Specification Document for
LockedMe.com Application Prototype

Gabriel Villar

Simplilearn Phase 1 Phase End Project

March 3, 2020

GitHub: https://github.com/gvillar08/HCL-Training-Projects.git

Table of Contents

Project Requirements and Overview

Lockers Pvt. Ltd. is aimed at digitizing their products and has selected LockedMe.com as their first project to strive towards their goal. As this application is still in its early prototype phase, user interface and interaction is limited to the command line for the time being. As this is the case, functionality takes precedence over UI/UX. The goal of this LockedMe prototype is to display a welcome screen, from which you can navigate to a secondary menu to interact with a working directory.

The welcome screen is to display the application name and developer details, along with the following user-selectable options: return the current file names in ascending order, navigate to the secondary menu, or quit. The first option utilizes merge sort to achieve the file sorting, which will be discussed in further detail in the Application Algorithms section.

When the second option is selected, the user is redirected to a separate menu for interacting with the directory. Here, the user can add a file, delete a file, search for a file, return to the main menu, or quit. For adding, deleting, or searching for a file, the user simply types in the desired file name and the respective action is completed. Adding adds the user's text entry to the ArrayList of known files. Deleting removes the file if found from the list of known files. Searching for a file returns the file's numeric position in the alphabetized file hierarchy if found. Both deleting and searching are case-sensitive activities.

Universally among the screens, the program can handle invalid user entries without impeding the user experience. All user entries and actions result in an explicit response. When applicable, the application will notify the user if the desired operation was successful or could not be completed. This prototype currently utilizes test files and is limited to its current working directory. It cannot navigate through folders or view folder hierarchy.

The code for the program can be found at https://github.com/gvillar08/HCL-Training-Projects.git under the lockedMe folder.

Agile Framework

The project is split into three sprints, with each sprint lasting one day

- Sprint 1: Create the project’s framework
    - Create Main class
    - Create class to handle screens
    - Create class to handle back-end directory functions
    - Create test data
- Sprint 2: Create the Welcome Screen
    - Create screen interface
    - Create back-end code for options to function
- Sprint 3: Create the Directory Interaction Screen
    - Create screen interface
    - Create back-end code for options to function
    - Ensure final prototype functions as intended
        - Ensure menu-switching options work in both screens

Scrum Reports

Sprint 1

During sprint 1, LockedMe's framework was successfully created. Within the Screens and Directory classes, each had their own ArrayList for storing options and files respectively. A method was added to Screens for adding test data. To test the framework's functionality, crude println statements have been implemented in the Main class for now.

