# Software Requirements

Version 51, last updated by [rylandm](https://uocis.assembla.com/profile/rylandm" \t "_blank) at 2018-02-02

# Software Requirements Specification (SRS)

Revision History:

|  |  |  |
| --- | --- | --- |
| Date | Author | Description |
| 1-18-18, 1-19-18 | Kaela S. | Adding/Editing Use Cases |
| 1-21-18 | Ryland M. | Con Ops, System Context, System Capabilities, Expected Subsets |
| 1-22-18 | Kaela S. | Updating Use Cases Based On Customer Meeting System Context Expected Changes |
| 1-23-18 | Ryland M. | Quality Requirements, System Inputs and Outputs, Detailed Output Behavior |
| 1-24-18 | Sam E. | Introduction |
| 1-29-18 | Kaela S | Use Cases for Iteration L1 Update I/O |
| 1-29-18 | Sam E. | Quality Requirements |
| 2-1-18 | Kaela S | Update I/O |
| 2-1-18 | Ryland M | Minor updates to sections to ensure they meet rubric specifications. |
| 2-1-18 | Freddie W | Updated Use Case for post-L2 |

Contents

[Software Requirements 1](#_Toc51163622)

[Software Requirements Specification (SRS) 1](#_Toc51163623)

[1.  Introduction 2](#_Toc51163624)

[1.1    Intended Audience and Purpose 2](#_Toc51163625)

[1.2    How to use the document 2](#_Toc51163626)

[2.  Concept of Operations 3](#_Toc51163627)

[2.1    System Context 3](#_Toc51163628)

[2.2 System capabilities 4](#_Toc51163629)

[3.  Use Cases 4](#_Toc51163630)

[Case 1: User Wants to Quit the Application 4](#_Toc51163631)

[Case 2: User Wants to Open Application 5](#_Toc51163632)

[Case 3: User Wants to Create a New Address Book 5](#_Toc51163633)

[Case 4: User Wants to Save Address Book 6](#_Toc51163634)

[Case 5: User Wants to Save Address Book Under New Name/Location (The "Save As..." Case) 6](#_Toc51163635)

[Case 6: User Wants to Close the Current Address Book 7](#_Toc51163636)

[Case 7: User Wants to Open Existing Address Book 7](#_Toc51163637)

[Case 8: User Wants to Sort Address Book 8](#_Toc51163638)

[Case 9: User Wants to Delete an Entry 8](#_Toc51163639)

[Case 10: User Wants to Add New Entry 8](#_Toc51163640)

[Case 11: User Wants to Edit an Entry 9](#_Toc51163641)

[Case 12: User Wants to Export Contacts To A TSV File 9](#_Toc51163642)

[Case 13: User Wants to Import Contacts From A TSV File 10](#_Toc51163643)

[Case 14: User Wants to Search for Contact(s) 10](#_Toc51163644)

[Case 15: User Wants to Rename a Address Book 11](#_Toc51163645)

[3.    Behavioral Requirements 11](#_Toc51163646)

[3.1 System Inputs and Outputs 11](#_Toc51163647)

[3.2 Detailed Output Behavior 13](#_Toc51163648)

[4.2   Quality Requirements 13](#_Toc51163649)

[5.    Expected Subsets 14](#_Toc51163650)

[6.    Fundamental Assumptions 14](#_Toc51163651)

[7.    Expected Changes 14](#_Toc51163652)

[8.    Appendices 15](#_Toc51163653)

[8.1    Definitions and acronyms 15](#_Toc51163654)

[8.2    References 16](#_Toc51163655)

## 1.  Introduction

### 1.1    Intended Audience and Purpose

This document is intended to provided information guiding the installation and development process, ensuring that all system requirements are met. The following entities may find the document useful:  
Primary Customer - This page will detail all of the application requirements as understood by the production team. The customer should be able to determine that their requirements will be correctly reflected in the final product through the information found on this page.  
User - A prospective user will be able to use this document to identify the main functionailty included in the application. Furthermore, the application will have a set of system requirements before the application can be run. Details regarding these requirements can be found here.  
Development Team - Details of specific requirements that the final software build must include will be located here. Developers can use this document to ensure the software addresses each of these requirements.  
QA Team - By developing testing procedures founded in the system requirements, the QA Team can create a comprehensive testing regimen that will guarantee requirements are met.

