| | |
|---|---|
| Project Name | Virtual Key for Your Repositories |
| Developers Name | Prathamesh Chinchamalatpure |

This document contains the following contents:

1. Sprint Planned and task achieved in them
2. Algorithm and flowcharts of the application
3. Core concepts used in the project
4. Links to the GitHub repository
5. Unique Selling Points
6. Conclusions

This project is hosted at https://github.com/prathameshcgit/Phase2ProjectDemo.git

The project is developed by Prathamesh Chinchamalatpure

**1.Sprint Planning and Task Achieved in them**

This project is planned to be completed in 3 sprint. Task that are assumed to be completed are:

1 Developing flow for project

2 Initializing git repositories to track changes as project progresses

3 Writing Java code to fulfill requirement for the project

4 Testing project by giving diverse input

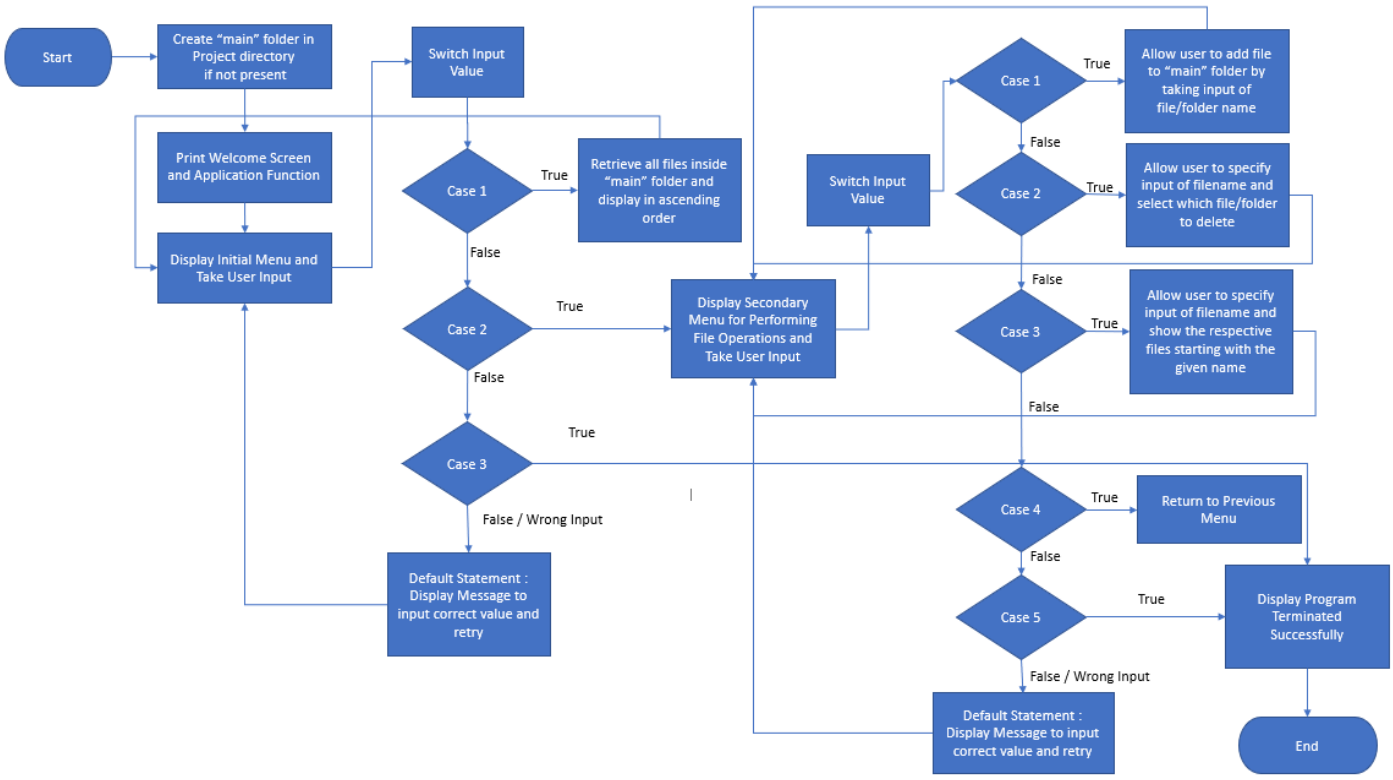
5 Pushing code to github

6 Creating document of specification highlighting application appearances , capabilities and user interactions

**2. Core concepts required in project**

String, Collection Framework, File Handling, Control constructs, Sorting ,Recursion, Exception Handling

### 3.Flow for project




### 4 . Product capabilities, appearance and user interactions

To demonstrate the product capabilities, below are the sub-sections configured to highlight appearance and user interactions for the project:

1 Creating the project in Eclipse

2 Writing a program in Java for the entry point of the application (LockedMeMain.java)

3 Writing a program in Java to display Menu options available for the user (MenuOptions.java)

4 Writing a program in Java to handle Menu options selected by user (HandleOptions.java)

5 Writing a program in Java to perform the File operations as specified by user (FileOperations.java)

6 Pushing the code to GitHub repository

### 1 Creating the project in Eclipse

1. Open Eclipse.
2. File->New->Project->Maven->Maven Project
3. Maven Project-> select archetype->Enter GroupId , ArtifactId
4. Click Finish.
5. Create package and create class

6. While creating class check the checkbox for public static void main(String[] args) And click on “Finish”.

**2 Writing a program in Java for the entry point of the application (LockedMeMain.java)**

```
package com.virtualkeyrepository;

public class LockedMeMain {

	public static void main(String[] args) {

	  FileOperations.createMainFolderIfNotPresent("Main");

	 MenuOptions.printWelcomeScreen("LockedMe","Prathamesh Chinchamalatpure");

	 HandleOptions.handleWelcomeScreenInput();

	}

}
```

**3 Writing a program in Java to display Menu options available for the user (MenuOptions.java)**

a.Create MenuOptions class in eclipse

b.Write a code in it to display menu in terms of following three parts.

1. Displaying Welcome Screen

2. Displaying Initial Menu

3. Displaying Secondary Menu for File Operations available

1.Displaying Welcome Screen

```
public static void printWelcomeScreen(String appName,String developerName) {

		String companyDetails=String.format("**********************************************************\n  Welcome to %s.com \n"+"  This application is developed by %s \n**********************************************************",appName,developerName);

		String appFunction="You can use this application to :-\n"
				+"Retrieve all file names in the Main folder \n"
				+"Search, Add , Delete files in main folder \n"
				+"\n**Kindly be careful to ensure the correct filename is provided for searching or deleting files.**\n";

		System.out.println(companyDetails);
		System.out.println(appFunction);


	}
```

**O/P:**

```
************************************************************
  Welcome to LockedMe.com
  This application is developed by Prathamesh Chinchamalatpure
************************************************************
You can use this application to :-
Retrieve all file names in the Main folder
Search, Add , Delete files in main folder

**Kindly be careful to ensure the correct filename is provided for searching or
deleting files.**
```

2. Displaying Initial Menu

```
public static void displayMenu() {
            String menu = "\n\n****** Select any option number from below and
press Enter ******\n\n"
            + "1) Retrieve all files inside \"main\" folder\n" + "2) Display
menu for File operations\n"
            + "3) Exit program\n";
            System.out.println(menu);
            }
```

**O/P:**

```
****** Select any option number from below and press Enter ******

1) Retrieve all files inside "main" folder
2) Display menu for File operations
3) Exit program
```

3. Displaying Secondary Menu for File Operations available

```
public static void displayFileMenuOptions() {
            String fileMenu = "\n\n****** Select any option number from below
and press Enter ******\n\n"
            + "1) Add a file to \"main\" folder\n" + "2) Delete a file from
\"main\" folder\n"
            + "3) Search for a file from \"main\" folder\n" + "4) Show Previous
Menu\n" + "5) Exit program\n";
            System.out.println(fileMenu);
      }
```

**O/P:**

```
****** Select any option number from below and press Enter ******

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program
```

**4 Writing a program in Java to handle Menu options selected by user (HandleOptions.java)**

1.Open Eclipse and create class HandleOptions.java

2.HandleOptions class contains two methods

a. `handleWelcomeScreenInput to handle primary menus`

`b. handleFileMenuOptions to handle secondary menus`

1. handleWelcomeScreenInput to handle primary menus

```
public static void handleWelcomeScreenInput() {
		boolean running = true;
		Scanner sc = new Scanner(System.in);
		do {
			try {
				MenuOptions.displayMenu();
				int input = sc.nextInt();
				switch (input) {
				case 1:
					FileOperations.displayAllFiles("Main");
					break;
				case 2:
					HandleOptions.handleFileMenuOptions();
					break;
				case 3:
					System.out.println("Program exited
successfully.");
					running = false;
					sc.close();
					System.exit(0);
					break;
				default:
					System.out.println("Please select a valid option
from  above.");
				}
			} catch (Exception e) {
				System.out.println(e.getClass().getName());
				handleWelcomeScreenInput();
			}
		} while (running == true);
	}
```

**O/P:**

```
****** Select any option number from below and press Enter ******

1) Retrieve all files inside "main" folder
2) Display menu for File operations
```

```
3) Exit program

1
Displaying all files with directory structure in ascending order

|-- Empty Directory

Displaying all files in ascending order



****** Select any option number from below and press Enter ******

1) Retrieve all files inside "main" folder
2) Display menu for File operations
3) Exit program

1
Displaying all files with directory structure in ascending order

|-- a.txt
|-- b.txt

Displaying all files in ascending order

a.txt
b.txt
```

## 2. handleFileMenuOptions to handle secondary menus

```
public static void handleFileMenuOptions() {
		boolean running = true;
		Scanner sc = new Scanner(System.in);
		do {
			try {
				MenuOptions.displayFileMenuOptions();
				FileOperations.createMainFolderIfNotPresent("main");
				int input = sc.nextInt();
				switch (input) {
				case 1:
					// File Add
					System.out.println("Enter the name of the file to
be added to the \"main\" folder");
					String fileToAdd = sc.next();
					FileOperations.createFile(fileToAdd, sc);
					break;
				case 2:
					// File/Folder delete
					System.out.println("Enter the name of the file to
be deleted from \"main\" folder");
					String fileToDelete = sc.next();

	FileOperations.createMainFolderIfNotPresent("main");
					List<String> filesToDelete =

	FileOperations.displayFileLocations(fileToDelete, "main");
					String deletionPrompt = "\nSelect index of which
file to delete?"
```

```
                                                + "\n(Enter 0 if you want to delete
all elements)";
                                    System.out.println(deletionPrompt);
                                    int idx = sc.nextInt();
                                    if (idx != 0) {

            FileOperations.deleteFileRecursively(filesToDelete.get(idx - 1));
                                    } else {
                                          // If idx == 0, delete all files displayed
for the name
                                          for (String path : filesToDelete) {

            FileOperations.deleteFileRecursively(path);
                                          }
                                    }
                                    break;
                              case 3:
                                    // File/Folder Search
                                    System.out.println("Enter the name of the file to
be searched from \"main\" folder");
                                    String fileName = sc.next();

            FileOperations.createMainFolderIfNotPresent("main");
                                    FileOperations.displayFileLocations(fileName,
"main");
                                    break;
                              case 4:
                                    // Go to Previous menu
                                    return;
                              case 5:
                                    // Exit
                                    System.out.println("Program exited
successfully.");
                                    running = false;
                                    sc.close();
                                    System.exit(0);
                              default:
                                    System.out.println("Please select a valid option
from above.");
                              }
                        } catch (Exception e) {
                              System.out.println(e.getClass().getName());
                              handleFileMenuOptions();
                        }
                  } while (running == true);
            }
}
```

**O/P:**

```
****** Select any option number from below and press Enter ******

1) Retrieve all files inside "main" folder
2) Display menu for File operations
3) Exit program

2
```

```
****** Select any option number from below and press Enter ******

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

1
Enter the name of the file to be added to the "main" folder
c.txt
c.txt created successfully
Would you like to add some content to the file? (Y/N)
Y


Input content and press enter

ACX is a car

Content written to file c.txt
Content can be read using Notepad or Notepad++


****** Select any option number from below and press Enter ******

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

2
Enter the name of the file to be deleted from "main" folder
c.txt


Found file at below location(s):
1: C:\Users\Prathamesh\SProjectPhase2\VirtualKeyForRepository\main\c.txt

Select index of which file to delete?
(Enter 0 if you want to delete all elements)
0
c.txt at C:\Users\Prathamesh\SProjectPhase2\VirtualKeyForRepository\main deleted
successfully


****** Select any option number from below and press Enter ******

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

3
Enter the name of the file to be searched from "main" folder
b.txt
```

```
Found file at below location(s):
1: C:\Users\Prathamesh\SProjectPhase2\VirtualKeyForRepository\main\b.txt


****** Select any option number from below and press Enter ******

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

3
Enter the name of the file to be searched from "main" folder
c.txt


***** Couldn't find any file with given file name "c.txt" *****




****** Select any option number from below and press Enter ******

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

4


****** Select any option number from below and press Enter ******

1) Retrieve all files inside "main" folder
2) Display menu for File operations
3) Exit program
```

