Smart property based land registration

Features that are already included

Registration

- **Seller & Buyer Registration**: Users can create accounts on the application.

For verification some have used land inspector feature which nothing but admin which will check and verify land And some have used Government authority feature which means government will approve the verification of land

Concept of Land Inspector Dashboard

- **Admin Role**: Land Inspector verifies Sellers and Buyers.
- **Land Transfer Approval**: Manages the land transfer process.

Government Authority Audit

- **Land Details Audit**: Government reviews user land details for approval.
- **Document Submission**: Users upload legal documents during registration.
- **Application Status**: Users receive notifications via NEXMO API and Nodemailer API for acceptance or rejection

After buying the land the whole process is governed by this land inspector or admin

Land Ownership Transfer

- **Transfer Process**: Managed by the Land Inspector for secure ownership transfer.

User Profile

- **View Profile**: Sellers and Buyers can view their profiles via their dashboards.
- **Edit Profile**: Users can update their profile information.

Seller Dashboard

- **Land Overview**: Displays added lands and options to manage them.
- **Add Land**: Sellers can add new land after verification.
- **Approve Land Request**: Sellers can approve Buyer requests for land purchases.

Buyer Dashboard

- **Available Lands**: Overview of lands for sale.
- **Request Land**: Buyers can send purchase requests to landowners.
- **Owned Lands**: Details of lands owned by the Buyer post-purchase.
- **Make Payment**: Complete payment to Sellers after approval.

- **View Lands**: Access comprehensive information, images, and documents of lands.

Transaction Process(through blockchain)

1. **Land Availability**: Once approved, landowners can list their land for sale.
2. **Buyer Request**: Buyers search and request to purchase available land.
3. **View Requests**: Sellers review Buyer requests and communicate with them.
4. **Process Request**: Sellers accept or decline Buyer requests based on criteria.
5. **Property Purchase**: Upon approval, payment is processed, transferring ownership and removing from Seller's assets.

Smart Contract Features

- **Immutability**: Ensures secure and tamper-proof transactions.
- **Transparency**: Reduces chances of fraud with a public ledger.
- **Digital Signing**: Facilitates transfer of land titles upon cryptocurrency payment.
- **Reduced Paperwork**: Streamlines processes, minimizing human error.

Support

- **Queries**: Users can contact for any questions or issues.

Possible gaps

Once the transaction is completed and the buyer receives the land, a PDF containing all the legal documents should be provided to the user. This document should include:

- Power of Attorney (POA)
- Occupancy Certificate
- Payment Receipts
- Affidavit of Title

The user should be able to download this PDF and print it as a hard copy.

Additionally, when the seller sells their property, the authorized government officer should be notified via email or through the website so that they can issue the necessary legal documents.

Procedure

1. **Step 1-** Land owners will register there land in the website by giving details of the land and scanned copy of the legal documents
2. **Step 2-** When the registrar will receive the request from the land owners he/she will start the verification work as soon as the verification work is completed the land details with the owner will be stored in a block in blockchain.
3. **Step 3-** As soon as the land gets registered it will be shown on the buyer portal buyers can checkout the details of the land
4. **Step 4-** if a buyer want to buy a property he/she can contact to the seller through the website itself if they agree
5. **Step 5-** the seller can notify the registrar for scheduling an offline meeting to get all the official work done like signature on legal documents thumb prints payments and witnesses
6. **Step 6 -** Ragistrar will see the request on there portals he/she can approve if the buyer profile verified
7. Step 7 - ones the verification is done email will be sent to both seller and buyer for the confirmation of offline meet
8. **Step 8-** if anyone from the buyer or seller fails to show up registrar can cancel the meeting
9. **Step 9 -** after all the legal work all the papers are then converted to pdfs and later to hash values via algorithms and these hash values will be stored in blocks of blockchain with new buyer name and their details. We can use smart-contract here to automate the process
10. **Step 10-** the physical papers will be returned to the new owner
11. **Step 11-** the registrar has the private key to the blocks which means they can delete or modify them via private key but the buyer will get the public keys to view the legal documents whose hash value can be calculated and matched with that stored in the block like this if someone tries to fake the papers of property by changing them the original details to there details they cant change anything on the e-documents.

