STORING DATA IN DECENTRALISED CLOUD USING BLOCKCHAIN

A MINI PROJECT REPORT

Submitted by

ARVIND PRASHANTH.S (311518104008)

GOPINATH.S (311518104014)

KABHISH.M(311518104018)

In partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

MEENAKSHI SUNDARARAJAN ENGINEERING COLLEGE,

KODAMBAKKAM, CHENNAI-24

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2021

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"STORING DATA IN DECENTRALISED CLOUD USING BLOCKCHAIN"** is the bonafide work of "**ARVIND PRASHANTH.S (311518104008), GOPINATH.S (311518104014) ,KABHISH.M(311518104018)"** who carried out the mini project work under my supervision.

B.Monica Jenefer

SIGNATURE

Dr.B.Monica Jenefer,M.E,Ph.D

HEAD OF THE DEPARTMENT

Computer Science and Engineering

Meenakshi Sundararajan Engineering College,

No.363, Arcot Road, Kodambakkam,

Chennai -600 024.

R.Venkatesh

SIGNATURE

Mr.R.Venkatesh, M.E.

ASSISTANT PROFESSOR

Computer Science and Engineering

Meenakshi Sundararajan Engineering College,

No.363, Arcot Road, Kodambakkam,

Chennai -600 024.

Submitted for the project viva voce of Bachelor of Engineering in Computer Science and Engineering held on 10-08-2021.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

First and foremost we express our sincere gratitude to our Respected Correspondent **Dr.K.S.Lakshmi**, our beloved Secretary **Dr.K.S.Babai**, Principal **Dr.P.K.Suresh** and Dean Academics **Dr.K.Umarani** for their constant encouragement, which has been our motivation to strive towards excellence.

Our primary and sincere thanks goes to **Dr.B.Monica Jenefer**, Head of the Department, Department of Computer Science and Engineering, for her profound inspiration, kind cooperation and guidance.

We're grateful to our project coordinators **Mrs.C.Jerin Mahibha**,Senior Assistant Professor, Department of Computer Science and Engineering and **Mr.R.Venkatesh**, Assistant Professor, Department of Computer Science and Engineering. We are extremely thankful and indebted for sharing expertise, and sincere and valuable guidance and encouragement extended to us.

Above all, we extend our thanks to God Almighty without whose grace and blessings it wouldn't have been possible.

ABSTRACT

In a landscape where most files are stored in centralized servers the need for privacy and data security is ever increasing. Hence, it is important to come up with a system that allows users to save their data and store them securely. Blockchain is a technical solution for storing, verifying, transmitting and communicating network data based on cryptography. This is a system for storing data on the blockchain environment with the help of smart contracts. A smart contract is a computer program which is intended to automatically execute, control or document legally relevant events and actions according to the terms of a contract or an agreement. The contracts are written in Solidity and are migrated to the Ethereum blockchain. Ethereum is an open source platform and a blockchain which functions on smart contracts. In addition to the usual cryptocurrency transactions, Ethereum features a decentralized Ethereum Virtual Machine (EVM) to deploy and apply smart contracts. In addition to this, Metamask, a cryptocurrency wallet, is used for easier ethereum transactions. The user's data is stored on the InterPlanetary File System which generates a hash value based on the content for the stored file and returns it. The returned hash value is then stored on the block.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**iii**
	LIST OF FIGURES	**vii**
	LIST OF TABLES	**viii**
1	**INTRODUCTION**	
	1.1 ABOUT THE PROJECT	11
	1.2 BLOCKCHAIN	11
	1.3 DOMAIN OVERVIEW	12
	1.4 EXISTING SYSTEM	13
	1.5 PROBLEM STATEMENT	13
	1.6 CHAPTER OVERVIEW	14
2	**LITERATURE SURVEY**	
	2.1 IMPLEMENTATION OF DISTRIBUTED FILE STORAGE AND ACCESS FRAMEWORK USING IPFS AND BLOCKCHAIN	15
	2.2 CLOUD STORAGE. A COMPARISON BETWEEN CENTRALIZED SOLUTIONS VERSUS DECENTRALIZED CLOUD STORAGE SOLUTIONS USING BLOCKCHAIN TECHNOLOGY	15

2.3 RESEARCH AND APPLICATION OF DATA SHARING PLATFORM INTEGRATING ETHEREUM AND IPFS TECHNOLOGY 16

3 SYSTEM ARCHITECTURE

3.1 PROJECT ARCHITECTURE 17
3.2 SYSTEM ARCHITECTURE 17
3.3 HARDWARE REQUIREMENTS 18
3.4 SOFTWARE REQUIREMENTS 19

4 SYSTEM MODELING

4.1 UNIFIED MODELING LANGUAGE 23
4.2 USE CASE DIAGRAM 24
4.3 CLASS DIAGRAM 25
4.4 SEQUENCE DIAGRAM 27
4.5 COLLABORATION DIAGRAM 29
4.6 ACTIVITY DIAGRAM 30
4.7 STATECHART DIAGRAM 31
4.8 COMPONENT DIAGRAM 32
4.9 PACKAGE DIAGRAM 34
4.10 DEPLOYMENT DIAGRAM 35

5 SYSTEM IMPLEMENTATION

5.1 PROPOSED SYSTEM 37
5.2 MODULE DESCRIPTION 37
5.2.1 PROCESS OF UPLOADING FILES 38
5.2.2 STORING THE HASH IN BLOCKCHAIN 38
5.2.3 RETRIEVING THE FILE FROM IPFS 38

6 SOFTWARE TESTING
6.1 INTRODUCTION 39
6.2 TESTING APPROACHES 39
6.3 TESTING LEVELS 40
6.4 TESTING TYPES 41
6.5 SMART CONTRACTS TESTING USING TRUFFLE 42

7 CONCLUSION AND FUTURE ENHANCEMENT
7.1 CONCLUSION 44
7.2 FUTURE ENHANCEMENT 44

APPENDIX SCREENSHOT 45

REFERENCES 49

LIST OF FIGURES

FIGURE NO.	NAME OF THE FIGURE	PAGE NO.
1.1	CONCEPT OF BLOCKCHAIN	10
3.1	SYSTEM ARCHITECTURE	20
4.1	USE CASE DIAGRAM	24
4.2	CLASS DIAGRAM	25
4.3	SEQUENCE DIAGRAM	27
4.4	COLLABORATION DIAGRAM	28
4.5	ACTIVITY DIAGRAM	30
4.6	STATE CHART DIAGRAM	32
4.7	COMPONENT DIAGRAM	34
4.8	PACKAGE DIAGRAM	35
4.9	DEPLOYMENT DIAGRAM	36
A.1	LOCALIZED DEVELOPMENT BLOCKCHAIN	50
A.2	CLIENT SERVER USER INTERFACE	51
A.3	USING DOCUMENT TO UPLOAD TO THE IPFS	51
A.4	AUTHORIZING PAYMENT TO STORE THE IPFS HASH IN THE BLOCKCHAIN	52
A.5	NEW BLOCK CONTAINING THE IFPS HASH IS ADDED TO THE DEVELOPMENT BLOCKCHAIN	54

LIST OF TABLES

TABLE NO.	NAME OF THE TABLE	PAGE NO.
3.1	HARDWARE REQUIREMENTS	20
3.2	SOFTWARE REQUIREMENTS	20

CO-PO MAPPING

Course Outcomes

C318.1	Ability to understand and develop blockchain concepts
C318.2	Develop a decentralized web application using ReactJS
C318.3	Ability to understand the functionalities of IPFS and implement them
C318.4	Ability to manage time, improve proper communication skills and team management

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C318.1	3	3	3	3	3	2	2	2	2	3	3	3	3	3	3
C318.2	3	3	3	2	3	2	2	3	3	3	3	3	3	2	3
C318.3	3	3	2	3	3	3	2	2	2	2	3	3	3	2	3
C318.4	3	3	3	3	3	3	2	3	3	3	3	3	3	3	3
C318	3	3	2.75	2.75	3	2.5	2	2.5	2.5	2.75	3	3	3	2.5	3

JUSTIFICATION:

C318.1	Learned to work with Truffle and Ganache for deploying smart contracts.
C318.2	Acquired knowledge on web development to build a decentralized web application using ReactJS.
C318.3	Understood the functionalities of IPFS and implemented them.
C318.4	Improved time management, tem work, and developed good communication skills.

CHAPTER-1

INTRODUCTION

1.1 ABOUT THE PROJECT

In a landscape where most files are stored in centralized servers the need for privacy and data security is ever increasing. Hence, it is important to come up with a system that allows users to save their data and store them securely. Blockchain is a technical solution for storing, verifying, transmitting and communicating network data based on cryptography. This is a system for storing data on the blockchain environment with the help of smart contracts. A smart contract is a computer program which is intended to automatically execute, control or document legally relevant events and actions according to the terms of a contract or an agreement. The contracts are written in Solidity and are migrated to the Ethereum blockchain. Ethereum is an open source platform and a blockchain which functions on smart contracts. In addition to the usual cryptocurrency transactions, Ethereum features a decentralized Ethereum Virtual Machine (EVM) to deploy and apply smart contracts. In addition to this, Metamask, a cryptocurrency wallet is used for easier ethereum transactions. The user's data is stored on the InterPlanetary File System which generates a hash value based on the content for the stored file and returns it. The returned hash value is then stored on the block.

1.2 BLOCKCHAIN

Blockchain is a technical solution for storing, verifying, transmitting and communicating network data based on cryptography. The core concept is to rely on cryptographic and mathematical decentralized algorithms. Blockchain can make participants reach a consensus without the intervention from the third party. The transaction on the blockchain is verified by every block on the blockchain instead of the third party. Each transaction will produce a unique hash value, and each block header contains the previous block hash value, which connects all blocks and forms a chain.

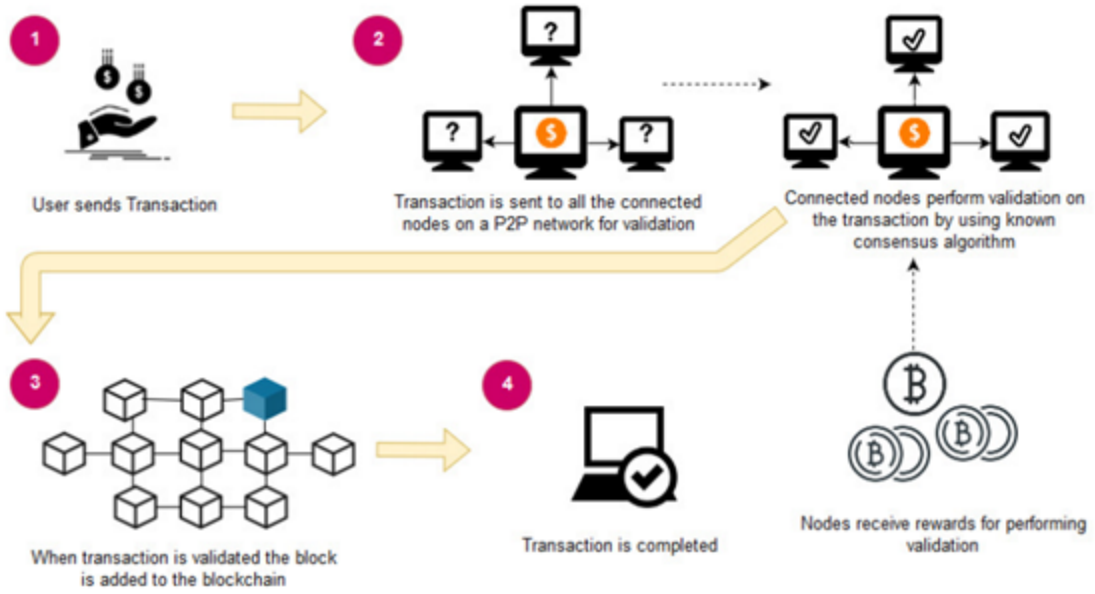


Fig 1.1 Concept of Blockchain

1.3 DOMAIN OVERVIEW

In this project, the main domains we concentrate on are blockchain and cloud computing. Blockchain is a chain of blocks that are connected together and are continuously growing by storing transactions on the blocks. This platform uses a decentralized approach that allows the information to be distributed and also ensures that each piece of distributed information commonly known as data has shared

ownership. Blockchains hold batches of transactions that are hashed thus providing them security and they are managed by peer-to-peer networks. A blockchain has certain benefits such as security, anonymity, and integrity of data with no third party intervention. These benefits make it a reasonable choice to store data on it, because the innovation of technology in the world has made the security of an user important.

1.4 EXISTING SYSTEM

The current system uses a centralized server architecture.In this type of system all users are connected to a central network owner or “server”.A limitation of this system is that if the server crashes, then the users cannot access their data.Centralized systems are systems that use client/server architecture where one or more client nodes are directly connected to a central server. This is the most commonly used type of system in many organisations where the client sends a request to a company server and receives the response.

1.5 PROBLEM STATEMENT

Every year around 1 million of data are being exposed, stolen or illegally disclosed because of this there is an absolute breach of the person’s privacy. In order to prevent the loss of such sensitive data, the private data of the people must be made more secure. Therefore, designing the system using blockchain is an appropriate solution in order to curb the loss of such data.

To design a portal for the people to easily upload their documents in an image format and achieve decentralisation on these stored files and also perform decryption to store these data more securely.

1.6 CHAPTER OVERVIEW

The project report is organized with various chapters that denote the various functionalities and aspects of the system being developed.

Chapter 1 gives a general description about the project. It represents the basic idea of the project and introduces the topics of the existing system and proposed system.

Chapter 2 deals with the related works of the project. A literature review for each related work is explained in detail.

Chapter 3 presents the system architecture and requirements. It specifies the hardware and software components that are required. It also lists the technologies used in the implementation of the project.

Chapter 4 explains the system design with the use of UML diagrams and the data flow diagrams.

Chapter 5 contributes a detailed description of different modules that are there in the design and how they are implemented.

Chapter 6 gives a detailed description of the different test cases that were performed on the system.

Chapter 7 provides the conclusion. It also elucidates how the project can be further enhanced.

CHAPTER 2
LITERATURE SURVEY

2.1 IMPLEMENTATION OF DISTRIBUTED FILE STORAGE AND ACCESS FRAMEWORK USING IPFS AND BLOCKCHAIN

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 transactions. 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 causing problems for storage but also in accessing the block transactions. an IPFS based blockchain storage model to solve the storage problem of transactions in a block along with access to transactions of a particular block. In the proposed storage model, the miners store transactions on IPFS distributed file system storage and get the returned IPFS hash of the 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 to a particular block content-addressed (IPFS hash) storage technique has been proposed.

Disadvantages:

- More bandwidth is used

2.2 CLOUD STORAGE. A COMPARISON BETWEEN CENTRALIZED SOLUTIONS VERSUS DECENTRALIZED CLOUD STORAGE SOLUTIONS USING BLOCKCHAIN TECHNOLOGY

The introduction of Renewable Energy Sources (RES) have caused a series of challenges from an engineering perspective mostly due to their unpredictable nature. In order to consolidate, improve production of energy, transmission and distribution, the traditional power system has evolved from a monolithic architecture to a decentralized architecture. It's not uncommon for Decentralized Control Systems also known as Distributed Control Systems (DCS) to process data locally close to the area where it's generated using edge computing.

- More than one block is needed for transaction validation

2.3 RESEARCH AND APPLICATION OF DATA SHARING PLATFORM INTEGRATING ETHEREUM AND IPFS TECHNOLOGY

The development of the Internet has increased the amount of data exponentially, and the reliable transmission, sharing, and storage of data have become the primary problems of data sharing. This article combines the decentralized and irreversible characteristics of the Ethereum blockchain, integrates IPFS distributed storage technology, and proposes a data sharing platform under a new technology environment to ensure data security, user rights protection, and high-speed data processing. Provide a new technology application environment for current data sharing.

Disadvantages:

- High chance of losing keys
- High cost for large storage

CHAPTER 3
SYSTEM ARCHITECTURE

3.1 PROJECT ARCHITECTURE

System Architecture defines a comprehensive solution based on principles, concepts, and properties logically related and consistent with each other. The architecture has features, properties, and characteristics satisfying, as far as possible, the problem or opportunity expressed by a set of system requirements and life cycle concepts (e.g., operational, support) and is implementable through technologies (e.g., software, services, procedures, human activity).

The Architecture explains how the people can upload their documents to the blockchain and also shows the presence of a nodejs development server involved in this process and various other modules like BNER and IPFS network which are further explained below.

3.2 SYSTEM ARCHITECTURE

The User scans and uploads the document.The document is then encrypted from the client side and sent to the nodejs development server.The nodejs development server sends this encrypted file to the IPFS network for storage. Once stored, it returns a file hash.The file hash is then returned back to the client app and then it is safely stored in the ethereum blockchain.

