Assignment Number: 01/12

Q. No. Question Expected Time to complete1

Objective:

To understand blockchain interaction by creating and using a cryptocurrency wallet and interacting with a blockchain network using Python and Web3 to check wallet balance and perform a basic transaction simulation.

Requirements:

• Install Python 3.x

• Set up VS Code with Python extension

• Install required Python libraries:

• pip install web3

• Use a test blockchain network (Ethereum Sepolia /

Ganache local blockchain)

• Basic understanding of blockchain wallets and private keys

Practical Description:

Step 1: Environment Setup

• Install Python and VS Code

• Install Web3.py library

• Create a Python file named wallet_interaction.py

Step 2: Wallet and Blockchain Interaction Script

Create a Python script that:

• Connects to a blockchain network

• Loads a wallet using a private key

• Fetches wallet address

• Checks wallet balance

• Demonstrates transaction preparation (without real funds)

Solution Code:

```
from web3 import Web3

import tkinter as tk
from tkinter import messagebox
import os

# Step 1: Connect to the blockchain network
# Using Ethereum Sepolia testnet with a public RPC endpoint
sepolia_url = 'https://ethereum-sepolia-rpc.publicnode.com'
web3 = Web3(Web3.HTTPProvider(sepolia_url))

class WalletApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Blockchain Wallet Interaction")
        self.root.geometry("600x500")

        # Check connection
        if not web3.is_connected():
            messagebox.showerror("Connection Error", "Failed to connect to Sepolia testnet")
            root.quit()
            return

        # Wallet variables
        self.account = None
        self.private_key = None
        self.address = None

        # GUI Elements
        self.create_widgets()

    def create_widgets(self):
        # Title
        title_label = tk.Label(self.root, text="Ethereum Sepolia Wallet", font=("Arial", 16,
"bold"))
        title_label.pack(pady=10)

        # Generate Wallet Button
        self.generate_btn = tk.Button(self.root, text="Generate New Wallet",
command=self.generate_wallet, bg="lightblue")
        self.generate_btn.pack(pady=5)

        # Wallet Info Frame
        self.info_frame = tk.Frame(self.root)
        self.info_frame.pack(pady=10)

        self.address_label = tk.Label(self.info_frame, text="Address: Not generated",
font=("Arial", 10))
        self.address_label.pack()

        self.balance_label = tk.Label(self.info_frame, text="Balance: N/A", font=("Arial",
10))
        self.balance_label.pack()
```

```
        # Check Balance Button
        self.balance_btn = tk.Button(self.root, text="Check Balance",
command=self.check_balance, state=tk.DISABLED)
        self.balance_btn.pack(pady=5)

        # Transaction Frame
        trans_frame = tk.Frame(self.root)
        trans_frame.pack(pady=10)

        tk.Label(trans_frame, text="Recipient Address:").grid(row=0, column=0, sticky="w")
        self.recipient_entry = tk.Entry(trans_frame, width=50)
        self.recipient_entry.grid(row=0, column=1)
        self.recipient_entry.insert(0, "0x742d35Cc6634C0532925a3b844Bc454e4438f44e")  #
Example

        tk.Label(trans_frame, text="Amount (ETH):").grid(row=1, column=0, sticky="w")
        self.amount_entry = tk.Entry(trans_frame, width=20)
        self.amount_entry.grid(row=1, column=1, sticky="w")
        self.amount_entry.insert(0, "0.01")

        # Prepare Transaction Button
        self.prepare_btn = tk.Button(self.root, text="Prepare Transaction",
command=self.prepare_transaction, state=tk.DISABLED)
        self.prepare_btn.pack(pady=5)

        # Transaction Info
        self.trans_info = tk.Text(self.root, height=10, width=70, state=tk.DISABLED)
        self.trans_info.pack(pady=10)

    def generate_wallet(self):
        # Generate new account
        self.account = web3.eth.account.create()
        self.private_key = self.account.key.hex()
        self.address = self.account.address

        # Update GUI
        self.address_label.config(text=f"Address: {self.address}")
        self.balance_label.config(text="Balance: Checking...")
        self.balance_btn.config(state=tk.NORMAL)
        self.prepare_btn.config(state=tk.NORMAL)

        # Show private key in message box (for demo only - in real app, don't show!)
        messagebox.showinfo("Private Key", f"Keep this secret!\n{self.private_key}")

        # Auto check balance
        self.check_balance()

    def check_balance(self):
        if not self.address:
            return

        try:
            balance_wei = web3.eth.get_balance(self.address)
```

```
            balance_eth = web3.from_wei(balance_wei, 'ether')
            self.balance_label.config(text=f"Balance: {balance_eth} ETH")
        except Exception as e:
            messagebox.showerror("Error", f"Failed to check balance: {e}")

    def prepare_transaction(self):
        if not self.address:
            messagebox.showwarning("Warning", "Generate a wallet first")
            return

        try:
            recipient = self.recipient_entry.get().strip()
            amount_str = self.amount_entry.get().strip()

            if not recipient or not amount_str:
                messagebox.showwarning("Warning", "Enter recipient address and amount")
                return

            amount_wei = web3.to_wei(float(amount_str), 'ether')

            # Get gas price
            gas_price = web3.eth.gas_price

            # Estimate gas
            try:
                gas_estimate = web3.eth.estimate_gas({
                    'to': recipient,
                    'from': self.address,
                    'value': amount_wei
                })
            except Exception as e:
                gas_estimate = 21000  # Fallback

            # Prepare transaction
            transaction = {
                'to': recipient,
                'value': amount_wei,
                'gas': gas_estimate,
                'gasPrice': gas_price,
                'nonce': web3.eth.get_transaction_count(self.address),
                'chainId': 11155111
            }

            # Sign transaction
            signed_txn = web3.eth.account.sign_transaction(transaction, self.private_key)

            # Display info
            info = f"Prepared Transaction:\n"
            info += f"To: {transaction['to']}\n"
            info += f"Value: {web3.from_wei(transaction['value'], 'ether')} ETH\n"
            info += f"Gas: {transaction['gas']}\n"
            info += f"Gas Price: {web3.from_wei(transaction['gasPrice'], 'gwei')} Gwei\n"
            info += f"Nonce: {transaction['nonce']}\n"
            info += f"Chain ID: {transaction['chainId']}\n"
```

```
            info += f"Signed Hash: {signed_txn.hash.hex()}\n\n"
            info += "Note: Transaction not sent (no funds in test account)"

            self.trans_info.config(state=tk.NORMAL)
            self.trans_info.delete(1.0, tk.END)
            self.trans_info.insert(tk.END, info)
            self.trans_info.config(state=tk.DISABLED)

        except Exception as e:
            messagebox.showerror("Error", f"Failed to prepare transaction: {e}")

# Main
if __name__ == "__main__":
    root = tk.Tk()
    app = WalletApp(root)
    root.mainloop()
```

output: