**Software Design Documentation**

**VaqPack**

Graduate-to-Professional

Aid Pack

Version 1.0

November 22nd, 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

**2. REFERENCES**…………………………………………………………………**3**

**3. DETAILED DESIGN**………………………………………………………..…4

3.1 MODULE DETAILED DESIGN.................................................................................4

3.1.01: VP\_BusinessCard.java.......................................................................................4

3.1.02: VP\_Button.java...................................................................................................5

3.1.03: VP\_Center.java...................................................................................................6

3.1.04: VP\_Contact.java................................................................................................9

3.1.05: VP\_CoverLetter.java......................................................................................9

3.1.06: VP\_DataManager.java...................................................................................10

3.1.07: VP\_DataToHtml.java.....................................................................................12

3.1.08: VP\_DatabaseManager.java...............................................................................12

3.1.09: VP\_DivisionLine.java...................................................................................15

3.1.10: VP\_ErrorHandler.java......................................................................................15

3.1.11: VP\_FieldLabel.java............................................................................................16

3.1.12: VP\_FileManager.java.......................................................................................16

3.1.13 VP\_Footer.java....................................................................................................17

3.1.14: VP\_GUIController.java.....................................................................................17

3.1.15: VP\_Header.java..................................................................................................20

3.1.16: VP\_HtmlToPdf.java...........................................................................................20

3.1.17: VP\_Loader.java................................................................................................21

3.1.18: VP\_Mail.java......................................................................................................21

3.1.19: VP\_PageDivision.java......................................................................................22

3.1.20: VP\_Paragraph.java............................................................................................22

3.1.21: VP\_PasswordField.java....................................................................................23

3.1.22: VP\_Resume.java...............................................................................................24

3.1.23: VP\_Sounds.java.................................................................................................24

3.1.24: VP\_TextField.java..............................................................................................25

3.1.25: VP\_Theme.java..................................................................................................25

3.1.26: VP\_Tree.java......................................................................................................25

3.1.27: VP\_User.java......................................................................................................26

3.1.28: VaqPack.java.....................................................................................................27

3.2 DATA DETAILED DESIGN…………………………………...……………..…….28

3.2.01 User Access Level...................................................................................................28

3.2.02 Business Card Information....................................................................................29

3.2.03 Business Card PDF.................................................................................................29

3.2.04 Contact Information................................................................................................29

3.2.05 Cover Letter Information........................................................................................29

3.2.06 Cover Letter PDF.........................................................................................30

3.2.07 Custom Theme.......................................................................................................30

3.2.08 Registering User...................................................................................................30

3.2.09 Reset Code...........................................................................................................30

3.2.10 Resume Information............................................................................................31

3.2.11 Resume HTML...................................................................................................31

3.2.12 Resume PDF.........................................................................................................31

3.2.13 User.......................................................................................................................31

3.2.14 User Data..............................................................................................................32

**4. USER INTERFACES BY USE CASE**………………………………………33

4.01 Case01-Initial Administrator System Configuration...........................................35

4.02 Case02-Administrator System Configuration Password Mismatch....................36

4.03 Case03-User Creates New Account....................................................................37

4.04 Case04-User Login...........................................................................................38

4.05 Case05-Update Personal Information..............................................................39

4.06 Case06-Update Personal Information-Required field......................................40

4.07 Case07-Update Personal Information-Incorrect format...................................41

4.08 Case08-User Forgets Password.......................................................................42

4.09 Case09-New Password Not Strong Enough....................................................44

4.10 Case10-Inactivity Timeout……………….......................................................45

**A. APPENDICES**

A.1 Appendix 1........................................................................................................

A.2 Appendix 2........................................................................................................

**1. Introduction**

**1.1 Purpose**

This Software Design Document is made with the purpose of explicitly outlining the software architecture and high level design of the VaqPack Graduate to Professional Aid Pack application. Through the use of various architectural decompositions, the various modules, data structures, databases and interfaces will be depicted. The intention of this document is to provide developers an insight into meeting our Client’s needs efficiently and effectively, corresponding with the requirements set forth in the SRS. Therefore, this document is mainly intended for the developers, present company as well as future organizations.

**1.2 Scope**

The software application described throughout this SDD is the VaqPack Graduate to Professional Aid Pack, or simply VaqPack. This SDD is intended for a base level system in order to provide a proof of concept for the use of building an evolutionary prototype that demonstrates the functionality specified by the corresponding SRS. This will be achieved through the use of use-case models, state models, class models and data flow models that will serve to clarify the development teams thought process during implementation.

**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** |
| SDD | Software Design Description; A written description of a software product, that a software designer writes in order to give a software development team overall guidance to the architecture of the software project. |
| SRS | Software Requirement Specification; A comprehensive description of the intended purpose and environment for software under development. The SRS fully describes what the software will do and how it will be expected to perform. |
| 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. |
| JDO | Java Data Object; A standard way to access persistent data in databases, using plain old Java objects (POJO) to represent persistent data. |
| POJO | Plain Old Java Object; An ordinary Java object, not bound by any special restriction. |
| IDE | Integrated Development Environment; software that provides tools for the development and organization of programming code. |
| JVM | Java Virtual Machine; Provides the necessary links allowing a java program to run on a machine using a particular operating system. |
| JRE | Java Runtime Environment; Including the Java Virtual Machine, all necessary components for a system to establish the environment in which Java programs will run. |
| DBA | Directs or performs all activities related to maintaining a successful database environment. |
| SQL | Structured Query Language; the standard relational database query language. |
| JDBC | Java Database Connectivity; a Java API developed by Oracle Corporation which provides methods for querying and updated a database. |
| XML | Extensible Markup Language; a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. |
| XSL | eXtensible Stylesheet Language; a style sheet language for XML documents. |
| HTML | HyperText Markup Language; the web standard language used in the delivery of online content, interpreted and rendered by web browsers. |
| PDF | Portable Document Format; a popular electronic document file type particularly used with rich-text or styled text. |

**2.** **References**

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

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

HTML - <http://www.w3schools.com/html/>

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/>

**3.** **Detailed Design**

**3.1 Module Detailed Design**

****

**3.1.01: VP\_BusinessCard.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_BusinessCard | |
| **Brief description:** The VP\_BusinessCard class is responsible for building the business card object. | |
| **Attributes (fields)** | **Attribute Description** |
| Integer themeId; | This is a variable used to identify the theme to be generated by generateXLS(). |
| VP\_User owner; | This is a declaration of a VP\_User object used by the VP\_BusinessCard class constructor. |
| StringProperty profession; | This is a declaration of a StringProperty object to be used throughout the class. |
| StringProperty companyName; | This is a declaration of a StringProperty object to be used throughout the class. |
| StringProperty companySlogan; | This is a declaration of a StringProperty object to be used throughout the class. |
| StringProperty webPage; | This is a declaration of a StringProperty object to be used throughout the class. |
| String xsl; | This is a variable used to store the XML string produced by generateXLS(). |
| String professionStored; | This is a variable used to store the value of the profession StringProperty. |
| String companyNameStored; | This is a variable used to store the value of the companyName StringProperty. |
| String companySloganStored; | This is a variable used to store the value of the companySlogan StringProperty. |
| String webPageStored; | This is a variable used to store the value of the webPage StringProperty. |
| Boolean startedBusinessCard; | This is a variable used to track if the business card process has been started. |
| Boolean completedBusinessCard; | This is a variable used to track if the business card process has been finished. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_BusinessCard() | This is a constructor method. |
| revert() | This a method which reverts bound properties to the last stored value. |
| save() | This is a method which stores \*Stored attribute values to the current bound properties. |
| clear() | This is a method which clears all data in the business card for the next user. |
| generateXLS() | This is a method which generates the XML stylesheet to be passed to html conversion. |

**3.1.02: VP\_Button.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Button | |
| **Brief description:** The VP\_Button class is a custom button class. | |
| **Attributes (fields)** | **Attribute Description** |
| No Attributes |  |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Button() | This is a default constructor method. |
| VP\_Button(String) | This is a constructor method which accepts a String parameter for the label of the button |
| VP\_Button(String, EventHandler) | This is a constructor method which accepts a String parameter for the label of the button and an EventHandler for the action event of the button. |

