Bhartiya Vidya Bhavan's

Sardar Patel Institute of Technology, Mumbai-400058
Department of Computer Science and Engineering
OEIT1:Blockchain Technology and Applications

MINI PROJECT REPORT

BRANCH: CSE (DS)
BATCH: E
DATE: 30/04/2024

Submitted By

UID:	**Name:**
2021700001	Anushka Acharya
2021700007	Asmi Bhanushali
2021700009	Aditi Bhutada
2021200114	Divya Suvarna

OUTLINE

PROBLEM STATEMENT

To develop a consumer friendly product verification application using Blockchain Technology for detecting genuine products in supply chain companies using QR code scanning.

OBJECTIVES

1. Create a reliable blockchain database that is connected to QR codes that are linked to every product With the help of this system, consumers will be able to easily authenticate products by simply scanning the QR code.
2. Develop a user-friendly verification application accessible to consumers, enabling real-time authentication of products.

TECH STACK

1. Solidity:

Solidity is a programming language used for writing smart contracts on the Ethereum blockchain. Your project likely includes Solidity smart contracts for implementing the logic of identifying fake products and managing interactions between various parties on the blockchain.

2. **Truffle:**

Truffle is a popular development framework for Ethereum that simplifies the process of building and deploying smart contracts. Your project uses Truffle for compiling Solidity contracts, managing migrations, and providing a development environment for testing and deployment.

3. **Ganache:**

Ganache is a local blockchain emulator that provides a simulated Ethereum blockchain environment for development and testing purposes. Your project uses Ganache to set up a local blockchain network where you can deploy and interact with smart contracts without incurring real Ethereum transaction costs.

4. Web3.js:

Web3.js is a JavaScript library that allows interaction with Ethereum nodes. It provides an interface for connecting to Ethereum networks, deploying smart contracts, and interacting with them. Your project uses Web3.js for connecting the front end to the blockchain network, enabling users to interact with smart contracts through the web interface.

5. Metamask:

Metamask is a browser extension that allows users to interact with Ethereum-based decentralized applications directly from their web browsers. Your project integrates with Metamask to enable users to connect their Ethereum accounts, sign transactions, and interact with the smart contracts deployed on the blockchain network.

6. Node.js and npm:

Node.js and npm (Node Package Manager) are used for managing project dependencies and running the development server.

- Frontend Web Application: Project includes a frontend web application built with **HTML, CSS, and JavaScript**. This application provides a user interface for interacting with the smart contracts deployed on the **Ethereum blockchain**.
- Users can perform actions such as registering products, verifying product authenticity, and viewing product information through this interface.

Overall, the project leverages these blockchain technologies to implement a decentralized application for verifying products using blockchain technology, enabling secure and transparent product verification on the Ethereum blockchain.

METHODOLOGY

DAPP SERVER

In your project setup, there's no explicit mention of a DApp (decentralized application) server. However, it does involve serving web content through a development server, which is commonly used in DApp development for hosting the front end of the application. This development server typically serves HTML, CSS, and JavaScript files to users' browsers and facilitates interaction with the blockchain through Web3.js.

Here's a breakdown of the components and their roles:

1. Frontend (HTML, CSS, JavaScript):

- The frontend components, including HTML for structure, CSS for styling, and JavaScript for interactivity, are served to users' browsers by the development server.
- These files contain the user interface elements of your DApp, allowing users to interact with the underlying blockchain functionalities.

2. Development Server:

- The development server is responsible for serving the frontend files to users' browsers.
- It may also handle routing requests and responses between the frontend and backend components of your application, though in a decentralized context, much of the application logic may reside in the frontend.

3. Web3.js:

- Web3.js is used in the frontend JavaScript code to interact with the Ethereum blockchain.
- It allows your DApp to send transactions, read data from smart contracts, and listen for events emitted by the blockchain.

4. Metamask Integration:

- DApp integrates with Metamask, a browser extension that provides users with a secure way to manage their Ethereum accounts and interact with decentralized applications.

- Metamask allows users to sign transactions and securely communicate with the Ethereum blockchain directly from their browsers.

GANACHE TEST BLOCKCHAIN

In this project, Ganache is used as a local blockchain for development and testing purposes. Here's how we implemented it:

1. Setup:

- After installation, we launched Ganache, to start a local Ethereum blockchain environment.

2. Integration with Truffle:

- Truffle, a popular development framework for Ethereum, is used to compile, deploy, and test smart contracts.
- We configured Truffle to connect to the Ganache blockchain environment for deployment and testing.
- This connection is established by specifying the appropriate network configuration in the Truffle project's configuration file (`truffle-config.js` or `truffle.js`).

3. Deployment:

- After writing and compiling smart contracts using Truffle, we deployed them to the Ganache blockchain environment.
- Truffle migrations handle the deployment process, ensuring that smart contracts are deployed to the correct Ganache network.

Overall, Ganache serves as a local Ethereum blockchain environment for development and testing, seamlessly integrated with Truffle to streamline the smart contract development workflow in the project.

EXPERIMENTAL SETUP

CLIENT SERVICE

In our project, the client service typically refers to the frontend or client-side application that interacts with the blockchain network and smart contracts.

1. Technologies:

- HTML, CSS, and JavaScript are used to build the frontend interface.
- Libraries and frameworks such as jQuery may also be utilized for DOM manipulation and asynchronous request handling.
- Web3.js is employed to interact with the Ethereum blockchain and smart contracts from the client-side.

2. Functionality:

- The client service provides a user interface through which users can interact with the blockchain application.
- Users can initiate transactions, and view relevant information presented in a user-friendly format.
- Forms and input fields are included to gather user input required for executing transactions and invoking smart contract functions.

