Automated System File Organizer

Overview:

This project involves creating a Python-based system that organizes files into folders based on their extensions. The application automatically sorts files into categories like Images, Documents, etc., making it easy to manage cluttered directories. It leverages core Python libraries such as os, shutil, and winreg, and also integrates with the Windows right-click context menu for quick execution.

Aim:

To develop a lightweight Python utility that enhances file management by automatically sorting files in a given folder into subfolders based on file extensions, improving organization and productivity.

Key Features:

- ❖ Dynamic folder path input
- ❖ Extension-based sorting (.jpg, .pdf, etc.)
- ❖ Automatic subfolder creation
- ❖ Windows context menu integration
- ❖ Efficient background execution
- ❖ Robust error handling and feedback

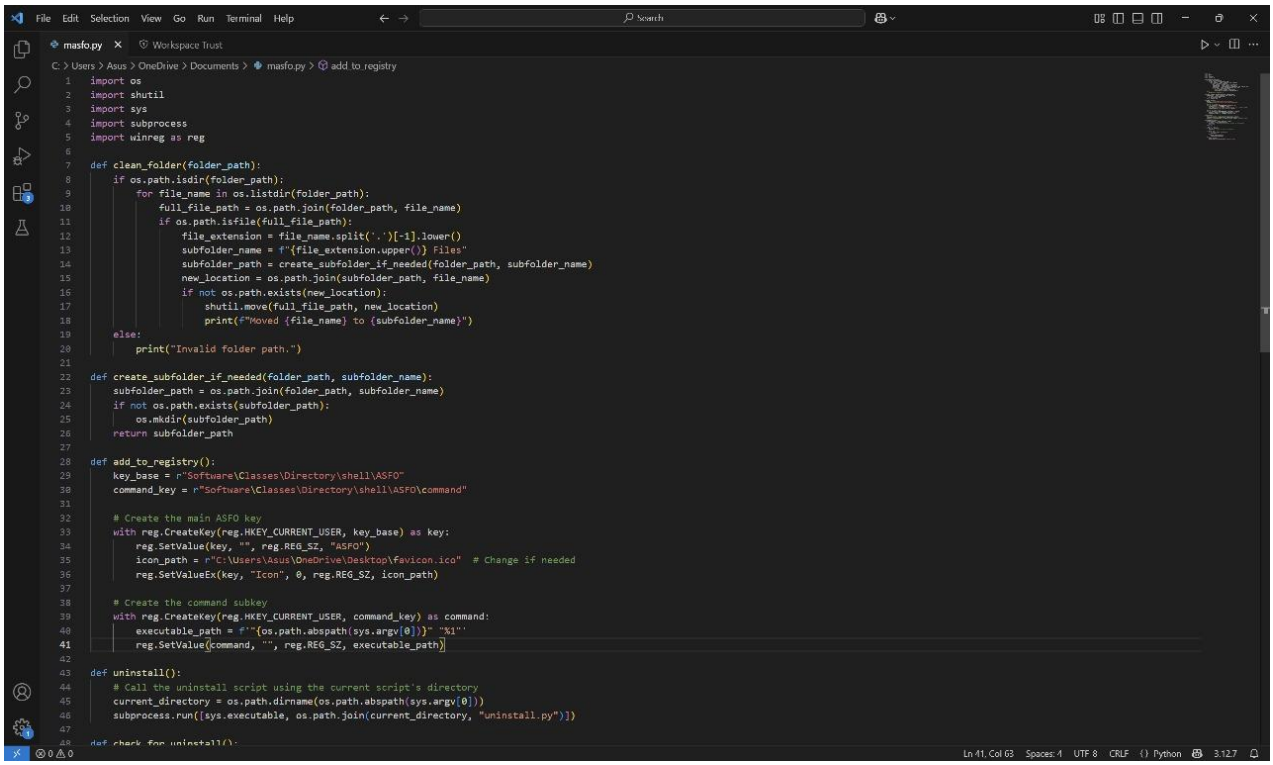
Purpose:

The purpose of the Automated System File Organizer (ASFO) is rooted in solving a common and persistent problem faced by computer users: digital clutter. As files accumulate over time—whether from downloads, project work, media storage, or documentation—directories become chaotic and unmanageable. This disorganization can result in decreased productivity, difficulty locating important files, and even accidental data loss.

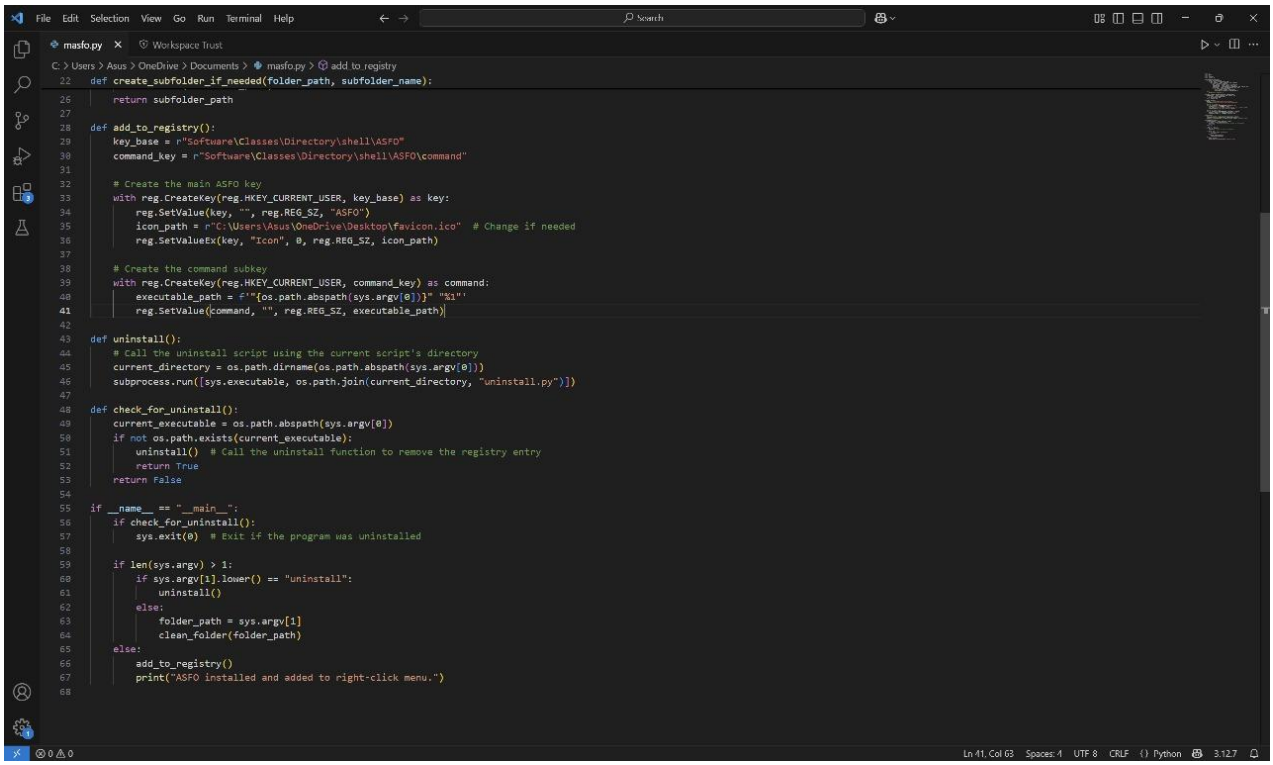
BEFORE :

AFTER :

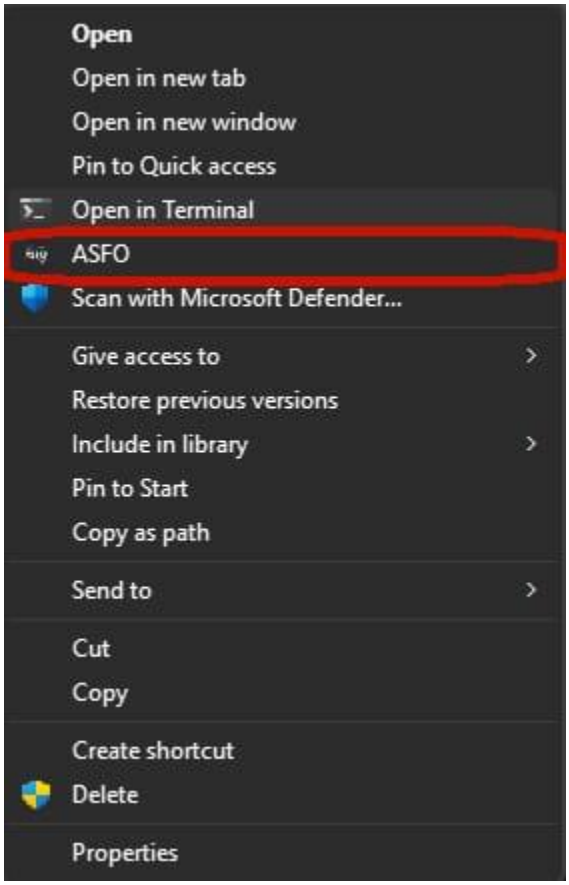
Procedure for Developing :



```python
import os
import shutil
import sys
import subprocess
import winreg as reg

def clean_folder(folder_path):
    if os.path.isdir(folder_path):
        for file_name in os.listdir(folder_path):
            full_file_path = os.path.join(folder_path, file_name)
            if os.path.isfile(full_file_path):
                file_extension = file_name.split('.')[-1].lower()
                subfolder_name = f"{file_extension.upper()} Files"
                subfolder_path = create_subfolder_if_needed(folder_path, subfolder_name)
                new_location = os.path.join(subfolder_path, file_name)
                if not os.path.exists(new_location):
                    shutil.move(full_file_path, new_location)
                    print(f"Moved {file_name} to {subfolder_name}")
    else:
        print("Invalid folder path.")

def create_subfolder_if_needed(folder_path, subfolder_name):
    subfolder_path = os.path.join(folder_path, subfolder_name)
    if not os.path.exists(subfolder_path):
        os.mkdir(subfolder_path)
    return subfolder_path

def add_to_registry():
    key_base = r"Software\Classes\Directory\shell\ASFO"
    command_key = r"Software\Classes\Directory\shell\ASFO\command"

    # Create the main ASFO key
    with reg.CreateKey(reg.HKEY_CURRENT_USER, key_base) as key:
        reg.SetValue(key, "", reg.REG_SZ, "ASFO")
        icon_path = r"C:\Users\Asus\OneDrive\Desktop\favicon.ico"  # Change if needed
        reg.SetValueEx(key, "Icon", 0, reg.REG_SZ, icon_path)

    # Create the command subkey
    with reg.CreateKey(reg.HKEY_CURRENT_USER, command_key) as command:
        executable_path = f'"{os.path.abspath(sys.argv[0])}" "%1"'
        reg.SetValue(command, "", reg.REG_SZ, executable_path)

def uninstall():
    # Call the uninstall script using the current script's directory
    current_directory = os.path.dirname(os.path.abspath(sys.argv[0]))
    subprocess.run([sys.executable, os.path.join(current_directory, "uninstall.py")])
```



```python
def create_subfolder_if_needed(folder_path, subfolder_name):
    return subfolder_path

def add_to_registry():
    key_base = r"Software\Classes\Directory\shell\ASFO"
    command_key = r"Software\Classes\Directory\shell\ASFO\command"

    # Create the main ASFO key
    with reg.CreateKey(reg.HKEY_CURRENT_USER, key_base) as key:
        reg.SetValue(key, "", reg.REG_SZ, "ASFO")
        icon_path = r"C:\Users\Asus\OneDrive\Desktop\favicon.ico"  # Change if needed
        reg.SetValueEx(key, "Icon", 0, reg.REG_SZ, icon_path)

    # Create the command subkey
    with reg.CreateKey(reg.HKEY_CURRENT_USER, command_key) as command:
        executable_path = f'"{os.path.abspath(sys.argv[0])}" "%1"'
        reg.SetValue(command, "", reg.REG_SZ, executable_path)

def uninstall():
    # Call the uninstall script using the current script's directory
    current_directory = os.path.dirname(os.path.abspath(sys.argv[0]))
    subprocess.run([sys.executable, os.path.join(current_directory, "uninstall.py")])

def check_for_uninstall():
    current_executable = os.path.abspath(sys.argv[0])
    if not os.path.exists(current_executable):
        uninstall()  # Call the uninstall function to remove the registry entry
        return True
    return False

if __name__ == "__main__":
    if check_for_uninstall():
        sys.exit(0)  # Exit if the program was uninstalled

    if len(sys.argv) > 1:
        if sys.argv[1].lower() == "uninstall":
            uninstall()
        else:
            folder_path = sys.argv[1]
            clean_folder(folder_path)
    else:
        add_to_registry()
        print("ASFO installed and added to right-click menu.")
```



1. Planning :

The first step was identifying user needs—specifically the challenge of managing cluttered folders. Sorting criteria were based on file extensions, which allowed reliable categorization.

2. Interface Design :

To keep the tool lightweight and easy to use, no GUI was built. Instead, the application integrates with the Windows right-click context menu. This allows users to trigger file organization with just a single click on any folder.

3. Interface Design :

Using Python's os and shutil libraries, the program scans the selected directory, creates subfolders based on file types (e.g., PDFs, Images), and moves each file into its corresponding folder automatically

Actual Implementation on Windows Desktop:

- Python script created with robust sorting logic
- Added to Windows context menu for one-click file organization
- Verified on folders with mixed file types
- Successfully handles empty folders and repeats without crashes

Conclusion:

The ASFO project successfully delivers a working utility that automates file organization on Windows systems. Its simplicity, reliability, and context menu integration make it a valuable tool for users looking to keep their digital space tidy. Future enhancements can include GUI addition and user-defined sorting rules.