**3.1.03: VP\_Center.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Center | |
| **Brief description:** The VP\_Center class is responsible for generating and managing the center view of the GUI. | |
| **Attributes (fields)** | **Attribute Description** |
| VP\_GUIController controller; | This is a declaration of a VP\_GUIController object to be used throughout the class. |
| VP\_TextField loginEmail; | This is a declaration of a VP\_TextField object to be used throughout the class. |
| VP\_TextField resetEmail; | This is a declaration of a VP\_TextField object to be used throughout the class. |
| VP\_TextField regLoginAccess; | This is a declaration of a VP\_TextField object to be used throughout the class. |
| VP\_TextField resetCode; | This is a declaration of a VP\_TextField object to be used throughout the class. |
| VP\_TextField registerEmail; | This is a declaration of a VP\_TextField object to be used throughout the class. |
| Label resetPassStrengthLabel; | This is a declaration of a Label object to be used throughout the class. |
| Label registerPassStrengthLabel; | This is a declaration of a Label object to be used throughout the class. |
| Label infoProgress; | This is a declaration of a Label object to be used throughout the class. |
| Label resumeProgress; | This is a declaration of a Label object to be used throughout the class. |
| Label bcardProgress; | This is a declaration of a Label object to be used throughout the class. |
| Label covletProgress; | This is a declaration of a Label object to be used throughout the class. |
| VP\_PasswordField loginPass; | This is a declaration of a VP\_PasswordField object to be used throughout the class. |
| VP\_PasswordField resetNewPass; | This is a declaration of a VP\_PasswordField object to be used throughout the class. |
| VP\_PasswordField resetNewPassConfirm; | This is a declaration of a VP\_PasswordField object to be used throughout the class. |
| VP\_PasswordField registerPass; | This is a declaration of a VP\_PasswordField object to be used throughout the class. |
| VP\_PasswordField registerPassConfirm; | This is a declaration of a VP\_PasswordField object to be used throughout the class. |
| VP\_DivisionLine loginErrorLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine accessInstructionsLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine loginButtonLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine accessLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine resetErrorLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine resetInstructions1Line; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine resetInstructions2Line; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine resetNewPassLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine resetNewPassConfirmLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine resetCodeLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine resetButLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine resetPassStrengthLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine registerErrorLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine personalInfoErrorLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_DivisionLine bcardErrorLine; | This is a declaration of a VP\_DivisionLine object to be used throughout the class. |
| VP\_Paragraph loginError; | This is a declaration of a VP\_Paragraph object to be used throughout the class. |
| VP\_Paragraph accessInstructions; | This is a declaration of a VP\_Paragraph object to be used throughout the class. |
| VP\_Paragraph resetError; | This is a declaration of a VP\_Paragraph object to be used throughout the class. |
| VP\_Paragraph resetInstructions1; | This is a declaration of a VP\_Paragraph object to be used throughout the class. |
| VP\_Paragraph resetInstructions2; | This is a declaration of a VP\_Paragraph object to be used throughout the class. |
| VP\_Paragraph registerError; | This is a declaration of a VP\_Paragraph object to be used throughout the class. |
| VP\_Paragraph overviewInfo; | This is a declaration of a VP\_Paragraph object to be used throughout the class. |
| VP\_Paragraph personalInfoError; | This is a declaration of a VP\_Paragraph object to be used throughout the class. |
| VP\_Paragraph bcardError; | This is a declaration of a VP\_Paragraph object to be used throughout the class. |
| VP\_Button submitResetBtn; | This is a declaration of a VP\_Button object to be used throughout the class. |
| ArrayList<VP\_Button> wizardMainButtons; | This is a variable used to collect VP\_Button objects in an array list. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Center(VP\_GUIController) | This is a constructor method which takes a VP\_GUIController object as a parameter |
| build() | This a method which builds the GUI center pane. |
| showScreen() | This is a method which makes all center stackpane levels invisible except for the desired level. |
| buildLoginScreen() | This is a method which creates the user login screen. |
| buildResetPasswordScreen() | This is a method which creates the reset password screen. |
| buildRegisterationScreen() | This is a method which creates the user registration screen. |
| buildOverviewScreen() | This is a method which builds the main application screen. |
| buildPersonalInfoScreen() | This is a method which builds the screen which allows the user to enter their personal information. |
| buildBusinessCardScreen() | This is a method which builds business card edit screen. |
| buildCoverLettersStartScreen() | This is a method which allows the creation of new cover letters and display existing cover letter. |
| buildCoverLettersEditScreen() | This is a method which builds the cover letter edit screen. |
| buildThemesStartScreen() | This is a method which builds the document theme manager. |
| buildThemesEditScreen() | This is a method which builds the theme edit screen. |
| buildDistributeScreen() | This is a method which builds the screen where the user selects which documents to send as attachments to a selected contact. |
| buildResumeStartScreen() | This is a method which builds the screen displaying the completion status of the various resume section. |
| resetLoginRegForms() | This is a method which restores the login page to its original state. |
| resetResetPasswordForms() | This is a method which restores the reset password page to its original state. |
| resetRegisterForms() | This is a method which restores the registration page back to its original state. |
| updateOverview() | This is a method which updates the overview page based on what the user has completed. |

**3.1.04: VP\_Contact.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Contact | |
| **Brief description:** The VP\_Contact class is responsible for building the user contact object(s). | |
| **Attributes (fields)** | **Attribute Description** |
| No Attributes |  |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Contact(String) | This is a constructor method which accepts a String parameter for the email of the contact. |

**3.1.05: VP\_CoverLetter.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_CoverLetter | |
| **Brief description:** The VP\_CoverLetter class is responsible for building the user cover letter object(s). | |
| **Attributes (fields)** | **Attribute Description** |
| Integer themeId; | This is a variable used to identify the theme to be generated by generateXLS(). |
| VP\_User owner; | This is a declaration of a VP\_User object used by the VP\_BusinessCard class constructor. |
| Boolean startedCoverLetter; | This is a variable used to track if the cover letter process has been started. |
| Boolean completedCoverLetter; | This is a variable used to track if the cover letter process has been finished. |
| String xsl; | This is a variable used to store the XML string produced by generateXLS(). |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_CoverLetter(VP\_User) | This is a constructor method which takes a VP\_CoverLetter object as a parameter. |
| revert() | This is a method which reverts bound properties to the last stored value. |
| save() | This is a method which sets the current bound properties to the stored values. |
| clear() | This is a method which clears all data in the cover letter for the next user. |
| generateXLS() | This is a method which generates the XML stylesheet to be passed to html conversion. |

**3.1.06: VP\_DataManager.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_DataManager | |
| **Brief description:** The VP\_DataManager class is responsible for storing the XML nodes defined or loaded by the user. | |
| **Attributes (fields)** | **Attribute Description** |
| VP\_GUIController controller; | This is a declaration of a VP\_GUIController object to be used throughout the class. |
| VP\_DatabaseManager dbManager; | This is a declaration of a VP\_DatabaseManager object to be used throughout the class. |
| VP\_FileManager fileM; | This is a declaration of a VP\_FileManager object to be used throughout the class. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_DataManager(VP\_GUIController) | This is a constructor method which takes a VP\_GUIController object as a parameter. |
| checkDBLocation() | This is a method which calls the checkDBRunning() method of the VP\_DatabaseManager object. |
| retrieveDBLocation() | This is a method which retrieves the database server url and port from the properties file via the VP\_FileManager object. |
| storeDBLocation() | This is a method which sends the database url and port to the VP\_DataBaseManager and VP\_FileManager objects for storage in the properties file. |
| getCurrentLocation() | This is a method which retrieves the current database url and port from the VP\_DatabaseManager object and returns a string array. |
| retrieveAdmin() | This is a method which retrieves the database admin credentials from the properties file and sends them to the VP\_DatabaseManager object. |
| storeAdminCred() | This is a method which sends the database admin credentials to the VP\_FileManager object and the VP\_DatabaseManager object. |
| checkDBSchemaExists() | This is a method which calls the checkSchema() method of the VP\_DatabaseManager object. |
| checkDBTable(Integer) | This is a method which checks a specific database table based on the parameter provided. |
| constructUserAccess() | This is a method which calls the buildAccessLevels() method of the VP\_DatabaseManager object. |
| searchforVPAdmin() | This is a method which determines if a VaqPack admin has been created via the findAdminUser() method of the VP\_DatabaseManger object. Returns a Boolean indicating if a VaqPack admin has been created in the database user table. |
| createVPAdmin(String[]) | This is a method which calls the createVaqPackAdmin() method of the VP\_DatabaseManager object with a string array of database admin and VaqPack admin user credentials. Returns a Boolean indicating the validity of the database admin user. |
| userLogin(String[]) | This is a method which sends user credentials to the VP\_DatabaseManager object via the string array parameter. Returns an integer indicated the attempted user login status. |
| verifyRegAccess(String[]) | This is a method which sends a string array of user credentials to the VP\_DatabaseManager object and returns a Boolean indicated the validity of the credentials. |
| resendAccess(String[]) | This is a method which sends a string array of user credentials to the VP\_DatabaseManager object. A new access code is sent is the user email is valid. |
| findUserForReset(String) | This is a method which searches the user database by email via the string parameter. Returns an integer indicating a user status. |
| resetPass(String[]) | This is a method which calls the resetPassword() method of the VP\_DatabaseManager object. Returns an integer indicating the status of the reset. |
| regUser(String[]) | This is a method which sends user credentials to the VP\_DatabaseManager object via the string array parameter. Returns an integer indicating the user registration status. |
| checkEmail(String) | This is a method which checks for a well-formed email address via the string parameter and a Regex pattern. Returns a Boolean indicating the email’s validity. |
| saveUserData() | This is a method which calls the storeUserData() method of the VP\_DatabaseManager object. |
| saveBCardData() | This is a method which calls the storeBCardData() method of the VP\_FileManager object. |
| generateAccessCode() | This is a method which generates a 16-character string access code for email verification. Returns a string. |
| hashPassword(String) | This is a method which hashes a password via the string parameter using SHA-256. Returns the hashed string. |

**3.1.07: VP\_DataToHtml.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_DataToHtml | |
| **Brief description:** The VP\_DataToHtml class is responsible for converting the XML data nodes into HTML. | |
| **Attributes (fields)** | **Attribute Description** |
| No Attributes |  |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_DataToHtml() | This is a private, static constructor that cannot be instantiated. |
| convert( File xmlFile, File xslFile, File htmlFile) | This is a static method which converts XML and XLS files to HTML via File object parameters. |

