**Lockers Pvt. Ltd.**

**LockedMe.com**

**Developer: Jyothi Kothapally**

|  |  |
| --- | --- |
| Author | Jyothi kothapally |
| Date | 14-aug-2021 |
| Version | 1.0 |
| Purpose | Specific documentation |

Version History

**Contents**

[**1.GitHub Link** 3](#_Toc79840309)

[**2.Modules in the project** 3](#_Toc79840310)

[**3.Flow chart of the application** 4](#_Toc79840311)

[**4.Sprint Planning and Task Completion** 4](#_Toc79840312)

[**5.Core concepts used in the project** 5](#_Toc79840313)

[**6.Demonstration the product capabilities, appearance and user interactions** 5](#_Toc79840314)

[**7.Pushing the code to GitHub repository** 27](#_Toc79840315)

[**8.Unique selling points of the application** 28](#_Toc79840316)

[**9. Conclusions** 29](#_Toc79840317)

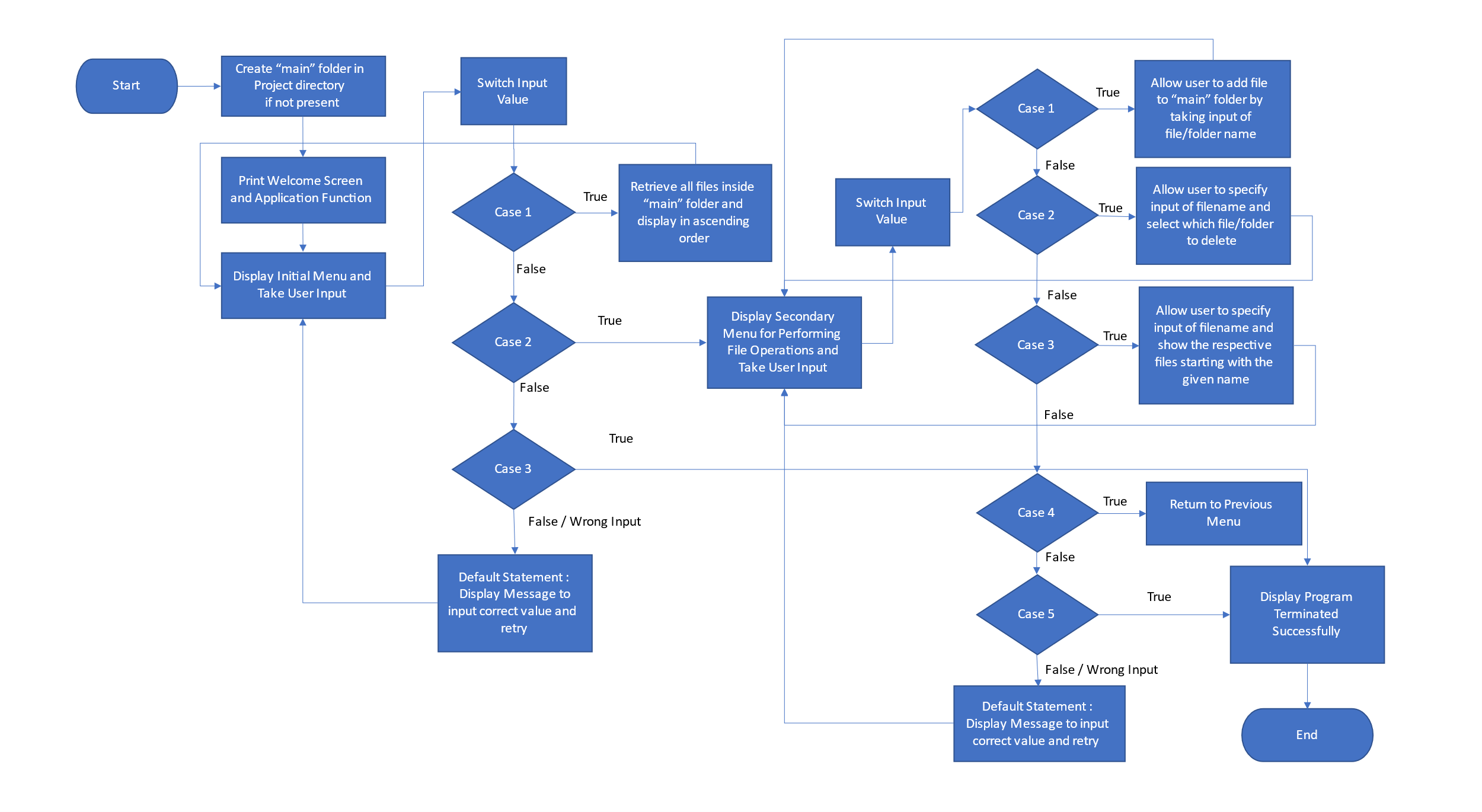
# **GitHub Link**

This code for this project is hosted at [kothapallyjyothi/LockedMe.com (github.com)](https://github.com/kothapallyjyothi/LockedMe.com).

# **Modules in the project**

1. The first option returns the current file names in ascending order. The directory contains few files in it.
2. The second option returns the details of the user interface such as options displaying the following:
   * Add a file to the existing directory.
   * Delete a user specified file from the existing directory.
   * Search a file from the main directory.
   * Option to navigate back to main Menu
   * Option to close the current application.
3. Last option is used to close the application.

# **3.Flow chart of the application**



# **4.Sprint Planning and Task Completion**

The Project is planned to be completed in 2 sprints. Tasks assumed to be completed in the sprint are:

**Sprint 1:**

* Creating the flow of the application.
* Initializing the git repository to track changes as development progress.
* Writing the Java program to ensure that requirements of the project.
* Main Menu Options
* Retrieve the files in a directory
* Close the application
* Testing the Java program with different kinds of user input values.

**Sprint 2:**

Added other modules programs to ensure the requirements of the projects.

* File Operations Menu.
* Adding the file into directory.
* Delete the file from the directory.
* Search a file in the directory.
* Testing the Java program with different kinds of user input values.
* Pushing the code to GitHub.
* Creating this specification document highlighting application capabilities, appearance and user interactions.

# **5.Core concepts used in the project**

* Collections framework
* Exception handling
* File Handling
* Control structures
* Recursion
* Sorting techniques

# **6.Demonstration 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 java program for the entry point of the application (LockedMeMain.java)

**Step-1: Creating the project in Eclipse.**

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

**Step -2: Develop the code in java for the Entry point of the application LockedMeMain.java(handling the WelcomeAndMenuOption.java)**

**package** com.lockedme;

**public** **class** WelcomeAndMenuOptions

{

**public** **static** **void** displayMenu()

{

Object companyName = "Lockedme";

Object developerName = " Jyothi Kothapally" ;

//welcome page

String companyDetails = String.*format*("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"+ "\*\* Welcome to %s.com. \n" + "\*\* This application was developed by %s.\n"+ "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n", companyName, developerName);String appFunction = "You can use this application to :-\n"+ "• Retrieve all file names in the \"directory\" folder\n"+ "• Search, add, or delete files in \"directory\" folder.\n"+ "\n\*\*Please be careful to ensure the correct filename is provided for searching or deleting files.\*\*\n";

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

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

// Creating a DisplayMenu

String menu = "\n\n\*\*\*\*\*\* Select any option number from below and press Enter \*\*\*\*\*\*\n\n"+ "1) Retrieve all files inside \"directory\" folder\n" + "2) Display menu for File operations\n"+ "3) Exit program\n";

System.***out***.println(menu);

System.***out***.println("Enter your choice: \n");

}

**public** **static** **void** displayFileMenuOption()

{

// Creating a DisplayFileMenuOption to Add,Search,and Delete File

String fileMenu = "\n\n\*\*\*\*\*\* Select any option number from below and press Enter \*\*\*\*\*\*\n\n"

