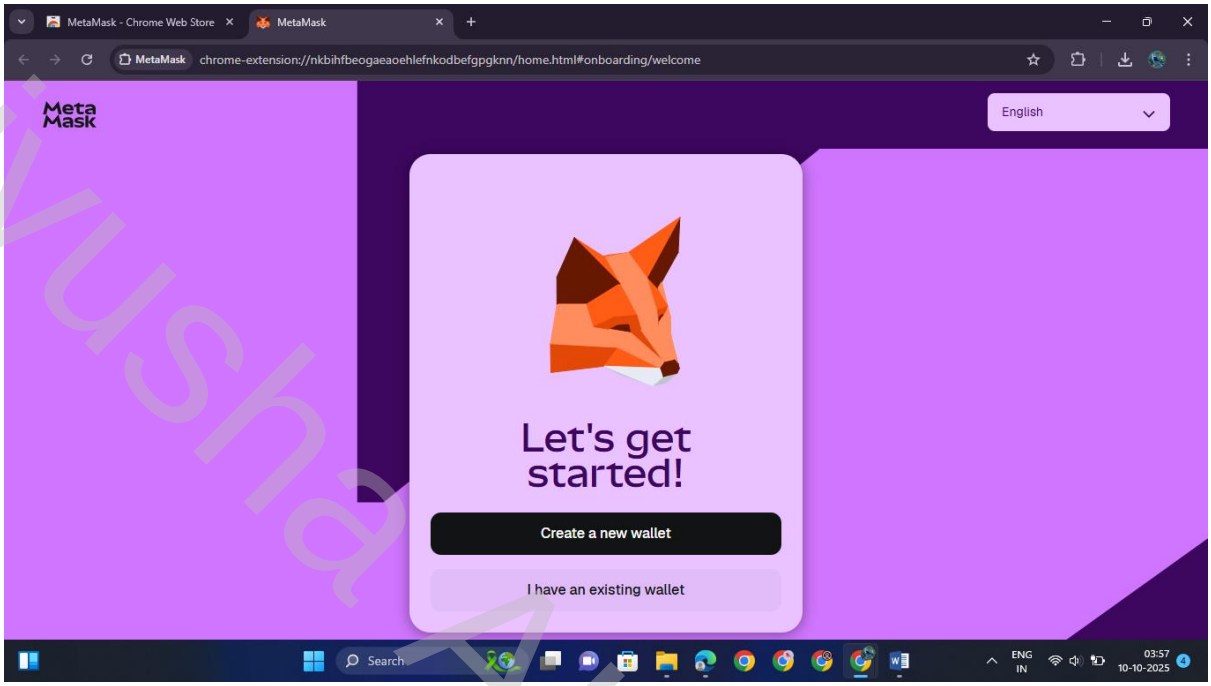
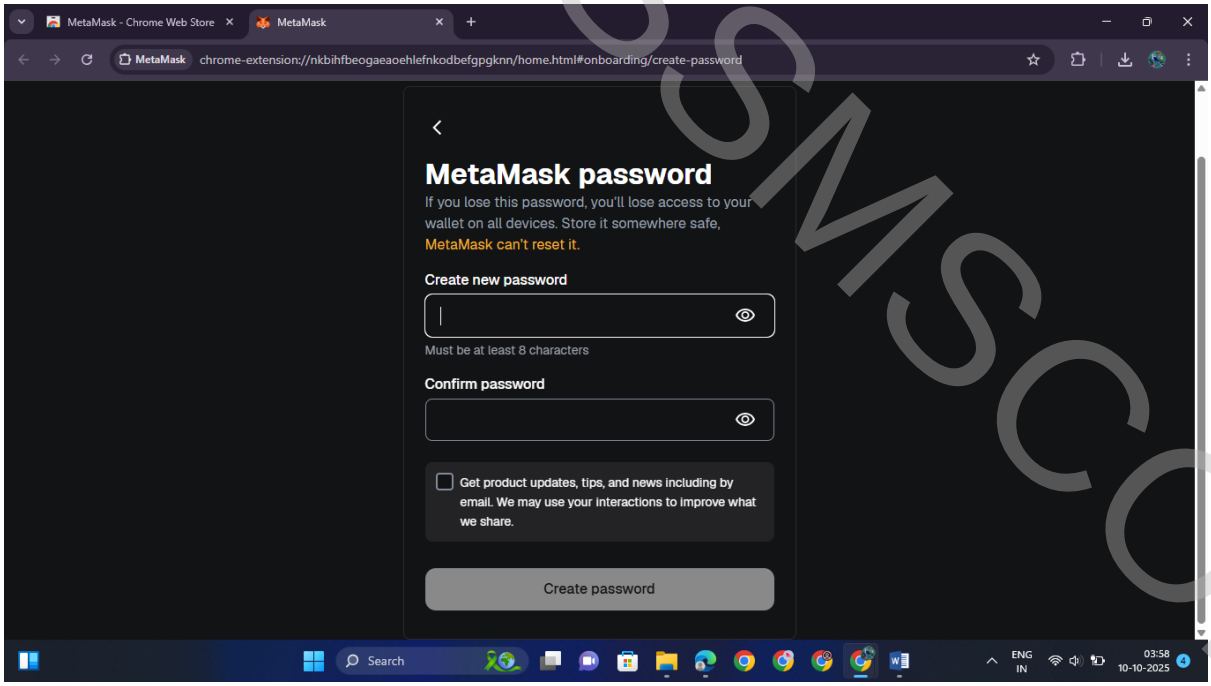
Practical 1: Installation of MetaMask and study spending Ether per transaction.

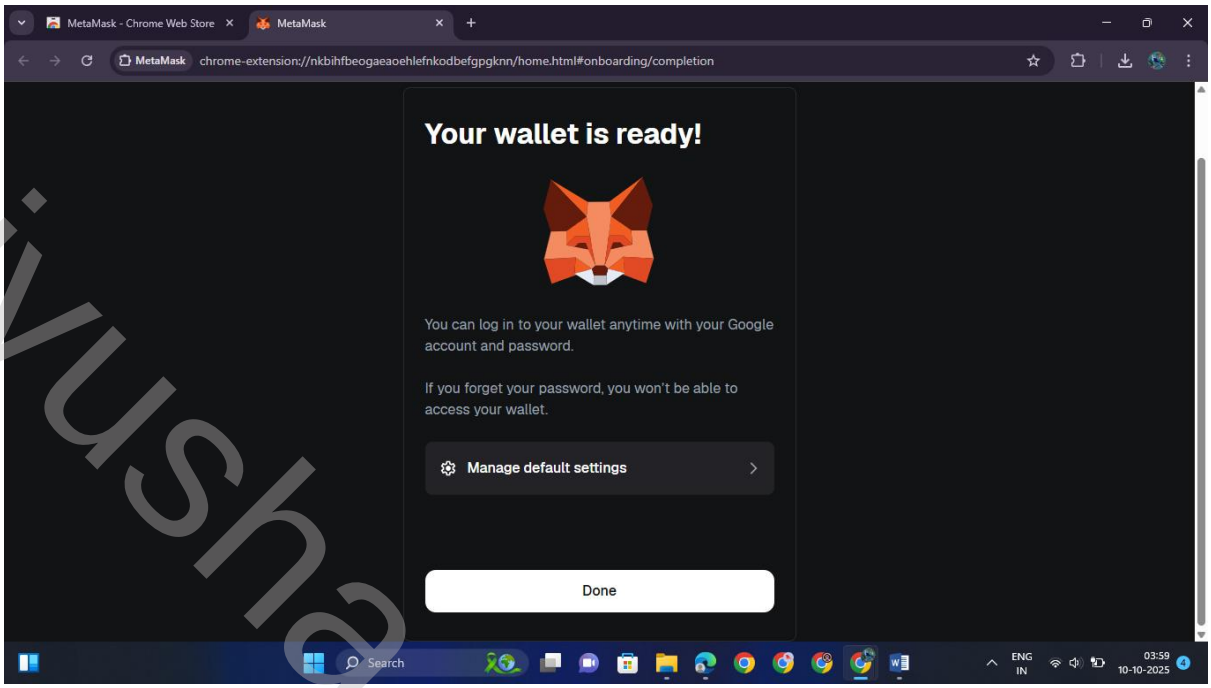
[1] Go to metamask chrome extension



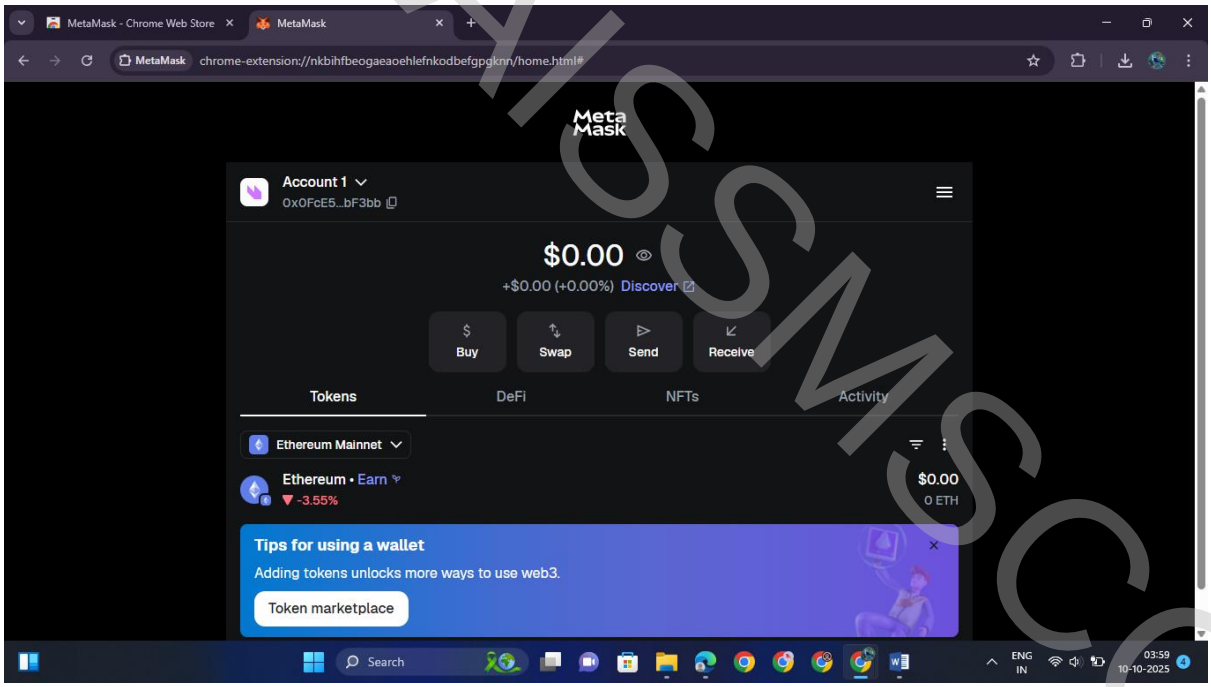
[2] With your google account create an account on MetaMask and set a password



