Specification Document - LockedMe Prototype

Author: Trey Turley

1. Introduction

1.1 Purpose

The purpose of this product is to create a prototype that will be demonstrated to stakeholders so that a decision can be made on whether to proceed and approve the required budget.

1.2 Intended Audience

The audience of the prototype are the stakeholders at Lockers Pvt. Ltd. who will be deciding on whether or not to approve the budget for the full application.

1.3 Intended Use

The prototype will be made available to the stakeholders so that they can assess its features.

1.4 Scope

The prototype being developed will be limited to just a few features. Users will be able to add a file, search for a file, and delete a file from the application. They will be able to go back to the main menu should they not want to complete an action that was in progress. The User will also be able to close the application.

1.5 Definitions and Acronyms

None

1.6 GitHub

This project utilizes GitHub. The repo can be found at: https://github.com/TreyT-FSD/Phase1_Project

2. Overall Description

2.1 User Needs

2.1.1 Primary User

In this prototype, all users will be considered Primary users. Primary users will have access to all features so that the complete application can be assessed.

2.2 Assumptions and Dependencies

- The provided specifications say "Option to add a user specified file to the application". In the LockedMe Prototype we are assuming that adding a user specified file means the user specifies a file name and that file is created and added to the LockedMe directory. In the context of the LockedMe Prototype, "adding a file" does not mean specifying the name or path of an **existing file** and having that file be copied into the LockedMe directory.

- The specifications says "Retrieving the file names in an ascending order". Because we currently do not know if there is a typical filename structure that is going to be used we cannot make any reasonable assumptions about a preferred ordering for the files. In the LockedMe Prototype we are assuming that the ascending order is an alphanumeric ordering of the file names.

- The specification uses language like "You can ignore the case sensitivity of the file names" and "You can add the case sensitivity on the file name to retrieve the correct file" to describe working with files. Due to how different operating systems handle case sensitivity for filenames, The LockedMe Prototype makes no attempt to handle case sensitivity. This should result in case sensitivity being OS dependent. For example, on a windows machine, the file a.txt and A.txt are considered to be the same. This means they can't both exist at the same time. It's possible that another OS may allow users to add or delete files with names like a.txt and A.TXT.

2.3 Application Appearance

This application is a prototype for LockedMe and therefore will not have a GUI. All user interaction will be via command line.

2.4 User Interaction

All user interaction will be via command line.

3. System Features and Requirements

3.1 Functional Requirements

3.1.1 Retrieve the list of files

- Retrieve the files that have already been added to the LockedMe directory
- Files in the LockedMe directory will be displayed in ascending alphabetical order

3.1.2 Add user specified file

- Users will be able to specify a filename and that file will be created and added to the LockedMe directory.

3.1.3 Delete user specified file

- Users will be able to specify a filename and that file will be deleted from the LockedMe directory.

3.1.4 Search for user specified file

- Users will be able to specify a filename and that file will be searched for in the LockedMe directory.

3.1.5 Close the application

- An option to close the application will be available

3.2 External Interface Requirements

None

3.3 System Features

The prototype will be designed and implemented such that it can be run from the command line. No GUI will be available and as such all interaction through the command line.

3.4 Non-Functional Requirements

No specifications were provided for any non-functional requirements such as SLAs for user interactions or response times. However, an effort will be made to use efficient algorithms where needed. **3.5**

Future Enhancements

To make the application more user friendly and to expand on its capabilities, a number of features could be added. Below are some examples.

3.5.1 Additional sorting options

The prototype lists the files in alphabetical order. Additional sorting options could be added such as

- Sort by file size
- Sort by file type
- Sort by date modified

3.5.2 Additional sorting orders

- Numerical order
- Natural order

3.5.3 Add multiple files at the same time

Allow users to specify multiple filenames at one time so that they can be added simultaneously.

3.5.4 Delete multiple files

Allow users to delete multiple files simultaneously.