+ "1) Add a file to \"directoryy\" folder\n" + "2) Delete a file from \"directory\" folder\n"

+ "3) Search for a file from \"directory\" folder\n" + "4) Show Previous Menu\n";

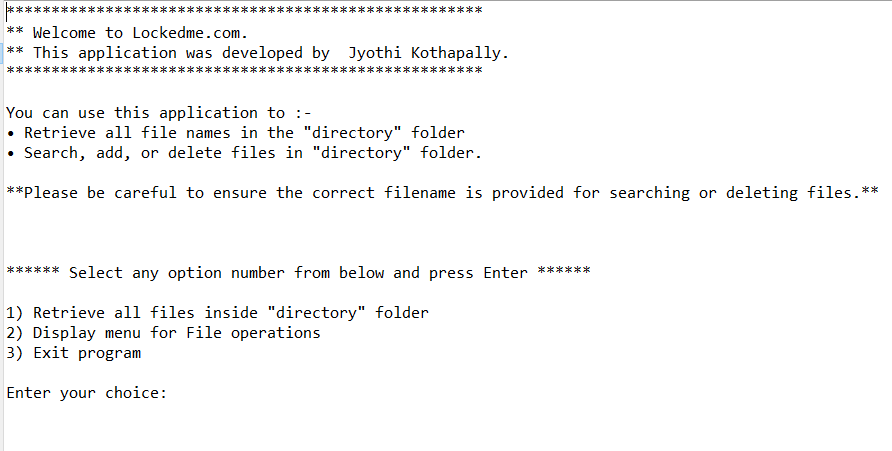
System.***out***.println(fileMenu);

System.***out***.println("Enter the option number:");

}

}

**Output**



**Step 3: Develop the code in java for the entry point of the application**

**LockedMeMain.java(FileOperations.java)**

**package** com.lockedme;

**import** java.io.File;

**import** java.io.FileWriter;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** FileOperations

{

/\*\*

\* This method will return the file names from the folder

\* **@param** folderpath

\* **@return**

\*/

**public** **static** List<String> getAllFiles(String folderpath)

{

//creating File Object

File f1 = **new** File(folderpath);

//Getting all the files into FileArray

File[] listOfFiles = f1.listFiles();

//Declare a list to store file names

List<String> fileNames = **new** ArrayList<String>();

//declare for each loop for get the file names

**for**(File f:listOfFiles)

fileNames.add(f.getName());

//return the List

**return** fileNames;

//

}

/\*\*

\* This method will create or append content into the file specified

\* **@param** folderpath

\* **@param** fileName

\* **@param** content

\* **@return** boolean

\*/

**public** **static** **boolean** createFiles(String folderpath,String fileName,List<String> content)

{

**try**

{

//created object of folderPath and fileName

File f1 = **new** File(folderpath, fileName);

FileWriter fw = **new** FileWriter(f1);

**for**(String s:content)

{

fw.write(s+"\n");

}

fw.close();

**return** **true**;

}

**catch**(Exception Ex)

{

**return** **false**;

}

}

/\*\*

\* This method will delete the file name if it exists.

\* **@param** folderpath

\* **@param** fileName

\* **@return**

\*/

**public** **static** **boolean** deleteFile(String folderpath, String fileName)

{

//adding folderpath with file name and creating file object

File file = **new** File(folderpath+"\\"+fileName);

**try**

{

**if**(file.delete())

**return** **true**;

**else**

**return** **false**;

}

**catch** (Exception Ex)

{

**return** **false**;

}

}

/\*\*

\* This method will search from the folder, whether searched file exists or not

\* **@param** folderpath

\* **@param** fileName

\* **@return**

\*/

**public** **static** **boolean** searchFile(String folderpath, String fileName)

{

//adding folderpath with file name and creating file object

File file = **new** File(folderpath+"\\"+fileName);

**if**(file.exists())

**return** **true**;

**else**

**return** **false**;

}

}

//Operations

**Output**

**Step 4: Develop the code in Java for the Entry point of the application LockedMeMain.java (handling the FileHandlingOptions.java)**

