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

DISTRIBUTED SYSTEMS (CSE -3261) MINI PROJECT REPORT ON

Title of the Mini Project

SUBMITTED TO

Department of Computer Science & Engineering

By

Dheeraj
210905402
62
VI-C

Pratham U
210905407
53
VI-C

Name & Signature of Evaluator 1

Name & Signature of Evaluator 2

(Jan 2024 – May 2024)

Table of Contents

1. INTRODUCTION

1.1 General

In today's interconnected digital landscape, the ability to share files efficiently and securely is paramount for individuals and teams alike. File sharing serves as a linchpin for collaboration, enabling seamless exchange and joint editing of documents, media files, and datasets across diverse platforms and devices. Recognizing the pivotal role of file sharing in modern workflows, this Python project embarks on creating a robust framework to facilitate this essential process.

At the heart of this endeavor lies the utilization of network communication protocols to establish a seamless flow of data between users, leveraging Python's versatility in network programming. The project harnesses the power of server-client architecture to create a cohesive environment where users can share, access, and collaborate on files with ease. Through intuitive interfaces and streamlined functionalities, it empowers users to effortlessly upload, download, and manage files within a collaborative ecosystem, fostering a dynamic environment where ideas can be exchanged, projects can evolve, and productivity can thrive. Central to the project's ethos is the implementation of robust security measures to safeguard sensitive data during transmission, employing encryption techniques offered by Python's cryptography libraries to ensure files remain protected from unauthorized access or interception. By laying the groundwork for a comprehensive file sharing solution, this Python project not only addresses immediate collaboration needs but also paves the way for future innovations in collaborative workflows, serving as a catalyst for enhanced productivity and collaboration in diverse contexts.

1.2 Objectives

This Python project aims to lay the groundwork for an efficient and secure file sharing system, with the following key objectives:

Seamless File Sharing:

Enable users to share files effortlessly across a network, facilitating smooth collaboration and information exchange.

Secure Data Transmission:

Implement robust encryption techniques to safeguard sensitive files and data during transmission, ensuring privacy and confidentiality.

User-Friendly Interface:

Develop an intuitive user interface that offers ease of navigation and efficient file management tools, enhancing the overall user experience.

Scalability and Reliability:

Design the system to be scalable and reliable, capable of handling varying loads and maintaining consistent performance under different usage scenarios.

2 Literature Review

2.1 Collaborative Text Editing

Collaborative text editing, also known as collaborative editing or collaborative writing, is a field of research and practice that focuses on enabling multiple users to work together on the same document simultaneously. Initially popularized by platforms like Google Docs, collaborative

text editing has become increasingly prevalent in various domains, including academia, business, and creative writing.

Early research in collaborative editing primarily focused on addressing technical challenges related to concurrency control, conflict resolution, and real-time synchronization. Traditional operational transformation (OT) algorithms, such as those proposed by Sun et al. and Ellis et al., laid the foundation for resolving conflicts and maintaining consistency in shared documents by transforming user operations in a manner that preserves their intent and order.

As the field evolved, researchers explored alternative approaches to collaborative editing, including distributed version control systems (DVCS) like Git and Mercurial, which offer decentralized collaboration models and sophisticated branching and merging capabilities. Additionally, operational transformation techniques were refined and extended to support more complex data types and application-specific semantics, leading to the development of operational transformation frameworks like ShareJS and Yjs.

2.2 Security and Encryption

Security is a critical consideration in collaborative text editing environments, especially when sensitive or confidential information is involved. Collaborative environments are susceptible to various security threats, including eavesdropping, data interception, and unauthorized access.

To mitigate these threats, robust security measures must be implemented. Encryption techniques, such as Fernet encryption, play a crucial role in securing data during transmission and storage. Fernet encryption ensures end-to-end encryption, meaning that data is encrypted on the client-side before being transmitted and decrypted on the server-side upon receipt, thus safeguarding against unauthorized access.

