Lockers Pvt. Ltd.

LockedMe.com

Developer: **Bharat Kumar Ponugupati**

Version History

Author	Bharat Kumar Ponugupati
Date	12-Aug-2021
Version	1.0
Purpose	Specification documentation

Contents

1. GitHub Link

- The application code and its version can be tracked in:

  https://github.com/Bharatkumarp28/LockedMe.com

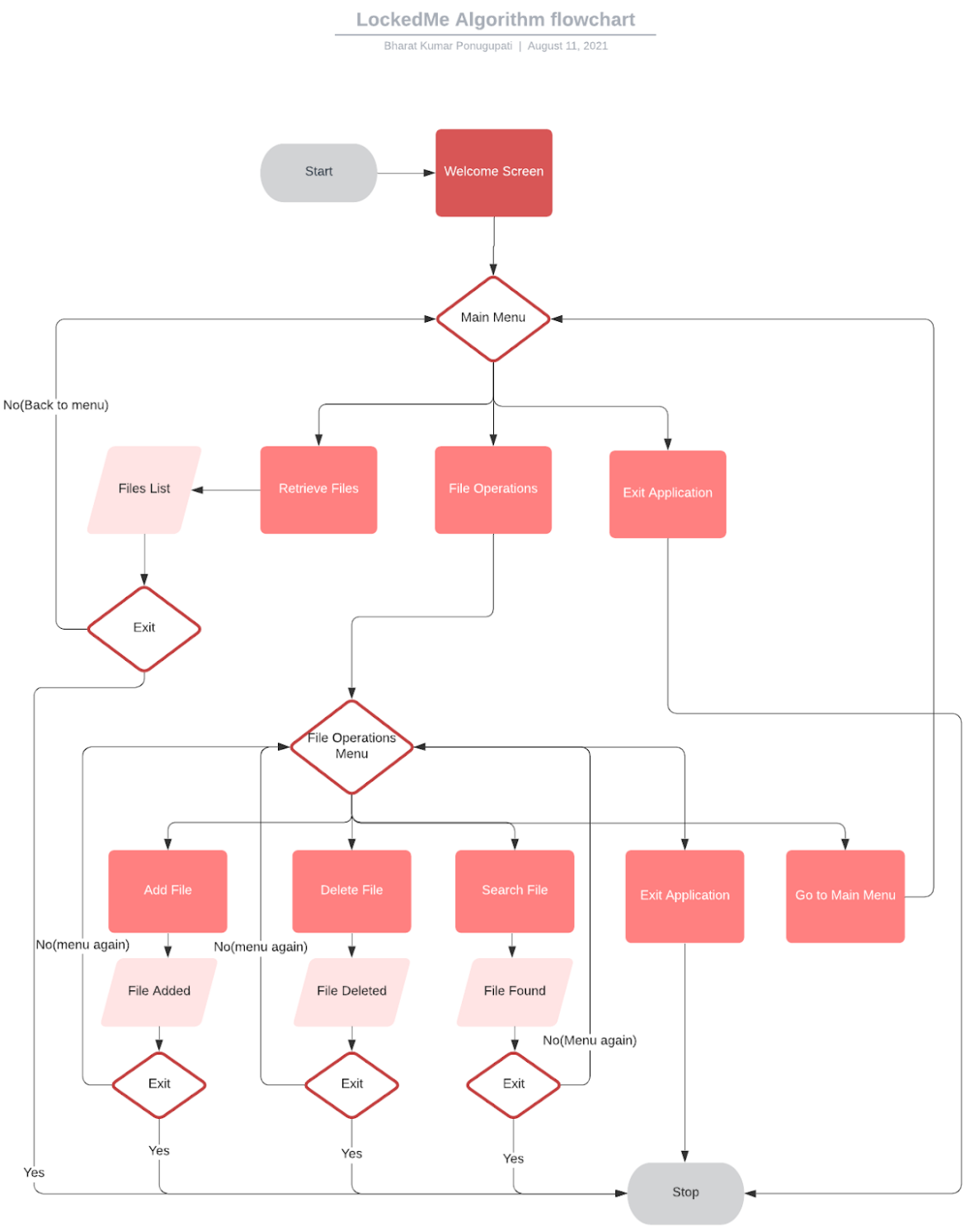
2. Modules in the project

- The first option returns the current file names in ascending order. The root directory can be either empty or contain few files or folders in it.
- The second option returns the details of the user interface such as options displaying the following:
  - Add a file to the existing directory list.
  - Delete a user specified file from the existing directory list.
  - Search a user specified file from the main directory.
  - Option to navigate back to the main context.
  - option to close the application.
- There is a third option to close the application.

3. Sprint Planning

S No.	Sprint Number	Modules Covered
1.	Sprint 1	• Main Menu Option. • Retrieve all files in a directory. • Close the application. • Testing above modules.
2.	Sprint 2	• File Operation Menu. • Adding the file into directory. • Delete the file in a directory. • Search a file in a directory. • Testing above modules.

