List of Experiments

Sr. No.	Title
1	Installation of MetaMask and study spending Ether per transaction.
2	Create your own wallet using Metamask for crypto transactions
3	Write a smart contract on a test network, for Bank account of a customer for following operations: • Deposit money • Withdraw Money • Show balance
4	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.
5	Write a survey report on types of Blockchains and its real time use cases.

LAB ASSIGNMENT 1

AIM :- Installation of MetaMask and study spending Ether per transaction.

REQUIREMENTS :- Windows 11/Linux.

THEORY :-

METAMASK :-

MetaMask is an extension for accessing Ethereum enabled distributed applications, or "Dapps" in your browser!

The extension injects the Ethereum web3 API into every website's javascript context, so that dapps can read from the blockchain.

MetaMask also lets the user create and manage their own identities (via private keys, local client wallet and hardware wallets like Trezor™), so when a Dapp wants to perform a transaction and write to the blockchain, the user gets a secure interface to review the transaction, before approving or rejecting it.

MetaMask also helps warn you when you navigate to sites that are known to have engaged in phishing, or that have names that are suspiciously similar to popular phishing targets.

Because it adds functionality to the normal browser context, MetaMask requires the permission to read and write to any webpage. You can always "view the source" of MetaMask the way you do any Chrome extension, or view the source code on Github:

https://github.com/MetaMask/metamask-extension

Enables access to:

Web 3.0

Dapps

NFTs

erc20

How to Install and Use Metamask on Google Chrome?

Step 1: Go to Chrome Web Store Extensions Section.

Step 2: Search *MetaMask*.

Step 3: Check the number of downloads to make sure that the legitimate MetaMask is being installed, as hackers might try to make clones of it.

Step 4: Click the *Add to Chrome* button.

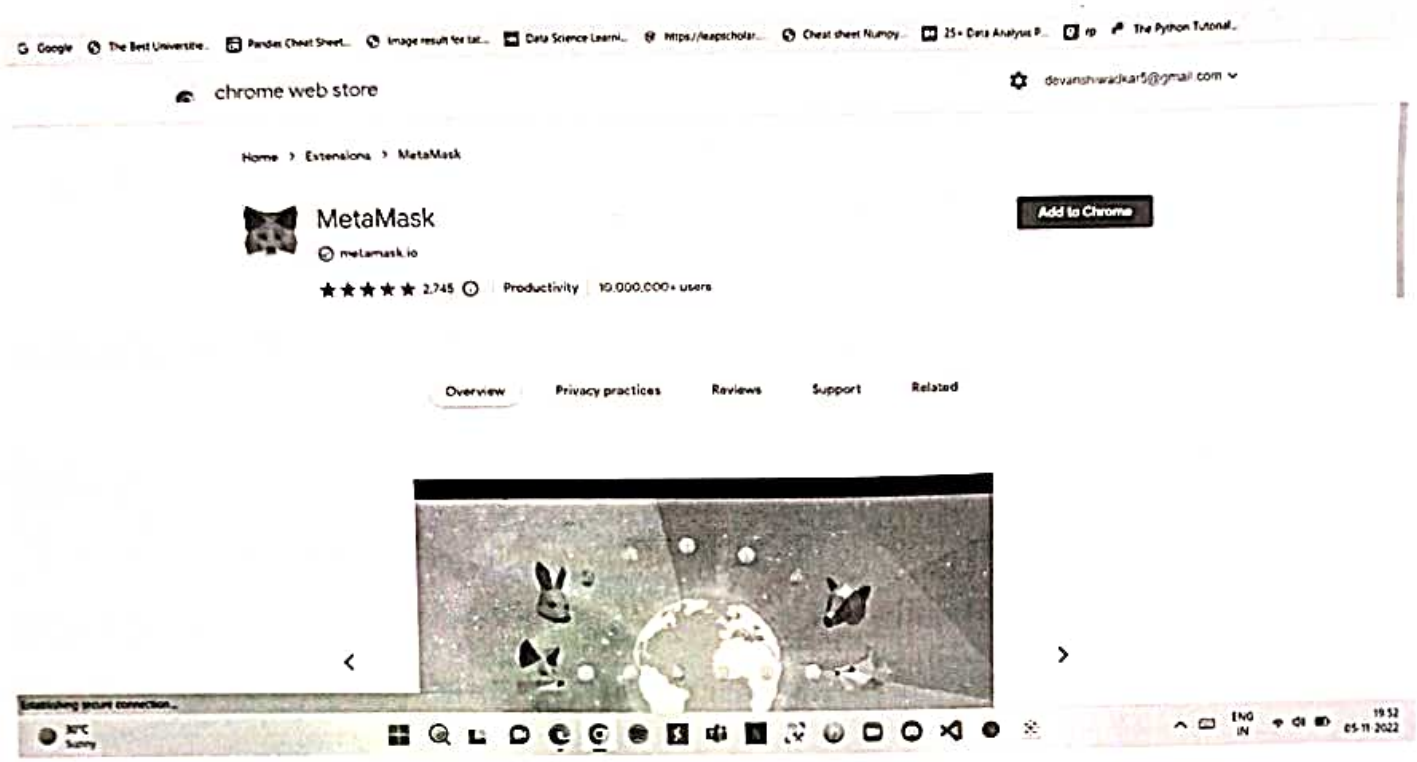
Step 5: Once installation is complete this page will be displayed. Click on the *Get Started* button.