**3.1.08: VP\_DatabaseManager.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_DatabaseManager | |
| **Brief description:** The VP\_DataManager class is responsible for handling connections and queries to a database. | |
| **Attributes (fields)** | **Attribute Description** |
| Connection con; | This is a variable used to store a connection session with a specific database. |
| Statement stm; | This is a variable used to store a static SQL statement. |
| ResultSet rts; | This is variable used to store a table of data representing a database result set. |
| String dbName; | This is a variable used to store a database name. |
| String url; | This is a variable used to store the url string of a database. |
| String port; | This is a variable used to store the port value of a database connection as a string. |
| String fullURL; | This is a variable used to store the full url string of a database connection. |
| String adminUserName; | This is a variable used to store a database admin user name. |
| String adminPassword; | This is a variable used to store a database admin password. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_DatabaseManager() | This is a constructor method which initializes the connection, statement and result set. |
| checkDBRunning() | This is a method which verifies that a MySQL database is running on the provided url and port. |
| checkSchema() | This is a method which creates the vaqpack\_db schema. |
| checkUserTable() | This is a method which checks the database user table via a defined SQL statement. |
| checkAccessLevelTable() | This is a method which checks the ‘access\_level’ database table via a defined SQL statement. |
| checkRegisteringUserTable() | This is a method which checks the ‘registering\_user’ database table via a defined SQL statement. |
| checkResetCodeTable() | This is a method which checks the ‘reset\_code’ table via a defined SQL statement. |
| checkBusinessCardTable() | This is a method which checks the ‘business\_card’ database table via a defined SQL statement. |
| checkContactTable() | This is a method which checks the ‘contact’ database table via a defined SQL statement. |
| checkCoverLetterTable() | This is a method which checks the ‘cover\_letter database table via a defined SQL statement. |
| checkResumeTable() | This is a method which checks the ‘resume’ database table via a defined SQL statement. |
| checkUserDataTable() | This is a method which checks the ‘user\_data’ database table via a defined SQL statement. |
| checkCustomThemeTable() | This is a method which checks the ‘custom\_theme’ database table via a defined SQL statement. |
| checkBusinessCardPDFTable() | This is a method which checks the ‘business\_card\_pdf’ database table via a defined SQL statement. |
| checkCoverLetterPDFTable() | This is a method which checks the ‘cover\_letter\_pdf’ database table via a defined SQL statement. |
| checkResPDFTable() | This is a method which checks the ‘resume\_pdf’ database table via a defined SQL statement. |
| checkResHTMLTable() | This is a method which checks the ‘resume\_table’ database table via a defined SQL statement. |
| findAdminUser() | This is a method which attempts to find at least one VaqPack database admin user, removes dead registration codes and dead reset codes. Returns a Boolean indicating the existence of a VaqPack admin. |
| createVaqPackAdmin(String[] cred, String code, String hashed) | This is a method which creates an admin VaqPack user using credentials stored in the string array parameter. Returns a Boolean indicating if the database admin user is valid. |
| buildAccessLevels() | This is a method which verifies the integrity of the ‘access\_level’ tables. |
| attemptUserLogin(String[], VP\_User) | This is a method which attempts to login the user if the user is found in the database. Returns an integer value indicating status of the user attempting login. |
| verifyUserAccessCode(String[]) | This is a method which verifies a user access code and completes user registration with a valid code. Credentials are stored in the string array parameter. Returns a Boolean the validity of the user’s registration. |
| resetPassword(String[]) | This is a method which attempts to reset the user’s password. Credentials are stored in the string array parameter. Returns an integer indicating the status of the reset. |
| findUserOrRegUserForReset(String email, String code) | This is a method which verifies the email passed via string parameter exists. The code string parameter verifies the validity of the user request. Returns an integer indicating the status of the reset request. |
| resendUserAccessCode(String[], String) | This is a method which stores a new registration code for a user if provided credentials are valid. Returns a Boolean indicating if the user exists. |
| registerUser(String[], String) | This is a method which stores a registration code for a new user. Returns an integer indicating if the user is new or not. |
| storeUserData(VP\_User) | This is a method which stores a user’s personal information via the VP\_User object parameter. |
| storeBCardData(VP\_User, File) | This is a method which stores business card data and associated pdf file. |
| connect(String) | This is a method which creates a database connection and initializes the statement. String parameter indicates the database to append to the connection url. |
| close() | This is a method which closes the result set, statement and connection for a database. |
| checkTable(String) | This is a method which connects to a database, executes the create table update and closes the connection. String parameter is the SQL statement. |

**3.1.09: VP\_DivisionLine.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_DivisionLine | |
| **Brief description:** The VP\_DivisionLine class extends HBox and represents a line of a page division. | |
| **Attributes (fields)** | **Attribute Description** |
| No Attributes |  |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_DivisionLine() | This is a default constructor. |
| VP\_DivisionLine(Node[]) | This is a constructor which allows the addition of nodes via the Nodes array parameter. |
| hide() | This is a method which hides and collapses the VP\_DivisionLine object. |
| show() | This is a method which shows and expands the VP\_DivisionLine object. |

**3.1.10: VP\_ErrorHandler.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_ErrorHandler | |
| **Brief description:** The VP\_ErrorHandler class reports the information related to an error by its code. | |
| **Attributes (fields)** | **Attribute Description** |
| String header; | This is as variable which stores the header text of the error message box. |
| String content; | This is a variable which stores the content of the error message box. |
| Boolean critical; | This is a variable which stores a flag indicating an critical error. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_ErrorHandler(Integer, String) | This is a constructor method. Integer parameter defines an error code, String parameter provides additional exception error information. |
| clearError() | This is a method which clears all object properties. |

**3.1.11: VP\_FieldLabel.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_FieldLabel | |
| **Brief description:** The VP\_FieldLabel class extends Label and provides a custom label class. | |
| **Attributes (fields)** | **Attribute Description** |
| No Attributes |  |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_FieldLabel() | This is a default constructor. |
| VP\_FieldLabel(String) | This is a constructor which defines a Label object text property via the string parameter. |
| VP\_FieldLabel(String, double) | This is a constructor which defines a Label object text and width properties via the string and double parameters. |

**3.1.12: VP\_FileManager.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_FileManager | |
| **Brief description:** The VP\_FileManager class manages reading and writing files. | |
| **Attributes (fields)** | **Attribute Description** |
| String keyPass; | This is a variable which stores a string used for encryption. |
| String vector; | This is a variable which stores a string used for encryption. |
| SecretKey key; | This is a variable used to store a SecretKey object used for encryption. |
| IvParameterSpec ivps; | This is a variable used to store a IvParameterSpec object used for encryption. |
| DocumentBuilderFactory icFactory; | This is a variable used to store a DocumentBuilderFactory object which is used to parse XML documents. |
| DocumentBuilder icBuilder; | This is a variable used to store a DocumentBuilder object for XML document processing. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_FileManager() | This is a constructor which initializes keypass and vector for encryption. |
| retrieveUrlPort() | This is a method which retrieves the database url and port from the properties file. Returns a string array. |
| storeUrlPort(String[]) | This is a method which stores the database url and port to the properties file via string array parameter. |
| retrieveAdminEncrypted(String url, String port) | This is a method which retrieves the encrypted database admin user and password from the properties file. String url parameter is the database url, and string port parameter is the database port. Returns a string array of decrypted admin user credentials. |
| storeAdminUserPass(String[]) | This is a method which stores the encrypted database admin user and password in the properties file. |
| generateBCardPDF(VP\_User) | This is a method which generates a PDF of the stored business card. VP\_User object parameter provides the related contact details. |
| encrypt(String) | This is a method which encrypts a string, provided by the string parameter. Returns an encrypted string. |
| decrypt(String) | This is a method which decrypts a string, provided by the string parameter. Returns a decrypted string. |

**3.1.13 VP\_Footer.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Footer | |
| **Brief description:** The VP\_Footer class extends HBox and controls the footer are of the GUI. | |
| **Attributes (fields)** | **Attribute Description** |
| VP\_GUIController controller; | This is a variable which stores a VP\_GUIController object for use throughout the class. |
| Pane footerLogo; | This is a variable which stores a Pane object to display a logo image in the footer. |
| Label footerCaption; | This is a variable which stores a Label object containing a caption. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Footer(VP\_GUIController) | This is a constructor which adds empty panes to hold footer information. Accepts a VP\_GUIController object as a parameter. |
| build() | This is a method which builds the GUI footer. |

