RIPHAH
INTERNATIONAL UNIVERSITY

Project Documentation

Operating Systems (OS)

Student Name: Muhammad Abbas & Mubbashir Ahmed
Professor: Sir Yawar Abbas
SAP ID: 53316 & 55214
Section: BS CY4-1

May 6, 2025

Duplicate File Finder and Remover: Project Documentation

Table of Contents

1. Introduction
2. Background
3. Literature Review
4. Tools and Techniques
5. System Model
6. References

---

1. Introduction

The **Duplicate File Finder and Remover** is a desktop application designed to identify and eliminate duplicate files within a specified directory. This tool addresses the common issue of redundant files that consume valuable storage space and clutter file systems.

By leveraging **cryptographic hash functions**, the application ensures accurate detection of identical files, even if they have different names or locations. The user-friendly graphical interface, built with **Python's Tkinter** library, allows users to:

- Select directories
- Choose hash algorithms
- View duplicate files
- Safely delete unwanted copies

This project aims to provide an **efficient, reliable, and accessible** solution for managing duplicate files, suitable for both individual and professional use.

---

2. Background

Duplicate files often accumulate due to repeated downloads, backups, or synchronization processes. This leads to:

- Inefficient storage utilization
- Organizational challenges

Manual identification of duplicates is time-consuming and error-prone, necessitating automated tools. Existing solutions range from command-line utilities to commercial software, but many lack:

- Usability
- Customization
- Open-source accessibility

This project was motivated by the need for a **lightweight, cross-platform application** that combines:

- Robust duplicate detection
- An intuitive interface

The use of **hash-based comparison** ensures precision, while **multi-threading** and **progress tracking** enhance performance and user experience.

---

3. Literature Review

The problem of duplicate file detection has been explored in both academic and practical contexts:

- **Douceur and Bolosky (1999)**: Found that up to 20% of enterprise storage may be occupied by redundant data. Their work on single-instance storage introduced hash-based deduplication, forming the basis of this project.
- **Forman et al. (2005)**: Highlighted the use of cryptographic hash functions like **MD5**, **SHA-1**, and **SHA-256** for balancing speed and collision resistance.

Existing tools like **dupeGuru** and **FSlint** utilize similar techniques but often lack:

- Customizable hash selection
- A modern GUI

This project builds upon their foundation by adding:

- **User-driven algorithm selection**
- A **Tkinter-based responsive GUI**

---

4. Tools and Techniques

Programming Language:

- **Python**: Chosen for simplicity, rich libraries, and cross-platform compatibility.

GUI Framework:

- **Tkinter**: Used for creating the graphical interface including:
  - Directory selection
  - Progress bars
  - Treeview displays
  - Action buttons

Core Libraries:

- **hashlib**: For generating file hashes (MD5, SHA-1, SHA-256)
- **os module**: For directory traversal and file handling
- **threading + queue**: For background scanning and GUI responsiveness

Treeview Widget:

- Displays file paths, sizes, and duplication info in a tabular form.

Supported Hash Algorithms:

- **MD5**: Fast but less secure – suitable for quick scans
- **SHA-1**: Balanced option – ideal for general use
- **SHA-256**: High security – preferred for sensitive applications

---

5. System Model

The application follows a **multi-threaded modular architecture**:

1. User Interface (UI)

Built using Tkinter, includes:

- Directory selection field with “Browse” button
- Dropdown menu to select hash algorithm
- Progress bar with scan updates
- Buttons: **Scan**, **Delete Selected**, **Delete All**
- Treeview to display duplicate file data

- Status bar for system messages

2. File Scanner (Threaded Process)

- Traverses directories using `os.walk()`
- Computes file hashes using selected algorithm
- Indexes and compares hashes to detect duplicates
- Sends progress updates to main thread using queue

3. Duplicate Manager

- Collects and stores detected duplicates
- Populates the Treeview with file info

4. Deletion Handler

- Deletes user-selected duplicates
- Includes double confirmation prompts to avoid accidental deletions
- Reports outcomes to the user

Workflow Summary:

1. User selects directory and hash algorithm
2. Scanner thread counts files and computes hashes
3. Duplicate info is displayed in Treeview
4. User selects files to delete or deletes all
5. Deletion handler removes files and updates UI

The system ensures:

- **Responsiveness**: via multi-threading
- **Accuracy**: through hash comparison
- **Safety**: with confirmation dialogs

---

6. References

- Douceur, J. R., & Bolosky, W. J. (1999). *A large-scale study of file-system contents*. ACM SIGMETRICS Performance Evaluation Review, 27(1), 59–70.
- Forman, G., Eshghi, K., & Suermondt, J. (2005). *Efficient detection of large-scale redundancy in enterprise file systems*. ACM SIGOPS Operating Systems Review, 39(1), 84–91.

- Python Software Foundation. (2023). *Python Documentation: Tkinter, hashlib, os*. https://docs.python.org/3/
- dupeGuru. (2023). *Open-source duplicate file finder*. https://dupeguru.voltaicideas.net/

7. Github Link:

https://github.com/Mubbashirrrrr/Duplicate-File-FInder-and-Remover-Python.git