4. Flow of the application

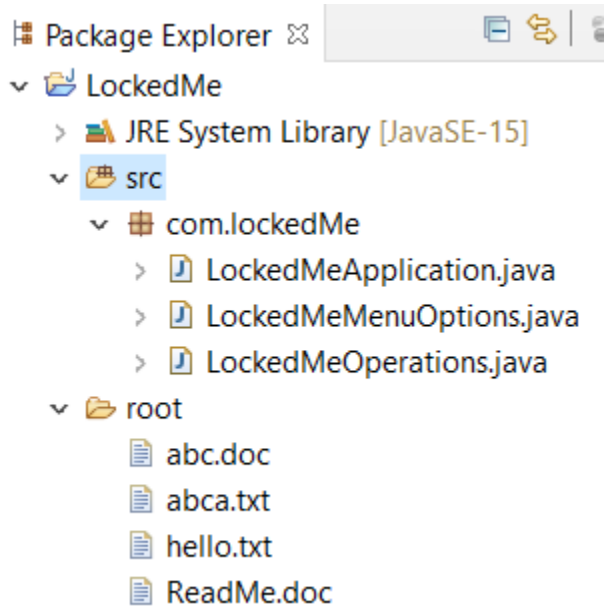


5. Core concepts and algorithms

1. Object Oriented Programming.
2. Collections Framework.
3. File handling.
4. Exception handling.
5. Control Structures.
6. Data Structures: Search and Sort techniques.

6. Project Programming Details

6.1 Folder Structure



6.2 Java Program to implement LockedMe.com

6.2.1 Creating new project in Eclipse

There are two ways you can perform this step; you can create a new Java project, or you can create a new Java class in the existing project. It is preferable to create a new Java class in the existing project but feel free to explore the first option. The steps mentioned below will work once you create a project in Java:

- Open Eclipse.
- [Right click] on the src folder of the project.
- Select New -> Java Class -> Enter the filename (follow camelCasing)
- Execute the code below resolving the warning and errors due compatibility-related issues

6.2.2 Java program for entry point for the application [LockedMeApplication.java]

```
//1. Class is having main method to execute this project
public class LockedMeApplication
{

    public static void main(String[] args)
    {
       //Welcome Page
       System.out.println("-------------------------------------------------
-----");
       System.out.println("*\t\tLockedMe.com\t\t\t    *");
       System.out.println("*\t---Developed by Bharat Kumar P---\t    *");
       System.out.println("-------------------------------------------------
-----");

      //Create Directory for application operations
       LockedMeOperations.makeDirectory();

      //Display Main Menu options
       LockedMeMenuOptions.mainMenuOptions();
    }


}
```

6.2.3 Java program to handle file operations selected by users [LockedMeOperations.java]

```
package com.lockedMe;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
```

```
import java.util.List;
import java.util.Scanner;

//2. Class which returns all the file operations
public class LockedMeOperations
{
    //Class variables
    //Get the current directory path
    static Path dirFullPath = Paths.get("");

    //path to create a root directory
    static final String FILE_HANDLING_DIRECTORY =
(dirFullPath.toAbsolutePath().toString()+ "\\root");

    //Scanner
    static Scanner sc = new Scanner(System.in);

    /**
     * method to create directory
     */
    public static void makeDirectory()
    {
       //Initializing file object
       File createDirectory = new File(FILE_HANDLING_DIRECTORY);

      //Exceptional handling
       try
       {
           //Make directory logic
           if(!createDirectory.exists())
           {
               createDirectory.mkdir();
               System.out.println("Created root folder for
Application Operations at: \n"+ "***" + FILE_HANDLING_DIRECTORY+
"***\n");
           }
      }
       catch(Exception ex)
       {
           System.out.println("\t***Error: An unexpected error
occured*** " + "\nPlease contact Admin@CompanyLockers.in");
       }
    }

    /**
     * method return list of files in the directory
```

```
     * @return List<String>
     */
    public static List<String> retrieveFileNames()
    {
       //Initializing List to store filenames
       List<String> fileNames = new ArrayList<String>();

      //Initializing file with directory path
       File locFiles = new File(FILE_HANDLING_DIRECTORY);

      //Fetch filename and add to List
       for(String file :locFiles.list())
           fileNames.add(file);

      return fileNames;
    }


    /**
     * method to return files are added into the directory or not
     * @param newFileName
     * @param countOfLines
     * @param dataIntoFile
     * @return true(if added)
     */
    public static boolean addFile(String newFileName, int
countOfLines,List<String> dataIntoFile)
    {
       //Declaring FileWriter
       FileWriter fileToWrite = null;

      //Exceptional handling
       try
       {
          //Initialize location to add file
           fileToWrite = new FileWriter(FILE_HANDLING_DIRECTORY +"\\"
+ newFileName);

          for(String dataParts: dataIntoFile)
               fileToWrite.write(dataParts);

          return true;

      }
       catch (IOException e)
       {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
            return false;
        }
        finally {
                try
                {
                    if(fileToWrite != null)
                        fileToWrite.close();
                }
                catch (IOException e)
                {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
      }
    }


    /**
     * method to return if file is deleted from the directory or not
     * @param fileNameToBeDeleted
     * @return true(if deleted)
     */
    public static boolean deleteFile(String fileNameToBeDeleted)
    {
       //Initialize location to delete file
       File fileToDelete = new File(FILE_HANDLING_DIRECTORY
                                    + "\\" +   fileNameToBeDeleted);

      //Delete user specified file
       if(fileToDelete.delete())
            return true;
       else
            return false;

    }


    /**
     * method return file found in a directory or not
     * @param fileNameToBeSearched
     * @return true(if file found)
     */
    public static boolean searchFile(String fileNameToBeSearched)
    {
```

```
      //Initialize location to search file
       File locFiles = new File(FILE_HANDLING_DIRECTORY + "\\" +
fileNameToBeSearched);

      //check if file exist in the location or not
       if(locFiles.exists())
           return true;
       else
           return false;
    }
}
```

