JSPM's
Jaywantrao Sawant College Of Engineering, Pune

Department of Computer Engineering

LABORATORY MANUAL

Program	Computer Engineering
Course	410246(2019)
Course Name	Laboratory Practice III
Class	BE Comp
Module Coordinator	Mr. Madhav D. Ingle
Course Coordinator	Mrs. Smita S. Wagh

Table of Contents

Vision of the Department

"To be a leading educational center grooming computer engineers to serve the society."

Mission of the Department

M1: To develop computer professionals by providing quality education.

M2: To assimilate academics, research and entrepreneurship skills to accomplish real word challenges.

Program Educational Objectives (PEOs):

PEO1: The graduates shall have an ability to identify, analyze & solve problems in computer engineering field using fundamental domain knowledge and programming tools

PEO2: The graduates shall have an ability to apply core technical competencies in diversified areas with good leadership and teamwork abilities.

PEO3: The graduates shall aspire for entrepreneurship, research and higher studies in computer engineering field

Program Outcomes (POs):

1. **Engineering knowledge:** The ability to apply knowledge of basic science, mathematics and engineering principles to solve computing and information processing problems.
2. **Problem analysis:** The ability to analyze problems and develop appropriate solutions.
3. **Design/development of solutions:** The ability to design solutions for complex engineering problems and design system components to understand the relationship between hardware and software systems.
4. **Conduct investigations of complex problems:** The ability to construct appropriate abstractions to manage complexity and to think creatively about new problems.
5. **Modern tool usage:** An ability to use current techniques, skills and tools necessary for computing practice.
6. **The engineer and society:** An ability to analyze the local and global impact of computing on individuals, organizations and society.
7. **Environment and sustainability:** The knowledge necessary to understand the impact of computing in a global, economic, environmental and societal context.
8. **Ethics:** The ability to understand professional ethics and responsibilities.
9. **Individual and team work:** The ability to work as an individual and in teams to achieve a common goal.
10. **Communication:** The ability to express ideas through written communication and oral presentations.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects.
12. **Life-long learning:** Recognize the need for, and develop an aptitude to be a life-long learner.

Program Specific Outcomes (PSOs):

PSO1: Specify design, develop and test the software systems in the areas of computer networking, database management, embedded systems, image processing, big data etc to satisfy user requirements.

PSO2: Analyze and optimize given algorithms or systems for performance improvements.

PSO3: Design hardware and software for concurrent and parallel programming.

Course Objectives

Understand and explore the working of Blockchain technology and its applications

Course Outcomes (COs): At the end of the course students will be able to-

CO6	Interpret the basic concepts in Blockchain technology and its applications

Mapping of COs with POs

Year	Course Name	Course Outcomes	Mapping with POs											
			1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
BE	LP-III	**CO6**	3	3	2	2	2	-	-	1	2	-	-	2

correlation levels 1, 2 or 3 - 1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High)

Mapping of COs with PSOs

Year	Course Name	Course Outcomes	Mapping with PSOs		
			PSO1	**PSO2**	**PSO3**
BE	LP-III	**CO6**	2	2	-

correlation levels 1, 2 or 3 - 1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High)

Experiment No 1

Title - Installation of Metamask and study spending Ether per transaction

Objective -

To study how to install Metamask from the browser, and how to spend ether pertransaction

Theory

Meta

mask

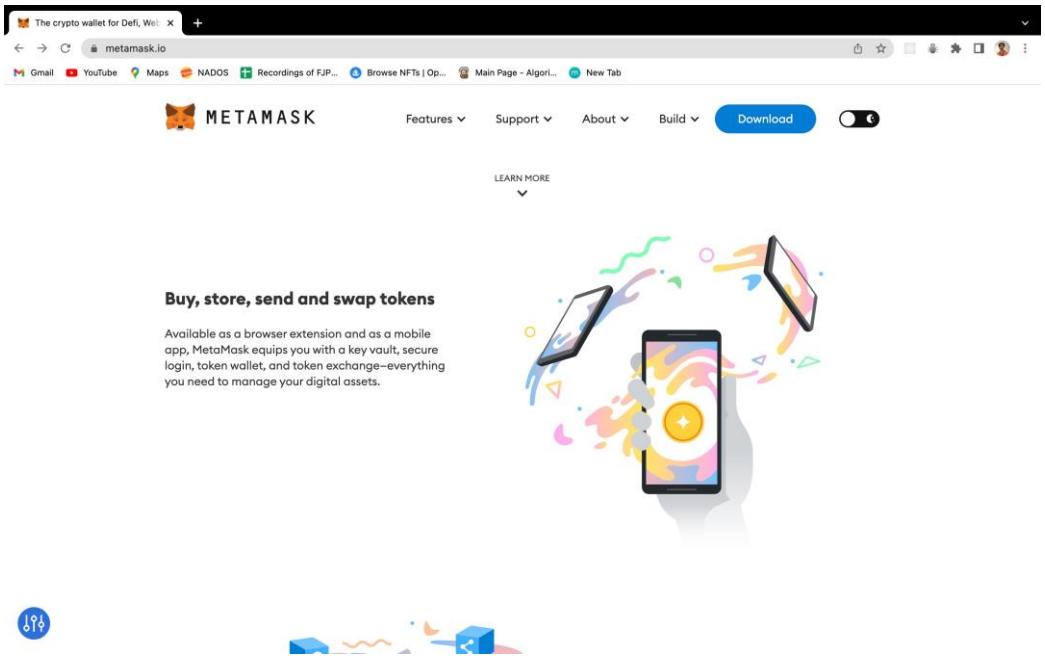
MetaMask is a free crypto wallet software that people can use to interact in the crypto world. It lets you buy, sell, and trade crypto assets for the Ethereum blockchain,much like how a real wallet lets you purchase items in the real world. It's been around since 2016. ConsenSys, the largest Ethereum development company in the world, launched it as a tool to simplify access to decentralized applications (DApps). It is possible on MetaMask to:

- Buy, receive, send and swap Ether (ETH), the main token of Ethereum
- Buy, receive, send and swap nonfungible tokens (NFTs) in marketplaces
- Connect to Ethereum dapps
- Connect to other crypto wallets
- Play blockchain-based games
- Access different networks such as the BNB Smart Chain and other tenets MetaMask is free to use and can be installed as an extension on internet browsers, Google Chrome, Firefox, Brave, and Edge, or downloaded as a smartphone application bothon iOS and Android. With over 30 million users, MetaMask is one of the most popular cryptocurrency wallets today.

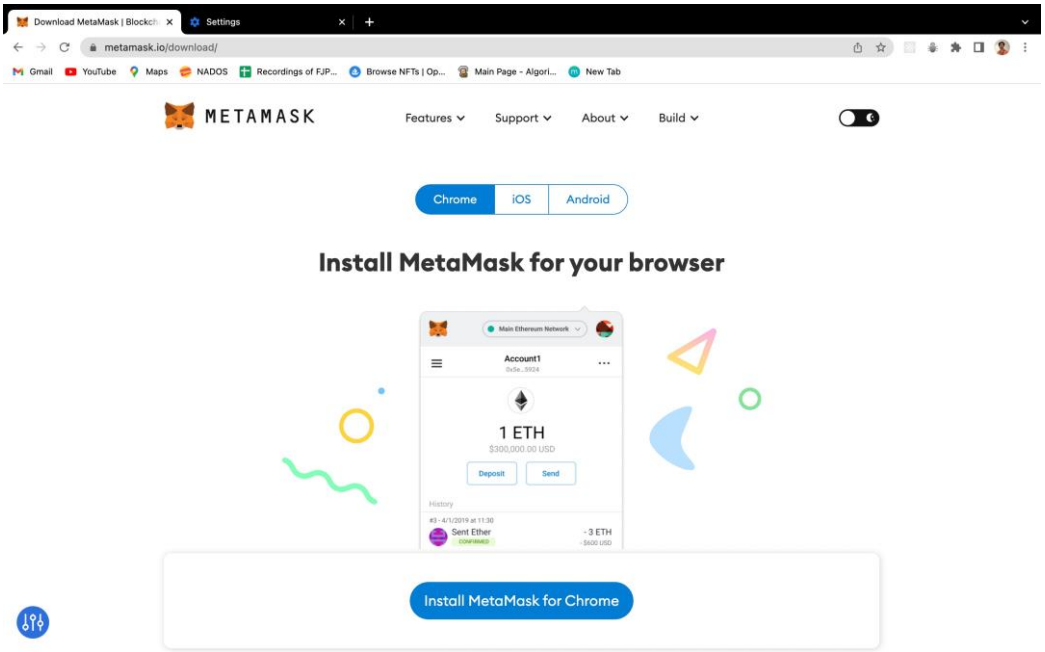
In this article, we'll go over some of the basics of using MetaMask, including how to create a wallet, make transfers and ways to secure it.

Installation of Metamask

Step 1 - Go to the site https://metamask.io/ and click on the “Download” button in the menu bar

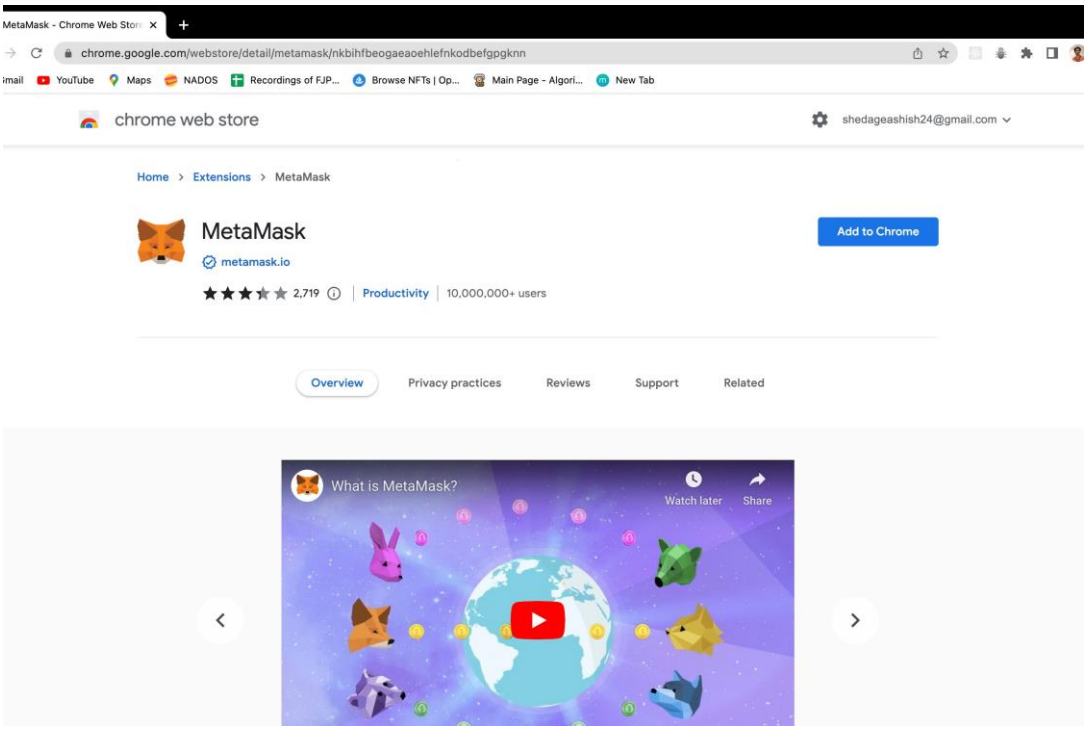


Step 2 - Click on the “Install metamask for chrome” you will be directed to the chrome web store

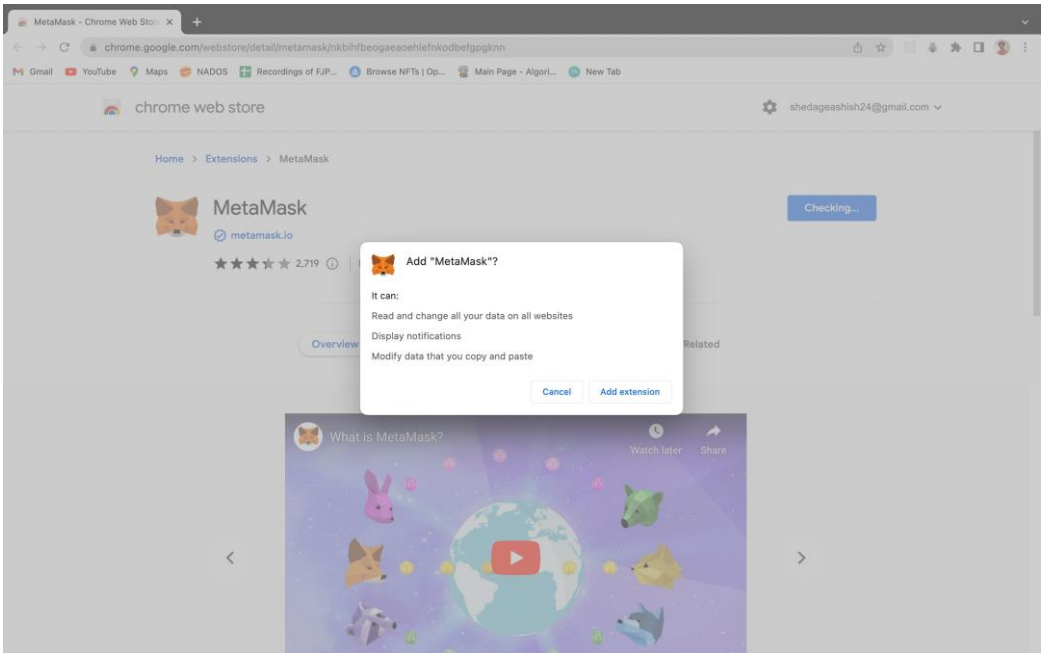


After clicked on “Download”

Step 3 - Click “Add to Chrome”



Step 4 - On the pop-up, click “Add Extension”



Ethereum

Ethereum operates on a decentralized computer Network or distributed ledgercalled a blockchain which manages and tracks the currency. It can be useful to thinkof a blockchain as running receipt of every transaction that's ever taken place in thecryptocurrency

Spending Ether per transaction

- Ethereum is a decentralized blockchain platform that established a peer-to-peer network that securely executes and verifies application code called a smart contract. Smart contracts allow the participant to transact with each other without a trusted central authority.

- The current Ethereum network can only support around 30 transactions per second which cause delays and congestion Ethereum 2.0 promises up to 100,000 transactions per second

- Ethereum's average transaction fee is at a current level of 0.9269, up from 0.69 yesterday and down from 3.870 one year ago. This is a change of 34.33% from yesterday and -76.05% from one year ago

- Ethereum blockchain is powered by its native cryptocurrency - ether(ETH) And enables developers to create new dApps through the use of smart contracts. The most common ETH-based cryptocurrencies are built on the ERC-20 token standard.

- ETH is generated by the Ethereum Network to reward mines for their work in adding blocks to the blockchain

Conclusion

Hence we have successfully installed the Metamask and learnt about the Ethereum per transaction

Questions

Q1) What is Blockchain as per your knowledge?

Q2) Can you explain the types of Blockchain?

Q3) What do you think are the key features of Blockchain?

Q4) What Is the Ethereum Network? Explain.

Q5) Who Is the Founder of Ethereum?

Q6) What Are the Real-World Use Cases of Ethereum?

Q7) What is Metamask ?

Q8) What Is the Difference Between Ethereum And Bitcoin Blockchain?

Experiment No 2

Title - Create your own wallet using Metamask for crypto transactions.

Objective -

To learn how to create your own wallet using Metamask for crypto transactions.

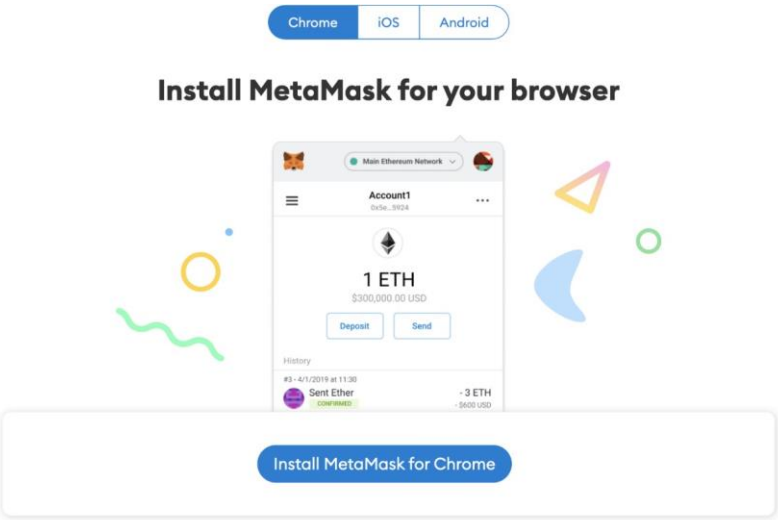
Theory -

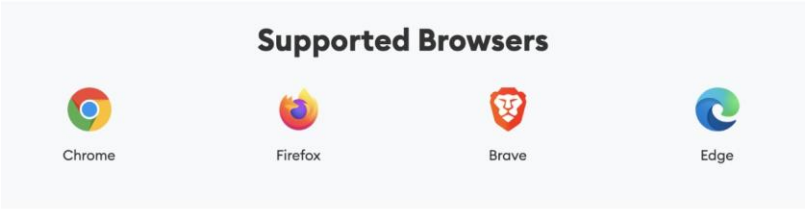
Metamask

MetaMask is a free crypto wallet software that people can use to interact in the crypto world. It lets you buy, sell, and trade crypto assets for the Ethereum blockchain,much like how a real wallet lets you purchase items in the real world.

Steps to create a wallet using Metamask

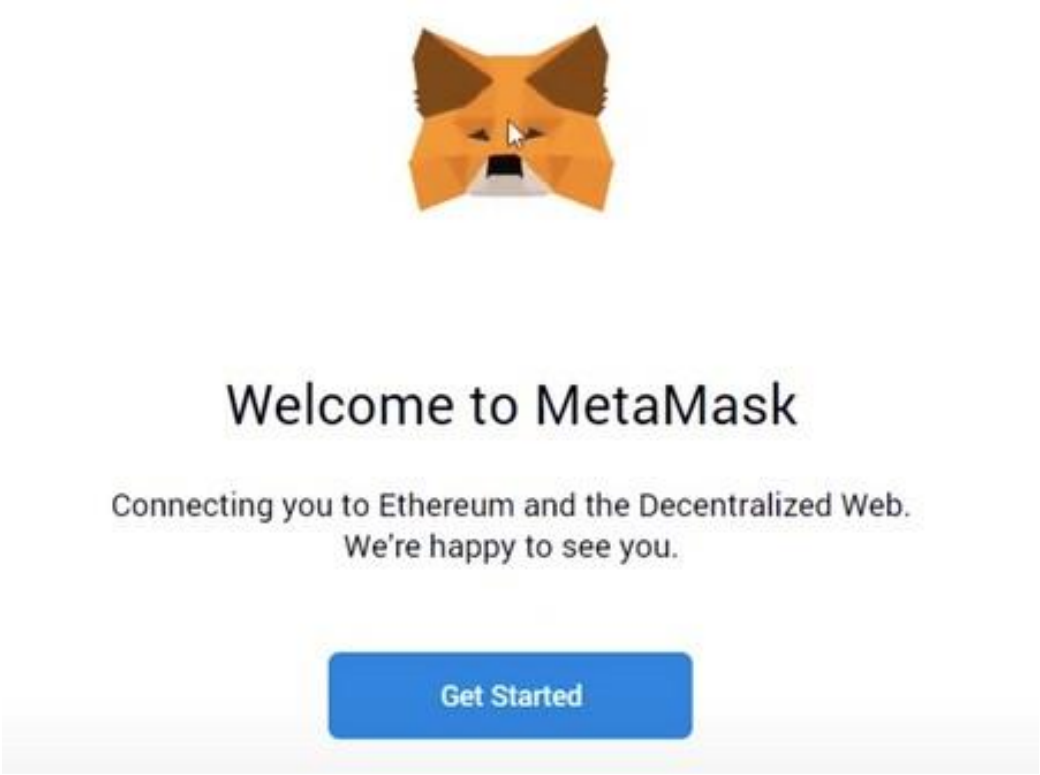
Step 1 - Go to https://metamask.io/ and click on "Download". Choose your preferred browser or mobile application and install the MetaMask extension.





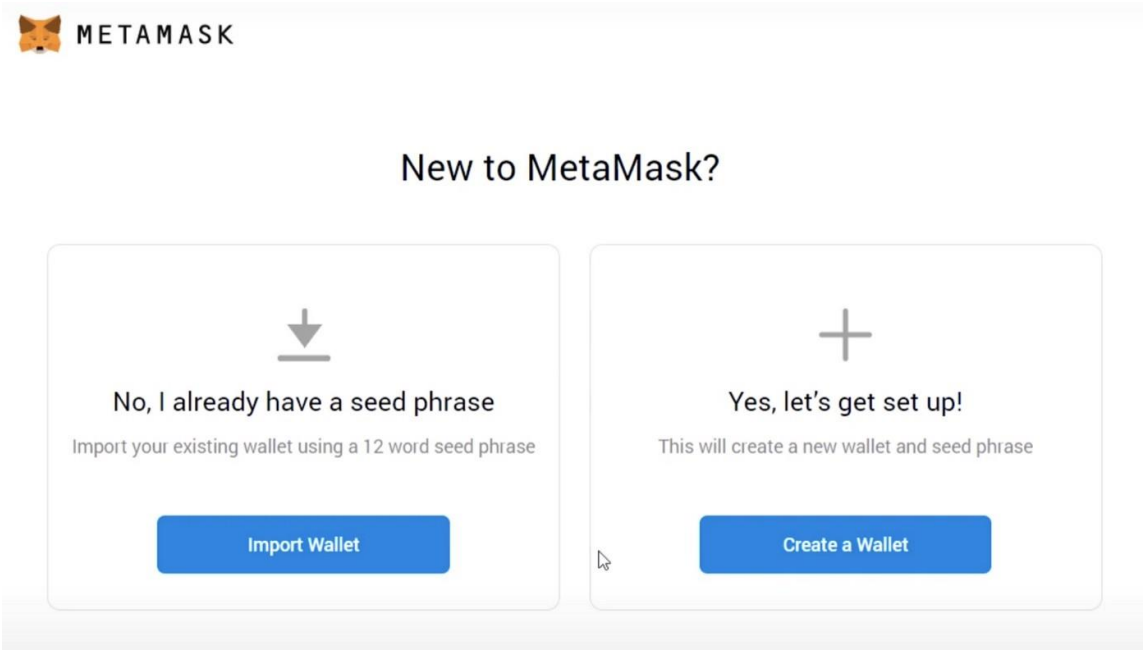
Step 2 - MetaMask wallet installation

Click on the MetaMask extension and click on “Get Started”.



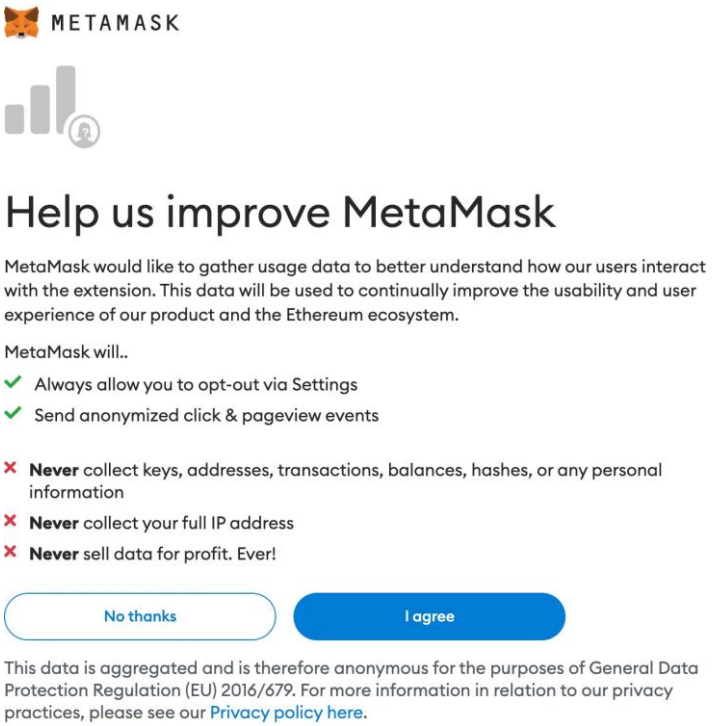
Caption

You can either import an existing wallet using the seed phrase or create a new one.



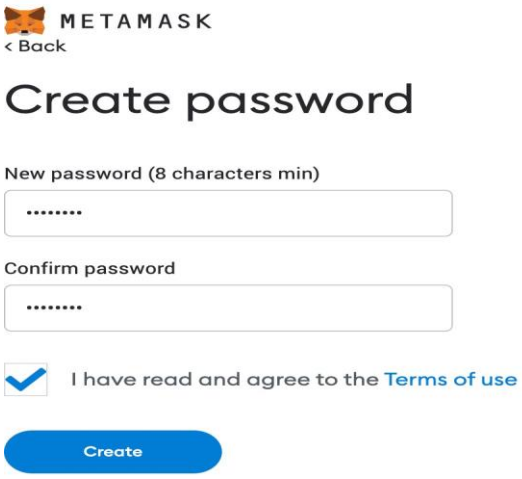
Step 3 - How to create a new MetaMask wallet

Click on “Create a Wallet” and on the next window click on “I agree” if you would like to help improve MetaMask or click on “No Thanks” to proceed.



Caption

Step 4 - Create a strong password for your wallet.

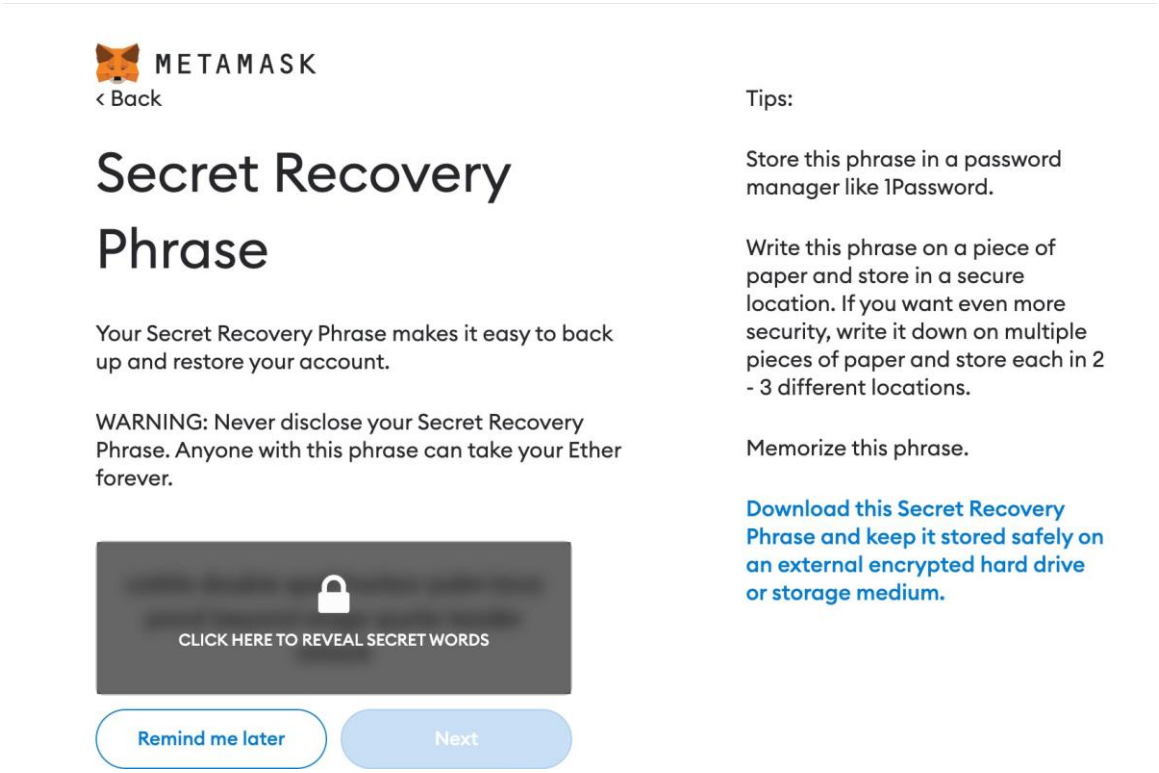


Step 5: Securely store the seed phrase for your wallet.

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.**

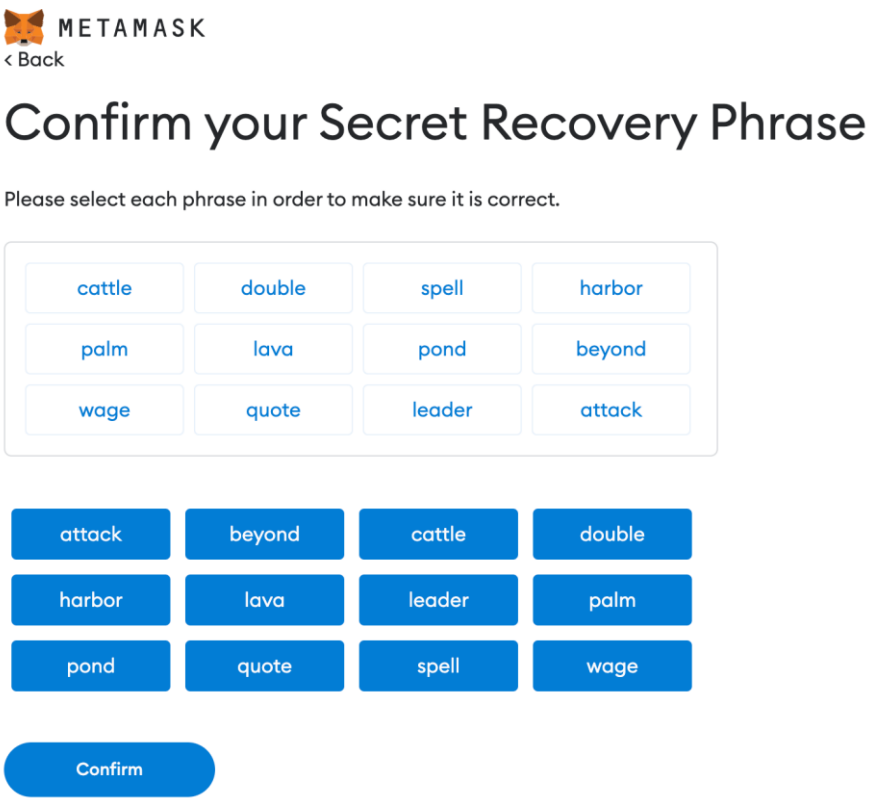
- Never share your seed phrase or your private key to anyone or any site, unless you want them to have full control over your funds.

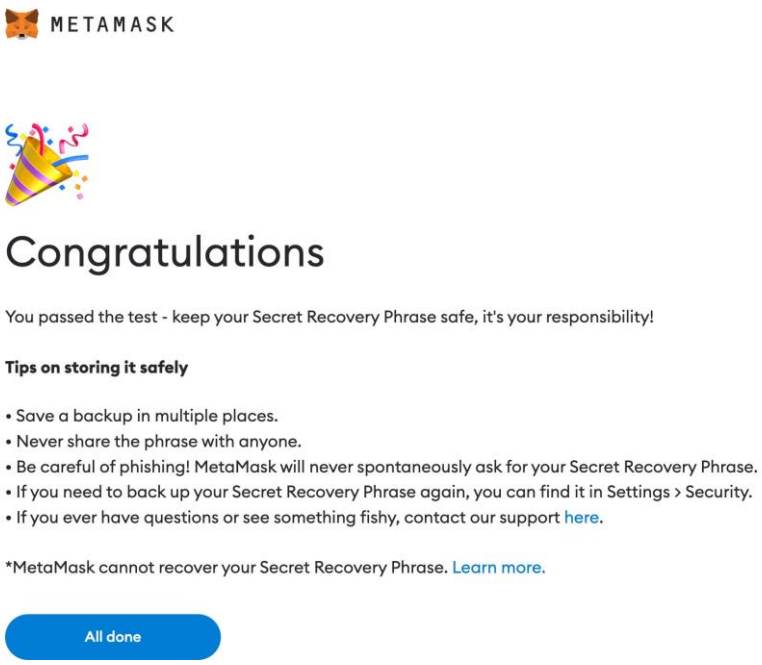


Caption

Step 6: Seed phrase confirmation

Confirm your secret backup phrase by clicking on each word in the order in whichthe words were presented on the previous screen. Click on “Confirm” to proceed.





Caption

Your MetaMask wallet has been set up successfully.You can now access your wallet by clicking on the MetaMask icon at the top-right-end corner of your preferred browser.

Cryptocurrency

Cryptocurrency is a digital payment system that doesn't rely on banks to verify transactions. It’s a peer-to-peer system that can enable anyone anywhere to send and receive payments. Instead of being physical money carried around and exchanged in the real world, cryptocurrency payments exist purely as digital entries to an online database describing specific transactions. When you transfer cryptocurrency funds, the transactions are recorded in a public ledger. Cryptocurrency is stored in digital wallets.

Cryptocurrency received its name because it uses encryption to verify transactions. This means advanced coding is involved in storing and transmitting cryptocurrency data between wallets and public ledgers. The aim of encryption is toprovide security and safety.

How does cryptocurrency work?

Cryptocurrencies run on a distributed public ledger called blockchain, a record of alltransactions updated and held by currency holders.

Units of cryptocurrency are created through a process called mining, which involves using computer power to solve complicated mathematical problems that generate coins. Users can also buy the currencies from brokers, and then store and spend themusing cryptographic wallets.

If you own cryptocurrency, you don’t own anything tangible. What you own is a keythat allows you to move a record or a unit of measure from one person to another without a trusted third party.

Although Bitcoin has been around since 2009, cryptocurrencies and applications ofblockchain technology are still emerging in financial terms, and more uses are expected in the future. Transactions including bonds, stocks, and other financial assets could eventually be traded using the technology.

What are the risks of using cryptocurrency?

Cryptocurrencies are still relatively new, and the market for these digital currencies isvery volatile. Since cryptocurrencies don't need banks or any other third party to

regulate them; they tend to be uninsured and are hard to convert into a form of tangible currency (such as US dollars or euros.) In addition, since cryptocurrencies are technology-based intangible assets, they can be hacked like any other intangible technology asset. Finally, since you store your cryptocurrencies in a digital wallet, if you lose your wallet (or access to it or to wallet backups), you have lost your entire cryptocurrency investment.

Conclusion

Hence we have learnt about how to create our own wallet using Metamask and what is crypto transactions and cryptocurrencies.

Questions

Q1) What do you understand by blocks in Blockchain technology?

Q2) What are the common types of ledgers that can be considered by the users?

Q3) Can you explain, what type of records can be kept in a Blockchain?

Q4) How are cryptocurrency transactions recorded?

Q5) Are blockchain and cryptocurrencies the same?

Q6) What is a crypto wallet?

Q7) Beyond a method for payment, what are the other functions of cryptocurrencies?

Q8) How safe is cryptocurrency?

Experiment No 3

Title - Write a smart contract on a test network, for the Bank account of a customerfor the following operations:

1. Deposit money
2. Withdraw Money
3. Show Balance

Objective -

To understand and explore the working of Blockchain technology and itsapplications.

Theory

Smart Contract

A "smart contract" is simply a program that runs on the Ethereum blockchain. It's a collection of code (its functions) and data (its state) that resides at a specific addresson the Ethereum blockchain.

Smart contracts are a type of Ethereum account. This means they have a balance andcan be the target of transactions. However they're not controlled by a user, instead they are deployed to the network and run as programmed. User accounts can then interact with a smart contract by submitting transactions that execute a function defined on the smart contract. Smart contracts can define rules, like a regular contract, & automatically enforce them via the code. Smart contracts cannot be deleted by default, and interactions with them are irreversible

Solidity

Solidity is an object-oriented, high-level language for implementing smart contracts.
Smart contracts are programs which govern the behaviour of accounts within the

Ethereum state.

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 detailsabout which languages Solidity has been inspired by in the

influences section.

language Solidity is statically typed, supports inheritance, libraries and complex user-defined types among other features.

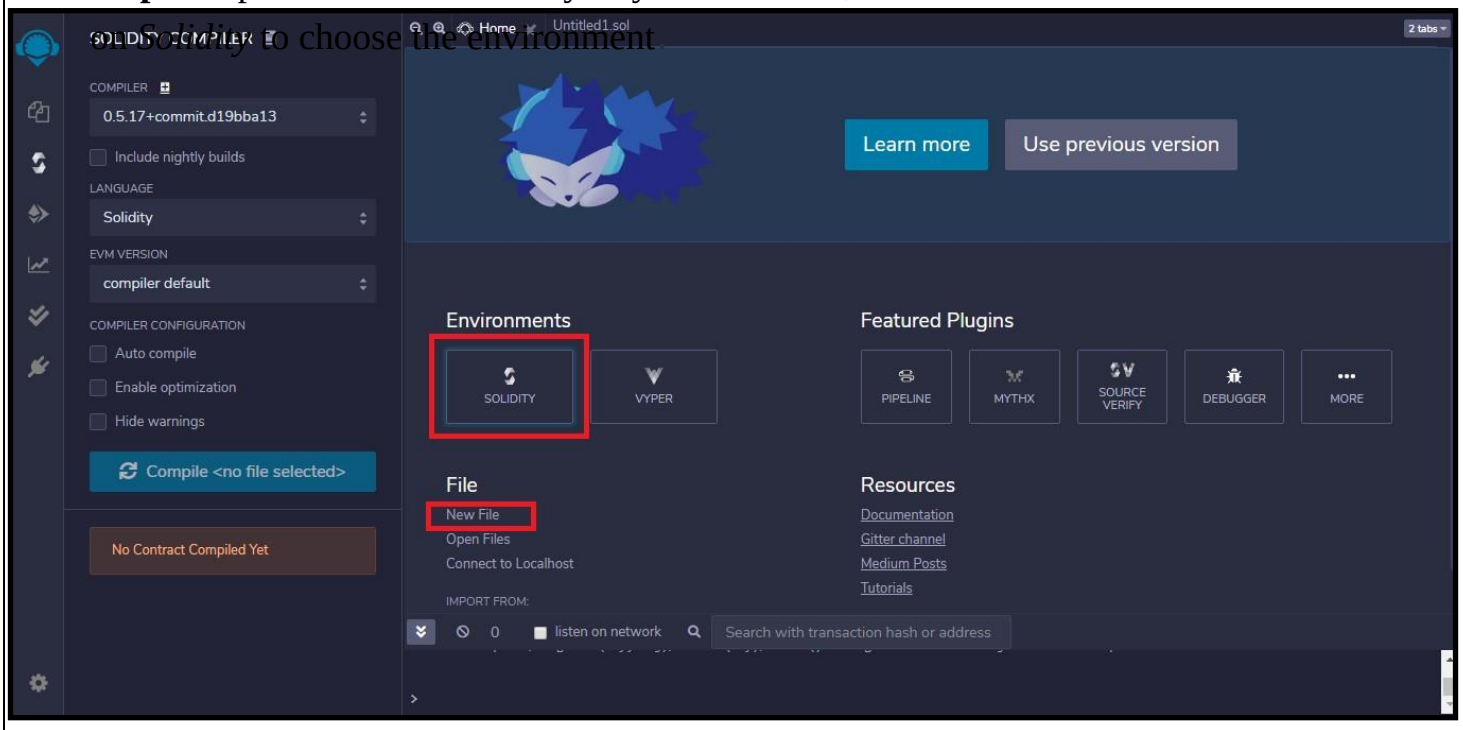
With Solidity you can create contracts for uses such as voting, crowdfunding, blind auctions, and multi-signature wallets.

Remix IDE

Remix IDE is an open source web and desktop application. It fosters a fast development cycle and has a rich set of plugins with intuitive GUIs. Remix is used for the entire journey of contract development with Solidity language as well as a playground for learning and teaching Ethereum

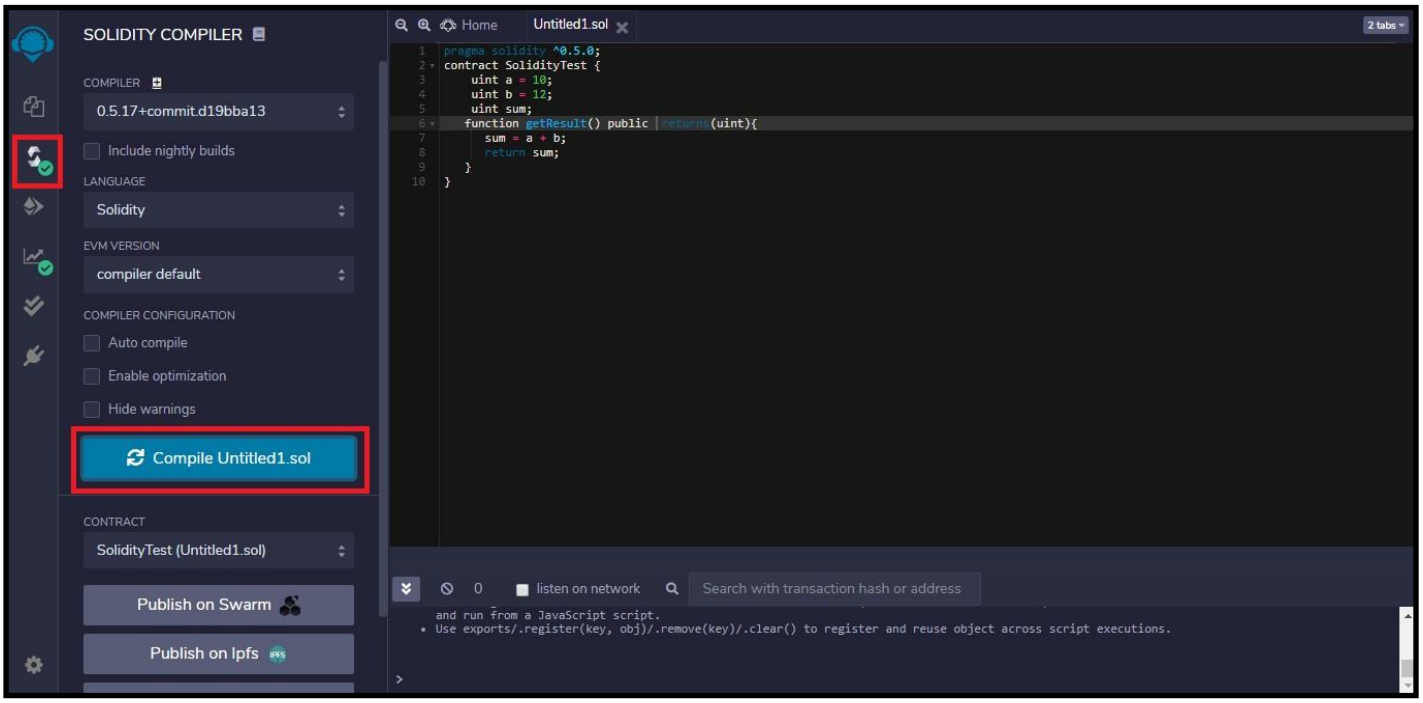
Steps for the compilation, execution, and debugging of the smart contract in Remix IDE.

