Listing Homes on the Blockchain

By Hunter Spence, Andre Montgomery,
Nicholas Pamboukis

Introduction

Project

- Design an application that can be used to buy and sell homes
    - Using python & blockchain technology
- Able to view and create listings
- Execute transactions on the blockchain

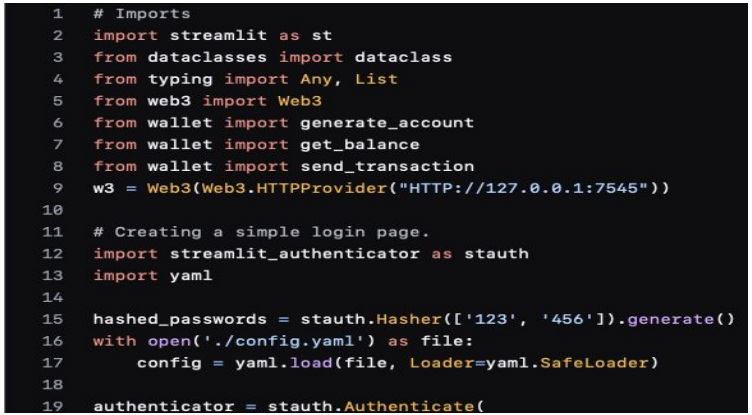
Mission

- Our goal is to provide a proof of concept demonstration on a payment solution for buying and selling houses using the blockchain. With our UI, users can view details of a house, then use their crypto wallet to transfer funds to the sellers address. In this solution, we use ETH.

Project Files

Three Main Files

- housing_streamlit.py
- wallet.py
- config.yaml



```python
# Imports
import streamlit as st
from dataclasses import dataclass
from typing import Any, List
from web3 import Web3
from wallet import generate_account
from wallet import get_balance
from wallet import send_transaction
w3 = Web3(Web3.HTTPProvider("HTTP://127.0.0.1:7545"))

# Creating a simple login page.
import streamlit_authenticator as stauth
import yaml

hashed_passwords = stauth.Hasher(['123', '456']).generate()
with open('./config.yaml') as file:
    config = yaml.load(file, Loader=yaml.SafeLoader)

authenticator = stauth.Authenticate(
```

housing_streamlit.py

- Footnotes:
  - Build a dApp
  - Created First during Development Phase
  - Useful for Ganache and back-testing preceding files

Goals for housing_streamlit.py

housing_streamlit.py

- Successfully create a decentralized application (dApp) using Streamlit …
  - Purchase & Sell Homes
  - Images of listed homes
    - Displays the contract address and amount for each home
  - Efficient & Secure Payments w/ a validated transaction hash
  - Authenticate users' login credentials
- Important Files:
  - config.yaml: Storage for usernames and passwords for all accounts on dApps platform
  - crypto_wallet.py: Generate new accounts, authenticate purchases, and provide an easy-to-use user interface (UI)

Partial Code for housing__streamlit.py

```python
# Imports
import streamlit as st
from dataclasses import dataclass
from typing import Any, List
from web3 import Web3
from crypto_wallet import generate_account
from crypto_wallet import get_balance
from crypto_wallet import send_transaction
w3 = Web3(Web3.HTTPProvider("HTTP://127.0.0.1:7545"))

# Creating a simple login page.
import streamlit_authenticator as stauth
import yaml

hashed_passwords = stauth.Hasher(['123', '456']).generate()
with open('./config.yaml') as file:
    config = yaml.load(file, Loader=yaml.SafeLoader)

authenticator = stauth.Authenticate(
    config['credentials'],
    config['cookie']['name'],
    config['cookie']['key'],
    config['cookie']['expiry_days'],
    config['preauthorized']
)

name, authentication_status, username = authenticator.login('Login', 'main')

if authentication_status:
    authenticator.logout('Logout', 'main')
    st.write('Welcome')
    st.title('Some content')
elif authentication_status == False:
    st.error('Username/password is incorrect')
elif authentication_status == None:
    st.warning('Please enter username and password')

# Creat background photo
import base64
```

```python

authenticator = stauth.Authenticate(
    config['credentials'],
    config['cookie']['name'],
    config['cookie']['key'],
    config['cookie']['expiry_days'],
    config['preauthorized']
)

name, authentication_status, username = authenticator.login('Login', 'main')

if authentication_status:
    authenticator.logout('Logout', 'main')
    st.write('Welcome')
    st.title('Some content')
elif authentication_status == False:
    st.error('Username/password is incorrect')
elif authentication_status == None:
    st.warning('Please enter username and password')

# Creat background photo
import base64
def add_bg_from_local(image_file):
    with open(image_file, "rb") as image_file:
        encoded_string = base64.b64encode(image_file.read())
    st.markdown(
    f"""
    <style>
    .stApp {{
        background-image: url({encoded_string.decode()});
        background-size: cover
    }}
    </style>
    """,
    unsafe_allow_html=True
    )
add_bg_from_local('background2.jpeg')
```

