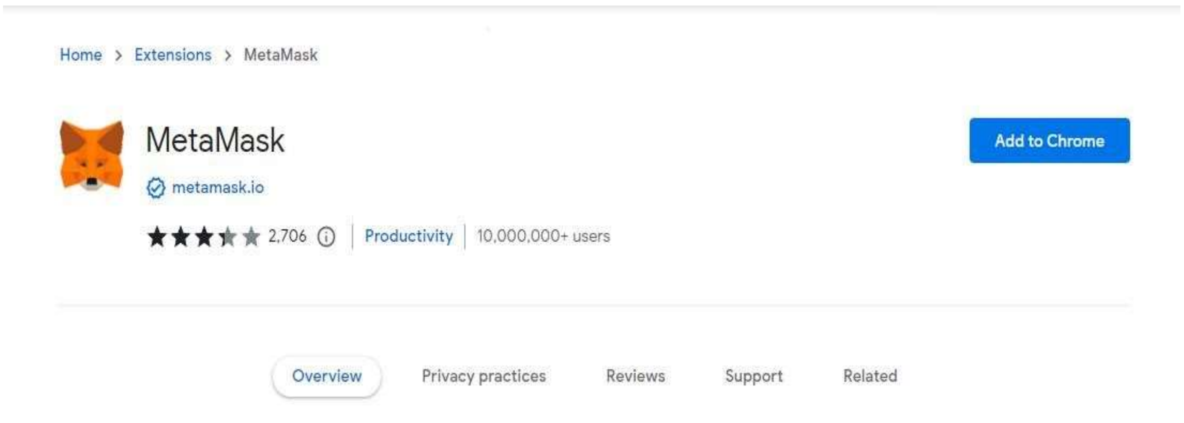
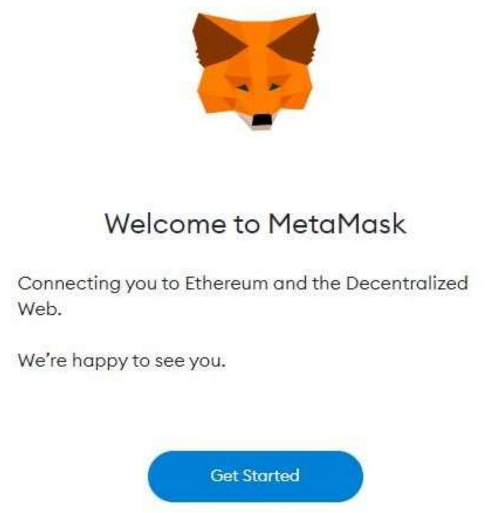
Assignment 1


Home > Extensions > MetaMask

MetaMask

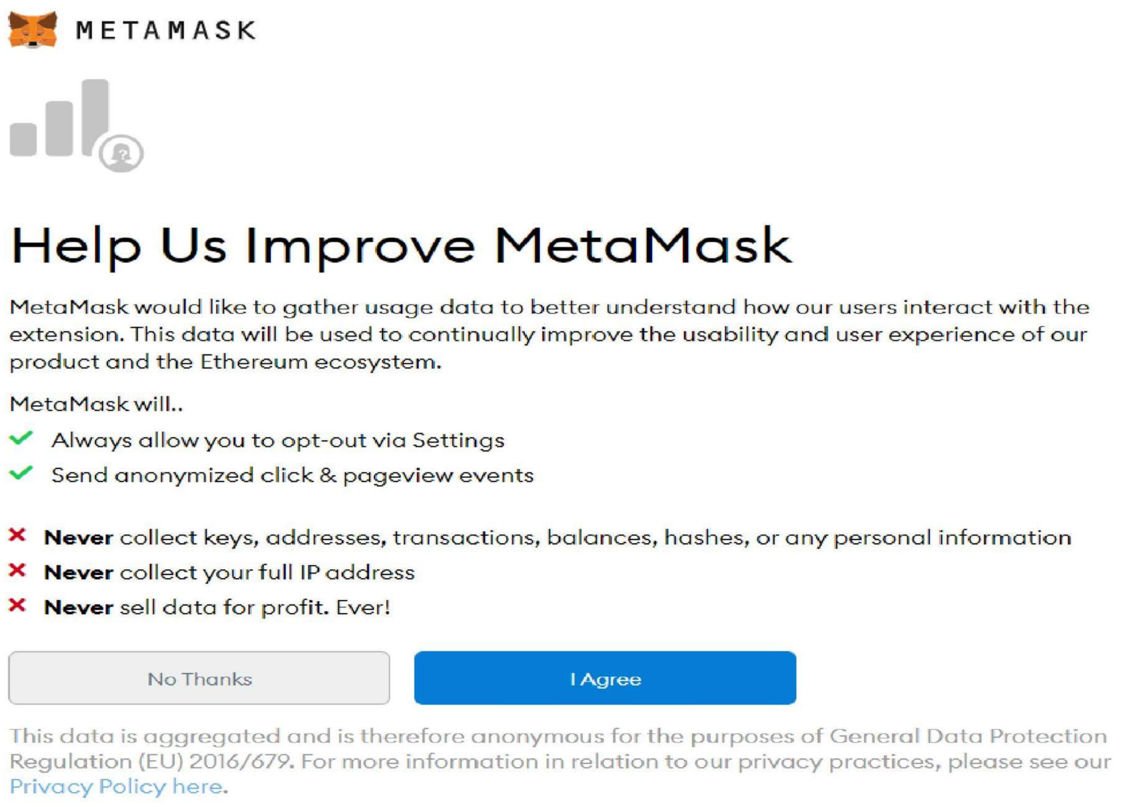
metamask.io

2,706 | Productivity | 10,000,000+ users


Welcome to MetaMask

Connecting you to Ethereum and the Decentralized Web.

We're happy to see you.



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.

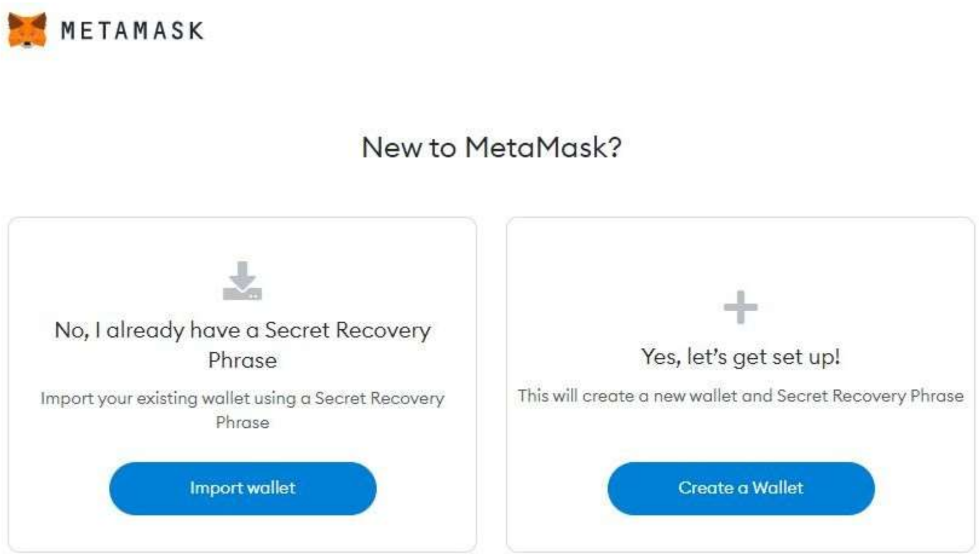
MetaMask will..

- ✓ 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!

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.

Assignment 2


METAMASK

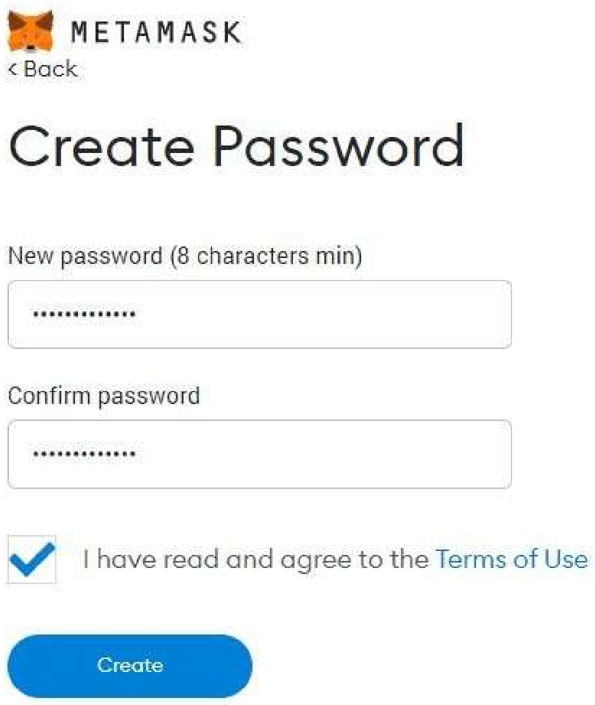
New to MetaMask?

No, I already have a Secret Recovery Phrase

Import your existing wallet using a Secret Recovery Phrase

Yes, let's get set up!

This will create a new wallet and Secret Recovery Phrase

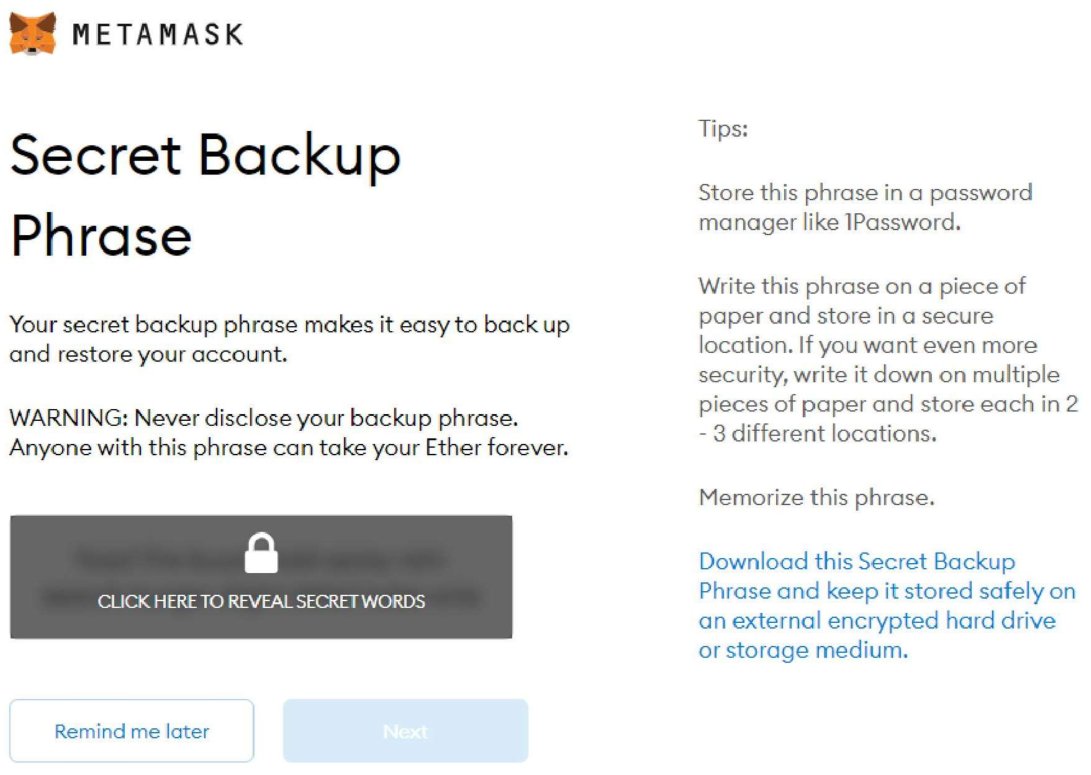

METAMASK

Create Password

New password (8 characters min): ••••••••••••••

Confirm password: ••••••••••••••

✓ I have read and agree to the Terms of Use



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.

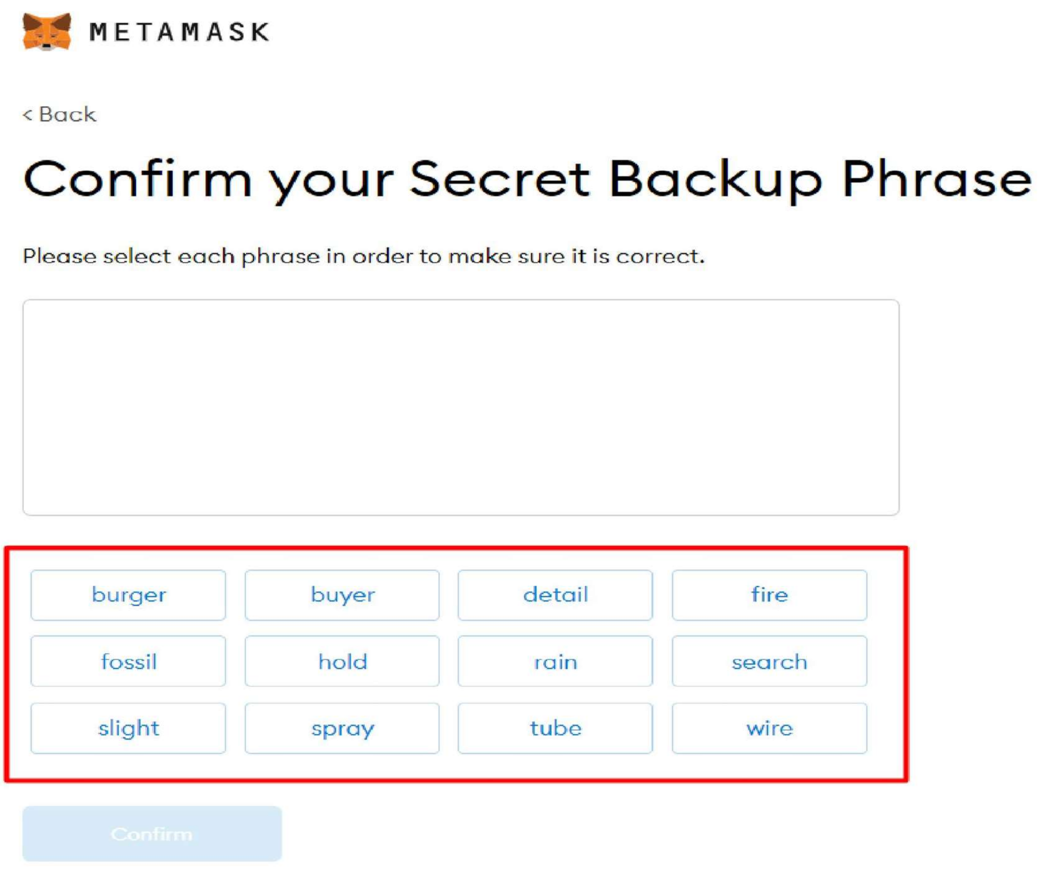
Tips:

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.



