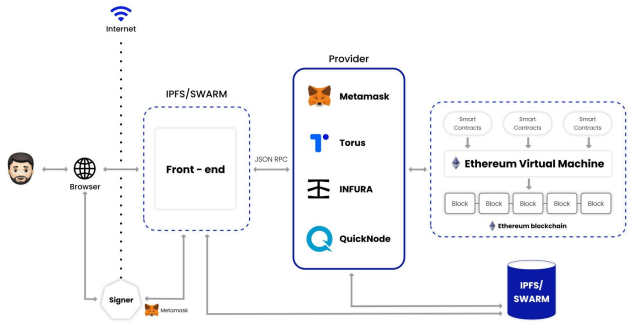
Solidity Contract



What's Solidity?

- Solidity is a high-level programming language that is designed specifically for writing smart contracts on the Ethereum blockchain.
- It is similar to other programming languages like C++ and JavaScript, but has some unique features that make it well-suited for developing decentralized applications (dApps) on the Ethereum platform.
- Solidity was developed by the Ethereum Foundation and is now widely used by developers around the world to build a variety of decentralized applications, including financial applications, decentralized autonomous organizations (DAOs), games, and more.

What's a Solidity Contract?

- A Solidity contract is a piece of code written in the Solidity programming language that is executed on the Ethereum blockchain. In other words, it is a smart contract that is stored and executed on the Ethereum network.
- Solidity contracts can be used to perform a wide range of functions on the blockchain, such as managing digital assets, creating and managing decentralized autonomous organizations (DAOs), implementing voting systems, and more.
- Solidity contracts are executed by the Ethereum Virtual Machine (EVM), which is a runtime environment that executes smart contracts on the Ethereum network.

2

Question?

Using a solidity Language, create a simple bank contract that allows a user to deposit, withdraw and view balance

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SimpleBank {
    mapping(address => uint256) private _balances;

    function deposit() public payable {
        _balances[msg.sender] += msg.value;
    }

    function withdraw(uint256 amount) public {
        require(_balances[msg.sender] >= amount, "Insufficient balance");
        payable(msg.sender).transfer(amount);
        _balances[msg.sender] -= amount;
    }

    function balance() public view returns (uint256) {
        return _balances[msg.sender];
    }
}
```

Explanation

- **pragma solidity ^0.8.0;** This is a version declaration that specifies which version of Solidity the contract is written in.
- **contract SimpleBank {** This line starts the definition of the SimpleBank contract.

- **mapping(address => uint256) private balances;** This line defines a private mapping of addresses to unsigned integers, which will be used to keep track of each user's balance.
- **function deposit() public payable {** This line defines a public function called deposit that can be called by anyone and accepts Ether as payment.
- **balances[msg.sender] += msg.value;** This line adds the amount of Ether sent in the deposit to the user's balance in the balances mapping.
- **function withdraw(uint256 amount) public {** This line defines a public function called withdraw that can be called by anyone and accepts a uint256 value as an argument.
- **require(balances[msg.sender] >= amount, "Insufficient balance");** This line checks that the user has enough balance to withdraw the requested amount and reverts the transaction if they don't.
- **payable(msg.sender).transfer(amount);** This line sends the requested amount of Ether to the user's address.
- **balances[msg.sender] -= amount;** This line subtracts the requested amount from the user's balance in the balances mapping.
- **function getBalance() public view returns (uint256) {** This line defines a public function called getBalance that can be called by anyone and returns the user's current balance.
- **return balances[msg.sender];** This line returns the balance of the user who called the getBalance function.

Example

To deposit ether, we can call the deposit function and send Ether to the wallet.

For example, if you want to deposit 1 Ether, you can call the deposit function with a value of 1 Ether:

SimpleBank contractInstance = SimpleBank (0x12345678901234);

contractInstance.deposit{value: 1 ether}();

To withdraw Ether from your bank account, you can call the withdraw function and specify the amount of Ether you want to withdraw.

For example, if you want to withdraw 0.5 Ether, you can call the withdraw function with an argument of 0.5 Ether:

SimpleBank contractInstance =SimpleBank(0x12345678901234);

contractInstance.withdraw(0.5 ether);

To check your current balance, you can call the getBalance function:

SimpleBank contractInstance = SimpleBank(0x12345678901234);

uint256 balance = contractInstance.getBalance();