config.yaml

yaml.config

- Simple enough… storage for a user's super secret password to access the dApp !!

```yaml
credentials:
  usernames:
    amont:
      email: PabloJordan@appleisbetter.com
      name: Pablo Jordan
      password: '123' # To be replaced with hashed password
    hspence:
      email: SeanMyers@shouldAlsoGetanApple.com
      name: Sean Myers
      password: '456' # To be replaced with hashed password
cookie:
  expiry_days: 0
  key: some_signature_key
  name: some_cookie_name
preauthorized:
  emails:
  - nick@gmail.com

```

Goals for wallet.py

crypto_wallet.py

MNEMONIC ?

`mansion minute runway cause vehicle power gun clay error unable fetch proof`

- Generate Account:
  - Loads user's mnemonic phrase
  - Mnemonic phrase is unique to the user's crypto wallet authentication keys
- Gives functionality to User Interface:
  - get_balance: Allows user to view balance of address
  - send_transaction:
    - Automatically estimate and set gas price strategies for all executed transactions
    - Converts eth into wei: required for gas fees
    - Sends a signed transaction for user confirmation

Code for wallet.py

```python
# Imports
import os
import requests
from dotenv import load_dotenv

load_dotenv()
from bip44 import Wallet
from web3 import Account
from web3 import middleware
from web3.gas_strategies.time_based import medium_gas_price_strategy

#############################################################################

def generate_account():

    # Fetch mnemonic
    mnemonic = "job evil town extra ten parent describe leg pride enhance grow hat"

    # Create Wallet Object
    wallet = Wallet(mnemonic)

    # Derive Ethereum Private Key
    private, public = wallet.derive_account("eth")

    # Convert private key into an Ethereum account
    account = Account.privateKeyToAccount(private)

    return account


def get_balance(w3, address):

    # Get balance of address in Wei
    wei_balance = w3.eth.get_balance(address)

    # Convert Wei value to ether
    ether = w3.fromWei(wei_balance, "ether")
```

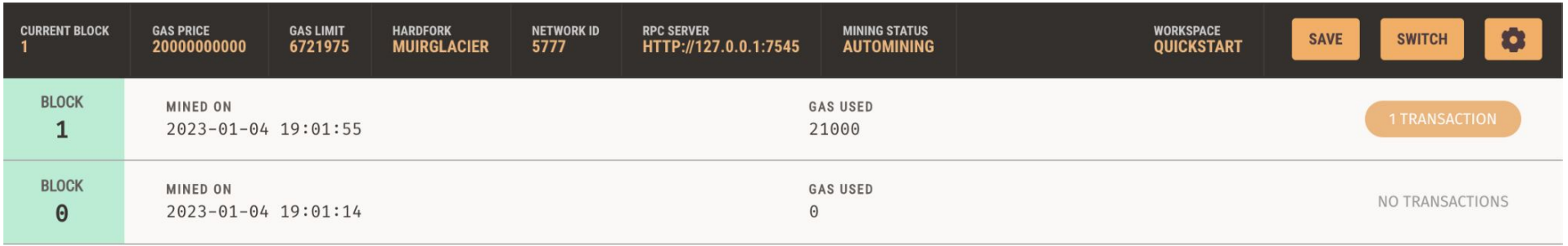
```python
    wei_balance = w3.eth.get_balance(address)

    # Convert Wei value to ether
    ether = w3.fromWei(wei_balance, "ether")

    # Return the value in ether
    return ether


def send_transaction(w3, account, to, wage):

    # Set gas price strategy
    w3.eth.setGasPriceStrategy(medium_gas_price_strategy)

    # Convert eth amount to Wei
    value = w3.toWei(wage, "ether")

    # Calculate gas estimate
    gasEstimate = w3.eth.estimateGas(
        {"to": to, "from": account.address, "value": value}
    )

    # Construct a raw transaction
    raw_tx = {
        "to": to,
        "from": account.address,
        "value": value,
        "gas": gasEstimate,
        "gasPrice": 0,
        "nonce": w3.eth.getTransactionCount(account.address),
    }

    # Sign the raw transaction with ethereum account
    signed_tx = account.signTransaction(raw_tx)

    # Send the signed transactions
    return w3.eth.sendRawTransaction(signed_tx.rawTransaction)
```

