**FINAL SPECIFICATION DOCUMENT**

**Simplilearn-**

**Full stack Java**

**Developer (Assignment-1)**

**Name of Project: Lockedme.com**

**Documentation prepared by: Anjali Gopalakrishnan**

**Date submitted:**

**21-09-2022**

**Version 0.1**

| REVISION HISTORY | | | |
| --- | --- | --- | --- |
| DATE | VERSION | DESCRIPTION | AUTHOR |
| 21/9/22 | 0.1 | First version of final specification document | Anjali Gopalakrishnan |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[INTRODUCTION 3](#_Toc115378077)

[PURPOSE 3](#_Toc115378078)

[PROJECT DETAILS 3](#_Toc115378079)

[SOFTWARE, PROGRAMMING LANGUAUGE AND OTHER CONCEPTS TO BE USED 3](#_Toc115378080)

[SCRUM PLANNING 4](#_Toc115378081)

[SPRINTS PLANNED 4](#_Toc115378082)

[FLOWCHART 5](#_Toc115378083)

[ALGORITHMS USED 6](#_Toc115378084)

[BUBBLE SORT 6](#_Toc115378085)

[BINARY SEARCH 6](#_Toc115378086)

[CORE JAVA CONCEPTS USED 6](#_Toc115378087)

[LINKS TO GITHUB REPOSITORY 7](#_Toc115378088)

[Project Folder, Final specification document and screenshots – 7](#_Toc115378089)

[CONCLUSION 7](#_Toc115378090)

# INTRODUCTION

## PURPOSE

Company Lockers Pvt. Ltd. aim to digitize their products and chose LockedMe.com as their first project to start with. The objective here is to develop a prototype of the application. The prototype of the application will be then presented to the relevant stakeholders for the budget approval. This document is prepared to give developers, users, testers, managers and stakeholders a clear idea of the project.

## PROJECT DETAILS

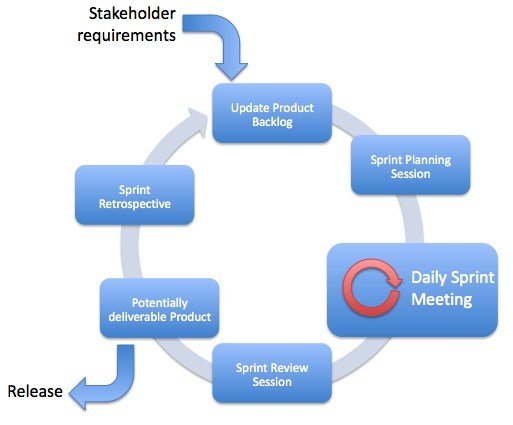
The goal of the company is to deliver a high-end quality product as early as possible. As this is a prototyped application, the user interaction will be via a command line. As the first step protype is created to perform following business operations.