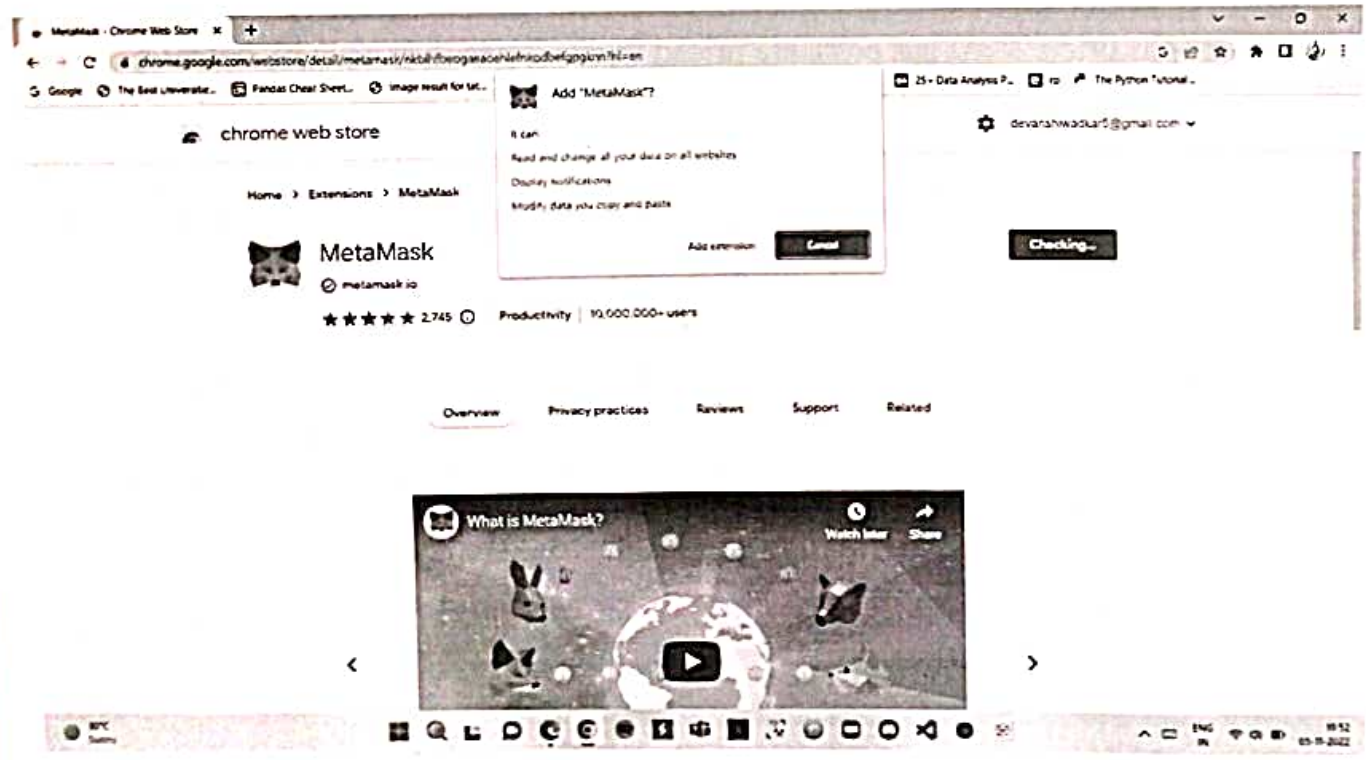
ETHER PER TRANSACTIONS

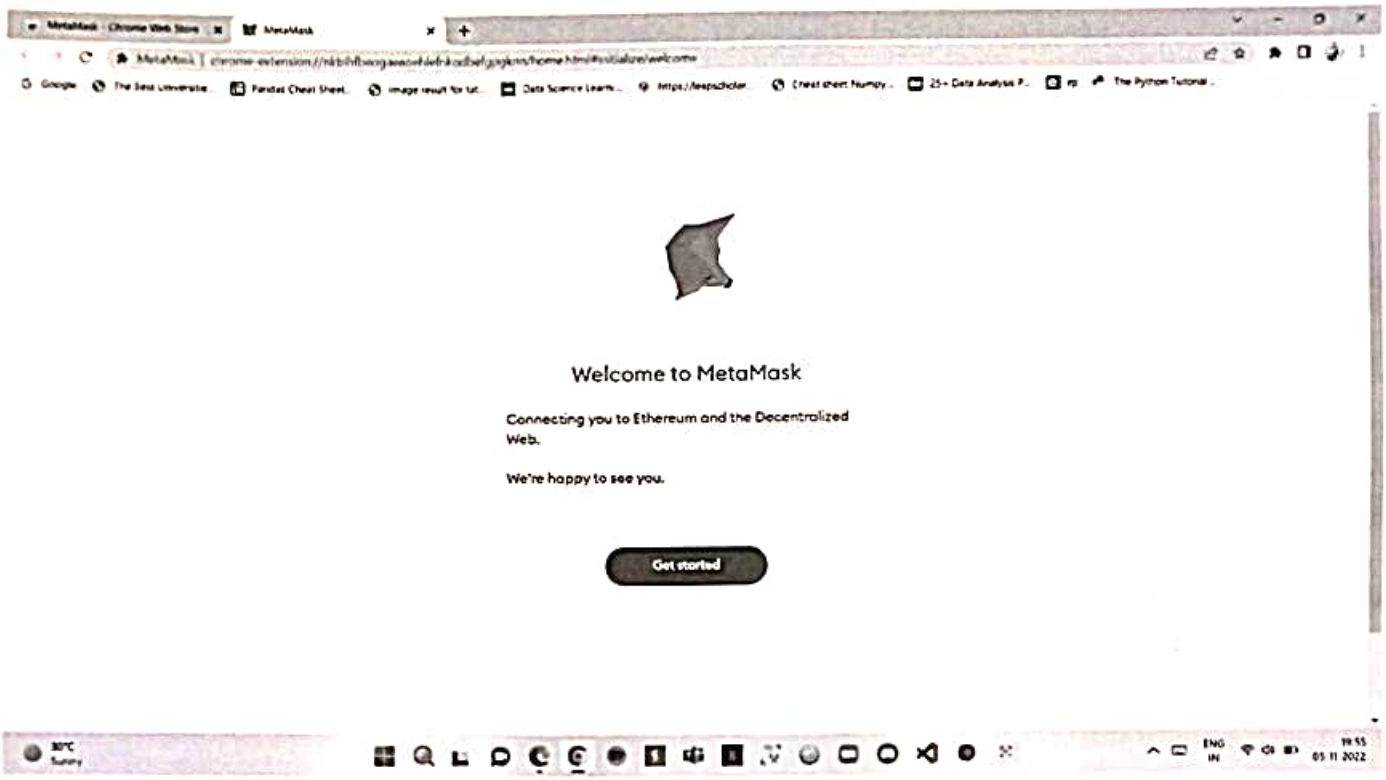
Practically all actions on the Ethereum blockchain require gas in order to be executed. Paid in Ethereum's native coin ether (ETH), this transaction fee on Ethereum is referred to as the gas fee — or gas price. Most gas costs are priced in gwei, which is a small denomination of ETH; 1 ETH equals 1 billion gwei. Gas is used to pay for ETH transactions, token minting, executing smart contracts, and powering decentralized applications (dApps). In August 2021, a network fork implemented a "base fee + tip" structure to create a more predictable and adaptable gas fee marketplace.

CONCLUSION :-

Hence we have successfully installed Metamask and studied how we spend ethers per transactions.


chrome web store
devanshwadkar5@gmail.com
Home > Extensions > MetaMask
MetaMask
metamask.io
2,745
Productivity
10,000,000+ users
Add to Chrome
Overview
Privacy practices
Reviews
Support
Related


chrome web store
Add "MetaMask"?
It can
Read and change all your data on all websites
Display notifications
Modify data you copy and paste
Add extension
Cancel
devanshwadkar5@gmail.com
Home > Extensions > MetaMask
MetaMask
metamask.io
2,745
Productivity
10,000,000+ users
Checking...
Overview
Privacy practices
Reviews
Support
Related
What is MetaMask?
Watch later
Share


Welcome to MetaMask
Connecting you to Ethereum and the Decentralized Web.
We're happy to see you.
Get started

LAB ASSIGNMENT 2

AIM:- Create your own wallet using Metamask for crypto transactions.

REQUIREMENTS :- Windows 11/ Linux , Metamask extension added to chrome and setup done.

THEORY :-

A blockchain wallet is a digital wallet that allows users to store and manage their Bitcoin, Ether, and other cryptocurrencies. Blockchain Wallet can also refer to the wallet service provided by Blockchain, a software company founded by Peter Smith and Nicolas Cary. A blockchain wallet allows transfers in cryptocurrencies and the ability to convert them back into a user's local currency.

E-wallets allow individuals to store cryptocurrencies and other digital assets. In the case of Blockchain Wallet, users can manage their balances of various cryptocurrencies such as the well-known Bitcoin and Ether as well as stellar, Tether, and Paxos Standard.

Creating an e-wallet with Blockchain Wallet is free, and the account setup process is done online. Individuals must provide an email address and password that will be used to manage the account, and the system will send an automated email requesting that the account be verified.

How a Blockchain Wallet Works?

Users can send a request to another party for a specific amount of bitcoin or other crypto-assets, and the system generates a unique address that can be sent to a third party or converted into a Quick Response code or QR code for short. A QR code is similar to a barcode, which stores financial information and can be read by a digital device.