**3.1.14: VP\_GUIController.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_GUIController | |
| **Brief description:** The VP\_GUIController controls events triggered by or altering the VaqPack GUI. | |
| **Attributes (fields)** | **Attribute Description** |
| Scene primaryScene; | This is a variable which stores a Scene object for use throughout the class. |
| String title; | This is a variable which stores a title string for use throughout the class. |
| StackPane mainLayout; | This is a variable which stores a StackPane object for use throughout the class. |
| VP\_Loader loader; | This is a variable which stores a VP\_Loader object for use throughout the class. |
| VP\_DataManager dataM; | This is a variable which stores a VP\_DataManager object for use throughout the class. |
| VP\_Header header; | This is a variable which stores a VP\_Header object for use throughout the class. |
| VP\_Tree leftTree; | This is a variable which stores a VP\_Tree object for use throughout the class. |
| VP\_Center center; | This is a variable which stores a VP\_Center object for use throughout the class. |
| VP\_Footer footer; | This is a variable which stores a VP\_Footer object for use throughout the class. |
| ArrayList<Runnable> guiTasks; | This is a variable which stores an array list of tasks implementing the Runnable interface. |
| ArrayList<Runnable>dbTasks; | This is a variable which stores an array list of tasks implementing the Runnable interface. |
| ArrayList<String> guiTaskLabels; | This is a variable which stores an array list of strings for use as task labels. |
| ArrayList<String> dbTaskLabels; | This is a variable which stores an array list of strings for use as task labels. |
| Integer USER\_PASSWORD\_MINIMUM, | This is a variable which stores an integer constant for the minimum password length. |
| Integer USER\_INACTIVITY\_LIMIT, | This is a variable which stores an integer constant for setting a user inactivity limit. |
| Integer sceneWidth, | This is a variable which stores an integer constant for the GUI scene width. |
| Integer sceneHeight; | This is a variable which stores an integer constant for the GUI scene height. |
| VP\_User currentUser; | This is a variable which stores a VP\_User object for use throughout the class. |
| Alert warnLogOut; | This is a variable which stores a Alert object for use throughout the class. |
| Timer timer; | This is a variable which stores a Timer object for use throughout the class. |
| Integer sessionSeconds; | This is a variable which stores an integer for tracking user inactivity. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_GUIController(Stage) | This a constructor which creates the empty scene and sets the title, creates a VP\_Loader object which is visible while the empty GUI is not visible. Creates a VP\_DataManager object. Calls the load() method. Accepts a Stage object as parameter. |
| errorAlert(Integer, String) | This is a method which loads an error into a new VP\_ErrorHandler object an alert with header and content determined by VP\_ErrorHandler. Integer parameter provides the error code to VP\_ErrorHandler, string parameter provides the error exception. String parameter can be empty. |
| logoutUser() | This is a method which calls logout method on current user to clear bound values and call newUserSet() to reset the GUI. |
| newUserSet() | This is a method which changes the GUI when the current user changes. Tracks inactivity of the logged in user. |
| createShell() | This is a method which creates an empty BorderPane object and related children. Returns the empty BorderPane. |
| load() | This is a method which Configues and/or checks the database and builds the GUI components as background tasks. |
| requestDBLocation(Integer) | This is a method which displays a dialog requesting the database server location. Integer parameter indicates if this dialog has displayed before. Returns a string array of the database server url and port. |
| requestAdminCred(Integer) | This is a method which displays a dialog requesting the database admin user credentials. If the user cancels this, the program exits. Parameter type. Parameter = 0 indicates first time showing this dialog. Parameter = 1 indicates that the dialog is being shown again. Returns a string array of the database admin username and password. |
| requestVPAdmin(Integer) | This is a method which display a dialog requesting the VaqPack admin user credentials. Integer parameter indicates if this dialog has displayed before. Returns a string array of the database admin and VaqPack admin usernames and passwords. |
| exposeGUI() | This is a method which hides the loader and makes the GUI visible. |
|  |  |

**3.1.15: VP\_Header.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Header | |
| **Brief description:** The VP\_Header class extends VBox and controls the header are of the GUI. | |
| **Attributes (fields)** | **Attribute Description** |
| VP\_GUIController controller; | This is a variable which stores a VP\_GUIController object for use throughout the class. |
| Stage primaryStage; | This is a variable which stores a Stage object for use throughout the class. |
| MenuBar menuBar; | This is a variable which stores a MenuBar object for use throughout the class. |
| HBox headerBar; | This is a variable which stores a HBox object for use throughout the class. |
| Pane headerLogo; | This is a variable which stores a Pane object for use throughout the class. |
| Label headerCaption; | This is a variable which stores a Label object for use throughout the class. |
| Menu adminMenu; | This is a variable which stores a Menu object for use throughout the class. |
| MenuItem userLogout; | This is a variable which stores a MenuItem object for use throughout the class. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Header(VP\_GUIController, Stage) | This is a constructor method which adds an empty menubar and header section to itself. Parameters define a VP\_GUIController object and Stage object. |
| build() | This is a method which builds the GUI header and menu. |
|  |  |

**3.1.16: VP\_HtmlToPdf.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_HtmlToPdf | |
| **Brief description:** The VP\_HtmlToPdf class converts HTML documents to PDF. | |
| **Attributes (fields)** | **Attribute Description** |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_HtmlToPdf() | This is a private constructor method which cannot be instantiated. |
| convert(File htmlFile, File pdfFile) | This is a method which converts a html File object to pdf format. Parameters are an html File object and a PDF File object. |

**3.1.17: VP\_Loader.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Loader | |
| **Brief description:** The VP\_Loader class controls the GUI loading screen | |
| **Attributes (fields)** | **Attribute Description** |
| StringProperty loadPercentProp; | This is a variable which stores the percent loading percent message. |
| StringProperty activity1; | This is a variable which stores a StringProperty object for use throughout the class. |
| StringProperty activity2; | This is a variable which stores a StringProperty object for use throughout the class. |
| Label percentLabel; | This is a variable which stores a Label object for use throughout the class. |
| Label activityLabel1; | This is a variable which stores a Label object for use throughout the class. |
| Label activityLabel2; | This is a variable which stores a Label object for use throughout the class. |
| Integer totalTasks; | This is a variable which stores the number of total tasks being run. |
| Integer completedTasks; | This is a variable which stores the number of tasks completed. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Loader(Integer sceneWidth, Integer sceneHeight) | This is a constructor which accepts scene width and height integer values as parameters. |
| incrementCompletedTasks() | This is a method which calculates the percentage of loading completion. |

**3.1.18: VP\_Mail.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Mail | |
| **Brief description:** The VP\_Mail class handles email preparation and sending, extends Thread. | |
| **Attributes (fields)** | **Attribute Description** |
| VP\_GUIController controller; | This is a variable which stores a VP\_GUIController object for use throughout the class. |
| String port; | This is a variable which stores a port number as a string. |
| String host; | This is a variable which stores a host string. |
| String userName; | This is a variable which stores a user name string. |
| String password; | This is a variable which stores a password string. |
| String recipient; | This is a variable which stores an email recipient. |
| String subject; | This is a variable which stores an email subject. |
| String message; | This is a variable which stores an email message. |
| String[] ccEmails; | This is a variable which store string array of email address. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Mail(VP\_GUIController controller, String recipient, String[] ccEmails, String subject, String message) | This is a constructor method. Parameter string recipient is the email address to receieve the message. Parameter string array ccEmails is the list of cc emails. Paramater string subject is the subject or title of the email. Parameter string message is the email's message body. |
| run() | This is a method which overrides the Thread run() method. |

**3.1.19: VP\_PageDivision.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_PageDivision | |
| **Brief description:** The VP\_PageDivision class extends VBox, creates a custom VBox page with a header title. | |
| **Attributes (fields)** | **Attribute Description** |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_PageDivision(String) | This is a constructor. String parameter sets the title of this component. |

**3.1.20: VP\_Paragraph.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Paragraph | |
| **Brief description:** The VP\_Paragraph class extends Label, creates a custom Label to be used as a paragraph within Page Divisions. | |
| **Attributes (fields)** | **Attribute Description** |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Paragraph(String) | This is a default constructor. |
| VP\_Paragraph(String) | This is a constructor. String parameter sets the text for the paragraph. |
| VP\_Paragraph(String, Boolean) | This is a constructor. String parameter sets the text for the paragraph. Boolean parameter indicates an error message and sets the text style to bold red. |
| setParaText(String) | This is a method which sets text and adds an indent to the first line. |

**3.1.21: VP\_PasswordField.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_PasswordField | |
| **Brief description:** The VP\_PasswordField class extends PasswordField and implements EventHandler. Provides a custom PasswordField with build int KeyEvent Handling. | |
| **Attributes (fields)** | **Attribute Description** |
| Integer limit; | This is a variable which stores the integer value defining the character limit of the text field. |
| Integer minimum; | This is a variable which stores the integer value defining the character minimum of the text field. |
| Label strengthLabel; | This is a variable which stores a Lable object to indicate the entered password strength. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_PasswordField(int columns, int limit, int minimum, Label strengthLabel) | This is a constructor. Parameter limit is the allowed character limit for this TextField. A limit of 0 or less means to not limit the text. A minimum of 0 or less means no minimum requirement and no strength checking for this pawwrod field. Parameter strengthLabel is the label that notes the strength if this label is not null. |
| handle() | This is an override of the EventHandler handle() method. Handles the KeyEvent, enforcing the limit, if any. Calculates the strength of the password if minimum > 0 and if the strengthLabel exists. Also clears the red tint that was applied when invalid. |
| showInvalid() | This is a method which applies a red tint to the field when the caller deems it as invalid. |
| showValid() | This is a method which removes the red tint when the caller deems it as valid. |
| assignEvents() | This is a method which applies any event handlers as needed. Done outside of the constructor to avoid potential problems. |
|  |  |

**3.1.22: VP\_Resume.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Resume | |
| **Brief description:** The VP\_Resume class defines the Resume object. | |
| **Attributes (fields)** | **Attribute Description** |
| Integer themeId; | This is a variable which stores an integer value indicating a theme id. Initialized at -1. |
| VP\_User owner; | This is a variable which stores a VP\_User object for use throughout the class. |
| Boolean startedResume; | This is a variable that indicates if a resume has started or not. |
| Boolean completedResume; | This is a variable that indicates if a resume has been completed or not. |
| String xsl; | This is a variable which stores XSL data as a string. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Resume(VP\_User) | This is a constructor. Parameter VP\_User object is the current user that is logged in. |
| revert() | This is a method which reverts bound properties to the last stored value. |
| save() | This is a method which sets the current bound properties to the stored values. |
| clear() | This is a method which clears all data in the resume for the next user. |
| generateXLS() | This is a method which generates an XML stylesheet to be passed to html conversion. |
|  |  |

**3.1.23: VP\_Sounds.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Sounds | |
| **Brief description:** The VP\_Sounds class defines a static object for GUI sound alerts. | |
| **Attributes (fields)** | **Attribute Description** |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Sounds() | This is a private constructor which cannot be instantiated. |
| play(Integer) | This is a method which plays a sound based on the provided integer parameter. Accepted values are -2 for error alert, -1 for general error, 0 for button action. |

**3.1.24: VP\_TextField.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_TextField | |
| **Brief description:** The VP\_TextField class extends TextField and implements EventHandler. It defines a custom TextField class with built in KeyEvent handling. | |
| **Attributes (fields)** | **Attribute Description** |
| Integer limit; | This is a variable which stores a character limit as an integer value. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_TextField(Integer columns, Integer limit) | This is a constructor which accepts a columns integer parameter and a limit integer parameter. The columns parameter sets the column count of the text field, and the limit parameter sets the character limit of the text field. |
| handle(KeyEvent) | This is a method which overrides the EventHandler method. It handles the KeyEvent and enforces any defined character limit. |
| showInvalid() | This is a method which applies a red tint to the field when the caller deems it as invalid. |
| showValid() | This is a method which removes the red tint when the caller deems it valid. |
| assignEvents() | This is a method which applies any event handlers needed. This is done outside of the constructor to avoid potential problem. |

**3.1.25: VP\_Theme.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Theme | |
| **Brief description:** The VP\_Theme class contains the default theme and stores custom themes. | |
| **Attributes (fields)** | **Attribute Description** |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Theme() | This is a private constructor which cannot be instantiated. |

**3.1.26: VP\_Tree.java**

|  |  |
| --- | --- |
| **Class Name:** VP\_Tree | |
| **Brief description:** The VP\_Tree class extends VBox. It defines a custom VBox class and handles tree view in the left pane of the GUI. | |
| **Attributes (fields)** | **Attribute Description** |
| VP\_GUIController controller; | This is a variable which stores a VP\_GUIController object for use throughout the class. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_Tree(VP\_GUIController) | This is a constructor which accepts a VP\_GUIController object as a parameter. |
| build() | This is a method which builds the tree view. Called in a task to build in the background. |

**3.1.27: VP\_User.java**

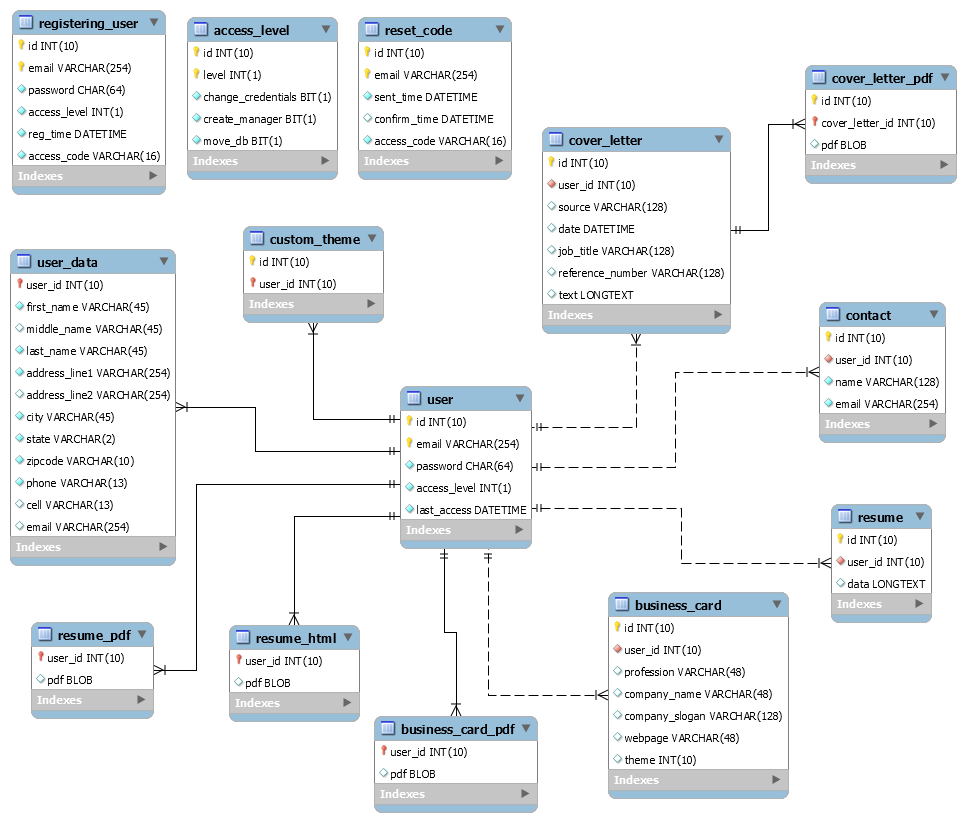
|  |  |
| --- | --- |
| **Class Name:** VP\_User | |
| **Brief description:** The VP\_User class defines the user object. | |
| **Attributes (fields)** | **Attribute Description** |
| VP\_BusinessCard bcard; | This is a variable which stores a VP\_BusinessCard object for use throughout the class. |
| VP\_Resume resume; | This is a variable which stores a VP\_Resume object for use throughout the class. |
| VP\_CoverLetter covlet; | This is a variable which stores a VP\_CoverLetter object for use throughout the class. |
| Integer accessLevel; | This is a variable which stores the access level of the user as an integer value. |
| Integer userID; | This is a variable which stores the user id of the user as an integer value. |
| boolean completedPersonalInfo; | This is a variable which indicates if a user has completed their personal information section. |
| StringProperty email; | This is a StringProperty object which stores an email address. |
| StringProperty firstName; | This is a StringProperty object which stores a first name. |
| StringProperty middleName; | This is a StringProperty object which stores a middle name. |
| StringProperty lastName; | This is a StringProperty object which stores last name. |
| StringProperty address1; | This is a StringProperty object which stores a mailing address. |
| StringProperty address2; | This is a StringProperty object which stores a mailing address continuation. |
| StringProperty city; | This is a StringProperty object which stores a city name. |
| StringProperty state; | This is a StringProperty object which stores a state name. |
| StringProperty zip; | This is a StringProperty object which stores a postal ZIP code. |
| StringProperty phone; | This is a StringProperty object which stores a phone number. |
| StringProperty cell; | This is a StringProperty object which stores a cell phone number. |
| StringProperty docEmail; | This is a StringProperty object which stores an email other than the user’s registered email. |
| String firstNameStored; | This is a string which stores a first name. |
| String middleNameStored; | This is a string which stores a middle name. |
| String lastNameStored; | This is a string which stores last name. |
| String address1Stored; | This is a string which stores a mailing address. |
| String address2Stored; | This is a string which stores a mailing address continuation. |
| String cityStored; | This is a string object which stores a city name. |
| String stateStored; | This is a string object which stores a state name. |
| String zipStored; | This is a string object which stores a postal ZIP code. |
| String phoneStored; | This is a string object which stores a phone number. |
| String cellStored; | This is a string object which stores a cell phone number. |
| String docEmailStored; | This is a string object which stores an email other than the user’s registered email. |
|  |  |
| **Methods (operations)** | **Method Description** |
| VP\_User | This is a constructor. |
| logout() | This is a method which clears all user data for the next user. |
| revert() | This is a method which reverts bound properties to the last stored value. |
| save() | This is a method which sets the current bound properties to the stored values. |

**3.1.28: VaqPack.java**

|  |  |
| --- | --- |
| **Class Name:** VaqPack | |
| **Brief description:** The VaqPack class extends Application and is the main class for the VaqPack application. | |
| **Attributes (fields)** | **Attribute Description** |
|  |  |
| **Methods (operations)** | **Method Description** |
| start(Stage) | This is a method which overrides the Application start method and accepts a Stage object as a parameter. |
| main(String[]) | The applications main method. Accepts launch arguments as a string array. |

**3.2 Data Detailed Design**

**3.2.01 User Access Level**



|  |  |
| --- | --- |
| **Table:** | **access\_level** |
| **Description** | Used to store and define the access level for each unique user. |
| **Attributes** | **Description** |
| id | ID, integer, not null, auto-increment. |
| level | Access Level, integer, not null. |
| change\_credentials | Privilege, bit, not null. |
| create\_manager | Privilege, bit, not null. |
| move\_db | Privilege, bit, not null. |

**3.2.02 Business Card Information**

|  |  |
| --- | --- |
| **Table:** | **business\_card** |
| **Description** | Used to store business card information. |
| **Attributes** | **Description** |
| id | ID, integer, not null, auto-increment. |
| user\_id | User ID, integer, not null. |
| profession | Use to store user profession. |
| company\_name | Use to store user company name. |
| company\_slogan | Use to store user company slogan. |
| webpage | Use to store user webpage address. |
| theme | Integer. Use to apply theme. Default value = -1 |

**3.2.03 Business Card PDF**

|  |  |
| --- | --- |
| **Table:** | **business\_card\_pdf** |
| **Description** | Used to store user business card PDF. |
| **Attributes** | **Description** |
| user\_id | User ID, integer, not null. |
| pdf | Business card PDF blob |

**3.2.04 Contact Information**

|  |  |
| --- | --- |
| **Table:** | **Contact** |
| **Description** | Used to store user contact information. |
| **Attributes** | **Description** |
| id | ID, integer, not null, auto-increment. |
| user\_id | User ID, integer, not null. |
| name | User contact name. |
| email | User contact email. |