Authentication mechanisms are also essential in ensuring the integrity and security of collaborative environments. User authentication verifies the identities of users accessing the system, preventing unauthorized access and protecting against potential security breaches. Techniques such as password authentication, multi-factor authentication, and biometric authentication can be employed to enhance security in collaborative text editing environments.

By incorporating these security measures, collaborative text editing systems can provide users with a secure and trusted environment for collaborating on documents, ensuring the confidentiality, integrity, and availability of sensitive information.

3 Methodology

3.1 Steps Taken

1.Project Planning and Requirements Gathering:

- Define the scope and objectives of the project.
- Identify the key features and functionalities required, such as real-time collaboration, document synchronization, and encryption.
- Gather user requirements through surveys, interviews, or user stories to understand the needs and preferences of potential users.

2.Design Phase:

- Design the system architecture, including the client-server communication protocol, data storage mechanism, and concurrency control strategy.
- Define the user interface design, considering factors such as usability, accessibility, and responsiveness.
- Choose appropriate technologies and frameworks for development, such as Python for server-side scripting, socket programming for network communication, and cryptography library for encryption.

3.Implementation:

- Set up the development environment and version control system (e.g., Git) to facilitate collaborative development.
- Develop the server-side code (Server.py) to handle client connections, authentication, message processing, and document synchronization.
- Implement the client-side code (Client.py) to establish connections with the server, handle user inputs, and display document updates.
- Integrate encryption functionality using the Fernet encryption scheme to secure communication between clients and the server.
- Test each component thoroughly, including unit testing, integration testing, and system testing, to ensure functionality and reliability.

4.Documentation:

- Provide clear explanations of the codebase, including comments, documentation strings, and inline documentation, to facilitate understanding and maintenance.

3.2 Features

Features of File Sharing System

1)File Upload and Download:

Users can easily upload files to the system, either individually or in bulk, and download files they have access to. File uploading may support drag-and-drop functionality, progress indicators, and file type validation.

2)Version Control:

Version control features enable users to track and manage different versions of a file, facilitating collaboration and preventing data loss. Users can view revision history, revert to previous versions, and compare changes between versions.

3)File Sharing and Collaboration:

Users can share files with other users or groups, enabling collaborative editing and real-time collaboration on shared documents. Features like commenting, annotations, and live editing enhance productivity and teamwork.

4)Encryption and Security:

Robust encryption techniques are employed to secure files and data both at rest and in transit. This includes encrypting files before uploading them to the server, using secure communication protocols like SSL/TLS, and implementing encryption keys and access controls to protect sensitive information.

5)Notifications and Alerts:

Users receive notifications and alerts for important events, such as file sharing invitations, comments on shared files, or changes to shared documents. This keeps users informed and engaged, fostering better communication and collaboration.

6)Cross-Platform Compatibility:

The file sharing system is accessible from various devices and platforms, including desktop computers, laptops, tablets, and smartphones. Native applications, web interfaces, and mobile apps ensure seamless access and usability across different devices.

7)Scalability and Performance:

The system is designed to scale horizontally to accommodate growing user bases and increasing file volumes. Load balancing, caching mechanisms, and optimization techniques ensure optimal performance and responsiveness, even under heavy usage.

