## Sprints planning and Task completion

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

* 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, Exception Handling, Streams API

Demonstrating the product capabilities, appearance, and user interactions

## **Step 1:** Creating a new project in intellij idea

* Open intellij
* Go to File -> New -> Project -> Name->Enter the name of project
* Type in any project name and click on “Create.”
* Select your project and go to src -> New -> java class.
* Enter **LockedMe** 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 (**LockedMe.java**)

**Step 2.1:** writing method to show primary menu or secondary meanu along with output :

public static final String *MAIN\_MENU\_PROMPT* =  
 "\nMAIN MENU - Select any of the following: \n"+  
 "1 -> List files in directory\n"+  
 "2 -> Add, Delete or Search\n"+  
 "3 -> Exit Program";  
  
public static final String *SECONDARY\_MENU\_PROMPT* =  
 " \nSelect any of the following: \n"+  
 " a -> Add a file\n"+  
 " b -> Delete a file\n"+  
 " c -> Search a file\n"+  
 " d -> GoBack";  
  
void showPrimaryMenu() {  
 System.*out*.println(*MAIN\_MENU\_PROMPT*);  
 try{  
 Scanner scanner = new Scanner(System.*in*);  
 int option = scanner.nextInt();  
 switch (option){  
 case 1 : {  
 showFiles();  
 showPrimaryMenu();  
 }  
 case 2 : {  
 showSecondaryMenu();  
 }  
 case 3 : {  
 System.*out*.println("Thank You");  
 System.*exit*(0);  
 }  
 default: showPrimaryMenu();  
 }  
 }  
 catch (Exception e){  
 System.*out*.println("Please enter 1, 2 or 3");  
 showPrimaryMenu();  
 }  
}  
  
void showSecondaryMenu() {  
 System.*out*.println(*SECONDARY\_MENU\_PROMPT*);  
 try{  
 Scanner scanner = new Scanner(System.*in*);  
 char[] input = scanner.nextLine().toLowerCase().trim().toCharArray();  
 char option = input[0];  
  
 switch (option){  
 case 'a' : {  
 System.*out*.print(" Adding a file...Please Enter a File Name : ");  
 String filename = scanner.next().trim().toLowerCase();  
 addFile(filename);  
 break;  
 }  
 case 'b' : {  
 System.*out*.print(" Deleting a file...Please Enter a File Name : ");  
 String filename = scanner.next().trim();  
 deleteFile(filename);  
 break;  
 }  
 case 'c' : {  
 System.*out*.print(" Searching a file...Please Enter a File Name : ");  
 String filename = scanner.next().trim();  
 searchFile(filename);  
 break;  
 }  
 case 'd' : {  
 System.*out*.println("Going Back to MAIN menu");  
 showPrimaryMenu();  
 break;  
 }  
 default : System.*out*.println("Please enter a, b, c or d");  
 }  
 showSecondaryMenu();  
 }  
 catch (Exception e){  
 System.*out*.println("Please enter a, b, c or d");  
 showSecondaryMenu();  
 }  
}

**Output:**

path : C:\Users\user\IdeaProjects\FIRST\_PROJECT\files

MAIN MENU - Select any of the following:

1 -> List files in directory

2 -> Add, Delete or Search

3 -> Exit Program

2

Select any of the following:

a -> Add a file

b -> Delete a file

c -> Search a file

d -> GoBack

**Step 2.2:** Writing method to show file for File Operation:

void showFiles() {  
 if (folder\_name.list().length==0)  
 System.*out*.println("The folder is empty");  
 else {  
 String[] list = folder\_name.list();  
 System.*out*.println("The files in "+ folder\_name +" are :");  
 Arrays.*sort*(list);  
 for (String str:list) {  
 System.*out*.println(str);  
 }  
 }  
}

**Output:**

MAIN MENU - Select any of the following:

1 -> List files in directory

2 -> Add, Delete or Search

3 -> Exit Program

1

The folder is empty

**Step 2.3:** Writing method to Add file for File Operation:

void addFile(String filename) throws IOException {  
 File filepath = new File(folder\_name +"/"+filename);  
 String[] list = folder\_name.list();  
 for (String file: list) {  
 if (filename.equalsIgnoreCase(file)) {  
 System.*out*.println("File " + filename + " already exists at " + folder\_name);  
 return;  
 }  
 }  
 filepath.createNewFile();  
 System.*out*.println("File "+filename+" added to "+ folder\_name);  
}

Output:

MAIN MENU - Select any of the following:

1 -> List files in directory

2 -> Add, Delete or Search

3 -> Exit Program

2

Select any of the following:

a -> Add a file

b -> Delete a file

c -> Search a file

d -> GoBack

a

Adding a file...Please Enter a File Name : lokesh

File lokesh added to C:\Users\user\IdeaProjects\FIRST\_PROJECT\files

Select any of the following:

a -> Add a file

b -> Delete a file

c -> Search a file

d -> GoBack

step 2.4: Writing method Delete file for File Operation:

void deleteFile(String filename) {  
 File filepath = new File(folder\_name +"/"+filename);  
 String[] list = folder\_name.list();  
 for (String file: list) {  
 if (filename.equals(file) && filepath.delete()) {  
 System.*out*.println("File " + filename + " deleted from " + folder\_name);  
 return;  
 }  
 }  
 System.*out*.println("Delete Operation failed. FILE NOT FOUND");  
}

Output :

Select any of the following:

a -> Add a file

b -> Delete a file

c -> Search a file

d -> GoBack

b

Deleting a file...Please Enter a File Name : lokesh

File lokesh deleted from C:\Users\user\IdeaProjects\FIRST\_PROJECT\files

Select any of the following:

a -> Add a file

b -> Delete a file

c -> Search a file

d -> GoBack

step 2.5: Writing method Search file for File Operation:

void searchFile(String filename) {  
 String[] list = folder\_name.list();  
 for (String file: list) {  
 if (filename.equals(file)) {  
 System.*out*.println("FOUND : File " + filename + " exists at " + folder\_name);  
 return;  
 }  
 }  
 System.*out*.println("File NOT found (FNF)");  
}

**Output:**

Select any of the following:

a -> Add a file

b -> Delete a file

c -> Search a file

d -> GoBack

c

Please enter a, b, c or d

Select any of the following:

a -> Add a file

b -> Delete a file

c -> Search a file

d -> GoBack

c

Searching a file...Please Enter a File Name : lokesh

File NOT found (FNF)

Select any of the following:

a -> Add a file

b -> Delete a file

c -> Search a file

d -> GoBack

Because we Delete the file all ready so we get File Not found (FNF)

**Step 2.5:** At the end we call the method

public static void main(String[] args) {  
 System.*out*.println();  
 LockedMe menu = new LockedMe();  
 menu.showPrimaryMenu();  
 }  
}

## **Step 3:** 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**