**Software Requirements**

**Specification**

**VaqPack**

Graduate-to-Professional

Aid Pack

Version 1.0

November 5th, 2015

**Lead Software Engineer:**

William Dewald

**Project Team:**

Fernando Bazan

Nathanael Carr

Erik Lopez

Raul Saavedra

Prepared for

Software Engineering

University of Texas Rio Grande Valley

Instructor: MK Quweider, Ph.D.

Fall 2015

**Table of Contents**

**1. INTRODUCTION**……………………………………………………..... **1**

1.1 PURPOSE………………………………………………………………...…. 1

1.2 SCOPE…………………………………………………………………...…. 1

1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS…………...…… 1

1.4 REFERENCES……………………………………………………...………. 3

1.5 OVERVIEW………………………………………………………...………. 3

**2. GENERAL DESCRIPTION**…………………………………………..... **4**

2.1 PRODUCT PERSPECTIVE………………………………………………... 4

2.2 PRODUCT FUNCTIONS………………………………………………...… 4

2.3 USER CHARACTERISTICS………………………………………………. 4

2.4 GENERAL CONSTRAINTS……………………………………………….. 5

2.5 ASSUMPTIONS AND DEPENDENCIES…………………………………. 5

**3. SPECIFIC REQUIREMENTS**…………………………………………. **6**

3.1 EXTERNAL INTERFACE REQUIREMENTS…………………………..... 6

3.1.1 User Interfaces……………………………………………………………….. 6

3.1.2 Hardware Interfaces…………………………………………………………. 6

3.1.3 Software Interfaces………………………………………………………..… 6

3.1.4 Communications Interfaces………………………………………………….. 6

3.2 FUNCTIONAL REQUIREMENTS……………………………………...… 6

3.2.1 Store User Login Credentials………………………………………….…….. 6

3.2.2 Store Collected User Information ……………………………………….….. 7

3.2.3 Store User’s Contacts ………………………………………………….…….. 7

3.2.4 Database Initialization ………………………...…………………………….. 7

3.2.5 Database Admin Connection ……………………………………………….. 8

3.2.6 Reset User Password …………………………………………….………….. 8

3.2.7 Database Migration ………………………………………………………….. 9

3.3 USE CASES ………………………………………………………………... 9

3.3.1 Create a VaqPack Account………………………………………………….. 9

3.3.2 User Creates a Resume…………………………………………………….... 10

3.3.3 User Creates a Business Card……………………………………………….. 10

3.3.4 User Emails Résumé / Business Card / Both………………………………... 11

3.3.5 User Updates Résumé / Business Card / Both………………………………. 11

3.4 CLASSES / OBJECTS…………………………………………….……….. 12

3.4.1 VaqPack…………………………………………………………………..… 12

3.4.2 VP\_GUIController………………………………………………………….. 12

3.4.3 VP\_GUIBuilder…………………………………………………………….. 12

3.4.4 VP\_Header…………………………………………………………………. 12

3.4.5 VP\_Tree……………………………………………………………………. 13

3.4.6 VP\_Center………………………………………………………………….. 13

3.4.7 VP\_Loader…………………………………………………………………. 13

3.4.8 VP\_DataManager…………………………………………………………... 13

3.4.9 VP\_DatabaseManager……………………………………………………… 13

3.4.10 VP\_FileManager…………………………………………………………… 13

3.4.11 VP\_DataToHtml…………………………………………………………… 13

3.4.12 VP\_HtmlToPdf…………………………………………………………….. 14

3.4.13 VP\_Mail…………………………………………………………………… 14

3.4.14 VP\_ErrorHandler…………………………………………………………... 14

3.4.15 VP\_User……………………………………………………………………. 14

3.4.16 VP\_Template………………………………………………………………. 14

3.4.17 VP\_Contact………………………………………………………………… 14

3.4.18 VP\_CoverLetter……………………………………………………………. 14

3.4.19 VP\_Resume………………………………………………………………… 15

3.4.20 VP\_BusinessCard …………………………………………………………. 15

3.5 NON-FUNCTIONAL REQUIREMENTS…………………………………. 15

3.5.1 Performance…………………………………………………………………. 15

3.5.2 Reliability……………………………………………………………………. 15

3.5.3 Availability………………………………………………………………….. 15

3.5.4 Security………………………………………………………………………. 15

3.5.5 Maintainability………………………………………………………………. 16

3.5.6 Portability……………………………………………………………………. 16

3.6 INVERSE REQUIREMENTS……………………………………………… 16

3.7 DESIGN CONSTRAINTS 4………………………………………………... 16

3.8 LOGICAL DATABASE REQUIREMENTS……………………………… 16

3.9 OTHER REQUIREMENTS……………………………………………….. 16

**4. ANALYSIS MODELS**…………………………..………………………. 17

4.1 SEQUENCE DIAGRAMS…………………………………………………. 17

4.2 DATA FLOW DIAGRAMS (DFD)………………………………………... 17

4.3 STATE-TRANSITION DIAGRAMS (STD)……………………………….. 17

**5. CHANGE MANAGEMENT PROCESS**……………………………….. 17

**A. APPENDICES**…………………………………………………………… 18

A.1 Appendix 1………………………………….…………………………….... 18

A.2 Appendix 2…………………………………………………………………. 18

A.3 Appendix 3…………………………………………………...…………….. 19

A.4 Appendix 4…………………………………………………..……………... 19

A.5 Appendix 5…………………………………………………..…………….. 20

**1. Introduction**

**1.1 Purpose**

The purpose of the Software Requirements Specification is to provide a detailed description of the VaqPack Graduate to Professional Aid Pack application. The intention of the SRS is to articulate the purpose and features of the application, along with its user and external interfaces, constraints, dependencies, functionality, and attributes. This artifact provides the guidelines for the design and implementation of the software, and clarifies the description of the software for the customer. Therefore, the intended audience of this document includes the client, users, and developers.

