**STUDY OF EXISTING LAND REGISTRY SYSTEM**

**Introduction**

Land registry in India is something that is handled by the individual authority of each state. In India, each and every state has their respective land and property registration authority.

As per Registration Act, 1908 The State Government may appoint Registrars of the several districts, and Sub-Registrar of the several subdistricts. The State Government shall establish in every district the office of the Registrar and in every sub-district an office of the Sub-Registrar or the offices of the Joint Sub-Registrars.

**Existing system of Registration of Land**

| **Standard Operating Procedure for Applicants** | |
| --- | --- |
| **Mandatory supporting  documents required** | 1. Document on proper Stamp Paper, prepared either by a licensed  document writer/advocate or by Claimant/Executant 2. Copy of the deed in filing sheet issued by the Department (to be kept as  record with Registration department) 3. Copy of the deed to be attached with the application for mutation of  transaction 4. Form IB (to be a part of the document and its copy) if building is also  transferred 5. Building valuation certificate from approved agency under section 28B or 28C of the Stamp Act as the case maybe. 6. Form I (Rule 3 of Prevention of Undervaluation Rules) 7. Application for TRR (Mutation) (To be forwarded to the Revenue  Department through online) 8. Form No 58 – Declaration regarding Excess land – To be signed by both  the parties 9. Form 60 (IT) – If any of the parties do not hold PAN card and the  transaction amount exceeds 10 Lakhs 10. NOC issued by the Collector, in case the property is restricted from transacting. 11. Required Registration fees 12. Original/Copies of previous Title Deeds 13. Identity Proof |
| **Procedure for Fees  payment** | **Stamp Duty payment**  Stamp duty less than Rupees One Lakh ‐  Stamp paper can be obtained from  Licensed Stamp Vendors ;   Stamp Duty of Rs. One Lakh and above – e‐Stamping mode  **Registration Fee payment** –  e‐payment/ ePos at Sub Registry Office |
| **Process description** | **Step 1: Preparation of sale deed**  Option 1‐Preparation of deed on proper stamp paper with the help of  a Licensed Document Writer/Advocate by paying his  charges.  Option 2–Download model deed from website and prepare the deed  with necessary modification by the parties to the  transaction and prepare the final document signed by  Executant /Claimant on proper stamp paper (Model deeds  can be downloaded from [www.keralaregistration.gov.in/pearlpublic](http://www.keralaregistration.gov.in/pearlpublic))    (Stamp  duty  according  to  the  value  of  transaction  or  fair  value  of  the  property  whichever is higher is to be imposed on all documents. For verification of fair value  the party may visit www.igr.kerala.gov.in)    **Step2** :  Signing of Document by the parties and witnesses, affixing  Photographs and Thumb Impression on the document.    **Step 3:**  User Registration in the department portal.  (www.keralaregistration.gov.in/pearlpublic)    **Step 4:**  After user registration >> login to portal.    **Step 5:**  Submit details of document to be registered  1. Enter details of document like, Details of the person who will  present the document for registration, Details of Executant,  Claimant, Title Deed, Property, Stamp Papers used, witnesses  for the transaction etc. of Presentation details, Document  details, Claimant details, etc.  2. Generate e‐stamp if the  stamp duty is Rs. One Lakh and above;  3. Pay necessary registration fee through e-payment  4. Selection of available time slot (Token)  5. Submit the document details and get the acknowledgement slip.    **Step  6:**   Present  the  original  deed  with  annexures  on  the  date  and  time  allotted.  The  Executants  in  the  document  shall  also  be  present  in  the  stipulated  time  at  the  office  with  exact  Registration  Fee  required and obtain receipt for the same.    **Step  7:**  Appear before the Registering Officer along with necessary identity proof for registration process.  (Hearing  by  the  Registering Officer,  affixing  thumb  impression  in  the  proper  registers,  remitting  the  additional fee if any etc.)    **Step 8:**  Completion of registration process and collection of original deed  back after three days.    **Step 9:**  Submission of attested copy of document along with application for  mutation (TRR Application) to the Village Officer for mutation. (For  districts where online mutation process is yet to be implemented)    Time line for  completing the process  3 days for Property Registration  7 days for the completion of mutation process |

| **Standard Operating Procedure for Department Approver** | |
| --- | --- |
| **Process description** | **Step 1:**  Sub Registrar – Receives the Document presented for registration and forwards it to the clerk for verification.  **Step 2:**  Clerk – Verifies the Copy of the deed and online data with the details in the Original Deed with the copy and online data. If any mismatch  is found, the same will be brought to the notice of Registering Officer  who will reject the online data and return the document for  resubmission  **Step 3:**  Sub Registrar (To be Scenario)  After successful verification, the clerk forwards the document and connected papers to the Sub Registrar for registration. He will verify the document for proper stamp duty and registration fee and will accept the registration.  The parties to the document will be identified by the Registering Officer and will record their admission of execution to the document by capturing their photo and thumb impression electronically.  If sufficient stamp duty is already paid, the document will be registered instantly.   If shortage of SD is due to non‐compliance of Fair Value, the party will be given notice to pay the deficit within 7 days and on payment of deficit, the document will be registered.  If shortage of SD is due to other reasons, the document will be impounded sent to the Collector (District Registrar) for determining proper stamp duty and necessary orders will be issued by him. On compliance of his orders, the document will be registered.  Registered deed will be forwarded to the Clerk for transcription.    **Step 4:**  Clerk– Transcribes the endorsement recorded on the document to the filing sheet and submits to the sub Registrar.    **Step 5:**  Sub Registrar – After authenticating the copy of the document, the  original deed will be issued to the party |

The steps such as pre-processing the data, payment of registration fees, selection of time slot and token generation are currently online; whereas verifying the person's identity, collecting fingerprints and photographs, transcription are offline activities.

**Reference**

[https://keralaregistration.gov.in/](https://keralaregistration.gov.in/?lang=eng)

**UNDERSTANDING OF LAND REGISTRY SYSTEM IN BLOCKCHAIN**

### **Introduction**

Land Registry is one of the use cases that involve a lot of intermediaries to put trust in the system. The existing solutions in place are out of date. Tracking who owns which pieces of property is challenging when you have thousands of land records to maintain.

It is quite common to confront discrepancies within the paperwork such as counterfeit titles, forged documents and a complete loss of the record. Such situations lead to expensive court battles between conflicted parties.

The transparent nature of blockchain can make it possible to trace how property changes hands. Blockchain’s immutable, auditable and traceable features are enticing governments around the world to implement the decentralized technology in the land registry process.

* **Purpose -** To stop the double registration and producing fake Documents for registration. Inside Attack / Traditional database related attacks by DB modification: Leads to displaying unreliable encumbrance (EC) with the modification details (modification owner name / property details).
* **Scope -** Land records data need to be accurately stored in the blockchain. The existing history of transactions on a piece of land first needs to be inserted into the blockchain. This should be verified by the local SRO(Sub register officer) and approved data will be digitally signed and stored.
* **Overview -**  we will move the existing land records into the blockchain, where the land registrations cannot be manipulated.

### **Present challenges:**

1. The Involvement of middlemen and brokers which leads to additional costs, making the entire ecosystem expensive.
2. The increasing number of fraud cases
3. Time Delays: Land Registry takes a considerably long time to complete title registrations.
4. Human error/intervention can increase the chances of errors in the land registry system.

### **Advantages of implementing Blockchain:**

1. **Accelerating the Process :** creating a digital title with a blockchain land registry platform can improve the process. With the blockchain’s potential to prove authenticity, homeowners can transfer the land ownership legitimately to the buyer without needing third-party verification.
2. **Reducing Fraud Cases :** By keeping an immutable record of transactions, blockchain can prove that you are the owner of the land title and prevent forgery of documents.
3. **Bringing Transparency with Smart Contracts:** .As soon as the registrar confirms the transfer of land title, smart contracts trigger to update ownership for a new buyer and transaction corresponding to it gets stored on the blockchain.In this way, it is always possible to trace back the history of ownership records.

### **Stakeholders involved in the Blockchain Land Registry Platform:**

1. **Buyer:** A person who buys the land and uses the platform to search the property, request access and interact with the seller and get the land title ownership.
2. **Seller:** A person who sells the land and uses the platform to manage properties and transfer land title to buyers.
3. **Authorized Govt. Department:** A person who uses the platform to manage property requests, view reports, confirm and initiate the transfer.

| **Land Registry System in Blockchain** | |
| --- | --- |
| **Process description** | **Step 1: Users register to the platform**  Users who either want to claim , sell or buy properties register to the blockchain land registry platform.  They can create the profile on the platform with details like name, government-issued ID proofs and designation. **Step 2:** **User claims his land** User claims his land by uploading necessary documents. The transaction corresponding to the seller’s action of uploading the property details is recorded on the blockchain.  **Step 3: SRO approves claim**  SRO approves the claim of the user by checking the uploaded docs.  **Step 3: Buyer makes offer on one from listed property**  A buyer interested in any specific property can make an offer on it. Sellers receive notification for property access requests. They can either deny or accept it.  Transactions corresponding to the requests made by both sellers and buyers are recorded on the blockchain to ensure authenticity and traceability.  **Step 4: Sellers approve the transfer request and land inspector gets the notification**  If the seller approves the land ownership transfer request, the Govt. Department officials get the notification to initiate the transfer of property. Smart contracts trigger to provide land documents’ access to the land inspector.  After the land inspector verifies the documents, they schedule the meeting for ownership transfer with buyer and seller. The meeting record is also added to the blockchain to solve property related disputes if they occur in the future.    **Step 5: Govt. Department official** **verifies the transaction and initiates the transfer**  Govt. Department official verifies the documents submitted by buyers and sellers and adds the authenticated records to the **blockchain land registry** platform.  Sellers and buyers sign the property ownership transfer document in front of the land inspector on the land registry platform.  The signed document gets saved in the database and transaction corresponding to it is recorded on the blockchain.The transfer is initiated and smart contracts trigger to send funds to the seller and title’s ownership to a new buyer.    **Step 6: Land Registry Document Validation and Authenticity**  In case of any disputes, any authorized party can upload the signed land registry document on the platform to check its authenticity and validate it.  If the hash generated after uploading the document is the same as that of the hash created at the time of signing the document, then the document is authenticated and no modifications have been made to the document.    To sign up to the application, buyers need to enter details like personal, financial and professional details to complete the verification process.  Buyer and seller both enter their private key to verify the identities. After verification, they digitally sign the agreement from the Govt. Department’s portal.  Land inspector captures the pictures of the seller and buyer to ensure authenticity. After getting the property document signed by the seller and purchaser, the land inspector can add the legal land record to the system. |

