

## **LockedMe – Virtual Key for Repositories (Write Up Specification Document)**

### **Version History**

Author	Rabin Kumar Pati
Doc purpose	Specification Document
Date	10-Aug -2021
Version	01

## Table of Contents

This document contains sections for: .....	3
1. Sprints planning and Task completion .....	4
2. Core concepts used in project .....	4
3. Flow Of Application.....	5
4. Demonstrating the product capabilities, appearance, and user interactions....	5
<b>4.1</b> Creating a new project in Eclipse.....	5
<b>4.2</b> Entry Point of Assessment Application .....	5
<b>4.3</b> Display Menu options .....	6
<b>4.4</b> User input process.....	8
<b>4.5</b> User input logical operation .....	9
5. Project package structure. ....	10
6. GitHub Process for Commit to Repository. ....	10

## Project Information

This section intends to provide a high-level picture of the project as well as document details of the project.

### Include the following in this section:

**Project Title:** Virtual Key repository

**Project version history:** Document version detail with editor and date.

**Project Summary:** This application designed with java core concepts includes the functionalities for users to create, view, update, and delete file from folder through command line.

**Project Timeframe:** 2 Weeks

**Prepared By:** Rabin Kumar Pati

**Attached Documentation:** The Projects Screen shots as well as Code Screen shots in pdf format with writeup specification document.

**Project Contacts:** Rabin Kumar Pati

### Project Details:

- Sprint planning and Task completion
- Technologies used in project
- Flow of the Application.
- Demonstrating the user interactions through Command Line.
- Project package structure.
- Pushing the code to GitHub repository
- Unique Selling Points of the Application
- Conclusions

The code for this project is hosted at  
<https://github.com/RABINPATI/LockedMeAssessment>

The project is developed by RABIN KUMAR PATI.

## 1. Sprints planning and Task completion

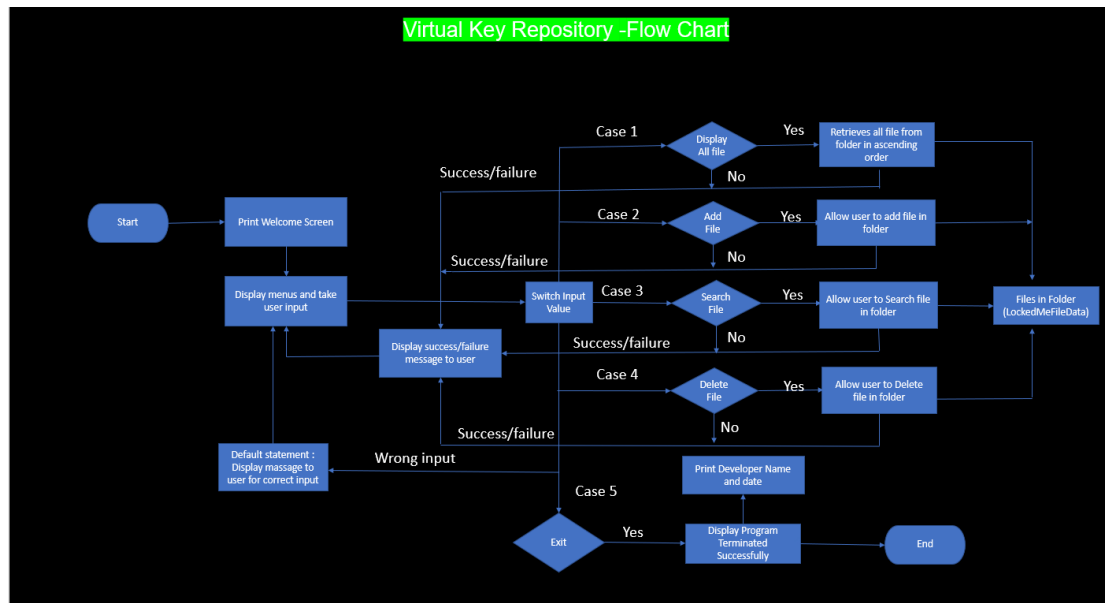
The project is planned to be completed in 3 sprints. Tasks assumed to be completed in the sprint are:

Sprint 1	Gathering the Requirements. Install related Software. Modularized the project. Design the process Flow.
Sprint 2	Development Module 1: Create Menus Module 2: Display all files. Module 3: Add file Testing all 3 modules Git Commit to repository
Sprint 3	Module 4: Search a file Module 5: Delete file Testing all 5 modules Git Commit to repository Create Documentation

## 2. Technologies used in project

Java Concepts	OOPS Concept File Handling Exception Handling Collections framework java 8 API
IDE	Eclipse
Software Tool	GitHub Desktop Snipping Tool
Documentation	MS Office

### 3. Flow Of Application



Virtual Key  
Repository -Flow Chart

#### 4. Demonstrating user interactions

- Creating the project in Eclipse.
- Writing a program in Java for the entry point of the application (**LockedMeMain.java**).
- Writing a program in Java to display Menu options available for the user (**MenuOptions.java**).
- Writing a program in Java to perform the File operations as specified by user (**FileDataOperation.java**).
- Writing a program in Java to perform the File operations logic to perform file operation (**FileDataManager.java**).

## 4.1 Creating a new project in Eclipse

- Open Eclipse
- Go to File -> New -> Project -> Java Project -> Next.
- Type in any project name and click on “Finish.”
- Select your project and go to File -> New -> Class.
- Enter **LockedMeMain** in any class name, check the checkbox “public static void main(String[] args)”, and click on “Finish.”

## 4.2 Entry Point of Assessment Application

```

1 package com.file.data;
2
3 public class LockedMeMain {
4
5     // Folder Path for file to access data
6
7     static final String FOLDER_PATH = "D:\\Project\\FSD\\Phase 1\\LockedMeAssessment\\LockedMeFileD
8
9     public static void main(String[] args) {
10
11         // Display Application name
12
13         MenuOptions.displayWelcomeHeaderNote();
14
15         // Display file processing operations
16
17         MenuOptions.menuProcessOperations(FOLDER_PATH);
18
19     }
20
21 }

```

<https://github.com/RABINPATI/LockedMeAssessment/blob/main/LockedMe/src/com/fileData/LockedMeMain.java>

### 4.3 Display Menu options

**MenuOptions** consists below methods for -:

- **displayWelcomeHeaderNote** for displaying Application name.
- **displayWelcomeFooterNote** for displaying developer name and current date.
- **displayFileMenuOptions** for displaying menus.
- **menuProcessOperatins** for file process operation.

```

1 package com.file.data;
2
3 import java.text.SimpleDateFormat;
4
5
6
7 public class MenuOptions {
8
9     /**
10      * Add the details for application Name in header part.
11      */
12
13     public static void displayWelcomeHeaderNote() {}
14
15
16
17
18     /**
19      * Add the details about developer and date in below part.
20      */
21
22     public static void displayWelcomeFooterNote() {}
23
24
25
26     /**
27      * Display the menu options
28      */
29
30     public static void displayFileMenuOptions() {}
31
32
33     /**
34      *
35      * @param folderPath
36      * File processing operation
37      * Retrieve files
38      * Add files,Search file, Delete file
39      */
40
41     public static void menuProcessOperations(String folderPath) {}
42
43 }

```

output:

```

Application Name: LockedMe.com

***** Select any number from below for file operation and press Enter *****

1) Retrieve all files
2) Add a file
3) Search a file
4) Delete a file
5) Exit program

Enter an option to proceed
5
Program exited successfully.

Developed By RABIN

Date:09-Aug-2021

```

<https://github.com/RABINPATI/LockedMeAssessment/blob/main/LockedMe/src/com/fileData/MenuOptions.java>

#### 4.4 User input process

- It consists below methods for user input process.
  - **createFiles** for process the data to FileManager class for add operation.
  - **displayFileNames** for process the data to FileManager class for retrieving all files in ascending order.
  - **searchFile** for process the data to FileManager class for searching a file from folder with case sensitive, if exists display success message if file not exist display file not found.
  - **deleteFile** for process the data to FileManager class delete a file from folder with case sensitive, if deleted display success message if file not exist display file not found.

```
1 package com.file.data;
2
3 import java.util.ArrayList;
4
5
6
7
8 public class FileDataOperation {
9
10     /**
11      * @param folderPath
12      * Add file
13      */
14
15     public static void createFiles(String folderPath) {}
16
17     /**
18      * @param folderPath
19      * Display Allfiles
20      */
21
22     public static void displayFileNames(String folderPath) {}
23
24     /**
25      * @param folderPath
26      * Search File
27      */
28
29     public static void searchFile(String folderPath) {}
30
31     /**
32      * @param folderPath
33      * Delete File
34      */
35     public static void deleteFile(String folderPath) {}
36
37 }
38
```

