**Software Design Documentation**

# VaqPack (Database)

Version 1.0

November 30, 2016

## Lead Software Engineer: Elijah Lopez

**Project Team:**

Michelle Garcia  
Josue Rodriguez  
Juan Delgado

Prepared for

Software Engineering

University of Texas Rio Grande Valley Instructor: MK Quweider, Ph.D.

Fall 2016

## Table of Contents

1. **INTRODUCTION**……………………………………………………………..**1** 1.1 PURPOSE………………………………………………………………………….1
   1. SCOPE………………………………………………………………………....…..1
   2. DEFINITIONS, ACRONYMS, AND ABBREVIATIONS……………………….1

1. **REFERENCES**………………………………………………………………**3**

1. **DETAILED DESIGN**………………………………………………………..…4

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

3.1.01: DBOperations.java....................................................................................................4

3.1.02: User.java.....................................................................................................................5

3.1.03: Utilities.java...............................................................................................................6

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

* + 1. User Access Level...................................................................................................28
    2. Business Card Information....................................................................................29
    3. Business Card PDF.................................................................................................29
    4. Contact Information................................................................................................29
    5. Cover Letter Information........................................................................................29
    6. Cover Letter PDF.........................................................................................30
    7. Custom Theme.......................................................................................................30
    8. Registering User...................................................................................................30
    9. Reset Code...........................................................................................................30
    10. Resume Information............................................................................................31
    11. Resume HTML...................................................................................................31
    12. Resume PDF.........................................................................................................31
    13. User.......................................................................................................................31
    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 database portion of the VaqPack application. The intention of this document is to provide developers an insight into utilizing the database to its fullest ability to serve each and every user, 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 only the database portion of the VacPack applications. 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. |
| Database | A structured collection of data that can be efficiently and conveniently accessed. |
| 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. |
| 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/>

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

Java Runtime Environment - [http://www.oracle.com/technetwork/java/javase/downloads/jre8downloads-2133155.html](http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html)

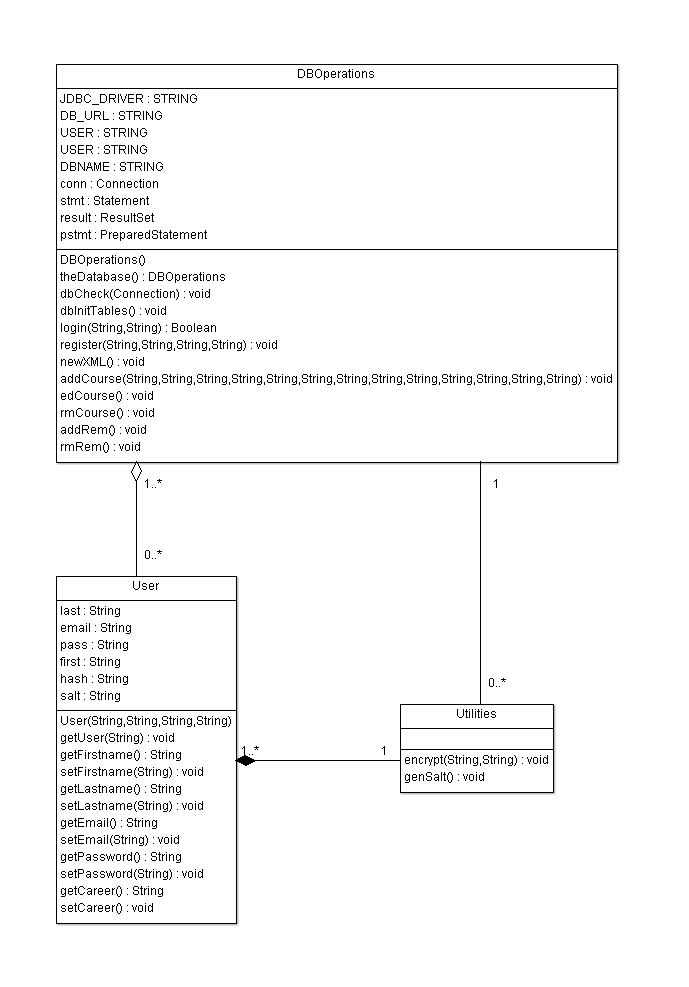
MySQL -<http://dev.mysql.com/downloads/mysql/>

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

JDBC - http://www.oracle.com/technetwork/java/javase/jdbc/index.html

### 3. Detailed Design

#### 3.1 Module Detailed Design

****

##### 

##### 3.1.01: DB.java

|  |  |
| --- | --- |
| **Class Name:** DB | |
| **Brief description:** The DB class is responsible for building and maintaining the database. | |
| **Attribute identifier and type** | **Attribute Description** |
| databaseSingleton | Creates only one instance of DB |
| String JDBC\_DRIVER; | This is a variable used to identify the JDBC driver. |
| String DB\_URL; | This is a variable used to identify the database URL. |
| String USER; | This is a variable used to identify the username. |
| String PASS; | This is a variable used to identify the user’s password |
| String DBNAME; | This is a variable used to identify the database name. |
| Connection conn; | This is a declaration of a Connection datatype to be used throughout the class. |
| Statement stmt; | This is a declaration of a Statement datatype to be used throughout the class. |
| ResultSet result; | This is a declaration of a ResultSet datatype to be used throughout the class. |
| PreparedStatement pstmt; | This is a declaration of a PreparedStatement datatype to be used throughout the class. |
|  |  |
| **Methods (operations)** | **Method Description** |
| Db() | This constructor will automatically connect to the database and check the server and check if the database exists. If not, it will create it. |
| theDatabase() | This is a method which returns the database object, which is a singleton. |
| dbCheck() | This is a method that accepts a parameter of Connection datatype and will check if the database exists. |
| dbInit() | This is a method that accepts a parameter of Connection datatype and initializes the database for the VaqPaq application. |
| closeConnection(Connection cn) | This is a method that takes a connection object to close it, and catches an SQL Exception |
| closeStatement(Statement st) | This is a method that closes and statement, or prepares one, and catches an SQL Exception |
| dbInitTables() | This is a method that initializes database tables for Users, Courses, Style, Reminders and User Courses. |
| login(User u) | This is a method that takes an object of User for logging a user in, and catches SQL Exception and Exception. |
| register(User u) | This is a method that takes an object of User for registering a new user, and catches SQL Exception. |
| newXml(String prefix, String courseNumber, String courseName) | This is a method that extracts information from new XML files, and creates the table statements to add them to the tables; and catches SQL Exceptions, FileNotFound Exception, and finalizes the connections and statements. |
| populateXMLFiles() | This is method that uses a InputStream and FileOutputStream to read and write XML files, and catches SQLException, FileNotFoundException, IOException and finalized connections and statements. |
| populateCSSFiles() | This is a method that retrieves the CSS files from the database. |
| addCourse(User user, Course courseToAdd) | This is a method that takes an object of and specific User and Course to add a Course to the User table. |
| edCourse() | This is a method that edits a course. |
| rmCourse() | This is a method that removes a course. |
| addRem() | This is a method that adds a reminder. |
| rmRem() | This is a method that removes a reminder. |

##### 

##### 3.1.02: User.java

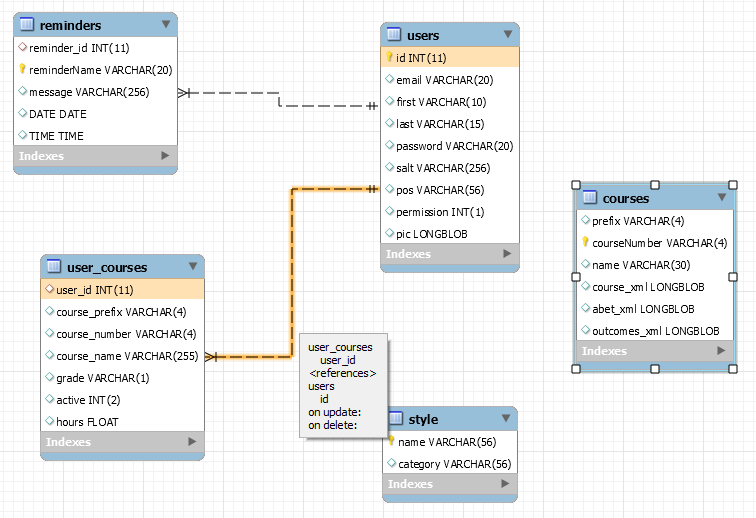
|  |  |
| --- | --- |
| **Class Name:** User | |
| **Brief description:** The User class is responsible for the creation of new users. | |
| **Attributes (fields)** | **Attribute Description** |
| No Attributes |  |
|  |  |
| **Methods (operations)** | **Method Description** |
| User(String, String, String, String) | This is a default constructor method. |
| getUser(String) | This is a method which accepts a String parameter getting Username info. |
| getFirstname() | This is a method for getting the User’s first name. |
| setFirstname(String) | This is a method that accepts a String parameter for setting the User’s first name. |
| getLastname() | This is a method for getting the User’s last name. |
| setLastname(String) | This is a method that accepts a String parameter for setting the User’s last name. |
| getEmail() | This is a method for getting the User’s email. |
| setEmail(String) | This is a method that accepts a String parameter for setting the User’s email. |
| getPassword() | This is a method for getting the User’s password. |
| setPassword(String) | This is a method that accepts a String parameter for setting the User’s password. |
| getCareer() | This is a method for getting the User’s career. |
| setCareer(String) | This is a method that accepts a String parameter for setting the User’s career. |

##### 3.1.03: Utilities.java

|  |  |
| --- | --- |
| **Class Name:** Utilities | |
| **Brief description:** The Utilities class is responsible for generating and managing the salt and encryption of passwords. | |
| **Attributes (fields)** | **Attribute Description** |
| No Attributes |  |
|  |  |
| **Methods (operations)** | **Method Description** |
| encrypt(String, String) | This method accepts two String parameters to encrypt a password. |
| genSalt() | This is a method which generates a salt to be added to a password. |

##### 

#### 3.2 Data Detailed Design



##### 3.2.01 Style

|  |  |
| --- | --- |
| **Table:** | **Style** |
| **Description** | Used to define a unique style for database entries |
| **Attributes** | **Description** |
| name | Name, primary key, bit, not null. |
| category | Style category, bit, default null. |

##### 3.2.01 Courses

|  |  |
| --- | --- |
| **Table:** | **Courses** |
| **Description** | Used to store all courses. |
| **Attributes** | **Description** |
| courseNumber | Course number, primary key, bit, not null. |
| prefix | Course prefix, bit, default null. |
| name | Course name, bit, not null. |
| course\_xml | Course XML, bit. |
| abet\_xml | XML, bit. |
| outcomes\_xml | Outcomes, bit. |

##### 3.2.01 User Courses

|  |  |
| --- | --- |
| **Table:** | **User\_Courses** |
| **Description** | Used to store and define courses the user has chosen. |
| **Attributes** | **Description** |
| user\_id | ID, integer, default null, foreign key. |
| course\_prefix | Prefix, bit, default null. |
| course\_number | Number of course, bit, default null. |
| course\_name | Name of course, bit, default null. |
| grade | Grade achieved, bit, null. |
| active | Status, integer, default null. |
| hours | Hours selected, float, default null. |

##### 3.2.02 User Information

|  |  |
| --- | --- |
| **Table:** | **Users** |
| **Description** | Used to store user information. |
| **Attributes** | **Description** |
| id | ID, integer, not null, auto increment. |
| email | User email, bit, default null. |
| first | User first name, bit, default null. |
| last | User last name, bit, default null. |
| password | User password, bit, default null. |
| salt | Password salt, bit, default null. |
| pos | Career path, bit, default null. |
| permission | Access-level, integer, default null. |
| pic | Picture, bit. |

##### 3.2.03 Reminders

|  |  |
| --- | --- |
| **Table:** | **Reminders** |
| **Description** | Used to store reminder information. |
| **Attributes** | **Description** |
| reminder\_id | ID of reminder, integer, default null. |
| reminderName | Primary key, name of reminder, bit, not null. |
| message | Message to be issued, bit, default null. |
| DATE | Current date, default null. |
| TIME | Current time, default null. |

### 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**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Elijah Lopez | 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

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |