BlockShareCloud: Blockchain-Based Decentralized Cloud Storage System

A project report

submitted to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

of

Bachelor of Technology

in

Information Technology

by

Anushree Jha

Reg. No. 225811164

Under the guidance of

Dr. Abhijeet Das
Assistant Professor
Department of Information Technology
Manipal Institute of Technology
Bengaluru, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Nov 2024

DECLARATION

I hereby declare that this project work entitled **BlockShareCloud: Blockchain-Based Decentralized Cloud Storage System** is original and has been carried out by me in the Department of Information and Communication Technology of Manipal Institute of Technology, Manipal, under the guidance of **Dr. Abhijeet Das**, **Assistant Professor – Senior Scale**, Department of Information and Technology, M. I. T., Bengaluru. No part of this work has been submitted for the award of a degree or diploma either to this University or to any other Universities.

Place: Manipal

Date: 29/10/2024

Anushree Jha

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

CERTIFICATE

This is to certify that this project entitled **BlockShareCloud: Blockchain-Based Decentralized Cloud Storage System** is a bonafide project work done by **Ms. Anushree Jha (Reg.No.:225811164)** at Manipal Institute of Technology, Manipal, independently under my guidance and supervision for the award of the Degree of Bachelor of Technology in Information Technology.

Dr. Abhijeet Das	Dr. Dayanand P
Assistant Professor	Professor & Head
Department of I & T	Department of I & T
Manipal Institute of Technology	Manipal Institute of Technology
Bengaluru, India	Bengaluru, India

Table of Contents

1. Introduction

1.1 Background

The project aims to solve two significant issues in modern cloud storage: data breaches and centralized control over user files. Centralized cloud storage systems, while convenient, often have vulnerabilities in terms of security, ownership, and data manipulation. If a single storage provider is compromised, sensitive user data could be at risk. In addition, centralized providers may be susceptible to outages, attacks, or surveillance.

To combat these issues, BlockShareCloud utilizes blockchain technology and the InterPlanetary File System (IPFS) to offer a secure, decentralized alternative. Blockchain technology ensures data integrity through immutable records of all file transactions. IPFS, a peer-to-peer file storage protocol, allows users to store and retrieve files in a decentralized network, removing the need for reliance on a central authority. Files are content-addressed, meaning that their location is irrelevant, as long as the correct cryptographic hash exists.

BlockShareCloud ensures that files are uploaded and stored in IPFS, while the blockchain logs the file metadata (such as the file name and its corresponding IPFS hash). This decentralized approach greatly enhances data security, prevents unauthorized tampering, and provides verifiable traceability for every uploaded file.

1.2 Objectives

The project has several key objectives:

- Develop a decentralized cloud storage solution that integrates IPFS and blockchain for enhanced security and traceability.
- Ensure user files are securely uploaded and stored in a decentralized manner.
- Provide an intuitive web interface that allows users to easily upload files to IPFS, retrieve IPFS hashes, and verify file authenticity using the blockchain.
- Ensure tamper-proof storage of file metadata via blockchain.

1.3 Scope

The current implementation of BlockShareCloud focuses on proof-of-concept functionality:

- **Decentralized Storage:** IPFS is used to store files in a decentralized network, removing dependency on a single server or cloud provider.
- **Blockchain Integration:** A custom, in-memory blockchain is used to log file metadata, ensuring verifiable traceability.

- **Web Interface:** A simple web interface is built for file upload and retrieval, ensuring ease of use for non-technical users.

While the system guarantees file integrity through blockchain metadata logging, it is currently limited to an in-memory blockchain, which does not persist data across server restarts. Future work may involve integrating a persistent blockchain solution, enhancing scalability, and expanding security features.

2. Literature Review

2.1 Overview

The increasing demand for secure and scalable cloud storage solutions has driven research into decentralized systems. Projects like *Filecoin, Sia,* and *Storj* have been pioneers in the field of decentralized file storage. Filecoin, built on IPFS, incentivizes storage by creating a marketplace where users can sell and buy unused storage space. Sia, on the other hand, utilizes smart contracts to ensure that storage providers are held accountable for file retention. Both projects share a common goal: reducing dependence on centralized servers while ensuring data integrity and security.

Blockchain technology, best known through *Bitcoin* and *Ethereum*, has revolutionized how we think about trust and traceability in digital transactions. Bitcoin introduced the concept of a decentralized, immutable ledger, and Ethereum expanded on this idea by implementing smart contracts that allow for automated and secure execution of agreements. The blockchain ensures that all transactions, including metadata for file storage, are stored in an unchangeable manner, preventing unauthorized tampering or deletion. These technologies offer the foundational principles that BlockShareCloud leverages: decentralization, security, and traceability.

2.2 Key Technologies

1. **Blockchain**: BlockShareCloud's use of blockchain ensures the ***immutability*** of file metadata (such as IPFS hashes). The concept of ***hashing*** is central here. In blockchain, a hash of each block is used to chain it to the previous block, ensuring that the entire chain remains tamper-proof. If any change is attempted in a block, it would invalidate the hash of that block and all subsequent blocks, making tampering evident and impossible without re-computing the entire chain's hash.

A key concept related to the project is **Proof of Work (PoW)** and **Proof of Stake (PoS)**, which are consensus mechanisms that ensure the validity of transactions on the blockchain. While PoW is more energy-intensive (used in Bitcoin), PoS is energy-efficient (used in newer blockchains like Ethereum 2.0). Though BlockShareCloud employs an in-memory blockchain without full consensus mechanisms, these theories lay the groundwork for potential future enhancements.

2. **IPFS (InterPlanetary File System)**: IPFS is central to BlockShareCloud for its decentralized storage capabilities. Unlike traditional client-server models, IPFS is a ***distributed system*** that uses ***content-addressed storage***. Files in IPFS are broken into smaller chunks, each of which is cryptographically hashed. These chunks are distributed across the network, and each file

can be retrieved via its unique hash. IPFS ensures that users are no longer dependent on a central entity for file retrieval and storage. This architecture reduces bottlenecks and enhances resilience against censorship or server outages.

Content Addressing is a key principle of IPFS. Files are not retrieved from a fixed location (as in URL-based systems), but from wherever their content is found on the network. This significantly enhances the **fault tolerance** of the storage system.

3. **Decentralization and Security**:

   Decentralization is a key focus for both blockchain and IPFS. Traditional centralized systems often present a single point of failure or attack. If a central server goes down or is compromised, all data stored on it is at risk. In contrast, decentralized systems distribute files or data across a peer-to-peer network, ensuring that there is no single point of failure. This decentralization provides not only resilience but also improved ***privacy***, as data cannot be easily surveilled or intercepted by a single entity.

Security is inherent in the use of **cryptographic hashing** for both blockchain and IPFS. In blockchain, hashes ensure that blocks of data are unmodifiable. In IPFS, they serve to uniquely identify content, ensuring that the correct data is retrieved without manipulation. Blockchain's immutability, coupled with IPFS's distributed storage, forms the backbone of a highly secure system for file storage.

BlockShareCloud combines the key principles of two leading technologies: blockchain and IPFS. By leveraging blockchain's immutability and IPFS's decentralized content addressing, the system offers a solution to the problems of centralization, data integrity, and security in cloud storage. While existing projects like Filecoin, Sia, and Storj focus on decentralizing the storage infrastructure, BlockShareCloud's innovative use of an in-memory blockchain adds a layer of metadata traceability, contributing to the evolving landscape of decentralized storage.

