Project Report

on

File Organizer

Submitted

In Partial Fulfillment of

MASTER OF COMPUTER APPLICATIONS (MCA)

Submitted by:

Akash Rawat
24/SCA/MCA/056

Under the Supervision of:

Dr Anupama Chadha , Assistant Professor

School of Computer Applications

Manav Rachna International Institute of Research and Studies

(DEEMED TO BE UNIVERSITY)

Sector-43, Aravalli Hills

Faridabad – 121001

June 2025

DECLARATION

I do hereby declare that this project work entitled "**File Organizer**" submitted by me for the partial fulfillment of the requirement for the award of **MASTER OF COMPUTER APPLICATIONS** is a record of my own work. The report embodies the finding based on my study and observation and has not been submitted earlier for the award of any degree or diploma to any Institute or University.

SIGNATURE

Name: Akash Rawat

Roll No:24/SCA/MCA/056

Date: 11/07/25

CERTIFCATE FROM THE GUIDE

This is to certify that the project report entitled "File Organizer" submitted in partial fulfillment of the degree of BACHELOR OF COMPUTER APPLICATIONS to Manav Rachna International Institute of Research and Studies, Faridabad is carried out by Mr Akash Rawat Roll No 24/SCA/MCA/056 under my guidance.

Head of Department Name:

Dr Suhail Javed Quraishi

Signature From the Guide:

Dr Anupama Chadha

ACKNOWLEDGEMENT

I gratefully acknowledge for the assistance, cooperation, guidance and clarification provided by Dr Anupama Chadha during the development of Project Report. My extreme gratitude to **Dr. Raj Kumar, Associate Professor & TPO** who guided us throughout the project. Without his willing disposition, spirit accommodation, frankness, timely clarification and above all faith in us, this project could not have been completed in due time. His readiness to discuss all important matters at work deserves special attention of.

I would like to extend my sincere gratitude to **Prof. (Dr.) Suhail Javed Quraishi – HOD, Prof. (Dr.) Rashmi Agrawal – Associate Dean and Prof. (Dr.) Brijesh Kumar – Dean** for their valuable teachings and advice. I want to thank all the department faculty members for their cooperation and support. I want to thank non-teaching staff of the department for their cooperation and support.

This opportunity is a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, to attain desired career objectives. I hope to continue cooperation with all of you in the future

INDEX

INTRODUCTION

1. About Organization:

a) About UniConverge Technologies(UCT) Pvt Ltd

A company established in 2013 and working in Digital Transformationdomain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies**

e.g. Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end etc.

b) **UCT IoT Platform** (uct Insight)

UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA

It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine

The IoT Academy

The IoT academy is Ed Tech Division of UCT that is running long executive certification programs in Collaboration with EICT Academy , IITK , IITR and IITG in multiple domain

Objectives of the program

- Get practical experience of working in the industry
- To solve real world problems
- To have improved job prospects
- To have improved understanding of our field and its application
- To have personal growth like better communication and problem solving

c) Based Solution

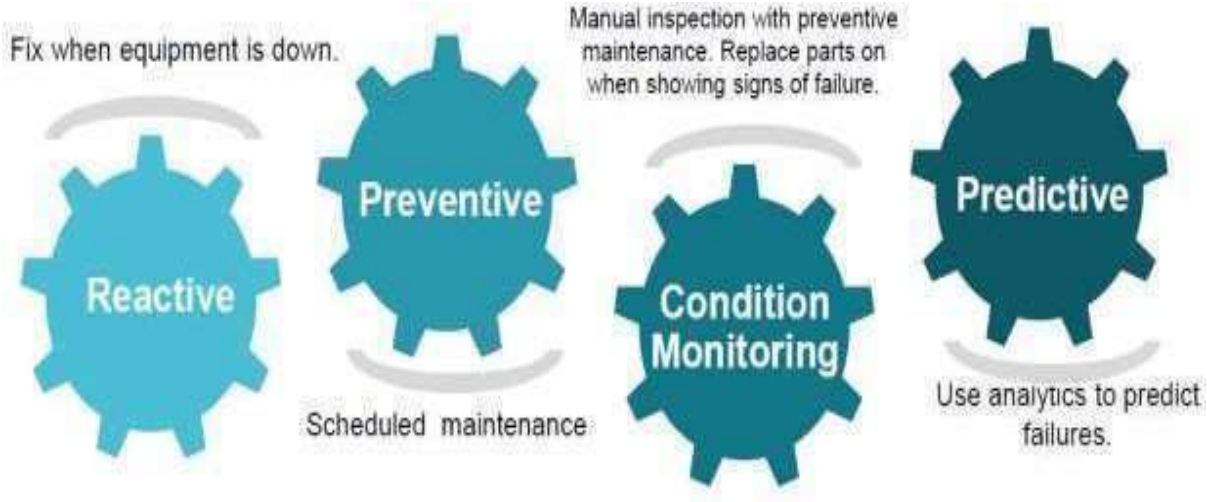
UCT is one of the early adopters of LoRAWAN technology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

i. Predictive Maintenance

UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.

It wirelessly connects devices to the internet and manages communication between end- node devices and network gateways.

LoRaWAN specification is open so anyone can set up and operate a LoRa network. LoRa is a wireless audio frequency technology that operates in a license-free radio frequency spectrum.



d) About upskill Campus (USC)

Upskill Campus is a fast-growing ed-tech platform that is meant to upskill students, freshers, working professionals, faculty, entrepreneurs, etc. We have the vision to provide an immersive experience for our learners to ensure their exhaustive growth. To provide 24x7 learning on the latest technologies that will not only help you get a better job but will also encourage you to do hands-on exercises and explore more.

Upskill campus along with The IoT Academy and in association with uniconverge technologies has facilitated the smooth execution of the complete internship process.

USC is a career development platform that delivers personalized executive coaching in a more affordable, scalable and measurable way

UpSkillCampus believe that the future is about Unified and Converged technologies as reflected in vision statement. Every aspect of life will eventually lead in bringing forth a unified world of immense possibilities.