Step 4: Java program to display menu options for users [LockedMeMenuOptions.java]

```
package com.lockedMe;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;

//3. Class which display menu options for user to work on.
public class LockedMeMenuOptions
{
    //Scanner
    static Scanner sc = new Scanner(System.in);

    /**
     * Display main menu Options
     */
    public static void mainMenuOptions()
    {
       //Declaring variables
       int mMenuvalue;
       boolean menuFlag;
       int gotInp;

      //Initializing flag for Menu options
       final char OPTIONS_LABLE = 'M';
```

```
      //do-while loop to repeat if input is numeric value other
then menu options.
       do
       {
           menuFlag = false;

          //Try-catch to handle exceptions
           try
           {
               //Main Menu Options
               System.out.println("\n   @Main Menu Options");

System.out.println("**************************************************
****");
               System.out.println("1. Retrieving the file names");
               System.out.println("2. Display options for file
operations");
               System.out.println("3. Close the application");

              //User input option

System.out.println("**************************************************
****");
               System.out.println("Enter your choice: ");
               mMenuvalue = Integer.parseInt(sc.nextLine());

              //Switch case to enter into logic of menu options
               switch(mMenuvalue)
               {
                   case 1:
                       System.out.println("Files retrieve from the
folder path: \n"+ LockedMeOperations.FILE_HANDLING_DIRECTORY + "
are:");

              //Calling getFiles method to fetch all files.
                       getFiles();

              //Calling method to display option after file actions
                       gotInp =
afterFileOperations(OPTIONS_LABLE);

                     //Menu to exit or return to main menu
                      if(gotInp == 1)
                          menuFlag = true;
                      else if(gotInp == 2)
                          exitApp();
```

```
                        break;
                    case 2:
                        //Calling file operation method to entered
into file options menu
                        fileOperations();
                        break;
                    case 3:
                        //Close the application
                        exitApp();
                        break;
                    default:
                        System.out.println("invalid option, Try
again!");
                        menuFlag = true;
                }
            }
            catch(NumberFormatException nfex)
            {
                System.out.println("\t***Error: You must enter a
numeric value***\n"
                        + "\t\t***Please try again***\n");
                menuFlag = true;
            }
            catch(Exception ex)
            {
                System.out.println("\t***Error: An unexpected error
occured*** \nPlease contact Admin@CompanyLockers.in");
            }

      }while(menuFlag);
    }

    /**
     * method to close the application
     */
    public static void exitApp()
    {
       //Display output to user
       System.out.println("Thank you, closing the Application");
       sc.close();
       System.exit(0);
    }

    /**
     * method to display all files in the directory
     */
```

```
    public static void getFiles()
    {
       //Get file names in a list
       List<String> filesInDirectory =
LockedMeOperations.retrieveFileNames();

      //logic to retrieve files and display Null if no files
       if(filesInDirectory.size() == 0)
           System.out.println("No files in the folder!");

      //Display output to user
       for(String file : filesInDirectory)
           System.out.println("- "+ file);
    }

    /**
     * method to add file into the directory
     */
    public static void addFile()
    {
       //List to collect data enter by user in String
       List<String> allDataCollected = new ArrayList<String>();

      //Getting the new File name from user
       System.out.println("Enter the name of the File: ");
       String newFileName = sc.nextLine();


      //Getting input from user to know how many lines of data to
enter
       System.out.println("How many line you want to write into file:
");
       int countOfLines = Integer.parseInt(sc.nextLine());

      //logic to take data from users and push into list
       for(int i = 1; i <= countOfLines; i++)
       {
           System.out.println("Enter the text for line " + i + ":");
           String dataIntoFile = sc.nextLine() + "\n";
           allDataCollected.add(dataIntoFile);
       }

      //Add data to file using the above List
       boolean isAdded =
LockedMeOperations.addFile(newFileName,countOfLines,allDataCollected)
;
```

```
      //Display output to user
       if(isAdded)
           System.out.println("File added successfully!");
       else
           System.out.println("File not added");
    }

    /**
     * method to show if file is removed from the directory
     */
    public static void removeFile()
    {
       //Get user specified file
       System.out.println("Enter name of the File to delete: ");
       String fileNameToBeDeleted = sc.nextLine();

      //Delete user specified file
       boolean isDeleted =
LockedMeOperations.deleteFile(fileNameToBeDeleted);

      //Display output to user
       if(isDeleted)
           System.out.println("File is succesfully deleted!");
       else
           System.out.println("File does not exist in the path to
delete!");
    }

    /**
     * method to display searched file found in the directory
     */
