WELCOME TO THE DECENTRALIZED PHARMACEUTICAL SUPPLY CHAIN

INTRODUCTION →

We at Dragonfly are thrilled to present our innovative prototype for pharmaceutical supply chain management. Our solution leverages the power of blockchain and web development to ensure the authenticity and traceability of drugs through a sophisticated system of QR codes and cryptographic keys. Additionally, our prototype tracks the environmental conditions, such as temperature and humidity, that the drugs experience during transit, using simulated IoT data.

For demonstration purposes, we have utilized a script to generate random temperature and humidity data, providing an interface to monitor these conditions. In a real-world application, this script could be replaced by actual IoT sensors, hardware, and oracle technologies to seamlessly integrate off-chain data with the blockchain.

Our supply chain model is designed to reflect real-world scenarios by managing drug transfers in batches (Collections of identical drugs) rather than individual units. This approach mirrors practical practices, as drugs are typically transferred in bulk from one stakeholder to another, rather than in single units.

To keep the prototype focused and manageable, we have excluded monetary transactions from the smart contract. Instead, our system highlights the drug transfer process in a simplified yet unique manner.

We hope you find our prototype insightful and appreciate the innovation behind our approach.

FILE MANAGEMENT →

Now introducing our whole file management process with important files and folders -

medKART Folder :

1. **frontend Folder :**
    - **migrations Folder**: Contains migration scripts for smart contracts.
    - **node_modules Folder**: Includes all dependencies for the frontend project.
    - **public Folder**: Stores static files such as images and other assets.
    - **src Folder**:
        - **components Folder**: Contains React components for the frontend.
        - **contracts Folder**: Houses smart contracts.
        - **truffle_abis Folder**: Contains JSON files for smart contract ABI after compilation.
        - **App.jsx**: Manages frontend routing and main application setup.
    - **truffle-config.js**: Configuration file for connecting to the local Ethereum environment (e.g., Ganache).
    - **iot.js**: Script for manual IoT interactions.
    - **package.json**: Defines frontend dependencies and scripts.
    - **package-lock.json**: Locks the frontend dependencies to specific versions.
2. **backend Folder :**
    - **config Folder**: Contains database connection details and configuration.

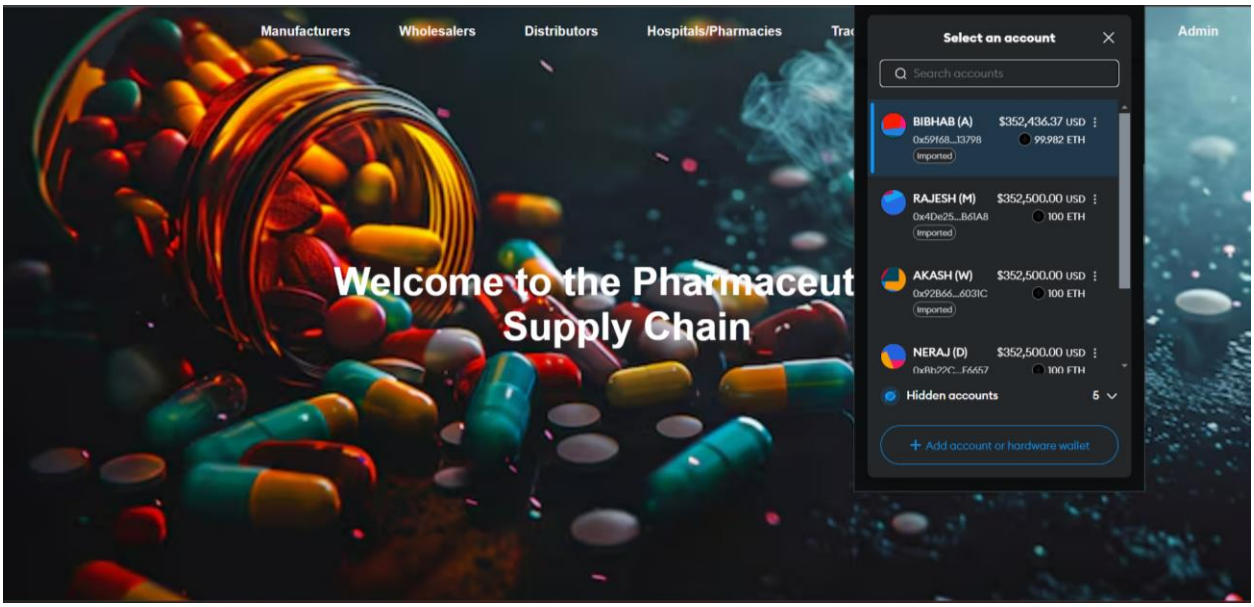
- **controllers Folder**: Contains code for handling API requests and business logic.
- **middlewares Folder**: Includes middleware for request validation and processing.
- **models Folder**: Defines database schemas and models.
- **node_modules Folder**: Includes all dependencies for the backend project.
- **routes Folder**: Contains route definitions and handlers.
- **utils Folder**:
  - **zod.js**: Validation details using Zod.
  - **jwt.js**: JWT authentication and validation utilities.
- **.env**: Contains sensitive environment variables, including secret keys and Mongoose connection URL.
- **package.json**: Defines backend dependencies and scripts.
- **package-lock.json**: Locks the backend dependencies to specific versions.
- **server.js**: Sets up the server, initializes routes, and connects to the database.

3. **output Folder :**
   - Stores output images and related files.
4. **documentation File :**
   - Contains documentation for the project.

PROTOTYPE →

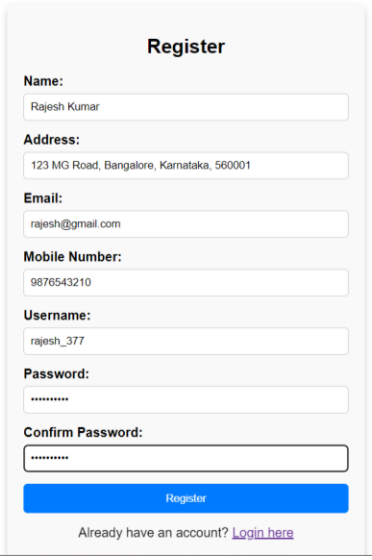
In this section, we will explore the complete supply chain process through detailed explanations and output photos. We will also illustrate how specific components and files contributed to the development of each page, providing a comprehensive overview of their roles and functionalities. Additionally, all routes used within the application are defined in the **App.jsx** file, ensuring clear routing management across the frontend –

1) MAIN HOME PAGE



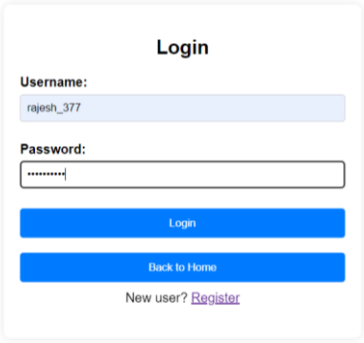
This is the first page or the landing page of our website. The code of this page is handled in **Home.jsx** file of the components folder. Here I opened the Metamask extension and uploaded the accounts from local Ganache environment for further processing of my website. Now it is the duty of every stakeholder to go and register themselves and send a request message to the admin(BIBHAB) of the supply chain so that they can become a part of the supply chain. Since I am using local Ganache environment so by default my 1st account is the admin account as the deployment of the smart contracts occurred from the first account which is named as BIBHAB(A) on the metamask.

2) REGISTRATION PAGE



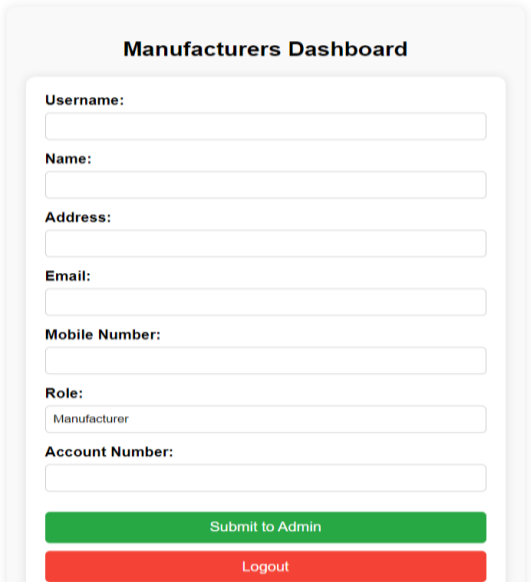
So this is the registration page for each and every stakeholder of the supply chain. On clicking respective buttons (Manufacturers/ Wholesalers/ Distributors/ Hospitals/Pharmacies) on the landing page if the user is not preregistered to any metamask account and using that account to open the respective stakeholder page then he/she will get landed to a login page where they will be asked to register if they are new user. So here on the register form they need to enter their details and register themselves for the supply chain. The registration page for manufacturers, wholesalers, distributors and hospitals/pharmacies is coded in **Register1.jsx, Register2.jsx, Register3.jsx and Register4.jsx** respectively. The admin should also register himself if he uses the supply chain for the first time and his registering process is coded at **Register6.jsx** . All these register files are present inside the components folder. Once each of them registered there details gets stored inside the centralised MongoDB database which stores information that are not related to the main supply chain.

3) LOGIN PAGE

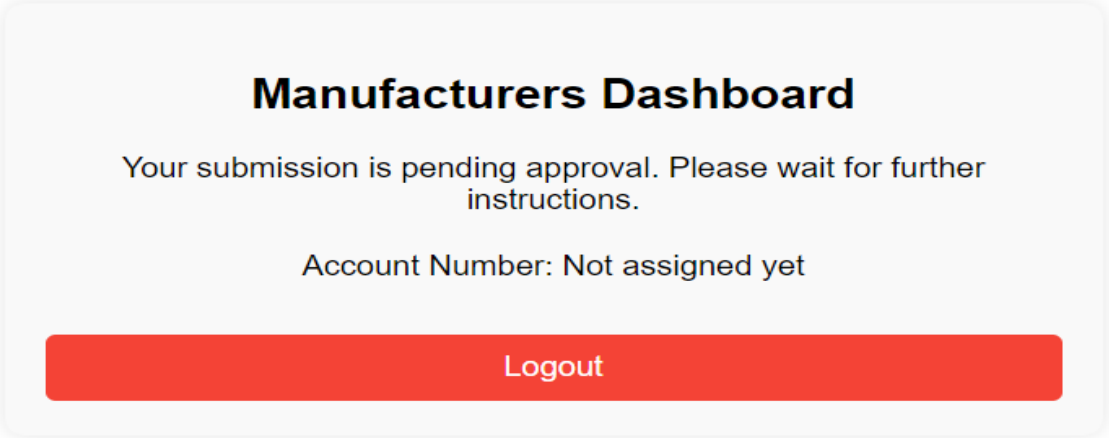


Once the user registers himself/herself he/she will get redirected to this login page where they need to login with their details to enter the respective stakeholder dashboard page. The code for login page are written on Login1.jsx, **Login2.jsx, Login3.jsx, Login4.jsx and Login6.jsx** files for Manufacturers, Wholesalers, Distributors, Hospitals/Pharmacies and Admin respectively inside the components folder.

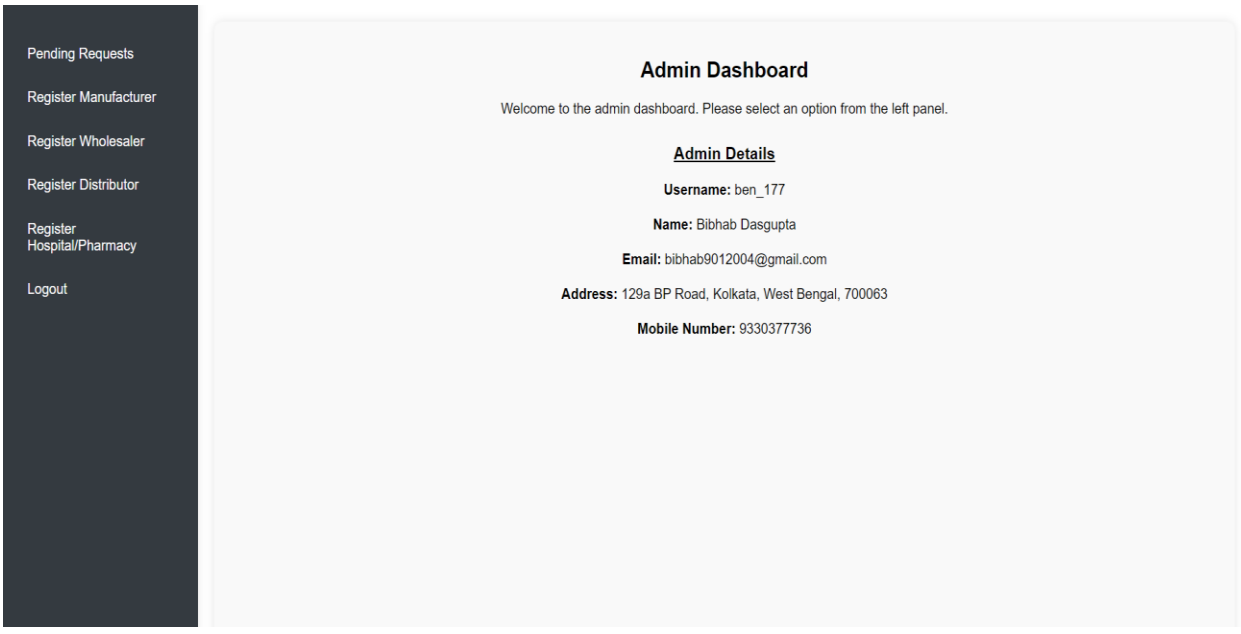
4) DASHBOARD PAGE



This is the dashboard page for the manufacturer. Similar dashboard pages are also there for remaining stakeholders which you can see from the output folder. The dashboard codes are written on **DashBoard1.jsx, DashBoard2.jsx, DashBoard3.jsx and DashBoard4.jsx** inside the components folder for manufacturers, wholesalers, distributors and hospitals/pharmacies respectively. If a user doesnot have any account from beforehand on clicking the login button they will get redirected to this dashboard page which will ask for the details of the user and on submitting this form from frontend a request will go the admin to accept the user as a part of the supply chain.



If the request of a particular user is still not accepted by the admin this page will come on dashboard when that user will again log in.

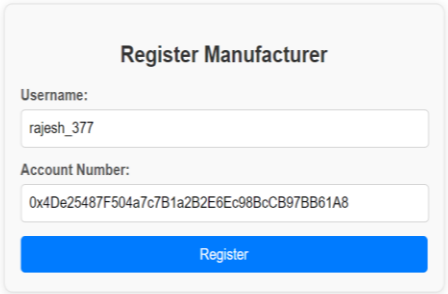


Now let us accept the request of our manufacturer. When the admin logs in with his credentials he will get his dashboard which I quite different from the dashboard of the other users. The code for dashboard page of admin is coded inside **adminDashBoard.jsx** file inside the components folder. The admin page has 4 routes for registering the stakeholders, 1 route to get pending requests and a logout route. All these routes are handled inside **Admin.jsx** file of the components folder. Now let us see the pending requests made to the admin by the stakeholders by going to Pending Requests route. (The account number is also a part of the table. Due to technical issue, it did not came on the screenshot. Sorry for the inconvenience caused)

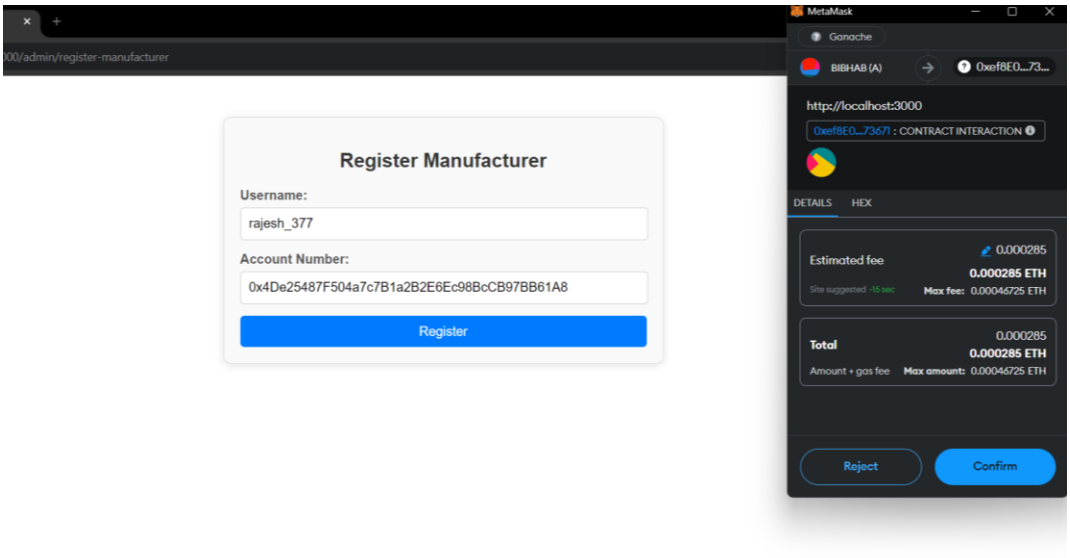
Pending Requests

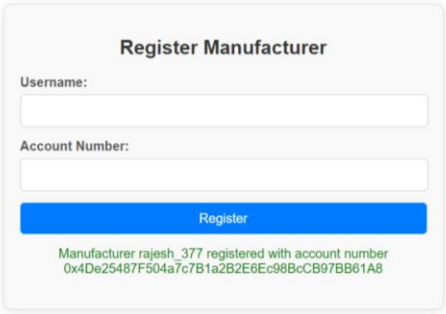
Username	Name	Address	Email	Mobile Number	Role
rajesh_377	Rajesh Kumar	123 MG Road, Bangalore, Karnataka, 560001	rajesh@gmail.com	9876543210	Manufacturer
akash_915	Akash Sharma	456 Park Street, Kolkata, West Bengal, 700016	akash@gmail.com	9123456789	Wholesaler
neraj_333	Neraj Chopra	789 Nehru Nagar, Mumbai, Maharashtra, 400013	neraj@gmail.com	9987654321	Distributor
prerna_275	Prerna Mehta	101 Jubilee Hills, Hyderabad, Telangana, 500033	prerna@gmail.com	9876512345	Hospital/Pharmacy

So these are the pending requests. Let the admin decided to register the manufacturer then he will go the register manufacturer route to register the manufacturer.



So here the admin will assign a account number to the manufacturer and register him/her with that account number which eventually gets stored in our PharmaChain smart contract inside the contracts folder.

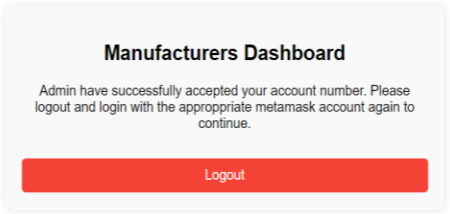




So congratulations!!! The manufacturer is finally registered with his unique account number.

After registering the manufacturer will get the below message if he/she doesnot logins with his respective metamask account number.

So the manufacturer needs to enter with his/her own metamask account number so that he/she can go to his/her home page.



One important thing to note is that in this documentation we showed only the example of how the manufacturer will become the part of the supply chain. The same procedure will follow for all other stakeholders of the supply chain and the necessary output pictures for all stakeholders are present inside the outputs folder.

The **Manufacturers.jsx, Wholesalers.jsx, Distributors.jsx and HospitalsPharmacies.js**x file inside the components folder contains all the essential logic related to the routing of the stakeholders. The **ManufacturerDetails.jsx, WholesalerDetails.jsx, DistributorDetails.jsx and HospitalPharmacyDetails.jsx** contain all the necessary codes for handling the forms which the respective stakeholders need to submit to the admin so that the admin can accept them to be a part of the supply chain.

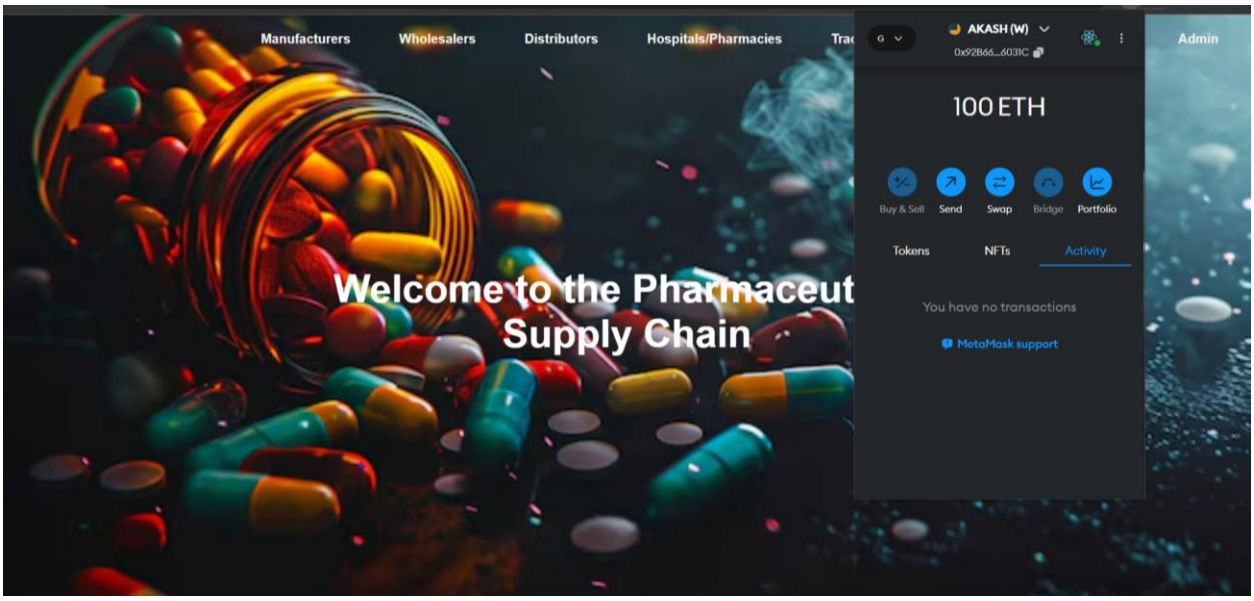
SO ALL THE STAKEHOLDERS HAVE SUCCESFULLY REGISTERED THEMSELVES TO BE A PART OF THIS INNOVATIVE SUPPLY CHAIN. NOW LET'S MOVE ONE STEP FORWARD AND SEE HOW THE SUPPLY CHAIN ACTUALLY WORKS AS A WHOLE!!!

5) THE SUPPLY CHAIN

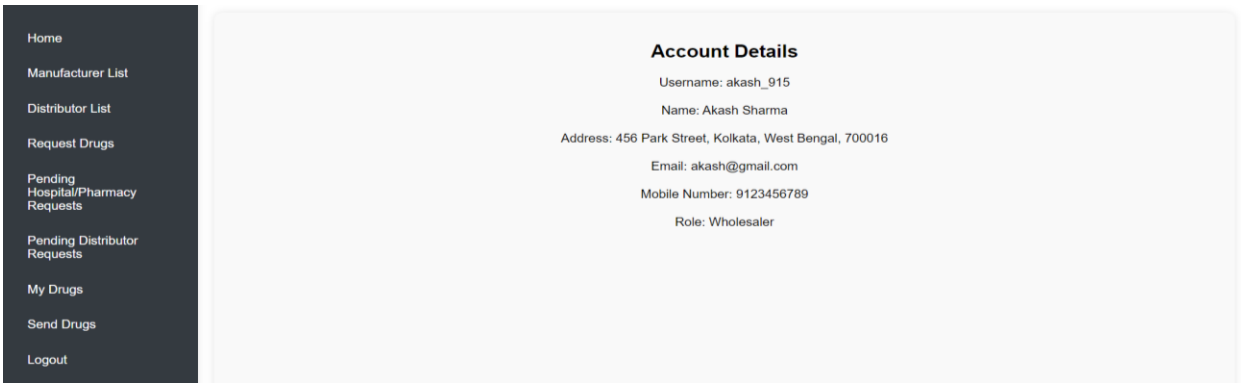
The whole supply chain is managed and controlled by the Drugs smart contract. We are doing to demonstrate the supply chain in 2 phases:

Phase 1 → The wholesaler asks for some batches of drugs from the manufacturer so that they can keep there stocks updated.

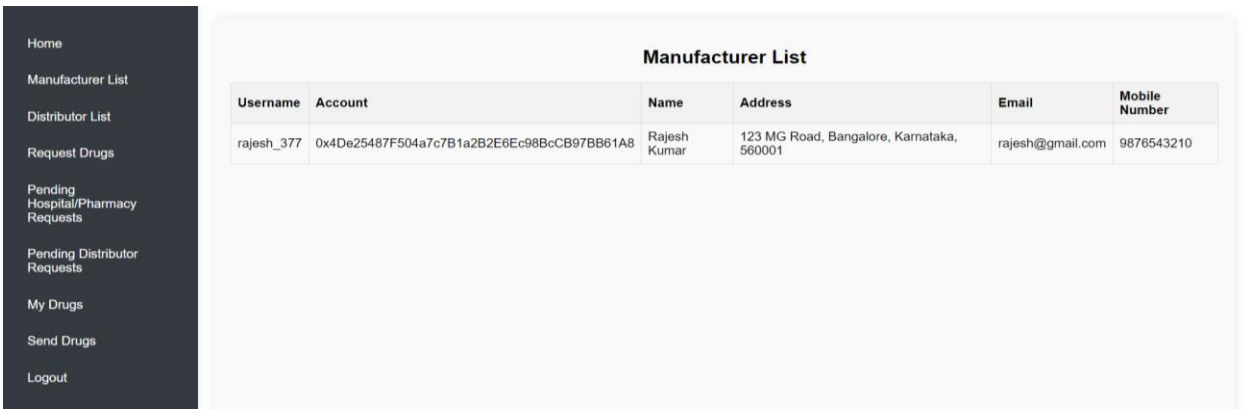
So for the 1st phase, the wholesaler logs into his account through metamask and then looks for the manufacturers he knows from the manufacturer list.



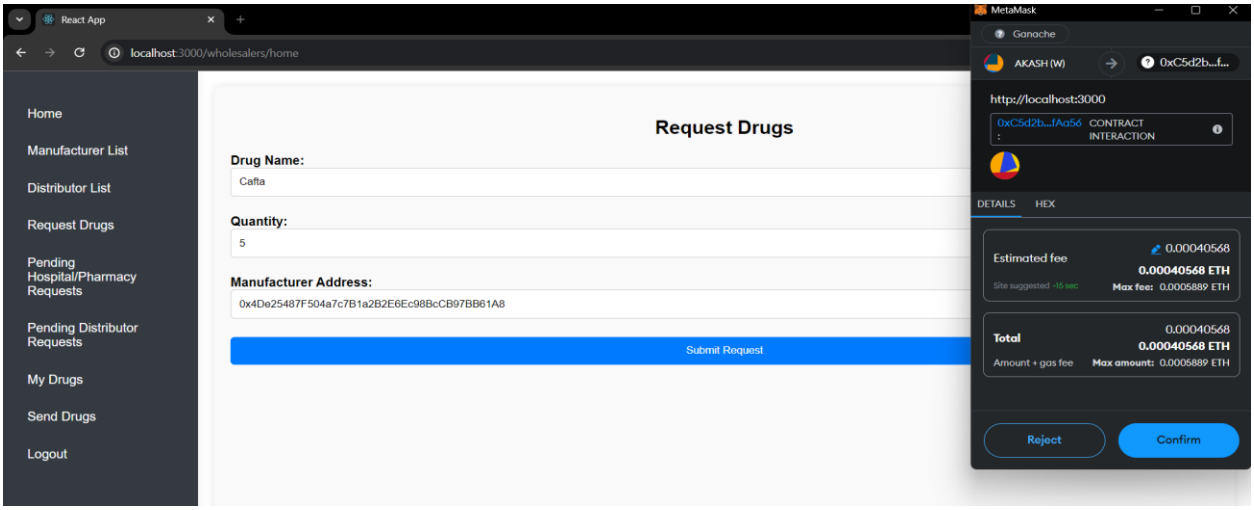
Note: The above picture shows how the wholesaler can navigate to his account using metamask. The same process is used for all the stakeholders to navigate to his/her account. So to keep the documentation precise we are not including the pictures of all the stakeholders navigating to his/her account. Although all the pictures related to this are available inside out output folder.



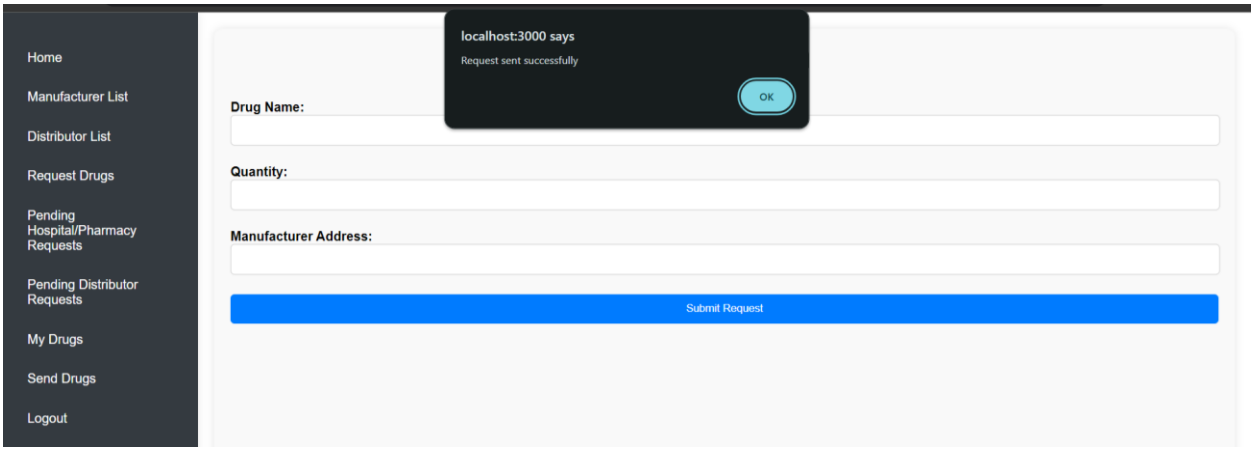
So after logging in the wholesaler reaches to his home page. The code of this page comes from **Home2.jsx** in the components folder. Now to fetch the manufacturers details he/she will go to the manufacturer list.



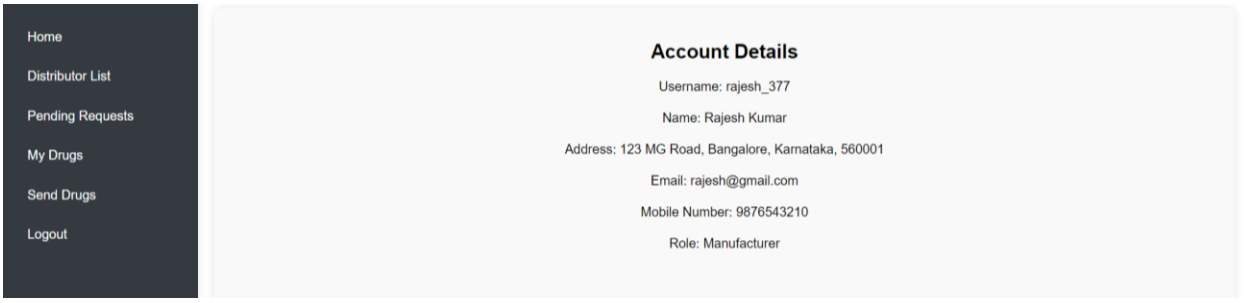
Now after fetching the details of a manufacturer he can choose any manufacturer from the table (here only 1 is there but in reality the smart contract can store multiple manufacturers) and then he/she should go to the Request Drugs option to request a drug from the manufacturer.



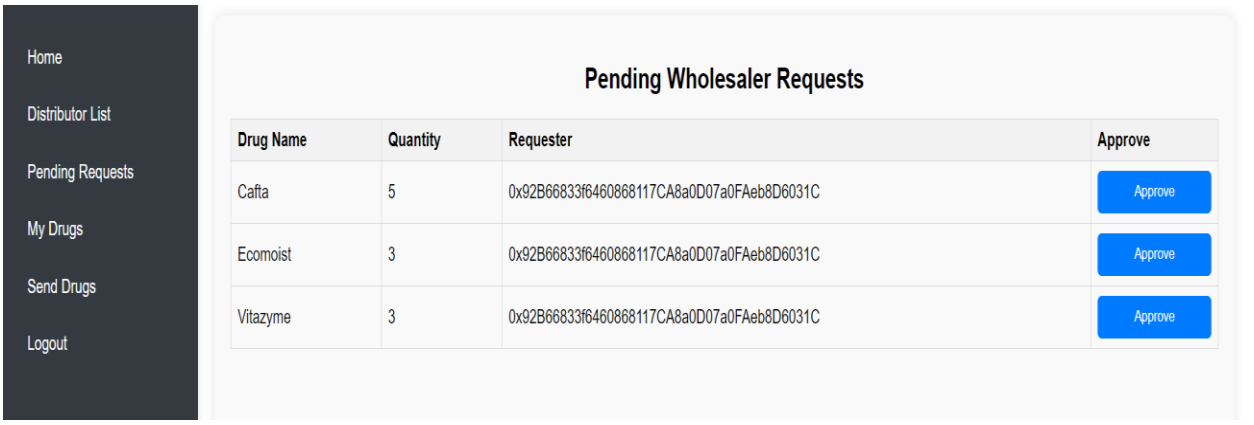
The wholesaler fills the form with his necessary drugs, quantity of drugs he/she require and the manufacturer address from where he/she is expecting the drug batch to come.



A success pop up comes after the wholesaler has successfully requested the drug. Now let us visit to the manufacturers home page to see whether he/she got the drug request.



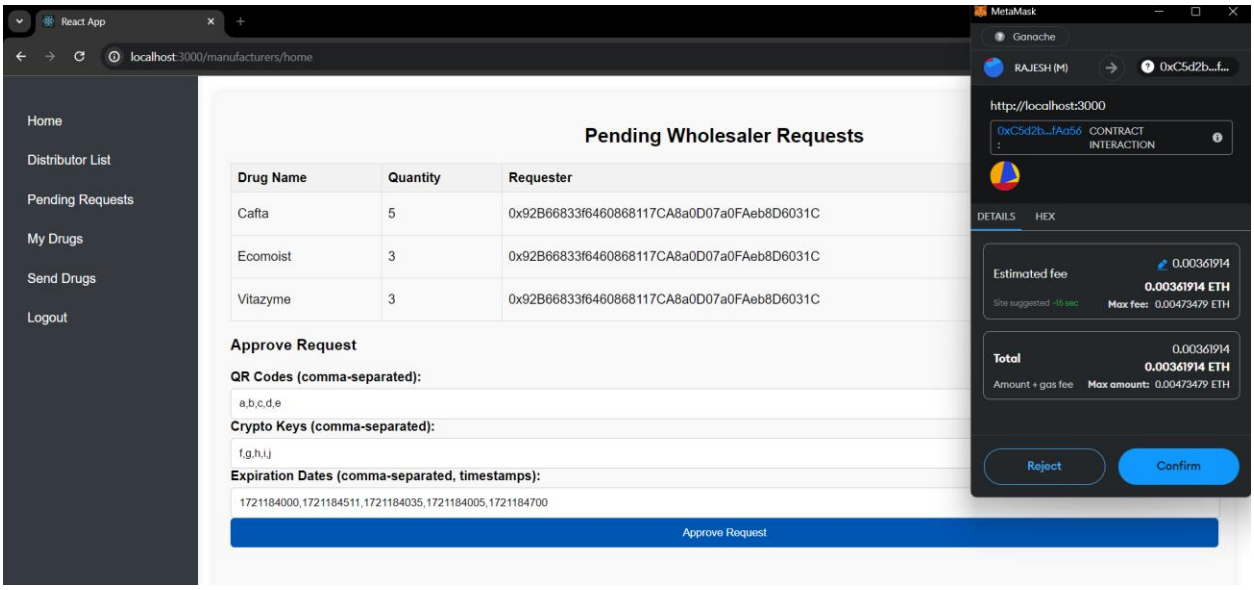
So this is the manufacturers home page. The code of this page is written on **Home1.jsx** inside the components folder. Now lets navigate to the Pending Requests button to see the pending requests.



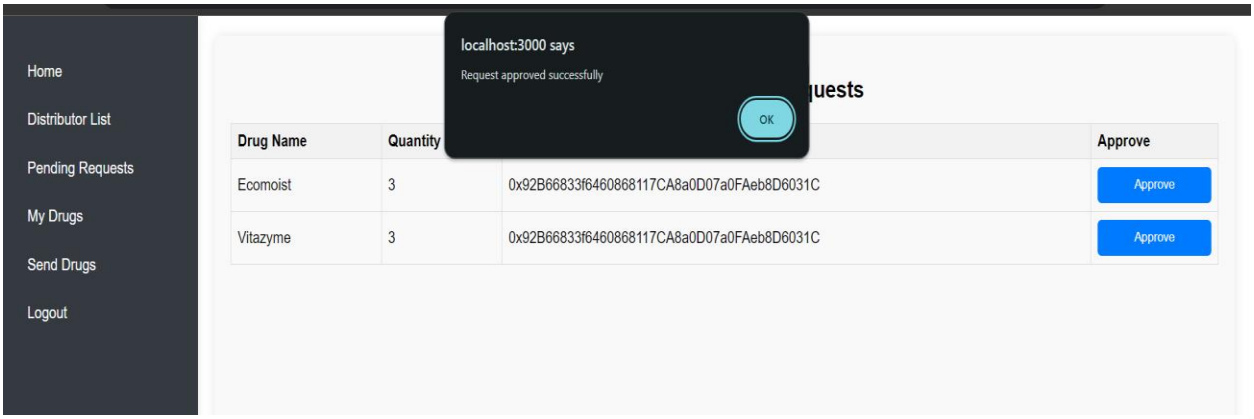
Note: The wholesaler made 2 more requests. The picture of those pages are included inside the output folder. For simplicity, we are showing based on one request.

Yay! The manufacturer got the requests from the wholesaler.

Now for approving the request made by the wholesaler, the manufacturer clicks the approve button.

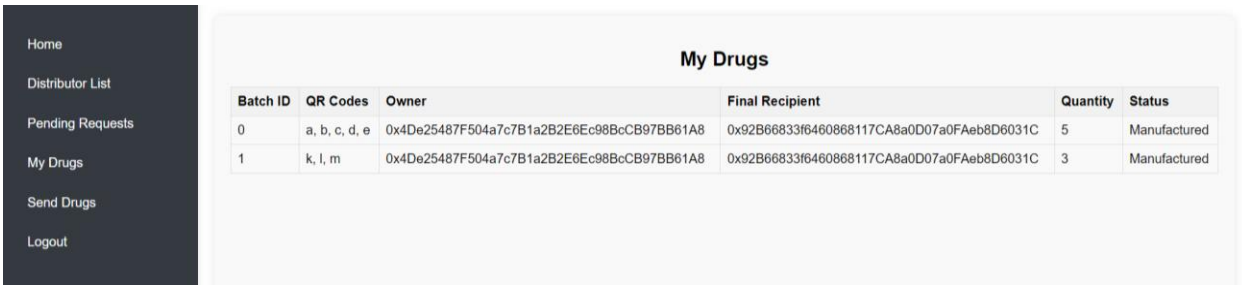


On clicking the approve button, a form appears asking for necessary details about the drugs. Since the manufacturer is the emerging point of any transaction he/she has the right to provide the qr codes and cryptographic keys along with expiration date to his/her drugs. After providing the necessary details the manufacturer approves the request using metamask and a pop up comes with an accept message.



