# A BLOCKCHAIN-BASED SECURE CLOUD FILES SHARING SCHEME WITH FINE-GRAINED ACCESS CONTROL

### **A PROJECT REPORT**

***Submitted by***

|  |  |
| --- | --- |
| **BAHADUR PRAVALLIKA** | **211418104032** |
| **DHANUSHA KARANAM** | **211418104048** |
| **SUNKESULA ELAHIN** | **211418104274** |

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**



## **PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**MAY 2022**

# PANIMALAR ENGINEERING COLLEGE

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**BONAFIDE CERTIFICATE**

Certified that this project report **“A BLOCKCHAIN-BASED SECURE CLOUD FILES SHARING SCHEME WITH FINE-GRAINED ACCESS CONTROL”** is the bonafide work of “**BAHADUR PRAVALLIKA [211418104032], DHANUSHA KARANAM [211418104048], SUNKESULA ELAHIN [211418104274]”** who

carried out the project work under my supervision.

# SIGNATURE SIGNATURE

### **Dr.S.MURUGAVALLI,M.E.,Ph.D** **Mrs.P.DEEPA**

HEAD OF THE DEPARTMENT SUPERVISOR

ASSOCIATE PROFESSOR

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE CHENNAI-600 123. CHENNAI-600 123.

Certified that the above mentioned students were examined in End Semester project viva-voice held on

.

### **INTERNAL EXAMINER EXTERNAL EXAMINER**

**DECLARATION BY THE STUDENT**

We BAHADUR PRAVALLIKA [211418104032], DHANUSHA KARANAM [211418104048], SUNKESULA ELAHIN[211418104274] hereby declare that this project report titled “**A BLOCKCHAIN-BASED SECURE CLOUD FILES SHARING SCHEME WITH FINE-GRAINED ACCESS CONTROL**”, under the guidance of **Mrs.P.DEEPA** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

**BAHADUR PRAVALLIKA**

**DHANUSHA KARANAM**

**SUNKESULA ELAHIN**

**ACKNOWLEDGEMENT**

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**,

**Dr.C.SAKTHI KUMAR,M.E.,Ph.D** and **Dr.SARANYASREE SAKTHI**

**KUMAR B.E.,M.B.A.,Ph.D.,** for providing us with the necessary facilities to

undertake this project.

We also express our gratitude to our Principal **Dr.K.MANI,M.E.,Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,**

for the support extended throughout the project.

We would like to thank our Coordinator **Dr. L. JABASHEELA,M.E.,Ph.D**., Project Guide **Mrs.P.DEEPA** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

**BAHADUR PRAVALLIKA**

**DHANUSHA KARANAM**

**SUNKESULA ELAHIN**

**ABSTRACT**

As cloud services greatly facilitate file sharing online, there's been a growing awareness of the security challenges brought by outsourcing data to a third party. Traditionally, the centralized management of cloud service provider brings about safety issues because the third party is only semi-trusted by clients. Besides, it causes trouble for sharing online data conveniently. In this paper, the Block chain technology is utilized for decentralized safety administration and provide more user-friendly service. Apart from that, Cipher Text-Policy Attribute Based Encryption is introduced as an effective tool to realize fine-grained data access control of the stored files. Meanwhile, the security analysis proves the confidentiality and integrity of the data stored in the cloud server. Finally, we evaluate the performance of computation overhead of our system.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** | **v** |
|  | **LIST OF FIGURES** | **viii** |
|  | **LIST OF SYMBOLS** | **ix** |
| **1.** | **INTRODUCTION** | **1** |
|  | 1.1 Problem Definition | 2 |
| **2.** | **LITERATURE SURVEY** | 3 |
| **3.** | **SYSTEM ANALYSIS** | 8 |
|  | 3.1 Existing System | 9 |
|  | 3.2 Proposed System | 9 |
|  | 3.3 Feasibility Study | 10 |
|  | 3.4 Hardware Environment | 11 |
|  | 3.5 Software Environment | 11 |
| **4.** | **SYSTEM DESIGN** | 12 |
|  | 4.1 ER diagram | 13 |
|  | 4.2 Data Flow Diagram | 14 |
|  | 4.3 UML Diagram | 16 |
| **5.** | **SYSTEM ARCHITECTURE** | 20 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | 5.1 Module Design Specification | 23 |
|  | 5.2 Algorithms | 24 |
| **6.** | **SYSTEM IMPLEMENTATION** | 25 |
|  | 6.1 Client-side coding | 26 |
|  | 6.2 Server-side coding | 28 |
| **7.** | **SYSTEM TESTING** | 35 |
|  | 7.1 Unit Testing | 36 |
|  | 7.2 Integration Testing | 36 |
|  | 7.3 Test Cases & Reports | 37 |
| **8.** | **CONCLUSION** | 40 |
|  | 8.1 Conclusion | 41 |
|  | APPENDICES | 42 |
|  | A.1 Sample Screens | 42 |
|  | **REFERENCES** | 46 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG NO** | **FIGURE DESCRIPTION** | **PAGE NO** |
| 4.1 | ER Diagram | 13 |
| 4.2 | Data Flow Diagram | 14 |
| 4.2.1 | Level 0 | 15 |
| 4.2.2 | Level 1 | 15 |
| 4.2.3 | Level 2 | 15 |
| 4.3 | UML Diagrams | 16 |
| 4.3.1 | Use case Diagram | 16 |
| 4.3.2 | Class Diagram | 17 |
| 4.3.3 | Sequence Diagram | 18 |
| 4.3.4 | Activity Diagram | 19 |
| 5.1 | System Architecture | 21 |
| A.1 | Sample Screens | 42 |
| A.1.1 | Run To Execute | 42 |
| A.1.2 | Main Page | 43 |
| A.1.3 | Data Sharing Page | 43 |
| A.1.4 | Uploading File | 44 |
| A.1.5 | Block Creation | 44 |
| A.1.6 | Downloading File | 45 |
| A.1.7 | Disconnect and return to home page | 45 |

**LIST OF SYMBOLS**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | | | | | **DESCRIPTION** |
| 1. | Class | |  | | --- | | *Class Name*  *-attribute* | | *-attribute* | | *+operation*  *+operation*  *+operation* |   *+ public*  *-private*  *#protected* | | | | | Represents a collection of similar entities grouped together. |
| 2. | Association | E   |  |  |  | | --- | --- | --- | | Class A | NAM | Class B | |  |  |  |  |  | | --- | --- | --- | | Class A |  | Class B | |  | | | | | | Associations represents static relationships between classes. Roles representsthe way the two classes see each other. |
| 3. | Actor |  | | | | | It aggregates several classes into a single classes. |
|  |  |  | | | | |  |
|  | Class A |  | Class A |  |
|  | | |  |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| 4. | Aggregation | ClassB Class B | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | **Relation**  (uses) | **uses** | Used for additional process communication. |
| 6. | Relation (extends) | extends | Extends relationship is used when one use case is similar to  another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processs. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | F inal state of the object |

|  |  |  |  |
| --- | --- | --- | --- |
| 11. | Control flow |  | Represents various  control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Use case | Uses case | Interact ion between the system and external environment. |
| 14. | Component |  | Represents physical modules which is a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of  components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or acion. |

|  |  |  |  |
| --- | --- | --- | --- |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between  processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**CHAPTER 1**

**INTRODUCTION**

1. **INTRODUCTION**

**1.1 Problem Definition**

Most software systems rely on cloud data storage for management of the data. With expeditious growth of digital world cloud storage has turned out into most reliable and convenient way of storing data. Data stored using cloud storage is stored in centralized manner. A major advantage of traditional cloud storage is; it is not only easy to handle but also easy to access the data. The data stored on cloud storage can be easily accessed by number of devices at a time. This way of storing data can cause single point failure, denial of service attack which may further lead to unavailability of data. Developing a system with decentralized storage of data can definitely overcome problems like single point failure, data unavailability etc. File uploaded on IPFS is stored in decentralized manner and cryptographic hash key returned by IPFS is stored on Block chain. Further, only authenticated user can access that particular data on IPFS by successful decryption of cryptographic hash key stored on Block chain.

**CHAPTER 2**

**LITERATURE SURVEY**

1. **LITERATURE SURVEY**

**1**.**Title**: An Efficient Secure System for Fetching Data from the Outsourced Encrypted Databases.

**Year**: 2021

**Author**: Sultan Almakdi ,Brajendra Panda , Mohammed Alshehri, and AbdulwahabAlazeb.

**Technique**: Cybersecurity, Privacy-Preserving, Encrypted Databases, SQL Queries, Cloud Computing.

**Advantages**: practical and efficient at reducing both the computation and space overhead when compared with state-of-the-art systems like CryptDB.

**Drawback**: The major disadvantage of this research is that only attributes with numeric values but not with string values were considered. Moreover, such a system only supports select statements.

**2.Title:** A Multi-Level Security Access Control Model Based on Blockchain Technology

**Year:** 2021

**Author:** Xiang Yu, Zhangxiang Shu, Qiang Li, Jun Huang.

**Technique:**BC-BLPM prototype system using the Hyperledger Fabric.

**Advantages:** BC-BLPM is a multi-level secure access control model using blockchain technology, which has the advantages of decentralization, auditability and scalability.

