A SYNOPSIS ON

REMOTE FILE MANAGER

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted by:

Aayush Mehra	**2261055**
Aman Negi	**2261086**
Himanshu Joshi	**2261268**
Tanuj Joshi	**2261571**

Under the Guidance of

Mr. Prince Kumar

Assistant Professor

Project Team ID: 34

Department of Computer Science & Engineering

Graphic Era Hill University, Bhimtal, Uttarakhand

March-2025

Graphic Era Hill University
BHIMTAL CAMPUS

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the Synopsis entitled **"Remote File Manager"** in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science & Engineering of the Graphic Era Hill University, Bhimtal campus and shall be carried out by the undersigned under the supervision of **Mr. Prince Kumar, Assistant Professor**, Department of Computer Science & Engineering, Graphic Era Hill University, Bhimtal.

Aayush Mehra	2261055
Aman Negi	2261086
Himanshu Joshi	2261268
Tanuj Joshi	2261571

The above mentioned students shall be working under the supervision of the undersigned on the
"Remote File Manager"

Signature	Signature
Supervisor	**Head of the Department**

Internal Evaluation (By DPRC Committee)

Status of the Synopsis: Accepted / Rejected

Any Comments:

Name of the Committee Members:	**Signature with Date**
1.	
2.	

Table of Contents

Chapter 1

Introduction and Problem Statement

1. Introduction

A Remote File Manager is a software application that enables users to access, manage, and manipulate files stored on a remote device over a network or the internet. It eliminates the need for physical access to a system, making it highly useful for individuals, businesses, and IT administrators who need to manage files across multiple devices securely.
Using various network protocols like FTP (File Transfer Protocol), SFTP (Secure File Transfer Protocol), SMB (Server Message Block), and NFS (Network File System), a remote file manager allows users to view, transfer, edit, and organize files on remote machines just as they would on a local system.

2. Problem Statement

In the modern digital landscape, individuals and businesses are increasingly relying on remote servers, cloud storage, and virtual machines to store and manage critical files. However, the complexity and technical knowledge required to manage files remotely often create significant barriers for non-technical users, making it difficult to perform basic file operations efficiently.
Existing solutions, such as FTP clients and cloud storage applications, often lack seamless integration, user-friendly interfaces, and support for multiple platforms, creating a fragmented experience. Additionally, many of these tools require manual configuration or are limited to specific protocols, leading to increased friction and decreased productivity.

To address these challenges, this project will develop a Remote File Manager that simplifies the process of managing files on remote servers or cloud storage systems, making it easier for both technical and non-technical users to interact with their remote files securely and efficiently.

Chapter 2

Background/ Literature Survey

2.1 Existing Remote File Manager Functionalities

1. Remote File Access and Navigation: Browse Remote Files, Directory Navigation
2. File Upload and Download: Upload Files, Download Files
3. File Manipulation: Rename Files/Directories, Delete Files/Directories, Move or Copy Files/Directories, Create Files/Folders
4. Search and Filter: Search for Files, Filter by File Type
5. File Permissions Management: Set Permissions, Change Ownership
6. Secure File Transfers: Encryption, Authentication, Integrity Checks
7. File Synchronization
8. File Compression and Decompression
9. Cross-Platform Support: Multi-Device Access, Web Interface
10. Backup and Restore: Scheduled Backups, Restore Files

2.1 Research on Remote File Manager

Remote file management has evolved significantly over the past decades, from simple FTP clients to comprehensive cloud-based solutions. While traditional methods focused on secure file transfer, modern systems now aim to provide cross-platform compatibility, high performance, enhanced security, and ease of use. The shift toward cloud storage, distributed file systems, and web-based management tools has simplified remote file management and expanded its use across industries. However, challenges like data security, system performance, and multi-platform support remain key areas of ongoing research and development.

Chapter 3

Objectives

The primary objective of this project is to design and implement a Remote File Manager in python with the following features:

1. Browse Remote Files: Allows users to browse the file system of remote servers or cloud storage in a hierarchical structure (folders and subfolders), similar to a local file explorer.
2. Rename Files/Directories: Users can rename files or directories on the remote server.
3. Delete Files/Directories: Allows users to delete files or entire directories remotely.
4. Move or Copy Files/Directories: Enables the transfer of files or directories between locations on the remote system (including between remote and local systems).
5. Create Files/Folders: Users can create new files or directories on the remote server.
6. Set Permissions: Allows users to view and set file permissions (e.g., read, write, execute) for remote files and directories, controlling who can access or modify them.
7. Compress Files: Allows users to compress files or directories into formats such as ZIP or TAR for easier transfer or storage.
8. Decompress Files: Enables users to extract compressed files (e.g., ZIP, TAR) directly on the remote server without downloading them first.
9. Auto-Sync: Automatically synchronizes files and directories between the local machine and the remote server or between multiple remote locations to keep them updated.
10. Search for Files: Provides a search functionality that allows users to find files based on filename, type, date, or other parameters.

.

Chapter 4

Hardware and Software Requirements

4.1 Hardware Requirements

Sl. No	Name of the Hardware	Specification
1	**Processor**	Minimum Intel Core i3 or equivalent
2	**RAM**	4GB (Recommended: 8GB for optimal performance)
3	**Network**	Internet connection for real-time communication
4	**Storage**	10GB free disk space

4.2 Software Requirements

Sl. No	Name of the Software	Specification
1	Python	Programming language used for client-server communication and encryption
2	Socket	Handles networking communication between client and server
3	MYSQL	For storing user data, file metadata (file names, paths, timestamps), access logs, etc
4	Threading	Enables handling multiple users simultaneously using threads
5	Tkinter	Used for creating the graphical user interface (GUI) for the client application
6	VScode	Compiler for compiling the source code

Chapter 5

Possible Approach/ Algorithms

5.1 Algorithm

1. **User Authentication:**
    - Users enter a username and password.
    - The password is verified against stored on the server.
2. **Secure File Transfer:**
    - The client encrypts file data using Fernet encryption before uploading.
    - Files are securely sent to the server over an encrypted channel.
    - The server decrypts and stores files in the user’s directory.
    - For downloads, the server encrypts file data before sending it to the client.
3. **Remote File Operations:**
    - The Client sends encrypted operation request (like list, delete, rename, move) with a valid session token.
    - The server verifies the session and performs the requested file operation.
    - Results are sent back to the client in encrypted form.
4. **Multi-User Access Management:**
    - The server maintains user-specific directories for secure file isolation.
    - Access control ensures that users cannot perform operations on files outside their permitted scope.
    - Optional role-based permissions (e.g., read-only vs read-write) can be enforced.

5.2 Implementation Strategy

1. **Develop the Server:**
    - Implement a multi-threaded server to handle multiple clients.
    - Integrate user authentication.
2. **Develop the Client:**
    - Create a GUI-based client using Tkinter.
    - File explorer to view, upload, download, delete, and rename files.
    - Secure communication layer for sending encrypted file data and commands.
    - Progress indicators for file uploads/downloads.
3. **Testing and Optimization:**
    - Test encryption and authentication for security vulnerabilities.
    - Simulate multiple users performing file operations concurrently.

References

1. Kurose, J. F., & Ross, K. W. (2017). Computer Networking: A Top-Down Approach. Pearson.
2. Tanenbaum, A. S., & Bos, H. (2015). Modern Operating Systems. Pearson.
3. Stevens, W. R. (1998). Unix Network Programming, Volume 1: The Sockets Networking API. Prentice Hall.
4. Stallings, W. (2013). Data and Computer Communications. Pearson.
5. Geeks for Geeks. (n.d.). How to Build a File Management System in Java.
6. A review: Remote file manager for Android platform using FTP4Android. (2013, July 1).