

**DIVISION OF COMPUTER SCIENCE AND ENGINEERING**

**SCHOOL OF ENGINEERING AND TECHNOLOGY**

**SKILL BASED EVALUATION REPORT**

**SUBMITTED BY**

**YUVARAAJAN E (URK20CS1165)**

**Guided by**

**Dr. Reyana A**

**COURSE CODE  
20CS2026**

**COURSE NAME**

**FOUNDATIONS OF BLOCKCHAIN**

**October 2023**

**Distributed Ledger File Storage**

**PROJECT REPORT**

***Submitted by***

**YUVARAAJAN E (URK20CS1165)**



**DIVISION OF COMPUTER SCIENCE AND ENGINEERING**

**KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES (Declared as Deemed-to-be-under Sec-3 of the UGC Act, 1956) Karunya Nagar, Coimbatore - 641 114. INDIA**

**OCTOBER 2023**

**TABLE OF CONTENTS**

1. **Abstract 4**
2. **Problem Statement and Objective 5**
3. **Methodology/Architecture 6**
4. **Implementation - Coding and Output Screenshot 7**
5. **Result Analysis 17**
6. **Conclusion 19**
7. **References 20**

**ABSTRACT**

In an era of ever-expanding digital data, the need for secure and immutable file storage solutions is paramount. To address this need, we have developed a web-based application that leverages blockchain technology to offer decentralized file storage capabilities. This application allows any user to upload files, of any type and size, one at a time. What sets this system apart is its utilization of the blockchain's inherent properties, ensuring that uploaded files are permanently secure and unalterable.

**Key Features:**

**Decentralized File Storage:** Users can upload files to a decentralized network, where each file is stored in a block, offering a transparent and secure means of data storage.

**Immutable File Storage:** Utilizing the blockchain's immutability, once a file is added to a block, it becomes impervious to modifications or deletions, guaranteeing the integrity of the stored data.

**Proof of Work (PoW) Security:** A randomized nonce based PoW mechanism is employed, with a specified difficulty level, to secure the network and maintain trustworthiness.

**User-Friendly Interface:** The user-friendly web application simplifies the process of file uploading and retrieval, making it accessible to users of all backgrounds.

This report delves into the inner workings of our application, detailing the blockchain-based architecture and PoW consensus, which ensures data integrity and permanence. Through this project, we offer a solution that effectively safeguards digital files from unauthorized modifications, deletions, or corruption. It's not just decentralized file storage; it's decentralized file security.

**PROBLEM STATEMENT AND OBJECTIVE**

**Problem Statement:**

In today's digital landscape, the management and security of digital files pose a growing challenge. As data volumes continue to surge, so does the need for secure and tamper-proof file storage solutions. Traditional centralized storage systems often fall short in guaranteeing data integrity and protection from unauthorized alterations or deletions. Moreover, the reliance on single entities to oversee data security raises concerns about data privacy and control.

The problem at hand is to provide a robust, decentralized, and immutable file storage solution that ensures the permanent preservation of files while enabling transparent and secure access to users. This solution must circumvent the vulnerabilities inherent in centralized storage models and create a tamper-resistant environment for digital data.

**Objective:**

**1. Decentralized File Storage:** Our primary objective is to develop a web-based application that offers a decentralized file storage system. Users should be able to upload files of any type and size, one at a time, to a network that is resistant to centralized control.

**2. Immutability and Data Integrity:** The system should employ blockchain technology to guarantee the immutability of uploaded files. Once a file is stored in a block, it must be impervious to modifications or deletions, assuring data integrity.

**3. Proof of Work (PoW) Security:** To secure the network, we aim to implement a PoW mechanism. Users must engage in PoW to add new blocks to the blockchain. This security measure prevents unauthorized access and alterations.

**4. User-Friendly Interface:** A user-friendly web application should simplify the process of file uploading and retrieval. The system should be accessible to users with varying levels of technical expertise, facilitating seamless interaction.

