CS5363 Blockchain Technologies and Applications: Hw8 – Report
Decentralized App (DApp)

Name: 卓岳霆
Student Id: 110062236

Task: A DApp on Web

Introduction

我的期末報告是要建立一個輕量級的跨鏈橋以降低延遲和複雜性，因此我在這份作業我實作了一個名為 ProofOfText 的簡化智能合約，它的核心功能是演示「訊息鎖定與查詢」機制，模擬跨鏈訊息記錄的基本功能。所以我建立了一個 Web DApp，使用者可以透過瀏覽器連接錢 MetaMask 錢包，並連接到 Sepolia 測試網路，輸入文字訊息，呼叫合約的 lockMessage 功能將其記錄在鏈上（並支付少量測試 SepoliaETH），並能查詢已記錄的訊息。

Design your contracts & your web app

合約 ProofOfText.sol:

```
// State variable: Mapping from user address to their last stored message
mapping(address => string) private userMessages;

// Event: Emitted when a message is successfully locked by a user
event MessageLocked(address indexed user, string message, uint value);
```

mapping 來儲存每個地址對應的最新訊息。MessageLocked 事件在訊息成功儲存時觸發，以便鏈下應用追蹤。

```
function lockMessage(string calldata text) external payable {
    address sender = msg.sender; // Get the address of the function caller
    userMessages[sender] = text; // Store the text, overwriting any previous message from this sender

    // Emit an event to log the action on the blockchain
    emit MessageLocked(sender, text, msg.value); // msg.value is the amount of ETH sent with the transaction
}

/**
 * @notice Retrieves the last message stored by a specific user.
 * @dev Reads from the userMessages mapping for the given user address.
 * This is a view function, meaning it doesn't cost gas to call from off-chain and doesn't modify state.
 * @param user The address of the user whose message to retrieve.
 * @return string The last message stored by the user. Returns an empty string if the user never stored a message.
 */
function getMessage(address user) external view returns (string memory) {
    return userMessages[user]; // Return the message associated with the user address
}
```

lockMessage 函式: 將 text 存入 userMessages[msg.sender]，成功後觸發 MessageLocked 事件。

getMessage 函式: 從 userMessages 讀取並回傳該地址對應的 string memory。此函式為 view，不消耗 Gas (若從鏈下調用) 且不修改狀態。若地址無記錄則返回空字串。

Web App:

frontend/組成: 由 index.html，style.css 和 index.js 組成。

核心邏輯 (index.js):

錢包連接: 使用 ethers.js (v6/v5) 提供的 BrowserProvider 透過 window.ethereum (MetaMask) 請求用戶連接錢包 (eth_requestAccounts) 並獲取 signer 物件。

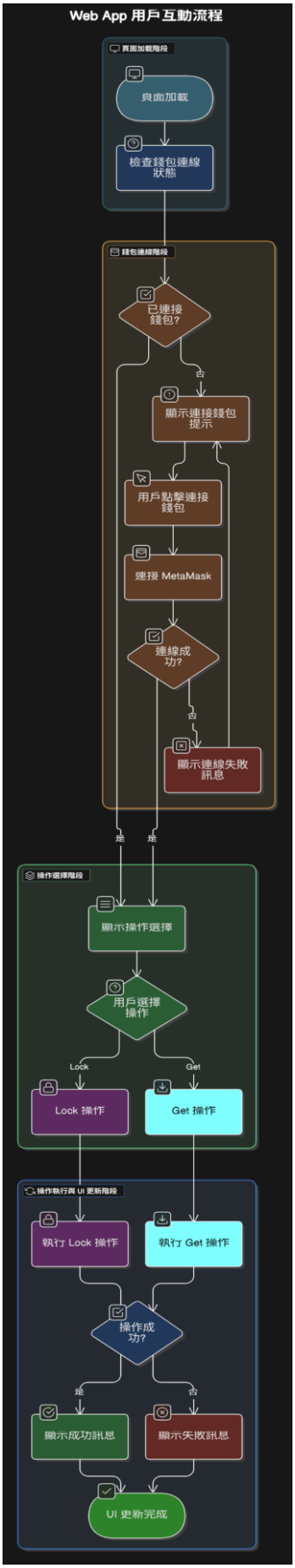
合約實例化: 使用 Sepolia 部署地址和合約 ABI (new ethers.Contract(...)) 創建可互動的合約物件。

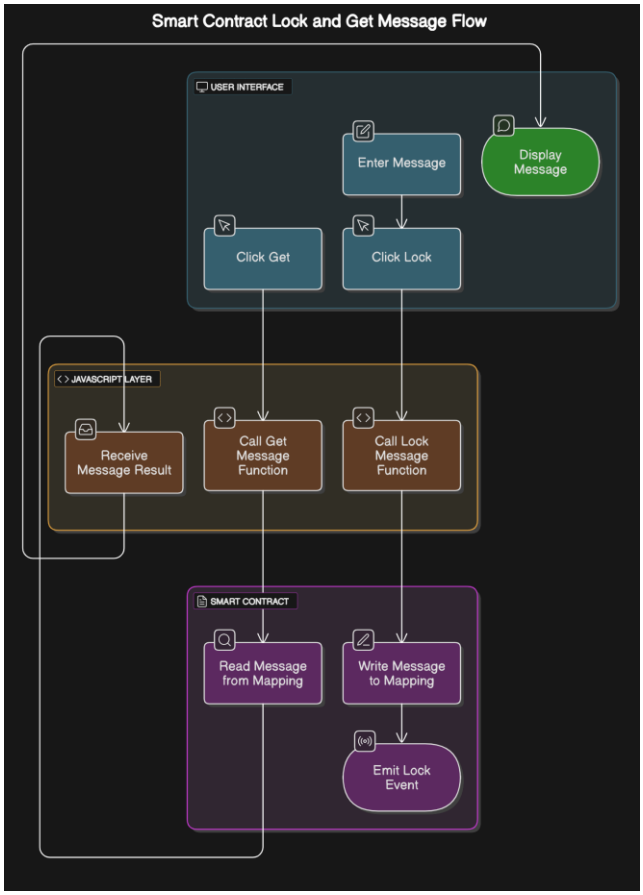
鎖定訊息: 監聽 "Lock Message" 按鈕點擊。觸發時，讀取 #msg 輸入框內容，調用 contract.lockMessage，彈出 MetaMask 要求用戶確認交易 (包含少量 Sepolia ETH)。交易過程中更新 #status 區域的文字和顏色。

查詢訊息: 監聽 "Get My Last Message"按鈕點擊。觸發時，獲取當前 signer 的地址，調用 contract.getMessage(userAddress)，將返回的訊息顯示在 #retrievedMessage 區域並更新。

狀態顯示: #status 用於顯示錢包連接狀態、交易進度與結果。 #retrievedMessage 用於顯示查詢結果。透過變更文字內容和 CSS class 提供視覺回饋。

Diagram & Flow Chart


Web App 用戶互動流程
頁面加載階段
頁面加載
檢查錢包連線狀態
錢包連線階段
已連接錢包?
否
顯示連接錢包提示
用戶點擊連接錢包
連接 MetaMask
連線成功?
否
顯示連線失敗訊息
是
是
操作選擇階段
顯示操作選擇
用戶選擇操作
Lock
Get
Lock 操作
Get 操作
操作執行與 UI 更新階段
執行 Lock 操作
執行 Get 操作
操作成功?
是
否
顯示成功訊息
顯示失敗訊息
UI 更新完成


Smart Contract Lock and Get Message Flow
USER INTERFACE
Enter Message
Display Message
Click Get
Click Lock
JAVASCRIPT LAYER
Receive Message Result
Call Get Message Function
Call Lock Message Function
SMART CONTRACT
Read Message from Mapping
Write Message to Mapping
Emit Lock Event

Test your contracts & your web app

Smart Contract Unit Testing

- 使用 Hardhat 框架，配合 Mocha 與 Chai。
- 執行 npx hardhat test 驗證合約邏輯：訊息儲存、ETH 接收、訊息覆蓋、事件觸發、訊息獨立性。

Local End-to-End Testing

- 啟動本地 Hardhat 節點 (npx hardhat node)，部署合約。
- 使用 MetaMask 連接本地節點。
- 透過 DApp 介面進行鎖定與查詢訊息測試，確認交易及 UI 反饋。

Public Testnet End-to-End Testing

- 合約部署至 Sepolia 測試網。
- 前端部署至 Netlify。
- 使用 MetaMask 在公開 DApp URL 上執行功能測試，確認操作流程與結果。

Release your contracts & your web app

合約部署至 Sepolia 測試網

- 本地測試後，使用 Alchemy/Infura 提供的 RPC URL，MetaMask 帳戶領取 Sepolia ETH。
- 設置環境變數於.env 並更新 hardhat.config.js。
- 執行部署指令：

npx hardhat ignition deploy ./ignition/modules/ProofOfTextModule.js --network sepolia

- 獲取合約地址並透過 Etherscan 驗證。

前端部署至 Netlify

- 更新 frontend/index.js 的合約地址與 ABI。

- 將 frontend 資料夾透過 Netlify Drop 部署。
- 獲得公開 URL 後進行功能驗證。

已部署合約資訊

- 網路：Sepolia Testnet
- 合約地址：0x87f728AdA36c5A35C9D69466F765aa7998F8e3D5
- 合約 URL:
- https://sepolia.etherscan.io/address/0x87f728AdA36c5A35C9D69466F765aa7998F8e3D5
- DApp URL：Netlify App

Other Discussion

此 MVP 的潛在未來增強功能 (Potential Future Enhancements for this MVP)：

- 在前端介面中實現更全面的錯誤處理機制和更友好的使用者回饋。
- 增強前端功能，允許使用者查詢任意指定地址的訊息，而不僅僅是當前連接帳戶的訊息。
- 為 lockMessage 函式附帶的 ETH 設計實際用途，例如作為操作手續費，或結合其他功能，並提供相應的、授權方可執行的提款功能。
- 引入基本的擁有權或權限控制邏輯，例如，僅允許訊息的原始儲存者更新其訊息。