**Drawback:** there are disadvantages in communication and computational overhead.

**3.Title:** Practical Medical Files Sharing Scheme Based on Blockchain and Decentralized Attribute-Based Encryption.

**Year:** 2021

**Author:** JIYU TAO AND LI LING.

**Technique:** Attribute-based encryption, blockchain, medical data sharing, access control, authorization.

**Advantages:** Attribute-based algorithm that support multi-person democratic decision making and dynamic personnel changes are designed to make the model much closer to the real scene.

**Drawback:** the data is stored in the data center of the cloud service provider where third-party service providers cannot be supervised.

**4.Title**:Blockchain-Based Multi-Party Authorization for Accessing IPFS Encrypted Data.

**Year**: 2020

**Author**: Ammar Ayman Battah, Mohammad Moussa Madine Hamad Alzabbi, Ibrar

**Technique**: Access control, authentication, encrypted les, multi-party authority.

**Advantages**: The distributed ledger of the proposed blockchain-based solution acts as

immutable evidence for all the transactions recorded on it. It provides traceability features for

access control related events.

**Drawback**: Some security concerns are not fully mitigated as they are very challenging.

**5.Title:** An Energy Trade Framework Using Smart Contracts

**Year:** 2020

**Author:** MoayadAloqaily, AzzedineBoukerche, Ouns Bouachir, Fariea Khalid, and Sobia

Jangsher

**Technique:** Energy trade framework using smart contracts.

**Advantages:** BC is an efficient, low-cost system due to its incorporation into microgrids,

which provide users with significantly lower energy costs than conventional systems.

**Drawback:** Block size is a bottleneck, and increasing it also increases the verification time

and probability of an orphan block.

**6.Title:** Decentralized Access Control for IoT Data Using Blockchain and Trusted

Oracles

**Year:** 2019

**Author****:** H. Albreiki, L. Alqassem, K. Salah, M. H. Rehman, D. Svetinovic

**Technique:** Trusted oracles, IoT data

**Advantages:**It increases system scalability as users and devices can be registered and

delisted from the system easily without the need to modify the system architecture.

**Drawback:**Data sharing may escalate the security risks and privacy concerns since

malicious users can gain illegal access to the data and perform unauthorized alteration.

**7.Title:** Security and Privacy in Decentralized Energy Trading through Multi-

Signatures, Blockchain and Anonymous Messaging Streams

**Year:** 2019

**Author:** Nurzhan ZhumabekulyAitzhan and Davor Svetinovic,

**Technique:** Smart grid systems, decentralized energy trading.

**Advantages**: Decentralized microgrids combined with digital currencies such as

Bitcoin can lead to a faster and more robust solution to power problems in such

environments and extreme conditions.

**Drawback:** Here are a number of technology-related issues that are visible.

**8.Title:**Blockchain-based Soybean Traceability in Agricultural Supply Chain.

**Year**: 2019

**Author**: K. Salah, N. Nizamuddin, R. Jayaraman and M. Omar

**Technique:** Agricultural Supply Chain, Food Safety, Blockchain, Smart contracts,

Traceability.

**Advantages**: Presents a blockchain-based solution and framework for traceability

and visibility in the soybean supply chain using Ethereum smart contracts

**Drawback**: Presented aspects and details are generic enough and can be applied to

provide trusted and decentralized traceability.

**9.Title:**Decentralized Access Control for IoT Data Using Blockchain and Trusted Oracles

**Year:** 2019

**Author:** H. Albreiki, L. Alqassem, K. Salah, M. H. Rehman, D. Svetinovic

**Technique:** trusted oracles, IoT data

**Advantages:** it increases system scalability as users and devices can be registered and

delisted from the system easily without the need to modify the system architecture.

**Drawback:** data sharing may escalate the security risks and privacy concerns since

Malicious users can gain illegal access to the data and perform unauthorized alteration.

**10.Title:** Authentication with Block-Chain Algorithm and Text Encryption Protocol in

Calculation of Social Network.

**Year:** 2017

**Author:** RUIGUO YU1, JIANRONG WANG, TIANYI XU, JIE GAO, YONGLI

GONG ZHANG, AND MEI YU.

**Technique:** Information protection, block chain, hash encryption, text encryption

protocol.

**Advantages:** Authentication mechanism based on the block-chain, and encrypt the

relationship with Hash function for better security.

**Drawback:** When someone forges the identity and key to match, the user cannot be

identified effectively, resulting in his private information being leaked.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.SYSTEM ANALYSIS**

**3.1 Existing system**

In traditional mechanism of centralized data handling system, it inevitably inherits the single point of failure drawback of relying on third party services. In some cases, cloud storage systems are backed up to avoid data unavailability. In many cases data security is at stake because cloud storage services providers need to suffer from unnecessary disputes such as political censorship.

**Dis Advantage**

* As users no longer physically possess the storage of their data, traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted.
* They do not perform the multiple auditing task in simultaneously.

**3.2 Proposed System**

The proposed system consists of Interplanetary File System (IPFS), Block chain and Smart Contracts. This architecture depicts exact file sharing mechanism proposed by this decentralized web application. Smart Contracts in the system are kept centre because they are responsible for carrying out different data operations in a secured manner

**Advantage**

* IPFS over traditional data storage system is that there is no central server
* High security provide for file sharing.

**3.3 Feasibility Study**

**3.3.1 Technical Feasibility Study**

**3.3.1.1 PYTHON**

Python simplifies developers' lives as it is a scripted language and doesn't need to be compiled. Python also offers the option of pre-compiling the code, and this makes it helpful for developers to work in blockchain.

**3.3.1.2 ETHEREUM BLOCKCHAIN**

Ethereum is a decentralized blockchain platform that establishes a peer-to-peer network that securely executes and verifies application code, called smart contracts. Smart contracts allow participants to transact with each other without a trusted central authority.

**3.3.1.3 IPFS**

The Inter-Planetary File System is a protocol and peer-to-peer network for storing and sharing data in a distributed file system. It relies on cryptographic hashes that can easily be stored on a blockchain.

**3.3.2 Economic Feasibility Study**

This blockchain technology also claims to speed up and reduce the cost of transactions, and boost financial inclusion by providing more opportunities for those without easy access to financial services.

**3.3.3 Social Feasibility Study:**

High Security provided for file sharing and if once the data is entered it cannot be changed or tampered.

**3.4 Hardware Requirements**

* 2 GB RAM
* 80 GB Hard Disk
* Above 2GHz Processor

**3.5 Software Requirements**

* Flask
* HTML5 , CSS , BOOTSTRAP 3
* Python 3

**CHAPTER 4**

**SYSTEM DESIGN**

**4.SYSTEM DESIGN**

**4.1 ER DIAGRAM**

**ER Diagram** stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique. The purpose of ER Diagram is to represent the entity framework infrastructure.

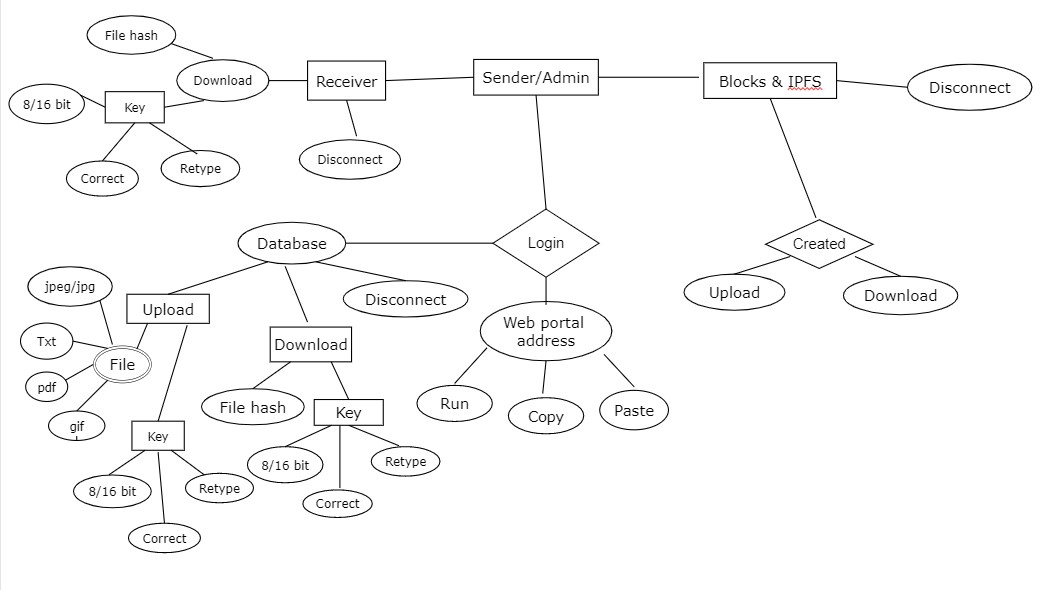


Fig.4.1 –ER Diagram

* 1. **DATA FLOW DIAGRAM**

A data flow diagram (DFD) is a graphical representation of the “flow” of data through an information system. It differs from the flowchart as it shows the data flow instead of the control flow of the program. A data flow diagram can also be used for the visualization of data processing. The DFD is designed to show how a system is divided into smaller portions and to highlight the flow of data between those parts.