### **General description**

* People can verify the data by searching the details.
* Admin - (SRO)

**Registrar(SRO):**

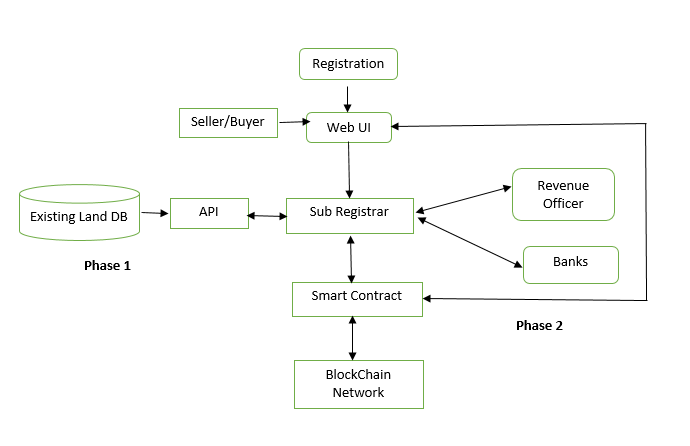
* Login:
  + Admin can login using id and password.
* Land List:
  + Add/update /view.
  + filter by state/city/District/Taluka/pin code
  + list of lands
  + Owner details past/present
  + (Verify data with block created to know if its tempered or no)
  + Sell/Transfer

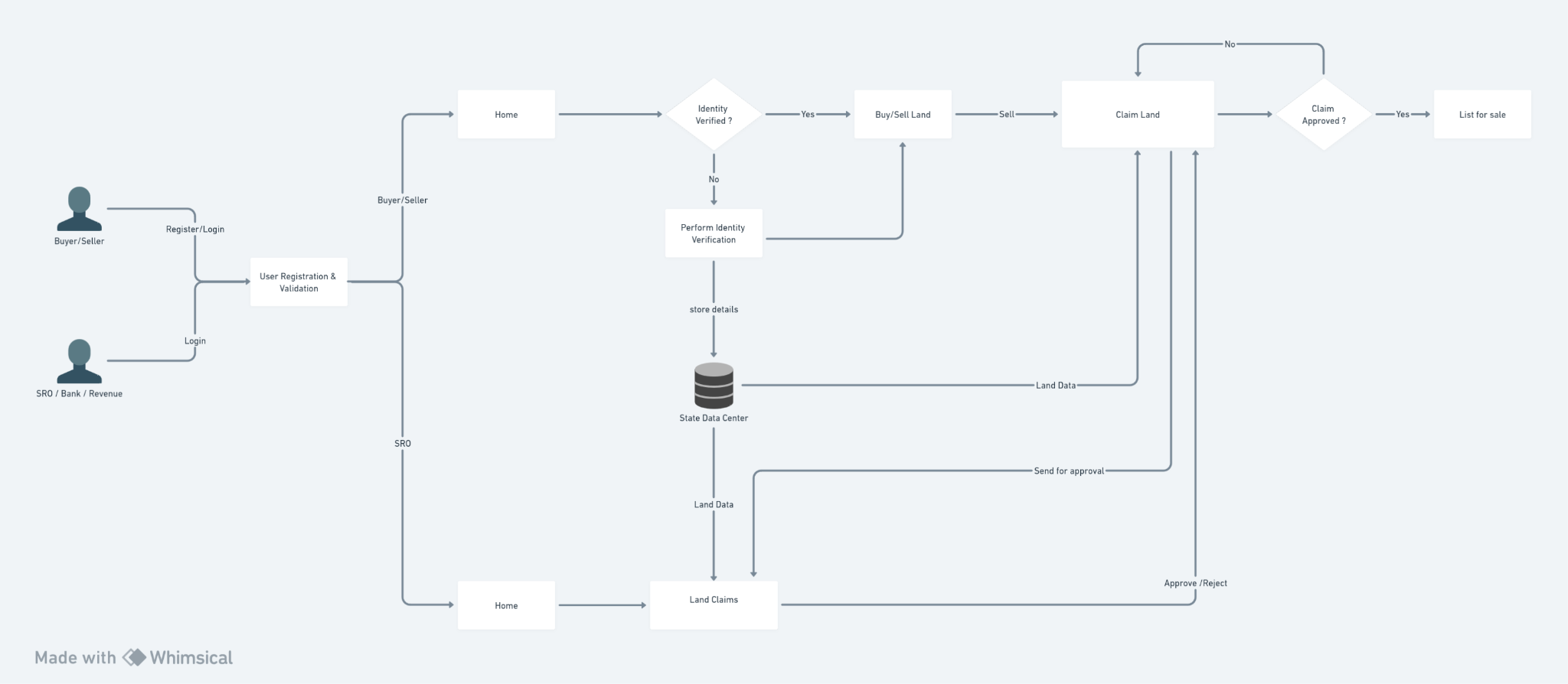
**User:**

* Register: Create an account to use the app.
* Login: User can login in his personal account using id and password.
* Profile: Users can view and update his profile.
* Claim, buy and sell land

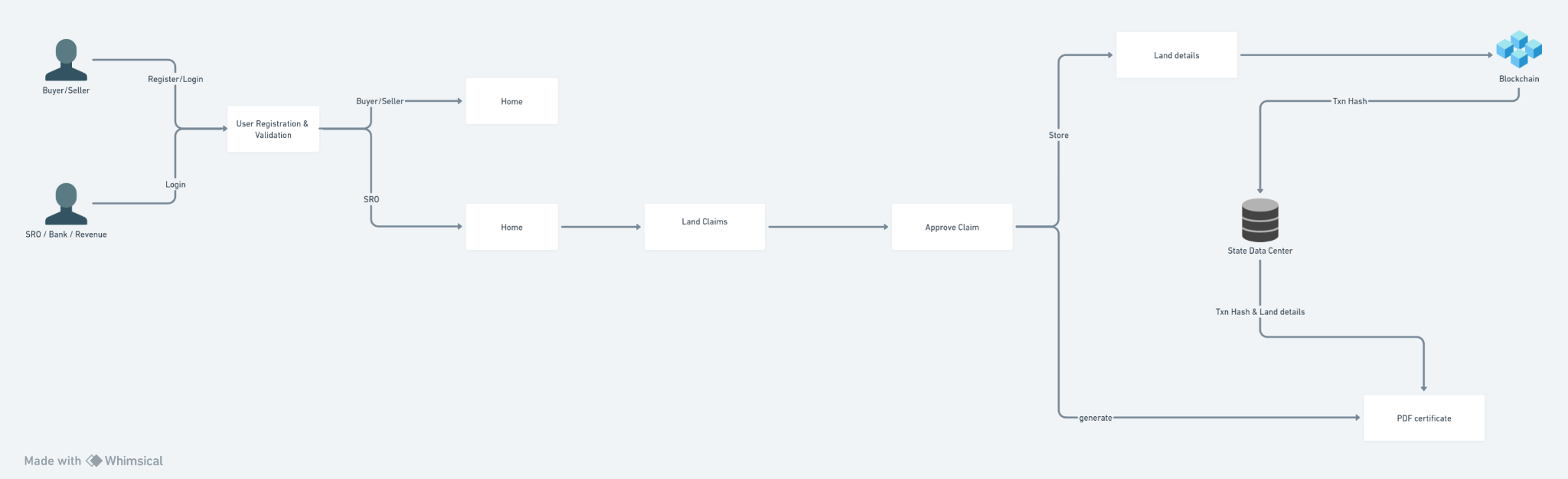
### **Tech Stack**

* Frontend - Next.js
* Backend - Node.js
* Database - MongoDB
* Blockchain - Hyperledger Fabric
* Chaincode - Javascript
* Deployment - Kubernetes & Docker

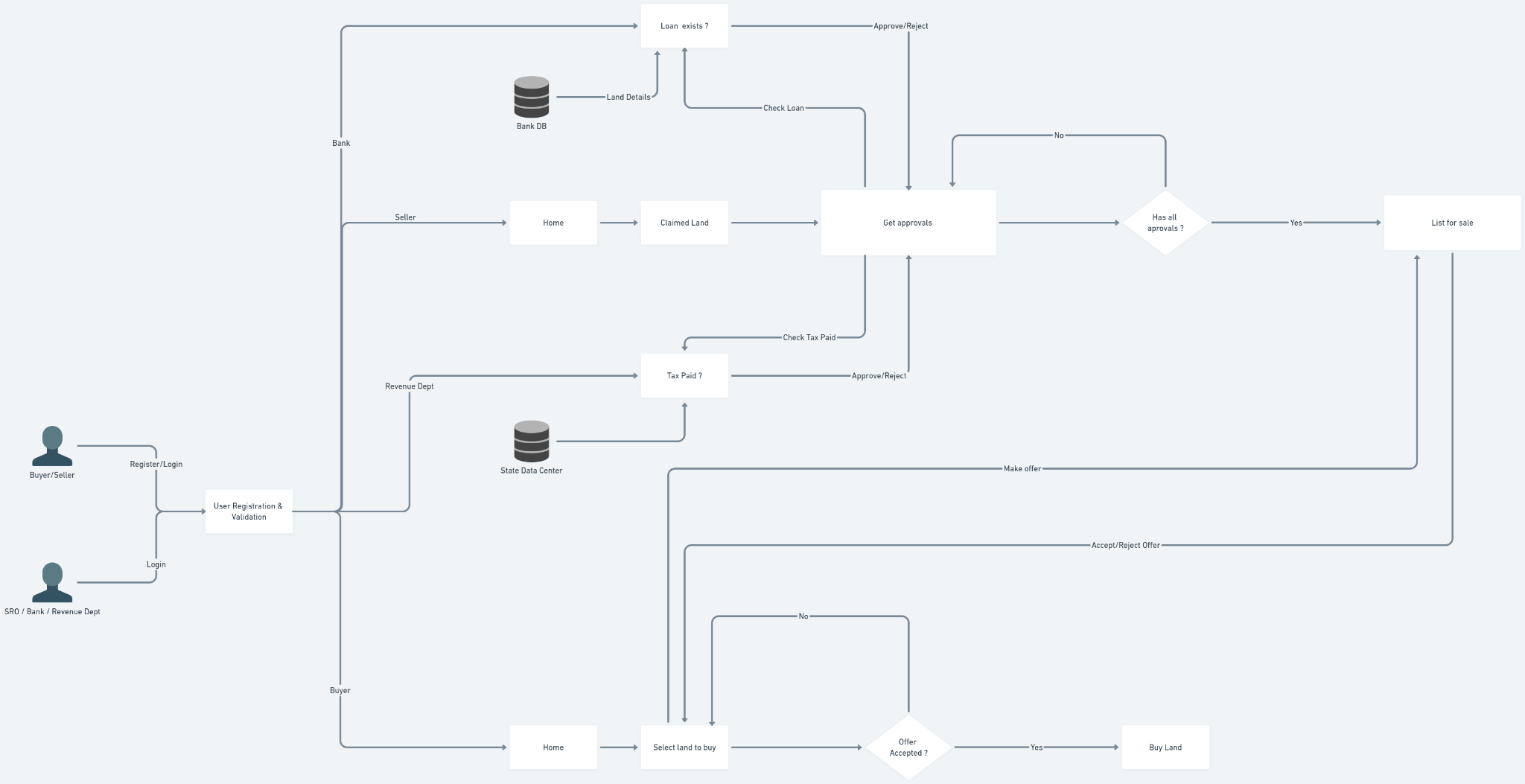




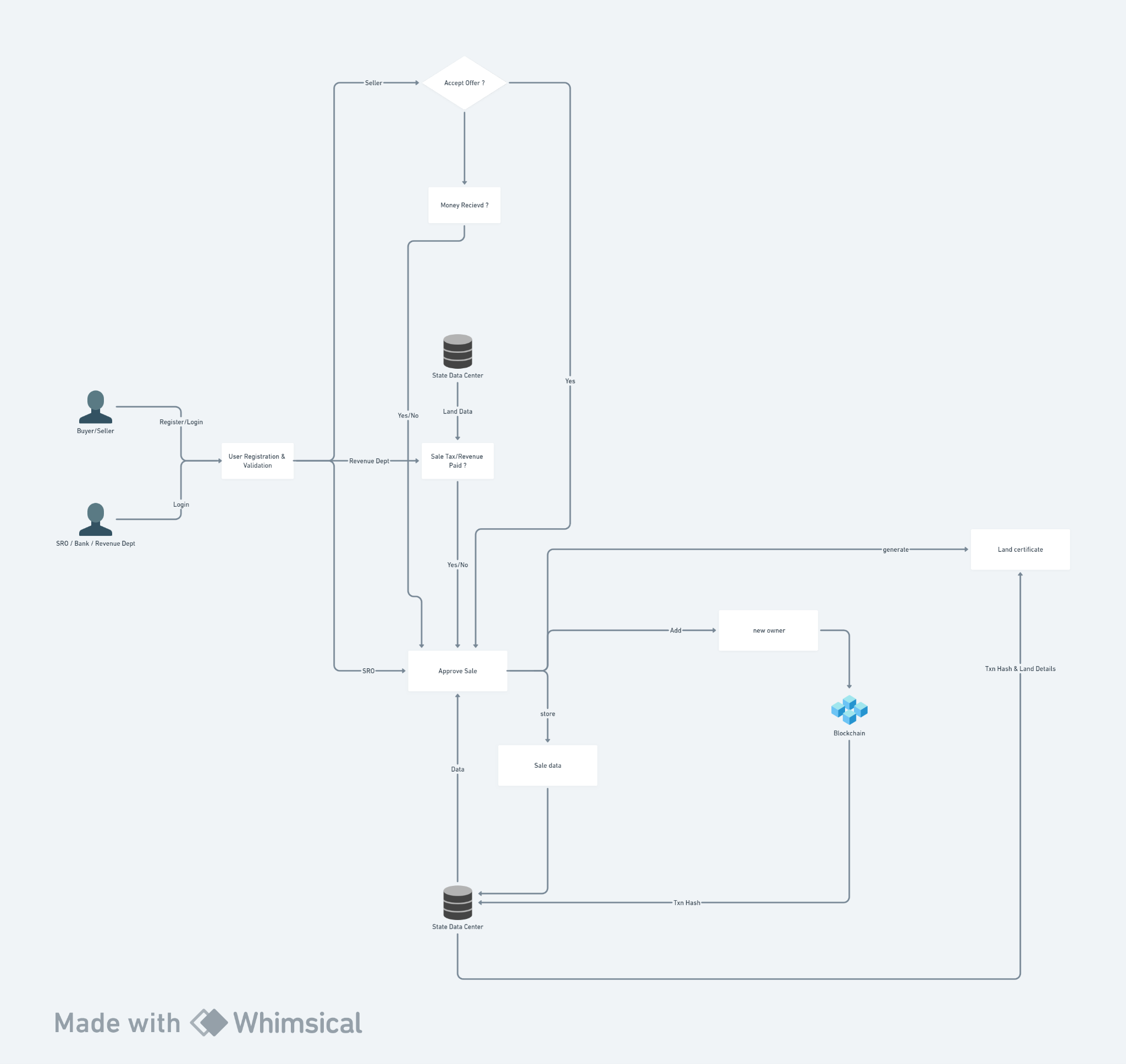
**Claim Land Workflow**

****

**Approve Claim Workflow**

****

**List Land for sale Workflow**

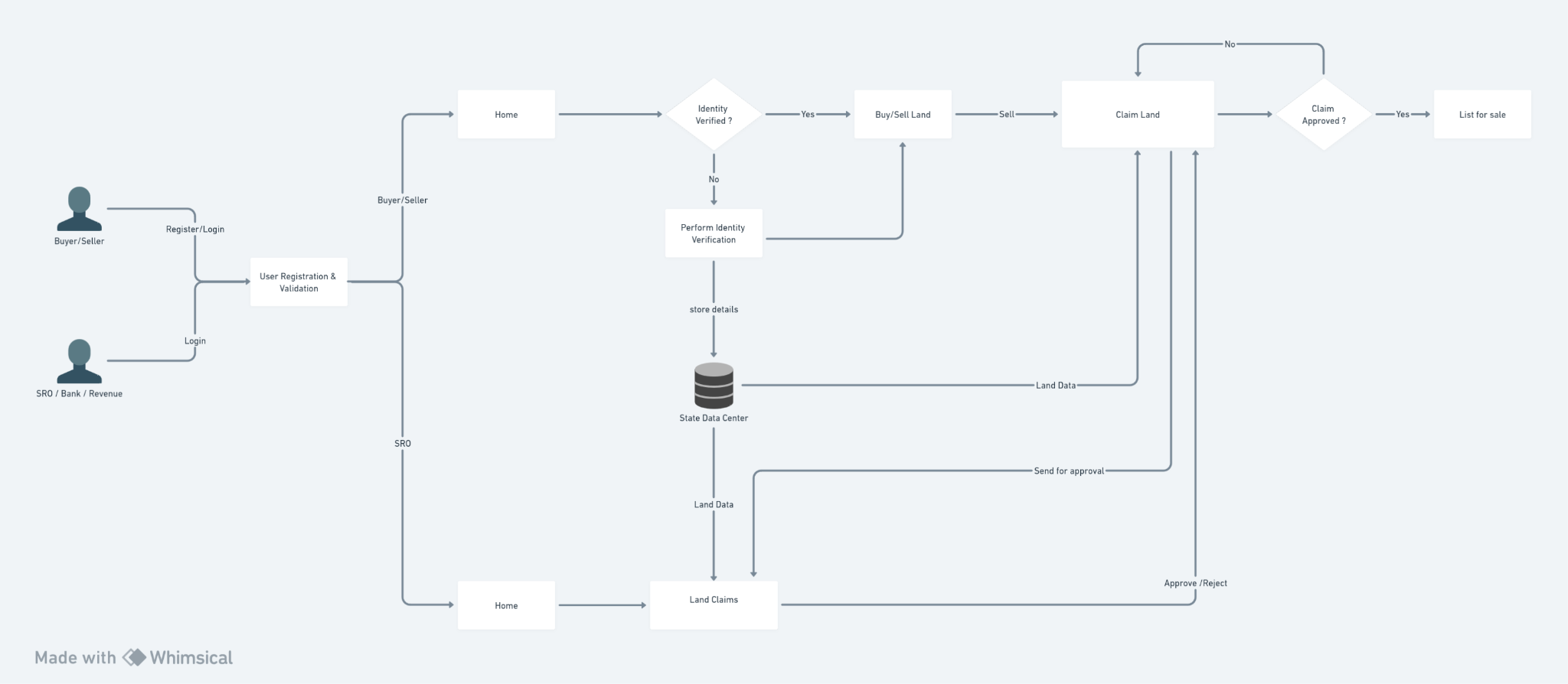
****

**Buy land Workflow**

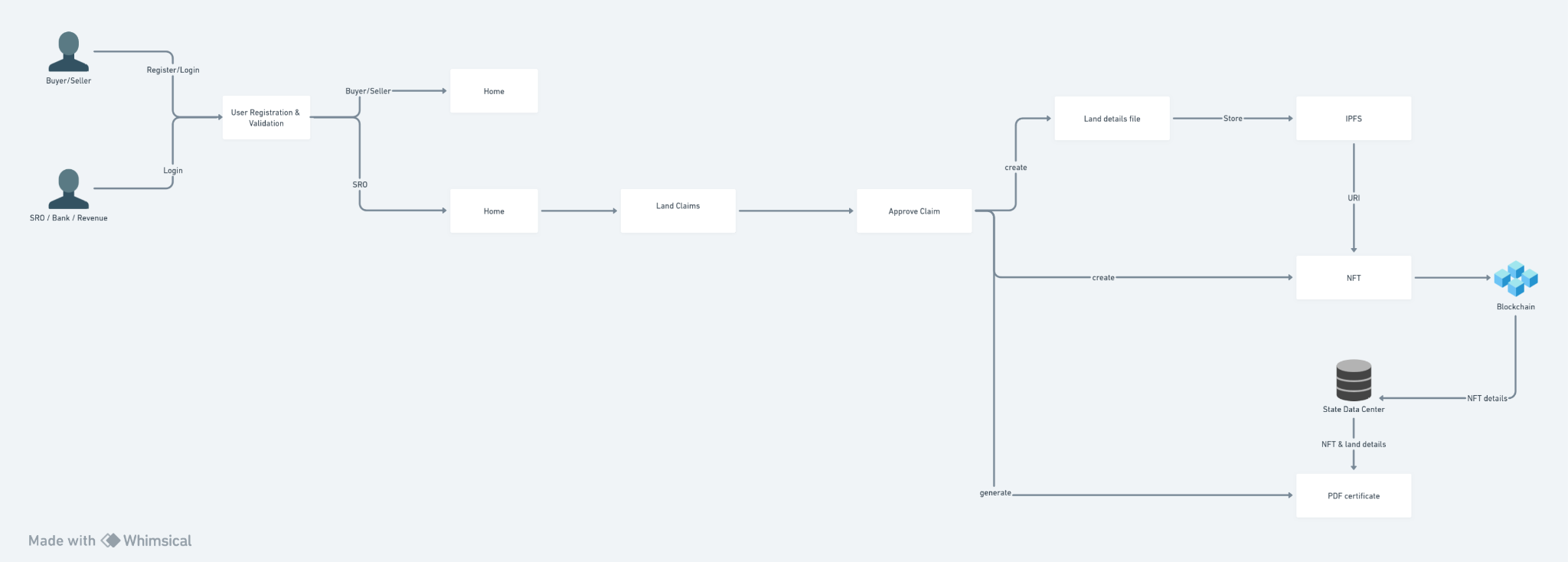
### **Using NFT’s in Hyperledger Fabric**

The application will be making use of fabric blockchain and usage of NFT concepts have been studied and some conclusions made. In public blockchains like ethereum , the land can be converted into an NFT and transferred between entities as everyone in the network possesses a wallet and an address. This is possible in fabric also , but for successful implementation with NFT’s we require that every user be registered as part of the network and every single user be provided wallet and credentials. This causes several hurdles and makes the whole process more complex and leads to storage issues also. So the usage of NFT’s have been sidelined for now.

