# B. TECH. PROJECT REPORT
## on

# Smartphone based Land Records Retrieval System

By
**Naman Jain**
**Pranshu Maheshwari**
**Saksham Tanwar**

**DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING**
**INDIAN INSTITUTE OF TECHNOLOGY INDORE**
May 21, 2021

# Smartphone based Land Records Retrieval System

**A PROJECT REPORT**

*Submitted in partial fulfillment of the*
*requirements for the award of the degrees*

*of*
**BACHELOR OF TECHNOLOGY**
in

**COMPUTER SCIENCE AND ENGINEERING**

*Submitted by:*
**Naman Jain**
**Pranshu Maheshwari**
**Saksham Tanwar**

*Guided by:*
**Dr. Gourinath Banda, Associate Professor**



**INDIAN INSTITUTE OF TECHNOLOGY INDORE**
**May 2021**

# CANDIDATE'S DECLARATION

We hereby declare that the project entitled **Smartphone based Land Records Retrieval System** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering completed under the supervision of Dr. Gourinath Banda, Associate Professor, IIT Indore is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

**Signature and name of the student(s) with date**

# CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my knowledge.

**Signature of BTP Guide(s) with dates and their designation**

## Preface

This report on Smartphone based Land Records Retrieval System is prepared under the guidance of Dr. Gourinath Banda.

Through this report we have tried to give a detailed design of an innovative car control utility for the vehicles and try to cover every aspect of the new design, if the design is technically and economically sound and feasible.

We have tried to the best of our abilities and knowledge to explain the content in a lucid manner. We have also added 3-D models and figures to make it more illustrative.

Naman Jain
Pranshu Maheshwari
Saksham Tanwar
B.Tech. IV Year Discipline of Computer Science and Engineering IIT Indore

# Acknowledgements

We wish to thank Dr. Gourinath Banda for his kind support and valuable guidance.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support this report would not have been possible.

Naman Jain
Pranshu Maheshwari
Saksham Tanwar
B.Tech. IV Year Discipline of Computer Science and Engineering IIT Indore

# Contents

# 1 Introduction

## 1.1 Generic Application

A typical application can be broken down into three logical layers, namely:-

- **Presentation Layer:** It is the layer responsible for displaying information and collect information from the user.

- **Business Logic Layer:** In this layer, information collected from the presentation layer is used to perform the various calculations and operations needed to be performed by the application.

- **Data Layer:** This layer is responsible for storing and retrieving data to be used by other layers.

Data Layer is typically implemented using a database, which is an organized collection of data. A database can be of two types: Centralized and Distributed.

## 1.2 Disadvantages of a centralized database

Most applications require data in some form or another and need to store the said data for future use, and this is where databases come into the picture. A database is an organized collection of data, generally stored and accessed electronically from a computer system. Access to this data is usually provided by a "database management system" (DBMS) consisting of an integrated set of computer software that allows users to interact with one database and provides access to all of the data contained in the database. DBMS's provide various functions that allow management of a database and its data, such as:-

- **Data definition**, which defines the database model, i.e. the logical structure of the data model

- **Updating** of actual data, which includes insertion, modification, and deletion of the actual data

- **Retrieval** of actual data, which may or may not include data processing

- **Administration** which includes allowing access to various users, enforcing data security and various other checks

A database model is a type of data model that determines the logical structure of a database and fundamentally determines how data can be stored, organized and manipulated. The most famous example of a database model is the relational model, which uses a table-based format. The ACID (Atomicity, Consistency, Isolation and Durability) database design is one of the oldest and most important database design. A relational database that fails to follow the ACID model cannot be considered reliable.

Databases can be categorized in many categories depending on how they store data, where it is stored, and the additional functionality the DBMS can provide on the stored data. Broadly databases can be categorized into two major categories centralized and distributed. They both have their advantages and disadvantages.

A centralized database is a database that is located, stored, and maintained in a single location and has a single DBMS but can be accessed by the users distributed in the network.

A distributed database system consists of a collection of local databases, geographically located at different points (nodes of a network of computers) and logically related by functional relations to be viewed globally as a single database [1]. There are two types of distributed databases homogeneous and heterogeneous. In a homogeneous distributed database, all the locations store the database identically, i.e. all locations have the same management system and schema. Whereas in a heterogeneous distributed database, different locations can have different management software, schemas. For a database management system to be distributed, it should be fully compliant with the twelve rules introduced by C.J. Date in 1987 [2] -:

1. Local Autonomy

2. The absence of a dependency from a central location

3. Continuous Operation

4. Location Independent

5. Fragmentation Independent

6. Replication Independent

7. Distributed Query Processing

8. Distributed Transaction Management

9. Hardware Independent

10. Operating System Independent

11. Independent of Communication Infrastructure

12. Independent of Database Management System

If a centralized database is used to store the data, then the whole system is prone to a Single Point of Failure; if for some reason the database fails, the whole system will fail. As all the data is stored in a single location, if there are no precautionary measures taken, then a hardware failure can lead to complete data loss. The databases are also vulnerable to cyberattacks which can cause data loss, data leaks, data inconsistency and many other vulnerabilities. This makes a centralized database dubious with very low reliability. However, at the same time, a centralized database ensures data consistency and easy management in compliance with ACID design [3].

A major advantage of distributed database is that by sharing a database across multiple nodes, the database can obtain a storage space extension and benefit from multiple processing resources. A distributed database system is robust to failure to some extent. Hence, it is reliable when compared to a centralized database system. It is also more robust than a centralized database as it does not have a single point of failure. If a node fails, another node or group of nodes can provide the necessary data. However, to make this possible complex software's are required, which incur additional costs and processing overheads. Marinating data integrity is difficult, and hence minimum redundancy and ACID properties are more relaxed than a centralized database [3]. The distributed environment also faces problems such as fragmentation and data replication. A data fragment constitutes some subset of the original database. A data replica constitutes some copy of the whole or part of the original database. The fragmentation and the replication can be combined: a relationship can be partitioned into several pieces and can have multiple replicas of each fragment [4].

## 1.3 Advantages of using Blockchain

Blockchain is a type of decentralized database where the data is stored in blocks, and each block is linked to the previous block using its cryptographic hash, thus forming a chain. As the blockchain grows in size, it becomes more and more difficult to tamper with the data in these blocks. Blockchain based databases are more trustworthy, reliable and secure than traditional databases.

A blockchain based system is decentralized. Hence, a centralized authority is not necessary. Decentralized systems are also resilient to single point of failure. A blockchain network comprises of various nodes, each maintaining its own copy of database. Hence, all the data is replicated, shared and synchronized across all the nodes in the network. This makes the data almost tamper proof as it will be very costly and thus highly unlikely for a malicious person to change the data across the whole network. No node can directly write data to the blockchain as it can make the data inconsistent, to avoid this the blockchain network uses a consensus algorithm which helps all the nodes in the network to come to a consensus and commit the data in the blockchain. This helps to keep all the nodes in the network synchronized with each other.

The data that is committed to the blockchain is immutable. After some data is inserted into the blockchain, it is very difficult to delete or alter it. This is realized by including, in each block, the cryptographic hash of its previous block. Because of this immutability, a blockchain database only supports create and read operations in contrast to traditional databases, which supports create, read, update and delete (CRUD) operations. To update the database state, a subsequent write is required. Old writes persist in the blockchain forever, making it convenient to trace how the blockchain database state is updated over time. This makes it possible to trace the actions that resulted in a particular database state.

Another benefit of blockchain is that it provides transparency to the data stored in it. This is achieved by replicating the blockchain data among multiple nodes present in the blockchain network. Nodes in the blockchain network can view the data already committed to the blockchain and participate in the process of validating the data to be committed in the blockchain. The transparency also helps users to cross verify some data against the blockchain.

A blockchain ledger is logically a single public ledger. Since every node updates its copy of the blockchain ledger in sync with each other and after consensus is reached, the whole network logically acts as one public ledger. This removes the complications present in traditional systems where different participants or organizations maintain multiple ledgers, which needs to be reconciled and synchronized from time to time.

Systems that are deployed on public blockchain network are more cost efficient as compared to traditional systems. This is because the system utilizes the processing power and resources of many nodes that are already connected to the blockchain network, thus significantly reducing the cost needed for setting up and maintaining centralized servers present in traditional systems.

Blockchain based systems are more secure and resilient to cyber attacks, which can cause data loss, data leaks, data inconsistency and many other vulnerabilities than traditional systems. Whenever a new block is introduced in the blockchain, its hash value is calculated, which also includes the hash of its previous block. If a malicious user fraudulently tries to tamper with the data inside a block, not only its hash value changes but the hashes of the following blocks get changed too. This, along with the fact that the data is stored in multiple nodes present in the blockchain network, makes it very difficult to tamper with the data present in the blockchain.

## 1.4  IPFS

The Interplanetary File System (IPFS) [5] is a peer to peer distributed file system for sharing and storing data. IPFS provides a high throughput storage system where a cryptographic hash uniquely identifies each file. In IPFS, multiple users store a piece of each others' data and actively participate in making it available in the network. In contrast to HTTP, which uses location based addressing, IPFS uses content based addressing. In content based addressing, user request data using its content identifier instead of the location where it is stored.

IPFS uses Distributed Hash Table (DHT) [6] to enable routing and discovery of peers and content on the network. The hash table is split across all the participating nodes in the IPFS. These nodes coordinate with each other to enable efficient lookup and retrieval of data on the network. IPFS stores data using a data structure called Merkle DAGS [5]. Each node in a Merkle DAG stores the cryptographic hash of each of its children. Thus, altering a node's data changes the hash value of the node and all its ancestors in the DAG. This makes the data stored on the IPFS immutable.

