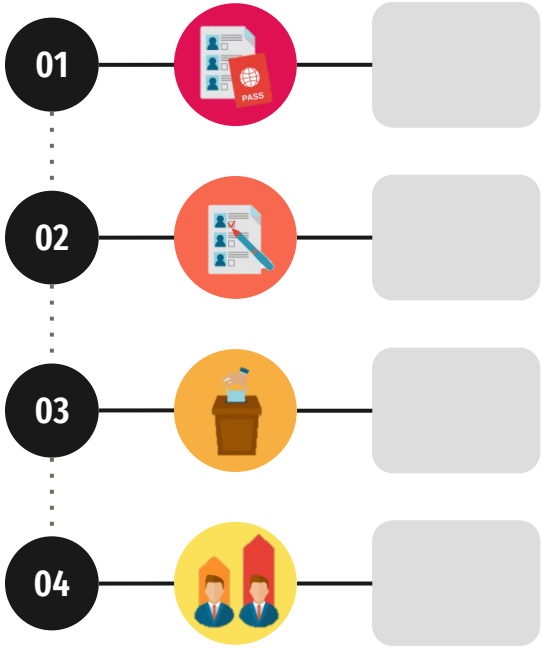
ELECTION

E-voting blockchain solution



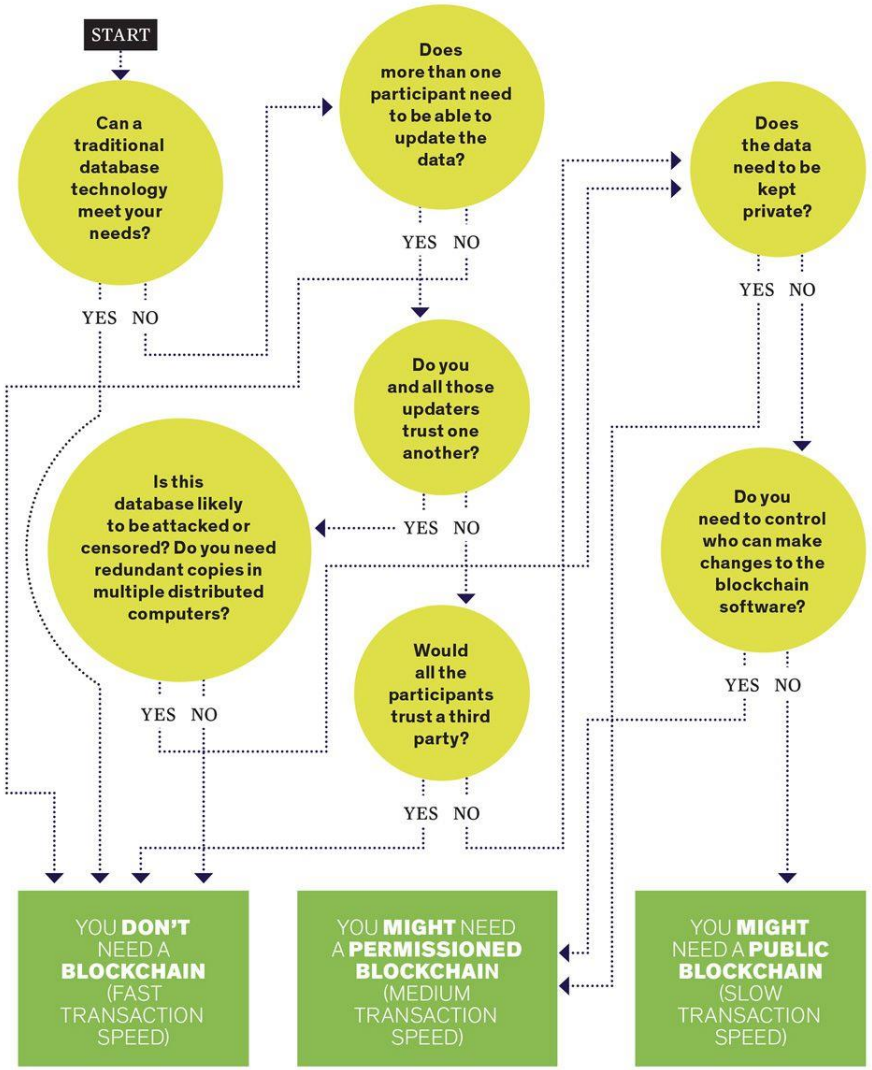
Background

Traditional election → Blockchain solution

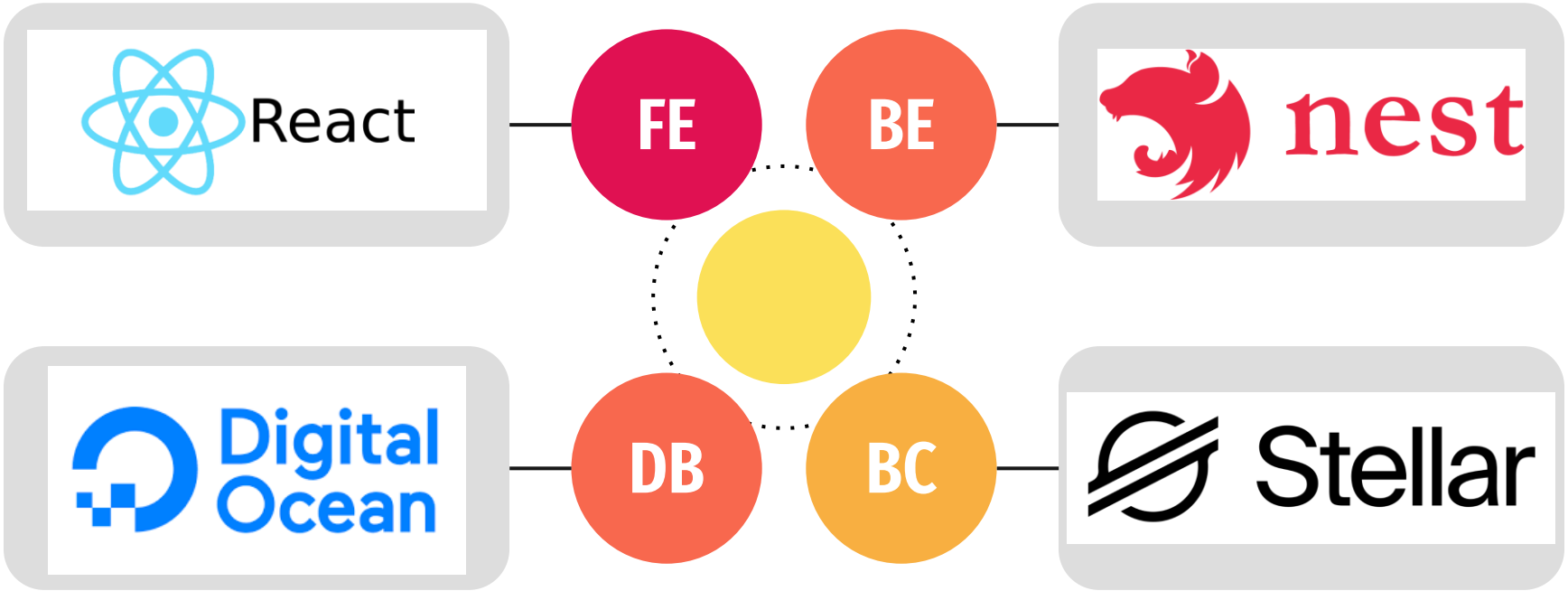
- Human error
- Corruption

- Transparent
- Verifiable

Why do we need blockchain



Implementation



Database

Tables

Gorv_data

- Citizen ID and area that citizen can vote

Candidate

- Candidate Info
- Candidate Account

Issuer

- Issuer Account

Roles

Issuer

- Create election