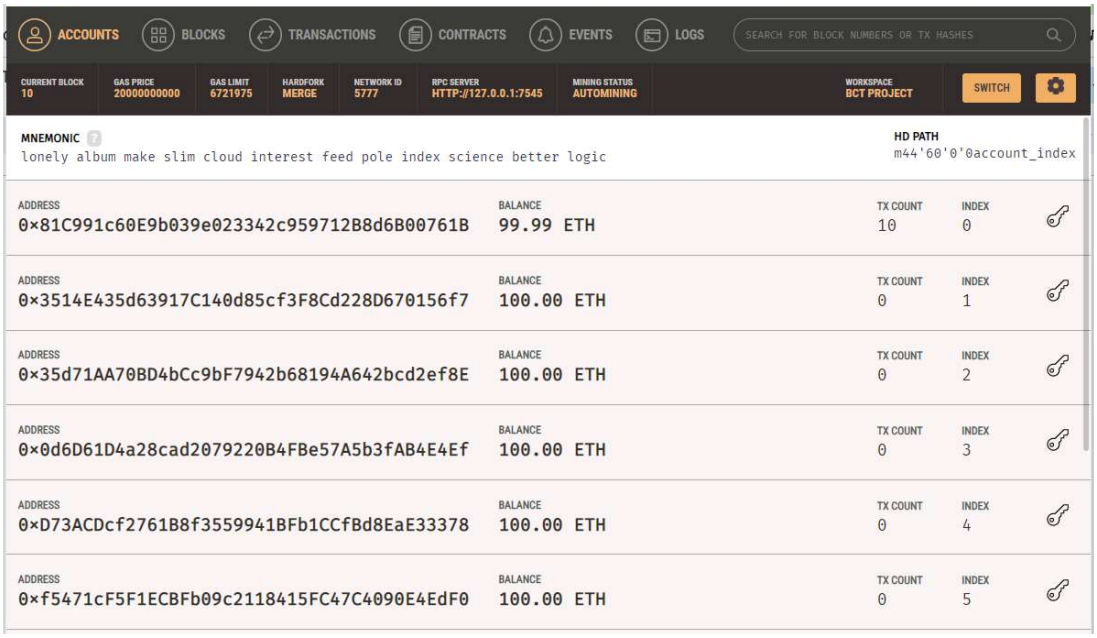
3. Integration with Blockchain:

- Web3.js is used to establish a connection between the client-side application and the Ethereum blockchain network.
- Smart contract ABI (Application Binary Interface) is provided to interact with deployed smart contracts from the client-side.
- Contract methods are accessed through Web3.js to perform transactions and call functions.

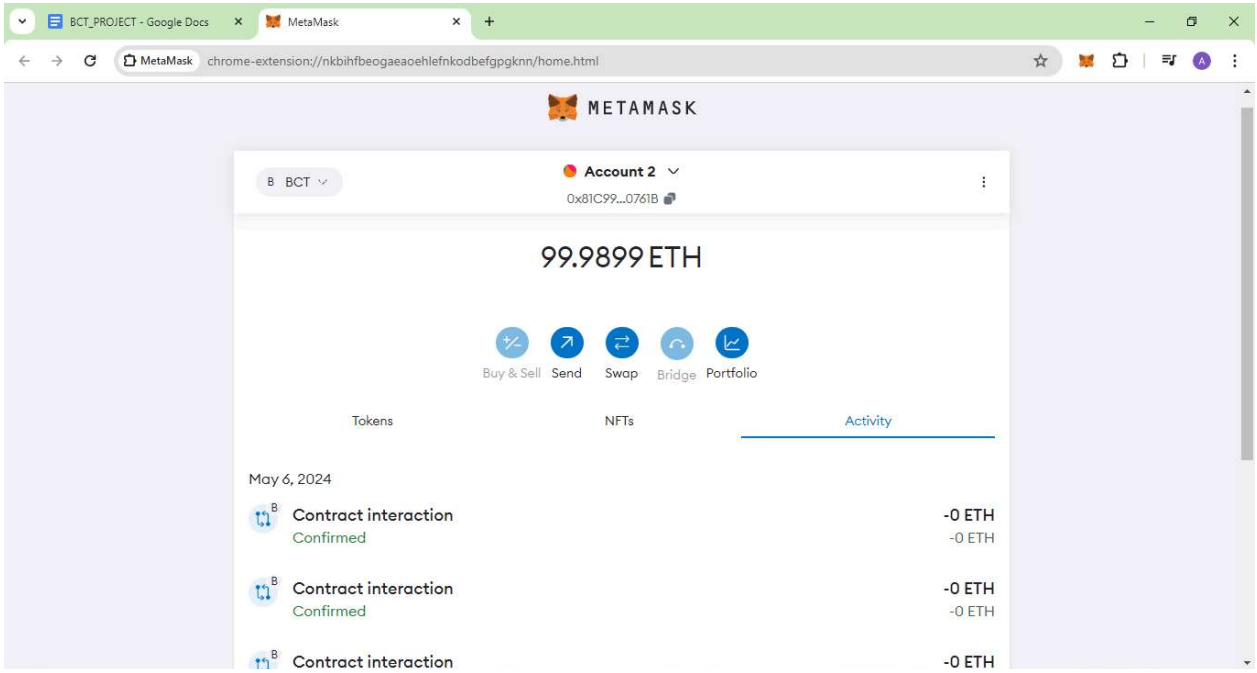
4. User Experience:

- The client service aims to provide an intuitive and seamless user experience, guiding users through the process of interacting with the blockchain application.
- User-friendly interfaces, clear instructions, and visual feedback contribute to enhancing the usability of the application.