## 1.5  Motivation

In India, currently, if a person is asked to prove if he/she is the owner of a piece of land, then all they can show for it is a sale deed which just proves that the person was the owner at a particular time, but that person cannot prove that he/she is still the owner of that land. For proving the same, he/she has to go to various government offices and collect various document showing that no sale deed has been registered for that land after the one which the person has. When a person wishes to buy land in India, they have to be very careful and perform various checks such as:-

- Check if the deed title is in the name of the seller and if he/she has the full right to sell it

- Procure a Encumbrance Certificate from sub-registrar's office where the deed is registered which declares that the land is free of any legal hassle and unpaid dues

- Check if the Property tax and other bills are paid in full and the seller has the respective original receipts

- Check if the loan on the land has been completely repaid

If a buyer is lethargic in verifying these documents, he/she can be easily duped into buying a disputed land parcel, or it may even happen that the land in question did not even legally belong to the said seller, that is, the seller was not the genuine owner of that land and provided fake documents. As these documents are primarily maintained offline as hard copies, these cases often occur.

Getting all the information, such as the sale deeds and ownership history for a land parcel, is cumbersome. A simple solution to this problem is a smartphone application with which users/potential buyers can get land ownership history and related documents for a particular land parcel. These documents and data can be stored in a database by the government and will be updated every time a new sale deed is registered, i.e. owner for a land parcel changes. This makes the process of acquiring the documents in question straightforward and hassle-free.

Indian government launched Digital India Land Record Modernization Programme in 2008. This programme aims to create a system of updated land records, automated and automatic mutation, integration between textual and spatial records, inter-connectivity between revenue and registration, to replace the present deeds registration and presumptive title system with that of conclusive titling with title guarantee [7]. In addition to this, the Indian government also plans on storing the digital records on a private blockchain and provide a system where citizens can view the ownership details and complete history of the property before going in for the purchase [8].

## 1.6  Proposed Solution

To tackle this problem of accessing land records, a smartphone based solution is proposed. Users can request land records for a particular land parcel by visiting the land parcel or marking it on the map. Land records are stored on a private blockchain which provides methods to access the data. For help in migrating old records to the new system, scanned documents can be uploaded. These documents are stored in a private IPFS cluster with their Content Identifier (CID) stored on the blockchain. A verification portal is also part of the solution where users, while accessing records, will be given a verification code which users can use to generate a digitally signed PDF for verifying the land records.

## 1.7  Organisation of Dissertation

A brief introduction to our proposed solution and the motivation behind it was discussed in chapter one. Chapter one also provided the generic application architecture and contrasted the centralized databases with blockchain databases. Chapter two provides an introduction to the various offerings of blockchain technology that makes it useful for the proposed solution. Chapter two also provides a summary of related works. Chapter three describes the proposed solution's architecture, including the various actors/users of the solution, various submodules as part of the solution and how they interact with one another. Chapter four provides details regarding the deployment of the proposed solution. Chapter five concludes with the main contributions of this work and proposes a direction for future work.

# 2 Advantages of Decentralized Solutions

## 2.1 Blockchain

[9] discusses various complications present in the current land record management system in India. The existing land records in India are not administered properly, due to which they often are not consistent with the ground truth. Various departments, at the village or district level, are involved in maintaining the ownership history of the land. Often, the data present with different departments are not in sync with each other, which results in discrepancies. Such discrepancies often result in conflicts in land ownership, double-spending problem, etc. The data is also susceptible to alteration by corrupt officials.

Blockchain provides a potential solution to various complications present in the current land record management system. Being a distributed database, Blockchain based land record system is resilient to the single point of failures. Data is replicated across multiple nodes in the Blockchain network. Hence, it prevents any undesirable data loss, as was seen during Haiti 2010 Earthquake [10].

A Blockchain ledger contains transactions that are endorsed by appropriate authorities. These transactions, being committed in chronological order, provides a verifiable audit trail for legal investigations [11]. Once committed to the Blockchain, the transactions cannot be modified or deleted, hence providing a trusted source of information.

Blockchain is a shared ledger. Each participant has a copy of the ledger. With necessary permissions, a participant can commit data into the ledger, which, after getting validated by other nodes, gets reflected across the entire Blockchain network. This removes the complications present in the current land record management systems where different departments maintain land data which needs to be reconciled and synchronized regularly.

## 2.2 IPFS

IPFS, being distributed, is resilient to a single point of failure. It is more secure to DDos attacks than a centralized system. Multiple nodes can simultaneously participate in distributing a particular file resulting in better throughput. The data retrieved from IPFS can be verified by calculating its hash. Hence, data stored on IPFS is self-verifiable. Data can be pinned on multiple nodes to ensure it persists on IPFS. This makes IPFS resilient to data loss.

## 2.3 Related Works

The Republic of Georgia, in 2016, undertook a phased project to create a Blockchain-based land titling system. The project has significantly improved

the government's efficiency and ensures the security and immutability of the data [12].

Sweden's land registry authority Lantmäteriet tested a blockchain-based solution to store land records and allow users to buy and sell land by using a smartphone application. The authenticity of the process and digital signatures are secured using the blockchain [13].

International Journal of Information Management [9] lists the various problems in the current land record management system in India, and suggests a blockchain-based solution and gives a detailed flow of the land registration process in India and how blockchain technology improves the registration system.

In 10th ICCCNT, [14] suggests a permissioned blockchain-based system for managing land records using Hyperledger Composer for permissioned blockchain implementation and IPFS for storing various documents.

Journal of King Saud University - Computer and Information Sciences [15] proposes a three-phase project for adopting a blockchain-based solution for land titling problem in Bangladesh and provides an implementation of a prototype model using Ethereum. Phase One proposes adopting public blockchain such as Ethereum to store timestamps of land transactions on the blockchain. Phase Two proposes a hybrid blockchain solution consisting of public and private blockchain networks, where key information is stored on the public blockchain. All data related to land is stored on a private blockchain network. Phase Three proposes using IPFS to store digital documents in the hybrid blockchain solution.

# 3 Proposed Solution Architecture

The proposed solution consists of:-

- A smartphone application where users can request land ownership records.

- A platform for users to request digitally signed certificate from the Land Registration Department to verify land records.

- A platform for Land Registration Department to enter land records, with support for uploading old scanned documents to help in the migration of old records.

Land Ownership Records and the ownership history are stored on a blockchain, making it tamper-proof. A potential land buyer can request the ownership history using the smartphone application after paying a minimal fee and can also request for a digitally signed certificate from the Land Registration Department to validate the data.

## 3.1 Overall Architecture
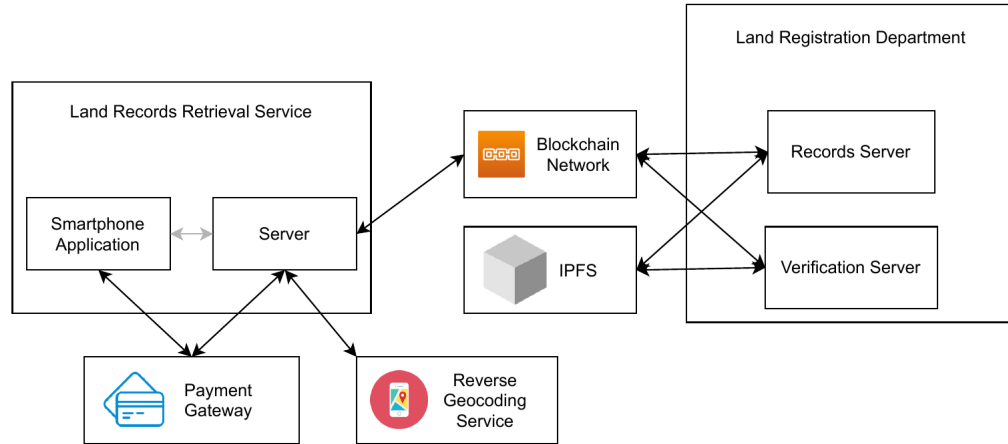


Figure 1: Overall Architecture

Overall solution architecture consists of the following (as seen in Figure 1):-

- **Land Records Retrieval Service:-** Responsible for providing land records service to the users who want to obtain land records by visiting the land parcel itself or marking the land on the map.

– **Smartphone Application:-** Provides a user interface for users to interact with where users can request land records for a particular land by marking the point on the map.

– **Server:-** Backend server for the Smartphone Application. It interacts with the blockchain network to compile the summary of the land records requested by the users.

- **Payment Gateway:-** Razorpay payment gateway service is used to process payments.

- **Reverse Geocoding Service:-** Currently, no service exists which has accurate data of villages or sub-districts. Under the Digital India programme, if some service is introduced for accurate reverse geocoding, that can be used. For the proposed solution, a sample service is used as described.

- **Blockchain Network:-** Blockchain network is responsible for storing all the land records consisting of details about all the land parcels and their transactions.

- **IPFS:-** Since a blockchain network is not ideal for storing files, IPFS is used to store files such as certificates and scanned documents.

- **Land Registration Department:-** Responsible for adding records to the blockchain and providing a verification portal where users can obtain digitally signed certificates to verify land records.

    – **Records Server:-** Responsible for serving the website using which Land Records can be added to the blockchain.

    – **Verification Server:-** Responsible for serving the website using which Land Records can be verified by allowing users to download digitally signed certificates containing the information stored on the blockchain.

