LABORATORY MANUAL

FOR

Laboratory Practice III(ML) (2019) (BE SEM I)

Department of Computer Engineering

International Institute of Information Technology

Hinjawadi,Pune - 411 057

www.isquareit.edu.in

Work Load	**Exam Schemes**		
Practical	**Term Work**	**Practical**	**Oral**
04 Hours/Week	50	50	--
List Of Assignments			
Sr. No.	**Title of Assignment**		**Time Span (No. of weeks)**

1)	Predict the price of the Uber ride from a given pickup point to the agreed drop-off location. Perform following tasks: 1. Pre-process the dataset. 2. Identify outliers. 3. Check the correlation. 4. Implement linear regression and random forest regression models. 5. Evaluate the models and compare their respective scores like R2, RMSE, etc. Dataset link: https://www.kaggle.com/datasets/yasserh/uber-fares-dataset	02
2)	Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method.Dataset link: https://www.kaggle.com/datasets/kyanyoga/sample-sales-data	01
3)	Implement K-Nearest Neighbors algorithm on diabetes.csv dataset. Compute confusion matrix, accuracy, error rate, precision and recall on the given dataset. Dataset link: https://www.kaggle.com/datasets/abdallamahgoub/diabetes	01
4)	Given a bank customer, build a neural network-based classifier that can determine whether they will leave or not in the next 6 months. Dataset Description: The case study is from an open-source dataset from Kaggle. The dataset contains 10,000 sample points with 14 distinct features such as CustomerId, Credit Score, Geography, Gender, Age, Tenure, Balance, etc. Link to the Kaggle project: https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling Perform following steps: 1. Read the dataset. 2. Distinguish the feature and target set and divide the data set into training and test sets. 3. Normalize the train and test data. 4. Initialize and build the model. Identify the points of improvement and implement the same. 5. Print the accuracy score and confusion matrix (5 points).	01
5)	Classify the email using the binary classification method. Email Spam detection has two states: a) Normal State – Not Spam, b) Abnormal State – Spam. Use K-Nearest Neighbors and Support Vector Machine for classification. Analyze their performance. Dataset link: The emails.csv dataset on the Kaggle ttps://www.kaggle.com/datasets/balaka18/email-spam-classification-dataset-csv	02
6)	Mini Project Mini Project - Use the following dataset to analyze ups and downs in the market and predict future stock price returns based on Indian Market data from 2000 to 2020. Dataset Link: ttps://www.kaggle.com/datasets/sagara9595/stock-data	02
7)	Installation of MetaMask and study spending Ether per transaction.	01

8)	Create your own wallet using Metamask for crypto transactions.	01
9)	Write a smart contract on a test network, for Bank account of a customer for following operations: □ Deposit money □ Withdraw Money □ Show balance	02
10)	Write a program in solidity to create Student data. Use the following constructs: □ Structures □ Arrays □ Fallback Deploy this as smart contract on Ethereum and Observe the transaction fee and Gas values.	02
11)	Write a survey report on types of Blockchains and its real time use cases	02
12)	Mini Project - Develop a Blockchain based application for health related medical records	

Reference Sites:

https://builtin.com/machine-learning/how-to-preprocess-data-python

https://levelup.gitconnected.com/random-forest-regression

Virtual Laboratory:

1. http://cse01-iiith.vlabs.ac.in/
2. http://vlabs.iitb.ac.in/vlabs-dev/labs/blockchain/labs/index.php
3. http://vlabs.iitb.ac.in/vlabs-dev/labs/machine_learning/labs/index.php

ASSIGNMENT NO. 3

TITLE: Write a smart contract on a test network, for Bank account of a customer**.**

Problem Statement: Write a smart contract on a test network, for Bank account of a customer for following operations: Deposit money Withdraw Money Show balance

Objective:

- To learn Solidity.
- To create Smart Contract using Solidity
- Write smart contract on a test network.

Theory:

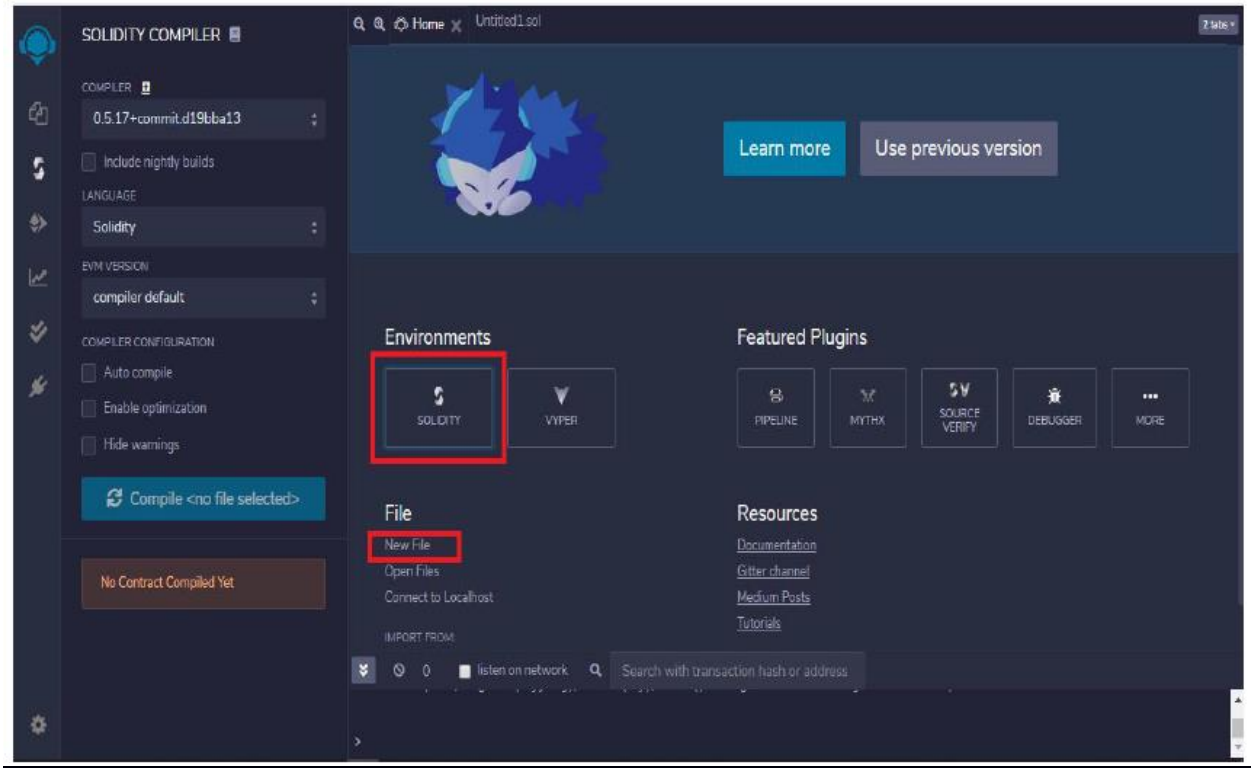
First, we need to understand the differences between a paper contract and a smart contract and the reason why smart contracts become increasingly popular and important in recent years. A contract, by definition, is a written or spoken (mostly written) law-enforced agreement containing the rights and duties of the parties. Because most of business contracts are complicated and tricky, the parties need to hire professional agents or lawyers for protecting their own rights. However, if we hire those professionals every time we sign contracts, it is going to be extremely costly and inefficient. Smart contracts perfectly solve this by working on 'If-Then' principle and also as escrow services. All participants need to put their money, ownership right or other tradable assets into smart contracts before any successful transaction. As long as all participating parties meet the requirement, smart contracts will simultaneously distribute stored assets to recipients and the distribution process will be witnessed and verified by the nodes on Ethereum network.

