SOFTWARE REQUIREMENTS ANALYSIS
PROJECT 1 : ADDRESS BOOK

Edited by

Frazer Bayley
Haley Whitman
Abdulaziz Al-Heidous
Alison Legge
Jeremy Brennan

Project 1 - Team 3

Contents

Revision History

Revision	Date	Author(s)	Description
.1	17.01.17	Haley Whitman	Created initial outline of document
.2	18.01.17	Haley Whitman	Working through document, sections 1 to 2
.2	22.01.17	Haley Whitman	Finished ConOps
.2	23.01.17	Haley Whitman	Established the sections 3-8

1 Introduction

1.1 Intended Audience

The following document covers aspects and functionality of visual interface for an Address Book that holds a collection of user entries which are comprised up of names, addresses, cities, states, ZIP codes, phone numbers, and email addresses. This document is intended for all stakeholders, which are listed below:

- **Customer (Priority 1):**
  The customer has given an initial list of requirements, which includes a minimum section of product functionality, the quality that this functionality should have, and a list of advanced possible features. Included in these requests are objectives that should be focused on throughout the project. This stakeholder should use this document to assert their own product visions and requests with the team's understanding of these requests.

- **Coding Oriented Group Members (Priority 2):**
  Coding oriented group members should use this document in conjunction with the Software Design Specification (SDS) to understand the needed functionality of the product, as well as to get a more in depth understanding of why the SDS is created and framed in a certain way.

- **Documentation Oriented Group Members (Priority 2):**
  Documentation oriented group members should use this document to help the coding oriented group members in creating a SDS, to ensure that the proper requirements are encapsulated correctly by how the SRS defines them.

1.2 How to Use this Document

This document is intended to serve as a reference for the list of requirements created by the customer stakeholder. This document has two uses:

- **Assertion:** To ensure that the list of requirements detailed here meet with the project specifications given by the customer.

- **Identifying User Activities:** To detail out user scenarios and activities to best understand how this application will be used and designed later in the SDS document.

The rest of this document is broken up into the following subsections.

- **2. Concept of Operations:** Will detail the system requirements that are expected from a user's point of view. On top of this, section 3 will provide several use case models.

- **3. Behavioral Requirements:** This will be "black-box" view of the system, provided high-level descriptions of possible user inputs and expected outputs from the system. This section will not describe code-level descriptions of inputs and outputs.

- **4. Quality Requirements:** Describes the additional higher-level system requirements from the product specifications such as performance, reliability, and maintainability.

- **5. Expected Subsets:** Describes the project increments, beginning with the minimal working product to the final version and all product increments in between.

- **6. Fundamental Assumptions:** Describes all product requirements and constraints that we do not expect to change during the lifetime of the product.

- **7. Appendices:** Describes any local definitions and acronyms/abbreviations used in this project.

2 Concept of Operations

2.1 System Context

The backend of the address book application will be stored in different TSV files that will be used to store address book data. Code is written in Java.
Broadly, the address book is capable of:

- Importing and Exporting Multiple address books.
- Add, edit, and delete contacts into an address book file.
- Sort, view, and search contacts.
- Hold information such as first and last name, ZIP code, state, phone number, and email address.

2.2 System Capabilities

1. Brief Description
This use case describes how the user loads exports an address book.
2. Actors
2.1 User
2.2 Application
3. Preconditions
The user has a mouse, keyboard, and the Address Book application open and visible on the display, with an address book loaded that the user wishes to export. There is enough memory on the computer to handle the size of a typical address book.
4. Basic Flow of Events
1. The Address book displays the contacts in the address book. **5. Alternative Flows**
5.1 User doesn't find the correct address book they wish to load
The user can change the directory to continue searching for an address book.
5.2 User loads incorrect address book
If in Step 5 the user realizes they loaded the incorrect address book, they can go back to Step 2.
6. Post Condition
An address book file is created and the application displays a screen to view all of the contacts.

