# Practical Session

Developing Stellar Transactions

# Outline

- Review Sample Application
  - Two-Party Escrow
- Setup Dev Environment
- Develop on Stellar
- Tasks
  - Create Account
  - Create Assets
  - Send Payments
  - Create Offers
  - Make Path Payment
  - Listen for actions on account

# Escrow Account Demo

The premise of the demo is as follows:

- Tunde would like to buy a car.
- A car dealer agrees to sell Tunde a car for 2 million NGN under the condition that the car dealer will return Tunde's funds if the car develops issues after 2 days. We call this the unlock period.
- Tunde does not trust the car dealer so he will only pay the funds after two days.

To solve the premise outlined above they decide to set up an escrow that will ensure:

- Funds will only be released after 2 days to car dealer.
- Tunde can get his money back on the 3rd day if there are any issues.

While the demo describes a contract with NGN, any token can be used.

# Escrow Tasks

Our smart contract will consist of the following tasks

- **Generate Keypairs**: Generate Stellar public and private keypairs for each account.
- **Create Accounts**: Fund the accounts with their corresponding starting balances (in the demo, 10m NGN).
- **Parties Sign**: Add the target account as a signer to the escrow accounts
- **Start Hold Period**: Builds the transaction to unlock the funds in the escrow. This transaction can only be submitted after the unlock time has passed.
- **Start Refund Period**: Build the transaction to recover the funds in the escrow if it remains unclaimed. This transaction can only be submitted after the recovery time has passed.
- **Fund Escrow**: Sends the funds to the escrow account. Can be done multiple times.
- **Unlock Funds**: Runs the unlock transaction built previously.
- **Recover Funds**: Run the recovery transaction built previously.

## Escrow Demo

Demo Link: https://lightyeario.github.io/escrow-demo/

# Let’s Code

# Develop on Stellar

Goal

- Create a  set of stellar transaction that implements automated asset conversion

# Tasks

Create an app with endpoints that perform the following

- Generate Keypairs
- Fund Accounts
- Create Trustlines (For issuing assets)
- Create Offers on Stellar Distributed Exchange (SDEX)
- Make Path Payments with offers created

# Setup Environment

## Dev

- Horizon Server: We will use SDF testnet
- NodeJS
- NPM
- (optional) install nodemon package globally: `npm install -g nodemon`
- IDE; eg. Visual Studio Code
- PostMan
- Download boilerplate code from https://github.com/poliha/demo-api-boilerplate
- OR full source code from: https://github.com/poliha/demo-api

## Non Dev

- Browser: https://www.stellar.org/laboratory/

# Create your Issuing Account

We need an Issuing account that is already active on the test network

- We will use the friendbot to create this on testnet.
- On Stellar lab, go to “Account Creator”
- Click on Generate Keypair.
- Note down the public and private keys
- Click on “Fund this account …”
- Click on Get test network lumens

# Generate Keypairs

In tasks.js

```
const pair =
StellarSdk.Keypair.random();
    return {
      publicKey: pair.publicKey(),
      privateKey: pair.secret()
    };
```

In Stellar laboratory

- Click on “Account Creator” tab
- Click on the “Generate Keypair” button
- Note down the public and secret keys

# Create Accounts

```
const issuerDetail = await
horizonServer.loadAccount(ISSUER.publicKey);
   console.log('issuerDetails: ', issuerDetail);
   // build a transaction
 let transaction = new
StellarSdk.TransactionBuilder(issuerDetail);
   // add a create account operation
   let operationObj = {
     destination: req.body.publicKey,
     startingBalance: String(config.get("startingBalance"))
   };
transaction.addOperation(StellarSdk.Operation.createAccount(o
perationObj));
   // build and sign transaction
   let builtTx = transaction.build();

builtTx.sign(StellarSdk.Keypair.fromSecret(ISSUER.privateKey))
;
   // submit transaction to horizonServer
   let result = await horizonServer.submitTransaction(builtTx);
```

In Stellar laboratory

- Go to the transaction builder tab
- Enter Source Account(The public key of your issuer account)
- Click "fetch next sequence number"
- Scroll down to operation type. Select create account
- Enter the destination(public key of your sender/receiver account)
- Enter starting balance (e.g: 1000)
- Scroll down click "sign transaction in Signer"
- Enter the secret key of the Issuer Account
- Click on Submit to Post transaction
- Click on submit.
- Once the transaction is successful, repeat the same steps until you have 2 created accounts: sender and receiver.

