**LockedMe – Virtual Key for Repositories**

This document contains sections for:

* [Sprint planning and Task completion](#Sprint_plan)
* [Core concepts used in project](#Core_concepts)
* [Flow of the Application](#Flow).
* [Demonstrating the product capabilities, appearance, and user interactions.](#Product_capability)
* [Unique Selling Points of the Application](#USP)
* [Conclusions](#Conclusions)

The code for this project is hosted at <https://github.com/Dwarakamayee-k/Phase1.git>

The project is developed by Dwarakamayee Kanuparthy

## Sprints planning and Task completion

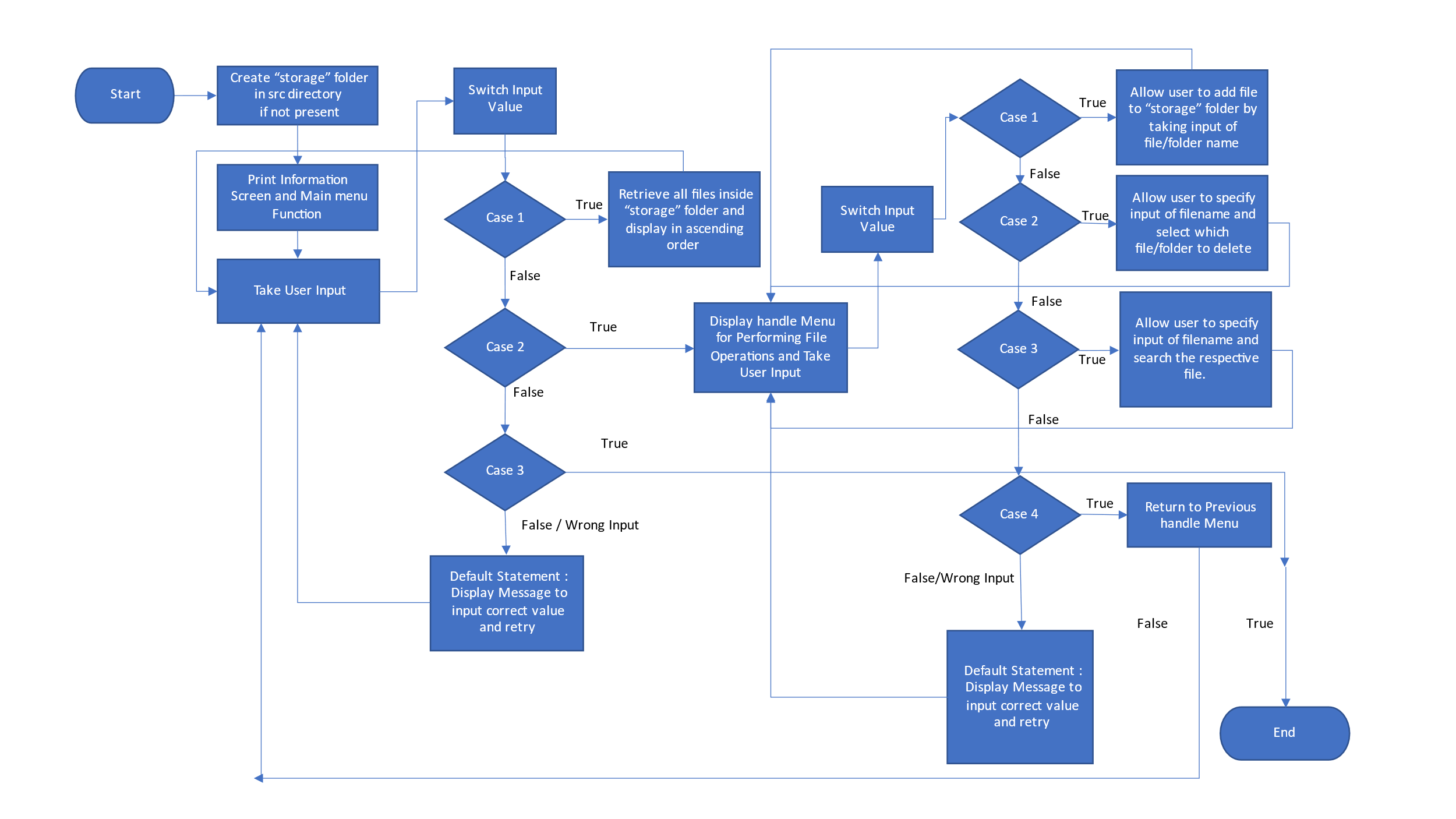
The project is planned to be completed in 1 sprint. Tasks assumed to be completed in the sprint are:

* 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, Flow Control, Exception Handling

## 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](#Step_1)
2. [Writing a program in Java for the entry point of the application (App**.java**)](#Step_2)
3. [Writing a program in Java to print list of files in ascending order (Ascending**.java**)](#Step_3)
4. [Writing a program in Java to handle file operations selected by user (**Operations.java**)](#Step_4)
5. [Pushing the code to GitHub repository](#Step_6)

## **Step 1:** Creating a new project in Eclipse

* Open Eclipse
* Go to File -> New -> Project -> Java Project -> Next.
* Type in project name “Phase1” and click on “Finish.”
* Select your project and go to File -> New -> Class.
* Enter **App.java** in 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 (**App.java**)

import java.util.\*;

public class App {

static Scanner sn = new Scanner(System.in);

public static void info() {

System.out.println("LockedMe.com");

System.out.println("Developed by Dwarakamayee Kanuparthy");

System.out.println("Description: Application/Prototype to add, delete, and search files");

}

public static void main() {

System.out.println("");

System.out.println("Main Menu");

System.out.println("Press 1 to show file in Ascending Order");

System.out.println("Press 2 to view file operations");

System.out.println("Press 3 to Exit from the application");

int choice = sn.nextInt();

handle(choice);

}

public static void handle(int num) {

switch(num) {

case 1:

Ascending.ascendingOrder();

break;

case 2:

Operations.FileOperations();

break;

case 3:

System.out.println("Exit");

System.exit(0);

break;

default:

System.out.println("Invalid input");

}

main();

}

public static void main(String[] args) {

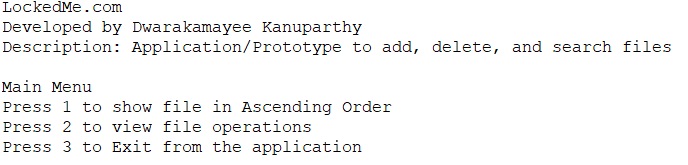
info();

main();

}

}

**Output:**



## **Step 3:** Writing a program in Java to print list of files in ascending order (Ascending**.java**)

* Select your project and go to File -> New -> Class.
* Enter Ascending in class name and click on “Finish.”

**Program:**

import java.io.\*;

import java.util.\*;

public class Ascending {

static String directory= "src/storage";

public static void ascendingOrder() {

File[] files = new File(directory).listFiles();

Set<String> a = new TreeSet<>();

for(File file : files) {

if (!file.isFile()) {

continue;

}

a.add(file.getName());

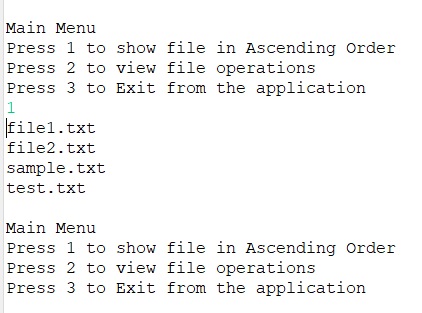
}

a.forEach(i->System.out.println(i));

}

}

**Output:**



## **Step 4:** Writing a program in Java to handle file operations selected by user (**Operations**.**java**)

* Select your project and go to File -> New -> Class.
* Enter **Operations** in class name and click on “Finish.”

**Program:**

import java.io.\*;

import java.util.\*;

public class Operations {

static Scanner sn=new Scanner(System.in);

public static void FileOperations() {

System.out.println("");

System.out.println("Press 1 to Add a file");

System.out.println("Press 2 to Delete a file");

System.out.println("Press 3 to Search a file");

System.out.println("Press 4 to go Back to the Main Menu");

String choice = sn.nextLine();

try {

handle(choice);

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public static void handle(String num)throws IOException {

switch(num) {

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");

App.main();

break;

default:

System.out.println("Invalid input");

}

FileOperations();

}

// to add a file

public static void add()throws IOException{

String path="src/storage/";

Scanner sc=new Scanner(System.in);

ArrayList<String> al=new ArrayList<>();

while(true) {

System.out.println("enter the file name");

String filename=sc.next();

String finalfile=path+filename;

File f=new File(finalfile);

boolean res=f.createNewFile();

if(res==false) {

System.out.println("file not created");

}

else {

al.add(filename);

System.out.println("file is created");

}

System.out.println(al);

break;

}

}

// to delete a file

public static void delete() throws IOException{

String path="src/storage/";

Scanner sc=new Scanner(System.in);

System.out.println("enter the file name to be deleted:");

String filedel=sc.next();

String finalfile=path+filedel;

File f=new File(finalfile);

f.delete();

System.out.println("file is deleted:");

}

//to search a file

public static void search() throws IOException{

String path="src/storage/";

Scanner sc=new Scanner(System.in);

File f=new File(path);

System.out.println("enter the file name");

String filesearch=sc.next();

File filen[]=f.listFiles();

int flag=0;

for(File ff:filen) {

if(ff.getName().equals(filesearch)) {

flag=1;

break;

}

else {

flag=0;

}

}

if(flag==1) {

System.out.println("file is found");

}

else {

System.out.println("file is not found");

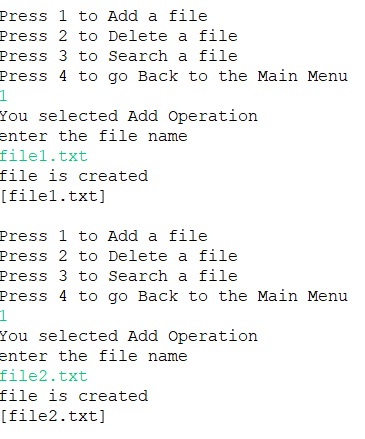
}

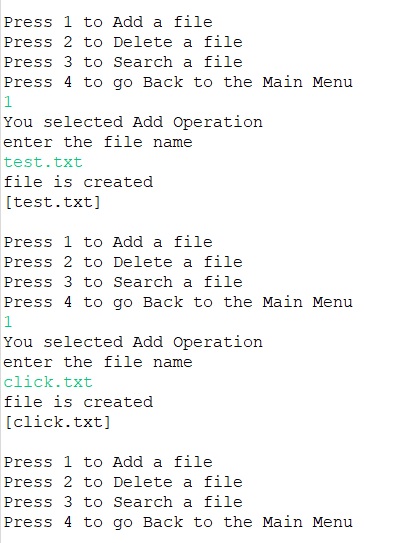
}

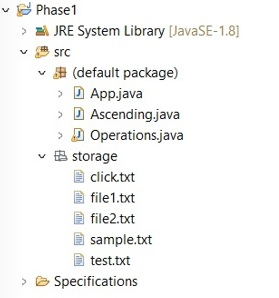
}

**Output:**

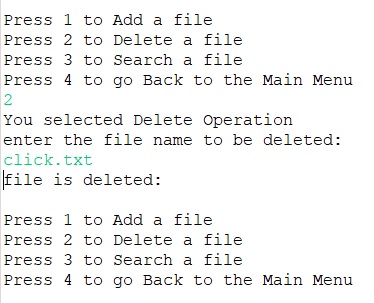
**Adding Files:**

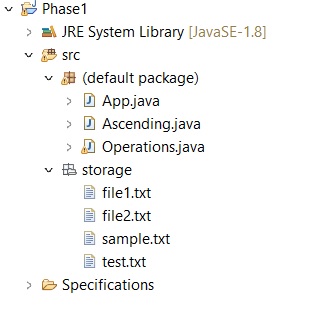




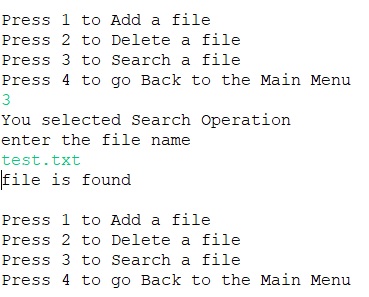


**Deleting Files:**

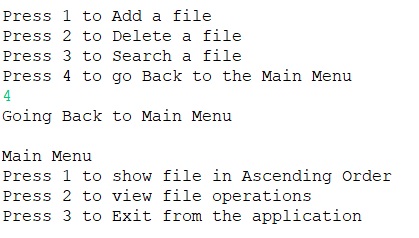




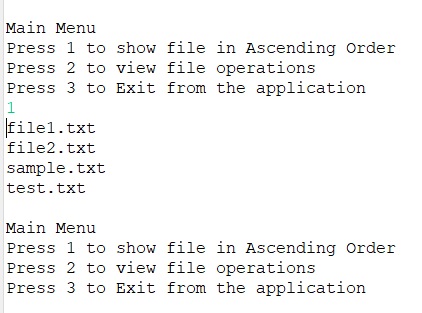
**Searching File:**



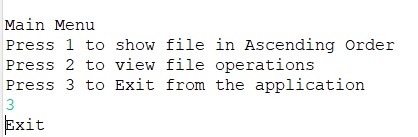
**Previous Menu:**



**Ascending Order Files:**



**Exit Program:**



## **Step 5:** 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 allows user to create new files and retrieve files in ascending order
3. The application also allows user to delete files.
4. 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.
5. 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.