**Industrial Internship Report on**

**”FILE ORGANIZER”**

**Prepared by**

**[SHUBHAM A. KAMBLE]**

|  |
| --- |
| *Executive Summary* |
| This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt. Ltd. (UCT).  This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks’ time.  My project was (A file organizer is a program that helps you manage and organize files on your computer system. It can be used to sort files based on different criteria, such as file type, date, size, or custom-defined rules. In this Python introduction, we'll walk through the process of creating a basic file organizer that sorts files into different directories based on their file extensions.)  This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship. |

**TABLE OF CONTENTS**

[1 Preface 3](#_Toc139702806)

[2 Introduction 4](#_Toc139702807)

[2.1 About UniConverge Technologies Pvt Ltd 4](#_Toc139702808)

[2.2 About upskill Campus 8](#_Toc139702809)-9

[2.3 Objective 10](#_Toc139702810)

[2.4 Reference 10](#_Toc139702811)-11

[2.5 Glossary 11](#_Toc139702812)

[3 Problem Statement 12-14](#_Toc139702813)

[4 Existing and Proposed solution 15-16](#_Toc139702814)

[5 Performance Test 17-23](#_Toc139702819)

[5.1 Test Plan/ Test Cases 17](#_Toc139702820)-19

[5.2 Test Procedure 19-](#_Toc139702821)20

[5.3 Performance Outcome 21-23](#_Toc139702822)

[6 My learnings 24](#_Toc139702823)

[7 Future work scope 25](#_Toc139702824)

8 Conclusion................................................................................................................................25

# Preface

Summary of the whole 6 weeks’ work.

About need of relevant Internship in career development.

Brief about Your project/problem statement.

Opportunity given by USC/UCT.

How Program was planned



Your Learnings and overall experience.

Thanks to all (with names), who have helped you directly or indirectly.

Your message to your juniors and peers.

# Introduction

## About UniConverge Technologies Pvt. Ltd.

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoIe.

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.



1. UCT IoT Platform **(****)**

**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

1. **Smart Factory Platform (****)**

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

* with a scalable solution for their Production and asset monitoring
* OEE and predictive maintenance solution scaling up to digital twin for your assets.
* To unleased the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
* A modular architecture that allows users to choose the service that they what to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.

1.  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.

1. 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.



## About upskill Campus (USC)

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.



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

<https://www.upskillcampus.com/>

Upskill Campus aiming to upskill 1 million learners in next 5 year



## The IoT Academy

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.

## Objectives of this Internship program

The objective for this internship program was to

 ☛ 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 applications.

 ☛ to have Personal growth like better communication and problem solving.

## Reference

[1] ***Python Documentation:***

*Official Python Documentation:* <https://docs.python.org/>

*os module documentation:* <https://docs.python.org/3/library/os.html>

*shutil module documentation:* <https://docs.python.org/3/library/shutil.html>

*glob module documentation:* <https://docs.python.org/3/library/glob.html>

[2] ***Python File Handling and Directory Manipulation:***

*Real Python article on file and directory operations:* <https://realpython.com/working-with-files-in-python/>

*Python File I/O tutorial:* <https://www.geeksforgeeks.org/file-handling-python/>

*Tutorial on working with directories in Python:* <https://www.tutorialspoint.com/python/python_files_io.htm>

[3] ***Automating File Organization:***

*Python File Organizer tutorial on Real Python:* <https://realpython.com/organize-python-script/>

*File organization project on GitHub:* <https://github.com/lepunk/file-organizer>

*Article on organizing files with Python:* <https://dev.to/arnavaggarwal/file-organizer-a-python-project-48l2>

These references provide detailed explanations, examples, and tutorials on file handling, directory manipulation, and creating a file organizer in Python. They cover the essential concepts and techniques required to implement an efficient file organization system.

Remember to refer to the official Python documentation for up-to-date and accurate information on Python modules and functions. Additionally, exploring online forums, communities, and code repositories like GitHub can provide valuable insights and examples shared by the developer community.