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

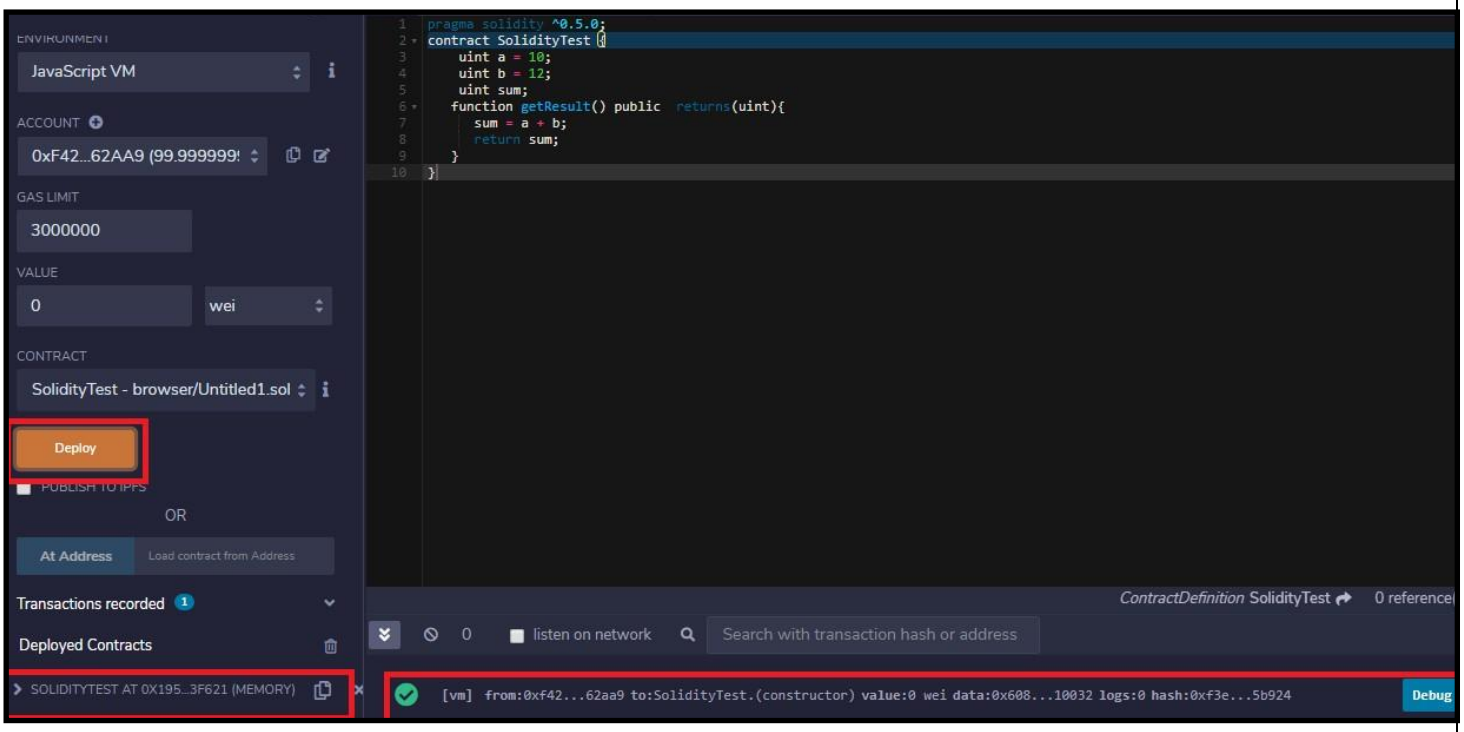


Step 2: Write the Smart contract in the code section, and click the *Compilebutton* under the Compiler window to compile the contract.

Caption

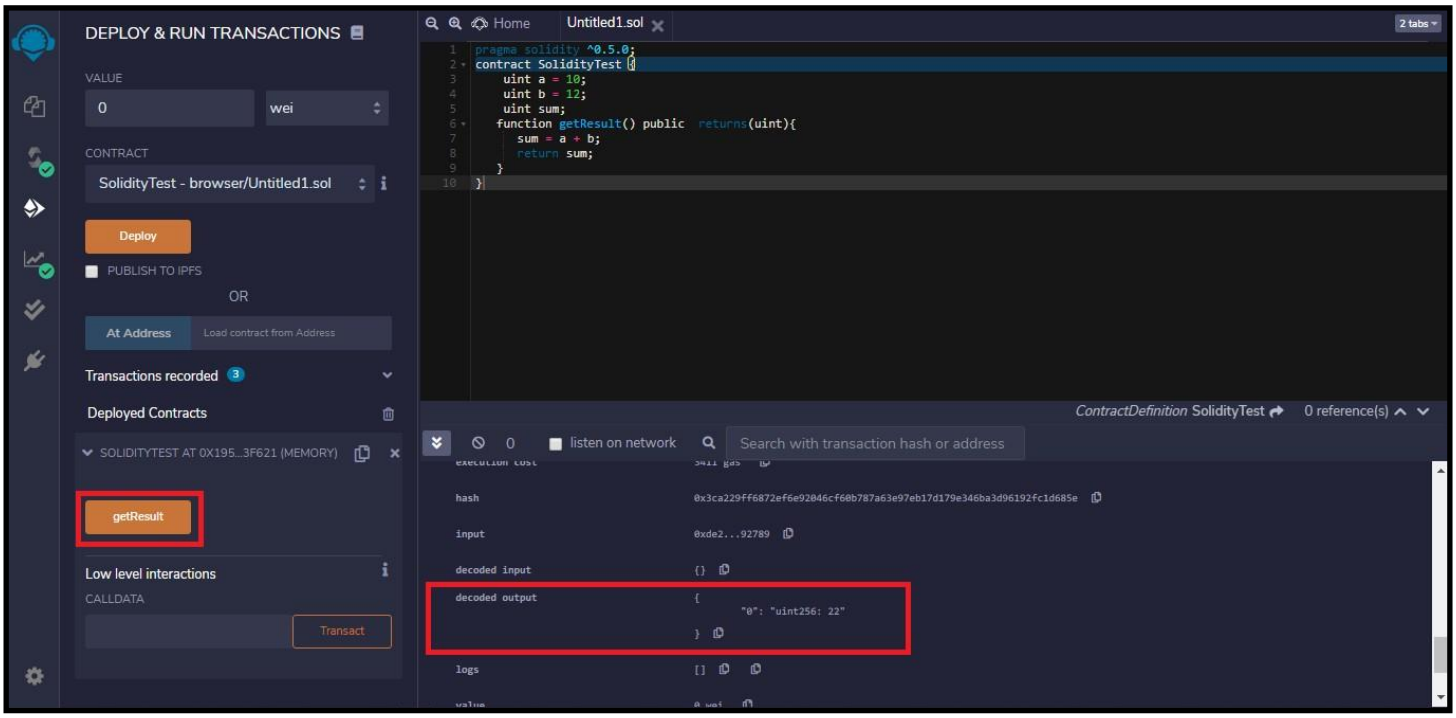


Step 3: To execute the code, click on the *Deploy button* under Deploy and Run Transactions window.

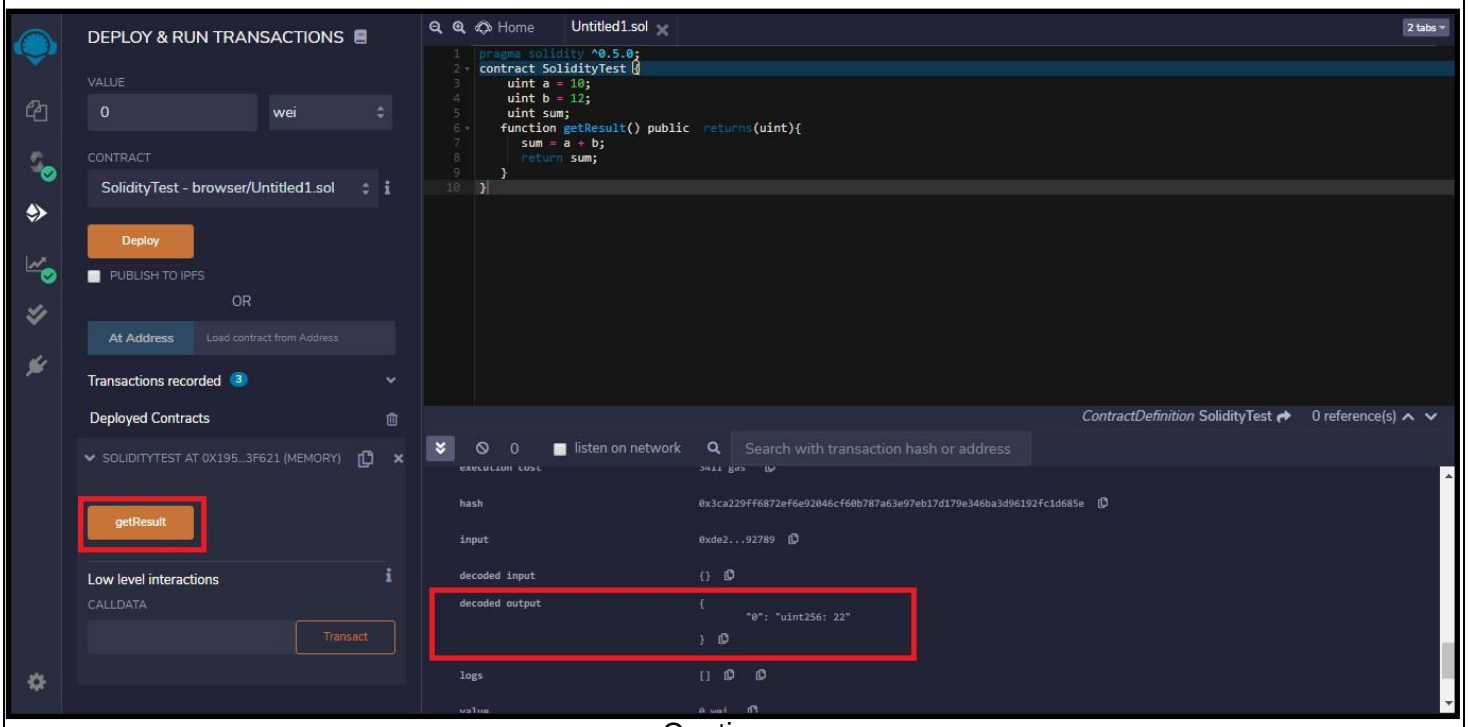


Caption

Step 4: After deploying the code click on the method calls under the drop-down of deployed contracts to run the program, and for output, check to click on the drop- down on the console.



Step 5: For debugging click on the *Debug button* corresponding to the method call inthe console. Here you can check each function call and variable assignments.



Caption

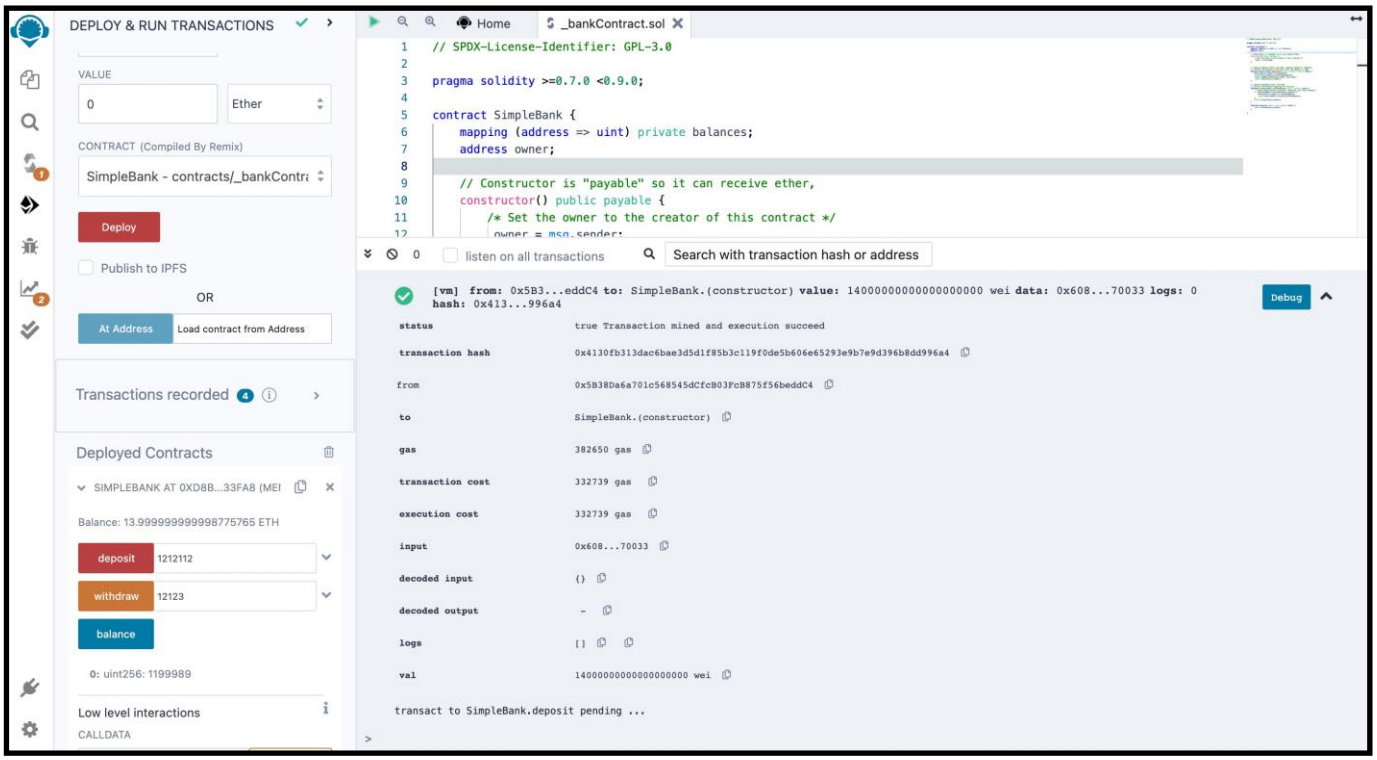
Program

Smart contract for the Bank account of the customer to do operations like Deposit,Withdraw and Show Balance

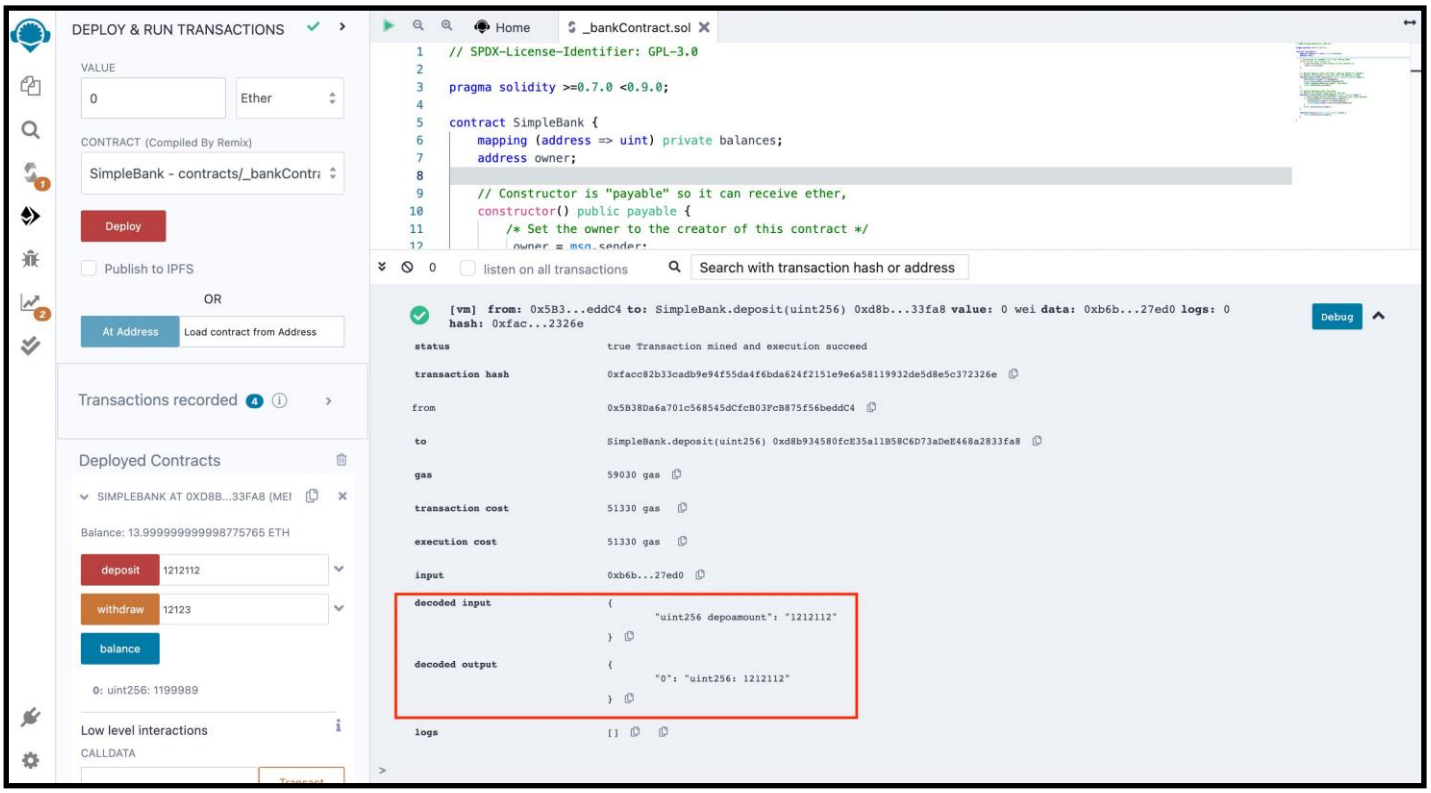
```
// SPDX-License-Identifier: GPL-3.0

pragma solidity >=0.7.0 <0.9.0;

contract SimpleBank {
    mapping (address => uint) private balances;
    address owner;

    // Constructor is "payable" so it can receive ether,
    constructor() public payable {
        /* Set the owner to the creator of this contract */
         owner = msg.sender;
    }


    /// @notice Deposit ether into bank, requires method is "payable"
    /// @return The balance of the user after the deposit is made
    function deposit(uint depoamount) public payable returns (uint) {
        balances[msg.sender] += depoamount;
        payable(msg.sender).transfer(depoamount);
        //emit LogDepositMade(msg.sender, msg.value);
        return balances[msg.sender];
    }

    /// @notice Withdraw ether from bank
    /// @return The balance remaining for the user
    function withdraw(uint withdrawAmount) public returns (uint) {
        // Check enough balance available, otherwise just return balance
        if (withdrawAmount <= balances[msg.sender]) {
            balances[msg.sender] -= withdrawAmount;
            payable(msg.sender).transfer(withdrawAmount);
        }
        return balances[msg.sender];
    }

    function balance() public view returns (uint) {
        return balances[msg.sender];
    }

}
```

