Software Requirements Specification

for

Assisted Living Facilities Inspection Scheduling Project

Version 0.1

Prepared by Team 7:   
Kenneth White  
Colton Johannson  
Micalyn Williams  
Brandtly Strobeck

EWU CS Department

Fall 2018/Winter 2019

Table of Contents

Table of Contents ii

Revision History ii

1. Introduction 1

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Mission Statement 1

1.5 References 2

2. Overall Description 2

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 Operating Environment 2

2.4 Design and Implementation Constraints 3

2.5 User Documentation 3

2.6 Assumptions and Dependencies 3

3. External Interface Requirements 3

3.1 User Interfaces 3

3.2 Software Interfaces 3

4. System Features 4

4.1 Import Existing Data 4

4.2 Intuitive Data 4

4.3 Enter Inspection Results 5

4.4 Schedule a Follow Up Inspection Immediately 6

4.5 Schedule a General Inspection Immediately 6

4.6 Help Organize Inspection 7

4.7 Manually Override Scheduling 8

4.8 Export Data 8

5. Other Requirements 9

6. Appendix A: Glossary 9

Appendix B: Analysis Models 10

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

Product: Assisted Living Facility Inspection Scheduler Version 0.1

The purpose of this Software Requirement Specification (SRS) is to detail the product in its entirety; From requirements the software will incorporate, to the specifications we will use to implement the requirements, the validation tests that will be used to ensure each requirement is implemented, and the verification tests that specifications were followed.

## Document Conventions

There is no particular priority associated with the order features are listed, however higher level headings are the primary objective to which the nested headings contribute to or elaborate on.

In general the headings will be designated in bold as a period separated list in the following format as applicable: Section . Subsection . (Requirement/Specification/Validation)

## Intended Audience and Reading Suggestions

This document is primarily intended for use by the project developers, however Sections 1 and 2 are applicable for any reader as they detail the purpose and high-level details of this project.

**Users**: May specifically find Sections 2.2, 2.6 and Section 4 helpful in seeing a listing of what functions the product can perform. In particular, 2.6 details where to find the complete user documentation.

**Troubleshooting**: Section 2, Section 3 and Section 5 may assist in troubleshooting issues with the software.

**Developers:** Sections 1 and 2 will provide an overview of the project, Sections 3 and 4 will specify the features of the product.

## Mission Statement

Our mission is to make the scheduling of Assisted Living Facility inspections an advantage to Washington DSHS and not a burden. Our goal is to provide a flexible software solution to help organize and automate this work so that they may better focus on their own mission of support and safety.

The product’s primary purpose is to automatically generate inspection schedules for ALF facilities. The software is also intended provide an improved and intuitive interface over the currently used Excel spreadsheets, while maintaining the portability of the Excel file format.

## References

[1] ALF Full Inspection Operational Principles and Procedures

<https://www.dshs.wa.gov/altsa/residential-care-services/alf-full-inspection-operational-principles-and-procedures>

[2] .NET Framework system requirements

https://docs.microsoft.com/en-us/dotnet/framework/get-started/system-requirements

# Overall Description

## Product Perspective

This SRS specifies a new self-contained product that aims to improve upon the current use of Excel to track and schedule ALF inspections. It is a client based system wrapping Excel files with a MVC application for Windows. Once an ALF facility has been inspected, the results are entered into the application, which is responsible for scheduling any necessary follow up inspections as well as the next compliance inspection in approximately 13-15 months.

## Product Functions

The primary function of this application is to automate the scheduling of the ALF inspections; Thus the information entered by the user is used to prioritize and generate subsequent inspections for

the facility.

The application will allow users to read in data from an existing Excel file and perform any of the following functions:

* View the data in a concise manner
* Edit existing data, or add new entries
* Sort, search and filter the data
* Manually adjust inspection dates
* Export/print data on specific facilities

## Operating Environment

This software is targeted towards the Windows platform, supporting the following operating systems.

|  |  |
| --- | --- |
| Windows Version | Supported Editions |
|  |  |
| Windows 10 | 32-bit and 64-bit |
| Windows 8 | 32-bit and 64-bit |

Source: [2] MSDN .Net Framework System Requirements

## Design and Implementation Constraints

There are a number of limitations the software must accommodate. This software will not have access to the customer’s database, and so it is designed around importing and exporting the data to and from Excel files. For this reason, the software is built around using the .Net Framework’s Interoperability library. The data the software will work with is sensitive in nature, so as a security consideration this software will not have any internet or network connectivity.

## User Documentation

This will be completed after the software has been finished.

## Assumptions and Dependencies

There are a number of assumptions in regards to this project and its requirements that are subject to change as the project progresses.

* The application will only need to target Windows operating systems.
* The Interoperability library the software will use requires the client machines have a copy of Excel installed.

# External Interface Requirements

## User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

This will be completed at a future point when we have considered the GUI design and spoken more with the client.

## Software Interfaces

The application will rely upon Excel 2008 or later, and work with .xls files in order to maintain the current product. The C# Interoperabilty library relies upon an existing installation of Excel in order to manipulate .xls files. Built-in functions of the Excel application will be called through the C# library, such as adding, removing, sorting and modifying rows, cells, and columns.

In addition, as an optional feature we are considering adding support for storing or exporting the data to a SQL database, time permitting. If we do choose to implement such features, we will update this document with the specifications.

# System Features

## Import Existing Data

4.1.1 Description and Priority

The application needs to read in the existing data in the form of an Excel spreadsheet. This is High priority for this project as it has a high benefit for the users in making it easy to transition to this application.

4.1.2 Stimulus/Response Sequences

4.1.2.1: As a user I want to launch the application and select the existing Excel file on my computer to load into the application.

4.1.2.2: As a user I want the application to allow me to both save to a new file, or edit the existing file as needed.

4.1.3 Functional Requirements

4.1.3.1: The application will load the three most commonly used files for Excel: xls, .xlsx, and .csv.

4.1.3.2: The application must allow the user to preserve the original file without changes.

4.1.4 Requirement Specifications

4.1.4.1: The allowed file types will be .csv, .xls, or .xlsx.

4.1.4.2: The application will prompt the user asking whether the file they are opening should be edited, or only saved to a new file.

4.1.5 Validation

4.1.5.1: The application is able to load table data from .csv, .xls, and .xlsx files.

4.1.5.2: The save behavior of the application is consistent with the selection the user made.

## Intuitive Data

4.2.1 Description and Priority

The existing data must be organized and concise with the ability to expand into more details. This is High priority for this project as it has a high benefit for the users in making it easy to transition to this application as well as providing all necessary information they otherwise could not access.

4.2.2 Stimulus/Response Sequences

4.2.2.1: As a user I want to be able to edit the data associated with a facility; phone number, address, license number, etc.

4.2.2.2: As a user I want to be able to easily search and find a given facility.

4.2.2.3: As a user I want to be able to see a quick calendar of upcoming facility inspections.

4.2.3 Functional Requirements

4.2.3.1: The user needs to select fields to be displayed in a concise view.

4.2.3.2: Fields needs to be color coded by their inspection status.

4.2.3.3: We will allow editing of all fields, and prompt for confirmation for changing the code result of a previous inspection.

4.2.3.4: The fields must conform to their specified data type.

4.2.3.5: Can filter and search on properties from facility information and inspector information.

4.2.4 Requirement Specifications

4.2.4.1: The fields that will be displayed in the default concise view will be: Facility name, location, last inspection date, last inspection code, next inspection date.

4.2.4.2: Inspection dates that have passed without having the resulting code entered will be denoted in red. Inspections occurring within the next month will be denoted green.

4.2.4.3: Prompt will only display for results that were entered before the current application session.

4.2.4.4: Input will be validated and flagged if incorrect.

4.2.4.5: Searching and sorting will use efficient algorithms to remain responsive.

4.2.5 Validation

4.2.5.1: Fields in concise view can be selected from all available columns.

4.2.5.2: Test dates falling within all three ranges to ensure are highlighted correctly.

4.2.5.3: Prompt is displayed only for the inspection code field, and only if it was set in a previous session.

4.2.5.4: Test that fields are properly vetted for the appropriate data type.

4.2.5.5: Show that searching and sorting the document is quick and responsive.

## 4.3 Enter Inspection Results

4.3.1 Description and Priority

The application needs allow the user to enter the result of an inspection, this result will be used to schedule subsequent inspections. This is a high priority feature as much of the application’s functionality will revolve around these results.

4.3.2 Stimulus/Response Sequences

4.3.2.1: As a user I want a convenient way to enter the code representing the inspection results.

4.3.2.2: As a user I want to be able to enter comments and notes associated with both the inspection and the facility.

4.3.2.3: As a user I want to be able to be able to record other kinds of investigations associated with a facility, such as complaints.

4.3.3 Functional Requirements

4.3.3.1: The application must provide a list of codes representing inspection results that can be selected easily.

4.3.3.2: The application must provide a means to record notes and comments under the facility and inspection.

4.3.3.3: The user must be able to record other types of investigations associated with a particular facility.

4.3.4 Requirement Specifications

4.3.4.1: The application will provide a dropdown list of inspection codes.

4.3.4.2: The user can add comments for the facility, inspection, or inspector.

4.3.4.3: The user may also record investigations requiring telephone verification, document/letter verification, and on-site verification.

4.3.5 Validation

4.3.5.1: Demonstrate that results can be entered via dropdown menu.

4.3.5.2: Demonstrate the program can add comments for the facility, inspection, or inspector.

4.3.5.3: Test that the application can also record the all of the alternate investigation types.

## Schedule a Follow Up Inspection Immediately

4.4.1 Description and Priority

The application needs to be able to be given a result code from an inspection, then based on the code, schedule the next inspection automatically.

4.4.2 Stimulus/Response Sequences

4.4.2.1: As a user I want any necessary follow up inspections to be scheduled.

4.4.2.2: As a user I want to be able to denote which kind of revisit is required.

4.4.3 Functional Requirements

4.4.3.1: Investigation results must be a part of set list of investigation codes.

4.4.3.2: Revisit type needs to be determined by result code.

4.4.3.3: Follow-up inspections need to be scheduled accordingly to code, and will affect schedule of that facility

4.4.4 Requirement Specifications

4.4.4.1: Allowed inspection codes are: NO, ENF, YES, CSAME, CSS, manual input.

4.4.4.2: Revisit must be within 45-60 days from when the attestation of correction was made by the inspector.

4.4.4.3: After the facility passes a follow up, next inspection will be scheduled 12 – 18 months away.

4.4.5 Validation

4.4.5.1: Display list of investigation codes.

4.4.5.2: Show revisits are scheduled within 45-60 days.

4.4.5.2: Check that after a passed follow up, inspection will be scheduled 12 – 18 months away.

## Schedule a General Inspection Immediately

4.5.1 Description and Priority

The application must be able to interpret the result code from an inspection, then based on the code, schedule the next inspection automatically.

4.5.2 Stimulus/Response Sequences

4.5.2.1: As a user, when I enter the inspection results, I want the next inspection to be scheduled for that facility following the 15-month average.

4.5.3 Functional Requirements

4.5.3.1: Facilities will be inspected at least every 18 months with an average inspection interval of 13-15 months between inspections.

4.5.3.2: Inspections for previously problematic facilities should average 9-12 months, good facilities will average 16-18 months.

4.5.3.3: Inspections for facilities with 3 consecutive inspections with no complaints or citations may be inspected up to 24 months apart.

4.5.4.3: After the inspection, at least 6 working days must be allowed for post-inspection processing to occur.

4.5.4 Requirement Specifications

4.5.4.1: Default inspection will be between 13-15 months, and cannot be within 2 weeks of the date of the last inspection the year before.

4.5.4.2: If problems were reported in the previous two inspections, the next inspection will be scheduled between 9-12 months, and vice-versa for good facilities inspected between 16-18 months.

4.5.4.3: If a facility has no problems in the past 3 consecutive inspections, the next scheduled inspection will be between 22-24 months.

4.5.5.3: If a facility has no problems in the past 3 consecutive inspections, the next scheduled inspection will be between 22-24 months.

4.5.5 Validation

4.5.5.1: Verify that average inspection date is within 13-15 months.

4.5.5.2: Verify that problem free facilities are scheduled within range of 16-18 months, and that passing with issues facilities are scheduled within 9-12 months.

4.5.5.3: Verify that problem free facilities aren’t scheduled as frequently.

4.5.5.4: Verify that the post-inspection process is allotted 6 days before marking the results as past the due date.

## Help Organize Inspection

4.6.1 Description and Priority

Once all appropriate data is entered into the program, the user should be able to generate an inspection schedule with an appropriate number of inspectors.

4.6.2 Stimulus/Response Sequences

4.6.2.1: As a user I want the number of inspectors needed for inspection to be estimated using the facility’s bed count.

4.6.2.2: I want for a given inspection to be able to enter the inspectors who will/did do the inspection.

4.6.3 Functional Requirements

4.6.3.1: An applicable Excel sheet about each facility. This includes all of their past scores on visits, to know how often a visit will be schedules, and number of inspectors needed.

4.6.3.2: Inspectors need to be assigned an ID number, so that one of the original inspectors who reported a problem should be reschedules for the revisit whenever possible.

4.6.4 Requirement Specifications

4.6.4.1: The Excel needs to be in a standard format, so the application can read it.

4.6.4.2: Each inspector will have a unique inspector ID.

4.6.5 Validation

4.6.5.1: Come up with a standard Excel sheet format, fill it with mock data, and run it through our application.

4.6.5.2: Demonstrate that one of the original inspectors was assigned for the revisit.

## Manually Override Scheduling

4.7.1 Description and Priority

The application should be able to override any inspection, even if it is rescheduled at a time where there is another inspection scheduled.

4.7.2 Stimulus/Response Sequences

4.7.2.1: To remain prepared for the unexpected, I want to be able to manually adjust inspection dates if needed.

4.7.2.2: If policies change, I want to be able to adjust how revisits are scheduled, e.g. second revisits change from being within 90 days of first failed revisit to within 60.

4.7.3 Functional Requirements

4.7.3.1: Adjusting an inspection will not affect other inspection without alert of conflict.

4.7.3.2: The application needs to have preferences.

4.7.4 Requirement Specifications

4.7.4.1: The application will only affect other inspections if a manual override occurs.

4.7.4.2: The application will have global preferences, and case-level preferences.

4.7.5 Validation

4.7.5.1: Demonstrate an alert from a conflict.

4.7.5.2: Display preferences.

## Export Data

4.8.1 Description and Priority

The application will be able to let users save inspection results to back to original or new Excel files. The user will also be able to print other details such as, yearly schedule, data reports, facility information, and inspector information.

4.8.2 Stimulus/Response Sequences

4.8.2.1: As a user I want to be able to save the data back to Excel files.

4.8.2.2: As a user I want to be able to create and save data/inspection reports on particular facilities.

4.8.2.3: As a user I want to be able to print the data reports and the yearly schedule calendar overview.

4.8.2.4: As a user I want the file format to be standard across different regional offices.

4.8.3 Functional Requirements

4.8.3.1: Exported data needs to be saved into a new file with a set format.

4.8.3.2: The application needs to print all information related the specific module.

4.8.4 Requirement Specifications

4.8.4.1: The data will be saved in a tables, with only one per sheet.

4.8.4.2: The application will print out detail specific to a module such as facility, inspector, and inspection.

4.8.5 Validation

4.8.5.1: Verify that exported data is in correct format.

4.8.5.2: Demonstrate the application prints out all information related to the specific module.

# Other Requirements

None at the moment.

# Appendix A: Glossary

### ALF

Assisted Living Facility – An assisted living facility provides long-term assistance and personal support to seniors with services such as meals, cleaning and transportation.

### DSHS

Department of Social and Health Services – According to the Washington DSHS website: “The Department of Social and Health Services is Washington's largest state agency. In any given month, DSHS provides some type of shelter, care, protection and/or support to 2.4 million of our state's 7.1 million people.”

### SRS

Software Requirement Specification – refers to this document.

### GUI

Graphical User Interface – this refers to graphical elements in an application, such as windows, icons and buttons.

### MVC

Model-View-Controller - this is an architectural paradigm in software that is commonly used when developing user interfaces. The MVC model splits the application into three interconnected parts; The Model that represents the data of the application, the View that the user sees, and the Controller through which the user manipulates the model.

Appendix B: Analysis Models

Current analysis models not ready. We will add these as we consider the design of the architecture.