Software Requirements Specification

Version 1.0

November 20, 2008

Disabilities Support Services Database Suite

Prepared by: Austyn Krutsinger and Iain Smith

Table of Contents

Table of Figures

1.0. Purpose

1.1. Introduction

This Software Requirements Specification provides a complete description of all the functions and specifications of the Southern Illinois University Carbondale-Disability Support Services Database Suite for iteration one and an overview of iteration two.

The expected audience of this document is the members of DSS, Michael Whitney, Tammy Keen, Jerimiah Womick, Rita VanPelt, including the faculty who will supervise development, Dr. Kenny Fong. It will also serve as a reference for future programmers and users of this database application.

1.2. Scope

This document contains the description of the functionality of the *DSS Database Suite* project for the first iteration of the program development. It consists of use cases, functional requirements, and nonfunctional requirements, which when combined together form a complete description of the first iteration in the software development.

1.3. Glossary

Term	**Definition**
Access Control List	A list that defines what privileges certain users get for specified applications.
Basic Student Information	Dawg Tag Number, Social Security Number, Name, Local/Home Address, [cell] Phone Number, Birth Date, Gender, Marital Status, Ethnicity, Current Major and Class, Semesters of Service, Veteran Status and Recent War Participation (Iraq/Afghanistan), Disability Codes and Needs (text conversion, DHS Aff), Assigned Case Worker, and Wheelchair Usage. Discussed further in section 3.4.
Client	Refers to the program used to input the student's data and save it to the database
Database	Centralized collection of information.
Dawg Tag	Unique number associated to every student and instructor enrolled at SIU.
DSS	Disabilities Support Services.

LAN	Local Area Network. A collection of computers connected to each other allowing then to share information and resources. These computers only have access to each other.
Server	A computer connected to a LAN and accessed by multiple clients and provides a service to each computer connected. In this case it provides database access.

1.4. References

-Vision and Scope v1.1

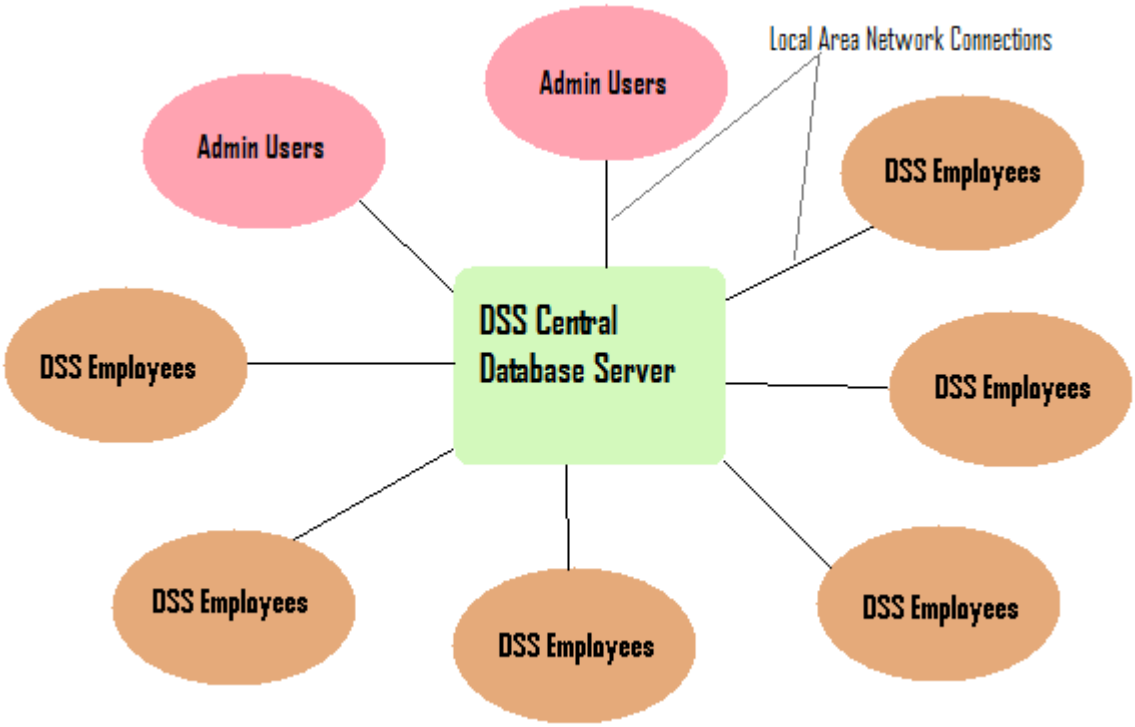
1.5. Document overview

The remainder of this document is in two chapters. The first providing a full description of the project for the DSS employee's and staff. It lists all the functions performed by the system for the first iteration of the project. The final chapter concerns details of each of the system functions and actions in full for the software developers' assistance. These two sections are cross-referenced by topic to increase understanding by both groups involved.