Output

Deploy

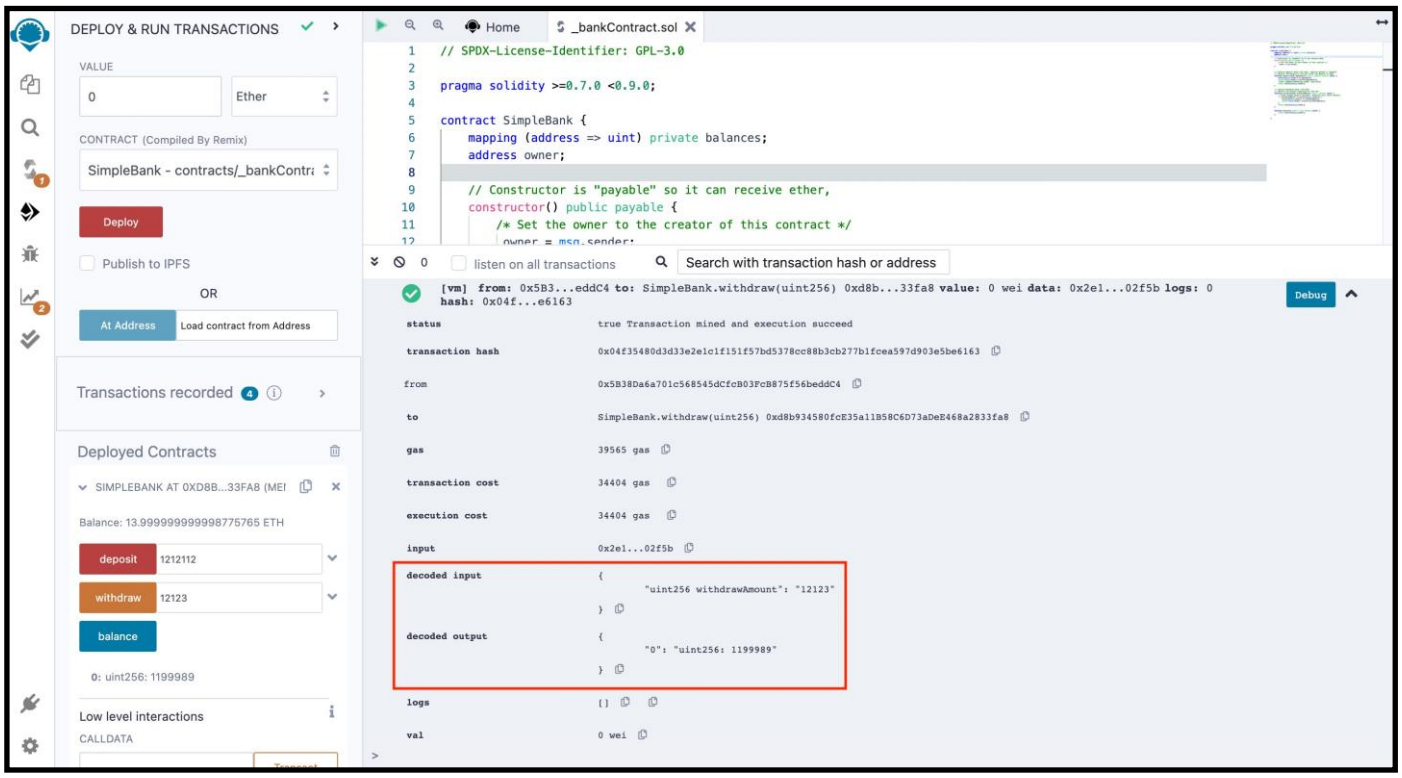


Deposit Money

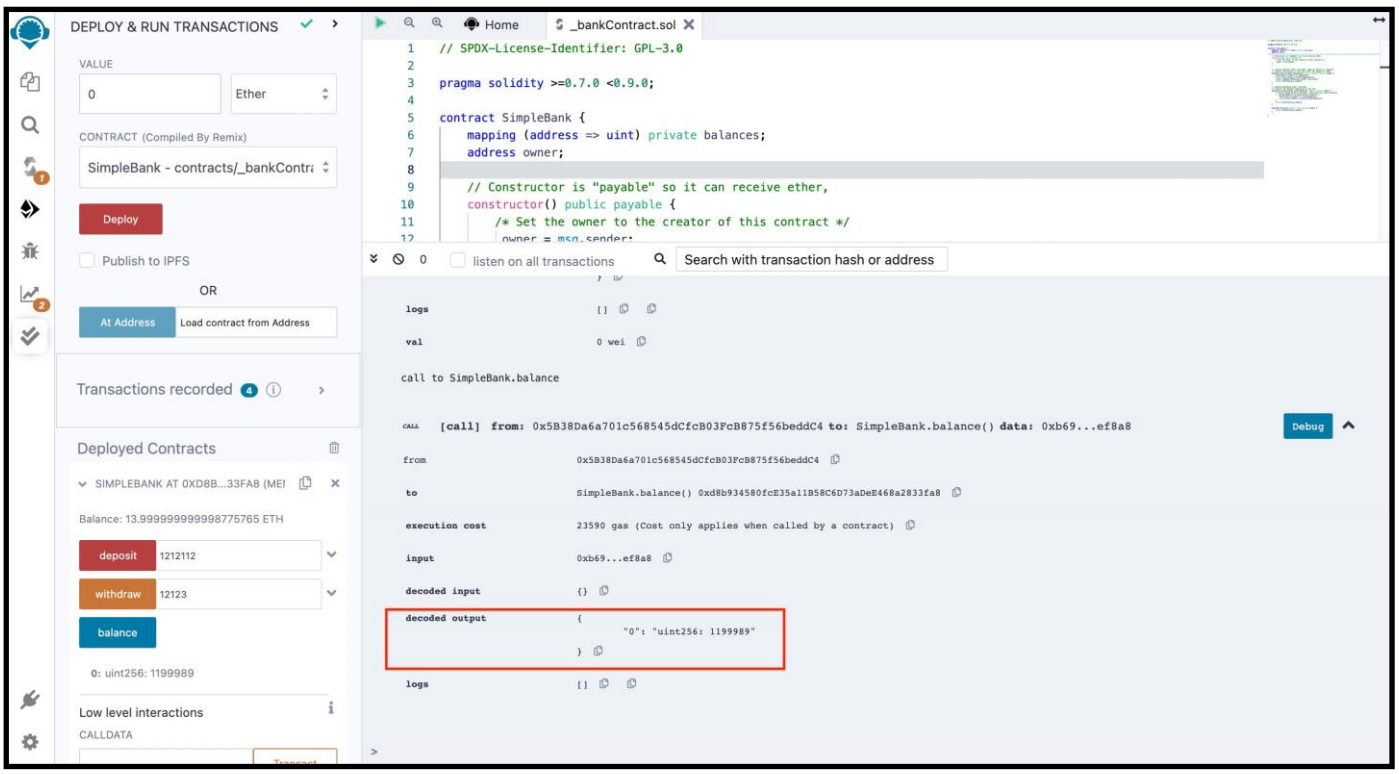


Caption

Withdraw Money



Balance



Caption

Conclusion

Hence we have learnt about how to create our own wallet using Metamask andwhat is crypto transactions and cryptocurrencies.

Questions

Q1) Define Smart Contracts and their advantages.
Q2) What is an EVM?
Q3) What is Solidity?
Q4) What types of applications can be developed using Solidity?
Q5) What are the benefits of using smart contracts on Ethereum?
Q6) What is a test network in Crypto?
Q7) Different test networks in Ethereum
Q8) What is the first thing specified in the Solidity file?

Experiment No 4

Title -

Write a program in solidity to create Student data. Use the following constructs:

1) Structures
2) Arrays
3) Fallback

Deploy this as a smart contract on Ethereum and Observe the transaction fee and Gasvalues.

Objective -

To understand and explore the working of Blockchain technology and its application

Theory

- **Solidity**

Solidity is an object-oriented programming language for implementing smart contracts on various blockchain platforms, most notably, Ethereum. It was developedby Christian Reitwiessner, Alex Beregszaszi, and several former Ethereum core contributors. Programs in Solidity run on Ethereum Virtual Machine.

Solidity was proposed in August 2014 by Gavin Wood; the language was laterdeveloped by the Ethereum project's Solidity team, led by Christian Reitwiessner.
Solidity is the primary language on Ethereum as well as on other private blockchains,such as the enterprise-oriented Hyperledger Fabric blockchain. SWIFT deployed a proof of concept using Solidity running on Hyperledger Fabric.

- **Solidity - Struct**

Structs in Solidity allow you to create more complicated data types that have multipleproperties. You can define your own type by creating a **struct**.
They are useful for grouping together related data.
Structs can be declared outside of a contract and imported in another contract. Generally, it is used to represent a record. To define a structure *struct* keyword isused, which creates a new data type.

Syntax:

```
struct <structure_name> {
    <data type> variable_1;
    <data type> variable_2;  }
```

Caption

- **Solidity - Arrays**

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.

Declaring Arrays

To declare an array of fixed size in Solidity, the programmer specifies the type of the elements and the number of elements required by an array as follows −

```
type arrayName [ arraySize ];
```

This is called a single-dimension array. The **arraySize** must be an integer constant greater than zero and **type** can be any valid Solidity data type. For example, todeclare a 10-element array called balance of type uint, use this statement −

```
uint balance[10];
```

To declare an array of dynamic size in Solidity, the programmer specifies the type of the elements as follows −

```
type[] arrayName;
```

Initializing Arrays

You can initialize Solidity array elements either one by one or using a single statement as follows −

```
uint balance[3] = [1, 2, 3];
```

The number of values between braces [ ] can not be larger than the number of elements that we declare for the array between square brackets [ ]. Following is an example to assign a single element of the array −

If you omit the size of the array, an array just big enough to hold the initialization is created. Therefore, if you write −

```
uint balance[] = [1, 2, 3];
```

You will create exactly the same array as you did in the previous example.`balance[2] = 5;`

The above statement assigns element number 3rd in the array a value of 5.

Creating dynamic memory arrays

Dynamic memory arrays are created using new keywords.uint size

= 3;

uint balance[] = new uint[](size);

Accessing Array Elements

An element is accessed by indexing the array name. This is done by placing the indexof the element within square brackets after the name of the array. For example −

uint salary = balance[2];

The above statement will take 3rd element from the array and assign the value to salary variable. Following is an example, which will use all the above-mentioned three concepts viz. declaration, assignment and accessing arrays −

- **Solidity – Fall Back Function**

The solidity fallback function is executed if none of the other functions match the function identifier or no data was provided with the function call. Only one unnamed function can be assigned to a contract and it is executed whenever the contract receives plain Ether without any data. To receive Ether and add it to the totalbalance of the contract, the fallback function must be marked payable. If no such function exists, the contract cannot receive Ether through regular transactions and will throw an exception.

Properties of a fallback function:

- Has no name or arguments.
- If it is not marked **payable**, the contract will throw an exception if it receives plainether without data.
- Can not return anything.
- Can be defined once per contract.
- It is also executed if the caller meant to call a function that is not available
- It is mandatory to mark it external.
- It is limited to 2300 gas when called by another function. It is so for as to make thisfunction call as cheap as possible.

Ethereum transaction fee

Ethereum transaction fees work differently in comparison to Bitcoin's. The fee takesinto account the amount of computing power needed to process a transaction, knownas gas. Gas also has a variable price measured in ether (ETH), the network's native token.

While the gas needed for a specific transaction can stay the same, gas prices can riseor fall. This gas price is directly related to network traffic. If you pay a higher gas price, miners will likely prioritize your transaction.

How are Ethereum transaction fees calculated?

The total gas fee is simply a price that covers the cost, plus an incentive to processyour transaction. However, you should also consider the gas limit, which defines what's the maximum price paid for that transaction or task.

In other words, the gas cost is the amount of work required, and the gas price is the price paid for “each hour” of work. The relation between these two and the gas limitdefines the total fee for an Ethereum transaction or smart contract operation.

Let's pick a random transaction on Etherscan.io as an example. The transaction cost21,000 gas, and the gas price was 71 Gwei. So, the total transaction fee was 1,491,000 Gwei or 0.001491 ETH.