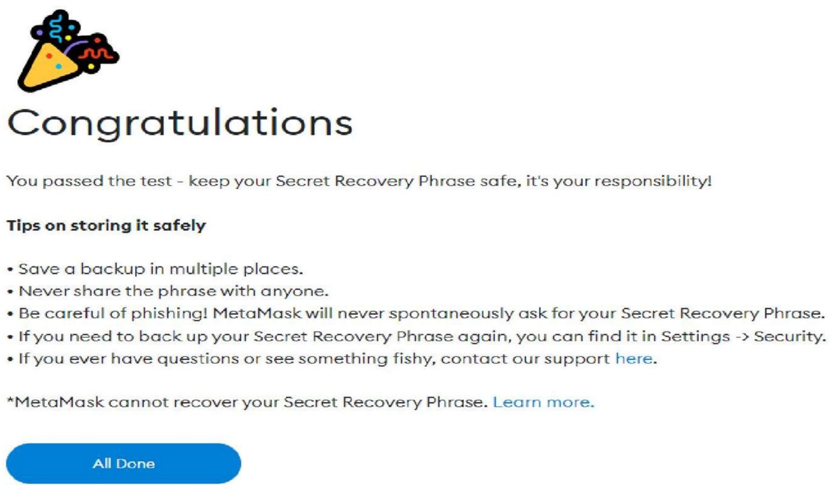
METAMASK

< 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



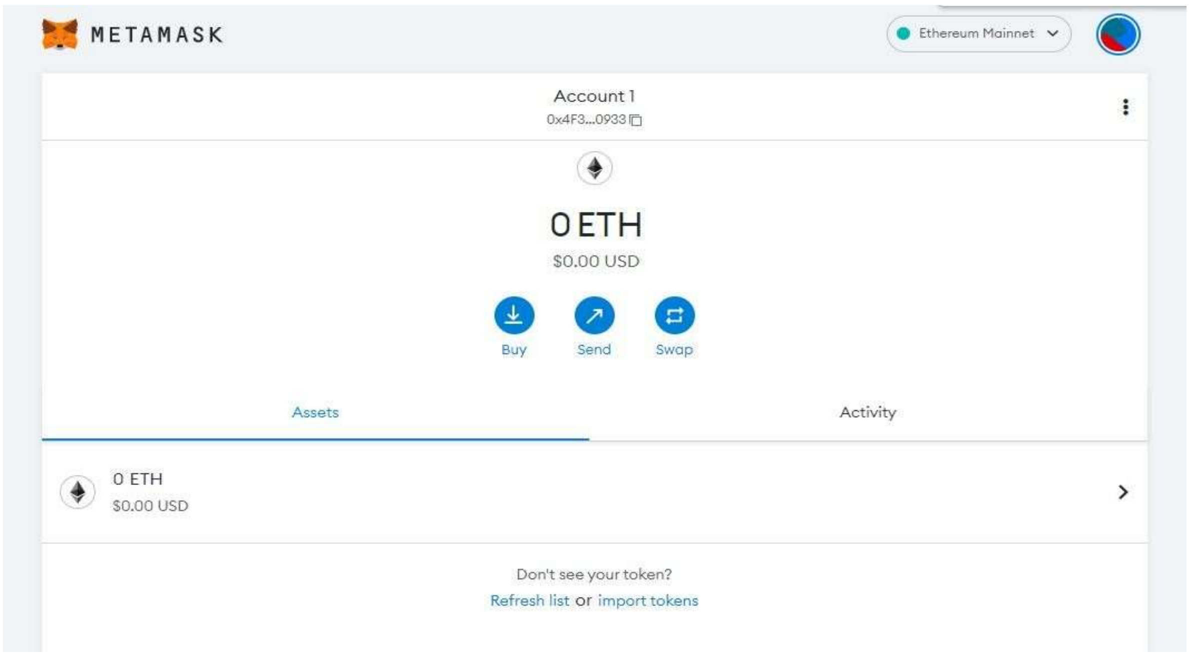
Congratulations

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.