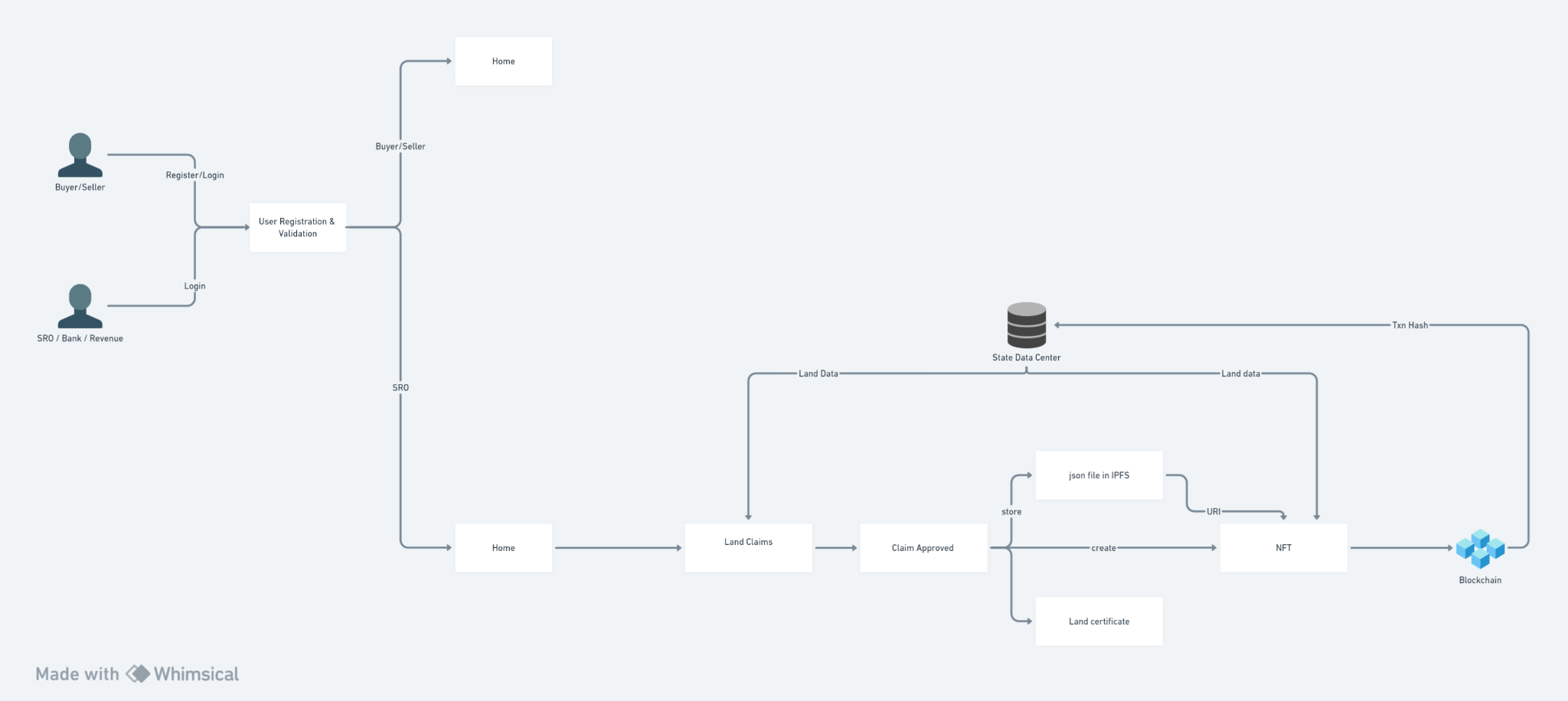
**Project implementation using NFT’s :**



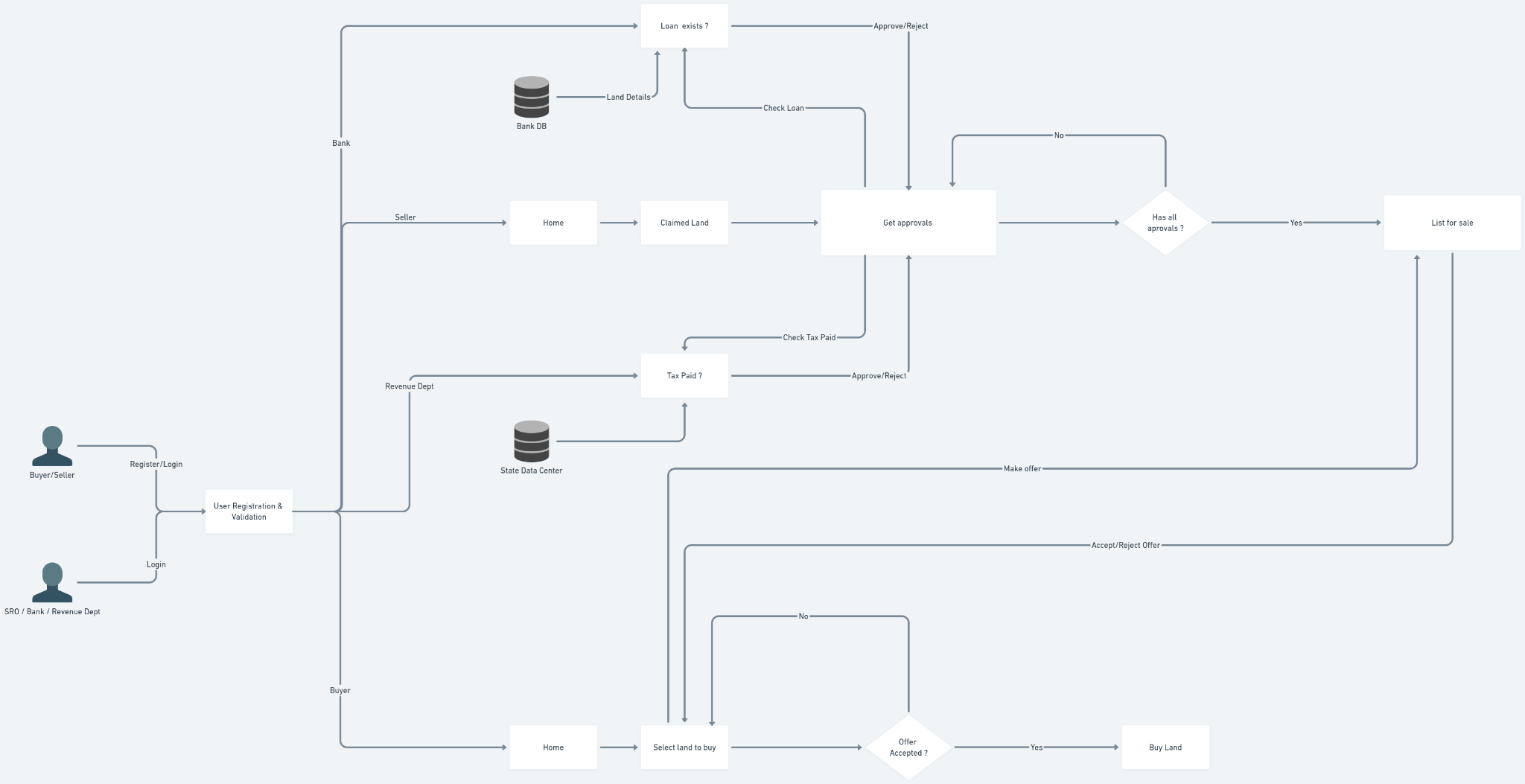
Claim land



Approve claim



Create NFT



List land



Buy land

# 

# **Steps to run the project on kubernetes**

## **Cluster Setup**

**I am using Google cloud platform (GCP). Create a cluster with 4 nodes , 2 CPU and 8GB RAM with ubuntu OS.**

Next step is to install Google Cloud SDK & Google Cloud CLI in local terminal :

Check docs- <https://cloud.google.com/sdk/docs/install-sdk>

Connect to cluster from local system

## **NFS (Network File Sharing) Server Setup**

**I am using Google cloud platform (GCP).**

**Create a VM instance with 2cpu 4gb or 8gb memory.**

**Select ubuntu machine with 50GB storage and Allow HTTP traffic and HTTPS traffic**

Go to vm instance.

