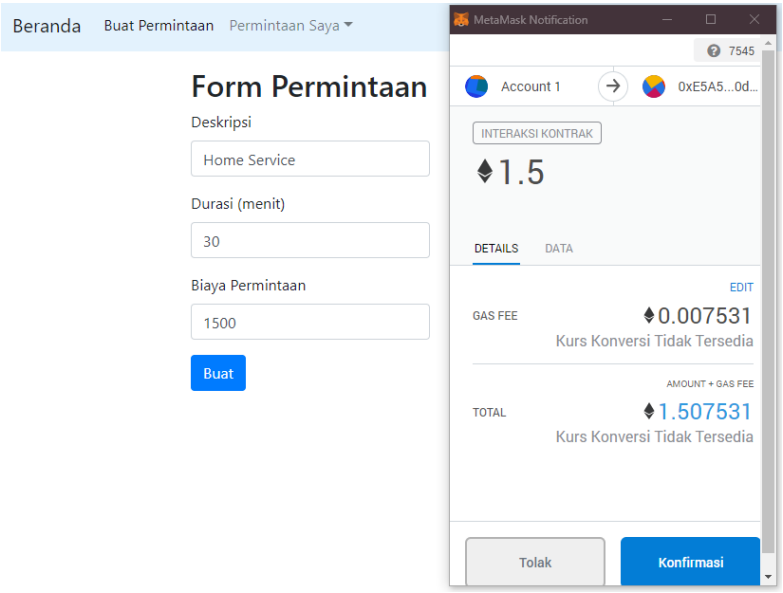
Aplikasi web dapat berkomunikasi dengan *web3 provider* dengan menggunakan *library* web3.js. Web3.js merupakan kumpulan modul yang dibuat khusus untuk berkomunikasi secara dua arah dengan web3 provider. Web3.js dapat menerima informasi yang disediakan oleh *web3 provider*.



Gambar 3.9 Detail akun Metamask

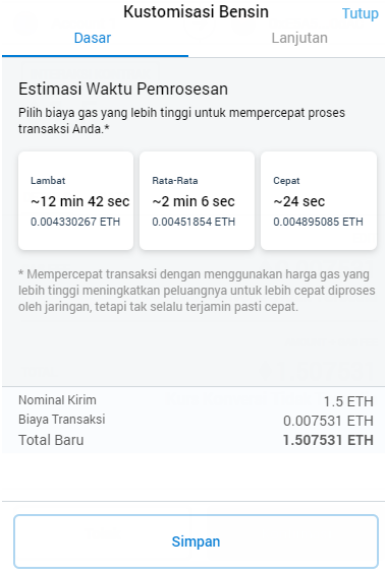
Metamask merupakan *web3 provider* yang penulis gunakan dalam pengembangan aplikasi. *Metamask* dapat diunduh dan digunakan sebagai ektensi yang tersedia pada beberapa browser. Gambar diatas merupakan tampilan untuk detail akun yang digunakan pada *metamask.*

Setiap akun memiliki informasi seperti alamat publik pengguna, jumlah e*ther* yang dimiliki, dan riwayat transaksi yang dilakukan oleh pengguna. Dari seluruh informasi tersebut, web3.js hanya dapat mengambil informasi alamat publik pengguna dan jumlah *ether* yang dimiliki. Web3.js tidak dapat mengambil informasi seperti riwayat transaksi pengguna dan alamat pribadi pengguna.



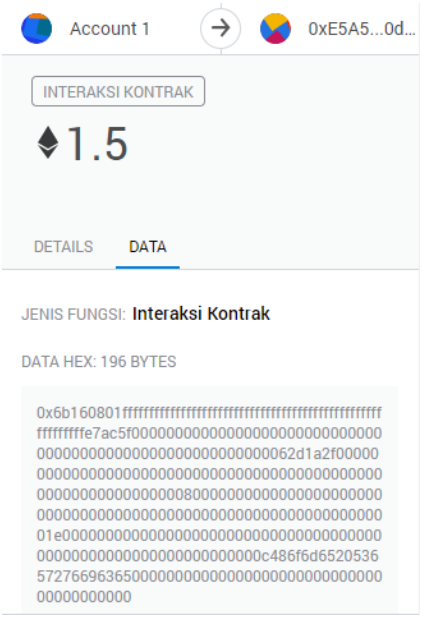
Gambar 3.10 Notifikasi transaksi pada Metamask

Gambar 3.10 merupakan jendela transaksi yang ditampilkan oleh *metamask* jika pengguna melakukan suatu transaksi pada aplikasi. Biaya permintaan pada form merupakan *ether* yang dikirim pada transaksi. Pengguna dapat memodifikasi biaya gas yang akan digunakan pada transaksi tersebut.



Gambar 3.11 Kostumisasi gas transaksi

Diatas merupakan tampilan pada jendela transaksi jika pengguna ingin memodifikasi biaya gas yang digunakan. Jumlah biaya gas yang digunakan akan mempengaruhi kecepatan transaksi untuk diverifikasi oleh *miner*. Selain itu, jendela transaksi juga akan menampilkan informasi terkait data yang akan dikirim pada transaksi.



Gambar 3.12 Data pada jendela transaksi

Data dari transaksi dapat dilihat pada jendela transaksi. Data yang ditampilkan adalah data yang telah dienkripsi. Data tersebut tidak akan disimpan dalam detail riwayat transaksi dalam *metamask*.

[*Halaman ini sengaja dikosongkan*]

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan tentang implementasi dari perancangan sistem yang telah dijelaskan pada bab sebelumnya. Setiap sistem akan dibahas proses pembuatannya dilengkapi dengan dan potongan kode dari sistem.

4.1 Lingkungan Implementasi

Digunakan lingkungan implementasi dengan spesifikasi perangkat lunak dan perangkat keras seperti terlihat pada :

Tabel 4.1 Spesifikasi Lingkungan Implementasi

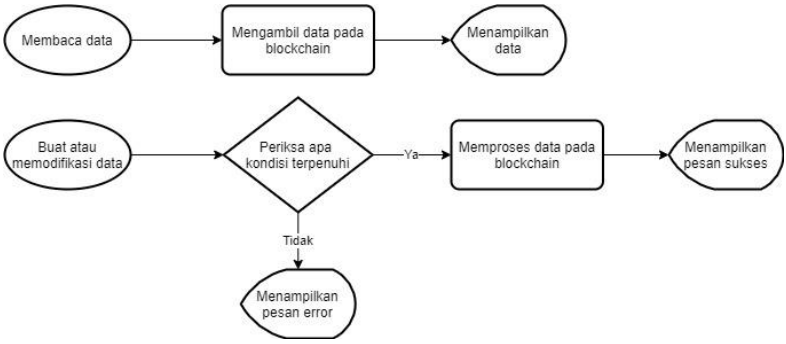
NO.	JENIS PERANGKAT	SPESIFIKASI
1	Perangkat Keras	• *Processor* Intel Core i7-7700HQ CPU @ 2.80GHz • *Memory* 16GB DDR4
2	Perangkat Lunak	• Sistem operasi Windows 10 Pro 64-bit • Google Chrome versi 83.0.4103.106 • Bahasa pemrograman JavaScript dan Solidity • Node.js versi 12.16.3 • Web3.js versi 1.2.1 • Truffle versi 5.1.7 • Ganache versi 2.4.0 • Metamask versi 7.7.9

4.2 Implementasi Basis Data

Aplikasi ini menggunakan *blockchain* sebagai basis data, penulis menggunakan aplikasi *Ganache* untuk menjalankan jaringan pribadi *Ethereum*.

Data yang disimpan pada blockchain biasanya hanya berupa transaksi *ether* biasa, di mana data hanya terdiri dari alamat publik yang dituju dan jumlah *ether* yang dikirim. *Smart contract* dapat menyimpan data selain alamat publik dan jumlah *ether* ke *blockchain*. Dengan demikian menggunakan *smart contract* pada setiap transaksi juga dapat berfungsi sebagai basis data, karena setiap data yang dikirim pada setiap transaksi menggunakan *smart contract* akan disimpan dalam suatu blok di jaringan *blockchain*.

Smart contract bertanggung jawab untuk setiap transaksi yang dilakukan oleh pengguna pada aplikasi. Karena data yang disimpan adalah *plain text*, jadi semua orang dapat mengakses data tersebut. Tetapi tidak semua orang dapat memodifikasi data yang telah disimpan dalam *smart contract*, data hanya dapat dimodifikasi ketika kondisi pada *smart contract* terpenuhi.



Gambar 4.1 Alur proses *smart contract*

4.2.1 Inisiasi Variable

Setiap data permintaan disimpan dalam sebuah *struct* yang diberi nama *Request* dalam *smart contract*, berikut merupakan isi dari *strict tersebut.*

Tabel 4.2 Tabel basis data *struct "Request"*

Field	*Data type*	Catatan
id	*Integer*	Nomor identitas permintaan
requestDesc	*String*	Deskripsi permintaan
requestEther	*Unsigned Integer*	*Ether* yang disimpan pada permintaan
startTime	*Unsigned Integer*	Waktu mulai permintaan
duration	*Unsigned Integer*	Durasi permintaan
endTime	*Unsigned Integer*	Waktu selesai permintaan
requestor	*Address*	Alamat publik pembuat permintaan
provider	*Address*	Alamat publik penyedia layanan
providerAddress	*Array of Address*	Alamat publik penyedia layanan yang mendaftar
providerTime	*Array of Unsigned Integer*	Waktu mulai setiap penyedia layanan yang mendaftar
providerTotal	*Unsigned Integer*	Jumlah penyedia layanan yang mendaftar
latitude	*Integer*	Garis lintang dari lokasi permintaan
longitude	*Integer*	Garis bujur dari lokasi permintaan

isCancel	*Boolean*	Status permintaan dibatalkan
isConfirm	*Boolean*	Status permintaan selesai dikerjakan

Beberapa data tidak disimpan dalam *struct* untuk melindungi data sensitif pengguna. Data tersebut disimpan dalam tipe data *mapping* dengan menggunakan alamat publik pengguna sebagai *Key*. *Mapping* memiliki fungsi yang sama seperti *hash table*. Basis data *mapping* yang digunakan dapat dilihat pada Tabel 3.4.

Tabel 4.3 Tabel basis data *mapping*

Field	*Key Type / Value Type*	Catatan
request	*Unsigned Integer / Struct "Request"*	Menyimpan data *struct "Request"* berdasarkan *id*
requestorReqList	*Address to Array / Integer*	Seluruh permintaan yang dibuat
providerCount	*Address / Unsigned Integer*	Jumlah permintaan yang dilayani
providerReqList	*Address to Array / Integer*	Seluruh permintaan yang dilayani
providerEther	Address / Unsigned Integer	Jumlah *Ether* yang dimiliki pengguna

Berikut adalah beberapa variabel global yang digunakan.

Tabel 4.4 Tabel basis data variabel global

Field	*Data type*	Catatan
countRequest	*Integer*	Jumlah seluruh permintaan
availableRequest	*Integer*	Jumlah permintaan yang tersedia

4.2.2 Implementasi Fungsi *Smart Contracts*

Seperti alur pada Gambar 4.1, kondisi pada fungsi yang diprogram dalam *smart contract* mempengaruhi keamanan data dari perubahan yang tidak diinginkan. Setiap transaksi mencatat alamat publik pengguna yang melakukan transaksi (*msg.sender*) dan jumlah *ether* yang dikirim pada transaksi tersebut (*msg.value*).

4.2.2.1 *Fungsi Membuat Permintaan*

Fungsi ini bertujuan untuk menginisiasi permintaan yang baru dibuat.

***Algorithm* 1**: Membuat permintaan

Input : *latitude (lat), longitude (lng), description (desc), duration (t)*

```
1.  require msg.value > 0
2.     countRequest ← countRequest + +
3.     availableRequest ← availableRequest + +
4.     request[countRequest].id ← countRequest
5.     request[countRequest].countRequest ← desc
6.     request[countRequest].startTime ← 0
7.     request[countRequest].duration ← t
8.     request[countRequest].endTime ← 0
9.     request[countRequest].latitude ← lat
10.    request[countRequest].longitude ← lng
11.    request[countRequest].requestEther ← msg.value
12.    request[countRequest].providerTotal ← 0
13.    request[countRequest].isConfirm ← false
14.    request[countRequest].isCancel ← false
15.    request[countRequest].requestor ← msg.sender
16.    requestorReqList[msg.sender] ← insert countRequest
17. else
18.    return false
```

Pada *Algorithm 1* variabel *countRequest* bertujuan untuk menghitung jumlah total permintaan sekaligus untuk menginisiasi *id* untuk permintaan yang akan dibuat. Variabel *availableRequest* hanya digunakan pada *front-end* untuk membuat *rendering* pada aplikasi lebih ringan. Variabel *requestorReqList* merupakan *array* dinamis dua dimensi berfungsi untuk menyimpan seluruh permintaan yang dikerjakan.

4.2.2.2 *Fungsi Membatalkan Permintaan*

Fungsi ini digunakan oleh pembuat permintaan untuk membatalkan suatu permintaan. Jika permintaan dibatalkan, *ether* yang disimpan pada permintaan tersebut akan langsung dipindahkan ke variabel *providerEther* milik pembuat permintaan.

***Algorithm* 2**: Membatalkan permintaan

```
Input : id (inpId)
1.  require request[inpId].requestor = msg.value
    and request[inpId].provider = null address
2.    availableRequest ← availableRequest++
3.    request[inpId].isCancel ← true
4.    providerEther[request[inpId].requestor] ←
      providerEther[request[inpId].requestor] +
      request[inpId].requestEther
5.    request[inpId].requestEther ← 0
```

Require merupakan kondisi yang perlu dipenuhi sebelum menggunakan fungsi. Jika *require* tidak dipenuhi maka fungsi tersebut akan memberikan pesan *error* ketika digunakan.

4.2.2.3 *Fungsi Memilih Permintaan*

Fungsi ini digunakan oleh penyedia layanan ketika memilih permintaan untuk dikerjakan.

***Algorithm* 3**: Memilih permintaan

```
Input : id (inpId), start time (inpTime)
1.  require request[inpId].providerAddress does
    not contain msg.sender
2.    request[inpId].providerAddress ← insert
      msg.sender
3.    request[inpId].providerTime   ←   insert
      inpTime
4.    request[inpId].providerTotal ←
      request[inpId].providerTotal++
```

4.2.2.4 *Fungsi Memilih Penyedia Layanan*

Fungsi ini digunakan oleh pemohon permintaan untuk memilih penyedia layanan yang telah mendaftar menggunakan *Algorithm 3.*

***Algorithm* 4**: Memilih penyedia layanan
Input : *id (inpId), indeks (inpIndex)*
1. **require** $request[inpId].requestor = msg.sender$ **and** $request[inpId].provider = null\ address$
2. *request[inpId].provider ← request[inpId].providerAddress[inpIndex]*
3. *request[inpId].startTime ← request[inpId].providerTime[inpIndex]*
4. *request[inpId].endTime ← request[inpId].Time[inpIndex]+(request[inpId].duration*1 minute)*
5. *providerReqList[request[inpId].providerAddress[inpIndex]] ←* **insert** *inpId*
6. *providerCount[request[inpId].providerAddress[inpIndex]] ← providerCount[request[inpId].providerAddress[inpIndex]]++*
7. *availableRequest ← availableRequest--*

4.2.2.5 *Fungsi Menyelesaikan Permintaan*

Fungsi ini digunakan oleh pembuat permintaan untuk menyelesaikan permintaan. Setelah permintaan diselesaikan, *ether* yang disimpan pada permintaan tersebut akan langsung dipindahkan ke variabel *providerEther* milik penyedia layanan.

***Algorithm* 5**: Menyelesaikan permintaan

Input : *id (inpId)*

1. **requre** $request[inpId].requestor = msg.sender$ **and** $request[inpId].provider != null\ address$ **and** $request[inpId].isConfirm = false$ **and** $request[inpId].isCancel = false$
2. `request[inpId].isConfirm ← true`
3. `providerEther[request[inpId].provider] ← providerEther[request[inpId].provider + request[inpId].requestEther`
4. `request[inpId].requestEther ← 0`
5. `providerCount[request[inpId].provider] ← providerCount[request[inpId].provider]--`

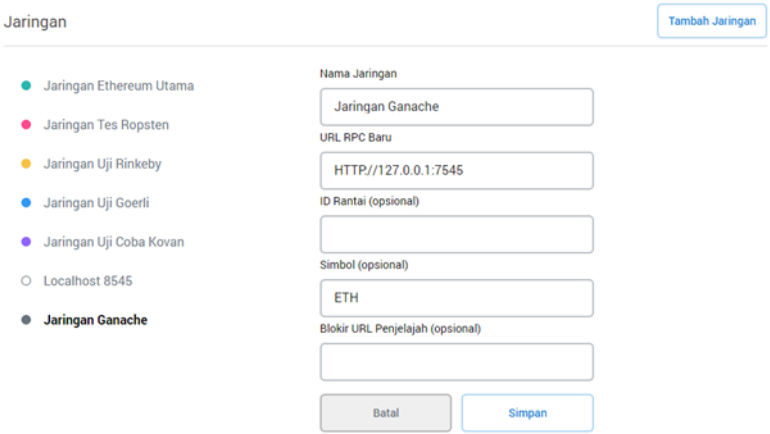
4.2.2.6 *Fungsi Mencairkan Ether*

Fungsi ini dapat digunakan oleh setiap pengguna untuk mencairkan *ether* yang disimpan pada *smart contract*. *Ether* yang disimpan pada variabel *providerEther* akan ditransfer ke akun *wallet* pengguna saat fungsi digunakan.