2.0. Overall description

The *DSS Database Suite* is a project designed to provide electronic filing for all records that the DSS department keeps on their students, past and present. This database system will allow administrative users to add, delete, and edit all information in the database. There will be another group of users, namely student employees, who will only have read privileges to certain information of the student with disibilities.

2.1. System environment



-Admin Users – Have permission to read from and write to the database

-Employees – Have strictly read permissions to the database.

Figure 1 System Design

The DSS Database will be stored on a central server in the DSS building. The server will be connected to all the employee's computers via a local area network. Since the database will be in a private network, it will not be able to be accessed from the outside,

providing security for the information stored in the database. However in the second iteration there will be access from the internet because students will be able to access some of their records from outside computers. When this comes into play, database access will be strictly controlled by access control lists.

There will be three types of users that will have access to the database: Administrative Users, DSS Employees, and in future iterations Students. The Administrative Users will have full access to the database. They will be the only ones who will be allowed to save information to any part of the database as well as access any information stored concerning a student. DSS Employees refer to anybody who does not have full access to all aspects of the database. Each employee will have specific privileges set to allow them access to only certain parts of the database. This access will be at the discretion of the admin users. In future iterations students will be allowed to add a test date to their records.

2.2. Functional requirements definitions

Functional Requirements are those that refer to the functionality of the system, i.e., what services it will provide to the user. Nonfunctional (supplementary) requirements pertain to other information needed to produce the correct system and are detailed separately.

2.3. Use cases

The system consists of a database of student/client entries with multiple sections. The first section is the Student Information section. This section contains the student's Basic Student Information and is the primary source of all of a student’s personal information. Not all data fields in the Student Information section need to be filled in. A student's Dawg Tag Number will be used as the database key. The rest of the sections will be linked to this section through these keys.

The second section is the Equipment Usage section. This section contains a record of all student equipment usage. These records consist of an equipment name, SIUC equipment number, check-out date, due date, and check-in date.

Use Cases - Iteration 1:

2.3.1. Use Case: Create New Student Entry

Name	**UC-1: Create New Student Entry**
Summary	A new student entry is created containing all student data provided and then added to the database.
Rationale	It is important to be able to add a new student to the database at any time a new one with sufficient need enrolls at the University.
Users	Admin.
Preconditions	The database is loaded.
Basic Course of Events	1. User indicates that the software is to create a new student entry. 2. Software responds by requesting all data fields of Basic Student Information. 3. The user inputs all data fields of Basic Student Information. 4. Software creates new data entry and appends it to the database.
Alternative Paths	1. In Step 4, an entry in the database already exists with the given key. In this case, the software returns that entry of the database to the user. 2. The user decides to abort the creation of a new student entry at any time prior to step 4. In this case, the software returns to the precondition state.
Postconditions	The student entry is on screen and the database is appended and ready for further use.

2.3.2. Use Case: Edit Student Entry

Name	**UC-2: Edit Student Entry**
Summary	An existing student's entry is modified in one or more fields to match new data entered.
Rationale	Some information such as address and phone number are subject to change over time. It is important to keep this information up-to-date.
Users	Admin.
Preconditions	An existing student entry is open.
Basic Course of Events	1. User indicates that software is to display data fields for editing. 2. The software responds by displaying data fields in an editable format.

	3. The user indicates that one or more fields are to be replaced and specifies values for these fields. 4. The user sends the edit request to the software. 5. The database updates the specified data fields.
Alternative Paths	1. The user decides to abort the editing of data entries at any time prior to step 4. In this case, the software returns to the precondition state.
Postconditions	Same as precondition, except one or more fields have been updated.

