

SISTEM VOTING ELEKTRONIK BERBASIS ETHEREUM *SMART CONTRACT*

TUGAS AKHIR

Diajukan guna memenuhi sebagian persyaratan dalam rangka menyelesaikan
Pendidikan Sarjana Strata Satu (S1) Program Studi Teknologi Informasi



**I DEWA GEDE DIRGA YASA
NIM: 1705551062**

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS UDAYANA
2022**

SISTEM VOTING ELEKTRONIK BERBASIS ETHEREUM *SMART CONTRACT*

TUGAS AKHIR

Diajukan guna memenuhi sebagian persyaratan dalam rangka menyelesaikan
Pendidikan Sarjana Strata Satu (S1) Program Studi Teknologi Informasi



**I DEWA GEDE DIRGA YASA
NIM: 1705551062**

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS UDAYANA
2022**

KATA PENGANTAR

Puji dan syukur saya panjatkan kehadapan Tuhan Yang Maha Esa sehingga penulis dapat menyelesaikan tugas akhir dengan judul “**Sistem Voting Elektronik Berbasis Ethereum Smart Contract**”. Dalam penyusunan Tugas Akhir ini, penulis mendapatkan masukan dan bimbingan dari berbagai pihak. Sehubungan dengan hal tersebut pada kesempatan ini penulis menyampaikan ucapan terima kasih kepada:

1. Ir. I Ketut Sudarsana, ST., Ph.D. selaku Dekan Fakultas Teknik Universitas Udayana.
2. Dr. Eng I Putu Agung Bayupati, S.T., M.T. selaku Koordinator Program Studi Teknologi Informasi Fakultas Teknik Universitas Udayana.
3. Bapak I Putu Arya Dharmadi, S.T.,M.T. selaku dosen pembimbing I yang telah banyak meluangkan waktu memberikan dorongan, bimbingan, tuntunan dan kesabaran selama penyusunan Tugas Akhir ini.
4. Bapak A. A. Kt. Agung Cahyawan Wiranatha, S.T.,M.T. selaku dosen pembimbing II yang juga telah banyak meluangkan waktu memberikan dorongan, bimbingan, tuntunan dan kesabaran selama penyusunan Tugas Akhir ini.
5. Segenap dosen pengajar yang turut memberikan ilmu dan pengetahuan selama penulis menempuh perkuliahan di Program Studi Teknologi Informasi Fakultas Teknik Universitas Udayana.
6. Bapak dan Ibu pegawai di Program Studi Teknologi Informasi Fakultas Teknik Universitas Udayana yang telah membantu selama proses perkuliahan dan administrasi.
7. Orang tua serta anggota keluarga yang telah banyak memberikan motivasi dan dukungan baik jasmani maupun rohani selama penulis menyusun Tugas Akhir ini.
8. Sarah Olivia Meily yang telah banyak memberikan motivasi dan dukungan kepada penulis untuk tetap semangat mengerjakan penelitian ini.

9. Teman-teman seperjuangan dan segenap civitas di Program Studi Teknologi Informasi Universitas Udayana yang telah memberikan sumbangan ide dan dukungan dalam penyusunan Tugas Akhir ini.
10. Berbagai pihak yang belum dapat disebutkan satu-persatu yang juga berperan penting dalam membantu, memberikan sumbangan ide, pemikiran dan dukungan dalam penyusunan Tugas Akhir ini

Penulis menyadari bahwa laporan ini jauh dari sempurna baik dalam materi maupun penulisannya. Berkaitan dengan hal tersebut, maka kritik dan saran yang bersifat membangun dari semua pihak sangat diharapkan. Akhir kata, semoga laporan ini dapat memberikan manfaat bagi semua pihak sesuai dengan yang diharapkan.

Denpasar, Mei 2022

I Dewa Gede Dirga Yasa

ABSTRAK

Pengambilan keputusan bersama menggunakan metode voting sudah banyak dilakukan, namun penggunaan sistem voting berbasis digital masih jarang ditemui. Hal ini dikarenakan sifat data digital yang mudah dimanipulasi tanpa meninggalkan jejak sehingga menimbulkan ketidakpercayaan masyarakat untuk menggunakan sistem digital sebagai media untuk mengambil keputusan yang bersifat sensitif. Penelitian ini bertujuan untuk mengembangkan sistem voting dengan integritas data yang terjaga sehingga dapat mengurangi ketidakpercayaan masyarakat tersebut. Pengembangan sistem voting ini menggunakan salah satu metode yang tergabung di dalam metode pengembangan Agile yaitu Scrum. Metode ini memampukan setiap pengerjaan fitur yang dibutuhkan oleh sistem memiliki siklus perencanaan, pengerjaan dan pengujianya masing-masing yang terstruktur dan terencana. Pengujian terhadap integritas data dilakukan dengan percobaan mengubah variabel yang tersimpan pada *database blockchain*. Berdasarkan pengujian tersebut, diperoleh hasil bahwa integritas data dapat terjaga dengan sangat baik, karena akses manipulasi *database* hanya diberikan kepada *node*. Selain itu fungsi pada *smart contract* untuk melakukan manipulasi data tidak lagi bisa digunakan, sehingga tidak ada cara lain yang dapat digunakan untuk melakukan manipulasi data, dengan kata lain *database* yang digunakan oleh *node* Ethereum sudah terkunci. Hal ini tentunya dapat membuktikan bahwa *blockchain* yang mendukung *smart contract* seperti Ethereum mampu menyimpan data dengan keamanan dan integritas yang sangat baik.

Kata kunci : Voting, *Blockchain*, Ethereum, *Smart Contract*

ABSTRACT

Joint decision making using the voting method has been widely carried out, but the use of digital-based voting systems is still rarely encountered. This is due to the nature of digital data that is easy to manipulate without leaving a trace, causing public distrust to use digital systems as a medium for making sensitive decisions. This study aims to develop a voting system with maintained data integrity to reduce public distrust. The development of this voting system uses one of the methods incorporated in the Agile development method, namely Scrum. This method enables each feature work required by the system to have its own structured and planned planning, execution and testing cycle. Testing the integrity of the data is done by trying to change the variables stored in the blockchain database. Based on these tests, the results obtained that data integrity can be maintained very well, because access to database manipulation is only given to nodes. In addition, the function of the smart contract to manipulate data can no longer be used, so there is no other way that can be used to manipulate data, in other words the database used by the Ethereum node is locked. This certainly proves that a blockchain that supports smart contracts like Ethereum can store data with excellent security and integrity.

Keywords : Voting, Blockchain, Ethereum, Smart Contract

DAFTAR ISI

HALAMAN SAMPUL.....	i
HALAMAN JUDUL	ii
KATA PENGANTAR.....	iii
ABSTRAK	v
DAFTAR ISI.....	vii
DAFTAR GAMBAR.....	xi
DAFTAR KODE PROGRAM	xv
DAFTAR TABEL.....	xvi
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan.....	3
1.4 Manfaat Penelitian.....	3
1.5 Batasan Masalah.....	3
1.6 Sistematika Penulisan.....	4
BAB II TINJAUAN PUSTAKA.....	5
2.1 <i>State of the Art</i>	5
2.2 Blockchain.....	8
2.3 Ethereum	9
2.3.1 Ethereum Accounts dan Transaction.....	9
2.3.2 Ethereum Code Execution	10
2.3.3 Openethereum.....	10
2.3.4 Smart Contract.....	11
2.3.5 Konsensus.....	12
2.3.6 Gas, Gas Fee dan Gas Limit	12
2.3.7 Protokol Komunikasi.....	13
2.4 VSCode	14
2.5 NodeJS	15
2.5.1 NestJS	16
2.5.2 Angular	17

2.5.3	Web3JS.....	19
2.6	MySQL.....	20
2.7	JSON	21
2.8	Google Cloud Platform	22
2.8.1	Compute Engine	22
2.9	Agile Software Development Method.....	23
2.9.1	Backlog.....	24
2.9.2	Sprints.....	24
2.9.3	Scrum Meeting	24
2.9.4	Demos.....	24
BAB III METODOLOGI DAN PERANCANGAN SISTEM	26	
3.1	Tempat dan Waktu Penelitian	26
3.2	Metode Penelitian.....	26
3.2.1	Penerapan Tahap Backlog	26
3.2.2	Penerapan Tahap Sprints	27
3.2.3	Penerapan Tahap Scrum Meeting.....	27
3.2.4	Penerapan Tahap Demos	27
3.3	Alat Penelitian Aplikasi	27
3.3.1	Kebutuhan Perangkat Lunak Implementasi Aplikasi	28
3.3.2	Kebutuhan Perangkat Keras Implementasi Aplikasi	28
3.4	Gambaran Umum Sistem	28
3.4.1	Use Case Diagram	30
3.4.2	Rancangan Alur Sistem	32
3.5	Rancangan Basis Data.....	46
3.5.1	Implementasi Basis Data	46
3.5.2	Struktur Data Tabel	48
3.6	Bahasa Pemrograman.....	51
3.7	Rancangan Antarmuka Aplikasi.....	51
3.8	Rancangan Spesifikasi API	57
BAB IV PENGUJIAN DAN ANALISA HASIL.....	65	
4.1	Infrastruktur Google Cloud	65
4.1.1	Persiapan Infrastruktur	65
4.1.2	Persiapan Virtual Machine	71
4.2	Konfigurasi Openethereum	75

4.2.1	Genesis Block	75
4.2.2	Konfigurasi Node Openethereum.....	77
4.2.3	Menjalankan Jaringan.....	79
4.3	Implementasi Desain Antarmuka	81
4.3.1	Membuat Pemilihan.....	81
4.3.2	Menambahkan Kandidat.....	83
4.3.3	Deploy Pemilihan	84
4.3.4	Mengikuti Pemilihan	87
4.3.5	Daftar Peserta Pemilihan	88
4.3.6	Memulai Pemilihan.....	89
4.3.7	Vote	91
4.3.8	Menghentikan Pemilihan.....	92
4.3.9	Melihat Hasil Pemilihan	93
4.4	Implementasi Desain Antarmuka Mobile	94
4.4.1	Login.....	94
4.4.2	Available Election	95
4.4.3	Followed Election.....	96
4.4.4	Detail Pemilihan	97
4.4.5	Ended Election.....	99
4.4.6	Hasil.....	100
4.5	Implementasi Rancangan API.....	101
4.5.1	Membuat Pemilihan.....	101
4.5.2	Menambahkan Kandidat.....	103
4.5.3	Deploy Pemilihan	104
4.5.4	Mengikuti Pemilihan	105
4.5.5	Daftar Peserta Pemilihan	106
4.5.6	Detail Pemilihan	107
4.5.7	Memulai Pemilihan.....	108
4.5.8	Vote	109
4.5.9	Menghentikan Pemilihan.....	110
4.5.10	Melihat Hasil Pemilihan	110
4.6	Pengujian Sistem Voting Elektronik	112
4.6.1	Pengujian API Aplikasi	112
4.6.2	Pengujian Integritas Data Blockchain	118

BAB V PENUTUP	132
5.1 Kesimpulan.....	132
5.2 Saran.....	132
DAFTAR PUSTAKA	134

DAFTAR GAMBAR

Gambar 2. 1 Arsitektur <i>Blockchain</i>	8
Gambar 2. 2 Diagram Komunikasi Node Ethereum	14
Gambar 2. 3 Visual Studio Code.....	15
Gambar 2. 4 Node.js.....	16
Gambar 2. 5 NestJS.....	17
Gambar 2. 6 Angular.....	18
Gambar 2. 7 Interaksi Web3.JS dengan Node Ethereum.....	19
Gambar 2. 8 <i>Agile Software Development Method</i>	23
Gambar 3. 1 Gambaran Umum Sistem	29
Gambar 3. 2 <i>Use Case Diagram</i>	31
Gambar 3. 3 Rancangan Alur Sistem (<i>Super Admin</i>)	33
Gambar 3. 4 Rancangan Alur Sistem (<i>Election Authority</i>)	34
Gambar 3. 5 Rancangan Alur Sistem (<i>Voter</i>)	36
Gambar 3. 6 Rancangan Alur Proses <i>Login</i>	38
Gambar 3. 7 Rancangan Alur Penambahan <i>Election Authority</i>	39
Gambar 3. 8 Rancangan Alur Pembuatan <i>Election</i> Baru.....	40
Gambar 3. 9 Rancangan Alur Penambahan Kandidat.....	41
Gambar 3. 10 Rancangan Alur <i>Deploy Election</i>	42
Gambar 3. 11 Rancangan Alur Registrasi <i>Voter</i>	43
Gambar 3. 12 Rancangan Alur Pendaftaran <i>Voter</i> dalam <i>Election</i>	44
Gambar 3. 13 Rancangan Alur Proses <i>Vote</i>	45
Gambar 3. 14 Implementasi Basis Data.....	47
Gambar 3. 15 Rancangan Antarmuka Buat Pemilihan	52
Gambar 3. 16 Rancangan Antarmuka Menambahkan Kandidat.....	53
Gambar 3. 17 Rancangan Antarmuka Daftar Pemilihan Siap <i>Deploy</i>	54
Gambar 3. 18 Rancangan Antarmuka Daftar Peserta Pemilihan	55
Gambar 3. 19 Rancangan Antarmuka Detail Pemilihan	56
Gambar 3. 20 Rancangan Antarmuka Hasil Pemilihan	57

Gambar 4. 1 Halaman Google Cloud untuk Mengelola <i>Firewall Rule</i>	66
Gambar 4. 2 <i>Firewall Rule</i> yang Diperlukan	67
Gambar 4. 3 Halaman Google Cloud untuk Mengelola <i>VM</i>	68
Gambar 4. 4 Spesifikasi <i>Virtual Machine</i>	69
Gambar 4. 5 Konfigurasi Jaringan <i>Virtual Machine</i>	70
Gambar 4. 6 Konfigurasi Security <i>Virtual Machine</i>	70
Gambar 4. 7 Halaman Compute Engine ketika berhasil membuat <i>VM</i>	71
Gambar 4. 8 <i>SSH</i> ke <i>Virtual Machine</i>	72
Gambar 4. 9 Proses Instalasi <i>Package</i>	72
Gambar 4. 10 Proses <i>Download Binary Openethereum</i>	73
Gambar 4. 11 Proses <i>Extract File</i>	73
Gambar 4. 12 Proses Ubah <i>Permission File</i>	74
Gambar 4. 13 Proses <i>Download Konfigurasi Node Ethereum</i>	74
Gambar 4. 14 Struktur <i>File Konfigurasi</i>	75
Gambar 4. 15 <i>Node</i> Terhubung	81
Gambar 4. 16 Tampilan Halaman Pembuatan Pemilihan	82
Gambar 4. 17 Tampilan Pesan Sukses	82
Gambar 4. 18 Tampilan Halaman Penambahan Kandidat	83
Gambar 4. 19 Tampilan Pesan Sukses	84
Gambar 4. 20 Tampilan <i>Detail Pemilihan</i>	85
Gambar 4. 21 Tampilan Konfirmasi	85
Gambar 4. 22 Tampilan Pesan Sukses	86
Gambar 4. 23 Tampilan Daftar Pemilihan yang Siap Deploy	86
Gambar 4. 24 Tampilan Pesan Sukses	87
Gambar 4. 25 Tampilan Daftar Pemilihan	87
Gambar 4. 26 Tampilan Pesan Sukses	88
Gambar 4. 27 Tampilan Daftar Peserta Pemilihan.....	89
Gambar 4. 28 Tampilan Konfirmasi Mulai Pemilihan.....	90
Gambar 4. 29 Tampilan Pesan Sukses	90
Gambar 4. 30 Tampilan <i>Vote</i> Sebuah Pemilihan	91

Gambar 4. 31 Tampilan Konfirmasi Memilih Kandidat.....	92
Gambar 4. 32 Tampilan Konfirmasi Menghentikan Pemilihan	92
Gambar 4. 33 Tampilan Pesan Sukses	93
Gambar 4. 34 Tampilan Hasil Pemilihan.....	94
Gambar 4. 35 Tampilan <i>Login Mobile</i>	95
Gambar 4. 36 Tampilan <i>Available Election</i>	96
Gambar 4. 37 Tampilan Followed Election	97
Gambar 4. 38 Tampilan Detail Pemilihan.....	98
Gambar 4. 39 Tampilan Vote	99
Gambar 4. 40 Tampilan Ended Election	100
Gambar 4. 41 Tampilan Hasil Pemilihan	101
Gambar 4. 42 <i>HTTP Request</i> untuk Membuat Pemilihan.....	102
Gambar 4. 43 <i>HTTP Response</i> Pembuatan Pemilihan	102
Gambar 4. 44 <i>HTTP Request</i> untuk Menambahkan Kandidat.....	103
Gambar 4. 45 <i>HTTP Response</i> Penambahan Kandidat.....	103
Gambar 4. 46 <i>HTTP Request Deploy</i> Pemilihan.....	104
Gambar 4. 47 <i>HTTP Response Deploy</i> Pemilihan	104
Gambar 4. 48 <i>HTTP Request</i> untuk Mengikuti Pemilihan	105
Gambar 4. 49 <i>HTTP Response</i> Mengikuti Pemilihan	105
Gambar 4. 50 <i>HTTP Request</i> Daftar Peserta Pemilihan	106
Gambar 4. 51 <i>HTTP Response</i> Daftar Peserta Pemilihan	106
Gambar 4. 52 <i>HTTP Request</i> Detail Pemilihan	107
Gambar 4. 53 <i>HTTP Response</i> Detail Pemilihan	107
Gambar 4. 54 <i>HTTP Request</i> Memulai Pemilihan.....	108
Gambar 4. 55 <i>HTTP Response</i> Memulai Pemilihan	108
Gambar 4. 56 <i>HTTP Request</i> Pemilihan Kandidat.....	109
Gambar 4. 57 <i>HTTP Response</i> Pemilihan Kandidat	109
Gambar 4. 58 <i>HTTP Request</i> Menghentikan Pemilihan	110
Gambar 4. 59 <i>HTTP Response</i> Menghentikan Pemilihan.....	110
Gambar 4. 60 <i>HTTP Request</i> Hasil Pemilihan.....	111
Gambar 4. 61 <i>HTTP Response</i> Hasil Pemilihan	111

Gambar 4. 62 Struktur <i>File Pengujian</i>	112
Gambar 4. 63 Hasil Pengujian <i>Endpoint Super Admin</i>	114
Gambar 4. 64 Hasil Pengujian <i>Endpoint Election Authority</i>	116
Gambar 4. 65 Hasil Pengujian <i>Endpoint Voter</i>	118
Gambar 4. 66 Tampilan.....	120
Gambar 4. 67 <i>Rlpdump Help</i>	121
Gambar 4. 68 <i>Ldb Help</i>	122
Gambar 4. 69 Tampilan Awal	122
Gambar 4. 70 <i>Deployed Smart Contract</i>	123
Gambar 4. 71 <i>Connected Smart Contract</i>	123
Gambar 4. 72 Tombol Get Person.....	124
Gambar 4. 73 Tombol Set Person	124
Gambar 4. 74 Tombol Get Person.....	125
Gambar 4. 75 Tombol Set Person	125
Gambar 4. 76 <i>Block Explorer</i>	126
Gambar 4. 77 Detail <i>Block 2</i>	127
Gambar 4. 78 Ethereum <i>Decoder</i>	128
Gambar 4. 79 <i>Param Generator</i>	128
Gambar 4. 80 Ethereum <i>Decoder</i>	129
Gambar 4. 81 <i>RocksDB Administrative Tools</i>	129
Gambar 4. 82 Struktur Data <i>RLP</i>	130
Gambar 4. 83 <i>Encoding RLP</i>	130
Gambar 4. 84 <i>Update Database RocksDB</i>	131

DAFTAR KODE PROGRAM

Kode Program 3. 1 Spesifikasi <i>Request</i> Pembuatan Pemilihan	57
Kode Program 3. 2 Spesifikasi Response Pembuatan Pemilihan	58
Kode Program 3. 3 Spesifikasi Request Penambahan Kandidat.....	58
Kode Program 3. 4 Spesifikasi <i>Response</i> Penambahan Kandidat.....	59
Kode Program 3. 5 Spesifikasi <i>Request Deployment</i> Pemilihan.....	59
Kode Program 3. 6 Spesifikasi <i>Response Deployment</i> Pemilihan	59
Kode Program 3. 7 Spesifikasi <i>Request</i> Peserta Pemilihan	59
Kode Program 3. 8 Spesifikasi <i>Resposne</i> Peserta Pemilihan.....	60
Kode Program 3. 9 Spesifikasi <i>Request Detail</i> Pemilihan.....	60
Kode Program 3. 10 Spesifikasi <i>Response Detail</i> Pemilihan	61
Kode Program 3. 11 Spesifikasi <i>Request</i> Hasil Pemilihan	61
Kode Program 3. 12 Spesifikasi <i>Response</i> Hasil Pemilihan	62
Kode Program 3. 13 Spesifikasi <i>Request</i> Memulai Pemilihan	62
Kode Program 3. 14 Spesifikasi <i>Response</i> Memulai Pemilihan	62
Kode Program 3. 15 Spesifikasi <i>Request Mengikuti Pemilihan</i>	62
Kode Program 3. 16 Spesifikasi <i>Response Mengikuti Pemilihan</i>	63
Kode Program 3. 17 Spesifikasi <i>Request</i> untuk Memilih Kandidat.....	63
Kode Program 3. 18 Spesifikasi <i>Response</i> untuk Memilih Kandidat	63
Kode Program 3. 19 Spesifikasi <i>Request</i> untuk Menghentikan Pemilihan.....	63
Kode Program 3. 20 Spesifikasi <i>Response</i> untuk Menghentikan Pemilihan	64
Kode Program 4. 1 <i>File</i> simvoni-spec.json	77
Kode Program 4. 2 <i>File</i> simvoni-config.toml	78
Kode Program 4. 3 Perintah Menjalankan Openethereum.....	79
Kode Program 4. 4 CURL untuk Membuat <i>Validator Account</i>	79
Kode Program 4. 5 CURL untuk Membuat <i>Validator Account</i>	80
Kode Program 4. 6 Menyimpan <i>Password Validator</i>	80
Kode Program 4. 7 Mendapatkan Enode	80
Kode Program 4. 8 Menambahkan Enode	80
Kode Program 4. 9 <i>Smart Contract</i>	120

DAFTAR TABEL

Tabel 2. 1 MySQL <i>Function</i>	20
Tabel 3. 1 Kebutuhan Perangkat Keras Implementasi Aplikasi	28
Tabel 3. 2 Struktur Data Tabel ref_user.....	48
Tabel 3. 3 Struktur Data Tabel ref_user_role.....	48
Tabel 3. 4 Struktur Data Tabel ta_candidate.....	49
Tabel 3. 5 Struktur Data Tabel ta_misi	49
Tabel 3. 6 Struktur Data Tabel ta_pengalaman.....	50
Tabel 3. 7 Struktur Data Tabel ta_election	50
Tabel 3. 8 Struktur Data Tabel ref_election_status.....	51
Tabel 4. 1 Skenario Pengujian <i>Endpoint Super Admin</i>	113
Tabel 4. 2 Skenario Pengujian <i>Endpoint Election Authority</i>	114
Tabel 4. 3 Skenario Pengujian <i>Endpoint Voter</i>	116

BAB I

PENDAHULUAN

Bab I berisi bahasan mengenai pendahuluan dari penelitian ini. Pendahuluan dijabarkan menjadi beberapa bagian, diantaranya latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah, dan sistematika penulisan.

1.1 Latar Belakang

Pengambilan keputusan bersama memiliki beberapa cara dalam pelaksanaannya, diantaranya musyawarah mufakat, voting dan aklamasi. Masing-masing metode pengambilan keputusan memiliki kelebihan dan kekurangan. Metode pengambilan keputusan yang paling sering digunakan adalah voting karena keputusan dapat dihasilkan dalam jangka waktu yang relatif cepat. Kegiatan yang paling sering menggunakan metode voting adalah pemilihan kepala daerah karena voting sesuai dengan prinsip demokrasi yang memberikan kesempatan setiap kalangan untuk memberikan hak suara. Seiring dengan berkembangnya teknologi, kecepatan dalam mengakses informasi adalah prioritas sehingga cara voting konvensional yang menggunakan kertas dan paku bukan menjadi pilihan yang optimal.

Berdasarkan permasalahan di atas, negara-negara di dunia mulai mengembangkan voting elektronik atau yang biasa disebut dengan *e-voting*. *E-voting* memiliki beberapa model yang digunakan yaitu *internet poll site voting*, *kiosk voting* dan *internet voting*. *Internet poll site voting* merupakan model *e-voting* yang menggunakan komputer di lokasi pemilihan dan menggunakan internet untuk mengirim data dari lokasi pemilihan ke otoritas penyelenggara pemilihan. Sehingga pemilih masih harus datang ke lokasi pemilihan untuk melakukan voting. *Kiosk voting* adalah model *e-voting* yang pemilihnya bisa menggunakan komputer di lokasi yang sudah menjalankan kerja sama dengan otoritas penyelenggara pemilihan misalnya komputer di perpustakaan daerah, sekolah dan *mall*. Sedangkan

internet voting adalah model *e-voting* yang sepenuhnya menggunakan internet sehingga pemilih dapat melakukan voting dari perangkat masing-masing (Muhammad Habibi, 2018). Model *e-voting* *internet poll site voting* dan *kiosk voting* dirasa kurang optimal karena walaupun sudah menggunakan teknologi internet, pemilih masih belum bisa menggunakan perangkat yang dimiliki sehingga tetap harus datang ke tempat pemilihan untuk melakukan voting. Saat ini, era dimana kecepatan dan kemudahan akses informasi menjadi prioritas, internet voting menjadi model *e-voting* yang paling optimal untuk dikembangkan.

Beberapa negara yang sudah mengembangkan *e-voting* model internet voting diantaranya Australia, Kanada, Jepang, Jerman dan Inggris (Muhammad Habibi, 2018). *E-voting* berjalan dengan lancar di Australia, Kanada dan Jepang namun gagal di Jerman dan Inggris. *E-voting* juga memiliki beberapa masalah yang membuat *e-voting* mendapatkan penolakan oleh masyarakat yaitu masalah kerahasiaan dan kepercayaan terhadap data. Salah satu teknologi yang dapat mengatasi masalah tersebut adalah Ethereum *Smart Contract*.

Ethereum *Smart Contract* adalah program komputer yang berjalan di dalam jaringan *blockchain* Ethereum. Program komputer tersebut melakukan transaksi dalam jaringan *blockchain* dan program komputer tersebut tidak dapat dikontrol oleh seorang pengguna. Jadi *smart contract* hanya akan berjalan sesuai dengan kode program yang disimpan. Karena sifatnya yang berjalan secara *autonomous* maka dapat menjadi solusi terhadap masalah kepercayaan penyimpanan data sensitif.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan sebelumnya, maka diperoleh beberapa rumusan masalah sebagai berikut.

- a. Bagaimana perancangan serta pembangunan Sistem Voting Elektronik berbasis Ethereum *smart contract*?
- b. Bagaimana kinerja keamanan Ethereum dalam Sistem Voting Elektronik berbasis Ethereum *smart contract*?

1.3 Tujuan

Berdasarkan rumusan masalah yang telah dipaparkan sebelumnya, maka diperoleh tujuan sebagai berikut.

- a. Mengetahui perancangan serta pembangunan Sistem Voting Elektronik berbasis Ethereum *smart contract*.
- b. Mengetahui hasil penelitian dan kinerja Sistem Voting Elektronik berbasis Ethereum *smart contract*.

1.4 Manfaat Penelitian

Penelitian ini dalam pelaksanaannya memiliki beberapa manfaat yang dijabarkan sebagai berikut.

- a. Menghilangkan *trust issue* dalam pelaksanaan voting.
- b. Meningkatkan keamanan data hasil voting.
- c. Menghemat waktu dalam proses pengambilan data voting.
- d. Meminimalisir biaya yang perlu dikeluarkan dalam proses pengambilan data voting.
- e. Mempermudah proses pengambilan data voting karena dapat dilakukan dari mana saja melalui perangkat pengguna yang terkoneksi internet.

1.5 Batasan Masalah

Penelitian ini dalam pelaksanaannya memiliki beberapa batasan yang bertujuan agar penggunaan aplikasi tidak terlalu luas. Batasan masalah yang dijabarkan adalah sebagai berikut.

- a. Penelitian yang dilakukan hanya berfokus pada Sistem Voting Elektronik.
- b. *Platform blockchain* yang digunakan adalah Ethereum.
- c. Sistem Voting Elektronik hanya dapat digunakan untuk melakukan proses voting.
- d. Keamanan yang diberikan oleh blockchain tidak bisa mencegah serangan yang disebabkan oleh *Social Engineering*.

1.6 Sistematika Penulisan

Sistematika penulisan merupakan bagian yang menjelaskan gambaran laporan dari semua dasar teori dan metode yang digunakan serta hasil yang diperoleh selama penggerjaan tugas akhir. Laporan tugas akhir ini dibagi menjadi lima bab sebagai berikut.

BAB I : **Pendahuluan**

Bagian ini membahas mengenai latar belakang, perumusan masalah, tujuan, manfaat, batasan masalah, serta sistematika penulisan laporan tugas akhir sistem *e-voting* berbasis Ethereum *smart contract*.

BAB II : **Tinjauan Pustaka**

Bagian ini menjelaskan tentang dasar teori, pengetahuan, dan referensi yang digunakan untuk memperkuat projek tugas akhir sistem *e-voting* berbasis Ethereum *smart contract*.

BAB III : **Metodologi dan Perancangan Sistem**

Bagian ini memuat metode yang digunakan dalam penelitian tugas akhir sistem *e-voting* berbasis Ethereum *smart contract* beserta dengan proses pembuatan laporan yang meliputi tempat dan waktu penelitian, sumber data dan metode pengumpulan, instrumen perancangan, algoritma pemrograman, hingga perancangan sistem dimulai dari rancangan prosedural hingga antarmuka (*interface*) sistem.

BAB IV : **Pengujian dan Analisa Hasil**

Bagian ini memuat rancangan yang telah dibuat, implementasi dari perangkat yang digunakan serta pengujian sistem dari sistem *e-voting* berbasis Ethereum *smart contract*.

BAB V : **Penutup**

Bagian ini memuat kesimpulan yang mengacu pada rumusan masalah dan tujuan penelitian beserta saran yang diberikan peneliti untuk kepentingan pengembangan aplikasi selanjutnya.

BAB II

TINJAUAN PUSTAKA

Bab II berisi bahasan mengenai teori atau materi pendukung yang digunakan sebagai acuan atau dasar teori dalam sistem *e-voting* berbasis Ethereum *smart contract*.

2.1 *State of the Art*

Perancangan untuk pengembangan Sistem Voting Elektronik memiliki persamaan dan atau keterikatan dari beberapa penelitian yang dilakukan beberapa diantaranya adalah sebagai berikut.

Penelitian pertama yang digunakan sebagai acuan dalam penelitian ini adalah penelitian mengenai pengembangan sistem pemungutan suara elektronik yang digunakan sebagai layanan pemilihan nasional politik dengan studi kasus di Pakistan. Secara umum, penelitian ini mengevaluasi kemampuan *distributed ledger technologies* dengan menggambarkan penyelidikan kontekstual untuk menyempurnakan proses keputusan pemilihan politik dan menggunakan aplikasi berbasis *blockchain* yang meningkatkan keamanan dan menurunkan biaya pelaksanaan pemilihan nasional. Sistem yang dikembangkan berbasis *blockchain* menggunakan *smart contract*. Penelitian ini memperoleh hasil bahwa sulit untuk menerapkan sistem pemungutan suara elektronik di Pakistan. Banyak kendala hukum dan teknis yang dapat mencegah *blockchain* digunakan sebagai sistem pemungutan suara elektronik. Pemungutan suara elektronik akan berdampak pada skalabilitas sistem berbasis *blockchain*. Semakin banyak *node* dalam jaringan *blockchain* memperburuk masalah skalabilitas (Hassan, et al., 2022).

Penelitian selanjutnya adalah penelitian mengenai sistem E-Voting yang juga menggunakan teknologi *blockchain*. Fitur utama dari sistem ini adalah memastikan integritas dan transparansi data serta memastikan satu suara per nomor ponsel untuk setiap suara dengan privasi yang terjamin. *Ethereum Virtual Machine*

(*EVM*) digunakan sebagai *runtime environment blockchain*, dimana *smart contract* yang transparan dan konsisten, akan digunakan oleh penyelenggara untuk setiap acara pemungutan suara untuk menjalankan aturan pemungutan suara. Pengguna diautentikasi melalui nomor ponsel mereka tanpa memerlukan *server* pihak ketiga. Hasil penelitian ini menunjukkan bahwa sistem ini layak dan dapat menawarkan langkah menuju sistem pemilihan yang lebih terdigitalisasi (Khoury, et al., 2018).

Sistem voting elektronik menggunakan *blockchain* selanjutnya adalah sistem dimana setiap pemilih diidentifikasi secara unik dengan nomor identitas bernama Aadhar yang disetujui Pemerintah India. Aplikasi menggunakan nomor ini untuk memastikan bahwa setiap pemilih hanya mendapat satu kesempatan untuk memilih. Ketika suara dikirimkan sebagai transaksi maka semua *peer* disinkronkan. Karena setiap *peer* dikaitkan dengan *public* dan *private key*, suara dienkripsi lalu di-*hash* dan ditambahkan ke *blockchain* untuk meningkatkan keamanan dan membentuk *chain of block*. Suara tidak dapat dilacak kembali ke pemilih. Dalam penelitian ini, jaringan *peer to peer* dibuat memiliki minimal tiga *peer*. Skalabilitas aplikasi *blockchain* tergantung pada batas memori sekunder dari *peer*. Aplikasi ini membahas faktor keamanan seperti verifikasi, transparansi penghitungan suara, integritas dan non-penolakan suara, namun tidak membahas otentikasi pemilih dengan mekanisme yang kuat seperti atribut biometrik. Ethereum dan *smart contract* adalah pencapaian progresif sejak *blockchain* itu sendiri, menghalangi kesan terbatas *blockchain* sebagai mata uang digital (koin), dan mengubahnya menjadi jawaban untuk beberapa masalah terkait internet di dunia saat ini, dan dapat memberdayakan pemanfaatan luas dari *blockchain* (Shukla, et al., 2020).

Penelitian selanjutnya menyajikan upaya untuk memanfaatkan *blockchain* seperti fondasi kriptografi dan transparansi untuk mencapai skema *e-voting* yang efektif. Penelitian ini menyajikan rincian skema *e-voting* yang diusulkan beserta implementasinya menggunakan *platform Multichain*. Fokus dari penelitian ini adalah peningkatan ketahanan teknologi *blockchain* terhadap masalah pengeluaran hasil ganda yang akan diterjemahkan sebagai 'pemilihan ganda' untuk sistem *e-voting*. Pengujian dilakukan dari beberapa langkah yaitu melakukan banyak transaksi, verifikasi transaksi, *mining* transaksi ke dalam *blockchain*, sinkronisasi

perubahan yang dibuat dalam *public ledger* ke semua *node* dalam jaringan dan kegunaan sistem. Uji coba dilakukan langsung di *Multichain* dengan memulai dari pembuatan aset/suara. Untuk melakukan transaksi di *Multichain*, dilakukan identifikasi alamat dan nilai di alamat *node Multichain* dari mana aset (suara) akan dikirim (Khan, et al., 2018).

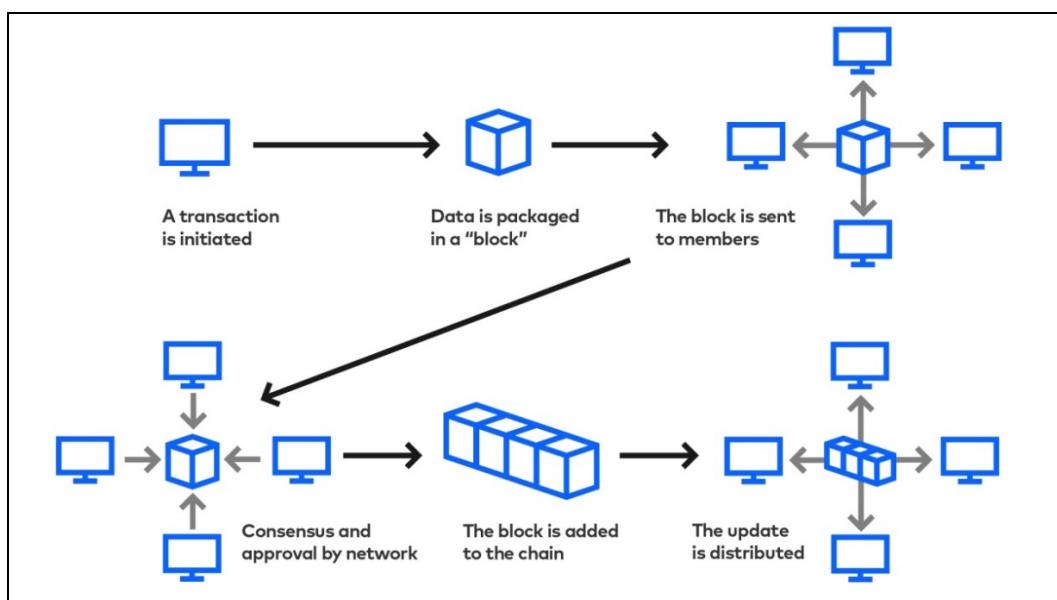
Perancangan sistem voting elektronik menggunakan *smart contract blockchain* juga diterapkan dalam penelitian ini. Sistem dirancang menggunakan bahasa pemrograman *solidity* dan *smart contract* menghasilkan kode unik pada setiap pemilihan baru. Pengujian menghasilkan bahwa manipulasi hasil pemungutan suara tidak dapat dilakukan karena setiap pemilih hanya dapat memiliki satu akun dan satu *address blockchain* (Setia & Susanto, 2019). Perancangan sistem *e-voting* juga diterapkan menggunakan metode enkripsi *blockchain*. Enkripsi *blockchain* merupakan salah satu cara pengamanan data yang tidak memungkinkan untuk dilakukan penyusupan data ke dalamnya serta menjaga kerahasiaan pada data pemilih karena dilakukan proses enkripsi pada data pemilih dan apa yang dipilih, akan tetap dapat diperoleh informasi pemilik suara terbanyak (Ardilla, 2018). Penelitian lain terhadap sistem *e-voting* menggunakan teknologi *blockchain* adalah penelitian oleh Kurnia Hu, dkk menggunakan *multichain tools*. Penelitian ini menghasilkan sistem voting elektronik yang dapat menyimpan data yang transparan dan dapat diakses oleh publik, menjaga identitas pemilih, menyimpan data suara yang tidak dapat diubah, digandakan, atau dihapus. Berdasarkan hasil kuesioner diperoleh persentase sebesar 55,6 % pengguna menilai baik dan 44,4 % pengguna menilai baik sekali mengenai keseluruhan aplikasi (Kurnia Hu, et al., 2019). Aplikasi *e-voting* selanjutnya adalah SEVA: *Secure E-Voting Application*. Sistem dirancang bertujuan untuk memungkinkan bagi para pemilih dapat melakukan pemilihan melalui *web* tanpa terbatas oleh lokasi. Aplikasi dikembangkan dengan Azure Service Fabric menggunakan arsitektur *microservice*. Aplikasi ini diterapkan pada *cluster five-node* menggunakan Server Fabric Local Cluster (Johari, et al., 2020).

Penelitian selanjutnya adalah penerapan teknologi *blockchain* dalam layanan *E-commerce* pada *OJS (Open Journal System)*. Penggunaan teknologi

blockchain dalam penelitian ini bertujuan untuk mengefisiensi manajemen identitas, membangun sistem pelacakan dan mengidentifikasi keaslian produk, dapat menyinkronkan data yang tersimpan dalam *blockchain* ke semua jaringan pengguna, dapat menjadikan sistem pembayaran yang lebih mudah, efisien dan terdokumentasikan dengan baik. Serta kemudahan bagi mahasiswa, dosen dan pihak lain dalam melakukan proses transaksi (Rahardja, et al., 2020).

2.2 Blockchain

Blockchain adalah teknologi pencatatan transaksi yang menghilangkan pihak ketiga atau *central authority* dalam melakukan pertukaran data atau transaksi. *Blockchain* juga dapat dikatakan sebagai buku besar digital dimana setiap transaksi yang terjadi di jaringan *blockchain* akan dicatat dan diamankan di dalam *peer* atau *node* yang terlibat di jaringan *blockchain*. Secara umum cara kerja *blockchain* adalah transaksi yang terjadi di jaringan *blockchain* akan dicatat dalam blok. Blok akan menghasilkan sebuah *hash*. *Hash* blok pada transaksi sebelumnya akan dicatat di blok saat ini. Begitu pula *hash* blok saat ini akan dicatat di blok selanjutnya sehingga akan menghasilkan struktur seperti rantai (Pierro, 2017).



Gambar 2. 1 Arsitektur *Blockchain*

Jika ada pihak yang ingin melakukan kecurangan dengan cara mengubah data transaksi dalam suatu blok, maka pihak tersebut harus melakukan *hashing* ulang dari blok yang diubah hingga blok yang terjadi saat ini. Tidak hanya itu, karena blok disimpan secara terdistribusi, maka pihak tersebut juga harus mengubah data blok di setiap *node* yang terlibat dalam jaringan. Hal inilah yang menjadi kekuatan *blockchain* sehingga dapat menghilangkan *central authority* dalam melakukan transaksi.

2.3 Ethereum

Ethereum adalah sebuah *blockchain* yang didirikan pada tahun 2015 yang digagas oleh Vitalik Buterin. Salah satu tujuan pengembangan Ethereum adalah untuk memberikan kemampuan eksekusi program komputer pada *blockchain*. Aplikasi terdesentralisasi yang memanfaatkan kemampuan eksekusi program komputer ini sering disebut dengan *dapps* (Szabo, 2018).

Ethereum digunakan pada penelitian ini karena Ethereum adalah platform *blockchain* pertama yang mengimplementasikan program yang dapat berjalan di dalam jaringan *blockchain*. Sehingga ekosistem pengembangan *smart contract* pada Ethereum sudah sangat *mature*. Selain itu *developer* dari Ethereum, yaitu Ethereum Foundation selalu waspada terhadap isu-isu keamanan yang terjadi di dalam jaringan Ethereum untuk memberikan *security update* sehingga tidak banyak isu keamanan yang fatal pada jaringan Ethereum.

2.3.1 Ethereum Accounts dan Transaction

Accounts adalah entitas yang terdapat dalam Ethereum yang dapat membuat *Transaction* dimana setiap *account* memiliki *address* yang panjangnya 20-bytes. Sebuah Etherum *account* memiliki empat *field* yaitu *nonce*, *ether balance*, *contract code* dan *storage*. *Nonce* adalah *counter* yang digunakan untuk memastikan bahwa setiap transaksi yang dibuat oleh sebuah *account* hanya diproses sekali. Ether atau (ETH) adalah mata uang digital yang digunakan untuk membayar biaya transaksi. Secara umum ada dua jenis account yang terdapat di Ethereum. Yang pertama adalah *Externally Owned Account (EOA)* yang dikontrol menggunakan *private key* dan yang kedua adalah *contract account* yang dikontrol

oleh *contract code*-nya. *Contract account* bisa dikatakan sebagai *autonomous agents* yang hidup didalam Ethereum *execution environment* yang selalu menjalankan perintah spesifik dari kode ketika menerima transaksi.

Istilah transaksi yang dimaksud dalam Ethereum adalah data yang telah ditandatangani secara digital oleh sebuah *EOA*. Sebuah transaksi berisi *address* penerima, *signature* untuk identifikasi pengirim, jumlah *ether* yang akan dikirim, *data field*, *gas* dan *gas price*. *Gas* adalah nilai yang merepresentasikan seberapa besar transaksi tersebut boleh menggunakan *resource* komputasi.

2.3.2 Ethereum Code Execution

Kode dalam kontrak Ethereum ditulis dalam sebuah bahasa *bytecode low-level* berbasis *stack* yang disebut sebagai Ethereum Virtual Machine Code atau EVM Code. Kode terdiri dari beberapa kumpulan *byte* dimana setiap kumpulan *byte* merepresentasikan sebuah operasi. Secara umum, *code execution* ada sebuah perulangan tak hingga yang melakukan *increment* penambahan satu yang dimulai dari nol hingga program mencapai *error* atau terdeteksi intruksi *return*. Setiap operasi memiliki akses ke tiga jenis penyimpanan data yaitu *stack*, *memory* dan *storage*. *Stack* adalah sebuah *contrainer last-in-first-out* dimana nilainya bisa di *pushed* atau *pop*. *Memory* adalah *expandable byte array* yang akan hilang ketika komputasi selesai dilakukan. *Storage* adalah penyimpanan persistent yang menyimpan data menggunakan model *key/value*.

2.3.3 Openethereum

Jaringan suatu *blockchain* tercipta dari jaringan komunikasi *peer-to-peer* dari beberapa *nodes*. *Nodes* adalah suatu program komputer yang berjalan berdasarkan spesifikasi suatu *blockchain*. Hal yang sama juga berlaku pada jaringan Ethereum. *Nodes* pada jaringan Ethereum adalah program yang berjalan berdasarkan spesifikasi Ethereum. Ada beberapa Ethereum *client* yang ada saat ini yaitu Geth, Openethereum, Nethermind dan Hyperledger Besu. Ethereum *client* tersebut menggunakan bahasa pemrograman yang berbeda namun tetap bisa saling berkomunikasi karena mengimplementasikan spesifikasi yang sama.

Openethereum adalah sebuah Ethereum *client* yang dibuat menggunakan bahasa pemrograman Rust. Keunggulan yang dimiliki oleh Openethereum adalah program yang ringan, kecepatan sinkronisasi dan dokumentasi yang jelas tentang bagaimana cara mengoptimalkan Openethereum untuk skenario penggunaan yang berbeda.

2.3.4 Smart Contract

Smart Contract adalah kumpulan dari kode dan data yang teridentifikasi menggunakan sebuah *address* pada *blockchain* Ethereum. *Smart contract* merupakan salah satu jenis Ethereum *account* sehingga sebuah *smart contract* bisa menyimpan ETH dan mengirim transaksi di jaringan *blockchain*. Akan tetapi *smart contract* tidak dikendalikan oleh seorang pengguna atau EOA. Sebuah *smart contract* akan di-deploy ke jaringan *blockchain* dan bekerja sesuai kode yang dimilikinya. Seorang pengguna bisa berinteraksi dengan sebuah *smart contract* dengan cara membuat sebuah transaksi dengan menggunakan *address smart contract* sebagai *recipient*.

Kode *smart contract* dapat dibuat dengan bahasa pemrograman yang lebih mudah dipahami yaitu Solidity. Solidity adalah bahasa pemrograman yang diusulkan oleh Gavin Wood pada tahun 2014. Hingga saat ini Solidity menjadi bahasa pemrograman terpopuler untuk membuat *smart contract* karena Solidity didesain menyerupai ECMAScript sehingga lebih familiar di kalangan *developer*.

Istilah *smart contract* diperkenalkan oleh Nick Szabo pada tahun 1996. Kontrak adalah sebuah kumpulan pernyataan yang disetujui dalam rangka “pertemuan pikiran” dalam sebuah hubungan misalnya hubungan bisnis. Kontrak tradisional seperti ini tidak bisa dilakukan secara langsung oleh kedua belah pihak karena diperlukan pihak ketiga seperti notaris dan beberapa saksi untuk memastikan keabsahan kontrak. Karena untuk membuat kontrak tradisional memerlukan banyak biaya yang besar dan waktu yang lama. Ide *smart contract* itu sendiri tercetus karena proses digitalisasi yang berkembang dengan pesat. Definisi *smart contract* menurut Nick Szabo adalah kumpulan pernyataan yang disetujui dalam bentuk digital dalam sebuah protokol dimana kedua pihak harus terlibat dalam protokol

tersebut. Definisi tersebut sesuai dengan pengertian program komputer yang berjalan pada Ethereum, dimana pihak yang akan melakukan kontrak harus tergabung di dalam jaringan Ethereum dan pernyataan yang disetujui dicantumkan di dalam program komputer. Maka dari itu program komputer yang berjalan pada Ethereum disebut *smart contract* (Szabo, 2018).

2.3.5 Konsensus

Secara umum pengertian konsensus adalah sebuah kesepakatan bersama yang telah dicapai oleh suatu kelompok terhadap suatu masalah. Konsensus dalam *blockchain* sangat penting karena data yang terdesentralisasi sehingga sangat memungkinkan terjadinya perbedaan daya yang tercatat. Dengan adanya algoritma konsensus ini maka jaringan tidak akan kebingungan ketika terjadi perbedaan pencatatan data dan menentukan data mana yang akan digunakan. Sebagian besar ekosistem *blockchain* saat ini menggunakan algoritma *Proof of Work* sebagai konsensus di jaringan *blockchain*.

Cara kerja algoritma konsensus *proof of work* adalah setiap *node* yang tergabung dalam sebuah jaringan *blockchain* akan berlomba untuk memproses transaksi yang terjadi dalam kurun waktu tertentu dan *node* tercepat yang bisa menyelesaikan pembuatan blok baru akan menyebarkan blok tersebut ke jaringan. *Account* yang terhubung dengan *node* tersebut akan mendapatkan insentif sesuai dengan mata uang jaringan *blockchain*. Sedangkan algoritma konsensus *Proof of Authority*, dalam jaringan sudah di atur *account* mana saja yang boleh melakukan pembuatan blok. Karena *account* yang melakukan pembuatan blok sudah ditentukan sehingga tidak akan terjadi persaingan dalam hal pembuatan blok sehingga tingkat *difficulty* dari pembuatan blok bisa diturunkan. Dengan tingkat *difficulty* yang rendah maka jaringan dapat berjalan dengan komputer dengan daya komputasi rendah sehingga bisa menghemat biaya infrastruktur.

2.3.6 Gas, Gas Fee dan Gas Limit

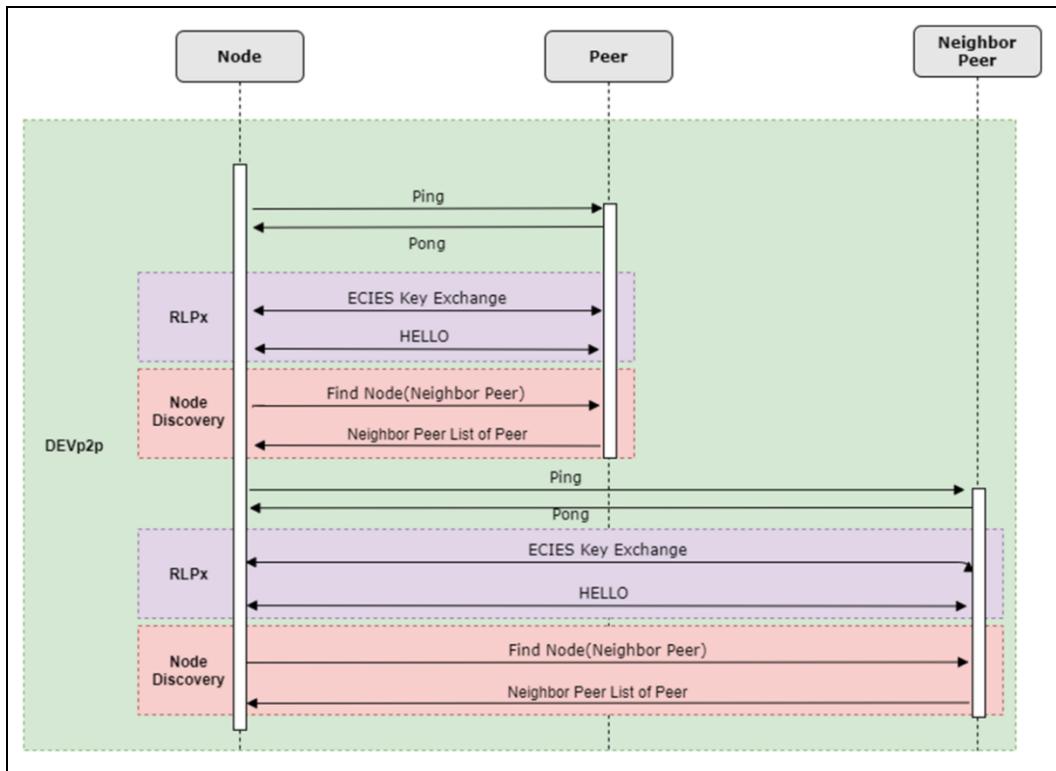
Gas adalah sebuah kalkulasi untuk menentukan seberapa besar biaya yang dibutuhkan untuk melakukan tindakan pada jaringan *blockchain*. Semua aktivitas

yang dilakukan pada jaringan Ethereum seperti pengiriman ETH hingga pemanggilan metode *smart contract* membutuhkan *gas*. *Gas* yang dibutuhkan harus dibayar menggunakan ETH. Besarnya biaya *gas* yang harus dibayarkan ini biasanya disebut dengan *gas fee*. *Gas limit* adalah batas maksimal *gas* yang boleh digunakan untuk melakukan sebuah transaksi. *Gas limit* sangat penting karena jika tidak ada *gas limit* maka jika terjadi suatu kesalahan misalnya kesalahan *smart contract* yang menyebabkan pemanggilan berulang akan terus menggunakan ETH yang terdapat pada *account eksekutor* untuk membayar *gas* hingga ETH pada *account* tersebut habis.

2.3.7 Protokol Komunikasi

Jaringan Ethereum adalah jaringan yang terdiri dari banyak *node* yang dikelola oleh pihak-pihak independen sehingga sangat mustahil untuk memelihara jaringan secara manual oleh sebuah pihak sentral karena *uptime* dari *node-node* yang terhubung berbeda-beda. Untuk memelihara jaringan tetap berjalan tanpa diganggu oleh *uptime node* yang berbeda-beda sehingga diperlukan sebuah *discovery protocol* untuk melakukan *lookup* terhadap *node* yang masih aktif pada jaringan. *Discovery protocol* yang digunakan oleh Ethereum dibuat seperti desain jaringan *peer-to-peer* yang didesain oleh Petar Maymounkov and David Mazières pada tahun 2002 yang diberi nama *Kademlia*. Secara sederhana *Kademlia* adalah *Distributed Hash Table (DHT)* yang menyimpan informasi alamat *node* yang terhubung pada suatu *node* sehingga masing-masing *node* pada jaringan bisa melakukan *lookup* secara berkala. *Discovery protocol* yang digunakan oleh Ethereum berjalan pada protokol *UDP*.

RLPx adalah protokol komunikasi berbasis *TCP* yang digunakan oleh Ethereum *node* untuk bertukar data. Protokol ini wajibkan setiap *node* yang akan melakukan pertukaran data untuk melakukan serangkaian proses enkripsi. Dalam jaringan Ethereum, *request* dan *response* memiliki waktu timeout sebesar 500 milidetik sedangkan operasi untuk protokol *RLPx* memiliki timeout sebesar 5 detik. Tabel yang menyimpan data peer diupdate secara berkala dalam waktu 30 detik.



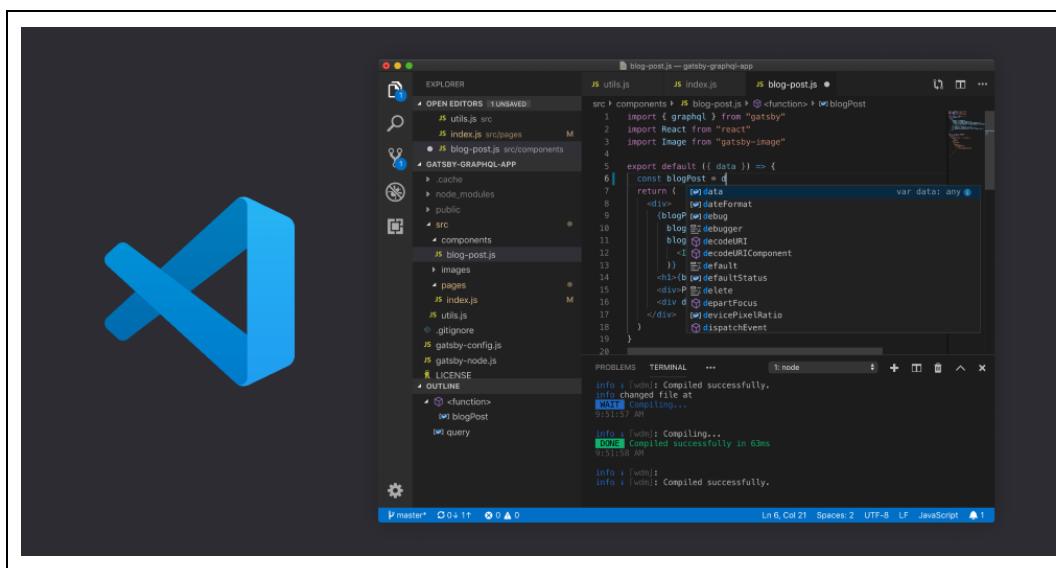
Gambar 2.2 Diagram Komunikasi Node Ethereum

Gambar 2.2 menjelaskan bahwa untuk memeriksa apakah sebuah *node* masih aktif atau tidak dengan mengirimkan pesan "ping". Jika dibalas dengan "pong" maka proses pengiriman data akan dilakukan yang diawali dengan pertukaran kunci ECIES. Sedangkan jika tidak ada balasan maka alamat *peer* akan dihapus dari DHT. Proses sinkronisasi terjadi dengan cara mengirim dan menerima data *block headers*, *block bodies* dan *receipts* secara berkala menggunakan protokol RLPx.

2.4 VSCode

Visual Studio Code adalah sebuah teks editor ringan dan handal yang dibuat oleh Microsoft untuk sistem operasi *multiplatform*, artinya tersedia juga untuk versi Linux, Mac, dan Windows. Teks editor ini secara langsung mendukung bahasa pemrograman JavaScript, Typescript, dan Node.js, serta bahasa pemrograman lainnya dengan bantuan *plugin* yang dapat dipasang melalui *marketplace* Visual Studio Code (seperti C++, C#, Python, Go, Java, dst).

Fitur-fitur yang disediakan oleh Visual Studio Code, diantaranya *Intellisense*, *Git Integration*, *Debugging*, dan fitur ekstensi yang menambah kemampuan teks editor. *Editor* ini dirancang untuk pengembang yang bekerja dengan teknologi *open source cloud* serta menggunakan NET untuk memberikan dukungan untuk ASP. Antarmukanya tergolong mudah untuk digunakan karena didasarkan pada gaya *explorer* umum, dengan *panel* di sebelah kiri yang menunjukkan semua *file* dan *folder* serta *panel editor* di sebelah kanan, yang menunjukkan isi dari *file* yang telah dibuka.



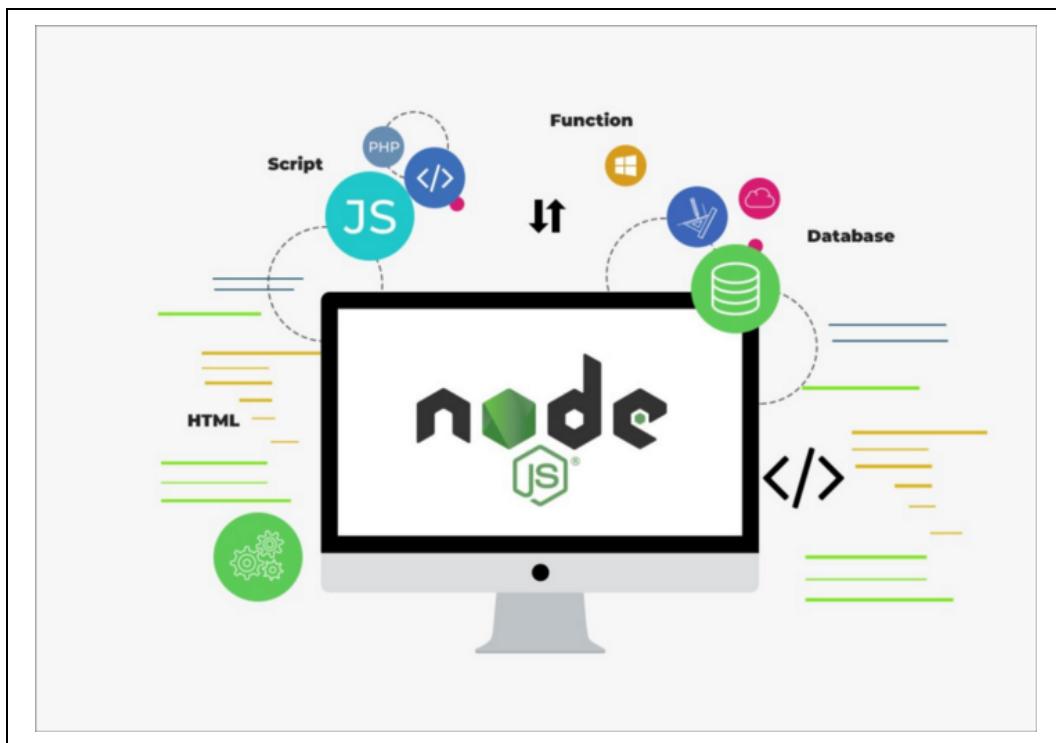
Gambar 2.3 Visual Studio Code

Microsoft telah menyediakan dokumentasi yang lengkap untuk membantu dan memudahkan penggunaan bagi para *developer*. Visual Studio Code ditargetkan pada pengembang scripting *server-side* dan dapat digunakan oleh siapa saja untuk membangun aplikasi berbasis *web* (Code, 2021).

2.5 NodeJS

Node.js merupakan salah satu *platform* pengembang yang dapat digunakan untuk membuat aplikasi berbasis *Cloud*. Node.js dikembangkan dari *engine* JavaScript yang dibuat oleh Google untuk *browser* Chrome ditambah dengan *libuv* serta beberapa pustaka lainnya. Node.js menggunakan JavaScript sebagai bahasa pemrograman dan *event-driven, non-blocking I/O (asynchronous)* model yang membuatnya ringan dan efisien. Node.js memiliki fitur built-in HTTP *server*

library yang menjadikannya mampu menjadi sebuah *web server* tanpa bantuan *software* lainnya seperti Apache dan Nginx.



Gambar 2. 4 Node.js

Sumber : <https://ichi.pro/id>

Node.js adalah sebuah *runtime environment* dan *script library*. Sebuah *runtime environment* adalah sebuah *software* yang berfungsi untuk mengeksekusi, menjalankan dan mengimplementasikan fungsi-fungsi serta cara kerja inti dari suatu bahasa pemrograman. Sedangkan *script library* adalah kumpulan, kompilasi atau bank data berisi skrip/kode-kode pemrograman. Node.js dibangun menggunakan JavaScript dan C++, terdapat arsitektur serta fungsi dari Google V8 di dalamnya yang berfungsi sebagai *compiler* ditulis dalam C++ dan library Libuv bekerja untuk menangani operasi *asynchronous I/O* dan *main event loop* (Wilson, 2013).

2.5.1 NestJS

Nest (NestJS) adalah *framework* untuk membangun *server-side applications* Node.js yang efisien. NestJS menggunakan JavaScript progresif yang dibangun dengan mendukung TypeScript secara penuh dan menggabungkan

elemen *OOP (Object Oriented Programming)*, *FP (Functional Programming)*, dan *FRP (Functional Reactive Programming)*. Nest menggunakan kerangka kerja *Server HTTP default* Express dan secara opsional dapat dikonfigurasi untuk menggunakan Fastify juga. Nest menyediakan level abstraksi di atas *framework* Node.js (Express / Fastify), tetapi juga mengekspos *API*-nya secara langsung ke *developer* sehingga memberi kebebasan bagi *developer* untuk menggunakan banyak sekali modul pihak ketiga yang tersedia untuk *platform* yang mendasarinya.



Gambar 2. 5 NestJS

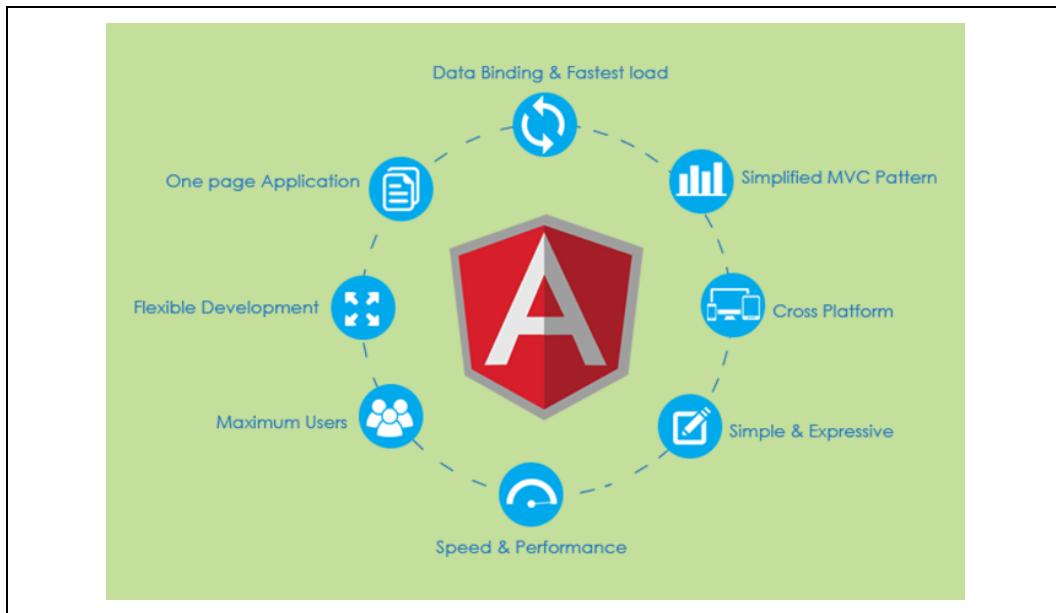
Sumber: <https://www.digitalkode.com/>

NestJS memiliki beberapa kelebihan atau fitur unggulan, diantaranya memberikan fleksibilitas dengan memungkinkan untuk penggunaan pustaka lain dengan adanya arsitektur modular (*extensible*), juga merupakan ekosistem yang dapat beradaptasi untuk semua jenis *server-side applications* serta bersifat progresif karena memanfaatkan fitur JavaScript yang terbaru (Mysliwiec, 2017).

2.5.2 Angular

Angular adalah *framework open-source single-page apps* berbasis *TypeScript*. Angular dirancang dari penulisan ulang Angular JS oleh Google. Angular dapat dikembangkan di seluruh *platform* baik *web*, *mobile web*, *native mobile*, maupun *native desktop*. Angular melakukan proses *load* dengan cepat melalui *component router* baru, yang memberikan *code-splitting* otomatis sehingga pengguna hanya perlu melakukan *load* kode yang diperlukan untuk *re-render* tampilan yang diminta. Angular dirancang untuk membuat sebuah pembaruan

semudah mungkin, sehingga *developer* dapat memanfaatkan perkembangan terbaru dengan *effort* yang minimal.



Gambar 2. 6 Angular

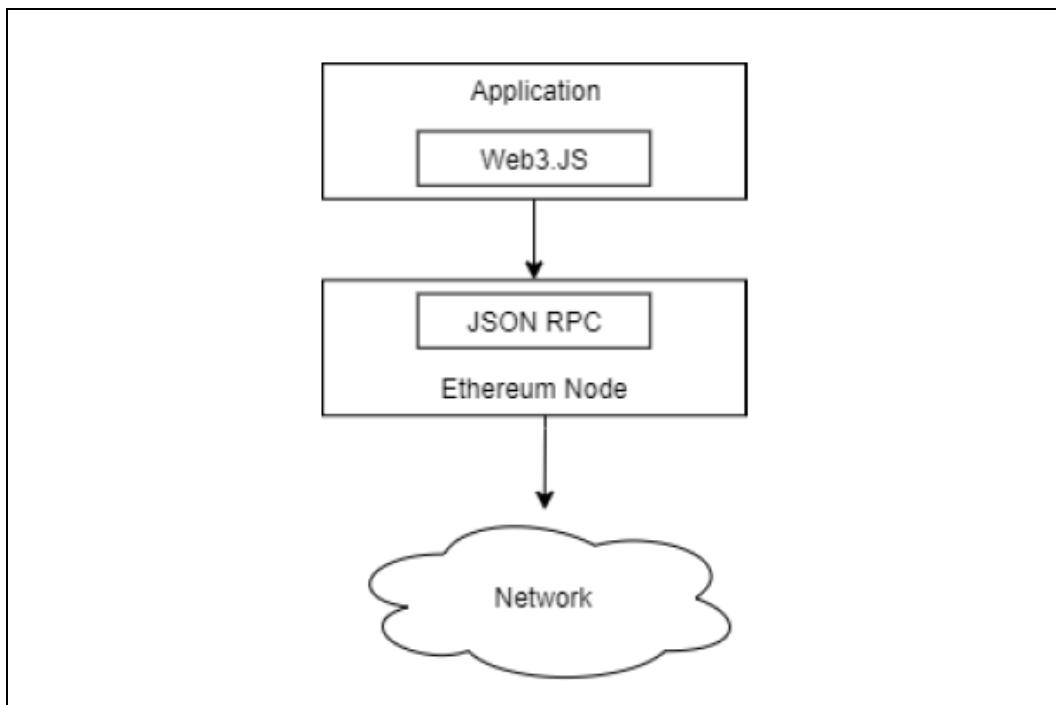
Angular menyediakan beberapa *library* pihak pertama. *Library-library* tersebut berfungsi untuk memperluas fungsionalitas aplikasi sekaligus berfokus pada fitur yang membuat aplikasi yang dikembangkan menjadi lebih unik. *Library* tersebut dirancang untuk diintegrasikan dengan lancar dan diperbarui secara bersamaan dengan kerangka kerja Angular. *Library-library* yang disediakan, antara lain (Angular, 2010) :

- a. *Angular Router*, navigasi dan *routing client-side* lanjutan berdasarkan komponen Angular. Mendukung *lazy-loading*, *nested routes*, *custom path matching* dan lainnya,
- b. *Angular Forms*, sistem yang seragam untuk partisipasi dan validasi *form*.
- c. *Angular HttpClient*, HTTP *client* yang kuat yang dapat mendukung komunikasi client -server yang lebih baik.
- d. *Angular Animations*, sistem yang kaya untuk menggerakkan animasi berdasarkan status aplikasi.
- e. *Angular PWA*, tools untuk membangun *Progressive Web Applications* (*PWA*) termasuk *service worker* dan *web app manifest*.

- f. *Angular Schematics, automated scaffolding, refactoring, dan update tools* yang menyederhanakan pengembangan dalam skala besar.

2.5.3 Web3JS

Web3js memungkinkan untuk berinteraksi dengan *node Ethereum* lokal atau jarak jauh, menggunakan koneksi *HTTP* atau *IPC*. Web3js merupakan pustaka yang memungkinkan untuk melakukan tindakan pengiriman seperti mengirim *ether* dari akun satu ke akun lain, membaca dan menulis *smart contract* dan lain sebagainya.



Gambar 2.7 Interaksi Web3.JS dengan Node Ethereum

Web3.js dapat digunakan untuk terhubung ke jaringan Ethereum melalui *node Ethereum* yang memungkinkan akses melalui HTTP. Akses tersebut dapat berupa *node* lokal, *node* yang di-hosting oleh penyedia DApp, atau *gateway* publik seperti Infura, yang mengoperasikan titik akses Ethereum gratis. Salah satu cara umum untuk mengintegrasikan aplikasi *browser web* dengan Ethereum adalah dengan menggunakan ekstensi *browser* Metamask dengan kombinasi Web3js (Web3JS, 2016).

2.6 MySQL

MySQL merupakan *software* yang tergolong sebagai *DBMS (Database Management System)* yang bersifat *open source*. *Open source* menyatakan bahwa *software* ini dilengkapi dengan *source code* selain bentuk *executable*-nya atau kode yang dapat dijalankan secara langsung dalam sistem operasi. MySQL memiliki fungsi-fungsi MySQL yang digunakan untuk mengakses *database server* MySQL. Fungsi ini berguna untuk mengantarkan perintah SQL pada PHP menuju ke *server* sehingga perintah tersebut dapat dieksekusi oleh semua *server* MySQL.

Tabel 2. 1 MySQL Function

No.	Function	Kegunaan
1.	<i>Mysql_connect()</i>	Membuat hubungan ke <i>database MySQL</i> yang terdapat pada suatu <i>host</i>
2.	<i>Mysql_close()</i>	Menutup hubungan ke <i>database MySQL</i>
3.	<i>Mysql_select_db()</i>	Memilih <i>database</i>
4.	<i>Mysql_query()</i>	Mengeksekusi permintaan terhadap sebuah tabel atau sejumlah tabel
5.	<i>Mysql_db_query()</i>	Menjalankan suatu permintaan terhadap suatu <i>database</i>
6.	<i>Mysql_num_rows()</i>	Memperoleh jumlah baris dari suatu hasil permintaan yang menggunakan <i>SELECT</i>
7.	<i>Mysql_affected_rows()</i>	Memperoleh jumlah baris yang dikenai operasi <i>INSERT, DELETE, UPDATE</i>
8.	<i>Mysql_num_fields()</i>	Memperoleh jumlah kolom pada suatu hasil permintaan
9.	<i>Mysql_fetch_row()</i>	Menghasilkan <i>array/baris</i> yang berisi seluruh kolom dari sebuah baris pada suatu himpunan hasil
10.	<i>Mysql_fetch_array()</i>	Menghasilkan <i>array/baris</i> yang berisi seluruh kolom dari sebuah baris pada suatu himpunan hasil yang akan disimpan dua kali pada <i>array</i> hasil
11.	<i>Mysql_fetch_field()</i>	Menghasilkan informasi suatu kolom
12.	<i>Mysql_data_seek()</i>	Memindahkan pointer pada suatu himpunan hasil supaya menunjuk ke baris tertentu
13.	<i>Mysql_field_seek()</i>	Memindahkan pointer pada suatu himpunan hasil supaya menunjuk ke kolom tertentu
14.	<i>Mysql_create_db()</i>	Membuat <i>database MySQL</i>
15.	<i>Mysql_drop_db()</i>	Menghapus <i>database MySQL</i>

No.	Function	Kegunaan
16.	<i>Mysql_list_dbs()</i>	Menghasilkan daftar <i>database MySQL</i>
17.	<i>Mysql_list_tables()</i>	Memperoleh daftar nama tabel dalam suatu <i>database</i>
18.	<i>Mysql_list_fields()</i>	Memperoleh daftar nama kolom dalam suatu <i>database</i>
19.	<i>Mysql_fetch_assoc()</i>	Mendapatkan array baris dari suatu <i>recordset</i>
20.	<i>Mysql_fetch_lengths()</i>	Mendapatkan panjang baris pada setiap isi <i>field</i>
21.	<i>Mysql_fetch_object()</i>	Menghasilkan baris dari <i>recordset</i> sebagai sebuah objek
22.	<i>Mysql_field_len()</i>	Mendapatkan informasi panjang maksimum <i>field</i> dalam sebuah <i>recorset</i>
23.	<i>Mysql_field_name()</i>	Mendapatkan informasi nama <i>field</i> dalam <i>recordset</i>
24.	<i>Mysql_ping()</i>	Memeriksa koneksi <i>server</i> dan akan mencoba untuk melakukan koneksi ulang jika koneksi terputus

Tabel 2.1 merupakan fungsi-fungsi dalam MySQL yang dapat digunakan untuk mengantarkan perintah SQL pada PHP menuju ke *server* sehingga perintah tersebut dapat dieksekusi oleh semua *server* MySQL.

Selain fungsi-fungsinya, MySQL juga memiliki beberapa keunggulan, seperti dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan masih banyak lainnya. MySQL didistribusikan sebagai perangkat lunak sumber terbuka, dibawah lisensi GPL sehingga dapat digunakan secara gratis serta dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik. MySQL memiliki kecepatan yang tinggi dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu. MySQL memiliki ragam tipe data yang sangat kaya, seperti *signed / unsigned integer, float, double, char, text, date, timestamp*, dan lain-lain. MySQL memiliki antar muka (*interface*) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi *API (Application Programming Interface)* (Kadir, 2008).

2.7 JSON

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari bahasa

pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 Desember 1999. *JSON* merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dan lainnya (Derizal, 2011).

2.8 Google Cloud Platform

Google Cloud Platform merupakan layanan *public cloud computing* dari Google yang terdiri dari beragam layanan. *Platform* dari Google ini menyediakan beragam layanan *hosting* mulai dari komputasi, *storage* dan *application development* yang berjalan pada *hardware* Google. Google Cloud Platform Service dapat diakses oleh pengembang *software*, *administrator cloud* dan profesional IT lainnya menggunakan internet publik atau melalui koneksi jaringan *dedicated* (Google, 2021).

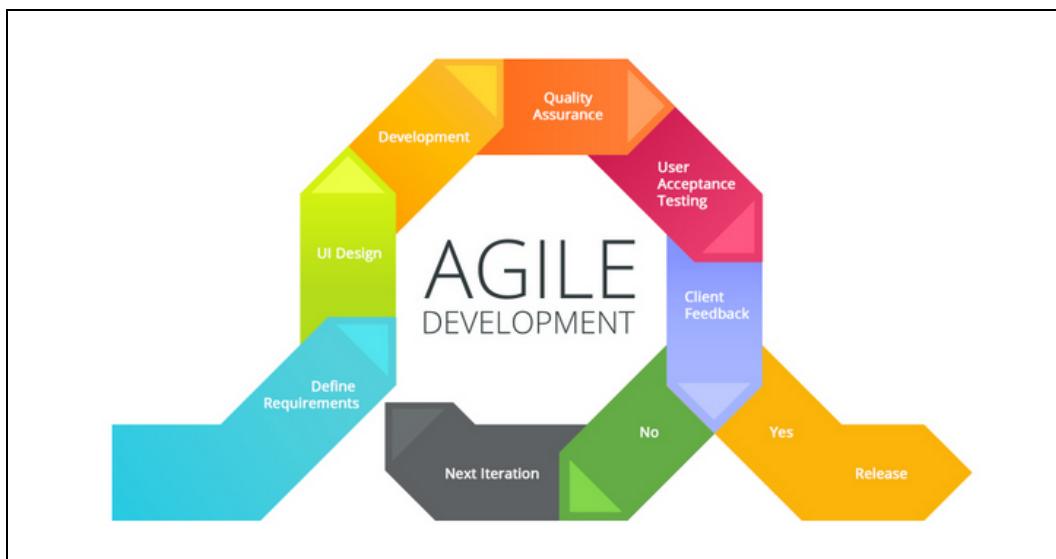
2.8.1 Compute Engine

Google Compute Engine memungkinkan untuk menjalankan beban kerja komputasi skala besar dengan infrastruktur yang sama yang menjalankan Google Search, Gmail dan Ads. Pengguna dapat meluncurkan *virtual machines* sesuai permintaan, mengelola konektivitas jaringan menggunakan solusi jaringan yang sederhana namun fleksibel, dan mengakses berbagai alternatif penyimpanan data dari *virtual machines*. Fitur-fitur Google Compute Engine antara lain (Google, 2021) :

- a. *Full Virtual Machines*: Mesin virtual yang di-*hosting* kernel yang menjalankan Ubuntu atau CentOs. Meluncurkan 1, 2, 4, atau 8 *virtual core instances* dengan memori 3.75 GB per *virtual core*.
- b. *Flexible Storage*: menawarkan opsi penyimpanan yang berbeda untuk memenuhi berbagai kebutuhan.
- c. *Flexible networking*: menawarkan solusi jaringan yang memudahkan untuk menghubungkan *virtual machines* satu sama lain dan ke Internet.
- d. *Open tooling*: menawarkan *UI* sederhana dan *command-line tooling* untuk mengonfigurasi dan meluncurkan *virtual machines*.

2.9 Agile Software Development Method

Metode *Agile* pertama kali diperkenalkan oleh Kent Beck bersama dengan 16 rekannya. Metode *agile* merupakan sekumpulan metode pengembangan perangkat lunak yang berbasis pada pengembangan iteratif yang dilakukan dengan kolaborasi yang terorganisir. Metode *agile* berfokus pada perkembangan yang cepat, perangkat lunak yang dirilis bertahap, mengurangi *overhead* proses, dan menghasilkan kode berkualitas tinggi dan pada proses perkembangannya melibatkan pelanggannya secara langsung (Sommerville, 2011).



Gambar 2.8 Agile Software Development Method

Gambar 2.8 *Agile* sendiri merupakan gabungan dari metode *incremental* dan *iterative*. Dibandingkan dengan metode *waterfall*, *agile* tidak memiliki fase-fase yang paten. Metode *agile* berproses iteratif dengan perputaran yang pendek. Kebutuhan direncanakan, diimplementasikan, diuji, dan dievaluasi secara berulang.

Metode *agile* masih terbagi menjadi beberapa jenis bagian, diantaranya *Adaptive Software Development (ASD)*, *Agile Modelling (AM)*, *Crystal*, *Dynamic System Development Method (DSDM)*, *Extreme Programming (XP)*, *Feature Driven Development (FDD)*, *Rational Unified Process*, dan *Scrum Methodology*. Jenis *agile* yang paling sering digunakan adalah *Adaptive Software Development*

(ASD), Dynamic System Development Method (DSDM), Extreme Programming (XP), dan Scrum (Pressman, 2010).

Penelitian ini akan menggunakan jenis metode *Scrum*. Walaupun sebenarnya metode ini biasanya digunakan secara tim, namun secara khusus tidak ada larangan untuk menggunakannya secara perorangan. Metode *scrum* membagi proses *development* menjadi beberapa *sprint*. *Sprint* ini akan bertindak sebagai *milestone* dalam pembuatan *software*. Berikut ini merupakan beberapa tahapan dalam *Scrum*.

2.9.1 Backlog

Tahap ini melakukan proses menyusun rincian prioritas pada fitur-fitur yang akan dibangun pada sistem yang akan dikembangkan. Umumnya, tahapan ini bertujuan untuk merancang pekerjaan-pekerjaan yang akan dilakukan berdasarkan permintaan atau kebutuhan *user*.

2.9.2 Sprints

Tahapan selanjutnya yaitu memasukan pekerjaan-pekerjaan yang harus dilakukan dan menyusun prioritasnya ke dalam *sprints*. Proses ini menyusun kegiatan yang akan dilakukan untuk memenuhi kebutuhan dan ditetapkan dalam *backlog* dengan durasi realisasi selama hari kerja yang ditentukan. Setiap sprint mengerjakan hampir keseluruhan proses yang disebutkan dalam *backlog* atau paling sedikit 3 dari 4 proses.

2.9.3 Scrum Meeting

Tahap ini dilakukan dengan menganalisis dari sisi pengembang untuk mengetahui seberapa banyak kemajuan dari kegiatan pengembangan sistem yang telah dilakukan atau dicapai.

2.9.4 Demos

Langkah selanjutnya adalah memperlihatkan fitur-fitur *software* yang telah dihasilkan dalam proses pengembangan untuk dievaluasi oleh pengguna sesuai dengan waktu yang telah ditentukan.

Sementara itu, di dalam setiap iterasi kegiatan pengembangan sistem yang telah dijelaskan sebelumnya, terdapat beberapa rangkaian kegiatan di dalamnya, yaitu sebagai berikut.

- a. Analisa Kebutuhan Sistem, yaitu Asesmen terhadap pengguna untuk menggali secara detail *software requirement* yang dibutuhkan oleh pengguna.
- b. Desain, yaitu tahap perancangan sistem, mulai dari desain arsitektur sistem, desain proses bisnis, desain *database*, hingga desain *user interface*.
- c. *Code Generation*, yaitu tahap penulisan kode program dan manajemen *database*
- d. *Testing*, yaitu proses meminimalisir kesalahan (*error*) dan memastikan *output* yang dihasilkan telah sesuai dengan *user requirement*.
- e. *Support*, yaitu proses menindaklanjuti perubahan yang diperlukan setelah proses *testing* dan evaluasi sesuai dengan kebutuhan pengguna.

BAB III

METODOLOGI DAN PERANCANGAN SISTEM

BAB III ini memuat metode yang digunakan dalam penelitian beserta dengan proses pembuatan laporan yang meliputi tempat dan waktu penelitian, sumber data dan metode pengumpulan, instrumen perancangan, dan gambaran umum sistem.

3.1 Tempat dan Waktu Penelitian

Penelitian ini bertempat di Kampus Bukit Universitas Udayana, Fakultas Teknik, Program Studi Teknologi Informasi yang berlokasi di Jalan Kampus Bukit UNUD Jimbaran, Kabupaten Badung, Bali. Kampus Bukit Universitas Udayana dipilih karena memiliki semua aspek pendukung agar penelitian dapat berjalan dengan baik. Penelitian dilaksanakan pada semester genap tahun ajaran 2019-2020 sampai dengan semester genap tahun ajaran 2021-2022 bulan Januari.

3.2 Metode Penelitian

Metode penelitian yang digunakan pada pembuatan sistem *e-voting* ini adalah *Agile Software Development Methods*. Seperti yang telah dijelaskan pada pembahasan sebelumnya, *Agile Software Development Methods* masih terbagi lagi menjadi beberapa jenis bagian dan penelitian ini menggunakan jenis metode *Scrum*. Metode *scrum* membagi proses *development* menjadi beberapa *sprint*. *Sprint* ini akan bertindak sebagai *milestone* dalam pembuatan *software*. Berikut ini merupakan implementasi penelitian ini berdasarkan beberapa tahapan dalam *Scrum*.

3.2.1 Penerapan Tahap Backlog

Tahap ini dilakukan dengan proses menyusun rincian prioritas pada fitur-fitur yang akan dikembangkan pada sistem *e-voting*. Berdasarkan implementasi tahap *backlog* ini diperoleh hasil bahwa proses-proses yang akan dilakukan pada perancangan sistem *e-voting* adalah perancangan *database*, perancangan *API*

backend, integrasi dengan *ethereum node*, dan perancangan *frontend*. Isi pada masing-masing proses secara rinci ditambahkan setiap memulai siklus pengembangan karena satu proses dan lainnya tidak diselesaikan secara langsung dalam suatu waktu sehingga dalam setiap proses dapat terjadi perubahan dan penambahan lebih lanjut.

3.2.2 Penerapan Tahap Sprints

Proses ini menyusun kegiatan dalam *backlog* dengan durasi realisasi selama hari kerja yang ditentukan yaitu selama 2 minggu. Prioritas pekerjaan yang harus dikerjakan sesuai waktu yang ditentukan adalah perancangan *database*, perancangan *API backend*, integrasi dengan *ethereum node*, sedangkan untuk perancangan *frontend*, di awal pengembangan masih belum mendapat prioritas utama, namun akan menjadi fokus setelah proses-proses lainnya mencapai hasil dengan persentase yang sudah lebih besar.

3.2.3 Penerapan Tahap Scrum Meeting

Tahap ini dilakukan dengan pengujian jenis *automated testing* yaitu *unit testing* dan *e2e testing* terhadap 10 *file* dengan masing-masing *file* terdiri atas kurang lebih 5 skenario pengujian.

3.2.4 Penerapan Tahap Demos

Tahapan ini dalam pengembangan sistem *e-voting* dibantu oleh Dosen Pembimbing 1 dan Dosen Pembimbing 2 tugas akhir secara rutin sebagai pihak yang memberi masukan dan saran atas pengembangan yang telah dilakukan untuk kemudian dilakukan proses evaluasi kembali atas hasil kerja berdasarkan proses penerapan tahap-tahap diatas.

3.3 Alat Penelitian Aplikasi

Alat penelitian adalah media pendukung untuk melakukan implementasi aplikasi dalam penelitian ini. Alat penelitian dibagi menjadi dua bagian yaitu perangkat lunak dan perangkat keras. Kebutuhan perangkat perangkat lunak dan

perangkat keras untuk implementasi aplikasi dalam penelitian ini dijelaskan pada sub bab dibawah ini.

3.3.1 Kebutuhan Perangkat Lunak Implementasi Aplikasi

Implementasi aplikasi Sistem Voting Elektronik memiliki kebutuhan perangkat lunak. Perangkat lunak yang dimaksud adalah sistem operasi berbasis Linux dan *javascript runtime*. *Virtual Machine* menggunakan sistem operasi Ubuntu versi 18.04 dan Node.js versi 14. Komunikasi dengan *Virtual Machine* memerlukan koneksi internet yang stabil. Kecepatan *ping* yang digunakan untuk implementasi Sistem Voting Elektronik adalah dibawah 500 ms.

3.3.2 Kebutuhan Perangkat Keras Implementasi Aplikasi

Implementasi aplikasi menggunakan empat *virtual machine* yang disediakan oleh Google Cloud Platform. Spesifikasi dari *virtual machine* disebutkan dalam Tabel

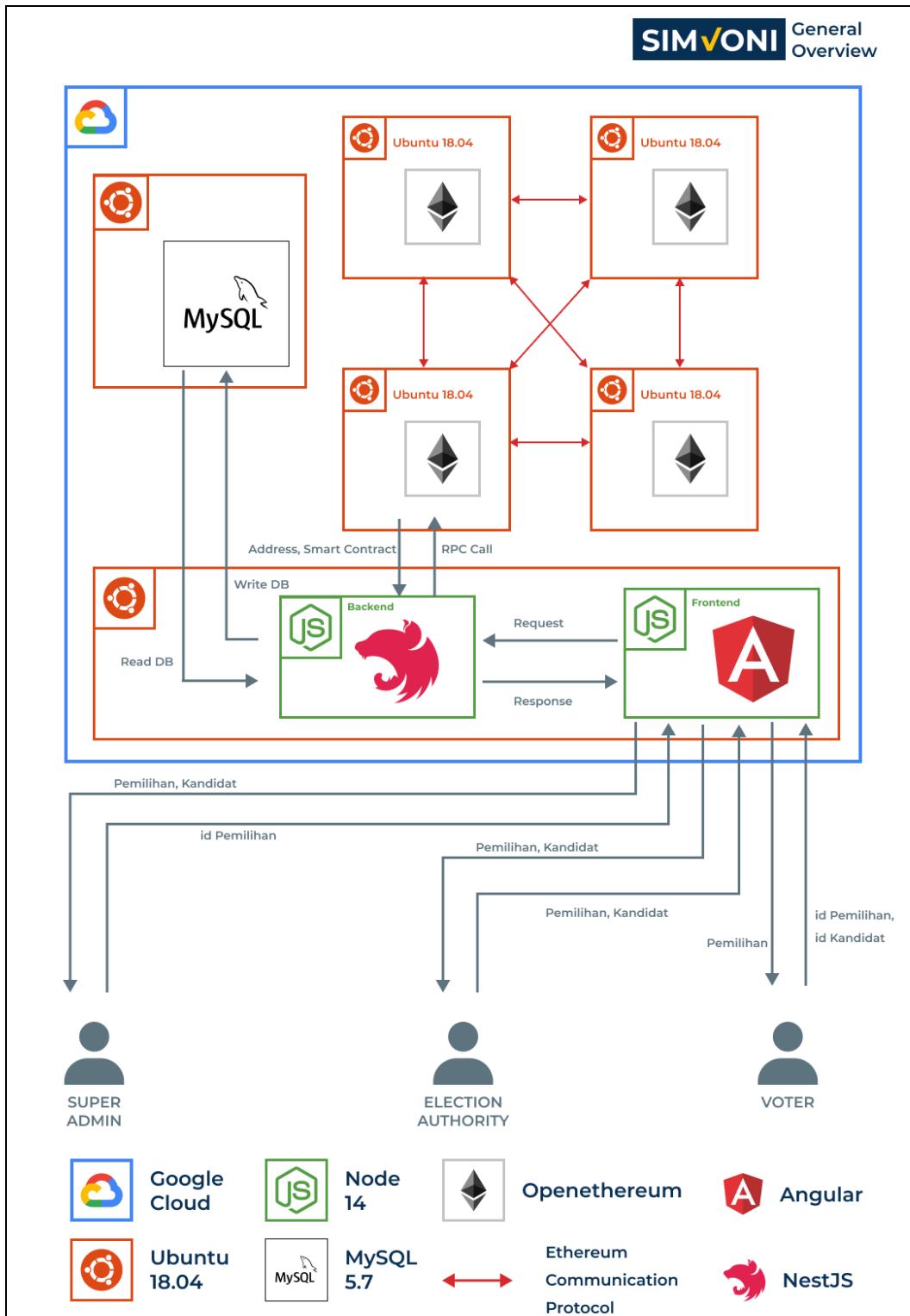
Tabel 3. 1 Kebutuhan Infrastruktur Implementasi Aplikasi

Hardware	Virtual Machine
Processor	2 vCPU
RAM	1 GB
Penyimpanan	10 GB

Tabel 3.1 menunjukkan spesifikasi dari infrastruktur yaitu *Virtual Machine* yang disediakan oleh Google Cloud. *VM* tersebut digunakan untuk melakukan implementasi terhadap Sistem Voting Elektronik.

3.4 Gambaran Umum Sistem

Sistem Voting Elektronik berbasis Ethereum *smart contract* merupakan sistem yang digunakan sebagai sarana pemilihan elektronik dengan menerapkan teknologi *blockchain*. Gambar 3.1 merupakan gambaran umum arsitektur sistem *e-voting* berbasis Ethereum *smart contract*. Gambaran umum arsitektur teknologi merupakan bagan yang menjelaskan alur kerja teknologi yang digunakan secara menyeluruh secara umum.



Gambar 3.1 Gambaran Umum Sistem

Gambar 3.1 merupakan gambaran umum dari Sistem Voting Elektronik berbasis Ethereum *smart contract*. Semua kebutuhan komputasi yaitu *virtual*

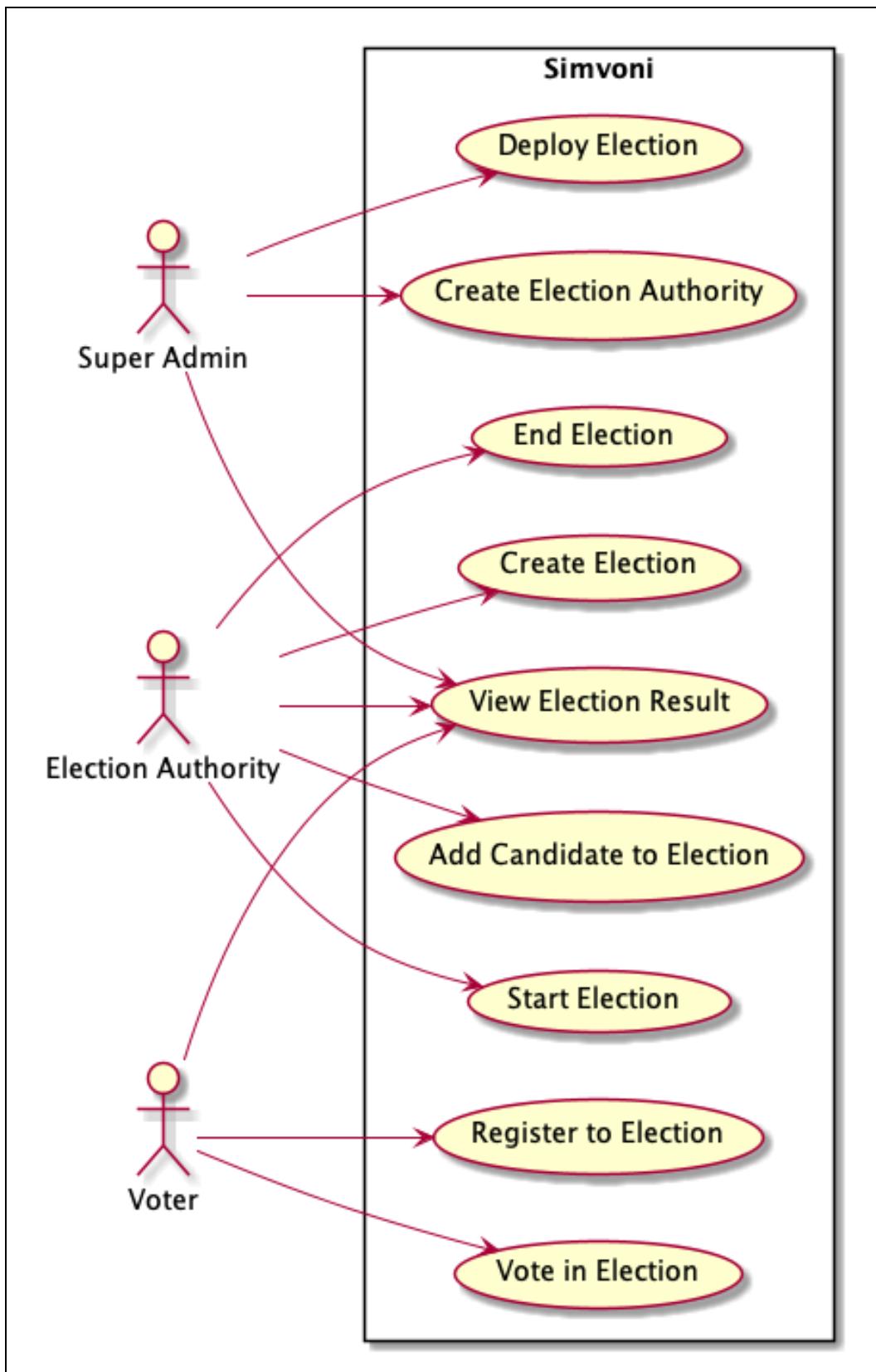
machine menggunakan penyedia Google Cloud. Terdapat empat *virtual machine* yang digunakan dalam sistem. Masing-masing *VM* menggunakan OS Ubuntu 18.04. Sebuah *VM* digunakan untuk menjalankan *RDBMS MySQL*, dua buah *VM* untuk menjalankan Openethereum dan yang terakhir digunakan untuk menjalankan dua proses *node* untuk aplikasi *backend* dan *frontend*.

Terdapat dua buah *node* Ethereum karena untuk membuat jaringan *private* yang semirip mungkin dengan jaringan *public* Ethereum misalnya Ethereum Mainnet dan Ethereum Testnet dimana terdapat lebih dari satu *node* yang saling berkomunikasi. Nestjs adalah *framework* yang digunakan untuk menjalankan *backend* sedangkan Angular adalah *framework* yang digunakan untuk menjalankan *frontend*. Baik Nestjs dan Angular dijalankan menggunakan *runtime* Nodejs versi 14.

Pertama *super admin* akan membuat akun untuk *election authority*. *Election authority* akan membuat membuat pemilihan dan menambahkan kandidat pada pemilihan. Pemilihan akan di *review* oleh *super admin* apakah diijinkan untuk di-deploy ke *blockchain*. Jika pemilihan sudah di-deploy maka *voter* dapat melakukan pemilihan dan data akan terekam pada *blockchain*. Kemudian pemilihan akan dihentikan oleh *election authority* dan *voter* dapat melihat hasil dari pemilihan tersebut.

3.4.1 Use Case Diagram

Use case diagram merupakan representasi interaksi *user* dengan sistem dan menggambarkan spesifikasi dari *use case*. *Use case* untuk sistem *e-voting* berbasis Ethereum *smart contract* ditunjukkan oleh Gambar 3.2.



Gambar 3.2 Use Case Diagram

Gambar 3.2 menunjukkan *use case* dari sistem *e-voting* berbasis Ethereum *smart contract*. *Use case* tersebut menggambarkan interaksi pengguna terhadap sistem *e-voting* dan interaksi antara satu elemen dengan elemen lainnya. *Use case* aplikasi sistem *e-voting* memiliki 3 aktor, diantaranya *super admin*, *election authority* dan *voter*.

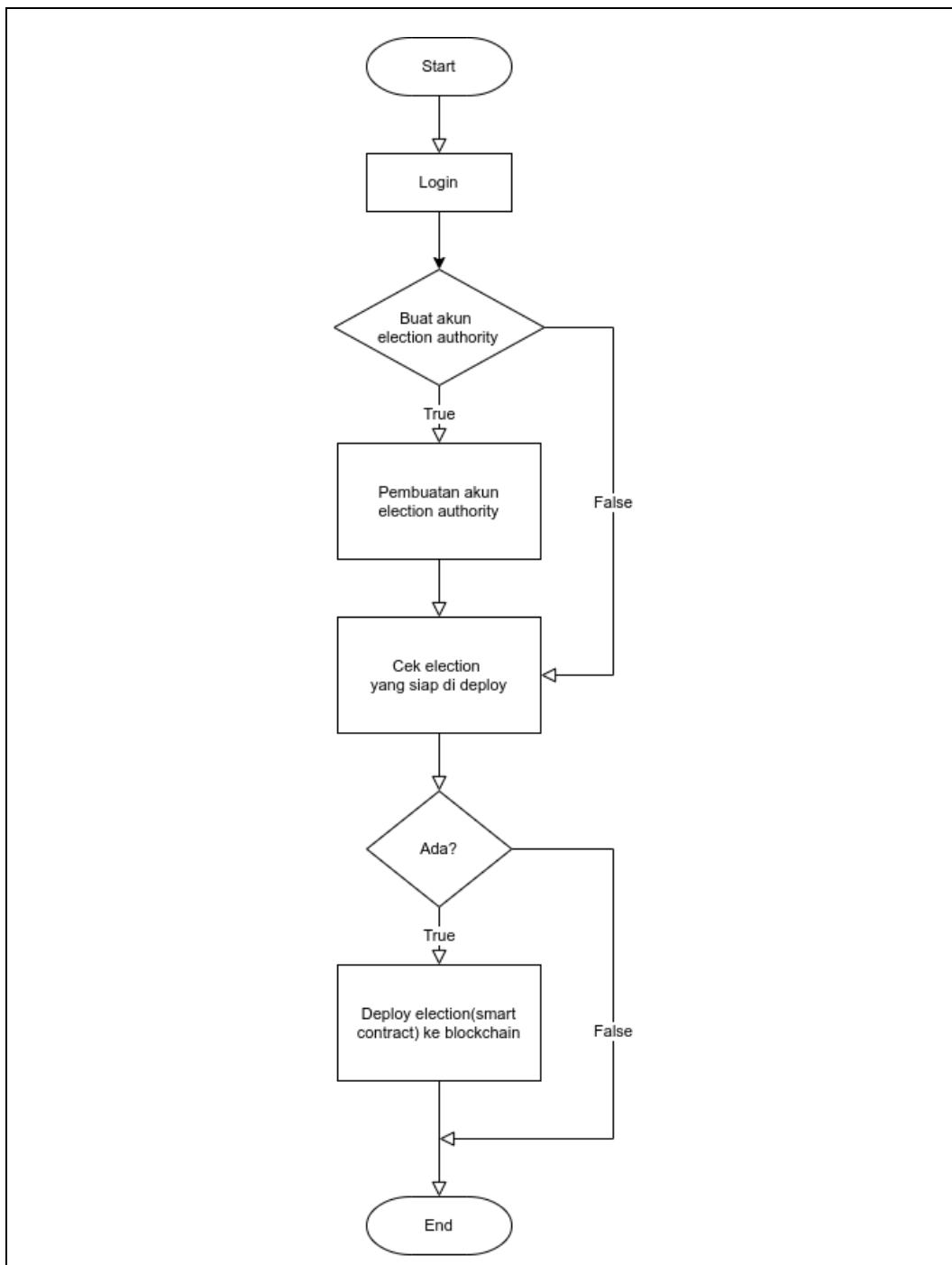
Masing-masing aktor memiliki peran dan akses yang berbeda-beda. *Super admin* dapat melakukan penambahan atau mendaftarkan *election authority* ke dalam sistem, melakukan proses *send ether* dan proses *deploy* atau aktivasi *election* sehingga dapat digunakan, serta melihat hasil *election*. *Election authority* dapat melakukan pembuatan *election* baru, menambahkan kandidat dalam *election* yang dibuat, menentukan awal dan berakhirnya *election*, serta melihat hasil *election*. *Voter* dapat melakukan pendaftaran dalam setiap *election* yang diikuti dan melakukan proses *vote* dalam *election* yang dipilih, serta melihat hasil *election*.

3.4.2 Rancangan Alur Sistem

Rancangan alur penggunaan aplikasi sistem *e-voting* berbasis Ethereum *smart contract* digunakan sebagai acuan untuk membuat antarmuka dari aplikasi.

3.4.2.1 Rancangan Alur Sistem (*Super Admin*)

Aplikasi sistem *e-voting* memiliki 3 jenis *user* yang memiliki peran dan akses yang berbeda-beda. Jenis *user* yang pertama adalah *super admin*. Rancangan alur sistem dengan *super admin* sebagai *user* dapat dilihat pada Gambar 3.3.

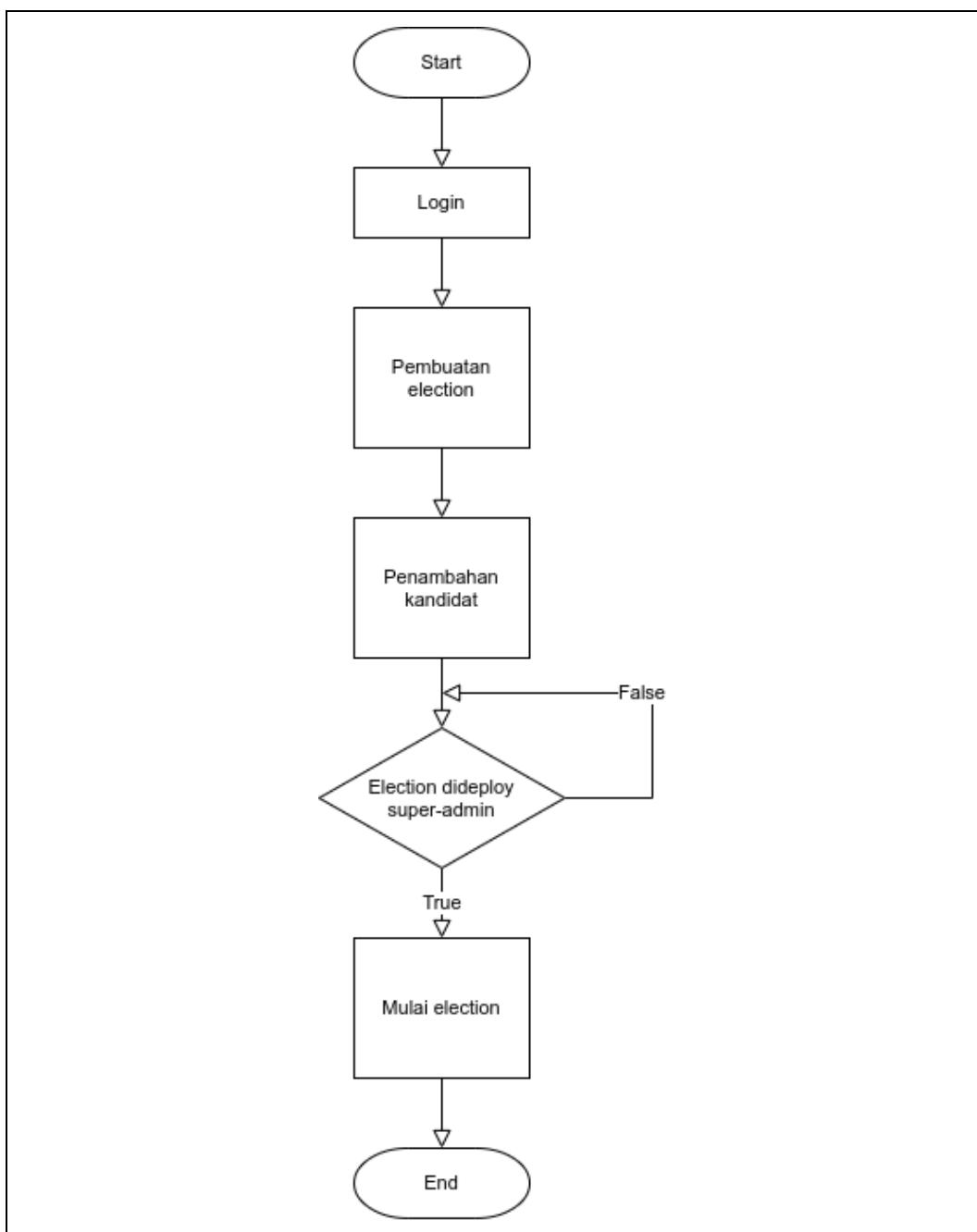


Gambar 3.3 Rancangan Alur Sistem (*Super Admin*)

Gambar 3.3 menggambarkan rancangan alur keseluruhan sistem dengan *super admin* sebagai *user*. Pertama-tama, *super admin* harus *login* terlebih dahulu ke dalam sistem. Setelah berhasil *login*, *super admin* dapat melakukan pembuatan akun bagi *election authority*. Apabila terdapat *election* baru yang siap di-deploy, maka *super admin* bertugas melakukan proses *deploy* ke *blockchain*.

3.4.2.2 Rancangan Alur Sistem (*Election Authority*)

Jenis *user* selanjutnya adalah *election authority*, yaitu *user* yang bertanggung jawab untuk membuat dan menangani satu atau lebih *election*. Rancangan alur sistem dengan *election authority* sebagai *user* dapat dilihat pada Gambar 3.4.

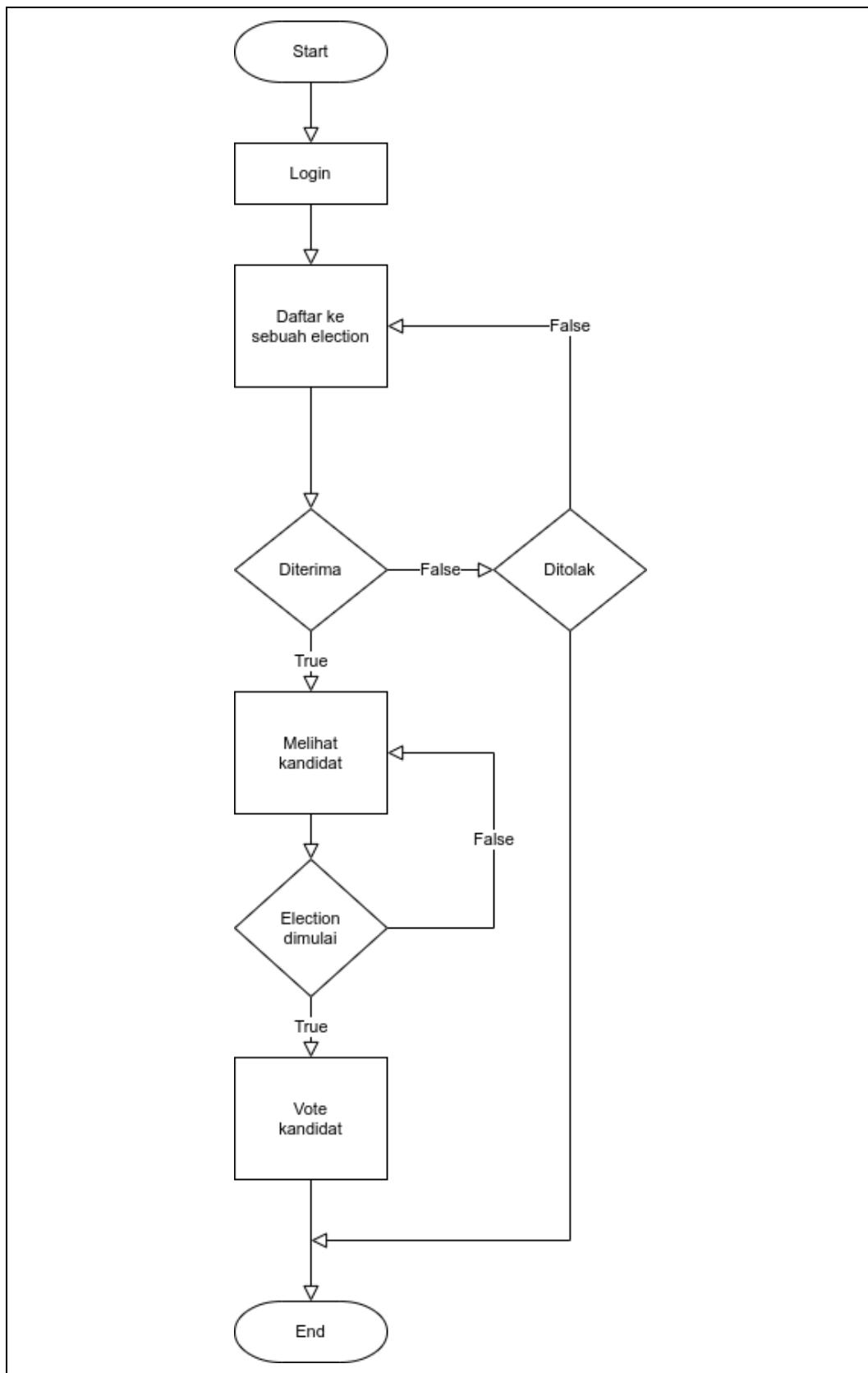


Gambar 3.4 Rancangan Alur Sistem (*Election Authority*)

Gambar 3.4 menggambarkan rancangan alur keseluruhan sistem dengan *election authority* sebagai *user*. Alur sistem diawali dengan proses *login* oleh *election authority* menggunakan akun yang telah didaftarkan atau dibuat oleh *super admin* sebelumnya. Setelah berhasil *login*, maka *election authority* dapat melakukan proses pembuatan *election* baru, diikuti dengan peambahan daftar kandidatnya. Setelah itu, apabila *election* yang dibuat telah di-*deploy* oleh *super admin*, maka *election authority* dapat memulai *election*. Namun, apabila *election* belum di-*deploy*, maka *election authority* harus menunggu sampai *election* berhasil di-*deploy* oleh *super admin* untuk dapat memulai *election*.

3.4.2.3 Rancangan Alur Sistem *Voter*

Jenis *user* yang terakhir adalah *voter*, yaitu *user* yang memiliki hak suara dan berhak memilih dalam satu atau lebih *election*. Rancangan alur sistem dengan *voter* sebagai *user* dapat dilihat pada Gambar 3.5.

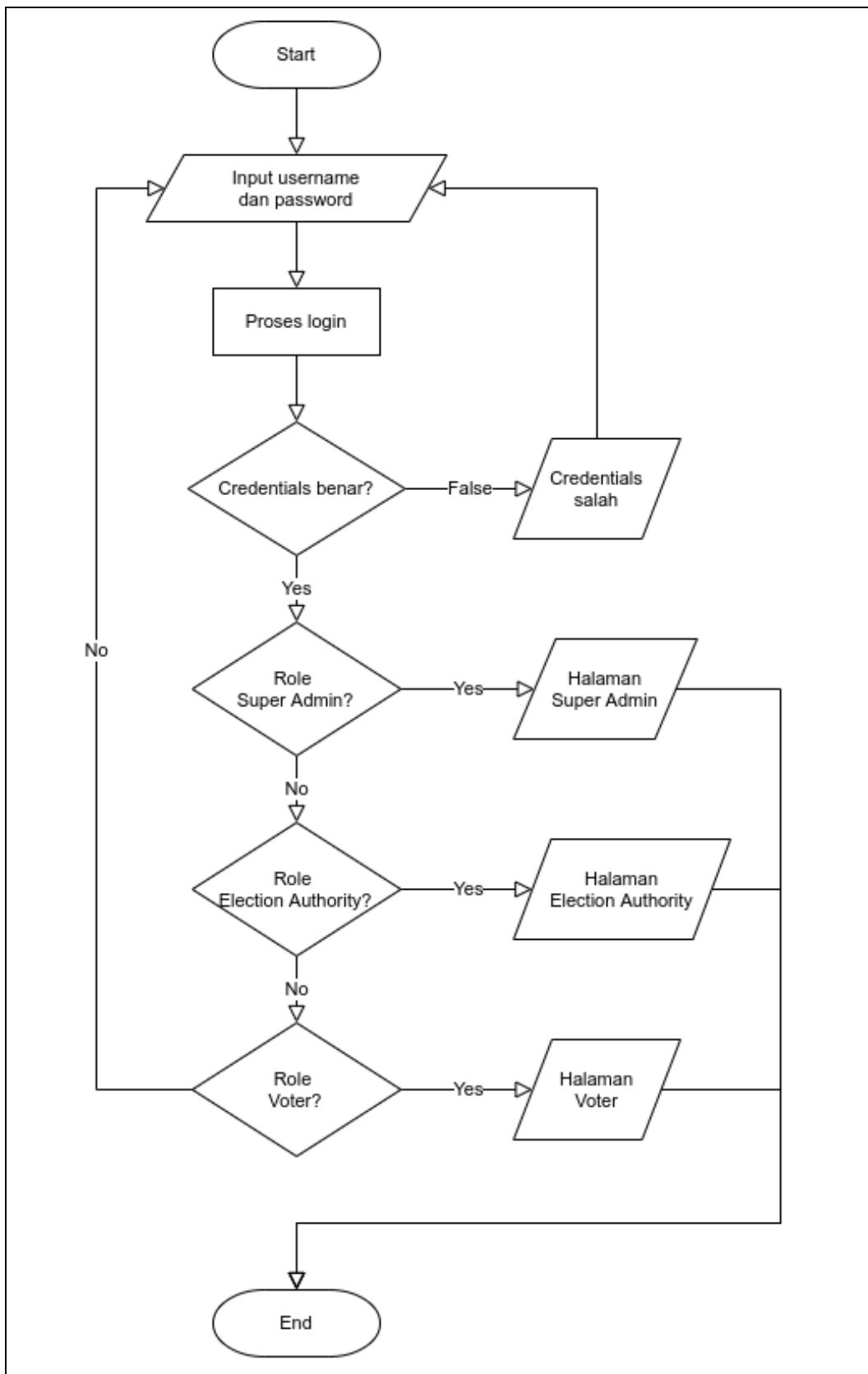


Gambar 3. 5 Rancangan Alur Sistem (*Voter*)

Gambar 3.5 menggambarkan rancangan alur keseluruhan sistem dengan *voter* sebagai *user*. Langkah pertama yang harus dilakukan oleh *voter* adalah *login* ke dalam sistem. Setelah berhasil *login*, maka *voter* dapat mendaftar ke dalam *election* yang akan diikuti. Pendaftaran *voter* dalam *election* harus terlebih dahulu disetujui untuk selanjutnya dapat melihat daftar kandidat dalam *election*, namun apabila pendaftaran ditolak, maka *voter* harus melakukan pendaftaran ulang sampai pendaftaran diterima untuk dapat mengikuti *election* tersebut. Proses pemungutan suara atau *vote* hanya dapat dilakukan oleh *voter* apabila *election* telah dimulai.

3.4.2.4 Rancangan Alur Proses *Login*

Proses *login* dapat dilakukan oleh 3 aktor yang terlibat dalam mengakses sistem. Rancangan alur untuk proses *login* pada sistem *e-voting* dapat dilihat pada Gambar 3.6.

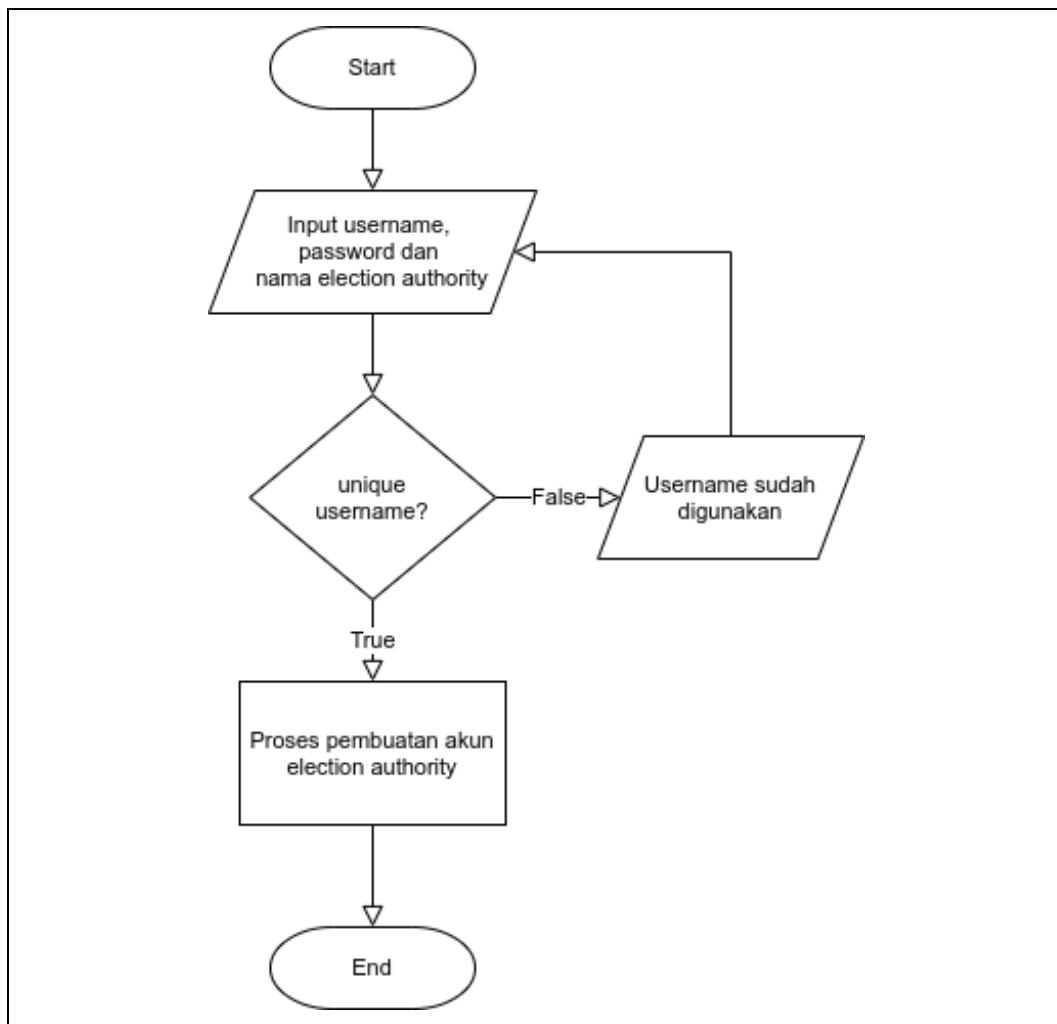


Gambar 3. 6 Rancangan Alur Proses Login

Gambar 3.6 memperlihatkan aktivitas untuk melakukan proses *login* untuk masuk atau mengakses sistem. Proses *login* juga menentukan hak akses dari masing-masing aktor yang terlibat. Setelah melakukan pengecekan data *login* yang dimasukkan maka dilakukan pengecekan terhadap *role* dari *user* yang akan *login*. Apabila dalam proses pengecekan tidak ada kesalahan, maka *user* akan dibawa menuju halaman sesuai dengan *user role*. Apabila terdapat kesalahan, maka *user* akan diminta memasukkan data *login* kembali.

3.4.2.5 Rancangan Alur Penambahan *Election Authority*

Rancangan alur untuk proses penambahan *election authority* pada sistem *e-voting* dapat dilihat pada Gambar 3.7.

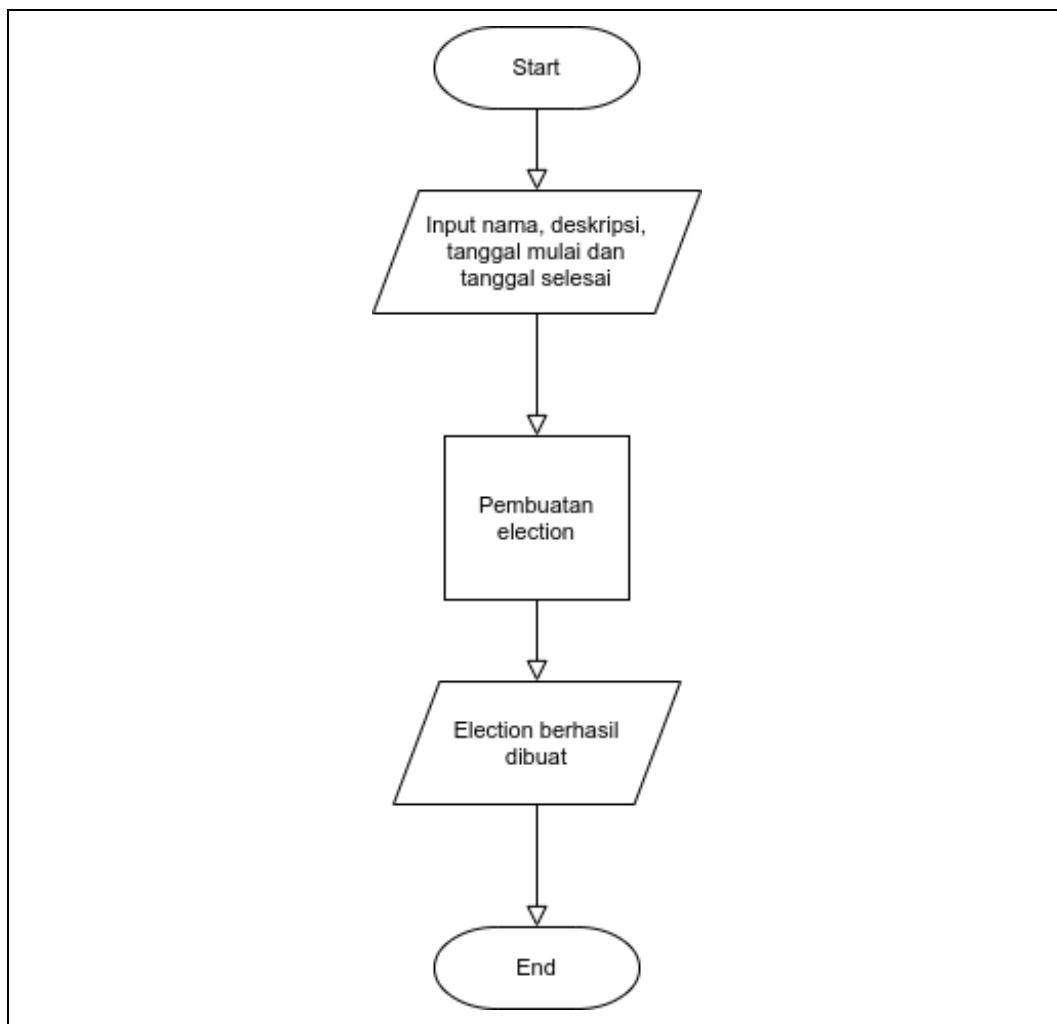


Gambar 3.7 Rancangan Alur Penambahan *Election Authority*

Gambar 3.7 memperlihatkan proses penambahan atau pembuatan *election authority* baru oleh *super admin*. Data yang dimasukkan untuk pembuatan *election authority* adalah nama, *username* dan *password*. *Username* harus bersifat *unique* atau belum pernah digunakan sebelumnya.

3.4.2.6 Rancangan Alur Pembuatan *Election*

Rancangan alur untuk proses pembuatan *election* baru pada sistem *e-voting* dapat dilihat pada Gambar 3.8.



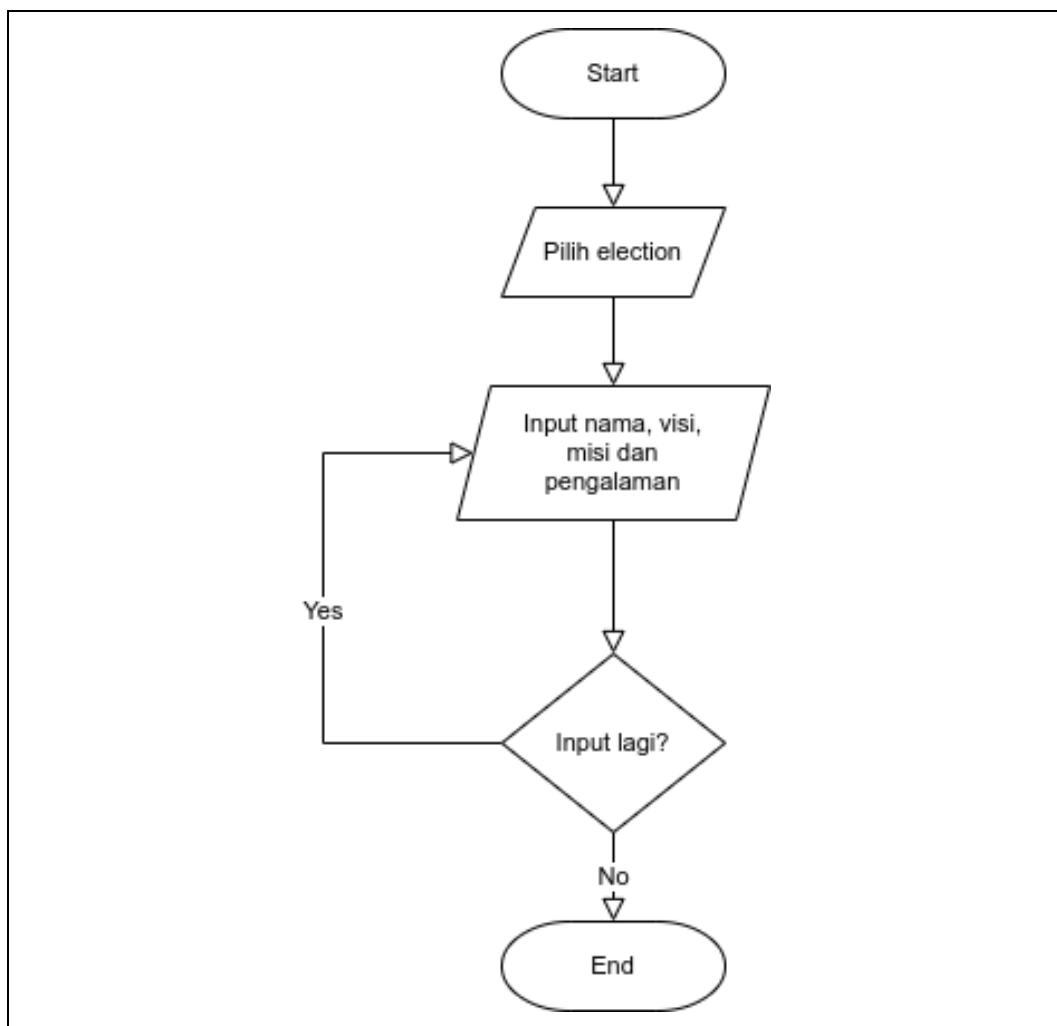
Gambar 3.8 Rancangan Alur Pembuatan *Election* Baru

Gambar 3.8 memperlihatkan aktivitas untuk melakukan penambahan atau pembuatan *election* baru. *Election authority* yang bertugas melakukan pembuatan

election baru meng-input-kan nama, deskripsi, serta tanggal berlangsungnya pemilihan. Setelah proses pembuatan *election*, maka *election* baru berhasil dibuat.

3.4.2.7 Rancangan Alur Tambah Kandidat

Rancangan alur untuk proses penambahan kandidat yang dapat dipilih dalam sebuah *election* pada sistem *e-voting* dapat dilihat pada Gambar 3.9.



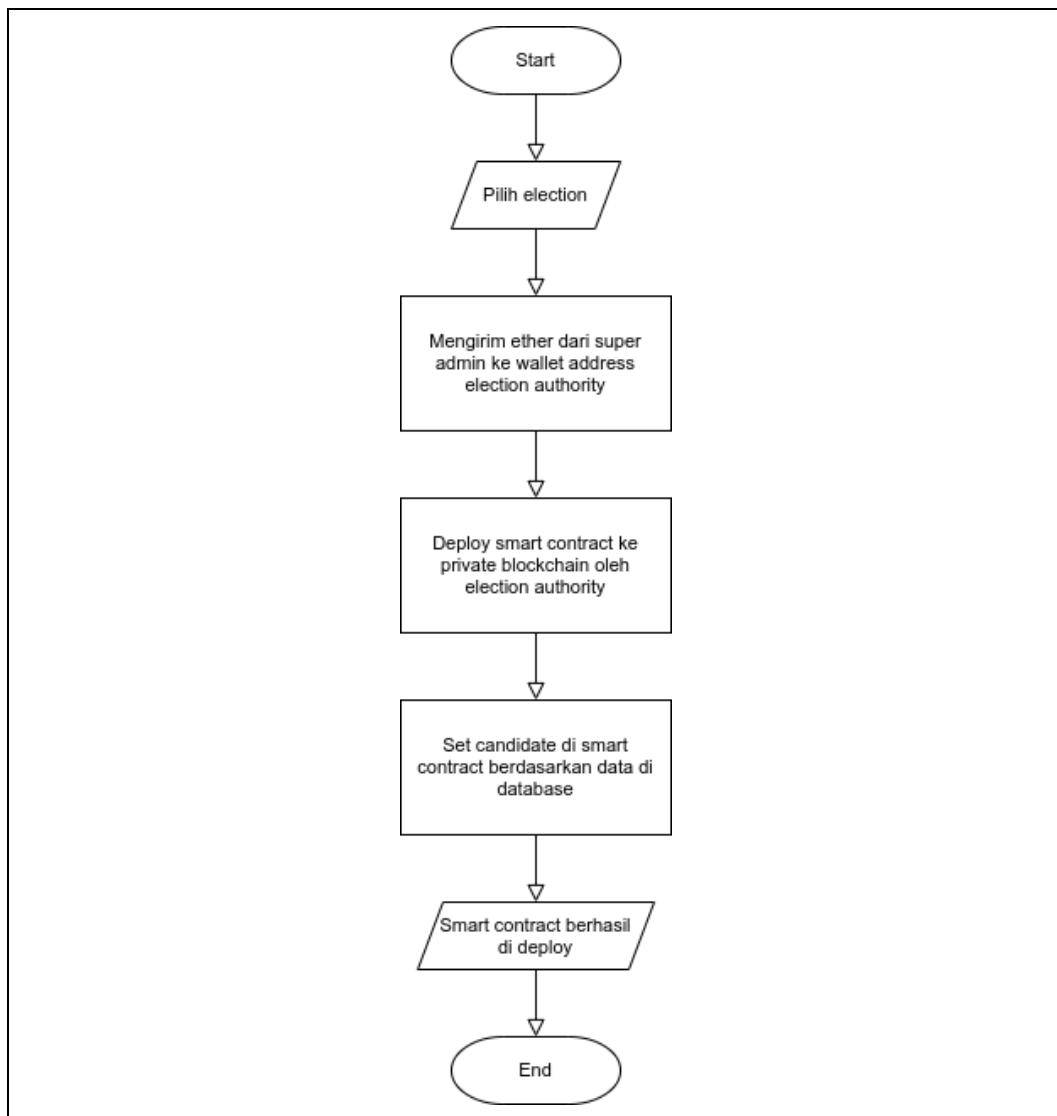
Gambar 3.9 Rancangan Alur Penambahan Kandidat

Gambar 3.9 memperlihatkan aktivitas untuk melakukan proses penambahan kandidat yang dapat dipilih dalam sebuah *election*. Sebelum melakukan penambahan kandidat, *election authority* harus terlebih dahulu membuat *election*. *Election* yang telah berhasil dibuat kemudian dapat dipilih untuk melakukan penambahan kandidat. *Election authority* memasukkan data nama, visi,

misi dan pengalaman dari kandidat. Proses penambahan kandidat dapat dilakukan secara berulang sebanyak jumlah kandidat dalam *election* yang dipilih.