**1.2 Scope**

The software application described throughout this SRS document is the VaqPack Graduate to Professional Aid Pack, or simply VaqPack. While this free desktop application can be used within any institution, it is primarily designed for the graduating students of the University of Texas Rio Grande Valley with the purpose of aiding their progression from academia to the professional world.

Using a graphical user interface, registered users of VaqPack can generate a resume, a business card, and cover letters. The information required for the generation of these objects is collected from the user, by means of input forms within a wizard, and then stored in a pre-existing MySQL database. Users can apply themes of their choice to these objects for personal style or look-and-feel. From these objects, the user may generate PDF documents which can be sent to contacts or potential employers via email. Additionally, the user may generate an HTML file from the resume data for use as a web page. Users may retrieve stored data for the purpose of editing or augmenting information, or to send documents to contacts at any given time.

**1.3 Definitions, Acronyms, and Abbreviations**

The following terms, acronyms, and abbreviations are used throughout this document and are presented in the table below by order of appearance.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| SRS | Software Requirement Specification |
| VaqPack | VaqPack Graduate to Professional Aid Pack, in short |
| GUI | Graphical User Interface; provides a visual, interactive means for a software user to manipulate the controls, commands, or features of that software. |
| Wizard | A sequential set of prompts for input, assisting in data collection and organized such that its implementation increases ease of use. |
| Database | A structured collection of data that can be efficiently and conveniently accessed. |
| PDF | Portable Document Format; a popular electronic document file type particularly used with rich-text or styled text. |
| HTML | HyperText Markup Language; the web standard language used in the delivery of online content, interpreted and rendered by web browsers. |
| IDE | Integrated Development Environment; software that provides tools for the development and organization of programming code. |
| Git | A version control system for the development of software. |
| GitHub | A web-based Git repository used by software development teams. |
| Java Virtual Machine | Provides the necessary links allowing a java program to run on a machine using a particular operating system. |
| Java Runtime Environment | Including the Java Virtual Machine, all necessary components for a system to establish the environment in which Java programs will run. |
| SQL | Structured Query Language; the standard relational database query language |
| JDBC | Java Database Connectivity; an Java API developed by Oracle Corporation which provides methods for querying and updated a database. |

**1.4 References**

Git - <https://git-scm.com/>

GitHub - <https://github.com/>

Java Virtual Machine - <https://java.com/en/download/>

Java Runtime Environment - <http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html>

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

MySQL - <http://dev.mysql.com/downloads/mysql/>

NetBeans - <https://netbeans.org/>

**1.5 Overview**

The remaining content of this SRS is organized in 5 sections: General Description, Specific Requirements, Analysis Models, Change Management Process, and the Appendices. The General Description section aims to make the requirements for the VaqPack application more easily understood from a high-level point of view, especially from the perspective of typical end-users. However, the Specific Requirements will define and describe the details of these requirements with the technical information needed by the developers. The Analysis Models section lists all of the models used in developing the specific requirements that are outlined in the previous section. Since the first version of VaqPack is currently being constructed, and since many requirements still need to be met, this section is subject to much change and many additions. The Change Management Process section outlines the procedures that must be followed when such changes occur throughout the development of VaqPack, including the updates to this SRS document. The Appendices include conceptual documents such as the initially provided high-level requirements and any conceptual diagrams or documents used by the developers. The documents in the Appendix may or may not be used in requirements definitions, but this is clearly specified for each document.

**2.** **General Description**

**2.1 Product Perspective**

The VaqPack product is independent in that it does not augment any existing product and is not intended to encapsulate another product. However, VaqPack must operate along with a MySQL server. Connectivity to a MySQL server is included with the software. VaqPack intends to replace an array of available paid-access products that focus on assisted resume and business card construction, ranging from desktop applications to online services.