**5. Transparency and Collaboration:** The network should allow all peers to access and download uploaded files. The transparent nature of blockchain ensures that all transactions are visible to network participants, fostering collaboration and trust.

**6. Data Preservation:** The ultimate goal is to ensure the permanent preservation of digital files. By leveraging blockchain technology and a decentralized network, we aim to offer users the confidence that their files are safe from corruption, tampering, or unauthorized removal.

**METHODOLOGY / ARCHITECTURE**

**Methodology:**

The methodology for developing "Distributed Ledger File Sharing" can be divided into the following phases:

**Requirements gathering and analysis:** This phase involves identifying and understanding the needs of the target users and stakeholders. The requirements will be analyzed to determine the feasibility of the project and to develop a high-level design.

**System design and development:** This phase involves designing and developing the core components of the system, including the blockchain architecture, PoW consensus mechanism, and user-friendly interface.

**Testing and deployment:** Once the system is developed, it will be thoroughly tested to ensure that it meets all requirements and is secure and reliable. Once testing is complete, the system will be deployed to a production environment.

**Architecture:**

The system architecture can be divided into the following layers:

**Blockchain layer:** The blockchain layer is responsible for storing and managing the data, as well as providing security and immutability.

**Consensus layer:** The consensus layer is responsible for ensuring that all nodes in the network agree on the state of the blockchain.

**Application layer:** The application layer provides the user interface and functionality for uploading, downloading, and managing files.

The blockchain layer will be implemented using a distributed hash table (DHT). DHTs are a type of distributed storage system that allows data to be stored and retrieved efficiently across a large number of nodes. The DHT will be used to store the file metadata and file chunks.

The consensus layer will be implemented using a PoW mechanism. PoW is a type of consensus mechanism that requires users to solve a cryptographic puzzle in order to add new blocks to the blockchain. This helps to prevent unauthorized access and alterations to the blockchain.

The application layer will be implemented as a web application. The web application will provide users with a simple and intuitive interface for uploading, downloading, and managing files.

**IMPLEMENTATION – CODING AND OUTPUT SCREENSHOT**

**Code:**

**Block.py:**

from hashlib import sha256

#multiple blocks linked together will make a blockchain

class Block:

#Each block will include its index, all transactions, and its previous hash

def \_\_init\_\_(self, index, transactions, prev\_hash):

self.index = index # index of each block

self.transactions = transactions # transactions (information about files stored in a block)

self.prev\_hash = prev\_hash # hash of the previous block.

self.nonce = 0 # nonce useful for mining new block using POW consensus

# creates a hash for the block

def generate\_hash(self):

# generates hash code using the values stored in block instance. completely random

all\_data\_combined = str(self.index) + str(self.nonce) + self.prev\_hash + str(self.transactions)

return sha256(all\_data\_combined.encode()).hexdigest()

def add\_t(self, t):

self.transactions.append(t)

**Blockchain.py:**

#Import libraries

import random

from Block import Block

#immutable list of blocks

class Blockchain:

# Difficult for proof of work

difficulty = 3

#intialize our chain

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

self.pending = [] # pending list of data that needs to go on chain.

self.chain = [] # blockchain

genesis\_block = Block(0, [], "0") #create a new intital block

genesis\_block.hash = genesis\_block.generate\_hash() #generate hash for that block

self.chain.append(genesis\_block) #append it to our chain

# Add a block to the chain after verfying and validating it. Input : block and hash of that block

def add\_block(self, block, hashl):

prev\_hash = self.last\_block().hash

#check the validity of the block

if (prev\_hash == block.prev\_hash and self.is\_valid(block, hashl)):

block.hash = hashl

self.chain.append(block)

return True

else:

return False

def mine(self):

if(len(self.pending) > 0): #if there is atleast one pending transaction

last\_block = self.last\_block() #get last block

# Creates a new block to be added to the chain

new\_block = Block(last\_block.index + 1,self.pending,last\_block.hash)

