RanchiMall Tron Web Wallet Tasks

Task 1 : Address Lookup

Allow search for any Tron blockchain address and display full transaction history.

Context :

The user enters a valid Tron address (starting with T). The system connects to the TronGrid (Shasta testnet or Mainnet) and:

1. Fetches the latest transactions made by that address
2. Displays important details like:
   - Transaction type
   - Amount sent or received
   - Sender,receiver and contract addresses
   - Transaction result
   - Date and time
   - Transaction Fee
   - Block number / hash

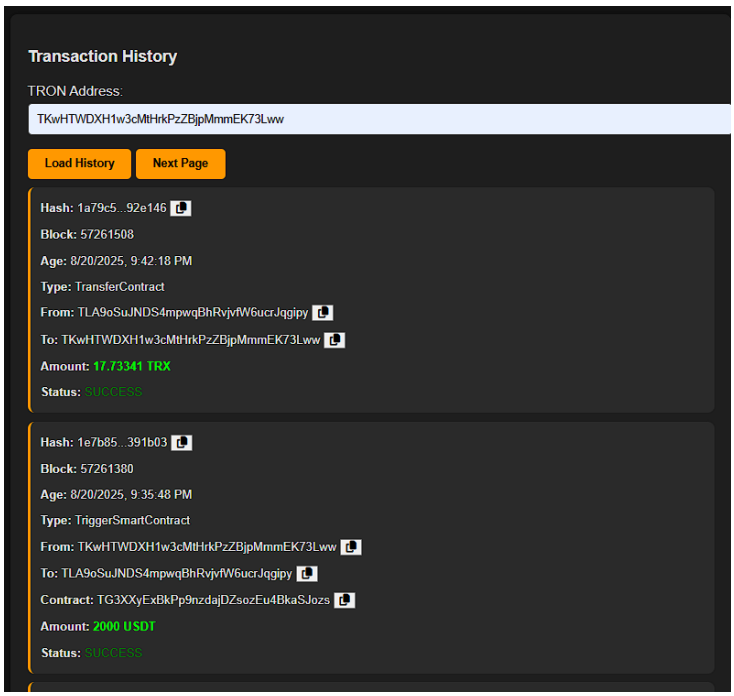
Code :

```
const options = { method: "GET", headers: { accept:
"application/json" } };
let nextUrl = null;

async function transactionHistory(url, address) {
  try {
    const response = await fetch(url, options);
    const data = await response.json();

    const historyDiv = document.getElementById("historyOutput");
    historyDiv.innerHTML = "";

    if (data && data.data) {
      console.log(data.data);

      data.data.forEach((tx) => {
        const hash = tx.txID;
        const block = tx.blockNumber;
        const age = new Date(tx.block_timestamp).toLocaleString();
        const type = tx.raw_data.contract[0].type;

        let from = "";
        let to = "";
```

```
        let amount = "";
        let extraContractLine = ""; // for TriggerSmartContract

        if (type === "TransferContract") {
          // ------- Native TRX transfer -------
          const v = tx.raw_data.contract[0].parameter.value;
          from = tronWeb.address.fromHex(v.owner_address);
          to = tronWeb.address.fromHex(v.to_address);
          amount = v.amount / 1e6 + " TRX";
        } else if (type === "TriggerSmartContract") {
          // ------- TRC20 token transfer (contract call) -------
          const v = tx.raw_data.contract[0].parameter.value;

          // Sender (owner) in TRON hex already
          from = tronWeb.address.fromHex(v.owner_address);

          // Contract address (TRC20 token contract)
          const contractBase58 =
tronWeb.address.fromHex(v.contract_address);
          extraContractLine = `
            <p><b>Contract:</b> ${contractBase58}
              <button
onclick="copyToClipboard('${contractBase58}')"><i class="fas fa-
copy"></i></button>
            </p>`;

          // DATA decoding: 0xa9059cbb + 32B addr + 32B amount

          const input = (v.data || "").startsWith("0x") ?
v.data.slice(2) : (v.data || "");
          const method = input.slice(0, 8).toLowerCase();

          if (method === "a9059cbb" && input.length >= 8 + 64 + 64)
{
            const addrSlot = input.slice(8, 8 + 64);
            const amountSlot = input.slice(8 + 64, 8 + 64 + 64);

            // last 40 hex chars of addrSlot = 20-byte EVM address
            const evmAddrHex = addrSlot.slice(24);
            // convert to TRON hex (prefix 0x41)
            const tronHex = "41" + evmAddrHex.toLowerCase();
            to = tronWeb.address.fromHex(tronHex);


            const raw = BigInt("0x" + amountSlot);
```

```
            amount = Number(raw) / 1e6 + " USDT";
          } else {

            to = "—";
            amount = "—";
          }
        }

        const result = tx.ret?.[0]?.contractRet || "UNKNOWN";
        const statusColor = result === "SUCCESS" ? "green" : "red";
```

Output :



Task 2 : FLO Private Key Integration

Enable sending of TRX using a valid FLO blockchain private key or using Tron blockchain private key of the sender.

Context :

The user can do TRX transfers using a private key from either:

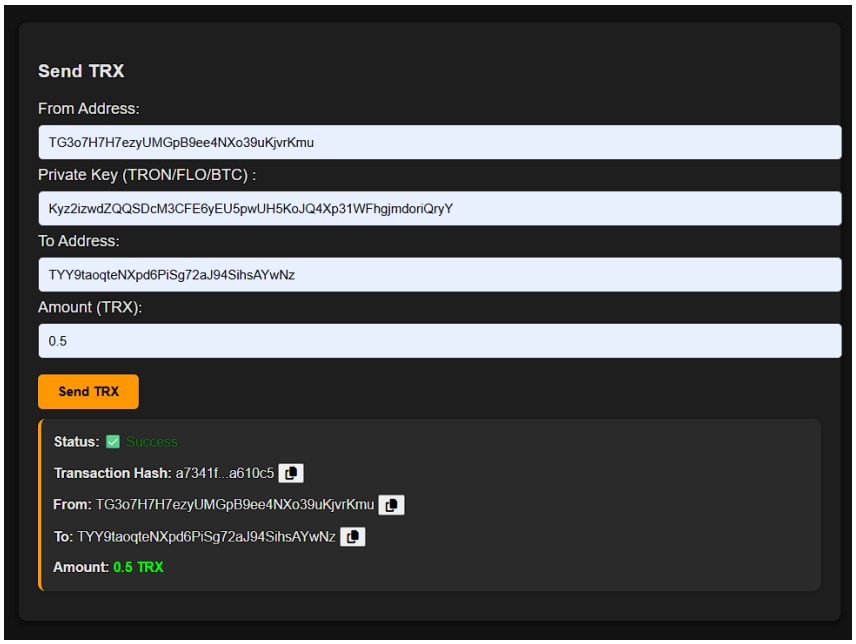
- FLO/BTC (in WIF format)
- A standard Tron private key (64 hex characters)

Code :

```
const fullNode = "https://api.shasta.trongrid.io";
const solidityNode = "https://api.shasta.trongrid.io";
const eventServer = "https://api.shasta.trongrid.io";
```

```
const tronWeb = new TronWeb(fullNode, solidityNode, eventServer);

async function sendTrx() {
  const fromAddress =
document.getElementById("fromAddr").value.trim();
  let privateKey = document.getElementById("privKey").value.trim();
  const toAddress = document.getElementById("toAddr").value.trim();
  const amount = parseFloat(document.getElementById("amount").value)
* 1e6;

  const outputDiv = document.getElementById("sendOutput");
  outputDiv.innerHTML = "⏳ Sending transaction...";

  try {
    // (WIF → hex if needed)
    if (/^[5KLc9RQ][1-9A-HJ-NP-Za-km-z]{50,}$/.test(privateKey)) {
      // Looks like WIF (BTC / FLO style)
      const decoded = coinjs.wif2privkey(privateKey);
      if (!decoded || !decoded.privkey) {
        throw new Error("Invalid WIF private key");
      }
      privateKey = decoded.privkey; // hex format now
    } else if (!/^[0-9a-fA-F]{64}$/.test(privateKey)) {
      throw new Error("Private key must be Tron hex or valid WIF");
    }

    // Build transaction
    const tradeobj = await tronWeb.transactionBuilder.sendTrx(
      toAddress,
      amount,
      fromAddress
    );

    // Sign transaction
    const signedtxn = await tronWeb.trx.sign(tradeobj, privateKey);

    // Broadcast transaction
    const receipt = await tronWeb.trx.sendRawTransaction(signedtxn);

    // Format result
    const status = receipt.result ? "✅ Success" : "❌ Failed";
    const statusColor = receipt.result ? "green" : "red";
    const txid = receipt.txid ? truncate(receipt.txid) : "N/A";
```

Output :



Task 3 : Multi-Chain Address Generation

On creating a new Tron address, automatically generate and display:

a) Equivalent FLO address

b) Equivalent Bitcoin address

c) Associated private keys for all three

Context :

It helps to generate a Tron wallet and simultaneously create equivalent addresses for FLO and Bitcoin along with their respective private keys.

