# File Organizer App: An Automated System Call-Based Approach to Intelligent File System Management

C Murali Madhav
Department of Computer Science and Artificial Intelligence
Rishihood University
Sonipat, Haryana, India
cmurali.m23csai@nst.rishihood.edu

Ravi Yadav
Department of Computer Science and Artificial Intelligence
Rishihood University
Sonipat, Haryana, India
ravi.y23csai@nst.rishihood.edu.in

*Abstract*—This paper presents the File Organizer App, an advanced Operating Systems engineering project designed to automate and enhance file system management using low-level POSIX system calls. The system combines a high-performance C backend with a Next.js visualization layer to deliver intelligent file classification, workspace virtualization, safe deletion architecture, AI-assisted natural language commands, and collaborative sharing capabilities. By leveraging `dirent.h`, `sys/stat.h`, and atomic file operations, the application ensures efficient directory traversal, metadata tracking, and controlled file manipulation within a sandboxed environment. Experimental results demonstrate the system's ability to transform unstructured directories into organized hierarchies while maintaining safety, scalability, and extensibility across Linux, macOS, and WSL environments.

*Index Terms*—Operating Systems, System Calls, File Management, C Programming, Next.js, Virtual Workspace, NLP

## I. INTRODUCTION

**Problem Definition:** Modern computing environments experience exponential file growth, leading to cluttered directories and inefficient manual organization.

**Project Goal:** To develop a system-level file management framework that integrates OS-level operations with modern full-stack and AI capabilities.

**Scope:** The system supports Linux, macOS, and WSL. It integrates a C backend for file operations and a Next.js frontend for visualization, AI interaction, and collaborative features.

## II. LITERATURE REVIEW

Automated file management systems reduce manual cognitive load while improving consistency and efficiency. POSIX-compliant system calls such as `readdir()`, `mkdir()`, and `rename()` provide direct interaction with file systems.

Modern trends demonstrate hybrid architectures combining low-level backends with web-based visualization layers for improved transparency and user interaction.

## III. METHODOLOGY

### A. System Architecture

The system follows a decoupled architecture:

- **Backend Layer (C):** Handles directory traversal, file classification, metadata tracking, and bin operations.
- **Frontend Layer (Next.js):** Provides visualization, AI command interface, search, tagging, and workspace management.

### B. System Calls Used

- `mkdir(2)` – Creates directories
- `readdir(3)` – Iterates directory entries
- `rename(2)` – Atomic file movement
- `stat(2)` – Retrieves file metadata

### C. File Classification Logic

TABLE I
FILE CLASSIFICATION LOGIC

| File Type | Extensions | Destination Folder |
|---|---|---|
| Documents | .txt, .pdf, .docx, .doc, .xlsx, .pptx | Documents/ |
| Images | .jpg, .jpeg, .png, .gif, .bmp, .svg | Images/ |
| Audio | .mp3, .wav, .aac, .flac, .ogg | Audio/ |
| Videos | .mp4, .mkv, .avi, .mov, .wmv | Videos/ |
| Others | All other types | Others/ |

## IV. EXTENDED SYSTEM CAPABILITIES

### A. Core File Management Capabilities

The system supports directory creation, renaming, downloading, and batch operations.

**Bulk Operations:** Multi-selection allows batch move, delete, or download.
**File Uploads:** Single and batch uploads supported.
**Visual Modes:**
- List View with metadata columns
- Grid View with thumbnails

### B. Safe Deletion and Recovery Architecture (Bin System)

**Soft Deletion:** Files are redirected to a hidden `.bin` directory.
**Metadata Tracking:** Each item stores:

- Original path
- Deletion timestamp
- UUID identifier

**Permanent Deletion:** Irreversible removal available within Bin interface.

## C. Advanced Organization and Customization

**Workspace Virtualization:** Users create isolated sandbox directories called Workspaces.

**Dynamic Folder Color Coding:** Custom color labels persist even after bin restoration.

**Favorites and Tagging:** Starred items appear in quick-access panel. Tagging improves retrieval.

## D. Intelligent Search and Retrieval

**Debounced Live Search:** Real-time filtering without excessive backend calls.

**Quick Navigation:** Sidebar tracks Recents, Favorites, and Shared items.

## E. AI-Assisted File Operations

**Natural Language Commands:** Plain-language prompts to create or delete Workspace , Organize Files .

**Context-Aware Suggestions:** Directory analysis enables intelligent action recommendations.

**Multi-Agent Tool Integration:** AI orchestrates folder creation, batch movement, and semantic search.

## F. Collaboration and Sharing

**Secure Link Generation:** Unique shareable links for files/folders.

**Shared Tracking:** Sidebar logs actively shared paths.

## V. RESULTS AND DISCUSSION

Testing demonstrated successful:

- Automatic file categorization
- Safe deletion and restoration
- Multi-file batch processing
- AI-driven command execution
- Workspace isolation

The architecture scales effectively across Linux, macOS, and WSL.
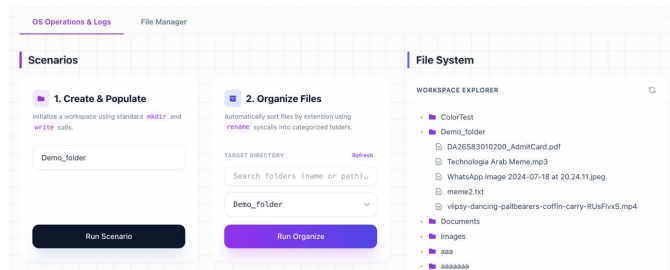
## A. Visual Results

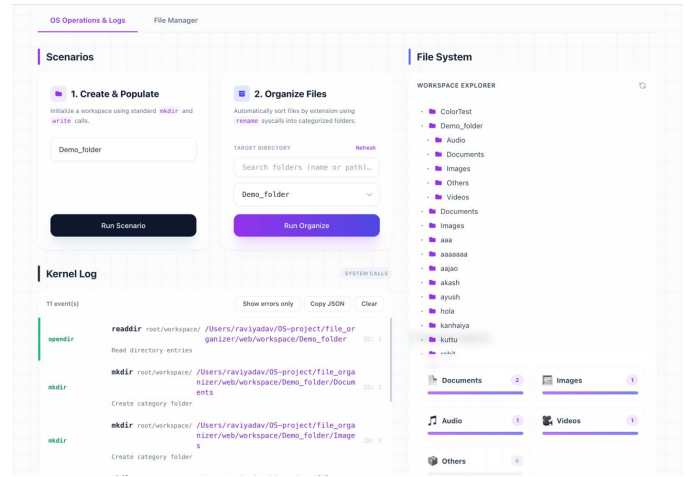

Fig. 1. Initial Unorganized Directory Structure



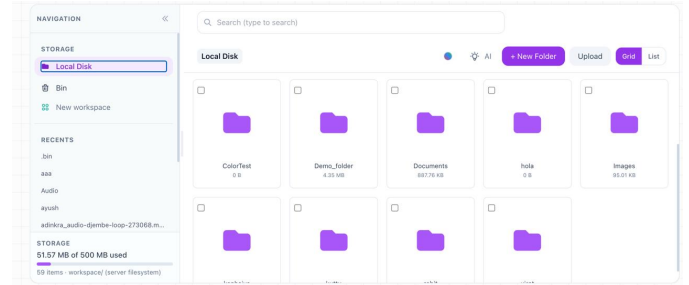Fig. 2. Organized Directory After Classification
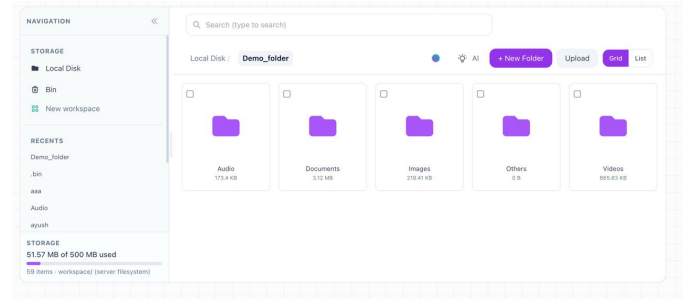