**2.2 Product Functions**

In a general high-level point of view, the VaqPack application will perform the following functions:

* Store user login credentials and privileges in a database.
* Store collected user information in a database.
* Store a list of the user’s contacts in a database.
* Provide a system for an admin user to connect to a MySQL database server.
* Provide a system for an admin user to initialize the database on first run.
* Retrieve individual collected user information for viewing or editing.
* Retrieve individual collected user contacts for viewing or editing.
* Generate a resume object from collected user information.
* Generate a business card object from collected user information.
* Generate employer-specific cover letter objects.
* Apply themes or styles to the produced objects.
* Generate printable, distributable PDF files of the objects.
* Generate an HTML file of the produced resume.
* Store generated files in the database for fast access.
* Email the generated files to selected user contacts.
* Email the generated files to the user for personal access.
* Prompt the user with a wizard for ease in data collection.
* Provide a system to reset a forgotten user password.
* Provide a system for an admin user to migrate the database.

**2.3 User Characteristics**

While the VaqPack product is primarily designed for the graduating students of the University of Texas Rio Grande Valley, it can be used by any individual who needs to prepare and distribute the personal information that is required for employment consideration within most professions. Additionally, the VaqPack user could simply be interested in generating a printable business card for distribution to clients or for business-related networking within his or her current profession.

**2.4 General Constraints**

In a general high-level point of view, the developers of the VaqPack application will have the following constraints:

* VaqPack must be developed using the NetBeans IDE.
* VaqPack must be MySQL database-driven
* Choice of local or remote database server must be present.
* The VaqPack GUI must be JavaFX-based.
* VaqPack must be desktop-based.
* Git and GitHub must be used for version control.

**2.5 Assumptions and Dependencies**

In a general high-level point of view, the developers of the VaqPack application requirements are currently influenced by the following assumptions and dependencies:

* It is assumed that VaqPack will run on a system with an operating system that has a compatible Java Virtual Machine and up to date Java Runtime Environment.
* It is assumed that VaqPack will run on or connect to a system with an existing MySQL server.
* It is assumed, in the event of remote MySQL server connectivity, the system on which VaqPack will run has the networking capabilities to connect to said database server.

**3.** **Specific Requirements**

**3.1 External Interface Requirements**

**3.1.1 User Interfaces**

Like the majority of modern desktop software applications, VaqPack provides a GUI for the user to interface with all of the functionality necessary to accomplish the user’s goals in a visual manner—creating distributable, styled resumes, business cards, and cover letters.

**3.1.2 Hardware Interfaces**

VaqPack requires the screen, keyboard and mouse that is typical for desktop machines. The hardware required is that of a modern desktop computer.

**3.1.3 Software Interfaces**

VaqPack is Java program and therefore interfaces with the Java Runtime Environment and Java Virtual Machine for whichever platform the program must run. VaqPack must operate along with a MySQL server. Connectivity to a MySQL server is included with the software.

**3.1.4 Communication Interfaces**

VaqPack can connect to a remote MySQL database which may require Internet connectivity. VaqPack can also send files as email attachments which requires Internet connectivity. The product communicates with the MySQL server via a driver that is embedded in the software and uses SQL standards.

**3.2 Functional Requirements**

**3.2.1 Store User Login Credentials**

3.2.1.1 Introduction

VaqPack stores the user’s chosen email and hashed password in a database.

3.2.1.2 Inputs

The user provides their email address and a password of their choosing.

3.2.1.3 Processing

VaqPack stores the email and hashed password in the ‘user’ database table.

3.2.1.4 Outputs

Upon entering the proper credentials, the user is allowed access to their stored data.

3.2.1.5 Error Handling

VaqPack verifies the correct email and password are used to login and that valid inputs are provided at the time of account creation.

**3.2.2 Store Collected User Information**

3.2.2.1 Introduction

VaqPack provides a sequential process, or ‘wizard’, for gathering the user’s data and storing that data in a database.

3.2.2.2 Inputs

The user provides their full name, mailing address, phone number(s), career objective(s), employment history, professional skills, and references.

3.2.2.3 Processing

VaqPack stores the collected data in the ‘user\_data’ and ‘resume’ tables of the database.

3.2.2.4 Outputs

No particular output unless retrieved via database query.

3.2.2.5 Error Handling

VaqPack sanitizes inputs using software security best practices.

**3.2.3 Store User’s Contacts**

3.2.3.1 Introduction

If the user enters one or more contacts, VaqPack stores those contacts in the database for future use.

3.2.3.2 Inputs

The user provides the full name and email address for each contact they wish to add.

3.2.3.3 Processing

VaqPack stores the collected data in the ‘contact’ tables of the database.

3.2.3.4 Outputs

No particular output unless retrieved via database query.

3.2.3.5 Error Handling

VaqPack sanitizes inputs using software security best practices.

**3.2.4 Database Initialization**

3.2.4.1 Introduction

Upon launch of VaqPack, if there is no existing database, VaqPack generates the required database and associated tables.

3.2.4.2 Inputs

A stored SQL script detailing the required tables and associated attributes.

3.2.4.3 Processing

VaqPack utilizes JDBC to create a database object with a predefined admin-level account.

3.2.4.4 Outputs

VaqPack produces a MySQL database with an admin-level user and predefined tables.

3.2.4.5 Error Handling

VaqPack verifies that the required database is non-existent before a new database is initialized and ensures database integrity post initialization.

**3.2.5 Database Admin Connection**

3.2.5.1 Introduction

Upon launch of VaqPack, if there is an existing database, VaqPack connects to that database as an admin user.

3.2.5.2 Inputs

VaqPack contains a pre-defined admin-level database user.

3.2.5.3 Processing

VaqPack connects to the database as an admin.

3.2.5.4 Outputs

VaqPack produces a liv admin-level database connection.

3.2.5.5 Error Handling

VaqPack notifies the user if the admin account cannot login and the reason for the connection error.

**3.2.6 Reset User Password**

3.2.6.1 Introduction

If the user forgets their password, VaqPack allows them to reset the password.

3.2.6.2 Inputs

The user inputs a code sent to the user’s email, of which must be entered within 60 minutes.

3.2.6.3 Processing

If user authentication passes, VaqPack allows the user to create a new password and login using that password.

3.2.6.4 Outputs

User regains access to their stored information.

3.2.6.5 Error Handling

VaqPack notifies the user if their authentication fails and limits the number of consecutive attempts.

**3.2.7 Database Migration**

3.2.7.1 Introduction

A system which allows the VaqPack user to migrate their database to another system as needed.

3.2.7.2 Inputs

The user provides a valid login.

3.2.7.3 Processing

If user authentication passes, VaqPack allows the user to login and export their data to a specified location. Alternatively, if the user is reestablishing their database, VaqPack allows import of a data file.

3.2.7.4 Outputs

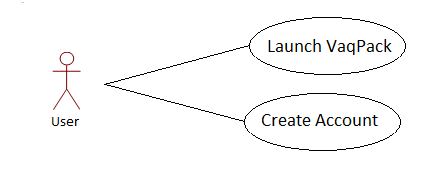
If exporting, the exported data file. If importing, a restored user database.

3.2.7.5 Error Handling

The user will be notified upon a failed export or import of the user data.

**3.3 Use Cases**

**3.3.1 Create a VaqPack Account**

****

**Goal:** User Creates an Account

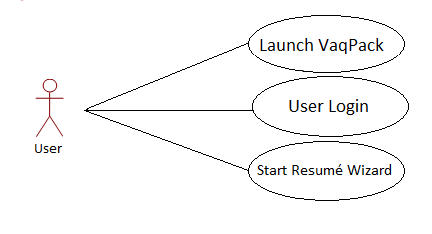
**Input:** Email/Password

**Output:** none.

**Main Scenario:** User creates an account to store user login credentials

**Pre-Condition:** VaqPack must be running.

**3.3.2 User Creates a Resume**

****

**Goal:** User Creates a Résumé

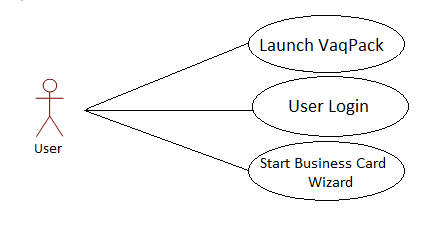
**Input:** User chooses a template & fills in all required fields in the résumé wizard.

**Output:** Resume PDF, Resume HTML,

**Main Scenario:** User wants to create a résumé.

**Pre-Condition:** VaqPack must be running, user must login.

**3.3.3 User Creates a Business Card**

****

**Goal:** User Creates a Business Card

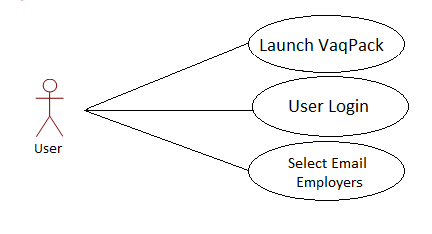
**Input:** User chooses a template & fills in all required fields in the business card wizard.

**Output:** Business Card

**Main Scenario:** User wants to create a business card.

**Pre-Condition:** VaqPack must be running & user must log in.

**3.3.4 User Emails Résumé / Business Card / Both**

****

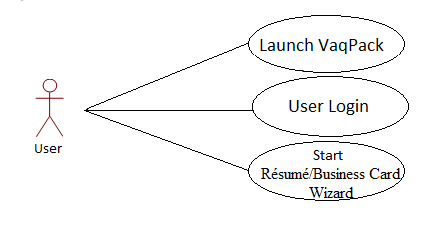
**Goal:** User Emails Potential Employers Résumé / Business Card

**Input:** User inputs destination email address and recipient information.

**Output:** Email containing Résumé and or Business Card.

**Main Scenario:** User wants to email Résumé/Business Card to potential employers.

**Pre-Condition:** VaqPack must be running, user must log in, Résumé Wizard and or Business Card Wizard must have already been completed.

**3.3.5 User Updates Résumé / Business Card / Both**

**Goal:** User Updates Résumé / Business Card

**Input:** User chooses which fields to update when running through the Wizard again.

**Output:** Résumé PDF / Résumé HTML / Business Card

**Main Scenario:** User wants to update personal information on Résumé / Business Card.

**Pre-Condition:** VaqPack must be running, user must log in, Résumé and or Business Card Wizards must be complete.

**3.4 Classes / Objects**

**3.4.1 VaqPack**

**3.4.1.1 Description:**

This class contains the main method entry point for the program. The JavaFX Stage is launched, the window is given a title, and a VP\_GUIController object is created which begins the construction of the user interface. Upon completion of the construction, the stage is shown to the user.