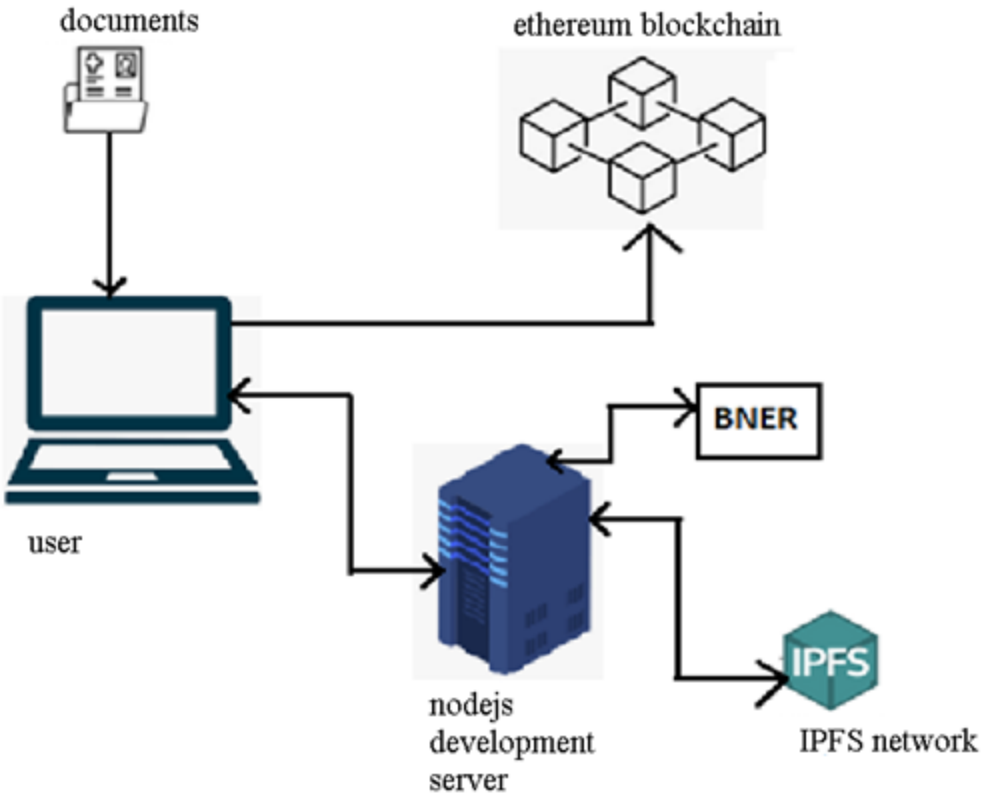


Figure 3.1 System Architecture

3.3 HARDWARE REQUIREMENTS:

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. A HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following subsections discuss the various aspects of hardware requirements.

S.No	REQUIREMENTS	RECOMMENDED	MINIMUM REQUIREMENTS
1	Operating System	Windows 10	Windows 8
2	RAM	4 GB	2 GB
3	HDD	1 TB	500 GB
4	Processor	Intel Quad Core PENTIUM IV	Intel Dual Core PENTIUM III

TABLE 3.1 Hardware Requirements

3.4 SOFTWARE REQUIREMENTS

Requirements	Specification
FRONT END	**React**
TOOL	**Ganache, Truffle, Metamask,**
CODING LANGUAGE	**Solidity**
FILE STORAGE	**IPFS**

Table 3.2 Software Requirements

REACT:

React is a open source, front end, javascript library that is used for building user interfaces and it helps to manage the view layer.It helps set up the development environment so that the latest JavaScript features can be used.It also provides a great developer experience, and optimizes the application for production.

TRUFFLE:

Truffle is a development environment and testing framework for blockchains using the Ethereum Virtual Machine(EVM). It has got built-in smart contract compilation and automated contract testing. Also provides package management with Node Package Manager(npm). And the truffle suite comes with an automated testing framework to test and run the smart contracts.

Truffle comes standard with npm integration, and is aware of the node_modules directory and therefore can be used to distribute contracts, dapps and Ethereum-enabled libraries via npm.

GANACHE:

Ganache is a personal blockchain for rapid Ethereum and Corda distributed

application development. Ganache can be used across the entire development cycle; thereby providing assistance to develop, deploy, and test the dApps in a safe and deterministic environment.

Ganache UI is a desktop application supporting both Ethereum and Corda technology. In addition, an Ethereum version of ganache is available as a command-line tool: ganache-cli (formerly known as the TestRPC). All versions of Ganache are available for Windows, Mac, and Linux.

METAMASK:

MetaMask is a cryptocurrency wallet as well as a web browser extension that is available in Chrome, Firefox and Brave.It is used inorder to store, send and receive Ethereum and ERC20. In other words, it allows users to make Ethereum transactions through regular websites. MetaMask can be used to store keys for Ethereum cryptocurrencies only.It is a wallet for the browser

INTERPLANETARY FILE SYSTEM:

IPFS is a distributed system for storing and accessing files, websites, applications, and data.IPFS makes it possible to download a file from many locations that aren't managed by one organization.

Instead of being location-based, IPFS addresses a file by what's in it, or by its content. The content identifier is a cryptographic hash of the content at that address. The hash is unique to the content that it came from, even though it may look short compared to the original content. It also allows one to verify that you got what you asked for

IPFS is based on the ideas of possession and participation, where many people possess each others' files and participate in making them available.

SOLIDITY:

Solidity is an object-oriented, high-level language for implementing smart contracts. Smart contracts are programs which govern the behaviour of accounts with Ethereum state. Solidity was influenced by C++, Python and JavaScript and is designed to target the Ethereum Virtual Machine (EVM).

CHAPTER 4

SYSTEM MODELING

4.1 UNIFIED MODELING LANGUAGE (UML)

Unified Modeling Language is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

The primary goals in the design of the UML as follows:

- Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
- Provide extensibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the OO tools market.

- Support higher-level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

4.2 USE CASE DIAGRAM

The use case diagram is used to define the core elements and processes that make up a system. The key elements are termed as “actors” and the processes are called “use cases”. The use case diagram shows which actors interact with each use case. This definition defines what a use case diagram is primarily made up of – actors and use cases.

In software and system engineering, a use case is a list of steps, typically defining interactions between a role (known in UML as an “actor”) and a system, to achieve a goal. The actor can be a human or an external system. In system engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals.

The purposes of use case diagrams can be as follows:

1. Used to gather requirements of a system.
2. Used to get an outside view of a system.
3. Identify external and internal factors influencing the system.
4. Showing the interacting among the requirements are actors.

Use cases help in identifying the operations that can be performed by an actor. It gives a list of the various applications that can be utilized by the system. The actor can be real time human or a system. It helps in identifying the various modules

present in the system. A single use case diagram captures a particular functionality of a system. Hence to model the entire system, a number of use case diagrams are used.

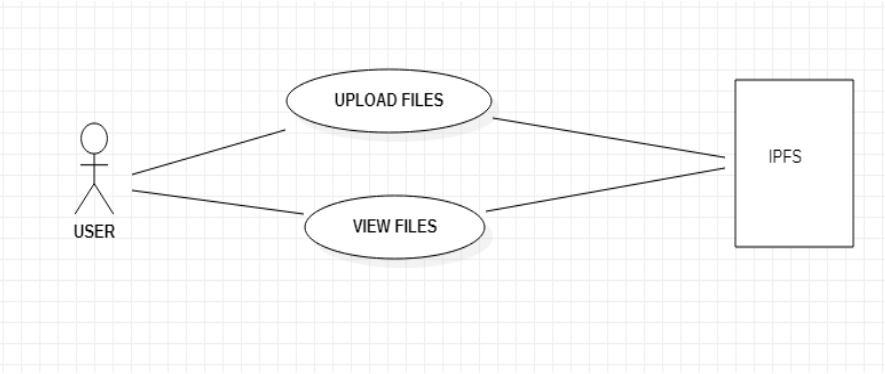


Figure 4.1 Use Case Diagram

In the system, the actors engaged are the user and the system. The documents are uploaded by the user to the system and stored in the IPFS server. The user can also view the uploaded document.The uploaded file is stored as hash in the blockchain

4.3 CLASS DIAGRAM

Class diagram is a static diagram. It is the building block of every object-oriented system and helps in visualizing and describing the system. A class diagram depicts the structure of the system through its classes, their attributes, operations and relationships among the object. A class is a blueprint that defines the variables and methods common to all objects of a certain kind. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. The characteristics of Class Diagram are:

1. Each class is represented by a rectangle having a subdivision of three compartments - name, attributes and operations.

2. There are three types of modifiers which are used to decide the visibility of attributes and operations : + is used for public visibility, # is used for protected visibility, – is used for private visibility.

