COURT LEDGER-DECENTRALIZED AND TAMPER-PROOF SOLUTION FOR STORING EVIDENCE

A PROJECT REPORT

Submitted by

THARUN KUMAR T	**211419205308**
HEMNATH V	**211419205304**
YOGESH Y	**211419205309**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PANIMALAR ENGINEERING COLLEGE, POONAMALLEE

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **COURT LEDGER-DECENTRALIZED AND TAMPER-PROOF SOLUTION FOR STORING EVIDENCE** is the bonafide work of **THARUN KUMAR T (21141205308), HEMNATH V (211419205304), YOGESH Y (211419205309)** who carried out the project under my supervision.

SIGNATURE	**SIGNATURE**
Dr. M. HELDA MERCY M.E., Ph.D.,	**Mrs. BENITHACHRISTINAL.J M. TECH., (Ph.D.,)**
	ASSISTANT PROFESSOR
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Information Technology	Department of Information Technology
Panimalar Engineering College	Panimalar Engineering College
Poonamallee, Chennai - 600 123	Poonamallee, Chennai - 600 123

Submitted for the project and viva-voce examination held on ______________

SIGNATURE	**SIGNATURE**
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

I hereby declare that the project report entitled **COURT LEDGER-DECENTRALIZED AND TAMPER-PROOF SOLUTION FOR STORING EVIDENCE** which is being submitted in partial fulfilment of the requirement of the course leading to the award of the 'Bachelor Of Technology in Information Technology' in **Panimalar Engineering College, Autonomous institution Affiliated to Anna university- Chennai** is the result of the project carried out by me under the guidance **in the Department of Information Technology**. I further declared that I or any other person has not previously submitted this project report to any other institution/university for any other degree/ diploma or any other person.

THARUN KUMAR T

HEMNATH V

Date**:**

YOGESH Y

Place: Chennai

It is certified that this project has been prepared and submitted under my guidance.

Date: BENITHA CHRISTINAL.J, M. TECH.,(Ph.D.,)

Place: Chennai (ASSISTANT PROFESSOR / IT)

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion . We wish to express our sincere thanks to all those who were involved in the completion of this project.

Our sincere thanks to **Our Honorable Secretary and Correspondent, Dr. P. CHINNADURAI, M.A., Ph.D.,** for his sincere endeavor in educating us in his premier institution.

We would like to express our deep gratitude to **Our Dynamic Directors , Mrs. C. VIJAYA RAJESHWARI and Dr. C. SAKTHI KUMAR, M.E.,M.B.A., Ph.D.,** and **Dr. SARANYA SREE SAKTHI KUMAR.,B.E.,M.B.A., Ph.D.,** for providing us with the necessary facilities for completion of this project.

We also express our appreciation and gratefulness to **Our Principal Dr. K. MANI, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our head of the department, **Dr. M. HELDA MERCY, M.E., Ph.D.,** Department of Information Technology, for her support and by providing us ample time to complete our project.

We express our indebtedness and gratitude to our Project co-ordinator **Mr. M. DILLI BABU**, **M.E.,(Ph.D.,)** Associate Professor, Department of Information Technology for his guidance throughout the course of our project. We also express sincere thanks to our supervisor **Mrs. BENITHA CHRISTINAL.J, M.TECH**., (**Ph.D.,) Assistant Professor** for providing the support to carry out the project successfully. Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

ABSTRACT

Court Ledger is an open-source blockchain application created to modernize and protect the administration of court data. The blockchain's immutability and invulnerability are put to use here to keep court documents incorruptible and easily accessible to legitimate users. Court Ledger eliminates the need for time-consuming and prone-to-error manual record-keeping by keeping track of and updating court records in real time. Furthermore, the system protects sensitive information while allowing authorized users (such as judges, lawyers, and litigants) access to court documents. Court Ledger has the ability to completely change the way courts handle their case files by providing a streamlined, easy-to-use platform for doing so. This abstract introduces the Court Ledger system, describing its features and describing its possible uses in the context of court record administration.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABB	**FULL FORM**
COC	Chain OF Custody
IPFS	Interplanetary File System
ACL	Access Control List
SSL	Secure Socket Layer
TLS	Transport Layer Security
JS	Java Script
RAM	Random Access Memory
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
UML	Unified Modeling Language
DFM	Design For Manufacturing
ER	Entity Relationship

CHAPTER-1

INTRODUCTION

1.1 Introduction

As we reap the full benefits of a data-driven society, the public has become increasingly concerned about the protection of users' personal information. Centralized organizations - both public and private - accumulate large amounts of personal and sensitive information. People have little to no control over what data is stored about them and how it is used. Over the past few years, public media have covered controversial privacy incidents on several occasions. Some of the best-known examples include the history of government surveillance and Facebook's large-scale scientific experiment, which apparently did not explicitly inform participants.

1.2 Overview Of Big Data In Court Ledger

The volume of data in our world is expanding rapidly. A recent report estimates that 20% of global data has been collected over the last several years. Facebook, the largest online social network, has collected 300 petabytes of personal data since its creation - about 100 times the amount that the Library of Congress has collected in more than 200 years. In the age of big data, data is continually collected and analyzed, fostering innovation and economic growth. Businesses and organizations use the data they gather to tailor services, optimize business decision-making, predict future trends and more. Data is a valuable asset to our economy these days. According to the Forbes report, 2.5 quintillion bytes of data are generated each day. Of the global data set, more than 90% of the data has been generated in the past two years. With such massive data growth, cloud storage is necessary for storing data.

Most of the data currently available on the Internet is highly centralized and stored with a handful of tech companies that have the experience and capital to do so to build massive data centers capable of dealing with this huge data. The challenge with that approach is data security. Because these data are stored centrally, if an attacker can access the server, they can easily view and change the data. A further problem related to this approach is the confidentiality of user data. In many cases, these data are used by third parties for data analytics and marketing purposes. Furthermore, the costs incurred to store data in centralized servers are increasing. The users have to pay for the whole plan which they did so even though they used only a fraction of the storage. Therefore, it does not give the user the flexibility to pay for just what they use. Another problem is the scalability of the system, it is difficult to adapt a centralized storage system to meet the growing demand. With zero confidence, two sides can perform transactions in Blockchain.

1.3 Challenges

One of the biggest challenges in digital forensic science is evidence management. Digital evidence plays a huge role in criminal investigations because it connects people to criminal activity. It is of the utmost importance to ensure the integrity, authenticity and auditor of digital cameras as it operates at different hierarchical levels, i.e. a first responder to senior authorities responsible for managing cybercrime investigations. Digital evidence brings with it its own chain of possession challenges. The ability of blockchain technology to segment the complete reading of the original return transactions provides a monumental promise for the rhetoric community. In essence, it is distributed information that sustains an ever-increasing number of tamper-resistant blocks that hold batches of individual transactions. It implements a decentralized registry fully replicated as an annex only in a peer-to-peer network, initially deployed for the Bitcoin cryptocurrency. All participating nodes keep a complete local copy of blockchain up-to-date. The blockchain is a set of blocks containing general ledger transactions. Transactions in blocks are sorted chronologically and each block contains a crypto hash for the previous block in the chain. Each block includes a timestamp and a data link pointing to an anterior block. Blockchain has four elements that are replicated: ledger, cryptographic, consensus and business logic. The Hyperledger is a project of the Linux Foundation that is a collaborative open-source effort created to advance cross-sectoral Blockchain technologies. The COC is widely used as evidence, to be acceptable before a court or in court proceedings, it must be proven that it was not altered during investigations. The issues of the chain of custody become very important because the authenticity of the evidence must be maintained in accordance with the condition once it has been first discovered until it is presented to the court. Thus, a decent COC method ought to use typical procedures for dealing and handling evidence (digital or not), in spite of whether or not the proof is utilized in an attempt or not.

1.4 Requirements

The primary requirements for a COC process are:

Integrity: Throughout the transfer, the evidence has not been altered or corrupted.

Traceability: Evidence must be derived from the time it is collected to the point it is ruined.

Authentication: Any witness who interacts with the associated evidence should present an associated positive sign as recognizable evidence of their identity.

Auditability: The entire methodology should be verifiable with each relevant entity within the methodology.

Security: Tampering proof - Changes in evidence may not be modified or corrupted.

Based on these characteristics, we identify an architecture that can support the chain of command process. In particular, we intend to offer a non-public permission blockchain and apply a judicious contract to keep track of changes in possession throughout the lifecycle of the evidence.

Blockchain is a registry of incorruptible digital transactions that can be programmed to record not just economic transactions, but almost all value transactions. Simply put, a blockchain is a time-stamped series of immutable data that is handled by a group of computers that are not the property of a single entity. Cryptocurrencies and blockchain applications have been one of the fastest growing domains of computing in recent years, resulting in a high demand for software applications. Blockchain is beginning to be explored gradually by various sectors looking to take advantage of the accuracy and pace that technology can offer.

1.5 Approach

The traditional digital forensics process is integrated with cloud technologies, often known as "cloud forensics". Simply put, cloud forensics is a combination of digital forensics and cloud-based forensics. Cloud Forensics is a digital forensics application that identifies crimes committed on the cloud and carries out the required investigation with a minimum of overhead and complications. Various categories of digital forensics such as computational forensics, network forensics, mobile device forensics, digital image forensics, Digital audio crime scene and memory crime scene have been developed over the years. Traditional medico-legal investigative processes are less effective and efficient because of decentralized data processing. In order to overcome these limitations of the traditional process, digital forensic science is integrated into the cloud. The evolution of the cloud presents a number of challenges and problems, especially in digital and criminal investigations. The investigative process for any type of platform consists of several phases such as- problem identification, data collection, crime scene review, and review of the investigation and presentation of the conclusion of the cases. Studies by different researchers have shown that implementing cloud-based digital forensic science is complex and there are many issues and challenges at play at every stage of cloud forensic science.

Some of the problems with cloud forensics are access to newspapers, collecting stable data, huge amounts of data, reconstructing the crime scenario, multinational laws, bringing evidence to court, and so on. Solutions such as log keeping, a separate plan for cloud-based data recovery, and legislative solutions are proposed in the document. As cloud-based forensic science is an increasingly interesting area of research, our focus is on conducting in-depth analysis of issues and challenges of cloud forensics based on the existing literature to present an analytic review. Some recent scientific articles from well-known academic databases are taken into consideration. The blockchain, a linear data structure, allows all interested parties to create a digital log to document and store the transactions (events/ files) exchanged on a distributed network of computers. This Blockchain structure has the potential to ensure the security and confidentiality of digital evidence for the conduct of cloud-based forensic science.

CHAPTER-2

LITERATURE SURVEY

[1] Users' privacy is a growing problem as our data-driven society begins to provide its full rewards. Large volumes of personal and sensitive data are stored in centralized institutions, both public and private. Individuals have almost little say over how their personal information is used or preserved. There have been other instances over the past few years where public media have covered problematic privacy events. Some of the most well-known instances are the long tradition of government spying and Facebook's massive scientific experiment, about which users were presumably not given full disclosure.

-[2] These past several years have seen widespread implementations of blockchain technology. Additionally, further possibilities for practical use are continuously being explored. Blockchain is predicated on and enables the actualization of the Decentralized Application (DApps). This allows for increased openness, decentralization, and adaptability inside the applications. Due to its complexity, blockchain integration poses unique challenges that need for specialized knowledge beyond what is typically used in app development. In this light, this article details our efforts to create a decentralized application (DApp) using Ethereum, one of the most widely used blockchain platforms.

[3] In order to facilitate private, pseudo-anonymous transactions, blockchain technology combines the benefits of immutability, consensus, and transparency. Decentralized applications (Daps) may now interact with the blockchain in a programmed manner thanks to smart contracts, which are constructed on top of the blockchain to allow for on-chain storage. Programmable blockchains have gained traction in the healthcare sector as a possible answer to addressing major issues such information gaps, slow clinical report delivery, and splintered patient data. This article presents assessment parameters for gauging the viability, intended capabilities, and compliance of blockchain-based Daps in the healthcare sector.

[4] While cloud services have made it much simpler to share files online, concerns have been raised about the security risks involved in storing sensitive information with a third party. Centralized cloud service provider management has historically resulted in security concerns due to clients' limited confidence in the provider as a third party. It's also a pain because it makes it harder to easily share information over the internet. In this work, blockchain technology is used to provide decentralized security administration and improved service quality. Moreover, Ciphertext-Policy Attribute Based Encryption is shown as a practical method for achieving granular control over who may access what in the stored data. Meanwhile, the security audit verifies that cloud-based information is kept private and unaltered. We then assess the computational overhead of our system and its impact on its performance.

[5] Smart contracts are used in the management, banking system, insurance, estate, Internet of Things, and other industries today because of their ability to conduct trustworthy and reversible transactions without the need for a third party. In this article, we show how Hyperledger Fabric, a kind of Permission Blockchain, may be used to make a smart contract. Hyperledger Explorer for Fabric 1.4.x on Ubuntu Linux has been used to display the tests.

[6] Many critical applications are designed on the distributed structure using the blockchain technology to ensure the availability, immutability, and security. However, these applications are facing the storage problem owing to the data volume growth of transaction. The number of transactions and its size in a block is growing in the blockchain day by day because of the feature of immutability and append-only. The growing nature of transactions in a block is not only making the problem for storage but also in access to the block transactions. In this paper, we propose an IPFS based blockchain storage model to solve the

storage problem of transaction in a block along with access of transaction of a particular block. In the propose storage model, the miners stores transaction on IPFS distributed file system storage and get the returned IPFS hash of transaction into the block of the blockchain. The feature of the IPFS network and its resultant hash reduce the size of transactions in a block. To secure access of transaction for a particular block content-addressed (IPFS hash) storage technique has been proposed. We have applied this scheme on a transaction which includes image storage on IPFS and hash storage into the blockchain. In this paper, we have also proposed the content-addressed technique in contrast to the location addressed for the access of transaction. To implement the framework we have used anaconda python, python flask, and IPFS.

[7] Blockchain is unquestionably a ground-breaking idea. To put it simply, this Technology is the foundation of bitcoin and other cryptocurrencies. Despite people's singular emphasis on blockchain, cryptocurrencies are already being used in daily businesses to make instant, third-party-free online payments in an effort to replace the tedious and antiquated practice of using cash. When individuals are willing to put their faith and money into the distributed ledger technology known as blockchain, it may be a useful tool for a wide range of businesses. Smart contracts, which are self-executing apps in the Ethereum world, make up the backbone of the blockchain and come with their own set of security concerns. This zero-trust network may be used to replace many of the contested processes or activities we face on a daily basis. The safety of an electronic voting system is a major issue for us. Blockchain is a transparent, immutable, append-only ledger that makes manipulation impossible. In this article, we show how to build a voting software that uses Ethereum network smart contracts and electronic wallets for voting. The Ethereum blockchain will ultimately store records of votes and voters, providing a transparent and trustworthy network where mistreatment is minimized after an election has taken place.

[8] With the proliferation of online resources, ensuring the secure transfer, distribution, and preservation of information have emerged as critical issues. This article proposes a data sharing platform under a new technological environment, combining the decentralized and irreversible properties of the Ethereum blockchain with the distributed storage technology of IPFS. This will guarantee data security, user rights protection, and rapid data processing. Provide a platform for modern information exchange based on cutting-edge application technology.

[9] Blockchains are inefficient for storing large files. A major drawback is that when more information is added to the blockchain, the amount of data that must be shared across nodes increases. Yet if the node operator doesn't need to see every file saved on the blockchain, the blockchain's widespread replication wastes a lot of storage space for no apparent reason. More data needs to be processed, transported, and stored, driving up the cost of running blockchain nodes. To store and distribute massive files more quickly and easily, IPFS may be used as a file sharing system. A blockchain is not required because it uses cryptographic hashes as its basis. Nevertheless, IPFS does not provide selective file sharing. Whenever sharing private or sensitive information, this is a must. For this reason, this article introduces an updated version of IPFS, one that makes use of Ethereum smart contracts to enable permissioned file sharing. The ACL is managed by the smart contract, and is enforced by the upgraded IPFS client software. To achieve this, it communicates with the smart contract anytime a file is moved, copied, or renamed. This experimental setting examines the effects of IPFS with restricted access and discusses the results.

[10] Blockchain is a new system for distributing and transacting data securely among a distributed group of individuals who may or may not be trustworthy to one another. It paves the way for novel distributed software architectures in which consensus may be reached on shared states without having to rely on a centralized

integration point. The complexity of blockchain technology in its various permutations is a significant challenge for architects developing blockchain-based application frameworks. The blockchain industry is still in its infancy, thus there is a lack of both product data and trustworthy technical evaluation that may be used to compare different blockchains. In this work, we offer a method for categorizing and comparing blockchains and blockchain-based systems to facilitate the design and assessment of their influence on software architectures. Our classification system reflects the broad architectural features of blockchains and the effects of their key design choices. The purpose of this taxonomy is to aid in making critical design decisions about the performance and quality characteristics of blockchain-based systems.

CHAPTER-3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

User Interface: The system would likely have a user interface that allows authorized users to upload, view, and search for evidence files. The interface may be web-based, with a modern design and intuitive navigation.

Database: The system would store evidence files and related metadata in a centralized database. The database would likely be scalable and designed to handle large volumes of data, with proper backup and recovery procedures in place.

Security: The system would need to be secure, with appropriate authentication, access control, and encryption measures in place to prevent unauthorized access or tampering. It may use SSL/TLS encryption to secure communications between the web application and the user's browser.

Integration with existing systems: The system may need to integrate with existing court systems or other third-party systems, such as e-discovery tools or case management systems, to facilitate data sharing and retrieval.

Compliance: The system would need to comply with relevant legal and regulatory requirements, such as data protection laws and court rules of evidence.

Testing: The system would need to be tested thoroughly, using a combination of manual and automated testing techniques, to ensure that it is functioning correctly and meeting all requirements.

Maintenance and Support: The system would require ongoing maintenance and support to ensure that it remains secure, reliable, and up-to-date. This may involve regular updates, bug fixes, and security patches, as well as technical support for users.

3.2 PROPOSED SYSTEM

In order to implement Court Ledger, a number of crucial parts would need to be in place. To begin, a blockchain network would be utilized because of its immutability and tamper-proof nature, both of which are essential in the management of court data.

After that, the system would provide a safe way for court staff to enter data and administer case files. There would be less room for human mistake and unnecessary paperwork if court documents could be tracked and updated in real time using this interface.

A secure access control method would enable only authorized users, such as judges, lawyers, and litigants, to see case files, protecting the confidentiality of sensitive data.

In addition, the system would promote openness in the administration of court records by enabling authorized parties to monitor the status and revisions of court documents.

Strong encryption techniques to safeguard sensitive information and the implementation of suitable security measures to prevent unauthorized access and cyber-attacks are required to guarantee the security and integrity of the system.
In order to ensure the safety and efficiency of court record keeping, the proposed

system for Court Ledger would employ a blockchain network, secure interfaces for court personnel and authorized parties, real-time tracking and updates, access control mechanisms, transparency, and appropriate security measures.

3.3 FEASIBILITY STUDY

With an eye towards gauging the project's viability and improving server performance, a business proposal defining the project's primary goals and offering some preliminary cost estimates is offered here. Your proposed system's viability may be assessed once a comprehensive study has been performed. It is essential to have a thorough understanding of the core requirements of the system at hand before beginning the feasibility study. The feasibility research includes mostly three lines of thought:

- Economical feasibility
- Technical feasibility
- Operational feasibility
- Social feasibility

3.3.1 ECONOMICAL FEASIBILITY

The study's findings might help upper management estimate the potential cost savings from using this technology. The corporation can only devote so much resources to developing and analyzing the system before running out of money. Every dollar spent must have a valid reason. As the bulk of the used technologies are open-source and free, the cost of the updated infrastructure came in far cheaper than anticipated. It was really crucial to only buy customizable products.

3.3.2 TECHNICAL FEASIBILITY

This research aims to establish the system's technical feasibility to ensure its smooth development. Adding additional systems shouldn't put too much pressure on the IT staff. Hence, the buyer will experience unnecessary anxiety. Due to the low likelihood of any adjustments being necessary during installation, it is critical that the system be as simple as possible in its design.

3.3.3 OPERATIONAL FEASIBILITY

An important aspect of our research is hearing from people who have actually used this technology. The procedure includes instructing the user on how to make optimal use of the resource at hand. The user shouldn't feel threatened by the system, but should instead see it as a necessary evil. Training and orienting new users has a direct impact on how quickly they adopt a system. Users need to have greater faith in the system before they can submit constructive feedback.

3.3.4 SOCIAL FEASIBILITY

During the social feasibility analysis, we look at how the project could change the community. This is done to gauge the level of public interest . Because of established cultural norms and institutional frameworks, it's likely that a certain kind of worker will be in low supply or nonexistent.

3.4 REQUIREMENT SPECIFICATION

3.4.1 HARDWARE REQUIREMENTS

Processor	: Pentium Dual Core 2.00GHZ
Hard disk	: 120 GB
RAM	: 2GB (minimum)
Keyboard	: 110 keys enhanced

3.4.2 SOFTWARE REQUIREMENTS

Operating system	: Windows7 (with service pack 1), 8, 8.1 and 10
Language	: Java Script
Technology	:Block Chain
Network	:Meta Mask, Bloxberg

3.5 LANGUAGE SPECIFICATION– JAVA SCRIPT

Java Script (JS) is a light weight , interpreted , or just-in-time compiled programming language with first-class function. While it is most well known as the scripting language for web pages, many non browser environment also use it, such as Node.js, Apache CouchDB and Adobe Acrobat.
In blockchain the usage of java script is more , its easy and flexible to connect Ethereum ,Solana ,Meta mask.
When developing a blockchain it is essential to use tools that you are confident with. A large portion of developers use JavaScript as a programming language, which may seem unfair since JavaScript in blockchain networks is not well documented. But you can in fact use JavaScript when creating a blockchain.

Using java script the interconnection of meta mask blockchain with web pages is so easy and flexible and also access of multiple modules is more flexible.

Initially, JavaScript was called "Live Script" but later its name got changed from to JavaScript to sound more familiar. The main motive behind building JavaScript was to make web pages lively. Those were the times when there used to be no fancy web pages hence it was the need of the hour. The web pages without JavaScript are called "Static" web pages while those with JavaScript are "Dynamic" web pages. Dynamic web pages are very user-friendly and intuitive.

- **Less server interaction:** You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.

- **Immediate feedback to the visitors:** They don't have to wait for a page reload to see if they have forgotten to enter something.
- **Increased interactivity**: You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

- **Richer interfaces:** You can use JavaScript to include such items as drag-and drop components and sliders to give a Rich Interface to your site visitors.

- **Java Script And Blockchain Simplicity:** Performing functions will become simple with blockchain. This will enable an easy and cost-efficient approach to mobile app development. Feature-rich apps without complexities can be created under reasonable pricing and with only moderate efforts.

- **Security**: Web Applications developed with blockchain technology are considered much more safer. As the most reliable and advanced cryptography is used, the highest level of safety is ensured here.

- **Reliability:** Reliability of wen pages when adds with java script and blockchain are about to increase significantly, The structure of blockchain itself being reliable and robust, the complete system is secured against any potential crash or collapse. Data exists in more than one place. This makes blockchain technology and java script all the more reliable.

- **Transparency:** Every transaction is recorded so that users are able to track these, when and as they intend. There’s no possibility of fabricated information or fraudulent transaction here. Valuable data is protected and customers’ trust is earned when blockchain technology is used.

CHAPTER-4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

This graphic provides a concise and understandable description of all the entities currently integrated into the system. The diagram shows how the many actions and choices are linked together. You might say that the whole process and how it was carried out is a picture. The figure below shows the functional connections between various entities.

