Project Report [40 marks]

A. Team Formation : [5 marks].

B. *Progress Report [10 marks]*: Include Introduction (i.e. sections 1-2) and any other parts of your work to show progress [you can update the remaining work later in the final report]. C. *Final Report [25 marks]:* Include the sections 1-6 of the Report template. **Project Report Template (4–6 Pages)**

1. Title

- Project title: Blockchain-Based Supply Chain Tracking System (BSTS)
- Course name and code: ICS 440 – Cryptography and Blockchain Applications
- Section Number: F03

- Student name(s) and ID(s) :Wala Aljobran -202154190 Renad Alqhtani -202169930

- Instructor's name: Md Mahfuzur Rahman
- Department / University: Computer Science-KFUPM
- Date of submission 27/11

2. Introduction (1 page): <u>10 points</u>

Transparency is essential in modern supply chains because products pass through multiple intermediaries—producers, suppliers, distributors, retailers, and consumers. Without clear visibility, it becomes difficult to confirm a product's origin, handling conditions, or whether it was altered at any stage. This lack of transparency leads to inefficiencies, weak decision-making, and reduced customer trust.

A major challenge in traditional supply chains is the rise of fraud, counterfeiting, and data manipulation. Counterfeit items can enter the chain undetected, records can be falsified, and centralized databases are vulnerable to unauthorized changes. These issues can damage brand reputation, cause financial loss, and create safety risks—especially in sectors like pharmaceuticals, electronics, and food.

Blockchain provides a strong solution through **immutability and traceability**. Once data is added to the blockchain, it cannot be modified or deleted because of cryptographic hashing and decentralized storage. Every transaction is timestamped and linked to previous records, forming a permanent and verifiable audit trail. This ensures that every step—from production to final delivery—is securely documented.

By removing reliance on a central authority and enabling all parties to access the same shared ledger, blockchain reduces information asymmetry and limits opportunities for fraud. Participants can independently verify product data, track updates in real time, and trust the accuracy of the recorded information.

The **Blockchain-Based Supply Chain Tracking System (BSTS)** leverages these advantages to strengthen supply chain operations. Key benefits include:

- End-to-end traceability across all stages
- Tamper-proof records through blockchain immutability
- Higher trust among producers, suppliers, retailers, and consumers
- Strong protection against fraud and counterfeiting
- Real-time visibility of product movement
- Cryptographically secured data ensuring authenticity and integrity
- Full transparency without relying on centralized control

Overall, BSTS shows how blockchain can turn traditional supply chains into more transparent, secure, and reliable systems.

3. System Design and Architecture (1–2 pages): <u>9 points</u>

This blockchain-based supply chain tracking system uses a decentralized architecture to ensure transparent product tracking, secure ownership transfers, and tamper-proof data. The system connects Producer, Supplier, and Consumer through a Solidity smart contract deployed on the Ethereum Sepolia test network, with all interactions performed through a MetaMask-enabled web interface.

3.1 Workflow Overview

1. Producer Stage

- Producer registers a product by entering a Product ID and product details.
- The smart contract:
  - Computes a keccak256 hash of the data.
  - Stores the product ID, owner address, timestamp, and hash on-chain.
- Blockchain immutably records the product creation and initial ownership.

2. Supplier / Retailer Stage

- Supplier receives the product and transfers ownership by providing:
  - Product ID
  - Next owner's Ethereum address
- Contract validates that only the current owner can transfer.
- Ownership is updated and appended to the ownershipHistory, and an event is emitted.

3. Consumer Stage

- Consumer verifies authenticity by submitting Product ID and product details.
- Contract recomputes the hash and compares it with stored data.
- Returns true if the information matches, and provides access to full ownership history.

4. Smart Contract Role

- Acts as the single decentralized source of truth by providing:
  - Immutable registration
  - Secure ownership transfer
  - Cryptographic verification
  - Public ownership history
- No centralized database or server controls product information

3.2 Functional Modules

1. User Interaction Module

- Web dashboard for Producer, Supplier, and Consumer.
- Forms for product registration, ownership transfer, verification, and history lookup.
- Uses Ethers.js to communicate with the smart contract through MetaMask.

2. Product Registration Module

- Handles new product creation.
- Generates and stores keccak256 hash, owner address, timestamp, and product ID.
- Verifies uniqueness of the Product ID.

3. Ownership Transfer Module

- Checks current owner authorization and existence of product.
- Updates products[_productId].owner and appends to ownershipHistory.
- Emits ProductTransferred event.

4. Product Verification Module

- Recomputes the hash using submitted product data.
- Compares with stored hash and returns a boolean authenticity result.

5. History Tracking Module

- Stores all past owners associated with each product.
- Exposes a function to retrieve the supply chain path for consumer verification.

6. Blockchain Transaction Module

- Requires MetaMask signature for all write operations.
- Provides transaction hash, gas fee confirmation, and ensures secure, tamper-proof execution.

3.3 Technology Stack

Technology Stack

- Blockchain: Ethereum Sepolia Testnet + PowFaucet for test ETH.
- Smart Contract: Solidity (0.8.x) deployed via Remix IDE.
- Frontend: HTML, CSS, JavaScript, Ethers.js, MetaMask.
- Backend: PHP on XAMPP for routing and dashboards.
- Tools: VS Code for development, XAMPP for server hosting.

4. Implementation and Results (1–2 pages): 9 points

All related screenshots are provided in the Appendix

4.1 Implementation Process

a) Smart Contract Design

The system was implemented using a Solidity smart contract deployed on the **Ethereum Sepolia** network. The contract handles four core functions:

- **Product Registration:** Stores each product using a **keccak256 hash** to protect original data while enabling authenticity checks.
- **Ownership Transfer:** Ensures only the **current owner** can transfer a product, preventing unauthorized updates.
- **Product Verification:** When a consumer enters the product ID and data, the contract recomputes the hash and compares it to the stored one to detect tampering.
- **History Logging:** Every transfer is recorded permanently to provide full traceability.

Deployment was done using **Remix IDE**, and SepoliaETH from **PowFaucet** was used for transaction execution.

b) Web Interface Integration

The frontend was built using **PHP, HTML/CSS, JavaScript, and Ethers.js**, with three role-based dashboards:

- **Producer Dashboard:** Register products and view their stored hash.
- **Supplier Dashboard:** Receive products and transfer them to the next party.
- **Consumer Dashboard:** Verify authenticity and view ownership history.