***Algorithm* 6**: Mencairkan *ether*
Input : –
1. $temp \leftarrow providerEther[msg.sender]$
2. $providerEther[msg.sender] \leftarrow 0$
3. **transfer** $temp$ **to** $msg.sender$

4.3 Implementasi Tampilan Antarmuka

Tampilan antarmuka dalam aplikasi ini dibuat menggunakan bahasa pemrograman *javascript* dengan *library react.js*. Sebelum pengguna dapat mengakses tampilan antarmuka, pengguna perlu mengatur jaringan di *Metamask* sesuai dengan *RPC Server* yang diberikan oleh *Ganache*.



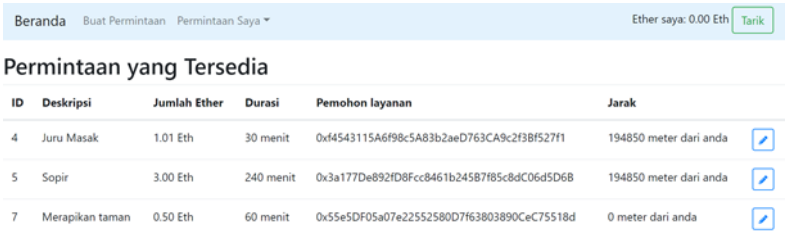
Gambar 4.2 Pengaturan jaringan RPC pada *Metamask*

Ganache menggunakan *RPC Server* lokal yang dapat diakses pada *port* 7545. Setelah pengguna mengatur jaringan pada

Metamask, pengguna akan langsung diarahkan ke halaman beranda aplikasi.

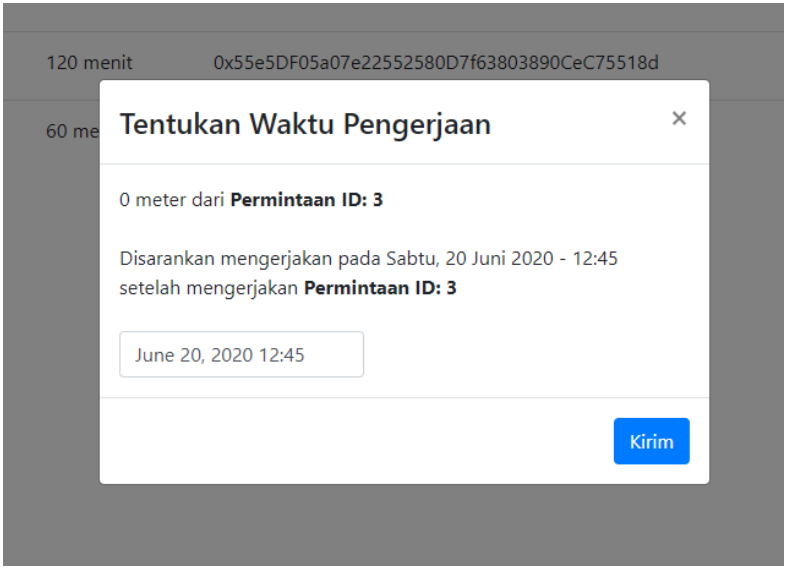
4.3.1 Halaman Beranda

Halaman beranda menampilkan daftar permintaan yang belum memiliki penyedia layanan atau tersedia untuk dikerjakan. Setiap permintaan memiliki informasi seperti deskripsi permintaan, jumlah *ether*, durasi, alamat publik pemohon, dan jarak permintaan dari pengguna yang sedang mengakses halaman.



Gambar 4.3 Tabel daftar permintaan yang tersedia

Tombol yang terletak di sebelah kanan tabel permintaan digunakan ketika pengguna aplikasi ingin mengerjakan permintaan tersebut. Tombol tersebut akan membuka jendela *input* untuk memberikan waktu mulai permintaan yang diinginkan.



Gambar 4.4 Tampilan jendela *input* saat memilih permintaan

Jendela *input* yang muncul menyediakan beberapa informasi, yaitu jarak terdekat permintaan tersebut dari permintaan yang akan dikerjakan oleh pengguna, dan rekomendasi untuk waktu pengerjaan. Jarak terdekat dari permintaan yang dipilih dapat ditentukan dengan menggunakan *Algorithm 7.*

***Algorithm* 7**: Menentukan permintaan terdekat

Require : rr_i, RP
Result : Rrp, Rd, Rt, Rst, Rft

1. $Rrp \leftarrow rp_i\ id$
2. $Rd \leftarrow distance\ rr_i\ to\ rp_i$
3. $Rt \leftarrow rr_i\ duration$
4. $Rst \leftarrow estimated\ start\ time$
5. $Rft \leftarrow estimated\ finish\ time$
6. **for** $i \leftarrow 1\ to\ RP.length$ **do**
7. **if** $i = 1$
8. $Rrp \leftarrow id\ of\ rp_i$
9. $Rd \leftarrow distance\ rr_i\ to\ rp_j$
10. $Rst \leftarrow rp_i\ start\ time + (\left(\frac{Rd}{200}\right) * 1\ minute)$
11. $Rft \leftarrow Rst + (Rt * 1\ minute)$
12. **end if**
13. **if** $d > distance\ rr_i\ to\ rp_j$
14. $Rrp \leftarrow id\ of\ rp_i$
15. $Rd \leftarrow distance\ rr_i\ to\ rp_j$
16. $Rst \leftarrow rp_i\ start\ time + (\left(\frac{Rd}{200}\right) * 1\ minute)$
17. $Rft \leftarrow Rst + (Rt * 1\ minute)$
18. **end if**
19. **end for**
20. return Rrp, Rd, Rt, Rst, Rft

Setelah hasil *Algorithm 7* diperoleh, variabel *Rd* dan *Rrp* akan ditampilkan pada jendela *input* seperti pada Gambar 4.4 sebagai jarak dan permintaan terdekat. Variabel lainnya akan digunakan untuk menentukan rekomendasi waktu pengerjaan permintaan tersebut dengan menggunakan *Algorithm 8.*

***Algorithm* 8**: Rekomendasi waktu mulai

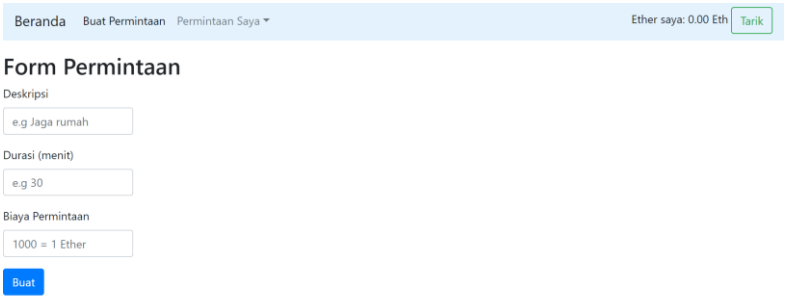
Require : Rrp, *Rd, Rt, Rst, Rft*
Result : Rrp, Rst, Rft

21. **for** $i \leftarrow 1$ *to* $RP.length$ **do**
22. **if** ($Rst < rp_i\ finish\ time$ **and** $Rst > rp_i\ start\ time$) **or**
23. ($Rft < rp_i\ finish\ time$ **and** $Rft > rp_i\ start\ time$) **or**
24. ($Rst > rp_i\ start\ time$ **and** $Rft < rp_i\ finish\ time$) **or**
25. ($Rst < rp_i\ start\ time$ **and** $Rft > rp_i\ finish\ time$)
26. $Rrp \leftarrow id\ of\ rp_i$
27. $Rst \leftarrow rp_i\ start\ time + (\left(\frac{Rd}{200}\right) * 1\ minute)$
28. $Rft \leftarrow Rst + (Rt * 1\ minute)$
29. **end if**
30. **end for**
31. return Rrp, Rst, Rft

Variabel Rrp dan Rst pada *Algorithm 8* akan ditampilkan pada baris kedua pada rekomendasi waktu mulai pengerjaan. *Algorithm 7* dan *Algorithm 8* dieksekusi menggunakan *javascript*.

4.3.2 Halaman untuk Membuat Permintaan

Halaman ini dapat diakses oleh semua pengguna yang ingin membuat permintaan. Formulir untuk membuat permintaan berisi deskripsi permintaan, durasi, dan biaya untuk mengerjakan permintaan (*ether*).

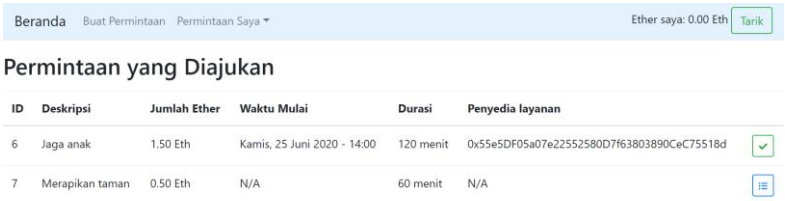


Gambar 4.5 Formulir membuat permintaan

Data yang diisi pada formulir akan digunakan sebagai *input* ke *Algorithm 1*. *Latitude* dan *longitude* pada algoritma tersebut menggunakan koordinat pada browser yang digunakan pengguna.

4.3.3 Halaman Pemohon Permintaan

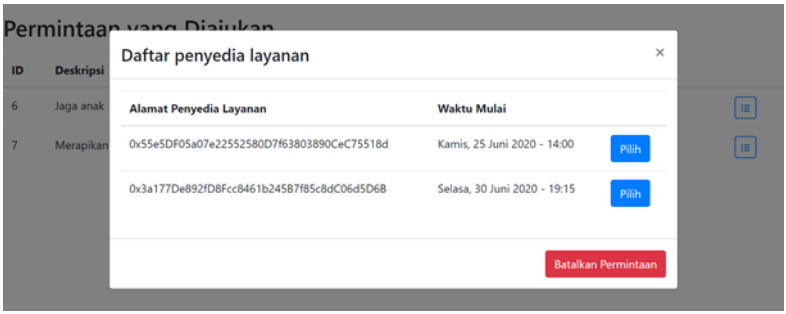
Halaman ini menampilkan daftar permintaan yang telah dibuat oleh pengguna. Informasi pada halaman ini berbeda untuk setiap pengguna, berdasarkan pada akun yang digunakan pada *Metamask*.



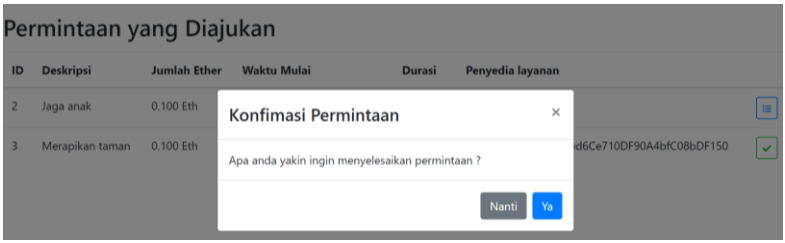
Gambar 4.6 Tabel daftar permintaan yang diajukan

Informasi untuk waktu mulai dan penyedia layanan akan ditampilkan ketika pelamar telah memilih penyedia layanan yang

diinginkan. Ada dua jenis tombol di sisi kanan tabel, yaitu tombol biru untuk memilih penyedia layanan dan tombol hijau untuk menyelesaikan permintaan. Tombol biru akan menampilkan daftar penyedia layanan yang mendaftar pada permintaan tersebut. Sementara tombol hijau akan menampilkan jendela konfirmasi untuk menyelesaikan permintaan.



Gambar 4.7 Tabel daftar penyedia layanan

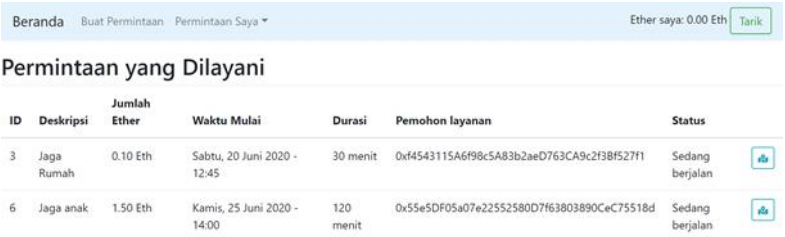


Gambar 4.8 Jendela konfirmasi

Jendela konfirmasi mencatat *id* dari permintaan yang dipilih dan akan menggunakannya sebagai *input* pada *Algorithm 5*. Pemohon permintaan hanya dapat membatalkan permintaan jika penyedia layanan belum ditentukan, *Algorithm 2* digunakan untuk membatalkan permintaan.

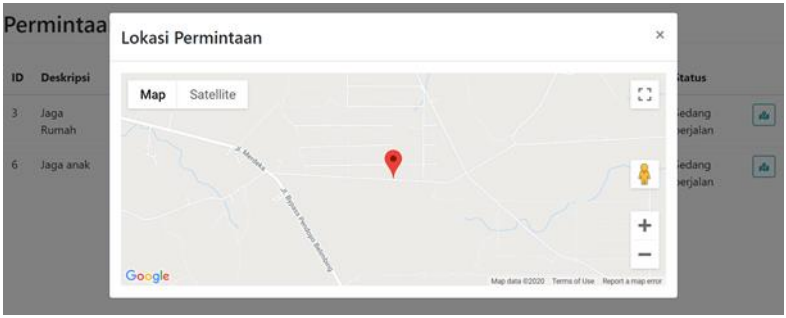
4.3.4 Halaman Penyedia Layanan

Halaman ini menampilkan daftar permintaan yang akan atau telah dikerjakan oleh pengguna. Sama seperti halaman pemohon permintaan, informasi yang ditampilkan pada halaman ini berbeda berdasarkan akun yang digunakan.



Gambar 4.9 Tabel daftar permintaan yang dilayani

Informasi yang ditampilkan pada Gambar 4.9 kira-kira sama dengan halaman pemohon permintaan, hanya berbeda di bagian alamat yang ditampilkan dan tombol di sisi kanan tabel. Alamat yang ditampilkan adalah alamat publik dari pemohon permintaan, dan tombol di sisi kanan tabel berfungsi untuk melihat lokasi permintaan.

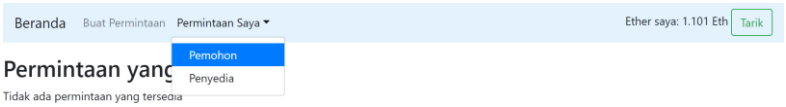


Gambar 4.10 Lokasi Permintaan

Untuk menampilkan lokasi permintaan, penulis menggunakan *Google Cloud Platform* untuk mendapatkan API untuk *Google Maps* dan *library google-maps-platform* untuk menampilkan peta.

4.3.5 Menu Navigasi

Menu navigasi pada aplikasi ini terdiri dari beranda, buat permintaan, permintaan saya sebagai pemohon, dan permintaan saya sebagai penyedia.



Gambar 4.11 Menu navigasi

Ether saya yang ditampilkan di sisi kanan menu adalah *ether* yang disimpan pada *smart contract* dan dapat dicairkan. Fungsi pada *Algorithm 6* akan digunakan ketika pengguna mengklik tombol tarik, dan jumlah *ether* yang ditampilkan akan ditransfer ke akun *wallet* yang digunakan

[*Halaman ini sengaja dikosongkan*]

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini akan dijelaskan tentang implementasi dari perancangan sistem pada tugas akhir ini yang telah dijelaskan pada bab sebelumnya.

5.1 Lingkungan Uji Coba

Digunakan lingkungan uji coba dengan spesifikasi perangkat keras dan perangkat lunak seperti terlihat pada Tabel 5.1.

Tabel 5.1 Spesifikasi lingkungan uji coba

NO.	JENIS PERANGKAT	SPESIFIKASI
1	Perangkat Keras	• *Processor* Intel Core i7-7700HQ CPU @ 2.80GHz • *Memory* 16GB DDR4
2	Perangkat Lunak	• Sistem operasi Windows 10 Pro 64-bit • Google Chrome versi 83.0.4103.106 • Bahasa pemrograman JavaScript dan Solidity • Node.js versi 12.16.3 • Web3.js versi 1.2.1 • Truffle versi 5.1.7 • Ganache versi 2.4.0 • Metamask versi 7.7.9

5.2 Skenario Uji Coba

Pada subbab ini akan menjelaskan skenario yang akan digunakan untuk melakukan pengujian terhadap implementasi yang dibuat sesuai dengan desain dan perancangan sistem. Skenario pengujian dibedakan menjadi dua bagian yaitu:

- **Uji Coba Fungsionalitas**

  Pengujian yang dilakukan untuk menguji fungsionalitas aplikasi.

- **Uji Coba Performa**

  Pengujian yang dilakukan untuk melihat performa pada penggunaan aplikasi.

5.2.1 Skenario Uji Coba Fungsionalitas

Uji coba fungsionalitas dibagi menjadi beberapa bagian yaitu pengujian membuat permintaan, memilih permintaan, memilih penyedia layanan, membatalkan permintaan, menyelesaikan permintaan, dan mencairkan *ether*.

5.2.1.1 *Membuat Permintaan*

Uji coba fungsionalitas membuat permintaan ini bertujuan untuk menguji apakah fungsi dalam bab 4.2.2.1 dapat digunakan. Uji coba ini dilakukan dengan pengguna mengakses halaman untuk membuat permintaan, mengisi formulir yang disediakan, dan mengkonfirmasi transaksi yang diberikan ketika formulir dikirim. Uji coba dianggap berhasil ketika transaksi yang dikonfirmasi sudah diverifikasi oleh *miner* dan permintaan yang dibuat dapat dilihat pada halaman permintaan saya sebagai pemohon.