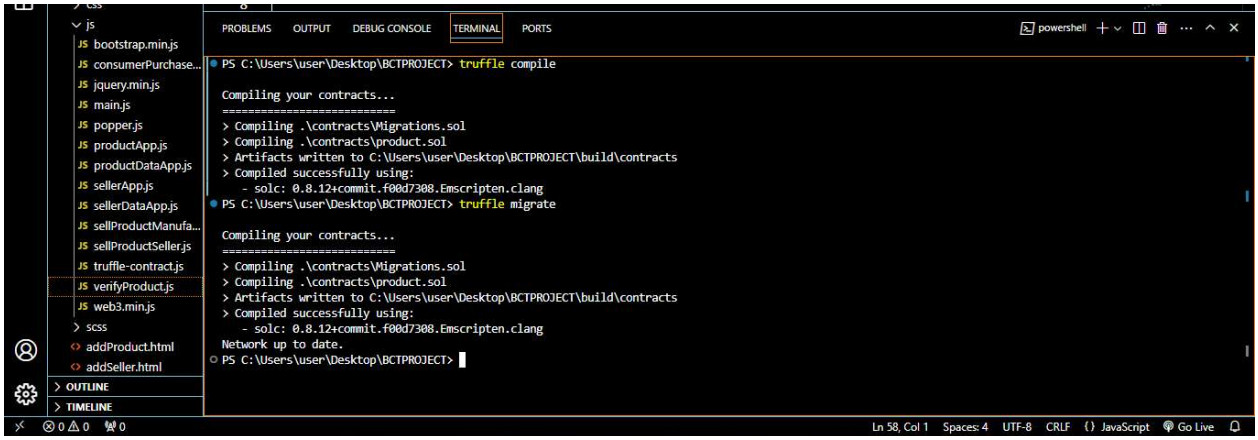
RESULT SCREENSHOTS



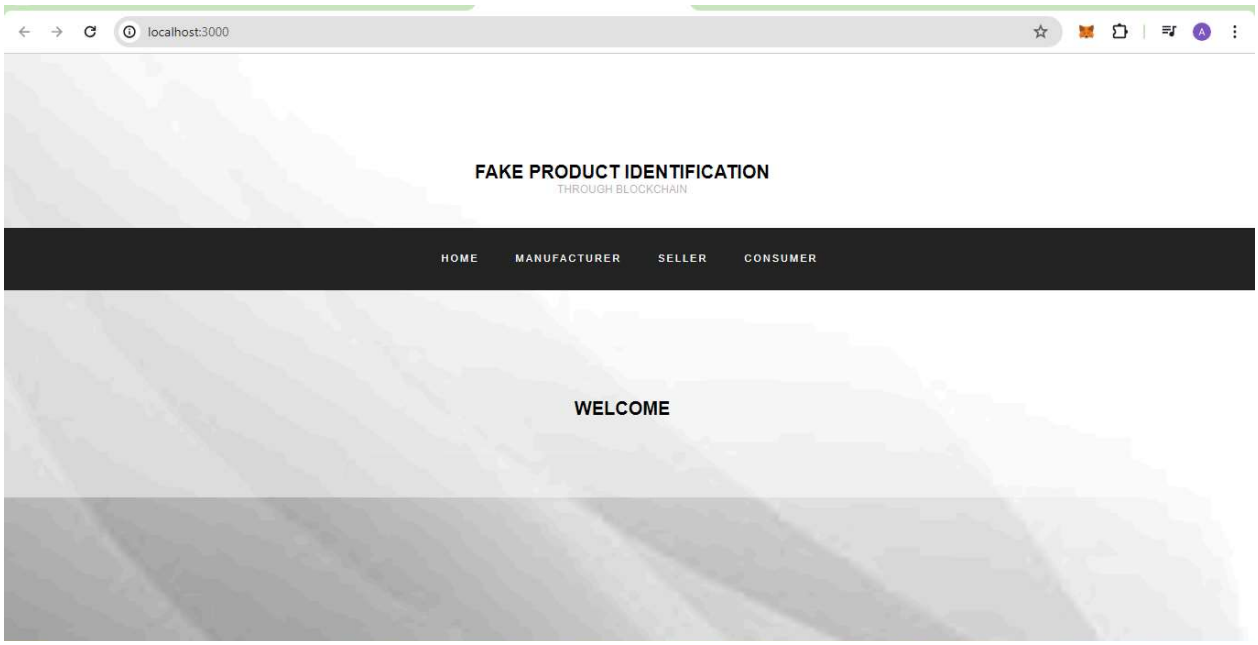
GANACHE



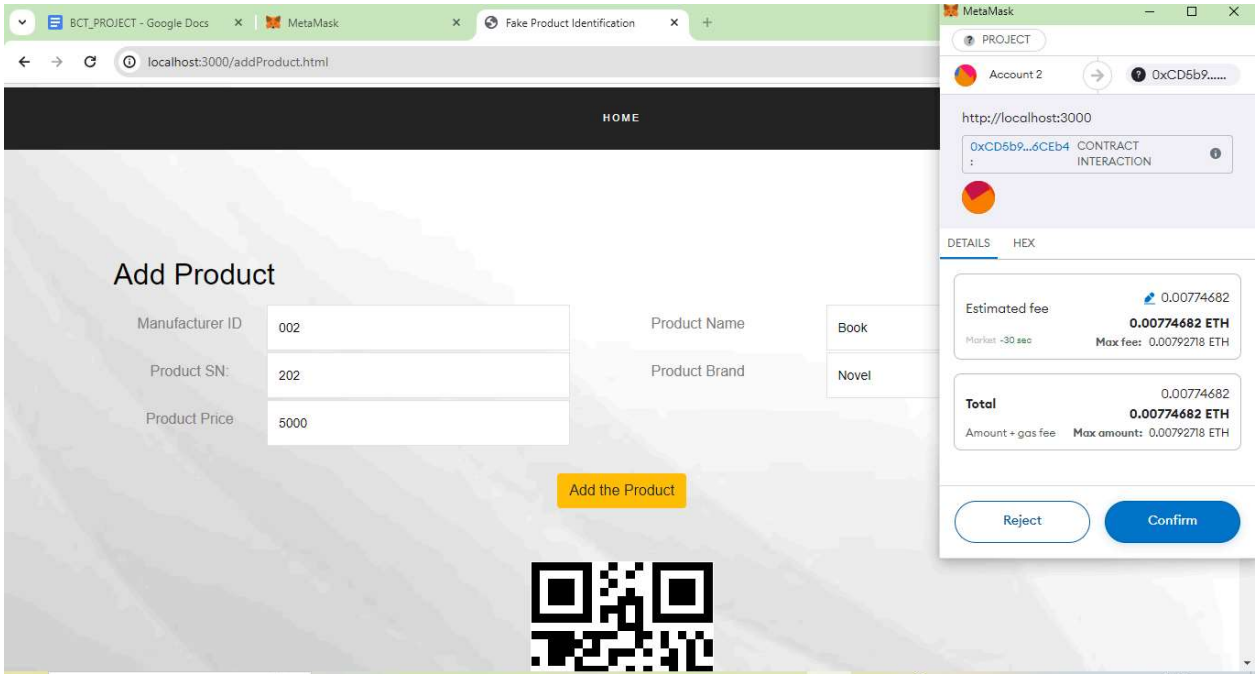
METAMASK



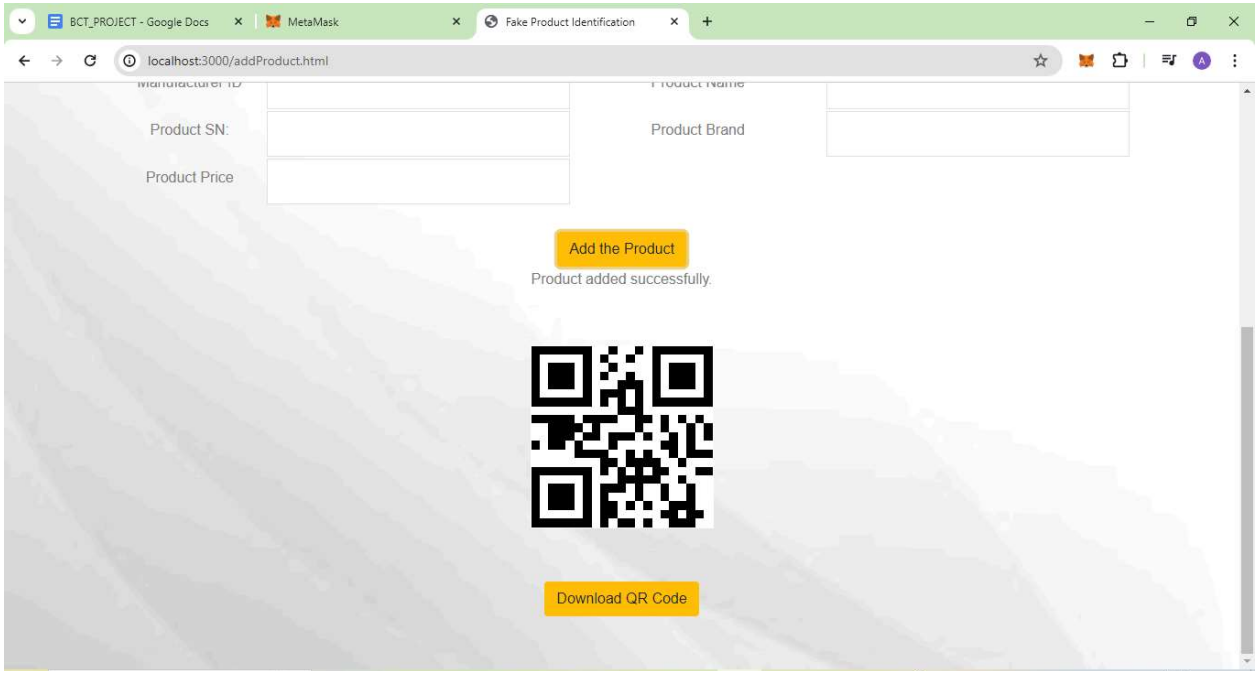
COMPILING CONTRACTS AND MIGRATING



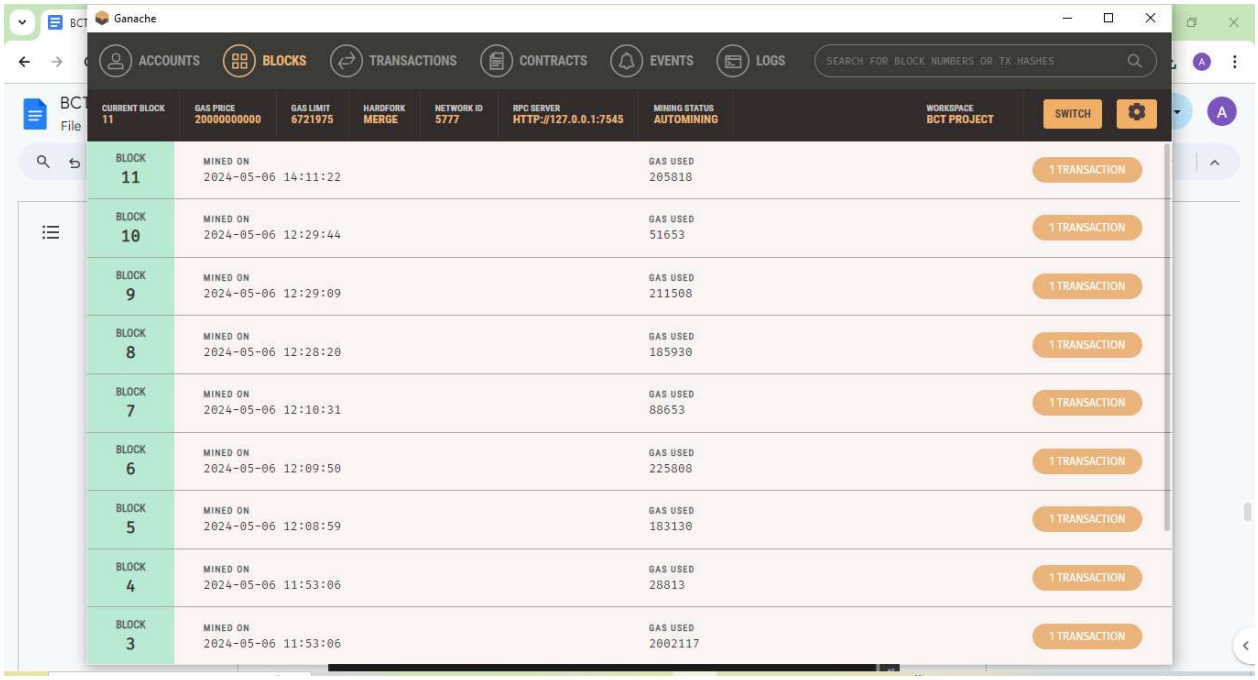
WEBSITE (HOMEPAGE) DEPLOYED ON LOCALHOST



