Specification Document for LockedMe.com

Company Name: Lockers pvt Ltd.

Product Name: Lockedme.com

Developer Details:
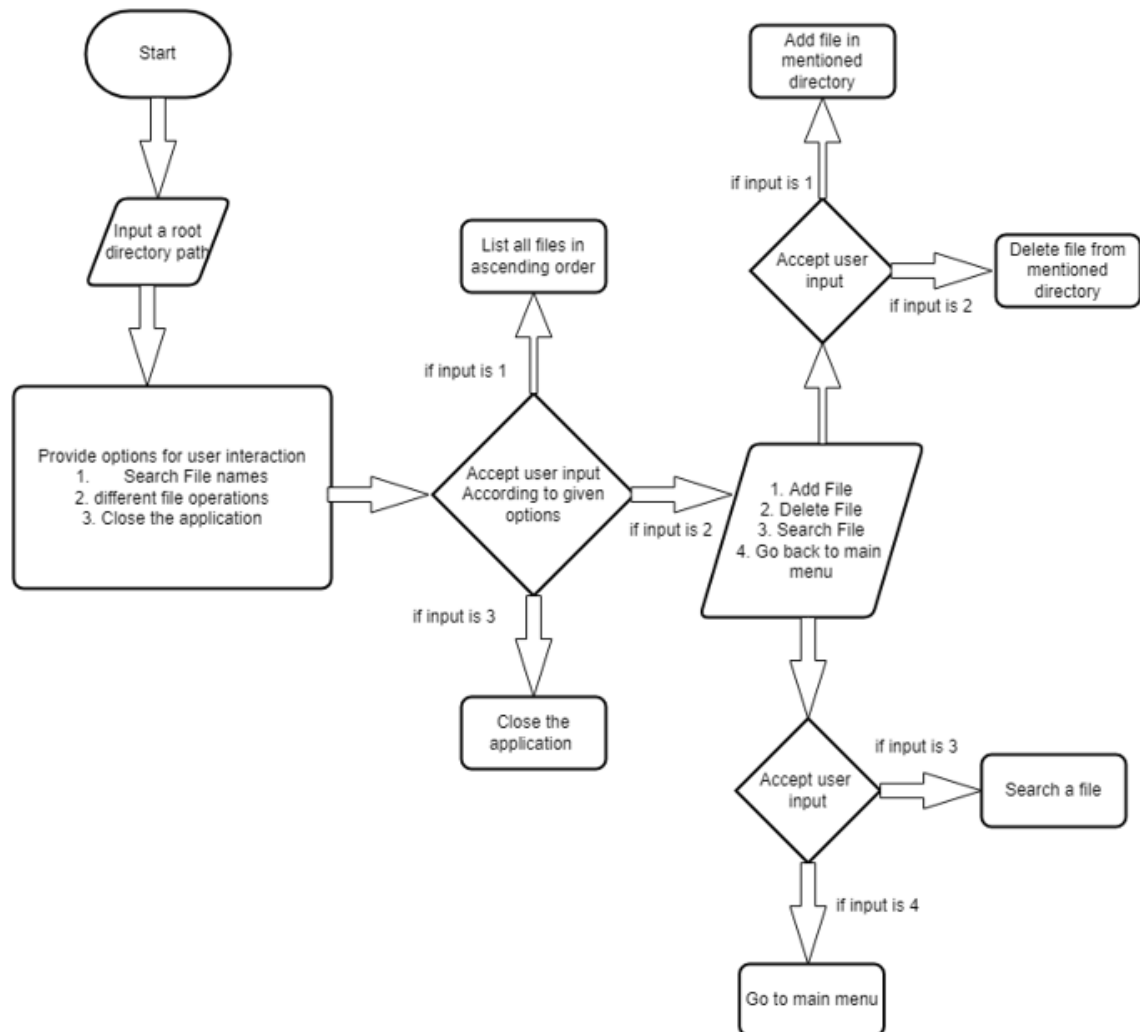
Name	Aditi Patankar
Email	aditibhat7@gmail.com
Contact Number	9769549091

Git Repository Link : https://github.com/Aditi0192/LockedMe.com

Product Capabilities:

- LockedMe.com is a java command line application in which user can make different file operations using a command line interaction.
- All the files are stores User wise under specific directory.
- For example, User1 will have a directory named as User 1 in which respective files are present. In this application these directories are called as Locker.
- User can perform add, delete, search and sort operations on the files present in the locker.
- The aim to develop this application is to make file operations easier for user.
- Also, all the operations are performed through user specific directory in order to avoid confusion on file names.
- File/Directory names are Not case sensitive. Windows operating system does not enforce case sensitivity on File/directory names.

- Following flowchart will explain about the product flow in detail

Sprint Planning

Sprint 1	
User Stories	**Story Points**
1. Understating the requirement	**1**
2. Design the architecture	**3**
3. Identifying the entities and create flowchart	**2**
4. Setup a Git account and project environment	**1**

Sprint 2	
User Stories	**Story Points**
1. Implementation of Menu driven interface	**1**
2. Implementation of logic for listing the files.	**2**
3. Implementation of add file functionality.	**2**
4. Implementation of logic of file name input	**1**
5. Testing the implementation	**2**

Sprint 3	
User Stories	**Story Points**
1. Implementation of delete file functionality	**2**
2. Implementation of search file functionality	**3**
3. Implementation of logic for closing the application	**1**
4. Testing the implementations and overall product	**3**

Technical Specification

Sr.no	Classes/Methods Name	Description
1	Class LockedMeMain	1. This class contains main() . 2. Switch case is used to have a Menu Driven operations. 3. In each case method for specific file operation is called using object of class **FileUserOperation**. 4. object of **FileUserOperation** is created in this class to pass root path value.
2	Class FileUserOperation	1. This class contains all the methods required to perform file operations. 2. **rootPath** is a class variable used for accepting root path of files and directories. 3. Parameterized constructor is created to accept a rootPath 4. This class has methods to perform File Operations. Specification of these methods is listed below
3	addUserFile()	1. This method first accepts directory name. 2. In the directory, user can add specific file 3. addUserFile() makes use of getUserFile(String dirN) to accept file details from user.
4	getUserFile(String dirN)	1. This method accepts parameter directory name and proceeds with creation of File.
5	deleteUserFile()	1. This method first asks for directory name to delete a file 2. If directory is present, delete a user specific file.
6	searchUserFile()	1. This method search a user specific file from all the directories and gives path whare it is stored.
7	getUserFileNames()	1. This function lists all the files in ascending order. Also, it gives all directory names present under a root folder.
8	Java Concepts used in the program	1. Exception handling with try, catch 2. Collections: In the program list is used to store file names. 3. sort() is used to sort file names in ascending order. 4. Switch and inner switch for menu driven application. 5. File operations