**A**

**Project**

**On**

# FileSortify

**For the partial fulfilment of the Course**

***Solving Problems With Design Thinking (CBCA 208)***

**Academic Year: 2024-25 (Even Semester)**


**Submitted by**

Harshit Maan, e23bcau0049

Nikhil Sharma, e23bcau0053

Shashank Chaudhary, e23bcau0046

**Under The Supervision of**

Gopal Singh Rawat

Dakshita Sharma




**SCHOOL OF COMPUTER SCIENCE ENGINEERING & TECHNOLOGY**

**BENNETT UNIVERSITY (THE TIMES GROUP),**

**GREATER NOIDA, UP-201310, INDIA**


**April 2025**

# ACKNOWLEDGEMENT

We wish to express our sincere gratitude to everyone who provided support and assistance throughout the development of the FileSortify project. We are particularly indebted to our project supervisor, Gopal Singh Rawat and Dakshita Sharma, for their invaluable guidance, insightful feedback, and continuous encouragement, which were crucial in navigating the challenges and shaping the direction of this work.

Our thanks also extend to the Bennett SCSET at for providing the necessary facilities and academic environment that made this project possible. We also appreciate the helpful discussions and suggestions from our colleagues and peers.

Finally, we would like to extend our heartfelt thanks to our friends and family for their unwavering patience, understanding, and moral support during the entire duration of this project development.

Harshit Maan, e23bcau0049

Nikhil Sharma, e23bcau0053

Shashank Chaudhary, e23bcau0046

# ABSTRACT

Managing and organizing digital files can be a time-consuming and often tedious task for computer users. This project addresses the challenge of manual file sorting by introducing "FileSortify," a dedicated desktop application designed to automate and simplify file organization. The core problem tackled is the inefficiency inherent in manually categorizing diverse file types scattered across storage.

Developed using the Java programming language, FileSortify implements a sorting mechanism based on file types, identified by their extensions. The application workflow involves users importing a selection of files; FileSortify then automatically processes and categorizes these files accordingly. To enhance organizational flexibility, users are also provided with the option to create new destination folders directly within the application interface for the sorted files.

The current result of this project is a functional desktop tool that successfully automates the sorting of files based on their extensions, providing a practical solution for users seeking basic file management assistance. Future development plans focus on expanding the application's utility by incorporating features such as downloading files directly via URLs, developing a sandboxed environment for the safe testing of potentially malicious files, and undertaking general improvements to the user interface for a more robust and user-friendly experience.

# TABLE OF CONTENTS

# INTRODUCTION

## Background of FileSortify:

The need for file organization arises from the increasing volume of digital data that users accumulate. Manually sorting files can be time-consuming and tedious. Operating systems provide basic file management, but dedicated tools can offer more advanced and automated solutions. File Sortify is a desktop application designed to address this need by automating the process of organizing files based on their type or extension. It's built using Java, a cross-platform language, making it potentially usable on various operating systems.

## Problem Statement:

Users often struggle with the disorganization of their digital files. Downloads, documents, and other files can accumulate in a single location, making it difficult to find specific items. Manually sorting these files into folders is a repetitive and time-consuming task. The problem is the inefficiency and time consumption associated with manual file management.

## Objective of FileSortify:

The primary objective of creating File Sortify is to provide users with a tool that automates the organization of their files. The application aims to:

1) Streamline the file sorting process.
2) Save users time and effort.
3) Improve file organization and accessibility.
4) Allow users to sort files by their file type/extension.
5) Enable users to create new folders for sorted files.

## Target User / Audience:

The target audience for File Sortify includes:

1) Individuals who download and manage a large number of digital files.
2) Users who want to keep their computers organized.
3) Anyone who finds manual file sorting to be tedious and time-consuming.
4) Users who need a simple and efficient way to categorize files (e.g., students, professionals, creatives).

# TOOLS AND TECHNOLOGIES USED

## Frontend:

Java Swing or JavaFX: These are the primary GUI (Graphical User Interface) toolkits for creating desktop applications in Java. Swing is older, while JavaFX is more modern and feature-rich.

