VIRTUAL KEY REPOSITORIES

Problem Statement:

Company Lockers Pvt. Ltd. hired you as a Full Stack Developer. They aim to digitize their products and chose LockedMe.com as their first project to start with. You're asked to develop a prototype of the application. The prototype of the application will be then presented to the relevant stakeholders for the budget approval. Your manager has set up a meeting where you're asked to present the following in the next 15 working days (3 weeks):

- Specification document - Product's capabilities, appearance, and user interactions
- Number and duration of sprints required
- Setting up Git and GitHub account to store and track your enhancements of the prototype
- Java concepts being used in the project
- Data Structures where sorting and searching techniques are used.
- Generic features and three operations:
    - Retrieving the file names in an ascending order
    - Business-level operations:
        - Option to add a user specified file to the application
        - Option to delete a user specified file from the application
        - Option to search a user specified file from the application

    - Navigation option to close the current execution context and return to the main context

- Option to close the application.

This document contains sections for:

- Sprint planning and task completion
- Core concepts used in the project
- Flow chart of the application
- Demonstrating the product capabilities, appearance and user interactions
- Unique selling points of the application
- Conclusion

The code for this project is hosted at https://github.com/Jothiprakash-8/Phase1project.git .

This project is developed by Ayya Jothi Prakash.

Sprint 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

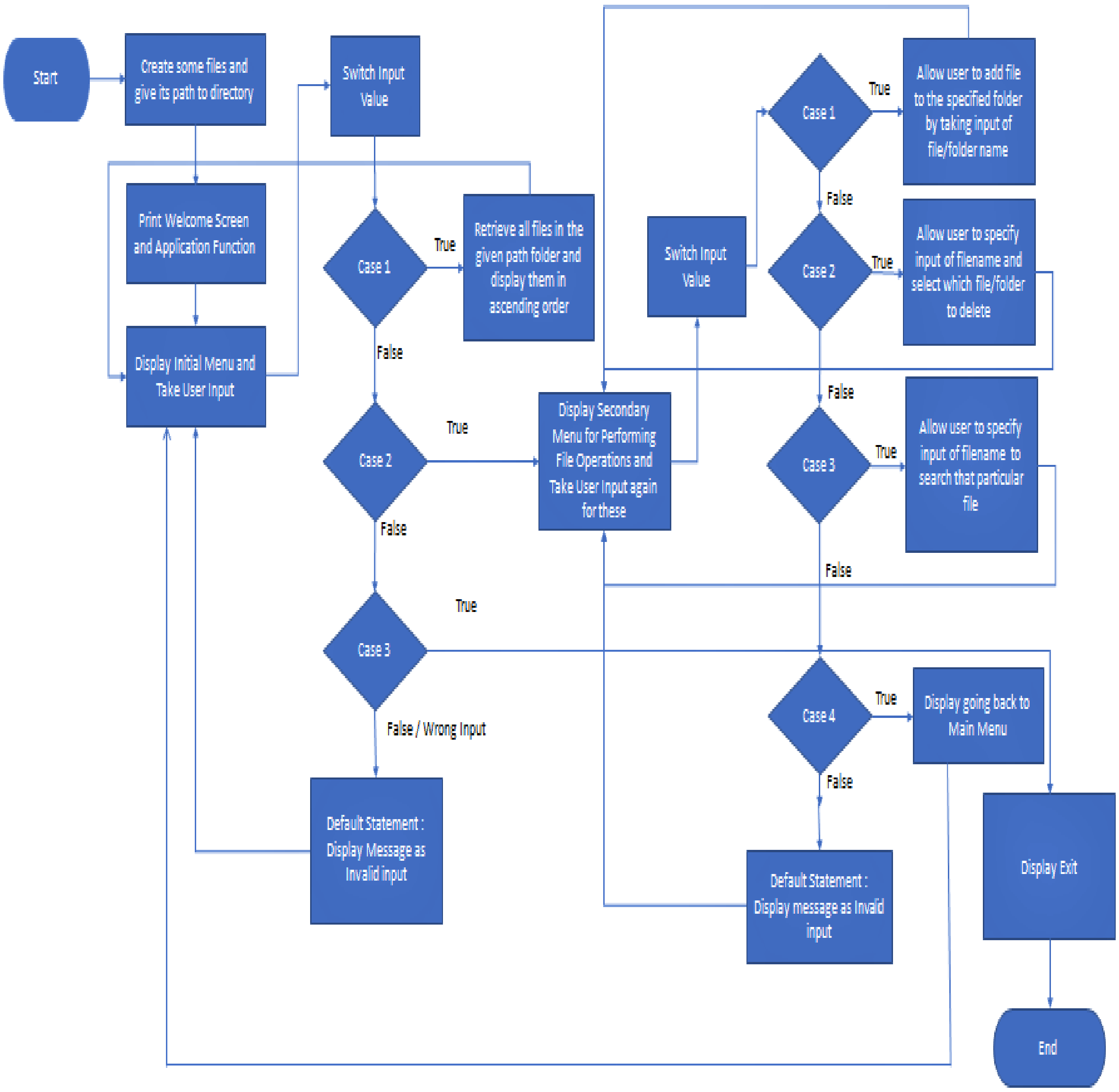
- Pushing code to GitHub.