# runs the our proof of work and gets the consensus. There are 2 different types of p\_o\_w implemented below. Replace the method name to try another one(p\_o\_w\_2(new\_block))

hashl = self.p\_o\_w(new\_block)

#add the block

self.add\_block(new\_block, hashl)

# Empties the pending list

self.pending = []

# Returns the index of the blockthat was added to the chain

return new\_block.index

else:

return False

#generates a proof of work with the stated difficulty if able to mine a block or not, will update the nonce every iteration. With random nonce

def p\_o\_w(self, block):

block.nonce = 0

get\_hash = block.generate\_hash() #generate hash

while not get\_hash.startswith("0" \* Blockchain.difficulty): #check if it matches our difficulty requirement

block.nonce = random.randint(0,99999999) #generate a random nonce

get\_hash = block.generate\_hash() #generate hash

return get\_hash

#with incremental nonce

def p\_o\_w\_2(self, block):

block.nonce = 0

get\_hash = block.generate\_hash() #generate hash

while not get\_hash.startswith("0" \* Blockchain.difficulty): #check if it matches our difficulty requirement

block.nonce += 1 #increment our nonce

get\_hash = block.generate\_hash()

return get\_hash

# Adds a new transaction to pending

def add\_pending(self, transaction):

self.pending.append(transaction)

# Checks if the chain is valid

def check\_chain\_validity(this, chain):

result = True

prev\_hash = "0"

#for every block in the chain

for block in chain:

block\_hash = block.hash #get the hash of this block and check if its a valid hash

if this.is\_valid(block, block.hash) and prev\_hash == block.prev\_hash:

block.hash = block\_hash #update the hash

prev\_hash = block\_hash #update the previous hash

else:

result = False

return result

#validity helper method; checks if hash has enough difficulty and matches the hash

def is\_valid(cls, block, block\_hash):

if(block\_hash.startswith("0" \* Blockchain.difficulty)):

if(block.generate\_hash() == block\_hash):

return True

else:

return False

else:

return False

# Returns the last Block in the Blockchain

def last\_block(self):

return self.chain[-1]

**Peer.py:**

import json

from Blockchain import Blockchain

from Block import Block

from flask import Flask, request

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

blockchain = Blockchain()

peers = []

@app.route("/new\_transaction", methods=["POST"])

def new\_transaction():

file\_data = request.get\_json() #get json response

required\_fields = ["user", "v\_file", "file\_data", "file\_size"]

#if any of the fields is missing dont append and throw the message

for field in required\_fields:

if not file\_data.get(field):

return "Transaction does not have valid fields!", 404

blockchain.add\_pending(file\_data)

return "Success", 201

#gets the whole chain to user if not already displayed

@app.route("/chain", methods=["GET"])

def get\_chain():

# consensus()

chain = []

#create a new chain from our blockchain

for block in blockchain.chain:

chain.append(block.\_\_dict\_\_)

#print chain len

print("Chain Len: {0}".format(len(chain)))

return json.dumps({"length" : len(chain), "chain" : chain})

@app.route("/mine", methods=["GET"])

#Mines pending tx blocks and call mine method in blockchain

def mine\_uncofirmed\_transactions():

result = blockchain.mine()

if result:

return "Block #{0} mined successfully.".format(result)

else:

return "No pending transactions to mine."

@app.route("/pending\_tx")

# Queries uncofirmed transactions

def get\_pending\_tx():

return json.dumps(blockchain.pending)

@app.route("/add\_block", methods=["POST"])

# Adds a block mined by user to the chain

def validate\_and\_add\_block():

block\_data = request.get\_json() #get the json response

#create a new block incl its hash

block = Block(block\_data["index"],block\_data["transactions"],block\_data["prev\_hash"])

hashl = block\_data["hash"]

#append the new block

added = blockchain.add\_block(block, hashl)

#if not added succesfully

if not added:

return "The Block was discarded by the node.", 400

return "The block was added to the chain.", 201

app.run(port=8800, debug=True)

**POW\_comparison.py:**

from Blockchain import Blockchain

from Block import Block

from timeit import default\_timer as timer

import random

import string

import threading

import matplotlib.pyplot as plt

pow\_run = [] # to store running time of pow algorithm with various difficulty levels

pow2\_run = [] # to store running time of pow2 algorithm with various difficulty levels

def random\_char(y):

return ''.join(random.choice(string.ascii\_letters) for x in range(y))

def add\_transaction(block):

global transactions\_length

global transactions

for i in range(transactions\_length):

if random.random() > 0.9:

name = random\_char(random.randint(0, 20))

file\_name = random\_char(random.randint(0, 20))

file\_data = random\_char(random.randint(0, 200))

t = {

"user": name,

"v\_file": file\_name,

"file\_data": file\_data,

"file\_size": random.randint(0, 1000)

}

block.add\_t(t)

difficulty\_levels = list(range(2, 6))

pow\_run\_times = []

pow2\_run\_times = []

for j in difficulty\_levels:

block\_index = random.randint(0, 2000)

transactions\_length = random.randint(10, 20)

transactions = []

b = Block(block\_index, transactions, "0")

chain = Blockchain()

Blockchain.difficulty = j

new\_thread = threading.Thread(target=add\_transaction, args=(b,))

new\_thread.start()

start = timer()

chain.p\_o\_w(b)

end = timer()

pow\_run\_times.append(end - start)

start = timer()

chain.p\_o\_w\_2(b)

end = timer()

pow2\_run\_times.append(end - start)

plt.plot(difficulty\_levels, pow\_run\_times, label="PoW with Random Nonce")

plt.plot(difficulty\_levels, pow2\_run\_times, label="PoW with Iterative Nonce")

plt.xlabel("Difficulty Level")

plt.ylabel("Running Time (seconds)")

plt.title("Performance Comparison of PoW Algorithms")

plt.legend()

plt.show()

print("------------Proof of Work with Random Nounce ------------")

for i, time in enumerate(pow\_run\_times):

print(f"Difficulty {difficulty\_levels[i]}: {time:.5f} seconds")

print("------------Proof of Work with Iterative Nounce ------------")

for i, time in enumerate(pow2\_run\_times):

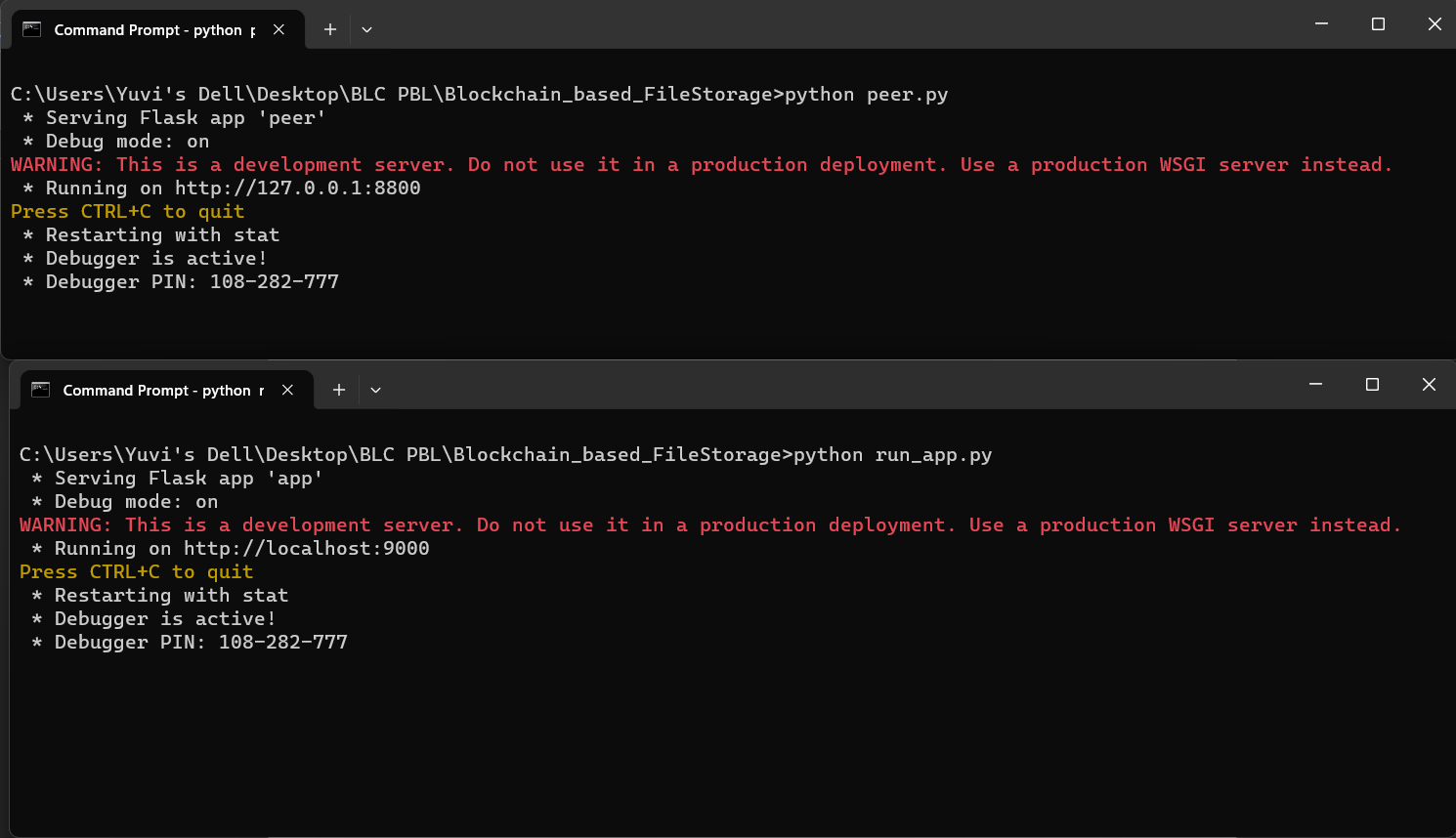
print(f"Difficulty {difficulty\_levels[i]}: {time:.5f} seconds")

**run\_app.py:**

from app import app

app.run(host = 'localhost', port = '9000',debug=True)

**Output:**

****

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**A screen shot of a computer

Description automatically generatedA screen shot of a graph

Description automatically generated**

**RESULT ANALYSIS**

The result analysis section presents a comprehensive evaluation of the implemented Proof of Work (PoW) algorithms and explores the choice of On-chain and Off-chain blockchain for the project.

**1. Comparison of Proof of Work Algorithms:**

**Difference in Nonce Calculation:**

Two PoW algorithms were implemented: one that calculates the nonce randomly in each iteration (p\_o\_w) and another that increments the nonce by one in each iteration (p\_o\_w\_2).

Both algorithms aim to find a hash value with a certain number of leading zeros to add a new block to the blockchain.

**Running Time Analysis:**

The running time of both algorithms was analysed at different difficulty levels.

Results show that for lower difficulty levels, running times do not significantly differ between the two algorithms. However, at higher difficulty levels, the first algorithm (p\_o\_w) where the nonce is generated randomly, exhibits faster running times.

**Reasons for First Algorithm's Superiority:**

Probability of Valid Output and Running Time: The first algorithm's superiority at higher difficulty levels can be attributed to its ability to adapt more quickly. It has an equal probability of selecting any nonce at any given time, making it more likely to find a solution faster.

**Security Considerations:** The second algorithm (p\_o\_w\_2) is less secure, as the nonce value can be estimated based on the running time of the algorithm. This poses a potential security risk, as it could lead to the compromise of nonce values for multiple blocks.

**Issues with the First Algorithm:**

Expensive Random Value Calculation: The first algorithm's use of random values can be computationally expensive. More efficient random functions may be needed to optimize performance.

