Documentation – “Total Explorer”

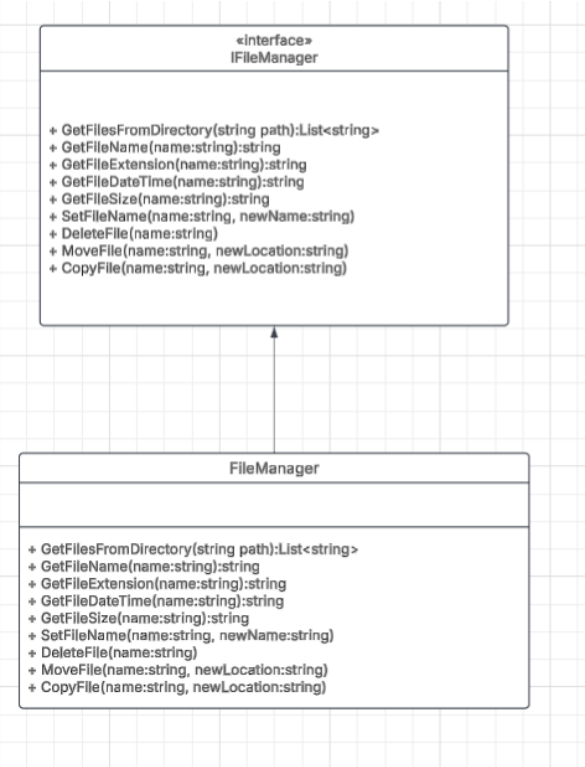
Team

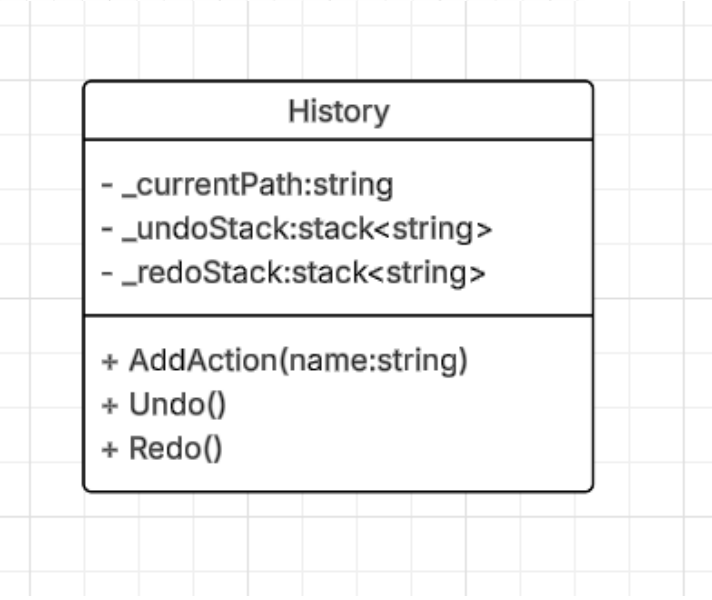
Paladi Andrei(1310A)
Ciobanu Andrei(1310A)
Rață Mihai-Gabriel(1310A)
Timofte Alin-Gabriel(1310A)

Total Explorer — Technical Documentation
Developed for the Software Engineering course

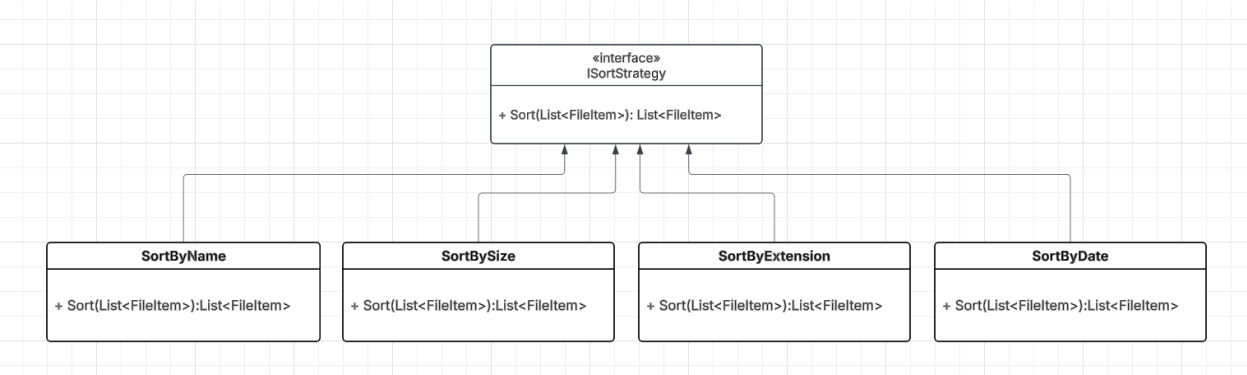
UML Diagrams

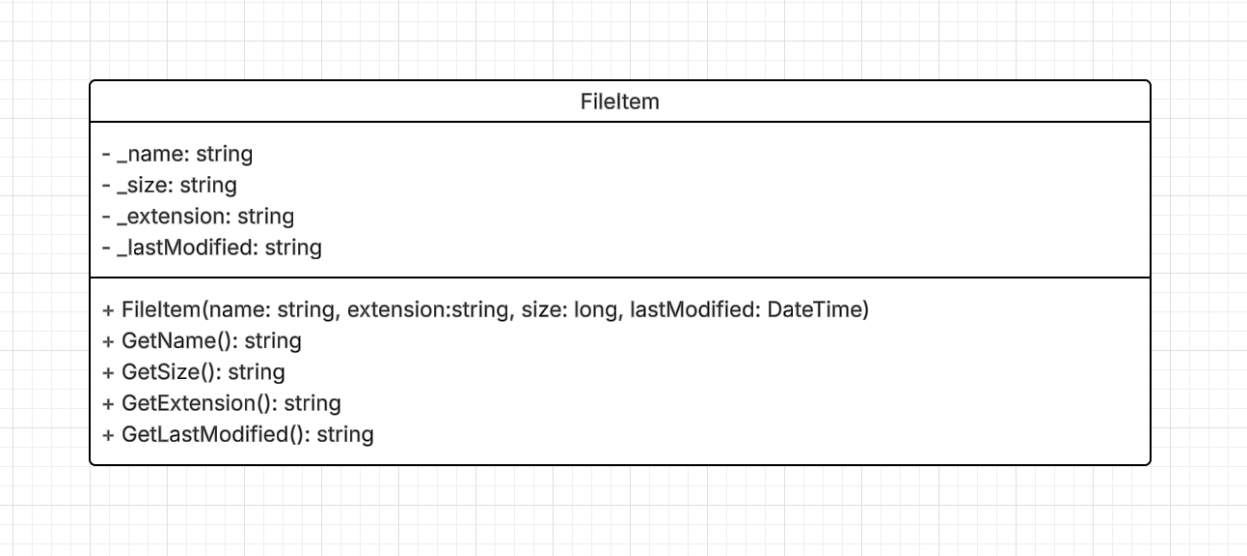
Class Diagrams:



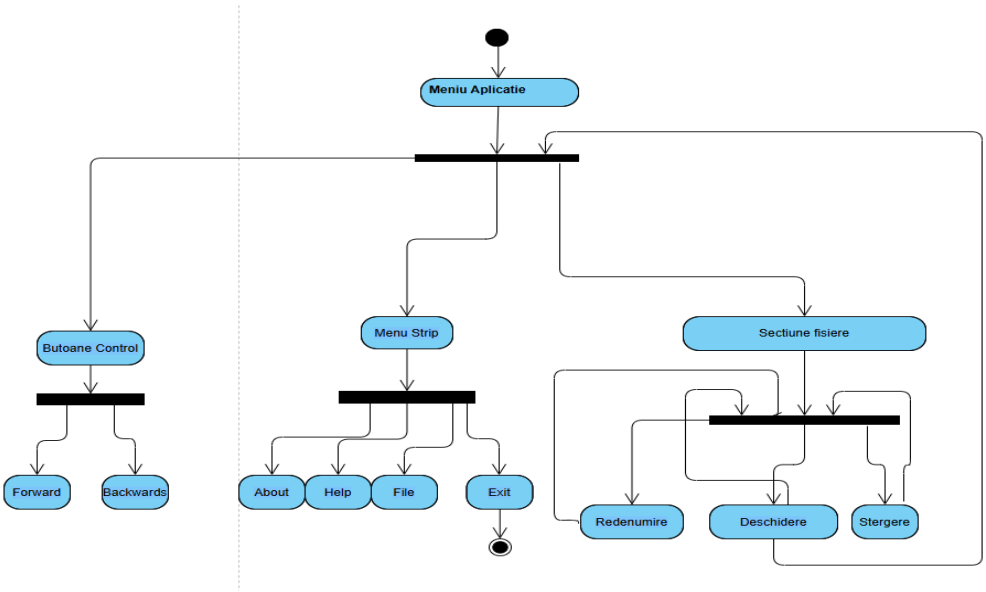


-Strategy Desing Pattern:

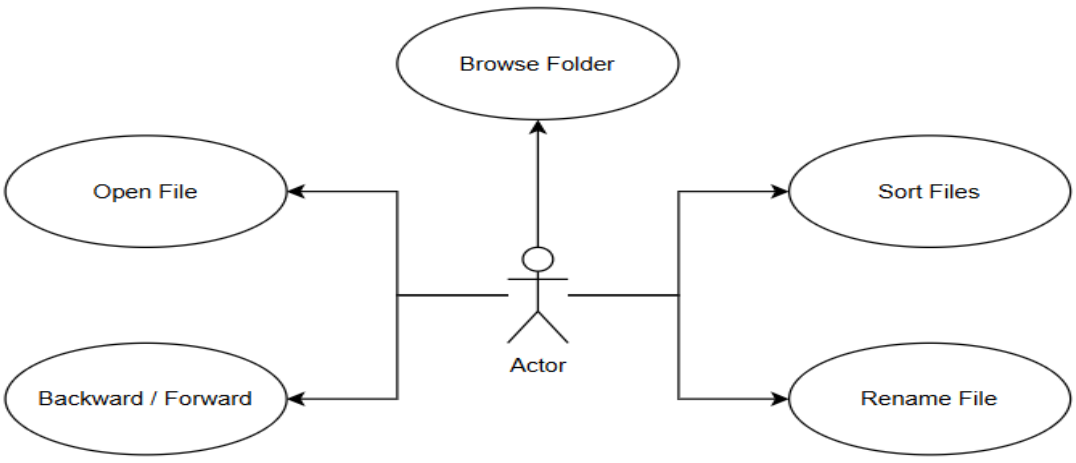




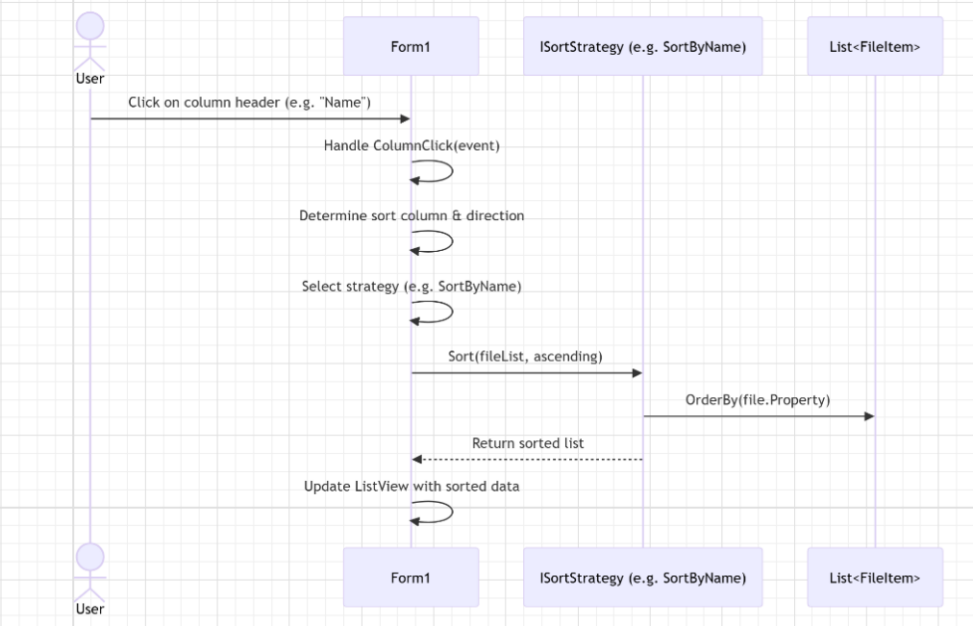
Activity Diagram:



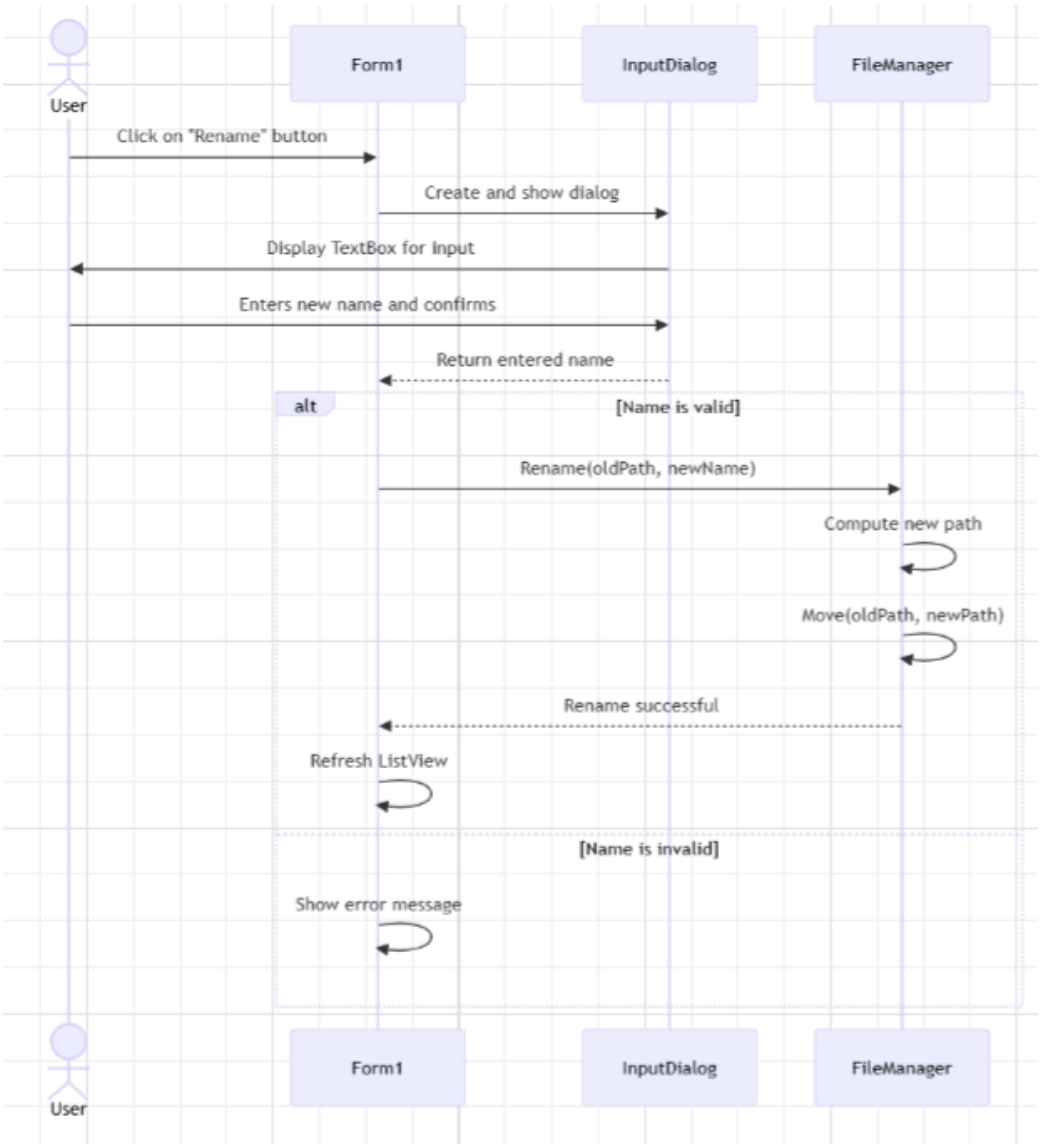
User Diagram:



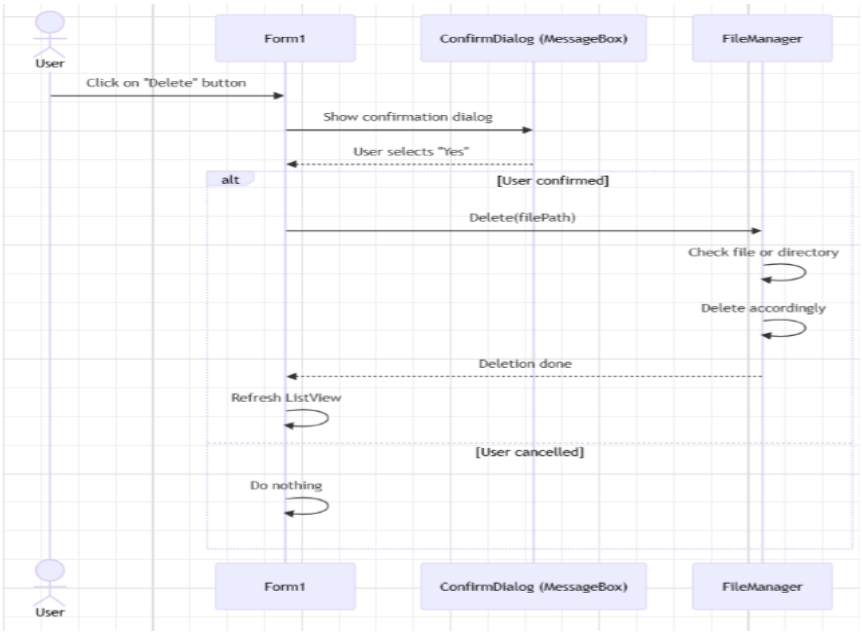
Sequence Diagrams:
Sort diagram:



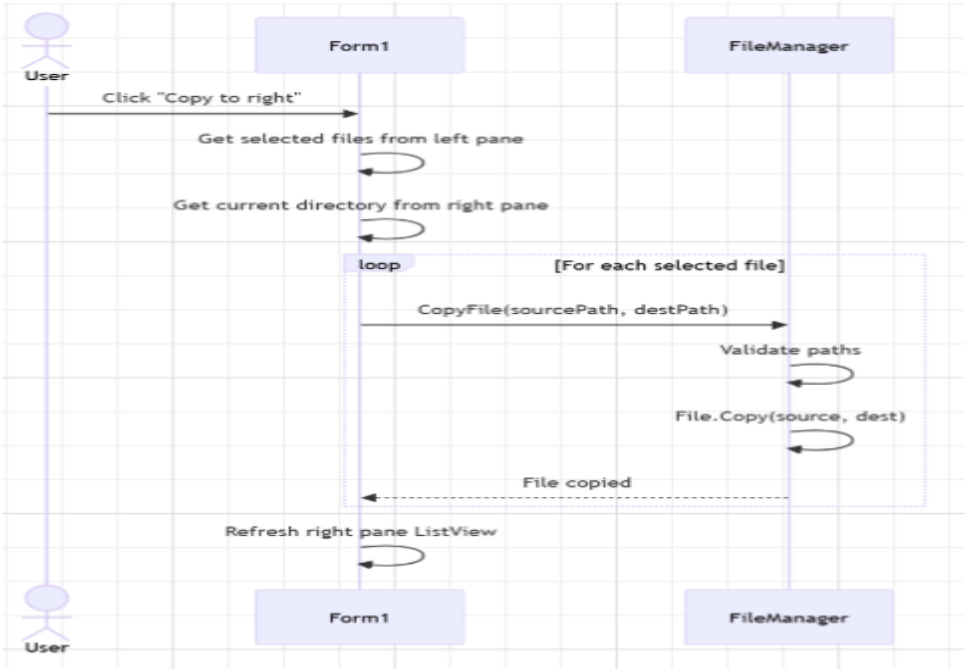
Rename diagram:



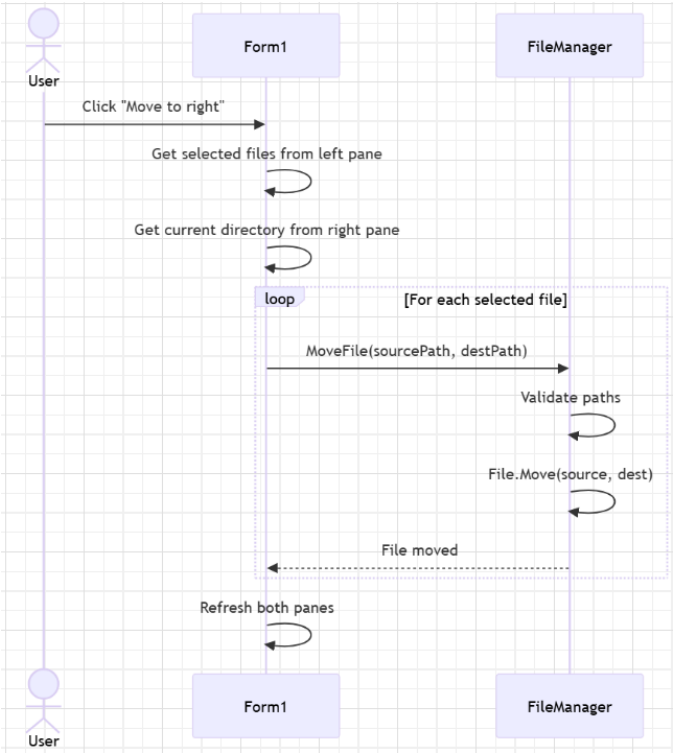
Delete diagram:



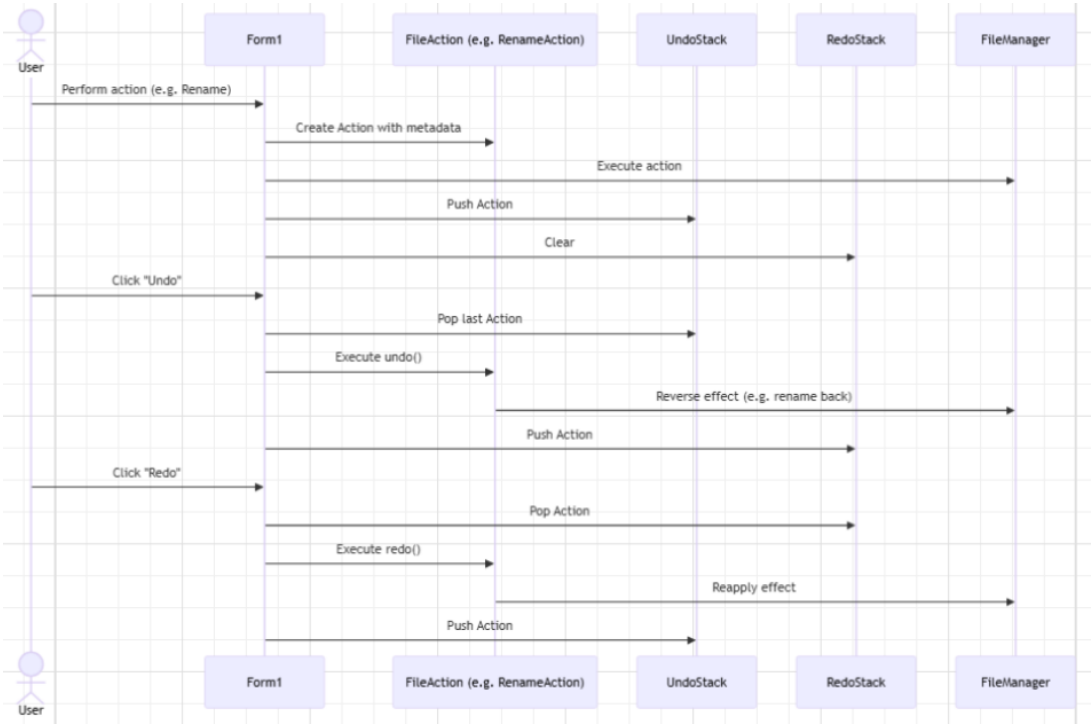
Copy diagram:



Move diagram:

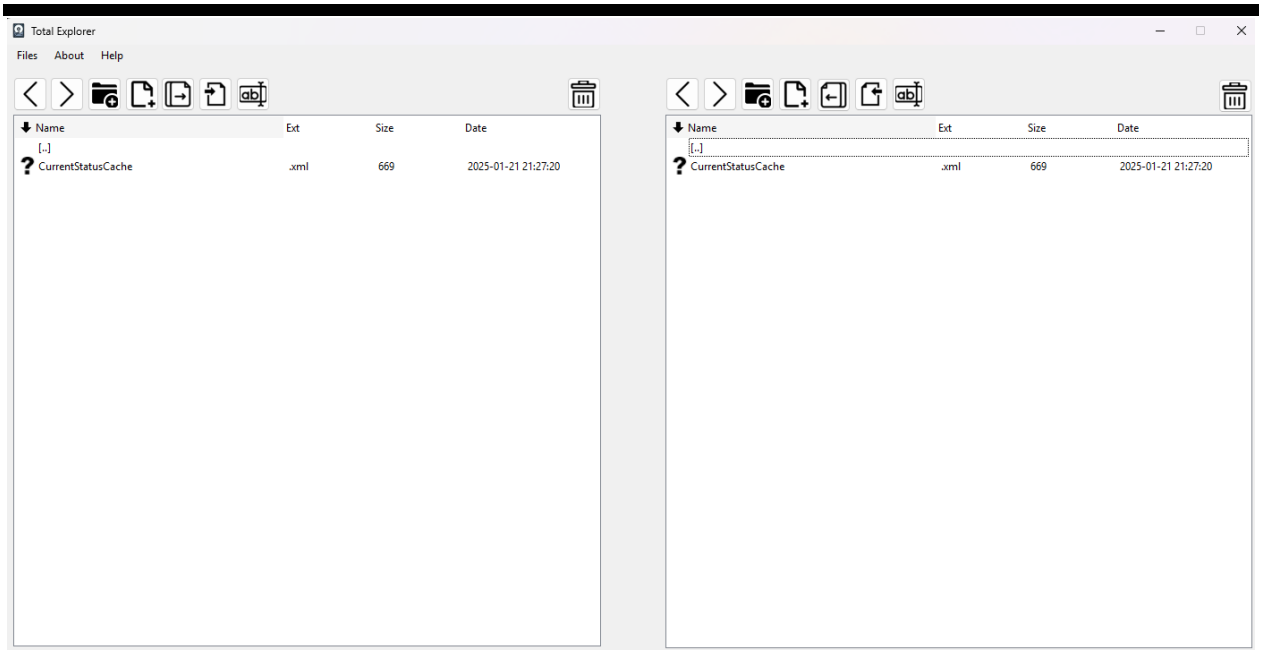


Undo/Redo diagram:



Usage instructions:

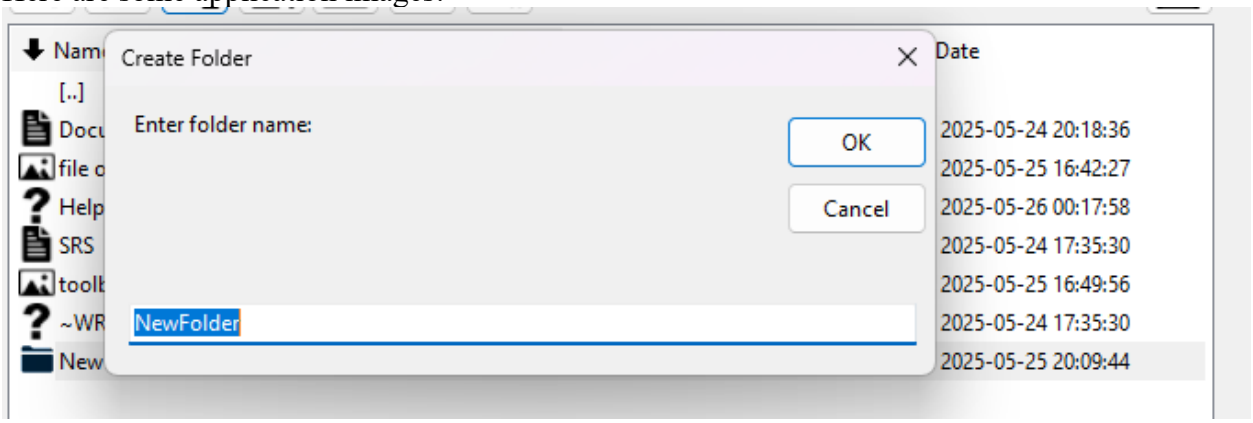
The user has a help file associated with the project. When first opening the app the user will be met with this user interface:

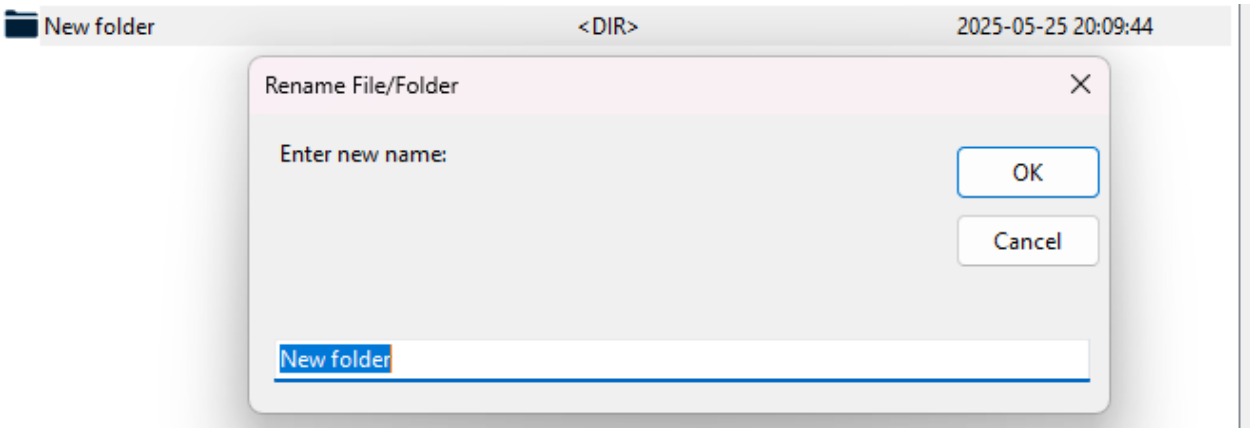


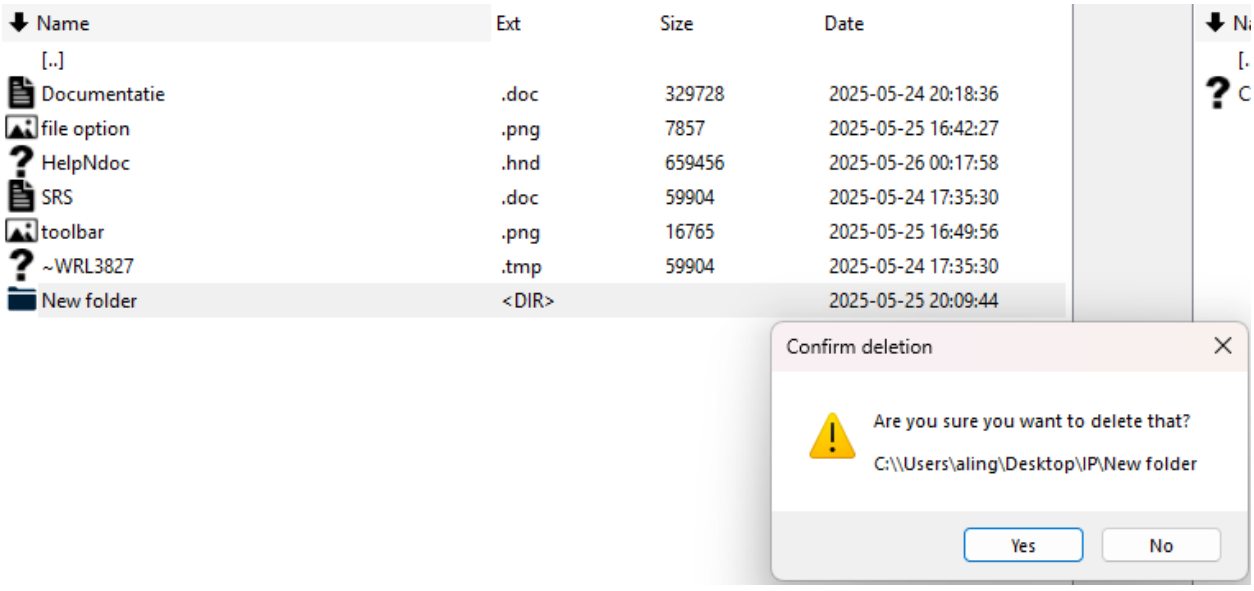
Actions that a user can make:

-Double left click: opens a file or a directory;
-Drag and drop: the user can drag a file from a window to another.
-Backwards and forwards: the user can press the buttons situated over the window in order to go back and forwards.
-Create directory: next to the backards and forward buttons is the create directory button that opens a pop-up box in order to specify the directory name.
-Create file: situated next to the „Create directory” button, will open a pop-up box in order to specify the file name with the extension added.
-Copy file: the next button can copy the item to the right window, if the button on the left is pressed and an item in the left item is selecte. The inverse is applied to the opposite side.
-Move file: the next button can move the item to the right window, if the button on the left is pressed and an item in the left item is selecte. The inverse is applied to the opposite side.
-Rename file: if an item is selected and the button is pressed, a pop-up box will show up in order to rename the file.
-Delete file: situated far from any other buttons, if an item is selected and the user presses the button, the item will be deleted.

Here are some application images:







Unit tests:

Nr.	Method	Input	Expected	Actual	Note
1	GetFilesFromDirectory	folder with 1 file	1 file	1 file	correct
2	GetFileName	C:\folder\test.txt	test.txt	test.txt	correct
3	GetFileExtension	C:\folder\test.pdf	.pdf	.pdf	correct
4	GetFileSize	file with '123'	> 0	> 0	correct
5	GetFileDateTime	existing file	valid date	valid date	correct
6	CreateFile	test.txt	file created	file created	correct
7	CreateFolder	NewFolder	folder created	folder created	correct
8	Delete (file)	existing file	file deleted	file deleted	correct
9	Delete (folder)	existing folder	folder deleted	folder deleted	correct
10	Rename (file)	old.txt -> new.txt	renamed	renamed	correct
11	Rename (folder)	old -> new	renamed	renamed	correct
12	CopyFile	source.txt -> copied.txt	copied	copied	correct
13	MoveFile	source.txt -> moved.txt	moved	moved	correct
14	Delete (non-existent)	notfound.txt	no error	no error	correct
15	CopyFile (invalid)	missing file	throws exception	throws	correct
16	MoveFile (invalid)	missing file	throws exception	throws	correct
17	GetFilesFromDirectory	empty folder	empty list	empty list	correct
18	Rename with extension	test.txt -> renamed.txt	renamed correctly	correct	correct
19	GetFileExtension	nonexistent file	<DIR>	<DIR>	correct
20	GetFileName	C:\temp\demo.docx	demo.docx	demo.docx	correct

Significant snippets of code

File loading:

```
/// <summary>
/// Loads files and directories into the specified ListView panel.
/// </summary>
/// <param name="index">
/// Panel index: 1 for the left panel (listView1), 2 for the right panel (listView2).
/// </param>
/// <remarks>
/// This method:
/// - Fetches all file paths from the current directory using IFileManager.
/// - Creates FileItem objects with name, extension, size, and date.
/// - Displays them in the UI.
/// - Adds error handling to catch and show messages for inaccessible files or folders.
/// </remarks>
/// <exception cref="IOException">
/// Thrown if the directory cannot be accessed.
/// </exception>
/// <exception cref="UnauthorizedAccessException">
/// Thrown if access to the directory or files is denied.
/// </exception>
private void LoadFilesIntoForm(int index)
{
   ListView listView = (index == 1) ? listView1 : listView2;
```

```
List<string> itemNames = new List<string>();
List<FileItem> fileItems;
IFileManager manager;

try
{
    if (index == 1)
    {
        manager = _manager1;
        itemNames = _manager1.GetFilesFromDirectory(_directory1);
        _fileItems1.Clear();
        fileItems = _fileItems1;
    }
    else
    {
        manager = _manager2;
        itemNames = _manager2.GetFilesFromDirectory(_directory2);
        _fileItems2.Clear();
        fileItems = _fileItems2;
    }

    listView.Items.Clear();

    foreach (string item in itemNames)
    {
        try
        {
```

```
            string path = item;
            string name = manager.GetFileName(path);
            string extension = manager.GetFileExtension(path);
            string size = manager.GetFileSize(path);
            string date = manager.GetFileDateTime(path);
            fileItems.Add(new FileItem(name, extension, size, date));

            ListViewItem listItem = new ListViewItem(name);
            listItem.SubItems.Add(extension);
            listItem.SubItems.Add(size);
            listItem.SubItems.Add(date);
            listView.Items.Add(listItem);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error reading item: {item}\n\n{ex.Message}", "Warning",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }

    RefreshList(index);
  }
  catch (Exception ex)
  {
    MessageBox.Show($"Error loading directory contents.\n\n{ex.Message}", "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
  }
}
```

File opening:

```
/// <summary>
/// Handles the double-click event on items in the left ListView (listView1).
/// </summary>
/// <param name="sender">The ListView control that received the double-click.</param>
/// <param name="e">Mouse event data.</param>
/// <remarks>
/// - If the user double-clicks "[..]", navigates one level up in the directory hierarchy.
/// - If the item is a directory ("&lt;DIR&gt;"), navigates into that directory and updates history.
/// - If the item is a file, attempts to open it using the default system application.
/// - Displays a warning message if an error occurs during navigation or file opening.
/// </remarks>
/// <exception cref="Exception">
/// Caught and displayed if directory traversal or file execution fails unexpectedly.
/// </exception>
private void listView1_MouseDoubleClick(object sender, MouseEventArgs e)
{
   try
   {
      if (listView1.SelectedItems.Count > 0)
      {
         if (listView1.SelectedItems[0].Index == 0 && listView1.SelectedItems[0].Text == "[..]")
         {
            int lastIndex = _directory1.LastIndexOf("\\");
            if (lastIndex > 2)
               _directory1 = _directory1.Substring(0, lastIndex);
            _history1.AddAction(_directory1);
```

```
            LoadFilesIntoForm(1);

            return;

        }

        if (listView1.SelectedItems[0].SubItems[1].Text == "<DIR>")

        {

            _directory1 += "\\" + listView1.SelectedItems[0].Text;

            _history1.AddAction(_directory1);

            LoadFilesIntoForm(1);

            RefreshList(1);

        }

        else

        {

            string filePath = _directory1 + "\\" + listView1.SelectedItems[0].Text +
listView1.SelectedItems[0].SubItems[1].Text;

            if (File.Exists(filePath))

            {

                Process.Start(new ProcessStartInfo(filePath) { UseShellExecute = true });

            }}}}

    catch (Exception ex)

    {

        MessageBox.Show($"Error: {ex.Message}", "Warning", MessageBoxButtons.OK,
MessageBoxIcon.Warning);

    }

}
```

Software Requirements Specification

for

Total Explorer

Version 1.0 approved

Prepared by Timofte Alin-Gabriel
Paladi Andrei
Rață Mihai-Gabriel
Ciobanu Andrei

24.05.2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Beta Total Explorer	24-05-25	Putting together all sources	1.0

1. Introduction

1.1 Purpose

Total Explorer is a file management application inspired by Total Commander and the default Windows File Explorer, it offers a dual pane interface for file navigation and easy file drag-and-drop action. The program offers actions like redo, undo, rename, delete and open. The program also offers extra information on file's data, such as last edit date, extension type and size.

1.2 Document Conventions

This document follows the IEEE standard formatting for software development. The standard defines a regular formatting this document follows including writing to be done in third-person, passive voice as well as readable and grammatically correct text.

1.3 Intended Audience and Reading Suggestions

This document is not intended for the end user because it provides a detailed specification of how the software is to be implemented. Since a user needs information on how to use the software and not how to implement it, this document is intended more to testers and software developers of the program. The document starts off with an overview of the functions and specifications for this application in section 2, then moves on to describe the requirements for interfacing with external hardware and software in section 3. Section 4 describes the program's functions in great detail and section 5 lists various requirements the program comply to after completion. It is suggested that all audiences of this document start with section 2 first to get a general idea of the software requirements. Testers should next read sections 5.1 through 5.4. This is to get an idea of how the program will affect them and the system they are running it on, as well as the aspirations for quality. Next a tester should read section 3.1 (user interfaces) followed by all of section 4 (system features). Reading the document in this order will give the tester an idea of what to expect in the interface at first glance, and then they may test all the individual functions to make sure they adhere to the specifications. After reading section 2, software developers should read the remaining sections in order because this document was designed specifically for the purpose of developing the application. The developer needs to get an overall idea in section 2. Then, how it needs to interface with everything else in section 3. Section 4 is the most important to a developer because it describes all the functions in great detail and it will help with making decisions in writing actual code for the application. Section 5 is considered least important but the developer should still read it to make sure the program still meets the requirements.

1.4 References

https://learn.microsoft.com/en-us/dotnet/
https://learn.microsoft.com/en-us/dotnet/csharp/

2. Overall Description

2.1 Product Perspective

The Total Explorer application is designed to operate as a standalone software application, providing a separate platform for file management and control. It offers many capabilities, which will be described further into the document, and it also provides a familiar interface intended to help users quickly adapt to the new environment. It is a C# implementation of a file management application inspired by Total Commander and File Explorer.

2.2 Product Functions

File opening and updating: it allows users to open a file with their default application, using the default windows libraries.
File listing and ordering: it allows users to see the specific files in a directory, and also allows users to sort the files by date, name, size, or extension.
File deleting: users can delete any file that is unrestricted to them. If multiple files are selected, a pop-up will show up.
Undo-Redo history: the application is able to go back and forward on operations.
File and directory creation: the program can create new files and directories.
Drag and drop: files can be moved between the 2 windows.

2.3 User Classes and Characteristics

Developers are the main users that need to implement the app's functionalities.
Testers should also play a role in ensuring the quality and the functionality of the application. They have knowledge of testing methods and test case creation.
The application is also intended to be used by any regular user, meaning that the application should be designed for ease of use. The interface should be simple and descriptive of their actions, including appropriate icons.

2.4 Operating Environment

Total Explorer is being developed in the C# environment. Users should use a Windows system, since it is a windows forms application. The application will be designed to work on any x86 and x64 system.

2.5 Design and Implementation Constraints

This application will require the 8.0 .NET Framework version in order to run. The limiting factor of this project is time, because it is a single semester project. Time restraints may require this team to scale back the project.

2.6 User Documentation

The file manager application will feature a built-in user manual, by clicking the help button. By clicking the button, it will open content-specific and relevant information related to all actions that the application is capable of. Users will also be informed in the application if an error has occurred, and the cause of it.

2.7 Assumptions and Dependencies

It is assumed that the user is familiar with either one of the two programs that inspired the project, and that the user has basic computer skills.
Users are assumed to have a Windows operating system, such as Windows 11, Windows 10, or Windows 7.
The project can be scaled down if the resources available are scarce, resources such as time, budget and human resources.
It is assumed that users can also understand how the Windows file system works, meaning that they know how to navigate a file system at least.

3. External Interface Requirements

3.1 User Interfaces

The software product will feature a graphical user interface (GUI) that enables users to interact with file system elements in a way similar to file managers such as File Explorer and Total Commander. The user interface will be intuitive and consistent, following a unified layout structure and interaction model. The application will consist of a dual-pane interface that allows users to browse, compare, and manage files and directories efficiently. Each pane will support independent navigation and will display a list of files and folders along with their attributes, such as name, size, type, and date modified.
The interface will present standard GUI for desktop applications. Common interface elements such as toolbars, menus, and context menus will be implemented to provide access to core functionalities including copy, move, delete, rename, and create folder. A consistent set of buttons such as “Files”, "Help" and “About” will appear on every screen. Keyboard shortcuts will be supported for common tasks, enhancing accessibility and improving user efficiency.
Error messages and system notifications will follow a standardized format to ensure clarity and consistency. Messages will inform users about the issue and suggest possible actions or solutions. Pop-up dialogs will be used to confirm potentially destructive operations, such as file deletion, and to handle error conditions gracefully.
The software components requiring user interface design include the main file browsing interface, file operation dialogs, and any modal windows used for interactions such as renaming files or selecting destination directories.

3.2 Hardware Interfaces

The application is designed to work with standard input devices, the mouse and keyboard. Users can interact using the mouse and multiple clicks, and also with keyboard shortcuts or key binds, such as F8 for deletion. The application does not require any specialized hardware.

3.3 Software Interfaces

The Total Explorer application is designed to work with libraries from the operating system that allows the program to access and gather information on the files which will be displayed on the screen.

3.4 Communications Interfaces

This application does not require any communication with the internet or any device. It is a solely offline application that will access local files in order to manipulate data from the local storage.

4. System Features

The system features are covered in-depth in a separate documentation that has use case scenarios and UML diagrams to better understand the inner-workings.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

The application must start in under 3 seconds on systems with at least a 2.0 GHz dual-core processor and 4 GB of RAM. Directory listings of up to 1,000 items must load in under 1 second. File operations such as copy, move, or delete should begin execution within 500 milliseconds of user action, depending on hardware.

5.2 Safety Requirements

If the user is not careful, he may delete his files. The program protects users from deleting protected files but not all files are protected.

5.3 Security Requirements

The application will not gather any private information, because the app does not connect to any server.

5.4 Software Quality Attributes

This software must be robust and as bug-free as possible to ensure the users have a positive experience. The program should be easy for a beginner to pick up and get started, with a minimal learning curve. The application should be flexible enough to allow the easy creation of additional content, while preserving the ease of use to the consumer. The app emphasizes reliability, ensuring consistent and error-free performance, it is designed to handle unexpected scenarios gracefully, maintaining stability and providing a reliable platform for users to manage their tasks without disruptions.

5.5 Business Rules

It is the policy of the development team to follow all codes of conduct established by the University.

6. Other Requirements

Appendix A: Glossary

N/A

Appendix B: Analysis Models

A class diagram is available in the documentation.

Appendix C: To Be Determined List

N/A