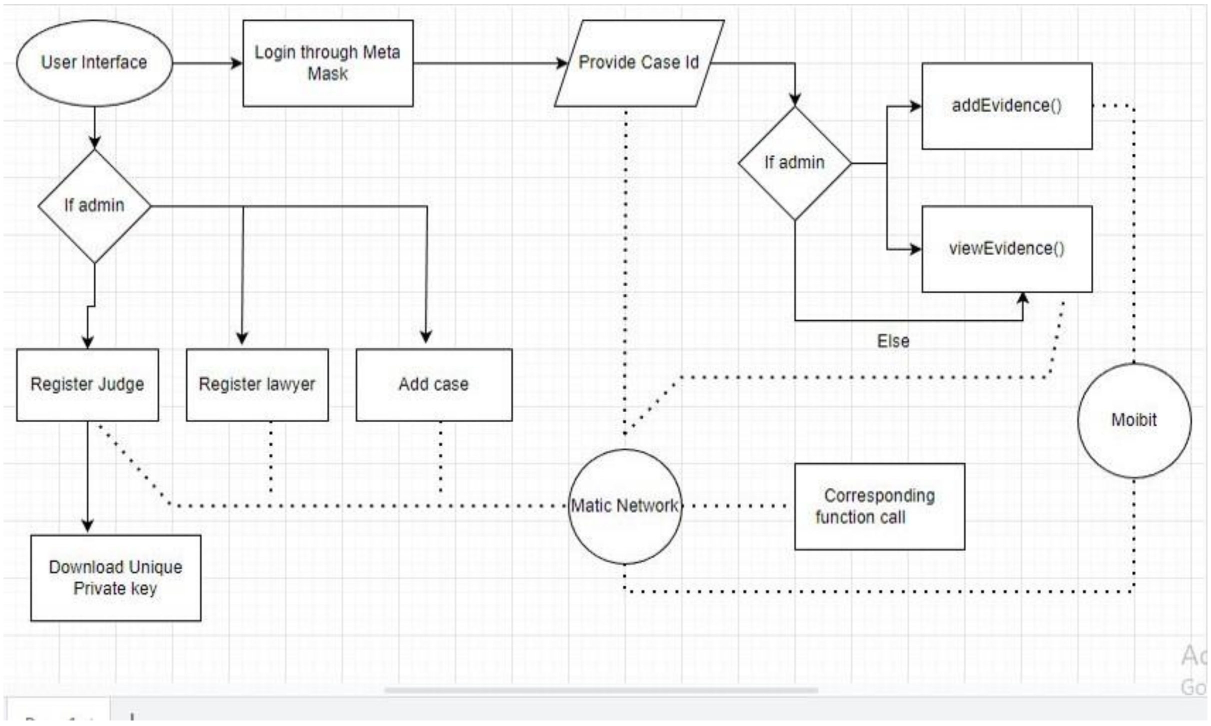


Fig 4.1 – Architecture Diagram

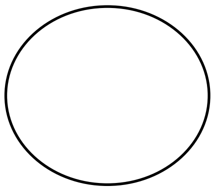
4.2 DATA FLOW DIAGRAM

To illustrate the movement of information throughout a procedure or system, one might use a Data-Flow Diagram (DFD). A data-flow diagram does not include any decision rules or loops, as the flow of information is entirely one-way. A flowchart can be used to illustrate the steps used to accomplish a certain data-driven task. Several different notations exist for representing data-flow graphs. Each data flow must have a process that acts as either the source or the target of the information exchange. Rather than utilizing a data-flow diagram, users of UML often substitute an activity diagram. In the realm of data-flow plans, site-oriented data-flow plans are a subset. Identical nodes in a data-flow diagram and a Petri net can be thought of as inverted counterparts since the semantics of data memory are represented by the locations in the network. Structured data modeling (DFM) includes processes, flows, storage, and terminators.

Data Flow Diagram Symbols

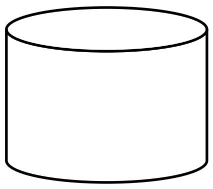
Process

A process is one that takes in data as input and returns results as output.



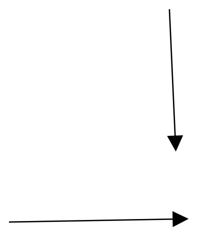
Data Store

In the context of a computer system, the term "data stores" is used to describe the various memory regions where data can be found. In other cases, "files" might stand in for data.



Data Flow

Data flows are the pathways that information takes to get from one place to another. Please describe the nature of the data being conveyed by each arrow.



External Entity

In this context, "external entity" refers to anything outside the system with which the system has some kind of interaction. These are the starting and finishing positions for inputs and outputs, respectively.

DATA FLOW DIAGRAM

The whole system is shown as a single process in a level DFD. Each step in the system's assembly process, including all intermediate steps, are recorded here. The "basic system model" consists of this and 2-level data flow diagrams.

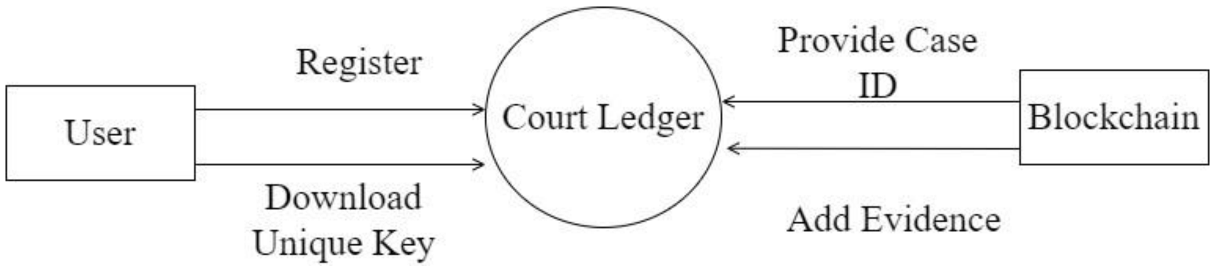


Fig 4.2 – Data Flow Diagram

4.3 ENTITY RELATIONSHIP DIAGRAM

Definition

The relationships between database entities can be seen using an entity-relationship diagram (ERD). The entities and relationships depicted in an ERD can have further detail added to them via data object descriptions. In software engineering, conceptual and abstract data descriptions are represented via entity-relationship models (ERMs). Entity-relationship diagrams (ERDs), entity-relationship diagrams (ER), or simply entity diagrams are the terms used to describe the resulting visual representations of data structures that contain relationships between entities. As such, a data flow diagram can serve dual purposes. To demonstrate how data is transformed across the system. To provide an example of the procedures that affect the data flow.

1. One-to-One

Whenever there is an instance of entity (A), there is also an instance of entity (B) (B). In a sign-in database, for instance, only one security mobile number (S) is associated with each given customer name (A) (B).

2. One-to-Many

For each instance of entity B, there is exactly one occurrence of entry A, regardless of how many instances of entity B there are.

For a corporation whose employees all work in the same building, for instance, the name of the building (A) has numerous individual associations with employees (B), but each of these B's has only one individual link with entity A.

3. Many-to-Many

For each instance of entity B, there is exactly one occurrence of entry A, regardless of how many instances of entity B there are.

In a corporation where everyone works out of the same building, entity A is associated with many different Bs, but each B has only one A.

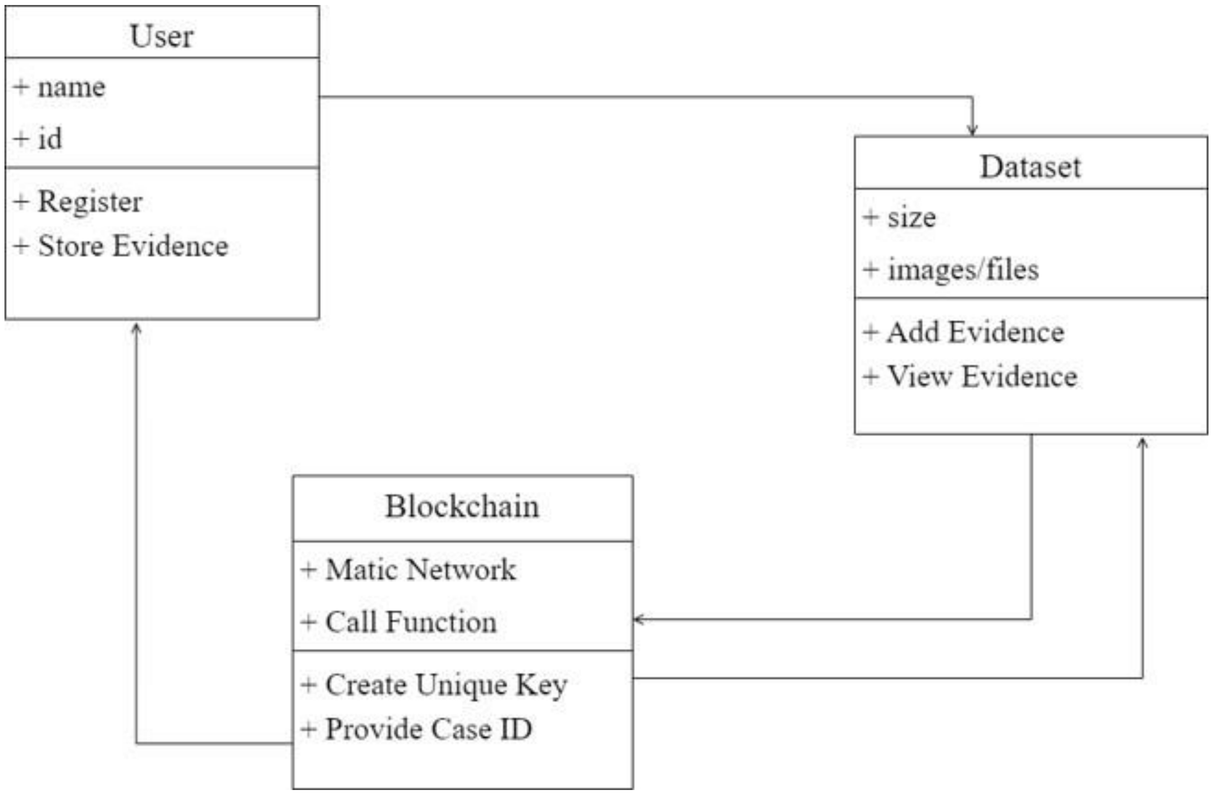


Fig 4.3 – Entity Relationship Class Diagram

4.4 USE-CASE DIAGRAM

The possible interactions between the user, the dataset, and the algorithm are often depicted in a use case diagram. It's created at the start of the procedure.

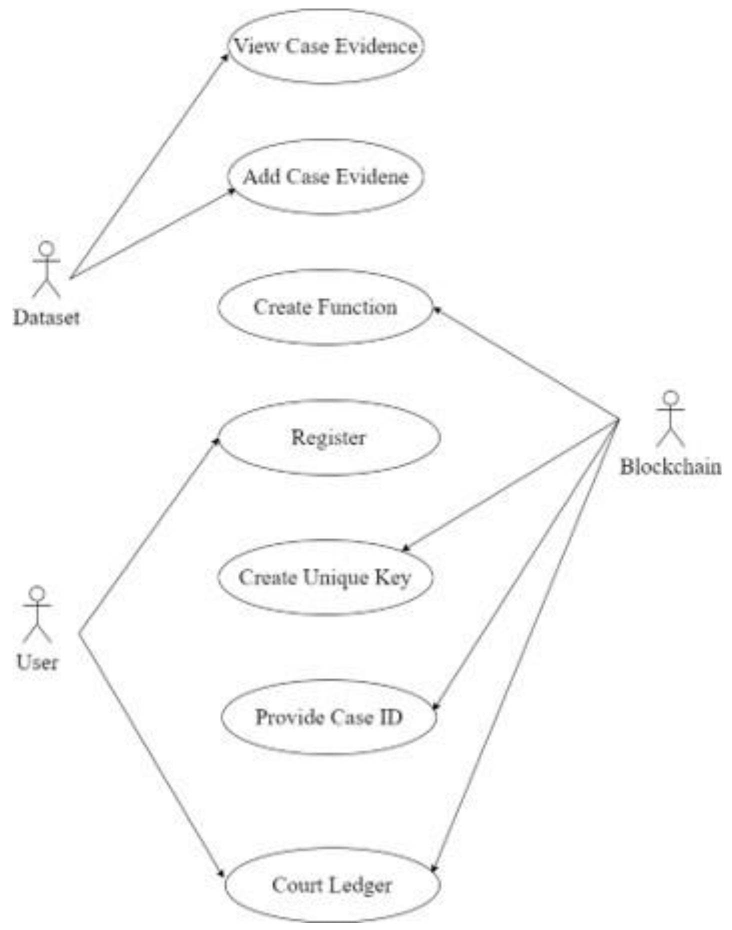


Fig 4.4– Use-Case Diagram

4.5 ACTIVITY DIAGRAM

An activity diagram, in its most basic form, is a visual representation of the sequence in which tasks are performed. It depicts the sequence of operations that make up the overall procedure. They are not quite flowcharts, but they serve a comparable purpose.

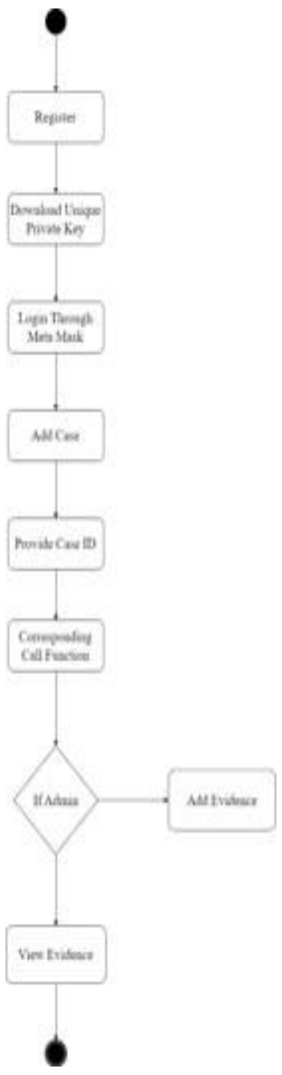


Fig 4.5– Activity Diagram

4.6 SEQUENCE DIAGRAM

These are another type of interaction-based diagram used to display the workings of the system. They record the conditions under which objects and processes cooperate.

