Rohan Bhattacharjee
4 AIML

Investigating how Blockchain can make Cross Border Payments faster and cost effective

Introduction

In this report, we investigate the creation of a blockchain-based cross-border payment system. By utilising the features of the Stellar blockchain, this project seeks to address the difficulties involved with cross-border transactions. We'll look into the application's technology, modules, and user experiences.

Project Overview

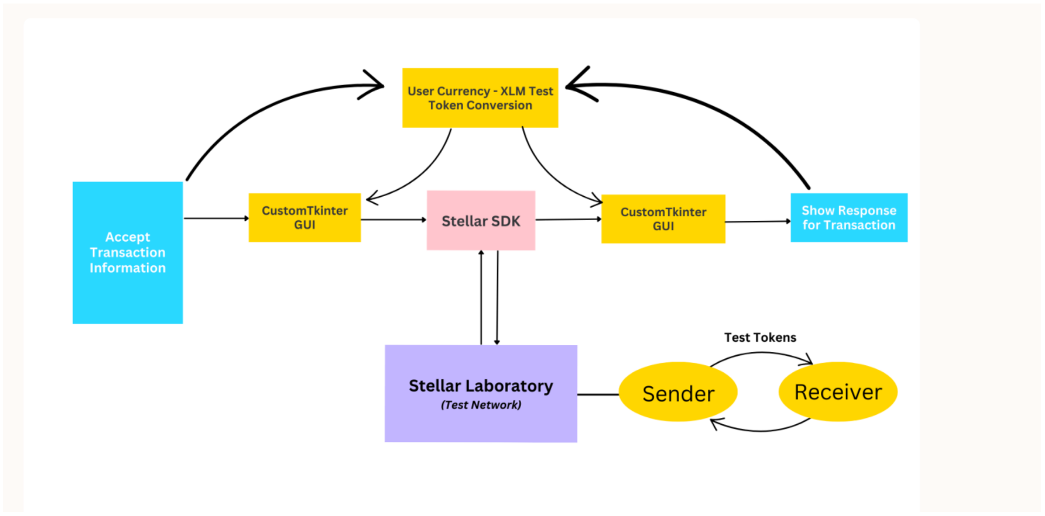
An essential component of financial technology, the simplification of cross-border payments, is the subject of our study. It seeks to offer an effective, user-friendly, and secure solution for cross-border transactions. The importance of this technology resides in its potential to completely alter how people and corporations send money across international borders.

Technologies and Libraries Used

We selected Stellar as our blockchain platform, known for its fast and cost-effective transactions.
Stellar SDK: The Stellar SDK played a pivotal role in integrating the Stellar blockchain into our application.
CustomTkinter: We enhanced the user interface using CustomTkinter, a custom GUI library designed for a visually appealing and user-friendly experience.

Coding Architecture

Our application is structured into modules, each serving a specific purpose. These modules work in harmony to create a seamless user experience and ensure the secure execution of cross-border transactions.

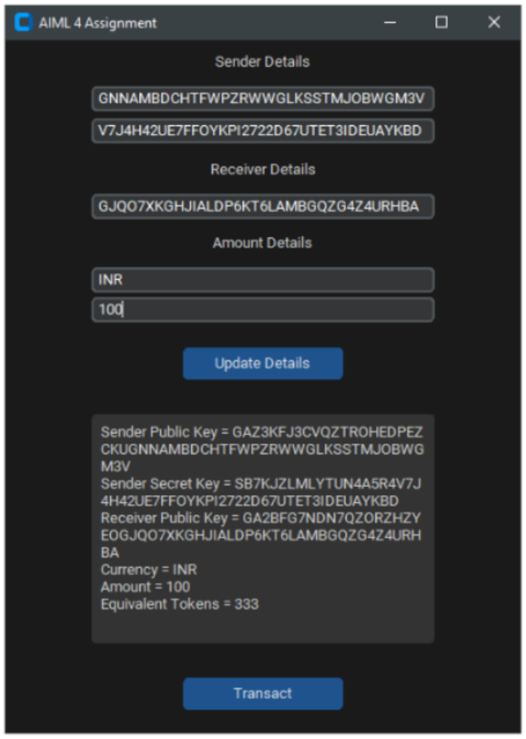

User Currency - XLM Test Token Conversion
Accept Transaction Information
CustomTkinter GUI
Stellar SDK
CustomTkinter GUI
Show Response for Transaction
Stellar Laboratory (Test Network)
Test Tokens
Sender
Receiver

Python Script:

```
from tkinter import *
import customtkinter
from stellar_sdk import TransactionBuilder, Asset, Network, Server, Keypair


customtkinter.set_appearance_mode("dark")
customtkinter.set_default_color_theme("dark-blue")

# Initialize global variables
your_public_key = ""
your_secret_key = ""
recipient_public_key = ""
cur = ""
val = ""
amount_in_xlm = 0


conversion_rates = {
    "USD": 0.5,  # 1 USD = 0.5 XLM
    "EUR": 0.4,  # 1 EUR = 0.4 XLM
    "INR": 0.3  # 1 INR = 0.3 XLM
}

rate = 0

# Function to capture user details
def details():
    global your_public_key
    global your_secret_key
    global recipient_public_key
    global cur
    global val
    global conversion_rates
    global amount_in_xlm
    global rate

    your_public_key = S_pk.get()
    your_secret_key = S_sk.get()
    recipient_public_key = R_pk.get()
    cur = currency.get()
    val = value.get()
    rate = conversion_rates[cur]
    amount_in_xlm = int(int(val) / rate)
    textbox.delete(1.0, END)
    deets = f"Sender Public Key: {your_public_key}\nSender Secret Key:
{your_secret_key}\nReceiver Public Key: {recipient_public_key}\nCurrency: {cur}\nAmount:
{val}\nEquivalent Tokens: {amount_in_xlm}"
    confirm.configure(text="Update Details")
    textbox.insert(END, deets, tags=None)

def transaction():
    global rate
    global your_public_key
    global your_secret_key
    global recipient_public_key
    global amount_in_xlm
    global cur

    stellar_server = Server("https://horizon-testnet.stellar.org")
    your_keypair = Keypair.from_secret(your_secret_key)
    your_account = stellar_server.load_account(your_keypair.public_key)
    asset = Asset("XLM")

    transaction = TransactionBuilder(
        source_account=your_account,
        network_passphrase=Network.TESTNET_NETWORK_PASSPHRASE,
        base_fee=100,
    ).append_payment_op(
        destination=recipient_public_key,
        asset=asset,  # Specify the asset (XLM)
        amount=str(amount_in_xlm),
    ).set_timeout(30).build()

    transaction.sign(your_keypair)
    try:
        response = stellar_server.submit_transaction(transaction)

        def check_transaction_status(transaction_hash):
            transaction = stellar_server.transactions().transaction(transaction_hash).call()
            return transaction['successful']

        status = check_transaction_status(response['hash'])
        textbox.delete(1.0, END)
        if status:
            received_xlm = amount_in_xlm
            received_amount_in_currency = received_xlm * rate
            deets = f"Transaction Successful!\n\nTransaction Hash: {response['hash']}\n\nSent
{received_xlm} XLM, Equivalent to {received_amount_in_currency:.2f} {cur}"
            textbox.insert(END, deets, tags=None)
        else:
            textbox.insert(END, f"Transaction Failed {status}", tags=None)

    except Exception as e:
        textbox.delete(1.0, END)
        textbox.insert(END, "Transaction Failed", tags=None)

root = customtkinter.CTk()
root.title('Stellar Transaction App')
root.geometry("450x600")

sender = customtkinter.CTkLabel(root, text="Sender Details", justify="left")
sender.pack(padx=5, pady=5)

S_pk = customtkinter.CTkEntry(root, height=10, width=300, text_color="white",
placeholder_text="Enter Sender's Public Key")
S_pk.pack(padx=2, pady=2)

S_sk = customtkinter.CTkEntry(root, height=10, width=300, text_color="white",
placeholder_text="Enter Sender's Secret Key")
S_sk.pack(padx=4, pady=4)

receiver = customtkinter.CTkLabel(root, text="Receiver Details", justify="left")
receiver.pack(padx=5, pady=5)

R_pk = customtkinter.CTkEntry(root, height=10, width=300, text_color="white",
placeholder_text="Enter Receiver's Public Key")
R_pk.pack(padx=2, pady=2)
```

```
amount = customtkinter.CTkLabel(root, text="Amount Details", justify="left")
amount.pack(padx=5, pady=5)

currency = customtkinter.CTkEntry(root, height=10, width=300, text_color="white",
placeholder_text="Enter Currency (e.g., INR, USD, EUR)")
currency.pack(padx=2, pady=2)

value = customtkinter.CTkEntry(root, height=10, width=300, text_color="white",
placeholder_text="Enter Currency Value")
value.pack(padx=2, pady=2)

confirm = customtkinter.CTkButton(root, text='Confirm Details', command=details)
confirm.pack(padx=20, pady=20)

details_frame = customtkinter.CTkFrame(root, height=200, width=300, corner_radius=10)
details_frame.pack(padx=10, pady=10)

textbox = customtkinter.CTkTextbox(details_frame, height=200, width=300)
textbox.pack()

transact = customtkinter.CTkButton(root, text='Execute Transaction', command=transaction)
transact.pack(padx=20, pady=20)

root.mainloop()
```

Application Snapshot:



Sender Details

GNNAMBDCHTFWPZRWWGLKSSTMJOBWGM3V

V7J4H42UE7FFOYKPI2722D67UTET3IDEUAYKBD

Receiver Details

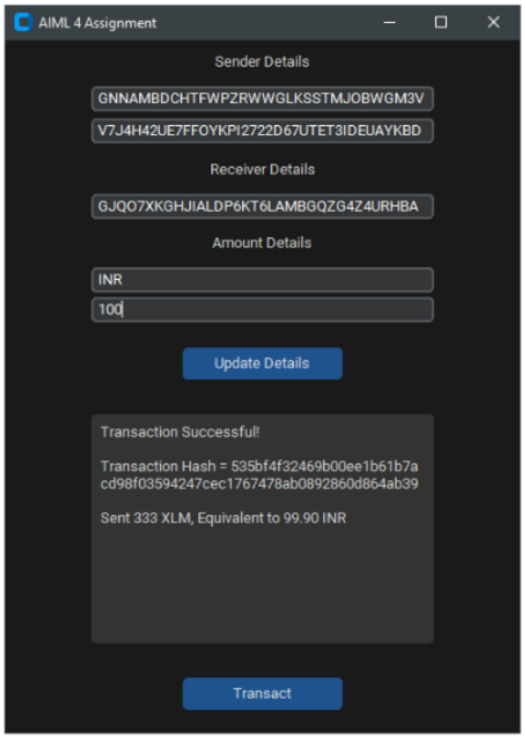
GJQO7XKGHJIALDP6KT6LAMBGQZG4Z4URHBA

Amount Details

INR

100

```
Sender Public Key = GAZ3KFJ3CVQZTROHEDPEZ
CKUGNNAMBDCHTFWPZRWWGLKSSTMJOBWG
M3V
Sender Secret Key = SB7KJZLMLYTUN4A5R4V7J
4H42UE7FFOYKPI2722D67UTET3IDEUAYKBD
Receiver Public Key = GA2BFG7NDN7QZORZHZY
EOGJQO7XKGHJIALDP6KT6LAMBGQZG4Z4URH
BA
Currency = INR
Amount = 100
Equivalent Tokens = 333
```



Sender Details

GNNAMBDCHTFWPZRWWGLKSSTMJOBWGM3V

V7J4H42UE7FFOYKPI2722D67UTET3IDEUAYKBD

Receiver Details

GJQO7XKGHJIALDP6KT6LAMBGQZG4Z4URHBA

Amount Details

INR

100

```
Transaction Successful!

Transaction Hash = 535bf4f32469b00ee1b61b7a
cd98f03594247cec1767478ab0892860d864ab39

Sent 333 XLM, Equivalent to 99.90 INR
```

Conclusion

In conclusion, our project shows a very basic usage of the vast Blockchain technology in the field of cross border payments. There are more complex and detailed solutions which are in practice in today's world. With more advancements it is only a matter of time until we experience these applications on a regular basis.