Name: Vivek Dubey

Roll No.: COBB108

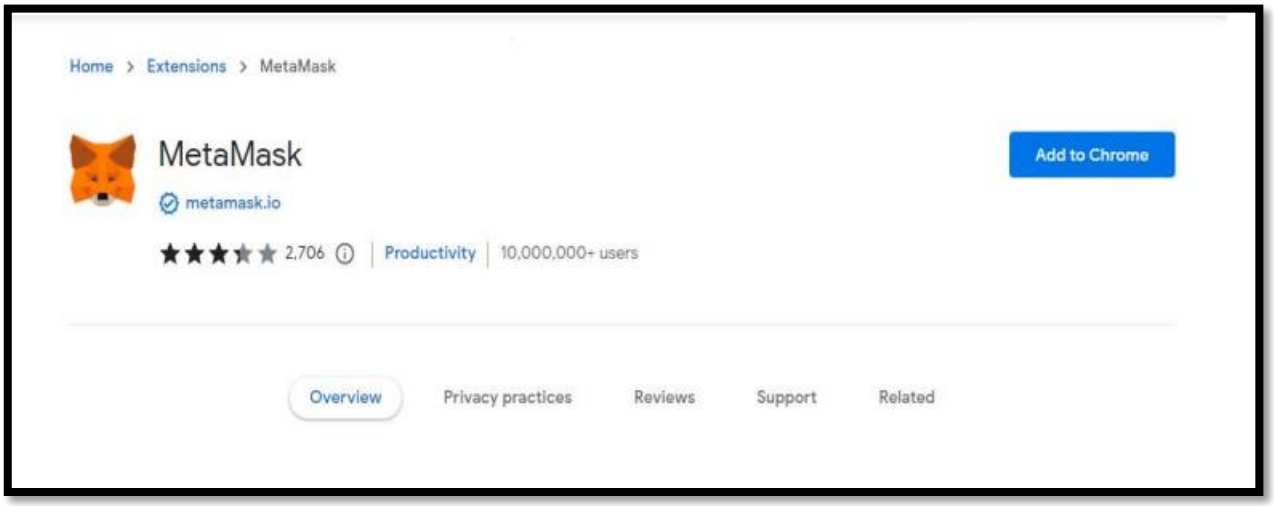
Assignment No.: 1

Title: Installation of MetaMask and study spending Ether per transaction.

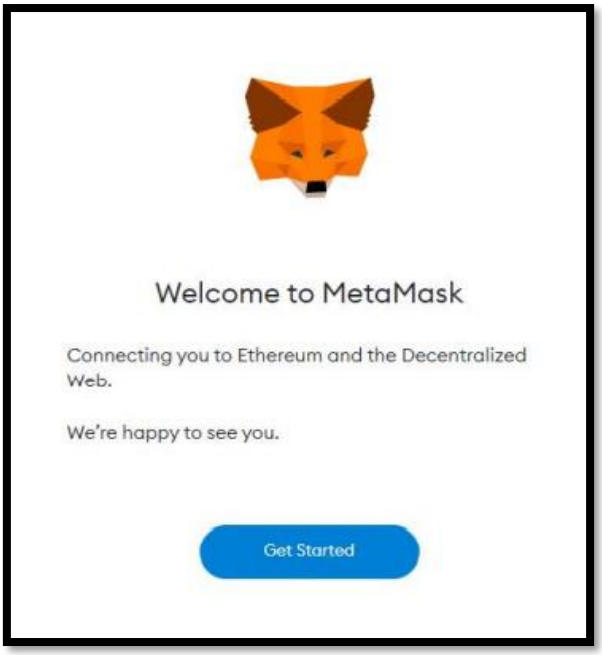
Output:

Steps to create MetaMask.

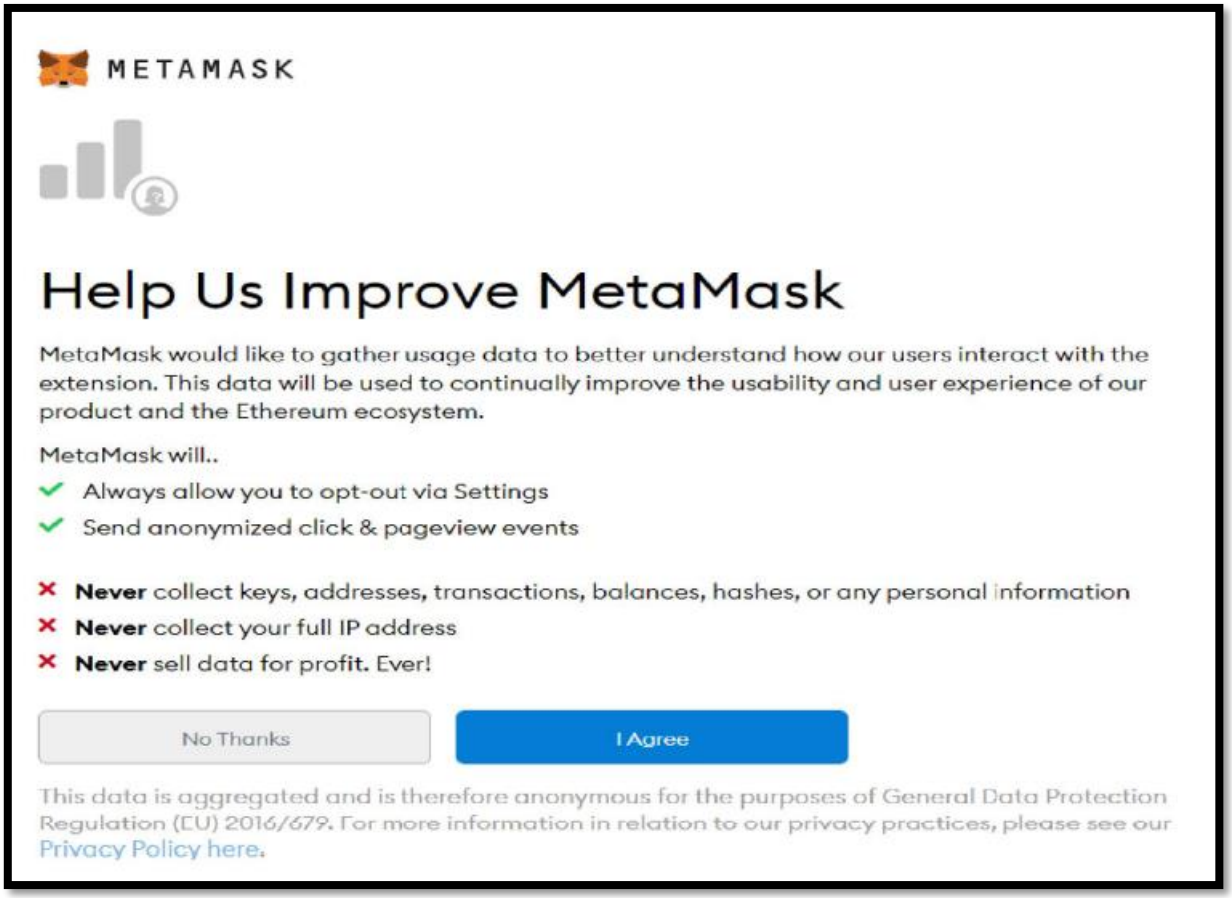
Step 1:



Step 2:



Step 3:


METAMASK
Help Us Improve MetaMask
MetaMask would like to gather usage data to better understand how our users interact with the extension. This data will be used to continually improve the usability and user experience of our product and the Ethereum ecosystem.
MetaMask will..
Always allow you to opt-out via Settings
Send anonymized click & pageview events
Never collect keys, addresses, transactions, balances, hashes, or any personal information
Never collect your full IP address
Never sell data for profit. Ever!
No Thanks
I Agree
This data is aggregated and is therefore anonymous for the purposes of General Data Protection Regulation (EU) 2016/679. For more information in relation to our privacy practices, please see our Privacy Policy here.

Name: Vivek Dubey

Roll No.: COBB108

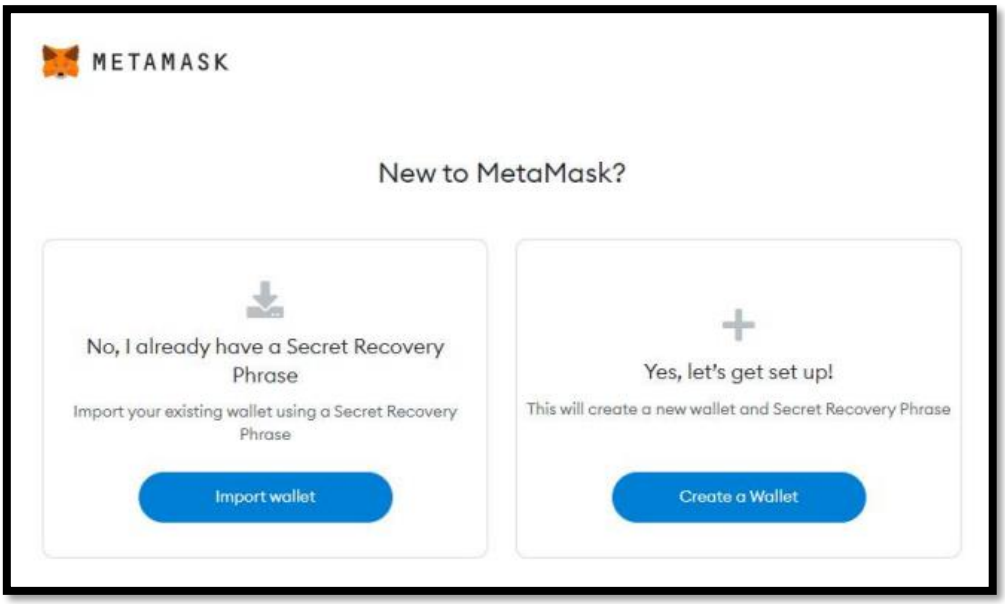
Assignment No.: 2

Title: Create your own wallet using MetaMask for crypto transactions.

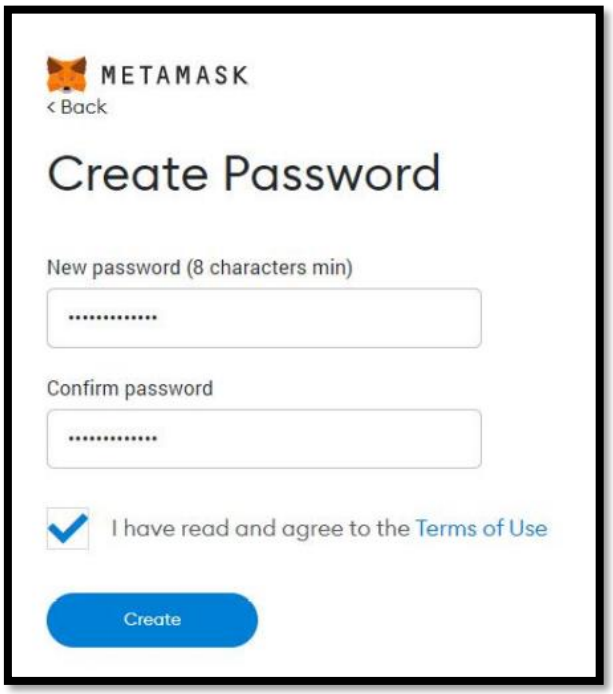
Output:

Steps to create MetaMask wallet.

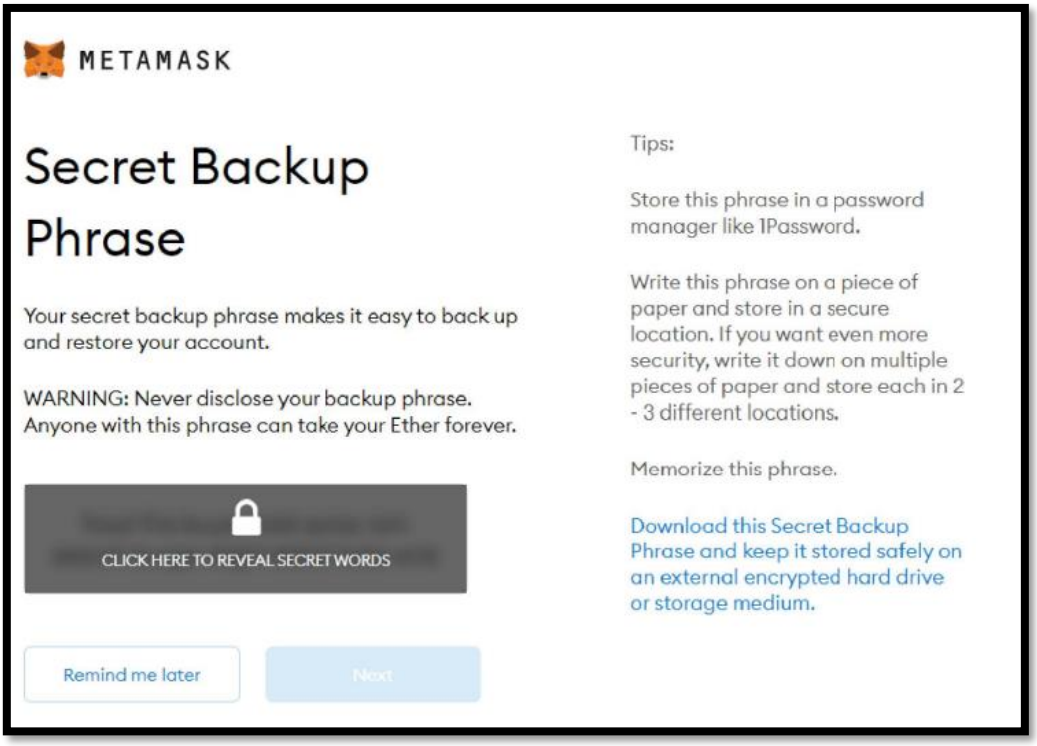
Step 1:



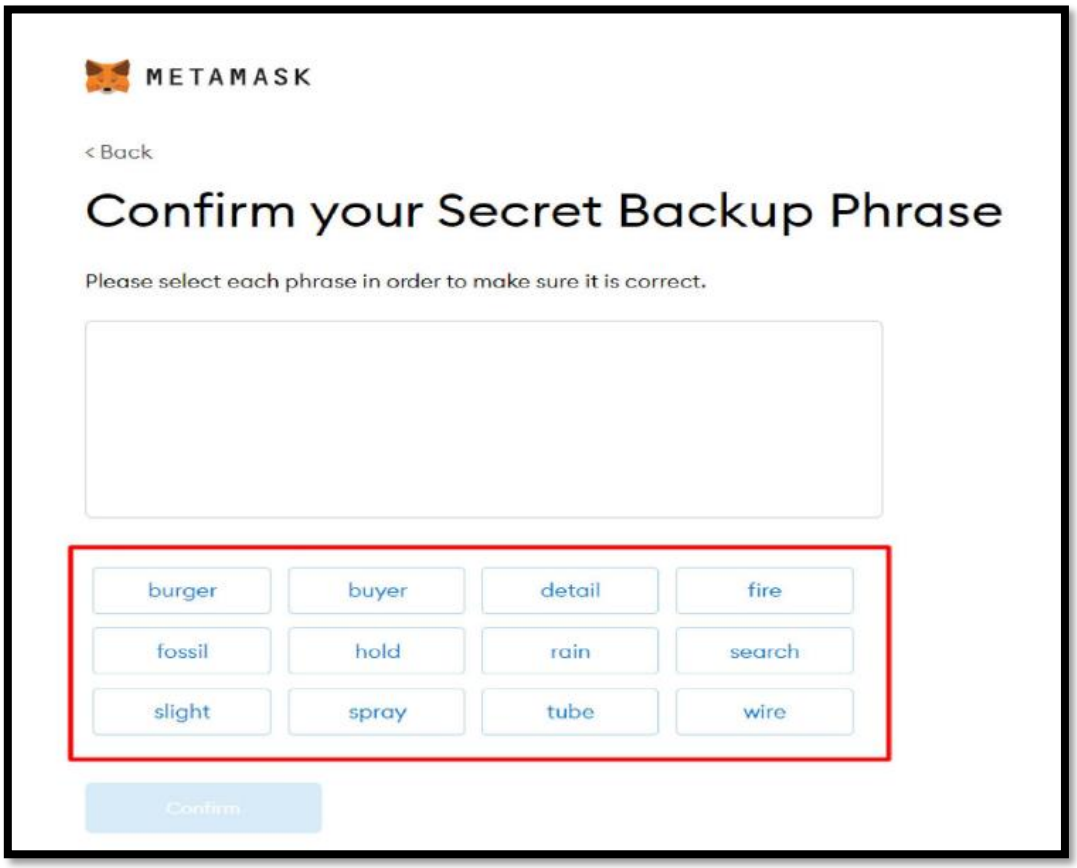
Step 2:



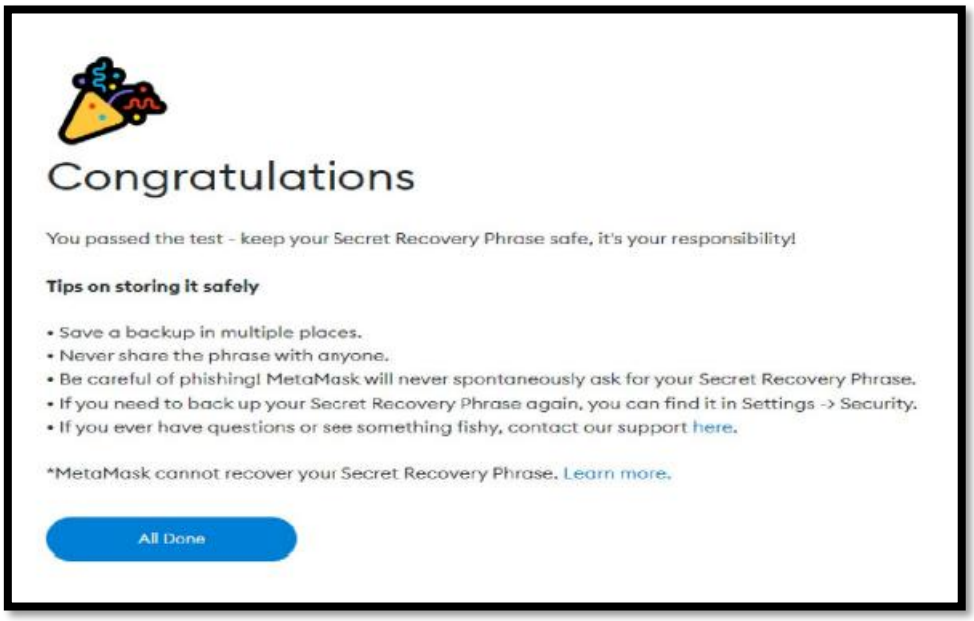
Step 3:


METAMASK
Secret Backup Phrase
Your secret backup phrase makes it easy to back up and restore your account.
WARNING: Never disclose your backup phrase. Anyone with this phrase can take your Ether forever.
CLICK HERE TO REVEAL SECRET WORDS
Remind me later
Next
Tips:
Store this phrase in a password manager like 1Password.
Write this phrase on a piece of paper and store in a secure location. If you want even more security, write it down on multiple pieces of paper and store each in 2 - 3 different locations.
Memorize this phrase.
Download this Secret Backup Phrase and keep it stored safely on an external encrypted hard drive or storage medium.

