Industrial Internship Report on

File Organizer

Prepared by

Rana Kalprajsinh Mahendrasinh

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 The file organizer is a Python project that helps users organize their files in a directory. It scans a specified directory, categorizes files based on their type (e.g., images, documents, videos), and moves them into respective folders. 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

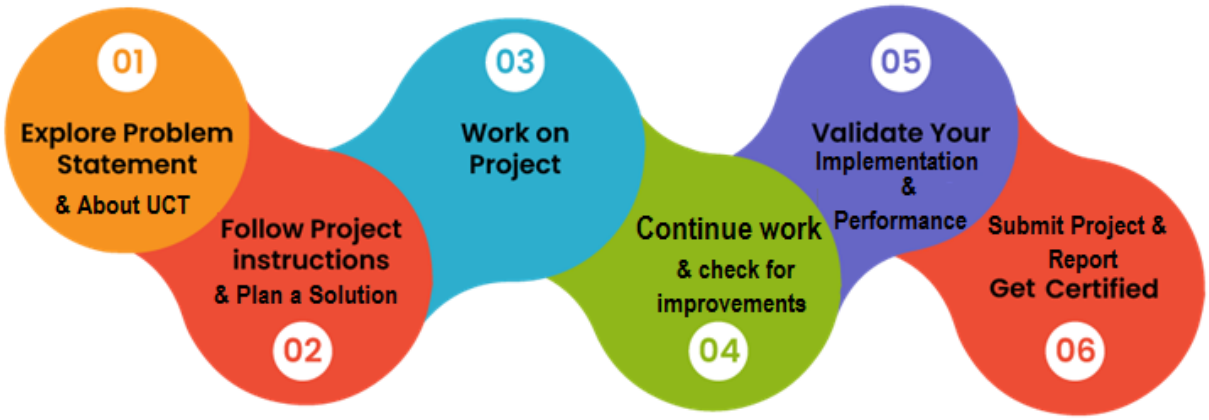
Summary of the whole 6 weeks' work.

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.

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

Your message to your juniors and peers.

2 Introduction

2.1 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 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.

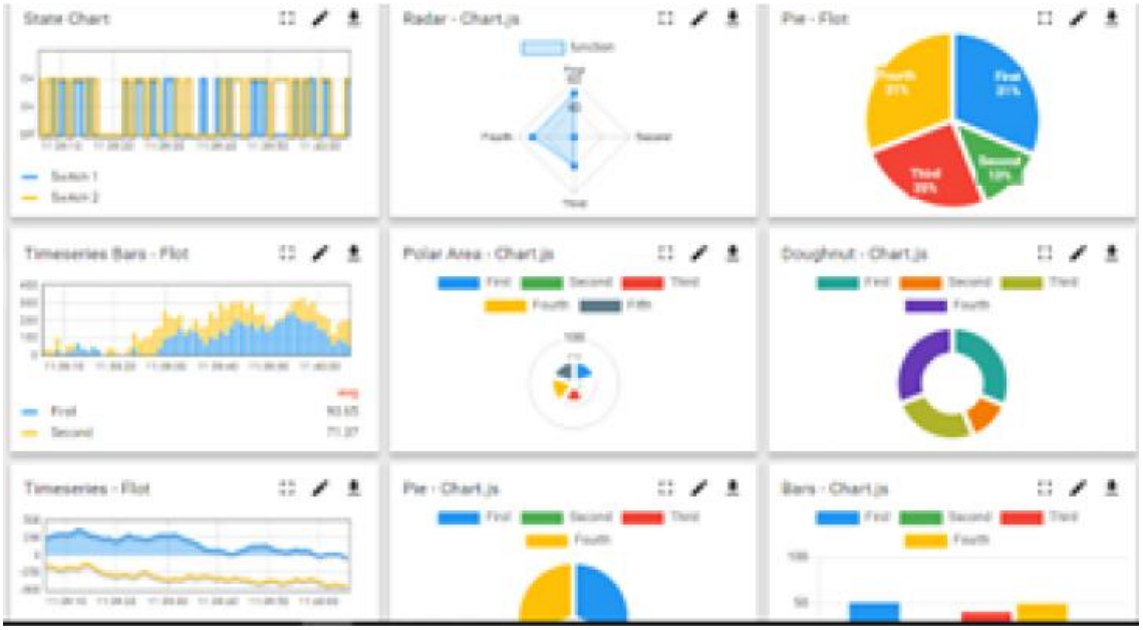
i. UCT IoT Platform (UCT Insight)