1. Brief Description
This use case describes how the user loads an address book.
2. Actors
2.1 User
2.2 Application
3. Preconditions
The user has a mouse, keyboard, and the Address Book application open and visible on the display, with the address book application just opened. There is enough memory on the computer to handle the size of a typical address book.
4. Basic Flow of Events
1. The Address Book displays the options to "Open an Address Book" and "Create an Address Book"
2. User selects to "Open an Address Book".
3. The Address Book will display the file directory where address books are saved.
4. User selects the Address Book file they wish to load.
5. Address book displays displays the contacts from the new address book.
5. Alternative Flows
5.1 User doesn't find the correct address book they wish to load
The user can change the directory to continue searching for an address book.
5.2 User loads incorrect address book
If in Step 5 the user realizes they loaded the incorrect address book, they can go back to Step 2.
6. Post Condition
An address book file is loaded, and the application displays the list of contacts.

1. Brief Description
This use case describes how the user saves an address book.
2. Actors
2.1 User
2.2 Application
3. Preconditions
The user has a mouse, keyboard, and the Address Book application open and visible on the display, with the address book application opened and changes made by the user. There is enough memory on the computer to handle the size of a typical address book.
4. Basic Flow of Events
1. The Address Book displays the options to "New", "Open", "Save", "Save as", "Close", and "Quit".
2. The user selects to "Save as".
3. Address Book will prompt for a name and file directory to save the file in.
4. The user inputs name and directory information.
5. Address book displays confirmation for saving address book.
5. Alternative Flows
5.1 User does not wish to save as a new file

If in step 3 the user does not wish to save anymore, the user can decide to select to exit. **6. Post Condition**
An address book is saved, and a file is left in the chosen file directory with all of the information that the user saved.

1. Brief Description
This use case describes how the user searches for a contact.
2. Actors
2.1 User
2.2 Application
3. Preconditions
The user has a mouse, keyboard, and the Address Book application open and visible on the display, with an address book loaded. There is enough memory on the computer to handle the size of a typical address book.
4. Basic Flow of Events
1. The Address Book displays the contacts.
2. User selects "Search for User".
3. Address Book prompts for a User entry of either a name, or zip code.
4. User enters the name they wish to search.
5. Address book displays searched name.
5. Alternative Flows
5.1 User searches for wrong contact
If in step 5 the contact the user wanted does not appear, the user can continue back at Step 2.
5.2 Searched user does not exist
If in step 5 the user does not find the searched for contact, they can continue back at Step 2 to try a different field.
6. Post Condition
The application displays the searched for user, giving options to edit and delete said user.

1. Brief Description
This use case describes how the user deletes an address in the address book.
2. Actors
2.1 User
2.2 Application
3. Preconditions
The user has a mouse, keyboard, and the Address Book application open and visible on the display, with an address book loaded. There is enough memory on the computer to handle the size of a typical address book.
4. Basic Flow of Events
1. The Address Book displays the contacts.
2. User selects contact.
3. Address Book prompts for "edit", "view", and "delete".
4. User selects delete.
5. Address Book prompts for confirmation.
6. User confirms.
5. Alternative Flows
5.1 User searches for contact.
If in step 2, the user searches for the contact, preform *Use Case: Contact Search* and then continue to step 3.
5.2 User Does not want to confirm deletion
If in step 5 the user does not wish to delete anymore, user selects to cancel and is brought back to step 1.
6. Post Condition
The deleted user's information is no longer in the address book file upon saving, but can be allowed to return upon the user's request.

1. Brief Description
This use case describes how a user interacts with the Address Book to add and edit a new entry of another person.
2. Actors
2.1 User
2.2 Application
3. Preconditions
The user has a mouse, keyboard, and the Address Book application open and visible on the display. There is enough memory on the computer to handle the size of a typical address book.
4. Basic Flow of Events
1. The use case begins when a user prepares to enter an address at the main screen of the Address Book
2. The Address Book presents several options representing different modes, such as "Create a new Contact," "Delete a Contact," "Import an Address Book," and "Export an Address Book."
3. The user selects to "Create a new Contact."
4. The Address Book presents the user to fill out information for the new contact.
5. The user fills out the name, address, city, state, zip, phone number, and email address.
6. Once filled out, the Address Book saves this information in the current address file.
7. The user is presented with the main menu mode of the Address Book application.
5. Alternative Flows
5.1 User not supplying all the information
If in step 5 the user does not fill out all of the information, then:
1. The system checks if the first/last name are filled out, if yes, continue to step 6.
2. If the first and last name are not filled out, stay at step 5 until completed.

