Lockedme.com – virtual key for your repositories

This document contains the following

Project and developer details

Sprint planning and tasks completion

Algorithms and flowchart of the application

Core concepts used in project

Links to the Github repository

Demonstrating of product capabilities, appearance and user interactions

Unique selling points of the application

Conclusion

Project and developer details:

Project objective:

The project objective of LockedMe.com is to create a simple file management system that allows users to retrieve, add, delete, and search files in a directory. The system should be user-friendly and have a clear, intuitive interface

that makes it easy for users to navigate and perform operations. The program aims to provide a hassle-free experience for users who want to manage their files without having to deal with complex software. Additionally, the program should be able to sort file names in ascending order for easy identification and organization. Overall, the objective of the project is to provide a convenient and efficient solution for file management needs.

Background of the problem statement:

Company Lockers Pvt. Ltd. hired you as a Full Stack Developer. They aim to digitize their products and chose LockedMe.com as their first project to start with. You're asked to develop a prototype of the application. The prototype of the application will be then presented to the relevant stakeholders for the budget approval. Your manager has set up a meeting where you're asked to present the following in the next 15 working days (3 weeks):

Specification document - Product's capabilities, appearance, and user interactions

Number and duration of sprints required

Setting up Git and GitHub account to store and track your enhancements of the prototype

Java concepts being used in the project

Data Structures where sorting and searching techniques are used.

Generic features and three operations:

Retrieving the file names in an ascending order

Business-level operations:

Option to add a user specified file to the application

Option to delete a user specified file from the application

Option to search a user specified file from the application

Navigation option to close the current execution context and return to the main context

Option to close the application

The goal of the company is to deliver a high-end quality product as early as possible.

The flow and features of the application:

Plan more than two sprints to complete the application

Document the flow of the application and prepare a flow chart

List the core concepts and algorithms being used to complete this application

Code to display the welcome screen. It should display:

Application name and the developer details

The details of the user interface such as options displaying the user interaction information

Features to accept the user input to select one of the options listed

The first option should return the current file names in ascending order. The root directory can be either empty or contain few files or folders in it

The second option should return the details of the user interface such as options displaying the following:

Add a file to the existing directory list

You can ignore the case sensitivity of the file names

Delete a user specified file from the existing directory list

You can add the case sensitivity on the file name in order to ensure that the right file is deleted from the directory list

Return a message if FNF (File not found)

Search a user specified file from the main directory

You can add the case sensitivity on the file name to retrieve the correct file

Display the result upon successful operation

Display the result upon unsuccessful operation

Option to navigate back to the main context

There should be a third option to close the application

Implement the appropriate concepts such as exceptions, collections, and sorting techniques for source code optimization and increased performance

You must use the following:

Eclipse/IntelliJ: An IDE to code for the application

Java: A programming language to develop the prototype

Git: To connect and push files from the local system to GitHub

GitHub: To store the application code and track its versions

Scrum: An efficient agile framework to deliver the product incrementally

Search and Sort techniques: Data structures used for the project

Specification document: Any open-source document or Google Docs

Following requirements should be met:

The source code should be pushed to your GitHub repository. You need to document the steps and write the algorithms in it.

The submission of your GitHub repository link is mandatory. In order to track your task, you need to share the link of the repository. You can add a section in your document.

Document the step-by-step process starting from sprint planning to the product release.

Application should not close, exit, or throw an exception if the user specifies an invalid input.

You need to submit the final specification document which includes:

Project and developer details

Sprints planned and the tasks achieved in them

Algorithms and flowcharts of the application

Core concepts used in the project

Links to the GitHub repository to verify the project completion

Your conclusion on enhancing the application and defining the USPs (Unique Selling Points)

The given code is a Java program for a file management system called LockedMe.com. This application has two levels of operations:

Retrieving file names in ascending order.

Business-level operations such as adding, deleting, and searching files.

The program starts by importing the Scanner class from the java.util package. It then displays a welcome message, followed by the main menu.

The main menu presents the user with three options:

Retrieve file names in ascending order.

Business-level operations.

Exit application.

The program then prompts the user to enter their choice. If the user chooses option 1, the program creates an instance of the buisnessLogic class and calls the sortFileNames() method to retrieve the file names in ascending order.

If the user chooses option 2, the program enters the business-level operations menu. The user is presented with four options:

Add a file to the existing directory list.

Delete a file from the existing directory list.

Search a file from the existing directory list.

Return to main menu.

Exit application.

The program prompts the user to enter their choice, and based on their input, the program creates an instance of the buisnessLogic class and calls the appropriate method to perform the requested operation. If the user chooses option 4, the program returns to the main menu. If the user chooses option 5, the program exits.

If the user chooses option 3 from the main menu, the program displays a farewell message and exits.

As a Full Stack Developer for LockedMe.com, I understand that the company aims to deliver a high-quality product as early as possible. In order to achieve this, I propose the following plan to develop the prototype of the application:

Specification Document:

Within the first 5 working days, I will create a detailed specification document that will cover the product's capabilities, appearance, and user interactions. This document will outline the features of the application and its functionalities to be implemented.

