**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 the project
* Links to the GitHub repository
* Demonstration of product capabilities, appearance, and user interactions
* Unique Selling Points of the application
* Conclusion

**Project and Developer details:**

**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.

**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 application prototype will then be presented to the relevant stakeholders for 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 the 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

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

**Developer Details:**

Aniket Fale

[Ani.rait1075@gmail.com](mailto:priyapillarisetty19@gmail.com)

Spring planning and Task completion:

This project is intended to be delivered in three sprints.

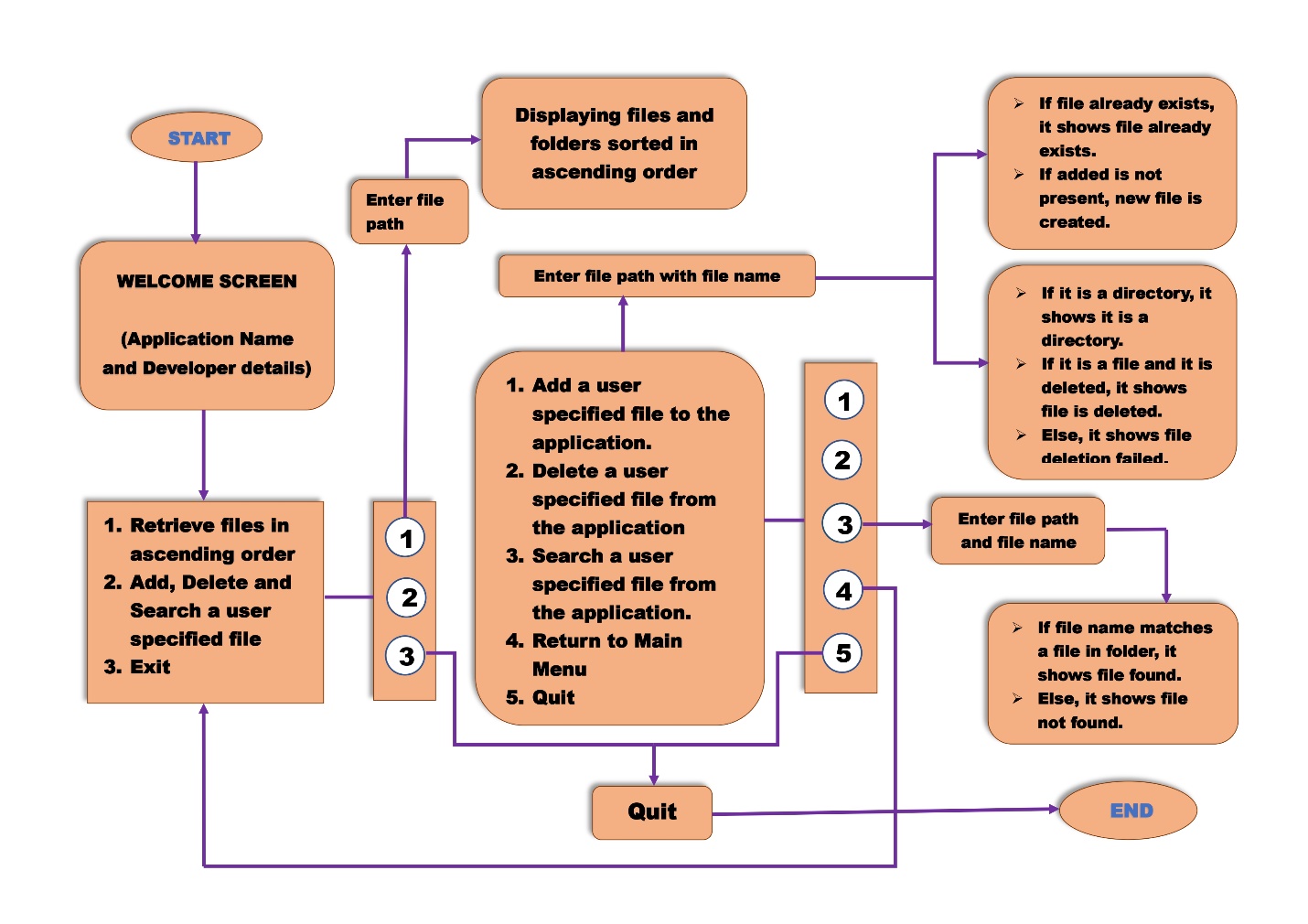
Sprint 1: Analysed the application’s features and prepared a flow chart and Git Repository.

Sprint 2: Wrote different sub-methods required for the application's main method followed b by the welcome screen and menu. Resolved exceptions and errors.

Sprint 3: Tested the application numerous times to ensure a high-end quality product and pushed it to GitHub. Prepared this document highlighting the application’s capabilities, appease appearance, and user interactions.

Algorithms and Flowchart of the application:

Flowchart:





Algorithms:

1. Retrieving files in ascending order:
   1. START
   2. Prompt path, file and assign all the files in the folder to the files list.
   3. Sort the list of files.
   4. If(files.length)>0 then print all the files using for each loop.
   5. Else print that the root directory is empty.
   6. END
2. Adding a user-specified file to the application:
   * + 1. START
       2. Prompt for the name of the file to be added with the path.
       3. Prompt for content to write onto the file using File output stream.
       4. Convert content to bytes.
       5. Write the content onto the file and close using fos.write(); and fos.close();.
       6. Print file is added to the directory.
       7. END
3. Deleting a user-specified file from the application:
4. START
5. Prompt for the name of the file to be deleted with the path.
6. If(file.isDirectory()), then print it as a directory.
7. Else if(file.delete()), then print file is deleted successfully.
8. Else print file deletion failed.
9. END
10. Searching a user-specified file from the application
11. START
12. Prompt for directory path and file name.
13. Assign all the files in that directory path to the files list.
14. Declare flag = false.
15. Compare the user-specified file with all the files in the directory using for loop. If matched, flag = true.
16. If flag = true, print file is found.
17. Else print file is not found.

Core concepts used in the project:

* File Handling
* Exception Handling
* Encapsulation
* Packages

Links to the GitHub repository:

https://github.com/Aniket-Fale/LockedMe.com.git

Demonstration of product capabilities, appearance, and user interactions:

The following sections have been organized to highlight the project's appearance and user interactions in order to demonstrate the capabilities of the product:

1. Creating a project in Eclipse
2. Writing a Java program for the main method ‘AppMain.java’ of the application ‘LockedMe.com’.
3. Writing a Java program for the sub-methods ‘BusinessLevelOperations.java’ of the application ‘LockedMe.com’.
4. Testing and recording the output
5. Pushing the code to GitHub Repository.
6. Creating a project in Eclipse:

* Open Eclipse IDE.
* Go to File 🡪 New 🡪 Project 🡪 Java Project 🡪 Next.
* Give the project name as ‘LockedMe.com’, uncheck ‘Create module-info.java file’ and click Finish.
* Right-click on the ‘LockedMe.com’ project in project explorer 🡪 New 🡪 Class.
* Give the class name ‘BusinessLevelOperations’ and click Finish.
* Create another class as ‘AppMain’ and check ‘public static void main(String[] args)’ and click on Finish.

1. Writing a Java program for the main method ‘AppMain.java’ of the application ‘LockedMe.com’ :

package mypackage;

import java.util.Scanner;

public class AppMain {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("---------------------------------------------------------------");

System.out.println("| WELCOME |");

System.out.println("| LockedMe.com |");

System.out.println("| |");

System.out.println("| developed by |");

System.out.println("| |");

System.out.println("| Aniket Fale |");

System.out.println("| Email ID: ani.rait1075@gmail.com |");

System.out.println("---------------------------------------------------------------");

int ch, ch2;

do {

System.out.println("Please choose one from the below options");