    public static void searchForFile()
    {
       //Get user specified file to search
       System.out.println("Enter name of the File to search: ");
       String fileNameToBeSearched = sc.nextLine();

      //Search user specified file
       boolean isFound =
LockedMeOperations.searchFile(fileNameToBeSearched);

      //Display output to user
       if(isFound)
           System.out.println("File found in the location");
       else
```

```
            System.out.println("File not found!");
    }

    /**
     * User input after performing file actions
     * @param gotLable
     * @return Option enter by user
     */
    public static int afterFileOperations(char gotLable)
    {
       //Declaring variables
       int inputOption = 0;
       boolean flag;

      //do-while loop to repeat options for user.
       do
       {
           flag = false;

          //Try-catch to handle exceptions
           try
           {
               //Option for user after file/menu operations
               if(gotLable == 'F')
                   System.out.println("\nDo you want \n"
                     + "1. File Operations Menu \n"
                     + "2. Exit Application\n"
                     + "Enter your choice: ");
               else if(gotLable == 'M')
                   System.out.println("\nDo you want \n"
                     + "1. Menu Options \n"
                     + "2. Exit Application\n"
                     + "Enter your choice: ");
               //Input from user
               inputOption = Integer.parseInt(sc.nextLine());
           }
           catch(NumberFormatException nfex)
           {
               System.out.println("\t***Error: You must enter a
numeric value***\n"
                       + "\t\t***Please try again***\n");
               flag = true;
           }
           catch(Exception ex)
           {
               System.out.println("\tError: Please contact
```

```
Admin@CompanyLockers.in");
            }
        }while(flag);

      return inputOption;

    }
    public static void fileOperations()
    {
       //Declaring variables
       boolean fMenuFlag;
       int userInputOption;
       int gotInp;

      //Initializing flag for file operations menu
       final char OPTIONS_LABLE = 'F';

      //do-while loop to repeat if input is numeric value other then
menu options.
       do
       {
           fMenuFlag = false;

          //Try-catch to handle exceptions
           try
           {
               //File Options Menu
               System.out.println("\n   @File Opertions Menu");

System.out.println("**************************************************
****");
               System.out.println("1. Add a user specified file to
the application");
               System.out.println("2. Delete a user specified file
from the application");
               System.out.println("3. Search a user specified file
from the application");
               System.out.println("4. Go to Main Menu");
               System.out.println("5. Close the application");

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

              //Input from user
               userInputOption = Integer.parseInt(sc.nextLine());
```

```
                //Switch case to go to the user given input operation
                 switch(userInputOption)
                 {
                 case 1:
                     //Option to Add File for user
                     addFile();

                    //Calling method to display option after file
actions
                     gotInp = afterFileOperations(OPTIONS_LABLE);

                    //Menu to exit or return to file operation menu
                     if(gotInp == 1)
                         fMenuFlag = true;
                     else if(gotInp == 2)
                         exitApp();
                     break;

                case 2:
                     //Option to delete file
                     removeFile();

                    //Calling method to display option after file
actions
                     gotInp = afterFileOperations(OPTIONS_LABLE);

                    //Menu to exit or return to file operation menu
                     if(gotInp == 1)
                         fMenuFlag = true;
                     else if(gotInp == 2)
                         exitApp();
                     break;

                case 3:
                     //Option to search file
                     searchForFile();

                    //Calling method to display option after file
actions
                     gotInp = afterFileOperations(OPTIONS_LABLE);

                    //Menu to exit or return to file operation menu
                     if(gotInp == 1)
                         fMenuFlag = true;
                     else if(gotInp == 2)
```

```
                        exitApp();
                    break;

                case 4:
                    //Display option to user
                    System.out.println("main menu");

                   //Go back to main menu
                    mainMenuOptions();
                    break;

                case 5:
                    //Close the Application
                    exitApp();
                    break;

                default:
                    System.out.println("Enter a valid input, Try
again!");
                    fMenuFlag = true;
                 }
            }
            catch(NumberFormatException nfex)
            {
                System.out.println("\t***Error: You must enter a
numeric value***\n"
                        + "\t\t***Please try again***\n");
                fMenuFlag = true;
            }
            catch(Exception ex)
            {
                System.out.println("\tError: Please contact
Admin@CompanyLockers.in");
            }

      }while(fMenuFlag);
    }
}
```

