Software Requirements Specification

Version 1.0

February 11, 2020

Mount Vernon Department of Watershed Software Upgrade Project

Michaela Brydon

Shane Canfield

Sejin Kim

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to present a detailed description of the software solution that is being developed for the Mount Vernon Department of Watershed. It will explain the purpose, features, and interfaces of the system, the constraints under which it must operate, and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system.

1.2 Scope of the Project

The software system will be an archive access system for administrators of the Department of Watershed, and should require minimal training.

This system will be designed to allow engineers and staffers to search for and view records related to service calls, account information, addresses, account numbers, and other notes through a secure web portal. The system may also include functionality to automatically generate reports in a human-readable and accessible format. The records in the system are designed to be archival, and thus are read-only. Additionally, we will implement functionality to create and administer user accounts.

1.3 Glossary

Term	Definition
Active actors	Types of people interacting with the final software system
Active entry	The account displayed after a search is made and a record is requested
Cooperating System	The working system

Database	Collection of all information monitored by the system
Docker®	A containerization technology that allows specialized applications to be run inside of a dedicated environment, regardless of the host
Field	A cell within a form
Host	The server or computer that the software runs on
CMI® UtyX©	The existing records database system, to be retired
Entry	An account number along with the information associated with it
Software Requirements Specification	A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document
Stakeholder	Any person with an interest in the project who is not a developer
User	Any person using the system to view entries

1.4 References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

1.5 Overview of the Document

The Overall Description section of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification given later.

The Requirement Specification section is written primarily for the developers and describes, in technical terms, the details of the functionality of the product.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

2. Overall Description

2.1 System Environment

[insert system environment diagram here]

The archive access system has two active actors (User and Administrator) and one cooperating system. Both the Users and the Administrator may access the system over the secure intranet portal. Any User interactions must be carried out through this web portal. The Administrator may access information through the web portal but must make changes through the secure command line. The operations that they may choose to execute shall be provided in a documentation manual, along with all syntactical resources for command-line interface executions.

2.2 Functional Requirements Specification

This section outlines the use cases for each of the active Users separately. The system's main actor is the Administrator, while the standard User only has one use case. The specifications for each case are provided below:

2.2.1 User Use Case

Use case: Search record

[insert diagram]

[Brief Description]

The user accesses the web portal, logs in, searches for a record, selects a record, and views it.

Step-by-Step Description:

Before this use case can be initiated, the User has already accessed the web portal.

1. The User logs in using their legitimate username and password.
2. The system, by default, displays a search page.
3. The User chooses to search by address or account number.
4. The system presents all applicable matches given the provided keywords.
5. The User chooses to open a select account.
6. The system provides the requested account information.

Use Case: View Comments

[insert diagram]

[Brief Description]

The user accesses the web portal, logs in, searches for a record, selects a record, and chooses to view comments.

Step-by-Step Description:

Before this use case can be initiated, the User has already accessed the web portal.

1. The User logs in using their legitimate username and password.
2. The system, by default, displays a search page.
3. The User chooses to search by address or account number.
4. The system presents all applicable matches given the provided keywords.
5. The User chooses to open a select account.
6. The system provides the requested account information, along with an option to view comments associated with that account.
7. The User chooses to view comments.

Use Case: View Text

[insert diagram]

[Brief Description]

The user accesses the web portal, logs in, searches for a record, selects a record, and chooses to view text.

Step-by-Step Description:

Before this use case can be initiated, the User has already accessed the web portal.

1. The User logs in using their legitimate username and password.
2. The system, by default, displays a search page.
3. The User chooses to search by address or account number.
4. The system presents all applicable matches given the provided keywords.
5. The User chooses to open a select account.
6. The system provides the requested account information, along with an option to view the text associated with that account.
7. The User chooses to view the text.

2.2.2 Administrator Use Cases

Use Case: Add User

[insert diagram]

[Brief Description]

The system administrator accesses the Docker® container and runs a pre-made shell script by passing in select arguments.

Initial Step-by-Step Description

Before this use case can be initiated, the Administrator has already accessed the correct directory with the shell script.

1. The Administrator invokes the shell script with root permissions, using sudo.
2. The system returns the arguments which must be filled.
3. The Administrator furnishes a username, user type, and a password.
4. The system will add the username and password to all applicable files (.htaccess) and inform the Administrator that the actions have been carried out.

Use Case: Deactivate user

[insert diagram]

Brief Description:

The system administrator accesses the Docker® container and runs a pre-made shell script to deactivate users by passing in select arguments.

Initial Step-by-Step Description:

Before this use case can be initiated, the Administrator has already accessed the correct directory with the shell script.