Ethers.js connects the UI to the blockchain through functions like registerProduct(), transferProduct(), verifyProduct(), and getProductHistory().
All blockchain writes are securely signed using **MetaMask**, while PHP handles role access and session management.

4.2 Screenshots and Outputs

- **Product Registration:** Screenshot shows a confirmed blockchain transaction with product ID, hash, producer address, and timestamp. Figure 1 Figure 14

- **Ownership Transfer:** Screens illustrate transfers across entities, each generating an on-chain event and updating the owner. Figure7 Figure 13 Figure 14
- **Verification Result:** System outputs **true** for authentic products and **false** for tampered ones. Figure 9 Figure 10
- **Traceability:** Full ownership history is displayed as a list of blockchain addresses representing each handler. Figure 8 Figure 20

4.3 Transparency and Integrity in BSTS

Transparency

BSTS achieves transparency by storing all product actions—registration, transfer, verification—on the public Ethereum blockchain. Anyone can view:

- Stored product hash
- Current owner
- Complete ownership history

This creates an auditable, decentralized supply chain.

Integrity

Integrity is maintained through:

- **Immutable blockchain records:** No data can be altered or deleted.
- **Cryptographic hashing:** Any modification to product details is immediately detectable.
- **Strict ownership validation:** Only the legitimate owner can transfer the product.

End-to-End Trust Mechanism

1. Producer registers product → hash saved on-chain
2. Supplier transfers product → event logged
3. Consumer verifies → hash rechecked
4. History remains publicly traceable → prevents manipulation

This ensures secure, trusted, tamper-proof supply chain tracking from start to finish.

5. Discussion and Conclusion (1 page): <u>7 points</u>

5.1 Discussion

The Blockchain Supply Tracking System (BSTS) shows how blockchain and cryptography enhance trust and transparency in supply chains. The system's core security mechanism is the **keccak256 hash function**, which protects product information by storing only a unique hash on-chain rather than raw data. This ensures:

- **Data Integrity:** Any modification to product details produces a different hash, making tampering easy to detect.
- **Non-Repudiation:** Each registered product is permanently linked to the wallet address that created it.
- **Lightweight Storage:** Hashing reduces gas cost and protects sensitive information.

Ethereum's public ledger further strengthens transparency by recording all actions—registration, transfers, and verification—with timestamps. Using MetaMask for digital signatures ensures secure authentication without managing centralized credentials.

5.2 Potential Improvements

Although BSTS achieves functional traceability and verification, several enhancements could increase its scalability and usability:

1. **Encrypted Off-Chain Storage (IPFS):** Supports larger product metadata while reducing on-chain load.
2. **On-Chain Role-Based Access:** Enforces strict permissions for producers, suppliers, and consumers.
3. **Analytics Dashboard:** Automatically visualizes transfer events and product lifetime metrics.
4. **QR Code Scanning:** Simplifies data entry and reduces human error.
5. **Multi-Product Management:** Enables batch operations for industrial-scale use.
6. **Anomaly Detection:** Flags suspicious transfers or counterfeit patterns.

5.3 Conclusion

The BSTS prototype demonstrates how blockchain, hashing, and decentralized identities can create a transparent and tamper-proof supply-chain ecosystem. By combining a Solidity smart contract with a functional web interface, the system delivers reliable traceability, enhances consumer trust, and protects against fraud. Blockchain immutability ensures every event in the product lifecycle remains permanently recorded. With future enhancements, BSTS can scale into an industry-ready solution capable of supporting real-world logistics and authentication challenges.

6. References

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008.

[2] Ethereum Foundation, “Ethereum Documentation – Smart Contracts & Accounts,” Available: https://ethereum.org

[3] ConsenSys, “Solidity Smart Contract Best Practices,” Available: https://consensys.github.io/smart-contract-best-practices

[4] Remix Project, “Remix IDE Documentation,” Available: https://remix.ethereum.org

[5] MetaMask Docs, “Using MetaMask for Web3 Applications,” Available: https://docs.metamask.io

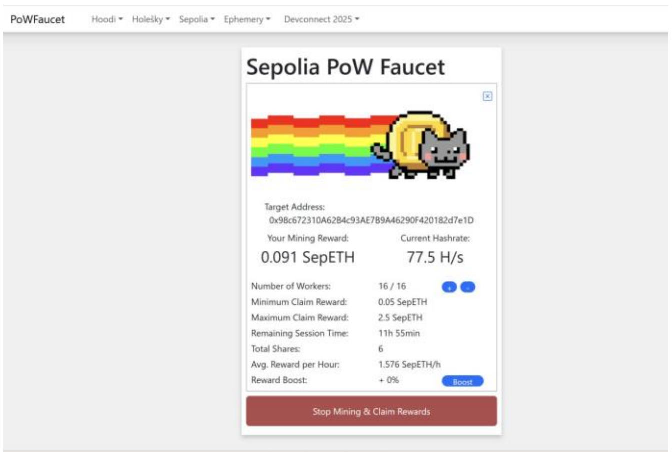
[6] Web3.js & Ethers.js Contributors, “Ethers.js Documentation,” Available: https://docs.ethers.org

[7] OWASP Foundation, “Blockchain Security Considerations,” 2023.

[8] [1] P. Kai, “Sepolia PoW Faucet,” *faucets.pk910.de*. Available: https://faucets.pk910.de/. Accessed: Dec. 9, 2025.