BLOCKS OF BLOCKCHAIN

```
+----------------------------------------------------+

|                 Block Header                   |

+----------------------------------------------------+

| Block Number: 1                                |

| Timestamp: 2024-10-21 12:00:00                    |

| Previous Block Hash: 00000000000000000000000000000000 |

| Nonce: 123456                                  |

| Merkle Root: A1B2C3D4E5F6G7H8I9J0K1L2M3N4O5P6Q7R8S9T |
+----------------------------------------------------+
|                Transaction List                    |
+----------------------------------------------------+
| Transaction ID: TX1                                |
| Buyer: Sarah                                       |
| Seller: John                                       |
| Property ID: PROP123                               |
| Sale Price: $300,000                               |
| Status: Completed                                  |
| Legal Document Hashes:                             |
| - Document-1 Hash: H1H2H3H4H5H6H7H8H9H0            |
| - Document-2 Hash: H2H3H4H5H6H7H8H9H0H1            |
| - Document-3 Hash: H3H4H5H6H7H8H9H0H1H2            |
+----------------------------------------------------+
| Transaction ID: TX2                                |
| Buyer: Alice                                       |
| Seller: Bob                                        |
| Property ID: PROP456                               |
| Sale Price: $250,000                               |
| Status: Pending                                    |
| Legal Document Hashes:                             |
| - Document-1 Hash: H4H5H6H7H8H9H0H1H2H3            |
| - Document-2 Hash: H5H6H7H8H9H0H1H2H3H4            |
| - Document-3 Hash: H6H7H8H9H0H1H2H3H4H5            |
+----------------------------------------------------+
```

Step-by-Step Development on Ethereum

Step 1: Landowner Registration and Document Upload

- **Frontend Development:**
  - HTML/CSS: For creating the registration form and user interface.
  - JavaScript (with React.js or Vue.js): React is commonly used for decentralized applications (DApps) because it easily integrates with Web3 tools.
- **Blockchain Interaction:**
  - Web3.js or Ethers.js: Libraries for connecting your frontend to the Ethereum blockchain.
  - IPFS: To upload and store the landowner's scanned documents in a decentralized way. IPFS generates a hash for the document, which you will store on the blockchain.
- **Backend (optional):**
  - Node.js/Express.js: If you need a backend server for handling more complex logic or communication with IPFS and Ethereum.
- **Smart Contracts:**
  - Solidity: The programming language for writing Ethereum smart contracts.
  - Truffle or Hardhat: Frameworks for testing, compiling, and deploying your smart contracts.

Step 2: Registrar Verification and Storing Land Details on Blockchain

- **Smart Contracts:**
  - Solidity: Write smart contracts that handle the verification process and store landowner details.
  - Events: Emit events from smart contracts to notify the system when verification is complete.
  - Truffle or Hardhat: For development, migration, and deployment of smart contracts.
- **Blockchain Interaction**:
  - Web3.js or Ethers.js: These JavaScript libraries will help the registrar's portal interact with Ethereum, to verify landowner details and create transactions.

Step 3: Display Registered Land on Buyer Portal

- **Frontend Development:**
  - React.js or Vue.js: Build the buyer portal that shows land details by fetching data from the Ethereum blockchain.
- **Blockchain Interaction:**
  - Web3.js or Ethers.js: To interact with the Ethereum blockchain and retrieve the land details for display.

Step 4: Contact Between Buyer and Seller

- **Frontend:**
  - React.js: For the messaging system between the buyer and seller.
- **Backend (optional):**

    - Node.js/Express.js: Handle the communication logic between buyer and seller.
- **Blockchain Interaction:**
    - Solidity: Write smart contracts that handle the creation of purchase intent from the buyer, and store it on the blockchain.
    - Web3.js or Ethers.js: To manage the interaction between the buyer/seller contact process and smart contracts.

