# Blockchain Supply Chain Management Smart Contract Report

**Introduction**

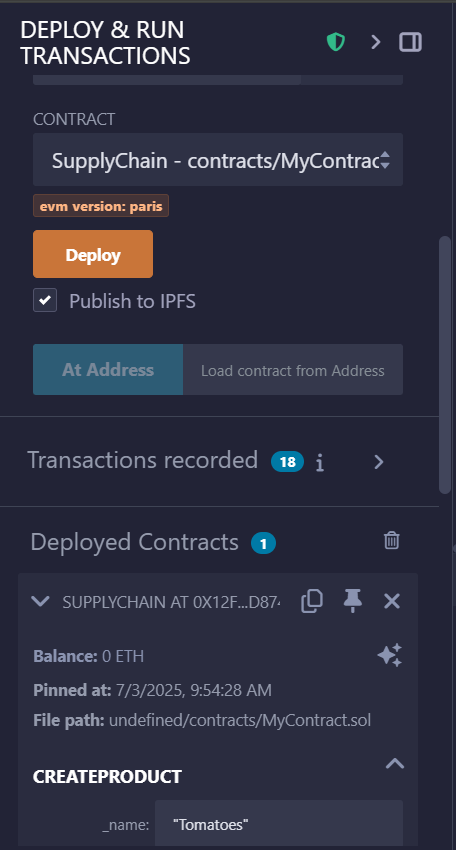
In this paper, the author provides the creation of a smart contract in the form of a blockchain supply chain manager. The deal increases accountability and transparency because every single phase of a product life cycle is reflected in the Ethereum blockchain.

**Description of Use Case**

The smart contract of the supply chain allows the tracking of products during supply chain process between manufacturer and retailer. Some of the notable tasks are; producing a product, delivering it to a new customer and verifying possession. The transfer of ownership is recorded in every transition and all the activities are captured on the blockchain.

**A Guide to Smart Contracts Code**

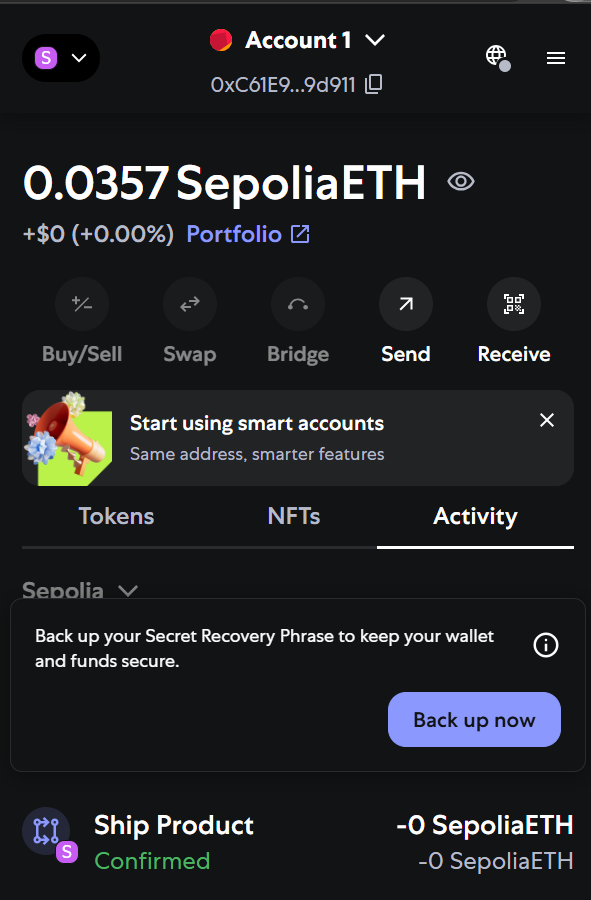
The contract is written using Solidity along with enums structs and mappings. It introduces an enum Status { Created, Shipped, Received } and a type struct Product that needs to store product data. Functions include modifiers and have require() statements which are used in access control.



*Shows the deployed contract using Remix.*

**Deployment Instructions**

* Launch Open Remix IDE and paste the code of the smart contract in a new Solidity file.
* Compile with Solidity compiler.
* Select the ENV to be Injected Web3 to connect to MetaMask.
* Send adequate ETH to the Sepolia testnet using the MetaMask.
* Once it is deployed, one can use Remix or front-end interface to communicate with the contract.



*MetaMask window showing connected wallet and Sepolia balance*.