7. Appendix

Include additional contents



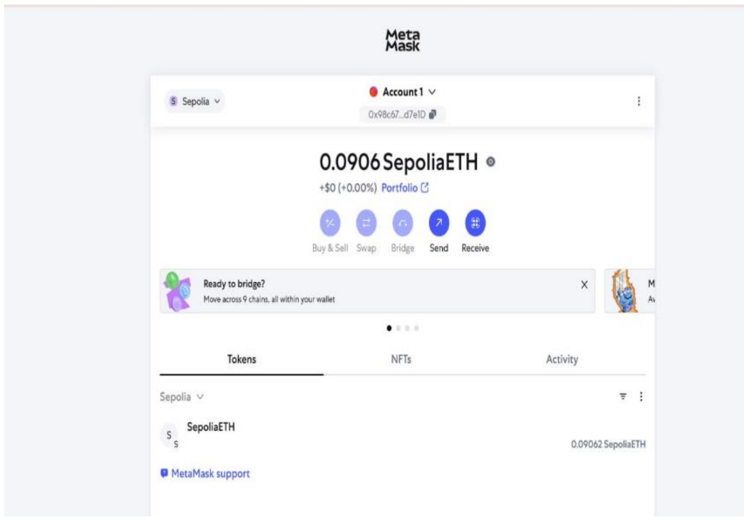
Sepolia PoW Faucet

Target Address:
0x98c672310A62B4c93AE7B9A46290F420182d7e1D

Your Mining Reward: 0.091 SepETH
Current Hashrate: 77.5 H/s

Number of Workers:	16 / 16
Minimum Claim Reward:	0.05 SepETH
Maximum Claim Reward:	2.5 SepETH
Remaining Session Time:	11h 55min
Total Shares:	6
Avg. Reward per Hour:	1.576 SepETH/h
Reward Boost:	+ 0%

Figure 1: Sepolia PoW Faucet Used to Obtain Test ETH for Smart Contract Transactions



0x98c67...d7e1D

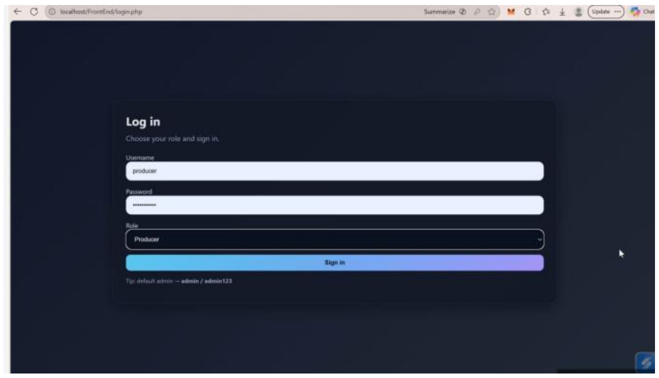
0.0906 SepoliaETH

+$0 (+0.00%) Portfolio

Ready to bridge?
Move across 9 chains, all within your wallet

SepoliaETH 0.09062 SepoliaETH

Figure 2: MetaMask Wallet Showing SepoliaETH Balance Used for Blockchain Transactions



Log in

Choose your role and sign in.

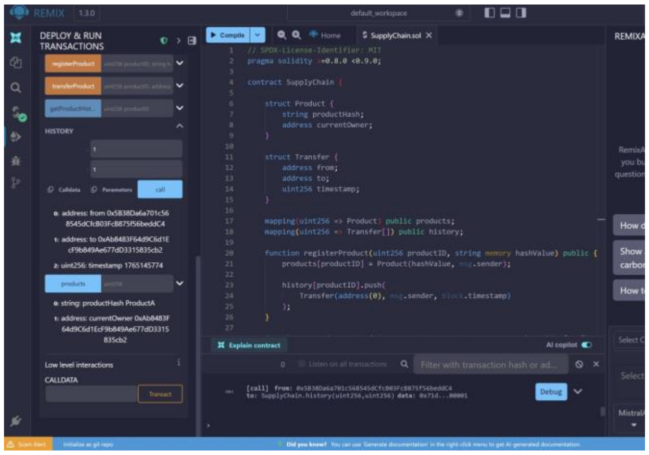
Username: producer

Password: •••••••••••

Role: Producer

Tip: default admin → **admin / admin123**

Figure 3 – User Login Interface with Role-Based Authentication



HISTORY: 1, 1

0: address: from 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4

1: address: to 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

2: uint256: timestamp 1765145774

products

0: string: productHash ProductA

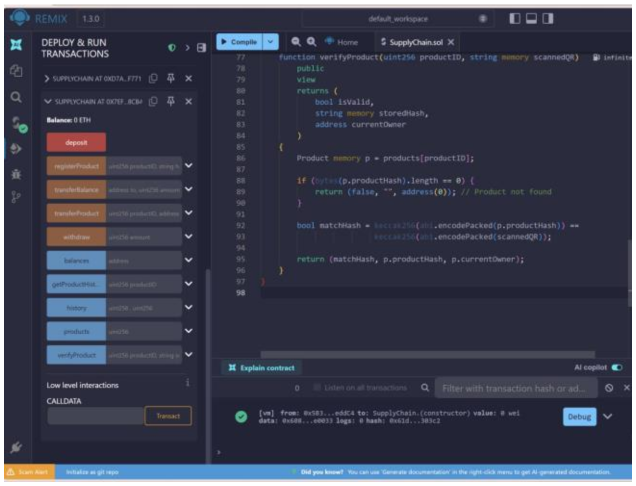
1: address: currentOwner 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.8.0 <0.9.0;

contract SupplyChain {

    struct Product {
        string productHash;
        address currentOwner;
    }

    struct Transfer {
        address from;
        address to;
        uint256 timestamp;
    }

    mapping(uint256 => Product) public products;
    mapping(uint256 => Transfer[]) public history;

    function registerProduct(uint256 productID, string memory hashValue) public {
        products[productID] = Product(hashValue, msg.sender);

        history[productID].push(
            Transfer(address(0), msg.sender, block.timestamp)
        );
    }
```

```
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to: SupplyChain.history(uint256,uint256) data: 0x71d...00001
```

Did you know? You can use 'Generate documentation' in the right-click menu to get AI-generated documentation.

Figure 4: Smart Contract Execution in Remix Showing Successful Product Registration



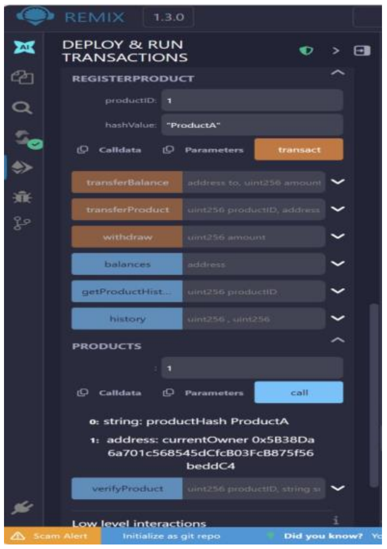
Balance: 0 ETH