4 Results and Discussion

4.1 Code:

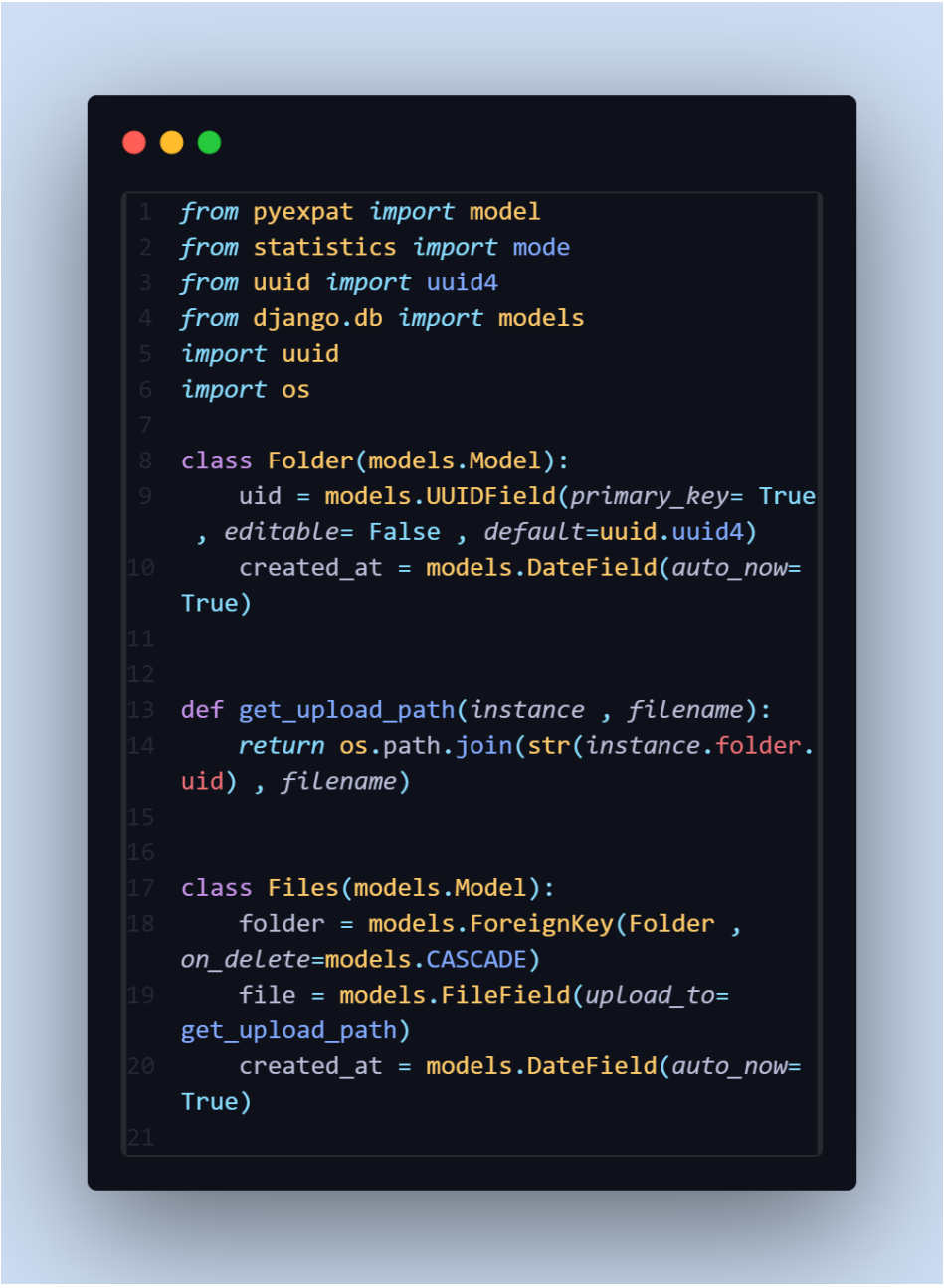
Serializers.py

```python
import shutil
from rest_framework import serializers
from .models import *


class FileSerializer(serializers.ModelSerializer):

    class Meta:
        model = Files
        fields = '__all__'

class FileListSerializer(serializers.Serializer):
    files = serializers.ListField(
        child = serializers.FileField(max_length = 100000 , allow_empty_file = False , use_url = False)
    )
    folder = serializers.CharField(required = False)

    def zip_files(self,folder):
        shutil.make_archive(f'public/static/zip/{folder}' , 'zip' ,f'public/static/{folder}' )

    def create(self , validated_data):
        folder = Folder.objects.create()
        files = validated_data.pop('files')
        files_objs = []
        for file in files:
            files_obj = Files.objects.create(folder = folder , file = file)
            files_objs.append(files_obj)


        self.zip_files(folder.uid)


        return {'files' : {} , 'folder' : str(folder.uid)}
```