A unique address is generated each time the user makes a request. Users can also send crypto-assets when someone provides them with a unique address.The send-and-receive process is similar to sending or receiving funds through PayPal but uses cryptocurrency

instead. PayPal is an online payment provider that acts as a go-between for customers and their banks and credit cards by facilitating online transfers through financial institutions.

Users can also exchange Bitcoin for other crypto-assets and visa-versa, known as swapping. This practice is an easy way to switch out crypto without leaving the security of the Blockchain Wallet. Users are shown a quote indicating how much they will receive based on the current exchange rate, with the rate changing depending on how long the user takes to complete the transaction.Swaps should take a couple of hours while the transactions are added to each currency's blockchain. However, if it takes longer than six hours, users should contact customer support.

Blockchain Wallet only allows six crypto-assets for swapping: Bitcoin, Ethereum, Bitcoin Cash, Stellar Lumens, Tether, USD Digital, Wrapped-DGLD.

How to create a wallet?

Step 1 :- This is the first time creating a wallet, so click the *Create a Wallet* button. If there is already a wallet then import the already created using the *Import Wallet* button.

Step 2:- Click *I Agree* button to allow data to be collected to help improve MetaMask or else click the *No Thanks* button. The wallet can still be created even if the user will click on the *No Thanks* button.

Step 3 :- 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*.

Step 4 :- Click on the dark area which says *Click here to reveal secret words* to get your secret phrase.

Step 5:- 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.

Step 6 :- Click the buttons respective to the order of the words in your seed phrase. In other words, type the seed phrase using the button on the screen. If done correctly the *Confirm* button should turn blue.

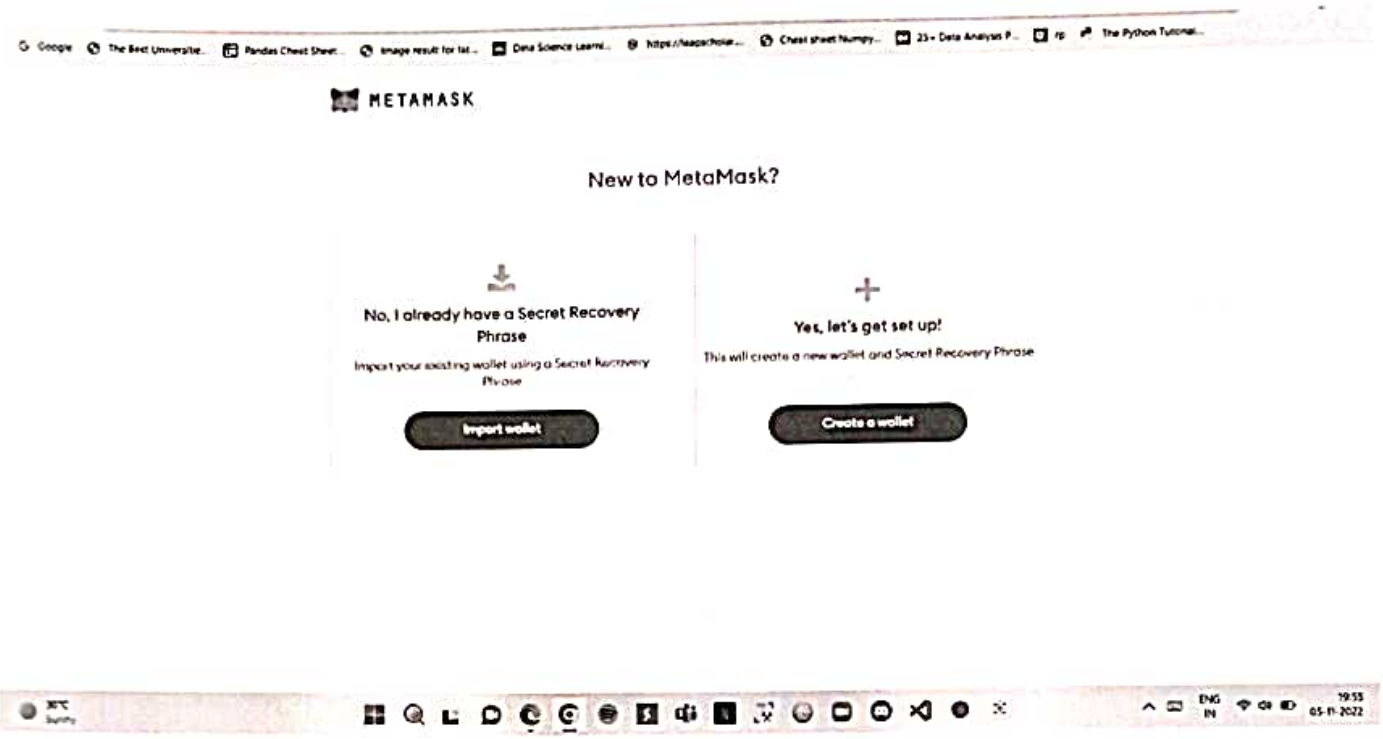
Step 7:- Click the *Confirm* button. Please follow the tips mentioned.

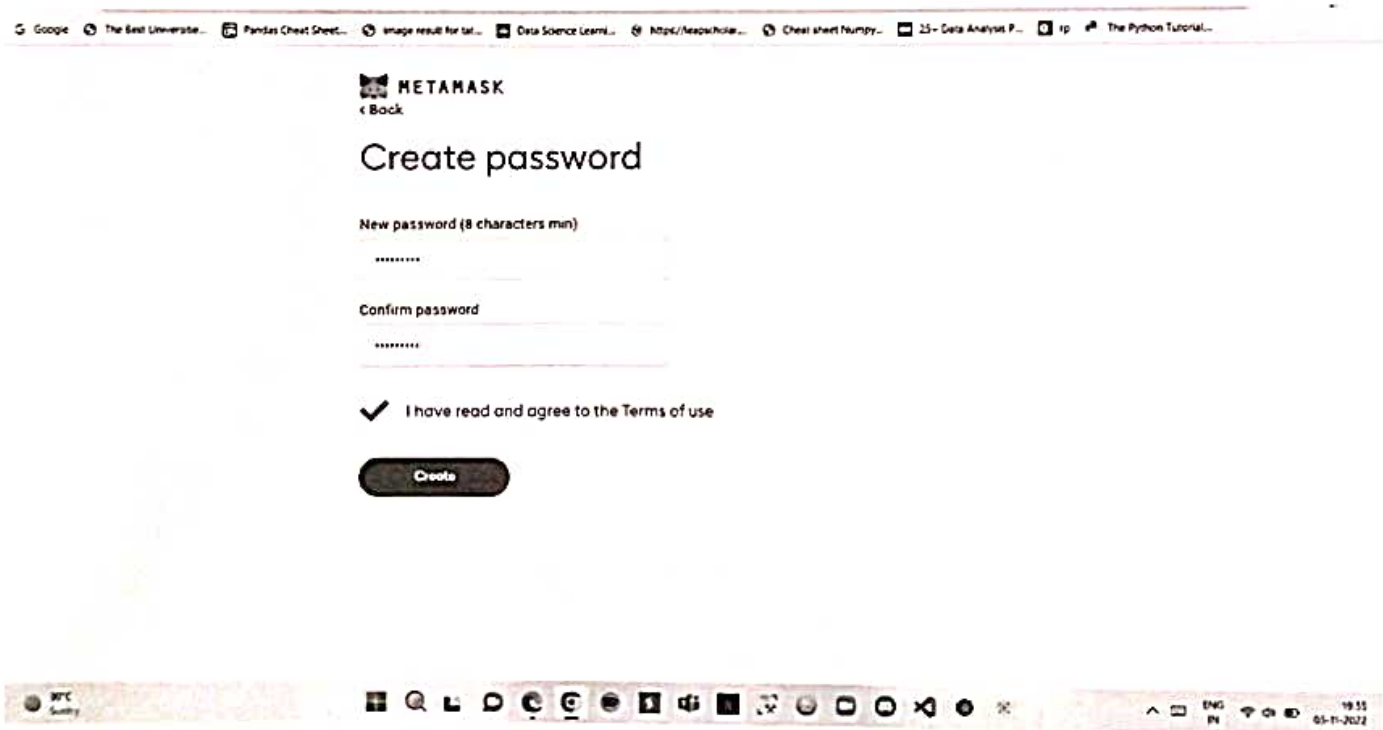
Step 8:- One can see the balance and copy the address of the account by clicking on the *Account 1* area.

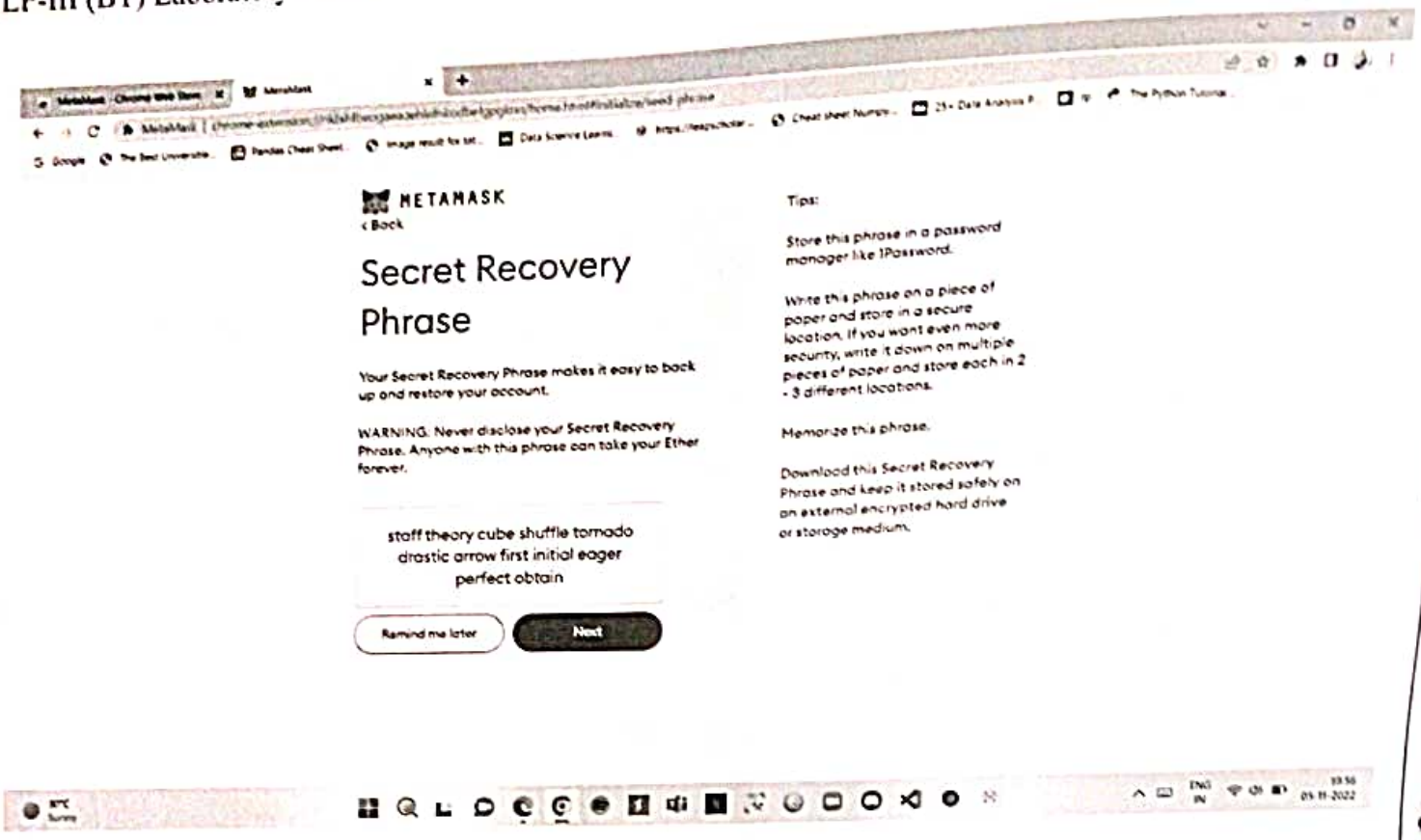
Step 9 :- One can access MetaMask in the browser by clicking the Foxface icon on the top right. If the Foxface icon is not visible, then click on the puzzle piece icon right next to it.