**3.2.05 Cover Letter Information**

|  |  |
| --- | --- |
| **Table:** | **cover\_letter** |
| **Description** | Used to store cover letter information. |
| **Attributes** | **Description** |
| id | ID, integer, not null, auto-increment. |
| user\_id | User ID, integer, not null. |
| source |  |
| date | Date. |
| job\_title |  |
| reference\_number |  |
| text | Longtext. |

**3.2.06 Cover Letter PDF**

|  |  |
| --- | --- |
| **Table:** | **cover\_letter\_pdf** |
| **Description** | Used to store cover letter PDF files |
| **Attributes** | **Description** |
| id | ID, integer, not-null, auto-increment. |
| cover\_letter\_id | Cover letter ID, integer, not null. |
| pdf | Cover letter PDF blob. |

**3.2.07 Custom Theme**

|  |  |
| --- | --- |
| **Table:** | **custom\_theme** |
| **Description** | Applies custom theme |
| **Attributes** | **Description** |
| id | ID, integer, not-null, auto-increment. |
| user\_id | User ID, integer, not-null. |

**3.2.08 Registering User**

|  |  |
| --- | --- |
| **Table:** | **registering\_user** |
| **Description** | Used to store user registration data. |
| **Attributes** | **Description** |
| id | ID, integer, not-null, auto-increment. |
| email | User email address for creating account. |
| password | User password. |
| access\_level | User access\_level, default = 0. |
| reg\_time | Current timestamp. |
| access\_code | Access code sent to user upon creating account. |

**3.2.09 Reset Code**

|  |  |
| --- | --- |
| **Table:** | **reset\_code** |
| **Description** | Used to store reset codes. |
| **Attributes** | **Description** |
| id | ID, integer, not-null, auto-increment. |
| email | User email account email address. |
| sent\_time | Current timestamp. |
| confirm\_time | Time user confirms access code. |
| access\_code | Access code sent to user upon resetting password. |

**3.2.10 Resume Information**

|  |  |
| --- | --- |
| **Table:** | **resume** |
| **Description** | Used to store resume information. |
| **Attributes** | **Description** |
| id | ID, integer, not-null, auto-increment. |
| user\_id | User ID. |
| data | Longtext containing user resume information. |

**3.2.11 Resume HTML**

|  |  |
| --- | --- |
| **Table:** | **resume\_html** |
| **Description** | Used to store user resume HTML |
| **Attributes** | **Description** |
| user\_id | User ID |
| html | Resume HTML blob |

**3.2.12 Resume PDF**

|  |  |
| --- | --- |
| **Table:** | **resume\_pdf** |
| **Description** | Used to store user resume PDF |
| **Attributes** | **Description** |
| user\_id | User ID |
| pdf | Resume PDF blob |

**3.2.13 User**

|  |  |
| --- | --- |
| **Table:** | **User** |
| **Description** | Used to store user credentials. |
| **Attributes** | **Description** |
| id | ID, integer, not-null, auto-increment. |
| email | User login email, not null. |
| password | User login password hash, not null. |
| access\_level | Access level for user, default = 0. |
| last\_access | Last time user accessed account, default = current timestamp. |

**3.2.14 User Data**

|  |  |
| --- | --- |
| **Table:** | **user\_data** |
| **Description** | Used to store basic user data. |
| **Attributes** | **Description** |
| user\_id | User ID, not null. |
| first\_name | First name, not null. |
| middle\_name | Middle name, optional. |
| last\_name | Last name, not null. |
| address\_line1 | Current address, not null. |
| address\_line2 | Additional address line, optional. |
| city | City, not null. |
| state | State, not null. |
| zipcode | Zip code, not null. |
| phone | Contact phone number, not null. |
| cell | Cell phone number, optional. |
| email | Email address, optional. |

**4. User Interfaces by Use Case Scenarios**

**4.1**

**Use Case 01: Initial Administrator System Configuration.**

VaqPack shall allow the system administrator to configure and connect to the MySQL DB using a previously established DBA account upon initial installation. Upon creating a VaqPack Administrator account the user shall have access to the special administrator functions that are locked within VaqPack to normal users that allow for the Administrator to govern within the program.

**A Valid Scenario**

1. User runs VaqPack

2. User correctly inputs MySQL URL and PORT#

3. User successfully inputs MySQL admin username and password

4. User successfully creates initial VaqPack administrator account

**Invalid Scenarios**

1. User runs VaqPack

2. User incorrectly inputs MySQL URL or PORT#

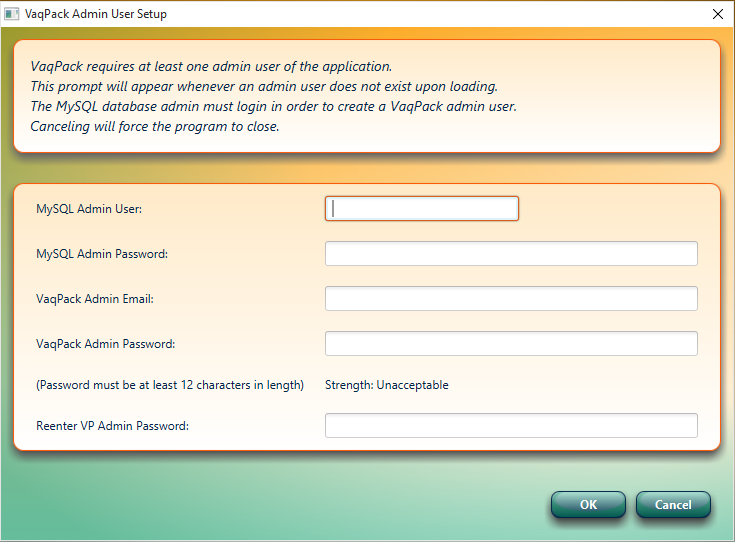
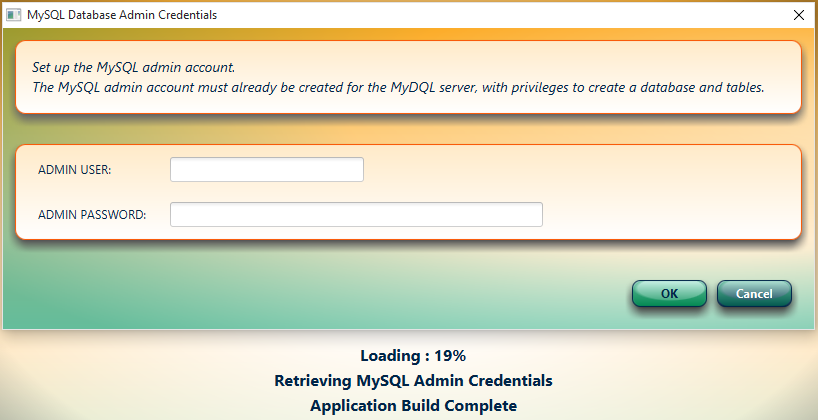
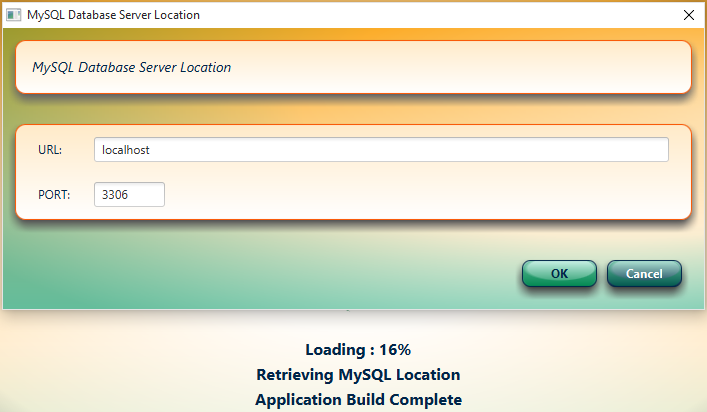
3. VaqPack prompts error

1. User runs VaqPack

2. User correctly inputs MySQL URL and PORT#

3. User unsuccessfully inputs MySQL admin or password

4. VaqPack prompts error

**Case 01 – Initial Administrator System Configuration**

**4.2**

**Use Case 02: Administrator System Configuration Password Mismatch**

**Case Scenario**

1. User runs VaqPack

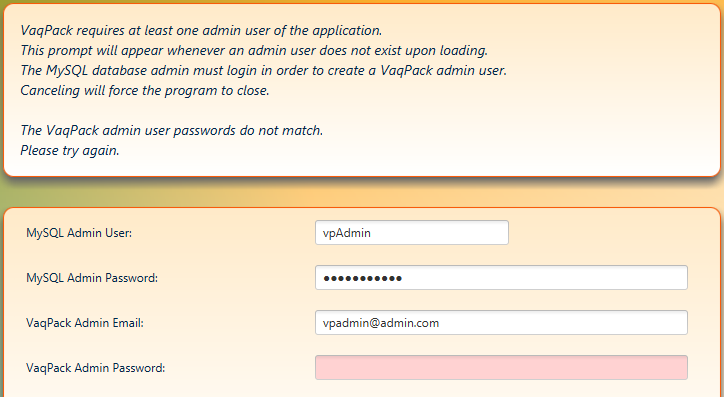
2. User correctly inputs MySQL URL and PORT#

3. User successfully inputs MySQL admin and password

4. User unsuccessfully inputs fields for initial VaqPack administrator account

5. VaqPack prompts error

**Case 02**



**4.3**

**Use Case03: User Creates New Account**

VaqPack shall allow and require users to create an account as stated in the SRS.