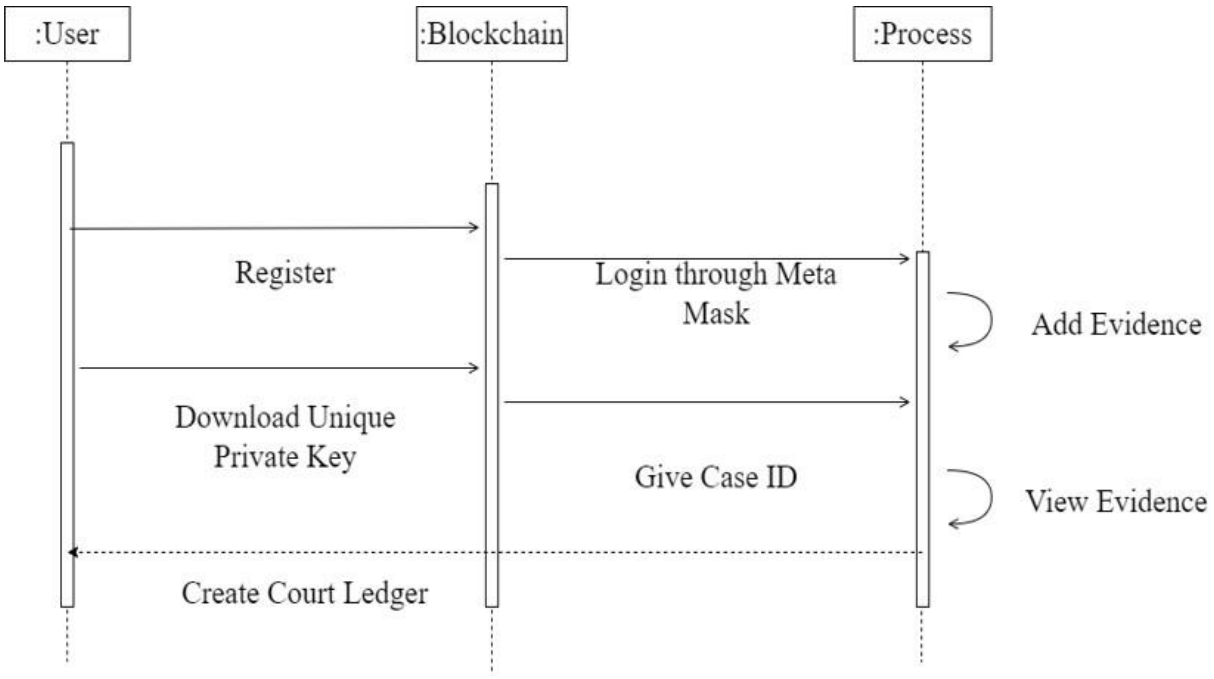


Fig 4.6 – Sequence Diagram

4.7 CLASS DIAGRAM

In essence, this is a "context diagram," another name for a contextual diagram. It simply stands for the very highest point, the 0 Level, of the procedure. As a whole, the system is shown as a single process, and the connection to externalities is shown in an abstract manner.

- A + indicates a publicly accessible characteristic or action.
- A - a privately accessible one.
- A # a protected one.
- A - denotes private attributes or operations.

4.8 ER DIAGRAM

The abbreviation ER refers to a connection between two entities. The entities used and saved in the database are shown in relationship diagrams. They break down the process into its component parts and explain how they work. Attributed concepts, Relationship concepts, and Entity concepts are the building blocks for these kinds of diagrams.

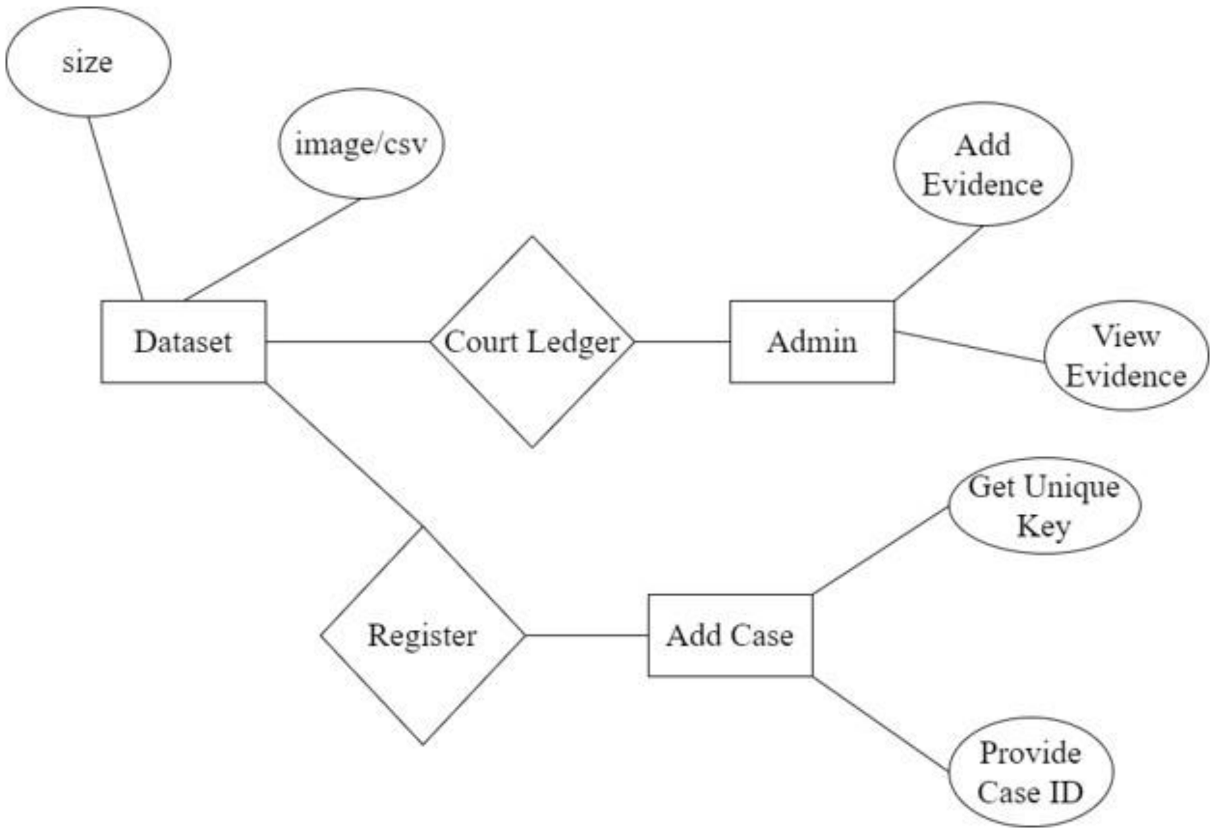


Fig 4.7 – ER Diagram

CHAPTER-5

MODULE DESCRIPTION

5.1 MODULE 1: EVIDENCE STORAGE

Our system relies heavily on the Evidence Storage Module, which provides a streamlined, distributed, and tamper-proof means of preserving evidence for use in legal processes. Court hearings, evidence, and other important documents will be archived in digital form on Moibits decentralized storage and made available to authorized users using this module.

Each piece of evidence that is kept on the system must be completely safe from tampering, and this module was made to guarantee that. The Ethereum blockchain, which serves as an immutable and transparent log of all transactions and data saved on the network, is integral to this goal.

Controlling who may view what in the evidence vault is another function of the module. The platform allows the admin, judges, and attorneys who have been given access to examine specific cases to log in and view those cases. This module adds another level of protection to the evidence by limiting access to it to those who need it.

The Evidence Storage Module is an integral part of our system, as it provides a safe and unalterable place to keep and retrieve evidences for use in legal processes.

5.2 MODULE 2: BLOCKCHAIN INTEGRATION

Our system relies heavily on the Blockchain Integration Module, which provides a straightforward, distributed, and tamper-proof means of archiving evidence for use in legal processes. This component is in charge of connecting the Ethereum blockchain to the platform so that all transactions and data can be tracked and verified without any chance of alteration.

Using smart contracts, the module stores and manages information on the blockchain in a way that prevents unauthorized parties from changing or erasing it. The module's auditable history of all transactions further facilitates monitoring and vetting the legitimacy of the evidence kept on the system.

Moreover, the Blockchain Integration Module handles all bitcoin transactions required to access the Ethereum blockchain. This feature makes it simple to monitor and control the platform's monetary inflows and outflows by ensuring the safety and openness of all bitcoin transactions.

In sum, the Blockchain Integration Module is crucial to the platform's safety and openness. Integrating the Ethereum blockchain, this module makes the platform's data immutable and transparent, making it a more reliable and efficient way of keeping evidence for legal procedures.

5.3 MODULE 3: USER MANAGEMENT

Our method for preserving evidence for legal processes in a way that is easy, distributed, and tamper-proof relies heavily on the User Management Module. The admin, the judges, and the attorneys are the primary users of the platform, and all three of these groups must be registered and authenticated in order to have access to and control the evidences.

This component facilitates a safe and straightforward account creation procedure for the users of the system. Only those who have been given permission to examine the evidences kept on the platform will be able to see them after logging into the system using their unique credentials.

Controlling who can view what in the database is another function of the User Management Module. In this way, the module adds another degree of protection to the evidence by limiting access to it to those who need it. Comparable to the current process, but with far better security, the admin has the ability to put cases and evidence into the cases.

Our system relies heavily on the User Management Module, which facilitates the registration and authentication of authorized users, as well as the management of access control to the stored evidence, in an easy and secure manner. A more reliable and efficient method of keeping evidence for legal procedures is provided by this module, which restricts access to it to just those who need it.

CHAPTER-6

TESTING

Discovering and fixing such problems is what testing is all about. The purpose of testing is to find and correct any problems with the final product. It's a method for evaluating the quality of the operation of anything from a whole product to a single component. The goal of stress testing software is to verify that it retains its original functionality under extreme circumstances. There are several different tests from which to pick. Many tests are available since there is such a vast range of assessment options.

Who Performs the Testing: All individuals who play an integral role in the software development process are responsible for performing the testing. Testing the software is the responsibility of a wide variety of specialists, including the End Users, Project Manager, Software Tester, and Software Developer.

When it is recommended that testing begin: Testing the software is the initial step in the process. begins with the phase of requirement collecting, also known as the Planning phase, and ends with the stage known as the Deployment phase. In the waterfall model, the phase of testing is where testing is explicitly arranged and carried out. Testing in the incremental model is carried out at the conclusion of each increment or iteration, and the entire application is examined in the final test.

When it is appropriate to halt testing: Testing the programme is an ongoing activity that will never end. Without first putting the software through its paces, it is impossible for anyone to guarantee that it is completely devoid of errors. Because the domain to which the input belongs is so expansive, we are unable to check every single input.

6.1 TYPES OF TESTING

There are four types of testing:

Unit Testing

The term "unit testing" refers to a specific kind of software testing in which discrete elements of a program are investigated. The purpose of this testing is to ensure that the software operates as expected.

Test Cases

1. Test for data encryption: Ensure that all data stored in the court ledger is encrypted and can only be accessed by authorized users with the necessary permissions.

2. Test for data integrity: Verify that the court ledger is tamper-proof by testing that all data stored in the ledger is immutable and cannot be altered once it has been added to the ledger.

3. Test for data availability: Ensure that the court ledger is available and accessible at all times, and can withstand high volumes of traffic and usage.

Integration Testing

The programme is put through its paces in its final form, once all its parts have been combined, during the integration testing phase. At this phase, we look for places where interactions between components might cause problems.

Test Cases

1. Test integration with existing court systems: Verify that the court ledger can integrate with existing court systems seamlessly, allowing for easy access and retrieval of stored evidence.

2. Test integration with blockchain: Verify that the court ledger integrates with the blockchain technology that is used to store and secure data.

3. Test integration with storage solutions: Ensure that the court ledger integrates with different types of storage solutions, such as cloud-based storage, to provide flexibility and scalability.

Functional Testing

One kind of software testing is called functional testing, and it involves comparing the system to the functional requirements and specifications. In order to test functions, their input must first be provided, and then the output must be examined. Functional testing verifies that an application successfully satisfies all of its requirements in the correct manner. This particular kind of testing is not concerned with the manner in which processing takes place; rather, it focuses on the outcomes of processing. Therefore, it endeavours to carry out the test cases, compare the outcomes, and validate the correctness of the results.