There are a couple of languages we can use to program smart contract. Solidity, an object-oriented and high-level language, is by far the most famous and well maintained one. We can use Solidity to create various smart contracts which can be used in different scenarios, including voting, blind auctions and safe remote purchase. In this lab, we will discuss the semantics and syntax of Solidity with specific explanation, examples, and practices.

After deciding the coding language, we need to pick an appropriate compiler. Among various compilers like Visual Code Studio, we will use Remix IDE in this and following labs because it can be directly accessed from browser where we can test, debug, and deploy smart contracts without any installation.

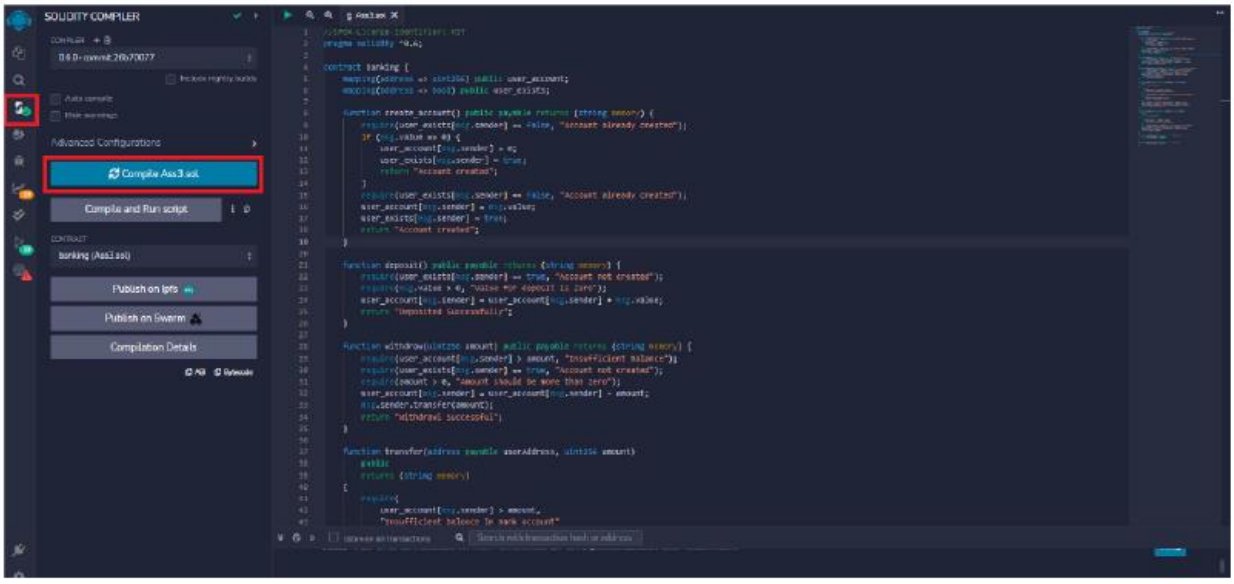
Steps to Execute Solidity Smart Contract using Remix IDE Remix IDE is generally used to compile and run Solidity smart contracts. Below are the steps for the compilation, execution, and debugging of the smart contract.

Step 1: Open Remix IDE on any of your browsers, select on the New File and click on Solidity to choose the environment.

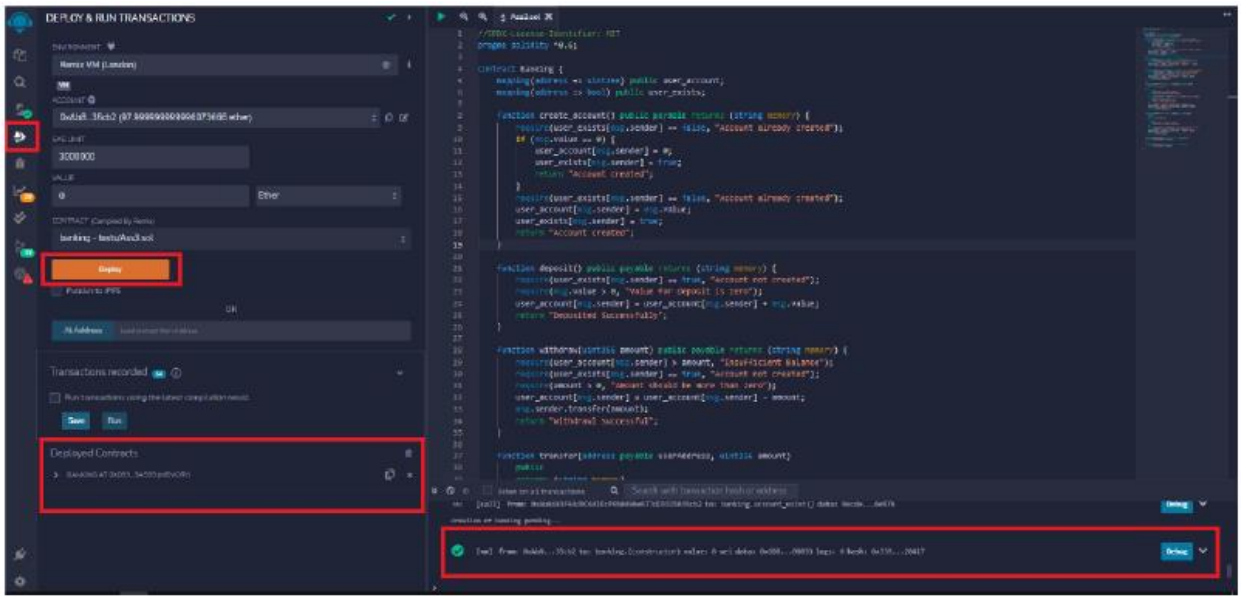


Step 2: Write the Smart contract in the code section, and click the Compile button under the

Compiler window to compile the contract.



Step 3: To execute the code, click on the Deploy button under Deploy and Run Transactions window. After deploying the code click on the drop-down on the console.



Code

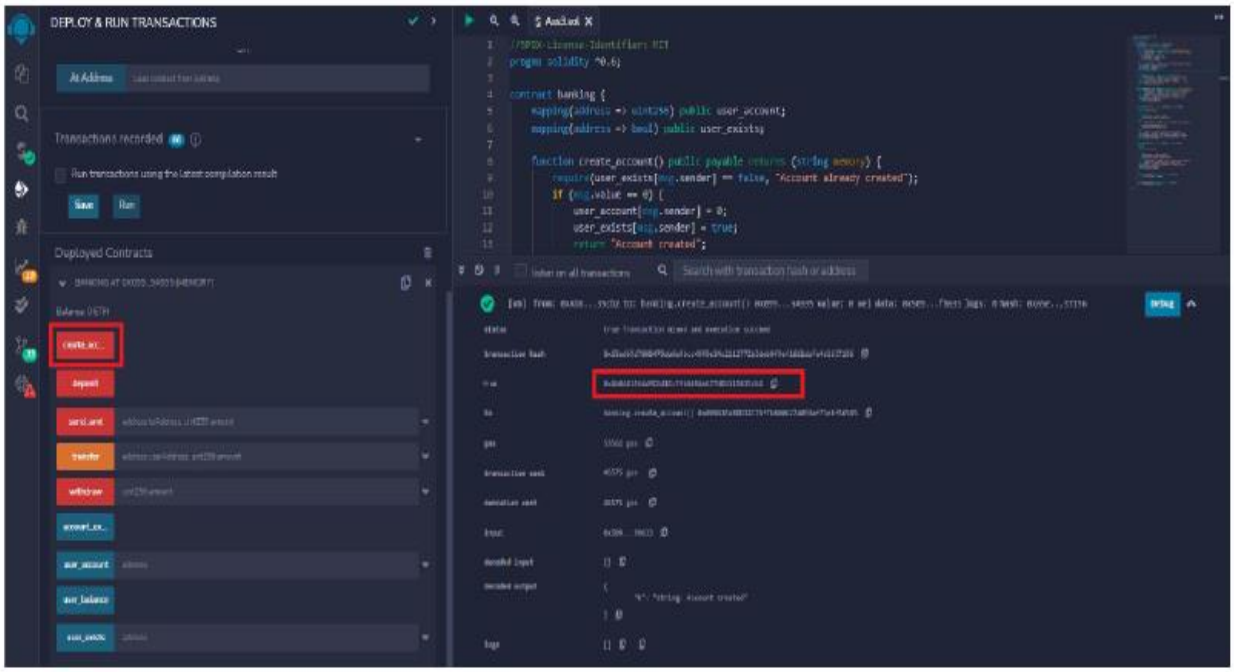
```
//SPDX-License-Identifier: MIT
pragma solidity ^0.6;

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)
```

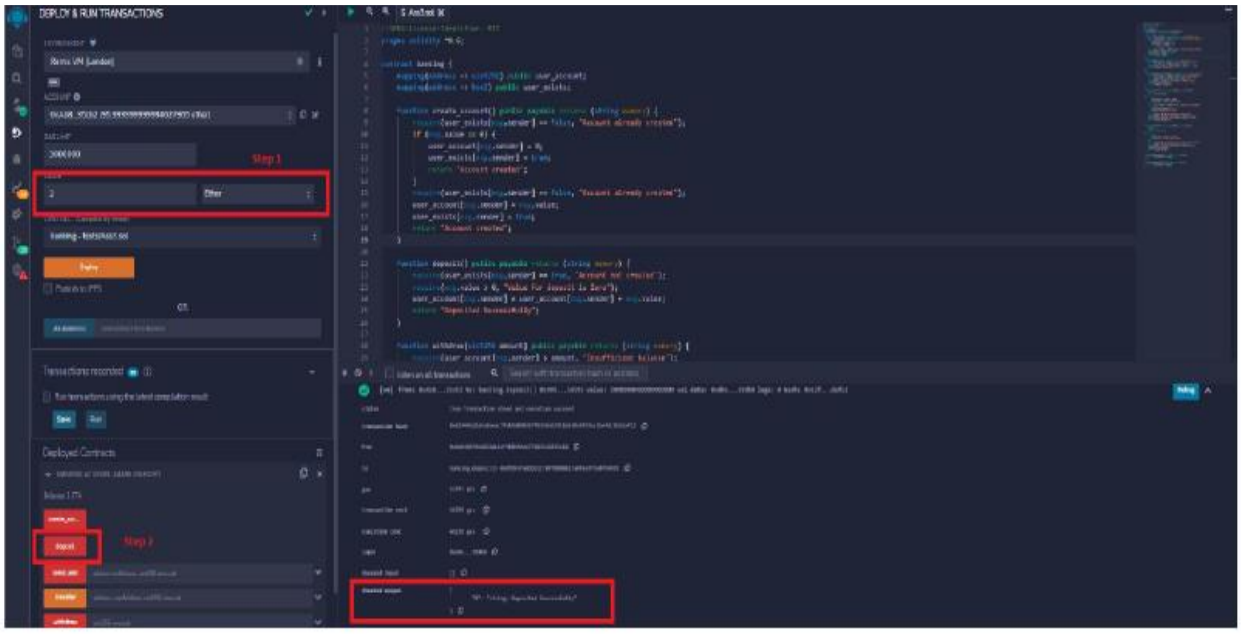
```
{
require(user_account[msg.sender]>amount,"Insufficeint balance in Bank account");
require(user_exists[msg.sender]==true,"Account is not created");
require(amount>0,"Amount should be more than zero");
user_account[msg.sender]=user_account[msg.sender]-amount;
toAddress.transfer(amount);
return "Transfer 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
After deploying the contact successful you can observe following buttonscreate_account, deposit, send_amt, transfer, account_exist, user_account, user_balance and user_exists. Refer the following output