### 1.2    How to use the document

Table of Contents:  
  
1. Introduction  
2. Concept of Operations - broad description of the purpose of the application  
  2.1 System Context - details any specific system requirements the application will require to run  
  2.2 System Capabilities - description in prose of all capabilities available to the user in the address book  
  2.3 Use cases - A detailed look at each functional requirement, describing the application context both before and after an action is taken  
3. Behavioral Requirements - How the application will interact with a user  
  3.1 Input and output requirments - A description of allowed inputs and generated outputs  
    3.1.1 Input - Describes any restrictions that will be placed on allowed input  
    3.1.2 Output - Describes the range of outputs that can be generated  
  3.2 Detailed Output Behavior - Output descriptions in prose  
4. Quality Requirements - Requirements not pertaining to the function of the application will be listed here  
5. Expected Subsets - Expected levels of functionality at checkpoints during development  
6. Fundamental Assumptions - Some specifics about input, output, or behavior upon which other requirements are founded will be listed here  
7. Expected Changes - Future features and directions the project is expected to take  
8. Appendicies - Details aiding the understanding of this document  
  8.1 Definitions and acronyms - Any technical terms or abbreviations will be spelled out here for ease of use of the document  
    8.1 Definitions - Definitions of technical or unusual terminology  
    8.1.2 Acronyms and Abreviations - Any abreviated terms will be expanded here  
  8.2 References - any external references necessary or helpful to understanding this document will be listed here

## 2.  Concept of Operations

  The goal is to create a user friendly, standalone address book application. It will allow its users to save multiple address books each with contacts' names, addresses, emails, and phone numbers. A user must have the application installed and Java installed on their machine. The application uses drop down menus, text boxes, and file system navigation windows to interact with the user. For more details on the usage and capabilities of the application read the section, [System Capabilities](https://uocis.assembla.com/spaces/cis422w18-team2/wiki/Software_Requirements#System%20Capabilities).

### 2.1    System Context

**System Requirements:**  
Requires a system with a GUI display because all of the operations are performed through a GUI. The application is in Java so users must have an updated version of Java installed on their machine to use the application.

Windows:

* Windows 10 (8u51 and above)
* Windows 8.x (Desktop)
* Windows 7 SP1
* Windows Vista SP2
* Windows Server 2008 R2 SP1 (64-bit)
* Windows Server 2012 and 2012 R2 (64-bit)
* RAM: 128 MB
* Disk space: 124 MB for JRE; 2 MB for Java Update
* Processor: Minimum Pentium 2 266 MHz processor

Mac OS X:

* Intel-based Mac running Mac OS X 10.8.3+, 10.9+
* Administrator privileges for installation

Linux:

* Oracle Linux 5.5+1
* Oracle Linux 6.x (32-bit), 6.x (64-bit)2
* Oracle Linux 7.x (64-bit)2 (8u20 and above)
* Red Hat Enterprise Linux 5.5+1, 6.x (32-bit), 6.x (64-bit)2
* Red Hat Enterprise Linux 7.x (64-bit)2 (8u20 and above)
* Ubuntu Linux 12.04 LTS, 13.x
* Ubuntu Linux 14.x (8u25 and above)
* Ubuntu Linux 15.04 (8u45 and above)
* Ubuntu Linux 15.10 (8u65 and above)

### 2.2 System capabilities

 The address book is a standalone application. Once downloaded, the user can open it as a normal application on their computer. After it launches it will open the last open address book (or a new one if it's the first time the application is being launched). If a new address book is opened, they will be prompted to name the address book, once named a blank address book will be open.  
  From this point the user may make changes to the address book. Allowed changes are: adding a new entry, editting an existing entry, and deleting an existing entry. If the user wishes to add a new entry, then they will be asked to provide information about the entry. At least one name (first or last), and one other field is required for a new entry. As long as the user gives the necessary information, that contact is added to their address book and appear in the UI. A user is allowed to edit any section(s) of an entry as long as their final edits include the required fields for creating an entry. If a user deletes an entry then they are prompted to confirm the delete. If they do, the entry no longer appears in the UI. None of these changes are permanent unless the user saves the address book.  
  The user is able to save their address books. When they do this for the first time they will be allowed to navigate their file system and choose where they would like to save the address book. In future saves the address book will be saved to the same location as it previously was. The alternative is to choose the "save as" option, which allows an address book to be saved to a new location and/or with a new name.  
  Each address book is sorted by last name by default. Users may choose to sort it by zip code instead. However, each time they close an address book it will revert to its default sorted state. The user may close a single open address book, when they do this they will be asked if they want to save any unsaved changes they have made to it. If the last open address book is closed, then the application quits. However, the user may also quit the application by selecting that option from the "File" menu. When they do this all open address books will attempt to close, they will be prompted to save each one with unsaved changes.

## 3.  Use Cases

### Case 1: User Wants to Quit the Application

**Players:**End User  
**Goals:**The end user wants to close the application and any open address books.  
**Preconditions:**The application is open and running.  
**Case:**  
1.1 From the File menu, the end user selects the "Close all and Quit" option.  
1.2 The application closes all currently open address books  
1.3 The application terminates itself.  
**Alternate Flows:**  
1.2.1 One or more of the address books that are open have unsaved changes  
         The user is warned about quitting before saving their changes  
      1.2.1.1 The user decides to save their changes (See Case 4)  
                  The application closes after all changes are saved.  
  
      1.2.1.2 The user decides to continue with the termination.  
                  No changes are saved. The application terminates.  
**Exception Flows:**  
1.2.2 The user forces the termination (by shutting down their machine, using Mac's Force Quit, Quitting from the Application menu on the Menu Bar on a Mac, etc)  
         No changes are saved.  
**Postconditons:**If a user saved the changes made to their address books, the changes should persist the next time that address book is opened. Otherwise, the address book should be in the same state as it was when it was last opened. No address books are open. If a user navigates to Task Manager (or equivalent process manager), there is no trace of the application still runnning on the user's system.

### Case 2: User Wants to Open Application

**Players:**End User  
**Goals:**The end user would like to run the application so that they can interact with their address books.  
**Preconditions:**The application is downloaded on the user's machine.  
**Case:**  
2.1 The end user opens the application  
2.2 The application opens with the most recently closed address book displayed.  
**Alternate Flows:**  
2.2.1 There is no "most recent address book" -- the application is opening for the first time or the most recent address book has been moved or deleted  
         The user is presented with a choice to either create a new address book or open an existing one  
**Exception Flows:**  
2.1.1 The application was not downloaded correctly.  
         The application does not open.  
**Postconditions:**The application is open and running, waiting for its next instruction from the end user.

### Case 3: User Wants to Create a New Address Book

**Players:** End User  
**Goal:** The user would like to create a new, empty address book.    
**Preconditions:** The application is open and running.  
**Case**:  
3.1 From the file menu, the end user selects the "New" option.  
3.2 The user is able to choose a name for their address book.  
3.3 A new window opens with a new, empty address book.  
**Alternate Flows**:  
3.1.1 From the initial state of the application (step 2.2 from Case 2), the end user chooses to create a new address book.  
         The flow resumes at step 3.2  
**Exception Flows:**  
3.2.1 The user decides to "cancel" the process after deciding to create a new address book.  
          The application returns to its initial state.  
**Postconditions:**The application is "holding" the new address book. A new file with the new address book's contents and metadata has not been created at this point.

### Case 4: User Wants to Save Address Book

**Players:** End User  
**Goal:**The end user wishes to save any changes made to the current active address book. If a user has executed this workflow, then the "Quit Application" workflow should not remind the user to save changes.  
**Preconditions:** The application is open and running.  
**Case:**  
4.1 From the file menu, the end user selects the "Save" option  
4.2 Application saves the currently active address book in its current location and with its current filename.  
**Alternate Flows:**  
4.2.1 If the address book has not ever been saved before, the user will be prompted to select a filename and navigate their file system to select a location for their address book.  
         The end user will then be returned to the current active address book.  
**Exception Flows:**  
4.2.2 If the filename or location of the address book is not valid, the end user will be warned by the application  
         The end user is prompted to enter valid data.  
4.2.3 If the name of the address book is the name of an existing address book, the application will warn the user  
  
**Postconditions:** The address book that has been saved will show recent changes if closed and reopened. The application is still running and any currently active windows are still open. If the user has made changes since the address book was opened, then the application "forgets" that the user has made changes and thus the user can exit the application without being warned about unsaved changes.

### Case 5: User Wants to Save Address Book Under New Name/Location (The "Save As..." Case)

**Players:**End User  
**Goals:**The user wishes to save their address book (or the changes to their address book) under a new name or in a new location.   
**Preconditions:**The application is open and running.  
**Case**:  
5.1 From the file menu, the end user selects the "Save As.." option  
5.2 The user visually navigates their system  
5.3 The user chooses a location and filename for the address book  
5.4 The user confirms their decision  
**Alternate Flows:**  
5.3.1 If only the location changes, the filename is assumed to stay the same.  
5.3.2 If only the filename changes, the location is assumed to stay the same.  
**Exception Flows:**  
5.4.1 The user provides an invalid filename or location for the save  
          The user is prompted by the application to enter valid data  
5.4.2 The user provides a filename that already exists  
          The application warns the user.  
**Postconditions:**The address book that has been saved will show recent changes if closed and reopened. The user's filesystem will be updated, the application is still running, and any currently active windows are still open. If the user has made changes since the address book was opened, then the application "forgets" that the user has made changes and thus the user can exit the application without being warned about unsaved changes.

### Case 6: User Wants to Close the Current Address Book

**Players:**End User  
**Goals:**The end user wants to close the window that contains the current address book.  
**Preconditions:**The application is open and running.  
**Case:**  
6.1 From the file menu, the end user selects the "Close" option  
6.2 The application closes the currently active address book.  
**Alternate Flows:**  
6.2.1 If the address book has been changed but not saved, the user will be prompted to save their changes  
         6.2.1.1 The user decides to save their changes which follows the case 4 workflow  
         6.2.1.1 The user decides to discard their changes  
                      The application "forgets" the changes made to the address book and closes it  
6.2.2 The user decides to "cancel" the close  
          The application returns to its initial state  
**Exception Flows:**  
See Case 4 and 5.  
**Postconditions:**The current active address book is closed.

### Case 7: User Wants to Open Existing Address Book

**Players**: End User  
**Goals:**The user wants to open an address book that already exists on their system. After opening this address book, it should be open in its own dedicated window.  
**Preconditions:** The application is open and running. The address book they want to open is saved somewhere on their system.  
**Case:**   
7.1 From the file menu, the end user selects the "Open" option  
7.2 The user visually navigates their file system.  
7.4 The user selects the address book they want to open  
7.5 The contents of their address book are loaded into the GUI.  
**Alternate Flows:**  
7.5.2 The end user changes their mind/cannot find the file they're looking for/etc and wants to effectively "cancel" the workflow.  
         The end user navigates out of the file system navigator  
          The application returns to its previous state  
7.5.3 The end user has an address book open prior to initiating this flow.  
         When the user chooses which address book they'd like to open  
         A new window is spawned   
**Exception Flows:**  
7.5.1 The selected file is an incorrect format, corrupted, etc.  
         The user tries to open it and is warned by the application.  
         The user returns to step 7.2  
**Postconditions:**The address book is open in its own window. The contacts are displayed in the GUI.

### Case 8: User Wants to Sort Address Book

**Players:**End User  
**Goals**: The user wants to sort the entries in their address book based on one of the pre-determined parameters.  
**Preconditions:**The application is open and running with an address book open.  
**Case:**  
8.1 The end user interacts with a module on the GUI that allows for changing of sorting parameters.  
8.2 If the end user wants to sort by last name, or ZIP, they can select the corresponding parameter from the module.  
**Postconditions:**The contacts are sorted by the parameter specified by the user. All the contacts in the address book are displayed. Nothing about the actual file has been changed. If the user were to exit the application and re-open, the sorted list would not necessarily persist.

### Case 9: User Wants to Delete an Entry

**Players:** End User  
**Goals**: The user wants to remove an entry from the current active address book.  
**Preconditions:** The application is open and running with an address book open.  
**Case:**  
9.1 The end user selects an address entry.  
9.2 The user chooses to delete that entry.  
9.3 The user is prompted to confirm their choice to delete the entry.  
9.4 The entry no longer appears in the address book.  
**Exception Flows:**  
9.3.1 The user chooses "cancel" when they are prompted to confirm the delete.  
9.3.2 The contact remains in the address book.  
**Postconditions:**The selected entry is no longer listed in the contacts list.