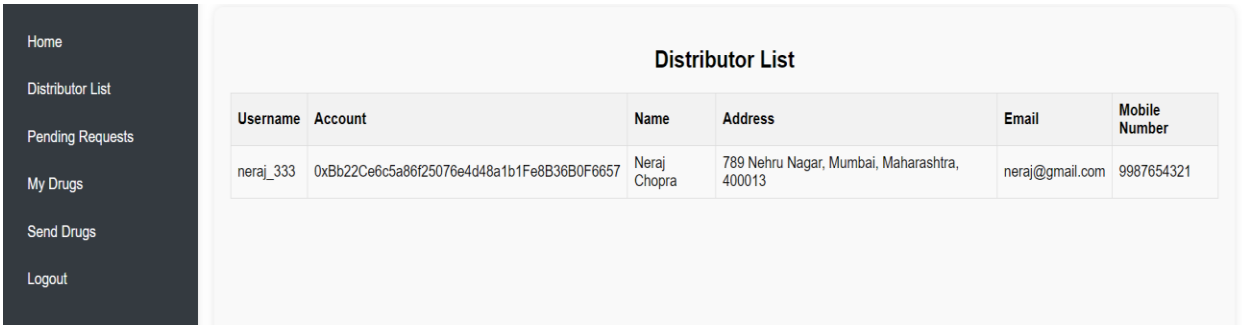
After approving the drug request the new drug/batch of identical drugs becomes a part of the supply chain and it initially gets stored on the my drugs page of the manufacturer because on approving the wholesaler request it is assumed that the

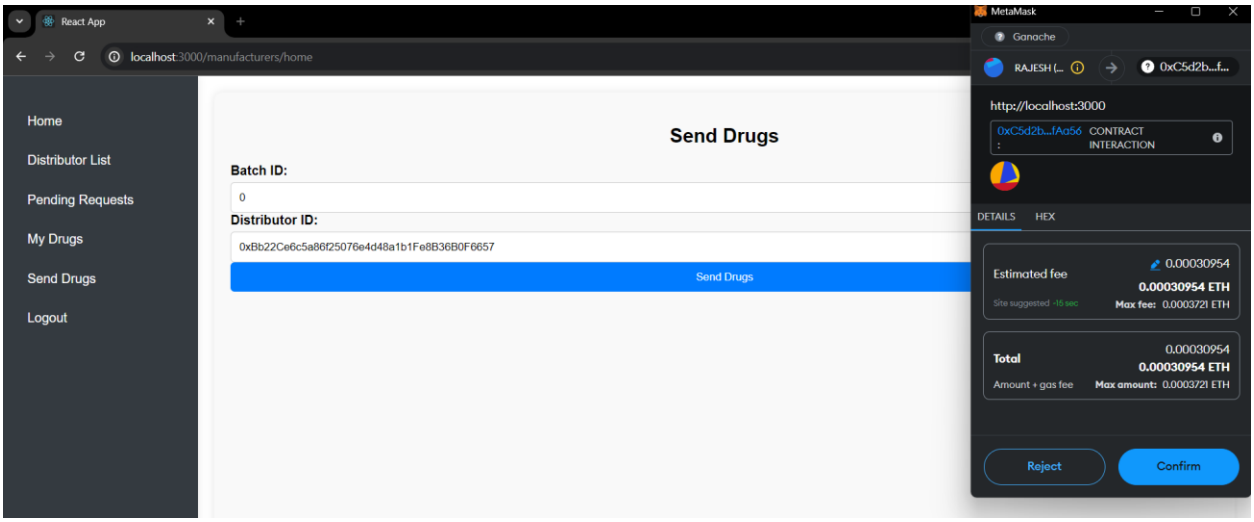
manufacturer packed the drugs based on the wholesalers requests and gave it a unique batch id so that it can be tracked throughout the supply chain.



So the manufacturer created Batch ID 0 for the wholesaler request we are dealing with in this documentation.

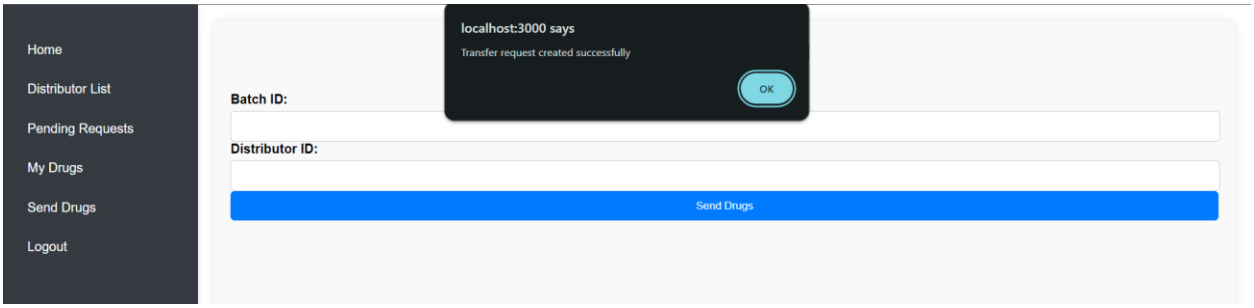
Now the manufacturer should see the distributor list form the Distributor List button so that he/she can choose a suitable distributor for the transaction of the drugs from him/her to the wholesaler.



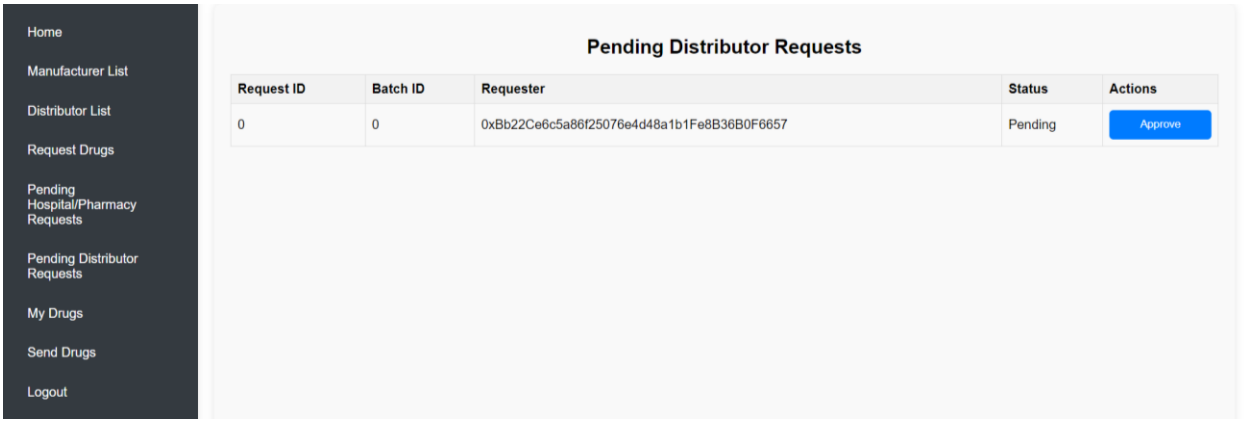


After choosing a suitable distributor the manufacturer should navigate to the Send Drugs button and send a delivery request to the distributor. This transaction also

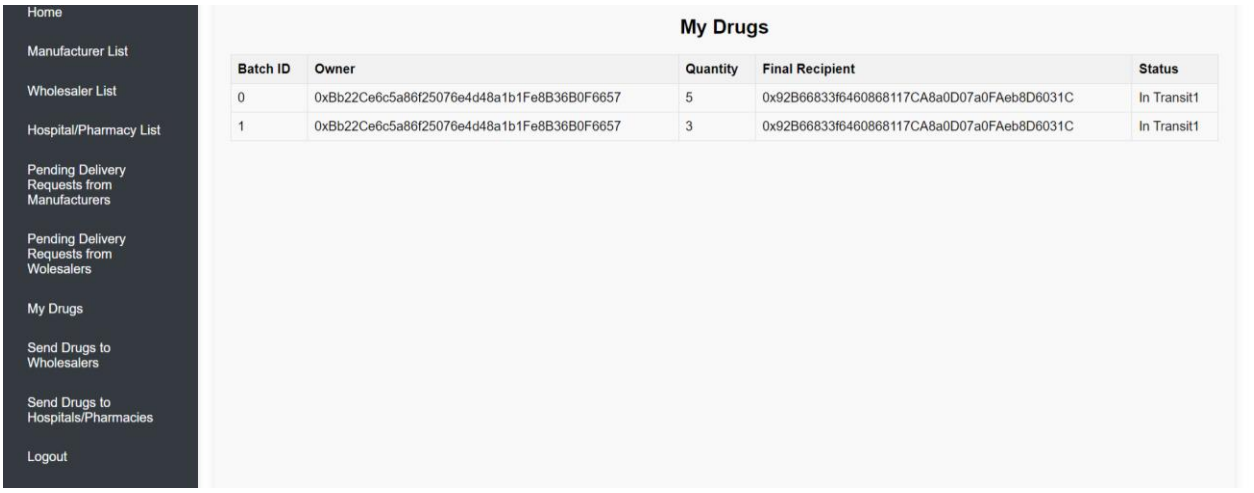
goes through metamask and an alert is generated after the manufacturer confirms the transaction though metamask.



Now lets go to the distributors home page (code written in **Home3.jsx** inside components folder) to see whether he/she got the delivery request or not.

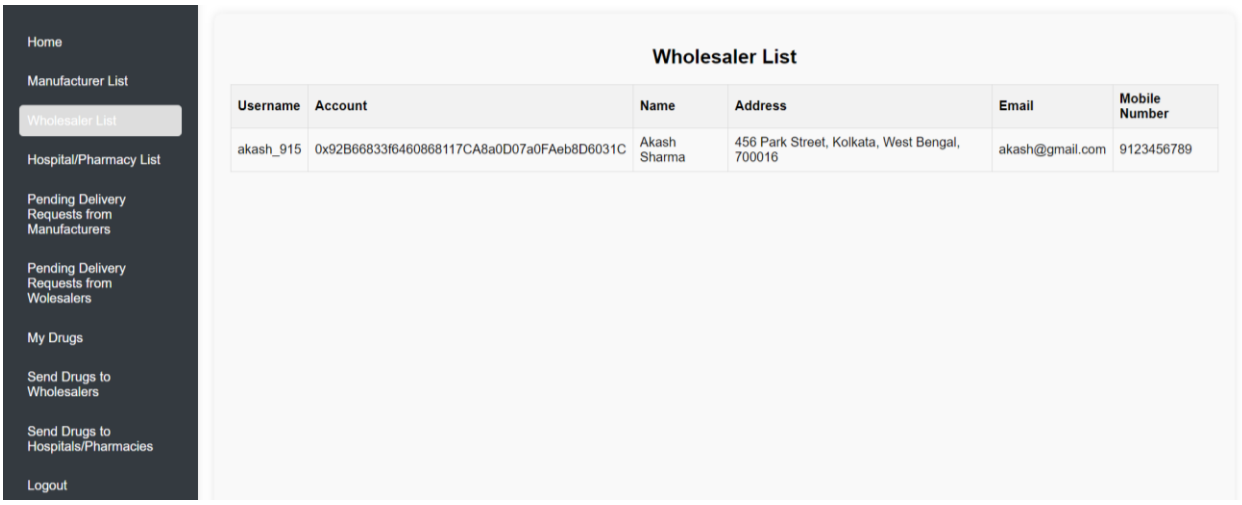


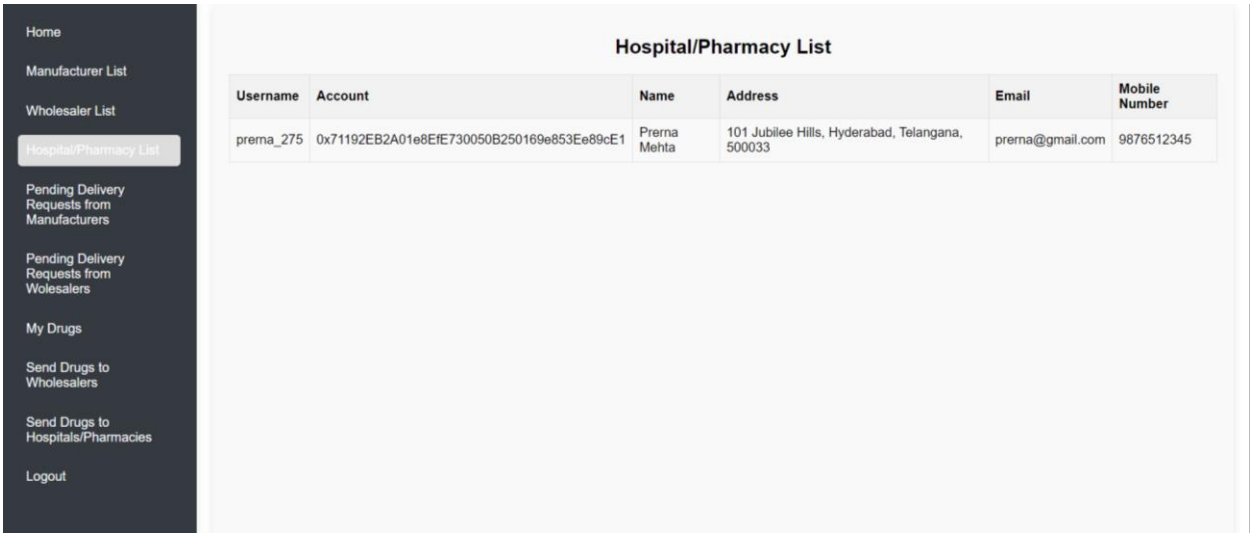
So the distributor got the delivery request which he/she need to approve so that the drug comes to him as an intermediatory stakeholder.



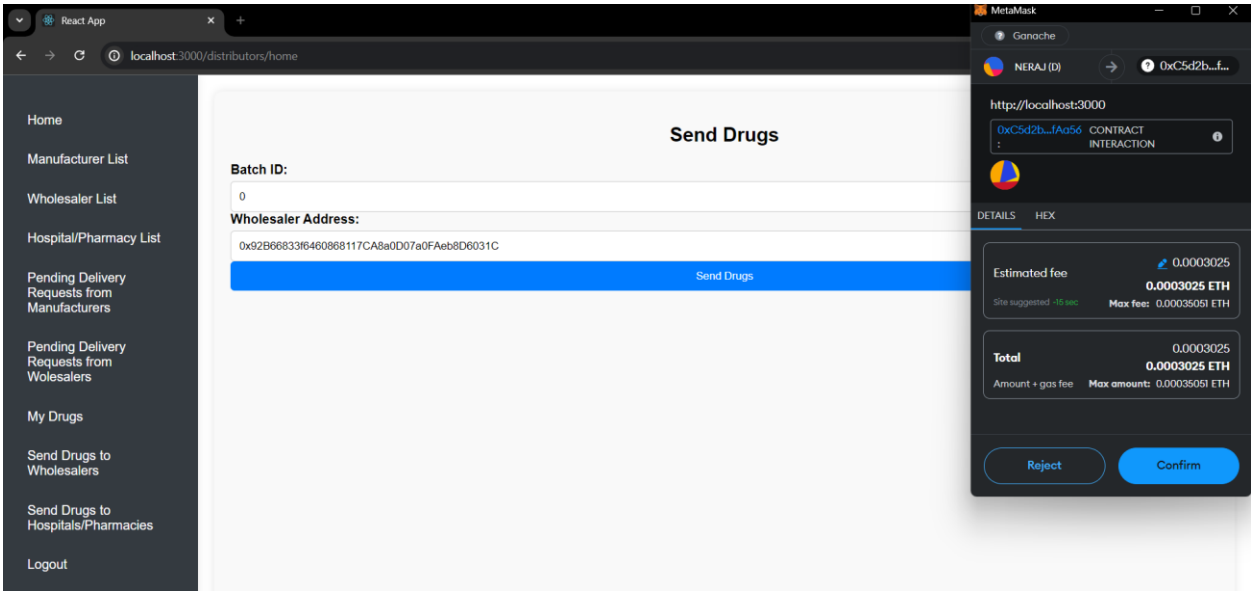
So the drugs with batch ID 0 which we are considering in our demonstration is now having the owner as the distributor at Transit 1 state because the ownership

of the drug is now passed from the manufacturer to the distributor. The distributor also gets to know about the final recipient of the drug from the table. He checks whether the final recipient is a wholesaler or hospitals/pharmacies from the Wholesaler and Hospital/Pharmacy List.

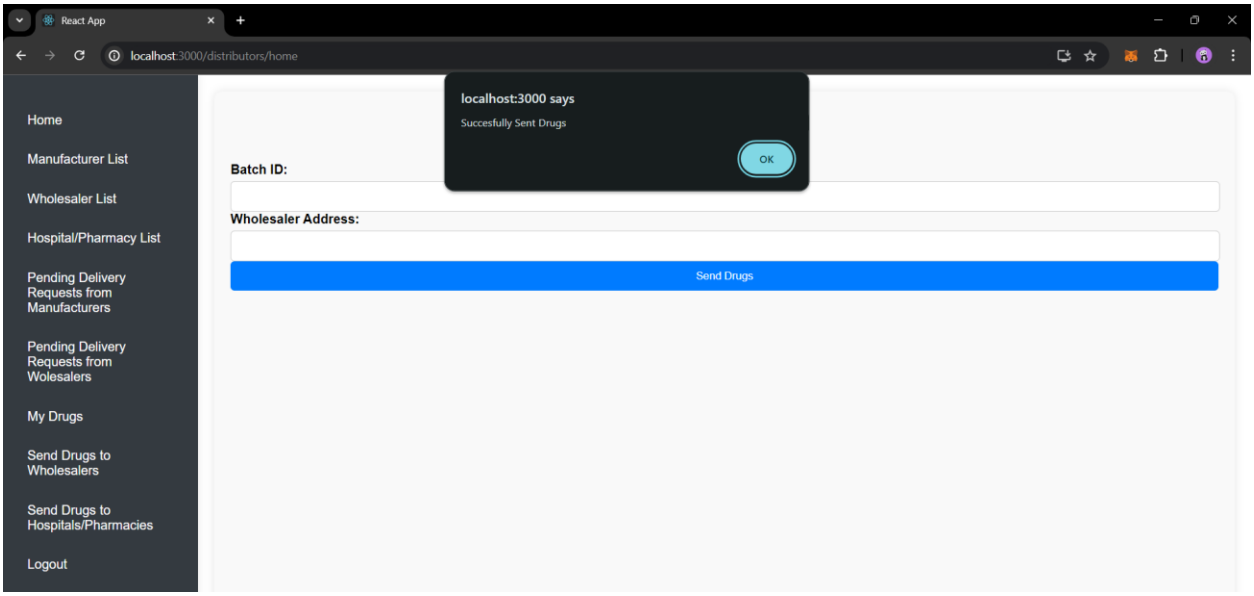




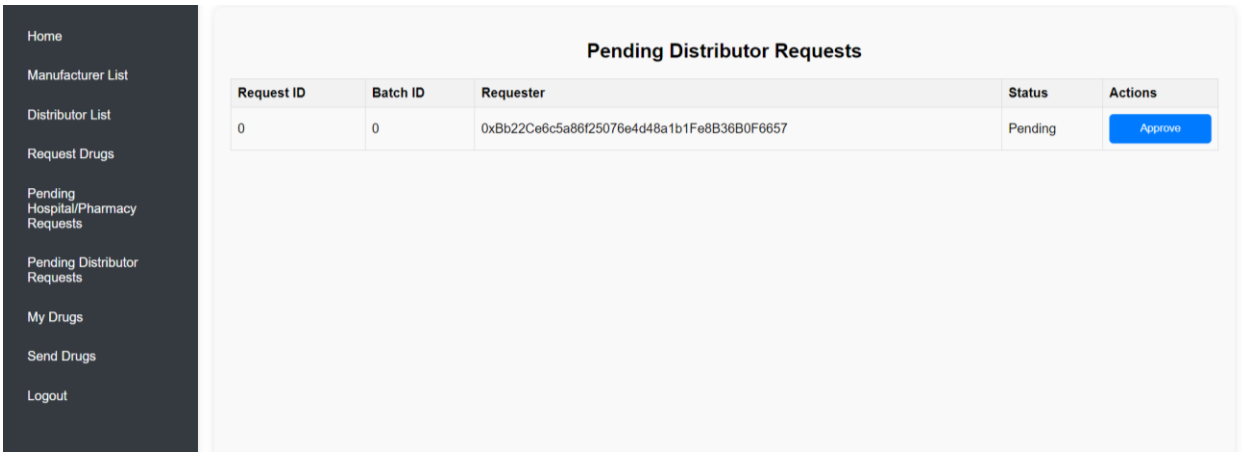
So now he proceeds for sending the drug to the appropriate recipient.



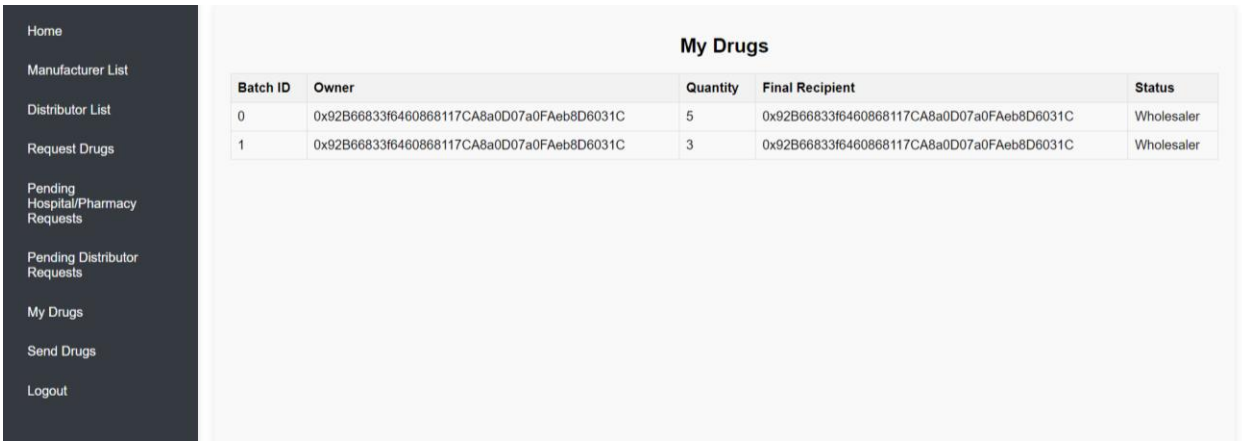
After clicking the confirm button on metamask the distributor finally sends a delivery request to the wholesaler.



Now let us check whether the wholesaler got the delivery request or not from the distributor.



So, the pending distributor requests are successfully fetched by the wholesaler and on approving this request via metamask the wholesaler will finally get the drug/batch of drugs from the manufacturer via a distributor.

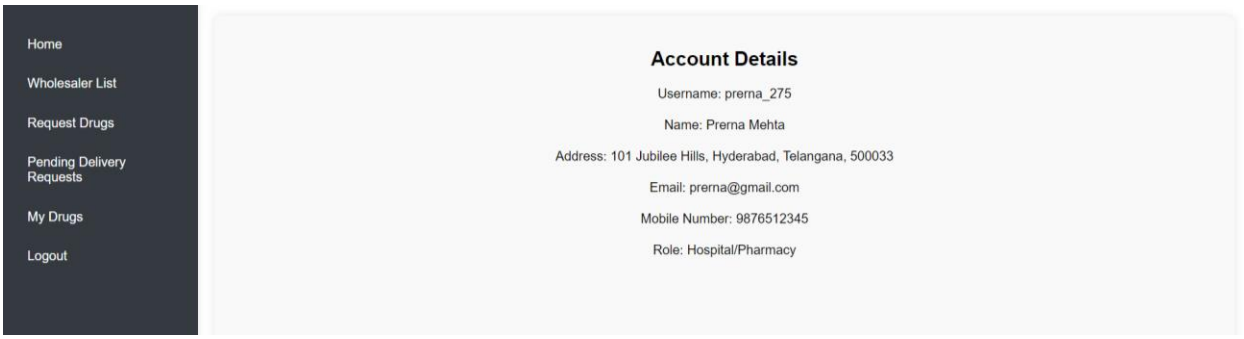


So, the wholesaler finally got the batch of drugs (BatchID=0 which we are dealing with) from the manufacturer via a distributor. We can see that the owner and the final recipient of the drugs are same. So, we can conclude that the drugs reached their final destination.

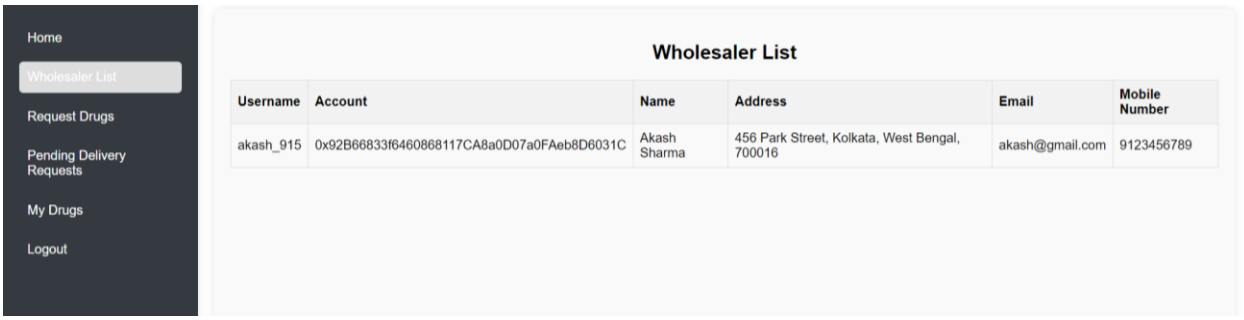
So that’s all about the first phase of our supply chain where we successfully sent the drugs to wholesalers from manufacturers based on their request via a distributor.

Phase 2 → The hospital asks for some batches of drugs from the wholesaler so that they can successfully treat their patients.

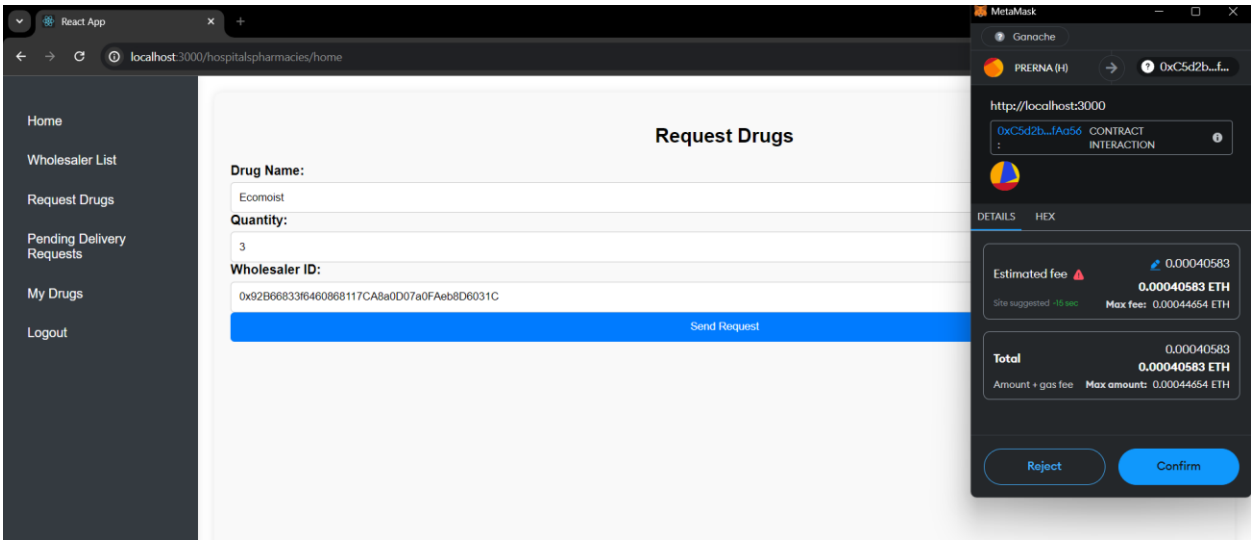
So for the 2nd phase, the hospital logs into his account through metamask and then looks for the wholesalers he/she knows from the wholesaler list.