**Interaction Guide**

The following are some of the public functions that are exposed in the contract:

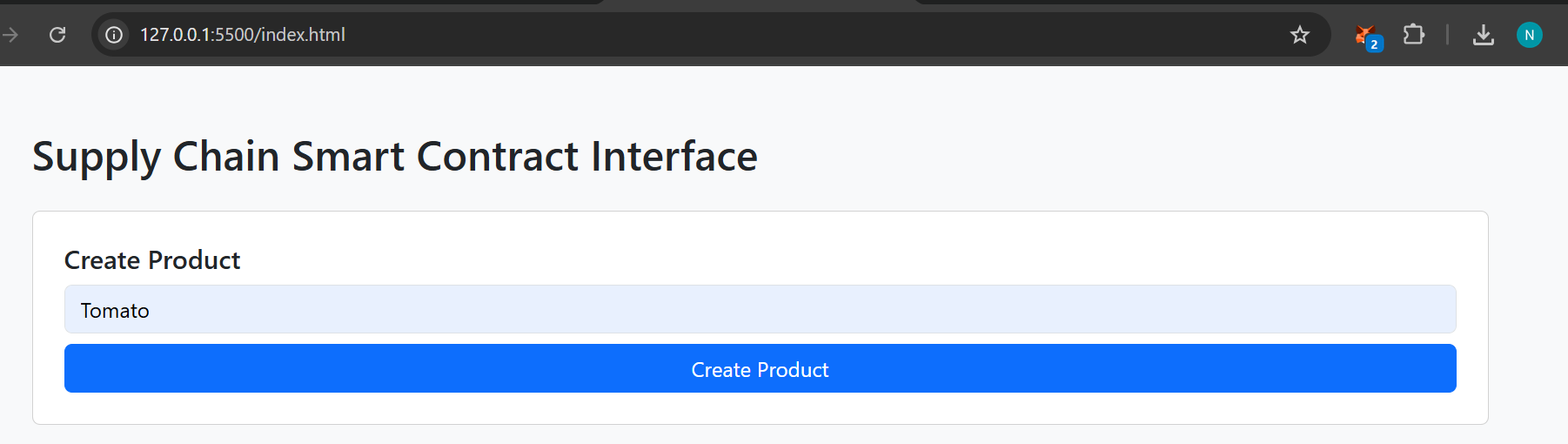
- createProduct(string \_name): It creates a new product.

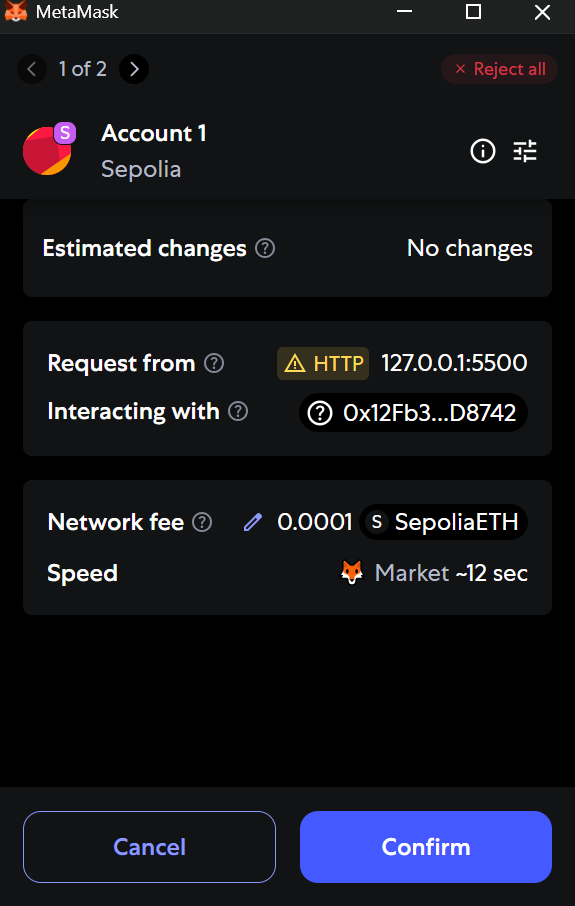
-shipProduct(uint256 \_id, address \_to): It sells the product to a new owner.

- receiveProduct(uint256 \_id): indicates that the product has been received.

- products(uint256 \_id): Provides data of a particular product.

Every transaction is registered on blockchain and it can be checked through Remix or MetaMask.