**A Valid Scenario**

1. User in login window, clicks on “need an account?”

2. User successfully inputs email and password

3. User retrieves VaqPack system generated code sent to the supplied email

4. User successfully inputs email password and code

5. User logs into VaqPack

**An invalid Scenario**

1. User in login window, clicks on “need an account?”

2. User unsuccessfully inputs email and password

3. VaqPack prompts error

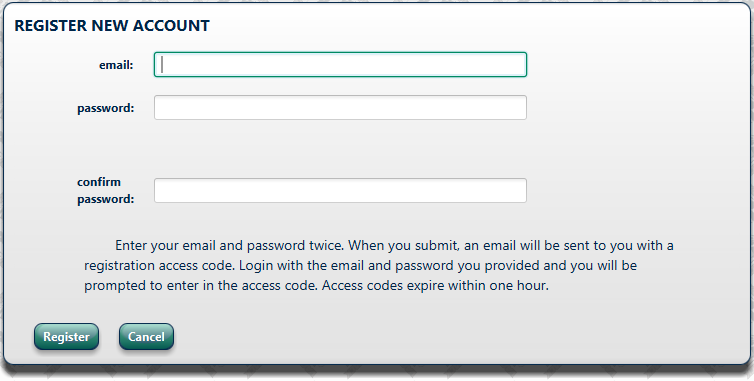
1. User in login window, clicks on “need an account?”

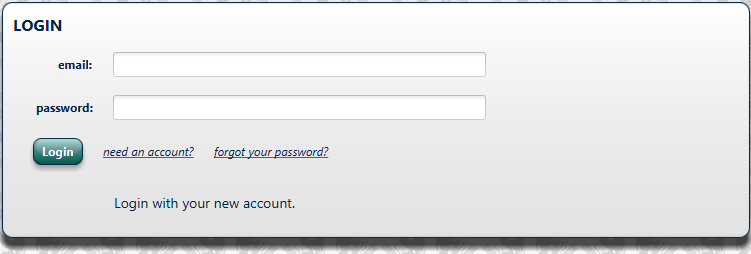
2. User successfully inputs email and password

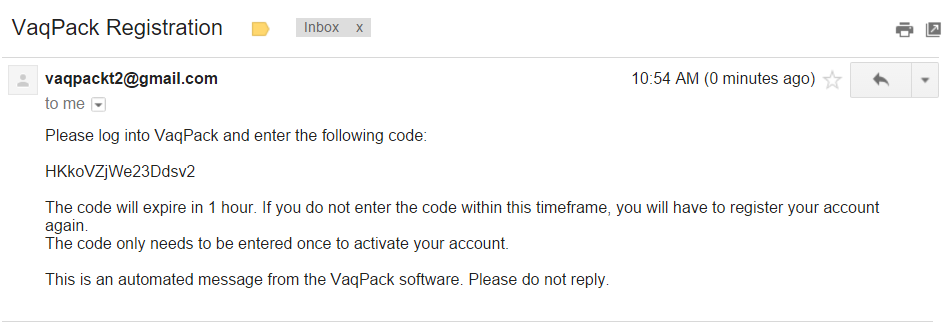
3. User unsuccessfully inputs email password and code

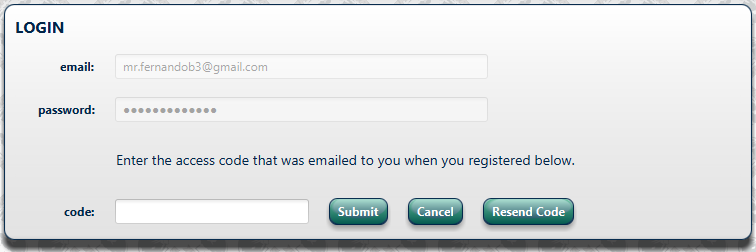
4. VaqPack prompts error

**Case 03 – User Creates New Account Successfully**

****

****

****

****

**4.4**

**Use Case04: User Login**

VaqPack shall allow users that have created an account to log in to gain access to their personal information as stated in the SRS.

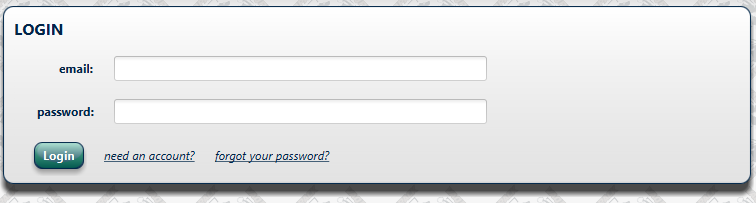
**Case Scenario**

1. User in login window

2. User successfully inputs email and password

3. User logs into VaqPack

**Case 04**

****

**4.5**

**Use Case05: User Updates Personal Information Successfully**

Upon successful log in, VaqPack will require users to Update Personal Information as an initial step.

**Case Scenario**

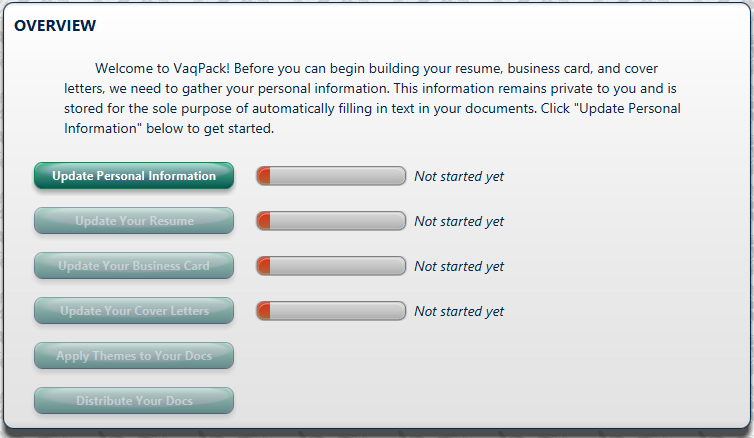
1. User is in the Overview page

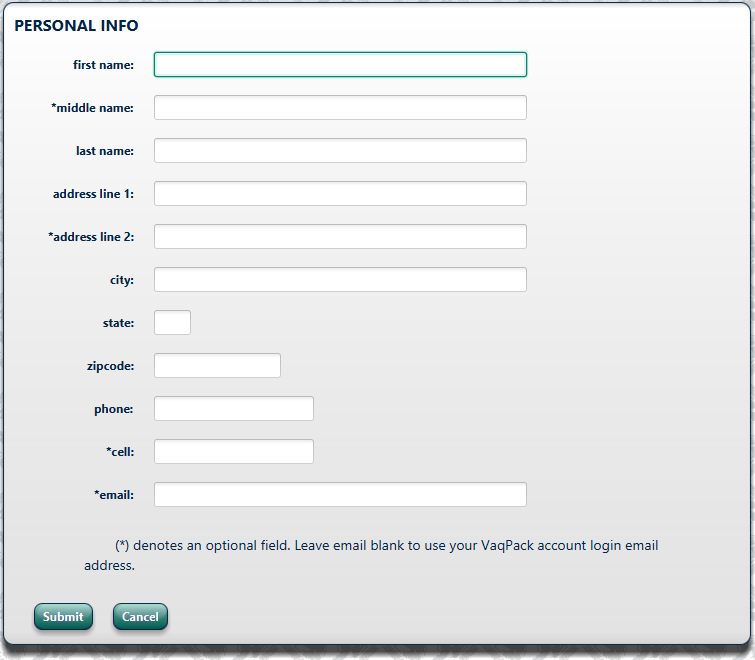
2. User clicks “Update Personal Information”

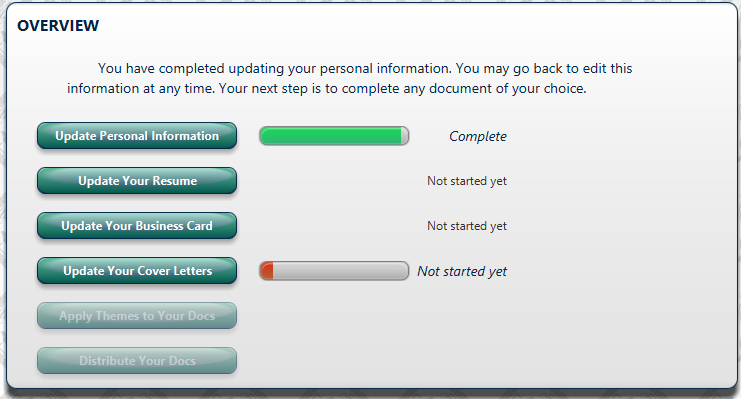
3. User successfully updates personal Information

