BlockChain Engineering Assignment-06

Name : B.Sarayu

Ht.no: 2303A51842

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")
```

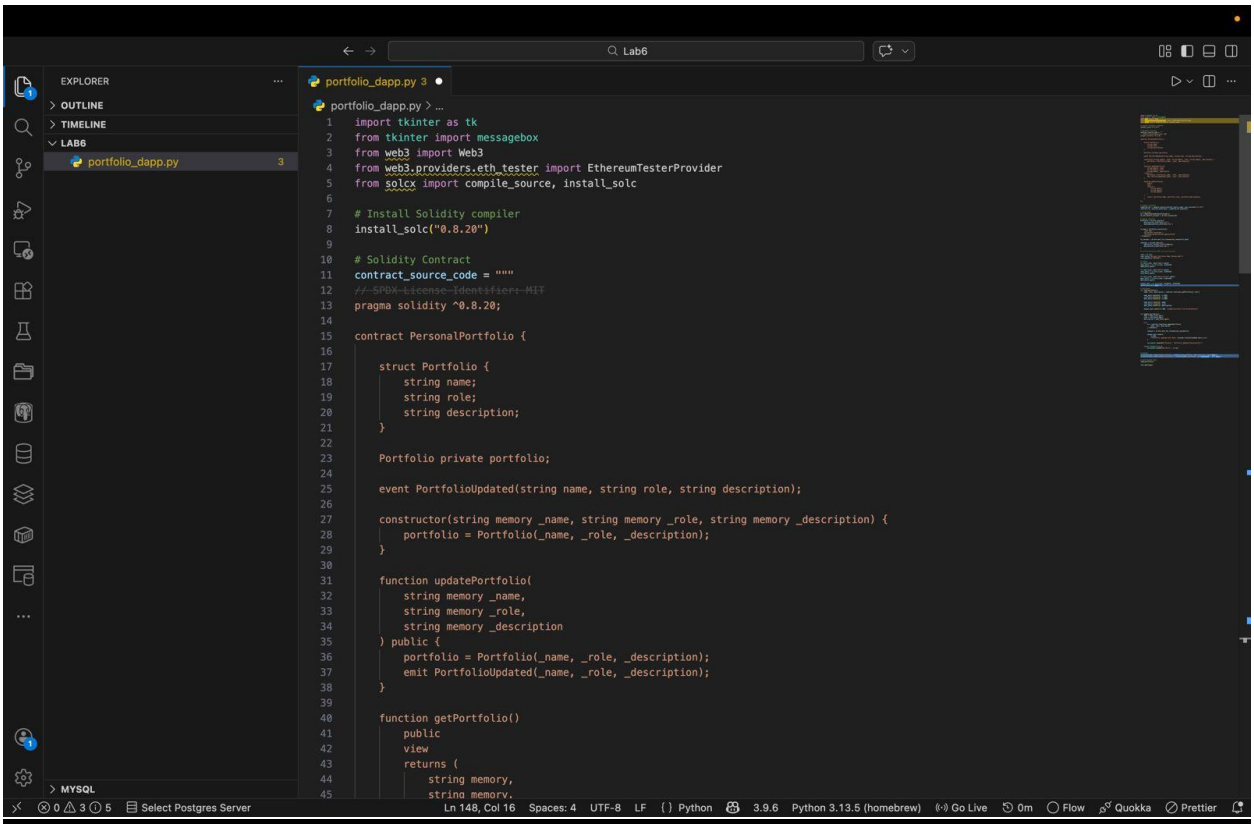
```
# Labels
tk.Label(root, text="Name").pack()
name_entry = tk.Entry(root, width=50)
name_entry.pack()