3. Methodology

3.1 Approach

The project follows a modular approach, integrating different technologies to achieve secure and decentralized file storage. The system is built around three key components:

1. **IPFS** (InterPlanetary File System) for decentralized file storage: When users upload files, they are split into chunks and distributed across the IPFS network. Each file is assigned a unique **hash**, which ensures its integrity and can be retrieved from any node within the network.
2. **Custom Blockchain for metadata storage**: The project implements a simple ***in-memory blockchain*** to track file metadata such as file names, timestamps, and IPFS hashes. The blockchain records every upload as a transaction, creating an immutable log that traces every file back to its origin, thereby preventing tampering or unauthorized modifications. Since the blockchain is in-memory, it does not persist across restarts, but it provides a proof-of-concept for how metadata can be tracked securely.
3. **Flask Web Framework** for the user interface: Flask serves as the backend for the web application, providing the routing and logic to allow users to upload and retrieve files via a simple, user-friendly interface. It interacts with IPFS and the blockchain to ensure seamless storage and traceability.

3.2 Tools and Technologies

1. **Flask**: A lightweight web framework in Python, Flask enables the construction of a web interface that handles user interactions. Flask routes requests from the frontend (HTML/CSS interface) to the backend logic, ensuring smooth communication between the user and the underlying storage mechanisms.
2. **IPFS (InterPlanetary File System)**: This decentralized peer-to-peer file storage protocol is used to store files across a distributed network. When a file is uploaded, IPFS creates a content-based unique hash for it and distributes the file chunks across the network. This ensures redundancy and prevents data loss or tampering. The hash is stored on the blockchain to track the file location.
3. **Blockchain (Custom In-Memory Implementation)**: The project uses a blockchain to store file metadata like filenames, timestamps, and IPFS hashes. This chain of metadata blocks provides an immutable history of all stored files. While in this prototype the blockchain does not persist data across restarts, it ensures traceability for the duration of system uptime. Each block contains a cryptographic hash linking it to the previous block, ensuring tamper-proof security.

4. **HTML/CSS**: For the frontend, simple HTML and CSS were employed to build a minimalistic and user-friendly interface. Users can interact with the system through this interface by uploading or retrieving files.
5. **Git**: Version control is managed using Git, facilitating collaboration, tracking changes, and ensuring the integrity of the codebase. The repository is hosted on GitHub, providing access to contributors and enabling easy tracking of project progress.
6. **Python**: Python serves as the core programming language for this project. Its versatility allows for seamless integration of Flask, IPFS, and blockchain mechanisms.

3.3 Data Collection

The system does not collect any real-time user data beyond what is necessary to perform its intended function: storing and retrieving files. The only data stored are the file names, their corresponding IPFS hashes, and timestamps, all of which are recorded in the blockchain for immutability and traceability. IPFS ensures that the actual file content is securely stored across a distributed network.

User privacy is preserved as no personal information or identifying data is stored within the system, except for the file-related metadata. This decentralized model, using IPFS and blockchain, eliminates the need for a central authority to manage or control user data. The use of IPFS hashes ensures that only those who possess the correct hash can retrieve a file, ensuring privacy and security.

This combination of a decentralized storage system (IPFS) and a custom blockchain allows for a highly secure, tamper-proof cloud storage solution, which can be further enhanced by integrating persistent blockchain mechanisms and user authentication in future iterations.

4. Implementation

4.1 Working

The project is developed using a structured approach, integrating different components to create a decentralized and secure file storage system. Below is a detailed description of each component and how they work together:

1. IPFS Integration:

The project leverages **IPFS** (InterPlanetary File System) for decentralized file storage. When a user uploads a file through the Flask web interface, the file is first stored temporarily on the local machine. Then, it is uploaded to the IPFS network, which splits the file into smaller chunks, distributing them across various nodes in the network. The IPFS network generates a ***unique hash*** for each file, representing the file's content. This ensures that the same file will always generate the same hash, guaranteeing integrity and facilitating file retrieval.

- **IPFS Hashing**: The system retrieves an IPFS hash once the file is uploaded, which uniquely identifies the file on the network.
- **Decentralized Storage**: Since the file is stored across multiple nodes, it is resilient to single points of failure. Even if one or more nodes go offline, the file can still be accessed using its hash.

2. Blockchain for Metadata:

The project uses a **custom in-memory blockchain** to store metadata associated with the uploaded files. The blockchain records the IPFS hash, file name, and a timestamp for each file in a block. Each block also contains the hash of the previous block, which maintains the integrity of the entire chain. This prevents tampering with any file metadata, as modifying any block would invalidate all subsequent blocks.

- **Blockchain Structure**: The blockchain consists of a series of blocks, where each block holds:
  - IPFS hash of the file.
  - Original file name.
  - Timestamp of when the file was uploaded.
  - Previous block's hash (to maintain the chain's integrity).

3. File Downloading:

To retrieve files, the user provides the IPFS hash. The system queries IPFS to locate and download the file. Based on the metadata stored in the blockchain, the system renames the file to its original name, ensuring the user gets the correct file.

Steps:

- The user enters the IPFS hash in the web interface.
- The system fetches the file from the IPFS network.
- The blockchain is queried to retrieve the original file name, and the file is renamed before it is returned to the user.

4. User Interface (Flask):

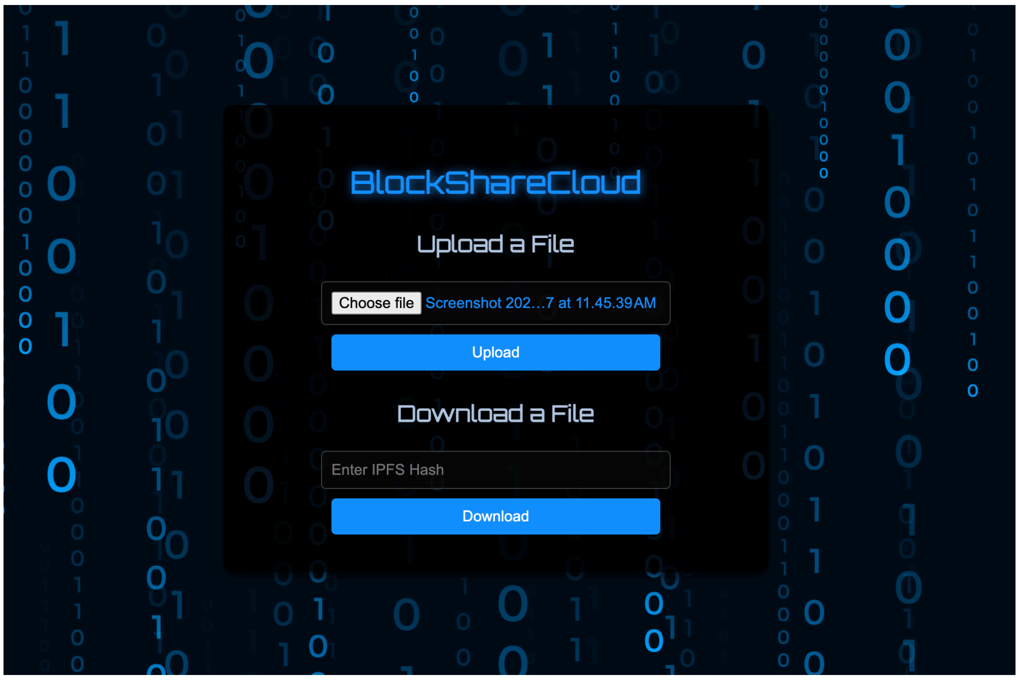
The front-end of **BlockShareCloud** is built using HTML and CSS, and Flask manages the backend. Users can easily upload files via the simple interface, and they can also input an IPFS hash to download files. Flask serves as the controller, handling user requests, processing them, and returning responses.