**3.4.1.2 Attributes:**

This class has no attributes.

**3.4.1.3 Functions:**

* main – launches JavaFX Stage
* start – creates a VP\_GUIController object, names the window, shows the stage.

**3.4.1.4 Use Case Reference:**

This class is inherently referenced by all use-cases since it is the main entry point for the program.

**<< THE FOLLOWING SPACE IN 3.4 IS RESERVED FOR THE CLASSES AND OBJECTS THAT ARE ANTICIPATED TO BE USED FOR THIS PROJECT. The descriptions and details for these classes and objects will be updated accordingly during the design process >>**

**3.4.2 VP\_GUIController**

**3.4.2.1 Description:**

**3.4.2.2 Attributes:**

**3.4.2.3 Functions:**

**3.4.2.4 Use Case Reference:**

**3.4.3 VP\_GUIBuilder**

**3.4.3.1 Description:**

**3.4.3.2 Attributes:**

**3.4.3.3 Functions:**

**3.4.3.4 Use Case Reference:**

**3.4.4 VP\_Header**

**3.4.4.1 Description:**

**3.4.4.2 Attributes:**

**3.4.4.3 Functions:**

**3.4.4.4 Use Case Reference:**

**3.4.5 VP\_Tree**

**3.4.5.1 Description:**

**3.4.5.2 Attributes:**

**3.4.5.3 Functions:**

**3.4.5.4 Use Case Reference:**

**3.4.6 VP\_Center**

**3.4.6.1 Description:**

**3.4.6.2 Attributes:**

**3.4.6.3 Functions:**

**3.4.6.4 Use Case Reference:**

**3.4.7 VP\_Loader**

**3.4.7.1 Description:**

**3.4.7.2 Attributes:**

**3.4.7.3 Functions:**

**3.4.7.4 Use Case Reference:**

**3.4.8 VP\_DataManager**

**3.4.8.1 Description:**

**3.4.8.2 Attributes:**

**3.4.8.3 Functions:**

**3.4.8.4 Use Case Reference:**

**3.4.9 VP\_DatabaseManager**

**3.4.9.1 Description:**

**3.4.9.2 Attributes:**

**3.4.9.3 Functions:**

**3.4.9.4 Use Case Reference:**

**3.4.10 VP\_FileManager**

**3.4.10.1 Description:**

**3.4.10.2 Attributes:**

**3.4.10.3 Functions:**

**3.4.10.4 Use Case Reference:**

**3.4.11 DataToHtml**

**3.4.11.1 Description:**

**3.4.11.2 Attributes:**

**3.4.11.3 Functions:**

**3.4.11.4 Use Case Reference:**

**3.4.12 VP\_** **HtmlToPdf**

**3.4.12.1 Description:**

**3.4.12.2 Attributes:**

**3.4.12.3 Functions:**

**3.4.12.4 Use Case Reference:**

**3.4.13 VP\_Mail**

**3.4.13.1 Description:**

**3.4.13.2 Attributes:**

**3.4.13.3 Functions:**

**3.4.13.4 Use Case Reference:**

**3.4.14 VP\_ErrorHandler**

**3.4.14.1 Description:**

**3.4.14.2 Attributes:**

**3.4.14.3 Functions:**

**3.4.14.4 Use Case Reference:**

**3.4.15 VP\_User**

**3.4.15.1 Description:**

**3.4.15.2 Attributes:**

**3.4.15.3 Functions:**

**3.4.15.4 Use Case Reference:**

**3.4.16 VP\_Template**

**3.4.16.1 Description:**

**3.4.16.2 Attributes:**

**3.4.16.3 Functions:**

**3.4.16.4 Use Case Reference:**

**3.4.17 VP\_Contact**

**3.4.17.1 Description:**

**3.4.17.2 Attributes:**

**3.4.17.3 Functions:**

**3.4.17.4 Use Case Reference:**

**3.4.18 VP\_CoverLetter**

**3.4.18.1 Description:**

**3.4.18.2 Attributes:**

**3.4.18.3 Functions:**

**3.4.18.4 Use Case Reference:**

**3.4.19 VP\_Resume**

**3.4.19.1 Description:**

**3.4.19.2 Attributes:**

**3.4.19.3 Functions:**

**3.4.19.4 Use Case Reference:**

**3.4.20 VP\_CoverLetter**

**3.4.20.1 Description:**

**3.4.20.2 Attributes:**

**3.4.20.3 Functions:**

**3.4.20.4 Use Case Reference:**

**3.5 Non-Functional Requirements**

**3.5.1 Performance**

There are no performance requirements at this stage of the development of VaqPack. However, the product performance shall perform under typical standards.

**3.5.2 Reliability**

There are no reliability requirements at this stage of the development of VaqPack. However, it is the goal of the development team to ensure maximum reliability.