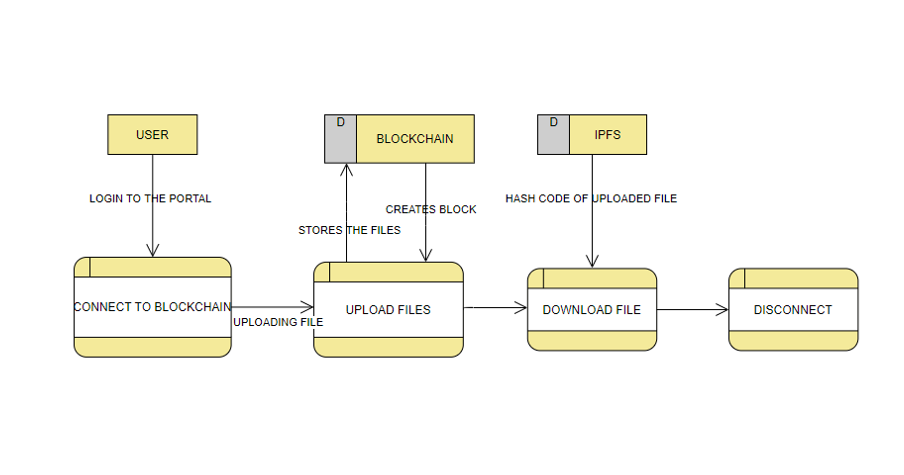
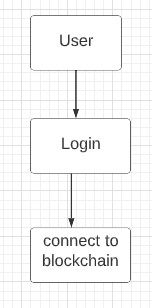
****

Fig 4.2 – Data Flow

**4.2.1 Level 0 4.2.1 Level 1**



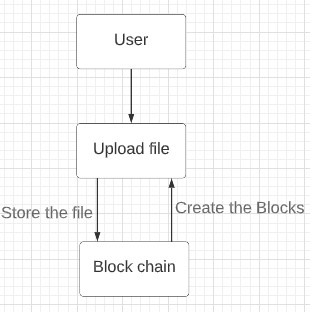
****

Fig 4.2.1-Level 0 Fig 4.2.2-Level1

**4.2.3 Level 2:**

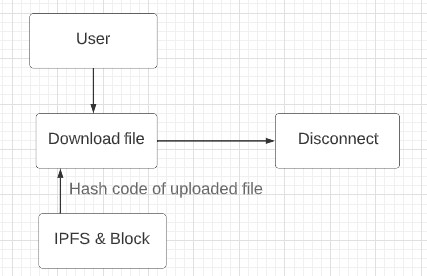
****

Fig 4.2.3-Level 2

* 1. **UML DIAGRAMS**
     1. **Usecase Diagram**

The use case diagram is the main building block of[objectoriented](http://en.wikipedia.org/wiki/Object_oriented)modeling.Itisusedbothforgeneral[conceptualmodeling](http://en.wikipedia.org/wiki/Conceptual_model)ofthesystematicof the application, and for detailed modeling translating the models into [programmingcode](http://en.wikipedia.org/wiki/Programming_code).

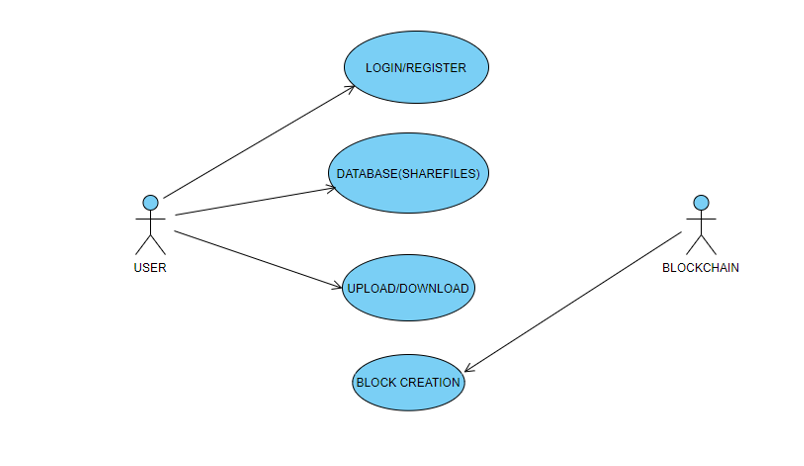
****

Fig 4.3.1 – Use case Diagram

* + 1. **Class Diagram**

Class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. The classes in a class diagram represent both the main objects and or interactions in the application and the objects.

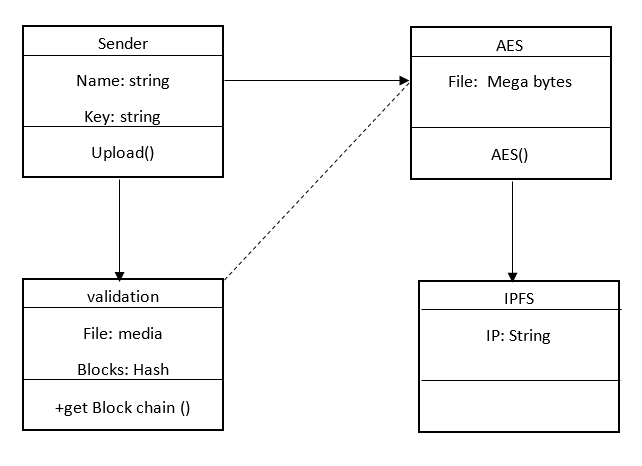
****

Fig.4.3.2 – Class Diagram

**4.2.3 Sequence Diagram**

A **sequence diagram** is the most commonly used **interaction**diagram.  An interaction diagram is used to show the **interactive behavior** of a system. Since visualizing the interactions in a system can be a cumbersome task, we use different types of interaction diagrams to capture various features and aspects of interaction in a system.

user

File upload /Download

Data base

Block Chain

Login Request

Valid User

Block Verified

Block created

Fig 4.3.2 – Sequence Diagram

* + 1. **Activity Diagram**

Activity diagram are a loosely defined diagram to show workflows of stepwise activitiesandactions,withsupportforchoice,iterationandconcurrency.UML,activity diagrams can be used to describe the business and operational step-by-step workflows ofcomponentsinasystem.UMLactivitydiagramscouldpotentiallymodeltheinternal logic of a complex operation. In many ways UML activity diagrams are the object- oriented equivalent of flow charts and data flow diagrams(DFDs)from structural development.

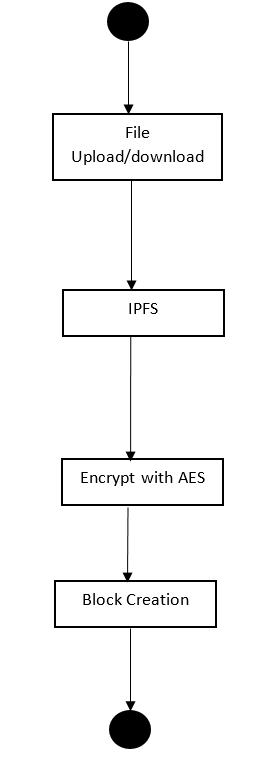


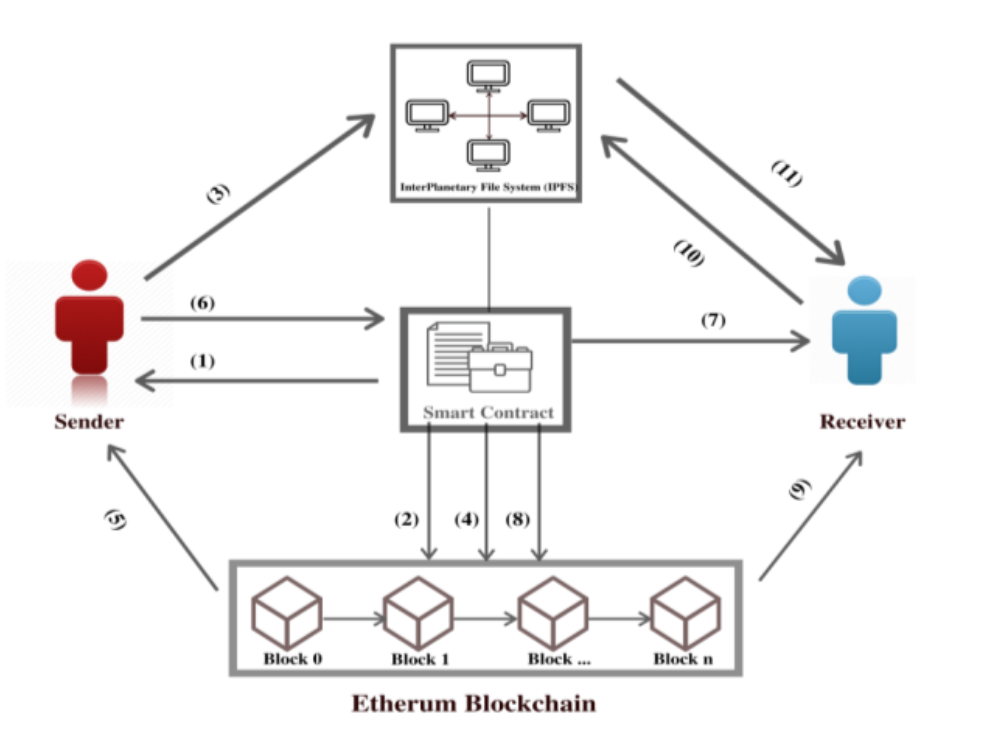
Fig 4.3.3 – Activity Diagram

**CHAPTER 5**

**SYSTEMARCHITECTURE**

**5. SYSTEM ARCHITECTURE**

**5.1 SYSTEM ARCHITECTURE DIAGRAM**



* 1. **MODULE DESIGN SPECIFICATION**
* Block Chain
* Peer to Peer Network
* AES
* Interplanetary File System

**5.1.1 Block chain**

In the proposed system, Block chain is used for authentication of users as well as to ensure integrity of data. As there are number of blocks in Block chain connected with each other in a chain like structure Block chain is called as Immutable way of storing data.