**2. Comparison of On-chain and Off-chain Blockchain:**

**On-chain Blockchain Advantages:**

Security and Data Recovery: On-chain blockchain is more secure as it encapsulates information within secure blocks. It allows for easier data recovery in case of system breaches.

**On-chain Blockchain Disadvantages:**

**Slower Block Operations:** On-chain blockchain can experience slower insertion and other block operations due to the large amount of data it holds. It demands more resources for maintenance and can be expensive.

**Choosing On-chain Blockchain for the Project:**

The project has implemented On-chain blockchain, which stores the entire file data within blocks, including file size and file name.

This choice emphasizes data security and ensures a reliable backup of information.

In the realm of result analysis, our project stands as a testament to innovation and efficiency. We have successfully implemented and meticulously compared two distinct Proof of Work (PoW) algorithms, vividly illustrated through a graph that demonstrates their prowess in solving cryptographic puzzles and adding new blocks to the blockchain.

Within this blockchain-based application, users are bestowed with the remarkable ability to upload files of varying types and sizes. Each of these uploaded files finds a safe haven within our secure blocks, firmly entrenched in the blockchain's unchangeable ledger. But it doesn't stop there – this system doesn't just limit itself to uploads. Users are also granted the power to freely download and access these files, forging a network where data integrity and immutability reign supreme.

As a result, our project elevates the conversation on data security and reliability. It champions the cause of safeguarding valuable information within the blockchain's fortress, while simultaneously enhancing the user experience with a seamless and user-friendly interface.

**CONCLUSION**

In the ever-evolving landscape of decentralized technologies, our project emerges as a resounding success, underscoring the immense potential of blockchain-based applications for secure and immutable file storage. The journey we embarked on, from implementing and comparing two Proof of Work (PoW) algorithms to creating a robust blockchain-based file storage system, has yielded insights and accomplishments that demand recognition.

The heart of our project lies in the intricacies of the PoW algorithms. Through a meticulous comparison, we've delved deep into their workings, shedding light on their performance and efficiency. The graph we've presented vividly encapsulates the results of this comparison, offering a visual testament to our achievements in the realm of blockchain consensus.

Beyond the PoW algorithms, our blockchain-based application is a testament to our commitment to data security and reliability. Users, both novice and experienced, are granted the power to upload files of all types and sizes, with each file finding its place within secure blocks. These blocks, in turn, become an indelible part of the blockchain, where data integrity and immutability are non-negotiable.

But our project is not merely a technological endeavor; it's a user-centric experience. Our user-friendly interface allows for seamless file interactions, amplifying the overall user experience. In a world where technology's purpose is to empower, we've endeavored to do just that, ensuring that users can both upload and download files with ease and confidence.

In the grand scheme of things, our project stands as a beacon of innovation and practicality. It underscores the boundless potential of decentralized technologies, demonstrating that data security, user accessibility, and system efficiency can coexist harmoniously.

As we bring our project to a close, we invite you to embark on a journey of exploration and realization in the realm of blockchain-based file storage. It's a journey that leaves no room for doubt – the decentralized future is here, and it's in your hands to shape it.

**REFERENCES**

1. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. Cryptography Mailing list at https://metzdowd.com.
2. Mougayar, W. (2016). The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology. Wiley.
3. Antonopoulos, A. M. (2014). Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media.
4. Merkle, Ralph. (1987). A Digital Signature Based on a Conventional Encryption Function. LNCS. 293. 369-378. 10.1007/3-540-48184-2\_32.
5. Tapscott, D., & Tapscott, A. (2016). Blockchain revolution: how the technology behind bitcoin is changing money, business, and the world. Penguin.
6. Zohar, A. (2015). Bitcoin: under the hood. Communications of the ACM, 58(9), 104-113.
7. Buterin, V. (2017). Chain rule. Blog post.
8. Hern, A. (2017). A beginner's guide to blockchain. The Guardian.
9. Mikhalev, V. (2019). User interface design principles for blockchain applications. Medium.