

| **Title: Chapter No:05 Prototype Implementation for the Mini-Project.** |
| --- |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected Outcome of Experiment:**

**CO3: Implement and prototype creation for the specified application.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

*[Students can mention websites/ books used in their project implementation]*

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**This write up will expect students to prepare chapter no 5 in the format given below**

**Chapter 5**

**Implementation and Development of the Prototype**

**Introduction:**

[System Implementation](https://www.sebokwiki.org/wiki/Implementation_(glossary)) uses the structure created during [architectural design](https://www.sebokwiki.org/wiki/Architecting_(glossary)) and the results of [system analysis](https://www.sebokwiki.org/wiki/System_Analysis) to construct [system elements](https://www.sebokwiki.org/wiki/System_Element_(glossary)) that meet the [stakeholder requirements](https://www.sebokwiki.org/wiki/Stakeholder_Requirement_(glossary)) and [system requirements](https://www.sebokwiki.org/wiki/System_Requirement_(glossary)) developed in the early [life cycle](https://www.sebokwiki.org/wiki/Life_Cycle_(glossary)) phases. These system elements are then integrated to form intermediate [aggregates](https://www.sebokwiki.org/wiki/Aggregate_(glossary)) and finally the complete [system](https://www.sebokwiki.org/wiki/System-of-Interest_(glossary))

The implementation and prototyping document should be presented with a description of following steps.

1. **Modules Description:**

Write input-output, properties, scenarios of important modules in the code in the given format.

| **Module** | **Name, Definition, purpose** |
| --- | --- |
| *Name* | *Definition:*  *Purpose:[ Identifier, name, description, type (hardware, software application, software piece, mechanical part, electric art, electronic component, operator role, procedure, protocol, manual, etc.) ]* |

*Activity: Here, team members will mention important modules coding details and input-output parameters of respective modules.*

1. **Integration:**

Integration is a critical step within software implementation and it involves migrating data, compiling all modules in one system. With proper integrations, one can expect the project to be producing expected output.

*Activity: Here, team members are required to document their integration strategy, mention about versions of module code if any, dependencies if any.*

1. **Dataset link (if applicable) with source/ Process of dataset selection with sample data along with data dictionary.**

**NA**

1. **Implementation details**

### 4.1 Frontend Implementation

The frontend is developed using React with TypeScript and enhanced using Vite for a faster and leaner development experience. It offers a responsive and intuitive user interface for all stakeholders — manufacturers, distributors, retailers, and consumers.

* RainbowKit is used for wallet connection and authentication.
* Wagmi is used to interact with Ethereum smart contracts directly from the frontend.
* Users can apply for roles, register drugs, update drug status, and verify product authenticity.

### 4.2 Backend Integration

The backend is built using Node.js and Apollo GraphQL Server. It handles off-chain logic such as:

* Validating user roles before allowing access.
* Storing metadata (like logs and analytics) in MongoDB.
* Serving GraphQL APIs to the frontend for role verification and transaction tracking.

The backend also communicates with smart contracts when users apply for roles or trigger drug-related transactions.

### 4.3 Blockchain Integration

The blockchain component is built on the Ethereum Sepolia Testnet using Solidity smart contracts. Smart contracts manage:

* Role assignment (manufacturer, distributor, etc.)
* Drug registration
* Status updates during drug transfer
* Verification requests

Each transaction is immutable and recorded on-chain, ensuring full traceability and transparency.

Smart contract functions were deployed using Remix IDE, and transactions are triggered from the frontend using Wagmi’s writeContract() method.

### 4.4 Data Flow and Transaction Lifecycle

1. User connects wallet → Wallet authentication is handled via RainbowKit.
2. User applies for a role → A GraphQL mutation is sent to the backend and passed to the blockchain to execute a smart contract that assigns the role.
3. Authorized users register or update drug data → Smart contracts are called with relevant parameters (drug ID, location, status).
4. Consumers verify authenticity → A readContract() call fetches immutable history from the blockchain.

### 4.5 Storage and Decentralization

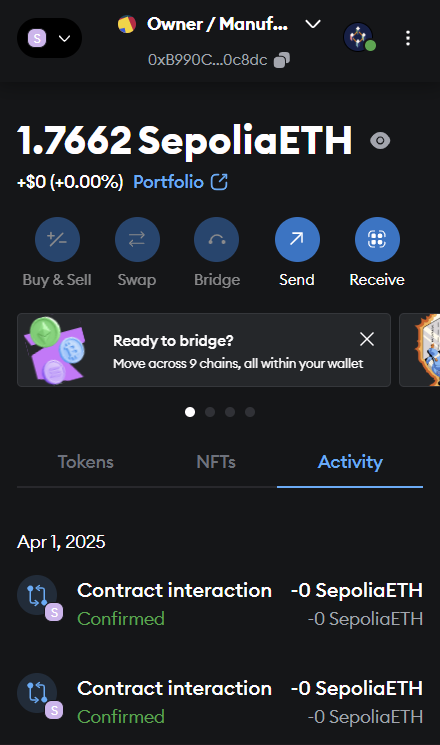
* All critical drug tracking data is stored on-chain using Ethereum.
* Non-critical metadata and logs are stored in MongoDB.
* The system uses IPFS (optional) for storing large metadata like certificates or labels.

### 4.6 Security Measures

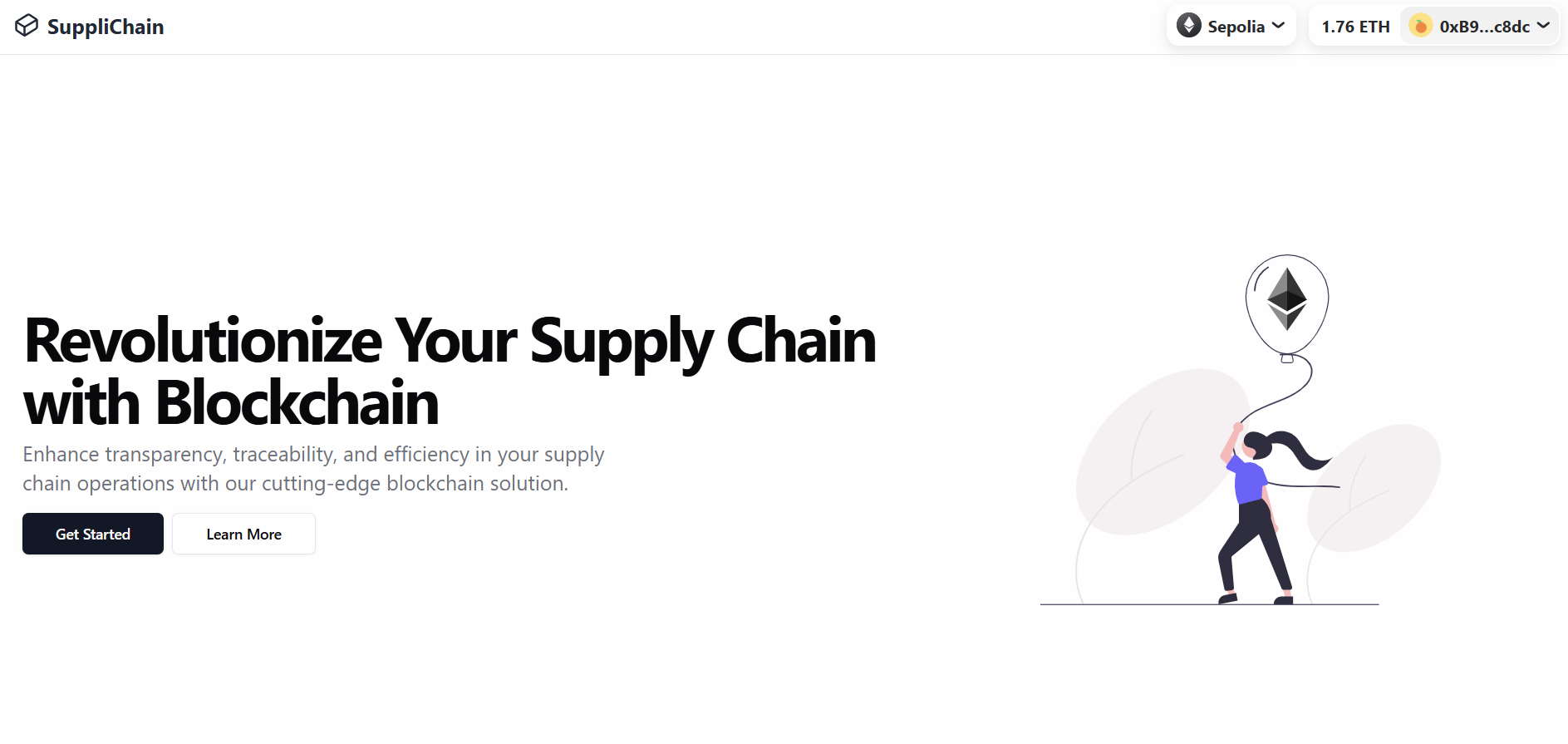
* Role-based access control ensures only authorized users can perform specific actions.
* Transactions are signed with the user's wallet, preventing spoofing.
* Immutable ledger guarantees tamper-proof history for each drug batch.

1. **Implementation Screenshots: Each team will present implemented module screenshots in accordance of process flow with small description**
2. **Wallet Connection**

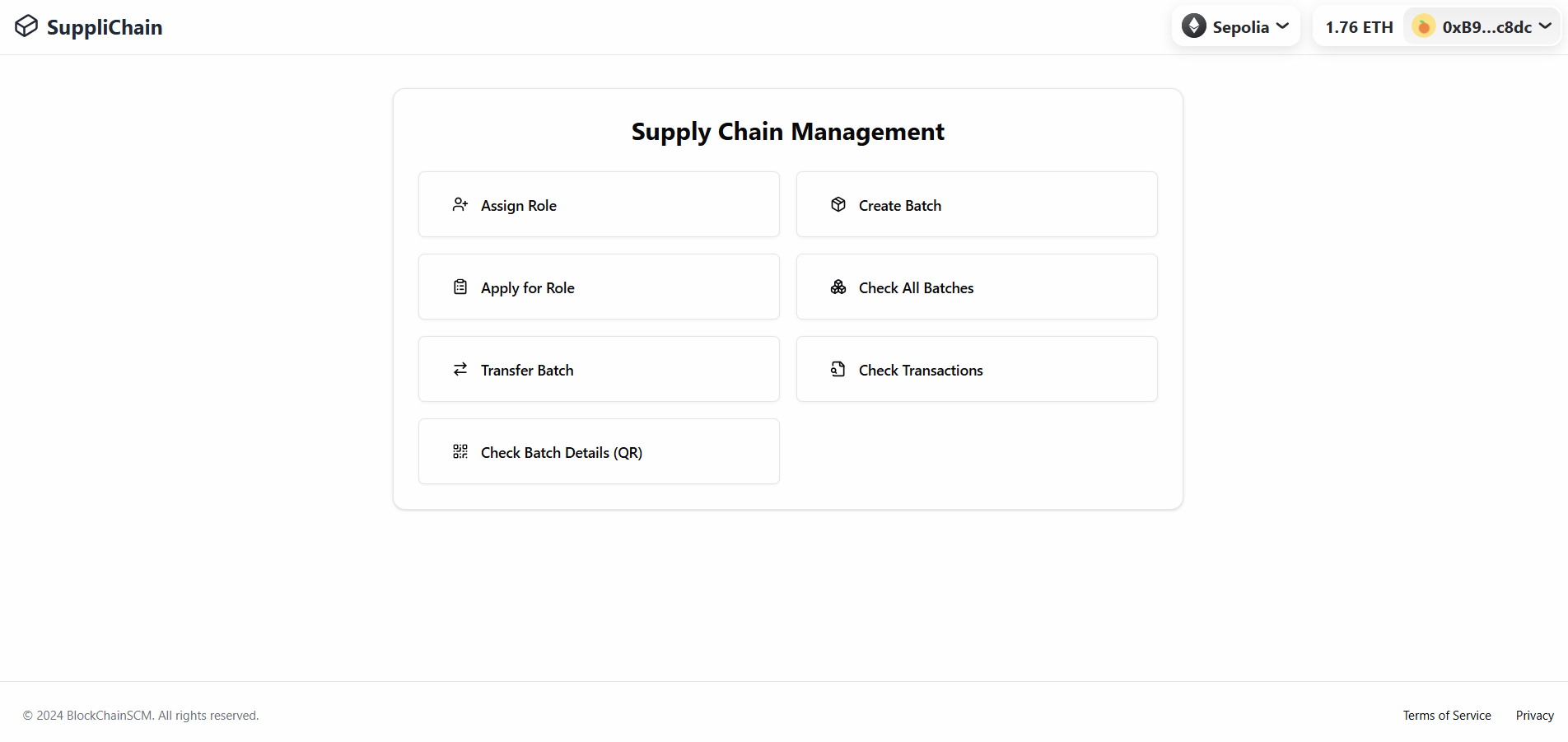
Users can connect their crypto wallets (e.g., MetaMask, RainbowKit) to access the application. Wallet authentication is required before accessing any role-based functionality.