1. The Administrator invokes the shell script with root permissions, using sudo.
2. The system returns the arguments which must be filled.
3. The Administrator furnishes a username.
4. The system will ask whether the Administrator would really like to deactivate the user.
5. The Administrator confirms the deactivation.
6. The system will deactivate the entry in the access file and inform the Administrator that the actions have been carried out.

Use Case: Add entry to read-only database

[insert diagram]

Brief Description:

The system administrator accesses the Docker® container and runs a pre-made shell script to add an entry into the database by passing in select arguments.

Initial Step-by-Step Description:

Before this use case can be initiated, the Administrator has already accessed the correct directory with the shell script.

1. The Administrator invokes the help menu for the script with root permissions, using sudo.
2. The system returns the arguments which must be filled.
3. The Administrator furnishes all of the necessary parameters to fill the new database row. If all necessary fields are not filled, the command will fail to run.
4. The system will ask whether the Administrator would really like to carry out the data addition with a keyboard-interactive prompt.
5. The Administrator confirms the addition.
6. The system adds the entry and informs the Administrator that the actions have been carried out.

Use Case: Append entry in the read-only database

[insert diagram]

Brief Description:

The system administrator accesses the Docker® container and runs a pre-made shell script to append the database entry by passing in select arguments.

Initial Step-by-Step Description:

Before this use case can be initiated, the Administrator already knows the account number to be modified and has accessed the correct directory with the script.

1. The Administrator invokes the keyboard-interactive prompts with root permissions, using sudo.
2. The system asks for the account number that must be modified.
3. The Administrator enters the account number exactly as it shows in the web portal.
4. The system will request that the new information be inputted in a keyboard-interactive environment, one parameter at a time.

5. The Administrator provides the information, one parameter at a time.
6. The system will ask whether the Administrator would really like to carry out the data append operation with another keyboard-interactive prompt.
7. The Administrator confirms the append operation.
8. The system appends the entry and informs the Administrator that the actions have been carried out.

2.3 User Characteristics

The Users are expected to be Internet literate and be able to use a search engine. The main screen of the database access web portal will have the search function and the types of searches.

The Administrator is expected to be reasonably literate with command-line interface and to be able to type in provided commands. To aid in this, the developers have elected to provide a documentation booklet with reasonably comprehensive information describing how to administer the system.

2.4 Non-Functional Requirements

The archive access system will be in a Docker® container on a server with a high-speed Ethernet connection to all client devices. The physical machine to be used will be determined by the City of Mount Vernon. The software developed here assumes the use of a modern web-browser, such as Google Chrome, Mozilla Firefox, or Microsoft Edge for connection between the web portal and the server. The speed of the User's connection will depend on the hardware used, rather than the characteristics of the system.

The administration console will run in an emulated terminal and will contain limited self-help documentation. For more information, the Administrator is encouraged to consult the provided, full documentation.

3. Requirements Specification

3.1 External Interface Requirements

There shall be no links to external systems.

3.2 Functional Requirements

3.2.1 User Record Access

Use Case Name	User Record Access
XRef	Section 2.2.1, User Record Access
Trigger	A User chooses to make a search
Precondition	The User has accessed and authenticated to the web portal.
Basic Path	1. The User logs in using their legitimate username and password. 2. The system, by default, displays a search page. 3. The User chooses to search by address or account number. 4. The system presents all applicable matches given the provided keywords. 5. The User chooses to open a select account. 6. The system provides the requested account information.
Alternative Paths	None.
Postcondition	The User is presented with matching entries based on their search parameters.
Exception Paths	The User can abandon or terminate the operation at any time with no ill effect.
Other	

3.2.2 Administrative user addition

Use Case Name	User addition
XRef	Section 2.2.2, User addition
Trigger	The Administrator invokes the user addition script.
Precondition	The Administrator has accessed and authenticated to an emulated terminal over secure shell.

Basic Path	1. The Administrator invokes the shell script with root permissions, using sudo. 2. The system returns the arguments which must be filled. 3. The Administrator furnishes a username, user type, and a password. 4. The system will add the username and password to all applicable files (.htaccess) and inform the Administrator that the actions have been carried out.
Alternative Paths	None.
Postcondition	The Administrator is presented with a notification that actions have been executed.
Exception Paths	The Administrator can terminate the operation at any time.
Other	

3.2.3 Administrative user deactivation

Use Case Name	User deactivation
XRef	Section 2.2.3, User deactivation
Trigger	The Administrator invokes the user deactivation script.
Precondition	The Administrator has accessed and authenticated to an emulated terminal over secure shell.
Basic Path	1. The Administrator invokes the shell script with root permissions, using sudo. 2. The system returns the arguments which must be filled. 3. The Administrator furnishes a username. 4. The system prompts the Administrator to confirm that they want to deactivate the user. 5. The system will deactivate the entry in the access file and inform the Administrator that the actions have been carried out.
Alternative Paths	None.
Postcondition	The Administrator is presented with a notification that actions have been executed.
Exception Paths	The Administrator can terminate the operation at any time.
Other	