**3.5.3 Availability**

The stored data for any particular user must be available upon request, provided that there are no other issues external to the VaqPack software such as hardware failure, the database server being inactive, or lack of or poor Internet connectivity preventing proper access to a remote MySQL database server.

**3.5.4 Security**

The following system is in place to avoid the compromise of user data. Users of VaqPack must login with a user and password. The password must be hashed before being stored in the database. A user may request to change his or her password, upon which an email with a confirmation code will be sent to the email associated with the user. The user must enter the code into the form provided in VaqPack within 60 minutes in order to successfully change the password.

**3.5.5 Maintainability**

No requirements concerning maintainability have been outlined by the customer. However, the VaqPack development team members have specific coding standards they must follow to enhance the readability of the programming code and thus the maintainability of the program. These coding standards can be found in Appendix 5.

**3.5.6 Portability**

No portability requirements have been defined by the customer. However, the requirement of Java as the programming language for the VaqPack project implies that the product is portable to any machine with an operating system for which some Java Runtime Environment with corresponding Java Virtual Machine exists.

**3.6 Inverse Requirements**

The VaqPack application will not provide any functionality without a connection to a MySQL database. Upon such an event, the user is to be notified, followed by an immediate closing of the program.

Besides any administrator user account, the typical user shall not ever be granted the privilege of accessing another user’s data.

**3.7 Design Constraints**

The VaqPack design must incorporate in some way a wizard, with which the program collects the data from the user in sequential, organized manner.

**3.8 Logical Database Requirements**

VaqPack must access and store data in a MySQL database. Functionality must be in place to allow the migration of data from a database server at one location to another. The only requirements for the types of data and capabilities are that they align with the completion of a working product that meets all other requirements. Additionally, the security requirements, as outlined in section 3.5.4 must be implemented.

**3.9 Other Requirements**

No other requirements exist at this stage of development.

**4.** **Analysis Models**

There are no models to present at this stage of development. However, a preliminary data model can be found in Appendix 4.

**4.1 Sequence Diagrams**

There are no sequence diagrams related to meeting requirements at this stage of development. This section is to be updated soon.

**4.2 Data Flow Diagrams (DFD)**

There are no data flow diagrams related to meeting requirements at this stage of development. This section is to be updated soon.

**4.3 State-Transition Diagrams (STD)**

There are no state-transition diagrams related to meeting requirements at this stage of development. This section is to be updated soon.

**5.** **Change Management Process**

The following procedures are required when changing this SRS document.

* Suggested changes by the development team to the scope or requirements outlined in this SRS document are to be presented independently by creating a branch for the edit in GitHub.
* The project manager will examine the changes, and if approved, will attach a signed document found in Appendix 1 to a copy of the updated SRS.
* If changes are dictated by the customer, whether verbally or written, the Project manager will create an updated copy of the SRS and attach a signed document found in Appendix 1.
* The project manager will present the updated SRS to the customer for final approval, upon which the customer will also sign the document found in Appendix 1.
* When both the project manager and the customer approve the updates to the SRS, the project manager will merge the official SRS with the updated SRS via Git.
* The project manager completes the procedure by recording the update in the document found in Appendix 2.

**A.** **Appendices**

**A.1 Appendix 1**

Required form for the approval of changes to this SRS document:

**Document Approval**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | William Dewald | Lead Software Engineer |  |
|  | Dr. M.K. Quweider | Instructor, CSCI-3340 |  |

