LockedMe Project Virtual Keys

This document contains sections for:

The code for this project is hosted at
https://github.com/Prasannasyam/LockedMe.git

The project is developed by Prasanna kumari.

Sprints planning and Task completion

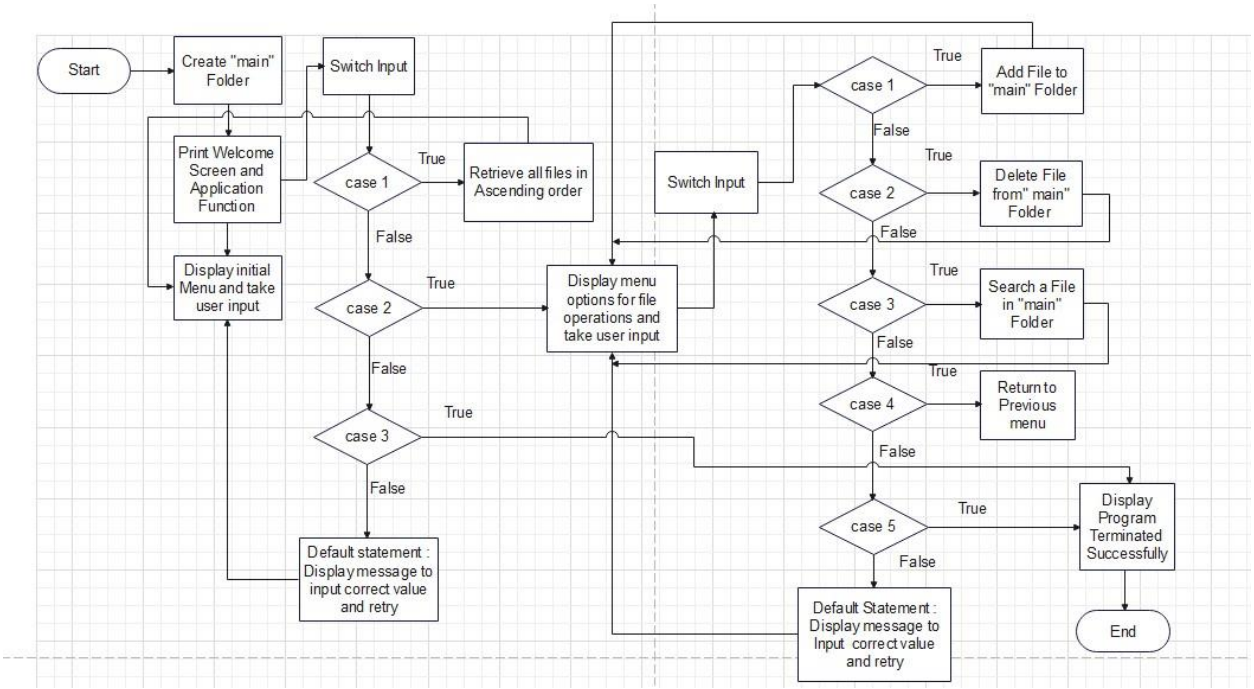
The project is planned to be completed in 1 sprint. Tasks assumed to be completed in the sprint are:

- Creating the flow of the application
- Initializing git repository to track changes as development progresses.
- Writing the Java program to fulfill the requirements of the project.
- Testing the Java program with different kinds of User input
- Pushing code to GitHub.
- Creating this specification document highlighting application capabilities, appearance, and user interactions.

Core concepts used in project

Collections framework, File Handling, Sorting, Flow Control, Recursion, Exception Handling, Streams API

Flow of the Application



Demonstrating the 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

Step 1: Creating a new project in Eclipse

- Open Eclipse
- Go to File -> New -> Project -> Java Project -> Next.
- Type in any project name and click on “Finish.”
- Select your project and go to File -> New -> Class.
- Enter **LockedMeMain** in any class name, check the checkbox “public static void main(String[] args)”, and click on “Finish.”

