TABLE OF CONTENTS

4.INTRODUCTION

In today's digital age, where we create and store a vast amount of digital content, file organizers play a crucial role in maintaining order and ease of access to these files. A file manager is a software tool or application designed to help individuals and businesses manage and arrange their digital files and documents in an organized and efficient manner. The Python File Manager project aims to address this need by creating a user-friendly, feature-rich file management tool using the Python programming language. This project is designed to provide users with a powerful and intuitive graphical interface to navigate, organize, and manipulate files and directories on their computer systems. We all know that to organize file in one place manually is boring and time taking task that’s why this project will automatically organizes all the files.

The Python File Manager project aims to automate and simplify this otherwise tedious and time-consuming task. In a digital age where information overload is a common concern, this project seeks to enhance user experience by automating the organization of files, ensuring that users can focus on their work without the burden of manual file management.

5.OBJECTIVE

- It will categorize all the random file on the pc based on the file extension.
- It will create the generic name folders then the code will drag the file to respective folder and remove empty directory.
- Improve accessibility and ease of locating files by grouping them into logical categories.
- Reduce manual file organization efforts and potential errors.
- Enhance the user's ability to maintain a well-structured digital file system.

6. SYSTEM ANALYSIS

6.1 Existing System:

The existing system represents the state of the system before running the Python script. Here are the key points for the existing system:

1.Challenges: The existing system lacks an automated file organization mechanism. Files are typically stored in a single directory without categorization based on their types or extensions.

2.Manual Organization: Prior to running the script, users may need to manually organize their files, which can be time-consuming and error-prone.

6.2 Proposed System:

The objective of the proposed system is to automate the organization of files into directories based on their file extensions. It categorizes files into predefined categories (e.g., ZIP, IMAGES, VIDEOS) based on their extensions.

1. Input: The input for this system is the current directory where the script is executed. It scans the files in this directory for organization.

2. Output: The organized files are moved into their respective category directories within the current working directory.

6.3 SOFTWARE AND HARDWARE REQUIREMENT

HARDWARE AND SOFTWARE

SOFTWARE USED

- VS code

HARDWARE USED

MINIMUM REQUIREMENT

- INTEL CORE i3 OR PENTIUM.
- 4 GB RAM
- SSD CAPACITY 128GB OR HDD.

MAXIMUM REQUIREMENT

NO LIMIT.

PROGRAMMING LANGUAGES USED

- PYTHON language

PLATFORM USED

- Windows 10 Pro

6.4 Module Description

The File organizer module is designed to automate the organization of files within a directory based on their file extensions. The module scans the files in the current directory and categorizes them into the appropriate category directories.

Functions:

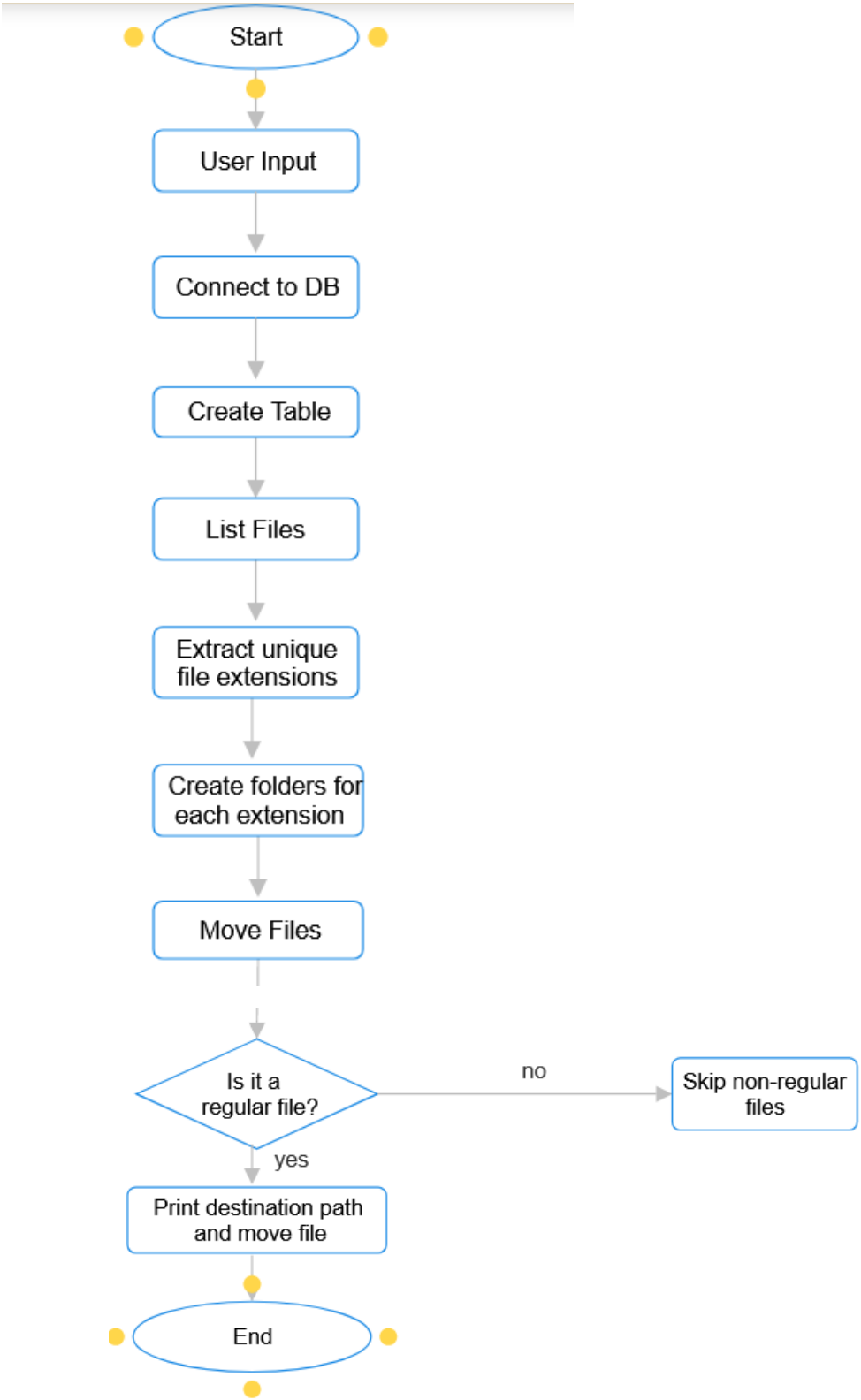
1. **GetExtintions():**
    - **Purpose:** Extracts unique file extensions from a list of files.
    - **Usage:** Called during the file organization process to identify distinct file extensions present in the specified directory.
2. **CreateFolder():**
    - **Purpose:** Creates folders for each unique file extension.
    - **Usage:** Invoked to generate directories corresponding to identified file extensions, facilitating the organized arrangement of files.
3. **MoveFiles():**
    - **Purpose:** Moves files to their corresponding folders based on file extensions.
    - **Usage:** Executed to relocate files to designated folders, streamlining the organization of the specified directory.
4. **main():**
    - **Purpose:** The main function for executing the file organization process.
    - **Usage:** The entry point of the script. When run, it connects to an SQLite database, creates necessary tables, collects user input for the directory path, organizes files, logs movements, views history, and closes the database connection.

Usage:

- Execute the script to initiate the file organization process.
- Input the path of the directory containing the files to be organized when prompted.
- The script will extract unique file extensions, create folders accordingly, move files to their designated folders, and log the movements in an SQLite database.
- After completing the file movements, the script displays the file movement history, including filenames, source paths, and destination paths.

7. System Analysis Design

7.1 FLOWCHART



7.2 Gantt Chart

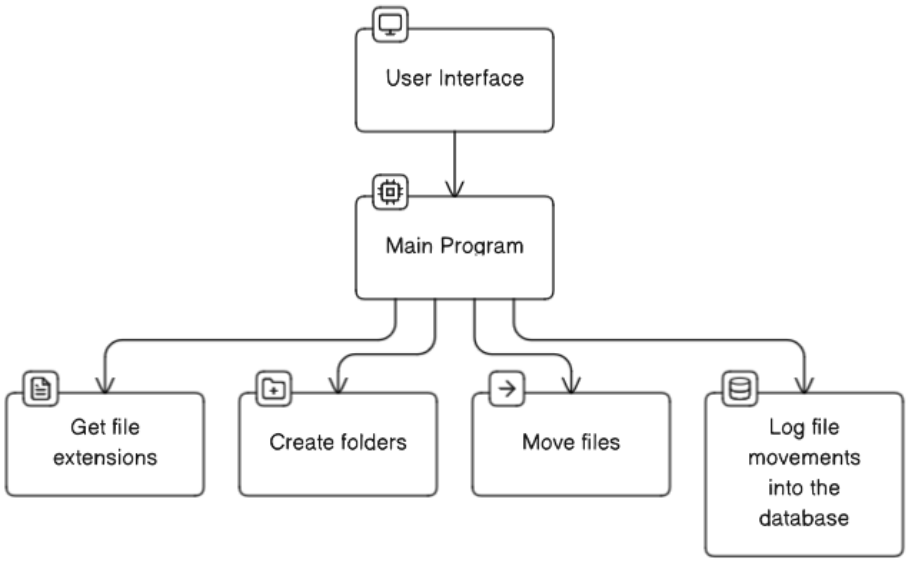
Task	Start Date	End Date
Planning	15-8-2023	30-8-2023
Research	30-8-2023	15-9-2023
Design	15--2023	30-10-2023
Follow up	30-10-2023	30-11-2023

Task Name	File Manager			
Date	15 Aug	30 Aug	15 sept	30 oct
Planning				
Research				
Design				
Follow UP				

7.3 ER DIAGRAM

7.4 DFD DIAGRAM

A Data Flow Diagram (DFD) typically illustrates the flow of data within a system.here's a DFD representation of File Manager



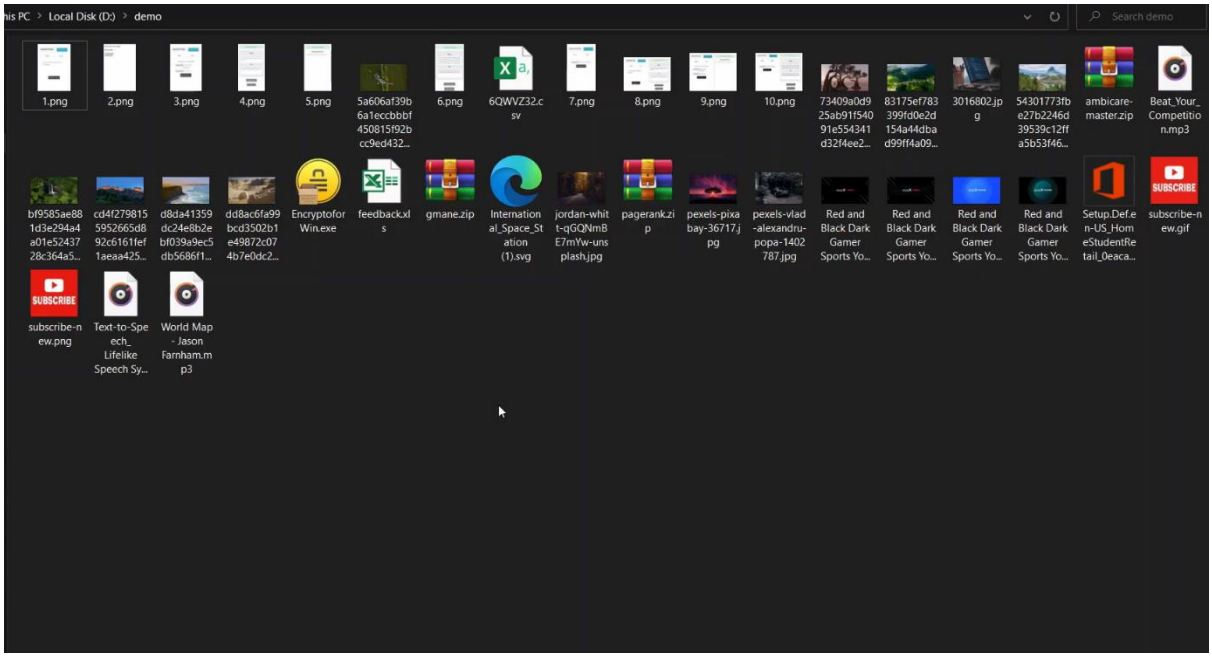
8. Appendix

8.1 CODING

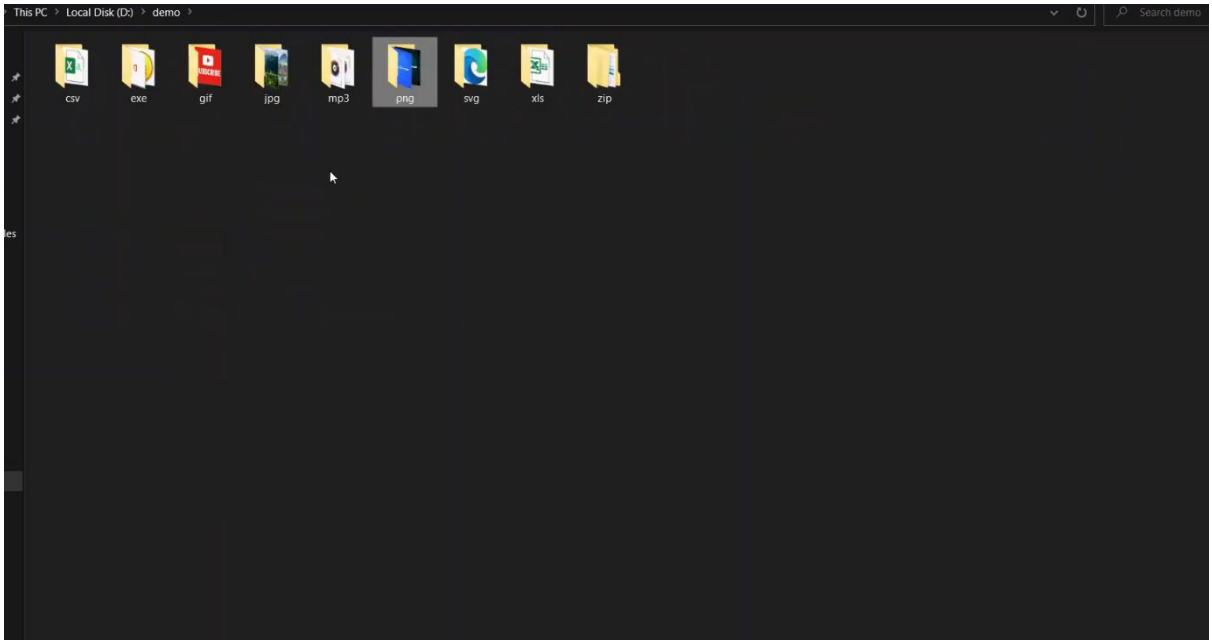
```
import pathlib as p
import os

Path = ""
files = None
ext = set()

def GetExtintions():

    for i in files:
        ext.add(i.suffix.replace(".",""))


def CreateFolder():

    for i in ext:
        try:
            os.mkdir( str(Path / i) )
        except:
            pass

def MoveFiles():
    for i in files:
        if i.is_file():
            print(Path / i.suffix.replace(".","") / i.name)
            os.rename(str(i), Path / i.suffix.replace(".","") / i.name)


def main():
    global files,Path

    Path = p.Path(input("Enter Path: "))

    files = list(Path.glob("*"))

    GetExtintions()
    CreateFolder()
    MoveFiles()


if __name__ == "__main__":
    main()
```

8.2 SCREENSHOT

Before Code Implementation



After Code Implementation



9. FUTURE SCOPE

The provided file organization script is a basic tool that can be extended and enhanced in various ways. Here are some potential future scopes and improvements:

1. **User Interface:**

- Develop a graphical user interface (GUI) to make the tool more user-friendly.

2. **Error Handling**:

- Improve error handling to provide informative messages when issues arise (e.g., permission errors, non-existent paths).

3.**Automated Sorting Rules**:

- Implement a rule-based system where users can define custom sorting rules for specific file types or conditions

10. Bibliography

- **"Let Us Python"** –-5th Edition - 2023 by Yashwant Kanetkar
- **"Learning Python"** –5th Edition, Mark Lutz, O'Reilly Media
- **"Python: For Beginners"** –1st Edition, Timothy C. Needham, September 2017

WEBSITE REFERENCES:

- https://www.learnpython.org/-Last Accessed: 19/08/2023, 07:14 PM
- https://www.w3schools.com/python/ -Last Accessed: 25/08/2023, 10:14 PM