5.2.1.2 *Memilih Permintaan*

Uji coba fungsionalitas memilih permintaan ini bertujuan untuk menguji apakah fungsi dalam bab 4.2.2.3 dapat digunakan. Pengguna hanya dapat memilih permintaan yang ingin dikerjakan jika permintaan tersebut belum memiliki penyedia layanan. Uji coba ini dilakukan dengan pengguna mengakses halaman beranda, memilih permintaan yang ingin dikerjakan dengan mengklik tombol yang disediakan pada setiap permintaan, mengisi kolom *input* yang diberikan, dan mengkonfirmasi transaksi pada Metamask. Uji coba dianggap berhasil ketika transaksi yang dikonfirmasi sudah diverifikasi oleh *miner* dan informasi yang dikirim akan dikirim ke pemohon permintaan untuk dipertimbangkan.

5.2.1.3 *Memilih Penyedia Layanan*

Uji coba fungsionalitas membatalkan permintaan ini bertujuan untuk menguji apakah fungsi dalam bab 4.2.2.4 dapat digunakan. Pemohon permintaan hanya dapat memilih penyedia layanan ketika permintaan tersebut belum memiliki penyedia layanan, penyedia layanan tidak dapat diganti jika sudah dipilih sebelumnya. Uji coba ini dilakukan dengan pengguna mengakses halaman permintaan permintaan saya sebagai pemohon, mengklik tombil tampilkan penyedia layanan, memilih penyedia layanan yang sesuai, dan mengkonfirmasi transaksi pada Metamask. Uji coba dianggap berhasil ketika transaksi yang dikonfirmasi sudah diverifikasi oleh *miner* dan kolom informasi penyedia layanan pada permintaan berubah menjadi alamat publik dari penyedia layanan yang dipilih.

5.2.1.4 *Membatalkan Permintaan*

Uji coba fungsionalitas membatalkan permintaan ini bertujuan untuk menguji apakah fungsi dalam bab 4.2.2.2 dapat digunakan. Pengguna hanya dapat membatalkan permintaan yang belum memiliki penyedia layanan, jika permintaan tersebut sudah memiliki penyedia layanan maka fungsi tidak akan bisa digunakan. Uji coba ini dilakukan dengan pengguna mengakses halaman permintaan saya sebagai pemohon, mengklik tombol tampilkan penyedia layanan, mengklik tombol batalkan permintaan, dan mengkonfirmasi transaksi pada Metamask. Uji coba dianggap berhasil ketika transaksi yang dikonfirmasi sudah diverifikasi oleh *miner* dan *ether* yang disimpan pada permintaan tersebut berubah menjadi 0. *Ether* yang disimpan pada permintaan akan dipindahkan ke variabel *providerEther* milik pemohon permintaan untuk dicairkan.

5.2.1.5 *Menyelesaikan Permintaan*

Uji coba fungsionalitas menyelesaikan permintaan ini bertujuan untuk menguji apakah fungsi dalam bab 4.2.2.5 dapat digunakan. Pengguna hanya dapat menyelesaikan permintaan jika penyedia layanan untuk permintaan tersebut telah dipilih. Uji coba ini dilakukan dengan pengguna mengakses halaman permintaan saya sebagai pemohon, mengklik tombol tandai selesai, dan mengkonfirmasi transaksi pada Metamask. Uji coba dianggap berhasil ketika transaksi yang dikonfirmasi sudah diverifikasi oleh *miner* dan *ether* yang disimpan pada permintaan tersebut berubah menjadi 0. Ether yang disimpan pada permintaan akan dipindahkan ke variabel *providerEther* milik penyedia layanan untuk dicairkan.

5.2.1.6 *Mencairkan Ether*

Uji coba fungsionalitas mencairkan *ether* ini bertujuan untuk menguji apakah fungsi dalam bab 4.2.2.6 dapat digunakan. Uji coba ini dilakukan dengan pengguna mengklik tombol yang ditampilkan di kanan atas aplikasi, dan mengkonfirmasi transaksi pada Metamask. Uji coba dianggap berhasil ketika transaksi yang dikonfirmasi telah diverifikasi oleh *miner* dan bertambahnya *ether* pada akun *wallet* yang digunakan pengguna.

5.2.2 Skenario Uji Coba Performa

Uji coba performa digunakan untuk menguji bagaimana performa aplikasi dalam menangani banyak permintaan dan banyak akses ke aplikasi. Hal tersebut bertujuan untuk melihat variasi waktu dan memori yang digunakan pada tampilan antarmuka untuk mengambil data dan menampilkan data dari *blockchain*.

5.3 Hasil Uji Coba

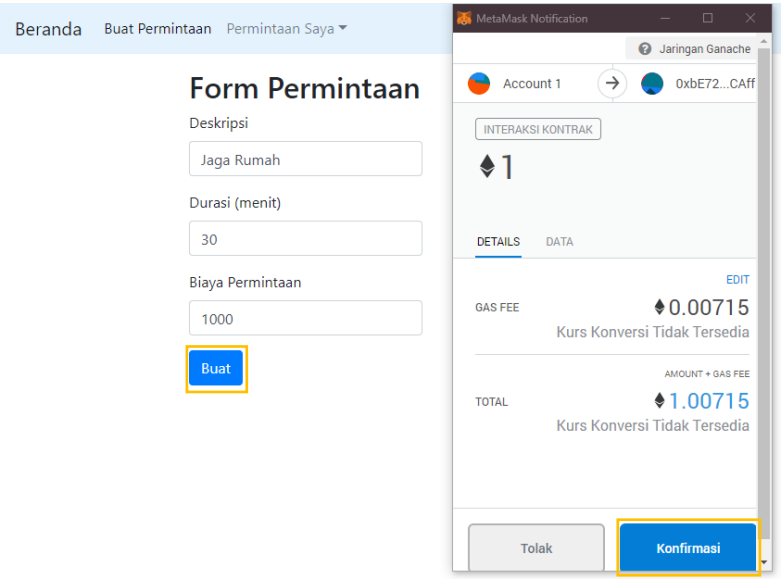
Berikut ini menjelaskan hasil uji coba berdasarkan skenario yang dijelaskan pada subbab sebelumnya.

5.3.1 Uji Fungsionalitas

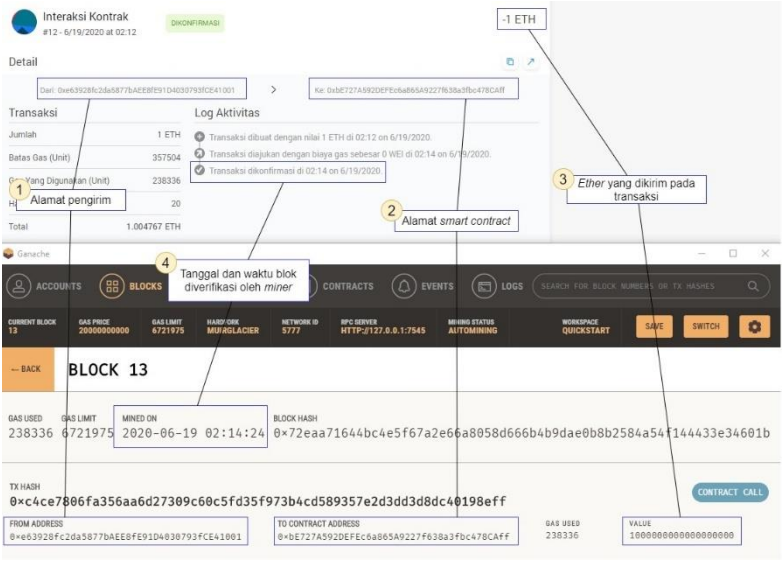
Berikut ini menjelaskan hasil dari pengujian fungsionalitas pada aplikasi yang sudah dibuat.

5.3.1.1 *Uji Fungsionalitas Membuat Permintaan*

Uji coba ini dilakukan dengan pengguna mengakses halaman buat permintaan, mengisi formulir pada halaman, dan mengkonfirmasi transaksi yang diberikan oleh Metamask ketika formulir dikirim seperti pada Gambar 5.1.