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)
```

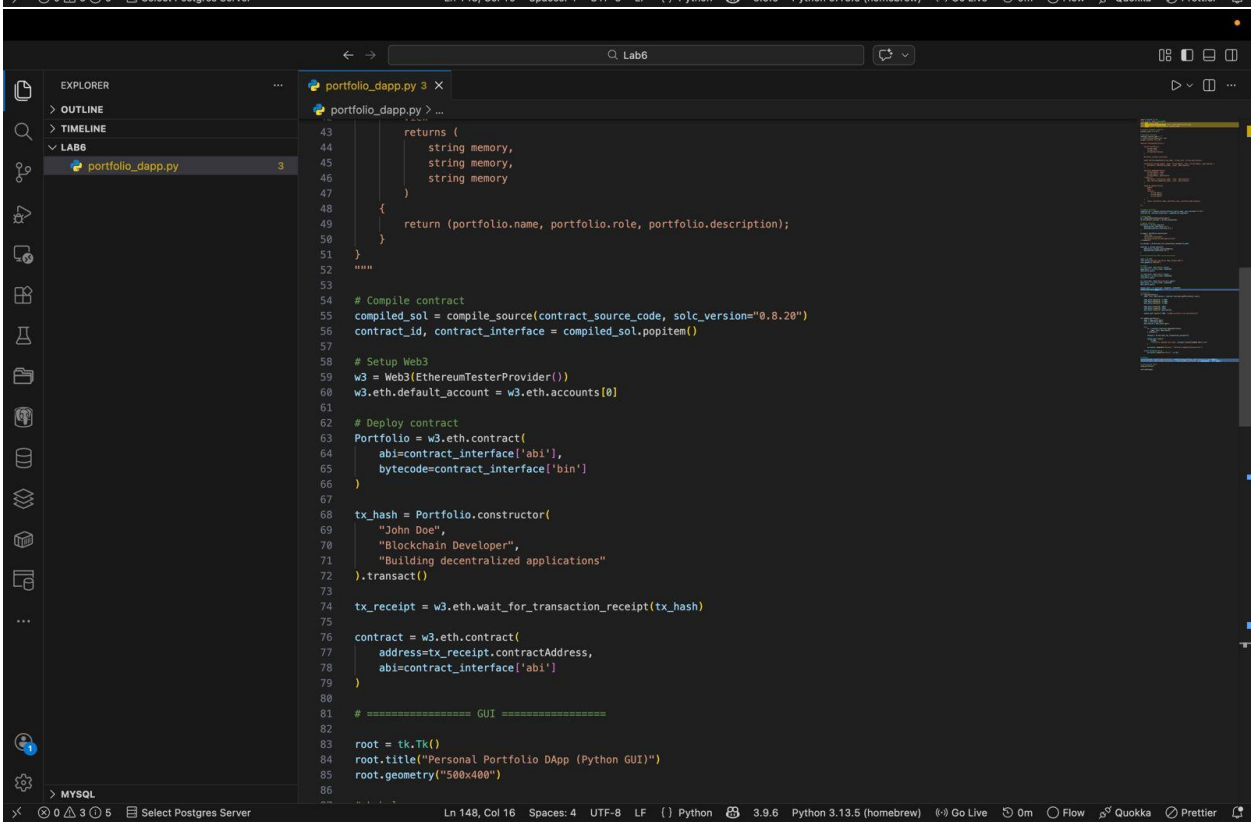
```
        output_text.insert(
            tk.END,
            f"Portfolio updated!\nTx Hash: {receipt.transactionHash.hex()}\n\n"
        )

        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;
```

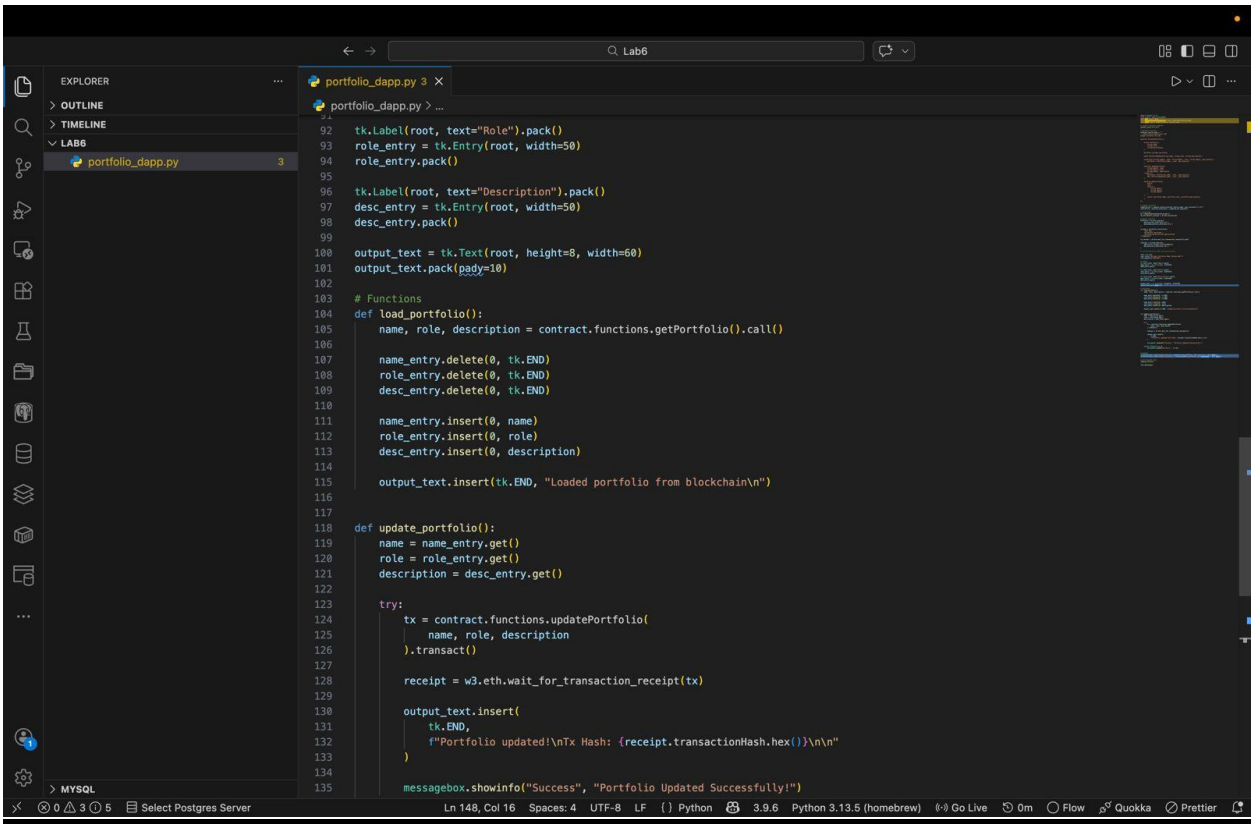
```
    event PortfolioUpdated(string name, string role, string description);

