BAB III
METODOLOGI PENELITIAN

3.1 OBJEK PENELITIAN

Objek pada penelitian ini adalah melakukan implementasi teknologi *Blockchain* dengan menggunakan platform *Hyperledger Fabric* pada sistem pemungutan suara elektronik berbasis *Android*.

3.2 ALAT DAN BAHAN PENELITIAN

Alat dan bahan yang digunakan peneliti untuk melakukan penelitian antara lain :

3.2.1 Perangkat Keras

Tabel 3.1 Spesifikasi alat perangkat keras

Komponen	Spesifikasi
Processor	Intel Core i7
RAM	16 GB
Storage	1TB

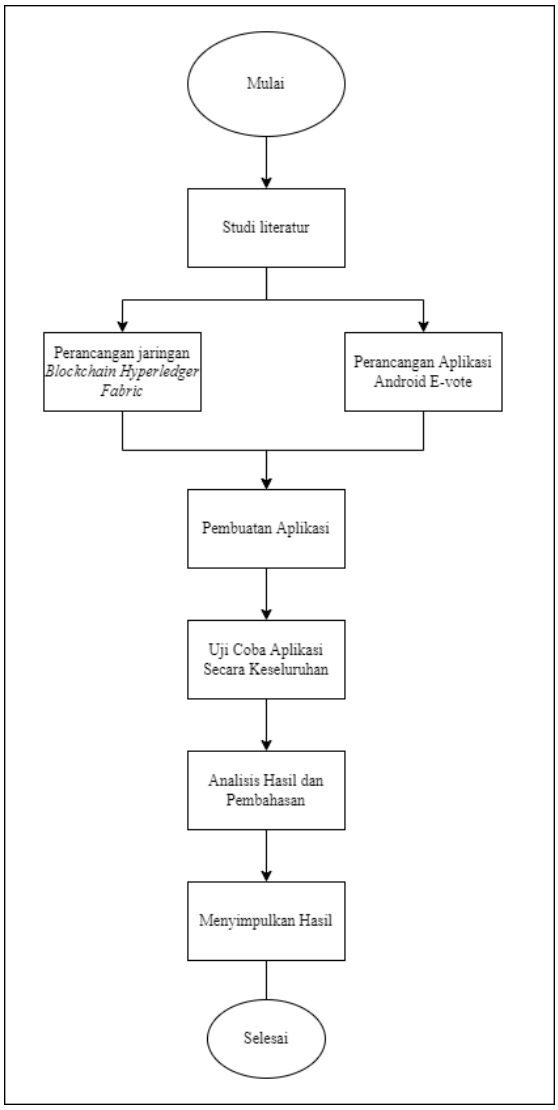
3.2.2 Perangkat Lunak

Tabel 3.2 Spesifikasi perangkat lunak

Kebutuhan	Keterangan	Fungsi
Sistem Operasi	Windows 11	Sistem operasi untuk menjalankan perangkat lunak.

Kebutuhan	Keterangan	Fungsi
Aplikasi	Docker	Membangun dan menjalankan aplikasi secara terisolasi.
	Figma	Membuat desain aplikasi
	Visual Studio Code	Alat untuk melakukan pengembangan aplikasi.
	Github	Menyimpan dan mengelola *source code* dari aplikasi

3.3 DIAGRAM ALIR PENELITIAN



Gambar 3.1 Flowchart alur penelitian

Untuk menjelaskan lebih detail terkait diagram alur penelitian pada Gambar 3.1, maka peneliti membuat sub bab yang berisikan penjelasan mengenai tiap tahap yang berada pada Gambar 3.1. Berikut sub bab yang telah dibuat:

3.3.1 Studi Literatur

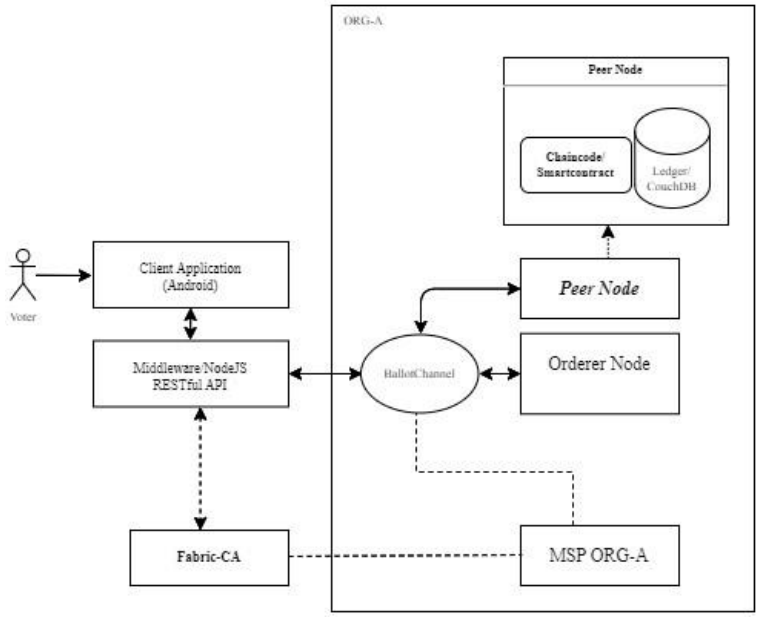
Pada tahap ini peneliti melakukan studi literatur dengan mengumpulkan data dari referensi serta teori-teori yang berkaitan dengan penggunaan teknologi *blockchain* pada sistem *e-vote*. Studi literatur dilakukan dengan membaca jurnal

ilmiah, buku, dan artikel pada internet yang berkaitan dengan topik yang diangkat yaitu implementasi teknologi *blockchain hyperledger fabric* pada aplikasi *e-vote*.

3.3.2 Perancangan jaringan *Blockchain Hyperledger Fabric*

Perancangan jaringan *Blockchain* pada *Hyperledger Fabric* bertujuan untuk memberikan gambaran bagaimana nantinya jaringan *Blockchain* bekerja pada sistem *e-vote*.

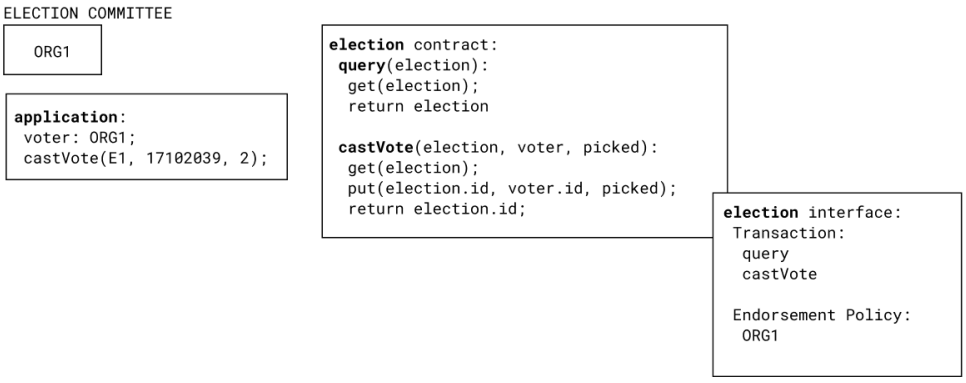
3.3.2.1 Arsitektur jaringan *blockchain* pada *Hyperledger Fabric*



Gambar 3.2 Rancangan Arsitektur Aplikasi E-Vote

Tahap ini peneliti membuat rancangan bagaimana nantinya jaringan *blockchain* menggunakan *hyperledger fabric* bekerja. Rancangan arsitektur Sistem *E-vote* menggunakan *Hyperledger Fabric* dapat dilihat pada Gambar 3.2. Sistem *E-vote* yang berjalan pada jaringan *Blockchain* menggunakan platform *Hyperledger Fabric* yang berada pada *container*. Pada saat pengguna atau *voter* melakukan aksi pada aplikasi maka pengguna atau *voter* akan memanggil *RESTful API* dan kemudian *RESTful API* akan mengeksekusi sebuah instruksi atau fungsi pada *chaincode* yang berada pada *Peer Node* dalam jaringan *Hyperledger Fabric*. Secara aktual seluruh data yang ada pada jaringan akan disimpan pada *CouchDB* sebagai *state-database* atau biasa disebut *world state* pada jaringan *blockchain*. *Block* yang

ada pada *blockchain* hanya menyimpan seluruh catatan transaksi yang telah dilakukan. Untuk melakukan sebuah transaksi pada jaringan *Hyperledger Fabric* dibutuhkan sebuah *smartcontract* atau disebut *chaincode*. *Chaincode* pada *Hyperledger Fabric* dapat dibuat dengan basis bahasa pemrograman *Javascript*. *Chaincode* berguna untuk mendefinisikan logika yang dapat dieksekusi oleh klien dan membuat transaksi baru yang nantinya akan ditambahkan ke *ledger* dan *world state*.



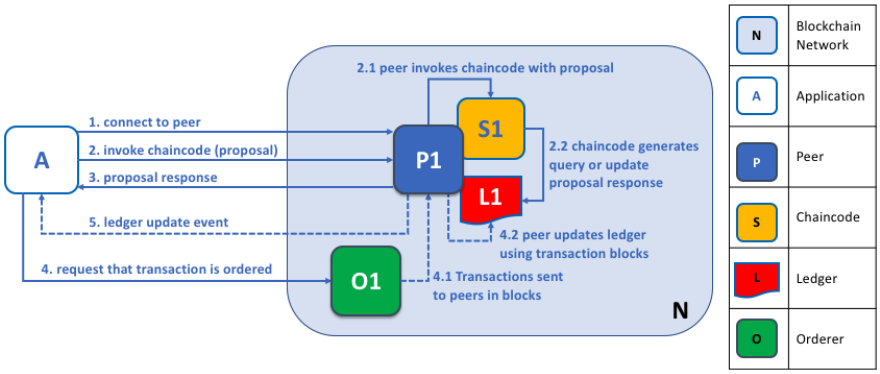
Gambar 3.3 *Chaincode* pada *E-Vote*

Pada Gambar 3.3 merupakan diagram desain *Chaincode* pada sistem *E-vote*, yang dimana terdapat satu organisasi yaitu ORG1 yang menetapkan sebuah *Chaincode* untuk menampilkan, dan menambahkan suara. Pada diagram *election contract* berisi semua program yang dapat dilakukan oleh sebuah klien seperti *query(election)* yang berarti menampilkan seluruh hasil voting dan terdapat fungsi *castVote* yang berguna untuk menambahkan suara pada saat pemilihan suara. Untuk dapat mengeksekusi kontrak tersebut, dibutuhkan sebuah aplikasi, seperti pada Gambar 3.3 terdapat diagram *application* yang berisi perintah pada saat *chaincode* tersebut telah dimasukkan kedalam jaringan *Hyperledger Fabric*, diagram tersebut berisi identitas *voter* yang berasal dari ORG1 dan mengeksekusi fungsi *castVote* pada kontrak yang dimana parameter yang dimasukkkan adalah E1 yaitu ID dari *election*, *voter.id* yang memiliki nilai 17102039 yang merupakan ID dari *voter*, dan *picked* yang memiliki nilai dua yang berarti nomor kandidat yang dipilih. Diagram *election interface* mendefinisikan transaksi-transaksi yang dapat dipanggil atau

dieksekusi oleh klien dan *Endrosement policy* merupakan variabel yang mendefinisikan organisasi atau *peer* yang mana yang harus melakukan *endorse* terhadap eksekusi proposal.

3.3.2.2 Alir transaksi pada jaringan *Hyperledger Fabric*.

Pada *Hyperledger Fabric* tidak terdapat *miner* seperti pada kebanyakan jaringan *blockchain* lainnya, *Hyperledger Fabric* bekerja secara berbeda, jika pada jaringan seperti *Ethereum* dan *Bitcoin* menggunakan konsensus probabilistik, maka *Hyperledger* menggunakan konsensus deterministik. *Hyperledger Fabric* memiliki suatu fitur *node* yang disebut *orderer*, yang dimana *node orderer* yang akan membentuk layanan pemesanan bersama dengan *node orderer* lainnya. Selain melakukan layanan pemesanan, *node orderer* juga akan memelihara daftar organisasi yang di izinkan untuk membuat saluran, daftar organisasi tersebutlah yang disebut sebagai *consortium*. *Node orderer* hanya dapat diatur oleh *orderer admin*. *Orderer* juga berperan untuk menerapkan kontrol akses dasar untuk *channel* seperti membatasi yang dapat melihat dan melakukan aksi ke *channel* tertentu, dan siapa yang dapat mengkonfigurasi *channel* tersebut.

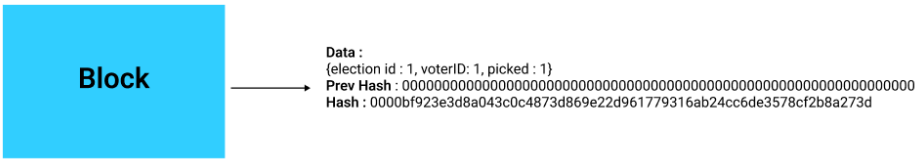


Gambar 3.4 Alir transaksi pada *Hyperledger Fabric* [19]

Pada dasarnya pada saat pertama kali pengguna melakukan transaksi maka klien mengirimkan proposal transaksi ke subset *peer* yang akan meminta *smart contract* untuk melakukan pembaruan data pada *ledger* dan melakukan *endorse* terhadap hasilnya. Apabila mayoritas *peer* menyetujui proposal, maka klien aplikasi mengirimkan tanggapan proposal kepada *node orderer*, yang kemdian *node orderer*

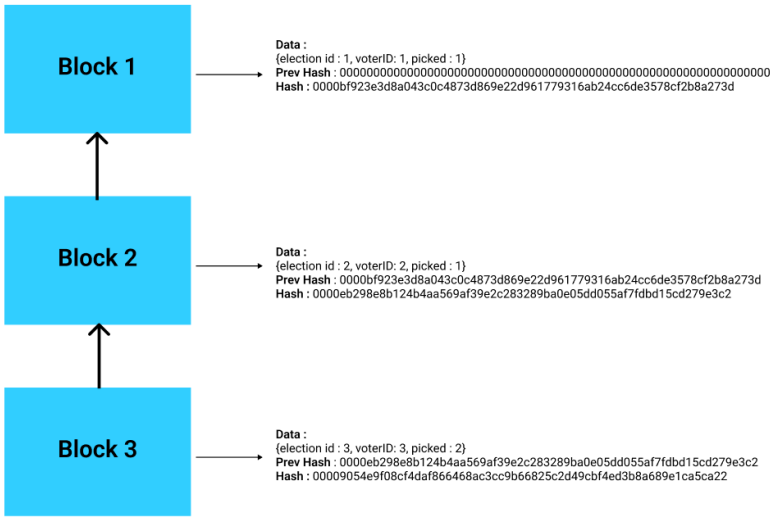
akan membuat *block* baru dan didistribusikan ke semua *peers* pada *channel* untuk validasi. Pada fase validasi, setiap *peer* akan memvalidasi *block* terdistribusi secara independen, tetapi dengan cara deterministik. Setelah semua *peer* melakukan validasi maka *ordering service* membuat blok baru yang berisi catatan transaksi yang telah dilakukan dan juga mencatat ke dalam *ledger*, alir transaksi dapat dilihat pada Gambar 3.4.

3.3.2.3 Keamanan dan performa jaringan *Hyperledger Fabric*



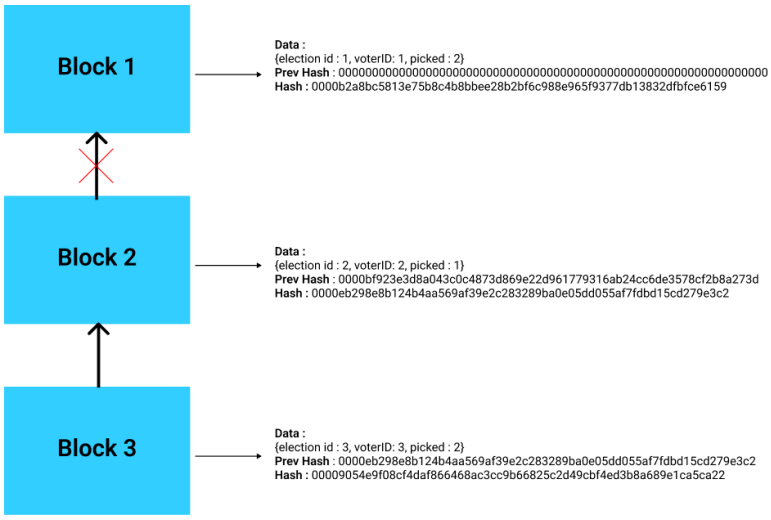
Gambar 3.5 Kalkulasi *Hash* dari sebuah blok

Blockheader pada *Hyperledger* merupakan hasil hash dari algoritma *SHA256* atau *SHA3*-256 yang dimana hash tersebut dihasilkan dari input *previousHash*, *Number*, dan hasil dari *dataHash* seperti pada Gambar 3.5. Pada saat data dalam *block* tersebut diganti maka otomatis *blockheader hashnya* akan berbeda dikarenakan *datahash* merupakan termasuk dalam *input hash* pada *blockheader*. Hal tersebutlah yang menyebabkan teknologi *blockchain tamper proof* atau jika disalah satu data pada *block* diganti maka *chain* atau rantai akan terputus dan *block* tersebut dianggap tidak valid. Untuk contohnya peneliti, mensimulasikan *insider-attack* dengan cara melakukan *tamper* disalah satu blok.



Gambar 3.6 *Block* awal sebelum dilakukan penggantian data

Pada Gambar 3.6 merupakan data awal pada *blockchain* yang dimana ingin dilakukan percobaan *tamper data*, pada simulasi ini, dilakukan *tamper* pada *block 1* dimana mengubah data *picked* menjadi nilai dua.



Gambar 3.7 *Block* setelah dilakukan penggantian data

Pada saat melakukan modifikasi data pada Block 1, maka tidak terjadi kecocokan nilai antara *prevHash* pada Block 2 dan nilai *hash Blockheader* pada Block 1 seperti pada Gambar 3.7. *Hash Blockheader* pada Block 1 bernilai ”0000b2a8b…” sedangkan pada Block 2 tercatat *prevHash* bernilai “0000bf923…”

dan pada saat jaringan *blockchain* mendeteksi hal tersebut maka *Block 2* dan seterusnya akan terputus sehingga data lainnya akan dianggap tidak valid oleh *Ledger* pada *Hyperledger Fabric*. Untuk melakukan *tamper data* perlu dilakukan *tampering* pada seluruh *block* yang saling terhubung pada jaringan *blockchain*.

Caliper report

Summary of performance metrics

Name	Succ	Fail	Send Rate (TPS)	Max Latency (s)	Min Latency (s)	Avg Latency (s)	Throughput (TPS)
createAsset with 1000 TX.	998	2	500.8	2.10	0.13	1.03	245.6
createAsset with 5000 TX.	4996	4	500.5	8.04	0.09	4.32	277.6
createAsset with 10000 TX.	10000	0	500.2	23.25	0.07	9.43	231.4
createAsset with 30000 TX.	29999	1	498.4	61.54	0.09	34.60	253.9
createAsset with 50000 TX.	49999	1	498.6	225.58	0.07	105.00	157.6
readAsset	85169	0	1443.5	0.06	0.00	0.01	1443.4

Gambar 3.8 Hasil pengujian jaringan *Hyperledger Fabric*

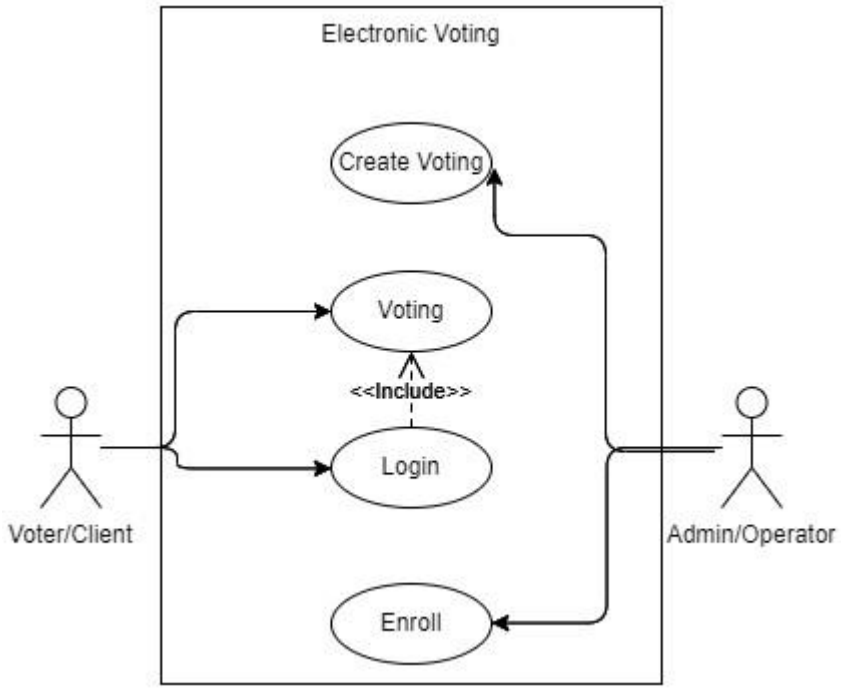
Pada Gambar 3.8 terdapat hasil dari pengujian *Hyperledger Fabric* menggunakan *Hyperledger Caliper*. Pada pengujian tersebut, masing-masing *Round* pengujian menggunakan *Rate Control* berjenis *fixed-rate* dan menggunakan *rate TPS (Transaction per second*) sebanyak 500. Pada saat pengujian melakukan *insert* atau pembuatan *block* yang berisi data berupa *string* yang berisi nilai antara 10 sampai dengan 100. Pengujian tersebut menghasilkan bahwa *Throughput* yang dihasilkan tidak bergantung pada jumlah transaksi yang dilakukan. Akan tetapi, jumlah transaksi akan memengaruhi *Latency* atau waktu yang diperlukan paket dari saat dikirim sampai diterima tujuan.

3.3.3 Perancangan Aplikasi Android *E-vote*

Setelah mendeskripsikan aliran transaksi pada jaringan *Hyperledger Fabric*, langkah selanjutnya ialah merancang Aplikasi pada Android, agar mempermudah interaksi pengguna untuk masuk dan melakukan transaksi pada jaringan *Hyperledger Fabric*. Untuk dapat mengakses jaringan tersebut, maka pengguna

perlu melakukan registrasi.Setelah pengguna melakukan pendaftaran atau masuk kedalam akun maka pengguna perlu terdaftar sebagai partisipan disuatu pemungutan suara yang telah ada.

Pengguna perlu melakukan verifikasi untuk memastikan keaslian akun. Pengguna dapat meminta izin kepada pemilik pemungutan suara atau pengguna juga dapat diberi izin terlebih dahulu oleh pemilik pemungutan suara. Jika pengguna sudah terdaftar sebagai partisipan, maka dapat langsung melakukan pemilihan.



Gambar 3.9 *Use Case Diagram* Akses user terhadap aplikasi

Gambar 3.9 merupakan *use case diagram* dari Aplikasi *E-vote*. Pada *use case diagram* terdapat satu aktor yaitu *voter* yang akan menggunakan aplikasi tersebut. Pengguna atau *voter* dapat melakukan registrasi atau masuk kedalam akun, tetapi jika ingin melakukan *voting* maka perlu masuk kedalam akun. Untuk memberikan penjelasan lebih detail terkait empat fitur yang ada pada Gambar 3.4, maka peneliti menggunakan Skenario dan *Activity Diagram*, sebagai berikut :

3.3.3.1 Skenario

Skenario adalah penjelasan interaksi antara pengguna dengan sebuah sistem yang akan dirancang melalui alur cerita, dengan tujuan pengguna dapat mengetahui bagaimana sistem tersebut digunakan. Berikut tabel dari masing-masing skenario :

Tabel 3.3 Skenario *Use Case Login*

No. *Use Case* : 1
Nama *Use Case* : Login/Register
Aktor : *Voter*/Pengguna
Include Use Case : -
Tujuan : Melakukan *login* ke dalam aplikasi
Kondisi Awal : Pengguna membuka aplikasi kemudian tampil halaman login.
Skenario Utama : 1. Pengguna masuk kehalaman login/register. 2. Aplikasi akan mendeteksi apakah pengguna telah terdaftar.
Kondisi Akhir : Aplikasi akan menampilkan halaman utama

Tabel 3.4 Skenario *Use Case Verify*

No. *Use Case* : 2
Nama *Use Case* : *Enroll*
Aktor : *Voter*/Pengguna
Include Use Case : -

Tujuan : Melakukan enroll untuk mendapatkan sertifikat autentikasi kedalam jaringan *Hyperledger Fabric*
Kondisi Awal : Pengguna membuka Halaman *enroll*
Skenario Utama : 1. Pengguna masuk kehalaman *enroll* 2. Pengguna akan mengisi form *enroll*. 3. Pengguna akan melakukan *enroll* atau masuk kedalam salah satu jaringan *Hyperledger Fabric*
Kondisi Akhir : Aplikasi akan menampilkan halaman utama jika *enroll* berhasil