Gambar 5.1 Formulir permintaan



Gambar 5.2 Transaksi berhasil diverifikasi

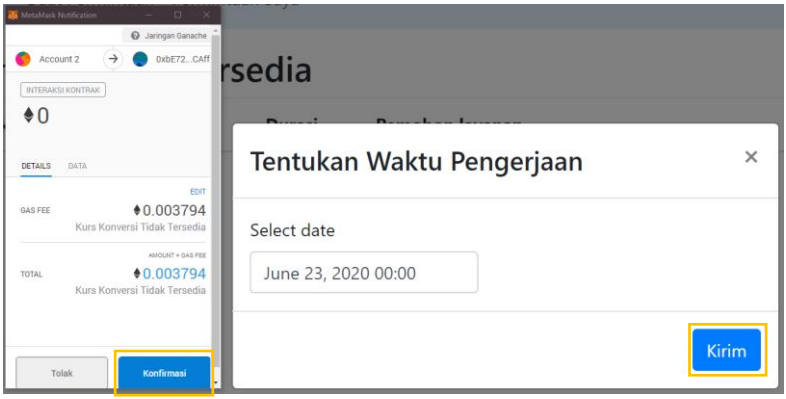
Untuk memeriksa apakah fungsi berhasil dijalankan adalah dengan melihat log aktivasi dalam riwayat transaksi pada Metamask, jika log aktivasi menunjukan informasi kapan blok tersebut dikonfirmasi maka data tersebut telah berhasil disimpan pada *blockchain.*

Tabel 5.2 Hasil uji fungsionalitas membuat permintaan

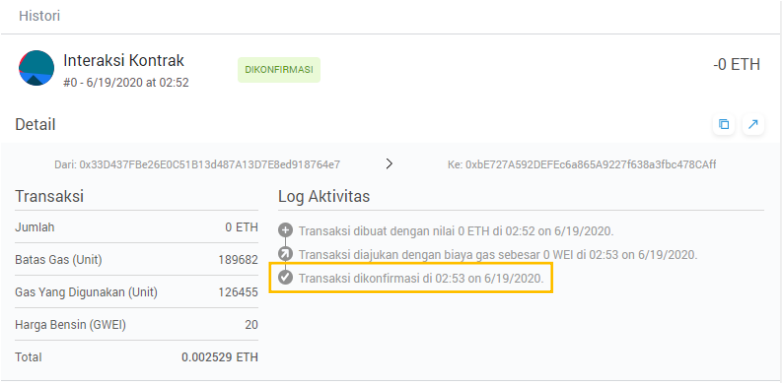
Fungsi	**Harapan**	**Hasil**
Membuat permintaan	Transaksi berhasil diverifikasi oleh *miner*	OK

5.3.1.2 *Uji Fungsionalitas Memilih Permintaan*

Uji coba ini dilakukan dengan pengguna mengakses halaman beranda, memilih permintaan yang ingin dikerjakan dengan mengklik tombol yang disediakan pada sisi kanan permintaan, mengisi kolom tanggal pengerjaan, dan mengkonfirmasi transaksi pada Metamask seperti pada Gambar 5.4.



Gambar 5.3 Formulir Permintaan



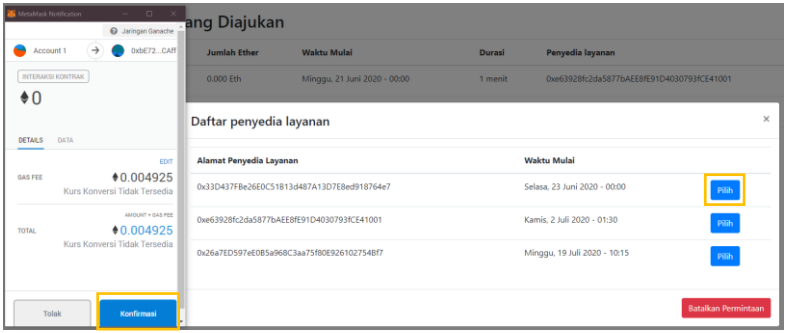
Gambar 5.4 Log aktivasi memilih permintaan

Tabel 5.3 Hasil uji fungsionalitas memilih permintaan

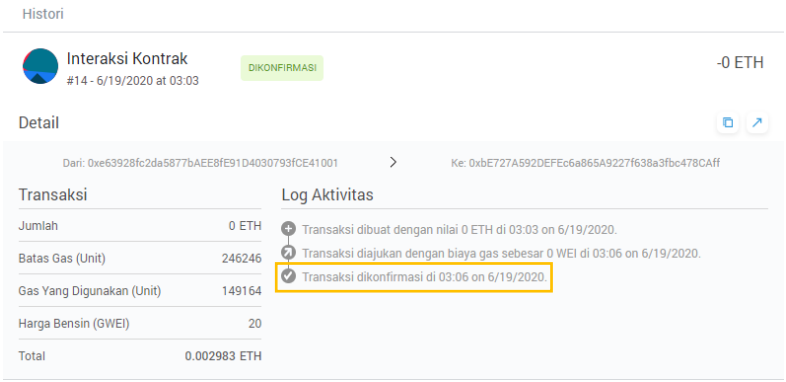
Fungsi	**Harapan**	**Hasil**
Memilih permintaan	Transaksi berhasil diverifikasi oleh *miner*	OK

5.3.1.3 *Uji Fungsionalitas Memilih Penyedia Layanan*

Uji coba ini dilakukan dengan pengguna mengakses halaman permintaan saya sebagai pemohon, mengklik tombol tampilkan penyedia layanan, memilih penyedia layanan yang sesuai, dan mengkonfirmasi transaksi pada Metamask seperti pada Gambar 5.6.



Gambar 5.5 Memilih penyedia layanan



Gambar 5.6 Log aktivasi memilih penyedia layanan

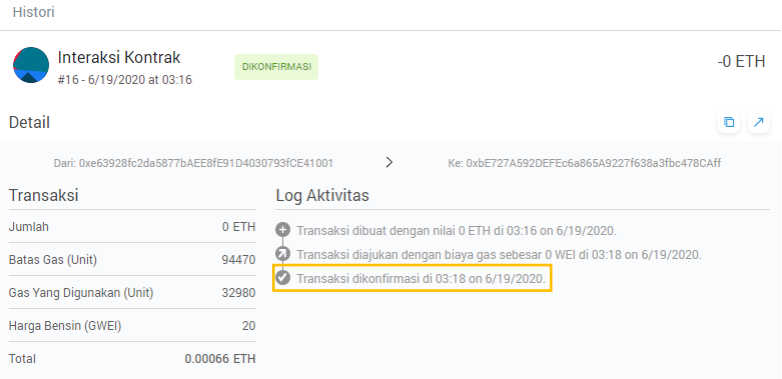
Tabel 5.4 Hasil uji fungsionalitas memilih penyedia layanan

Fungsi	**Harapan**	**Hasil**
Memilih penyedia layanan	Transaksi berhasil diverifikasi oleh *miner*	OK

5.3.1.4 *Uji Fungsionalitas Membatalkan Permintaan*

Uji coba ini dilakukan dengan pengguna mengakses halaman permintaan saya sebagai pemohon, mengklik tombol tampilkan penyedia layanan, mengklik tombol batalkan permintaan, dan mengkonfirmasi transaksi pada Metamask seperti pada Gambar 5.8.

Gambar 5.7 Membatalkan permintaan



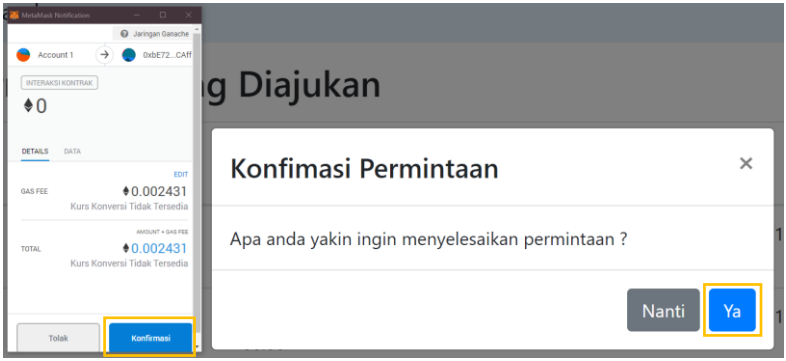
Gambar 5.8 Log aktivasi membatalkan permintaan

Tabel 5.5 Hasil uji fungsionalitas membatalkan permintaan

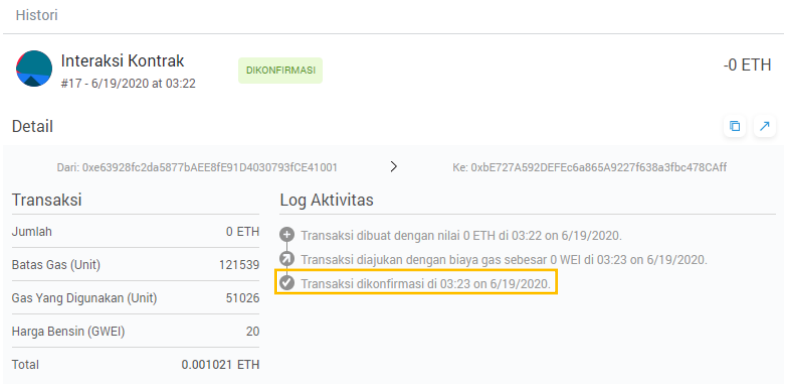
Fungsi	**Harapan**	**Hasil**
Membatalkan permintaan	Transaksi berhasil diverifikasi oleh *miner*	OK

5.3.1.5 *Uji Fungsionalitas Menyelesaikan Permintaan*

Uji coba ini dilakukan dengan pengguna mengakses halaman permintaan saya sebagai pemohon, mengklik tombol tandai selesai, dan mengkonfirmasi transaksi pada Metamask.