4. User gain access to additional program features via enabled buttons

**Case 05**

****

****

****

**4.6**

**Use Case06: User Updating Personal Information – Missing Required Field**

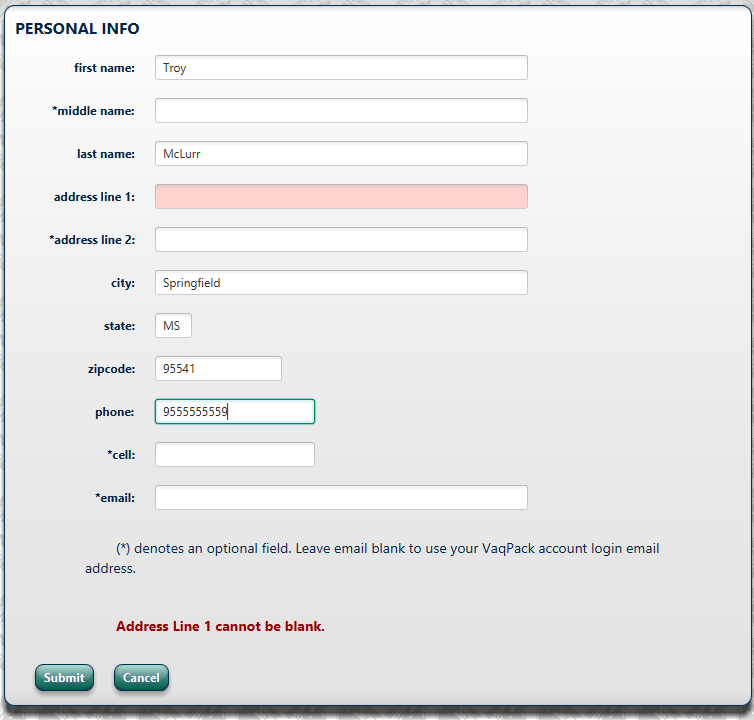
**Case Scenario**

1. User is in the Personal Info page

2. User leaves a required field blank

3. VaqPack highlights missing field, text displays which fields are missing

**Case 06**

****

**4.7**

**Use Case07: User Updating Personal Information – Incorrect Format**

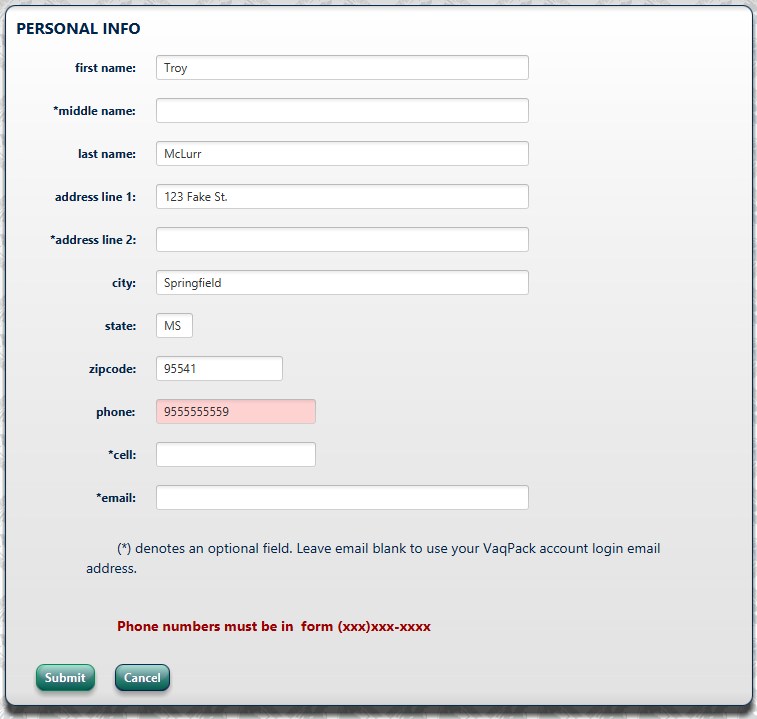
**Case Scenario:**

1. User is in the Personal Info page

2. User inputs a required field incorrectly

3. VaqPack highlights problem field and supplies correct format parameters

**Case 07**

****

**4.8**

**Use Case08: User Forgets Password**

VaqPack shall allow users to reset their passwords upon completing a verification process consisting of confirming a code sent to the email supplied upon creating the account.

**Case Scenario:**

1. User is in login page

2. User clicks on “forgot your password?”

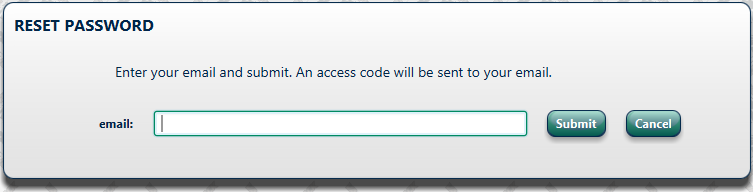
3. User successfully inputs email and clicks submit.

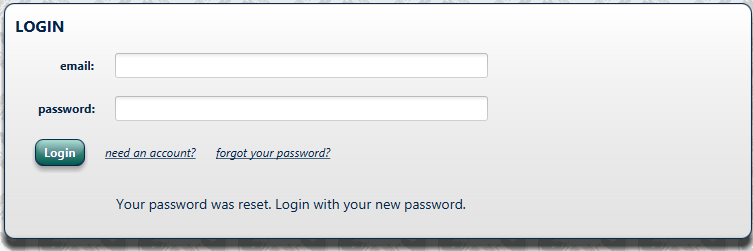
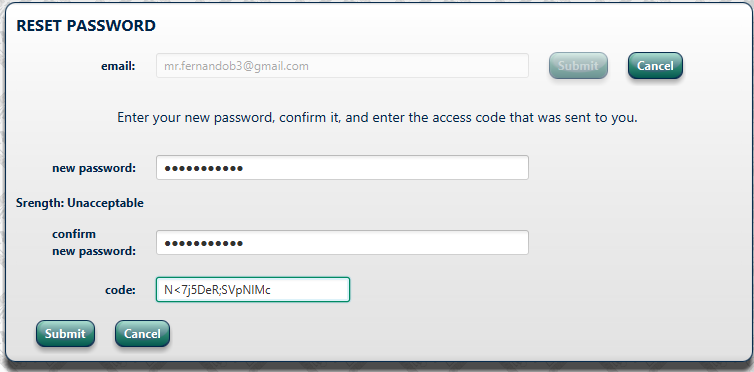
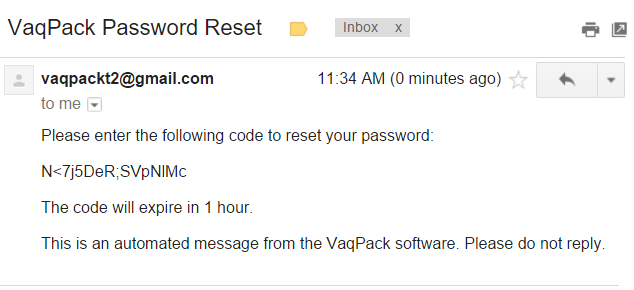
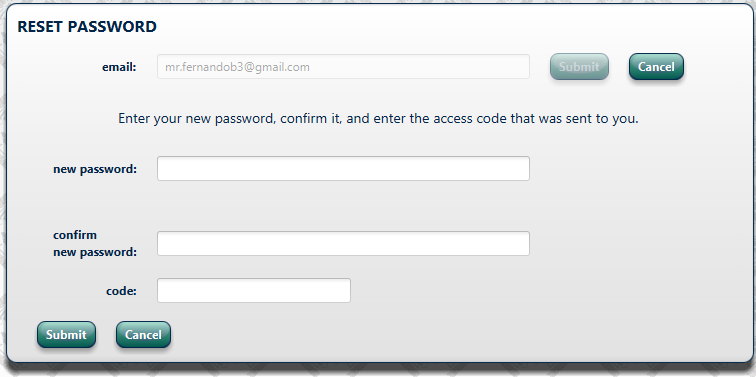
4. VaqPack sends email

5. User retrieves code from email

6. User updates password upon code verification

**Case 08 – User Forgets Password**

****

****

**4.9**

**Use Case09: Users New Password Not Strong Enough**

VaqPack requires a certain minimum strength/length before it accepts any users password choice.

**Case Scenario:**

1. User is in login page

2. User clicks on “forgot your password?”

3. User successfully inputs email and clicks submit

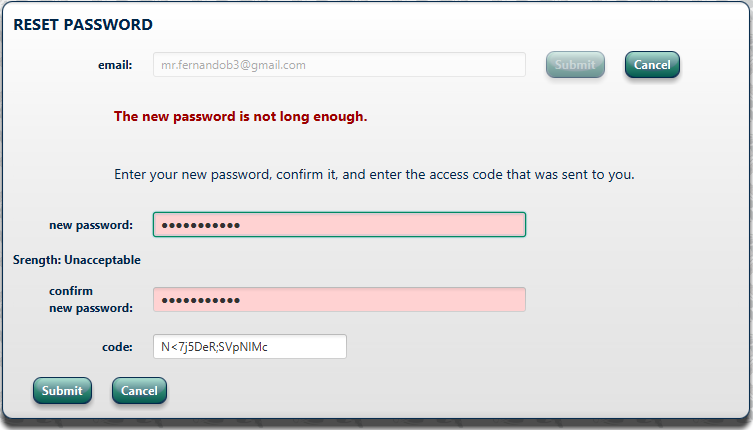
4. VaqPack sends email

5. User retrieves code from email

6. User updates password but doesn’t supply a strong/long enough password

4. VaqPack informs user to choose a stronger password

**Case 09 – User New Password Not Strong Enough**

****

**4.10**

**Use Case10: Users is inactive, automatic logout**

VaqPack shall have a built in logout timer to log out a user after 5 minutes of inactivity. A warning window will popup when there is 30 seconds left. Any mouse movement or keyboard key presses will reset the timer.

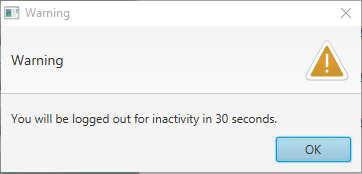
**Case Scenario:**

1. User is inactive for 4minutes 30 seconds

2. VaqPack prompts warning window

3. 30 seconds later the user will automatically be logged out

**Case 10 – User Is Inactive, automatic logout**

****

**A.** **Appendices**

**A.1 Appendix 1**

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

**Document Approval**

The following Software Design 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** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |