Virtual Key for Repositories

The code for the project is hosted on: GitHub
Project Developer: **Vijaysai Yekbote**

This document contains the following sections:

Sprint Planning and Task Completion

Note: A separate document for sprint planning is attached to the Zip.

The project is planned to be completed in two Sprints. The task assumed to be completed are:

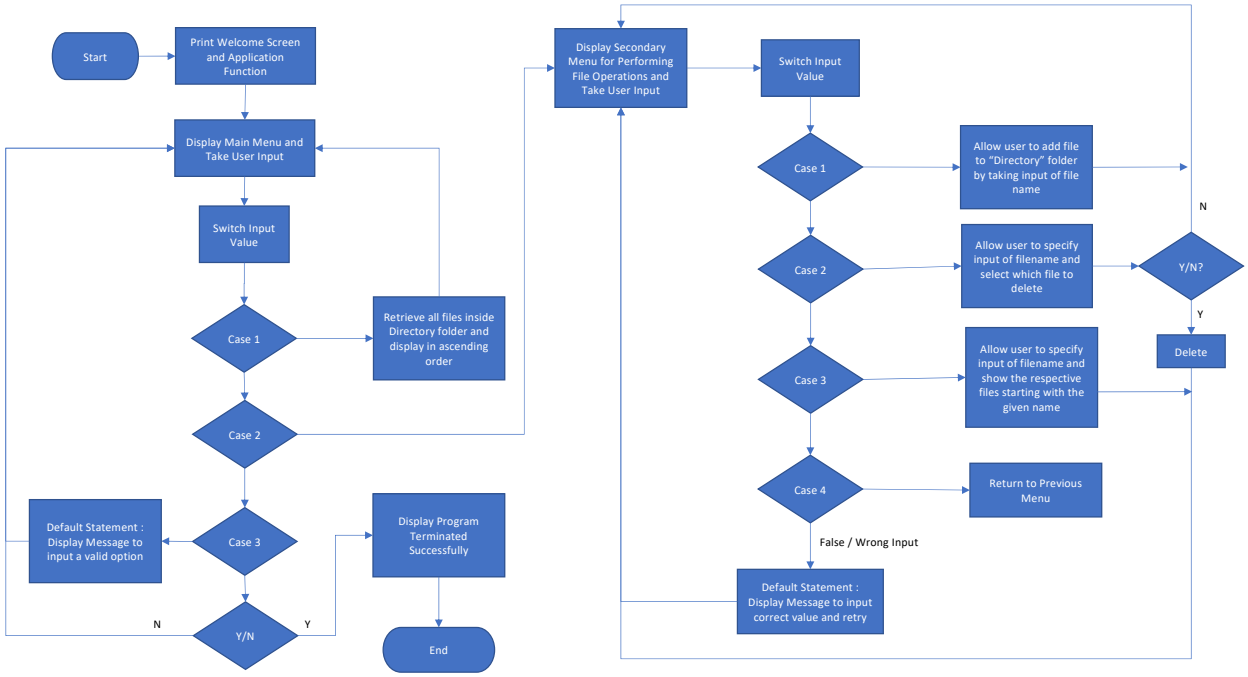
Sprint 1:

- Flow chart of the application
- Initializing Git repository to track changes of development
- Writing the source code **Sprint 2:**
- Testing the program with various user inputs
- Pushing project to GitHub
- Creating documentation

Core Concepts Used in the Project

- Collection Framework
- Flow control
- Recursion
- File Handling
- Exception Handling

Flow Chart of the Application



Product Capabilities

The application has the following capabilities:

1. Show files – shows files from the directory in ascending order (Sorted).
2. Show files menu – shows the following file operations:
   2.1. Add a file – adds files to the directory.
   2.2. Delete a file – deletes files from the directory.
   2.3. Search for a file – searches for file in the directory.
   2.4. Return to main menu -Returns to main menu.
3. Exit the application.

Source code for each of the capabilities with output is shown below: **Note:** Complete source code of the app is zipped please check zip/Gitrepo for reference.

Welcome Screen of the Application:

```
*****************************************************************
* Welcome to the Virtual Key For Your Repositories applicaiton! *
* Developer: Vijaysai Yekbote                                   *
*****************************************************************

Main Menu
1.Show files
2.File Options Menu
3.Exit the application
```

Show files:

```
private void ShowFiles() {
    Directory obj = new Directory(); //Retrieve files from directory
obj.getFiles();
}
public ArrayList<File> ListFiles() { //sort files
File[] directoryFiles = Dfiles.listFiles();
files.clear();
    for(File directoryFile : directoryFiles) {
if(directoryFile.isFile()) {
files.add(directoryFile);
        }
    }
    Collections.sort(files);
return files;
}
public ArrayList<File> getFiles() {
ListFiles();
    if (files.isEmpty()){
        System.out.println("No files exist");}
else {
        System.out.println("Existing files: ");
for (File file : ListFiles()) {
            System.out.println(file.getName());
        }
}
        return files;
    }
```

Output:

```
Main Menu
1.Show files
2.File Options Menu
3.Exit the application

1
Existing files:
akanskha.txt
ascii.txt
iphone.txt
logitech.txt
macbookPro.txt
vijay.txt
```

File Options Menu: (Add, delete, search, return)

```
public class FileOptionsMenu{

    private Directory dir = new Directory();
public void MenuHandler() {
        System.out.println("1. Add a File\n" +
                           "2. Delete a file\n" +
                           "3. Search for a file\n" +
"4. Return to Main Menu");          boolean running =
true;
        Scanner option = new Scanner(System.in);
do {             try {
                int input = option.nextInt();
switch (input) {
                    case 1:  //Adds file to
directory                           this.AddFile();
this.show();                           break;
                    case 2: // Deletes file from
directory                           this.DeleteFile();
this.show();                           break;
                    case 3: // search for file in
directory                           this.SearchFile();
this.show();                           break;
                    case 4: // return to main menu
                        WelcomeScreen obj = new WelcomeScreen();
obj.MainMenu();                           break;
default:
                        System.out.println("Invalid Option");
break;
                }
            } catch (InputMismatchException e) {
                System.out.println(e + ": Please select a valid option");
this.MenuHandler();
            }
}
        while (running = true);

}
 public void show() {
        System.out.println("\nFile Options Menu");
        MenuHandler();
        }
    private String getInputSting() {
        Scanner in = new Scanner(System.in);
return (in.nextLine());
    }
    private void AddFile() {
        System.out.println("Enter file name:");
        String filename = this.getInputSting();
System.out.println("Adding file:" + filename);          try
{
            Path path = FileSystems.getDefault().getPath(Directory.pathName
+ filename).toAbsolutePath();
            File file = new File(dir.getPathName() + filename);
```

```
            if (file.createNewFile()) {
                System.out.println("File added: " + file.getName());
dir.ListFiles().add(file);
            } else {
                System.out.println("File already Exists");
            }
        } catch (IOException e) {
            System.out.println(e);
        }
}
    private void DeleteFile() {
        Directory obj = new Directory();
        obj.getFiles();
        System.out.println("Enter filename to delete:");
        String filename = this.getInputSting();
        System.out.println("Deleting file: " + filename + "\nAre you sure?
(Y/N)");
        String sure = this.getInputSting();
if(sure.equals("y") || sure.equals("Y")) {  //asks sure?
            Path path = FileSystems.getDefault().getPath(Directory.pathName
+ filename).toAbsolutePath();
            File file = path.toFile();
if (file.delete()) {
                System.out.println("Deleted file: " + file.getName());
dir.ListFiles().remove(file);
            } else {
                System.out.println("Failed, file not found");
            }
}
else
{
            return;
        }
}
    private void SearchFile() {
boolean found = false;
        System.out.println("Enter file name:");
        String fileName = this.getInputSting();
        System.out.println("Searching for file: " + fileName);
        ArrayList<File> files = dir.ListFiles();
for (File file : files) {
            if (file.getName().equals(fileName)) {
System.out.println("Found " + fileName);                    found
= true;
            }
}
        if (found == false) {
            System.out.println("File not found");
        }
    }
}
```

Output:

FileMenu output:

```
Main Menu
1.Show files
2.File Options Menu
3.Exit the application

2

File Options Menu
1. Add a File
2. Delete a file
3. Search for a file
4. Return to Main Menu
```

Add file output:

```
File Options Menu
1. Add a File
2. Delete a file
3. Search for a file
4. Return to Main Menu
1
Enter file name:
java.pdf
Adding file:java.pdf
File added: java.pdf
```

Delete a file output:

```
File Options Menu
1. Add a File
2. Delete a file
3. Search for a file
4. Return to Main Menu
2
Existing files:
akanskha.txt
ascii.txt
iphone.txt
java.pdf
logitech.txt
macbookPro.txt
vijay.txt
Enter filename to delete:
java.pdf
Deleting file: java.pdf
Are you sure? (Y/N)
y
Deleted file: java.pdf
```

Search for a file output:

```
File Options Menu
1. Add a File
2. Delete a file
3. Search for a file
4. Return to Main Menu
3
Enter file name:
logitech.txt
Searching for file: logitech.txt
Found logitech.txt
```

Return Main Menu output:

```
File Options Menu
1. Add a File
2. Delete a file
3. Search for a file
4. Return to Main Menu
4

Main Menu
1.Show files
2.File Options Menu
3.Exit the application
```

Exit the Application:

```
System.out.println("Quitting the application....");
System.out.println("Are you sure? Y/N");
Scanner sure = new Scanner(System.in);
String s = sure.nextLine(); if(s.equals("y")
|| s.equals("Y")) {
running = false;
System.exit(0);} else
{
    MainMenu();
}
```

Output:

```
Main Menu
1.Show files
2.File Options Menu
3.Exit the application

3
Quitting the application....
Are you sure? Y/N
y
Applicaiton terminated

Process finished with exit code 0
```

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 type of file name as input.
3. The application provides the list of files while delete. User can enter exact file name from the list to delete.
4. The application seamlessly switches between options and return to previous menu after certain file operation is done(adding/deleting/searching)
5. The application designed is purely modular. If one wants to update the path, they can change it through source code.

Conclusion:

Further improvements to the application can be done which may include:

- Functionality to create folders in directory.
- Add and append data to files.
- Search for file based on Type, last modified.