**package** com.lockedme;

**import** java.util.Scanner;

**public** **class** FileHandlingOptions {

**public** **static** **boolean** displayFirstMenuOptions()

{

**do** {

**try**

{

Scanner sc = **new** Scanner(System.***in***);

**int** input;

WelcomeAndMenuOptions.*displayMenu*();

input = Integer.*parseInt*(sc.nextLine()); //Taking Input from the user

**switch** (input)

{

**case** 1:

LockedMeMain.*getAllFiles*(); //Display files in the directory

**break**;

**case** 2:

FileHandlingOptions.*welcomeScreenOptions*();

**break**;

**case** 3:

System.***out***.println("Program exited successfully."); //exit the program

**return** **true**;

**case** 4:System.*exit*(0);

**break**;

**default**:

System.***out***.println("Please select a valid option from above.");//In case of invailed Entery

}

}

**catch** (Exception ex)

{

System.***out***.println(ex.getClass().getName());

}

}**while** (**true**);

}

**public** **static** **boolean** welcomeScreenOptions()

{

**do** {

//Variable declaration

Scanner sc = **new** Scanner(System.***in***);

**int** ch;

WelcomeAndMenuOptions.*displayFileMenuOption*();

**try**

{

//Menu

ch=Integer.*parseInt*(sc.nextLine());

//switch

**switch**(ch)

{

**case** 1 : LockedMeMain.*createFiles*(); //getting the filenames from folderpath

**break**;

**case** 2 :LockedMeMain.*deleteFile*(); // creating new file in folderpath

**break**;

**case** 3: LockedMeMain.*searchFile*(); // deleting a file from folderpath

**break**;

**case** 4: **return** **false**; //Search a file from given folderpath

**case** 5: System.*exit*(0); //

**break**;

**default** : System.***out***.println("Invalid Option");

**break**;

}

}

**catch**(Exception e)

{

System.***out***.println(e.getClass().getName());

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}**while** (**true**);

}

**public** **static** **boolean** FileHandlingDisplayMenu()

{

**do**

{

**try**

{

//Menu

WelcomeAndMenuOptions.*displayMenu*();

//Variable declaration

Scanner obj = **new** Scanner(System.***in***);

**int** ch;

ch=Integer.*parseInt*(obj.nextLine());

//

**switch**(ch)

{

**case** 1 : LockedMeMain.*getAllFiles*(); //getting the filenames from folderpath

**break**;

**case** 2 : LockedMeMain.*createFiles*(); // creating new file in folderpath

**break**;

**case** 3: LockedMeMain.*deleteFile*(); // deleting a file from folderpath

**break**;

**case** 4: LockedMeMain.*searchFile*(); //Search a file from given folderpath

**break**;

**case** 5: System.*exit*(0);

System.***out***.println("Program exited successfully.");//exit the program

**break**;

**default** : System.***out***.println("Invalid Option");

**break**;

}

}

**catch** (Exception Ex)

{

System.***out***.println(Ex.getClass().getName());

// FileHandlingDisplayMenu();

}

}**while** (**true**);

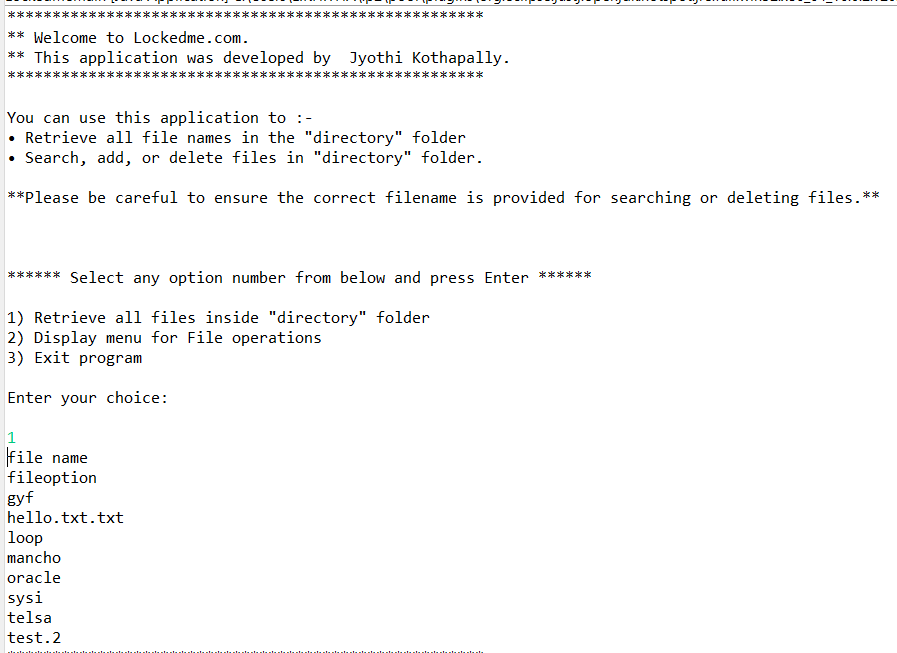
}

}

**Output:**

**Case :1. Retrieving Files**

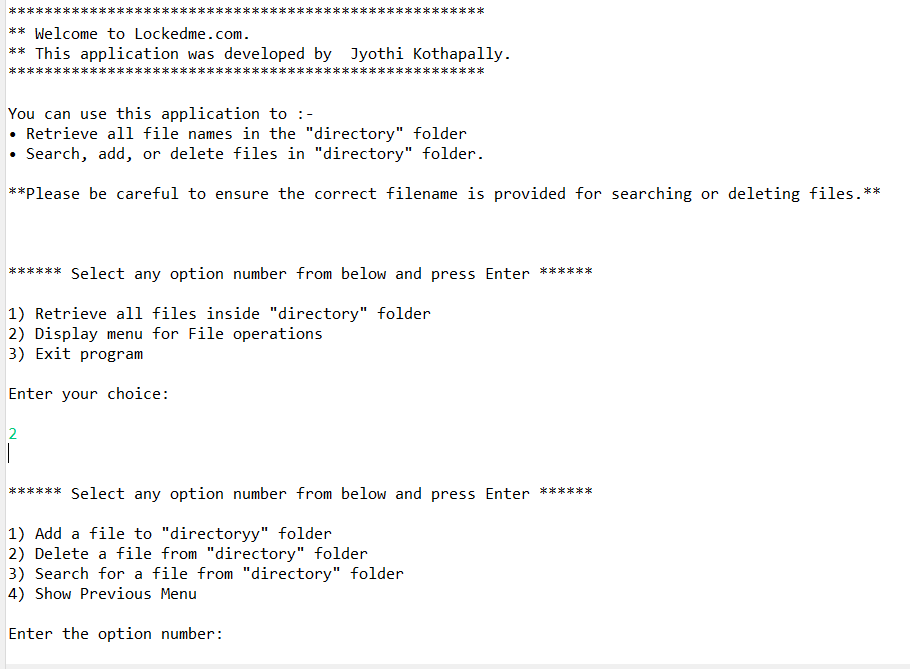
(WelcomeAndMenuOption to retrieve all the file in the directory)



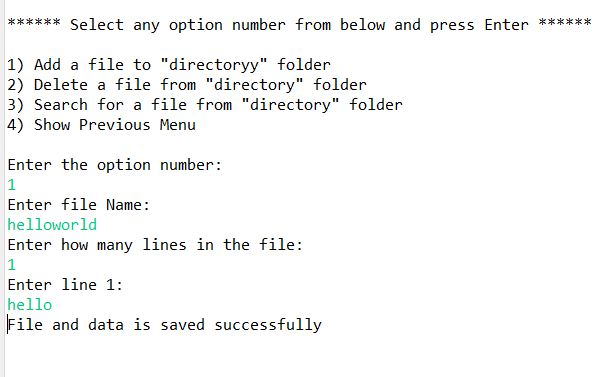
**Case 2: Display File Menu Operations**