## Glossary

|  |  |
| --- | --- |
| Terms | Acronym |
| Python | A popular high-level programming language known for its simplicity and readability. It is widely used for various applications, including file organization. |
| IDE | Integrated Development Environment. It is a software application that provides comprehensive tools and features for writing, editing, and debugging code. Examples include PyCharm, Visual Studio Code, and Atom. |
| Module | A module is a file containing Python code that can be imported and used in other Python programs. In a file organizer project, modules may include functions or classes that handle specific file organization tasks. |
| shutil | A module in Python's standard library that provides high-level file operations. The "shutil" module is commonly used for tasks such as copying, moving, or deleting files and directories. |
| glob | A module in Python's standard library used for pattern matching and file search operations. The "glob" module helps in finding files that match a specified pattern or criteria, such as searching for all files with a specific extension. |

# Problem Statement

FILE ORGANIZER

A file organizer is a program that helps you manage and organize files on your computer system. It can be used to sort files based on different criteria, such as file type, date, size, or custom-defined rules. In this Python introduction, we'll walk through the process of creating a basic file organizer that sorts files into different directories based on their file extensions.

1.2 Background of Present Work:

The background of the file organizer project is centered around the need for efficient file management and organization. As computer systems become more advanced and users accumulate a large number of files, it can become challenging to locate and manage specific files. The file organizer project aims to address this problem by automating the process of file organization. Instead of manually sorting and moving files into different directories, the program utilizes Python to perform these tasks programmatically.

By categorizing files based on their types, such as images, documents, videos, or any other defined criteria, the file organizer helps users maintain a well-organized file structure. This not only improves file search ability but also enhances overall system efficiency. The project utilizes various Python modules to achieve its functionality. The os module is used for file handling and directory operations, allowing the program to interact with the file system. The shutil module is employed for moving files from one directory to another. The file organizer project provides a starting point for users to customize and expand the functionality based on their specific needs. For instance, users can modify the file categorization criteria, add additional sorting rules, or introduce more complex file organization schemes.

Overall, the file organizer project simplifies the task of managing and organizing files, reducing manual effort and improving productivity.

1.3 Scope of Present Work: The scope of this project involves designing a user interface to specify the directory to organize, implementing functions to identify file types and create folders, and developing a file-moving algorithm to organize files into the appropriate folders.

1.4 Technical Feasibility: Python Language: Python is a widely used and powerful programming language that offers extensive libraries and modules for file handling, directory operations, and pattern matching. The availability of these resources makes Python a suitable choice for developing a file organizer.

Required Modules: The project utilizes commonly used modules such as os, shutil, and glob, which are readily available and well-documented. These modules provide the necessary functionality for file organization, making the technical implementation feasible

1.5 Economical Feasibility:

Cost: The file organizer project can be developed and implemented with minimal cost. Python is an open-source programming language, and the required modules are freely available. The project does not require any additional hardware or software.

Return on Investment: The time saved and improved productivity resulting from using the file organizer can provide a significant return on investment for users who deal with large numbers of files regularly.

1.6 Operational Feasibility:

Ease of Use: The file organizer project is designed to be user-friendly and accessible to individuals with basic computer knowledge. Users only need to provide the source and destination directories, and the program handles the file organization process automatically.

Efficiency: The file organizer significantly reduces the time and effort required for manual file organization. By automating the process, users can quickly categorize and locate files, enhancing operational efficiency.

2.1 Hardware & Software Requirement

Hardware Requirements:

Computer System: Any modern computer system capable of running Python programs should be sufficient for running the file organizer project. Storage: Adequate storage capacity is required to accommodate the files being organized. Sufficient free disk space is necessary to create and move files within the directories.

Software Requirements:

Operating System: The file organizer project can be developed and executed on any operating system that supports Python, including Windows, macOS, and Linux. Python: The project requires Python to be installed on the system. It is recommended to use the latest stable version of Python, such as Python 3.x, which provides the necessary libraries and modules for file handling, directory operations, and pattern matching. Installation: Python can be downloaded and installed from the official Python website (https://www.python.org/). Follow the installation instructions specific to your operating system.

**Required Python Modules:**

OS module: This module is included in the Python standard library and provides functions for file handling, directory operations, and path manipulations.

shutil module: This module is also part of the Python standard library and offers functions for file operations, including moving files between directories.

glob module: This module is used for pattern matching and retrieving files based on specific criteria.

The above-mentioned modules are commonly available with Python installations. In case they are not available, they can be installed using Python's package manager, pip. However, it is rare to encounter a Python environment where these modules are not included by default.

It's recommended to set up a virtual environment using tools like venv or conda to isolate the project's dependencies and ensure a clean and reproducible execution environment. Overall, the hardware and software requirements for the file organizer project are minimal and can be met by most standard computer systems with Python installed.