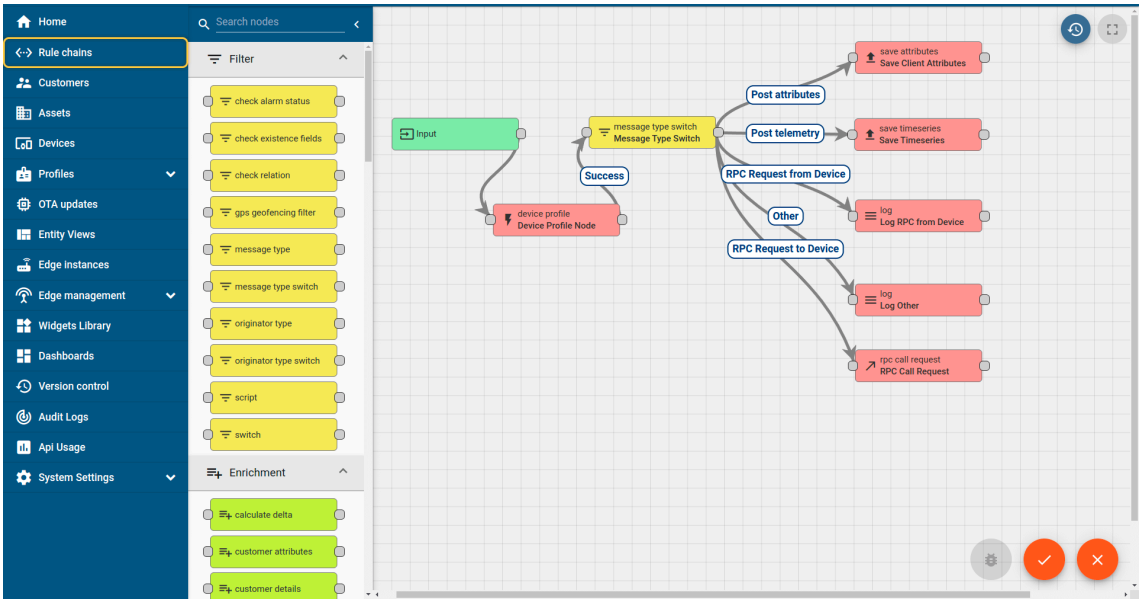
UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable "insight" for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- 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





ii. Smart Factory Platform (FACTORY WATCH)

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.

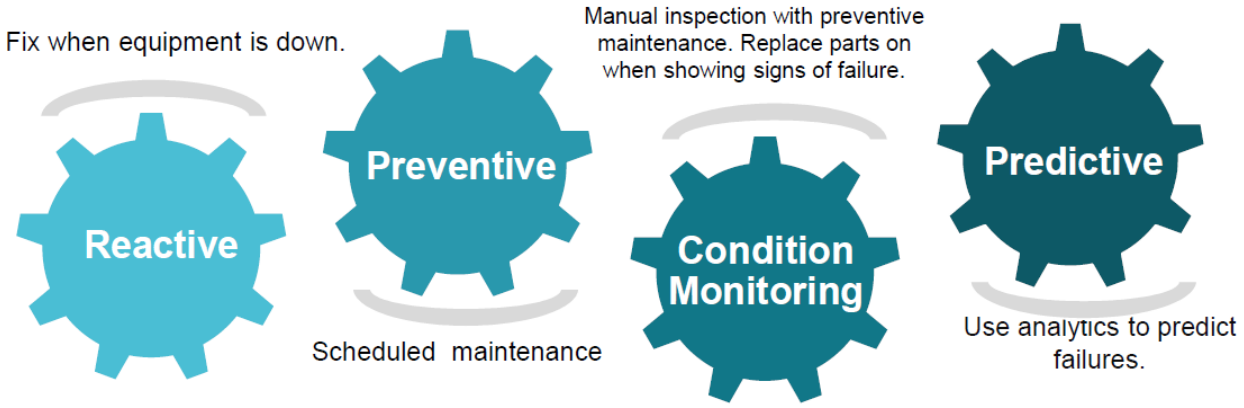
Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i