Users who use the proposed solution:-

- **Land Information Seeker:-** These users need to access land information; they can be the prospective buyers for a particular land looking to validate the current owner's claim on the land.

- **Land Registration Department Admin:-** These users enter information on the blockchain by logging in to the Records Portal.

## 3.2  Records Server

This serves a web application that allows users in Land Registration Department to enter and query land records on the blockchain. To support the

migration of old land records onto the new platform, uploading scanned documents with each transaction such as adding a land record, transferring a land or splitting a land record is supported. All the documents are stored in the IPFS.

For each land record, a PDF certificate, digitally signed by the Land Registration Department, is generated, which includes all the information about the land such as the current owner, khasra number etc. This certificate is stored in the IPFS.

For each land transfer, a PDF certificate, digitally signed by the Land Registration Department, is generated, which includes all the information related to the land transfer. This certificate is stored in the IPFS. In a land transfer, the land record certificate is also updated to reflect the current owner.

This web application provides the following operations to the user(Land Registration Department Admin):-

- Add Land Record:- Add a land record.

- Transfer Land:- Transfer land ownership.

- Split Land:- Splits a land record into two separate land records, useful in cases when a part of land needs to be transferred.

- Query Records:- Users can query records in any particular State, Sub-District, District or Village.

- Query Ownership History:- Users can query the ownership history of a particular land.

Records Server is implemented using Express framework, a NodeJS module.

### 3.2.1  Add Land Record

Adding a land record entails:-

1. Get input from the user.

2. Generate certificate from the details.

3. Upload certificate to IPFS and receive CID for the certificate.

4. Create land record on the blockchain.
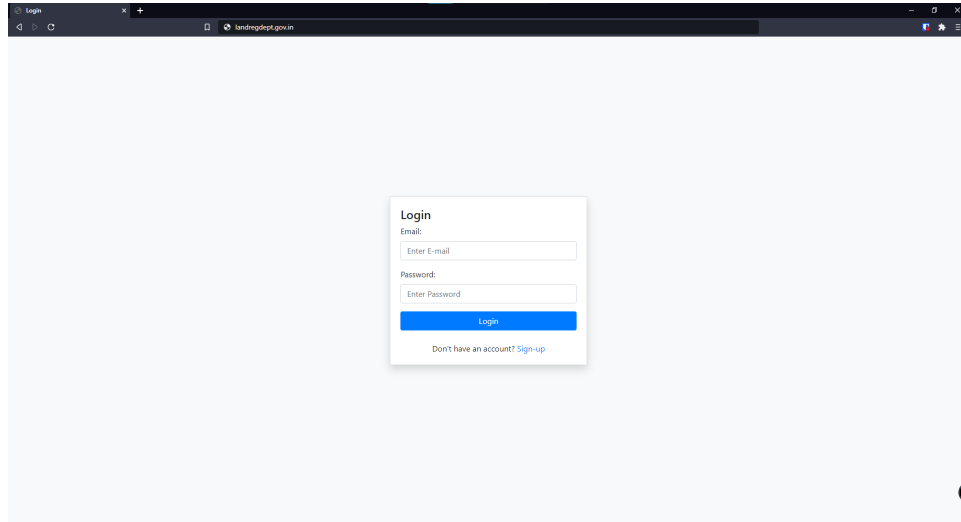
Figure 2: Login Page for the Records Server
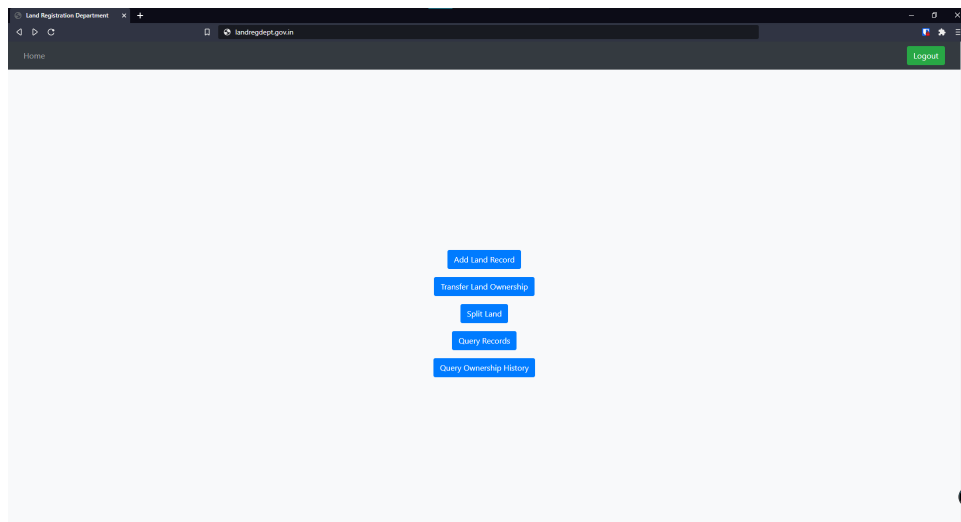


Figure 3: Home Page for the Records Server

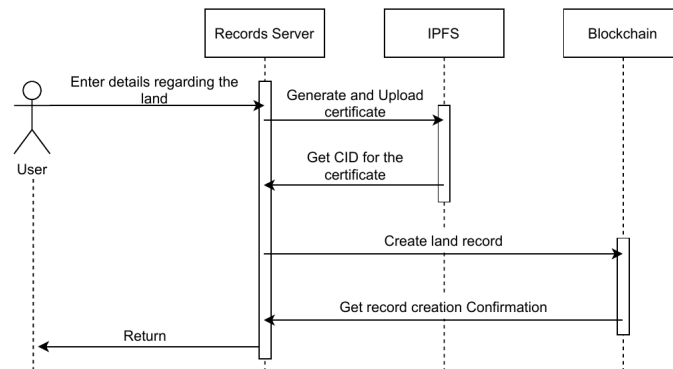Figure 4: Add Land Records Page for the Records Server



Figure 5: Sequence Diagram for Adding Land Records

### 3.2.2 Transfer Land

Transferring a land record entails:-

1. Get input from the user.

2. Generate land transfer certificate containing details about the land transfer.

3. Generate land record certificate containing details about the land and current owner.

4. Upload certificates to IPFS and receive CID's for the certificates.

5. Create land transfer record and update the land record on the blockchain.



Figure 6: Transfer Land Page for the Records Server

### 3.2.3 Split Land Record

Splitting a land record entails:-

1. Get input from the user.

2. Generate certificates for the two land records.

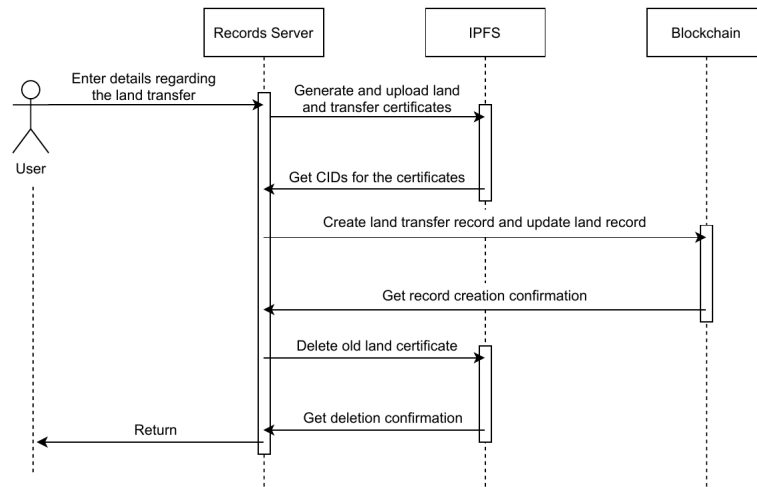3. Add two new land records and mark the old land record as expired on the blockchain.

20

Figure 7: Sequence Diagram for Transferring Land Records



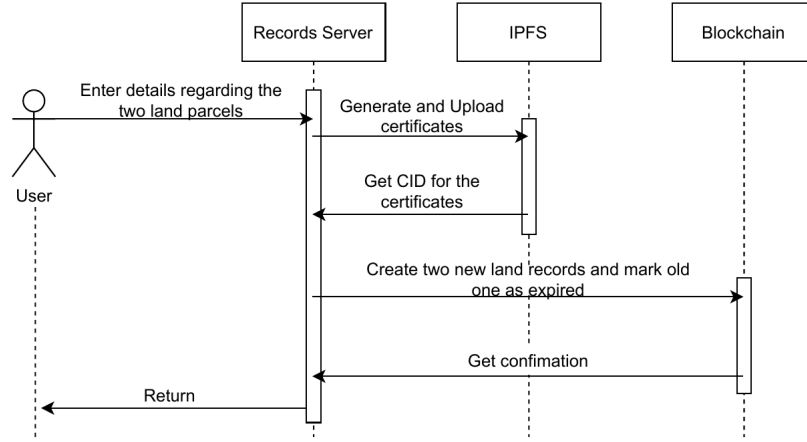Figure 8: Split Land Record Page for the Records Server

Figure 9: Sequence diagram for Splitting a Land Record

### 3.2.4 Query Records

Users can query all the records in a particular village, sub-district, district or state. Querying records entails:-

1. Get input from the user.

2. Query records from the blockchain.

### 3.2.5 Query Ownership History

Users can query the ownership history of a particular land. Querying ownership history entails:-