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

```java
package com.lockedme;

public class LockedMeMain {

    public static void main(String[] args) {
        FileOperations.createMainFolder("main");
        MenuOptions.printWelcomeScreen("LockedMe", "Prasanna kumari Chitta.");
        HandleOptions.WelcomeScreen();
    }
}

```

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

- Select your project and go to File -> New -> Class.
- Enter **MenuOptions** in class name and click on “Finish.”
- **MenuOptions** consists methods for -:

3.1. Displaying Welcome Screen

3.2. Displaying Initial Menu

3.3. Displaying Secondary Menu for File Operations available

Step 3.1: Writing methods to display Welcome Screen.

```java
package com.lockedme;

public class MenuOptions {

    public static void printWelcomeScreen(String appName, String developerName) {
        String companyDetails = String.format("-----------------------------------------------------
                + "** Welcome to %s.com. \n" + "** This application was developed by : %s.\n"
                + "-----------------------------------------------------------\n", appName, devel
        String appFunction = "You can use this application to :-\n"
                + "1. Retrieve all file names in the \"main\" folder.\n"
                + "2. Search, add, or delete files in \"main\" folder.\n";
        System.out.println(companyDetails);

        System.out.println(appFunction);
    }
```

Output:

```
-----------------------------------------------------------------
** Welcome to LockedMe.com.
** This application was developed by : Prasanna kumari Chitta..
-----------------------------------------------------------------

You can use this application to :-
1. Retrieve all file names in the "main" folder.
2. Search, add, or delete files in "main" folder.
```

Step 3.2: Writing method to display Initial Menu

```java
public static void displayMenu() {
    String menu = "\n\n-:folder Options:-\n\n"
            + "1. Files inside \"main\" folder.\n" + "2. File Operations.\n"
            + "3. Exit.\n";
    System.out.println(menu);

}
```

Output:

```
-:folder Options:-

1. Files inside "main" folder.
2. File Operations.
3. Exit.
```

Step 3.3: Writing method to display Secondary Menu for File Operations

```
public static void displayFileMenuOptions() {
    String fileMenu = "\n\n-: File Operations:-\n\n"
            + "1. Add file. \n" + "2. Delete file. \n"
            + "3. Search file.\n" + "4. Previous Menu.\n" + "5. Exit.\n";

    System.out.println(fileMenu);
}
```

Output:

```
-: File Operations:-

1. Add file.
2. Delete file.
3. Search file.
4. Previous Menu.
5. Exit.
```

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

- Select your project and go to File -> New -> Class.
- Enter **HandleOptions** in class name and click on "Finish."
- **HandleOptions** consists methods for -:

4.1. Handling input selected by user in initial Menu

4.2. Handling input selected by user in secondary Menu for File Operations

Step 4.1: Writing method to handle user input in initial Menu

```java

import java.util.List;

public class HandleOptions {
    public static void WelcomeScreen() {
        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());
                WelcomeScreen();
            }
        } while (running == true);
    }
```

Output:

```
You can use this application to :-
1. Retrieve all file names in the "main" folder.
2. Search, add, or delete files in "main" folder.



-:folder Options:-

1. Files inside "main" folder.
2. File Operations.
3. Exit.

1
Displaying all files with directory structure in ascending order

|-- eee.txt
|-- kjb.txt
|-- prasannakumari

Displaying all files in ascending order

eee.txt
kjb.txt
prasannakumari
```

Step 4.2: Writing method to handle user input in Secondary Menu for File Operations

```java
    public static void handleFileMenuOptions() {
        boolean running = true;
        Scanner sc = new Scanner(System.in);
        do {
            try {
                MenuOptions.displayFileMenuOptions();
                FileOperations.createMainFolder("main");

                int input = sc.nextInt();
                switch (input) {
                case 1:
                    System.out.println("-:Enter Filename:-");
                    String fileToAdd = sc.next();

                    FileOperations.createFile(fileToAdd, sc);

                    break;
                case 2:
                    System.out.println("-:Enter Filename:-");
                    String fileToDelete = sc.next();

                    FileOperations.createMainFolder("main");
                    List<String> filesToDelete = FileOperations.displayFileLocations(fileToDelete, "main");

                    String deletionPrompt = "\nDo you want to delete?"
                            + "\n(Enter \"0\")";
                    System.out.println(deletionPrompt);

                    int idx = sc.nextInt();

                    if (idx != 0) {
                        FileOperations.deleteFile(filesToDelete.get(idx - 1));
                    } else {
                        for (String path : filesToDelete) {
                            FileOperations.deleteFile(path);
                        }
                    }


                    break;
                case 3:
                    System.out.println("Enter the name of the file to be searched from \"main\" folder");
                    String fileName = sc.next();

                    FileOperations.createMainFolder("main");
                    FileOperations.displayFileLocations(fileName, "main");


                    break;
                case 4:
                    return;
                case 5:
                    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);
    }
}
```