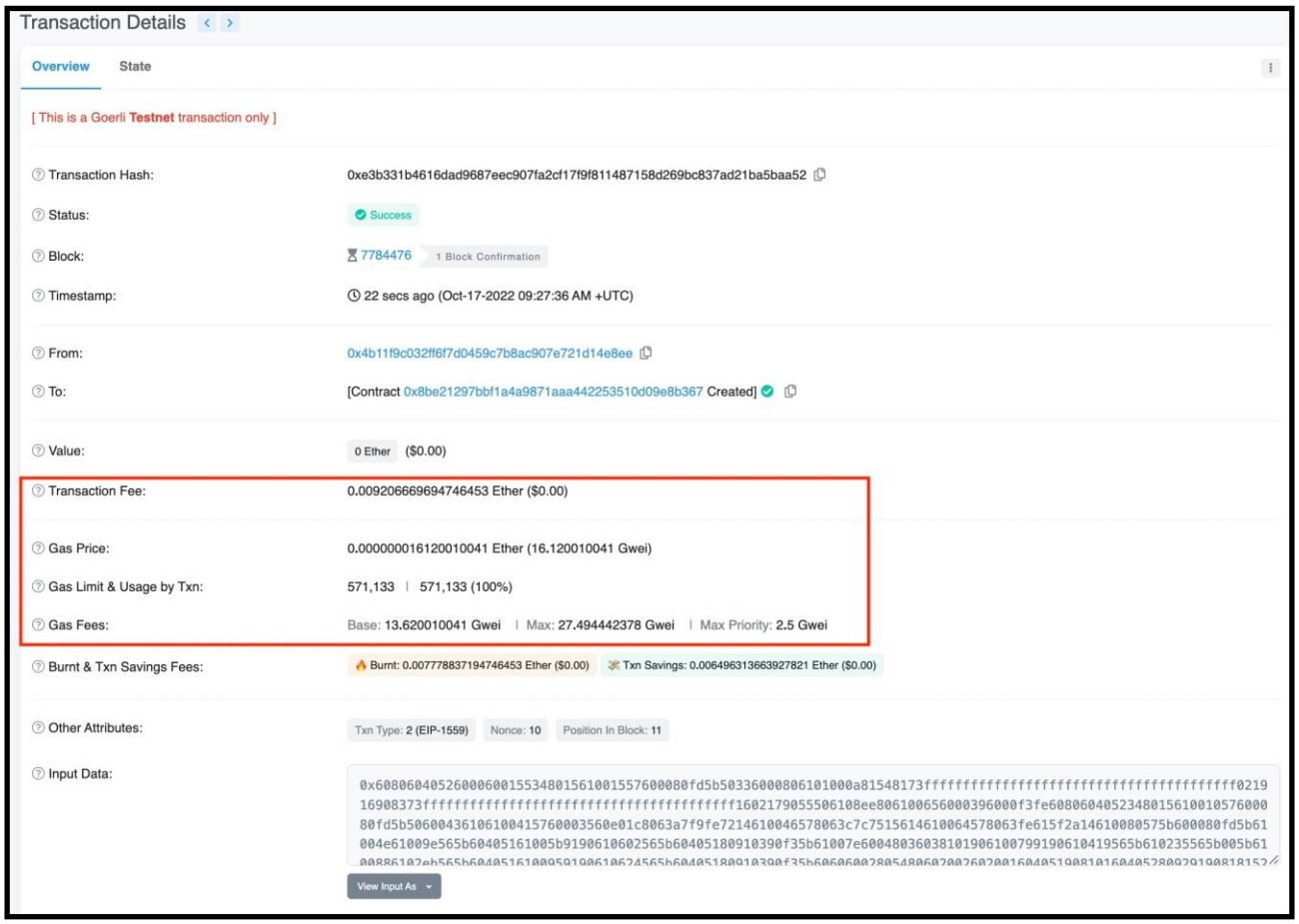
As Ethereum makes its way towards a Proof of Stake model (see Casper), there is anexpectation that gas fees will decrease. The amount of gas needed to confirm a transaction will be lower as the network will need only a fraction of the computational power to validate transactions. But, network traffic can still affect transaction fees as validators prioritize higher-paying transactions.

Program

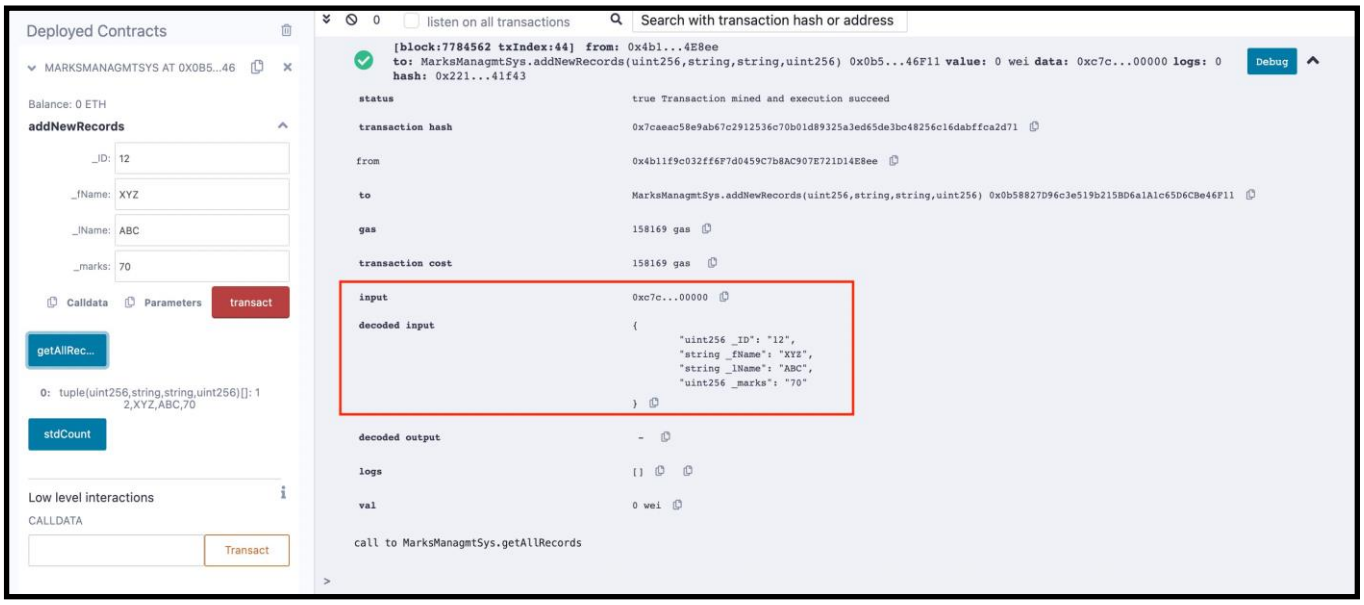
```
//SPDX-License-Identifier: UNLICENSED

pragma solidity >= 0.7.0 <0.9.0;


// Build the Contract
contract MarksManagmtSys
{
    // Create a structure for
    // student details
    struct StudentStruct
    {
        uint ID;
        string fName;
        string lName;
        uint marks;
    }

    address owner;

    uint public stdCount = 0;

    //Create Array to store Student data
    StudentStruct[] stdRecords;

    constructor()
    {
        owner=msg.sender;
    }

    // Create a function to add
    // the new records
    function addNewRecords(uint  ID,
                           string memory _fName,
                           string memory _lName,
                           uint _marks) public payable
    {
        // Increase the count by 1
        stdCount = stdCount + 1;

        //Adding data into array
        stdRecords.push(StudentStruct(_ID , _fName , _lName , _marks));

    }

    function getAllRecords() public view returns(StudentStruct[] memory)
{
        return stdRecords;
    }
}
```

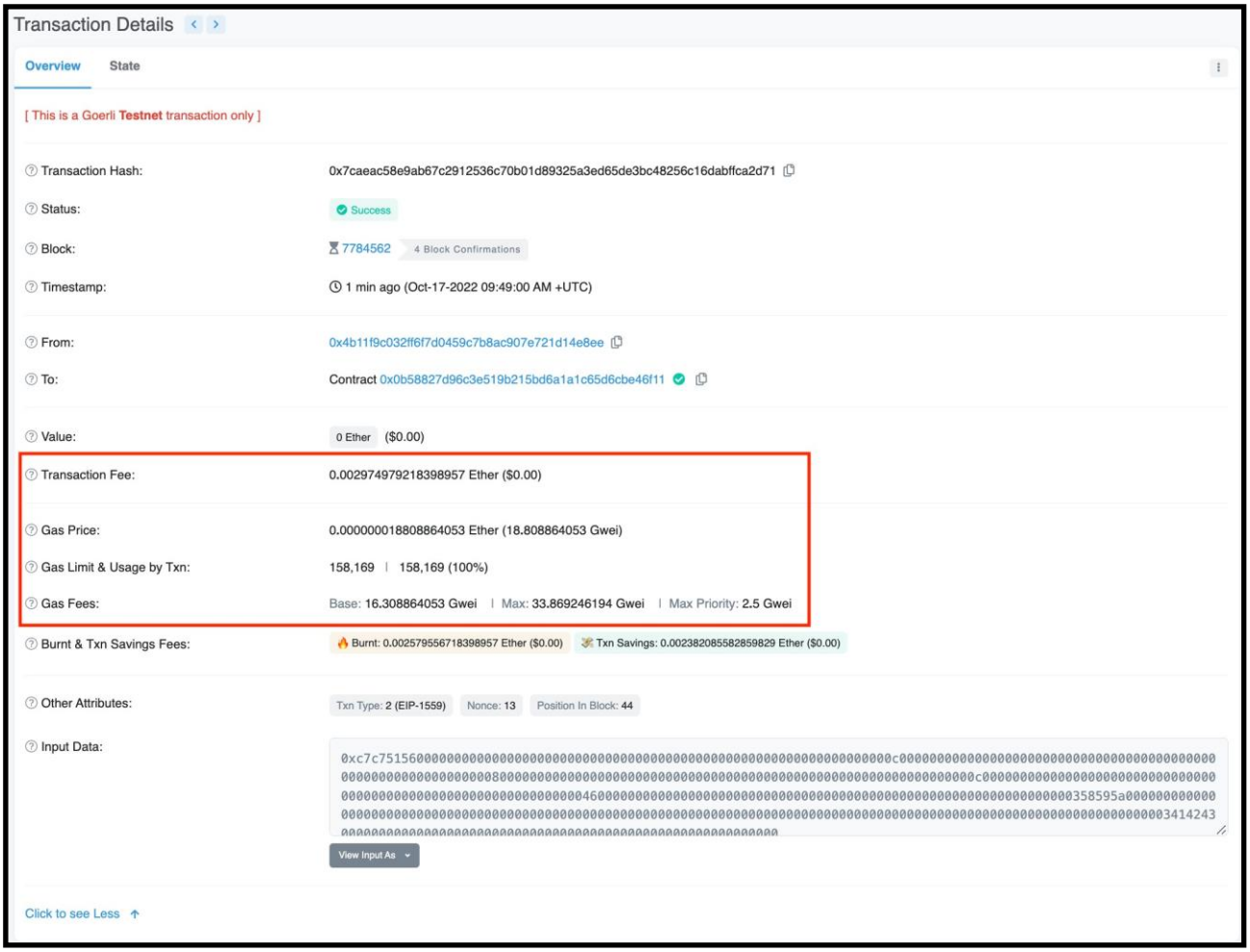
Output

1) An analysis of the transaction fee and gas fee required for contract deployment



2) An analysis of the transaction fee and gas fee required for smart contracts transactions





Caption

Conclusion

Hence, we learned about the Basic Fundamentals of Solidity language and itsvarious attributes as well as the Ethereum transaction fee and Gas fee.

Questions

Q1) What is Solidity used for?
Q2) Which type of language is Solidity?
Q3) What types of applications can be developed using Solidity?
Q4) What are the main differences between Solidity and other programminglanguages like Python, Java, or C++?
Q5) Why is there gas, more precisely?
Q6) What does the gas usage in a transaction depend on?
Q7) How is the transaction fee calculated?
Q8) If an execution of a Smart Contract costs less than the specified gas, does theuser get a refund?