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
April 26, 2021

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 display information and collect information from the user.
- **Business Logic Layer:** In this layer, information collected from 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, this is where databases come into 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 databases 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 and Administration which includes allowing access to various users, enforcing data security and various other checks. [1] A database model is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized and manipulated. The most popular 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 oldest and most important database design. A relational database that fails to follow the ACID model cannot be considered reliable.

Databases can 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 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 in different points (nodes of a network of computers) and logically related by functional relations so that they can be viewed globally as a single database [2]. 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 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 [3]: local autonomy; the absence of a dependency from a central location; continuous operation; location independent; fragmentation independent; replication independent; distributed query processing; distributed transaction management; hardware independent; operating system independent; independent of communication infrastructure; independent of database management system.

If a centralized database is used to store the data then the whole system is prone to 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 vulnerability. This makes centralized database dubious with very low reliability. But at the same time, centralized database ensures data consistency and easy management in compliance with ACID design [4].

A major advantage of distributed database is that by sharing a database across multiple nodes can obtain a storage space extension and also can 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 compared to a centralized database as it doesn't have a single point of failure, if a node fails another node or group of nodes can provide the necessary data. But 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 compared to a centralized database [4]. 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 [5].

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 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 to get 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 read and write operations in contrast to traditional databases which supports read, write, update and delete 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 not only can view the data already committed to the blockchain but can also 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 blockchain ledger in sync with each other and after consensus is reached, the whole network logically act as one single public ledger. This removes the complications present in traditional systems where different participants or organizations maintains multiple ledgers which needs to be reconcile and synchronized time to time.

Systems that are deployed on public blockchain network are more cost efficient as compared to the traditional systems. This is because the system utilizes the processing power and resources of a large number of 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 vulnerability, than traditional sytems. 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 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 the data present in the blockchain.

1.4 Motivation

In India, currently if 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 provided fake documents. As these documents are mostly maintained offline as hard copies these cases occur often.

Getting all the information such as the sale deeds and ownership history for a land parcel is very 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 very easy and hassle free.

Indian government launched Digital India Land Record Modernization Programme in 2008. Main aim of this programme is 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 [6]. In addition to this, 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 [7].

1.5 Proposed Solution

To tackle the problem of accessing and verifying the land ownership records, this paper proposes a smartphone based solution. Using which users can request land ownership records for a particular land parcel by taking the advantage of the GPS module available in smartphones. These land ownership records are stored in a permissioned blockchain and to make it backwards compatible i.e. migrating old records to the new system, scanned copies of the documents can be stored in permissioned IPFS cluster with their Content Identifier (CID) stored on the same blockchain. Now whenever users want the ownership history of a land parcel they use the smartphone application, mark the piece of land and after successful completion of the process they get a PDF report with the ownership history of that said land parcel. This PDF report also contains verification codes for each ownership record they hold, these verification codes can then be used on a verification portal to generate a digitally signed PDF verifying that the given information is true. This solution hence provides a trustful and hassle free mechanism for people to access and verify land ownership records.

2 Chapter 4

2.1 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

```
# Install required NodeJs modules
npm install

# Transpile
npm run build

# Start server
CERT=<cert_path> IPFS_CLUSTER=<ipfs_cluster_link> node dist/app
.js

```

Listing 1: Records Server

2.2 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)

```
        # Install required NodeJs modules
        npm install

        # Transpile
        npm run build

        # Start server
        CERT=<cert_path> node dist/app.js

```

Listing 2: Verification Server

2.3 LRSP Server

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

LRSP 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.
- **Reverse Geocoding Service :-** The server is using **MapMyIndia** Reverse geocoding service. Provide the MapMyIndia API Key in the file *src/services/coordResolver.ts*
- **RazorPay Payment Integration :-**RazorPay Payment service is used to initiate and verify the payment.

LRSP server requires the following environment variables :-

- **RZRPAY_KEY_ID**
- **RZRPAY_KEY_SECRET**

```
# Install required NodeJs modules
npm install

# Transpile
npm run build

# Start server
RZRPAY_KEY_ID=<razorpay_key_id> RZRPAY_KEY_SECRET=<
razorpay_key_secret> node dist/main.js

```

Listing 3: LRSP Server

2.4 Smartphone Application

The smartphone application is built using React Native a NodeJS module, it 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.

2.4.1 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

2.4.2 iOS

To build iOS IPA, version 10 or newer 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 which is a dependency manager for Swift and Objective-C Cocoa projects. It can be installed using the command mentioned in Listing 4.

```
# Install cocoapods
sudo gem install cocoapods

```

Listing 4: Cocoapods installation

```
# Install required NodeJS modules
npm install

# Start Metro Bundler
npx react-native start

# Build and run Android debug apk
npx react-native run-android

# Build and run iOS debug apk
cd ios; pod install; cd ..; npx react-native run-ios

# Build Android release apk
cd android; ./gradlew assembleRelease

```

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.

2.5 IPFS

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

- **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.

```
# Modify init.sh file permissions
chmod 740 init.sh

# In local development environment, network can be started by using the command:-
docker-compose up

```

Listing 6: IPFS

2.6 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.

LRSP server run as organization 1(digibank) application.
Verification Server and **Records Server** run as organization 2(magnetocorp) application.

References

[1] Wikipedia.

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

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

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

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

[6] Department of land resources.

[7] Centre of excellence in blockchain technology.