جامعة عبد المالك السعدي
Université Abdelmalek Essaâdi

كلية العلوم و التقنيات بطنجة
Faculté des Sciences et Techniques de Tanger

ROYAUME DU MAROC
UNIVERSITÉ ABDELMALEK ESSAADI
FACULTÉ DES SCIENCES ET TECHNIQUES - TANGER

CYCLE MASTER EN SCIENCES ET TECHNIQUES

FILIÈRE : INTELLIGENCE ARTIFICIELLE ET SCIENCES DE DONNÉES

BLOCKCHAIN

(TWITTER DAPP)

RÉALISÉE PAR :
NOUIH Omar

ENCADRÉ PAR :
Pr. BENADBELOUAHAB Ikram

IASD
MASTER INTELLIGENCE ARTIFICIELLE ET SCIENCES DE DONNÉES

Année Universitaire 2024 - 2025

Contents

Chapter 1

Introduction

1.1 Objective

The purpose of this report is to document the development of a decentralized application (DApp) named ”Mini Twitter DApp” as part of the Atelier 5 project. This DApp leverages blockchain technology using Solidity for the backend (smart contracts) and a frontend developed in HTML, JavaScript, and Web3.js.

1.2 Technologies Used

- **Solidity**: A programming language for writing smart contracts on Ethereum.
- **Web3.js**: A JavaScript library that allows interaction with the Ethereum blockchain.
- **HTML/CSS/JavaScript**: For frontend user interface.

Chapter 2

Solidity Smart Contract

The smart contract is the backend of the Mini Twitter DApp. It handles the creation, editing, liking, and commenting on posts. The contract has been implemented in Solidity and deployed on a local blockchain.

2.1 Contract Overview

The contract includes the following main functionalities:

- **publishPost**: Allows users to create new posts.
- **editPost**: Enables the author to edit their post.
- **likePost** and **dislikePost**: Allow users to like or dislike posts.
- **addComment**: Allows users to add comments to posts.
- **likeComment**: Enables users to like a comment.

2.2 Code Explanation

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

contract MiniSocial {
    event NewPostEvent(uint indexed postId, string message, address indexed author, uint timestamp);
    event PostUpdatedEvent(uint indexed postId, string newMessage, uint lastModified);

    struct Comment {
        address author;
        string content;
        uint numLikes;
    }

    struct Post {
        uint id;
        string message;
        address author;
        uint numLikes;
        uint numDislikes;
        uint timestamp;
        uint lastModified;
        Comment[] comments;
    }

    Post[] private posts;
    mapping(uint => mapping(address => bool)) private hasLiked;
    mapping(uint => mapping(address => bool)) private hasDisliked;

    function publishPost(string memory _message) public { ... }
```

```
    function editPost(uint _postId, string memory _newMessage) public { ... }
    function likePost(uint _postId) public { ... }
    function dislikePost(uint _postId) public { ... }
    function addComment(uint _postId, string memory _content) public { ... }
    function likeComment(uint _postId, uint _commentId) public { ... }
}
```

Chapter 3

Frontend Development

The frontend is a web-based interface allowing users to interact with the smart contract. It includes functions for connecting to MetaMask, creating, editing, liking, disliking, and viewing posts.

3.1 HTML and JavaScript Code

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Mini Twitter DApp</title>
    <script src="https://cdn.jsdelivr.net/npm/web3@latest/dist/web3.min.js"></script>
</head>
<body>
    <h1>Mini Twitter DApp</h1>
    <button id="connectWalletBtn" onclick="connectWallet()">Connect Wallet</button>
    <p id="userAddress"></p>
    <h2>Add a New Post</h2>
    <textarea id="postMessage" placeholder="What's on your mind?" maxlength="140"></textarea>
    <button onclick="addPost()">Post</button>
    <h2>Feed</h2>
    <div id="feed"></div>

    <script type="text/javascript">
        let web3;
        let contract;
        const contractAddress = 'YOUR_CONTRACT_ADDRESS';
        const abi = [...];

        async function connectWallet() {
            if (window.ethereum) {
                web3 = new Web3(window.ethereum);
                await window.ethereum.request({ method: 'eth_requestAccounts' });
                contract = new web3.eth.Contract(abi, contractAddress);
                loadPosts();
            }
        }

        async function loadPosts() { ... }
        async function addPost() { ... }
        async function likePost(postId) { ... }
        async function dislikePost(postId) { ... }
        async function editPost(postId) { ... }
    </script>
</body>
</html>
```

The full source code of the project, including the Solidity smart contract and the frontend application, can be found at the following GitHub repository:

https://github.com/OmarNouih/BLOCKCHAIN

Chapter 4

Deployment and Testing

The application was deployed on a local Ethereum test network using MetaMask and Ganache. The contract was tested for various functions, including publishing, editing, liking, and commenting on posts.

4.1 Screenshots

Below are screenshots of the DApp in action.

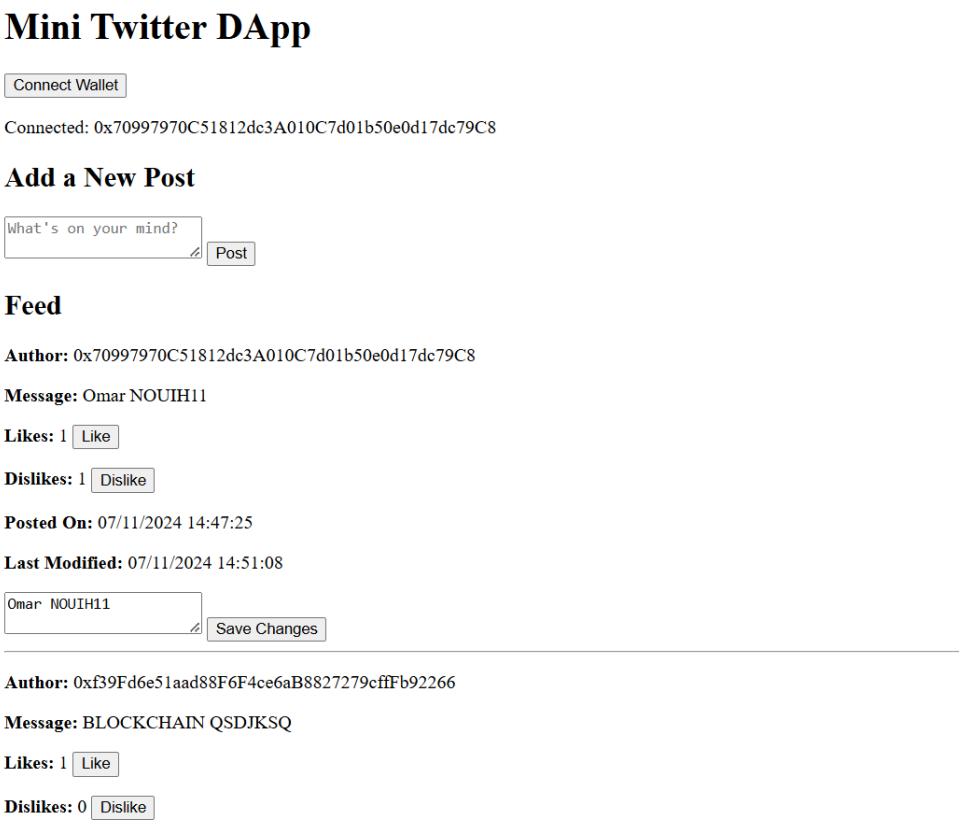

Figure 4.1: Screenshot of the Mini Twitter DApp interface.

Figure 4.2: Example of a post with likes, dislikes, and editing options.

Chapter 5

Conclusion

This report details the development of a blockchain-based Mini Twitter DApp that uses Solidity for smart contract development and a web-based frontend using Web3.js. The DApp allows users to create posts, edit them, add likes and dislikes, and add comments, all stored immutably on the blockchain.