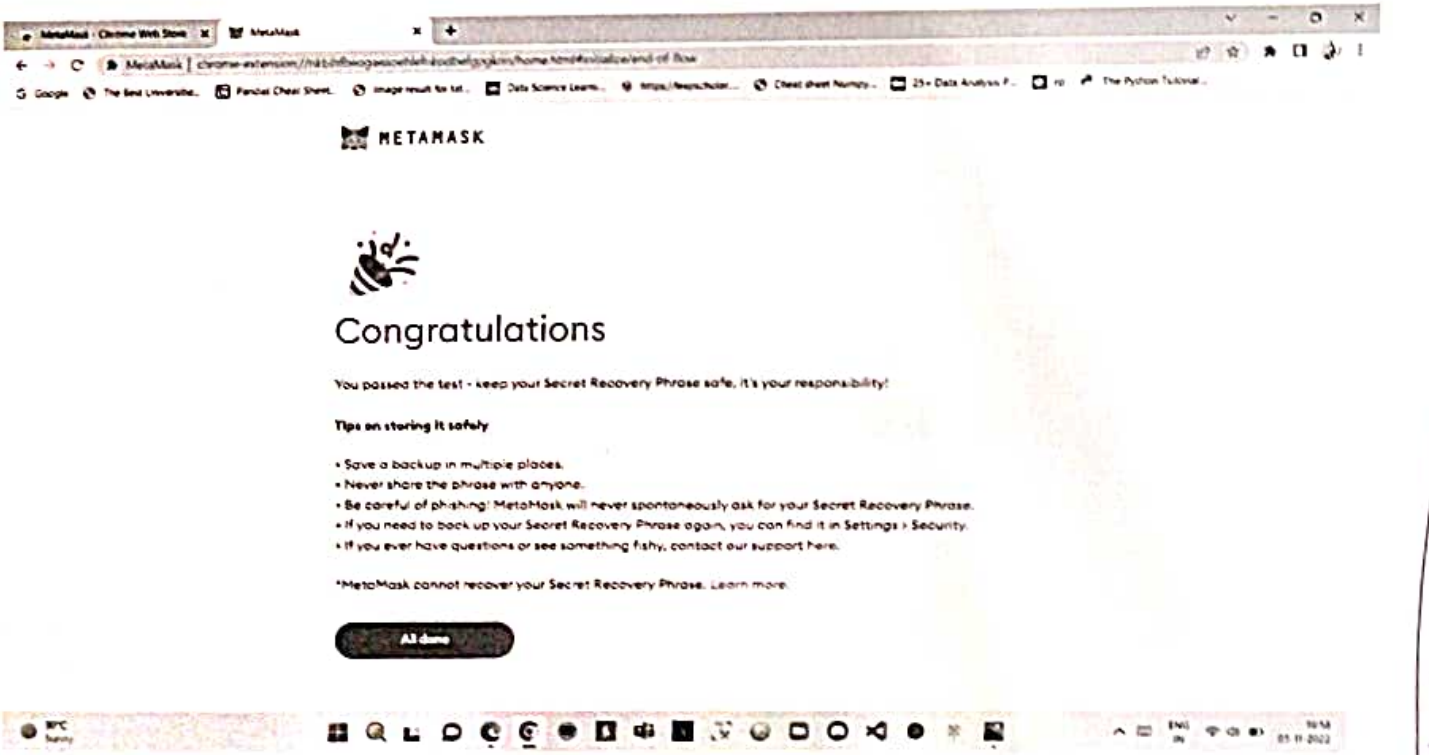
CONCLUSION :-

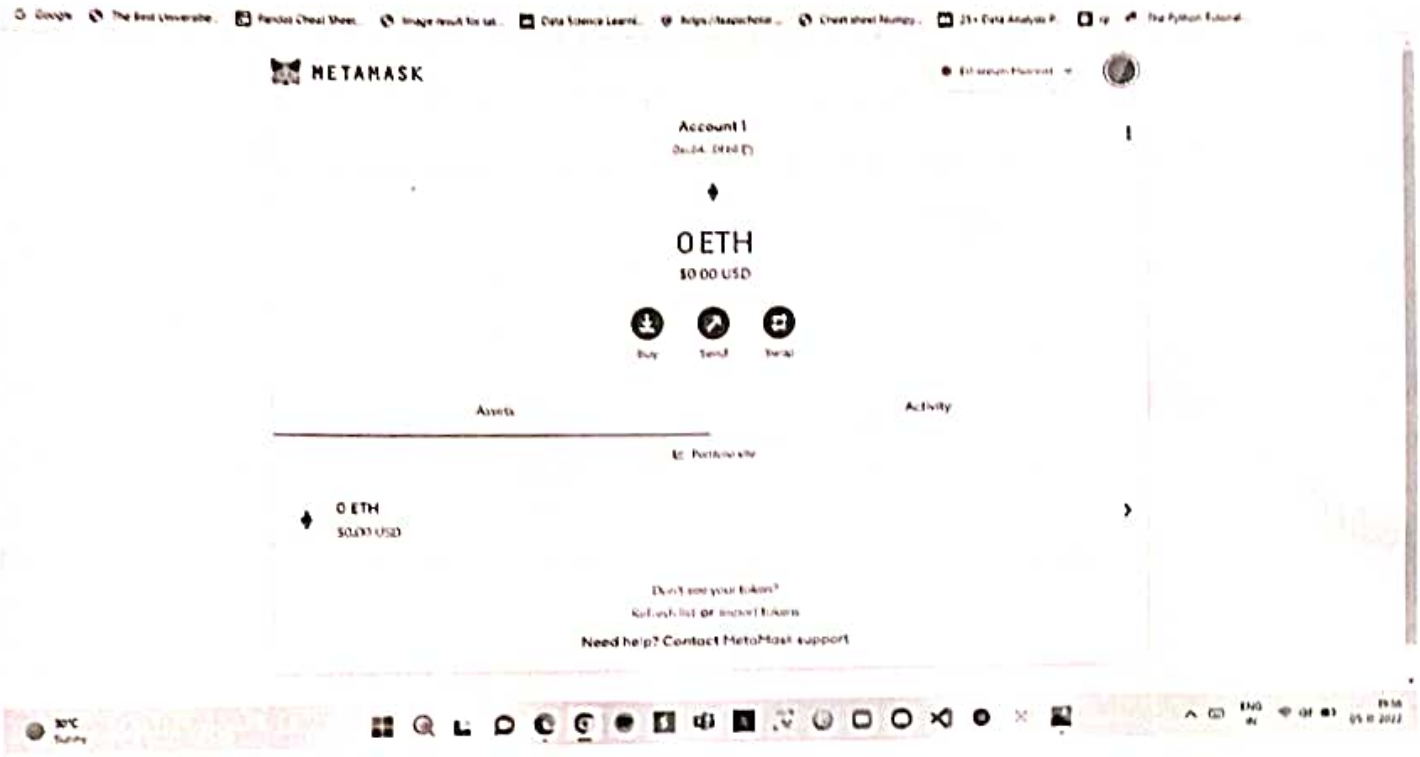
Hence, we have successfully created a wallet using Metamask for crypto-transactions.


METAMASK
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


METAMASK
‹ Back
Create password
New password (8 characters min)
Confirm password
I have read and agree to the Terms of use
Create


METAMASK
‹ Back
Secret Recovery Phrase
Your Secret Recovery Phrase makes it easy to back up and restore your account.
WARNING: Never disclose your Secret Recovery Phrase. Anyone with this phrase can take your Ether forever.
staff theory cube shuffle tornado drastic arrow first initial eager perfect obtain
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 Recovery Phrase and keep it stored safely on an external encrypted hard drive or storage medium.


METAMASK
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


METAMASK
Account 1
0 ETH
$0.00 USD
Buy
Send
Swap
Assets
Activity
0 ETH
Need help? Contact MetaMask support

LAB ASSIGNMENT 3

AIM:- Write a smart contract on a test network, for Bank account of a customer for following operations:

- **Deposit money**
- **Withdraw Money**
- **Show balance**

REQUIREMENTS :- Windows 11/Linux,Remix IDE installed,Solidity compiler.

THEORY :-

Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met. They typically are used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary's involvement or time loss. They can also automate a workflow, triggering the next action when conditions are met.

How smart contracts work ?

Smart contracts work by following simple "if/when...then..." statements that are written into code on a blockchain. A network of computers executes the actions when predetermined conditions have been met and verified. These actions could include releasing funds to the appropriate parties, registering a vehicle, sending notifications, or issuing a ticket. The blockchain is then updated when the transaction is completed. That means the transaction cannot be changed, and only parties who have been granted permission can see the results.

Within a smart contract, there can be as many stipulations as needed to satisfy the participants that the task will be completed satisfactorily. To establish the terms, participants must determine how transactions and their data are represented on the blockchain, agree on the "if/when...then..." rules that govern those transactions, explore all possible exceptions, and define a framework for resolving disputes.

Then the smart contract can be programmed by a developer – although increasingly, organizations that use blockchain for business provide templates, web interfaces, and other online tools to simplify structuring smart contracts.

Benefits of smart contracts :-

Speed, efficiency and accuracy

Once a condition is met, the contract is executed immediately. Because smart contracts are digital and automated, there's no paperwork to process and no time spent reconciling errors that often result from manually filling in documents.

Trust and transparency

Because there's no third party involved, and because encrypted records of transactions are shared across participants, there's no need to question whether information has been altered for personal benefit.

Security

Blockchain transaction records are encrypted, which makes them very hard to hack. Moreover, because each record is connected to the previous and subsequent records on a distributed ledger, hackers would have to alter the entire chain to change a single record.