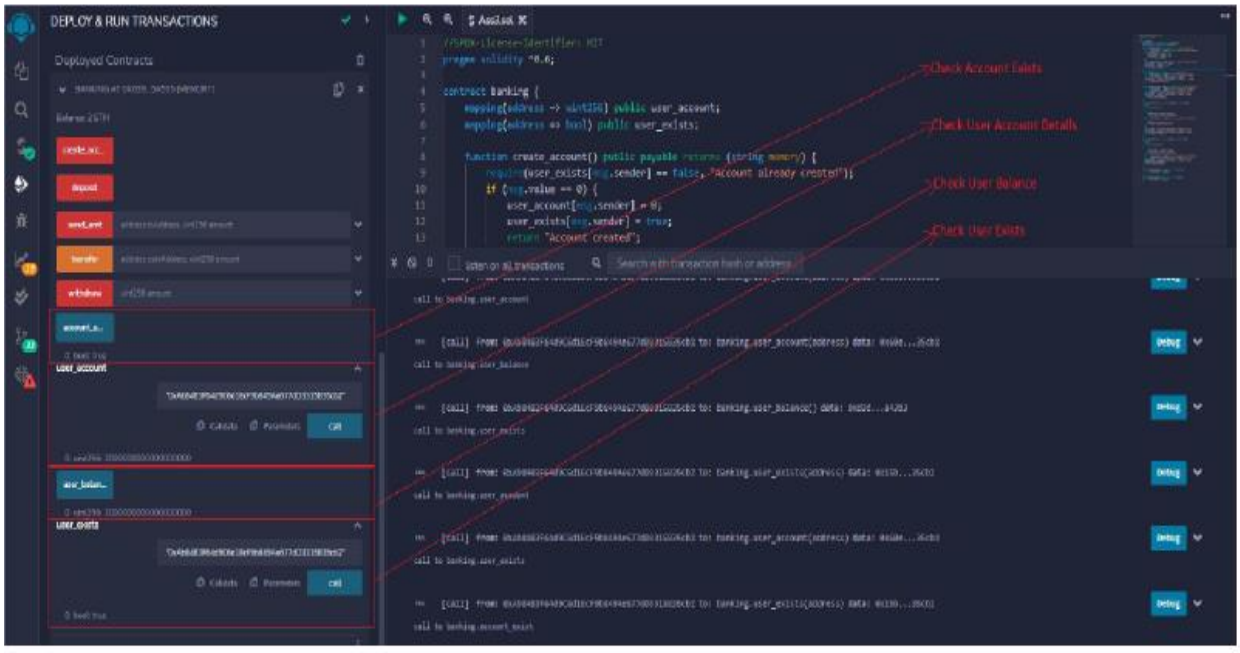
- Create account



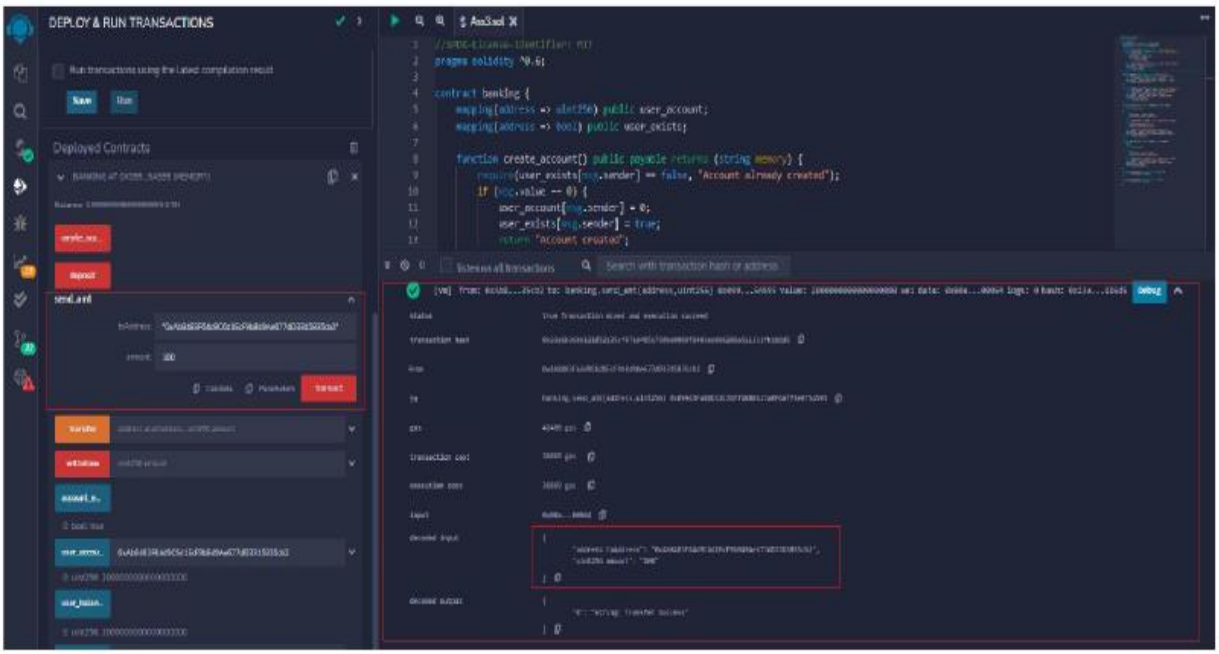
- Deposit Amount



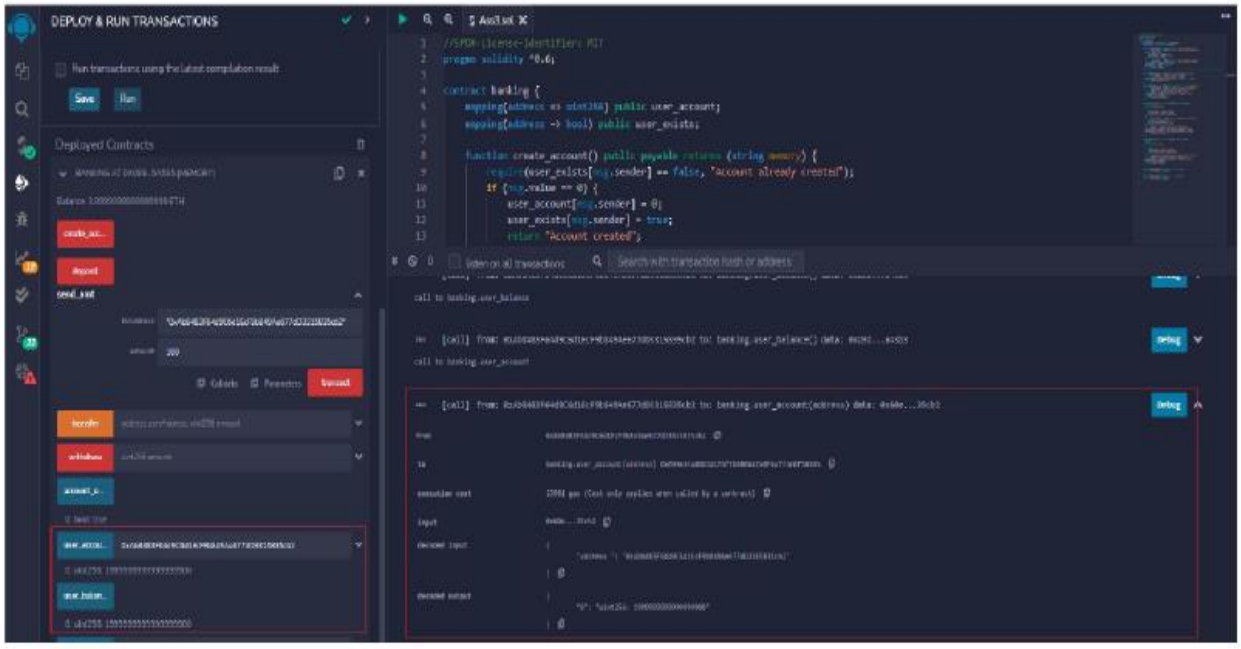
- Check Account Exists
- Check User Account Exists
- Check User Balance
- Check User Exists



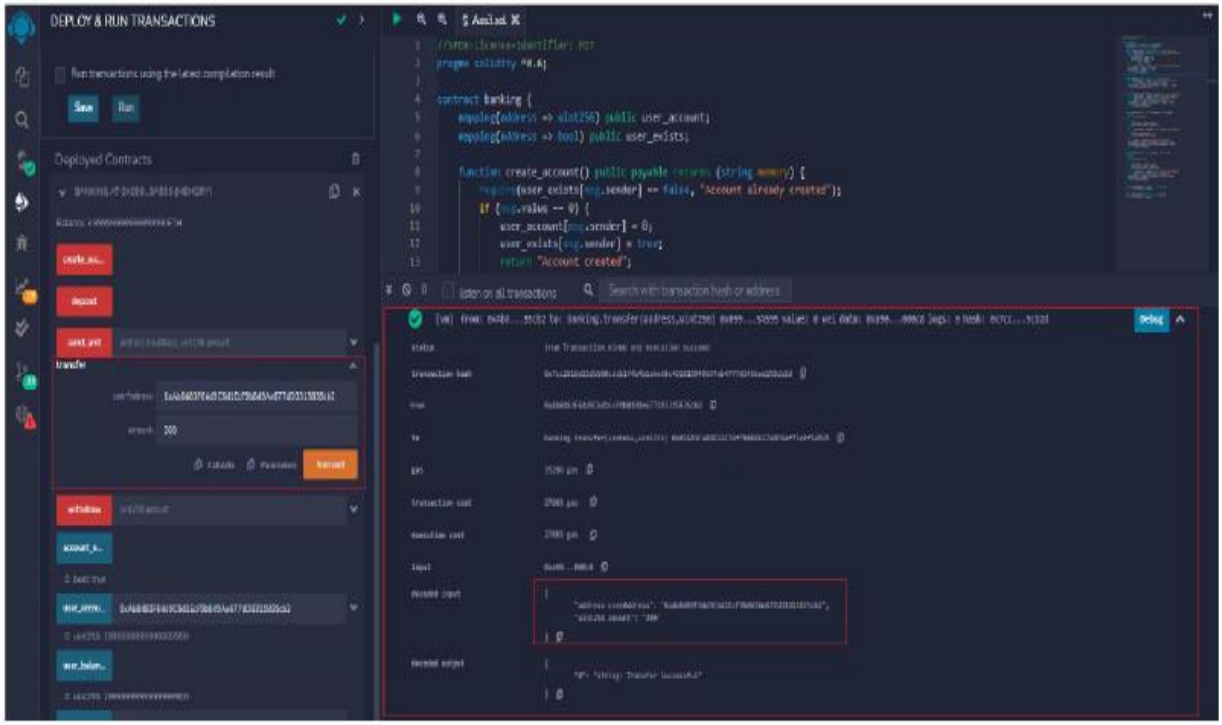
- Send Amount



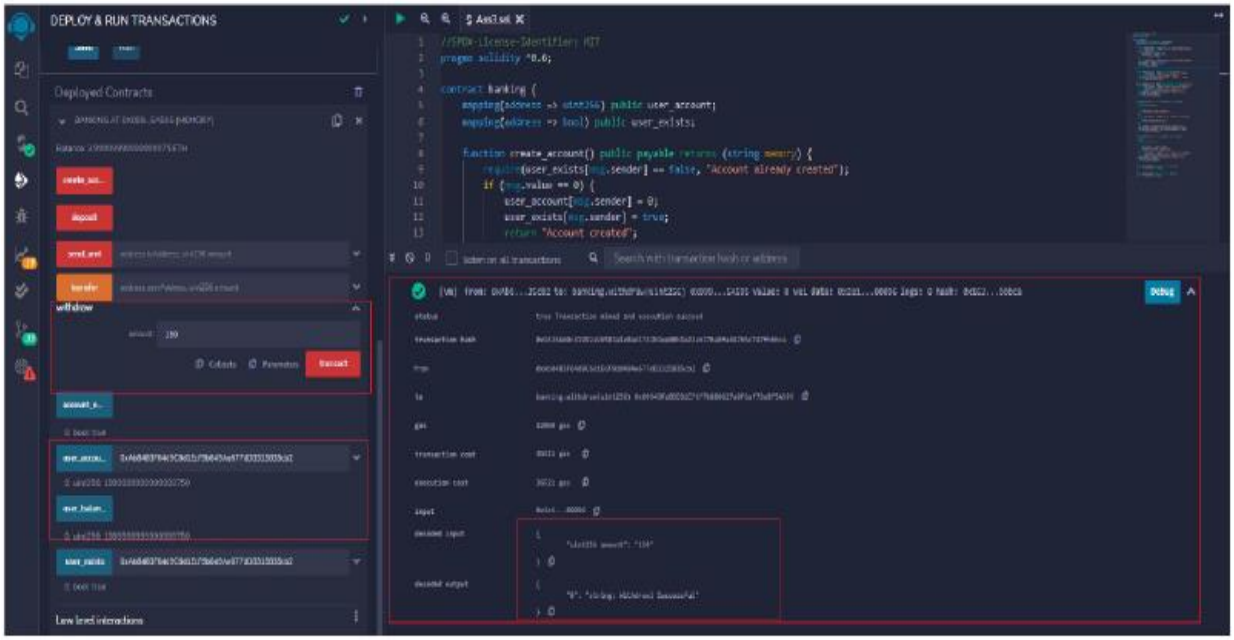
- Check User Account Balance



- Transfer Amount and Check User Account Balance



- Withdraw Amount and Check User Account Balance