3.4.2.8 Rancangan Alur Deploy Election

Rancangan alur untuk proses aktivasi *election* yang telah dibuat sehingga dapat digunakan atau *deploy election* dapat dilihat pada Gambar 3.10.



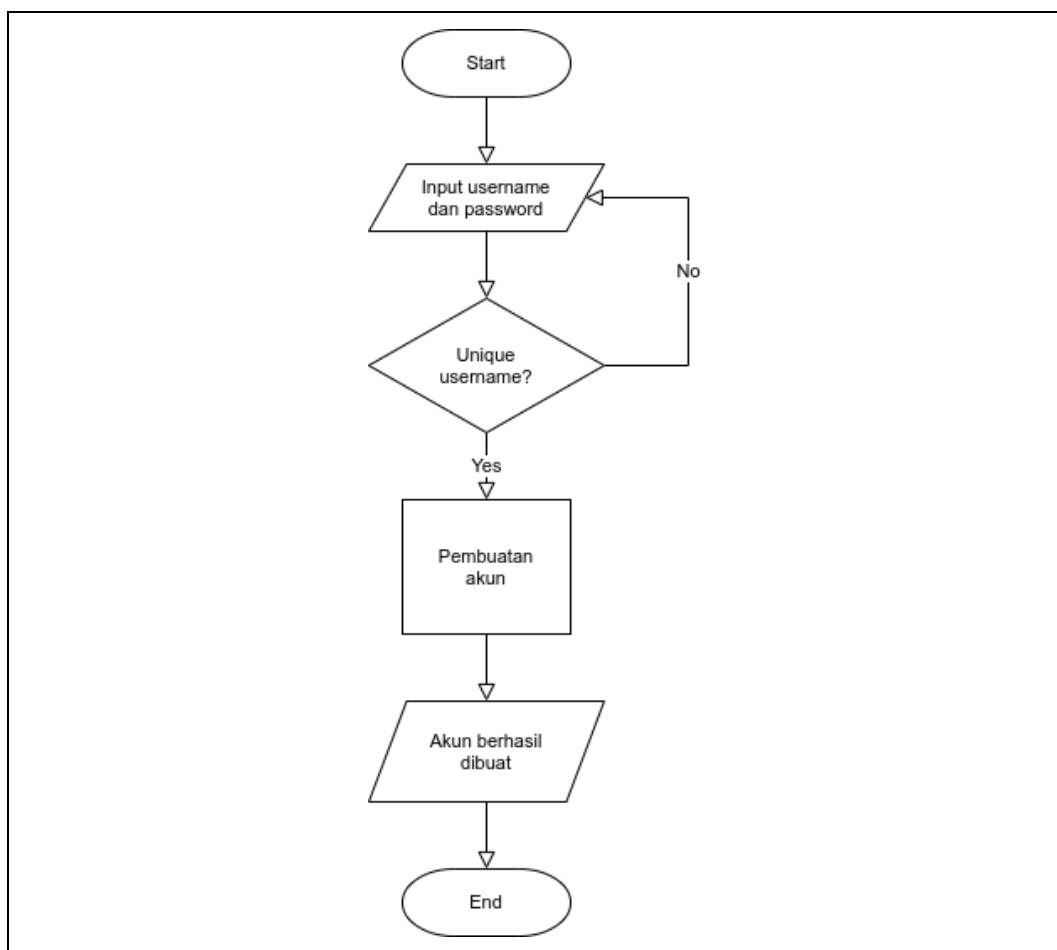
Gambar 3. 10 Rancangan Alur Deploy Election

Gambar 3.10 memperlihatkan aktivitas untuk melakukan proses mengaktifkan *election* yang telah dibuat oleh *election authority* sebelumnya atau proses membawa *smart contract* ke *blockchain* sehingga dapat digunakan. Setelah

election yang akan di-deploy dipilih, maka akan dilakukan proses pengiriman *ether* dari *super admin* ke *wallet address election authority*. Selanjutnya dilakukan proses *deploy smart contract* ke *private blockchain* dan penentuan kandidat di *smart contract* berdasarkan data yang tersimpan di basis data. Setelah semua proses selesai dilakukan maka *smart contract* telah berhasil di-deploy dan sudah dapat digunakan.

3.4.2.9 Rancangan Alur Registrasi *Voter*

Sebelum dapat melakukan proses *voting*, setiap *voter* harus terdaftar pada sistem terlebih dahulu. Rancangan alur untuk proses pendaftaran *voter* dapat dilihat pada Gambar 3.11.



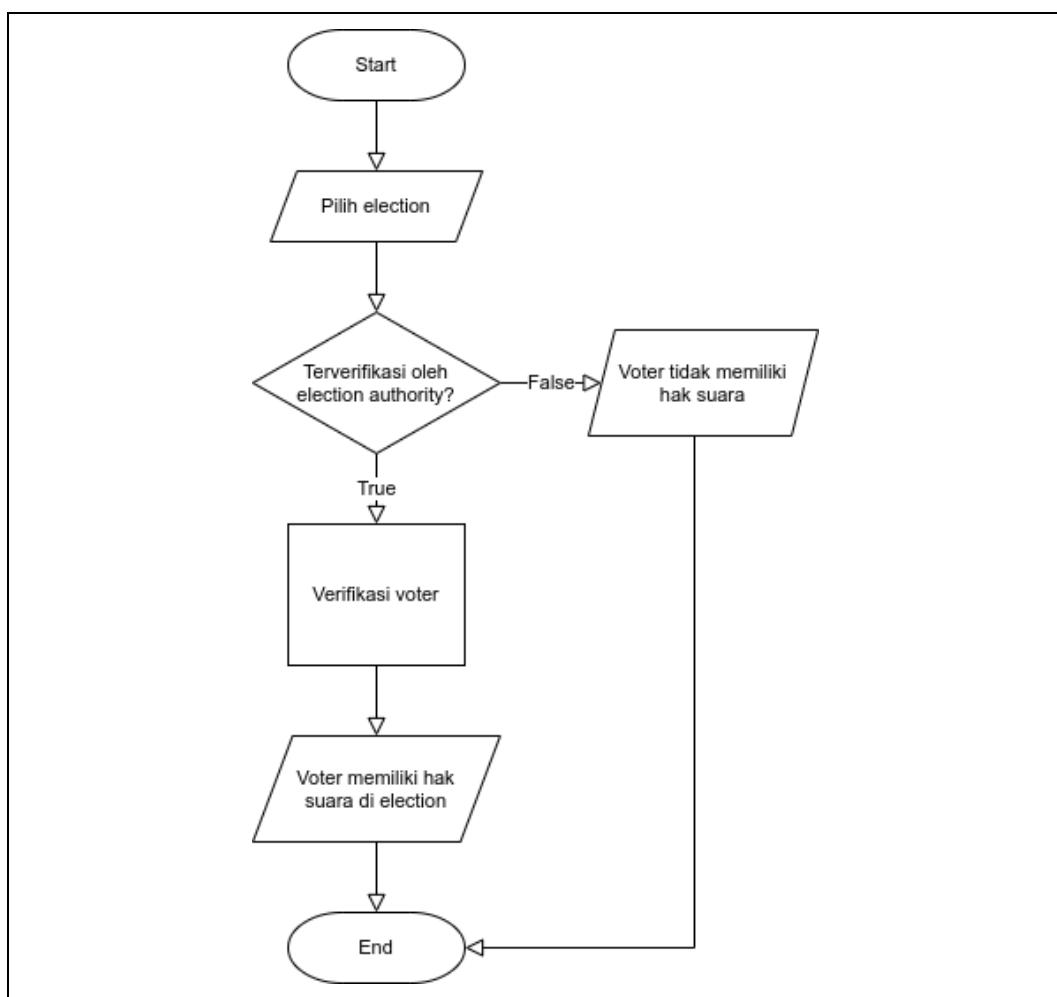
Gambar 3.11 Rancangan Alur Registrasi *Voter*

Gambar 3.11 memperlihatkan aktivitas untuk melakukan proses pendaftaran masing-masing *voter* dalam sistem *e-voting*. *Voter* memasukkan

username yang bersifat unik dan *password* untuk melakukan proses pembuatan akun yang terdaftar dalam sistem.

3.4.2.10 Rancangan Alur Pendaftaran *Voter* dalam *Election*

Setiap *voter* yang telah terdaftar di sistem akan melakukan voting dan juga harus terdaftar pada masing-masing *election* yang akan diikuti. Rancangan alur untuk proses pendaftaran *voter* dapat dilihat pada Gambar 3.12.



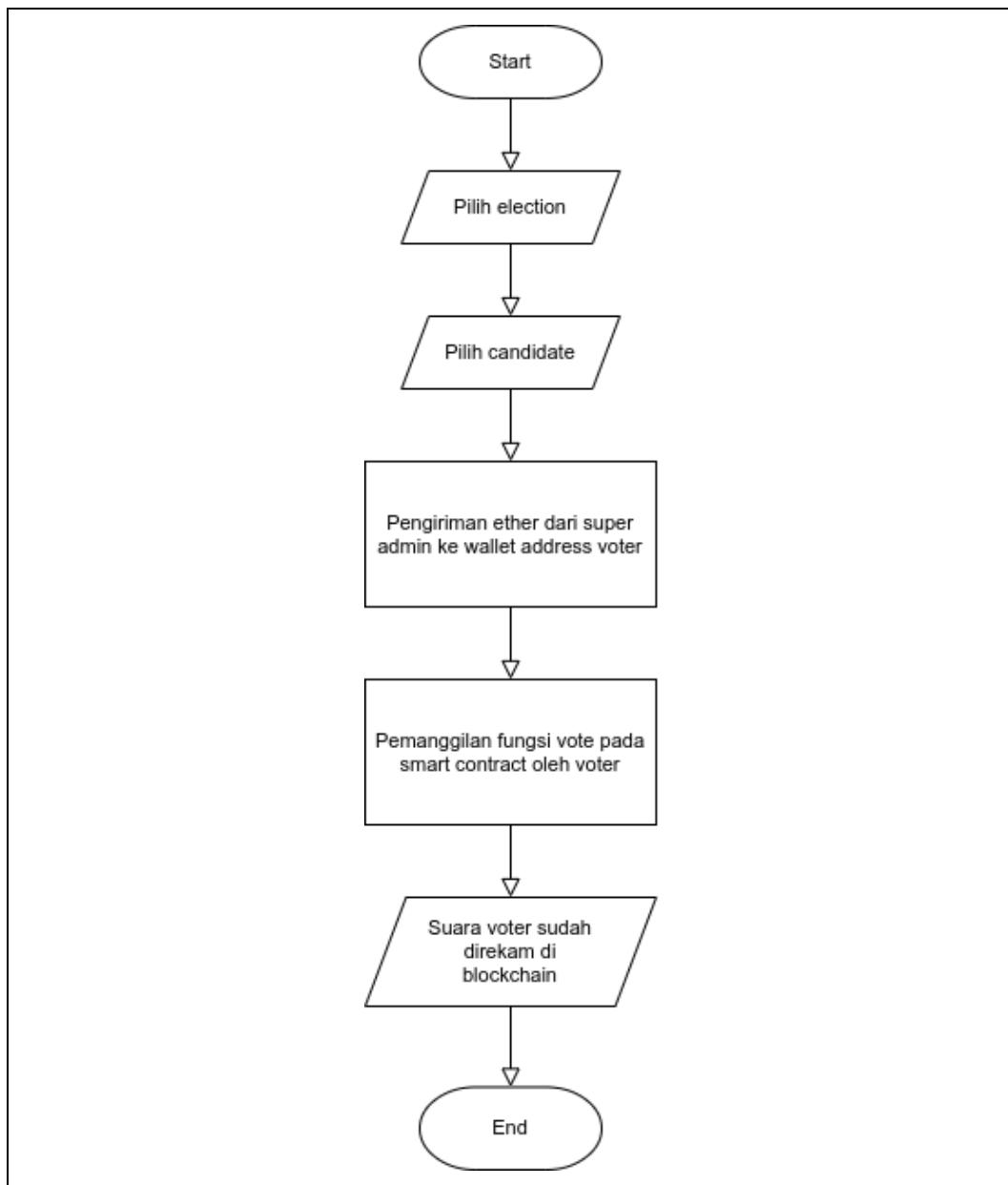
Gambar 3. 12 Rancangan Alur Pendaftaran *Voter* dalam *Election*

Gambar 3.12 memperlihatkan aktivitas untuk melakukan proses pendaftaran masing-masing *voter* dalam setiap *election* yang diikuti. Setelah melakukan pemilihan *election* yang diikuti, maka *election authority* akan

memverifikasi apakah *voter* tersebut memiliki hak suara atau tidak dalam *election* yang dipilih.

3.4.2.11 Rancangan Alur Proses *Vote*

Rancangan alur untuk pemberian suara atau *vote* oleh setiap *voter* dalam sebuah *election* pada sistem *e-voting* dapat dilihat pada Gambar 3.13.



Gambar 3. 13 Rancangan Alur Proses *Vote*

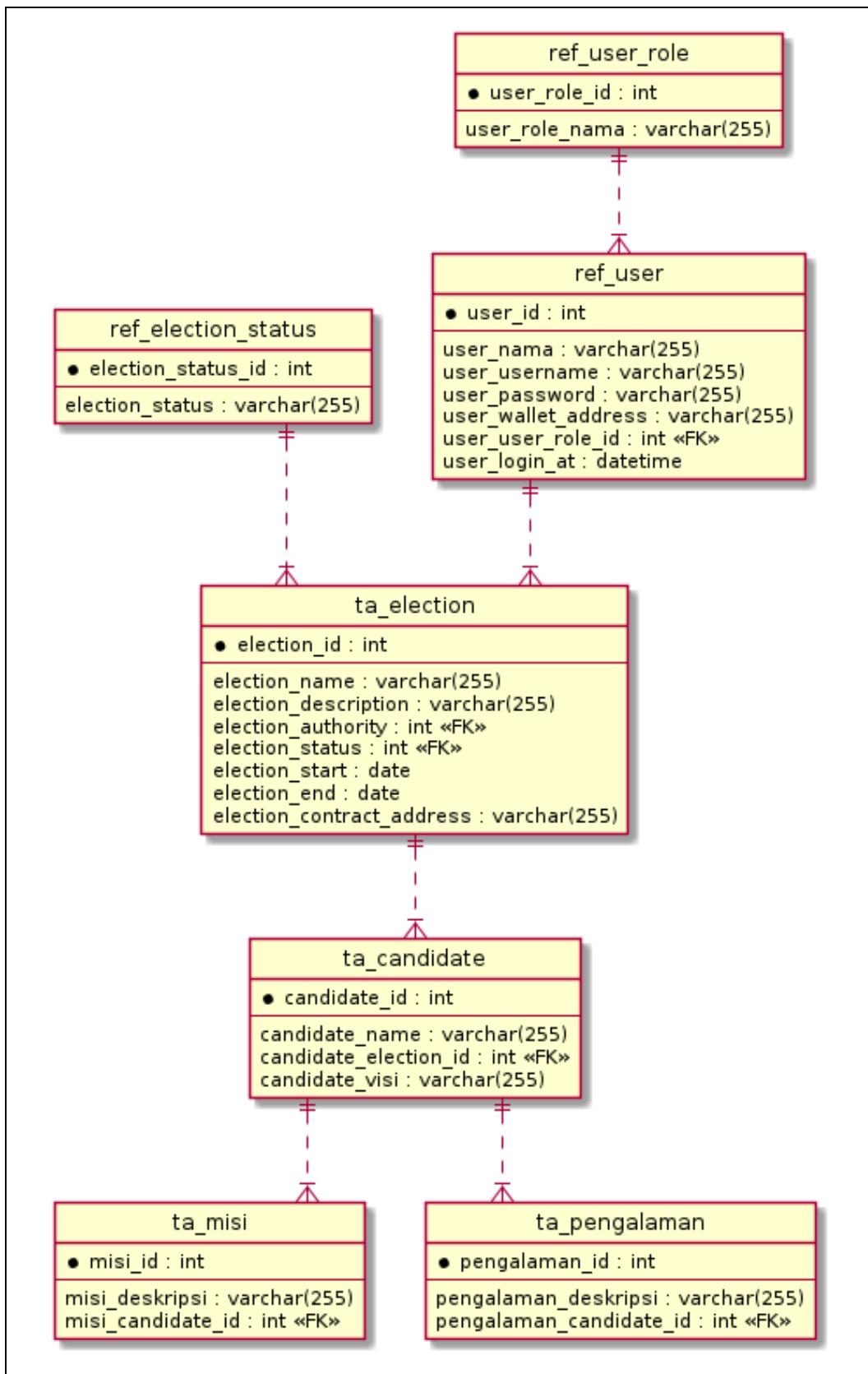
Gambar 3.13 memperlihatkan aktivitas untuk melakukan proses *vote* oleh masing-masing *voter*. *Voter* terlebih dahulu melakukan pemilihan *election* dan selanjutnya memilih kandidat yang akan di *vote*. Setelah itu dilakukan proses pengiriman *ether* dari *super admin* ke *wallet address voter* dilanjutkan dengan proses pemanggilan fungsi *vote* pada *smart contract* oleh *voter*. Apabila semua proses telah selesai dilakukan maka suara dari *voter* telah berhasil terekam di *blockchain*.

3.5 Rancangan Basis Data

Perancangan basis data merupakan suatu langkah dalam merancang basis data yang akan digunakan pada sistem. Perancangan basis data digunakan sebagai pendekatan struktur dengan menggunakan prosedur, teknik, alat serta bantuan dokumen untuk membantu dan memudahkan dalam proses perancangan.

3.5.1 Implementasi Basis Data

Implementasi basis data diambil berdasarkan perancangan basis data yang telah dibuat, secara fisik, implementasi basis data diimplementasikan menggunakan perangkat lunak MySQL. Struktur tabel basis data dapat dilihat pada Gambar 3.14.



Gambar 3.14 Implementasi Basis Data

Gambar 3.14 merupakan implementasi basis data dari *e-voting* berbasis Ethereum *smart contract*. Basis data tersebut digunakan sebagai media penyimpanan informasi data *user*, *election* dan *candidate*.

3.5.2 Struktur Data Tabel

Basis data merupakan kumpulan data yang disatukan di dalam kumpulan tabel. Struktur data dari masing-masing tabel dalam perancangan basis data dijelaskan sebagai berikut.

Tabel user diberi nama tabel `ref_user`. Tabel ini menyimpan seluruh data dari masing-masing user yang terdaftar dalam sistem *e-voting*. Struktur data tabel `ref_user` dapat dilihat pada Tabel 3.2.

Tabel 3.2 Struktur Data Tabel `ref_user`

Nama Field	Tipe Data	Panjang	Keterangan
<code>user_id</code>	int		Primary key
<code>user_nama</code>	varchar	255	
<code>user_username</code>	varchar	255	
<code>user_password</code>	varchar	255	
<code>user_wallet_address</code>	varchar	255	
<code>user_role_id</code>	Int		Foreign key
<code>user_login_at</code>	datetime		

Tabel 3.2 merupakan struktur data dari tabel `ref_user` yang memiliki `user_id` sebagai *primary key* beserta dengan *field* `user_nama`, `user_username`, `user_password`, `user_wallet_address`, `user_role_id`, dan `user_login_at`.

Tabel selanjutnya adalah tabel `ref_user_role`. Tabel ini menyimpan data dari *user role* yang ada dalam sistem *e-voting*. Struktur data tabel `ref_user_role` dapat dilihat pada Tabel 3.3.

Tabel 3.3 Struktur Data Tabel `ref_user_role`

Nama Field	Tipe Data	Panjang	Keterangan
<code>user_role_id</code>	int		Primary key
<code>user_role_nama</code>	varchar	255	

Tabel 3.3 merupakan struktur data dari tabel `ref_user_role` yang memiliki `user_role_id` sebagai *primary key* beserta dengan *field* `user_role_nama`.

Tabel selanjutnya adalah tabel `ta_candidate`. Tabel ini menyimpan seluruh data dari masing-masing kandidat yang terdaftar dalam sistem *e-voting*. Struktur data tabel `ta_candidate` dapat dilihat pada Tabel 3.4.

Tabel 3.4 Struktur Data Tabel `ta_candidate`

Nama Field	Tipe Data	Panjang	Keterangan
<code>candidate_id</code>	int		Primary key
<code>candidate_name</code>	varchar	255	
<code>candidate_election_id</code>	int	255	Foreign key
<code>candidate_visi</code>	varchar	255	

Tabel 3.4 merupakan struktur data dari tabel `ta_candidate` yang memiliki `candidate_id` sebagai *primary key* beserta dengan *field* `candidate_name`, `candidate_election_id`, dan `candidate_visi`.

Tabel selanjutnya adalah tabel `ta_misi`. Tabel ini menyimpan seluruh data misi dari masing-masing kandidat yang terdaftar dalam sistem *e-voting*. Struktur data tabel `ta_misi` dapat dilihat pada Tabel 3.5.

Tabel 3.5 Struktur Data Tabel `ta_misi`

Nama Field	Tipe Data	Panjang	Keterangan
<code>misi_id</code>	int		Primary key
<code>misi_deskripsi</code>	varchar	255	
<code>misi_candidate_id</code>	int	255	Foreign key

Tabel 3.5 merupakan struktur data dari tabel `ta_misi` yang memiliki `misi_id` sebagai *primary key* beserta dengan *field* `misi_deskripsi` dan `misi_candidate_id`.

Tabel selanjutnya adalah tabel `ta_pengalaman`. Tabel ini menyimpan seluruh data pengalaman dari masing-masing kandidat yang terdaftar dalam sistem *e-voting*. Struktur data tabel `ta_pengalaman` dapat dilihat pada Tabel 3.6.

Tabel 3. 6 Struktur Data Tabel ta_pengalaman

Nama Field	Tipe Data	Panjang	Keterangan
pengalaman_id	int		Primary key
pengalaman_deskripsi	varchar	255	
pengalaman_candidate_id	int	255	Foreign key

Tabel 3.6 merupakan struktur data dari tabel `ta_pengalaman` yang memiliki `pengalaman_id` sebagai *primary key* beserta dengan *field* `pengalaman_deskripsi` dan `pengalaman_candidate_id`.

Tabel selanjutnya adalah tabel `ta_election`. Tabel ini menyimpan seluruh data *election* yang terdaftar dalam sistem *e-voting*. Struktur data tabel `ta_election` dapat dilihat pada Tabel 3.7.

Tabel 3. 7 Struktur Data Tabel ta_election

Nama Field	Tipe Data	Panjang	Keterangan
election_id	int		Primary key
election_name	varchar	255	
election_description	varchar	255	
election_authority	int		Foreign key
election_status	int		Foreign key
election_start	date		
election_end	date		
election_contract_address	varchar	255	

Tabel 3.7 merupakan struktur data dari tabel `ta_election` yang memiliki `election_id` sebagai *primary key* beserta dengan *field* `election_name`, `election_description`, `election_authority`, `election_status`, `election_start`, `election_end`, dan `election_contract_address`

Tabel selanjutnya adalah tabel `ref_election_status`. Tabel ini menyimpan seluruh data *election status* yang terdaftar dalam sistem *e-voting*. Struktur data tabel `ref_election_status` dapat dilihat pada Tabel 3.8.

Tabel 3.8 Struktur Data Tabel ref_election_status

Nama Field	Tipe Data	Panjang	Keterangan
election_status_id	int		Primary key
election_status	varchar	255	

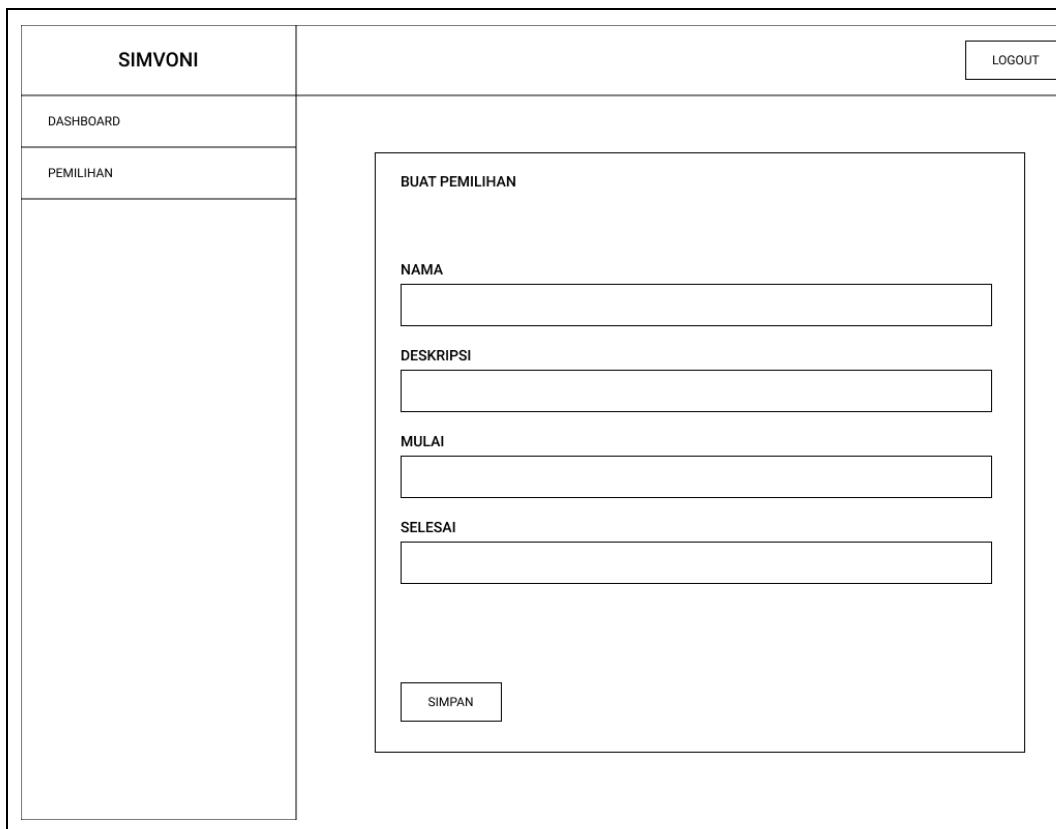
Tabel 3.8 merupakan struktur data dari tabel `ref_election_status` yang memiliki `election_status_id` sebagai *primary key* beserta dengan *field* `election_status`.

3.6 Bahasa Pemrograman

Implementasi Sistem Voting Elektronik berbasis Ethereum *smart contract* merupakan aplikasi berbasis *web*. Aplikasi ini dibangun dengan bahasa pemrograman Typescript untuk *frontend* dan *backend* dan Solidity untuk membuat *smart contract*. Adapun *database* yang digunakan adalah *database* MySQL.

3.7 Rancangan Antarmuka Aplikasi

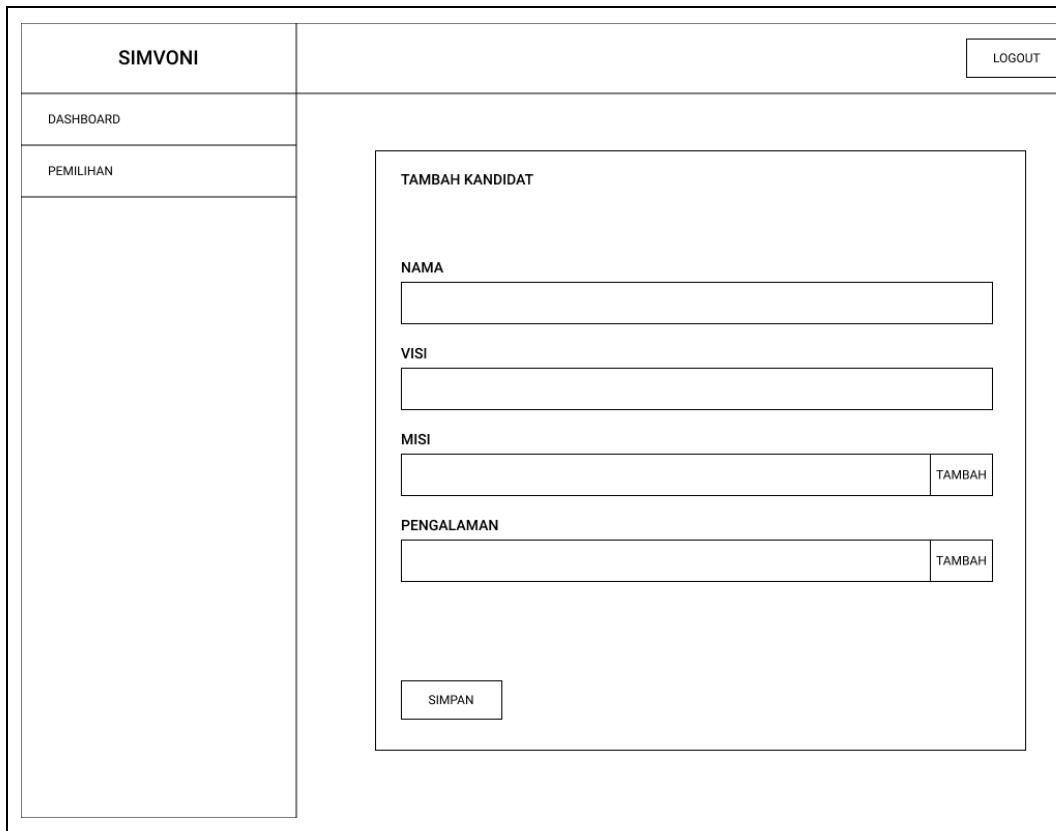
Rancangan antarmuka aplikasi adalah rancangan terhadap desain tampilan dari sistem voting elektronik. Keseluruhan tampilan dari sistem voting elektronik adalah berbasis *web*.



The diagram illustrates a user interface design for creating a poll. It features a sidebar on the left labeled 'SIMVONI' with 'DASHBOARD' and 'PEMILIHAN' options. The main area is titled 'BUAT PEMILIHAN' and contains four input fields: 'NAMA', 'DESKRIPSI', 'MULAI', and 'SELESAI'. A 'LOGOUT' button is located in the top right corner.

Gambar 3. 15 Rancangan Antarmuka Buat Pemilihan

Gambar 3.15 merupakan rancangan tampilan untuk membuat pemilihan pada sistem voting elektronik. Rancangan tampilan tersebut menampilkan sebuah *form* yang terdiri dari 4 *input* yaitu nama, deskripsi, mulai dan selesai serta sebuah tombol simpan.



Rancangan antarmuka (UI) untuk menambahkan kandidat dalam sistem voting elektronik. Tampilan ini terdiri dari dua bagian utama: sidebar kiri dan form tambah kandidat di sebelah kanan.

- Bagian Kiri (Sidebar):** Terdapat logo "SIMVONI" di bagian atas. Di bawahnya ada dua menu: "DASHBOARD" dan "PEMILIHAN".
- Bagian Kanan:** Judul "TAMBAH KANDIDAT" di bagian atas. Form yang memungkinkan pengisian empat data:
 - NAMA:** Input field.
 - VISI:** Input field.
 - MISI:** Input field dengan tombol "TAMBAH" di sebelahnya.
 - PENGALAMAN:** Input field dengan tombol "TAMBAH" di sebelahnya.
- Tombol Simpan:** Tombol "SIMPAN" di bagian bawah form.

Gambar 3.16 Rancangan Antarmuka Menambahkan Kandidat

Gambar 3.16 merupakan rancangan tampilan untuk menambahkan kandidat pada sistem voting elektronik. Rancangan tampilan tersebut menampilkan sebuah *form* dengan 4 *input* yaitu nama, visi, misi dan pengalaman. *Input* misi dan pengalaman digunakan untuk menyimpan data dari *multiple input* sehingga terdapat tombol tambah di sebelah kanan *input field*.

SIMVONI						LOGOUT
DASHBOARD		PEMILIHAN				
NO	NAMA	MULAI	SELESAI	ELECTION AUTHORITY	AKSI	
1	PEMIRA HMTI	20 AUG 2020	23 AUG 2020	HMTI	<input type="button" value="DEPLOY"/>	
2	PEMIRA SMFT	20 JUL 2020	23 JUL 2020	SMFT	<input type="button" value="DEPLOY"/>	

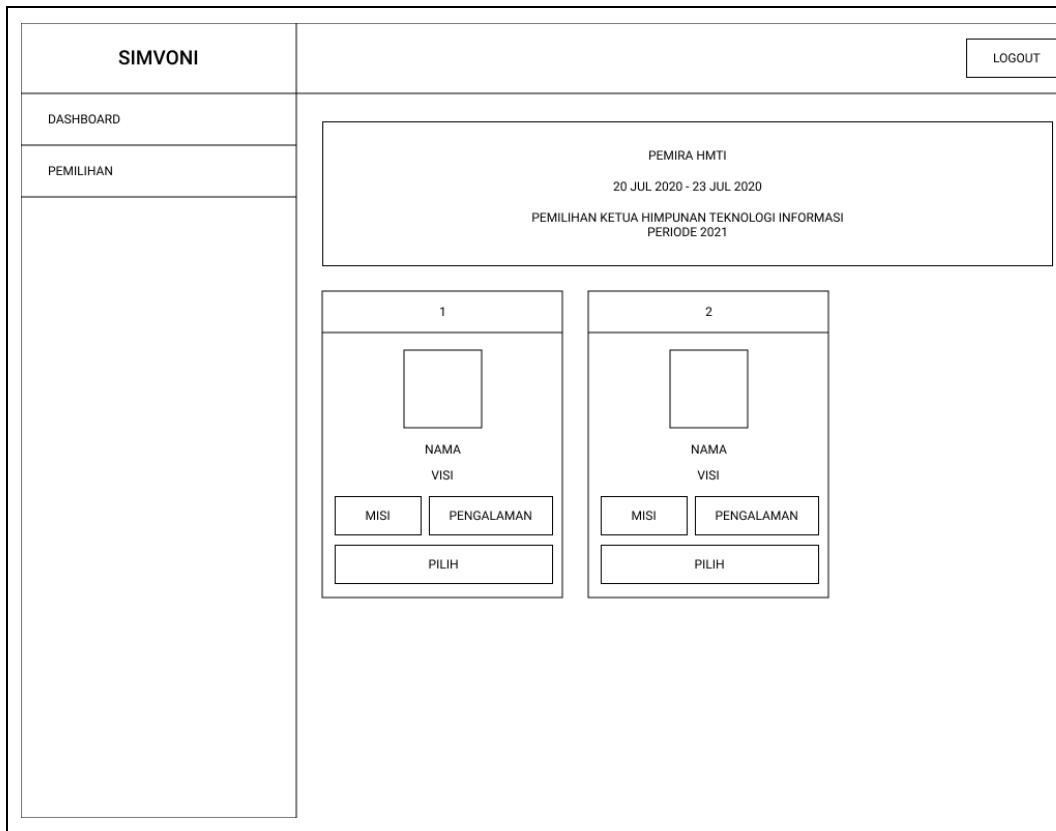
Gambar 3. 17 Rancangan Antarmuka Daftar Pemilihan Siap Deploy

Gambar 3.17 merupakan rancangan tampilan untuk melihat daftar dari pemilihan yang siap di-deploy ke jaringan Ethereum. *Field* yang ditampilkan adalah nama pemilihan, tanggal mulai, tanggal selesai dan *election authority* yang membuat pemilihan tersebut serta sebuah tombol *deploy*.

SIMVONI				LOGOUT
DASHBOARD				
PEMILIHAN				
NO	USERNAME	STATUS	AKSI	
1	voter-satu	pending	<input type="button" value="TERIMA"/>	<input type="button" value="TOLAK"/>
2	voter-dua	accepted	<input type="button" value="TERIMA"/>	<input type="button" value="TOLAK"/>

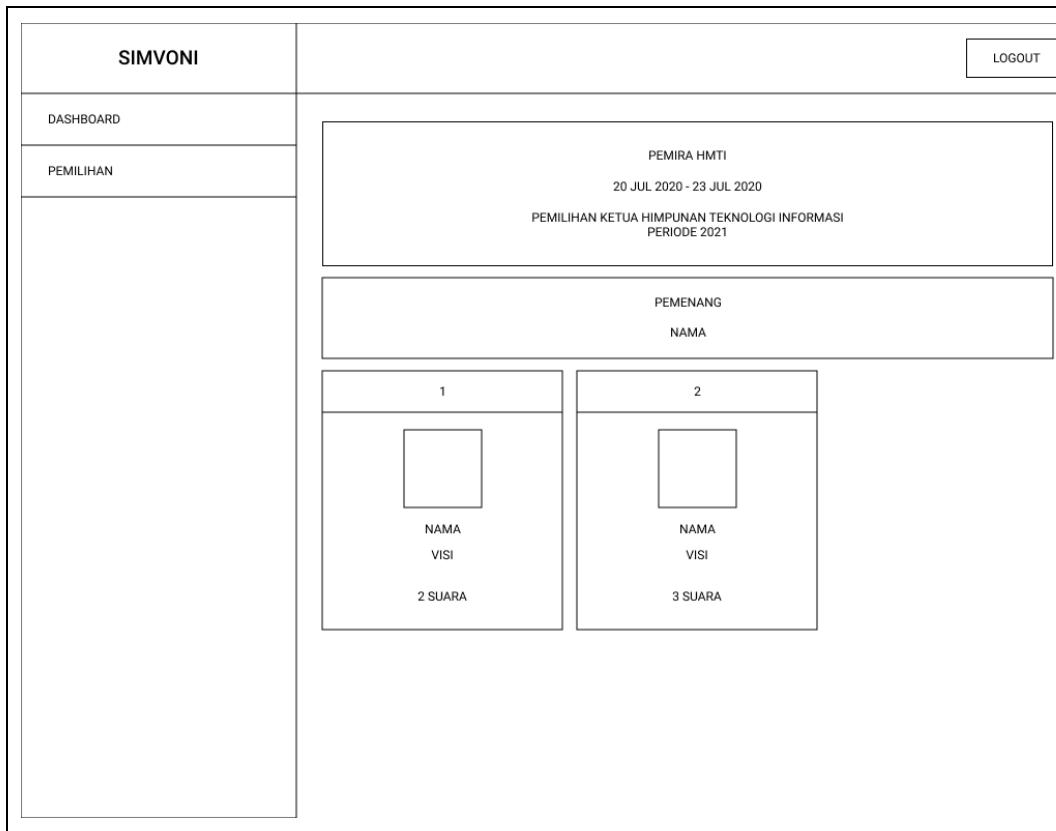
Gambar 3. 18 Rancangan Antarmuka Daftar Peserta Pemilihan

Gambar 3.18 merupakan rancangan tampilan untuk melihat daftar peserta suatu pemilihan pada sistem voting elektronik. *Field* yang ditampilkan adalah *username*, status keikutsertaan dan tombol untuk menerima atau menolak.



Gambar 3.19 Rancangan Antarmuka Detail Pemilihan

Gambar 3.19 merupakan rancangan tampilan untuk melihat detail dari sebuah pemilihan pada sistem voting elektronik. Informasi yang ditampilkan adalah nama pemilihan, tanggal mulai dan selesai dan deskripsi pemilihan. Selain itu terdapat daftar kandidat yang masing-masing menampilkan nama, visi, misi, pengalaman dan tombol untuk memilih kandidat tersebut.



Gambar 3. 20 Rancangan Antarmuka Hasil Pemilihan

Gambar 3.20 merupakan rancangan tampilan untuk melihat hasil dari sebuah pemilihan pada sistem voting elektronik. Informasi yang ditampilkan adalah nama pemilihan, tanggal mulai dan selesai, deskripsi pemilihan dan pemenang. Selain itu terdapat daftar kandidat yang masing-masing menampilkan nama, visi dan jumlah suara yang diperoleh.