Sprints:

I will plan for 3 sprints, each with a duration of 5 working days each. This will help to break down the development process into manageable parts, allowing for more efficient development and easier tracking of progress.

Git and GitHub Account:

I will create a Git and GitHub account for version control and to store and track enhancements to the

prototype. This will help to ensure that all changes are tracked and can be easily reverted if necessary.

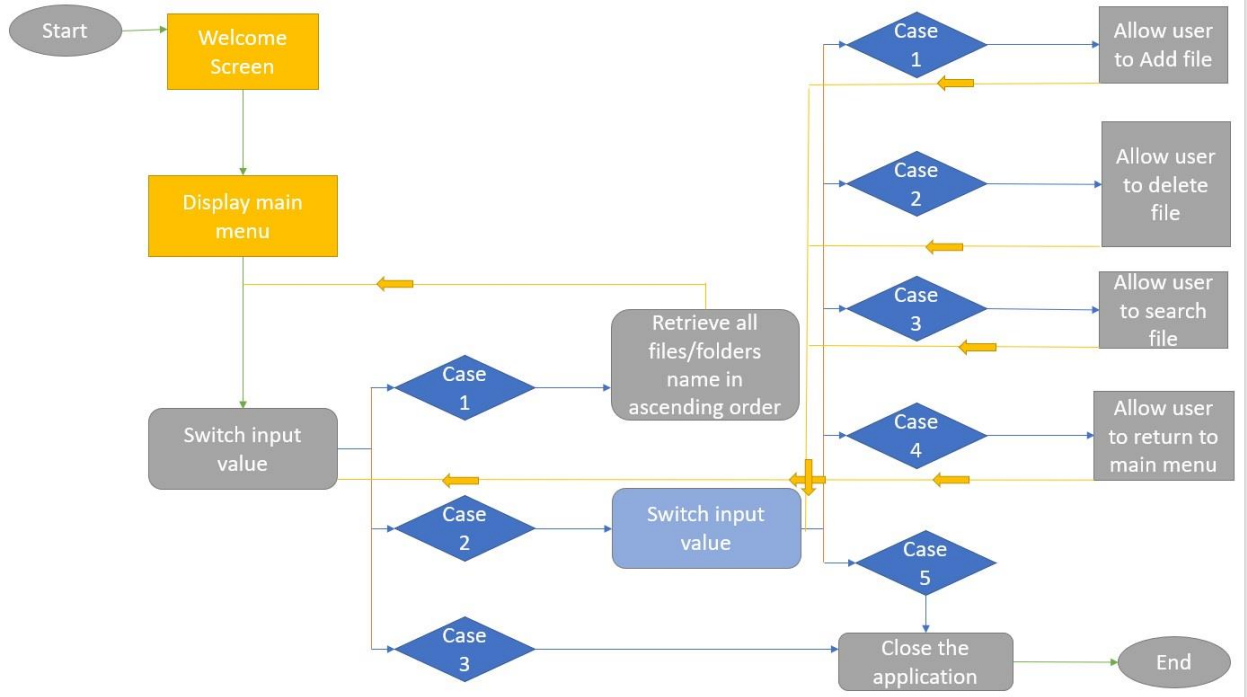
Java Concepts:

The prototype will be developed using core Java concepts such as classes, objects, interfaces, exception handling, collections, and file handling. These concepts will be utilized to develop the core functionalities of the application.

Data Structures:

Data structures such as arrays, linked lists, and binary search trees will be used for sorting and searching techniques. These data structures will be implemented to improve the performance and efficiency of the application.

Flow Chart:



I will create a flow chart to document the flow of the application. This will help to ensure that all features are implemented correctly and that the application flows logically for the end-user.

Core Concepts and Algorithms:

The core concepts and algorithms being used for this application include file handling, exception handling, sorting, and searching techniques.

Welcome Screen:

The welcome screen will display the name of the application and the developer details. It will also display the user interface options for the user to select from.

Option 1: Retrieve File Names in Ascending Order:

This option will display the current file names in ascending order. If the root directory is empty, it will return an appropriate message to indicate this.

Option 2: Business-Level Operations:

This option will have three sub-options:

Add a user specified file to the application

Delete a user specified file from the application

Search a user specified file from the application

The add and delete options will ignore and consider the case sensitivity of file names, respectively. The

search option will allow the user to retrieve a file based on their specific query.

Upon successful completion, the result of the operation will be displayed. If unsuccessful, an appropriate error message will be displayed.

Option 3: Navigation and Closing:

This option will allow the user to navigate back to the main context or close the application entirely.

Implementing appropriate concepts:

I will utilize appropriate concepts such as exceptions, collections, and sorting techniques to optimize the source code and increase the performance of the application.

In conclusion, I believe that by following the above plan, I will be able to deliver a high-quality prototype of the LockedMe.com application within the given timeline of 15 working days.