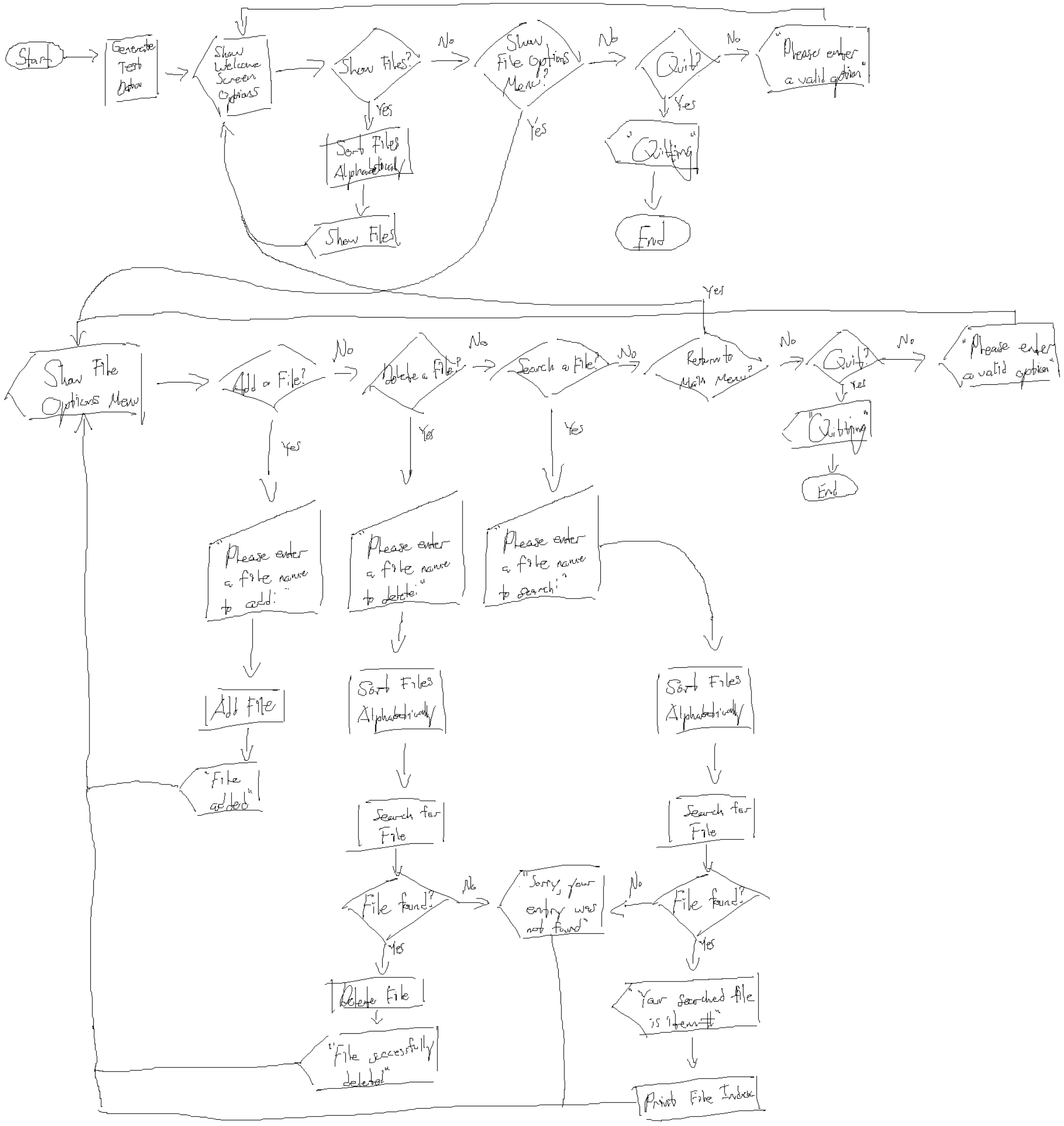
Sprint 2

In this sprint, the Welcome Screen and relevant methods were created. Simple println statements were used for satisfying the requirement to output the home screen header and developer details. From here, a while loop was used for keeping the user within the menu until they quit or leave to another menu. Screens.PrtSc() was created to handle printing the options and clearing them for the next menu to utilize the options ArrayList. The switch created for the option's functionality utilizes the Screens.getOption() method. This method asks for user input and handles exceptions for non-numeric entries. sortDirectory() and mergeSort() were created in Directory for sorting the list of files. A merge sort algorithm was utilized for the sorting, which is described in further detail in the Application Algorithms section. Directory.printFiles() was created for simply printing all of the files. The second option for switching menus currently does not work, so selecting this option returns the user to the options interface. The third option successfully quits the program. The default case handles notifying the user of any input that does not match one of the three options. The Main class's println statements were swapped for a couple lines to boot the Screens.AddTestData and Screens.WelcomeScreen methods.