Step 5: Scheduling an Offline Meeting

- **Smart Contracts:**
    - Solidity: Create a smart contract that stores meeting requests and sends notifications.
    - Web3.js: To trigger the meeting scheduling events from the seller and registrar side.
- **Notifications:**
    - Node.js: You can use Node.js with libraries like Nodemailer to send email notifications based on smart contract events.

Step 6: Registrar Verification for Buyer Profile

- **Smart Contracts:**
    - Solidity: Registrar checks the buyer's profile stored in the blockchain and verifies eligibility.
- **Frontend:**
    - React.js: To display buyer details and verification status.
- **Blockchain Interaction:**
    - Web3.js or Ethers.js: For interaction between the frontend and smart contracts to fetch verification status.

Step 7: Notification of Offline Meeting

- **Smart Contracts:**
    - Solidity: The registrar approves the meeting and triggers the next step.
- **Notifications:**
    - Node.js with Nodemailer for sending email notifications to both buyer and seller after confirmation.

Step 8: Canceling Meetings

- **Smart Contracts:**
    - Solidity: If either party doesn't show up, the registrar can cancel the meeting via a smart contract method. The status is updated on the blockchain.
- **Frontend:**
    - React.js: To display meeting status updates.
- **Blockchain Interaction:**
    - Web3.js or Ethers.js: To update meeting details on the blockchain and notify users.

Step 9: Hashing Legal Documents and Updating Blockchain

- **Hashing:**
  - Use SHA-256 or similar algorithms in Node.js to convert the signed legal documents into hash values.
- **Smart Contracts:**
  - Solidity: Store the hash values along with the new buyer's details in the smart contract.
- **IPFS:**
  - Upload signed legal documents (PDFs) to IPFS, and store the IPFS hash in the blockchain.

Step 10: Return of Physical Papers

- **Frontend:**
  - React.js or Vue.js: To provide a notification system and record for paper handover.

Step 11: Registrar Controls (Private Key) and Buyer Access (Public Key)

- **Blockchain Interaction:**
  - Solidity: Ensure that the registrar can modify or delete records using their private key, but buyers get public keys to verify documents.
  - Use Ethereum wallets like MetaMask to handle public/private key management.
- **Frontend:**
  - Web3.js or Ethers.js: To allow buyers to access the blockchain using their public key and view the hash of their legal documents