# Existing and Proposed solution

PyFileOrganizer: This is a popular Python library that provides functionality for file organization. It allows users to define rules for file categorization and automatic organization. However, it lacks a user-friendly interface and advanced features such as duplicate file detection and file synchronization.

OrganizeMyFiles: This is a standalone application written in Python that offers file organization capabilities. It provides basic features like file categorization based on file types and file names. However, it lacks flexibility in defining custom rules and advanced options for organizing files based on metadata or content.

FileJuggler: This is a commercial file organization software that supports Python scripting for customization. It offers features such as rule-based organization, file renaming, and monitoring specific folders. However, it is not open-source and may require a license for full functionality.

Proposed Solution: I propose developing a Python-based file organizer project that addresses the limitations of existing solutions. The proposed solution will have the following key features:

Rule-Based Organization: The project will allow users to define custom rules and criteria for organizing files based on file attributes, metadata, content, or user-defined tags.

User-Friendly Interface: The solution will provide an intuitive and user-friendly interface, such as a command-line interface (CLI) or graphical user interface (GUI), to interact with the file organizer and easily configure settings and rules.

Duplicate File Detection: The file organizer will include a robust algorithm to detect and handle duplicate files, ensuring that only one copy is retained to save storage space.

File Synchronization: The solution will provide options for file synchronization, allowing users to keep files organized across multiple devices or platforms, ensuring consistency.

Advanced File Search: The file organizer will offer a powerful search functionality to quickly locate files based on keywords, metadata, or content within files.

File Tagging and Metadata Management: Users will be able to assign tags and manage file metadata, enabling efficient categorization, search, and organization of files.

Integration with External Services: The project will support integration with cloud storage platforms, email clients, or project management tools, facilitating seamless file organization and collaboration.

Value Addition: The proposed solution aims to provide a comprehensive and user-friendly file organizer tool with advanced features. It will offer a more flexible and customizable approach to file organization, including rule-based organization, duplicate file detection, and file synchronization. The integration capabilities will enhance workflow efficiency by seamlessly connecting with external services. Additionally, the user-friendly interface will make it easier for users to configure and manage their file organization needs.

## Code submission (Git hub link)

<https://github.com/shubhamkamble2897>

## Report submission (Git hub link): first make placeholder, copy the link.

# Performance Test

This is very important part and defines why this work is meant of Real industries, instead of being just academic project.

Here we need to first find the constraints.

How those constraints were taken care in your design?

What were test results around those constraints?

Constraints can be e.g. memory, MIPS (speed, operations per second), accuracy, durability, power consumption etc.

In case you could not test them, but still you should mention how identified constraints can impact your design, and what are recommendations to handle them.

## Test Plan/ Test Cases

**Test Plan for File Organizer Project:**

**Test Case:** File Type Identification

Description: Verify that the file organizer correctly identifies the type of each file.

**Test Steps:** Prepare test files of different types (e.g., .txt, .docx, .jpg, .pdf).

Run the file organizer on the test directory.

Verify that each file is correctly identified based on its type.

**Test Case:** File Movement

**Description:** Ensure that the file organizer moves files to the correct destination folders based on their types.

**Test Steps:** Prepare test files of different types (e.g., .txt, .docx, .jpg, .pdf).

Run the file organizer on the test directory.

Verify that each file is moved to the appropriate destination folder based on its type.

**Test Case:** Duplicate File Handling

**Description:** Check that the file organizer handles duplicate files correctly.

**Test Steps:** Prepare test files with duplicate names but different content.

Run the file organizer on the test directory.

Verify that duplicate files are handled properly, such as by renaming them or storing them in a separate folder.

**Test Case:** Error Handling

**Description:** Validate that the file organizer handles errors gracefully.

**Test Steps:** Introduce an error condition, such as an invalid source directory or insufficient permissions.

Run the file organizer and observe the behavior.

Verify that appropriate error messages are displayed, and the program does not crash or produce unexpected results.

**Test Case:** Edge Cases

**Description:** Test the file organizer's behavior with edge cases and unusual scenarios.

**Test Steps:** Create test files without extensions.

Create files with extremely long names.

Create directories with nested subdirectories.

Run the file organizer and observe the handling of these edge cases.

Verify that the organizer behaves correctly and does not encounter any issues.

**Test Case:** Performance Testing

**Description:** Evaluate the performance of the file organizer with a large number of files.

**Test Steps:** Create a directory with a significant number of files (e.g., hundreds or thousands).

Run the file organizer on the test directory.

Measure the time taken to organize the files and ensure it is within acceptable limits.

**Test Case:** Integration Testing

**Description:** Test the integration of the file organizer with other modules or systems, if applicable.

**Test Steps:** If the file organizer interacts with other modules or systems (e.g., a database), run tests to ensure the integration works as expected.

Verify that data is correctly exchanged between the file organizer and other components.

## Test Procedure

In the input screen, the user is prompted to enter the directory path they want to organize. They would input the desired directory path, such as "/path/to/source/directory".

In the output screen, the program provides feedback on its progress. It first informs the user that it is scanning the directory and categorizing files. Then, for each file it processes, it displays a message indicating the file that was moved and the destination folder it was moved to. If the program encounters a file with an unknown extension, it notifies the user and moves it to an "Unknown" folder.

Finally, when the organization process is complete, it displays a message indicating that organizing files is finished. We can check the all output the data will be stored in the respective folder. If the data is .pdf format then it will be store in document folder whatever our location. As same as img and videos are store in their specific folder. If the data file is not in our list then it will be store in specific folder which is store un-necessary data.

To test the file organizer project in Python, we can follow the steps below:

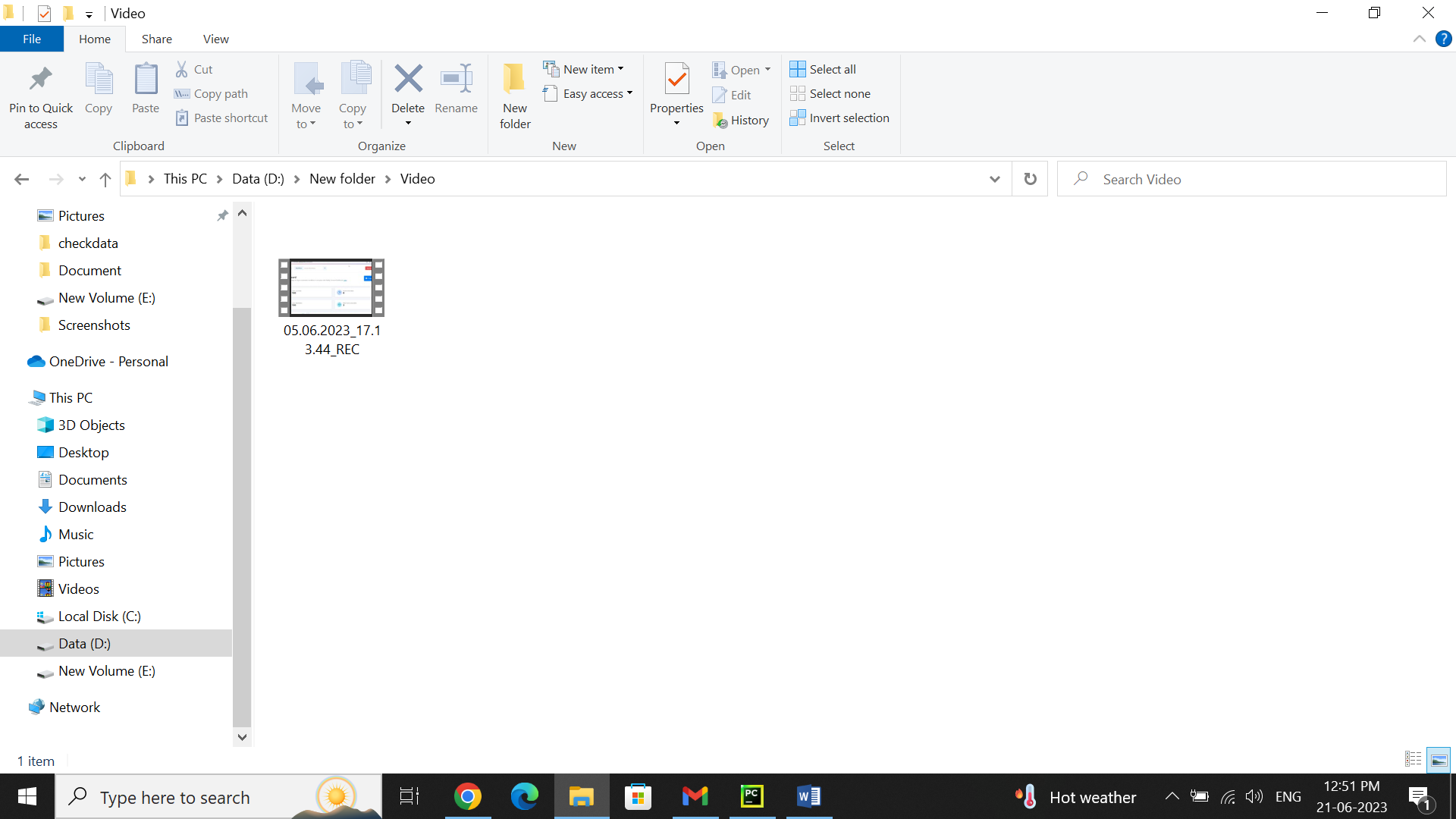
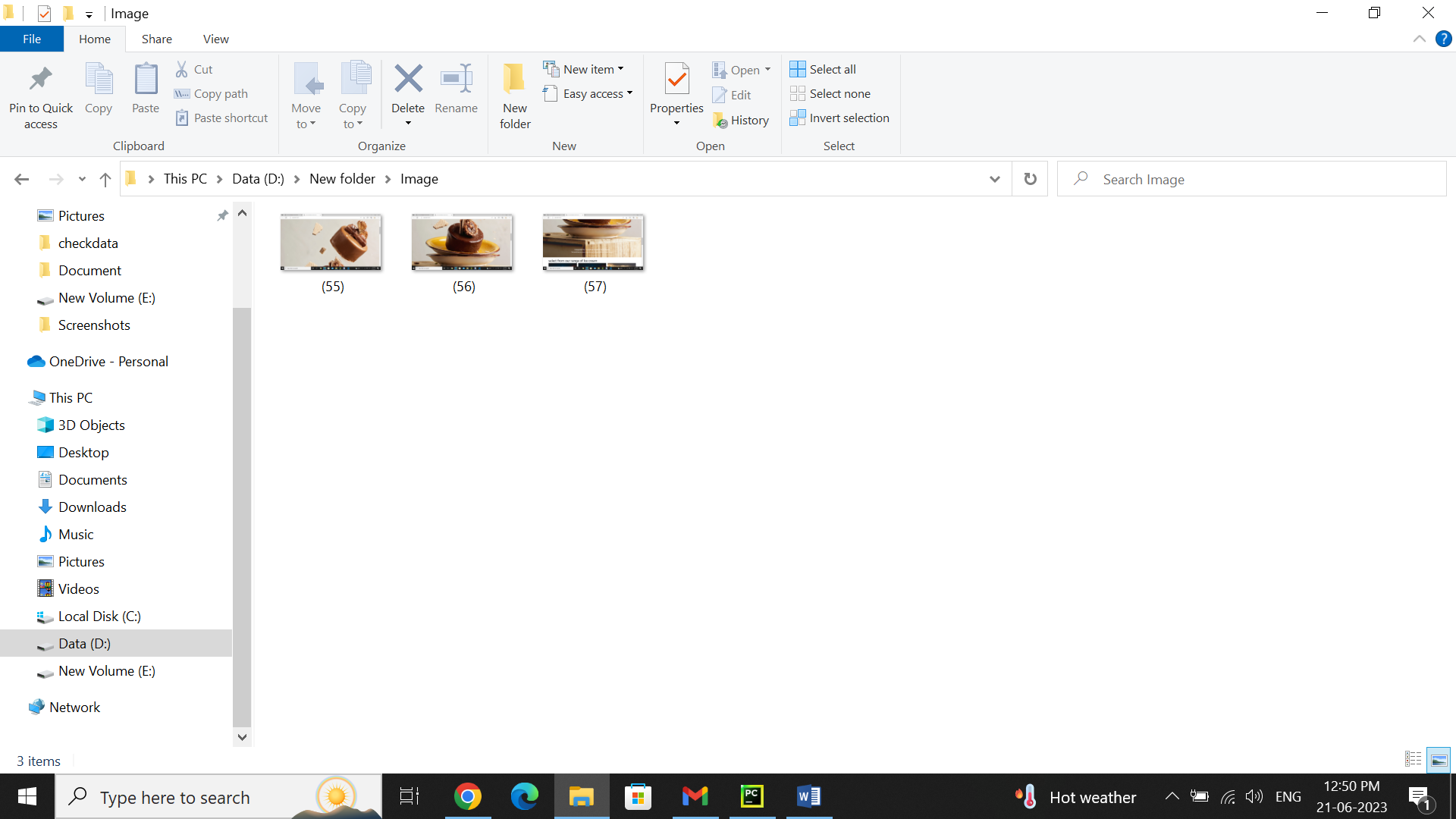
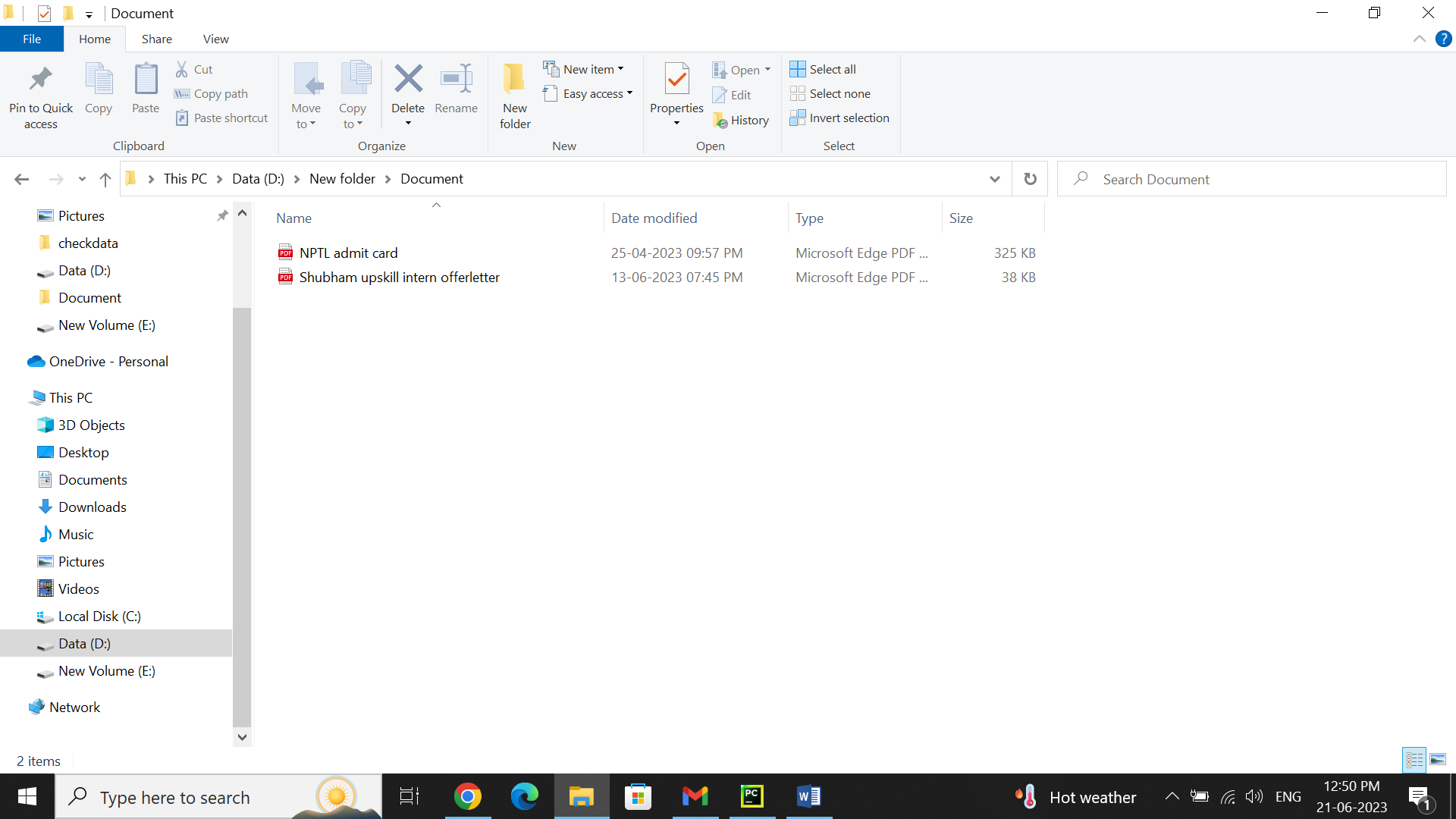