**A.2 Appendix 2**

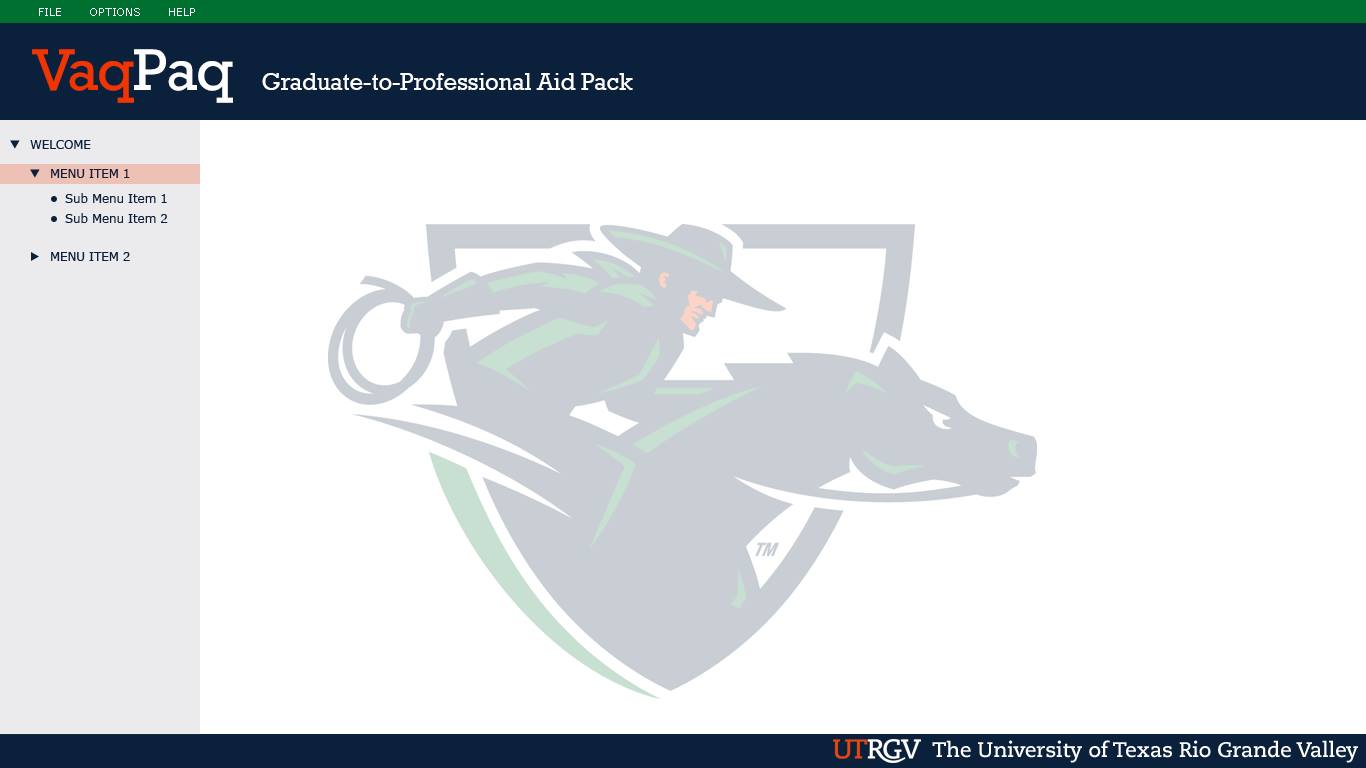
Required form must be attached to the end of this document if there are any changes after its initial completion:

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

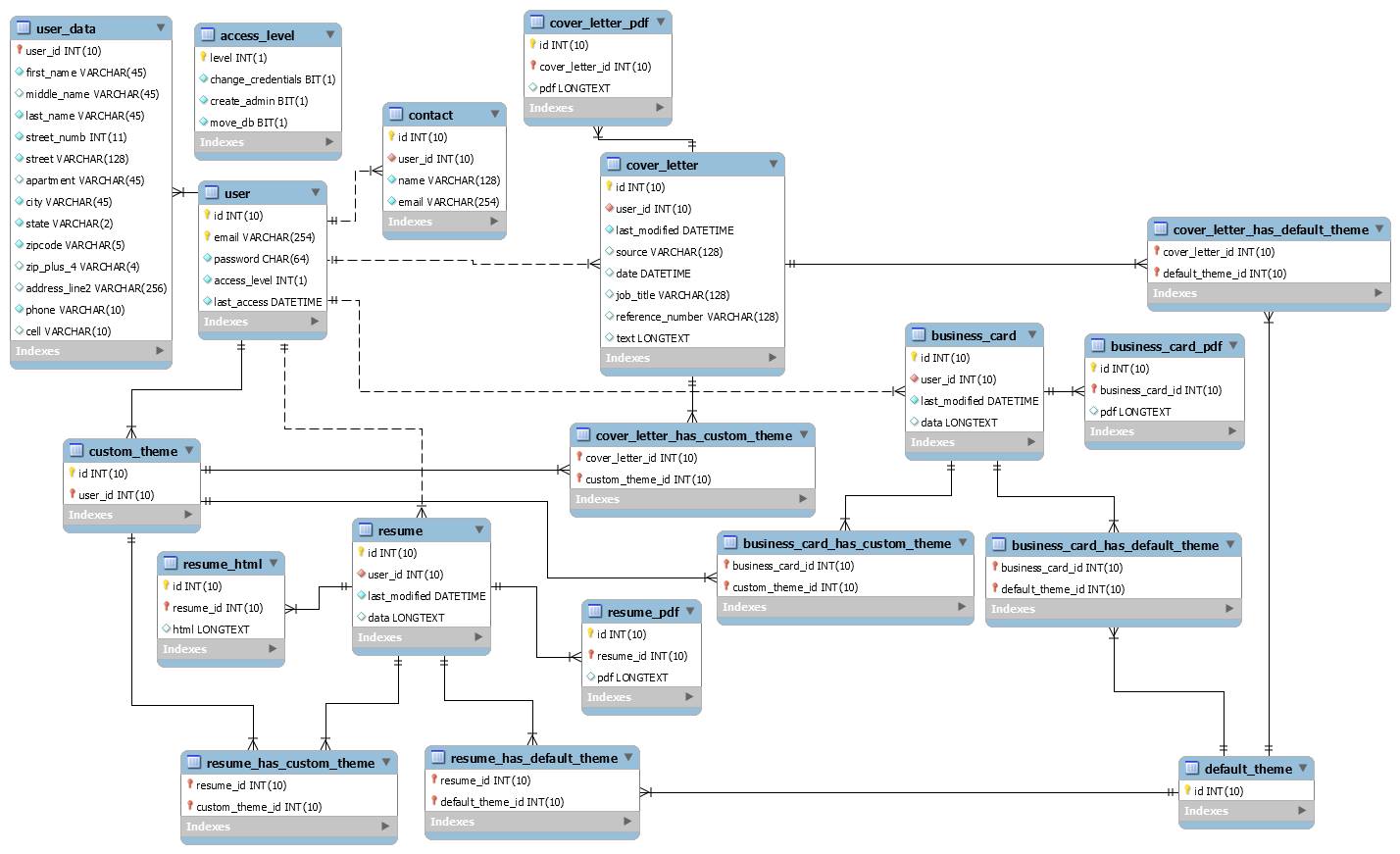
**A.3 Appendix 3**

The following is a preliminary GUI design concept and is not to be officially considered as part of the requirements.

