Instructor

Dr. Zubair Ahmad

FILE EXPLOER WITH THREAT DETECTION

Ehtisham Ul Hassan (202152)

ISLAM UDDIN (2024247) Walwal Khan Afridi (2024660)

Problem Description and Project Aim

The project aims to simulate a **file management and security system** that allows users to navigate, search, and organize files efficiently within a virtual file system. Modern operating systems and applications require effective file exploration and management to prevent unnecessary time and resource wastage. The project is based and implemented through Data structures and Algorithms we studied.

In addition to file organization, the project also focuses on **data security**. When a user wants to protect a file or share it securely, the system can **encrypt** it using basic encryption algorithms. The encrypted text becomes unreadable without the correct decryption key.

Furthermore, a **malware detection feature** is included. It scans uploaded files and matches their contents against known malware patterns to identify possible threats. This integration combines **data structure concepts** with practical cybersecurity functionality.

Key Data Structures and Algorithms and integration of concepts learned in class.

Key Data Structures

(These may change/added during implementation.)

1. **Tree** – represents the hierarchical folder and file structure (parent–child relationships).
2. **Stack** – manages navigation history (back and forward operations).
3. **Queue** – handles scanning and encryption tasks sequentially.
4. **Hash Table / Map** – stores and verifies file hashes for integrity checking.
5. **Linked List** – maintains operation logs (create, delete, scan, encrypt actions).
6. **Graph** – models file dependencies and visualizes infection spread.

Main Algorithms

1. **Tree Traversal (DFS/BFS)** – for navigating folders and displaying contents.
2. **Sorting (Quick Sort / Merge Sort)** – for sorting files alphabetically or by size.
3. **Search (Linear / Binary Search)** – for finding files by name.
4. **Hashing Algorithm (custom or SHA-like)** – to generate unique file fingerprints for tamper detection.
5. **String Matching (KMP / Rabin–Karp)** – for malware pattern detection inside files.
6. **Encryption (Caesar Cipher / Vigenère / XOR)** – for secure file encryption and decryption.
7. **Graph Traversal (DFS)** – for analysing infection spread between related files.

Data Flow: Input, Processing, and Output

When the program runs, a **menu** is displayed to the user with available actions:

1. Create a folder/file
2. Move / copy / rename a file or folder
3. Delete a folder/file
4. Search for a folder/file
5. Upload a folder/file
6. Encrypt a file
7. Decrypt a file
8. Create a hash
9. Check for malware
10. Exit

Processing Flow

- Based on the user's selection, the program executes the corresponding operation:
  - **Create / Delete / Rename / Move:** Updates the virtual file system structure (Tree).

- **Search / Sort:** Uses search and sorting algorithms to display results.
- **Upload / Encrypt / Decrypt:** Handles file content operations and security features.
- **Hash Creation:** Generates a file hash for integrity verification.
- **Malware Detection:** Scans file content using string-matching algorithms to detect malicious patterns.

Output

Results (such as success messages, detected malware alerts, or encrypted data) are displayed directly to the user in a structured menu format.