Test Cases

1. Test for adding evidence: Verify that the court ledger can accept evidence files and store them securely, with proper encryption and data integrity checks.

2. Test for retrieving evidence: Ensure that authorized users can easily retrieve evidence from the court ledger, with proper authentication and access control measures in place.

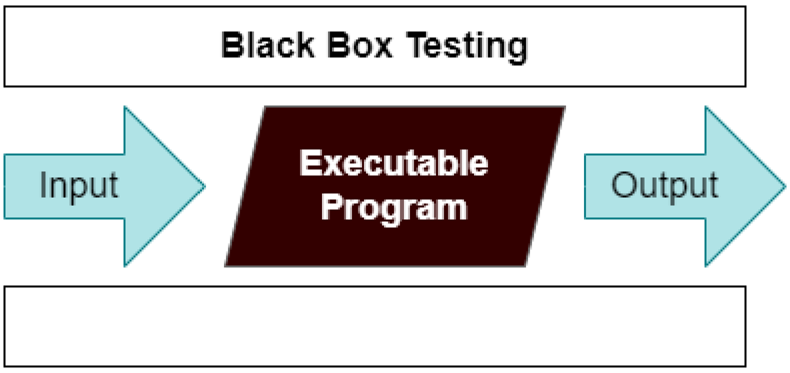
3. Test for searching evidence: Verify that the court ledger provides a comprehensive search function that allows authorized users to quickly and easily find specific evidence files.

6.2 TESTING TECHNIQUES

There are many different techniques or methods for testing the software, including the following:

BLACK BOX TESTING

During this kind of testing, the user does not have access to or knowledge of the internal structure or specifics of the data item being tested. In this method, test cases are generated or designed only based on the input and output values, and prior knowledge of either the design or the code is not necessary. The testers are just conscious of knowing about what is thought to be able to do, but they do not know how it is able to do it.



For example, without having any knowledge of the inner workings of the website, we test the web pages by using a browser, then we authorise the input, and last, we test and validate the outputs against the intended result.

Test Cases

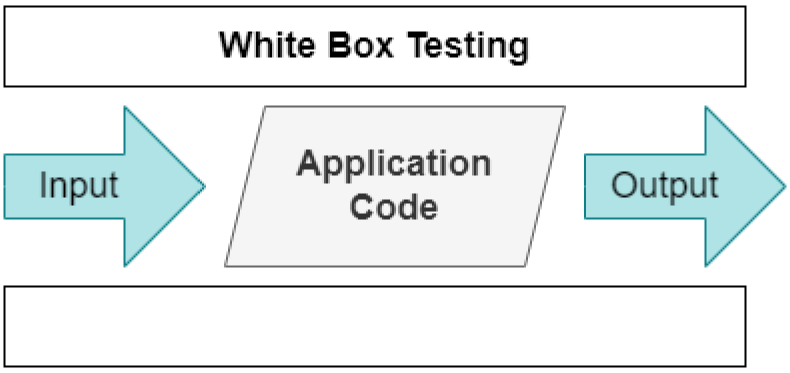
1. Test for user interface: Verify that the user interface of the court ledger is intuitive and user-friendly, with all necessary features easily accessible.

2. Test for compatibility: Ensure that the court ledger is compatible with different types of devices, browsers, and operating systems.

3. Test for accessibility: Verify that the court ledger meets accessibility standards, allowing users with disabilities to access the system.

WHITE BOX TESTING

During this kind of testing, the user is aware of the internal structure and details of the data item, or they have access to such information. In this process, test cases are constructed by referring to the code. Programming is extremely knowledgeable of the manner in which the application of knowledge is significant. White Box Testing is so called because, as we all know, in the tester's eyes it appears to be a white box, and on the inside, everyone can see clearly. This is how the testing got its name.



As an instance, a tester and a developer examine the code that is implemented in each field of a website, determine which inputs are acceptable and which are not, and then check the output to ensure it produces the desired result. In addition, the decision is reached by analyzing the code that is really used.

Test Cases

1. Test for data encryption: Verify that all data stored in the court ledger is encrypted using industry-standard encryption algorithms, with proper key management and storage.

2. Test for data integrity: Ensure that the court ledger uses cryptographic techniques, such as hash functions, to ensure that data stored in the ledger is tamper-proof and cannot be altered.

3. Test for smart contract logic: If the court ledger uses smart contracts, verify that the smart contracts are coded correctly, with no vulnerabilities that could be exploited by attackers.

CHAPTER-7

CONCLUSION

7.1 CONCLUSION

As a result, Court Ledger is an encouraging blockchain-based system that aims to enhance the administration of court records by enhancing their openness, accessibility, and safety. Court Ledger's use of blockchain technology ensures that any data entered into its system cannot be altered, which improves the reliability of court records and lessens the likelihood of mistakes. The proposed Court Ledger system has safeguards in place to protect the privacy and security of users' personal information, as well as real-time tracking and updates, access control mechanisms, transparency, and other safety features. Court Ledger has the ability to completely change the way court records are managed, drastically increasing productivity while decreasing the overhead of retaining paper documents. Court Ledger may facilitate more expedient and accurate legal procedures by eliminating the need for physical document management by providing authorized parties with instant, secure access to court records. As a whole, Court Ledger is an exciting new tool with the potential to revolutionize the legal sector by facilitating the safe and effective administration of court records and enhancing their openness and accessibility to all authorized parties.

CHAPTER-8

APPENDIX 1

CODING:

Encryption

Ethcrypto.js

```
const EthCrypto = require('eth-crypto');

// create identitiy with key-pairs and address
const alice = EthCrypto.createIdentity();
// console.log(alice.publicKey)
const secretMessage = 'My name is Satoshi Buterin';

encrypt = async function (_alice, message){
   return await EthCrypto.encryptWithPublicKey(_alice.publicKey, message)
}
decrypt = async function (_alice, _encrypted) {
   return await EthCrypto.decryptWithPrivateKey(_alice.privateKey,
_encrypted);
}



async function run(_alice, _secretMessage) {
   let encrypted = await encrypt(alice, _secretMessage);
   console.log(encrypted)
   let decrypted = await decrypt(alice, encrypted);
   console.log(decrypted);
}

run(alice, secretMessage)
```

Add Case

Addcase.js

```
import React, { Component } from "react";

import { Form, Icon, Input, Button, Checkbox } from "antd";
import styles from "./styles/addCase.module.css";
import "antd/dist/antd.css";
import loading from "../loading.gif";
class AddCase extends Component {
  handleSubmit = e => {
    e.preventDefault();
    var lg = document.body.querySelector("#loadinggif");
    lg.style = "display:inline";
    var p = this.props.passableItems;
    this.props.form.validateFields((err, values) => {
      if (!err) {
        console.log("Received values of form: ", values);
        this.newCase(
          values.JudgeId,
          values.Lawyer1Id,
          values.Lawyer2Id,
          values.Party1,
          values.Party2,
          values.Details,
          p
        );
      }
    });
  };

  async newCase(
    judgeId,
    lawyer1Id,
    lawyer2Id,
    party1name,
    party2name,
    details,
    p
  ) {
    const { account, court, GAS, GAS_PRICE } = p;
    var h3 = document.body.querySelector("#lawyerId");
    h3.style = "display:none";
    await court.methods
      .newCase(judgeId, lawyer1Id, lawyer2Id, party1name, party2name, details)
      .send({ from: account, gas: GAS, gasPrice: GAS_PRICE })
      .then(r => {
        console.log(r);
        // get Case ID
        h3.style = "display:block";
        var lg = document.body.querySelector("#loadinggif");
        lg.style = "display:none";
        this.getValue(court);
      })
      .catch(e => {
        console.log(e);
      });
  }
  getValue = async court => {
    var events = await court.events
      .caseCreated({ fromBlock: 0 })
      .on("data", event => {
        this.setState({ lawyerId: event.returnValues._caseId });
      })
      .on("changed", function (event) {
        console.log("NEWWW", event);
      })
      .on("error", console.error);
  };
  async checkAuth(props) {
    // console.log(props)
    const { account, court } = this.props.passableItems;
    // console.log("HII", account, court);
    var owner;
    await court.methods.owner().call((err, res) => {
      owner = res;
    });
    // console.log(owner)
    if (owner === account) {
      this.setState({ auth: true });
    }

  }
```

```
constructor(props) {
  super(props);
  this.state = {
    account: "",
    loading: true,
    auth: false
  };
}
async componentDidMount() {
  await this.checkAuth(this.props);
}
render() {

  const { getFieldDecorator } = this.props.form;
  return (
    this.state.auth ? (<div className={styles.parentContainer}>
      <div className={styles.containerForm}>
        <Form
          onSubmit={this.handleSubmit}
          className="login-form"
          style={{ margin: "2em" }}
        >
          <h2> Add Case </h2>
          <Form.Item>
            {getFieldDecorator("JudgeId", {
              rules: []
            })(
              <Input
                prefix={
```

```
      <Icon type="user" style={{ color: "rgba(0,0,0,.25)" }} />
    }
    placeholder="Judge Id"
  />
 )}
</Form.Item>
<Form.Item>
 {getFieldDecorator("Lawyer1Id", {
  rules: []
 })(
  <Input
    prefix={
      <Icon type="user" style={{ color: "rgba(0,0,0,.25)" }} />
    }
    placeholder="Lawyer 1 ID"
  />
 )}
</Form.Item>
<Form.Item>
 {getFieldDecorator("Lawyer2Id", {
  rules: []
 })(
  <Input
    prefix={
      <Icon type="user" style={{ color: "rgba(0,0,0,.25)" }} />
    }
    placeholder="Lawyer 2 ID"
  />
 )}
```

```
</Form.Item>
<Form.Item>
  {getFieldDecorator("Party1", {
    rules: []
  })(
    <Input
      prefix={
        <Icon type="user" style={{ color: "rgba(0,0,0,.25)" }} />
      }
      placeholder="Party 1"
    />
  )}
</Form.Item>
<Form.Item>
  {getFieldDecorator("Party2", {
    rules: []
  })(
    <Input
      prefix={
        <Icon type="user" style={{ color: "rgba(0,0,0,.25)" }} />
      }
      placeholder="Party2 Name"
    />
  )}
</Form.Item>
<Form.Item>
  {getFieldDecorator("Details", {
    rules: []
  })(
    <Input
      prefix={
        <Icon type="user" style={{ color: "rgba(0,0,0,.25)" }} />
      }
      placeholder="Details"
    />
  )}
</Form.Item>

<Form.Item className={styles.formBottom}>
  <Button
    type="primary"
    htmlType="submit"
    className="login-form-button"
  >
    Proceed to Add the case
  </Button>
  <br />
  <img
    style={{ display: "none" }}
    id="loadinggif"
    src={loading}
    alt=""
    height="100px"
  />
  <br />
</Form.Item>
<h3 id="lawyerId" style={{ display: "none" }}>
  Your Case Id is: {this.state.lawyerId}
```

```
          </h3>
         </Form>
        </div>
      </div>) : (<h1> YOU ARE NOT AN ADMIN</h1>)
    );
  }
}

const WrappedNormalAddCaseForm = Form.create({ name: "add_case"
})(AddCase);
export default WrappedNormalAddCaseForm;
```