- Creating this specification document highlighting application capabilities, appearance, and user interactions.

Core concepts used in this project:

Collections framework, File handling, Searching, Sorting, Exception handling.

Flow chart of the application:


Start
Create some files and give its path to directory
Print Welcome Screen and Application Function
Display Initial Menu and Take User Input
Switch Input Value
Case 1
True
Retrieve all files in the given path folder and display them in ascending order
False
Case 2
True
Display Secondary Menu for Performing File Operations and Take User Input again for these
False
Case 3
True
False / Wrong Input
Default Statement : Display Message as Invalid input
Switch Input Value
Case 1
True
Allow user to add file to the specified folder by taking input of file/folder name
False
Case 2
True
Allow user to specify input of filename and select which file/folder to delete
False
Case 3
True
Allow user to specify input of filename to search that particular file
False
Case 4
True
Display going back to Main Menu
False
Default Statement : Display message as Invalid input
Display Exit
End

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 sorting the files in ascending order.

(Orderfile.java)

3. Writing a program in Java to display the contents in main menu

(Application.java)

4. Writing a program in Java to do fileoperations

(Fileoperations.java)

5. 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 Phase1finalproject and click on “Finish.”
- Select your project and go to File -> New -> Class.
- Enter respective class names.

Step 2: Writing a program in Java for sorting the files in ascending order. (Orderfile.java)

```
package project;
import java.io.*;
import java.io.File;
import java.util.Set;
import java.util.TreeSet;

public class Orderfile {
	static String directory= "C://users//ayyaj/P1project";
	public static void sort() {

		File[] files = new File(directory).listFiles();
		Set<String> s = new TreeSet<>();
		for(File file : files) {
			if (!file.isFile()) {
				continue;
			}
			s.add(file.getName());
		}
		s.forEach(i->System.out.println(i));

	}

}
```

Step 3: Writing a program in Java to display the contents in main menu (Application.java)

```
package project;

import java.util.Scanner;
import project.Orderfile;
import project.Fileoperations;

public class Application {
	static Scanner sn = new Scanner(System.in);

	public static void display() {
		System.out.println("Project:LockedMe.com");
		System.out.println("Developed by Ayya Jothi Prakash");
		System.out.println("Description: Application to add, delete, and 
search files");
	}
	public static void choose() {
		System.out.println("");
		System.out.println("Main Menu");
		System.out.println("Press 1 to Retrieving the file names in an 
ascending order");
		System.out.println("Press 2 to Perform business level operations");
		System.out.println("Press 3 to Close the application");
```

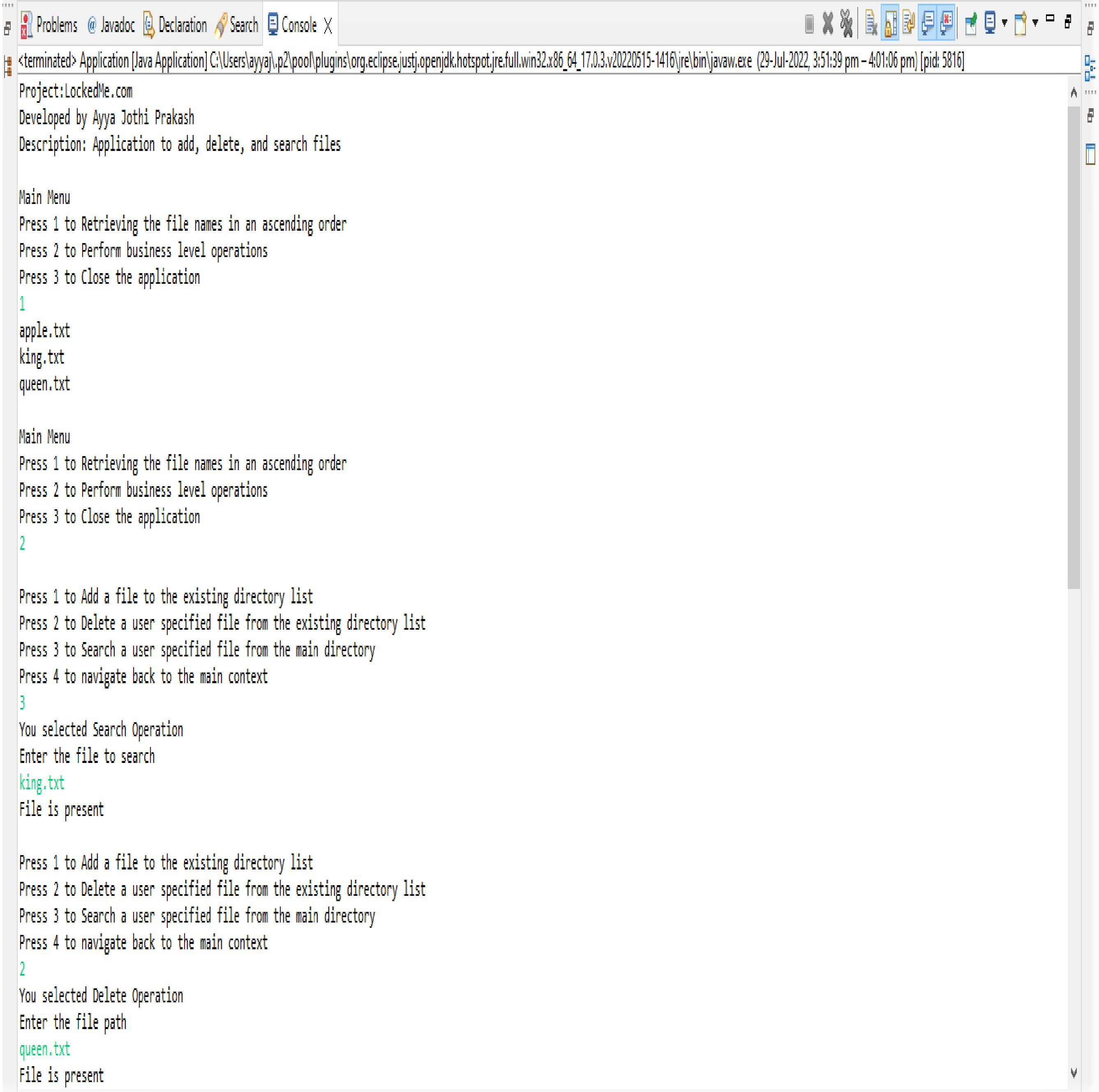
```
        int choice = sn.nextInt();
        cases(choice);
    }
    public static void cases(int a) {
        switch(a) {
            case 1:
                Orderfile.sort();
                break;
            case 2:
                Fileoperations.fileoperations();
                break;
            case 3:
                System.out.println("Exit");
                System.exit(0);
                break;
            default:
                System.out.println("Invalid input");
        }
        choose();
    }

    public static void main(String[] args) {
        display();
        choose();


    }

}
```

Step 4: Writing a program in Java to do fileoperations (Fileoperations.java).