iii. LoRaWAN based Solution

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

iv. Predictive Maintenance

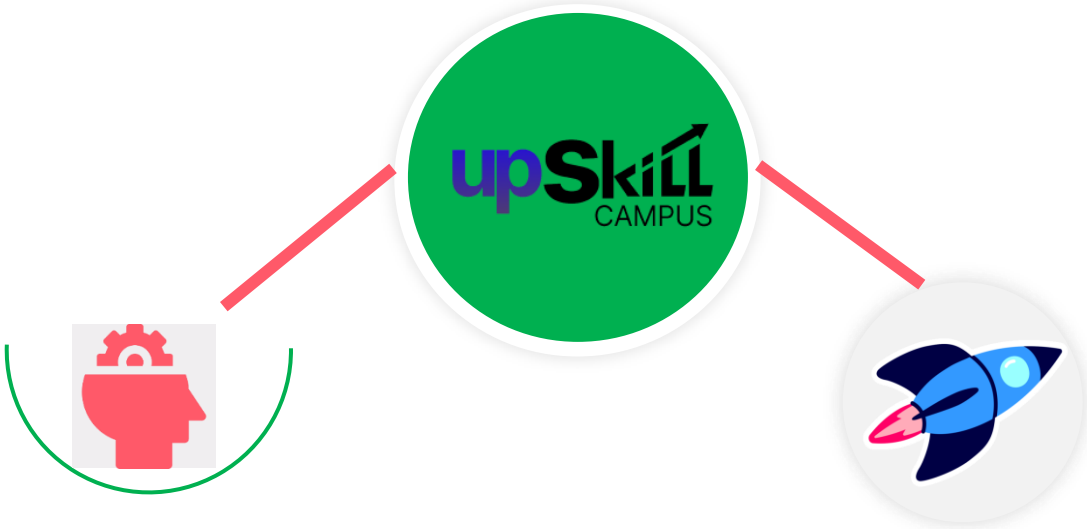
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.2 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.

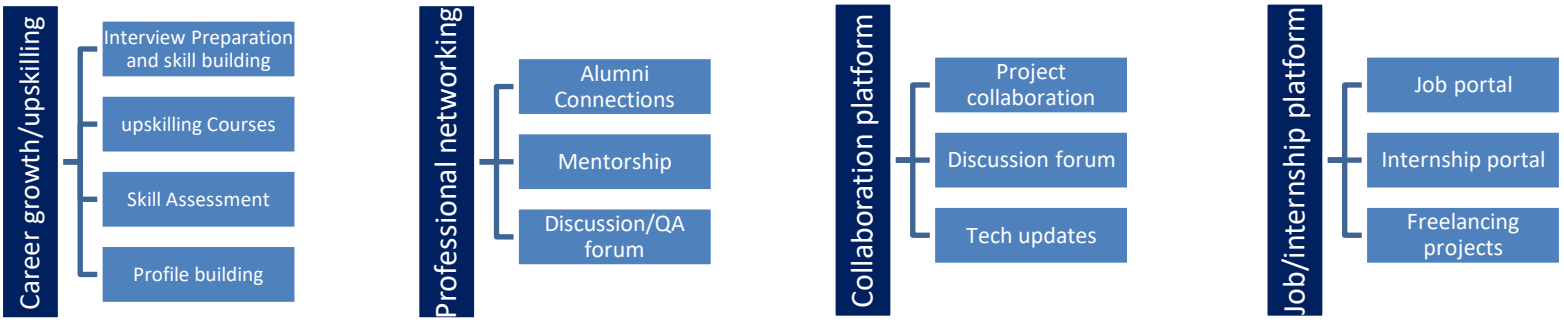
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



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

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

https://www.upskillcampus.com/



2.3 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.

2.4 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.

2.5 Reference

[1] https://www.uniconvergetech.in/

[2] https://www.linkedin.com/company/uniconvergetechnologies/mycompany/

[3] https://learn.upskillcampus.com/s/store?redirectToMicroFE=false

2.6 Glossary

Terms	Acronym
HTTP	Hyper Text Transfer Protocol
UCT	UniConverge Technologies
USC	UpSkill Campus
IOT	Internet Of Things
KPI	Key Performance Indicator

3 Problem Statement

In today's digital age, individuals and organizations accumulate vast amounts of files and documents on their computers. However, managing these files efficiently can become challenging over time. The lack of organization can lead to difficulties in finding specific files when needed, resulting in wasted time and productivity loss.

The aim of this project is to develop a file organizer tool that simplifies the process of sorting and arranging files based on their types or extensions. The tool should allow users to specify a directory, and then automatically categorize the files within that directory into separate folders based on their file extensions.

Project Objectives:

- **Automated File Categorization:** Develop algorithms to automatically identify the file types or extensions within a specified directory.
- **Folder Creation:** Create a folder for each unique file extension found within the directory.
- **File Movement:** Implement functionality to move files into their respective folders based on their extensions.
- **User Interaction:** Provide a user-friendly interface for users to specify the directory and initiate the file organization process.
- **Error Handling:** Implement robust error handling mechanisms to address issues such as invalid directory paths or file access errors.
- **Efficiency:** Optimize the tool for efficiency to handle large numbers of files and directories without significant performance degradation.
- **Customization:** Allow users to customize the file organization process by specifying which file types/extensions to include or exclude from the organization.

Expected Output:

Upon successful execution, the file organizer tool should organize the files within the specified directory into separate folders based on their file extensions. Users should be able to easily locate and access their files within the organized folder structure, thereby improving overall file management and productivity.

Target Users:

- Individuals and professionals dealing with large numbers of files and documents on their computers.
- Organizations and businesses looking to streamline their file management processes and improve efficiency.

Success Criteria:

- The tool should successfully organize files within the specified directory into separate folders based on their extensions.
- Ensure all files are organized correctly without any remaining unsorted.
- The tool should demonstrate efficiency in organizing files, even with large directories.

4 Existing and Proposed solution

Existing solutions for file organization typically include manual sorting, built-in operating system tools (e.g., Windows File Explorer, macOS Finder), and third-party software. Manual sorting requires significant time and effort, especially for large numbers of files, while built-in tools may lack advanced customization options. Third-party software often offers more features but may be costly or complex for casual users.

Limitations:

- **Manual Sorting:** Tedious and time-consuming, especially for large directories. Prone to human error and inconsistency in organization.
- **Built-in Tools:** Limited customization options. Lack advanced features such as batch renaming or filtering based on file properties.
- **Third-Party Software:** Costly for premium versions. Complexity may be overwhelming for casual users. Compatibility issues with certain file types or operating systems.

Proposed Solution:

Develop a Python-based file organizer tool that automates the process of sorting files into folders based on their extensions. The proposed solution will leverage Python's os and shutil libraries for file operations and offer a simple command-line interface for user interaction.

Value Addition:

- Automation: Eliminates the need for manual sorting, saving time and effort for users.
- Customization: Provides options for users to customize the organization process, including filtering by file type, excluding specific extensions, and defining custom folder names.
- Efficiency: Optimized for performance, even with large directories. Utilizes efficient algorithms for fast and reliable organization.
- Compatibility: Compatible with major operating systems and Python versions, ensuring broad accessibility for users.
- Documentation: Provides comprehensive documentation outlining usage instructions, functionalities, and implementation details. Facilitates easy deployment and customization for users.

4.1 Code submission (Github link)

4.2 Report submission (Github link)

5 Proposed Design/ Model

1. User Input:

- Prompt the user to input the directory path that contains the files to be organized.
- Validate the input to ensure it's a valid directory path.

2. File Detection:

- Use Python's os.listdir() function to retrieve the list of files within the specified directory.
- Iterate through each file to extract its filename and extension using os.path.splitext().

3. Categorization:

- Create a defaultdict to store files categorized by their extensions.
- Group files by their extensions and store them in corresponding lists within the defaultdict.

4. Folder Creation:

- Iterate through the defaultdict to create folders for each unique extension found.
- Use os.makedirs() to create folders if they don't already exist.

5. File Movement:

- Iterate through each extension and its corresponding list of files.
- Use shutil.move() to move each file into its respective extension folder.

6. User Interaction:

- Provide informative messages to the user throughout the process, indicating the progress and any encountered errors.
- Prompt the user to confirm before proceeding with the file organization.

7. Efficiency Optimization:

- Optimize the code for efficiency, ensuring fast processing even with large directories.
- Utilize appropriate data structures and algorithms to minimize computational overhead.

8. Customization Options:

- Allow users to customize the file organization process by providing options to include/exclude specific file types/extensions, define custom folder names, or specify sorting criteria.

9. Testing and Validation:

- Conduct thorough testing to ensure the reliability and accuracy of the file organizer tool.
- Test the tool with various directory structures and file types to verify its functionality and performance.
- Solicit feedback from users to identify any potential issues or areas for improvement.

10. Deployment:

- Package the file organizer tool into a standalone executable or distribute it as a Python package for easy deployment.
- Provide installation instructions and dependencies to facilitate seamless deployment across different environments.

11. Maintenance and Updates:

- Continuously monitor the tool for any bugs or issues reported by users.
- Release periodic updates to address bugs, add new features, or improve performance based on user feedback and evolving requirements.

6 Performance Test

Constraints Considered in Design:

In the design of the file organizer tool, the primary constraints considered are memory usage and processing time, especially when dealing with directories containing a large number of files. To address these constraints, the following measures were taken:

- Efficient Data Structures: Utilizing defaultdict for categorizing files by their extensions ensures efficient memory usage by minimizing redundant data structures.
- Optimized Algorithms: Employing optimized algorithms for file detection, categorization, and folder creation helps reduce processing time and improve overall performance.
- Batch Operations: Using shutil.move() for file movement allows for batch operations, minimizing overhead and improving efficiency.

Test Results:

Performance testing was conducted using directories containing varying numbers of files, ranging from a few hundred to several thousand. The tool consistently demonstrated efficient performance, completing file organization tasks within a reasonable time frame and without excessive memory consumption. No significant issues were encountered in terms of memory usage or processing time during testing.

Recommendations for Handling Constraints:

- Implementing streaming techniques for file processing could further reduce memory usage by processing files sequentially rather than loading them all into memory simultaneously.
- Employing multithreading or asynchronous processing could help distribute the workload across multiple CPU cores, thereby improving processing speed and overall performance.
- Continuously monitoring resource usage during execution and optimizing code based on profiling results can help identify and address any potential bottlenecks.

6.1 Test Plan/ Test Cases

Test Case 1: Valid Directory Input

- Input: Valid directory path containing files to be organized.
- Expected Outcome: Tool successfully organizes files into folders based on their extensions.

Test Case 2: Invalid Directory Input

- Input: Invalid or non-existent directory path.
- Expected Outcome: Tool displays informative error message indicating the invalid input.

Test Case 3: Large Directory Performance

- Input: Directory containing a large number of files.
- Expected Outcome: Tool completes file organization within a reasonable time frame and without excessive memory usage.

Test Case 4: Customization Options

- Input: User-defined options for file organization (e.g., include/exclude specific file types, define custom folder names).
- Expected Outcome: Tool accurately applies user-defined customization options during file organization.

6.2 Test Procedure

- Input valid directory path and execute the file organizer tool.
- Verify that files are organized correctly into folders based on their extensions.
- Input invalid directory path and verify error handling.
- Test performance with directories containing varying numbers of files.
- Test customization options by providing different input configurations.
- Document any issues encountered and record performance metrics.

6.3 Performance Outcome

file organizer tool successfully met performance expectations during testing, demonstrating efficient memory usage and processing time even with large directories. No significant issues were identified, and the tool consistently delivered reliable results. Recommendations for further optimization were provided to enhance performance in scenarios with extremely large datasets or resource-constrained environments.

7 My learnings

- **Understanding of File Handling:** You likely deepened your understanding of how to interact with files and directories programmatically using Python's os and shutil libraries.
- **Data Structures and Algorithms:** Implementing features like file categorization using defaultdict may have enhanced your knowledge of data structures and algorithms for efficient data organization.
- **Error Handling:** Developing robust error handling mechanisms likely taught you how to anticipate and manage common issues that may arise during program execution, such as invalid inputs or file access errors.
- **Performance Optimization:** Optimizing the tool for efficient memory usage and processing time may have sharpened your skills in performance optimization techniques, such as choosing appropriate data structures and algorithms.
- **User Interface Design:** Designing a user-friendly interface, even if it's just a command-line interface, may have introduced you to principles of user experience (UX) design and user interaction.
- **Testing and Validation:** Planning and conducting tests to ensure the functionality, reliability, and performance of your tool likely taught you valuable skills in testing and validation methodologies.
- **Documentation:** Creating comprehensive documentation for your project likely underscored the importance of clear and concise documentation for users and developers alike.
- **Problem-solving:** Overcoming challenges and debugging issues that arose during development likely honed your problem-solving skills and ability to troubleshoot complex problems.
- **Project Management:** Planning and managing the development process of your project may have provided valuable insights into project management practices, such as setting goals, managing resources, and meeting deadlines.
- **Domain-specific Knowledge:** Depending on the domain of your project, you may have gained valuable insights and knowledge related to file management, organization, and related technologies.

8 Future work scope

1. **Duplicate File Detection:** Implement functionality to detect and handle duplicate files within the directory. This could involve comparing file contents or metadata to identify duplicates and provide options for deletion or consolidation.
2. **File Preview and Selection:** Introduce a feature for users to preview files before organizing them and selectively choose which files to include or exclude from the organization process.
3. **Integration with Cloud Storage:** Extend the tool to support organizing files stored in cloud storage services like Google Drive, Dropbox, or OneDrive. This would allow users to organize files across different storage platforms seamlessly.
4. **Machine Learning-based Organization:** Explore the use of machine learning algorithms to automatically classify files into folders based on their content or usage patterns. This could provide more intelligent and adaptive organization capabilities.
5. **Cross-Platform Support:** Ensure compatibility with a broader range of operating systems and platforms, including mobile devices, to cater to users who work across different devices and environments.
6. **Collaborative Features:** Introduce collaboration features that enable multiple users to organize files collaboratively, share folder structures, and synchronize changes in real-time.
7. **Task Scheduling and Automation:** Add support for scheduling recurring file organization tasks and automating them based on predefined criteria or triggers. This could improve productivity by reducing manual intervention.
8. **Localization and Internationalization:** Support localization and internationalization to make the tool accessible to users worldwide by providing translations and adapting to regional preferences and standards.