كلية العلوم و التقنيات بطنجة
ⵜⴰⵖⵉⵡⴰⵏⵜ ⵏ ⵜⵎⴰⵙⵙⴰⵏⵉⵏ ⴷ ⵜⵙⵏⵉⵜⵉⵏ ⵏ ⵟⴰⵏⵊⴰ
Faculté des Sciences et Techniques de Tanger

11/6/2024

SMART CONTRACT "MINI SOCIAL NETWORK" IN SOLIDITY

Idriss Khattabi
BLOCKCHAIN & SECURITE APPLICATIVE

Introduction:

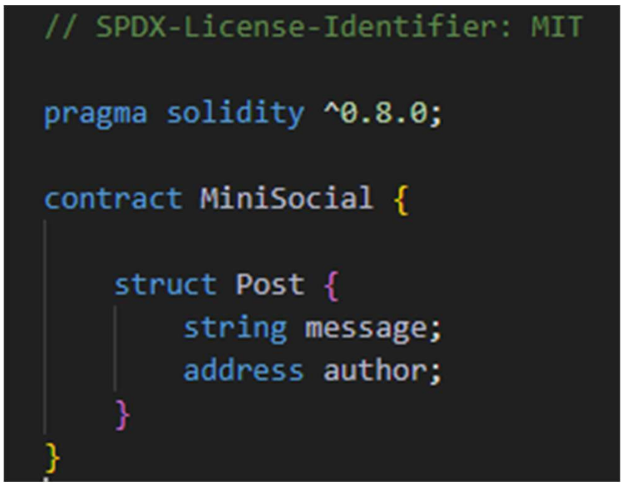
The purpose of this task is to develop a basic smart contract in Solidity for a "Mini Social Network" that allows users to publish, view, and count posts. The project is designed to be deployed on a test network and includes the ability to manage posts in a decentralized way, using the Ethereum blockchain. This report provides a step-by-step overview of each development phase, including source code creation, deployment, testing, and results.

Task Details and Implementation:

Step 1: Declaration of the Smart Contract

After the creation of 'MiniSocial.sol' file in Remix IDE, lets define the Contract, The smart contract 'MiniSocial' was declared. The Post structure was defined to contain two essential pieces of information:

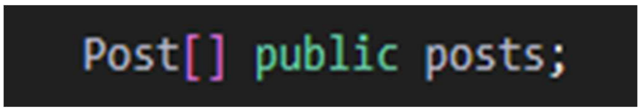
- **message**: A string variable to hold the content of the post.
- **author**: An address variable to store the Ethereum address of the user who published the post.



```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract MiniSocial {

    struct Post {
        string message;
        address author;
    }
}
```

Step 2: Declaring the List of Posts

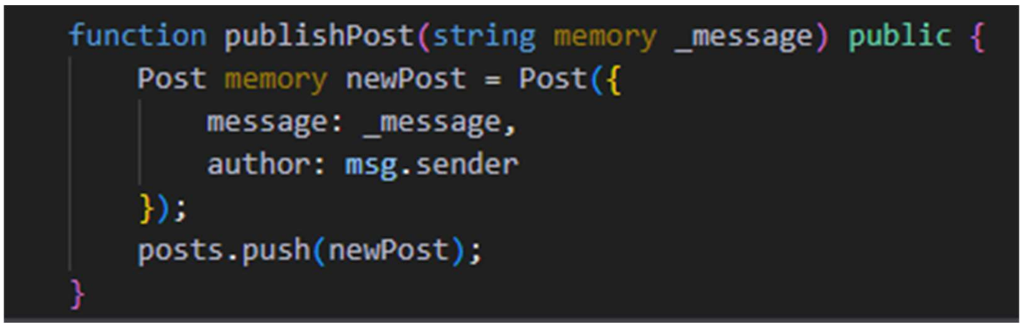
- Let's declare a dynamic array named posts of type Post[] was declared to store all published messages. This array will hold instances of the Post structure.



```
Post[] public posts;
```

Step 3: Publish Post Function

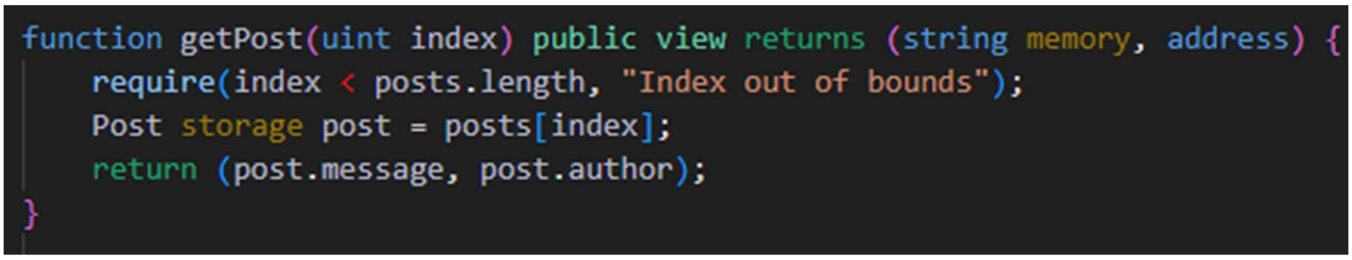
- Creates a ‘publishPost(string memory _message)’ function that allows users to publish a message. This function should create a new Post with the user's message and address (msg.sender) and then add it to the posts array.



```
function publishPost(string memory _message) public {
    Post memory newPost = Post({
        message: _message,
        author: msg.sender
    });
    posts.push(newPost);
}
```

Step 4: Retrieve a Specific Post

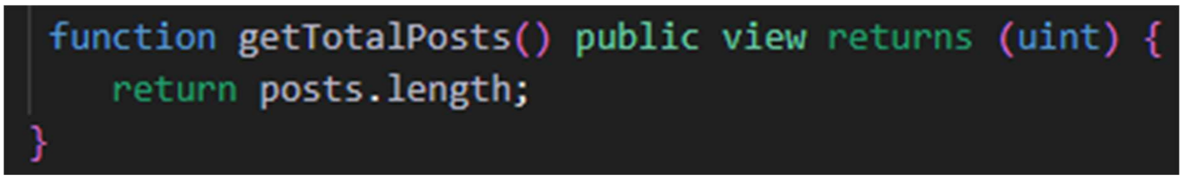
- The ‘getPost’ function allows users to retrieve a post by its index.
    - **Parameters**: Accepts an ‘uint’ parameter index to specify the post location in the array.
    - **Return Values**: Returns the message and author fields of the specified Post.
    - **Access Type**: The function is public view, meaning it can only read data and cannot modify the state.



```
function getPost(uint index) public view returns (string memory, address) {
    require(index < posts.length, "Index out of bounds");
    Post storage post = posts[index];
    return (post.message, post.author);
}
```

Step 5: Function to Retrieve the Total Number of Posts

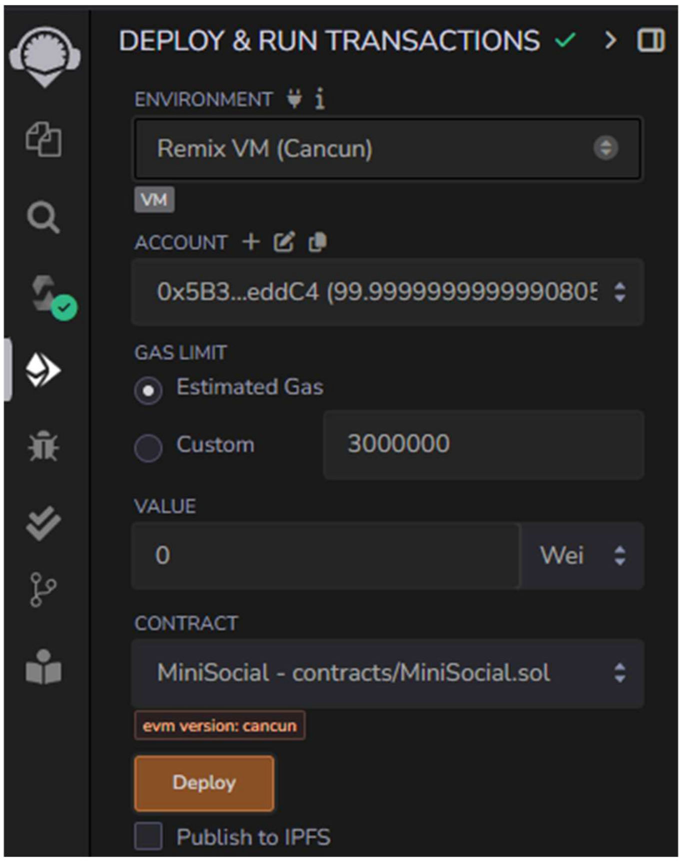
- ‘getTotalPosts’ function returns the total number of posts in the posts array.



```
function getTotalPosts() public view returns (uint) {
    return posts.length;
}
```

Step 6: Deployment and Testing

- **Deployment:** After writing the smart contract, Let's deploy it on a test network in Remix



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

Remix VM (Cancun)

VM

ACCOUNT

0x5B3...eddC4 (99.9999999999990805

GAS LIMIT

● Estimated Gas

○ Custom 3000000

VALUE

0 Wei

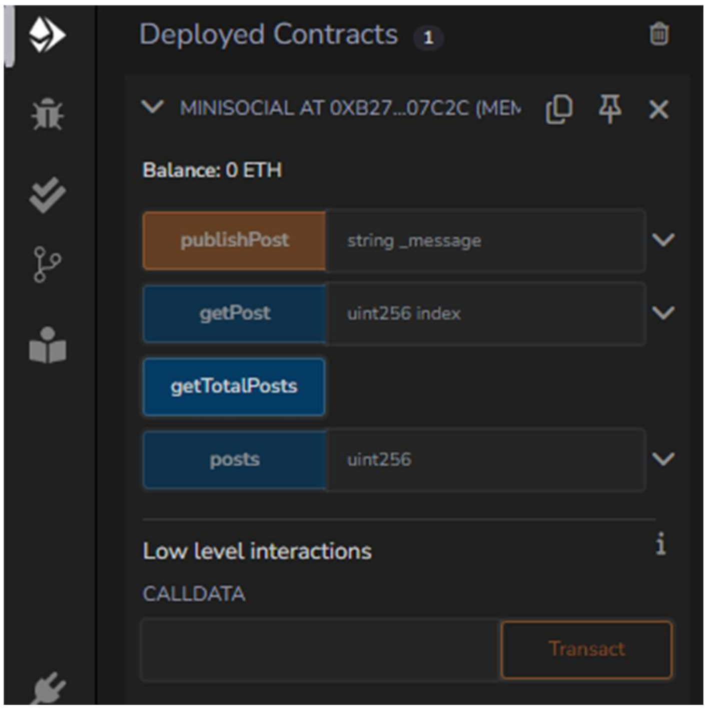
CONTRACT

MiniSocial - contracts/MiniSocial.sol

evm version: cancun

Deploy

Publish to IPFS



Deployed Contracts 1

MINISOCIAL AT 0XB27...07C2C (MEM

Balance: 0 ETH

publishPost string _message

getPost uint256 index

getTotalPosts

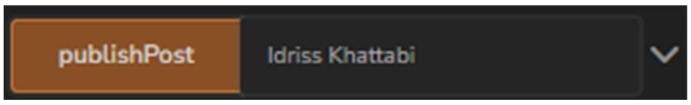
posts uint256

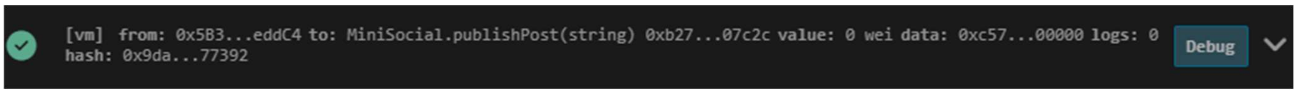
Low level interactions

CALLDATA

Transact

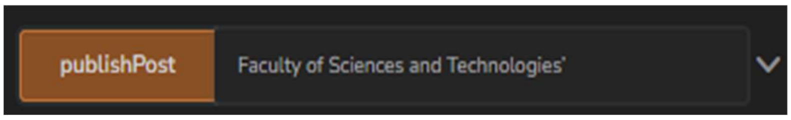
- **Testing**:
  - **Publishing Messages**: Let's publish 2 post :
    - **<u>Post 1 :</u>** 'Idriss Khattabi'

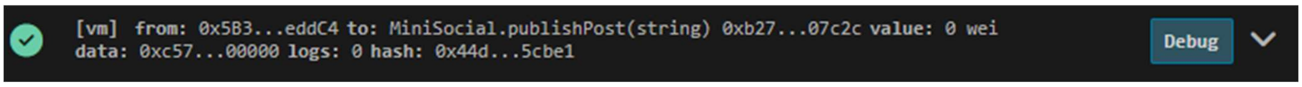




```
[vm] from: 0x5B3...eddC4 to: MiniSocial.publishPost(string) 0xb27...07c2c value: 0 wei data: 0xc57...00000 logs: 0
hash: 0x9da...77392
```

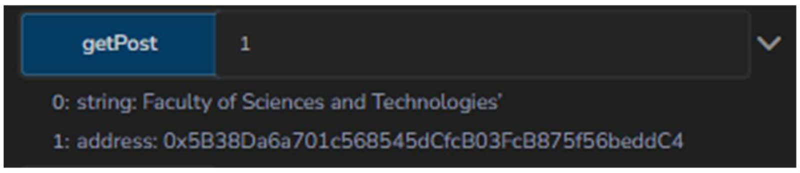
    - <u>Post 2 :</u> 'Faculty of Sciences and Technologies'





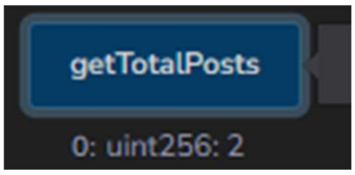
```
[vm] from: 0x5B3...eddC4 to: MiniSocial.publishPost(string) 0xb27...07c2c value: 0 wei
data: 0xc57...00000 logs: 0 hash: 0x44d...5cbe1
```

  - **Retrieving Messages**: Let's retrieve the second Post



```
0: string: Faculty of Sciences and Technologies'
1: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
```

  - **Counting Messages**: Let's get the Total of Posts



```
0: uint256: 2
```

Conclusion:

The "Mini Social Network" smart contract provides a foundational implementation for decentralized social media. This task demonstrated several key Solidity programming principles, including data storage, user interaction, and state retrieval. This mini social network could be expanded with additional features, such as editing or deleting posts, adding timestamps, and implementing a like or comment system.