A SYNOPSIS ON

FILE SYSTEM SIMULATOR

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

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted by:

Rakshita Joshi	2261462
Abhishek Majila	2261066
Pankaj Singh Raikwal	2261410
Vivek Melkani	2261620

Under the Guidance of

Anubhav Bewerwal

Assistant Professor

Project Team ID: 46

Department of Computer Science & Engineering

Graphic Era Hill University, Bhimtal, Uttarakhand

March-2025

Graphic Era Hill University
BHIMTAL CAMPUS

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the Synopsis entitled **"File System Simulator"** 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 **Anubhav Bewerwal, Assistant Professor**, Department of Computer Science & Engineering, Graphic Era Hill University, Bhimtal.

Rakshita Joshi	2261462
Abhishek Majila	2261066
Pankaj Singh Raikwal	2261410
Vivek Melkani	2261620

The above mentioned students shall be working under the supervision of the undersigned on the
"File System Simulator"

Signature	Signature
Supervisor	**Head of the Department**

<u>Internal Evaluation (By DPRC Committee)</u>

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

- The file system plays a crucial role in managing how data is stored, organized, and retrieved on storage devices. Every operating system incorporates some type of file system which ensures data is stored efficiently and accessed quickly when needed. A File System Simulator is an educational and practical tool that allows users to understand how different file storage and allocation techniques work internally in operating systems.

- This project aims to create a GUI-based file system simulator that replicates key functions of file handling, such as creating, writing, reading, listing, and deleting files. It also visualizes how storage blocks are allocated using various allocation strategies including Contiguous, Linked, and Indexed Allocation Methods. This project is ideal for understanding core OS concepts in a hands-on, visual, and interactive way.

- By simulating these mechanisms, the system helps bridge the gap between theoretical learning and practical understanding. It is particularly useful for students and educators in the fields of computer science and software engineering who wish to explore how files are stored and managed under the hood of an operating system.

A File System Simulator serves as an educational and practical tool that bridges this gap by allowing users to visually interact with and simulate the behavior of real-world file systems. The simulator replicates essential file operations such as creating, reading, writing, deleting, and listing files using various file allocation strategies like Contiguous Allocation, Linked Allocation, and Indexed Allocation.

2. Problem Statement

File management is a fundamental and challenging component of operating systems. A file system simulator must accurately replicate the behavior of real-world file systems, and poor design or inadequate simulation strategies can lead to issues such as:

- Fragmentation – Files are stored in non-contiguous blocks, making access slower and management complex.

- Inefficient space utilization – Disk space may remain underutilized due to improper allocation methods.

- Slow file access – Improper allocation and indexing strategies can degrade file access performance.

- Lack of flexibility – Users are often unable to visualize the internal working of file systems, limiting understanding.

Traditional operating system textbooks and learning environments often fail to provide interactive, hands-on tools to help students and developers observe and analyze the inner workings of file systems. Most theoretical models only describe how file systems work without offering a practical, visual method of understanding the impact of various strategies.

This project proposes the development of a user-friendly, interactive file system simulator that enables users to:

- Simulate various file allocation methods such as Contiguous, Linked, and Indexed Allocation.
- Visualize how files are stored and accessed using different strategies.
- Observe the impact of fragmentation on disk performance and space utilization.
- Create, read, write, and delete files, reflecting realistic file system behavior.
- Experiment with different scenarios to compare efficiency, space management, and performance.
- Understand the role of OS concepts like memory management, file indexing, and disk allocation.

This project aims to bridge the gap between theory and practice by providing a visual, interactive tool to simulate file system operations. By implementing different storage allocation methods, users can compare their efficiency and limitations, making learning more engaging and practical.

Chapter 2

Background/ Literature Survey

The file system is a fundamental component of modern operating systems, responsible for organizing, storing, retrieving, and managing data on storage devices. It acts as an interface between users and the hardware, abstracting the complexity of data storage and presenting it in a structured and accessible manner. With the increasing size of data and complexity of applications, the efficient management of file systems has become more crucial than ever. Academic understanding of file systems is often limited to theoretical study, which lacks the visualization and hands-on experience necessary for complete comprehension. This is where a File System Simulator plays a significant role.

Historically, several allocation methods have been employed in file systems to optimize disk space usage and data retrieval efficiency. The Contiguous Allocation method stores a file in a single continuous block of memory, offering high-speed sequential access. However, it suffers from external fragmentation and difficulty in accommodating file growth. The Linked Allocation method stores file blocks scattered across the disk and connects them using pointers. It resolves the fragmentation issue but causes slower access due to the need to follow links. Indexed Allocation, on the other hand, uses an index block to store all block addresses of a file, balancing speed and flexibility. These methods are implemented in various real-world file systems like FAT (File Allocation Table), NTFS, each choosing a method that best suits their performance and scalability goals.

Numerous research studies and educational tools have explored file system structures and operations. Works such as Andrew S. Tanenbaum’s Modern Operating Systems and Abraham Silberschatz’s Operating System Concepts provide detailed theoretical foundations for file systems and their implementations. However, these textbooks do not offer interactive or visual elements that can aid in experiential learning. To address this gap, educators have proposed simulation-based learning tools. For example, universities often include basic file system simulators in OS laboratory coursework, typically developed using command-line interfaces or basic GUI frameworks. Still, many of these tools are limited in scope and do not support dynamic file operations or multiple allocation methods.

Recent advancements in programming environments, such as Python’s ease of use and libraries like Tkinter and Streamlit, have enabled the development of user-friendly educational simulators. These tools allow learners to simulate and visualize file system behavior interactively, reinforcing concepts such as memory management, allocation strategies, file access, and fragmentation. Streamlit, in particular, enables the creation of web-based applications with minimal coding effort, offering visualizations and real-time interactions without requiring deep web development knowledge.

This File System Simulator project builds upon this literature and fills the gap by integrating a graphical interface with the core file system mechanisms. By simulating different allocation strategies, managing free and occupied blocks, and performing file operations like create, read, write, and delete, the simulator becomes a powerful tool for both learning and teaching. It provides a practical platform for experimenting with real-world file system behaviors without altering the underlying OS, thus fostering a deeper understanding of operating system principles in a controlled and visual environment.

.

Chapter 3

Objectives

The specific objectives of the proposed work are as follows:

1. To develop an interactive file system simulator with multiple allocation strategies.

- Design a user-friendly interface using Streamlit to allow users to perform file operations easily.
- Support file creation, deletion, reading, and writing through a simple and intuitive GUI.
- Allow dynamic file management without needing to interact with the actual operating system.

2. To implement different file allocation methods such as Contiguous, Linked, and Indexed Allocation.

- Simulate the behavior of each allocation method using virtual block structures.
- Compare their efficiency in terms of storage utilization, access time, and fragmentation handling.
- Provide internal views to help users understand how files are stored and accessed.

3. To visualize file storage and memory blocks in real-time.

- Display allocated and free blocks using visual indicators.
- Show how each file occupies disk space based on the chosen allocation strategy.
- Make fragmentation clearly visible and interpretable by users.

4. To simulate file operations such as create, delete, write, and read with backend data handling.

- Enable users to enter content and simulate actual file writing to disk blocks.
- Manage file metadata and file content using JSON storage for persistent simulations.
- Allow partial or full content reading and deletion of files with automatic space recovery.

5. To reinforce understanding of file system and OS-level memory management concepts.

- Help students and learners connect theoretical concepts with practical behavior.
- Encourage experimentation with allocation strategies to understand real-world implications.
- Support learning by offering a safe, visual simulation of low-level file management.

6. To provide a platform for future enhancements and academic use.

- Lay the groundwork for implementing directory structures, user permissions, or disk scheduling.
- Facilitate improvements like fragmentation analysis tools and data visualization dashboards.
- Serve as a base for further development by students, educators, and researchers.

Chapter 4

Hardware and Software Requirements

4.1 Hardware Requirements

Sl. No	**Name of the Hardware**	**Specification**
1	Processor	Intel Core i5/i7 or equivalent
2	RAM	Minimum 4GB
3	Hard Disk	Minimum 500GB
4	Monitor	Standard HD display

4.2 Software Requirements

Sl. No	**Name of the Software**	**Specification**
1	Operating System	Windows/Linux/MacOS
2	Development Tools	Visual Studio Code, Jupyter Notebook
3	Programming Language	Python
4	Frameworks/Libraries	Streamlit,JSON,matplotlib,NumPy

Chapter 5

Possible Approach / Algorithms

The File System Simulator is designed to replicate the essential behaviors of an operating system’s file management functionalities. It simulates the creation, deletion, reading, and writing of files, along with the allocation of disk space using different file allocation strategies. The project leverages the following core approaches and algorithms:

1.Contiguous Allocation

In this method, each file occupies a set of contiguous blocks on the disk.It is implemented by scanning the list of free blocks to find a sequence of free spaces large enough to hold the file.

2.Linked Allocation

This method overcomes the problem of external fragmentation by storing file blocks anywhere on the disk.Each block contains a pointer to the next block in the file, forming a linked list structure.

3.Indexed Allocation

In this strategy, an index block is created that contains pointers to all the blocks used by a file.It allows direct access to file blocks and supports both sequential and random access.The simulator picks one free block as an index and then allocates additional blocks for file content.

4.File Metadata Management

Each file’s metadata includes its name, size, allocation method, and list of blocks used.A dictionary structure is used to manage metadata, simulating a simple file allocation table (FAT).This approach ensures quick access and modification of file details.

5.Memory Block Simulation

The simulator creates a virtual disk space of fixed size .A free block list is maintained to keep track of unused blocks.When a file is created, blocks are allocated based on the selected method and removed from free list.When a file is deleted, its blocks are returned to free list, simulating deallocation.

6.Read/Write Simulation

Read operation fetches content from stored file data.Write operation updates content in a specific file, stored in a dictionary structure.

7.Graphical User Interface (GUI)

Developed using Streamlit or Tkinter, the GUI allows users to perform file operations interactively.Inputs like file name, size, and allocation method are taken from the user and processed accordingly.Outputs such as file status, content, and block allocation are displayed visually.

Progress Visualization

The File System Simulator will feature an intuitive and dynamic interface built using Streamlit, offering real-time feedback and visual insights into file system operations. The simulation will integrate graphical elements to help users understand core OS file management concepts interactively.

Key visual components include:

- Disk Block Visualization: Display memory as a grid of blocks showing allocated, free, and indexed sections using color-coded indicators.
- File Allocation Process: Visually demonstrate how files are stored using Contiguous, Linked, and Indexed allocation strategies.
- File Table Display: Show file metadata like name, size, starting block, and allocation details in real time.
- Live Operation Updates: Real-time updates for file creation, deletion, reading, and writing.
- Fragmentation Indicators: Highlight fragmentation issues (internal/external) as they occur.
- Performance Metrics: Provide instant feedback on memory usage, number of files, and free space after every action.

With seamless user interaction and animated transitions, the simulator makes learning file systems engaging and impactful—ideal for both academic learning and practical understanding.

References

[1] Silberschatz, A. (2003). *Operating Systems Concepts*

[2] Tanenbaum, A. S., & Wetherall, D. (2011). *Computer networks*. Pearson Education.

[3] Stallings, W. (2009). *Operating systems: Internals and Design Principles*. Prentice Hall.

[4] Computer Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual.

[5] Tanenbaum, A. S. (2009). *Modern operating systems*. Pearson-Prentice Hall.

[6] Patt, Y. N., & Patel, S. J. (2004). *Introduction to computing systems: From Bits and Gates to C and Beyond*.