4.2 Challenges Faced:

1. **Blockchain Persistence**:
    - **Issue**: The blockchain is currently stored in memory, meaning all records are lost when the server restarts.
    - **Solution Considerations**: To persist the blockchain, potential solutions include saving the blockchain to a file or using an existing blockchain platform such as Ethereum. This would ensure the data persists across server restarts and can scale beyond a simple in-memory implementation.
2. **IPFS Daemon Dependency**:
    - **Issue**: The system relies on a locally running IPFS daemon. This could be a limitation for users who cannot set up or maintain the daemon on their machines.
    - **Solution Considerations**: Using a hosted IPFS service (e.g., Infura) or deploying the daemon in a cloud environment could overcome this challenge and make the system more scalable and user-friendly.

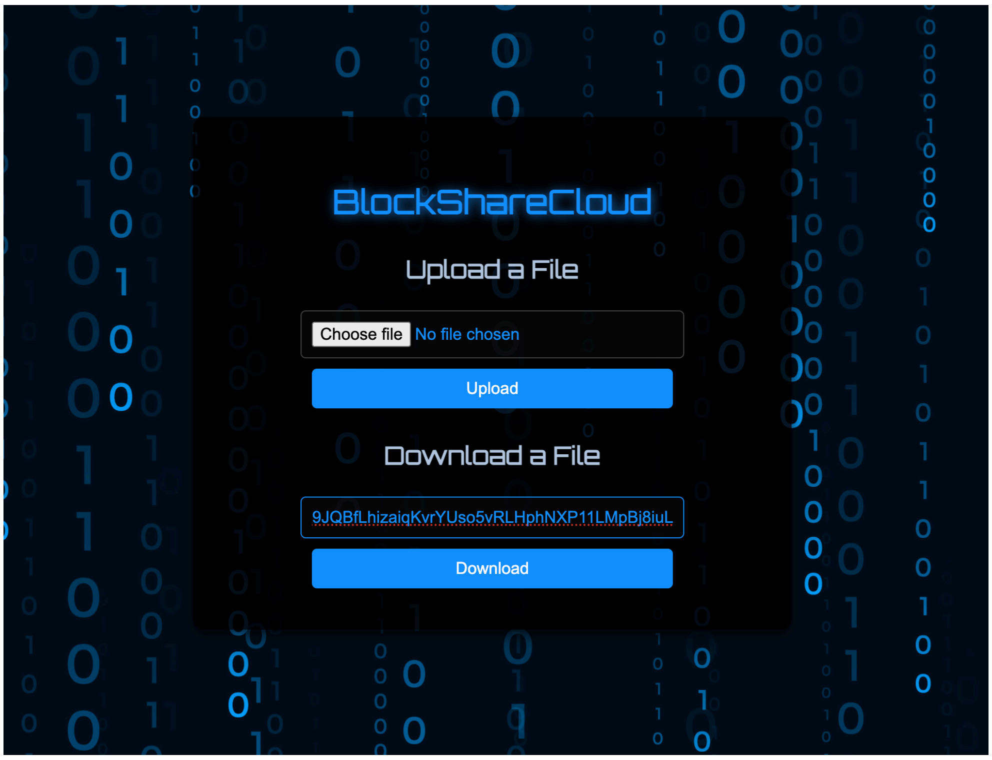
4.3 Screenshots/Diagrams:

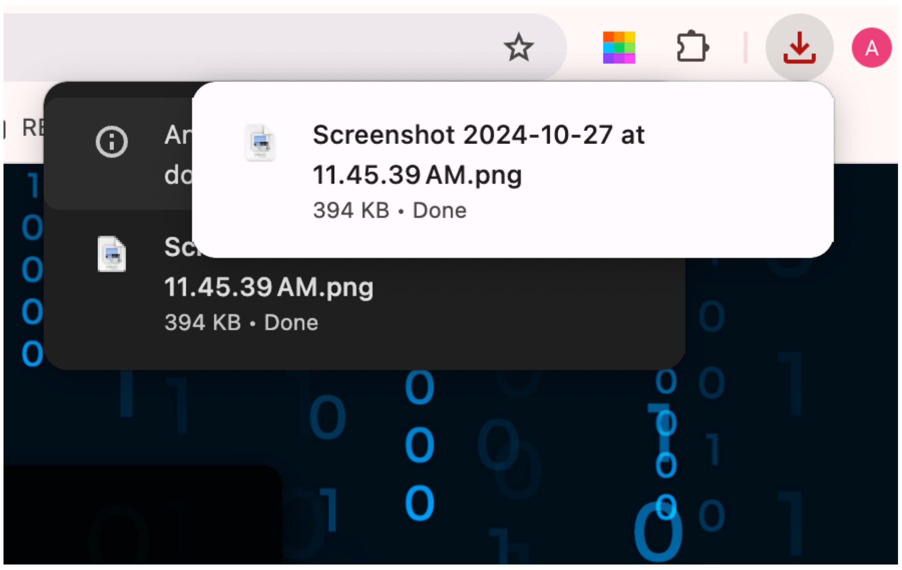
1. Web Interface for Uploading Files: The upload page provides a simple form where users can choose a file to upload. Upon submission, the file is stored in IPFS, and the metadata is recorded in the blockchain. (Screenshots can be inserted here showing the file upload interface and blockchain visualizations)



File uploaded and stored in IPFS! Hash: QmVRFk9JQBfLhizaiqKvrYUso5vRLHphNXP11LMpBj8iuL

2. File Retrieval Interface: The retrieval page allows users to input the IPFS hash and retrieve their files. The file is renamed based on metadata stored in the blockchain. (Screenshots can be inserted here showing file retrieval by IPFS hash)




Screenshot 2024-10-27 at 11.45.39 AM.png

394 KB • Done

Sc 11.45.39 AM.png

394 KB • Done

4.4 Overcoming Challenges:

The two major challenges in the current implementation were addressed in the following ways:

- **IPFS Daemon Dependency**: While the current implementation assumes that the IPFS daemon is running locally, future implementations could abstract this dependency by using hosted IPFS services or Docker containers to automate the setup process for end-users.
- **Blockchain Persistence**: For the prototype, the blockchain remains in memory, but future iterations could save the blockchain to disk or integrate a fully decentralized blockchain for better security and data persistence.

5. Results and Discussion

5.1 Outcomes

The BlockShareCloud system successfully provides a decentralized solution for secure file storage. Users can upload files, receive an IPFS hash, and store it in the blockchain for traceability. File downloads work as expected, with users retrieving files based on the stored metadata.

5.2 Analysis of Results

The integration of IPFS ensures decentralized file storage, preventing risks associated with centralized systems. Blockchain metadata storage adds tamper-proof traceability, aligning with the project's objectives. However, the limitation of the in-memory blockchain reduces its effectiveness for real-world persistent usage.

5.3 Comparison with Expected Results

The project achieved its core functionalities, as expected, by allowing secure file uploads, decentralized storage, and traceable metadata management. Challenges like blockchain persistence were anticipated, and improvements could focus on making the blockchain scalable and persistent for broader applications. Implementing a persistent ledger or integrating an external blockchain solution (e.g., Ethereum) would resolve the current limitations. Moreover, the reliance on a local IPFS daemon for file handling limits deployment flexibility, but this can be addressed by configuring remote IPFS nodes for better scalability.

Overall, the system demonstrated the practicality of using IPFS and blockchain in tandem for decentralized file storage, but real-world scalability, security, and persistence enhancements are recommended for future iterations.

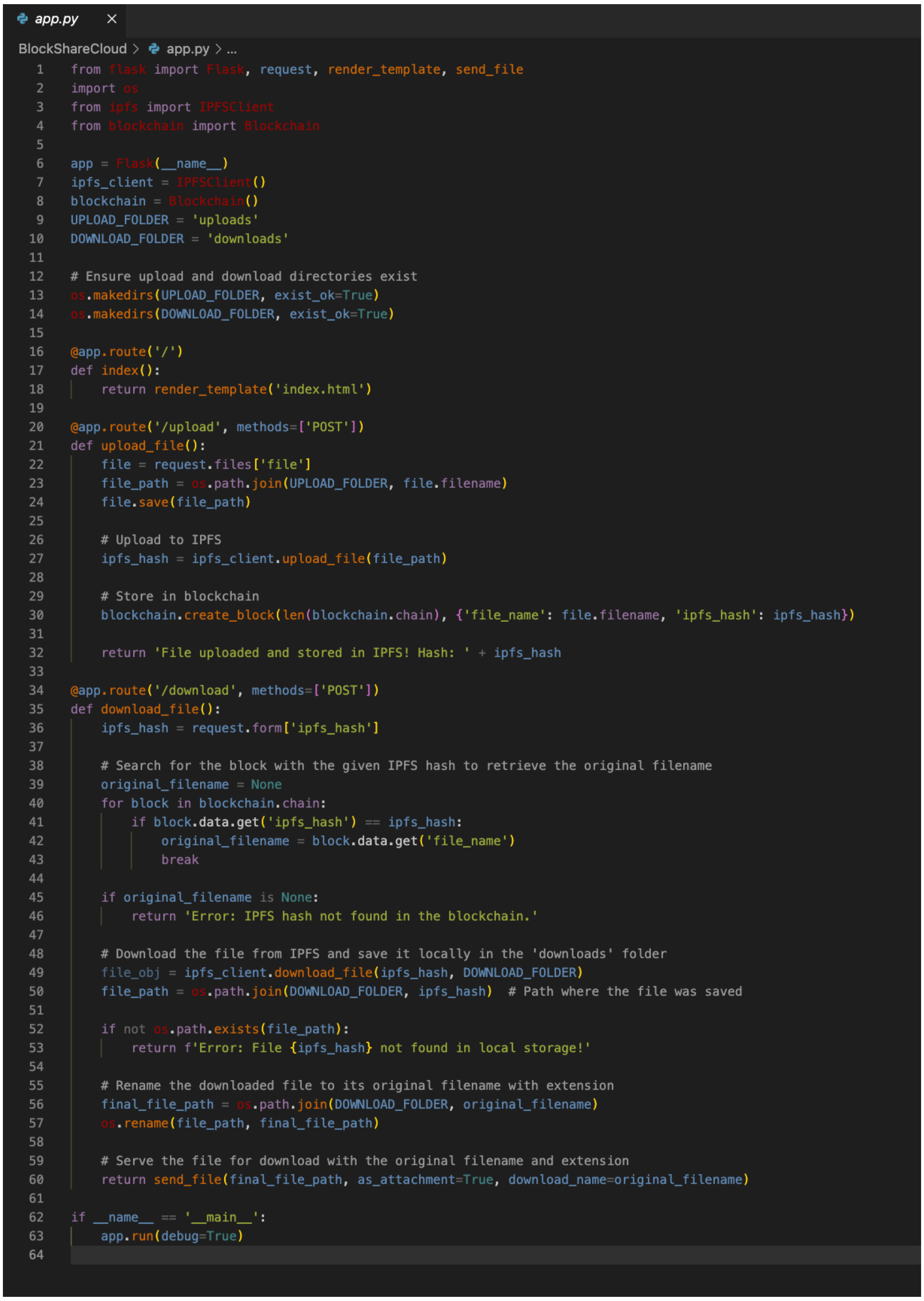
6. Conclusion

The BlockShareCloud project successfully demonstrates the feasibility of combining IPFS for decentralized file storage with blockchain technology for metadata traceability. The system achieves its goal of creating a secure, transparent, and tamper-resistant cloud storage system. Key findings include effective file storage on IPFS and the ability to trace file history via an in-memory blockchain. This work showcases the advantages of decentralization in enhancing data security and privacy.

Future developments should focus on making the blockchain persistent for real-world applications, improving scalability for handling larger datasets, and enhancing user experience by refining the interface and making the system more robust for broader use. This would position BlockShareCloud as a viable decentralized alternative to traditional cloud storage solutions, capable of handling production-scale operations with enhanced reliability. Potential future integrations with third-party blockchain solutions or distributed networks could extend the system's capability, leading to a stronger, real-world application.

7. Code Snapshots

7.1 app.py

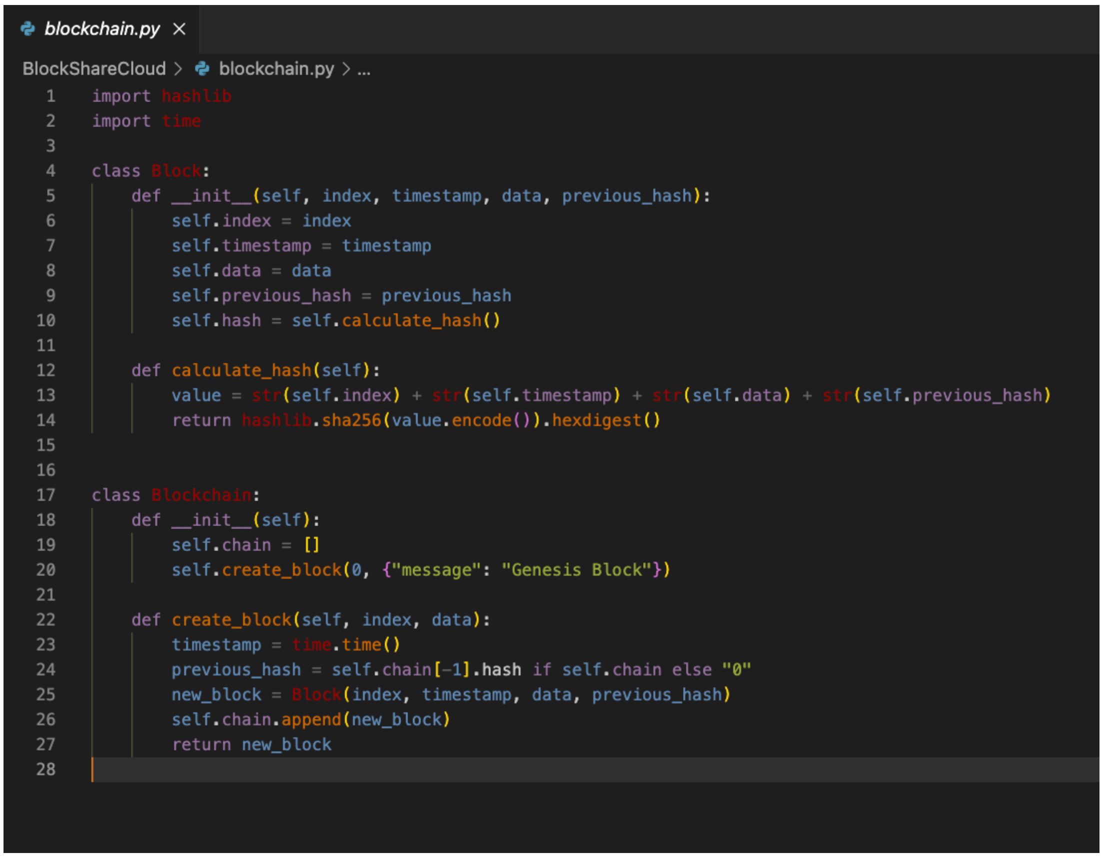


app.py

BlockShareCloud > app.py > ...

```python
from flask import Flask, request, render_template, send_file
import os
from ipfs import IPFSClient
from blockchain import Blockchain

app = Flask(__name__)
ipfs_client = IPFSClient()
blockchain = Blockchain()
UPLOAD_FOLDER = 'uploads'
DOWNLOAD_FOLDER = 'downloads'

# Ensure upload and download directories exist
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(DOWNLOAD_FOLDER, exist_ok=True)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    file = request.files['file']
    file_path = os.path.join(UPLOAD_FOLDER, file.filename)
    file.save(file_path)

    # Upload to IPFS
    ipfs_hash = ipfs_client.upload_file(file_path)

    # Store in blockchain
    blockchain.create_block(len(blockchain.chain), {'file_name': file.filename, 'ipfs_hash': ipfs_hash})

    return 'File uploaded and stored in IPFS! Hash: ' + ipfs_hash

@app.route('/download', methods=['POST'])
def download_file():
    ipfs_hash = request.form['ipfs_hash']

    # Search for the block with the given IPFS hash to retrieve the original filename
    original_filename = None
    for block in blockchain.chain:
        if block.data.get('ipfs_hash') == ipfs_hash:
            original_filename = block.data.get('file_name')
            break

    if original_filename is None:
        return 'Error: IPFS hash not found in the blockchain.'

    # Download the file from IPFS and save it locally in the 'downloads' folder
    file_obj = ipfs_client.download_file(ipfs_hash, DOWNLOAD_FOLDER)
    file_path = os.path.join(DOWNLOAD_FOLDER, ipfs_hash)  # Path where the file was saved

    if not os.path.exists(file_path):
        return f'Error: File {ipfs_hash} not found in local storage!'

    # Rename the downloaded file to its original filename with extension
    final_file_path = os.path.join(DOWNLOAD_FOLDER, original_filename)
    os.rename(file_path, final_file_path)

    # Serve the file for download with the original filename and extension
    return send_file(final_file_path, as_attachment=True, download_name=original_filename)

if __name__ == '__main__':
    app.run(debug=True)
```

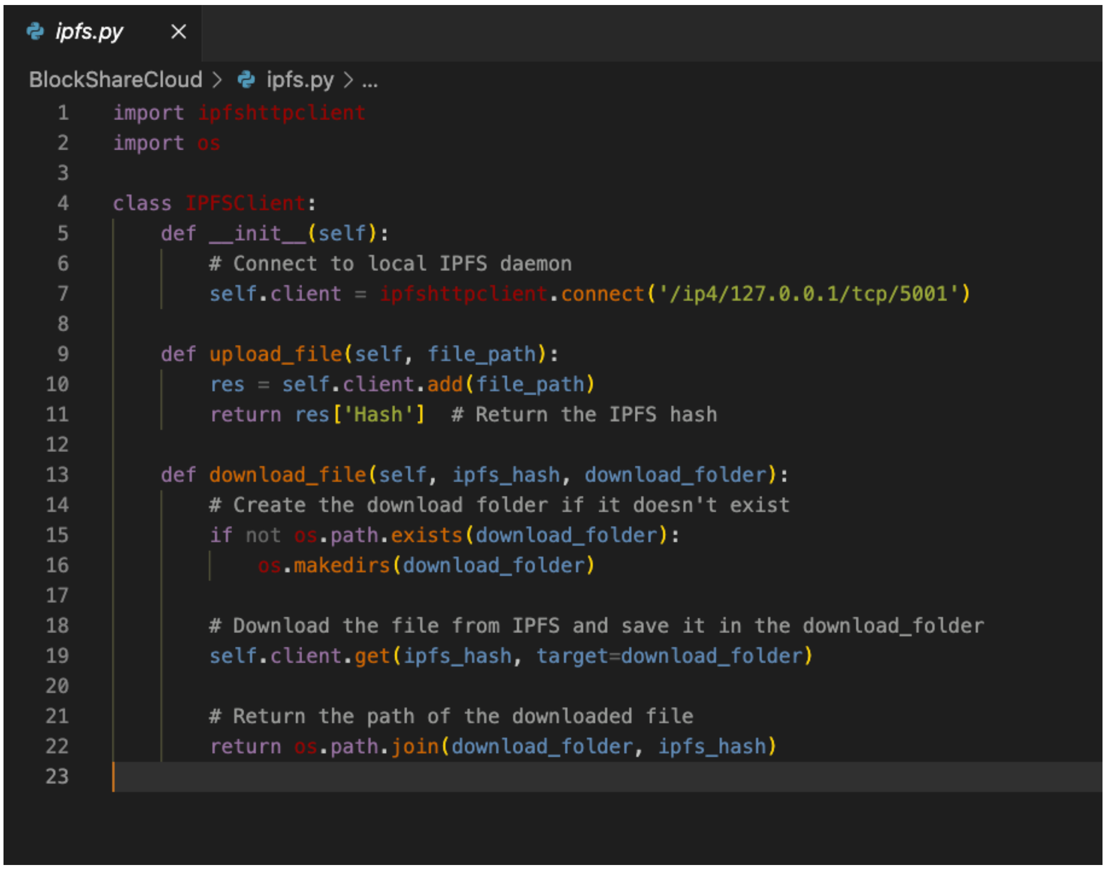
7.2 blockchain.py


blockchain.py ×

BlockShareCloud > blockchain.py > ...

```python
import hashlib
import time

class Block:
    def __init__(self, index, timestamp, data, previous_hash):
        self.index = index
        self.timestamp = timestamp
        self.data = data
        self.previous_hash = previous_hash
        self.hash = self.calculate_hash()

    def calculate_hash(self):
        value = str(self.index) + str(self.timestamp) + str(self.data) + str(self.previous_hash)
        return hashlib.sha256(value.encode()).hexdigest()


class Blockchain:
    def __init__(self):
        self.chain = []
        self.create_block(0, {"message": "Genesis Block"})

    def create_block(self, index, data):
        timestamp = time.time()
        previous_hash = self.chain[-1].hash if self.chain else "0"
        new_block = Block(index, timestamp, data, previous_hash)
        self.chain.append(new_block)
        return new_block

```

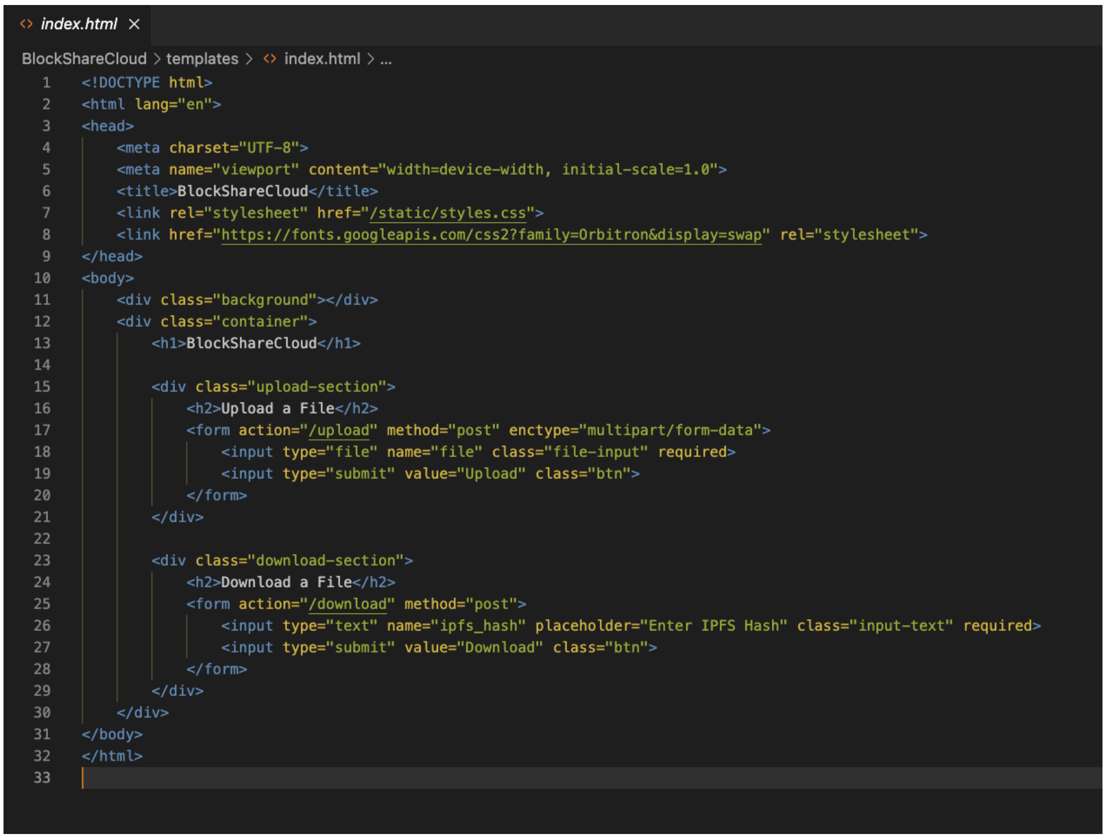
7.3 ipfs.py


ipfs.py ×

BlockShareCloud > ipfs.py > ...

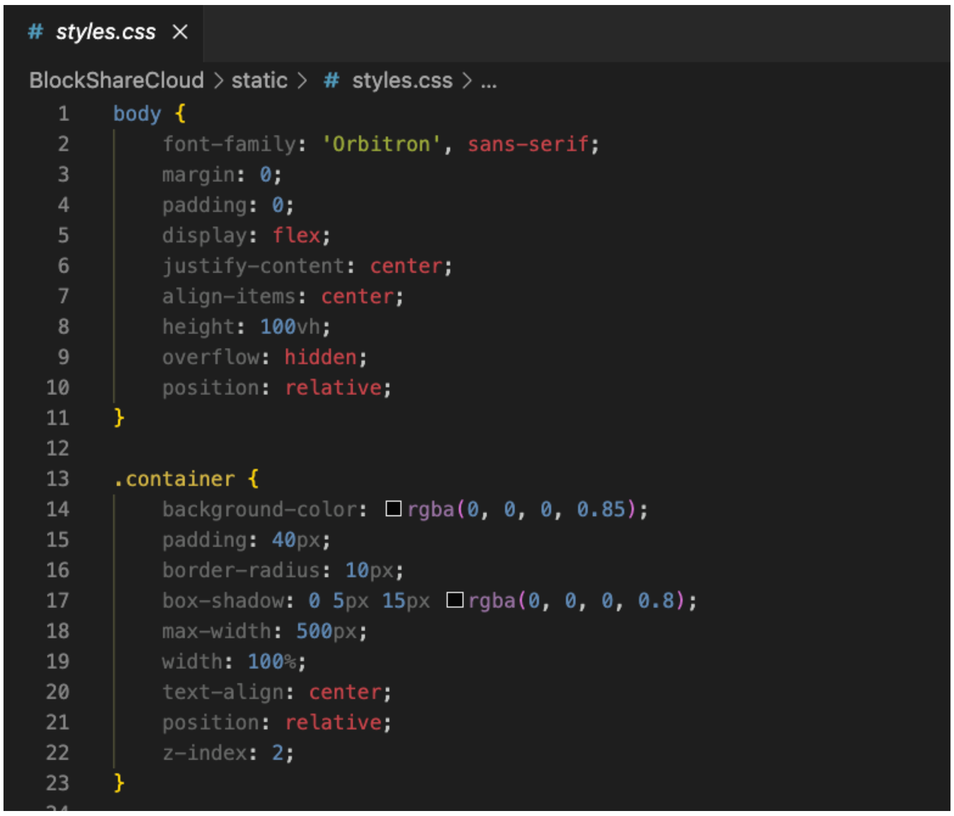
```python
import ipfshttpclient
import os

class IPFSClient:
    def __init__(self):
        # Connect to local IPFS daemon
        self.client = ipfshttpclient.connect('/ip4/127.0.0.1/tcp/5001')

    def upload_file(self, file_path):
        res = self.client.add(file_path)
        return res['Hash']  # Return the IPFS hash

    def download_file(self, ipfs_hash, download_folder):
        # Create the download folder if it doesn't exist
        if not os.path.exists(download_folder):
            os.makedirs(download_folder)

        # Download the file from IPFS and save it in the download_folder
        self.client.get(ipfs_hash, target=download_folder)

        # Return the path of the downloaded file
        return os.path.join(download_folder, ipfs_hash)

```

7.4 index.html

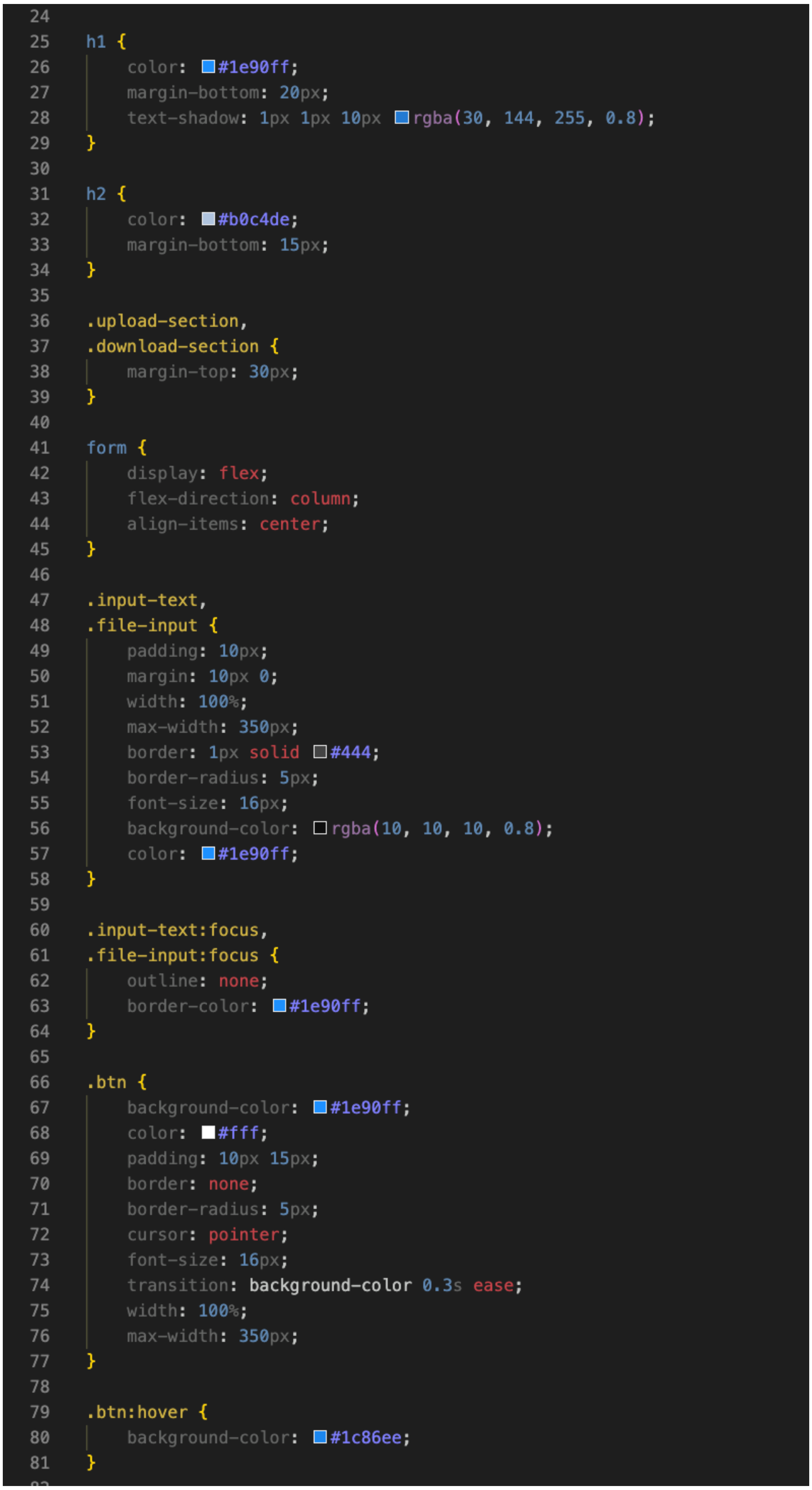


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>BlockShareCloud</title>
    <link rel="stylesheet" href="/static/styles.css">
    <link href="https://fonts.googleapis.com/css2?family=Orbitron&display=swap" rel="stylesheet">
</head>
<body>
    <div class="background"></div>
    <div class="container">
        <h1>BlockShareCloud</h1>

        <div class="upload-section">
            <h2>Upload a File</h2>
            <form action="/upload" method="post" enctype="multipart/form-data">
                <input type="file" name="file" class="file-input" required>
                <input type="submit" value="Upload" class="btn">
            </form>
        </div>

        <div class="download-section">
            <h2>Download a File</h2>
            <form action="/download" method="post">
                <input type="text" name="ipfs_hash" placeholder="Enter IPFS Hash" class="input-text" required>
                <input type="submit" value="Download" class="btn">
            </form>
        </div>
    </div>
</body>
</html>

```

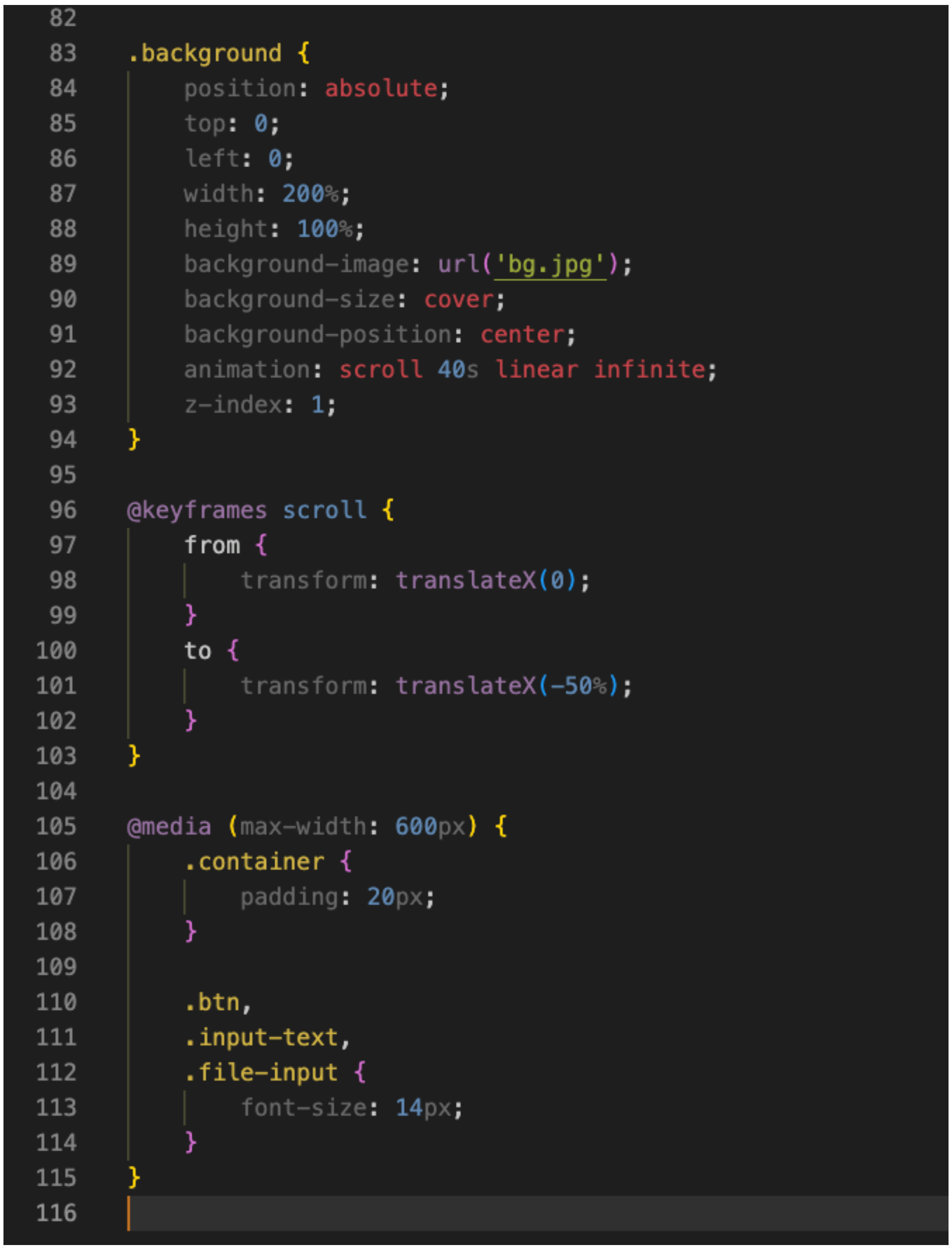
7.5 styles.css



```css
body {
    font-family: 'Orbitron', sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    overflow: hidden;
    position: relative;
}