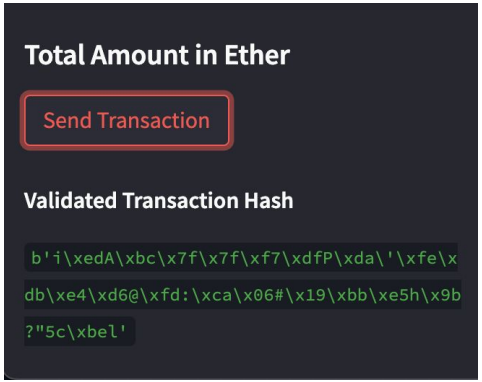
Testing Transactions

Backtesting & Validating Results

- Import functions into housing_stremlit.py (dApp)
  - Feed incorrect passwords and wallet addresses to accurately receive error messages
  - After, correctly enter a user's information for a successful send_transaction
- Validate - User can visibly identify:
  - Account has been successfully created
  - Account balance is accurate with user's current crypto wallet amount (eth)
  - Receive a signed transaction after a contract has been fulfilled
- Ganache: - The transaction can be viewed on the blockchain test network once completed

Validated Transaction ✓



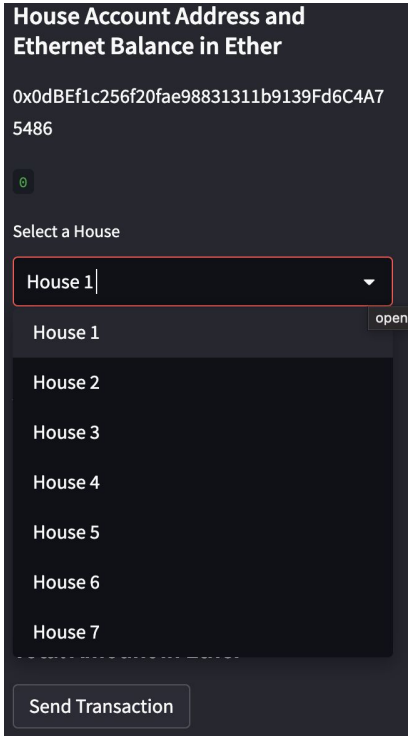


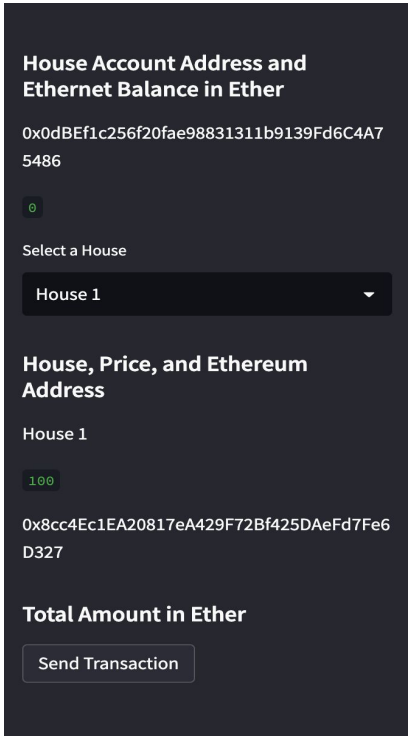
Potential for the Future

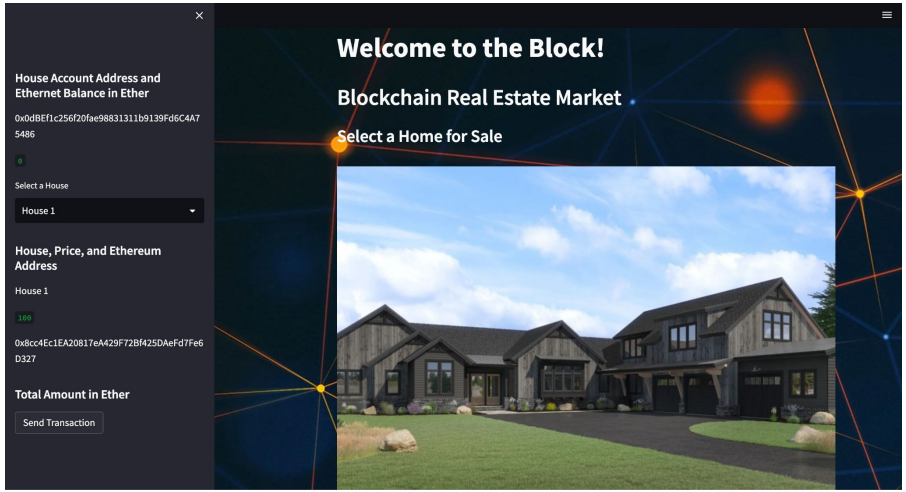
Next Steps

- Explore ways to create unique NFT art for individual listings
  - Possibility of Minting
- Add an option for the homeowner to list and receive private mortgage loans
  - Professions such as Lien Specialist/ Officer and Private Mortgage Lenders then could also utilize the dApp
- Change price to USD on streamlit, convert to ETH for transaction
- Host the website on a cloud server
- Add specific wallet configuration to streamlit
- Add separate page, allowing users to create a listing

Pictures of dApp







Questions?