<https://github.com/RABINPATI/LockedMeAssessment/blob/main/LockedMe/src/com/fileData/FileDataOperation.java>



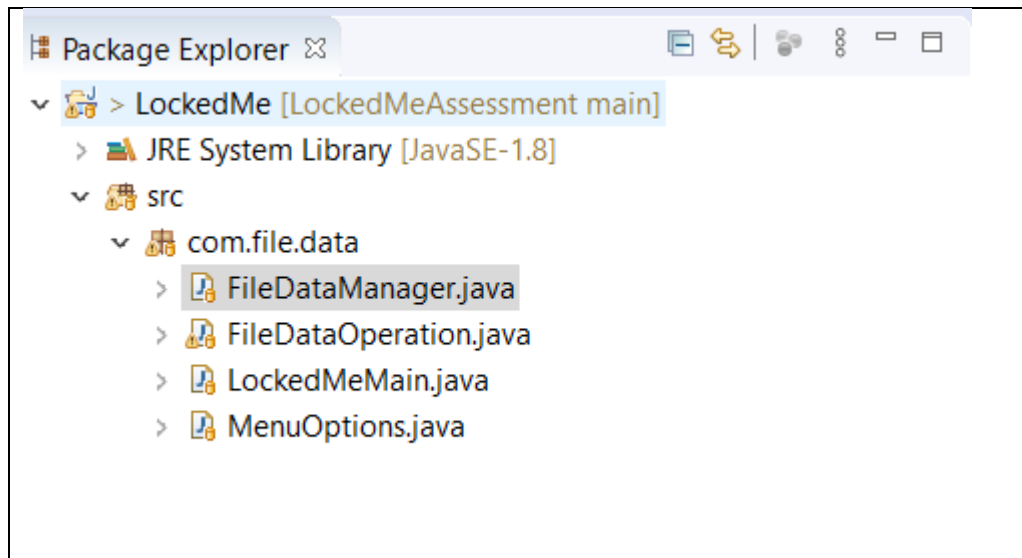
## 4.5 User input logical operation

- It consists below methods for user input process
  - **createFiles** for add into folder.
  - **displayFileNames** for retrieving all files.
  - **searchFile** for searching a file from folder with case sensitive, if exists display success message if file not exist display file not found.
  - **deleteFile** for delete a file from folder with case sensitive, if deleted display success message if file not exist display file not found.

```
1 package com.file.data;
2
3 import java.io.File;
4
5 public class FileDataManager {
6
7     /**
8      *
9      * @param folderPath
10     * @param fileName
11     * @param dataContent
12     * @return file in directory
13     */
14     public static boolean createFiles(String folderPath, String fileName, List<String> dataContent) {
15
16         /**
17          *
18          * @param folderPath
19          * @return all files
20          */
21         public static List<String> getAllFiles(String folderPath) {
22
23             /**
24              *
25              * @param folderPath
26              * @param file_Name
27              * @return true/false based on search results.
28              */
29             public static boolean searchFile(String folderPath, String file_Name) {
30
31                 /**
32                  *
33                  * @param folderPath
34                  * @param file_Name
35                  * @return true/false based on delete results.
36                  */
37                 public static boolean deleteFile(String folderPath, String file_Name) {
38
39                 }
40             }
41         }
42     }
43 }
```

<https://github.com/RABINPATI/LockedMeAssessment/blob/main/LockedMe/src/com/fileData/FileDataManager.java>

## 5. Project package structure.



## 6. GitHub Process for Commit to Repository.

- Create a new repository

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository? [Import a repository.](#)

Owner \* RABINPATI / Repository name \* LockedMeAssessment ✓

Great repository names are Your new repository will be created as -LockedMeAssessment. ilant-bassoo

Description (optional)

☒ Public  
Anyone on the internet can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

Initialize this repository with:  
Skip this step if you're importing an existing repository.