Tabel 3.5 Skenario *Use Case Voting*

No. *Use Case* : 3
Nama *Use Case* : *Voting*
Aktor : *Voter*/Pengguna
Include Use Case : *Login, Enroll*
Tujuan : Pengguna melakukan *voting*
Kondisi Awal : Pada halaman utama pengguna memilih menu voting kemudian tampil halaman voting
Skenario Utama : 1. Pengguna masuk kehalaman *Voting* 2. Jika pengguna belum melakukan *login/enroll* maka diarahkan kehalaman *login/enroll*. 3. Pengguna memilih *voting* yang tersedia 4. Pengguna melakukan *voting*

5. Pengguna dapat melihat hasil voting (jika diizinkan)
Kondisi Akhir : Aplikasi akan menampilkan pesan *voting* berhasil dan dapat melihat keseluruhan hasil voting (jika diizinka)

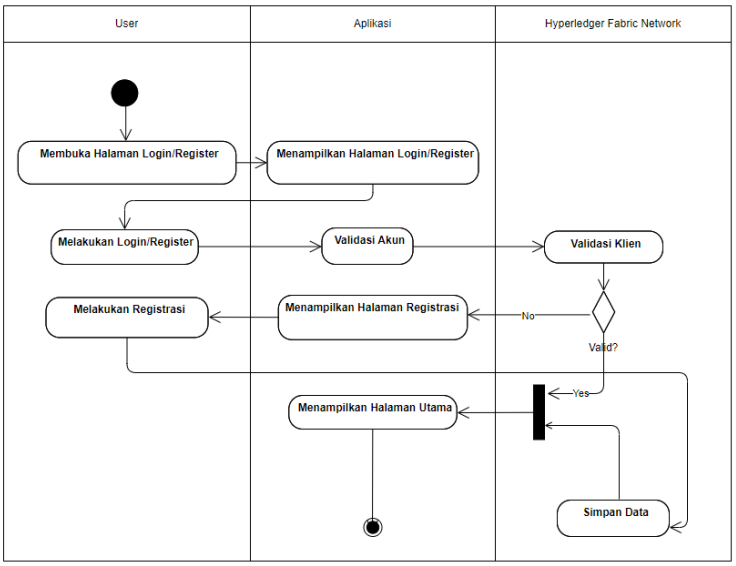
Tabel 3.6 Skenario *Use Case Create Voting*

No. *Use Case* : 4
Nama *Use Case* : *Create Voting*
Aktor : Admin/Operator
Include Use Case : -
Tujuan : Admin/Operator membuat *voting*
Kondisi Awal : -
Skenario Utama : 1. Admin/Operator melakukan invoke melalui *Command Line Interface* pada salah satu *peer*.
Kondisi Akhir : -

3.3.3.2 *Activity Diagram*

Activity diagram digunakan untuk menggambarkan bagaimana alur aktivias dari sistem yang dirancang. Berikut adalah *activity diagram* pada aplikasi Android *E-vote* :

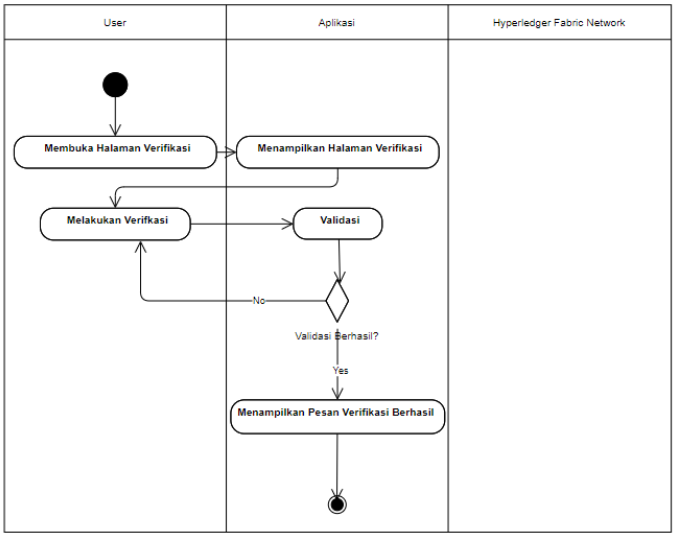
1. *Activity diagram Login*



Gambar 3.10 Activity diagram Login/register

Pada Gambar 3.10 merupakan gambar *activity diagram* dengan aktor *voter*. Fungsi *Login/Register* tersebut sebagai langkah awal pada saat pengguna membuka aplikasi. Maka aplikasi akan mencocokkan apakah pengguna telah terdaftar ke dalam sebuah organisasi pada *Hyperledger Fabric*. Jika sudah terdaftar maka diarahkan kehalaman utama, jika sistem menduga otorisasi tidak ada, maka pengguna dapat mengembalikan akun dengan memasukkan kode yang diberikan pada saat awal pendaftaran.