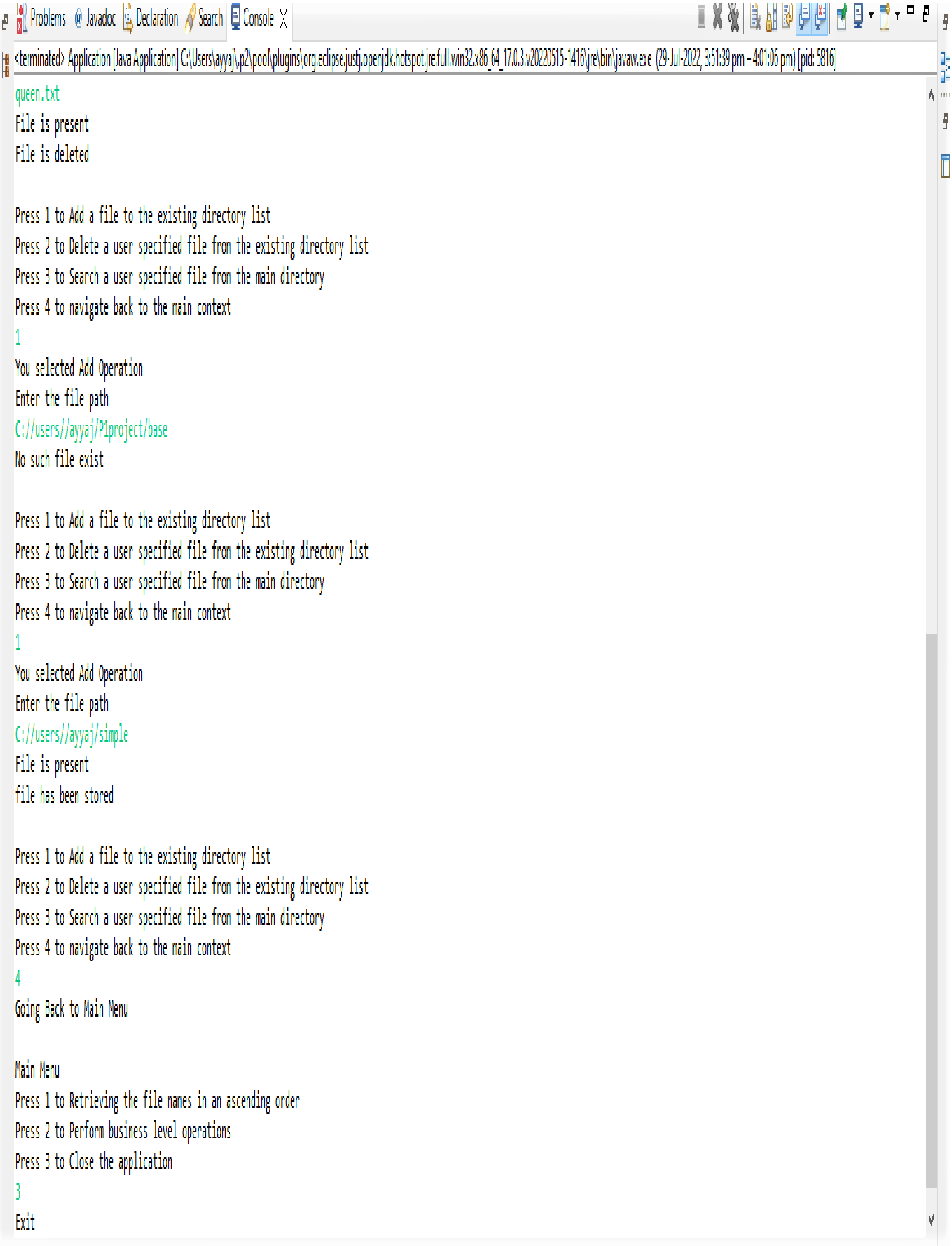
```
package project;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.InvalidPathException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.Scanner;
import project.Application;

public class Fileoperations {
    static Scanner sn = new Scanner(System.in);
    static String directory = "C://users//ayyaj/P1project";

    public static void fileoperations() {
        System.out.println("");
        System.out.println("Press 1 to Add a file to the existing directory 
list");
        System.out.println("Press 2 to Delete a user specified file from the 
existing directory list");
        System.out.println("Press 3 to Search a user specified file from the 
main directory");
```

```java
        System.out.println("Press 4 to navigate back to the main context");
        String s1 = sn.nextLine();
        cases(s1);
    }

    public static void cases(String s) {
        switch(s) {
            case "1":
                System.out.println("You selected Add Operation");
                add();
                break;
            case "2":
                System.out.println("You selected Delete Operation");
                delete();
                break;
            case "3":
                System.out.println("You selected Search Operation");
                search();
                break;
            case "4":
                System.out.println("Going Back to Main Menu");
                Application.choose();
                break;
            default:
                System.out.println("Invalid input");
        }
        fileoperations();
    }


    public static void add() throws InvalidPathException {
        System.out.println("Enter the file path");
        String input = sn.nextLine();
        Path path;
        try {
            path = Paths.get(input);
        } catch (Exception e) {
            System.out.println("Invalid input");
            return;
        }

        if (!Files.exists(path)) {
            System.out.println("No such file exist");
            return;
        }else {
            System.out.println("File is present");

        }

        String newPath = directory + "/" + path.getFileName();
        int x = 0;
        while (Files.exists(Paths.get(newPath))) {
            x++;
            newPath = directory + "/" + x + "_" + path.getFileName();
        }

        try {
            Files.copy(path,  Paths.get(newPath));
            System.out.println("file has been stored");
        } catch (IOException e) {
            System.out.println("Not able to store the file");
            System.out.println(e);
        }

    }



    public static void delete() throws InvalidPathException {
        System.out.println("Enter the file path");
        String inputfile = sn.nextLine();
        String Path = directory + "/" + inputfile;
        Path path;

        try {
            path = Paths.get(Path);
        } catch (Exception e) {
            System.out.println("Invalid input");
            return;
        }

        if (!Files.exists(path)) {
            System.out.println("File not found");
            return;
        } else {
            System.out.println("File is present");
        }

        File Delete = new File(Path);
        try {
            Delete.delete();
            System.out.println("File is deleted");
        }
        catch (Exception e) {

            System.out.println("Not able to delete file");
            System.out.println(e);
        }
    }



    public static void search() throws InvalidPathException{
        System.out.println("Enter the file to search");