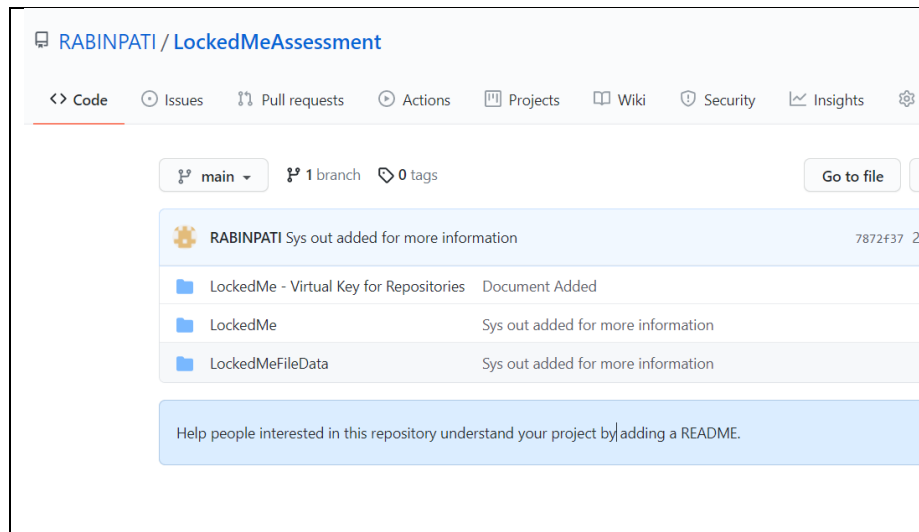
☐ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

☐ Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)

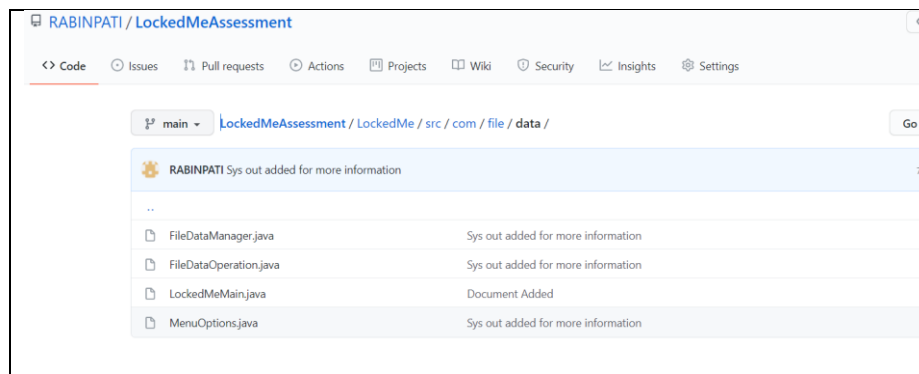
☐ Choose a license  
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

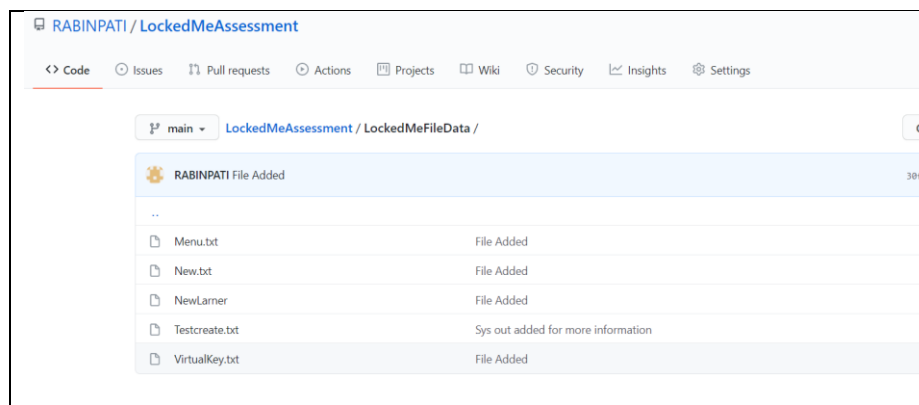
➤ Code and files committed in GitHub Repository



➤ Java Files in repository



➤ Text files in repository



➤ Documents in repository