## Backend:

Java: Since it's a Java application, the core logic and file manipulation would be handled by Java itself.

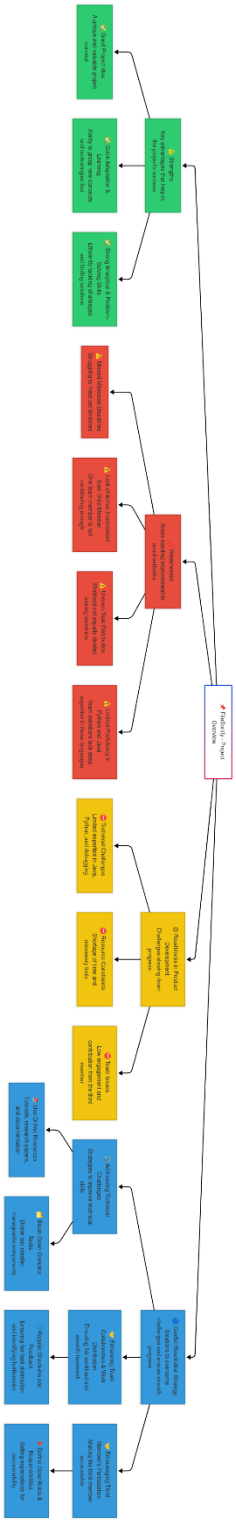
## Libraries:

Java Standard Library: This provides the essential classes for file system operations, GUI creation, and general application logic.

## Hosting / Deployment Platforms:

Desktop applications are not typically "hosted" in the same way as web applications. Deployment involves packaging the application for distribution to the user's computer (e.g., creating an executable JAR file, or platform-specific installers) which will be done in the future with more utilities added in the project.

# SYSTEM AND ARCHITECTURE DESIGN



# IMPLEMENTATION AND DEVELOPMENT

## Pages/Views:

1) **Main View:**
    - The primary window of the application.
    - contains the core functionality for file sorting and download management.
    - Elements:

      **File/Folder Selection**: The image shows an "Add Folder..." button, suggesting users can select directories.

      **Sorting Options**: The application seems to automatically categorize files into groups like "Compressed," "Documents," "Images," etc., as seen in the left panel.

      **Destination Folder Selection**: The application allows users to specify where downloaded/sorted files should be placed.

      **Download Management**: The table in the main view displays download-related information, including "File Name," "Size," "Status," "Progress," "Transfer Rate," and "Last Activity."

      **Task Management**: Buttons like "Resume," "Stop," "Stop All," and "Delete" indicate that the application manages download tasks.

      **Queues**: The application supports "Unfinished," "Finished," and "Queues," suggesting a system for managing download queues

## Key Features:

1. **File Sorting:** The core feature of the application.
    - Sorting by File Extension: Sort files into folders based on their file type (e.g., .txt, .pdf, .jpg).
    - Sorting by Date: Sort files based on their creation or modification date.
2. **Folder Creation**: Allows users to create their own folder / directory to sort the data accordingly
3. **Batch Processing**: Sort multiple files at once

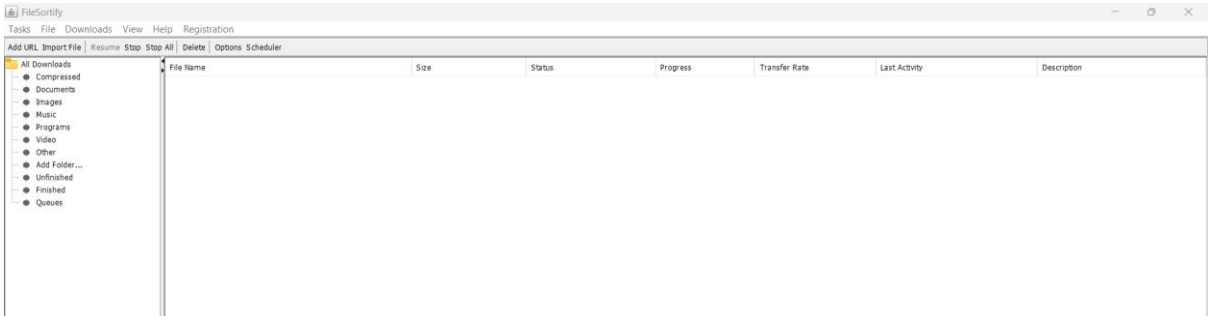
## Code Structure Explanation:

FileSortify is a Java-based desktop application that provides file management and download functionality with automatic categorization. The application follows a standalone desktop architecture using Java Swing for its GUI components.

**Core Components:**

1. **Main Application Class (FileSortify)**:
    - Extends JFrame to create the primary application window
    - Controls the overall application flow and manages UI components
    - Contains nested classes for background processing
2. **GUI Components**:
    - Split pane layout with category tree on left and file table on right
    - Menu bar with various functional menus (Tasks, File, Downloads, etc.)
    - Toolbar with common action buttons
    - Status bar for user feedback
    - Dialog boxes for various operations
3. **Background Processing**:
    - DownloadTask inner class extends SwingWorker for non-blocking downloads
    - DownloadProgress helper class for tracking and communicating download status
4. **File Management**:
    - Creates a base directory in the user's home folder
    - Organizes files into category subdirectories based on file extensions
    - Supports dynamic creation of new categories

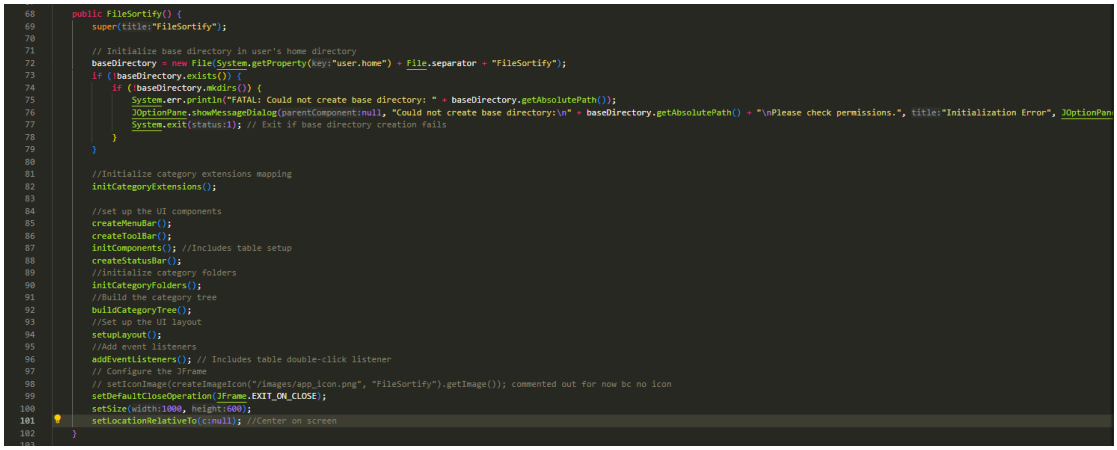
## Screenshot of FileSortify :