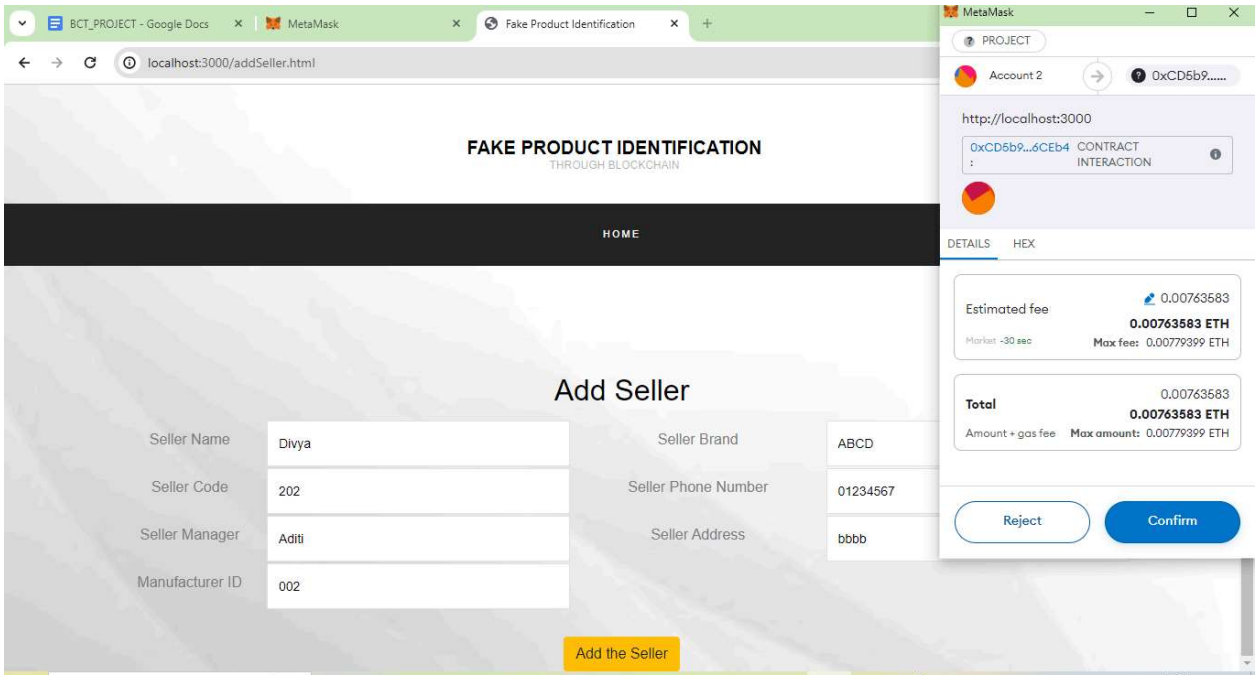
ADDING PRODUCT



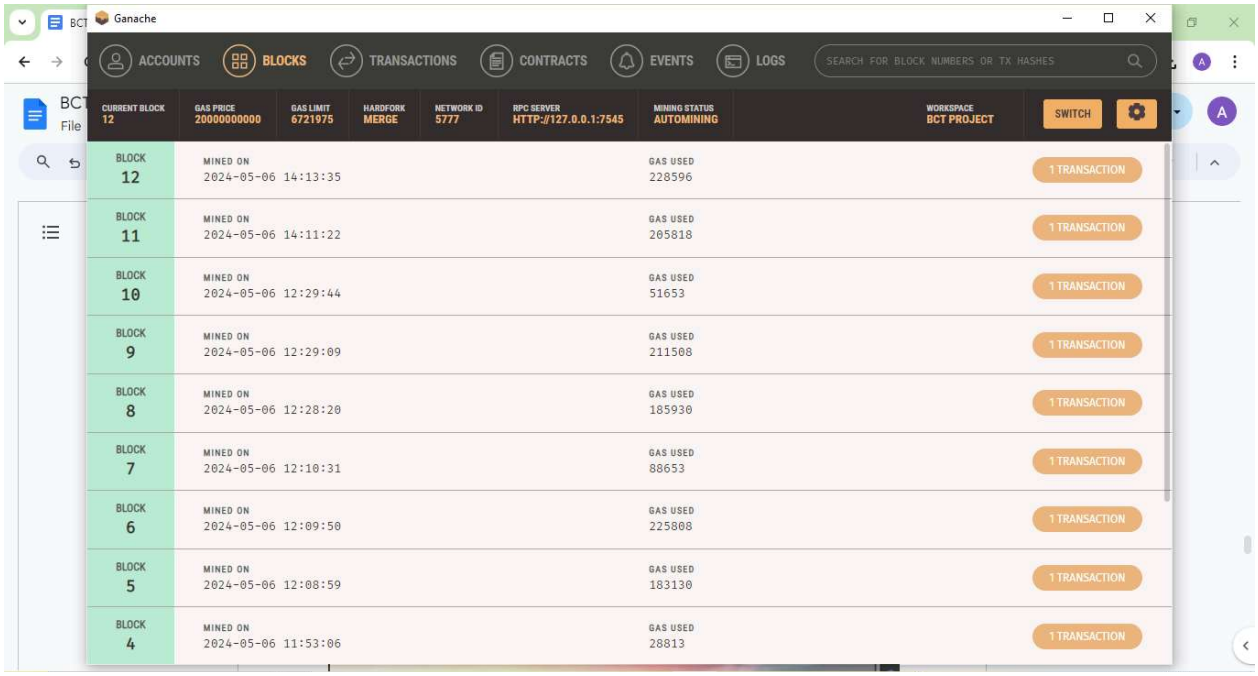
PRODUCT ADDED SUCCESSFULLY AND QR CODE GENERATED



