

ASSIGNMENT – 3

NAME: SRAVANI

HT NO:2303A510G7

BATCH NO:30

OBJECTIVE

To study the core concepts of Solidity smart contract development through the implementation of a basic Ethereum smart contract that enables user input, securely stores the data on the blockchain, and allows retrieval of the stored information.

PROBLEM STATEMENT

Develop a basic Solidity smart contract that allows users to:

- Store a message on the blockchain
- Update the message
- Retrieve the stored message

CODE:

```
# MessageStorage simulation in Python
# This simulates the Solidity smart contract behavior
class MessageStorage:
    def __init__(self, message):
        # Constructor: initialize message
        self.message = message
    def setMessage(self, new_message):
        # Update message
        self.message = new_message
    def getMessage(self):
        # Retrieve message
```

```
    return self.message
```

```
# Deploy contract with initial message
```

```
contract = MessageStorage("Hello Blockchain")
```

```
# Retrieve initial message
```

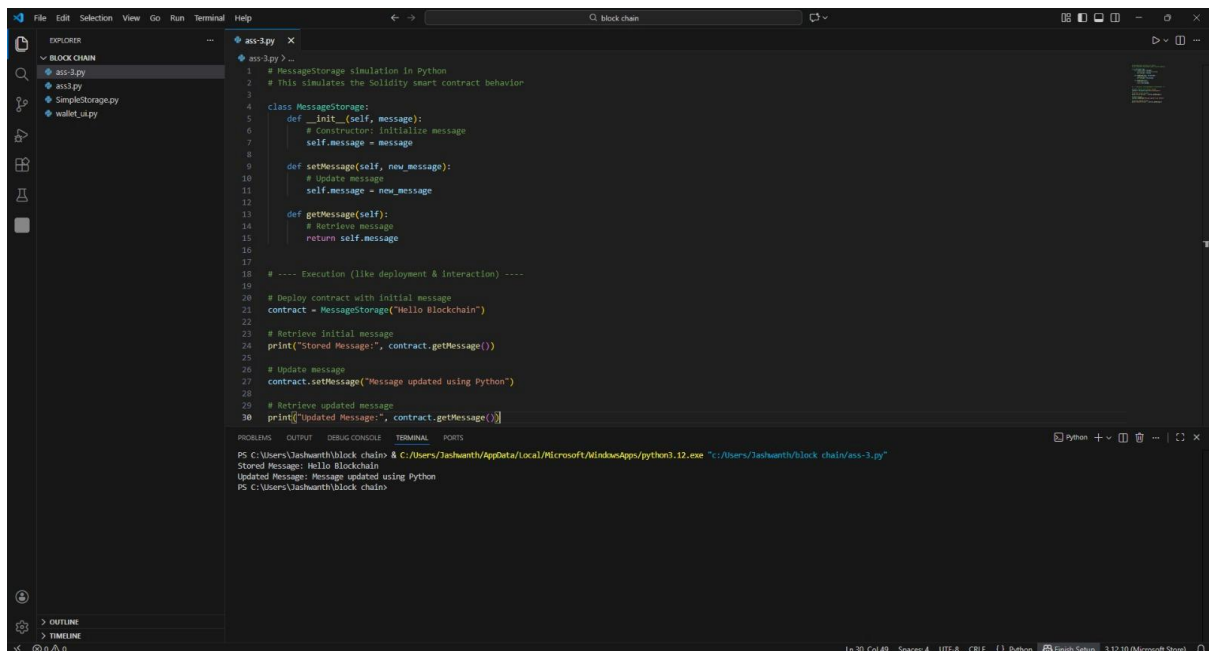
```
print("Stored Message:", contract.getMessage())
```

```
# Update message
```

```
contract.setMessage("Message updated using Python")
```

```
# Retrieve updated message
```

```
print("Updated Message:", contract.getMessage())
```



The screenshot shows a Visual Studio Code editor window with a file explorer on the left displaying a project named 'BLOCK CHAIN' containing files 'ass3.py', 'SimpleStorage.py', and 'wallet.py'. The main editor displays the content of 'ass3.py', which is a Python script simulating a Solidity smart contract. The script defines a 'MessageStorage' class with methods for initialization, setting a message, and retrieving a message. Below the class definition, there is a section for execution (deployment and interaction) that creates a 'contract' object, prints the initial message, updates the message, and prints the updated message. At the bottom, a terminal window shows the output of running the script: 'Stored Message: Hello Blockchain' and 'Updated Message: Message updated using Python'.

```
1 # MessageStorage simulation in Python
2 # This simulates the Solidity smart contract behavior
3
4 class MessageStorage:
5     def __init__(self, message):
6         # Constructor: Initialize message
7         self.message = message
8
9     def setMessage(self, new_message):
10        # Update message
11        self.message = new_message
12
13    def getMessage(self):
14        # Retrieve message
15        return self.message
16
17
18 # ---- Execution (like deployment & interaction) ----
19
20 # Deploy contract with initial message
21 contract = MessageStorage("Hello Blockchain")
22
23 # Retrieve initial message
24 print("Stored Message:", contract.getMessage())
25
26 # Update message
27 contract.setMessage("Message updated using Python")
28
29 # Retrieve updated message
30 print("Updated Message:", contract.getMessage())
```

PS C:\Users\Jashwanth\block chain> C:\Users\Jashwanth\AppData\Local\Microsoft\WindowsApps\python3.12.exe "c:\Users\Jashwanth\block chain\ass-3.py"

Stored Message: Hello Blockchain

Updated Message: Message updated using Python

PS C:\Users\Jashwanth\block chain>