- Create Coin and give to people
- Create candidate account

People

- Verify Citizen ID
- Create account
- Vote candidate

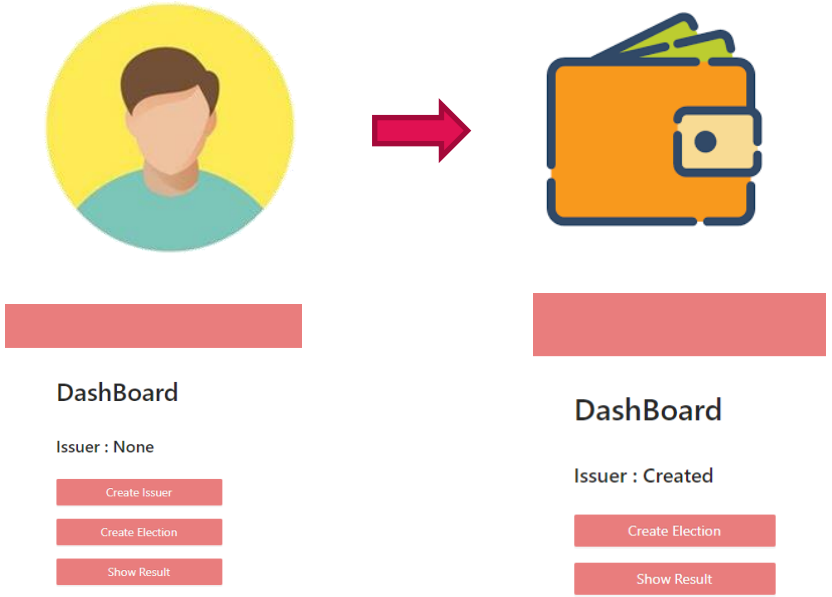
Candidates

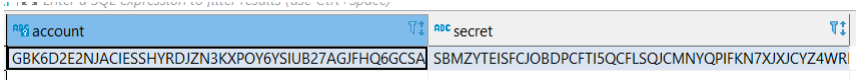
- Register with issuer
- Get account from Issuer
- Wait for voting result

Activity Flow

Create Election

1. Create Issuer Account
2. Create Election
3. Add candidates
4. Recheck candidates information
5. Generate candidate account
6. Add information and account to database
7. Show information and account