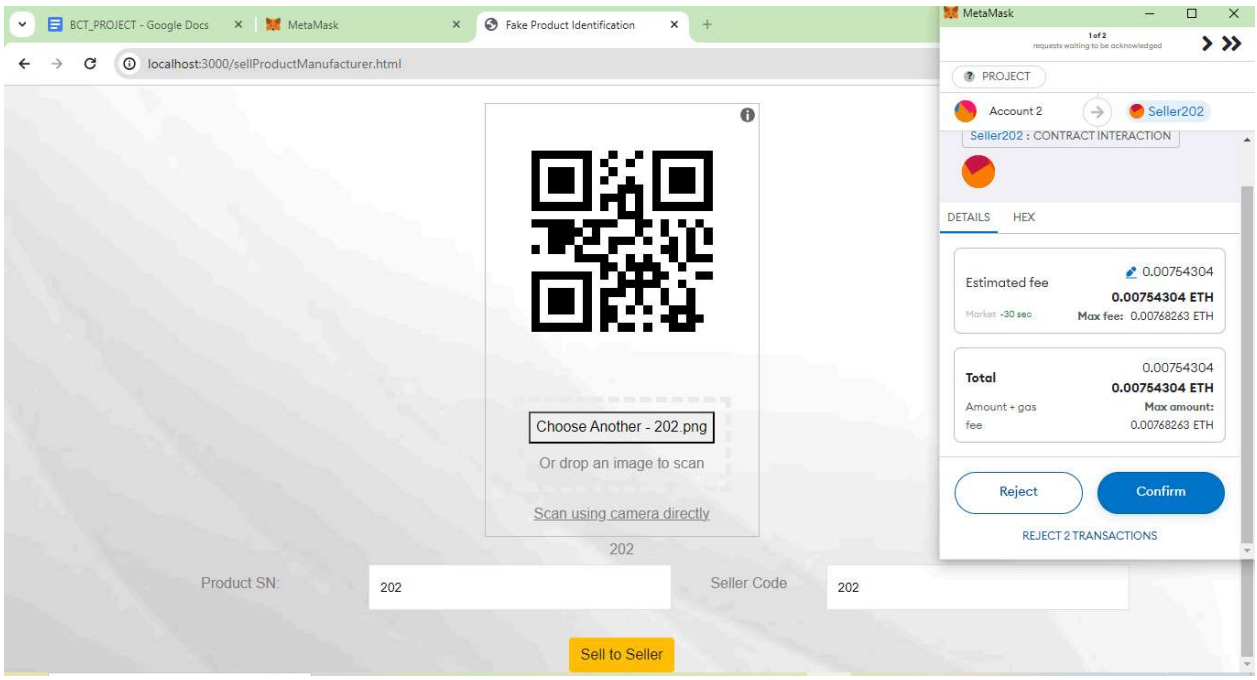
PRODUCT DATA ADDED TO BLOCKCHAIN : SUCCESSFUL TRANSACTION