---

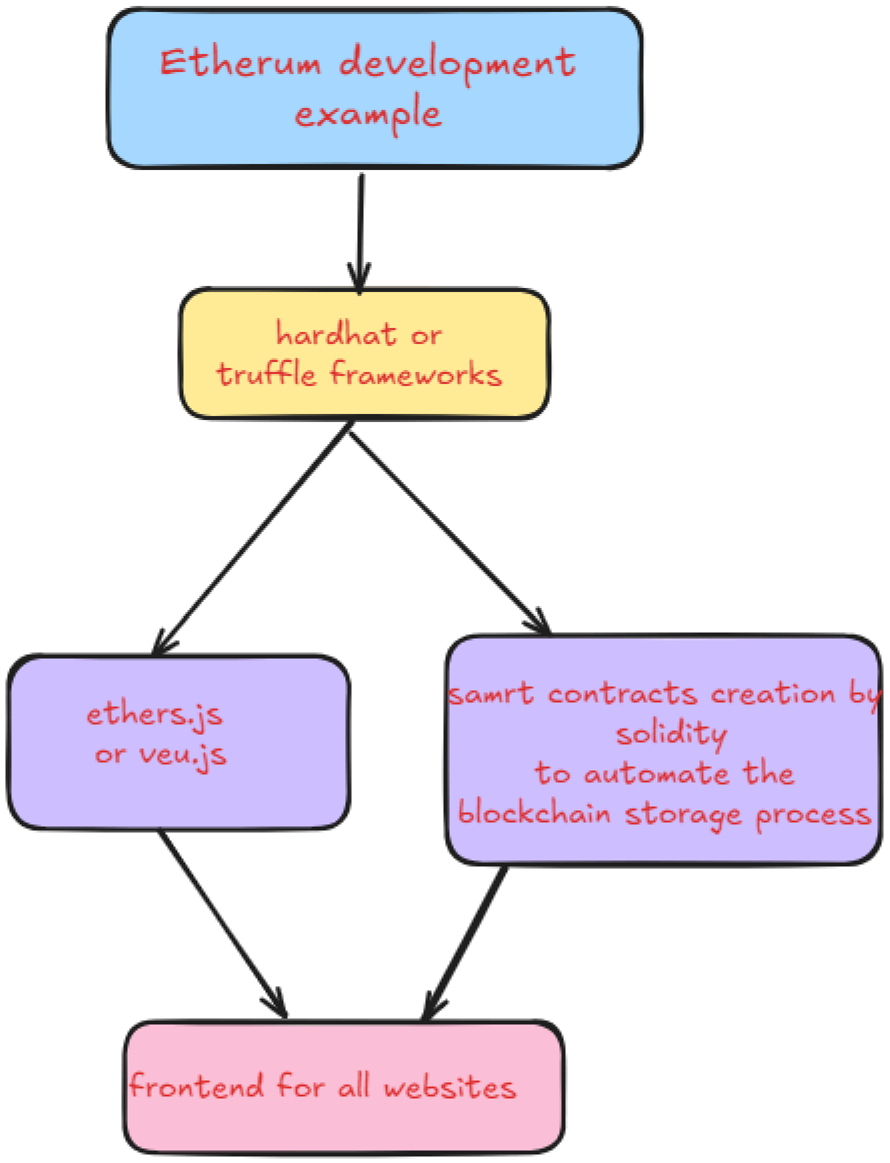
What is IPFS?