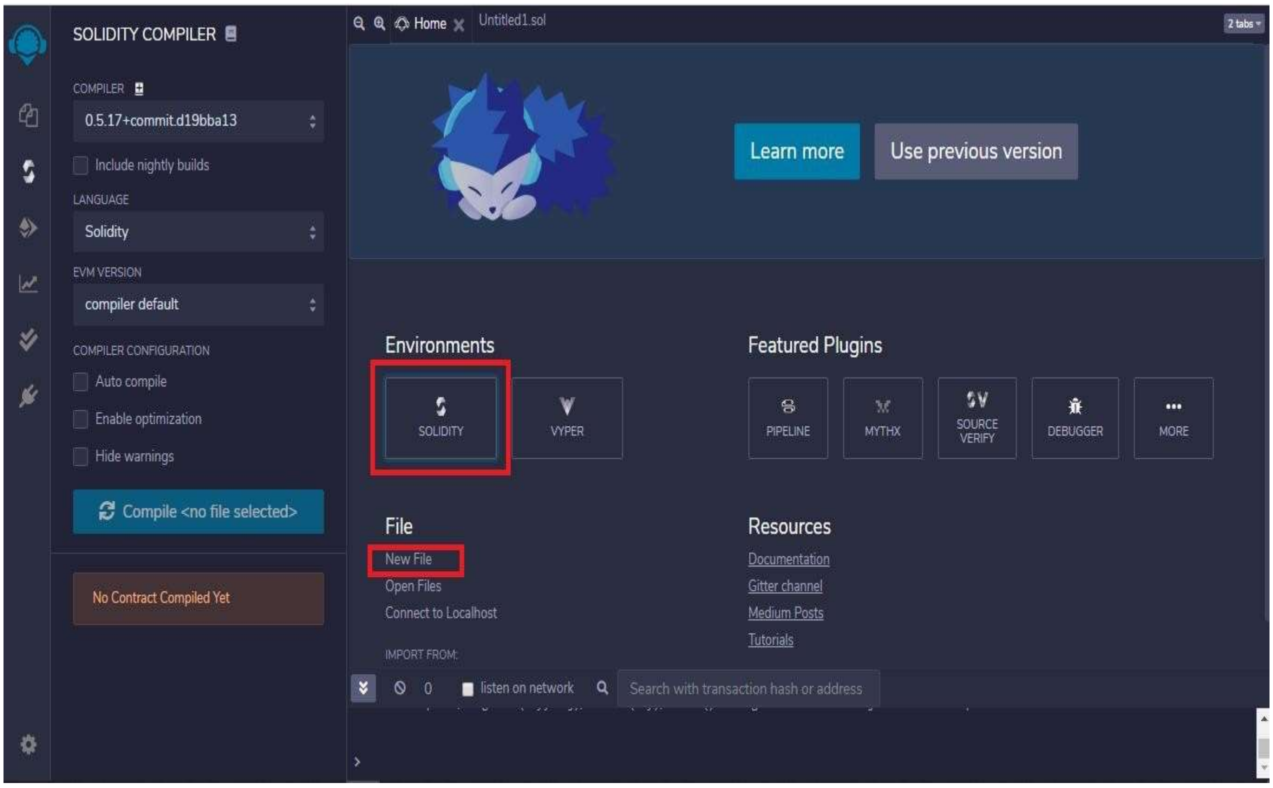
Account 1
0x4F3...0933

0 ETH
$0.00 USD

0 ETH
$0.00 USD

Don't see your token?
Refresh list or import tokens

Assignment 3


SOLIDITY COMPILER

COMPILER: 0.5.17+commit.d19bba13

Include nightly builds

LANGUAGE: Solidity

EVM VERSION: compiler default

COMPILER CONFIGURATION

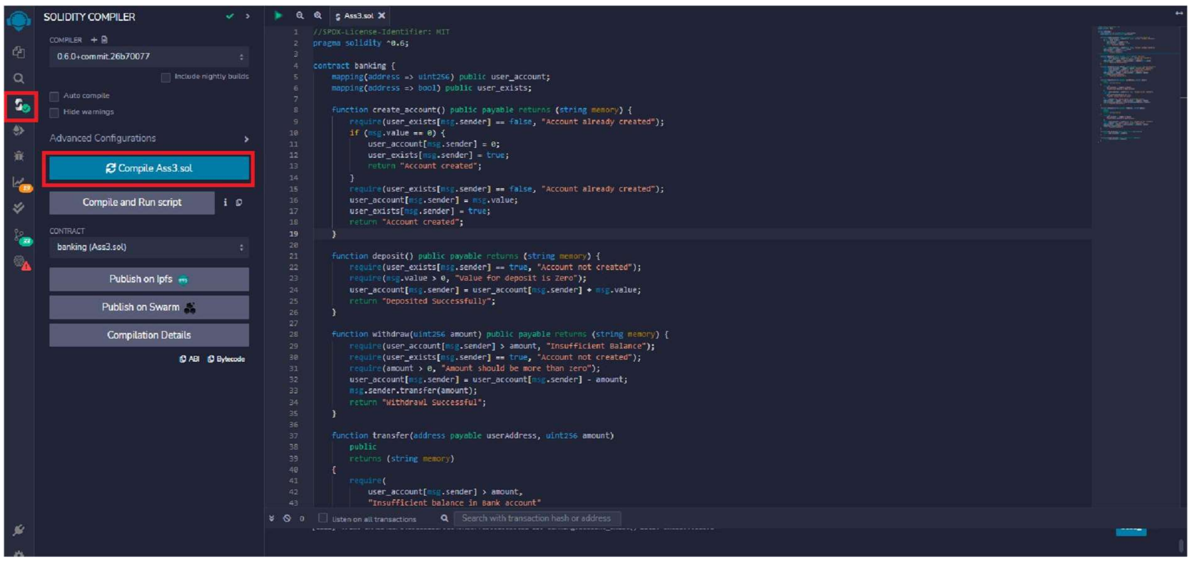
Auto compile

Enable optimization

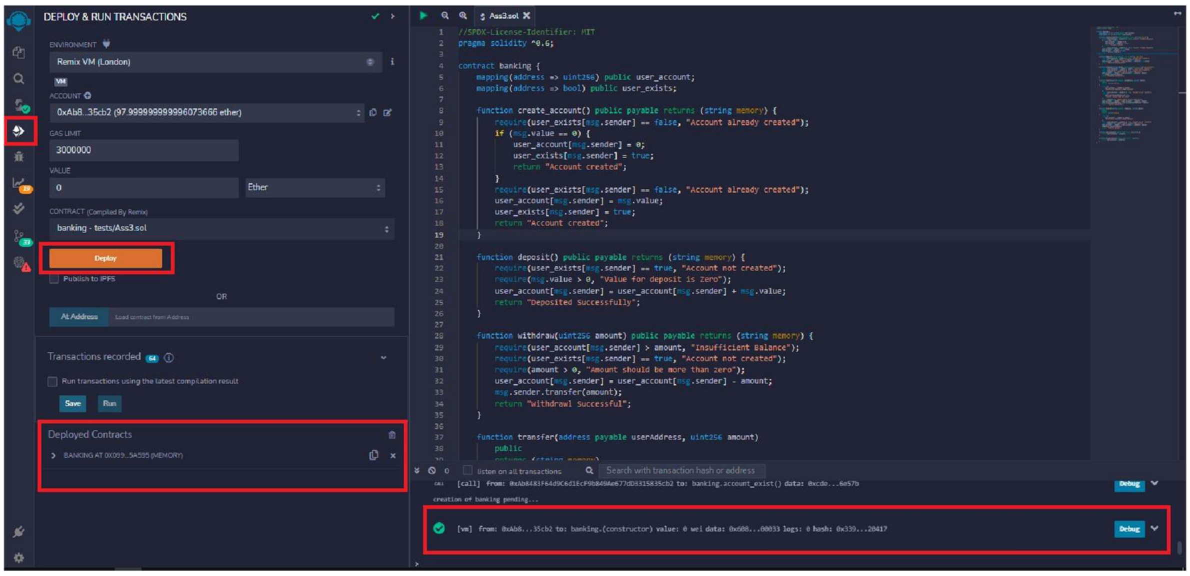
Hide warnings

Compile <no file selected>

No Contract Compiled Yet



```
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;

contract banking {
    mapping(address => uint256) 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(uint256 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, uint256 amount)
        public
        returns (string memory)
    {
        require(
            user_account[msg.sender] > amount,
            "Insufficient balance in Bank account"
```


DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (London)
VM
ACCOUNT
0xAb8...35cb2 (97.999999999996073666 ether)
GAS LIMIT
3000000
VALUE
0
Ether
CONTRACT (Compiled By Remix)
banking - tests/Ass3.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Transactions recorded
Run transactions using the latest compilation result
Save
Run
Deployed Contracts
BANKING AT 0X099...5A555 (MEMORY)
Ass3.sol
listen on all transactions
Search with transaction hash or address
Debug

```
[call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: banking.account_exist() data: 0xcde...6e57b
creation of banking pending...
[vm] from: 0xAb8...35cb2 to: banking.(constructor) value: 0 wei data: 0x608...00033 logs: 0 hash: 0x339...20417
```

Code

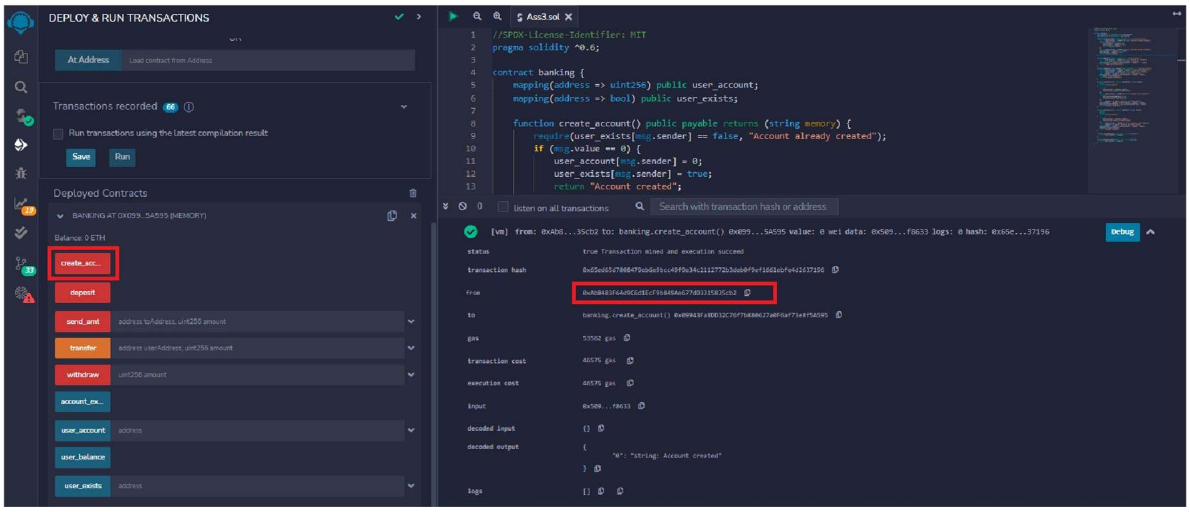
//SPDX-License-Identifier: MIT

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];
    }
}
```

Sample Output

- Create account

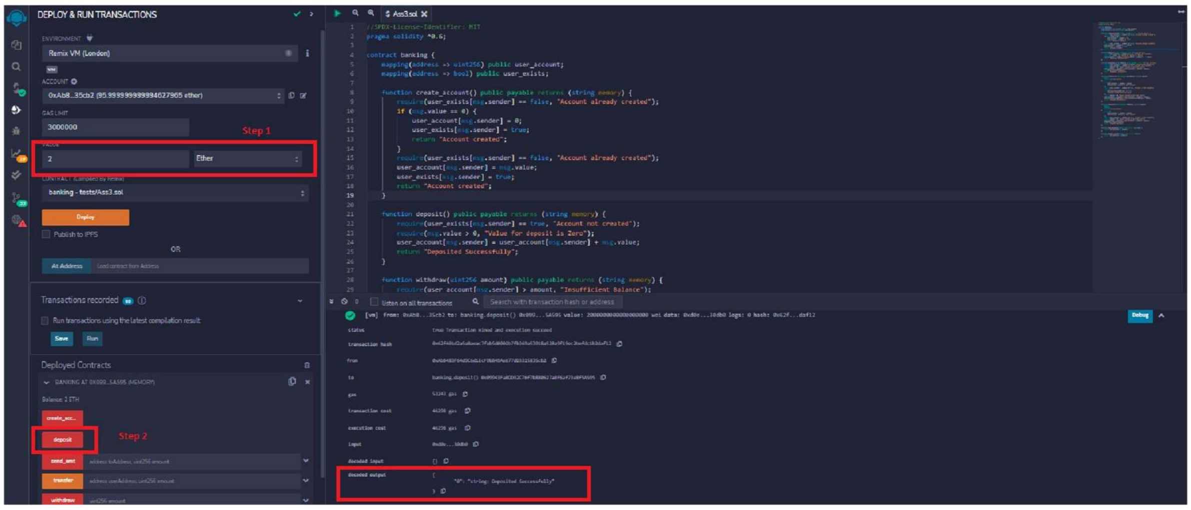


DEPLOY & RUN TRANSACTIONS
At Address
Load contract from Address
Transactions recorded 66
Run transactions using the latest compilation result
Save
Run
Deployed Contracts
BANKING AT 0X099...5A595 (MEMORY)
Balance: 0 ETH
create_acc...
deposit
send_amt
address toAddress, uint256 amount
transfer
address userAddress, uint256 amount
withdraw
uint256 amount
account_ex...
user_account
address
user_balance
user_exists
address
Ass3.sol
listen on all transactions
Search with transaction hash or address
Debug

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;

contract banking {
    mapping(address => uint256) 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";
```

```
[vm] from: 0xAb8...35cb2 to: banking.create_account() 0x099...5A595 value: 0 wei data: 0x509...f8633 logs: 0 hash: 0x65e...37196
status              true Transaction mined and execution succeed
transaction hash    0x65ed65d7808479eb6e9bcc49f9e34c2112772b3deb0f9ef1661ebfe4d2637196
from                0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to                  banking.create_account() 0x09943Fa8DD32C76f7b880627a0F6af73e8f5A595
gas                 53562 gas
transaction cost    46575 gas
execution cost      46575 gas
input               0x509...f8633
decoded input       {}
decoded output      {
                        "0": "string: Account created"
                    }
logs                []
```

- Deposit Amount



ENVIRONMENT: Remix VM (London)
ACCOUNT: 0xAb8...35cb2 (95.999999999994627965 ether)
GAS LIMIT: 3000000
VALUE: 2 Ether
CONTRACT: banking - tests/Ass3.sol

Run transactions using the latest compilation result

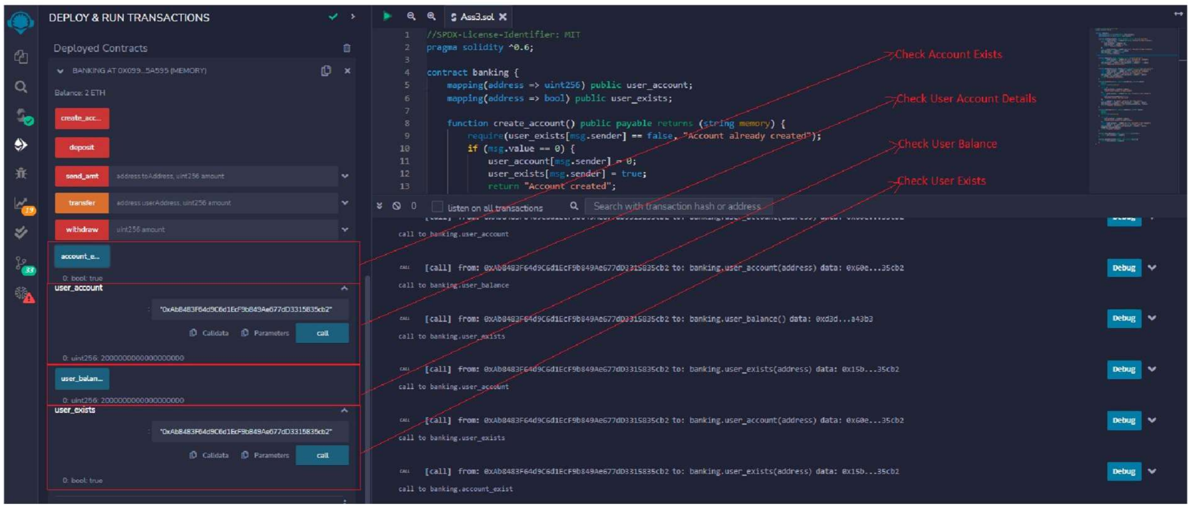
BANKING AT 0X099...5A595 (MEMORY)
Balance: 2 ETH

send_amt: address toAddress, uint256 amount
transfer: address userAddress, uint256 amount
withdraw: uint256 amount

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;

contract banking {
    mapping(address => uint256) 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(uint256 amount) public payable returns (string memory) {
        require(user_account[msg.sender] > amount, "Insufficient Balance");
```

```
listen on all transactions    Search with transaction hash or address
[vm] from: 0xAb8...35cb2 to: banking.deposit() 0x099...5A595 value: 2000000000000000000 wei data: 0xd0e...30db0 logs: 0 hash: 0x62f...dbf12
status              true Transaction mined and execution succeed
transaction hash    [illegible]
from                [illegible]
to                  banking.deposit() [illegible]
gas                 53243 gas
transaction cost    46298 gas
execution cost      46298 gas
input               0xd0e...30db0
decoded input       {}
decoded output      {
                        "0": "string: Deposited Successfully"
                    }
```

- Check User Exists



BANKING AT 0X099...5A595 (MEMORY)
Balance: 2 ETH

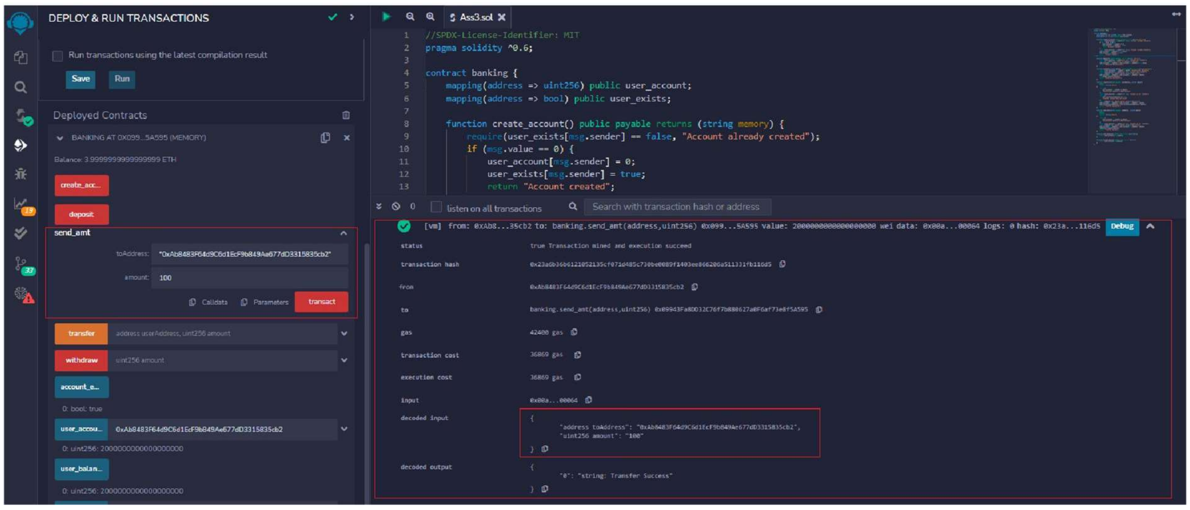
send_amt: address toAddress, uint256 amount
transfer: address userAddress, uint256 amount
withdraw: uint256 amount

account_e... → 0: bool: true
user_account: "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2" → 0: uint256: 2000000000000000000
user_balan... → 0: uint256: 2000000000000000000
user_exists: "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2" → 0: bool: true

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;

contract banking {
    mapping(address => uint256) 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";
```

```
call to banking.user_account

CALL [call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: banking.user_account(address) data: 0x60e...35cb2
call to banking.user_balance

CALL [call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: banking.user_balance() data: 0xd3d...a43b3
call to banking.user_exists

CALL [call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: banking.user_exists(address) data: 0x15b...35cb2
call to banking.user_account

CALL [call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: banking.user_account(address) data: 0x60e...35cb2
call to banking.user_exists

CALL [call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: banking.user_exists(address) data: 0x15b...35cb2
call to banking.account_exist
```

- Send Amount



Run transactions using the latest compilation result

BANKING AT 0X099...5A595 (MEMORY)