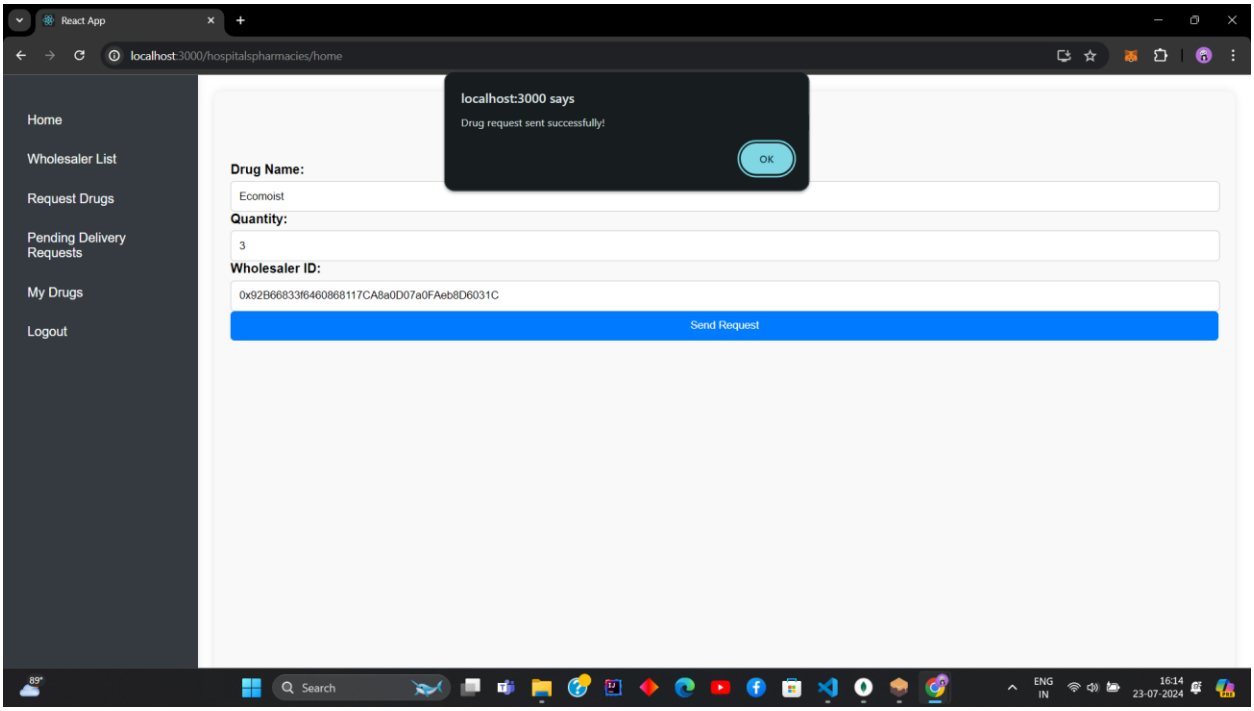
So after logging in the hospital reaches to his/her home page. The code of this page comes from **Home4.jsx** in the components folder. Now to fetch the wholesalers details he/she will go to the wholesaler list.



Now after fetching the details of wholesalers he/she can choose any wholesaler from the table (here only 1 is there but in reality the smart contract can store multiple wholesalers) and then he/she should go to the Request Drugs option to request a drug from the wholesaler.

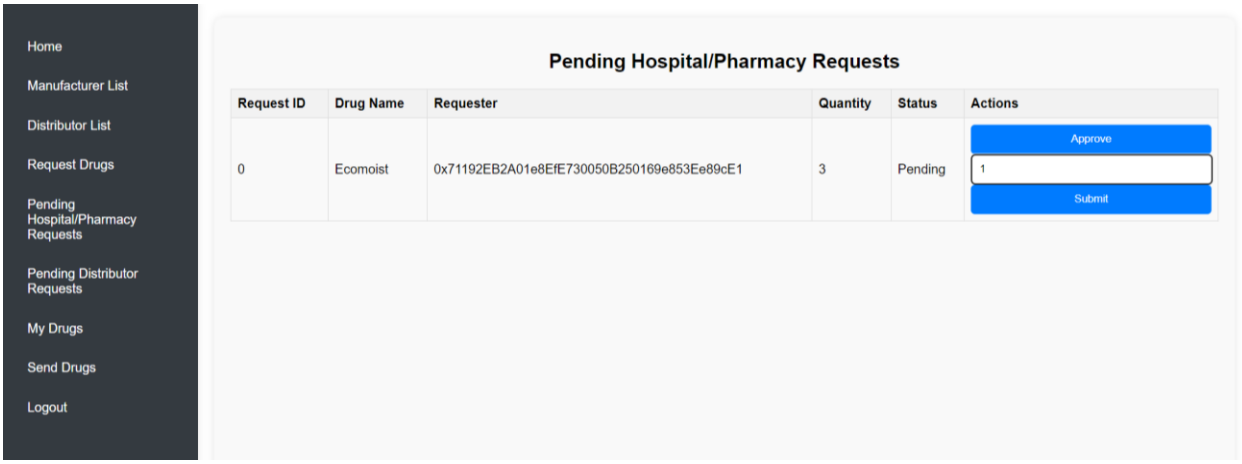


The hospital fills the form with his necessary drugs, quantity of drugs he/she require and the wholesaler address from where he/she is expecting the drug batch to come.



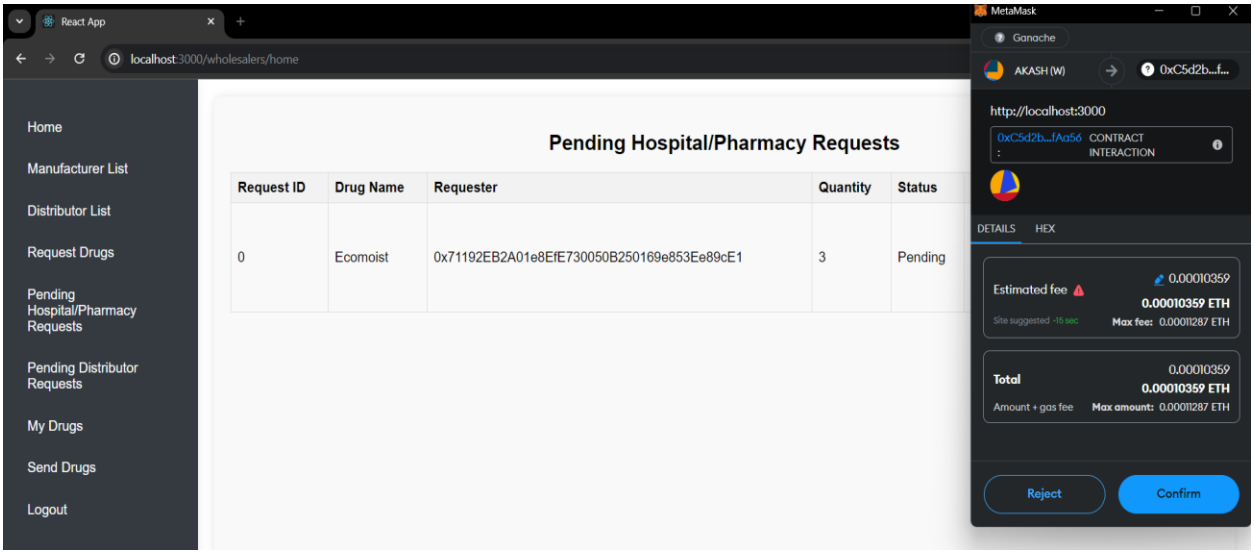
A success pop up comes after the hospital has successfully requested the drug. Now let us visit to the wholesalers home page to see whether he/she got the drug request.

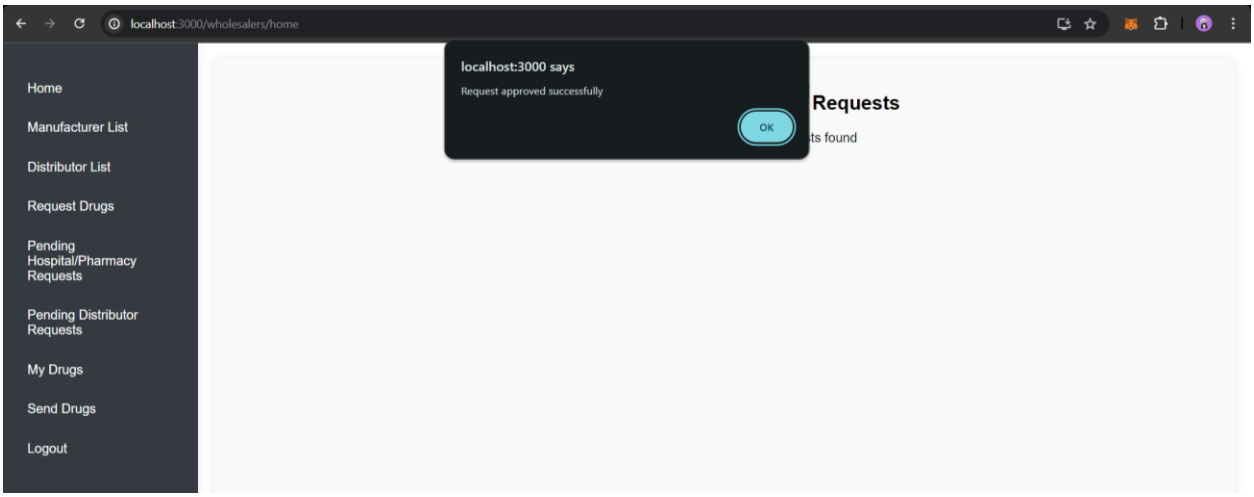
<u>Imp Note:</u> For the phase 2 of the supply chain we are considering the batch with BatchID 1 and not the batch with BatchID 0 which we considered in phase 1. This situation may arise in a real life scenario where a wholesaler has various batches of drugs but the hospital need some specific drugs only. In maximum cases, wholesalers have a more stock than the hospitals/pharmacies.



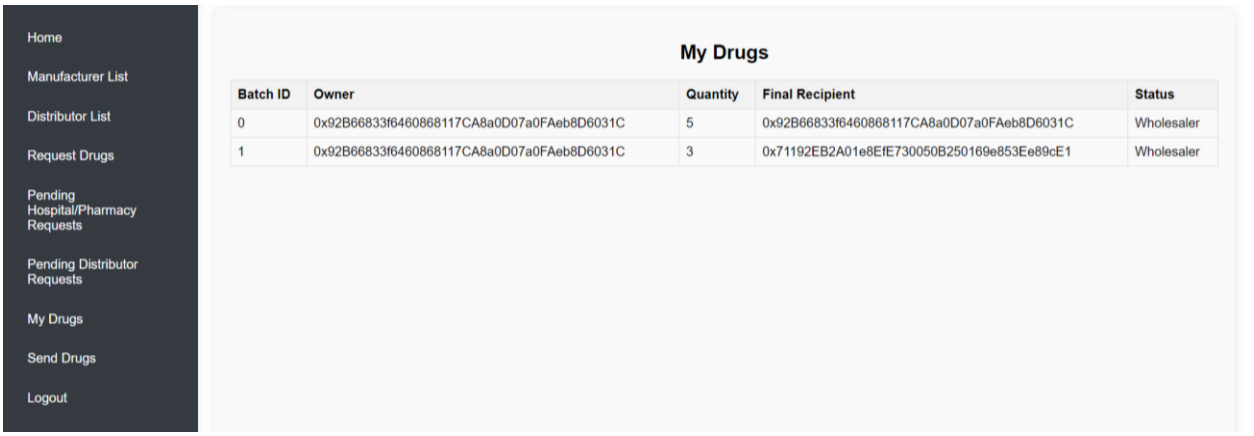
Yay! The wholesaler got the requests from the hospital.

Now for approving the request made by the hospital, the wholesaler clicks the approve button which then asks for the BatchID the wholesaler is wanting to send to the hospital and on further clicking the submit button after setting a BatchID a metamask transaction occurs which finally approves the request of the hospital.



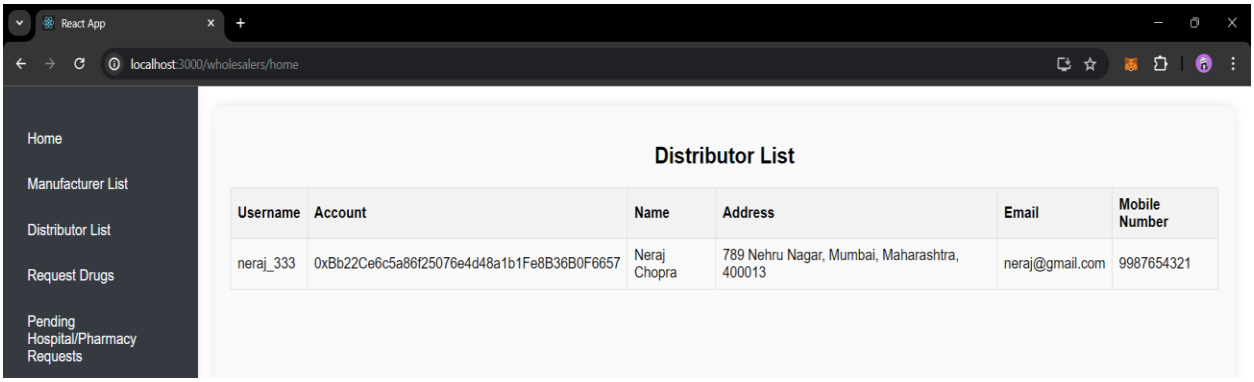


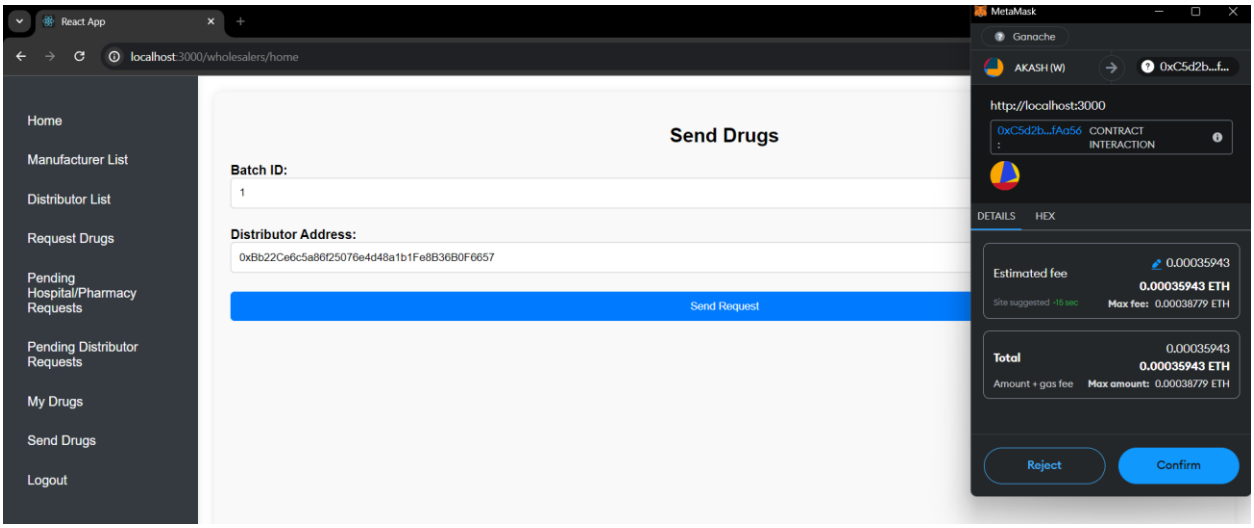
After approving the drug request a success alert appears on screen and the new drug/batch of identical drugs has its new final recipient as the hospital address.



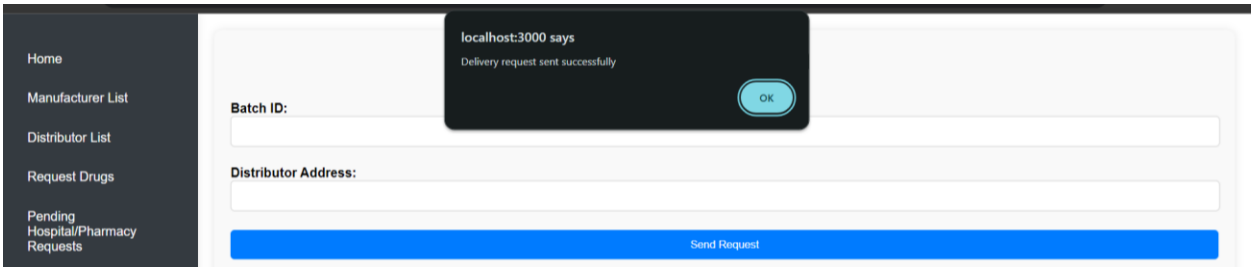
It can be seen here that the final recipient of the drug with BatchID 1 changed.

Now the wholesaler should see the distributor list form the Distributor List button so that he/she can choose a suitable distributor for the transaction of the drugs from him/her to the hospital.

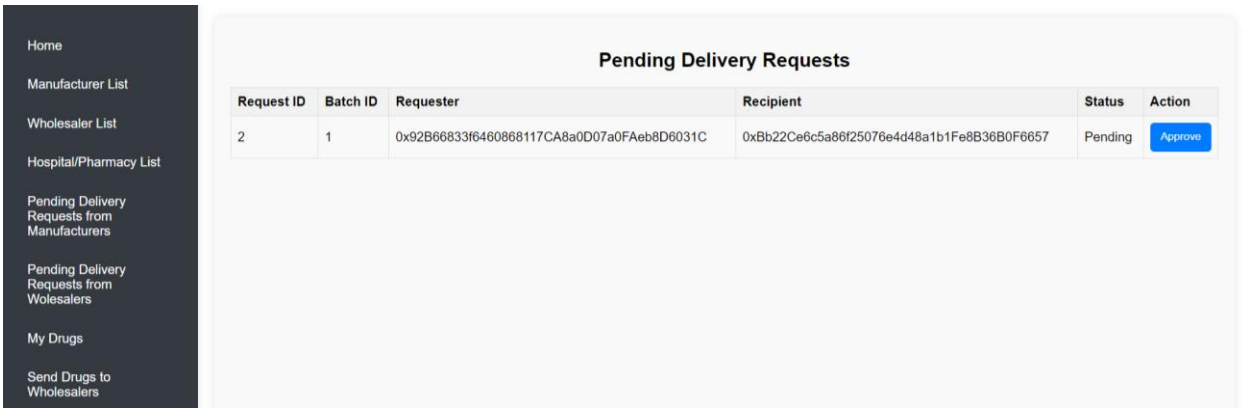




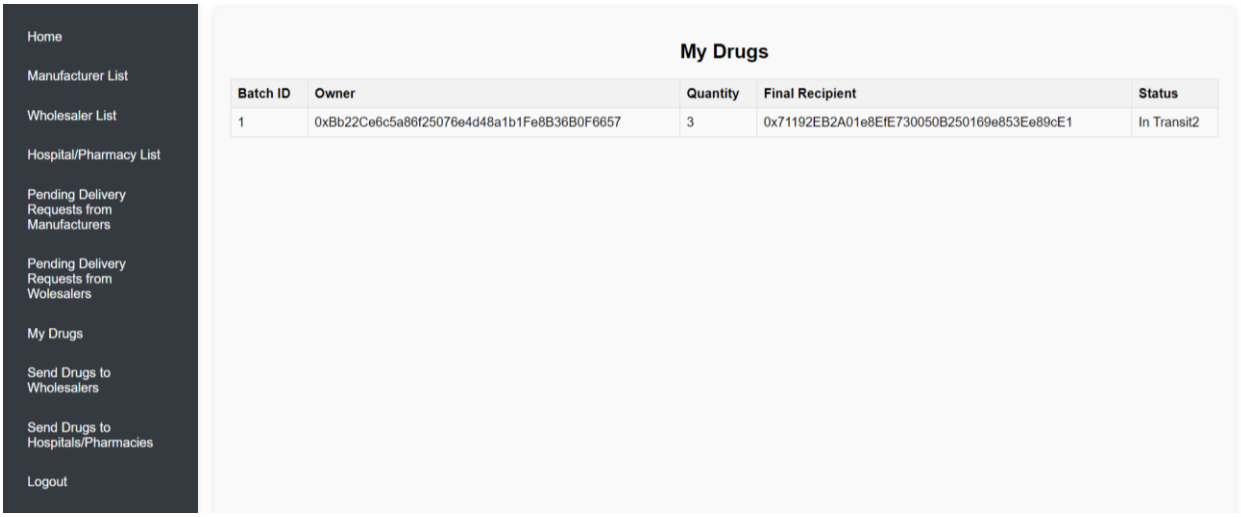
After choosing a suitable distributor the wholesaler should navigate to the Send Drugs button and send a delivery request to the distributor. This transaction also goes through metamask and an alert is generated after the manufacturer confirms the transaction though metamask.



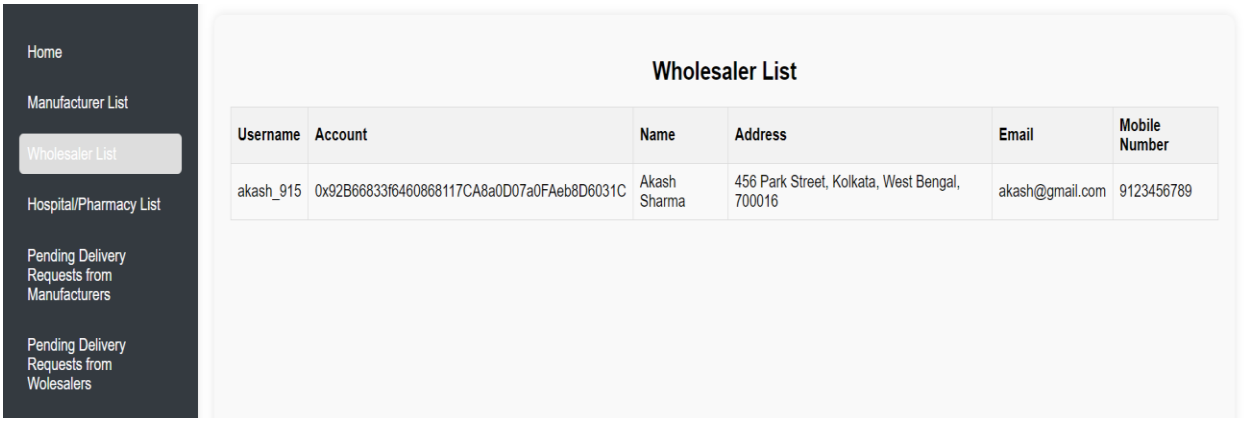
Now lets go to the distributors home page (code written in **Home3.jsx** inside components folder) to see whether he/she got the delivery request or not.

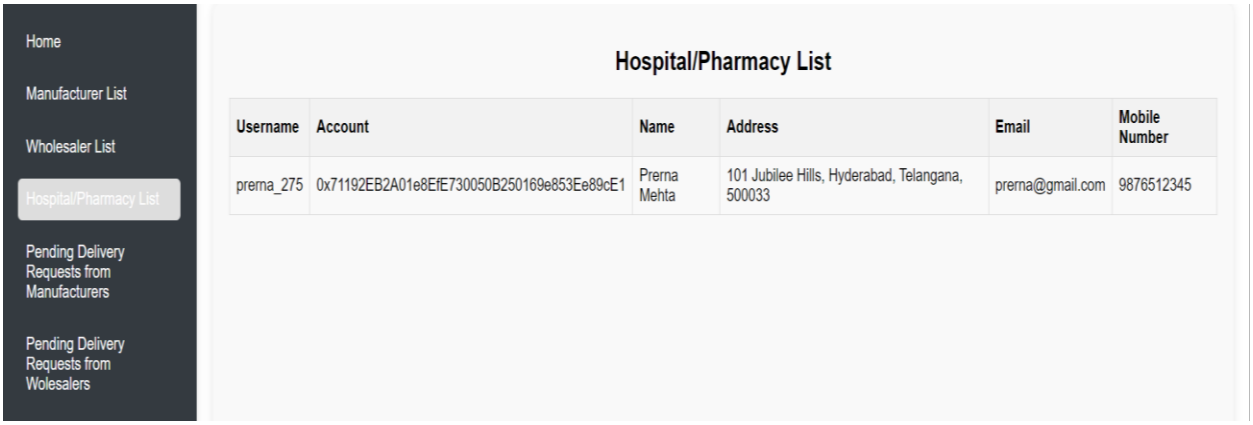


So the distributor got the delivery request which he/she need to approve so that the drug comes to him as an intermediatory stakeholder.

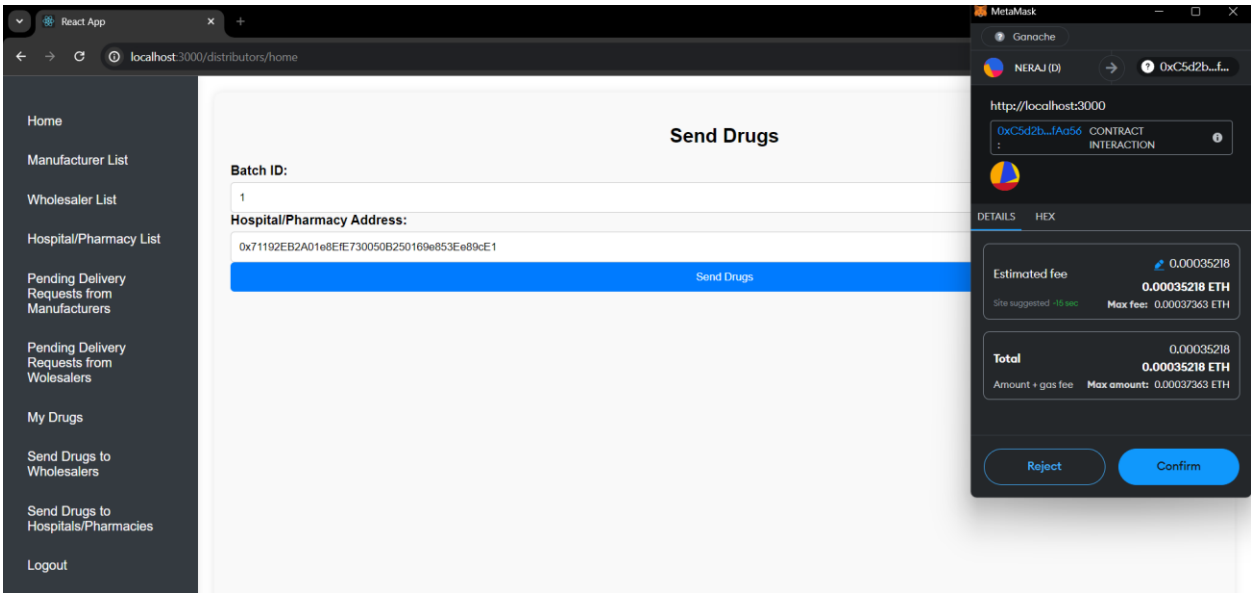


So the drugs with Batch ID 1 which we are considering in our demonstration is now having the owner as the distributor at Transit 2 state because the ownership of the drug is now passed from the wholesaler to the distributor. The distributor also gets to know about the final recipient of the drug from the table. He checks whether the final recipient is a wholesaler or hospitals/pharmacies from the Wholesaler and Hospital/Pharmacy List.