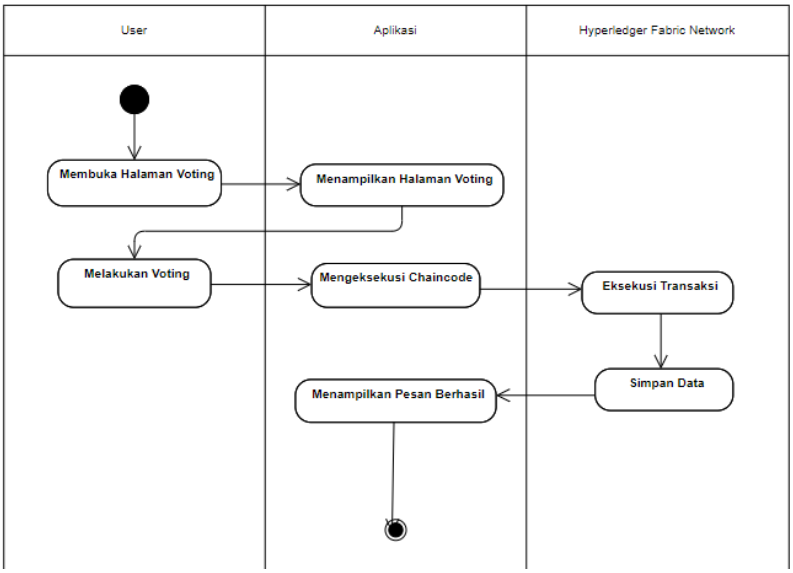
2. *Activity diagram Verify/Enroll*



Gambar 3.11 ***Activity diagram verify***

Pada Gambar 3.11 terdapat *activity diagram* pada saat pengguna melakukan verifikasi, verifikasi dilakukan untuk memastikan keaslian akun pengguna. Pengguna dapat masuk ke dalam jaringan *blockchain* jika telah berhasil melakukan registrasi.

3. *Activity diagram Voting*

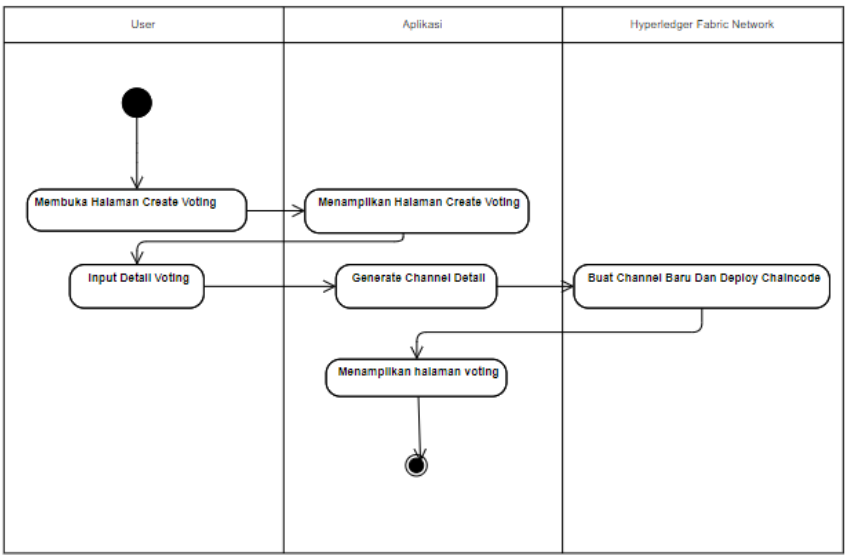


Gambar 3.12 ***Activity diagram voting***

Gambar 3.12 merupakan *activity diagram* pada saat pengguna melakukan pemungutan suara atau *voting*. Pada saat pengguna melakukan voting maka sistem

akan mengirim detail transaksi ke jaringan *blockchain* dan menyimpan data pada *ledger*.

4. *Activity diagram Create Voting*



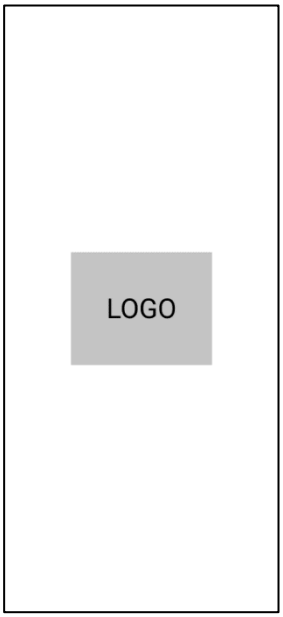
Gambar 3.13 *Activity diagram create voting*

Gambar 3.13 merupakan *activity diagram* pada saat admin melakukan pembuatan *voting*.

3.3.3.3 Desain *User Interface*

1. *Splash Screen*

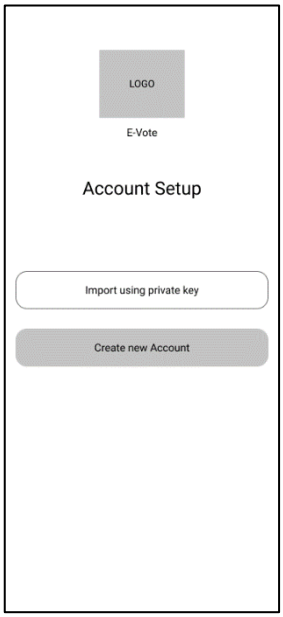
Pada Gambar 3.14 merupakan tampilan *Splash Screen* yaitu tampilan pada saat awal aplikasi dibuka.