Fig. 3. File Manager Interface



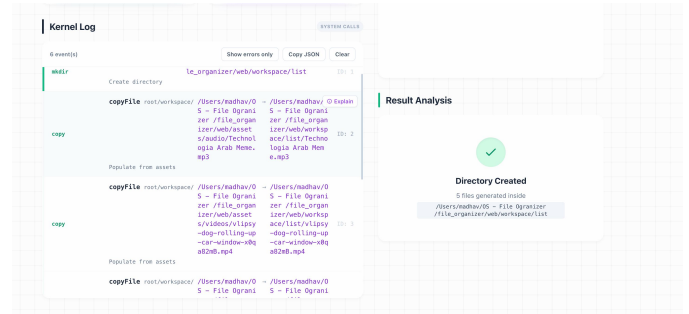Fig. 4. Organized Folders inside Directory



Fig. 5. Kernel log Interface
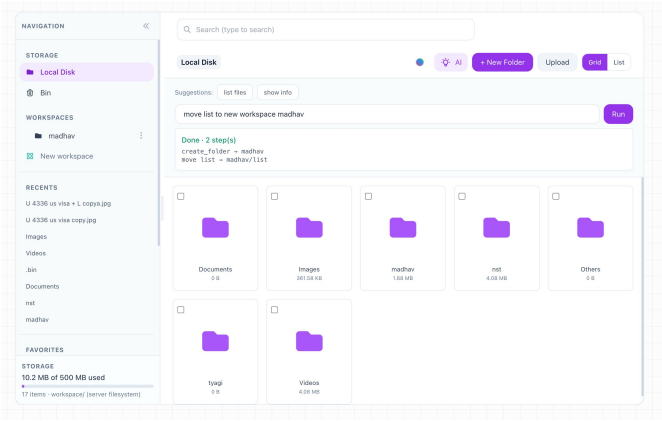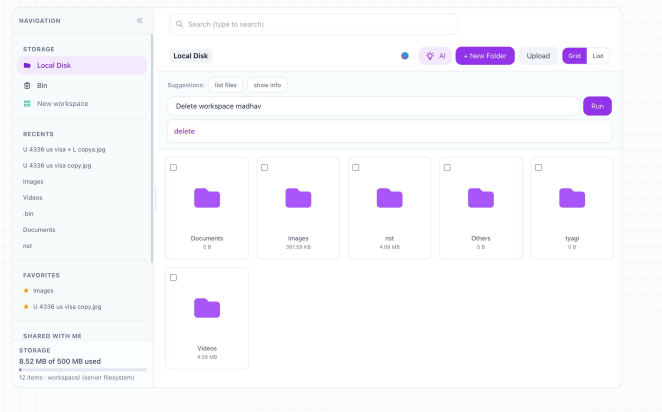
Fig. 6. Workspace creation by AI
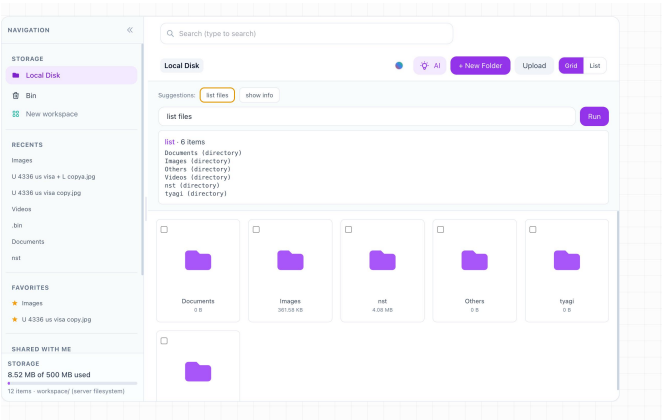


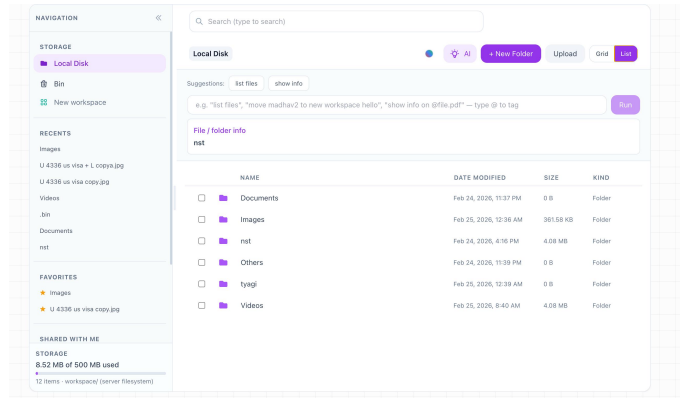Fig. 7. Deleting Workspace by AI



Fig. 8. Listing Files/Folders in Grid



Fig. 9. List View of Files/Folders

## VI. SYSTEM ENHANCEMENT IMPACT

The integration of virtualization, soft-deletion architecture, AI-assisted operations, and collaboration transforms the system from a basic OS utility into a scalable file management framework.

This project demonstrates:

- Practical application of POSIX system calls
- Metadata persistence strategies
- Secure sandbox enforcement
- Human-computer interaction via NLP

## VII. CONCLUSION AND FUTURE WORK

The File Organizer App successfully bridges low-level system programming with modern full-stack and AI-driven design.

**Future Enhancements:**

- Recursive multi-depth scanning
- Duplicate detection via hashing
- Role-based access control
- Cloud storage integration

## REFERENCES

[1] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," Proc. 19th ACM Symposium on Operating Systems Principles (SOSP), 2003.

[2] M. Satyanarayanan, "A Survey of Distributed File Systems," Annual Review of Computer Science, vol. 4, pp. 73–104, 1990.

[3] A. S. Tanenbaum and H. Bos, Modern Operating Systems, 4th ed., Pearson, 2015.

[4] R. Love, Linux System Programming, 2nd ed., O'Reilly Media, 2013.

[5] W. R. Stevens and S. A. Rago, Advanced Programming in the UNIX Environment, 3rd ed., Addison-Wesley, 2013.

[6] M. K. McKusick, W. N. Joy, S. J. Leffler, and R. S. Fabry, "A Fast File System for UNIX," ACM Transactions on Computer Systems, vol. 2, no. 3, pp. 181–197, 1984.

[7] C. A. N. Soules, G. R. Goodson, J. D. Strunk, and G. R. Ganger, "Metadata Efficiency in Versioning File Systems," Proc. 2nd USENIX Conference on File and Storage Technologies (FAST), 2003.

[8] D. Mazieres, "A Toolkit for User-Level File Systems," Proc. USENIX Annual Technical Conference, 2001.

[9] P. J. Denning, "The Working Set Model for Program Behavior," Communications of the ACM, vol. 11, no. 5, pp. 323–333, 1968.

[10] J. H. Howard et al., "Scale and Performance in a Distributed File System," ACM Transactions on Computer Systems, vol. 6, no. 1, pp. 51–81, 1988.

[11] B. Kahanwal, "File System – A Component of Operating System," arXiv preprint arXiv:1312.1810, 2013.

[12] R. H. Katz, "Contemporary File System Architecture," IEEE Computer, vol. 23, no. 5, pp. 23–32, 1990.

[13] S. Quinlan and S. Dorward, "Venti: A New Approach to Archival Storage," Proc. USENIX FAST Conference, 2002.

[14] G. R. Ganger and M. F. Kaashoek, "Embedded Inodes and Explicit Grouping: Exploiting Disk Bandwidth for Small Files," Proc. USENIX Annual Technical Conference, 1997.

[15] J. Nielsen, Usability Engineering, Morgan Kaufmann, 1994.

[16] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol – HTTP/1.0," IETF RFC 1945, 1996.