### Case 10: User Wants to Add New Entry

**Players:** End User  
**Goals:**The user would like to add a new contact to their list of contacts in this particular address book.  
**Preconditions:**The application is open and running with an address book open.  
**Case**:  
10.1 The end user chooses to add a new entry to their address book  
10.2 The end user is presented with a form to fill and submit for the new entry.  
10.3 Upon successful completion, the entry is added to the list of the end user's contacts  
**Alternate Flows:**  
10.2 The user decides to "cancel" the workflow  
         The application returns to its initial state   
**Exception Flows:**  
10.3.1 The data provided by the end user is inconsistent with U.S. postal address standards or incomplete.  
           The application warns the end user about why their data has not been accepted.  
            The end user is then able to change the data they provided and resubmit.  
10.3.2 If there is another entry with the same information (e.g. first name, last name, address), the application creates a duplicate entry.  
**Postconditions:**There is a new contact in the contact list. It is marked to be saved at the next save point. The address book is aware that it has been altered.

### Case 11: User Wants to Edit an Entry

**Players:**End User  
**Goals:**The user would like to update information about a particular contact in a particular address book.  
**Preconditions**: The application is open and running with an address book open.  
**Case**:  
11.1 The end user selects an address entry  
11.2 The user to decide to edit that entry, moving it into "edit mode"  
11.3 The user makes edits  
11.4 The user saves those edits  
**Alternate Flows:**  
11.4.1 The end user tries to change the entry's information in a way that makes it noncompliant with U.S. postal address standards or incomplete.  
           When the end user tries to save the changes to this entry, they are warned about why the changes have been rejected.  
           The end user is then able to change the data they provided and resubmit.  
**Postconditions:**When the user views the edited contact, they can see their changes (if they made any). The address book is aware that is has been altered.

### Case 12: User Wants to Export Contacts To A TSV File

**Players:**End User  
**Goals:**The user would like to export all of their contacts to a TSV file.  
**Preconditions**: The application is open and running with an address book open where the book has at least one contact.  
**Case**:  
11.1 The end user selects to export their contacts  
11.2 The user is given the option to visually navigate their system to choose where they will save their TSV file  
11.3 The user provides the name for the TSV file  
11.4 The user saves the TSV file with the given name in the given location  
**Alternate Flows:**  
11.4.1 The end user chooses to "cancel" the process  
           The application returns to its initial state  
**Exception Flows:**  
11.4.2 The end user tries to give the file an invalid file name  
           The user is asked to provide a valid file name  
           The flow resumes with step 11.4  
**Postconditions:** There is a file on the user's system <filename>.tsv. The user's address book has not been changed in any way. The application is in its initial state.

### Case 13: User Wants to Import Contacts From A TSV File

**Players:**End User  
**Goals:**The user would like to add contacts to their address book using a TSV file  
**Preconditions**: The application is open and running with an address book open. The file must be in the specified format.  
**Case**:  
11.1 The end user selects to import contacts  
11.2 The user visually navigates their system to find where the file is saved  
11.3 The user chooses to import from that file  
11.4 All of the contacts are loaded into the user's address book  
**Alternate Flows:**  
11.3.1 The user chooses to "cancel" the process  
            The application returns to its initial state  
**Exception Flows:**  
11.4.1 One or more of the contacts do not fulfill the minimum requirements to create a new contact.  
           After import, the user is informed of how many contacts failed and the reason why.  
**Postcondition:**The user can find the new contacts that successfully imported in their address book. The address book is aware that is has been altered. The .tsv file is unchanged.

### Case 14: User Wants to Search for Contact(s)

**Players:**End User  
**Goals:**The user would like to search through their contacts for any instances of specific criteria.  
**Preconditions**: The application is open and running with an address book open.  
**Case**:  
11.1 The end user enters the criterion/criteria that they wish to search for.  
11.2 The user initiates the search  
11.3 The system finds all contacts that contain any instance of the search criteria/criterion  
11.4 The contacts that match are shown to the user.  
**Alternate Flows:**  
11.3.1 The system finds no contacts that contain any instance of the search criteria/criterion  
            The system waits for the user to verify that their search has returned no results.  
            The application returns to its initial state  
**Exception Flows:**  
11.4.1 The user decides to "cancel" the search process  
            The application returns to its initial state.  
**Postconditions:** The search result is visible to the user.

### Case 15: User Wants to Rename a Address Book

**Players:**End User  
**Goals:**The user would like to change the name of an address book.  
**Preconditions**: The application is open and running with an address book open.  
**Case**:  
11.1 The end user selects to rename the address book they are currently opened  
11.2 The user enters a new name  
11.3 The system updates the address book with name given by user  
**Alternate Flows:**  
11.2.1 The user chooses to "cancel" the process  
            The system will keep the original name of current address book  
            The application returns to its initial state  
**Postconditions:** The address book will be updated with a new name that is visible to the user.

