

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
<b>Program Name:</b> B. Tech		<b>Assignment Type:</b> Lab	
<b>Course Coordinator Name</b>		Dr. Jagjit Singh Dhatterwal	
<b>Instructor(s) Name</b>		Dr. Jagjit Singh Dhatterwal	
<b>Course Code</b>	23CS201PE401	<b>Course Title</b>	Blockchain Engineering
<b>Year/Sem</b>	III/II	<b>Regulation</b>	R25
<b>Date and Day of Assignment</b>	12-01-2026	<b>Time(s)</b>	9:00AM to 11:00AM
<b>Duration</b>	2 Hours	<b>Applicable to Batches</b>	(23CSBTB19, 23CSBTB20 23CSBTB21, 23CSBTB22 23CSBTB23, 23CSBTB24 23CSBTB25, 23CSBTB26)
<b>Assignment Number:</b> 02/12			
<b>Q. No.</b>	<b>Question</b>		<b>Expected Time to complete</b>
1	<p><b>Title</b></p> <p>To design, deploy, and interact with a Simple Storage smart contract using the Solidity programming language in the Remix Ethereum IDE, in order to demonstrate fundamental blockchain concepts such as state variables, smart contract functions, and blockchain transactions.</p>		
	<p><b>Tools / Software Required</b></p> <ul style="list-style-type: none"> <li>• Web Browser (Chrome / Firefox)</li> <li>• <b>Remix Ethereum IDE</b> (online)</li> <li>• Solidity Compiler (built-in with Remix)</li> <li>• JavaScript VM (Remix default environment)</li> </ul>		
	<p><b>Theory (Brief)</b></p> <p>A <b>smart contract</b> is a self-executing program stored on the blockchain.</p> <p>The <b>Simple Storage contract</b> is a beginner-level Solidity contract that allows users to:</p> <ul style="list-style-type: none"> <li>• Store a value on the blockchain</li> <li>• Retrieve the stored value</li> </ul> <p>Remix IDE is a web-based development environment used to write, compile, deploy, and test Solidity smart contracts.</p>		

## Requirements

- Create a Solidity smart contract named SimpleStorage
- Declare a state variable to store data
- Implement functions to:
  - Store a value
  - Retrieve the stored value
- Deploy the contract using Remix
- Test the contract with multiple inputs
- Add comments explaining the logic
- Capture screenshots of:
  - Contract compilation
  - Contract deployment
  - Function execution

## Procedure / Steps

### Step 1: Open Remix IDE

1. Open a web browser
2. Go to: <https://remix.ethereum.org>
3. Create a new file named:
4. SimpleStorage.sol

### Step 2: Write the Solidity Code

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

// Simple Storage Smart Contract
contract SimpleStorage {

    // State variable to store a number
    uint256 private storedValue;

    // Function to store a value on the
    // blockchain
    function set(uint256 _value) public {
        storedValue = _value;
    }

    // Function to retrieve the stored value
    function get() public view returns
    (uint256) {
        return storedValue;
    }
}
```

## Code Explanation

- `pragma solidity ^0.8.0;`  
Specifies the Solidity compiler version.
- `uint256 private storedValue;`  
Declares a private state variable stored on the blockchain.
- `set(uint256 _value)`  
Stores a value by creating a transaction.
- `get()`  
Reads the stored value without modifying blockchain data.

### Step 3: Compile the Contract

1. Click on **Solidity Compiler** tab
2. Select compiler version `0.8.x`
3. Click **Compile SimpleStorage.sol**
4. Ensure there are **no errors**

### Step 4: Deploy the Contract

1. Go to **Deploy & Run Transactions** tab
2. Select **JavaScript VM (London)** as environment
3. Click **Deploy**
4. Contract instance appears under **Deployed Contracts**

## Code Screenshot GUI:

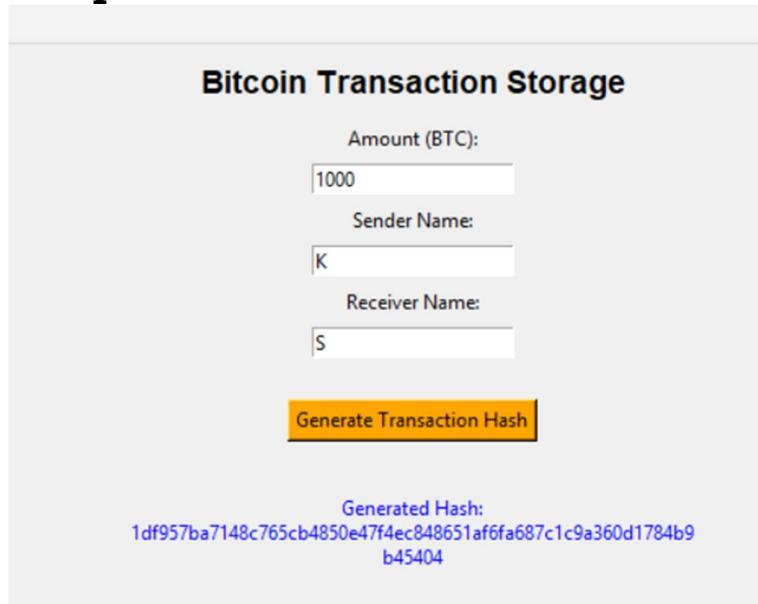
The screenshot shows a Python code editor interface. The code is written in Python and uses the Tkinter library to create a simple GUI for generating a Bitcoin transaction hash. The code includes functions for generating the hash, validating input fields, and displaying the result.

```
PYTHON_ X
PYTHON_ > ...
1 import tkinter as tk
2 from tkinter import messagebox
3 import hashlib
4
5 def generate_hash():
6     # Get values from the GUI input boxes
7     amount = entry_amount.get()
8     sender = entry_sender.get()
9     receiver = entry_receiver.get()
10
11    if not amount or not sender or not receiver:
12        messagebox.showwarning("Input Error", "Please fill all fields")
13        return
14
15    # Create the Hashing Value (The Encryption Check)
16    raw_data = f"{amount}{sender}{receiver}"
17    tx_hash = hashlib.sha256(raw_data.encode()).hexdigest()
18
19    # Display the result in the GUI
20    label_hash_result.config(text=f"Generated Hash:\n{tx_hash}")
21
22    # Instructions for the user
23    print(f"--- Copy this to Remix ---")
24    print(f"Amount: {amount}")
25    print(f"Hash: {tx_hash}")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR
```

Output terminal:  
--- Copy this to Remix ---  
Amount: 1000  
Hash: 1df957ba7148c765cb4850e47f4ec848651af6fa687c1c9a360d1784b9b45404  
--- Copy this to Remix ---  
Amount: 1000  
Hash: 1df957ba7148c765cb4850e47f4ec848651af6fa687c1c9a360d1784b9b45404  
--- Copy this to Remix ---  
Amount: 1000  
Hash: 1df957ba7148c765cb4850e47f4ec848651af6fa687c1c9a360d1784b9b45404

## Output :



## Ethernium Code :

```
REMIX 1.5.1 | default_workspace | Login with GitHub | * Theme |
```

Compiled Home SimpleStorage.sol

```
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.0;
3
4 // Simple Storage Smart Contract
5 contract SimpleStorage {
6     // State variable to store a number
7     uint256 private storedValue;
8
9     // Function to store a value on the blockchain
10    function set(uint256 _value) public {
11        storedValue = _value;
12    }
13
14    // Function to retrieve the stored value
15    function get() public view returns (uint256) {
16        return storedValue;
17    }
18 }
19 }
```

Explain contract

creation of SimpleStorage pending...

[vm] from: 0x5B3...eddC4 to: SimpleStorage.(constructor) value: 0 wei data: 0x608...f0033 logs: 0 hash: 0x822...12707

0 Listen on all transactions Filter with transaction hash or address

Debug

## **Output :**

The screenshot shows the REMIX IDE interface. At the top, there's a navigation bar with tabs for 'REMIX' and '1.5.1'. To the right of the tabs are icons for saving, opening, and closing the workspace, labeled 'default\_workspace'. Below the navigation bar is a toolbar with various icons: a green play button, a blue 'DEBUGGER' button, a magnifying glass, a file icon, and a gear icon. The main workspace contains a code editor with the following Solidity code:

```
000 PUSH1 0x00 - LINE 5
002 PUSH1 0x00 - LINE 5
004 MSTORE - LINE 5
005 CALVALUE - LINE 5
006 DUP1 - LINE 5
007 SZERO - LINE 5
008 PUSH1 0x00 - LINE 5
010 IMP1 - LINE 5
```

Below the code editor is a message box stating 'Use generated sources(Solidity >= v0.7.2)' with the identifier '0x82230b3b55cf952b30100e04964d01536ff7e38143331210565fd986c1212707'. A large blue button labeled 'Stop debugging' is centered at the bottom of the workspace.

## DEBUGGER

Use generated sources(Solidity >= v0.7.2)

0x82230b3b55cf952b30100e04964d01536ff7e38f43331210565fd986c1212707

Stop debugging



▼ RETURN VALUE ▶

0: Object

### ▼ Global Variables ▶

```
block.chainid: 3333
block.coinbase: 0x0000000000000000000000000000000000000000000000000000000000
block.difficulty: 0
block.gaslimit: 135336
block.number: 3
block.timestamp: 1769661887
msg.sender: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
msg.sig:
msg.value: 0 Wei
tx.origin: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
block.basefee: 1 Wei (1)
```