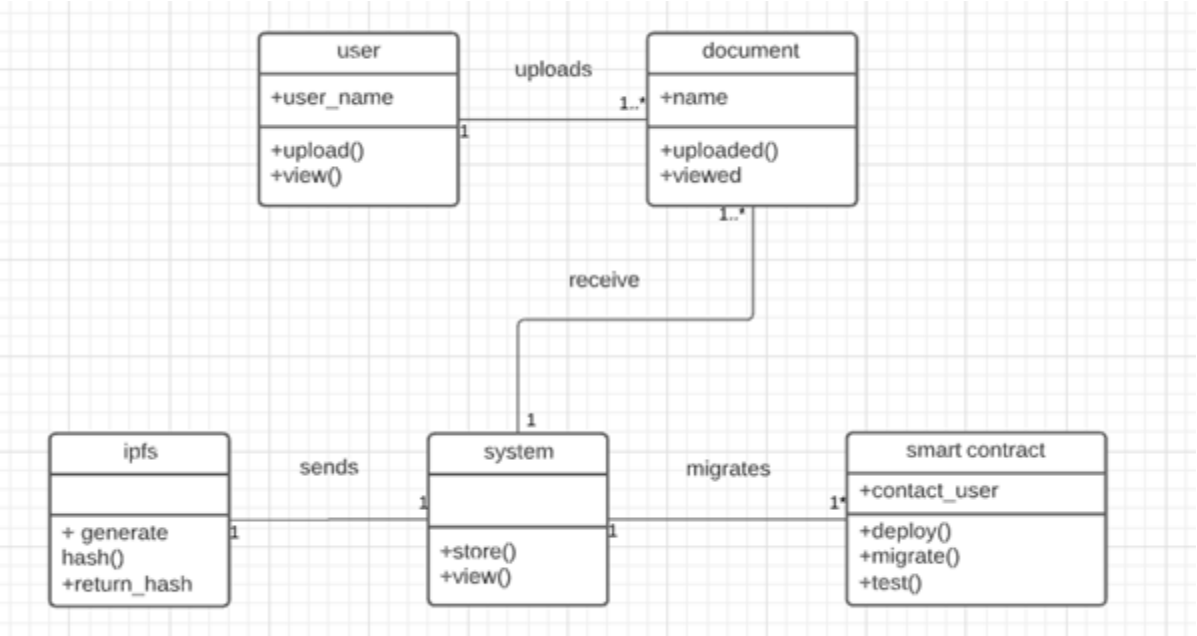


Figure 4.2 Class Diagram

In the diagram, classes are represented with boxes that contain three compartments. The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized. The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase. The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

The main modules that are involved in this system are user,document, IPFS, smart contract and the system. The user is the front end user of the system who uploads the documents. The system stores the uploaded documents using IPFS which generates and returns a hash value. The smart contract is migrated to the blockchain.

4.4 SEQUENCE DIAGRAM

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in which order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence.

Sequence diagrams are a popular dynamic modeling solution in UML because they specifically focus on lifelines, or the processes and objects that live simultaneously, and the messages exchanged between them to perform a function before the lifeline ends.

It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

A sequence diagram shows different processes or objects that live simultaneously as parallel vertical lines (lifelines) and the messages exchanged between them and the order in which they occur as horizontal arrows.

The main purpose of the Sequence diagram is

- To capture the dynamic behavior of a system.
- To describe the message flow in the system.

- To describe the structural organization of the objects.
- To describe the interaction among objects.

Sequence diagrams can be used

- To model the flow of control by time sequence.
- To model the flow of control by structural organizations.
- For forward engineering
- For reverse engineering

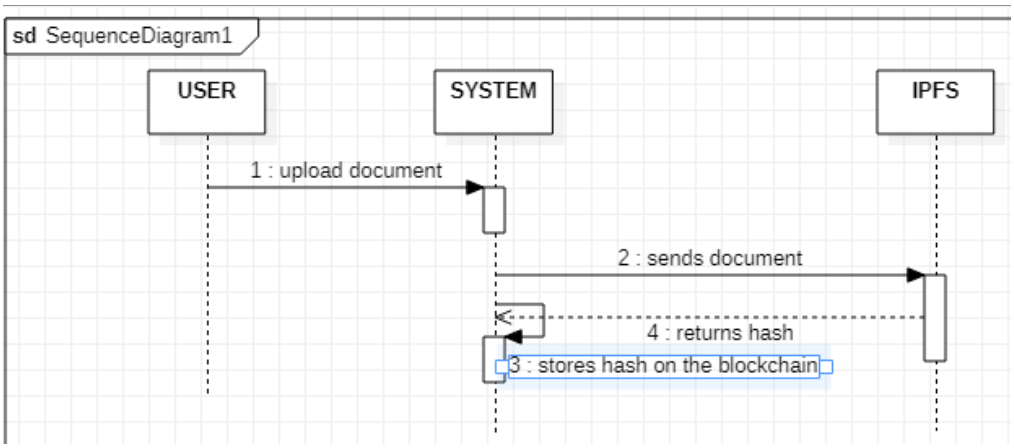


Figure 4.3 Sequence Diagram

In this system, the user, system and ipfs are the lifeline elements of the system. Initially the user uploads the document and It is stored in the hash of the blockchain.

4.5 COLLABORATION DIAGRAM

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among objects in the Unified Modeling Language (UML).

Collaboration diagrams convey the same information as sequence diagrams, but focus on object roles instead of the timings of messages. It illustrates messages being sent between classes and objects (instances).

Collaboration diagrams represent a combination of information taken from class, sequence and use case diagrams describing both the static structure and dynamic behavior of a system. The collaboration diagram describes the messages or roles sent between objects.

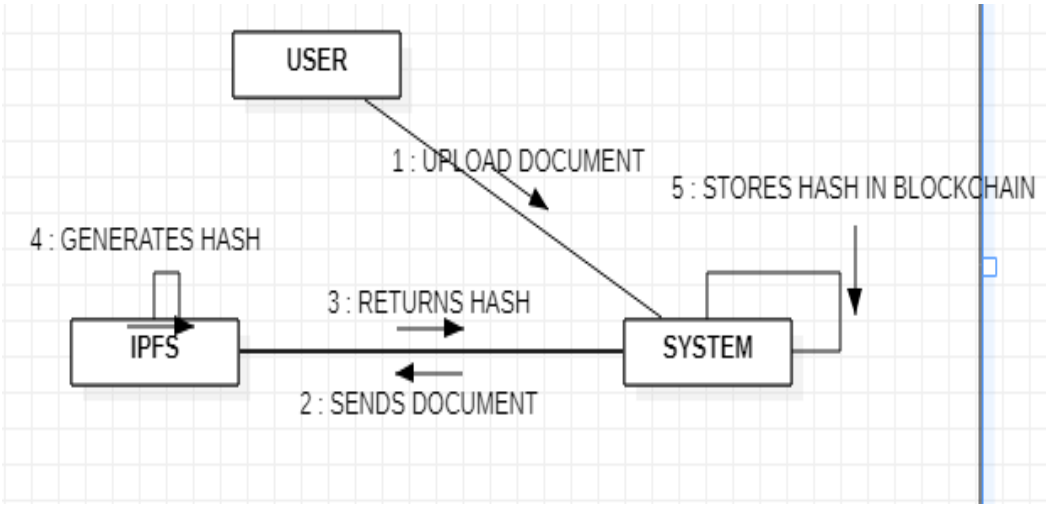


Figure 4.4 Collaboration Diagram

In this system, the user, system and ipfs are the lifeline elements of the system. Initially the user uploads the document and It is stored in hash of the blockchain

4.6 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent.

Activity diagrams deal with all types of flow control by using different elements such as fork, join, etc. Activity diagrams are constructed from a limited number of shapes, connected with arrows.

The most important shape types:

- rounded rectangles represent actions;
- diamonds represent decisions;
- bars represent the start (split) or end (join) of concurrent activities;
- a black circle represents the start (initial node) of the workflow;
- an encircled black circle represents the end (final node).

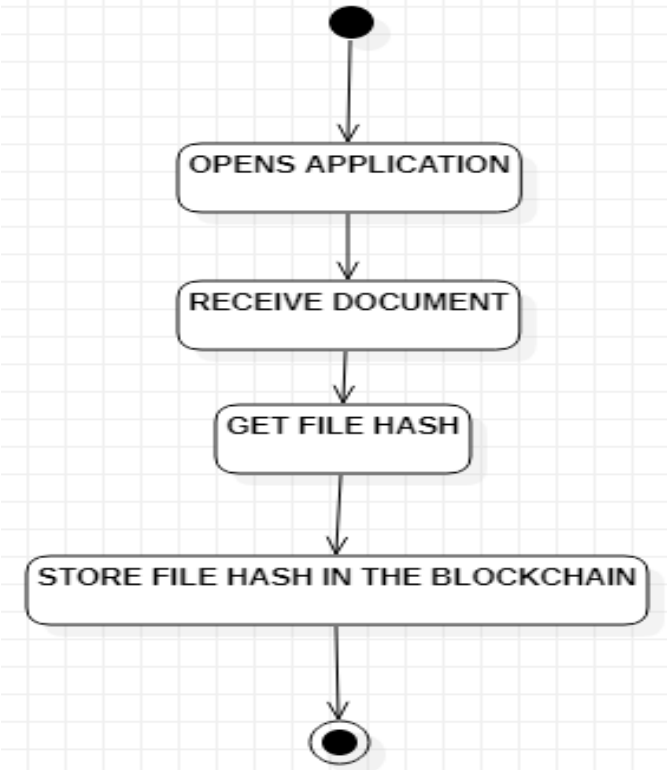


Figure 4.5 Activity Diagram

In this system, initially the web application is opened and the user upload the document in the application, the document is decrypted and stored in the hash of the blockchain

4.7 STATE CHART DIAGRAM

A State chart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events. It describes different states of a component in a system. The states are specific to a component/object of a system. State chart diagrams are used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. State chart diagrams are useful to model the reactive systems.

State chart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of State chart diagrams is to model the lifetime of an object from creation to termination.

The main purposes of using State chart diagrams

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time
- an encircled black circle represents the end (final node).

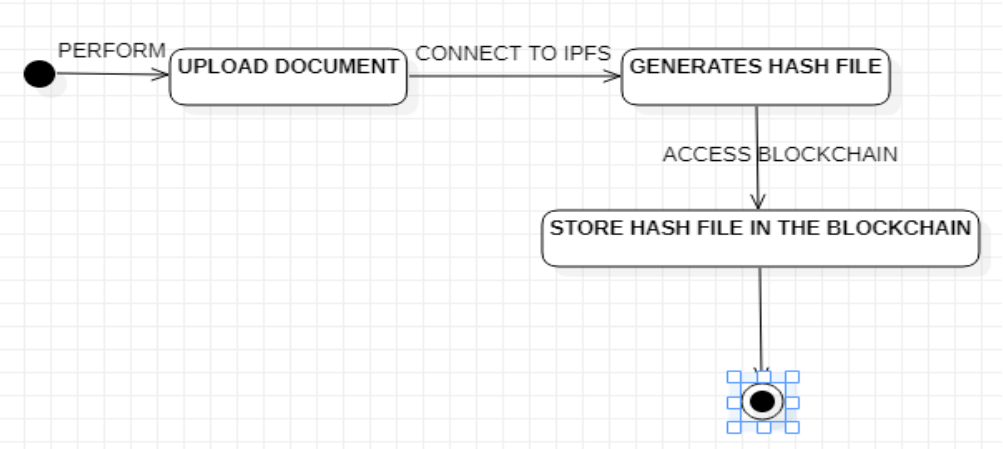


Figure 4.6 State Chart Diagram

In this system, initially the web application is opened and the user upload the document in the application, the document is decrypted and stored in the hash of the blockchain

4.8 COMPONENT DIAGRAM

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components or software systems. They

are used to illustrate the structure of arbitrarily complex systems. Component diagrams are different in terms of nature and behavior. Component diagrams are used to model the physical aspects of a system. Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node.

Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems. Component diagram is a special kind of diagram in UML.

The purpose is also different from all other diagrams. It does not describe the functionality of the system but it describes the components used to make those functionalities. Component diagrams can also be described as a static implementation view of a system. Static implementation represents the organization of the components at a particular moment.

The purpose of the component diagram can be summarized as Visualizing the components of a system.

- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the component

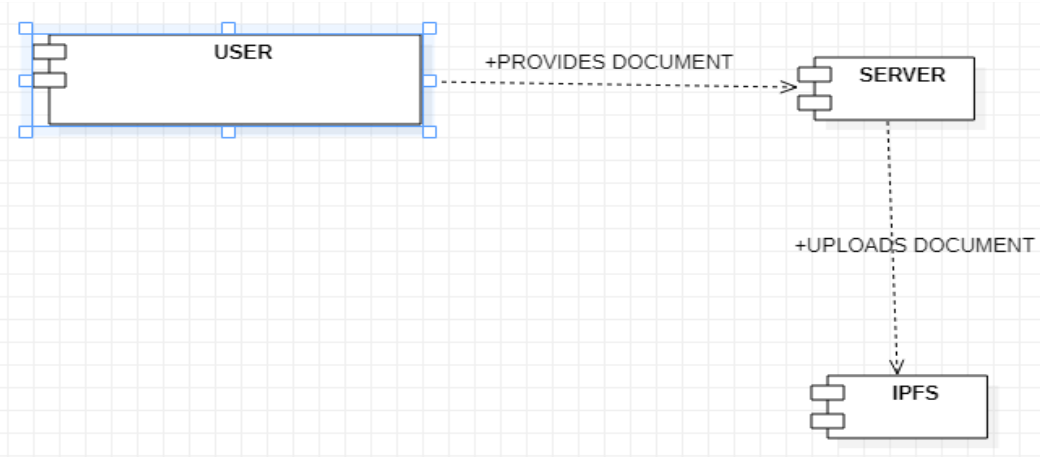


Figure 4.7 Component Diagram

In this system, there are three main components- the user, the server and the ipfs.

4.9 PACKAGE DIAGRAM

Package diagram is a UML structure diagram which shows packages and dependencies between the packages. The package diagram shows the arrangement and organization of the model elements in a middle to large scale project. The package diagram can show both the structure and dependencies between subsystems or modules. A package is rendered as a tabbed folder – a rectangle with a small tab attached to the left side of the top of the rectangle. If the members of the package are not shown inside the package rectangle, then the name of the package should be placed inside. The members of the package may be shown within the boundaries of the package. In this case, the name of the package should be placed on the tab. A diagram showing a package with content can show only a subset of the contained elements according to some criterion.

Members of the package may be shown outside of the package by branching lines from the package to the members. A plus sign (+) within a circle is drawn at the end attached to the namespace (package).

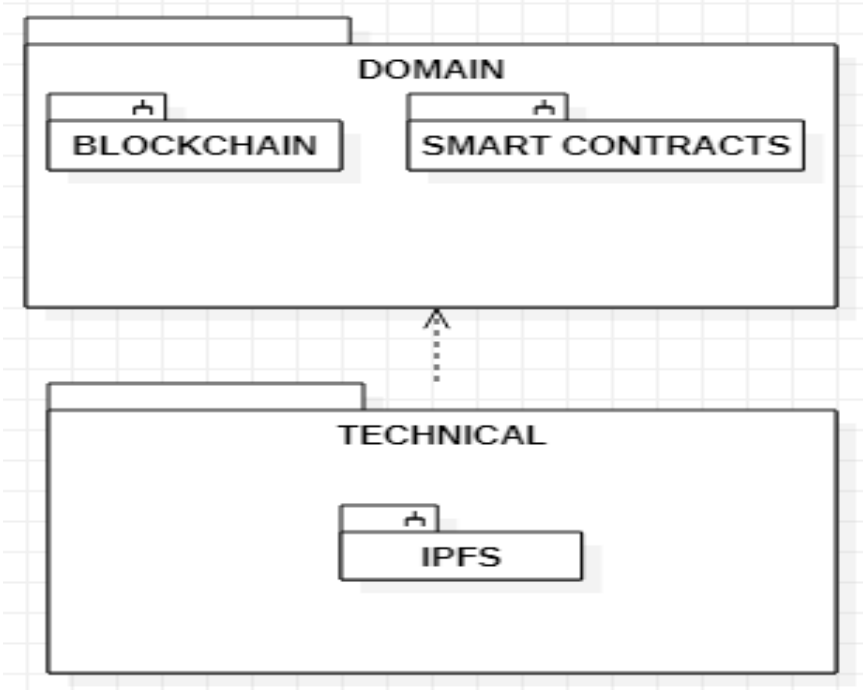


Figure 4.8 Package Diagram

In this system, there are packages that are necessary for any process or system namely ipfs Interface, blockchain and smart contracts. The ipfs interface is where the document is encrypted and stored in the blockchain.

4.10 DEPLOYMENT DIAGRAM

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a website, for example, a deployment diagram would show what hardware components (“nodes”) exist (e.g., a web server, an application server, and a database server), what software components (“artifacts”) run on each node (e.g., web application, database), and how the different pieces are connected (e.g., JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub nodes, which appear as nested

boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers. Deployment diagrams are used by system engineers.

The purposes of deployment diagrams can be as follows:

1. Visualize the hardware topology of a system.
2. Describe the hardware components used to deploy software components.
3. Describe the runtime processing nodes.

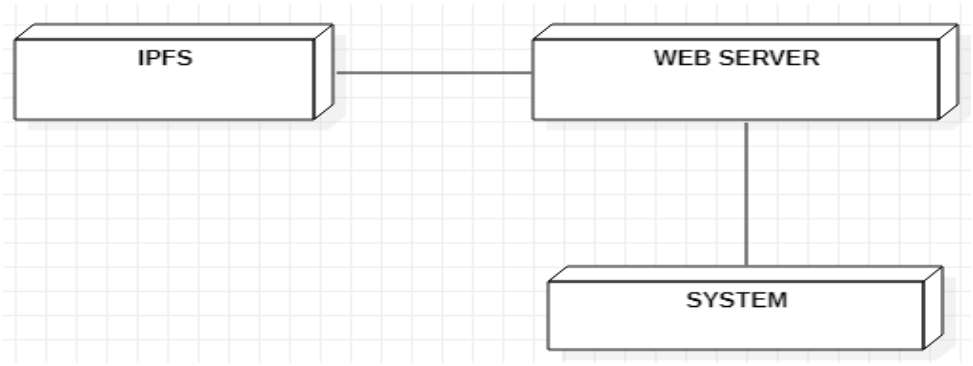


Figure 4.9 Deployment Diagram

In this system, the users upload data to the system using the web application. The document is encrypted and stored in the hash of the blockchain.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 PROPOSED SYSTEM:

A system for managing documents on the blockchain environment with the help of smart contracts is proposed. The contracts are written in Solidity and are migrated to the Ethereum blockchain. Also a cryptocurrency wallet is used for the easier ethereum transactions. The documents uploaded by the people are stored on the InterPlanetary File System which generates a hash value for the stored file and returns it. The returned hash value is then stored on the block.