## 3.    Behavioral Requirements

### 3.1 System Inputs and Outputs

#### 3.1.1 Inputs

Inputs to the application come from the user. The user can select the following options from the "File" menu: "New, "Open", "Save", "Save As...", "Import", "Export", "Close", and "Close all and Quit". Within an address book they have the ability to: "Add an Entry", "Delete an Entry", "Edit an Entry", "Rename Address Book", "Sort", and "Search" by interacting with buttons and other fields in the GUI. For a particular entry in an address book the user gives input by filing in provided text boxes.  
  
   Inputs When Adding/Editing An Entry:  
      \*General: At least two inputs will be required for a new contact. One of those two inputs must be a name. There is no length limit on any of the fields below.  
      \* Name: A contact can have a first name, or last name, or both. The first and last names will be separate fields. The system will accept any characters in the first and last name fields.  
      \* Street Address: Addresses will be split into **four** subsections: Delivery Address, Address Line 2, City, and State. This type of input will be checked against USPS Address Standards, but will not be checked against actual addresses. Address Line 2 has no input restrictions but City and State will be checked for alphabetical characters. If non-alphabetical characters are found, it is assumed that the City and/or State are not adhering to the USPS Address Standards. Similarly, the delivery address will be checked to see if the house number, street name(s), and street type contains only alphanumerical characters. If non-alphanumerical characters are encountered in the delivery address, it is assumed that the delivery address does not adhere to USPS Address Standards. The only exception is the hyphen character; this character is considered valid in address inputs.  
      \* Zip Codes: Zip codes that are accepted as valid by the system can be 5 or 9 digits long. The 9 digit zip code can have a hyphen. The only acceptable inputs for a valid input will be numerical characters ranging 0-9 and in the aformentioned case, a hypen. If characters outside of this range are encountered, it is assumed that the zip code violates USPS Address Standard and the user is warned of their non-standard input but allowed to continue if they want to. Non-alphanumeric characters with the exception of hyphens are not allowed as input.  
      \* Telephone Numbers: Only U.S. Phone numbers will be considered valid. The only acceptable inputs will be numerical characters ranging 0-9 and the special characters '(', ')', and '-'.  
      \* Email Address: Email addresses will be a single field. Acceptable inputs will be alphanumeric and must include '@' and '.' symbols. If the system does not detect '@' and '.', it is assumed that the email address is invalid.  
  
Importing:  
       \* A file must be in a very specific format for importing. Details about the format specifications can be found in the [Restrictions](https://uocis.assembla.com/spaces/cis422w18-team2/wiki/Software_Documentation#tsvFile) section of the User Guide. The file must be of .tsv extension type.  
  
Address book names will not be subject to restrictions of length or character type. The address book name will not be required to match the filename that the address book is saved to/opened from.  
Searches will not be restricted as far as input is concerned.

#### 3.1.2 Outputs

A GUI displaying all open address books. Each address book contains any created contacts and interactable buttons for the user.  
   Outputs to The User:  
      Editing/Adding Contacts:  
      \* If the system receives incorrect/invalid input from the user at the point of editing a contact or creating a new contact, the system will warn the user that their input has been flagged and the user will either be able to move forward with their process or enter valid data. The user will need to let the system know when the input has been changed by re-clicking the confirm button.  
  
      Viewing Contacts:  
      \* If our system flags a contact's field as invalid, that field will be highlighted in yellow when viewing the contact. This yellow highlighting will persist until the user changes the field to something the system views as "valid".  
       
      Saving Address Books:  
      \* The user will be warned if they are about to lose changes to their address book(s) and be allowed to save before exiting.  
  
      Searching:  
      \* If an address book does not contain what the user is searching for, then the resulting list of contacts will be blank.  
  
     Importing:  
       \* If any of the contacts in an import file don't fulfill the minimum requirements to create a new contact, the user will be informed of how many contacts failed after the import has finished.  
  
      Sorting:  
      \* Contact lists in an address book will be sortable by last name or by zip code. Ties will be broken in the following fashion:  
          - Sorted by last name: Initial ties broken by first name, secondary ties broken by zip code.  
          - Sorted by zip code: Initial ties broken by last name, secondary ties broken by first name.  
      \* Any contacts that do not have any input in a given search field will be sorted to the end of the contact list. Ties will be broken normally within these contacts.