2.3.3. Use Case: Archive Student Entry

Name	**UC-3: Archive Student Entry**
Summary	The student entry is moved to a secondary server that stores inactive entries.
Rationale	Student entries are only heavily used during times when the student is actively enrolled at the University. Archiving of inactive students cuts down on search and retrieval times for enrolled student entries while keeping older records available for occasional use or re-enrollment.
Users	Admin.
Preconditions	The database is open.
Basic Course of Events	1. User indicates that the software is to archive a student entry. 2. The software responds by requesting the key of the entry to be archived. 3. The user inputs the key. 4. The software archives the entry.
Alternative Paths	1. In Step 2, the user indicates that the software is to archive all students no longer enrolled. In this case, the software queries enrollment and archives all students no longer enrolled. 2. The user decides to abort the student entry archiving prior to step 3. In this case, the software returns to the precondition state.
Postconditions	The student entry is stored in a secondary database and primary database is open with archived entry removed.

2.3.4. Use Case: Query Student Entries

Name	**UC-4: Query Student Entries**
Summary	Returns to user all student entries that meet a specific user enter data field criteria.
Rationale	Sometimes the user is not looking for a specific student entry, but rather all students who meet certain criteria. This can be for statistical purposes or other reasons. This serves that functionality while removing the tedium of doing it manually throughout the expected average enrollment

	of 2.5k-3k student entries.
Users	Admin and DSS Employees.
Preconditions	The database is loaded.
Basic Course of Events	1. User indicates that the software is to query its student entries. 2. The software responds by requesting the data field and value to be queried. 3. The user inputs a data field and value to query. 4. The software queries the database.
Alternative Paths	1. The user decides to abort the query request prior to step 4. In this case, the software returns to the precondition state.
Postconditions	A summary of all students who meet the criteria is returned to the user.

Use Cases - Iteration 2:

2.3.5. Use Case: Check-out Equipment

Name	**UC-5: Check-out Equipment**
Summary	Stores information pertaining to the student checkout of University equipment.
Rationale	It is important that all University equipment is accounted for and charged for if not returned in due time.
Users	Admin
Preconditions	An existing student entry is open.
Basic Course of Events	1. User indicates that software is to check-out equipment. 2. The software responds by asking for equipment information (Name and University ID Number). 3. The user inputs equipment information. 4. The software checks out equipment on a user defined date and due a predefined amount of time later.
Alternative Paths	1. In Step 3, user inputs check-out date. In this case, the software uses user specified check-out date and sets due date accordingly. 2. The user decides to abort the query request prior to step 4. In this case, the software returns to the precondition state.
Postconditions	An existing student entry is open with one or more pieces of equipment checked out.

2.3.6. Use Case: Check-in Equipment

Name	**UC-6: Check-in Equipment**
Summary	Records the return of checked-out University equipment.

Rationale	It is important that all University equipment is accounted for and charged for if not returned in due time.
Users	Admin
Preconditions	An existing student entry is open that has equipment checked-out.
Basic Course of Events	1. User indicates that software is to check-in equipment. 2. The software responds by returning checked-out equipment information. 3. The user specifies equipment to check-in. 4. The software checks in equipment on current date.
Alternative Paths	1. In step 3, user specifies equipment check-in date. In this case, the software uses this date for the check-in date. 2. The user decides to abort the query request prior to step 4. In this case, the software returns to the precondition state.
Postconditions	An existing student entry is open with one or more pieces of equipment returned.

3.0. Requirement specifications

3.1. External interface specifications

None

3.2. Functional Requirements

3.2.1. Database Access

Name:	**FR-1: Database Access**
Summary	The client user must be able to connect to the database to allow for reading and writing student information as well as querying the inputted data.
Rationale	The user must be able to access the database for this application to be relevant.
Requirements	When the user wants to connect to the database, the software must give the user a list of the available databases to connect to. The databases should have plain English names that the user can easily recognize that correspond to which database the user wants to save/load data to/from. If the client cannot connect to the database, the application will display a message to the user informing them of the problem that occurred.
Reference	UC-1: Create New Student Entry, UC-2: Edit Student Entry, UC-3: Archive Student Entry, UC-4: Query Student Entries.