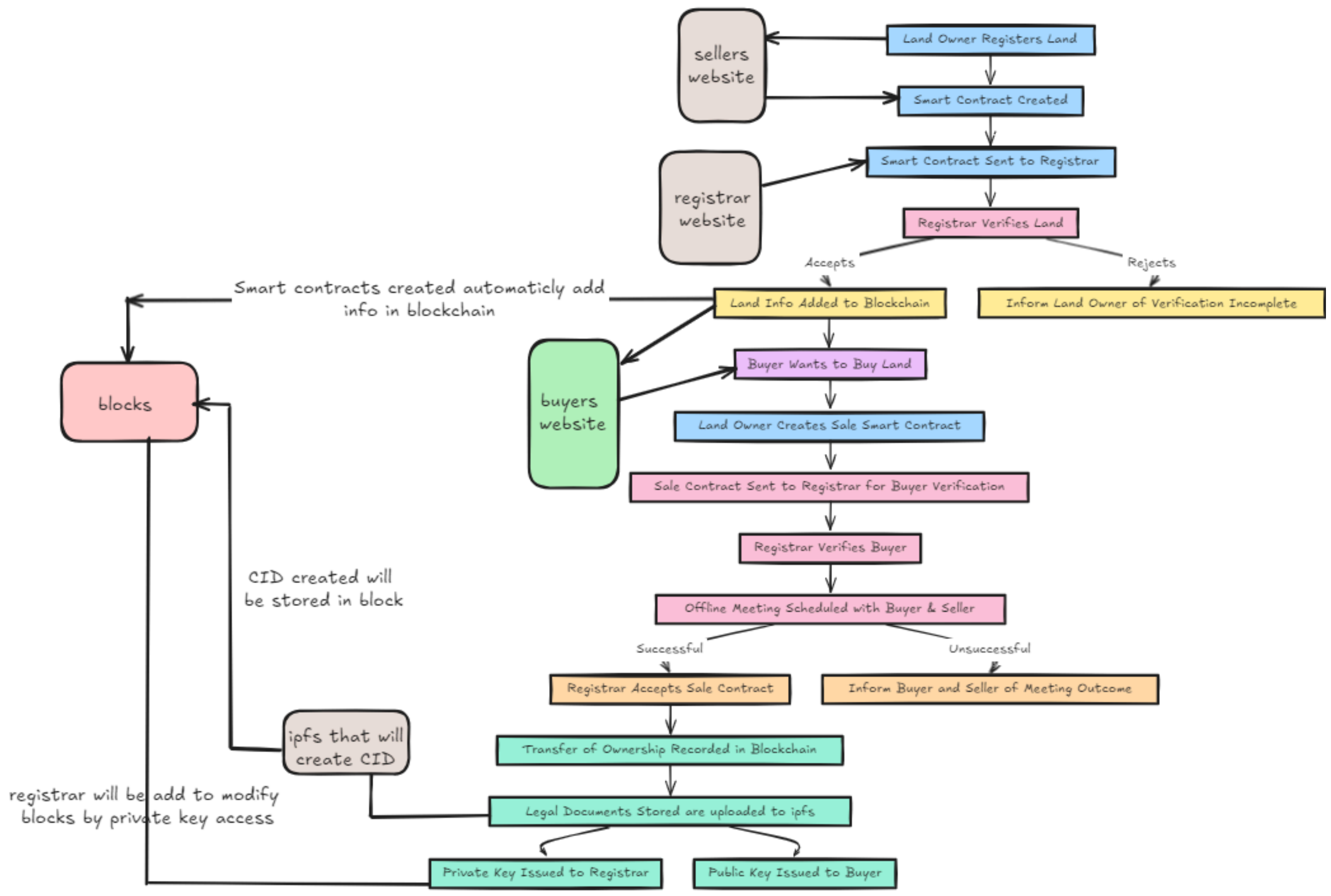
IPFS (InterPlanetary File System) is a decentralized storage system that allows files to be stored across multiple nodes. Instead of storing files on a centralized server (like AWS), IPFS breaks files into smaller pieces, distributes them across different nodes, and generates a hash (unique identifier) for each file.

How IPFS Works with Blockchain (Ethereum)

1. **Upload the Document to IPFS:**
   - The landowner's scanned legal documents (PDFs, images, etc.) will be uploaded to IPFS.
   - IPFS generates a unique hash for each file, which you can store on the blockchain.
2. **Store the IPFS Hash on Ethereum:**
   - After uploading the documents, you get the hash (e.g., `Qm...` format).
   - Store this hash in your smart contract (on Ethereum) to keep a reference to the document. The actual document is stored off-chain in IPFS.
3. **Verify Documents Later:**

- Whenever someone wants to verify the land document, they can download the file from IPFS using the hash stored on the blockchain, ensuring they are viewing the original, untampered file.


Etherum development example
hardhat or truffle frameworks
ethers.js or veu.js
samrt contracts creation by solidity to automate the blockchain storage process
frontend for all websites


sellers website
Land Owner Registers Land
Smart Contract Created
registrar website
Smart Contract Sent to Registrar
Registrar Verifies Land
Accepts
Rejects
Smart contracts created automaticly add info in blockchain
Land Info Added to Blockchain
Inform Land Owner of Verification Incomplete
blocks
buyers website
Buyer Wants to Buy Land
Land Owner Creates Sale Smart Contract
Sale Contract Sent to Registrar for Buyer Verification
Registrar Verifies Buyer
CID created will be stored in block
Offline Meeting Scheduled with Buyer & Seller
Successful
Unsuccessful
Registrar Accepts Sale Contract
Inform Buyer and Seller of Meeting Outcome
ipfs that will create CID
Transfer of Ownership Recorded in Blockchain
registrar will be add to modify blocks by private key access
Legal Documents Stored are uploaded to ipfs
Private Key Issued to Registrar
Public Key Issued to Buyer