1. Get input from the user.

2. Query ownership history from the blockchain.

## 3.3 Smartphone Application

This application is used to interact with the LRRS Server and provide a user-friendly mechanism for the users to request land ownership history records. The application is built using React Native, a JavaScript library used to build cross-platform applications using a single React codebase, i.e. we can build both Android and iOS application using the same code base.
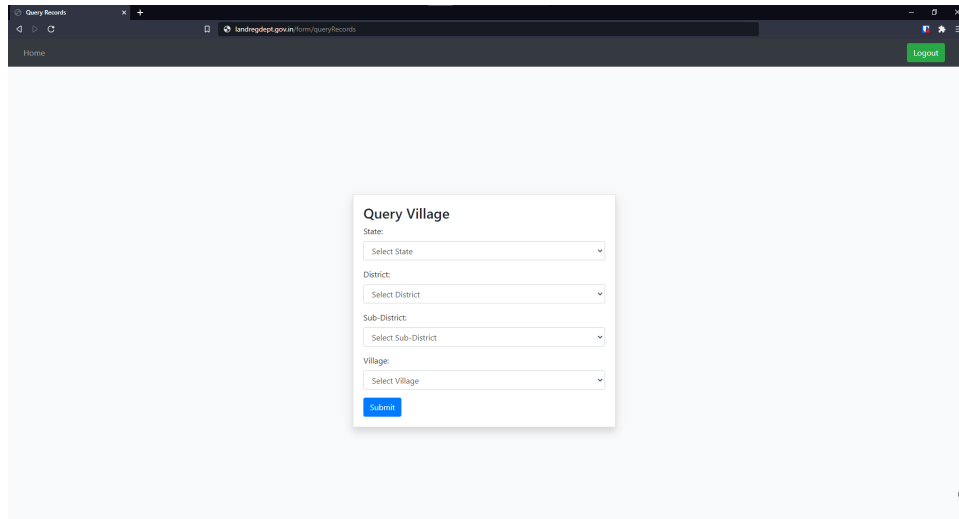
Figure 10: Query Land Records Page for the Records Server
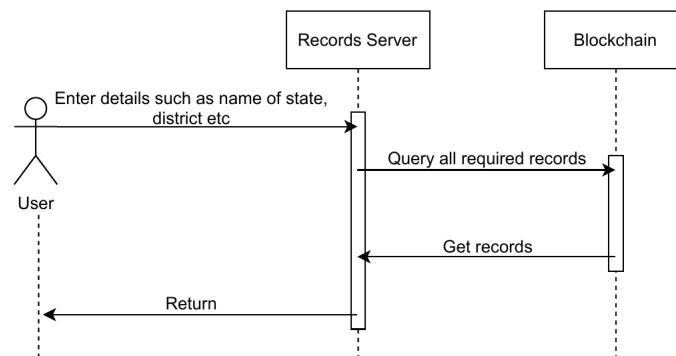


Figure 11: Sequence diagram for Querying Land Records
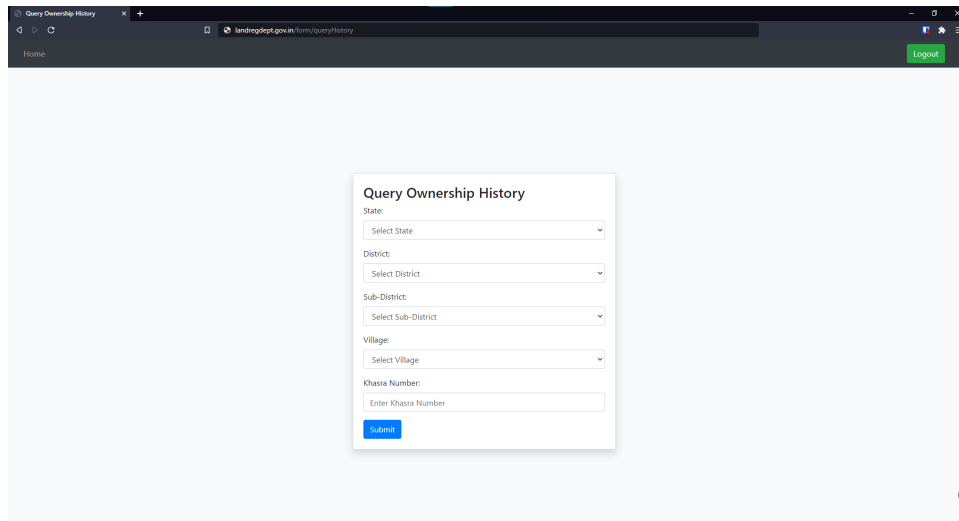
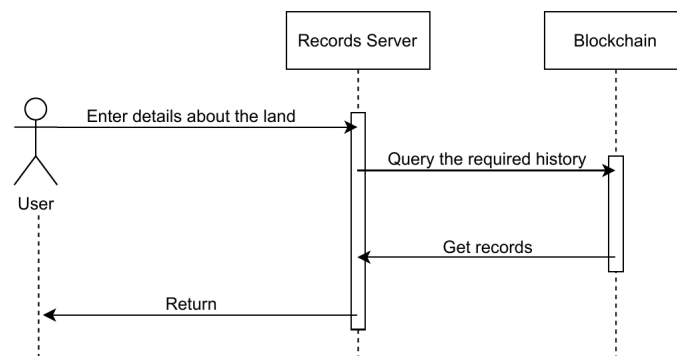Figure 12: Query Land Ownership History Page for the Records Server



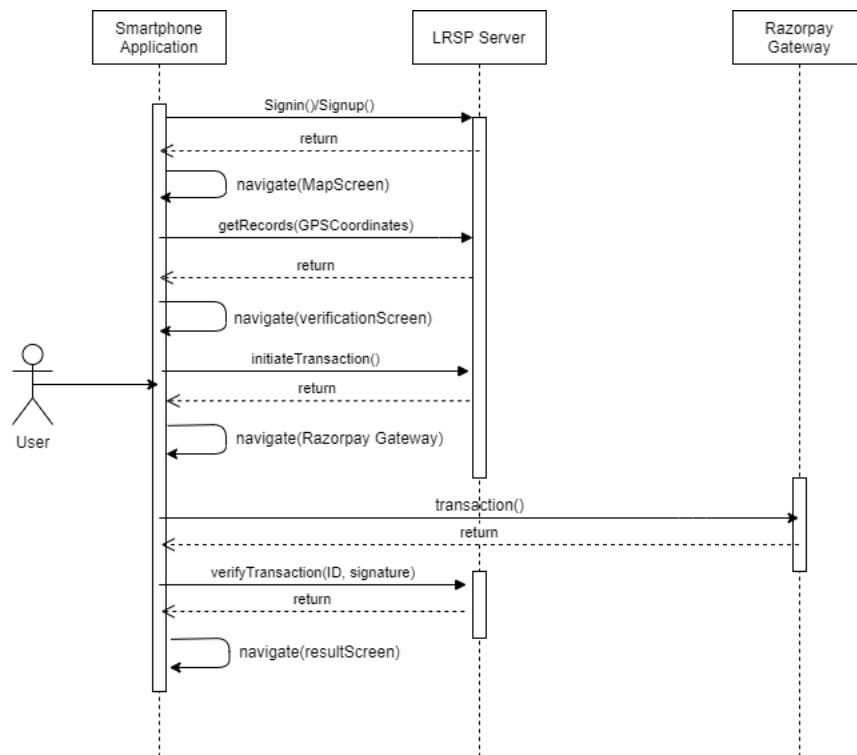Figure 13: Sequence diagram for Querying Land Ownership History

Figure 14: Sequence diagram for Smartphone Application
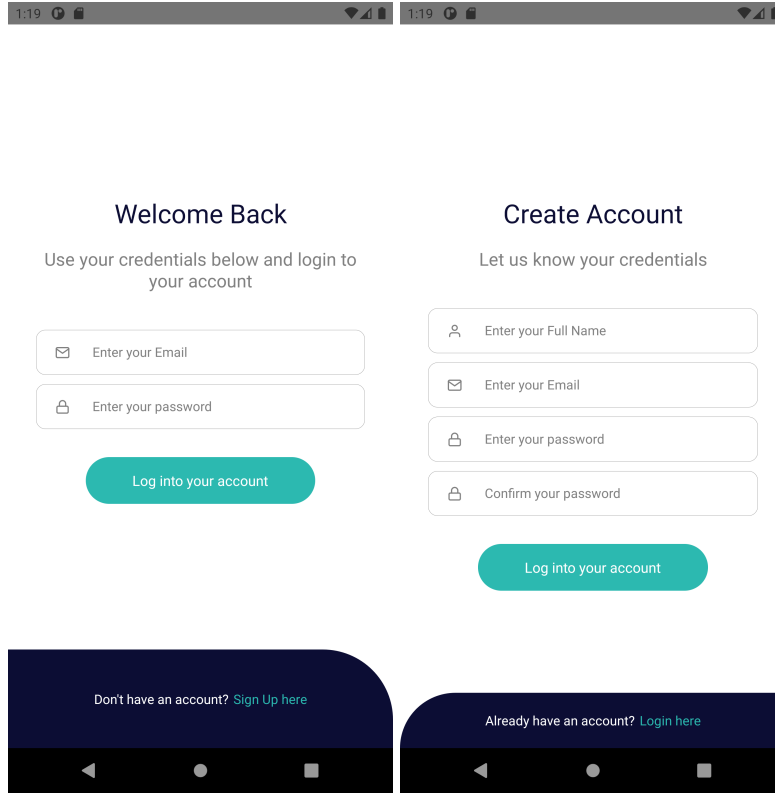
Figure 15: Login and Signup Screen for Smartphone Application

### 3.3.1 Sequence Flow

Initially, the user has to log in to the application using an email and password. If they are new to the platform, they can register a new account, as seen in Figure 15.