Balance: 3.9999999999999999 ETH

send_amt

toAddress: "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"

amount: 100

address userAddress, uint256 amount

uint256 amount

0: bool: true

0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

0: uint256: 2000000000000000000

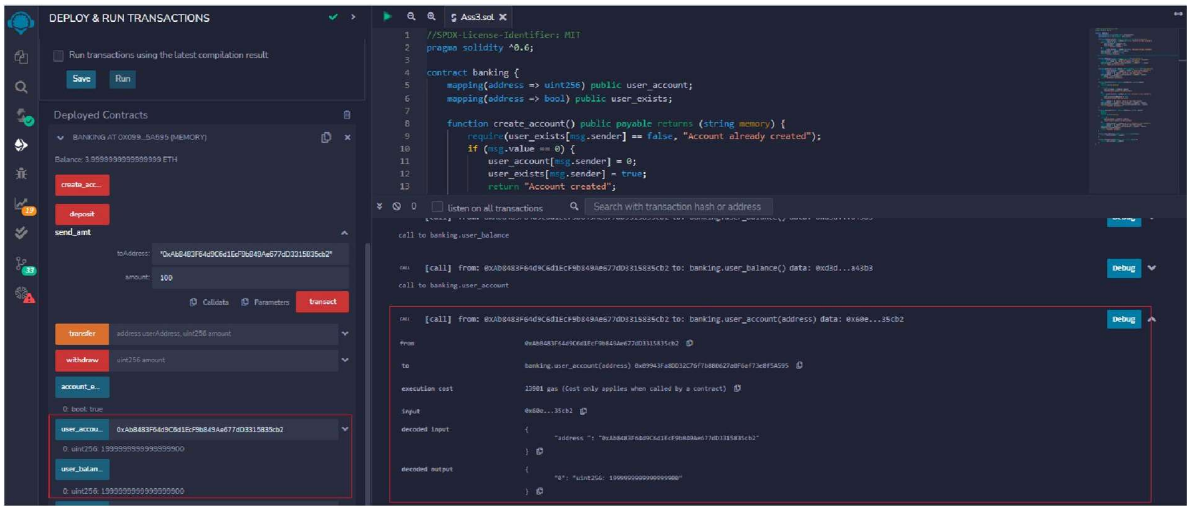
0: uint256: 2000000000000000000

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;

contract banking {
    mapping(address => uint256) 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";
```

listen on all transactions   Search with transaction hash or address

```
[vm] from: 0xAb8...35cb2 to: banking.send_amt(address,uint256) 0x099...5A595 value: 2000000000000000000 wei data: 0x00a...00064 logs: 0 hash: 0x23a...116d5

status                 true Transaction mined and execution succeed
transaction hash       0x23a6b36b6121052135cf071d485c730be0089f1403ee866206a511331fb116d5
from                   0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to                     banking.send_amt(address,uint256) 0x09943Fa8DD32C76f7b880627a0F6af73e8f5A595
gas                    42400 gas
transaction cost       36869 gas
execution cost         36869 gas
input                  0x00a...00064
decoded input          {
                               "address toAddress": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                               "uint256 amount": "100"
                       }
decoded output         {
                               "0": "string: Transfer Success"
                       }
```

- Check User Account Balance



Run transactions using the latest compilation result

BANKING AT 0X099...5A595 (MEMORY)

Balance: 3.9999999999999999 ETH

send_amt

toAddress: "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"

amount: 100

address userAddress, uint256 amount

uint256 amount

0: bool: true

0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

0: uint256: 1999999999999999900

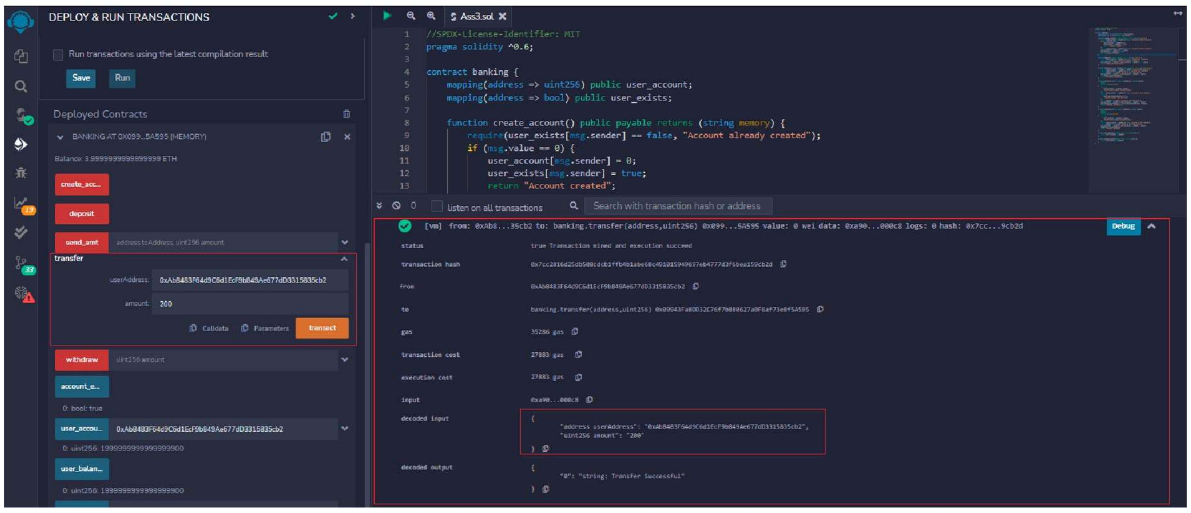
0: uint256: 1999999999999999900

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;

contract banking {
    mapping(address => uint256) 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";
```

listen on all transactions   Search with transaction hash or address

```
call to banking.user_balance

CALL  [call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: banking.user_balance() data: 0xd3d...a43b3
call to banking.user_account

CALL  [call] from: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2 to: banking.user_account(address) data: 0x60e...35cb2
from                   0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to                     banking.user_account(address) 0x09943Fa8DD32C76f7b880627a0F6af73e8f5A595
execution cost         23901 gas (Cost only applies when called by a contract)
input                  0x60e...35cb2
decoded input          {
                               "address ": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2"
                       }
decoded output         {
                               "0": "uint256: 1999999999999999900"
                       }
```

- Transfer Amount and Check User Account Balance

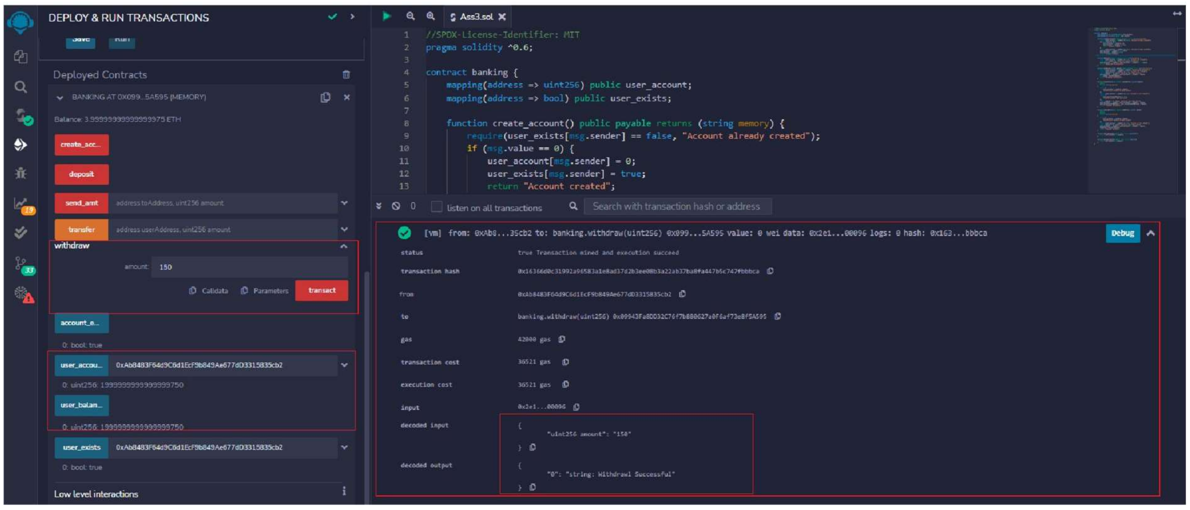


```
Run transactions using the latest compilation result

Deployed Contracts
BANKING AT 0X099...5A595 (MEMORY)
Balance: 3.9999999999999999 ETH
send_amt      address toAddress, uint256 amount
transfer
    userAddress: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
    amount: 200
withdraw      uint256 amount
account_e...
0: bool: true
user_accou... 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
0: uint256: 1999999999999999900
user_balan...
0: uint256: 1999999999999999900
```

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;

contract banking {
    mapping(address => uint256) 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";
```

