Vaq-Paq

Software Requirements Specification

Version 1.0

Nov, 7 2016

Stephen Lightcap
Lily Martinez
Victor Rodriguez
Francisco Humberto Hernández Isaac

Prepared for
Software Engineering
Instructor: MK Quweider, Ph.D.
Fall 2016

Revision History

Date	Description	Author	Comments
Nov-7-16	Version 1.0	Stephen Lightcap	First revision

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	Stephen Lightcap	Project Manager	Nov – 7 - 2016
	MK Quweider	Instructor, CptS 322	

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification SRS is to give a detailed description and use of services and the functionality of the Vaq-Paq, which is a piece of Software for UTRGV. This document is to serve as a manual for the clients, developers, and other stakeholders.

1.2 Scope

This piece of Software(s), libraries, and packages are for the Vaq-Pac – including all its services and implementations. The software will provide a GUI (Graphical User Interface) for the clients to interact with and to deploy underlying services the software provides. Services include: file formatting (pdf, html and xml), DBMS (Database Management Service) - for storing said files, University Information (such as courses, program of studies, and transcripts). It will also implement network to provide said software to connect to the DBMS with login access to a personal account.

1.3 Definitions, Acronyms, and Abbreviations

XML – Extensible Markup Language
TXT – File to save simple text characters
SRS – Software Requirements Specification
PDF – Portable Document File
POS – Program of Study
IDE – Interactive Development Environment
Java – OOP language used to implement said software
JavaFX – GUI views for Java
MySQL – Local Database
HTML – Hypertext Markup Language
GUI – Graphical User Interface
SRS – Software Requirement Specification.
EventHandlers – Reading user inputs into the software.
OS – Operating System
Vaq-Paq – This Piece of Software
DBMS – Database Management Systems
UTRGV – University of Texas at Rio Grande Valley
XSL – language for expressing style sheets
Network – Transfer of data between Database and Vaq-Paq

1.4 References

Netbeans - https://netbeans.org/
MySQL - https://www.mysql.com/
Github – https://www.github.com/

JavaFX - http://docs.oracle.com/javase/8/javase-clienttechnologies.htm

1.5 Overview

The sections are divided into 3 categories – General Description, Requirements, and Analysis.

2. General Description

2.1 Product Perspective

The software will have a friendly GUI that the client interacts with to view courses and POS for UTRGV. It will store the data on a Database and connect with it through a secure network. The data will be course, transcript, POS files which will be in PDF, XML, and HTML formats – these files can be generated on the fly in the software as a service.

2.2 Product Functions

- *User Friendly GUI*
- *Database Connection*
- *Network Connection*
- *File Conversion*
- *File Manipulation*

2.3 User Characteristics

The user should be able to read and write. Also, basic understanding of computer interfaces and hardware – i.e. button controls and mouse w/ keyboard.

2.4 General Constraints

- *JavaFX for GUI*
- *MySQL for Database*
- *XML to hold data*
- *Github for Version Control*

2.5 Assumptions and Dependencies

It is assumed that the user will be running Windows OS, able to run java programs with Java runtime, have a keyboard w/ mouse, and network connectivity.

3. Specific Requirements

3.1 External Interface Requirements

The Database will have an external interface for the maintenance of said database. If the user chooses to view a certain PDF, HTML of XML outside of the software, a browser may be viewed as an external interface, but isn't necessary.

3.1.1 User Interfaces

The GUI will have input fields, input buttons or any such component that requires an input from an external piece of hardware. These input fields will connect the user to services – i.e. Converters, Generators, Viewers, Networks, etc. There will be two interfaces – Administration and one for the general user (Student).

3.1.2 Hardware Interfaces

- *Client Side: Java and the most minimum requirements for the home PC, MAC, and small consumer computers will suffice.*
- *Server Side: Sufficient RAM and Storage.*
- *Network: Network Connection.*

3.1.3 Software Interfaces

- *Database Server (MySQL) – Operating System (any)*
- *Client Application – Vaq-Paq GUI.*
- *Java Runtime – Run program/JAR.*

3.1.4 Communications Interfaces

Client will be using HTTP/HTTPS, but the connectivity will all be achieved within the Vaq-Paq.

3.2 Functional Requirements

- *User Friendly GUI*
  - *Software will have all the needed controls to operate the included functions.*
  - *GUI will have initiate network and database functions w/out without the client noticing.*
- *Database Connection*
  - *The DB will have a connection to the Vaq-Paq.*
  - *The DBMS will be able to modify, insert, and delete records.*
  - *The DB will have attributes that will make accessing it viable.*
- *Network Connection*
  - *The network will connect to the Vaq-Paq.*
  - *The network will connect to the DB.*
  - *The network will be secure.*
- *File Conversion*
  - *The file conversion will operate decupled from all other components of the Vaq-Paq.*
  - *The file conversion will be fast – 2 to 3 seconds is the limit.*
  - *The file conversion will convert XML to HTML and HTML to PDF*
  - *The file conversion will send the output files into the database for storage.*
- *File Manipulation*
  - *File manipulation will be able to merge documents together.*

3.2.1 User Friendly GUI

Purpose	To assist the user in handling Events and using internal functions.
Inputs	Button clicks and Text inputs.
Processing	Use of EventHandlers to bring to the forefront of the GUI the users desired results by

	starting internal functions.
Outputs	Starting network transfer to database, converting/merging files, or displaying other GUI components.
Error Handling	Account for chaotic button clicks (threads) and invalid user input (try catch).

3.2.2 Database connection

Purpose	To store persistent data across application runs.
Inputs	Records – user info, courses, and POS.
Processing	Saving tuples into a table.
Outputs	Send info to the GUI.
Error Handling	Tell GUI data cannot be inserted, deleted, or modified (try catch).

3.2.3 Network connection

Purpose	To send data between application and DB server.
Inputs	Packets.
Processing	Through sockets.
Outputs	Connection established.
Error Handling	Send error to GUI to be seen be the user (try-catch that the connection was not established, lost or not secure).

3.2.4 Files

Purpose	To convert files for user’s viewing, storage or manipulation.
Inputs	Files: XML, XSL, HTML, and PDF
Processing	Converts (Factory Libraries) to output desired file.
Outputs	Another file in a different format or a merged one.
Error Handling	File cannot be manipulated (try-catch)

3.3 Use Cases

3.3.1 Convert XML to HTML

The goal in this use case is to convert XML to HTML. This is for the raw data (XML) to look presentable to the user (HTML). To do this the XML will be fed through a library - Transformer.

XML *-> XSL* *-> HTML*

```
<Data>
   <More-Data/>
</Data>
```

```
<Template>
   …
   <html>
   … <XSL select = data>
   </html>
</Template>
```

```
<html>

<body>
</body>

</html>
```

3.3.2 Convert HTML to PDF

The goal in this use case is to convert HTML to PDF. This is for the HTML web view to be converted to a document format - PDF. This is achieved through a library called ITextRenderer.

3.4 Classes / Objects

3.4.1 Courses / Course

3.4.1.1 Attributes

- *Name*
- *Divide*
- *Prefix*
- *Description*
- *Number*
- *Department*
- *Prerequisites*
- *Coerequisites*
- *Cross listed*
- *Legacy Number*

3.4.1.2 Functions

Setters and Getters for each Attribute.

3.4.2 XML2HTML

3.4.2.1 Attributes

- *Path (string)*
- *ArrayList (String)*
- *File*

3.4.2.2 Functions

- *ConvertToHtml*
- *ConvertListToHtml*
- *GetPath*

3.4.3 HTML2PDF

3.4.2.1 Attributes

- *Path (string)*
- *ArrayList (String)*
- *File*

3.4.2.2 Functions

- *ConvertToPDF*
- *ConvertListToPDF*
- *GetPath*

3.5 Non-Functional Requirements

3.5.1 Performance

All functions should not take more than a few seconds. Network connections is at the whim of Client's own internet connection.

3.5.2 Reliability

The software should not crash.

3.5.3 Availability

The software should be available 24/7 on local host machine and when an internet connection can be connected, it should be able to connect.

3.5.4 Security

Secure connection w/ a secure DB server.

3.5.5 Maintainability

The Admin will oversee maintenance of the server and the application should require little to no maintenance.

3.5.6 Portability

Software will run on desktop like machine, but can be ran on other if the other machines have Java.

3.6 Inverse Requirements

User may not be able to modify already created courses.

3.7 Design Constraints

N/A currently

3.8 Logical Database Requirements

RAID, or Certain requirements such as storage have not been made.

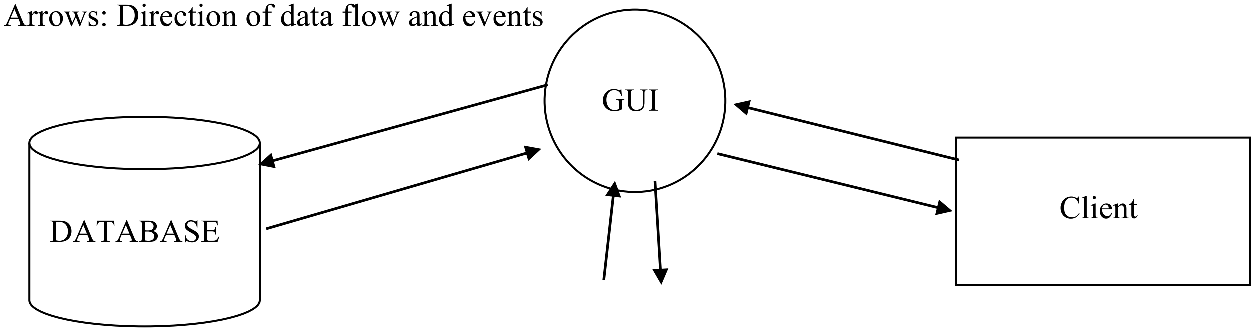
3.9 Other Requirements

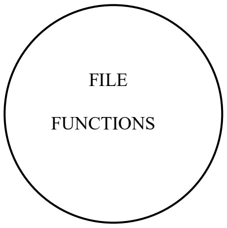
N/A.

4. Analysis Models

4.1 Sequence Diagrams

Arrows: Direction of data flow and events





4.3 Data Flow Diagrams (DFD)

4.2 State-Transition Diagrams (STD)

5. Change Management Process

Changes can be made by any on the developer teams. But to commit the change to the final SRS, you must seek approval from Instructor.

A. Appendices

Appendices may be used to provide additional (and hopefully helpful) information. If present, the SRS should explicitly state whether the information contained within an appendix is to be considered as a part of the SRS's overall set of requirements.

Example Appendices could include (initial) conceptual documents for the software project, marketing materials, minutes of meetings with the customer(s), etc.

A.1 Appendix 1

A.2 Appendix 2