After successful authentication, the user is redirected to a screen, as shown in Figure 16 where they can request the land ownership history of a land parcel by placing the marker anywhere in the land area and pressing Submit button. The app then makes a GET request to the LRRS server's /landrecord endpoint with the marked GPS coordinates as the parameters and, in response, receives the data for the land parcel, which then displayed in a confirmation screen (Figure 17) where the user confirms if the data fetched is for the correct land parcel and if so proceeds with payment by pressing on the Continue with Payment button. By pressing on the button, the user initiates a transaction with the payment gateway, i.e. Razorpay gateway and is redirected to the said gateway, as shown in Figure 18. If the transaction is completed successfully, then the Payment ID and Payment Signature of the transaction are sent to the LRRS server via a GET request to the /payment/verify endpoint for verification. If

26
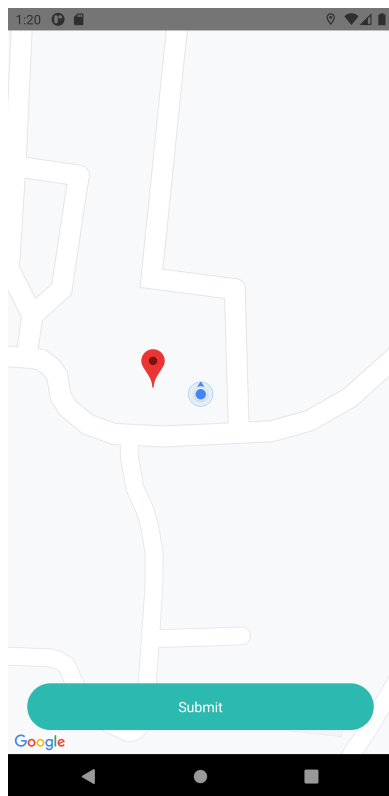
Figure 16: Request Screen



Figure 17: Confirmation Screen
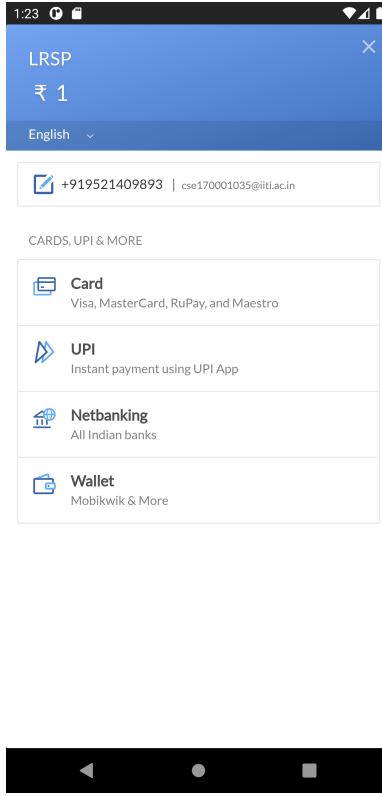
Figure 18: Payment Screen

the transaction is successfully verified, then the app redirects to a Success screen (Figure 19) and the ownership history report of that land parcel is emailed to the user. If the payment transaction fails for any reason, then a failure screen (Figure 20) is displayed, prompting the user to redo the process.

## 3.4  LRRS Server

It is a Node server that interacts with the smartphone application to provide the requested land parcel details to the user. It acts as an intermediary between the LRRS smartphone application and the Blockchain network. The application sends GPS coordinates to the LRRS server requesting for the land record. The server resolves it and fetches the corresponding land record information from the Blockchain. The land parcel details are then e-mailed to the user once the user completes the payment. It supports HTTPS for secured interaction.

The server also provides user authentication functionality to the LRRS system. It uses a centralized database to register users with the system. It uses
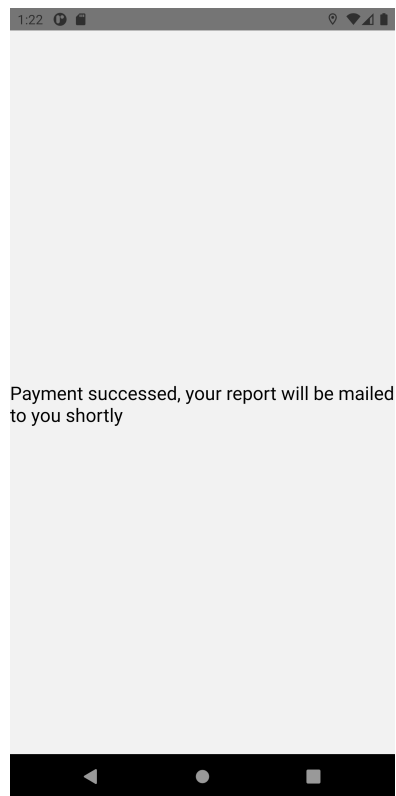
Figure 19: Payment Success Screen
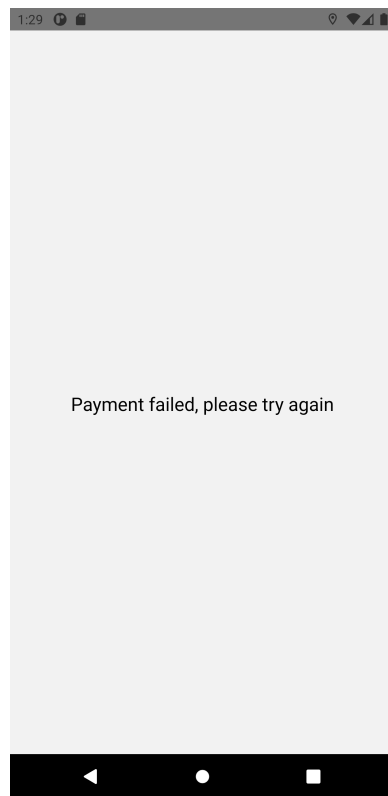


Figure 20: Payment Failure Screen

JWT-based stateless authentication to allow access to protected routes.

LRRS server is implemented using Express framework, a NodeJS module.

### 3.4.1 Sequence Flow

The server resolves the GPS coordinates obtained from the application to get the corresponding State, District, Sub-District and Village information. This is accomplished by using a Reverse Geocoding Service. The server interacts with the Blockchain network to fetch the corresponding record. The records are queried at the village level, and the land parcel which contains the GPS coordinate inside its polygon boundary is searched. Once the record is found, the PDF (as seen in Figure 22) is generated and mailed to the user after payment is verified.

The generated PDF contains general details about the land parcel on top. It also includes a Verification Code that can be used to verify the current owner of the land parcel using the Verification server and selecting the record type as Land Record. The PDF also contains a table comprising the ownership history of the land parcel from genesis with the Verification Code for each Transfer Record, which can be used to verify each transaction. These verification codes are the hash ID's of the records stored on the IPFS. The users can request these certificates from the verification portal to validate the land record and the transfer records.
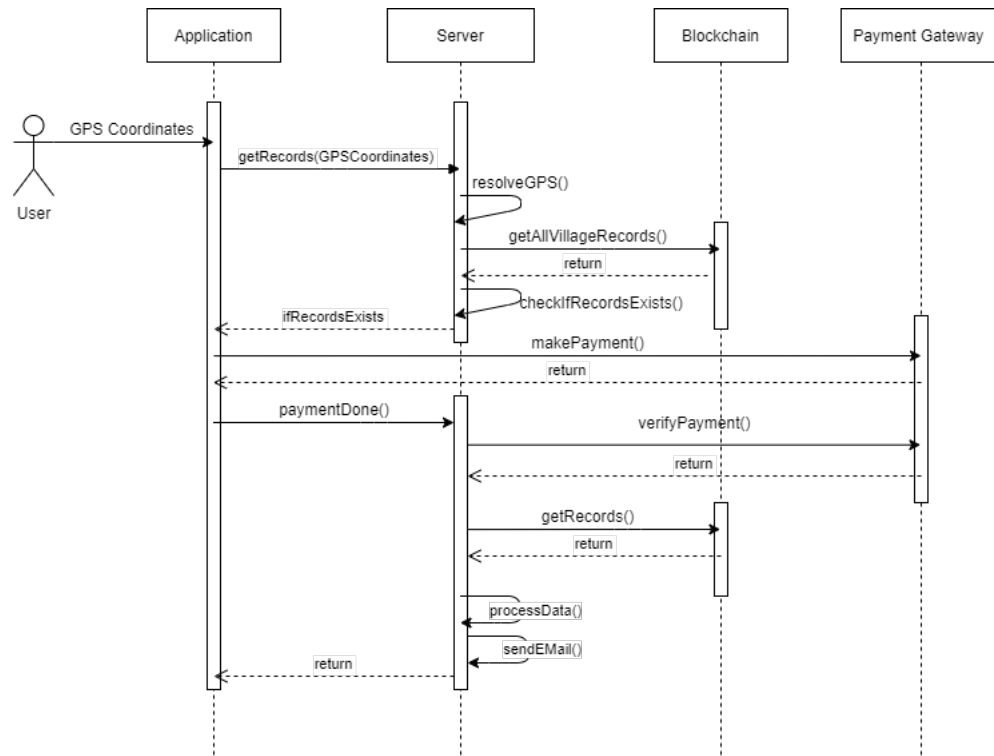
Figure 21: Sequence diagram for LRRS Server

**Land Ownership History**

**Land Parcel:-**

| | | |
|---|---|---|
| **Khasra No :-**<br>101 | **Village :-**<br>aurangabad | **Sub-District :-**<br>aurangabad |
| **District :-**<br>aurangabad | **State :-**<br>maharashtra | **Area :-**<br>1003 sq m |

**Verification Code :-** QmYaWMa2UecxsvQXHywoi869osdjtvuA4JPjrk2RnFZPxx

**Ownership History :-**

| S. No. | Transaction Details | | | |
|---|---|---|---|---|
| 0 | **Khasra No :-**<br>101 | | **Owner :-**<br>DEF Person | |
| 1 | **Khasra No :-**<br>101 | **Date :-**<br>19/4/2021 | **Seller :-**<br>ABC Person | **Buyer :-**<br>DEF Person |
| | **Verification Code :-** QmcCkB2feeXEomHztQREePMBkdfjkTrtt9rp2fsmxfcHyM | | | |
| 2 | **Khasra No :-**<br>101 | **Date :-**<br>15/5/2021 | **Seller :-**<br>DEF Person | **Buyer :-**<br>GHI Person |
| | **Verification Code :-** QmeTyqks3VWYUKvfemHrgWMstb2fpsvigzfgYEFqUZoZ6a | | | |

Figure 22: Land Ownership History PDF

### 3.4.2 Usage

Following are the web endpoints exposed by the LRRS server:

- **POST** /signup:

  This endpoint is used to register a new user to the LRRS server. If successful, it returns the jwt access token in the response body. The jwt token expires in 1 hour.

  **Request Parameters:**

  - name: user name
  - email: user email address
  - password: user password

  **Success Response:**

```
{
    "success": true
    "token": <jwt_token>
}
```

- **POST** /login:

  This endpoint is used to get the jwt access token to access protected endpoints. The jwt token expires in 1 hour.

  **Request Parameters:**

  - email: user email address
  - password: user password

  **Success Response:**

  ```
  {
      "success": true
      "token": <jwt_token>
  }
  ```

- **GET** /landrecord

  This endpoint is used to get the land information corresponding to a particular GPS coordinates that include khasra no, subdistrict, district, state and vertex points of land boundary.

  **Require Authentication:** True

  **Request Parameters:**

  - lat: GPS coordinate latitude
  - lon: GPS coordinate longitude

  **Success Response:**

```json
{
    "success": true,
    "data": {
        "khasra": "1",
        "village": "abu said",
        "subDistrict": "ajnala",
        "district": "amritsar",
        "state": "punjab",
        "points": [
            {
                "lat": "31.894083",
                "lon": "74.834157"
            },
            {
                "lat": "31.894229",
                "lon": "74.835573"
            },
            {
                "lat": "31.892662",
                "lon": "74.835723"
            },
            {
                "lat": "31.892708",
                "lon": "74.833642"
            }
        ]
    }
}
```

- **GET** /payment/initiate

    This endpoint is used to initiate a payment request to obtain the requested land record.

    **Require Authentication:** True

    **Request Parameters:**

    – khasraNo: khasra no of land

    – village: Village name corresponding to land.

    – subDistrict: Sub-District name corresponding to land

    – district: District name corresponding to land

    – state: State name corresponding to land

**Success Response:**

```json
{
    "order_id": 1234
    "amount": 100
}
```

- **GET** /payment/verify

  This endpoint is used to submit details about the successful transaction to the server for verification.

  **Require Authentication:** True

  **Request Parameters:**

  - order_id:
  - razorpay_order_id:
  - razorpay_payment_id:
  - razorpay_signature:

  **Success Response:**

```json
{
    "success": true,
}
```

## 3.5   Verification Server

The verification server provides a portal to obtain the PDF certificates issued by Land Registration Department. The certificates are digitally signed by the respective authorities and contain information regarding land records and transactions. These certificates provide proof for land ownership and transactions and can be used for validation purposes.

Verification Server is implemented as a Node server using ExpressJS.

### 3.5.1   Types of Certificates

Land Registration Department issues two types of certificates:

- **Land Ownership Certificate:-** This certificate is issued corresponding to each land parcel and provides the ownership detail of a land. This certificate contains the following fields:

  - **Khasra No** - Khasra Number of land parcel
  - **Village** - Name of the village
  - **Sub-District** - Name of the sub-district
  - **District** - Name of the district
  - **State** - Name of the state
  - **Area** - Area of land parcel in sq m
  - **Khata No** - Khata number of the current owner
  - **Owner Name** - Name of the current owner



Digitally Signed by Land Registration Department

Land Record

Khasra No: 2
Village: daultabad
Sub-District: gurgaon
District: gurgaon
State: haryana
Area: 1200 sq m
Khata No: 4
Owner Name: Jane Doe

Figure 23: Land Ownership Certificate

- **Land Transaction Certificate:-** This certificate is issued for each land transaction and provides the land transaction details. This certificate contains the following fields:

  - **Khasra No** - Khasra Number of the land parcel

- **Village** - Name of the village
- **Sub-District** - Name of the sub-district
- **District** - Name of the district
- **State** - Name of the state
- **Transfer Date** - Date on which land transfer takes place
- **Seller Khata No** - Khata number of the seller of the land
- **Seller Name** - Name of the seller of the land
- **Buyer Khata No** - Khata number of the buyer of the land
- **Buyer Name** - Name of the buyer of the land
- **Price** - Selling Price of the land in Indian Rupees



Digitally Signed by Land Registration Department

Land Transfer Record

Khasra No: 2
Village: daultabad
Sub-District: gurgaon
District: gurgaon
State: haryana
Transfer Date: Fri Apr 30 2021
Seller Khata No: 3
Seller Name: John Doe
Buyer Khata No: 4
Buyer Name: Jane Doe
Price: 2000

Figure 24: Land Transfer Certificate

### 3.5.2 Sequence Flow

The sequence flow of the verification server comprises of the following steps:-

1. Get input from the user, which includes the IPFS hash of the certificate and record details.

2. Fetch the corresponding record from the Blockchain network.

3. Validate the provided certificate IPFS hash against the one associated with the record in the Blockchain.

4. If certificate hashes matched, fetch the stored certificate from IPFS.

5. Add digital signature to certificate with the current timestamp and send it to the user.
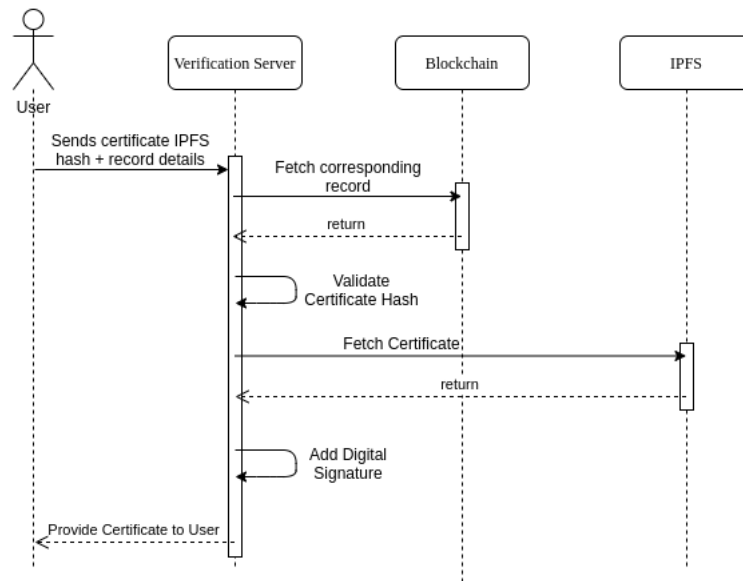


Figure 25: Sequence diagram Verification Server
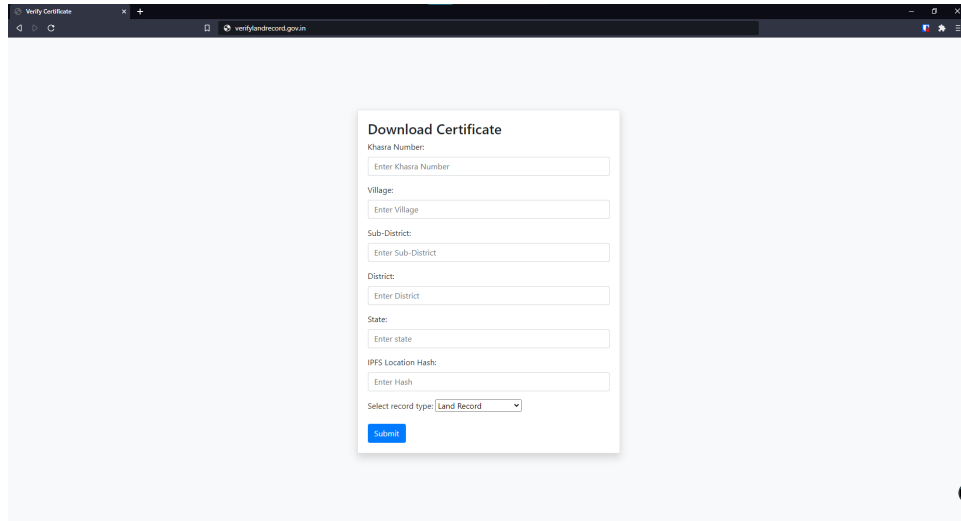
Figure 26: Home Page for Verification Server

## 3.6 Blockchain

All land records and land transfer information is stored on the blockchain. Blockchain is implemented using Hyperledger Fabric.
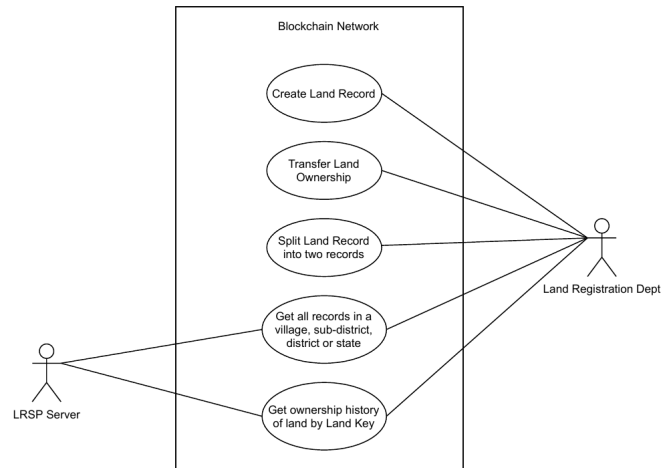


Figure 27: Use Case diagram for Blockchain

Figure 27 shows the various use cases supported by the blockchain. These are implemented by writing smart contracts for each of the five operations shown.