Activity Flow

Create Election

01 Create Issuer Account

02 Create Election

03 Add candidates

04 Recheck candidates information

05 Generate candidate account

06 Add information and account to database

07 Show information and account

Activity Flow

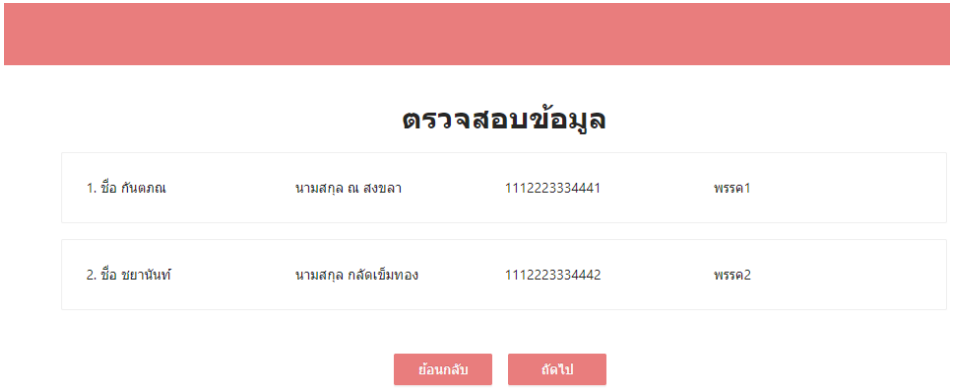
Create Election

- Create Issuer Account 01
- Create Election 02
- Add candidates 03
- Recheck candidates information 04
- Generate candidate account 05
- Add information and account to database 06
- Show information and account 07