**5.1.2 Peer to Peer Network**

In order to create a peer to peer network (p2p) for the Block chain to function, all the connected nodes must be in the same network. Only those users who are connected to the block chain's p2p network should have access to the block chain's data. This p2p network is created using Socket Programming. We are working on a permissioned block chain which require access to be a part of the Block chain.

**5.1.3 AES**

We use the AES algorithm to generate a unique hash of the entire block that is used by the corresponding blocks to form the chain (via the previous hashes). IPFS as well uses this algorithm to generate the hash of the shared file.The AES Encryption algorithm (also known as the Rijndael algorithm) is a symmetric block cipher algorithm with a block/chunk size of 128 bits. It converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the cipher text.

**5.1.4 Interplanetary File System**

Our Block chain relies on IPFS for keeping it lightweight and scalable. If the files were stored directly on the Block chain, it would render the Block chain very heavy and inefficient. Combining IPFS and Block chain, we get to access the IPFS’s power of decentralized storage and enhance the block chain's security and accessibility. Instead of storing the file directly on the Block chain, we store the files on the IPFS network while the Block chain stores only the file’ hash

**5.2 ALGORITHMS:**

**AES ALGORITHM**

* The AES Encryption algorithm (also known as the advanced encryption standard algorithm) is a symmetric block cipher algorithm with a block/chunk size of 128 bits.
* It converts these individual blocks using keys of 128, 192, and 256 bits. Once it encrypts these blocks, it joins them together to form the cipher text.
* We use AES algorithm to generate a unique hash of the entire block that is used by the corresponding blocks to form the chain (via the previous hashes). IPFS as well uses this algorithm to generate the hash of the shared file.
* AES’s results are so successful that many entities and agencies have approved it and utilize it for encrypting sensitive information.
* AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.
* AES is a block cipher.
* The key size can be 128/192/256 bits.
* Encrypts data in blocks of 128 bits each.

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

**6.SYSTEM IMPLEMENTATION**

**6.1 Client-side coding**

**Blockchain.py**

#import datetime

import time

import hashlib

import json

from flask import Flask, jsonify, request

import requests

from urllib.parse import urlparse

# Building a Blockchain

class Blockchain:

def \_\_init\_\_(self):

self.chain = []

self.create\_block(proof = 1, previous\_hash = '0' , sender = 'N.A' , receiver = 'N.A' , file\_hash = 'N.A')

self.nodes = set()

self.nodes.add("127.0.0.1:5111")

def create\_block(self, proof, previous\_hash, sender, receiver, file\_hash):

block = {'index': len(self.chain) + 1,

'timestamp': str(time.strftime("%d %B %Y , %I:%M:%S %p", time.localtime())),

'proof': proof,

'previous\_hash': previous\_hash,

'sender': sender,

'receiver':receiver,

'shared\_files': file\_hash}

self.chain.append(block)

return block

def get\_previous\_block(self):

return self.chain[-1]

def proof\_of\_work(self, previous\_proof):

new\_proof = 1

check\_proof = False

while check\_proof is False:

hash\_operation = hashlib.sha256(str(new\_proof\*\*2 - previous\_proof\*\*2).encode()).hexdigest()

if hash\_operation[:4] == '0000':

check\_proof = True

else:

new\_proof += 1

return new\_proof

def hash(self, block):

encoded\_block = json.dumps(block, sort\_keys = True).encode()

return hashlib.sha256(encoded\_block).hexdigest()

def is\_chain\_valid(self, chain):

previous\_block = chain[0]

block\_index = 1

while block\_index<len(chain):

block = chain[block\_index]

if block['previous\_hash'] != self.hash(previous\_block):

return False

previous\_proof = previous\_block['proof']

proof = block['proof']

hash\_operation = hashlib.sha256(str(proof\*\*2 - previous\_proof\*\*2).encode()).hexdigest()

if hash\_operation[:4] != '0000':

return False

previous\_block = block

block\_index += 1

return True

def add\_file(self, sender, receiver, file\_hash):

previous\_block = self.get\_previous\_block()

index = previous\_block['index'] + 1

previous\_proof = previous\_block['proof']

proof = self.proof\_of\_work(previous\_proof)

previous\_hash = self.hash(previous\_block)

self.create\_block(proof, previous\_hash, sender, receiver, file\_hash)

return index

def replace\_chain(self):

network = self.nodes

longest\_chain = None

max\_length = len(self.chain)

for node in network:

response = requests.get(f'http://{node}/get\_chain')

if response.status\_code == 200:

length = response.json()['length']

chain = response.json()['chain']

if length >max\_length and self.is\_chain\_valid(chain):

max\_length = length

longest\_chain = chain

if longest\_chain:

self.chain = longest\_chain

return True

return False

**6.2 Server-side coding**

Server.py

import os

import urllib.request

import ipfshttpclient

from my\_constants import app

import pyAesCrypt

from flask import Flask, flash, request, redirect, render\_template, url\_for, jsonify

from flask\_socketio import SocketIO, send, emit

from werkzeug.utils import secure\_filename

import socket

import pickle

from blockchain import Blockchain

import requests

socketio = SocketIO(app)

blockchain = Blockchain()

def allowed\_file(filename):

return '.' in filename and filename.rsplit('.', 1)[1].lower() in app.config['ALLOWED\_EXTENSIONS']

def append\_file\_extension(uploaded\_file, file\_path):

file\_extension = uploaded\_file.filename.rsplit('.', 1)[1].lower()

user\_file = open(file\_path, 'a')

user\_file.write('\n' + file\_extension)

user\_file.close()

def decrypt\_file(file\_path, file\_key):

encrypted\_file = file\_path + ".aes"

os.rename(file\_path, encrypted\_file)

pyAesCrypt.decryptFile(encrypted\_file, file\_path, file\_key, app.config['BUFFER\_SIZE'])

def encrypt\_file(file\_path, file\_key):

pyAesCrypt.encryptFile(file\_path, file\_path + ".aes", file\_key, app.config['BUFFER\_SIZE'])

def hash\_user\_file(user\_file, file\_key):

encrypt\_file(user\_file, file\_key)

encrypted\_file\_path = user\_file + ".aes"

client = ipfshttpclient.connect('/dns/ipfs.infura.io/tcp/5001/https')

response = client.add(encrypted\_file\_path)

file\_hash = response['Hash']

return file\_hash

def retrieve\_from\_hash(file\_hash, file\_key):

client = ipfshttpclient.connect('/dns/ipfs.infura.io/tcp/5001/https')

file\_content = client.cat(file\_hash)

file\_path = os.path.join(app.config['DOWNLOAD\_FOLDER'], file\_hash)

user\_file = open(file\_path, 'ab+')

user\_file.write(file\_content)

user\_file.close()

decrypt\_file(file\_path, file\_key)

with open(file\_path, 'rb') as f:

lines = f.read().splitlines()

last\_line = lines[-1]

user\_file.close()

file\_extension = last\_line

saved\_file = file\_path + '.' + file\_extension.decode()

os.rename(file\_path, saved\_file)

print(saved\_file)

return saved\_file

@app.route('/')

def index():

return render\_template('index.html')

@app.route('/home')

def home():

return render\_template('index.html')

@app.route('/upload')

def upload():

return render\_template('upload.html' , message = "Welcome!")

@app.route('/download')

def download():

return render\_template('download.html' , message = "Welcome!")

@app.route('/connect\_blockchain')

def connect\_blockchain():

is\_chain\_replaced = blockchain.replace\_chain()

return render\_template('connect\_blockchain.html', chain = blockchain.chain, nodes = len(blockchain.nodes))

@app.errorhandler(413)

def entity\_too\_large(e):

return render\_template('upload.html' , message = "Requested Entity Too Large!")

@app.route('/add\_file', methods=['POST'])

def add\_file():

is\_chain\_replaced = blockchain.replace\_chain()

if is\_chain\_replaced:

print('The nodes had different chains so the chain was replaced by the longest one.')

else:

print('All good. The chain is the largest one.')

if request.method == 'POST':

error\_flag = True

if 'file' not in request.files:

message = 'No file part'

else:

user\_file = request.files['file']

if user\_file.filename == '':

message = 'No file selected for uploading'

if user\_file and allowed\_file(user\_file.filename):

error\_flag = False

filename = secure\_filename(user\_file.filename)

file\_path = os.path.join(app.config['UPLOAD\_FOLDER'], filename)

user\_file.save(file\_path)

append\_file\_extension(user\_file, file\_path)

sender = request.form['sender\_name']

receiver = request.form['receiver\_name']

file\_key = request.form['file\_key']

try:

hashed\_output1 = hash\_user\_file(file\_path, file\_key)

index = blockchain.add\_file(sender, receiver, hashed\_output1)

except Exception as err:

message = str(err)

error\_flag = True

if "ConnectionError:" in message:

message = "Gateway down or bad Internet!"

else:

error\_flag = True

message = 'Allowed file types are txt, pdf, png, jpg, jpeg, gif'

if error\_flag == True:

return render\_template('upload.html' , message = message)

else:

return render\_template('upload.html' , message = "File succesfully uploaded")

@app.route('/retrieve\_file', methods=['POST'])

def retrieve\_file():

is\_chain\_replaced = blockchain.replace\_chain()

if is\_chain\_replaced:

print('The nodes had different chains so the chain was replaced by the longest one.')

else:

print('All good. The chain is the largest one.')

if request.method == 'POST':

error\_flag = True

if request.form['file\_hash'] == '':

message = 'No file hash entered.'

elifrequest.form['file\_key'] == '':

message = 'No file key entered.'

else:

error\_flag = False

file\_key = request.form['file\_key']

file\_hash = request.form['file\_hash']

try:

file\_path = retrieve\_from\_hash(file\_hash, file\_key)

except Exception as err:

message = str(err)

error\_flag = True

if "ConnectionError:" in message:

message = "Gateway down or bad Internet!"

if error\_flag == True:

return render\_template('download.html' , message = message)

else:

return render\_template('download.html' , message = "File successfully downloaded")

# Getting the full Blockchain

@app.route('/get\_chain', methods = ['GET'])

def get\_chain():

response = {'chain': blockchain.chain,

'length': len(blockchain.chain)}

return jsonify(response), 200

@socketio.on('connect')

def handle\_connect():

print('Client connected')

print(request)

@socketio.on('add\_client\_node')

def handle\_node(client\_node):

print(client\_node)

blockchain.nodes.add(client\_node['node\_address'])

emit('my\_response', {'data': pickle.dumps(blockchain.nodes)}, broadcast = True)

@socketio.on('remove\_client\_node')

def handle\_node(client\_node):

print(client\_node)

blockchain.nodes.remove(client\_node['node\_address'])

emit('my\_response', {'data': pickle.dumps(blockchain.nodes)}, broadcast = True)

@socketio.on('disconnect')

def handle\_disconnect():

print('Client disconnected')

print(request)

if \_\_name\_\_ == '\_\_main\_\_':

socketio.run(app, host = '127.0.0.1', port= 5111, debug=True)

**My\_constants.py**

from flask import Flask

UPLOAD\_FOLDER = 'F:/secure file sharing/uploads'

DOWNLOAD\_FOLDER = 'F:/secure file sharing/downloads'

app = Flask(\_\_name\_\_)

app.secret\_key = "secret key"

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

app.config['DOWNLOAD\_FOLDER'] = DOWNLOAD\_FOLDER

app.config['ALLOWED\_EXTENSIONS'] = set(['txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif'])

app.config['BUFFER\_SIZE'] = 64 \* 1024

app.config['MAX\_CONTENT\_LENGTH'] = 16 \* 1024 \* 1024

**CHAPTER 7**

**SYSTEM TESTING**

**7. SYSTEM TESTING**

* 1. **UNITTESTING**

Unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In proceduralprogramming,aunitcouldbeanentiremodule,butitismorecommonlyan individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short codefragmentscreatedbyprogrammersoroccasionallybywhiteboxtestersduringthe developmentprocess.

Unit testing is software verification and validation method in which the individual units of source code are tested fit for use. A unit is the smallest testable part of an application. In this testing, each class is tested to be working satisfactorily.

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration.

# INTEGRATIONTESTING

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with. Individual modules, which are highly prone to interface errors, should not be assumed to work instantly when put together. The problem of course, is “putting them together”- interfacing. There may be the chances of data lost across on another’s sub functions, when combined may not produce the desired major function; individually acceptable impressionmaybemagnifiedtounacceptablelevels;globaldatastructurescanpresent problems.

Integrationtestingisthephaseinsoftwaretestinginwhichindividualsoftwaremodules are combined and tested as a group. Integration testing takes as its input modules that havebeenunittested,groupstheminlargeraggregates,appliestestsdefinedinan integrationtestplantothoseaggregates,anddeliversasitsoutputtheintegratedsystem ready. All the errors found in the system are corrected for the nextphase.

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystemsareexercisedthroughtheirinputinterface.Testcasesareconstructedtotest whether all the components within assemblages interact correctlyfor example across procedurecallsorprocessactivations,andthisisdoneaftertestingindividualmodules,

i.e. unit testing.

**7.3 TEST CASE & REPORT**

1. **LOGIN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **ACTION TO PERFORM** | **EXCEPTED RESULT** | **ACTUAL RESULT** | **RESULT(**  **PASS/FAIL)** |
| 1. | RUN | Run the code | As Excepted | Pass |
| 2. | Copy | Copy the web portal address | As Excepted | Pass |
| 3. | Paste | Paste the copied web portal address in web page and click enter | As Excepted | Pass |
| 4. | Home page | Connect to the Block chain | As Excepted | Pass |

1. **UPLOAD FILE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **ACTION TO PERFORM** | **EXCEPTED RESULT** | **ACTUAL RESULT** | **RESULT(**  **PASS/FAIL)** |
| 1. | Upload | After connecting to block chain, click on upload file | As Excepted | Pass |
| 2. | Sender(User) | Fill user details | As Excepted | Pass |
| 3. | Receiver | Fill receiver details | As Excepted | Pass |
| 4. | Key | Key value should be either 8 or 16 bit | As Expected | Pass |
| 5. | File | Choose a file to be uploaded | As Excepted | Pass |
| 6. | Submit | After the completion of uploading, click on submit. | As Excepted | Pass |

**3.DOWNLOAD FILE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.NO** | **ACTION TO PERFORM** | **EXCEPTED RESULT** | **ACTUAL RESULT** | **RESULT(**  **PASS/FAIL)** |
| 1. | Download | After the completion of uploading, click on download | As Excepted | Pass |
| 2. | File Hash | Copy the file hash of the uploaded file | As Excepted | Pass |
| 3. | Key | Fill the Key details(given by the Sender) | As Excepted | Pass |
| 4. | Submit | Click on submit button (after the completion of filling details) | As Excepted | Pass |
| 5. | Disconnect | Click on disconnect, return back to home page(After downloading file) | As Excepted | Pass |

**CHAPTER 8**

**CONCLUSION**

**8. CONCLUSION**

This project proposes the design and architecture of an advanced as well as secured web application for storing and sharing the data. A simple, affordable, easy to use and most secured system is proposed to solve the data security issues like integrity, authenticity and data unavailability.

**APPENDICES**

**A.1 SAMPLE SCREENS**

**A.1.1 RUN TO EXECUTE**

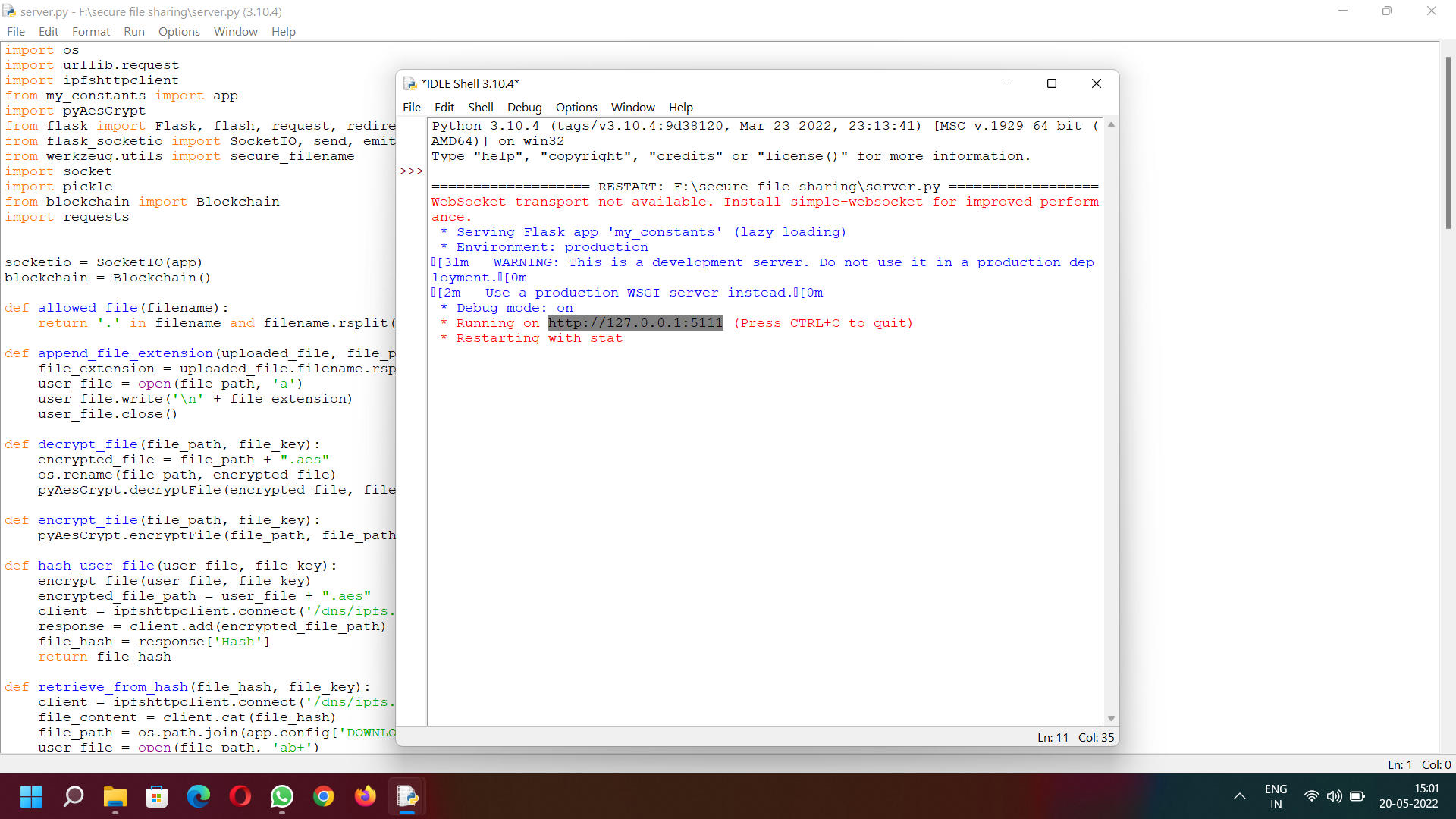
****

Fig.A.1.1-Run to Execute

**A.1.2HOME PAGE**

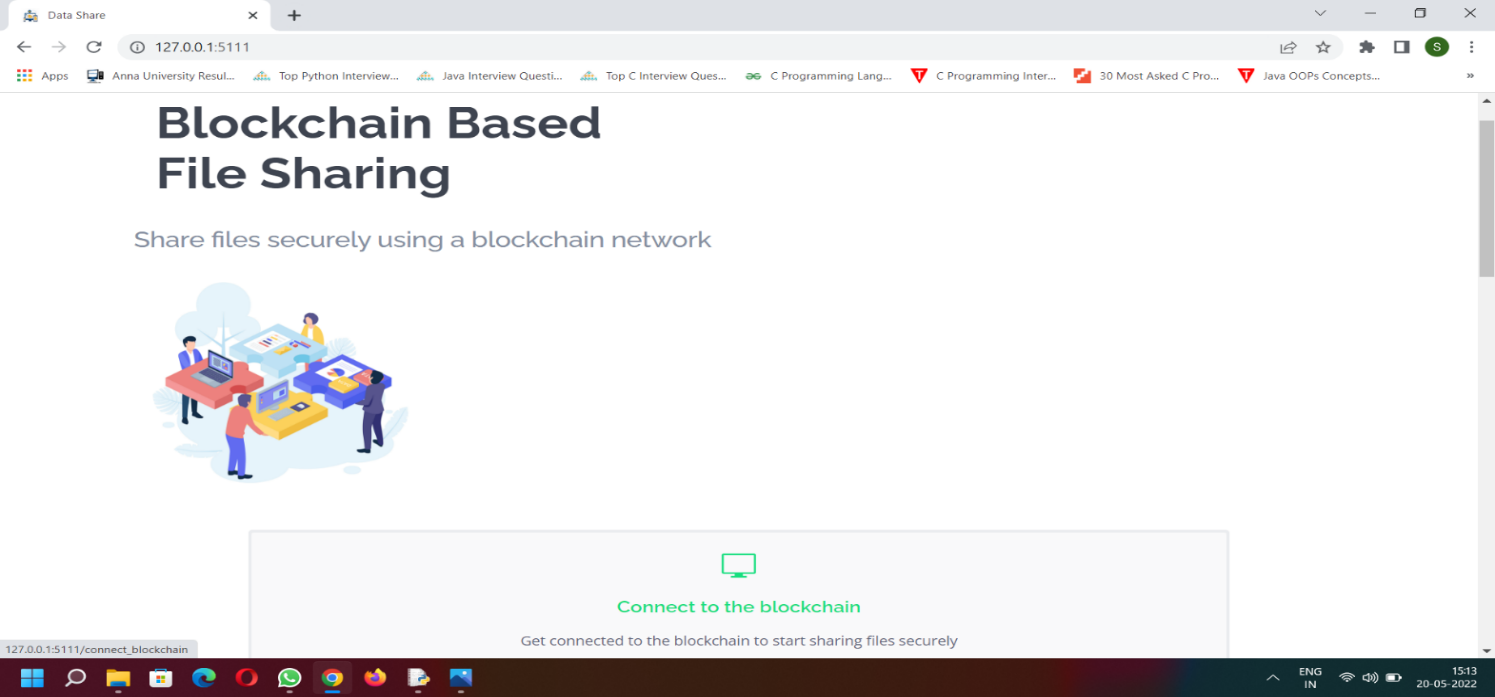


Fig.A.1.2-Main Page

**A.1.3 DATA SHARING PAGE**

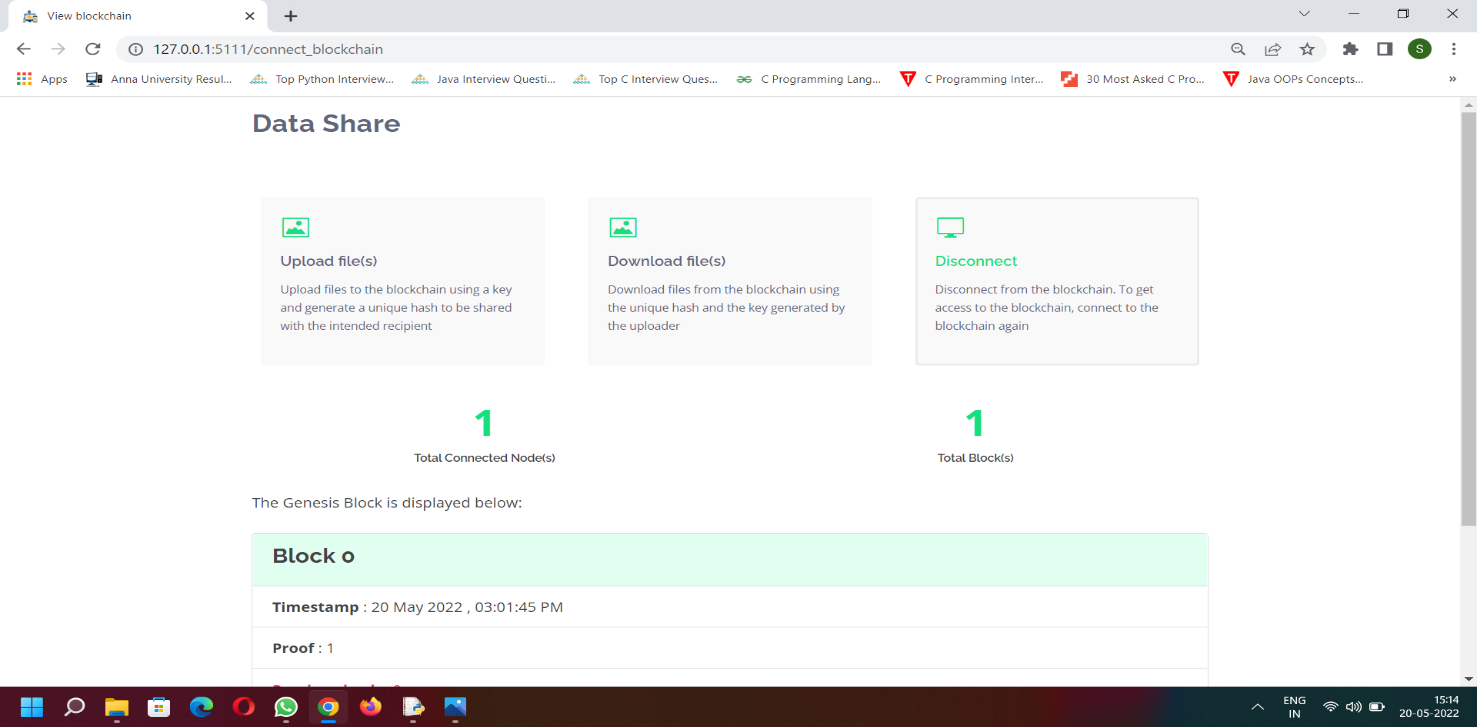
****

Fig.A.1.3-Data Sharing Page

**A.1.4 UPLOADING FILE**

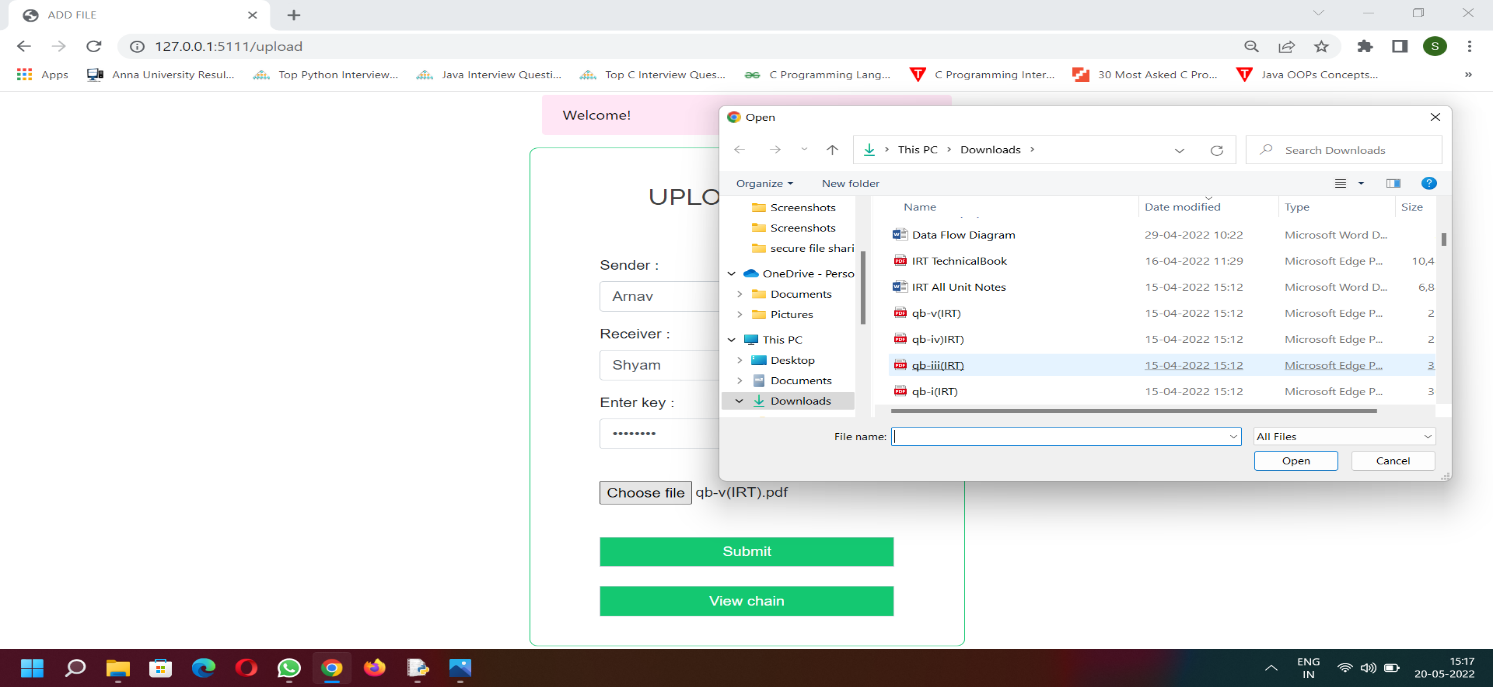
****

Fig.A.1.4-Uploading File

**A.1.5 BLOCK CREATION**

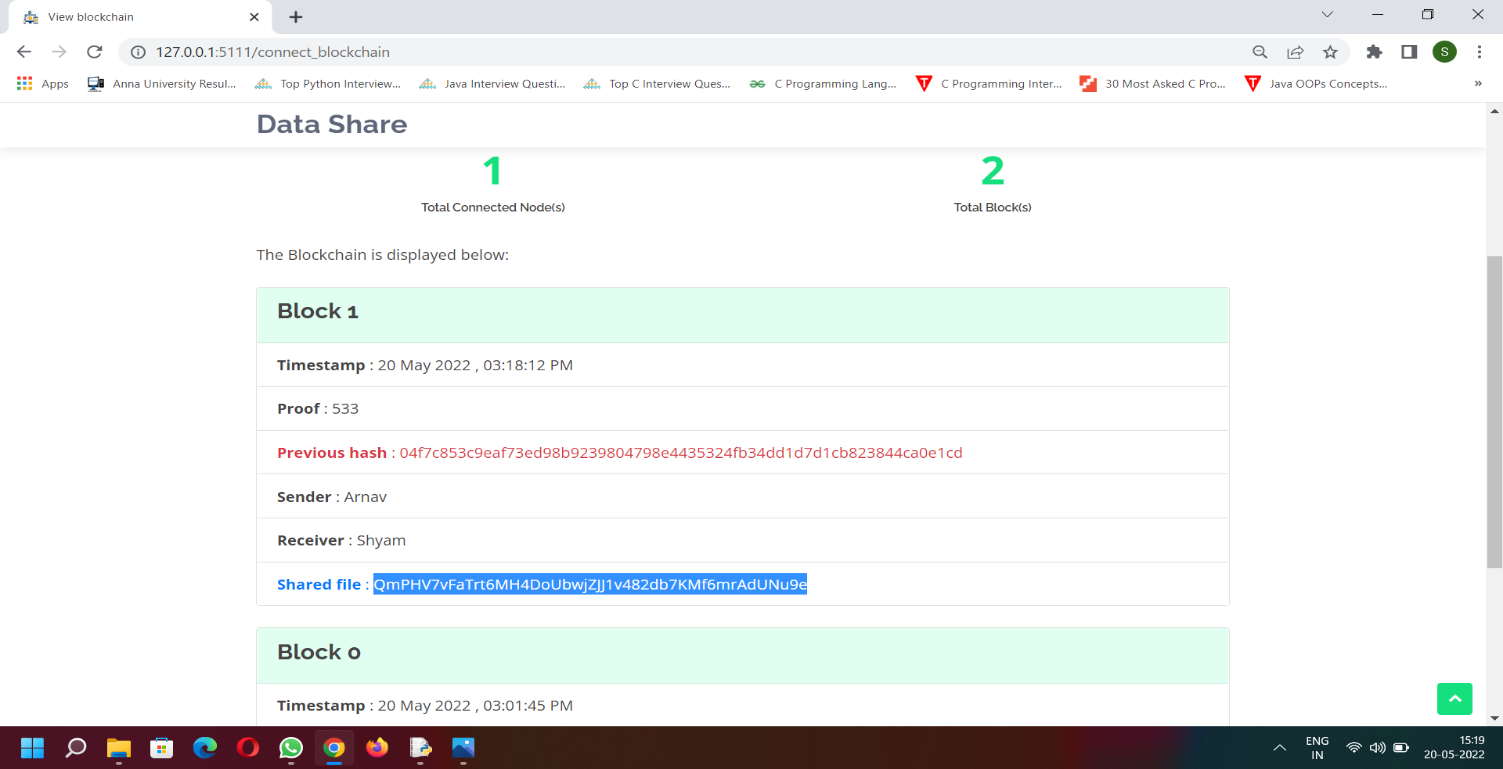
****

Fig.A.1.5-Block Creation

**A.1.6 DOWNLOADING FILE**

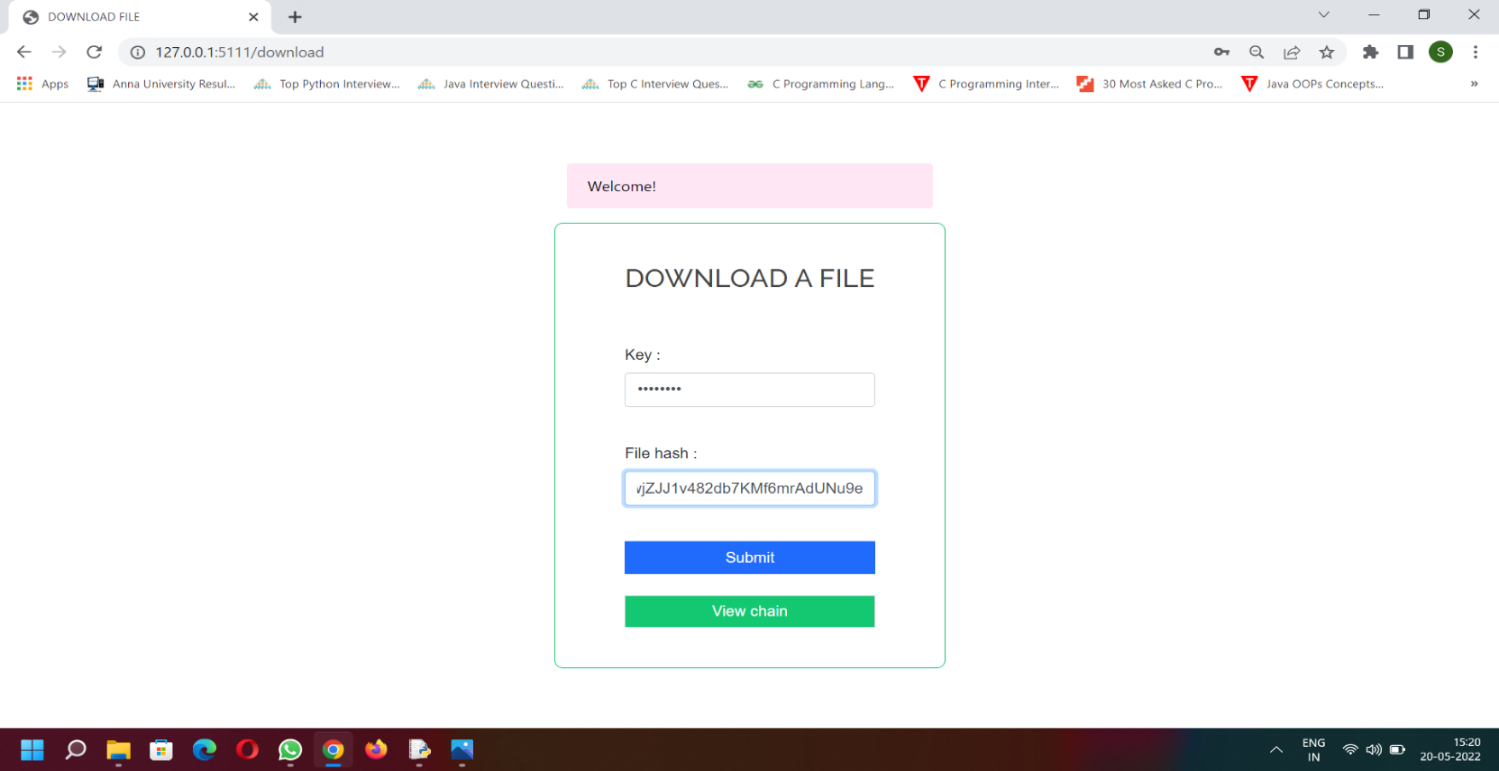
****

Fig.A.1.6-Downloading File

**A.1.7 DISCONNECT & RETURN TO HOME PAGE**

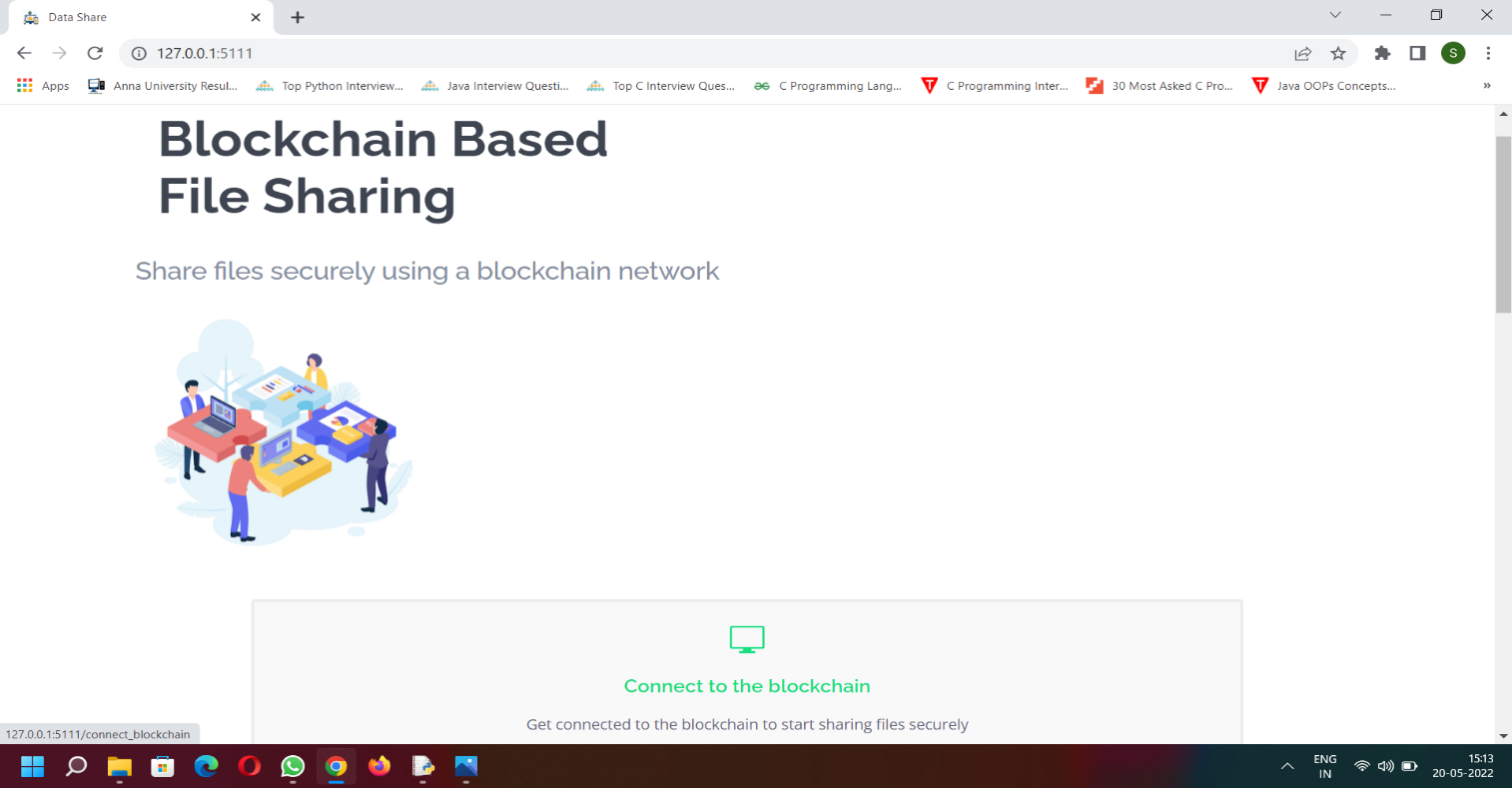
****

Fig.A.1.7-Disconnect and Return to Home Page

**REFERENCES**

[1]. Sultan Almakdi ,Brajendra Panda , Mohammed Alshehri, and AbdulwahabAlazeb. “An Efficient Secure System for Fetching Data from the Outsourced Encrypted Databases”, 2021.

[2].Xiang Yu, Zhangxiang Shu, Qiang Li, Jun Huang.“ A Multi-Level Security Access Control Model Based on Blockchain Technology”, 2021.

[3].H.Albreiki, L. Alqassem, K. Salah, M. H. Rehman, D. Svetinovic. “Decentralized Access Control for IoT Data Using Blockchain and Trusted Oracles”. 2019.

[4]. Nurzhan ZhumabekulyAitzhan and Davor Svetinovic.“Security Privacy in

Decentralized Energy Trading through Multi -Signatures,Blockchain and Anonymous

Messaging Streams. 2019

[5]. K. Salah, N. Nizamuddin, R. Jayaraman and M. Omar. “Blockchain - based Soybean

Traceability in Agricultural Supply Chain. 2019

[6]. RUIGUO YU1, JIANRONG WANG, TIANYI XU, JIE GAO, YONGLI,

“Authentication with Block-Chain Algorithm and Text Encryption Protocol inCalculation

of Social Network. 2017.

[7]. I. Agudo, D. Nu˜ nez, G. Giammatteo, P. Rizomiliotis, and C. Lambrinoudakis, “Cryptography goes to the cloud,” in Secure and Trust Computing, Data Management, and Applicat., 2011, pp. 190–197.

[8]. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, “Provable data possession at untrusted stores,” in Proc. 14th ACM conf. Compu. Commun. Security (CCS), 2007,pp.598–609.

[9]. G. Ateniese, A. Faonio, and S. Kamara, “Leakage-resilient identification schemes from zero-knowledge proofs of storage,” in IMA Inte. Conf. Cryptography and Coding, 2015, pp. 311–328.

[10]. G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, “Scalable and efficient provable data possession,” in Proc. 4th Int. Conf. Secure and Privacy in Commun. Netw. (SecureComm), 2008, pp. 1–10.

[11]. K. D. Bowers, A. Juels, and A. Oprea, “Proofs of retrievability:theory and implementation,” in Proc. 2009 ACM Workshop Cloud Computing Security (CCSW), 2009, pp. 43–54.

[12].O.Bouachir, M. Aloqaily, L. Tseng, and A. Boukerche, ‘‘Blockchain and fog computing for cyberphysical systems: The case of smart industry,’’ Computer, vol. 53, no. 9, pp. 36–45, Sep. 2020.

[13]. M. Aloqaily, A. Boukerche, O. Bouachir, F. Khalid, and S. Jangsher, ‘‘An energy trade framework using smart contracts: Overview and challenges,’’ IEEE Netw., vol. 34, no. 4, pp. 119–125, Jul. 2020.

[14]. Q. Chen, G. Srivastava, R. M. Parizi, M. Aloqaily, and I. A. Ridhawi, ‘‘An incentive-aware blockchain-based solution for Internet of fake media things,’’ Inf. Process. Manage., vol. 57, no. 6, Nov. 2020, Art. no. 102370.

[15]. H. R. Hasan and K. Salah, ‘‘Blockchain-based solution for proof of delivery of physical assets,’’ in Blockchain–ICBC, S. Chen, H. Wang, and L.-J. Zhang, Eds. Cham, Switzerland: Springer, 2018, pp. 139–152.