In the system construction, we use ReactJS as the front end of the system, and Ganache is used as the blockchain environment for simulation. Besides, to write and test smart contracts, we use Solidity. In the file upload section, we apply the IPFS API to uploading files to IPFS. The architecture of the proposed system is similar to that of storj. Both systems use smart contracts to register and obtain documents, and utilize the Ethereum environment.

5.2 MODULE DESCRIPTION:

In the system construction, we use ReactJS as the front end of the system, and Ganache is used as the blockchain environment for simulation. Besides, to write and test smart contracts, we use Solidity. In the file upload section, we apply the IPFS API to uploading files to IPFS. The architecture of the proposed system is similar to that of storj. Both systems use smart contracts to register and obtain documents, and utilize the Ethereum environment.

The first step is to stimulate the Ethereum blockchain using Ganache and the Truffle project. Then the migrations are performed using the “truffle migrate”

command. Secondly, the ganache and the Metamask(cryptocurrency wallet) are to be connected by just importing the account by using its private key. The following are the modules present in the system and its associated processes.

5.2.1 PROCESS OF UPLOADING FILES:

The user can upload any necessary document to the web application by paying a specific ETH in the Ethereum blockchain.

5.2.2 STORING THE HASH IN BLOCKCHAIN:

The document uploaded is stored on the IPFS which in turn returns a hash value and this hash value is returned to the saveFile function in the smart contract and stored in the blockchain.

5.2.3 RETRIEVING THE FILE FROM IPFS:

The ipfs hash previously stored in the blockchain is retrieved using the corresponding hash and is shown to the user.

CHAPTER 6

SYSTEM TESTING

6.1 INTRODUCTION

Software testing is a process of evaluating a software in a comprehensive manner in order to verify whether it is running in the desired fashion and to identify any bugs present if any. This is done in order to keep the quality of the deliverable high.

In accordance with the standard of ANSI, the definition of Software Testing is – A process of analyzing a software item to detect the differences between existing and required conditions (i.e., defects) and to evaluate the features of the software item.

6.2 TESTING APPROACHES

There are two main kinds of testing approaches under software testing, which are:

i. White Box Testing
ii. Black Box Testing

6.2.1 WHITE BOX TESTING

White box testing involves analyzing the internal structures of the code and not merely the functionality. It is used to identify any errors present in the code structure. It is usually done for unit testing although it can be done for integration and system testing too. White box testing is used for debugging code within a subsystem, between subsystems and so on. In white-box testing, an internal perspective of the system, as well as programming skills, are used to design test cases. It is also called Glass Box, Clear Box, Structural Testing.

Advantages of White Box Testing

- White Box Testing has simple and clear rules to let a tester know when the testing is done.
- White Box Testing techniques are easy to automate, this results in a developer having to hire fewer testers and smaller expenses.
- It shows bottlenecks which makes the optimization quite easy for the programmers.

6.2.2 BLACK BOX TESTING

As opposed to white box testing, black box testing is used to analyze the functionality of the software. It can be done in all levels of testing from unit testing to system testing. It is generally done for testing higher levels of code. It is also called Behavioral/Specification-Based/Input-Output Testing.

Advantages of Black Box Testing

- Black box tests are always executed from a user's point of view since it would help in exposing discrepancies significantly.
- Black box testers also do not need to know any programming languages.

6.3 TESTING LEVELS

Tests are grouped together based on the level of detailing they contain.The purpose of levels of testing is to make software testing systematic and easily identify all possible test cases at a particular level. In general, there are four levels of testing:

i. Unit Testing
ii. Integration Testing
iii. System Testing
iv. Acceptance Testing.

6.3.1 UNIT TESTING

A Unit is a smallest testable portion of a system or application which can be compiled, linked, loaded, and executed. This kind of testing helps to test each module separately.

6.3.2 INTEGRATION TESTING

Integration means combining. In this testing phase, different software modules are combined and tested as a group. Integrating testing checks the data flow from one module to other modules.

6.3.3 SYSTEM TESTING

System testing is performed on a complete, integrated system. It does checking of the system's compliance as per the requirements. It tests the overall interaction of components. It involves load, performance, reliability and security testing. System testing most often the final test to verify that the system meets the specification. It evaluates both functional and non-functional needs.

6.3.4 ACCEPTANCE TESTING

Acceptance testing is a test conducted to find if the requirements of a specification or contract are met as per its delivery. Acceptance testing is basically done by the user or customer. However, other stockholders can be involved in this process.

6.4 TESTING TYPES

There are two types of testing which are classified as such based on the manner in which the testing is done. They are:

i. Manual Testing

ii. Automation Testing

6.4.1 MANUAL TESTING

Manual testing is the process of testing software by hand. This usually includes verifying all the features specified in requirements documents, but often also includes the testers trying the software with the perspective of their end users in

mind. Manual test plans vary from fully scripted test cases, giving testers detailed steps and expected results to high-level guides that steer exploratory testing sessions.

6.4.2 AUTOMATION TESTING

Automation testing is the process of testing the software using an automation tool to find the defects. In this process, testers execute the test scripts and generate the test results automatically by using automation tools.

6.5 SMART CONTRACTS TESTING USING TRUFFLE

Truffle uses the Mocha testing framework and Chai for assertions to provide a solid framework to write JavaScript tests. The Mocha framework consists of the contract() function by which the smart contracts are redeployed to the running Ethereum client so the tests within it run with a clean contract state. Apart from this the artifacts.require() method, provided by Truffle, allows to request a usable contract abstraction for a specific Solidity contract. The truffle test command is executed to compile the smart contracts and check if they are deployed properly.

6.6 TEST RESULTS

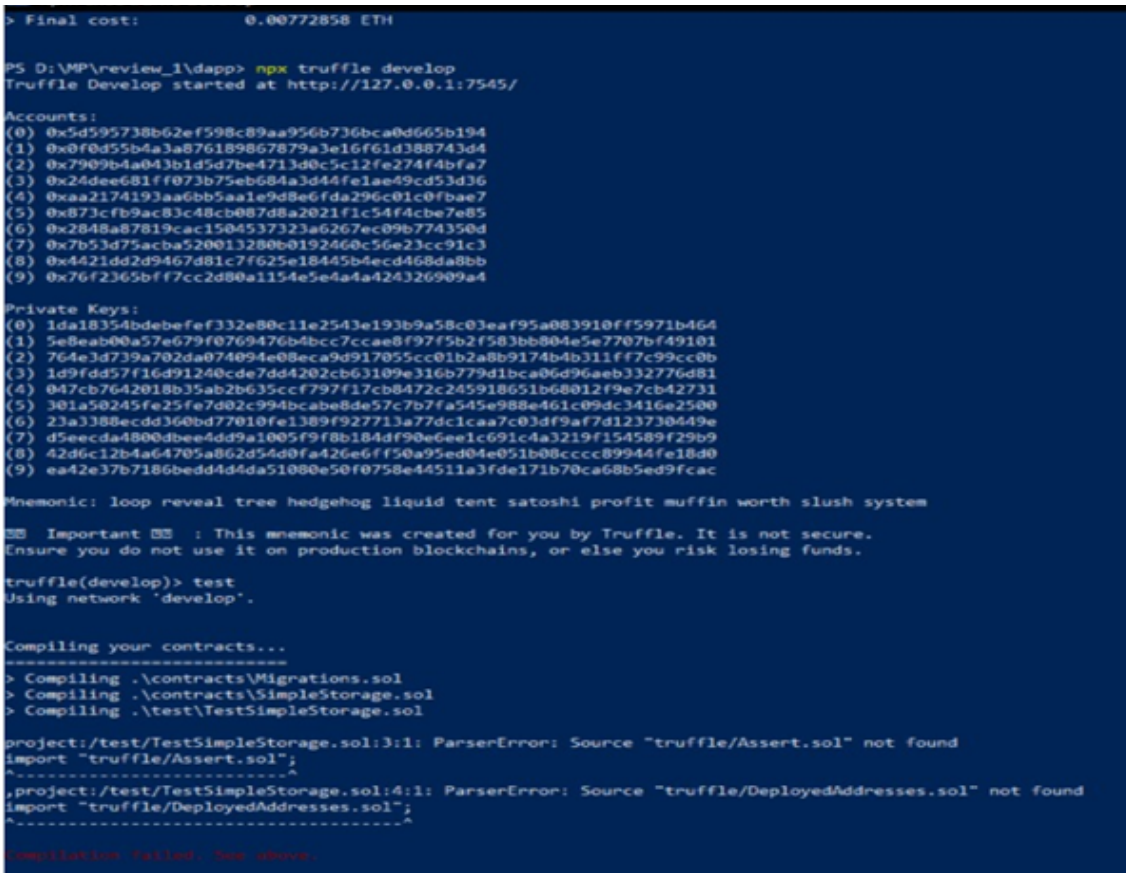


FIGURE 6. Truffle testing

CHAPTER 7
CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

A management system for storing documents is created with the help of React for creating the frontend and InterPlanetary File System (IPFS) for storing the person's private documents on a blockchain environment. The decentralised structure is achieved with the help of IPFS. Also the stimulation of the blockchain environment has been done using Ganache and a cryptocurrency wallet namely Metamask, an extension on the chrome browser that enables the user to clearly comprehend the gas price involved in each transaction. Also, a neural named entity recognition and multi-type normalization tool called BERN is used for hashing

7.2 FUTURE ENHANCEMENT

The proposed system does help the person in storing their documents securely. However, in the future, further enhancements are expected to be made, such as a similar portal on the many company ends and also another portal for insurance claims. Cloud storage may be utilised in the future for storage using Ethers, which is the native cryptocurrency of the Ethereum Blockchain platform. In the future, we will conduct the simulations of our system in the environment of private chains and public chains so that it can be closer to the actual applications.

APPENDIX-I

SCREENSHOTS

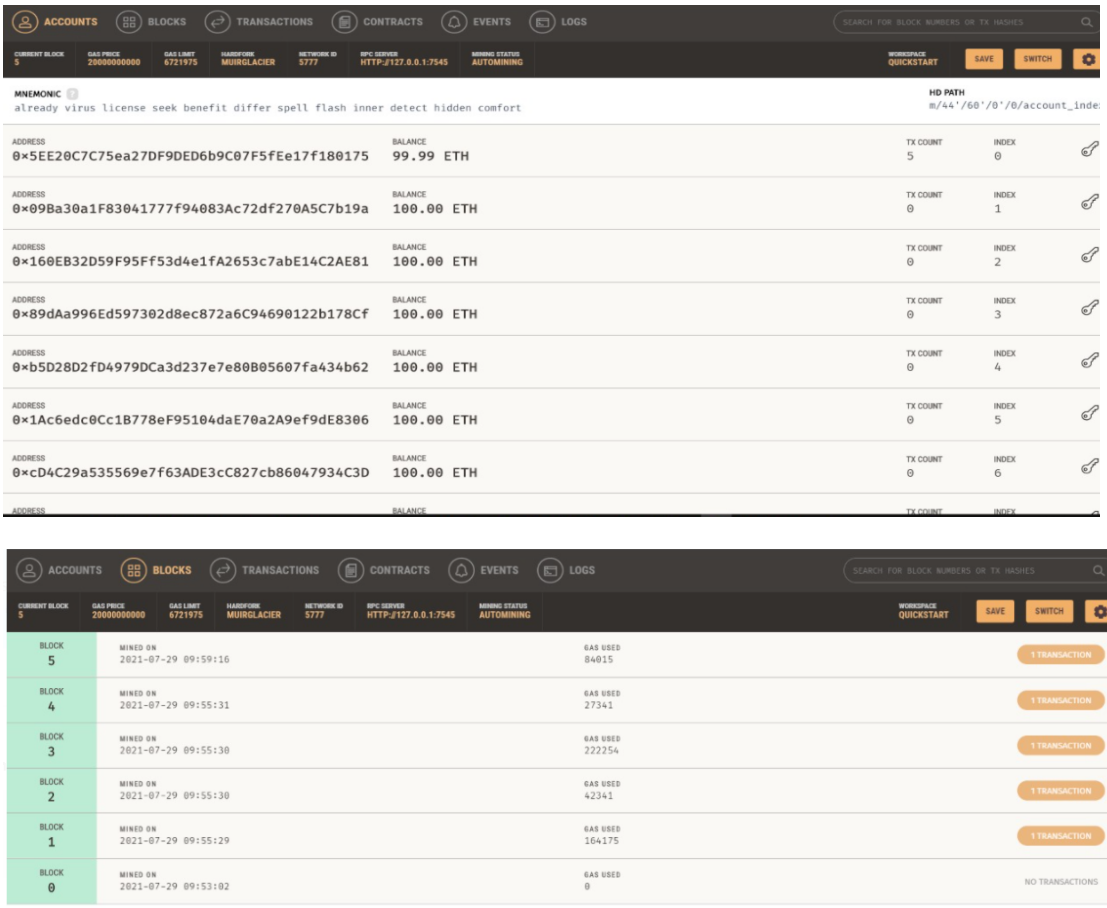


Figure A.1 LOCALIZED DEVELOPMENT BLOCKCHAIN.