Activity Flow

Create Election

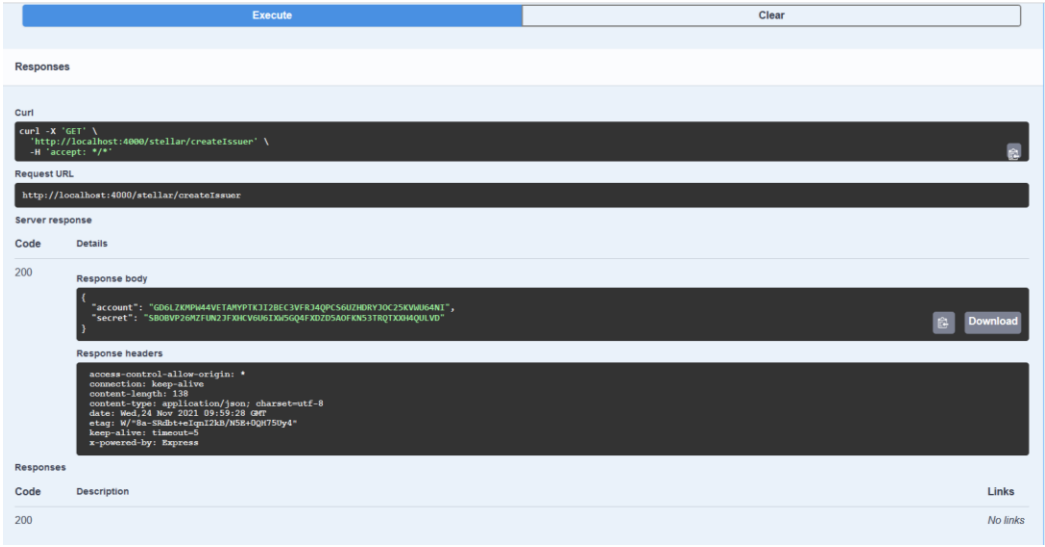
- Create Issuer Account 01
- Create Election 02
- Add candidates 03
- Recheck candidates information 04
- Generate candidate account 05
- Add information and account to database 06
- Show information and account 07



Activity Flow

Create Election

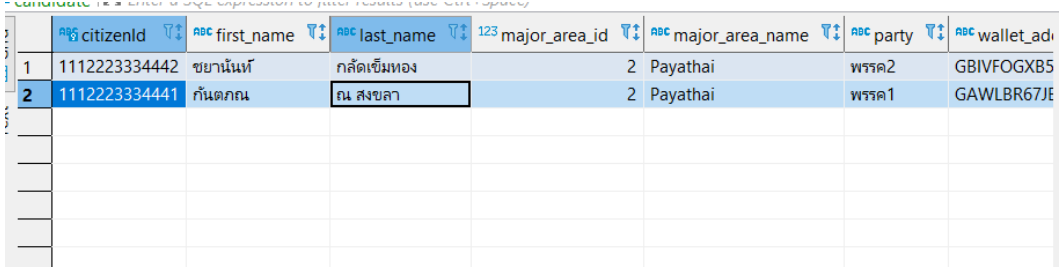
- Create Issuer Account — 01
- Create Election — 02
- Add candidates — 03
- Recheck candidates information — 04
- Generate candidate account — 05
- Add information and account to database — 06
- Show information and account — 07



Activity Flow

Create Election

- Create Issuer Account 01
- Create Election 02
- Add candidates 03
- Recheck candidates information 04
- Generate candidate account 05
- Add information and account to database 06
- Show information and account 07