In SSH -> view gcloud command -> copy command and paste in the local terminal.

Access to the VM is established from the local terminal.

OR just click on ssh button to open cloud shell in browser.

**sudo su**

**cd /**

**sudo apt update**

**sudo apt install nfs-kernel-server**

**sudo mkdir -p /mnt/nfs\_share**

**sudo chown -R nobody:nogroup /mnt/nfs\_share/**

Check if ownership updated :

**ls -la /mnt/nfs\_share**

Give executable permission to the directory :

**sudo chmod 777 /mnt/nfs\_share/**

Check the nfs server configuration file :

**cat /etc/exports**

Configure the NFS (Network File System) server by adding an entry for the nfs\_share to the /etc/exports file. This will expose the folder and \* means any ip can connect to this server folder or we can specify our custom IP there if you want.

**echo "/mnt/nfs\_share \*(rw,sync,no\_subtree\_check,insecure)" | sudo tee -a /etc/exports**

Check the nfs server configuration file , the new config will be added at end:

**cat /etc/exports**

Export the file :

**sudo exportfs -a**

Restart the nfs server :

**sudo systemctl restart nfs-kernel-server**

Now nfs server is configured and up and running.

**Now as we created the nfs server in GCP we need to expose the nfs port 2049**

Go to cloud console -> go to compute engine -> vm instances -> select nfs server vm -> press 3 dot at end -> view network details -> firewall -> create firewall rule -> add rule for TCP port 2049

Refer - <https://www.youtube.com/watch?v=-RjDWwTZUnc>

**Note : add the rule for TCP 2049 port**

Now we have to create a nfs client on our local machine. The client will be connected to the server from the host machine and this enables us to perform file operations from our loacal machine which will be reflected in our nfs server.

Install on local machine .Open a terminal

**sudo apt install nfs-common**

Create the client directory in the local machine.

**mkdir nfs\_clientshare**

Mount the nfs client to server by passing external IP address of your server :

**sudo mount -o nolock -t nfs 34.100.184.178:/mnt/nfs\_share ./nfs\_clientshare**

To Check if mount was successful :

Go to server terminal -> cd into nfs\_share -> check if any files present there

Now go to local terminal -> cd into nfs\_clientshare -> create a sample file there

Check in server terminal if the file is also created in nfs\_share folder

Github repo link -

Clone the repo

Open in VScode

## **Persistent Volume (PV) , PVC and Test Pod Setup**

In the nfs server we are setting up PV and creating a PVC and a pod to test out if the nfs server is working correctly with kubernetes objects.

**Note: in pv.yaml file replace IP with your own server IP**

Deploy files :

**kubectl apply -f 1.nfs/.**

## **Fabric CA Setup**

From local terminal

We are moving the files in the prerequisite folder of hlf-kubernetes folder to nfs\_clientshare to get it inside the nfs server.

**sudo cp -R prerequsite/\* ../nfs\_clientshare/**

In server terminal

**sudo su**

**cd /mnt/nfs\_share**

**chmod +x scripts/ -R**

**chmod 777 organizations/ -R**

Deploy ca files

**kubectl apply -f 2.ca/.**

## **Generating Certificates and Artifacts**

### Generate Certificates for peers and orderers:

**kubectl apply -f 3.certificates/.**

Wait for the job to be completed

### Creating Genesis Block and Channel Transaction :

**kubectl apply -f 4.artifacts/.**

Wait for the job to be completed

## **Creating Orderers**

Deploy files :

**kubectl apply -f 5.orderer/.**

## **Configmap for peer**

Deploy files :

**kubectl apply -f 6.configmap/.**

## **Peer Deployment , service and cli files**

Deploy files :

**kubectl apply -f 7.peers/.**

Check if all the objects are up and running on lensIDE or from the terminal.

### Creating Application Channel

Go to lensIDE or use terminal

Go to cli pod of peer0Sro ie, cli-peer0-sro

Open shell

The scripts folder will be mounted from nfs server inside our pod , we use createAppChannel.sh to execute the command for channel creation.

Run the script :

**./scripts/createAppChannel.sh**

**ls channel-artifacts/**

See if mychannel.block is present, if yes and no errors in the shell then successful.

### Joining Channel

To Join channel , Go to lensIDE

Go to cli pod of peer0Sro ie, cli-peer0-sro

Open shell

Run this command :

**peer channel join -b ./channel-artifacts/landchannel.block**

Check if joined :

**peer channel list**

**Do the same in cli pod of all peers**

### Anchor peer update

Go to lensIDE or use terminal

Go to cli pod of peer0Sro ie, cli-peer0-sro

Open shell

Run the script :

**./scripts/updateAnchorPeer.sh SroMSP**

For org 2 peer in cli pod of peer0bank:

**./scripts/updateAnchorPeer.sh BankMSP**

For org3 peer in cli pod of peer0revenue:

**./scripts/updateAnchorPeer.sh RevenueMSP**

## **Chaincode Operations**

## Packaging Chaincode

We have the chaincode already available in the peers as it's there in the nfs server.

For packaging chaincode we need 2 files , connection.json and metadata.json.

These 2 are available in chaincode/fabland/packaging/ folder.

Go to the nfs\_clientshare local folder from local terminal

The files required for chaincode are inside it.

**cd nfs\_clientshare/chaincode/fabland/packaging**

We are creating a tar file and bringing the connection.json file inside it

**sudo tar cfz code.tar.gz connection.json**

Do ls to see code.tar.gz file created

We are packaging the code.tar.gz file with metadat.json and storing it in fabland-sro.gz

**sudo tar cfz fabland-sro.tgz code.tar.gz metadata.json**

The above is for peer0 of sro

Now create package for other peers

**sudo rm code.tar.gz**

Edit the connection.json file to change address to fabland-bank:7052 for bank :

**sudo vim connection.json**

Save it and run :

**sudo tar cfz code.tar.gz connection.json**

**sudo tar cfz fabland-bank.tgz code.tar.gz metadata.json**

Do same for revenue

**sudo rm code.tar.gz**

Edit the connection.json file to change address to fabland-revenue:7052 for revenue :

**sudo vim connection.json**

Save it and run :

**sudo tar cfz code.tar.gz connection.json**

**sudo tar cfz fabland-revenue.tgz code.tar.gz metadata.json**

## Install chaincode

Go to lensIDE

Go to cli pod of peer0Sro ie, cli-peer0-sro

Open shell

**cd /opt/gopath/src/github.com/chaincode/fabland/packaging**

**ls**

We can see chaincode there

Install chaincode :

**peer lifecycle chaincode install fabland-sro.tgz**

Copy the package identifier and store it in a file for later use

Example - fabland:8f3330e90a76faec707dcdddc598067b64121e3b2ab767242495e2183db9a92c

Go to cli pod of peer0bank ie, cli-peer0-bank

Open shell

**cd /opt/gopath/src/github.com/chaincode/fabland/packaging**

Install chaincode :

**peer lifecycle chaincode install fabland-bank.tgz**

Copy the package identifier and store it in a file for later use

Example - fabland:cd8a488154bbe7fb0c2f08f7a01db7ff433bda2d1caa00bb52a10828ba39ae92

Go to cli pod of peer0revenue ie, cli-peer0-revenue

Open shell

**cd /opt/gopath/src/github.com/chaincode/basic/packaging**

Install chaincode :

**peer lifecycle chaincode install fabland-revenue.tgz**

Copy the package identifier and store it in a file for later use

Example - fabland:795425d30ddde9fce106626fdf96ca47a3e3b6c76ee5534cc556634c24d8f506

## **Create Chaincode deployment and services**

**\*\*We have to containerize our chaincode and push it to dockerhub to use it in our deployment files.**

**\*\*Also we pass the chaincode ID ie , package identifier we received from the previous packaging step in the deployment files**

Deploy files :

**kubectl apply -f 9.cc-deploy/fabland/.**

## Approve Chaincode

Go to lensIDE

Go to cli pod of peer0Sro ie, cli-peer0-sro

Open shell

Run the approve command :

**Note :** **Substitute with your package id**

**peer lifecycle chaincode approveformyorg --channelID landchannel --name fabland --version 1.0 --package-id fabland:2cc3affe748a554ed496414d600e74f96fb1d00f25df7378b935e8c83ef48a36 --sequence 1 -o orderer:7050 --tls --cafile $ORDERER\_CA**

Go to cli pod of peer0Bank ie, cli-peer0-bank

Open shell

Run the approve command :

**Note :** **Substitute with your package id**

**peer lifecycle chaincode approveformyorg --channelID landchannel --name fabland --version 1.0 --package-id fabland:5135c39e5ccea99d71d968c7870adf1e700e36e88360f6e3f3bf2a77d053c969 --sequence 1 -o orderer:7050 --tls --cafile $ORDERER\_CA**

Go to cli pod of peer0Revemue ,ie, cli-peer0-revenue

Open shell

Run the approve command :

**Note :** **Substitute with your package id**

**peer lifecycle chaincode approveformyorg --channelID landchannel --name fabland --version 1.0 --package-id fabland:7d4c77810d67e7fae3241d9af189a10e13fb25294953c32de206e4e918fa2f9e --sequence 1 -o orderer:7050 --tls --cafile $ORDERER\_CA**

## Check commit readiness

Check commit readiness from any of the cli :

**peer lifecycle chaincode checkcommitreadiness --channelID landchannel --name fabland --version 1.0 --sequence 1 -o -orderer:7050 --tls --cafile $ORDERER\_CA**