Views.py

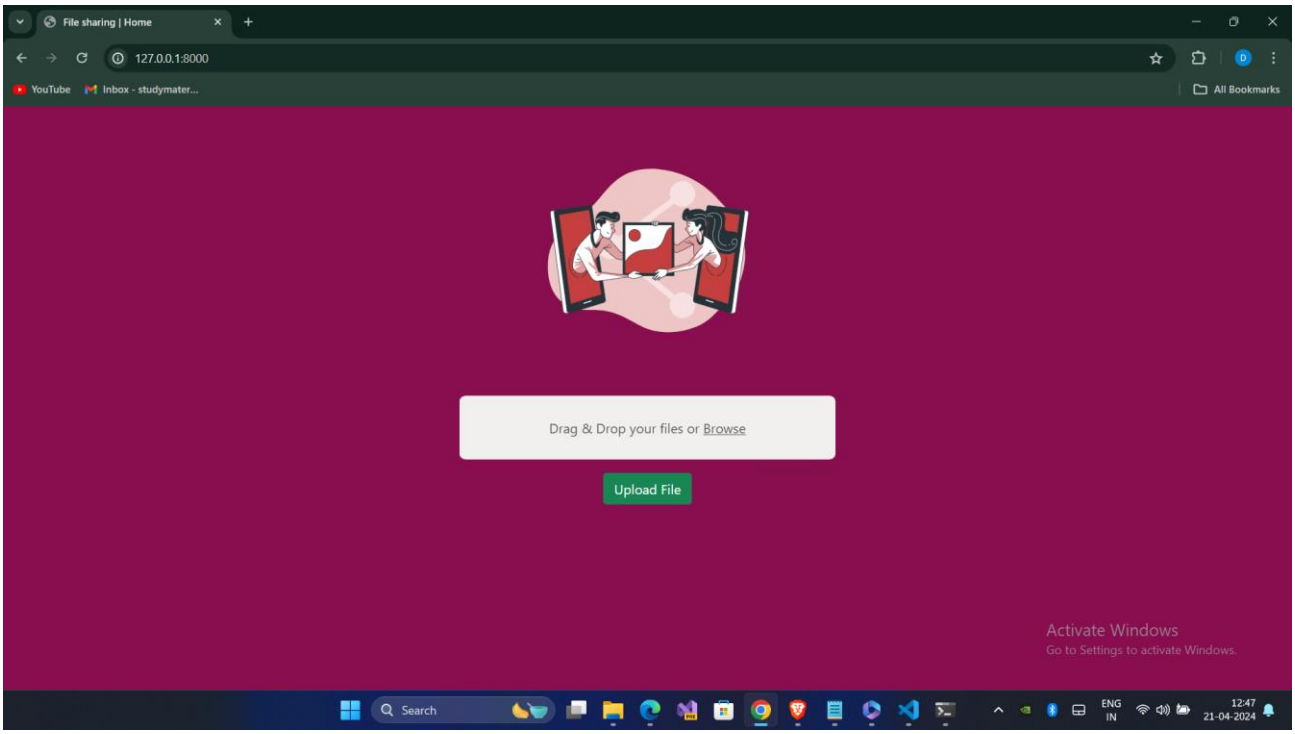
```python
from django.shortcuts import render
from rest_framework.views import APIView
from rest_framework.response import Response

from .serializers import *
from rest_framework.parsers import
 MultiPartParser


def home(request):
    return render(request ,'home.html')



def download(request , uid):
    return render(request , 'download.html'
, context = {'uid' : uid})

class HandleFileUpload(APIView):
    parser_classes = [MultiPartParser]

    def post(self , request):
        try:
            data = request.data

            serializer = FileListSerializer(
data = data)

            if serializer.is_valid():
                serializer.save()
                return Response({
                    'status' : 200,
                    'message' : '
files uploaded successfully',
                    'data' : serializer.data
                })

            return Response({
                'status' : 400,
                'message' : '
something went wrong',
                'data'   : serializer.errors
            })
        except Exception as e:
            print(e)

