**LockedMe.com Documentation**

This document contains sections for:

* Sprint Planning and Task Completion.
* Core concepts used in the project.
* Flow of the Application
* Demonstrate the product capabilities, appearance and user interactions.
* Unique Selling Points of the Application

The code for this project is hosted at https://github.com/KulneetKaur/Project1 .

The project is developed by Kulneet Kaur.

**Sprints Planning and Task Completion**

The project is planned to be completed in 2 sprints each of 2 weeks duration. Tasks assumed to be completed in 1st sprint are:

* Creating the flow of the application
* 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

Tasks assumed to be completed in 2nd sprint are:

* Pushing code to GitHub.
* Creating this specification document highlighting application capabilities, appearance, and user interactions.

The task was completed within the above timeframe of both the sprints successfully.

**Flow of the application**

## **Diagram Description automatically generated****Demonstrating the product capabilities, appearance, and user interactions**

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

1. Creating the project in Eclipse
2. Writing a program in Java for the entry point in applications. (Application.java)
3. Writing a program in Java to perform file operations specified by the user (fileOperations.java)
4. Pushing the code to GitHub repository.

## **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 **Application** 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 in applications. (Application.java)

**import** java.util.Scanner;

**public** **class** Application {

**public** **static** **void** menuOptionsDisplay() {

System.***out***.println("Select any of the following");

System.***out***.println("1: To display all files in directory \n"

+ "2. To create a file in directory \n"

+ "3. To delete a file in directory \n"

+ "4. To search a file in directory \n"

+ "5. To exit the application. \n");

}

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

System.***out***.println("Application Name: LockedMe.com");

System.***out***.println("Developer Name: Kulneet Kaur");

**boolean** running = **true**;

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

**do** {

Application.*menuOptionsDisplay*();

**int** a=sc.nextInt();

**switch** (a) {

**case** 1:

System.***out***.println("Displaying all files with directory structure in ascending order\n");

fileOperations.*display*();

**break**;

**case** 2:

System.***out***.print("Enter file name to be created: ");

Scanner sc1= **new** Scanner(System.***in***);

String filename = sc1.nextLine();

fileOperations.*createFile*(filename);

**break**;

**case** 3:

Scanner sc2 = **new** Scanner(System.***in***);

System.***out***.print("Enter file to be deleted: ");

String filename1 = sc2.nextLine();

fileOperations.*deleteFile*(filename1);

**break**;

**case** 4:

System.***out***.print("Enter file to be searched for: ");

Scanner sc3 = **new** Scanner(System.***in***);

String filename11 = sc3.nextLine();

fileOperations.*searchFile*(filename11);

**break**;

**case** 5:

System.***out***.println("Application exited successfully.");

running = **false**;

sc.close();

System.*exit*(0);

**break**;

**default**:

System.***out***.println("Please enter a valid option.");

}

}**while**(running==**true**);

}

}

**Step 3:** Writing a program in Java to perform file operations specified by the user (fileOperations.java)

**import** java.io.File;

**import** java.util.Arrays;

**public** **class** fileOperations {

**public** **static** **void** display() {

String path1 = System.*getProperty*("user.dir");

String[] dirListing = **null**;

File dir = **new** File(path1);

dirListing = dir.list();

Arrays.*sort*(dirListing);

System.***out***.println(Arrays.*deepToString*(dirListing));

}

**public** **static** **void** createFile(String filename) {

File file = **new** File(filename);

**try** {

**boolean** value = file.createNewFile();

**if** (value) {

System.***out***.println("The new file is created.");

}

**else** {

System.***out***.println("The file already exists.");

}

}

**catch**(Exception e) {

e.getStackTrace();

}

}

**public** **static** **void** deleteFile(String fileName) {

**try**

{

File f = **new** File(fileName);

**if**(f.delete())

{

System.***out***.println(f.getName() + " deleted");

}

**else**

{

System.***out***.println("File not found");

}

}

**catch**(Exception e)

{

e.printStackTrace();

}

}

**public** **static** **void** searchFile(String fileName) {

File directory = **new** File("").getAbsoluteFile();

String[] flist = directory.list();

**int** flag = 0;

**if** (flist == **null**) {

System.***out***.println("Empty directory.");

}

**else** {

// Linear search in the array

**for** (**int** i = 0; i < flist.length; i++) {

String filename = flist[i];

**if** (filename.equalsIgnoreCase(fileName)) {

System.***out***.println(filename + " found");

flag = 1;

}

}

}

**if** (flag == 0) {

System.***out***.println("File Not Found");

}

}

}

**Step 4:** 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, appropriate option needs to be selected.
2. The application can take any file name as input.
3. 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.