|  |
| --- |
| **LockedMe.com**  **(Specifications and Scrum Details)** |

**The sections in this document are:-**

|  |
| --- |
| Sprint Planning and Task Completion |
| Core concepts used in the Project |
| Flow Diagram of the Application |
| Demonstrating the Project Capabilities, appearance, and user Interactions |
| Unique selling points of the Application |
| Conclusions |

**The code for this project is hosted at**

|  |
| --- |
| **Git-Hub link:-** |

**The project is developed by Swathi Kanduri**

**Sprint Planning and Task Completion**

|  |
| --- |
| The project is planned to be completed in 2 sprints. |

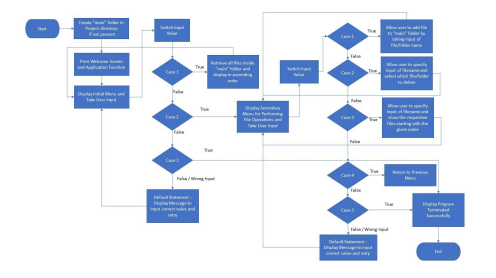
**Tasks completed in this sprint are**:-

|  |
| --- |
| * Creating the flow of the application. * Initializing the git repositories to track changes as development progresses. * Writing the java code to complete the requirements of the project. * Testing the program with different types of user input. * Pushing the code to Git Hub. * Creating this specification document highlighting application capabilities, appearance and user interaction. |

**Core concepts used in the project are:-**

|  |
| --- |
| * **Collections Framework** * **File Handling** * **Sorting** * **Flow control** * **Exception Handling** |

**Flow Chart 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

2 Writing a program in Java for the entry point of the application

(**LockedMeMain.java**)

3 Writing a program in Java to display Menu options available for the use (**MenuOptions.java**)

4 Writing a program in Java to handle Menu options selected by user (**HandleOptions.java**)

5 Writing a program in Java to perform the File operations as specified by user (**FileOperations.java**)

**(FileManager.java)**

6 Pushing the code to Git Hub repository

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

* Open Eclipse
* Go to File -> New -> Project -> Java Project -> Next.
* Type project name and click on “Finish.”
* Right click on project and select new ->package->
* Give name of the package and click on finish
* Right click on package select new->Class, give the name of class and check the checkbox “public static void main(String[] args)”, and click on finish

**Step 2: Write a java program to entry point of the application (LockedMeMain.java)**

**package** com.lockedmeprojectpackage;

**public** **class** LockedMeMain

{

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

{

FileOperations.*createMainFolderIfNotPresent*("main");

MenuOptions.*printWelcomeScreen*("LockedMe", "Swathi Kanduri");

HandleOptions.*handleWelcomeScreenInput*();

}

}

**Step 3: Write a java program to display Menu options available for the user (HandleOptions.java)**

* Right click on the package and select New->class
* Enter class name and click on finish

HandleOptions.java consists of

3.1 Welcome Screen

3.2 Handle File Menu Options

**Step 3.1 Write method to display Welcome Screen**

**public** **static** **void** handleWelcomeScreenInput()

{

**boolean** running = **true**;

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

**do** {

**try** {

MenuOptions.*displayMenu*();

**int** input = sc.nextInt();

**switch** (input) {

**case** 1:

System.***out***.println("Display the name of the files added to the \"main\" folder");

FileOperations.*readFilefromFolder*("main");

**break**;

**case** 2:

System.***out***.println("Enter the name of the file to be added to the \"main\" folder");

FileOperations.*writeContentToNewFileName*("main");

**break**;

**case** 3:

System.***out***.println("Enter the name of the file to be deleted from \"main\" folder");

FileOperations.*deleteFileFromFolder*("main");

**break**;

**case** 4:

System.***out***.println("Enter the name of the file to be searched from \"main\" folder");

FileOperations.*searchFileNameInFolder*("main");

**break**;

**case** 5:

System.*exit*(0);

**break**;

**default**:

System.***out***.println("Please select a valid option from above.");

}

} **catch** (Exception e)

{

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

*handleFileMenuOptions*();

}

} **while** (running == **true**);

//sc.close(); }

**Output:**-

|  |
| --- |
| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*\* Welcome to LockedMe.com.  \*\* This application was developed by Swathi Kanduri.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  You can use this application to :-  • Retrieve all file names in the "main" folder  • Search, add, or delete files in "main" folder.  \*\*Please be careful to ensure the correct filename is provided for searching or deleting files.\*\* |

**Step 3.2 Write method to Handle File Menu Options**

**public** **static** **void** handleFileMenuOptions()

{

**boolean** running = **true**;

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

**do** {

**try** {

MenuOptions.*displayMenu*();

**int** input = sc.nextInt();

**switch** (input) {

**case** 1:

System.***out***.println("Display the name of the files added to the \"main\" folder");

FileOperations.*readFilefromFolder*("main");

**break**;

**case** 2:

System.***out***.println("Enter the name of the file to be added to the \"main\" folder");

FileOperations.*writeContentToNewFileName*("main");

**break**;

**case** 3:

System.***out***.println("Enter the name of the file to be searched from \"main\" folder");

FileOperations.*searchFileNameInFolder*("main");

**break**;

**case** 4:

System.***out***.println("Enter the name of the file to be deleted from \"main\" folder");

FileOperations.*deleteFileFromFolder*("main");

**break**;

**case** 5:

System.*exit*(0);

**break**;

**default**:

System.***out***.println("Please select a valid option from above.");

}

} **catch** (Exception e) {

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

*handleFileMenuOptions*();

}

} **while** (running == **true**);

//sc.close();

}

}

**Output:**-

|  |
| --- |
| @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  COMPANY LOCKERS PVT.LTD  LOCKED ME.COM  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  1. Display all files  2. Add New File  3. Delete a file  4. Search a file  5. Exit  Enter your choice:  1  Display the name of the files added to the "main" folder  file name :: Hello.txt  file name :: Student.txt  file name :: studentname.txt  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  COMPANY LOCKERS PVT.LTD  LOCKED ME.COM  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  1. Display all files  2. Add New File  3. Delete a file  4. Search a file  5. Exit  Enter your choice:  2  Enter the name of the file to be added to the "main" folder  enter file name ::  two.txt  enter content to write in to file ::  xgfxg  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  COMPANY LOCKERS PVT.LTD  LOCKED ME.COM  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  1. Display all files  2. Add New File  3. Delete a file  4. Search a file  5. Exit  Enter your choice:  3  Enter the name of the file to be deleted from "main" folder  Enter the file name to be deleted:  two.txt  File deleted successfully |

**Step 4: Write a java program to perform the File operations as specified by user**

* Right click on the package and select New->class
* Enter class name and click on finish.
* FileOperations consists methods for:-

4.1. Creating “main” folder in project if it’s not already present

4.2. Displaying all files in “main” folder in ascending order and also with directory

structure.

4.3. Creating a file/folder as specified by user input.

4.4. Search files as specified by user input in “main” folder and it’s subfolders.

4.5. Deleting a file/folder from “main” folder

**Step 4.1:-Write method to create “main” folder in project if it’s not present**

**public** **static** **void** createMainFolderIfNotPresent(String folderName) {

File file = **new** File(folderName);

// If file doesn't exist, create the main folder

**if** (!file.exists()) {

file.mkdirs();

}

}

|  |
| --- |
|  |

**Step 4.2:-Write method to display all files in “main” folder in ascending order and also with directory structure**

**public** **static** **void** displayAllFiles(String path)

{

FileOperations.*createMainFolderIfNotPresent*("main");

// All required files and folders inside "main" folder relative to current // folder

System.***out***.println("Displaying all files with directory structure in ascending order\n");

// listFilesInDirectory displays files along with folder structure

List<String> filesListNames = FileOperations.*listFilesInDirectory*(path, 0, **new**

ArrayList<String>());

System.***out***.println("Displaying all files in ascending order\n");

Collections.*sort*(filesListNames);

filesListNames.stream().forEach(System.***out***::println);

}

**public** **static** List<String> listFilesInDirectory(String path, **int** indentationCount,

List<String> fileListNames) {

File dir = **new** File(path);

File[] files = dir.listFiles();

List<File> filesList = Arrays.*asList*(files);

Collections.*sort*(filesList);

**if** (files != **null** && files.length > 0) {

**for** (File file : filesList) {

//System.out.print(" ".repeat(indentationCount \* 2));

**if** (file.isDirectory()) {

System.***out***.println("`-- " + file.getName());

// Recursively indent and display the files

fileListNames.add(file.getName());

*listFilesInDirectory*(file.getAbsolutePath(), indentationCount

+ 1, fileListNames);

} **else** {

System.***out***.println("|-- " + file.getName());

fileListNames.add(file.getName());

}

}

} **else** {

//System.out.print(" ".repeat(indentationCount \* 2));

System.***out***.println("|-- Empty Directory");

}

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

**return** fileListNames;

}

**Output:**-

|  |
| --- |
| @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  COMPANY LOCKERS PVT.LTD  LOCKED ME.COM  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  1. Display all files  2. Add New File  3. Delete a file  4. Search a file  5. Exit  Enter your choice:  1  Display the name of the files added to the "main" folder  file name :: Hello.txt  file name :: Student.txt  file name :: studentname.txt  file name :: two.txt |

**Step 4.3:-Write method to create a file/folder as specified by user input.**

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

{

**try**

{

//to create new file and write into that file

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

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

//for loop to write content into file

**for**(String text:content){

fw.write(text);

}

fw.close();

}

**catch**(Exception Ex)

{

**return** **false**;

}

**return** **true**;

}

**Output:-**

|  |
| --- |
| @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  COMPANY LOCKERS PVT.LTD  LOCKED ME.COM  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  1. Display all files  2. Add New File  3. Delete a file  4. Search a file  5. Exit  Enter your choice:  2  Enter the name of the file to be added to the "main" folder  enter file name ::  two.txt  enter content to write in to file ::  xgfxg |

**Step 4.4:-Write method to search for all files as specified by user input in main folder and its subfolders**

**public** **static** **void** searchFileNameInFolder(String folderpath)

{

String fileName=**null**;

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

System.***out***.println("Enter File name to search ::");

fileName = in.nextLine();

List<String> fileNames=FileManager.*getAllFiles*(folderpath);

fileNames=FileManager.*getAllFiles*(folderpath);

**if**(fileNames.contains(fileName))

System.***out***.println("File exists in folder");

**else**

System.***out***.println("File does not exists in folder");

//in.close();

}

**Output:-**

|  |
| --- |
| @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  COMPANY LOCKERS PVT.LTD  LOCKED ME.COM  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  1. Display all files  2. Add New File  3. Delete a file  4. Search a file  5. Exit  Enter your choice:  4  Enter the name of the file to be searched from "main" folder  Enter File name to search ::  two.txt  File does not exists in folder  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  COMPANY LOCKERS PVT.LTD  LOCKED ME.COM  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  1. Display all files  2. Add New File  3. Delete a file  4. Search a file  5. Exit  Enter your choice:  4  Enter the name of the file to be searched from "main" folder  Enter File name to search ::  Hello.txt  File exists in folder |

**Step 4.5:-** **Writing method to delete file/folder specified by user input in “main” folder.**

**public** **static** **void** deleteFileFromFolder(String folderpath)

{

String fileName=**null**;

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

////scanner object to read input from the user

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

fileName=in.nextLine();

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

**if**(isDeleted)

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

**else**

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

// in.close();

}

}

**Output:-**

To verify if file is deleted on Eclipse, right click on Project and click “Refresh”

|  |
| --- |
| @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  COMPANY LOCKERS PVT.LTD  LOCKED ME.COM  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  1. Display all files  2. Add New File  3. Delete a file  4. Search a file  5. Exit  Enter your choice:  3  Enter the name of the file to be deleted from "main" folder  Enter the file name to be deleted:  two.txt  File deleted successfully |

|  |
| --- |
| @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  COMPANY LOCKERS PVT.LTD  LOCKED ME.COM  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  1. Display all files  2. Add New File  3. Delete a file  4. Search a file  5. Exit  Enter your choice:  5 |

**Pushing the code to Git Hub 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 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.

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.

5. They can specify the starting input, and the program searches all  files / folder starting with the value and displays it. The user is then provided the option to select all files or to select a specific index to delete.

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

7. 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.

8. When the option to retrieve files in ascending order is selected, user is displayed  with two options of viewing the files.

8.1. Ascending order of folders first which have files sorted in them,

8.2. Ascending order of all files and folders inside the “main” folder.

9. 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 “hard coding” 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.