A Project report on

FILE MANAGEMENT SYSTEM

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

in

Computer Science and Engineering

Submitted by

K. Dhathri Guptha-20H51A05J9

B. Manichandana-20H51A05N0

M. Laya -20H51A05P4

Under the esteemed guidance of
(Dr.V. Venkataiah)
(Associate Professor
& Additional Controller
Of Examinations)

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING& TECHNOLOGY

(UGC Autonomous)

*Approved by AICTE *Affiliated to JNTUH *NAAC Accredited with A^+ Grade

KANDLAKOYA , MEDCHAL ROAD, HYDERABAD - 501401.

2020- 2024

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini Project II report entitled **"File Management System"** being submitted by K.Dhathri Gupta-20H51A05J9 B.Manichandana-20H51A05N0 M.Laya-20H51A05P4 in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

Dr.V. Venkataiah
Associate Professor & Additional Controller Of Examinations.
Dept. of CSE

Dr. Siva Skandha Sanagala
Associate Professor and HOD
Dept. of CSE

Acknowledgement

With great pleasure I want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

I am grateful to **Dr.V. Venkataiah, Associate Professor & Additional Controller of Examinations**, Dept of Computer Science and Engineering for his valuable suggestions and guidance during the execution of this project work.

I would like to thank **Dr.S. Shiva Skanda**, Head of the Department of Computer Science and Engineering, for his moral support throughout the period of my study in CMRCET.

I am highly indebted to **Dr. V A Narayana**, Principal CMRCET for giving permission to carry out this project in a successful and fruitful way.

I would like to thank the Teaching & Non- teaching staff of Department of Computer Science and Engineering for their co-operation

Finally, I express my sincere thanks to **Mr. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, for his continuous care. I sincerely acknowledge and thank all those who gave support directly and indirectly incompletion of this project work.

K. Dhathri Gupta -20H51A05J9
B. Manichandana-20H51A05N0
M.Laya -20H51A05P4

TABLE OF CONTENTS

ABSTRACT

This project presents a file management system that provides an efficient and user-friendly solution for managing files and directories. A file manager (destiny) is a computer program that provides a user interface to manage files and folders of different formats (e.g.: - pdf's, doc's, jpg files etc). The most common operations performed on files or groups of files includes opening (e.g., viewing, playing, editing or printing), renaming, moving or copying, deleting and searching for files. Folders and files may be displayed in a hierarchical tree based on their directory structure. Additionally, the system should be customizable, allowing users to add extra features and functionalities as needed.With the ability to sort and merge files, users can better organize their data, making it easier to find what they need. Some data structures that will be used in the program are linked list, queue and tree. It is a program encrypted in tkinter python language and will also include searching and sorting's.

CHAPTER1
INTRODUCTION

OVERVIEW:

The File Management System is designed to be flexible, efficient, and user-friendly, and it is intended to make file management tasks easier and more convenient for users.
The system can be easily customized and extended to meet the changing needs of the users, and it provides a powerful tool for managing files and directories.

APPLICATIONS:

The File Management System is a software application that provides users with a simple and efficient way to manage their files and directories.
The system is built using Python, and it integrates the "os" and "shutil" modules to provide a wide range of file management functionalities.
The system also provides a graphical user interface (GUI) built using Tkinter, which makes it easy for users to interact with the system and perform common file management tasks.

Some of the key features of the File Management System include:
File Sorting: The system allows users to sort their files and directories in a variety of ways, including by name, size, type, and date modified.

File Merging: The system allows users to merge multiple files into a single file, for example, merge all text files in a folder into one file.

PROBLEM STATEMENT:

- A file manager (destiny) is a computer program that provides a user interface to manage files and folders of different formats (E.g.: - pdfs, doc's, jpg files etc..). The most common operations performed on files or groups of files includes opening (E.g., viewing, playing, editing or printing), renaming, moving or copying, deleting and searching for files.
- Given a folder with multiple files in different formats (text, image, video, etc.), create a system that allows users to easily sort, merge, improving their productivity. The system should have a graphical user interface (GUI) that makes it easy for users to perform these tasks, reducing the need for technical expertise. Additionally, the system should be customizable, allowing users to add extra features and functionalities as need.

SCOPE OF PROJECT:

The organization of the project can be depicted using a high-level architecture diagram, which provides a visual representation of the various components and their relationships. This can help stakeholders to understand the overall structure and design of the project and to identify any potential areas for improvement.