[3] The wallet will be ready in a few seconds



[4] The dashboard of your wallet will be displayed on successful creation

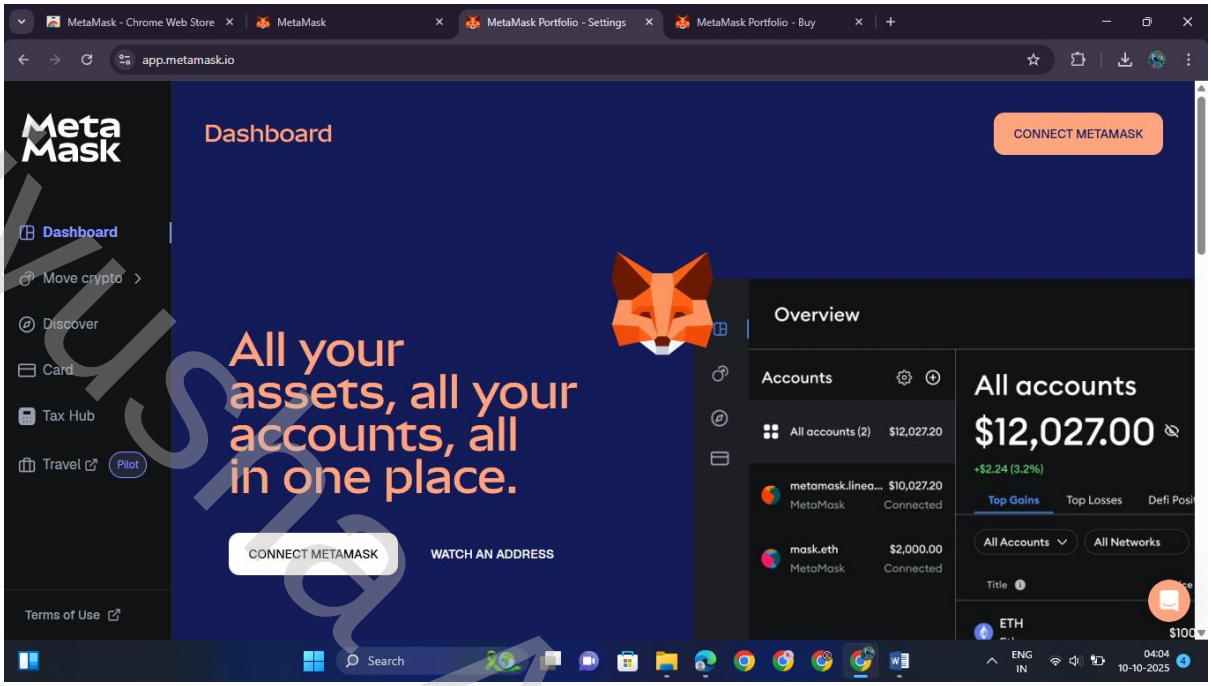


CONCLUSION:

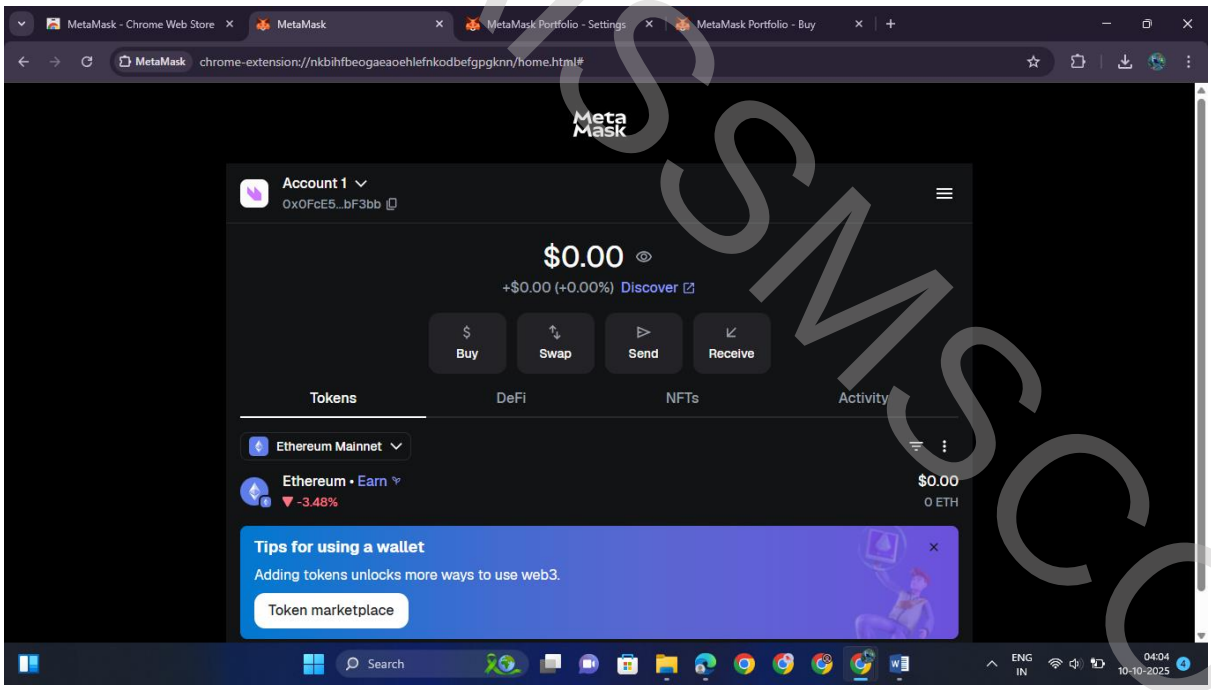
Thus we successfully installed metamask and created an account for transactions and studied ether per transaction

Practical 2: Create your own wallet using Metamask for crypto transactions.

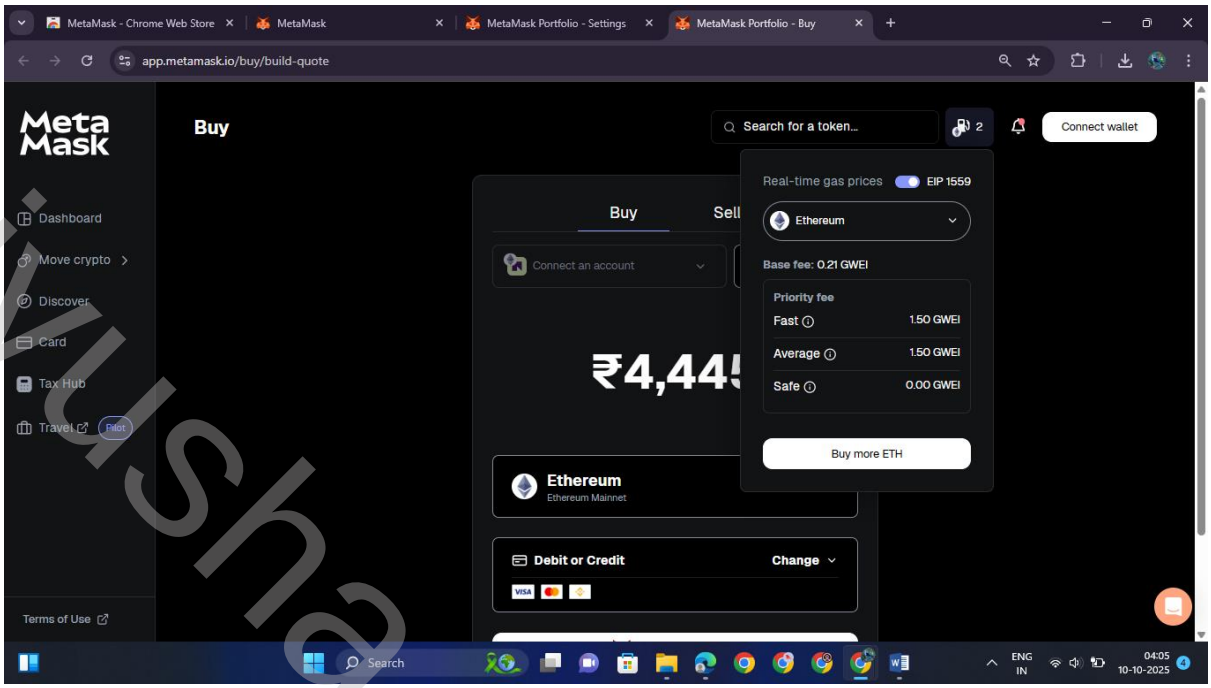
[1] Login to your account

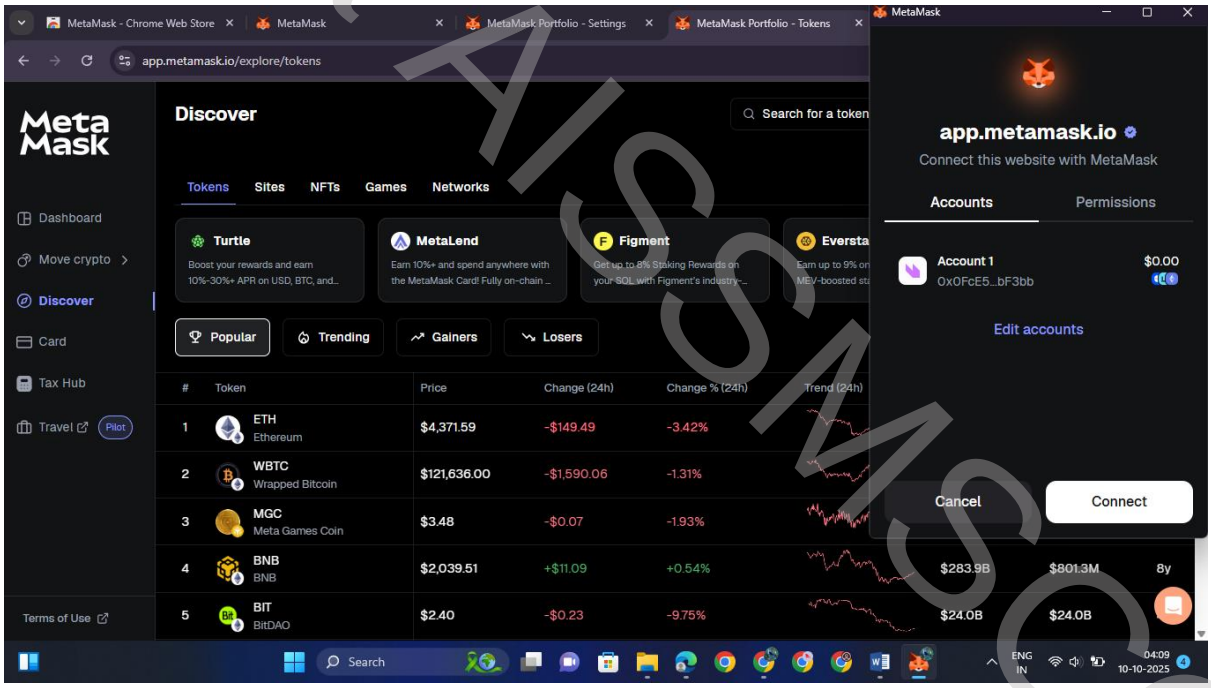


[2] The default wallet will be created already



[3] Navigate to the dashboard and explore the wallet



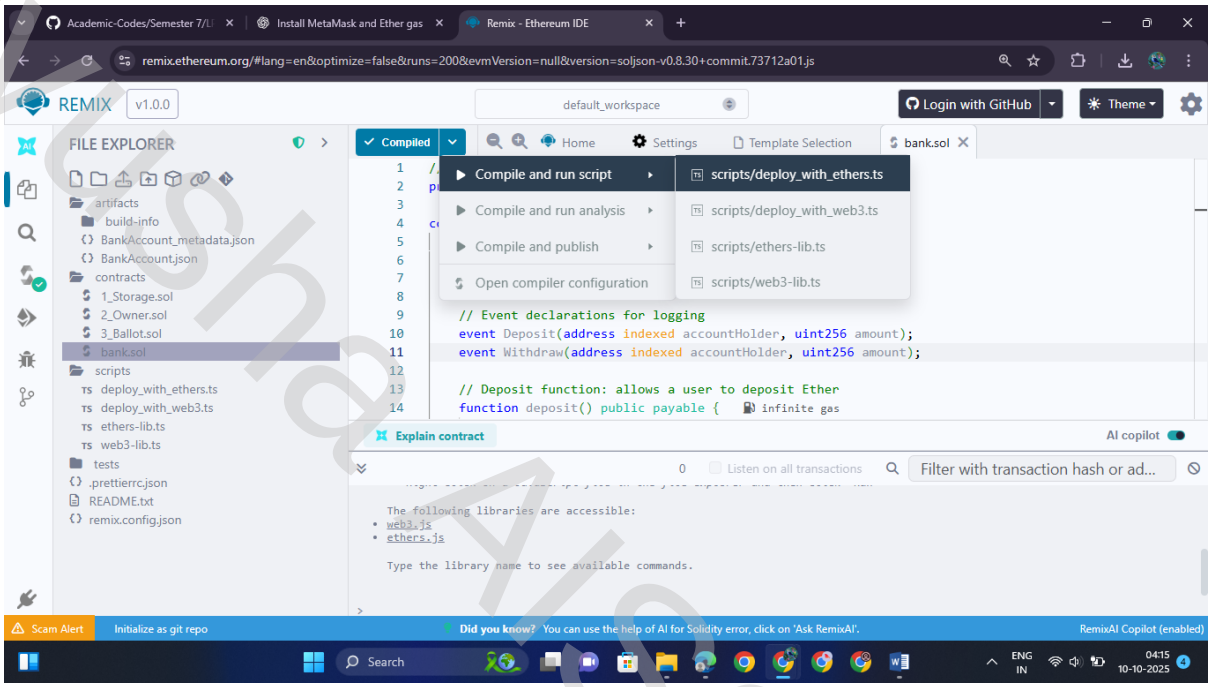


CONCLUSION:

Thus we successfully created a metamask wallet using metamask for crypto

Practical 3: Write a smart contract on a test network, for Bank account of a customer for following operations:

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




```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
contract BankAccount {
  // Mapping to store balance of each account holder
  mapping(address => uint256) private balances;
  // Event declarations for logging
  event Deposit(address indexed accountHolder, uint256 amount);
  event Withdraw(address indexed accountHolder, uint256 amount);
  // Deposit function: allows a user to deposit Ether
  function deposit() public payable {
    require(msg.value > 0, "Deposit amount must be greater than zero");
    balances[msg.sender] += msg.value;
    emit Deposit(msg.sender, msg.value);
  }
```

```
    // Withdraw function: allows a user to withdraw Ether
    function withdraw(uint256 _amount) public {
        require(_amount > 0, "Withdrawal amount must be greater than zero");
        require(balances[msg.sender] >= _amount, "Insufficient balance");
        balances[msg.sender] -= _amount;
        // Transfer Ether to the caller
        payable(msg.sender).transfer(_amount);
        emit Withdraw(msg.sender, _amount);
    }
    // Show balance function: returns the balance of the caller
    function showBalance() public view returns (uint256) {
        return balances[msg.sender];
    }
}
```

```
Withdraw:
AccountHolder: 0xUserAddress123
Amount: 0.4 ETH
```

```
Balance: 1.3 ETH
```

Practical 4: Write a program in solidity to create Student data. Use the following constructs:

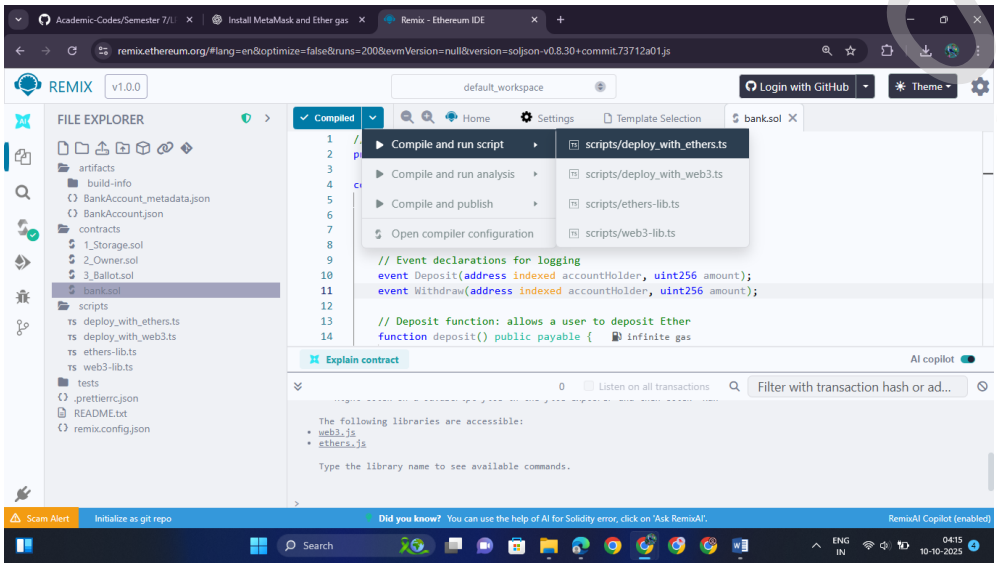
- **Structures**
- **Arrays**
- **Fallback**

Deploy this as smart contract on Ethereum and Observe the transaction fee and Gas values.

```
/ SPDX-License-Identifier: MIT
pragma solidity >=0.6.2 <0.9.0;
contract TestPayable {
   uint x;
   uint y;
   receive() external payable { x = 2; y = msg.value; }
}
contract StudentRegister {
   mapping (uint => Student) private students;
   address public owner;
   constructor() public payable {
      /* Set the owner to the creator of this contract */
      owner = msg.sender;
   }
   modifier onlyOwner {
      require(msg.sender == owner);
      _;
   }
   /// Student structure
   struct Student {
      uint studentId;
      string name;
      /* Marks array */
      uint[] marks;
      uint percentage;
      bool exist;
   }
```

```
    function register(
        uint studentId,
        string memory name,
        uint[] memory marks
    ) public onlyOwner returns (uint) {
        require(students[studentId].exist == false, "Student data already exist.");
        require(marks.length == 3, "Only 3 subjects are available. Array length should be 3.");
        uint totalMarks = getArraySum(marks);
        uint percentage = (totalMarks * 100) / 150;
        students[studentId] = Student(
            studentId,
            name,
            marks,
            percentage,
            true
        );
        return percentage;
    }
    /// @notice Get student details from the record
    /// @return Student id, name, marks, percentage of the student
    function getStudentDetails(
        uint studentId
    ) public view returns (uint, string memory, uint[] memory, uint) {
        require(students[studentId].exist == true, "No student data available.");
        /* Access student from the registed using studentId */
        Student memory student = students[studentId];
        return(
            student.studentId,
            student.name,
            student.marks,
            student.percentage
        );
    }
    /// @notice Get sum of the array
```

```
    /// @return sum of the array
    function getArraySum(uint[] memory array) private pure returns (uint sum) {
        sum = 0;
        for (uint i = 0; i < array.length; i++) {
            require(0 <= array[i] && array[i] <= 100, "Marks should be between 0 and 100.");
            sum += array[i];
        }
    }
    function callTestPayable(TestPayable test) public returns (bool) {
        (bool success,) = address(test).call(abi.encodeWithSignature("nonExistingFunction()"));
        require(success);
        // results in test.x becoming == 1 and test.y becoming 0.
        (success,) = address(test).call{value: 1}(abi.encodeWithSignature("nonExistingFunction()"));
        require(success);
        // results in test.x becoming == 1 and test.y becoming 1.
        // If someone sends Ether to that contract, the receive function in TestPayable will be called.
        // Since that function writes to storage, it takes more gas than is available with a
        // simple ``send`` or ``transfer``. Because of that, we have to use a low-level call.
        (success,) = address(test).call{value: 2 ether}("");
        require(success);
        // results in test.x becoming == 2 and test.y becoming 2 ether.
        return true;
    }
}
```



Practical 5: Write a survey report on types of Blockchains and its real time use cases.

Introduction

Blockchain technology is a decentralized, distributed ledger system that ensures transparency, security, and immutability of data. It has emerged as a revolutionary tool in finance, supply chain, healthcare, and many other sectors. The core concept involves a chain of blocks, each containing transactional data, cryptographically secured and linked to the previous block. The decentralized nature eliminates the need for a central authority, reducing the risks of fraud and enhancing trust. Today, blockchains are broadly classified into different types based on accessibility and governance, each suited for specific real-world applications.

1. Types of Blockchains

1.1 Public Blockchain

A public blockchain is open to everyone. Anyone can join, read, write, or participate in the consensus process.

- **Features**: Decentralized, transparent, highly secure.
- **Examples**: Bitcoin, Ethereum.
- **Advantages**: No central control, strong security through consensus mechanisms like Proof of Work or Proof of Stake.
- **Challenges**: High energy consumption, slower transaction speeds.

1.2 Private Blockchain

Private blockchains are restricted networks where only selected participants can access and validate transactions.

- **Features**: Controlled access, faster transactions, customizable governance.
- **Examples**: Hyperledger Fabric, R3 Corda.
- **Advantages**: High efficiency, better privacy, lower energy requirements.
- **Challenges**: Centralization risk, requires trust among participants.

1.3 Consortium Blockchain

Also known as federated blockchain, it is controlled by a group of organizations rather than a single entity.

- **Features**: Partially decentralized, pre-selected nodes validate transactions.
- **Examples**: Energy Web Chain, Quorum.
- **Advantages**: Faster than public blockchain, collaborative governance, enhanced security.
- **Challenges**: Limited transparency, dependent on consortium members' integrity.

1.4 Hybrid Blockchain

Hybrid blockchains combine features of public and private blockchains, allowing controlled access to some data while keeping other parts public.

- **Features**: Flexible, scalable, selective transparency.
- **Examples**: Dragonchain, IBM Food Trust.
- **Advantages**: Balance between privacy and transparency, ideal for enterprise use.
- **Challenges**: Complex implementation, requires careful governance.

2. Real-Time Use Cases

2.1 Finance and Banking

- **Cryptocurrencies** like Bitcoin and Ethereum enable peer-to-peer payments without intermediaries.
- **Cross-border payments** become faster and cheaper using blockchain networks.
- **Smart contracts** automate lending, insurance claims, and trading.

2.2 Supply Chain Management

- Blockchain provides end-to-end visibility of products from manufacturing to delivery.
- Example: Walmart uses blockchain to trace the origin of fresh produce, reducing contamination risks.
- Ensures transparency, accountability, and reduced fraud in logistics.

2.3 Healthcare

- Patient data and medical records can be securely shared across hospitals using private or consortium blockchains.
- Example: Guardtime uses blockchain to protect health records and maintain data integrity.
- Facilitates research, reduces duplication, and ensures privacy compliance.

2.4 Real Estate

- Property ownership records and land registries can be digitized on blockchain.
- Example: Propy allows buyers and sellers to execute property deals via smart contracts.
- Reduces paperwork, prevents fraud, and accelerates transactions.

2.5 Voting Systems

- Blockchain enables tamper-proof, transparent voting systems.
- Example: Voatz in the United States pilots blockchain-based mobile voting.

- Ensures integrity, prevents vote manipulation, and increases voter trust.

2.6 Energy and Utilities

- Blockchain enables peer-to-peer energy trading.
- Example: Power Ledger allows consumers to sell excess solar energy to neighbors securely.
- Promotes decentralization and renewable energy adoption.

Conclusion

Blockchain technology is transforming multiple industries by offering transparency, security, and decentralization. Each type of blockchain—public, private, consortium, and hybrid—serves distinct purposes, and its adoption depends on factors like privacy, speed, and governance needs. Real-time applications in finance, supply chain, healthcare, and energy highlight blockchain's potential to revolutionize traditional processes. As the technology matures, hybrid and consortium blockchains are expected to see more enterprise adoption, balancing privacy with transparency and efficiency. Overall, blockchain represents a cornerstone for future digital infrastructure.