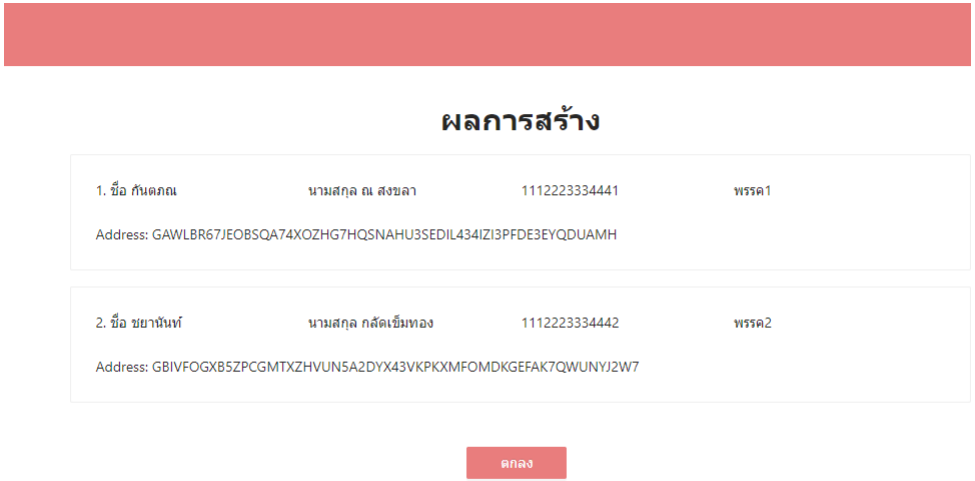


	citizenId	first_name	last_name	major_area_id	major_area_name	party	wallet_ad
1	1112223334442	ชยานันท์	กลัดเข็มทอง	2	Payathai	พรรค2	GBIVFOGXB5
2	1112223334441	กันตภณ	ณ สงขลา	2	Payathai	พรรค1	GAWLBR67JE

Activity Flow

Create Election

- Create Issuer Account 01
- Create Election 02
- Add candidates 03
- Recheck candidates information 04
- Generate candidate account 05
- Add information and account to database 06
- Show information and account 07



Activity Flow

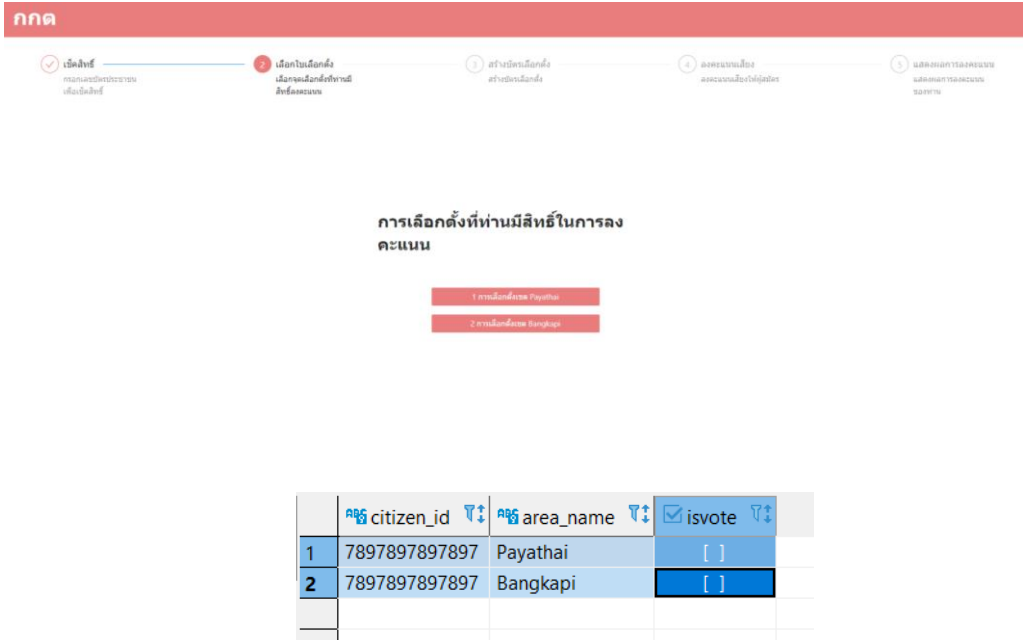
Voting

- Verify Citizen ID 01
- Check availiable voting area 02
- Create wallet and get coin 03
- Choose candidate and vote 04
- Show vote result 05
- Show voting summary 06

Activity Flow

Voting

- Verify Citizen ID — 01
- Check availiable voting area — 02
- Create wallet and get coin — 03
- Choose candidate and vote — 04
- Show vote result — 05
- Show voting summary — 06



	citizen_id	area_name	isvote
1	7897897897897	Payathai	[ ]
2	7897897897897	Bangkapi	[ ]

Activity Flow

Voting

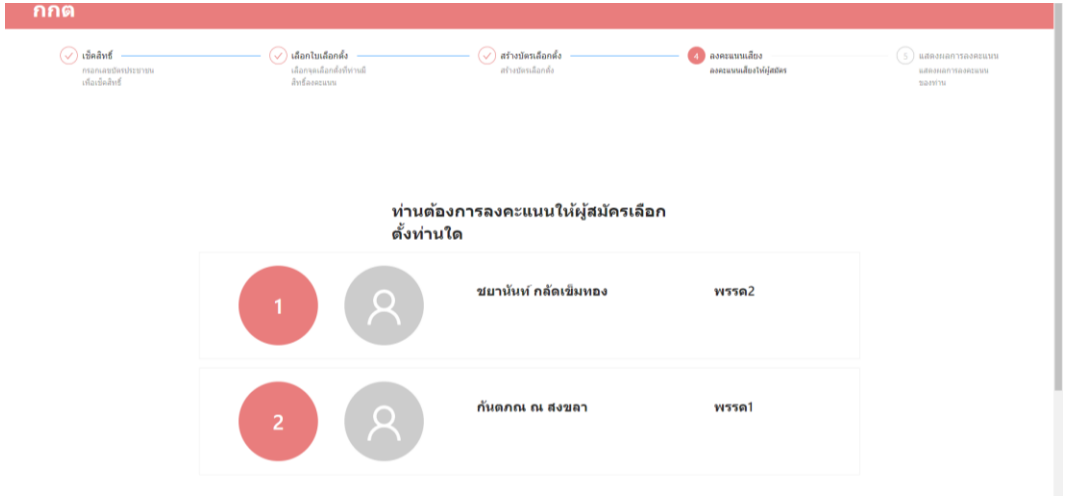
- Verify Citizen ID — 01
- Check availiable voting area — 02
- Create wallet and get coin — 03
- Choose candidate and vote — 04
- Show vote result — 05
- Show voting summary — 06


```
},
"balances": [
  {
    "balance": "1.0000000",
    "limit": "1.0000000",
    "buying_liabilities": "0.0000000",
    "selling_liabilities": "0.0000000",
    "last_modified_ledger": 1151591,
    "is_authorized": true,
    "is_authorized_to_maintain_liabilities": true,
    "asset_type": "credit_alphanum12",
    "asset_code": "Payathai",
    "asset_issuer": "GBK6D2E2NJACIESSHYRDJZN3KXPOY6YSIUB27AGJFHQ6GCSAHGUSIKRC"
  },
```

Activity Flow

Voting

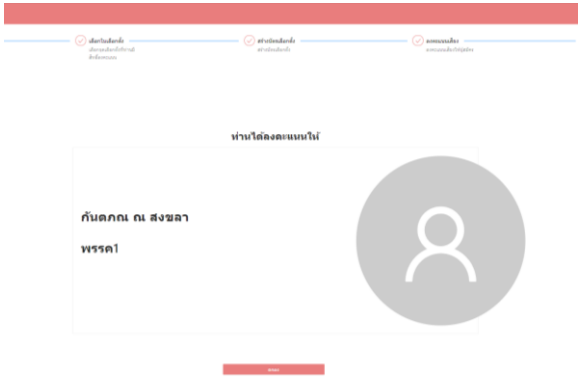
Verify Citizen ID	01
Check availiable voting area	02
Create wallet and get coin	03
Choose candidate and vote	04
Show vote result	05
Show voting summary	06