Participated in UniConverge Technologies at Convergence Indea – IoT Expo ,Pragati Maidan, New Delhi March 27-29, 2023

upSkill CAMPUS

Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects,interaction with Industry experts, Careergrowth Services

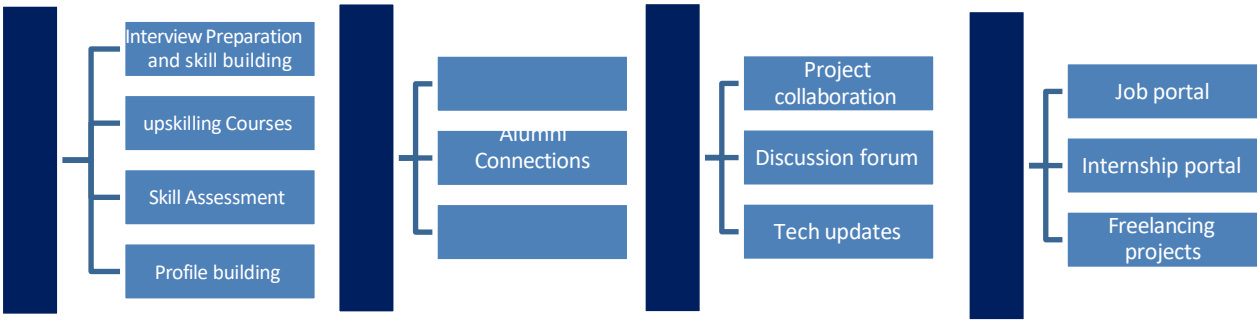
Upskil campus aiming 1 million learners in next 5 years

The IoT academy is Edtech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

f) Company Objective

“We aim to lead and support the early learning community in building the bestfoundation for creative students

“To create a platform where every student has the right to Understand and learn at their own pace.”



g) Manpower

Currently 200 People are working in the Organisation.

2. Project Objective

The File Organizer project's goal is to provide an automation solution that uses Python to help users effectively manage and arrange congested folders. This utility automatically transfers files into categories including Images, Documents, Videos,

Audio, and Archives after scanning a user-selected folder and identifying file kinds based on their extensions. Saving time and minimizing manual labor are the goals, particularly when managing high volumes of mixed files. Additionally, the project uses tkinter to create a graphical user interface (GUI) that makes the tool easy to use and accessible to non-technical people. Other features include an undo tool that improves usability and safety by returning files to their original locations using a JSON log file.To consistently execute file operations, the program makes use of common Python modules like os, shutil, and json. This project shows how Python may be used practically to automate the solution of real-world issues. Additionally, it seeks to create maintainable, modular code that may be added to with new features in the future. The File Organizer's overall goals are to increase productivity, organize digital files better, and lay the groundwork for more sophisticated file management solutions.

2.1 Problem Statement:

Users frequently gather a lot of data in one directory in today's digital world, including compressed archives, papers, audio files, videos, and photos. This eventually results in cluttered folders that are challenging to handle and traverse. In addition to being time-consuming, manually sorting these files is repetitive and prone to errors, particularly when dealing with hundreds of files or a variety of file kinds. In addition to having a detrimental effect on productivity, this disarray raises the possibility of misplacing or losing crucial files.

An automated system that can classify and arrange files according to their type, extension, or purpose is obviously needed. It should also have the option to reverse modifications if necessary. Digital file management would be streamlined by a tool that scans a user-selected directory, determines the kind of each item, and arranges them into certain categories (such as "Images," "Documents," and "Videos"). By providing a Python-powered, graphical user interface (GUI)-based file organizer that is easy to use and performs well, this project seeks to solve that issue

2.2 Programming Language

Python, a flexible, high-level programming language renowned for its ease of use and readability, was used to carry out the job.

Python was selected because of its broad support for file management, cross-platform interoperability, and automation.

It has a comprehensive collection of built-in libraries, including as os, shutil, and json, which were crucial for handling data and communicating with the file system.

A graphical user interface (GUI) was made possible using the tkinter library, making it simple for non-technical users to interact with the application.

It has a comprehensive collection of built-in libraries, including as os, shutil, and json, which were crucial for handling data and communicating with the file system.

A graphical user interface (GUI) was made possible using the tkinter library, making it simple for non-technical users to interact with the application.

System Study

a) Existing System along with limitations

Previously Users preferred to manage and arrange their files manually prior to the File Organizer project's implementation. Users had to make folders, recognize each type of file, and move files in accordance with those directories in order to complete tasks like classifying papers, photos, videos, and other file types. Especially when working with high numbers of mixed file formats, this method frequently resulted in congested directories, missing or misplaced data, and significant time waste.

The process was ineffective and prone to human error because the current approach lacked any kind of automation, real-time sorting, or error management. Because there were no useful tools or notifications, misclassification and redundancy were rampant, and non-technical users frequently struggled to continuously maintain an orderly file system.

Limitation

- Users without technical expertise found it challenging to manage files effectively due to the manual process's absence of an intuitive user interface.

- Repetitive and time-consuming procedures resulted from the lack of automation in file classification and movement.

- It was challenging to manage large files or switch to custom file rules when user requirements increased due to the system's lack of scalability and flexibility.

- Unintentional file movements could not be undone or tracked down because there was no logging or undo functionality.

- Users were unable to manage their files consistently across several computers due to the lack of a cross-platform solution.

- Unexpected failures resulted from the lack of protections against mistakes such permission problems or incompatible file formats.

b) Proposed System along with advantages

During the internship, a Python-based File Organizer System was created to get around the drawbacks of the manual file management procedure. By scanning a chosen directory, recognizing file kinds based on extensions, and transferring them into categorized folders like Images, Documents, Videos, Audio, and Archives, this system automates the process of organizing files. Even non-technical people may use the application thanks to its user-friendly graphical user interface, which makes use of tkinter.

Improved features including file movement recording with the json module and an undo capability that lets users transfer files back to their original locations if necessary are also included in the system. The program is platform-independent, lightweight, and efficient, having been built with standard Python modules such as os, shutil, and json. The program's modular design facilitates future growth and simple maintenance.

Advantages:

☑ Automation of file categorization and movement, saving time and effort.

☑ Graphical user interface enables smooth interaction, even for non-programmers.

☑ Undo feature increases safety by allowing users to reverse file movements.

☑ File movement logs ensure transparency and traceability.

☑ Cross-platform compatibility due to Python’s standard libraries.

☑ Modular and scalable design allows for future feature enhancements.

Feasibility Study

a) Technical Feasibility

The File Organizer project is technically feasible as it is developed using widely available and well-supported technologies. Python is a cross-platform and open-source programming language that works with the majority of contemporary operating systems, such as Windows, macOS, and Linux. Standard Python libraries, like os for file system operations, shutil for file movement, json for data logging, and tkinter for graphical user interface creation, are necessary for the fundamental functionality and are pre-installed in Python.

The program runs well on systems with simple hardware setups and doesn't require a lot of system resources. It is scalable and maintainable due to its modular code structure, which makes upgrades and improvements simple. Furthermore, there is no need for external servers, databases, or APIs, which streamlines deployment and lowers technical obstacles.

The project is theoretically feasible for both academic and practical application due to the ease of use of the tools and the clarity of implementation, making it completely doable with only rudimentary to basic Python programming expertise.

b) Behavioural Feasibility

The File Organizer project is behaviorally feasible as it addresses a real and relatable problem faced by many users—managing disorganized files in their systems. application doesn't require any prior technical knowledge to utilize. Users can choose a folder and arrange their files with a single click thanks to the tool's straightforward tkinter-built GUI, which makes it usable and acceptable by a variety of users.

Features like file movement logs and undo functionality increase usability and trust by giving users a sense of control over the program's modifications. Users responded well to its ease of use, clarity, and simplicity during testing and demonstration, suggesting a high degree of user adoption. By saving time, lowering manual labour, and improving file system organization, the project provides genuine value and fits in nicely with user behaviour patterns.

Overall, the system's problem-solving capabilities, safety features, and convenience of use guarantee that users will not only be eager to utilize it on a daily basis, but are also likely to do so.

c) Economic Feasibility

The File Organizer project is very cost-effective because it requires little to no funding for both development and implementation. Python, a free and open-source programming language, is used throughout its development. All of the project's libraries—os, shutil, json, and tkinter—are included in the Python standard library and don't need to be licensed or purchased separately.

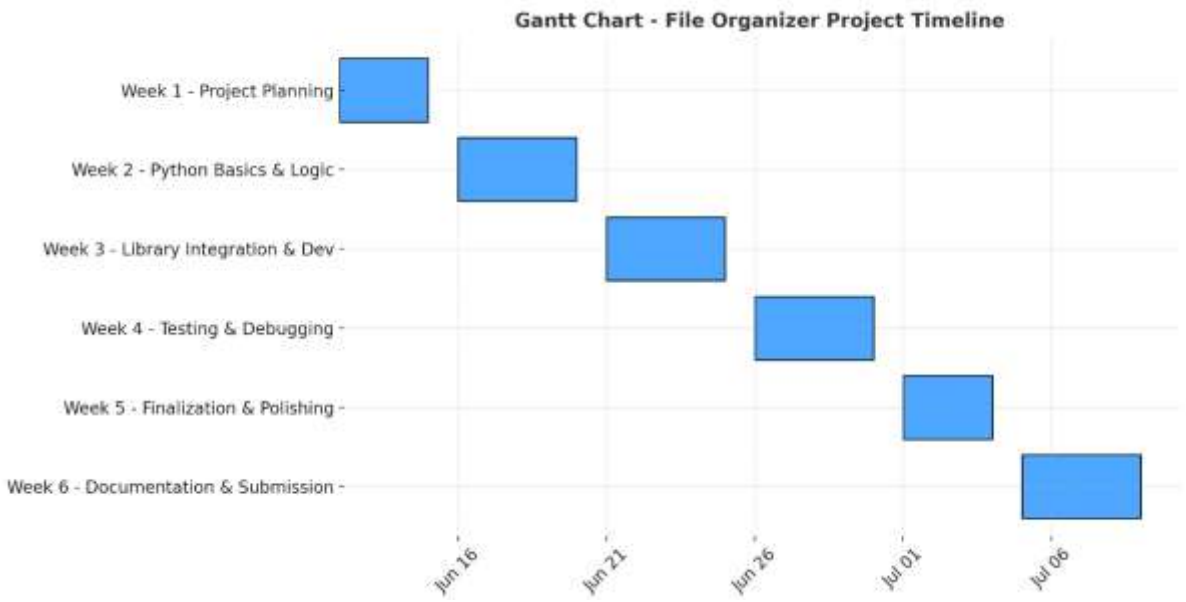
Since the application operates locally on the user's computer, hosting is free. Furthermore, it requires very little hardware, thus it can be used on laptops and regular personal computers without requiring any changes.

The modular codebase guarantees that bug fixes or enhancements may be applied with little expense or effort from a maintenance standpoint. The instrument offers indirect economic benefits in the form of time savings since it increases productivity by automating a laborious and manual operation.

In conclusion, the solution offers a high return on investment in terms of utility and efficiency, is reasonably priced, and doesn't require any external tools or equipment.

Project Monitoring System

a) Gantt Chart



The Gantt chart illustrates the six-week development timeline of the File Organizer project, spanning from June 12 to July 9. Each week focused on a specific milestone:

Week 1: Project planning and tool selection

Week 2: Learning Python basics and applying logic for file handling

Week 3: Integrating libraries (os, shutil, json, tkinter) and building core functionality

Week 4: Testing with live data, debugging, and refining the undo feature

Week 5: Finalizing UI, improving user interaction, and polishing the application

Week 6: Creating documentation, adding visuals, and submitting the completed project

System Analysis

a) Requirement Specification

The File Organizer project was developed as a desktop application using Python to automate file management tasks. Below are the detailed functional and non-functional requirements:

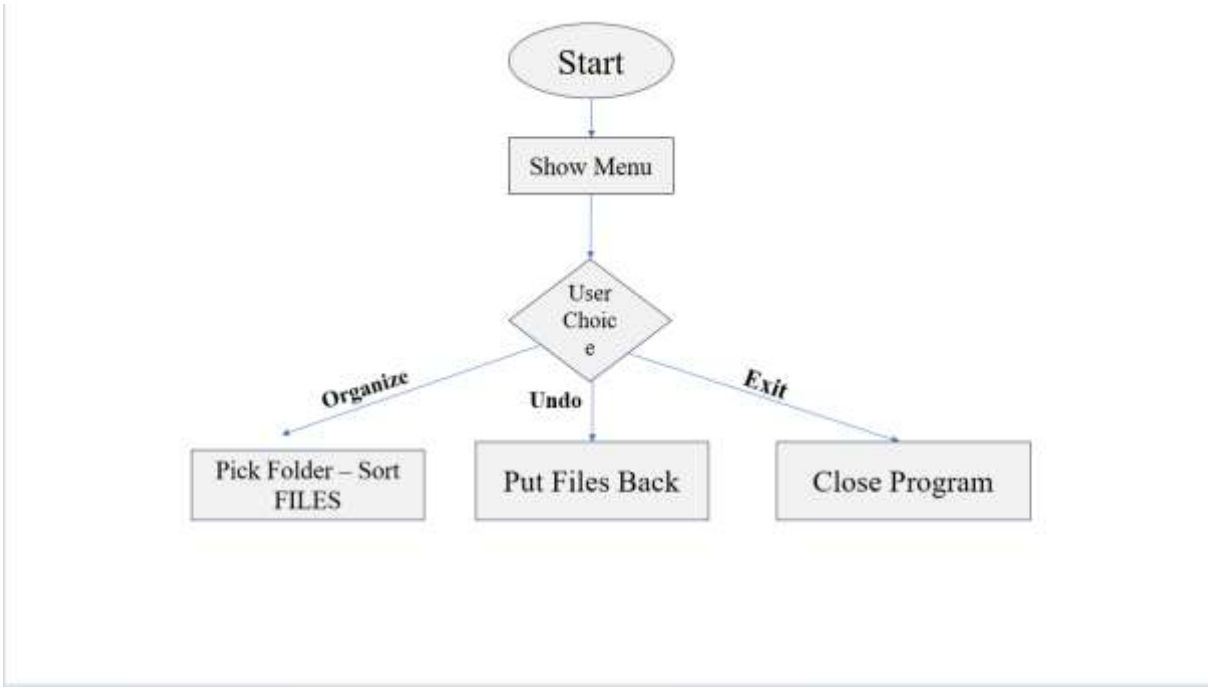
1. Functional Requirements:

- The system shall allow users to select a target folder using a GUI.
- It shall scan all files in the selected directory.
- It shall categorize files based on their extensions into predefined folders like Images, Documents, Videos, Audio, and Archives.
- It shall create folders dynamically if they do not already exist.
- It shall move files into their respective category folders.

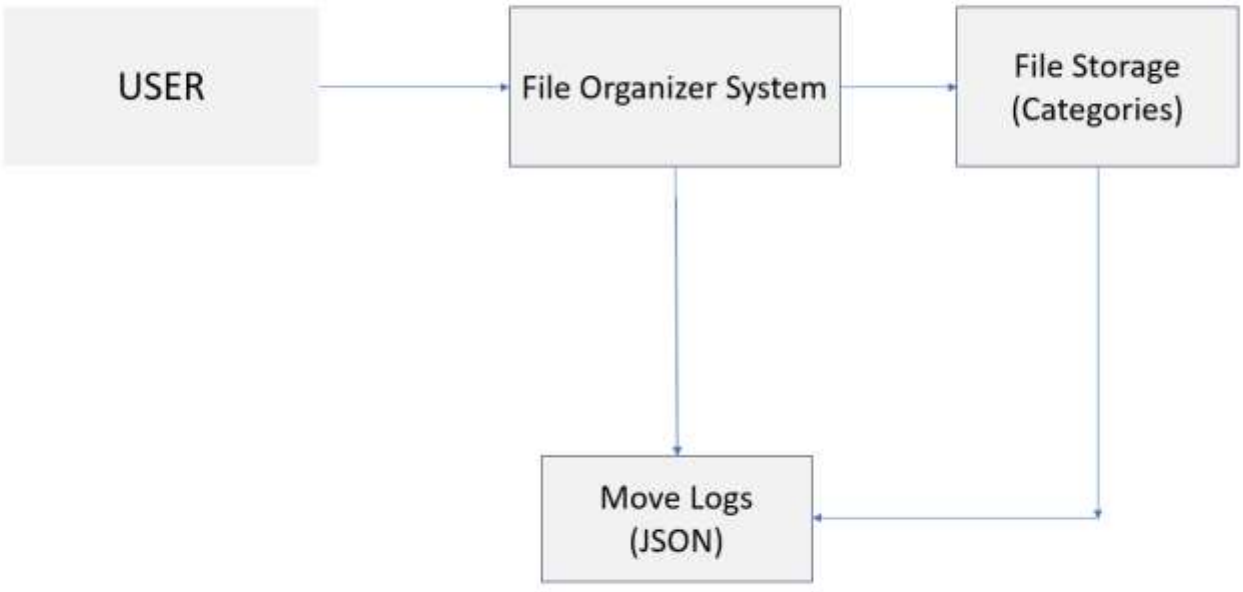
2. Non-Functional Requirements:

- The system shall be user-friendly and accessible through a simple graphical interface using tkinter.
- The software shall execute on any machine with Python installed (cross-platform).
- It shall have low memory usage and not impact system performance.
- The application should handle exceptions like permission errors, invalid paths, or unsupported formats gracefully.

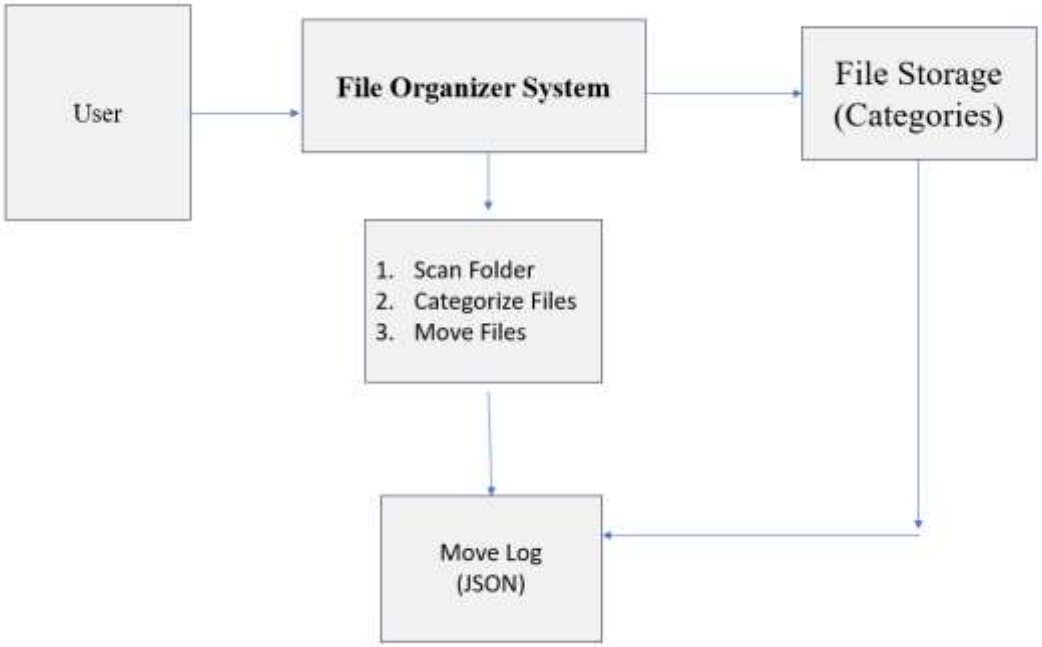
b) System Flowcharts


Start
Show Menu
User Choic e
Organize
Undo
Exit
Pick Folder – Sort FILES
Put Files Back
Close Program

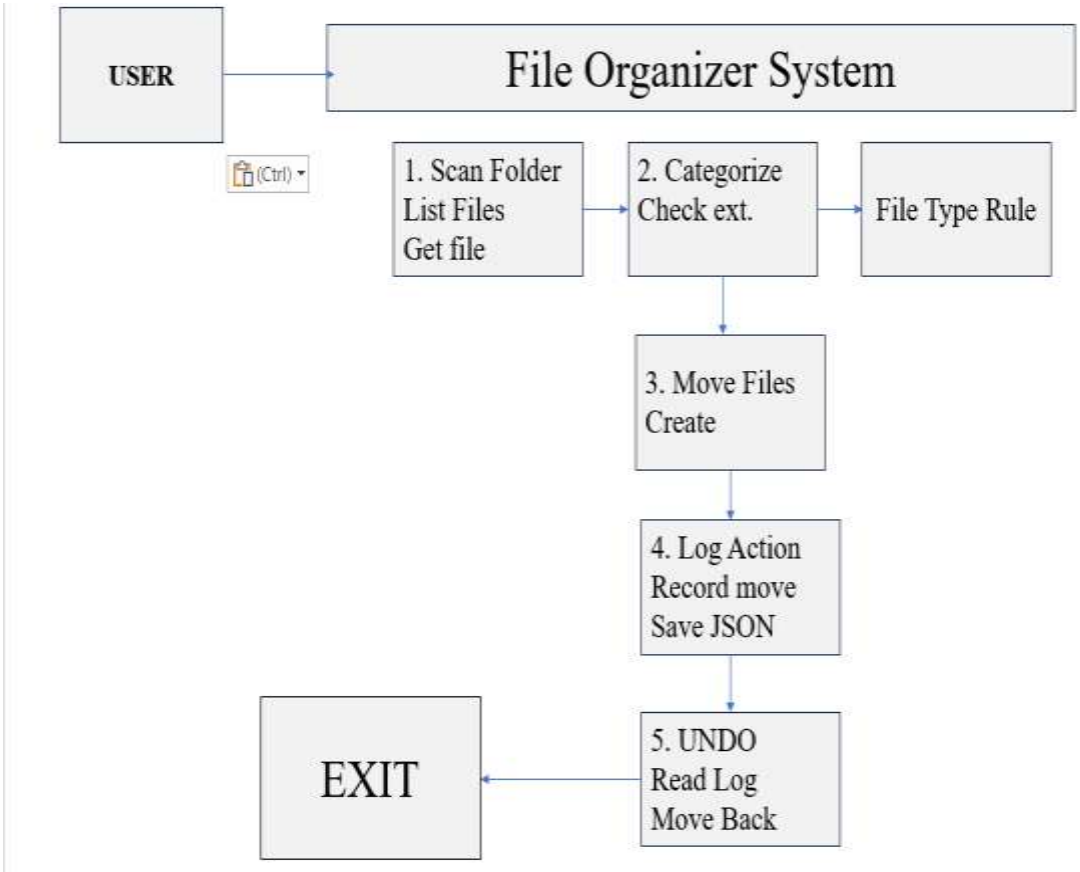
c) DFD (Level 0)


USER
File Organizer System
File Storage (Categories)
Move Logs (JSON)

Level 1 DFD



Level 2 DFD



System Design

The File Organizer system uses a simple yet effective data design structure that supports its automation and undo functionality. Instead of relying on databases, the system uses the file system and a JSON file for data handling, which is sufficient for the scale and scope of this desktop automation tool.

The core data element in this system is the move_log.json file. This log records every file that was moved during an organization session. Each entry in this JSON file contains:

The original file path ("from")

The new file path after movement ("to")

Optional metadata can be added later (e.g., timestamp)

This log-based design supports:

Undo functionality: allowing users to reverse the organization operation

Session persistence: making the tool useful beyond a single-use interaction

The system maps file extensions to categories like "Images," "Documents," "Videos," and so on using an internal dictionary-based structure called FILE_TYPES. The system is versatile because this dictionary serves as the main categorization logic and may be expanded.

Since everything is file-driven, no external databases are needed. The system remains lightweight and deployable thanks to this design decision.

Example Log file:

[{ "from": "C:/Users/Akash/Downloads/file1.jpg",

"to": "C:/Users/Akash/Downloads/Images/file1.jpg"

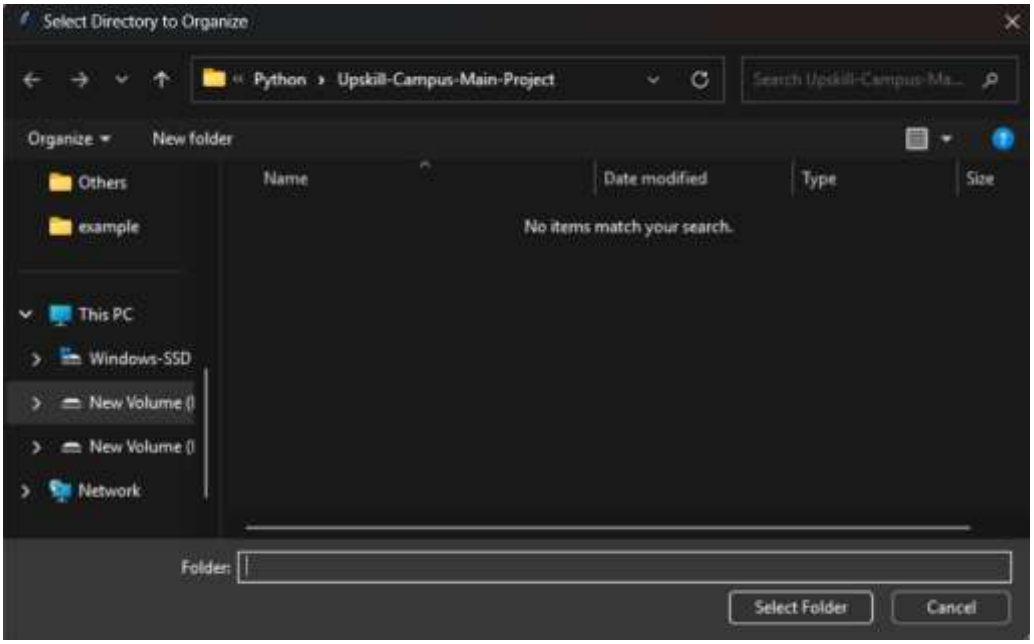
}, ...]

Input / Output Form Design

a) Screen Design

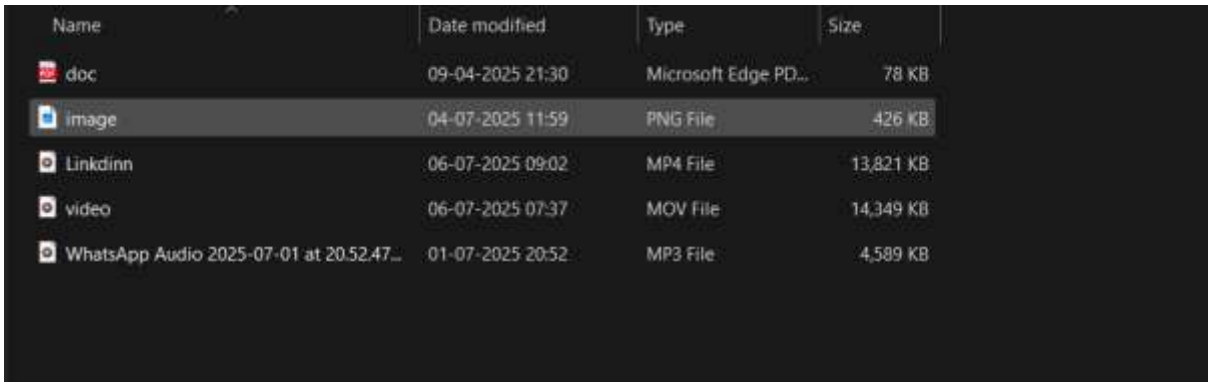
1. Selecting GUI

2. Selecting Folder

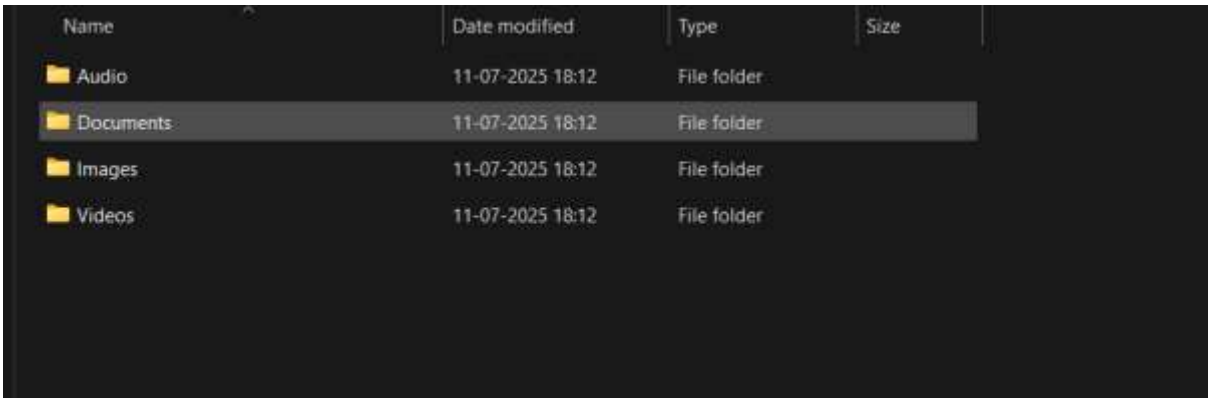


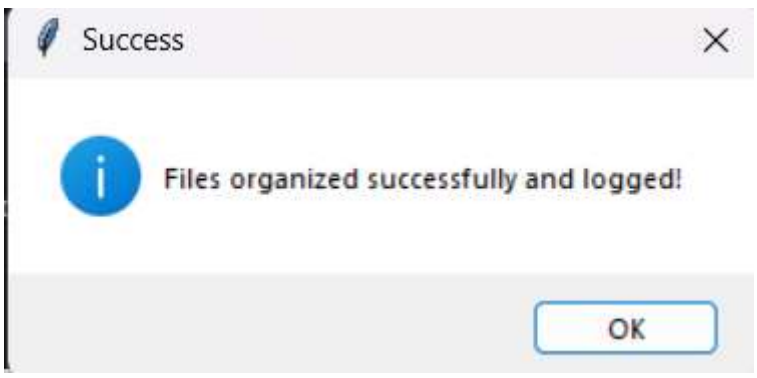
3.

4. **Folder With Different File Type**

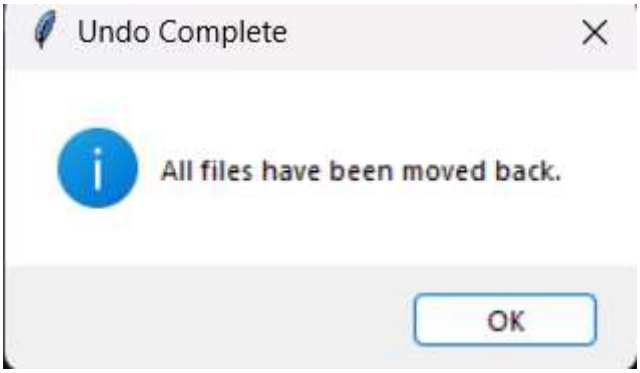


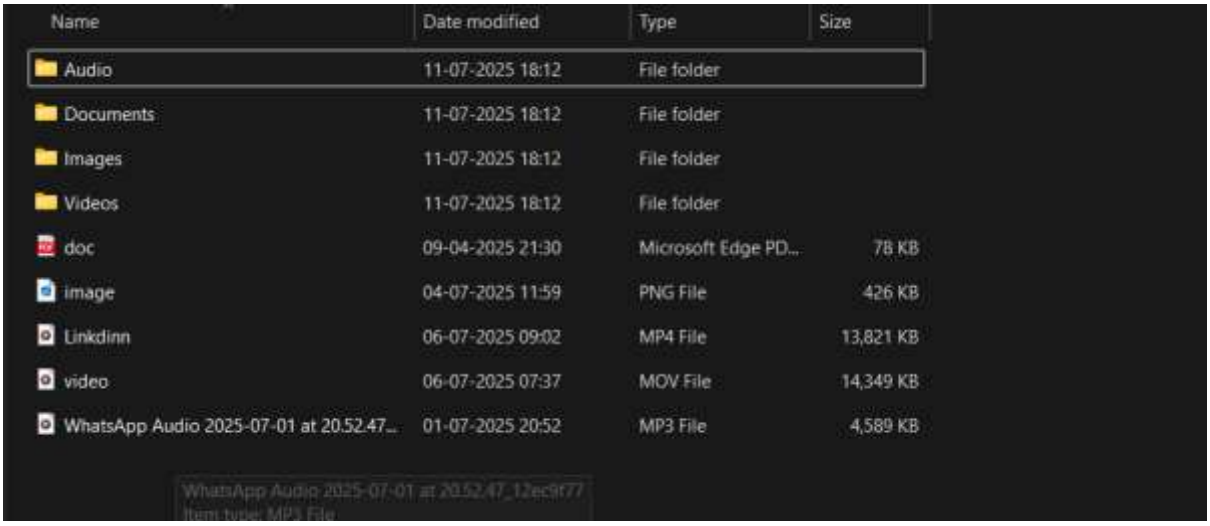
5. **After Organized**





6. UNDO Feature





7. EXIT

```
Exited
```

b) Report Design

The File Organizer project's report design is centred on how the system clearly and systematically conveys operational results to the user. The application keeps a log file (move_log.json) as a permanent record of all file movements and offers real-time feedback through pop-up dialogues rather than creating intricate printable reports.

The tkinter.messagebox module is used to construct GUI-based message boxes, which serve as the main reporting method. The system shows a confirmation message like "Files organized successfully and logged!" once the organization process is finished. This eliminates the need for the user to manually verify the file system and provides instant indication that the task has been done.

The system also has built-in error and warning alerts. When the user doesn't choose a directory, tries to undo without a proper log file, or encounters file access errors, these are displayed. By outlining exactly what went wrong and what the user has to fix, these alerts enhance the user experience.

The move_log.json file, which serves as a digital report, is another crucial component of the report design. It records the original path and new destination of each file that was moved during the session. Because the system may reverse the modifications by reading and analyzing this log, this JSON-based report is essential for implementing the undo functionality.

The following features could be added to the reporting system in later versions:

- Reports that provide a summary (e.g., files per category, total files moved)
- Reportlab and other libraries allow you to export text or PDF reports.
- logs for auditing reasons that include timestamps and file metadata

Proposed Design/ Model

1. Start (User Interaction): The process begins with the user launching the application. A simple GUI built using tkinter allows the user to select a target directory through a folder selection dialog.

2. Intermediate Steps (Processing & Logic): The application searches through every file in the directory after it has been chosen. It classifies files by reading their extensions and applying a pre established mapping (.jpg = Image, for example). If there are no equivalent folders (such as "Documents" or "Videos"), the application uses os.makedirs() to create them dynamically.

3. File Movement & Logging: Using the shutil library, the application moves each file to its appropriate folder. Simultaneously, a json file logs the original and new paths of all moved files to enable undo functionality.

4. Final Outcome (Completion & Undo Option): Once the process completes, the user receives a success message via the GUI. If needed, the user can choose to undo the organization, which reverses all file movements based on the log data. This step-by-step design ensures that the application is easy to use, efficient, and offers both control and safety to the user during file management tasks.

System Testing

a) Preparation of Test Data

Carefully crafted test data was created to make sure the File Organizer system operates as intended in a variety of situations. Simulating a real-world setting with a variety of file kinds, sizes, and naming patterns was the aim of this test data. This made it easier to confirm that the organization logic and file classification were accurate, flexible, and robust.

A single test folder contained a number of files that made up the test data. Among these files were:

a. File types: .jpg, .jpeg, .png, and .gif

b. File Types: PDF, DocX, TXT, and PPT

c. Audio files: mp3 and .wav format

d. Videos: .mp4 and .mkv formats

e. File formats: .zip, .rar

f. Files having unusual extensions or no extensions at all (such as.xyz or README) are referred to as other or unknown files.

Other test scenarios were as follows:

- No-extension files (to test "Others" category)
- Duplicate-named files (to test handling or overwriting behaviour)
- Restricted access files (to test permission alerts and error handling)

Every test case was created to cover potential scenarios and make sure the software could:

- Classify file types accurately.
- Only create destination folders as required.
- Make sure to accurately record every file movement in move_log.json.
- Correctly handle undo operations for every file that is categorized.

b) Test Cases and Result

The File Organizer system was evaluated using live data to replicate real-world usage scenarios following initial verification with sample test data. This required executing the application on actual user directories, like the Desktop, Downloads folder, or other frequently used directories where different kinds of files are usually kept together.

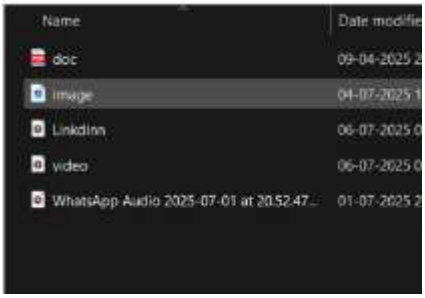
Test Scenarios:

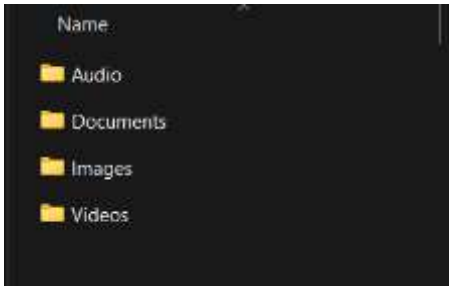
1) Directory Selection (GUI):

- **Test 1:** Choose a legitimate folder, which should permit further processing.
- **Test 2:** Selecting the Cancel folder should result in a warning and not be allowed to continue.
- **Test 3:** Choose a folder that has no files; a "no files to organize" notice ought to appear.

2) File Organization and Categorization:

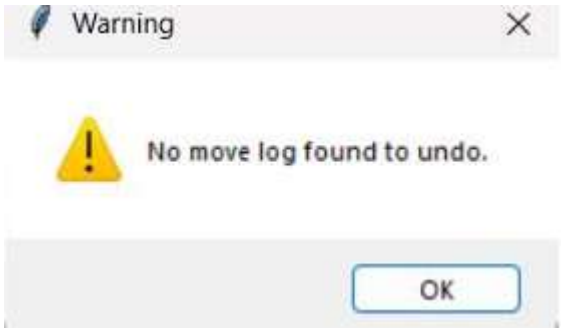
- **Test 1**: Sort mixed files; each file should be moved to the appropriate category folder.
- **Test 2**: Deal with files with unknown extensions; these belong in the "Others" folder.
- **Test 3:** Confirm folder construction; if category folders aren't already there, they should be made.

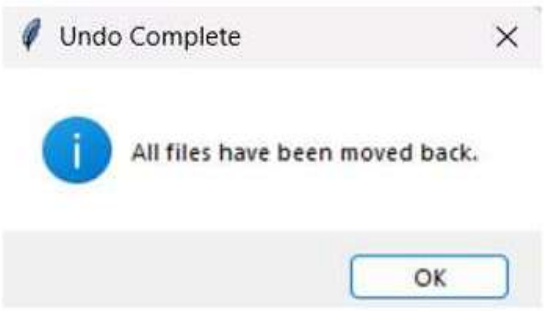




3) File Movement Logging:

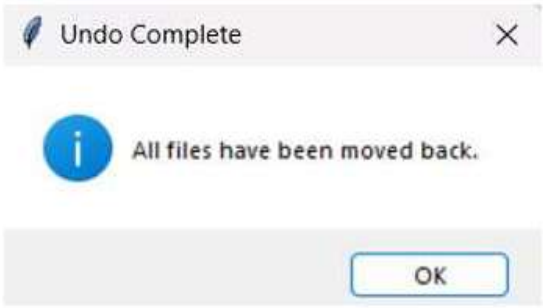
- **Test 1:** After organizing, move_log.json should be created.
- **Test 2:** Log should correctly map each file's old and new location.
- **Test 3:** Handle log overwrite — old log is replaced with each new organization session.

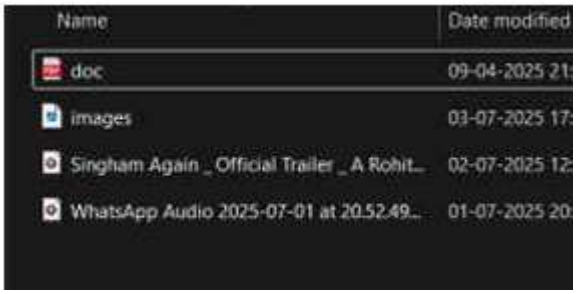




4) Undo Operation:

- **Test 1:** Use the log to trigger undo, which should return all files to their original positions.
- **Test 2:** A warning should appear and the undo should be skipped if the log is absent.
- **Test 3**: Handle partial undo failures (e.g., moved files deleted) gracefully with error message





5) GUI and Application Behavior:

- **Test 1:** The program shouldn't crash when input is incorrect or when directories are restricted.
- **Test 2:** Pressing the ext button should instantly close the window.
- **Test 3:** All information, warning, and error message boxes ought to show the proper messages.

System Implementation

Python was used to create the desktop automation program known as the File Organizer project. It does not require high-end requirements because it is made to function well on general-purpose PCs. The software and hardware needed to successfully implement and carry out the project are listed below.

Hardware specifications

- Processor: Intel i3 or higher is required at minimum; Intel i5 or higher is advised.
- RAM: 4 GB at minimum; 8 GB is advised for best results.
- Hard Drive: A minimum of 100 MB of available disc space
- Monitor: at least 1024 x 768 pixels
- Devices for input: a standard keyboard and mouse

Software prerequisites

- Operating System: Linux, macOS, or Windows 7/10/11
- Version of Python: 3.6 or later
- Python Libraries Needed:
- OS: for file path and directory operations
- Shutil is used to move and copy files.
- JSON is used to record file operations.
- For creating the graphical user interface (GUI), utilize tkinter.

Tools for Development and Execution

- IDE/Text Editor: Visual Studio Code, PyCharm, or any other IDE that supports Python

- Run the Python script tkinter in the terminal or command prompt to create the graphical user interface (GUI).

Documentation

a) User Documentation

- **Overview of the Project**

A desktop Python application called The File Organizer was created to automate the process of organizing and maintaining files in a designated directory. It uses a user-friendly GUI created with Tkinter to classify files according to their extensions (such as Images, Documents, and Videos) and move them into the appropriate directories. The project makes use of the built-in Python libraries tkinter, os, shutil, and json. It also has an undo capability that lets users use a produced move_log.json file to reverse file movements. By lowering human labour, this technology improves productivity and streamlines digital file management.

.

How to Launch the App

- Install at least Python 3.6.
- File_organizer.py can be double-clicked or executed in a terminal or command prompt using Python.

How to Use It:

- A graphical user interface will appear.
- Choose "Select Folder and Organize"
- Choose the desired directory (such as Downloads) by browsing through it.
- Files will be automatically sorted into folders (Images, Documents, etc.) by the application.
- Click "Undo Last Organization" if necessary to use move_log.json to recover files.

- Press "Exit" to end the application.

b) Technical Documentation

The File Organizer system is built using Python with a modular design that simplifies maintenance and future updates. It uses the tkinter library for the graphical interface, allowing users to select folders and trigger file organization. Internally, the system scans files using the os module, identifies their types based on extensions, and moves them to categorized folders using shutil. Each file movement is recorded in a move_log.json file using the json module, enabling an undo feature. All core functions are organized logically to ensure readability, reusability, and smooth file handling across different platforms.

1. Project Structure

- file_organizer.py is the primary Python script that may be executed and contains the GUI and application logic.
- move_log.json is a JSON file that contains a log of files that have been moved and has undo capabilities.

2. Key Python Library used:

- OS: File paths, directory checks, and file scanning from the chosen folder are handled by the used operating system.
- Shutil: In charge of transferring files to the designated destination folders.
- JSON: Used to write and read the move log file so that operations may be tracked and undone.
- Tkinter: Offers a basic graphical user interface (GUI) that enables users to choose folders, arrange files, reverse actions, and end the application.

3. **Main Functional Component**:

- Choose_and_organize() : is one of the primary functional components. It is triggered when the user chooses "Select Folder and Organize." It opens a file dialogue and provides the chosen path to the organizing function.

- The function organize_files(directory) looks through every file in the chosen directory, uses extension mapping to identify the file type, creates appropriate folders if none already exist, and then transfers files into those folders.

- get_category(extension): Uses a dictionary to match file extensions with preset categories (such as images, documents, videos, etc.).

4. **Handling Errors and User Feedback:**

   Try-except blocks and user-friendly message pop-ups using tkinter.messagebox are used to handle all errors (such as missing folders, permission problems, and no undo log discovered).

5. **Simplicity of Design**

   The application uses the native filesystem for organization and JSON for structured logging, eschewing complicated databases or other dependencies. This makes it simple to update and expand the system, for instance, by including more UI components, new categories, or report production capabilities.

Scope of the project

The goal of the File Organizer project is to automate the management and categorization of files in a directory that the user specifies. The system's main functions include scanning files, classifying them according to their file extensions, putting them in the right places, and creating folders with categories like Images, Documents, Videos, Audio, Archives, and Others. This enhances desktop and directory organization while lowering manual labour.

The tkinter library was used to create a straightforward and user-friendly graphical user interface (GUI) that enables user interaction. After the user chooses a directory, the program processes all of the files within that folder, arranges them using a pre-established mapping (kept in a Python dictionary), and creates category folders on its own if none already exist. To provide seamless file system interaction, the OS and Shutil libraries are used for each operation.

Using a structured JSON log file (move_log.json) to store the metadata of every moved files, including their original and destination paths, is a crucial component of the project's scope. This file is essential for turning on the undo function, which lets the system undo file movements at the user's request.

In the future, this project could be expanded to accommodate more features like:

- Setting up automatic cleanup on a regular basis
- Creating file statistics summary reports
- Keeping track of segmented file sizes and counts for analytics

The scope includes both practical usability and technical execution, providing a comprehensive solution for fundamental file organizing requirements with potential for future improvements. Students, office workers, and everyone else wishing to effectively automate repetitious file sorting activities will find it ideal

Bibliography

1. https://www.smartdraw.com/flowchart/examples/
2. https://www.geeksforgeeks.org/levels-in-data-flow-diagrams-dfd/
3. https://github.com/topics/ui-design
4. https://docs.python.org/3/
5. https://www.geeksforgeeks.org/search/?gq=python
6. https://www.upskillcampus.com/
7. https://www.w3schools.com/CSS/css3_user_interface.asp
8. https://github.com/topics/file-organizer
9. https://realpython.com/
10. https://www.youtube.com/watch?v=QXeEoD0pB3E&list=PLsyeobzWxl7poL9JTVyndKe62ieoN-MZ3