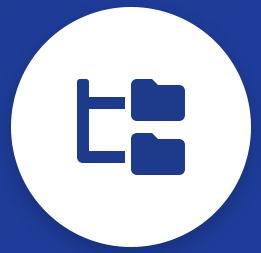


File System Directory Tree Analyzer

Comprehensive Analysis of File System Directories

Project Team

Team Members: Mohamed Osama Zahran, Mohamed Elsayed Zahran, Ali Ibrahim Fahmy

Academic Advisor: E: Moustafa E.

Department/Institution: Shorouk Academy

May 2025

Table of Contents

Introduction

Problem Statement

- File systems become increasingly complex and disorganized over time
- Difficulty in visualizing directory structures and understanding storage usage
- Manual identification of duplicate files and wasted storage space is time-consuming
- Locating specific files across large directory structures is challenging

Project Motivation

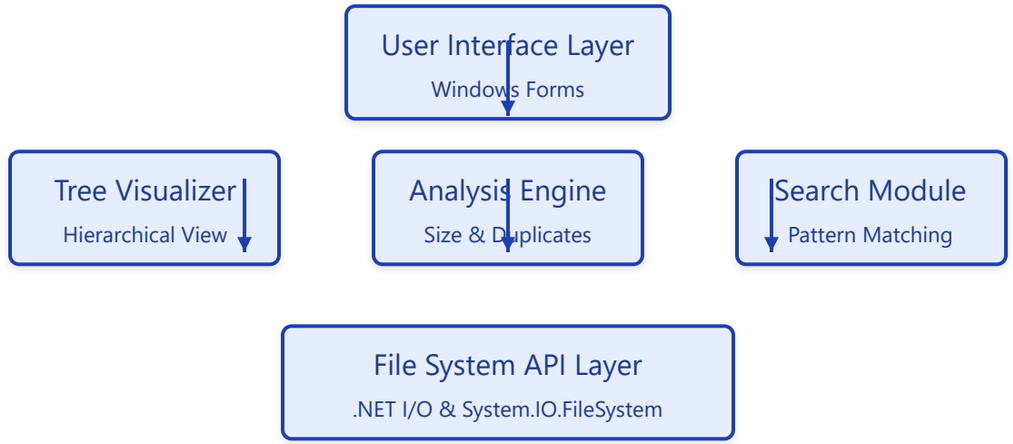
- Need for intuitive visual representation of file system hierarchies
- Demand for tools that provide storage usage insights at a glance
- Desire to improve file system organization and reduce redundancy
- Requirement for enhanced file discovery and management capabilities

Project Objectives

- Create a comprehensive visualization tool for directory structures
- Develop advanced size analysis features to identify storage usage patterns
- Implement powerful search capabilities with multiple filtering options
- Design efficient duplicate file detection algorithms to optimize storage
- Build a responsive, user-friendly interface for improved file system management

System Overview

High-Level Architecture



Key Components

- UI Components: Tree view, tabbed interface, progress indicators
- Directory Scanner: Recursive file system traversal
- Size Analyzer: Calculates directory sizes, statistics
- File Matcher: Pattern matching, search algorithms
- Duplicate Detector: Hashing algorithms, content comparison
- Async Manager: Background processing, UI responsiveness

Technology Stack

.NET 8.0 | C# | Windows Forms

- System.IO.FileSystem: Directory and file manipulation
- Task Parallel Library: Asynchronous processing
- System.Security.Cryptography: File hashing
- System.Text.RegularExpressions: Pattern matching
- System.Drawing: UI components and visualization

Main Features

Directory Tree Visualization

- Hierarchical view of directories
- ASCII-style tree structure
- Proper indentation for nesting
- File metadata display
- Real-time structure updates

Directory Size Analysis

- Total size calculations
- Human-readable formats (KB, MB, GB)
- Size distribution statistics
- Largest files identification
- Space usage percentage breakdown

File Search Capabilities

- Wildcard pattern matching (*, ?)
- Extension-based filtering
- Case sensitivity options
- Regular expression support
- Size-based search criteria

Duplicate File Detection

- MD5/SHA-256 content hashing
- Group visualization of duplicates
- Wasted space calculation
- Color-coded identification
- Parallel processing for speed

Modern UI

- Tabbed interface organization
- Expandable tree nodes
- Sortable list views
- Progress indicators
- Responsive during operations

Performance Optimizations

- Asynchronous processing
- Background worker threads
- Efficient memory management
- Pre-filtering techniques
- Cancelable long-running tasks

Directory Tree Visualization

Tree Structure Representation

```
C:\Documents\
├── Projects\
│   ├── Project_A\
│   │   ├── src\
│   │   │   ├── main.cs          (45 KB)
│   │   │   └── utils.cs         (18 KB)
│   │   ├── tests\
│   │   │   └── test_main.cs     (12 KB)
│   │   └── README.md            (4 KB)
│   └── Project_B\
```

ASCII-style tree visualization with proper indentation shows the hierarchical relationship between directories and files.

Key Capabilities

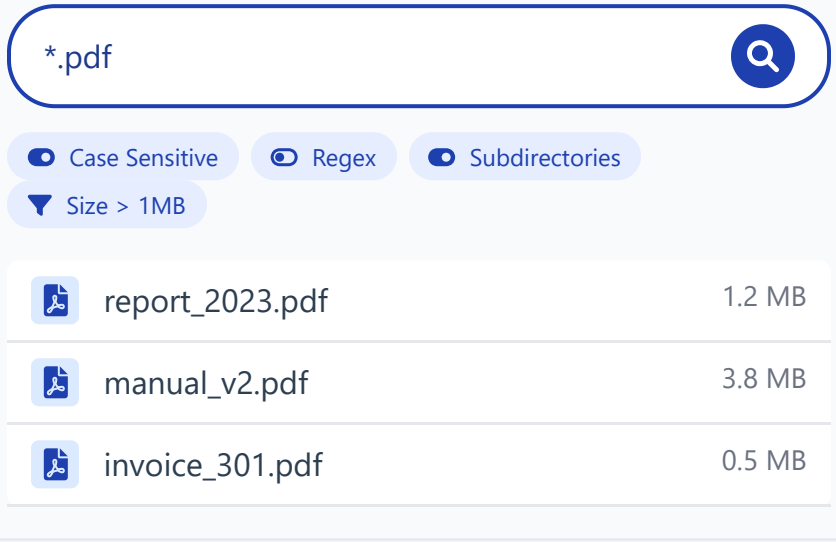
- **Hierarchical Representation** - Nested directory structure with clear parent-child relationships
- **Visual Connectors** - ASCII line characters (├── , │ , └──) show relationships
- **Metadata Display** - Shows file sizes in human-readable format
- **File Type Identification** - File extensions clearly visible
- **Expandable Nodes** - Interactive collapsing/expanding of directories
- **Real-time Updates** - Tree refreshes automatically when file system changes

Implementation Highlights

- **Recursive Directory Traversal** - Efficiently maps directory structure
- **Character Mapping Algorithm** - Generates proper ASCII tree characters
- **FileSystemWatcher Integration** - Enables real-time directory monitoring

File Search Capabilities

Advanced Search Interface



Pattern Matching Examples

`*.jpg` → All JPEG images

`report_????.pdf` → Reports with 4-character years

`data[0-9]*.csv` → CSV files starting with "data" followed by numbers

Key Capabilities

- **Wildcard Pattern Matching** - Support for wildcards like * (any characters) and ? (single character)
- **Extension-Based Search** - Find files by type (.pdf, .jpg, .docx)
- **Case Sensitivity Options** - Toggle between case-sensitive and case-insensitive search
- **Regular Expression Support** - Advanced pattern matching with regex syntax
- **Size-Based Filtering** - Find files larger or smaller than specified sizes
- **Date-Based Filtering** - Search by creation, modification, or access date

Implementation Highlights

- **Optimized String Matching** - Efficient algorithms for pattern comparison
- **Multi-threaded Search** - Parallel processing for faster results in large directories
- **Search History** - Remembers recent searches for quick access
- **Sortable Results** - Order results by name, size, type, or date

Duplicate File Detection

Detection Process

Duplicate File Groups

Image Group #1 `f8e3...`

vacation.jpg Pictures/Vacation/	2.4 MB
beach_copy.jpg Documents/Travel/	2.4 MB
backup_img3.jpg Backups/Summer2023/	2.4 MB

Potential Space Savings	Duplicate Files Found	Duplicate Groups
3.2 GB	**128**	**42**

Key Capabilities

- **Content-Based Hashing** - MD5/SHA-256 algorithms provide reliable file content comparisons regardless of filename
- **Cross-Directory Detection** - Identifies duplicates across different locations in the entire directory tree
- **Intelligent Grouping** - Organizes identical files into logical groups for easier management
- **Visual Differentiation** - Color-coding distinguishes original files from duplicates
- **Storage Impact Analysis** - Calculates potential space savings from removing duplicates

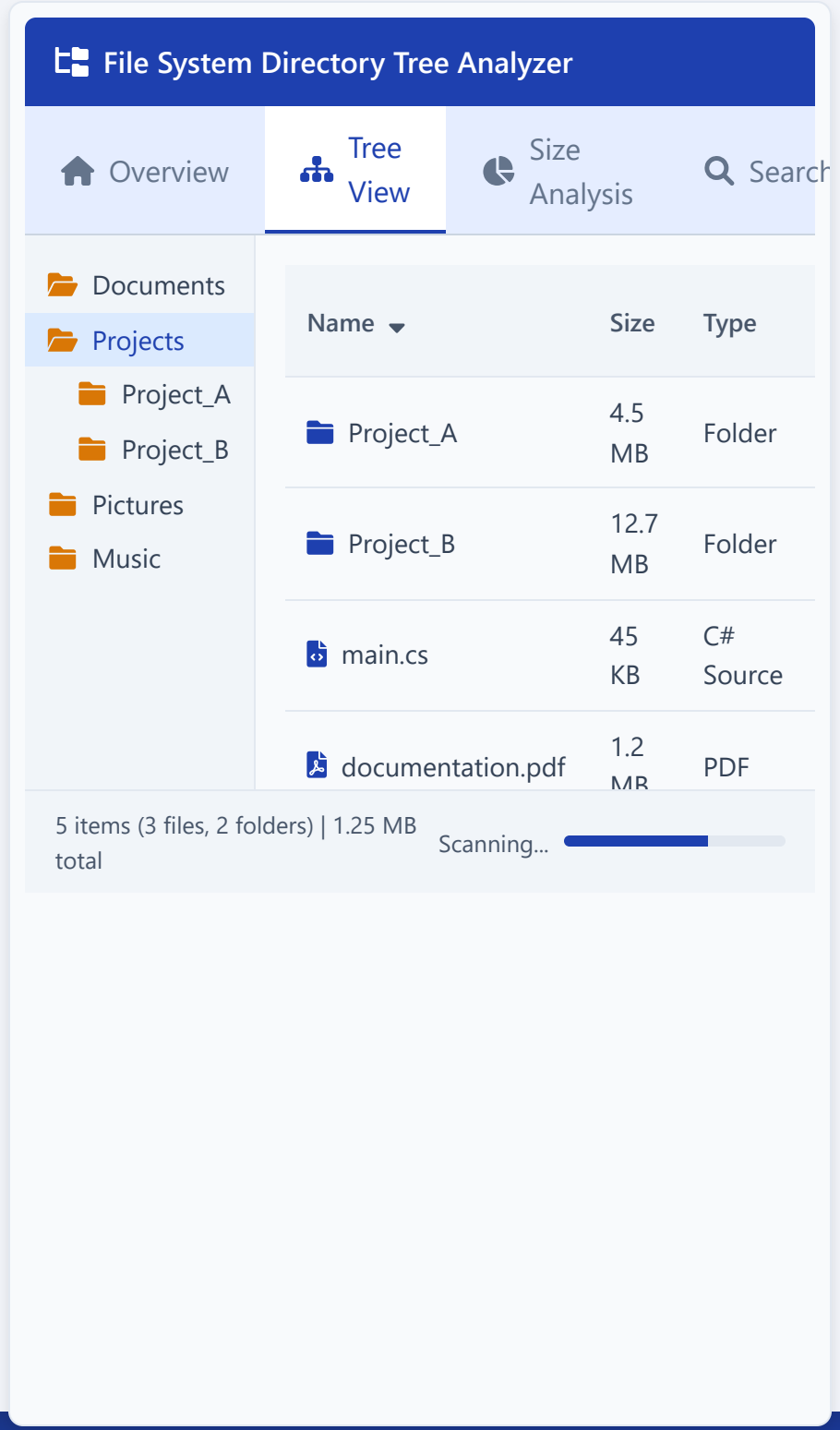
Performance Optimizations

- **Size-based Pre-filtering** - Files with different sizes cannot be duplicates, allowing early elimination
- **Parallel Processing** - Utilizes multiple threads to hash and compare files simultaneously
- **Progressive Scanning** - Processes files in batches, providing incremental results
- **Cancelable Operation** - User can interrupt lengthy scans of large directories
- **Memory Efficiency** - Hash storage optimizations prevent excessive memory usage with large file sets

File System Directory Tree Analyzer

Modern UI

Intuitive User Interface



UI Design Principles

- **Tabbed Interface** - Organizes different functions into separate tabs for easy navigation and reduced cognitive load
- **Hierarchical Tree View** - Expandable/collapsible nodes for intuitive directory navigation
- **Sortable List Views** - Click column headers to sort by name, size, type, or date