```
[vm] from: 0xAb8...35cb2 to: banking.transfer(address,uint256) 0x099...5A595 value: 0 wei data: 0xa90...000c8 logs: 0 hash: 0x7cc...9cb2d
status              true Transaction mined and execution succeed
transaction hash    0x7cc2816d25db508cdcb1ffb4b1abe68c491015949697eb4777d3f6bea159cb2d
from                0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to                  banking.transfer(address,uint256) 0x09943Fa8DD32C76f7b880627a0F6af73e8f5A595
gas                 35286 gas
transaction cost    27883 gas
execution cost      27883 gas
input               0xa90...000c8
decoded input       {
                        "address userAddress": "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2",
                        "uint256 amount": "200"
                    }
decoded output      {
                        "0": "string: Transfer Successful"
                    }
```

- Withdraw Amount and Check User Account Balance

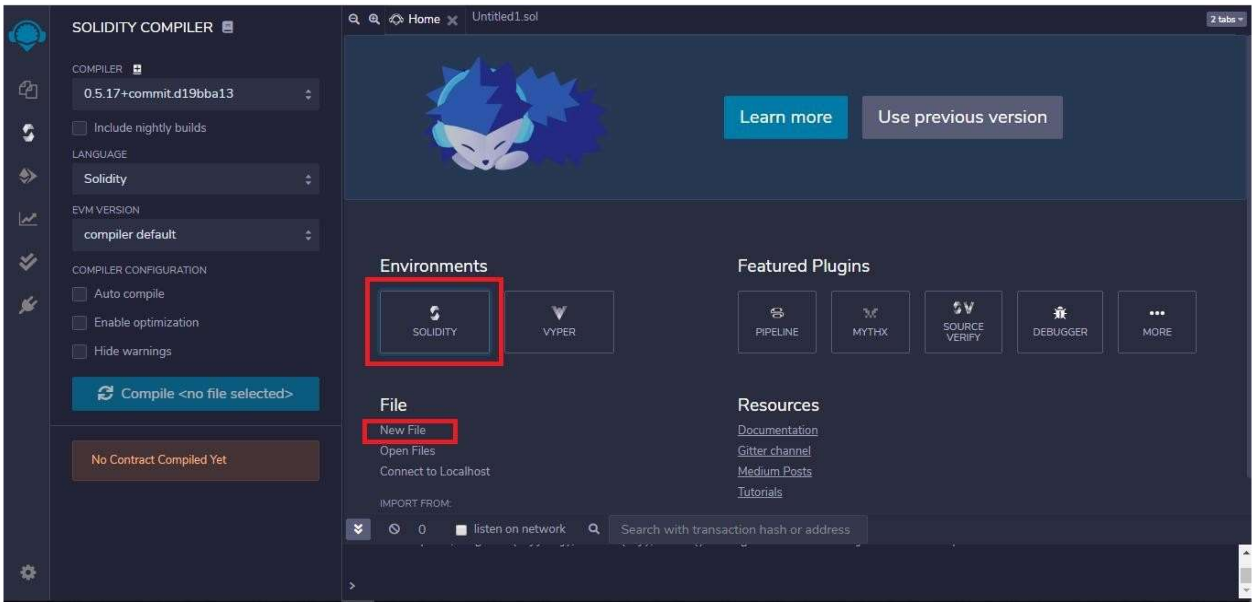


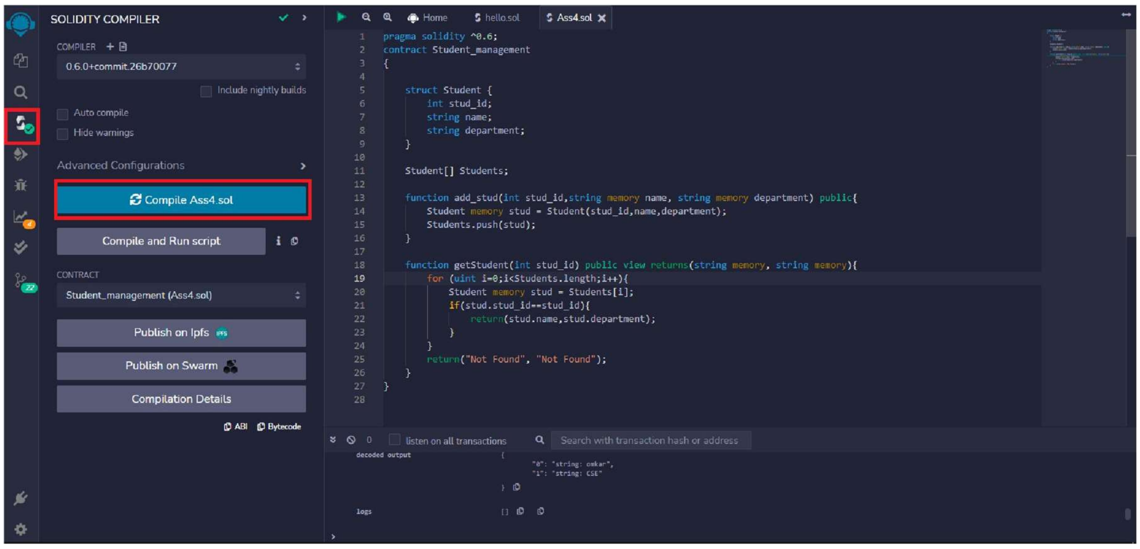
```
Deployed Contracts
BANKING AT 0X099...5A595 (MEMORY)
Balance: 3.9999999999999999975 ETH
send_amt      address toAddress, uint256 amount
transfer      address userAddress, uint256 amount
withdraw
    amount: 150
account_e...
0: bool: true
user_accou... 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
0: uint256: 1999999999999999750
user_balan...
0: uint256: 1999999999999999750
user_exists   0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
0: bool: true
```

```
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;

contract banking {
    mapping(address => uint256) 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";
```

```
[vm] from: 0xAb8...35cb2 to: banking.withdraw(uint256) 0x099...5A595 value: 0 wei data: 0x2e1...00096 logs: 0 hash: 0x163...bbbca
status              true Transaction mined and execution succeed
transaction hash    0x16366d0c31992a96583a1e8ad37d2b3ee08b3a22ab37ba8fa447b5c747fbbbca
from                0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to                  banking.withdraw(uint256) 0x09943Fa8DD32C76f7b880627a0f6af73e8f5A595
gas                 42000 gas
transaction cost    36521 gas
execution cost      36521 gas
input               0x2e1...00096
decoded input       {
                        "uint256 amount": "150"
                    }
decoded output      {
                        "0": "string: Withdrawl Successful"
                    }
```

Assignment 4

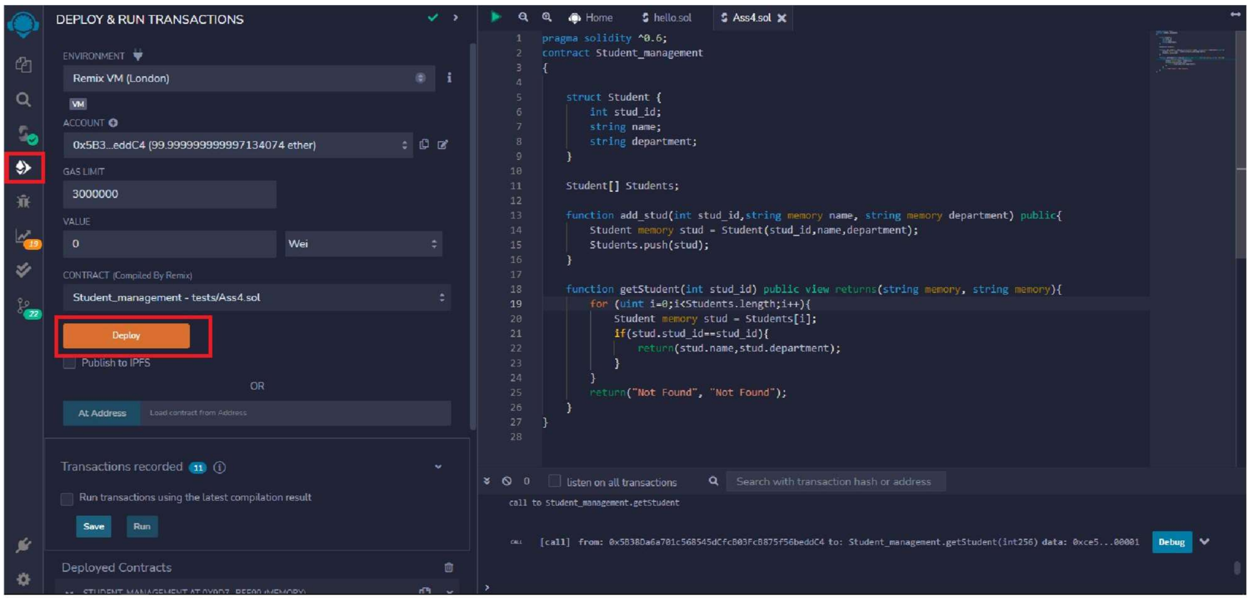