## Commit Chaincode

Commit from any of the cli :

**peer lifecycle chaincode commit -o orderer:7050 --channelID landchannel --name fabland --version 1.0 --sequence 1 --tls true --cafile $ORDERER\_CA --peerAddresses peer0-sro:7051 --tlsRootCertFiles /organizations/peerOrganizations/sro.land.com/peers/peer0.sro.land.com/tls/ca.crt --peerAddresses peer0-bank:7051 --tlsRootCertFiles /organizations/peerOrganizations/bank.land.com/peers/peer0.bank.land.com/tls/ca.crt --peerAddresses peer0-revenue:7051 --tlsRootCertFiles /organizations/peerOrganizations/revenue.land.com/peers/peer0.revenue.land.com/tls/ca.crt**

**peer lifecycle chaincode querycommitted --channelID landchannel**

## Invoke & Query

Do from any of the cli :

Invoke :

**peer chaincode invoke -o orderer:7050 --tls true --cafile $ORDERER\_CA -C landchannel -n fabland --peerAddresses peer0-sro:7051 --tlsRootCertFiles /organizations/peerOrganizations/sro.land.com/peers/peer0.sro.land.com/tls/ca.crt --peerAddresses peer0-bank:7051 --tlsRootCertFiles /organizations/peerOrganizations/bank.land.com/peers/peer0.bank.land.com/tls/ca.crt --peerAddresses peer0-revenue:7051 --tlsRootCertFiles /organizations/peerOrganizations/revenue.land.com/peers/peer0.revenue.land.com/tls/ca.crt -c '{"Args":["createLand","land1","Kozhikode","Koduvalli","Koduvalli","Koodaranhi","1","1","2","10","John","sea","jack","wall","no remarks","Abhith","2","1"]}' --waitForEvent**

Query :

**peer chaincode query -C landchannel -n fabland -c '{"Args":["readLand","land1"]}'**

# **Accessing CouchDB**

We need to do port forwarding to access the couchDb i.e., expose the port of couchDB service.

The couchDB container is running inside the pods for peer containers. From the service file for the peer pod in lensIDE we can see the couchDB is running on port 5984.

In local terminal :

**kubectl port-forward services/peer0-sro 5984:5984**

In the above command the first 5984 is the port on the container and next one is the port we expose on the local machine.

Do not close the running terminal

Go to on browser - <http://127.0.0.1:5984/_utils>

This will open up couchDB in the browser. Similarly can be done for other peers' couchdbs if needed.

# **Backend**

Create a docker file for the backend. Containerize , push to docker hub and we use the image in a deployment file.

Mention the image in 10.api/api.yaml file.

Deploy files :

**kubectl apply -f 10.api/.**

Running the backend on local-system :

**kubectl port-forward <backend pod name> 4000:4000**

# **Frontend**

Create a docker file for the frontend. Containerize , push to docker hub and we use the image in a deployment file.

Mention the image in the 11.frontend/frontend.yaml file.

Deploy files :

**kubectl apply -f 11.frontend/.**

Running the backend on local-system :

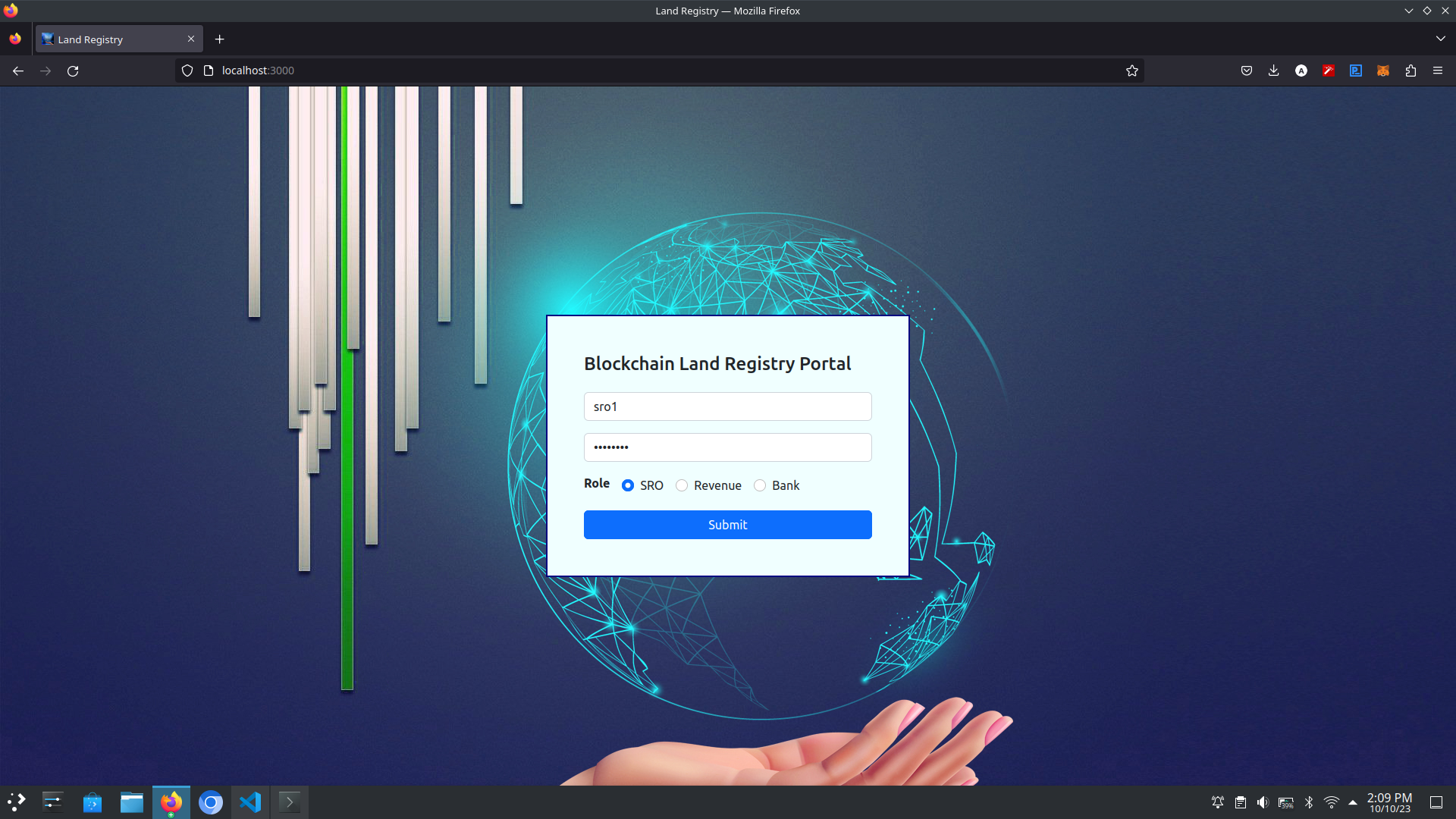
**kubectl port-forward <backend pod name> 3000:3000**