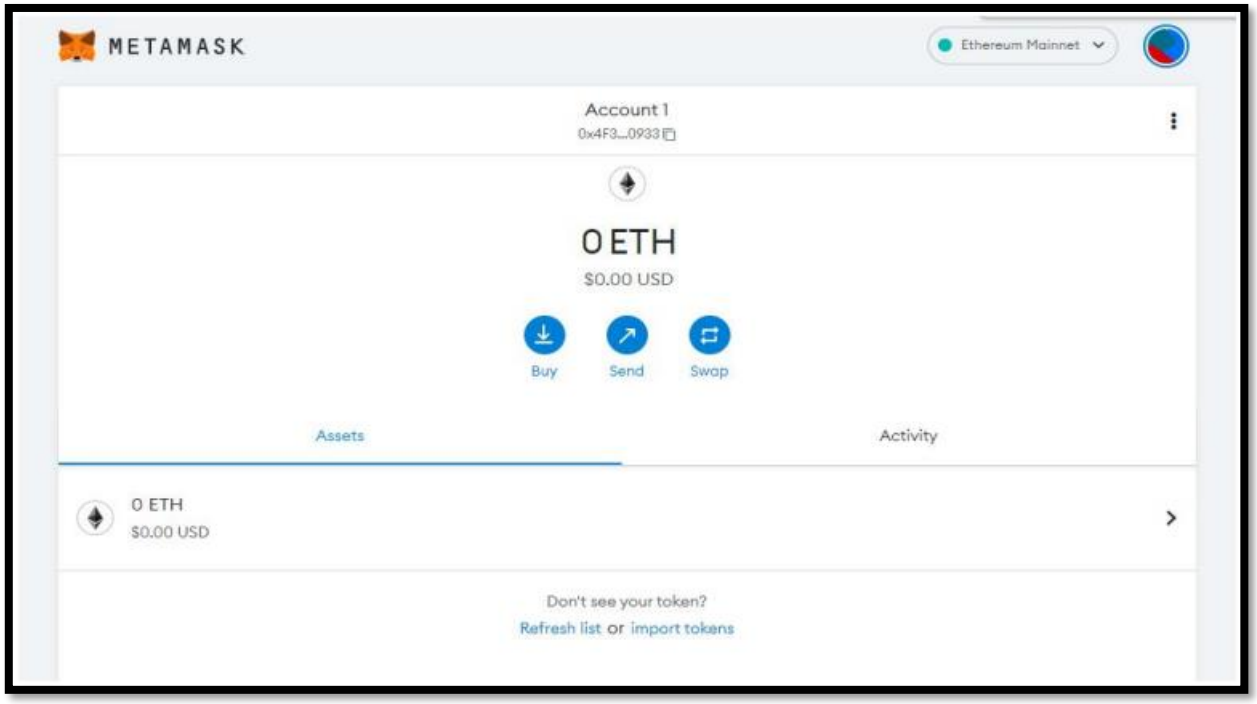
Step 4:


METAMASK
< Back
Confirm your Secret Backup Phrase
Please select each phrase in order to make sure it is correct.
burger
buyer
detail
fire
fossil
hold
rain
search
slight
spray
tube
wire
Confirm

Step 5:


Congratulations
You passed the test - keep your Secret Recovery Phrase safe, it's your responsibility!
Tips on storing it safely
• Save a backup in multiple places.
• Never share the phrase with anyone.
• Be careful of phishing! MetaMask will never spontaneously ask for your Secret Recovery Phrase.
• If you need to back up your Secret Recovery Phrase again, you can find it in Settings -> Security.
• If you ever have questions or see something fishy, contact our support here.
*MetaMask cannot recover your Secret Recovery Phrase. Learn more.
All Done

Step 6:


METAMASK
Ethereum Mainnet
Account 1
0x4F3...0933
0 ETH
$0.00 USD
Buy
Send
Swap
Assets
Activity
0 ETH
$0.00 USD
Don't see your token?
Refresh list or import tokens

Name: Vivek Dubey

Roll No.: COBB108

Assignment No.: 3

Title: Write a smart contract on a test network for Bank account of a customer for following operations: • Deposit Money • Withdraw Money • Show Balance

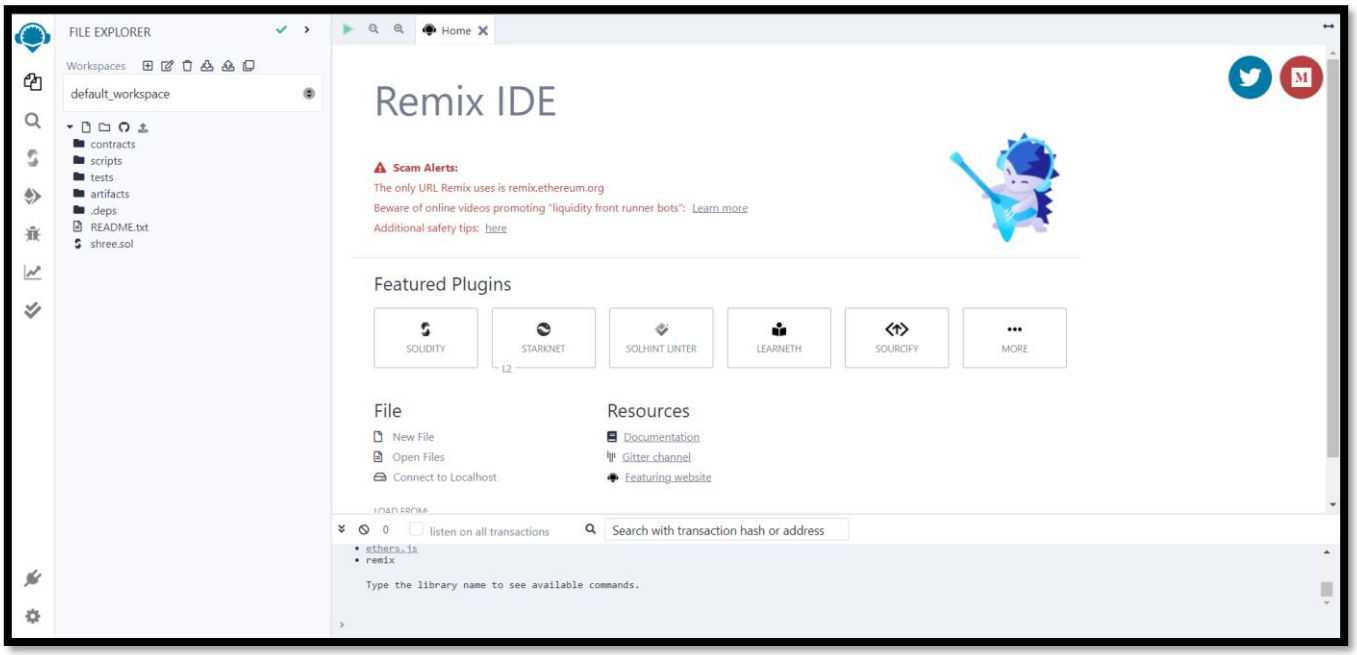
Code:

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;
contract banking
{
 mapping(address=>uint) public user_account;
 mapping(address=>bool) public user_exists;
 function create_account() public payable returns(string memory)
 {
 require(user_exists[msg.sender]==false,'Account already created');
 if(msg.value==0)
 {
user_account[msg.sender]=0;
user_exists[msg.sender]=true;
 return "Account created";
 }
 require(user_exists[msg.sender]==false,"Account already created");
user_account[msg.sender]=msg.value;
user_exists[msg.sender]=true;
 return "Account created";
 }
 function deposit() public payable returns(string memory)
 {
 require(user_exists[msg.sender]==true,"Account not created");
 require(msg.value>0,"Value for deposit is Zero");
user_account[msg.sender]=user_account[msg.sender]+msg.value;
 return "Deposited Successfully";
 }
 function withdraw(uint amount) public payable returns(string memory)
 {
 require(user_account[msg.sender]>amount,"Insufficient Balance");
 require(user_exists[msg.sender]==true,"Account not created");
 require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
msg.sender.transfer(amount);
 return "Withdrawl Successful";
 }
 function transfer(address payable userAddress, uint amount) public returns(string memory)
 {
 require(user_account[msg.sender]>amount,"Insufficient balance in Bank account");
 require(user_exists[msg.sender]==true,"Account is not created");
 require(user_exists[userAddress]==true,"Transfer account does not exist");
```

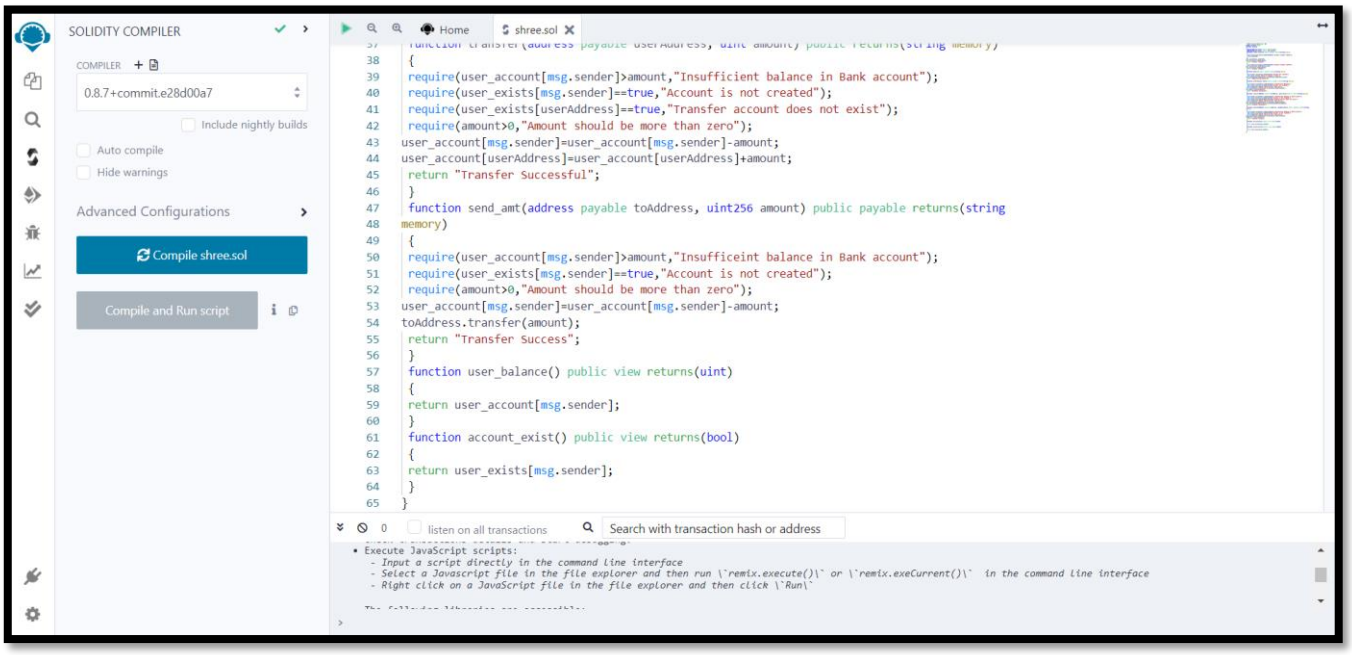
```
 require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
user_account[userAddress]=user_account[userAddress]+amount;
 return "Transfer Successful";
 }
 function send_amt(address payable toAddress, uint256 amount) public payable
returns(string
memory)
 {
 require(user_account[msg.sender]>amount,"Insufficeint balance in Bank account");
