Name: **Piyusha Rajendra Supe**

Enrolment Number: **23CO315** (TE-B)

Practical 05

Aim: Setup your own cloud for Software as a Service (SaaS) over the existing LAN in your laboratory. In this assignment you have to write your own code for cloud controller using open-source technologies to implement with HDFS. Implement the basic operations may be like to divide the file in segments/blocks and upload/ download file on/from cloud in encrypted form

How to Run the Web app?

1. **Install dependencies**

Make sure you have Python and pip installed.

pip install flask cryptography

2. **Run the Flask App**

python app.py

It will start a local server on:

http://127.0.0.1:5000/

Using the Web App

- **Upload** any file.
- It gets encrypted, split, and stored in 3 cloud/nodeX folders.
- Click **Download** to reconstruct and download the original file.

Code is as follows:

1] index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Mini HDFS Cloud</title>
</head>
<body>
```

```
    <h2>Upload File to Cloud</h2>
    <form method="POST" enctype="multipart/form-data">
        <input type="file" name="file" required>
        <button type="submit">Upload</button>
    </form>
    <h2>Download Reconstructed File</h2>
    <a href="/download"><button>Download</button></a>
</body>
</html>
```

2] app.py

```
from cryptography.fernet import Fernet

def split_file(data, size=1024):
    return [data[i:i+size] for i in range(0, len(data), size)]

def merge_blocks(blocks):
    return b"".join(blocks)

def generate_key():
    key = Fernet.generate_key()
    with open("secret.key", "wb") as key_file:
        key_file.write(key)

def load_key():
    return open("secret.key", "rb").read()

def encrypt_file(file_path):
    key = load_key()
    f = Fernet(key)
    with open(file_path, "rb") as file:
```

```
        original = file.read()
    encrypted = f.encrypt(original)
    return encrypted

def decrypt_data(data):
    key = load_key()
    f = Fernet(key)
    return f.decrypt(data)

from flask import Flask, render_template, request, send_file, redirect, url_for
#from utils.crypto import generate_key, encrypt_file, decrypt_data
#from utils.block_handler import split_file, merge_blocks
import os

app = Flask(__name__)
UPLOAD_FOLDER = "uploads"
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
STORAGE_NODES = ["cloud/node1", "cloud/node2", "cloud/node3"]
for node in STORAGE_NODES:
    os.makedirs(node, exist_ok=True)

generate_key()

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        file = request.files["file"]
        file_path = os.path.join(UPLOAD_FOLDER, file.filename)
        file.save(file_path)
        upload_file(file_path)
        return redirect(url_for("index"))
```

```
    return render_template("index.html")


@app.route("/download")
def download():
    download_file("downloaded_sample.txt")
    return send_file("downloaded_sample.txt", as_attachment=True)


def upload_file(file_path):
    encrypted_data = encrypt_file(file_path)
    blocks = split_file(encrypted_data)
    for i, block in enumerate(blocks):
        node_index = i % len(STORAGE_NODES)
        with open(f"{STORAGE_NODES[node_index]}/block_{i}.bin", "wb") as f:
            f.write(block)


def download_file(output_path):
    blocks = []
    for node in STORAGE_NODES:
        for filename in sorted(os.listdir(node)):
            if filename.startswith("block_"):
                with open(os.path.join(node, filename), "rb") as f:
                    blocks.append(f.read())
    decrypted = decrypt_data(merge_blocks(blocks))
    with open(output_path, "wb") as f:
        f.write(decrypted)


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

3] main.py

```
from cryptography.fernet import Fernet
def split_file(data, size=1024):
```

```
    return [data[i:i+size] for i in range(0, len(data), size)]


def merge_blocks(blocks):
    return b"".join(blocks)


def generate_key():
    key = Fernet.generate_key()
    with open("secret.key", "wb") as key_file:
        key_file.write(key)


def load_key():
    return open("secret.key", "rb").read()


def encrypt_file(file_path):
    key = load_key()
    f = Fernet(key)
    with open(file_path, "rb") as file:
        original = file.read()
    encrypted = f.encrypt(original)
    return encrypted


def decrypt_data(data):
    key = load_key()
    f = Fernet(key)
    return f.decrypt(data)


#from utils.crypto import generate_key, encrypt_file, decrypt_data
#sfrom utils.block_handler import split_file, merge_blocks
import os


STORAGE_NODES = ["cloud/node1", "cloud/node2", "cloud/node3"]
```