****

**A.4 Appendix 4**

The following is a preliminary database design concept and is not to be officially considered as part of the requirements.

****

**A.5 Appendix 5**

The following are the coding standards used by the VaqPack development team. Following these standards is a self-imposed requirement agreed upon by the team members.

CodingStandards: This file is subject to change throughout the course of the project. If there needs to be a change or something added, contact the project manager so that we can discuss it.

###################################################################################

JDK: Java: 1.8.0\_60

If VP\_GUIEvents.java does not recognize

import javafx.scene.control.Alert;

import javafx.scene.control.Alert.AlertType;

then you have to update your runtime environment and set NetBeans to use it.

###################################################################################

FORMATTING:

We originally discussed the way to handle brackets with if statements. However, this will cause a mess in NetBeans since be default, braces begin to the right of the predicate and close below the statement.

To avoid any problems, before saving and especially before commits in Git, do Ctrl+A to select all of the code in the .java file and then do Alt+Shift+F

This will not only place brackets where they should be according to NetBeans, it will format just about everything such as spaces on each side of an operator.

###################################################################################

UNARY OPERATORS:

We have agreed to avoid using pre increment and pre decrement operators

DO NOT DO

++i

DO

i++

This should be avoided in for loops as well.

###################################################################################

COMMENTS:

We have not discussed this together yet... but let's agree to always place a "flower box" above every subclass and function with the following formats:

/\*------------------------------------------------------------------------\*

\* functionName()

\* - Description.

\* - parameter 1 description.

\* - parameter 2 description.

\* - What it returns, if anything, and why

\*------------------------------------------------------------------------\*/

The () will indicate that it is a function. Since we will describe the parameters, there is no need to indicate them along with the name. It is one less thing to update if the parameters change.

The following is a flower box for a subclass

/\*------------------------------------------------------------------------\*

\* Name Subclass

\* - Description

\*------------------------------------------------------------------------\*/

Most of our subclasses will be the action events with an overriden handle() function.

Others might be classes inheriting from some abstract super or interface.

Unless the code is complex, we should not have to comment with a description of every variable... provided that every variable has a meaningful name.

###################################################################################

ERROR HANDLING:

Also, let's agree to never let a possible exception go uncaught.

If it is something that the user should know, or if it is an error that is critical and will cause the program to fail, an alert should be displayed first.

Example from VP\_GUIEvents.java:

int test = 0;

try {

int test = Integer.parseInt("not an integer");

} catch (Exception e) {

errorAlert(1100, e.toString());

}

If the error comes from a different class file, such as the database manager class, then an example would be:

int test = 0;

try {

int test = Integer.parseInt("not an integer");

} catch (Exception e) {

GE.errorAlert(1200, e.toString());

}

errorAlert() is a function in the VP\_GUIEvents class. It initializes a VP\_ErrorHandler object with an error code and the exception as a string. VP\_ErrorHandler maintains a switch for the 4-digit error codes and sets the content for the alert box and decides whether it is critical or not. For debugging purposes, the first two digits of the code should correspond to a .java class file. As an example (not necessarily the classes we are using):

11xx from VP\_GUIEvents

12xx from VP\_DatabaseManager

13xx from VP\_DataManager

14xx from VP\_DataToHTML

15xx from VP\_HTMLToPDF

16xx from VP\_FileManager

17xx from VP\_TemplateManager

18xx from VP\_NetworkManager

There are no codes 10xx handled by the error handler because this would be from VP\_GUIBuild. The error handler cannot handle exceptions that might come before the GUI is constructed. If there is something that should be reported before the GUI is completely built, the error should print to the console with error code 10xx.

If there is an error that should just be ignored, catch the exception and print it to the console, but allow the program to continue execution. This is needed sometimes. An example is when you traverse directories. In windows, some directories visible by java are actually junctions. isHidden() will return false, isDirectory() will return true, and canRead() will return true... however, when you have java attempt to list the files within the directory, it will cause an exception. Since a try and catch is the only way to test for junctions, the exception must simply be ignored so that the program can continue on.

###################################################################################

INSTANTIATION AND INITIALIZATION

All instantiation and initialization should be at the top of functions. An exception to this is when you need to create only temporary variables and objects used within an inner block of code, such as a loop.

###################################################################################

CLASS FILE ORGANIZATION

members

constructors

protected functions

private functions

subclasses

setters and getters