```
pragma solidity ^0.6;
contract Student_management
{

    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];
            if(stud.stud_id==stud_id){
                return(stud.name,stud.department);
            }
        }
        return("Not Found", "Not Found");
    }
}
```

```
decoded output          {
                                "0": "string: omkar",
                                "1": "string: CSE"
                        }

logs                    []
```

.


DEPLOY & RUN TRANSACTIONS
Home
hello.sol
Ass4.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Save
Run
Debug
Deployed Contracts
STUDENT_MANAGEMENT AT 0X9D7...BEF00 (MEMORY)

ENVIRONMENT: Remix VM (London)

ACCOUNT: 0x5B3...eddC4 (99.999999999997134074 ether)

GAS LIMIT: 3000000

VALUE: 0 Wei

CONTRACT (Compiled By Remix): Student_management - tests/Ass4.sol

Transactions recorded 11

Run transactions using the latest compilation result

```
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];
            if(stud.stud_id==stud_id){
                return(stud.name,stud.department);
            }
        }
        return("Not Found", "Not Found");
    }
}
```

listen on all transactions

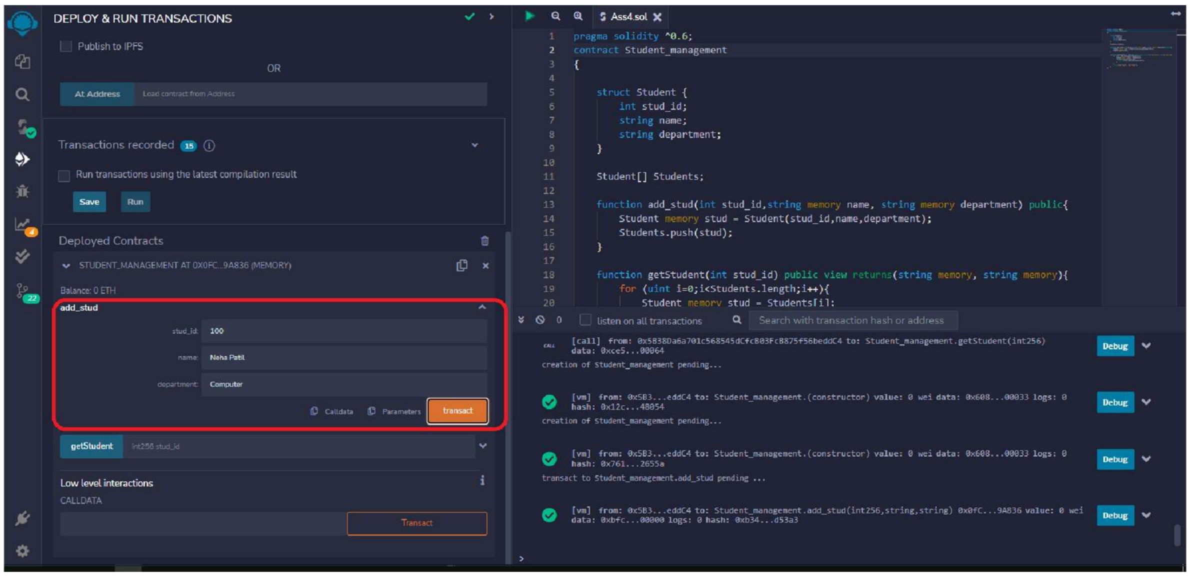
Search with transaction hash or address

```
call to Student_management.getStudent

[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Student_management.getStudent(int256) data: 0xce5...00001
```

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");
   }
}
```

Sample Output



DEPLOY & RUN TRANSACTIONS

Publish to IPFS

OR

At Address | Load contract from Address

Transactions recorded 15

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 | int256 stud_id

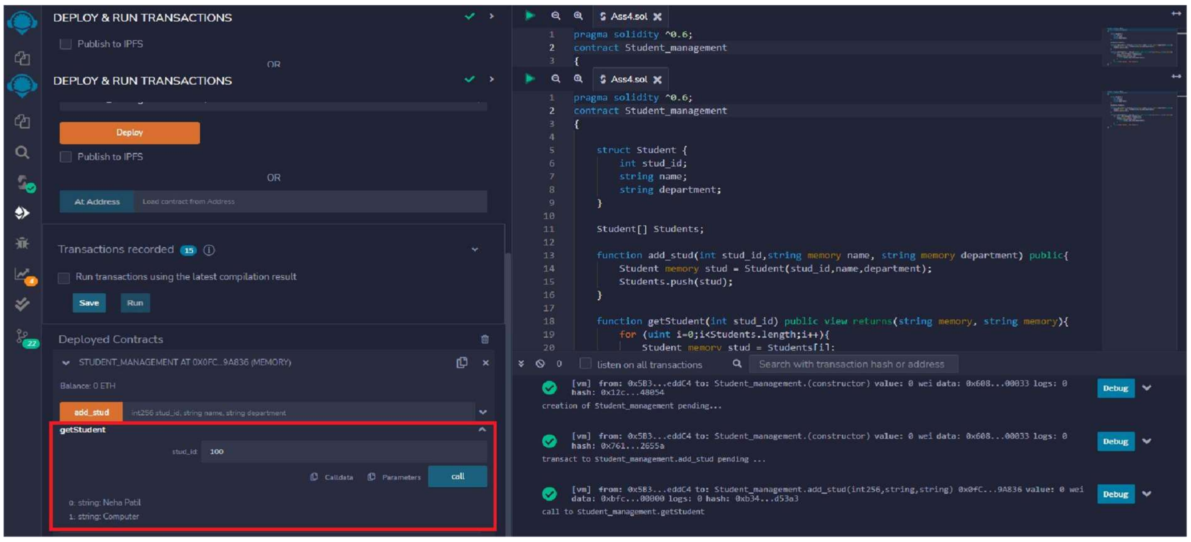
Low level interactions

CALLDATA

Transact

```
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];
```

```
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: Student_management.getStudent(int256)
data: 0xce5...00064
creation of Student_management pending...

[vm] from: 0x5B3...eddC4 to: Student_management.(constructor) value: 0 wei data: 0x608...00033 logs: 0
hash: 0x12c...48054
creation of Student_management pending...

[vm] from: 0x5B3...eddC4 to: Student_management.(constructor) value: 0 wei data: 0x608...00033 logs: 0
hash: 0x761...2655a
transact to Student_management.add_stud pending ...

[vm] from: 0x5B3...eddC4 to: Student_management.add_stud(int256,string,string) 0x0fC...9A836 value: 0 wei
data: 0xbfc...00000 logs: 0 hash: 0xb34...d53a3
```



DEPLOY & RUN TRANSACTIONS

Deploy

Publish to IPFS

OR

At Address | Load contract from Address

Transactions recorded 15

Run transactions using the latest compilation result

Save | Run

Deployed Contracts

STUDENT_MANAGEMENT AT 0X0FC...9A836 (MEMORY)

Balance: 0 ETH

add_stud | int256 stud_id, string name, string department

getStudent

stud_id: 100

Calldata | Parameters | call

0: string: Neha Patil

1: string: Computer

```
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];
```

```
[vm] from: 0x5B3...eddC4 to: Student_management.(constructor) value: 0 wei data: 0x608...00033 logs: 0
hash: 0x12c...48054
creation of Student_management pending...

[vm] from: 0x5B3...eddC4 to: Student_management.(constructor) value: 0 wei data: 0x608...00033 logs: 0
hash: 0x761...2655a
transact to Student_management.add_stud pending ...

[vm] from: 0x5B3...eddC4 to: Student_management.add_stud(int256,string,string) 0x0fC...9A836 value: 0 wei
data: 0xbfc...00000 logs: 0 hash: 0xb34...d53a3
call to Student_management.getStudent
```