*Proof that transaction was initiated and accepted by the user.*

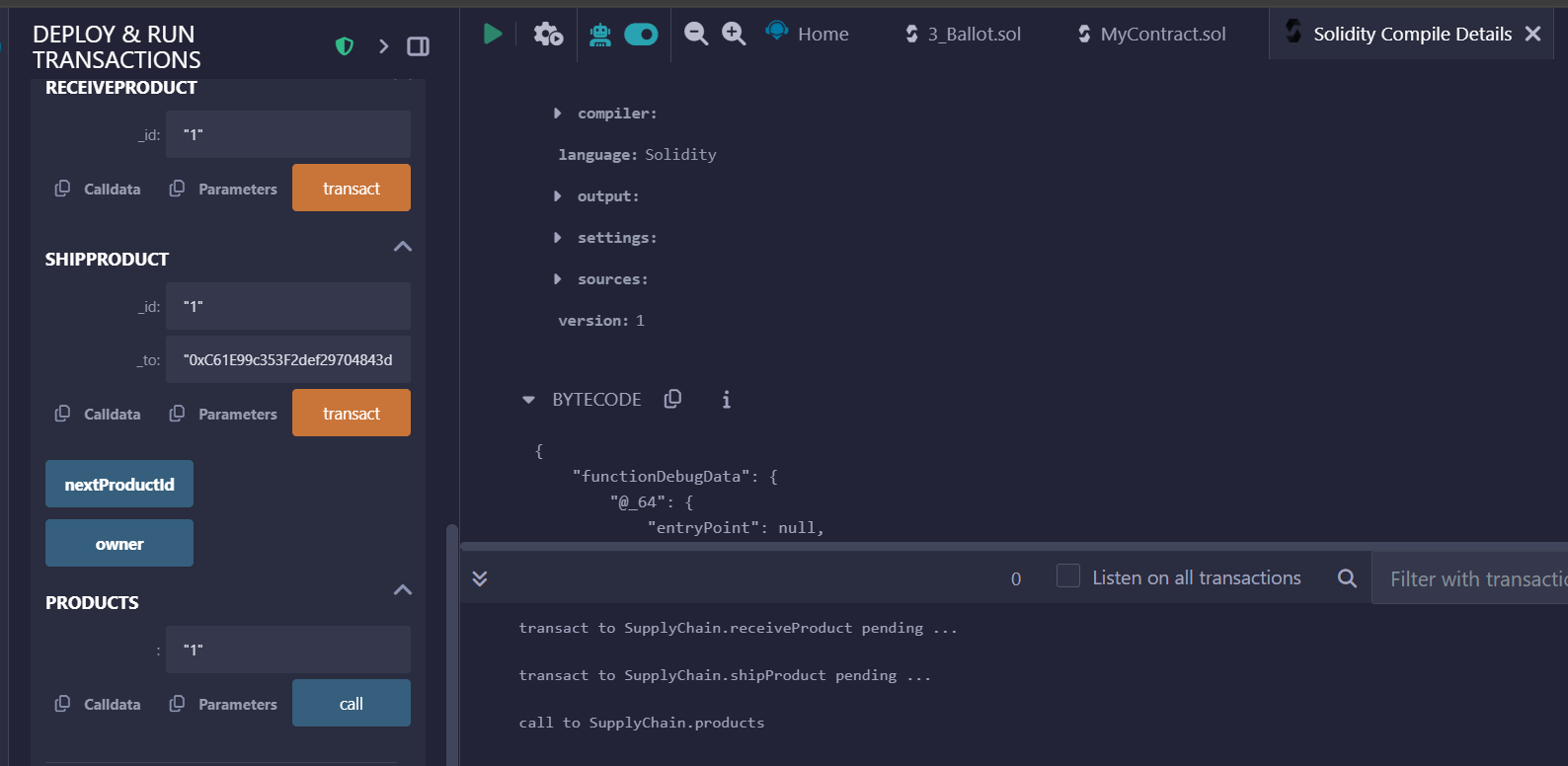
**Security Considerations**

The agreement installs some important security policies:

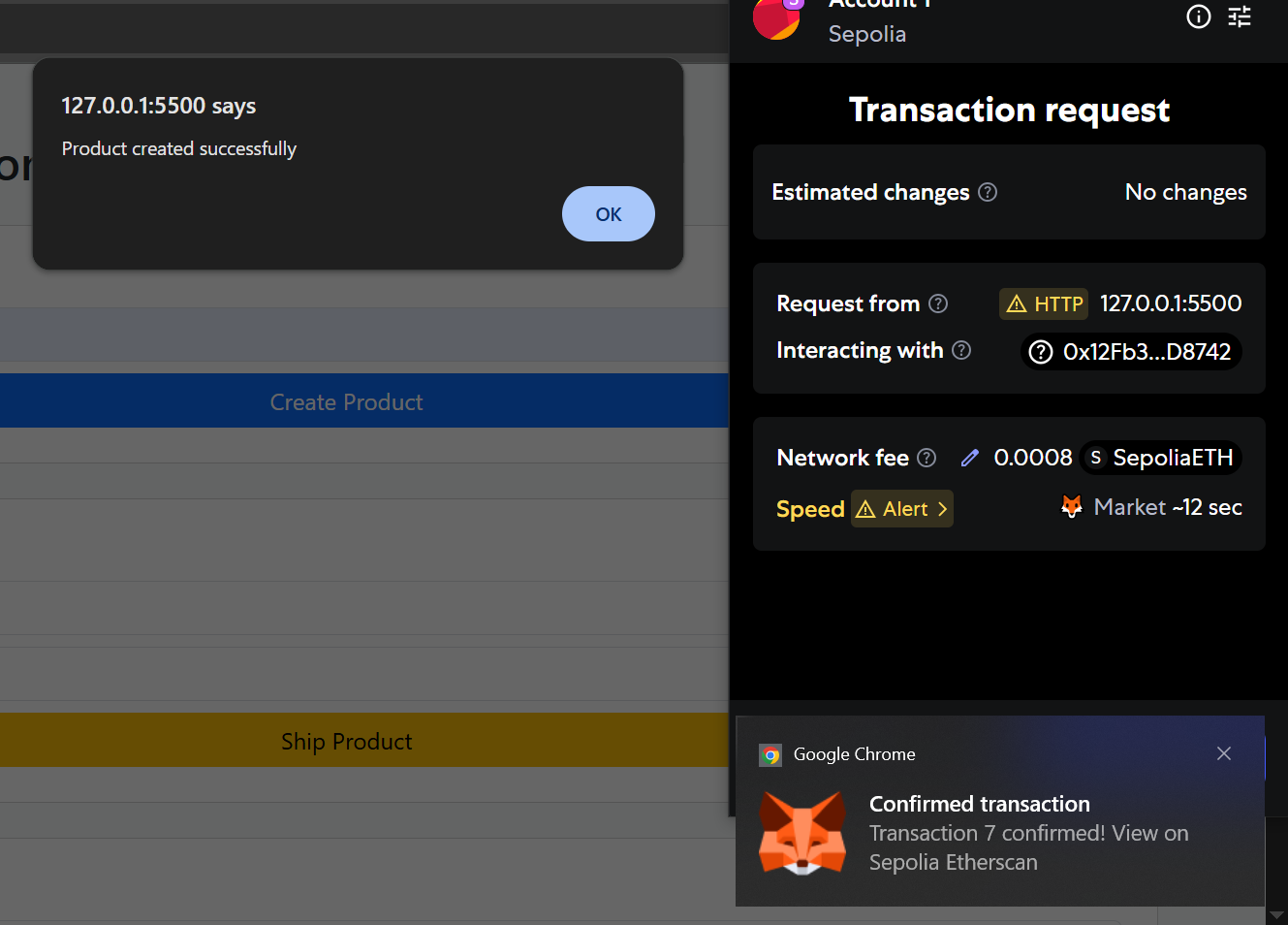
* Supply require() in order to validate and control access.
* OnlyOwner modifier was applied to limit some functions.
* Published events (ProductCreated, ProductShipped, ProductReceived) in order to achieve transparency and tracking.

**Output and test results**

To test all functions (createProduct, shipProduct, receiveProduct), it was used on the Sepolia staging environment MetaMask. Transactions were made successfully and upgraded the status of the products. The front-end has been connected successfully and real transaction has been sent using front-end.



*Shows function calls and return data (e.g., pending → success).*

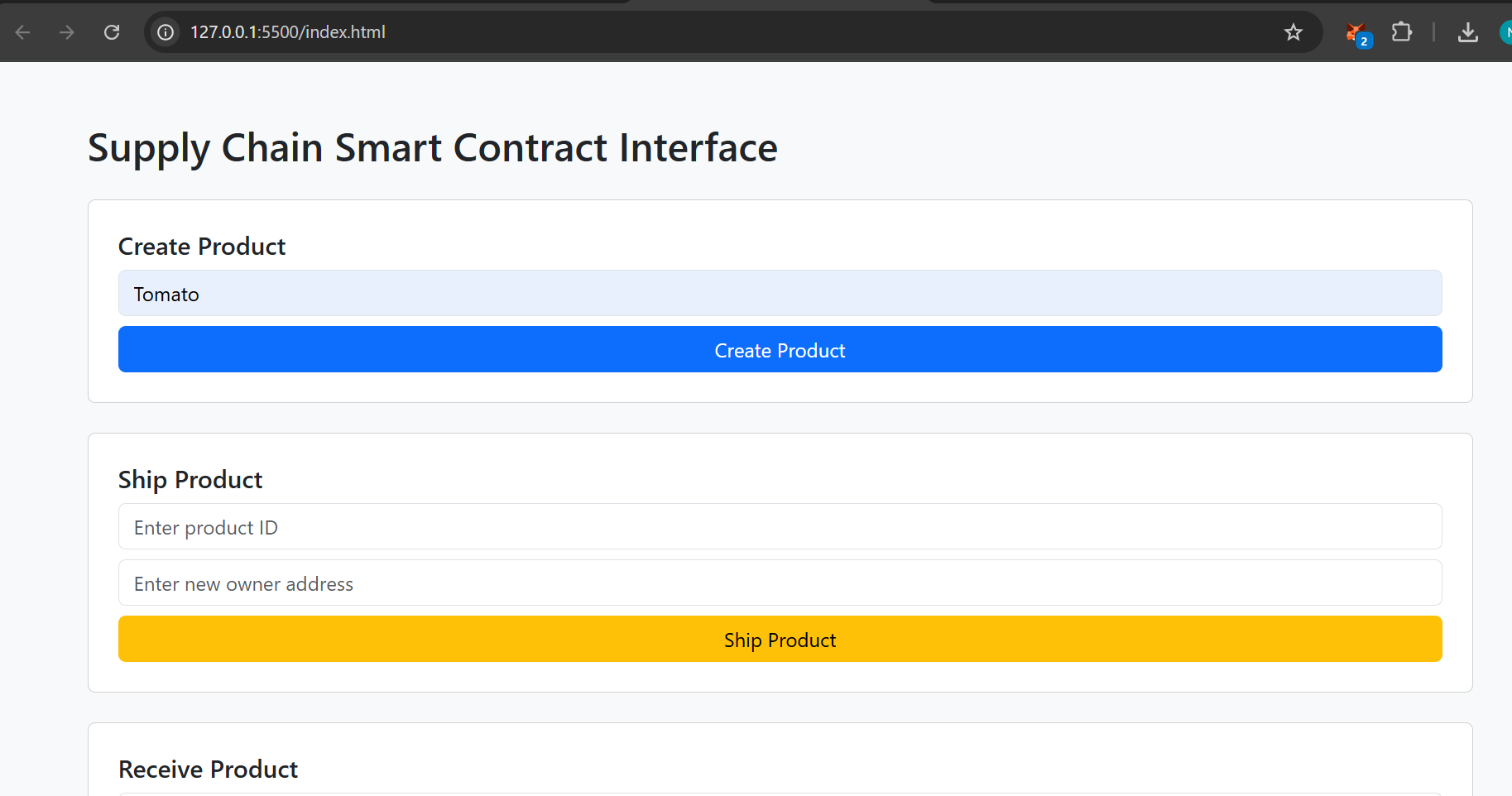


*Confirms create/ship/receive product worked from UI.*

**Summary of Front-End Interface**

HTML, Bootstrap, and Ethers.js are utilized in the design of the front-end interface. It adds smart contract interaction forms. One can also hook their MetaMask wallet and engage in activities such as creating, shipping and receiving products.

The meta mask was attached to Sepolia network, and the front-end was working correctly with all operations passing the expected response.



*Demonstration of connection of dApp UI to blockchain.*

**Additional Development Plan**

The third step will be to place the front-end interface online with GitHub Pages or Netlify and build it into a complete dApp. A future enhancement can consist of:

* Monitoring of events in real time
* History and product status filters
* Improvement of UI/UX
* Authentication of the user

**Presentation Script**

Good afternoon,

My name is Emmanuel Njunge, and I’m going to walk you through a blockchain solution I developed for a Supply Chain Management use case. The purpose of this smart contract is to ensure transparency and accountability in the transfer of products from manufacturers to retailers.  
  
The contract is written in Solidity and deployed on the Sepolia test net using Remix and MetaMask. It defines a product lifecycle using enums and tracks ownership with mappings and event logs. Functions allow creation, shipping, and receiving of products while enforcing access control.  
  
Security is implemented through modifiers like `onlyOwner` and `require()` checks. This prevents unauthorized access and ensures the integrity of the supply chain data on the blockchain.  
  
Finally, I documented the full development cycle and prepared this report. In the next phase, I will extend this to a full decentralized application (dApp) with a front-end interface for better user interaction.  
  
Thank you.