    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(
```

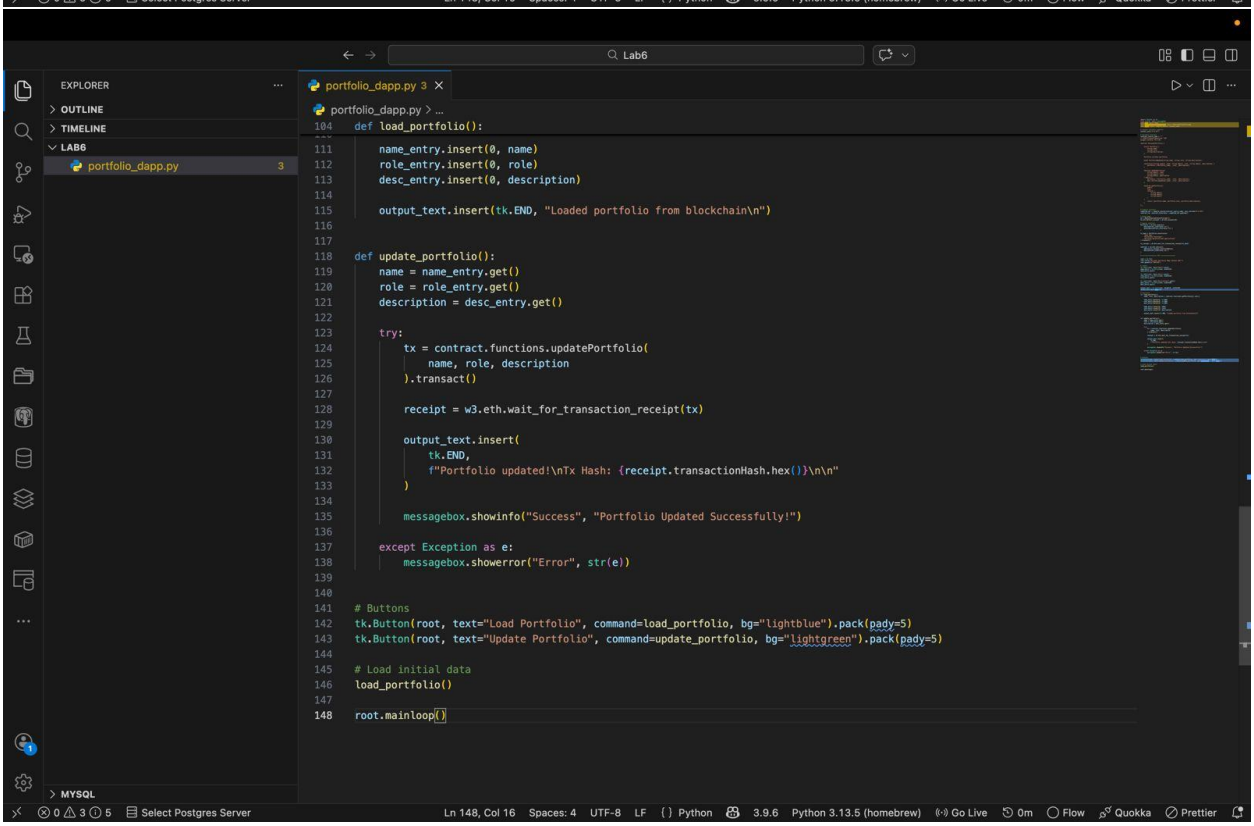
```
    "John Doe",
    "Blockchain Developer",
    "Building decentralized applications"
).transact()