# Status Check

Let us see what we have done so far.

- We will query the account details of each account, to see what changes have occured.
- On Stellar lab, go to “Endpoint explorer”
- Click on Accounts > Single Account
- Enter the Public key and click on Submit

  What is the Account balance?

# Create Trustlines

```
const receiverDetail = await
horizonServer.loadAccount(RECEIVER.publicKey);
  console.log('rcvrDetails: ', receiverDetail);
  // build a transaction
  let transaction = new
StellarSdk.TransactionBuilder(receiverDetail);
  // add a change trust operation
  let customAsset = new StellarSdk.Asset(ASSETCODE,
ISSUER.publicKey);
  let operationObj = {
    asset: customAsset
  };
transaction.addOperation(StellarSdk.Operation.changeTrust(ope
rationObj));
  // build and sign transaction
  let builtTx = transaction.build();
builtTx.sign(StellarSdk.Keypair.fromSecret(RECEIVER.privateKe
y));
  // submit transaction to horizonServer
  let result = await horizonServer.submitTransaction(builtTx);
```

In Stellar laboratory

- Go to the transaction builder tab
- Enter Source account (Receiver Account)
- Click on fetch next sequence number
- Scroll down, add select operation type: change trust
- Select Alphanumeric 4
- Enter Asset Code: RWF
- Enter Issuer Account ID: Issuer Account
- Scroll down click "sign transaction in Signer"
- Enter the secret key of the receiver Account
- Click on Submit to Post transaction
- Click on submit.

# Status Check

Let us see what we have done so far.

- We will query the account details of each account, to see what changes have occured.
- On Stellar lab, go to “Endpoint explorer”
- Click on Accounts > Single Account
- Enter the Public key and click on Submit

  Do you notice anything different?

# Create Offers

```
const issuerDetail = await
horizonServer.loadAccount(ISSUER.publicKey);
  // build a transaction
  let transaction = new StellarSdk.TransactionBuilder(issuerDetail);
  let customAsset = new StellarSdk.Asset(ASSETCODE,
ISSUER.publicKey);
  let operationObj = {
    selling: customAsset,
    buying: StellarSdk.Asset.native(),
    amount: String(5000000),
    price: 0.002, //500RWF = 1XLM
    offerId: 0
  };
transaction.addOperation(StellarSdk.Operation.manageOffer(operatio
nObj));
  // build and sign transaction
  let builtTx = transaction.build();
  builtTx.sign(StellarSdk.Keypair.fromSecret(ISSUER.privateKey));
  // submit transaction to horizonServer
  let result = await horizonServer.submitTransaction(builtTx);
```

In Stellar laboratory

- Go to the transaction builder tab
- Enter Source account (Issuer Account)
- Click on fetch next sequence number
- Scroll down, add select operation type: manage offer
- For Selling, select Alphanumeric 4
- Enter Asset Code: RWF
- Enter Issuer Account ID: Issuer Account
- For Buying, select native
- Enter amount 5000000
- Enter price: 0.002
- Enter offer Id: 0
- Scroll down click "sign transaction in Signer"
- Enter the secret key of the receiver Account
- Click on Submit to Post transaction
- Click on submit.

# Status Check

Let us see what we have done so far.

- We will query the orderbook on SDEX, to see what changes have occured.
- On Stellar lab, go to “Endpoint explorer”
- Click on Orderbook > Details
- For Selling: Click on Alphanum 4, Asset Code: RWF. Enter public key of your asset issuer
- For Buying, Click on native
- Click on Submit.

Can you see your offer?

# Path Payment

This operation requires 2 steps

- Find path for exchange to occur
- Make the path payment operation if path is found.

Stellar Labs does not have option for Path finding, so we will do this in programatically before implementing the path payment operation.

This part will be done in the code editor.

# Status Check

Let us see what we have done so far.

- We will query the account details of each account, to see what changes have occured.
- On Stellar lab, go to “Endpoint explorer”
- Click on Accounts > Single Account
- Enter the Public key and click on Submit

  Do you notice anything different?

# Outcome

- Accounts created
- Trustlines created
- Offers created
- Paths found for automatic asset exchange
- Payment was sent in XLM, recipient got RWF
- Stellar Rocks!!

# Conclusion

- Questions
- Comments
- Full source code available at https://github.com/poliha/demo-api
- More information on: https://www.stellar.org/developers

## Thank You