* 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

## SOFTWARE, PROGRAMMING LANGUAUGE AND OTHER CONCEPTS TO BE USED

* 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

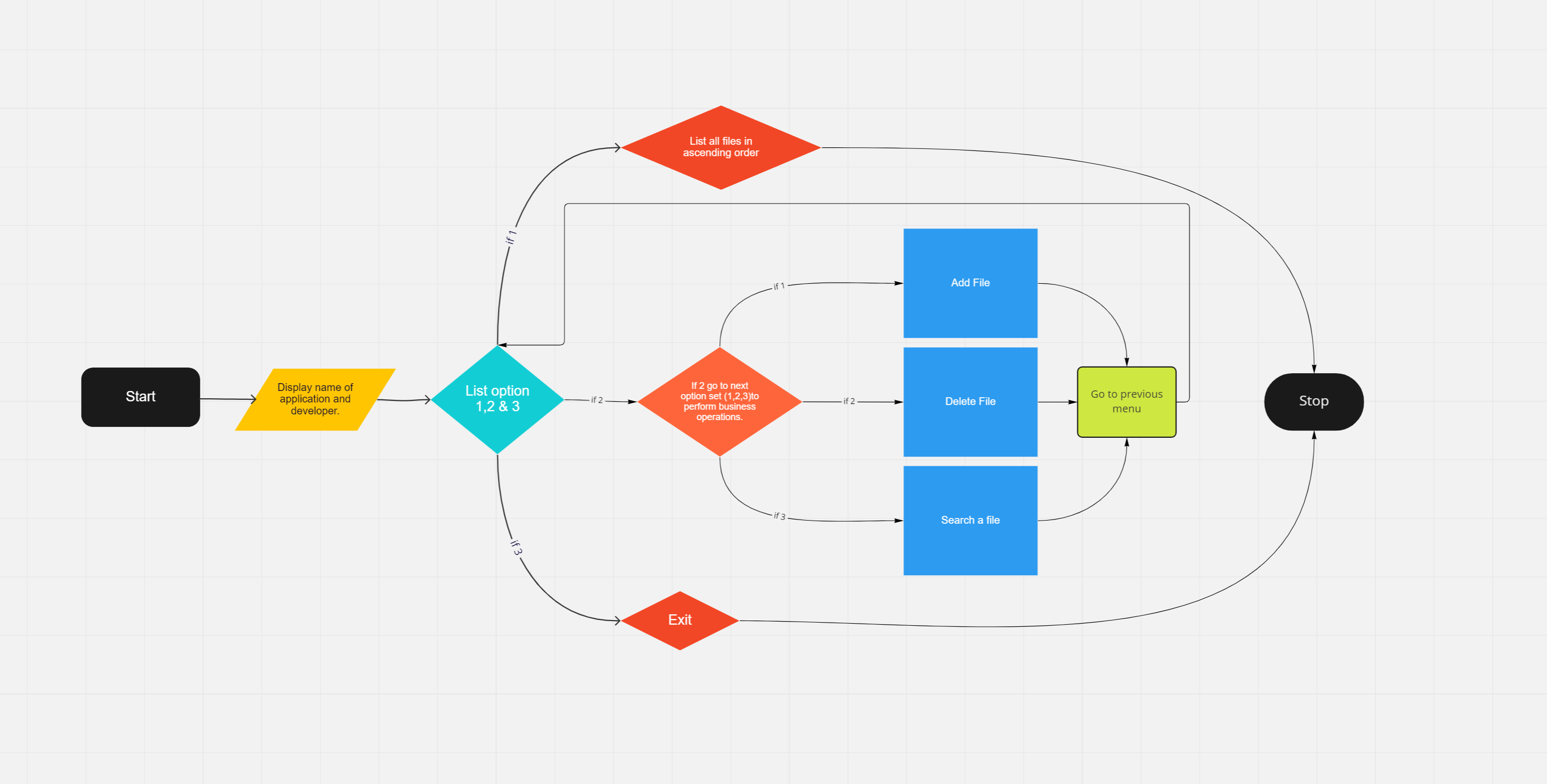
# SCRUM PLANNING

****

## SPRINTS PLANNED

* SPRINT 1
  + Goal to be accomplished – Creation of specification document and uploading to GitHub
  + Time allocated – 19/9/22 to 21/9/22
  + Software – MS Word
  + Deliverable product – The final product specification document
* SPRINT 2
  + Goal to be accomplished – Coding the welcome screen
  + Time allocated – 22/9/22 to 23/9/22
  + Software – Eclipse (IDE), Java, GitHub
    - Subtasks
      * 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
      * Uploading code to GitHub
  + Deliverable product – Code for the welcome screen
* SPRINT 3
  + Goal to be accomplished – Coding business operations
  + Time allocated – 23/9/22 to 29/9/22
  + Software – Eclipse (IDE), Java, GitHub
    - Subtasks
      * Code the three options to perform user defined operations
      * 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
      * Uploading code to GitHub
  + Deliverable product – The final prototype implementing all functions

# FLOWCHART



# ALGORITHMS USED

## BUBBLE SORT

|  |  |
| --- | --- |
| **DESCRIPTION** | Files in the given directory are sorted using the bubble sort algorithm. CompareTo() is used to compare two file names lexicographically. Each character of both strings is converted into its unicode value. Lexicographical order is nothing but alphabetical order. This method returns an int data-type which is based on the comparison between the two file names. If it returns>0 then the parameter passed to compareTo() method is lexicographically first whereas if returns < 0 then file name calling the method is lexicographically correct. |
| **ALGORITHM** | Step 1: Get the file names in the ArrayList  Step 2: Compare first and second filename in the ArrayList using compareTo()  Step 3: If compareTo() returned an integer >0, swap the numbers  Step 4: Continue the steps to compare all filenames in the ArrayList  Step 5: Print the sorted ArrayList |

## BINARY SEARCH

|  |  |
| --- | --- |
| **DESCRIPTION** | Binary search algorithm is used to check if a file is present in the given directory. This algorithm search element in a sorted arraylist by repeatedly dividing the search interval in half. Begin with an interval covering the whole arraylist. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty |
| **ALGORITHM** | Step 1: Find the middle element  Step 2: Compare file name to be found with the value in the middle element using compareTo()  Step 3: If the value returned by compareTo() is 0, print “File exists”.  Step 4: If the value returned by compareTo() is greater than 0, then filename can only lie in the right half subArrayList after  the mid element. So, we recur for right half.  Step 5: Else filename is in left half subArrayList and recur for the left half.  Step 6: Repeat till compareTo() returns 0, then print “File Exists”  Step 7: Else print “File does not Exists” |

# CORE JAVA CONCEPTS USED

* Exceptions
* Collections
* Binary Search
* Bubble sort
* Static variables
* Files
* Scanner class

# LINKS TO GITHUB REPOSITORY

## Project Folder, Final specification document and screenshots –

[AnjaliRadhe/LockedMe (github.com)](https://github.com/AnjaliRadhe/LockedMe)

# CONCLUSION

A prototype to perform basic operations has been created. More functionalities and an interface for improved user experience can be added in the following releases. Fast responses and uncluttered view give better shopping experience for the user.