Register Judge

Regjudge.js

```
import React, { Component } from "react";
import { Form, Icon, Input, Button, Checkbox } from "antd";
import styles from "./styles/register.module.css";
import "antd/dist/antd.css";
import loading from "../loading.gif";
const EthCrypto = require("eth-crypto");

class Register extends Component {
  async registerJudge(name, phone, email, address, pubkey, p) {
    const { account, court, GAS, GAS_PRICE } = p;
    console.log(court);
    await court.methods
      .registerJudge(name, phone, email, address, pubkey)
      .send({ from: account, gas: GAS, gasPrice: GAS_PRICE })
      .then(r => {
        console.log(r);
      var h3 = document.body.querySelector("#lawyerId");
      h3.style = "display:block";
      var lg = document.body.querySelector("#loadinggif");
      lg.style = "display:none";
      this.getValue(court);
    });
  }

  getValue = async court => {
    var events = await court.events
      .judgeRegistered({ fromBlock: 0 })
      .on("data", event => {
        this.setState({ lawyerId: event.returnValues._judgeId });
      })
      .on("changed", function(event) {
        console.log("NEWWW", event);
      })
      .on("error", console.error);
  };

  downloadPrivateKey(_blobData) {
    var blob = new Blob([_blobData + "\n" + "keep this key saved"], {
      type: "text/plain"
    });
    let url = window.URL.createObjectURL(blob);
    var a = document.createElement("a");
    document.body.appendChild(a);
    a.style = "display:none";
  a.href = url;
  a.download = "private_key";
  a.click();
  document.body.removeChild(a);
  // document.location.reload();
}

handleSubmit = e => {
  e.preventDefault();
  var lg = document.body.querySelector("#loadinggif");
  lg.style = "display:inline";
  var p = this.props.passableItems;
  this.props.form.validateFields((err, values) => {
    if (!err) {
      console.log("Received values of form: ", values);
      const person = EthCrypto.createIdentity();
      console.log("add public key to contract", person.publicKey);
      var name = values.Name;
      var email = values.Email;
      var phone = values.Phone_number;
      var address = p.account;
      var pubk = person.publicKey;
      // console.log(name, email, phone, address)
      this.registerJudge(name, phone, email, address, pubk, p);

      this.downloadPrivateKey(person["privateKey"]);
    }
  });
};
```

```
constructor(props) {
  super(props);
  this.state = {
    account: "",
    loading: true
  };
}

render() {
  const { getFieldDecorator } = this.props.form;
  return (
    <div className={styles.parentContainer}>
      <div className={styles.containerForm}>
        <h1>Register Judge</h1>
        <Form
          onSubmit={this.handleSubmit}
          className="login-form"
          style={{ margin: "2em" }}
        >
          <Form.Item>
            {getFieldDecorator("Name", {
              rules: []
            })(
              <Input
                prefix={
                  <Icon type="user" style={{ color: "rgba(0,0,0,.25)" }} />
                }
                placeholder="Name"
              />
            )}
          </Form.Item>
          <Form.Item>
            {getFieldDecorator("Email", {
              rules: []
            })(
              <Input
                prefix={
                  <Icon type="mail" style={{ color: "rgba(0,0,0,.25)" }} />
                }
                placeholder="Email"
              />
            )}
          </Form.Item>
          <Form.Item>
            {getFieldDecorator("Phone_number", {
              rules: []
            })(
              <Input
                prefix={
                  <Icon type="number" style={{ color: "rgba(0,0,0,.25)" }} />
                }
                placeholder="Phone Number"
              />
            )}
          </Form.Item>
          <Form.Item>
            {getFieldDecorator("Address", {
              rules: []
            })(
              <Input
                prefix={
                  <Icon type="lock" style={{ color: "rgba(0,0,0,.25)" }} />
                }
                placeholder={`Eth address: ${this.props.passableItems.account}`}
                disabled
              />
            )}
          </Form.Item>
          {/* <Form.Item>
                          {getFieldDecorator("District", {
                              rules: [

                              ]
                          })(
                              <Input
                                  prefix={
                                      <Icon
                                          type="info"
                                          style={{ color: "rgba(0,0,0,.25)" }}
                                      />
                                  }
                                  placeholder="District"
                              />
                          )}
                      </Form.Item> */}
          {/* <Form.Item>
```

```
            {getFieldDecorator("PublicKey", {
              rules: [

              ]
            })(
              <Input
                prefix={
                  <Icon
                    type="lock"
                    style={{ color: "rgba(0,0,0,.25)" }}
                  />
                }
                placeholder="Public Key"
              />
            )}
          </Form.Item> */}

        <Form.Item className={styles.formBottom}>
          <Button
            type="primary"
            htmlType="submit"
            className="login-form-button"
          >
            Proceed to Add the user
          </Button>
          <br />
          <img
            style={{ display: "none" }}
            id="loadinggif"
```

```
              src={loading}
              alt=""
              height="100px"
            />
            <br />
          </Form.Item>
          <h3 id="lawyerId" style={{ display: "none" }}>
            Your Judge Id is: {this.state.lawyerId}
          </h3>
        </Form>
      </div>
    </div>
  );
 }
}

const WrappedNormalRegisterForm = Form.create({ name: "register"
})(Register);
export default WrappedNormalRegisterForm;
```

Register lawyer

Reglawyer.js

```
import React, { Component } from "react";
import { Form, Icon, Input, Button, Checkbox } from "antd";
import styles from "./styles/register.module.css";
import "antd/dist/antd.css";
import loading from "../loading.gif";
const EthCrypto = require("eth-crypto");
```

```
class Register extends Component {
  async registerLawyer(name, phone, email, address, pubkey, p) {
    const { account, court, GAS, GAS_PRICE } = p;
    console.log(court);
    await court.methods
      .registerLawyer(name, phone, email, address, pubkey)
      .send({ from: account, gas: GAS, gasPrice: GAS_PRICE })
      .then(r => {
        console.log(r);
        //get Lawyer Id
        var h3 = document.body.querySelector("#lawyerId");
        h3.style = "display:block";
        this.getValue(court);
        var lg = document.body.querySelector("#loadinggif");
        lg.style = "display:none";
      });
  }

  getValue = async court => {
    var events = await court.events
      .lawyerRegistered({ fromBlock: 0 })
      .on("data", event => {
        this.setState({ lawyerId: event.returnValues._lawyerId });
      })
      .on("changed", function (event) {
        console.log("NEWWW", event);
      })
      .on("error", console.error);
  };
```

```
downloadPrivateKey(_blobData) {
  var blob = new Blob([_blobData + "\n" + "keep this key saved"], {
    type: "text/plain"
  });
  let url = window.URL.createObjectURL(blob);
  var a = document.createElement("a");
  document.body.appendChild(a);
  a.style = "display:none";
  a.href = url;
  a.download = "private_key";
  a.click();
  document.body.removeChild(a);
  // document.location.reload();
}

handleSubmit = e => {
  e.preventDefault();
  var lg = document.body.querySelector("#loadinggif");
  lg.style = "display:inline";
  var p = this.props.passableItems;
  this.props.form.validateFields((err, values) => {
    if (!err) {
      console.log("Received values of form: ", values);
      const person = EthCrypto.createIdentity();
      console.log("add public key to contract", person.publicKey);
      var name = values.Name;
      var email = values.Email;
      var phone = values.Phone_number;
      var address = p.account;
      var pubk = person.publicKey;
      // console.log(name, email, phone, address)
      this.registerLawyer(name, phone, email, address, pubk, p);
      this.downloadPrivateKey(person["privateKey"]);
    }
  });
};

constructor(props) {
  super(props)
  this.state = {
    account: '',
    loading: true,
    lawyerId: ''
  }
}
render() {
  const { getFieldDecorator } = this.props.form;
  return (
    <div className={styles.parentContainer}>
      <div className={styles.containerForm}>
        <h1>Register Lawyer</h1>
        <Form
          onSubmit={this.handleSubmit}
          className="login-form"
          style={{ margin: "2em" }}
        >
          <Form.Item>
            {getFieldDecorator("Name", {
```

```
    rules: []
  })(
    <Input
      prefix={
        <Icon type="user" style={{ color: "rgba(0,0,0,.25)" }} />
      }
      placeholder="Name"
    />
  )}
</Form.Item>
<Form.Item>
  {getFieldDecorator("Email", {
    rules: []
  })(
    <Input
      prefix={
        <Icon type="mail" style={{ color: "rgba(0,0,0,.25)" }} />
      }
      placeholder="Email"
    />
  )}
</Form.Item>
<Form.Item>
  {getFieldDecorator("Phone_number", {
    rules: []
  })(
    <Input
      prefix={
        <Icon type="number" style={{ color: "rgba(0,0,0,.25)" }} />
      }
      placeholder="Phone Number"
    />
  )}
</Form.Item>
<Form.Item>
  {getFieldDecorator("Address", {
    rules: []
  })(
    <Input
      prefix={
        <Icon type="lock" style={{ color: "rgba(0,0,0,.25)" }} />
      }
      placeholder={`Eth address: ${this.props.passableItems.account}`}
      disabled
    />
  )}
</Form.Item>
{/* <Form.Item>
                {getFieldDecorator("District", {
                    rules: [

                    ]
                })(
                    <Input
                        prefix={
                            <Icon
                                type="info"
                                style={{ color: "rgba(0,0,0,.25)" }}
```

```
                  />
                }
                placeholder="District"
              />
            )}
          </Form.Item> */}
  {/* <Form.Item>
            {getFieldDecorator("PublicKey", {
              rules: [

              ]
            })(
              <Input
                prefix={
                  <Icon
                    type="lock"
                    style={{ color: "rgba(0,0,0,.25)" }}
                  />
                }
                placeholder="Public Key"
              />
            )}
          </Form.Item> */}
<Form.Item className={styles.formBottom}>
  <Button
   type="primary"
   htmlType="submit"
   className="login-form-button"
  >
```

```
              Proceed to Add the user
            </Button>
            <br />
            <img
              style={{ display: "none" }}
              id="loadinggif"
              src={loading}
              alt=""
              height="100px"
            />
            <br />
          </Form.Item>
          <h3 id="lawyerId" style={{ display: "none" }}>
            Your Lawyer Id is: {this.state.lawyerId}
          </h3>
        </Form>
      </div>
    </div>
  );
 }
}
const WrappedNormalRegisterForm = Form.create({ name: "register"
})(Register);
export default WrappedNormalRegisterForm;
```

CHAPTER-9

APPENDIX 2

SCREEN SHOTS:

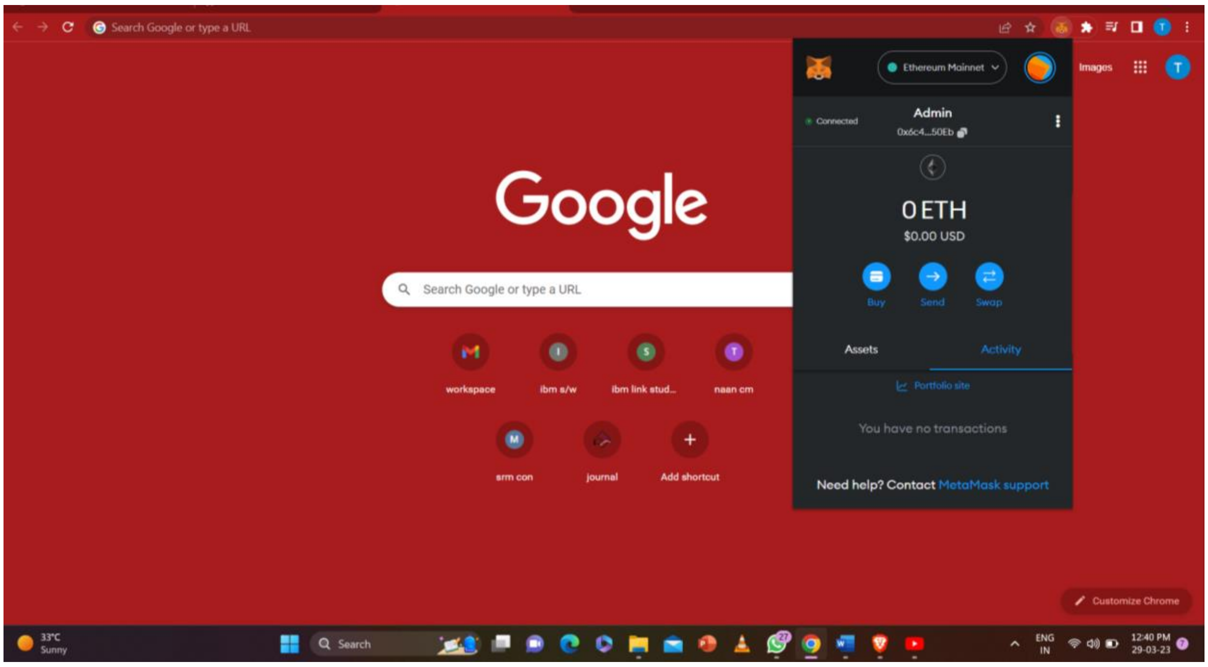
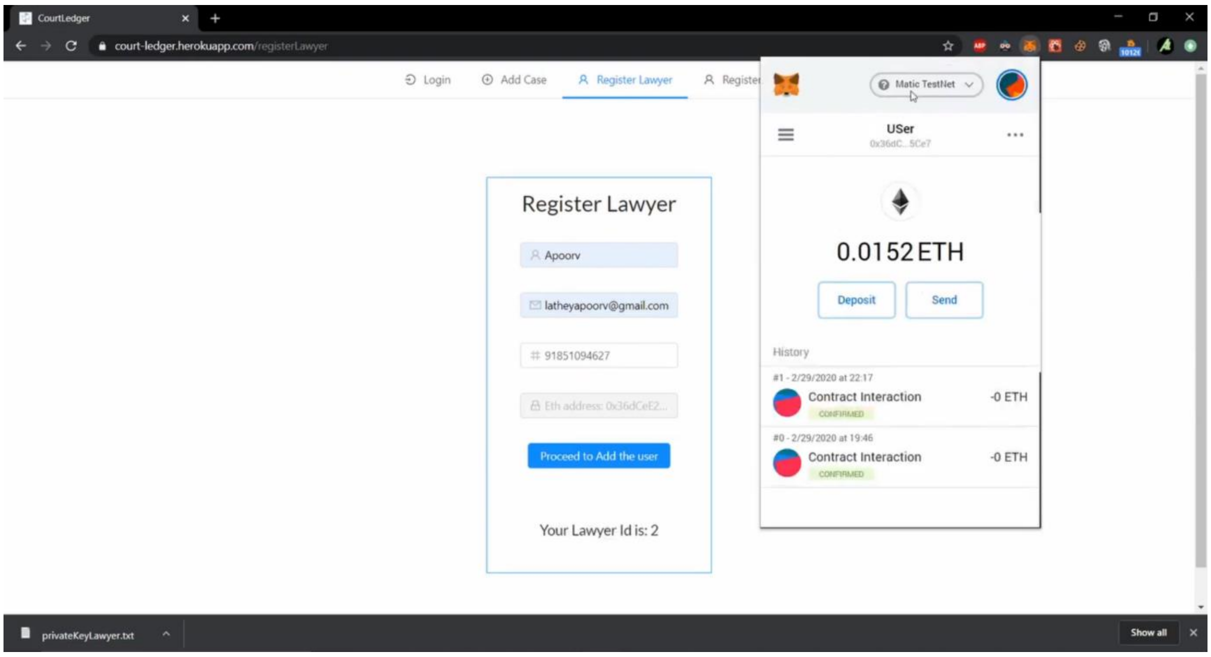