Savings

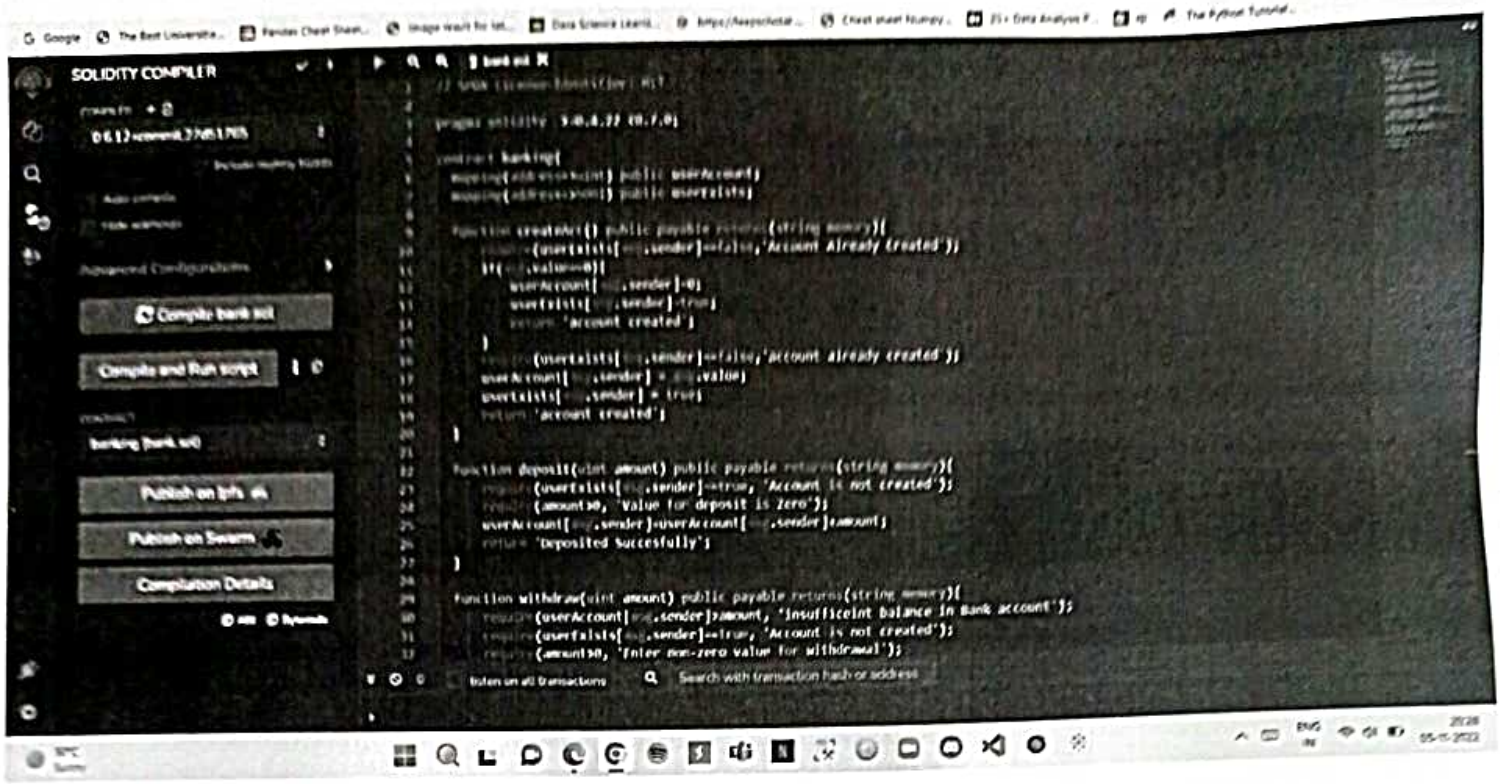
Smart contracts remove the need for intermediaries to handle transactions and, by extension, their associated time delays and fees.

Applications of smart contracts :-

- Safeguarding the efficacy of medications
- Increasing trust in retailer-supplier relationships
- Making international trade faster and more efficient

CONCLUSION:-

Hence we have successfully written and executed the smart contract for a bank account of customer.


SOLIDITY COMPILER
Compile bank.sol
Compile and Run script
Publish on Ipfs
Publish on Swarm
Compilation Details
bank.sol
contract banking{
mapping(address=>uint) public userAccount;
mapping(address=>bool) public userExists;
function createAcc() public payable returns(string memory){
require(userExists[msg.sender]==false,'Account Already Created');
if(msg.value==0){
userAccount[msg.sender]=0;
userExists[msg.sender]=true;
return 'account created';
}
require(userExists[msg.sender]==false,'account already created');
userAccount[msg.sender] = msg.value;
userExists[msg.sender] = true;
return 'account created';
}
function deposit(uint amount) public payable returns(string memory){
require(userExists[msg.sender]==true, 'Account is not created');
require(amount>0, 'Value for deposit is zero');
userAccount[msg.sender]=userAccount[msg.sender]+amount;
return 'Deposited Succesfully';
}
function withdraw(uint amount) public payable returns(string memory){
require(userAccount[msg.sender]>amount, 'insufficeint balance in Bank account');
require(userExists[msg.sender]==true, 'Account is not created');
require(amount>0, 'Enter non-zero value for withdrawal');
Search with transaction hash or address


SOLIDITY COMPILER
Compile bank.sol
Compile and Run script
Publish on Ipfs
Publish on Swarm
Compilation Details
bank.sol
userAccount[msg.sender]=userAccount[msg.sender]-amount;
msg.sender.transfer(amount);
return 'withdrawal Succesful';
}
function TransferAmount(address payable userAddress, uint amount) public returns(string memory){
require(userAccount[msg.sender]>amount, 'insufficeint balance in Bank account');
require(userExists[msg.sender]==true, 'Account is not created');
require(userExists[userAddress]==true, 'to Transfer account does not exists in bank accounts ');
require(amount>0, 'Enter non-zero value for sending');
userAccount[msg.sender]=userAccount[msg.sender]-amount;
userAccount[userAddress]=userAccount[userAddress]+amount;
return 'transfer succesfully';
}
function sendAmount(address payable toAddress , uint256 amount) public payable returns(string memory){
require(amount>0, 'Enter non-zero value for withdrawal');
require(userExists[msg.sender]==true, 'Account is not created');
require(userAccount[msg.sender]>amount, 'insufficeint balance in Bank account');
userAccount[msg.sender]=userAccount[msg.sender]-amount;
toAddress.transfer(amount);
return 'transfer success';
}
function userAccountBalance() public view returns(uint){
return userAccount[msg.sender];
}
function accountExist() public view returns(bool){
return userExists[msg.sender];
}
}
Search with transaction hash or address

LAB ASSIGNMENT 4

AIM:- 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.

REQUIREMENTS :- Windows 11 /Linux, Remix IDE, Solidity compiler installed, knowledge of solidity.

THEORY :-

SOLIDITY :-

Solidity is an object-oriented, high-level language for implementing smart contracts. Smartcontractsareprogramswhichgovernthebehaviorofaccountswithin theEthereumstate.

Solidity is a curly-bracket language designed to target the Ethereum Virtual Machine(EVM). It is influenced by C++, Python and JavaScript. You can find more details aboutwhichlanguagesSolidityhasbeeninspiredbyinthelanguageinfluencessection.

Solidityisstaticallytyped,supportsinheritance,librariesandcomplexuser-definedtypes

among other features.

With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

When deploying contracts, you should use the latest released version of Solidity. Apart from exceptional cases, only the latest version receives security fixes. Furthermore, breaking changes as well as new features are introduced regularly. We currently use a 0.y.z version number to indicate this fast pace of change.

What is Ethereum?

Ethereum is a decentralized ie. blockchain platform that runs smart contracts i.e. applications that run exactly as programmed without any possibility of downtime, censorship, fraud or third-party interference.