3.2.2. Save Student Information

Name:	**FR-2: Save Student Information**
Summary	The program must save the information inputted by the user into the database

	with that particular record indexed by the student's Dawg Tag.
Rationale	The user wants the information that corresponds to a particular student to be saved in a record that belongs only to that student. Then the user will want to be able to retrieve the student's information by only supplying the student's Dawg Tag. That is why the student's information must be indexed by their Dawg Tag.
Requirements	The main program has all the forms needed for the user to fill out with the information for each student. Once all the information is inputted and the user wants to save the information, the user will select Save from a menu. If the program is not connected to the database, it will prompt the user informing them that there is no connection and offer an option to connect. Once a connection is established the program will continue and save all the inputted data into a record in the database corresponding to that particular student.
Reference	UC-1: Create New Student Entry, UC-2: Edit Student Information, UC-5: Check-out Equipment, UC-6: Check-in Equipment.

3.2.3. Search for Student Information

Name:	**FR-3: Search for Student Information**
Summary	Retrieving specific information that the user wants to find about a certain student or group of students.
Rationale	If the user wants to find all the students with a certain criteria (e.g. uses a wheelchair) they will be able to search the database for every student who has a wheelchair. To extend that, the user will also be able to combine fields to query. For example, if you want to find all students males who used a wheelchair last semester.
Requirements	The program must supply enough search criteria to the user to allow them to search for any possibility they want. The criteria will be reflected from the type of document they are searching through. If the user is searching basic student information then they should only be shown search options for basic student information.
Reference	UC-4:Query Student Entries.

3.3. Non-functional requirements

There are requirements that are not functional in nature. Specifically, these are the constraints the system must work within.

3.3.1. Intuitive User Interface

Name:	**NF-1: Intuitive User Interface**
Summary	Design the user interface so that it is organized and intuitive for the user to use
Rationale	The easier it is for the user to use the program the more work can be done. More work is not only done because they spend less time searching for the feature they want to use, but also to ensure the user does not get frustrated by not being able to easily find what they are looking for.

Requirements	Program user interface should look clean and organized and allow the user to easily know where the features are that they want to access.
Reference	UC-1: Create New Student Entry, UC-2: Edit Student Entry, FR-3: Search for Student Information.

3.3.2. Program Package

Name:	**NF-2: Program Package**
Summary	The user should not have to go searching for any needed programs to make the *DSS Database Suite* work properly.
Rationale	If the user is hassled with extra work of figuring out why the program does not work, they will not like the work done. Anything needed by the program should be included in an encompassing package and given to the user(s).
Requirements	The user should be given a package that includes all files necessary to make the *DSS Database Suite* work properly. This includes: 1) The main executable 2) The MySQL library for connection to the database.
Reference	None

3.4. Detailed non-functional requirements

The main form in the program will provide a way to input the information listed above. This information will be used by DSS for their own records and to assist in deciding what type of services will benefit the student the most. Along with this list of basic information, there will be other documentation and forms that, if needed, will be available for each student.

A basic guide for the development environment and hardware requirements for the finished product:

Hardware:
  -Client: Employee's Workstation
  -Server: Department Server
Operation System:
  -Client: Window XP/Vista with .NET Framework and MySQL extension
  -Server: Windows Server 2000/2003/2008 with .NET Framework and MySQL
Network Connection: LAN
Code Standard: The database will be MySQL

	The client program will by coded in C# using Microsoft Visual Studio 2005
	The connection to the DSS Database will be done internally with C# using the MySQL ODBC extension
Performance:	The system should save and load the records in the appropriate table of the DSS Database 100% of the time

3.5. System Evolution

In the future, this system will be updated with more features documented in the Vision and Scope document. The development will be done in iterations. During each iteration specific features will be added to the project. The order of the features being added is based off of relative necessity to the DSS staff for getting the students into the program and providing the services they need. The major feature added in the next iteration will be the addition of DSS forms. These forms are:

1. Supplemental Information Sheet,
2. Medical Documentation Retrieval Form,
3. Medical Documentation Release Form,
4. Alternate Format Request Form,
5. Text Conversion Services Agreement,
6. Accommodations Agreement,
7. Equipment Load Records, and
8. Test Scheduling.

These forms will be stored electronically instead of the hard copies that the office currently uses. The purpose for the electronic copies is to allow the DSS employees to pull up one of these documents from the for the *DSS Database Suite* to automatically fill in any information fields it can with the information stored in that particular student's records.