 require(user_exists[msg.sender]==true,"Account is not created");
 require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
toAddress.transfer(amount);
 return "Transfer Success";
 }
 function user_balance() public view returns(uint)
 {
 return user_account[msg.sender];
 }
 function account_exist() public view returns(bool)
 {
 return user_exists[msg.sender];
 }
}
```

Output:

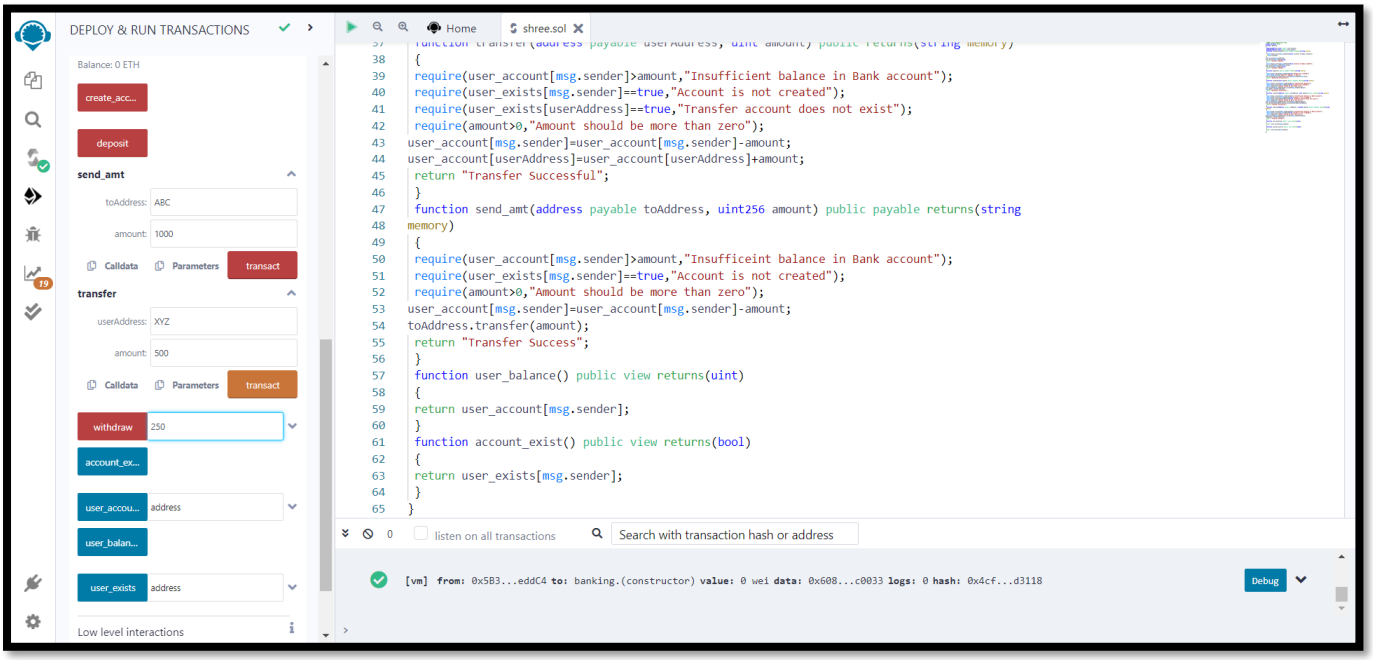
Step 1:



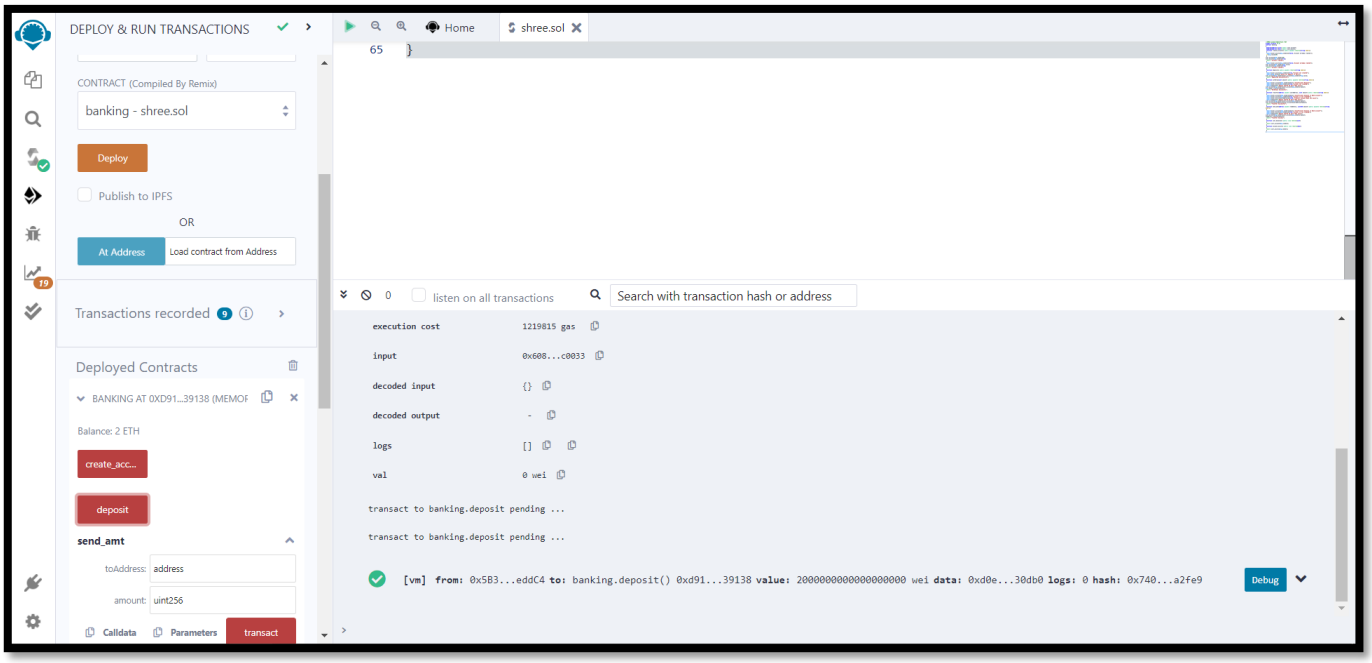
Step 2:


SOLIDITY COMPILER
COMPILER
0.8.7+commit.e28d00a7
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile shree.sol
Compile and Run script
Home
shree.sol
{
require(user_account[msg.sender]>amount,"Insufficient balance in Bank account");
require(user_exists[msg.sender]==true,"Account is not created");
require(user_exists[userAddress]==true,"Transfer account does not exist");
require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
user_account[userAddress]=user_account[userAddress]+amount;
return "Transfer Successful";
}
function send_amt(address payable toAddress, uint256 amount) public payable returns(string
memory)
{
require(user_account[msg.sender]>amount,"Insufficeint balance in Bank account");
require(user_exists[msg.sender]==true,"Account is not created");
require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
toAddress.transfer(amount);
return "Transfer Success";
}
function user_balance() public view returns(uint)
{
return user_account[msg.sender];
}
function account_exist() public view returns(bool)
{
return user_exists[msg.sender];
}
}
listen on all transactions
Search with transaction hash or address
Execute JavaScript scripts:
- Input a script directly in the command line interface
- Select a Javascript file in the file explorer and then run \`remix.execute()\` or \`remix.exeCurrent()\` in the command line interface
- Right click on a JavaScript file in the file explorer and then click \`Run\`