Output:

```
1. Files inside "main" folder.
2. File Operations.
3. Exit.

2


-: File Operations:-

1. Add file.
2. Delete file.
3. Search file.
4. Previous Menu.
5. Exit.

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


Found file at below location(s):
1: C:\Users\91965\Desktop\LockedMe-main\LockedMe-main\main\prasannakumari


-: File Operations:-

1. Add file.
2. Delete file.
3. Search file.
4. Previous Menu.
5. Exit.
```

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

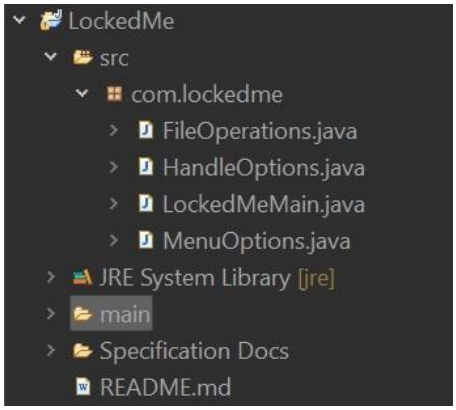
- Select your project and go to File -> New -> Class.
- Enter **FileOperations** in class name and click on "Finish."
- **FileOperations** consists methods for -:

5.1. Creating "main" folder in project if it's not already present
5.2. Displaying all files in "main" folder in ascending order and also with directory structure.
5.3. Creating a file/folder as specified by user input.
5.4. Search files as specified by user input in "main" folder and it's subfolders.
5.5. Deleting a file/folder from "main" folder

Step 5.1: Writing method to create "main" folder in project if it's not present

```
18    public static void createMainFolder(String folderName) {
19
20        File file = new File(folderName);
21        if (!file.exists()) {
22            file.mkdirs();
23        }
24    }
```

Output:



Step 5.2: Writing method to display all files in “main” folder in ascending order and also with directory structure. (“`--" represents a directory. “|--” represents a file.)

```java
    public static void displayAllFiles(String path) {
        FileOperations.createMainFolder("main");
        System.out.println("Displaying all files with directory structure in ascending order\n");List<String> filesListNames = FileOpe

        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());

                    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;
    }
```

Output:

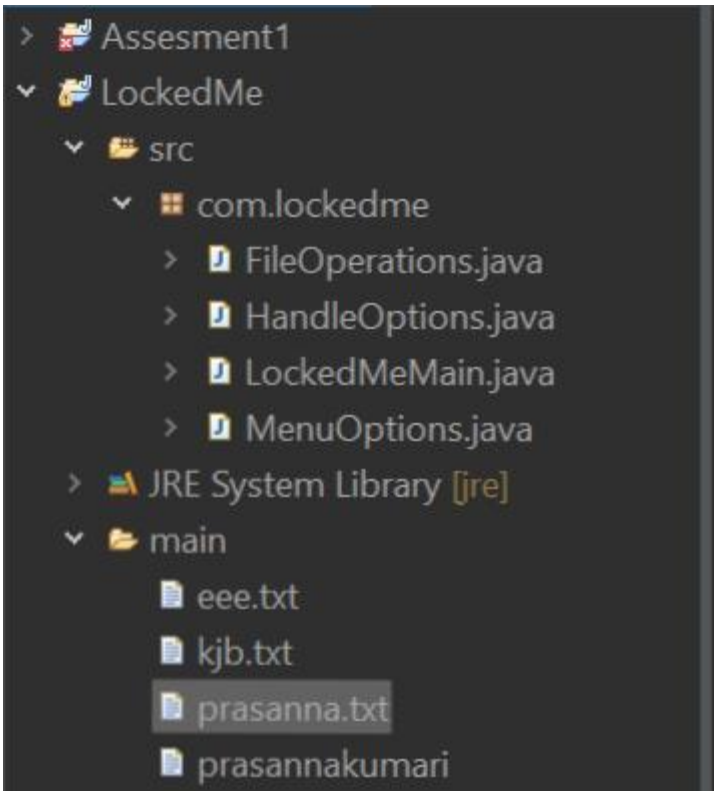
```
-------------------------------------------------------------
** Welcome to LockedMe.com.
** This application was developed by : Prasanna kumari Chitta..
-------------------------------------------------------------

You can use this application to :-
1. Retrieve all file names in the "main" folder.
2. Search, add, or delete files in "main" folder.



-:folder Options:-

1. Files inside "main" folder.
2. File Operations.
3. Exit.

1
Displaying all files with directory structure in ascending order

|-- eee.txt
|-- kjb.txt
|-- prasannakumari

Displaying all files in ascending order

eee.txt
kjb.txt
prasannakumari
```

Step 5.3: Writing method to create a file/folder as specified by user input.

```java
    public static void createFile(String fileToAdd, Scanner sc) {
        FileOperations.createMainFolder("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());
        }
    }
```

Output:

```
-: File Operations:-

1. Add file.
2. Delete file.
3. Search file.
4. Previous Menu.
5. Exit.

1
-:Enter Filename:-
prasanna.txt
prasanna.txt created successfully
Would you like to add some content to the file? (Y/N)
n


-: File Operations:-

1. Add file.
2. Delete file.
3. Search file.
4. Previous Menu.
5. Exit.
```


Assesment1
LockedMe
src
com.lockedme
FileOperations.java
HandleOptions.java
LockedMeMain.java
MenuOptions.java
JRE System Library [jre]
main
eee.txt
kjb.txt
prasanna.txt
prasannakumari

Step 5.4: Writing method to search for all files as specified by user input in “main” folder and it’s subfolders.

```java
    public static List<String> displayFileLocations(String fileName, String path) {
        List<String> fileListNames = new ArrayList<>();
        FileOperations.searchFile(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 searchFile(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());
                }

                                if (file.isDirectory()) {
                    searchFile(file.getAbsolutePath(), fileName, fileListNames);
                }
            }
        }
    }
```

Output:

```
-: File Operations:-

1. Add file.
2. Delete file.
3. Search file.
4. Previous Menu.
5. Exit.

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


Found file at below location(s):
1: C:\Users\91965\Desktop\LockedMe-main\LockedMe-main\main\prasanna.txt


-: File Operations:-

1. Add file.
2. Delete file.
3. Search file.
4. Previous Menu.
5. Exit.
```

Step 5.5: Itses the searchFilesRecursively method and prompts user to specify which index to delete. If folder selected, all it's child files and folder will be deleted recursively. If user wants to delete all the files specified after the search, they can input value 0.