```
    function verifyProduct(uint256 productID, string memory scannedQR)
        public
        view
        returns (
            bool isValid,
            string memory storedHash,
            address currentOwner
        )
    {
        Product memory p = products[productID];

        if (bytes(p.productHash).length == 0) {
            return (false, "", address(0)); // Product not found
        }

        bool matchHash = keccak256(abi.encodePacked(p.productHash)) ==
                         keccak256(abi.encodePacked(scannedQR));

        return (matchHash, p.productHash, p.currentOwner);
    }
}
```

```
[vm] from: 0x5B3...eddC4 to: SupplyChain.(constructor) value: 0 wei
data: 0x608...e0033 logs: 0 hash: 0x61d...303c2
```

Did you know? You can use 'Generate documentation' in the right-click menu to get AI-generated documentation.

Figure 5: Remix Screenshot Showing verifyProduct() Execution and Hash-Based Product Verification

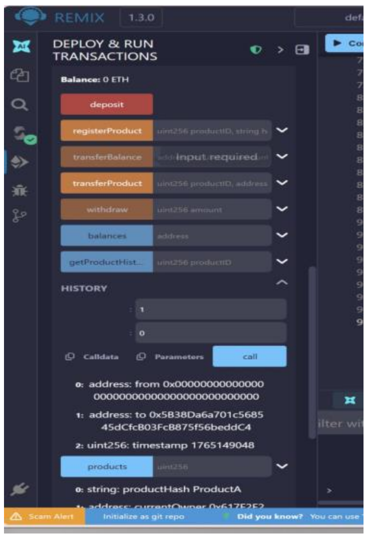


```
productID: 1
hashValue: "ProductA"
```

```
: 1
```

```
0: string: productHash ProductA
1: address: currentOwner 0x5B38Da
   6a701c568545dCfcB03FcB875f56
   beddC4
```

Figure 6: registerProduct() Execution and On-Chain Product Data Displayed in Remix



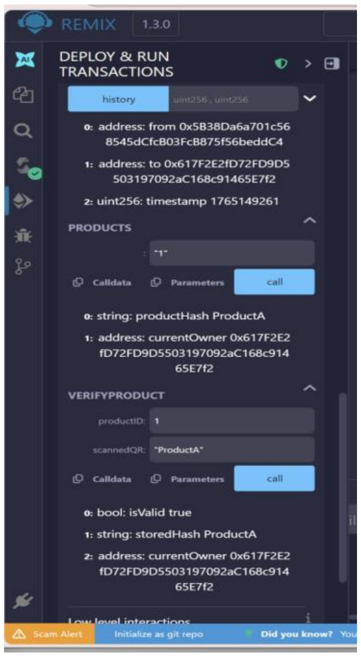
```
Balance: 0 ETH
```

```
: 1
: 0
```

```
0: address: from 0x000000000000000
   0000000000000000000000000
1: address: to 0x5B38Da6a701c5685
   45dCfcB03FcB875f56beddC4
2: uint256: timestamp 1765149048
```

```
0: string: productHash ProductA
```

Figure7: On-Chain Ownership History Retrieved Using getProductHistory() in Remix



```
0: address: from 0x5B38Da6a701c56
   8545dCfcB03FcB875f56beddC4
1: address: to 0x617F2E2fD72FD9D5
   503197092aC168c91465E7f2
2: uint256: timestamp 1765149261
```

PRODUCTS

"1"

```
0: string: productHash ProductA
1: address: currentOwner 0x617F2E2
   fD72FD9D5503197092aC168c914
   65E7f2
```

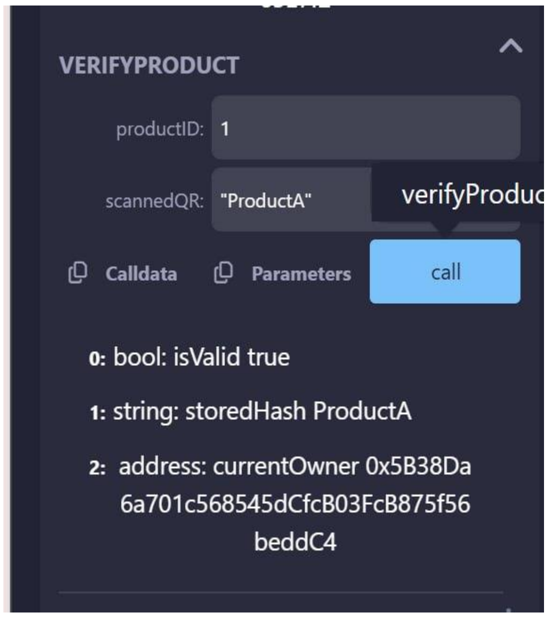
VERIFYPRODUCT

productID: 1

scannedQR: "ProductA"

```
0: bool: isValid true
1: string: storedHash ProductA
2: address: currentOwner 0x617F2E2
   fD72FD9D5503197092aC168c914
   65E7f2
```

Figure 8: On-Chain Product History and Verification Output Displayed in Remix

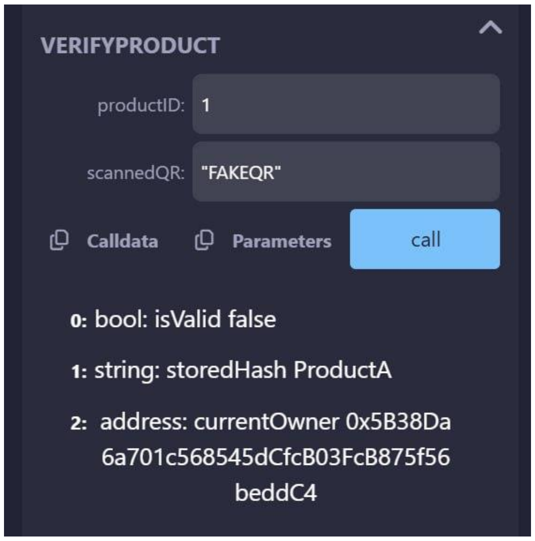

VERIFYPRODUCT

productID: 1

scannedQR: "ProductA"

```
0: bool: isValid true
1: string: storedHash ProductA
2: address: currentOwner 0x5B38Da
   6a701c568545dCfcB03FcB875f56
   beddC4
```

Figure 9: verifyProduct() Output Showing Successful Product Authentication in Remix



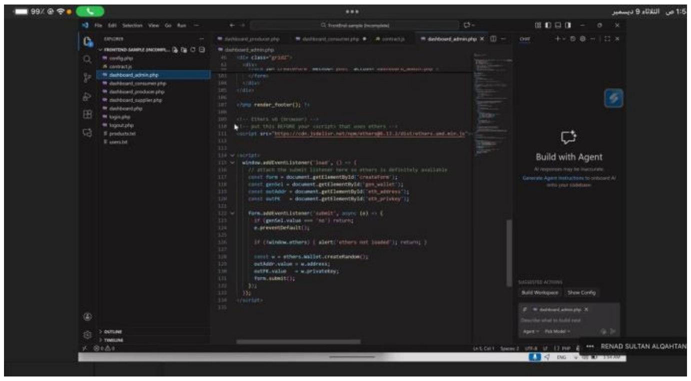
VERIFYPRODUCT

productID: 1

scannedQR: "FAKEQR"

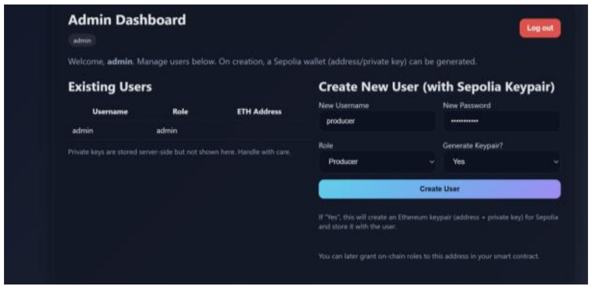
```
0: bool: isValid false
1: string: storedHash ProductA
2: address: currentOwner 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
```

Figure 10: Failed Product Verification Showing isValid = false for Incorrect QR/Data



```
<div class="grid2">
  <div>
    </form>
  </div>
</div>

<?php render_footer(); ?>

<!-- Ethers v6 (browser) -->
<!-- put this BEFORE your script that uses ethers -->
<script src="https://cdn.jsdelivr.net/npm/ethers@6.13.2/dist/ethers.umd.min.js"></script>


<script>
  window.addEventListener('load', () => {
    // attach the submit listener here so ethers is definitely available
    const form = document.getElementById('createForm');
    const genSel = document.getElementById('gen_wallet');
    const outAddr = document.getElementById('eth_address');
    const outPK   = document.getElementById('eth_privkey');

    form.addEventListener('submit', async (e) => {
      if (genSel.value === 'no') return;
      e.preventDefault();