**5 Writing a program in Java to perform the File operations as specified by user (FileOperations.java)**

1.Create a FileOperations class

2. FileOperations consists methods for

a. Creating “main” folder in project if it’s not already present

b. Displaying all files in “main” folder in ascending order and also with directory structure.

c. Creating a file/folder as specified by user input.

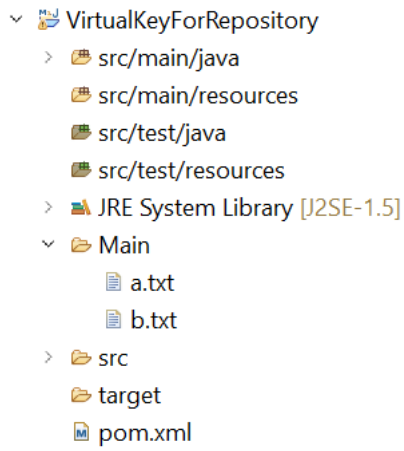
d. Search files as specified by user input in “main” folder and it’s subfolders.

e. Deleting a file/folder from “main” folder

a. Creating “main” folder in project if it’s not already present

```
public static void createMainFolderIfNotPresent(String folderName) {
		File file=new File(folderName);

		if(!file.exists()) {
			file.mkdir();
		}
	}
```

**O/P:**




b. Displaying all files in “main” folder in ascending order and also with directory structure.

```
public static void displayAllFiles(String path) {
		FileOperations.createMainFolderIfNotPresent("Main");
		// All required files and folders inside "main" folder relative to
current
		// folder
		System.out.println("Displaying all files with directory structure in
ascending order\n");
		// listFilesInDirectory displays files along with folder structure
		List<String> filesListNames =
FileOperations.listFilesInDirectory(path, 0, new ArrayList<String>());
		System.out.println("Displaying all files in ascending order\n");
		Collections.sort(filesListNames);
		filesListNames.stream().forEach(System.out::println);

	}



public static List<String> listFilesInDirectory(String path, int indentationCount,
			List<String> fileListNames) {
		File dir = new File(path);
		File[] files = dir.listFiles();
		List<File> filesList = Arrays.asList(files);
```

```
            Collections.sort(filesList);
            if (files != null && files.length > 0) {
                  for (File file : filesList) {
                        System.out.print(" ".repeat(indentationCount * 2));
                        if (file.isDirectory()) {
                              System.out.println("`-- " + file.getName());
                              // Recursively indent and display the files
                              fileListNames.add(file.getName());
                              listFilesInDirectory(file.getAbsolutePath(),
indentationCount
                                          + 1, fileListNames);
                        } else {
                              System.out.println("|-- " + file.getName());
                              fileListNames.add(file.getName());
                        }
                  }
            } else {
                  System.out.print(" ".repeat(indentationCount * 2));
                  System.out.println("|-- Empty Directory");
            }
            System.out.println();
            return fileListNames;
      }
```

**O/P :**

```
****** Select any option number from below and press Enter ******

1) Retrieve all files inside "main" folder
2) Display menu for File operations
3) Exit program

1
Displaying all files with directory structure in ascending order

|-- Empty Directory

Displaying all files in ascending order



****** Select any option number from below and press Enter ******

1) Retrieve all files inside "main" folder
2) Display menu for File operations
3) Exit program

1
Displaying all files with directory structure in ascending order

|-- a.txt
|-- b.txt

Displaying all files in ascending order

a.txt
b.txt
```