```

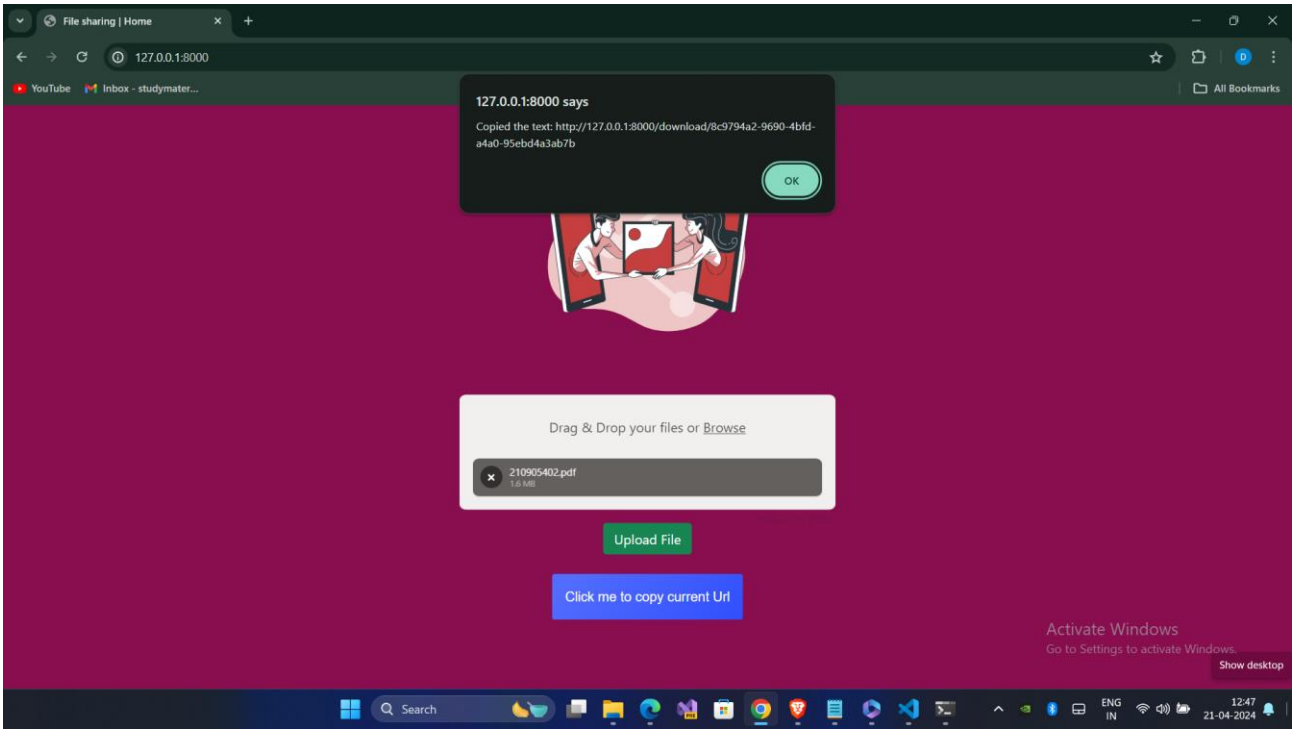
Models.py

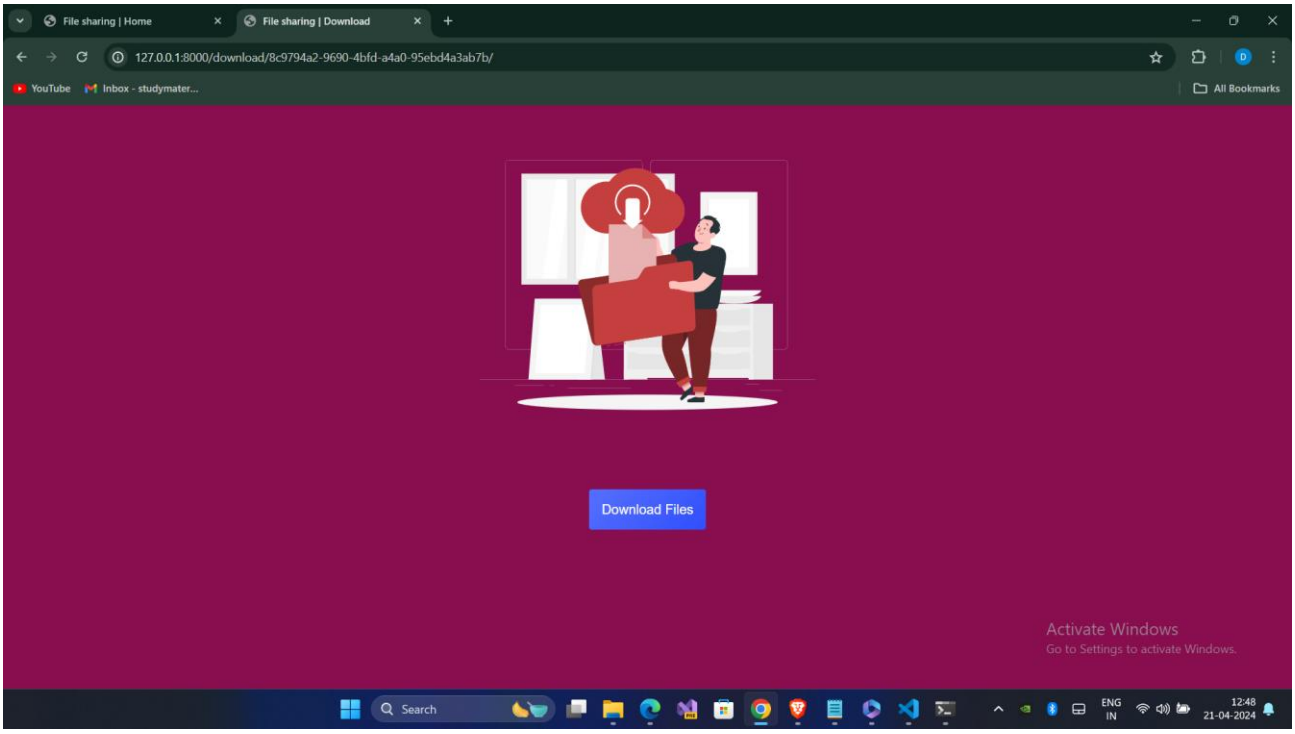


```python
from pyexpat import model
from statistics import mode
from uuid import uuid4
from django.db import models
import uuid
import os

class Folder(models.Model):
    uid = models.UUIDField(primary_key= True , editable= False , default=uuid.uuid4)
    created_at = models.DateField(auto_now= True)


def get_upload_path(instance , filename):
    return os.path.join(str(instance.folder.uid) , filename)


class Files(models.Model):
    folder = models.ForeignKey(Folder , on_delete=models.CASCADE)
    file = models.FileField(upload_to= get_upload_path)
    created_at = models.DateField(auto_now= True)

```

4.2 **Output's:**


File sharing | Home
127.0.0.1:8000
YouTube
Inbox - studymater...
All Bookmarks
Drag & Drop your files or Browse
Upload File
Activate Windows
Go to Settings to activate Windows.
Search
ENG
IN
12:47
21-04-2024


