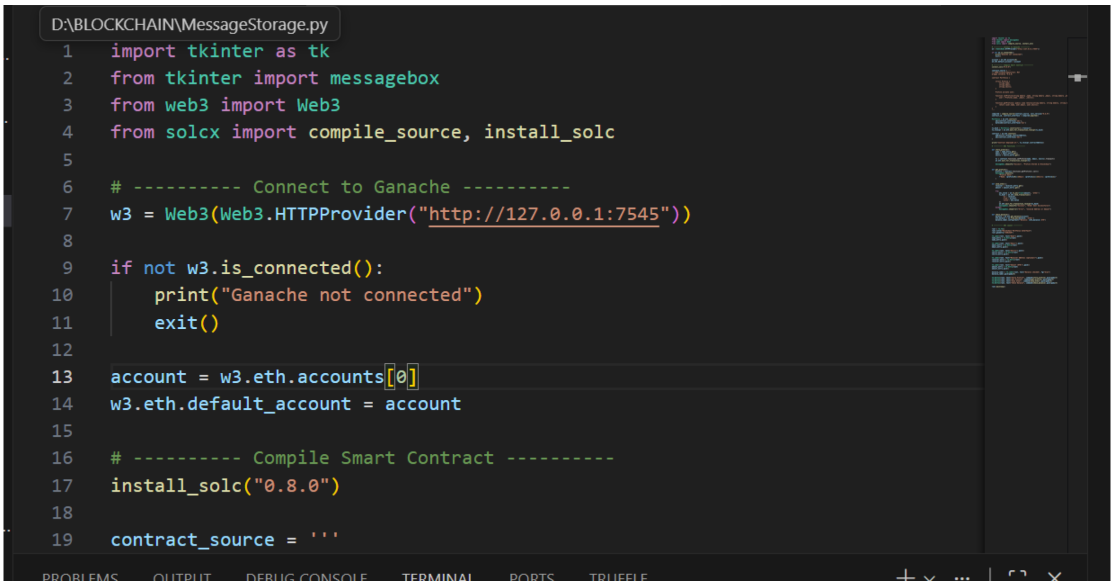
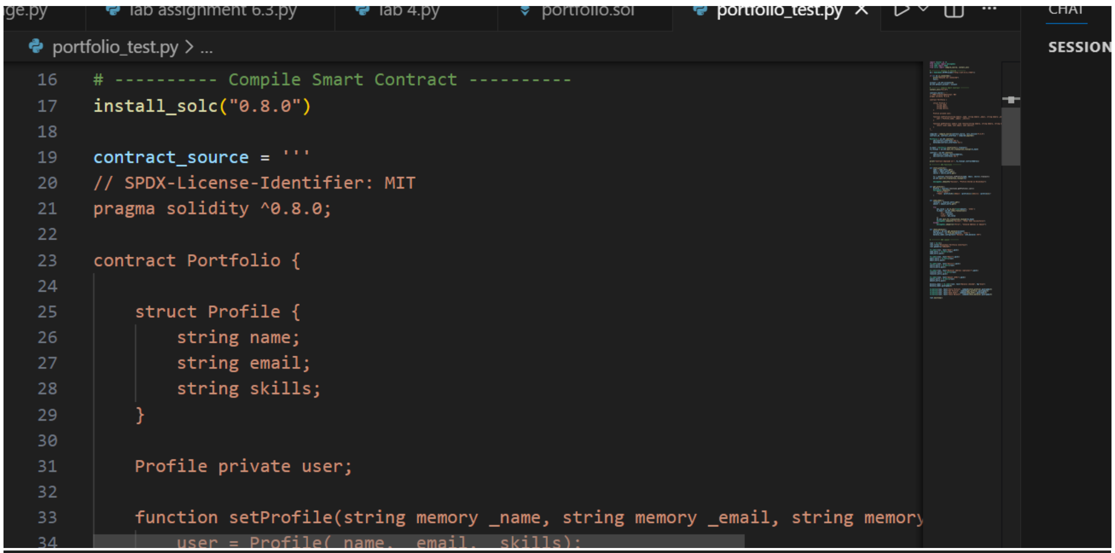
Assignment _005