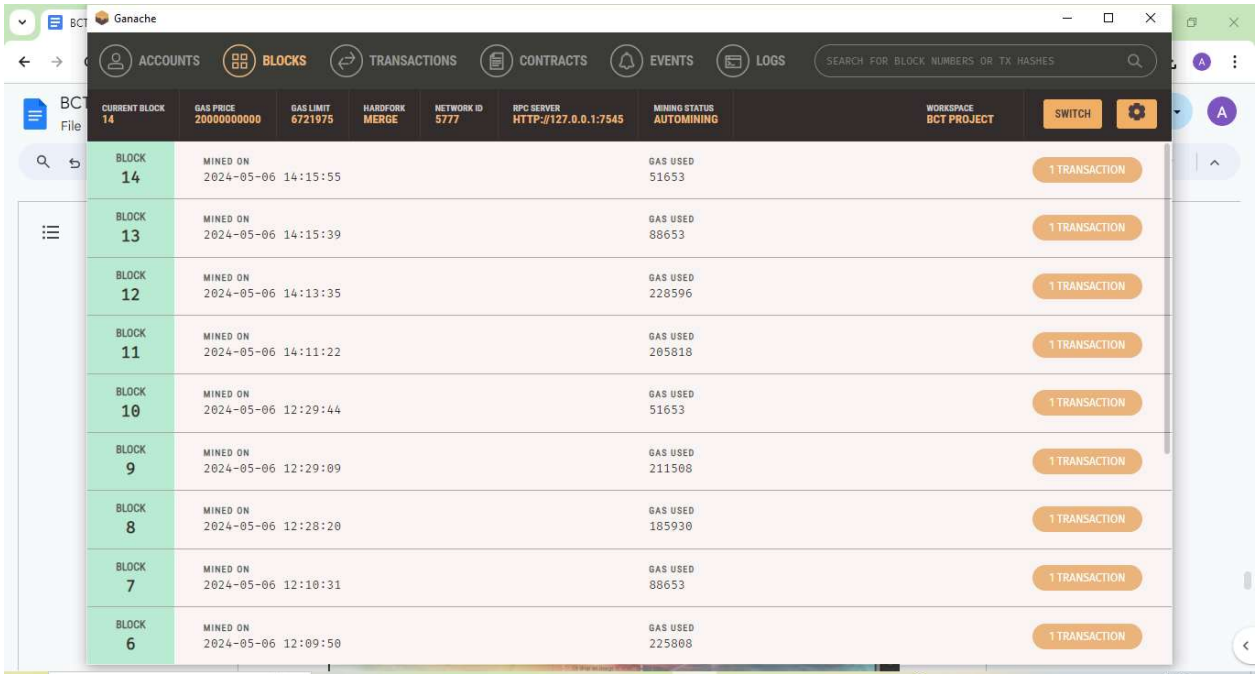
SELLER ADDED SUCCESSFULLY



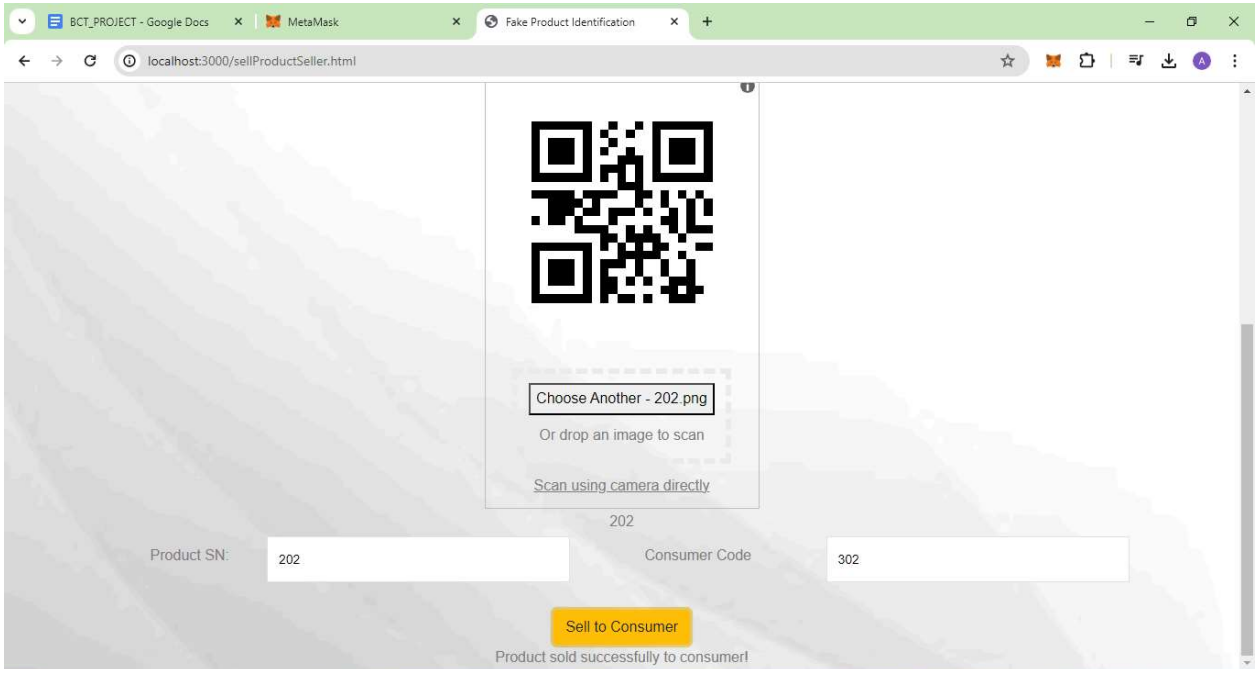
SUCCESSFUL TRANSACTION OF ADDING SELLER



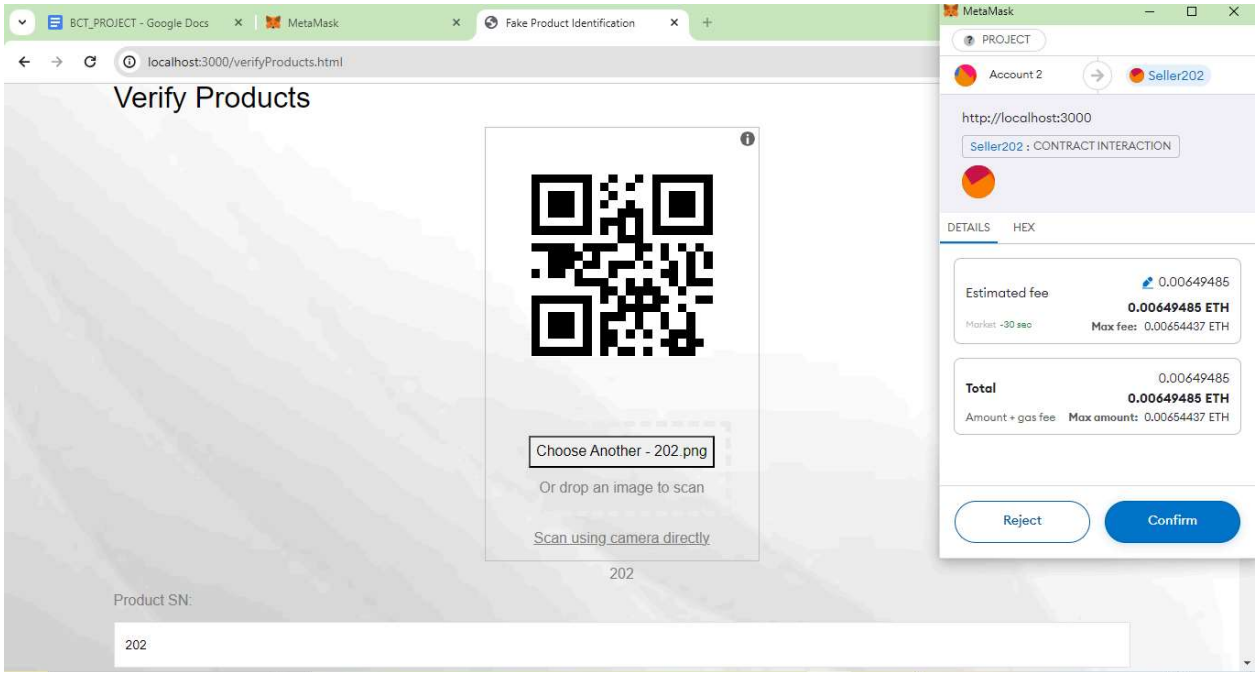
UPLOADING QR TO SELL PRODUCT TO SELLER : PRODUCT SOLD SUCCESSFULLY



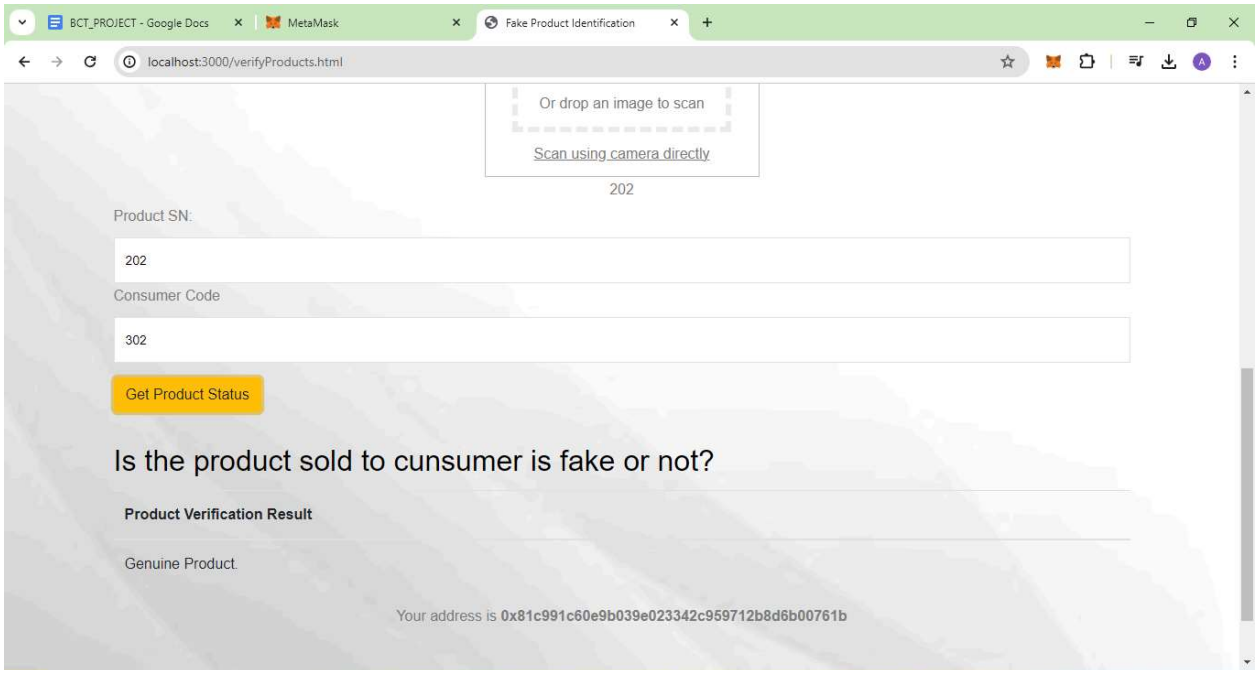
SUCCESSFUL TRANSACTION FOR SELL PRODUCT TO SELLER



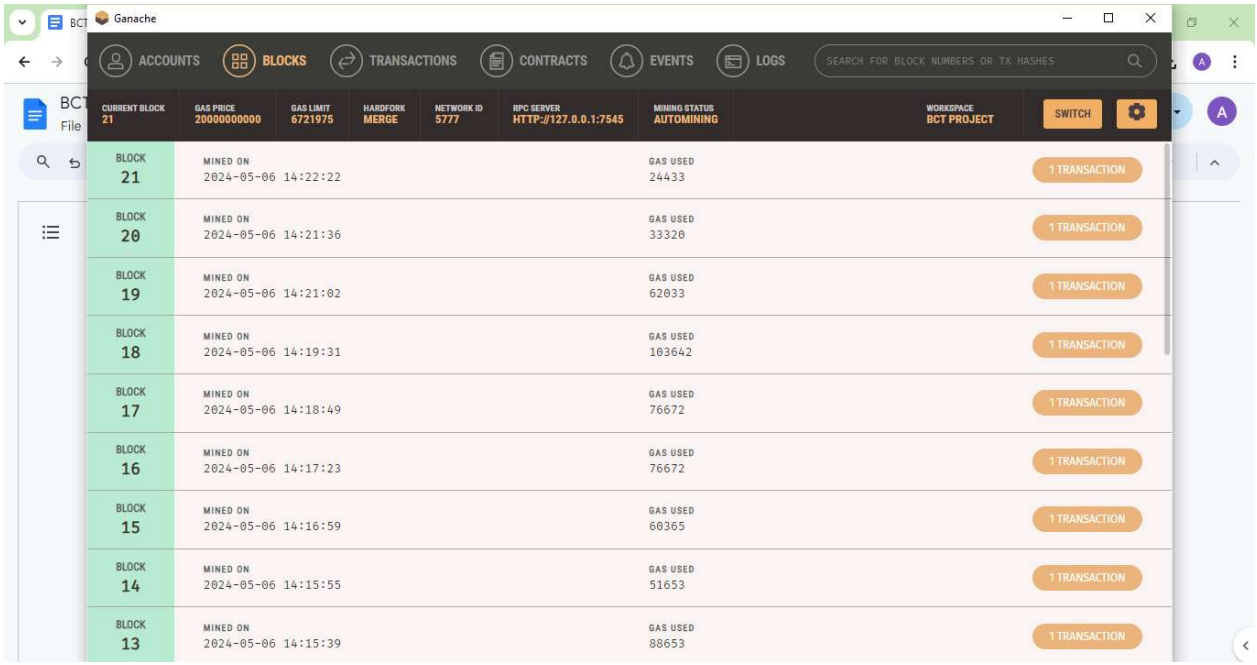
UPLOADING QR TO SELL PRODUCT TO CONSUMER : PRODUCT SOLD SUCCESSFULLY



UPLOADING QR FOR VERIFICATION



GENUINE PRODUCT DETECTED WITH ACCOUNT ADDRESS



SUCCESSFUL TRANSACTION FOR VERIFICATION

CONCLUSION

By leveraging the decentralized nature of blockchain, the project ensures that product information, including serial numbers and transaction history, is securely recorded on the blockchain ledger. This provides consumers, manufacturers, and sellers with a reliable and transparent means of verifying the authenticity of products at every stage of the supply chain.