(WelcomeAndMenuOptions displays Display menu for File Operation )

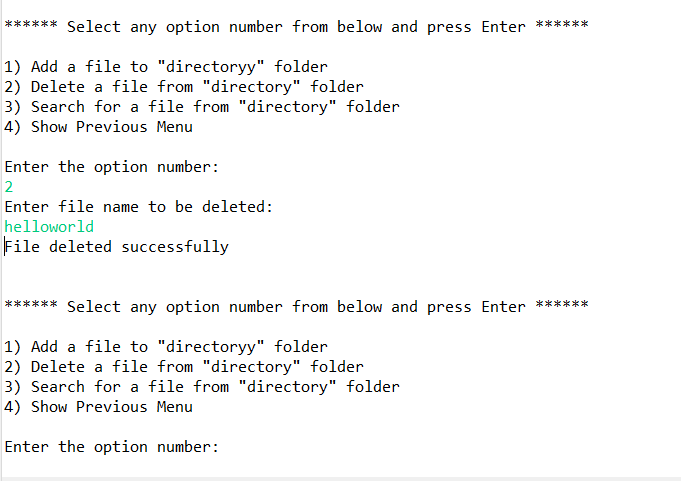
* Second option should return the details of the user interface such as options displaying the following



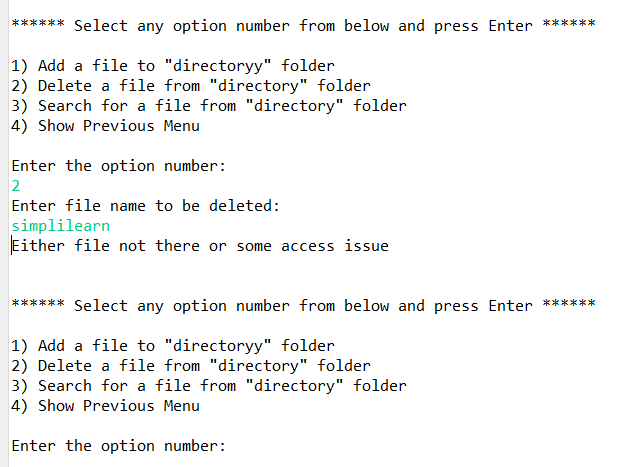
**Case 2.1: add a file to the existing directory**



**Case 2.2: delete a file from the existing directory**

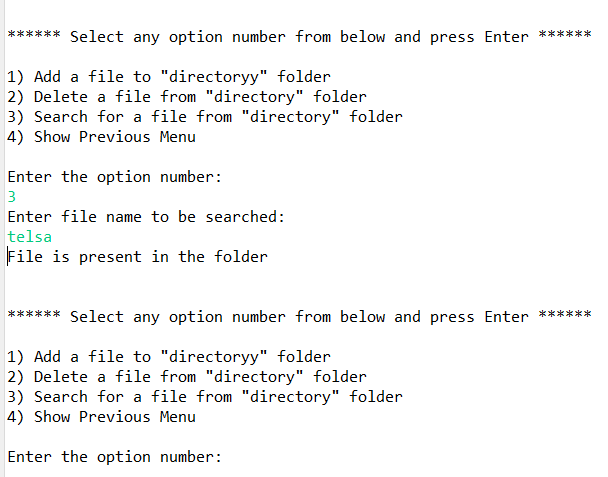


* Return a message if FNF (File not found)

