

Group C  
Blockchain Technology

By  
Faculty Name  
Ms P J Hajare

# Index

<b>GroupC:BlockchainTechnology</b>		
Any 5 assignments and 1 Mini project are mandatory.		
<b>Sr · No</b>	<b>Title</b>	<b>Pg No ·</b>
1	Installation of MetaMask and study spending Ether per transaction.	3
2	Create your own wallet using Metamask for crypto transactions.	7
3	Write a smart contract on a test network, for Bank account of a customer for following operations: <ul style="list-style-type: none"> <li>• Deposit money</li> <li>• Withdraw Money</li> <li>• Show balance</li> </ul>	14
4	Write a program in solidity to create Student data. Use the following constructs: <ul style="list-style-type: none"> <li>• Structures</li> <li>• Arrays</li> <li>• Fallback</li> </ul> Deploy this as smart contract on Ethereum and observe the transaction fee and Gas values.	25
5	Write a survey report on types of Blockchains and its real-time use cases.	31
6	Write a program to create a Business Network using Hyperledger	
7	<b>Mini Projects</b> <b>Mini Project</b> - Develop a Blockchain based application and App (de-centralized app) for e-voting system.	
	<b>Mini Project</b> - Develop a Blockchain based application for transparent and genuine charity	
	<b>Mini Project</b> - Develop a Blockchain based application for health-related medical records	
	<b>Mini Project</b> - Develop a Blockchain based application for mental health	

<b>Assignment No.</b>	1
<b>Title</b>	Installation of MetaMask and study spending Ether per transaction
<b>Subject</b>	Blockchain Technology
<b>Class</b>	BE (CE)
<b>Roll No</b>	
<b>Date</b>	
<b>Signature</b>	

# AssignmentNo: 1

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

**Objective:** To understand and explore the working of Blockchain Technology and its applications.

**Course Outcome:**

CO6: Interpret the basic concepts in Blockchain technology and its applications.

**Description:**

MetaMask is available as an in-browser application for desktop or laptop computers, as well as a smartphone app available on both major app stores. What makes it so popular is its seamless integration with many different major crypto websites, including NFT marketplace Open Sea, and decentralized exchanges including 1inch, Uniswap, and Quickswap.

MetaMask also works with hardware cryptocurrency wallets ledger and Trezor, allowing users to transfer crypto and NFTs from the software-based hot wallet to the hardware based cold wallets for secure storage.

It is easy to use MetaMask on a connected website to send, receive, or trade tokens. Before exchanging any tokens, MetaMask will pop up in your browser to confirm the details, including the contracts, price. Before confirming the transaction, user will be able to adjust their gas limits to either pay more to speed it up, or reduce their max price while showing down the confirmation.

## Steps to create MetaMask.


**Step 1:** Go to [Chrome Web Store Extensions Section](#). Search MetaMask.

**Step 2:** Download the METAMASK .Click on the button “Add to Chrome”.

[Home](#) > [Extensions](#) > [MetaMask](#)



MetaMask

 [metamask.io](https://metamask.io)

★★★★★ 2,706 ⓘ | [Productivity](#) | 10,000,000+ users

Add to Chrome

Overview

Privacy practices

Reviews

Support

Related

**Step 3:** MetaMask wallet installation. Click on the MetaMask extension and click on “Get Started”.



Welcome to MetaMask

Connecting you to Ethereum and the Decentralized Web.

We're happy to see you.

Get Started

**Step 4:** Click on “I Agree.”



## 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](#).

**Conclusion:** Hence, we have studied the installation of MetaMask.

<b>Assignment No.</b>	2
<b>Title</b>	Create your own wallet using MetaMask for crypto transactions.
<b>Subject</b>	Blockchain Technology
<b>Class</b>	BE (CE)
<b>Roll No</b>	
<b>Date</b>	
<b>Signature</b>	

## Assignment No. 2

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

**Objective:** Understand and explore the working of Blockchain technology and its applications.

**Course Outcome:**

CO6: Interpret the basic concepts in Blockchain technology and its application.

**Description:**

MetaMask is a cryptocurrency wallet used to interact with the Ethereum Blockchain. It can be accessed through an app or through a browser extension. MetaMask is a popular cryptocurrency wallet known for its ease of use, availability on both desktops and mobile devices, the ability to buy, send, and receive cryptocurrency from within the wallet, and collect non-fungible tokens (NFTs) across two block chains. While experienced crypto users will appreciate the simplicity and fast transactions, those new in the space are at a higher risk of losing their tokens from lost secret phrases, malicious websites, and other cryptocurrency scams.

Similarly, you'll need a crypto wallet to transact with a Blockchain.

