

BlockChain Engineering

Assignment-

06

Name : Manupati Eshwar

Ht.no: 2303A51495

Code:

```
import tkinter as tk
from tkinter import messagebox
from web3 import Web3
from web3.providers.eth_tester import EthereumTesterProvider
from solcx import compile_source, install_solc

# Install Solidity compiler
install_solc("0.8.20")

# Solidity Contract
contract_source_code = """
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract PersonalPortfolio {

    struct Portfolio {
        string name;
        string role;
        string description;
    }

    Portfolio private portfolio;

    event PortfolioUpdated(string name, string role, string description);

    constructor(string memory _name, string memory _role, string memory _description) {
        portfolio = Portfolio(_name, _role, _description);
    }

    function updatePortfolio(
        string memory _name,
        string memory _role,
        string memory _description
    ) public {
        portfolio = Portfolio(_name, _role, _description);
        emit PortfolioUpdated(_name, _role, _description);
    }

    function getPortfolio()
    public
```

```

view
returns (
    string memory,
    string memory,
    string memory
)
{
    return (portfolio.name, portfolio.role, portfolio.description);
}
}

# Compile contract
compiled_sol = compile_source(contract_source_code, solc_version="0.8.20")
contract_id, contract_interface = compiled_sol.popitem()

# Setup Web3
w3 = Web3(EthereumTesterProvider())
w3.eth.default_account = w3.eth.accounts[0]

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

tx_hash = Portfolio.constructor(
    "John Doe",
    "Blockchain Developer",
    "Building decentralized applications"
).transact()

tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)

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

# ===== GUI =====

root = tk.Tk()
root.title("Personal Portfolio DApp (Python GUI)")
root.geometry("500x400")

```

```

# Labels
tk.Label(root, text="Name").pack()
name_entry = tk.Entry(root, width=50)
name_entry.pack()

tk.Label(root, text="Role").pack()
role_entry = tk.Entry(root, width=50)
role_entry.pack()

tk.Label(root, text="Description").pack()
desc_entry = tk.Entry(root, width=50)
desc_entry.pack()

output_text = tk.Text(root, height=8, width=60)
output_text.pack(pady=10)

# Functions
def load_portfolio():
    name, role, description = contract.functions.getPortfolio().call()

    name_entry.delete(0, tk.END)
    role_entry.delete(0, tk.END)
    desc_entry.delete(0, tk.END)

    name_entry.insert(0, name)
    role_entry.insert(0, role)
    desc_entry.insert(0, description)

    output_text.insert(tk.END, "Loaded portfolio from blockchain\n")

def update_portfolio():
    name = name_entry.get()
    role = role_entry.get()
    description = desc_entry.get()

    try:
        tx = contract.functions.updatePortfolio(
            name, role, description
        ).transact()

        receipt = w3.eth.wait_for_transaction_receipt(tx)
    
```

```

        output_text.insert(
            tk.END,
            f"Portfolio updated!\nTx Hash: {receipt.transactionHash.hex()}\n\n"
        )

    messagebox.showinfo("Success", "Portfolio Updated Successfully!")

except Exception as e:
    messagebox.showerror("Error", str(e))

# Buttons
tk.Button(root, text="Load Portfolio", command=load_portfolio, bg="lightblue").pack(pady=5)
tk.Button(root, text="Update Portfolio", command=update_portfolio, bg="lightgreen").pack(pady=5)

# Load initial data
load_portfolio()

root.mainloop()

import tkinter as tk
from tkinter import messagebox
from web3 import Web3
from web3.providers.eth_tester import EthereumTesterProvider
from solcx import compile_source, install_solc

# Install Solidity compiler
install_solc("0.8.20")

# Solidity Contract
contract_source_code = """
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract PersonalPortfolio {

    struct Portfolio {
        string name;
        string role;
        string description;
    }

    Portfolio private portfolio;
}

```

```

event PortfolioUpdated(string name, string role, string description);

constructor(string memory _name, string memory _role, string memory _description) {
    portfolio = Portfolio(_name, _role, _description);
}

function updatePortfolio(
    string memory _name,
    string memory _role,
    string memory _description
) public {
    portfolio = Portfolio(_name, _role, _description);
    emit PortfolioUpdated(_name, _role, _description);
}

function getPortfolio()
public
view
returns (
    string memory,
    string memory,
    string memory
)
{
    return (portfolio.name, portfolio.role, portfolio.description);
}
}

"""

# Compile contract
compiled_sol = compile_source(contract_source_code, solc_version="0.8.20")
contract_id, contract_interface = compiled_sol.popitem()

# Setup Web3
w3 = Web3(EthereumTesterProvider())
w3.eth.default_account = w3.eth.accounts[0]

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

tx_hash = Portfolio.constructor(

```

```

    "John Doe",
    "Blockchain Developer",
    "Building decentralized applications"
).transact()

tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)

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

# ===== GUI =====

root = tk.Tk()
root.title("Personal Portfolio DApp (Python GUI)")
root.geometry("500x400")

# Labels
tk.Label(root, text="Name").pack()
name_entry = tk.Entry(root, width=50)
name_entry.pack()

tk.Label(root, text="Role").pack()
role_entry = tk.Entry(root, width=50)
role_entry.pack()

tk.Label(root, text="Description").pack()
desc_entry = tk.Entry(root, width=50)
desc_entry.pack()

output_text = tk.Text(root, height=8, width=60)
output_text.pack(pady=10)

# Functions
def load_portfolio():
    name, role, description = contract.functions.getPortfolio().call()

    name_entry.delete(0, tk.END)
    role_entry.delete(0, tk.END)
    desc_entry.delete(0, tk.END)

    name_entry.insert(0, name)
    role_entry.insert(0, role)

```

```

desc_entry.insert(0, description)

output_text.insert(tk.END, "Loaded portfolio from blockchain\n")

def update_portfolio():
    name = name_entry.get()
    role = role_entry.get()
    description = desc_entry.get()

    try:
        tx = contract.functions.updatePortfolio(
            name, role, description
        ).transact()

        receipt = w3.eth.wait_for_transaction_receipt(tx)

        output_text.insert(
            tk.END,
            f"Portfolio updated!\nTx Hash: {receipt.transactionHash.hex()}\n\n"
        )

        messagebox.showinfo("Success", "Portfolio Updated Successfully!")

    except Exception as e:
        messagebox.showerror("Error", str(e))

# Buttons
tk.Button(root, text="Load Portfolio", command=load_portfolio, bg="lightblue").pack(pady=5)
tk.Button(root, text="Update Portfolio", command=update_portfolio, bg="lightgreen").pack(pady=5)

# Load initial data
load_portfolio()

root.mainloop()

```

This screenshot shows the Visual Studio Code interface with the file `portfolio_dapp.py` open in the editor. The code defines a Solidity smart contract named `PersonalPortfolio` with a single function `updatePortfolio`. The interface includes a sidebar with icons for file operations like copy, paste, and delete, and a bottom status bar showing the current file path, line count (Ln 148), column count (Col 16), and other system information.

```
portfolio_dapp.py 3
portfolio_dapp.py > ...
1 import tkinter as tk
2 from tkinter import messagebox
3 from web3 import Web3
4 from web3.providers.eth_tester import EthereumTesterProvider
5 from solcx import compile_source, install_solidity
6
7 # Install Solidity compiler
8 install_solidity("0.8.20")
9
10 # Solidity Contract
11 contract_source_code = """
12 // SPDX-License-Identifier: MIT
13 pragma solidity ^0.8.20;
14
15 contract PersonalPortfolio {
16
17     struct Portfolio {
18         string name;
19         string role;
20         string description;
21     }
22
23     Portfolio private portfolio;
24
25     event PortfolioUpdated(string name, string role, string description);
26
27     constructor(string memory _name, string memory _role, string memory _description) {
28         portfolio = Portfolio(_name, _role, _description);
29     }
30
31     function updatePortfolio(
32         string memory _name,
33         string memory _role,
34         string memory _description
35     ) public {
36         portfolio = Portfolio(_name, _role, _description);
37         emit PortfolioUpdated(_name, _role, _description);
38     }
39
40     function getPortfolio()
41     public
42     view
43     returns (
44         string memory,
45         string memory,
46         string memory
47     )
48     {
49         return (portfolio.name, portfolio.role, portfolio.description);
50     }
51 }
52 """
53
54 # Compile contract
55 compiled_sol = compile_source(contract_source_code, solc_version="0.8.20")
56 contract_id, contract_interface = compiled_sol.popitem()
57
58 # Setup Web3
59 w3 = Web3(EthereumTesterProvider())
60 w3.eth.default_account = w3.eth.accounts[0]
61
62 # Deploy contract
63 Portfolio = w3.eth.contract(
64     abi=contract_interface['abi'],
65     bytecode=contract_interface['bin']
66 )
67
68 tx_hash = Portfolio.constructor(
69     "John Doe",
70     "Blockchain Developer",
71     "Building decentralized applications"
72 ).transact()
73
74 tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)
75
76 contract = w3.eth.contract(
77     address=tx_receipt.contractAddress,
78     abi=contract_interface['abi']
79 )
80
81 # ===== GUI =====
82
83 root = tk.Tk()
84 root.title("Personal Portfolio DApp (Python GUI)")
85 root.geometry("500x400")
86
```

This screenshot shows the continuation of the `portfolio_dapp.py` code in VS Code. It includes the deployment of the Solidity contract and the creation of a Python GUI using Tkinter. The code defines a `Portfolio` class that interacts with the deployed contract. The interface remains consistent with the previous screenshot, featuring the same sidebar and status bar.

```
portfolio_dapp.py 3
portfolio_dapp.py > ...
43     returns (
44         string memory,
45         string memory,
46         string memory
47     )
48     {
49         return (portfolio.name, portfolio.role, portfolio.description);
50     }
51 }
52 """
53
54 # Compile contract
55 compiled_sol = compile_source(contract_source_code, solc_version="0.8.20")
56 contract_id, contract_interface = compiled_sol.popitem()
57
58 # Setup Web3
59 w3 = Web3(EthereumTesterProvider())
60 w3.eth.default_account = w3.eth.accounts[0]
61
62 # Deploy contract
63 Portfolio = w3.eth.contract(
64     abi=contract_interface['abi'],
65     bytecode=contract_interface['bin']
66 )
67
68 tx_hash = Portfolio.constructor(
69     "John Doe",
70     "Blockchain Developer",
71     "Building decentralized applications"
72 ).transact()
73
74 tx_receipt = w3.eth.wait_for_transaction_receipt(tx_hash)
75
76 contract = w3.eth.contract(
77     address=tx_receipt.contractAddress,
78     abi=contract_interface['abi']
79 )
80
81 # ===== GUI =====
82
83 root = tk.Tk()
84 root.title("Personal Portfolio DApp (Python GUI)")
85 root.geometry("500x400")
86
```

```
portfolio_dapp.py 3
portfolio_dapp.py > ...
92 tk.Label(root, text="Role").pack()
93 role_entry = tk.Entry(root, width=50)
94 role_entry.pack()
95
96 tk.Label(root, text="Description").pack()
97 desc_entry = tk.Entry(root, width=50)
98 desc_entry.pack()
99
100 output_text = tk.Text(root, height=8, width=60)
101 output_text.pack(pady=10)
102
103 # Functions
104 def load_portfolio():
105     name, role, description = contract.functions.getPortfolio().call()
106
107     name_entry.delete(0, tk.END)
108     role_entry.delete(0, tk.END)
109     desc_entry.delete(0, tk.END)
110
111     name_entry.insert(0, name)
112     role_entry.insert(0, role)
113     desc_entry.insert(0, description)
114
115     output_text.insert(tk.END, "Loaded portfolio from blockchain\n")
116
117
118 def update_portfolio():
119     name = name_entry.get()
120     role = role_entry.get()
121     description = desc_entry.get()
122
123     try:
124         tx = contract.functions.updatePortfolio(
125             name, role, description
126         ).transact()
127
128         receipt = w3.eth.wait_for_transaction_receipt(tx)
129
130         output_text.insert(
131             tk.END,
132             f"Portfolio updated!\nTx Hash: {receipt.transactionHash.hex()}\n"
133         )
134
135         messagebox.showinfo("Success", "Portfolio Updated Successfully!")
136
137     except Exception as e:
138         messagebox.showerror("Error", str(e))
139
140
141 # Buttons
142 tk.Button(root, text="Load Portfolio", command=load_portfolio, bg="lightblue").pack(pady=5)
143 tk.Button(root, text="Update Portfolio", command=update_portfolio, bg="lightgreen").pack(pady=5)
144
145 # Load initial data
146 load_portfolio()
147
148 root.mainloop()
```