**Virtual Key for Repositories**

**GTIHUB LINK**: [Link for GitHub Resources](https://github.com/Vamshi2169/JavaFSD/tree/main/Virtual%20key)

## **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 chart
* 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.

## **concepts used in the project:**

Collections framework, File Handling, Sorting, Flow Control, Recursion, Exception Handling, Streams API

## **Step 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.”

**Step 2:** Writing a program in Java for the entry point of the application (**LockedMeMain.java**)

## **Step 3:** Writing a program in Java to display Menu options available for the user (**MenuOptions.java**)

* Select your project and go to File -> New -> Class.
* Enter **MenuOptions** in class name and click on “Finish.”
* **MenuOptions** consists of methods for -:

1. [Displaying Welcome Screen](bookmark://Step_3_1)

2. [Displaying Initial Menu](bookmark://Step_3_2)

3. [Displaying Secondary Menu for File Operations available](bookmark://Step_3_3)

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

* Select your project and go to File -> New -> Class.
* Enter **Handle Options** in class name and click on “Finish.”
* **Handle Options** consists of methods for -

1.[Handling input selected by user in initial Menu](bookmark://Step_4_1)

2.[Handling input selected by user in secondary Menu for File Operations](bookmark://Step_4_2)

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

* Select your project and go to File -> New -> Class.
* Enter **FileOperations** in class name and click on “Finish.”
* **FileOperations** consists methods for -:

1. [Creating “main” folder in project if it’s not already present](bookmark://Step_5_1)
2. [Displaying all files in “main” folder in ascending order and also with directory structure.](bookmark://Step_5_2)
3. [Creating a file/folder as specified by user input.](bookmark://Step_5_3)
4. [Search files as specified by user input in “main” folder and it’s subfolders.](bookmark://Step_5_4)
5. [Deleting a file/folder from “main” folder](bookmark://Step_5_5)

## **Step 6:** 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, an 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. Users are also provided the option to write content if they want into the newly created file.
4. The application doesn’t restrict users from specifying the exact filename to search/delete file/folder. 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.
5. The application also allows the user to delete folders which are not empty.
6. The user is able to seamlessly switch between options or return to the previous menu even after any required operation like adding, searching, deleting or retrieving of files is performed.
7. When the option to retrieve files in ascending order is selected, the user is displayed with two options of viewing the files.
   1. Ascending order of folders first which have files sorted in them,
   2. Ascending order of all files and folders inside the “main” folder.
8. 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 the 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.

**FLOW CHART:**