3.8 Rancangan Spesifikasi API

Rancangan spesifikasi *endpoint* adalah rancangan terhadap *request* dan *response* yang diberikan oleh *backend* sistem *e-voting* melalui protokol *http*.

```
POST /election-authority/election
{
    "name": string,
    "description": string,
    "start": date,
    "end": date,
}
```

Kode Program 3. 1 Spesifikasi *Request* Pembuatan Pemilihan

Kode Program 3.1 merupakan spesifikasi *request* untuk membuat pemilihan. Metode yang digunakan adalah `POST` serta *field* yang dibutuhkan adalah `name` dan `description` dengan tipe data `string` serta `start` dan `end` dengan tipe data `date`.

```
{
    "message": string,
    "data": {
        "id": number,
        "name": string,
        "description": string,
        "start": date,
        "end": date,
        "status": string,
        "ea": string,
    }
}
```

Kode Program 3. 2 Spesifikasi Response Pembuatan Pemilihan

Kode Program 3.2 merupakan spesifikasi *response* dari hasil pembuatan pemilihan. *Field* yang ditampilkan adalah *field* yang sebelumnya digunakan untuk membuat pemilihan ditambah dengan `id` dengan tipe data `number`. `status` dan `ea` dengan tipe data `string`.

```
POST /election-authority/add-candidate/:electionId
{
    "name": string,
    "visi": string,
    "misi": [string],
    "pengalaman": [string]
}
```

Kode Program 3. 3 Spesifikasi Request Penambahan Kandidat

Kode Program 3.3 merupakan spesifikasi *request* untuk menambahkan kandidat pada sebuah pemilihan. Metode yang digunakan adalah `POST` serta *field* yang dibutuhkan adalah `name` dan `visi` dengan tipe data `string` serta `misi` dan `pengalaman` dengan tipe data `array of string`.

```
{
  "message": string,
  "data": {
    "id": number,
    "name": string,
    "visi": string,
    "candidateSlug": string,
    "election": string
  }
}
```

Kode Program 3.4 Spesifikasi *Response* Penambahan Kandidat

Kode Program 3.4 merupakan spesifikasi *response* dari hasil penambahan kandidat. *Field* yang ditampilkan adalah `id` dengan tipe data `number` serta `name`, `visi`, `candidateSlug` dan `election` dengan tipe data `string`. *Field* `candidateSlug` adalah nama kandidat yang diolah sehingga bisa digunakan di *smart contract*.

```
POST /super-admin/deploy-election/:electionId
```

Kode Program 3.5 Spesifikasi *Request Deployment* Pemilihan

Kode Program 3.5 merupakan spesifikasi *request* untuk melakukan *deployment* sebuah pemilihan ke jaringan Ethereum. Metode yang digunakan adalah `POST` dan tidak menggunakan *request body*.

```
{
  "message": string,
  "data": {
    "address": string
  }
}
```

Kode Program 3.6 Spesifikasi *Response Deployment* Pemilihan

Kode Program 3.6 merupakan spesifikasi *response* dari hasil *deployment* sebuah pemilihan ke jaringan Ethereum. *Field* yang ditampilkan adalah `address` dengan tipe data `string`.

```
GET /election-authority/election-participant/:electionId
```

Kode Program 3.7 Spesifikasi *Request Peserta Pemilihan*

Kode Program 3.7 merupakan spesifikasi *request* untuk mendapatkan semua peserta yang mengikuti sebuah pemilihan. Metode yang digunakan adalah `GET` tanpa menggunakan *request body*.

```
{
  "message": string,
  "data": {
    "electionId": number,
    "electionName": string,
    "participant": [
      {
        "participationId": number,
        "userId": number,
        "username": string,
        "status": string
      }
    ]
  }
}
```

Kode Program 3.8 Spesifikasi *Response* Peserta Pemilihan

Kode Program 3.8 merupakan spesifikasi *response* untuk menampilkan peserta sebuah pemilihan. *Field* yang ditampilkan adalah `electionId` dengan tipe data `number`. `electionName` dengan tipe data `string`. Dan untuk *field* `participant` adalah array dari suatu object. *Field* object tersebut adalah `participationId` dan `userId` dengan tipe data `number` serta `username` dan `status` dengan tipe data `string`.

```
GET /voter/election-detail/:electionId
```

Kode Program 3.9 Spesifikasi *Request Detail* Pemilihan

Kode Program 3.9 merupakan spesifikasi *request* untuk mendapatkan informasi detail sebuah pemilihan. Metode yang digunakan adalah `GET` tanpa menggunakan *request body*.

```
{
  "message": string,
  "data": {

    "id": number,
    "name": string,
    "description": string,
    "start": date,
    "end": date,
    "status": string,
    "ea": string,
    "participation_status": string,
    "candidates": [
      {
        "id": number,
        "name": string,
        "party": string
      }
    ]
  }
}
```

```

        "id": number,
        "visi": string,
        "misi": [string],
        "pengalaman": [string]
    }
]
}
}

```

Kode Program 3. 10 Spesifikasi *Response Detail Pemilihan*

Kode Program 3.10 merupakan spesifikasi *response* untuk menampilkan informasi detail sebuah pemilihan. *Field* yang ditampilkan adalah `id` dengan tipe data `number`. `name`, `description`, `status`, `ea` dan `participation_status` dengan tipe data `string`, `start` dan `end` dengan tipe data `date`. Dan untuk *field* `candidates` adalah array dari suatu object. Spesifikasi object tersebut adalah `id` dengan tipe data `number`, `visi` dengan tipe data `string` serta `misi` dan `pengalaman` dengan tipe data `array of string`.

```
GET /voter/ended-election-detail/:electionId
```

Kode Program 3. 11 Spesifikasi *Request Hasil Pemilihan*

Kode Program 3.11 merupakan spesifikasi *request* untuk mendapatkan informasi hasil dari sebuah pemilihan yang telah selesai. Metode yang digunakan adalah `GET` tanpa menggunakan *request body*.

```

{
  "message": string,
  "data": {

    "id": number,
    "name": string,
    "description": string,
    "start": date,
    "end": date,
    "status": string,
    "ea": string,
    "winner": string,
    "candidates": [
      {
        "id": number,
        "visi": string,
        "vote_count": number,
        "misi": [string],
        "pengalaman": [string]
      }
    ]
  }
}

```

```

        }
    ]
}
}
```

Kode Program 3. 12 Spesifikasi *Response* Hasil Pemilihan

Kode Program 3.12 merupakan spesifikasi *response* untuk menampilkan hasil dari pemilihan yang telah selesai. *Field* yang ditampilkan adalah `id` dengan tipe data `number`. `name`, `description`, `status`, `ea` dan `winner` dengan tipe data `string`, `start` dan `end` dengan tipe data `date`. Dan untuk *field* `candidates` adalah array dari suatu object. Spesifikasi object tersebut adalah `id` dengan tipe data `number`. `visi` dengan tipe data `string`. `vote_count` dengan tipe data `number` serta `misi` dan pengalaman dengan tipe data `array of string`.

```
POST /election-authority/start-election/:electionId
```

Kode Program 3. 13 Spesifikasi *Request* Memulai Pemilihan

Kode Program 3.13 merupakan spesifikasi *request* untuk memulai sebuah pemilihan. Metode yang digunakan adalah `POST` tanpa *request body*. Parameter yang digunakan adalah `id` dari pemilihan yang akan dimulai.

```
{
  "message": string,
  "data": {
    "election": string
  }
}
```

Kode Program 3. 14 Spesifikasi *Response* Memulai Pemilihan

Kode Program 3.14 merupakan spesifikasi *response* yang ditampilkan untuk memulai sebuah pemilihan. *Field* yang ditampilkan adalah `election` dengan tipe data `string`.

```
POST /voter/join/:electionId
```

Kode Program 3. 15 Spesifikasi *Request* Mengikuti Pemilihan

Kode Program 3.15 merupakan spesifikasi *request* untuk mengikuti sebuah pemilihan. Metode yang digunakan adalah `POST` tanpa *request body*. Parameter yang digunakan adalah `id` dari pemilihan yang akan diikuti.

```
{
  "message": string,
  "data": {
    "election": string
  }
}
```

Kode Program 3.16 Spesifikasi *Response* Mengikuti Pemilihan

Kode Program 3.16 merupakan spesifikasi *response* yang ditampilkan untuk mengikuti sebuah pemilihan. *Field* yang ditampilkan adalah *election* dengan tipe data *string*.

```
POST /voter/vote
{
  "election_id": number,
  "candidate_id": number
}
```

Kode Program 3.17 Spesifikasi *Request* untuk Memilih Kandidat

Kode Program 3.17 merupakan spesifikasi *request* untuk memilih kandidat dalam sebuah pemilihan. Metode yang digunakan adalah *POST* dan *field* yang dibutuhkan adalah *election_id* dan *candidate_id* dengan tipe data *number*.

```
{
  "message": string,
  "data": {
    "election": string
  }
}
```

Kode Program 3.18 Spesifikasi *Response* untuk Memilih Kandidat

Kode Program 3.18 merupakan spesifikasi *response* yang ditampilkan untuk memilih kandidat dalam sebuah pemilihan. *Field* yang ditampilkan adalah *election* dengan tipe data *string*.

```
POST /election-authority/end-election/:electionId
```

Kode Program 3.19 Spesifikasi *Request* untuk Menghentikan Pemilihan

Kode Program 3.19 merupakan spesifikasi *request* untuk menghentikan sebuah pemilihan. Metode yang digunakan adalah *POST* tanpa *request body*. Parameter yang digunakan adalah *id* dari pemilihan yang akan dihentikan.

```
{
  "message": string,
```

```
    "data": {  
        "election": string  
    }  
}
```

Kode Program 3. 20 Spesifikasi *Response* untuk Menghentikan Pemilihan

Kode Program 3.20 merupakan spesifikasi *response* yang ditampilkan untuk menghentikan sebuah pemilihan. *Field* yang ditampilkan adalah `election` dengan tipe data `string`.

BAB IV

PENGUJIAN DAN ANALISA HASIL

BAB IV menjelaskan hasil dari penelitian ini yaitu berupa hasil pengembangan aplikasi, pengujian aplikasi, analisa aplikasi serta implementasinya ke infrastruktur *production* yaitu *Virtual Machine* Google Cloud.

4.1 Infrastruktur Google Cloud

Google Cloud adalah *cloud provider* berskala internasional yang sudah terbukti menyediakan layanan yang handal untuk keperluan industri di bidang teknologi informasi. Layanan yang ditawarkan oleh Google Cloud sangat beragam diantaranya yang digunakan dalam pengembangan aplikasi ini adalah *Virtual Machine* dengan spesifikasi yang fleksibel sesuai kebutuhan dan biaya sehingga semua sumber daya bisa digunakan secara efisien. Google Cloud juga telah menyelesaikan pembangunan *data center* untuk *region* Jakarta sehingga bisa meminimalisir latensi.

4.1.1 Persiapan Infrastruktur

Virtual Machine yang dibutuhkan untuk membuat jaringan *private blockchain* adalah dua buah *virtual machine* sesuai dengan Gambaran Umum aplikasi.

Name	Type	Targets	Filters	Protocols / ports	Action	Priority	Network	Logs
all-out	Egress	all-out	IP ranges: 0.0.0.0/0	all	Allow	1000	default	Off
trobos-aja-anjing-eks	Egress	trobos-aja-anjing-eks	IP ranges: 0.0.0.0/0	all	Allow	1000	default	Off
all-in	Ingress	all-in	IP ranges: 0.0.0.0/0	all	Allow	1000	default	Off
allow-9090	Ingress	http-server	IP ranges: 0.0.0.0/0	tcp:9090	Allow	1000	default	Off
default-allow-http	Ingress	http-server	IP ranges: 0.0.0.0/0	tcp:80	Allow	1000	default	Off
default-allow-https	Ingress	https-server	IP ranges: 0.0.0.0/0	tcp:443	Allow	1000	default	Off
mysql	Ingress	mysql	IP ranges: 0.0.0.0/0	tcp:3306	Allow	1000	default	Off
parsec	Ingress	parsec	IP ranges: 0.0.0.0/0	udp:8000-8011	Allow	1000	default	Off
trobos-aja-anjing	Ingress	trobos-aja-anjing	IP ranges: 0.0.0.0/0	all	Allow	1000	default	Off
default-allow-icmp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	icmp	Allow	65534	default	Off
default-allow-internal	Ingress	Apply to all	IP ranges: 10.128.0.0/9	tcp:0-65535 udp:0-65535 icmp	Allow	65534	default	Off
default-allow-rdp	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:3389	Allow	65534	default	Off
default-allow-ssh	Ingress	Apply to all	IP ranges: 0.0.0.0/0	tcp:22	Allow	65534	default	Off

Gambar 4. 1 Halaman Google Cloud untuk Mengelola *Firewall Rule*

Gambar 4.1 menunjukkan halaman Google Cloud untuk mengelola *firewall rule*. *Firewall rule* adalah informasi untuk melakukan *filter* terhadap *request* yang diterima *virtual machine* pada suatu *port*. Umumnya *firewall rule* diatur dalam *virtual machine* dengan perintah *ufw*. Namun untuk *virtual machine* pada Google Cloud terdapat halaman terpisah untuk mengelola *firewall rule* yang digunakan oleh *virtual machine*.

Direction of traffic ?

- Ingress
- Egress

Action on match ?

- Allow
- Deny

Targets

Specified target tags

Target tags *

all-in X

Source filter

IPv4 ranges

Source IPv4 ranges *

0.0.0.0/0 X for example, 0.0.0.0/0, 192.168.2.0/24

Second source filter

None

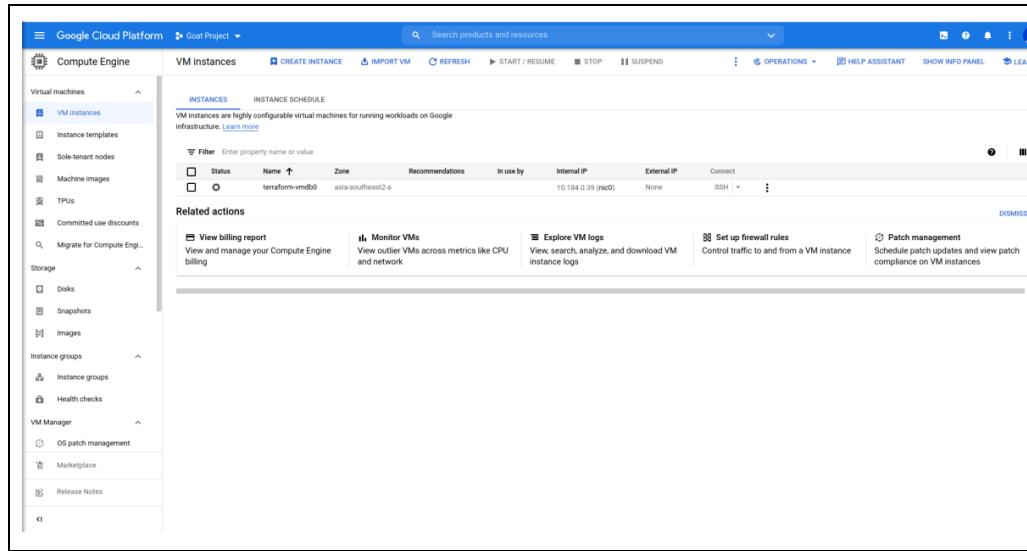
Protocols and ports ?

- Allow all
- Specified protocols and ports

Gambar 4.2 Firewall Rule yang Diperlukan

Gambar 4.2 menunjukkan halaman pembuatan *firewall rule* untuk *virtual machine* yang akan dibuat. Hal yang perlu diperhatikan adalah *Direction of traffic* adalah *Ingress* yang artinya *traffic* berasal dari luar menuju ke arah *virtual machine*. *Action on match* adalah *Allow* yang artinya memperbolehkan *traffic* untuk diteruskan apabila sesuai dengan *firewall rule*. *Target tags* bisa disesuaikan dengan ketentuan penamaan masing-masing, untuk studi kasus pada penelitian ini menggunakan nama *all-in*. *Source IPv4 ranges* adalah 0.0.0.0/0 yang artinya *IP* apa saja boleh mengakses *virtual machine*. Sedangkan *Protocols and ports* adalah

Allow all yang artinya memberbolehkan semua protokol komunikasi diantaranya *http* dan *udp*.



Gambar 4.3 Halaman Google Cloud untuk Mengelola VM

Gambar 4.3 menunjukkan halaman Google Cloud untuk mengelola Compute Engine. Compute Engine adalah nama layanan Google Cloud yang memberikan *virtual machine* yang dapat bekerja seperti layaknya *server*.

Machine family

GENERAL-PURPOSE **MEMORY-OPTIMIZED**

Machine types for common workloads, optimized for cost and flexibility

Series — E2

CPU platform selection based on availability

Machine type — e2-micro (2 vCPU, 1 GB memory)

	vCPU	Memory
	1 shared core	1 GB

▼ CPU PLATFORM AND GPU

Display device

Enable to use screen capturing and recording tools.

Enable display device

Confidential VM service ?

Enable the Confidential Computing service on this VM instance.

Container ?

Deploy a container image to this VM instance

DEPLOY CONTAINER

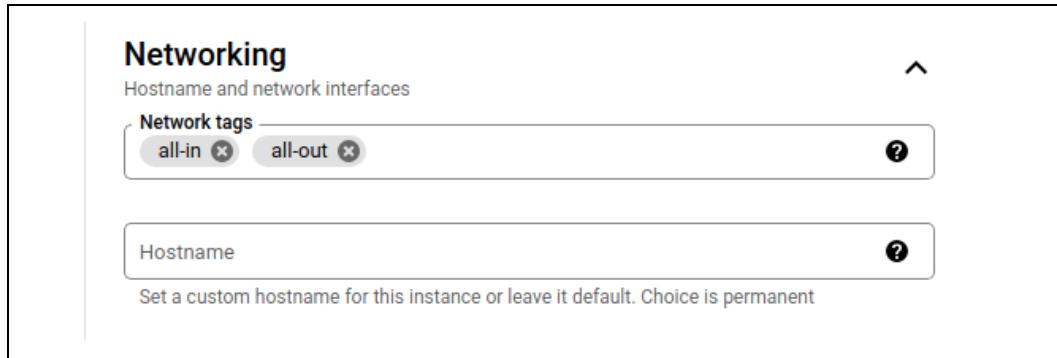
Boot disk ?

Type	New balanced persistent disk
Size	10 GB
Image	 Ubuntu 18.04 LTS

Gambar 4.4 Spesifikasi *Virtual Machine*

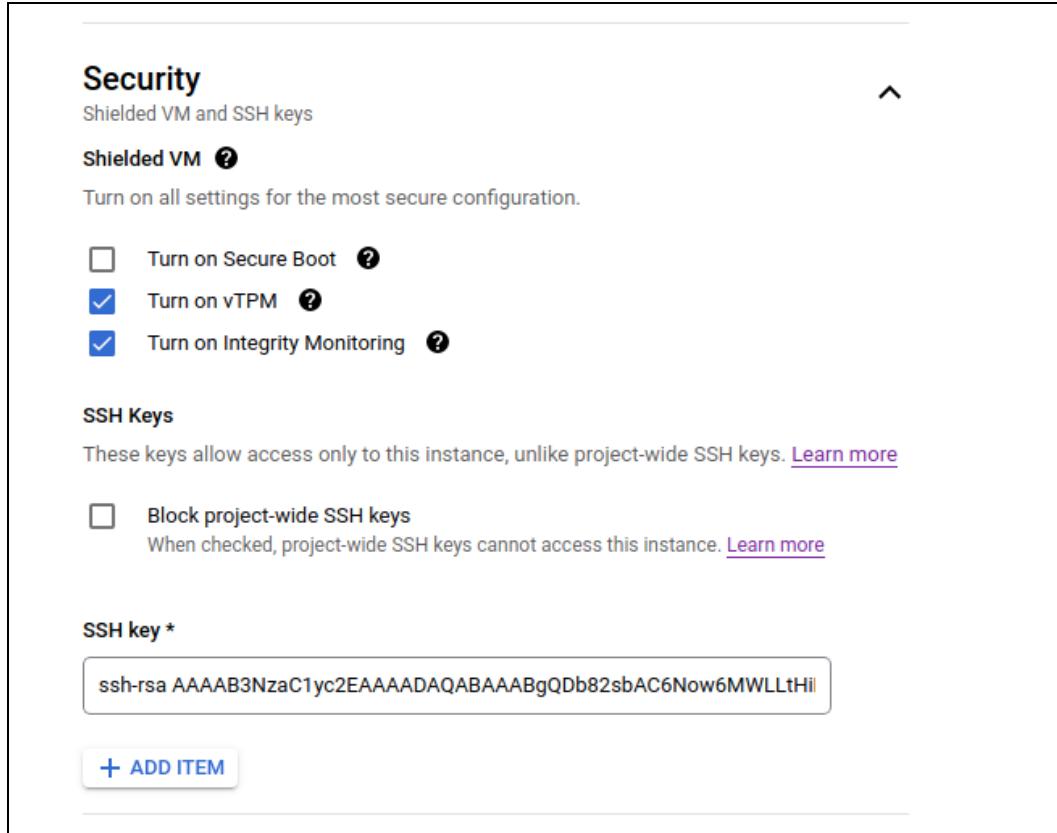
Gambar 4.4 menunjukkan spesifikasi *Virtual Machine* yang akan digunakan sebagai *Node Ethereum*. Spesifikasi dari *virtual machine* adalah

menggunakan 2vCPU dengan 1GB RAM. Sistem Operasi yang digunakan adalah Ubuntu 18.04 LTS.



Gambar 4. 5 Konfigurasi Jaringan *Virtual Machine*

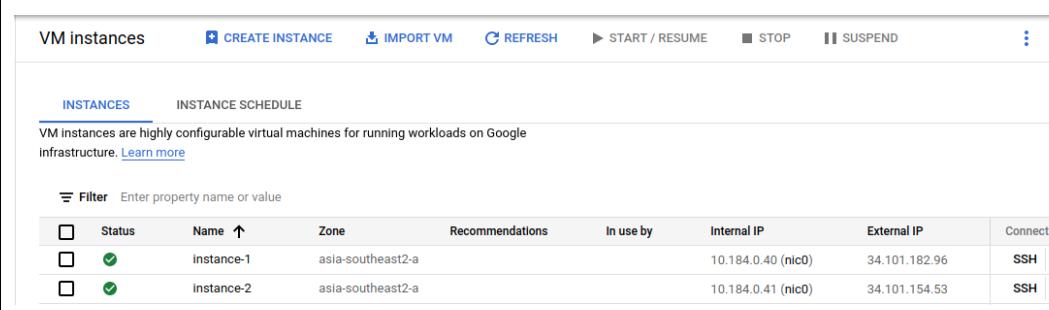
Gambar 4.5 menunjukkan *form* untuk melakukan konfigurasi *networking virtual machine*. Gunakan *tags* yang sudah dibuat sebelumnya yaitu *all-in* dan *all-out*. Ini bertujuan untuk memperbolehkan akses ke *virtual machine* dari berbagai *port* dan berbagai protokol komunikasi.



Gambar 4. 6 Konfigurasi Security *Virtual Machine*

Gambar 4.6 menunjukkan *form* untuk melakukan konfigurasi *security virtual machine*. Hal yang perlu diperhatikan adalah pada *input SSH key*. Pastikan menggunakan *public key* dari *ssh key* yang berada pada mesin lokal sehingga nantinya bisa diakses melalui *ssh*.

Lakukan hal yang sama dimulai dari pembuatan *virtual machine* sekali lagi sehingga terdapat dua *virtual machine* untuk dijadikan *node Ethereum*.



The screenshot shows the Google Cloud Compute Engine interface for managing VM instances. At the top, there are buttons for 'CREATE INSTANCE', 'IMPORT VM', 'REFRESH', 'START / RESUME', 'STOP', and 'SUSPEND'. Below this is a navigation bar with 'INSTANCES' and 'INSTANCE SCHEDULE' tabs, and a note about VM instances being highly configurable for running workloads on Google infrastructure. A 'Filter' input field is present. The main table lists two instances:

Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	Instance-1	asia-southeast2-a			10.184.0.40 (nic0)	34.101.182.96	SSH
<input checked="" type="checkbox"/>	Instance-2	asia-southeast2-a			10.184.0.41 (nic0)	34.101.154.53	SSH

Gambar 4.7 Halaman Compute Engine ketika berhasil membuat *VM*

Gambar 4.7 menunjukkan pada halaman pengelolaan Compute Engine terdapat dua *virtual machine* yang telah dibuat. Masing-masing akan mendapatkan sebuah *IP Public* yang nantinya akan digunakan untuk mengakses *virtual machine*. *IP Public* yang diperoleh bersifat *ephemeral* atau tidak statis, jadi jika *virtual machine* dimatikan kemudian dihidupkan kembali pasti akan mendapatkan *IP* yang berbeda. Setiap pengguna Google Cloud hanya diberi kuota empat *IP Public* untuk sebuah *region* untuk *virtual machine* pada Compute Engine sehingga harus digunakan dengan bijak.

4.1.2 Persiapan Virtual Machine

Setelah *virtual machine* berhasil dibuat, ada beberapa *package* yang perlu di-*install* pada *virtual machine* sebelum siap dijadikan *node ethereum*. Hal yang dibutuhkan antara lain adalah *package libsnappy-dev* dan *unzip*, *binary openethereum* serta *file* konfigurasi *node Ethereum*.

```

▶ ssh 34.101.182.96
The authenticity of host '34.101.182.96 (34.101.182.96)' can't be established.
ECDSA key fingerprint is SHA256:+NzTpxgc0En0PAvnbNBb9ImaW23wIcSh2omMIpT8rz0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '34.101.182.96' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 5.4.0-1057-gcp x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

System information as of Mon Dec 6 13:17:11 UTC 2021

System load: 0.0 Processes: 105
Usage of /: 17.1% of 9.52GB Users logged in: 0
Memory usage: 20% IP address for ens4: 10.184.0.40
Swap usage: 0%

0 updates can be applied immediately.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

```

Gambar 4.8 SSH ke Virtual Machine

Gambar 4.8 menunjukkan sukses melakukan *ssh* ke *virtual machine* yang sebelumnya telah dibuat. Pada studi kasus ini *IP* dari *virtual machine* yang pertama adalah 34.101.182.96.

```

cekingx@instance-1:~$ sudo apt install libsnappy-dev unzip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libnumal
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  libsnappy1v5
Suggested packages:
  zip
The following NEW packages will be installed:
  libsnappy-dev libsnappy1v5 unzip
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 211 kB of archives.
After this operation, 723 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://asia-southeast2.gce.archive.ubuntu.com/ubuntu bionic-updates/main amd64 unzip amd64 6.0-21ubuntu1.1 [168 kB]
Get:2 http://asia-southeast2.gce.archive.ubuntu.com/ubuntu bionic/main amd64 libsnappy1v5 amd64 1.1.7-1 [16.0 kB]
Get:3 http://asia-southeast2.gce.archive.ubuntu.com/ubuntu bionic/main amd64 libsnappy-dev amd64 1.1.7-1 [27.2 kB]
Fetched 211 kB in 1s (367 kB/s)
Selecting previously unselected package unzip.
(Reading database ... 65646 files and directories currently installed.)
Preparing to unpack .../unzip 6.0-21ubuntu1.1_amd64.deb ...
Unpacking unzip (6.0-21ubuntu1.1) ...
Selecting previously unselected package libsnappy1v5:amd64.
Preparing to unpack .../libsnappy1v5_1.1.7-1_amd64.deb ...
Unpacking libsnappy1v5:amd64 (1.1.7-1) ...
Selecting previously unselected package libsnappy-dev:amd64.
Preparing to unpack .../libsnappy-dev_1.1.7-1_amd64.deb ...
Unpacking libsnappy-dev:amd64 (1.1.7-1) ...
Setting up unzip (6.0-21ubuntu1.1) ...
Setting up libsnappy1v5:amd64 (1.1.7-1) ...
Setting up libsnappy-dev:amd64 (1.1.7-1) ...
Processing triggers for mime-support (3.60ubuntu1) ...
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...

```

Gambar 4.9 Proses Instalasi Package

Gambar 4.9 menunjukkan proses instalasi *package libsnappy-dev* dan *unzip* menggunakan *package manager* yang terdapat pada sistem operasi Ubuntu. Perintah yang dijalankan adalah `sudo apt install libsnappy-dev unzip`.

```
cekingx@instance-1:~$ cd /usr/local/bin
cekingx@instance-1:/usr/local/bin$ sudo wget https://github.com/openethereum/openethereum/releases/download/v3.2.6/openethereum-linux-v3.2.6.zip
--2021-12-06 13:18:46-- https://github.com/openethereum/openethereum/releases/download/v3.2.6/openethereum-linux-v3.2.6.zip
Resolving github.com (github.com)... 28.205.243.166
Connecting to github.com (github.com)|28.205.243.166|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/283873079/9600fb80-b48c-11eb-9c4d-272bd82febcc?X-Amz-Algorithm=AWS4-HMAC-SHA256
-X-Amz-Credential=AKIAIWJYAX4CSVEH53A%2F20211206%2Fus-east-1%2F53%2Faws4_request&X-Amz-Date=20211206T131846Z&X-Amz-Expires=300&X-Amz-Signature=f9c49e5d966c6150e9b303ca
f2cd380b2baecb411f870939e0aecab5b668e6X-Amz-SignedHeaders=host&actor_id=0&repo_id=283873079&response-content-disposition=attachment%3B%20filename%3Dopenethe
um-linux-v3.2.6.zip&response-content-type=application%2Foctet-stream [following]
--2021-12-06 13:18:46-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/283873079/9600fb80-b48c-11eb-9c4d-272bd82febcc?X-Amz-Algorithm=A
4-HMAC-SHA256&X-Amz-Credential=AKIAIWJYAX4CSVEH53A%2F20211206%2Fus-east-1%2F53%2Faws4_request&X-Amz-Date=20211206T131846Z&X-Amz-Expires=300&X-Amz-Signature=f9c49e5d9
c6150e9b303ca0f2cd380b2baecb411f870939e0aecab5b668e6X-Amz-SignedHeaders=host&actor_id=0&repo_id=283873079&response-content-disposition=attachment%3B%20file
me%3Dopenethereum-linux-v3.2.6.zip&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15122615 (14M) [application/octet-stream]
Saving to: 'openethereum-linux-v3.2.6.zip'

2021-12-06 13:18:50 (4.66 MB/s) - 'openethereum-linux-v3.2.6.zip' saved [15122615/15122615]
```

Gambar 4.10 Proses *Download Binary Openethereum*

Gambar 4.10 menunjukkan proses *download* dari *binary Openethereum*. Proses yang dilakukan adalah mengubah *directory* menjadi `/usr/local/bin`. Kemudian menggunakan perintah `wget` <https://github.com/openethereum/openethereum/releases/download/v3.2.6/openethereum-linux-v3.2.6.zip>. Setelah *download* selesai maka akan ada *file* dengan nama `openethereum-linux-v3.2.6.zip`

```
cekingx@instance-1:/usr/local/bin$ sudo unzip openethereum-linux-v3.2.6.zip
Archive: openethereum-linux-v3.2.6.zip
  inflating: ethkey
  inflating: ethstore
  inflating: openethereum
  inflating: openethereum-evm
```

Gambar 4.11 Proses *Extract File*

Gambar 4.11 menunjukkan proses *extract file* yang telah di-*download*. *Extract file* menggunakan perintah *unzip* yang telah di-*install* sebelumnya. Perintah yang dijalankan adalah `sudo unzip openethereum-linux-v3.2.6.zip` Ada empat *file* yang muncul jika proses *extract* berhasil yaitu `ethkey`, `ethstore`, `openethereum` dan `openethereum-evm`.

```
cekingx@instance-1:/usr/local/bin$ sudo chmod 755 openethereum openethereum-evm ethkey ethstore
cekingx@instance-1:/usr/local/bin$ ll
total 52660
drwxr-xr-x  2 root root    4096 Dec  6 13:19 .
drwxr-xr-x 10 root root    4096 Nov 15 17:29 ../
-rw-r-xr-x  1 root root 2435576 May 14 2021 ethkey*
-rw-r-xr-x  1 root root 2537936 May 14 2021 ethstore*
-rw-r-xr-x  1 root root 25835608 May 14 2021 openethereum*
-rw-r-xr-x  1 root root 7972160 May 14 2021 openethereum-evm*
-rw-r--r--  1 root root 15122615 Aug 17 13:10 openethereum-linux-v3.2.6.zip
```

Gambar 4. 12 Proses Ubah Permission File

Gambar 4.12 menunjukkan proses mengubah *permission* dari *file* yang sebelumnya di-*extract*. Perintah yang dijalankan adalah sudo chmod 755 openethereum openethereum-evm ethkey ethstore. Hal ini dilakukan agar *binary* bisa dijalankan.

```
cekingx@instance-1:/usr/local/bin$ cd ~
cekingx@instance-1:~$ git clone https://github.com/cekingx/simvoni-PoA.git
Cloning into 'simvoni-PoA'...
remote: Enumerating objects: 35, done.
remote: Counting objects: 100% (35/35), done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 35 (delta 16), reused 29 (delta 10), pack-reused 0
Unpacking objects: 100% (35/35), done.
cekingx@instance-1:~$ ll
total 40
drwxr-xr-x  6 cekingx cekingx 4096 Dec  6 13:20 .
drwxr-xr-x  4 root   root   4096 Dec  6 13:11 ../
-rw-r--r--  1 cekingx cekingx 220 Apr  4 2018 .bash_logout
-rw-r--r--  1 cekingx cekingx 3771 Apr  4 2018 .bashrc
drwx----- 2 cekingx cekingx 4096 Dec  6 13:17 .cache/
drwx----- 3 cekingx cekingx 4096 Dec  6 13:17 .gnupg/
-rw-r--r--  1 cekingx cekingx 807 Apr  4 2018 .profile
drwx----- 2 cekingx cekingx 4096 Dec  6 13:11 .ssh/
-rw-r--r--  1 root   root   165 Dec  6 13:18 .wget-hsts
drwxrwxr-x  3 cekingx cekingx 4096 Dec  6 13:20 simvoni-PoA/
```

Gambar 4. 13 Proses Download Konfigurasi Node Ethereum

Gambar 4.13 menunjukkan proses *download* konfigurasi *node ethereum* untuk jaringan *private blockchain*. Hal yang dilakukan adalah mengubah *directory* menjadi ~. Dalam sistem operasi berbasis *linux directory* ~ artinya *directory home* dari sebuah *user*. Proses *download* dimulai dengan menjalankan perintah git clone https://github.com/cekingx/simvoni-PoA.git . Setelah *download* selesai maka akan ada *folder* dengan nama simvoni-PoA.

Lakukan proses hal yang sama pada *virtual machine* yang lainnya karena persiapan perlu dilakukan pada kedua *virtual machine*.

4.2 Konfigurasi Openethereum

Openethereum memerlukan beberapa *file* konfigurasi untuk bisa membuat jaringan *private blockchain* karena secara *default* Openethereum akan berjalan pada jaringan utama Ethereum. Terdapat beberapa konfigurasi khusus untuk membuat sebuah *Private Network*. *Private network* adalah sebuah jaringan *blockchain* yang terisolasi dari jaringan *blockchain* utama sehingga kita bisa menyesuaikan jaringan sesuai dengan kebutuhan. Jaringan utama *blockchain* Ethereum menggunakan konsensus Proof of Work dimana konsensus ini menggunakan sangat banyak daya komputasi. Untuk keperluan voting lembaga independen seperti misalnya organisasi mahasiswa yang memiliki dana terbatas, tentunya penghematan biaya adalah prioritas. Menggunakan konsensus PoA dalam jaringan *blockchain private* adalah sebuah solusi karena jaringan tidak membutuhkan daya komputasi yang besar sehingga *VM* dengan spesifikasi rendah sudah cukup untuk menjalankan jaringan. Konsensus PoA tidak memerlukan daya komputasi yang besar karena *difficulty* untuk membuat sebuah blok relatif kecil

```
~/simvoni-PoA main ✓
tree
.
├── node.pwds
├── simvoni-config.toml
└── simvoni-spec.json

0 directories, 3 files
```

Gambar 4.14 Struktur *File* Konfigurasi

Gambar 4.14 menunjukkan struktur *file* dari *git repository* yang sudah di-*clone* sebelumnya. *File* yang dibutuhkan berupa `simvoni-spec.json`, `simvoni-config.json` dan `node.pwds`.

4.2.1 Genesis Block

Genesis block adalah blok pertama yang terdapat dalam sebuah *blockchain* yang nantinya akan disambung oleh blok lainnya sehingga bisa menjadi sebuah

rantai blok. Konfigurasi bagaimana sebuah jaringan *blockchain* bekerja tersimpan dalam *genesis block*. Jaringan *blockchain private* yang digunakan untuk implementasi Sistem Voting Elektronik bernama SimvoniPoA. Dalam jaringan SimvoniPoA terdapat dua *node* yang berada pada *Virtual Machine* yang berbeda. Kedua *node* tersebut berperan sebagai *sealer*. Dalam jaringan *blockchain* yang menggunakan konsensus PoW, *account* yang berhak membuat blok disebut dengan *miner*. Sedangkan dalam jaringan *blockchain* yang menggunakan konsensus PoA, *account* yang berhak membuat blok disebut dengan *sealer*.

```
{
    "name": "SimvoniPoA",
    "engine": {
        "authorityRound": {
            "params": {
                "stepDuration": "5",
                "validators": {
                    "list": [
                        "0x00aa39d30f0d20ff03a22ccfc30b7efbfca597c2",
                        "0x002e28950558fbebe1a9675cb113f0bd20912019"
                    ]
                }
            }
        },
        "params": {
            "gasLimitBoundDivisor": "0x400",
            "maximumExtraDataSize": "0x20",
            "minGasLimit": "0x1388",
            "networkID": "0x2323",
            "eip150Transition": "0x0",
            "eip160Transition": "0x0",
            "eip161abcTransition": "0x0",
            "eip161dTransition": "0x0",
            "eip155Transition": "0x0",
            "eip98Transition": "0x7fffffffffffff",
            "validateChainIdTransition": 0,
            "eip140Transition": "0x0",
            "eip211Transition": "0x0",
            "eip214Transition": "0x0",
            "eip658Transition": "0x0",
            "eip145Transition": "0x0",
            "eip1014Transition": "0x0",
            "eip1052Transition": "0x0"
        },
        "genesis": {
            "seal": {
                "authorityRound": {

```

Kode Program 4. 1 File simvoni-spec.json

Kode Program 4.1 *genesis block* yang terdapat pada file simvoni-spec.json. *Genesis block* tersebut menunjukkan bahwa jaringan SimvoniPoA berjalan menggunakan *authorityRound* sebagai *engine* konsensus. *authorityRound* adalah implementasi dari konsensus PoA. *Difficulty* dari jaringan ini adalah 0x20000 atau jika dikonversi menjadi desimal adalah 131.072. Angka tersebut relatif kecil jika dibandingkan dengan *difficulty* jaringan utama yang bisa mencapai 17.179.869.184. *Gas limit* dari jaringan ini adalah 0x5B8D80 atau jika dikonversi menjadi desimal adalah 6.000.000 unit.

4.2.2 Konfigurasi Node Openetherium

Node dalam sebuah jaringan Ethereum menggunakan protokol *devp2p* untuk berkomunikasi satu sama lain. *Devp2p* adalah protokol komunikasi *peer-to-peer* khusus yang dibuat untuk Ethereum. *Devp2p* menggunakan protokol komunikasi *UDP* yang secara *default* menggunakan *port* 30303. Selain berkomunikasi dengan *node* lainnya dalam jaringan Ethereum, *node* juga perlu

dikonfigurasi agar bisa berkomunikasi dengan aplikasi yang membutuhkan integrasi dengan *blockchain*. Protokol komunikasi yang digunakan adalah protokol *http*. Bentuk pesan yang dikirimkan dari aplikasi ke *node Ethereum* adalah *jsonrpc* yang artinya aplikasi menggunakan pesan dengan format *json* untuk melakukan *remote procedure call* pada *node Ethereum*. Terdapat beberapa *jsonrpc api* yang disediakan oleh *node Ethereum* yaitu *all*, *safe*, *debug*, *web3*, *net*, *eth*, *pubsub*, *personal*, *signer*, *parity*, *parity_pubsub*, *parity_accounts*, *parity_set*, *traces* dan *secretstore*. Konfigurasi *CORS* juga perlu diperhatikan apabila aplikasi yang ingin melakukan *http request* ke *node* menggunakan bahasa pemrograman JavaScript karena tanpa adanya *CORS*, *http request* yang dilakukan oleh aplikasi tersebut akan gagal.

```
[parity]
chain = "simvoni-spec.json"
base_path = "$HOME/openethereum-node"
[network]
port = 30300
allow_ips = "all"
nat = "0.0.0.0"
[rpc]
interface = "all"
port = 8540
apis = ["all"]
cors = ["all"]
hosts = ["all"]
[websockets]
port = 8450
interface = "all"
origins = ["all"]
[account]
password = ["node.pwds"]
[mining]
# engine_signer = ""
reseal_on_txs = "all"
reseal_max_period = 4000
```

Kode Program 4.2 File simvoni-config.toml

Kode Program 4.2 adalah konfigurasi dari *node* yang terdapat dalam file *simvoni-config.toml*. Dalam bagian *parity*, konfigurasi *chain* adalah *file genesis block* yang digunakan dalam jaringan. Konfigurasi *base_path* adalah direktori penyimpanan data *blockchain*. Dalam bagian *network*, konfigurasi *port* adalah *port* yang digunakan oleh *node* untuk melakukan komunikasi *peer-to-peer* di dalam

jaringan. Dalam bagian *rpc*, konfigurasi *interface* adalah *jsonrpc api* yang boleh digunakan pada *node*. Nilai *all* artinya semua *jsonrpc api* yang tersedia boleh untuk digunakan. Konfigurasi *cors* adalah daftar *IP* mana saja yang boleh melakukan *http request* menggunakan *javascript*. Nilai *all* artinya semua *IP* boleh melakukan *http request* dari *javascript*.

4.2.3 Menjalankan Jaringan

Setelah memenuhi kebutuhan untuk menjalankan *binary* Openethereum diperlukan beberapa langkah tambahan agar jaringan *private blockchain* dapat berjalan.

```
openethereum --config simvoni-config.toml --nat
extip:34.101.182.96 --no-color
```

Kode Program 4.3 Perintah Menjalankan Openethereum

Kode Program 4.3 adalah perintah yang digunakan untuk menjalankan Openethereum dalam sebuah *virtual machine*. Sesuaikan *IP virtual machine* pada bagian *extip*.

```
curl -X POST \
-H "Content-Type: application/json" \
-d '{
"jsonrpc":"2.0",
"method":"parity_newAccountFromPhrase",
"params":["node1", "node1-password"],
"id":0
}' \
http://34.101.182.96:8540
```

Kode Program 4.4 CURL untuk Membuat *Validator Account*

Kode Program 4.4 adalah *curl* yang digunakan untuk membuat *validator account* pada *virtual machine* dengan IP 34.101.182.96.

```
curl -X POST \
-H "Content-Type: application/json" \
-d '{
"jsonrpc":"2.0",
"method":"parity_newAccountFromPhrase",
"params":["node2", "node2-password"],
"id":0
}' \
http://34.101.182.96:8540
```

```
}' \
http://34.101.154.53:8540
```

Kode Program 4.5 CURL untuk Membuat *Validator Account*

Kode Program 4.5 adalah *curl* yang digunakan untuk membuat *validator account* pada *virtual machine* dengan IP 34.101.154.53.

```
echo "node1-password" > node.pwds
```

Kode Program 4.6 Menyimpan *Password Validator*

Kode Program 4.6 adalah perintah yang digunakan untuk menyimpan *password* dari *validator account* pada *file* node.pwds.

Ulangi kembali langkah untuk menjalankan Openethereum sehingga Openethereum sudah mendapatkan *update* konfigurasi.

Setiap *node* Ethereum yang berjalan diidentifikasi dengan sebuah *url enode*. Cara menghubungkan *node* Ethereum adalah dengan mendapatkan *url enode* dari sebuah *node* kemudian menambahkan *url* tersebut ke *node* lainnya

```
curl --data '{
  "jsonrpc": "2.0",
  "method": "parity_enode",
  "params": [],
  "id": 0
}' -H "Content-Type: application/json"
-X POST
34.101.182.96:8540
```

Kode Program 4.7 Mendapatkan Enode

Kode Program 4.7 adalah perintah untuk melakukan *http request* dengan bentuk *jsonrpc* ke sebuah *node* Ethereum untuk mendapatkan *url enode* dari *node* tersebut.

```
curl --data '{
  "jsonrpc": "2.0",
  "method": "parity_addReservedPeer",
  "params": ["enode://RESULT"],
  "id": 0
}' -H "Content-Type: application/json"
-X POST
34.101.154.53:8540
```

Kode Program 4.8 Menambahkan Enode

Kode Program 4.8 adalah perintah untuk melakukan *http request* dengan bentuk *jsonrpc* ke *node* Ethereum lainnya dengan argumen *url enode* yang sudah didapatkan sebelumnya.

2021-12-06 13:49:03 UTC	0/25 peers	1 KiB chain 0 bytes queue	RPC: 0 conn,	0 req/s,	44 μ
2021-12-06 13:49:33 UTC	0/25 peers	1 KiB chain 0 bytes queue	RPC: 0 conn,	0 req/s,	44 μ
2021-12-06 13:50:03 UTC	1/25 peers	1 KiB chain 0 bytes queue	RPC: 0 conn,	0 req/s,	44 μ
2021-12-06 13:50:33 UTC	1/25 peers	1 KiB chain 0 bytes queue	RPC: 0 conn,	0 req/s,	44 μ
2021-12-06 13:51:03 UTC	1/25 peers	1 KiB chain 0 bytes queue	RPC: 0 conn,	0 req/s,	44 μ

Gambar 4. 15 Node Terhubung

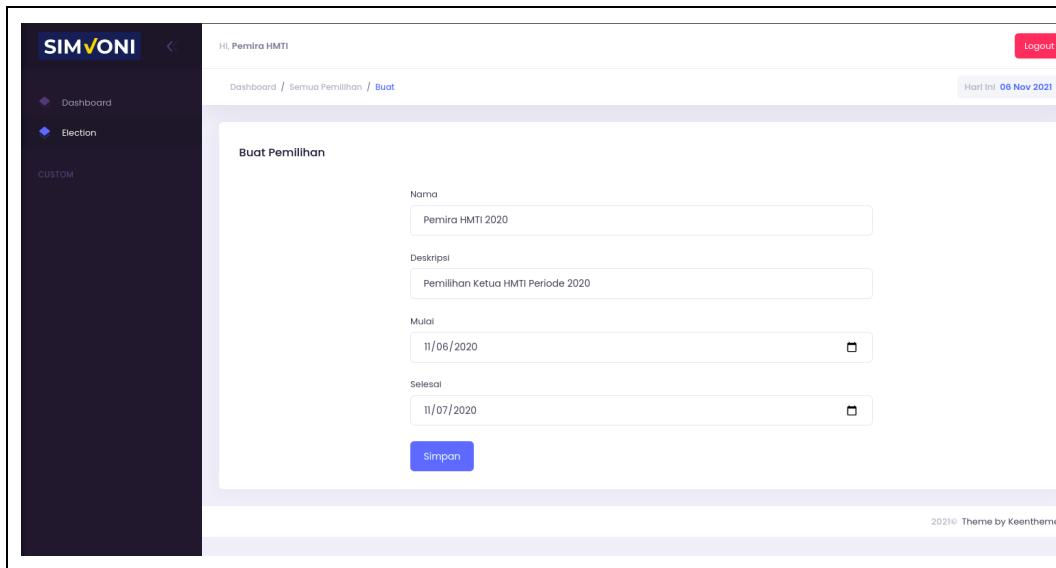
Gambar 4.15 menunjukkan perbedaan ketika *node* Ethereum yang belum terhubung dan sudah terhubung. Perbedaannya adalah ketika *node* belum terhubung, *log* yang muncul adalah 0/25 *peer*. Jika sudah terhubung *log* akan berubah menjadi 1/25 *peer*.

4.3 Implementasi Desain Antarmuka

Implementasi Desain Antarmuka adalah proses pembuatan tampilan yang bisa digunakan dalam sistem voting elektronik. Aplikasi dapat digunakan oleh 3 jenis user role yaitu *Super Admin*, *Election Authority* dan *Voter*.

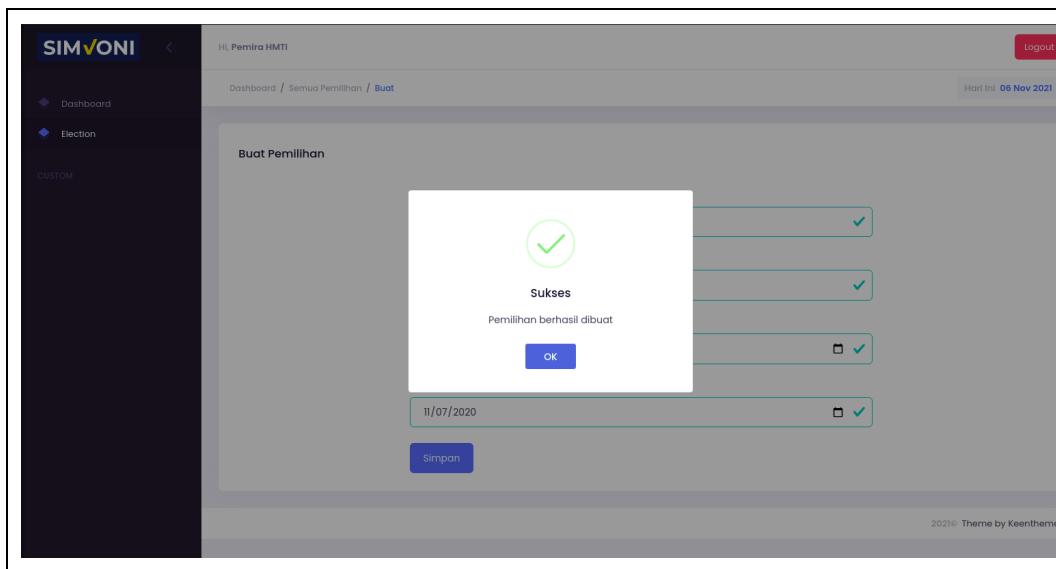
4.3.1 Membuat Pemilihan

Halaman pembuatan pemilihan adalah tampilan awal dari sebuah proses voting elektronik. Halaman ini terdiri dari sebuah *form* dengan empat buah *input* serta sebuah *button*.



Gambar 4. 16 Tampilan Halaman Pembuatan Pemilihan

Gambar 4.16 menunjukkan *form* untuk membuat sebuah pemilihan terdiri dari empat buah *input* yaitu Nama, Deskripsi, Mulai dan Selesai. Serta sebuah *button* Simpan untuk melakukukan *http request* ke *backend*.



Gambar 4. 17 Tampilan Pesan Sukses

Gambar 4.17 menampilkan pesan “Pemilihan Berhasil Dibuat” yang diberikan ketika berhasil melakukan *http request* untuk membuat sebuah pemilihan.

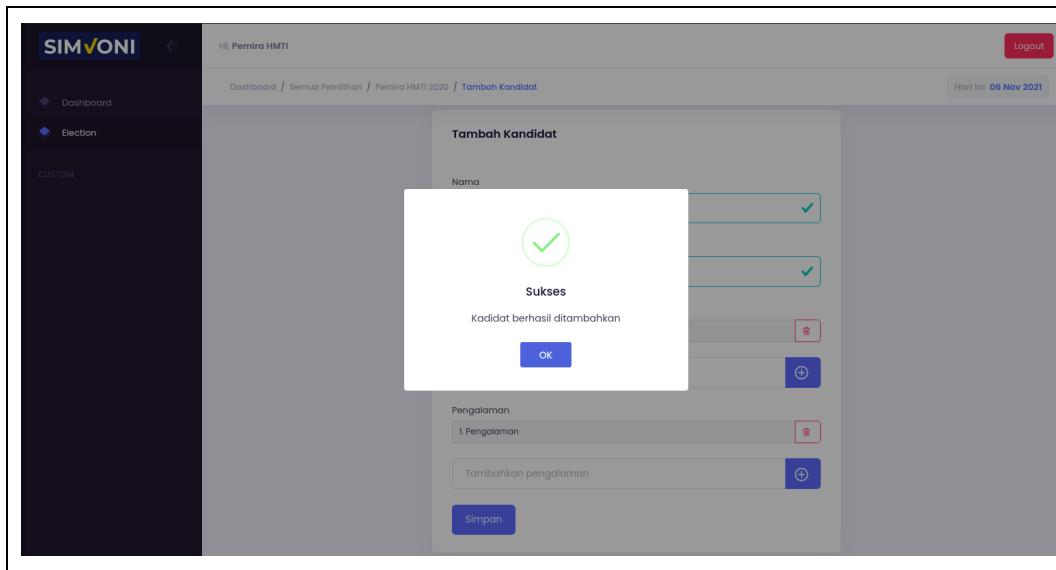
4.3.2 Menambahkan Kandidat

Halaman penambahan kandidat adalah tampilan untuk menambahkan kandidat pada sebuah pemilihan yang telah berhasil dibuat. Halaman ini terdiri dari sebuah *form* yang terdiri dari empat *input* dan sebuah *button*.

The screenshot shows the 'Tambah Kandidat' (Add Candidate) page. The left sidebar has 'Dashboard' and 'Election' sections. The main header says 'Hi, Pemira HMTI'. Below it, the breadcrumb navigation shows 'Dashboard / Semua Pemilihan / Pemira HMTI 2020 / Tambah Kandidat'. On the right, there's a 'Logout' button and the date 'Hari Ini 08 Nov 2021'. The central form has fields for 'Nama' (Name) containing 'Kandidat Satu', 'Visi' (Vision) containing 'Visi', 'Misi' (Mission) with a list '1. Misi' and a '+' button to add more, and 'Pengalaman' (Experience) with a list '1. Pengalaman' and a '+' button to add more. A blue 'Simpan' (Save) button is at the bottom.

Gambar 4.18 Tampilan Halaman Penambahan Kandidat

Gambar 4.18 menunjukkan *form* untuk menambahkan seorang kandidat pada sebuah pemilihan terdiri dari empat *input* yaitu Nama, Visi, Misi dan Pengalaman. Khusus untuk *input* Misi dan Pengalaman bersifat *multiple input*. Serta sebuah *button* Simpan untuk melakukan *http request* ke *backend*.

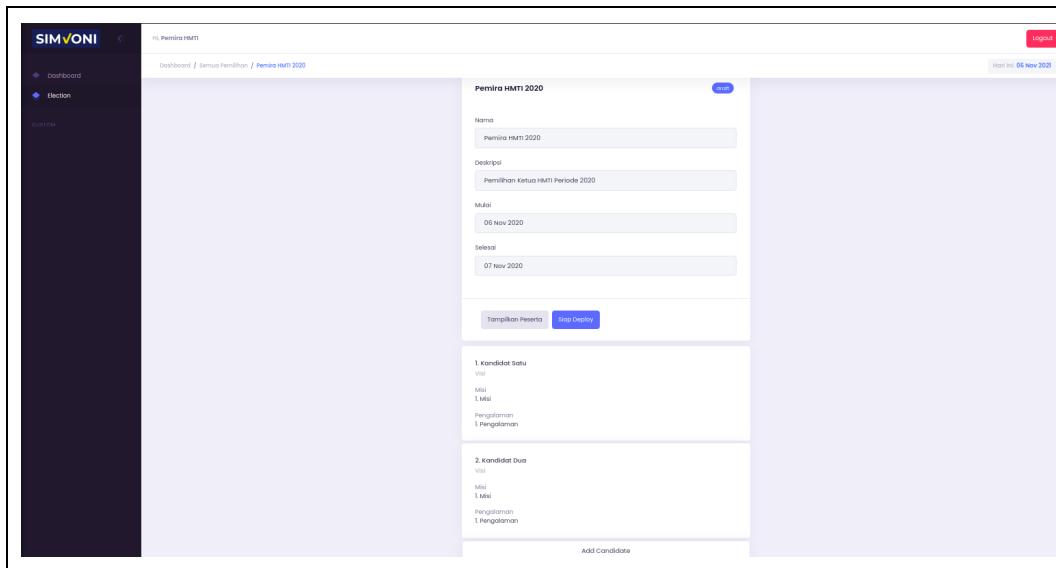


Gambar 4. 19 Tampilan Pesan Sukses

Gambar 4.19 menampilkan pesan “Kandidat Berhasil Ditambahkan” yang diberikan ketika berhasil melakukan *http request* untuk menambahkan seorang kandidat.

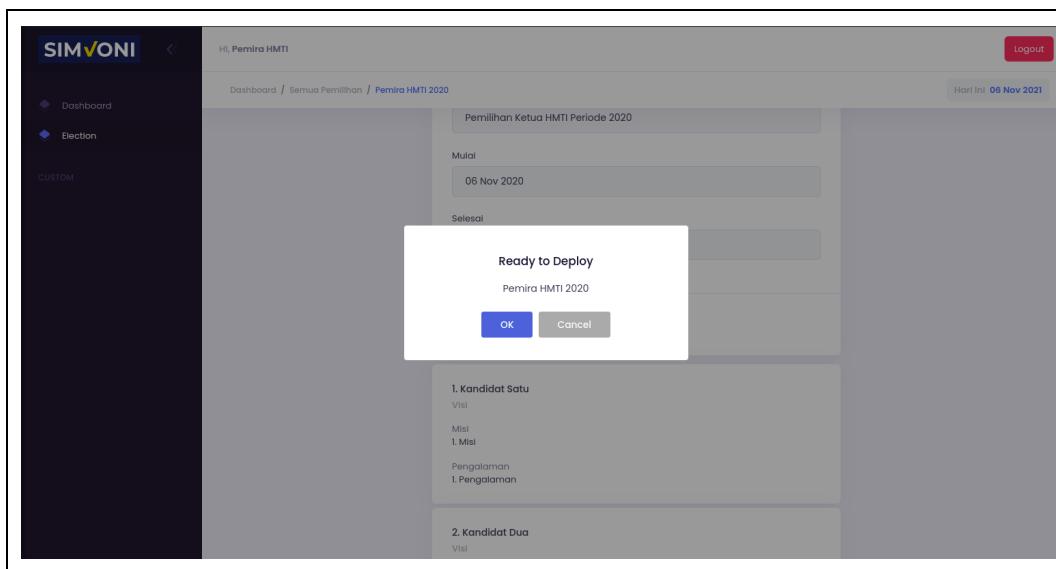
4.3.3 Deploy Pemilihan

Melakukan *deployment* sebuah pemilihan memerlukan langkah-langkah yang lebih panjang yaitu status pemilihan akan diubah menjadi “Siap Deploy” oleh *Election Authority* kemudian akan di-*deploy* oleh *Super Admin*.



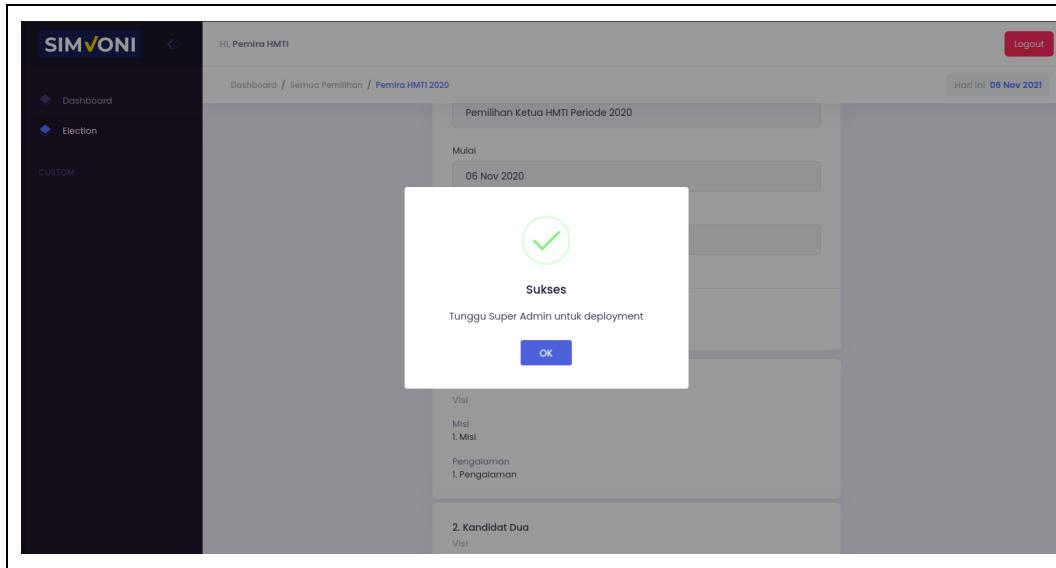
Gambar 4. 20 Tampilan *Detail* Pemilihan

Gambar 4.20 menunjukkan tampilan *detail* sebuah pemilihan dari sisi *Election Authority*. Tampilan *detail* pemilihan terdapat informasi dasar pemilihan seperti Nama, Deskripsi, Mulai dan Selesai serta daftar kandidat. Pada tampilan tersebut terdapat *button* “Siap Deploy” untuk mengubah status pemilihan.



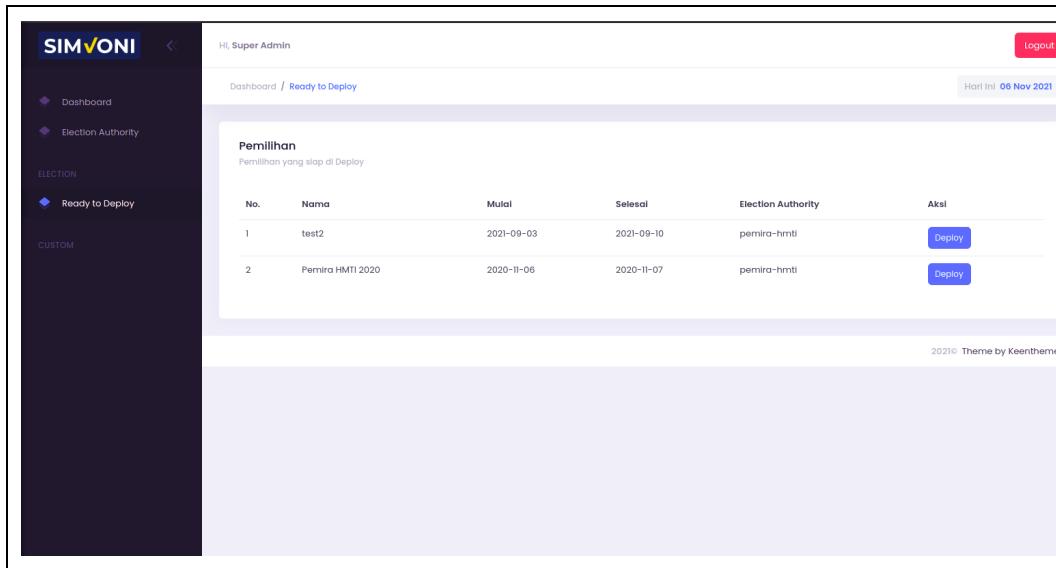
Gambar 4. 21 Tampilan Konfirmasi

Gambar 4.21 menampilkan *modal* untuk konfirmasi apakah yakin untuk mengubah status pemilihan tersebut. Jika memilih “Cancel” tidak akan terjadi apa-apa, sedangkan jika memilih “OK” maka akan membuat *http request* ke *backend*.



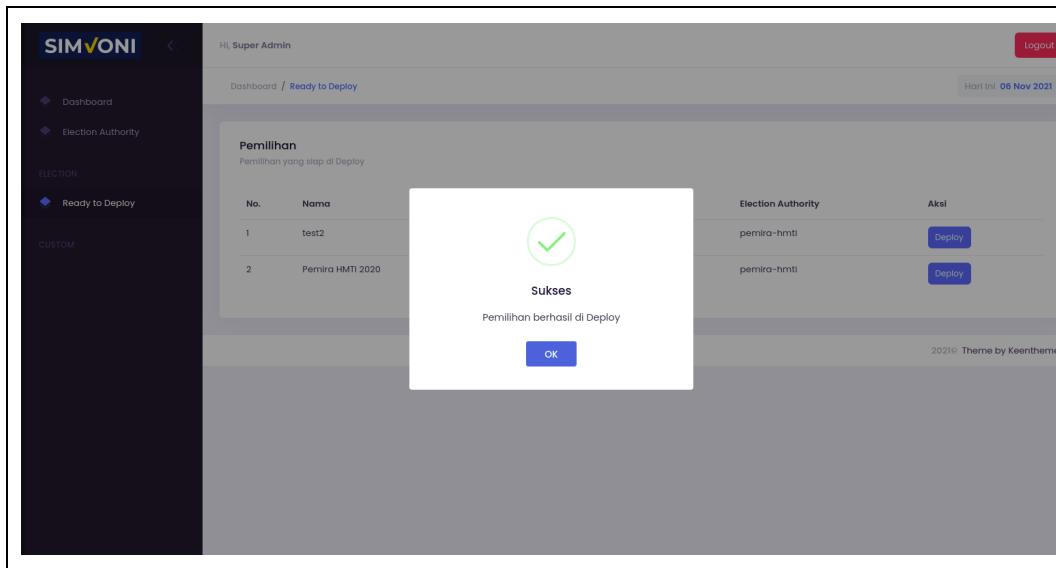
Gambar 4.22 Tampilan Pesan Sukses

Gambar 4.22 menampilkan pesan “Tunggu Super Admin Untuk Deployment” yang diberikan ketika berhasil melakukan *http request* untuk mengubah status pemilihan.



Gambar 4.23 Tampilan Daftar Pemilihan yang Siap Deploy

Gambar 4.23 menunjukkan daftar pemilihan yang statusnya siap untuk di-*deploy*. Halaman ini hanya bisa dilihat oleh *Super Admin*. Terdapat button “Deploy” untuk melakukan *deployment* terhadap sebuah pemilihan.

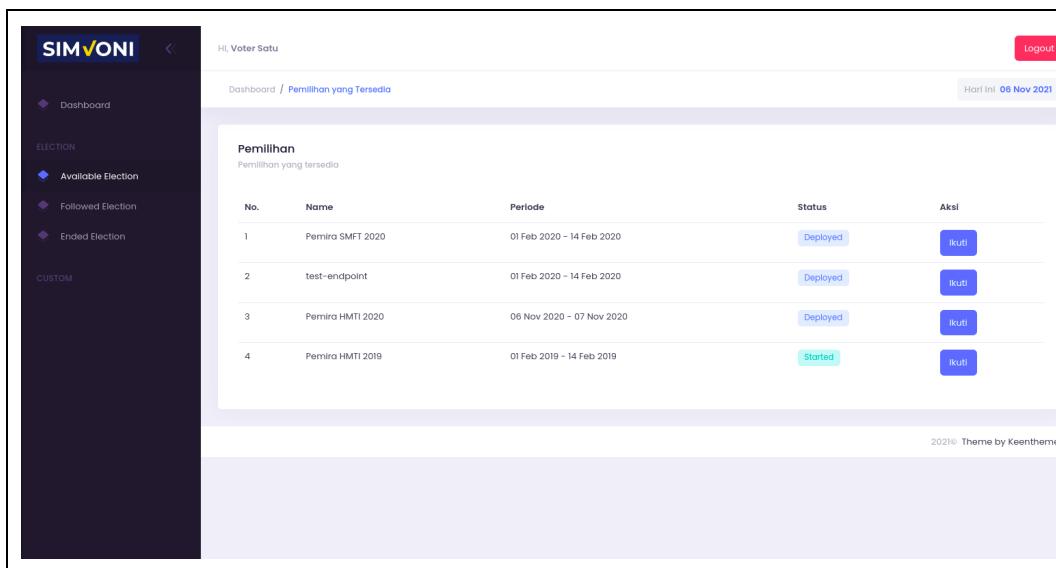


Gambar 4. 24 Tampilan Pesan Sukses

Gambar 4.24 menampilkan pesan “Pemilihan Berhasil di Deploy” yang diberikan ketika sukses melakukan *http request* untuk *deploy* pemilihan.

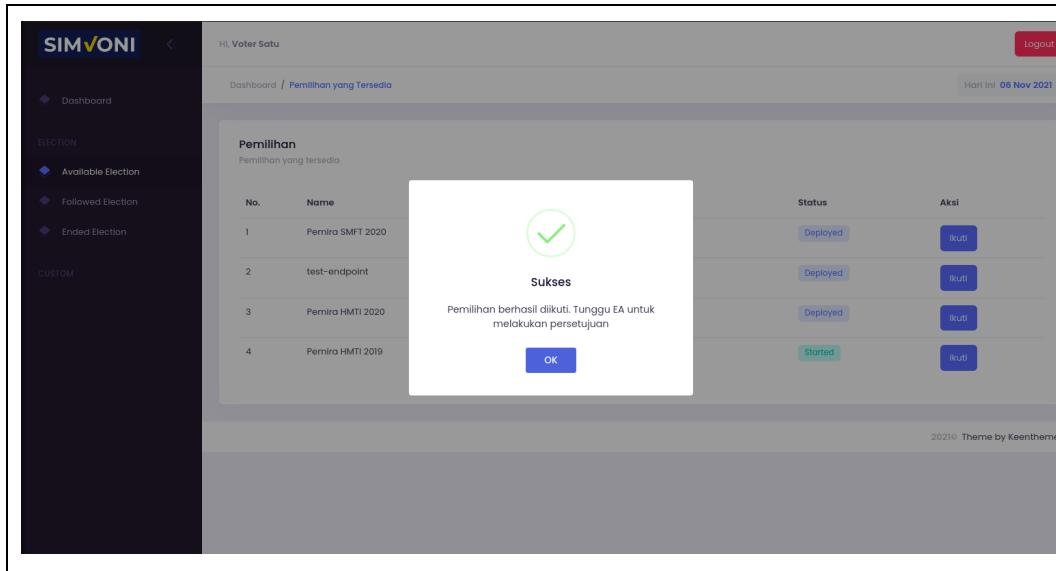
4.3.4 Mengikuti Pemilihan

Halaman ikuti pemilihan adalah tampilan *Voter* untuk mengikuti sebuah pemilihan. Halaman ini terdiri dari daftar pemilihan serta *button* pada masing-masing pemilihan.



Gambar 4. 25 Tampilan Daftar Pemilihan

Gambar 4.25 menunjukkan tampilan halaman untuk mengikuti pemilihan. Terdapat daftar pemilihan yang tersedia di sistem voting elektronik. Informasi yang ditampilkan masing-masing pemilihan adalah Nama, Periode, Status serta *button* untuk mengikuti pemilihan.

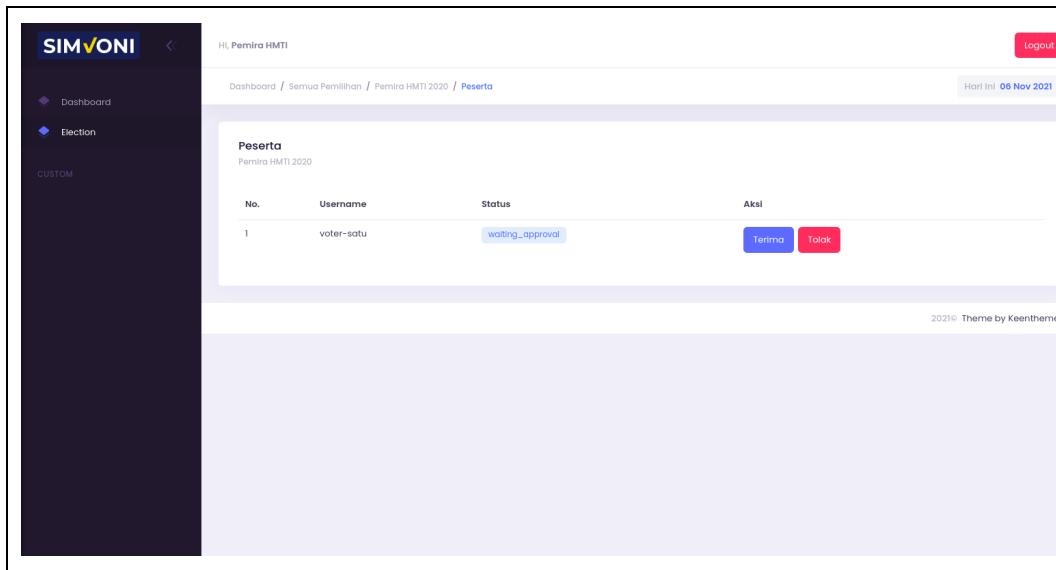


Gambar 4. 26 Tampilan Pesan Sukses

Gambar 4.26 menampilkan pesan “Pemilihan berhasil dilikti. EA adalah akronim dari *Election Authority*. Tunggu EA untuk melakukan persetujuan” ketika berhasil melakukan *http request* untuk mengikuti pemilihan.

4.3.5 Daftar Peserta Pemilihan

Halaman daftar peserta pemilihan adalah tampilan pada *Election Authority* untuk melihat dan melakukan aksi terhadap peserta pemilihan.

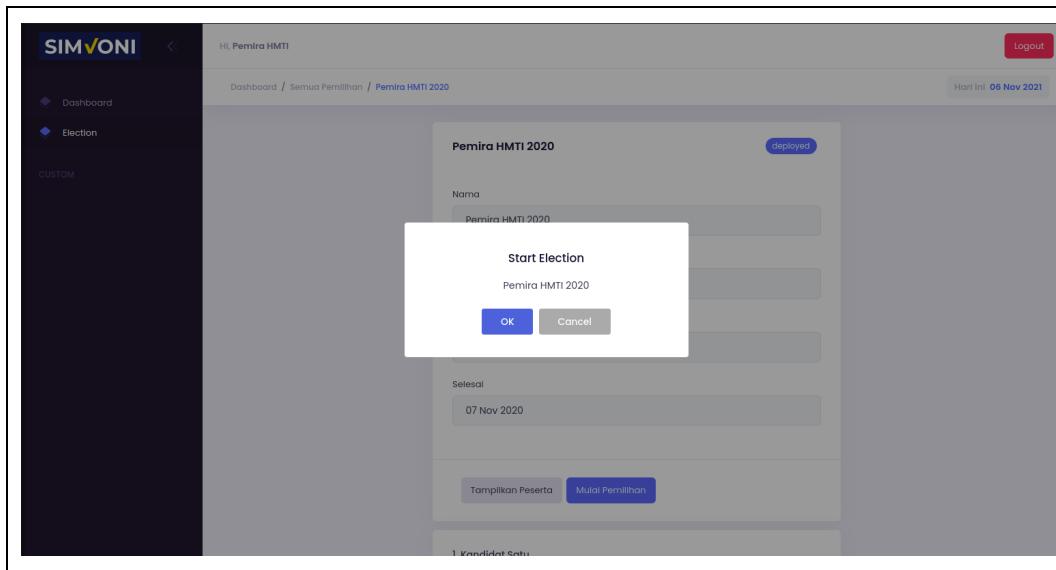


Gambar 4.27 Tampilan Daftar Peserta Pemilihan

Gambar 4.27 menunjukkan daftar peserta dalam sebuah pemilihan. Informasi yang ditampilkan dari masing-masing peserta adalah *Username*, *Status* serta dua buah *button* aksi.

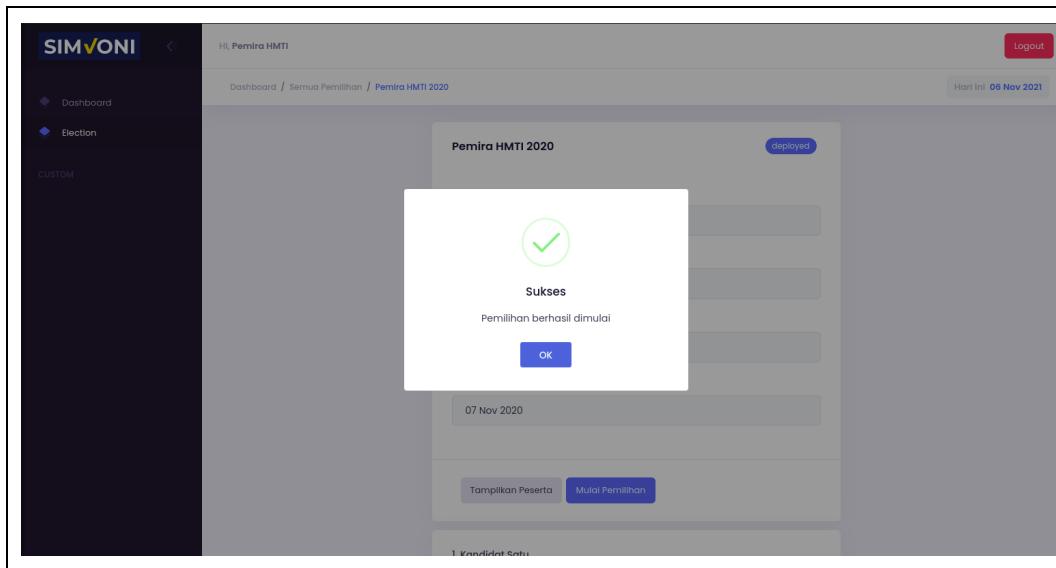
4.3.6 Memulai Pemilihan

Halaman mulai pemilihan adalah tampilan *detail* pemilihan namun terdapat *button* yang berbeda dari detail pemilihan yang sebelumnya yaitu *button* Mulai Pemilihan.



Gambar 4.28 Tampilan Konfirmasi Mulai Pemilihan

Gambar 4.28 menunjukkan *modal* yang muncul ketika menekan *button* “Mulai Pemilihan” pada halaman detail pemilihan. Jika menekan “Cancel” maka tidak akan terjadi apa-apa sedangkan menekan tombol “OK” akan melakukan *http request* ke *backend* untuk memulai pemilihan.

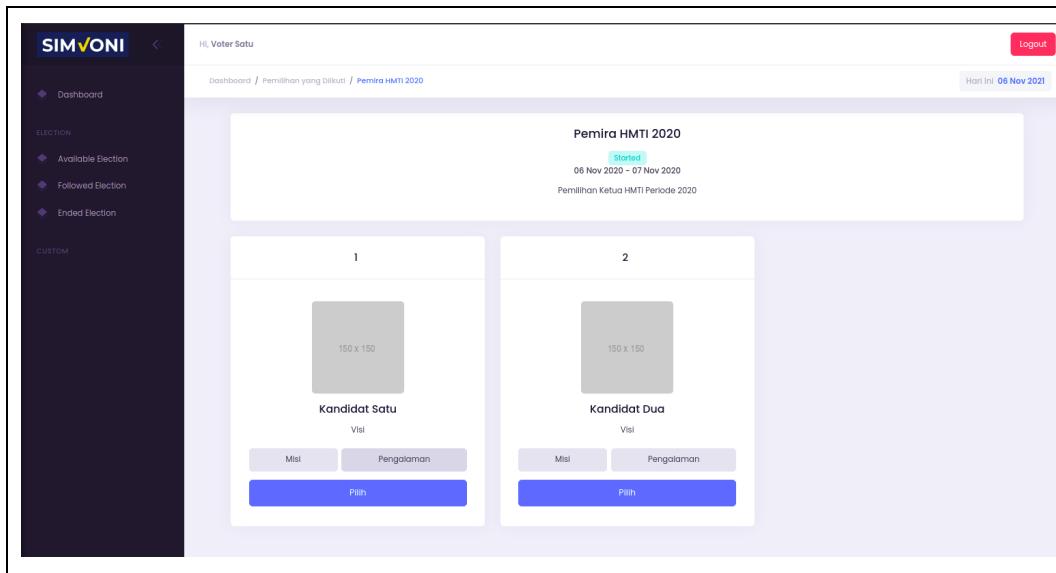


Gambar 4.29 Tampilan Pesan Sukses

Gambar 4.29 menampilkan pesan “Pemilihan berhasil dimulai” ketika sukses melakukan *http request* ke *backend* untuk memulai pemilihan.

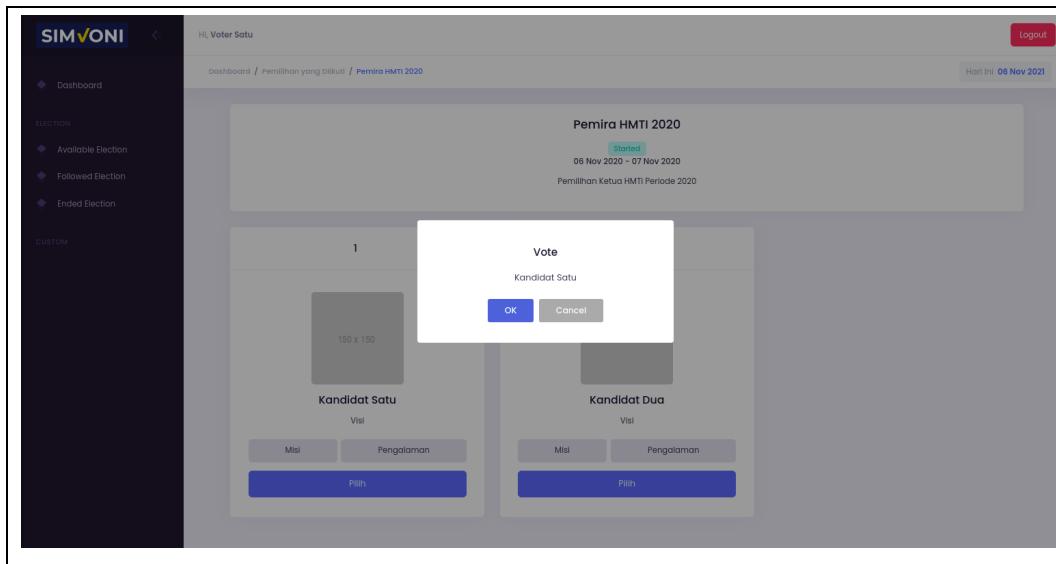
4.3.7 Vote

Halaman *vote* adalah tampilan pada sisi *Voter* yang sudah terdaftar dalam sebuah pemilihan untuk memilih seorang kandidat. Terdapat daftar kandidat serta *button* untuk memilih.



Gambar 4.30 Tampilan *Vote* Sebuah Pemilihan

Gambar 4.30 menunjukkan tampilan halaman untuk memilih seorang kandidat. Terdapat *card* untuk masing-masing kandidat. Setiap *card* memiliki Nomor Urut, Nama, Visi, *button* untuk melihat Misi dan Pengalaman serta *button* untuk Memilih.

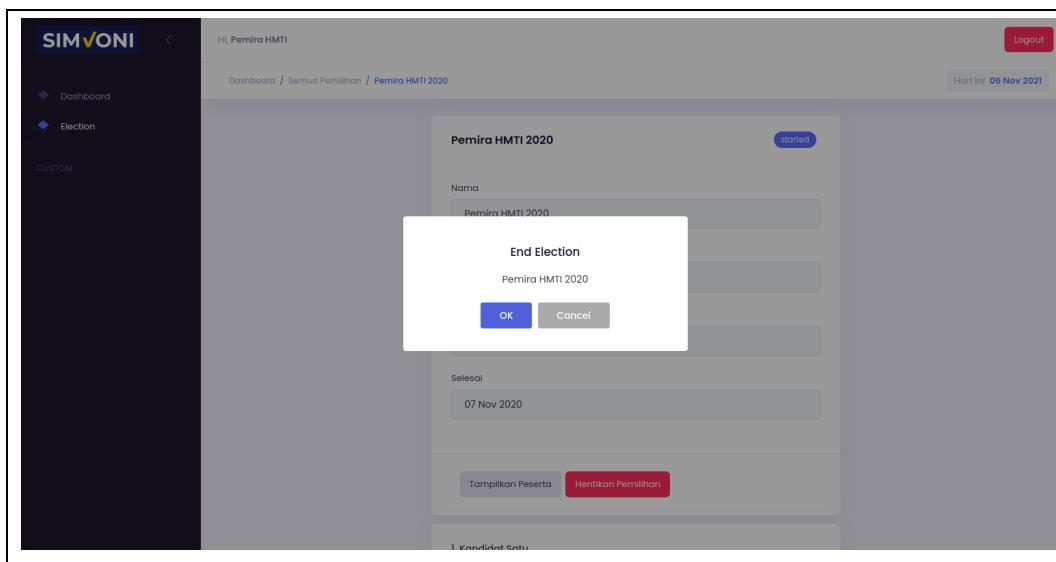


Gambar 4.31 Tampilan Konfirmasi Memilih Kandidat

Gambar 4.31 menunjukkan *modal* yang muncul untuk melakukan konfirmasi terhadap pemilihan kandidat. Jika memilih “Cancel” maka tidak akan terjadi apa-apa sedangkan menekan “OK” akan melakukan *http request* ke *backend* untuk memilih kandidat.

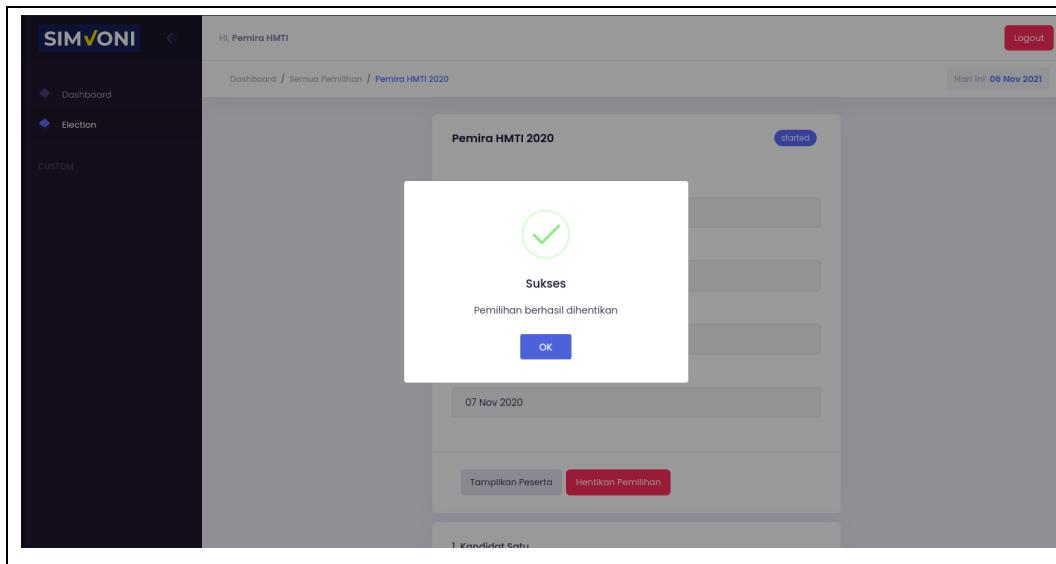
4.3.8 Menghentikan Pemilihan

Halaman menghentikan pemilihan adalah tampilan pada sisi *Election Authority* untuk menghentikan sebuah pemilihan yang telah berjalan.



Gambar 4.32 Tampilan Konfirmasi Menghentikan Pemilihan

Gambar 4.32 menunjukkan modal yang muncul ketika menekan button “Hentikan Pemilihan” pada tampilan detail pemilihan. Jika menekan “Cancel” tidak akan terjadi apa-apa sedangkan menekan “OK” akan melakukan *http request* ke *backend* untuk menghentikan pemilihan.

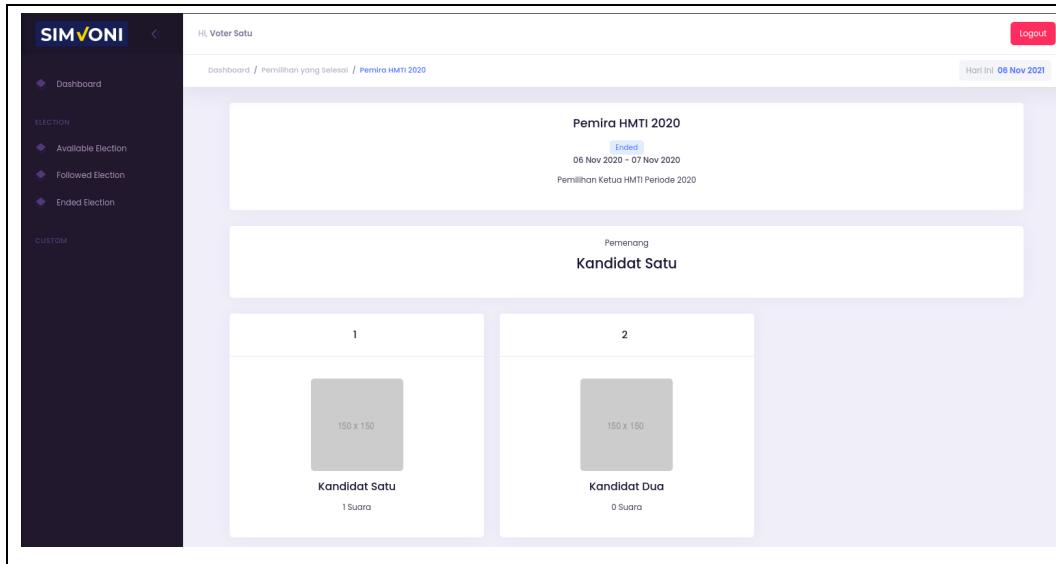


Gambar 4.33 Tampilan Pesan Sukses

Gambar 4.33 menampilkan pesan “Pemilihan berhasil dihentikan” ketika sukses melakukan *http request* ke *backend* untuk menghentikan pemilihan.

4.3.9 Melihat Hasil Pemilihan

Halaman hasil pemilihan adalah tampilan dari sisi *Voter* untuk melihat hasil dari pemilihan yang telah diikuti sebelumnya.



Gambar 4.34 Tampilan Hasil Pemilihan

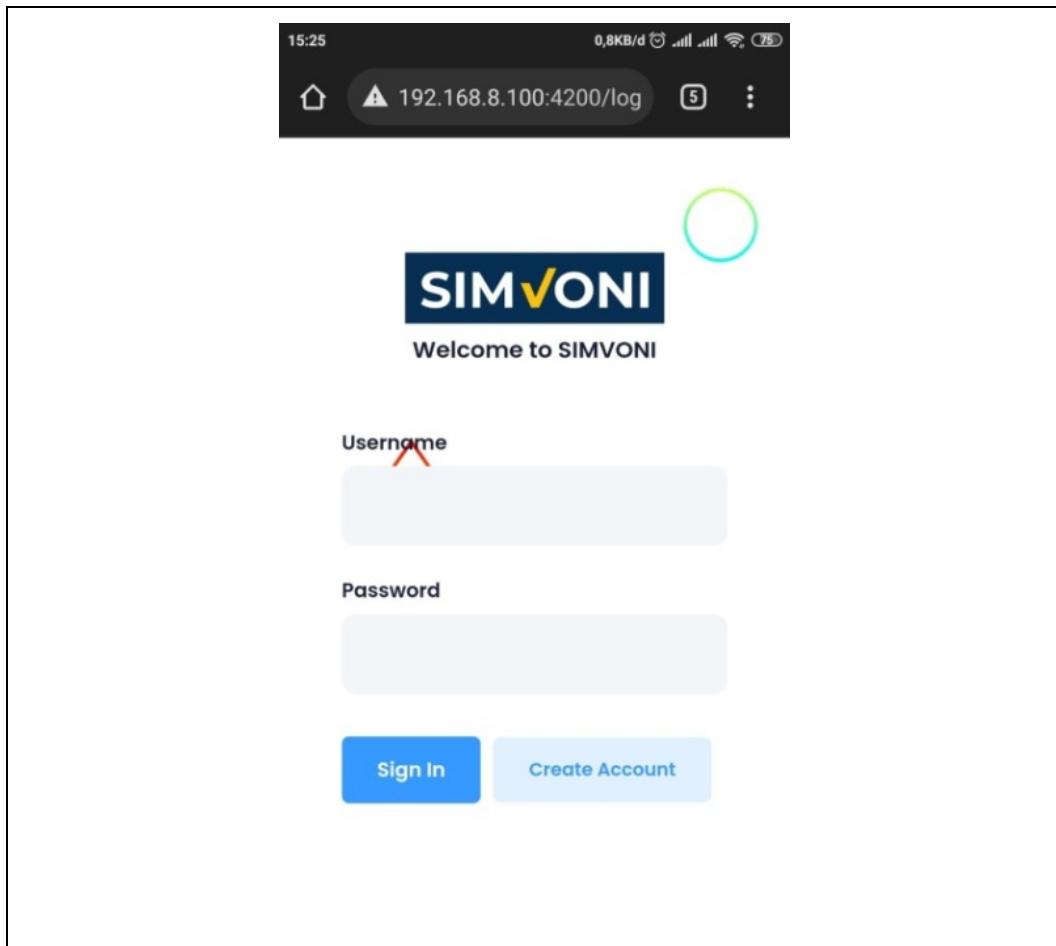
Gambar 4.34 menunjukkan tampilan untuk melihat hasil dari sebuah pemilihan. Tampilan ini menyerupai tampilan *Vote* namun terdapat perbedaan yaitu terdapat *card* tambahan untuk melihat pemenang pemilihan serta *button* pada *card* kandidat diganti menjadi jumlah suara yang diperoleh.

4.4 Implementasi Desain Antarmuka Mobile

Implementasi Desain Antarmuka Mobile adalah pembuatan tampilan untuk *Voter* yang memiliki tampilan *responsive* sehingga bisa digunakan pada perangkat *mobile*.