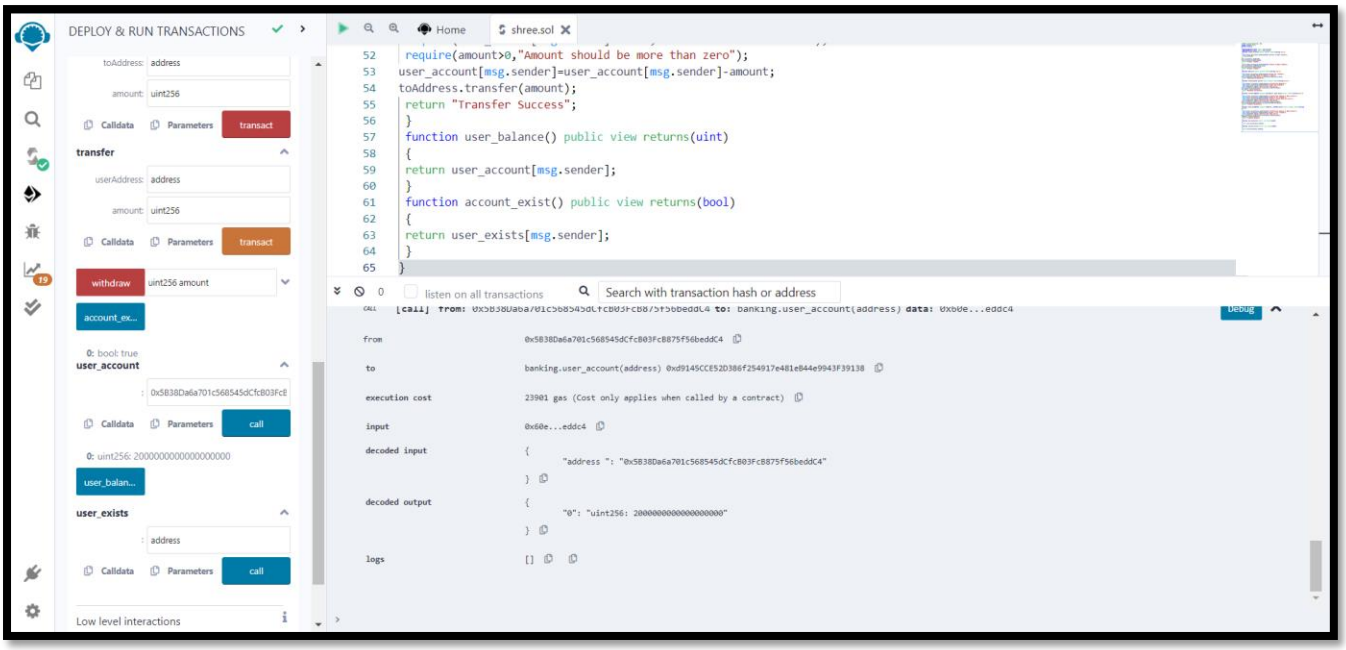
Step 3:


DEPLOY & RUN TRANSACTIONS
Balance: 0 ETH
create_acc...
deposit
send_amt
toAddress: ABC
amount: 1000
Calldata
Parameters
transact
transfer
userAddress: XYZ
amount: 500
Calldata
Parameters
transact
withdraw
250
account_ex...
user_accou...
address
user_balan...
user_exists
address
Low level interactions
Home
shree.sol
{
require(user_account[msg.sender]>amount,"Insufficient balance in Bank account");
require(user_exists[msg.sender]==true,"Account is not created");
require(user_exists[userAddress]==true,"Transfer account does not exist");
require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
user_account[userAddress]=user_account[userAddress]+amount;
return "Transfer Successful";
}
function send_amt(address payable toAddress, uint256 amount) public payable returns(string
memory)
{
require(user_account[msg.sender]>amount,"Insufficeint balance in Bank account");
require(user_exists[msg.sender]==true,"Account is not created");
require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
toAddress.transfer(amount);
return "Transfer Success";
}
function user_balance() public view returns(uint)
{
return user_account[msg.sender];
}
function account_exist() public view returns(bool)
{
return user_exists[msg.sender];
}
}
listen on all transactions
Search with transaction hash or address
[vm] from: 0x5B3...eddC4 to: banking.(constructor) value: 0 wei data: 0x608...c0033 logs: 0 hash: 0x4cf...d3118
Debug

- Deposit Amount:



- Check Account Exists
- Check User Account Exists
- Check User Balance
- Check User Exists



Name: Vivek Dubey

Roll No.: COBB108

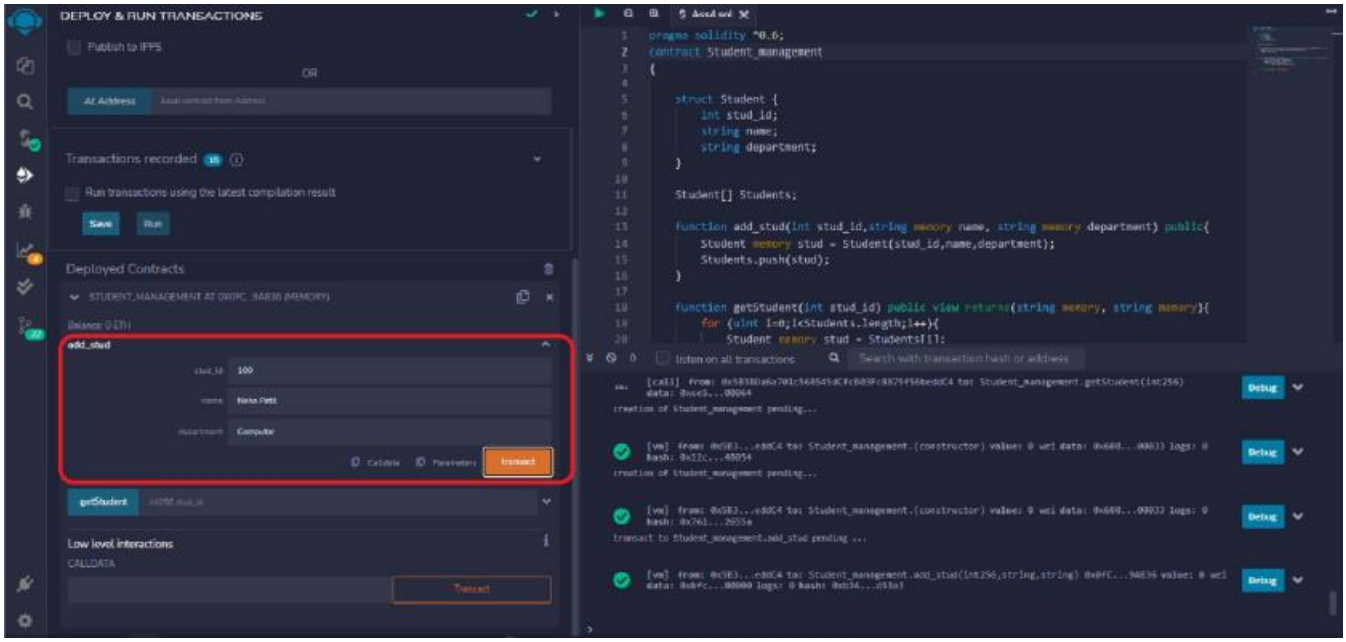
Assignment No.: 4

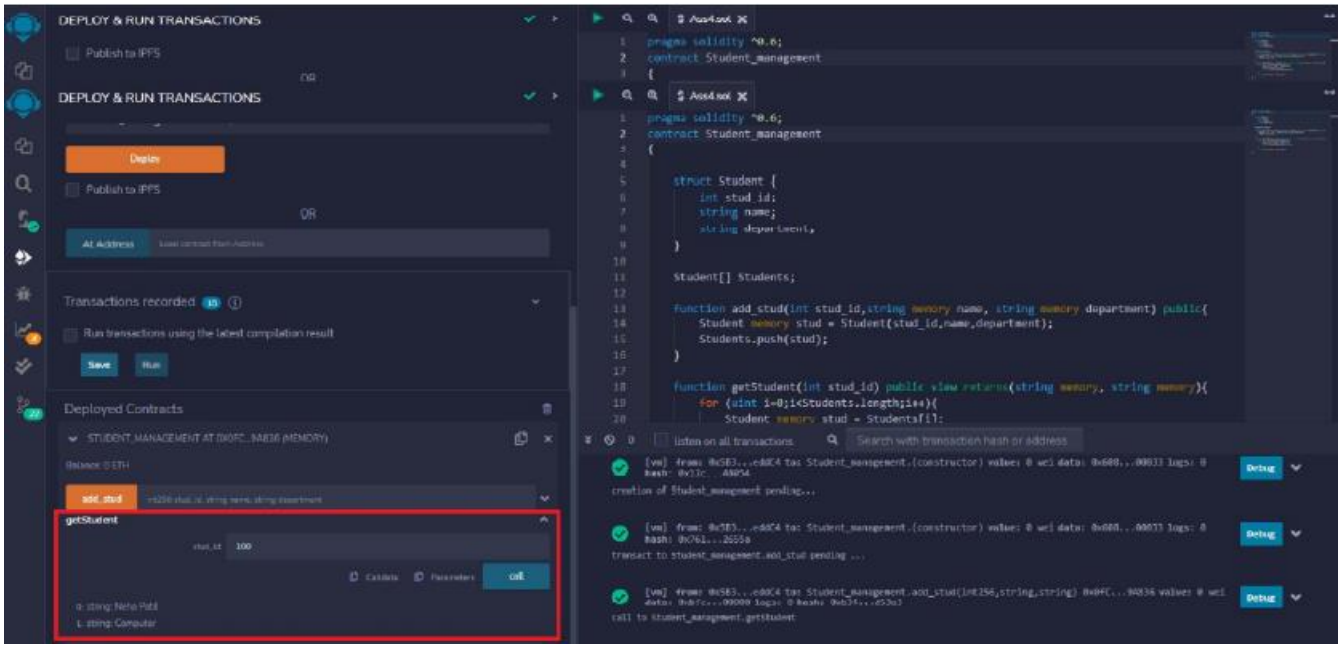
Title: Write a program in solidity to create Student data. Use the following constructs: • Structures • Arrays • Fallback

Code:

```
pragma solidity ^0.6;
contract Student_management
{
struct Student {
 intstud_id
 string name;
 string department;
 }
 Student[] Students;
 function add_stud(intstud_id,string memory name, string memory department) public{
 Student memory stud = Student(stud_id,name,department);
Students.push(stud);
 }
 function getStudent(intstud_id) public view returns(string memory, string memory){
 for (uinti=0;i<Students.length;i++){
 Student memory stud = Students[i];
 if(stud.stud_id==stud_id){
 return(stud.name,stud.department);
 }
 }
 return("Not Found", "Not Found");
 }
}
```

Output:


DEPLOY & RUN TRANSACTIONS
Publish to IPFS
OR
At Address
Transactions recorded
Run transactions using the latest compilation result
Save
Run
Deployed Contracts
STUDENT_MANAGEMENT AT 0X0FC...9A836 (MEMORY)
Balance: 0 ETH
add_stud
stud_id 100
name Neha Patil
department Computer
Calldata
Parameters
transact
getStudent
Low level interactions
CALLDATA
Transact
Ass4.sol
pragma solidity ^0.6;
contract Student_management
{
struct Student {
int stud_id;
string name;
string department;
}
Student[] Students;
function add_stud(int stud_id,string memory name, string memory department) public{
Student memory stud = Student(stud_id,name,department);
Students.push(stud);
}
function getStudent(int stud_id) public view returns(string memory, string memory){
for (uint i=0;i<Students.length;i++){
Student memory stud = Students[i];
listen on all transactions
Search with transaction hash or address
Debug


DEPLOY & RUN TRANSACTIONS
Publish to IPFS
OR
DEPLOY & RUN TRANSACTIONS
Deploy
Publish to IPFS
OR
At Address
Transactions recorded
Run transactions using the latest compilation result
Save
Run
Deployed Contracts
STUDENT_MANAGEMENT AT 0X0FC...9A836 (MEMORY)
Balance: 0 ETH
add_stud
getStudent
stud_id 100
Calldata
Parameters
call
0: string: Neha Patil
1: string: Computer
Ass4.sol
pragma solidity ^0.6;
contract Student_management
{
struct Student {
int stud_id;
string name;
string department;
}
Student[] Students;
function add_stud(int stud_id,string memory name, string memory department) public{
Student memory stud = Student(stud_id,name,department);
Students.push(stud);
}
function getStudent(int stud_id) public view returns(string memory, string memory){
for (uint i=0;i<Students.length;i++){
Student memory stud = Students[i];
listen on all transactions
Search with transaction hash or address
Debug