2303A510H5
Batch_30

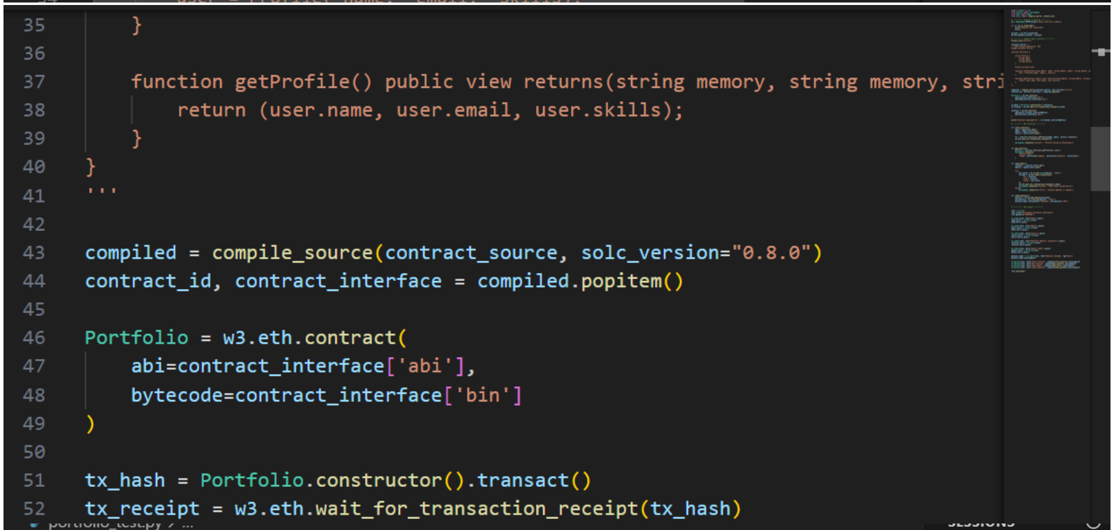
To develop and perform unit testing of smart contracts using the Truffle framework
Objective
Implement a simple Solidity Smart Contract and perform UNIT Testing using Truffle to verify contract functionality.
Problem Statement (Equivalent to Given Cipher Practical)
Develop a basic Personal Portfolio Smart Contract that stores and retrieves user profile data, and validate its behavior using Truffle unit tests.
Requirements
Development Environment
• Install Node.js
• Install Truffle Framework
• Install Ganache (Local Blockchain)
• Use VS Code with:
o Solidity Extension
o JavaScript Extension


D:\BLOCKCHAIN\MessageStorage.py

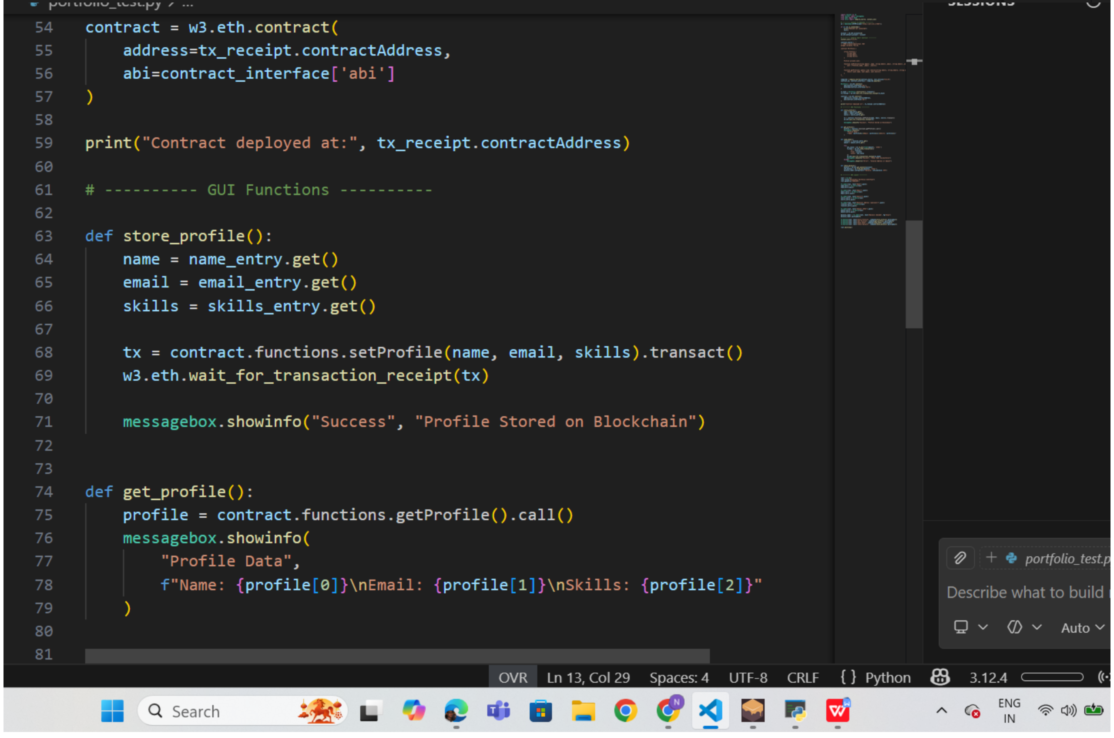
```python
import tkinter as tk
from tkinter import messagebox
from web3 import Web3
from solcx import compile_source, install_solc

# ---------- Connect to Ganache ----------
w3 = Web3(Web3.HTTPProvider("http://127.0.0.1:7545"))

if not w3.is_connected():
    print("Ganache not connected")
    exit()

account = w3.eth.accounts[0]
w3.eth.default_account = account

# ---------- Compile Smart Contract ----------
install_solc("0.8.0")

contract_source = '''
```



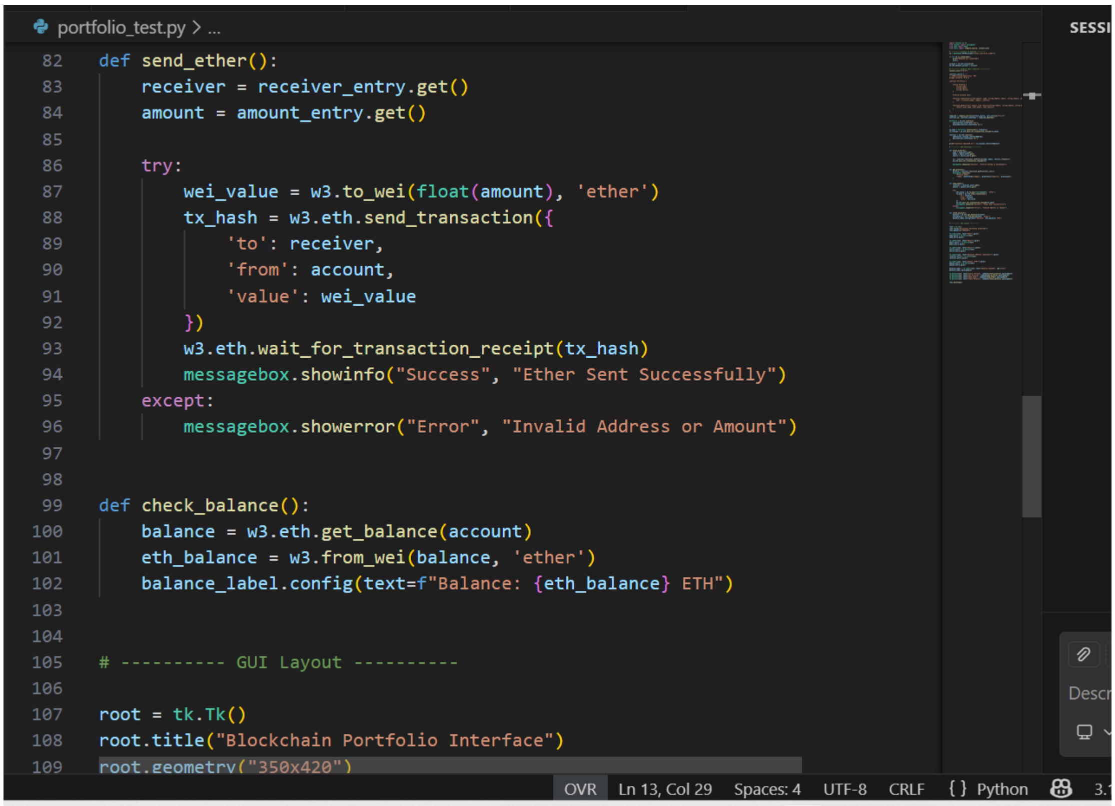
```
# ---------- Compile Smart Contract ----------
install_solc("0.8.0")

contract_source = '''
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Portfolio {

    struct Profile {
        string name;
        string email;
        string skills;
    }

    Profile private user;

    function setProfile(string memory _name, string memory _email, string memory
        user = Profile(_name, _email, _skills);
```



```
    }

    function getProfile() public view returns(string memory, string memory, stri
        return (user.name, user.email, user.skills);
    }
}
'''

compiled = compile_source(contract_source, solc_version="0.8.0")
contract_id, contract_interface = compiled.popitem()

Portfolio = w3.eth.contract(
    abi=contract_interface['abi'],
    bytecode=contract_interface['bin']
)

tx_hash = Portfolio.constructor().transact()
tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)
```



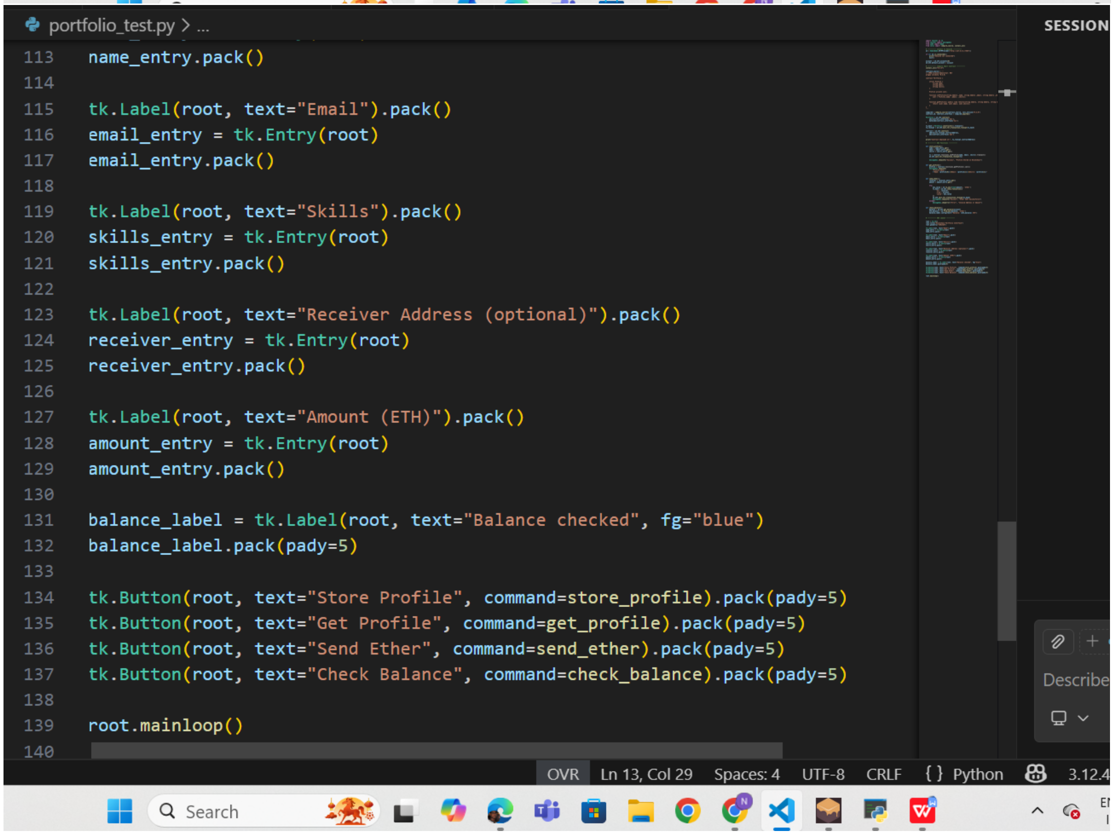
```
contract = w3.eth.contract(
    address=tx_receipt.contractAddress,
    abi=contract_interface['abi']
)

print("Contract deployed at:", tx_receipt.contractAddress)

# ---------- GUI Functions ----------

def store_profile():
    name = name_entry.get()
    email = email_entry.get()
    skills = skills_entry.get()

    tx = contract.functions.setProfile(name, email, skills).transact()
    w3.eth.wait_for_transaction_receipt(tx)

    messagebox.showinfo("Success", "Profile Stored on Blockchain")


def get_profile():
    profile = contract.functions.getProfile().call()
    messagebox.showinfo(
        "Profile Data",
        f"Name: {profile[0]}\nEmail: {profile[1]}\nSkills: {profile[2]}"
    )

```



portfolio_test.py > ...

```python
def send_ether():
    receiver = receiver_entry.get()
    amount = amount_entry.get()

    try:
        wei_value = w3.to_wei(float(amount), 'ether')
        tx_hash = w3.eth.send_transaction({
            'to': receiver,
            'from': account,
            'value': wei_value
        })
        w3.eth.wait_for_transaction_receipt(tx_hash)
        messagebox.showinfo("Success", "Ether Sent Successfully")
    except:
        messagebox.showerror("Error", "Invalid Address or Amount")


def check_balance():
    balance = w3.get_balance(account)
    eth_balance = w3.from_wei(balance, 'ether')
    balance_label.config(text=f"Balance: {eth_balance} ETH")


# ---------- GUI Layout ----------

root = tk.Tk()
root.title("Blockchain Portfolio Interface")
root.geometry("350x420")
```

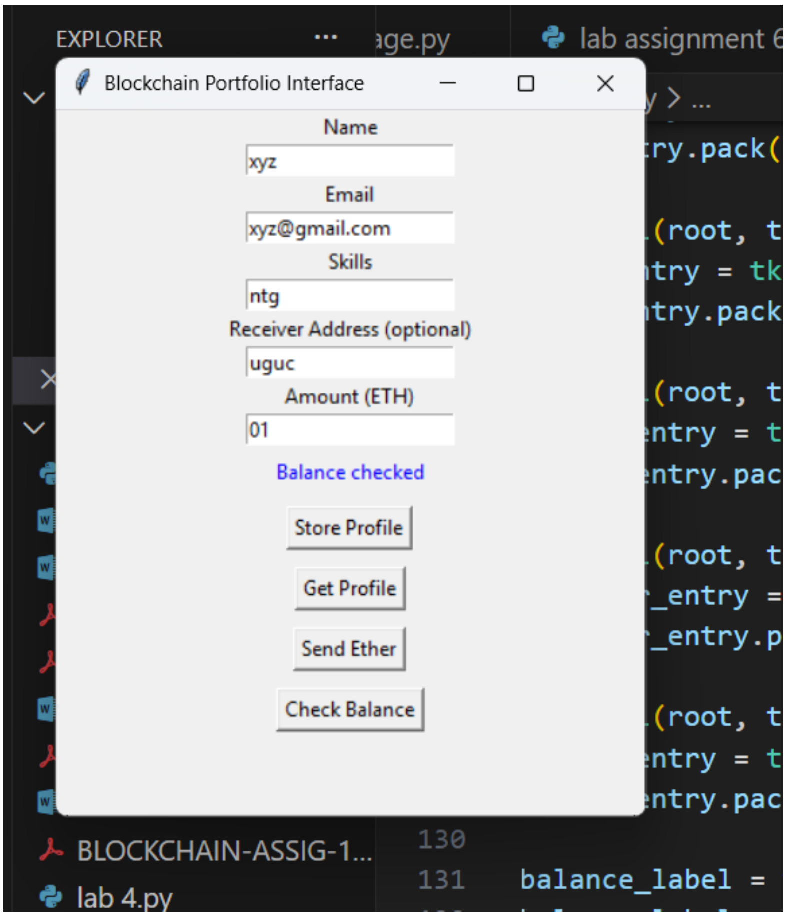
OVR Ln 13, Col 29 Spaces: 4 UTF-8 CRLF { } Python



portfolio_test.py > ...

```python
name_entry.pack()

tk.Label(root, text="Email").pack()
email_entry = tk.Entry(root)
email_entry.pack()

tk.Label(root, text="Skills").pack()
skills_entry = tk.Entry(root)
skills_entry.pack()

tk.Label(root, text="Receiver Address (optional)").pack()
receiver_entry = tk.Entry(root)
receiver_entry.pack()

tk.Label(root, text="Amount (ETH)").pack()
amount_entry = tk.Entry(root)
amount_entry.pack()

balance_label = tk.Label(root, text="Balance checked", fg="blue")
balance_label.pack(pady=5)

tk.Button(root, text="Store Profile", command=store_profile).pack(pady=5)
tk.Button(root, text="Get Profile", command=get_profile).pack(pady=5)
tk.Button(root, text="Send Ether", command=send_ether).pack(pady=5)
tk.Button(root, text="Check Balance", command=check_balance).pack(pady=5)

root.mainloop()
```

OVR Ln 13, Col 29 Spaces: 4 UTF-8 CRLF { } Python 3.12.4



Blockchain Portfolio Interface

Name: xyz

Email: xyz@gmail.com

Skills: ntg

Receiver Address (optional): uguc

Amount (ETH): 01

Balance checked

```
balance_label =
```