      if (!window.ethers) { alert('ethers not loaded'); return; }

      const w = ethers.Wallet.createRandom();
      outAddr.value = w.address;
      outPK.value   = w.privateKey;
      form.submit();
    });
  });
</script>
```

Figure 11: Frontend Development Session in VS Code Showing Integration with Ethers.js for Wallet and User Management



Admin Dashboard

admin

Welcome, **admin**. Manage users below. On creation, a Sepolia wallet (address/private key) can be generated.

Existing Users

Username	Role	ETH Address
admin	admin	

Private keys are stored server-side but not shown here. Handle with care.

Create New User (with Sepolia Keypair)

New Username: producer

New Password: ••••••••••••

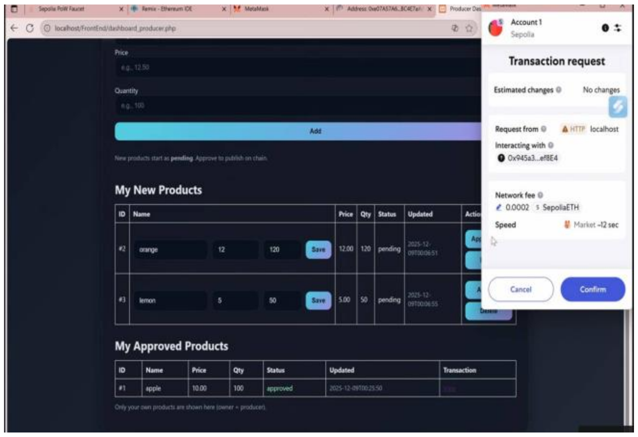
Role: Producer

Generate Keypair?: Yes

If "Yes", this will create an Ethereum keypair (address + private key) for Sepolia and store it with the user.

You can later grant on-chain roles to this address in your smart contract.

Figure 12: Admin Dashboard Interface for Managing Users and Generating Sepolia Keypairs



New products start as **pending**. Approve to publish on chain.

My New Products

ID	Name	Price	Qty	Status	Updated	Actio
#2	orange 12 120 Save	12.00	120	pending	2025-12-09T00:06:51	
#3	lemon 5 50 Save	5.00	50	pending	2025-12-09T00:06:55	

My Approved Products

ID	Name	Price	Qty	Status	Updated	Transaction
#1	apple	10.00	100	approved	2025-12-09T00:25:50	[illegible]

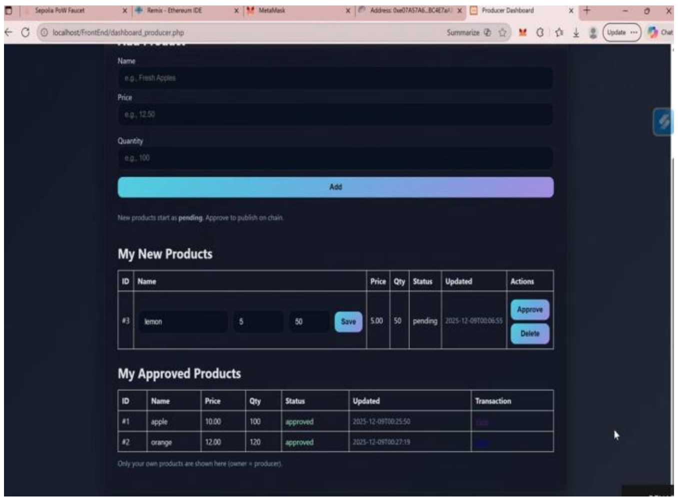
Only your own products are shown here (owner = producer).

Account 1
Sepolia

Transaction request

Estimated changes — No changes
Request from — HTTP localhost
Interacting with — 0x945a3...ef8E4
Network fee — 0.0002 SepoliaETH
Speed — Market ~12 sec

Figure 13: Producer Dashboard Triggering an On-Chain Transaction Confirmed Through MetaMask



New products start as **pending**. Approve to publish on chain.

My New Products

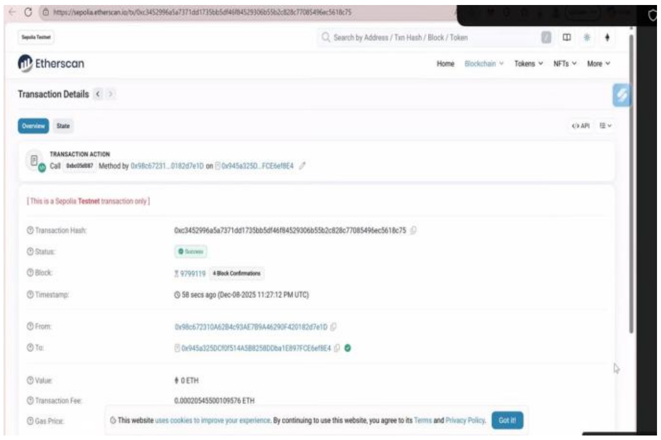
ID	Name	Price	Qty	Status	Updated	Actions
#3	lemon 5 50 Save	5.00	50	pending	2025-12-09T00:06:55	Approve Delete

My Approved Products

ID	Name	Price	Qty	Status	Updated	Transaction
#1	apple	10.00	100	approved	2025-12-09T00:25:50	[illegible]
#2	orange	12.00	120	approved	2025-12-09T00:27:19	[illegible]

Only your own products are shown here (owner = producer).

Figure 14: Producer Dashboard Displaying Pending and Approved Products with On-Chain Transaction Links



https://sepolia.etherscan.io/tx/0xc3452996a5a7371dd1735bb5df46f84529306b55b2c828c77085496ec5618c75

Sepolia Testnet

Etherscan

Transaction Details

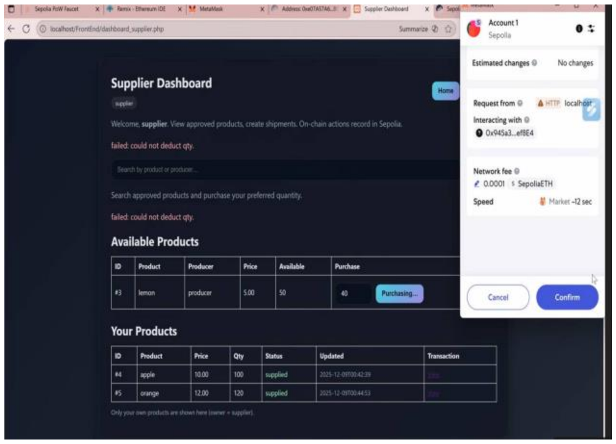
TRANSACTION ACTION
Call 0xbc05d087 Method by 0x98c67231...0182d7e1D on 0x945a325D...FCE6ef8E4

[This is a Sepolia Testnet transaction only]

Transaction Hash:	0xc3452996a5a7371dd1735bb5df46f84529306b55b2c828c77085496ec5618c75
Status:	Success
Block:	9799119 4 Block Confirmations
Timestamp:	58 secs ago (Dec-08-2025 11:27:12 PM UTC)
From:	0x98c672310A62B4c93AE7B9A46290F420182d7e1D