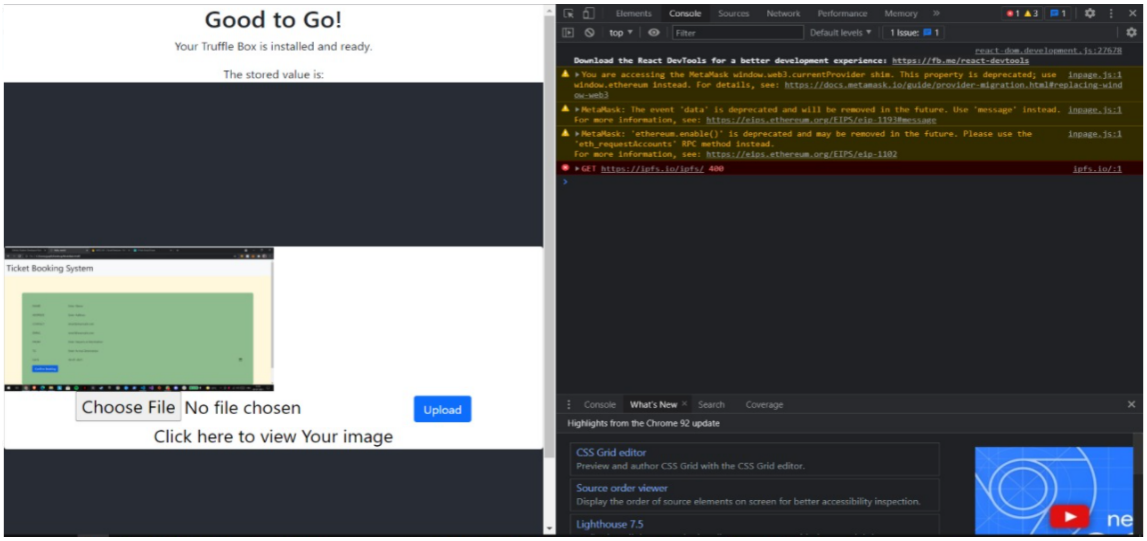


Figure A.2 CLIENT SERVER USER INTERFACE

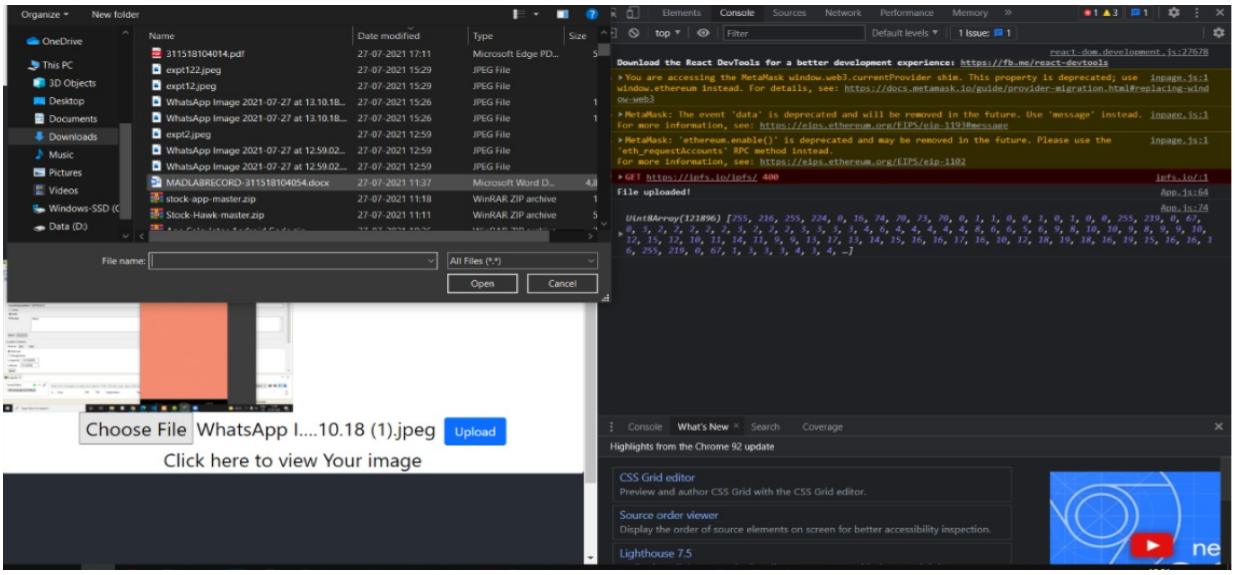
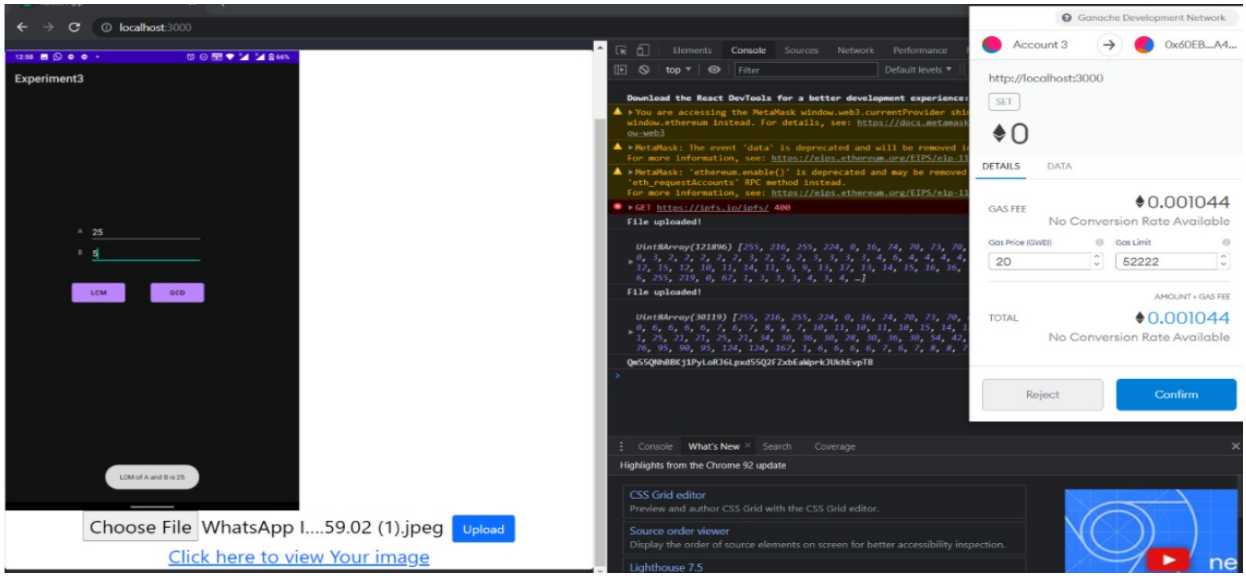


Figure A.3 USING DOCUMENT TO UPLOAD TO THE IPFS



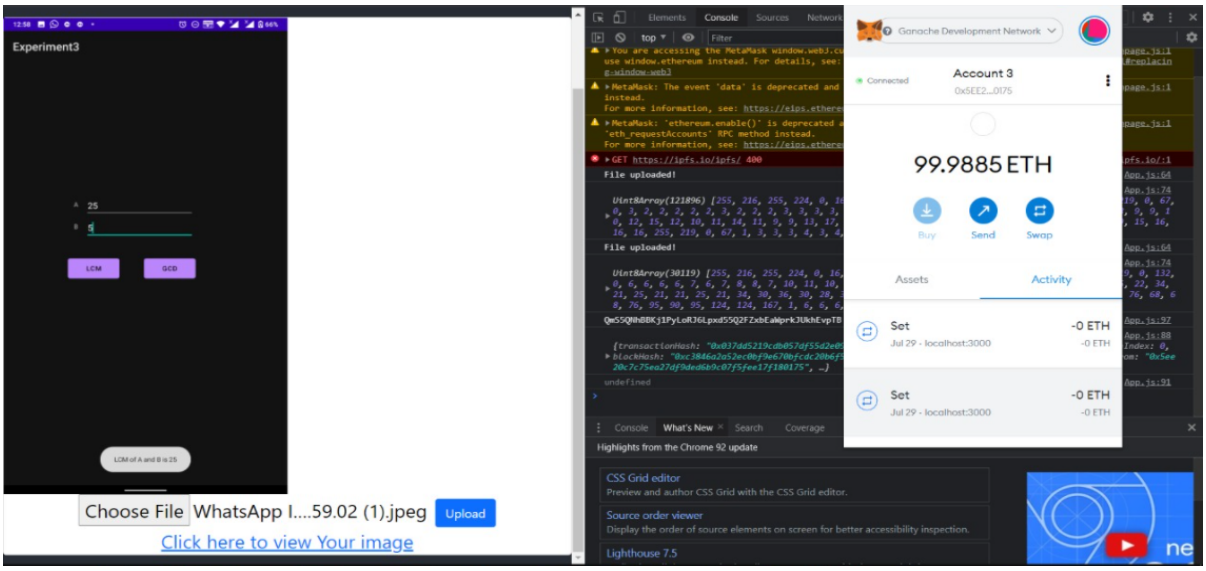


Figure A.4 AUTHORIZING PAYMENT TO STORE THE IPFS HASH IN THE BLOCKCHAIN

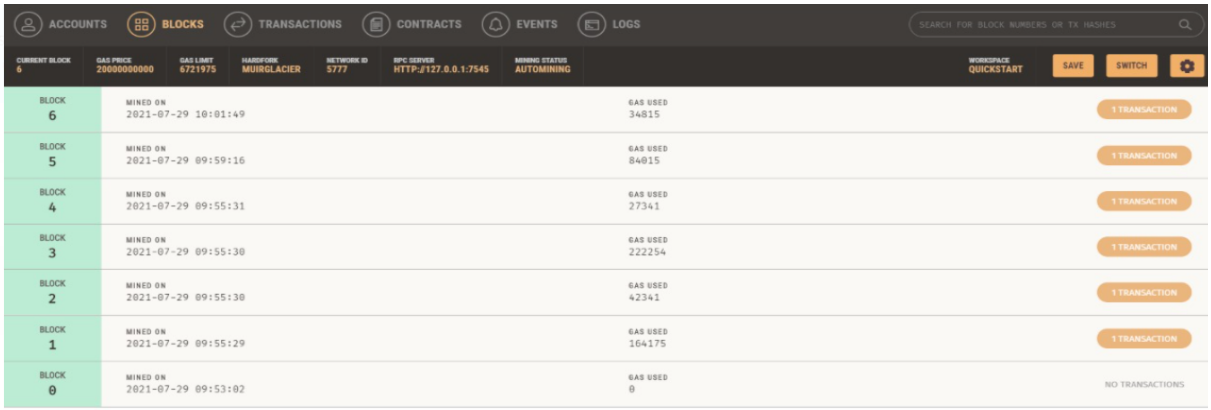


Figure A.5 NEW BLOCK CONTAINING THE IPFS HASH IS ADDED TO THE DEVELOPMENT BLOCKCHAIN

REFERENCES

- Meet Shah, Mohammedhasan Shaikh, Vishwajeet Mishra, Grinal Tuscano (2020) Decentralized Cloud Storage Using Blockchain
- R. Kumar and R. Tripathi, "Implementation of Distributed File Storage and Access Framework using IPFS and Blockchain," *2019 Fifth International Conference on Image Information Processing (ICIIP)*, 2019, pp. 246-251, doi: 10.1109/ICIIP47207.2019.8985677
- T. Gabriel, A. Cornel-Cristian, M. Arhip-Calin and A. Zamfirescu, "Cloud Storage. A comparison between centralized solutions versus decentralized cloud storage solutions using Blockchain technology," 2019 54th International Universities Power Engineering Conference (UPEC), 2019, pp. 1-5, doi: 10.1109/UPEC.2019.8893440.
- S. Jianjun, L. Ming and M. Jingang, "Research and application of data sharing platform integrating Ethereum and IPFs Technology," 2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), 2020, pp. 279-282, doi: 10.1109/DCABES50732.2020.00079.
- M. Steichen, B. Fiz, R. Norvill, W. Shbair and R. State, "Blockchain-Based, Decentralized Access Control for IPFS," 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2018, pp. 1499-1506, doi: 10.1109/Cybermatics_2018.2018.00253.

- R. Kumar and R. Tripathi, "Implementation of Distributed File Storage and Access Framework using IPFS and Blockchain," 2019 Fifth International Conference on Image Information Processing (ICIIP), 2019, pp. 246-251, doi: 10.1109/ICIIP47207.2019.8985677.