6.2.4 Pushing the code to your GitHub repositories

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

  cd <folder path>

- Initialize your 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 “Changes have been committed.”

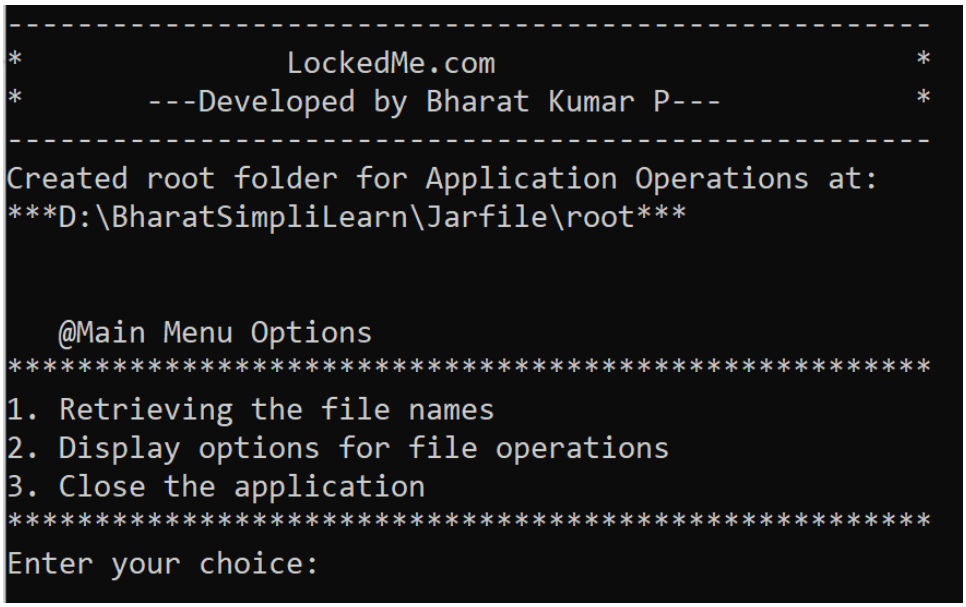
- Push the files to the folder you initially created using the following command:

  git push -u origin master

7. Project Appearance and user Interaction

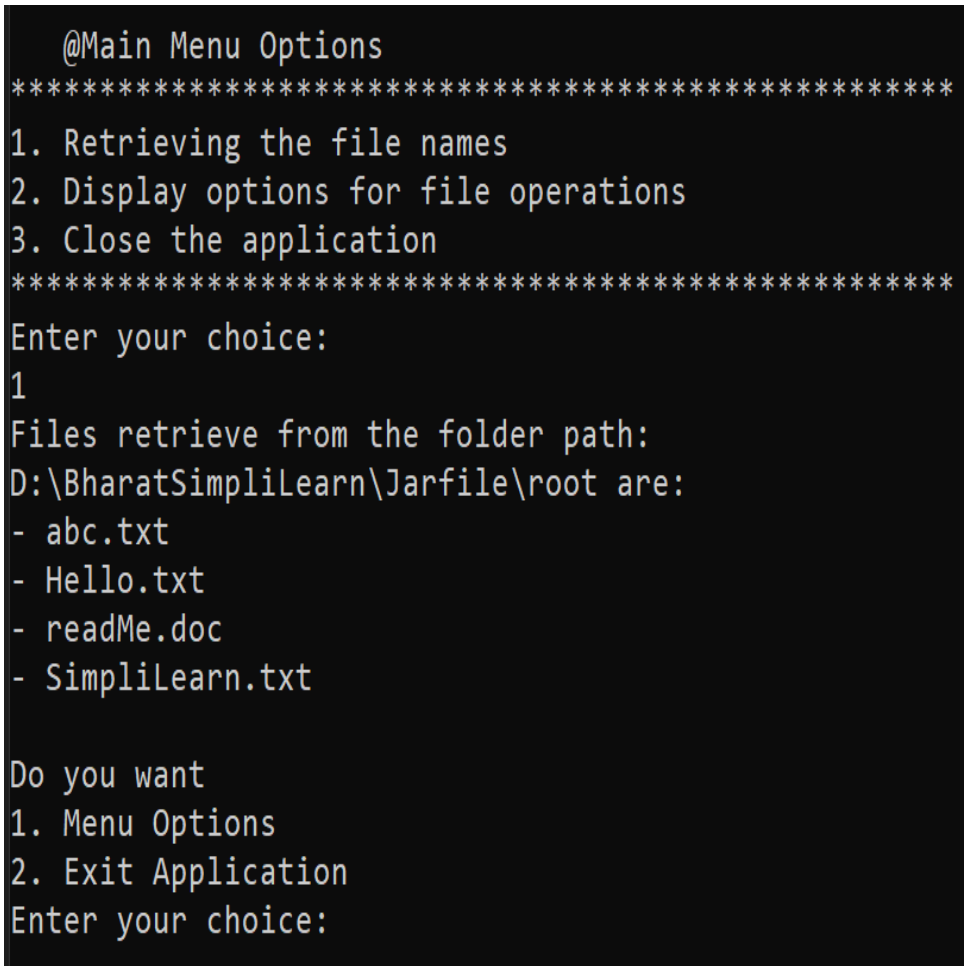
7.1 Welcome Screen

- Application name and the developer details
- The details of the user interface such as options displaying the user interaction information
- Features to accept the user input to select one of the options listed



7.2 Retrieving Files

- First option to return the current file names in ascending order. The root directory can be either empty or contain few files or folders in it



7.3 Display File Menu Operations

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

```
   @Main Menu Options
*****************************************************
1. Retrieving the file names
2. Display options for file operations
3. Close the application
*****************************************************
Enter your choice:
2

   @File Opertions Menu
*****************************************************
1. Add a user specified file to the application
2. Delete a user specified file from the application
3. Search a user specified file from the application
4. Go to Main Menu
5. Close the application
*****************************************************
Enter your choice:
```

- Add a file to the existing directory list

```
   @File Opertions Menu
*****************************************************
1. Add a user specified file to the application
2. Delete a user specified file from the application
3. Search a user specified file from the application
4. Go to Main Menu
5. Close the application
*****************************************************
Enter your choice:
1
Enter the name of the File:
SimpliLearn.txt
How many line you want to write into file:
2
Enter the text for line 1:
This is SimpliLearn
Enter the text for line 2:
You are here for Java FSD!
File added successfully!

Do you want
1. File Operations Menu
2. Exit Application
Enter your choice:
```

- Delete a user specified file from the existing directory

```
Do you want
1. File Operations Menu
2. Exit Application
Enter your choice:
1

   @File Opertions Menu
*****************************************************
1. Add a user specified file to the application
2. Delete a user specified file from the application
3. Search a user specified file from the application
4. Go to Main Menu
5. Close the application
*****************************************************
Enter your choice:
2
Enter name of the File to delete:
SimpliLearn.txt
File is succesfully deleted!

Do you want
1. File Operations Menu
2. Exit Application
Enter your choice:
```

- Return a message if FNF (File not found)

```
   @File Opertions Menu
*****************************************************
1. Add a user specified file to the application
2. Delete a user specified file from the application
3. Search a user specified file from the application
4. Go to Main Menu
5. Close the application
*****************************************************
Enter your choice:
2
Enter name of the File to delete:
bharat.txt
File does not exist in the path to delete!

Do you want
1. File Operations Menu
2. Exit Application
Enter your choice:
```

- Search a user specified file from the main directory
- Display the result upon successful operation

```
   @File Opertions Menu
*****************************************************
1. Add a user specified file to the application
2. Delete a user specified file from the application
3. Search a user specified file from the application
4. Go to Main Menu
5. Close the application
*****************************************************
Enter your choice:
3
Enter name of the File to search:
readMe.doc
File found in the location

Do you want
1. File Operations Menu
2. Exit Application
Enter your choice:
```

- Display the result upon unsuccessful operation

```
Do you want
1. File Operations Menu
2. Exit Application
Enter your choice:
1

   @File Opertions Menu
*****************************************************
1. Add a user specified file to the application
2. Delete a user specified file from the application
3. Search a user specified file from the application
4. Go to Main Menu
5. Close the application
*****************************************************
Enter your choice:
3
Enter name of the File to search:
name.csv
File not found!

Do you want
1. File Operations Menu
2. Exit Application
Enter your choice:
```

- Option to navigate back to the main context

```
   @File Opertions Menu
*****************************************************
1. Add a user specified file to the application
2. Delete a user specified file from the application
3. Search a user specified file from the application
4. Go to Main Menu
5. Close the application
*****************************************************
Enter your choice:
4
main menu

   @Main Menu Options
*****************************************************
1. Retrieving the file names
2. Display options for file operations
3. Close the application
*****************************************************
Enter your choice:
```

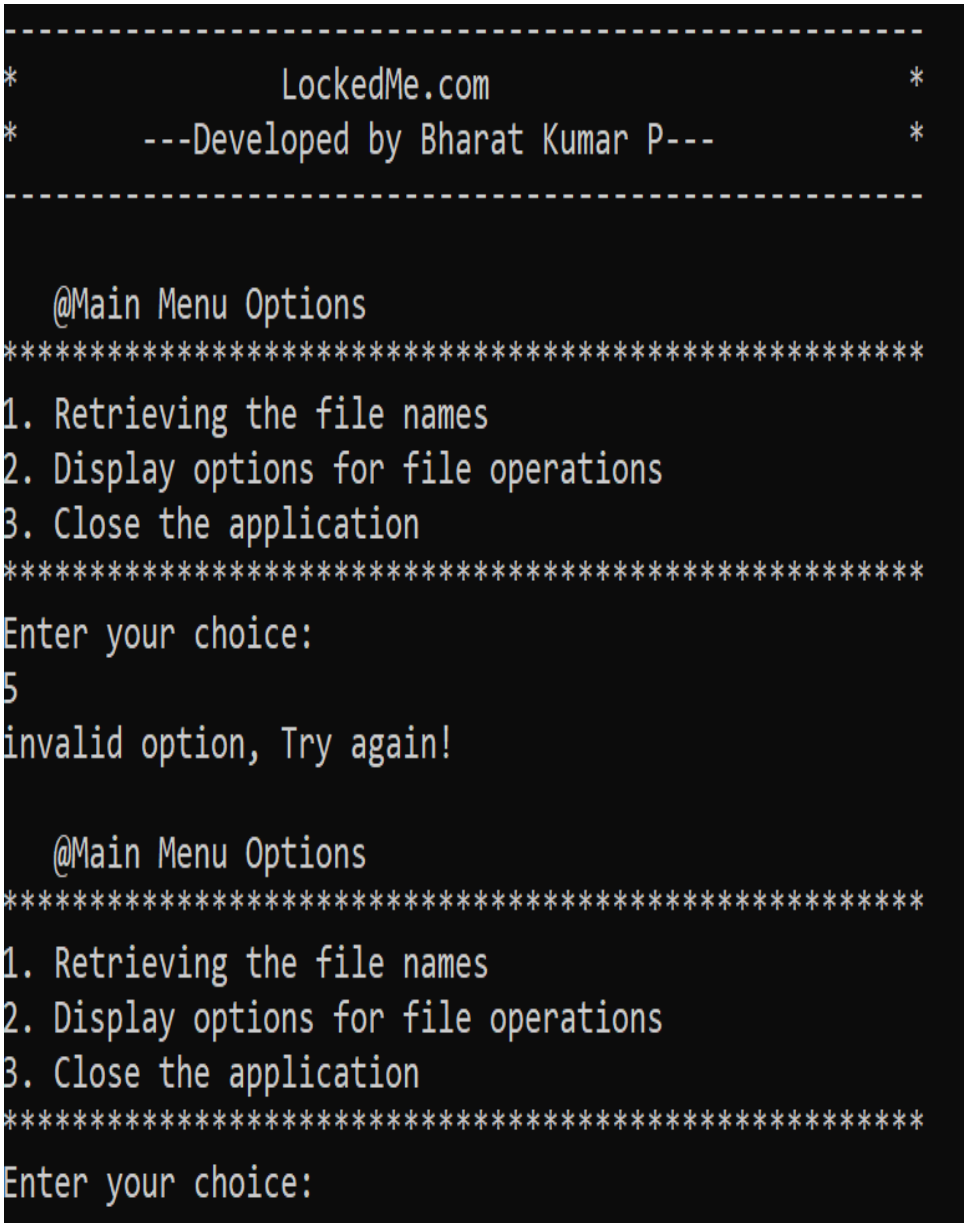
7.4 Close Application

- Third option to close the application

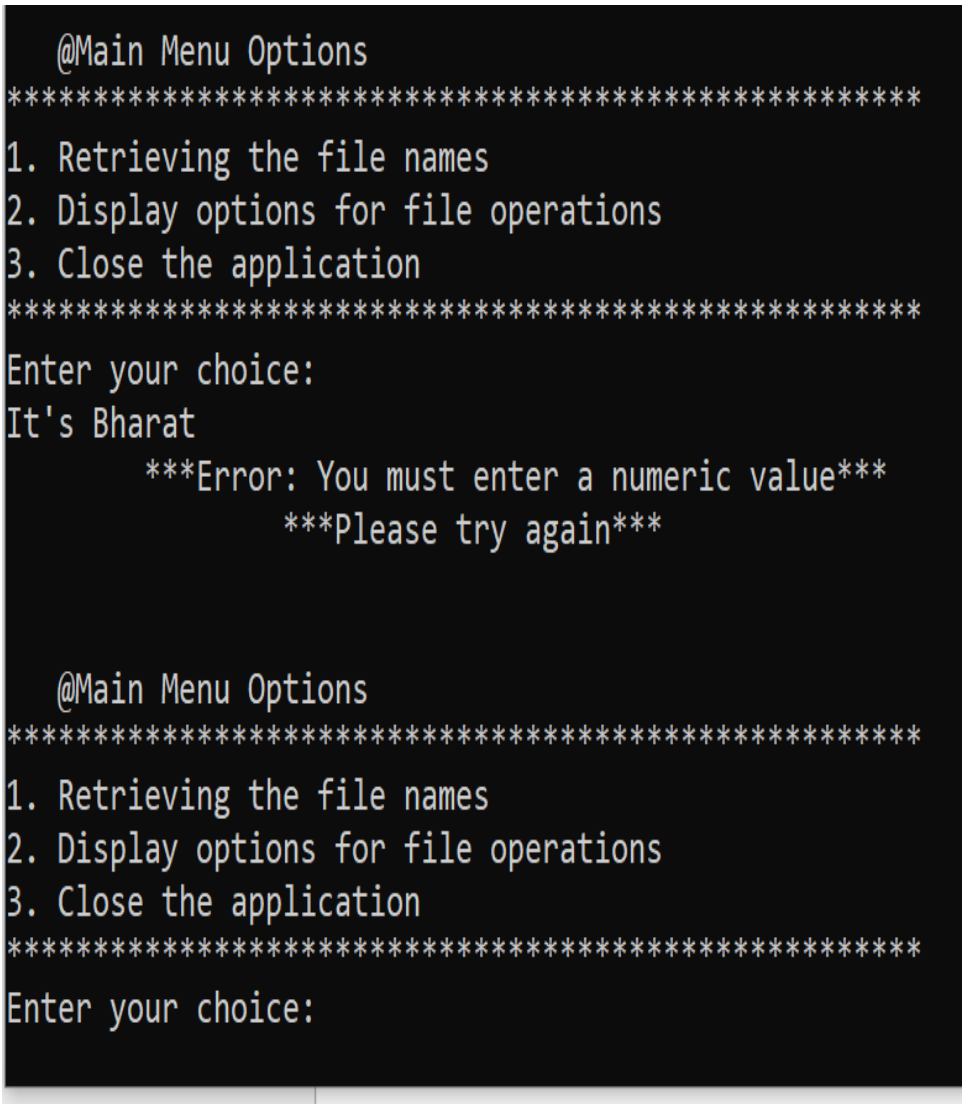
```
   @Main Menu Options
*****************************************************
1. Retrieving the file names
2. Display options for file operations
3. Close the application
*****************************************************
Enter your choice:
3
Thank you, closing the Application

D:\BharatSimpliLearn\Jarfile>
```

7.5 Exception Handling

- Users enter any other options then the Menu options



- User inputs a String instead of a numeric value in options



7.6 Additional Feature

- Options to exit or go to main menu after a completion of an operation by user.

```
   @Main Menu Options
*****************************************************
1. Retrieving the file names
2. Display options for file operations
3. Close the application
*****************************************************
Enter your choice:
1
Files retrieve from the folder path:
D:\BharatSimpliLearn\Jarfile\root are:
- abc.txt
- Hello.txt
- readMe.doc

Do you want
1. Menu Options
2. Exit Application
Enter your choice:
```

- Options to exit or go to file menu after a completion of an operation by user.

```
   @File Opertions Menu
*****************************************************
1. Add a user specified file to the application
2. Delete a user specified file from the application
3. Search a user specified file from the application
4. Go to Main Menu
5. Close the application
*****************************************************
Enter your choice:
1
Enter the name of the File:
Hello.txt
How many line you want to write into file:
0
File added successfully!

Do you want
1. File Operations Menu
2. Exit Application
Enter your choice:
```

8. Application Unique Selling Points

- The application to take user inputs without any exceptions or errors. To provide options after every completion of task using menu options in order to ease the process.
- To terminate the application with appropriate options after every task.
- The application take care of creating directory in the present working folder making ease for user to avoid extra work of creating a folder separately.
- User is also provided the option to write content if they want into the newly created file.
- The application also allows user to delete files which are not empty.
- The user can seamlessly switch between options or return to previous menu even after any required operation like adding, searching, deleting, or retrieving of files is performed.
- User can retrieve files list in the workspace in ascending order.
- The application is designed with modularity in mind. Making work of the user very to by providing the required options to time to time to avoid complexity of the Application.

9. Conclusion

The purpose of this Application was to identify effective strategies for dealing with Virtual key for Repositories by making the process of the application to benefit the users. It conserves resources and time and is one of those which helps user to ease the process and have best virtual key for their repositories. Providing users, the options to retrieve files in ascending order, add files of their choice into the repository, more options like delete and search for a file. This Application also manages the exception and error for users to guide them in a better way and give the best solution while working. Future exploration into behavior modification techniques could be useful to finding further techniques to work with the repositories.