1. **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.

1. **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.

1. **Application Name**

LockedMe.com

1. **Developer Details**

Anurag Pal, 216 Java SL (Evening Batch), Phase-1 Final Assessment

1. **Sprint Planning**

To complete this application, three sprints were planned to correspond with the main features of the required application.

**Sprint 1 –** Displayed Application Name and Developer Details, along with the details of the user interface, i.e. a menu displaying three options – Retrieve file names, Business-level Operations, Close. The user is asked to choose one option. This menu is implemented using switch case.

I also implemented the first case, in which a method is called, which I’ve created to retrieve file names from a directory. In this method, user is asked for a directory path which is then checked to verify whether the user has given a proper directory path or not.

If the condition is true, then all the names of the files in that directory are inserted into a string array. And then, elements of that string array is stored into a string ArrayList. Selection sort is used to sort the file names, and then display it.

If the directory check condition is false, then an error message is displayed.

**Sprint 2 –** In this sprint, I implemented the second case and two of its options. Displayed Business-Level Operations and the details of its user interface, i.e. a menu displaying four options – Add a file, Delete a file, Search for a file, Go back to Main Menu. The user is asked to choose one option. This menu is implemented using switch case.

For this, a method is called, which I’ve created to implement Business-Level Operations. In this method, a switch case is implemented for the four options.

In the first case, I ask the user to provide a directory path and file name. Then using a method of the File class, a new file is created. If the file name already exists, then an error message is displayed. The user is then asked for the data that is to be written into that newly created file. After the data is written, a successful message is displayed.

In the second case. I ask the user to provide a directory path and file name. If the file exists, then it is deleted using a method of the Files NIO class and a successful message is displayed. If the file doesn’t exist, then an error message is displayed.

**Sprint 3 –** In the final sprint, I implement the remaining options of Business-Level Operations and the close option from the Main Menu.

In the third case of the method created for Business-Level Operations, I ask the user to provide the directory path where the file needs to be searched. Then it’s checked whether the user has entered a valid directory path or not.

If the condition is true, then I ask the user to provide the name of the file that is to be searched. I also store all the names of the files present in the provided directory path into a string array. After this, I checked all the file names against the file name provided by the user using linear search. If the file was found, then a successful message was displayed, otherwise an error message was displayed.

If the directory check condition is false, then an error message is displayed.

For the fourth case, I used a loop within which all the previous options including this one is present and gave a condition where if the user inputs 4, then a successful message is displayed and the loop is broken. This takes the control back to the previous Main Menu.

And finally, for the third case of the Main Menu, I used a loop within which all the options of the Main Menu including this one, is present and gave a condition where if the user inputs 3, then an exit message is displayed and the loop is broken.

All the sprints were of 1 hour or 60 minutes each.

1. **Algorithm**

a) Start do-while loop

b) Print “Project Name, Developer Details, Menu”

c) Print “1. Get file list, 2. Business-level operations, 3. Close”

d) Print “Enter your choice”

e) Read c

f) If (c==1), call method getFileList()

g) getFileList()

h) Print “Enter directory name”

i) Check if directory exists

j) If true

k) Put all filenames in a string array l

l) Put all the elements of l into a string arraylist

m) Use selection sort to sort the elements and print

n) If false the print “Please enter valid directory”

o) If (c==2), call method bleveloperations()

p) bleveloperations()

q) Strat a do=while loop

r) print “Business-Level Operations”

s) print “1.Add a file, 2. Delete a file, 3. Search for a file, 4. Go back to main menu”

t) print “Enter your choice”

u) read ch

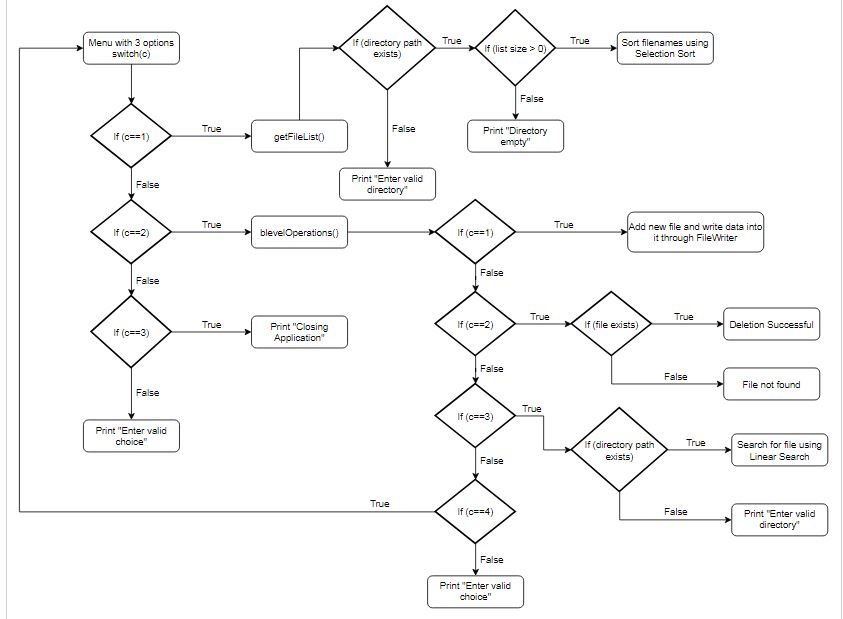
v) if (ch==1) then add new file using FileWriter

w) if (ch==2) then delete file using java.nio.Files.delete

x) if (ch==3) then search for file name using linear search

y) End do-while loop with condition while(ch!=4)

z) If (c==3), print closing message and end do-while loop with condition while (c!=3)

1. **Flow Chart**
2. **Core Concepts Used**

a) Switch Case to map user’s different inputs into cases corresponding to do those inputs.

b) Variables to store user’s inputs and some temporary variables used during Selection sort.

c) Do-while loops to loop the switch cases unless user inputs a specific option.

d) Different methods for different options which are called from the switch cases.

e) ArrayList is used to store the filenames which are then sorted in ascending order using Selection sort.

f) Linear search is used to find the specified file name provided by the user.

g) Different methods of File class and Java NIO class are used to list out all the files of a directory, create a new file, and delete an existing file.

h) To handle any exceptions, all the switch cases and the methods are executed within a try-catch block. Once the exception is found, printstacktrace() method is used to handle those exceptions and errors. This method prints the throwable along with other details like the line number and class name where the exception occurred.

1. **Conclusion**

This application is designed mainly using data structures and collections. Also, selection sort was used for sorting. This can be improved upon with either merge sort or quick sort so that the time complexity goes down. Along with this, linear search was used to search a filename. This can also be improved upon by using binary search or exponential search that will reduce the time complexity as well.

**Unique Selling Points (USPs)**

1. File management can be done efficiently with this application.
2. Works relatively fast even with large of files.
3. Exceptions that arise, if any, are handled immediately.
4. Appropriate messages and options help to guide the user during their first use.
5. User friendly menu implementation helps users to navigate through different options smoothly.