3.2.4 Administrative R-- (read-only) Addition

Use Case Name	Database addition

XRef	Section 2.2.4, Database addition
Trigger	The Administrator invokes the database addition script.
Precondition	The Administrator has accessed and authenticated to an emulated terminal over secure shell.
Basic Path	1. The Administrator invokes the help menu for the script with root permissions, using sudo. 2. The system returns the arguments which must be filled. 3. The Administrator furnishes all of the necessary parameters to fill the new database row. If all necessary fields are not filled, the command will fail to run. 4. The system will ask whether the Administrator would really like to carry out the data addition with a keyboard-interactive prompt. 5. The Administrator confirms the addition. 6. The system adds the entry and informs the Administrator that the actions have been carried out.
Alternative Paths	None.
Postcondition	The Administrator is presented with a notification that actions have been executed. The new entry has been added to the database
Exception Paths	The Administrator can terminate the operation at any time during or before confirmation.
Other	

3.2.5 Administrative R-- (read-only) Append

Use Case Name	Database modification operation
XRef	Section 2.2.5, Database modification operation
Trigger	The Administrator invokes the database modification operation script.
Precondition	The Administrator has accessed and authenticated to an emulated terminal over secure shell.
Basic Path	1. The Administrator invokes the keyboard-interactive prompts with root permissions, using sudo. 2. The system asks for the account number that must be modified. 3. The Administrator enters the account number exactly as it is shown in the web portal. 4. The system will request that the new information be inputted in a keyboard-interactive environment, one parameter at a time.

	5. The Administrator provides the information, one parameter at a time. 6. The system prompts the Administrator to confirm that they would really like to carry out the data modification operation. 7. The Administrator confirms the modification operation. 8. The system modifies the entry and informs the Administrator that the actions have been carried out.
Alternative Paths	None.
Postcondition	The Administrator is presented with a notification that actions have been executed. The database now contains the modified version of the relevant entry.
Exception Paths	The Administrator can terminate the operation at any time during or before confirmation.
Other	

3.3 Detailed Non-Functional Requirements

3.3.1 Logical Structure of the Data

The logical structure of the data to be stored in the internal information database is given below.

[insert diagram here]

3.3.2 Security

The server on which the Docker® container resides will have its own security protocol to prevent unauthorized Read/Write/eXecute access. All permissions are restricted by system policy. The administrative console also has its own security protocol. The Administrator is required to have root rights to make changes, and must furnish their password.

3.3.3 Proposed Technology Stack

We will architect and build the software according to our recommended technology stack.

Function	**Technology**
Server-Side Programming Language	C++14
Client-Side Programming Language	HTML5, CSS3, jQuery, Bootstrap 4.3
Database	MySQL 5.6
Web Server	Apache 2.x.x OR Nginx 1.x.x
Local Hosting Hypervisor	Oracle VirtualBox 6
Guest Operating System	Ubuntu Server 18.04 LTS
Containerization Technology	Docker®, running Kubernetes® microservices
Cloud Storage	None recommended
Content Delivery Network (CDN)	None recommended
Session & Cache Storage	None recommended
Version Control System	Git

For security reasons, local hosting is recommended for hosting the platform, and the minimum server requirements are as mentioned below.

- LAMP Stack - *https:///en.wikipedia.org/wiki/LAMP_(software_bundle)*
- Operating System - Linux x86, x86-64
- PHP version 7.0 or greater - *http://php.net*
- MySQL version 5.6 or greater - *https://www.mysql.com/*

- Apache 2.x.x (mod_rewrite module enabled, *https://httpd.apache.org/*) OR Nginx 1.x.x (*https://www.nginx.com/resources/wiki/*)
- Required PHP extensions
    - OpenSSL
    - XML
    - Ctype
    - JSON
- Docker® 18 or greater - *https://www.docker.com/*

For better performance, reliability, security, and scalability, we suggest the following:

- Securing the website by SSL certificate.

We recommend deploying the software solution in a virtual machine, using the Oracle VirtualBox hypervisor. This would allow you to run Unix executable files on your Windows Server-based servers. We are only supporting deployments on bare metal with Windows Server 2012 or later, Ubuntu Server 16.04 LTS (xenial) or 18.04 LTS (bionic), Debian 10 (buster), and CentOS 7.

We are unable to support macOS Server deployments and will not support Windows Server 2008 or earlier. If your server environments are still running on Windows Server 2008 or earlier, we require you to upgrade to Server 2012 or later, in order to receive the latest security and feature updates from Microsoft.