Fig 9.1 – Connect Blockchain



Register Lawyer

Apoorv

latheyapoorv@gmail.com

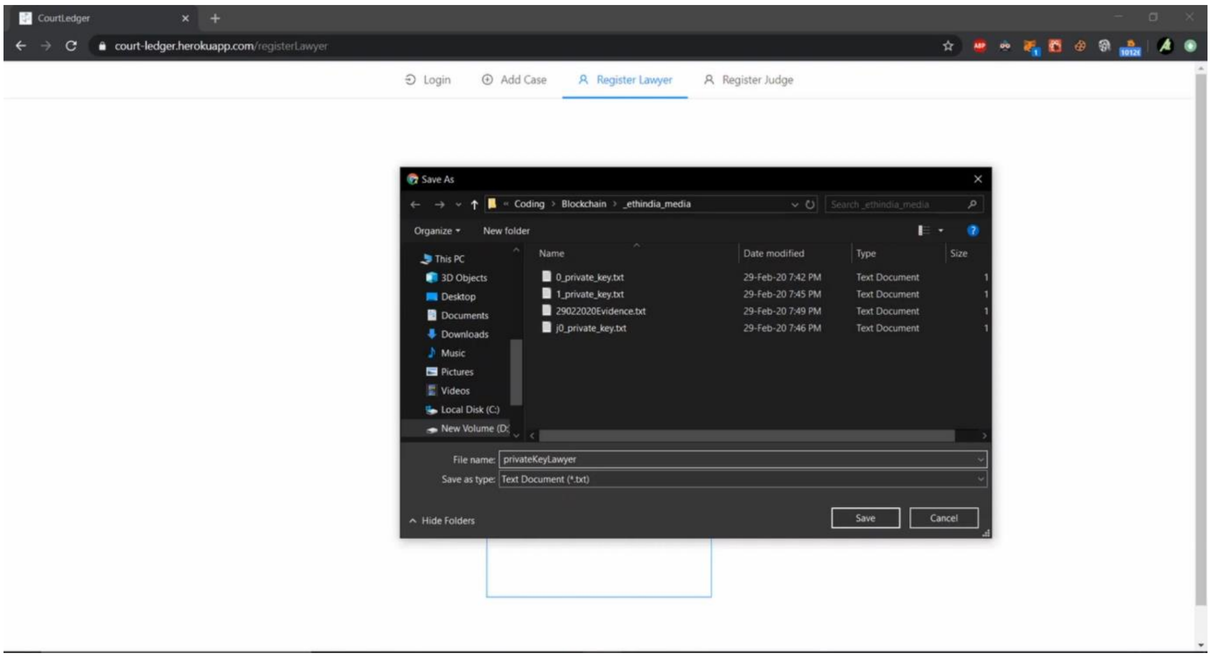
91851094627

Eth address: 0x36dCeE2...

Proceed to Add the user

Your Lawyer Id is: 2

Fig 9.2 – Register Lawyer



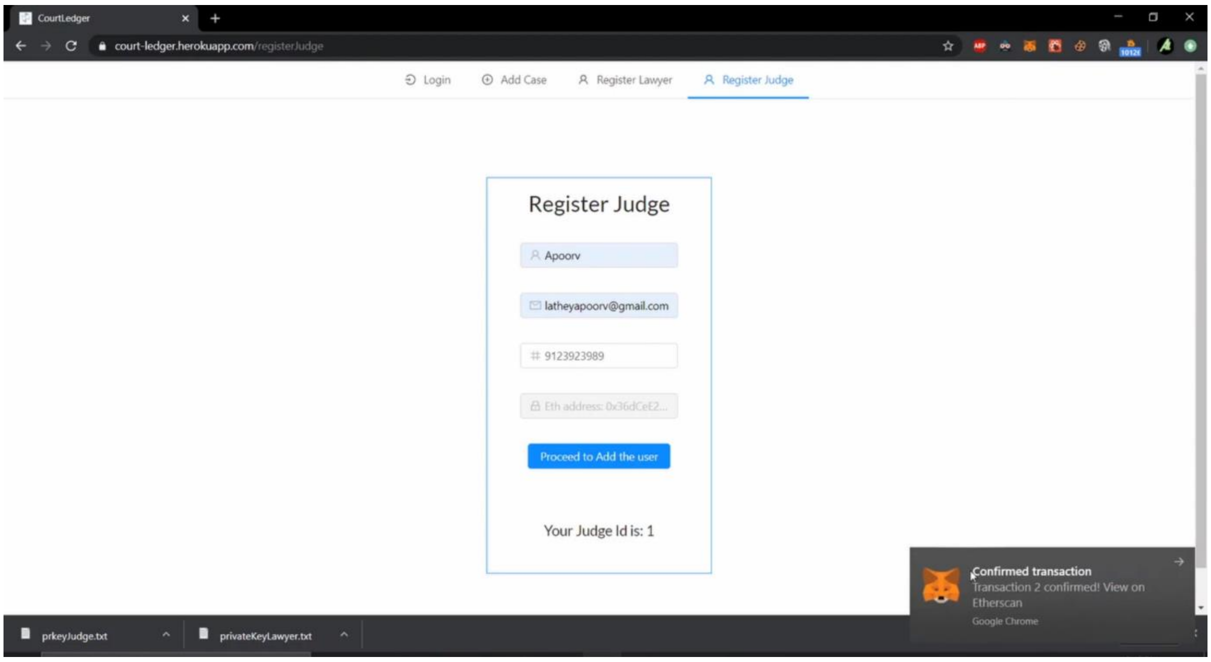
court-ledger.herokuapp.com/registerLawyer

Coding > Blockchain > _ethindia_media

Name	Date modified	Type	Size
0_private_key.txt	29-Feb-20 7:42 PM	Text Document	1
1_private_key.txt	29-Feb-20 7:45 PM	Text Document	1
29022020Evidence.txt	29-Feb-20 7:49 PM	Text Document	1
j0_private_key.txt	29-Feb-20 7:46 PM	Text Document	1

File name: privateKeyLawyer
Save as type: Text Document (*.txt)

Fig 9.3 – Lawyer private Key



court-ledger.herokuapp.com/registerJudge

Register Judge

Apoorv

latheyapoorv@gmail.com

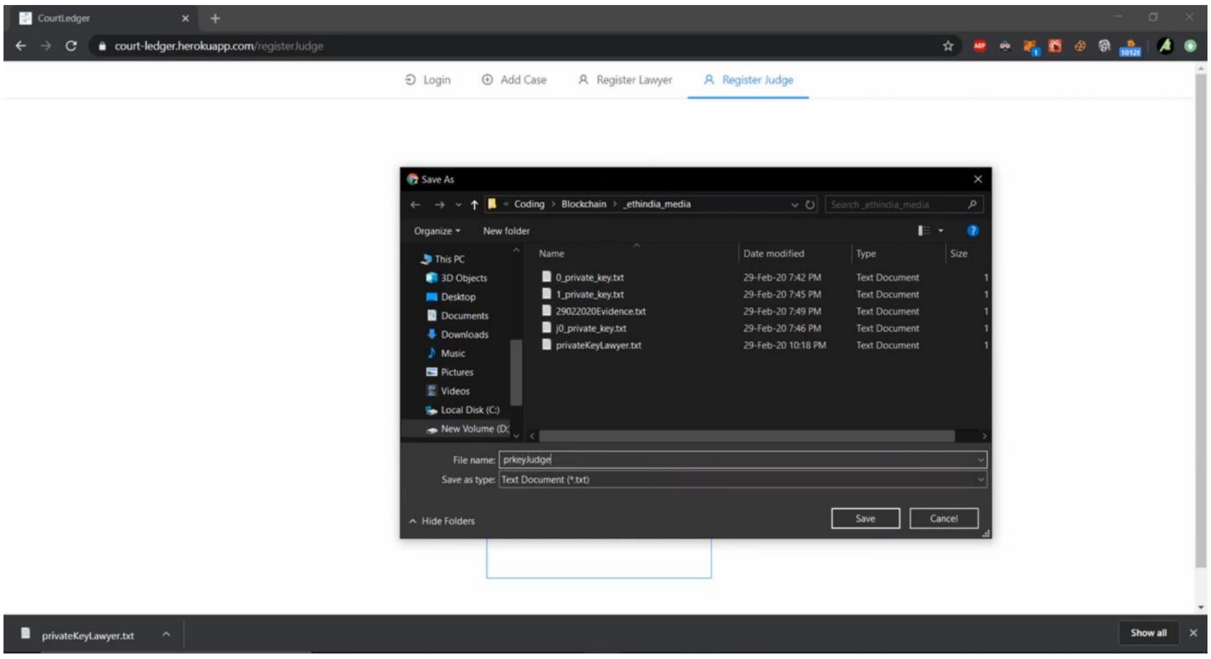
9123923989

Eth address: 0x36dCeE2...

Your Judge Id is: 1

Confirmed transaction
Transaction 2 confirmed! View on Etherscan

Fig 9.4 – Register Judge

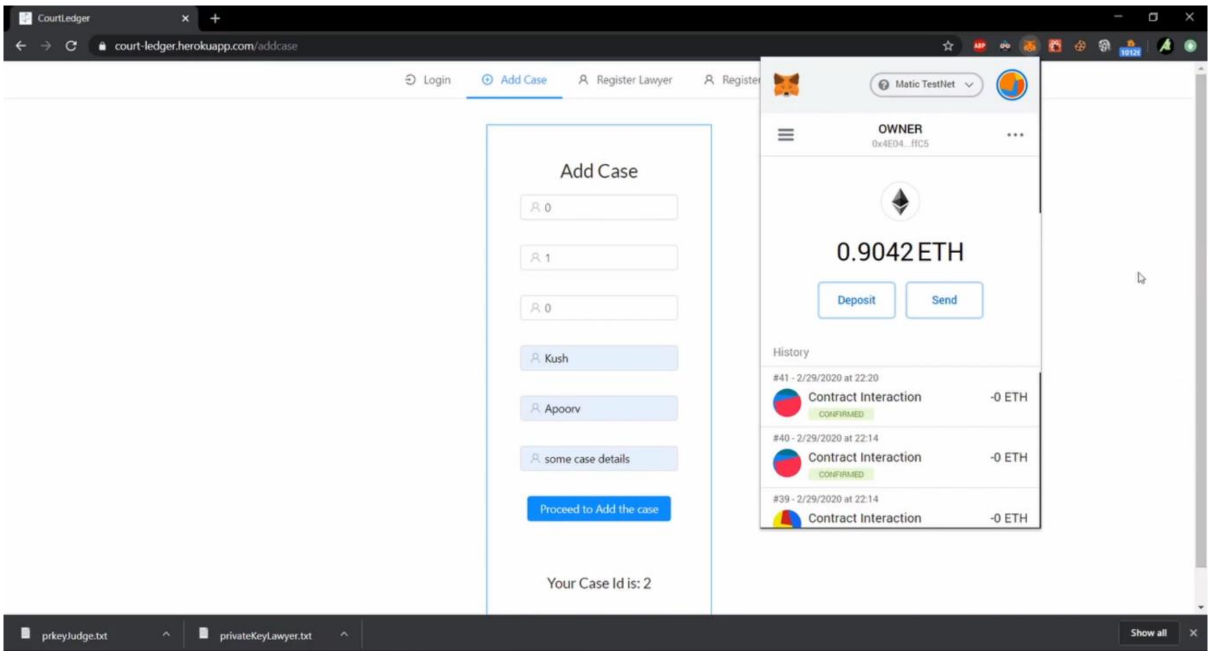


Name	Date modified	Type	Size
0_private_key.txt	29-Feb-20 7:42 PM	Text Document	1
1_private_key.txt	29-Feb-20 7:45 PM	Text Document	1
29022020Evidence.txt	29-Feb-20 7:49 PM	Text Document	1
j0_private_key.txt	29-Feb-20 7:46 PM	Text Document	1
privateKeyLawyer.txt	29-Feb-20 10:18 PM	Text Document	1