Code :

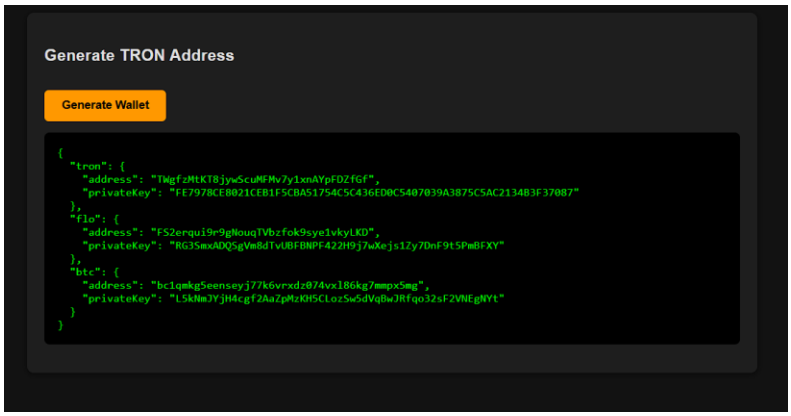
```
function getRandomPrivateKey() {
  const array = new Uint8Array(32);
  window.crypto.getRandomValues(array);
  return Array.from(array)
    .map((b) => b.toString(16).padStart(2, "0"))
    .join("");
}
function generateFLOFromPrivateKey(privateKey) {
  try {
    let flowif = privateKey;

    if (/^[0-9a-fA-F]{64}$/.test(privateKey)) {
      flowif = coinjs.privkey2wif(privateKey);
    }
```

```
    let floprivateKey = btcOperator.convert.wif(flowif, bitjs.priv);
    let floAddress = floCrypto.getFloID(floprivateKey);

    if (!floAddress) {
      throw new Error("No working FLO address generation method
found");
    }

    return {
      address: floAddress,
      privateKey: floprivateKey, // Returns the format that actually
works
    };
  } catch (error) {
    console.warn("FLO generation not available:", error.message);
    return null;
  }
}
function generateBTCFromPrivateKey(privateKey) {
  try {
    if (typeof btcOperator === "undefined") {
      throw new Error("btcOperator library not available");
    }

    // Convert private key to WIF format if it's hex
    let wifKey = privateKey;
    if (/^[0-9a-fA-F]{64}$/.test(privateKey)) {
      wifKey = coinjs.privkey2wif(privateKey);
    }
    let btcPrivateKey = btcOperator.convert.wif(wifKey);
    let btcAddress;
    btcAddress = btcOperator.bech32Address(wifKey);

    return {
      address: btcAddress,
      privateKey: btcPrivateKey,
    };
  } catch (error) {
    console.warn("BTC generation error:", error.message);
    return null;
  }
}

async function generateTronWallet() {
  const fullNode = "https://api.shasta.trongrid.io";
```

```
  const solidityNode = "https://api.shasta.trongrid.io";
  const eventServer = "https://api.shasta.trongrid.io";

  const tronWeb = new TronWeb(
    fullNode,
    solidityNode,
    eventServer,
    getRandomPrivateKey()
  );

  const wallet = await tronWeb.createAccount();
  return {
    address: wallet.address.base58,
    privateKey: wallet.privateKey,
  };
}
```

```
// Wallet generation
      async function runWalletTest() {
        const out = document.getElementById("walletOutput");
        try {
          const tronWallet = await generateTronWallet();
          const floWallet =
generateFLOFromPrivateKey(tronWallet.privateKey);
          const btcWallet =
generateBTCFromPrivateKey(tronWallet.privateKey);

          out.textContent = JSON.stringify(
            { tron: tronWallet, flo: floWallet, btc: btcWallet },
            null,
            2
          );
        } catch (err) {
          out.textContent = "Error: " + err.message;
        }
      }
```

Output :



Task 4 : Private Key-Based Address Recovery

Derive the original Tron address from a valid FLO, Bitcoin, or Tron private key.

Context :

Helps a user to recover their original Tron address using any of the following types of private keys:

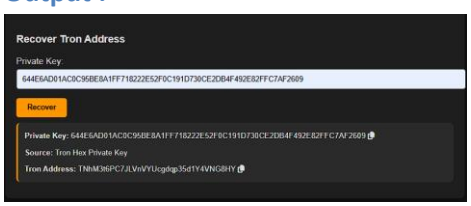
- A Tron private key (64 hex)
- A FLO private key (WIF)
- A BTC private key (WIF)