The Ethereum Virtual Machine (EVM)

The Ethereum Virtual Machine, also known as EVM, is the runtime environment for smart contracts in Ethereum. The Ethereum Virtual Machine focuses on providing security and executing untrusted code by computers all over the world.

The EVM specialized in preventing Denial-of-service attacks and ensures that programs do not have access to each other's state, ensuring communication can be established without any potential interference.

The Ethereum Virtual Machine has been designed to serve as a runtime environment for smart contracts based on Ethereum.

WhatisSmartContract?

A smart contract is a computer protocol intended to digitally facilitate, verify, or enforcethe negotiation or performance of a contract. Smart contracts allow the performance ofcredible transactions without third parties. These transactions are trackableandirreversible.

The concept of smart contracts was first proposed by Nick Szabo in 1994. Szabo is a legalscholarandcryptographerknownforlayingthegroundworkfordigitalcurrency.

SOLIDITYELEMENTS:-

STRUCTURES:

Struct types are used to represent a record. Suppose you want to keep track of your booksinalibrary.Youmightwanttotrackthefollowingattributesabouteachbook−

- Title
- Author
- Subject
- BookID

ARRAYS:

Arrayisadatastructure,whichstoresafixed-sizesequentialcollectionofelementsofthesametype.Anarrayisusedtostoreacollectionofdata,butitisoftenmoreusefultothink

of an array as a collection of variables of the same type.

Instead of declaring individual variables, such as number0, number1, ..., and number99, you declare one array variable such as numbers and use numbers[0], numbers[1], and ..., numbers[99] to represent individual variables. A specific element in an array is accessed by an index.

In Solidity, an array can be of compile-time fixed size or of dynamic size. For storage array, it can have different types of elements as well. In case of memory array, element type can not be mapping and in case it is to be used as function parameter then element type should be an ABI type.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