4.4.1 Login

Halaman login adalah halaman yang digunakan *user* untuk masuk ke dalam aplikasi simvoni. Diperlukan *username* dan *password* yang sesuai untuk berhasil masuk ke dalam sistem.

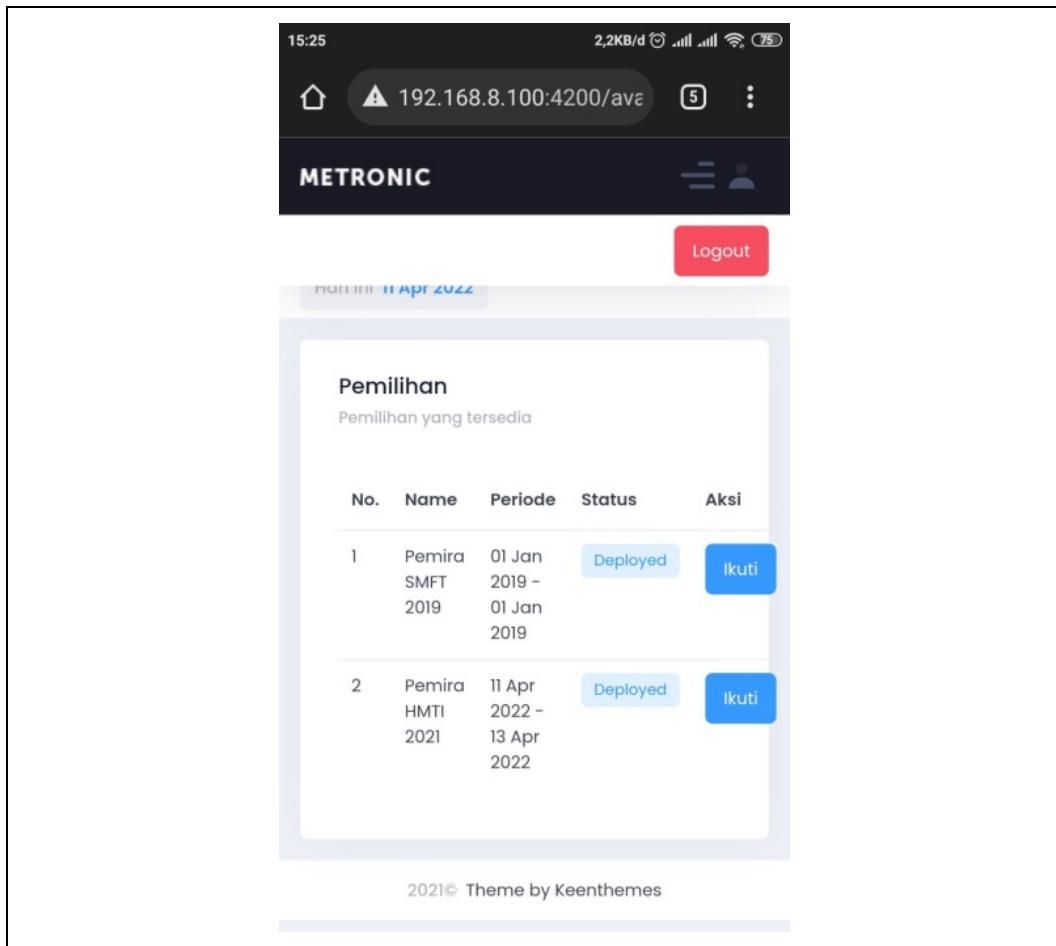


Gambar 4. 35 Tampilan Login Mobile

Gambar 4.35 merupakan tampilan halaman login yang digunakan pada perangkat *mobile*. Terdapat sebuah *form* dengan dua *input* yaitu *input username* dan *password* serta 2 buah tombol yaitu tombol *sign in* dan *create account*.

4.4.2 Available Election

Halaman *available election* adalah halaman yang menampilkan seluruh daftar pemilihan yang tersedia pada aplikasi simvoni dan belum diikuti oleh *voter*. Pada halaman ini nantinya *voter* dapat melihat dan memilih pemilihan mana saja yang dapat diikuti.

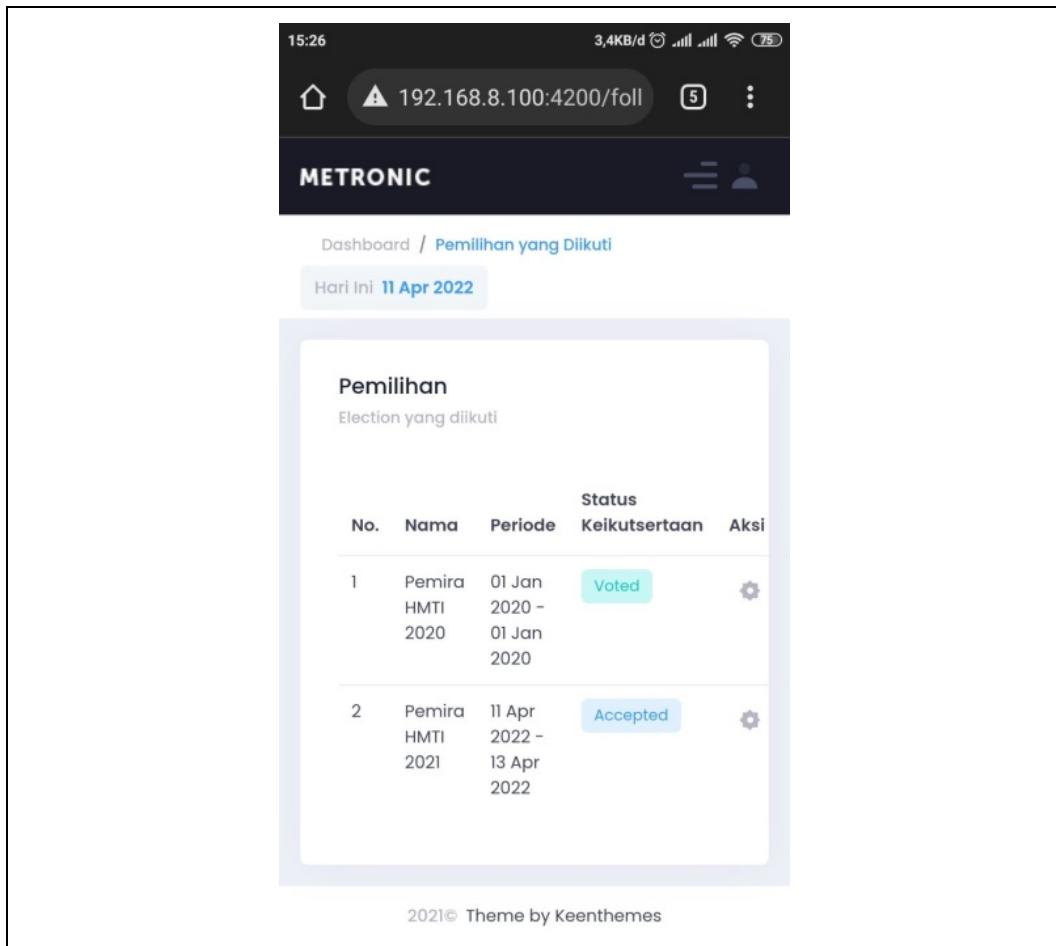


Gambar 4. 36 Tampilan *Available Election*

Gambar 4.36 merupakan tampilan *available election* yang digunakan untuk melihat daftar pemilihan yang tersedia dan belum diikuti oleh *voter*. Setiap pemilihan memiliki empat komponen, yaitu nama pemilihan, periode pemilihan, status pemilihan dan tombol untuk mengikuti pemilihan.

4.4.3 Followed Election

Halaman *followed election* adalah halaman yang menunjukkan daftar pemilihan yang telah diikuti oleh *voter*, baik pemilihan yang belum dimulai maupun pemilihan yang sedang berjalan yang diikuti oleh *voter*.

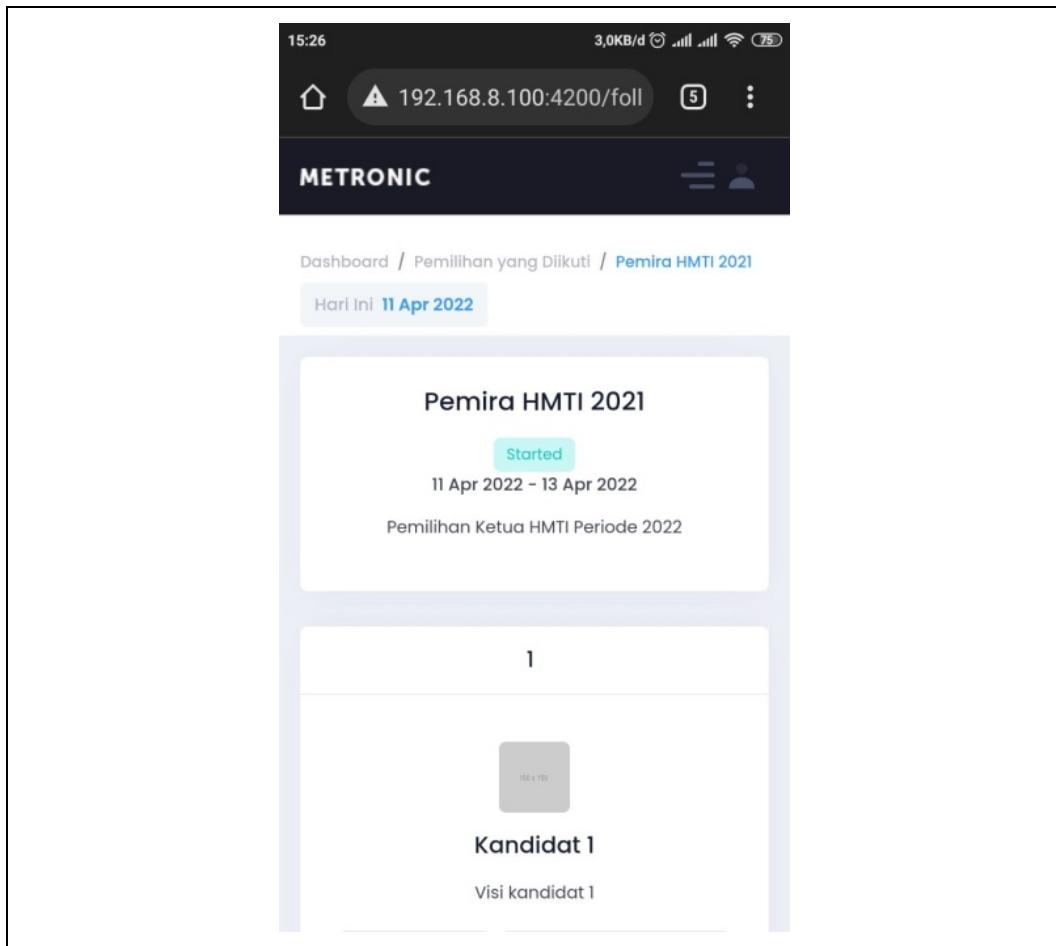


Gambar 4. 37 Tampilan Followed Election

Gambar 4.37 merupakan tampilan *followed election* yang digunakan untuk melihat pemilihan yang akan diikuti maupun yang sedang diikuti oleh *voter*. Setiap pemilihan memiliki empat komponen yaitu nama pemilihan, periode pemilihan, status keikutsertaan pemilihan dan tombol aksi yang berfungsi mengarahkan *voter* ke halaman detail pemilihan.

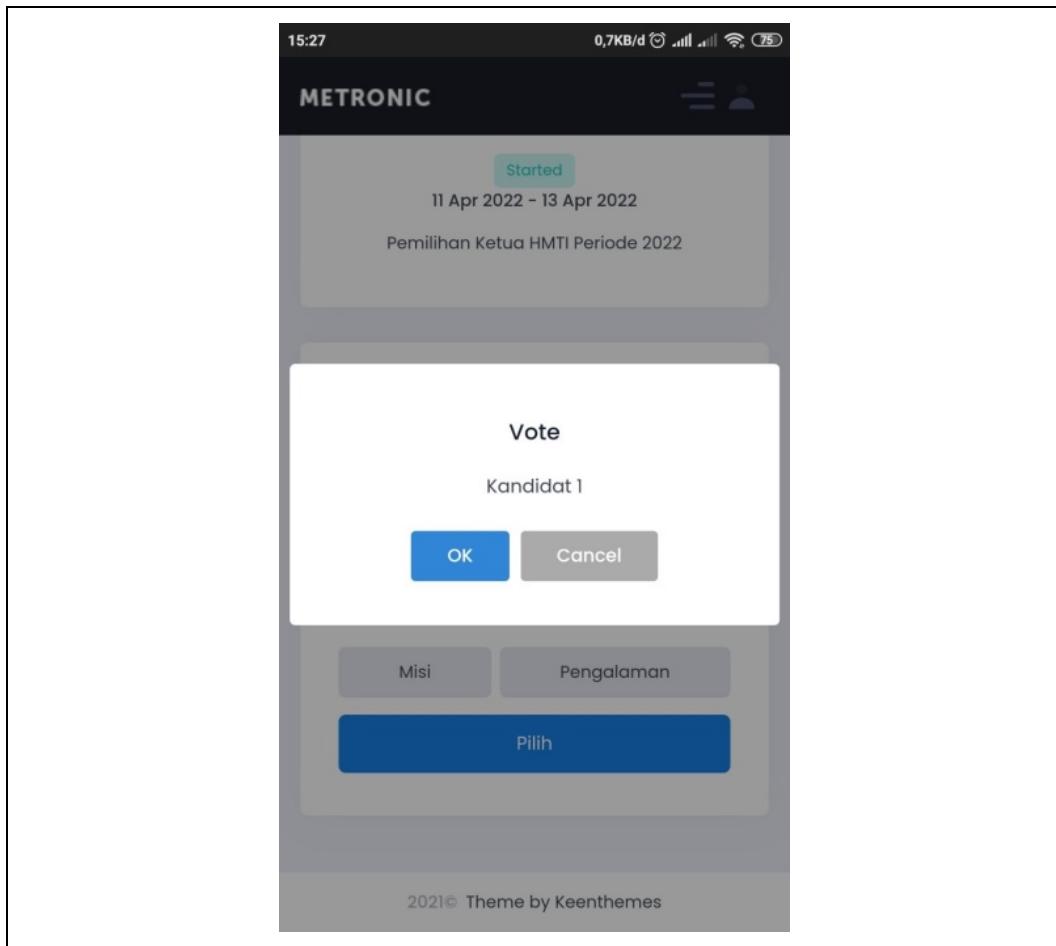
4.4.4 Detail Pemilihan

Halaman detail pemilihan adalah halaman yang menunjukkan informasi detail dari sebuah pemilihan yaitu status pemilihan dan daftar kandidat. Pada halaman ini juga *voter* dapat langsung melakukan *vote* secara langsung pada kandidat yang dipilihnya.



Gambar 4.38 Tampilan Detail Pemilihan

Gambar 4.38 merupakan tampilan detail pemilihan yang digunakan untuk melihat informasi detail dari sebuah pemilihan, diantaranya nama pemilihan, status pemilihan, periode pemilihan, deskripsi pemilihan serta daftar kandidat yang terdaftar dalam pemilihan tersebut. Masing-masing kandidat memiliki informasi yang ditampilkan berupa nomor urut, nama kandidat, visi, misi, pengalaman dan tombol yang berfungsi bagi *voter* untuk memilih kandidat tersebut.

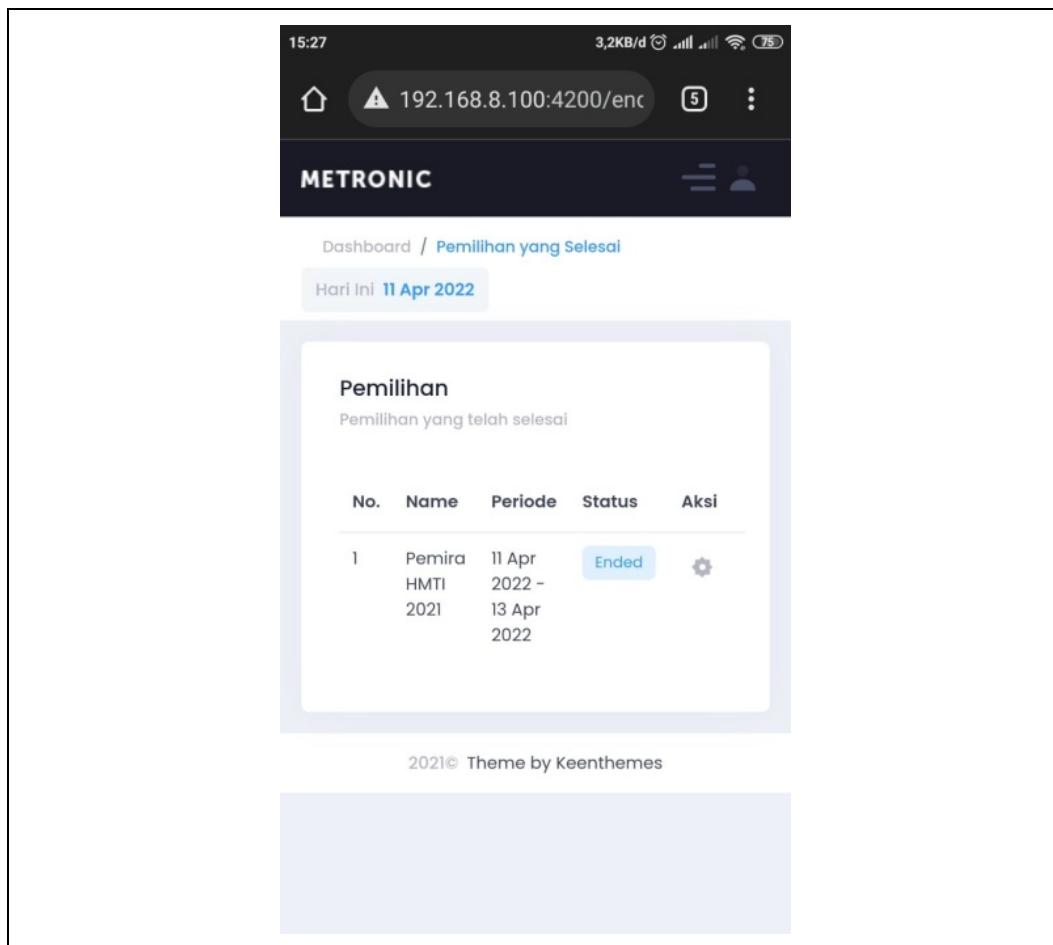


Gambar 4.39 Tampilan Vote

Gambar 4.39 merupakan tampilan yang muncul ketika *voter* melakukan aksi pada tombol Pilih untuk memilih seorang kandidat. Terdapat *popup* yang berisi nama kandidat yang dipilih, tombol untuk melakukan konfirmasi dan tombol untuk membatalkan pilihan.

4.4.5 Ended Election

Halaman *ended election* adalah halaman yang menampilkan seluruh daftar pemilihan yang terdapat dalam aplikasi simvoni dengan periode pemilihan yang telah selesai.

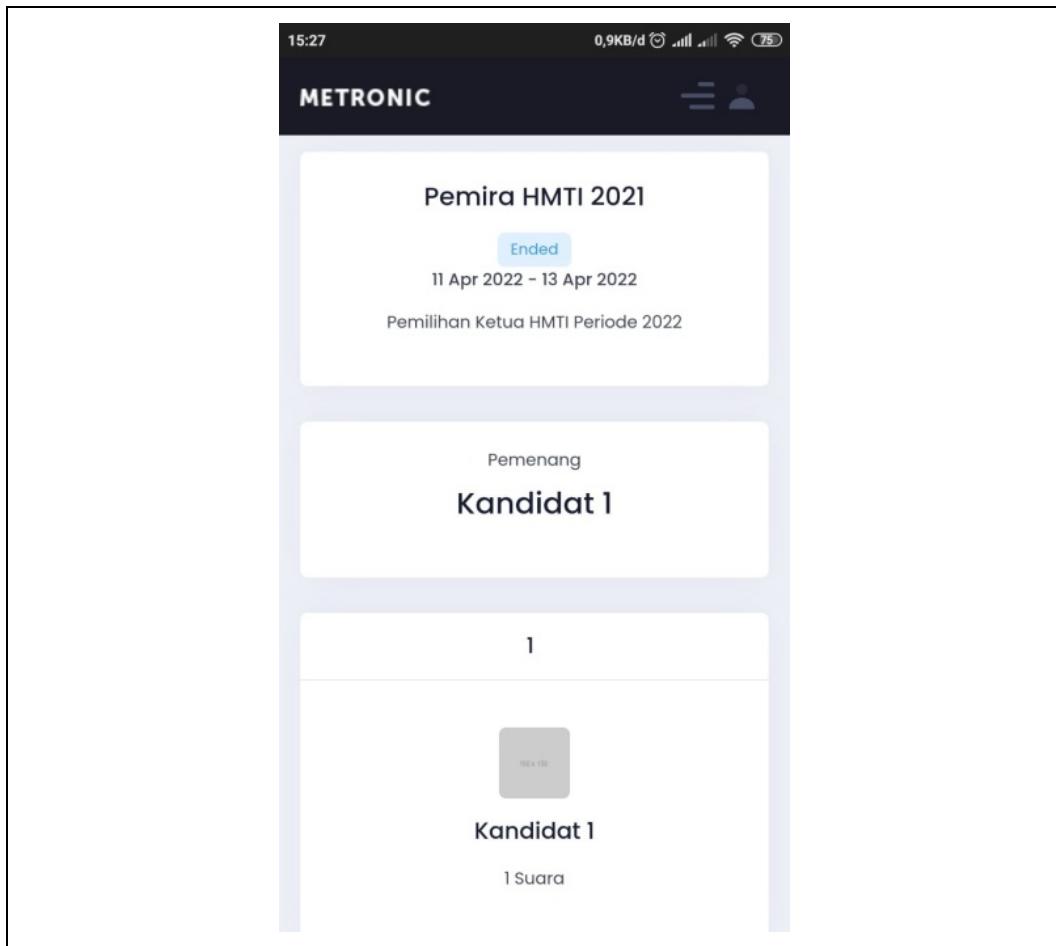


Gambar 4.40 Tampilan Ended Election

Gambar 4.40 merupakan tampilan *ended election* yang menampilkan daftar pemilihan yang telah selesai. Setiap pemilihan memiliki empat komponen yaitu nama pemilihan, periode pemilihan, status pemilihan yang sudah Ended dan tombol untuk melihat detail pemilihan tersebut.

4.4.6 Hasil

Halaman hasil merupakan halaman yang ditampilkan ketika voter memilih tombol detail dari halaman *ended election* yang menunjukkan hasil dari sebuah pemilihan.



Gambar 4.41 Tampilan Hasil Pemilihan

Gambar 4.41 merupakan tampilan hasil dari sebuah pemilihan yang telah selesai. Informasi yang ditampilkan adalah nama kandidat yang terpilih menjadi pemenang dengan masing-masing jumlah suara yang diperoleh setiap kandidat.

4.5 Implementasi Rancangan API

Implementasi rancangan *API* merupakan implementasi spesifikasi *request response* ketika melakukan komunikasi dengan aplikasi *backend*. Semua proses bisnis sistem voting elektronik termasuk penyimpanan data ke *database* dan komunikasi dengan *node blockchain* dilakukan oleh aplikasi *backend*.

4.5.1 Membuat Pemilihan

Sebuah pemilihan dibuat dengan cara melakukan *http request* dengan metode `POST` ke *endpoint* `/election-authority/election`. *Request body* yang dibutuhkan adalah `name`, `description`, `start` dan `end`.

```
curl -s -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmtZSI6InB1
bwlyYS1obXRpIiwicm9sZSI6ImVsZWN0aW9uX2F1dGhvcmloeSIsImlhhdCI6MTYzNDk20TkwMCwiZXhwIj
xNjM0OTc3MTAwfQ.2sm8jVB3WMPRSX5PvntmRy06joGbzmZlKMy1Zm6cJxk" \
-d '{
    "name": "Pemira HMTI 2020",
    "description": "Pemilihan Ketua HMTI",
    "start": "2020-02-01",
    "end": "2020-02-14"
}' \
http://localhost:3000/election-authority/election | jq -M
```

Gambar 4.42 *HTTP Request* untuk Membuat Pemilihan

Gambar 4.42 menunjukkan *http request* yang digunakan untuk membuat pemilihan Pemira HMTI 2020. *Format* tanggal yang digunakan pada *field start* dan *end* adalah *date string* yang disimpan dalam *database* yaitu YYYY-MM-DD. *Token* yang digunakan pada *header Authorization* berasal dari *credentials* yang dimiliki oleh *Election Authority* yaitu Pemira HMTI.

```
{
  "message": "Sukses Membuat Pemilihan",
  "data": {
    "id": 16,
    "name": "Pemira HMTI 2020",
    "description": "Pemilihan Ketua HMTI",
    "start": "2020-02-01",
    "end": "2020-02-14",
    "status": "draft",
    "ea": "pemira-hmti"
  }
}
```

Gambar 4.43 *HTTP Response* Pembuatan Pemilihan

Gambar 4.43 menunjukkan *http response* yang diberikan ketika sukses membuat sebuah pemilihan. Pesan sukses yang diberikan adalah Sukses Membuat Pemilihan. Data yang ditampilkan memiliki tambahan beberapa *field* dari *request* yaitu *id*, *status* dan *ea*. Yang dimaksud *field ea* adalah *Election Authority* yang membuat pemilihan tersebut.

4.5.2 Menambahkan Kandidat

Kandidat suatu pemilihan ditambahkan dengan cara melakukan *http request* dengan metode `POST` ke *endpoint* `/election-authority/add-candidate/:electionId`. Bagian `:electionId` adalah nilai dinamis sehingga disesuaikan dengan `id` dari pemilihan yang akan ditambahkan kandidat.

```
curl -s -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InB
bwLyYSlobXRpIiwicm9sZSI6ImVsZWN0aW9uX2F1dGhvcmloeSIsImhdCI6MTYzMjNkZ0TkWMCwiZXhwIj
xNjM0OTc3MTAwfQ.2sm8jVB3WMPRSX5PvntmRy06joGbzmZlKMy1Zm6cJxk" \
-d '{
  "name": "Kandidat",
  "visi": "Visi",
  "misi": [
    "misi",
    "misi"
  ],
  "pengalaman": [
    "pengalaman",
    "pengalaman"
  ]
}' \
http://localhost:3000/election-authority/add-candidate/16 | jq -M
```

Gambar 4.44 *HTTP Request* untuk Menambahkan Kandidat

Gambar 4.44 menunjukkan *http request* yang digunakan untuk menambahkan kandidat pada pemilihan dengan `id` 16. Karena hubungan antara kandidat dengan misi dan pengalaman bersifat *one-to-many* sehingga nilai dari *field* `misi` dan `pengalaman` adalah *array of string*.

```
{
  "message": "Sukses Menambahkan Kandidat",
  "data": {
    "name": "Kandidat",
    "visi": "Visi",
    "nameSlug": "kandidat",
    "election": {
      "id": 16,
      "name": "Pemira HMTI 2020",
      "description": "Pemilihan Ketua HMTI",
      "start": "2020-02-01",
      "end": "2020-02-14",
      "contractAddress": null
    },
    "id": 13
  }
}
```

Gambar 4.45 *HTTP Response* Penambahan Kandidat

Gambar 4.45 menunjukkan *http response* yang diberikan ketika sukses menambahkan kandidat pada sebuah pemilihan. Pesan sukses yang diberikan adalah `Sukses Menambahkan Kandidat`. *Field* `nameSlug` adalah nama kandidat yang diubah menjadi kebab-case. Hal ini diperlukan untuk identifikasi kandidat pada *smart contract* nantinya.

4.5.3 Deploy Pemilihan

Sebuah *smart contract* bisa digunakan apabila sudah di-*deploy* ke jaringan Ethereum untuk mendapatkan *address*. *Address* digunakan untuk mengidentifikasi setiap *smart contract* yang terdapat pada jaringan Ethereum. Jadi walaupun *source code smart contract* sama akan menghasilkan *address* yang berbeda setiap *deployment*. Sebuah pemilihan di-*deploy* ke dengan cara melakukan *http request* dengan metode `POST` ke *endpoint* `/super-admin/deploy-election/:electionId`. Bagian `:electionId` adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang akan di-*deploy*.

```
curl -s -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmtZSI6InN
cGVyLWFkbWluIiwicm9sZSI6InN1cGVyX2FkbWluIiwiawF0IjoxNjM0OTY50DUzLCJleHAiOjE2MzQ5Nz
wNTN9.hRK055tWAvrqD0gIwGrVUHPZxdSBmGlghPr5CEzRzCQ" \
http://localhost:3000/super-admin/deploy-election/16 | jq -M
```

Gambar 4.46 HTTP Request Deploy Pemilihan

Gambar 4.46 menunjukkan *http request* yang digunakan untuk melakukan *deployment smart contract* pemilihan ke jaringan Ethereum. *Token* yang digunakan pada *header Authorization* berasal dari *credentials Super Admin* karena hanya *Super Admin* yang berhak melakukan *deployment*.

```
{
  "message": "Sukses Deploy Pemilihan",
  "data": {
    "address": "0x0fF3E41fDB3b5472b852444DB9f307D7950f030E"
  }
}
```

Gambar 4.47 HTTP Response Deploy Pemilihan

Gambar 4.47 menunjukkan *http response* yang diberikan ketika sukses melakukan *deployment smart contract* ke jaringan Ethereum. Pesan Sukses yang diberikan adalah `Sukses Deploy Pemilihan`. *Field address* adalah *address* dari *smart contract* untuk pemilihan yang di-deploy.

4.5.4 Mengikuti Pemilihan

User dengan *role Voter* dapat ditambahkan sebagai pemilih dalam sebuah pemilihan dengan cara melakukan *http request* dengan metode `POST` ke *endpoint* `/voter/join/:electionId`. Bagian `:electionId` adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang akan diikuti.

```
curl -s -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmtZSI6InZdGVyLXNhdHUiLCJyb2xlIjoidm96ZXIiLCJpYXQiOjE2MzQ5Njk5MTUsImV4cCI6MTYzNDk3NzExNX0.L6FcwCtcBZm5b2BXQ0b_9xqTl2mNAAE6w1_0X-rrtY" \
http://localhost:3000/voter/join/16 | jq -M
```

Gambar 4.48 *HTTP Request* untuk Mengikuti Pemilihan

Gambar 4.48 menunjukkan *http request* yang digunakan untuk mengikuti sebuah pemilihan dengan `id` 16. *Token* yang digunakan pada *header Authorization* berasal dari *credentials Voter*.

```
{
  "message": "Sukses Mengikuti Pemilihan",
  "data": {
    "id": 16,
    "name": "Pemira HMTI 2020",
    "description": "Pemilihan Ketua HMTI",
    "start": "2020-02-01",
    "end": "2020-02-14",
    "status": "deployed",
    "ea": "Pemira HMTI"
  }
}
```

Gambar 4.49 *HTTP Response* Mengikuti Pemilihan

Gambar 4.49 menunjukkan *http response* yang diberikan ketika sukses mengikuti sebuah pemilihan. Pesan sukses yang diberikan adalah `Sukses Mengikuti Pemilihan.`

4.5.5 Daftar Peserta Pemilihan

User dengan *role Election Authority* dapat melihat daftar peserta yang mengikuti sebuah pemilihan dengan cara melakukan *http request* dengan metode GET ke *endpoint* `/election-authority/election-participant/:electionId`. Bagian `:electionId` adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang akan dibutuhkan.

```
curl -s -X GET \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6InB
bWlyYSlobXRpIiwicm9sZSI6ImVsZWNoaW9uX2F1dGhvcmloeseSISImLhdCI6MTYzMjNk20TkwcwizXhwIJ
xNjM0OTc3MTAwfQ.2sm8jVB3WMPSRX5PvntmRy06joGbzmZlKMy1Zm6cJxk" \
http://localhost:3000/election-authority/election-participant/16 | jq -M
```

Gambar 4. 50 *HTTP Request* Daftar Peserta Pemilihan

Gambar 4.50 menunjukkan *http request* yang digunakan untuk menampilkan daftar peserta yang mengikuti sebuah pemilihan dengan `id` 16. *Token* yang digunakan pada *header* `Authorization` adalah *credentials Election Authority*.

```
{
  "message": "Sukses",
  "data": {
    "electionId": 16,
    "electionName": "Pemira HMTI 2020",
    "participant": [
      {
        "participationId": 7,
        "userId": 23,
        "username": "voter-satu",
        "status": "waiting_approval"
      }
    ]
  }
}
```

Gambar 4. 51 *HTTP Response* Daftar Peserta Pemilihan

Gambar 4.51 menunjukkan *http response* yang diberikan ketika sukses menampilkan daftar peserta yang mengikuti pemilihan. Karena sebuah pemilihan dapat diikuti oleh banyak *user* maka tipe data dari *field participant* adalah array of object.

4.5.6 Detail Pemilihan

User dengan *role Voter* dapat melihat detail dari sebuah pemilihan dengan cara melakukan *http request* dengan metode GET ke *endpoint* /voter/election-detail/:electionId. Bagian :electionId adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang dibutuhkan.

```
curl -s -X GET \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmtZSI6InZ
dGVyLXNhdHUiLCJyb2xlIjoidm90ZXIiLCJpYXQiOjE2MzQ5NzgwMTAsImV4cCI6MTYzNDk4NTIxMH0.0Z
pz61zoTp1kZtQJ6KP8djnvI-qCR0mGlqUKl6AWY4" \
http://localhost:3000/voter/election-detail/16 | jq -M
```

Gambar 4.52 *HTTP Request* Detail Pemilihan

Gambar 4.52 menunjukkan *http request* yang digunakan untuk mendapatkan detail sebuah pemilihan dengan id 16. *Token* yang digunakan pada *header Authorization* adalah *credentials Voter*.

```
{
  "message": "Sukses",
  "data": {
    "id": 16,
    "name": "Pemira HMTI 2020",
    "description": "Pemilihan Ketua HMTI",
    "start": "2020-02-01",
    "end": "2020-02-14",
    "status": "deployed",
    "ea": "Pemira HMTI",
    "participation status": "waiting_approval",
    "candidates": [
      {
        "id": 13,
        "name": "Kandidat",
        "visi": "Visi",
        "misi": [
          "misi",
          "misi"
        ],
        "pengalaman": [
          "pengalaman",
          "pengalaman"
        ]
      }
    ]
  }
}
```

Gambar 4.53 *HTTP Response* Detail Pemilihan

Gambar 4.53 menunjukkan *http response* yang diberikan ketika sukses mendapatkan detail dari sebuah pemilihan. Karena sebuah pemilihan terdapat lebih dari satu kandidat maka *field candidates* memiliki tipe data *array of object*.

4.5.7 Memulai Pemilihan

User dengan *role Election Authority* dapat memulai sebuah pemilihan yang telah di-deploy ke jaringan Ethereum dengan cara melakukan *http request* dengan metode `POST` ke *endpoint* `/election-authority/start-election/:electionId`. Bagian `:electionId` adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang akan dimulai.

```
curl -s -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmtZSI6InB
bwLyYSlobXRpIiwi cm9sZSI6ImVsZWNoaW9uX2F1dGhvcmloeSIsImhdCI6MTYzMdk3Nzk5NSwiZXhwIj
xNjM00Tg1MTk1fQ.zylx02syC0INyLGwE3q94XP3qcg62HcTv_c5u4YbxBY" \
http://localhost:3000/election-authority/start-election/16 | jq -M
```

Gambar 4.54 *HTTP Request* Memulai Pemilihan

Gambar 4.54 menunjukkan *http request* yang digunakan untuk memulai sebuah pemilihan dengan `id` 16. *Token* yang digunakan pada *header Authorization* adalah *credentials Election Authority*.

```
{
  "message": "Sukses Memulai Pemilihan",
  "data": {
    "election": "Pemira HMTI 2020"
  }
}
```

Gambar 4.55 *HTTP Response* Memulai Pemilihan

Gambar 4.55 menunjukkan *http response* yang diberikan ketika sukses memulai sebuah pemilihan. Data yang diberikan adalah pesan sukses yaitu Sukses Memulai Pemilihan dan nama pemilihan yang berhasil dimulai yaitu Pemira HMTI 2020.

4.5.8 Vote

User dengan role Voter dapat memilih seorang kandidat pada sebuah pemilihan dengan cara melakukan *http request* dengan metode POST ke endpoint /voter/vote.

```
curl -s -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmtZSI6InZdGVyLXNhdHUiLCJyb2xlIjoidm90ZXIiLCJpYXQiOjE2MzQ5NzgwMTAsImV4cCI6MTYzMjQ4NTIxMH0.0Zpz61zoTp1kZtQJ6KP8djnvI-qCR0mGlqUKl6AWY4" \
-d '{
    "election_id": 16,
    "candidate_id": 13
}' \
http://localhost:3000/voter/vote | jq -M
```

Gambar 4. 56 *HTTP Request* Pemilihan Kandidat

Gambar 4.56 menunjukkan *http request* yang digunakan untuk memilih seorang kandidat. *Request body* yang digunakan ada 2 yaitu `election_id` dan `candidate_id`. *Field* `election_id` berisi nilai `id` dari sebuah pemilihan dan *field* `candidate_id` berisi nilai `id` dari kandidat yang dipilih. *Token* yang digunakan pada *header* `Authorization` adalah *credentials* `Voter`.

```
{
  "message": "Sukses Memberikan Suara",
  "data": {
    "election": "Pemira HMTI 2020"
  }
}
```

Gambar 4. 57 *HTTP Response* Pemilihan Kandidat

Gambar 4.57 menunjukkan *http response* pemilihan kandidat pada sebuah pemilihan. Data yang diberikan adalah pesan sukses yaitu Sukses Memberikan Suara serta nama pemilihan yang diikuti yaitu Pemira HMTI 2020.

4.5.9 Menghentikan Pemilihan

User dengan *role Election Authority* dapat menghentikan pemilihan yang telah dimulai dengan cara melakukan *http request* dengan metode `POST` ke *endpoint* `/election-authority/end-election/:electionId`. Bagian `:electionId` adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang akan dihentikan.

```
curl -s -X POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmtZSI6InB
bwlyYSlobXRpIiwicm9sZSI6ImVsZWN0aW9uX2F1dGhvcmloSISImLhdCI6MTYzMdk3Nzk5NSwiZXhwIj
xNjM0OTg1MTk1fQ.zylx02syC0INyLGwE3q94XP3qcg62HcTv_c5u4YbxBY" \
http://localhost:3000/election-authority/end-election/16 | jq -M
```

Gambar 4. 58 *HTTP Request* Menghentikan Pemilihan

Gambar 4.58 menunjukkan *http request* yang digunakan untuk menghentikan sebuah pemilihan dengan `id` 16. *Token* yang digunakan pada *header Authorization* adalah *credentials Election Authority*.

```
{
  "message": "Sukses Menghentikan Pemilihan",
  "data": {
    "election": "Pemira HMTI 2020"
  }
}
```

Gambar 4. 59 *HTTP Response* Menghentikan Pemilihan

Gambar 4.59 menunjukkan *http response* yang diberikan ketika sukses menghentikan sebuah pemilihan. Data yang diberikan adalah pesan sukses yaitu Sukses Menghentikan Pemilihan serta nama pemilihan yang dihentikan yaitu Pemira HMTI 2020.

4.5.10 Melihat Hasil Pemilihan

Hasil sebuah pemilihan dapat dilihat dengan cara melakukan *http request* dengan metode `GET` ke *endpoint* `/voter/ended-election-detail/:electionId`. Bagian `:electionId` adalah nilai dinamis sehingga disesuaikan dengan pemilihan yang dibutuhkan.

```
curl -s -X GET \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmtZSI6InZv
dGVyLXNhdHUiLCJyb2xlIjoidm90ZXIiLCJpYXQiOjE2MzQ5NzgwMTAsImV4cCI6MTYzNDk4NTIxMH0.0ZI
pz61zoTp1kZtQJ6KP8djnvI-qCR0mGlqUKl6AWY4" \
http://localhost:3000/voter/ended-election-detail/16 | jq -M
```

Gambar 4.60 *HTTP Request* Hasil Pemilihan

Gambar 4.60 menunjukkan *http request* yang digunakan melihat hasil pemilihan dengan *id* 16. *Token* yang digunakan pada *header Authorization* adalah *credentials Voter*.

```
{
  "message": "Sukses",
  "data": {
    "id": 16,
    "name": "Pemira HMTI 2020",
    "description": "Pemilihan Ketua HMTI",
    "start": "2020-02-01",
    "end": "2020-02-14",
    "status": "ended",
    "ea": "Pemira HMTI",
    "winner": "Kandidat",
    "candidates": [
      {
        "id": 13,
        "name": "Kandidat",
        "visi": "Visi",
        "vote_count": 1,
        "misi": [
          "misi",
          "misi"
        ],
        "pengalaman": [
          "pengalaman",
          "pengalaman"
        ]
      }
    ]
  }
}
```

Gambar 4.61 *HTTP Response* Hasil Pemilihan

Gambar 4.61 menunjukkan *http response* yang diberikan ketika sukses mendapatkan hasil sebuah pemilihan. *Response* yang diberikan mirip seperti

response detail pemilihan. Yang berbeda adalah pada *object candidate* terdapat *field vote_count* yang berisi nilai total suara yang didapatkan oleh seorang kandidat. *Field winner* yang berisi nilai nama dari kandidat yang memenangkan sebuah pemilihan.

4.6 Pengujian Sistem Voting Elektronik

Pengujian Sistem Voting Elektronik menggunakan dua metode yang berbeda yaitu pengujian otomatis dan pengujian manual. Pengujian otomatis adalah pengujian yang menggunakan *automation tools* untuk membandingkan hasil yang diharapkan dengan hasil yang diperoleh dalam suatu proses. Karena penelitian ini berfokus pada penggunaan *smart contract* maka pengujian dibatasi pada *backend* dan *node Ethereum*.

4.6.1 Pengujian API Aplikasi

Test runner adalah *automation tools* yang digunakan untuk menjalankan skenario pengujian yang telah disiapkan. Pada penelitian ini *test runner* yang digunakan adalah *Jest*. Pengujian dibagi menjadi 3 buah *file* yang masing-masing menguji *endpoint* untuk *Super Admin*, *Election Authority* dan *Voter*.

```
simvoni-nestjs/jest/test development x
▶ tree
.
├── election-authority.e2e-spec.ts
├── jest-e2e.json
├── shared.ts
└── super-admin.e2e-spec.ts
    └── voter.e2e-spec.ts

0 directories, 5 files
```

Gambar 4.62 Struktur File Pengujian

Gambar 4.62 menunjukkan struktur *file* yang dibutuhkan *Jest* untuk melakukan pengujian *endpoint*. *File shared.ts* digunakan untuk menyimpan nilai

constant yang dibutuhkan untuk pengujian. File `jest-e2e.json` adalah konfigurasi *Jest* menjalankan pengujian. File `super-admin.e2e-spec.ts` adalah pengujian untuk *endpoint Super Admin*. File `election-authority.e2e-spec.ts` adalah pengujian untuk *endpoint Election Authority*. Dan file `voter.e2e-spec.ts` adalah pengujian untuk *endpoint Voter*.

4.4.1.1 Pengujian API Super Admin

Pengujian manipulasi data yang dilakukan adalah melakukan *http request* ke *endpoint* yang hanya boleh diakses oleh *Super Admin* kemudian mencocokan *http response* yang diperoleh.