3.5.5 Delete all files

Allow users to delete all files currently added to the application

3.5.6 Add all files from a user specified directory

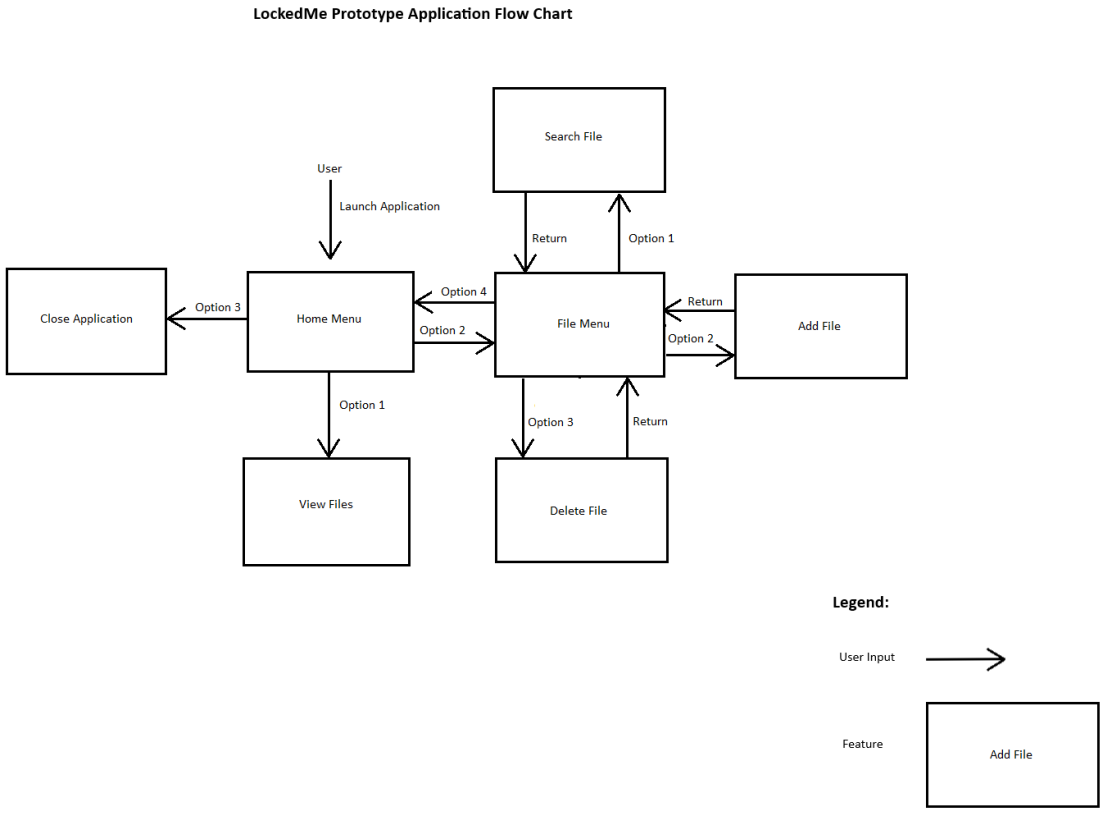
Allow users to specify a directory from which all files will be added.

3.5.7 Add one or more files from a specified directory

Allow users to specify a directory and then select one or more files from that directory to be added.

3.6 System Interaction Flow Chart

As all interaction is done through the command line, the user will be presented a number of options that will let them navigate through the different features of the application. The flow chart below illustrates the different interactions that lead to the various features of the application.



4. Application Details

4.1 Java Concepts

Classes, Objects, and Encapsulation

The LockedMe Prototype uses two classes. One class, LockedMeProgramLoop, contains the Main function and is the entry point for program execution. The second, LockedMeApp, encapsulates the functionalities of the LockedMeProtype into a single unit so that all functionalities are logically grouped together and operate on the same LockedMe directory and files.

4.2 Data Structures

The application makes use of a TreeSet to store files. This has the benefit of automatically sorting the files so that they can be displayed in ascending order.

4.3 Algorithms

The only algorithm we would have needed would be for sorting the files for showing them to the user. We used a TreeSet to handle so no discrete sorting algorithm was needed.

5. Sprint Planning

Below is the breakdown of the Sprints, Stories, and Tasks for implement the LockedMe Prototype.

Sprint 1 (1 week):

Story 4:
As a developer, I want to store all of the user files in one directory so that they remain organized in one place.

Task 4: Implement directory creation on program startup.

Story 7:
As an user, I want to be able to view all of the files in alphabetical order so that I can see what is currently available in the application.

Task 10: Implement retrieving all of the file from the directory.

Story 8:
As an user, I want to search for a file to see if it has been added so that I can see if it exists or not already.

Task 11: Implement file search.

Story 5:
As an user, I want to be able to add a file with a name I specify to the application so that I can create files with the name I want.

Task 5: Implement adding a file to the directory.

Story 6:
As an user, I want to be able to delete files that I no longer need so that I can reduce the number of unnecessary files.

Task 9: Implement deleting a file from the directory.

Sprint 2 (1 week):

Story 1:
As an user I want to be able to perform multiple interactions with the application without having it close or restart so that I am able to work faster.

Task 1: Implement main program loop.

Story 2:
As an user, I want to be able to select a specific operation from a menu so that I can choose what actions the application performs.

Task 2: Accept user input in the main program loop.

Task 7: Implement Home menu.

Task 8: Implement File menu.

Sprint 3 (1 week):

Story 3:
As an user, I want to be able to close the application gracefully so that I have a good experience.

Task 3: Ensure application closes gracefully.

Task 12: Ensure exceptions are handled appropriately.