1. **Home Page**

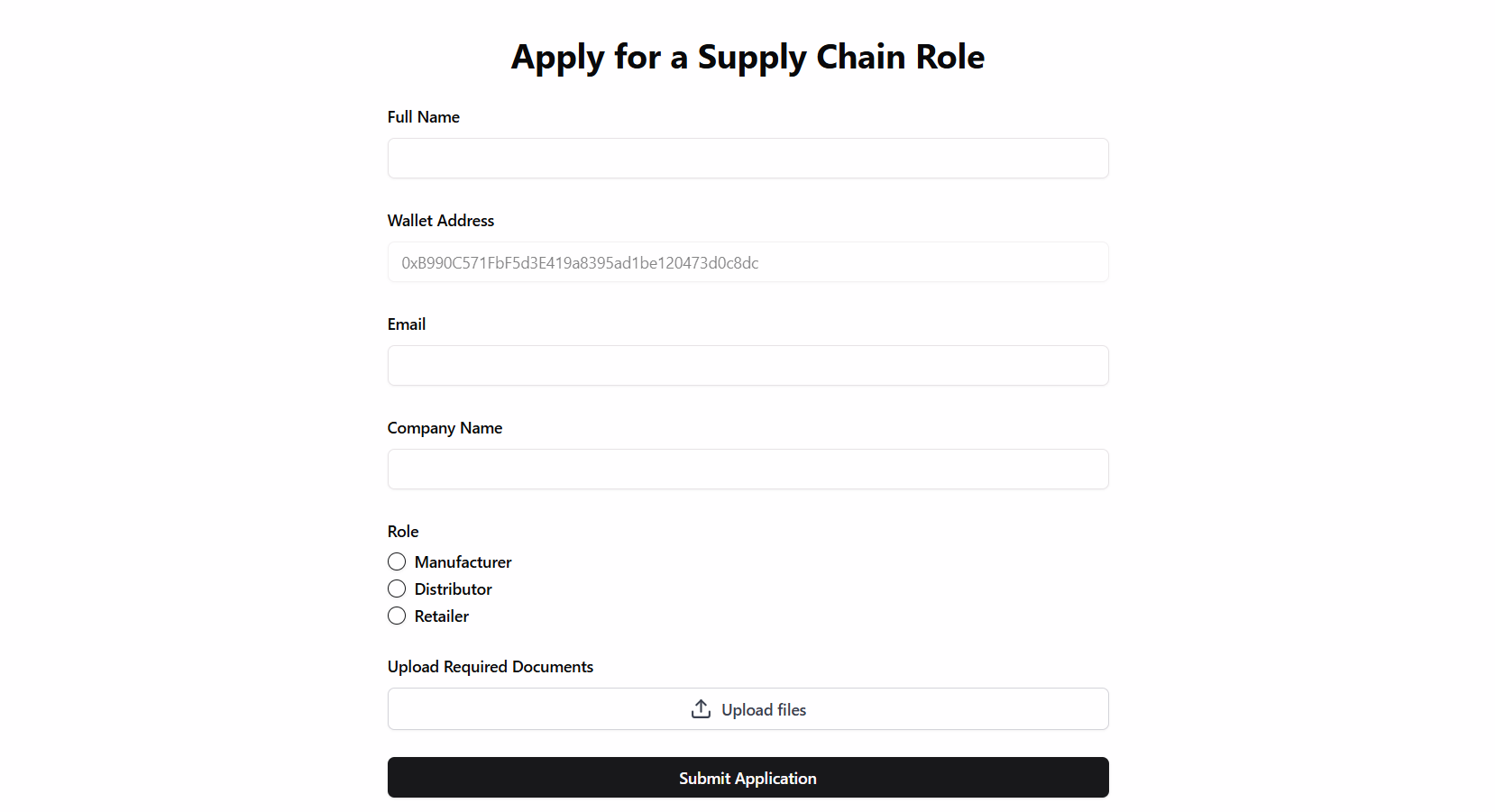
****

1. **Dashboard**

****

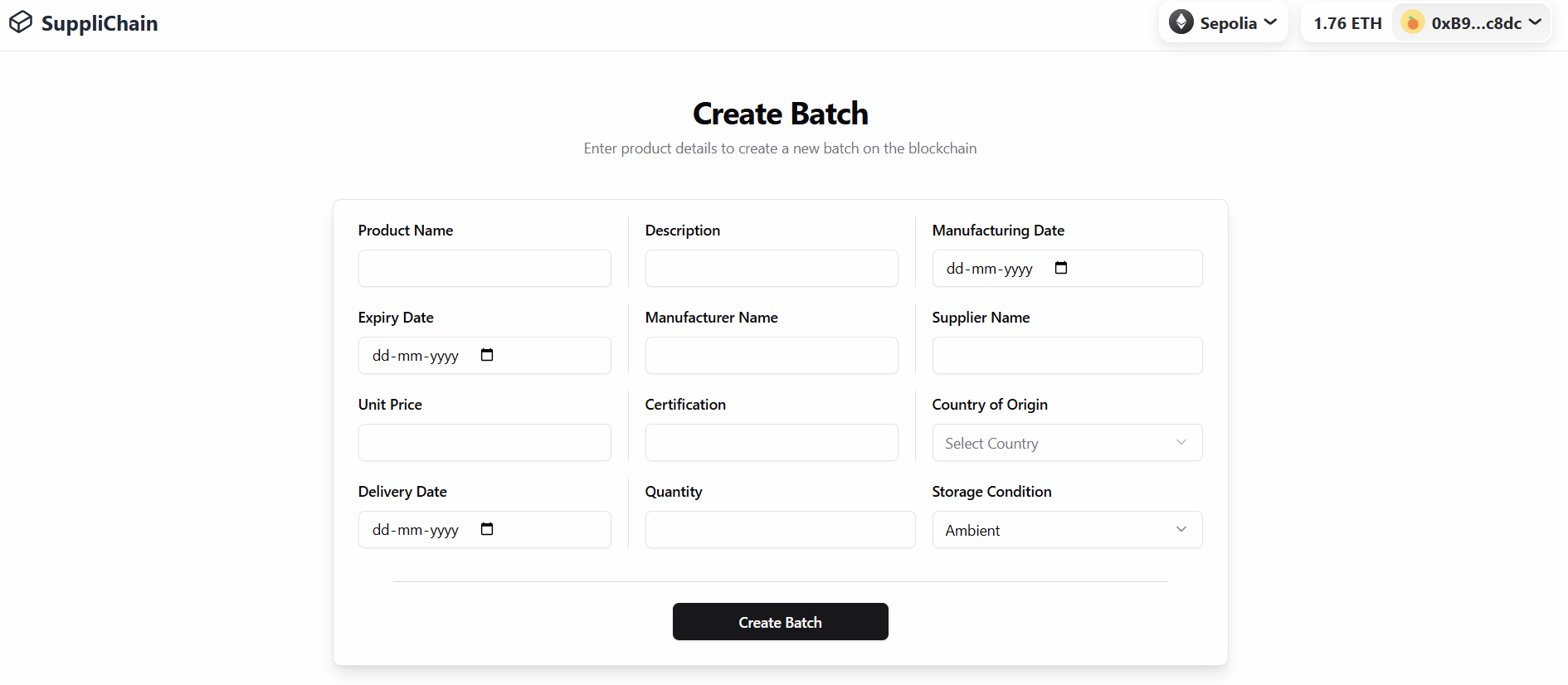
1. **Apply for Role**

After wallet connection, users can apply for a specific role (Manufacturer, Distributor, Retailer, or Consumer). The request is sent to the backend and logged into the blockchain after verification.