- **Progress Indicators** - Visual feedback during long-running operations with percentage completion
- **Status Updates** - Real-time information about current operations and scanning progress
- **Consistent Color Scheme** - Flat design with blue/white palette for professional appearance

Responsiveness

- **Background Processing** - UI remains responsive during intensive operations
- **Cancelable Operations** - Users can cancel long-running tasks at any time
- **Lazy Loading** - Tree nodes load content on demand to handle large directory structures
- **Smooth Transitions** - Subtle animations make UI changes more natural and intuitive
- **Keyboard Shortcuts** - Power users can navigate efficiently using keyboard commands

Team & Acknowledgments

Mohamed Osama Zahran

Mohamed Elsayed Zahran

Ali Ibrahim Fahmy

Special Thanks

Dr. Moustafa E.
Academic Advisor - For guidance throughout the development process and technical consultation on algorithmic optimizations.

Shorouk Academy
For providing resources, lab access, and academic support throughout the project development lifecycle.

Beta Testers & Early Users
For providing valuable feedback, use cases, and suggestions that helped shape the final product's functionality and usability.

Questions & Answers

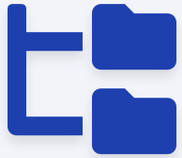
We welcome your questions about the

File System Directory Tree Analyzer

- Ask about implementation details, technology choices, or algorithm specifics
- Inquire about performance considerations or scalability challenges
- Explore potential applications for your specific use cases

Learn more about future development plans and feature enhancements

Thank You!



The File System Directory Tree Analyzer provides comprehensive solutions for:

- Directory Visualization
- Storage Analysis
- Advanced Search
- Duplicate Detection
- Performance Optimization

We appreciate your attention and feedback

File System Directory Tree Analyzer

Mohamed Osama Zahran • Mohamed Elsayed Zahran • Ali Ibrahim Fahmy

Shorouk Academy • 2023