### 3.6.1 Data Format
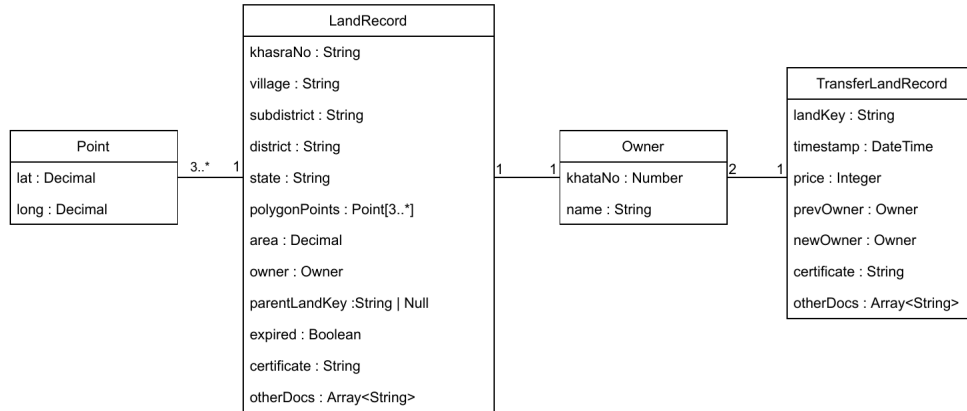


Figure 28: Data Format for Blockchain

**Point** stores coordinates of a particular point and has the following fields:-

- **lat** - Latitude
- **lon** - Longitude

**Owner** stores detail about a land owner and has the following fields:-

- **khataNo** - Khata Number
- **name** - Name of Owner

**LandRecord** stores details about a particular land. Each land record is uniquely identified by a land key which is constructed as - *state:district:subdistrict:village:khasraNo* Following fields are present in a LandRecord:-

- **khasraNo** - Khasra Number
- **village** - Name of the Village
- **subdistrict** - Name of the Sub-District
- **district** - Name of the District
- **state** - Name of the State

- **polygonPoints** - Coordinates of the vertices of the land polygon

- **area** - Area of land parcel in sq m

- **owner** - Current land owner

- **parentLandKey** - In case the land record is formed after splitting from another land record, this stores the land key of the land from which this record was created

- **expired** - Boolean to store whether the record is expired or not; a record is marked as expired when the land parcel is split into two land parcels

- **certificate** - CID of the certificate file stored in IPFS

- **otherDocs** - Array of CID's of the files uploaded to IPFS with the land record

**TransferLandRecord** stores details about a particular land transfer and has the following fields:-

- **landKey** - Land key of the land to which this transfer record belongs

- **timestamp** - UNIX timestamp of the transaction

- **price** - Selling Price of land in Rupees

- **prevOwner** - Seller of the land

- **newOwner** - Buyer of the land

- **certificate** - CID of the certificate file stored in IPFS

- **otherDocs** - Array of CID's of the files uploaded to IPFS with the land transfer record

### 3.6.2   Create Land Record

Creating a land record entails:-

1. Check if land already exists for given Khasra Number, Village, Sub-District, District and State. If yes, throw an error.

2. Create LandRecord data structure with the information provided and store it on the blockchain.
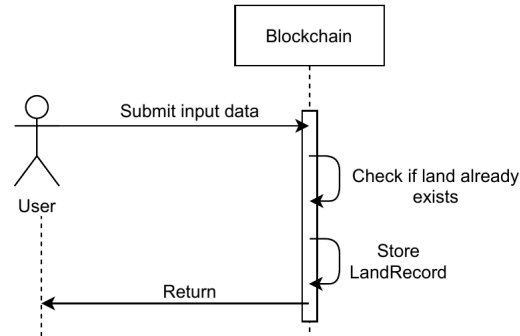
Figure 29: Sequence Diagram for Creating a Land Record

### 3.6.3 Transfer Land Ownership

Transferring land ownership entails:-

1. Get land record stored on the blockchain for given Khasra Number, Village, Sub-District, District and State. If it does not exist, throw an error.

2. Check if the land record is expired. If yes, throw an error.

3. Check if the land's current owner is same as the land seller. If no, throw an error.

4. Update LandRecord data structure with new owner details and store it on the blockchain.

5. Create TransferLandRecord with the information provided and store it on the blockchain.

### 3.6.4 Split Land Record into Two Records

Splitting a land record into two records entails:-

1. Get land record stored on the blockchain for given Khasra Number, Village, Sub-District, District and State for the land record which will be split. If it does not exist, throw an error.

2. Check if that land record is expired. If yes, throw an error.

3. Mark the land record as expired.

4. Create two new LandRecord data structures with the information provided.

5. Update the original land record and add the two new records on the blockchain.
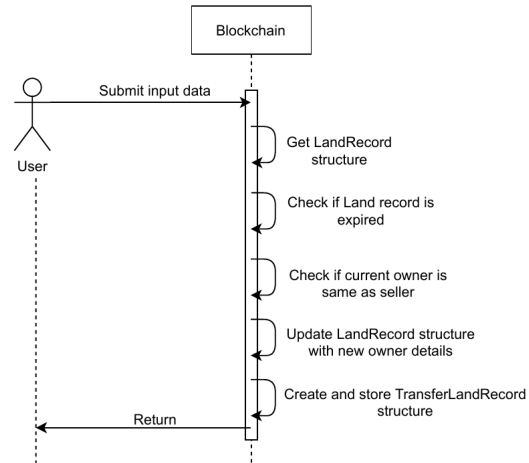
42

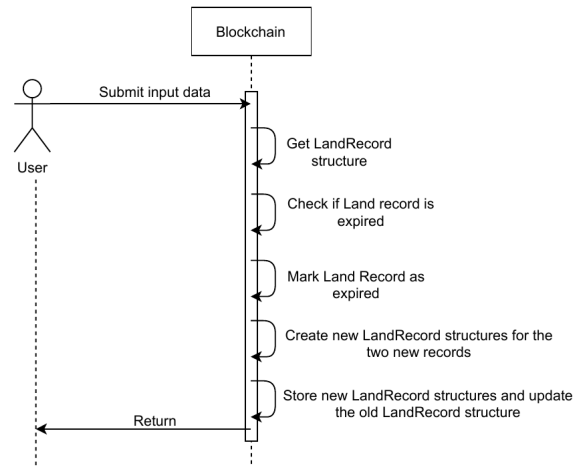Figure 30: Sequence Diagram for Transferring Land Ownership



Figure 31: Sequence Diagram for Splitting a Land Record

### 3.6.5 Get all records in a village/sub-district/district/state

Getting all records in a village/sub-district/district/state entails:-

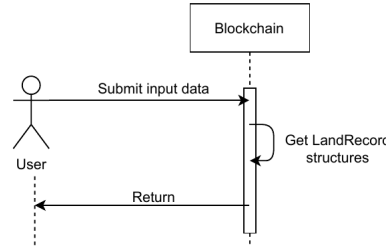1. Get records from the blockchain with the information provided.

2. Return the records.



Figure 32: Sequence Diagram for Getting all Records in a Village/Sub-District/District/State

### 3.6.6 Get ownership history of a land

Getting ownership history of land entails:-

1. Get land record stored on the blockchain for given Khasra Number, Village, Sub-District, District and State. If it does not exist, throw an error.

2. Get all TransferLandRecord data structures stored on the blockchain for the given record.

3. If parentLandKey field of LandRecord is not empty, Get all TransferLandRecord data structures stored on the blockchain for the parent land.

4. Repeat the above steps for each parent land till a LandRecord does not have a parentLandKey field.

## 3.7 Reverse Geocoding Service

Due to the absence of any external service with accurate and up to date village and sub-district data, a dummy service is created. Under the Digital India programme, if such a service is made available, then that can be used in place of this. Reverse Geocoding Service takes GPS coordinates as input and returns the names of State, District, Sub-District and Village where the coordinate belong to. Names and coordinates of places are stored in JSON with the following schema:-
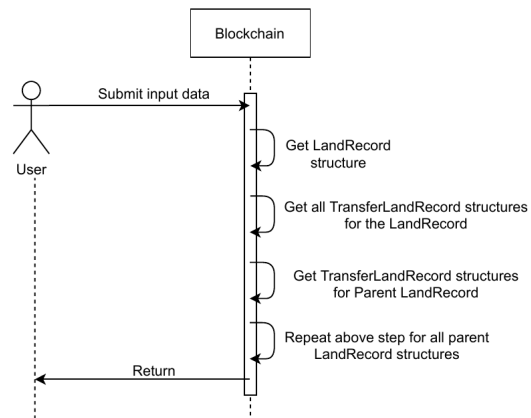
Figure 33: Sequence Diagram for Getting Ownership History of a Land

```
{
    "name": "Name of the place",
    "points": "Array of polygon coordinates of the place",
    "subdivision": "Array of information regarding
    the places in the lower administrative level
    inside this place"
}
```

# 4 Deployment Details

## 4.1 Records Server

Records Server is implemented using NodeJs module Express. It is written in typescript and needs to be transpiled.