A wallet is your personal key to interact with the cryptographic world. It powers you to buy, sell or transfer assets on the Blockchain.

And MetaMask is a wallet for the most diverse Blockchain in existence—Ethereum. It's your gateway to its DeFi ecosystem, non-fungible tokens (NFTs), ERC-20 tokens, and practically—everything Ethereum.

It's available as an app for iOS and Android. In addition, you can use this as an extension with a few web browsers: Chrome, Firefox, Brave, and Edge.

Let's take a look at some of the prominent features of this wallet:

### **Ease of use**

Starting with MetaMask is easy, quick, and anonymous. You don't even need an email address. Just set up a password and remember (and store) the secret recovery phrase, and you're done.

### **Security**

Your information is encrypted in your browser that nobody has access to. In the event of a lost password, you have the 12-word secret recovery phase (also called a seed phrase) for



recovery. Notably, it's essential to keep the seed phrase safe, as even MetaMask has no information about it. Once lost, it can't be retrieved.

### **Built-In Crypto Store**

If you're wondering, no, you can't buy Bitcoin with MetaMask. It only supports Ether and other Ether-related tokens, including the famous ERC-20 tokens. Cryptocurrencies (excluding Ether) on Ethereum are built as ERC-20 tokens.

### **Backup and Restore**

MetaMask stores your information locally. So, in case you switch browsers or machines, you can restore your MetaMask wallet with your secret recovery phrase.

### **Community Support**


As of August 2021, MetaMask was home to 10 million monthly active users around the world. It's simple and intuitive user interface keeps pushing these numbers with a recorded 1800% increase from July 2020.

Conclusively, try MetaMask if hot wallets are your pick. Let's begin with the installation before moving to its use cases. Further sections entail the illustration for Chrome web browser and Android mobile platform.

**Step 1:** Click on the "Create a wallet".




## New to MetaMask?



No, I already have a Secret Recovery Phrase

Import your existing wallet using a Secret Recovery Phrase

Import wallet



Yes, let's get set up!

This will create a new wallet and Secret Recovery Phrase

Create a Wallet

**Step 2:** Create a password for your wallet. This password is to be entered every time the browser is launched and wants to use MetaMask. A new password needs to be created if chrome is uninstalled or if there is a switching of browsers. In that case, go through the Import Wallet button. This is because MetaMask stores the keys in the browser. Agree to Terms of Use.



## Create Password

New password (8 characters min)

Confirm password



I have read and agree to the [Terms of Use](#)

Create

**Step 3:** Click on the dark area which says *Click here to reveal secret words* to get your secret phrase.

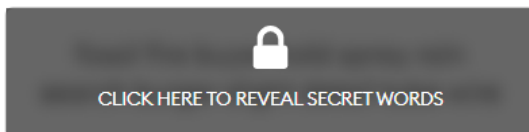
**Step 4:** This is the most important step. Back up your secret phrase properly. Do not store your secret phrase on your computer. Please read everything on this screen until you understand it completely before proceeding. The secret phrase is the only way to access your wallet if you forget your password. Once done click the Next button.



## 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.



[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 5:

Securely store the seed phrase for your wallet. Click on “Click here to reveal secret words” to show the seed phrase.

MetaMask requires that you store your seed phrase in a safe place. It is the only way to recover your funds should your device crash or your browser reset. We recommend you write it down. The most common method is to write your 12-word phrase on a piece of paper and store it safely in a place where only you have access. Note: if you lose your seed phrase, MetaMask can’t help you recover your wallet and your funds will be lost forever.

Never share your seed phrase or your private key to anyone or any site, unless you want them to have full control over your funds.



< 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 6:** Click the *Confirm* button. Please follow the tips mentioned.



## Congratulations

You passed the test - keep your Secret Recovery Phrase safe, it's your responsibility!

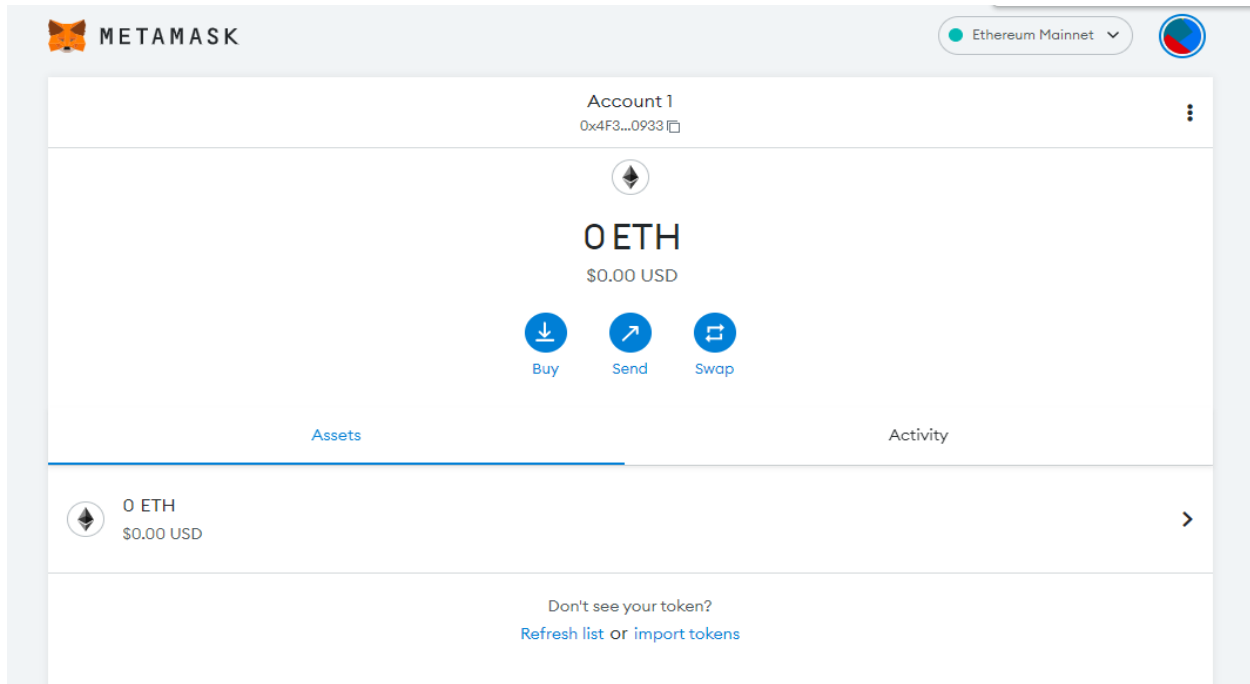
### Tips on storing it safely

- 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 7:** One can see the balance and copy the address of the account by clicking on the *Account 1* area.



**Conclusion:** Hence, we have studied to create a wallet using MetaMask.

<b>Assignment No.</b>	3
<b>Title</b>	Write a smart contract on a test network for Bank account of a customer for following operations. <ul style="list-style-type: none"> <li>• Deposit Money</li> <li>• Withdraw Money</li> <li>• Show Balance</li> </ul>
<b>Subject</b>	Blockchain Technology
<b>Class</b>	BE (CE)
<b>Roll No</b>	
<b>Date</b>	
<b>Signature</b>	

## 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

**Objective:** Understand and explore the working of Blockchain technology and its applications.

**Course Outcome:**

CO6: Interpret the basic concepts in Blockchain technology and its application.

**Description:**

First of all, 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.

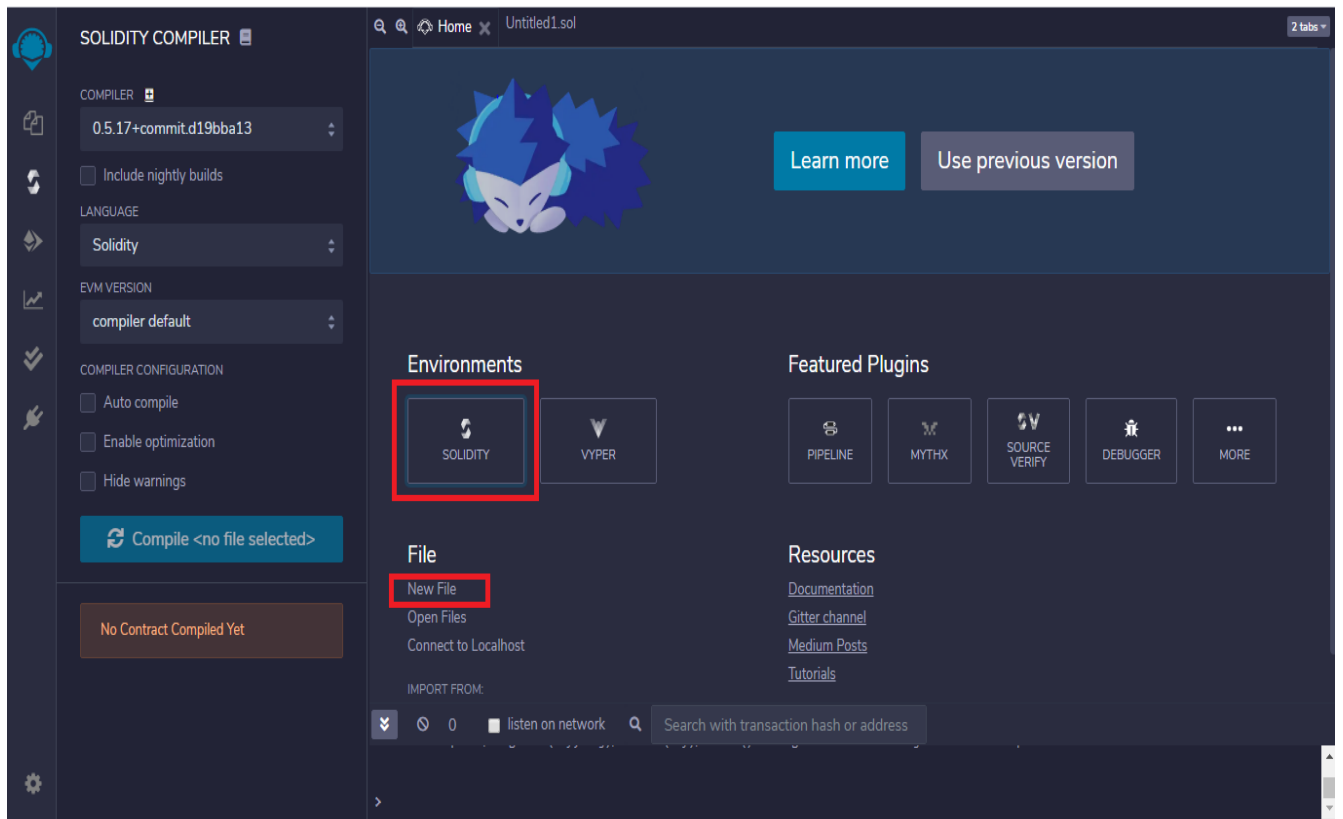
There are a couple of languages we can use to program smart contract. Solidity, an object-oriented and high-level language, is by far the most famous and well maintained one. We can use Solidity to create various smart contracts which can be used in different scenarios, including voting, blind auctions 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

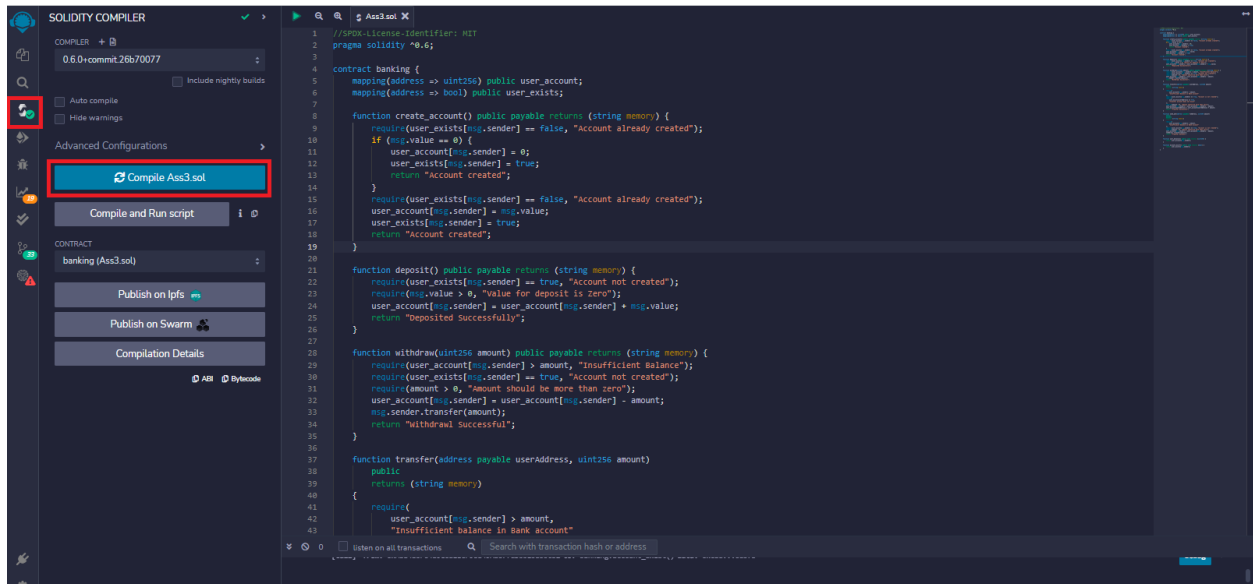
Remix IDE is generally used to compile and run Solidity smart contracts. Below are the steps for the compilation, execution, and debugging of the smart contract.

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

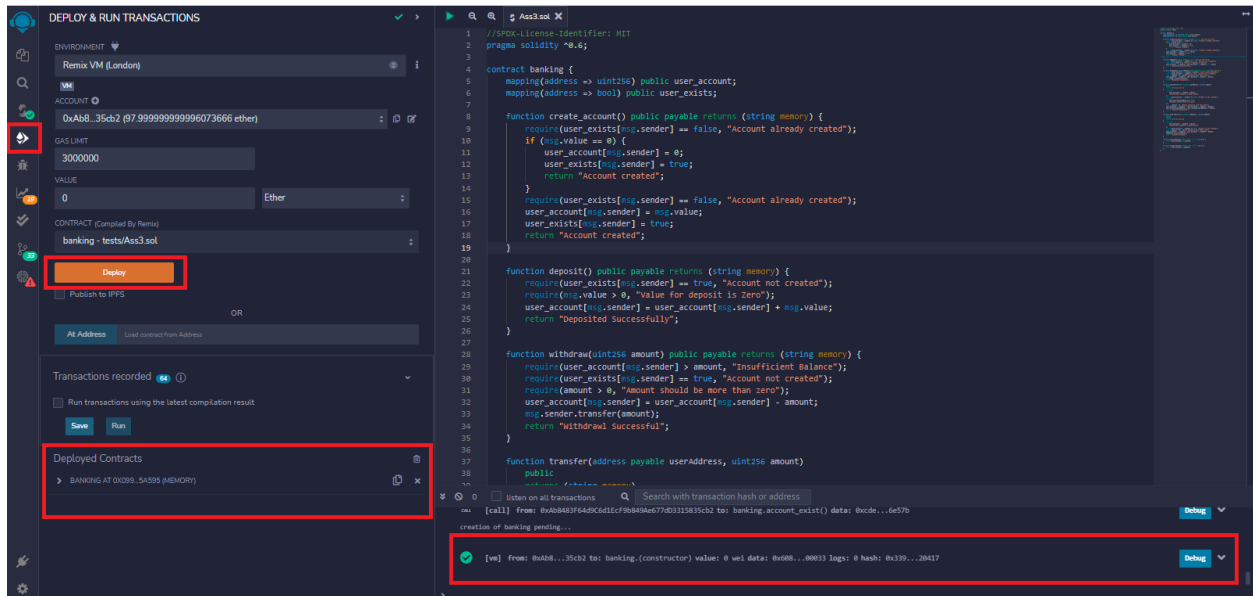


**Step 2:** Write the Smart contract in the code section, and click the *Compile button* under the Compiler window to compile the contract.





**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 contract successful you can observe following buttons create\_account, deposit, send\_amt, transfer, account\_exist, user\_account, user\_balance and user\_exists.

Refer the following output

- Create account

The screenshot displays a web application interface for a banking contract. On the left, a sidebar contains a 'Deployed Contracts' section with a list of transactions. The 'create\_acc' button is highlighted with a red box. The right panel shows the Solidity code for the 'banking' contract, with the 'create\_account' function highlighted. Below the code, a transaction log shows the successful execution of the 'create\_account' function, with the transaction hash and gas details highlighted.

- Deposit Amount

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

- Send Amount

The screenshot displays the Remix IDE interface. On the left, the 'Deployed Contracts' panel shows the 'BANKING' contract deployed at address 0x099...5A595. The 'send\_amt' function is selected in the function list. The main editor shows the Solidity code for the 'BANKING' contract. The right sidebar shows the 'Debug' console with the transaction details, including the transaction hash, gas used, and the decoded input/output.

```

1 //SPDX-License-Identifier: MIT
2 pragma solidity ^0.6;
3
4 contract banking {
5     mapping(address => uint256) public user_account;
6     mapping(address => bool) public user_exists;
7
8     function create_account() public payable returns (string memory) {
9         require(user_exists[msg.sender] == false, "Account already created");
10        if (msg.value == 0) {
11            user_account[msg.sender] = 0;
12            user_exists[msg.sender] = true;
13            return "Account created";
14        }
15    }
16 }

```

Transaction details from the debug console:

- status: true transaction mined and execution succeed
- transaction hash: 0x240b306d12182113c48716485c730b0889f1483ee86286d511317911605
- from: 0x0A843F649C6d1Ecf9B49A677d3115835c2
- to: banking.send\_amt(address,uint256) 0x8943fa80c12c767f888627a86a73e8f1A595
- gas: 42000 gas
- transaction cost: 36800 gas
- execution cost: 36800 gas
- input: 0x08a...00064
- decoded input: { "address": "0x0A843F649C6d1Ecf9B49A677d3115835c2", "uint256 amount": "100" }
- decoded output: { "0": "string: Transfer Success" }

- Check User Account Balance

The screenshot displays the Remix IDE interface. On the left, the 'Deployed Contracts' panel shows the 'BANKING' contract deployed at address 0x099...5A595. The 'user\_balance' function is selected in the function list. The main editor shows the Solidity code for the 'BANKING' contract. The right sidebar shows the 'Debug' console with the transaction details, including the transaction hash, gas used, and the decoded input/output.

```

1 //SPDX-License-Identifier: MIT
2 pragma solidity ^0.6;
3
4 contract banking {
5     mapping(address => uint256) public user_account;
6     mapping(address => bool) public user_exists;
7
8     function create_account() public payable returns (string memory) {
9         require(user_exists[msg.sender] == false, "Account already created");
10        if (msg.value == 0) {
11            user_account[msg.sender] = 0;
12            user_exists[msg.sender] = true;
13            return "Account created";
14        }
15    }
16 }

```

Transaction details from the debug console:

- call to banking.user\_balance
- call [call] from: 0x0A843F649C6d1Ecf9B49A677d3115835c2 to: banking.user\_balance() data: 0xd3d...843b3
- call to banking.user\_account
- call [call] from: 0x0A843F649C6d1Ecf9B49A677d3115835c2 to: banking.user\_account(address) data: 0x08a...35c2
- from: 0x0A843F649C6d1Ecf9B49A677d3115835c2
- to: banking.user\_account(address) 0x8943fa80c12c767f888627a86a73e8f1A595
- execution cost: 23981 gas (Cost only applies when called by a contract)
- input: 0x08a...35c2
- decoded input: { "address": "0x0A843F649C6d1Ecf9B49A677d3115835c2" }
- decoded output: { "0": "uint256: 1999999999999999999" }

- Transfer Amount and Check User Account Balance

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the 'BANKING AT 0x099...5A595 (MEMORY)' contract. The 'transfer' function is selected, with the 'userAddress' field set to '0xA8483F64dC615fF8484A677d3315835c2' and the 'amount' field set to '200'. The 'transact' button is highlighted. On the right, the Solidity code for the 'banking' contract is visible, including the 'create\_account' function. Below the code, the transaction details are shown, including the status 'true Transaction mined and execution succeed', the transaction hash, and the decoded input/output. The decoded input shows the 'address userAddress' and 'uint256 amount' as '200'. The decoded output shows the string 'Transfer Successful'.

- Withdraw Amount and Check User Account Balance

The screenshot displays the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the 'BANKING AT 0x099...5A595 (MEMORY)' contract. The 'withdraw' function is selected, with the 'amount' field set to '150'. The 'transact' button is highlighted. On the right, the Solidity code for the 'banking' contract is visible, including the 'create\_account' function. Below the code, the transaction details are shown, including the status 'true Transaction mined and execution succeed', the transaction hash, and the decoded input/output. The decoded input shows the 'uint256 amount' as '150'. The decoded output shows the string 'Withdrawal Successful'.

**Conclusion:** Hence, we have studied a smart contract on a test network for Bank account of a customer



<b>Assignment No.</b>	4
<b>Title</b>	Write a program in solidity to create Student data. Use the following constructs: <ul style="list-style-type: none"> <li>• Structures</li> <li>• Arrays</li> <li>• Fallback</li> </ul>
<b>Subject</b>	Blockchain Technology
<b>Class</b>	BE (CE)
<b>Roll No</b>	
<b>Date</b>	
<b>Signature</b>	

# Assignment No. 4

**Title:** 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 value.

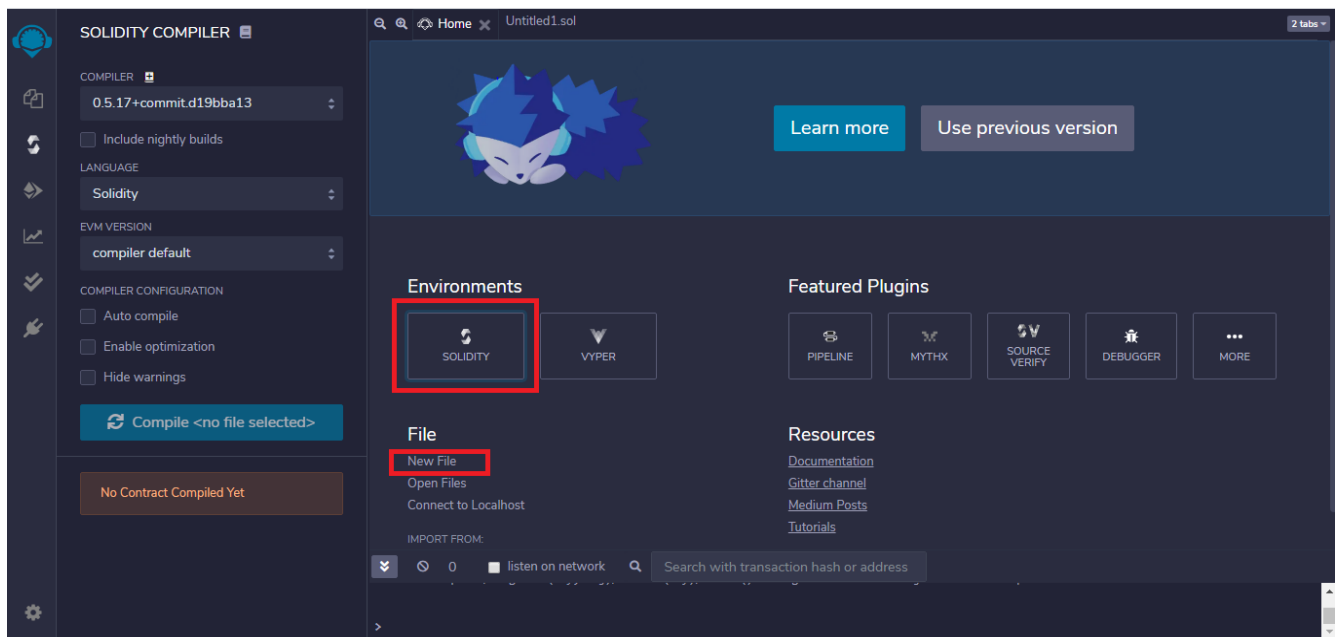
**Objective:** Understand and explore the working of Blockchain technology and its applications.

**Course Outcome:**

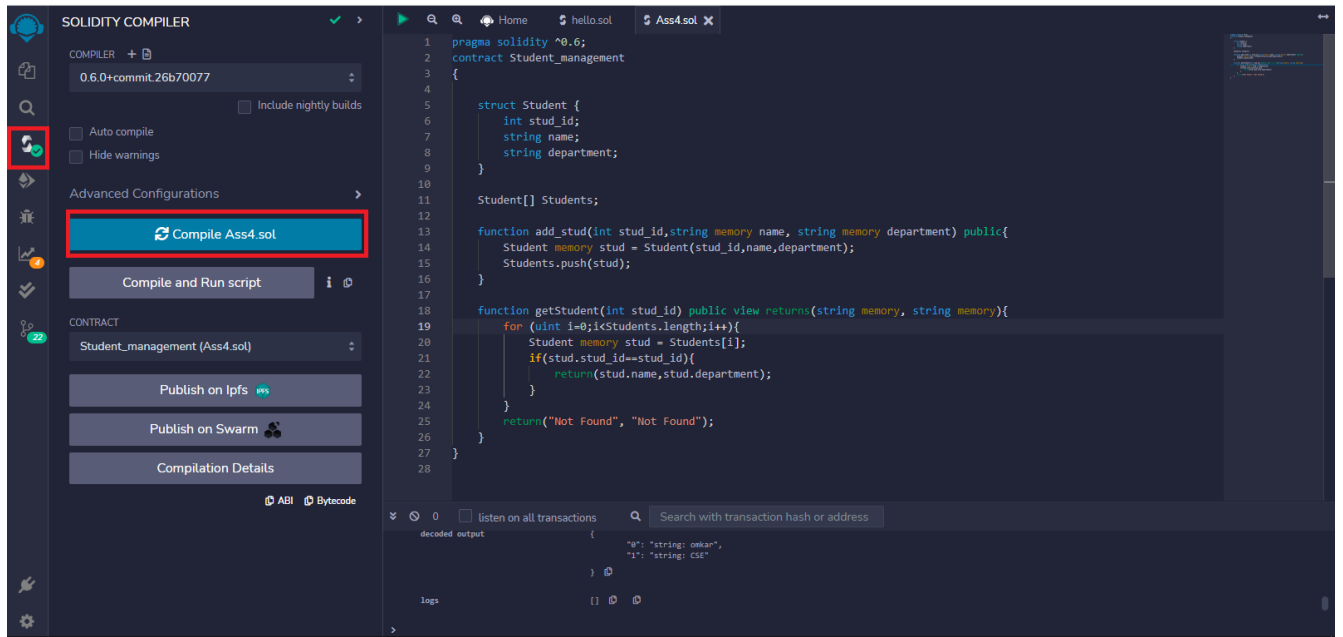
CO6: Interpret the basic concepts in Blockchain technology and its application.

**Description:**

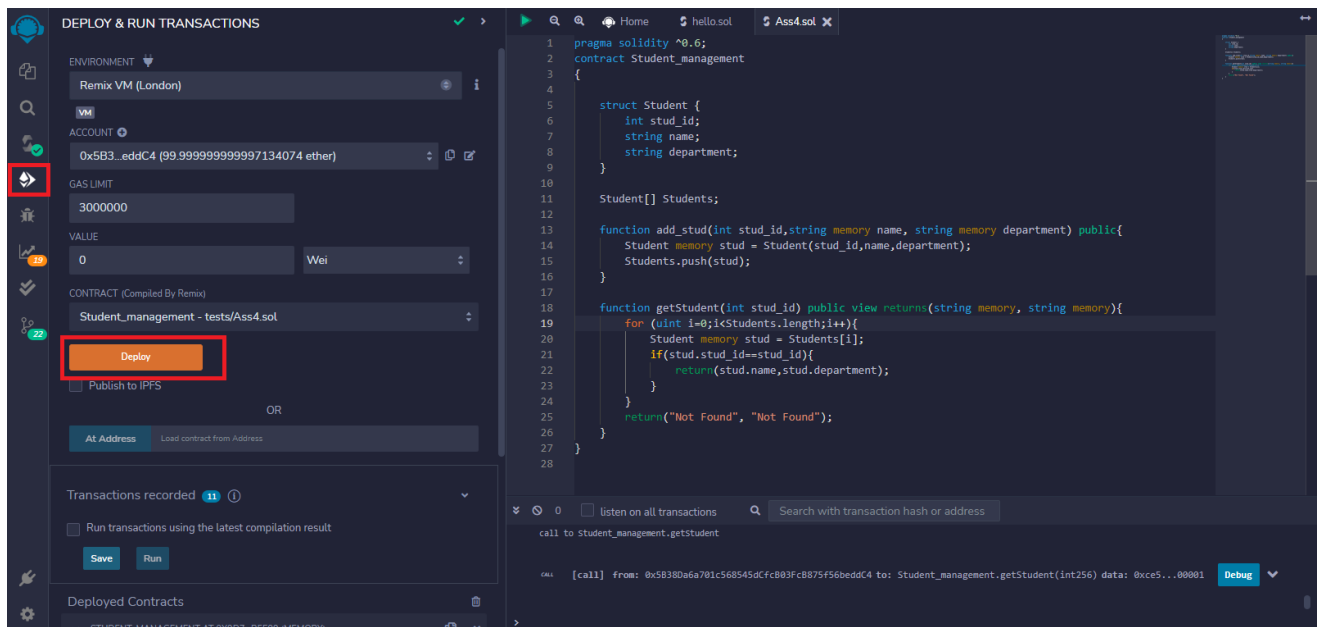
**Step 1:** Open Remix IDE on any of your browsers, select on the *New File* and click on *Solidity* to choose the environment.



**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 {
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

After deploying the contract successfully you can observe two buttons `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

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active. Under 'Deployed Contracts', the 'STUDENT\_MANAGEMENT' contract is selected. The 'add\_stud' function is highlighted with a red box. The input fields are: `stud_id` (100), `name` (Neha Patti), and `department` (Computer). The 'Transact' button is visible. Below the function call, the 'Low level interactions' section shows the 'CALLDATA' field with a 'Transact' button. On the right, the Solidity code for the 'Student\_management' contract is displayed. The code defines a `Student` struct with `int stud_id`, `string name`, and `string department`. It includes an `add_stud` function that pushes a new student to the `Students` array and a `getStudent` function that returns the student's name and department.

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is active. Under 'Deployed Contracts', the 'STUDENT\_MANAGEMENT' contract is selected. The 'getStudent' function is highlighted with a red box. The input field is: `stud_id` (100). The 'Call' button is visible. Below the function call, the 'Low level interactions' section shows the 'CALLDATA' field with a 'Transact' button. On the right, the Solidity code for the 'Student\_management' contract is displayed. The code defines a `Student` struct with `int stud_id`, `string name`, and `string department`. It includes an `add_stud` function that pushes a new student to the `Students` array and a `getStudent` function that returns the student's name and department.

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

<b>Assignment No.</b>	5
<b>Title</b>	Write a survey report on types of Blockchain and its real time use cases.
<b>Subject</b>	Blockchain Technology
<b>Class</b>	BE (CE)
<b>Roll No</b>	
<b>Date</b>	
<b>Signature</b>	

## Assignment No. 5

**Title:** Write a survey report on types of Blockchains and its real time use cases.

**Objective:** Understand and the types of Blockchain technology and its real time applications.

**Course Outcome:**

CO6: Interpret the basic concepts in Blockchain technology and its application.

**Description:**

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.