        String input = sn.nextLine();
        String Path = directory + "/" + input;
        Path path;

        try {
            path = Paths.get(Path);
        } catch (Exception e) {
            System.out.println("Invalid input");
            return;
        }

        if(!Files.exists(path)) {
            System.out.println("No such file exist");
            return;
```

```
                } else {
                        System.out.println("File is present");
                }

        }

}
```



```
<terminated> Application [Java Application] C:\Users\ayyaj\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20220515-1416\jre\bin\javaw.exe  (29-Jul-2022, 3:51:39 pm – 4:01:06 pm) [pid: 5816]
Project:LockedMe.com
Developed by Ayya Jothi Prakash
Description: Application to add, delete, and search files

Main Menu
Press 1 to Retrieving the file names in an ascending order
Press 2 to Perform business level operations
Press 3 to Close the application
1
apple.txt
king.txt
queen.txt

Main Menu
Press 1 to Retrieving the file names in an ascending order
Press 2 to Perform business level operations
Press 3 to Close the application
2

Press 1 to Add a file to the existing directory list
Press 2 to Delete a user specified file from the existing directory list
Press 3 to Search a user specified file from the main directory
Press 4 to navigate back to the main context
3
You selected Search Operation
Enter the file to search
king.txt
File is present

Press 1 to Add a file to the existing directory list
Press 2 to Delete a user specified file from the existing directory list
Press 3 to Search a user specified file from the main directory
Press 4 to navigate back to the main context
2
You selected Delete Operation
Enter the file path
queen.txt
File is present
```



```
<terminated> Application [Java Application] C:\Users\ayyaj\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.3.v20220515-1416\jre\bin\javaw.exe (29-Jul-2022, 3:51:39 pm – 4:01:06 pm) [pid: 5816]
queen.txt
File is present
File is deleted

Press 1 to Add a file to the existing directory list
Press 2 to Delete a user specified file from the existing directory list
Press 3 to Search a user specified file from the main directory
Press 4 to navigate back to the main context
1
You selected Add Operation
Enter the file path
C://users//ayyaj/P1project/base
No such file exist

Press 1 to Add a file to the existing directory list
Press 2 to Delete a user specified file from the existing directory list
Press 3 to Search a user specified file from the main directory
Press 4 to navigate back to the main context
1
You selected Add Operation
Enter the file path
C://users//ayyaj/simple
File is present
file has been stored

Press 1 to Add a file to the existing directory list
Press 2 to Delete a user specified file from the existing directory list
Press 3 to Search a user specified file from the main directory
Press 4 to navigate back to the main context
4
Going Back to Main Menu

Main Menu
Press 1 to Retrieving the file names in an ascending order
Press 2 to Perform business level operations
Press 3 to Close the application
3
Exit
```

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 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.

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

3. 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.

4. The application is designed with modularity in mind. Even if one wants to update the path, they can change it through the source code.

Conclusion:

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

- Allowing user to append data to the file.