Conclusion: Thus, we have studied a smart contract on a test network for Bank account of a Customer.

ASSIGNMENT NO. 4

TITLE: Write a program in solidity to create Student data**.**

Problem Statement Write a program in solidity to create Student data. Use the following constructs for following operations:

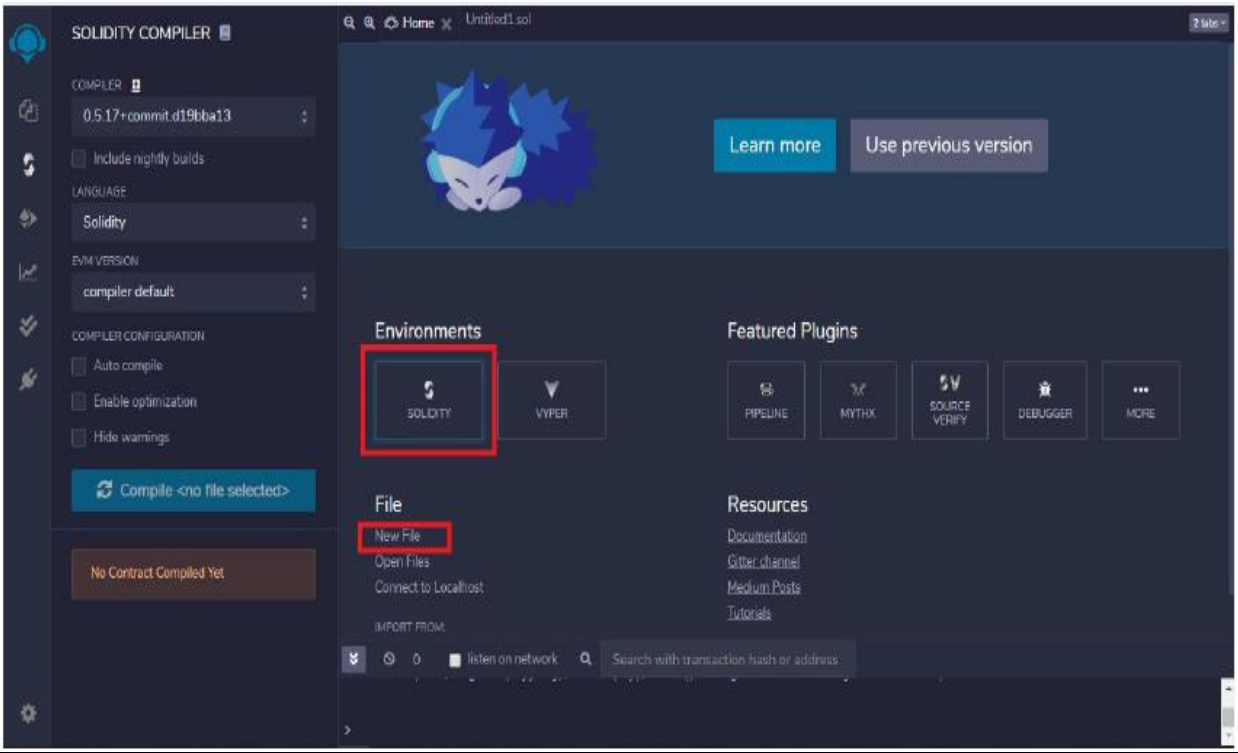
- Structures
- Arrays
- Fallback

Objective:

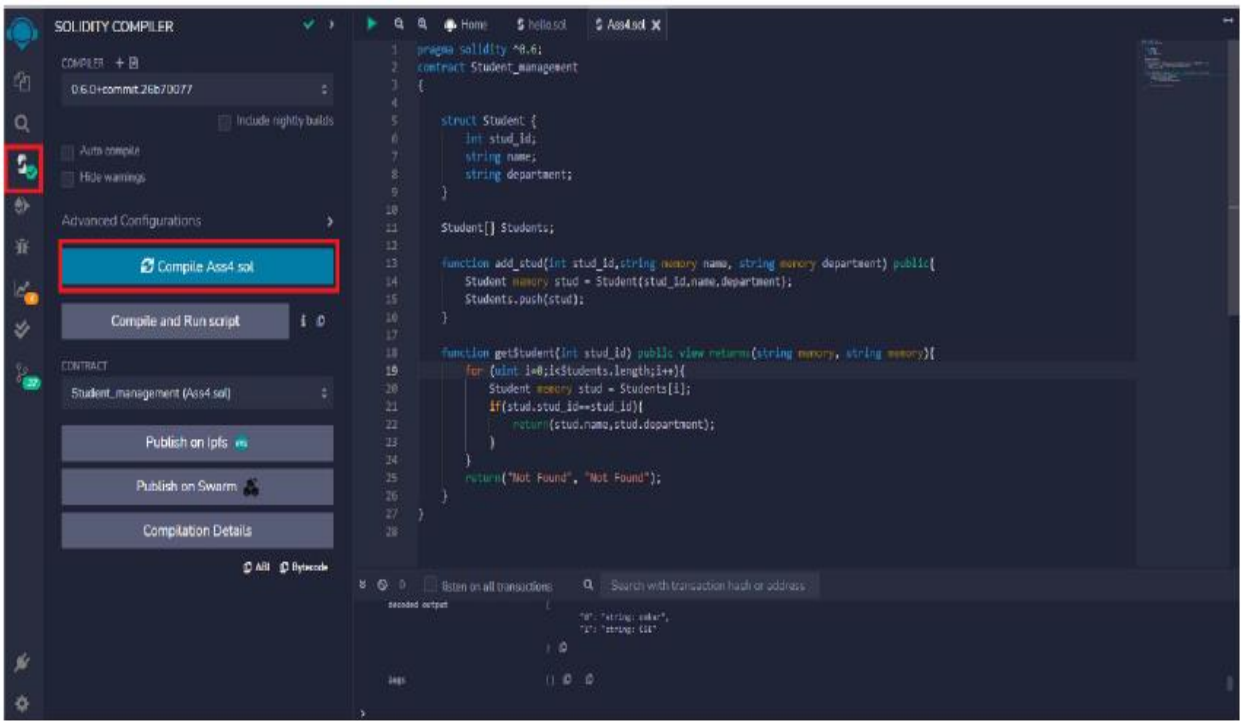
- Understand and explore the working of Blockchain technology and its applications

Theory:

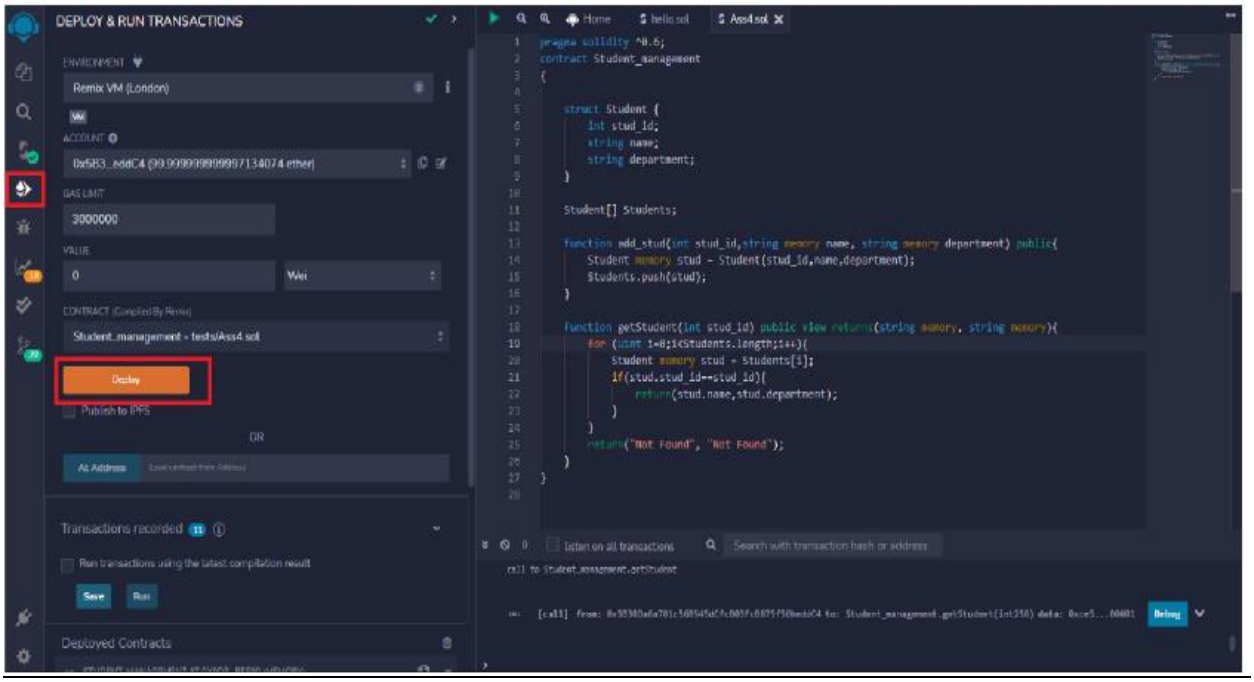
Step 1: Open Remix IDE on any of your browsers, select on the *New File* and click on *Solidity* to choose the environment.



Step 2: Write the Student Management code in the code section, and click the *Compile button* under the Compiler window to compile the contract



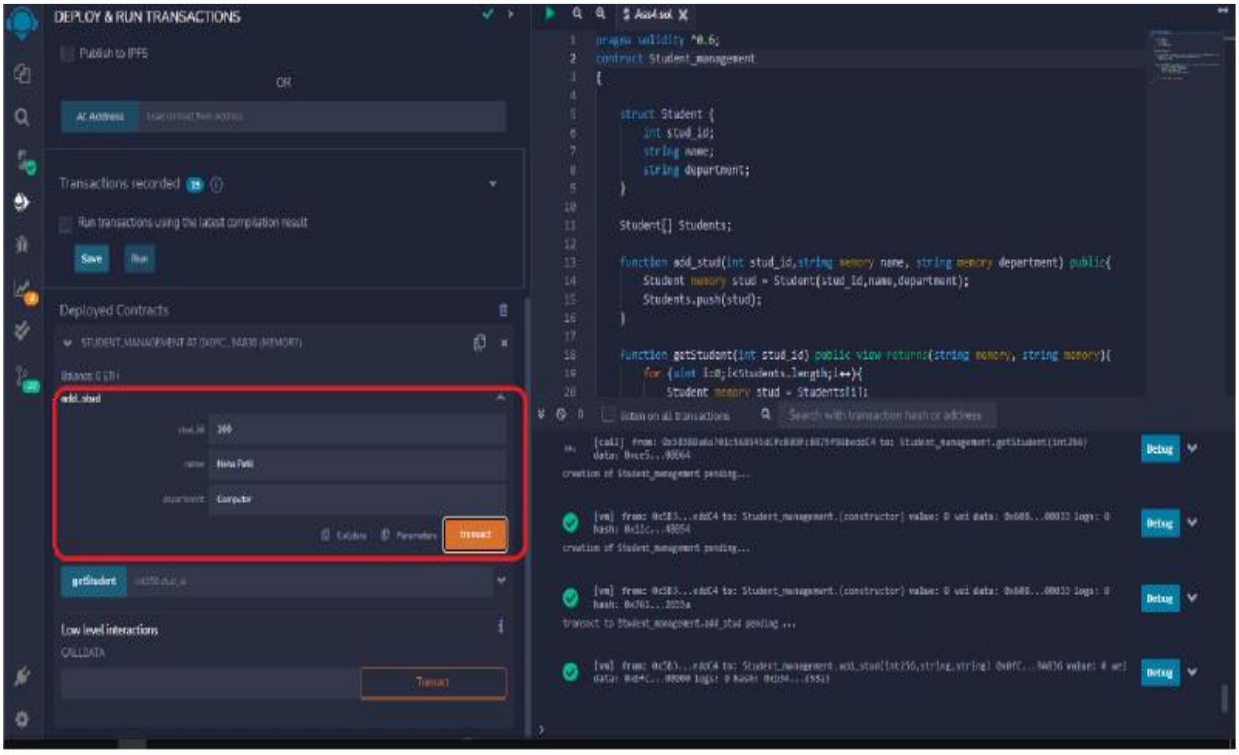
Step 3: To execute the code, click on the *Deploy button* under Deploy and Run Transactions window. After deploying the code click on the drop-down on the console.

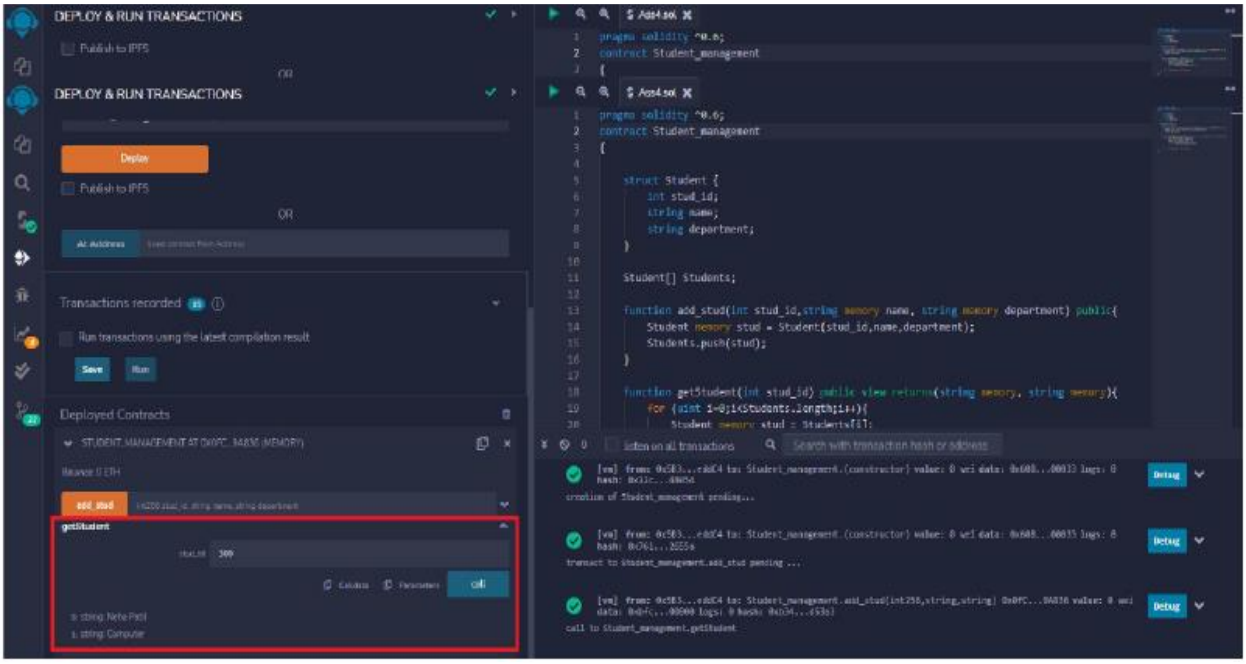


Code