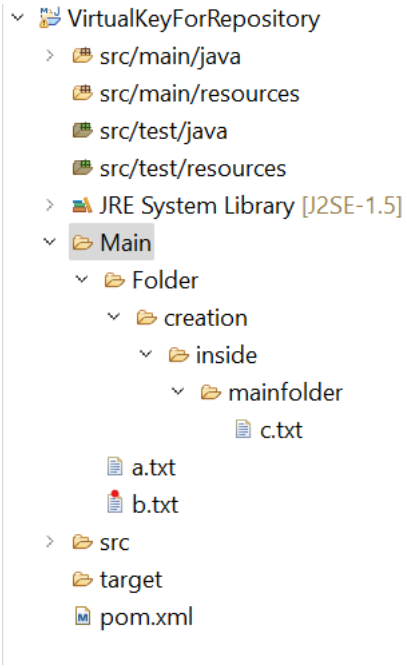
c. Creating a file/folder as specified by user input.

```
public static void createFile(String fileToAdd, Scanner sc) {
		FileOperations.createMainFolderIfNotPresent("main");
		Path pathToFile = Paths.get("./main/" + fileToAdd);
		try {
			Files.createDirectories(pathToFile.getParent());
			Files.createFile(pathToFile);
			System.out.println(fileToAdd + " created successfully");
			System.out.println("Would you like to add some content to the
file? (Y/N)");
			String choice = sc.next().toLowerCase();
			sc.nextLine();
			if (choice.equals("y")) {
				System.out.println("\n\nInput content and press
enter\n");
				String content = sc.nextLine();
				Files.write(pathToFile, content.getBytes());
				System.out.println("\nContent written to file " +
fileToAdd);
				System.out.println("Content can be read using Notepad
or Notepad++");
			}
		} catch (IOException e) {
			System.out.println("Failed to create file " + fileToAdd);
			System.out.println(e.getClass().getName());
		}
	}
```

**O/P:**

```
****** Select any option number from below and press Enter ******

1) Retrieve all files inside "main" folder
2) Display menu for File operations
3) Exit program

2


****** Select any option number from below and press Enter ******

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

1
Enter the name of the file to be added to the "main" folder
Folder/creation/inside/mainfolder/c.txt
Folder/creation/inside/mainfolder/c.txt created successfully
```

```
Would you like to add some content to the file? (Y/N)
Y


Input content and press enter

Hello you are in c.txt

Content written to file Folder/creation/inside/mainfolder/c.txt
Content can be read using Notepad or Notepad++
```




d. Search files as specified by user input in “main” folder and it’s subfolders.

```
	public static List<String> displayFileLocations(String fileName, String
path) {
		List<String> fileListNames = new ArrayList<>();
		FileOperations.searchFileRecursively(path, fileName, fileListNames);
		if (fileListNames.isEmpty()) {
			System.out.println("\n\n***** Couldn't find any file with
given file name \""
					+ fileName + "\" *****\n\n");
		} else {
			System.out.println("\n\nFound file at below location(s):");
			List<String> files = IntStream.range(0, fileListNames.size())
					.mapToObj(index -> (index + 1) + ": " +

	fileListNames.get(index)).collect(Collectors.toList());
			files.forEach(System.out::println);
		}
		return fileListNames;
	}
```

```
	public static void searchFileRecursively(String path, String fileName,
List<String>
	fileListNames) {
		File dir = new File(path);
		File[] files = dir.listFiles();
		List<File> filesList = Arrays.asList(files);
		if (files != null && files.length > 0) {
			for (File file : filesList) {
				if (file.getName().startsWith(fileName)) {
					fileListNames.add(file.getAbsolutePath());
				}
				// Need to search in directories separately to ensure
all files of required
				// fileName are searched
				if (file.isDirectory()) {
					searchFileRecursively(file.getAbsolutePath(),
fileName,
							fileListNames);
				}
			}
		}
	}
```

**O/P:**

```
****** Select any option number from below and press Enter ******

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

3
Enter the name of the file to be searched from "main" folder
c


Found file at below location(s):
1: C:\Users\Prathamesh\SProjectPhase2\VirtualKeyForRepository\main\c
2: C:\Users\Prathamesh\SProjectPhase2\VirtualKeyForRepository\main\Folder\creation
3:
C:\Users\Prathamesh\SProjectPhase2\VirtualKeyForRepository\main\Folder\creation\in
side\mainfolder\c.txt
```

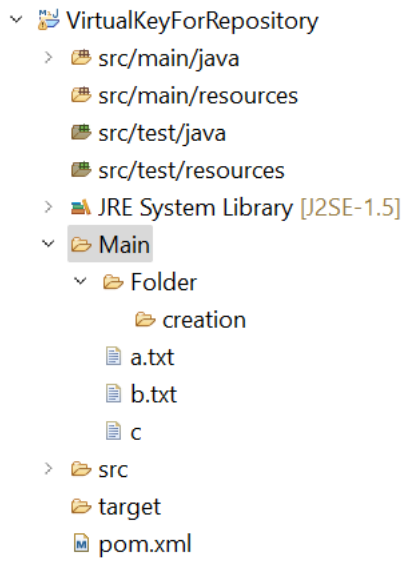
e. Deleting a file/folder from “main” folder

```
public static void deleteFileRecursively(String path) {
		File currFile = new File(path);
		File[] files = currFile.listFiles();
		if (files != null && files.length > 0) {
			for (File file : files) {
```

```
                        String fileName = file.getName() + " at " +
file.getParent();
                        if (file.isDirectory()) {
                                deleteFileRecursively(file.getAbsolutePath());
                        }
                        if (file.delete()) {
                                System.out.println(fileName + " deleted
successfully");
                        } else {
                                System.out.println("Failed to delete " +
fileName);
                        }
                }
        }
        String currFileName = currFile.getName() + " at " +
currFile.getParent();
        if (currFile.delete()) {
                System.out.println(currFileName + " deleted successfully");
        } else {
                System.out.println("Failed to delete " + currFileName);
        }
    }
```

## O/P:

```
****** Select any option number from below and press Enter ******

1) Add a file to "main" folder
2) Delete a file from "main" folder
3) Search for a file from "main" folder
4) Show Previous Menu
5) Exit program

2
Enter the name of the file to be deleted from "main" folder
inside


Found file at below location(s):
1: C:\Users\Prathamesh\SProjectPhase2\VirtualKeyForRepository\main\Folder\creation\inside

Select index of which file to delete?
(Enter 0 if you want to delete all elements)
1
c.txt at
C:\Users\Prathamesh\SProjectPhase2\VirtualKeyForRepository\main\Folder\creation\inside\mainfolder
deleted successfully
mainfolder at C:\Users\Prathamesh\SProjectPhase2\VirtualKeyForRepository\main\Folder\creation\inside
deleted successfully
Failed to delete mainfolder at
C:\Users\Prathamesh\SProjectPhase2\VirtualKeyForRepository\main\Folder\creation\inside
inside at C:\Users\Prathamesh\SProjectPhase2\VirtualKeyForRepository\main\Folder\creation deleted
successfully
```




6. **Pushing the code to GitHub repository**

1. Open your command prompt and navigate to the folder where you have created your files.

2. Initialize repository using the following command: **git init**

3. Add all the files to your git repository using the following command: **git add .**

4. Commit the changes using the following command: **git commit . -m**

5. Push the files to the folder you initially created using the following command:

   **git push -u origin master**

## Unique Selling Points of the Application

1 The application is designed to keep on running and taking user inputs even after exceptions occur. To terminate the application, appropriate option needs to be selected.

2 The application can take any file/folder name as input. Even if the user wants to create nested folder structure, user can specify the relative path, and the application takes care of creating the required folder structure.

3 User is also provided the option to write content if they want into the newly created file.

4 The application doesn't restrict user to specify the exact filename to search/delete file/folder. They can specify the starting input, and the program searches all files/folder

starting with the value and displays it. The user is then provided the option to select all files or to select a specific index to delete.

5 The application also allows user to delete folders which are not empty.

6 The user is able to seamlessly switch between options or return to previous menu even after any required operation like adding, searching, deleting or retrieving of files is performed.

7 When the option to retrieve files in ascending order is selected, user is displayed with two options of viewing the files.
   7.1 Ascending order of folders first which have files sorted in them,
   7.2 Ascending order of all files and folders inside the “main” folder.

## Conclusions

1. Further enhancements to the application can be made which may include:
2. Conditions to check if user is allowed to delete the file or add the file at the specific locations.
3. Asking user to verify if they really want to delete the selected directory if it’s not empty.
4. Retrieving files/folders by different criteria like Last Modified, Type, etc.
5. Allowing user to append data to the file