File name: prkeyJudge

Save as type: Text Document (*.txt)

Fig 9.5 – Judge Private Key



Add Case

0

1

0

Kush

Apoorv

some case details

Your Case Id is: 2

0.9042 ETH

#41 - 2/29/2020 at 22:20 Contract Interaction CONFIRMED -0 ETH

#40 - 2/29/2020 at 22:14 Contract Interaction CONFIRMED -0 ETH

#39 - 2/29/2020 at 22:14 Contract Interaction -0 ETH

Fig 9.6 – Add Case

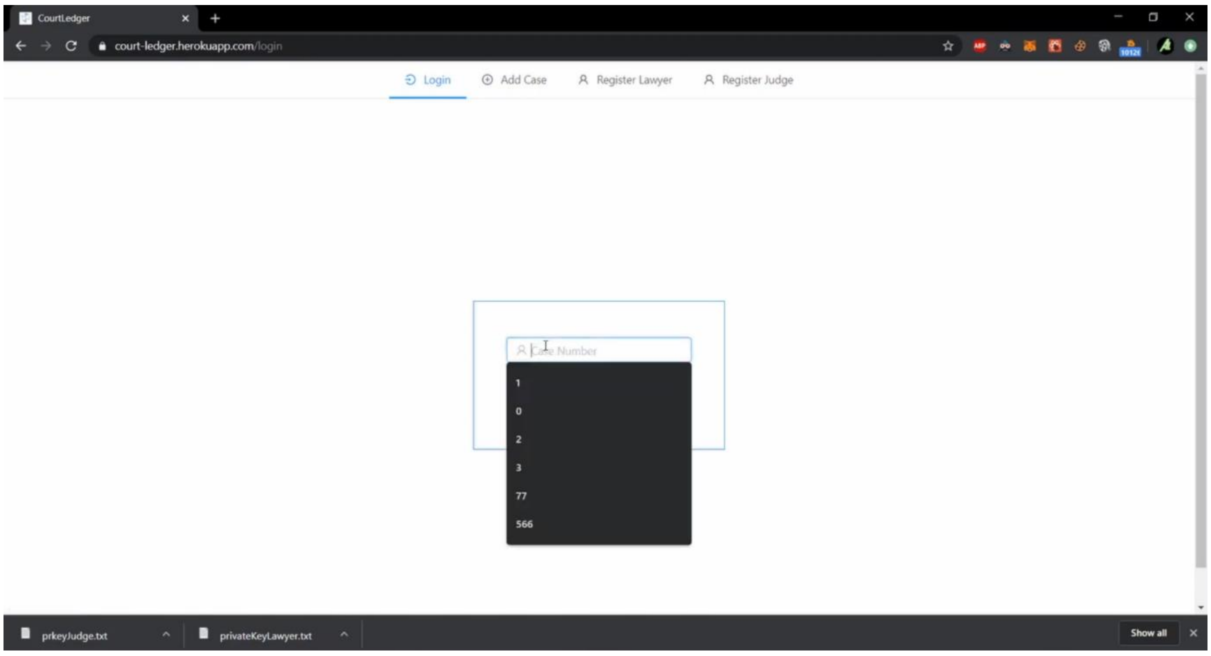
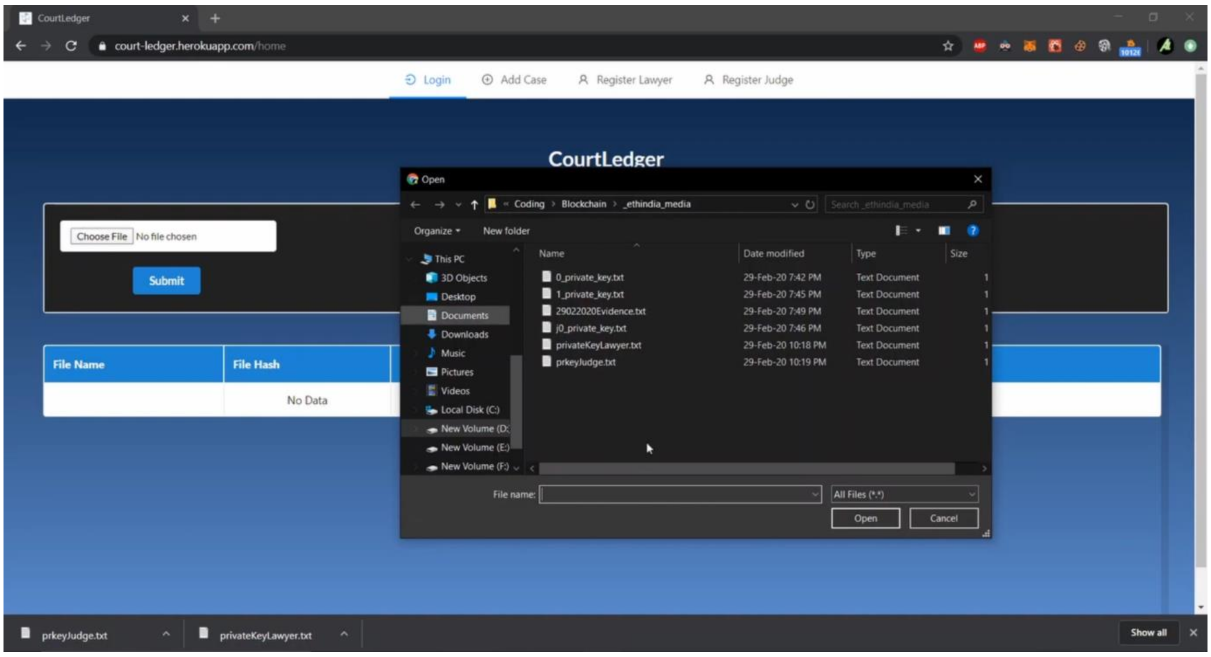
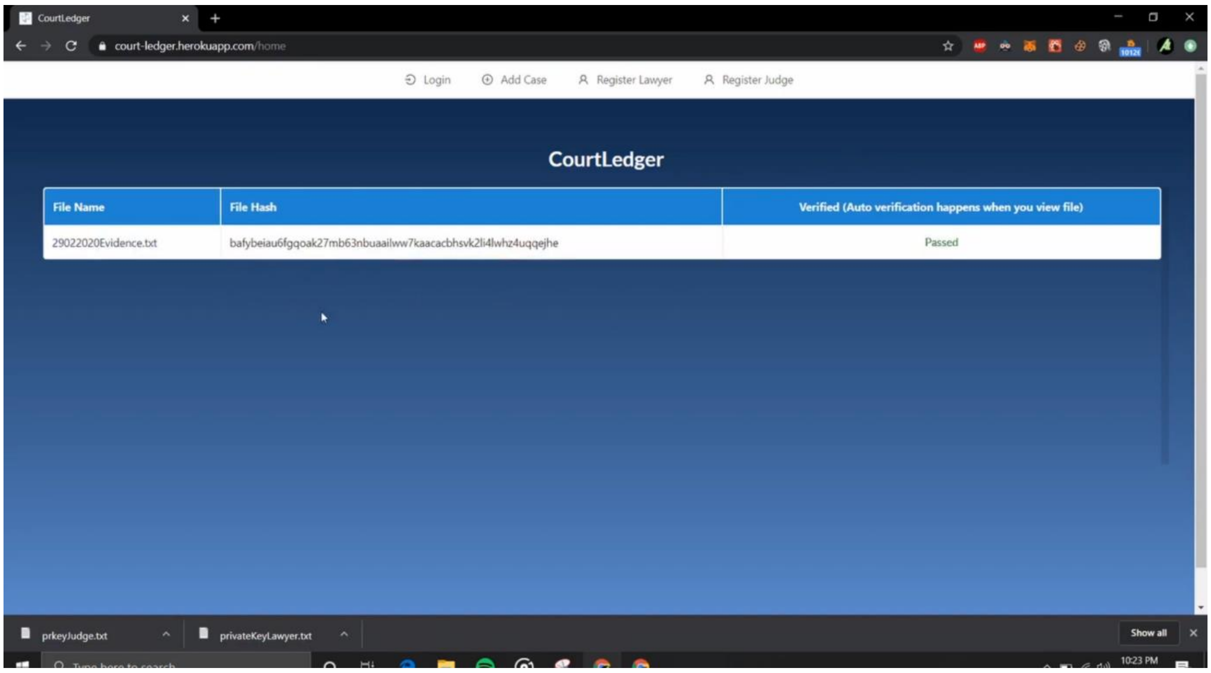


Fig 9.7 – Enter Case



Name	Date modified	Type	Size
0_private_key.txt	29-Feb-20 7:42 PM	Text Document	1
1_private_key.txt	29-Feb-20 7:45 PM	Text Document	1
29022020Evidence.txt	29-Feb-20 7:49 PM	Text Document	1
j0_private_key.txt	29-Feb-20 7:46 PM	Text Document	1
privateKeyLawyer.txt	29-Feb-20 10:18 PM	Text Document	1
prkeyJudge.txt	29-Feb-20 10:19 PM	Text Document	1

Fig 9.8 – Upload Evidence



CourtLedger

File Name	File Hash	Verified (Auto verification happens when you view file)
29022020Evidence.txt	bafybeiau6fgqoak27mb63nbuaailww7kaacacbhsvk2li4lwhz4uqqejhe	Passed

Fig 9.9 – View Evidence

REFERENCES

1. S. Surya, T. N. Janakiraman, and G. Chandrasekaran, "Secure Court Case Management using Blockchain," International Journal of Advanced Science and Technology, vol. 29, no. 10, pp. 3048-3055, 2020.

2. J. P. Singh, "Blockchain in the Legal Industry: Applications, Challenges and Opportunities," International Journal of Advanced Research in Computer Science, vol. 9, no. 2, pp. 267-273, 2018.

3. A. Al-Osaimi, M. S. Al-Sayed, and N. Al-Masri, "Towards Enhancing Court Procedures using Blockchain Technology," International Journal of Computer Science and Mobile Computing, vol. 8, no. 8, pp. 47-56, 2019.

4. M. Bicchierini, G. Conti, and A. Detti, "Blockchain for Public Administration: A Review," IEEE Access, vol. 8, pp. 13929-13951, 2020.

5. S. J. Ahmed, R. Ahmed, A. Abuhussein, and A. Alamri, "A Hybrid Blockchain-Based Solution for Court Case Management," in Proceedings of the 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 2019, pp. 1-5.

6. R. Kumar and R. Tripathi, "Implementation of Distributed File Storage and Access Framework using IPFS and Blockchain", Proc. IEEE Int. Conf. Image Inf. Process., 2019.

7. R. A. Canes sane, N. Srinivasan, A. Beria, A. Singh and B. M. Kumar, "Decentralized Applications Using Ethereum Blockchain", 5th Int. Conf. Sci. Technol. Eng. Math. ICONSTEM 2019,2019.

8. S. Jinju, L. Ming and M. Ingang, "Research and application of data sharing platform integrating Ethereum and IPFs Technology", Proc. - 2020 19th Diatribe. Compute. Appl. Bus. Eng. Sci. DCABES 2020,2020.

9. M. Steichen, B. Fizz, R. Norvell, W. Shari and R. State, "Blockchain-Based Decentralized Access Control for IPFS", Proc. - IEEE 2018 Int. Conger. Cinematics 2018 IEEE Conf. Internet Things Green Compute. Common. Cyber Phys. Soc. Compute. Smart Data Blockchain Compute. Inf. Technol. things/Gere, 2018.

10. X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, et al., "A taxonomy of blockchain-based systems for architecture design", 2017 IEEE International Conference on Software Architecture (ICSA), 2017.