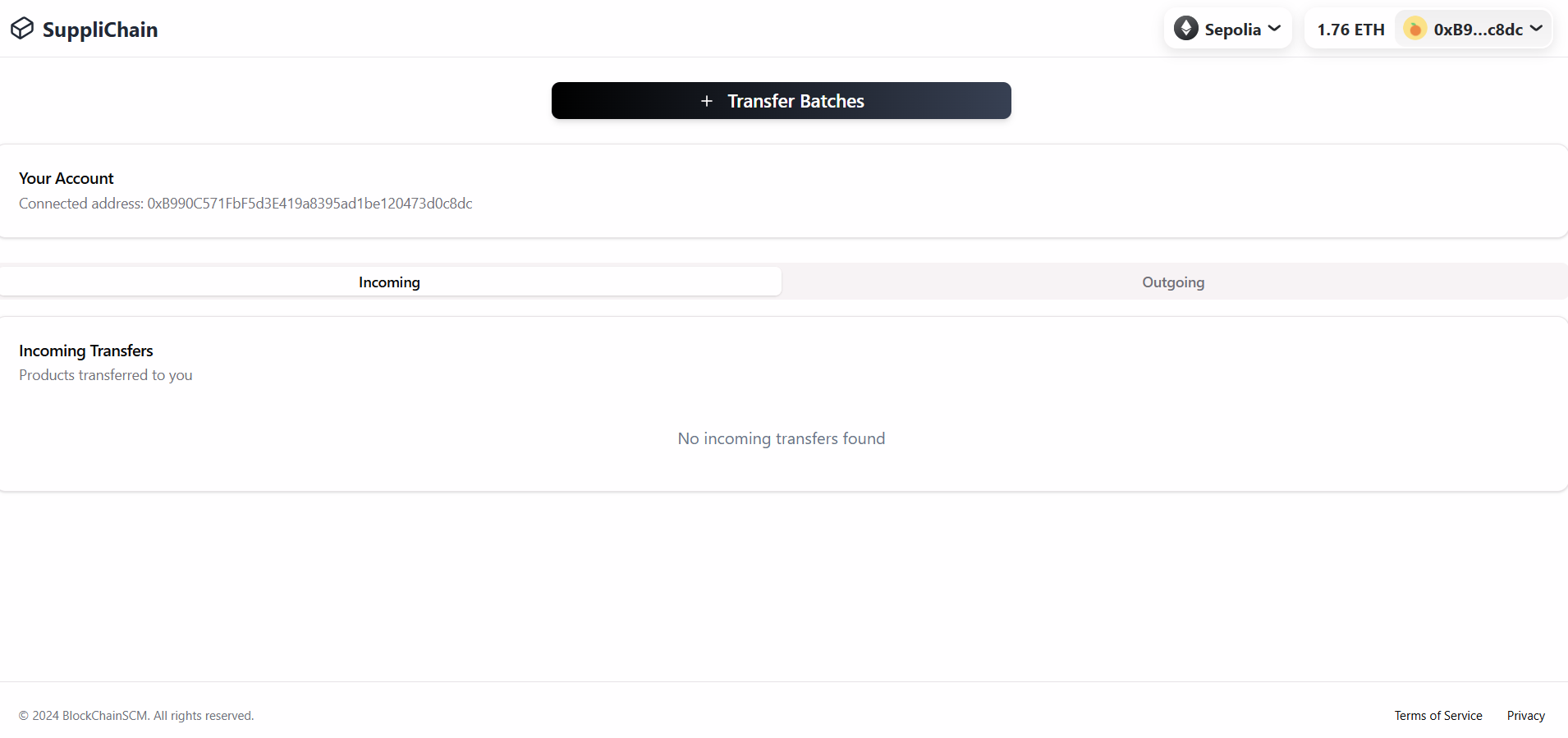
1. **Batch Registration Form**

Manufacturers can input drug details including name, batch ID, manufacture date, and expiry. Once submitted, a smart contract is triggered to record the drug information immutably on the blockchain.