System.out.println("1. Retrieve files in ascending order");

System.out.println("2. Add, Delete and Search a user specified file");

System.out.println("3. Exit");

ch = sc.nextInt();

switch(ch) {

case 1 :

BusinessLevelOperations retrifilesobj = new BusinessLevelOperations();

retrifilesobj.sortingFileNames();

System.out.println();

System.out.println("Retrieved files in an ascending order");

break;

case 2 :

do {

System.out.println("Please choose one from the business level operations below");

System.out.println("1. Add a user specified file to the application");

System.out.println("2. Delete a user specified file from the application");

System.out.println("3. Search a user specified file from the application");

System.out.println("4. Return to Main Menu");

System.out.println("5. Quit");

ch2 = sc.nextInt();

switch(ch2) {

case 1 :

BusinessLevelOperations addfilesobj = new BusinessLevelOperations();

addfilesobj.addingAFile();

break;

case 2 :

BusinessLevelOperations delfilesobj = new BusinessLevelOperations();

delfilesobj.deletingAFile();

break;

case 3 :

BusinessLevelOperations serfilesobj = new BusinessLevelOperations();

serfilesobj.searchingAFile();

break;

case 4 :

System.out.println("Welcome back to Main Menu");

break;

case 5 :

System.out.println("Thank you for using LockedMe.com.");

System.exit(ch2);

default :

System.out.println("Invalid choice. Please select the correct choice");

break;

}

} while(ch2!=4);

break;

case 3 :

System.out.println("Thank you for using LockedMe.com.");

break;

default :

System.out.println("Invalid Choice. Please select the correct choice");

break;

}

} while(ch!=3);

}

}

1. Writing a Java program for the sub methods ‘BusinessLevelOperations.java’ of the application ‘LockedMe.com’ :

package mypackage;

import java.io.File;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Arrays;

import java.util.Scanner;

public class BusinessLevelOperations {

Scanner sc = new Scanner(System.in);

protected void sortingFileNames() {

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");

}

protected void addingAFile() {

System.out.println("Enter name of the file to be added with path : ");

String fileName = sc.nextLine();

try {

File file = new File(fileName);

FileOutputStream fos = new FileOutputStream(fileName);

System.out.println("Enter content to be entered in the file : ");

String content = sc.nextLine();

byte[] b = content.getBytes();

fos.write(b);

fos.close();

System.out.println("'" + file.getName() + "'" + " file is added to the directory");

} catch (IOException e) {

e.printStackTrace();

}

}

protected void deletingAFile() {

System.out.println("Enter name of the file to be deleted 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.getName() + "'" + " file is deleted successfully");

} else

System.out.println("'" + file.getName() + "'" + " file deletion failed");

} catch (Exception e) {

e.printStackTrace();

}

}

void searchingAFile() {

System.out.println("Enter the directory path: ");

String path = sc.nextLine();

System.out.println("Enter the desired file name with extension: ");

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");

}

}

1. Testing and recording the output:

* Right-click on Project 🡪 Hover on ‘Run as’ 🡪 Java Program.

1. Pushing the code to GitHub Repository:

* Open Git Bash and navigate to the folder where you have created your files.
* Initialize the repository using the below command

**git init**

* Add all the files to your git repository using the below command

**git add .**

* Commit the changes using the below command

**git commit -m “first commit”**

* Add the URL for the remote repository where your local repository will be pushed

**git remote add origin https://github.com/Aniket-Fale/Phase1.git**

* Push the files to the folder you initially created using below command

**git push -u origin master**

Unique Selling Points of the Application:

* The application keeps running and taking user inputs even after exceptions occur as the exceptions are handled using try catch blocks.
* User is given the option to append content on to the newly created file.

Conclusion:

The application has been developed according to all the required features mentioned in the project description. Further enhancements to the application can be made by the following:

* Asking if the user wants to delete the file if is not empty.
* Allowing user to retrieve files sorted by Data modified or Type.