5.2 User entered an address that conflicts with another address
If in step 6 the Address Book finds another address is already entered with the same information, then:
1. The application displays "Current Address is already found in Address Book."
2. The use case resumes at step 2.

6. Post Condition

3 Behavioral Requirements

3.1 System Inputs and Outputs

3.1.1 Inputs

- **Import/Export Buttons:** User clickable, and brings up a file directory.
- **Close:** Either (1) if in edit contact, will close the pop-up menu for the text field entries, or (2) if viewing the entire book, will close the currently open book.
- **Quit:** Only available if viewing contacts, this will close the program.
- **New:** Only available if viewing contacts, will create a new address book. This will prompt to save the currently loaded address book.

- **Open:** Only available if viewing contacts, will display a file directory of a list of currently available address book files.
- **Save:** Will save the current address book file.
- **Save As:** Will save the current address book file, with the option of saving it as a new separate file.
- **Add:** Will create the pop up to allow new entries in the address book file.
- **Edit:** Once a contact is selected, this will allow to change any fields in the contact.
- **Delete:** Will delete the currently selected contact.
- **Text Fields:** Will allow the user to click or tab through entries, and fill them out with alphanumeric letters.
- **Sort By:** Will allow the user to choose sorting the contact list by name or phone number.

3.1.2 Outputs

- **Display Contacts:** Address book returns a list of contacts that are found the TSV file.
- **Display Editing Contact:** A pop-up menu, with text fields to edit Names, phone numbers, email addresses, ZIP codes, State, and street name for the selected contact.
- **Display File Directory:** Directed by the operating system, can be used to manage what field is loaded, or the exported file's name.

4 Quality Requirements

- **Performance**
  - The address book needs to load and export address book files quickly, and respond to user requests such as sorting and adding new entries in less than 500ms.
- **Reliability**
  - Application should ensure that the user not unknowingly delete any data.
  - File locations should be saved, so the user does not have to search for where they are saved.
- **Standards**
  - The address book will require a certain amount of information, or will not accept an entry into the complete address book file.
- **Consistency**
  - The address book needs to be able to change over time, allowing for deletion, and editing, of any contact.
- **Ease of Change**
  - The code itself should be modular and follow good object oriented coding practices. This will ensure that future developments can be easily adapted to the code.

5 Expected Subsets

- **Subset 1:**
  - Implement basic interaction with a GUI. Needs to be able to save to an external backend file.
  - Add an address to the address book.
  - Retrieve/view an address.
- **Subset 2:**
  - All previous subset requirements, plus:
  - Store standard postal address and phone number.
  - Sort addresses by name or zip.
  - Edit, save, and recall addresses.
  - Separate GUI to enter address information.
- **Subset 3:**
  - All previous subset requirements, plus:
  - Import/Export standard TSV files of addresses.
  - Ability to search the address book.

6 Fundamental Assumptions

We expect that the user is loading this Java application on a Windows 8+ or OSX computer. The user needs a keyboard, mouse, and display as well to properly use the computer. This program is compiled with JDK 8.

7 Expected Changes

There are several requirements that may change after meeting with the customer after every iteration. Possibilities include:

- The schedule of deliverables before the final deliverable date. This may change based on the group figuring out their working schedule.
- User Interface design may change, to allow for changes based on the customer's preferences. Most likely these changes will be due to how information gets saved in the address book, and the functionality behind prompting the user for saving.
- Formatting of the TSV file has been unspecified, and will likely be due to change as per customer's request.

8 Appendices

8.1 Definitions and Acronyms

8.1.1 Definitions

8.1.2 Acronyms and Abbreviations

GUI	Graphical User Interface
SDS	Software Design Specification
SRS	Software Requirement Specification
TSV	Tab-Separated-Values