PROJECT ABSTRACT

Duplicate File Finder Desktop Utility

A desktop utility that finds duplicate files by SHA-256 content hash — not just file names — groups identical files, auto-suggests the newest copy to keep, and deletes the rest safely via the Recycle Bin with an exportable log — built-to-order with source code, report, PPT and viva kit.

01. Title

Duplicate File Finder Desktop Utility

02. Introduction / Background

Downloads folders accumulate “project-report-final(1).pdf”, “project-report-final(2).pdf”; datasets get copied between Documents and project folders; phone photo backups duplicate entire camera rolls. Name-based duplicate finders miss the real problem: the same content under different names, and different content under the same name. Content hashing solves it properly — files with identical bytes have identical SHA-256 hashes regardless of name, location or timestamps. This project builds a desktop duplicate finder around that principle: scan selected folders, hash every file, group identical hashes, present the groups sorted by wasted space, and clean up with per-group keep/delete control plus a safety-first deletion flow.

03. Problem Statement

Duplicate files silently consume gigabytes — students run out of disk before submissions, and manual hunting through folders is hopeless. Existing free tools either match by name only (missing renamed copies) or bundle adware. A final-year project needs an original, explainable implementation: hash-based detection the student can defend, with safety guarantees (never touch system folders, always recoverable deletion).

04. Objectives

- Implement recursive folder scanning with configurable filters (subfolders, minimum size, extensions).
- Hash file contents with SHA-256 and group files by identical hash.
- Present duplicate groups sorted by recoverable space with per-file details.
- Provide smart selection (keep newest per group) plus manual per-file checkboxes.
- Delete via the system Recycle Bin/Trash so every deletion is recoverable.
- Export a CSV deletion log for the user’s records.
- Guarantee system folders are never scanned or touched.

05. Proposed Methodology

Scanning is two-phase for speed: first group files by size (files with a unique size cannot have duplicates), then SHA-256-hash only the files in size groups with more than one member — this avoids hashing tens of thousands of unique files. Hashing reads files in chunks so multi-gigabyte files never exhaust memory. Groups are formed by exact hash equality; within a group the newest file (by modification time) is suggested as the keeper. Deletion moves files to

the Recycle Bin rather than permanent deletion, and every action is appended to a CSV log. System directories are excluded by an allowlist of user-selected scan roots.

06. System Architecture / Working

Three stages: (1) Crawler — recursive walk of selected roots honoring filters; (2) Hasher — size pre-grouping, chunked SHA-256, group formation; (3) Review UI — group list sorted by wasted space, smart-select, per-file checkboxes, delete-to-recycle-bin with log export. The working flow is: choose folders → start scan → review groups → smart-select or hand-pick → delete to Recycle Bin → verify the recovery summary.

07. Hardware / Software Requirements

- Windows/macOS/Linux desktop or laptop — no special hardware
- Python (hashlib, os) with a desktop GUI toolkit, or Electron
- File-system APIs for Recycle Bin/Trash integration
- Fully offline; no internet needed

08. Expected Outcomes

A working desktop utility that scans selected folders, finds byte-identical duplicates by SHA-256 (including renamed copies), and recovers disk space through recoverable deletion with a CSV log. Space-recovery figures in the demo are illustrative of the sample dataset; hash correctness is exact by construction of the algorithm.

09. Applications

- Reclaiming disk space before project submissions
- Deduplicating photo and dataset backups
- Cleaning download folders
- Coursework demonstration of hashing and file systems

10. Future Scope

- Similar-image detection via perceptual hashing
- Scheduled background scans
- Duplicate detection across network drives
- Dry-run mode with space forecast