Gambar 3.14 Halaman *SplashScreen*

2. Halaman *Login/Register*

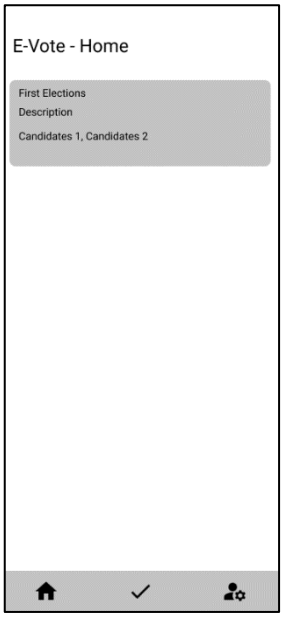
Gambar 3.15 merupakan tampilan halaman setelah *Splash Screen*, pada halaman ini jika user telah melakukan *login* maka otomatis akan diarahkan ke halaman utama, jika belum melakukan *login* maka halaman ini yang akan tampil, pada halaman ini terdapat dua pilihan yaitu membuat akun baru atau masuk dengan menggunakan private key yang telah diberikan sebelumnya jika pengguna telah melakukan registrasi.



Gambar 3.15 Halaman *Login/Register*

3. Halaman *Dashboard*

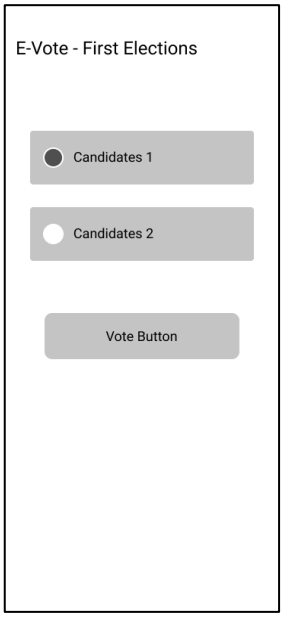
Gambar 3.16 merupakan halaman *Dashboard* merupakan halaman utama dari aplikasi, halaman ini berisi tentang *voting* yang tersedia jika pengguna telah melakukan verifikasi dan *enroll* kedalam jaringan *Hyperledger Fabric*.



Gambar 3.16 Halaman *Dashboard*

4. Halaman *Voting*

Gambar 3.17 merupakan tampilan halaman pada saat pengguna melakukan pemungutan suara



Gambar 3.17 Halaman *Voting*

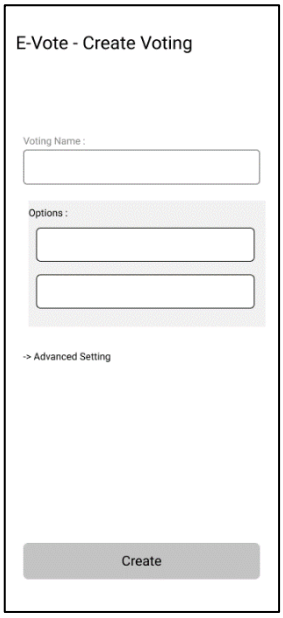
5. Halaman *Verify/Enroll*

Gambar 3.18 merupakan tampilan halaman pada saat pengguna melakukan verifikasi data untuk validasi keabsahan akun.

Gambar 3.18 Halaman *Verify*

6. Halaman *Create Voting*

Gambar 3.19 merupakan tampilan halaman jika pengguna ingin membuat *voting*. Pengguna akan diminta untuk mengisi data seperti nama *voting*, Opsi atau pilihan kandidat, dan dapat melakukan pengaturan lebih lanjut.



Gambar 3.19 Halaman *Create Voting*

3.3.4 Pembuatan Aplikasi

Setelah melakukan perancangan terhadap jaringan *Hyperledger Fabric* dan perancangan sistem aplikasi pada Android, maka dilanjutkan dengan tahap pembuatan aplikasi. Pada tahap ini peneliti akan membangun sebuah jaringan *blockchain* menggunakan *Hyperledger Fabric* dan membuat sebuah aplikasi *e-vote* berbasis Android sebagai antarmuka pengguna dengan jaringan tersebut.

3.3.5 Uji Coba Aplikasi Secara Keseluruhan

Setelah aplikasi *Android*, jaringan *Blockchain Hyperledger Fabric*, dan *RESTful API* telah dibuat maka akan dilakukan pengujian satu persatu. Pengujian sangat penting dilakukan untuk memastikan bahwa aplikasi tersebut sudah berjalan sesuai yang diharapkan. Untuk pengujian *chaincode* atau *smartcontract* pada jaringan *blockchain* dapat menggunakan *Fabric-mock-stub*. Untuk pengujian *RESTful API* dapat menggunakan *SOAPUI*. Dalam pengujian aplikasi android, pengujian dilakukan dengan menggunakan kerangka kerja *JUnit*.