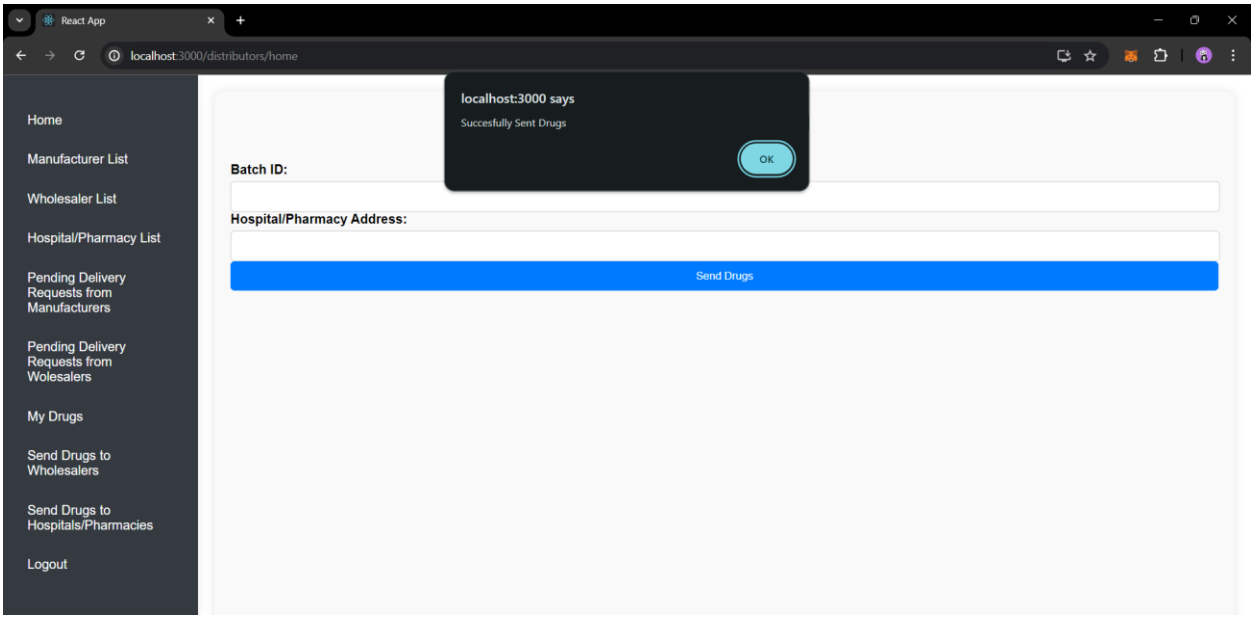




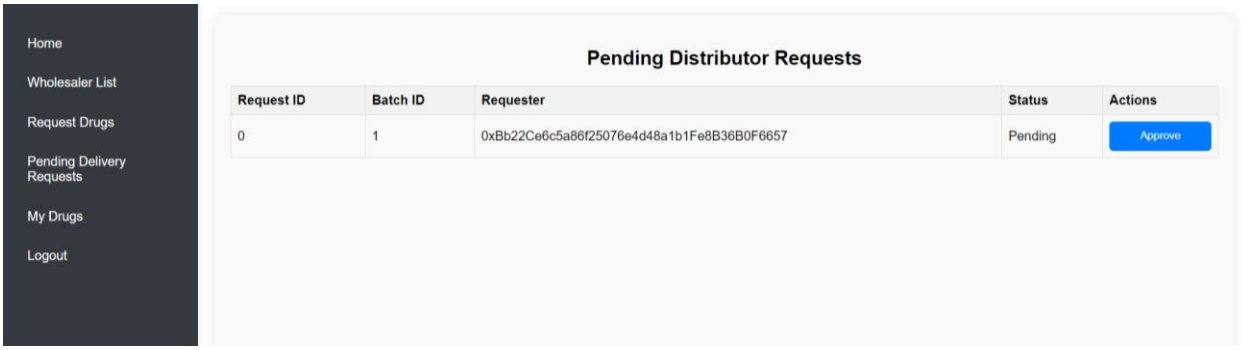
So now he proceeds for sending the drug to the appropriate recipient.



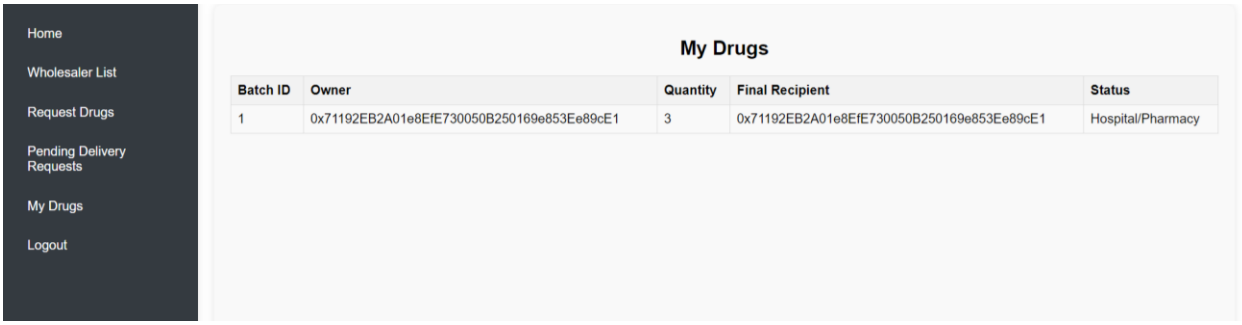
After clicking the confirm button on metamask the distributor finally sends a delivery request to the hospital.



Now let us check whether the wholesaler got the delivery request or not from the distributor.



So, the pending distributor requests are successfully fetched by the hospital and on approving this request via metamask the hospital will finally get the drug/batch of drugs from the wholesaler via a distributor.



So, the hospital finally got the batch of drugs (BatchID=1 which we are dealing with) from the wholesaler via a distributor. We can see that the owner and the final recipient of the drugs are same. So, we can conclude that the drugs reached their final destination.

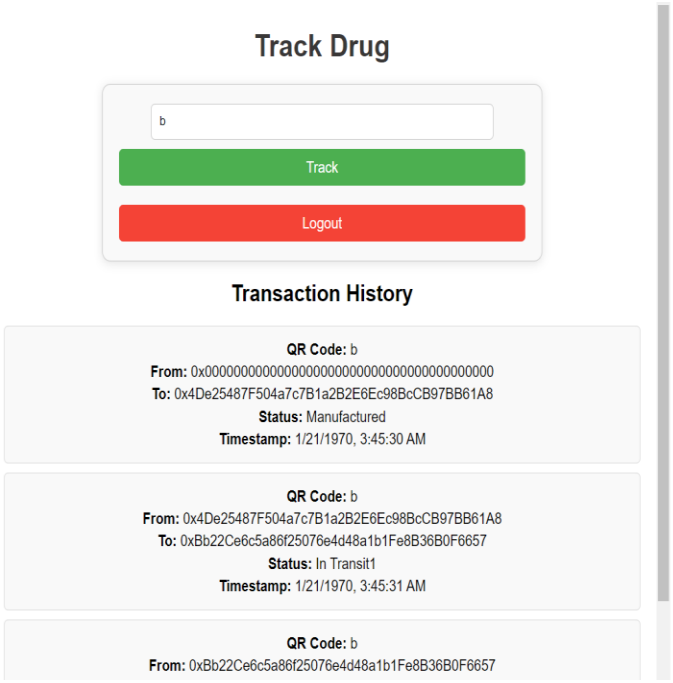
So that’s all about the second phase of our supply chain where we successfully sent the drugs to hospitals from wholesalers based on their request via a distributor.

So that’s all about the whole pharmaceutical supply chain which we are dealing with. It leverages transparent and secure blockchain technology, along with QR codes and cryptographic key verifications, to ensure user satisfaction and verification at each step of the process. Now, let's take a closer look at how we use QR codes to track drugs and verify temperature and humidity data using IoT through our Drugs smart contract.

6) **QR CODE TRACKING**

In our pharmaceutical supply chain prototype we used qr code verification for each drugs so that we can track each and every drugs inside the supply chain. We used the **Track.jsx** file inside components folder to bring out the functionality of the qr code tracking.

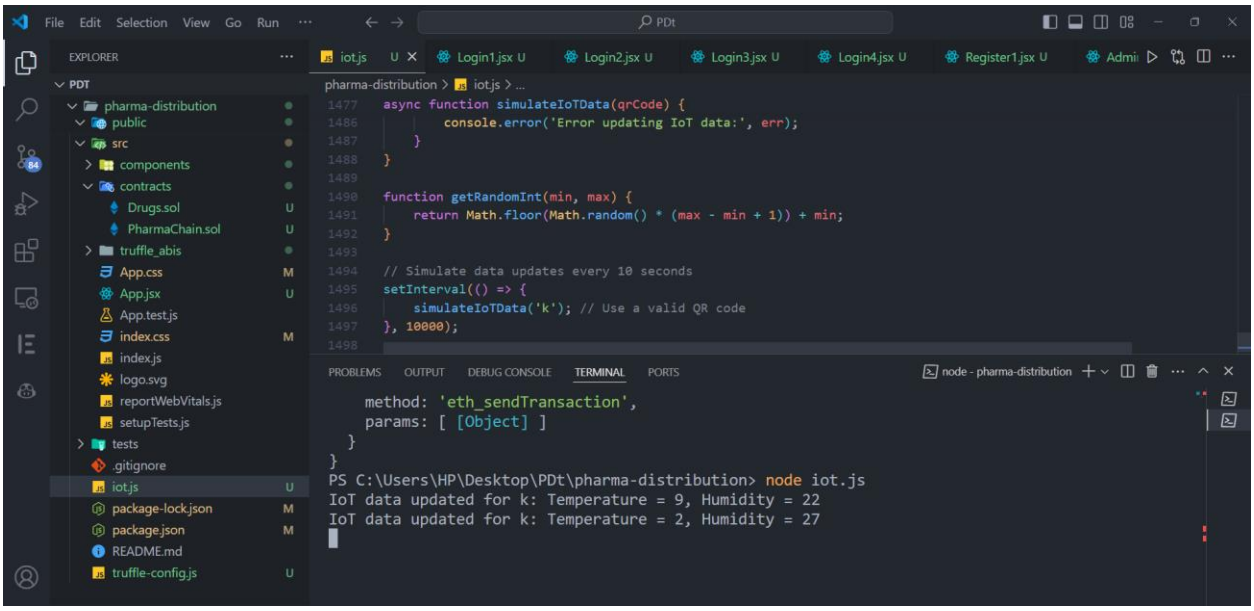
To track a drug with its qr code one should go to the Track Drugs route which is present on the landing page. On clicking the Track Drugs button the person will be redirected to a page asking for a qr code and when the person will give the qr code and hit the track button the list of all transactions of the drug with that qr code will be rendered on the screen. This will help stakeholders verify the status of the drug at each step of the supply chain.



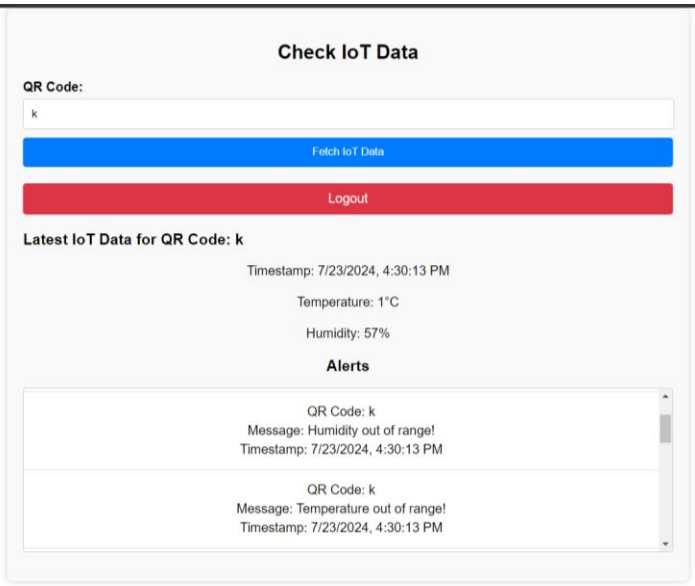
6) **TEMPERATURE/HUMIDITY TRACKING**

In our pharmaceutical supply chain prototype we used IOT script verification for each drugs so that we can track each and every drugs inside the supply chain and can create a alert message if the drug goes out of its recommended temperature and humidity. We used the **Iot.jsx** file inside components folder to bring out the functionality of the temperature/humidity tracking.

To track a drug with its qr code one should go to the Track Temperature/Humidity route which is present on the landing page. On clicking the Track Temperature/Humidity button the person will be redirected to a page asking for a qr code and when the person will give the qr code and hit the Fetch Iot Data button the latest temperature/humidity data along with list of all alerts will be rendered on the screen. This will help stakeholders verify the status of the drug at each step of the supply chain.



We used the iot.js script to auto-generate random temperature and humidity data for demonstration purposes. This script creates an interface to monitor these conditions, simulating how real-world applications could operate. In a practical scenario, this script would be replaced by actual IoT sensors, hardware, and oracle technologies to seamlessly integrate off-chain data with the blockchain.



So that's all about our decentralized pharmaceutical supply chain.

In conclusion, we want to mention that our pharmaceutical supply chain management project combines blockchain technology, IoT integration, and QR code verification to ensure a secure and transparent supply chain. By leveraging these advanced technologies, we provide enhanced tracking, verification, and user satisfaction throughout the entire supply chain process. Our innovative approach not only improves efficiency but also ensures the integrity and authenticity of pharmaceutical products.

Thank you for your attention and support.

Team Name: DragonFly

Team Members:

1. Bibhab Dasgupta
2. Riddhi Mandal

THANK YOU!