Gambar 5.9 Menyelesaikan permintaan



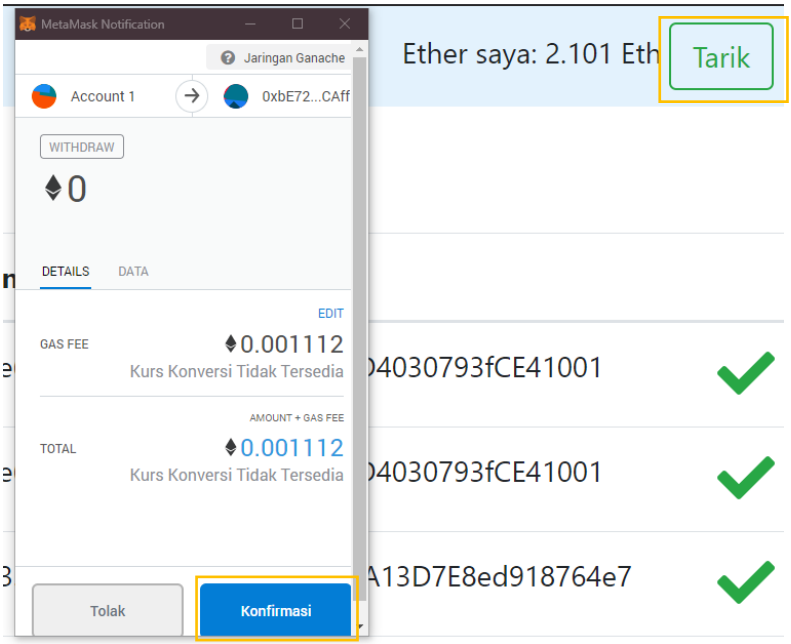
Gambar 5.10 Log aktivasi menyelesaikan permintaan

Tabel 5.6 Hasil uji fungsionalitas menyelesaikan permintaan

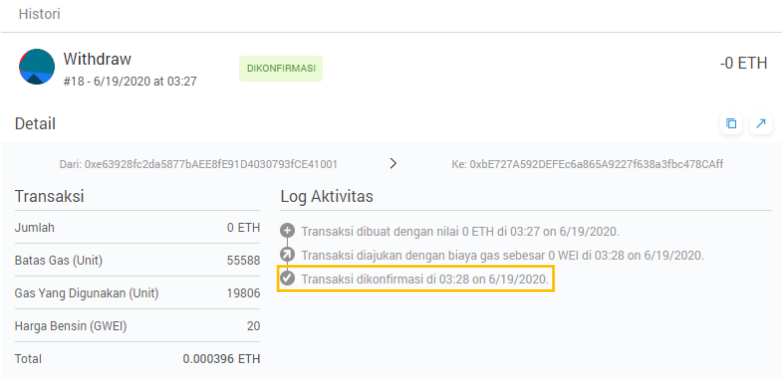
Fungsi	**Harapan**	**Hasil**
Menyelesaikan permintaan	Transaksi berhasil diverifikasi oleh *miner*	OK

5.3.1.6 *Uji Fungsionalitas Mencairkan Ether*

Uji coba ini dilakukan dengan pengguna mengklik tombol yang ditampilkan di kanan atas aplikasi, dan mengkonfirmasi transaksi pada Metamask.



Gambar 5.11 Mencairkan *ether*



Gambar 5.12 Log aktivitas mencairkan *ether*

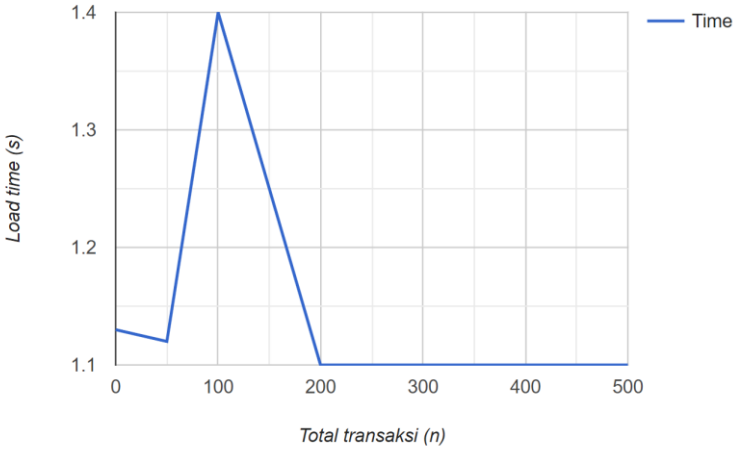
Tabel 5.7 Hasil uji fungsionalitas mencairkan *ether*

Fungsi	**Harapan**	**Hasil**
Mencairkan *ether*	Transaksi berhasil diverifikasi oleh *miner* dan menerima *ether* dari *smart contract*	OK

5.3.2 Uji Performa

Uji Performa dilakukan pada browser dan *virtual machine* yang digunakan. Pengujian menampilkan data dari *blockchain* pada browser dilakukan dengan menampilkan banyak data transaksi untuk mengetahui waktu dan memori yang digunakan untuk memproses data dari *blockchain*. Pengujian pada *Ethereum Virtual Machine* berfungsi untuk mengetahui berapa banyak memori dan CPU yang digunakan oleh jaringan *blockchain*. Pengujian pada *virtual machine* diambil saat kondisi mesin sedang *idle*, mengeksekusi transaksi, dan menampilkan data transaksi.

5.3.2.1 *Memuat Data Transaksi*



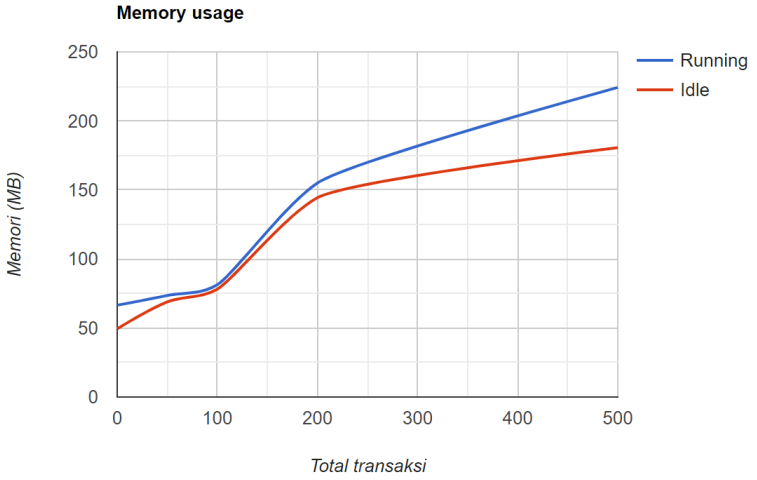
Gambar 5.13 Grafik waktu eksekusi browser

Tabel 5.8 Hasil uji performa waktu eksekusi

Total Transaksi	***Load Time* (s)**
0	1.130
50	1.119
100	1.135
200	1.103
500	1.1

Gambar 5.13 merupakan grafik hasil uji performa untuk waktu eksekusi pada pengambilan data transaksi saat memuat halaman web aplikasi. Dapat dilihat pada grafik tersebut penggunaan waktu untuk memuat data saat belum ada transaksi tidak memiliki banyak perbedaan dengan saat memuat 500 data transaksi. Perbandingan perbedaan waktu tidak dipengaruhi oleh

virtual machine yang digunakan, tetapi perangkat keras dan kondisi dari perangkat tersebut saat mengakses aplikasi. Data dari hasil uji performa untuk waktu eksekusi dapat dilihat pada Tabel 5.8.



Gambar 5.14 Grafik penggunaan memori browser

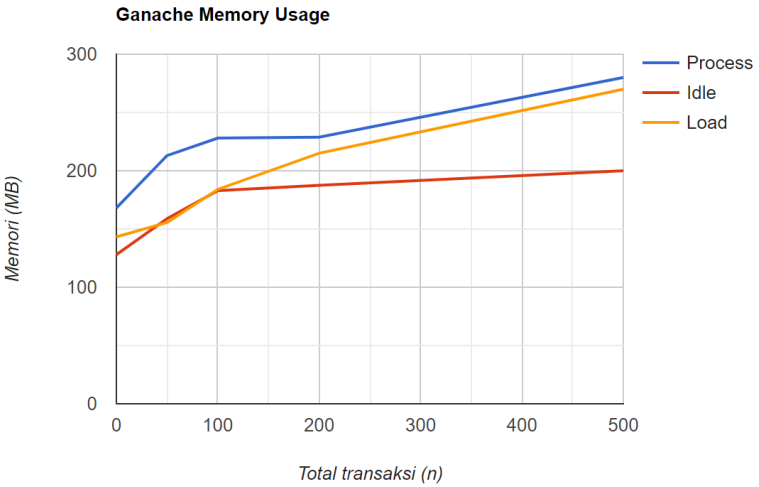
Tabel 5.9 Hasil uji performa penggunaan memori

Total Transaksi	**Penggunaan Memori saat *running* (MB)**	**Penggunaan Memori saat *idle* (MB)**
0	66.512	49.440
50	73.542	68.878
100	81.388	78.124
200	150.024	144.392
500	224.244	180.584

Gambar 5.24 merupakan grafik perbandingan penggunaan memori saat aplikasi sedang dalam kondisi sedang tidak digunakan

dan sedang aktif digunakan. Dapat dilihat pada grafik tersebut penggunaan memori akan meningkat sesuai dengan banyaknya data transaksi yang akan ditampilkan. Data hasil uji performa untuk penggunaan memori dapat dilihat pada Tabel 5.9.

