CS2043 Project - Requirements Documentation

Alec Sobeck
Kieran Lea

Thursday, May 22, 2014

Table of Contents

Introduction

Purpose

The purpose of this design document is to describe the functionality, features, and requirements of our CS2043 project which will feature a personal data store for billboards.

Scope

This project will be used to store and manipulation information about billboards for a single user. The user will have access to a simple graphical interface to add, remove, modify, and search through their billboards, while the more complex operations such as reading and writing xml will be handled by the application logic.

Definitions, acronyms, and abbreviations

Billboard	"A large sign for advertisements that is next to a road, on the side of a building, etc" - Merriam-Webster dictionary.
User	The person that will be using the software to manage their billboard collection.
GUI	Stands for "graphical user interface". The graphical user interface is a window that the user can use to interact with the program.
XML	Shorthand for "EXtensible Markup Language" - a file format that is used to store data. It is highly descriptive and can be read in a text editor.
JFileChooser	A Java GUI component that allows the user to select a directory and file name based on their operating system.
Java StaX API	A Java API that allows the reading and writing of XML files.
API	Short for "Application Programming Interface", which is a software component that must be interfaced with in some specific way.

References

1. Refers to the *CS2043 Team Project Requirements* document for format and content requirements

Overview

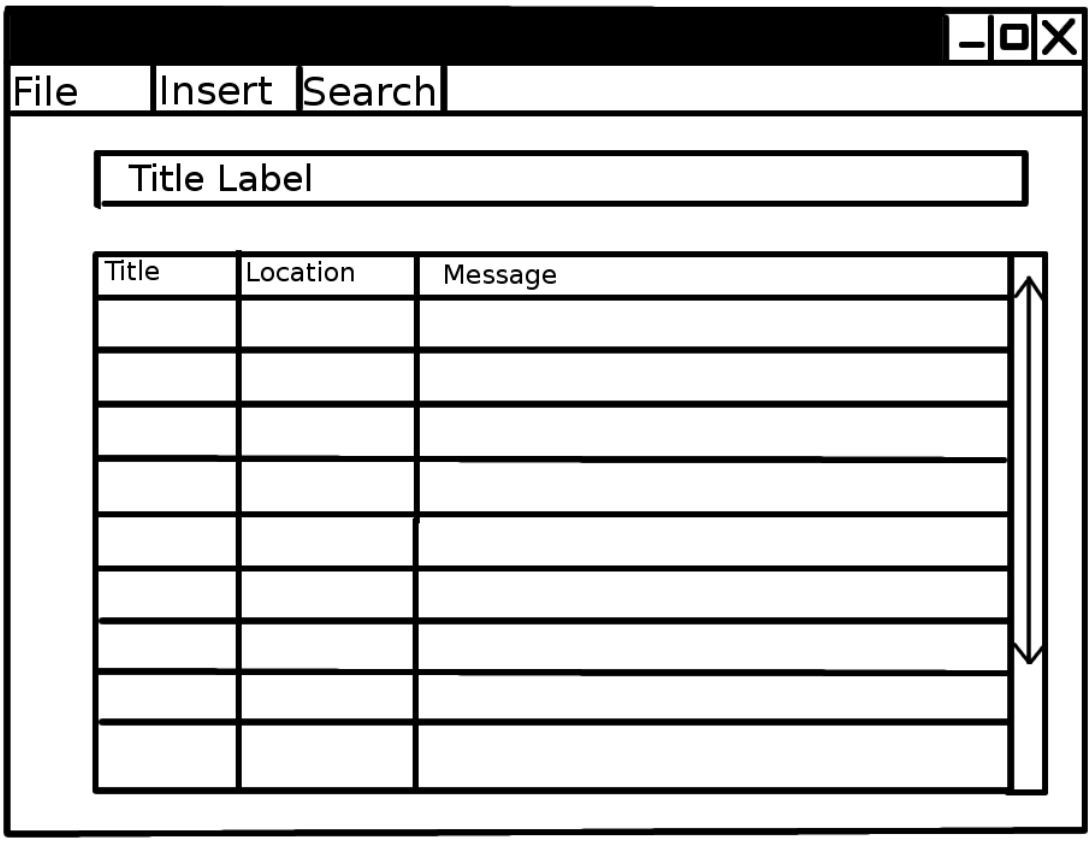
This document is structured in accordance with a simplified version of *IEEE Standard 830-1998: Recommended Practice for Software Requirements Specification,* as provided in the *CS2043 Team Project Requirements* document. Section one of the document contains an introduction to the software, while section two details technical information such as the requirements, constraints, and non-functional requirements. Section three contains use cases for the different software functions.

Overall description

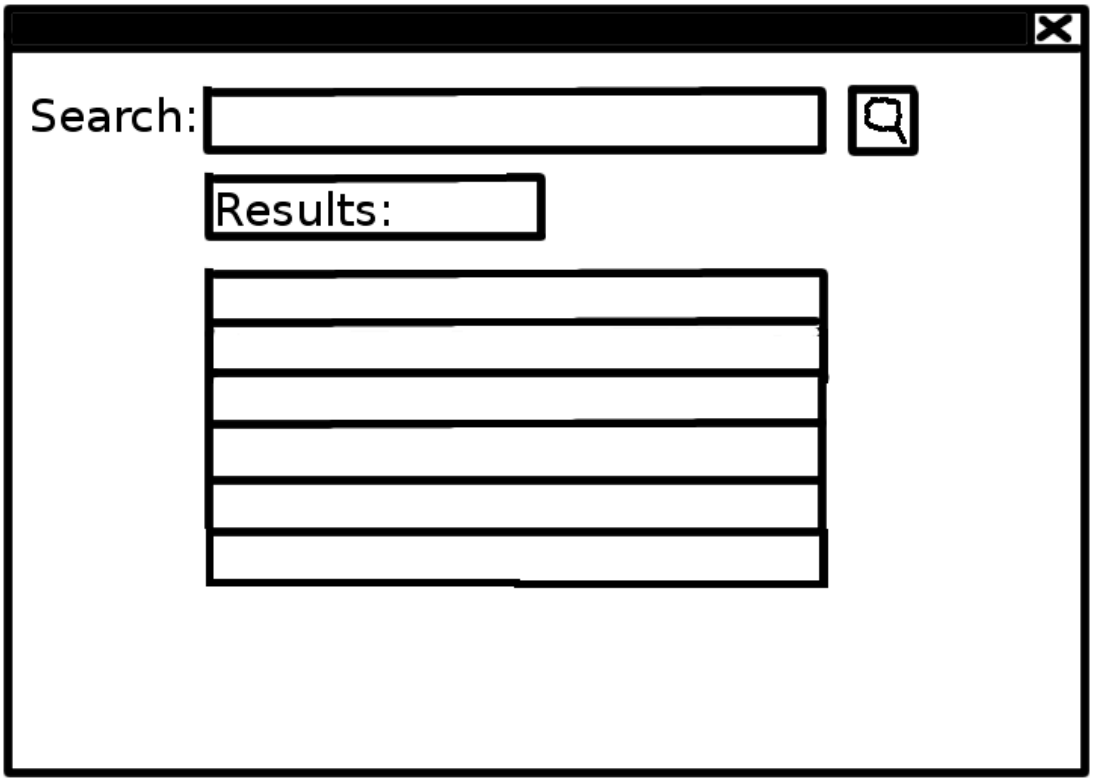
Product perspective

- System and Hardware Interfaces:
  - Makes use of functionality in the Operating System to read and write files to the hard drive.
- User Interfaces:
  - The product will make use of three graphical interfaces. One to view and manipulate the data, one to search for a billboard and one to add a new billboard. Black and white approximations of the user interfaces are included below.
  - The application will make use of a basic menu bar at the top of the window to allow the user to perform operations such as: add an item, remove and item, and search for an item, start a new file, open an existing file, save a file, and save a file to a specific location.

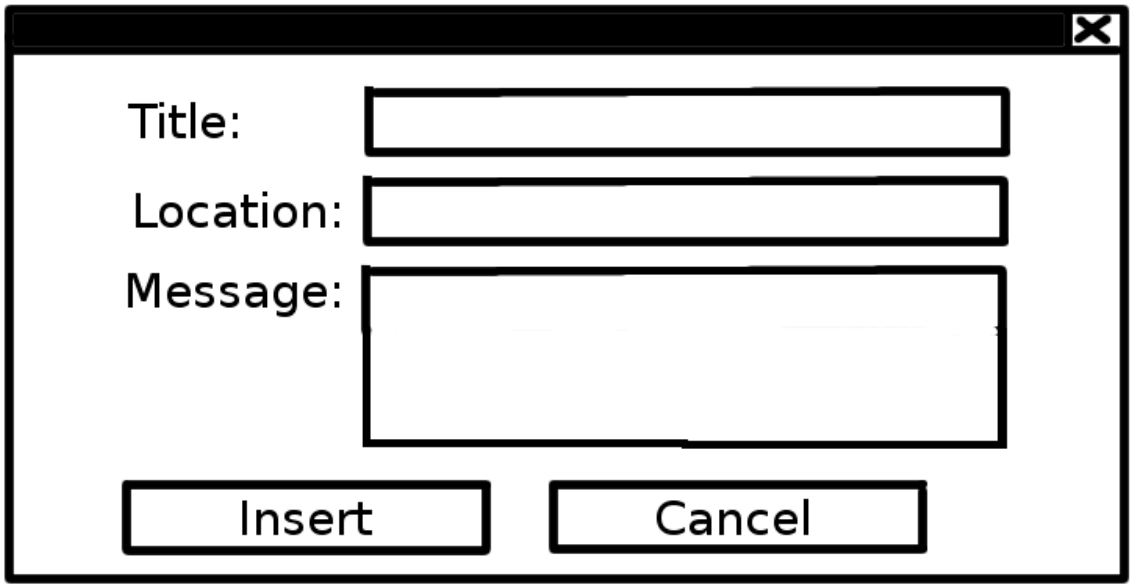
A black and white image approximating the look of the main window's GUI:



A black and white image approximating the look of the search window’s GUI:



A black and white image approximating the look of the insert window’s GUI:



Product Functional Requirements

- Supports a personal data store of billboards that is in XML format.
- Has a functional graphical user interface
- Written fully in Java 1.7
- Makes use of the Java StaX API to read and write XML
- Creates a new XML data file
- Read an existing XML data file

- Add a new element to the XML data file
- Remove an element from the XML data file
- Modify an existing element from the XML data file
- Search for an item in the XML data file

Design Constraints and Non-functional requirements

Non-Functional Requirements:

- File I/O completes in less than 10 seconds 95% of the time.
- Adding a new element, removing an existing element, and modifying an element complete in less than 3 seconds 95% of the time.
- Search completes in 6 less than seconds 95% of the time
- Billboard information may be searched for using either the title or location

Constraints:

- Makes use of the Java StaX API for reading and writing XML
- Programmed entirely in Java Version 1.7
- Must have a graphical user interface
- Platforms: Supports any versions of Windows, Mac, Linux, and Solaris that have the Java Virtual Machine version 1.7+ installed
- Portability: The software will function on any system that has the Java Virtual Machine version 1.7+ installed with no additional changes to the code. This is because of Java's compile once and execute anywhere ideology, and the fact that file paths will be handled with a JFileChooser and not hard coded at all.

User characteristics

- The user speaks English at least moderately well, such that they can understand and make use of an English GUI.
- Minimal technical experience. Users are however expected to have a basic understanding of how their operating system's file path works so they can navigate the JFileChooser.
- Age 15-75
- Male or Female
- Some idea about what a billboard is, in order to accurately enter or modify the information stored in the system

Specific requirements

Use Cases

Use case: New Billboard list (in memory)
Actor: Program user
Precondition: The software is fully initialized.

(a) Main Case:

Case 1: Successfully create a new billboard list in memory
User: Selects the File-New menu item
System: Creates a new billboard list in memory
Postcondition: A new and empty billboard list is present in memory.

Use case: Insert New Billboard
Actor: Program user
Precondition: The software is fully initialized and a file is loaded.

(a) Main Case:

Case 1: Successfully insert a new billboard
System: Displays the insert GUI which asks for billboard data
User: Enters in billboard data and clicks the ok button
System: Checks that the data fields are not empty
System: Checks that valid data is entered
System: Inserts the new element into the XML file in memory
System: Inserts a new row in the table GUI to reflect new data
Postcondition: An item is inserted to the XML file and table

(b) Alternate Cases:

(i) Billboard data not fully entered

System: Displays the insert GUI which asks for billboard data
User: Enters incomplete billboard data and clicks the ok button
System: Checks that the data fields are not empty
System: Checks that valid data is entered
System: Displays message that all fields must be filled in
Postcondition: Item is not inserted into XML file

(ii) Invalid Characters Entered

System: Displays the insert GUI which asks for billboard data
User: Enters billboard data with invalid characters and clicks ok
System: Checks that the cell is empty
System: Checks that valid data is entered
System: Displays message that characters are not valid
Postcondition: Item is not inserted into XML file

(iii) User declines to insert

System: Displays the insert GUI which asks for billboard data
User: Presses the decline button on the insert GUI

Postcondition: Item is not inserted into XML file

Use case: Remove Item
Actor: Program user
Precondition: The software is fully initialized and a file is loaded.

(a) Main Case:

Case 1: Successfully remove an item
User: User attempts to delete a row of the table
System: Checks if there are rows to delete
System: Deletes row
System: Removes the corresponding element of the XML file in memory
System: Refreshes table GUI to reflect new data
Postcondition: An item is removed from the XML file in memory

(b) Alternate Cases:

(i) Case: No rows to remove

User: Attempts to delete a row of the table
System: Checks if there are rows to delete
System: Displays message that there are no rows to remove
Postcondition: No item is removed from the XML file in memory

Use case: Modify Item
Actor: Program user
Precondition: The software is fully initialized and a file is loaded.

(a) Main Case:

Case 1: Successfully modify values from the data
User: Attempts to modify a cell.
System: Receives a notification that a cell has been altered
System: Checks that valid data is entered
System: Updates the XML file in memory with new data
System: Refreshes table GUI to reflect new data
Postcondition: The XML file in memory was successfully modified.

(b)Alternate Cases:

(i) Invalid Characters entered

User: Attempts to modify a cell by inputting invalid data
System: Checks that the cell has data to modify
System: Checks that valid data is entered
System: Displays message that characters are not valid
Postcondition: Item is not modified in the XML file in memory.

Use case: Search for data
Actor: Program user
Precondition: The software is fully initialized and a file is loaded.

(a) Main Case:

Case 1: Search for data and a unique entry is found
User: Attempts to search for an item
System: Takes search parameters and searches the XML file for matches
System: Finds matching information once
System: Displays matching data
Postcondition: Search completed successfully

(b) Alternate Cases:

(i) Duplicates found
User: Attempts to search for an item
System: Takes search parameters and searches the XML file for matches
System: Finds duplicate information
System: Displays the data of all duplicates
Postcondition: Search completed but with duplicates found

(ii) Information entered was not found
User: Attempts to search for an item
System: Takes search parameters and searches the XML file for matches
System: Finds no matches
System: Displays that no matches were found
Postcondition: Search completed but with no matching information

Use case: Save an XML File
Actor: Program user
Precondition: The software has started up and the GUI is fully initialized

(a) Main Case:

Case 1: Billboard list successfully resaves
Actor: selects File-Save
System: checks for a billboard list in memory to save
System: checks if the billboard list in memory has a resave location
System: finds the list's resave location on disk
System: writes the XML file successfully
System: issues a success message
Postcondition: An XML file is written to disk

(b) Alternative Cases:

(i) No billboard list in memory to save
Actor: selects File-Save
System: checks for a billboard list in memory to save
System: can't find a billboard list to save to disk
System: issues error message
Postcondition: No XML file is saved

(ii) File path not found
Actor: selects File-Save
System: checks for a billboard list in memory to save
System: checks if the billboard list in memory has a resave location

System: checks for the billboard lists resave location in the file system but determines that it doesn't exist
System: issues error message
Postcondition: no XML file is saved

(iii) Billboard list never saved before
Actor: selects File-Save
System: checks for a billboard list in memory to save
System: checks if the billboard list in memory has a resave location
System: determines resave location is null (there isn't one)
System: invokes the "Save As..." functionality to get a file path for the save
Postcondition: An XML file is written to disk and the resave location is stored in the billboard list

Use case: Save a billboard list to a particular directory with a specific name
Actor: Program user
Precondition: The software has started up and the GUI is fully initialized

(a) Main Case:

Case 1: Successfully save the billboard list
Actor: selects File-Save As...
System: opens a JFileChooser
Actor: selects a directory and file name
Actor: presses the accept button in the JFileChooser
System: retrieves the file path from the JFileChooser and saves the billboard list to an XML file there.
System: stores the location the billboard list was saved to (for File-Save functionality)
System: issues success message
Postcondition: the XML file is written to disk

(b) Alternative Cases:

(i) User declines to save
Actor: selects File-Save As...
System: opens a JFileChooser
Actor: presses the cancel button on the JFileChooser
Postcondition: no XML file is saved

(ii) Saving Fails
Actor: selects File-Save As...
System: opens a JFileChooser
Actor: selects a directory and file name
Actor: presses the accept button in the JFileChooser
System: determines that the selected directory and file name combination does not exist on disk.
System: issues error message

Postcondition: no XML file is written to disk

Use Case: Open an XML File
Actor: Program user
Precondition: The software has started up and the GUI is fully initialized

(a) Main Case:

Case 1: Successfully open a billboard list file

Actor: selects File-Open
System: opens a JFileChooser
Actor: selects a directory and file name
Actor: presses the accept button in JFileChooser
System: opens the specified file and reads in the billboard list
System: updates the GUI with the new billboard list
Postcondition: the GUI is populated with the data from the opened billboard list, and that billboard list is stored in memory.

(b) Alternative Cases:

(i) User declines to open a file

Actor: selects File-Open
System: opens a JFileChooser
Actor: presses the cancel button in the JFileChooser
Postcondition: no XML file is opened

(ii) User selects an invalid file format

Actor: selects File-Open
System: opens a JFileChooser
Actor: selects a file and directory using the JFileChooser
Actor: clicks the “accept” button in the JFileChooser
System: finds the file specified on disk
System: can’t read in the data because it has an invalid format
System: issues error message
Postcondition: no XML file is opened

(iii) File Not Found

Actor: selects File-Open
System: opens a JFileChooser
Actor: selects a file and directory using the JFileChooser
Actor: clicks the accept button in the JFileChooser
System: can’t find the specified file on disk
System: issues error message
Postcondition: no XML file is opened