Business logic class code:

```
package simplilearnProject;
import java.io.*;
```

```
import java.util.*;


public class buisnessLogic {
Scanner sc = new Scanner(System.in);
protected void addFile() {
	System.out.println("Enter file name to add (with path):
");
	String fileName=sc.nextLine();
	File newFile = new File(fileName);
	try {
		if(newFile.createNewFile()) {
		FileOutputStream fos = new
FileOutputStream(fileName);
		System.out.println("Enter the file content: ");
		String content = sc.nextLine();
		byte[] b = content.getBytes();
		fos.write(b);
		fos.close();
		System.out.println("File is saved on given
location");
```

```
            }
            else {
                System.out.println("file '"+fileName+"' already exists.");
        }
        }
        catch(Exception e) {
            System.out.println("Failed to create "+fileName+" file.");
            e.printStackTrace();
        }
}
protected void deleteFile() {
        System.out.println("Enter the file name to delete (with path): ");
        String fileName = sc.nextLine();
        try {
            File file = new File(fileName);
            if(file.isDirectory()) {
                System.out.println(file.getName()+" is a directory");
```

```
            }
            else if(file.delete())
                        System.out.println("File  '"  +fileName+
"' deleted successfully.");
                    else
                        System.out.println("Failed  to  delete  '"
+fileName + "' file.");

        }catch(Exception e) {
            e.printStackTrace();
        }
}
protected void searchFile() {
        System.out.println("Enter the directory path: ");
        String path = sc.nextLine();
        System.out.println("Enter the file name to search: ");
        String fileName = sc.nextLine();
        File file = new File(path);
        String[] list = file.list();
        boolean flag = false;
```

```
        for(int i=0;i<list.length;i++) {
            if(fileName.equals(list[i]))
                flag = true;
        }
        if(flag)
            System.out.println("file found");
        else
            System.out.println("file not found");
}
protected void sortFileNames() {
        System.out.println("Enter a path");
        String path = sc.nextLine();
        File file = new File(path);
        File[] files = file.listFiles();
        Arrays.sort(files);

        if(files.length>0) {
            System.out.println("files are: ");
            for(File f : files) {
                System.out.println(f.getName());
```

```
            }
        }
        else
            System.out.println("The root directory is empty");
}
}
```

Executable logic class

```
package simplilearnProject;
import java.util.Scanner;
public class executableLogic {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

    System.out.println("==========================
==============");
        System.out.println("");
        System.out.println("~ Welcome to
LockedMe.com ~");
```

```
            System.out.println("");
            System.out.println("");
            System.out.println("");
            System.out.println("");
            System.out.println("Developed by: Om Shankar Mishra");
            System.out.println("");
            System.out.println("Email Id: omshankar92229@gmail.com");
            System.out.println("");

        System.out.println("==========================================");
            System.out.println("");
            int ch,sbch;
            do {

        System.out.println("**********************************************");
                System.out.println("Main Menu");
                System.out.println();
```

```
                System.out.println("1. Retrieve file names in
ascending order");
                System.out.println("2.          Buisness-level
operations (add, delete & search)");
                System.out.println("3. Exit application");

        System.out.println("**********************************
**********");
                System.out.println();
                System.out.println("Enter your choice: ");

                ch = sc.nextInt();

                switch(ch) {
                case 1:
                    buisnessLogic retfileobj = new
buisnessLogic();
                    retfileobj.sortFileNames();
                break;
                case 2:
                    do {
```

```
        System.out.println("==========================
==============");
                        System.out.println("Buisness-level
operations");
                        System.out.println();
                        System.out.println("1. Add a file to the
existing directory list");
                        System.out.println("2. Delete a file from
the existing directory list");
                        System.out.println("3. Search a file from
the existing directory list");
                        System.out.println("4. Return to main
menu");
                       System.out.println("5. Exit application");

        System.out.println("==========================
==================");
                        System.out.println();
                        System.out.println("ENTER YOUR
CHOICE: ");
```

```
                    sbch = sc.nextInt();
                    switch(sbch){
                    case 1:
                        buisnessLogic addfileobj = new
buisnessLogic();
                        addfileobj.addFile();
                        break;
                    case 2:
                        buisnessLogic delfileobj = new
buisnessLogic();
                        delfileobj.deleteFile();
                        break;
                    case 3:
                        buisnessLogic serfileobj = new
buisnessLogic();
                        serfileobj.searchFile();
                        break;
                    case 4:
                        System.out.println("Return to main
menu");
                        break;
```

```
                    case 5:
                        System.out.println("--Exiting
program--");
                        System.out.println("Thank you for
using LockedMe!");
                        System.exit(0);
                        default:
                            System.out.println("please
enter valid choice: ");
                            break;
                    }
                    }
                    while(sbch!=4);
                    break;
                case 3:
                    System.out.println("--Exiting program--
");
                    System.out.println("Thank you for using
LockedMe!");
                    System.exit(0);
                    break;
```

```
                default:
                        System.out.println("--Invalid
choice--");
                        System.out.println("Please  enter  a
valid choice");
                        break;
                }

        }
        while(ch!=3);
    }
}
```