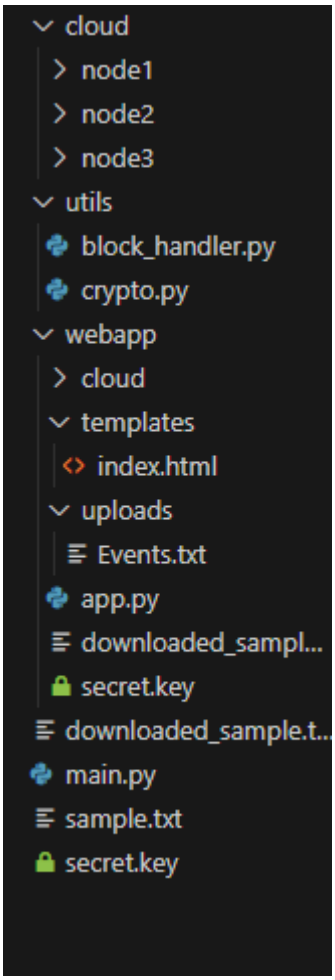
```
# Ensure all node directories exist
for node in STORAGE_NODES:
    os.makedirs(node, exist_ok=True)

def upload_file(file_path):
    encrypted_data = encrypt_file(file_path)
    blocks = [encrypted_data[i:i + 1024] for i in range(0, len(encrypted_data), 1024)]

    for i, block in enumerate(blocks):
        node_index = i % len(STORAGE_NODES)
        with open(f"{STORAGE_NODES[node_index]}/block_{i}.bin", "wb") as f:
            f.write(block)
    print("Upload successful!")

def download_file(output_path):
    blocks = []
    for node in STORAGE_NODES:
        for filename in sorted(os.listdir(node)):
            if filename.startswith("block_"):
                with open(os.path.join(node, filename), "rb") as f:
                    blocks.append(f.read())
    decrypted = decrypt_data(b"".join(blocks))
    with open(output_path, "wb") as f:
        f.write(decrypted)
    print("Download successful!")

if __name__ == "__main__":
    generate_key()
    upload_file("sample.txt")
    download_file("downloaded_sample.txt")
```

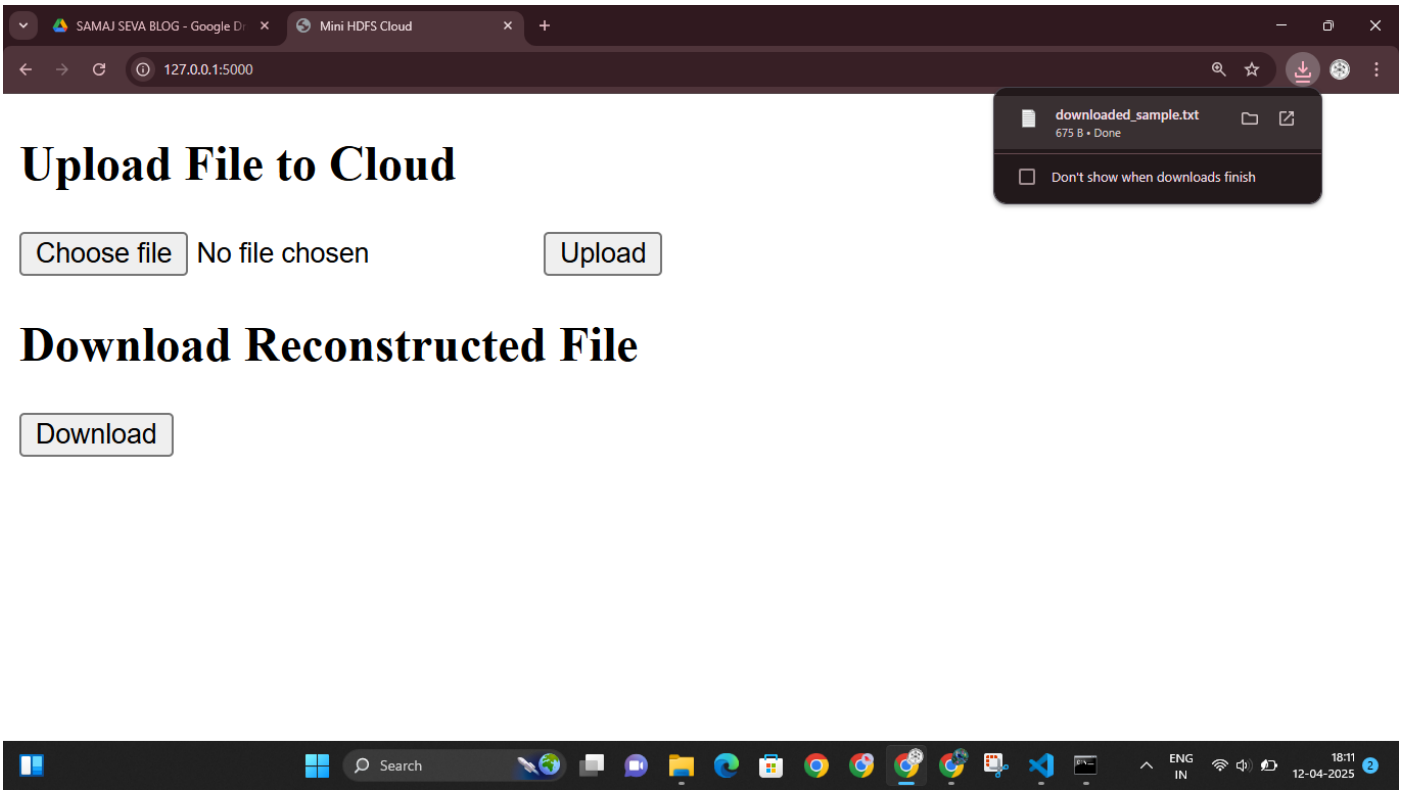


Upload File to Cloud

Choose file Events.txt Upload

Download Reconstructed File

Download


SAMAJ SEVA BLOG - Google Dr
Mini HDFS Cloud
127.0.0.1:5000
downloaded_sample.txt
675 B • Done
Don't show when downloads finish
Upload File to Cloud
Choose file
No file chosen
Upload
Download Reconstructed File
Download
Search
ENG
IN
18:11
12-04-2025