To:	0x945a325DCf0f514A5B8258DDba1E897FCE6ef8E4
Value:	0 ETH
Transaction Fee:	0.00020545500109576 ETH
Gas Price:	

This website uses cookies to improve your experience. By continuing to use this website, you agree to its Terms and Privacy Policy.

Figure 15: Etherscan Record Showing Successful On-Chain Transaction Executed by the BSTS System



localhost/FrontEnd/dashboard_supplier.php

Supplier Dashboard

supplier

Welcome, **supplier**. View approved products, create shipments. On-chain actions record in Sepolia.

failed: could not deduct qty.

Search approved products and purchase your preferred quantity.

failed: could not deduct qty.

Available Products

ID	Product	Producer	Price	Available	Purchase
#3	lemon	producer	5.00	50	40 Purchasing...

Your Products

ID	Product	Price	Qty	Status	Updated	Transaction
#4	apple	10.00	100	supplied	2025-12-09T00:42:39	[illegible]
#5	orange	12.00	120	supplied	2025-12-09T00:44:53	[illegible]

Only your own products are shown here (owner = supplier).

Account 1
Sepolia

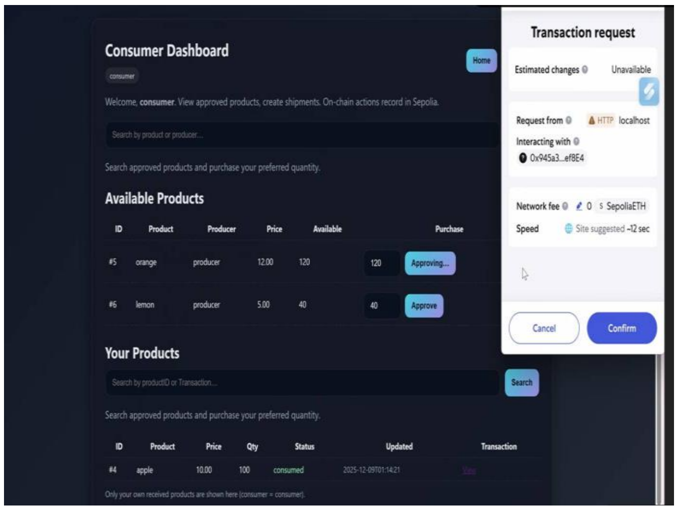
Estimated changes: No changes

Request from: HTTP localhost
Interacting with: 0x945a3...ef8E4

Network fee: 0.0001 SepoliaETH
Speed: Market ~12 sec

Cancel Confirm

Figure 16: Supplier Dashboard Purchase Action Triggering On-Chain MetaMask Transaction


Consumer Dashboard

Welcome, **consumer**. View approved products, create shipments. On-chain actions record in Sepolia.

Search approved products and purchase your preferred quantity.

Available Products

ID	Product	Producer	Price	Available	Purchase
#5	orange	producer	12.00	120	120 Approving...
#6	lemon	producer	5.00	40	40 Approve

Your Products

Search approved products and purchase your preferred quantity.

ID	Product	Price	Qty	Status	Updated	Transaction
#4	apple	10.00	100	consumed	2025-12-09T01:14:21	

Only your own received products are shown here (consumer = consumer).

Transaction request

Estimated changes: Unavailable

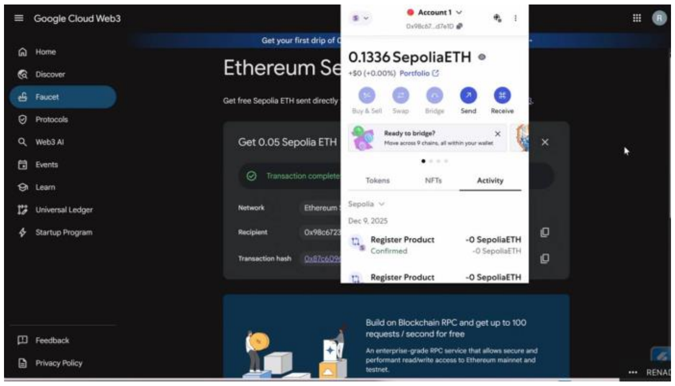
Request from: HTTP localhost

Interacting with: 0x945a3...ef8E4

Network fee: 0 SepoliaETH

Speed: Site suggested ~12 sec

Figure 17: Consumer Dashboard Purchase Flow Triggering an On-Chain MetaMask Transaction


Google Cloud Web3 — Ethereum Se… Faucet

Get free Sepolia ETH sent directly…

Get 0.05 Sepolia ETH — Transaction complete

Network: Ethereum …

Recipient: 0x98c672…

Transaction hash: 0x87c609…

Account 1 — 0x98c67...d7e1D

0.1336 SepoliaETH

+$0 (+0.00%) Portfolio

Activity — Sepolia — Dec 9, 2025

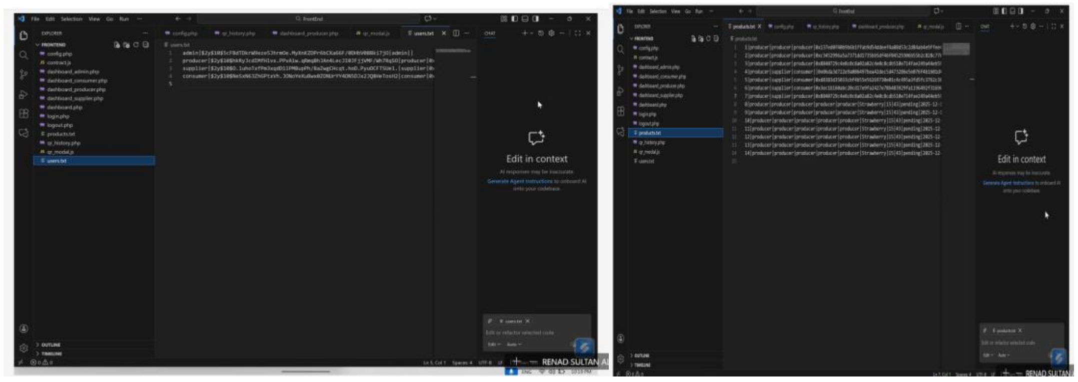
Register Product — Confirmed — -0 SepoliaETH

Register Product — -0 SepoliaETH

Build on Blockchain RPC and get up to 100 requests / second for free

An enterprise-grade RPC service that allows secure and performant read/write access to Ethereum mainnet and testnet.

Figure 18: Google Cloud Web3 Faucet Funding and MetaMask Activity Showing Successful SepoliaETH Transactions

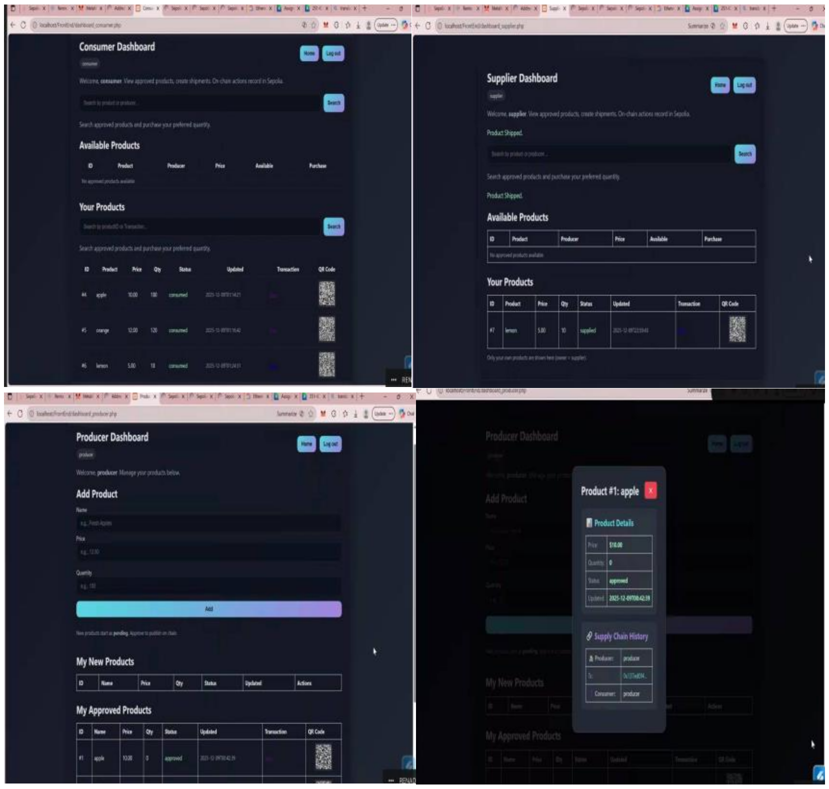

users.txt