**Access application from** [**http://localhost:3000**](http://localhost:3000)

# 

# **Images of the Application**

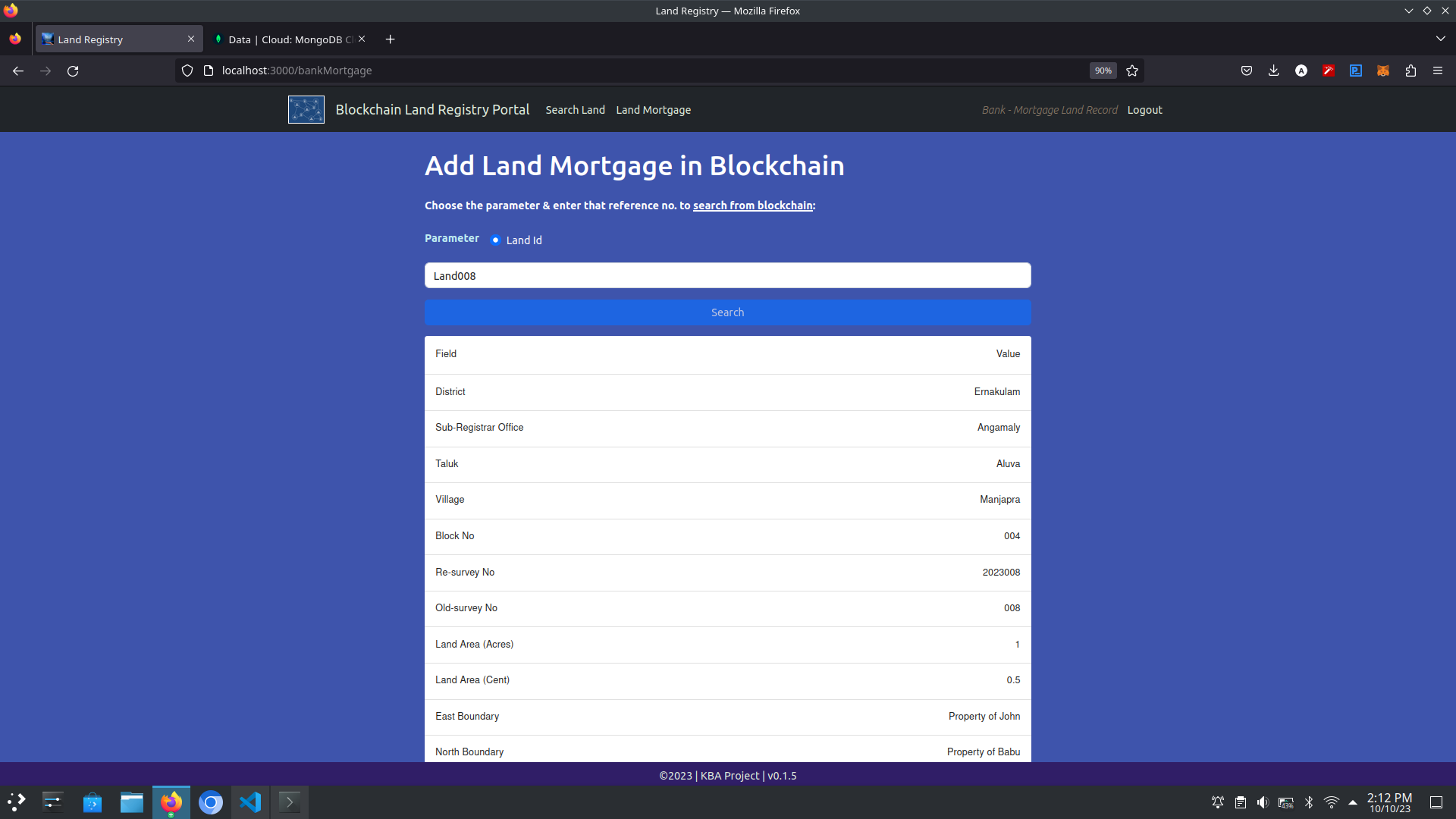
## **Login :**



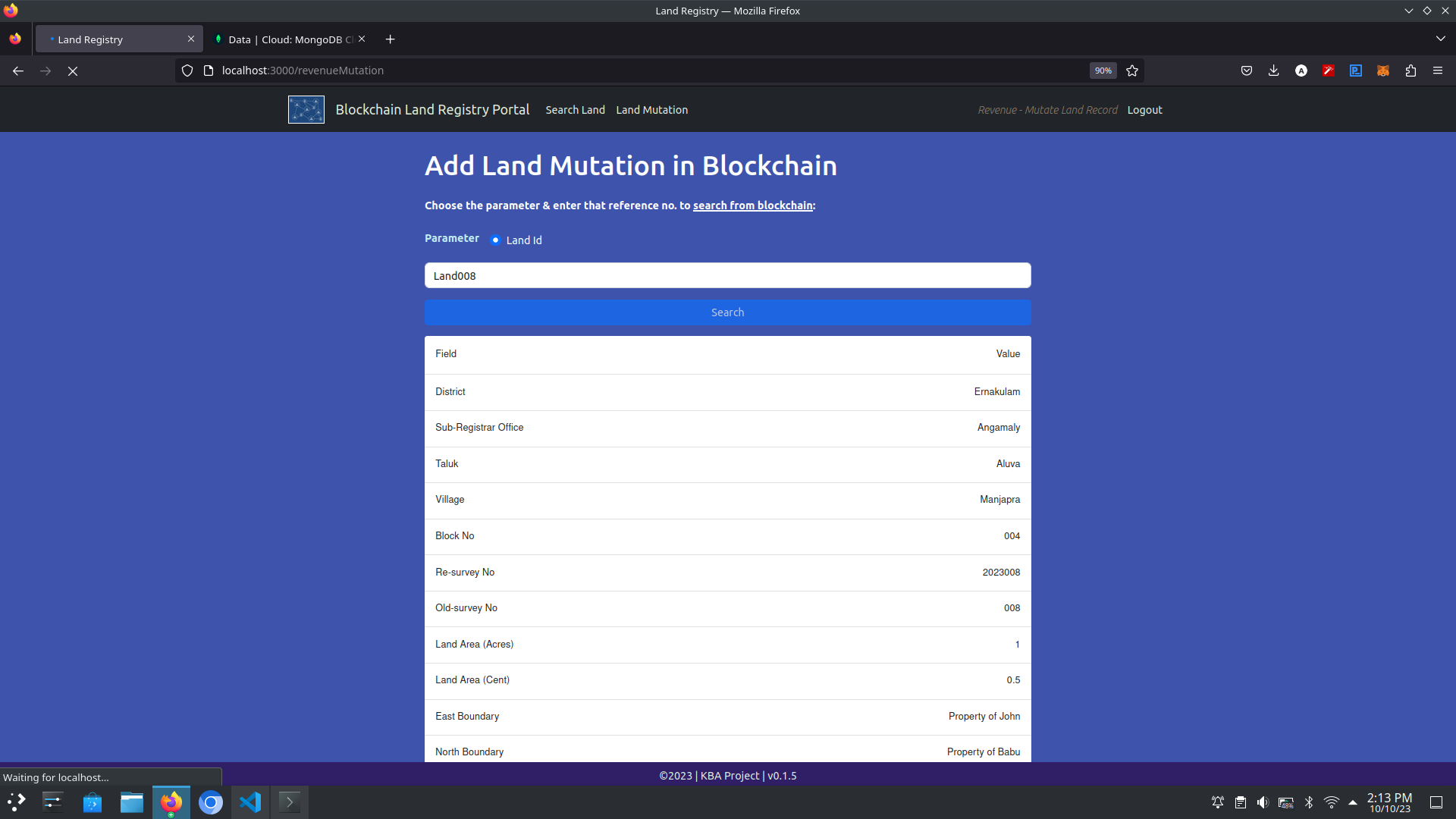
## **Add Land to blockchain :**



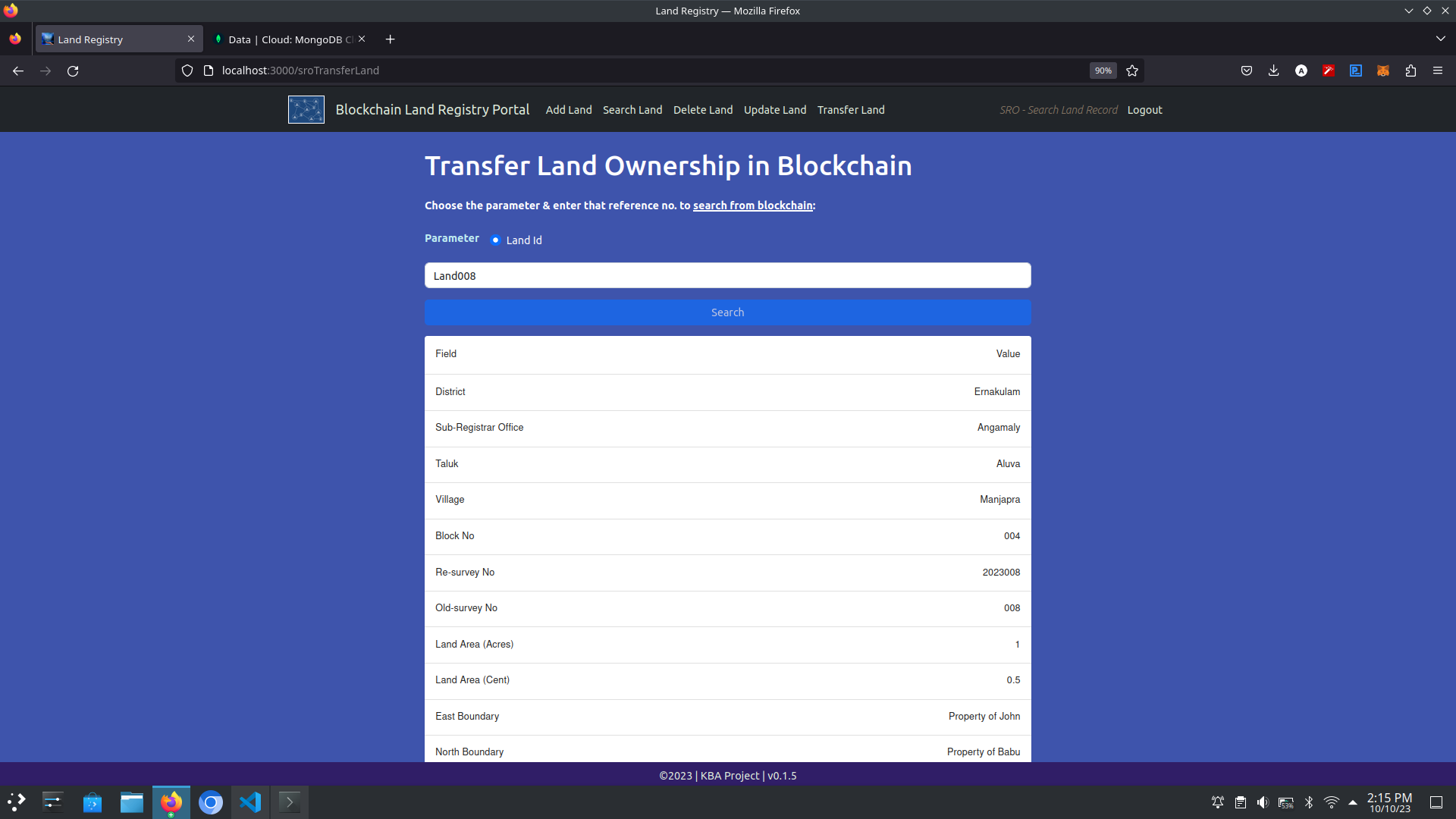