5.3.2.2 *Virtual Machine*

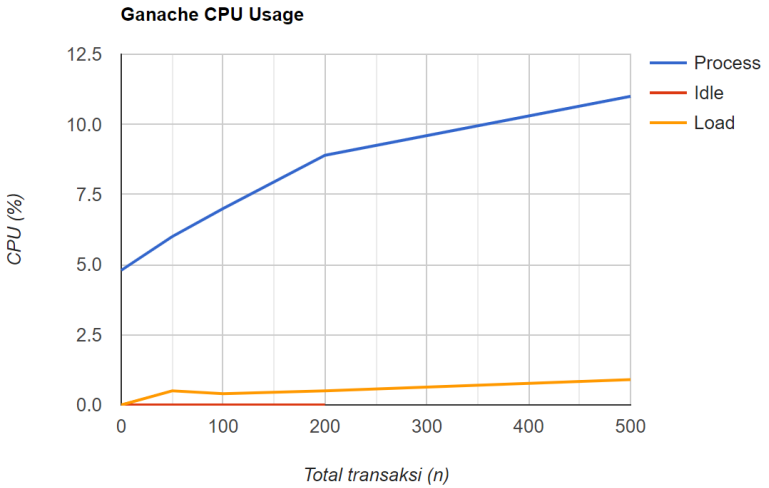


Gambar 5.15 Grafik penggunaan memori *ganache*

Tabel 5.10 Hasil uji performa penggunaan memori *ganache*

Ganache (n transaction)	***Memory (MB)***		
	Idle	***Execute Transaction***	***Load Transaction***
0	128	168	143.3
50	158.9	213	155.5
100	183	228	184
200	187	228.8	215
500	200	280	270

Gambar 5.15 merupakan grafik perbandingan penggunaan memori pada *virtual machine*. Pada gambar tersebut dapat dilihat penggunaan memori ketika *virtual machine* dalam kondisi memproses transaksi menggunakan memori yang lebih besar dibanding lainnya. Kondisi *virtual machine* saat kondisi *idle* cenderung lebih stabil, kondisi ini tidak memiliki perbedaan yang signifikan ketika jaringan *blockchain* memproses banyak data transaksi. Kondisi saat *load* atau menampilkan data, menggunakan memori yang hampir sama saat *virtual machine* memproses data transaksi. Data hasil uji performa pada *virtual machine* dapat dilihat pada Tabel 5.10.



Gambar 5.16 Grafik penggunaan CPU *ganache*

Tabel 5.11 *Hasil uji performa penggunaan CPU ganache*

Ganache (n transaction)	***CPU (%)***		
	Idle	***Execute Transaction***	***Load Transaction***
0	0	4.8	0
50	0	6	0.5
100	0	7	0.4
200	0	8.9	0.5
500	0	11	0.9

Gambar 5.16 merupakan grafik perbandingan pernggunaan CPU pada *virtual machine*. Pada gambar tersebut dapat dilihat penggunaan CPU ketika sedang memproses transaksi jauh lebih tinggi dibanding saat *load data* atau *idle*. *Virtual machine* tidak memakai CPU ketika sedang tidak digunakan, dan tidak menggunakan terlalu banyak CPU ketika *load data* dari

blockchain. Data hasil uji performa pada *virtual machine* dapat dilihat pada Tabel 5.11.

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan kesimpulan dan saran dari hasil uji coba yang telah dilakukan. Selain itu terdapat beberapa saran yang dapat dijadikan acuan untuk melakukan pengembangan dan penelitian lebih lanjut.

6.1 Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Penggunaan metode *crowdsourcing* dengan menggunakan *Ethereum* dapat menyelesaikan permasalahan dari modifikasi informasi, dan pengguna tidak perlu memberikan informasi sensitif ketika ingin berpartisipasi dalam menggunakan aplikasi ini.
2. Rekomendasi waktu yang diberikan kepada pengguna dapat membuatnya lebih mudah bagi penyedia layanan untuk menentukan waktu mulai tanpa harus memeriksa permintaan yang akan dilakukan dan menghitung waktu mulai. Ini dapat mempermudah penyedia layanan dalam menentukan waktu mulai untuk setiap layanan.
3. *Ethereum virtual machine* tidak mempengaruhi performa pada halaman antarmuka aplikasi. Jaringan *blockchain* hanya menyediakan informasi dan *metamask* berfungsi untuk melakukan *request* terhadap setiap data yang disimpan di aplikasi. Dengan *metamask* yang memfilter setiap request yang dilakukan pengguna, dapat melindungi jaringan *blockchain* dari kemungkinan spam *request*.

6.2 Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi, dan pengujian yang telah dilakukan. Di antaranya adalah sebagai berikut:

1. Menambahkan *tag* dan fungsi *search* agar mempermudah penyedia layanan menentukan permintaan yang ingin dilayani.
2. Menambahkan fungsi *filter* berdasarkan *tag*.

DAFTAR PUSTAKA

[1] X. Xu, Q. Liu, X. Zhang, J. Zhang, L. Qi and W. Dou, "A Blockchain-Powered Crowdsourcing Method With Privacy Preservation in Mobile Environment," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 6, pp. 1407-1419, 2019.

[2] Crowdsourcing Week, "What is Crowdsourcing?," Crowdsourcing Week, [Online]. Available: https://crowdsourcingweek.com/what-is-crowdsourcing/. [Accessed 25 May 2020].

[3] T. M. Fernández-Caramès and P. Fraga-Lamas, "Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks," *IEEE Access*, vol. 8, pp. 21091 - 21116, 14 November 2020.

[4] Wikipedia, "Ethereum," Wikipedia, 15 May 2020. [Online]. Available: https://en.wikipedia.org/wiki/Ethereum. [Accessed 25 May 2020].

[5] Ethereum, "Solidity Documentation," Ethereum, 24 April 2020. [Online]. Available: https://solidity.readthedocs.io/en/v0.6.8/. [Accessed 25 May 2020].

[6] A. Rosic, "Smart Contracts: The Blockchain Technology That Will Replace Lawyers," Blockgeeks, 2016. [Online]. Available: https://blockgeeks.com/guides/smart-contracts/. [Accessed 25 May 2020].

[7] Wikipedia, "JavaScript - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/JavaScript.

[8] Wikipedia, "React (web framework) - Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/React_(web_framework).

[9] Truffle Blockchain Group, Inc., "Truffle Overview," Truffle Blockchain Group, Inc., [Online]. Available: https://www.trufflesuite.com/docs/truffle/overview. [Accessed 25 May 2020].

[10] Truffle Blockchain Group, Inc., "Drizzle Overview," Truffle Blockchain Group, Inc., [Online]. Available: https://www.trufflesuite.com/docs/drizzle/overview. [Accessed 25 May 2020].

[11] Truffle Blockchain Group, Inc., "Ganache Overview," Truffle Blockchain Group, Inc., [Online]. Available: https://www.trufflesuite.com/docs/ganache/overview. [Accessed 25 May 2020].

[12] MetaMask, "MetaMask," [Online]. Available: https://metamask.io/. [Accessed 25 May 2020].

[13] npm, "geolib - npm," [Online]. Available: https://www.npmjs.com/package/geolib.

[*Halaman ini sengaja dikosongkan*]

BIODATA PENULIS

M. Hazdi Kurniawan, lahir di Jambi pada tanggal 20 November 1998. Penulis merupakan anak pertama dari tiga bersaudara. Penulis telah menempuh pendidikan formal di TK Al-Azhar Kota Jambi, SD Negeri 25 Kota Jambi (2004-2010), SMP Negeri 8 Kota Jambi (2010-2013), dan SMA Negeri 1 Kota Jambi (2013-2016). Penulis melanjutkan studi dengan berkuliah pada program sarjana (S1) di Departemen Informatika, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember. Selama menempuh perkuliahan penulis mendapatkan beasiswa dari Beasiswa Unggulan yang diadakan oleh Kementrian Pendidikan dan Kebudayaan.