```
pragma solidity ^0.6.
contract Student_management
{
struct Student {
int stud_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(int stud_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
After deploying the contact successful you can observe two button add_stud and getStudents. Give the input stud_id, name dept and click on getStudents button, enter the stud_id which you have given as an Input and get the information of Students name and department.
Refer the following output





Conclusion: Hence, we have studied a program in solidity to create Student data.

ASSIGNMENT NO. 5

TITLE: Write a survey report on types of Blockchain and its real time use cases.

Problem Statement Write a survey report on types of Blockchain and its real time use cases.

Objective:

- Understand and the types of Blockchain technology and its real time applications

Theory:

A blockchain is a digital ledger of all cryptocurrency transactions. It is constantly growing as "completed" blocks are added to it with a new set of recordings. Each block contains a cryptographic hash of the previous block, a timestamp, and transaction data. Bitcoin nodes use the block chain to differentiate legitimate Bitcoin transactions from attempts to re-spend coins that have already been spent elsewhere.

Types of Blockchain

There are three types of blockchains- public, private and consortium.

1. Public Blockchain: A public blockchain has absolutely no access restrictions. Anyone with an internet connection can send transactions to it as well as become a validator (i.e. participate in the consensus process). Bitcoin is the best example of a public blockchain.

2. Private Blockchain: A private blockchain is a little more centralized. Here, a central authority controls who can access the network and who can become a validator. Validators on a private blockchain are typically vetted by the central authority. Permissioned blockchains are often used in enterprise settings where centralized control is necessary. An example of a private blockchain is Hyperledger Fabric.

3. Consortium Blockchain: A consortium blockchain is a hybrid of the public and private blockchain. Here, a group of companies or organizations control who can access the network and who can become a validator. The selection of validators is typically done through a voting process. Consortium blockchains are often used in cases where multiple parties need to collaborate but no party is fully trusted. An example of a consortium blockchain is the R3 Corda platform.

Real-Time Use Cases of Blockchain

1. Supply Chain Management Blockchain can be used to create an immutable record of all the transactions in a supply chain. This can help to increase transparency and traceability in the supply chain.

2. Identity Management Blockchain can be used to create a digital identity for individuals, organizations, and devices. This can be used for KYC (know your customer) and AML (anti-money laundering) compliance.

3. Payments Blockchain can be used to process payments. This can be done using cryptocurrencies or fiat currencies.

4. Data Management Blockchain can be used to store data in a tamper-proof and decentralized manner. This can be used for data sharing and data security.

5. IoT Blockchain can be used to create a decentralized network of IoT devices. This can be used for data sharing and data security.

6. Predictive Analytics Blockchain can be used to create a decentralized network of predictive analytics models. This can be used for data sharing and data security.

Conclusion: Hence, we have studied to write a survey report on types of Blockchains and its real time use cases.