Records Server requires the following environment variables:-

- **CERT:-** Path to P12 certificate file used to sign the PDF documents (certificates).

- **IPFS_CLUSTER:-** Link to IPFS cluster node.

```
1       # Install required NodeJs modules
2       npm install
3
4       # Transpile
5       npm run build
6
7       # Start server
8       CERT=<cert_path> IPFS_CLUSTER=<ipfs_cluster_link> node dist/app
    .js
9
```

Listing 1: Records Server

## 4.2 Verification Server

Verification Server is implemented using NodeJs module Express. It is written in typescript and needs to be transpiled.

Verification Server requires the following environment variables:-

- **CERT:-** Path to P12 certificate file used to sign the PDF documents (certificates).

```
1           # Install required NodeJs modules
2           npm install
3
4           # Transpile
5           npm run build
6
7           # Start server
8           CERT=<cert_path> node dist/app.js
9
```

Listing 2: Verification Server

## 4.3  LRRS Server

LRRS Server is implemented using NodeJs module Express. It is written in typescript and needs to be transpiled.

LRRS Server uses the following external services:-

- **MongoDB:-** A running MongoDB instance is required for user authentication purposes. Provide the URI to the MongoDB database in the file *src/setup/db.ts*. To set up a MongodB instance locally, follow this.

- **RazorPay Payment Gateway:-** RazorPay Payment Gateway is used to initiate and verify the payment.

LRRS Server requires the following environment variables :-

- **RZRPAY_KEY_ID:-** ID provided by RazorPay.

- **RZRPAY_KEY_SECRET:-** Secret provided by RazorPay.

```
1       # Install required NodeJs modules
2       npm install
3
4       # Transpile
5       npm run build
6
7       # Start server
8       RZRPAY_KEY_ID=<razorpay_key_id> RZRPAY_KEY_SECRET=<
    razorpay_key_secret> node dist/main.js
9
```

Listing 3: LRRS Server

## 4.4  Smartphone Application

The smartphone application is built using React Native, a NodeJS module, and is written in Typescript. To build Android APK, Linux, Windows and macOS are the supported development OS, but to build iOS IPA, only macOS is supported.

### 4.4.1  Android

To build Android APK, Android Studio is required with the following installed along with it:-

- Android 10 (Q) SDK

- Android SDK Platform 29

- Android Virtual Device

- If not already using Hyper-V: Performance (Intel ® HAXM)

After successful installation of Android Studio, the following environment variable needs to be set :-

- **ANDROID_HOME** to the Android SDK install location

### 4.4.2  iOS

To build iOS IPA, version 10 or a newer version of Xcode is required, which can be installed from the Mac App Store, and after installation, Xcode Command Line Tools needs to installed from the Xcode Preferences tab. Along with Xcode, Cocoapods needs to installed, a dependency manager for Swift and Objective-C Cocoa projects. It can be installed using the command mentioned in Listing 4.

```
1          # Install cocoapods
2          sudo gem install cocoapods
3
```

Listing 4: Cocoapods installation

```
1        # Install required NodeJS modules
2        npm install
3
4        # Start Metro Bundler
5        npx react-native start
6
7        # Build and run Android debug apk
8        npx react-native run-android
9
10       # Build and run iOS debug apk
11       cd ios; pod install; cd ..; npx react-native run-ios
12
13       # Build Android release apk
14       cd android; ./gradlew assembleRelease
15
```

Listing 5: Smartphone Application

To publish the Android APK to the Play Store, a Google Developer account is required. After building a release APK, the same can be uploaded on the Play Store Console from where it is published on the Play Store.

To build a release version of the app for iOS, an Apple developer account is required. After signing in using the developer account in the Xcode, a build button is available, which builds a release IPA for iOS and pushes it to App Store Connect, where the IPA is inspected by Apple for any guideline violation. Upon successful inspection, the app can be published to the App Store using the App Store Connect Console.

## 4.5 IPFS

A private IPFS network is set up. IPFS-Cluster is used to manage the cluster. This requires two environment variables to be set:-

- **SWARM_KEY:-** Required for private IPFS network, can be generated by following point 2 in this link.

- **CLUSTER_SECRET:-** Required to manage the private IPFS cluster

These can be provided in a .env file in the same directory as the docker-compose.yml file. Make sure that the init.sh file can be executed.

```
1         # Modify init.sh file permissions to allow execution
2         chmod 740 init.sh
3
4         # In local development environment, IPFS network can be started
   by using the command:-
5         docker-compose up
6
```

Listing 6: IPFS

## 4.6 Blockchain

Smart contract and applications that interact with blockchain run on the test net using the commercial paper example replacing the smart contract and applications with our implementations.

One can follow the tutorial in the example linked and replace the smart contract with the implementation on the Github repo for our project.

**LRRS server** runs as an organization 1(digibank) application.
**Verification Server** and **Records Server** run as organization 2(magnetocorp) applications.

# 5 Conclusion

## 5.1 Conclusion

The traditional land record retrieval system has many shortcomings, as discussed in Chapter one. To address these shortcomings, a blockchain-based solution is proposed. The proposed solution takes into account various features of Blockchain technology like immutability, security, transparency, etc., to overcome the shortcomings present in the traditional land record retrieval system. The proposed solution makes it convenient to reconcile and synchronize data across various government departments, thus eliminating discrepancies in data. The proposed solution facilitates the retrieval and verification of ownership history of land parcel and associated documents digitally. To validate the integrity of a land record, the user can request a digitally signed certificate using the Verification portal. This eliminates the need for cumbersome processes and delays in the current system to verify the land ownership of a land parcel. The blockchain-based solution, being immutable and transparent, makes the land data tamper-proof. This also checks any possible corruption in government departments.

## 5.2 Future Work

In the proposed solution, we are using Hyperledger Fabric [16] which is a private permissioned Blockchain. This entails configuring and deploying our own distributed network infrastructure for production usage. Various components need to be set up and configured to deploy the solution at the production level. This includes certificate authorities (CA), Membership Service Providers(MSP), ordering services, peer and ordering nodes, etc. The current system has been tested on dummy data. The land information needs to be digitized and fed into the system database to use the system in a real scenario.

For transfer of land, the proposed solution requires the transfer to be manually entered into the blockchain records by the Land Registration Department. This can be improved to remove the dependency on Land Registration Department and allow users to directly sell and buy land without the intermediate regulatory authority.

Land Registration Department Record's portal can be further improved to add additional metadata for every record entered into the blockchain, such as which user added a particular record into the blockchain.

# Abbreviations

**CID** Content Identifier. 12, 17, 20, 41

**LRRS** Land Records Retrieval Service. 5, 6, 22, 26, 28, 30–32, 47, 49

# References

[1] M.T. Ozsu and P. Valduriez. *Principles of Distributed Database Systems.* Springer, New York, 3rd edition, 2011.

[2] C. Date. *An Introduction to Database Systems. Vols. I and II.* Addison-Wesley Publishing Co., 4th edition, 1987.

[3] Mirela Liliana Moise Nicoleta Magdalena Iacob (Ciobanu). Centralized vs. distributed databases. case study. *Academic Journal of Economic Studies*, 4:119–130, 2015.

[4] Korth H.F. Silberschatz, A. and S. Sudarshan. *Database System Concepts.* McGraw-Hill, 6th edition, 2010.

[5] Juan Benet. Ipfs - content addressed, versioned, p2p file system. 07 2014.

[6] Klaus Wehrle, Stefan Götz, and Simon Rieche. 7. distributed hash tables. volume 3485, pages 79–93, 01 2005.

[7] Department of land resources.

[8] Centre of excellence in blockchain technology.

[9] Vinay Thakur, M.N. Doja, Yogesh K. Dwivedi, Tanvir Ahmad, and Ganesh Khadanga. Land records on blockchain for implementation of land titling in india. *International Journal of Information Management*, 52:101940, 2020.

[10] Christopher Mellon J. Michael Graglia. Blockchain and property in 2018: At the end of the beginning. 2018.

[11] Emma Sahlin and Rebecka Levenby. Blockchain in audit trails : An investigation of how blockchain can help auditors to implement audit trails. 2018.

[12] Qiuyun Shang and Allison Price. A Blockchain-Based Land Titling Project in the Republic of Georgia: Rebuilding Public Trust and Lessons for Future Pilot Projects. *Innovations: Technology, Governance, Globalization*, 12(3-4):72–78, 01 2019.

[13] Joon Ian Wong. Sweden's blockchain-powered land registry is inching towards reality. 2017.

[14] H. Mukne, P. Pai, S. Raut, and D. Ambawade. Land record management using hyperledger fabric and ipfs. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–8, 2019.

[15] Kazi Masudul Alam, J.M. Ashfiqur Rahman, Anisha Tasnim, and Aysha Akther. A blockchain-based land title management system for bangladesh. *Journal of King Saud University - Computer and Information Sciences*, 2020.

[16] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolic, and Jason Yellick. Hyperledger fabric: A distributed operating system for permissioned blockchains. 01 2018.