```
admin|$2y$10$[illegible]|admin||
producer|$2y$10$[illegible]|producer|0x
supplier|$2y$10$[illegible]|supplier|0x
consumer|$2y$10$[illegible]|consumer|0x
```

products.txt

```
1|producer|producer|producer|0x[illegible]
2|producer|producer|producer|0x[illegible]
3|producer|producer|producer|0x[illegible]
4|producer|supplier|consumer|0x[illegible]
5|producer|supplier|consumer|0x[illegible]
6|producer|supplier|consumer|0x[illegible]
7|producer|supplier|producer|0x[illegible]
8|producer|producer|producer|producer|producer|Strawberry|15|43|pending|2025-12-
9|producer|producer|producer|producer|producer|Strawberry|15|43|pending|2025-12-
10|producer|producer|producer|producer|producer|Strawberry|15|43|pending|2025-12-
11|producer|producer|producer|producer|producer|Strawberry|15|43|pending|2025-12-
12|producer|producer|producer|producer|producer|Strawberry|15|43|pending|2025-12-
13|producer|producer|producer|producer|producer|Strawberry|15|43|pending|2025-12-
14|producer|producer|producer|producer|producer|Strawberry|15|43|pending|2025-12-
```

Figure 19 – User Credentials Stored in Hashed Format



Consumer Dashboard

consumer

Welcome, **consumer**. View approved products, create shipments. On-chain actions record in Sepolia.

Search approved products and purchase your preferred quantity.

Available Products

ID	Product	Producer	Price	Available	Purchase
No approved products available					

Your Products

Search approved products and purchase your preferred quantity.

ID	Product	Price	Qty	Status	Updated	Transaction	QR Code
#4	apple	10.00	100	consumed	2025-12-09T01:14:21		
#5	orange	12.00	120	consumed	2025-12-09T01:16:42		
#6	lemon	5.00	10	consumed	2025-12-09T01:24:31		

Supplier Dashboard

supplier

Welcome, **supplier**. View approved products, create shipments. On-chain actions record in Sepolia.

Product Shipped.

Search approved products and purchase your preferred quantity.

Product Shipped.

Available Products

ID	Product	Producer	Price	Available	Purchase
No approved products available					

Your Products

ID	Product	Price	Qty	Status	Updated	Transaction	QR Code
#7	lemon	5.00	10	supplied	2025-12-09T22:59:41		

Only your own products are shown here (owner = supplier).

Producer Dashboard

producer

Welcome, **producer**. Manage your products below.

Add Product

Name: e.g., Fresh Apples

Price: e.g., 12.00

Quantity: e.g., 100

Add

New products start as **pending**. Approve to publish on chain.

My New Products

ID	Name	Price	Qty	Status	Updated	Actions

My Approved Products

ID	Name	Price	Qty	Status	Updated	Transaction	QR Code
#1	apple	10.00	0	approved	2025-12-09T00:42:39		

Product #1: apple

Product Details

Price:	$10.00
Quantity:	0
Status:	approved
Updated:	2025-12-09T00:42:39

Supply Chain History

Producer:	producer
Tx:	0x137e8094...
Consumer:	producer

Figures 20 – QR Codes Across All User Roles

The QR code feature is available across all system roles — **Producer, Supplier, and Consumer** — enabling each user to scan, verify, and track product information at different stages of the supply chain. This unified QR mechanism enhances transparency and ensures end-to-end traceability throughout the BSTS workflow.

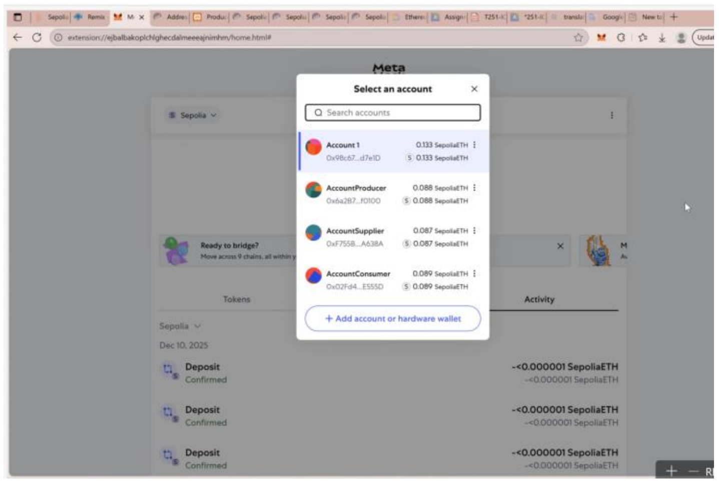

Meta
Select an account
Search accounts
Account 1 0.133 SepoliaETH
0x98c67...d7e1D 0.133 SepoliaETH
AccountProducer 0.088 SepoliaETH
0x6a2B7...f0100 0.088 SepoliaETH
AccountSupplier 0.087 SepoliaETH
0xF755B...A638A 0.087 SepoliaETH
AccountConsumer 0.089 SepoliaETH
0x02Fd4...E555D 0.089 SepoliaETH
+ Add account or hardware wallet
Sepolia
Ready to bridge?
Tokens
Activity
Dec 10, 2025
Deposit Confirmed -<0.000001 SepoliaETH
Deposit Confirmed -<0.000001 SepoliaETH
Deposit Confirmed -<0.000001 SepoliaETH

Figure 21: MetaMask accounts for producer, supplier, and consumer—each pre-funded with SepoliaETH to support blockchain transactions

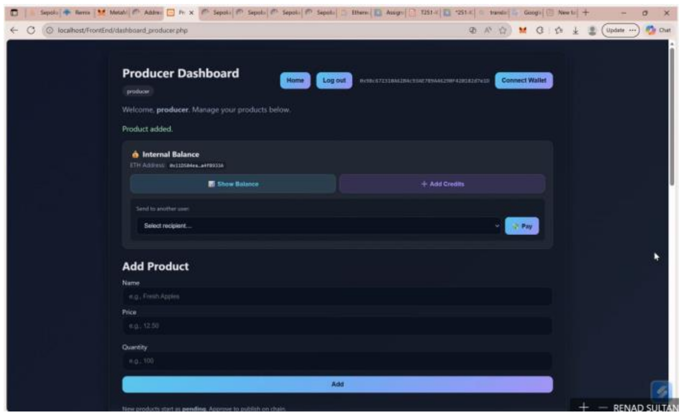


Figure 22 – Producer Dashboard Showing Internal Wallet Balance and Product Registration Interface

Formatting Guidelines

- Length: 4–6 pages (excluding title, references, appendix)
- Font: Times New Roman, 12 pt
- Line spacing: 1.15
- Margins: 1 inch on all sides
- File type: PDF
- Figures: Must be numbered and captioned

Tips for Students

- Keep explanations concise and focused on your implementation.
- Include diagrams and clear screenshots to highlight and illustrate key points, if necessary.
- Use bullet points for clarity, if applicable.
- Write in your own words — avoid copy-pasting from sources
- **Attach Smart Contract file and Web Implementation code in a separate folder**