**FINAL SPECIFICATION DOCUMENT**

**Virtual key for Your Repositories**

**Table of Contents**

|  |  |
| --- | --- |
| **1** | Project and Developer details |
| **2** | Sprints planned and the task achieved |
| **3** | Flowcharts of the application |
| **4** | Core concepts used in the project |
| **5** | Links to the GitHub repository to verify the project completion |
| **6** | Conclusion on enhancing the application and defining the USPs(Unique Selling Point) |

The code for this project is hosted at………………………………...

The project is developed by Yobiga D

**Sprints Planning and Task completion**

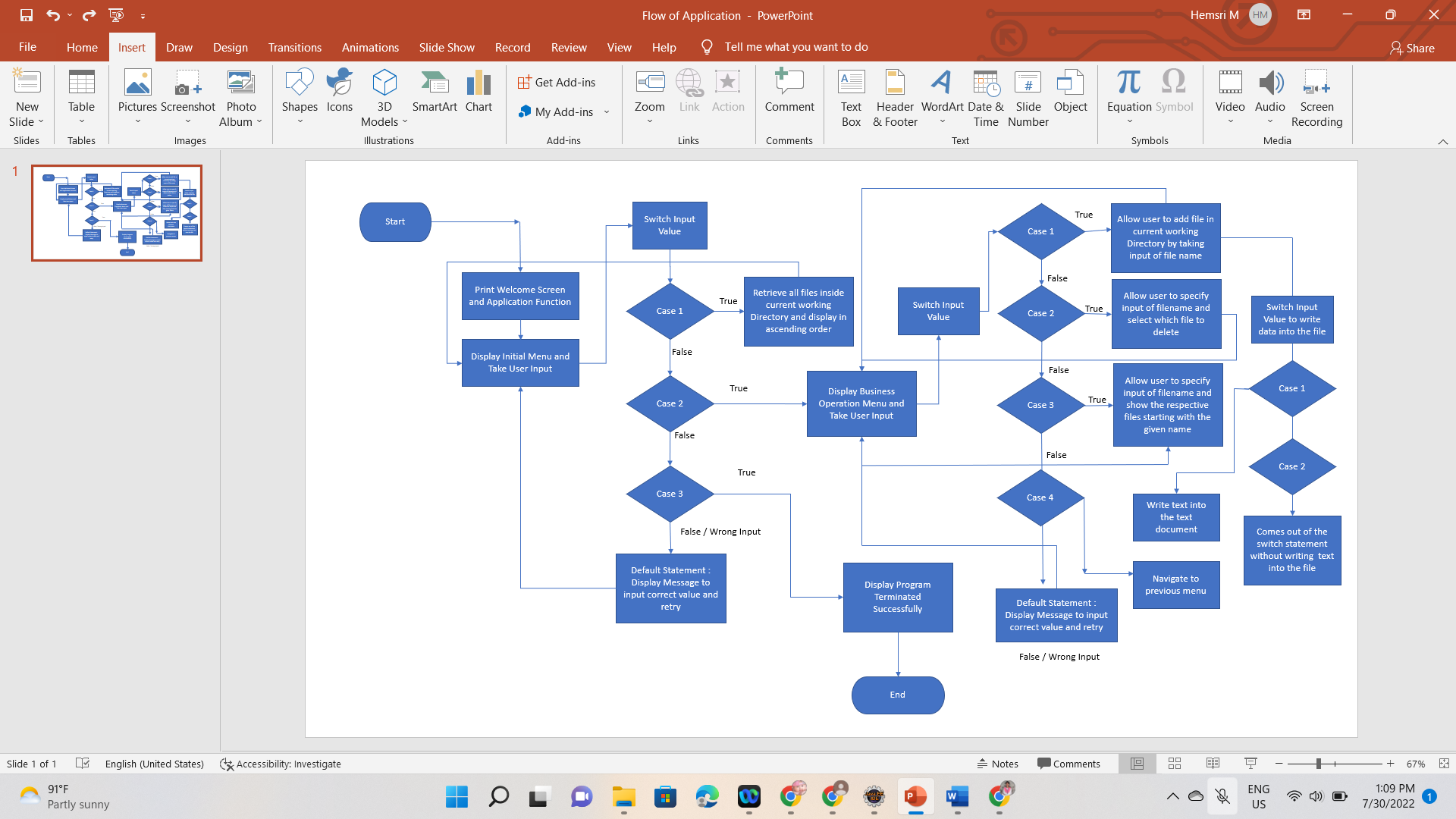
The project is planned to be completed in 1 sprint. Tasks assumed to be completed in the sprint are:

* Creating the flow of the application
* Initializing git repository to track changes as development progresses.
* Writing the Java program to fulfil 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.

**Core Concepts used in the Project:**

* Collection framework
* File handling
* Sorting
* Flow Control
* Recursion
* Exception Handling

**Flow of the Application:**



**Demonstrating the product capabilities, appearance, and user interactions:**

To demonstrate the product capabilities, below are the sub-sections configured to highlight appearance and user interactions for the project:

* Create the project in Eclipse.
* Write a program in java to retrieve the files located in the file path.
* Write a program in java to handle Business Operation.
* Pushing the code to GitHub.

**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 Virutualmain in any class name, check the checkbox “public static void main(String[] args)”, and click on “Finish.”

Source Code consists of:

**package** virtualkey;

**import** java.util.Scanner;

**import** java.io.IOException;

**public** **class** lockerkey {

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

**int** ch=0, choice=0;

Scanner sc =**new** Scanner(System.***in***);

System.***out***.println("\t VIRTUAL KEY");

**while**(**true**)

{

System.***out***.println("Please choose one of the following options :");

System.***out***.println("1. List Current Files");

System.***out***.println("2. Business Operations");

System.***out***.println("3. Close Application");

**try**{

ch = sc.nextInt();

}

**catch**(Exception e)

{

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

}

**switch**(ch)

{

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

businessoperation.*listFiles*();

**break**;

**case** 2:

System.***out***.println("Please choose one of the following options :");

System.***out***.println("1. To Add a File");

System.***out***.println("2. To Delete a File");

System.***out***.println("3. To Search for a File");

**try**{

choice = sc.nextInt();

}

**catch**(Exception e)

{

System.***out***.println("exception occured");

}

**switch**(choice)

{

**case** 1:

//Creation of a file takes place

System.***out***.println("Enter the Name of a file to be Created: ");

String fileCreate = sc.next();

// Calling the function to create the file

businessoperation.*createFile*(fileCreate);

**break**;

**case** 2:

//deletion of a file takes place

System.***out***.print("Enter the Name of a File to be Deleted: ");

String fileDelete = sc.next();

// Calling the function to delete the file

businessoperation.*deleteFile*(fileDelete);

**break**;

**case** 3:

//Search for a file takes place

System.***out***.println("Enter the Name of a File to be Searched: ");

String fileSearch = sc.next();

// Calling the function to search the file

businessoperation.*searchFile*(fileSearch);

**break**;

**default**:

//In the case of unprecedented input execute this

System.***out***.println("Invalid process!!");

**break**;

}

**break**;

**case** 3:

//Voluntarily exiting the application

sc.close();

System.***out***.println("\n It was nice having you here!!");

System.*exit*(1);

**break**;

**default**:

//In the case of unprecedented input execute this

System.***out***.println(" !!Invalid Input!!");

**break**;

}

}

}

}

**STEP 2 :**[**Writing a program in Java to display the Current file Option available for the user (DisplayFiles.java)**](#Step_3)

* Select your project and go to File -> New -> Class.
* Enter businessoperation in class name and click on “Finish.”
* List all the files in ascending order
* Files to be searched
* File to be added and write the content in the file

**package** virtualkey;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.ArrayList;

**public** **class** businessoperation {

//Bubble sort to sort file in ascending order

**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;

}

//File listing function

**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);

}

}

//File delete function

**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 -(File Not Found)");

}

}

//File search function

**protected** **static** **void** searchFile(String fileToBeSearched) {

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

//Check whether file whether file exists or not.

//If yes then display associated message

**if**(file.exists()) {

System.***out***.println("File Has Been Found");

} **else** {

System.***out***.println("Sorry,File is not here-(File Not Found)");

} PrintWriter pw;

**try** {

pw = **new** PrintWriter(fileToBeSearched); //may throw exception

pw.println("saved");

}

// providing the checked exception handler

**catch** (FileNotFoundException e) {

System.***out***.println(e);

}

}

//File creation function

**protected** **static** **void** createFile (String fileToBeCreated) {

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

**try** {

**if** (file.createNewFile() ) {

System.***out***.println("File Has Beeen Created!");

} **else** {

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

}

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

**Step 3: Pushing the code to GitHub.**

* 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

GitHub link: <https://github.com/yobiga-dhanasekaran/Assisted-projects/tree/main/virtualkey>

**UNIQUE SELLING POINTS OF THE APPLICATION:**

* The application is made to continue functioning and accepting user input even in the face of errors. The proper option must be chosen in order to terminate the application.
* The application can take any file/folder name as input. The user can specify a relative path and the application will create the necessary folder structure even if the user wants to create nested folder structures.
* Additionally, the user is given the choice to add content to the newly generated file if they so choose.
* 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:**

* 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 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.