**VIRTUAL KEY FOR YOUR REPOSITORIES**

**PROJECT OBJECTIVE:**

As a Full Stack Developer, complete the features of the application by planning the development in terms of sprints and then push the source code to the GitHub repository. As this is a prototyped application, the user interaction will be via a command line.

**SPRINTS PLANNING AND TASK ACHIEVED:**

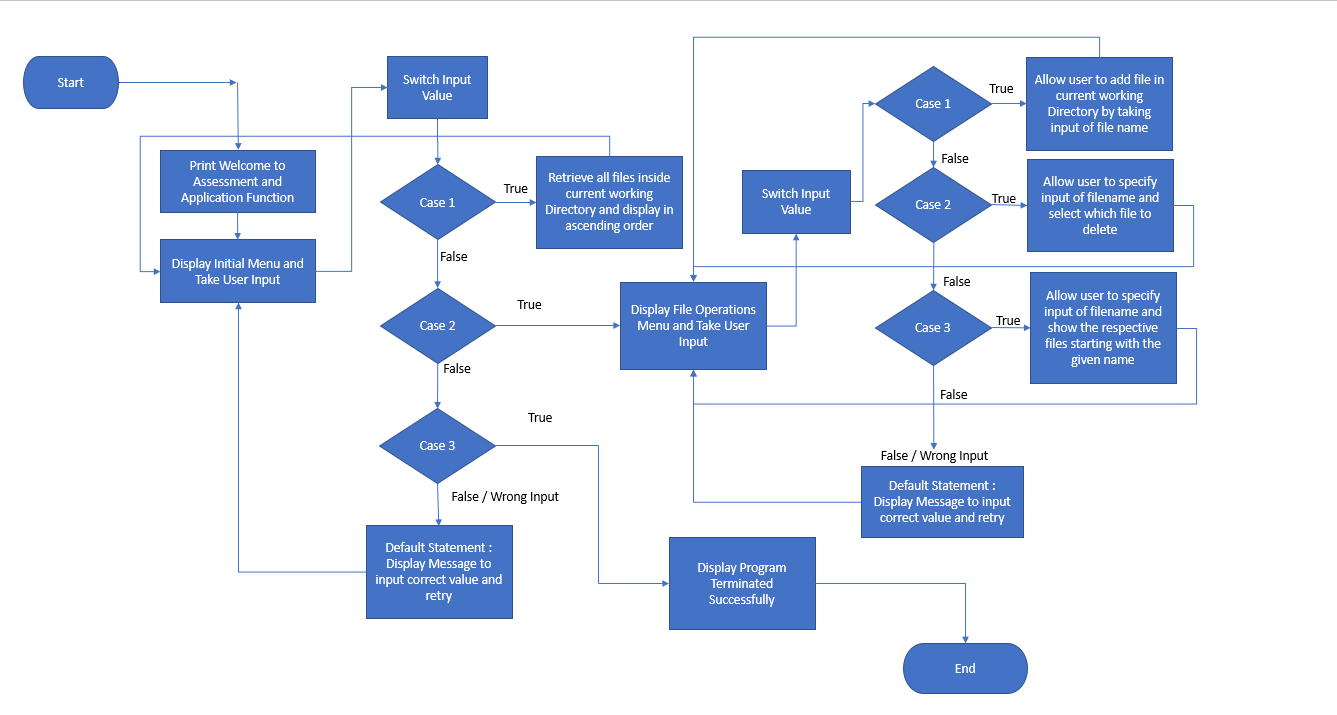
The project is planned to be completed in 1 sprint. The sprints are,

* Creating the flow of the application
* Initializing git repository to track changes as development progresses.
* Writing the Java program to full fill the requirements of the project.
* Testing the program with different kinds of User input.
* Pushing code to GitHub.
* Creating this specification document highlighting application capabilities, appearance, and user interactions.

**CORE CONCEPTS USED IN PROJECT:**

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

**FLOWCHART:**



**ALGORITHM:**

This, steps are to create the main method in class.

**STEP 1:**

* Creating a new project in Eclipse
* Open Eclipse
* Go to File -> New -> Project -> Java Project -> Next.
* I have given the project name as Assessment and click on Finish.
* Select the project and go to File -> New -> Package.
* I have given package name as Assessment1.
* Created the class as Main.
* Enter Main in class name, check the checkbox “public static void main(String[] args)”, and click on Finish.

package.Assessment1:

public class Main {

public static void main(String[] args) throws IOException {

int ch=0, choice=0;

Scanner sc =new Scanner(System.in);

System.out.println("\t Welcome to Assesment! ");

System.out.println(" Name: Vishnu Vardhini \n Learning Platform: Simplilearn \n");

**STEP 2:** Now, created the program to choose any one of the options like list the current files, operations of files and to close the program.

while(true)

{

System.out.println("Choose one of the following options :");

System.out.println("1-> List out the Current Files");

System.out.println("2-> File Operations");

System.out.println("3-> Close the Program");

try{

ch = sc.nextInt();

}

catch(Exception e)

{

System.out.println("Null Exception occurred");

}

**STEP 3:** The main method is created to list the current files.

switch(ch)

{

case 1: //List function to list all the files in ascending order.

FileOperations.listFiles();

break;

**STEP 4:** Here case 2 is for choose any one of the options to perform the operations.

case 2:

System.out.println("Choose one of the following options :");

System.out.println("1-> Add a File");

System.out.println("2-> Delete a File");

System.out.println("3-> Search for a File");

try{

choice = sc.nextInt();

}

catch(Exception e)

{

System.out.println("Null Exception occurred");

}

**STEP 5:** This case is to create a file.

switch(choice)

{

case 1:

System.out.println("Enter the name of a file to be created: ");

String fileCreate = sc.next();

FileOperations.createFile(fileCreate);

break;

**STEP 6:** Deletion of file is created in case 2.

case 2:

System.out.print("Enter the name of a file to be deleted: ");

String fileDelete = sc.next();

FileOperations.deleteFile(fileDelete);

break;

**STEP 7:** This is to search the file in this list, and if the user gives wrong file name it print as invalid name.

case 3:

System.out.println("Enter the name of a file to be searched: ");

String fileSearch = sc.next();

FileOperations.searchFile(fileSearch);

break;

default:

System.out.println("\n Invalid name,Do the Process again!\n");

break;

}

break;

**STEP 8:** This is created to exit the program and prints the output, if the user gives another number, it prints as invalid number and select within the range.

case 3:

sc.close();

System.out.println("\n Thank You..");

System.exit(1);

break;

default:

System.out.println("\n Invalid Number, Select within the range of 1 to 3\n");

break;

}

}

}

}

The following steps are to create the program for file operations and called the function from the main method.

**STEP 1:**

* Created the class as FileOperations.
* Enter FileOperations in class name, check the checkbox “public static void main(String[] args)”, and click on Finish.
* The sorting is done in this program.

package.Assessment1:

public class FileOperations {

protected static String[] sort\_sub(String array[], int size) {

String temp = "";

for(int i=0; i<size; i++){

for(int j=1; j<(size-i); j++){

if(array[j-1].compareToIgnoreCase(array[j])>0){

temp = array[j-1];

array[j-1]=array[j];

array[j]=temp;

}

}

}

return array;

}

**STEP 2:** This code is written to file is created or the file is already exists or not.

protected static void createFile (String fileToBeCreated) {

File file = new File( (System.getProperty("user.dir") ) + "\\" + fileToBeCreated );

try {

if (file.createNewFile() ) {

System.out.println("File Created!");

} else {

System.out.println("File already exists :(");

}

} catch (IOException e) {

e.printStackTrace();

}

}

**STEP 3:** The program is to list the files in ascending order, and sort the added files to it.

protected static void listFiles() {

int fileCount = 0;

ArrayList<String> filenames = new ArrayList<String>();

File directoryPath = new File(System.getProperty("user.dir"));

File[] listOfFiles = directoryPath.listFiles();

fileCount = listOfFiles.length;

System.out.println("Files in ascending order: ");

for (int i = 0; i < fileCount; i++) {

if (listOfFiles[i].isFile()) {

filenames.add(listOfFiles[i].getName());

}

}

String[] str = new String[filenames.size()];

for (int i = 0; i < filenames.size(); i++) {

str[i] = filenames.get(i);

}

String[] sorted\_filenames = sort\_sub(str, str.length);

for(String currentFile: sorted\_filenames) {

System.out.println(currentFile);

}

}

**STEP 4:** The code is called from the function to search the files, if the files are available it prints as file found or it prints not found.

protected static void searchFile(String fileToBeSearched) {

File file = new File( (System.getProperty("user.dir") ) + "\\" + fileToBeSearched );

if(file.exists()) {

System.out.println("File found!");

} else {

System.out.println("Sorry, File Not Found");

} PrintWriter pw;

try {

pw = new PrintWriter(fileToBeSearched);

pw.println("saved");

}

catch (FileNotFoundException e) {

System.out.println(e);

}

}

**STEP 5:** This function is called to delete a file from the list, if the file deleted it prints file deleted successfully or file wasn’t deleted.

protected static void deleteFile(String fileToBeDeleted) {

File file = new File( (System.getProperty("user.dir") ) + "\\" + fileToBeDeleted );

if(file.exists()) {

if ( file.delete() ) {

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

}

}

else {

System.out.println("Sorry, File wasn't deleted");

}

}

}

**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**

* 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.
* 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.
* User is also provided the option to write content if they want into the newly created file.
* The application doesn’t restrict user to specify 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.
* The application also allows user to delete folders which are not empty.
* 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.
* When the option to retrieve files in ascending order is selected, user is displayed with two options of viewing the files.
* Ascending order of folders first which have files sorted in them,
* Ascending order of all files and folders inside the “main” folder.
* 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.

**CONCLUSION:**

The 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.