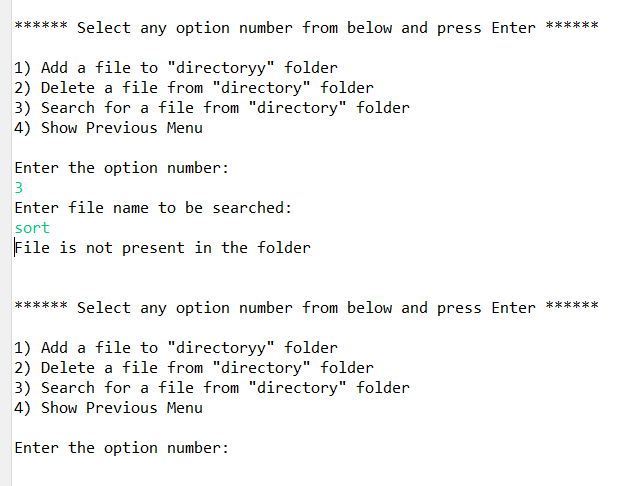


**Case 2.3: search a user specified file from the main directory**

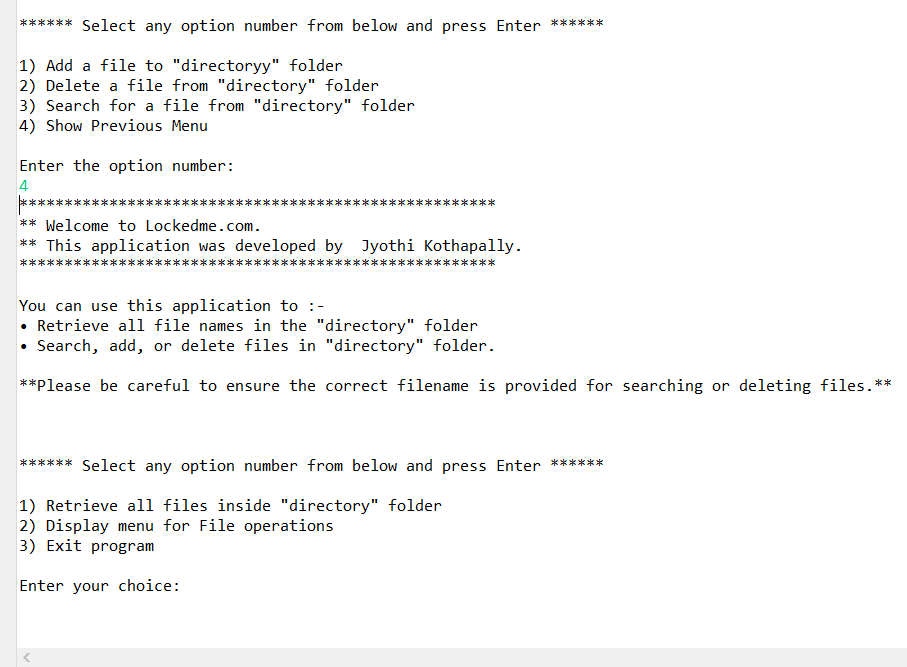
* Displays the result upon successful operation



* Display the result upon successful operation

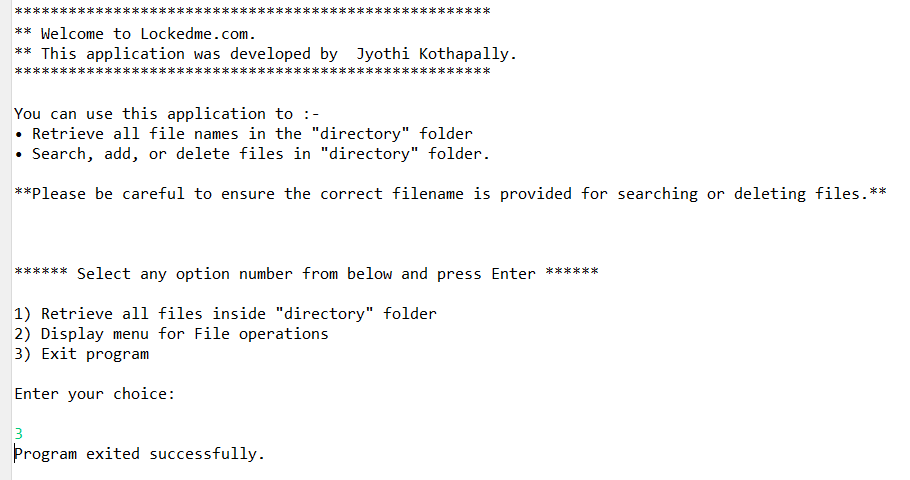


**Case 2.4: option to navigate back to main context**



**Case 3: Exit program.**

(WelcomeAndMenuOptions Exits the program)