Sprint 3

This sprint successfully implemented the Directory Service screen. This screen is simply another method in the Screens class. The Welcome Screen's code was copied and pasted initially as the two screens should function quite similarly. What's new in this method are the functions for the first four options. The first option utilizes a new Screens.fileInput method for receiving user input that is non-numeric. This method's result is passed to a new Directory.appender method to create and add a new file. The second option passes the fileInput() to a new Directory.delete method for deleting a specified file. This option was created after creating the third option since it utilizes a Directory.searchFile method for searching the provided file before deleting it. searchFile() sorts the files before searching for the user input. If it is a match, it returns the index of the file, otherwise, it handles notifying the user of no match. If the file is found, delete() deletes the file found at the returned index and notifies the user. As already described, option 3 utilizes the searchFile method to search for a user’s entry via the fileInput method. This option then prints the index of the file plus one if it is found. The fourth option calls the WelcomeScreen() to return the user to that menu. Similarly, this feature was implemented in the WelcomeScreen method as well. Aside from ensuring the menu-swapping options work, the code worked as intended with no further needs to make changes to the Main class for it to run.

Application Flowchart


Start
Generate Test Data
Show Welcome Screen Options
Show Files?
No
Yes
Sort Files Alphabetically
Show Files
Show File Options Menu?
No
Yes
Quit?
No
Yes
"Please enter a valid option"
"Quitting"
End
Yes
Show File Options Menu
Add a File?
No
Yes
"Please enter a file name to add:"
Add File
"File added"
Delete a File?
No
Yes
"Please enter a file name to delete:"
Sort Files Alphabetically
Search for File
File found?
No
Yes
Delete File
"File successfully deleted"
"Sorry, your entry was not found"
Search a File?
No
Yes
"Please enter a file name to search:"
Sort Files Alphabetically
Search for File
File found?
No
Yes
"Your searched file is 'Item #'"
Print File Index
Return to Main Menu?
No
Quit?
No
Yes
"Please enter a valid option"
"Quitting"
End

Application Algorithms and Core Concepts

The application utilizes only one algorithm, which is used for sorting the files. The algorithm is well documented via comments in the code, but to reiterate it, in the Directory class, a merge sort algorithm is implemented over the course of two methods: sortDirectory and mergeSort. As this is a "divide and conquer" sorting algorithm, sortDirectory handles the "dividing" while mergeSort handles the "conquering." sortDirectory recursively calls itself to split the utilized ArrayList full of files into right and left segments until it reaches one element in either segment. It then calls mergeSort to compare the values in either segment and it works its way back up the chain of recursion till it is completely sorted at the top. This algorithm can be easily modified for different input types, such as strings or integers. In this case, strings are used, and to ensure capitalization does not interfere, strings are compared all in lowercase.

Core concepts utilized for this LockMe application to function are exceptions, ArrayLists, and switches. InputMismatchException is utilized in Screens.getOption() when checking if the user's input is an integer via a try-catch statement. ArrayLists are employed twice for handling arrays of dynamic lengths. This first array is for storing the menu options, while the second array is used for storing the files. Switches are critical to the application as they serve as the means for the user to have their option input be translated to the option's respective action.

Key Features and Future

The unique selling points of this LockMe application are its robustness and modularity. Currently, the application handles user input very well and has not faced any compilation or runtime issues. As for its modularity, introducing new screens or building on existing ones would require little modification to existing code to “plug-in” the new code. Building the first screen took the most work out of the three sprints. When it came time for the third sprint for creating the second screen, it was fairly easy as many of the necessary methods were already in place.

Although the code has a couple selling points, the LockMe application leaves much to be desired aesthetically and functionally. Starting on the front-end, the program is not user-friendly as it utilizes an archaic command line GUI, which only accepts numeric entries to navigate. Even if made to run in the user’s web browser, as LockMe.com’s name implies, the application should have the ability to accept user interaction in the form of a cursor and arrow keys, among many other functions. As for the functionality, the program should at the very least be configured to handle the user’s directory, or at least a specified portion of it. It should be able to navigate nested directories and discern folders from files. When adding new files, the user should be able to choose if they are creating a new folder or file. Considering the user can create files, the user should be able to do basic text editing in applicable files and read them. The searching function should be expanded to search all of the nested directories and changed to return the file/folder directory path instead of a numeric position. When adding or deleting, the scope for these should be expanded by the user specifying the file path, rather than being limited to the current working directory. With all of this in mind, LockMe is definitely in its early prototype stages and has plenty of work ahead of it before becoming a final user product.