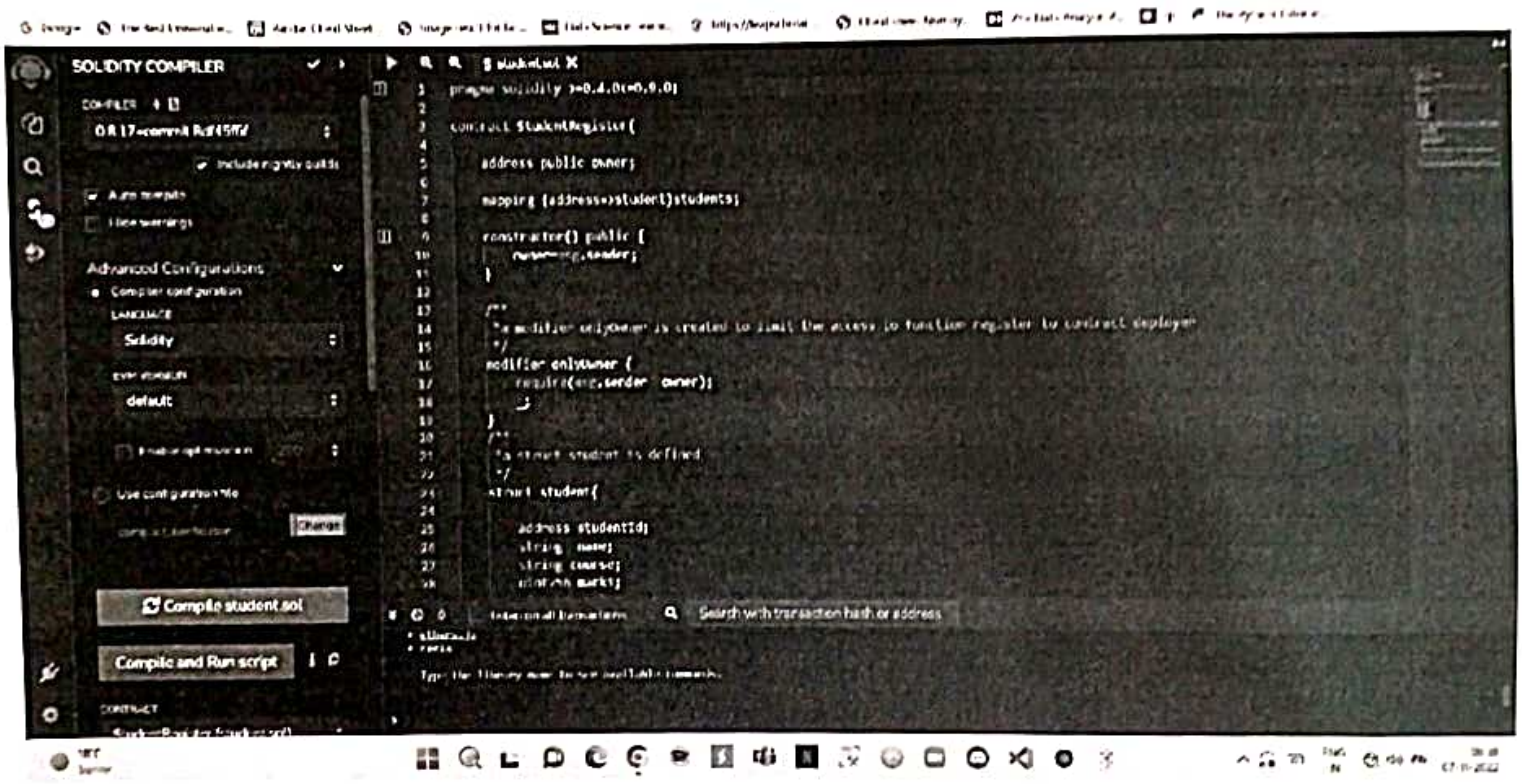
FALLBACK :-

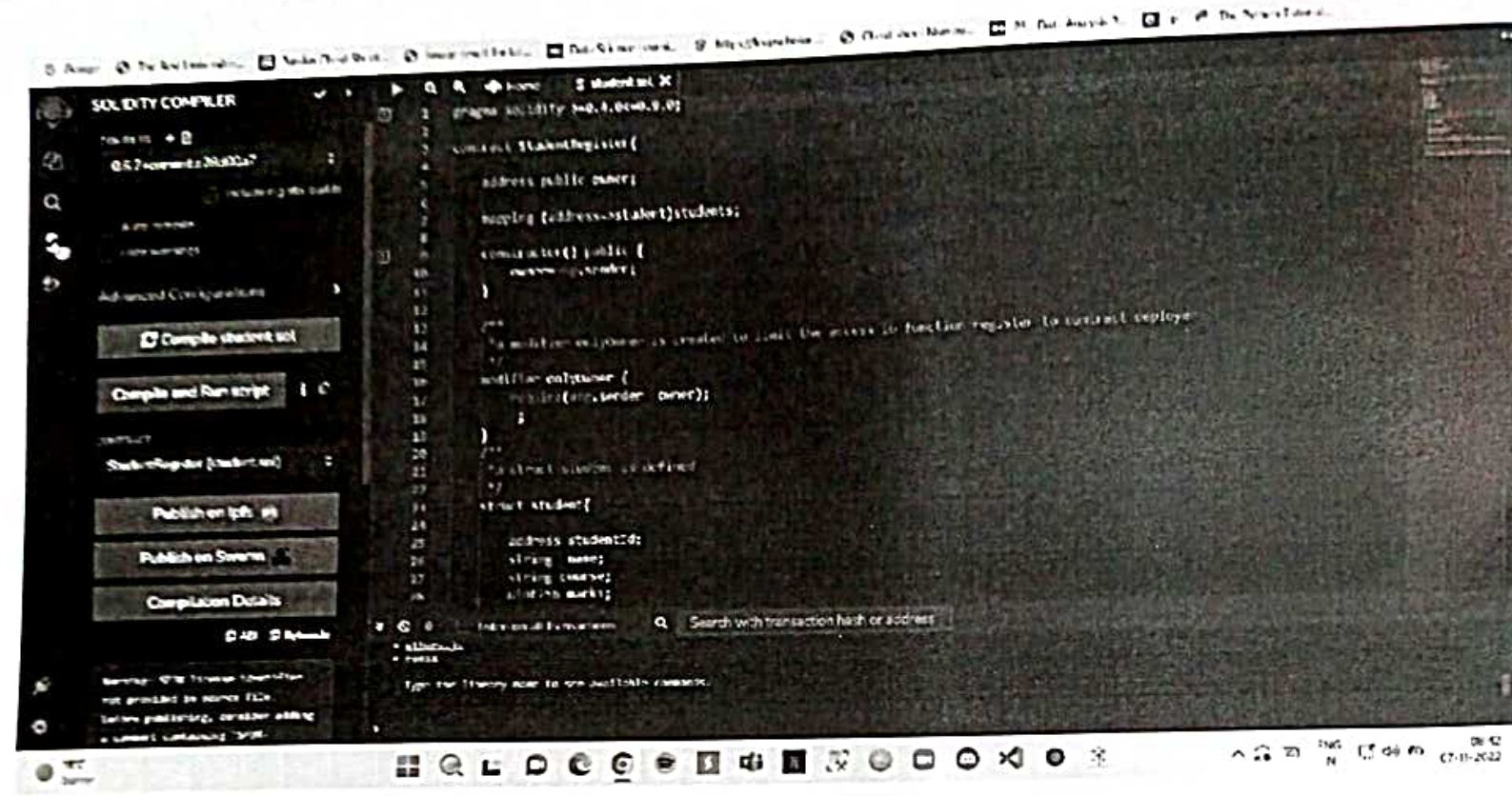
Fallback function is a special function available to a contract. It has following features –

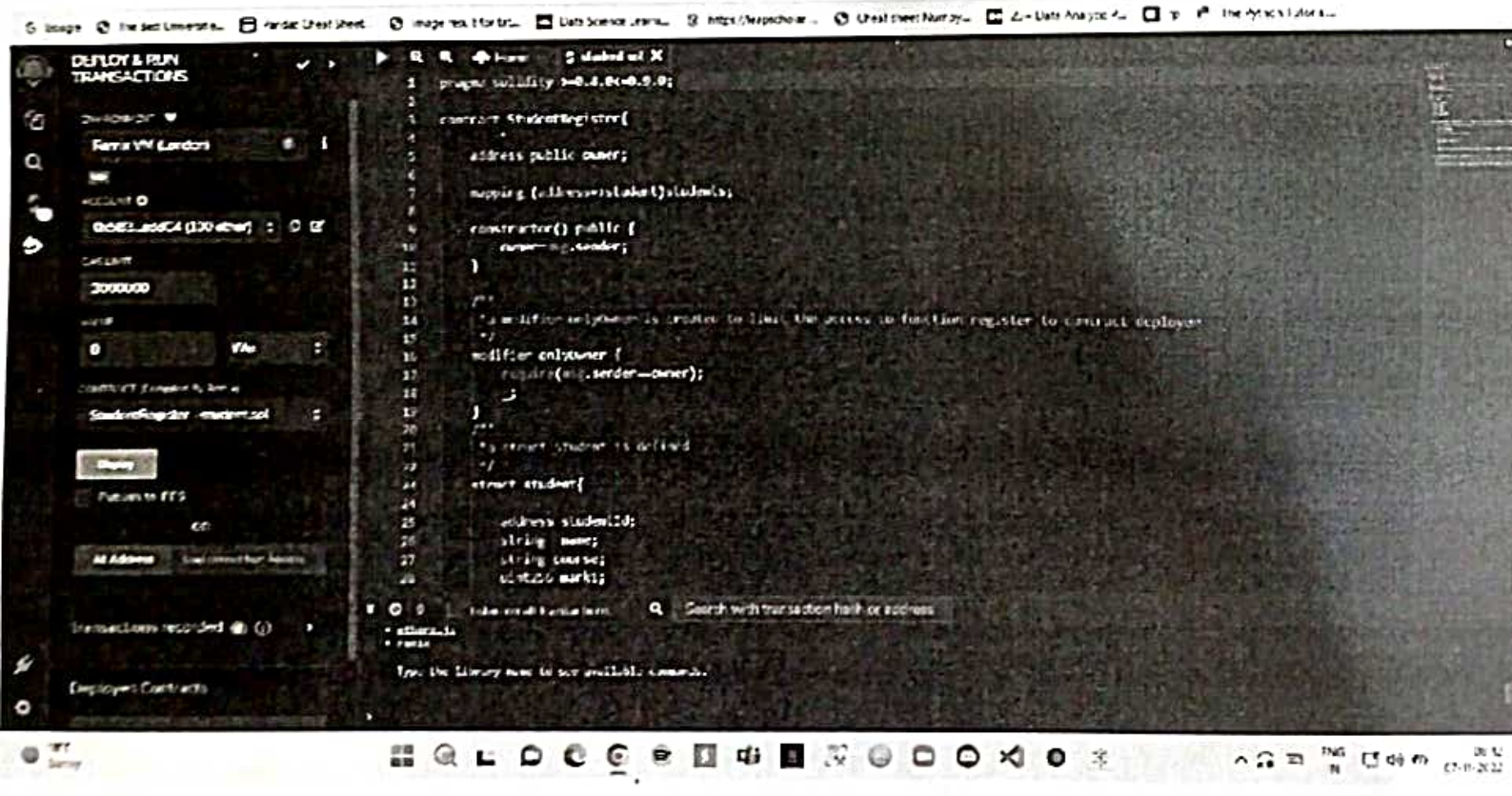
1. Itiscalledwhenanon-existentfunctioniscalledonthecontract.
2. Itisrequiredtobemarkedexternal.
3. Ithasnoname.
4. Ithasnoarguments
5. Itcannotreturnanything.
6. Itcanbedefinedonepercontract.
7. Ifnotmarkedpayable,itwillthrowexceptionifcontractreceivesplainetherwithoutdata.

CONCLUSION :-

Hence we have successfully implemented the smart contract and deployed it on the ethereum blockchain network.




SOLIDITY COMPILER
Compile student.sol
Compile and Run script
Publish on Ipfs
Publish on Swarm
Compilation Details
pragma solidity >=0.4.0<0.9.0;
contract StakeholRegister{
address public owner;
mapping (address=>stakeholder)students;
constructor() public {
}
/**
* a modifier onlyowner is created to limit the access to function register to contract deployer
*/
modifier onlyowner {
require(msg.sender == owner);
_;
}
/**
* a struct student is defined
*/
struct student{
address studentID;
string name;
string course;
uint256 marks;
Search with transaction hash or address


DEPLOY & RUN TRANSACTIONS
pragma solidity >=0.4.0<0.9.0;
contract StudentRegister{
address public owner;
mapping (address=>stakeholder)students;
constructor() public {
}
/**
* a modifier onlyowner is created to limit the access to function register to contract deployer
*/
modifier onlyowner {
require(msg.sender==owner);
_;
}
/**
* a struct student is defined
*/
struct student{
address studentID;
string name;
string course;
uint256 marks;
Search with transaction hash or address