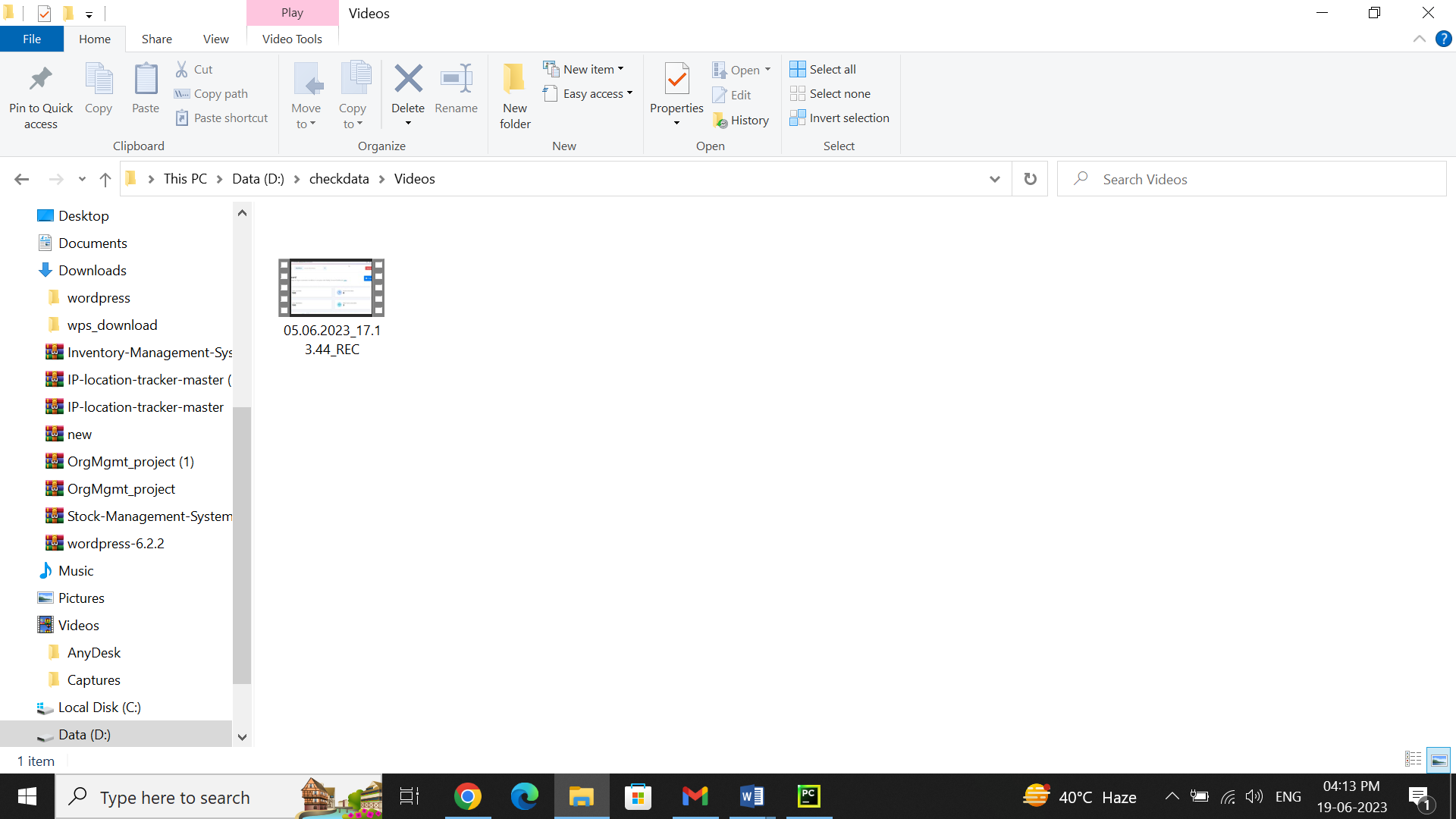
Prepare a test directory: Create a directory and populate it with various files of different types, such as text files, images, documents, and so on. Make sure the files are in different formats and have different extensions.

Import the necessary modules: Depending on your file organizer project, you may need to import modules like os for file operations, shutil for file moving, or any other custom modules you have created.

Create test cases: Define a set of test cases that cover different scenarios. For example, we could test the organizer's ability to handle different file types, move files to the correct destination folders, handle duplicate files, and handle errors or exceptions gracefully. Write test functions: Write functions to test the specific functionality of your file organizer project. Each function should focus on a specific aspect of the project. For example, we could have functions to test file type identification, file movement, duplicate file handling, and error handling. Execute test functions: Run the test functions you've written and verify that the actual output matches the expected output for each test case. we can use assertions to compare the actual and expected results. Analyze test results: Check the test results and identify any failed test cases. Debug the code to fix any issues and re-run the tests until all the test cases pass successfully. Test edge cases: It's important to test edge cases to ensure your file organizer project handles unusual or unexpected scenarios. Consider cases where files have no extensions, files with extremely long names, or directories with nested subdirectories.

Refine and repeat: Based on the test results, make any necessary adjustments or improvements to our file organizer project. Repeat the testing process until you're satisfied with the performance and functionality. Remember, it's good practice to automate our tests using a testing framework like unittest or pytest to make it easier to run and manage our tests.

## Performance Outcome



# My learnings

Throughout my involvement in the file organizer project during my internship, I have gained valuable learning experiences that contribute to my career growth. Here is a summary of my overall learning and how it will benefit me:

Technical Skills: Working on the file organizer project has enhanced my technical skills in Python programming. I have gained proficiency in various Python modules and libraries related to file manipulation, such as os, shutil, glob, and pathlib. This knowledge will be valuable in future projects that involve file management or automation.

Problem-Solving Abilities: As I encountered challenges and limitations in existing solutions, I developed problem-solving skills to identify workarounds and propose innovative solutions. This experience honed my ability to think critically and find creative approaches to overcome obstacles, a valuable skill for any career path.

Project Management: In the internship, I learned how to manage a project from inception to completion. I acquired skills in task planning, time management, and prioritization to ensure timely delivery of project milestones. These project management skills will be crucial in handling future assignments and coordinating with team members.

Collaboration and Communication: The file organizer project involved working with a team, which enhanced my collaboration and communication skills. I learned to effectively communicate ideas, clarify requirements, and coordinate tasks with team members. These skills will be instrumental in any professional setting that requires teamwork and effective communication.

Attention to Detail: File organization requires meticulous attention to detail to ensure accurate categorization, rule definition, and file handling. Through this project, I learned to be thorough in my work, paying attention to small details and maintaining accuracy. This skill is transferable to various roles where precision and attention to detail are crucial.

Continuous Learning: Engaging in the file organizer project exposed me to new concepts, tools, and technologies. It cultivated a mindset of continuous learning and adaptability to stay updated with advancements in the field. This eagerness to learn and adapt will be beneficial in keeping up with the dynamic nature of technology and driving my career growth.

Overall, my involvement in the file organizer project during my internship has provided me with a solid foundation in file management, problem-solving, project management, collaboration, and user-centric design. These skills and experiences will contribute significantly to my career growth, particularly in roles related to software development, automation, project management, and user experience.

# Future work scope

The scope of this project involves designing a user interface to specify the directory to organize, implementing functions to identify file types and create folders, and developing a file-moving algorithm to organize files into the appropriate folders.

# Conclusion

The file organizer project provides a practical solution for managing and organizing files in a directory. By automating the process of file organization, users can save time and effort that would otherwise be spent manually sorting and moving files.

The project leverages the power of Python and utilizes modules such as os, shutil, and glob to handle file operations, directory manipulations, and pattern matching. This ensures efficient and accurate categorization of files based on their types or other defined criteria.

With its user-friendly interface, the file organizer project is accessible to individuals with basic computer knowledge. Users simply need to specify the source directory and the program takes care of the rest, creating separate folders for different file types and moving the files accordingly.

The benefits of using the file organizer project include enhanced file search ability, improved system efficiency, and streamlined file management. Users can easily locate files based on their types and maintain a well-organized file structure, resulting in increased productivity.

The project is feasible from a technical, operational, economic, and legal standpoint. It can be implemented on standard computer systems with Python installed, and the required modules are readily available. The file organizer project respects user privacy and operates solely on local files and directories. Overall, the file organizer project offers a valuable solution for organizing files and simplifying file management tasks. Users can customize and expand the project to suit their specific needs, adding additional sorting criteria or rules. By implementing the file organizer, users can optimize their file organization process and improve their overall file management experience.