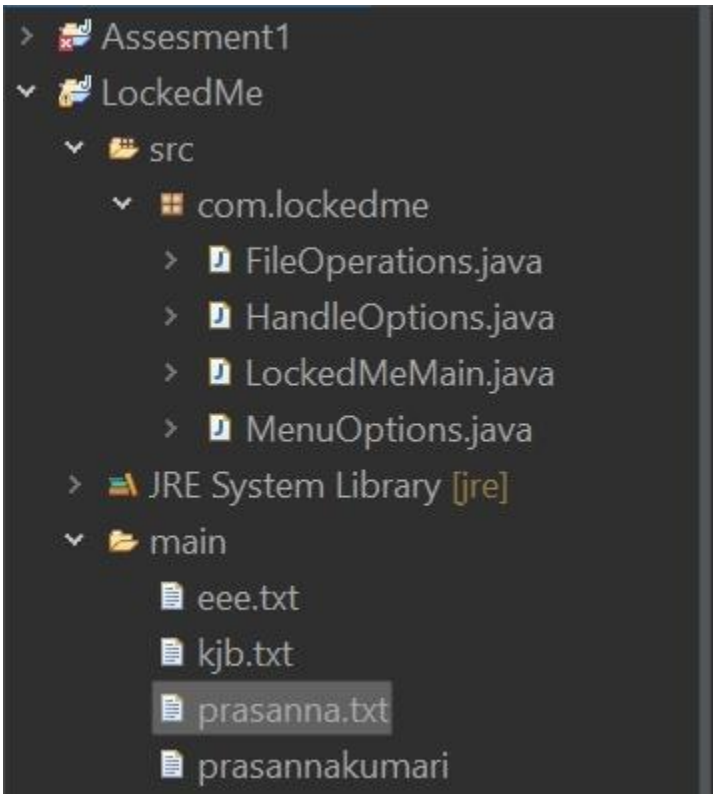
```java
    public static void deleteFile(String path) {

        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()) {
                    deleteFile(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);
        }
    }
}
```

Output:

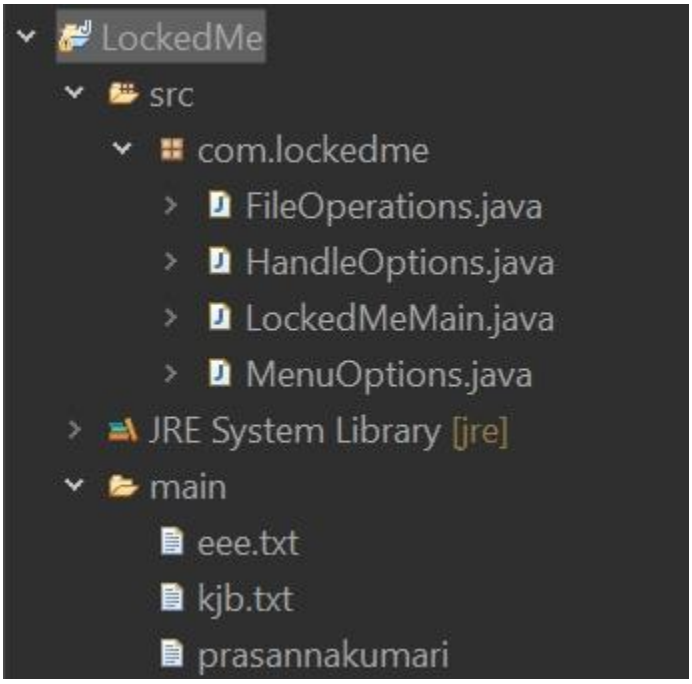
To verify if file is deleted on Eclipse, right click on Project and click "Refresh

```
-: File Operations:-

1. Add file.
2. Delete file.
3. Search file.
4. Previous Menu.
5. Exit.

2
-:Enter Filename:-
prasanna.txt


Found file at below location(s):
1: C:\Users\91965\Desktop\LockedMe-main\LockedMe-main\main\prasanna.txt

Do you want to delete?
(Enter "0")
0
prasanna.txt at C:\Users\91965\Desktop\LockedMe-main\LockedMe-main\main deleted successfully


-: File Operations:-

1. Add file.
2. Delete file.
3. Search file.
4. Previous Menu.
5. Exit.
```

Before :



After file deleted:



Step 6: Pushing the code to GitHub repository

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

  cd <folder path>

- Initialize repository using the following command:

  git init

- Add all the files to your git repository using the following command:

  git add .

- Commit the changes using the following command:

  git commit . -m <commit message>

- 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.

8. The application is designed with modularity in mind. Even if one wants to update the path, they can change it through the source code. Application has been developed keeping in mind that there should be very less "hardcoding" of data.

Conclusions

Further enhancements to the application can be made which may include:

- Conditions to check if user is allowed to delete the file or add the file at the specific locations.
- Asking user to verify if they really want to delete the selected directory if it's not empty.
- Retrieving files/folders by different criteria like Last Modified, Type, etc.

- Allowing user to append data to the file.