## Screenshots of Code Snippets:

Snapshot of import statements are used to bring in various Java classes and libraries that are needed for the program to function.

```
FileSortify.java > ...
1   import javax.swing.*;
2   import javax.swing.border.EmptyBorder;
3   import javax.swing.event.TreeSelectionEvent;
4   import javax.swing.event.TreeSelectionListener;
5   import javax.swing.table.DefaultTableModel;
6   import javax.swing.table.TableColumn;
7   import javax.swing.tree.DefaultMutableTreeNode;
8   import javax.swing.tree.DefaultTreeCellRenderer;
9   import javax.swing.tree.DefaultTreeModel;
10  import java.awt.*;
11  import java.awt.event.ActionEvent;
12  import java.awt.event.ActionListener;
13  import java.awt.event.MouseAdapter;
14  import java.awt.event.MouseEvent;
15  import java.io.*;
16  import java.net.HttpURLConnection;
17  import java.net.MalformedURLException;
18  import java.net.URI;
19  import java.net.URISyntaxException;
20  import java.net.URL;
21  import java.nio.charset.StandardCharsets;
22  import java.nio.file.*;
23  import java.text.SimpleDateFormat;
24  import java.util.*;
25  import java.util.List;
26  import java.util.concurrent.CancellationException;
27  import java.util.concurrent.ExecutionException;
```



```
public FileSortify() {
    super(title:"FileSortify");

    // Initialize base directory in user's home directory
    baseDirectory = new File(System.getProperty(key:"user.home") + File.separator + "FileSortify");
    if (!baseDirectory.exists()) {
        if (!baseDirectory.mkdirs()) {
            System.err.println("FATAL: Could not create base directory: " + baseDirectory.getAbsolutePath());
            JOptionPane.showMessageDialog(parentComponent:null, "Could not create base directory:\n" + baseDirectory.getAbsolutePath() + "\nPlease check permissions.", title:"Initialization Error", JOptionPane
            System.exit(status:1); // Exit if base directory creation fails
        }
    }

    //Initialize category extensions mapping
    initCategoryExtensions();

    //set up the UI components
    createMenuBar();
    createToolBar();
    initComponents(); //Includes table setup
    createStatusBar();
    //initialize category folders
    initCategoryFolders();
    //Build the category tree
    buildCategoryTree();
    //Set up the UI layout
    setupLayout();
    //Add event listeners
    addEventListeners(); // Includes table double-click listener
    // Configure the JFrame
    // setIconImage(createImageIcon("/images/app_icon.png", "FileSortify").getImage()); commented out for now bc no icon
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setSize(width:1000, height:600);
    setLocationRelativeTo(c:null); //Center on screen
}
```

```
    private void createMenuBar() {
        menuBar = new JMenuBar();

        // --- Tasks Menu ---
        tasksMenu = new JMenu(s:"Tasks");
        tasksMenu.add(new JMenuItem(text:"New Task (Add URL)")); // Link to Add URL action
        tasksMenu.add(new JMenuItem(text:"Add Batch Download")); // Placeholder
        tasksMenu.add(new JMenuItem(text:"Export Tasks")); // Placeholder
        tasksMenu.addSeparator();
        tasksMenu.add(new JMenuItem(text:"Exit")).addActionListener(e -> System.exit(status:0));
        menuBar.add(tasksMenu);

        // --- File Menu ---
        fileMenu = new JMenu(s:"File");
        fileMenu.add(new JMenuItem(text:"Open Selected File Folder")).addActionListener(e -> openSelectedFileLocation());
        fileMenu.add(new JMenuItem(text:"Open Base Directory")).addActionListener(e -> openBaseDirectory());
        fileMenu.addSeparator();
        fileMenu.add(new JMenuItem(text:"Properties")); // Placeholder
        menuBar.add(fileMenu);

        // --- Downloads Menu ---
        downloadsMenu = new JMenu(s:"Downloads");
        downloadsMenu.add(new JMenuItem(text:"Start All")); // Placeholder
        downloadsMenu.add(new JMenuItem(text:"Stop All")).addActionListener(e -> stopAllDownloads());
        downloadsMenu.add(new JMenuItem(text:"Remove All Completed")); // Placeholder
        downloadsMenu.addSeparator();
        downloadsMenu.add(new JMenuItem(text:"Speed Limiter")); // Placeholder
        menuBar.add(downloadsMenu);

        // --- View Menu ---
        viewMenu = new JMenu(s:"View");
        viewMenu.add(new JMenuItem(text:"Refresh")); // Placeholder
        viewMenu.add(new JMenuItem(text:"Sort By")); // Placeholder
        viewMenu.addSeparator();
        JCheckBoxMenuItem showStatusCb = new JCheckBoxMenuItem(text:"Show Status Bar", b:true);
        showStatusCb.addActionListener(e -> statusBar.setVisible(showStatusCb.isSelected()));
```

```
        // --- View Menu ---
        viewMenu = new JMenu(s:"View");