Step 4: develop the code in java for the Entry point of the application LockedMain.java

**package** com.lockedme;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** LockedMeMain {

**static** **final** String ***folderpath***="C:\\MyPhase1Project\\SourceCode\\LockedmeProject\\directory";

**public** **static** **void** main(String[] args)

{

//Display Main Menu Options

FileHandlingOptions.*displayFirstMenuOptions*();

}

**public** **static** **void** getAllFiles()

{

//Getting the files

// **TODO** Auto-generated method stub

//get file names

//Using forEach loop to get all file names

List<String> fileName = FileOperations.*getAllFiles*(***folderpath***);

**for** (String f :fileName)

System.***out***.println(f);

}

/\*\*This class to create files calling method from FileOperations

\*

\*/

**public** **static** **void** createFiles()

{

//Adding file name

//Variable declaration

Scanner obj = **new** Scanner(System.***in***);

String fileName;

**int** linesCount;

List<String> content = **new** ArrayList<String>();

//Read file name from user

System.***out***.println("Enter file Name:");

fileName=obj.nextLine();

//Read number of lines from user

System.***out***.println("Enter how many lines in the file:");

linesCount=Integer.*parseInt*(obj.nextLine());

//Read Line from user

**for**(**int** i=1;i<=linesCount;i++)

{

System.***out***.println("Enter line "+i+":");

content.add(obj.nextLine());

}

//save the content into the file

**boolean** isSaved = FileOperations.*createFiles*( ***folderpath***, fileName, content);

**if**(isSaved)

System.***out***.println("File and data is saved successfully");

**else**

System.***out***.println("Some error occured. Please contact admin@jyo.com");

}

**public** **static** **void** deleteFile()

{

//Code for deleting a file

String fileName;

Scanner obj = **new** Scanner(System.***in***);

System.***out***.println("Enter file name to be deleted:");

fileName=obj.nextLine();

//deletes the specific file from folderpath

**boolean** isDeleted = FileOperations.*deleteFile*(***folderpath***, fileName);

**if**(isDeleted)

System.***out***.println("File deleted successfully");

**else**

System.***out***.println("Either file not there or some access issue");

}

**public** **static** **void** searchFile()

{

//Code for searching a file

String fileName;

Scanner obj = **new** Scanner(System.***in***);

// \*\*input from user

System.***out***.println("Enter file name to be searched:");

fileName=obj.nextLine();

//

**boolean** isFound = FileOperations.*searchFile*(***folderpath***, fileName);

**if**(isFound)

System.***out***.println("File is present in the folder");

**else**

System.***out***.println("File is not present in the folder");

}

}

# **7.Pushing the code to GitHub repository**

Instructions are to be followed:

* First of all, we need account in Github to push the code in GitHub.
* Create account in GitHub.
* Create new repository.
* Open your command prompt and navigate to the folder where you have created your files.

cd

* Initialize repository using the following command:

git init

* Add all the files to your git repository using the following the 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 main

# **8.Unique selling points of the application**

1. The application is designed to keep on running and taking the user inputs even after exceptions occur though. To terminate the application option needs to be selected.
2. The application can take any file 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 string input, and the program searches all files/folder names starting with the value and display it. User is then provided the option to select all files or 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 switch between menu /options or return to previous menu even after any required operation like creating, searching, deleting or retrieving of files is to be performed.
7. When the option to retrieve files in ascending order is selected, user is displayed with two options of viewing the files.
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.

# **9. 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 the user to append data to the file.