Tabel 4.1 Skenario Pengujian *Endpoint Super Admin*

No	Endpoint yang Diuji	Skenario Pengujian	Hasil yang Diharapkan
1.	<code>/super-admin/election-authority</code>	Menggunakan <i>POST request</i> untuk membuat <i>Election Authority</i>	- Data berhasil disimpan di <i>database</i> - Mendapatkan <i>http code 201</i>
2.	<code>/super-admin/election-authority</code>	Menggunakan <i>GET request</i>	- Mendapatkan semua <i>Election Authority</i> - Mendapatkan <i>http code 200</i>
3.	<code>/super-admin/election-authority/:id</code>	Menggunakan <i>GET request</i> dengan <i>id = 2</i>	- Mendapatkan <i>Election Authority</i> dengan <i>id = 2</i> - Mendapatkan <i>http code 200</i>
4.	<code>/super-admin/election-authority/set-wallet-address/:id</code>	Menggunakan <i>POST request</i> untuk menambahkan <i>wallet address</i>	- <i>Address</i> berhasil tersimpan di <i>database</i> - Mendapatkan <i>http code 201</i>
5.	<code>/super-admin/election/ready-to-deploy</code>	Menggunakan <i>GET request</i>	- Mendapatkan Pemilihan yang siap <i>di-deploy</i> - Mendapatkan <i>http code 200</i>
6.	<code>/super-admin/deploy-election/:electionId</code>	Menggunakan <i>POST request</i> untuk <i>deploy</i> pemilihan	- Berhasil <i>deploy</i> pemilihan ke jaringan <i>Ethereum</i> - Mendapatkan <i>http code 201</i>

Pengujian terhadap *endpoint Super Admin* dijalankan dengan cara menjalankan perintah `npm run test:e2e super-admin.e2e-spec.ts` pada *terminal*.

```

> npm run test:e2e super-admin.e2e-spec.ts --color=false
> simvoni-backend@0.0.1 test:e2e
> jest --config ./test/jest-e2e.json "super-admin.e2e-spec.ts" --color=false

PASS test/super-admin.e2e-spec.ts (5.243 s)
  SuperAdminController (e2e)
    POST /super-admin/election-authority
      ✓ Create Election Authority (183 ms)
    GET /super-admin/election-authority
      ✓ Get All Election Authority (11 ms)
    GET /super-admin/election-authority/:id
      ✓ Found EA with id 2 (8 ms)
      ✓ Not found EA with id 3 (12 ms)
    POST /super-admin/election-authority/set-wallet-address/:id
      ✓ Create Address for user with id 1 (413 ms)
    GET /super-admin/election/ready-to-deploy
      ✓ Get ready-to-deploy election (28 ms)
    POST /super-admin/deploy-election/:electionId
      ✓ Deploy election to blockchain (298 ms)

Test Suites: 1 passed, 1 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        5.312 s, estimated 19 s
Ran all test suites matching /super-admin.e2e-spec.ts/i.
Jest did not exit one second after the test run has completed.

```

Gambar 4. 63 Hasil Pengujian *Endpoint Super Admin*

Gambar 4.63 menunjukkan bahwa setiap skenario pengujian yang telah dibuat terhadap *endpoint Super Admin* berhasil dijalankan.

4.4.1.2 Pengujian API *Election Authority*

Pengujian manipulasi data yang dilakukan adalah melakukan *http request* ke *endpoint* yang hanya boleh diakses oleh *Election Authority* kemudian mencocokan *http response* yang diperoleh.

Tabel 4. 2 Skenario Pengujian *Endpoint Election Authority*

No	Endpoint yang Diuji	Skenario Pengujian	Hasil yang Diharapkan
1.	/election-authority/election	Menggunakan <i>POST request</i> untuk membuat pemilihan	<ul style="list-style-type: none"> - Data berhasil disimpan di <i>database</i> - Mendapatkan <i>http code</i> 201
2.	/election-authority/elections	Menggunakan <i>GET request</i>	<ul style="list-style-type: none"> - Mendapatkan semua pemilihan - Mendapatkan <i>http code</i> 200

No	Endpoint yang Diuji	Skenario Pengujian	Hasil yang Diharapkan
3.	/election-authority/election/:id	Menggunakan <i>GET request</i> dengan <i>id</i> = 1	<ul style="list-style-type: none"> - Mendapatkan pemilihan dengan <i>id</i> 1 - Mendapatkan <i>http code</i> 200
4.	/election-authority/add-candidate/:id	Menggunakan <i>POST request</i> untuk menambahkan kandidat	<ul style="list-style-type: none"> - Data berhasil disimpan di <i>database</i> - Mendapatkan <i>http code</i> 201
5.	/election-authority/election/:id/candidates	Menggunakan <i>GET request</i>	<ul style="list-style-type: none"> - Mendapatkan kandidat dari sebuah pemilihan - Mendapatkan <i>http code</i> 200
6.	/election-authority/election/:id/ready	Menggunakan <i>POST request</i> untuk mengubah status pemilihan	<ul style="list-style-type: none"> - Status pemilihan di <i>database</i> berubah - Mendapatkan <i>http code</i> 200
7.	/election-authority/election-participant/accept/:id	Menggunakan <i>POST request</i> untuk menerima pemilih	<ul style="list-style-type: none"> - Status pemilih di <i>database</i> berubah - Mendapatkan <i>http code</i> 201
8.	/election-authority/election-participant/reject/:id	Menggunakan <i>POST request</i> untuk menolak pemilih	<ul style="list-style-type: none"> - Status pemilih di <i>database</i> berubah - Mendapatkan <i>http code</i> 201
9.	/election-authority/start-election/:id	Menggunakan <i>POST request</i> untuk memulai pemilihan	<ul style="list-style-type: none"> - Status pemilihan berubah di <i>database</i> - Mendapatkan <i>http code</i> 201
10.	/election-authority/end-election/:id	Menggunakan <i>POST request</i> untuk menghentikan pemilihan	<ul style="list-style-type: none"> - Status pemilihan berubah di <i>database</i> - Mendapatkan <i>http code</i> 201

Pengujian terhadap *endpoint Election Authority* dijalankan dengan cara menjalankan perintah `npm run test:e2e election-authority.e2e-spec.ts` pada *terminal*.

```

▶ npm run test:e2e election-authority.e2e-spec.ts -- --color=false
> simvoni-backend@0.0.1 test:e2e
> jest --config ./test/jest-e2e.json "election-authority.e2e-spec.ts" "--color=false"

PASS test/election-authority.e2e-spec.ts (10.869 s)
  ElectionAuthorityController (e2e)
    POST /election-authority/election
      ✓ Create Election (89 ms)
    GET /election-authority/elections
      ✓ Get All Election Created by Election Authority (18 ms)
    GET /election-authority/election/:id
      ✓ Get Election By Id 1 (25 ms)
    POST /election-authority/add-candidate/:id
      ✓ Add Candidate to An Election (30 ms)
    GET /election-authority/election/:id/candidates
      ✓ Show All Candidate of An Election (39 ms)
    POST /election-authority/election/:id/ready
      ✓ Set Election Status to ready_to_deploy (42 ms)
    POST /election-authority/election-participant/accept/:id
      ✓ Accept Participant (53 ms)
    POST /election-authority/election-participant/reject/:id
      ✓ Reject Participant (26 ms)
    POST /election-authority/start-election/:id
      ✓ Start Election (231 ms)
    POST /election-authority/end-election/:id
      ✓ End Election (69 ms)

Test Suites: 1 passed, 1 total
Tests:       10 passed, 10 total
Snapshots:   0 total
Time:        10.933 s
Ran all test suites matching /election-authority.e2e-spec.ts/i.
 Jest did not exit one second after the test run has completed.

```

Gambar 4.64 Hasil Pengujian *Endpoint Election Authority*

Gambar 4.64 menunjukkan bahwa setiap skenario pengujian yang telah dibuat terhadap *endpoint Election Authority* berhasil dijalankan.

4.4.1.3 Pengujian API Voter

Pengujian manipulasi data yang dilakukan adalah melakukan *http request* ke *endpoint* yang hanya boleh diakses oleh *Voter* kemudian mencocokan *http response* yang diperoleh.

Tabel 4.3 Skenario Pengujian *Endpoint Voter*

No	Endpoint yang Diuji	Skenario Pengujian	Hasil yang Diharapkan
1.	/voter/join/:id	Menggunakan <i>POST request</i> untuk mengikuti pemilihan	<ul style="list-style-type: none"> - Data partisipasi berhasil disimpan di <i>database</i> - Mendapatkan <i>http code</i> 201

No	Endpoint yang Diuji	Skenario Pengujian	Hasil yang Diharapkan
2.	/voter/election-participation	Menggunakan <i>GET request</i>	<ul style="list-style-type: none"> - Mendapatkan semua partisipasi <i>user</i> terkait - Mendapatkan <i>http code 200</i>
3.	/voter/available-election	Menggunakan <i>GET request</i>	<ul style="list-style-type: none"> - Mendapatkan semua pemilihan yang tersedia - Mendapatkan <i>http code 200</i>
4.	/voter/followed-election	Menggunakan <i>GET request</i>	<ul style="list-style-type: none"> - Mendapatkan semua pemilihan yang diikuti - Mendapatkan <i>http code 200</i>
5.	/voter/ended-election	Menggunakan <i>GET request</i>	<ul style="list-style-type: none"> - Mendapatkan semua pemilihan yang telah selesai - Mendapatkan <i>http code 200</i>
6.	/voter/election-detail/:id	Menggunakan <i>GET request</i>	<ul style="list-style-type: none"> - Mendapatkan detail sebuah pemilihan - Mendapatkan <i>http code 200</i>
7.	/voter/vote	Menggunakan <i>POST request</i> untuk memilih kandidat	<ul style="list-style-type: none"> - Data berhasil tersimpan di <i>database</i> - Mendapatkan <i>http code 201</i>
8.	/voter/ended-election-detail/:id	Menggunakan <i>GET request</i>	<ul style="list-style-type: none"> - Mendapatkan detail pemilihan yang telah selesai - Mendapatkan <i>http code 200</i>

Pengujian terhadap *endpoint Voter* dijalankan dengan cara menjalankan perintah `npm run test:e2e voter.e2e-spec.ts` pada *terminal*.

```

▶ npm run test:e2e voter.e2e-spec.ts -- --color=false

> simvoni-backend@0.0.1 test:e2e
> jest --config ./test/jest-e2e.json "voter.e2e-spec.ts" "--color=false

PASS test/voter.e2e-spec.ts (11.172 s)
  VoterController (e2e)
    POST /voter/join/:id
      ✓ Join Election (87 ms)
    GET /voter/election-participation
      ✓ Get Participation (21 ms)
    GET /voter/available-election
      ✓ Get Available Election (26 ms)
    GET /voter/followed-election
      ✓ Get Followed Election (17 ms)
    GET /voter/ended-election
      ✓ Get Ended Election (8 ms)
    GET /voter/election-detail/:id
      ✓ Get Election Detail (17 ms)
    POST /voter/vote
      ✓ Vote on Election (300 ms)
    GET /voter/ended-election-detail/:id
      ✓ Get Ended Election Detail (393 ms)

Test Suites: 1 passed, 1 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        11.231 s
Ran all test suites matching /voter.e2e-spec.ts/i.
Jest did not exit one second after the test run has completed.

```

Gambar 4.65 Hasil Pengujian *Endpoint Voter*

Gambar 4.65 menunjukkan bahwa setiap skenario pengujian yang telah dibuat terhadap *endpoint Voter* berhasil dijalankan.

4.6.2 Pengujian Integritas Data Blockchain

Penelitian ini menggunakan Openethereum sebagai Ethereum *client* sehingga pengujian ini akan melakukan percobaan untuk mengubah data yang tersimpan pada *database* yang digunakan oleh Openethereum. Berdasarkan repositori Openethereum, *database* yang digunakan adalah RocksDB. RocksDB adalah sebuah *database* dengan paradigma *key-value* yang mengutamakan *performance* sehingga hampir tidak ada *client GUI* yang dapat digunakan untuk melakukan interaksi dengan *database* tersebut. Pada pengujian ini ada beberapa hal yang perlu dipersiapkan.

4.5.2.1 *Node Openethereum*

Node Openethereum yang dibutuhkan adalah *node* yang memiliki konfigurasi semirip mungkin dengan *node* Openethereum yang digunakan untuk Sistem Voting Elektronik namun dengan langkah persiapan yang lebih mudah. Jumlah *validator* yang dibutuhkan adalah 1 dan dijalankan pada *localhost*.

4.5.2.2 *Block Explorer*

Block Explorer digunakan untuk memberikan gambaran bagaimana kondisi *block* yang telah terbentuk dalam *blockchain*. Repositori untuk projek ini terdapat pada <https://github.com/cekingx/explorer.git>. Langkah-langkah yang perlu dijalankan adalah *clone repository* ke *local*. Kemudian mengubah variabel *remoteUrl* pada *file app.js* menjadi `http://localhost:8540`. Kemudian jalankan perintah `npm install`. Setelah selesai melakukan instalasi, jalankan perintah `npm start` dan buka *url* <http://localhost:8000> pada *browser*.

4.5.2.3 *Ethereum Decoder*

Ketika melakukan pemanggilan fungsi sebuah *smart contract*, hal yang tersimpan didalam *blockchain* adalah *function signature* dari fungsi yang dipanggil dan disimpan dalam bentuk *hex* sehingga diperlukan *decoder* untuk bisa membaca *hex* tersebut. Ethereum *decoder* yang akan digunakan dapat diakses pada *url* <https://eth-decoder.cekingx.com>.

4.5.2.4 *Smart Contract*

Smart Contract yang digunakan adalah *smart contract* dengan logika yang sederhana dimana hanya terdapat dua fungsi yang bisa digunakan yaitu `getPerson` dan `setPerson`.

```
contract OneTime {
    bool private isLocked;
    string private person;

    constructor() {
        isLocked = false;
        person = "";
    }
}
```

```

}

modifier unlocked() {
    require(isLocked == false, "Contract Locked");
    _;
}

function getPerson() public view returns (string memory) {
    return person;
}

function setPerson(string memory _person) public unlocked {
    person = _person;
    isLocked = true;
}
}

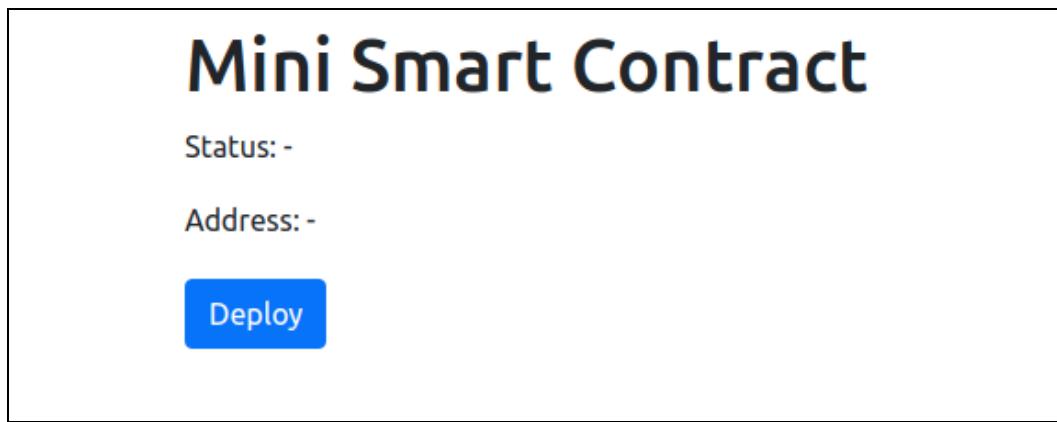
```

Kode Program 4.9 Smart Contract

Kode Program 4.9 merupakan *source code* dari *smart contract* yang digunakan pada pengujian ini. *Source code* tersebut hanya memiliki dua buah *function* yaitu `getPerson` dan `setPerson`.

4.5.2.5 Tampilan

Tampilan tidak terlalu dibutuhkan karena untuk berinteraksi dengan *ethereum client* hanya membutuhkan *library javascript ethers.js*. Menggunakan *runtime javascript* seperti *node* sudah bisa berinteraksi dengan *ethereum client* namun untuk memudahkan penjelasan pengujian ini akan menggunakan *tech stack* yang umum yaitu *html, css* dan *javascript*.



Gambar 4.66 Tampilan

Gambar 4.66 menunjukkan tampilan yang akan digunakan untuk berinteraksi dengan *smart contract* yang terdapat pada *blockchain*.

4.5.2.6 *Rlpdump*

Rlpdump adalah *tools* yang dapat digunakan untuk melakukan *encoding* maupun *decoding* struktur data *rlp*. *RLP*(*Recursive Length Prefix*) adalah standar digunakan oleh *ethereum* untuk menyimpan struktur data *array* dalam sebuah *hex*.

```
rlpdump --help
Usage: rlpdump [-noascii] [-hex <data>][-reverse] [filename]
-hex string
    dump given hex data
-noascii
    don't print ASCII strings readably
-reverse
    convert ASCII to rlp
-single
    print only the first element, discard the rest

Dumps RLP data from the given file in readable form.
If the filename is omitted, data is read from stdin.
```

Gambar 4.67 *Rlpdump Help*

Gambar 4.67 menunjukkan perintah yang disediakan serta deskripsi singkat tentang apa saja yang bisa dilakukan oleh *rlpdump*.

4.5.2.7 *Ldb*

Ldb dideskripsikan sebagai *RocksDB Administrative Tools* yaitu *tools CLI* yang dapat melakukan operasi *read* dan *write* ke *database RocksDB*. *Ldb* berperan penting dalam pengujian ini karena *ldb* adalah satu-satunya cara untuk berinteraksi dengan data yang tersimpan pada *database RocksDB*.

```

ldb - RocksDB Tool
commands MUST specify --db=<full_path_to_db_directory> when necessary
commands can optionally specify --env_uri=<uri_of_environment> or --fs_uri=<uri_of_filesystem> if necessary
The following optional parameters control if keys/values are input/output as hex or as plain strings:
--key_hex : Keys are input/output as hex
--value_hex : Values are input/output as hex
--hex : Both keys and values are input/output as hex

The following optional parameters control the database internals:
--column_family=<string> : name of the column family to operate on. default: default column family
--ttl with 'put','get','scan','dump','query','batchput' : DB supports ttl and value is internally timestamp-suffixed
--try_load_option : Try to load option file from DB.
--disable_consistency_checks : Set options.force_consistency_checks = false.
--ignore_unknown_options : Ignore unknown options when loading option file.
--bloom_bits=<int,e.g.:14>
--fix_prefix_len=<int,e.g.:14>
--compression_type=<no|snappy|zlib|bzip2|lz4|lz4hc|xpress|zstd>
--compression_max_dict_bytes=<int,e.g.:16384>
--block_size=<block_size_in_bytes>
--auto_compaction=<true|false>
--db_write_buffer_size=<int,e.g.:16777216>
--write_buffer_size=<int,e.g.:4194304>
--file_size=<int,e.g.:2097152>

Data Access Commands:
put <key> <value> [--create_if_missing] [--ttl]
get <key> [-ttl]
batchput <key> <value> [<key> <value>] [...] [--create_if_missing] [--ttl]
scan [--from] [--to] [--ttl] [--timestamp] [--max_keys=<N>q] [--start_time=<N>:- is inclusive] [--end_time=<N>:- is exclusive] [--no_value]
delete <key>
deleterange <begin key> <end key>
query [-ttl]
    Starts a REPL shell. Type help for list of available commands.
approxsize [-from] [--to]
checkconsistency
list_file_range_deletes [--max_keys=<N>] : print tombstones in SST files.

```

Gambar 4.68 Ldb Help

Gambar 4.68 menunjukkan perintah yang disediakan serta deskripsi singkat tentang apa saja yang bisa dilakukan oleh *ldb*. Perintah yang dibutuhkan dalam pengujian ini adalah perintah *get* dan perintah *put*.

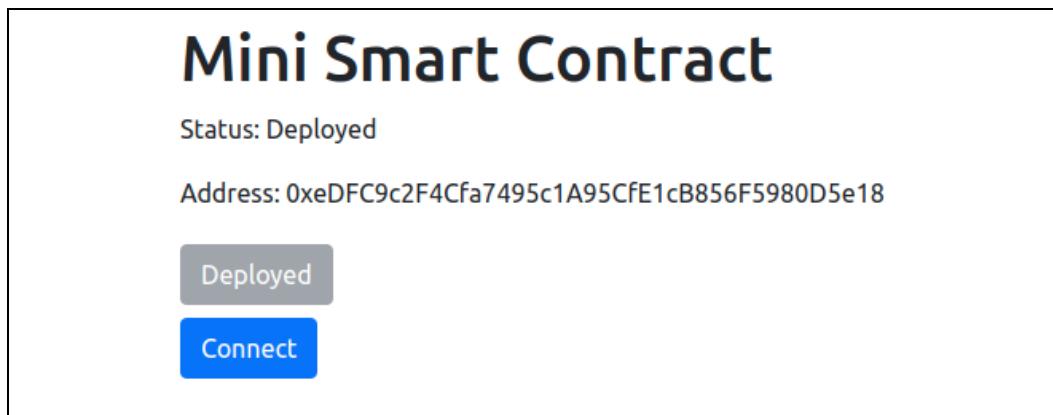
4.5.2.8 Penggunaan *Smart Contract*

Smart Contract yang digunakan untuk pengujian memiliki skenario penggunaan yang sederhana yaitu mendapatkan nama ketika selesai *deploy smart contract*, menentukan nama, mendapatkan nama setelah nama ditentukan dan mencoba kembali untuk menentukan nama.



Gambar 4.69 Tampilan Awal

Gambar 4.69 menunjukkan tampilan awal dari projek yang digunakan untuk melakukan interaksi dengan *smart contract*. Nama projek ini adalah *Mini Smart Contract*. Seperti yang terlihat karena belum dilakukan *deployment*, bagian *status* dan *address* masih kosong.



Gambar 4. 70 Deployed Smart Contract

Gambar 4.70 menunjukkan tampilan ketika *smart contract* telah di-deploy pada jaringan Ethereum yang telah disiapkan. Pada bagian *status* akan menampilkan *Deployed* dan pada bagian *address* akan menampilkan *address* dari *smart contract* tersebut. Untuk bisa menggunakan fungsi yang terdapat pada *smart contract*, *ethers.js* harus dihubungkan dengan *smart contract* tujuan menggunakan *address* yang telah didapatkan.



Gambar 4. 71 Connected Smart Contract

Gambar 4.71 menunjukkan tampilan ketika berhasil terhubung dengan *smart contract* yang diinginkan. Terdapat dua tombol yaitu *Get Person* dan *Set Person*. Tombol *Get Person* akan memanggil fungsi *getPerson* pada *smart contract*. Sedangkan tombol *Set Person* akan memanggil fungsi *setPerson* pada *smart contract* dengan menggunakan nilai dari *input Person Name* sebagai parameter.



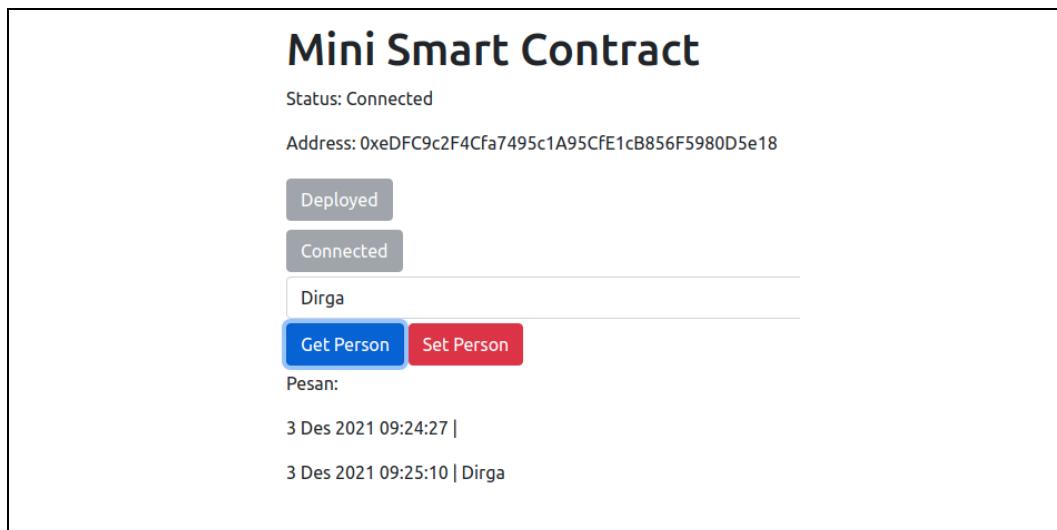
Gambar 4. 72 Tombol Get Person

Gambar 4.72 menunjukkan tampilan ketika tombol *Get Person* digunakan tanpa menentukan nama menggunakan tombol *Set Person*. Hasil yang terlihat adalah pada bagian pesan terdapat *timestamp* tanpa nilai.



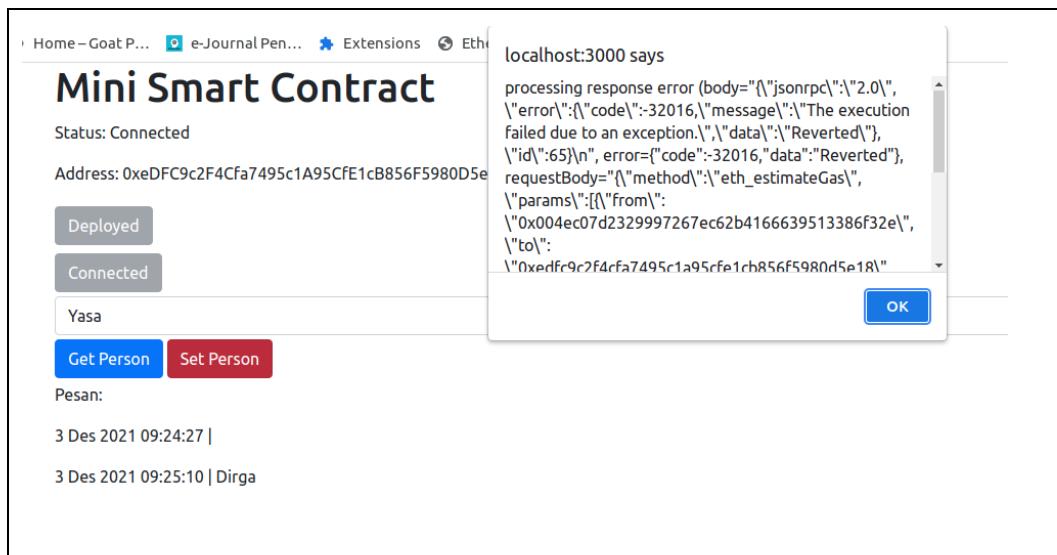
Gambar 4. 73 Tombol Set Person

Gambar 4.73 menunjukkan tampilan ketika menggunakan tombol *Set Person*. Tidak ada *feedback* dari tampilan namun proses yang berjalan adalah *ethers.js* akan mengambil nilai dari *input field* dan menggunakannya sebagai parameter ketika memanggil fungsi *setPerson* pada *smart contract*.



Gambar 4. 74 Tombol Get Person

Gambar 4.67 menunjukkan tampilan ketika menggunakan tombol *Get Person* setelah menentukan nama. Hasil yang terlihat adalah terdapat *timestamp* dengan nilai yang sama seperti apa yang telah ditentukan menggunakan tombol *Set Person*.



Gambar 4. 75 Tombol Set Person

Gambar 4.75 menunjukkan tampilan ketika menggunakan tombol *Set Person* lagi setelah sebelumnya sudah menentukan nama. Hasil yang didapatkan adalah akan ada *alert* dengan pesan *error* karena fungsi *setPerson* pada *smart contract* memang dibuat agar hanya bisa digunakan sekali saja.

4.5.2.9 Pengujian

Smart contract yang digunakan pada pengujian ini memiliki sebuah fungsi untuk melakukan perubahan data yaitu *setPerson*. Namun fungsi tersebut hanya bisa digunakan sekali saja, sehingga pengujian ini akan berusaha mengubah data yang telah tersimpan di dalam *database* Openethereum.

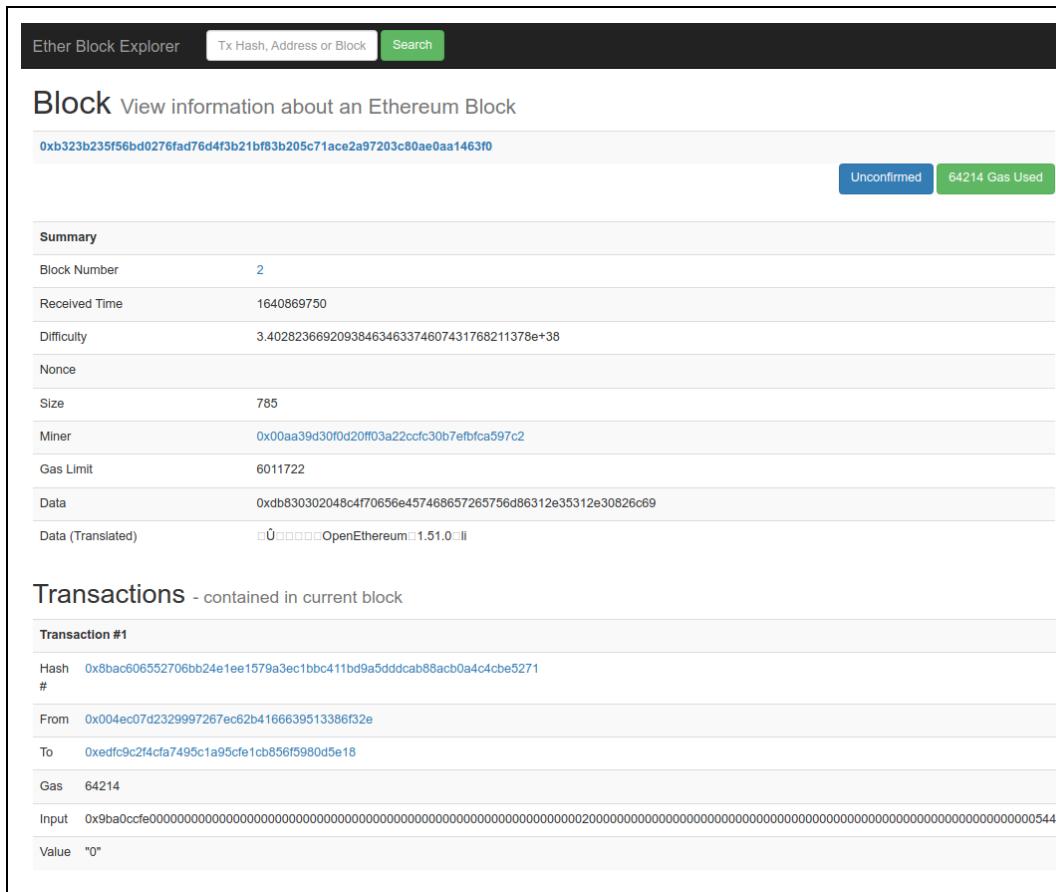


The screenshot shows the Etherparty Block Explorer interface. At the top, there is a search bar with the placeholder "Tx Hash, Address or Bloc" and a green "Search" button. Below the search bar, the text "Welcome to the Etherparty Block Explorer!" is displayed. Underneath this, the text "Latest Block: 2" is shown. A table follows, with columns labeled "Block #", "Tx #", "Size", and "Timestamp". The table contains three rows of data:

Block #	Tx #	Size	Timestamp
2	1	785	1640869750
1	1	2405	1640869369
0	0	533	0

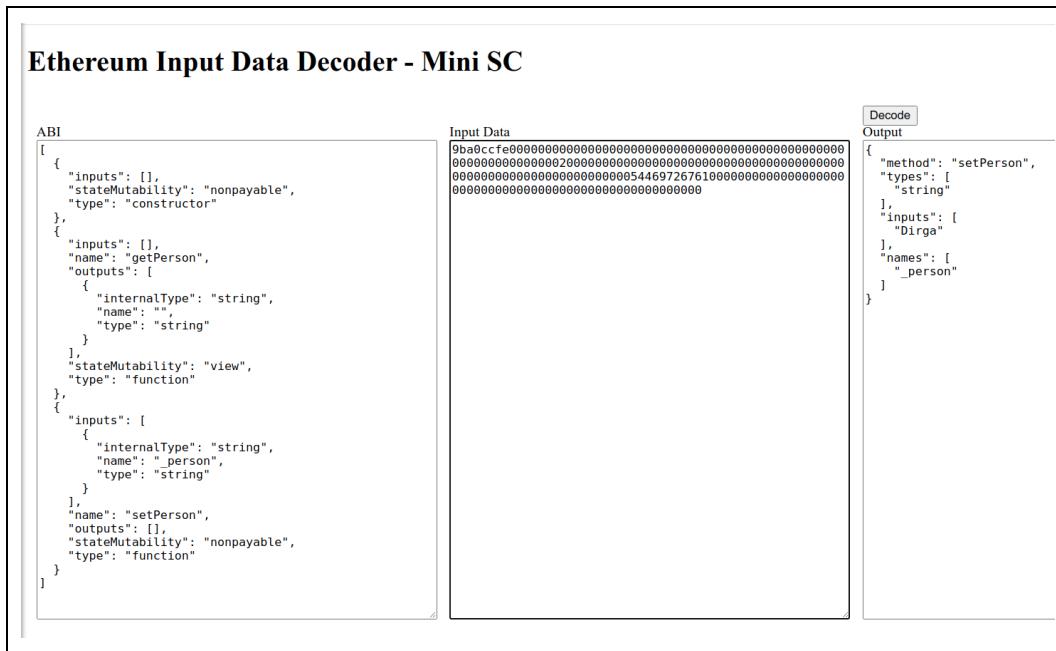
Gambar 4.76 Block Explorer

Gambar 4.76 menunjukkan tampilan dari *block explorer* terhadap jaringan *ethereum* yang telah disiapkan. Bisa terlihat terdapat 2 *block* dimana *block* 1 terdapat data *deploy smart contract* dan *block* 2 tersimpan data pemanggilan fungsi *setPerson* dari proses sebelumnya.



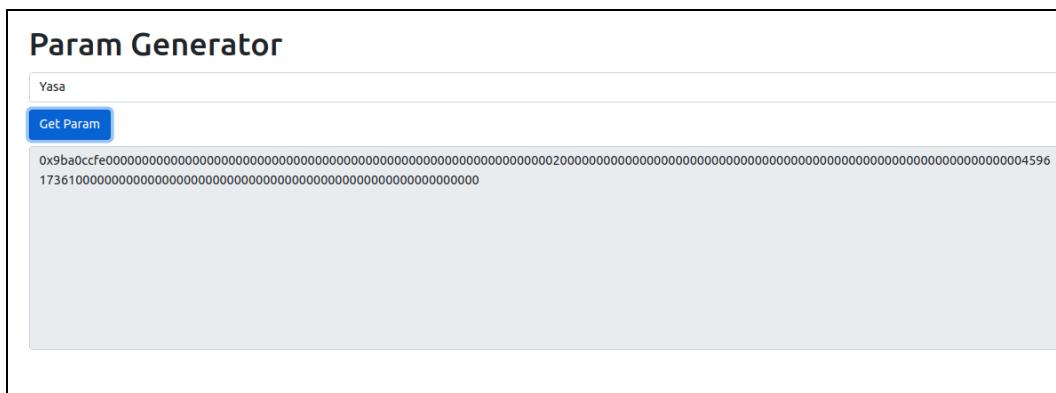
Gambar 4.77 Detail Block 2

Gambar 4.77 menunjukkan tampilan dari detail *block* 2. Informasi penting yang nantinya digunakan adalah *block hash* dengan nilai 0xb323b235f56bd0276fad76d4f3b21bf83b205c71ace2a97203c80ae0aa1463f0 serta nilai *hash* yang terdapat *field input* pada bagian Transaction #1.



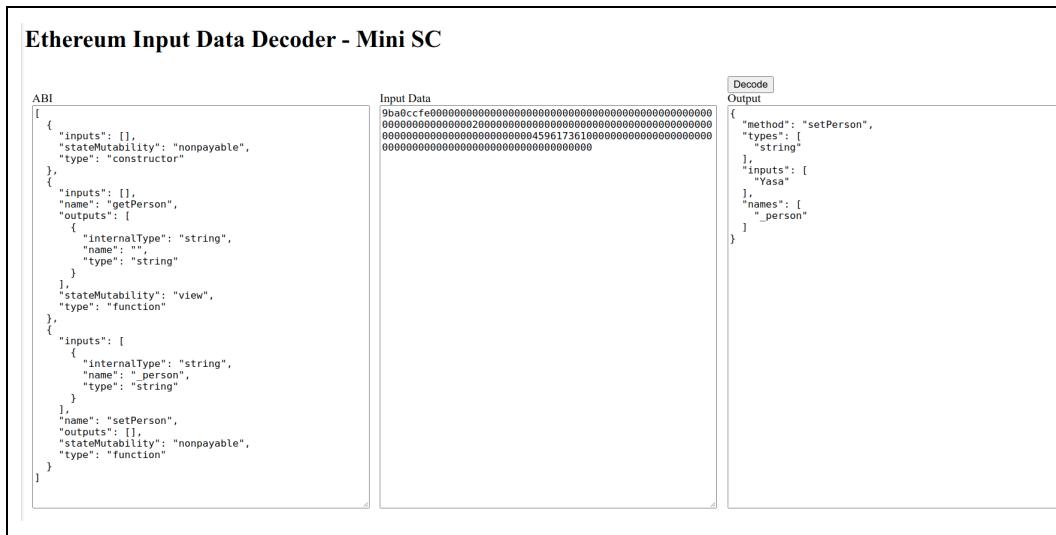
Gambar 4. 78 Ethereum Decoder

Gambar 4.78 menunjukkan tampilan dari hasil *decoding* dari *input parameter* yang tersimpan di dalam *blockchain* menggunakan *ethereum decoder*. Seperti yang ditampilkan, *string parameter* yang tersimpan adalah “Dirga” sesuai dengan proses penggunaan *smart contract*.



Gambar 4.79 Param Generator

Gambar 4.79 menunjukkan tampilan untuk membuat *input parameter* untuk *blockchain*. Pada pengujian ini akan berusaha mengubah *string parameter* yang tersimpan dari “Dirga” menjadi “Yasa”.



Gambar 4.80 Ethereum Decoder

Gambar 4.80 menunjukkan hasil *decoding input parameter* yang dihasilkan oleh *param generator*. Seperti yang ditampilkan, *string parameter* dari *input parameter* tersebut adalah “Yasa”.

Gambar 4.81 *RocksDB Administrative Tools*

Gambar 4.81 menunjukkan *value* yang dipasangkan terhadap sebuah *key* yaitu *hash* dari *block 2*. *Hash* yang ditampilkan memiliki struktur data *rlp* sehingga diperlukan *tools rlpdump* untuk bisa melihat isi di dalamnya.

Gambar 4. 82 Struktur Data *RLP*

Gambar 4.82 menunjukkan *rlp* yang telah di-*decoding* menggunakan *rlpdump*. Hal yang perlu diubah adalah elemen ke-6 pada *array* terkecil pertama menjadi *hash* yang telah diperoleh menggunakan *param generator*.

Gambar 4.83 Encoding RLP

Gambar 4.83 menunjukkan proses untuk mendapatkan struktur *rlp* yang baru. Proses yang dilakukan adalah memasukkan *rlp* yang telah di-*decode* dan elemen ke-6 pada *array* terkecil sudah dimodifikasi disimpan pada *file file.txt*.

Gambar 4.84 Update Database RocksDB

Gambar 4.84 menunjukkan proses *update value* untuk *key* 0xb323b235f56bd0276fad76d4f3b21bf83b205c71ace2a97203c80ae0aa1463f. Seperti yang ditampilkan, hasilnya kita tidak bisa melakukan perubahan data secara langsung ke *database* yang digunakan oleh *node ethereum*. Sehingga ini membuktikan data yang tersimpan pada *blockchain* sangat terjaga integritasnya.

BAB V

PENUTUP

BAB V membahas mengenai kesimpulan yang mengacu pada rumusan masalah dan tujuan penelitian beserta saran yang diberikan peneliti untuk kepentingan pengembangan aplikasi selanjutnya.

5.1 Kesimpulan

Simpulan yang dapat diperoleh dari pengembangan Sistem Voting Elektronik berbasis Ethereum *smart contract* adalah sebagai berikut:

- a. Jaringan Ethereum *private* yang digunakan oleh sistem voting elektronik ini menggunakan Openethereum sebagai Ethereum *client*. Dokumentasi yang dimiliki oleh Openethereum sangat lengkap sehingga pengguna bisa membuat jaringan Ethereum *private* sendiri sesuai dengan kebutuhan.
- b. Sebagian pengujian *backend* sistem voting elektronik dilakukan secara otomatis menggunakan *library* Jest. Implementasi *automated testing* dalam sistem voting elektronik ini bisa mempersingkat proses iterasi penggerjaan sebuah fitur karena pengujian bisa dijalankan dalam hitungan detik.
- c. Uji coba integritas terhadap data yang tersimpan dalam Ethereum *client* menunjukkan bahwa *database* yang digunakan oleh Ethereum *client* terkunci ketika Ethereum *client* dijalankan. Cara lain untuk melakukan manipulasi data adalah dengan cara memanggil fungsi yang terdapat pada *smart contract*. Jika fungsi pada *smart contract* tersebut sudah terkunci maka tidak ada cara lain lagi untuk melakukan manipulasi data.

5.2 Saran

Penelitian sistem e-voting berbasis Ethereum *smart contract* ini masih jauh dari kata sempurna dan masih banyak yang memiliki kekurangan. Namun, Penulis

berharap bahwa penelitian ini dapat menjadi inspirasi dalam pengembangan sistem yang lebih baik lagi kedepannya. Hal ini dikarenakan penelitian terhadap Ethereum *smart contract* saat ini masih jarang ditemui dan minimnya studi literatur yang membahas penelitian yang serupa. Namun, hal ini justru dapat dijadikan peluang bagi para *developer* untuk melakukan eksplorasi, pengembangan dan penyempurnaan lebih lanjut. Masih sangat banyak potensi sistem berbasis *smart contract* yang bisa digali, dengan kata lain tidak menutup kemungkinan bahwa *smart contract* dapat diimplementasikan terhadap sistem-sistem lainnya, mengingat integritas data yang dihasilkan dari penelitian ini sangat baik.

DAFTAR PUSTAKA

- Adhi, R. A. & Harjono, 2014. Rancang Bangun Sistem Informasi E-Voting Berbasis SMS. *JUITA*, III(2), pp. 85-93.
- Angular, 2010. *Angular*. [Online] Available at: <https://angular.io/> [Accessed 31 March 2021].
- Ardilla, R., 2018. Rancang Bangun Sistem E – Voting Dengan Metode Enkripsi Blockchain Di Kota Mojokerto. *Repository Universitas Islam Majapahit*.
- Code, V. S., 2021. *Visual Studio Code*. [Online] Available at: <https://code.visualstudio.com> [Accessed 31 March 2021].
- Copes, F., 2019. *The Next.js Handbook*. s.l.:s.n.
- Derizal, 2011. Panduan Lengkap PHP, Ajax, jQuery. *PHP Ajax Javascript jQuery Tutorial Indonesia*, pp. 27-39.
- Google, C., 2021. *Google Cloud*. [Online] Available at: <https://cloud.google.com/> [Accessed 31 March 2021].
- Hassan, C. A. u. et al., 2022. A Liquid Democracy Enabled Blockchain-Based Electronic Voting System. *Hindawi*, Volume 2022, pp. 1-10.
- Johari, R. et al., 2020. SEVA: Secure E-Voting Application in Cyber Physical System. *Cyber-Physical Systems*, pp. 1-31.
- Kadir, A., 2008. *Tuntunan Praktis Belajar Database Menggunakan MySQL*. Yogyakarta: Andi Offset.
- Khan, K. M., Arshad, J. & Khan, M. M., 2018. Secure Digital Voting System based on Blockchain Technology. *CORE*.
- Khoury, D., Kfouri, E. F., Kassem, A. & Harb, H., 2018. Decentralized Voting Platform Based on Ethereum Blockchain. *IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*.
- Kurnia Hu, S. D., Palit, H. N. & Handojo, A., 2019. Implementasi Blockchain: Studi Kasus e-Voting. *Jurnal Infra*, VII(1).
- Mysliwiec, K., 2017. *NestJS*. [Online] Available at: <https://nestjs.com/> [Accessed 31 March 2021].

- Petersen, K., Wohlin, C. & Baca, D., 2009. *The Waterfall Model in Large-Scale Development*. Findland, 10th International Conference.
- Pierro, M. D., 2017. What Is the Blockchain?. *Computing in Science & Engineering*, Volume 19, pp. 92-95.
- Pressman, R., 2010. *Software Engineering : a practitioner's approach*. New York: McGraw - Hill.
- Rahardja, U., Aini, Q., Yusup, M. & Edliyanti, A., 2020. Penerapan Teknologi Blockchain Sebagai Media Pengamanan Proses Transaksi E-Commerce. *Journal of Computer Engineering System and Science*, V(1), pp. 28--32.
- Ridwan, M., Arifin, Z. & Yulianto, 2016. Rancang Bangun E-Voting Menggunakan Keamanan RSA. *Jurnal Informatika Mulawarman*.
- Setia, T. E. H. & Susanto, A., 2019. Smart Contract Blockchain pada E-Voting. *Jurnal Informatika Upgris* , V(2), pp. 188-191.
- Shukla, S., D O, S., A N, T. & H R, D. M., 2020. *ONLINE VOTING APPLICATION USING ETHEREUM BLOCKCHAIN*. s.l., UTC from IEEE Xplore.
- Sommerville, I., 2011. *Software Engineering*. 9 ed. United States of America: Pearson Education Inc., publishing as Addison-Wesley.
- Szabo, N., 2018. Smart Contracts : Building Blocks for Digital Markets.
- Web3JS, 2016. *web3.js*. [Online] Available at: <https://web3js.readthedocs.io/> [Accessed 31 March 2021].
- Wilson, J. R., 2013. *Node.js the Right Way*. 1st ed. United States of America: The Pragmatic Programmers, LLC..