        viewMenu.add(new JMenuItem(text:"Refresh")); // Placeholder
        viewMenu.add(new JMenuItem(text:"Sort By")); // Placeholder
        viewMenu.addSeparator();
        JCheckBoxMenuItem showStatusCb = new JCheckBoxMenuItem(text:"Show Status Bar", b:true);
        showStatusCb.addActionListener(e -> statusBar.setVisible(showStatusCb.isSelected()));
        viewMenu.add(showStatusCb);
        JCheckBoxMenuItem showToolbarCb = new JCheckBoxMenuItem(text:"Show Toolbar", b:true);
        showToolbarCb.addActionListener(e -> toolBar.setVisible(showToolbarCb.isSelected()));
        viewMenu.add(showToolbarCb);
        menuBar.add(viewMenu);

        // --- Help Menu ---
        helpMenu = new JMenu(s:"Help");
        helpMenu.add(new JMenuItem(text:"Help Contents")); // Placeholder
        helpMenu.add(new JMenuItem(text:"Check for Updates")); // Placeholder
        helpMenu.addSeparator();
        helpMenu.add(new JMenuItem(text:"About FileSortify")); // Placeholder
        menuBar.add(helpMenu);

        // --- Registration Menu (Optional) ---
        registrationMenu = new JMenu(s:"Registration");
        registrationMenu.add(new JMenuItem(text:"Register")); // Placeholder
        registrationMenu.add(new JMenuItem(text:"Enter License Key")); // Placeholder
        menuBar.add(registrationMenu);

        setJMenuBar(menuBar);
    }
```

```
    private void createToolBar() {
        toolBar = new JToolBar();
        toolBar.setFloatable(b:false);
        toolBar.setBorder(BorderFactory.createEtchedBorder());

        // Create buttons (icons would be better)
        addUrlButton = createToolbarButton(text:"Add URL", iconName:"url_icon.png");
        importFileButton = createToolbarButton(text:"Import File", iconName:"import_icon.png");
        resumeButton = createToolbarButton(text:"Resume", iconName:"resume_icon.png");
        stopButton = createToolbarButton(text:"Stop", iconName:"stop_icon.png");
        stopAllButton = createToolbarButton(text:"Stop All", iconName:"stop_all_icon.png");
        deleteButton = createToolbarButton(text:"Delete", iconName:"delete_icon.png");
        optionsButton = createToolbarButton(text:"Options", iconName:"options_icon.png");
        schedulerButton = createToolbarButton(text:"Scheduler", iconName:"scheduler_icon.png");

        // Add buttons to toolbar
        toolBar.add(addUrlButton);
        toolBar.add(importFileButton);
        toolBar.addSeparator();
        toolBar.add(resumeButton);
        toolBar.add(stopButton);
        toolBar.add(stopAllButton);
        toolBar.addSeparator();
        toolBar.add(deleteButton);
        toolBar.addSeparator();
        toolBar.add(optionsButton);
        toolBar.add(schedulerButton);

        // Disable unimplemented features
        resumeButton.setEnabled(b:false);
        // optionsButton.setEnabled(false);
        // schedulerButton.setEnabled(false);
    }
```

```
553     private void addNewFolder() {
554         String folderName = JOptionPane.showInputDialog(this,
555                 message:"Enter name for new category folder:",
556                 title:"Add New Folder",
557                 JOptionPane.PLAIN_MESSAGE);
558 
559         if (folderName != null && !(folderName = folderName.trim()).isEmpty()) {
560             //Basic validation for folder name (avoiding invalid chars)
561             if (folderName.matches(regex:".*[\\\\/:*?\"<>|].*")) {
562                 JOptionPane.showMessageDialog(this, message:"Folder name contains invalid characters.", title:"Invalid Name", JOptionPane.WARNING_MESSAGE);
563                 return;
564             }
565             //Check if category/folder already exists (case-insensitive)
566             boolean exists = false;
567             for (String existing : categoryExtensions.keySet()) {
568                 if (existing.equalsIgnoreCase(folderName)) {
569                     exists = true;
570                     break;
571                 }
572             }
573             //also check other static folders like "Finished", etc.
574             if (!exists) {
575                 String[] additionalNodes = {"Unfinished", "Finished", "Queues"};
576                 for (String nodeName : additionalNodes) {
577                     if (nodeName.equalsIgnoreCase(folderName)) {
578                         exists = true;
579                         break;
580                     }
581                 }
582              }
583 
584             if (exists) {
585                 JOptionPane.showMessageDialog(this, message:"A category or folder with this name already exists.", title:"Folder Exists", JOptionPane.WARNING_MESSAGE);
586                 return;
587             }
```

# TESTING

## How the Application was Tested:

Manually: This involved us to directly interact with the application's GUI to verify its functionality, such as:

- **File sorting**: We selected various files and folders, configure different sorting options, and check if the files are correctly sorted into the expected directories.
- **Usability:** Ensure the application is easy to use, and the GUI is intuitive.

## Browser Compatibility Tests:

FileSortify isn't a browser compatible web app at the moment, though in the future we plan on having FileSortify interact with web URLs to download data and file right from the web.

## Bug Fixing Process:

- **Bug reporting:** bugs such as the toolbar buttons making the another folder instead of doing their functionality.
- **Bug Analysis:** analysis of why, what, and how of the bug issue.
- **Code Fix:** fixed the bug by working on the code and on their functionality and if some feature wasn't / hasn't been implemented then a small prompt saying "this feature isn't implemented" or so
- **Testing:** The fix is tested again and again until the bug is fixed, tested by the user who had reported the bug
- **Deployment:** add the code changes and deploy the app (later on)

# CHALLENGES FACED

## Technical Issues:

1. File System Handling:
    - Ensuring robust and reliable file operations across different operating systems (Windows, macOS, Linux) can be complex. Java helps with cross-platform compatibility, but subtle differences in file systems can still cause issues.
    - Handling file permissions, especially when moving or deleting files, can be tricky.
    - Dealing with very large numbers of files or very large files efficiently.
2. GUI Development:
    - Creating a user-friendly and responsive GUI using Swing or JavaFX can be challenging.
    - Ensuring the UI is consistent across different operating systems and screen resolutions.
    - Handling UI updates efficiently, especially when performing long-running operations like file sorting or large downloads, to prevent the application from freezing.
3. Concurrency: Properly using threads and synchronization to ensure smooth performance and prevent data corruption.

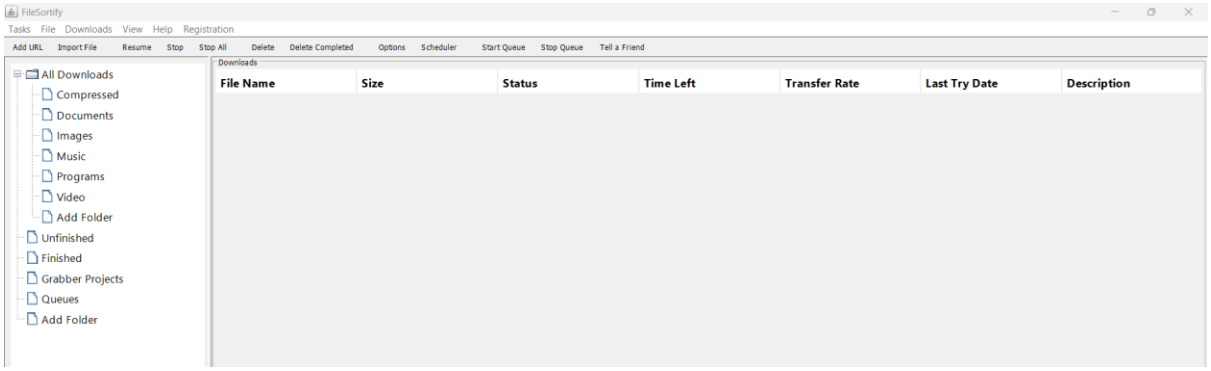
## Time Management:

Balancing the development of different features within the time constraints which led to leaving some features for the future updates. Prioritizing essential features and cutting down the less potential ones (such as tell a friend, delete completed etc which will be implemented later on, right now the button just sits and gives a prompt when clicked). Estimating development time was difficult, also keeping up with the milestones while having to work on other projects was hectic.

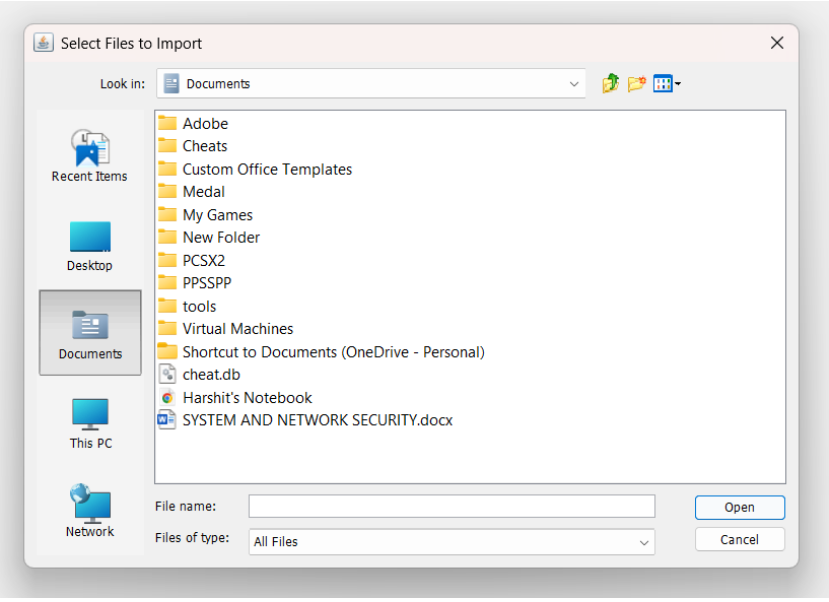
# RESULTS

## Final Output:

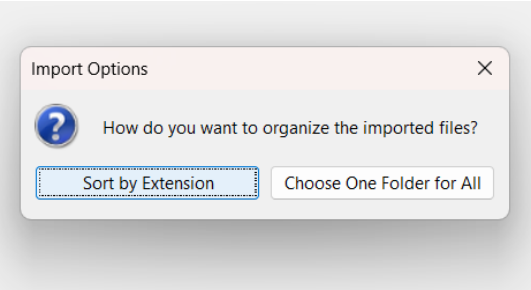
Working application snapshot:



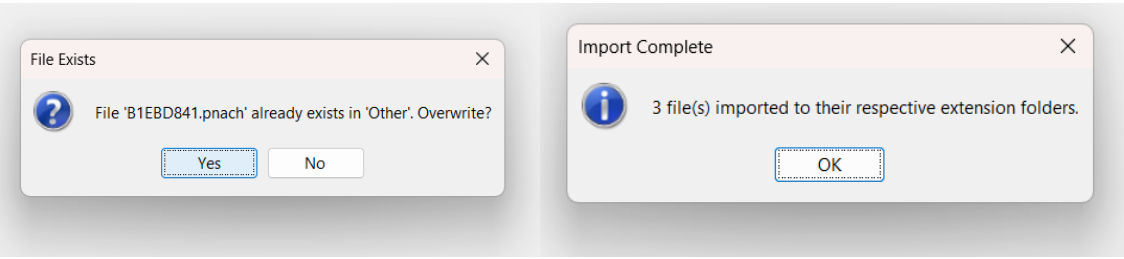

Here we click on the import file option (next to add URL) and a new screen opens up, using which we can navigate through our file system and choose any file / folder to sort (showed in the next snapshots)

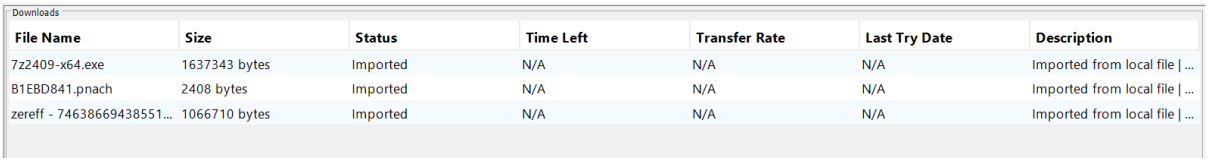


Now we select the file(/s) we want to import/sort, here we can either choose to sort by their respective extensions or create a folder for the multiple files (applied only when importing multiple files, for single files we can either let it sort automatically or create a new folder for it)

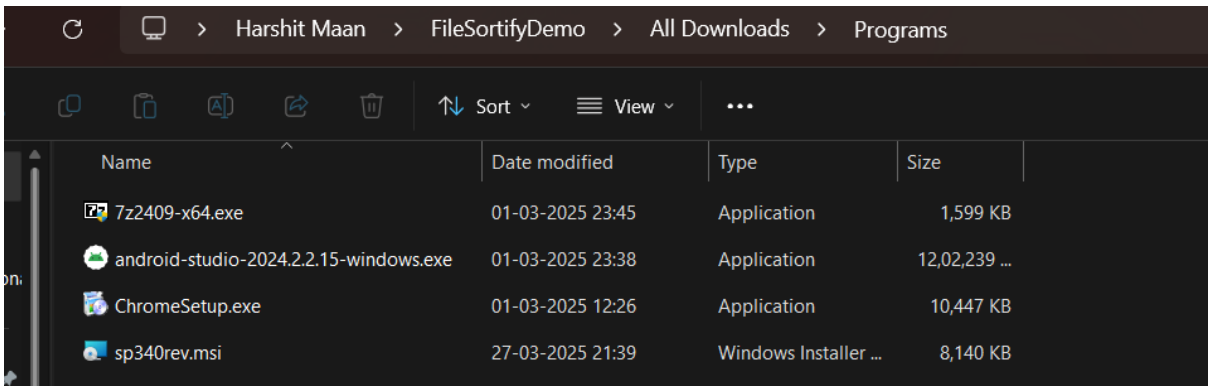



Overwrite functionality if the file already exists + the import complete prompt




When we click on the files from in the download menu, it takes us to the respective folder of that file



## Feedback received:

Positive feedback from users regarding the application's ease of use and time-saving features.

Suggestions for the app are similar to the features we are going to add in later phases

# CONCLUSION

## What I Learned:

As I developed a java desktop application, I got to understand (Swing, JavaFX) for file sorting and management. Also learnt about implementing core functionalities such as file system handling, sorting algorithms, GUI design etc.

## How the objectives were achieved:

- The primary objective of creating a tool to automate file organization was achieved by implementing the file sorting functionality. Which also addresses the problem of time-consuming manual file management
- The application provides a user-friendly interface, allowing users to efficiently sort files.

## Future Improvements:

1. Enhance the file sorting capabilities:
    - Support more sorting options (e.g., by file size, file attributes). (feedback)
    - Allow users to define custom sorting rules with more flexibility. (feedback)
    - Implement more advanced file organization features (e.g., duplicate file handling, symbolic linthk management).
2. Expand Download Management:
    - Work on the “add URL” feature which takes up a URL from the web and downloads the data
    - Then add functionality to support more download protocols (e.g., BitTorrent).
    - Adding a functionality for Scheduling and prioritizing downloads
    - Also, add a sandbox feature for testing of suspicious files
3. Improve user interface:
    - Modernize the look, as well as add a theme toggling button for users to switch between themes
    - Enhancing the Functionality by adding a drag and drop functionality, better visual cues
4. Add Advanced features:
    - Implement file synchronization between directories

- Providing support for file extraction and file compression
- Integrating with cloud storage services / Database services

# REFERENCES


**W3Schools: www.w3schools.com**

**Java the Complete Edition by Herbert schildt**

**GeeksforGeeks: geeksforgeeks.org**

And any other sources that I came across while looking for help.

# APPENDIX

## Full Source Code:

The full source code for FileSortify is available on Harshit Maan's GitHub, it includes the Java file with the full code as of now and our report which we are presenting.

GitHub repo: https://github.com/sadgefujin/File-Sortify