Activity Flow

Voting

1. Verify Citizen ID
2. Check availiable voting area
3. Create wallet and get coin
4. Choose candidate and vote
5. Show vote result
6. Show voting summary




```
"balances": [
  {
    "balance": "1.0000000",
    "limit": "922337203685.4775807",
    "buying_liabilities": "0.0000000",
    "selling_liabilities": "0.0000000",
    "last_modified_ledger": 1151687,
    "is_authorized": true,
    "is_authorized_to_maintain_liabilities": true,
    "asset_type": "credit_alphanum12",
    "asset_code": "Payathai",
    "asset_issuer": "GBK6D2E2NJACIESSHYRDJZN3KXPOY6YSIUB27AGJFHQ6GCSAHGUSIKRC"
  },
  {
```

Activity Flow

Voting

1. Verify Citizen ID
2. Check availiable voting area
3. Create wallet and get coin
4. Choose candidate and vote
5. Show vote result
6. Show voting summary


```
"balances": [
  {
    "balance": "2.0000000",
    "limit": "922337203685.4775807",
    "buying_liabilities": "0.0000000",
    "selling_liabilities": "0.0000000",
    "last_modified_ledger": 1151724,
    "is_authorized": true,
    "is_authorized_to_maintain_liabilities": true,
    "asset_type": "credit_alphanum12",
    "asset_code": "Payathai",
    "asset_issuer": "GBK6D2E2NJACIESSHYRDJZN3KXPOY6YSIUB27AGJFHQ6GCSAHGUSIKRC"
  },
```

Stellar Implementation

Create Wallet

Generate Key Pair with citizen ID, laser ID, password

```
getKeyPair(UserWallet: CreateUserWallet) {
    const citizenId = UserWallet.citizenId
    const backCard = UserWallet.backCard
    const password = UserWallet.password
    const temp = citizenId + backCard + password
    const seed = StellarSdk.StrKey.encodeEd25519SecretSeed(temp).substring(0,32)
    const keyPair = StellarSdk.Keypair.fromRawEd25519Seed(seed);
    return keyPair;
}
```

Add fund to account

```
async addFund(account: string)  {
    return await axios.get("/friendbot", {
        baseURL: 'https://horizon-testnet.stellar.org',
        params: { addr: account }
    })
}
```

Generate Key Pair with random

```
async createIssuer() {
    const keyPair = StellarSdk.Keypair.random();
    const secret = keyPair.secret();
    const account = keyPair.publicKey();
```

Stellar Implementation

Create Coin and trust coin

Issue new coin

```
const coin = new StellarSdk.Asset(coinName, issuer);
```

```
async trust(account: string, coin, secret) {
    server
    .loadAccount(account)
    .then(function (receiver) {
        var transaction = new StellarSdk.TransactionBuilder(receiver, {
        fee: 100,
        networkPassphrase: StellarSdk.Networks.TESTNET,
        })
        .addOperation(
            StellarSdk.Operation.changeTrust({
            asset: coin,
            }),
        )
        .setTimeout(100)
        .build();
        transaction.sign(secret);
        return server.submitTransaction(transaction);
    })
    .then(console.log)
    .catch(function (error) {
        console.error("Error!", error);
    });
}
```

Trust Coin

- Make wallet trust new issued coin. So, the coin can be used by the wallet
- Limit can be added to thrust to limit how much the wallet can keep the coin If the limit is set to 0 then the wallet will untrust the coin
- Limit is max int64 by default