### 3.2 Detailed Output Behavior

The output is a GUI that provides the user with the ability to give inputs to the stored address book. "Open" creates a new window with an empty address book. "Save" takes changes made by the inputs from the user and stores them in the address book file, the inputs from the user that change an address book are immediately reflected in the GUI. "Save As..." has the same functionality as "Save" but it also prompts the user to give a name for the save file and a location on their machine. "Import" and "Export" options allow the user to import a .tsv into their address book or save their current address book as a .tsv. The "Close" option closes the current address book, while "Close and Quit" closes all open address books and quits the program.  
  "Add a Contact" brings up a window for the user to provide more inputs in the form of name, phone number, and zip code. Once they have provided the necessary information the new contact appears in their address book. "Delete an Entry" removes the entry from the the display, it is not removed from the address book file until the user saves. "Edit an Entry" brings up a window that allows the user to change their previous inputs, changes are reflected immediately in the GUI but aren't permanent until a save occurs. "Sort" allows the user to choose between "Sort by Name" and "Sort by Zip" which changes the order in which the contacts appear in the address book. "Searching" contacts by keyword or phrase is also supported.  
  The application contains the base system requirements supplied by the user and the ability to search and have multiple address books open at once. These features were chosen over other advanced features (like the original plan of supporting an "Open Recent") because they provided the most user friendly experience. Specifically for the "Open Recent" option, it was a lower priority feature (that could be addded in the future if the project was extended) because the application automatically opens the previous open address book on launch.

### 4.2   Quality Requirements

The application must be competitive with similar applications in regards to performance, reliability, consistency, and scalability.  
  
Performance: Responsiveness to user input  
      \* Standard actions that manipulate address books or their contents should not exceed 500ms execution time.  
      \* Searching on address books is a possible exception to the above standard, as search performance will depend on the size of the address book.  
Reliability: Confidence that actions taken will not result in errors, and that changes made to address books are persistent  
      \* User input should not produce faults or errors that impact or hinder use of the application  
      \* Any modifications to an address book should produce a lasting change that persists through any following series of actions taken by the user.  
Consistency: Persistent data in address book contents  
      \* Address book contents should be modifiable after being added in such a way that target fields can be changed without affecting data in other fields.  
Scalability: Ease of extending application capabilities  
      \* Application should be modularized such that adding/extending features and functions only require changes to a single component and the interface with that component, if applicable.  
  
Furthermore, the application should adhere to U.S. Postal standards, validating input against the appropriate specifications. Input that deviates from the U.S postal standards will not be rejected, but the user will be notified that the input deviates from standards.

## 5.    Expected Subsets

L0:  
-Basic GUI with the ability to send data to and recall data from the backend storage  
-Ability to add address to address book  
-Ability to view an entry in an address book  
  
L1:  
-Address book capable of: storing complete entries, sorting, editting entries, and saving  
-Complete GUI for access to address book  
-Ability to import and export address books  
-Open recent

## 6.    Fundamental Assumptions

The application can run on any system that is capable of running Java Version 8.  
The application will not terminate when all windows are closed.  
Software updates will be downloaded by the end user as opposed to pushed out by the developers.

## 7.    Expected Changes

   Features to Add:  
      Import and Exporting Sets of Addresses  
      User Defined Fields  
      Links to Social Media  
      Deletion of Many Contacts at Once  
      Importing a Photo for Each Contact  
      View Contacts By Photo  
      Search Contacts By Photo  
      Voice-To-Text Searching  
      Ability to "Turn Off" Yellow Highlighting for Invalid Fields

   Future Platforms:  
      Write Front End For Andriod  
      Port Application to iOS

## 

## 8.    Appendices

### 8.1    Definitions and acronyms

#### 8.1.1    Definitions

|  |  |
| --- | --- |
| **Keyword** | **Definitions** |
|  |  |
|  |  |
|  |  |
|  |  |

#### 8.1.2    Acronyms and abbreviations

|  |  |
| --- | --- |
| **Acronym or**  **Abbreviation** | **Definitions** |
| GUI | Graphical User Interface |
|  |  |
|  |  |

### 8.2    References

Comments are disabled for this space. In order to enable comments, Messages tool must be added to project.

You can add Messages tool from Tools section on the Admin tab.