.container {
    background-color: rgba(0, 0, 0, 0.85);
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0 5px 15px rgba(0, 0, 0, 0.8);
    max-width: 500px;
    width: 100%;
    text-align: center;
    position: relative;
    z-index: 2;
}
```

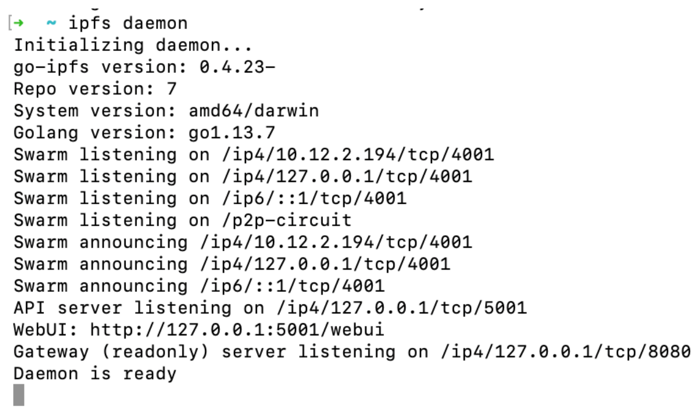


```css

h1 {
    color: #1e90ff;
    margin-bottom: 20px;
    text-shadow: 1px 1px 10px rgba(30, 144, 255, 0.8);
}

h2 {
    color: #b0c4de;
    margin-bottom: 15px;
}

.upload-section,
.download-section {
    margin-top: 30px;
}

form {
    display: flex;
    flex-direction: column;
    align-items: center;
}

.input-text,
.file-input {
    padding: 10px;
    margin: 10px 0;
    width: 100%;
    max-width: 350px;
    border: 1px solid #444;
    border-radius: 5px;
    font-size: 16px;
    background-color: rgba(10, 10, 10, 0.8);
    color: #1e90ff;
}

.input-text:focus,
.file-input:focus {
    outline: none;
    border-color: #1e90ff;
}

.btn {
    background-color: #1e90ff;
    color: #fff;
    padding: 10px 15px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s ease;
    width: 100%;
    max-width: 350px;
}

.btn:hover {
    background-color: #1c86ee;
}
```

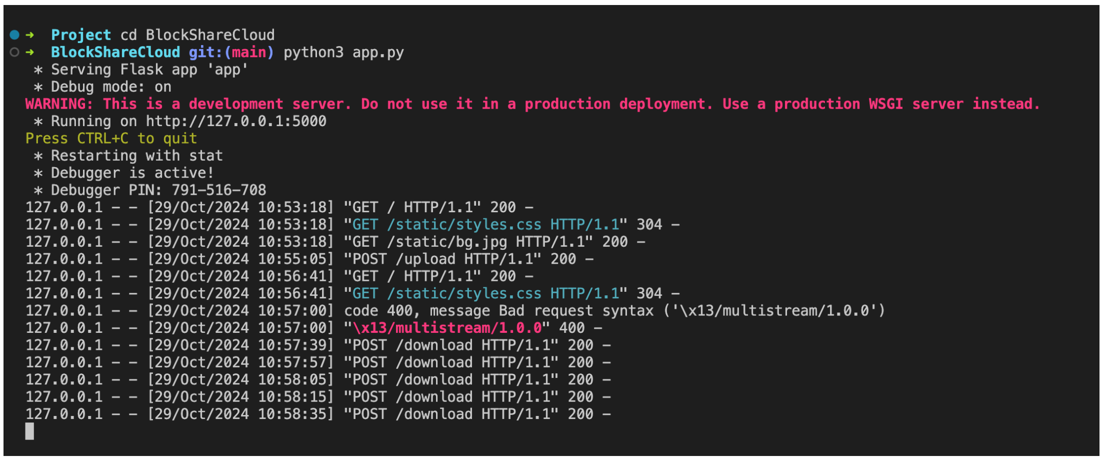


```css

.background {
    position: absolute;
    top: 0;
    left: 0;
    width: 200%;
    height: 100%;
    background-image: url('bg.jpg');
    background-size: cover;
    background-position: center;
    animation: scroll 40s linear infinite;
    z-index: 1;
}

@keyframes scroll {
    from {
        transform: translateX(0);
    }
    to {
        transform: translateX(-50%);
    }
}

@media (max-width: 600px) {
    .container {
        padding: 20px;
    }

    .btn,
    .input-text,
    .file-input {
        font-size: 14px;
    }
}

```

7.1 Server End



```
➜  ~ ipfs daemon
Initializing daemon...
go-ipfs version: 0.4.23-
Repo version: 7
System version: amd64/darwin
Golang version: go1.13.7
Swarm listening on /ip4/10.12.2.194/tcp/4001
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip6/::1/tcp/4001
Swarm listening on /p2p-circuit
Swarm announcing /ip4/10.12.2.194/tcp/4001
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip6/::1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
WebUI: http://127.0.0.1:5001/webui
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```



```
● ➜  Project cd BlockShareCloud
○ ➜  BlockShareCloud git:(main) python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 791-516-708
127.0.0.1 - - [29/Oct/2024 10:53:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2024 10:53:18] "GET /static/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [29/Oct/2024 10:53:18] "GET /static/bg.jpg HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2024 10:55:05] "POST /upload HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2024 10:56:41] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2024 10:56:41] "GET /static/styles.css HTTP/1.1" 304 -
127.0.0.1 - - [29/Oct/2024 10:57:00] code 400, message Bad request syntax ('\x13/multistream/1.0.0')
127.0.0.1 - - [29/Oct/2024 10:57:00] "\x13/multistream/1.0.0" 400 -
127.0.0.1 - - [29/Oct/2024 10:57:39] "POST /download HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2024 10:57:57] "POST /download HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2024 10:58:05] "POST /download HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2024 10:58:15] "POST /download HTTP/1.1" 200 -
127.0.0.1 - - [29/Oct/2024 10:58:35] "POST /download HTTP/1.1" 200 -
```

References

1. Protocol Labs. (2015). IPFS - Content Addressed, Versioned, P2P File System. Retrieved from https://ipfs.io/
2. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from https://bitcoin.org/bitcoin.pdf
3. Antonopoulos, A. M. (2017). Mastering Bitcoin: Unlocking Digital Cryptocurrencies. O'Reilly Media
4. Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P Hypermedia Protocol. Retrieved from https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6NxTq5K9mA8vxoKr1g6nsuGzUwwF
5. Protocol Labs. (2017). Filecoin: A Decentralized Storage Network. Retrieved from https://filecoin.io/
6. David Vorick & Luke Champine (2018). Sia: Simple Decentralized Storage. Retrieved from https://sia.tech/sia.pdf
7. Wood, G. (2014). Ethereum: A Secure Decentralised Generalised Transaction Ledger. Retrieved from https://ethereum.org/en/whitepaper/
8. Voshmgir, S. (2020). Token Economy: How Blockchains and Smart Contracts Revolutionize the Economy. BlockchainHub
9. Flask Documentation. (2024). Flask Web Framework. Retrieved from https://flask.palletsprojects.com/
10. GitHub Documentation. (2024). Version Control System (Git). Retrieved from https://git-scm.com/
11. Marr, B. (2019). Blockchain Explained in Simple Terms. Forbes. Retrieved from https://www.forbes.com/

GitHub Link

(with Usage Guide & Report Copy)

https://github.com/anushreejha/BlockShareCloud