Stellar Implementation

Create Coin and trust coin

coin1

Trust: coin2
Limit: 1

coin1

Trust: coin1
No limit

coin2

Trust: coin2
Limit: 1

coin2

Trust: coin2
No limit

Stellar Implementation

Send coin

```
async sendCoin(account, secret, coin, destination) {
    server
    .loadAccount(account)
    .then(function (voter) {
        var transaction = new StellarSdk.TransactionBuilder(voter, {
        fee: 100,
        networkPassphrase: StellarSdk.Networks.TESTNET,
        })
        .addOperation(
            StellarSdk.Operation.payment({
            destination: destination,
            asset: coin,
            amount: "1",
            }),
        )
        .addOperation(
            StellarSdk.Operation.changeTrust({
            asset: coin,
            limit: "0",

            }),
        )
        .setTimeout(100)
        .build();
        transaction.sign(secret);
        return server.submitTransaction(transaction);
```

Send Coin

- Send coin to destination wallet
- The transaction will fail if the destination wallet doesn't trust the coin

Stellar Implementation

Get Balance

```
async getBalance(account: Account) {
    const sAccount = await server.loadAccount(account.account);
    const balance = parseInt(sAccount.balances[0]["balance"])
    return balance;
}
```


```
    "balance": "2.0000000",
    "limit": "922337203685.4775807",
    "buying_liabilities": "0.0000000",
    "selling_liabilities": "0.0000000",
    "last_modified_ledger": 1151724,
    "is_authorized": true,
    "is_authorized_to_maintain_liabilities": true,
    "asset_type": "credit_alphanum12",
    "asset_code": "Payathai",
    "asset_issuer": "GBK6D2E2NJACIESSHYRDJZN3KXPOY6YSIUB27AGJFHQ6GCSAHGUSIKRC"
  },
  {
    "balance": "9999.9999900",
    "buying_liabilities": "0.0000000",
    "selling_liabilities": "0.0000000",
    "asset_type": "native"
  }
```


Get Balance

- Will return all coins in the wallet

Concern

Real-time score

เนื่องจากเป็นการโหวตโดยใช้ Blockchain ทำให้ข้อมูลการโหวตทั้งหมดเป็น public และสามารถดูแบบ real-time ได้ ซึ่งสามารถนำไปสู่การชักจูงได้

วิธีการแก้ปัญหาคือ ไม่เปิดผลโหวตแบบ real-time ณ จุดลงคะแนนเลือกตั้ง เพื่อไม่ให้เกิดการชักจูง

Identifiable Private Key

เนื่องจากในการสร้าง wallet account จะใช้ ed25519 ในการhash รหัสบัตรประชาชน, เลขหลังบัตร, และ รหัสผ่าน เพื่อใช้เป็น private key ดังนั้นหากรู้ข้อมูลเหล่านี้จะทำให้สามารถระบุได้ว่าเจ้าของ wallet โหวตใครไป ซึ่งอาจนำไปสู่ปัญหาการซื้อเสียงที่สามารถยืนยันได้

Concern

Duplicate coin

เนื่องจาก Issuer หรือคนออกเหรียญสามารถออกเหรียญให้เหรียญโดยที่ไม่จำเป็นต้องมีเหรียญของตนในบัญชีได้ ดังนั้นกรณีที่ถูกรู้ key ของ Issuer จะทำให้เกิดการส่งเหรียญให้กับผู้สมัครได้

วิธีการแก้คือ ให้ Issuer สร้างเหรียญและส่งไปยัง account สนามเลือกตั้งตามจำนวนผู้มีสิทธิ์เลือกตั้งในพื้นที่นั้น ๆ จากนั้นในการขอเหรียญของ ประชาชนจะขอจาก account สนามเลือกตั้งแทน

Delay

เนื่องจากในการโหวตใช้ Blockchain ทำให้ในการสร้าง account, การโอนเหรียญจะต้องใช้เวลาในการ validate ทำให้มี delay เกิดขึ้น