File sharing | Home
127.0.0.1:8000
YouTube
Inbox - studymater...
All Bookmarks
127.0.0.1:8000 says
Copied the text: http://127.0.0.1:8000/download/8c9794a2-9690-4bfd-a4a0-95ebd4a3ab7b
OK
Drag & Drop your files or Browse
210905402.pdf
1.6 MB
Upload File
Click me to copy current Url
Activate Windows
Go to Settings to activate Windows.
Show desktop
Search
ENG
IN
12:47
21-04-2024



4.3 Working

The file sharing system operates through a client-server architecture, where users interact with the system through client applications or web interfaces. Upon authentication, users can upload files to the server, which stores them securely in a centralized repository. Access control mechanisms ensure that only authorized users can view, edit, or delete files based on predefined permissions. When a user wishes to share a file, they specify the recipients and permissions, and the system generates unique access links or invites users to collaborate directly. Files are transmitted securely over the network using encryption techniques such as SSL/TLS, ensuring confidentiality and integrity during transmission. Version control mechanisms track changes to files, enabling users to revert to previous versions if needed. Additionally, the system provides features such as search, filtering, and notifications to facilitate efficient file management and collaboration. Continuous monitoring and maintenance ensure the system's availability, performance, and security, while user training and support contribute to a positive user experience.

5 Conclusion and Future Enhancements

5.1 Conclusion

In conclusion, the development of a file sharing system represents a significant step towards enhancing collaboration, productivity, and information exchange within organizations and communities. By leveraging advanced technologies, robust security measures, and user-friendly features, such a system offers a comprehensive solution for sharing, managing, and collaborating on files and documents in a secure and efficient manner.

Throughout this project, we have explored the key components, methodologies, and features involved in building a file sharing system. From requirement analysis and system design to implementation, testing, and deployment, each stage has been carefully executed to ensure the system's functionality, reliability, and usability.

In essence, the development of a file sharing system represents a testament to the power of technology in enabling seamless collaboration and information exchange, empowering individuals and organizations to achieve their goals more effectively and efficiently in today's dynamic and interconnected digital landscape.

5.2 Future Scope

The file sharing system developed herein lays a solid foundation for collaboration and information exchange, yet there are several avenues for future expansion and enhancement. One potential area for growth is the integration of artificial intelligence (AI) and machine learning (ML) algorithms to automate file organization, categorization, and recommendation processes. By analyzing user behavior, content similarities, and contextual information, AI-driven algorithms can intelligently suggest relevant files, predict user preferences, and streamline the file discovery process. Additionally, advancements in blockchain technology offer intriguing possibilities for enhancing the security, transparency, and decentralization of file sharing systems. By leveraging blockchain's immutable ledger and smart contract capabilities, users can establish verifiable ownership, enforce access control policies, and ensure tamper-proof auditing of file transactions. Furthermore, as the Internet of Things (IoT) continues to proliferate, integrating IoT devices with the file sharing system opens up new opportunities for seamless file synchronization, remote access, and collaborative workflows. IoT-enabled sensors, smart appliances, and wearable devices can generate and consume data that seamlessly integrates with the file sharing ecosystem, enabling innovative use cases in sectors such as healthcare, logistics, and smart cities. Overall, the future of file sharing systems is characterized by a convergence of emerging technologies, driven by a shared goal of enhancing collaboration, efficiency, and security in the digital age. Continued research and development efforts in these areas promise to unlock new levels of innovation and value for users and organizations alike.

REFERENCES

[1]. J. S. Turner, “New directions in communications,” IEEE Journal on Selected Areas in Communications, vol. 13, no. 1, pp. 11-23, Jan. 1995.

[2]. D. B. Payne and J. R. Stern, “Wavelength-switched passively coupled single-mode optical network,” Proceedings of the 2nd International Conference on Optical Fiber Sensors, Boston, MA, USA, 1985, pp. 585–590.

[3]. Tannenbaum A.S. – “Computer Networks”, Edn. 3 , Prentice Hall of India (EE edition), New Delhi, 1998.

[4]. Frame Based Feature Vectors, https://www.quora.com/