In conclusion, the organization of the File Management System project is a critical factor in its success, as it affects the efficiency, maintainability, and scalability of the system. A well-organized project can help to ensure that the system meets the needs of the users and provides a powerful tool for managing files and directories.

CHAPTER2
BACKGROUNDWORK

BACKGROUND WORK

PYTHON: high-level, general-purpose, and very popular programming language. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in Software Industry.

Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard libraries which can be used for the following:

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, Beautiful Soup, Selenium)
- Test frameworks
- Multimedia
- Scientific computing
- Text processing and many more.

Easy to Code

Python is a very developer-friendly language which means that anyone and everyone can learn to code it in a couple of hours or days. As compared to other object-oriented programming languages like Java, C, C++, and C#, Python is one of the easiest to learn.

Open Source and Free

Python is an open-source programming language which means that anyone can create and contribute to its development. Python has an online forum where thousands of coders gather daily to improve this language further. Along with this Python is free to download and use in any operating system, be it Windows, Mac or Linux.

Support for GUI

GUI or Graphical User Interface is one of the key aspects of any programming language because it has the ability to add flair to code and make the results more visual. Python has support for a wide array of GUIs which can easily be imported to the interpreter, thus making this one of the most favourite languages for developers.

Object-Oriented Approach

One of the key aspects of Python is its object-oriented approach. This basically means that Python recognizes the concept of class and object encapsulation thus allowing programs to be efficient in the long run.

High-Level Language

Python has been designed to be a high-level programming language, which means that when you code in Python you don't need to be aware of the coding structure, architecture as well as memory management.

Integrated by Nature

Python is an integrated language by nature. This means that the python interpreter executes codes one line at a time. Unlike other object-oriented programming languages, we don't need to compile Python code thus making the debugging process much easier and efficient. Another advantage of this is, that upon execution the Python code is immediately converted into an intermediate form also known as byte-code which makes it easier to execute and also saves runtime in the long run.

Highly Portable

Suppose you are running Python on Windows and you need to shift the same to either a Mac or a Linux system, then you can easily achieve the same in Python without having to worry about changing the code. This is not possible in other programming languages, thus making Python one of the most portable languages available in the industry.

Highly Dynamic

As mentioned in an earlier paragraph, Python is one of the most dynamic languages available in the industry today. What this basically means is that the type of a variable is decided at the run time and not in advance. Due to the presence of this feature, we do not need to specify the type of the variable during coding, thus saving time and increasing efficiency.

Extensive Array of Library

Out of the box, Python comes inbuilt with a large number of libraries that can be imported at any instance and be used in a specific program. The presence of libraries also makes sure that you don't need to write all the code yourself and can import the same from those that already exist in the libraries.

Support for Other Languages

Being coded in C, Python by default supports the execution of code written in other programming languages such as Java, C, and C#, thus making it one of the versatile in the industry.

MODULES:

Tkinter: Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. **Tkinter** is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit,

We used python language because there are useful modules such as SQL. Connector and tkinter. We used SQL. Connector module to link between SQL and python and tkinter module to show the interface and to show how the graphical password authentication works. We used SQL to store datasets of different images

Shutil:

The "shutil" module in Python is a built-in library that provides a set of high-level operations for file copying and archive management. The "shutil" module is designed to be used as a replacement for the low-level "copy" and "copy tree" functions from the "distutils. file_util" module.

Some of the key features of the "shutil" module include:

- Copying files and directories
- Archiving and compression
- File permissions and metadata

The "shutil" module is a versatile and user-friendly tool for managing files and directories, and it is widely used in many different applications, including backup and restore utilities, file management systems, and archive management tools.

In conclusion, the "shutil" module is a valuable addition to the Python standard library, providing a set of high-level functions for file management and archive management. It is easy to use, versatile, and reliable, and it is an excellent choice for developers looking to build powerful file management solutions using Python.

OS:

The "os" module in Python is a built-in library that provides a set of functions and tools for working with the operating system. The "os" module is a low-level module that provides access to the underlying operating system, allowing you to perform a wide range of tasks, including file management, process management, and environment management.

Some of the key features of the "os" module include:

- File management
- Process management
- Environment management
- Path management

The "os" module is a powerful and flexible tool for working with the operating system, and it is widely used in many different applications, including file management utilities, process management tools, and environment management solutions.

In conclusion, the "os" module is an essential part of the Python standard library, providing a set of low-level functions and tools for working with the operating system. It is flexible, reliable, and widely used, making it an excellent choice for developers looking to build powerful applications and utilities using Python.

CHAPTER3
PROPOSEDSYSTEM

PROPOSED SYSTEM

We proposed a system which can help the users to sort, merge the multiple files in different format like text, image, video files at a time .A file manager (destiny) is a computer program that provides a user interface to manage files and folders of different formats (E.g.: - pdfs, doc's, jpg files etc..). The most common operations performed on files or groups of files includes opening (E.g., viewing, playing, editing or printing), renaming, moving or copying, deleting and searching for files.

Given a folder with multiple files in different formats (text, image, video, etc.), create a system that allows users to easily sort, merge, improving their productivity. The system should have a graphical user interface (GUI) that makes it easy for users to perform these tasks, reducing the need for technical expertise. Additionally, the system should be customizable, allowing users to add extra features and functionalities as need.

CODE SNIPPETS

List of Packages imported in the Project:

```
from tkinter import *
import tkinter as tk
import tkinter.filedialog as fd
from tkinter import filedialog
import tkinter.messagebox as mb
import os
import shutil
```

functions to open and copy files:

```
def open_file():
  file = fd.askopenfilename(title='Choose a file of any type', filetypes=[("All files", "*.*")])
os.startfile(os.path.abspath(file))

def copy_file():
file_to_copy = fd.askopenfilename(title='Choose a file to copy', filetypes=[("All files",
```

"*.*")])
dir_to_copy_to = fd.askdirectory(title="In which folder to copy to?")
  try:
shutil.copy(file_to_copy, dir_to_copy_to)
mb.showinfo(title='File copied!', message='Your desired file has been copied to your desired location')
  except:
mb.showerror(title='Error!', message='We were unable to copy your file to the desired location. Please try again')

functions to rename and delete a file:

```
def delete_file():
  file = fd.askopenfilename(title='Choose a file to delete', filetypes=[("All files", "*.*")])
os.remove(os.path.abspath(file))
mb.showinfo(title='File deleted', message='Your desired file has been deleted')

def rename_file():

  file = fd.askopenfilename(title='Choose a file to rename', filetypes=[("All files", "*.*")])
rename_wn = Toplevel(root)
rename_wn.title("Rename the file to")
rename_wn.geometry("250x70"); rename_wn.resizable(0, 0)
Label(rename_wn, text='What should be the new name of the file?', font=("Times New Roman", 10)).place(x=0, y=0)
new_name = Entry(rename_wn, width=40, font=("Times New Roman", 10))
new_name.place(x=0, y=30)   #dialogue box
new_file_name = os.path.join(os.path.dirname(file),

new_name.get()+os.path.splitext(file)[1])
os.rename(file, new_file_name)
mb.showinfo(title="File Renamed", message='Your desired file has been renamed')
```

functions to open, rename, delete folders:

```
def open_folder():
  folder = fd.askdirectory(title="Select Folder to open")
os.startfile(folder)

def delete_folder():
folder_to_delete = fd.askdirectory(title='Choose a folder to delete')
os.rmdir(folder_to_delete)
mb.showinfo("Folder Deleted", "Your desired
folder has been deleted")
def move_folder():
folder_to_move = fd.askdirectory(title='Select the folder you want to move')
mb.showinfo(message='You just selected the folder to move, now please select the desired destination where you '
                    'want to move the folder to')
  destination = fd.askdirectory(title='Where to move the folder to')
  try:
shutil.move(folder_to_move, destination)
mb.showinfo("Folder moved", 'Your desired folder has been moved to the location you wanted')
  except:
mb.showerror('Error', 'We could not move your folder. Please make sure that the destination exists')
```

functions to list all files in a folder:

```
def list_files_in_folder():
i = 0
  folder = fd.askdirectory(title='Select the folder whose files you want to lisT
 files = os.listdir(os.path.abspath(folder))
list_files_wn = Toplevel(root)
list_files_wn.title(f'Files in {folder}')
```

```
list_files_wn.geometry('250x250')
list_files_wn.resizable(0, 0)
listbox = Listbox(list_files_wn, selectbackground='SteelBlue', font=("Georgia", 10))
listbox.place(relx=0, rely=0, relheight=1, relwidth=1)
  scrollbar = Scrollbar(listbox, orient=VERTICAL, command=listbox.yview)
scrollbar.pack(side=RIGHT, fill=Y)
listbox.config(yscrollcommand=scrollbar.set)
  while i<len(files):
listbox.insert(END, files[i])
i += 1
```

function to sort all the files in a folder:

```
sort_files():

src_folder = filedialog.askdirectory(title="Select the source folder")
dst_folder = filedialog.askdirectory(title="Select the destination folder")

   for filename in os.listdir(src_folder):
file_path = os.path.join(src_folder, filename)
      if os.path.isfile def (file_path):
         extension = os.path.splitext(filename)[1][1:]
         if extension:
extension_folder = os.path.join(dst_folder, extension)
            if not os.path.exists(extension_folder):
os.makedirs(extension_folder)
shutil.move(file_path, os.path.join(extension_folder, filename))


  root = tk.Tk()
root.withdraw()
tk.messagebox.showinfo(title="Done", message="Files sorted successfully.")
```

function to split files:

```
def split_file():
file_path = filedialog.askopenfilename(title="Select file to split")
chunk_size = tk.simpledialog.askinteger("Chunk Size", "Enter chunk size in MB", minvalue=1)
chunk_size *= 1024 * 1024
file_name, file_extension = os.path.splitext(file_path)
chunk_num = 1

    with open(file_path, "rb") as f:
        chunk = f.read(chunk_size)
        while chunk:
chunk_file = f"{file_name}_{chunk_num}{file_extension}"
            with open(chunk_file, "wb") as chunk_f:
chunk_f.write(chunk)
chunk_num += 1
            chunk = f.read(chunk_size)
    root = tk.Tk()
root.withdraw()
tk.messagebox.showinfo(title="Done", message="File split successfully.")
```

functions to merge two files into another file:

```
def merge_files():
    files=filedialog.askopenfilenames(title="Select files to merge");
dst_file = filedialog.asksaveasfilename(title="Save merged file as", defaultextension=".merged"
    with open(dst_file, "wb") as f:
        for file_path in files:
            with open(file_path, "rb") as src_f:
```

```
f.write(src_f.read())
    root = tk.Tk()
root.withdraw()

tk.messagebox.showinfo(title="Done", message="Files merged successfully.")
```

GUI Code

```
title = 'File Manager'
background = 'Yellow'
button_font = ("Times New Roman", 13)
button_background = 'Turquoise'
# Initializing the window
root = Tk()
root.title(title)
root.geometry('250x500')
root.resizable(0, 0)
root.config(bg=background)
```

#Creating and placing the components in the window:

```
Label(root, text=title, font=("Comic Sans MS", 15), bg=background, wraplength=250).place(x=20, y=0)
Button(root, text='Open a file', width=20, font=button_font, bg=button_background, command=open_file).place(x=30, y=50)

Button(root, text='Copy a file', width=20, font=button_font, bg=button_background, command=copy_file).place(x=30, y=90)

Button(root, text='Rename a file', width=20, font=button_font, bg=button_background, command=rename_file).place(x=30, y=130)

Button(root, text='Delete a file', width=20, font=button_font, bg=button_background, command=delete_file).place(x=30, y=170)
```

Button(root, text='Open a folder', width=20, font=button_font, bg=button_background, command=open_folder).place(x=30, y=210)

Button(root, text='Delete a folder', width=20, font=button_font, bg=button_background, command=delete_folder).place(x=30, y=250)

Button(root, text='Move a folder', width=20, font=button_font, bg=button_background, command=move_folder).place(x=30, y=290)

Button(root, text='List all files in a folder', width=20, font=button_font, bg=button_background,command=list_files_in_folder).place(x=30, y=330)

Button(root,text='sort all the files',width=20,font=button_font,bg=button_background,command=sort_files).place(x=30,y=370)

Button(root,text='split the file',width=20,font=button_font,bg=button_background,command=split_file).place(x=30,y=410)

Button(root,text='merge files',width=20,font=button_font,bg=button_background,command=merge_files).place(x=30,y=450)

 # Finalizing the window:

root.update()

root.mainloop()

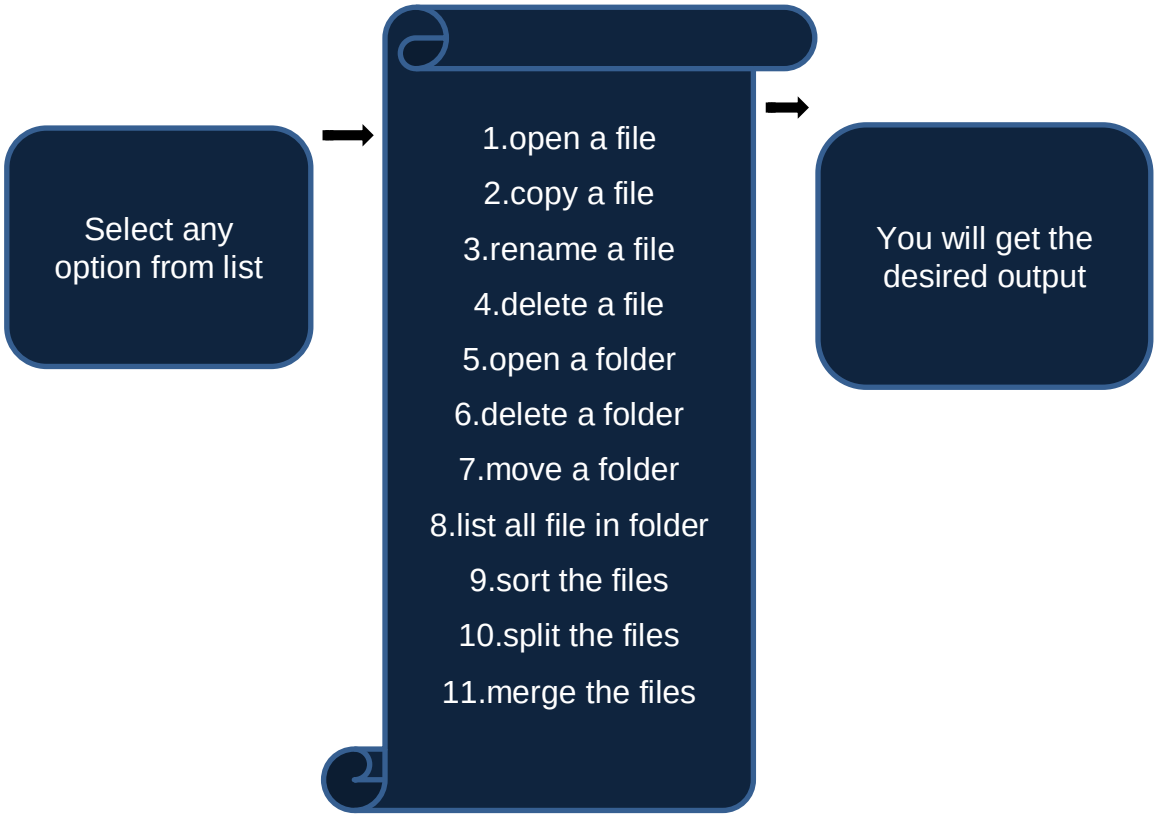
Code for Searching a file (with different searching criteria)

```
def search_files(root_path, search_query, search_criteria):
    results = []
    for dirpath, dirnames, filenames in os.walk(root_path):
```

```
        for filename in filenames:
            if search_criteria == 'name':
                if search_query in filename:
results.append(os.path.join(dirpath, filename))
elifsearch_criteria == 'extension':
                if search_query in filename.split('.')[-1]:
results.append(os.path.join(dirpath, filename))
elifsearch_criteria == 'size':
file_path = os.path.join(dirpath, filename)
ans=os.path.getsize(file_path)*1024
                if ans<= int(search_query):
results.append(file_path)
elifsearch_criteria == 'date_modified':
file_path = os.path.join(dirpath, filename)
modified_time = os.path.getmtime(file_path)
modified_time = datetime.fromtimestamp(modified_time)
                if search_query in modified_time.strftime('%Y-%m-%d'):
results.append(file_path)
    return results

def Search():
search_criteria_var = StringVar()
search_query_var = StringVar()

search_frame = Frame(root)
search_frame.pack(fill=X)

search_criteria_label = Label(search_frame, text='Search Criteria:')
search_criteria_label.pack(side=LEFT, padx=5, pady=5)
search_criteria_options = ['name', 'extension', 'size', 'date_modified']
search_criteria_menu = OptionMenu(search_frame, search_criteria_var,
*search_criteria_options)
```

```
search_criteria_menu.pack(side=LEFT, padx=5, pady=5)
search_query_label = Label(search_frame, text='Search Query:')
search_query_label.pack(side=LEFT, padx=5, pady=5)
search_query_entry = Entry(search_frame, textvariable=search_query_var)
search_query_entry.pack(side=LEFT, padx=5, pady=5)
# create search results window:
    def search():
search_results_window = Toplevel(root)
search_results_window.title('Search Results')
search_results_text = Text(search_results_window)
search_results_text.pack(fill=BOTH, expand=True)

current_path = fd.askdirectory(title="Select Folder to Search")
search_criteria = search_criteria_var.get()
search_query = search_query_var.get()
        if search_criteria == '' or search_query == '':
messagebox.showerror('Error', 'Please select a search criteria and enter a search query.')
            return
search_results = search_files(current_path, search_query, search_criteria)
        if len(search_results) == 0:
messagebox.showinfo('Search Results', 'No results found.')
            return
search_results_text.delete('1.0', 'end')
        for result in search_results:
search_results_text.insert('end', result + '\n')
search_button = Button(search_frame, text='Search', command=search)
search_button.pack(side=LEFT, padx=5, pady=5)
```

CHAPTER4
DESIGNING



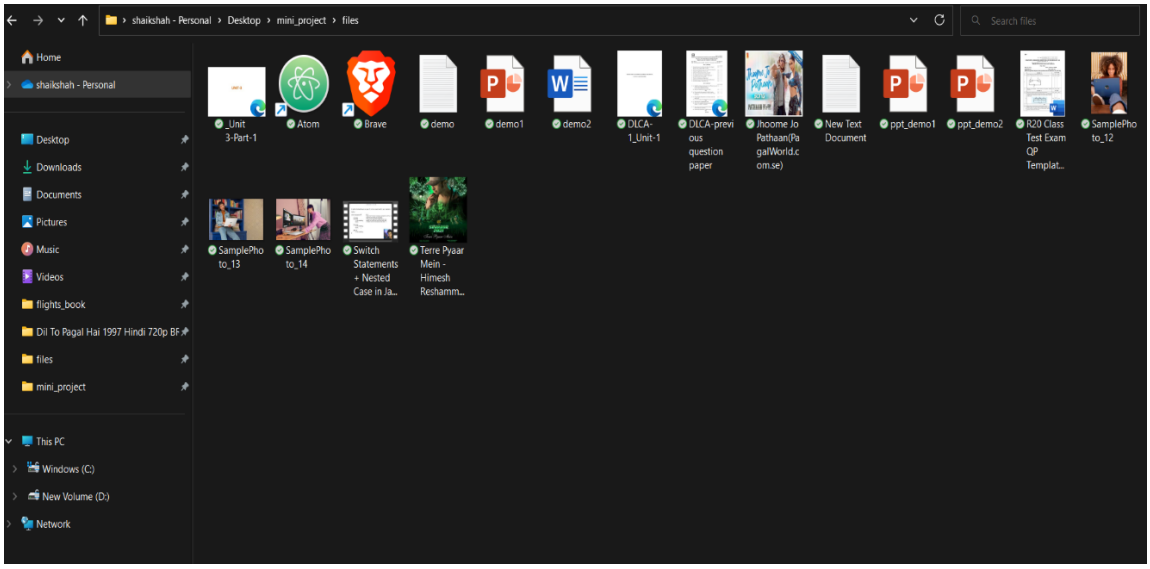
DESIGN OF THE PROJECT

CHAPTER5
RESULTS

RESULT

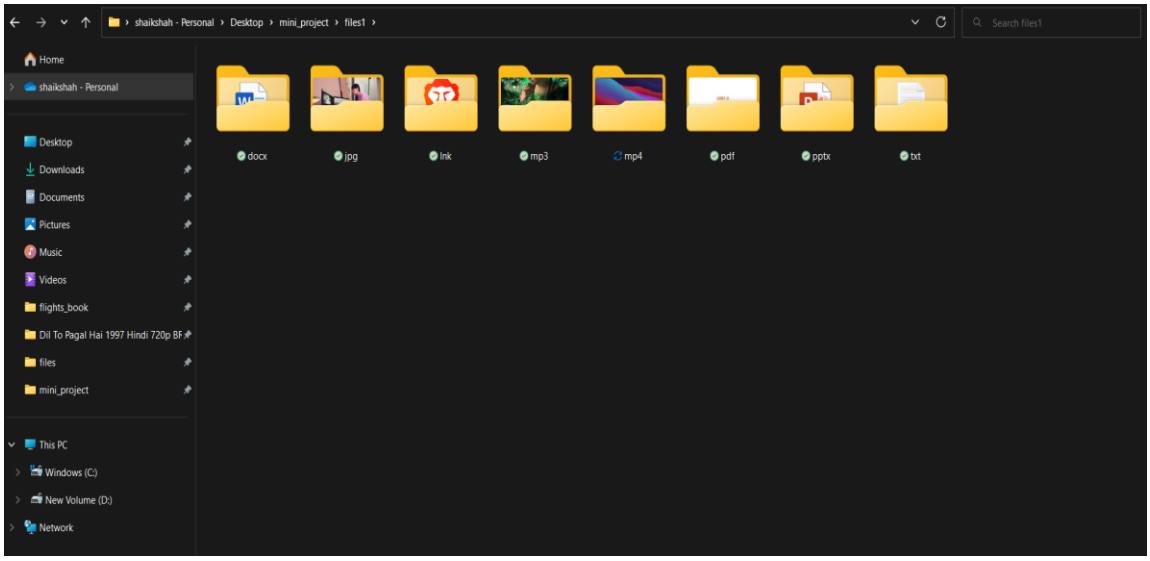
1.Sorting

Before sorting (Unsorted Files):



PIC 1

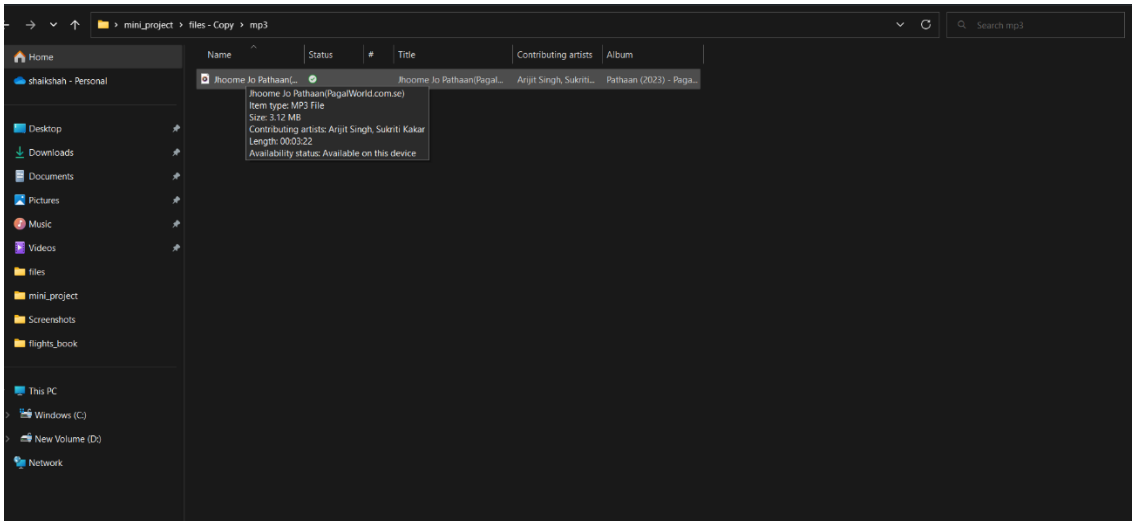
After sorting (Sorted Files):



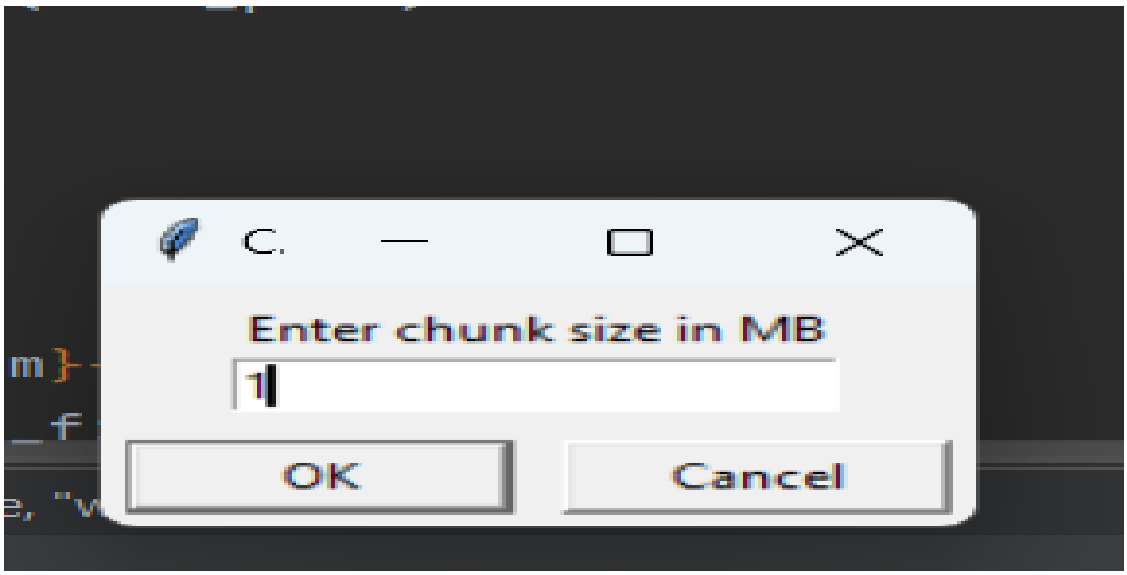
PIC 2

2.Split the Files (audio and video)

1.before splitting

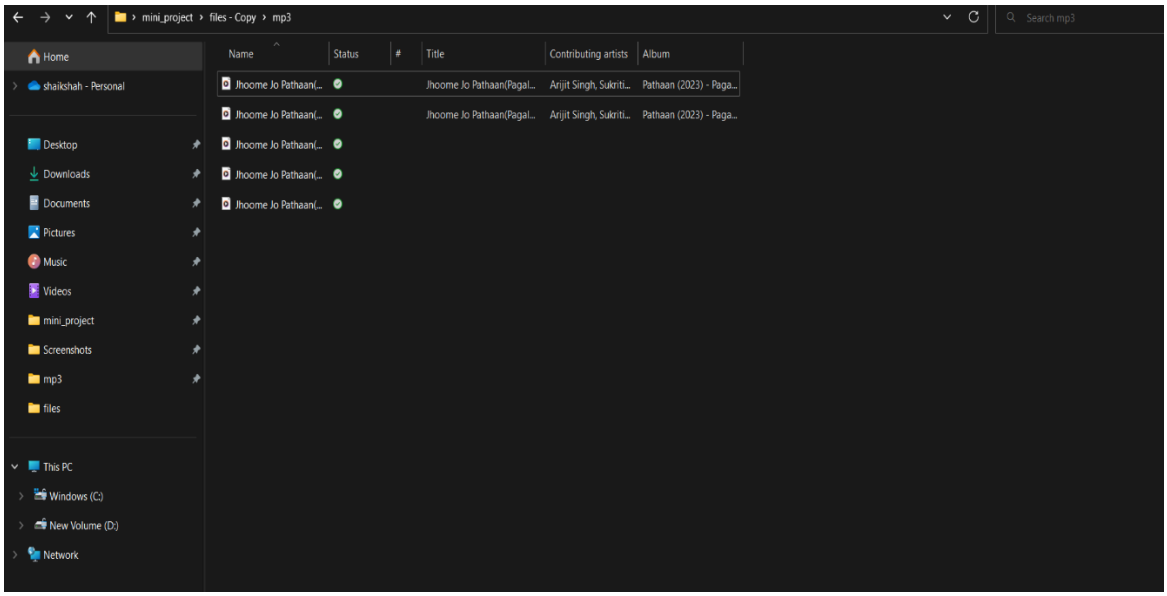


PIC.3



PIC.4

After splitting:



PIC.5

CHAPTER 6
CONCLUSION AND FUTUREWORK

Conclusion

The above-discussed file management system project provides an efficient and user-friendly way of managing files and directories. With the help of os and shutil modules in Python, the system can perform tasks such as file sorting, merging, splitting, backup, and encryption/decryption, making it a comprehensive solution for file management needs.

The project can be further improved by adding additional functionalities like:

1. Compression and decompression of files
2. Search and filter files based on different criteria
3. Add version control for files
4. Enable multi-user support for shared file management
5. Integration with cloud storage services for backup and file sharing.

In conclusion, the file management system project provides a strong foundation for managing files and directories in a computer system. The future scope of the project is immense, with several possibilities for further improvement and enhancement, making it a continuously evolving solution.

CHAPTER6
REFERENCES

REFERENCES

1. https://www.ijariit.com/manuscripts/v3i6/V3I6-1480.pdf
2. https://www.mysql.com/products/connector/
3. https://myfik.unisza.edu.my/www/fyp/fyp19sem2/report/047263.pdf