## **Check if land is mortgaged :**

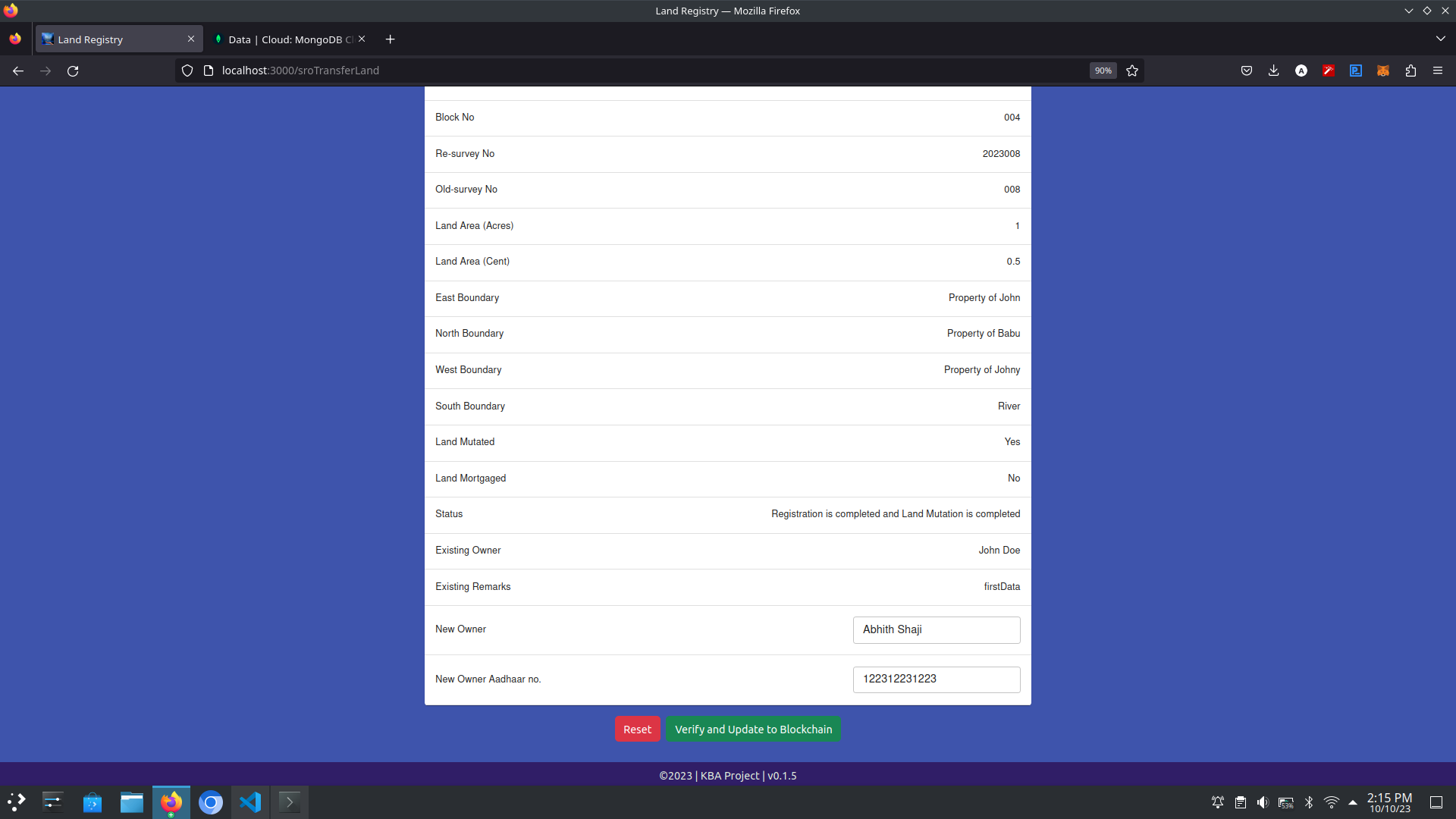


## **Approve or reject land mutation :**

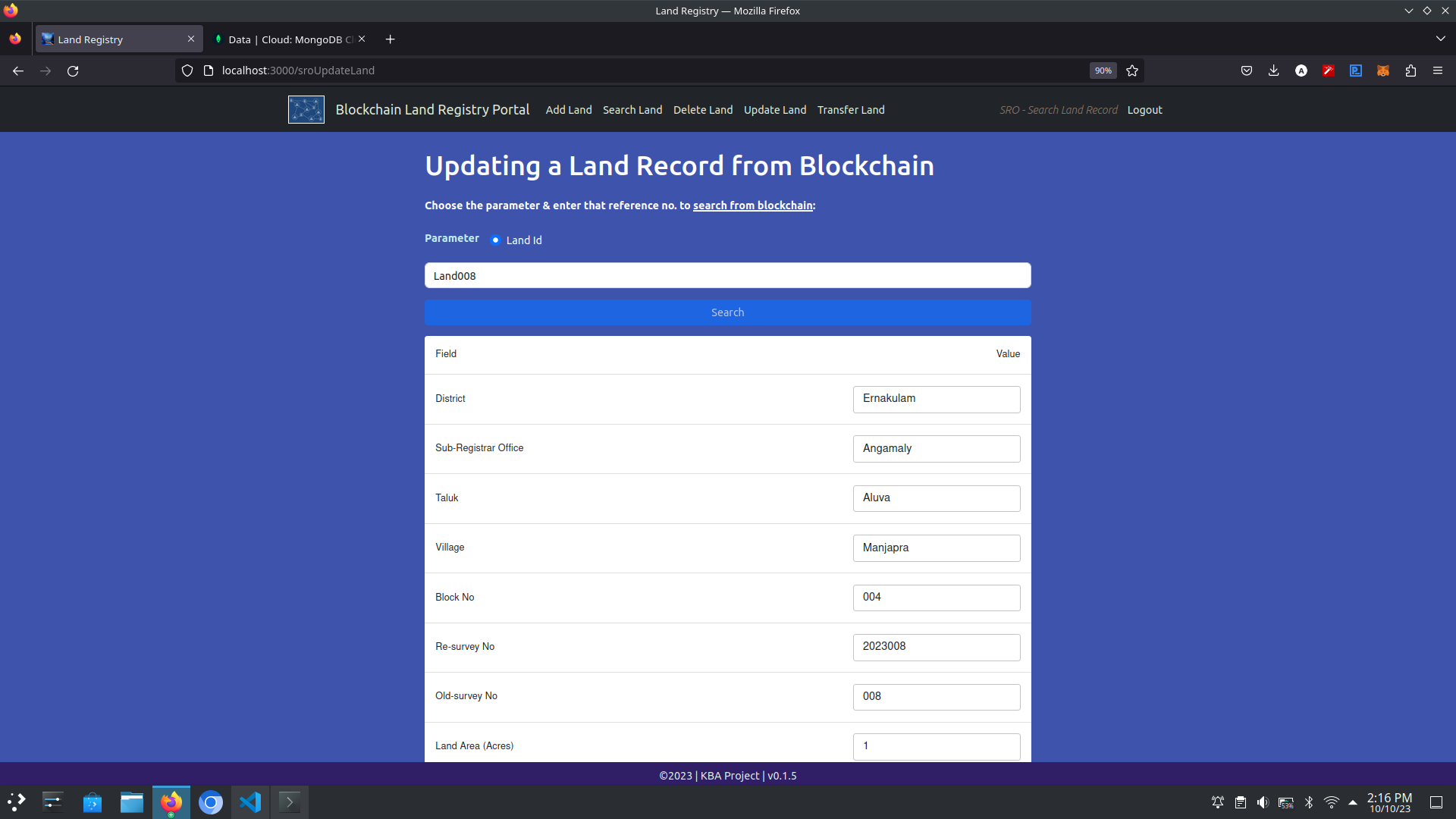


## **Transfer land ownership :**

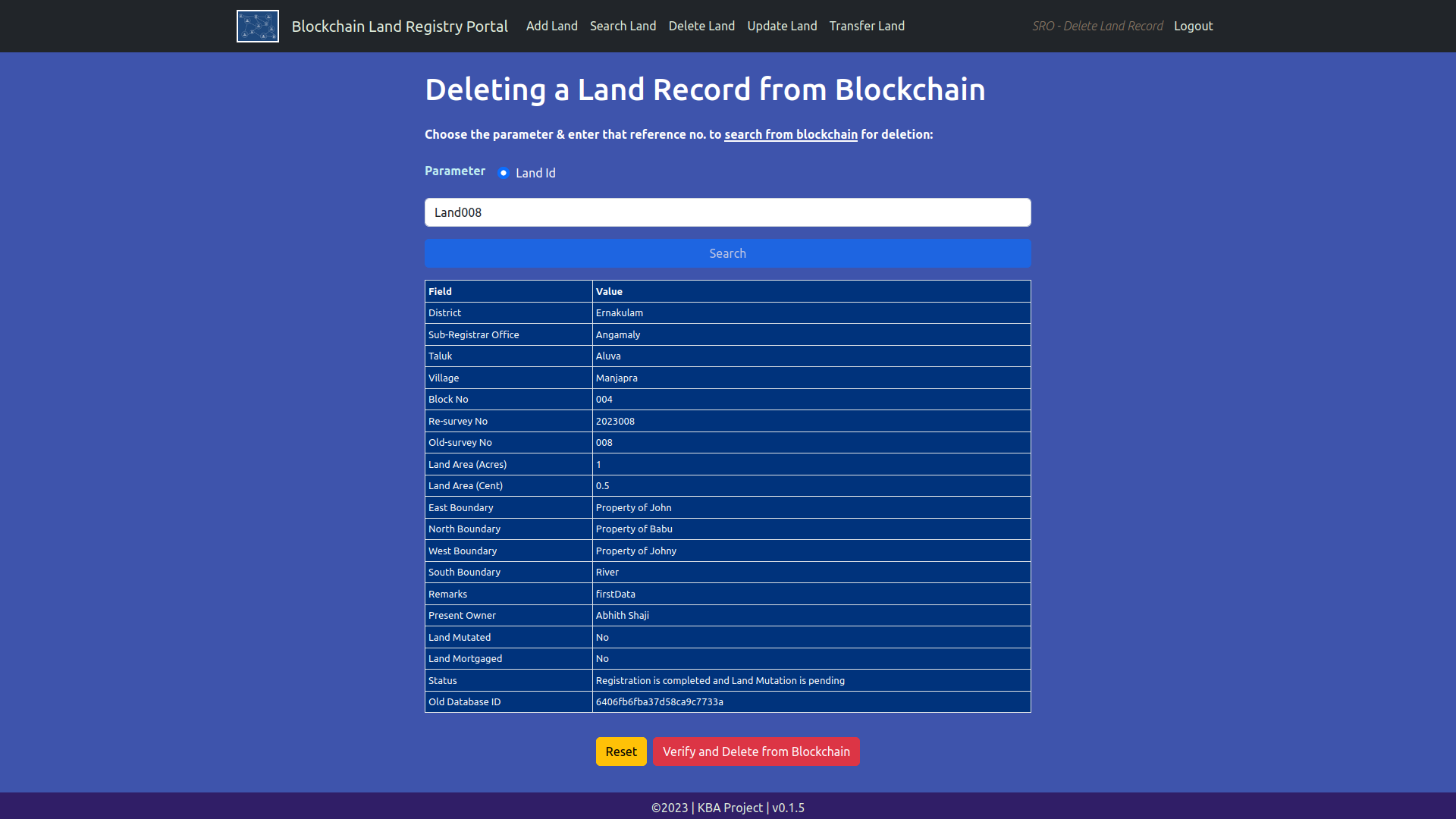




## **Update land details :**



## **Delete land :**



## **Get land history :**