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()
```


Lab6
EXPLORER
portfolio_dapp.py 3
OUTLINE
TIMELINE
LAB6
portfolio_dapp.py 3
MYSQL
portfolio_dapp.py > ...
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,
Select Postgres Server
Ln 148, Col 16
Spaces: 4
UTF-8
LF
Python
3.9.6
Python 3.13.5 (homebrew)
Go Live
0m
Flow
Quokka
Prettier


Lab6
EXPLORER
portfolio_dapp.py 3
OUTLINE
TIMELINE
LAB6
portfolio_dapp.py 3
MYSQL
portfolio_dapp.py > ...
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")
Select Postgres Server
Ln 148, Col 16
Spaces: 4
UTF-8
LF
Python
3.9.6
Python 3.13.5 (homebrew)
Go Live
0m
Flow
Quokka
Prettier


Lab6
EXPLORER
portfolio_dapp.py 3
OUTLINE
TIMELINE
LAB6
portfolio_dapp.py 3
MYSQL
portfolio_dapp.py > ...
92 tk.Label(root, text="Role").pack()
93 role_entry = tk.Entry(root, width=50)
94 role_entry.pack()
96 tk.Label(root, text="Description").pack()
97 desc_entry = tk.Entry(root, width=50)
98 desc_entry.pack()
100 output_text = tk.Text(root, height=8, width=60)
101 output_text.pack(pady=10)
103 # Functions
104 def load_portfolio():
105 name, role, description = contract.functions.getPortfolio().call()
107 name_entry.delete(0, tk.END)
108 role_entry.delete(0, tk.END)
109 desc_entry.delete(0, tk.END)
111 name_entry.insert(0, name)
112 role_entry.insert(0, role)
113 desc_entry.insert(0, description)
115 output_text.insert(tk.END, "Loaded portfolio from blockchain\n")
118 def update_portfolio():
119 name = name_entry.get()
120 role = role_entry.get()
121 description = desc_entry.get()
123 try:
124 tx = contract.functions.updatePortfolio(
125 name, role, description
126).transact()
128 receipt = w3.eth.wait_for_transaction_receipt(tx)
130 output_text.insert(
131 tk.END,
132 f"Portfolio updated!\nTx Hash: {receipt.transactionHash.hex()}\n\n"
133)
135 messagebox.showinfo("Success", "Portfolio Updated Successfully!")
Select Postgres Server
Ln 148, Col 16 Spaces: 4 UTF-8 LF Python 3.9.6 Python 3.13.5 (homebrew) Go Live 0m Flow Quokka Prettier


Lab6
EXPLORER
portfolio_dapp.py 3
OUTLINE
TIMELINE
LAB6
portfolio_dapp.py 3
MYSQL
portfolio_dapp.py > ...
104 def load_portfolio():
111 name_entry.insert(0, name)
112 role_entry.insert(0, role)
113 desc_entry.insert(0, description)
115 output_text.insert(tk.END, "Loaded portfolio from blockchain\n")
118 def update_portfolio():
119 name = name_entry.get()
120 role = role_entry.get()
121 description = desc_entry.get()
123 try:
124 tx = contract.functions.updatePortfolio(
125 name, role, description
126).transact()
128 receipt = w3.eth.wait_for_transaction_receipt(tx)
130 output_text.insert(
131 tk.END,
132 f"Portfolio updated!\nTx Hash: {receipt.transactionHash.hex()}\n\n"
133)
135 messagebox.showinfo("Success", "Portfolio Updated Successfully!")
137 except Exception as e:
138 messagebox.showerror("Error", str(e))
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)
145 # Load initial data
146 load_portfolio()
148 root.mainloop()
Select Postgres Server
Ln 148, Col 16 Spaces: 4 UTF-8 LF Python 3.9.6 Python 3.13.5 (homebrew) Go Live 0m Flow Quokka Prettier