from web3 import Web3

import json

# Replace these values

INFURA\_URL = "https://goerli.infura.io/v3/YOUR\_INFURA\_PROJECT\_ID"

CONTRACT\_ADDRESS = "0xYourDeployedContractAddress" # Contract address after deployment

PRIVATE\_KEY = "0xYourPrivateKey" # Private key of your Ethereum account (owner or buyer)

ACCOUNT\_ADDRESS = "0xYourAccountAddress" # Your wallet address (must match private key)

# Load contract ABI (paste ABI JSON here or load from file)

ABI = json.loads('''[

{

"inputs": [

{ "internalType": "string", "name": "name", "type": "string" },

{ "internalType": "string", "name": "symbol", "type": "string" },

{ "internalType": "uint256", "name": "\_ticketPrice", "type": "uint256" },

{ "internalType": "uint256", "name": "\_maxTickets", "type": "uint256" },

{ "internalType": "string", "name": "\_baseTokenURI", "type": "string" }

],

"stateMutability": "nonpayable",

"type": "constructor"

},

{

"inputs": [],

"name": "buyTicket",

"outputs": [],

"stateMutability": "payable",

"type": "function"

},

{

"inputs": [{ "internalType": "uint256", "name": "ticketId", "type": "uint256" }],

"name": "isTicketUsed",

"outputs": [{ "internalType": "bool", "name": "", "type": "bool" }],

"stateMutability": "view",

"type": "function"

},

{

"inputs": [{ "internalType": "uint256", "name": "ticketId", "type": "uint256" }],

"name": "verifyTicket",

"outputs": [],

"stateMutability": "nonpayable",

"type": "function"

}

]''')

# Connect to Web3 provider

w3 = Web3(Web3.HTTPProvider(INFURA\_URL))

# Check connection

if not w3.isConnected():

print("Failed to connect to Ethereum node.")

exit()

contract = w3.eth.contract(address=Web3.to\_checksum\_address(CONTRACT\_ADDRESS), abi=ABI)

def buy\_ticket(ticket\_price\_wei):

nonce = w3.eth.get\_transaction\_count(ACCOUNT\_ADDRESS)

txn = contract.functions.buyTicket().build\_transaction({

'from': ACCOUNT\_ADDRESS,

'value': ticket\_price\_wei,

'nonce': nonce,

'gas': 300000,

'gasPrice': w3.to\_wei('10', 'gwei')

})

signed\_txn = w3.eth.account.sign\_transaction(txn, private\_key=PRIVATE\_KEY)

tx\_hash = w3.eth.send\_raw\_transaction(signed\_txn.rawTransaction)

print(f"Buying ticket... tx hash: {tx\_hash.hex()}")

receipt = w3.eth.wait\_for\_transaction\_receipt(tx\_hash)

print(f"Transaction mined in block {receipt.blockNumber}")

def verify\_ticket(ticket\_id):

nonce = w3.eth.get\_transaction\_count(ACCOUNT\_ADDRESS)

txn = contract.functions.verifyTicket(ticket\_id).build\_transaction({

'from': ACCOUNT\_ADDRESS,

'nonce': nonce,

'gas': 200000,

'gasPrice': w3.to\_wei('10', 'gwei')

})

signed\_txn = w3.eth.account.sign\_transaction(txn, private\_key=PRIVATE\_KEY)

tx\_hash = w3.eth.send\_raw\_transaction(signed\_txn.rawTransaction)

print(f"Verifying ticket {ticket\_id}... tx hash: {tx\_hash.hex()}")

receipt = w3.eth.wait\_for\_transaction\_receipt(tx\_hash)

print(f"Transaction mined in block {receipt.blockNumber}")

def check\_ticket\_used(ticket\_id):

used = contract.functions.isTicketUsed(ticket\_id).call()

print(f"Ticket {ticket\_id} used? {'Yes' if used else 'No'}")

if \_\_name\_\_ == "\_\_main\_\_":

print("Choose an action:")

print("1 - Buy ticket")

print("2 - Verify ticket (owner only)")

print("3 - Check ticket status")

choice = input("Enter choice: ")

if choice == "1":

# Set ticket price in wei (must match contract ticketPrice)

ticket\_price\_wei = w3.to\_wei(0.05, 'ether') # example 0.05 ETH

buy\_ticket(ticket\_price\_wei)

elif choice == "2":

ticket\_id = int(input("Enter ticket ID to verify: "))

verify\_ticket(ticket\_id)

elif choice == "3":

ticket\_id = int(input("Enter ticket ID to check: "))

check\_ticket\_used(ticket\_id)

else:

print("Invalid choice.")