Documentation of XRPL Transaction

Problem Statement :- XRPL Payment Channels and Streaming Payments

For this we need follow following Steps given below:-

Step 1: Set up a Payment Channel

- **Install XRPL Development Tools:** You'll need to install the necessary tools and libraries to interact with the XRPL. You can use libraries like ripple-lib for JavaScript.
- **Generate XRPL Accounts:** Generate two XRPL accounts (sender and receiver) using the XRPL tools. You'll need the secret keys and public addresses for both accounts.
- **Fund the Sender Account**: To create a payment channel, you'll need to fund the sender's account with enough XRP to cover the planned streaming payments and transaction fees.

- **Open Payment Channel:** The sender initiates the payment channel by creating a transaction to open a payment channel to the receiver's account. This involves locking a specific amount of XRP in the channel.

- **Generate Payment Channel Transaction:** Use your chosen development library to create and sign the transaction that opens the payment channel. The transaction should specify the channel amount and the destination (receiver's) account.

- **Submit Transaction:** Submit the transaction to the XRPL network using your chosen library. This will create the payment channel on the XRPL.

Step 2: Initiate a Streaming Payment

- **Create a Payment Channel Claim:** The sender will create a "claim" transaction that specifies the amount of XRP they intend to send per second. This claim transaction will be sent repeatedly to update the channel's state.

- **Sign and Submit Claim Transaction:** Sign the claim transaction using the sender's secret key and submit it to the XRPL network. This claim transaction is the mechanism for streaming payments.

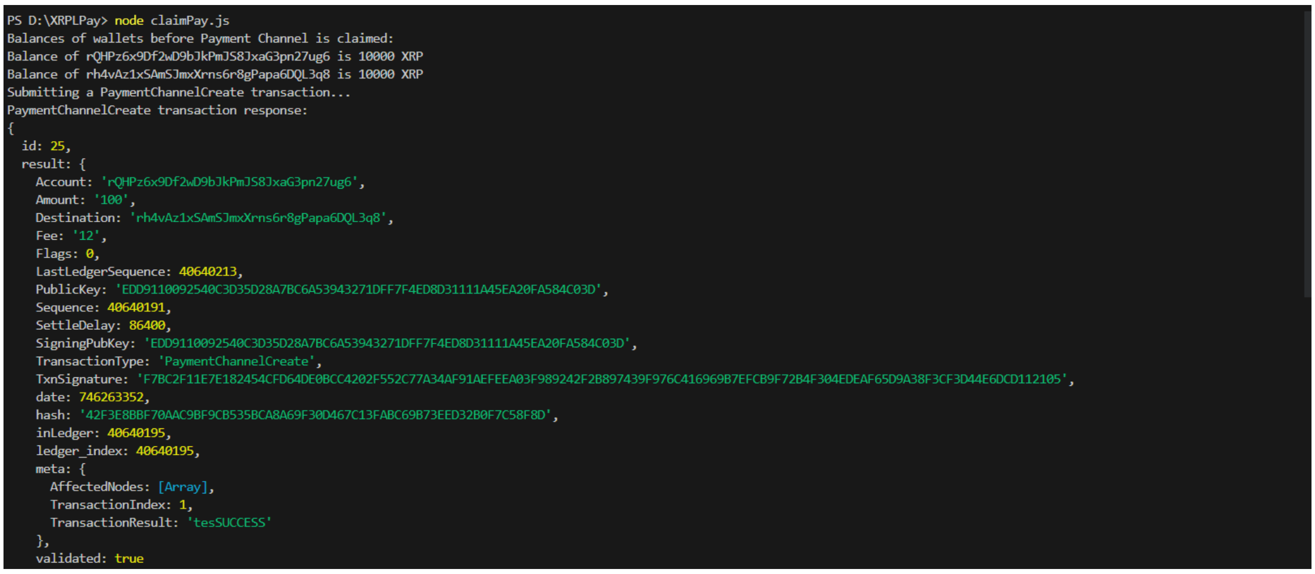
Step 3: Monitor and Calculate Total Amount Transferred

- **Monitor Channel State:** Both the sender and receiver can monitor the payment channel's state. This involves checking the latest claim transaction's details, such as the current amount sent.

- **Calculate Total Amount:** To calculate the total amount transferred over a specific time period, you'll need to keep track of the claimed amounts and their timestamps. By comparing timestamps, you can determine how much XRP was sent during the specified time frame.

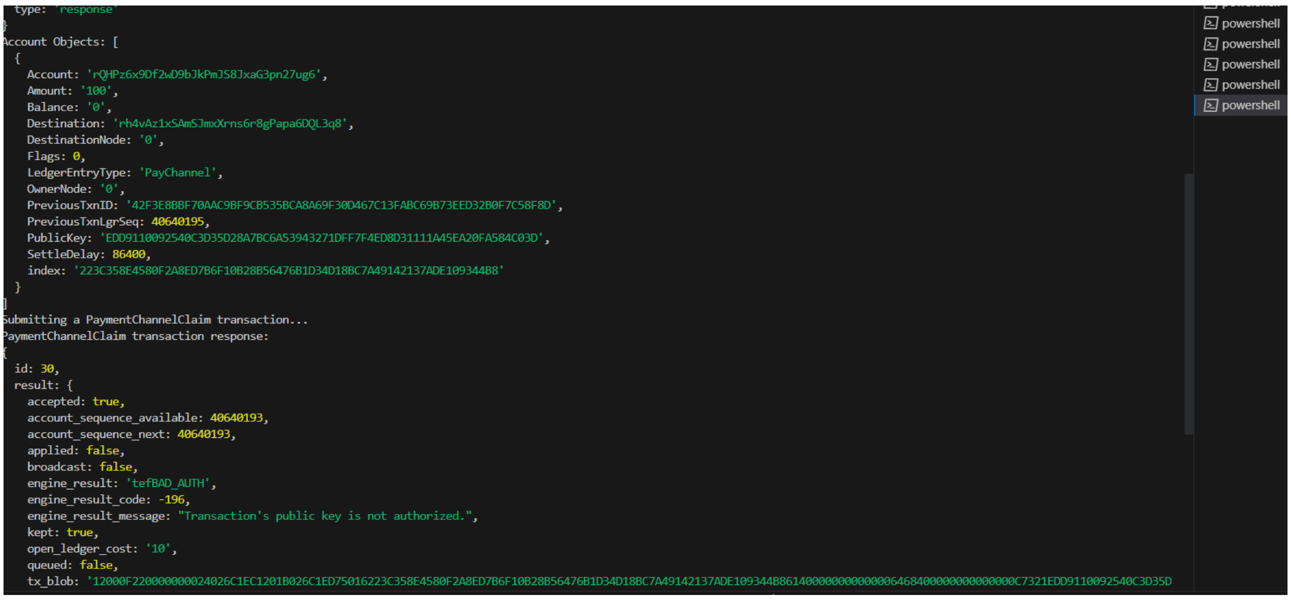
Screen Shot and Diagram:-

Output:



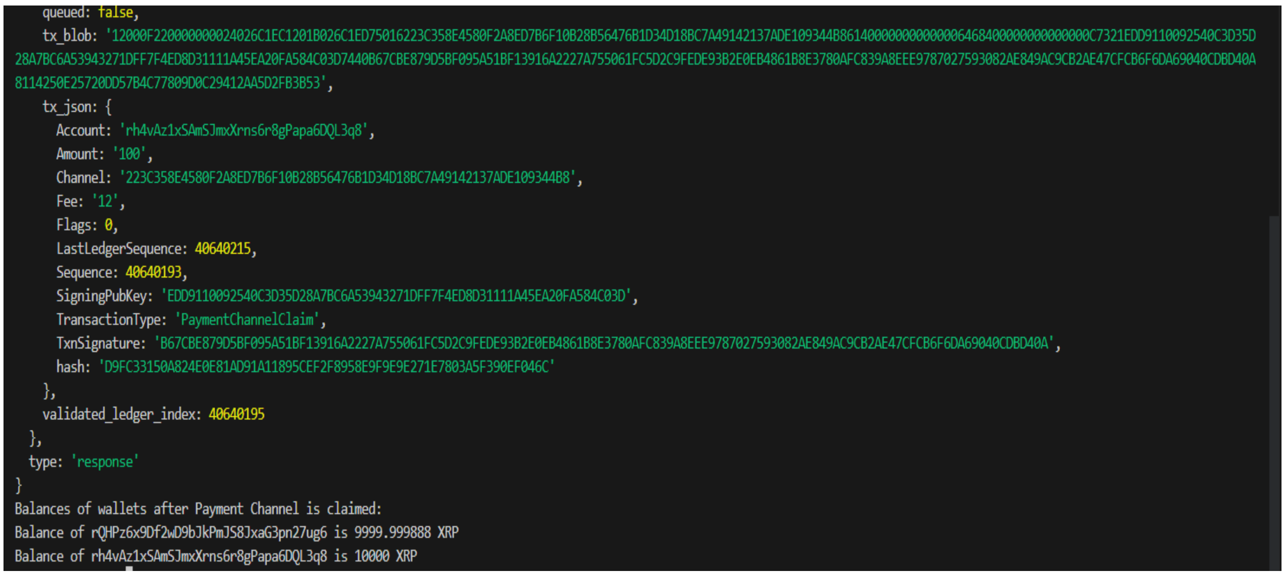
```
PS D:\XRPLPay> node claimPay.js
Balances of wallets before Payment Channel is claimed:
Balance of rQHPz6x9Df2wD9bJkPmJS8JxaG3pn27ug6 is 10000 XRP
Balance of rh4vAz1xSAmSJmxXrns6r8gPapa6DQL3q8 is 10000 XRP
Submitting a PaymentChannelCreate transaction...
PaymentChannelCreate transaction response:
{
  id: 25,
  result: {
    Account: 'rQHPz6x9Df2wD9bJkPmJS8JxaG3pn27ug6',
    Amount: '100',
    Destination: 'rh4vAz1xSAmSJmxXrns6r8gPapa6DQL3q8',
    Fee: '12',
    Flags: 0,
    LastLedgerSequence: 40640213,
    PublicKey: 'EDD9110092540C3D35D28A7BC6A53943271DFF7F4ED8D31111A45EA20FA584C03D',
    Sequence: 40640191,
    SettleDelay: 86400,
    SigningPubKey: 'EDD9110092540C3D35D28A7BC6A53943271DFF7F4ED8D31111A45EA20FA584C03D',
    TransactionType: 'PaymentChannelCreate',
    TxnSignature: 'F7BC2F11E7E182454CFD64DE0BCC4202F552C77A34AF91AEFEEA03F989242F2B897439F976C416969B7EFCB9F72B4F304EDEAF65D9A38F3CF3D44E6DCD112105',
    date: 746263352,
    hash: '42F3E8BBF70AAC9BF9CB535BCA8A69F30D467C13FABC69B73EED32B0F7C58F8D',
    inLedger: 40640195,
    ledger_index: 40640195,
    meta: {
      AffectedNodes: [Array],
      TransactionIndex: 1,
      TransactionResult: 'tesSUCCESS'
    },
    validated: true
```

validation



```
  type: 'response'
}
Account Objects: [
  {
    Account: 'rQHPz6x9Df2wD9bJkPmJS8JxaG3pn27ug6',
    Amount: '100',
    Balance: '0',
    Destination: 'rh4vAz1xSAmSJmxXrns6r8gPapa6DQL3q8',
    DestinationNode: '0',
    Flags: 0,
    LedgerEntryType: 'PayChannel',
    OwnerNode: '0',
    PreviousTxnID: '42F3E8BBF70AAC9BF9CB535BCA8A69F30D467C13FABC69B73EED32B0F7C58F8D',
    PreviousTxnLgrSeq: 40640195,
    PublicKey: 'EDD9110092540C3D35D28A7BC6A53943271DFF7F4ED8D31111A45EA20FA584C03D',
    SettleDelay: 86400,
    index: '223C358E4580F2A8ED7B6F10B28B56476B1D34D18BC7A49142137ADE109344B8'
  }
]
Submitting a PaymentChannelClaim transaction...
PaymentChannelClaim transaction response:
{
  id: 30,
  result: {
    accepted: true,
    account_sequence_available: 40640193,
    account_sequence_next: 40640193,
    applied: false,
    broadcast: false,
    engine_result: 'tefBAD_AUTH',
    engine_result_code: -196,
    engine_result_message: "Transaction's public key is not authorized.",
    kept: true,
    open_ledger_cost: '10',
    queued: false,
    tx_blob: '12000F220000000024026C1EC1201B026C1ED75016223C358E4580F2A8ED7B6F10B28B56476B1D34D18BC7A49142137ADE109344B8614000000000000064684000000000000000C7321EDD9110092540C3D35D
```

Result



```
    queued: false,
    tx_blob: '12000F220000000024026C1EC1201B026C1ED75016223C358E4580F2A8ED7B6F10B28B56476B1D34D18BC7A49142137ADE109344B8614000000000000064684000000000000000C7321EDD9110092540C3D35D
28A7BC6A53943271DFF7F4ED8D31111A45EA20FA584C03D7440B67CBE879D5BF095A51BF13916A2227A755061FC5D2C9FEDE93B2E0EB4861B8E3780AFC839A8EEE9787027593082AE849AC9CB2AE47CFCB6F6DA69040CDBD40A
8114250E25720DD57B4C77809D0C29412AA5D2FB3B53',
    tx_json: {
      Account: 'rh4vAz1xSAmSJmxXrns6r8gPapa6DQL3q8',
      Amount: '100',
      Channel: '223C358E4580F2A8ED7B6F10B28B56476B1D34D18BC7A49142137ADE109344B8',
      Fee: '12',
      Flags: 0,
      LastLedgerSequence: 40640215,
      Sequence: 40640193,
      SigningPubKey: 'EDD9110092540C3D35D28A7BC6A53943271DFF7F4ED8D31111A45EA20FA584C03D',
      TransactionType: 'PaymentChannelClaim',
      TxnSignature: 'B67CBE879D5BF095A51BF13916A2227A755061FC5D2C9FEDE93B2E0EB4861B8E3780AFC839A8EEE9787027593082AE849AC9CB2AE47CFCB6F6DA69040CDBD40A',
      hash: 'D9FC33150A824E0E81AD91A11895CEF2F8958E9F9E9E271E7803A5F390EF046C'
    },
    validated_ledger_index: 40640195
  },
  type: 'response'
}
Balances of wallets after Payment Channel is claimed:
Balance of rQHPz6x9Df2wD9bJkPmJS8JxaG3pn27ug6 is 9999.999888 XRP
Balance of rh4vAz1xSAmSJmxXrns6r8gPapa6DQL3q8 is 10000 XRP
```

Problem Statement :- XRPL Decentralized Exchange (DEX) Integration

For this we need follow following Steps given below:-

Step 1: Choose Tokens

Choose two different tokens issued on the XRPL for the token swap, such as Token A and Token B.

Step 2: Write the Script

Write a script that integrates with the XRPL DEX to perform a token swap. Here's a simplified outline of the script's functionality:

Import Libraries: Import the necessary libraries for interacting with the XRPL. You might use libraries ripple-lib (JavaScript).

Initialize Account Information: Set up the account credentials for the user placing the limit order. You'll need the user's XRPL address and secret key.

Place a Limit Order: Create a limit order transaction specifying the following:

- The source token (Token A) and the destination token (Token B).
- The amount of Token A the user wants to trade.
- The desired exchange rate between Token A and Token B.
- Set the order expiration time.

Sign and Submit Transaction: Sign the limit order transaction using the user's secret key and submit it to the XRPL network.

Step 3: Monitor the Order Book

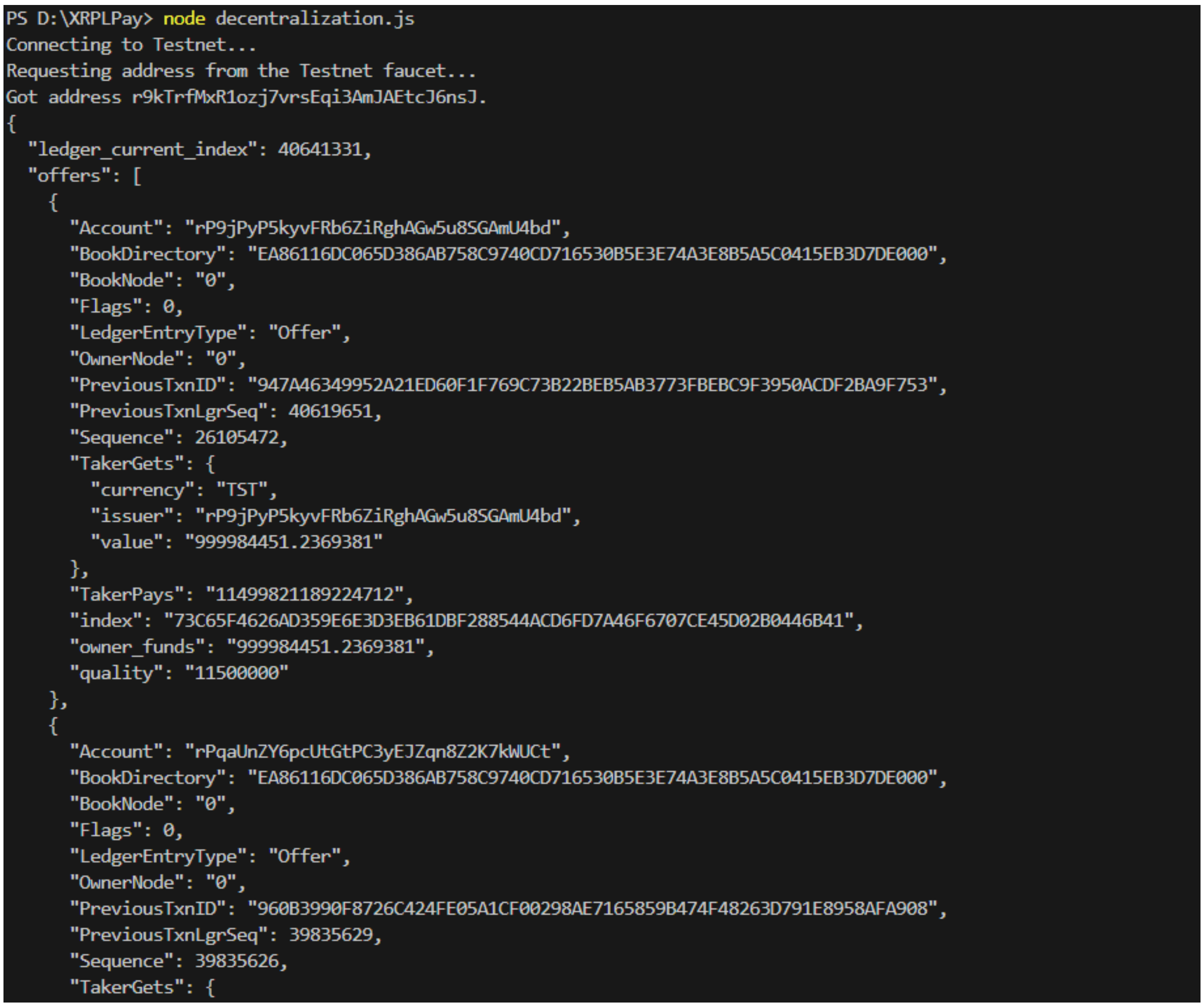
To monitor the order book and manage the order's lifecycle:

Check Order Status: Use the XRPL libraries to periodically check the status of the placed limit order. You can query the XRPL to get the latest information about the order, such as whether it has been partially filled, fully filled, or canceled.

Modify or Cancel Order: Based on market conditions, you can modify the order by adjusting the exchange rate or cancel the order if needed. To modify, create a new limit order transaction with the updated parameters and submit it to the XRPL.

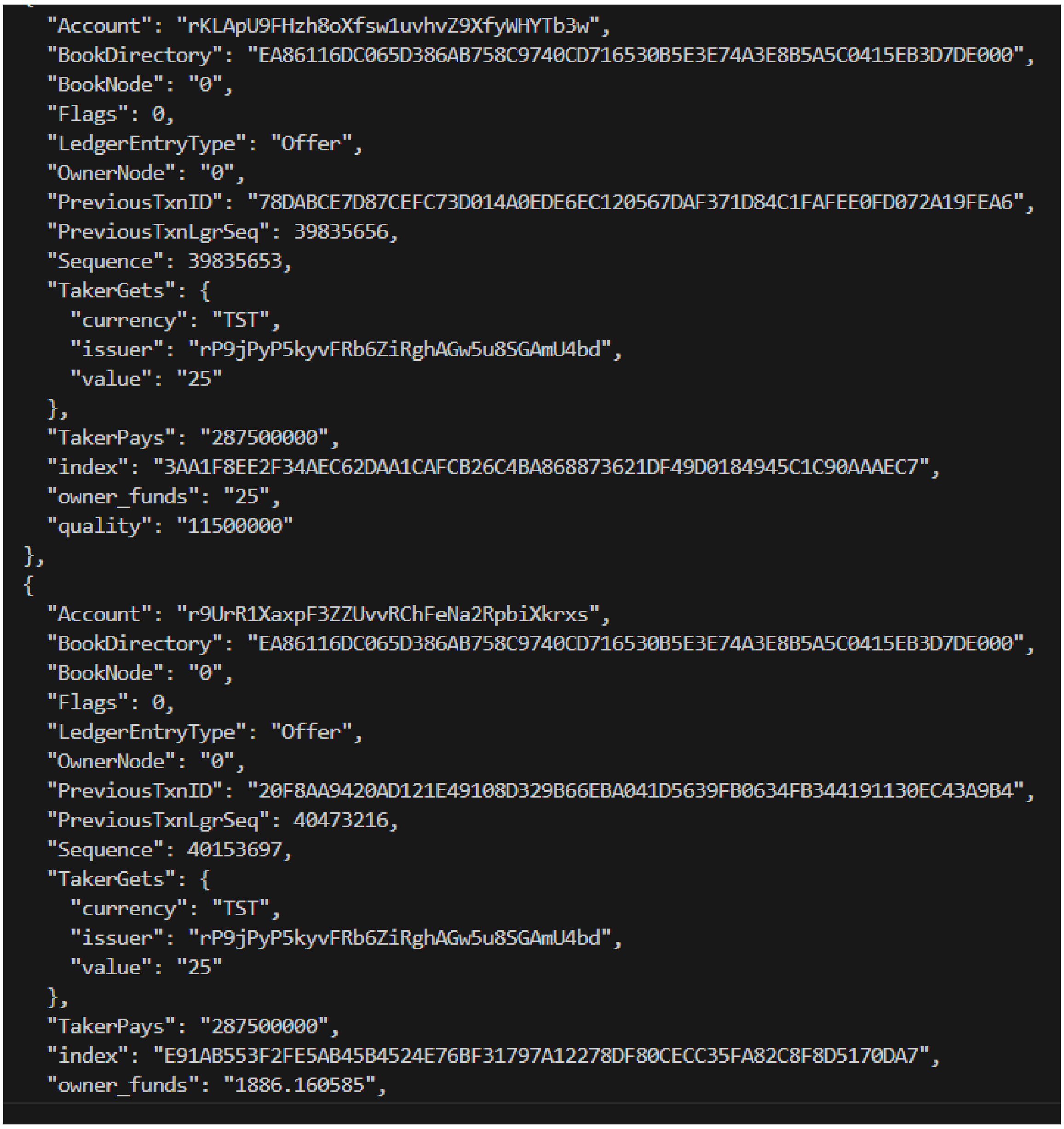
Screen Shot and Diagram:-

Output:



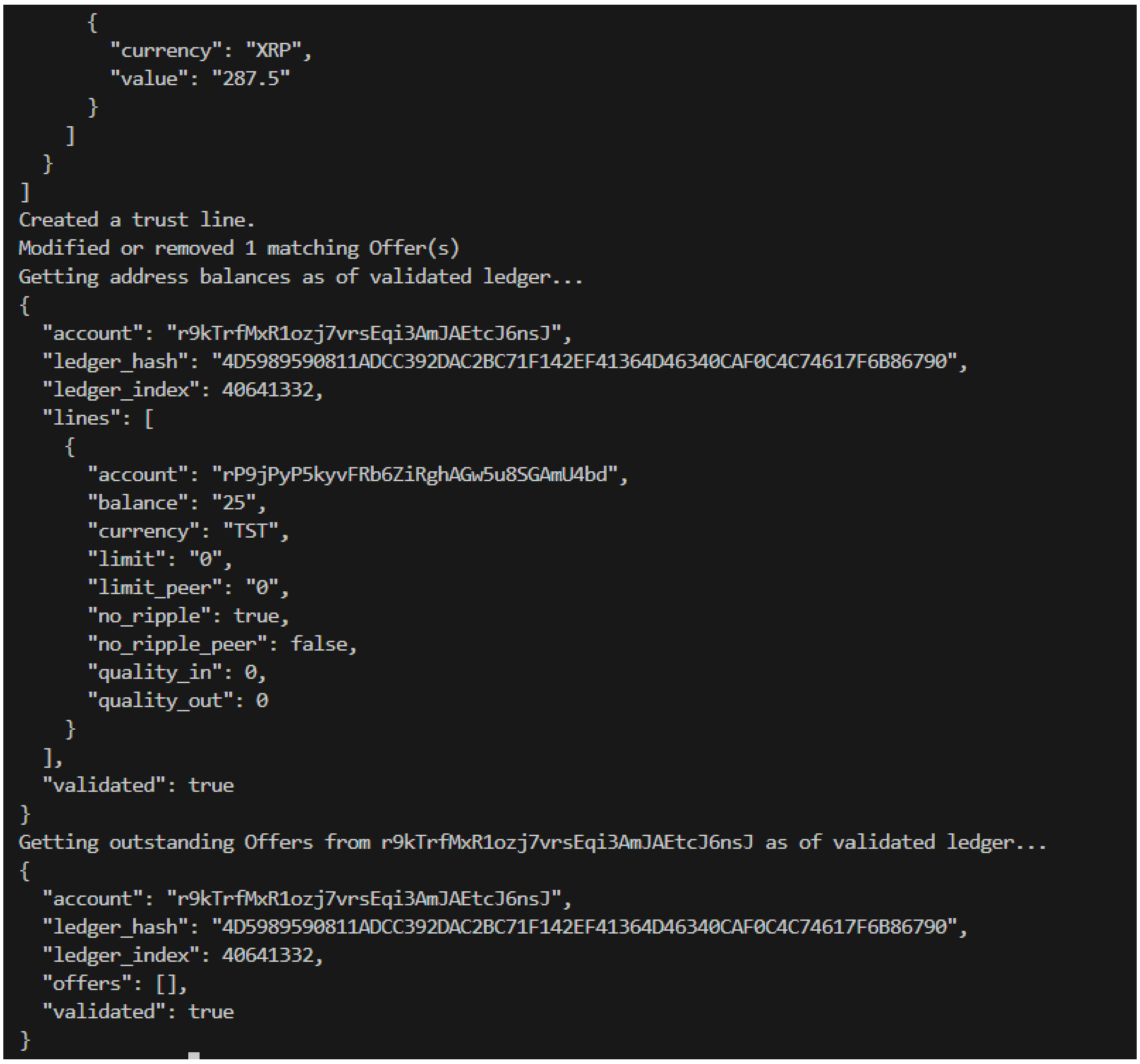
```
PS D:\XRPLPay> node decentralization.js
Connecting to Testnet...
Requesting address from the Testnet faucet...
Got address r9kTrfMxR1ozj7vrsEqi3AmJAEtcJ6nsJ.
{
  "ledger_current_index": 40641331,
  "offers": [
    {
      "Account": "rP9jPyP5kyvFRb6ZiRghAGw5u8SGAmU4bd",
      "BookDirectory": "EA86116DC065D386AB758C9740CD716530B5E3E74A3E8B5A5C0415EB3D7DE000",
      "BookNode": "0",
      "Flags": 0,
      "LedgerEntryType": "Offer",
      "OwnerNode": "0",
      "PreviousTxnID": "947A46349952A21ED60F1F769C73B22BEB5AB3773FBEBC9F3950ACDF2BA9F753",
      "PreviousTxnLgrSeq": 40619651,
      "Sequence": 26105472,
      "TakerGets": {
        "currency": "TST",
        "issuer": "rP9jPyP5kyvFRb6ZiRghAGw5u8SGAmU4bd",
        "value": "999984451.2369381"
      },
      "TakerPays": "11499821189224712",
      "index": "73C65F4626AD359E6E3D3EB61DBF288544ACD6FD7A46F6707CE45D02B0446B41",
      "owner_funds": "999984451.2369381",
      "quality": "11500000"
    },
    {
      "Account": "rPqaUnZY6pcUtGtPC3yEJZqn8Z2K7kWUCt",
      "BookDirectory": "EA86116DC065D386AB758C9740CD716530B5E3E74A3E8B5A5C0415EB3D7DE000",
      "BookNode": "0",
      "Flags": 0,
      "LedgerEntryType": "Offer",
      "OwnerNode": "0",
      "PreviousTxnID": "960B3990F8726C424FE05A1CF00298AE7165859B474F48263D791E8958AFA908",
      "PreviousTxnLgrSeq": 39835629,
      "Sequence": 39835626,
      "TakerGets": {
```

Validation:



```
    "Account": "rKLApU9FHzh8oXfsw1uvhvZ9XfyWHYTb3w",
    "BookDirectory": "EA86116DC065D386AB758C9740CD716530B5E3E74A3E8B5A5C0415EB3D7DE000",
    "BookNode": "0",
    "Flags": 0,
    "LedgerEntryType": "Offer",
    "OwnerNode": "0",
    "PreviousTxnID": "78DABCE7D87CEFC73D014A0EDE6EC120567DAF371D84C1FAFEE0FD072A19FEA6",
    "PreviousTxnLgrSeq": 39835656,
    "Sequence": 39835653,
    "TakerGets": {
      "currency": "TST",
      "issuer": "rP9jPyP5kyvFRb6ZiRghAGw5u8SGAmU4bd",
      "value": "25"
    },
    "TakerPays": "287500000",
    "index": "3AA1F8EE2F34AEC62DAA1CAFCB26C4BA868873621DF49D0184945C1C90AAAEC7",
    "owner_funds": "25",
    "quality": "11500000"
  },
  {
    "Account": "r9UrR1XaxpF3ZZUvvRChFeNa2RpbiXkrxs",
    "BookDirectory": "EA86116DC065D386AB758C9740CD716530B5E3E74A3E8B5A5C0415EB3D7DE000",
    "BookNode": "0",
    "Flags": 0,
    "LedgerEntryType": "Offer",
    "OwnerNode": "0",
    "PreviousTxnID": "20F8AA9420AD121E49108D329B66EBA041D5639FB0634FB344191130EC43A9B4",
    "PreviousTxnLgrSeq": 40473216,
    "Sequence": 40153697,
    "TakerGets": {
      "currency": "TST",
      "issuer": "rP9jPyP5kyvFRb6ZiRghAGw5u8SGAmU4bd",
      "value": "25"
    },
    "TakerPays": "287500000",
    "index": "E91AB553F2FE5AB45B4524E76BF31797A12278DF80CECC35FA82C8F8D5170DA7",
    "owner_funds": "1886.160585",
```

Result:



```
      {
        "currency": "XRP",
        "value": "287.5"
      }
    ]
  }
]
Created a trust line.
Modified or removed 1 matching Offer(s)
Getting address balances as of validated ledger...
{
  "account": "r9kTrfMxR1ozj7vrsEqi3AmJAEtcJ6nsJ",
  "ledger_hash": "4D5989590811ADCC392DAC2BC71F142EF41364D46340CAF0C4C74617F6B86790",
  "ledger_index": 40641332,
  "lines": [
    {
      "account": "rP9jPyP5kyvFRb6ZiRghAGw5u8SGAmU4bd",
      "balance": "25",
      "currency": "TST",
      "limit": "0",
      "limit_peer": "0",
      "no_ripple": true,
      "no_ripple_peer": false,
      "quality_in": 0,
      "quality_out": 0
    }
  ],
  "validated": true
}
Getting outstanding Offers from r9kTrfMxR1ozj7vrsEqi3AmJAEtcJ6nsJ as of validated ledger...
{
  "account": "r9kTrfMxR1ozj7vrsEqi3AmJAEtcJ6nsJ",
  "ledger_hash": "4D5989590811ADCC392DAC2BC71F142EF41364D46340CAF0C4C74617F6B86790",
  "ledger_index": 40641332,
  "offers": [],
  "validated": true
}
```

Problem Statement :- : XRPL Cross-Border Payments and Compliance

For this we need follow following Steps given below:-

Step 1: Choosing Fiat Currencies and XRPL Accounts:-

let's choose USD and EUR as the fiat currencies. We'll also assume two XRPL accounts, one located in the United States and another in the European Union.

Step 2: Cross-Border Payment Plan:-

Initiating the Payment:

- Account A (USD) initiates a payment to Account B (EUR).

- Account A sends a payment transaction on the XRPL network, specifying the recipient as Account B and the amount in USD.

Decentralized Exchange Conversion:

- The XRPL's built-in decentralized exchange automatically processes the payment conversion.
- The XRPL matches the USD-to-EUR order with available offers on the exchange.
- The conversion rate is determined based on the available offers at the time of the transaction.

Payment Settlement:

- Once the conversion is executed, Account B receives the converted amount in EUR.
- The transaction is recorded on the XRPL's ledger.

Step 3: Compliance Checks (KYC and AML):-
To ensure compliance with KYC and AML regulations, the following steps are taken:

a) Know Your Customer (KYC) Checks:-

- Both Account A and Account B are required to complete KYC verification with their respective XRPL service providers.
- This involves providing identity documents, proof of address, and other required information.
- The service providers validate the provided information and approve the accounts for cross-border transactions.

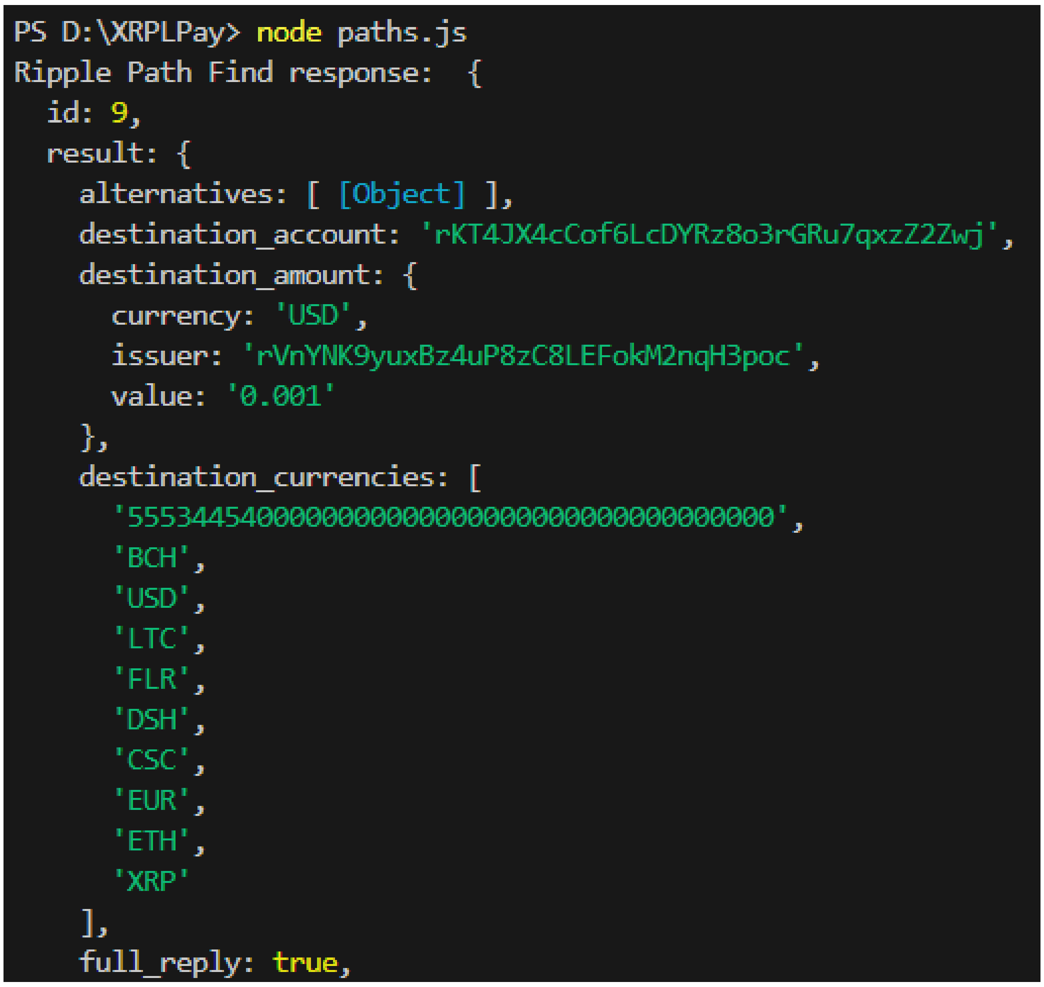
b) Anti-Money Laundering (AML) Checks:

- During the payment initiation and conversion process, the XRPL's decentralized exchange platform monitors transaction patterns.

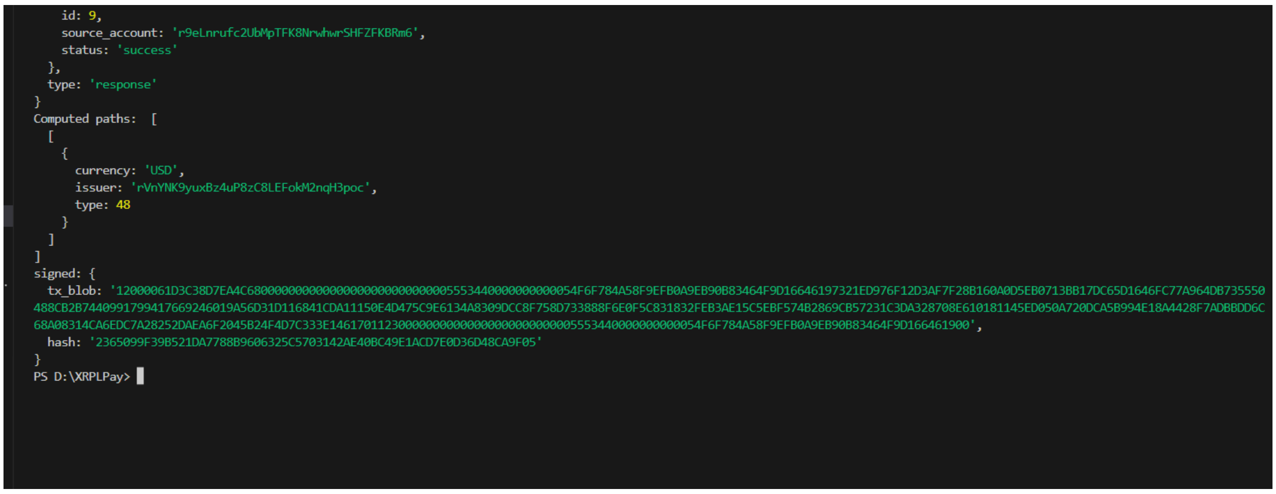
- Unusual or suspicious transactions, such as large or frequent transfers, trigger alerts for further AML investigation.
- If a transaction is flagged as potentially suspicious, additional documentation and information may be requested from the account holders.

Screen Shot and Diagram:-

Output:



```
PS D:\XRPLPay> node paths.js
Ripple Path Find response:  {
  id: 9,
  result: {
    alternatives: [ [Object] ],
    destination_account: 'rKT4JX4cCof6LcDYRz8o3rGRu7qxzZ2Zwj',
    destination_amount: {
      currency: 'USD',
      issuer: 'rVnYNK9yuxBz4uP8zC8LEFokM2nqH3poc',
      value: '0.001'
    },
    destination_currencies: [
      '5553445400000000000000000000000000000000',
      'BCH',
      'USD',
      'LTC',
      'FLR',
      'DSH',
      'CSC',
      'EUR',
      'ETH',
      'XRP'
    ],
    full_reply: true,
```



```
    id: 9,
    source_account: 'r9eLnrufc2UbMpTFK8NrwhwrSHFZFKBRm6',
    status: 'success'
  },
  type: 'response'
}
Computed paths:  [
  [
    {
      currency: 'USD',
      issuer: 'rVnYNK9yuxBz4uP8zC8LEFokM2nqH3poc',
      type: 48
    }
  ]
]
signed: {
  tx_blob: '12000061D3C38D7EA4C68000000000000000000000000000000555344000000000000054F6F784A58F9EFB0A9EB90B83464F9D16646197321ED976F12D3AF7F28B160A0D5EB0713BB17DC65D1646FC77A964DB735550
488CB2B744099179941766924601 9A56D31D116841CDA11150E4D475C9E6134A8309DCC8F758D733888F6E0F5C831832FEB3AE15C5EBF574B2869CB57231C3DA328708E610181145ED050A720DCA5B994E18A4428F7ADBBDD6C
68A08314CA6EDC7A28252DAEA6F2045B24F4D7C333E14617011230000000000000000000000000000000555344000000000000054F6F784A58F9EFB0A9EB90B83464F9D166461900',
  hash: '2365099F39B521DA7788B9606325C5703142AE40BC49E1ACD7E0D36D48CA9F05'
}
PS D:\XRPLPay> █
```