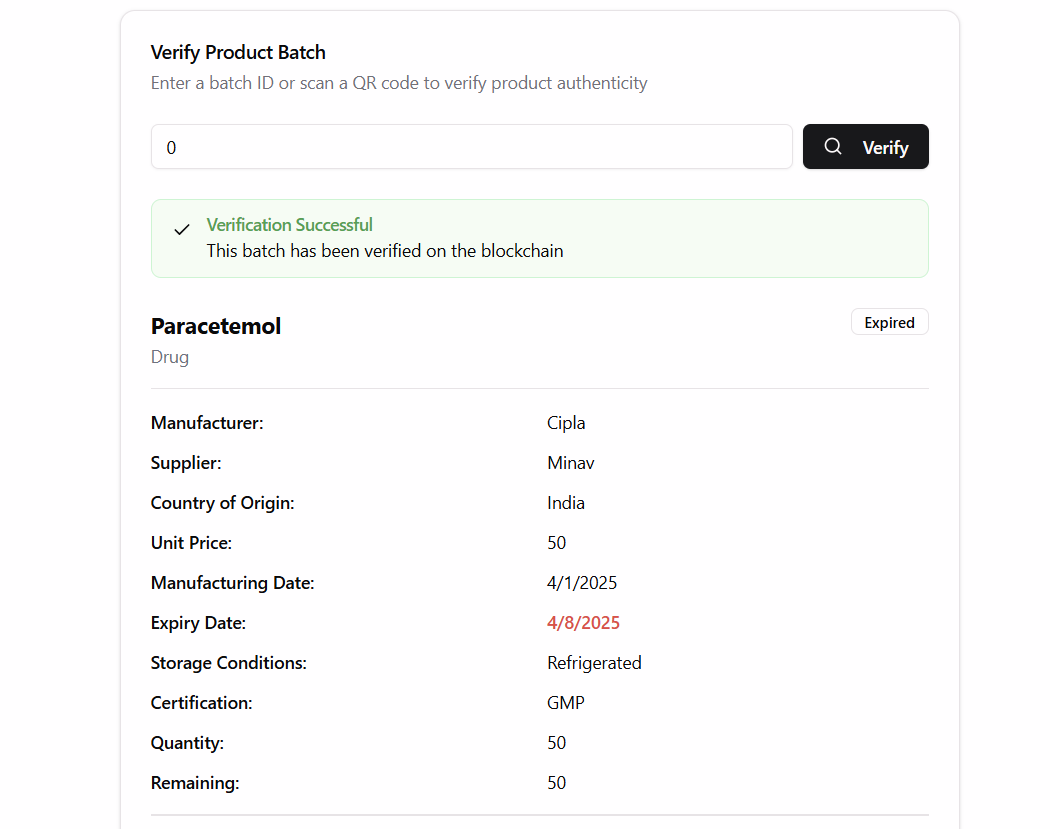


1. **Batch Transfer Form**

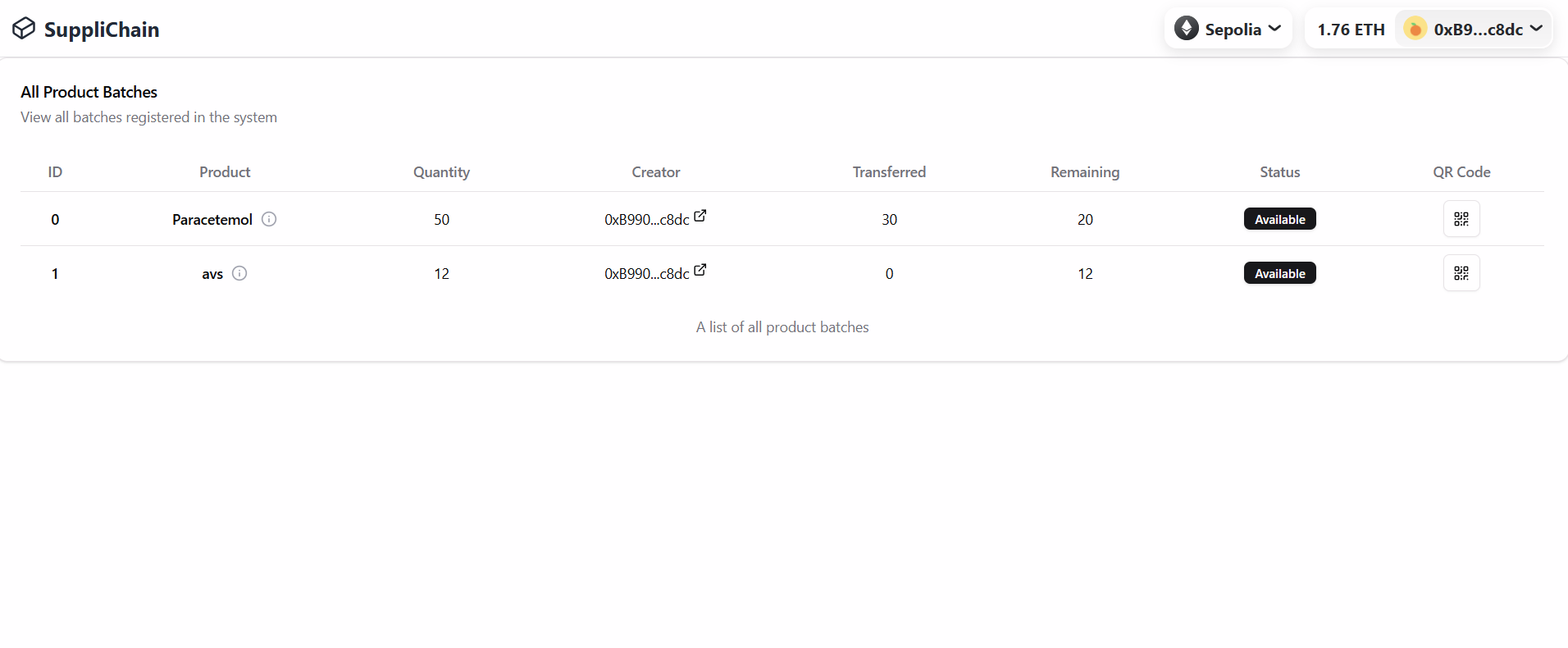
Distributors and retailers can update the status of the drug. Each transaction is recorded on the Ethereum blockchain using smart contracts.



1. **Batch Verification Page**Consumers can scan or input a drug ID to verify its authenticity and trace its history through the supply chain. The system fetches data directly from the blockchain using readContract via Wagmi.

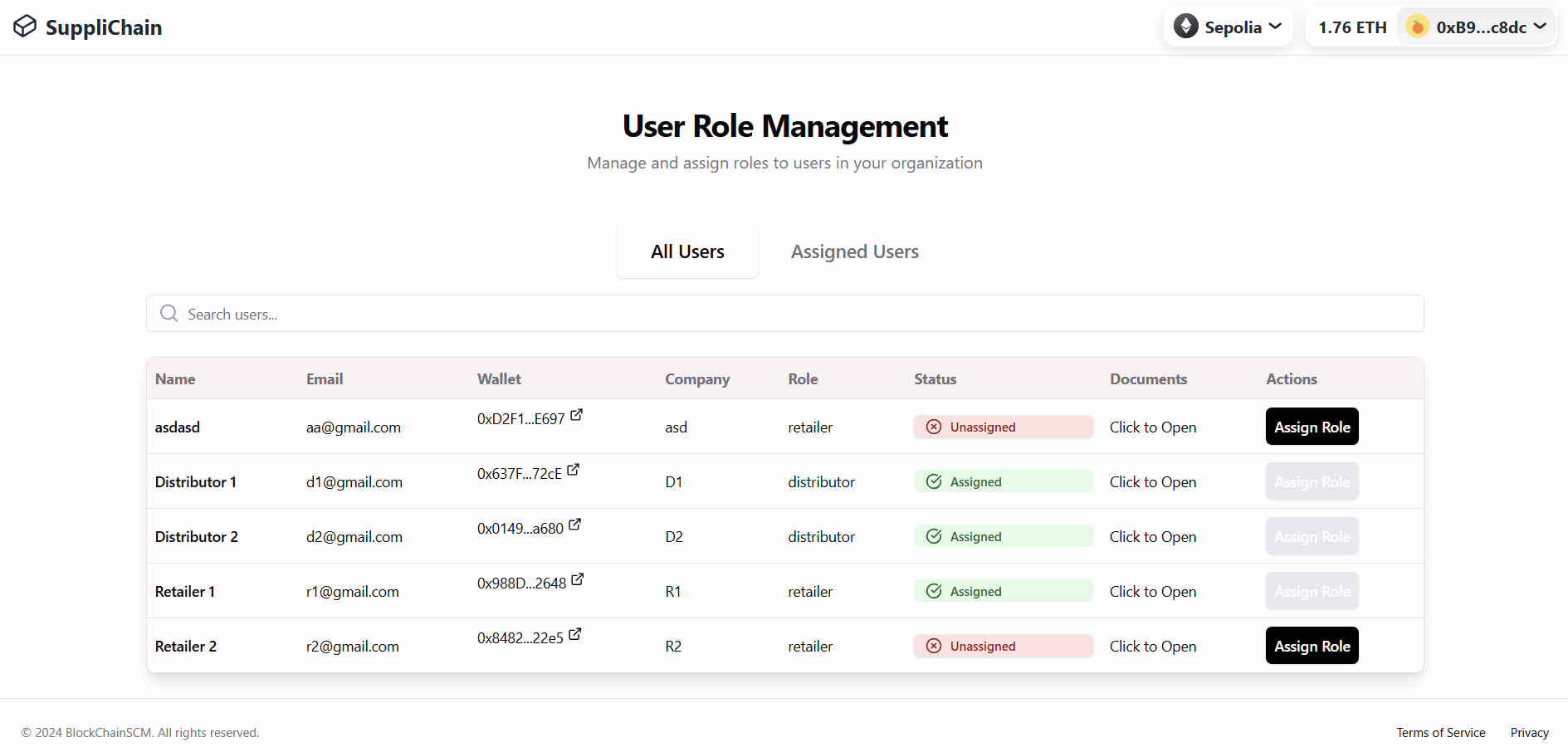


1. **Check All Batch Details**



1. **Role Approval Dashboard (Admin)**

Admins can approve or reject role requests. This interface interacts with backend services which trigger role-setting smart contracts if approved.



1. **Git link (Maintain Coding standards)**

**- Link**

[**https://github.com/MinavKaria/Pharma-SupplyChain-Blockchain.git**](https://github.com/MinavKaria/Pharma-SupplyChain-Blockchain.git)

**- Screenshot of contribution over repository**

****