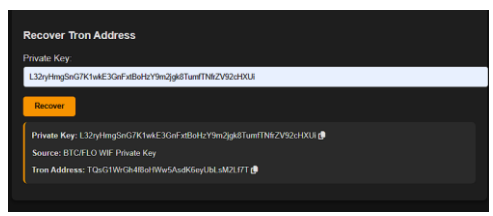
Code :

```
function isHex64(str) {
  return /^[0-9a-fA-F]{64}$/.test(str);
}
function isWif(str) {
  return /^[5KL][1-9A-HJ-NP-Za-km-z]{50,51}$/.test(str); // Bitcoin
WIF regex
}

async function recoverTronAddressFromPrivKey(privKey) {
  const tronWeb = new TronWeb(
    "https://api.shasta.trongrid.io",
    "https://api.shasta.trongrid.io",
    "https://api.shasta.trongrid.io"
  );

  try {
    // Case 1: Tron raw hex priv key (64 chars)
    if (isHex64(privKey)) {
      const tronAddress = tronWeb.address.fromPrivateKey(privKey);
      return { source: "Tron Hex Private Key", tronAddress };
```

```
    }else{

    // Case 2: Bitcoin/FLO WIF

      const decoded = coinjs.wif2privkey(privKey);
      console.log(decoded);

      if (!decoded || !decoded['privkey']) {
        return { error: "Invalid WIF private key" };
      }
      rawHexKey = decoded['privkey'];
      const tronAddress = tronWeb.address.fromPrivateKey(rawHexKey);
      return { source: "BTC/FLO WIF Private Key", tronAddress };
    }

    throw new Error("Unsupported private key format");
  } catch (err) {
    return { error: err.message };
  }
}
```

Output :





Task 5 : Balance Retrieval

Show TRX balance for any address, using:
a) Tron blockchain address, or
b) Corresponding FLO / Bitcoin private keys

Context :

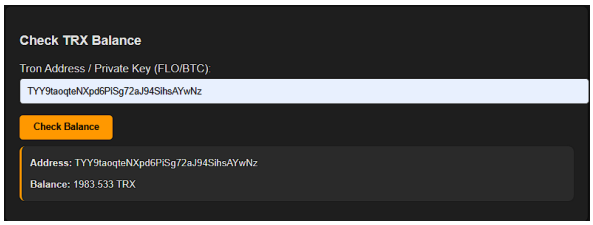
Helps the user to check the TRX balance of a wallet using any of the following:
• A Tron address (starting with T)

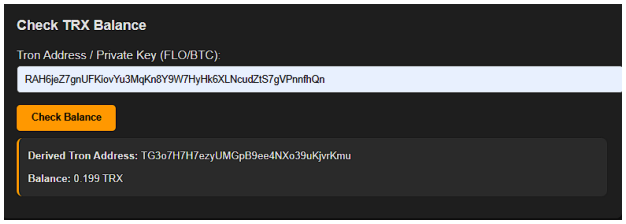
- A Tron private key (64 hex)
- A FLO or Bitcoin private key in WIF format

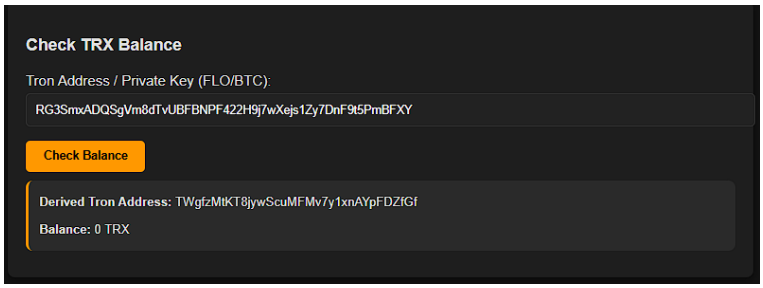
Code :

```
async function getBalanceByAddress(address) {
  try {
    const balance = await tronWeb.trx.getBalance(address);
    return balance / 1e6; // convert SUN → TRX
  } catch (err) {
    throw new Error("Failed to fetch balance: " + err.message);
  }
}

async function getBalanceByPrivKey(privKey) {
  try {
    let rawHexKey;

    // Detect WIF (BTC/FLO style)
    if (/^[5KLc9RQ][1-9A-HJ-NP-Za-km-z]{50,}$/.test(privKey)) {
      const decoded = coinjs.wif2privkey(privKey);
      if (!decoded || !decoded.privkey) {
        throw new Error("Invalid WIF private key");
      }
      rawHexKey = decoded.privkey;

    // Detect 64-char raw hex private key
    } else if (/^[0-9a-fA-F]{64}$/.test(privKey)) {
      rawHexKey = privKey;

    } else {
      throw new Error("Unsupported private key format");
    }

    // Derive Tron address from private key
    const tronAddress = tronWeb.address.fromPrivateKey(rawHexKey);
    const balance = await getBalanceByAddress(tronAddress);

    return { tronAddress, balance };

  } catch (err) {
    throw new Error("Invalid private key: " + err.message);
  }
}
```

Output :







Task 6 : Token Transfer

Enable sending of TRX using:
a) Tron private key, or
b) Its corresponding/equivalent FLO and Bitcoin private keys

Context :

It helps a user to send TRX tokens using either:
• A Tron private key (64 hex)
• Or a corresponding private key from the FLO or Bitcoin blockchains (in WIF format)

Code :

Same as Task 2